Notes on the back story of this letter:

I wrote this *16-page* letter to ***John McGechie*** on Oct, 16 (1980, though it was actually sent on Oct, 22) to congratulate him for *Technical Notes V2*, commenting on the recently released **TRS-80 Pocket Computer** (a rebranding of the original **SHARP PC-1211**, the very first pocket computer), and giving him a number of scoops, including the imminent release of the **HP-41CV** and the ***Quad Memory Module***, among many other comments. Then I describe in detail my four articles included with the letter, for publication in a future *TN* issue.

Next, a long 2-page post-data follows. featuring additional comments on assorted things, including details of my ***Othello*** program's internals and caveats on attempting to improve its strategy without breaking anything. A long tirade on the still-unfinished *PPC ROM* goes next, including my comments on some bugs I found, my questions about using **`STO/RCL b`** and **`GTO`** global labels within the *ROM*, and extensive comments on PRIVATE, plus many more, including books and the final "rant" section (by the way, John told me in a subsequent letter that *Wickes* agreed with me on the authoship recognition part of the "rant").

Finally, I include the following materials, 4 articles, all of them for the **HP-41C**:

- ***Technical Subjects,*** a thorough 3-page discussion on *long synthetic labels* (say, **`LBL "HEWLETT-PACKARD"`**) and their behavior, including a number of experiments I conducted to gather evidence, then some keen sleuthing and conclusions. I also discuss three weird *synthetic assignments* and my experiments and subsequent sleuthing that resulted in some very worthwhile uses for them, including another new way to break **`PRIVATE`**,

- ***MR***, *NxN Matrix Operations*, a 5-page article including a program to perform individual or chained *NxN* matrix operations using a 2-level *RPN* stack, with *N* limited only by available memory. It implements operations for reading/writing matrices from/to magnetic cards, keyboard input, display output, +, -, matricial and scalar multiplications, inverse, plus stack operations, all of them callable from other programs, and further the user can easily add its own extensions. Full usage instructions plus 3 worked examples are included,

- ***3D-PLOT***, a 3-page article which features a program which can plot a user-defined function in such a way that the resulting plot seems almost 3D, which is good for fun and as a demo program to show off the capabilities of your new printer. It provides 5 parameters to fine-tune for the best results. Full instructions and six sample runs are included.

- ***Graphic Dice***, a very short and fast *synthetic* subroutine to accumulate into the printer buffer the graphic representation of the face of a die given as input the number of pips. Very useful for spicing up dice-based programs. Full description, usage instructions, and six sample runs.

Valentin Albillo, 05-03-2022

J.E. McGechie
Philosophy Department
Monash University
Clayton, Victoria
AUSTRALIA , 3168

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

Oct, 16

Dear John:

How are you ? Very busy ? Did you receive my letter dated Sept, 27 ? I hope the answer is yes. I am lacking from foreign correspondence these days. First, PPC CJ does not arrive (the last issue I have is the Jul/Aug one, have you received the Sept issue ?). Only Key Notes arrived, but its contents are very poor: just Wand introduction, NiCd batteries coverage, and that's almost all. Very few routines this time. Another thing I received was an update to the UPLE catalog , (User's Program Library Europe). I am a member of the European Library, so I have full access to over 300 programs documented in English, 60 in French, 25 in German, etc, both for the 41c and the 67. If this is of any use to you ... (By the way, I am also subscripted to Scientific American and BYTE).

Let's keep an order. First, congratulations for PPC TN V2. It was even better than V1 (and that one was very good, indeed !). There were several things that attracted my attention: first, the article about "New, Low price calculators /computers". I saw an ad about the TRS-80 Pocket Computer in - BYTE. It seems very good for its price. Here in Spain, it is - sold by 24000 pts (some 350 US $) which is very cheap (bare bones 41C is priced 35000). The only thing I couldn't find in the ad is its operating speed. Is it faster than the 41c? Much faster?. What about the "42c" and the new Texas machine ? Can you give me some information? Now, I told you before I'm introduced in local HP. So, I got some information, that seems to be true: (a) a new version of the 41c will go into market soon; it will be called HP-41cV, and basically is a 41c with full memory: all 4 RAMs are already included in the mainframe, leaving 4 ports free. It has also a better display. The quad-density RAM will be released at the same time. (b) an interface module is to be released that will allow to connect a 41c to a standard TV set. I thought it to be impossible, but my source insists it is true. The module will also add several new functions to the 41c, so to allow listings and low resolution plots to be carried out. (c) a new printer-plotter, which is supposed to be better than the present one. It will include much better plotting capabilities, larger buffer, faster. It is also more expensive.

I also liked the "New algorithm prime tester & fast factoriser". This last one was amazing: if the two factors of the number were close in size, it took very short times to find them. But if the two factors were, say, 3 and several millions, it took a very long time. This is exactly the opposite of a conventional factoriser. The programmable byte jumper is a very clever idea. I guess my 15-character labels can be used as 15-byte byte jumper, supressing the SF 25 and the - long execution time for that particular case. Maybe Geoff Smith would like to try it. I also liked your detailed explanation - about Word Processor. It gave me many ideas about similar programs & improvements to yours.

I also want to thank you for including so much of my materials. I hope it will be useful to the members of the glorious Melbourne Chapter. I see that Ernest Gibbs liked my matrix inverter. I want to thank him for taking care of adapt my program to run with a printer. I never do it myself, unless the program itself requires a printer. If he likes math type programs he'll be happy with my next submittals. Remember, math & games are by far my favourite topics, but not the only ones.

This one is to be relatively short: here included are several things:

- NxN matrix operations , as I stated in my last letter. This program should fill a real gap for those interested in bussiness requiring matrix handling. The program is relatively short (299 lines, 514 bytes), fast (I optimized the algorithms for speed) and complete. It includes the 11 most popular operations including: matrix read (from cards), matrix write (onto cards), matrix arithmetic (+,-,x), matrix inversion, multiplication by a constant, as well as supporting routines such as input, output, enter, exchange. All operations act upon general NxN square matrices, where N is limited only by available memory. The program simulates a RPN stack of 2 NxN matrices, so allowing both independent and chained operations. The program does not attempt to include an exhaustive collection of routines, but the user is supposed to provide its own routines as well (for instance, matrix transpose, determinant, and the like). The program is easily expandable. It is great to have two 10x10 matrices multiplied by simply pressing ×, don't you think? All operations are programmable, and can be called as subroutines. The program is not adapted to run with a printer, so if Ernie cares ...

- 3D-PLOT , is capable of plotting a user's defined function in such a way that it seems almost tridimensional. I am certainly sure that all members of Melbourne Chapter will like this one (this is, members with a printer!). The program is good for fun, and specially as a demo program of the capabilities of the 41c printer. The idea for writing the program was found in a book edited by Creative Computing. I thought it could be adapted for the 41c printer, and I wrote the program. There are 5 degrees of freedom (5 parameters) so that you can choose the best combination. I suggest you could include some plot (PEPE for instance) made using this program in the cover of a future issue of PPC TN, much like the cover of PPC TN V1 which featured a plot made using the HP-85 printer (plotter?). A mag card is included for you to try it.

- GRAPHIC DICE is a much improved version of my routine published in PPC TN V1 N2. The previous routine was 58 lines, 112 bytes long, and used no synthetics. This new version is just 30 lines, 84 bytes long, does exactly the same (same specifications) and runs much faster. It uses the techniques of create BLDSPEC characters via a text line. Synthetic text lines are used. The program is good for two reasons: first, it is useful by itself, as a subroutine for a main dice game: it is short and fast. Second, it is a very good example of the power of synthetics: no one can achieve such a short program without using synthetics. It should be pointed that anyone not using or understanding synthetics is missing a very good chance to drastically improve his/her programs. Note: a mag card is also included, as the texts are somewhat laborious to produce. Do not try to print the program: control characters in the text cause the printer to change state while printing the program, passing from lower to uppercase, single to double width, etc.

- Technical subjects is a discussion about some topics I have been studying a little. It includes some remarks about how GTO(alpha) finds its LBL(alpha), a method to create LBL(alpha) of 15 characters, some discoveries about "odd" assignments, featuring a new method to break PRIVATE using one of them. It's quite possible you knew it in advance. Hope not.

That's enough for this small letter. The multiple/polynomial regression program was written and functioned, but I discovered a new method, which saves many bytes. I'll submit it soon. The same applies to the book for the 34c. Tom Cadwallader sent a mysterious letter, including a Byte Counter program (very good, better than mine) without a single note or explanation. Do you know the reasons for he acting so? Is he angry with me?

Oct, 21 - long post-data I received, within a few days, Keynotes September PPC CJ, and your two letters. I noticed you received my letter dated 27th Sept, but posted 9th Oct. The reason for it being posted so late is that, once a letter is written, I wait a few days to see if I receive some journal, or another letter from you, or the like, so that I can include answers or comments. This is the case for the present letter: It was written a few days ago, but it will be posted tomorrow (Oct,22), and in the mean time I've received a large amount of foreign post, which I can comment on now.

Very happy you liked the small envelope; yes, TN 2 arrived and was much appreciated. I don't keep PPC TN for my exclusive use: I share it with local HP people which are very pleased with it, and usually made copies. You are by now a very popular folk among local HP people. Your clock program has been a best-copied program. I will enclose mag cards the problem being I have very, very few mag cards, most of them are full with programs. However, I'll try to get some cards , either free or paying for them, so that I can send you both - program descriptions & magnetic cards. Hope you'll not mind if the cards are not exactly brand-new: I am using old cards with 67/97 programs previously recorded on them. The old 67/97 title is covered with black ink, so as to not confuse you.

About Othello, yes, I knew it could be beaten quite easily. It is unavoidable if the strategy is strictly used (I am supposing you know perfectly the strategy the program uses by now. It is a quite simple positional strategy, which uses a list of pre-determined best positional moves, and simply choses the first legal move it finds in the list, going from top to bottom. The list of moves is arranged in order of descending priority). The strategy can be improved quite a lot using a new method I developed, but it would be very time-consuming, and I do not want to make the program slower, it would spoil the fun. The new strategy has been tried in a computer uisng a BASIC program written by myself, and found very effective. I'll send you the details, if you like. You mention that some kind of fix has been found to make OTHELLO play better: please, be most careful. The program has some very critical - parts, and it's quite easy to create bugs in it, specially if you try to alter it in some way. For instance, every time a move is selected and found legal (HP's or user's move), it is "marked" so that it will not be tried once again. This results in a faster speed near the end of the game, as most moves are known to be not legal (already occupied) and are not tried at all, so saving time. If your fix does not take this into account, it is quite possible to have a misbehaving program.

About the PPC Custom ROM: I read the pertinent pages of last PPC CJ (sept. By the way, this issue was unusually "empty". Most of its contents were either very little interesting, or already known to me), and I scanned - the proposed ROM routines: find included a copy of a comment sent to PPC. I found bugs in the routines !!! This is most intolerable ! We (PPC) are always saying that HP makes very poor software, full of bugs and inconvenience, but now, it is quite possible that our own software will finally have bugs. Besides, as most routines call each other as subroutines, any one bug can cause real trouble. Some questions: several routines make use of STO b, RCL b, to address some sections of code. Will this work in a ROM ? I have been told that the RAM addresses and the ROM ones are quite different. I have no - doubts about the routines performing fairly well in RAM, but I have serious doubts about if they will ever work in a ROM. I am inclined to believe that the pointer will be sent to RAM if STO b is used. Also, using GTO(alpha) extensively in the ROM is both slow and byte-consuming. GTO local should be used instead. See additional comments in the enclosed copy. By the way, if the ROM is going to include 67/97 compatibility routines, what happened to mine ? They remain unpublished, despite the fact that two or three P( )S are published in every PPC iss.

To stop with these comments, do you know the address of Keith Jarett ? I would like to be able to contact directly with him, to report all bugs & comments about the ROM I can find. Doing so via PPC seems fairly slow and uncertain.

About PRIVATE: I feel everybody's exagerating a lot ! the PRIVATE feature is useless as originally intended. I have never seen nor used a single PRIVATE program. Everybody records their programs without using the PRIVATE feature, even people who submit their programs to local HP for some commercial interests (publication. some kind of compensation such as getting a 41c free, etc). I am pretty sure that anyone interested in protecting his programs would rather protect his card holder from being stolen than his programs from being copied. Local HP people does not care at all about the PRIVATE feature being defeated. They know that were PPC to publish such methods, extremely very few people would understand them or know how to implement them , here in Spain. Do you really believe that normal people using the 41c know about bug 2, STO b, byte-jumper, etc ? They would rather pay for the program, instead of attempting to understand the method, get copies of KA, get a HEX table, etc. So, I'm sure that if PRIVATE is publicly defeated, almost no one is going to be affected by such event, <u>except</u> HP which will be forced to recognize that its PRIVATE implementation was very poor. This is commercially bad for the reputation of the firm. By the way, even the SECURE feature of the HP-85 can be defeated, one of my friends has the binary utility routines needed to remove SECURE from a program (all types of security) and this was shown to local HP people, who had to also recognize that the SECURE feature was easily broken. To resume, it seems that PPC (US) is exagerating too much. No one would care at all if PRIVATE is disclosed. By the way, I am neutral, it is the same to me if PRIVATE is publicly broken or not, I don't care at all. However, the ROM should <u>not</u>, <u>not</u>, <u>not</u>, <u>not</u>, be made PRIVATE !!! This is only useful to - frustrate membership, all are disadvantages. See copy.

About the books, I promise that, at least the 34c book, will be included with my next letter, in a tough envelope. The 41c book will probably take a while, to allow barcode to be included. By the way, many thanks for the Sharp brochure, the machine is sold in Spain for 24000 (350 US$),but I have not tested it yet. I am specially interested in its - operating speed, and wether GOSUB takes 1 byte or 5 bytes (one for each character). Got the idea ?

Very interested in microcode peeking (and poking, if possible !), much obligued for your long letter full of copies of interesting materials. Specially the long, quite detailed, brochure of the Sharp machine. Spanish brochures are less detailed. The article in BYTE is interesting, too.

As a final comment (I am never satisfied, you see !), don't you feel that a little is present in PPC headquarters ? Most group routines for the ROM have been "contributed" by . I guess that the group will have many routines by , group will include many 's routines, etc. I think it is not fair that will also "contribute" so many routines. I said "contribute", because many of their "contributions" are merely improvements of someone's program. For instance, it is clear that was developed by you . However, is acknowledged as the author of the routine for the ROM. I think it would be better if <u>all</u> people who, either created or improved the program, would be recognised as authors, and not just the last folk who added or deleted a feature. Do you agree ?

Best regards,

TECHNICAL SUBJECTS

Well, I've been dealing here and there with synthetics, and there are some things I want to tell you, though it's quite possible they are not unknown to you. Here included is a DEMO mag card, for you to test these matters. The demo card contains the following: (remember to read it in USER mode!!)

```
01 LBL"DEMO"
02 LBL"U"
03 LBL"DEABCDE"
04 LBL"VALENTI"
05 LBL"ALENTIN"
06 LBL"VALENTIN"
07 GTO"ALENTIN"
08 GTO"VALENTI"
09 GTO"V ALENTIN"
10 GTO"VALENTIN"
11 GTO""VALENTIN"
12 GTO"$$VALENTIN"
13 GTO"U    "
14 GTO"U UTS"
15 GTO"ABCDEABCDE"
16 LBL"ABCDEABCDE"
17 LBL"U "
18 GTO"U "
19 GTO"U"
20 LBL"HEWLETT-PACKARD"
21 -
22 Σ-
23 X)Y?
24 10^X
25 %
26 X)Y?
27 CHS
28 CHS
29 RCL 13
30 LN
31 -
32 /
33 MOD
34 -
35 SQRT
36 X(Y?
37 .END.
```

as you can see, this is a collection of synthetic & normal labels & GTO's. A little experimentation will show that:

line 07 (GTO"ALENTIN") goes to line05 (LBL"ALENTIN"), despite the fact that the label is shorter, and not to the line 06 (LBL"VALENTIN"), which is the right length

line 08 (GTO"VALENTI ") gives NONEXISTent, despite the fact that there is one LBL"VALENTI" and another LBL"VALENTIN".

line 09 (GTO"V ALENTIN") goes also to line 05, so it seems that GTO looks only at the 7 or less rightmost characters.

line 10 (GTO"VALENTIN" also goes to line 05, LBL"ALENTIN". The same is true for line 11 and line 12.

line 13 (GTO"U   ") goes to line 02 (LBL"U"), so it seems that all characters being null or following a null are ignored by the GTO.

line 14 (GTO"U UTS") goes also to line 02 (LBL"U"), so the UTS characters after the null are not tested.

All this makes me think that the micro code routine for the GTO does not take into account the text byte, but just looks at the 7 (or less) rightmost characters (if more than 7), and stops considering characters as soon as it finds a null character (looking from left to right. Now, to test this further:

line 15 GTO"ABCDEABCDE" goes to line 03 LBL"DEABCDE", and not to the line 16, which is its corresponding LBL"ABCDEABCDE".

line 18, GTO"U " goes to line 02 LBL "U" and not to line 17, which is LBL"U ". Of course, the LBL"U " is looked before the LBL"U", but it is not accepted. This is because the GTO"U " searchs for a U, the null character is not taken into account. The same is true for line 19, GTO"U", which ignores the LBL"U ".

line 20 LBL"HEWLETT-PACKARD" is a 15-character label. Now well, we have been told that the 3rd byte in an alpha label is a text byte which indicates how many of the following bytes are to be taken as a text. This byte is always 1 more than the number of characters, so that an FF byte produces a 14-char. label. So, 15-char. labels seem impossible. That's not so. Inserting a F0 byte produces a 15-char. label. Which is more, the text is not really in the label, SST shows the 15 characters as separate instructions, not counting the 1st. What is the first instruction, then ? The key assignment for the 15-ch label ! . For instance, line 20 is a

15-ch label, LBL"HEWLETT-PACKARD". If you read the magnetic card in USER mode, you'll notice that LBL"HEWLETT-PACKARD" is assigned to the LN key. You can test this by pressing (and holding!!!) the LN key in USER mode, you'll see LETT-PACKARD then NULL. Do not release the key , as this causes a crash. In PRGM mode, it just causes a XEQ"PACKARD" to be loaded into program memory. So, now it is easy to understand the characters in lines 21 thru 36. Line 21 is -, keycode 41, the assignmento to the 15 (LN) key. Tho rest are the H,E,W,...,R,D. Changing any of these instructions changes the text of the label. Changing the keycode for the assignment is not as easy. The assignment is changed, but it does not become active. A simple way is to, then, record the LBL in USER mode, and read back the LBL in USER mode. The assignment is activated. There is a simpler way, but it's so obvious that I'll let you to discover it. (Never forget to previously PACK)

Are this 15-ch. labels any useful ? Maybe. At least, they are a convenient way of personalizing software. In the case of the 15-ch label, it is so easy to change the text (simply change the instructions after the LBL) and the assignment, that they are readily changed to display whatever. Besides, they do appear in the catalog. Now you can have a LBL"(JOHN McGECHIE)" for your use, or LBL"MELBOURNE CHAPT", or whatever, or LBL"TOM CADWALLADER" which wasn't possible - before !.

I tested some other odd GTO's. For instance, GTO null (1D F1 00) is a crash when executed. The same is true for GTO"" (1D F0).

So much for the GTO's & LBL's. I also tested a litt le more the "odd" assignments(0,51),(0,55),and (0,60). I told you some experiences I had with them in a previous letter. Now, I pursued a more systematic way of testing its properties, and I found some interesting things (after several hundred crashes, of course!).

To do all my experiences I used a previously recor ded status card (only side 2), with 4 assignments on it: (0,55,24),(0,60,23),(0,51,-15),(145,124,-13). Of course, all the previous experiences continue to perform the same (for - instance, a single digit entry in program mode inmediately erased using back arrow, is restored in a different place),but I am not going to repeat myself herein: (KEY is the assignment)

0,51 in PRGM mode, if you previously do GTO.000, SST if desired , now, execute the function: KEY , display blinks
press any key (say, +) : + , two prompts appear
after whatever is in the display. If you fill the prompts with numbers (keyboard mapping is not allowed), STO 03 is loaded as a line in the program. If you delete the prompts using backarrow, the display wraps around.If you press now any key, you are set to RUN mode inmediatly. A very curious fact is - that if the prompts are attached to the 00 REG nn, and then filled, the STO 03 appears at line 32. We shall see that this is most useful.

Another thing: in PRGM mode, previously do GTO.000. Now, press X=Y? (line 01 X=Y?) , and now, KEY. You should see YES or NO (in PRGM!) depending on the outcome of the test. So, the X=Y? is executed even in PRGM mode. Testing with other functions, LN and 1/X always gave DATA ERROR , BEEP and all TONEs execute inmediatly. AVIEW & VIEW also execute, and you'll VIEW the selected register in PRGM mode. PI also executes, and leaves PI in X. Some, as SQRT, seem to do nothing. One other related trick: in PRGM mode, press GTO.000, now switch to RUN. pressing KEY, then + (or almost any other key) causes two prompts to be appended at the end of any number or alpha previously in the X register. Filling the prompts causes nothing if the mode wasn't FIX 9, and a crash otherwise. Deleting the prompts results in a wrap-around

display, etc. The crash event also makes the current program pseudo PRIVATE. This private condition is easily removed by simply pressing GTO.000.

(0,55) This one performs much like the previously presented (0, 51), and, in fact, they appear to do the same things, except that one is STO 03 and the other is STO 07.

I tried an interesting trick with this one. I started from MEMORY LOST, with 3 RAMs + card reader attached, and loaded the previously mentioned status card (just side 2). Then, I set PRGM mode (00 REG 44), GTO.000, set USER, KEY, + the display locked as 00 REG 44__ . I then filled the prompt using 55. The display turned out to be 31 STO 07. Now, out of USER (otherwise you can get into trouble), and BST . You should get:

30 STO 07	24 ""	now, you don't need to be
29 ""	23 RCL 09	specially smart to notice
28 STO 02	22 STO c	that this is nothing but a
27 STO 07	21 RCL 02	listing of the assignment
26 HMS+	20 STO 12	registers, loaded with the
25 STO 03	19 ""	4 assignments I mentioned before.

So, the (0,55) assignments allows an easy way of getting into the assignments registers. All you need to do is perform this sequence of operations into the first program in the catalog. As soon as you BST, you'll find into the assignments registers.

But this fact can be exploited even further. If there is another program before the one in which we are performing all these things, it is just logical to expect the pointer to situate in the END of the previous program when we BST. And this fact can be used to break a PRIVATE program ! So, I want to present another method of breaking PRIVATE, different from those already known (or suspected):

How to break a PRIVATE program : all you need is to have (0,55) assigned to a KEY for its execution in USER mode.

1) load the private program from a card. Do nothing but the instructions given herein. For instance, do not set PRGM mode to test that the program is PRIVATE indeed, or the like. It is essential that you never actually see PRIVATE in the display.
2) GTO ..
3) set PRGM, GTO.000 , set USER, KEY , press 99 to fill the prompt. You should see 31 STO 07 .
4) out of USER, BST , you are now at the END of the PRIVATE program.
5) SST and you'll be at the very beginning of the PRIVATE program you can see it at will using SST (never BST). You can record it on cards by simply passing them thru the card reader. The program in the cards is not PRIVATE.

Well, this method is different from my previous (using STO b, RCL b) which entered the program directly, or from the byte jumper method, which entered forwards. This one enters backwards !

There are still many, many tricks related to these "odd" assignments, but it is enough for this letter, this must be really boring for you, specially if you knew all of it in advance, as I suspect.

NxN MATRIX OPERATIONS Here introduced is a set of routines which allows the user to perform matrix operations, either individually or chained. It is not the purpose of this article to present an exhaustive set of routines, but just the basic operations, to - which the user may add its own developed ones.

The program simulates a RPN stack of 2 nxn matrices, where n is chosen by the user and is limited only by - available memory. The stack is composed of 2 registers, A, B, and each register holds a nxn square matrix. A is the "display", and B is the "upper" register. The following operations are available:

MR (matrix read) : reads A from magnetic cards: the contents of the card(s) are placed into A. B remains undisturbed. Minimum SIZE n^2+9

MW (matrix write): writes A onto magnetic cards. The matrix in A is recorded on mag cards. A & B, undisturbed. Minimum SIZE n^2+9

M+ (matrix add.) : Computes B+A. The result is placed in A, and B remains undisturbed. SIZE $2n^2+9$

B, A $\xrightarrow{M+}$ B, B+A

M- (mat.substract): Computes B-A. The result is placed in A, and B remains undisturbed. SIZE $2n^2+9$

B, A $\xrightarrow{M-}$ B, B-A

M✕ (mat. multiplic): Computes BxA. The result is placed in A, and B remains undisturbed. SIZE $2n^2+n+9$

B, A $\xrightarrow{M\times}$ B, BxA

M/ (mat. inversion): Computes the inverse of A. The result is - placed in A, B remains undisturbed. SIZE n^2+9

B, A $\xrightarrow{M/}$ B, A^{-1}

MI (mat. input): input A. Asks for the order n, and the elements of matrix A. They are placed in A, B remains undisturbed. SIZE n^2+9

B, - $\xrightarrow{MI}$ B, A

MO (mat.output): output A. Shows in the display all the elements of matrix A. Both A,B, remain undisturbed. SIZE n^2+9

B, A $\xrightarrow{MO}$ B, A

M↑ (mat. ENTER): duplicates the contents of A into B. A remains undisturbed. SIZE $2n^2+9$

B, A $\xrightarrow{M\uparrow}$ A, A

A()B (m.exch.): Exchanges the contents of A and B. SIZE $2n^2+9$

B, A $\xrightarrow{A()B}$ A, B

KM (const.mult): Given a constant k, computes k.A, and places the result in A. B remains undisturbed. SIZE n^2+9

B, A $\xrightarrow{KM}$ B, k.A

Characteristics: The program is 299 lines, 514 bytes. All operations may be used either individually or chained. The minimum SIZE for each is the required SIZE for that operation to be used individually. If some of them are to be used chained, the SIZE required is the maximum among the individual ones. All operations are programmable, they can be called as - subroutines of another program. See DEMO program. Here are some execution times:

M+ : N=1, 2 s. , N=3 , 4 s., N= 5 , 10 s., N= 7, 17 s. N=10, 38s
M- : the same as M+
M✕ : N=2, 7 s.,N=3 , 15s.,N=5 , 55 s., N=7 , 2m16s, N=9 , 4m37s.
M/ : N=3, 22 s.,N=5, 1m23s,N=7 , 3m35s, N=10 , 10m, N=13 , 21m.
KM : N=2, 2 s., N=4, 4 s. , N=6, 6 s., N=8 , 11 s, N=10 , 16 s.
M↑ : N=2, 2 s., N=4, 4 s. , N=6, 9 s., N=8 , 15 s, N=10 , 21 s.
A()B: same times as M↑

Warnings : - in chained operations, set always the maximum SIZE among those of the individual operations. In general for operations involving only A, SIZE n^2+9 is correct. For operations involving A & B, SIZE $2n^2+9$ will do, except for M×, which requires SIZE $2n^2+n+9$.

- the matrix inversion subroutine will give ERROR if all the elements in the main diagonal are found to be zero at the same time in any part of the inversion process. This is an infrequent case, but be aware that it may occur. This routine uses all flags from 0 to n-1. They are later reset to 0.

How to use : -simply, input matrices using MI, perform desired operation(s), then output results whenever desired using MO. Just be aware of the modifications caused to the stack A,B, by every operation. For operations involving A,B, both matrices must be of the same order n.

MI is used as follows: XEQ "MI" → N=? , enter the order, n
n R/S → A1,1=? , enter a_{11}
a_{11} R/S → A1,2=? , enter a_{12}
... ...
a_{nn} R/S → program stops

MO is used as follows: XEQ "MO" → A1,1=its value
R/S → A1,2=its value
... ...
R/S → An,n=its value
R/S → program stops

you do not need to output all elements of the matrix, you may - quit at any moment. All operations stop with some value in the display as soon as the operation is complete. To output the result, use MO. (MO always outputs A. To output B, A()B, then MO)

KM is used as follows: enter k: k, XEQ "KM" → k
the whole matrix in A has been multiplied by the constant k

MR is used as follows: be sure the order n is in R00. This can be done in two ways: n STO 00, or
XEQ "MI" → N=? , n R/S , then quit.

now , XEQ "MR" → CARD , prompts you to enter the card(s) once this is done, the contents of the cards have been loaded into A.

MW is used as follows: simply, XEQ "MW" → RDY kk of NN
pass cards until the operation is complete. Now, A is stored on the cards.

Very important: the stack A,B, does not raise when you enter a matrix via MI. The matrix overwrites previous contents of A, the previous A is not lifted to B. So, when chaining operations, remember to save previous result in B manually executing M↑ (enter) before entering a new matrix into A.

It is recommended to assign the functions to - keys. For instance, M+ to +, M- to -, M× to ×, M↑ to ENTER, etc.

EXAMPLES : Find the inverse of the given 4x4 matrix:

$$\begin{bmatrix} 2 & 2 & 3 & 2 \\ 2 & 2 & 3 & 1 \\ 11 & 5 & 4 & 6 \\ 2 & 1 & 1 & -9 \end{bmatrix}$$

-enter the matrix: XEQ "MI" → N=? , 4 R/S →
→ A1,1=? , 2 R/S → A1,2=? , 2 R/S → A1,3=? ,
3 R/S → A1,4=? , 2 R/S → A2,1=? , ...,
-9 R/S → 1.0040 , the matrix is entered in A.

-now, invert the matrix: XEQ "M/" → (after 42 sec) → 4.0030
-output the inverse: XEQ "MO" → A1,1=70.0000, R/S →
→ A1,2=-71.0000 , R/S → A1,3=-1.0000 , ... , R/S →
→ A4,4= 2.0000E-11, so the inverse is:

$$\begin{bmatrix} 70 & -71 & -1 & 7 \\ -252 & 255 & 4 & -25 \\ 121 & -122 & -2 & 12 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

another example: given A,P, find Q = PxAxP^{-1}

$$A=\begin{bmatrix}1&2&1\\3&2&5\\6&3&8\end{bmatrix},\quad P=\begin{bmatrix}5&2&3\\2&6&1\\5&3&3\end{bmatrix}$$

-to perfectly understand what happens to the 2-level stack, see how it changes after every operation:

the required sequence to compute Q is the following: input P, A()B, input A, M×, A()B , M/, M×, MO

stack	input P	A()B	input A	M×	A()B	M/	M×	MO
B	-	P	P	P	PA	PA	PA	PA
A	P	-	A	PA	P	P^{-1}	PAP^{-1}	PAP^{-1}

so, simply: input P: XEQ "MI" → N=? , 3 R/S → A1,1=?, 5 R/S →
→ A1,2=?, 2 R/S → A1,3=?, 3 R/S → A2,1=?,
2 R/S → A2,2=?, 6 R/S → A2,3=?, 1 R/S → A3,1=?,
5 R/S → A3,2=?, 3 R/S → A3,3=?, 3 R/S → value

now, XEQ"A()B", XEQ "MI" → N=? , 3 R/S → A1,1=?, 1 R/S → A1,2=?,
2 R/S → A1,3=?, 1 R/S → A2,1=?, 3 R/S → A2,2=?,
2 R/S → A2,3=?, 5 R/S → A3,1=?, 6 R/S → A3,2=?,
3 R/S → A3,3=?, 8 R/S → some value.

and now, XEQ "M×" , XEQ "A()B", XEQ "M/", XEQ "M×",

and output result: XEQ "MO" → A1,1=-524.0000 , R/S →
→ A1,2=-108.0000, R/S → A1,3=573.0000 , R/S →
→ A2,1=-589.0000, R/S → A2,2=-122.0000 , R/S →
→ A2,3=643.0000 , R/S → A3,1=-601.0000 , R/S →
→ A3,2=-124.0000, R/S → A3,3=657.0000 , R/S → value

so, the result is:

$$Q = PxAxP^{-1} = \begin{bmatrix}-524&-108&573\\-589&-122&643\\-601&-124&657\end{bmatrix}$$

third example: write a program to compute M^3+M^2+M for a given M

-the required sequence is: input M, M↑ , M×, M+,
, M×, M+, MO.

stack	input M	M↑	M×	M+	M×	M+	MO
B	-	M	M	M	M	M	M
A	M	M	M^2	M^2+M	M^3+M^2	M^3+M^2+M	M^3+M^2+M

so, load a program as this:

```
01 LBL"DEMO"
02 XEQ "MI"
03 XEQ "M↑"
04 XEQ "M×"
05 XEQ "M+"
06 XEQ "M×"
07 XEQ "M+"
08 XEQ "MO"
09 END
```

now, let's run the program for M = $\begin{pmatrix}4&3&1\\1&1&3\\6&5&9\end{pmatrix}$

so, XEQ "DEMO" → N=? , 3 R/S → A1,1=? , 4 R/S →
→ A1,2=?, 3 R/S → A1,3=? , 1 R/S → A2,1=?,
1 R/S → A2,2=?, 1 R/S → A2,3=?, 3 R/S →
→ A3,1=?, 6 R/S → A3,2=? , 5 R/S → A3,3=?,
9 R/S →
→ program resumes execution, then:

→ A1,1=281.0000, R/S → A1,2=228.0000,
R/S → A1,3=306.0000, R/S → A2,1=321.0000,
R/S → A2,2=263.0000, R/S → A2,3=393.0000,
R/S → A3,1=1101.0000, R/S → A3,2=900.0000,
R/S → A3,3=1316.0000,R/S → program stops

so, M^3+M^2+M has been computed = $\begin{bmatrix}281&228&306\\321&263&393\\1101&900&1316\end{bmatrix}$

VALENTIN ALBILLO (4747)

NxN MATRIX OPERATIONS

```
01 LBL"MR"
02 XEQ 05
03 RDTAX
04 RTN
05 LBL"MW"
06 XEQ 05
07 WDTAX
08 RTN
09 LBL"M+"
10 CF 29
11 GTO 01
12 LBL"M-"
13 SF 29
14 LBL 01
15 XEQ 05
16 LBL 00
17 RCL IND 01
18 RCL IND 02
19 FS? 29
20 CHS
21 +
22 STO IND 02
23 ISG 01
24 ENTER
25 ISG 02
26 GTO 00
27 RTN
28 LBL"A()B"
29 XEQ 05
30 LBL 11
31 X() IND 02
32 X() IND 01
33 X() IND 02
34 ISG 01
35 ENTER
36 ISG 02
37 GTO 11
38 RTN
39 LBL"KM"
40 XEQ 05
41 X()Y
42 LBL 03
43 ST× IND 02
44 ISG 02
45 GTO 03
46 RTN
47 LBL 05
48 RCL 00
49 X↑2
50 9
51 +
52 STO 01
53 DSE X
54 1 E3
55 /
56 9
57 +
58 STO 02
59 RTN
60 LBL"MI"
61 "N=?"
62 PROMPT
63 STO 00
64 CF 19
65 GTO 01
66 LBL"MO"
67 SF 19
68 LBL 01
69 CF 29
70 9
71 STO 04
72 RCL 00
73 1 E3
74 /
75 1
76 +
77 STO 05
78 STO 02
79 STO 03
80 LBL 10
81 FIX 0
82 "A"
83 ARCL 02
84 "⊢,"
85 ARCL 03
86 "⊢="
87 FIX 4
88 FS? 19
89 ARCL IND 04
90 FC? 19
91 "⊢?"
92 PROMPT
93 FC? 19
94 STO IND 04
95 ISG 04
96 X()Y
97 ISG 03
98 GTO 10
99 RCL 05
100 STO 03
101 ISG 02
102 GTO 10
103 RTN
104 LBL"M×"
105 RCL 00
106 1 E3
107 STO 01
108 /
109 STO 06
110 STO 05
111 9
112 STO 04
113 RCL 00
114 X↑2
115 +
116 STO 02
117 STO 08
118 LASTX
119 +
120 STO 07
121 RCL 08
122 RCL 00
123 ST- 02
124 +
125 1
126 ST+ 05
127 -
128 RCL 01
129 /
130 ST+ 08
131 RCL 02
132 RCL 06
133 10
134 ×
135 +
136 RCL 01
137 /
138 LBL 14
139 ST+ 04
140 RCL 04
141 STO 02
142 RCL 07
143 STO 01
144 XEQ 13
145 RCL 04
146 STO 03
147 CLX
148 STO 02
149 RCL 08
150 LBL 12
151 ST+ 02
152 RCL 07
153 STO 01
154 CLX
155 LBL 08
156 RCL IND 02
157 RCL IND 01
158 ×
159 +
160 ISG 01
161 ENTER
162 ISG 02
163 GTO 08
164 STO IND 03
165 RCL 06
166 ISG 03
167 GTO 12
168 1.001
169 ISG 05
170 GTO 14
171 RTN
172 LBL"M/"
173 RCL 00
174 1
175 -
176 1 E3
177 /
178 STO 05
179 STO 01
180 XEQ 09
181 LBL 91
182 FS? IND 01
183 GTO 90
184 RCL 01
185 RCL 01
186 RCL 00
187 ×
188 +
189 9
190 STO 06
191 STO 08
192 +
193 RCL IND X
194 X=0?
195 GTO 90
196 1/X
197 STO IND Y
198 STO 04
199 X()Y
200 RCL 01
201 ST+ 08
202 -
203 RCL 05
204 STO 03
205 STO 02
206 +
207 STO 07
208 LBL 04
209 RCL 02
210 RCL 01
211 X=Y?
212 GTO 06
213 RCL 03
214 X=Y?
215 GTO 01
216 RCL IND 07
217 RCL IND 08
218 ×
219 RCL 04
220 ×
221 ST- IND 06
222 LBL 01
223 1
224 ST+ 06
225 ST+ 07
226 ISG 03
227 GTO 04
228 RCL 00
229 ST- 07
230 LBL 02
231 ST+ 08
232 RCL 05
233 STO 03
234 ISG 02
235 GTO 04
236 GTO 01
237 LBL 06
238 RCL 00
239 ST+ 06
240 GTO 02
241 LBL 01
242 STO 06
243 RCL 00
244 ST× 06
245 RCL 01
246 ST+ 06
247 ×
248 +
249 9
250 ST+ 06
251 +
252 STO 02
```

```
253 LBL 07
254 RCL 01
255 RCL 03
256 X=Y?
257 GTO 07
258 RCL 04
259 ST* IND 06
260 CHS
261 ST* IND 02
262 LBL 07
263 RCL 00
264 ST+ 06
265 ISG 02
266 X()Y
267 ISG 03
268 GTO 07
269 SF IND 01
270 RCL 05
271 STO 01
272 LBL 16
273 FC? IND 01
274 GTO 91
275 ISG 01
276 GTO 16
277 RCL 05
278 LBL 09
279 CF IND X
280 ISG X
281 GTO 09
282 RTN
283 LBL 90
284 ISG 01
285 GTO 91
286 RCL 05
287 XEQ 09
288 "ERROR"
289 PROMPT
290 LBL"M/"
291 XEQ 05
292 LBL 13
293 RCL IND 02
294 STO IND 01
295 ISG 01
296 ENTER
297 ISG 02
298 GTO 13
299 END
```

```
+++++++++++++
+ 299 lines +
+ 514 bytes +
+++++++++++++
```

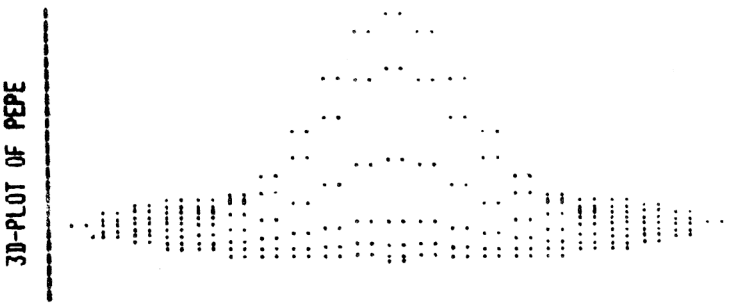
registers :

00	= n	
01	= scratch	
..	...	
08	= scratch	
09	= a_{11}	A
10	= a_{12}	A
..	...	A
n^2+8	= ann	A
n^2+9	= b_{11}	B
...	...	B
$2n^2+8$	= bnn	B
$2n^2+9$	= scratch	
...	... (*)	
$2n^2+n+8$	= scratch	

VALENTIN ALBILLO 4747

41C - 3D - PLOT OF FUNCTIONS

This little program (just 105 lines, some 190 bytes) allows you to plot the family of curves of a given function f(z). The resulting plot looks almost tridimensional, hence the title of the program. The program fits on a single magnetic card, and is good for demo purposes, or to show your friends the capabilities of your little 41c & printer.



This is a plot of a function defined as LBL "PEPE". The function itself was:

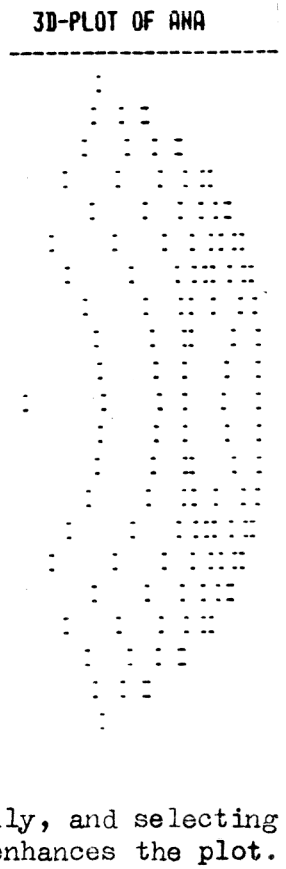
$$y = e^{-x^2/100}$$

The given function, y = f(z) is plotted as rising out of the X,Y plane, where both X and Y are inside a circle of radius 30. So, your f(z) is evaluated from z = -30 to z=30, so it must be defined in that interval. You should define your function anywhere in program memory, under a global label, and terminate it with an END. Your function must take its only argument from the X register, and return the correspondent y-value there.

Program characteristics .- This program is about 190 bytes - long, fits in a magnetic card, and requires SIZE 014. It prompts you for the name of your function, and several other parameters:

NAME? prompts you for the name of your function

ACCOL? prompts for the desired accol number of the column which will be used to plot each point of the function. If you want a single point, specify ACCOL=1. This gives a somewhat sparse appearance. Specifiying ACCOL=33 gives 2 points, "filling" more the curve. If you chose ACCOL=127, each point turns into a full column, increasing visibility. See examples.



INCX? prompts for the desired increment of the X-values. A large increment means the whole graph being printed in very few lines. A small increment represents a very elongated plot, and very long execution times.

YMAX? prompts for a multiplier constant.The value of y returned by your function is multiplied by this constant. A large YMAX gives better aspect to the plot. Do not exceed too much, or the buffer will print prematurely. A small YMAX gives a flat appearance.

INCY? prompts for the number of columns to be acumulated horizontally. The greater INCY, the less columns. A small INCY means more accurate plot, longer computing times, and, if too small, buffer printing prematurely.

AXIS? refers to the column position (relative to YMAX) in which the X axis of the plot is supposed to be. It performs a translation of the plot horizontally, and selecting a good value together with YMAX often enhances the plot.

See examples to appreciate the variations in the appearance of a given curve when the parameters are changed. A little experi-

mentation from your part will quickly give you the required experience to select the best parameters.

HOW TO USE : As always happens with any plot, some functions give better results than others. Select your favorite one and proceed to define it:

Example: to plot PEPE , $y = e^{-x^2/100}$

GTO .. , PRGM , LBL"PEPE", x^2, CHS, 1 E2, /, e^x , GTO.. PRGM

now, XEQ "3D-PLOT" → NAME? , PEPE , R/S → ACCOL?

we'll use 2 points per column, to give a more "concentrated appearance". 2 points per column is ACCOL=33

33 R/S → INCX?

the whole plot will be drawn in 21 lines, so 60/(21-1) equals 3:

3 R/S → YMAX?

our function is always positive, and between 0 and 1, so, we have 60 double-width columns to plot it:

60 R/S → INCY?

to get a maximum of 12 columns per horizontal row, we need 60/12 = 5

5 R/S → AXIS?

to obtain a good centering of the plot, the axis should be at the 22th column:

22 R/S → the plot is drawn. See graphics.

If you don't like the appearance, try new parameters, XEQ again 3D-PLOT, and enter the new parameters.

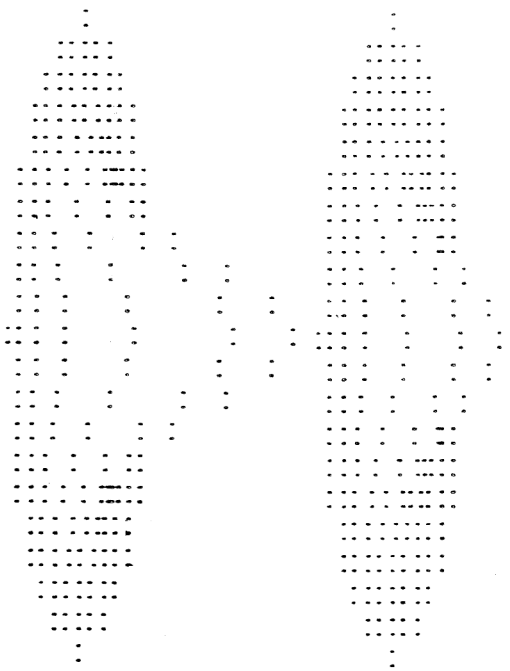
VALENTIN ALBILLO (4747)

```
01♦LBL "3D-PLOT"
02 CF 12
03 "NAME?"
04 AON
05 PROMPT
06 AOFF
07 ASTO 09
08 "ACCOL?"
09 PROMPT
10 STO 04
11 30
12 STO 05
13 CHS
14 STO 00
15 X↑2
16 STO 10
17 "INCX?"
18 PROMPT
19 STO 06
20 "YMAX?"
21 PROMPT
22 STO 11
23 "INCY?"
24 PROMPT
25 STO 13
26 CHS
27 STO 08
28 "AXIS?"
29 PROMPT
30 STO 12
31 ADV
32 "   3D-PLOT OF "
33 ARCL 09
34 PRA
35 "-------"
36 ACA
37 ACA
38 ACA
39 ACA
40 ADV
41 SF 12
42♦LBL 00
43 CLX
44 STO 01
45 STO 02
46 RCL 10
47 RCL 00
48 X↑2
49 -
50 SQRT
51 RCL 13
52 /
53 INT
54 RCL 13
55 *
56 STO 03
57 CHS
58 STO 07
59♦LBL 01
60 RCL 00
61 X↑2
62 RCL 03
63 X↑2
64 +
65 SQRT
66 XEQ IND 09
67 RCL 11
68 *
69 RCL 03
70 .7
71 *
72 -
73 RCL 12
74 +
75 INT
76 RCL 01
77 X<>Y
78 X<=Y?
79 GTO 02
80 STO 01
81 ENTER↑
82 X<> 02
83 -
84 1
85 -
86 SKPCOL
87 RCL 04
88 ACCOL
89♦LBL 02
90 RCL 08
91 ST+ 03
92 RCL 03
93 RCL 07
94 X<=Y?
95 GTO 01
96 PRBUF
97 RCL 06
98 ST+ 00
99 RCL 05
100 RCL 00
101 X<=Y?
102 GTO 00
103 CF 12
104 ADV
105 END
```

```
XEQ "3D-PLOT"
NAME?
PEPE                    RUN
ACCOL?
          33.0000       RUN
INCX?
           3.0000       RUN
YMAX?
          60.0000       RUN
INCY?
           5.0000       RUN
AXIS?
          22.0000       RUN
```

YMAX = 30

3D-PLOT OF PEPE

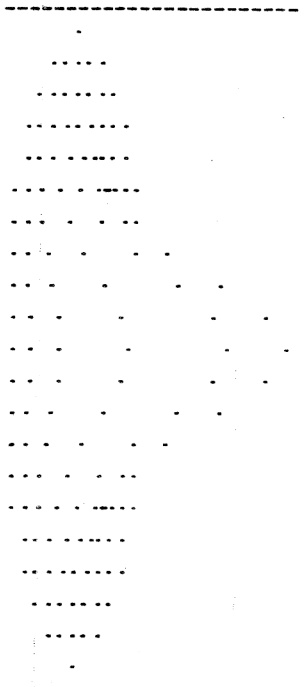


3D-PLOT OF PEPE

INCY=3
ACCOL=73
CF 12

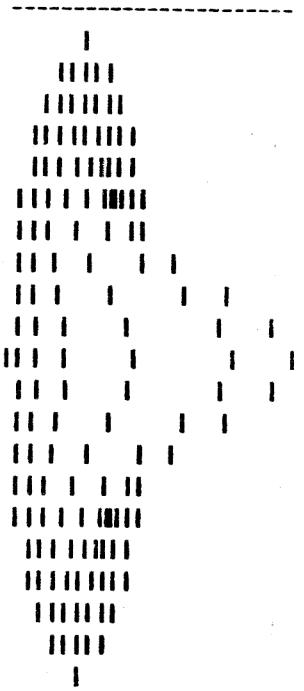
ACCOL = 1

3D-PLOT OF PEPE

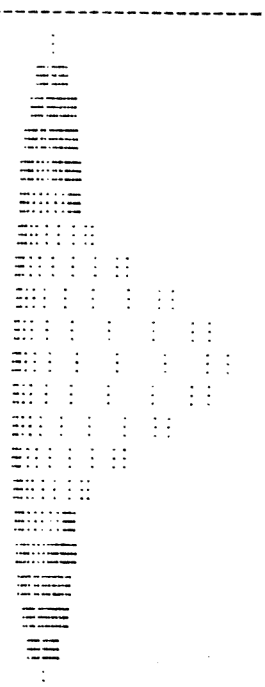


ACCOL = 127

3D-PLOT OF PEPE



3D-PLOT OF PEPE



41C - GRAPHIC DICE SUBROUTINE

This little subroutine will be very useful to those people who like computer games, specially dice type games (such as craps, etc), to add some graphics capabilities to their beloved programs.

This subroutine takes an input of 1,2,3,4, 5 or 6 in X, and accumulates in the printer buffer the graphic representation of the actual die. As it is intended to be used as a subroutine of a main program, the die is not printed by the subroutine. Press ADV or PRBUF to do so manually.

The subroutine is just 30 lines long, some 84 bytes, so it fits on a single side of a mag card. It is much shorter & faster than a previous routine (using only standard functions) which was 58 lines, 112 bytes long. This one uses the techniques pointed out in V7 N6 P27-28 to create the synthetic text lines. The required BLDSPEC string is written down previously, using the techniques described by Wickes in V7 N5 P56 (Understanding BLDSPEC), so that every 7 columns are accumulated into the printer buffer as a BLDSPEC character by the byte-saver procedure of creating a string representing the desired character, then RCL M, ACSPEC to accumulate the special character (first 7 columns of the dice representation).

```
                                        CAT 1
01◆LBL "◿"      LBL'◿
02 SF 12        END          84 BYTES
03 "Ω"          .END.        07 BYTES
04 XEQ IND X
05 RCL [                      0.25 STO 01
06 ACSPEC                     1.00
07 65                            XEQ "◿"
08 ACCOL        □
09 127                        2.00
10 ACCOL                         XEQ "◿"
11 CF 12        □
12 RTN                        3.00
13◆LBL 01                        XEQ "◿"
14 "⊢μö2▼"      □
15 RTN                        4.00
16◆LBL 02                        XEQ "◿"
17 "⊢μX0▼"      □
18 RTN                        5.00
19◆LBL 03                        XEQ "◿"
20 "⊢μX2▼"      □
21 RTN                        6.00
22◆LBL 04                        XEQ "◿"
23 "⊢◿X0▼"      □
24 RTN
25◆LBL 05       LINE 03 IS F2:11:FE
26 "⊢◿X2▼"      L14=F6:7F:0C:18:32:60:C1
27 RTN          L17=F6:7F:0C:58:30:60:D1
28◆LBL 06       L20=F6:7F:0C:58:32:60:D1
29 "⊢◿X5▼"      L23=F6:7F:0D:58:30:60:D5
30 END          L26=F6:7F:0D:58:32:60:D5
                L29=F6:7F:0D:58:35:60:D5
```

The remaining 2 columns are always the same so they are accumulated using ACCOL.

Once more, this is an example of the power and convenience of the synthetic functions: this routine is much faster & shorter than the equivalent one using just "normal" functions. All are advantages, no disadvantage at all. All those who are reluctant to use or take the time to understand synthetics are missing a good chance to drastically improve their pro - grams.

HOW TO USE :

call the subroutine , with 1,2,3,4,5,or 6 in X. After the control returns to your main program, the dice representation has been acumulated in the printer buffer.

VALENTIN ALBILLO (4747)