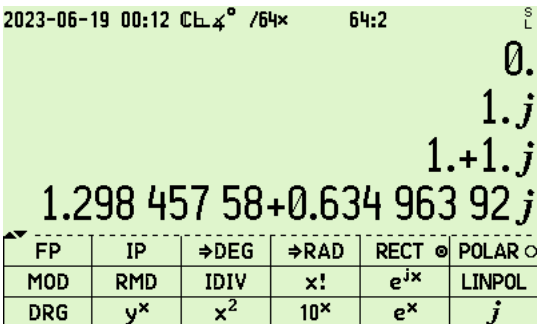
i-evolution

As I previously alluded to an upcoming i-evolution here, we expanded numerous aspects of complex operations on C47:

Complex re-arrangements started with the visual impact: the **i** appeared on the SIN button during the last re-arrangement of the template, but not much was further said about this. The new screen fonts match the bezel **i** nicely, blue on gSIN, and yes, you also need to install the new C47 fonts on your pc to use the simulator. The new **i**-key, as the name suggests, inputs in rectangular mode regardless of the global POLAR/RECT setting, eg. 1 i 2 ENTER for the obvious (1. + ix2.). Alternatively, **i** prior to number entry, results in the unity complex operator **i** (1.i) being placed on the stack. **i** also sits by default on your MyMenu F6, and can be used as one of the three ways to input complex numbers in addition to the existing 42S-flavoured COMPLEX on f[ENTER], or the WP43-flavoured CC in the CPX menu on F6. Of course, you can place any of **i**, CC or COMPLEX in MyMenu or assign to a specific key.

I mentioned "1.i" above; that was no accident: we added simple complex notation as the default. We found that the presentation of a complex number in the known dot or cross forms (0.+i∘0. and 0.+i×0) was not sufficient so we added a third display, doing two things: it ignores the real part if zero just like we write when hand writing math and it swaps the "i" to after the imaginary part. That is, 0.+i∘1. will be displayed as "1.i" by default. Or as I prefer, "1.j" with CPXj selected.

.



20230619-00121900.bmp (12.31 KiB) Viewed 214 times
CPXj set, with examples of the new complex presentation option

Furthermore, we updated the equation editor to be complex capable with a i/j handy, we upgraded the programmed sums for complex results. Σn and Πn (the Real not the integer versions) now respect the CPXRES setting, meaning if CPXRES is set (as per default), complex results are possible, for example: Σ√x from -1 to -10 in steps of -1 results in 22.46j. A nice worked example is to check the Taylor series

and to manually calculate SIN(1+j) or the sine of any other number for that matter, as per the following example which I already grouped in parentheses for the sum program TT. TT calculates two terms per cycle for convenience, not to bother with the plus and minus terms:

I programmed the generic formula as label TT (p47 file in the zipped release) (but I know if Didier sees my work he will rightly half the size again ... 😉), either way TT for two terms done as:

```
LBL “TT”
STO 01
2
+
STO 02
RCL 00
RCL 01
Y^X
RCL 01
x!
÷
RCL 00
RCL 02
Y^X
RCL 02
x!
÷
-
RTN
```

and with 1+j stored in R00, I ran n from 1 to 33 in steps of 4, to cover all terms x^1/1! to x^35/35! and thereby we show, using the sum function on C47 that the manual Taylor expansion for sin(1+j) up to …x^35/35! is:

```
1 ENTER 33 ENTER 4 Σn ‘TT’:
1.298457581415977294826042365807815 + 𝑖0.634963914784736108255082202991509781
(C47 Σn TT)
1.298457581415977294826042365807815620… + 0.634963914784736108255082202991509781… I
(WolframAlpha sin(1+i))
```

But I digress. Let’s stay with C47,

Since I started the WP43C fork years ago, the global complex display mode for the calculator frustrated me. POLAR/RECT (p/r) changed the mode globally for all stack registers simultaneously. This methodology

dates from the 42S, which did it the same with two visible stack registers with complex numbers. Now, with C47 complex types are displayed independently, each register tags its own p/r display state and when you STO or RCL, the p/r tag (hence p/r display mode) goes with. The global p/r setting is used to determine complex input via COMPLEX and CC and output only, similar in operation as the ADM is for the angle modes. Angle control of Real as well as Complex types is using DRG or the direct casting ➪DEG (et al) functions which change the register angle display of a complex number and also change the presentation of X to polar. Why not? You are changing the angle mode, so why not also change to polar display. Also, the blue >P and >R on COS and TAN will recognise a complex number in X and will change the p/r presentation directly instead of erroring on the complex type in X. It may sound a little complex (excuse the pun) but in fact is very intuitive.

This brings me to the traditional >R and >P commands (these are the blue labels on COS and TAN). If X & Y are both real types, it retains the old standard way (pioneered by 41C in 1979) of having coordinate conversion (in sequence of entry) from y, x to Θ, r in registers Y and X. This also is the default on C47, so it does what the 41C, 11C, 15C and 42S did.

LBL 'HP.RP'. Note this label, I am coming back to it.

This last point in the I-evo (>R and >P) brings us to the inconsistency issue which is not new. Let me start with the root of the problem starting with the 41C in 1979 which introduced P-R and R-P. Shortly after that, in 1981/2 the 11C/15C featured ->R and ->P and in 1988 the 42S featured the same functions now called ->REC and ->POL. The concept was clear: you place x in X and y in Y, or Re in X and Im in Y, and you convert to Polar with r in X, and Θ in Y, and vice versa. In a sequence reversed to how we are used to write.

HP however considered complex numbers for the first time on 15C in 1982. 15C offered a "natural way" (the preferred way (in the manual)) and an alternative (legacy, compatible) way of entering complex numbers into its twin stack. The preferred entry was using "I", which was the 'modern' natural input in the sequence Re Im, i.e. entry: Re ENTER Im f(I). This "natural" way as HP coined the term, was compatible with the future 42S complex number type. The conservative way though, still compatible with the old coordinates system, still reversed the input sequence, i.e. Im f[Re⇆Im] Re ENTER. When HP introduced the HP42S in 1988, the new complex type in a

single register stack also had their “natural” way of COMPLEX entry. This was the only way of entry, in the natural form Re ENTER Im f[COMPLEX], which resembled the natural hand-written sequence of real then imaginary components. We, the WP43S developers voted some 4 years ago and adopted this “natural” sequence for CC entry, see the Gitlab Issue.

Now that you had the introduction you should now be aware of the dilemma: On C47 we have the 42S complex entry which is entered as Re ENTER Im COMPLEX, or Re g(i) Im ENTER, or Re CC Im ENTER. At the same time, you have the blues on COS and TAN >R and >P which work on the 41C/11C/15C/42S reversed way, which is fully incompatible with said COMPLEX entry! As I said this is not new, but HP did not really push the conversation!

The final part of the i-evolution now offers a reversed >R >P system on C47, to change the coordinate system to the natural entry order. We do understand that this section may well be rather upsetting to staunch supporters of the old coordinate system, so this section from the label **LBL ‘HP.RP’** is under an option aptly called HP.RP, which is set by default and defaults to the old HP way. This option offers a method to interact with numbers between the real number P<>R system and the complex number entry system, i.e. you can interchangeably grab Real coordinates into a Complex type. Note that without changing this setting – this does not happen and all are as per HP examples.

Reference: From C47 release notification:
https://forum.swissmicros.com/viewtopic.php?f=2&t=3448&p=27634&hilit=I+evolution#p27634

Author Jaco Mostert