How to use HiTechnic touch sensor multiplexer in NXT++? Mindstorms NXT 2.0

f you are advanced user of NXT++, and want to add code to your existing project just go to point (II).

(I) Quick start in Visual Studio 2010 for begginers

1) install microsoft visual studio 2010 with visual c++,
2) install drivers of NXT_32 or NXT_64 depending on your system,
3) if you wish to use bluetooth connection with NXT brick, install bluetooth drivers (in case of problem with bluetooth drivers, uninstall drivers provided by computer dealer and use standard windows drivers,
4) download library http://wmii.uwm.edu.pl/~artem/nxtpp0-6-3.zip (the library include color sensor support, sensor multiplexer (SMUX) support, and touch sensor multiplexer (TMUX) support, the original one without our changes is available in http://sourceforge.net/projects/nxtpp/),
5) extract file nxtpp0-6-3.zip, and move the folder nxtpp0-6-3 directly to drive C:\ (because the solution is configured on this path, if you have different path just change links in your project,
6) open file: C:\nxtpp0-6-3\FINAL\examplesTMUX\vs10\re4.sln
if you have different path to re4.sln file, open solution re4.sln from your path, and in re4 properties, particularly in:
- configuration properties/C/C++/general Additional Include Directories/
and
- configuration properties/Linker/General/Additional Library Direcories/
change the links on yours,
7) plug in your sensors to proper sensor multiplexer subports,
8) turn on the power supply of sensor multiplexer, and then turn on NXT brick,
7) connect your NXT brick via cable or bluetooth and run the solution. In our example from file C:\nxtpp0-6-3\FINAL\examplesTMUX\vs10\re4\re4\main.cpp we have fixed Solution Configurations as Release and Solution Platforms as Win32. And we made assumption that sensor multiplexer is connected to port IN_1, change if you use different port.

(II) Usage for advanced users:

Paste the following code into the file NXT++.h:

```
/*! Sets a touch sensor multiplexer (TMUX) in a specified port,
\param port can be the numbers 0-3 or IN_1, IN_2, IN_3, or IN_4.*/
void SetTMUX(Comm::NXTComm* comm, int port);

/*! Retrieves the value of a touch sensor connected to specified subport of touch sensor
multiplexer, return 1-pressed touch sensor, return 0-sensor not pressed, or missing,
\param port The port in which the sensor multiplexer is connected, can be the numbers 0-3 or
IN_1, IN_2, IN_3, or IN_4.
\param subport - is the number of touch sensor multiplexer subport, can by 1, 2, 3 or 4*/
int GetTMUXvalue(Comm::NXTComm* comm, int port, int subport);
```

Paste the following code into the file nxt++.cpp:

```
void NXT::Sensor::SetTMUX(Comm::NXTComm* comm, int port)
{
ViUInt8 directCommandBuffer[] = { 0x05, port, 0x07, 0x80};
comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
}

int NXT::Sensor::GetTMUXvalue(Comm::NXTComm* comm, int port,int subport)
{
 int state=0;
 int odczyt;
 ViUInt8 directCommandBuffer[] = { 0x07, port };
 ViUInt8 responseBuffer[] = { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 };

 comm->SendDirectCommand( true, reinterpret_cast< ViByte* >( directCommandBuffer ),  sizeof(
directCommandBuffer ), reinterpret_cast< ViByte* >( responseBuffer ),
sizeof( responseBuffer ));

 odczyt=responseBuffer[12]*256+responseBuffer[11];

 if(subport==1)
  {
  if(odczyt==1 || odczyt==2 || odczyt==8 || odczyt==13 || odczyt==14 || odczyt==23 ||
odczyt==24 || odczyt==18 || odczyt==19 || odczyt==27 || odczyt==35)
   {
    state=1;
   }
  }
 if(subport==2)
  {
   if(odczyt==4 || odczyt==5 || odczyt==8 || odczyt==16 || odczyt==25 || odczyt==26 ||
odczyt==18 || odczyt==19 || odczyt==27 || odczyt==33 || odczyt==35)
    {
     state=1;
    }
   }
 if(subport==3)
  {
   if(odczyt==11 || odczyt==13 || odczyt==14 || odczyt==16 || odczyt==30 || odczyt==18 ||
odczyt==19 || odczyt==33 || odczyt==35)
    {
     state=1;
    }
   }
 if(subport==4)
  {
   if(odczyt==21 || odczyt==23 || odczyt==24 || odczyt==25 || odczyt==26 || odczyt==30 ||
odczyt==27 || odczyt==33 || odczyt==35)
   {
    state=1;
   }
  }
      return state;
}
```

If you use Visual Studio, open and rebuild solution NXT++.sln

And use nxt sensor in main.cpp of your project:

Exemplary usage:

```
/*the touch sensor multiplexer (TMUX) specified in port IN_1*/

// Tells the NXT brick to set touch sensor multiplexer in port IN_1,
```

```
this operation should be done only once, before reading from sensor multiplexer,
NXT::Sensor::SetTMUX(&comm, IN_1);

//how to read status from sensors connected to touch sensor multiplexer?
//function GetTMUXvalue return the status of chosen multiplexer subport,
//return value 1- if touch sensor connected to chosen subport is pressed, return 0- if not
pressed, or missing,
int status=NXT::Sensor::GetTMUXvalue(&comm, IN1, subport);

//parameter subport could be:
//'1' - setting to read from subport 1
//'2' - setting to read from subport 2
//'3' - setting to read from subport 3
//'4' - setting to read from subport 4

//If we want to check status of touch sensor connected to subport 4 of touch sensor
multiplexer connected to port IN_1 of NXT brick, we coud use the following code,
status=NXT::Sensor::GetTMUXvalue(&comm, IN1, 4);
```

To see demonstration of touch sensor multiplexer, you can put the following code to your solution main.cpp file:

```
#include "NXT++.h"
#include <conio.h>
Comm::NXTComm comm;

int main()
{
 std::cout << "\naquiring signal ... ";
 int infinity_loop_pointer=1;

 if(NXT::OpenBT(&comm))/*initialize the NXT via cable or bluetooth and continue if it
succeeds*/
#include "NXT++.h"
#include <conio.h>
Comm::NXTComm comm;

int main()
{
 std::cout << "\naquiring signal ... ";
 int infinity_loop_pointer=1;
 if(NXT::OpenBT(&comm))/*initialize the NXT via cable or bluetooth and continue if it
succeeds*/
  //if(NXT::Open(&comm))
  {
   std::cout<< "signal found";
   std::cout<<"\nbattery Level = "<<NXT::BatteryLevel(&comm)/100<<" percent";
   std::cout<<"\navailable flash memory = "<<NXT::GetAvailableFlash(&comm);

  // In this example touch sensor multiplexer is connected to port IN_1
   NXT::Sensor::SetTMUX(&comm,IN_1);

   char decyzja;
   std::cout<<"\nPress S to get sensor values from sensor multiplexer ports";
   std::cout<<"\nPress K to stop the program";

   do
    {//beggining of main inifinity loop

     if(kbhit()==true)
      {
       decyzja=getch();

     if(decyzja=='K' || decyzja=='k')
      {
       break;
```

```
      }
     if(decyzja=='S' || decyzja=='s')
      {
  // In this example sensor multiplexer is connected to port IN_1
  /*NXT::Sensor::GetTMUXvalue(&comm, port, subport);
  \param port - the port of sensor multiplexer connection with NXT brick IN_1, IN_2, IN_3
       or IN_4,
   \param subport - multiplexer subport number - 1,2,3 or 4,*/
  // function GetTMUXvalue return values: 1-pressed touch sensor, 0-not pressed, or missing,
     std::cout<<"\n\nreading from subport 1 = "<<NXT::Sensor::GetTMUXvalue(&comm,IN_1,1);
     std::cout<<"\nreading from subport 2 = "<<NXT::Sensor::GetTMUXvalue(&comm,IN_1,2);
     std::cout<<"\nreading from subport 3 = "<<NXT::Sensor::GetTMUXvalue(&comm,IN_1,3);
     std::cout<<"\nreading from subport 4 = "<<NXT::Sensor::GetTMUXvalue(&comm,IN_1,4);
                 }
            continue;
               }
             }// the end of main inifinity loop
             while(infinity_loop_pointer!=2);

             NXT::StopProgram(&comm);
      }
      NXT::Close(&comm); //close the NXT
      return 0;
}


//the touch sensor multiplexer (TMUX) support for NXT++ developed by artem
http://wmii.uwm.edu.pl/~artem
//the original library without our changes is available in:
//http://sourceforge.net/projects/nxtpp/
```