TIC-80

Display: 240x136, 16 colors, Input: 4 pads, 8 buts, kb/mouse

Sprites: 256 8x8 fore sprites, 256 8x8 bg tiles

Map: 240x136 cells, 1920x1088 (240*8, 136*8)

Sound: 4 channels/envelopes, Code: 64KB (pro 512, 8 banks)

Mem map:

00000	SCREEN	16320	240x136=4b/pix
03FC0	PALETTE	48	16x24b RGB
03FF0	PALETTE MAP	8	16x4b color indexes
03FF8	BORDER COLOR	1	4b color
03FF9	SCREEN OFFSET	2	horz/vert -128+127
03FFB	MOUSE CURSOR	1	index of mouse curs
03FFC	...	4	
04000	BG SPRITES	8192	256 8x8 4b 0..255
06000	FG SPR/TILES	8192	256 8x8 4b 256..512
08000	MAP	32640	8x8 240x136 cells
0FF80	GAMEPADS	4	state of 4 gpads
0FF84	MOUSE	4	mouse X/Y/buttons
0FF88	KEYBOARD	4	codes: 4 keys max
0FF8C	...	16	
0FF9C	SOUND REGS	72	18 byte x 4 ch
0FFE4	WAVEFORMS	256	16 wave/ 32x4b each
100E4	SFX	4224	64 sounds
11164	MUSIC PATTERNS	11520	64 rowsx 60 patts
13E64	MUSIC TRACKS	408	8 tracks
13FFC	MUSIC POS	4	state of music
14000	...	0	

Cart metadata:

```
dofile("example.lua")
-- title: game title
-- author: author
-- desc: short desc
-- script: lua (moon/wren/js/fennel)
-- input: gamepad (mouse/keyboard)
-- saveid: MyAwesomeGame
```

Place "example.lua" in TIC dir to edit using external editor

Callbacks:

```
TIC() -> called once per frame
SCN(line) -> called once per scanline
OVR() -> called once per frame, overlay layer
```

Palette:

Build palette here then add palette setter, ex pico8:

```
palet="0000001d2b537e255383769cab5236008751ff004d5f574
fff77a8ffa300c2c3c700e436ffccaa29adffffec27fff1e8"
for i=1,palet:len() do
print(0x3fc0+i-1,tonumber("0x"..palet:sub(i,i)));end
```

Key Codes:

01 A	02 B	03 C	04 D	05 E	06 F	07 G	08 H
09 I	10 J	11 K	12 L	13 M	14 N	15 O	16 P
17 Q	18 R	19 S	20 T	21 U	22 V	23 W	24 X
25 Y	26 Z	27 0	28 1	29 2	30 3	31 4	32 5
33 6	34 7	35 8	36 9	37 -	38 =	39 (	40)
41 \	42 ;	43 '	44 `	45 ,	46 .	47 /	
48 SPC	49 TAB	50 RET	51 BKSP	52 DEL	53 INS	54 PGUP	55 PGDN
56 HOME	57 END	58 UP	59 DOWN	60 LEFT	61 RGHT	62 CAPS	63 CTRL
64 SHFT	65 ALT						

Graphics:

```
cls(color=0)
pix(x,y[color]) [-> color]
circ(x,y,r,color) -- filled circle
circb(x,y,r,color) -- border circle
rect(x,y,w,h,color) -- filled rect
rectb(x,y,w,h,color) -- border rect
line(x0,y0,x1,y1,color)
spr(id,x,y,colorkey=-1,scale=1,flip=0,
    rotate=0,w=1,h=1)
  -- colorkey: opaque (-1) or color index
  -- flip: 0,1,2,3 -> no,horiz,vert,both
  -- rotate: 0,1,2,3 -> 0, 90, 180, 270
  -- w,h: how many sprites to draw
tri(x1,y1,x2,y2,x3,y3,color)
textri(x1,y1,x2,y2,x3,y3,u1,v1,u2,v2,u3,v3,
    use_map=false,colorkey=-1)
  -- use_map: sprites, tiles -> false, true
  -- colorkey: opaque (-1) or color index(s)
map(x=0,y=0,w=30,h=17,sx=0,sy=0,colorkey=-1,
    scale=1,remap=nil)
  -- x,y,w,h: rect of map tiles to draw
  -- colorkey: opaque (-1) or color index
  -- scale: scaling drawn tiles?
  -- remap: func(tile,x,y)->tile,flip,rot
font(text,x,y,colorkey,charwidth,charheight,
    fixed=false,scale=1) -> width
  -- bmpfont using sprites, fixed=true -> mono
  -- start @ sprite 256 is '0'
print(text,x=0,y=0,color=15,fixed=false,
    scale=1,smallfont=false) -> width
clip(x,y,w,h)
fget(index,flag:0..7) -> bool -- check spr flag
fset(index,flag:0..7,set) -- re/set spr flag
```

Sound:

```
sfx(id,note,duration=-1,channel=0,volume=15,
    speed=0)
music(track=-1,frame=-1,row=-1,loop=true)
```

Input:

```
btn(id:0..31) -> pressed
btnp(id:0..31,[hold],[period]) -> pressed
key(code) -> pressed -- key state in cur frame
keyp(code,hold=0,period=0)
  -- key just pressed, or held after $hold ticks
  -- $period is ticks til next true if $hold
mouse() -> x,y,left,middle,right,scrollx,scrolly
```

Memory:

```
peek(address) -> value
peek4(address) -> value -- 4 bits
poke(address,value) -> value
poke4(address,value) -> value -- 4 bits
pmem(index:0..255,[value]) [-> value]
  -- load/save int from/to persistent mem
memcpy(dest_addr,source_addr,length)
mget(x,y) -> id -- get bgspr id at map x,y
mset(x,y,id) -- change bgspr id at map x,y
```

System:

```
trace(msg,color)
time() -> milliseconds since game start
tstamp() -> current unix timestamp
exit()
reset() -- reset cart
sync([mask=0],[bank=0],[tocart=false])
  -- pro: bank switching any section
```

Tracker

```
A: break, SPC: prev note, RET: play/stop frame
SHFT+RET: play pat @ cursor
Note NNOSSV -> "NN" note[#],"O" octave:1..8,
    "SS" sfx voice:0..63, "V" volume:0..F
```

Editor:

```
F1: code, F2: sprites, F3: map, F4: sfx, F5: mus
F6: crt, F7: assign cover img, F8: screenshot
F9: GIF record, F11: window mode
```