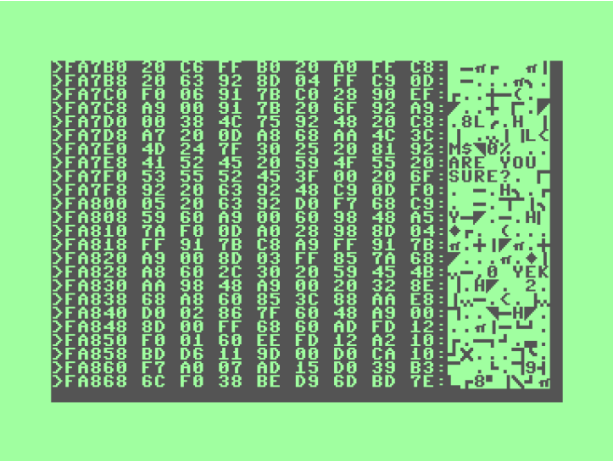
Commodore C128

Built-in Machine Language Monitor

By lived.ch (C)2024, October, version 1.0



The built-in ML Monitor of Commodore C128 is offering all you need to modify every single byte and bit and also every BANK and so within seconds copy the preset character sets from BANK 14 {ROM} to any other BANK {RAM} and modify the character the way you want to have them.

You could even quickly add your own font creation to be default and use it while programming.

Furthermore, you can directly write an assembly code or a subroutine and use it in connection with BASIC 7.0 or any other programming language.

With ease you can fill the area with necessary values using **F** command or transfer the existing code to another area with **T** command . Try not to mix these two commands or it's game over very quickly, in any case faster than you can blink!

Save and load files isn't a mystery.

You can quickly make tables to use with your code. e.g. you could make a table with x,y, positions for your sprites and then just address the location and get the data.

So many things are possible, but let us start with the Commands ML Monitor is offering.

Info:
I tried to make this well understandable, even for beginners {like I am too, since I started again in November 2023 with **Commodore C128** after close to 4 decades of absence}

Commands for Commodore C128 built-in ML Monitor

Command	Example	Info
A	**A 0b00 lda #$00**	start writing an assembly code { . can also be used }
C	**C 0b00 0b0a 0c00**	{start, end, compare start address}
D	**D 0b00 0b0a**	disassembling the code {shows 21 bytes or up to 19 lines}
F	**F 0400 053f 51**	fills the 1st 8 screen lines with balls
G	**G 0b00**	execute the code at $0b00 {use it with precaution, can freeze}
H	**H 0b00 0b0c 9d 00 04**	will find "sta $0400,x", $0b04 { . 0b04 **9d 00 04** sta $0400,x }
H	**H 1000 2000 'HELLO**	will find "HELLO", $01c08 { because of 10 PRINT"HELLO" }
J	**J 0b00**	execute the subroutine and stay in Monitor {end subroutine with **RTS**}
L	**L "mysprites",08**	load a file to the location where it was saved
L	**L "mysprites",08,3000**	load a file to start address $3000 {it's always hex numbers}
M	**M 1c00**	will list 96 bytes starting from $1c00
M	**M 1c00 1c00**	will list 8 bytes starting from $1c00
M	**M 1c00 1c08**	will list 16 bytes starting from $1c00
>	**0400 01**	displays A at left upper screen corner {it's like poke 1024,1}
R	**R**	will show the status of the registers {after **G** or **J** commands}
	Registers :	PC - program counter SR - status register AC - **LDA** {accumulator} XR - **LDX** {X-register} YR - **LDY** {Y-register} SP - stack pointer
S	**S "GO",08,f1300,f1c00**	saves "GO" from **$1300 to $1bff** in BANK 15, device 08
S	**S "0:GO",09,1300,1c00**	saves "GO" from **$1300 to $1bff** in BANK 0, drive 0, device 09 **Hint**: always add a byte to end address to save your code in full
T	**T ed00 ed7ff f2000**	transfers ROM-Character Set 1 to $2000 - $27ff {8192 – 10239}
V	**V "checkthis",08**	will check the file with the start address in memory from the point where the "checkthis" was saved
V	**V "checkthis",08,2000**	will execute verifaction of the file starting at memory address $2000
X	**X**	exits ML-Monitor and returns to BASIC 7.0
@	**@**	73, CBM DOS V3.0 1571,00,00 {version, disk drive status}
@	**@08** {or **@ 08**}	just status check device 8
@	**@08,$** {or **@ 08,$}**	list directory {**Info**: NO SPACE allowed between colon and $}
@	**@08,$0:spr***	a catalog of all files starting with SPR on drive 0, device 08
@	**@08,S0:spr***	**all files starting with "spr" will be deleted! {S is S, not $}** **{use this with precaution as there is no "are you sure"!!!}** 01, files scratched,04,00 = 4 files deleted!

Now let us make a few examples for better understanding:

Commodore C128 *default screen area is starting at $0400 {dec 1024} and end at $07e7 {dec 2023}.
Color map is starting at $d800 {dec 55295} and end at $dbe7 {dec 56295}

Info: default text screen area can be relocated and then of course other addresss would be valid.

Memory area of 58 bytes is free from $0ac6 to $0aff {dec 2758 – 2815} and is perfect for small 'projects' and help utilities!

This little routine will clear the default sprite area!

```
a 00ac6  a2 ff    ldx #$ff
a 00ac8  a9 00    lda #$00
a 00aca  9d 00 0e sta $0e00,x
a 00acd  9d ff 0e sta $0eff,x
a 00ad0  ca       dex
a 00ad1  d0 f5    bne $0ac8
a 00ad3  60       rts
a 00ad4
```

This fills the default sprite area to make blocks!

```
a 00ac6  a2 ff    ldx #$ff
a 00ac8  a9 00    lda #$ff
a 00aca  9d 00 0e sta $0e00,x
a 00acd  9d ff 0e sta $0eff,x
a 00ad0  ca       dex
a 00ad1  d0 f5    bne $0ac8
a 00ad3  60       rts
a 00ad4
```

They look the same, but the effect is the opposite.

As you can see, with ease you can make subroutine to use in BASIC program and help yourself gain on speed with task that would take a bit with BASIC. And on top of that you don't need to start up a super fancy assembler editor. It can be done with built-in Machine Language monitor just fine and the code can't be called complex. {**Note:** right ML Code above, need to be in memory before you start this BASIC example}

```
10 scnclr:print
15 ti$ = "000000":a$=" seconds"
20 for a = 0 to 511
30 poke 3584+a,255
40 next
45 print "basic 7.0  :";
50 print ti/60;:print tab(27)a$
60 ti$="000000"
65 print
70 sys 2758
75 print "ml code    :";
80 print ti/60;:print tab(27)a$
90 end
```

As you can see, **assembly with 0.03 seconds** compared to **BASIC 6.8 seconds**, is assembly way faster in case you need to have sprites as blocks or we could call them bricks. Or use it just as a function to delete specific screen area with small code adaptation.Enough room, we used just 14 out of 58 bytes!

BASIC 7.0 : 6.8 seconds
ML code : .0333333333 seconds

T **T ed00 ed7ff f2000** transfers ROM-Character Set 1 to $2000 - $27ff {8192 – 10239}

Type this above ' **T ed00 ed7ff f2000 '** and then return to BASIC with **X**

This new area can be activated with **poke 2604,24** {so the system will see it}
To modify **@** to be a full block type in direct BASIC mode : **for a=0 to 7:poke 8192+a,255:next**
As you see, you could modify all characters, go to **MONITOR** save it with:
S "mychars",08,f2000,f2800 > and load with **L"mychars",08**
then type poke 2604,24 and there you go, your character set is active!
{just don't use hi-res, e.g. graphic 1,1 etc.! your characters are there and that would clear that area}
{back to ROM-Character Set with poke 2604,20}

Type **graphic 1,1:graphic 0** and hit enter to delete the transferred characters!

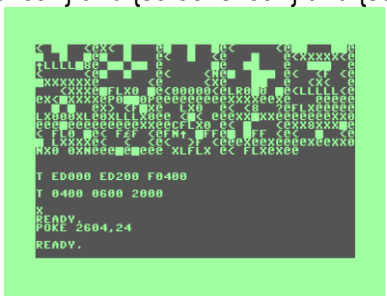
Info: you cannot transfer Character Sets from ROM directly to text screen area $0400 because 1 field {cell} is representing 1 byte here {values between #$00 - #$ff} and it needs 8 bytes per character. Meaning, an A will be spread over 8 fields {cells} and a result would be messy nothing!

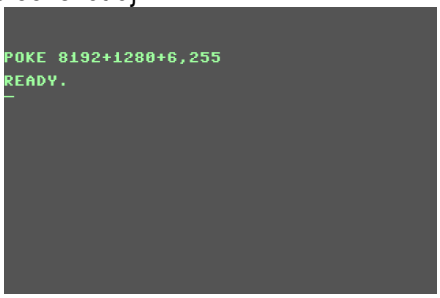
High-Resolution area is made of 8 bytes / Cell and therefore a transfer is possible.

Try this here now to make it more visually understandable to you:
Type **MONITOR** to enter the ML Monitor, then type **T ed000 ed200 f0400** and press on **ENTER**
Info: {e is BANK 14, where the Character Set is} and transfer is done to BANK 15 {e}}

Well, not much to see, right! {screenshot 1}
However, if you transfer now this to **$2000** with **T 0400 0600 2000**
then exit the Monitor with X and type in in BASIC direct mode: **poke 2604,24** {screenshot 2}, then it's on.

<<< {screenshot 1} and {screenshot 2} and {screenshot 3} >>>





Picture isn't the same any more and that is because **we transferred only the first 64 characters** = 64 x 8 bytes = 512 bytes {$0200}. Cursor won't be visible btw., no panic please, you can still write!
Cursor is ASCII Character code location 160 {$a0). To change it we need to poke 160 x 8 = 1280!
Conclusion: **poke 8192** {that's our start address, $2000} **+ 1280** {that's' our cursor start location} **+ 6** {that's one byte before the last one {**7**} of the full cursor}**, 255** {that's a full underline} will bring it back {**partly**} because we would need all 8 bytes {**0 - 7**} to make a full modification of the cursor.

Now that we know all that: Type in direct mode in BASIC **poke 8192+1280+6,255** then hit **ENTER** and there it is, underline cursor {screenshot 3} >>> Make all undone with **poke 2604,20**