Appendix 2: A textual description language equivalent to the NXT-G graphical language

Author: Michele Moro

Objective

Definition of a textual language exactly equivalent to NXT-G to be used in presenting NXT programming code in substitution of large snapshots of the NXT-G GUI. It is defined using English acronyms but several national versions can be easily provided.

General requirements

- A name can be enclosed within “” when spaces are present in the name
- Blocks are functionally represented with a recognizable name, with an optional label for forward/backward references
- Any block parameter is specified as a couple <paramname>=<value> where value can be a predefined constant (e.g. ON, OFF, COUNT), a numerical or logical value (e.g. 123, TRUE), or the output of a previous block connected through a data wire
- An output of a block is used as a qualifying name of the label representing such a block when such an output must be connected to another block input: thus the qualified label is a way to represent an input data wire coming out from the labelled block (e.g. SUM1:MathOp(Type=ADD, A=37, B=56) . . . MathOp(Type=MUL, A=SUM1.Res, B=-23) SUM1.Res represents a data wire connecting the Res(ult) output of the SUM1 adding block with the input A of the multiplier block)
- The list of possible input/output connections is given at the end of the description of each command block. When a input connection has its corresponding output, the two connections share the same name: for example, in the Motor block labelled with M1, the Port parameter represents also an input/output connection, that is its may be defined as the output of one other block, as explained above, but also M1.Port could be the definition of a parameter of another block.

Structure

- Variable declaration is an off-line activity

VarDecl (Name=<varName>, Type=LOGIC|NUM|TXT)

- MyBlock definition

```
MyBlock (Name=<BlockName>, InParams=(<InputParamList>),
  OutParams=(<OutputParamList>)) {
        -- code for the command block
<BlockName>}
```

where inputParamList and outputParamList are lists of qualified names:
<ParamName>.[LOGIC|NUM|TXT]

In the block body the reading references to the input parameters (i.e. the 'abstract' input data wires of the block) are made with the name of the parameter, optionally qualified with its type indicator (e.g. Par1.NUM) for better readability. 'Abstract' output data wired, used to set the output parameters, are expressed by the block :

```
SetOut(
        Name=<OutputParamName>.[LOGIC|NUM|TXT]
        Val = TRUE | FALSE – if .Type==LOGIC
                <IntVal> -- if .Type==NUM
                <Text> -- if .Type==TXT
        )
```

- Multitasking is equivalent to a fork in the flow control; it can be nested in more than one level

```
Fork
{1 <commands> 1}                {1 . . .{1.1 . . . 1.1} 1}
{2 <commands> 2}
```

- Control structures define blocks of code, delimited by square bracket, generally nestable.

```
Loop
<lab>:Loop(. . .) [
        -- loop body
<lab>]
```

Switch

1. Two choices (equivalent to an if-then-else)

```
<lab> Switch(. . .) [<lab>.IF
        -- true part
<lab>.IF]
[<lab>.ELSE
        -- else part
<lab>.ELSE]
```

2. Multi-choice (equivalent to a switch-case)

```
<lab> Switch(. . .) [<lab>.case1
        -- case 1 part
<lab>.case.1]
```

[<lab>.case.2
-- case 2 part
<lab>.case.2]
. . .
[<lab>.case.n
-- case n part
<lab>.case.n]

Common palette

Move (

Ports = A | B | C | AB | AC | BC | ABC
Dir = FD | BK | STOP
StLt = A | B | C
StRt = A | B | C
Steer = <-100 .. 100>
Pwr = <0 .. 100>
Dur = FOREVER | <IntVal>.[DEG | ROT | SEC]
Next = BRK|COA
)
In=MotorL(=1|2|3), MotorR(=1|2|3), MotorO(=1|2|3), Dir(=TRUE|FALSE), Steer, Pwr,
Dur, Next (=TRUE|FALSE)
Out=see In

Record (

Act=REC | PLAY
Name = <Name>
Ports = A | B | C | AB | AC | BC | ABC
Time = <NatVal> -- if Act==REC
)
In=Act (=0|1), Name, PortA(=TRUE|FALSE), PortB(=TRUE|FALSE), PortC(=TRUE|FALSE), Time, Rate(=<0..255> Hz)
Out=see In

Sound (

Act = FILE | TONE
Ctrl = PLAY | STOP
Vol = <0..100>
Rep = ON | OFF
File = <Text> -- if Act==FILE
Note = <C1..B3> -- if Act==TONE
Dur = <Val> -- if Action==TONE
Wait = ON | OFF – if Rep==OFF
)
In=Act(=0|1), File, Freq(=<NatVal> Hz), Ctrl (=0|1), Vol,

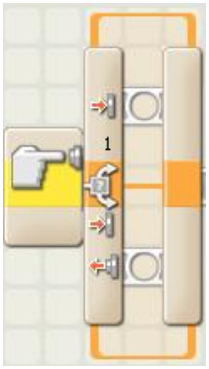
Dur (=<NatVal> in ms)
Out=see In

Display (

Act = IMG | TXT | DRAW | RESET
Clr = ON | OFF
File = <Name> -- if Act==IMAGE
Txt = <Text> -- if Act== TEXT
Type = POINT | LINE | CIRCLE -- if Act=DRAW
PosX = <0..99> -- if Act!=RESET
PosY = <0..63> -- if Act!=RESET
LastX = <0..99> -- if Act==DRAW && Type=LINE
LastY = <0..63> -- if Act==DRAW && Type=LINE
Line = <1..8> -- if Act==TEXT
Rad = <0..120> -- if Act==DRAW && Type=CIRCLE
)
In=Act(0=IMG, 1=TXT, 2=POINT, 3=LINE, 4=CIRCLE, 5=RESET),
Clr(=TRUE|FALSE), File, Text, PosX, PosY, LastX,
LastY, Rad
Out=see In

Wait (
Ctrl = SENSOR | TIME
Sensor= SOUND | TOUCH | LIGHT | BUTTONS | RCVMSG | ROT | TIMER
SONAR | TOUCH* | LIGHT* | ROT* | TEMP* -- if Ctrl== SENSOR
Port = 1 | 2 | 3 | 4 -- if Ctrl==SENSOR && Sensor==SOUND, TOUCH, LIGHT,
-- SONAR, TOUCH*, LIGHT*, ROT*, TEMP*
Button = OK | LEFT | RIGHT -- if Ctrl==SENSOR && Sensor==BUTTONS
Msg = TEXT | NUMBER | LOGIC -- if Ctrl==SENSOR && Sensor = RCVMSG
Port = A | B | C -- if Ctrl==SENSOR && Sensor==ROT
Timer = 1 | 2 | 3 -- if Ctrl==SENSOR && Sensor == TIMER
Until = [GT | LT] <0 .. 100> -- if Ctrl==SENSOR &&
-- Sensor==SOUND, LIGHT, LIGHT*
[FD | BK] [GT LT] <IntVal>.[DEG | ROTS]
-- if Ctrl==SENSOR && Sensor == ROT

```
                [GT | LT] <IntVal> -- if Ctrl==SENSOR && Sensor==TIMER
                [GT | LT] <0 .. 250> -- if Ctrl==SENSOR &&
                                        -- Sensor==SONAR && Show == CM
                [GT | LT] <0 .. 100> -- if Ctrl==SENSOR &&
                                        -- Sensor==SONAR && Show == IN
                [FD | BK] [GT | LT] <IntVal> -- if Ctrl==SENSOR &&
Sensor==ROT*
                [GT | LT] <-20 .. 70> -- if Ctrl==SENSOR &&
                                        -- Sensor==TEMP* && Show = CEL
                [GT | LT] <-4 .. 158> -- if Ctrl==SENSOR &&
                                        -- Sensor==TEMP* && Show = FAR
                 <NatVal> -- if Ctrl==TIME
        Act = PRESS | REL | BUMP -- if Ctrl==SENSOR &&
                                                -- Sensor = TOUCH, BUTTONS,
                                        TOUCH*
        Mailbox = <1 .. 10> -- if Ctrl==SENSOR && Sensor = RCVMSG
        CmpWith = <Text> -- if Ctrl==SENSOR &&
                                -- Sensor==RCVMSG && Msg == TEXT
                        <IntVal> -- if Ctrl==SENSOR &&
                                -- Sensor==RCVMSG && Msg == NUMBER
                        TRUE | FALSE -- if Ctrl==SENSOR &&
                                        -- Sensor==RCVMSG && Msg ==
                                        LOGIC
        Show = CM | IN -- if Ctrl==SENSOR && Sensor == SONAR
                CEL | FAR -- if Ctrl==SENSOR && Sensor == TEMP*
        Light = ON | OFF -- if Ctrl==SENSOR && Sensor == LIGHT
        )
        In=//
        Out=//
Loop (
        Ctrl = FOREVER | SENSOR | TIME | COUNT | LOGIC
        -- if Ctrl==SENSOR || Ctrl==TIME see the options in Wait block
        Until = <NatVal> -- if Ctrl==COUNT
                TRUE | FALSE -- if Ctrl=LOGIC
        Dis = ON | OFF
        )
        In=Until (=TRUE | FALSE if Ctrl=LOGIC)
        Out=Cnt
Switch (
        Ctrl = SENSOR | VAL
        -- if Ctrl==SENSOR see the options in Wait block
        -- for fields Sensor, Cmp and CondUp
        Type = LOGIC | NUMBER | TXT -- if Ctrl==VAL
```



Dis=ON|OFF
CondUp=TRUE|FALSE -- if Ctrl==VAL && Type==LOGIC
Cond<1..n>=<IntVal>.DFLT-- if Ctrl==VAL && Type==NUMBER && Dis==OFF
Cond<1..n>=<Text>.DFLT-- if Ctrl==VAL && Type==TXT && Dis==OFF
)
In=Val (if Ctrl==VAL)
Out=//

Action palette

Motor (
Port = A | B | C
Dir = FD | BK | STOP
Act = CONST | RAMPUP | RAMPDW -- if Dur.Type==DEG || Dur.Type==ROT
Pwr = <0 .. 100>
PwrCtrl = ON | OFF
Dur = FOREVER | <IntVal>.[DEG | ROT | SEC]
Wait = ON | OFF – if Dur.Type==DEG || Dur.Type==ROT
Next = BRK|COA -- if Wait==ON || Dur.Type=SEC
)
In=Port, Dir(=TRUE|FALSE), Act(=0|1|2), Pwr, PwrCtrl(=TRUE|FALSE), Dur, Wait (=TRUE|FALSE),Next (=TRUE|FALSE)
Out=see In, DirOut(=TRUE|FALSE), DegOut(=<NatVal>)

SendMsg (
Conn = 0 | 1 | 2 | 3
MsgType = TXT | NUM | LOGIC
Msg = <Text> -- if MsgType==TXT
Msg = <IntVal> -- if MsgType==NUM
Msg = TRUE | FALSE -- if MsgType==LOGIC
Mbx = <1..10>
)
In=Conn, Mbx, Txt (if MsgType==TXT), Val (if MsgType==NUM), Bool (if MsgType==LOGIC)
Out=see In

Motor* (
Port = A | B | C
Dir = FD | BK | STOP
Pwr = <0 .. 100>
)
In=Port, Dir(=TRUE|FALSE), Pwr
Out=see In

Lamp* (
Port = A | B | C
Act = ON | OFF
Int = <0 .. 100>
)
In=Port, Act (=TRUE|FALSE), Int
Out=see In

Sensors palette

TouchSens (
Port = 1 | 2 | 3 | 4
Act = PRESS | REL | BUMP
)
In=Port, Act(=0|1|2)
Out=see In, Res(=TRUE|FALSE), Raw (=<0..1024>)

SoundSens (
Port = 1 | 2 | 3 | 4
Cmp = [GT | LT] <0 .. 100>
)
In=Port, Cmp, Gt(=TRUE|FALSE), dbA (=TRUE|FALSE)
Out=see In, Res(=TRUE|FALSE), Lev (=<0..100>), Raw (=<0..1024>)

LightSens (
Port = 1 | 2 | 3 | 4
Cmp = [GT | LT] <0 .. 100>
Light = ON | OFF
)
In=Port, Cmp, Gt(=TRUE|FALSE), Light
Out=see In, Res, (=TRUE|FALSE), Int (=<0..100>), Raw (=<0..1024>)

SonarSens (
Port = 1 | 2 | 3 | 4
Cmp = [GT | LT] <0 .. 250> -- if Show == CM
Cmp = [GT | LT] <0 .. 100> -- if Show == IN
Show=CM | IN
)
In=Port, Cmp, Gt(=TRUE|FALSE)
Out=see In, Res, Dist

Buttons (
Button = OK | LEFT | RIGHT
Act = PRESS | REL | BUMP
)
In=Button(=1|2|3), Act(=0|1|2)
Out=see In, Res

RotSens (

Port = A | B | C
Act = RD | RST
Cmp = [FD | BK] [GT LT] <IntVal>.[DEG | ROTS]
)
In=Port, Tdir(=TRUE|FALSE), Cmp, Gt(=TRUE|FALSE), Rst(=TRUE|FALSE)
Out=see In, Res(=TRUE|FALSE), Dir(=TRUE|FALSE), Deg(=<NatVal>)

Timer (
Num = 1|2|3
Act = RD | RST
Cmp = [GT LT] <0..100>
)
In=Num, Cmp, Gt(=TRUE|FALSE), Rst(=TRUE|FALSE)
Out=see In, Res(=TRUE|FALSE), Val(=<NatVal>)

RcvMsg (
MsgType = TXT | NUM | LOGIC
Cmp = <Text> -- if MsgType==TXT
Cmp= <IntVal> -- if MsgType==NUM
Cmp = TRUE | FALSE -- if MsgType==LOGIC
Mbx = <1..10>
)
In= Mbx, Txt (if MsgType==TXT), Val (if MsgType==NUM), Bool (if MsgType==LOGIC)
Out=see In, Rcv(=TRUE|FALSE), Cmp(=TRUE|FALSE), TxtOut(if MsgType==TXT), NumOut(if MsgType==NUM), BoolOut(if MsgType==LOGIC)

TouchSens* (
Port = 1 | 2 | 3 | 4
Act = PRESS | REL | BUMP
)
In=Port, Act(=0|1|2)
Out=see In, Res(=TRUE|FALSE), Raw (=<0..1024>)

RotSens* (
Port = 1 | 2 | 3 | 4
Act = READ | RESET
Cmp = [FD | BK] [GT LT] <NatVal>
)
In=Port, Tdir(=TRUE|FALSE), Cmp, Gt(=TRUE|FALSE), Rst(=TRUE|FALSE)
Out=see In, Res(=TRUE|FALSE), Dir(=TRUE|FALSE), Tick(=<NatVal>)

LightSens* (
Port = 1 | 2 | 3 | 4
Cmp = [GT | LT] <0 .. 100>
)
In=Port, Cmp, Gt(=TRUE|FALSE)
Out=see In, Res, (=TRUE|FALSE), Int (=<0..100>), Raw (=<0..1024>)

TempSens* (
Port = 1 | 2 | 3 | 4
Cmp = [GT | LT] <-20 .. 70> -- if Show = CEL
[GT | LT] <-4 .. 158> -- if Show = FAR
)
In=Port, Cmp, Gt(=TRUE|FALSE)
Out=see In, Res, (=TRUE|FALSE),
Temp (=<-20..70> if Show=CEL, <-4 .. 158> if Show = FAR),
Raw (=<0..1024>)

Control palette

Stop (
//
)
In=Cond(=TRUE|FALSE)
Out=see In

Data palette

LogicOp (
Type = AND | OR | XOR | NOT
A=TRUE | FALSE
B=TRUE | FALSE
)
In=A, B
Out=see In, Res(=TRUE|FALSE)

MathOp (
Type = ADD | SUB | MUL | DIV
A=<IntVal>
B=<IntVal>
)
In=A, B
Out=see In, Res(=<IntVal>)

CmpOp (
Type = LT | GT | EQ
A=<IntVal>
B=<IntVal>
)

```
        In=A, B
        Out=see In, Res(=TRUE|FALSE)
Range   (
        Type = IN | OUT
        A = <0..B>
        B = <A..100>
        Val = <IntVal>
        )
        In=A, B, Val
        Out=see In, Res(=TRUE|FALSE)
Rand (
        A = <0..B>
        B = <A..100>
        )
        In=A, B
        Out=see In, Val(=<A..B>)
Var (
        Name = <Name>.[LOGIC | NUM | TXT]
        Act = RD | WR
        Val = TRUE | FALSE -- if .Type==LOGIC && Act==WR
                <IntVal> -- if .Type==NUM && Act==WR
                <Text> -- if .Type==TXT && Act==WR
        )
        In=Val (if Act==WR)
        Out=Val
```

Advanced palette

```
Text (
        A = <Text>
        B = <Text>
        C = <Text>
        )
        In=A, B, C
        Out=see In, Val(=<Text>)
N2Txt (
        Num = <IntVal>
        )
        In=Num
        Out=see In, Txt(=<Text>)
KeepAl (
        //
        )
        Out=Time(=<intVal> in ms)
```

File (
Act = RD | WR | CL | DL
Name = <Filename>
Type = TXT | NUM -- if Act==RD || Act==WR
Txt = <Text> -- if Act==WR && Type==TXT
Val = <IntVal> -- if Act==WR && Type==NUM
)

In=Name, Dim(=<NatVal> if Act==WR && file doesn't exist),
Txt(if Act==WR && Type==TXT), Val(if Act==WR && Type==NUM)
Out=see In, Err(=TRUE|FALSE), TxtOut(if Act==RD && Type==TXT),
NumOut(if Act==RD && Type==NUM)

Calib (
Port = 1 | 2 | 3 | 4
Sens = SOUND | LIGHT | LIGHT*
Act = CLB | ERA
Val = MAX | MIN -- if Act==CLB
)

In=Port, Max(=TRUE|FALSE), Era(=TRUE|FALSE)
Out=see In

Reset (
A = ON | OFF
B = ON | OFF
C = ON | OFF
)

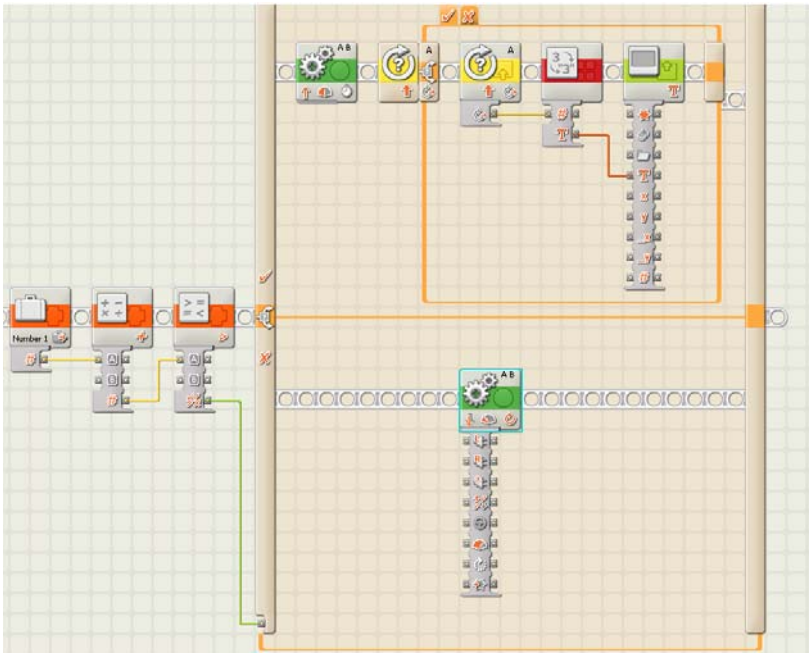
In=A, B, C
Out=see In

Example

```
V1:     Var(Name="Number 1".NUM, Act=RD)
A1:     MathOp(Type=ADD, A=V1.Val, B=20)
C1:     CmpOP(Type=GT, A=A1.Res, B=30)
Sw1:    Switch(Ctrl=VAL, Type=LOGIC, Val=C1.Res, Dis=ON, CondUp=TRUE)
[Sw1.IF
            Move(Ports=AB, Dir=FD, StLt = A, StRt = B,
              Steer=0, Pwr=60, Dur=2.SEC, Next=BRK)
        Sw2:    Switch(Ctrl=SENSOR, Sensor=ROT, Dis=OFF, Port=A, Act=RD,
                  CondUp=FD GT 360.DEG) [Sw2.IF
                Rs1:    RotSens(Port = A, Act=READ, Cmp=GT 0.DEG)
                Nt1:    N2Txt(Num=Rs1.Sound)
                        Display(Act=TXT,    Clr=ON,    Txt=Nt1.Txt,    PosX=8
        PosY=32, Line=4)
                Sw2.IF]
```

```
            [Sw2.ELSE      Sw2.ELSE]
        Sw1.IF]
        [Sw1.ELSE
            Move(Ports=AB, Dir=BK, StLt = A, StRt = B,
              Steer=0, Pwr=30, Dur=1.ROT, Next=BRK)
        Sw1.ELSE]
```

It corresponds to:



The My Block example:

```
MyBlock(Name=sumis0, InParams=(Par1.NUM, Par2.NUM),
  OutParams=(Sum.NUM, Is0.LOGIC)) {
        S01: MathOp(Type=ADD, A=Par1.NUM, B=Par2.NUM)
        S02: CmpOp (Type=EQ, A=S01.Res, B=0)
        SetOut(Name=Sum.NUM; Val=S01.Res)
        SetOut(Name=Is0.LOGIC; Val=S02.Res)
sumis0}
```

it corresponds to the following: