

Programming – Motors and Timers

Aim: How can we write basic NQC programs with motors and timers?

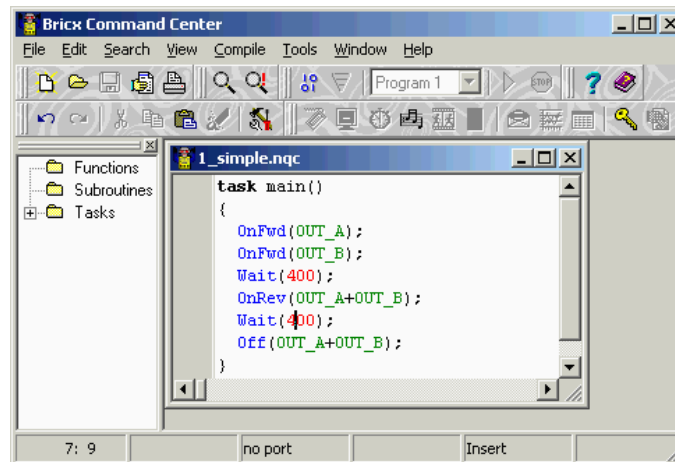
Goals

- Develop a feel for programming in NQC and navigating around Brick Command Center
- Understand the functions of the basic commands – outputs, timers
- Know how to interpret programs – the “language” of describing what’s going on

Getting Started...

- Yesterday we discussed how to open Brick Command Center, and looked at its features and menus
- Remember that once you open the software, you need to hit File >> New to open the window where you write your program

File >> New File



Starting a Program

- Programs are written as a series of typed commands that dictate what you want the Robot to do
- Each set of commands can form a task
- Tasks are enclosed within brackets
- All tasks have names
 - Every program must have a task named main
- At this point, we will be writing our entire programs as one main task: `task main () { }`

Yesterday, you wrote this program...

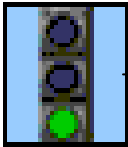
```

task main ( )
{
    OnFwd(OUT_A + OUT_C);
    Wait(200);
    OnRev(OUT_A + OUT_C);
    Wait(200);
    Off(OUT_A + OUT_C);
}

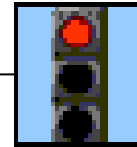
```

A set of commands is enclosed within the task main (between the brackets).

Robolab Translation



Blah Blah Blah Blah Blah Blah
Blah



Becomes

```

task main ( )
{
    Blah blah blah;
    Blah blah blah;
    Blah = Blah * Blah;
    Blah += 5;
}

```

Task can also be found in Templates...

Templates	
Definition Commands	<pre> task .. (..) sub .. (..) void .. (..) int .. #define .. #include .. #pragma reserve .. </pre>
	<pre> SetUpperLimit (...); SetLowerLimit (...); UpperLimit (...); LowerLimit (...); SetHysteresis (...); Hysteresis (...); SetClickTime (...); ClickTime (...); SetClickCounter (...); ClickCounter (...); </pre>
Structures	<pre> # (...) # (...) # (...) else { } while (...) do { } while (...) repeat (...) until (...) until (...) switch (...) { case ...: break; default ...: } acquire (ACQUIRE_...); catch (...) monitor (EVENT_MASK (...)); catch (...) </pre>
	<pre> OnFwd (...); OnRev (...); Off (...); Float (...); OnFor (...); SetPower (...); Wait (...); </pre>
	<pre> SetSensorType (..._TOUCH); SetSensorType (..._LIGHT); SetSensorType (..._TEMPERATURE); SetSensorType (..._ROTATION); SetSensorMode (..._RAW + ...); SetSensorMode (..._BOOL); SetSensorMode (..._EDGE); SetSensorMode (..._PULSE); SetSensorMode (..._PERCENT); SetSensorMode (..._CELSIUS); SetSensorMode (..._FAHRENHEIT); SetSensorMode (..._ROTATION); ClearSensor (...); Sensor (...); Message () SendMessage (...); ClearMessage (); CreateDatalog (...); AddToDatalog (...); Timer (...); ClearTimer (...); SetTimer (...); FastTimer (...); </pre>
	<pre> Counter (...); ClearCounter (...); IncCounter (...); DecCounter (...); PlaySound (_CLICK); PlaySound (_DOUBLE_BEEP); PlaySound (_DOWN); PlaySound (_UP); PlaySound (_LOW_BEEP); PlaySound (_FAST_UP); PlayTone (...); MuteSound (...); UnmuteSound (...); ClearSound (...); SetUserDisplay (...); SetRandomSeed (...); SetWatch (...); SetSleepTime (...); SleepNow (...); BatteryLevel () </pre>
	<pre> SetPriority (...); ActiveEvents (...); SetEvent (...); ClearEvent (...); ClearAllEvents (...); EventState (...); CalibrateEvent (...); </pre>

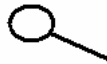
Diagram illustrating the mapping of Robolab commands to their corresponding functions in the Templates library. The diagram shows a grid of templates categorized into Definition Commands and Structures. Specific functions are highlighted with callouts:

- Output Commands:** A callout box points to the `ClickCounter (...)` function in the Structures column.
- Sensor Commands:** A callout box points to the `SetSensor (..._TOUCH);` through `SetSensor (..._EDGE);` functions in the Structures column.
- Sound Statements:** A callout box points to the `PlaySound (...)` and `PlayTone (...)` functions in the Structures column.

Semicolons

- As you may have noticed, NQC requires that you place a semicolon at the end of each statement
- Bricx gets pretty pissed if you don't put one in...
- Lack of a semicolon will result in a program error

One small mistake...

```
task main ()  
{  
    OnFwd(OUT_A + OUT_C);  
    Wait (900)  So, I forgot  
}
```

So when I went to compile the program, the error list gave me this...

```
line 6: Error: parse error
```

A parse error is a syntax error (no semicolon, something's not capitalized, etc)

Motor Commands

- Need 3 things
 - 1) Direction – Forward or Reverse
 - 2) Port Assignments – A, B, C
 - 3) Power Level – 0, 1, 2, 3, 4, 5, 6, 7
- In NQC, there are many output commands
- Some commands can be used to give more than one instruction to the robot, and others are more specific (1 thing at a time)
- Fortunately, most of them are pretty straightforward
- Examples:

```
OnFwd ( ) ;
```

```
OnRev ( ) ;
```

- OnFwd** – turns motors on and runs them in the forward direction
- OnRev** – turns motors on and runs them in reverse

Port Assignments

- Output ports are assigned with these simple statements:
 - `OUT_A, OUT_B, OUT_C`
- Note: NQC is case sensitive
 - Capitalization counts
- Port Assignments go in the parentheses that follow the `OnFwd` and `OnRev` commands

Example

```
OnFwd (OUT_A + OUT_C) ;  
OnRev (OUT_A + OUT_C) ;
```

- Multiple port assignments can be made using a '+' sign
- Using these commands, we can assign direction, ports, and turn the motors on in one brief statement

Let' say...

- I screwed up...I didn't capitalize
- Don't worry...there's still hope
- Bricx is nice enough to colorize (green, blue, red) certain key words when they're typed properly
- That way errors become more obvious
- Example:

```
OnFwd (OUT_A + OUT_C) ;  
OnRev (OUT_A + OUT_C) ;
```

What's the error?

Other Motor Commands

- We have other commands that can be used to individualize tasks
 - `On (OUT_A)`
 - `SetOutput (OUT_A + OUT_C, OUT_ON)`
 - `Fwd (OUT_A), Rev (OUT_A), Toggle (OUT_A)`
 - `SetDirection (OUT_A, OUT_FWD)`
 - `OUT_REV` – sets the motors in reverse
 - `OUT_TOGGLE` – reverses the direction the motor was previously moving in

Equivalent Statements

- 1) `OnFwd(OUT_A + OUT_C);`
- 2) `SetDirection(OUT_A + OUT_C, OUT_FWD);`
`On(OUT_A + OUT_C);`
- 3) `Fwd(OUT_A + OUT_C);`
`SetOutput(OUT_A + OUT_C, OUT_ON);`

Turning the Motors Off

● Looking back at the two ways we turn the motors on (given below), what do you suppose are two ways to turn them off?

```
On(OUT_A + OUT_C);  
  
SetOutput(OUT_A + OUT_C, OUT_ON)|  
  
Off(OUT_A + OUT_C);  
SetOutput(OUT_A + OUT_C, OUT_OFF)
```

Float Stop

- There's another type of stop, or "Off," called a Float stop.
- A float stop brakes the motors (rather than cutting their power off immediately)
- In Robolab, the yellow stop sign indicates a Float stop.

Float Stop Commands

● Looking back at the two ways we turn the motors on, what do you suppose are two ways to set a float stop?

```
Float(OUT_A + OUT_C);  
SetOutput(OUT_A + OUT_C, OUT_FLOAT);
```

Power Levels

- Motors can move at different power levels
- Different voltages will be output depending on how you program the robot
- The power levels range from 0-7
 - 0 – lowest power level = slower robot
 - 7 – greatest power level = faster robot
- If no power level is assigned, what power level do you suppose Brick defaults to? 7

Assigning Power Levels

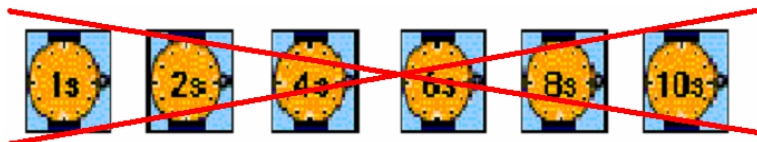
- Give the SetDirection and SetOutput commands (shown below), how do you think we assign power levels?

```
SetPower (OUT_A + OUT_C, 6);
```

- You can assign multiple motors to a particular power level in 1 statement
- Power levels must be assigned before the motor is turned on
- If you want to choose the power level in the middle of the program, you must have another SetPower statement

Timers

- Remember these?



- We don't need several timer icons anymore
- There is one important "Wait For" statement. What is it? Wait()
- Measures in hundredths of a second

Programming with the Wait Command

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

Random Numbers

- The Brick software can generate random numbers for us to make the motion of our robots less predictable
- We use the command: `Random(#)`;
- ie: `Wait(Random(300))`;
What is 300 in this case? Maximum Random Number

OnFor Command

- The `OnFor` command allows us to turn the motors on and assign a time in 1 statement.
- For example:

```
OnFor(OUT_A + OUT_C, 400);  
Off(OUT_A + OUT_C);
```

The basic setup is: `OnFor(Ports, Time in Hundredths of a Second)`

Equivalent Programs

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

```
task main()  
{  
    SetDirection(OUT_A + OUT_C, OUT_FWD);  
    OnFor(OUT_A + OUT_C, 400);  
    SetDirection(OUT_A + OUT_C, OUT_REV);  
    OnFor(OUT_A + OUT_C, 400);  
    Off(OUT_A + OUT_C);  
}
```

Summary of Commands

- `task main () { }`
- `OnFwd(Ports)`, `OnRev(Ports)`
- `On(Ports)`, `Off(Ports)`, `Float(Ports)`
- `Fwd(Ports)`, `Rev(Ports)`, `Toggle(Ports)`
- `SetOutput(Ports, Command)` – `OUT_ON`, `OUT_OFF`, `OUT_FLOAT`
- `SetDirection(Ports, Direction)` – `OUT_FWD`, `OUT_REV`, `OUT_TOGGLE`
- `SetPower(Ports, Power Level)` – 0-7
- `Wait` (Time in Hundredths of a Second)
- `OnFor` (Ports, Time in 1/100ths of a Sec)

Comments

- We can add comments to our programs to make them more readable

- Comments do not get read by the compiler

- Can be added in one of two ways

 - // - Double slash will make everything on the rest of a line a comment

 - /*, */ - For long comments

Examples

- Short comments

```
OnFwd(OUT_A + OUT_C); // Motors A and C move forward at Power Level 7
```

- Long Comments

```
OnFwd(OUT_A + OUT_C); Compiler Reads  
  
/* It's a rare condition, this day and age,  
To read any good news on the newspaper page.  
Love and tradition of the grand design,  
Some people say it's even harder to find.  
Well then there must be some magic clue  
Inside these tearful walls  
  
Cause all I see is a tower of dreams  
Real love burstin' out of every seam.  
As days go by, we're gonna fill our house with happiness.  
The moon may cry,  
We're gonna smother the blues with tenderness.  
  
When days go by, there's room for you, room for me,  
For gentle hearts an opportunity.  
As days go by, It's the bigger love of the family.  
*/  
  
OnFwd(OUT_A + OUT_C); Compiler Reads
```

Comments -
Compiler
Doesn't Read

Examples

```
task main()  
{  
    OnFwd(OUT_A + OUT_C);  
    Wait(700);  
    Off(OUT_A);  
    Wait(300);  
    OnRev(OUT_C);  
    Wait(800);  
}
```

```
task main()  
{  
    SetPower(OUT_A, 4);  
    SetDirection(OUT_A + OUT_C, OUT_FWD);  
    On(OUT_A + OUT_C);  
    Wait(600);  
    On(OUT_B); // A lamp is connected to Port B  
    Wait(150);  
    SetDirection(OUT_C, OUT_TOGGLE);  
    Wait(100);  
    SetDirection(OUT_C, OUT_TOGGLE);  
    Wait(200);  
    Float(OUT_A + OUT_C);  
}
```