Notes on the back story of this letter:

This *6-page* letter is my reply to the *26-page* handwritten letter that *Melbourne PPC Chapter's* ***John McGechie*** sent to me dated *1980-08-05*. John's letter was hard to read because of his handwriting, but mine was written on both sides of poor-quality grid-paper sheets so we were even.

I began by briefly apologizing for my criticism of some *PPC* aspects; I didn't feel like apologizing at all but I valued John's good intentions and didn't want to needlessly upset him, so that was that. On a more positive note, I expressed my appreciation for his comments and suggestions about my **CP** program, and my intention to experiment with the **.END.** to make it work. I also included a short but detailed analysis of **CP**, explaining what it did and how.

I also mentioned his ***KA*** *#13/14* programs, but regretting that they did require the printer plugged in because they used the **BLDSPEC** function. I also commented on his ***b2*** routine, and my similar routine ***W*** (which I included with the letter,) followed by my comments on *RAM* tests and the problem with *normalization* on recall, mentioning too my experiments with accessing *microcode*.

Next, I commented on what the called *"NN catalogs" (flag 30 catalogs)*, including my list featuring the details of ***70*** such *"ghost"* functions, sent to *PPC* but unpublished thus far, showing how to generate *non-standard labels* using some of those functions in cahoots with status register ***Q***.

Answering one of his questions, I gave him permission to use my materials as he saw fit and expressed my interest in joining the *Melbourne PPC Chapter* as a full-fledged, paying member, as well as my interest in back copies of their *Newsletter*.

I then listed the materials included with the letter, namely: the list of *ghost* functions, the fastest routines to find the address of the statistics block and perform a complete decoding of register ***c***, my **HP-34C** program to non-iteratively find all roots real and/or complex of equations of degrees 2, 3, 4 by using exact formulas, my fast **HP-41C** ***Matrix Inversion*** program (up to 16x16), and the game ***Othello***, all of them duly sent to *PPC* but still unpublished at the time, which I resented.

The letter ends with two pages of my sleuthing experiments, including creating a *number* that when used as argument to the *numeric* function **[1/x]** results in the *text* "**BABY**" in the ***X*** register, experiments with the *ghost* functions **(0,51/55/60)** and their extremely *weird* behavior, assorted comments on **GTO 15** (local & short form) and on synthetic labels including *non-standard* characters.

I also discussed extensively (1 page) my interest in creating a ***CAT Assignments*** function and subsequent attempts to implement it. Additional comments include the creation a *Madrid Chapter*, scoops about the incoming release of the *Quad Memory Module* and tape storage for the *HP-41C*, praise of his ***AN*** routine and my 6-byte optimization of it plus my ideas for optimizing his ***b2*** routine, and my little programming tip for the ***Morse Code*** program which saved 50+ bytes by using *single-letter global labels*, easily created via three *ghost functions* for **LBL**, **GTO** and **XEQ**, respectively.

Valentin Albillo, 05-01-2022

J.E. McGechie

Philosophy Department
Monash University
Clayton , Victoria
AUSTRALIA 3168

Valentin Albillo (4747)
Padre Rubio, 61 - 2-C
Madrid 29
SPAIN

Dear John:

Very pleasant to receive your letter so soon ! It arrived home just 5 days after you sent it in. Excuse the poor quality of this paper, but I couldn't find any other. The reason for the 2-sided - writing is to reduce the postage cost (my last letter was worth about - 10 $ in stamps, a little fortune !)

You were right, do not deserve such criticisms from me, I apologize. I greatly appreciate your - comments & suggestions about my CP program. It seems that it would work if the right END is inserted in the right place. A little experimentation from my part will surely put the question to rest for a while.

I do not have the KA #13 & 14. It seems that a printer is required to run them. I do not have a printer. I may dispose of a printer occasionally, but do not have a printer of my own. So, any KA program that uses BLDSPEC is of little use to me.

Your "b2" program seems to be quite useful. I sent a similar routine to PPC, as an add-on to my already published programs INSERT & DECODE (cf. V7N4P28-29). This routine (W),included herein, is - only 29 bytes, and performs a bug 2-like operation. See instructions. Any copies of the different versions of KA are welcomed, of course !.You ask about the register in which the .END. is stored. Of course it is below R00 ! I include a short analysis of the CP program: LBL 01 is some version of Wickes' UNBLD ; all lines from 01 thru the first synthetic text line are used to find the address (decimal) of the top of program memory. The top is simply the first register below R00 (R-01, say). Then , Rc is altered so that this register becomes R00, and the hex code for .END. - (C0/00/2D) is stored there. Rc is restored, and execution halts.Why C0002D? Charles Close writes: "... One global address is reserved for the statement at the top of the program registers and that is hex C0/00 ... " (V7N3P8c) That's the reason for the C0/00, so the global linkage is irrelevant. There is no need to include the "curtain" address in the statement. It is a fixed C0/00.

About RAM tests: I thought also about the F7/FF/FF/.../FF, and I knew how to test RAM completely, but what I considered impossible is to test RAM (data,programs,KA) without disturbing their original contents. This is possible with data registers. Simply, recall the register, load the FF/.../FF, recallthe FF/.../FF, test to see if it is correct, then store the original contents back. But, with program registers, you can't do that: as soon as you recall , the contents are normalized, and it is not possible to restore the original program. Normalization strikes back again ! You mention a program (or two) from Graeme Dennes to test RAM. - Could you send me a copy ? Thank you.

About accessing microcode: it seems easy. In V7N3P21c, Jim de Arras writes: "... the first 12288 (ROM addresses) are for system ROM". If those addresses may be coded and used together with the ROM program of Wickes, it seems feasible to wake up in the system ROMs. I've tried, but, how does one distinguish microcode from conventional code ? I've been in zones where the program turned PRIVATE or ROM or both at the same time. Even tried to copy it to RAM using the COPY function. Most times, it simply resulted in a PRIVATE (private ROM ?) message, but one time I succeeded and something odd was copied down in RAM (from where ? I had no ROM plugged at all !) I couldn't analyze it, as a MEMORY LOST did take place, due to an unforgiving STO c.

Yes, you can be sure that I read very, very carefully every article submitted by Wickes. They are outstanding.

About NN catalogs: no, I don't agree. They are not normal catalogues with the read out alpha deranged. Here included is a list of ghost functions (submitted to PPC long ago), which features 70 entries. All of these functions appear in the NN catalogs, and many of them behave

differently that "normal" functions (see the byte jumper, for instance). If a function has a different name, and behaves differently, then it is just logical to consider it a different function altogether, and a catalog in which the function appears, cannot be considered a deranged normal catalog, but a new catalog. The table list the inputs for your KA program to assign the functions. I am sure that you also know about these functions, but here included is the table, anyway. You will also notice that I know about the "digit" functions, and uses of register Q to produce arbitrary LBL,GTO,& XEQ. The question is: the NN catalogues list much more than 70 functions. How are generated the rest ??? Any ideas ? Do you know anything about registers P & Q ? For instance, PRP may be loaded into program memory, but it results in NONEXISTENT if executed. If you load the name of the program into Q, will the printer print that program, then resume execution ? This is, the sequence: (alpha) PEPE (alpha), RCL M, STO Q, PRP (executed in PRGM mode), results in the program called PEPE being printed ?

There is no need to fill the prompt with HEX digits ! (with regard to finding a STO _ _ that would also generate STO M, etc).For instance, the $T+N IA _ _ _ , generated using 0,229 as inputs for the KA program,generates XEQ 00, 01, ...,99,00,01,A,B,...,M,N,...,e, by simply filling the prompt with an adequate number. For instance, filling the prompt with 117, generates a XEQ M (local!) both in PRGM or RUN modes. So, if, say a #$%)(_ _ _ is synthesyzed to represent STO, pressing 117 would generate a STO M both in RUN and PRGM. Why HEX digits, at all ?

Following with your letter, of course you may dispose of my material as you like ! I would greatly appreciate that you bring my programs, etc, to everyone interested, or publish it in your local newsletters. By the way, I know I live a little far from Australia, but, can I be considered as a member of the Melbourne Chapter (?) I would pay any necessary fee gladly. I am very interested in receiving your local news - letter, and if it is necessary to be a local member, that is what I want to become.

Before passing to technical topics, some words: I am interested in your word processor (I have 3 modules), as I tried to program such a thing, following the guidelines given by a number of BYTE about a text editing program written in BASIC. It is difficult, indeed, as the 41c lacks most string functions (LEN, for instance). A good idea for the ROM, indeed. I carefully followed your printer listings about the CP program experimentation. Incredible, indeed. Your patience must be as large as your kindness, to say the least ! It's terrific ! I am now studying all of it as carefully as I can, specially your "annotated" listings (KA,b2,etc)

Here included are the following:

-the list of ghost functions: name, generators, true function & notes
-the fastest routine to find the address of the statistic block: it executes in 4 seconds, uses no register,disturbs no flag, and provides the absolute address of the statictic block, the relative address, the absolute address of the curtain (R00), all at the same time, only 4 seconds. The optional add-on provides the absolute address of the final end, in just 2 seconds thus completely decoding the contents of Rc in 6 seconds, at most.
-the 34c program I forgot to include in my last letter. It solves any equation of degrees 2,3, or 4: finds all roots,whether real or complex, to 10 digit accuracy, within 20 seconds. (If 4-degree equation)
-A 41c program to invert matrices.(excuse me, but I am not very sure if I sent you this program included in my last letter. I believe that I sent - a "Systems of differential equations" solver, not a "matrix inverter") Inverts up to 16x16 matrices. Includes all required prompts for input & output. It is the fastest version I know of, much faster than the MATH pac program. It fits onto basic 41c. Uses synthetic functions to save registers and restore flags. A 16*16 matrix is inverted in over half an hour.
-The game of OTHELLO for the 41c: If you like games, you'll love this one. It includes full graphics. If you use a printer, it will print the board, etc. You can make a good use of it, not only for personal amusement, but also for shows and demos. Any improvements to the playing strategy are encouraged and welcomed. You'll need 3 RAMs to run this program. Printer is not required. No card reader or data cards needed.

I hope you will like some of this material. All of it was contributed to PPC some time ago. All of it remains unpublished, however.

Now, the technical subjects (it's just a joke!).

a) A challenge: use my "R" (INSERT) program (cf. V7N4P29)with the following inputs:

1 ENTER 264 ENTER 4.495914826 EEX 77 , XEQ "R" → -4.4959148 77

switch out of user mode, and press 1/X . You should get "BABY" . The question is: Why does inverting such a number causes an alpha result?(it is not the display of an alpha text, but an actual alpha text in X). It seems that several mathematical routines may give alpha results ! Why ? Is this any useful ? (the arguments is stored into last x, of course). (the SQRT behaves similarly)

b) if you analyze the table of ghost functions, you will notice some missing entries. The question is:

the inputs (for your KA program): 0 ENTER 51 ENTER key
0 ENTER 55 ENTER key
0 ENTER 60 ENTER key

what functions do assign ?

I've tried to study the functions generated by these assignments. Here is a brief resume of my discoveries:

0 ENTER 51 assigned to -15. The sequence (in PRGM) fE, 7, backarrow, fE caused a text composed of 2 boxstars to appear as a line in program memory, alpha mode is set,and the 41c starts to program itself with Ø,until it PACKS and TRY AGAIN. Repeating the sequence causes the same , except no auto-programming. Repeating once more causes a crash, with a PRIVATE message in the display. Using 8, instead of 7, causes DATA ERROR to appear in PRGM mode. Using 9, YES or NO appeared in PRGM mode.

In another try, the sequence 7,backarrow,fE, caused the 7 to be restored (though I had just delete it), and placed it one line back.Using SF,backarr, fE, caused RAD mode (in PRGM) and crash. Using STO,backarr,fE, then any number, caused RAD,SF 03 (annunciators, not instructions), and no crash.

0 ENTER 55 assigned to 24. It didn't do anything. Then , pressing a series of keystrokes (including keys 24 and -15) caused an uninterrupted scroll of non-standard characters across the display, from right to left. They scrolled at the normal rate, repeating periodically. As soon as I pressed any key to stop the scroll, they began to scroll at super-speed, blurring the display. This super fast scroll locked the keyboard.

0 ENTER 60 it always gave OUT OF RANGE when executed in PRGM mode, or nothing, depending on the previous keystroke. Further experimentation showed that pressed after any key, it caused nothing, whereas, if the previous key was numeric, OUT OF RANGE was displayed (in PRGM mode). Executed in RUN mode, it changed the FIX or SCI mode. It also bring some unknown number to the X register, while preserving the previous contents of x in last x. It altered the x register even if executed in PRGM mode. Some experimentation showed that changing flags, the recalled number changed also, and , after considerable test, it was clear that the number recalled into X by pressing the key was exactly twice the contents of register d !!! This is, if register d (the flag register) contains FF/20/03/31/09/80/00 then pressing the ass. key, causes F1/E4/00/66/21/30/01 to appear in the X register. This number is exactly 2*Rd !!! Holding the key pressed in RUN mode, causes the display to flash

Any ideas ? Have you tried something ?

c) The synthetic GTO 15 (local & short form) is a true NOP. It searchs not the LBL 15. It simply does nothing.

d) Synthetic labels (GTO,XEQ) composed of non-standard characters (or non-allowed characters, such as .,:) are easily created using REG Q and the generic alpha LBL,GTO,XEQ, (see ghost functions table). They do work, if accessed in PRGM or indirectly, by GTO IND #, XEQ IND #. Labels composed of more than 7 characters (and GTO,XEQ) are easily created using

a bug-2 like procedure. However, although they appear in the catalogs, a, say, GTO"HEWLETT-PACKARD" does not search a LBL"HEWLETT-PACKARD", and if both exist in program memory, executing the GTO causes NONEXISTENT to appear. So GTO,XEQ of more than 7 alpha characters do not find its LBL, even if the LBL exists.

e) I am very interested in programming a CAT ASS function. Its purpose is to "catalog" all existing assignments for those users without a printer (like me).

Ideally, it should work as follows: XEQ "CATA" → KEY 11: 145,124
→ KEY-13; 144,124
... ...
→ KEY 81: 159,159
→ NO MORE KEYS

where the first line, for instance, means that DEC(145),DEC(124) is assigned to key 11. This is, STO b is assigned to the sigma plus key. It is a very comfortable output: simply have the HEX table near you, and 145,124 becomes STO b at once. Besides, 145 and 124 are the input for the KA program to create the function, even if the function has no known name.

The program was structured so: first, change Rc so that the assignment registers become accessible using STO,RCL of normal registers (very much like your technique of creating assignments by storing directly into the assignment registers). Then , RCL an assignment register. Its contents are normalized from F0/... to 10/..., but the rest remains intact (apart from the F0 to 10). Then, suitable decoding routines are used to extract the pertinent information. Of course, Rc is restored to its previous status as soon as the RCL has been performed.

It looks nice. But it does not work ! It works, because it displays the KEY 11: 145,124, etc. But it does not work because if you turn off the machine, it is very likely that you would not be able to turn it on again. It crashes. Battery pack removal does not work. Only module pulling saves the day. There is something worse, yet: even if the machine does not crash, all assignments are removed by the routine, and changed to some odd XROM. To make things clear, the critical part of the process is recalling the assignment register. I use this sequence:

```
LBL "CATA"      text 1 is F5/01/69/0C/00/BF
ENTER           ideally, the routine leaves the contents (norma-
text 1          lized) of the specified assignment register in X,
ASTO X          while leaving Rc restored to its previous status.
X() c           To recall the contents of the 1st ass. reg, for
RCL IND Y       instance, simply: 0 , XEQ "CATA". For the next
X()Y            register, simply: 1, XEQ "CATA".
X() c           It should work! But, for unknown reasons, as soon
X()Y            as the RCL is executed, all assignments are lost.
RTN             Why ? The same is true for RCL 00, or RCL IND, or
                ARCL, VIEW, etc. Only STO seems to leave every-
```

thing unaffected. I have not tested other 41c's. Does this routine work in your machine ? If not, do you know a cure ? Another totally different way of doing the same ? Is a CAT ASS possible ?

That's all. Hope you will have some fun with all this stuff. I was very delighted with your letter, and I want you to be the same about mine. Hoping news & comments from you,

Sincerely

So long !

(4747)

PD: feel free to use the material as you like. Only thing I ask for is credit

I didn't see any comments from you about my method to break PRIVATE (point 5 of the STO b, RCL b article). Do you know of a better method ? Or is my method the best possible ?

FURTHER NOTES (13 de agosto de 1980)

Curiously enough, your short letter arrived one day after your long letter did (although it was sent 2 days earlier). Very good idea to submit the July newsletter, thank you. No, I am not thinking about trying to establish a Madrid chapter, as it is impossible nowadays. I am unable to find other members, apart from me and a friend of mine. No other PPC members do live in Madrid, as far as I know, and you'll agree that a Chapter of one (my friendhas retired from PPC activity) is not a Chapter at all. So much for the Madrid Chapter !

About Double density RAMs: I have 3 modules (single density), and can easily get the 4th one. Besides, I know some people who have both the tools and the knowledge to modify single RAMs to DD RAMs. These people have already converted several RAMs, very successfully. The only reason that halts me from converting my RAMs to DD RAMs (at no charge at all) is the fact that HP is going to release quad-modules within a year. They will be like a normal RAM, but not double density: they will be double double density: the full power of 4 RAMs taking only one port ! Its price is expected to be exactly the same as 4 normal RAMs. Of course, my purpose is to buy the quad-RAM as soon as it is available. So, it will be needed to sell my normal single density RAMs. That is fairly easy as long as they remain unaltered. That's the reason I have to not modify my RAMs: they will be harder to sell if modified.

Another good new: a telex was received from Corvallis, to announce the new add-on to the 41c system: a data cartridge drive. It will use the same type of cartridges as the HP-85A computer. It will store as much as 200 K of either data or program material,in just a single cartridge. Additional functions include linking programs under program control. This would allow to run indefinitely long programs (up to 200 k!),because one can call up another, and so on. A good new, isn't it ? Now the bad news: its price will be about the same as the printer. The search speed, etc, are the same that those of the HP-85 cartridge.

Now, technical subjects: your routine AN is amazing ! You were right, it runs in about 1 second, the display is perfectly legible as soon as you get used to the :,;,?, etc, and its very short lenght makes it incredibly good for the ROM. It easily beats Wickes' DECODE program (not to mention my own DECODE).I think every PPC member should have and use this routine. After studying carefully the routine, I made an improvement. This simple improvement saves 6 bytes, and runs faster as well. Nothing is modified or changed, just improved.

YOUR VERSION	MY VERSION
01 LBL"AN"	01 LBL"AN"
... ...	... ...
06 XEQ 01	06 XEQ 01
07 "*****"	07 "⊢**"
08 ARCL Y	08 X() N
09 ASTO Y	09 CLA
10 "**"	10 STO M
11 ARCL Y	11 ASTO Y
12 ASHF	12 CLA
13 ASTO Y	... ...
14 CLA	30 END
... ...	
32 END	

This simple modification saves 2 lines, 6 bytes, and runs slightly faster, at no cost

I have been also analyzing your "little b2" routine. I think it may be made shorter and possibly faster reprogramming it another way. the basic idea is as follows: the absolute address (in decimal form) of the register in which the NNN is to be stored is needed. This can be extracted using a procedure very similar to that used in my Byte counter (cf. V7N5P17) to convert two contents of Rb to decimal numbers. It used some sort of UNBLD. The decimal address is separated of the byte number using MOD instead of clearing flags and using FRC. Once the desired address in decimal form is ready, Rc is altered, so that the required register is accesible by normal STO. The store operation is performed by a STO IND instruction. This is exactly the opposite of your technique: you have a variable contents of Rc (depend upon the register), but store in a fixed register (STO 00), while my "program" would have a fixed Rc (loaded as a synthetic text line),although altered to move the curtain but will store in a variable address (STO IND #). All in all, my technique should be faster, as all CF, X()d, etc are not used at all. I'll try to convert these guidelines to an actual working routine, and let you know. You may try also, if you want.

A little programming tip

Several programs have appeared in the PPC journal requiring the use of single letter alpha labels indirectly. For instance, the Morse Code program (cf. V7 N2 P51) uses LBL"AA", LBL"BB",...,,LBL"ZZ",...,etc, for use indirectly using XEQ IND.Now, what's the - reason to duplicate letters, say, why LBL"PP" instead of LBL"P" ? Simply: local labels A,B,...,J cannot be accessed indirectly. But LBL"AA" can.

John Kennedy (918) proposes this as a solution, in V7N3P5, together with the following routine:

```
CLA
ARCL 01
ARCL 01
ASTO X
XEQ IND X
```

to overcome the problem. I propose exactly the opposite: instead of duplicate all letters, make all of them single-letter. This saves many bytes. For instance, the Morse Code program is shortened by 50+ bytes.

To avoid the NONEXISTENT message when A is stored in a register #, and XEQ IND # is attempted, we just need a LBL"A" (not local A, but alpha A). The same applies to B,...,J. This labels are easily created using a single assignment: its input for the KA program to generate it are: 0 ENT 205 ENT key. It appears as a $\overline{\ \ }$ if pressed. To generate a, say, LBL"A", do as follows: (in PRGM mode)

-press XEQ A (XEQ A is loaded as a line in the program)
-press assigned key in user mode
-fill the prompt using any digits, say 01: a LBL"A" is loaded into program memory.
-backstep and delete the XEQ A

It is easy to see why it works. The dummy XEQ A loads an A into the Q register. The a 00 takes its argument from register Q, thus becoming the LBL"A".

The GTO"A", XEQ"A", may be generated the same way, using functions (0,29) and (0,30) respectively, and following the same procedure. The label "A" may be executed if called in program mode by a GTO"A" or XEQ"A", or indirectly. In run mode, may be called using GTO IND, or XEQ IND, with the A loaded in the register used indirectly.

Yours,

(the enclosed articles and routines have been cut to their original PPC size of 14 cm. wide. This lowers weight, and helps to keep post cost low. However, if you prefer their original size, with blank margins, tell me so, and I will not cut them)

I'm also interested about back issues of your local newsletter !