TSconf: Utils

Development

TSconf Developer Tools

- ASM includes
- Graphic conversion, graphics editor
- Spg builder

ASM includes

I usually write all the code in sublime text + z80 highlight by psb, for standard sjasmplus. To export a binary, use the binary save:

```
device zxspectrum128
org #8000
start
;         ...
;         ...
end
include "tsconfig.asm"
savebin "_spg/lirus_main.bin",start,end-start
```

As a result of compilation, the file lirus_main.bin will be generated in the _spg folder, which will later go to the collector.

In my previous articles, I constantly used the names of the ports instead of their numbers. Alas, most often the names carry more semantic load than just numbers :)

To use the names of ports and significant bits of the system, you should add them to your source with the include operator "tsconfig.asm"

Content tsconfig.asm looks like this:

```
; -$\,$-$\,$-$\,$-$\,$-$\,$-$\,$- definitions
; -$\,$- TS-config port regs

VCONFIG         equ \$00AF
STATUS          equ \$00AF
VPAGE           equ \$01AF
GXOFFSL         equ \$02AF
GXOFFSH         equ \$03AF
GYOFFSL         equ \$04AF
GYOFFSH         equ \$05AF
TSCONFIG        equ \$06AF
PALSEL          equ \$07AF
BORDER          equ \$0FAF
PAGE0           equ \$10AF
```

```
PAGE1           equ \$11AF
PAGE2           equ \$12AF
PAGE3           equ \$13AF
FMADDR          equ \$15AF
TMPAGE          equ \$16AF
T0GPAGE         equ \$17AF
T1GPAGE         equ \$18AF
SGPAGE          equ \$19AF
DMASADDRL       equ \$1AAF
DMASADDRH       equ \$1BAF
DMASADDRX       equ \$1CAF
DMADADDRL       equ \$1DAF
DMADADDRH       equ \$1EAF
DMADADDRX       equ \$1FAF
SYSCONFIG       equ \$20AF
MEMCONFIG       equ \$21AF
HSINT           equ \$22AF
VSINTL          equ \$23AF
VSINTH          equ \$24AF
DMALEN          equ \$26AF
DMACTR          equ \$27AF
DMASTATUS       equ \$27AF
DMANUM          equ \$28AF
FDDVIRT         equ \$29AF
INTMASK         equ \$2AAF
T0XOFFSL        equ \$40AF
T0XOFFSH        equ \$41AF
T0YOFFSL        equ \$42AF
T0YOFFSH        equ \$43AF
T1XOFFSL        equ \$44AF
T1XOFFSH        equ \$45AF
T1YOFFSL        equ \$46AF
T1YOFFSH        equ \$47AF

; TS parameters
FM_EN           equ \$10

; VIDEO
VID_256X192     equ \$00
VID_320X200     equ \$40
VID_320X240     equ \$80
VID_360X288     equ \$C0
VID_RASTER_BS equ 6

VID_ZX          equ \$00
VID_16C         equ \$01
```

```
VID_256C        equ \$02
VID_TEXT        equ \$03
VID_NOGFX       equ \$20
VID_MODE_BS equ 0

; PALSEL
PAL_GPAL_MASK equ \$0F
PAL_GPAL_BS equ 0
PAL_T0PAL_MASK equ \$30
PAL_T0PAL_BS equ 4
PAL_T1PAL_MASK equ \$C0
PAL_T1PAL_BS equ 6

; TSU
TSU_T0ZEN equ \$04
TSU_T1ZEN equ \$08
TSU_T0EN equ \$20
TSU_T1EN equ \$40
TSU_SEN equ \$80

; SYSTEM
SYS_ZCLK3_5 equ \$00
SYS_ZCLK7 equ \$01
SYS_ZCLK14 equ \$02
SYS_ZCLK_BS equ 0

SYS_CACHEEN equ \$04

; MEMORY
MEM_ROM128 equ \$01
MEM_W0WE equ \$02
MEM_W0MAP_N equ \$04
MEM_W0RAM equ \$08

MEM_LCK512 equ \$00
MEM_LCK128 equ \$40
MEM_LCKAUTO equ \$80
MEM_LCK1024 equ \$C0
MEM_LCK_BS equ 6

; INT
INT_VEC_FRAME equ \$FF
INT_VEC_LINE equ \$FD
INT_VEC_DMA equ \$FB

INT_MSK_FRAME equ \$01
```

```
INT_MSK_LINE equ \$02
INT_MSK_DMA equ \$04

; DMA
DMA_WNR         equ \$80
DMA_SALGN       equ \$20
DMA_DALGN       equ \$10
DMA_ASZ         equ \$08

DMA_RAM         equ \$01
DMA_BLT         equ \$81
DMA_FILL     equ \$04
DMA_SPI_RAM     equ \$02
DMA_RAM_SPI     equ \$82
DMA_IDE_RAM     equ \$03
DMA_RAM_IDE     equ \$83
DMA_RAM_CRAM    equ \$84
DMA_RAM_SFILE   equ \$85

; SPRITES
SP_XF equ \$80
SP_YF equ \$80
SP_LEAP equ \$40
SP_ACT equ \$20

SP_SIZE8 equ \$00
SP_SIZE16 equ \$02
SP_SIZE24 equ \$04
SP_SIZE32 equ \$06
SP_SIZE40 equ \$08
SP_SIZE48 equ \$0A
SP_SIZE56 equ \$0C
SP_SIZE64 equ \$0E
SP_SIZE_BS equ 1

SP_PAL_MASK equ \$F0

SP_XF_W equ \$8000
SP_YF_W equ \$8000
SP_LEAP_W equ \$4000
SP_ACT_W equ \$2000

SP_SIZE8_W equ \$0000
SP_SIZE16_W equ \$0200
SP_SIZE24_W equ \$0400
SP_SIZE32_W equ \$0600
```

```
SP_SIZE40_W equ \$0800
SP_SIZE48_W equ \$0A00
SP_SIZE56_W equ \$0C00
SP_SIZE64_W equ \$0E00
SP_SIZE_BS_W equ 9

SP_X_MASK_W equ \$01FF
SP_Y_MASK_W equ \$01FF
SP_TNUM_MASK_W equ \$0FFF
SP_PAL_MASK_W equ \$F000

; TILES
TL_XF equ \$40
TL_YF equ \$80
TL_PAL_MASK equ \$30
TL_PAL_BS equ 4

TL_XF_W equ \$4000
TL_YF_W equ \$8000

TL_TNUM_MASK_W equ \$0FFF
TL_PAL_MASK_W equ \$3000
TL_PAL_BS_W equ 12
```

The use of port names and bits greatly improves the reading of the source.

Graphic conversion, graphics editor

To use graphics in a demo, you need to draw / cook it in a graphic editor.

I use Photoshop, in which I perform all the operations for preparing graphics - resizing, clipping, etc.

At the last stage, conversion into indexed colors is performed. For tiles / sprites you need to remember that one of the colors will be used as the transparency color.

At the last stage, everything is unloaded in tga format (256 colors, 8 bits) - even if 16 or less colors are used.

To set the desired color as the first (0th, transparency) in the palette, you can use a very pleasant and useful for our purposes Usenti editor. Its biggest advantage is the means for working with the palette - sorting, cell exchange, and so on. And draw them convenient, try.

We receive the received tga in the wonderful converter with which help separate files of pixel data of the image and its palette will be prepared.

The tga2ts.exe converter itself is here , you need both files for it to work - both tga2ts.exe and levels.map.

Example of use:

```
tga2ts.exe back9.tga
```

we will get the following set of files:

```
back9.tga.pal
back9.tga.pix
back9.tga.pix4
```

* .pal - palette, 512 bytes (256 * 2 bytes per color). When using 16 color images, I use pens to remove the zeros, leaving only 32 bytes of the palette.

* .pix - image in 256 color format - byte per point (byte per color)

* .pix4 - image in 16 color format - byte on two points

All image data is linear.

So, the converted images are received. Given the possible large size - these files must be placed in memory for use. Naturally, we can save files to a floppy disk, write a bootloader, and load the file in the old way, as usual. But what if you need to immediately load into memory more than 600 KB? for example, megabyte so *jwa* ?

For this, it makes sense to use the SPG collector.

SPG collector

This program is designed to generate * .spg files. This format is a smart snapshot of the pentevo memory pages used in the program, and when loading it places the data blocks in the necessary pages, after which it starts at the address specified in the spg header.

* .spg allows you to run WildCommander

Collector can be found here , data on the location of files in memory are described in spgbld.ini.

Information in spgbld.ini about the location in memory is set as follows:

```
Desc = lirus    ; name without quotes
Start = #8000    ; start address
Stack = #5fff    ; initial stack location
Resident = #5B00    ; location of the resident
Page3 = 0    ; home page
Clock = 2    ; processor speed. 0 - 3.5 MHz / 1 - 7 MHz / 2 - 14 MHz / 3 - 14 MHz overcl
INT = 0    ; interrupts are off
Pager = 0    ; Page manager address, 0 - without manager

;Block = #e000,  5,violent.bin    ; specify the location address in memory, page, file.
Block = #c000,  2,lirus_main.bin    ; this file will be located at #8000
Block = #c000,  0,lirus_p0.bin    ; and this one is in pag 0, address #c000

Block = #c000, #10,player49152.bin
Block = #c800, #10,nq-first_warning-8.pt3
Block = #e000, #10,outro.pt3
Block = #c000, #11,tiles2.tga.pix4
Block = #d000, #12,geebeeyay-8x8.tga.pix4
```

A feature of the collector is that:

- Location address must be a multiple of 512 (#200)
- The address is indicated as a position in the memory page, starting with the address #c000.

  i.e. address #6000 is page 5, address #e000 (offset #2000); #b000 is page 2, address #f000 (offset #3000)

  Clarification - everything lies in our pag that can be connected to different windows (0-3).

  Accordingly, the specified address will always be given in the form #0000 - #3e00, and **the upper two bits of the address will be RESET OSEN** as unnecessary.

  Accordingly, if we want to load the block into the 5th page from the address #6000 - we indicate:

  Block = #2000, 5, file.bin; this file will be located at #2000 of the included window.

  If we include this pag in window 1 - this is 6000, if in window 2 - then the address is a000, if in window 3 - e000
- The size of the data block may exceed the standard page size (16kb), while the data will occupy subsequent pages of memory

After doing

```
spgbld.exe -b spgbld.ini TEST.spg -c 0
```

the TEST.spg file is assembled and created, which can be opened both on PentEVO from WC and in the emulator.

The -c 0 parameter indicates that no compression is needed during assembly. If you set this flag to 1, then the assembly will call the wonderful mhmt.exe compressor from our dear friend lvd , which will test for the best compression and the best block in size will fit into the assembly.

Naturally, it takes time - both when building and running on real hardware, consider the moment of waiting for decompression when starting the file on PentEVO.

Questions

:? I can not download files

! Contact

Literature: Doc on CG builder