HP Forum Archive 14

[Return to Index | Top of Index]

A little HP-15C Quiz for the weekend

*Message #1 Posted by **Valentin Albillo** on 20 Feb 2004, 5:09 a.m.*

Hi all:

Whether you consider yourself an HP-15C expert or not, here's a little HP-15C quiz you may want to try your hand at. It goes like this:

The Quiz

The HP-15C uses a numerical, mostly positional, merged keycode system to identify instructions stored in program memory. Thus we have keycodes such as:

```
  SIN     ->        23
g ABS     ->     43 16
f LBL A   ->  42,21,11
```

Of course, keycodes are generated and displayed only for *programmable* instructions (thus GTO CHS 123 generates no keycode at all, since it's not programmable). Taking all this into account, try and answer the following questions:

1. Just how many **different** keycodes can be generated in an HP-15C ? 100 ? 200 ? More ? Exactly how many ?

   For instance, f LBL A (42,21,11) and f LBL B (42,21,12) are two such standard, different keycodes (instructions). We're only interested in normally generable, actually keyable, standard keycodes. There are some weird techniques to place non-standard, 'synthetic' keycodes in program memory but we won't consider them here.

2. Would all of them fit into program memory at once ? If not, what's the maximum number of *different* keycodes (instructions) that can be recorded in program memory *at the same time* ?

3. How much RAM would be needed to store them all at once ?

4. In the HP-15C, every programmable instruction uses up one or two bytes of RAM. If every combination could be 'synthesized' in memory, how many different 'instructions' would be possible to generate and store as program lines ? How much RAM would they require to store them all simultaneously ? Would 32 Kb be

enough ?

5. If you succeed at this, you're a worthy HP-15C guru and thus you're up to greater challenges. Try this one to begin with: starting from a just master-cleared machine, i.e:

```
[OFF]
[ON]+[-]  ->  Pr Error
[any key] ->  0.0000
```

try and reproduce the following display with a *minimum number of operations* (e.g. STO+5 is *one* operation, but *three* keystrokes):

There, that should keep you HP-busy this weekend. Solutions, next Monday.

Best regards from V.

Re: A little HP-15C Quiz for the weekend

*Message #2 Posted by **Eric Smith** on 20 Feb 2004, 12:05 p.m.,*
in response to message #1 by Valentin Albillo

In number 4, do you want both "legal" (keyable) and "illegal" instructions? I'm guessing that you do, because otherwise I don't see how the answer to #4 is different than #1 and #3.

If you do want to include "illegal" instructions, I think the answers to #4 are 4336 instructions and 8432 bytes. Or it might be 4080 instructions and 7920 bytes; I don't how the 15C parses one particular bit pattern of illegal instructions, so I'm not sure if they would count.

I discovered some of the 15C synthetic techniques and most of the instruction codings in 1982 in calculus class when I should have been studying, which is probably why I don't know calculus nearly as well as I should. Someone else also discovered this stuff, and unlike me actually published their findings.

I've got answers to the other questions, but I'd better not post them just yet. However, I'll note that the answer to #5 does need only operations listed in the 15C owner's manual.

Re: A little HP-15C Quiz for the weekend

*Message #3 Posted by **Patrick** on 21 Feb 2004, 2:54 a.m.,*
in response to message #1 by Valentin Albillo

But, Valentin, I can't generate that display with ANY number of keystrokes... the HP-15C I have doesn't have a logo :-(

Re: A little HP-15C Quiz for the weekend

*Message #4 Posted by **Valentin Albillo** on 22 Feb 2004, 12:33 a.m.,*
in response to message #3 by Patrick

Posted by Patrick : *"But, Valentin, I can't generate that display with ANY number of keystrokes... the HP-15C I have doesn't have a logo :-("*

All *six* of them !? :-/

Anyway, it'll be enough if you manage to reproduce the black symbols within that round-cornered, depressed zone. I mean, the sizeable black thingies, not the dust specs.

Best regards from V.

My answers: HP-15C Quiz

*Message #5 Posted by **Karl Schneider** on 21 Feb 2004, 7:03 p.m.,*
in response to message #1 by Valentin Albillo

Valentin --

I'd thought about this before, but never analyzed it.

The information needed to takle this problem is found in the HP-15C Owhner's Handbook, "Function Summary and Index" (pp. 272-280, blue-edged), and "Two-byte Program Instructions" on p. 218.

I counted the following:

```
1-byte operators: 81
2-byte operators: 28
(Overlapped): 8 [STO, RCL, x<>, LBL, GTO, GSB, DSE, ISG]


1-byte complete instructions: 228
2-byte complete instructions: 455
```

There are 2^8 = 256 unique values in an 8-bit byte.

If my numbers are correct, it seems significant that there are 228 1-byte complete instructions (each presumably requiring a unique code) and 28 2-byte operators (which require some identifier in the second byte to complete). If a unique 1-byte "operation code" is established for each 2-byte instruction,

228 + 28 = 256, which is exactly how many 8-bit codes are available. To achieve this may explain why eight of the operators listed above can be 1- or 2-byte instructions, depending on the identifier used.

BTW, this exercise indicates an omission on p. 218 of the HP-15C Owner's Handbook -- "GSB . *label*" is also a 2-byte instruction.

So, to answer your questions 1-3:

1. 228 + 455 = 683 programmable instructions
2. These could not fit into the 64 x 7 = 448 bytes of programmable RAM. 228 + (448-228)/2 = 338 unique instructions could be stored.
3. (228 x 1) + (455 x 2) = 1138 bytes would be required to store all every possible instruction.
4. To answer this one would require knowledge, not speculation, of the instruction-coding method.
5. Is there some "byte-grabber" keystroke sequence to do this?

That's how I see it. I'll await your answers,

-- Karl

Re: A little HP-15C Quiz for the weekend

*Message #6 Posted by **Andrés C. Rodríguez (Argentina)** on 21 Feb 2004, 7:47 p.m.,*
in response to message #1 by Valentin Albillo

V:

Thank you very much for posting these kind of challenges, and so well presented. While I have almost no HP15C experience (I have used one briefly, many years ago and, frankly, I didn't liked it as much as the 25, 41, and 42), I know it certainly is a valuable jewel from the good old times. So I pass for the first questions, most interesting ones related to opcode space allocation and memory usage.

I have a possible answer for #5 but, not to spoil your challenge, would not state my unproven answer, but I just suggest that ...

Such display:

... could be obtained using very common keys in a particular manner,

... will give the owner some peace of mind,

... will hardly appear in a software-emulated 15C calculator,

... does not come from synthetics, byte jumpers or similar artifacts.

See you on Monday, and thanks again!

I haven't got solutions...

*Message #7 Posted by **Tizedes Csaba** on 23 Feb 2004, 5:41 p.m.,*
in response to message #1 by Valentin Albillo

It's not a big thing... I found just this... I hope it will give some idea for somebody.

Csaba

Ps.: I'm sorry, Valentin, I haven't got enough time in this weeks...

HP-15C Quiz: My Solutions

*Message #8 Posted by **Valentin Albillo** on 24 Feb 2004, 5:21 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

Thanks to those of you who were interested in my little 15C Quiz and posted some inputs. Regrettably, and despite my initial belief that many of this Forum's regular and not so regular contributors were in fact proud HP-15C owners and would happily take to the task, it seems that I was being a little overoptimistic. Sorry, perhaps next time I'll come up with a more enthralling challenge ... :-(

However, for the time being and for what is worth, here you are, my own solutions to the questions raised, namely:

1. **Just how many *different* keycodes can be generated in an HP-15C ?**

   There are exactly **700** different, normally keyable, programmable instructions (hence keycodes), as well as nearly 500 different non-programmable ones, not to mention an undefined number of possible 'synthetic' ones.

2. **Would all of them fit into program memory at once ? If not, what's the maximum number of different keycodes (instructions) that can be recorded in program memory at the same time ?**

   Said 700 different instructions do **not** fit in the maximum 448 bytes allocatable for program memory. In that much RAM you can only fit a maximum of **338** different instructions, as Karl Schneider correctly figured out.

3. **How much RAM would be needed to store them all at once ?**

   A total of **1,150 bytes** would be required to store all 700 different instructions at once.

4. **... if every combination could be 'synthesized' in memory, how many different 'instructions' would be possible to generate and store as program lines ? How much RAM would they require to store them all simultaneously ? Would 32 Kb be enough ?**

   There are 228 one-byte instructions, plus 28 two-byte prefixes. Adding an arbitrary byte to every one of those 28 prefixes would give a grand total of 28*256 + 28 = **7,168** different instructions possible. They would require **14,364 bytes** so **yes**, 32 Kb would be enough.

5. **Try and reproduce the following display with a minimum number of operations**:

   

   Several of you gave very clear 'hints' on how to attack this particular question, but none gave a solution. I won't do it, either, for the moment being.

   Let's see if those of you who boldly said *"I've got answers to the other questions, but I'd better not post them just yet"* can deliver the goods, or it was just a case of Fermat-like *"I've got a truly wonderful solution but this Forum is only too small to contain it"* :-) Eric ? Are you listening ? :-)

Thanks again and best regards from V.

Re: HP-15C Quiz: My Solutions

*Message #9 Posted by **Eric Smith** on 24 Feb 2004, 2:23 p.m.,*
in response to message #8 by Valentin Albillo

> Quote:
>
> Eric ? Are you listening ?

I don't have my 15C handy, so I can't try this to verify that my memory is correct, but I believe my usual procedure for producing that synthetic matrix descriptor required six "operations":

1, 0, 1, 1, ON-D, ON-D

Depending on the fourth digit, you get various synthetic matrix descriptors, one of which is the one you've photographed.

The ON-D operation is documented in the manual as being used to reset the calculator, and it is stated that the X register contents are altered and that you should clear the X register afterward. The details are not given, but in practice it rotates the X register right by 22 bits. So the method I've given winds up with the third and fourth digits, "11", in the mantissa sign and leading mantissa digit.

Matrix descriptors have a 1 in the sign digit, and the specific matrix is chosen in the leading mantissa digit. For normal matrices, that digit must be hexadecimal A through E. The other 11 values are synthetic matrices, and some of those display with strange symbols. "11" is particularly useful for synthetic programming, as the resulting matrix points to the base of the free registers. It can be used to indirectly access program memory and the internal ("status") registers.

It is possible that there is a shorter sequence to get the "11" matrix descriptor. I did not spend much time searching for shorter key sequences that would result in the right bit pattern to be shifted once or twice with ON-D.

> Quote:
>
> There are 228 one-byte instructions, plus 28 two-byte prefixes.

I believe you are mistaken. My investigations found that there are 240 one-byte instructions, and 16 possible two-byte prefixes.

But even if you were correct, your math would be wrong. If there were 28 two-byte prefixes, and an arbitrary byte could be used for the suffix, the total possible combinations would be 228 + 28 * 256, which is 7396, requiring 14564 bytes to store all combinations.

However, any leading byte with the low four bits all ones (i.e., xF hexadecimal) is a two-byte prefix, and all others are single-byte. Of the 16 possible two-byte prefixes, only the six with leading digits A through F (hex) are used for legal instructions,

If an arbitrary byte was used as the suffix, this would allow 4336 instruction codes: 240 single-byte and 4096 double-byte. It would require 8432 bytes of program memory to store all of those.

However, HP did not ever use a second byte with the low four bits all one, because that would confuse the BST code. Therefore, I don't consider those to be possible instruction codes. That leaves 4080 possibilities, 240 single-byte and 3840 double-byte. It would take 7920 bytes of memory to store all of those.

Either way, 32K of program memory would be more than sufficient.

Edited: 24 Feb 2004, 7:37 p.m. after one or more responses were posted

Re: HP-15C Quiz: My Solutions

*Message #10 Posted by **Eric Smith** on 24 Feb 2004, 7:33 p.m.,*
in response to message #9 by Eric Smith

I still don't have my 15C handy, but I think I can get it down to four "operations":

9, 1/x, ON-D, ON-D

I think three "operations" would be the theoretical minimum, but I haven't yet found a sequence that short.

Edited: 24 Feb 2004, 7:35 p.m.

Re: HP-15C Quiz: My Solutions

*Message #11 Posted by **Valentin Albillo** on 25 Feb 2004, 6:09 a.m.,*
in response to message #9 by Eric Smith

Hi, Eric:

Eric posted: *"I believe you are mistaken. My investigations found that there are 240 one-byte instructions, and 16 possible two-byte prefixes."*

Well, the HP-15C Owner's Handbook includes a list with all operations requiring two-bytes, and they certainly belong to 27 distinct prefixes listed (LBL, GTO, SF, STO+, MATRIX, etc, for certain ranges of arguments) plus 1 additional unlisted prefix (GSB). So, they make 28 in total, unless that whole section of the Handbook is completely wrong, which I don't believe as I've tried all instructions and ranges there and they certainly use up two bytes of RAM ...

... unless there's some internal mechanism to map said 28 prefixes into only 16 internal two-byte prefixes, which is a definite possibility, as none of those prefixes has as many as 256 possible arguments. Some clever internal programming could use one of them for several separate instructions, being interpreted as one or the other depending on the argument (following byte). It might be ...

*"But even if you were correct, your math would be wrong. If there were 28 two-byte prefixes, and an arbitrary byte could be used for the suffix, the total possible combinations would be 228 + 28 * 256, which is 7396, requiring 14564 bytes to store all combinations."*

Yes, you're right. I had the concept absolutely clear, but the similarity of 28 and 228 made me incur in that stupid typo. The correct calculation is indeed 28*256+228, not +28.

"I still don't have my 15C handy, but I think I can get it down to four "operations": 9, 1/x, ON-D, ON-D I think three "operations" would be the theoretical minimum, but I haven't yet found a sequence that short."

The theoretical minimum would be two: place some suitable number in the display with a single operation, then perform an ON-D operation, for a total of two operations. Assuming entering individual digits is considered one operation per digit, decimal point, change sign, or exponent entered, it seems fairly difficult to do it in only two operations.

However, starting from a master-cleared machine as stated in the original quiz, there's a solution in just *three* operations. And there's an equally short solution to generate a similar display, only with a lonesome lowercase "y" at the left end of the display instead :-)

Thanks for your clever considerations and for your interest in my little quiz and

Best regards from V.

Re: HP-15C Quiz: My Solutions

*Message #12 Posted by **Eric Smith** on 25 Feb 2004, 11:58 a.m.,*
in response to message #11 by Valentin Albillo

> Quote:
>
> unless there's some internal mechanism to map said 28 prefixes into only 16 internal two-byte prefixes,

The number of prefix keys/operations has very little to do with how the instructions are encoded.

All twenty-eight keystroke prefixes are mapped into only **five** prefix bytes out of the sixteen posible. Except that some key prefixes need a second byte for some suffixes and not others.

> Quote:
>
> as I've tried all instructions and ranges there and they certainly use up two bytes of RAM ...

You didn't keep track carefully, then.

Many prefixed key sequences are actually mapped to a single byte. For instance, "GTO 6" is the single byte 16 (hex), while "GTO .6" is FF 16. But you shouldn't conclude that a "." in the sequence automatically adds a prefix; "RCL .6" is the single byte 56. But user-mode "RCL C" is two bytes: BF 3C.

Each of the sixteen possible prefixes has 240 possible suffixes. The suffix can be any byte not of the form xF. The the and BF prefixes in my examples above are used for a lot more operations than just "GTO .x" and user mode RCL.

> Quote:
>
> The theoretical minimum would be two: place some suitable number in the display with a single operation, then perform an ON-D operation, for a total of two operations.

The way you defined "operation" originally led me to believe that entering a number with multiple keystrokes would be multiple operations, because it would be equivalent to multiple program steps. In other words, I thought that entering the number 127.5 was five operations.

Yes, it's trivial to solve in two operations if you can enter an arbitrary (normalized) floating point constant as a single operation. A trivial example is 1.000000044 ON-D. However, entering a suitable number takes many keystrokes.

The shortest I've found using a single floating-point number and ON-D is eight keystrokes, if ON-D as counted as one. The second sequence I've posted in this thread takes only four keystrokes.

Re: HP-15C Quiz: My Solutions

*Message #13 Posted by **Valentin Albillo** on 25 Feb 2004, 1:14 p.m.,*
in response to message #12 by Eric Smith

Hi again, Eric:

Eric posted: *"The way you defined "operation" originally led me to believe that entering a number with multiple keystrokes would be multiple operations, because it would be equivalent to multiple program steps. In other words, I thought that entering the number 127.5 was five operations."*

Absolutely correct, you understood it alright.

"Yes, it's trivial to solve in two operations if you can enter an arbitrary (normalized) floating point constant as a single operation."

As that's not the case, solving it in just two operations is far from trivial. Matter of fact, I believe it to be impossible. A three-operation solution, however, is perfectly possible within the original requirements of the quiz.

BTW, have you published your HP-15C internal coding findings somewhere accessible on-line, such as a web site of yours ? I would be interested to have a look, if you don't mind.

Thanks and best regards from V.

Re: HP-15C Quiz: My Solutions

*Message #14 Posted by **Eric Smith** on 25 Feb 2004, 6:06 p.m.,*
in response to message #13 by Valentin Albillo

Quote:

> Absolutely correct, you understood it alright.

Then why, when I said that I thought the theoretical minimum was three operations, did you "correct" me and say that it was two? There is not a single operation among the 700+ available (700 programmable plus some non-programmable), which from a cold start will leave a suitable number in x. The requirements are that the 8th mantissa digit be 0, 4, or 8, the 9th digit be 4, and the 10th digit be 4, 5, 6, or 7.

I suspect that there is a single digit entry (or pi) followed by a trig, inverse trig, log, exponential, or square root function that will produce a suitable number, but I haven't yet found it.

> Quote:
>
> BTW, have you published your HP-15C internal coding findings somewhere accessible on-line, such as a web site of yours ?

No. I only have some handwritten notes. My handwriting wasn't good to begin with, and the ink has faded, so I can't read much of it. I think there was an article in the PPC Journal that had more information than I'd figured out.

> Quote:
>
> And there's an equally short solution to generate a similar display, only with a lonesome lowercase "y" at the left end of the display instead

There are three-operation solutions for most of the synthetic matrices. I've found a bunch of them. I just haven't yet found a three-operation solution for the problem you posed, which is rather irritating since that is the most useful of the synthetic matricies.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall