Curious?

Qubero

I've been experimenting with hexadecimal notations for Qubero (binary editor). I'm trying to find a robust, clean way to represent base-16 that makes it look different from decimal, for a user-friendly full featured graphical hex editor. It needs to be a way to draw hex digits so any one can see they're different to ordinary decimal numbers, and users with any level of experience will not encounter too much confusion.

This first page looks at single lines or small segments of hex that may need to be explicitly identified as radix-16. I'm not a fan of traditional programming notations which use a **`0x`** prefix, or an **`h`** postfix (eg **`0x3F`** or **`3Fh`**). These notations are confusing, and break the rule of using both start and end tags for easier parsing by both humans and computers. They also become very difficult to read when used with more than a few bytes of hex.

NOTATIONS FOR SMALL SEGMENTS OF HEXADECIMAL:

$\underline{51\ 75\ 62\ 65\ 72\ 6f}_{16}$

$0x[5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F]$

$0_0\text{–}F_F{:}[\ 5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F\ ]$

$0_0 0_0{:}\ 5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F\ :$

$\text{hex}(5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F)$

$[5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F]\ \text{radix}_{16}$

$\#\ 5_1\ 7_5\ 6_2\ 6_5\ 7_2\ 6_F\ h$

The second page looks at larger chunks of hex, and how the up-down notation looks in aggregate. I've deliberately avoided using actual subscripts, and instead used lowered full-sized characters. This is to avoid confusion with (what are meant to be) unrelated notations that might use post-subscripts. Unrelated notations include: doing something like 1037_8 which is used to indicate the number (1037) is in octal (radix-8), and probably some chemistry notation. My up-down notation is simply to differentiate higher and lower order nibbles, in what is essentially base-256.

ARIAL BLACK BYTES:

5_1	7_5	6_2	6_5	7_2	4_F
A_1	B_8	0_0	C_5	D_2	2_F
5_8	7_5	F_2	A_A	7_2	1_1
D_E	A_D	C_A	F_E	B_A	B_E
0_0	0_1	2_3	4_5	6_7	8_9

ARIAL BLACK BYTES, "SMALL CAPS":

51	75	62	65	72	4F
A1	B8	00	C5	D2	2F
58	75	F2	AA	72	11
DE	AD	CA	FE	BA	BE
00	01	23	45	67	89

WOOPASS:

F_F	A_5	6_2	B_3	3_F	0_F
C_1	7_5	0_9	6_E	A_2	6_F
5_1	F_5	1_1	1_2	1_3	0_8

PLAIN ESSAY:

51	75	62	65	72	4f	51	75	62	65	72	4f
a1	b8	00	c5	d2	2f	a1	b8	00	c5	d2	2f
58	75	f2	aa	72	1f	58	75	f2	aa	72	1f

This final page looks at ways of representing 4-byte chunks of data which have been identified as 32-bit units with a byte order. Note that Intel processors are known for having a byte order that gives greatest weight to the right-most byte, making it the opposite to what you'd expect if you're familiar with the normal decimal system which has thousands are on the left, and "ones" on the right. Big-Endian processors don't do this bizarre byte-reversal trick, and many modern processors can handle either format equally.

I've looked at ways to represent both byte-order systems in intuitive ways, extending the up-down notation to cover 4-bytes (or 8 hex-digits) at a time. It's not hard to imagine an extension to other word sizes.

INT32-BIGENDIAN:

$\text{5A13}_{\text{84DF}}$ $\text{55CF}_{\text{53B2}}$ $\text{FE42}_{\text{80DF}}$ $\text{FA4F}_{\text{9313}}$

$\text{0512}_{\text{80DF}}$ $\text{2A1F}_{\text{5342}}$ $\text{0512}_{\text{80DF}}$ $\text{7A1F}_{\text{5342}}$

$\text{5A13}_{\text{84DF}}$ $\text{55CF}_{\text{53B2}}$ $\text{FE42}_{\text{80DF}}$ $\text{FA4F}_{\text{9313}}$

$\text{0512}_{\text{80DF}}$ $\text{2A1F}_{\text{5342}}$ $\text{0512}_{\text{80DF}}$ $\text{7A1F}_{\text{5342}}$

INT32-INTEL:

$\text{0512}^{\text{80DF}}$ $\text{5342}^{\text{7A1F}}$

INT32-INTEL, *OVER-EXAGGERATE:*

$\underline{\text{05}}\underline{\text{12}}\underline{\text{80}}\underline{\text{DF}}$ $\underline{\text{53}}\underline{\text{42}}\underline{\text{7A}}\underline{\text{1F}}$

INT32-INTEL, SMALL CAPS VARIANTS:

051280DF 83A2A037 (1)

53427A1F 72A20AF3 (2)

53427A1F (3)

53427A1F (4)