Notes on using the X16 Edit ROM version

May 11, 2022

1 Introduction

This document describes the steps required to install and use the X16 Edit ROM version:

- Prepare a custom ROM image
- Create startup code using the X16 Edit API

The methods described herein are tested on Kernal version R39.

2 Prepare a custom ROM image

The first task is to prepare a custom ROM image that may be mounted in the emulator.

The standard ROM image is stored in the file rom.bin, normally found in the same folder as the emulator. If you compile the Kernal yourself, rom.bin is in the Kernal project's build folder.

The standard ROM image uses banks 0–8 as follows:

0. KERNAL
1. Keyboard layout tables
2. CBM-DOS
3. GEOS KERNAL
4. BASIC interpreter
5. Machine language monitor
6. Charset data
7. CodeX
8. Demo

The most user-friendly option would be to distribute a complete custom ROM image with banks 0–8 from the standard rom.bin, and with X16 Edit ROM code in bank 9.

The X16 Kernal is, however, not free software, and I am probably not allowed to distribute it in its original form or in an amended form. Consequently, the X16 Edit ROM version contains only the X16 Edit code (one bank, 16 kB). To get a usable ROM image, you need to append the X16 Edit ROM code to the standard rom.bin file.

In Linux and MacOS this may be done with the cat utility as follows:

```
cat rom.bin x16edit-rom-x.x.x.bin > customrom.bin
```

If you're using Windows, and cat is not available, you may try using the type command instead (not tested):

```
type rom.bin x16edit-rom-x.x.x.bin > customrom.bin
```

Select the custom rom on starting the emulator:

```
xemu -rom customrom.bin -sdcard sdcard.img
```

3 Create startup code using the X16 Edit API

Starting the X16 Edit ROM version requires that you enter a small startup routine in RAM.

The startup code must use the public API of the program, which currently consists of the entry points described below.

3.1 Default entry point

Description:	The program's default entry point, opens the editor with an empty new buffer
Call address:	$C000
Params:	.X = First RAM bank used by the program .Y = Last RAM bank used by the program

Example startup code entered into the builtin monitor:

```
.A 9E00 LDA $01         ;Get current ROM bank...
.A 9E03 PHA             ;... and store it on the stack
.A 9E04 LDA #$09        ;Store bank 9...
.A 9E06 STA $01         ;...in the ROM select
.A 9E09 LDX #$01        ;Set first RAM bank used to 1
.A 9E0B LDY #$FF        ;Set last RAM bank used to 255
```

```
.A 9E0D JSR $C000       ;Call editor entry point
.A 9E10 PLA             ;Retrieve original ROM bank...
.A 9E11 STA $01         ;...and write it to ROM select
.A 9E14 RTS
```

3.2 Load text file entry point

Description: Loads a text file into the editor on startup
Call address: $C003
Params: .X = First RAM bank used by the program
.Y = Last RAM bank used by the program
R0 ($02–$03) = Pointer to file name, least significant byte first
R1 ($04) = File name length, or 0 = no file

Example startup code entered into the builtin monitor:

```
.A 9E00 LDA $01         ;Get current ROM bank...
.A 9E03 PHA             ;... and store it on the stack
.A 9E04 LDA #$09        ;Store bank 9...
.A 9E06 STA $01         ;...in the ROM select
.A 9E09 LDX #$01        ;Set first RAM bank used to 1
.A 9E0B LDY #$FF        ;Set last RAM bank used to 255
.A 9E0D LDA #$23        ;Get filename AddressL...
.A 9E10 STA $02         ;...and store in $02
.A 9E12 LDA #$9E        ;Get filename AddressH...
.A 9E15 STA $03         ;...and store in $03
.A 9E17 LDA #$05        ;Get filename length...
.A 9E19 STA $04         ;...and store in $04
.A 9E1B JSR $C003       ;Call editor entry point
.A 9E1E PLA             ;Retrieve original ROM bank...
.A 9E1F STA $01         ;...and write it to ROM select
.A 9E22 RTS

.M 9E23 <FILE NAME DATA, 5 BYTES IN THIS EXAMPLE>
```

3.3 Load text file with options entry point

Description: Applies specified editor options, and then loads a text file into the editor on startup
Call address: $C006
Params: .X = First RAM bank used by the program
.Y = Last RAM bank used by the program

R0 ($02–$03) = Pointer to file name, least significant byte first
R1L ($04) = File name length, or 0 = no file
R1H ($05) = Auto indent on/off (bit 0), word wrap on/off (bit 1)
R2L ($06) = Tab width (1..9)
R2H ($07) = Word wrap position (10..250)
R3L ($08) = Current device number (8..30)
R3H ($09) = Screen color (bits 0-3 text, bits 4-7 background)
R4L ($0A) = Header color (bits 0-3 text, bits 4-7 background)
R4H ($0B) = Status bar color (bits 0-3 text, bits 4-7 background)

Options out of range are silently ignored, and the editor default values are used instead. If a color value of 0 is given, the editor will use the default value. And if the file name length is 0, the editor will not try to load a file on startup.

Example startup code entered into the builtin monitor:

```
.A 9E00 LDA $01         ;Get current ROM bank...
.A 9E03 PHA             ;... and store it on the stack
.A 9E04 LDA #$09        ;Store bank 9...
.A 9E06 STA $01         ;...in the ROM select
.A 9E09 LDX #$01        ;Set first RAM bank used to 1
.A 9E0B LDY #$FF        ;Set last RAM bank used to 255
.A 9E0D LDA #$F0        ;Get filename AddressL...
.A 9E10 STA $02         ;...and store in $02
.A 9E12 LDA #$9E        ;Get filename AddressH...
.A 9E15 STA $03         ;...and store in $03
.A 9E17 LDA #$05        ;Get filename length...
.A 9E19 STA $04         ;...and store in $04
.A 9E1B LDA #$01        ;Auto-increment on, and...
.A 9E1D STA $05         ;word wrap off
.A 9E1F LDA #$05        ;Set tab width...
.A 9E21 STA $06         ;to 5
.A 9E23 LDA #$28        ;Set word wrap position...
.A 9E25 STA $07         ;to 40
.A 9E27 LDA #$08        ;Set current device...
.A 9E29 STA $08         ;to 8
.A 9E2B LDA #$01        ;Set screen color to...
.A 9E2D STA $09         ;white on black
.A 9E2F LDA #$00        ;Set header color to...
.A 9E3B STA $0A         ;default value
.A 9E3D STA $0B         ;Set status bar color to default
.A 9E3F JSR $C006       ;Call entry point
```

```
.A 9E4C PLA             ;Retrieve original ROM bank...
.A 9E4D STA $01         ;...and write it to ROM select
.A 9E4F RTS

.M 9EF0 <FILE NAME DATA, 5 BYTES IN THIS EXAMPLE>
```

4 X16 Edit application signature

Even though it is likely that X16 Edit will be stored in the first available ROM bank, it is not recommended to make that assumption when you write code that starts the program.

The Kernal ROM bank layout may change over time, and the user may also store other ROM based utilities.

Instead of hard coding the ROM bank, you may let your program step through all ROM banks looking for the X16 Edit application signature stored at address $FFF0.

The application signature consists of ten bytes: the text "X16EDIT" followed by the version number (major, minor and patch) stored as three byte values, for example as follows:

```
$58,$31,$36,$45,$44,$49,$54,$00,$05,$00 ;X16EDIT v 0.5.0
```

5 Final comments

The installation and startup process is not very convenient for the end-user at this stage.

In order for a ROM based program to be usable, there needs to be a BASIC command to run the program.

For now the X16 Edit ROM version it is of most interest to those who want to integrate a text editor with their program, for example assemblers, compilers and other development tools.