Subject: Extending C64 BASIC
Posted by lawless.jim on Mon, 17 Mar 2014 01:24:09 GMT
View Forum Message <> Reply to Message

After recent discussions about extending BASIC, I tinkered a bit and came up with a simple ML kernel that will allow one to add single-letter command extensions to C64 BASIC.

Please see my write-up here:

http://lawlessguy.wordpress.com/2014/03/16/extending-commodo re-64-basic/

This is simply the starting-point. I wanted to get it out there to make it easier for other experimenters could start working on their own extensions.

Subject: Re: Extending C64 BASIC
Posted by winston19842005@yahoo on Mon, 17 Mar 2014 06:11:58 GMT
View Forum Message <> Reply to Message

On Sunday, March 16, 2014 9:24:09 PM UTC-4, lawle...@gmail.com wrote:
> After recent discussions about extending BASIC, I tinkered a bit and came up with a simple ML kernel that will allow one to add single-letter command extensions to C64 BASIC.
>
>
>
> Please see my write-up here:
>
>
>
> http://lawlessguy.wordpress.com/2014/03/16/extending-commodo re-64-basic/
>
>
>
> This is simply the starting-point. I wanted to get it out there to make it easier for other experimenters could start working on their own extensions.

Interesting. My old TI-99 had some nice linking abilities built in. The TI-99/4 had CALL statements which, in themselves, were extensions to the language... and in Extended Basic, you could write BASIC subprograms to be called by the language by name (with local variables). In GPL (Graphics Programming Language, which BASIC was written in) you could add subprograms... this is where all the usual BASIC CALLs are stored, such as CALL CLEAR, CALL SOUND, etc.

In assembly, which required either the Editor Assembler module or the TI Mini Memory, or Extended Basic modules (the former to write and execute, the last to merely execute or poke (CALL LOAD) programs into memory), there were linkage facilities.

CALL INIT - initialize memory for assembly use, loaded some pre-defined routines into memory, setup pointers, etc.

CALL LOAD - Usually just a POKE routine (ex: CALL LOAD(-31871,256,20) would poke 256 & 20 starting at CPU location -31871), but if you passed it a device.filename, it would load assembly code into memory.
CALL PEEK - the standard PEEK utility
CALL LINK - Given the entry point of an assembly routine (a name, up to 6 characters in length), control is passed to that routine. Up to 16 parameters, string or numeric, could be passed to the program (I believe that the cartridge used might vary the number of parameters...)

The Mini Memory added a few CALLs, including...
CALL POKEV, PEEKV (poke and peek directly to the video memory inside the TMS9918(A)

(Sources besides my old & feeble memory:
http://www.mainbyte.com/ti99/software/s_carts/mini.html)

Subject: Re: Extending C64 BASIC
Posted by Anton Treuenfels on Mon, 17 Mar 2014 13:54:21 GMT
View Forum Message <> Reply to Message

<lawless.jim@gmail.com> wrote in message
news:2fa1d2e8-0d17-4e93-89db-c7b24cf4044e@googlegroups.com...
> After recent discussions about extending BASIC, I tinkered a bit and came
> up with a simple ML kernel that will allow one to add single-letter
> command extensions to C64 BASIC.
>
> Please see my write-up here:
>
> http://lawlessguy.wordpress.com/2014/03/16/extending-commodo re-64-basic/
>
> This is simply the starting-point. I wanted to get it out there to make
> it easier for other experimenters could start working on their own
> extensions.

Before that, you might consider changing this:

```
dispatch = *
jsr CHRGET
cmp #'a'
bcs contin1
jmp SNERR
contin1 = *
cmp #'z'
bcc contin2
jmp SNERR
contin2 =  *
sec
sbc #'a'
cmp #'z'
```

```
asl
tax
lda table+1,x
pha
lda table,x
pha
jmp CHRGET
```

to this:

```
dispatch = *
jsr CHRGET
cmp #'z'+1
bcs badcmd
sbc #'a'-1
bcc badcmd
asl
tax
lda table+1,x
pha
lda table,x
pha
jmp CHRGET

badcmd = *
jmp SNERR
```

Also, because error detection is one of the things that I fetishize, I'd write "@c" like this:

```
; syntax;  @c
;clear the screen

do_cls = *
bne badcmd
lda #147
jmp $ffd2
```

Now if "@c" will complain if followed by anything but end-of-line or ":".

- Anton Treuenfels
