NQC – Not Quite C

- A subset of ANSI-C for use with the RCX
  - RCX API
- Running the command line version
  C:> nqc –Susb –d –Trcx2 pgm_name.nqc
- GUI Interface to NQC – BricxCC
- Both available for free at:
  - http://bricxcc.sourceforge.net/nqc/
- References:
  - http://www.cs.binghamton.edu/~reckert/480/NQC Guide.pdf
  - Chapters 2-4, Appendix B of Baum textbook
- Two sets of example program listings:
  - http://www.cs.binghamton.edu/~reckert/480/480pgms.htm

Some NQC Program Control Statements

- while (condition) {body}
  - Repeats { } as long as condition is true
- until (condition) { }
  - Repeats { } until condition is true
  - Equivalent to: while (!condition) { }
  - Most frequently used with empty body:
    - until(SENSOR_1==1); // wait until SENSOR_1 is pressed
- repeat (expression) { }
  - Repeats { } the number of times 'expression' evaluates to

Output Commands (Motors)

- Motor IDs: OUT_A, OUT_B, OUT_C
  - Can use combinations: OUT_A + OUT_C
- Motor attributes:
  - Modes: OUT_ON, OUT_OFF, OUT_FLOAT
  - Directions: OUT_FWD, OUT_REV, OUT_TOGGLE
  - Power: 0 - 7
- Basic commands:
  - SetOutput(motorIDs, mode)
    - e.g., SetOutput(OUT_A, OUT_ON);
  - SetDirection(motorIDs, direction)
    - e.g., SetDirection(OUT_A+OUT_C, OUT_FWD);
  - SetPower(motorIDs, power)
    - e.g., SetPower(OUT_B, 4);
- Input Function:
  - OutputStatus(n); // n must be 0, 1, or 2
    - Returns current power setting for motor n

More NQC Control Statements

if (condition) then {if clause}
else {then clause}

- Just like regular C 'if-then-else' statement

for (statement1; condition; statement2) { }

- Just like regular C 'for' statement

switch (expression)
{case (const expression): statements; break
case (const expression): statements; break
etc.
default: statements; break
}

- Just like regular C 'switch/case' statement

Motor Convenience Functions

- On (motorIDs)
- Off (motorIDs)
- Float (motorIDs)

  SetOutput(--, --)

- Fwd (motorIDs)
- Rev (motorIDs)
- Toggle (motorIDs)

  SetDirection(--, --)

- OnFwd (motorIDs)
- OnRev (motorIDs)
- OnFor (motIDs, time)

  Fwd(--); On(--);
  Rev(--); On(--);
  OnFwd(--); Wait(time);

  - Wait(time); // pauses for 'time' 10 msec intervals

Some Example Programs

- 1_speed
  - Use of motor commands
- 3_spiral
  - Make robot turn several times
  - Use of repeat, constants, & variable
- 3_random
  - While loop and random numbers
    - Random(n) -- returns a pseudorandom number between 0 and n
- 4_if
  - Turns left or right randomly, uses if-then-else

Sounds on RCX

- Built-in speaker can play pre-programmed sounds or specified tones for specified intervals
- PlaySound(sound_ID_number)
  - Predefined sounds: SOUND_CLICK, SOUND_DOUBLE_BEEP, SOUND_DOWN, SOUND_UP, SOUND_LOW_BEEP, SOUND_FAST_UP
- PlayTone(Frequency, Duration)
  - Frequency in Hz
  - Duration in 10 msec (1/100 sec) intervals
    - must be a constant
- See Piano tool in bricxCC
- Example program: 7_sounds
  - Plays all the predefined sounds
- Example program: 7_music
  - Plays a tune

Sensor Functions

- SetSensor(SensorID, Configuration)

  SetSensor(SENSOR_1, SENSOR_TOUCH);
- SetSensorType(SensorID, Type)

  SetSensorType(SENSOR_1, SENSOR_TYPE_NONE);
- SetSensorMode(SensorID, Mode)

  SetSensorMode(SENSOR_2, SENSOR_MODE_RAW);
- ClearSensor(SensorID)
  - Clears pulse/edge/rotation count to zero

- Reading a Sensor

  x=SENSOR_n or x=SensorValue(n)
  - Returns current value of sensor

  x=SensorValueBool(n)
  - Converts value to a 1 or 0

Sensors

- 3 of them numbered 0, 1, 2 (1,2,3 on RCX)
  - Also named SENSOR-1, SENSOR-2, SENSOR-3
    - These names can be used whenever program wants to read the value of a sensor

      x = SENSOR_1; same as x = SensorValue(0);
  - Each converted digital value comes from a 10-bit ADC
  - Standard firmware samples each sensor every 3 msec.
- Sensor Types
  - SENSOR_TYPE_NONE (to read raw values)
  - SENSOR_TYPE_TOUCH, SENSOR_TYPE_TEMPERATURE
  - SENSOR_TYPE_LIGHT, SENSOR_TYPE_ROTATION
  - Determine how RCX interacts with the sensor
    - e.g., touch sensor is read passively
    - Light sensor must have power supplied to it since it sends out light and records light reflected back

Some Sensor Example Programs

- 5_touch
  - Robot backs away and turns right if left touch sensor is pressed
- 5_light1
  - Goes forward until it detects black, then backs and turns away
    - Threshold set to 37
- pulse-sensor-sound
  - Plays a sound after 4 presses of touch sensor

Sensor Modes & Configurations

- Sensor Modes – how RCX interprets sensor's value
  - SENSOR_MODE_RAW, 0x0, (values 0-1023)
  - SENSOR_MODE_BOOL, 0x2, (values 0 or 1) (>562 or <460)
  - SENSOR_MODE_PERCENT, 0x8, (values 0 to 100)
  - SENSOR_MODE_CELCIUS, 0xA, (values 0 to 100)
  - SENSOR_MODE_FAHRENHEIT, 0xC (values 0-212)
  - SENSOR_MODE_EDGE, 0x4, (counts incoming Boolean transitions)
  - SENSOR_MODE_PULSE, 0x6, (counts incoming rising edges)
  - SENSOR_MODE_ROTATION, 0xE (angle: 0-16, 22.5 degree intervals)
- Sensor Configurations – combination of Type & Mode
  - SENSOR_TOUCH (type Touch, mode Bool)
  - SENSOR_LIGHT (type Light, mode percent)
  - SENSOR_ROTATION (type Rotation, mode Rotation)
  - SENSOR_CELCIUS, CENSOR_FAHRENHEIT (type Temperature)
  - SENSOR_PULSE (type Touch, mode Pulse)
  - SENSOR_EDGE (type Touch, mode Edge)

Multitasking in NQC

- NQC can run up to 10 "simultaneous" tasks
- Task main() begins automatically
- Any task can start/stop any other task
- Define as follows:

  task task_name() { };
- Starting:

  start task_name;
- Stopping:

  stop task_name;

Simple multitasking examples

- 7_drive_music
  - Task main() starts task music() and continually drives forward and backward
    - Task music() plays a tune
- 6_tasks
  - RCX moves in a square until it hits an obstacle
  - Task main() starts two tasks:
    - move_square()
    - check_sensors()

Macros

- Give small pieces of code a name
- Like inline functions in that each time invoked a new copy of the code is generated
- Can have arguments
  - Just placeholders for values to be used when invoked
- Defining:

  #define macro_name(argument_list) statements;
  - If more than one line is needed, must use '\' at end of line
- Example program: 6_macro
  - Power & time are arguments to forwards(s,t), backwards(s,t), turn_right(s,t), turn_left(s,t) macros

Subprograms

- Subroutine
  - Code that can be executed from many places in a program
  - Like procedures, but with restrictions
    - Up to 8 allowed
    - No parameters, no result returned
    - Cannot be nested
    - No recursive calls
    - Risky to call from different tasks
    - Code is only stored once, so efficient use of memory
  - Defining:

    sub sub_name() { };
  - Example: 6_subs
    - Main calls a subroutine that makes RCX turn 360 degrees several times

RCX Timers

- Four of them
  - Count from 0 to 32767 in 1/10 second increments
  - Then rollover to zero
  - Reading a timer:

    x = Timer(n)
  - Resetting a timer:

    ClearTimer(n) // Reset to zero

    SetTimer(n, value) // Reset to specified value
- Timers can also be read more precisely

  x = FastTimer(n) // 1/100 sec. (10 msec.) Intervals
- Example program: 12_timers
  - Go forward & turn randomly until timer times out

Inline Functions

- More like C functions
  - No return value (type void)
  - Can have value and reference parameters
  - Each time invoked a new copy of code is generated
    - Can use a lot of memory
  - No limit on number of inline functions
- Defining:

  void function_name(parameters) { };
  - Just like in C
- Example programs:
  - 6_inline2 (parameter is value of turn time)
  - 6_inline_by_ref)
    - Reference parameter increments n, which is used in caller for delays between outputting a sound

LCD Display

- RCX LCD has 8 display modes
  - DISPLAY_WATCH show system time, default
  - DISPLAY_SENSOR1 show value of sensor 1
  - DISPLAY_SENSOR2 show value of sensor 2
  - DISPLAY_SENSOR3 show value of sensor 3
  - DISPLAY_OUT_A show setting for output A
  - DISPLAY_OUT_B show setting for output B
  - DISPLAY_OUT_C show setting for output C
  - DISPLAY_USER show something else
- Set mode with SelectDisplay(mode)

LCD DISPLAY_USER Mode

- Continually read a source & update LCD display with value
  - Source can be a sensor, timer, global variable, etc.
  - Can display values with a decimal point

    SetUserDisplay (source, digits-after-dec-point)
- Example Programs:
  - timer_display, timer_display_ok

IR Communication

- RCX can send/receive messages using its IR port
- Message values: 0 to 255
- To retrieve most recently-sent message #:

  x = Message(); // 0 returned ⇙ no message received
- Sending a message:

  SendMessage(msg_number)
  - Receiving is disabled while sending
- Clearing the RCX's message buffer:

  ClearMessage();
- Example programs:
  - 11_Master, 11_Slave
    - Master RCX sends out messages to tell slave to go forward, backward, or stop
  - 11_leader
    - Robots decide who is master and who is slave
  - 9_proximity
    - Use IR messages & light sensor to get proximity to an object