by Peter Dell (‘JAC!')

based on slides by Sven Oliver Moll ('SvOlli')

Programming the Atari 2600 Video Computer System

by Peter Dell (‘JAC!')
based on slides by Sven Oliver Moll ('SvOlli')

Personal background (aka Moore's Law)

Soldered a ZX 81 together in 1984 (1k RAM)

Got an Atari 800 XL in 1986 (64k RAM)

Got an Amiga 500 in 1989 (512k RAM)

Got an Amiga 1200 in 1992 (2MB RAM)

Got a PC in 1995 (8 MB RAM)

Got a Book on the Atari VCS from 1977 in 2009

Got a VCS in 2010 (128 B RAM)

Work in HANA System in 2014 (100+TB RAM)

Personal background (aka Moore's Law)

Soldered a ZX 81 together in 1984 (1k RAM)

Got an Atari 800 XL in 1986 (64k RAM)

Got an Amiga 500 in 1989 (512k RAM)

Got an Amiga 1200 in 1992 (2MB RAM)

Got a PC in 1995 (8 MB RAM)

Got a Book on the Atari VCS from 1977 in 2009

Got a VCS in 2010 (128 B RAM)

Work in HANA System in 2014 (100+TB RAM)

1.000.000.000.000 times

What is it?

(and how could that work?)

Atari’s Rise in Game History

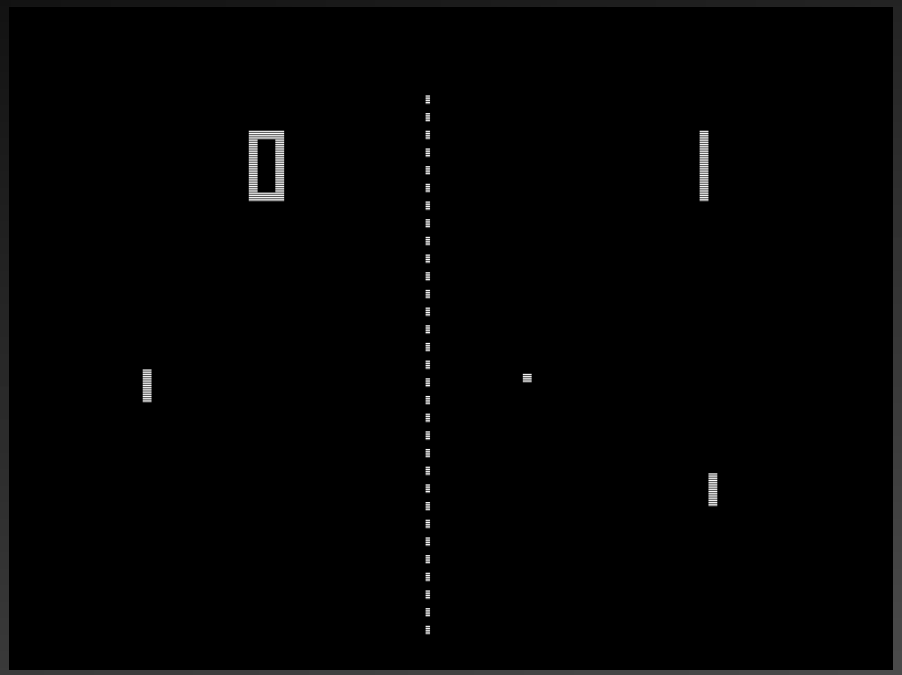
Founded 1972 by Nolan Bushnell and Ted Dabney

"Pong" (1972) and "Tank" (1974) were arcade hits

Atari’s Rise in Game History

Founded 1972 by Nolan Bushnell and Ted Dabney

"Pong" (1972) and "Tank" (1974) were arcade hits



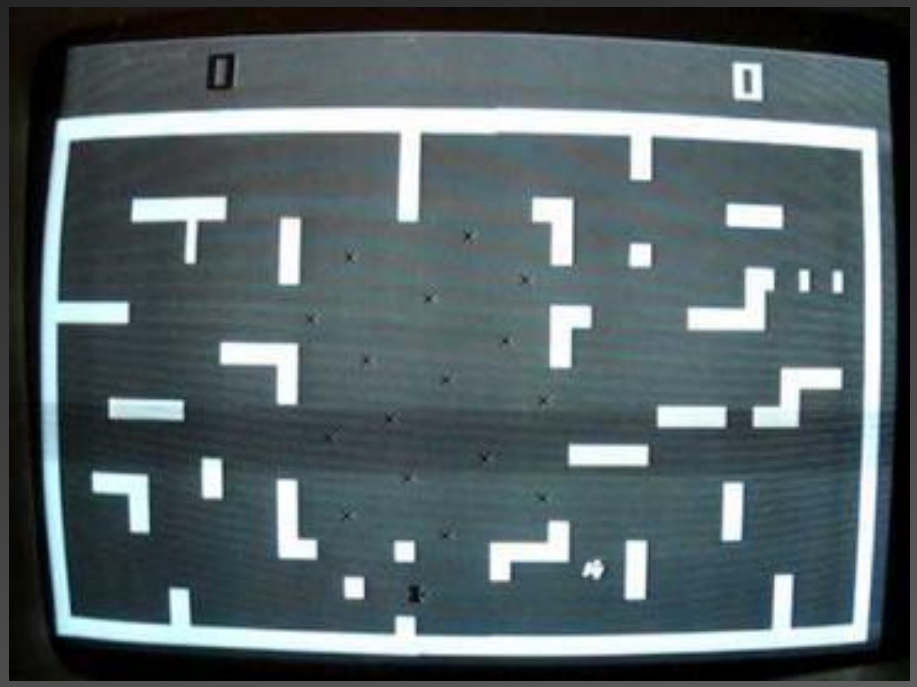


Image courtesy of Wikipedia, public domain

Atari's Rise in Game History

Atari's first home release was "Home Pong"

In 1975, Atari decided to produce a home game console based on a programmable design code named "Stella"

Primary goal: Make lots, lots, lots of money

- Sell lots of consoles (CPU,I/O,TV,RAM)

- Sell many game cartridges per console (ROM)

Primary issue: Cost of console parts ($200 limit)

Primary design principle: Save cost for console

Produced until 1992 (15 years), 30 mio. units sold

Revision Overview (1)

6 switch model, wood design, heavy (1977)

Image courtesy of retrogamescollector.com, used by permission

Revision Overview (2)

6 switch model, wood design (1978, PAL also)

Image courtesy of www.ccmuseum.de, used by permission

Revision Overview (3)

4 switch model, black design (1982)

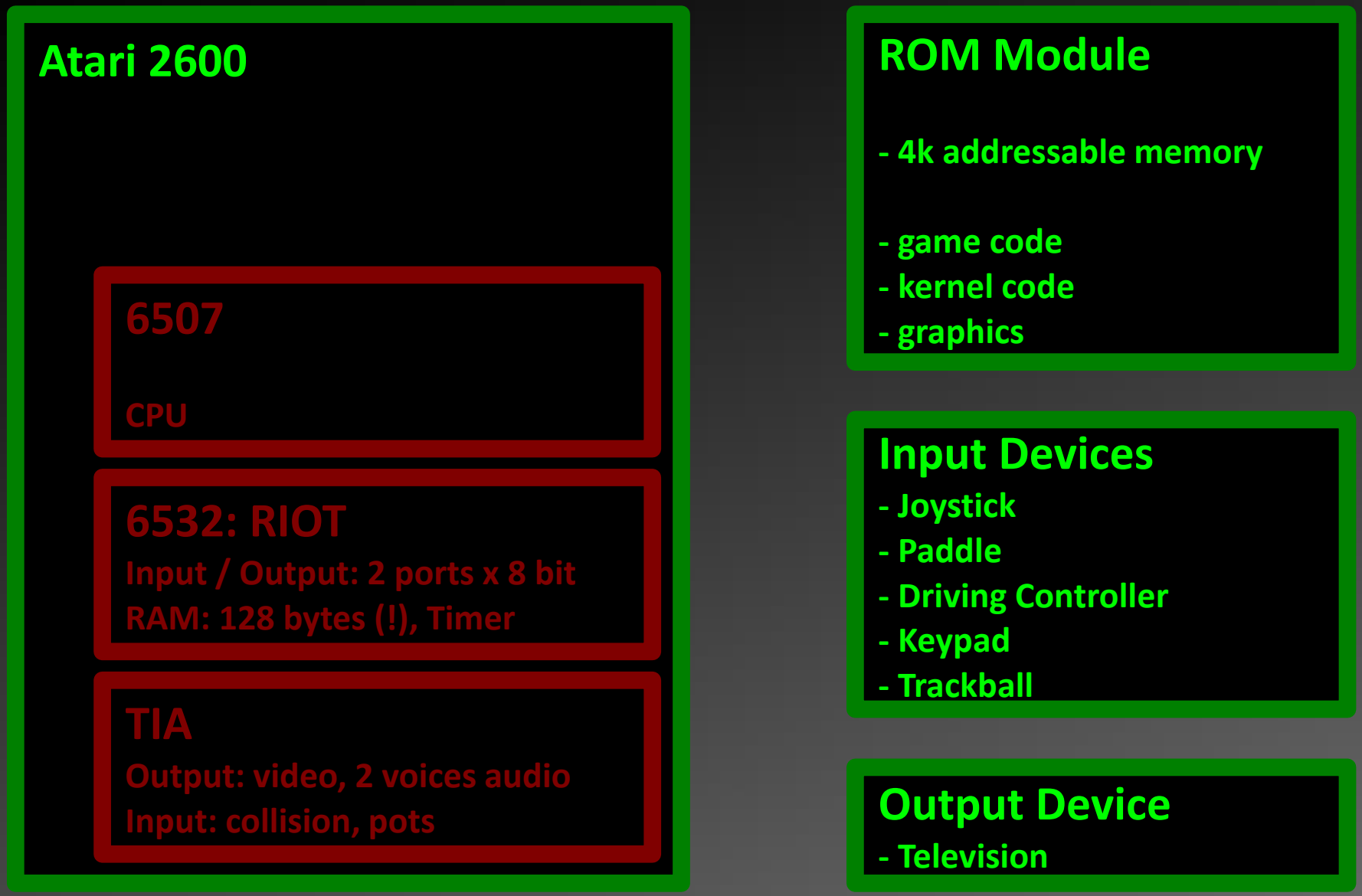
Image courtesy of Ewan-Alan, Wikipedia, public domain

Revision Overview (4)

Atari 2600 Jr (1984)

Image courtesy of Ewan-Alan, Wikipedia, public domain

Hardware block diagram



Design Decisions (1)

- CPU and chipset were off-the-shelf components
  (total $12 vs. Intel/Motorola: $150 - $200)
- 6507 instead of 6502 CPU
  (same CPU, less pins, smaller address space)
- Use 6532 I/O chip for controller because 128 Bytes of RAM and a timer are already included
  (no separate RAM chip required)
- No ROM with OS or charset in console
  (game cartridge is cheaper to produce anyway)

Design Decisions (2)

No frame buffer
(TV screen has ~32k pixels, nobody could pay 32k of RAM and CPU could not address it, too)

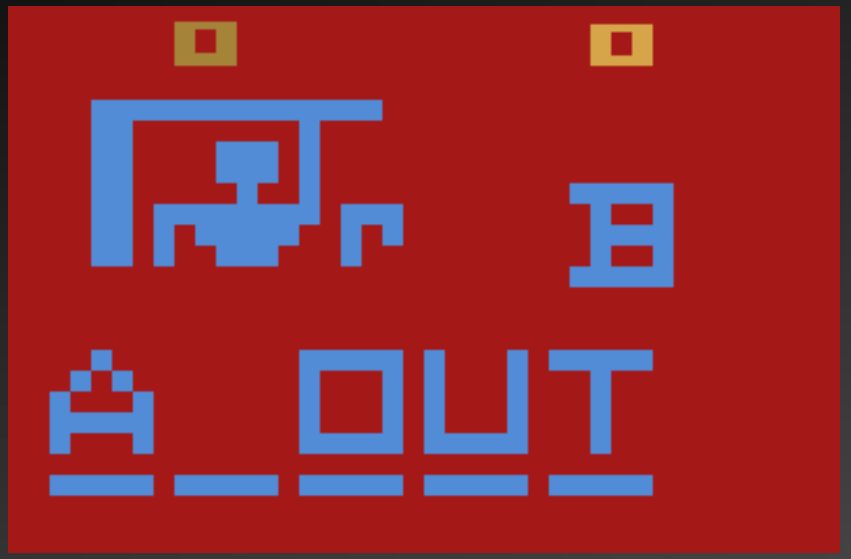
CPU drives display to keep graphics chip simple
(no frame buffer/DMA/IRQ, "Racing the Beam")

CPU creates single line of TV picture at a time

- 21 bits of “screen memory”, 40 pixel playfield
- 1 bit for the “ball”
- 8 bits each for “player 1” and “player 2” (8 pixel)
- 1 bit (pixel) each for “missile 1” and “missile 2”

Game impressions

Basketball (Atari, 1978)



Hangman (Atari, 1978)

Pelé's Soccer (Atari, 1981)

Real Sports Soccer (Atari, 1987)

People learn to use it

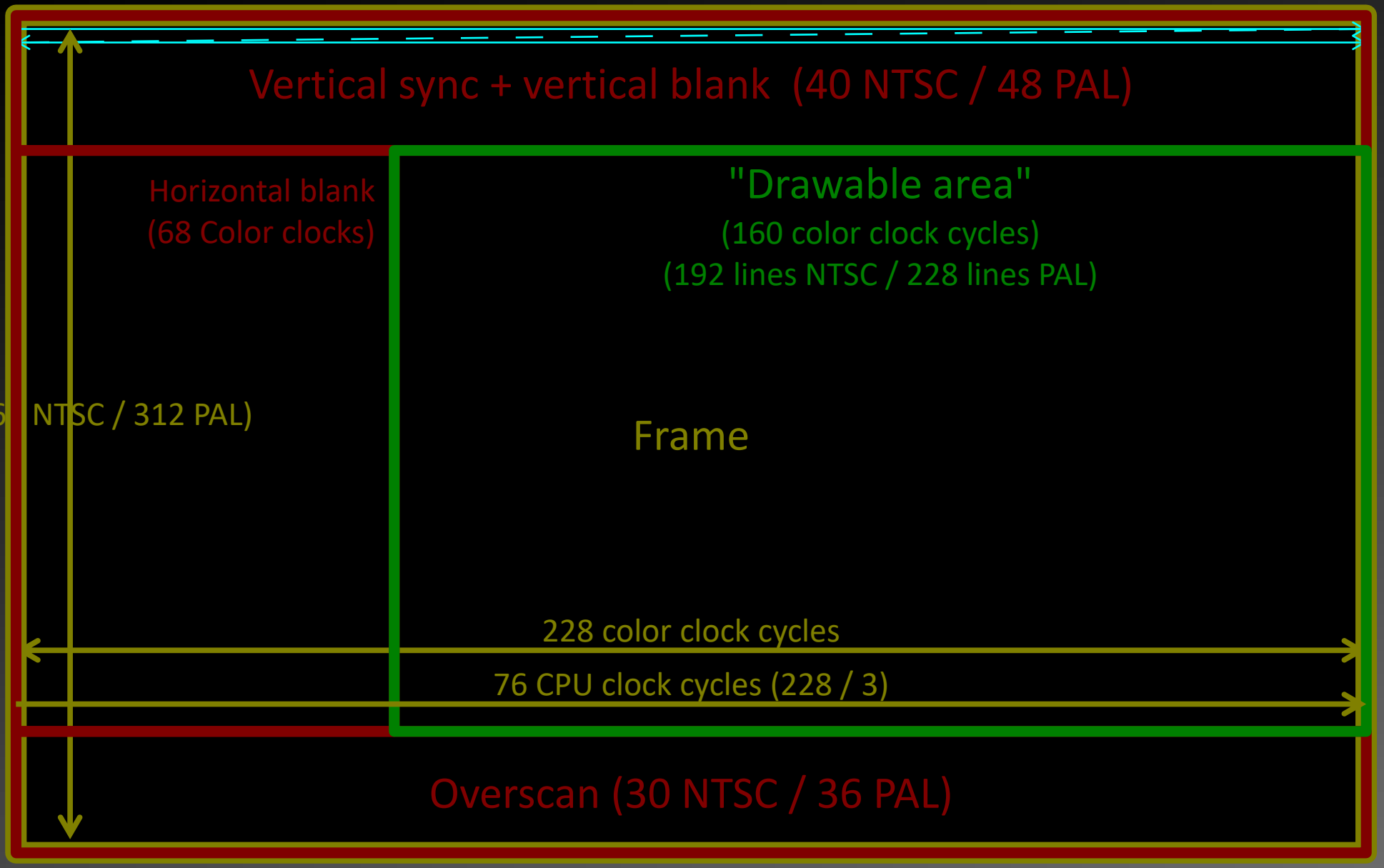
Pitfall (Activision, 1982)

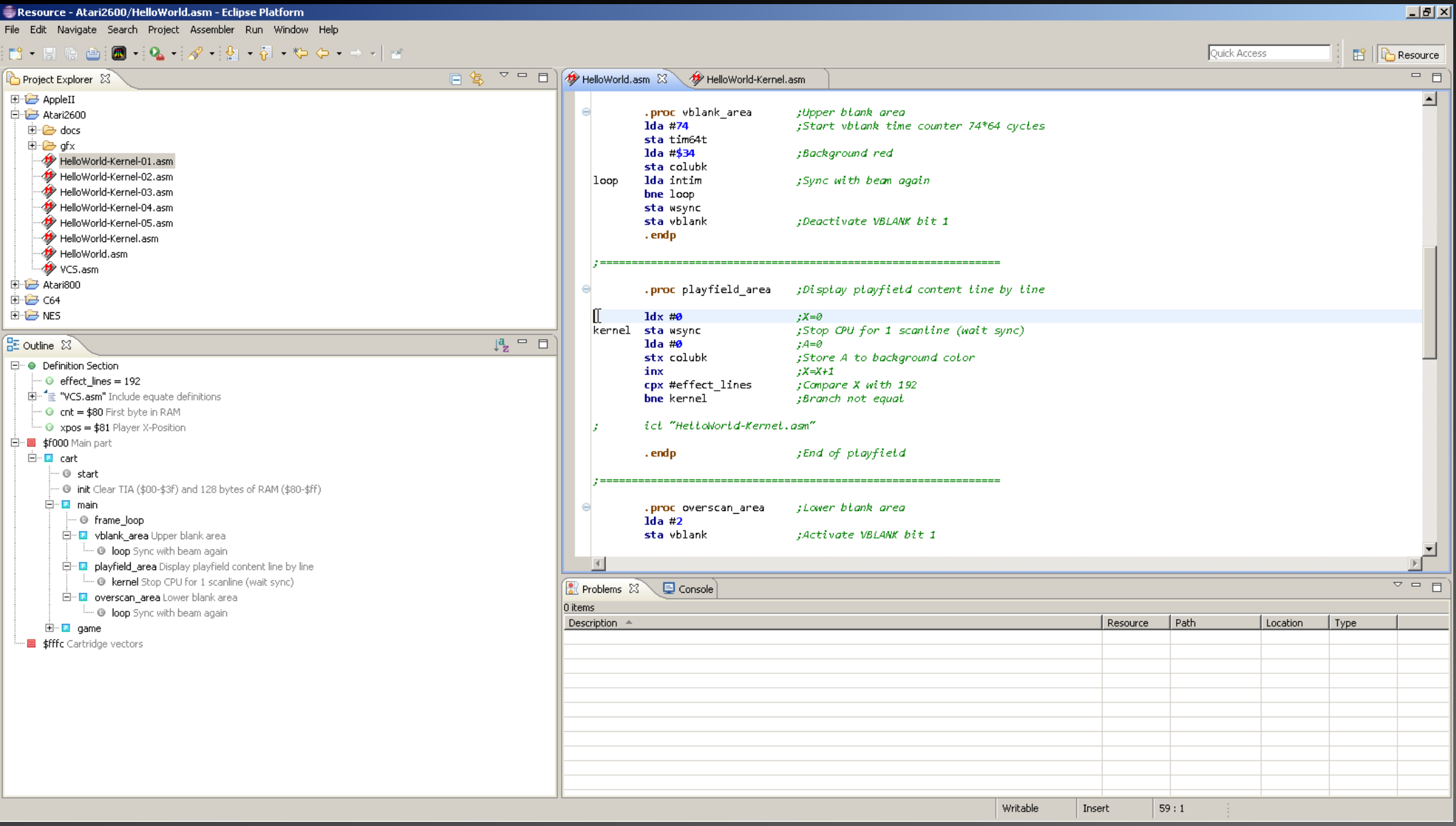
Solaris (Atari, 1986)

Beam Racer Demo (JAC!, 2011)

ISO Demo (JAC!, 2011)

Display


Vertical sync + vertical blank (40 NTSC / 48 PAL)
Horizontal blank
(68 Color clocks)
"Drawable area"
(160 color clock cycles)
(192 lines NTSC / 228 lines PAL)
es (26 NTSC / 312 PAL)
Frame
228 color clock cycles
76 CPU clock cycles (228 / 3)
Overscan (30 NTSC / 36 PAL)


Resource - Atari2600/HelloWorld.asm - Eclipse Platform
File Edit Navigate Search Project Assembler Run Window Help
Quick Access
Resource
Project Explorer
AppleII
Atari2600
docs
gfx
HelloWorld-Kernel-01.asm
HelloWorld-Kernel-02.asm
HelloWorld-Kernel-03.asm
HelloWorld-Kernel-04.asm
HelloWorld-Kernel-05.asm
HelloWorld-Kernel.asm
HelloWorld.asm
VCS.asm
Atari800
C64
NES
Outline
Definition Section
effect_lines = 192
"VCS.asm" Include equate definitions
cnt = $80 First byte in RAM
xpos = $81 Player X-Position
$f000 Main part
cart
start
init Clear TIA ($00-$3f) and 128 bytes of RAM ($80-$ff)
main
frame_loop
vblank_area Upper blank area
loop Sync with beam again
playfield_area Display playfield content line by line
kernel Stop CPU for 1 scanline (wait sync)
overscan_area Lower blank area
loop Sync with beam again
game
$fffc Cartridge vectors
HelloWorld.asm
HelloWorld-Kernel.asm
.proc vblank_area ;Upper blank area
lda #74 ;Start vblank time counter 74*64 cycles
sta tim64t
lda #$34 ;Background red
sta colubk
loop lda intim ;Sync with beam again
bne loop
sta wsync
sta vblank ;Deactivate VBLANK bit 1
.endp
;==========
.proc playfield_area ;Display playfield content line by line
ldx #0 ;X=0
kernel sta wsync ;Stop CPU for 1 scanline (wait sync)
lda #0 ;A=0
stx colubk ;Store A to background color
inx ;X=X+1
cpx #effect_lines ;Compare X with 192
bne kernel ;Branch not equal
; icl "HelloWorld-Kernel.asm"
.endp ;End of playfield
;==========
.proc overscan_area ;Lower blank area
lda #2
sta vblank ;Activate VBLANK bit 1
Problems
Console
0 items
Description
Resource
Path
Location
Type
Writable
Insert
59 : 1

HelloWorld-Kernel.asm

```
;       Change color per line
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        ldx #0                  ;X=0
kernel  sta wsync               ;Stop CPU for 1 scanline (wait sync)
        txa                     ;A=X
        and #1                  ;A=A & 1
        asl                     ;A=A *2
        sta colubk              ;Set background color
        inx                     ;X=X+1
        cpx #effect_lines       ;Compare X with 192
        bne kernel              ;Branch not equal
```

HelloWorld-Kernel.asm

```
;       Change color per line and within line (76 cylces)
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        ldx #0                  ;X=0
kernel  sta wsync               ;Stop CPU for 1 scanline (wait sync)
        txa                     ;A=X
        and #1                  ;A=A & 1
        asl                     ;A=A *2
        stx colubk              ;Set background color to x
:16     nop                     ;Waste 32 cycles = 96 color clocks (pixel)
;       sta colubk              ;Set background color to A
;       stx colubk              ;Set background color to X
;       sta colubk              ;Set background color to A
;       stx colubk              ;Set background color to X
        inx                     ;X=X+1
        cpx #effect_lines       ;Compare X with 192
        bne kernel              ;Branch not equal
```

HelloWorld-Kernel.asm

```
;       Playfield graphics (21 bits / 40 pixel)
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        mva #14 colupf          ;Playfield color = white
        mva #0 ctrlpf           ;Playfield repeat (0)/mirror (1)
        ldx #0                  ;X=0
kernel  sta wsync               ;Stop CPU for 1 scanline (wait sync)
        txa                     ;A=X
        lsr                     ;A=A/2
        lsr                     ;A=A/2
        tay                     ;Y=A
        lda game.graphics+8,y   ;A=graphics[8+Y]
        sta pf1                 ;Store A into playfield 1
        inx                     ;X=X+1
        cpx #effect_lines       ;Compare X with 192
        bne kernel              ;Branch not equal
```

HelloWorld-Kernel.asm

```
;       Player graphics
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        mva cnt colup0          ;Player 0 color flashing

        ldx #0                  ;X=0
:10     nop                     ;Wait 60 pixel
        sta resp0               ;Position player 0

        lda xpos                ;A=xpos
:3      lsr                     ;A=A/8
        sta nusiz0              ;Store are to number & size

kernel  sta wsync               ;Stop CPU for 1 scanline (wait sync)
        lda game.graphics,x     ;A=graphics[X]
        sta grp0                ;Store A into player 0
        inx                     ;X=X+1
        cpx #effect_lines       ;Compare X with 192
        bne kernel              ;Branch not equal
```

HelloWorld-Kernel.asm

```
;       Playfield and player graphics
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        mva #14 colupf          ;Playfield color = white
        mva #1 ctrlpf           ;Playfield repeat (0)/mirror (1)
        mva cnt colup0          ;Player 0 color flashing

        ldx #0                  ;X=0
:10     nop                     ;Wait 60 pixel
        sta resp0               ;Position player 0
        lda xpos                ;A=xpos
:3      lsr                     ;A=A/8
        sta nusiz0              ;Store A to number and size

kernel  sta wsync               ;Stop CPU for 1 scanline (wait sync)
        txa                     ;A=X
        lsr                     ;A=A/2
        lsr                     ;A=A/2
        tay                     ;Y=A
        lda game.graphics+8,y   ;A=graphics[8+Y]
        sta pf1                 ;Store A into playfield 1
        lda game.graphics,x     ;A=graphics[X]
        sta grp0                ;Store A into player 0
        inx                     ;X=X+1
        cpx #effect_lines       ;Compare X with 192
        bne kernel              ;Branch not equal
```

HelloWorld-Kernel.asm

```
;       Sound effects
;
;       @com.wudsn.ide.asm.hardware=ATARI2600
;       @com.wudsn.ide.asm.mainsourcefile=HelloWorld.asm

        mva xpos audc0          ;Divider control
        mva cnt audf0           ;Frequency 0..31

        lda #0                  ;A=0
        bit inpt4               ;Test fire button
        bmi not_pressed         ;Branch
        lda #12                 ;A=12
not_pressed
        sta audv0               ;Volume 0..15
```

Famous last words…

Original Atari documentation from 1979

3.0 RAM

The PIA has 128 bytes of RAM located in the Stella memory map from HEX address 80 to FF. The microprocessor stack is normally located from FF on down, and variables are normally located from 80 on up (hoping the two never meet).

Get the free IDE and start coding yourself…

http://www.wudsn.com