Notes on the back story of this letter:

This *26-page* (!) handwritten letter is *Melbourne PPC Chapter's* ***John McGechie***'s reply to my previous letter to him dated *1980-07-15*. John's handwriting takes some effort to decipher at times, but it's totally worth it.

He begins with some very detailed comments (several pages in fact !) about my attempt to create a **CP** *(Clear all Programs)* synthetics-based routine for the ***PPC ROM***, including a number of suggestions on how to successfully implement it using his ***b2*** routine, which he'd submitted to ***Richard Nelson*** but hadn't been published (doh!), so he published it in the *Melbourne Chapter Newsletter*. He then discusses in great detail the various types of **END** and **.END.**, and his attempts to create a *RAM* test to check their locally-created double-density *RAM* modules.

As I'd asked how to access microcode, he next discusses this and mentions ***Tom Cadwallader***'s method of identifying the addresses of the *ROM* routines by analyzing status register ***Q***, as well as ***Bill Wickes***' method to execute normal code in *ROM* starting at arbitrary addresses (and ending at the first **END** or **RTN** encountered), which John appraises as *"very ingenious"*, explaining in detail why (and indeed it is !). On how to *"modify microcode"* (see my previous letter), he offers some comments but the less said about my naive question, the better.

He then addresses the *"NN catalogues"* (*Flag 30 Catalogs)* and his own exploring using his ***KA*** *(Key Assignment)* program, commenting on the weird behavior of assigned *numbers* (inserting the reverse of register ***Q***'s contents as a text line in program memory, anyone ?) and **eG0BEEP.**

Next are his extensive comments about some aspects of the *PPC Club* and *Mr. Nelson*'s handling of it and his judgment on what to publish and what not, as well as some comments on the *PPC ROM*'s contents, inserting a *"shameless plug"* (XD!) for his own >1,000-byte word processing program for the *HP-41C*, and ending the main part of his letter at just 12 pages.

The letter doesn't end here, however, but continues (on *6th August !*) with two pages including additional detailed suggestions on how to make my **CP** program work, and more. He also included magnetic cards for his programs ***KA(#14)***, ***b2***, ***ARA***, ***AN***, ... everything but the girl !. Even the *P.S.* is long and meaty !

Surely the letter ends now ? Surely not ! These are just the first 14 pages, followed now by another *12* extra pages detailing his many experiments trying out the techniques and suggestions he discussed in the main part, and featuring extensively annotated thermal-paper printouts of everything but I won't discuss them here, this is just a *Notes*, examine them yourselves and enjoy.

John McGechie was an outstanding person by all accounts, including technically, philosophically and humanly. He was always enthusiastic, never had a bad word about anything, and would help anyone as best he could, even a 20+ newbie from 20,000 Km. away like me.

May he rest in peace.

Valentin Albillo, 21-12-2021

Department of McGeehie Philosophy

Monash University, Clayton, Victoria, 3168

AIRMAIL

IN FEE MATTER

POSTAGE

CLAYTON -7.8.80 VIC 3168

AUSTRALIA POSTAGE PAID

1.85

P.B. 14706D

To V. Albillo

℅ Padre Rubio 61-2º C

~~6[illegible]~~ Madrid 29

SPAIN

Cards in a few days - separately.

Philosophy Dept.
Monash University
CLAYTON VICTORIA AUSTRALIA 3168
TELEPHONE: 03 541 0811 TELEGRAMS: Monashuni Melbourne
TELEX: MONASH 32691
DEPARTMENT OF PHILOSOPHY

5th August '80

Dear Valentin,

More detail on your long letter to follow up the short note sent yesterday. Since some of the matters you raise are quite serious for PPC I'll give them pretty careful attention — others are serious another way — for 41c ! (Does you have the latest KA programs - #13+14?)

(1) The Custom ROM and your routine. 'CP'

One way to clear all pgms without packing is to delete the ENDs, GTO .000 and XEQ, D, E, L, EEX, 999 This clears memory without moving .END. Useful when using B2 (a my "little b2" — do you have that? RCL b — into X and then XEQ b2. The contents of Y are stored in the register where RCL b was effected. Much better than Bill Wickes version — about 1 second, and dont need to know address at all.) (about 70 bytes — good one for ROM I think !!) (Published in our local Newsletter last March — but Richard hasn't published ..... !)

According to Charles Close there are 3 kinds of END's and correspondingly 3 kinds of .END. Byte 3 is:

	Packed by PACK	Packed by GTO..	not packed - distorted by PGM entry
END	09	20	0D
.END.	29	20 not sure	2D

Have you tried all 6 ??? Charles unclear over this in his article (some data in enclosed print out)

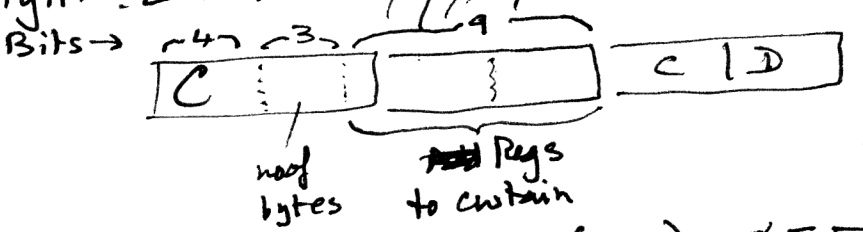
He is not quite clear in writing on this and I've not yet tried his suggestions — but programming it is a good scheme (Why do so few PPC members try this? Apparently I was the only one to try programming key assignations. Why?) (I have over 20 different programs from the original bug 2 version of January 2nd this year) (written up in those 16 pages — I'll send a copy if you wish)

I've not yet analysed your program (I will next week when Univ. classes end for 3 weeks & will have some spare time) but you say you put the new .END. at the "top of program memory". Where? It should not be in ROD but below it.

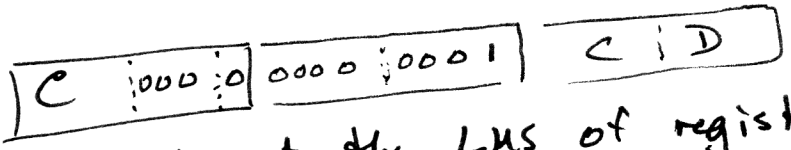
See print out

That "LBL 10, LBL 14" sounds familiar. Several times when I've disturbed an END or alpha label (global) I've lost program and found a 'LBL 14' (or is it?) which will not erase. Thats probably a remnant of the inserted .END. Where is "CP" put in memory? At the top? Have you tried altering it so that the .END. is put in just below the END of "CP"? Remember that there should be a hex digit set in the .END. to link it to the certain address. Preserve the 'chain' intact and it may work — ie make your entered .END. include the link to the certain address. (I'll try to work out what the format should be and see how to modify your program to make the

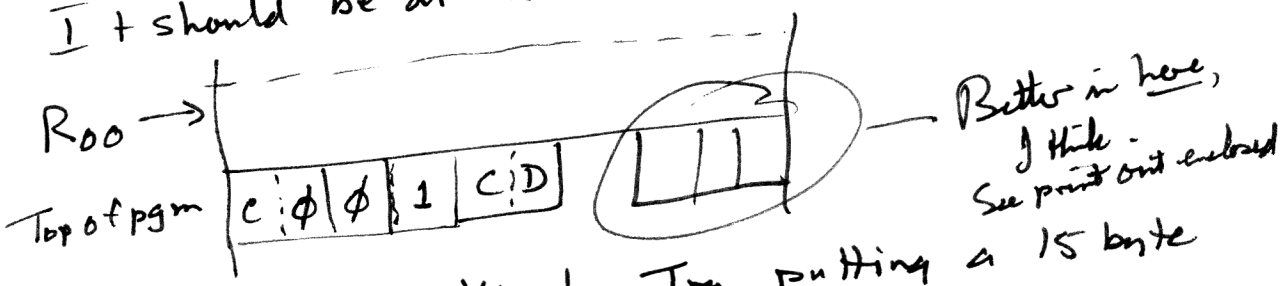
right .END. C0/00/2D should contain, perhaps



So if contain is at (say) 0FFF (size 000 on Basic 41c) and the .END. is dumped exactly one reg below, it should then be



It should be at the LHS of register — perhaps



I've not checked this! Try putting a 15 byte jump instruction (eg CLD) at line 001, then the .END. next to it, byte jump, and analyse alpha (you know that the bytes jumped are read out into alpha?) I'll enclose my new DECODE for you to use. Do the byte jump, read in my "AN" then RCL M, XEQ "AN". Will this help? Let you know.

See print out.

Then knowing the structure needed, you can redesign your "CP" to make the right NNN.

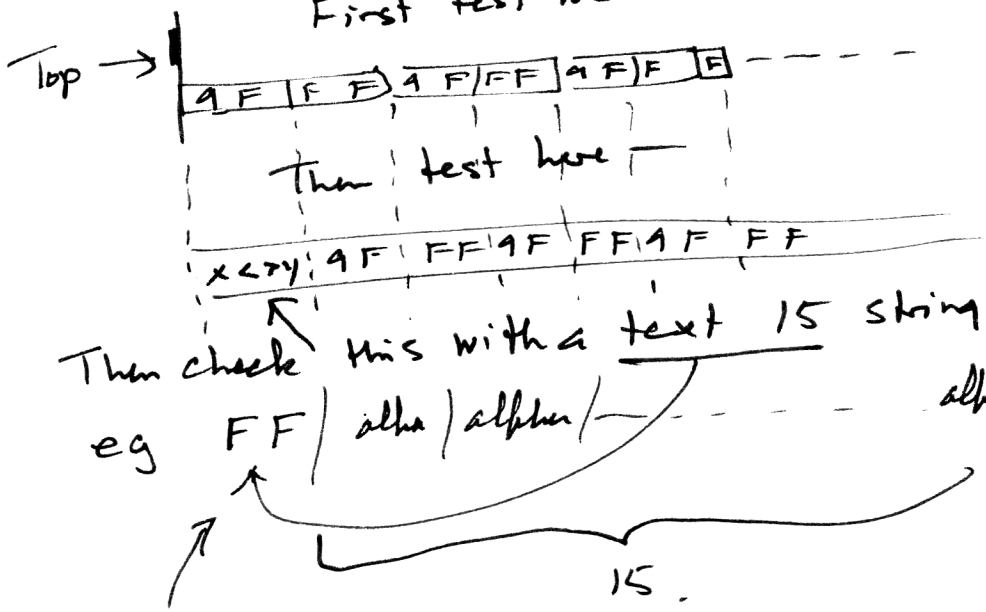
It may be that your last X<>c should be a STO c ? The STO IND Z (in the artificial Reg c contents) may derange the following X<>Y, X<>c, RTN ?

Lots to try here!

RAM tests.

As you'll have read by now, I had to devise RAM tests to check the locally made DD modules (I have one – will have two very soon)

First one was exhaustive, but can't program it quite. Tone 1ND e is 9F/FF. Size to 000 (top of DD module on trial) and Fill it with Tone 1ND e. Print out. Then put an X⇐>Y at top of menus, PACK, print out.



That checks the first byte.

So all bits checked in program mode that way.

Next – an NNN (I forget which I chose – had this as an alpha string: /F7/FF/FF/FF/FF/FF/FF/FF/F I think read ~~that~~ into Reg M.

RCL M produces the NNN

FF/FF/FF/FF/FF/FF/FF

in X. STO + RCL this in every reg and

on a SIZE including as the lowest numbered data reg. one known to be good (I used a SIZE of 001 on basic 41c, did a STO, RCL in R00 to get normalised NNN — an alpha string which is, I think, 10/FF/FF/FF/FF/FF/FF

need a counter in the stack.

This is X=Y? on every register's normalised version of the FF/FF/FF/FF/FF/FF/FF

Prints as "rrrrrr" from Print reg.

If one does not produce the same, alarm bells ring, whistles blow, etc.

Now another two to store recall negative normal numbers to check sign digit. That leaves two bits in the sign digit untested.

and MSD of mantissa

in this STO/RCL mode — but they are not used for STO/RCL, anyway, of data.

I have these on cards and Graeme Dennes has combined the STO/RCL routines into a single program. Takes about 1 minute to run I think. (? or 8 minutes?)

I can't think of any more tests — the mantissa in one STO RCL is 7 (bit pattern 0111) and the other is 9 (bit pattern 1001) to test all bits. Only bits 2 and 3 of sign digit not tested — but NNN normalise probably checks them.

Any ideas?

Accessing microcode

I doubt whether this can be done. Tom Cadwallader had identified the addresses of the ROM routines via an analysis of Register Q (I'll enclose his details) and it may be possible do do something with this.

No - send later if you wish - this letter too long already!

Bill Wickes ROM access accesses normal code in ROMs (I have a routine ARA which was designed to do this, but didn't hit on Bill's ~~wan~~ round the problem — ARA — jump to any RAM address and can be initialised to go directly to any address in RAM, XEQ from location addressed to a RTN or END. In use, RCL b, STO M, (at start location)

ASTO nn — in a reg. For (say) 10 addresses, do a RCL b, STO M, ASTO 00, ---- ASTO 01 -- ASTO 09. (On enclosed card "ARI" with "ARA")

Then 1, XEQ ARA will XEQ from 1st RCL b location to RTN and come to line after 1, XEQ ARA, etc. 2, XEQ ARA does the second etc. (The Reg b address is recovered by CLA, ARCL IND X, RCL M) Wouldn't jump to a RCL b'ed ROM address! I sent details to Bill, and 4 weeks later back came his ROM access! I've not worked on ARA since!

His method very ingenious – the jump into ROM is effected via an artificial return stack with the the ROM address one up in the stack. I had tried just a STO b of the ROM address.

Read Bills article very carefully. The form of ROM addresses in Reg b is different. I'll add details of same decoding I did. on the printer ROM routines (normal code) and the WAND routine. I don't quite understand their addressing system yet. There's so much to be done!

RTN format the same as RCL b format unlike RAM RTN addresses in RTN stack. See Bill W's Game Cover article in PPCJ.

Modifying ROMs? Sounds a bit dangerous!

The ROMs in the 41c are not magnetic but in microchips in Selftwin Silicon. Hardwired! If you could push bits into them and change – you may destroy them! They are not the EPROMs of computer type.

No! this later also!

But an analysis of the access code of the 41c via the ports may be feasible?

Remember the work on the 67 in late 1978? See also the HP article on microcode in the HP Digest no 3. (Can you get hold of it? Can send a PC if not.)

The HP 85 may use EPROMs (its a normal computer, though compact)

I think the NN catalogues of the 41c are the normal catalogues with the read out alpha deranged. Their patten follows the normal catalogues. I copied out many of those catalogues last year and found some that went through the same cycle. Where one starts in some of the catalogues depends on the settings of flags — I forget which now. (in the 30's from memory) The resemblances to Synthetic prompts is accidental, I suspect.

I have an enormous sheet of paper with 256 x 256 boxes on it sampling the possible 2 byte assigned functions. Try IND on some with prompt lines.

Even two byte assignment can be made by KA — but there are 256 x 256 of them !! I've sampled the lower numbered ones. Assigned numbers (not alpha' numbers') are useful — they put unterminated numbers plus the reverse of Reg Q in an up to 7 byte text line into program. I will be writing this up very soon — if you wish I'll send a quick summary. (attached in printout!)

See printout!

I doubt (there I go again!) whether there is a STO -- which will produce the status functions - you would need to fill the prompts with hex digits - and then don't enter from the number keys! If they did, high numbered XROM's could be made. The highest are given by eGOBEEP -- with IND 99 The IND (shift key) makes them jump up by 128. (Details if you wish - but put eGOBEEP on a key - experiment! Other XROM prefixes produce fewer XROM nos. - or often only one.

even in a disguised form!

Can use KA to make any individual one on a key, though.

Some take all/her

PPE comments

I think you are a bit unfair about Save -

Some in the Journal is of little value to many. But some is not. Many think the NNN stuff should not be there -- about 80% is of no use to me, another 80% is of no use to another number, and so on.

What is valuable? Remember Dave Kemper's discovery (one you made independently) of the Subroutine return stack on the 67? See what that led to!!

Richard Nelson doesn't understand most that he receives — some quite awful stuff I know. I sent material on formal logic programming two years ago — an utterly new area (about 8 people I now know of have independently done very interesting work in the area) and offered it to Richard. No response. It's since been published here. John Kennedy read a copy I sent to Richard, and wrote very excitedly to me about it — he thought he was the only one too — and we have corresponded ever since.

Much is just for fun! (Programmable calcs. are fun!) Some is mathematically interesting, some are useless (who wants 100 decimal printouts of π, or alphanetics or .....) What will be fruitful? We can't judge always.

And Richard does work his tail off for PTC. He is at it 7 days a week — comes home from work — eats his tea — has a short sleep, then at the Club house from 9pm to 3am every night. He does it all!!

That so much good stuff is published is a miracle I think. Having had grizzles here in Melbourne over the contents of our Newsletter (all of which I've written, for most months) from those who've done nothing toward it, I've some idea what he must do. He should have editorial help, but doesn't very often.

All one can do, I guess, is send in material nearly finished in copy ready form, brief as possible, clear as possible, etc. I think his judgement could often be better — but my own has its lapses - - - -

As to the ROM contents - - - what do PPC membership want of the ROM? Programs in it useful to many. But the range of conflicting interests! Look at the Users Library programs. Which of those 7000 would you want in a ROM? Why? Send them an argued, documented 'wish list' of what you think desirable. I think alpha sort routines, as flexible as possible, are probably the most valuable — and, for example, extend RAM (Jon d'Arcas stuff) access routines for long free 41c's as very desirable.

Sub routines to assemble into your pgms are best - but not with those damned long alpha labels that HP use!

And what of routines that would allow the printer to Graphics do pictorial work? A multiple function plot of sequential computation of functions — why not? Animated displays via plots on the printer? And the recording on micro cassette of your program library accessed under program control? ..... (Thats where the latest wand work points)

I dont think the PPC ROM will suit every user. I suspect that 3/4 of its programs won't be wanted by me — but then again...

Must close for now. Many more things in your valuable packet of material I'll cover later — but may I use some of this in our Technical Notes? I'll assume you've no objection to my reporting it there?

Yours John

(3324)

I have a word processor program for the 41c — about 1000 bytes at least is needed. That would be a wish I would have for a ROM! Will be written up soon.

6th August!

Here is the check up. I think your routine will work if

(1) You run it

(2) Get above inserted .END.

(3) Insert an instruction to unhack ~~it~~ inserted .END. – or at least to jog/jar/distort it (!!!) and then pack. You may need to have its chain to top of memory correct, though it looks as though this is not necessary. C0/00/2D, an 'unhacked' .END. may be the one to store.

C0/01/29 didn't work although this led to lost memory – but haven't tried an inserted instruction above it followed by a PACK.

C0/01/20 with 7 pluses above it went PRIVATE and I think this is an END, not an .END. (Charles Close vague about this) but PGMs cleared – had to master clear afterwards.

STwing 00/00/00/00/C0/01/20 and inserting a byte (a +) above it, cleared pgms ok, ~~but left~~ twice! Turned into a broken .END.

C0/01/CD/00/00/00/00 at top of memory cleared pgms ~~that~~ when a + inserted above it and then a PACK but left junk in pgm – a RCL 13 that caused trouble.

SO!

I think your answer is this:

Make your NNN to put ~~the~~ immediately below R00 in the form

$$\phi\phi/\phi\phi/\phi\phi/\phi\phi/c\phi/\phi 1/2\phi$$

Bytes 1 2 3 4 5 6 7

where bytes 5 to 7 are those of an ~~e~~backed END (packed by GTO.. according to Charles Close) and putting an instruction above it, then testing clears pgm, puts in only that instruction before the .END. that results.

I have still not analysed your program — spent some time trying + showing you all this. I think "AN" is a good routine — with byte finder to read out pgm bytes like very quick — only about 1 second.

I hope this all helps! Took a lot of time to do and to annotate — so just see you appreciate it!!

I'll send cards of KA #14, b2, ARA, AN etc for you to try. (You proud of AN — ~~take~~ you should take credit for making me think of it!)

John.

Remember the END and .END. formats! See 1st 2 Corvallis Columns in PPCJ. Have to check up on Charles Close's data too. But later on!!! You nice to have another 'foreign correspondent' for our local Chapter!

First I read these pgms into the tele

```
CAT 1
LBL'KA
LBL'KC      This is the #14 (printer) version
LBL'NN
END          287 BYTES
LBL'b2      Synthetic B2
END          93 BYTES
LBL'AN      Natural(tele) language NNN decode
END          75 BYTES
LBL'ARA     Auto RAMaccess
LBL'IRA     Initialisation for ARA
END          53 BYTES
LBL'ROM     Bill Wickes ROMaccess
END          57 BYTES
.END.        09 BYTES
```

First making some key assignments needed at the start.

```
XEQ "KA"
PRS?
    5.    RUN
KEY1.
  241. ENTER↑
   62. ENTER↑   an E+
   11.    RUN
KEY2.
  144. ENTER↑
  124. ENTER↑
   12.    RUN
KEY3.
  145. ENTER↑
  117. ENTER↑
   13.    RUN
KEY4.
  144. ENTER↑
  117. ENTER↑
   14.    RUN
KEY5.
    4. ENTER↑
   21. ENTER↑
   15.    RUN
KEY6.
  145. ENTER↑
  121. ENTER↑
  -13.    RUN
KEY7.
PRKEYS
```

Byte jumper (one instruction)

Recall b or 1/x

STO M or

RCL M

S ass. to LN

STO Q

```
USER KEYS:
 11 XROM 04,62
 12 XROM 01,60
 13 XROM 05,53
-13 XROM 05,57
 14 XROM 01,53
 15 X
 25 "AN"
 01♦LBL "TST"
 02 Σ0
 03 RCL b
 04 STO [
 05 RCL [
 06 X
```

Byte Jumper
RCL b
STO M
RCL Q
RCL M
S ass to key.
NNN analyser.

alpha print after RDN of b2

```
PRP "TST"
01♦LBL "TST"
 02 "♦"          Text line - one null
 03 RCL b
 04 STO [
 05 RCL [
 06 5  }
 07 "" }  put in by the S ass to key — see later.
 08 STO 14 ← and this!
 09 .END.
09 STOP  — ooops! not really intended
02 STO _    STO Q
02 +
03 +
04 +
05 +
06 +
07 +
08 +
09 +
10 +
11 +
12 +
13 +
14 +
15 +
16 +
```

Some filled bytes as a storage location for later.

```
PRP "TST"
01♦LBL "TST"
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
 08 +
 09 +
 10 +
 11 +
 12 +
 13 +
 14 +
 15 +
 16 +
 17 .END.
   XEQ "NN"
 33.    RUN
111.    RUN
108.    RUN
108.    RUN
105.    RUN
 98.    RUN
 65.    RUN
   GTO "TST"
   GTO .010
     RCL b
   XEQ "b2"
Y=NNN, X=Rb?
    RUN
  "p=idÃã"   ***
"iQ0@
   GTO "TST"
   PRP "TST"
```

OK? So this is how it now is!

↑ this instruction.

Address of

Abillo!

Prompt line to warn that correct format is in stack.

Print after RUN

```
01♦LBL "TST"
 02 +
 03 +
 04 RCL 01
 05 OCT
 06 HMS
 07 HMS
 08 FRC
 09 FACT
 10 -
 11 +
 12 +
 13 +
 14 +
 15 +
 16 +
 17 .END.
    GTO .001
02 X
    PRP "TST"

01♦LBL "TST"
 02 5
 03 "x"
 04 +
 05 +
 06 RCL 01
 07 OCT
 08 HMS
 09 HMS
 10 FRC
 11 FACT
 12 -
 13 +
 14 +
 15 +
 16 +
 17 +
 18 +
 19 .END.
       RDN
 "A"   ***
       RDN
  ""   ***
       RDN
-1.757272696-59   ***
     STO _
   GTO "TST"
02 X
    PRP "TST"

01♦LBL "TST"
 02 5
 03 "TST"
 04 5
 05 "x"
 06 +
 07 +
 08 RCL 01
 09 OCT
 10 HMS
 11 HMS
 12 FRC
 13 FACT
 14 -
 15 +
```

Artificial B2 storage — OK?

this is the actual byte where Reg b was recalled

02 X this is the pgm entry from of 5 assigned to a key.

Note indentation → Some of the bytes here case in lower ½ of table.

hex is the result! ← where from? Aha!

Still there!

What's in stack from "b2"

that's the b2 stored NNN still there

store NNN in Reg Q (wait!)

NB

02 X → Put 5 in Pgm from ass. key

put in key 5 on key.

```
 16 +
 17 +
 18 +
 19 +
 20 +
 21 .END.
    GTO .001
      STO _
02 X
    PRP "TST"

01♦LBL "TST"
 02 5
 03 "Abillo!"
 04 5
 05 "TST"
 06 5
   07 "x"
 08 +
 09 +
 10 RCL 01
 11 OCT
 12 HMS
 13 HMS
 14 FRC
 15 FACT
 16 -
 17 +
 18 +
 19 +
 20 +
 21 +
 22 +
 23 .END.
    GTO .001
      STO _
02 X
03 X
04 X
05 X
    PRP ""

01♦LBL "TST"
 02 5555
 03 ""
 04 ""
 05 ""
 06 "Abillo!"
 07 5
 08 "Abillo!"
 09 5
 10 "TST"
 11 5
   12 "x"
 13 +
 14 +
 15 RCL 01
 16 OCT
 17 HMS
 18 HMS
 19 FRC
 20 FACT
 21 -
 22 +
 23 +
 24 +
 25 +
 26 +
 27 +
 28 .END.
```

Store NNN in Q again.

5 on ass. key.

Wow!!

02 X, 03 X, 04 X, 05 X } Put in 4 5's by ass. key.

Reg Q is emptied by first 5 (ie ↑)

Get the idea? Assign E to a key and the E's chain up — So do decimal points. E on a key followed by a number key puts E 3 (EEX, 3) in pgm. See KA below.

```
                GTO "KA"
                GTO .000
01 XEQ 05
01 LASTX
                LIST 010

01 LASTX
02♦LBL "KA"
03 "PRS?"
04 PROMPT
05  E
06 -
07  E3
08 /
09 STO 02

10♦LBL "KC"
   GTO .002
      PACK
PACKING
      Σ0
@♦s♦KA
      RCL [
     XEQ "AN"
0.120015300   ***
00<000?3004;41
0.000000000   ***
      RDN
 "41"   ***
      RDN
"?3004;"  ***
      RDN
"00<000"  ***
      RDN
   GTO "KA"
   GTO .001
    DEL 001
   GTO .000

01 LASTX
02 END
     CAT 1
    PRP ""
01 LASTX
02 END
   GTO .002
      Σ0
@♦◿♦♦♦
      RCL [
0.<0000=000   ***
     XEQ "AN"
0.<0000=000   ***
00<0000=000000
     CAT 1
   GTO .000
ALPHA DATA
01 +
02 +
03 +
04 +
05 +
06 +
07 +

   PACK
PACKING
    PRP ""
```

```
01 +
02 +
03 +
04 +
05 +
06 +
07 +
08 LASTX
09 END
   GTO .009
      Σ0
@♦oF♦s
      RCL [
 1.200010260+?3   ***
     XEQ "AN"
 1.200010260+?3   ***
00<00009<600?3
     CAT 1
   GTO .000

01 X<>Y
      PRP ""
01 X<>Y
02 +
03 +
04 +
05 +
06 +
07 +
08 +
09 LASTX
10 END
   GTO .010
      Σ0
@♦◿F♦s
      RCL [
 1.200014260+?3   ***
     XEQ "AN"
 1.200014260+?3   ***
00<0000=<600?3
     CAT 1
   GTO .001
     CLP ""
PACKING
01 +
02 +
03 +
04 +
05 +
06 +
07 +

   PACK
PACKING
   GTO .001
   LIST 010

01 +
02 +
03 +
04 +
05 +
06 +
07 +

08♦LBL "KA"
09 "PRS?"
10 PROMPT
```

```
                XEQ "NN"
                GTO "KA"
                GTO .000
                   RCL b
                XEQ "AN"
  0.000000000-44    ***
0000000000016<
                GTO "KA"
                GTO .001
                   RCL b
                XEQ "AN"
  0.000000000-44    ***
0000000000016<
                GTO "KA"
                GTO .002
                   RCL b
                XEQ "AN"
  0.000000006-45    ***
0000000000616;
                GTO "KA"
                GTO .003
                   RCL b
                XEQ "AN"
  0.000000005-45    ***
0000000000516;
                XEQ "KC"

KEY7.
Recall C        144. ENTER↑
Shifted LOG     125. ENTER↑
                -14.    RUN
KEY8.
Recall L        144. ENTER↑
Shifted LN      122. ENTER↑
                -15.    RUN
KEY9.
                   RCL c
          "p×iäAQ"    ***
                XEQ "AN"
          "p×iäAQ"    ***
17700169l6<111
                GTO "KA"
                GTO .000
                   RCL b
                XEQ "AN"
  0.000000000-44    ***
0000000000016<
                   PACK
PACKING
                GTO "KA"
                GTO .000
                   RCL b
                XEQ "AN"
  0.000000000-44    ***
0000000000016<
                   STO [
                   RCL c
          "p×iäAQ"    ***
                XEQ "AN"
          "p×iäAQ"    ***
17700169l6<111
                ΣREG 00
                   RCL c
                XEQ "AN"
          "@×iäAQ"    ***
16<0016916<111
                GTO "TST"
                GTO .028
                   RCL b
                XEQ "AN"
  0.000000002-88    ***
00000000002112
                GTO "TST"
                GTO .027
 28 LASTX
                GTO .029
                   PACK
PACKING
                   RCL c
                XEQ "AN"
          "@×iäAQ"    ***
16<0016916<111
                GTO "TST"
                GTO .029
                   Σ0
L
                GTO "TST"
                GTO .028
 29 CLD
                   PACK
PACKING
                GTO .030
                   Σ0
LA)********
                ASTO 10
                   CLA
                ARCL 10
LA)***
                   RCL [
                XEQ "AN"
     0.<<0829000    ***
00<<0829000000
                GTO "TST"
                   RCL b
                XEQ "AN"
  0.000000002-80    ***
0000000000211:
                GTO "TST"
                GTO .031
                   RCL b
                XEQ "AN"
  0.000000000-88    ***
00000000000112
                   CAT 1
                   CAT 1

LBL'KA
LBL'KC
LBL'NN
END          294 BYTES
LBL'b2
END           93 BYTES
LBL'AN
END           75 BYTES
LBL'ARA
LBL'IRA
END           53 BYTES
LBL'ROM
END           57 BYTES
LBL'TST
.END.         65 BYTES
                PPP "KA"
```

mistake!

top of file
address of Reg ΦΦ

Byte Φ, register Hex 16C
$= 16^2 + 6\times16 + C$
$= 256 + 96 + 12$
$= 364$ decimal.
(SIZE = 20)
2 modules in

same address!

byte 6 of reg. 16A = address of "+" in line 1

address of line 2, bytes 5

"KA continue"

Byte before (a few before) .END. on byte 2 of reg (hex) 112 though .END. in reg (hex) 111

to jump 6 bytes

.END. reg the same shift.

Byte jump 6 bytes first were nulls — then 4 or acc

actually a CC — see below

To jump 15 bytes

11 bytes — so first 4 were nulls

remove some of RH nulls from alpha — put in RegM

LA > in RegM

X reg = RegM
ΦΦ | CC | 08 | 29 | 00
.END. packed form

address of END before LBL 'TST'
Byte 2 of hex 11A

Byte before final .END.
Byte Φ of 112
9 regs below END before LBL TST

CCOOP — not in TRACE mode

16D = R00

Shifted LOG in user mode
X register = Reg C

analysis

Σ reg on hex 177 (Reg 10)

'Cold start' const.

address of .END.

X register = Reg C

Unchanged!

Σ Reg now starts with R00 = 16C (hex)

X register = Reg C

Notice sign digit of 1 makes recall an alpha string.

Next page for listing for you.

```
+ + + + + + +

 08*LBL "KA"
"PRS?"  PROMPT   E  -
 E3  /  STO 02

 16*LBL "KC"
          17 FIX 0

          18*LBL 06
             PRP "KA"

          01 +
          02 +
          03 +
          04 +
          05 +
          06 +
                PACK

PACKING
             PRP "KA"

          01*LBL "KA"
          02 "PRS?"
          03 PROMPT
          04  E
          05 -
          06  E3
          07 /
          08 STO 02

          09*LBL "KC"
          10 FIX 0

          11*LBL 06
           12 ""
          13 ASTO 00
          14 .5
          15 XEQ 07
          16  E
          17 XEQ 07
          18 RCL 02
          19 RCL c
          20 RCL [
          21 "×i▲ " F5|ϕ1|69|ϕc|00|BF
          22 ASTO c
          23 STO IND Z
          24 RDN
          25 STO c
          26 ISG 02
          27 GTO 06
          28 BEEP
          29 "WSTS"
          30 PROMPT
          31 WSTS
          32 RTN

          33*LBL 07
          34 RCL 02
          35 INT
          36 +
          37 ENTER↑
          38 +
          39 "KEY"
          40 ARCL X
          41 TONE 9
          42 PROMPT
          43 STO 01

          44 RDN
          45 STO 03
          46 RDN
          47 XEQ 00
          48 -24
          49 X<> 03
          50 XEQ 00
          51 10
          52 ST/ 01
          53 RCL 01
          54 ABS
          55 INT
          56 LASTX
          57 FRC
          58 ST* Z
          59 .1
          60 -
          61 X=0?
          62 GTO 05
          63 RDN
          64 4
          65 X=Y?
          66 ISG Z

          67*LBL 05
          68 RDN
          69 STO Z
          70 X<>Y
          71 8
          72 X<>Y
          73 ST* Y
          74 RDN
          75 +
          76 CHS
          77 44
          78 +
          79 ST+ 03
          80 SF 25
          81 SF IND X
          82 DSE Z
          83 ""
          84 X<> 01
          85 X<0?
          86 GTO 01
          87 RCL '
          88 XEQ 03
          89 STO '
          90 GTO 08

          91*LBL 01
          92 8
          93 ST+ Z
          94 RDN
          95 RCL e
          96 XEQ 03
          97 STO e

          98*LBL 08
          99 RDN
         100 RDN
         101 X<>Y
         102 16
         103 *
         104 +
```

Woops! in Trace mode!

I cleared these by backarrow and packed.

This is KA #14 — a printer version — compact and very fast — leaves 14 regs on basic 41c for assignments.

Connection for row 4 here →

You know how this part of KA works do you on? Combining flag no. and byte no. for key chosen from its key code — in stack at same time! Very tough stack analysis to work this out — took about 30 or so hours of working out last February.

← F1/Fϕ Prefix for ass. reg contents

(corrected) Flag no in R03 and X →

} check for flag no over 29.

Fϕ (Text ϕ)

Closes up — no point print as space in text line

x i ▲ ↓

F5|ϕ1|69|ϕc|00|BF

1 prints here as space

When this is ASTO'd in c, c contains 1ϕ|ϕϕ|ϕ1|69|ϕc|ϕϕ|BF

I notice you use BF also — cϕ is ok also — still stores 1st ass. in ϕcϕ. Using ASTOc saves lines and bytes!!

Puts ϕϕ|ϕϕ|ϕ1|69|ϕc|BF into Reg M

ie an alpha string of 4 bytes.

set flag ind for ↑ or ↑ (unshifted keys)

Shifted keys here

set flag ind for e

Byte number for key code in ass. reg now in X.

When Y is a normal number (any) and X = 1 to 127, BLDSPEC puts byte 1 to 127 in X and lifts stack – Y & Z Lost, T = Z and T And Z = t.
When 't' ie byte 127 in Y and Ø is in X, BLDSPEC produces Byte 129 in X (as alpha char)
't' in Y, 1 in X → Byte 129
etc upto 't' in Y, 127 in X produces byte 255 in X as RH. Byte. This BLD uses this printer ROM function! Very proud of this one – short and fast.

When X=0, empty alpha string in X see test at line 128

```
105♦LBL 00
106 128
107 X<>Y
108 X<Y?
109 GTO 10
110 X<>Y
111 -
112 ENTER↑
113 ENTER↑
114 127
115 BLDSPEC
116 X<>Y

117♦LBL 10
118 BLDSPEC
119 CLA
120 X<> [
121 ASHF
122 ASTO X
123 CLA
124 ASTO Y
125 ARCL 00
126 ARCL X
127 ASTO 00
128 X≠Y?
129 RTN
130 "⊢♦"
131 ASTO 00
132 RTN
```

Very fast, short, BLD using BLDSPEC!

This ensures that only right most byte in X is isolated – when t in Y. (Produces 2 bytes in X for byte no over 127.) only right most wanted.

```
133♦LBL 03
134 FS?C 25
135 GTO 09
136 CLA
137 X<> \
138 "⊢***"
139 X<> \
140 X<> d
141 SF IND 03
142 X<> d
143 X<> \
144 "⊢****"
145 X<> ]
146 RTN

147♦LBL 09
148 X<> d
149 SF IND 01
150 X<> d
151 RTN

152♦LBL "NN"
153 CLA
154 ASTO 00

155♦LBL 02
156 STOP
157 XEQ 00
158 RCL [
159 GTO 02
160 END
```

Synth. B3 needed? If Flag 25 clear it is.

Synthesised Bug 3 to set flags over 29 (LH column keys). Flag no was entered in R03

Synthetic B3 not needed

Universal NNN generator – not part of KA – but uses LBL 05 (lines 105 to 132)

This is the ~~entered~~ automatic B2 for non-B2 machines. Bill Wickes method automated.

```
PRP "b2"

01♦LBL "b2"
02 TONE 0
03 TONE 9
04 "Y=NNN, X=Rb?"
05 PROMPT
06 CLA
07 X<> [
08 "⊢♦♦♦♦"
09 X<> [
10 X<> d
11 CF 00
12 CF 01
13 CF 02
14 CF 03
15 SF 07
16 SF 08
17 X<> d
18 FRC
19 CLA
20 X<> [
21 "⊢♦"
22 CLX
23 STO \
24 "⊢♦♦"
25 X<> \
26 X<> [
27 ASTO X
28 "×i"
29 ARCL X
30 "⊢"
31 RDN
32 RCL c
33 X<>Y
34 ASTO c
35 STO 00
36 RDN
37 STO c
38 RTN
39 END
```

necessary warning!

Enter pgm with ~~#~~ contents to be stored in pgm in Y, address via RCLb in X and dumps Y at that address.

Probably not needed

Clear Byte number from RCLb digits.

for FRC to shift digits from RCLb one digit left.

UUW

3 right most digits from RCLb isolated – shifted one digit left (FRC), made into uu/wØ in X

F2/01/69

Append BD (I think!) now have 01/69/uu/wØ/BD in X.

```
    PRP "AN"

01♦LBL "AN"
02 FIX 9
03 PRX
04 XEQ 01
05 XEQ 01
06 XEQ 01
07 "****"
08 ARCL Y
09 ASTO Y
10 "**"
11 ARCL Y
12 ASHF
13 ASTO Y
14 CLA
15 ARCL T
16 ARCL Z
17 ARCL Y
18 PRA
19 STOP

20♦LBL 01
21 CLA
22 X<> [
23 "⊢♦♦♦"
24 X<> [
25 "⊢♦♦"
26 RCL \
27 "**"
28 ARCL X
29 ASHF
30 ASTO X
31 X<>Y
32 END
```

This is the analysis routine, in 4/c hex digits of any no in X. Much faster + shorter than Bill Wickes version 'Decode' version — suggested by you decode into octal digits — easier to read. Must be FIX 9 — Hex digits always show in display if exponent < 10 (including 0A etc) So break up 14 digits in alpha into blocks, ARCL X and isolate blocks of 6 at a time — as alpha in stack and reassemble after last one is cut to 1st two characters (6 + 6 + (6-4) = 14)

← isolate 6 digits from X — the left most 6.

```
  XEQ "ARA"
```

Woops!

```
ALPHA DATA
    PRP "ARA"

01♦LBL "ARA"
02 STO L
03 RDN
04 CLA
05 RCL b
06 X<> [
07 "⊢♦♦♦"
08 STO \
09 "⊢♦♦"
10 X<> \
11 STO [
12 ARCL IND L
13 RDN
14 ASTO b
15 RTN

16♦LBL "IRA"
17 CLA
18 STO [
19 ASTO IND Y
20 AVIEW
21 RTN
22 END
```

Use, eg, 5, XEQ ARA in Pgm, and goes to RAM address put in Reg 05 by IRA.

Enter with ind. address of reg. with RCL b address in it.

Initialise ~~RCL~~ b ~~at wanted entry location~~, put reg no in X, RCL b at wanted location, XEQ IRA. To run from that location, key in Reg. number, XEQ ARA.

Bill Wickes ~~ROM~~. In Journal.

Does what my ARA failed to do.

```
    PRP "ROM"

01♦LBL "ROM"
02 SF 01
03 STO [
04 CLX
05 RCL b
06 X<> [
07 FC?C 01
08 RTN
09 "ABCDE"
10 X<> [
11 STO a
12 X<> \
13 X<> [
14 ARCL 10
15 X<> [
16 STO \
17 CLX
18 X<> a
19 X<> [
20 "⊢AB"
21 STO [
22 X<> \
23 STO b
24 END
```

END and .END. analyses.

```
          GTO "TST"
           GTO .002
              RCL b
             XEQ "AN"
  0.000000002-80    ***
0000000000211:
          GTO "TST"
 30 +
              PACK
PACKING
          PRP "TST"

       01*LBL "TST"
        02 5555
        03 ""
        04 ""
        05 ""
       06 "Abillo!"
        07 5
       08 "Abillo!"
        09 5
        10 "TST"
        11 5
          12 "x"
        13 +
        14 +
        15 RCL 01
        16 OCT
        17 HMS
        18 HMS
        19 FRC
        20 FACT
        21 -
        22 +
        23 +
        24 +
        25 +
        26 +
        27 +
        28 LASTX
        29 CLD
        30 +
        31 .END.
          GTO .030
               Σ0
@***LA)********
          CLX
        GTO "TST"
         GTO .029
          DEL 001
 28 CLX
              PACK
PACKING
          PRP ""

       01*LBL "TST"
        02 5555
        03 ""
        04 ""
        05 ""
       06 "Abillo!"
        07 5
       08 "Abillo!"
        09 5
        10 "TST"
```

Byte 2 of reg. (hex) 11A

Extra byte occupied in TST

Sorry! misspelt!! You should have only 7 letters in your name!

⟺ Byte 7F ↑ for 15 bytes!

15 bytes

Jump 15 starting with this + at line 30

— to jump 7 bytes

-END. is cc|08|29 ↑1 ↑2 ↑3 Byte: 1, 2, 3

```
        11 5
          12 "x"
        13 +
        14 +
        15 RCL 01
        16 OCT
        17 HMS
        18 HMS
        19 FRC
        20 FACT
        21 -
        22 +
        23 +
        24 +
        25 +
        26 +
        27 +
        28 CLX
        29 +
        30 .END.
          GTO .028
          GTO .029
               Σ0
@****LA
 7
 02 +
 03 +
          GTO "TST"
              PACK
PACKING
          GTO .025
          LIST 010

        25 +
        26 +
        27 +
        28 +
        29 +
        30 CLX
        31 +
        32 .END.
          GTO .030
          GTO .031
               Σ0
@**LA)*
          ASTO X
"@"    ***
          XEQ "AN"
"@"    ***
GTO "TST"
 GTO .031
 GTO .010
 11 +
 12 +
              PACK
PACKING
          GTO .028
          LIST 010

        28 +
        29 +
        30 +
        31 +
        32 CLX
        33 +
        34 .END.
```

= 77 (byte 119)

Woops! error!

Jump 7

to remove nulls between the jumping location and .END.

Woops! Error again!

Display in X

— Print of X! note lower half of hex table bytes not printed. and Ø8 not printed after CC!

L Δ

1040000<<0829

alpha prefix + cc|08|29

error!

remove those nulls!

moved two bytes toward .END.

next page!

Here I make the NNN

C0/01/29/00/00/00/00/ ← 'packed' .END.

192 1 41

to go at the top of file 1 – an .END. which is one reg from R00 – chain intact. to R00

```
                GTO .033
                   Σ0
@L&)***
                RCL [
               XEQ "AN"
    -0.<<0829000    ***
40<<0829000000
+ cc/08/29     GTO "TST"
                GTO .034
                RCL b
               XEQ "AN"
  0.000000003-88    ***
00000000003112
               GTO "TST"
                GTO .002
                RCL b
               XEQ "AN"
  0.000000002-80    ***
0000000000211:
               GTO "TST"
                GTO .003
                RCL b
               XEQ "AN"
  0.000000001-80    ***
0000000000111:
               GTO "TST"
                RCL b
               XEQ "AN"
  0.000000002-95    ***
0000000000211:
               GTO "KA"
               GTO .000

 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +

                PACK

PACKING

               GTO .001
               LIST 014

 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +

 08*LBL "KA"
 09 "PRS?"
 10 PROMPT
 11  E
 12 -
 13  E3
 14 /
```

print of alpha

Back to last file

Byte 3 of reg (here) 112 = location of last +

location of 1st byte of LBL "TST" = Byte 2 of 11A

Byte 1 of 11A

Byte 2 of 11B (END before LBL TST

7 bytes at top of pgm memory

Top of KA pgm. in 1st pgm file.

```
                XEQ "NN"
  13.00000000    RUN
                XEQ "NN"
 192.0000000     RUN
 1.000000000     RUN
 41.00000000     RUN
 0.000000000     RUN
 0.000000000     RUN
 0.000000000     RUN
 0.000000000     RUN

               GTO "KA"
               GTO .004
                  RCL b
                 XEQ "b2"
Y=NNN, X=Rb?
                    RUN
                   PACK
PACKING
                   CAT 1
                 GTO "KA"
                 GTO .001
                   PRP ""

 01 .END.
               GTO "KA"
                 PRP ""

 01 .END.
               GTO "KA"
               LIST 005

 02*LBL "KA"
 03 "PRS?"
 04 PROMPT
 05  E
 06 -
                   CAT 1
LBL'KA
LBL'KC
LBL'NN
                   PACK
PACKING
                 GTO "KA"
                 GTO .001
                 GTO "KA"
                   RCL b
  0.000000000-44    ***
                 ENTER↑
                   CLX
    0.0000000000    ***
                 XEQ "b2"

              01*LBL "b2"
                  TONE 0
                  TONE 9
          "Y=NNN, X=Rb?"
                 PROMPT
Y=NNN, X=Rb?
                    RUN
NONEXISTENT
```

error!

Didn't clear pgms

this is the .END. made + put in just now.

Both 7 bytes still locked up

Notice line number!

KA still all there.

attempt to remove that .END. at top of memory.

lock out of keyboard here – batteries out, in and ---

memory lost here!

I read in KA and b2 once more + RCL b and byte jumps from cards.

```
                SIZE 020
                  GTO ..
PACKING
                   CAT 1
LBL'KA
LBL'KC
LBL'NN
END          287 BYTES
LBL'b2
.END.        98 BYTES
                   PACK
PACKING
                   CAT 1
LBL'KA
LBL'KC
LBL'NN
END          287 BYTES
LBL'b2
.END.        98 BYTES
                GTO "KA"
                GTO .000
 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
                   PACK
PACKING
                XEQ "NN"

            159*LBL "NN"
                   CLA
                 ASTO 00

             162*LBL 02
          192.0000    RUN
            1.0000    RUN
           45.0000    RUN
                      RUN
OUT OF RANGE
                XEQ "NN"
          192.0000    RUN
            1.0000    RUN
           45.0000    RUN
            0.0000    RUN
            0.0000    RUN
            0.0000    RUN
                   FIX 9
     0.000000000    RUN
    -0.012=00000    ***
                GTO "KA"
                GTO .004
                     1/X
                  LASTX
                  RCL b
                XEQ "b2"
Y=NNN, X=Rb?
                      RUN
                   PACK
PACKING
```

PGM at top of memory

Errors here!

−cϕ 1st byte
−ϕ1 2nd byte
29 3rd byte
Packed .END. reg from R00

4 nulls

Print of NNN

error! not in USER mode for RCL b!

NNN stored in 1st register of file 1 just below R00

attempt to clear pgm memory

```
                   CAT 1
                   CAT 1
LBL'KA
LBL'KC
LBL'NN
END          294 BYTES
LBL'b2
.END.        98 BYTES
                GTO "KA"
                PRP "KA"

.END.

                GTO .000
                   PACK
PACKING
                   PRP ""

.END.
                GTO "KA"
                LIST 004

 02*LBL "KA"
"PRS?"  PROMPT   E
                GTO "KA"
                GTO .000
                  RCL b
                 ENTER↑
                    CLX
                XEQ "b2"
Y=NNN, X=Rb?
                      RUN
NONEXISTENT
                GTO ..
PACKING
PACKING
TRY AGAIN
                SIZE 020
                  FIX 9
                  GTO ..
PACKING
                GTO "KA"
                GTO .000
 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
                   PACK
PACKING
                XEQ "NN"
     192.0000000    RUN
     1.000000000    RUN
     32.00000000    RUN
     0.000000000    RUN
     0.000000000    RUN
     0.00<001200    ***
     0.000000000    RUN
     0.<00120000    ***
     0.000000000    RUN
    -0.012000000    ***
```

Didn't work again!

new pgm lost in .END.

note line no. : .END. is 'line' 001!

lockout – bad output/in new lost.

Same result again! Not sure why or where!

KA and b2 put in again etc.

cϕ
ϕ1
2ϕ

```
            XEQ "NN"
   192.0000000    RUN
0.000000000+<0    ***
   1.000000000    RUN
   00.0000000<    ***
   32.00000000    RUN
0.000001200-80    ***
   0.000000000    RUN
   0.0000<0012    ***
   0.000000000    RUN
            XEQ "NN"
   32.00000000    RUN
            XEQ "NN"
   192.0000000    RUN
   1.000000000    RUN
   32.00000000    RUN
   0.000000000    RUN
   0.000000000    RUN
   0.000000000    RUN
   0.000000000    RUN
            GTO "KA"
            GTO .005
               RCL b
            XEQ "b2"
Y=NNN, X=Rb?
               RUN
              PACK
PACKING
              CAT 1
LBL'KA
LBL'KC
LBL'NN
END         294 BYTES
LBL'b2
END          93 BYTES
.END.        05 BYTES
            GTO "KA"
              PACK
PACKING
              CAT 1
LBL'KA
LBL'KC
LBL'NN
END         294 BYTES
LBL'b2
END          93 BYTES
.END.        05 BYTES
            GTO "KA"
            GTO .000
 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
              PACK
PACKING
PRIVATE
            GTO "KA"
NONEXISTENT
              CAT 1
END         14 BYTES
PRIVATE
```

Start again

ad again (my muddle-thought was wrong - but ok)

cØ
Ø1
2Ø

.END. after a GTO ..
Pack.
(acc. to Close)

Store .END. at toh of file!

Remove Pgm?

No!

Try again.

No!

Put in before the .END.

Pgm at toh was PRIVATE! but that Pack cleared all pgm apart from 7+ = private pgm.

```
            GTO ..
PACKING
              CAT 1
END         10 BYTES
.END.       04 BYTES
              CAT 1
END         10 BYTES
PRIVATE
PRIVATE
PRIVATE
              RUN
PRIVATE
              RUN
PRIVATE
              CAT 1
END         10 BYTES
PRIVATE
PRIVATE
PRIVATE
              Σ0
×iØØ
            GTO ..
PACKING
              CAT 1
END         10 BYTES
PRIVATE
              CAT 1
END         10 BYTES
.END.       04 BYTES
```

both the .END. put in is now a (pseudo?) private END, not a .END.

Won't run!

Trying anything! print of

169 regs on SIZE 020
master Clear - KA + b2 in again.

```
            SIZE 020
              GTO ..
PACKING
            XEQ "NN"
          152+LBL "NN"
              CLA
             ASTO 00
          155+LBL 02
   192.0000    RUN
    1.0000    RUN
   32.0000    RUN
            GTO "KA"
            GTO .000
 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
              PACK
PACKING
```

make NNN ØØ/ØØ/ØØ/ØØ/CØ
↳ Ø1/2Ø

Storage location at toh of file!

```
                GTO .005
                   RCL b
                  XEQ "b2"
Y=NNN, X=Rb?
                      RUN
                GTO "KA"
                LIST 003

             02♦LBL "KA"
             03 "PRS?"
             04 PROMPT
                GTO .000
                LIST 005
.END. REG 115 →
             01 .END.
                  PRP ""

             01 .END.
 01 +
                     PACK
PACKING
                    CAT 1
                    CAT 1
.END.          07 BYTES
.END. REG 170
```

NNN stored.

result as before

put in above the newly inserted .END

Pgms all cleared after the PACK.

```
                    CAT 1
 LBL'KA
 LBL'KC
 LBL'NN
 END           294 BYTES
 LBL'b2
 .END.          98 BYTES
                  GTO "KA"
                  GTO .001
 01 +
                     PACK
 PACKING
                    CAT 1
 .END.          07 BYTES
                  GTO ..
 PACKING
                 GTO "KA"
                 XEQ "NN"

              152♦LBL "NN"
                     CLA
                   ASTO 00
       192.0000000    RUN
       1.000000000    RUN
       45.00000000    RUN
       0.000000000    RUN
       0.000000000    RUN
       0.000000000    RUN
       0.<0012=000    ***
       0.000000000    RUN
                 GTO "KA"
                 GTO .002
                   RCL b
                  XEQ "b2"
Y=NNN, X=Rb?
                      RUN
                    CAT 1
                 GTO .000
 01 +
                     PACK
PACKING
                   PRP ""

              01 +
              02 RCL 13
              03 END
                    CAT 1
END           11 BYTES
PACKING             PACK
```

No?

up at top of PGM memory. key in byte

Pgms gone. read in again.

in trace!

c♦/♦1/2D/♦♦/♦♦ unhacked .END.

loaded into PGM.

Repeat-

```
                GTO ..
PACKING
                GTO "KA"
                GTO .000
 01 +
 02 +
 03 +
 04 +
 05 +
 06 +
 07 +
                   PACK
PACKING
                GTO .005
                XEQ "NN"

             159♦LBL "NN"
                    CLA
                  ASTO 00

             162♦LBL 02
                   STOP
          192.0000    RUN
            1.0000    RUN
           32.0000    RUN
                  FIX 9
                GTO "KA"
                GTO .005
                   RCL b
                  XEQ "b2"
Y=NNN, X=Rb?
                      RUN
                     PACK
PACKING
```

at top of memory.

error!

in TRACE mode!

value ♦♦/♦♦/♦♦/♦♦/ c♦/♦1/2♦

Store NNN

Remove PGM?

#