Turbo-tape Loader source

Depending on the selected options, this is the absolute minimum that is required to load Turbo-tape data. Sources are in XA assembler format. This assembles right out of the box to stack $0100. Use SYS 256 or SYS 2^8 to launch the program. :) Uncomment DEFINEs for the different options. Enjoy!

Enthusi 11/2010

```
.word $0100

  zp1       =$c3
  zp2       =$c4
  BITCOUNT  =$02
  bytebuffer    =$bd
  zp3       =$c1
  zp4       =$c2
  chkbyte=$10
;---------------------------------
  *=$0100;
#define FX 1
#define CHKSUM 1
#define BASICRUN 1
#define MESSAGES 1
#define RESETMODE 1

  sei
begin

loader
init
#ifdef MESSAGES
  jsr $f817 ; print PRESS PLAY ON TAPE message
  ldy #00   ; Y is wasted then
#endif
  lda #$1d
  sta $01     ;  motor on
  ror $d011   ; screen blanked to d020
#ifdef CHKSUM
#ifndef RESETMODE
  sty chkbyte ;chose one that's 0 already per default? chkbyte=$10 is 0
after RESET
#endif
#endif
  sty $dd05 ;Y still 0
  lda #$fe  ; timer threshold for TurboTape
  sta $dd04
  jsr   sync    ;-> x=0
  stx   zp3    ; instead of a later lda, sta
  jsr   get_byte    ; important trick! Save time/bytes by using y-indexing
```

```
  tay
initzp
  jsr   get_byte
  sta   zp4,x    ;X still 0
  inx
  cpx #$03
  bne initzp
debug
  jsr sync  ;x=0
  ;---------------------------------------
  ;essential code from here on
e_start
load
  jsr   get_byte
load1
  dec $01   ;to load below i/o area!
  sta   (zp3),y
#ifdef CHKSUM
  eor chkbyte
  sta chkbyte
#endif
  inc $01
  iny
  bne   next
  inc   zp4
next
  cpy   zp1
  lda   zp4
  sbc   zp2
  bcc   load
#ifdef CHKSUM
  jsr get_byte
  cmp chkbyte
  beq ok
    inc $d020 ; color cycling in case of checksum error
    bvc *-3
#endif
ok
  rol $d011 ;screen back on
  lda   #$37  ;default setting
  sta   $01
#ifdef BASICRUN
  jsr $e453 ; prepare BASIC pointers for RUN
  jsr $a660 ;
  jsr $a68e ; RUN
  jmp $a7ae ;
#else
  jmp $080d ;exomizer, or just launch the loaded code directly
#endif
;---------------------------------
get_byte
```

```
  lda   #$01
  sta   bytebuffer    ;init the to-be-read byte with 1
next_bit
  jsr   get_bit
  rol   bytebuffer
  bcc   next_bit    ;is the initial 1 shiftet into carry already?
  lda   bytebuffer    ; much nicer than ldx #8: dex: loop
  rts
;---------------------------------
get_bit
  lda   #$10
bit_loop
  bit   $dc0d
  beq   bit_loop    ;busy loop to detect signal
  lda   $dd0d
  pha
  lda   #$19
  sta   $dd0e
  pla
#ifdef FX
  sta $d020 ;use bitvalue as d020 effect (I love that one)
#endif
  lsr
  rts

e_end
;---------------------------------
sync
  jsr get_bit
  rol bytebuffer
  lda bytebuffer
  cmp #$02
  bne sync
  ldx #$09  ;9,8,... is real turboTape
sync2       ;I sometimes used 8,7,6... to avoid it being listed in vice :)
  jsr get_byte
  cmp #$02
  beq sync2
sync3
  cpx bytebuffer
  bne sync
  jsr get_byte
  dex
  bne sync3
  rts
```