Programming the Commodore 64 again

Written by Jens Christian Ingvartsen Thomsen 2020

Intro

My name is Jens Christian Ingvartsen Thomsen and I used to program the Commodore 64 way way back in the eighties, when I was in my start twenties.
I first programmed the VIC-20 and then the C64, C128 and the Amiga. Then I shifted to PC where I was programming in Pascal, Visual Basic, Visual Basic.net and eventually ended up in C#, which I'm today is using in my daily job.

Then last year just before x-mas I bought a theC64 Mini and short after I ordered the C64 (maxi) and I remembered the excitement I felt way back when I was programming on my old Commodore 64.

So, to see if I could get some of that excitement back, I made a new year's resolution to myself.
I want to program at least one game in C64 basic and at least one game in C64 assembly.

So, I bought a Commodore 64c, because I could not find the power supply to the Commodore 64c I already had. Of course, I found the power supply after I bought the new one.

I then started to program on the machine, but I moved pretty fast to do it in the Vice emulator on my Mac. Using the emulator was a bit slow so I decided to code in Visual Studio code and then copy the code to the emulator. I did this for a while until I discovered CBM Prg Studio and changed to my PC to program my game. In CMB Prg Studio I could also design the sprites, characters and screens

Later I started to use SublimeText and Kick Assembler on the Mac so in this book I will show the assembler code for CBM Prg Studio first and later for Kick Assembler.

After two weeks in January my first basic game was finished, with a few minor bugs, but I was satisfied. The game was a Tic Tac Toe clone called Ticle Tocle. It can be found online at
http://games.trisect.dk/ticle-tocle

The basic listing can be found in a later chapter.

After the basic game it was now the assembler game I was going to make. As my assembly knowledge is a bit rusty. I had to learn a lot of it again and I decided to write this book at the same time, writing down the things I learned, so others could learn from it.

So, let's get started and I hope you will enjoy this together with me.

Jens

Foreword

This book was written because I needed a place to store all the information I have gathered while I was programming my game in Commodore 64 assembler.

But I'm glad I did. It could be useful for others.

It was fun learning assembler again and learning new tools like CBM Prg Studio.
Later I had to learn Kick Assembler for this book.

The Kick Assembler was not so easy, because it's a little different from the CBM Prg Stuido assembler. But I got some help online, like in Discord. So, you can use this book if you are using CBM Prg Studio or Kick Assembler.

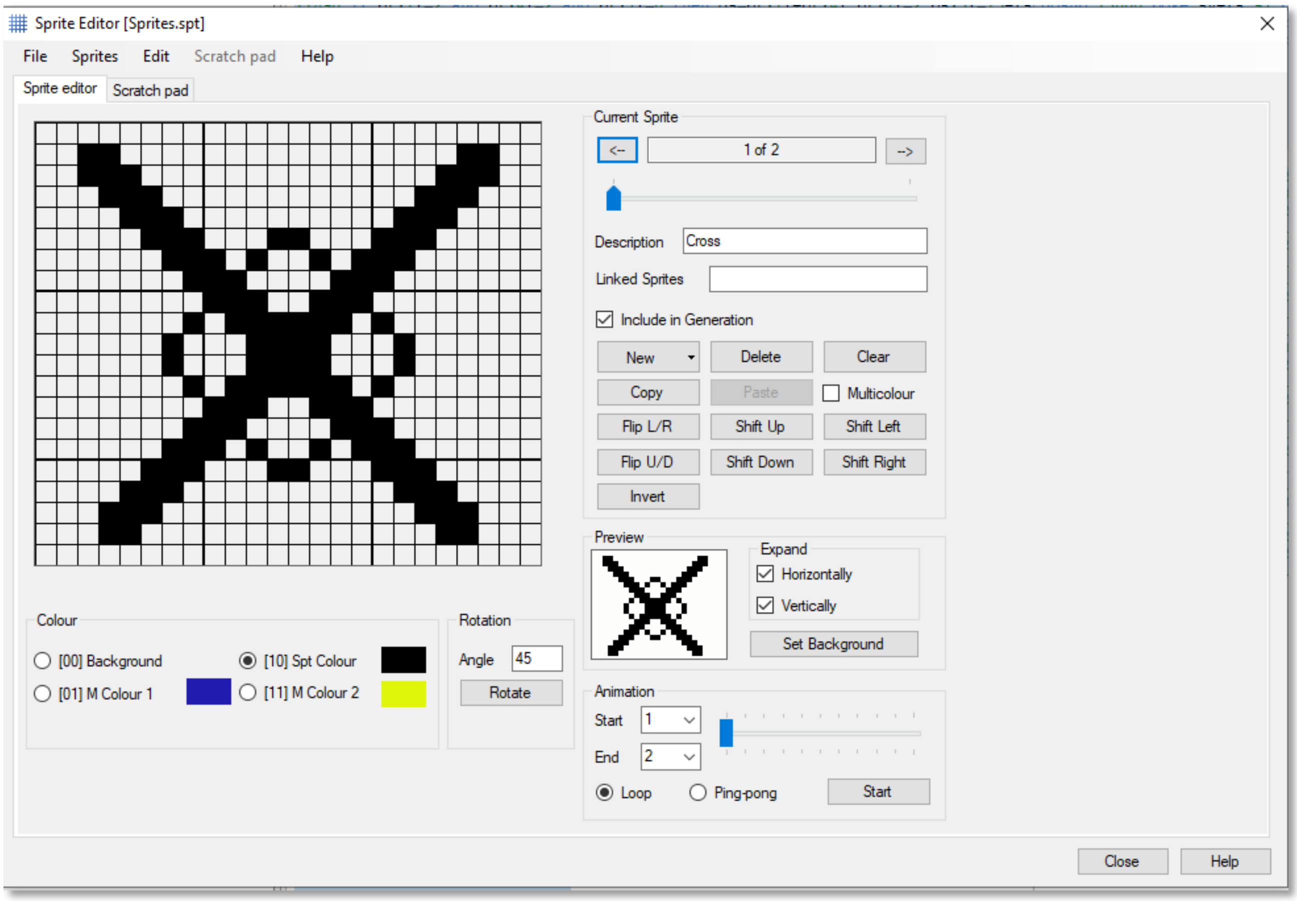
So, I hope you will enjoy reading this book.

My first game for my 2020 new year's resolutions.

The basic listing can be seen in the back of this book.

As I mentioned before, my first game for 2020 was a Tic Tac Toe clone in basic called Ticle Tocle.
I was programming in VSCode and copying it to the emulator, and it was working fine until I needed to do a renumbering of my code.
I wrote a small program in C# that could load my code and do the renumbering, but it was not so good at handling the *goto* and *gosub* commands. So that's when I found the CBM Prg Studio, because it had that renumbering tool and it worked just fine. Besides it also had the sprite, charset and screen designers.

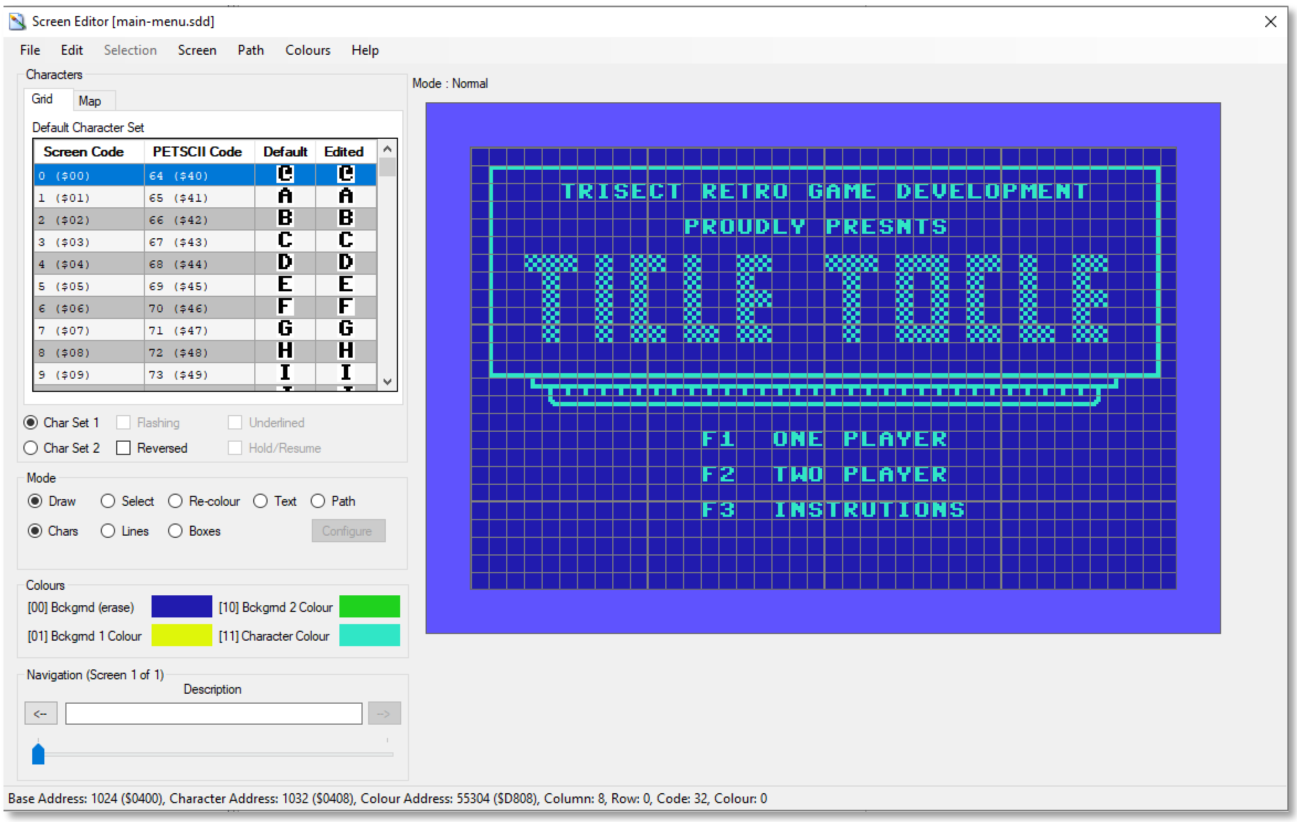
So, I used CBM Prg Studio to finish my game.


Sprite Editor [Sprites.spt]
File Sprites Edit Scratch pad Help
Sprite editor Scratch pad
Current Sprite
<-- 1 of 2 -->
New Delete Clear
Copy Paste
Flip L/R Shift Up Shift Left
Flip U/D Shift Down Shift Right
Invert
Preview
Set Background
Rotate
Start
Close Help

Description: Cross
Linked Sprites:
[✓] Include in Generation
[ ] Multicolour
Expand: [✓] Horizontally [✓] Vertically
Colour: ( ) [00] Background ( ) [01] M Colour 1 (●) [10] Spt Colour ( ) [11] M Colour 2
Rotation: Angle 45
Animation: Start 1, End 2, (●) Loop ( ) Ping-pong

In the sprite designer I designed my sprites for the game and exported them as basic code which I then could use in my code.

After designing the sprites, I used the Screen Editor to create the screens I used in the game.



Default Character Set

Screen Code	PETSCII Code	Default	Edited
0 ($00)	64 ($40)	@	@
1 ($01)	65 ($41)	A	A
2 ($02)	66 ($42)	B	B
3 ($03)	67 ($43)	C	C
4 ($04)	68 ($44)	D	D
5 ($05)	69 ($45)	E	E
6 ($06)	70 ($46)	F	F
7 ($07)	71 ($47)	G	G
8 ($08)	72 ($48)	H	H
9 ($09)	73 ($49)	I	I

[✓ Char Set 1] Char Set 2 ☐ Flashing ☐ Reversed ☐ Underlined ☐ Hold/Resume

Mode: [✓ Draw] Select Re-colour Text Path
[✓ Chars] Lines Boxes

Colours

[00] Bckgrnd (erase)	🟦	[10] Bckgrnd 2 Colour	🟩
[01] Bckgrnd 1 Colour	🟨	[11] Character Colour	🟦

Navigation (Screen 1 of 1)
Description

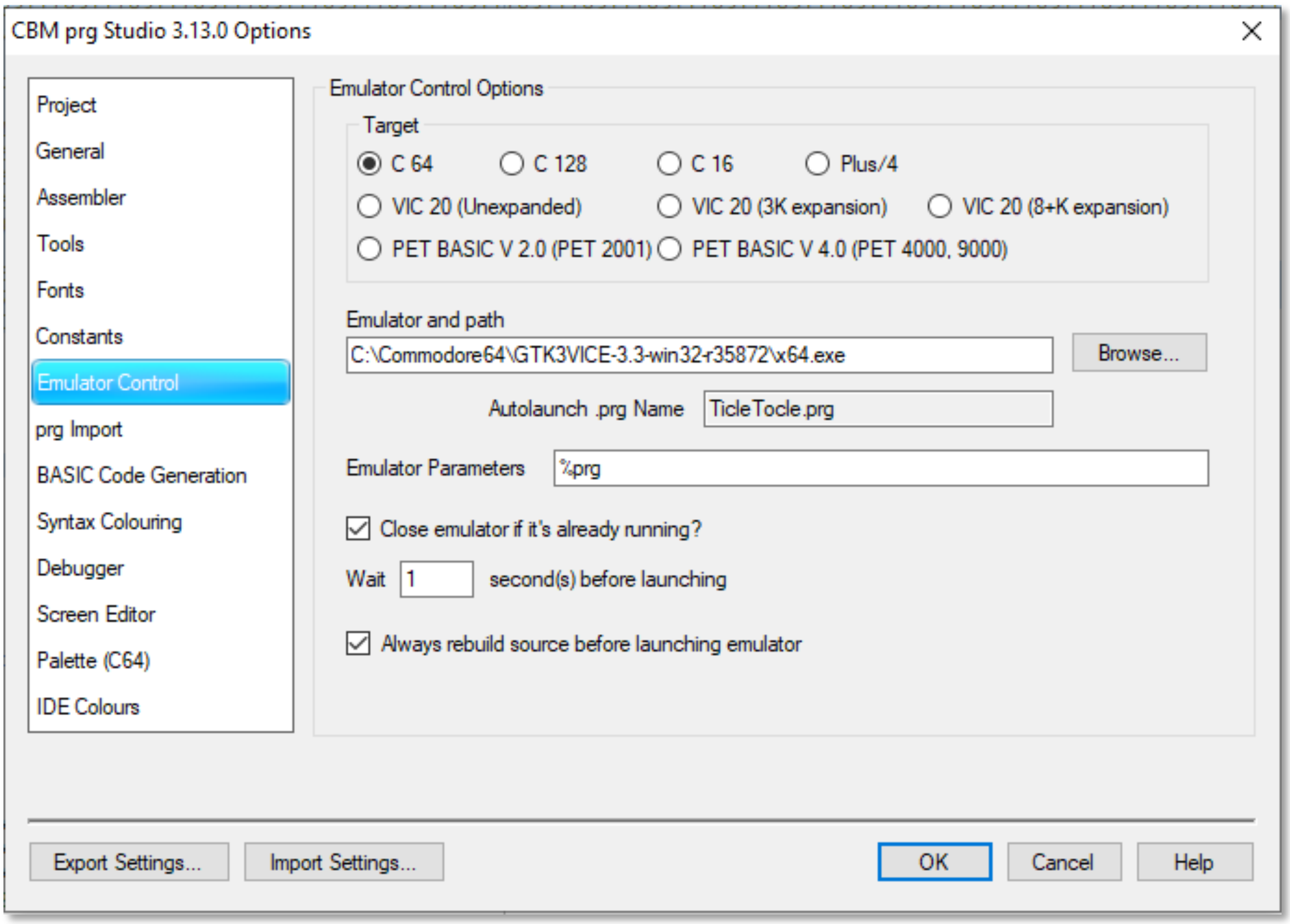
Base Address: 1024 ($0400), Character Address: 1032 ($0408), Colour Address: 55304 ($D808), Column: 8, Row: 0, Code: 32, Colour: 0

Here I designed the menu screen. You can also design your own character set and use them in the screen editor. I did not do that for my basic game.
After you have finished designing the screen you can export them to basic. Choose export BASIC and the program creates the screen in Basic for you.

```
830 print ""
840 print "
{cyan}O{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{
183}{183}{183}{183}{183}{183}{183}{183}{183}{183}{183}P"
850 print " {180}   trisect retro game development   {170}"
860 print " {180}                                    {170}"
870 print " {180}         proudly presents           {170}"
880 print " {180}                                    {170}"
890 print " {180} {166}{166}{166} {166} {166}{166} {166}  {166}{166}   {166}{166}{166} {166}{166}{166} {166}{166} {166}  {166}{166}
{170}"
900 print " {180}  {166}  {166} {166}  {166}  {166}     {166}  {166} {166} {166}  {166}  {166}   {170}"
910 print " {180}  {166}  {166} {166}  {166}  {166}{166}    {166}  {166} {166} {166}  {166}  {166}{166}  {170}"
920 print " {165}  {166}  {166} {166}  {166}  {166}     {166}  {166} {166} {166}  {166}  {166}   {167}"
930 print " {165}  {166}  {166} {166}{166} {166}{166} {166}{166}    {166}  {166}{166}{166} {166}{166} {166}{166} {166}{166}  {167}"
940 print " {165}                                    {167}"
950 print "
L{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{175}{1
75}{175}{175}{175}{175}{175}{175}{175}{175}{175}{186}"
960 print "
{173}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{178}{17
8}{178}{178}{178}{178}{178}{178}{189}"
970 print "    JCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCK"
980 print ""
990 print "             f1  one player"
1000 print ""
1010 print "             f2  two player"
1020 print ""
1030 print "             f3  instrutions"
1040 print ""
1050 print ""
1060 print ""
1070 print ""
```

Here’s the code that the Screen Editor exports.

Then it was just to code the game and test it every time I made some changes.
In order to make CBM Prg Studio open the Vice64 emulator you must go in under settings and set it up.



CBM prg Studio 3.13.0 Options

Emulator Control Options

Target: [✓ C 64] C 128 C 16 Plus/4 VIC 20 (Unexpanded) VIC 20 (3K expansion) VIC 20 (8+K expansion) PET BASIC V 2.0 (PET 2001) PET BASIC V 4.0 (PET 4000, 9000)

Emulator and path: C:\Commodore64\GTK3VICE-3.3-win32-r35872\x64.exe

Autolaunch .prg Name: TicleTocle.prg

Emulator Parameters: %prg

[✓] Close emulator if it's already running?

Wait 1 second(s) before launching

[✓] Always rebuild source before launching emulator

Here you can see the it points to the emulator and that it is a C64.

To learn more about CBM Prg Studio go to the authors webpage.

http://www.ajordison.co.uk/

But now on to the really exciting stuff, assembler code.

Programming in assembler on the Commodore 64

In this book I will be using CBM Prg Studio assembler and later show the code for Kick Assembler.

To start programming in assembler, create a new project in CBM Prg Studio and called it whatever you want. Just make sure you select C64 when you create the project.

On the left side in the Studio right click on Assembly files and Add new file. Call it main.asm.

You now have a completely empty file and can start coding your assembler code.
Go a few lines down and enter

```
*=$0810
```

This tells the assembler where we want our program to start.
Below that we write.

```
        lda #02
        sta $d020
        sta $d021
        rts
```

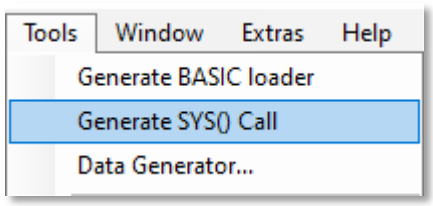
Now save your file and hit F5 to compile the program and start the emulator. The emulator should start up, but nothing happens. This is because it does not know where to start the program, there is no basic to tell it what to do. We could just write sys 2064. <u>2064 is decimal for $0810.</u>

Now both the background and the border are red. Here is an explanation for the code.

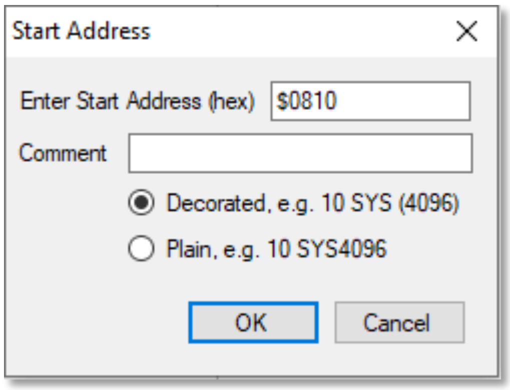
```
        lda #02         ;** Load the color 2 (red) into the accumulator
        sta $d020       ;** Store the accumulator value into $d021 -> Border
        sta $d021       ;** Store the accumulator value into $d020 -> Background
        rts             ;** Return from subroutine.
```

All is working fine but we don't want to write sys 2064 every time we want to test our program.

Go to the top of our code file.
Under the Tool menu choose Generate SYS() Call.



In the next screen write the start address we typed in earlier.



Enter Start Address (hex) $0810
Comment
[✓ Decorated, e.g. 10 SYS (4096)]
Plain, e.g. 10 SYS4096

Click OK and see what your code look like now.

```
; 10 SYS (2064)

*=$0801
            BYTE    $0E, $08, $0A, $00, $9E, $20, $28, $32, $30, $36, $34, $29, $00, $00, $00


*=$0810

            lda #02
            sta $d020
            sta $d021
            rts
```

Hit F5 to run your program. Now the program starts itself, it is no longer needed to type the sys command.

You have created, compiled and run your first assembly program on the C64.

In the next sections of this book I will show you small code listings that do different things. Such as showing sprites, writing text to the screen and using the joystick and so on.
And in the end, we should be able to program a little C64 game.
And in every example, I will try to explain what it does.
As I learn new things you will learn new things.

The example you just saw was written in CBM Prg Studio assembler.
The same code in Kick Assembler would look like this.

```
BasicUpstart2(entry)

*=$0810
entry:
            lda #02
            sta $d020
            sta $d021
            rts
```

It does the same, except it uses the BasicUpstart2 command which is a Kick command to make it start in basic. The assembler code will be show in CBM Prg Studio and Kick assembler code style in the back of the book.

Table of contents.