# Let Me Show You Something

# How Tinkering With A 65C02 Led To Building A Computer

by Sven Oliver ('SvOlli') Moll

# Motivation

- Talk about first home computers in two months
- Focus is on 6502 based machines
- Reading about using a Raspberry Pi Pico / RP2040 to interface with classic machines → I want todo something like this, too
- Maybe working with a 65C02 first hand would be fun?

# Alternatives: Retro Machines

- Apple 1 clones:
  - Board replicas with original chips
  - Replica 1
  - Mega6502
- KIM 1 clones:
  - Board replicas replacing the 6530 RRIOTs with 6532 RIOTs
  - KIM UNO
- PET clones, TIM 1 recreation

# Alternatives: Machines In Active Development

- Commander X16
- Steckschwein
- Ben Eater's 65C02 on a breadboard
- Pico Computer
- 65uino
- SBCs by Western Design Center
  - W65C02SBX, W65C134SBX, W65C816SBX, W65C265SBX, etc.

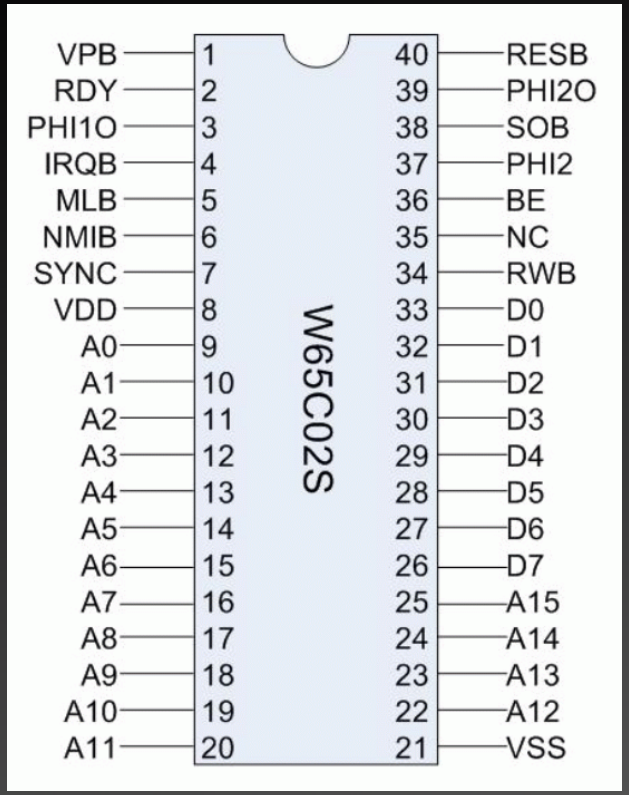
# Alternatives: Downsides

- Most are not available at the moment
- Expensive
- Most are kits → big effort to set up
- Still a bit "too high-level" for showing off how the CPUs works
- No machine fits the idea of tinkering
- → How about building an own computer?

# Requirements

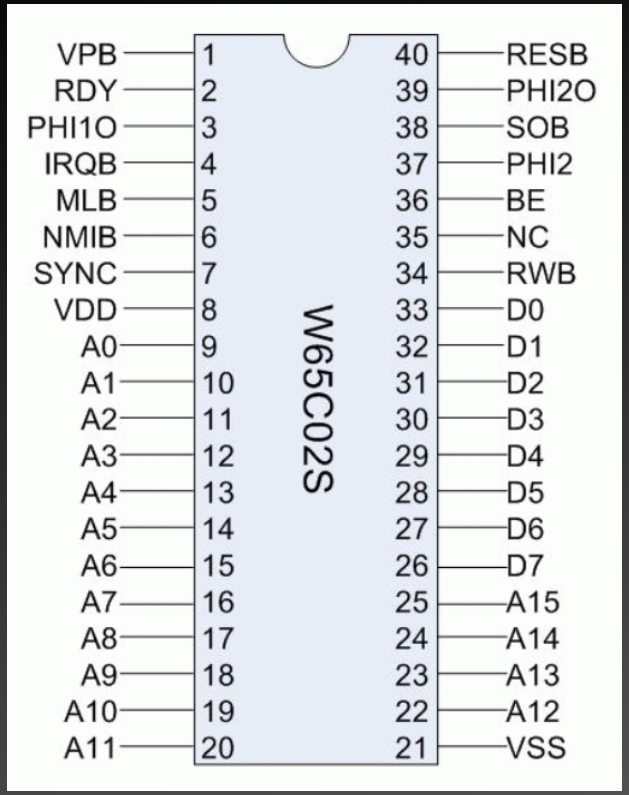
- More a toy than a full featured computer
- Quick: two months
- Learn about 6502 → show CPU in action
- Keep it simple, stupid → easy to solder
- Open source
- Expandable / modular
- Milestones: run like old computer, run diagnostic monitor
- Cheap: ~~€10~~ → €15

# 65C02 (1)




- Required pins:
  - VSS (GND), VDD (VCC)
  - A0-A15, D0-D7
  - RWB
  - PHI2
  - IRQB / NMIB / RESB
  - RDY
- 32 pins

# 65C02 (2)




- Not required pins:
  - NC, PHI1O, PHI2O
  - SOB: Set Overflow
  - VPB: Vector Pull
  - BE: Bus Enable
  - MLB: Memory Lock
  - SYNC: Sync opcode fetch
- Some are even recommended not to use

# Raspberry Pi Pico

- Cheap versatile microcontroller board
- Available for ~$4
- 26 GPIO pins
- 2MB flash
- USB type micro-B connector
- Optional WiFi
- Driven by RP2040 microcontroller

# RP2040 microcontroller

- Cheap versatile microcontroller
- Available for ~$1
- 30 GPIO pins
- → 4 GPIOs are "held back" for LED or WiFi

# Chinese Clone: "Purple Board"

- Same microcontroller
- Available for ~$2.50
- 30 GPIO pins
- 16M flash
- USB type C connector
- Different pinout
- Also: different power supply

# Architecture (1)

- Just connect 65C02 to RP2040 board
- Power supplied by RP2040 board
- Main interface: USB UART
- How to enable tinkering / expanding?
  → expose bus
    → like in a 1970s computer?

# Architecture (2)

- Passive backplane to connect everything
- 65C02 on one card
- RP2040 based board on the other card
- Provide more then two slots for other cards
- → and just call it a computer

# Which Machine To Implement?

JOLT Computer (1975-12)

- RAM: 576 bytes
- ROM: 1024 bytes
- I/O: PIA 6820
- I/O: RRIOT 6530-004
- UI: terminal

MOS KIM-1 (1976-Q2)

- RAM: 1152 bytes
- ROM: 2048 bytes
- I/O: RRIOT 6530-002
- I/O: RRIOT 6530-003
- UI: terminal
  + 6 digit display
  + keypad

Apple Computer 1 (1976-09)

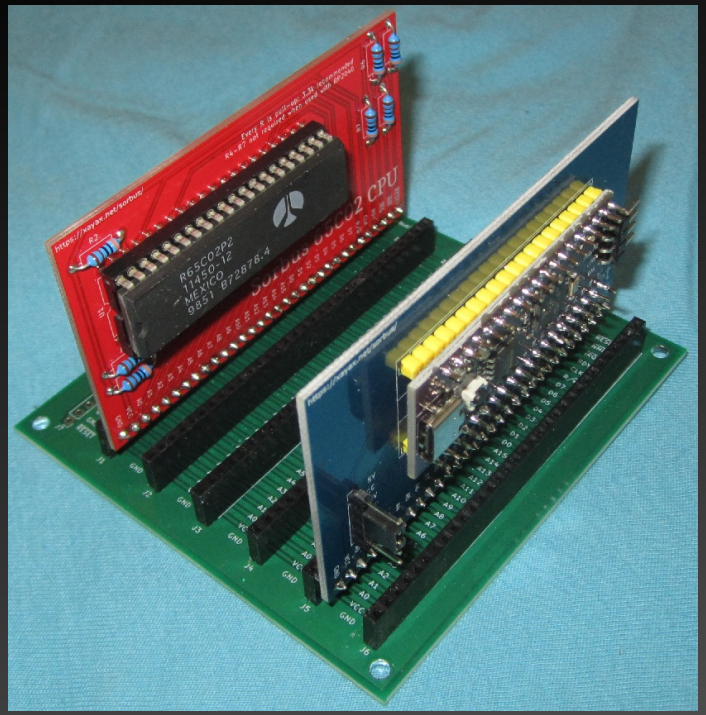
- RAM: 4096 bytes
- ROM: 256 bytes
- I/O: PIA 6820
- UI: terminal simulating CRT
  + keyboard

# Sorbus Computer

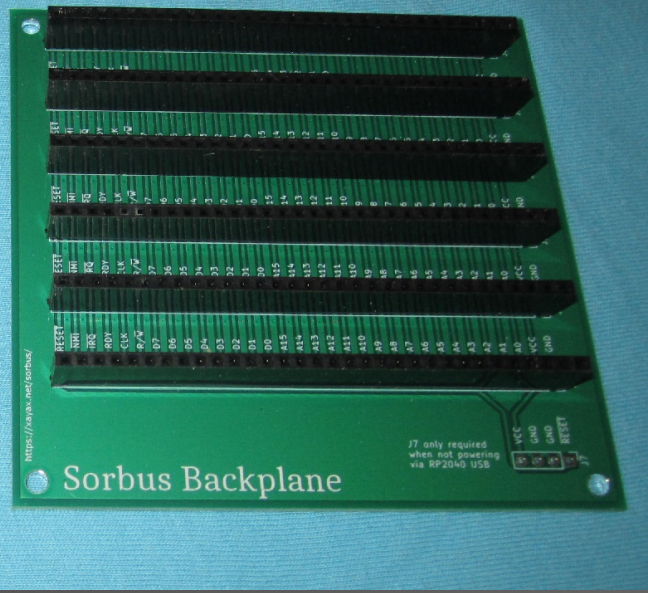
- Pick a name → some kind of fruit
- Is uses a Raspberry to run like an Apple
- Sorbus (German: "Vogelbeere")
- Mother always told me: "Don't eat"
  → so don't use it for primary purpose
- But... as a kid playing with those berries... I had a lot of fun

Image: Martin Olsson / Wikipedia

# Images


R65C02P2
11450-12
MEXICO
9851 B72878.4


Sorbus RP2040 Chipset
Sorbus 65C02 CPU
R65C02P2
11450-12
MEXICO
9851 B72878.4


Sorbus Backplane

# Also A System To Teach

- MCP:
  Monitor Command Prompt
- Loosely based on monitor of the JOLT Computer and also monitor software for later machines like C64
- Run at 1Hz to 0.1MHz

- Available commands
  - help: display help
  - freq: set frequency
  - cpu: show cpu type
  - reset, irq, nmi
  - s: run number of steps
  - m: dump memory
  - f: fill memory
  - :: write to memory (colon)

# Timing Can Be Tricky

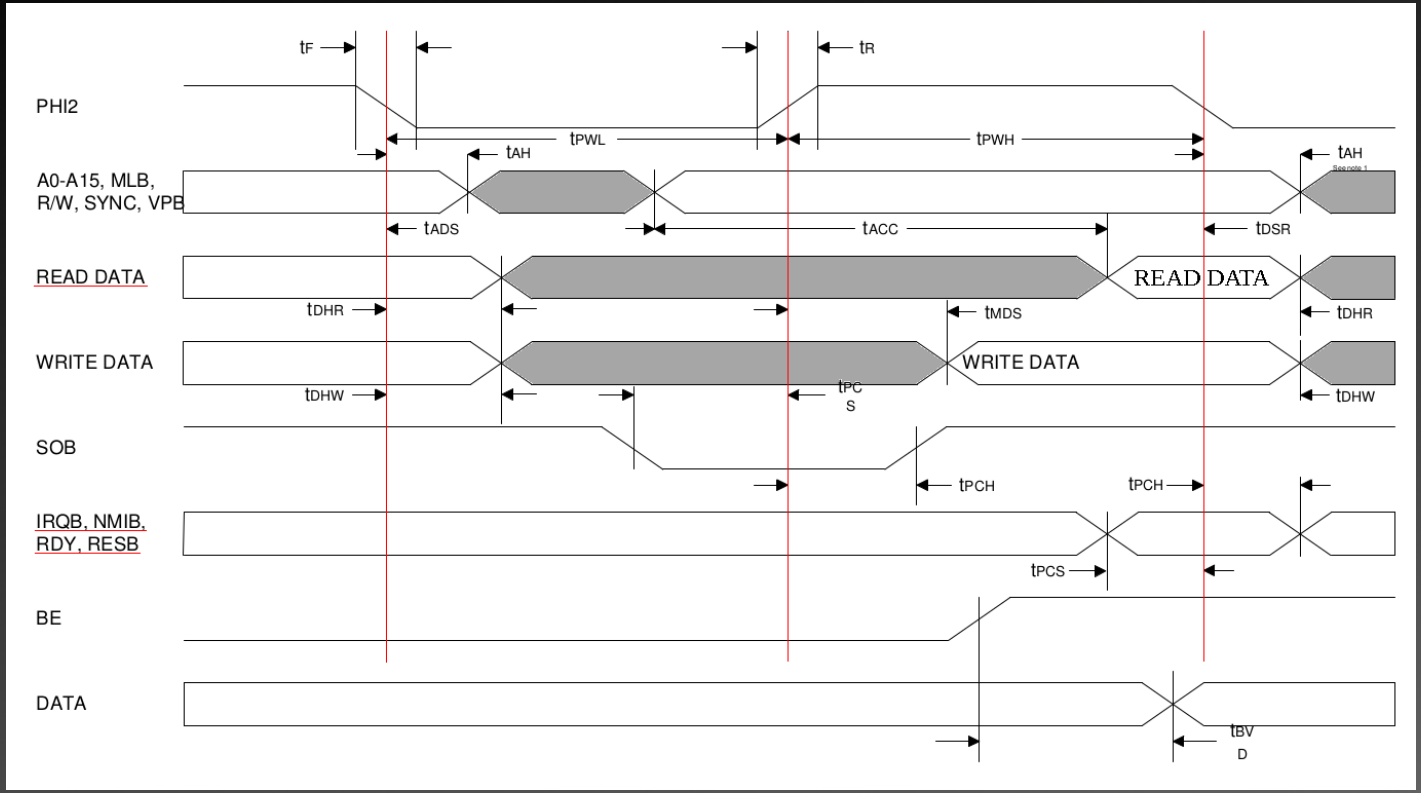



Diagram taken from WDC W65C02 datasheet, modified for readablity

# Still A Lot That Can Be Done

- Improve the MCP with disassembler and direct-assembler
- Implement (Super) JOLT Computer
- Implement KIM-1 → design the hex input/output PCB
- Implement 65816 "chipset" for > 64k RAM
- Create a PCB for a 6532
- Create a PCB for a General Instruments AY-3-8910 (aka Yamaha YM2149F) soundchip (or similar)

# Thanks & Disclaimer

- Thanks
  - My dad for soldering all 90 boards with over 12000 solder joints
  - Jedi for looking over my PCB designs
- Disclaimer
  - I created this computer
    … to have a toy
  - It's open source so that you can build your own
    … and improve it
  - However: I will not produce and sell them in a larger scale
    … except for the ones available for this talk / workshop

# Lessons

- Here a few things to do
- Just type in the commands after the ">" prompt
- After that, let's discuss, what we see there

# Output Explained

- 100:0433 r fe R  >
- "100:": number of steps left to execute
- "0433" current value of address bus
- "r": state of read/write line (either 'r' or 'w')
- "fe": current value on data bus
- "R  ": Reset line low (active) (can also be "N"MI or "I"RQ)
- ">": command starts after this prompt

# Lesson 1: Reset And NOP

- > f 0000 FFFF EA
- > reset 5
- > s 20

# Lesson 2: Another Type Of NOP

- > f 0000 FFFF 33
- > reset 5
- > s 20

# Lesson 3: A Simple Program

- Power off an on again
- > f 1000 1100 00
- > : FFFC 00 04
- > reset 5
- > s 345
- > freq 1000
- > freq 5
- > m 1000

# Lesson 4: Subroutine

- > : FFFC 03 04
- > reset 5
- > s 35

# Lesson 4: Data On Stack

- > : FFFC 06 04
- > reset 5
- > s 30

# Lesson 6: Interrupt Via IRQ Line

- > : FFFC 09 04
- > reset 5
- > s 30
- > irq 9
- > s 35

# Lesson 7: Interrupt Via BRK Instruction

- > : FFFC 0C 04
- > reset 5
- > s 60

# Lesson 8: Learn How To Use An Apple Computer 1

- Insert USB cable with BOOTSEL button pressed
  → device now appears as mass storage
- Copy "apple1.uf2" to "RPI-RP2"
  → the device will automatically reboot
- Run terminal software again, "`" key is reset
- Let's explore WozMon
- Let's do a "Hello world" in BASIC