Notes on the back story of this letter:

I sent this *24-page* letter to ***John McGechie*** after a long while since my previous letter, including extensive comments on my *HP*-related activities, the recently arrived *PPC Technical Notes* issues 3 & 4, the *PPC Calculator Journal* October & November issues and some ***HP-41C*** microcode notes sent by Tom Cadwallader, plus my detailed comment on the long letter John sent to Hugh Kenner and some peculiarities of the English language which I found somewhat funny, etc. .

Next, I comment on an interesting letter from Mr. Wickes and John's reply, focusing on the huge differences between John's style as an editor vs. Mr. Nelson's and how that personally affected me, eventually making me switch my priorities. I also include my criticism of some perceived attitudes at *PPC CJ*, finishing with this plea to John: *"Please don't Richardnelsonize yourself !"*.

Now, I comment on the proposed *PPC ROM*, which I considered likely to be obsoleted soon by powerful *HP* models just about to be released, and my experiences with some clients and their programming needs, plus some ideas for the ***HP-41*** to be able to survive vs. new rumoured models such as the *Kangaroo* (which came up to be the **HP-75C**, hardly a replacement for the *HP-41C*.)

Finally I include the following materials (featuring full documentation, examples and listings for programs/routines *(4), (5)* and *(6)*; see below) for John's benefit and/or their possible publication in some issue of the *Melbourne Chapter's Technical Notes* magazine, namely:

(1) A (french) article from *L'Ordinateur Individuel* describing in detail how to interface an *EPROM* to a ***TI-57/58*** calculator, which could likely be adapted to work with *HP* calculators.

(2) The first *9 pages* of the English section of the *User's Library Europe*, for John to see if it could be of any use to him or *PPC TN*, in which case I offered to send the whole catalog (English, Spanish, French, Italian, etc. sections) as well as future updates.

(3) A version of my ***HP-41 Othello*** program converted to run on the ***HP-85*** computer, with the obvious advantages of much faster execution (essentially instantaneously), being able to see the board on the *CRT* screen, and easier-to-understand *BASIC* source code, thus making it that much easier to modify/improve and even replace the move logic altogether.

(4) ***MLR***, a *Multiple Linear/polynomial Regression* ***HP-41*** program I wrote, which given a set of data can compute (in the least-squares sense) the coefficients of either a linear regression in N independent variables or else the N^{th}-degree polynomial regression (with N limited only by available RAM), allowing for adding/deleting data and computing projections

(5) ***CHKERS***, *Checkers 3.0*, another complex ***HP-41*** game program I wrote which can play *Checkers* vs. the user, allowing multiple jumps and optionally printing the 8x8 board.

(6) ***VAL***, a small routine which is the "inverse" of **`ARCL`**, as it takes numeric data stored as text in the *Alpha* register and converts them to numeric data proper in the *X* register. It's also intended as an example of calling *PPC ROM* routines from your own programs.

Last but certainly not least, the package also included (and extensively commented on) the ***HP-34C Matematica Avanzada*** *Solutions Book* commissioned by *HP Spain* for the local market. Including it was **extremely** expensive because of the package's hefty weight and farthest destination (Australia!.)You can download this *78-page PDF* book (in Spanish) using this link (26 Mb download):

https://albillo.hpcalc.org/programs/HP%20Program%20VA340%20-%20HP-34C%20Matematica%20Avanzada%20(with%20Notes).pdf

Valentin Albillo, 07-09-2022

John McGechie
Philosophy Department
Monash University
Clayton, Victoria 3168
Australia

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

december 12, 1980

Dear John:

Hello, how are you ? It's been a long period of silence, since my last letter. There have been several good reasons for me to stay silent: We had the S.I.M.O here: the SIMO is a commercial show about computers and all kinds of materials for business and offices. I stayed at the Hewlett-Packard stand, showing people the features of the 85 and the 41c, etc. The SIMO lasted from early november to mid november (8 days, to be exact), but its consequences lasted- all the rest of the month long (commercial calls, sales, and the like) so I've had almost no free time at all, impossible to write cards, letters or whatever, just work, work, work.

I dedicated some spare times (coffee, lunch,...) to the microcode notes kindly sent by Tom (Cad - wallader, who else ?), that seem most promising. I wrote him a letter, that will be sent at the same time that this one , in which I asked him some questions about the possibilities- of microcode programming and/or utilisation in the HP-41c.Of course, I don't expect Tom to know the answers, I just sta - ted the questions for him to consider them.

Now, before beginning the comments on the material herein included, I want to comment some points- in your letters (and photocopied letters) and PPC TN 3 & 4 , which finally arrived (I say finally, because I didn't receive much material in november, just some things from you). In fact, I am still waiting for PPC October and November. TN 3 and 4 took to long, too, and I was thinking they could be possibly lost), they were full of interesting things (not counting those contributed by me, which , of course, do not have interest at all for me. I already know them, to be sure !):

First on the letters: I was amazed by your extense reply to Hugh: it was some kind of auto-biography, I don't think I understood more than a fraction of it , (but I liked it anyway) because my English is very poor, as - you should have noticed by now. My Spanish is much better , as you will see when you read the 34c book, also included. I like very much the English language: I think of it as the - best language among all world's idioms, for its beauty, its- inmense vocabulary, and its incredibly simple gramatic. Besides, English has the property that almost anything written in English is shorter and takes less space to say the same - than if written in any other language, and this I have had - several occasions to test when translating something written in English to Spanish: it took more words (and often larger- words) to say the same. English would be near perfect, were- it not for a major drawback: its inmensely difficult pronunciation (remember the arguments about how McGechie is pronounced!): this is mostly a problem to us, non-English speaking people. On the other hand , Spanish is a phonetic language , its letters and syllables are always pronounced the same, regardless of the particular word. A very good joke I remember now about english pronunciation is the following: consider - the word "ghoti" ; now, pronounce the gh as in "tough", o as in "women", and ti as in "emotion", what do you get ? fish ! That would never happen in Spanish. By the way, how is my - surname ("Albillo") pronounced in English ? In Spanish, it- is: A (as in "car"),l,bi (as in "bill"),llo (as in "young").

Thank you for your kind words about - me in the letter to Hugh, but I don't consider myself as - being "utterly incredible" !.

When I wrote you my first letter asking for information about NNN's in the HP-67, I was an experienced programmer, having owned an HP-25, and HP-67, and an HP-41c, not to mention several other non-programmables. I also knew two high-level languages: BASIC and FORTRAN (by the way, I rather-prefer BASIC than FORTRAN, but that's another story), and my programming experience was quite high at that time: several collections (which I still keep) of programs for the HP-25 (well over 200), for the HP-67 (over 300, 100 written by myself and 200 from the European Library), and about 20 written for the 41c, not to count BASIC & FORTRAN programs written for the DIGITAL[tm] microcomputer of my University (it is not a Polytechnique, but rather a private university, the). I have always been very fond of Mathematics, specially Numerical Analysis. When I had no calculator at all, I performed all kinds of intrincate calculations by hand, and when I got my first 4-function calculator, I wrote "programs" (sequences of keystrokes) for it. Then I got the HP 25, and everything changed (when I bought the 25, I didn't know a single bit about RPN, and in fact, I had to read the Owner's handbook to learn how to add 2 to 2 in the 25. It was miraculous how, without knowing how to use RPN, and existing so many algebraic calculators, I bought a Hewlett-Packard nevertheless!). Buying the 25 was a great effort for me. My monetary condition has never been good, and the 25 costed 13000 ptas (exactly the value of the 41c right now in the US). To buy a year later the 67, I had to previously sell the 25, and to buy the 41, two years later, I had to sell the 67. It is just logical to expect that, in order to buy the Kangaroo, I must previously sell the 41c and all of its accesories (card reader and 3 modules, and a math pac). That's the reason why I'll never buy the printer, nor the wand, nor any other future accesory HP could possibly make for the 41c. They would be very difficult to sell all at once together with the 41c, and I can't afford the luxury of having two top-of-the-line calculators (or computers) at one time. I expect you to understand the reasons.

Much obligued for the bar code of OTHELLO. I have not tested it yet, but I'll get a wand as soon as I can. (As you will see, not having a printer does not hinder my writing programs which require a printer: 3D PLOT is such an example. The same applies to the wand). The only thing regrettable is that Othello's barcode was headed by a: "V ABILLIO's Othello". I know my first name is far from being common , but my eyes ached when they saw that Abillio!!

Very interesting the letter from Wickes and your reply. I agree with William: I hope you will never become the RN of Australia. I have nothing to object to RN, of course but as William says, I can't consider RN as a friend, as you are to me. You have seen the inmense quantities of material I sent in the past to PPC. Each of those materials were sent together with a letter, in which I gave opinions to Richard, and also asked questions. I don't know if the opinions were taken into account, but the questions were certainly never answered, neither personally nor in the journal. I tried to get some advice, some comments from Richard, but to no purpose. Despite the 20 letters I wrote him, I never got a single note from him I even offered myself to carry a section in the journal about numerical analysis, and gave him full details on my ideas, and the way to go on with that section (kind of information, type of topics, etc, etc, etc, with great detail). I got no response at all, wether "yes, go on" or "sorry, but no" or "are you joking, you poor little beginner". Nothing at all.

Then I realized that someone existed called-

John McGechie (McGeechee), who seemed to know something about NNN's, and I decided to try it. I sent that fellow a letter asking for information. You were kind enough to send back a much detailed answer (several pages), and I decided you were certainly friendlier than RN. When I saw (later) your reports on the Melbourne Chapter newsletter, I thought: "This is my natural - Chapter (regardless of it being in the Antipodes), I must join now" and the rest is not unknown to you. Another point I want to state now: in the past, I had been sending very much material to PPC (extremely very few published yet), original material, and most things sent to you were copies of it. Now, it is absolutely different: I sent you original materials, and PPC has a second priority for me now. The reason is quite obvious: the Club is a great thing, but it is not friendly at all, a given individual has almost no chance at all to communicate with the "White House"(RN & co.), and all he can do is pay, receive periodically PPC CJ, and contribute material that may be or may be not published, regardless of it being a drastic discovery (byte-jumper and the like) or an stupid nonsense (no examples given here, but there are a lot of them).

RN always stated that the solution for the club was growth. I don't agree: growth may be good for the economy of the Club, to make PPC CJ cheaper, but it is bad for the communication and friendship. Imagine Melbourne Chapter to have just 1000 members, writing you, asking you questions, and sending you material, expecting you to do the same for them, or at least, to answer them personally. You would resign at once! RN does not resign, but does not answer, too. That's not good. I wouldn't care having to pay much more than my membership fee if I could get some human satisfaction. Sending blindly materials, to (perhaps) see them published is no satisfaction for me. What I look for is friendship, sharing of ideas. That is what you and Melbourne Chapter are doing for me, and I gladly pay for that (if you think I am not paying, just look at the stamps in this envelope. Sum its value, divide by 73, and you will know how many US $ has costed me to send you this letter).

To resume: please, don't Richardnelsonize yourself ! It would be a pity having to create a Melbourne Chapter Chapter.

Following with Bill's letter: I think it is a little too late for both the PPC Custom ROM and the Synthetic programming book. It is not late for future devices made for the 41c. I'll explain. Obviously, a new machine is due for release in a near future. That machine will have a much expanded memory, and new features, including high-level language, such as BASIC, etc. It will be the new top-of-the-line, and inevitably will send the 41c into oblivion. No crys, please. It has to happen. The same happened to the 67. It was a wonderful machine, for which hundreds of NNN tricks have been discovered. But no serious PPC fanatic will spend time playing or making further discoveries in the 67 if he can buy a 41. I had simultaneously a 67 and a 41c during a month. I dedicated to the 41c that month. The 67 seemed odd, very unfriendly. No alpha prompts, program steps were keycodes, etc. And I had owned that 67 for nearly 2 years, during which I made hundreds of programs for it. Yet, it seemed unfriendly and uncomfortable. I guess exactly the same will happen when we have the Kangaroo, or the like, in our hands, we will forget about the 41c. So, the ROM and the book will be no more than mere curiosities. We would have no time for them, then. Just to discover the bugs and the tricks of our newer machines. So, anything dealing with Synthetics will have no market in the future. The same does not apply to regular (non-fanatic) 41c user's. They'll find interesting every accesory released by HP for their machine (video interface, plotter, tape drive,...) and they will not show a great interest in the new machine, as long as they already have one that suits their needs. I make personalized programs: someone (generally an HP customer) asks me, gives me neccesary

data on required inputs and outputs, and I make the program - for him. For instance, last week I made a program (for the 41) implementing a method to win at the roulette. It's not a joke, the person who asked me for the program was absolutely serious and was an expert gambler, too. He had just heard about the - 41c, its big memory, and its pocket size. He thought he could use the machine to run the calculations he performed (in a much restricted way) by hand, at the casino, and bought the machine. He also bought 4 RAMs (but no card reader). I made the program , a very complex program, requiring full memory, and having 500 lines, extensive indirect control and many options possible - to its user. The program performed in seconds (10), on the go, intrincate calculations (statistics included). I asked the - equivalent of 150 US $ for it, and they were given to me without hesitation. What I am trying to point out is that specialized customers require specialized machines. This man bought the 41c because he needed portability and power simultaneously. He would have never bought an Apple, say, because he nee - ded fast, accurate computations right in front of the roule - tte table, and pressing very few keys, easy to handle (I used key assignments, together with an overlay). Some other people require pipes computation, financial, etc, always very specialized. No one of these people will ever want to hear about - synthetics. Nor about other machines, as long as their 41c - fills the need. These people could possibly buy accesories - for their 41's (the man I mentioned before is thinking about- buying the card reader in a future, to use his 41 to yet another purposes), so the market for the 41 will not dissapear - after the Kangaroo is introduced, but the market for 41c synthetics will not go any further, because people who like synthetics, the fanatics, will inmediately buy the newer machine.

This is getting (or is it becoming?) a little long. I'll stop now (if something else crosses through my mind, I'll include it in a post-data): as you may see, you - are not the only one who "talks and writes too much". And, by the way, I was recognised in my school as being the only one - to know PI to 20 places (3.14159265358979323 84, I still remember), etc. (square root of 5 is 2.236067977499... for the same price)(I once made a list of all known constants I could - remember, and I filled a whole blackboard. It represents no - effort to me to keep them in memory, they simply stay there , and I can recall at any moment that square root of 10 is - 3.162277660, say)

The post-data comes now: the only thing that can save the 41c a little more against the Kangaroo is: a)memory expansion, either via tape drive or via memory extensions a la Jim de Arras, and b) increased speed, either via hardware changes, such as the clock rate, or internal ROMs update, or if we could use microcode as subroutines, or program in - microcode in RAM, directly. Tom's works seem to point at this, c) new and powerful accesories: -tape drive (extended memory) -video interface and, -8K BASIC in ROM. An HP ROM (such as an application ROM), of 8K written entirely in microcode, implementing BASIC. This ROM would also have 1K of RAM built-in for its internal workings, buffers and the like, and it would allow a quite good BASIC to be used in conjunction with the - 2.1 K of RAM of a 41c enhanced with a quad-RAM. I think of - this one as a so good an idea, that I can't see why HP has - not released it yet. Perhaps due to technical problems.

The "letter" part is finished, the rest are comments on the material included herein, and the "technical subjects". Blank cards are much, much, much appreciated, because I had not a single card when you sent those, and was thinking about sending you the programs without cards at all. Your cards avoided that, fortunately.

Best wishes and merry Christmas

-here included are: MISCELLANEOUS

- a photocopy taken from a french journal called L'Ordinateur Individuel. I noticed you requested in a PPC TN methods to add more memory to the 41c (you asked Jim de Arras, in particular.) This article describes completely how to interface an EPROM (Erasable Programmable ROM) to a TI-57, or 58 calculator. The article is full of schematics and instructions, and, as far as I can tell, the method used seems to be not-dependent on the calculator used, becauses it does not interface to the bus, but rather to the keyboard, using a multiplexer. So, to program the EPROM (once interfaced) you simply press the required keystrokes, which are recorded by the EPROM, and can be retrieved later. When reading back the program, the EPROM manages to "press" the neccesary keys, and that can be done at a nominal speed of 20 keystrokes per second, which can be presumably improved on HP machines. The concept seems to work for virtually any calculator, and in particular for the 34c, too . The EPROM stores keystrokes, which result in programs, or data, or whatever (even assignations). Up to 800 or more keystrokes may be storable in a single EPROM. Imagine a 34c expanded to have 800 steps of programs in an EPROM, retrievable at anymoment. The article is written in french, I understand it without problem. If you can't translate it to english your self, ask some friend in your University. It is worth the trouble. Once programmed, the EPROM behaves like a ROM, no energy required to maintain the memory, and it can be erased via ultraviolet light. 2-sided copies

- photocopies of the english section of the User's Library in Europe. 9 pages. If you think they can be useful, I'll send you the french section, the german section, the italian section, even the spanish section if you like, as well as future updates I'll eventually receive. 2-sided copies

- a version of the Othello for the 41c, this one written for the HP-85 computer. This version is exactly the same, except that I didn't include the possibility of making the machine play for the user, or even give the final score, as I made it to test the program for the 41c. The 85 program runs much faster, so I could test different strategies in a few hours, and not in days. The program for the 85 is both fast and very short. If you have an 85 near, you can play Othello very fastly, or give it to an interested friend, to improve the strategy, or make changes, or whatever.

PROGRAMS : 2 programs sent herein

- MLR - multiple linear/polynomial regression : as I stated in a previous letter. Given a set of data, the program computes the coefficients of a linear regression of n independent variables, or the polynomial regression of degree n. This is, it fits, in the least-squares sense, either a linear expression in n variables or a polynomial of degree n, depending on the type of data. The user can add or delete points at any moment, even after the coefficients have been computed. No data cards required, of course. N is limited only by available memory. Once the coeffcients have been computed, you can perform projections based in the regression just computed. Fanatics of statistics will love this one. 3 cards. Printer-compatible.
- CHKERS - Checkers 3.0 : if you liked Othello, you'll probably feel the same about this program. It plays checkers against the user, and includes printing of the board as an option. The program itself is printer-compatible, runs the same with or without printer. If you have a printer, it prints the board, so avoiding all the handling. Without a printer, you must take care of the board yourself, but you save 100 bytes of memory, and runs 30% faster. It plays checkers in the ame

rican style of the game: the board is 8x8, each player has 12 pieces, which always move forward, diagonally. They can either move one square at a time, diagonally (and forwards), or take and adjacent enemy piece, by jumping over it diagonally and - putting the piece on a free location. The jumped piece is taken off the board. Capture is compulsory. Multiple jumps are allowed in a single move. Pieces capture in the direction they move, they cannot capture backwards, of course. If a piece - reaches the opposite side of the board, it becomes a king. A - king moves the same as a normal piece, but it can move both forwards and backwards (always diagonally, of course), and - captures the same, forwards and backwards. The game is won if either side captures all enemy's pieces. A multiple jump is possible when you capture a piece, jump over it, and then, from the location your piece is just now, you can capture an other piece. This can be repeated as many times as possible, always capturing with the same piece. If you have several possible captures at a move, you may select any of them, not - just the one which takes the maximum number of pieces.

The "print board" routine is separated from the main program, so that it can be loaded or not at will. You may change it with equal ease if you like. It is just about 100 bytes, and is recorded on the 2nd side of card 4.

This program is similar to Othello, but :

-it is much longer than Othello, due to its complexity. Do not make any changes at all. There are sequences that, at a first glance, seem stupid, and easy to improve. Do not do so, or you'll find that your modified program makes illegal moves.

-its strategy is fairly good for such a complex game. It plays an acceptable game. Due to its complexity, it takes from 1 to 2 minutes to make each move (no printer. 35 % more with a - printer attached), for which it has to analyse the position, the better posibilities, etc. You'll find it challenging.

-It provides a YOU WIN message, but no I WIN. If the machine happens to beat you, it will continue to request for your move, even if you have no piece at all. Simply be kind, and recognise the defeat or draw by stopping the game.

There are many other things to say about CHECKERS 3.0. See enclosed documentation. It also allows the user to select who moves first. Unlike OTHELLO, it does not check the legality of your moves, nor allows the possibility of playing for the user. The board, if printed, is always printed after HP moves, cannot be printed after your moves, to - force you to save paper and time.

Hope it will be received by all members of the glorious Melbourne Chapter with the same kindness as OTHELLO.

ROUTINES : a short routine which I claim it to be the first - one using the PPC ROM. It performs the same as VAL performs in the 85. Given a number, written in alpha using alpha characters, it puts its numeric value in X. For instance, the alpha characters 1.23456789E-41 result in 1.23456789E-41 in X. It uses no register at all, is short, and allows the user to treat mixed alpha: for instance, COST=23.48 gives 23.48 in X and leaves COST in alpha. It may be useful that way.

TECHNICAL SUBJECTS

This is to be long, but later. I remember now that previously I must comment:

HP-34 BOOK : Here, at last, is included a copy of the book for the 34c. It has some NOPs marked with a pencil - for you to notice. My name appears nowhere, as it is not the policy of HP to include the names of the programmers in their software. I managed, however to leave my "mark" indelibly -

stamped, in the form of related to me. For instance, Of course. no one would ever believe that is just luck to find so it is - clear that they identify the author. (It is better, in the 41c book, because the alpha capability allowed me to include , etc.)

The book is fairly well designed, figures and text are not mine (this is, they were typed or photocomposed by people at the press) (of course I originally wrote the texts and figures !). It was very much work, , but it has to be done, and I did it. Due to my typing the listings, they are error-free (except for 2 steps) and all programs can be keyed from the book and executed - without trouble.

The programs are already known to you (the titles, I mean), and cover a wide range of mathematical applications. They are mostly unique, except perhaps for the Triangle Solutions, which is a clear topic (though different from other HP's triangle solution programs). The original set of programs contained 2 more programs. Both were deleted from the final book, to make it strictly mathematic.

There are quite good programs among them: the longest is the Polynomial Solutions, which took a great effort to make it fit the 34c. The harmonic analysis program can be improved, as I discovered shortly after the book was finally released. Too late, then, to make changes. The improvements are not drastic, however, just saving some bytes. The "Polynomial fit through equally spaced points" is very good: it involves nested looping and other sophisticated techniques. The same is true for "Summation of series".

The programs are written to take full advantage of the characteristics of the 34c. Most of them allow the user to use INTEGRATE or SOLVE on the computed fit, etc.

The eigenvalues program uses the SOLVE key to find a real eigenvalue, after which the equation is reduced to 2nd degree. The Double integral program uses the built-in INTEGRATE function to compute the innermost integral, and Gaussian integration for the outermost one.

"Talks and writes too much" That's you. And me. I'll shout up. Let's finish.

Well, John, I hope you'll find no trouble to - translate the spanish text to English (believe me, it is - trivial ... for me). If you happen to find any problem, ask me for the translation. Show the book to Richard Collet, he will like it, I'm sure. I cannot afford sending another copy to RN, but you can probably make a good photocopy of the original and send this photocopy to him, for his library. (By the way, I want you to personally keep the book. Do not give it to any person, nor even to RN. The book is <u>yours</u>, not of the Melbourne Chapter)

Till my next letter (early 81)

I noticed you (and Phill Jury) noticed my new method to break PRIVATE, and, in your opinion, was eclipsed by the good old byte-jumper (a short disquisition: I realize the importance and usefulness the byte jumper has, unmatched by any other technique, but, I don't exactly know why, I have never liked it, and the only use I've made of it is its capability of recalling without normalization. Were not for this characteristic, it would have no use for me at all. When I have to load a synthetic instruction line, I always use the STO b, RCL b method, never the byte jumper. And its other characteristics are most useful), you are right, I knew the method to break PRIVATE using the byte jumper as soon as I knew of the byte jumper itself. It is obvious.

My new method to break PRIVATE was not presented as the best, or even as an easy one, but just as another method to do it. But I hope that you'll find my efforts worthwhile, because I've managed to discover still another method to do it, and this one is, by far, the easiest method possible, because it does not require synthetics at all !!! anyone, having a bug-free, bare bones 41c can break any private program as easily as compute the factorial of 69. No synthetics, no byte jumpers, nothing but a fixed procedure, without need for a card reader, without need for any assignment at all. It can't be easier than that, you'll agree.

In fact, it is so easy that PRIVATE is not private any more. Now, anyone can have all his programs recorded private, to avoid disturbing them, and, if needed, unPRIVATE them using a few keystrokes. Because of this possibility, I ask you to finally show the membership the exact method once and for all. It is very useful now, and the stupid fears are more than balanced by the benefits to everybody. Now PRIVATE is finally useful to everybody, not to protect programs from being copied, but to protect them from being inadvertently disturbed. No need to keep 2 copies, one private, one not. Who would need a second copy when he can unprivate the privated one pressing 2 keys, say ? I myself use it now quite a lot. All my programs are private now. I even used this method at the SIMO. Why on earth should we keep this method secret ? It is useful, and it is not fair to our member partners to keep knowledge restricted. If not, Bill Wickes could have kept secret everything. Tom could have kept secret his microcode exploration, etc.

Believe me. Now that PRIVATE is finally useful, we can no longer (honestly) keep the defeating methods for ourselves. It would not be fair. To benefit the community of users, we shall not hesitate to demolish the PRIVATE idol.

You must be thinking by now: But, how do - you do it ???. Well, here included is an article titled "Bug 9 - The catalog bug". It describes a bug I've found (I am not sure if it is exactly a bug. Any comments will be very much appreciated) which is present in all and every 41c, which does not require synthetics at all, and that is useful for:

- -breaking PRIVATE in any machine from the keyboard
- -getting into the assignments registers in any machine, without need to do module pulling or the like. Just keyboard operations.
- -of course, create the byte jumper in a bare bones, bugless, RAMless 41c, through keyboard operations.
- -maybe opens a new door to the microcode operations of the 41c. Thanks to bug 9, new odd behaviours are coming to light now.

I should point out once more than bug 9 is present in every machine, that involves no synthetics, and that requires just keyboard operations, no module pulling or the like.

Interested ? Read the article. I wait for your comments. I think I can take credit for this one, for as far as I know, nothing is known to the membership, no one is aware of the method, but myself.

BUG 9 - THE CATALOG BUG

Introducing here what I call (if it is inexact, let me know ...) bug 9. This bug, which is present in all and every 41c, will be described in a moment. It proves useful to:

- create the byte jumper in a bug-free, bare bones 41c. No RAMs or other peripherals needed. Just a 41c, any 41c
- of course, free access to the assignments registers at any moment, in any 41c. This can be done without disturbing anything in the machine. Once in the assignments registers you can create the byte jumper or any other thing easily.
- break a PRIVATE program. Nothing required, just a few keystrokes. No synthetics needed, just normal functions in normal operation.
- access to some new features not fully understood at this stage, that may reveal a new open door to microcode routines.

To wit: bug 9 allows, using normal operations, without synthetics, the possibility to create synthetics using just a 41c, breaking PRIVATE without synthetics being ever needed, and some other things.

a) description of bug 9

Bug 9 is readily understood. Imagine you are listing CAT 1, while in program mode. At any moment, you manually stop the catalog, using R/S, and press the XEQ key : you will see 00 XEQ _ _ instantaneously. Why the 00 line ? As the catalog progresses, the program pointer is positioned at the corresponding label or end. But it does not know the line number (it has to count from the beginning of the program to know this), so as soon as you press the XEQ key, that catalog is inmediately disabled, the XEQ appears in the display, and it gets a 00 as its corresponding line number. If you now fill the prompt with the name or number of a routine, the display updates the 00 to the real line number, say 51 or 317. But the important thing here is that the 41c does not know the line number, and does not calculate this number until the instruction is recorded. The 00 is the logical consequence of the proper number not being present in register e, because as the catalog progresses, the program pointer is updated, but the line number is not, as it would be rather time-consuming if it were computed between catalog entries. Thus, it is set to 0.

Now, what happens if the XEQ is filled with the name of a non-recordable mainframe function (such as PACK) that can't be loaded into program memory ? of course, the display is updated to show the name (and prompts) of the function, this time without line number, so that the line number remains as 00 as it is not computed (this computation takes place after the function has been loaded into memory).

But (and what a "but" !) if the mainframe function is DEL, once you fill the prompts with 001 (say), the function is performed, the present line is deleted, the machine acknowledges that lines had been deleted, and the line number is set to the previous minus one, while the pointer is placed at the line before the one which was deleted. Simple, isn't it ? The only thing is that the present line was 00, so the line number becomes 4094, and the pointer jumps to the END of the previous program (if any) !!! Now, for the applications:

b) How to access the assignments registers

You can access the assignments registers using bug 9, without any need for other bugs or synthetics at all. Follow these guidelines:

-go to the first program in program memory, and insert an END at line 01 (this is, create line 01 END) so as to make a dummy END at the top of program memory which can be deleted later. This can be done so:
CAT 1 (stop at once), RTN, PRGM, END (XEQ END), PRGM
this is useful to avoid deleting the label of the first program in program memory. If this is irrelevant, you can skip this procedure.

-now, set PRGM mode and CAT 1: you should stop it immediatly, as soon as you see the first END we created before. This is, stop the catalog at its first entry.

-XEQ (you should see 00 XEQ _ _), alpha,D,E,L,alpha (you should see DEL _ _ _), 001. See line 4094 any.

-Now, BST to see the contents of the first assignment register. If you had, say, SIN assigned to LOG and COS assigned to LN, you should see:

```
4092 STO 01     key location, LOG
4091 SIN        SIN assigned
4090 LBL 03     separator
4089 -          key location, LN
4088 COS        COS assigned
4087 LBL 03     separator
4086 ""         assignment register
```

It was easy, and didn't require synthetics at all.

c) How to create the byte jumper in a bugless,bare bones 41c

You can create the byte jumper without any need for a memory module. You just need a 41c.

Let's assume a 41c without memory modules (if you have modules, take them out). Furthermore, let's assume that MEMORY LOST has just taken place.

-assign: ASN SIN (LN), ASN SIN (LOG)
-set program mode: see 00 REG 45
-now, CAT 1 (stop at once, immediately), see .END. REG 45
-XEQ (see 00 XEQ _ _), alpha,D,E,L,alpha (see DEL _ _ _)
-fill the prompt with 001 : see 4094, then .END. REG 45
-now BST : see 4093 DEC, BST: see 4092 -
BST : see 4091 SIN, delete this using backarrow,
see 4090 LBL 03, delete it using backarrow,
see 4089 STO 01
-now, press: alpha, A, alpha, see 4090'A
-now, GTO .. , see 00 REG 45, out of program mode

the work is done. Set USER mode and press and hold the LN key, to see XROM 05,01. The byte jumper has been assigned to the LN key without RAMs being ever needed

d) How to break any PRIVATE program

Everyone knows that there are a great variety of methods to break PRIVATE. But this one is the best among them all, as it does not require synthetics at all, nor programming overhead, just normal functions during normal operation. Follow these guidelines:

- load the PRIVATE program from a card into memory, GTO ..
- set PRGM mode
- CAT 1. AS SOON AS IT REACHES .END. REG nn, stop it using R/S. This is, stop the catalog as soon as the END of the

PRIVATE program has been seen. Or, in other words, stop it when you see the .END. for the first time. If you wait for the catalog to stop by itself, it won't work.

-now, press XEQ (you should see 00 XEQ _ _), alpha,D,E,L, alpha, (you should see DEL _ _ _), 001.

you should see 4094 END. This is the END of the PRIVATE program, so simply SST and you'll be seeing the very beginning of it. You can now see the whole PRIVATE program using SST in PRGM mode. You cannot use BST, nor record, nor delete anything, but if you pass now cards thru the card reader, the PRIVATE program will be recorded onto them, without PRIVATE this time, so that you can now clear the still PRIVATE program from memory, load the unprivated version from the cards, and make any changes you want.

So, this method is the best of all: nothing but a few keystrokes, no synthetics, no nothing. Anyone can now break a PRIVATE program at once, without creating assignments or having to load cards. Just his/her 41c and a little skill.

PRIVATE has finally been defeated

e) Odd behaviours related to bug 9

We will see some rather obscure aspects of bug 9, the catalog bug. I am very far from fully understanding these, but a few examples will give you the clues you'll need to begin experimenting a little.

Experiment 1 : assuming you have the card-reader plugged in (and not the printer).

CAT 2 , stop at MRG using R/S, XEQ BST , see CARD READER press backarrow, see GR CARD READ. Pressing ENTER takes us out of the catalog.

This procedure performs the same with all catalogs (including synthetic ones). If SST or BST are executed using XEQ in the midst of the catalog, pressing backarrow causes then a wrap-around display, that often stays steady a while, then scrolls right, showing a crosshatch in the left side of the display. Pressing some other keys instead of backarrow, causes DATA ERROR or NONEXISTENT to appear, so that it seems that some microcode is accessed if a catalog is enabled-disabled-enabled by this procedure. For instance, using CAT 3, stop when you see ADV, then XEQ BST, see ACOS, press SST, see DATA ERROR.

Experiment 2 : To show this one, assume a MEMORY LOST.

-set program mode: see 00 REG 46
-now, CAT 1 (stop at once, immediately), see .END. REG 46
-set alpha , press A (see 01▸A)
B (see 01▸--) (what happened to AB ?)
C (see 01▸---)
D (see 01▸---▣)
E (see 01▸---▣A) (here's the A !!)
F (see 01▸---▣AB) (and the B too !)

you may experiment at will. The box star is the text byte. You can quit at any moment, so creating a line of nulls in program memory rather easily. You can play with append too. The results vary a lot, you can play with it if you want.

Welcome, bug 9.

VALENTIN ALBILLO (4747)

LINEAR/POLYNOMIAL REGRESSION by Valentin Albillo (4747)

This program computes either a linear regression of n variables, or a polynomial regression of degree n, where n is chosen by the user.

The user decides what kind of regression is to be computed (linear/polynomial), and provides a set of data, either of the form:

$(x_1,x_2,x_3,\ldots,x_n,y)$ (multiple linear r.)

or:

(x,y) (polynomial r.)

Once all data have been entered, the program computes the coefficients of the best fit (in the least squares sense) of the form:

$$y = a_o + a_1x_1 + a_2x_2 + \ldots + a_nx_n \quad \text{(lin.)}$$

or:

$$y = a_o + a_1x + a_2x^2 + \ldots + a_nx^n \quad \text{(pol.)}$$

So this program is a much upgraded version of the similar one found in the Statistics ROM, which is limited to 3 independent variables, or degree 3, whereas this program is limited only by available memory.

It uses the following method (for the polynomic case):

-to find the coefficients of the best polynomial fit of the form: $y = a_o+a_1x+a_2x^2+\ldots+a_nx^n$, we solve the system:

$$\begin{aligned}
&s_oa_o + s_1a_1 + s_2a_2 + \ldots + s_na_n = t_o\\
&s_1a_o + s_2a_1 + s_3a_2 + \ldots + s_{n+1}a_n = t_1\\
&\ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\\
&s_na_o + s_{n+1}a_1 + \ldots\ \ldots\ \ldots + s_{2n}a_n = t_n
\end{aligned}$$

where $s_k = \sum_1^n x_i^k$, $t_k = \sum_1^n y_i x_i^k$

and a similar treatment is applied in the linear case.

The system is solved using matrix inversion, using a method that requires no more registers than those used by the original matrix. Which is more, once computed the coefficients, it is possible to add or delete data, without having to reintroduce anything at all. Similarly, you can compute predictions (or extrapolations) based in the regression just calculated.

Characteristics :-number of independent variables limited only by available memory. The same applies to the maximum degree if polynomial case chosen.

-built-in SIZE detector. Prompts with a SIZE nnn message if present SIZE is insufficient to run the program

-allow insertion or deletion of data after the coefficients have been computed.

-allows predictions based in the regression.

This program is 375 lines, 651 bytes long, so it requires at least one module to be run. Uses all flags from 00 thru n (where n is either the degree or the number of independent variables). It uses local labels A thru E to select the required functions, so it must be run in user mode.

The required SIZE for degree n (or n ind. var) is SIZE $n^2+4n+14$. The built-in SIZE detector prompts if the SIZE is less than this value.

Warnings : due to the method used to compute predictions, you cannot enter x=0 to make a prediction of y(0). This should be no problem at all, since y(0) is exactly a_o. If desired, input a very small value, say 1E-9 or so. The linear case has no similar limitation.

HOW TO USE IT

1) -load the program,
2) -execute: XEQ"MLR" → LIN ? , asks you if you want a multiple linear regression.

-if you want the linear case, simply: R/S
-if you want the polynomial case: N , R/S

a) if the linear case has been selected:

→ LIN → N. VAR. X ? , asks you for the number of independent variables (x_i)

n , R/S → if SIZE nn appears, your present SIZE is insufficient to run the program. Resize to the value indicated, then R/S to resume

→ POINT 1 → X1=? , enter the value of x_1 in the first point.

x_1, R/S → X2=? , enter the value of x_2 in the first point

x_2, R/S → ... ...

... ... → XN=? , enter the value of x_n in the first point

x_n, R/S → Y=? , enter the value of the dependent variable, y

y, R/S → POINT 2 → X1=?, repeat this process until all data points have been entered:

... y R/S → POINT k → X1=? , all points have been entered.

b) if the polynomial case has been selected:

→ POL → DEGREE? , asks you for the degree of the polynomial to be fitted

n, R/S → if SIZE mm appears, your present SIZE is insufficient. Resize to the indicated value then R/S to continue:

→ POINT 1 → X=? , enter the x of the first point

x , R/S → Y=? , enter the y of the first point

y , R/S → POINT 2 → X=?, repeat this procedure until all data points have been entered:

... y R/S → POINT k → X=? , all points entered.

3) - now, to compute the coefficients of

$$y = a_o + a_1x_1 + a_2x_2 + \dots + a_nx_n$$

or

$$y = a_o + a_1x + a_2x^2 + \dots + a_nx^n$$

simply: press C → coefficients are computed now. After a while, you'll get:

→ AØ=its value
R/S → A1=its value
... ...
R/S → An=its value

4) - to review coefficients once again, press D

5) - to perform predictions based in the computed regression,

press E → if polynomial: X=? , enter the value of X
x, R/S → Y=its value
to perform another prediction, simply: R/S → X=?, etc.

- if linear :

→ X1=? , enter the value of x_1

x_1, R/S → X2=? , enter the value of x_2

... ... → Xn=? , enter the value of x_n

x_n, R/S → Y=its predicted value

to perform more predictions, press R/S → X1=?, etc.

6) to add more points to the set:

- press B: wait some time while the matrix is reinverted again to get the original matrix. Once this is done, you are back were you left introducing points, and the calculator prompts for the new point:

→ POINT → either X=? or X1=?, depending on the type of regression being computed.

proceed to introduce the new point, as in steps (a) or (b). Once the point processed, ignore the prompt for point number xx, and either go to step (3) or to (6), as required.

EXAMPLES : (1) find the polynomial of degree 2 that passes through (0,2), (1,4),(2,16). Find y for x=8

XEQ "MLR" → LIN ? , N, R/S → POL → DEGREE?, 2, R/S →
→ POINT 1 → X=? , 0 R/S → Y=? , 2 R/S →
→ POINT 2 → X=? , 1 R/S → Y=? , 4 R/S →
→ POINT 3 → X=? , 2 R/S → Y=? ,16 R/S →
→ POINT 4 → X=?

now, C → AØ= 2.0000 , so the polynomial is
R/S → A1=-3.0000
R/S → A2= 5.0000

$$y = 2 - 3x + 5x^2$$

to find y(8) , E → X=? , 8 R/S → Y=298.0000, y(8)=298

- - - - - - - - - - - - -

(2) find a linear expression in two independent variables that passes through (0,0,5),(1,0,7),(0,1,8),(1,1,10)

XEQ "MLR" → LIN ? , R/S → LIN → N. VAR. X ? , 2 R/S →
→ POINT 1 → X1=?, 0 R/S → X2=?, 0 R/S → Y=?, 5 R/S →
→ POINT 2 → X1=?, 1 R/S → X2=?, 0 R/S → Y=?, 7 R/S →
→ POINT 3 → X1=?, 0 R/S → X2=?, 1 R/S → Y=?, 8 R/S →
→ POINT 4 → X1=?, 1 R/S → X2=?, 1 R/S → Y=?,10 R/S →
→ POINT 5 → X1=?

C → AØ=5.0000 , so the linear expression is
R/S → A1=2.0000
Find the va- A2=3.0000

$$y = 5 + 2x_1 + 3x_2$$

to find y(3,3) , E → X1=? , 3 R/S → X2=?, 3 R/S → Y=20.0000

so y(3,3) = 20

- - - - - - - - - - - - - - - - -

(3) Fit a 4th-degree polynomial to the following set of data:
(.1,5.1234),(.2,5.3057),(.3,5.5687),(.4,5.9378),(.5,6.4370),
(.6,7.0978),(.7,7.9493),(.8,9.0253),(.9,10.3627).
Find the value for x=0, x=.45, and x=1.

XEQ "MLR" → LIN ?, N R/S → POL → DEGREE? , 4 R/S → SIZE 46,
XEQ SIZE 46 R/S → POINT 1 → X=? , .1 R/S → Y=?, 5.1234 R/S →
→ POINT 2 → X=? , .2 R/S → Y=?, 5.3057 R/S →
→ POINT 3 → X=? , .3 R/S → Y=?, 5.5687 R/S →
→ POINT 4 → X=? , .4 R/S → Y=?, 5.9378 R/S →
→ POINT 5 → X=? , .5 R/S → Y=?, 6.4370 R/S →
→ POINT 6 → X=? , .6 R/S → Y=?, 7.0978 R/S →
→ POINT 7 → X=? , .7 R/S → Y=?, 7.9493 R/S →
→ POINT 8 → X=? , .8 R/S → Y=?, 9.0253 R/S →
→ POINT 9 → X=? , .9 R/S → Y=?,10.3627 R/S →
→ POINT 10 → X=?

-now, compute coefficients: C → AØ=5.0010
R/S → A1=0.9919
R/S → A2=2.0177
R/S → A3=2.9895
R/S → A4=0.9993

so, the required polynomial is:

$$y = 5.0010 + 0.9919x + 2.0177x^2 + 2.9895x^3 + 0.9993x^4$$

to find predictions: E → X=? , .45 R/S → Y=6.1693
R/S → X=? , 1 R/S → Y=11.9994
R/S → X=? , E-9 R/S → Y=5.0010
to test, R/S → X=? , .7 R/S → Y=7.9493, exact

That's all. Good programming, folks !

VALENTIN ALBILLO

```
01 LBL"MLR"
02 SF 27
03 CF 17
04 CF 29
05 SF 21
06 CLRG
07 FIX 0
08 "LIN ?"
09 AON
10 PROMPT
11 AOFF
12 ASTO X
13 "N"
14 ASTO Y
15 FC? 55
16 CF 21
17 ADV
18 X=Y?
19 GTO 47
20 "LIN"
21 AVIEW
22 PSE
23 CF 16
24 "N. VAR. X ?"
25 GTO 49
26 LBL 47
27 "POL"
28 AVIEW
29 PSE
30 SF 16
31 "DEGREE?"
32 LBL 49
33 PROMPT
34 1
35 +
36 STO 00
37 X2
38 RCL 00
39 2
40 ✱
41 +
42 10
43 +
44 SF 25
45 RCL IND X
46 FS?C 25
47 GTO 48
48 1
49 +
50 FIX 0
51 "SIZE "
52 ARCL X
53 PROMPT
54 LBL 48
55 CF 00
56 RCL 00
57 XEQ 00
58 RCL 00
59 X2
60 LAST X
61 +
62 10
63 +
64 ENTER
65 ENTER
66 RCL 00
67 +
68 1
69 -
70 1 E3
71 /
72 +
73 STO 01
74 STO 09
75 GTO A
76 LBL B
77 SF 17
78 LBL A
79 FS?C 00
80 XEQ I
81 CF 00
82 FIX 0
83 RCL 10
84 1
85 +
86 FC? 55
87 CF 21
88 ADV
89 "POINT "
90 FC? 17
91 ARCL X
92 AVIEW
93 PSE
94 SF 21
95 RCL 09
96 STO 01
97 1
98 STO 02
99 STO IND 01
100 ISG 01
101 SF 21
102 FS? 16
103 GTO 50
104 LBL 51
105 "X"
106 ARCL 02
107 "⊢=?"
108 PROMPT
109 STO IND 01
110 ISG 02
111 X()Y
112 ISG 01
113 GTO 51
114 GTO 53
115 LBL 50
116 "X=?"
117 PROMPT
118 ENTER
119 ENTER
120 ENTER
121 1
122 LBL 52
123 ✱
124 STO IND 01
125 ISG 02
126 X()Y
127 ISG 01
128 GTO 52
129 LBL 53
130 "Y=?"
131 PROMPT
132 STO IND 01
133 RCL 09
134 STO 01
135 1 E-3
136 +
137 STO 02
138 10
139 STO 03
140 LBL 15
141 RCL IND 01
142 RCL IND 02
143 ✱
144 FS? 17
145 CHS
146 ST+ IND 03
147 ISG 03
148 CLX
149 ISG 01
150 GTO 15
151 RCL 09
152 STO 01
153 ISG 02
154 GTO 15
155 CF 17
156 GTO A
157 LBL I
158 RCL 00
159 1
160 -
161 1 E3
162 /
163 STO 01
164 STO 05
165 LBL 91
166 FS? IND 01
167 GTO 90
168 RCL 01
169 RCL 01
170 RCL 00
171 ✱
172 +
173 10
174 STO 06
175 STO 07
176 +
177 RCL IND X
178 X=0?
179 GTO 90
180 1/X
181 STO IND Y
182 STO 04
183 X()Y
184 RCL 01
185 ST+ 07
186 -
187 RCL 05
188 STO 03
189 STO 02
190 +
191 STO 08
192 LBL 04
193 RCL 02
194 RCL 01
195 X=Y?
196 GTO 07
197 RCL 03
198 X=Y?
199 GTO 06
200 RCL IND 08
201 RCL IND 07
202 ✱
203 RCL 04
204 ✱
```

```
205 ST- IND 06
206 LBL 06
207 1
208 ST+ 06
209 ST+ 08
210 ISG 03
211 GTO 04
212 RCL 00
213 ST- 08
214 LBL 05
215 ST+ 07
216 RCL 05
217 STO 03
218 ISG 02
219 GTO 04
220 GTO 14
221 LBL 07
222 RCL 00
223 ST+ 06
224 GTO 05
225 LBL 14
226 STO 06
227 RCL 00
228 ST* 06
229 RCL 01
230 ST+ 06
231 *
232 +
233 10
234 ST+ 06
235 +
236 STO 02
237 LBL 08
238 RCL 01
239 RCL 03
240 X=Y?
241 GTO 09
242 RCL 04
243 ST* IND 06
244 CHS
245 ST* IND 02
246 LBL 09
247 RCL 00
248 ST+ 06
249 ISG 02
250 X()Y
251 ISG 03
252 GTO 08
253 SF IND 01
254 RCL 05
255 STO 01
256 LBL 13
257 FC? IND 01
258 GTO 91
259 ISG 01
260 GTO 13
261 RCL 00
262 LBL 00
263 CF IND X
264 DSE X
265 GTO 00
266 RTN
267 LBL 90
268 ISG 01
269 GTO 91
270 "ERROR"
271 PROMPT
272 LBL C
273 XEQ I
274 10
275 STO 03
276 RCL 09
277 STO 01
278 1.001
279 RCL 00
280 *
281 -
282 STO 02
283 STO 04
284 CLX
285 LBL 01
286 RCL IND 02
287 RCL IND 03
288 *
289 +
290 ISG 03
291 STO 06
292 ISG 02
293 GTO 01
294 STO IND 01
295 RCL 04
296 STO 02
297 CLX
298 ISG 01
299 GTO 01
300 LBL D
301 SF 21
302 ADV
303 CLX
304 STO 02
305 RCL 09
306 STO 01
307 LBL 02
308 FIX 0
309 "A"
310 ARCL 02
311 "⊢ ="
312 FIX 4
313 ARCL IND 01
314 AVIEW
315 ISG 02
316 STO X
317 ISG 01
318 GTO 02
319 CLX
320 RTN
321 LBL E
322 SF 21
323 FIX 0
324 ADV
325 RCL 09
326 STO 01
327 FS? 16
328 GTO 23
329 1
330 STO 02
331 RCL IND 01
332 STO 03
333 ISG 01
334 LBL 25
335 "X"
336 ARCL 02
337 "⊢=?"
338 PROMPT
339 RCL IND 01
340 *
341 ST+03
342 ISG 02
343 X()Y
344 ISG 01
345 GTO 25
346 RCL 03
347 GTO 24
348 LBL 23
349 "X=?"
350 PROMPT
351 1/X
352 ENTER
353 ENTER
354 ENTER
355 RCL IND 01
356 ISG 01
357 LBL 26
358 *
359 RCL IND 01
360 +
361 ISG 01
362 GTO 26
363 X()Y
364 RCL 00
365 1
366 -
367 y^x
368 /
369 LBL 24
370 FIX 4
371 "Y="
372 ARCL X
373 AVIEW
374 GTO E
375 END
```

```
+++++++++++++
+ 375 lines  +
+            +
+ 651 bytes  +
+++++++++++++
```

SIZE $n^2+4n+14$

CHECKERS 3.0 - by Valentin Albillo (4747)

This program allows the user to play a game of checkers against the 41c. Checkers is a most popular - game, known all over the world, though the rules vary slightly- from country to country.

This program, similarly to OTHELLO, is printer-compatible (runs with or without printer), but if a printer is present, it will print the board automatically, freeing - the user of the handling of the board. The printer routine is - independent from Checkers 3.0 , so members without a printer do not have to make any changes to the program.

```
I MOVE
FROM 06 TO 15

  0 1 2 3 4 5 6 7
7 - O - O - O - O
6 - - O - O - O -
5 - O - O - O - O
4 O - - - - - - -
3 - ▒ - ▒ - - - -
2 - - ▒ - - - ▒ -
1 - ▒ - ▒ - ▒ - ▒
0 ▒ - ▒ - ▒ - ▒ -

FROM?
```

It allows the user to select who moves first. It plays as fast as possible for such a complex game, taking from one to 2 minutes to perform its move (if no printer. Printer - slows down the execution some 35%). The strategy used is quite good, offering a real challenge for everyone.

Checkers is played on an 8x8 board. One player plays white (represented by O) , the other plays black (represented by the checquerboard character). In this version, you - play always black, though you can select who makes the first move.

Each player has 12 pieces at the beginning, set as in the standard position (see- below). To make a move, you should place one of your pieces into one of the adjacent squares to the one in which your piece - was, and diagonally in the direction of the enemy (this is, always forwards for you, always backwards for the machine pieces) This is, you can move a piece by placing it into a free location situated diagonally adjacent (and in front) of the location where your piece is actually. Your pieces always move forwards, machine - ones always backwards (this is, each - player moves in the direction of the - enemy side). Or you can take a piece of the enemy, which has to be located diagonally adjacent to one of your pieces, by jumping with your piece over it, and placing your piece in a free location - (the next free location diagonally adjacent to the enemy's piece).

```
XEQ "CHKERS"
  0 1 2 3 4 5 6 7
7 - O - O - O - O
6 O - O - O - O -
5 - O - O - O - O
4 - - - - - - - -
3 - - - - - - - -
2 ▒ - ▒ - ▒ - ▒ -
1 - ▒ - ▒ - ▒ - ▒
0 ▒ - ▒ - ▒ - ▒ -

HP 1ST?
```

Standard position

```
  0 1 2 3 4 5 6 7
7 - - - # - O - O
6 - - O - O - O -
5 - - - - - # - O
4 - - O - - - - -
3 - - - - - ▒ - ▒
2 ▒ - 8 - - - - -
1 - - - ▒ - - - ▒
0 ▒ - 8 - - - ▒ -
```

Multiple jumps are allowed: when you capture a piece, jump over it to the next free location and, if from that location you can make another capture with the same piece , you - must do it, taking as many pieces on the move as you legally can.

Capture is compulsory: if you can take a piece, you must do it. However, if - you have several different captures possible you may select any of them, not just the one which results in the maximum number of pieces taken. Pieces capture in the direction - they move. If a piece reaches the opposite - side of the board (row 0 or 7), it becomes a king. A king moves and takes exactly as a normal piece, but can do it both - forwards and backwards (always diagonally, of course). The kings may be distinguished from normal pieces as follows: white one are represented as 8, black ones as a crosshatch ▒. See the figure at the left. Each location is represented by a 2-digit number following the coordinate system of the figures: for instance, 15 means 1 horizontal, 5 vertical. In the figure, there is one black king at 37 and another at 55, one white king at 22, another at 20. If white moves, it can play from 20 to 42 (taking the black piece) to 64 (taking another piece), or from 46

to 64 (taking the black king) to 42 (taking another black piece). Or even from 22 to 40 (taking the black piece). Or from 66 to 44 (taking a black king) to 62 (taking another piece). No other moves are legal, as capture is compulsory. If black plays, he may move from 37 to 15 (taking a piece) to 33 (taking another piece) to 11 (taking a white king). A good example of multiple jump !!, or move from 31 to 13 (taking the king) to 35 (takes another - piece) to 17 (taking still another, and becoming a king). That one is even better !. This should make clear the mechanics of - multiple jumps. And how a piece becomes a king, too.

HOW TO USE

```
XEQ "CHKERS"

   0 1 2 3 4 5 6 7
7 - O - O - O - O
6 O - O - O - O -
5 - O - O - O - O
4 - - - - - - - -
3 - - - - - - - -
2 ▒ - ▒ - ▒ - ▒ -
1 - ▒ - ▒ - ▒ - ▒
0 ▒ - ▒ - ▒ - ▒ -

HP 1ST?
N                 RUN
FROM?
               02 RUN
TO?
               13 RUN
I MOVE
FROM 15 TO 04
```

-if you have a printer, load the printing - rotine first, then GTO.., and load the main program. Set the printer to NORM position.

-if you have no printer, load just the main program.

The program is 502 lines (760 bytes)
the routine is 56 lines (102 bytes)

and requires SIZE 084 to run. If you use - the printer routine, you'll need 3 RAMs, if you do not print, do not load the printer routine, and you'll need just 2 RAMs.

(If you do not intend to print the board , unplug the printer now)

now, XEQ "CHKERS" → the board will be printed now, reflecting the standard initial position, and you'll be prompted:

→ HP 1ST?

```
   0 1 2 3 4 5 6 7
7 - O - O - O - O
6 O - O - O - O -
5 - - - O - O - O
4 O - - - - - - -
3 - ▒ - - - - - -
2 - - ▒ - ▒ - ▒ -
1 - ▒ - ▒ - ▒ - ▒
0 ▒ - ▒ - ▒ - ▒ -

FROM?
```

-if you want HP to make the first move, press R/S. I MOVE appears while the machine thinks its move.

-if you want to make the first move, press N, then R/S → FROM?

a) if the machine moves (I MOVE in the display), it will think its move for a while (from 1 to 2 minutes), then print:

I MOVE → FROM xx TO yy

(where xx, yy are the coordinates of the initial location of the moved piece, and the location where it moves to)

if it is a multiple jump, TO zz appears, showing the new locations where the piece is placed successively.

If a printer is present, the resulting board position is printed now. Otherwise, actualize the board yourself, removing the jumped pieces from the board, if necessary. Once the board is printed, the machine prompts for your move: → FROM?

b) if you move: (you have been prompted by FROM?)

-input the xx of the location where the piece you want to move is now: xx R/S → TO?

-input the yy of the location where the piece moves to: yy R/S

(very important: your move is not tested for legality, so you - must be very careful not to make illegal moves. Remember, always move diagonally and forwards (except if a king is moved, which can move backwards, too), one square at a time, except when taking a piece, jump over it. Capture is compulsory.)

-if you took a piece, the machine asks you for another jump (making thus possible a multiple jump). If you can jump

once more (over another enemy piece, of course), input the zz coordinates of the location where the piece moves to.

Thus, the sequence is: the machine prompts:

+TO? , enter coordinates of additional jump

zz R/S → +TO?

the machine continues asking for yet another jump. If you cannot jump any more, or if you cannot make a multiple jump at all, answer the +TO? prompt with a minus one.

+TO? , -1 R/S → I MOVE

examples will totally clear this. Once your move is done, the I MOVE display appears, and the machine proceeds to make its move. This sequence is repeated until the game ends. The player who takes all enemy's pieces wins. If neither is capable of taking all the pieces of the enemy, it is a draw.

If you happen to take the last piece of the machine, it will display **I MOVE**, then **YOU WIN** , to acknowledge your victory. No provisions have been taken for the case in which the machine takes all your pieces. You have lost, of course, but it will continue to request for your move. Be kind and acknowledge its victory by stopping the program.

REMARKS : -thinking time for each machine's move is almost fixed, it does not depend on the game being near its end.

-If you are performing a jump (single or multiple), all your moves (if multiple) must be jumps. This is, you cannot jump over a piece, then respond to the +TO? prompt with a move which is not a jump. And of course, you have to jump always with the same piece which started the multiple jump.

-if a multiple jump results in one of your pieces becoming a king (reaches row 7), it can continue jumping (now as a king), if legal. See examples.

EXAMPLES

left : typical beginning of a game

```
XEQ "CHKERS"
   0 1 2 3 4 5 6 7
 7 - O - O - O - O
 6 O - O - O - O -
 5 - O - O - O - O
 4 - - - - - - - -
 3 - - - - - - - -
 2 ▒ - ▒ - ▒ - ▒ -
 1 - ▒ - ▒ - ▒ - ▒
 0 ▒ - ▒ - ▒ - ▒ -
```

initial board

your pieces

```
HP 1ST?
N                    RUN
FROM?
              42     RUN
TO?
              33     RUN
I MOVE
FROM 15 TO 04
```

who moves first?

you move first?

from 42

to 33

41c moves

```
   0 1 2 3 4 5 6 7
 7 - O - O - O - O
 6 O - O - O - O -
 5 - - - O - O - O
 4 O - - - - - - -
 3 - - - ▒ - - - -
 2 ▒ - ▒ - - - ▒ -
 1 - ▒ - ▒ - ▒ - ▒
 0 ▒ - ▒ - ▒ - ▒ -
```

resulting board

```
FROM?
              02     RUN
```

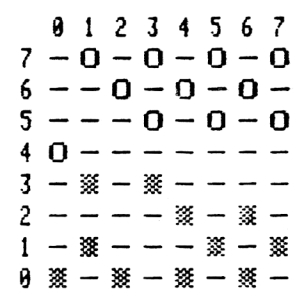
your move?

right: end of a game

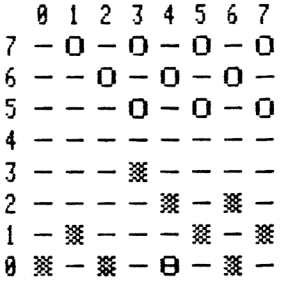
```
I MOVE
FROM 22 TO 11

   0 1 2 3 4 5 6 7
 7 - - - - - - - -
 6 - - - - - - - -
 5 - - - - - - - -
 4 - - - - - - - -
 3 - O - - - - - -
 2 ▒ - - - ▒ - - -
 1 - O - - - - - ▒
 0 ▒ - ▒ - ▒ - - -
```

multiple jump from 00 to 22 to 04, taking the pieces at 11 and 13, so winning

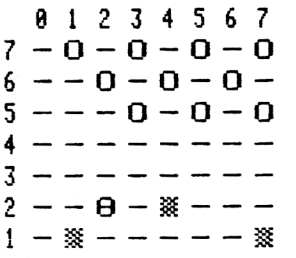
```
FROM?
              00     RUN
TO?
              22     RUN
+TO?
              04     RUN
+TO?
              -1     RUN
I MOVE
```



FROM?
40 RUN
TO?
31 RUN
I MOVE
FROM 04 TO 22
TO 40



FROM?
62 RUN
TO?
53 RUN
I MOVE
FROM 40 TO 62
TO 44
TO 22



FROM?

left

examples of multiple jumps

if you move from 40 to 31,

then the piece at 04 jumps to 22 (taking the piece at 13), and to 40 (taking the piece at 31 and becoming a king)

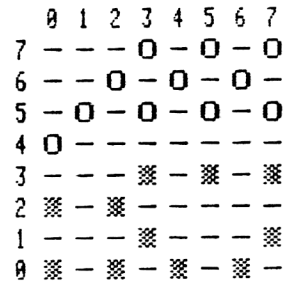
if you then move from 62 to 53

the king at 40 jumps to 62 (taking the piece at 51), to 44 (taking the piece at 53), to 22 (taking the piece at 33)

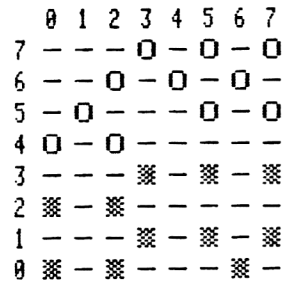
good jump!

right

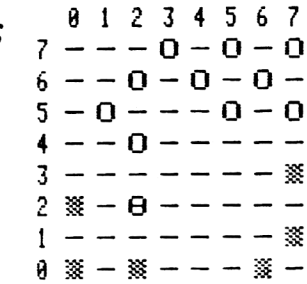
examples of jumps



FROM?
40 RUN
TO?
51 RUN
I MOVE
FROM 35 TO 24



FROM?
22 RUN
TO?
13 RUN
I MOVE
FROM 04 TO 22
TO 40
TO 62
TO 44
TO 22



if you move from 22 to 13, the machine makes a jump from 04 to 22, (taking the piece at 13) to 40 (taking the piece at 31 and becoming a king), to 62 (the piece continues its multiple jump, now being a king and taking the piece at 51), to 44 (taking the piece at 53), to 22 (the piece, being a king, jumps both backwards and forwards, so it takes the piece at 33). Incredible jump!

REMARKS :-no moves are random. The same game is played if you make always the same moves. The level of play is quite good for such a difficult game, as you'll see.

-do not make any change to the program. There are sequences of instructions that seem to be easily improved (for instance, GTO 27 ... LBL 27, XEQ 28, RTN , LBL 28, ..., RTN). Do not "improve" them or you'll find yourself with a program which makes illegal moves from time to time.

TEST GAME: if desired, test that your program is correctly loaded by executing the following game, HP first:

15-04/02-13, 06-15/11-02, 15-24/22-33, 04-22/31-13, 26-15/13-04, 17-26/62-53, 55-64/33-44, 46-55/42-33, 24-42/51-33,64-42/40-31, 35-53/02-13, 55-64/13-24, 64-73/24-06, 75-64/06-17,66-75/17-35, 73-62/04-15, 64-73/15-06, 75-64/06-17, 62-51/33-24,51-40/24-15, 40-22/20-11, 22-33/11-22, 33-11/00-22, 42-31/60-51,31-20/

35-44, 20-11/44-62, 11-33/17-26, 33-24/15-06, 37-15/06-17, 24-35
17-06, 35-26/06-24, 26-17/, etc, etc, etc.

STATUS : 502 + 56 lines, 760 + 102 bytes (2/3 RAMs), SIZE 084

registers: 00 - 19: scratch, 20-83: board

Note: if you are playing without a printer, and you fail to see the move of the machine: FROM ab TO cd, you can view it, once the FROM? is in the display, as follows:

VIEW 10 (a), VIEW 11 (b), VIEW 12 (c), VIEW 13 (d)

That's all, folks ! Hope you'll enjoy it !

VALENTIN ALBILLO (4747)

```
01♦LBL "CHKERS"
02 CF 12
03 FIX 0
04 CF 29
05 CLRG
06 1
07 STO 20
08 STO 36
09 STO 52
10 STO 68
11 STO 29
12 STO 45
13 STO 61
14 STO 77
15 STO 22
16 STO 38
17 STO 54
18 STO 70
19 CHS
20 STO 33
21 STO 49
22 STO 65
23 STO 81
24 STO 26
25 STO 42
26 STO 58
27 STO 74
28 STO 35
29 STO 51
30 STO 67
31 STO 83
32 -99
33 STO 09
34 FS? 55
35 XEQ "P"
36 CF 23
37 "HP 1ST?"
38 AON
39 PROMPT
40 AOFF
41 FS?C 23
42 GTO 46
43♦LBL 20
44 "I MOVE"
45 AVIEW
46 7 E-3
47 STO 00
48♦LBL 09
49 7 E-3
50 STO 01
51♦LBL 10
52 -1
53 RCL 00
54 RCL 01
55 XEQ 13
56 STO 15
57 X>Y?
58 GTO 00
59 X=Y?
60 XEQ 01
61 RCL 15
62 -2
63 X=Y?
64 XEQ 02
65♦LBL 00
66 ISG 01
67 GTO 10
68 ISG 00
69 GTO 09
70 GTO 38
71♦LBL 01
72 STO 02
73 XEQ 00
74 1
75 STO 02
76 RCL 03
77 XEQ 00
78 RTN
79♦LBL 02
80 -1
81 STO 02
82 XEQ 00
83 1
84 XEQ 00
85 1
86 STO 02
87 CHS
88 XEQ 00
89 1
90 XEQ 00
91 RTN
92♦LBL 00
93 STO 03
94 RCL 00
95 INT
96 RCL 02
97 +
98 STO 04
99 RCL 01
100 INT
101 RCL 03
102 +
103 STO 05
104 X<0?
105 RTN
106 7
107 X<Y?
108 RTN
109 RCL 04
110 X>Y?
111 RTN
112 X<0?
113 RTN
114 RCL 05
115 XEQ 13
116 X=0?
117 GTO 27
118 X<0?
119 RTN
120 RCL 02
121 ST+ 04
122 RCL 03
123 ST+ 05
124 RCL 05
125 X<0?
126 RTN
127 7
128 X<Y?
129 RTN
130 RCL 04
131 X>Y?
132 RTN
133 X<0?
134 RTN
135 RCL 05
136 XEQ 13
137 X=0?
138 XEQ 28
139 RTN
140♦LBL 27
141 XEQ 28
142 RTN
143♦LBL 28
144 RCL 05
145 X≠0?
146 GTO 00
147 RCL 00
148 RCL 01
149 XEQ 13
150 -1
151 X≠Y?
152 GTO 00
153 2
154 ST+ 06
155♦LBL 00
156 RCL 01
157 INT
158 RCL 05
159 -
160 ABS
161 2
162 X≠Y?
163 GTO 00
164 5
165 ST+ 06
166♦LBL 00
167 7
168 RCL 01
169 INT
170 X≠Y?
171 GTO 00
172 2
173 ST- 06
174♦LBL 00
175 RCL 01
176 INT
177 X=0?
178 ISG 06
179 X=0?
180 GTO 03
181 7
182 RCL 04
183 X=Y?
184 ISG 06
185♦LBL 03
186 -1
187 XEQ 04
188 1
189 XEQ 04
190 CLX
191 X<> 06
192 RCL 09
193 X=Y?
194 RTN
195 X>Y?
196 RTN
197 X<>Y
198 STO 09
199 RCL 00
200 INT
201 STO 10
202 RCL 01
203 INT
204 STO 11
205 RCL 04
206 STO 12
207 RCL 05
208 STO 13
209 RTN
210♦LBL 04
211 STO 08
212 7
213 RCL 04
214 RCL 08
215 +
216 X<0?
217 RTN
218 X>Y?
219 RTN
220 RCL 05
221 1
222 -
223 X<0?
224 RTN
225 XEQ 13
226 X<0?
227 GTO 05
228 RCL 04
229 RCL 08
230 -
231 STO 18
232 X<0?
233 RTN
234 7
235 X<Y?
236 RTN
237 RCL 05
238 1
239 +
240 STO 07
241 X>Y?
242 RTN
243 RCL IND 19
244 X<=0?
245 RTN
246 RCL 18
247 RCL 07
248 XEQ 13
249 X=0?
250 GTO 00
251 RCL 00
252 INT
253 RCL 18
254 X≠Y?
255 RTN
256 RCL 01
257 INT
258 RCL 07
259 X≠Y?
260 RTN
261♦LBL 00
262 2
263 ST- 06
264 RTN
265♦LBL 05
266 1
267 ST+ 06
268 RTN
269♦LBL 38
270 RCL 09
271 -99
272 "YOU WIN"
273 X=Y?
274 PROMPT
275 "FROM "
276 ARCL 10
277 ARCL 11
278♦LBL 39
279 STO 09
280 "├ TO "
281 ARCL 12
282 ARCL 13
283 AVIEW
284 BEEP
285 PSE
286 RCL 13
287 X=0?
288 GTO 07
289 RCL 10
290 RCL 11
291 XEQ 13
292♦LBL 40
293 STO 16
294 RCL 12
295 RCL 13
296 XEQ 12
297 0
298 RCL 10
299 RCL 11
300 XEQ 12
301 RCL 10
302 RCL 12
303 -
304 ABS
305 2
306 X≠Y?
307 GTO 45
308 0
309 RCL 10
310 RCL 12
311 STO 00
312 +
313 2
314 /
315 RCL 11
316 RCL 13
317 STO 01
318 +
319 2
320 /
321 XEQ 12
322 RCL 16
323 -1
324 X≠Y?
325 GTO 00
326 ST+ X
327 STO 02
328 XEQ 06
329 2
330 STO 02
331 RCL 03
332 XEQ 06
333♦LBL 00
334 RCL 16
335 -2
336 X≠Y?
337 GTO 00
338 STO 02
339 XEQ 06
340 2
341 XEQ 06
342 2
343 STO 02
344 CHS
345 XEQ 06
346 2
347 XEQ 06
348♦LBL 00
349 RCL 09
350 -99
351 X=Y?
352 GTO 45
353 CLA
354 GTO 39
355♦LBL 07
356 -2
357 GTO 40
358♦LBL 06
359 STO 03
360 RCL 00
361 INT
362 RCL 02
363 +
364 STO 04
365 RCL 01
366 INT
367 RCL 03
368 +
369 STO 05
370 X<0?
371 RTN
372 7
373 X<Y?
374 RTN
375 RCL 04
376 X>Y?
377 RTN
378 X<0?
379 RTN
380 RCL 05
381 XEQ 13
382 X≠0?
383 RTN
384 RCL 02
385 2
386 /
387 RCL 00
388 +
389 RCL 03
390 2
391 /
392 RCL 01
393 +
394 XEQ 13
395 X>0?
396 GTO 28
397 RTN
398♦LBL 45
399 FS? 55
400 XEQ "P"
401♦LBL 46
402 "FROM?"
403 PROMPT
404 10
405 /
406 INT
407 STO 14
408 LASTX
409 FRC
410 10
411 *
412 STO 15
413 "TO?"
414 PROMPT
415 10
416 /
417 INT
418 STO 00
419 STO 02
420 LASTX
421 FRC
422 10
423 *
424 STO 03
425 STO 01
426 RCL 14
427 RCL 15
428♦LBL 08
429 XEQ 13
430 RCL 02
431 RCL 03
432 XEQ 12
433 0
434 STO IND 19
435 RCL 14
436 RCL 02
437 -
438 ABS
439 2
440 X≠Y?
441 GTO 00
442 0
443 RCL 14
444 RCL 02
445 +
446 2
447 /
448 RCL 15
449 RCL 03
450 +
451 2
452 /
453 XEQ 12
454 "+TO?"
455 PROMPT
456 X<0?
457 GTO 00
458 10
459 /
460 INT
461 LASTX
462 FRC
463 10
464 *
465 X<> 03
466 STO 15
467 X<>Y
468 X<> 02
469 STO 14
470 X<>Y
471 GTO 08
472♦LBL 00
473 RCL 03
474 7
475 X≠Y?
476 GTO 20
477 2
478 RCL 02
479 RCL 03
480 XEQ 12
481 GTO 20
482♦LBL 13
483 X<>Y
484 8
485 *
486 +
487 20
488 +
489 STO 19
490 RDN
491 RCL IND T
492 RTN
493♦LBL 12
494 X<>Y
495 8
496 *
497 +
498 20
499 +
500 RDN
501 STO IND T
502 END
```

```
01♦LBL "P"
02 ADV
03 31
04 STO 03
05 35
06 STO 04
07 45
08 STO 02
09 79
10 STO 01
11 16
12 STO 00
13 48.055
14 CF 12
15 25
16 GTO 00
17♦LBL 01
18 11
19♦LBL 00
20 SKPCOL
21 X<>Y
22 ACCHR
23 ISG X
24 GTO 01
25 PRBUF
26 27.08308
27 STO 05
28 8
29♦LBL 02
30 1
31 -
32 STO 14
33 ACX
34 SF 12
35 2
36 SKPCOL
37♦LBL 10
38 2
39 SKPCOL
40 RCL IND 05
41 +
42 RCL IND X
43 ACCHR
44 ISG 05
45 GTO 10
46 PRBUF
47 65
48 ST- 05
49 CF 12
50 RCL 14
51 X≠0?
52 GTO 02
53 ADV
54 ADV
55 ADV
56 END
```

```
LBL'CHKERS
END          760 BYTES
LBL'P
END          102 BYTES

STATUS:
SIZE= 084
```

VAL - AN EXAMPLE OF THE USE OF PPC ROM AS A SUBROUTINE

The PPC Custom ROM must be a useful set of routines, to be called from a main program in RAM, so saving bytes and programming effort. If those routines are properly chosen, it is possible to write complex programs with little effort, using the capabilities of the routines in the ROM extensively. More important, it would be possible to write programs requiring the use of synthetics without knowing a single word about the internal operation of synthetics.

This can be a very good reason to buy the ROM: imagine a PPC member who is afraid of synthetics, or simply does not have the time to fully understand their behavior. He may think that the PPC ROM, especially the routines about synthetics, is of very little use to him. But he must realize that the routines in the ROM will allow him to write programs using just normal functions and including calls to routines in the ROM that perform the synthetic part of his programming. All he must know about the routines is what the inputs should be and what the outputs are, to begin using synthetics like a Wickes.

Here included is an example: the routine included herein, called VAL, performs exactly the opposite of ARCL (applied to numbers). If you have 3.24E-35 in X and you use ARCL X (in FIX 2), you'll have the number converted to ALPHA characters in the ALPHA register. This routine takes a number expressed in ALPHA characters in the ALPHA register and places its numeric value in X. For instance, if you have in ALPHA -3.2365E-48, after executing VAL, you'll have -3.236500000 E-48 in X.

```
01 LBL"VAL"      27 X() L
02 CF 00         28 R↑
03 CLST          29 10↑X
04 SF 25         30 ST* Y
05 LBL A         31 ST/ X
06 XEQ"C↑D"      32 ST+ L
07 ABS           33 X() L
08 RDN           34 RDN
09 XEQ IND L     35 +
10 FS? 25        36 RTN
11 GTO A         37 LBL 45
12 FS?C 00       38 CHS
13 *             39 RTN
14 RTN           40 LBL 46
15 LBL 48        41 RCL Z
16 LBL 49        42 ST- T
17 LBL 50        43 10↑X
18 LBL 51        44 /
19 LBL 52        45 RTN
20 LBL 53        46 LBL 69
21 LBL 54        47 10↑X
22 LBL 55        48 ENTER
23 LBL 56        49 CLX
24 LBL 57        50 SF 00
25 48            51 END
26 ST- L
```

This routine is written using just normal functions, no synthetics at all. The routine calls the ROM subroutine called C/D (see listings in V7N7P19, lines 164 thru 191). Flag 10 is assumed to be clear, so that the byte decoded is removed (you may insert a CF 10 after LBL "VAL"). This routine uses no register at all, just the stack.

Detailed explanation : lines 05 thru 11 are the loop. Line 06 calls the ROM routine, which places into X the decimal value of each byte in ALPHA, from right to left, removing the byte. Line 09 uses that value as an indirect address to call the appropriate routine directly, no tests required. If the value is 48 thru 57, the character was 0 thru 9. If it was 46, it was the decimal point. 69 is the exponent, 45 is the change of sign. In each case, XEQ IND call the proper routine.

All routines perform intrincate stack manipulations, to avoid using registers, and when finally XEQ IND fails to find its label (detected a character other than 0,...,9,E,.,-,) lines 12 to 14 show the result in X.

EXAMPLES : place -1.23456789E-41 (as written) in ALPHA

FIX 9, XEQ"VAL" → -1.2345678-41 (the 9 is present)

place in alpha: PEPE=-12,-.1,28E4

FIX 4, now, XEQ"VAL" → 280000.0000

XEQ"VAL" → -0.1000

XEQ"VAL" → -12.0000

Warnings = -A maximum of 15 characters in alpha will be considered as a single number. For instance, -1.23456789E-41 has 15 characters. If you try to find the value of -1.234567809E-41 (16 characters) the minus sign will not be taken into account. This is a limitation of C71D

-the routine stops considering characters as soon as it finds a character other than 0,...,9,E,-,. . This makes it possible to decode several numbers at a time in alpha, together with alphanumerics. For instance,

in alpha: COST=23,32,28

```
FIX 2, XEQ"VAL" → 28.00      (in alpha remains: COST=23,32)
       XEQ"VAL" → 32.00      (in alpha remains: COST=23    )
       XEQ"VAL" → 23.00      (in alpha remains: COST       )
```

so the first non-numeric character (-, E, . not included) is removed and execution stops.

-numbers are formatted: in alpha: 20E1, XEQ "VAL" → 200.00

-number cannot have separators, and . must be used to indicate decimals. This is, instead of 123.456.789,0 you should have 123456789.0 in order to find its value correctly. This is always the case if previously: CF 29, SF 28

-This routine is the equivalent of the VAL function in the HP-85 computer. It may be useful to separate alphanumeric information into alpha and numeric for easy storage, etc.

That's all. Hope it will be useful.

VALENTIN ALBILLO (4747)