Notes on the back story of this letter:

I sent this *17-page* letter to ***John McGechie*** just a couple of months before my departure to comply with my mandatory *Military Service*, which would mean I'd stay far from home for a looong time and unable to contribute any materials for the duration, so I took this opportunity to include a new batch and share some comments re the newly released ***SHARP PC-1211*** *Pocket Computer aka* ***TRS-80*** *Pocket Computer*.

The letter begins with my telling him about my many activities and among them my attempt to create a *Madrid Chapter of PPC* by preaching the *HP RPN* evangelion to interested people so that they would join me and the Chapter would be up and running *ASAP*. I then told him about the various people who'd sent me letters (Jakub Tatarkiewicz, Keith Jarett, Tom Cadwallader, John Dearing, etc.) and what they told me in them, plus some comments on the recent materials we both sent each other.

Next, I comment on a number of *HP* products rumoured to be released soon, including details on the ***"Kangaroo"*** (which was indeed released as the ***HP-75***) and new peripherals for the ***HP-41***, among them the improved printer (released), some *"universal interface"* (released as ***HP-IL***), and a tape drive (released as an *HP-IL* peripheral), so the rumours I heard were correct for the most part.

Now there follows a *6-page* section where I comment very extensively and enthusiastically on the newly released ***SHARP PC-1211*** *Pocket Computer*, with full details and many comparisons vs. the ***HP-41***, including relevant examples throughout.

Finally I include the following materials for their possible publication in some issue of the *Melbourne Chapter's Technical Notes* magazine, namely:

(1) ***O2***, a new version of *Othello* featuring 2 levels of play, *Level 1* playing exactly the same as my original *Othello* program, and *Level 2* featuring better play at the expense of slower running times. The code is improved and includes some additional features while still fitting in 672 bytes and optionally using the printer to print the board, moves and all messages.

(2) ***SEED***, a quasi-truly *random number generator (RNG)*, which takes no input and generates a *seed (0 <= seed < 1)* to be used to initialize another *RNG*, thus no need to ask the user for an initial seed. It uses a number of *synthetics* and normal techniques to ensure that different seeds are generated every time it's called, even within a loop. It's not intended to be used as *the* RNG for the program but just for obtaining a seed to initialize the one used.

Last but not least, the letter also included some postcards featuring locations in Madrid, as John had previously sent me one featuring a view of Melbourne and I wanted very much to reciprocate.

Valentin Albillo, 13-06-2022

John McGechie
Philosophy Department
Monash University
Clayton, Victoria 3168
AUSTRALIA

Valentin Albillo
Padre Rubio,61- 2ºC
Madrid 29
SPAIN

april, 10 - 1981

Dear John:

As you may see, a long period of time has elapsed since my last letter, dated december 12. The keyword is time. Time, time! It's horrible the absolute lack of free time I have suffered these last months. In the mornings, I stayed at the office, doing everything from selling calculator and computer-related products, to (more often), specialized software (technical programs, extremely complex: one of them is so large, that it has been partitioned in 9 parts, 400+ bytes each, and data used by the program is packed in the registers, allowing matrices as large as 80x80 to be treated -sparse matrices, of course-). During the afternoons, I acted as a particular professor, teaching mathematics to several people. I also dedicated the afternoons, after the classes, to write programs for some customers that asked for a particular, personal, topic, and wanted someone to write the program for them. I earn a lot of money this way, but it is quite time-consuming. Other personal problems added up to shorten my free time even more. For instance, I recruited many new members for PPC: I meet some people, all of them own a 41c. Then I introduced them to the magic and mystery of 41c synthetic (and non-synthetic) programming: in fact, I almost opened a "third eye" to each and every of them. Then, I convinced them to subscribe to PPC CJ. Now there are at least 7 members of PPC in Madrid, and probably more, because every PPC member acts as a nucleus of crystallization that atracts other people towards PPC.

The purpose of all this was to create a Madrid Chapter of PPC. I planned meetings with all those people, and even some type of publication. Very nice, isn't it ? But can't be done ! Simply because I cannot dedicate any time at all to the projected Chapter: I must fulfill my Military Service this year, beginning late May or June. That's 18 full months out of home, far from my ordinary activities. I cannot create a Madrid Chapter, then let it die. I am now beginning to prevent all my friends of my next departure.

Well, I hope this prologue explains my silence . I have received letters from several people: Jakub Tatarkiewicz, which told me some problems he had experienced while trying bug 9. As far as I can tell, he simply misunderstood the explanations (perhaps you edited too much the article). I sent him a photocopy of the full article, for him to check. Keith Jarett was kind enough to answer one of my letters regarding the PPC Custom ROM. He was also responsible for my program CHECKEPS being published in PPC CJ Jan-Feb, for which I am greatly indebted to him. Tom Cadwallader sent some interesting information about synthetic XROMs (pseudo-XROMS as well), and, as it's usual, without the slightest note or explanation. John Dearing asked for my written permission to use some of my routines-tips appeared in PPC CJ, to be published in a book which is for the 41c/cV the same as the "Better Programming" was for the 67/97. I like the idea very much, and firmly encourage all PPC members to send their inputs to John, so that this book can be as good and complete as we all need and want. I myself sent him 6 pages (or more, can't remember now) full of tips, little routines, utilities, etc. I also received a letter on the PPC Custom ROM: it seems that 4 routines in the ROM will be microcode. That's fantastic !! We'll have 4 new functions in our beloved calculator !! They include non-normalized extended STO,RCL and programmable TONE and PAUSE. The non-normalized RCL will be extremely useful. You just can't imagine the incredible routines which may be programmed using this function.

By the way, I have not yet received the PPC CJ of March. (But RN was kind enough to send me not one but three (!?) issues of the October-80 number of PPC CJ , which was missing). I also received PPT TN V5, and this was very comforting to me, as I finally had a confirmation that my last letter (incredibly expensive to send it) had arrived

(though in a very deplorable condition) safely. As it seems none of its contents get lost. The 34c book was brand-new when I sent it. I hope it arrived too in a good state. Now, as you can see, this letter has traveled enclosed in a much tougher envelope, surrounded by cello tape, to - avoid open edges and the like. Check that the enclosed mag cards are - still in place.

As I was commenting, PPC TN V5 arrived safely. I have not yet received PPC TN V6: did you send it ? If you did, it is quite obvious that it was lost. Send me a copy, will you? As I suppose, V6 shall include very interesting articles on microcode, ROM 0, etc, and I am anxious to have a look at it. You can continue to send PPC TN to my address, even if I have to go to some other place during my Military Service, because my relatives will take care of forwarding the issues of both PPC CJ and PPC TN to my temporary address, so that I can be as informed and "on the line" as possible. I do not plan to stay quiet while I am on the Army: on the other side, it can be one of my best periods since my only occupation will be the Service itself and my dearest - hobby: programming!

I received several letters from you, containing pre-production articles from PPC TN V5 and V6. If you sent something else, it was lost, no doubt. On the other side, I have sent nothing since my letter dated december 12, except a Christmas card. Did you receive it?

The news here are not particularly amazing: just rumours on new HP products:

-the famous "Kangaroo": it is not very clear to me whether it may be released into the market soon or not. Some sources point that it may be sold in Spain as soon as in October, but I do not believe this to be possible. Not until 82, at least. Besides, the only things known about the machine are its pocket size, its BASIC language, and its 9K of RAM, not to mention rumours about video interfaces, and the like. I know that you cannot publish information about future products in PPC TN, so I ask you to give me your impressions on the subject in a future letter.

-new peripherals for the 41c/cV: this is a never-ending topic: Last news are: (just rumours):

-improved printer/plotter: features presumably larger buffer plotting capabilities, programmable paper feed, allowing real dot-by-dot graphics to be produced, and capable of printing any program in RAM as barcode.

-universal interface: will allow the 41c to control any device connected to its microprocessor via the universal interface (IEEE ,perhaps?). This can be useful to:
-use almost any commercial printer with the 41c
-interface the calculator to a standard TV set
-interface the 41c to a standard cassette tape-rec.
-use the 41c as a low-cost controller

-tape drive: ala HP-85, should allow storage of program and data at a high speed, fully under program control. Directory and peripheral functions will allow chaining of programs and storage and retrieval of data.

-timer ROM: a ROM featuring a timer with all standard functions plus several programmable ones, to use time as an element in programs (such functions as: ON TIMER 1 GOTO or GOSUB, ala HP-85)

Will any of these peripherals become a reality ?? I really do not know: I suppose the Universal interface and the Timer ROM to be already produced, to be commercialized very soon. The tape drive and the printer seem somewhat more utopic. Any news (or comments) ?

A friend of mine, who recently bought an HP-85 computer (to which I have a relatively easy access), also got something more: he - owned a 41c. But , after he bought the 85, and after he was used to BASIC, he began to find the 41c unfriendly: BASIC and the "machine language type" of the 41c have very little in common. So he decided he needed some type of calculator using a language more similar to that of the 85. The solution: he sold his 41c, and bought the SHARP PC-1211 Pocket computer, programmable in BASIC. Now my friend needed not to change his mind when turning from his 85 to his calculator or viceversa, because both were certainly compatible: a program written for the 85 is, in most cases, easily adapted to the Sharp, and viceversa. To give an example: one who speaks English, understands both English people, or US people. Or Australian people, or ... . The point is that the local implementation of the language slightly varies from country to country, but the language itself is the - same. Something similar is applicable to the 85 and the Sharp. The 85 - features a very extended 32K BASIC, while the Sharp has a less powerful one. But both are BASIC, and you do not have to change from English to Chinese to communicate with it: just adapt yourself to the "accent".

The point of all this is that my friend allowed me to use the machine for a week. So I had time enough to fully appreciate it. Now I want to tell you my impressions about the machine, together with a brief, but detailed, description of it, and some comments on the product.

The Sharp PC-1211 (marketed in the US under the name TRS-80 Pocket computer, though they are exactly the same machine; the - Tandy machine has not more memory than the Sharp.)is a very slim, compact calculator type machine. At a first glance, it looks very pretty, certainly prettier than the old 41c. It is metallic, has a 24-character dot matrix display that produces better looking alpha-and numeric characters than the 14-segment characters display of the 41c. The keys provide some tactile feedback when pressed, though not as strong as HP keyboards.

Besides, the machine is almost inexpensive: bought in Spain, it costed just over 210 US$, including its cassete interface. This is very cheap: a bare bones 41c costs twice as much, card reader not included. If you want card reader, you need another 250 US$. And once you have spent that little fortune, you are left with a machine having 445 bytes of programmable memory. The Sharp features 1424 bytes, almost 1000 bytes more. You'll agree it is really cheap.

The Sharp machine has, like the 41c, continuous memory: program, data, and assignations are preserved when the machine is turned - off. The angular mode and the RUN,PRO,etc. mode are preserved too (the 41c has the advantage of preserving all of its contents, including the program pointer, so you can turn off the machine, then on, and resume program execution.).There is a master clear for the Sharp: the button labelled "all - reset" clears, if pressed, the whole machine. This is very convenient if a static crash takes place: just press the button with a pencil, and you may use the machine again. This is more convenient than having to take - out the batteries to reset. On the other side, changing batteries must be a nuissance: it is necessary to unscrew 4 screws, in order to remove the back cover, to replace batteries. Fortunately, batteries seem impossible to discharge.

Used in manual calculation, the Sharp is a delight. I never liked the AOS system of Texas Instruments calculators. You get lost using brackets and functions, but it was just because you couldn't see - the computations in the display, just the results.Once introduced a sequence of operations, you had no access at all at the operands and operators. You just pressed the equals key and got a result. No way to know if it was correct or wrong. Further, if you were sure the result to be wrong, no way to know were the mistake had taken place. And besides, the AOS system of Texas Instruments calculators is not really algebraic: to compute SIN 32, you had to press 32, SIN, which does not conform with algebraic logic. It is easy to see why most people prefer RPN: no messy operation, you have - complete access to intermediate results, less keystrokes, parentheses are not needed. But RPN is not without blame, either. First, the 4-level - stack is insufficient to attack most complex problems left-to-right, and some reordering is necessary prior to press any keys. With a little prac-

tice is not difficult at all, but the ideal machine should not require any effort on the user, just key-pressing, and with a 4-level stack, thinking is required, no doubt. Second, we may love RPN, but most computations are written down, in paper, in algebraic form, and to compute them using a RPN calculator requires some internal conversion.

Sharp uses BASIC to solve computations manually. BASIC of course, uses algebraic hierarchy to do the work. To resume, the calculator mode of the 85 and the RUN mode of the Sharp work alike: you enter the expression as it is written in true algebraic form. True means that to - compute SIN 32 you enter SIN 32, not 32, SIN. Sharp eliminates this way - most drawbacks of TI AOS: now you can't get lost while you are entering your numbers and operations, because you see them in the display as you enter them. If the expression is longer than 24 characters, the display scrolls automatically. You can enter expressions up to 80 bytes long (more than 80 characters: SIN is 3 characters, 1 byte). If you don't remember whether you entered a parentheses or not, you can use the cursor controls to scroll the display either direction. Then, once you entered the whole expression press ENTER and the result is displayed at once.

If you made an error, an error code appears, and you simply press any cursor control to recall to the display the whole expression, with the cursor blinking at the portion where an error was first detected. So you can easily correct any mistake: you can at will delete, insert or replace any character or characters in the expression, using the cursor controls (that repeat if held pressed).

Which is more, similarly to the LAST X function of - the 41c, which recalls to the display the last number or alpha in it, pressing a key recalls to the display the whole expression which was in it before executing a computation. You can then edit the expression and recompute again. This procedure may be repeated as often as desired. Also, you can compute several expressions at one time, simply separe them using commas. Another feature is that the result of an expression may be used in another, by simply typing new expressions immediately:

Examples: to compute 2+3, then add 4 to the result:

```
2+3 ENTER   - (display) - 5
 +4         - (display) - 5+4
    ENTER   - (display) - 9
```

To find the roots of a quadratic equation (manual calculation):

press: A=1,B=5,C=-6,(-B+√(BB-4AC))/2A

pressing ENTER will cause the above computations to be executed, and the value for the root being displayed. To compute the other root, press the cursor key (the whole expression reappears), place the cursor over the + then press -, ENTER and the 2nd root appears. To solve another equation, press the cursor key (the expression reappears once more), and change or edit the values for A,B, and/or C as desired, then press ENTER to compute the root(s). As you may see, it is truly astounding what you can do in just manual calculations. That can't be done with the 41c, unless programmed.

There is another useful feature: the Sharp machine uses a full algebraic hierarchy, including implied multiplication and - logical (Boolean) computations. The machine has a 8-level stack for numbers (intermediate results) and 15-level stack for functions. So almost any expression can be entered left to right. For instance, the well known HP example to compute the Mach number can be computed in the Sharp by - keying it as it's written. Besides, all internal computations and functions are computed with 12-digit mantissas, truncated to 10 places for presentation. Thus, there are 2 guard digits, to increase accuracy.

To help save unnecessary keystrokes, the Sharp allows implied multiplication. To compute 4x A x C, you just press 4AC. TO compute 2PI, simply 2π. To compute sin(2x), simply SIN 2X. As you may see, the parentheses before the 2X are not needed. Similarly, to compute e^{-5x}, simply EXP -5X. No parentheses, no x sign. Also, final parentheses need not be closed, ENTER closes all final parentheses. This techniques save bytes when programming, as we'll see.

Under manual calculation, you may use memories, if

you want to. You have (if no program) as much as 204 registers (each register is 8 bytes). Each registers holds either a number or 7 alpha characters. To store SIN 30 in register A, simply: A=SIN 30, ENTER. To recall the contents of A, simply A, ENTER.

You can perform logic computations, too: for instance: A= 2*(B=C)+3*(C+D)>=SIN(H-G)) , if computed would result in A having a value of 0 if both B=C and C+D)>=SIN(H-G) are false, 2 if one of them is true (the 1st) and the other is false, or 5 if both are true, or 3 if the 2nd is true and the 1st is false. Logical computations, as you can see, can be mixed with normal calculations. This, we shall see, is very useful when programming.

Another feature: continually typing SIN for calculating a sine can be tedious. Similarly to the 41c, you can assign functions (and programs) to keys, for its execution at the press of a key. But there are two fundamental differences. First, assigning functions to keys does not consume program or data memory. There is a separate 48-byte memory to store assignments. Assigning a function takes a minimum of 2 bytes, thus you can theoretically assign a maximum of 24 keys. But this maximum is never reached. The advantage: having assignments does not use program memory. Disadvantage: máximum of 18 different assignments at one time. But there is more: in the 41c, you are limited to 1 or 2-byte assignments. There is no similar limitation in the Sharp: you can assign any sequence of functions or keystrokes to a maximum of 47 bytes. Thus, you can have the sequence √(XX+YY+2XY) assigned to the (shift)A key: pressing (shift) A would enter that sequence in the display either in DEF,RUN, or PRO (program) mode. This is very convenient: very used sequences are: 1"A" REM" which assigns a program to the shftA key and provides a title, or S=E2↑S-INT E2↑S which generates a pseudo-random number, etc. Another feature: the same key can be assigned simultaneously to a sequence and to a program.

This briefly resumes manual calculation. Programming is also truly remarkable. As you know, the Sharp uses BASIC language. BASIC is a high-level language, which allows the user to comunicate with the machine more easily. Of course, there are opinions and there are opinions, but I, who have programmed RPN HP calculators for years, and who also have learnt how to program in BASIC and FORTRAN, I consider myself as being qualified to express an impartial opinion about both ways of programming.

Sharp uses BASIC. How good is its BASIC ? The answer: surprisingly good ! Of course, due to the pocket size of the machine, and other similar limitations, its BASIC language is somewhat limited in some aspects. But, believe it or not, there are some fields in which the BASIC of the Sharp exceeds that of the HP-85 itself.

First of all, programming in BASIC is very comfortable. The 41c uses a "machine language-like" language, that is, no language at all. Just programmed versions of sequences of keystrokes. When programming the 41c, you must take care of a lot of details.For instance, if a loop is to be set up, you must take care of placing a label for the loop, deciding whether it is to be incrementing (ISG) or decrementing (DSE), etc. As a result, making a program for the 41c is far more complex than if written in BASIC or other high level language. This is true. It does not matter that you are very used to the 41c and feel with it very comfortable. Being equally used to BASIC, you would realize that making a program is by far less time consuming, and the program flow is much more clear. I'll give examples in a moment.

The BASIC of the SHARP is quite complete. Most instructions are extended. For instance:

to input data: INPUT "ENTER A",A,"N=";A(27),A(I) when executed in a program, this line would display ENTER A. You then enter a value or any expression, the machines computes it and enters its value in the A memory. Then N= appears, etc.

As you may see, a single INPUT is capable of prompting for the value, or values, with a message if required. You can enter numbers, alphas, or expressions. You do not need to (alpha) (alpha) to enter characters, just press the keys. As you can input expressions, which are automatically evaluated, you can enter more easily values such as SQR(2), etc. (You cannot do this in the 41. If a program prompts for a value, you cannot be sure about if you can disturb the stack or not, unless specified, so, if you

attempt to evaluate an expression, you risk to disturb the stack, probably causing incorrect execution when resuming.)

A BASIC line can contain as many as 80 bytes. Multi-instruction lines are allowed. You can enter the instructions with arbitrary spacing: the machine formats each line before entering, supressing innecessary spaces, or adding spaces just before keywords: for instance, entering 10IN P UTA causes the machine to display 10:INPUT A when the ENTER key is pressed. You can list a program, view it line by line (scrolling the line if necessary), clear lines by simply entering its line number, edit lines using the cursor controls, and insert, delete or replace characters: the changes are not active in memory until you press ENTER. As the changes may take place on the line number, it is very easy to duplicate lines (most of these things you can't do with a 41c). You always know the exact number of bytes free to program, using the MEM function. A program can be given alpha labels, for reference in RUN mode.

Further, almost every keyword has an abbreviated form, to help when manually loading programs: for instance, INPUT can be entered as I. GOTO as G., NEXT as N., etc. When you press ENTER, the machine converts the abbreviations to its full name. Each keyword (such as INPUT) takes only 1 byte of memory. Implied multiplication, final parentheses supression, final quotation marks supression, multi-statement lines, all these things help to save a lot of bytes.

The Sharp has not as many functions as the 41c has, but it does have a good deal of them: all trigonometrics, exponential, logarithmic, plus absolute, sign, integer part, all arithmetics, exponentiation, angle conversions, all logic operations, etc, form a good bibliotheque of functions. It operates with 12-digit mantissas, in all 3 angular modes. It includes a beeper for audio output.

The GOTO (and GOSUB) instructions is very powrful: typical examples: GOTO 10 , GOTO 2*SIN(A+B)+EXP X (which implements indirect GOTO. The 85 does not allow this. It allows ON ... GOTO, which is less powerful), GOTO "SOLVE" (goes to the line labelled SOLVE), GOTO H$ (goes to the line labelled as the alpha characters stored in variable H$. This allows indirect alpha GOTO). As you may see, you have the flexibility of label (alpha or numeric) addressing of the 41c together with the line number addressing of the 85.

Conditionals are another strong point of the SHARP. In the 41c, you are just allowed to test x=y, or x<y, or a flag, and then, if the test is met, you have one line to decide what to do, a GOTO in most cases. In the Sharp, you can do things like:

```
IF A+B-C>=J+H-SIN (A-B) BEEP 1: PRINT "NOT LEGAL":N=N+1: GOTO 80
```

as you can see, if the test (quite complex, by the way) is met, the machine will beep once, then print a warning message, increment a counter, and go to line 80. If not, it will simply resume execution in the line after the IF.

Which is more, due to the logical operations, you can perform intrincate tests easily. For instance, you want to go to line 120 if A equals B or if C is greater than D and H equals B+D simultaneously. This is accomplished this way:

```
IF (A=B)+((C>D)*(H=B+D)) GOTO 120
```

simple, isn't it ? Imagine how to perform this test using the 41c!! It would take a lot of GTO's and LBL's, not to count programming effort to accomplish the programming itself. Besides, IF can be followed by any instruction: even a FOR-NEXT loop, or another IF (those things you cannot do with the 85). Logical programming becomes terribly easy with the Sharp.

The PRINT (and PAUSE) instructions are quite powrful, too: for instance, the line:

```
PRINT USING "##.###";C+SIN(D+H);"=COST";A$(I);USING "#.#^";J
```

prints simultaneously numbers, texts, alpha variables, and computed values, all of them formatted as desired by USING (the equivalent of FIX and SCI).

Loops are possible using the FOR TO STEP NEXT construction. For instance, the line:

```
FOR A(H) = B+C TO EXP(B-C) STEP 5J+2
```

would set up a loop, with the variable A(H) taking the initial value of B+C and being incremented by steps of 5J+2 until it reaches or exceeds EXP(B-C). Thus, it is very easy to set up a loop, which can be incrementing or decrementing. The limitations are that only 4 nested loops can be executed at one time, and that, similarly to the 41c, the maximum value for the final value

must be only 3 digits (from -999 to 999), though the STEP value can be from -999 to 999 (in the 41c, only 1 thru 99). There are no limitations on the initial value, it may be PI, for instance. Subroutines are also limited to 4 levels (6 in the 41c), which proves sufficient in most cases.

There are many more features and particularities not described here, because I do not want to convert this letter in the owner's handbook of the Sharp machine. Just point out that programs can be very easily edited, that you can single-step (single-line, to be exact) thru a program, seeing the line which has just been executed, and being able to perform manual calculations between one step and another, resuming the single stepping. When an error is detected during program execution, an error code appears, and if you press the cursor key, you can see in the display the faulty portion of the line, with the cursor blinking at the location where the error was found. Another very, very useful feature: you can stop momentarily a program, make any hand computations, then resume the program execution. That you cannot do with the 41c !!! If you stop a program running in the 41c (at an arbitrary point), then try to perform intrincate computations by hand, you will disturb the stack no doubt. Then, when you press R/S to resume execution, the machine will find a different stack, and can give place to erroneous execution thereafter. So, if you are running a very long program (such as computing PI to a thousand places), you are condemned to let it execute completely: if you dare to stop the program, make a complex calculation, then press R/S, chances are you have spoiled the execution.

That cannot happen with the Sharp: the Sharp has no user stack, so stopping a program and making hand calculus cannot spoil the "stack" at all, because calculus are made via an input buffer of 80 bytes, and internal stacks for data and functions. Once you are done with the calculus, executing CONT will resume program execution from where it left, without any disturbance at all. You can, thus, run a very large program and, if necessary, halt it at any moment, use the machine as a manual calculator, then resume with the program. Great, isn't it ?.

Programs can be run using either RUN (which can be followed by a line number: the execution begins there, or an alpha label, or even a variable (indirect RUN), or by assigning them to keys. To assign a program to a key, simply include a "(letter)" after the line number of its first line. Pressing (shift)(letter) in DEF mode will start program execution. Line numbers and /or labels are arbitrary, and can be introduced in any order: they are sorted and placed properly as they are entered. Editing lines takes less time than similar manipulation of 41c programs: no PACK, no delays, no nulls scattered here and there. GOTO's and GOSUB's are not compiled, either, so program execution is not as fast as in the 41c. The 41c is faster. But writing a program in the 41c and entering it using the keyboard takes much more time than in the Sharp. See examples. The BASIC of the SHARP allows unidimensional arrays, such as A(8),A(J), A(23+47I-SIN(C+H)),which is the equivalent of indirect store-recall in the 41c, though the Sharp allows up to 15 levels of indirect addressing: for instance, A(A(A(A(A(B+C))))) is allowed (it is like RCL IND IND IND IND IND (B+C)). Most operations on the SHARP allow this indirect scheme: GOTO's,GOSUB's, BEEP, etc.

Finally, cassette control: it is not as convenient as magnetic cards, because, though the machine has full control of start and stop, record and play, it cannot control fast rewind or fast forward. This results in more time needed to find and load a program in the tape. Programs are stored and retrieved by file name. A single C-60 tape can store over 40K of programs. Taking note of tape counter numbers, it is quite easy to load a program from tape to the machine. There exist an equivalent to the VERify function of the 41c. Data can also be stored and retrieved either manually or under program control. Assignments can be stored and read, too. Furthermore, programs can be chained: a program can call another program from the tape, which is inmediately loaded and executed, without manual intervention. That program can still call another program, etc. Thus any program which can be segmented in portions of up to 1424 bytes can be executed automatically by the machine, without the presence of the user being ever required. The machine takes care by itself of the tape recorder, via control (remote). This is impossible with the 41c: the user is needed to feed the cards thru the card reader, cards will never get out of the card holder to insert themselves at proper times into the slot. Very useful feature, indeed. Besides, a home cassette recorder is much cheaper than a magnetic card reader, and C-60 tapes are much cheaper than 200 or more mag. cards.

Well, we come finally to the end of this extremely long description of the Sharp machine (see my motivations later). Just present some examples of its programming :

Example 1) We try to find the numbers of 3 digits, of the form $\overline{ABC}$, such that they be equal to the sum of the cubes of its digits. For instance, 153 is such a number because $153=1^3+5^3+3^3$.

The Sharp program is:

```
10 N=100: FOR A=1 TO 9: FOR B=0 TO 9: FOR C=0 TO 9:
   IF AAA+BBB+CCC=N PRINT N
20 N=N+1: NEXT C: NEXT B: NEXT A
```

63 bytes
takes 40 seconds to find that 153 is a solution (54 full loops)

Now, please, be kind and write the same program for the 41c. You'll realize that it is much more difficult, and once written, the program flow is less clear: you'll easily understand what the BASIC program does by simply looking at its listing. Much more difficult looaking at the 41c listing. Besides, the BASIC program can be written "in the head", no need to use paper to write it. Impossible to do the same with the 41c equivalent program. And I was kind enough that once the test is met, only printing N is required. If I had - stated: IF AAA+BBB+CCC BEEP 1: PRINT "SOLUTION IS ";N , the 41c program would have needed to call a subroutine to do all these things.

Example 2) We want to compute sin(x) from its Taylor series expansion, carrying accuracy to 10 significant digits:

The Sharp program is: (assigned to the = key, automatic input)

```
1 "=" AREAD X: Y=X, K=-XX, T=X, N=3, L=E-90
2 T=KT/(NN-N), Y=Y+T, N=N+2: IF ABS TL GOTO 2
3 PRINT Y
```

This program is 71 bytes long (though a few can be saved). To find sin 1, simply: 1 (shift) = gives 0.8414709847 within 5 seconds.

The Sharp machine comes with 3 books: 2 of them are the "owner's handbook" and the "beginner's guide to BASIC". Both are well written, though not as complete as HP books, but give many examples, and discuss almost all aspects of machine operation, including synthetics (the Sharp has synthetics, of course, though unexplored yet). Examples, exercices and its solutions are given, to fully appreciate all machine characteristics. On the other hand, the "Standard applications" clearly beats the similar book for the 41: It is a 300+ pages book, with contains over 130 programs in BASIC, fully documented, algorithms included, with examples and exhaustive documentation. These programs include Linear Equations, Matrix Inversion, Eigenvalues, Root Solving, ..., full statistics programs (linear, exponential,...,etc regressions, etc), engineering, commercial, even games. This book should serve as an example for other manufacturers (HP included: the tiny 41c Applications Book seems ridiculous when compared to this one).

As a final resume: The 41c, upgraded with memory modules, is more powerful than the Sharp: RPN "language" is more flexible and byte-saver than BASIC, so that a program written in the 41c takes generally less bytes - than an equivalent one in BASIC, and runs faster, too. But the Sharp BASIC is much more easy to use than RPN, and allows programs to be written and developed much quicker, as well as taking much less effort to write a program in BASIC than the same program in RPN. The resulting program is compact, and readily understood and modified, even months after you wrote it. Don't forget that BASIC is a high-level language, while RPN is a machine language.

However, the Sharp is, no doubt, a top-of-the-line calculator, without equal but the 41c . (The Sharp admits peripherals, too. A printer using non-thermal paper is sold in Spain at half the price of that for the 41c). This is the reason why I am telling you all these things: I feel it is a pity that so good a calculator-computer passes almost inadverted, just because it is not an HP product. Now HP and TI have a very strong competer: Sharp, and probably Casio and other japanese enterprises. If Sharp goes on like this, it will finally market a product far outgrowing any HP product: the Sharp PC-1211 is very near the 41c, if not better for some kind of programmers. Just think on the many books on the market plenty of BASIC programs, readily adapted to the Sharp. Just think that the Sharp, including interface and printer cost just 390 US$ here in Spain. A real bargain !

That's all about the Sharp. Excuse the extremely long length of the description: it is just that I wanted you to learn something about this machine, not only the few aspects you can observe in a brochure or having the machine in your hand a moment or two. I remember the times when HP machines where considered "odd" because they used RPN instead of algebraic. Let's not commit the same error twice, considering worse a machine just because it uses BASIC instead of RPN ! HP is good, but it is not God. There may be better things after all, and the japanese are on the way.

= = = = = = = = = = = = = = = = = =

Here included are several things: as I told you before, I have had almost no time to dedicate to programming or experimenting (and it's a real pity: as soon as I can I will dedicate all my efforts to internal ROMs microcode and its decoding. It is the most promising field of new discoveries) , but , at some spare times, I had time enough to complete a whole program and a synthetic routine (which could have been a PRC ROM routine):

-the full program is a new version of Othello : I call it Othello level 2, because it features 2 levels of play in the same program. Level 1 plays exactly the same as the previous Othello program, and in the same time,or faster. Level 2 features an improved level of play, which results in most of the typical errors of level 1 being avoided, and offers a strong challenge to the user. However, level 2 takes much more time than level 1 to perform a move, - about 6 times more time. Fortunately, you can change from level 1 to 2 or viceversa at any moment during the play. You can change sides, too, or can make the 41c to play for you, or a whole game against itself, etc. Most options of the previous program have been conserved, while new options have been added. Improved techniques make the program more efficient in both levels.

Despite being much more complex, Othello 2 is the same length as Othello: 672 bytes, so it fits into 3 mag cards. But it requires a separate data card, which must be loaded at the beginning of each game. The data card contains the strategy ranked in hierarchies, the moves array and several other needed constants. Othello 2 is printer compatible, and will print the - board automatically after HP moves if a printer is present. However, Othello 2 needs more registers to do its works than Othello, so 3 single density modules are required to run it.

That's all. If you liked Othello, you'll find Othello 2 a real challenge. Try Level 2, and see if you can defeat it. Magnetic cards and full description included. (I also have a version of Othello written for the Sharp in BASIC. It plays the same as Othello for the 41c. If any member of the Melbourne Chapter has a Sharp and wants the program, ask him to write)

-the routine is a quasi-truly random number generator. I've seen many RNG, but all of them required an initial seed to be introduced. I've also seen seed generators, but they also required an initial seed, or they produced always the same set of seeds !! What the membership needs and wants is a routine that you call, and every time you call it you get a different number, so that you would never predict which one you would get. Such a routine is (on computers) based on internal timing: each time you call the RANDOMIZE function of the HP-85,it elaborates a seed based on internal timing, which has millisecond resolution at least, so the seed you get varies over 1000 times a second. This would be - ideal if we could implement such a timing-based routine on the 41c. But we can't. So I present here SEED, a routine which generates a seed based on the internal status of the calculator: flags, characters in alpha, final end, address of R00, program pointer, subroutine addresses, etc. So, the seed you get entirely depends on the location where the XEQ"SEED" is executed, the flag - status, the current SIZE, etc,etc. Besides, calling SEED in a loop, always - produces different seeds (not recurrent seeds). Minimal changes in configuration (changing or editing a program, etc) completely change the resulting - seed. So, include a call to SEED in any program using random numbers, and you are 99.9 % guaranteed to obtain an unpredictable sequence.

That's all. My free time has elapsed. Find included photocopies and magnetic cards of the programs. Also find included some views of Madrid, the capital of Spain. They can give you some idea about the town (I received a post card from you which featured a view of Melbourne, I think. Madrid is no less than 600 km apart of the nearest beach, right in the exact center of Spain, far from the Sea).

Well, John, it has been a real pleasure to "talk" with you once more. Please, write as soon as you can.

Yours sincerely:

post-data : I'll send more material as soon as I can

xxx OTHELLO 2 xxx

I know that it may be a little redundant, but I - feel that this new version of my program OTHELLO is a signifi - cant improvement both in level of play and possibilities over - the previous one, published in PPC Technical Notes V1N2P44-50 (PPC TN is a publication of the Melbourne Chapter. If you are - interested, you may obtain membership forms and sample copies by writing to J.E.McGechie, Philosophy Dept.,Monash University , Clayton, Victoria, Australia 3168).

```
I PLAY 64
FLIP 2

  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - ▒ O - - -
5 - - - ▒ O O - -
6 - - - ▒ ▒ ▒ - -
7 - - - - - - - -
8 - - - - - - - -
```

All descriptions of the Othello game - (also known as Reversi, Samurai, etc) may be found in the previously referenced is- sue of PPC TN. A brief description of the game will be given herein, for the sake of completeness.

Othello is played in an 8x8 board. There are two standard openings: parallel and diagonal . One of the players plays white (the O's), the other plays - black (the chequerboard symbols).

To make a move, a player places one of his pieces into an empty location (represented by a dash) in such a way that:

-it should be adjacent to an enemy piece.

-at least one enemy piece must be enclosed between the just pla- ced piece and another one previously placed. Any number of pie- ces enclosed in between are flipped: they became of the captu - rer's colour. No empty locations can be enclosed: only full rows of enemy's pieces can be flipped. Capture is possible either in horizontal, vertical or diagonal directions; if more than 1 row is enclosed at the same time, all are flipped.

For instance, look at the printout: black plays his piece at location 46 and → captures the white piece at 45 (4 vertical 5 horizontal), enclosed between the piece just placed at 46 and the one at 44,and al so captures the white piece at 55 (enclo - sed bet. 46 and 64), and the one at 56 (en closed bet. 46 and 66). Should white play now at 67, he would capture the black pie- ces at 64,65,66.

```
I PLAY 63
FLIP 1

  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - ▒ O - - -
5 - - - O O O - -
6 - - O ▒ ▒ ▒ - -
7 - - - - - - - -
8 - - - - - - - -
```

PROGRAM CHARACTERISTICS

"OTHELLO 2" is exactly 672 bytes long so it exactly fits onto 3 magnetic cards. Additionally, the program uses a data card which contains data used in the program. You must create the data card in order to run the program. See listings.

This program needs 3 RAMs (single den sity) in order to run. It requires SIZE of at least .You need the card reader to read the data card at the very beginning. Once you've read it, the machine turns it- self off, to allow you unplug the card re- ader and plug the printer, instead. If you plug the printer, the board will be printed automatically after every machine move. If you have no printer, it does not matter,as the program is printer-compatible.

```
MOVE?
                46    RUN
YOU PLAY 46
FLIP 3

  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - ▒ ▒ ▒ - -
5 - - - O ▒ ▒ - -
6 - - O ▒ ▒ ▒ - -
7 - - - - - - - -
8 - - - - - - - -
```

This program differs of the previous version in several aspects:

1)-it features 2 levels of play : in level 1, the program plays exactly the same as the previous version, and in the same times: about 25 minutes for a whole play (30 machine moves) if no printer, 60 if printing boards.

Level 1 features a surprisingly strong level of play in moderate thinking times: from some 70 seconds for a move at the early stages of the game, to 12 seconds for a move near the end of the game.

On the other hand, the brand-new Level 2 offers a somewhat stronger level of play, at the expense of more time. Level 2 avoids most errors commited by level 1, by exploring the position more in depth. This often results in improved game play. However, level 2 requires much more time than level 1: average time is 4 minutes per move. Maximum is about 10 minutes, but thinking time decreases very quickly as the game progresses, to a minimum of a few seconds per move. So, don't fear if the first moves seem to take too long. All in all, the whole game (30 machine moves) takes about 1½ hours if no printer, and about 2 hours if printer is present. As a guide, level 2 is 6 times slower than level 1.

Both levels play no random move at all. The same game is played if you repeat the same moves. This can be used to correct any mistakes that caused you to lose a game. Both levels are greatly optimized to play as fast as possible, using techniques much more efficient than that of the previous version.

A good advice is that you play level 1 until you become a good player. Then, switch to level 2 for improved games.

2) - flags give the user complete control on the program behaviour. You may select:
   - who plays black and who plays white (either you or HP)
   - what level (1 or 2) is the machine playing
   - whether the board is printed after every move or only after machine moves.

The first feature may be used to make the machine play against itself, by changing sides (you were playing black, you now decide to play white). More simply, you may change sides whenever you want, even in the middle of the game. This can be definitely useful if you are losing !! The second feature is useful to switch from level 1 to 2 or viceversa even in the middle of the game. For instance, play the first 6 moves in level 1, then switch to level 2 for improved performance for the rest of the game.
The 3rd feature is useful to save paper and time, while allowing the user the possibility of having all and every positions of the game recorded on the paper.

INSTRUCTIONS : load the program. XEQ "02" → CARD
-load data card, both sides. After reading the card, the machine shuts off. Unplug the card reader, to make place for the printer (if you have double or quad density modules, or if you do not have a printer, this step is not necessary). Plug the printer, and turn the 41c on. Now, R/S → LEVEL 2?

- if you want to play level 2, simply R/S →
- id. id. id. 1, N , R/S → DIAG?

- if you want diagonal opening, simply R/S →
- id. id. parallel opening, N , R/S → HP 1ST?

- if you want HP to make the 1st move, R/S →
- if you want to make the 1st move, N , R/S →

Initially, you play black, and HP plays white. You may change sides at any time, while the machine is at a halt. To change sides, use the flag 00:

SF 00: HP plays black, you white
CF 00: HP plays white, you black

This can be used to make the machine play against itself (see later)

-now there are 2 possibilities:

HP moves : if HP moves, it will think its move for a while, then display:

→ I PLAY nn
→ FLIP p

where nn is the vert/horiz reference of the square where it moves to, and p is the number of flipped pieces. If printer is present, the board is printed now, reflecting the new position, and your move is requested with :

→ MOVE?

(if no move is possible for HP, it will display NO MOVE, then request your move). (if no printer, please actualize the position yourself, putting an HP piece in the indicated square, and flipping the pertinent pieces)

YOU move : if you move (you have been prompted by MOVE?)

-enter your move (vert/hor): xy R/S

your move is checked for legality. If it is found to be illegal, BAD MOVE will be displayed, and you will be requested for another move.

if your move is found to be legal: → YOU PLAY xy
→ FLIP p

is displayed, and HP proceeds to think its move.

-if you cannot make any legal move, enter Ø as your move

Ø R/S → HP acknowledges that you have no move at all and resumes to think its own move.

Once the last player makes the last move, the board is printed, and :

→ GAME OVER
→ HP: xx, YOU: yy
→ HP (I) or YOU WON

where xx is the number of HP's pieces on the board, yy is the number of your own pieces. The player which has more pieces at the end of the game, wins the game. If both have the same number, it's a tie, and no I or YOU WON message is displayed.

REMARKS : if you don't use a printer and missed the I PLAY xx display, simply press backarrow once to clear the MOVE? prompt, and the last HP move will be in the X register.
-to change the level of play at any moment: wait the HP to be at a halt, then: SF 09 for level 1 , CF 09 for level 2

-to change sides at any moment: wait for the 41c to be halted, then: CF 00 : HP plays white, SF 00: you play white

this can be used to: simply change sides, either to move twice (HP or you) consecutively, or to avoid losing, if HP is winning: have HP playing your losing pieces instead of you. It can be used to have the machine play aganist itself: for instance, HP has just played white, and MOVE? is in the display. Now, to make HP play against itself, change sides (SF 00: now HP plays black), and enter Ø as your move (Ø R/S): HP now will play the black pieces against its white pieces. This sequences can be repeated as often as desired (so you can have a whole machine-machine game if you want so, or, more simply, a "good" advice if you do not know what to play).

```
                                RUN
LEVEL 2?
                                RUN
DIAG?
                                RUN

   1 2 3 4 5 6 7 8
1  - - - - - - - -
2  - - - - - - - -
3  - - - - - - - -
4  - - - ▒ O - - -
5  - - - O ▒ - - -
6  - - - - - - - -
7  - - - - - - - -
8  - - - - - - - -


HP 1ST?
                                RUN
I PLAY 65
FLIP 1

   1 2 3 4 5 6 7 8
1  - - - - - - - -
2  - - - - - - - -
3  - - - - - - - -
4  - - - ▒ O - - -
5  - - - O O - - -
6  - - - - O - - -
7  - - - - - - - -
8  - - - - - - - -


MOVE?
                        33      RUN
BAD MOVE
MOVE?
                        66      RUN
YOU PLAY 66
FLIP 1
I PLAY 56
FLIP 1

   1 2 3 4 5 6 7 8
1  - - - - - - - -
2  - - - - - - - -
3  - - - - - - - -
4  - - - ▒ O - - -
5  - - - O O O - -
6  - - - - O ▒ - -
7  - - - - - - - -
8  - - - - - - - -


   1 2 3 4 5 6 7 8
1  ▒ O ▒ ▒ O ▒ O ▒
2  ▒ O O ▒ ▒ ▒ O O
3  O ▒ O ▒ ▒ O O O
4  ▒ ▒ ▒ O ▒ ▒ O ▒
5  ▒ ▒ ▒ ▒ O O ▒ ▒
6  ▒ ▒ ▒ O ▒ ▒ ▒ ▒
7  ▒ O O O O ▒ ▒ O
8  ▒ ▒ ▒ ▒ ▒ ▒ O O

GAME OVER
HP  24  YOU  40
```

EXAMPLE

this printout reflects the first few moves of the beginning of a game.

Here level 2 is being played.

Diagonal opening has been selected, and HP plays the white pieces.

HP also makes the first move.

It plays 65, so it flips the black piece at 55.

Black's answer is 33, which is rejected, as it is illegal.

The game continues ...

(By the way, the game normally ends when the whole board is full of pieces, but it can also end if no player can make a legal move. Counting of the pieces is not performed in such a case. It is suggested that you make XEQ 06 to force printing of the position (if not already printed) , then count the pieces manually to decide the winner. This case is infrequent, however).

TEST EXAMPLES : to test correct loading

level 1: diagonal op, HP first (YOU-HP)

-65/46-33/64-63/43-66/72-53/67-81/42-68
75-36/35-84/86-51/31-56/27-18/57-85/83-58
76-41/61-34/62-74/24-13/25-16/26-52/32-47
23-14/15-73/17-37/38-48/78-82/71-87/12-11
0-21/77-88/22-28

final: HP:47, YOU:17 : HP WON

level 2: diagon op, HP first

-65/66-56/46-33/63-64/53-43/23-36/34-35
74-76/87-83/26-47/57-42/41-24/13-31/21-14
15-52/62-25/32-86/85-61/58-68/78-38/67-16
17-84/82-22/75-73/37-48/28-72/81-71/51-77
88-27/18-0 /12-11

final: HP:16, YOU:48 : YOU WON

(as you may see, even level 2 can be defeated if you play well enough)

Typical end of a game, here you have 40 black pieces on the board, while HP has only 24, so you won.

That's all ! Happy programming, and thank you for your kindness.

VALENTIN ALBILLO (4747)

```
01♦LBL "02"
02 FIX 0
03 CF 00
04 CF 29
05 CF 12
06 CLRG
07 13.04
08 RDTAX
09 OFF
10 "LEVEL 2?"
11 CF 23
12 AON
13 PROMPT
14 CF 09
15 FS?C 23
16 SF 09
17 "DIAG?"
18 PROMPT
19 RCL 15
20 STO 75
21 RCL 16
22 STO 85
23 FS?C 23
24 X<>Y
25 STO 74
26 X<>Y
27 STO 84
28 XEQ 06
29 "HP 1ST?"
30 PROMPT
31 AOFF
32 FS?C 23
33 GTO 00
34 65
35 GTO 08
36♦LBL 14
37 CF 07
38 CLX
39 STO 06
40 23.04
41 STO 07
42♦LBL 11
43 RCL IND 07
44 X=0?
45 GTO 12
46♦LBL 13
47 RCL 13
48 *
49 STO 11
50 INT
51 STO 10
52 CF 03
53 CF 08
54 XEQ 97
55 FC?C 04
56 XEQ 98
57 FS?C 08
58 GTO 14
59 RCL 11
60 FRC
61 X≠0?
62 GTO 13
63♦LBL 12
64 ISG 07
65 GTO 11
66 FC? 07
67 GTO 12
68 RCL 09
69♦LBL 08
70 STO 10
71 CF 03
72 XEQ 97
73♦LBL 14
74 CF 03
75 "I"
76 BEEP
77 XEQ 08
78 GTO 00
79♦LBL 12
80 "NO"
81♦LBL 60
82 "├ MOVE"
83 AVIEW
84 TONE 9
85 PSE
86♦LBL 00
87 RCL 10
88 "MOVE?"
89 PROMPT
90 STO 10
91 X=0?
92 GTO 14
93 SF 03
94 XEQ 97
95 "BAD"
96 FS?C 04
97 GTO 60
98 "YOU"
99 XEQ 08
100 GTO 14
101♦LBL 08
102 "├ PLAY "
103 ARCL 10
104 AVIEW
105 23
106 STO 00
107 SIGN
108♦LBL 31
109 STO 02
110 RCL IND 00
111 X=0?
112 GTO 08
113♦LBL 32
114 RCL 13
115 *
116 INT
117 RCL 10
118 X=Y?
119 GTO 00
120 RCL 13
121 ST* 02
122 LASTX
123 FRC
124 X≠0?
125 GTO 32
126♦LBL 08
127 SIGN
128 ST+ 00
129 GTO 31
130♦LBL 00
131 "FLIP "
132 ARCL 01
133 AVIEW
134 RCL IND 00
135 RCL 02
136 *
137 INT
138 LASTX
139 FRC
140 RCL 13
141 *
142 FRC
143 +
144 RCL 02
145 /
146 STO IND 00
147 FC? 02
148 GTO 06
149 FS? 03
150 GTO 12
151♦LBL 06
152 FC? 55
153 GTO 12
154 ADV
155 31
156 STO 00
157 45
158 STO 01
159 79
160 STO 02
161 2.01
162 STO 03
163 8
164 SKPCOL
165 49.056
166 STO 04
167♦LBL 02
168 RCL 21
169 SKPCOL
170 X<>Y
171 ACCHR
172 ISG X
173 GTO 02
174 PRBUF
175 41.048
176 STO 05
177♦LBL 09
178 RCL 04
179 ACCHR
180 RCL 17
181 SKPCOL
182 SF 12
183♦LBL 10
184 RCL IND 05
185 RCL 15
186 +
187 RCL IND X
188 ACCHR
189 RCL 03
190 SKPCOL
191 ISG 05
192 GTO 10
193 PRBUF
194 ST+ 05
195 CF 12
196 ISG 04
197 GTO 09
198 ADV
199 ADV
200 ADV
201♦LBL 12
202 DSE 14
203 RTN
204 FC? 02
205 GTO 12
206 FS?C 03
207 XEQ 06
208♦LBL 12
209 32
210 "GAME OVER"
211 41.118
212 AVIEW
213 0
214♦LBL 07
215 RCL IND Y
216 +
217 ISG Y
218 GTO 07
219 2
220 /
221 X<>Y
222 RDN
223 ST- Z
224 +
225 ADV
226 "HP: "
227 ARCL X
228 "├, YOU: "
229 ARCL Y
230 AVIEW
231 BEEP
232 ADV
233 PSE
234 X=Y?
235 STOP
236 "I"
237 X<Y?
238 "YOU"
239 "├ WON"
240 PROMPT
241♦LBL 98
242 SF 07
243 SF 08
244 FS? 09
245 RTN
246 23
247 RCL 07
248 INT
249 2
250 MOD
251 LASTX
252 -
253 RCL 07
254 INT
255 +
256 X<Y?
257 RTN
258 1 E3
259 /
260 +
261 STO 08
262♦LBL 56
263 RCL IND 08
264 X=0?
265 GTO 12
266♦LBL 57
267 RCL 13
268 *
269 STO 12
270 INT
271 SF 06
272 SF 03
273 XEQ 99
274 FC?C 04
275 GTO 00
276 RCL 12
277 FRC
278 X≠0?
279 GTO 57
280♦LBL 12
281 ISG 08
282 GTO 56
283 RTN
284♦LBL 00
285 RCL 06
286 RCL 08
287 INT
288 X>Y?
289 STO 06
290 X>Y?
291 CF 08
292 RCL 10
293 FC?C 08
294 STO 09
295 30
296 +
297 0
298 STO IND Y
299 119
300 STO 05
301♦LBL 05
302 RCL IND 05
303 RCL 16
304 ST* IND Y
305 ST- 05
306 DSE 01
307 GTO 05
308 RTN
309♦LBL 97
310 CF 06
311♦LBL 99
312 SF 04
313 30
314 +
315 STO 00
316 RCL IND X
317 X≠0?
318 RTN
319 FC? 06
320 STO 01
321 15.022
322 STO 02
323 119
324 STO 05
325 SIGN
326 FS? 00
327 CHS
328 FC? 03
329 CHS
330 STO 04
331♦LBL 01
332 RCL 00
333 RCL IND 02
334 +
335 STO 03
336 RCL IND X
337 RCL 04
338 X≠Y?
339 GTO 12
340♦LBL 03
341 LASTX
342 ST+ 03
343 RCL IND 03
344 RCL 04
345 X=Y?
346 GTO 03
347 CHS
348 X≠Y?
349 GTO 12
350 CF 04
351 FS? 06
352 RTN
353 STO IND 00
354♦LBL 04
355 LASTX
356 ST- 03
357 RCL 00
358 RCL 03
359 X=Y?
360 GTO 12
361 STO IND 05
362 RCL 16
363 ST* IND Y
364 ST- 01
365 ST- 05
366 GTO 04
367♦LBL 12
368 ISG 02
369 GTO 01
370 .END.
```

```
R13=  100,0000000
R14=  61,00000000
R15=  1,00000000
R16= -1,00000000
R17=  9,00000000
R18= -9,00000000
R19=  10,00000000
R20= -10,00000000
R21=  11,00000000
R22= -11,00000000
R23=  0,81881118
R24=  0,00000000
R25=  0,83866168
R26=  0,31381316
R27=  0,63663336
R28=  0,00000000
R29=  0,84855158
R30=  0,41481415
R31=  0,64655356
R32=  0,43463435
R33=  0,74755257
R34=  0,42472425
R35=  0,73766267
R36=  0,32372326
R37=  0,82877178
R38=  0,21281217
R39=  0,72772227
R40=  0,00000000
```

SEED - A QUASI-TRULY RANDOM NUMBER GENERATOR

```
01 LBL"SEED"    17 LBL 00
02 Ø            18 CLA
03 RCL M        19 STO M
04 XEQ 00       20 "⊢∠"
05 RCL b        21 X( ) M
06 XEQ 00       22 CLA
07 RCL c        23 STO M
08 XEQ 00       24 ASTO L
09 RCL d        25 "∠"
10 XEQ 00       26 ARCL L
11 4            27 X( ) M
12 /            28 LN1+X
13 R-D          29 FRC
14 FRC          30 +
15 STO M        31 END
16 RTN
```

62 bytes

Random number generating routines are not exactly new among PPC members. Many of them have been proposed. I have always preferred the routine

```
RCL nn
R-D
FRC
STO nn
```

because of its simplicity and its speed. Some random number generator is included in the PPC ROM.

But all of these routines require an initial seed, some number which is used to generate the whole series of random numbers. Generally, if you input the same seed, you get the same numbers. And you must use some care when selecting your starting seed, because some seeds cause the random number algorithm to enter a loop, thus repeating once and again the same series of RN.

On the other hand, here presented is a routine, called SEED, which is capable of generating a quasi-truly random seed to be used with any RNG algorithm. SEED can also act as a RNG by itself. SEED does not require any input at all.

To use it, you simply: XEQ "SEED " either manually or in a program, and you'll have a random seed (between 0 and 1) in the X register, to be used as you wish.

SEED only uses the stack and alpha register. It does not use nor disturb any other register, status, or flag.

CHARACTERISTICS : Is there any procedure to obtain a truly random seed from the 41c? Yes. If we could get readings from its internal clock (which varies at a very high speed), we would have random numbers based on internal time-keeping. But we aren't able (yet). So we must accept quasi-truly random seeds. "Quasi" means that, in certain conditions, they are predictable.

SEED generates a seed that depends (and varies) on:

- the characters in the alpha register: depending of what characters are in the alpha-register when you call SEED, you'll get a seed or another.
- the point in a program of the XEQ"SEED" instruction, or the current position of the program pointer when you call SEED.
- the address of the sigma registers, the final END, the 00 register, and printer status.
- the status of all 56 flags.

All these higly variable conditions are merged into a numeric value between 0 and 1, which is the resulting seed. As you may see, the value you obtain depends on the entire status of the calculator, and the slightest variation will cause a different seed to be produced. For instance, you won't get the same seed calling SEED from line 02 in your program that you would obtain calling it from line 23. Also, if you change SIZE, or order of programs in memory, or edit a program, or change FIX 2 to 4, or almost any other change, another seed would be generated. In other words, the seed you get depends completely on the whole status of the calculator when you call "SEED". Which is more, calling SEED repeatedly from the same location would generate different ramdom seeds.

This ensures that you obtain a remarkably random seed, because, when you execute your program (including the call to SEED), the seed you get will depend on so many conditions that is highly - unlikely that you would get the same seed twice. This can hap - pen: for instance, loading and executing your program exactly after a master clear results in the same seed being generated. This is completely unavoidable, unless internal clocks are accessed, which is not possible nowadays. In any other circumstances, you can be sure to get a random seed.

As stated before, SEED produces different outputs - each time it is called, so it can be used as a RNG, though I do not recommend this, as SEED takes 3 seconds te generate a random number.

HOW IT WORKS : the contents of M,b,c and d are converted to a numerical (positive) value between 0 and 1, added together and the result, divided by 4, is the final output. This output is previously "randomized" a little more using R-D,FRC, and the final result is stored in M, to ensure a different output if SEED is called in a loop. Those steps, R-D,FRC,STO M are essential. Removing the STO M will cause SEED to always produce the same output, if used in a loop. Removing the R-D, FRC will cause the different seeds to converge to a final value, repeated thereafter.

LBL 00 takes 7 bytes or less in the X register, and converts them into a positive numeric value between 0 and 1, performing the addition, too. The instruction LN1+X acts upon numeric, but non-normalized, values, normalizing them, and contributing some randomness. LN1+X is used instead of LN, to - avoid DATA ERROR if X contains Ø. X cannot contain a negative number at that stage, so the error message is avoided. The - process used to convert any contents of X to numeric values is very simple: basically it consists in changing the sign to 1: this forces the 7 bytes to be considered as positive, numeric values.

Nothing more. Happy programming.

VALENTIN ALBILLO (4747)