Notes on the back story of this letter:

This *5-page* letter is also one of the first I sent to ***Richard Nelson***, who was in charge of *PPC* back then. By now some of my contributions had been published, at long last, and my fledgling self-taught English was improving a little, which encouraged me to send more materials.

In this letter I focused mainly on various aspects of the *PPC ROM Project* logistics, asking a number of questions (which, as always, went unanswered by Mr. Nelson) and suggesting that it would be highly desirable and useful to publish an special issue (which could well be a double one) fully dedicated to the many routines being submitted for the *ROM*, to raise public awareness, discussion, improvement and even voting. I also included a couple of additional contributions for the *ROM*, (which weren't selected) namely these two synthetic routines:

- A short, efficient *Byte Counter* routine which would help determine the length of any program or consecutive programs without requiring a printer, as well as the byte count between arbitrary lines in program memory, which the printer wouldn't provide. It would even automatically create a required synthetic assignment, needing no inputs at all.

- The second one was *Reset Flags*, a very short synthetic routine which would reset the status of all flags from *00* to *55* to their *Master Clear* default values in one go, taking just half a second and leaving all registers and the whole stack + **`LAST X`** undisturbed.

I also included an input for the *"Feedback"* or *"41 notes"* sections, providing a list with names and functionalities for ~ fifty *"Flag 30 Catalog"* functions, giving the actual function being executed upon filling up their prompts (digits, alpha, **`IND`**, stack, etc.), which featured such gems as **`eG0BEEP`** and **`$T+N IA`**, allowing programming printer functions without a printer, all 127 local labels, etc.

Valentin Albillo, 21-12-2021

Richard Nelson
Editor, PPC Journal
2541 W. Camden Place
Santa Ana , CA 92704
U. S. A.

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

Hello, Richard , how are you ?

Too much work for you, isn't it ? I hope the Club copier project is going allright. I'll send you some dollar bills as soon as I can get them (it is not very easy here in Spain). Do you liked any of my past contributions ? As you may have noticed, I am very interested in the PPC Custom ROM. If its contents are what they promise to be, I may order as many as 5 ROMs : they will be a fantastic publicity for the Club; this is, I am not very sure if members are - allowed to buy more than one . If a member can only buy a single ROM , sorry. Now , a question : are there any legal restraints to buy a ROM in the US from here in Spain ? (I mean Custom laws, or the like). Can foreign members buy ROMs freely ? Will they have to pay more for the ROM than US members ? .

Anyway, here is a suggestion: it would be a very good idea to dedicate a special number of PPC CJ to the already submitted routines for the ROM (much like the "Special Routines issue (V5 N7) ". Most of the submitted routines will appear in that number, so that members can ~~bring~~ opinions, criticism, comments, even improvementto some of the routines. I , of course, agree with the open review procedure policy. I am very amazed that, although we are almost in mid summer, so few - routines for the ROM have appeared in the PPC CJ. I am very sure that many, many members should have submitted quite a lot of routines, ideas,etc. Why have them not been published ? You should probably publish them all in the previously mentioned special - issue (which may be a double issue if that is necessary) : members should vote every routine, together with comments. Someone (the ROM Committee, I guess) should compile the votes and comments, to make a final selection for the ROM. That selection must be public. So, even members which are not going to buy the ROM will take advantage of the ROM routines for their use in RAM .

Well, here included are some submittals:

- Two routines for the ROM : one is a Byte Counter routine, as described in V6 N8 P15. It will count the bytes used by any program, or the bytes between any two specified memory locations. It allows the user to determine program length without the use of a printer. Even members with printer will find this routine useful,as the printer - does not give byte count between program lines, or other than full programs. The present routine uses no register, disturbs no flag, and is completely automatic. Even the necessary assignment of RCL b to a key is done automatically by the routine if requested, requiring no input at all from the user, in less than a second. Byte count itself takes less than 5 seconds, regardless of the number of bytes. To count the bytes, all the user has to do is going to the first location, press R/S, go to the second location, press R/S, XEQ "BC" and the number of bytes appears in the display, in less than 5 seconds, labeled as nnn BYTES.Magnetic card included.

  The other is a RESET FLAGS routine: it is only 27 bytes long, uses no register, leaves the whole stack X,Y,Z,T,L, unaltered, executes in less than a second, and resets the status of all flags 00-55 to its MASTER CLEAR default value. It takes into account if a printer is present or not: if the printer is present, flags 21 & 55 are set. If not, both are cleared. This routine can be very useful: as an initialization routine, it is unique, as it specifies FIX 4, DEG, beeper active, USER off, ALPHA off, etc. (including flag 21 matched to flag 55), almost instantaneously (about half a second). It uses synthetic text lines and functions.Mag card included

- A list of "ghost" functions , as ASCI _ _, etc, whose names appear in the flag 30 Catalogs. The list includes the name and known function(s) of most of them which are known to me. All those functions may be generated using the KEY ASSIGNMENTS program with appropriate inputs. Some of them are very useful, as they can generate unsupported local labels directly, together with its GTO, XEQ. The "digit" functions, allow the editing as a line of program of any combination of digits, .,-,exponent, etc. Some functions as PRIVATE _ _, and 00 REG _ _ _ are not documented, yet, but they do exist and seem to execute in program memory. They are not included in the list, as I am not very sure how they are generated, not to mention its function.

Excuse me for writing you so frequently. Hope you like some of this stuff.

Best regards :

(4747)

P.D : Magnetic cards are included
with the ROM routines recorded onto them

-this is an input to "FEEDBACK", or "41 notes" , etc...

- -

- The discovery of the "flag 30 Catalogs" (see V7 N4 P26 , 27) lead many members to think what those odd names of "ghost functions" may indicate. Here included is a list of the identification of several of the functions that appeared in those 3 catalogs:

NAME	FUNCTION	
(blank)	∅ (zero)	
$T+N IA _ _ _	1, XEQ local(all locals, even A, etc)	
_ _	2, RCL 06	
µ	3, GTO 00 , RCL 15	
DCAB⊏⊏¯a	4	
X	5	
H_LD	6	
A¯	7	
¯2 _ _	8, CHS, spare, GTO nn, RCL 08, STO 05-15	
OD	9, RCL 11, STO 08 , XEQ local	
-	., XEQ local	
H	E, LBL	
⊏ AH̄H̄	GTO alpha	
A	RCL 01, RCL 12, GTO 00	
IEW̄	RCL 02	
C _ _	RCL 03	
	RCL 04, STO 13, STO 14, GTO 00	
ᵀbD̄CAB⊏⊏¯a	RCL 05, RCL 10	
∅	RCL 07, GTO 00, XEQ 00 , ∅	
≻Y	RCL 09	
dSƆ_µ	RCL 13	
X(=Ȳ?	RCL 14, X(=Y?	
ASCI _ _	STO 00	(Note : I have seen many -
9 _ _	STO 01	others, such as PRIVATE _ _
RĒG _ _	STO 02	but its function is not
STX	STO 04	clear to me up to now. All
		this functions may be crea-
·8 _ _	STO 06	ted using the SYNTHETIC KEY
+	STO 10 , +	ASSIGNMENTS program in -
≻SF _	STO 11	V7 N3 P3, using 0 ENTER
LST	GTO 00	nnn ENTER key R/S)
W	GTO 00	
↗_ _	XROM 00,00 thru XROM 03,52 (at least)	
µ	XROM 04,00 thru XROM 07,52 (id.)	
∅B̄ĒEP _	XROM 08,00	
S _	XROM 16,00	(local labels A,...,J,T,Z,
T	XROM 24,00	X,Y,L,M,...,etc exist)
eG∅BEEP _ _	XROM 28,00 thru XROM 29,35 , LBL alpha	
∠=∅?	XROM 12,00	(there are 127 local la -
B _ _	GTO local	bels. All are useful. For
$ _ _	GTO local	instance, GTO X (local)
MS	GTO 00	addresses LBL X (local))
Wᵀ	SPARE	
d∠	GTO local	
≻ (f̄l̄ying g.)	GTO 00	
D	GTO 00	
R	GTO 00	
DCAB⊏⊏¯a _ _	XEQ nn	
¯a	XEQ 00	
⅄ _ _	XEQ local (even local X,Y,Z,T,L)	

This list is not exhaustive at all. Not all possible functions belonging to each name are given. All underscore symbols are prompts. Most prompts may be filled using numbers, alpha characters, IND, ST (stack),X,Y,Z,T,L, and even +,-,*,/ (they become ST+,ST-,ST*,ST/ , then). Some of this functions are very useful. For instance, $T+N IA _ _ _ , if you fill the prompt, becomes XEQ 00 thru XEQ e (including - A,B,...,J,T,X,Y,Z,L,M,...,append, etc, local labels)

-here is an input to the ROM PROGRESS column

- -

Certainly, the ROM ideas suggested in V6 N8 P15 are a source of inspiration to all those people interested in the ROM progress. Here included is a routine that should be considered for its inclusion in the PPC Custom ROM . It is my answer to the request for a Byte Counter, defined as : " ... gives program byte count without printer attached".

This routine, called "BC" (byte counter), allows the user to count the bytes used by any program, or between any location in program memory and another location. Each location may be in a different program , as there are no limits to the procedure. It may also be useful to find how many program memory is used at any moment. Simply, use the routine to count the bytes between the first line of the first program in memory, and the final END, and there you are.

The routine is fully automatic : you simply, go to the beginning of the part whose bytes you want to - count, press a key, then go to the last line, press the same - key, XEQ "BC" , and a message nnn BYTES will appear in the display. No registers, flags, are used at all.

```
01 LBL"BI"     19 XEQ 01     37 Ø          55 "⊦¯¯¯¯天"
02 "天圖⊦¯圖"   20 -          38 X() N      56 RCL M
03 ASTO X      21 CLA        39 XEQ 01     57 FRC
04 X() c       22 RCL d      40 7          58 10
05 "圖圖圖8¯¯¯" 23 FIX 0      41 *          59 *
06 RCL M       24 CF 29      42 X()Y       60 X() M
07 STO 00      25 ARCL Y     43 XEQ 01     61 INT
08 X()Y        26 "⊦ BYTES"  44 STO M      62 1
09 X() c       27 STO d      45 16         63 *
10 RCL ⊦       28 PROMPT     46 ST/ M      64 16
11 X() d       29 LBL 01     47 MOD        65 *
12 SF 04       30 X()Y       48 1792       66 LBL 02
13 X() d       31 CLA        49 *          67 RCL M
14 STO ⊦       32 STO M      50 +          68 INT
15 "OK"        33 "⊦■■■■■■■" 51 GTO 02     69 +
16 PROMPT      34 CLX        52 LBL 01     70 END
17 LBL"BC"     35 X() N      53 STO M
18 XEQ 01      36 "⊦*"       54 RDN
```

This routine is 70 lines, 156 bytes. Does not use any register (except the stack and status registers), no flags are changed at all. The first part,"BI", is an optional initialization routine that automatically - assigns RCL b (XROM 01,60) to the R/S key. No inputs are required. This is done to fully automatize the procedure. If the user has already assigned RCL b to a key, he or she may skip this initialization, and use its own RCL b key instead of R/S. Otherwise, simply XEQ "BI" and , in less than a second, the message OK appears in the display. Now RCL b is assigned to the R/S key for its execution in USER mode. This is convenient as : although R/S is assigned, it will stop any program, catalog listing, etc, even in USER mode, regardless of the assignment, so R/S does not lose all of its functions. Besides, it may be reassigned at will, using ASN (any) R/S.

Executing BI does not use any register, does not change any flag. No "ghost" assignments are created, only RCL b to the R/S key. However, if they were previous assignments, the last two will be lost, and changed to ABS. All other assignments remain unchanged. If you want to preserve all your previous assignements, simply assign anything to any two keys not already assigned, then XEQ "BI" . The dummy assignments will be cleared, but no other assignment will be changed at all.

Line 02 is F501690C00BF
line 05 is F7F0907C38000000

line 55 is "append 0000000001" , this is
F67F0000000001

Detailed explanation

BI : lines 02-04 create the new temporal contents of Rc, which is 100169CC00BF. lines 05-07 store the new assignment register , F0907C38000000 , in the place of the last two - assignments (if any). F0 is the first byte of an assignment register. 907C is the code for RCL b. 38 is the code for the R/S key. The nulls are necessary to make a full 7 bytes. Lines 08-09 restore the original contents of Rc. Lines 10-12, set the appropiate flag (04) in the proper status register, to confirm the assignment to the R/S key. No other assignments are disturbed or created. lines 13-14, restore the status of all flags, and the work is done.

BC : assuming the two necessary pointer locations are in X,Y , routine 01 (first) is executed twice, to convert each address to a decimal number. Both are sustracted, to yield the number of bytes between both addresses. Lines 22,27 assure that no status information is changed (FIX, etc). lines 30-38, separate the two bytes of the address, before converting them to decimal. the second LBL 01 is a modified version of UNBLD (see V7 N2 P37). Changes to BC are very critical, as most intermediate results, addresses, etc , are carried stored in the stack.

Characteristics : -completely automatic
-no registers used
-no flags used

-if BI is not used, nothing is changed in machine status
-if BI is used, RCL b is assigned to R/S . Any assignment to the R/S key is lost, of course. The last two assignments are changed to ABS (default). No ghost assignments are created. R/S is still useful to stop any program, etc.
-execution times : BI , less than a second
BC , 5 seconds

INSTRUCTIONS : -first of all, (a) if you already have RCL b assigned to some key, and you want to use that key, use it instead of R/S (b) if you haven't RCL b assigned, there are two possibilities : (1) you have no assignments at all (user's program assignments don't count, as they are stored with the program label, not in the assignments registers) , then XEQ "BI", will assign RCL b to R/S . The same is valid if you have some assignments, but don't care about the last two being removed. (2) if you have some assignments, and you don't want to lose the last two, then assign any function (not program) to two keys, then XEQ "BI". This will assign RCL b to R/S, and the dummy assignments will be removed, leaving all others undistrubed.

```
                    -the following assumes RCL b is assigned to R/S
 ...                -suppose you want to count the bytes used by the
 END                 program called "PEPE". Do as follows:
|LBL"PEPE" ]
|...       |         GTO "PEPE" , (user) R/S (user)
|...       }         GTO "JOSE" , (user) R/S (user)
|END       ]         XEQ "BC" → nnn BYTES
|LBL"JOSE" ]        -now, you want to count the bytes used by the
|...       |         program "JOSE", which is the last program in
|...       }         memory; do the following:
|END       ]         GTO "JOSE" , (user) R/S (user)
 .END.               GTO ..     , (user) R/S (user)
                     XEQ "BC" → nnn BYTES
===========
 ...                -now, to count the bytes between line 25 and
|25 STO 38 ]         47, both included, do as follows:
|...       }         GTO .025 , (user) R/S (user)
|47 LAST X ]         GTO .048 , (user) R/S (user)
 48 SIN              XEQ "BC" → nnn BYTES
 ...
```

-to count all used bytes in program memory:

```
 LBL"FIRST"  }  CAT 1 , R/S (as soon as the first LBL or
 ...         }               END appears)
 ...         }  RTN , (user) R/S (user)
             }  GTO ..(user) R/S (user)
 .END.          XEQ"BC" → nnn bytes
```

-of course, many keystrokes may be saved, by simply not switching out of USER if it is not necessary (i.e, if only R/S is assigned, or CAT and GTO are not assigned)

TEST EXAMPLE : assume you have just loaded Byte Counter. and you havent RCL b assigned. To count the bytes used by BI, then by BC, then by the whole program:

```
 01 LBL "BI"   }
 ...           }
 17 LBL "BC"   }
 ...           }  XEQ"BI"  → OK   (now RCL b is in R/S)
 70 END        }  GTO "BI" , (user) R/S (user)
 71 .END.         GTO "BC" , (user) R/S (user)
                  XEQ "BC" → 44 BYTES  occupied by BI

 GTO "BC" , (user) R/S (user)
 GTO ..   , (user) R/S (user)
 XEQ "BC" → 112 BYTES ocuupied by BC

                  GTO "BI" , (user) R/S (user)
                  GTO ..   , (user) R/S (user)
                  XEQ "BC" → 156 BYTES occupied by the whole
```

-of course, 44 + 112 = 156 bytes.

WARNINGS : -the line or the program may be accessed by any means, GTO alpha, or GTO .nnn, or GTO.., or CAT.

- BI must not be used if there are no free registers in program memory to store a single assignment (this is, there is not a single assignment already made, nor place for any).

VALENTIN ALBILLO (4747)

I read carefully the letter of Larry Fox (2596) published in the ROM PROGRESS column of the MAY issue (V7 N4 P12). I think most of the routines he suggests for the ROM are intended to be micro code level routines: for instance, FS?S , FC?S are only suitable as microcode. Otherwise, FS?S is equivalent to FS?,then SF, which would be horribly ineficcient in a user-level ROM routine.

He asks for a RESET FLAGS routine, which should reset the status of all flags to its initialization (MEMORY LOST) default values. Here included is a routine, to be considered for its inclusion into the PPC Custom ROM, which does exactly that: all flags are reset to its MASTER CLEAR value: no registers,etc are used. Only - - - the ALPHA register is cleared. X,Y,Z,T & LAST X remain unchanged. The routine takes into account the value of flag 55 before the call. If flag 55 is set (printer present), flag 55 and flag 21 are set. If flag 55 is clear (no - printer), flag 55 and flag 21 are cleared.

```
01 LBL"FR"
02 ",人▩‾"
03 FS? 55
04 "人,人▩天"
05 ASTO d
06 CF 03
07 CLA
08 END
```

The routine, called FR (Flag Reset) , is 8 lines, 27 bytes. Uses no register, leaves the whole stack, X,Y,Z,T,L , unaltered. Clears ALPHA register. To use, simply XEQ "FR" .

flags 00-20,22-25,27,30-36,38-39,41-54 : clear
flags 26,28-29,37,40 : set
flags 21,55 : if printer present (55 set): set
              if no printer (55 cleared):clear

- this exactly reproduces the status of all flags after a master clear. Of course, the display will be FIX 4, DEG, comma separators active, beep active, out of USER & alpha, etc.

(execution time: less than 1 sec) VALENTIN ALBILLO (4747)

line 02 is F42C048000 ; line 04 is F5042C048001