

Not eXactly C (NXC)

NXC est un langage simple pour la programmation du produit LEGO NXT MINDSTORMS. Nous décrivons dans ce document uniquement les éléments dont vous avez besoin pour la traduction du modèle demandé. Vous pouvez compléter votre générateur (et donc votre DSML !) pour couvrir plus largement le langage NXC. Pour cela, l'API est disponible à l'adresse : http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf

Structure d'un programme :

Un programme NXC est très proche d'un programme C. Il est donc composé essentiellement de fonctions. NXC introduit néanmoins la notion de tâche (Task), correspondant directement à un thread sur la brique NXT. Une tâche est définie à l'aide du mot clé Task suivant la syntaxe suivante :

```
task name(){
    // the task's code is placed here
}
```

Un programme NXC doit absolument avoir au moins une tâche nommée « main » qui sera le point d'entrée dans le programme.

API NXC (extrait!) :

Wait (time)

Function

Make a task sleep for specified amount of time (in 1000ths of a second). The time argument may be an expression or a constant.

```
Wait(1000); // wait 1 second
Wait(Random(1000)); // wait random time up to 1 second
```

Stop (bvalue)

Function

Stop the running program if bvalue is true. This will halt the program completely, so any code following this command will be ignored.

```
Stop(x == 24); // stop the program if x==24
```

Random (n)

Value

Return an unsigned 16-bit random number between 0 and n (exclusive). n can be a constant or a variable.

```
x = Random(10); // return a value of 0..9
```

Random ()

Value

Return a signed 16-bit random number.

```
x = Random();
```

SetSensorLowspeed (port)

Function

Configure the sensor on the specified port as an I2C digital sensor (9V powered). The port may be specified using a constant (e.g., S1, S2, S3, or S4) or a variable.

```
SetSensorLowspeed(S1);
```

SensorUS (n)

Value

Return the processed sensor reading for an ultrasonic sensor on port n, where n is 0, 1, 2, or 3 (or a sensor port name constant). Since an ultrasonic sensor is an I2C digital sensor its value cannot be read using the standard Sensor(n) value. A variable whose value is the desired sensor port may also be used.

```
x = SensorUS(S4); // read sensor 4
```

Output Port Constants	Value
OUT_A	0x00
OUT_B	0x01
OUT_C	0x02
OUT_AB	0x03
OUT_AC	0x04
OUT_BC	0x05
OUT_ABC	0x06

Figure 1 : Constantes des ports de sortie

Off(outputs)

Function

Turn the specified outputs off (with braking). Outputs can be a constant or a variable containing the desired output ports. Predefined output port constants are defined in Figure 1.

```
Off(OUT_A); // turn off output A
```

OnFwdSync(outputs, pwr, turnpct)

Function

Run the specified outputs forward with regulated synchronization using the specified turn ratio. Outputs can be a constant or a variable containing the desired output ports. Predefined output port constants are defined in Figure 1.

```
OnFwdSync(OUT_AB, 75, -100); // spin right
```

OnRevSync(outputs, pwr, turnpct)

Function

Run the specified outputs in reverse with regulated synchronization using the specified turn ratio. Outputs can be a constant or a variable containing the desired output ports. Predefined output port constants are defined in Figure 1.

```
OnRevSync(OUT_AB, 75, -100); // spin left
```

RotateMotorEx(outputs, pwr, angle, turnpct, sync, stop) Function

Run the specified outputs forward for the specified number of degrees. Outputs can be a constant or a variable containing the desired output ports. Predefined output port constants are defined in Figure 1. If a non-zero turn percent is specified then sync must be set to true or no turning will occur. Specify whether the motor(s) should brake at the end of the rotation using the stop parameter.

```
RotateMotorEx(OUT_AB, 75, 360, 50, true, true);
```

PlayToneEx(frequency, duration, volume, bLoop)

Function

Play a single tone of the specified frequency, duration, and volume. The frequency is in Hz. The duration is in 1000ths of a second. Volume should be a number from 0 (silent) to 4 (loudest). All parameters may be any valid expression.

```
PlayToneEx(440, 500, 2, false);
```

NumOut(x, y, value, clear = false)

Function

Draw a numeric value on the screen at the specified x and y location. Optionally clear the screen first depending on the boolean value of the optional "clear" argument. If this argument is not specified it defaults to false.

```
NumOut(0, LCD_LINE1, x);
```

TextOut(x, y, msg, clear = false)

Function

Draw a text value on the screen at the specified x and y location. Optionally clear the screen first depending on the boolean value of the optional "clear" argument. If this argument is not specified it defaults to false.

```
TextOut(0, LCD_LINE3, "Hello World!");
```

Button Constants	Value
BTN1, BTNEXIT	0
BTN2, BTNRIGHT	1
BTN3, BTNLEFT	2
BTN4, BTNCENTER	3
NO_OF_BTNS	4

Figure 2 : Constantes des boutons

ButtonPressed(btn, reset)

Value

Return whether the specified button is pressed. Optionally clear the press count. Valid values for the btn argument are listed in Figure 2.

```
value = ButtonPressed(BTN1, true);
```