Table of Contents

Contact Information

Primary Contact:

Steven Noyes

NoiseTECH@me.com

Users Guide:

http://www.noisetech-software.com/Documents/YASC/V2/UsersGuide.pdf

Function Guide:

http://www.noisetech-software.com/Documents/YASC/V2/AppendixA.pdf

Document: NT1000-11
Revision: B

Introduction

Yet Another Stupid Calculator (YASC) is a calculator designed for the iPhone that offers the following features:

- iOS 4 optimized to use fast application switching.
- A unique keyboard for entering Binary, Octal, Decimal and Hex.
- Four user definable layouts for a Default, Trigonometric, Statistical and Financial function layouts.
- Built around an RPN type of input with up to 4 registers visible at the same time.
- Each Stack Item can be a different data type or be in a different display base.
- Switching between keyboards utilizes a standard springboard type swipe gesture.
- Transparently works in any base. This includes all floating point operations. Yes you can see that π is $3.3M5Q3M2CQ_{27}$
- A configurable keyboard. All the basic operation keys can be reconfigured to any of about 100 function/operations.
- Work with signed or unsigned integers.
- Work with integer word sizes of 8, 16, 32 and 64 bits.
- Save any number of registers.
- Includes a single array register for working with statistical functions.

Overview

YASC, meaning Yet Another Stupid Calculator is a simple program used to explore different means of programming the iPhone OS. It tackles several design goals including standard gestures as well as defining new application defined gestures.

Also included in the program is a custom floating point input and output routine that allows working with any base from 2 to 36. For example, the value of π can be entered as:

$$11.00100100001111110110101010001000100001011010011_2$$

and take the *sin, cos* or even *ln* of that number. The answer will be returned as a floating point number representation in binary.

The following figure shows the default base 16 (or Radix 16) display of YASC:

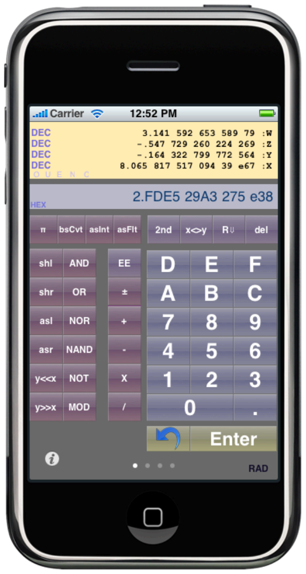

Carrier 12:52 PM
DEC 3.141 592 653 589 79 :W
DEC -.547 729 260 224 269 :Z
DEC -.164 322 799 772 564 :Y
DEC 8.065 817 517 094 39 e67 :X
2.FDE5 29A3 275 e38
HEX
π bsCvt asInt asFlt 2nd x<>y R⇩ del
shl AND EE D E F
shr OR ± A B C
asl NOR + 7 8 9
asr NAND - 4 5 6
y<<x NOT x 1 2 3
y>>x MOD / 0 .
Enter
RAD

Figure SC-1: Basic screen of YASC in right handed Hex mode.

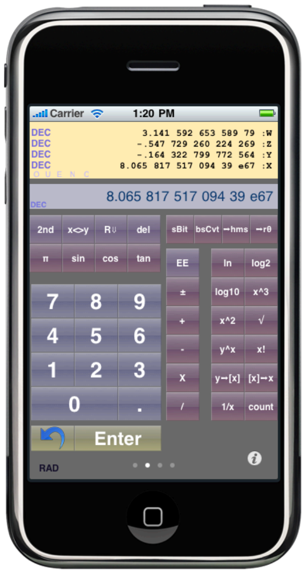

Carrier 1:20 PM
DEC 3.141 592 653 589 79 :W
DEC -.547 729 260 224 269 :Z
DEC -.164 322 799 772 564 :Y
DEC 8.065 817 517 094 39 e67 :X
8.065 817 517 094 39 e67
DEC
2nd x<>y R⇩ del sBit bsCvt →hms →rθ
π sin cos tan
EE ln log2
7 8 9 ± log10 x^3
4 5 6 + x^2 √
1 2 3 - y^x x!
0 . x y→[x] [x]→x
/ 1/x count
Enter
RAD

Figure SC-1: Basic screen of YASC in left handed Decimal mode.

User Interface Screen Transitions

YASC uses 4 unique keyboards, transitioned with a swipe motion, to allow user input. These 4 keyboards represent the primary screens used for interaction. There are also two other screens used for configuration. There is the "*info*" screen used to configure data type and display information and a keyboard configuration screen used to allow many of the function keys to be redefined.

At this point, YASC does not include a landscape mode for displaying data.

Calculator input screens

YASC transitions between the 4 base display keyboards shown in Figure CI-1 when either:

- a left/right swipe gesture is used within the movable keyboard area.
- The page control Icon (● ● ● ●) located at the bottom of the screen is used.

When this is done, the displayed base of the X Entry line will change to the selected screen. The bottom value on the stack, the "X:" line, will not change. Likewise, swiping to a different base does not end the editing session allowing additional values to be entered into the X Entry line if needed.

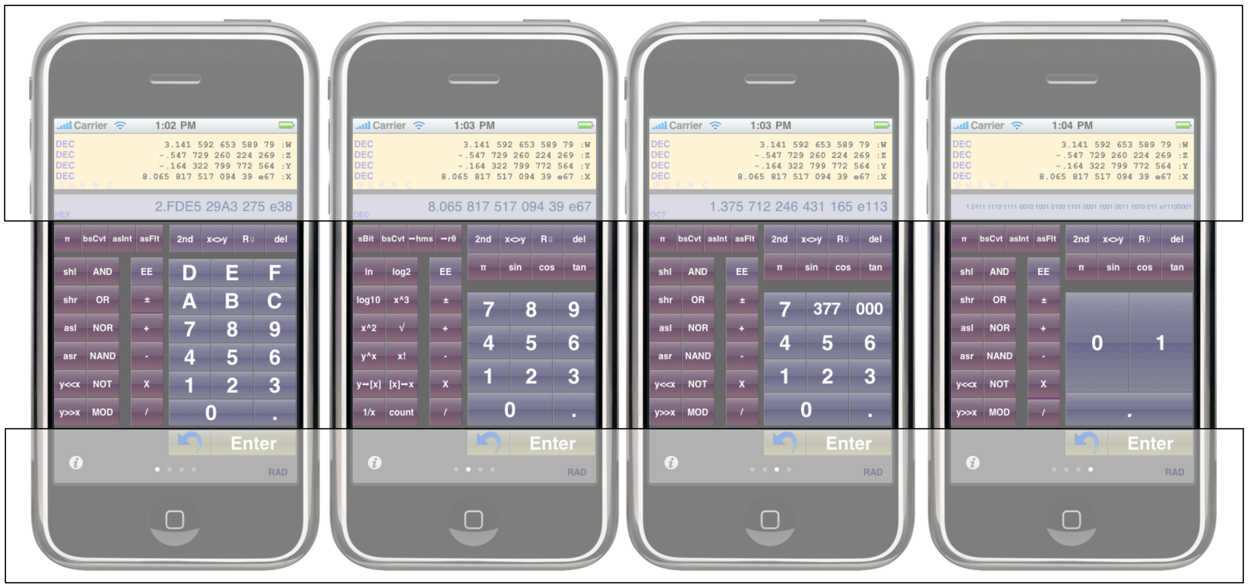
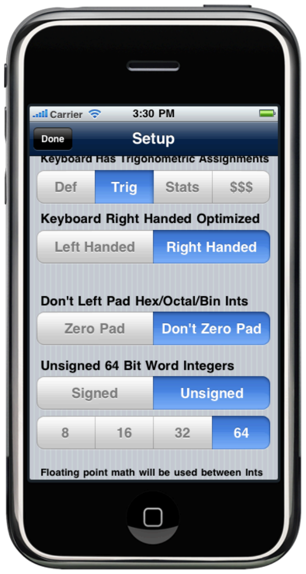

```
DEC      3.141 592 653 589 79 :W
DEC    -.547 729 260 224 269 :Z
DEC    -.164 322 799 772 564 :Y
DEC  8.065 817 517 094 39 e67 :X
```
HEX 2.FDE5 29A3 275 e38
DEC 8.065 817 517 094 39 e67
OCT 1.375 712 246 431 165 e113
π bsCvt asInt asFlt 2nd x<>y R↓ del
sBit bsCvt →hms →rθ
shl AND EE
shr OR ±
asl NOR +
asr NAND -
y<<x NOT x
y>>x MOD /
ln log2 log10 x^3 x^2 √ y^x x! y→[x] [x]→x 1/x count
π sin cos tan
D E F A B C 7 8 9 4 5 6 1 2 3 0 .
377 000
Enter
RAD

Figure CI-1: Image showing the 4 base input screens of HEX, DEC, OCT and BIN.

Datatype configure screen

It is possible to enter the Datatype Configuration screen by pressing the "ⓘ" button in the bottom right corner of the input screen Similar to configuration on many utility Apps (such as the weather app) pressing this will slip the screen to the Datatype Configuration Screen shown in Figure DC-1.


Carrier 3:30 PM
Done Setup

Keyboard Has Trigonometric Assignments
Def [✓ Trig] Stats $$$

Keyboard Right Handed Optimized
Left Handed [✓ Right Handed]

Don't Left Pad Hex/Octal/Bin Ints
Zero Pad [✓ Don't Zero Pad]

Unsigned 64 Bit Word Integers
Signed [✓ Unsigned]
8 16 32 [✓ 64]

Floating point math will be used between Ints

Figure DC-1: Basic datatype configuration screen.

Likewise, once any modification are done within this screen, the Done button (Done Setup) will return YASC to the normal keyboard screen.

Keyboard configuration screen

Many of the buttons can be reprogrammed as mention in the introduction. To perform this, press and hold any of the non-grayed out buttons in Figure KC-1 for about 1 second. After that time, a new screen will be peeled down as shown in Figure KC-2.

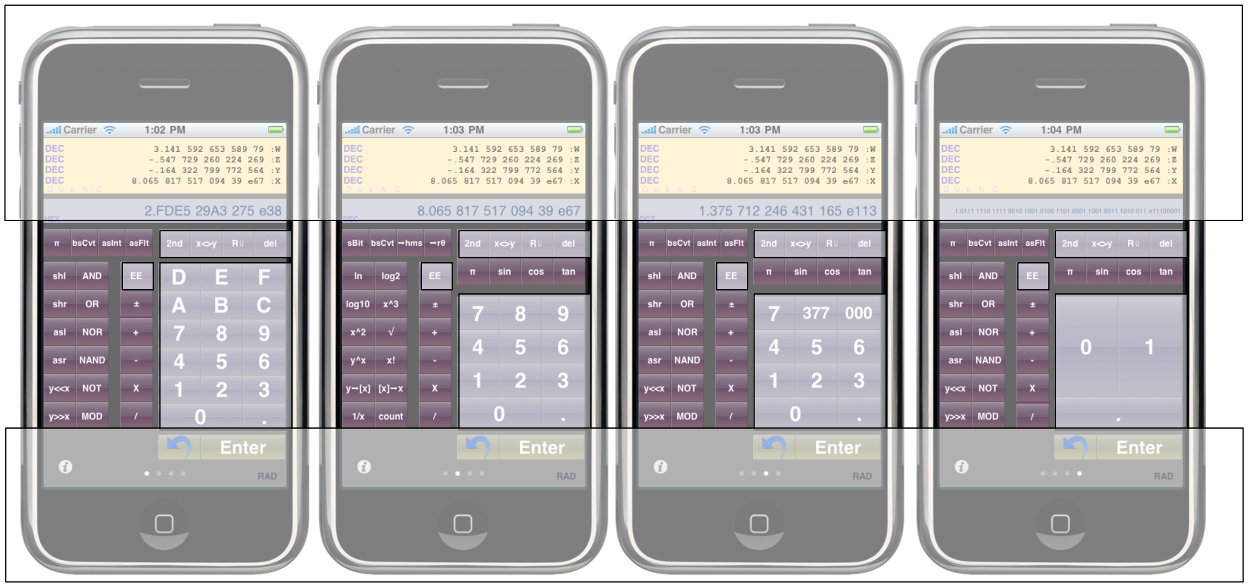


Figure KC-1: Keys that are allowed to be reprogrammed.

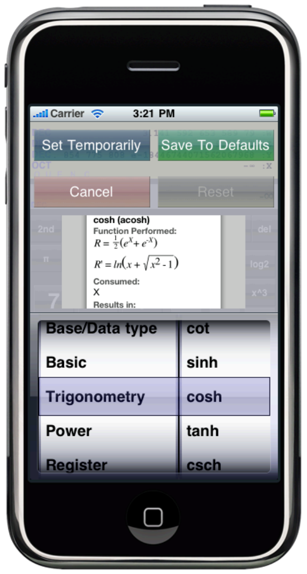


Figure KC-2: Display showing the keyboard reconfiguration screen.

To exit out of this screen, any of the 4 top buttons (Set Temporarily, Save To Defaults, Cancel, Reset) will return YASC to the main keyboard.

Documentation screen

Finally, there is the Documentation Screen that will display a copy of the manual's PDF for reference. It is displayed simply by rotating the iPhone/iPod Touch to a landscape mode while the calculator input screens are displayed. A sample of this screen is shown in Figure DS-1.

Y.A.S.C. Y.A.S.C Users Guide Page:6
Document: NT1000-11
Revision: A

Figure DS-1: Display showing the documentation screen.

Stack description

The YASC uses a basic RPN (Reverse Polish Notation) style of notation. It includes a stack with no real limits but only 10 items will be stored on the stack when the program is exited. The notation will typically be:

W: Top of stack

Z:

Y:

X: Bottom of stack

but some array operators (such as taking the standard deviation) will denote the stack as:

S7: Top of Stack

S6:

S5:

S4:

S3: Same as W:

S2: Same as Z:

S1: Same as Y:

S0: Bottom of stack, same as X:

For those unfamiliar with Reverse Polish Notation, the concept is rather simple and very natural once it is learned. On a classical calculator, equations are entered left to right using the parenthesis to force order of operation. For example

- 5.1 + (23.5 * 17.1) - (5.8 * 5%)

would be entered exactly as shown. However, a slightly more complex equation:

- sin(2 / 3 * π)

would be entered as:

- 2 / 3 * π = sin

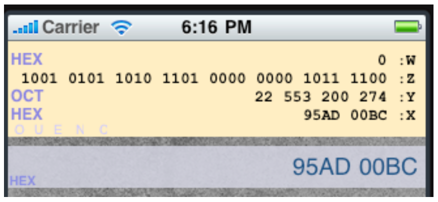
For an RPN calculator, items are pushed on to the stack and then later pulled down (or consumed) to perform an operation. The results are then pushed back onto the stack. For example, the previous 2 examples would have been entered as:

- 5.1 ***<Enter>*** 23.5 ***<Enter>*** 17.1 * + 5.8 ***<Enter>*** 5 % * -
- 2 ***<Enter>*** 3 / π * sin

It is interesting, that even with a classical calculator, the "sin" is entered last similar to the RPN format. What makes using an RPN unique on a large touch screen display is being able to see 4 stack items simultaneously. As a result, it is very obvious what a "+" will do as well as "ln"/, "sinh" or a host of other functions. Likewise, the stack also works as a great place to hold intermediate answers for comparisons.

The biggest advantage of an RPN calculator, however, is it does away with "C" and "AC". There is no need for the often confusing and ambiguous distinction of "C" and "AC".

Likewise, ability of YASC being able to handle a different display base on each line of the stack, makes it very simple to get different representations on the screen concurently as shown in Figure SD-1 that shows the same 32 bit, zero padded number in Hex, Octal and Binary.



```
HEX                                              0  :W
 1001 0101 1010 1101 0000 0000 1011 1100  :Z
OCT                              22 553 200 274  :Y
HEX                                  95AD 00BC  :X

                                     95AD 00BC
HEX
```

Figure KC-2: Display showing the keyboard reconfiguration screen.

Datatypes

YASC provides for several basic datatypes and they are allowed to be freely mixed with few exceptions (Refer to Appendix A for any restrictions). For example, a Floating Point number divided by an Integer will result in a floating point number. Integers mixed with Integers will result in integer results.

This might lead to some interesting results especially when dividing two numbers. When two base 10 integer numbers are divided:

$$\frac{32}{4}$$

The results are a perfect 8. However, a similar example such as:

$$\frac{33}{4}$$

will also result in 8 and not the expect 8.25 because the remainder is simply truncated with integer division. To perform a floating point value, simply change one of the values to floating point by adding a single decimal point:

$$\frac{33}{4.}$$

to get the result of 8.25 as YASC promotes different datatype to type of higher priorities.

Likewise, when adding, subtracting or multiplying numbers, it is also possible to actually overrun the number of bits set aside for the operation. Again, the extra bits are simply discarded and the “O” and “C” flags will be set to indicate an Overflow/Carry being set. Table DT-1 is provided to give the ranges of various data types.

Table DT-1: Table showing the basic datatypes and their priority and ranges.

Type	Priority	minimum	maximum
Floating Point	1 (highest)	Smallest:±1.797693e-308	Largest: ±1.797693e+308

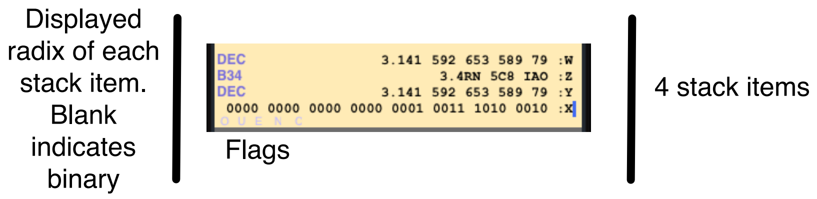
Type	Priority	minimum	maximum
64 bit Integer	2	Signed: -9,223,372,036,854,775,808 Unsigned: 0	Signed: 9,223,372,036,854,775,807 Unsigned: 18,446,744,073,709,551,615
32 bit Integer	3	Signed: -2,147,483,648 Unsigned: 0	Signed: 2,147,483,647 Unsigned: 4,294,967,295
16 bit Integer	4	Signed: -32768 Unsigned: 0	Signed: 32767 Unsigned: 65535
8 bit Integer	5 (lowest)	Signed: -128 Unsigned: 0	Signed: 127 Unsigned: 255

Main Screen Views

All software on an iPhone/iPod Touch is composed of views. A view may be a table, a “picker”, a button or a simple image. Views can be combine and reused in unique ways. YASC includes multiple views for entering and viewing the current state of the program.

Stack View

The Stack View is the primary view for examine data on the stack. It is shown in Figure SV-1. It includes a means of seeing the bottom 4 items on the stack, their associate base and a host of flags to indicate Overflow, Underflow and other flags.



```
DEC                  3.141 592 653 589 79 :W
B34                        3.4RN 5C8 IAO :Z
DEC                  3.141 592 653 589 79 :Y
 0000 0000 0000 0000 0001 0011 1010 0010 :X
```

Figure SV-1: Sample of the data on the Stack View.

YASC provides that the stack view can show 4 stack items with each item having a label of to the left indicating the base it is displayed in:

- Base 2 shows no indication.
- Base 8 shows “OCT”
- Base 10 shows “DEC”
- Base 16 shows “HEX”

- All other bases (from 3-36) not listed above as "B" with the base in Radix 10 format. eg; Base 22 would show as B22.

A new value moved from the "Y" register to the "X" register is "base converted" to match the existing display keyboard.

The Stack View also includes 5 additional flags to provide slightly additional information. These include the following flags:

• O - Overflow	($12.23e200^2$)
• U - Underflow	($12.23e-200^2$)
• E - Error	ln(0)
• N - Negative	-12
• C - Carry	8 >> 4

Stack Item Formatting

NOTE: The default decimal separator YASC uses is the "." and is based on the North American/India/Asia standard. HOWEVER, on startup, YASC will query the system and get locale information regarding group separators and decimals. The buttons will have the titles updated and the displays will output localized values. This discussion will assume the North American/India/Asia standard.

YASC provides simple rules based formatting to assist in reading the values on the iPhone's screen. Each base and datatype has subtle differences to aid in interpretation of the number. For example, a number such as:

Value entered	Formatted
110542.2154_{B10}	110,542.215 4
11.001001000011_{B2}	11.0010 0100 0011
$FEED.CAFEBEEF_{B16}$	FEED.CAFE BEEF
$78198.5231e123_{B10}$	78,198.523 1 e123 When multiplied by 1: 7.819 852 31 e127

Table SV-1: Examples of various numbers and how they would be represented in YASC.

Controlling displayed digits on floating point

YASC also provides a method to control how many decimals are displayed on the output of a floating point number. Because YASC can easily work on any base, it accomplishes this by specifying the number of significant BITS and not the number of significant digits. Like many things in YASC, an operation is used to actually set the number of significant BITS:

sBit

sBit, will take the value off of the stack (in X) and it must meet the following conditions:

- Be an integer
- Be equal to 0 OR (greater than 3 AND less than 49)

A value of 0 sets the minimum precision to 48 BITS and is the initial default value. Other values are then used to calculate the number of significant digits based on the current display base of a value. Table SV-2 shows some common settings for setting the number of significant digits. For most work for engineering, physics, chemistry and ... well... most things, a value between 12 and 24 bits is more than adequate. NOTE: This ***only*** impacts the displayed precision of a number and not the internal representation used for calculations.

	Radix			
bits	2	8	10	16
4	4	2	2	1
8	8	3	3	2
12	12	4	4	3
16	16	6	5	4
20	20	7	7	5
24	24	8	8	6
32	32	11	10	8
40	40	14	13	10
48	48	16	15	12

Table SV-2: Lookup table showing the relationship between significant BITS and significant digits with different bases.

Stack Item X Value in Sync with XEntry Field

YASC will keep the X Entry Field and the Stack item, X, in sync with their *values* always being the same. However, it is possible to have the Display Base be different. For example, only the X Entry Field will be updated when a swipe gesture is used to change the keyboard. The X Stack item, on the other hand, will remain unchanged until some other operation (such as adding a character or doing math) is performed. At that point, the X Stack Item will update to the new keyboard.

This has been designed to allow quick comparison and to allow the same number to be easily seen in two different bases simultaneously.

X Entry Field View

The XEntry Field is used to actually type in data to the Stack. It includes a blinking cursor when new text will be appended to the back of the entered value. It also contains a single indicator at the bottom left to indicate the base being displayed at that point in time.

With a single exception, the base of the X Entry Field will match the selected keyboard. The exception is the “bsCvt” function. as detailed in Appendix A, this operation will take the designated base in X and convert the displayed base of the value in X and return the value into X.

There are two modes of operation when data is entered into the X Entry field. The first is demonstrated with a blinking cursor on the right side of the text field as shown in figure XF-1. When this is blinking, the “del” and all numbers will simply append to the existing value.

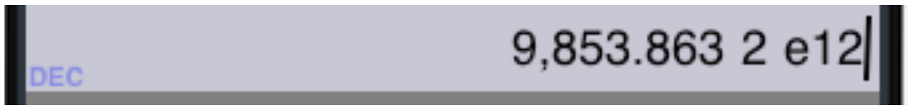


Figure XF-1: Sample of the XEntry Field in “DEC” with a blinking cursor.

The second is when there is no blinking cursor. The text will turn a slight blue color as shown in Figure XF-2. This is the result of performing any function or operation, by pressing the “Enter” key or tapping the field to bring up the “Copy” menu as shown in Figure XF-3. The next key will “push” a new value on the stack promoting the displayed value to “Y”.

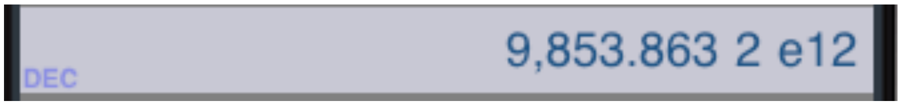


Figure XF-2: Sample of the XEntry Field in “DEC” with no blinking cursor.

The software **shall *[YASC-SRD-2480]*** display the system wide “Copy” menu as shown in Figure XF-3 when the XEntry Field is touched.

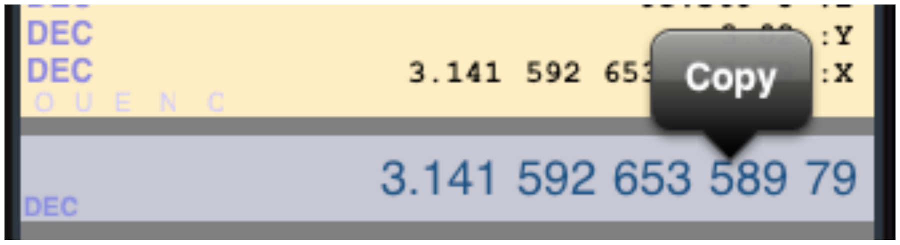


Figure XF-3: Sample of the XEntry Field in “DEC” with no blinking cursor.

Footer View
The footer view includes four controls/sub-views and shown in Figure FV-1. The first is a DEG/RAD control used in trigonometric calculations; the second an “Enter”; The third is an “ⓘ” key and the fourth is a “Page Control” icon (• • • •). The “ⓘ” and “• • • •” have already been explained in the User Interface Screen Transitions.

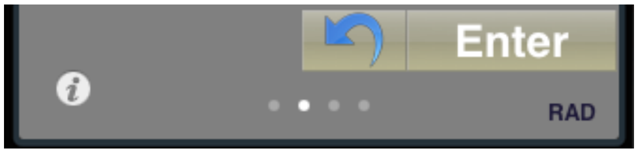


Figure FV-1: Footer View showing the DEG/RAD, Enter and ⓘ.

It is important to understand the units being used when executing trigonometric functions. The default is Radians and is based off of a unit circle of radius 1. To shift between Radians and Degree modes, the DEG/RAD button the bottom left of the screen is used. A simple tap creates a toggle. Appendix A details the changes made by pressing this.

Keyboard Views
As shown in Figure CI-1, YASC provides 4 unique views onto the keyboard. Each one holding the keys needed for the Radix the screen is designed for. Transitioning between them is done just like the Springboard used to hold applications on the iPhone main screen. There are several “classes” of buttons including:

- Numeric entry. These include 0-9, A-F, “.” and EE.
- Stack manipulation. These include x<>y, R⇓, clr, del
- 2nd key modifier. This is the 2nd key and is used to get to many of the alternate functions.
- Operation keys. These will include sin, cos, AND, NOR and nearly 100 other possible functions. These keys are configurable.

Numeric Entry
The four unique keyboards target a single “key” entry at a time. The “EE” key is actually encoded internally as a “;” and takes up one space in the input string. YASC simply appends the value to the end of the current X Entry Field when the following conditions are met:

- The key is within the range of 0-9, A-F, “.”.
- The key will not result in an overflow of the current datatype.
- The length of the field does not exceed the values in Table KV-1.

Table KV-1: Maximum length of XEntry Field based on base.

Base	Max Length
2	64
8	22
10	21
16	16

Either the EE key or the decimal key "." is used to change a number from Integer to Floating point

- The key is ".".
- The length of the field does not exceed the values in Table KV-1.

NOTE: The above is how floating point numbers are represented. The value "1432" is Integer and the value "1432." is floating point.

NOTE: There is no restriction on entering floating point based on Radix. The routines are designed to work in floating point in any base.

Stack manipulation
Like all RPN calculators, YASC includes a series of keys designed to manipulate the stack. There are no realistic limits (except memory) to the size of the stack in YASK. However there are some practical limitations the program does put limits on:

- Array items used for statistics are limited to 64 values.
- Only 10 items are saved from the stack when the program exits.

If you need to store more than 10 items on the stack when the program exits, store those values (up to 64) into the stack array register used for statistics.

There are three keys used to provide control of data on the stack and they are NOT configurable and they appear at the same place on each keyboard. In the upper right part of each key board are the keys shown in Figure KV-1a.

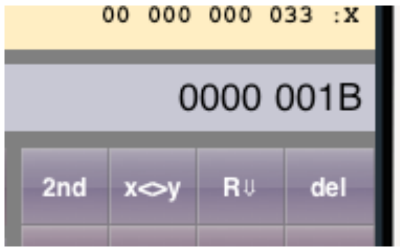


Figure KV-1a: figure showing the 3 stack manipulation keys. "2nd" is not a stack manipulation key.

The “R⇓” is used to throwaway the current X stack item and drop every other item down and functions as a clear. The “x<>y” swaps the X and the Y values on the stack and is popular to swap a divisor and a dividend.

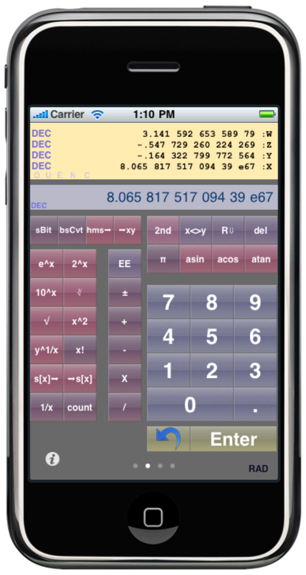
The “del” is used in normal editing and is used to delete the last entered key (del).

2nd key modifier
There is a single “2nd” found on the Decimal keyboard used to get to many alternate functions. For example:

Table KV-1: Examples of 2nd key functions.

norm	2nd
sin	asin
ln	e^x
x^2	√
√	x^2

The design of the 2nd key is around a shift lock. Once engaged, all keys impacted with a 2nd function will shift to a new color and the 2nd key will stay engaged until it is specifically disabled. Figure KV-1 shows an example of the keyboard with the 2nd key engaged.



```
DEC          3.141 592 653 589 79 :W
DEC        -.547 729 260 224 269 :Z
DEC        -.164 322 799 772 564 :Y
DEC      8.065 817 517 094 39 e67 :X
```

8.065 817 517 094 39 e67

Figure KV-1: Sample of keyboard with the 2nd key selected.

Operation keys

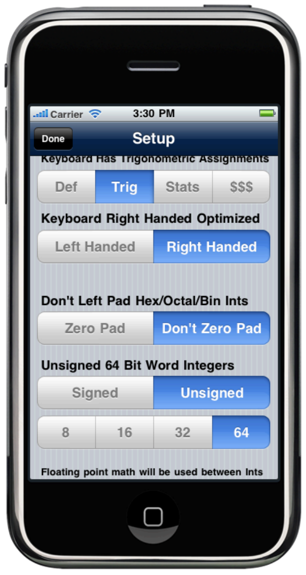
This brings us to the final set of keys, the operation keys. These keys are teh ones not blocked out in Figure KC-1. All of these keys:

- Will execute some function or operation based on their label. The description of the function is defined in Appendix A.
- Are fully configurable to any available function or operation. This is enabled when the key is held for > 2 seconds. When selected a screen will peel down and present a picker view with two components.

Supplemental Screen Views

There are currently 2 supplemental screens used for various configuration. The first is used to configure data types and displays. The second is used to modify the function of the operation keys.

Datatype Configuration Screen



Keyboard Has Trigonometric Assignments
Def [✓ Trig] Stats $$$

Keyboard Right Handed Optimized
Left Handed [✓ Right Handed]

Don't Left Pad Hex/Octal/Bin Ints
Zero Pad [✓ Don't Zero Pad]

Unsigned 64 Bit Word Integers
Signed [✓ Unsigned]

8 16 32 [✓ 64]

Floating point math will be used between Ints

Figure DC-1 (repeat): Basic datatype configuration screen

Word size and presentation control

As shown in Figure DC-1 (they are the same image), it is possible to configure the display and word size of Integer datatypes. These controls are:

- Allowing a “0” pad in-front of non-0 numbers. This will allow a 32 bit value of FEED$_{16}$ to be presented as “0000 FEED” making it easier to deal with some Integer numbers.
- Controlling if Integers will be signed or unsigned. If a 16 bit signed number of FFFF$_{16}$ is entered, the program will display and treat it as a “-1”.
- Simply controlling the word size of values. Values of 8, 16, 32 or 64 bit may be selected. NOTE: At this time, binary numbers are best used at values of 32 bit due to display space limitations. Longer text values will be truncated with an ellipsis (...) shown. It is possible to permanently assign a key or temporarily.

Auto Promotion of Integers to floating point.

Likewise, it is possible to configure if the calculator will do basic math like add, subtract, multiply and divide by first extending all integers to floating point. If the option is set to the default of NO, two integers will result in an integer answer. So performing the following operation:

120 + 120 = -16

when the word size is set to 8 bits and signed will result in -16. While this is correct for integer math, it may not be the answer that is expected. If the option to perform all operations in floating point is enabled, the result will return:

120 + 120 = 240.

Notice the following "." on the answer. YASC recognized the need to promote the answer to floating point and did so. This is also useful when doing division. Since a simple operation like:

21 / 22 = 0

while being correct for integers may not be the desired outcome in floating point of:

21 / 22 = .954 545 454 545 455

Keyboard Assignments
YASC provides for 4 keyboard layouts optimized for 4 different functionalities. These include:

Trigonometric layouts including:

- preprogrammed keys for radian and spherical coordinate conversions.
- preprogrammed keys for csc, sec and cot.
- preprogrammed keys for →h.ms, .hms+, .hms-, h.ms+ and h.ms-.

Statistical layouts including:

- preprogrammed keys for *e*
- preprogrammed keys for mean, stdev, yChsX, ∑S[], ∑S^2[].

Financial layouts including:

- preprogrammed keys for ∑S[].
- preprogrammed keys for fVal, pVal Δ%, %, x→%, X*%.

Keyboard layout for left handed or right handed use.
This is one of the nicest features of YASC (And that is saying allot). It provides to unique layouts for each keyboard. One is dedicated to left handed people and the other to right handed people. Basically, this is done by simply moving the numeric/stack edit keys from the right side (for right handed people) to the the left side (for left handed people) of the screen.

This allows left handed people to use their left hand for numeric data entry without covering up the function keys on the right side of the screen. Figure KL-1 shows the same screen with the two layouts.

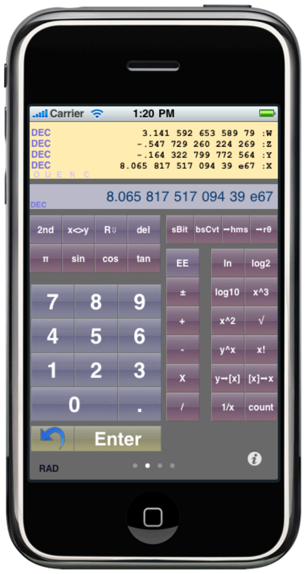
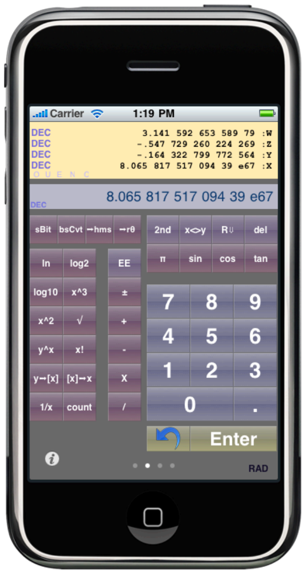



Figure KL-1: Sample of key position when configured for Left / Right handed.

Reprogramming operation keys
As shown in Figure SS-1, there is a screen used to reconfigure the actions of many of the keys. Once any of the keys in Figure KC-1 is held down for 2 seconds, the action of the key is canceled and the option is given to:

- Read the documentation on that key.
- Reconfigure the meaning of the key to any of the nearly 100 built in functions.

The Cancel is used to return YASC back into the normal keyboard views.

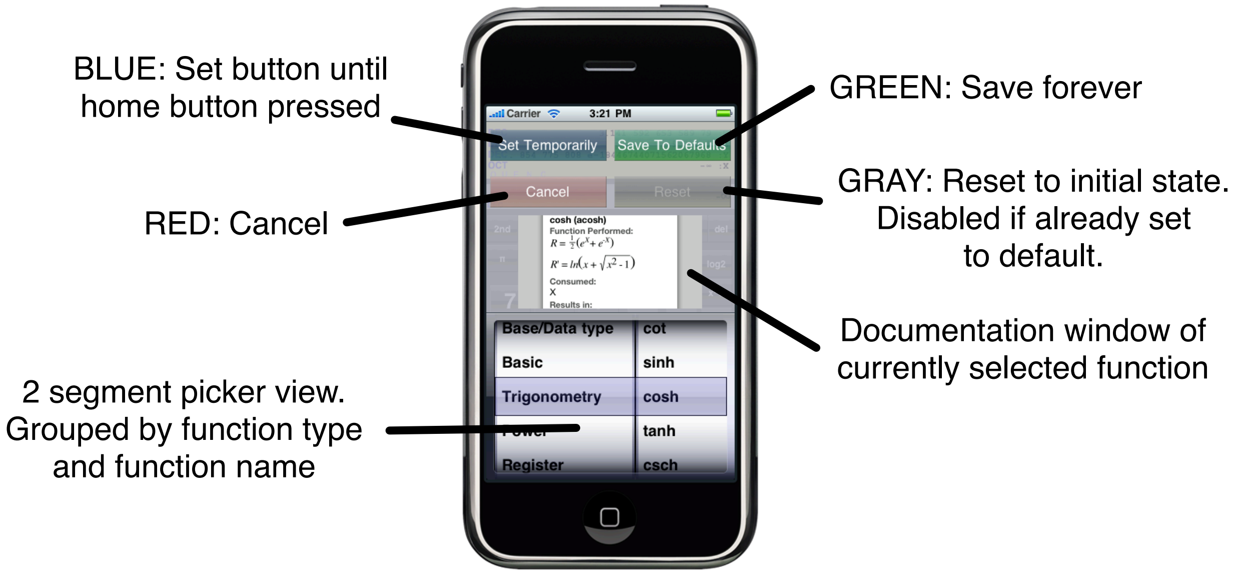

BLUE: Set button until home button pressed
GREEN: Save forever
RED: Cancel
GRAY: Reset to initial state. Disabled if already set to default.
Documentation window of currently selected function
2 segment picker view. Grouped by function type and function name
Set Temporarily
Save To Defaults
Cancel
Reset
Base/Data type
Basic
Trigonometry
Power
Register
cot
sinh
cosh
tanh
csch

Figure SS-1: All the elements of the keyboard reconfiguration/help screen.

Persistent data

As many know, the iPhone does not include the ability for 3rd party applications to multi-task. To compensate for this, it is desired for applications to save the state of the application when an home screen is pressed. This section simply details the items that will be saved to the Defaults. NOTE: All data is wrapped in a 32 bit CRC to verify the integrity of the data being stored.

- Currently selected keyboard screen. NOTE: This does not include configuration screens.
- Current value of the XEntry Field. This includes the cursor state.
- Stack items up to 10 items deep.
- All memory registers.
- The Array memory register.
- The configured state of key saved to the Standard User Defaults.
- The current selected default Integer Datatype.
- The current selected default Integer isSigned status.
- Keyboard Layout for left/right handed use.
- The current Zero Pad setting.
- The current RAD/DEG setting.
- Significant BITS for display.
- Undo stack.
- The current display position on the documentation screen.