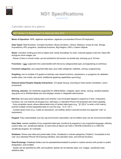
ND1 Specifications


pdf (1/2/12)

Mission Statement

ND1 aims to be a great type-rich everyday scientific/graphing calculator for anyone with high-end calculator needs. It is very programmable and very customizable.

ND1 is not easy to categorize. If you're evaluating it, here're the most important things you need to know:

- ND1 follows the HP RPN/RPL calculator model and largely operates like one (HP-28, HP-48, 49, 50g)
- ND1 doesn't yet have a CAS (Computer Algebra System), although it features some level of symbolic manipulation (incl. symbolic vectors and matrices) and does perform functions, such as BigNum arithmetic and prime number factorization, that are sometimes listed as CAS functionality
- ND1 is extremely programmable. It's the first JavaScript-programmable calculator and its approach to programmability is deep. RPL, as a language (but not the command set) is on the level of the hp 50g, and executes much faster than on this device. RPL+, a modern-day update to RPL, is also available
- ND1 has extensive, yet easy, sharing abilities. If you're a user, you will benefit from its programmability by being able to download great additions to your calculator

To evaluate ND1, we recommend you start with the Visual Guide to Capabilities. [This overview is now a bit dated.]

If you are a past or present user of high-end HP calculators, please see Comparison to HP Calculators.

Calculator Specs at a Glance

ND1 Version 1.5, MorphEngine 1.5, dated October 18th, 2013.

Modes of Entry and Operation: RPN, algebraic, mixed mode.

Data Types: Real Numbers, Fractions, Continued Fractions, Complex Numbers, Binary Numbers, Big Integers, Big Floats, Matrices (can contain any data type), Vectors/Arrays/Lists (can contain any data type), Strings, Expressions, RPL programs, JavaScript functions, URLs, TaggedObjects, JavaScript Objects.

Custom Data Types: first-class user-defined data types that leverage and extend the calculator, such as Chem.

Stack: unlimited, holding any data type with "pretty formatting" for each, including mixing of text and graphics: matrices appear multi-line, data URL strings as inline images, etc.
Shows 4-6 lines in normal mode; can be switched to full-screen via double-tap, showing up to 21 lines.
Can be emailed with email command.

Functions: >500; organized into customizable soft-menus by category/data type, and appearing on soft-keys.

Function categories: any supported data type, plus math categories, statistics, solving, programming.

Graphing: one to multiple 2-D graphs of arbitrary real-valued functions, expressions, or programs; *for statistics*: scatter plots, line charts, pie charts. Display of images. Line plots through point arrays. Parametric cartesian, integer plots. [Additional graphing capabilities upcoming.]

Gesture-based Graphic Display Interactions: 2-D graph tracing (drag), recording of trace points (tap); translation, zoom (two-finger drag, pinch); screenshot (two-finger tap).

Solving, calculus: *not directly supported*; for differentiation, integrals, taylor series, solving, variable isolation, the app goes out to *WolframAlpha.com* and displays results in integrated web-browser.
Numerical integration supported. A CAS option is upcoming.

Edit line: text input area holding data to be entered. Can be double-tapped to expand to 5-8 lines. Characters, functions, etc. are entered via keypad (incl. soft-keys), or standard iOS keyboard with insert capability.
  Auto-completes inputs, allows abbreviated entry of certain data types (e.g., “[[1,2[3,4” to enter a 2x2 matrix), permits multiple comma-separated data on one line (e.g., “3,’in’,’cm’”).
  Context-aware entry (e.g., inside an expression, *cos* soft-key doesn’t execute function but appends “cos(“ to expression).

Keypad: *Fully customizable*; any key cap and function association can be edited; keys can be removed and added. Keys can be enlarged via shake of the device.

User Data: *named* variables of any supported data type, functions & programs in any supported language, arbitrary custom data, incl. JavaScript objects; no size limits (a datum can hold a 100-KiB screenshot or a 1000-line JavaScript program, for example).

Database: Stores user data and preset data (such as *Units*, *Constants*) in named folders. Accessible to the user via calculator keys, internal functions, and a standard iOS browsing interface.

Data Exchange: Database folders can be uploaded/downloaded to preset or custom servers with private or public designation, and emailed.
  Assets can be specified by URL and localized; assets can be arbitrary data, incl. images, JavaScript code, statistics data.

Shared Data: Shared folders can be downloaded in-app with a single tap; a growing list is available. (Announcements and details for each folder can be found in the forums.)

Programmability: JavaScript (functions, extensions (via injections), custom data types), RPL+, HP’s RPL (currently excluding interactive commands) with the language (but not the command set) on the level of the hp 50g, RPL User Functions (mixing RPL and algebraic expressions), action recording (similar to keystroke recording), "natural math notation" (e.g., “f(x) = x^2”).
  RPL and JavaScript and "natural math notation" functions can call each other (i.e., they can be mixed and matched) with no special syntax required.
  Standard JavaScript frameworks are available (

```
jquery1.3.2
mootools1.2.2
prototype1.6.0.2
mochikit 1.4.2
canvasgraph
```

) and can be employed by user code.
  These are loaded dynamically on-demand via an API function (*window.require(URL)*). Arbitrary user .js code can be localized, or referenced remotely, and loaded through this mechanism.
  JavaScript has an extensive, documented API. RPL can leverage any of the built-in and user functions.
  Code (JavaScript, RPL+, RPL) can be written within the app, or imported/exported via *Data Exchange*.

JavaScript APIs: for writing user functions, accessing stack or graphics display (HTML5 canvas), extending/replacing built-in functions, making user functions built-in functions, providing custom data types, dynamically loading localized or remote user code.

Networking: networking functionality deployable from JavaScript in common ways; API support for synchronous

Networking: networking functionality deployable from JavaScript in common ways; API support for synchronous and async HttpXMLRequest, switching to a URL. Asset localization in database.

Unit conversions: *extensive, fully-customizable* unit conversion system. Supports 160 conversions of a variety of SI units and composites like energy, pressure, etc. All SI prefixes supported.
User-extensible and -editable, incl. ability to define new unit conversion categories.

Notable Data Type Details:
- Expressions can contain any combination of algebraic expression elements, functions (user, built-in), programs that return a value, and literal data types (incl. real numbers, vectors, matrices).
- Arrays (vectors and matrices) can contain any data types for their elements, incl. expressions (themselves containing variables, functions, etc.).

For more information, see the Data Types documentation.

3rd Party Credits

ND1 gives access to JavaScript frameworks and libraries, and uses some of them in portions of code:

The Prototype JavaScript framework, released under the MIT license. Copyright 2006-2007 Prototype Core Team.
The jQuery JavaScript framework, released under the MIT license. Copyright 2010 The jQuery Project.
The Mootools JavaScript framework, released under the MIT license. Copyright 2006-2009 Valerio Proietti.
The Mochikit JavaScript framework, released under the MIT license. Copyright © Mochi Media, Inc.

Graph charting is based on Alastair Tse's CanvasGraph library, released under the BSD license.
Copyright 2002-2009 Alastair Tse.
Text in the HTML5 Canvas object is based on Jim Studt's CanvasText, released into the public domain.
BigInt support is based on Tom Wu's BigInteger library. Copyright (c) 2003-2005 Tom Wu. All Rights Reserved.
Portions of Matrix support are based on James Coglan's Sylvester library, released under the MIT license.
Copyright (c) 2007 James Coglan.
Numerical integration is based on Future Team Aps's code, released under the Future Team Software License Agreement. Copyright (c) 2007-2009 Future Team Aps.
Portions of Fractions support are based on Erik Garrison's fraction class code, released under the MIT license.
Copyright (c) 2009 Erik Garrison.
Portions of BigFloat support are based on Jonas Raoni Soares Silva's BigNumber class code.
Copyright (c) Jonas Raoni Soares Silva.