

NQC User Manual

Version 2.2 r1, written by Dave Baum

Introduction

NQC stands for Not Quite C, and is a simple language for programming the LEGO RCX.

The preprocessor and control structures of NQC are very similar to C. NQC is not a general purpose language - there are many restrictions that stem from limitations of the standard RCX firmware.

This document describes how to use the command line version of the NQC compiler. For information about the NQC language and/or the RCX API refer to the *NQC Programmer's Guide*.

For up-to-date information and documentation for NQC, visit the NQC Web Site at

<http://www.enteract.com/~dbaum/nqc>

Usage

Invoking NQC without any argument will print the version number.

```
nqc version 2.2 b1 (built Jun 23 2000, 11:20:44)
Copyright (C) 1998-2000 David Baum. All Rights Reserved.
Usage error: try 'nqc -help' to display options
```

The `-help` option will print complete usage information.

```
nqc version 2.2 b1 (built Jun 23 2000, 11:20:44)
Copyright (C) 1998-2000 David Baum. All Rights Reserved.
Usage: nqc [options] [actions] [ - | filename ] [actions]
- : read from stdin instead of a source_file
Options:
-I: use NQC 1.x compatibility mode
-T<target>: target can be RCX, CM, Scout, or RCX2
-d: download program
-B: prevent the system file (rcx.nqh) from being included
-D<sym>[=<value>] : define macro <sym>
-E[<filename>] : write compiler errors to <filename> (or
stdout)
-I<spath>: search <spath> for include files
-L[<filename>] : generate code listing to <filename> (or
stdout)
-O<outfile>: specify output file
-S<portname>: specify serial port
```

```
-U<sym>: undefine macro <sym>
Actions:
-run: run current program
-pgm <number>: select program number
-datalog | -datalog_full: upload datalog
-near : set IR to near mode
-far : set IR to far mode
-watch <time> | now : set RCX time
-firmware <filename> : download firmware
-firmfast <filename> : download firmware at quad speed
-sleep <timeout> : set RCX sleep timeout
-msg <number> : send IR message to RCX
-raw <data> : format data as a packet and send to RCX
-remote <value> <repeat> : send a remote command to the RCX
-clear : erase all programs and datalog in RCX
-api : dump the standard api file (e.g. rcx.nqh) to stdout
-help : display command line options
```

The most typical use of NQC is to compile a source program and download it to the RCX:

```
nqc -d your_file
```

Source and Output Files

NQC will only accept a single filename for compilation. If `'-'` is used instead of a filename, `stdin` is compiled. If a filename ends in `.rcx`, then it is assumed to be an RCX image file (from a previous compile) and will be used as is (for downloading or listing).

NQC will automatically create an output file for you if all of the following conditions are met:

- You are compiling a file (not `stdin`)
- Neither `-d` or `-L` have been specified
- `-O` has not been used.

In this case the output file is named the same as the input file, but with an extension of `".rcx"` instead of `".nqc"`.

If the `-O` option is used and a source file (or `stdin`) is compiled, then the output file is generated even if `-d` or `-L` are used.

Preprocessor Options

NQC always searches the current directory for include files. Additional directories may be specified using either the NQC_INCLUDE environment variable or the `-I` option. Multiple `-I` options may be specified. Searching is done in the following order:

- Current Directory
- Directory specified by NQC_INCLUDE (if any)
- Directories specified by `-I` options (if any)

When using the `-I` option, a directory delimiter (`\` for Windows, `/` for Unix, `:` for MacOS) will be appended if necessary. For example, the following Windows command lines both add `C:\NQC` to the include search path:

```
ngc -IC: \NQC\ foo.ngc
ngc -IC:\NQC foo.ngc
```

The RCX API is defined in a special include stored within the compiler itself and is usually included automatically before any compilation. The `-n` option disables this feature and compiles the source file without the RCX API being defined. To list the contents of the RCX API, use the `-api` option:

```
ngc -api
```

Two other preprocessor options are available:

- Dsymbol*[=*value*] defines symbol in the preprocessor
- Usymbol* removes symbol's definition

Serial Ports

The default serial port used by NQC is system dependent:

- Windows: COM1
- Macintosh: modem port
- Linux: `/dev/ttyS0`

This default can be overridden either by setting the system environment variable `RCX_PORT`, or by using the `-S` option on the command line. If both are specified, the command line option takes precedence. For example, to use COM2 under Windows, a command line such as this might be used:

```
ngc -SCOM2 -d foo.ngc
```

Miscellaneous Options

The `-d` option can be used to automatically download the compiled file to the RCX.

Compiler error messages are normally reported to `stderr`. If you want to redirect these messages, use the `-E` option. If you specify a filename (e.g. `-Error_file`) the messages will be redirected into a file. If no filename is specified (e.g. `-E`) then the errors will be placed on `stdout` instead of `stderr`.

A program listing may be generated by using the `-L` option. If a filename is specified (e.g. `-Llist_file`), then the listing is written to the named file. Otherwise the listing is written to `stdout`.

The `-1` option make NQC compatible with version 1.3 of the NQC compiler. This includes some changes in syntax as well as the 1.3 APIs for the RCX.

The `-Target` option is used to specify the kind of target that NQC should compile for and/or download to. Target names are case insensitive and should be one of the following:

name	device
RCX	RCX
RCX2	RCX running 2.0 firmware
CM	CyberMaster
Scout	Scout

The default target is RCX. Defining the target has several effects:

- A compile time symbol is defined for the appropriate target (`__RCX_CM`, or `__SCOUT`).
- The number of available tasks, subroutines, and variables is determined by the target type.
- The API functions are somewhat different between targets. For example, only the RCX supports the Datalog features.
- Communication with the target differs slightly.

Actions

Actions look similar to options, but they have some subtle differences. In general, options setup things (such as a serial port) for later use, while actions cause something to happen. Actions are executed in the order that they appear on the command line. In addition, actions appearing before the source file happen before compilation, while actions after the source file happen after compilation. For historical reasons, downloading the compiled file is treated as an option and not an action.

For example, if you want to compile and download `foo.nqc` to program slot #2 and run it after downloading, you'd use the following command:

```
nqc -d -pgm 2 foo.nqc -run
```

Note that the `-d` option appears before any actions. The `-pgm` action must appear before the source file, otherwise the program change would happen after the program was downloaded. Finally, the `-run` action appears at the end of the line so that the program is started after download is complete

A complete list of appears below:

Action	Meaning
<code>-run</code>	Run the currently selected program.
<code>-pgm number</code>	Select a program number on the RCX.
<code>-datalog</code>	Upload the RCX's datalog and print it to stdout
<code>-datalog_full</code>	Same as <code>-datalog</code> , but with more verbose output

<code>-near</code>	Set the RCX's IR mode to short-range
<code>-far</code>	set the RCX's IR mode to long-range
<code>-watch time</code>	Set the RCX's clock to the specified time. If "now" is specified, then the host's current time is used.
<code>-firmware file</code>	Download a firmware file to the RCX.
<code>-sleep timeout</code>	Set the auto shutoff timeout (in minutes) for the RCX.
<code>-msg number</code>	Send a message to the RCX. This is useful if you write a program that allows external control via IR messages.
<code>-raw data</code>	Send an arbitrary message to the RCX and print the reply (if any) to stdout. The data should be a hexadecimal string, with no spaces, a zero-padded so that it is an even number of characters (although it may be an odd number of bytes). For example, to read the contents of variable 1, you could use <code>-raw 120001</code> . The bytecodes for raw messages can be found on web sites that document the RCX protocol.
<code>-clear</code>	Erase all programs and any datalog present in the RCX.
<code>-remote cmds repeat</code>	Send 'remote' commands (identical to those generated by the LEGO remote). <code>cmds</code> must be a four-nibble hex number which represents the bitwise OR of the remote commands. The repeat value should be a decimal integer indicating how many times the command packet should be sent.

Environment Variables

The following environment variables may be used for NQC:

`RCX_PORT` - If defined, specifies the default serial port to be used. The `-S` option will always override this setting.

`NQC_INCLUDE` - If defined, a single directory to search for include files. Any `-I` options on the command line are searched in addition to `NQC_INCLUDE`.

`NQC_OPTIONS` - If defined a series of options to be used in addition to anything present on the command line. These options are inserted between the command name and any command line arguments (thus they get processed before any command line actions). Individual arguments are delimited by whitespace unless the space is enclosed in double quotes.