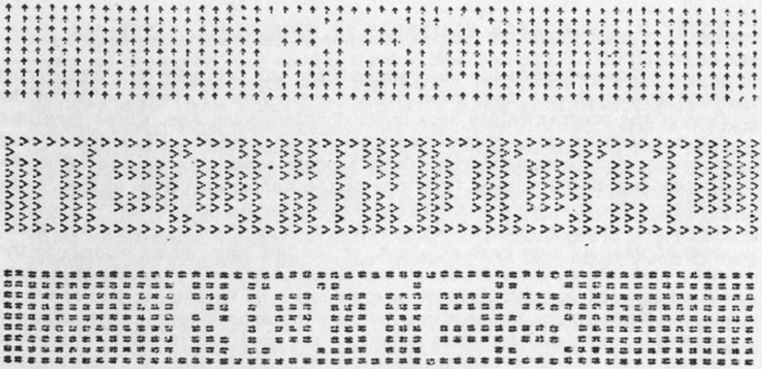


PPC TECHNICAL NOTES #13

CONTENTS

REGISTERED BY AUSTRALIA POST PUBLICATION NO. VBG4080 (CATEGORY B)

TIC TAC TOEING THROUGH THE TULIPS — *Tiny Chris Rath (6287)*

Tic Tac Toe seems to be one of those games that PPC owners like to programme into their machines; if they have the memory. So, let's talk about some of the major points one encounters during the writing of such a programme.

One of the first points that appears is, "How to store the board?" Remember, that it, your method, must be memory efficient in two ways: 1) Data storage space; 2) Programme memory used to access a board location.

There are three methods that quickly spring to mind: 1) Devoting a register to each square; 2) Using flags; 3) Or using a single register for the entire board, and nine digits right of the decimal.

Devoting a reg./sq. violates our memory efficiency decision, and is not really a consideration at all. The other two are fine to use; and also, they do not require synthetics, a big plus (not implying the first uses them). Number 3) is the better of the two though. It allows distinction between 'Unoccupied', 'X-occupied', and 'O-occupied'; whereas the flags do not. This may or may not be worth considering, depending upon your 'system to win'.

Let us consider the board for a moment. If we number the squares as showen in fig. 1, then they correspond to the keyboard; making things easy for the user.

7:8:9
4:5:6
1:2:3

fig.1

Also, about the board; as well as nine squares, there are nine ways to win. Let us call these 'win lines'. Obviously, each time the 41c is required to make a move, it must look at all nine win lines (in the worst case). A question thus arises: Do we keep track of the win lines in memory, or do we recalculate each time? Execution time, and available memory are the consideration here. If you can spare the memory, then it speeds execution time greatly if a running status of the win lines is kept.

The next decision is a fairly major one; how to represent the win lines in storage, or in the programme after calculation? This will determine whether to assign a single reg. to each line, or to use the flag or single reg. for all options.

We can immediately cross the flag option off our list. It only allows a there/not there, representation; which is insufficient. The difference between individual or single reister usage is also great (from a programme execution point of view). The key point being that while a single reg. for all allows sufficient differentation, it does not allow negative number representation.

Win Line States	Man Combinations	#3
1 Null Status	No men in line	Ø
2 Possible threat	1 opponent in line	-1
3 Possible win	1 of own in line	1
4 Must block	2 of opponent	-2
5 Will win	2 of own in line	2
6 No Threat	2 and 1 combination One of each	1 Ø

Fig. 2

This may seem trivial, but, what kind of change are you going to effect to the win line when a piece is played? There are 8 combinations of men, but only six win line states. Because it is the win line state that ultimately matters, it is probably best if we store that, as opposed to the actual man combinations.

There are, without a doubt, many ways to store this status. But let us look at one that is very simple to visualize and use. Using the single reg./win line; begin with all the registers at zero. Now, what happens if we add 1 to the appropriate registers when the 41c moves, and subtract 1 when its opponent moves. As you can see, by column #3 in fig. 2, the states are very well defined, and look simple to test. Remember that there is always more than one register to add to or subt. from; with square 5, as an example, there are four reg.s to effect changes to. This method of representation is simple enough that is probably worth translation to and from this notation if you are using the single reg. for all method.

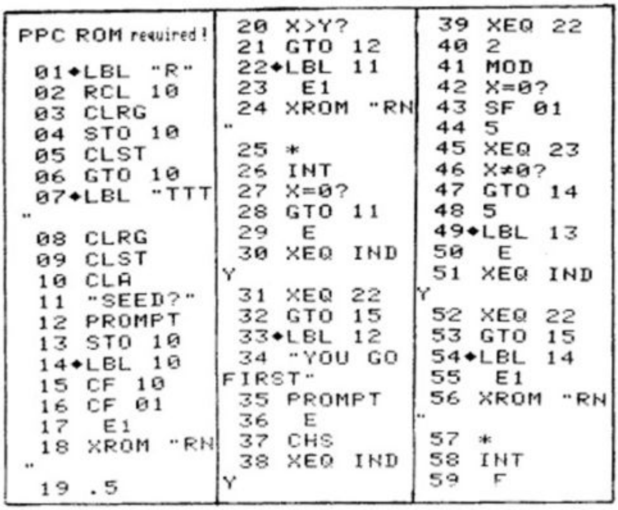
The only remaining question concerns strategy. It is important to note here that sq.5 is used in the most win lines, and that the next most used sq. is any one of the corners. Since it is possible to tie, or win, starting from any square, we can conclude that the first move made in a game is of no concern. The second move made does, however, have an optimum: If the centre was not taken by the first move then it should be the second move; if the centre is already occupied then any one of the corners is the best move. There are other subtleties, but the interested reader can play out several games and determine them his-her-self.

* * * * * * * * * * *

The Tic Tac Toe programme showen here uses a reg./win line, and a single reg. for all squares, for board storage. Because win lines are kept current, the board storage only needs to indicate (un)occupied. I didn't use flags because I have too many other routines that use them.

To play ^TTT, just XEQ^TTT. When you are prompted for a "SEED?", enter a number between Ø and 1. The 41c will then respond with either its first move, or the message, "YOU GO FIRST", depending upon the fisrt RN chosen. THE PPC ROM is required, or check back issues of the Journel for a listing of ^RN. To enter your move, or interpret the 41c's,use the digits one to nine of the keyboard, as I suggested in the discussion.

2EØ,
Chris Rath (6287)
1281 Agincourt Rd., Ottawa, Ontario, Canada, K2C 2J3



PPC ROM required!

```
 01♦LBL "R"
 02 RCL 10
 03 CLRG
 04 STO 10
 05 CLST
 06 GTO 10
 07♦LBL "TTT
"
 08 CLRG
 09 CLST
 10 CLA
 11 "SEED?"
 12 PROMPT
 13 STO 10
 14♦LBL 10
 15 CF 10
 16 CF 01
 17  E1
 18 XROM "RN
"
 19 .5
 20 X>Y?
 21 GTO 12
 22♦LBL 11
 23  E1
 24 XROM "RN
"
 25 *
 26 INT
 27 X=0?
 28 GTO 11
 29  E
 30 XEQ IND
Y
 31 XEQ 22
 32 GTO 15
 33♦LBL 12
 34 "YOU GO
FIRST"
 35 PROMPT
 36  E
 37 CHS
 38 XEQ IND
Y
 39 XEQ 22
 40 2
 41 MOD
 42 X=0?
 43 SF 01
 44 5
 45 XEQ 23
 46 X≠0?
 47 GTO 14
 48 5
 49♦LBL 13
 50  E
 51 XEQ IND
Y
 52 XEQ 22
 53 GTO 15
 54♦LBL 14
 55  E1
 56 XROM "RN
"
 57 *
 58 INT
 59  F
```

```
60 X=Y?
61 GTO 13
62 CLX
63 3
64 X=Y?
65 GTO 13
66 CLX
67 7
68 X=Y?
69 GTO 13
70 CLX
71 9
72 X=Y?
73 GTO 13
74 GTO 14
75♦LBL 15
76 STOP
77  E
78 CHS
79 XEQ IND
Y
80 XEQ 22
81 8
82 3
83 ENTER↑
84 XEQ 24
85 8
86 2
87 ENTER↑
88 XEQ 20
89 FS?C 00
90 GTO 15
91 8
92 -2
93 ENTER↑
94 XEQ 20
95 FS?C 00
96 GTO 15
97 5
98 XEQ 23
99 X≠0?
100 GTO 16
101 5
102  E
103 XEQ IND
Y
104 XEQ 22
105 GTO 15
106♦LBL 16
107 8
108  E
109 ENTER↑
110 XEQ 20
111 FS?C 00
112 GTO 15
113 8
114 0
115 ENTER↑
116 XEQ 20
117 FS?C 00
118 GTO 15
119 RTN
```

```
120♦LBL 20
121 CLX
122 RCL IND
Z
123 X=Y?
124 GTO 21
125 DSE Z
126 GTO 20
127 RTN
128♦LBL 21
129 CLX
130 30
131 ST+ Z
132 XEQ IND
Z
133 CLX
134  E
135 XEQ IND
Y
136 XEQ 22
137 8
138 3
139 ENTER↑
140 XEQ 24
141 R↑
142 SF 00
143 RTN
144♦LBL 22
145 X<>Y
146 CHS
147  E1
148 X<>Y
149 Y↑X
150 ST+ 00
151 LASTX
152 ABS
153 RTN
154♦LBL 23
155 ENTER↑
156 ENTER↑
157  E
158 -
159  E1
160 X<>Y
161 Y↑X
162 RCL 00
163 *
164 FRC
165  E1
166 *
167 INT
168 RTN
169♦LBL 24
170 CLX
171 RCL IND
Z
172 ABS
173 X=Y?
174 GTO 25
175 DSE Z
176 GTO 24
177 RTN
```

```
178♦LBL 01
179 ST+ 01
180 ST+ 04
181 ST+ 08
182 RTN
183♦LBL 02
184 ST+ 04
185 ST+ 07
186 RTN
187♦LBL 03
188 ST+ 02
189 ST+ 03
190 ST+ 04
191 RTN
192♦LBL 04
193 ST+ 05
194 ST+ 08
195 RTN
196♦LBL 05
197 ST+ 01
198 ST+ 02
199 ST+ 05
200 ST+ 07
201 RTN
202♦LBL 06
203 ST+ 03
204 ST+ 05
205 RTN
206♦LBL 07
207 ST+ 02
208 ST+ 06
209 ST+ 08
210 RTN
211♦LBL 08
212 ST+ 06
213 ST+ 07
214 RTN
215♦LBL 09
216 ST+ 01
217 ST+ 03
218 ST+ 06
219 RTN
220♦LBL 31
221  E
222 XEQ 23
223 X=0?
224 RTN
225 5
226 XEQ 23
227 X=0?
228 RTN
229 9
230 XEQ 23
231 X=0?
232 RTN
233♦LBL 32
234 3
235 XEQ 23
236 X=0?
237 RTN
238 5
239 XEQ 23
```

```
240 X=0?
241 RTN
242 7
243 XEQ 23
244 X=0?
245 RTN
246♦LBL 33
247 3
248 XEQ 23
249 X=0?
250 RTN
251 6
252 XEQ 23
253 X=0?
254 RTN
255 9
256 XEQ 23
257 X=0?
258 RTN
259♦LBL 34
260  E
261 XEQ 23
262 X=0?
263 RTN
264 2
265 XEQ 23
266 X=0?
267 RTN
268 3
269 XEQ 23
270 X=0?
271 RTN
272♦LBL 35
273 4
274 XEQ 23
275 X=0?
276 RTN
277 5
278 XEQ 23
279 X=0?
280 RTN
281 6
282 XEQ 23
283 X=0?
284 RTN
285♦LBL 36
286 7
287 XEQ 23
288 X=0?
289 RTN
290 8
291 XEQ 23
292 X=0?
293 RTN
294 9
295 XEQ 23
296 X=0?
297 RTN
298♦LBL 37
299 2
300 XEQ 23
301 X=0?
```

```
302 RTN
303 5
304 XEQ 23
305 X=0?
306 RTN
307 8
308 XEQ 23
309 X=0?
310 RTN
311♦LBL 38
312  E
313 XEQ 23
314 X=0?
315 RTN
316 4
317 XEQ 23
318 X=0?
319 RTN
320 7
321 XEQ 23
322 X=0?
323 RTN
324 "CAT'S G
OT IT"
325 PROMPT
326 GTO "R"
327♦LBL 25
328 LASTX
329 -3
330 "I WIN !
!!"
331 X=Y?
332 "YOU WON
...."
333 PROMPT
334 GTO "R"
335 END

LBL'R
LBL'TTT
END
590 BYTES
```

PPC ROM required!

Micro code version listed above -

ROM XRUMS AND RUM XROMS *Chris Rath (6287)*

A question that you, as a 41c user, have probably thought about is, "Why don't XROM's prompt for their data, like STO, etc." This after looking at EPROM Box functions such as NSTO, NRCL and others.

The truth is that some XROM functions do prompt. The Printer function PRP is a good example of this. But, PRP and its other prompting friends are exceptions to the rule; so let's consider them first.

As anyone who has tried to put PRP in a programme can tell you, once you get it there, it looses its prompting ability and doesn't work quite the way you would like it to. This is the biggest reason that XROM functions do not prompt: The 41c assumes that all 'programmable' XROM functions are non-prompting. This is a reasonable enough answer to our first question, but as our nature would have it, another question arises: "Why does the 41c assume this?"

The explanation for this came when Paul Lind wrote a routine called PCAT. It is a port addressable CAT function, which really is the way Mother HP should have written the function in the first place, but that is a comment for the wish list. In any case, Paul made his function nonprogrammable, and prompting, like the origonal. Without a printer one would nover notice its, PCAT's, quirk, instead of taking the prompt responce and printing that, it takes the XROM number and interprets that as the data the user entered.

So, our question is answered: The 41c uses the same register to save XROM numbers as it does to save numerical responces to prompts. We know that Alpha responces to prompts are saved alright because the responce to PRP is correct, when printed.

There is also a second and more obvious answer to the second question. If programmable XROM functions were allowed to prompt then the postfix would be stored in programme memory after them. If, then, the postfix didn't happen to correspond to a standalone function, what would happen to programme integrity when the external ROM was unplugged, and the programme viewed? The programme would, obviously, not appear correctly to us. But, is this really a reason to give in responce to the question.

If Mother HP had really wanted she could have allowed programmable XROM's, and the Assembly Language programmer still can. If the postfix were to be stored in programme memory as an XROM number, then it would not cause a loss of programme integrity. All that is needed is a routine to interpret the XROM postfix as data, and a storage routine to place it in memory. This is not as hard as it sounds. Since a programmable XROM function will allow itself to be stored in programme memory, we have half the problem solved for us. All that remains is how to effect storage of the postfix. What is required here is a short, nonprogrammable function that prompts for the postfix. The responce to this prompt is then translated into the appropriate XROM and then stored in programme memory, just following the previously entered programmable XROM. The ØØ,XX XROM's might be appropriate for use here.

The function that we want to be programmable and prompting, will do its own prompting when executed from the keyboard, in run mode. So, if the routine tests to see if a programme is running, and then gets its data from programme memory following it automatically, then our 'problem' is solved; if a programme is not running then it knows that its data has already been gotten for it by the mainframe.

So, now we know why XROM's don't allow postfixes, and we also know how to solve the problem.

THE MILLER'S HIGH SPEED ALPHA PACKER — Gerard Westen (4780)

Not content with shortening routines (anyone would think he hated program instructions, the way he gets rid of them), Gerard now has reduced the execution time of alpha packers to around the three second mark. The listing of his latest remarkable routine appears on the opposite page. Previous packers were in the Journal last year, with the inventor, Jake Schwartz' original, taking 28 seconds odd (PPCJV8N3P15), in TN with around eight seconds, due to Richard Collett and .ED. (TN9p11), improved by Gerard again in TN (#11pp.67-70), and then Gerard's further reduction of Jake's original methods. Now this. Of course the best is Jim de Arras' microcode version, which takes no time at all. . .

For those who have not seen these routines before, know this: an alpha packer takes up to ten alpha characters from the alpha registers, allocates five bits to each, and codes them into an alpha string in X for storage in a single register. The unpacker does the opposite, taking the contents of X, placing the resulting string of ten or less characters into alpha. But whether this is your desire or not, study Gerard's programming to learn how a master does it. Your Alpha Packing - .ED.

* *

Labradacadian Euchre Player Waves Magic Wand — Chris Rath (6287)

Chris is a man of many parts (now THAT is a Sweeney Todd remark if there ever was one...), and card sharping (EUCHRE card sharping is a contradiction in terms??) seems to be one of them. (His feet, actually.) Here is his contribution, not so much to playing games (sinful) with the HP-41c, as to being assisted in the playing of games BY the HP-41c, which as everybody knows, is not at all sinful. More of Chris' work appears below, in this issue... .ED.

* * * * * * * * * * * * * * * * * * *

Dear John: *13/7/82*

I promised you the tic tac toe article over a year ago, and here it finally is. The XROM article should be of interest to some, if only from an idea point of view.

I have been meaning to drop you a line about the Bug Survey for several months. I have had about 30 responses; not enough to warrant any further follow up article, and I have returned the international postal coupons to those who sent them,and SAE.

I am sorry that I have caused a wastage of space in the TN's, it was not my intention to do so. I have access to an electric typewriter for the next few weeks, and I will try to get a few more submissions typed and sent off to you. If they are not suitable for the TN's maybe you could send them off to Richard with your normal batch of TN material.

If there is anyone in your chapter who has built an MLDL, you might let them know that I have written an Assembler. It takes ASCII from the keyboard and stores the opcode in memory. It calculates jump offsets for relative jumps and allows entry of 244 Hex Codes as well. It fits on 13 cards and requires REG to ROM, or a substitute for entry from the cards. Also one memory module, single density. The module is only used during entry of the programme to the ROME. Any interested in it can obtain a copy by sending 13 cards for me to record it on.

Happy Programming,

Chris Rath

WAND READS, BUT DOES NOT CHEAT AT EUCHRE

Chris Rath (6287)

Presented here is another WAND application. There is a game on the market that will play Bridge. The game uses bar codes on the cards to effect entry of the cards. The first of the routines here allows entry of a Euchre hand. It stores the hand in four registers, one for each suite. The bridge game inspired this.

By using the WNDSCN function, and the stick-on labels 2Ø, 25 to 29, 3Ø, 35 to 39, 4Ø, 45 to 49 and 5Ø, 55 to 59, we can effect an easy entry of the cards into memory. The ten's digit indicates which suite, and which register, and the one's digit indicates the specific card:

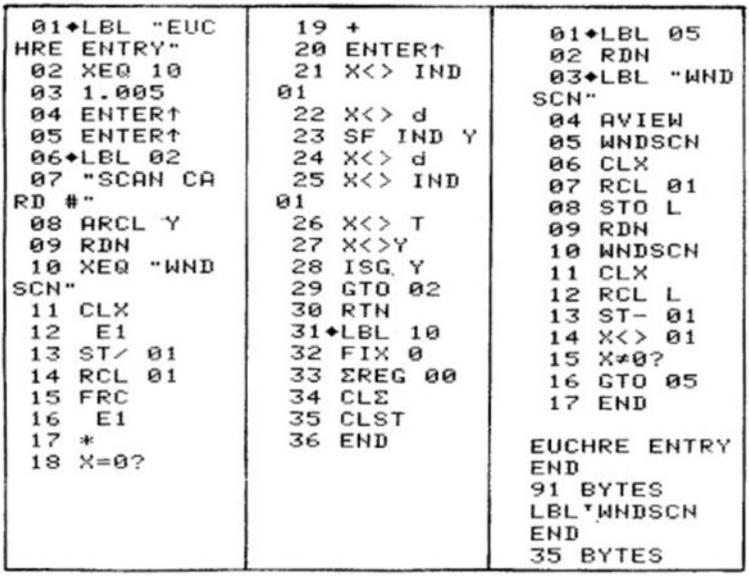
9H-25 1ØH-26 JH-27 QH-28 KH-29 AH-2Ø etc.

Using the indirect register exchange, and the exchange 'd', we can put the appropriate register (suite) into the flag register, and then set the flag corresponding to the scanned card. Of course a digit Ø must indicate flag 1Ø, or an indication would show on the display. Two extra flags must be used to indicate the Bauers, and we may as well arbitrarily pick 12 and 13.

The labels should be trimmed before they are stuck on the cards to remove the printed 'identifier' that tells what number the card (barcode) is. The label can be more than adequately protected by placing Scotch Brand Magic Tape over the label. The tape has a matte finish, and does not hinder SCAN'ing of the code.

Because there is no checksum in this type of barcode, it is imperative that the 41c check to see that it has read the card correctly. This is the function of the LBL "WNDSCN" routine. Only the X and L registers are used to effect the check, the stack is otherwise just as the scan left it. The routine is entered at LBL "WNDSCN", and is really a multipurpose routine, and should have a global label on it.

2EØ, Chris Rath (6287)
1281 Agincourt Road, Ottawa, Ontario, Canada, K2C 2J3



```
 01◆LBL "EUC
HRE ENTRY"
 02 XEQ 10
 03 1.005
 04 ENTER↑
 05 ENTER↑
 06◆LBL 02
 07 "SCAN CA
RD #"
 08 ARCL Y
 09 RDN
 10 XEQ "WND
SCN"
 11 CLX
 12 E1
 13 ST/ 01
 14 RCL 01
 15 FRC
 16 E1
 17 *
 18 X=0?
```

```
 19 +
 20 ENTER↑
 21 X<> IND
01
 22 X<> d
 23 SF IND Y
 24 X<> d
 25 X<> IND
01
 26 X<> T
 27 X<>Y
 28 ISG Y
 29 GTO 02
 30 RTN
 31◆LBL 10
 32 FIX 0
 33 ΣREG 00
 34 CLΣ
 35 CLST
 36 END
```

```
 01◆LBL 05
 02 RDN
 03◆LBL "WND
SCN"
 04 AVIEW
 05 WNDSCN
 06 CLX
 07 RCL 01
 08 STO L
 09 RDN
 10 WNDSCN
 11 CLX
 12 RCL L
 13 ST- 01
 14 X<> 01
 15 X≠0?
 16 GTO 05
 17 END

EUCHRE ENTRY
END
91 BYTES
LBL'WNDSCN
END
35 BYTES
```

ASSEMBLY LANGUAGE PROGRAMMING HINTS — *Chris Rath (6287)*

Ah well, everybody knows what Chris really means by the above title. It's MALMAC, or its MICROCODE, or its MACHINE LANGUAGE, or its MACHINE CODE. Whatever- it is what the HP-41c microprocessor gobbles up, and it is also what there hasn't been enough of around the shop. It will be seen from the following that Chris is ahead of all bar a few in the area. The only sad thing about all of this, is that it all could have happened a year earlier, had HP allowed a peek, if not a poke inside some of those Books and Manuals. It comes as a real shock to realise that the codes/mnemonics for well known microprocessors were way out in the public domain almost before the positive holes started to flow through the chips....

One must remember that many Class 2 instructions, refering to A=A+B thru C=-C-1, will affect the state of the 'carry' bit. So, one must be extra careful if the XQ/GO & relative jumps or conditional returns are placed after one of these Class 2 instructions. For example; in the 41c mainframe this feature is used to test for the ON/OFF key. One of the first things that occurs when the 41 detects a keystroke is the addition of 8 to the 'row' nybble of the keycode. All the keys have row numbers between ∅and 8; but ON is the only key in row 8. Therefore, when 8 is added to a row value of 8,then a carry results and the machine knows to branch to the OFF routine. Don't say to yourself, as you place a "?c XQ " after "C=C+1 M",that you don't have to worry because there will never be a carry; for if at some time there is garbage in the register, the results could be more than you bargained for. I had a routine overwrite 4K of my MLDL; causing 2K of code to be lost!

Don't get caught up using only the "LDI S&X" or only the "LD@R- d". They both have their advantages, and it only takes a minute to write out both and count the bytes. "LD@R" requires that the R value be set, an extra byte; but, it decrements the pointer after loading, and the digit is loaded exactly where you want it. "LDI" on the other hand, does not require a pointer value, and can load up to three digits at a time (in some cases); but it often requires that an RCR be used to shift the data string into position before the load, and sometimes again after too. It must also be remembered that LDI clears bits 2&3 of digit 2; even though no data is there to be loaded. (This implies that it may be possible to load 12 bits with a LDI; if ROME were 12 bits wide.) 3 digit loads may require a bit shift after loading.

Don't worry too much about destroying the return stack; except your own pending returns (I am refering to the cpu stack). If you do it inadvertantly the machine will just go into Standby mode; and if you do it intentionally you can always use a GOTO ADR to end your routine and send execution back to the mainframe.

Because no bit level shifts are provided, the easiest way to perform bit shifts is to remember that RCR one bit,is just a C=C+C. The appropriate combination of C=C+C's and RCR's will fill one's needs. Remember that you are shifting zeros in from the right when you perform a C=C+C; regardless of the field! This hint seems trivial, but it took me two days to figure out- I hadn't been thinking.

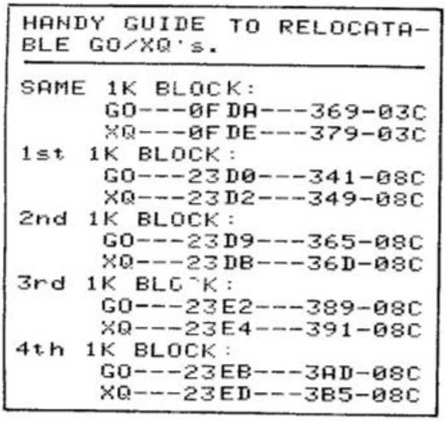
2E∅,
Chris Rath (6287)
1281 Agincourt Rd., Ottawa, Ont., Canada, K2C 2J3

TWO TEN BITS OF WORDS TO THE WISE FROM YR VERY OWN CHRIS RATH (6287)

When you are using ?c XQ and ?nc XQ be careful that you do not, unintentionally, send execution to an address containing Hex ØØØ. If the first word an execute encounters is ØØØ then the cpu treats it as a RTN; and so returns immediately.

This is the method that the 41c firmware uses to check for Printer existence,and the Diagnostic module. The 41c just makes a call to the printer/diagnostic routine; and if the module/plug is in place then execution of the routine is effected. Otherwise there is an immediate return, and the 41c continues its normal execution path.

Chris Rath (6287)



```
HANDY GUIDE TO RELOCATA-
BLE GO/XQ's.

SAME 1K BLOCK:
      GO---0FDA---369-03C
      XQ---0FDE---379-03C
1st 1K BLOCK:
      GO---23D0---341-08C
      XQ---23D2---349-08C
2nd 1K BLOCK:
      GO---23D9---365-08C
      XQ---23DB---36D-08C
3rd 1K BLC¯K:
      GO---23E2---389-08C
      XQ---23E4---391-08C
4th 1K BLOCK:
      GO---23EB---3AD-08C
      XQ---23ED---3B5-08C
```

JEREMY SMITH
19451 Mesa Drive
Villa Park
California 92667
USA
29 October, 1982

Dear Chapter Co-ordinator

Please find enclosed a complimentary copy of the new 'HP-41 SYNTHETIC Quick Reference Guide' and cover letter.I would be most grateful if you would present this booklet at your next meeting and allow all present a chance to see it. The October issue of PPC CJ V9N7 should have this mentioned in the trading post column. The back cover of the booklet shows the contents for a quick review.

Each order consists of the booklet and the cover letter. Each order is available for $5.00 and either S.A.S.E. or $1.00 postage. PPC members: $4.00 plus postage. All orders and enquiries to Jeremy Smith #6676, at the above address.

Outside USA add $2 for post & handling

PPCTN COMES OF AGE

Chris Rath has been busy for us, not only in playing cards on our behalf, nor yet on writing and tipping on THAT many-named stuff, but he has performed the ultimate thankless task: we have from him an index for TN1 to 12. As it is already in small print format, we will run off a separate small booklet, and enclose it with TN#14. (Yes folks! There is to be such a beast.) Here is Chris on the subject. Our thanks to him. Preparing such an index is a long task, but important in our area. Even though I sometimes feel I know most of most issues of TN by heart, at least for several weeks after final copy is ready, I often have to hunt through the contents on the covers for a while to find what I need. For others it is harder.

24/9/82

Dear John:

I had some computer time left after this months Assignments were handed in. So, I used the Editor to do this listing.

There are several entries missing, as follows:

1) The Editorials entries for TN#1 and #2.

2) From TN#9 on entries of the type:(and I give as example) the block of contents listings that runs from "Visitors to Melbourne......." to "bar code, 51." in TN#9. These 'blocks' of entries take a long time to enter. If I can manage to get this saved on tape then I will update it from time to time, and send you listings. If not, then this will be the only listing. This file takes ALL of my temporary disk storage allotment, and I will have to delete it, or save it on tape, before I run my next assignment. I don't know, yet, whether or not I have tape privileges; I will check Monday.

If I get a chance I will add the missing entries and send a listing of the update. In any case, I hope that I have saved someone the work of doing this.

While I was typing this in I noticed that Gerard Westen had already written on RAM RTN's (n8p11). It makes my submission to Richard (I sent you a copy) on the subject rather redundant. But, I can't find a CJ item about the subject. (Is it there??).

Enjoying School,
Chris Rath
(6260)