Arduino Programming Environment and Library

If you have not already, please refer to NXShield User Guide for download instructions of Arduino Programming Environment and Library, at following url:
http://www.openelectrons.com/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=7

Structure of Arduino Program for NXShield

In the example program below, a NXT Touch sensor and Light sensor is attached to NXShield. Each time the touch sensor is pressed, the active/passive mode of light sensor is toggled.

The program also takes readings from light sensor continuously.

You can find this program in the library distribution for NXShield.

Refer to comments below for explanations of parts of the program.

```
#include <Wire.h>
#include <NXShield.h>

#include <NXTTouch.h>
#include <NXTLight.h>

//
// declare the NXShield(s) attached to your Arduino.
//
NXShield    nxshield;

//
// declare analog devices attached to nxshields.
//
NXTTouch    touch1;
NXTLight    light1;



void setup()
{

    Serial.begin(115200);       // start serial for output
    delay(500);                 // wait, allowing time to
                                // activate the serial monitor
    Serial.println ("Initializing the devices ...");
    //
    // Initialize the protocol for NXShield
    // It is best to use Hardware I2C (unless you want to use Ultrasonic).
    //
```

Comment [DGP1]: These two include statements are required.

Comment [DGP2]: Include all header files for the devices you will be attaching to your NXShield

Comment [DGP3]: This is the C++ Object variable for the NXShield.

Comment [DGP4]: Declare the C++ Object variables for the NXT Light sensor and NXT Touch sensor used in this program.

Comment [DGP5]: Usual setup function in Arduino sketch. This function is called once when your program begins to run. Initialize your devices and protocols in this function.

Comment [DGP6]: It's a good idea to use Serial device for printing your messages.

```
    nxshield.init( SH_HardwareI2C );

    //
    // Wait until user presses GO button to continue the program
    //
    Serial.println ("Press GO button to continue");
    nxshield.waitForButtonPress(BTN_GO);

    //
    // initialize the analog sensors.
    // NXT light sensor is not supported on BAS1.
    // connect a NXT light sensor to BAS2,
    // connect a touch sensor to BAS1
    //
    touch1.init( &nxshield, SH_BAS1 );
    light1.init( &nxshield, SH_BAS2 );
}

bool lastTouch, touchPressed;

void loop()
{
    char          str[256];
    int lightReading;

    Serial.println("Into loop ----------------------------------");

    touchPressed = touch1.isPressed();
    sprintf (str, "touch1: is pressed : %s", touchPressed?"true":"false");
    Serial.println(str);

    if ( touchPressed != lastTouch ) {
        if ( touchPressed == true ) {
            Serial.println( "Changing light sensor to reflected light mode" );
            light1.setActive();
        } else {
            Serial.println( "Changing light sensor to ambient light mode" );
            light1.setPassive();
        }
        lastTouch = touchPressed;
    }
    lightReading = light1.readRaw();
    sprintf (str, "Light sensor Reading: %d", lightReading);
    Serial.println (str);

    delay (500);
}
```

Comment [DGP7]: Initialize the NXShield.

Comment [DGP8]: Generally it is a good idea to wait until user presses GO button.

Comment [DGP9]: Initialize the sensors you have attached to NXShield.

Comment [DGP10]: Usual loop function in Arduino sketch. This function is repeated continuously. This is where your main actions should be.

Comment [DGP11]: Look if the touch sensor is pressed.

Comment [DGP12]: The status of touch sensor is printed in the Serial window.

Comment [DGP13]: Set the light sensor in reflected light mode (active mode).

Comment [DGP14]: Set the light sensor in ambient light mode (passive mode).

Comment [DGP15]: Take a reading from the light sensor

Comment [DGP16]: Wait for half second before it loops again.

Another Example of Arduino Program for NXShield

In this example program Two NXT Ultrasonic sensors are attached to NXShield. The program reads the Sensor information and displays it in Serial window.

```
#include <Wire.h>
#include <NXShield.h>

#include <NXTUS.h>

//
// declare the NXShield(s) attached to your Arduino.
//
NXShield    nxshield;

//
// declare the i2c devices used on NXShield(s).
//
NXTUS       sonar1;
NXTUS       sonar2;

void setup()
{
  char str[256];

  Serial.begin(115200);  // start serial for output
  delay(500); // wait, allowing time to activate the serial monitor

  Serial.println (__FILE__);
  Serial.println ("Initializing the devices ...");
  //
  // Initialize the protocol for NXShield
  // It is best to use Hardware I2C (unless you want to use Ultrasonic).
  //
  nxshield.init( SH_SoftwareI2C );

  //
  // Wait until user presses GO button to continue the program
  //
  Serial.println ("Press GO button to continue");
  nxshield.waitForButtonPress(BTN_GO);

  //
  // Initialize the i2c sensors.
  //
  sonar1.init( &nxshield, SH_BBS2 );
  sonar2.init( &nxshield, SH_BAS2 );

}

void loop()
{
  char aa[80];
  char str[256];
  int  bb_us;

  strcpy(aa, sonar1.getFirmwareVersion() );
  sprintf (str, "sonar1: FirmwareVersion: %s", aa);
```

Comment [DGP17]: Include header file for sensor used. (This is the only sensor used in this example).

Comment [DGP18]: Declare the variables for two Ultrasonic sensors.

Comment [DGP19]: Since we are using Ultrasonic sensors, use SoftwareI2C protocol. (Ultrasonic sensors don't work with hardware i2c protocol).

Comment [DGP20]: The two ultrasonic sensors are attached to BBS2 and BAS2 respectively.

Comment [DGP21]: Get firmware version from sensor.

Comment [DGP22]: Format the version string for printing.

```
Serial.println(str);

strcpy(aa, sonar1.getDeviceID() );
sprintf (str, "sonar1: DeviceID: %s", aa);
Serial.println(str);

strcpy(aa, sonar1.getVendorID() );
sprintf (str, "sonar1: VendorID: %s", aa);
Serial.println(str);

bb_us = sonar1.getDist();
sprintf (str, "sonar1: Obstacle at: %d mm", bb_us );
Serial.println(str);

strcpy(aa, sonar2.getFirmwareVersion() );
sprintf (str, "sonar2: FirmwareVersion: %s", aa);
Serial.println(str);
strcpy(aa, sonar2.getDeviceID() );
sprintf (str, "sonar2: DeviceID: %s", aa);
Serial.println(str);
strcpy(aa, sonar2.getVendorID() );
sprintf (str, "sonar2: VendorID: %s", aa);
Serial.println(str);
bb_us = sonar2.getDist();
sprintf (str, "sonar2: Obstacle at: %d mm", bb_us );
Serial.println(str);

Serial.println( "----------" );
delay (1500);
}
```

Comment [DGP23]: Print the version string in serial window.

Comment [DGP24]: Get device id from sensor.

Comment [DGP25]: Get manufacturer information from sensor.

Comment [DGP26]: Get distance to obstacle from sensor.