Bricx Command Center

Useful Buttons

Loading Programs
- Use the Open button or File-menu option to open an existing program – these have a "*.nqc" file extension
- Sample programs have been provided for you

Editing Programs
- When you load a program it will be displayed in an editor window

Compiling Programs
- To compile a program click the Compile button (or use the Compile menu, or just press F5)
- If successful, no errors will be reported
- If there is an error in your program, it will be reported

Running Programs
- Once successfully compiled, a program is downloaded to the RCX using the Download button (Download option in the Compile menu, or F6)
- The RCX is capable of storing 5 different programs; use the drop-down box next to the download button to select which program "slot" you wish to download the program into
- Once a program has been downloaded to the RCX, you can run it by clicking the Run button on the tool-bar
- Alternatively run the program by clicking the green run button on the RCX itself
Getting Help

- For standard RCX functions (controlling motors, setting sensors, dealing with timers, etc.), there's a handy Templates window – this can be found via the View menu if not visible.
- This also gives a pretty concise overview of what the RCX is capable of.

Documentation

- Documentation on NQC is available through the on-line help.
- Click the obvious icon, or use the Help menu to get at the on-line help.
- You can browse the help documentation using the index.

A Sample Program

```
task main()
{
OnFwd(OUT_A+OUT_C);
OnFwd(OUT_C);
Wall(400);
OnRev(OUT_A+OUT_C);
Wall(400);
Off(OUT_A+OUT_C);
}
```

This program
- turns outputs A and C (motors) on (forwards)
- drives forwards for 4 seconds (400 clicks)
- puts outputs A and C into reverse
- drives backwards for 4 seconds
- then stops

```
1_simple.nqc
```

Another Sample Program

```
task main()
{
OnFwd(OUT_A+OUT_C);
Wait(100);
turn_around();
Wait(200);
turn_around();
Wait(100);
turn_around();
Off(OUT_A+OUT_C);
}
```

This program makes use of a subroutine named turn_around

```
void turn_around()
{
OnRev(OUT_C);
Wait(400);
OnFwd(OUT_A+OUT_C);
}
```

This shows how subroutine turn_around is defined

```
6_inline.nqc
```

Yet Another Sample Program

```
#define THRESHOLD 40
```

This defines the value of a constant

```
task main()
{
SetSensor(SENSOR_2_SENSORT_LIGHT);
OnFwd(OUT_A+OUT_C);
while (true)
{
if (SENSOR_2 > THRESHOLD)
{
OnRev(OUT_C);
Wait(10);
until (SENSOR_2 <= THRESHOLD);
OnFwd(OUT_A+OUT_C);
}
}
```

This defines a sensor

This creates an infinite loop

If sensor is > than threshold, reverse motor C, wait 0.1 seconds then check if sensor is below threshold; if it is go forwards

```
5_light.nqc
```
A Further Sample Program

/* Define inputs and outputs */
/* These should match the robot's physical configuration */

#define BUMP_LEFT SENSOR_3
#define BUMP_RIGHT SENSOR_1
#define DRIVE_LEFT_OUT_C
#define DRIVE_RIGHT_OUT_A

task main ()
{
    /* Initialise sensors and make sure the motors are off */
    SetSensor(BUMP_LEFT, SENSOR_TOUCH);
    SetSensor(BUMP_RIGHT, SENSOR_TOUCH);
    OR(DRIVE_LEFT + DRIVE_RIGHT);
    /* Set motor power */
    SetPower(DRIVE_LEFT, 2);
    SetPower(DRIVE_RIGHT, 2);
}

A Further Sample Program 2

/* Drive forward until left bumper is hit */
OnFwd(DRIVE_LEFT + DRIVE_RIGHT);

while(BUMP_LEFT) { /* Do nothing until BUMP_LEFT = 1, so BUMP_LEFT = 0 */
}

/* Stop when this loop is exited */
do_stop();

/* Define the do_stop subroutine */
void do_stop ()
{
    OR(DRIVE_LEFT + DRIVE_RIGHT);
}

Resources

- Lego® Mindstorms™ kit with extras (1 per team)
- PC running the Brick Command Center programming environment (1 per team in Eiel)
- Team/schedule details (in course booklet)
- Copy of these slides (in course booklet)
- Copy of challenge description (in course booklet)
- Mindstorms For Schools booklet (1 per team)
- Programming Lego Robots using NQC (2 per team) - please ensure these are returned!
- Sample robots and software to use as templates for individual designs
- Copy of Getting Started instruction sheet (in course booklet)

Project Programme

Wednesday 25 July
09.00 Introduction to project (LR4)
09.45 Introduction to equipment (Eiel)
10.00 Design starts (Eiel)
11.00 Break
11.30 Design session (Eiel)
Construction, programming and testing begin when ready
12.30 Lunch
14.15 Construction, programming and testing (Eiel)
15.00 Break
15.30 Some: leave for supervisions (various locations)
Others: project session continues (Eiel)
16.30 Others: leave for supervisions (various locations)
Some: project session (Eiel)
17.30 All return to host College

Thursday 26 July
09.00 Tinkering, programming and testing (Eiel)
10.00 Final testing (Eiel)
10.30 Break (end of project)
Getting Started

Login  (This probably will have already been done!)
Username: ew1**\lego  (where ew1** is the number of the machine, e.g. ew101)
Password: mindstorms

Connecting the tower
Plug in the USB IR tower. Make sure you plug it into the front of the correct machine – see the number on the monitor ew1**.

Starting the software
On the Desktop Click on Bricx Command Centre
A window will appear called ‘Searching for the brick’. Select Automatic, RCX and standard options.
Bricx Command Centre is the program you will be using to write your program and to download it to your robot

Writing your first program
A number of basic programs have been written to help get you started. These are described in the 'Programming Lego Robots using NQC' guide.

To load these programs, click on File --> Open, then select the file you want to view. The examples are in C/Program Files/BricxCC/examples

The first program to look at is 1_simple.nqc. This program is very simple and can be used to test out the software and motors. It is described on page 6 of the 'Programming Lego Robots using NQC' tutorial. PLEASE READ.

The robot has 6 sockets on the top. The three black sockets are marked A, B, C and are motor ports. For this exercise plug 2 motors in ports A and B.

The three grey sockets are marked 1,2,3 and are sensor ports. In the first exercise you will not need any sensors.

You can now save a copy of the program so that you can edit it. To do this click on:

File --> Save as

Save on the desktop. DON'T USE RANDOM NAMES, AS YOU WILL FORGET WHAT THEY MEAN LATER!
Compiling and downloading the program

Now, you can compile your program – this means asking the computer to translate your program into something the robot can understand.

To do this, go to: Compile --> compile

If the program is bug free, this will not give any errors. If there are you will need to find the problem in your program and fix it, then try again. The error message will give a line number to help you identify the error.

The compiled program can now be downloaded to the robot. To do this, turn on the robot, and place it so that its IR receiver (window at the front) is pointing at the tower.

Then go to: Compile --> download

Up to 5 programs can be stored in the robot. Press the grey 'Prgm' button on the robot to change the program number.

Running the program

Now you can run your program. Press the green run button on your robot or use the run option in the Brix command window.

Does the robot do what you expect it to do? If not the wires check the wires are properly connected into the correct sockets.

Play time

Now use the tutorial to investigate
- speed of motors p8
- making turns p9
- using sensors p16

Once you have investigated the basic programs you can start writing your own.

When you make a large change to this program you can download it to a different slot on the robot. As before, remember to save your file with a new name every time you make a big change. Otherwise, if it goes wrong, you're in trouble!!

Finally

REMEMBER TO SAVE YOUR PROGRAM AS YOU WORK!

Have some fun!