
Fit for teaching!

# OMEN Bravo

## Want to be a system programmer ?

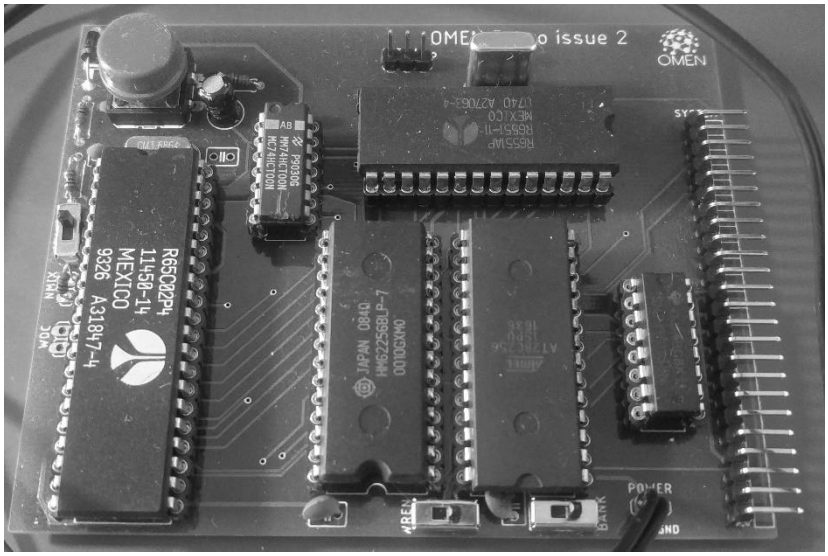
The 6502 CPU is still alive! This legendary CPU, empowering a lot of great computers of 70's and 80's, like the Apple II, The Beeb or the best selled 8bit computer, the Commodore C64, is the heart of this super simple computer, called OMEN Bravo.

So why not to try program this great CPU nowadays? Just 5 ICs to make a great single board computer. Just buy Bravo, build it - and ENTER!


OMEN

OMEN Bravo / issue 4

Technical Documentation




## INTRODUCTION

==========

The OMEN Bravo computer kit is a low-cost computer trainer, based on the 65C02 CPU. It has these features:

- 65C02 CPU working at 3.6864 MHz
- 32 kB RAM
- 8 kB EEPROM
- Serial port up to 19.200 Bd / MOS 6551 ACIA
- Application system bus


ASSEMBLY INSTRUCTIONS

==========

1. Solder sockets for the integrated circuits
2. Test all soldered connections
    a. Test if all pins are well connected
    b. Check if GND is not short connected to Vcc
    c. Check if each IC has properly connected GND and Vcc
3. Solder all passive parts /capacitors, diode, resistors, push button, crystals/ - pay attention, the Bravo contains two crystals, one for CPU, marked Q1 with value 3.6864, one for serial port, marked "1.8432"
4. Connect the power adapter and check
5. Insert the CPU into its socket /keep the proper orientation!/ and try to power it up. Check if oscillator lives /at CPU pin 39/
6. Insert the essential ICs: 7400, 74138, 62256, 6551 and AT28C64. Again: keep the proper orientation! Bad orientation can damage the IC!
7. Connect the serial pins TxD, RxD and GND /pinhead JP1/ to the TTL-to-USB converter
8. Start the serial terminal on your PC, select proper serial port and set the parameters to 19.200 Bd, 8 data bits, no parity, 1 stop bit.
9. Power your Bravo and check the terminal.


MONITOR

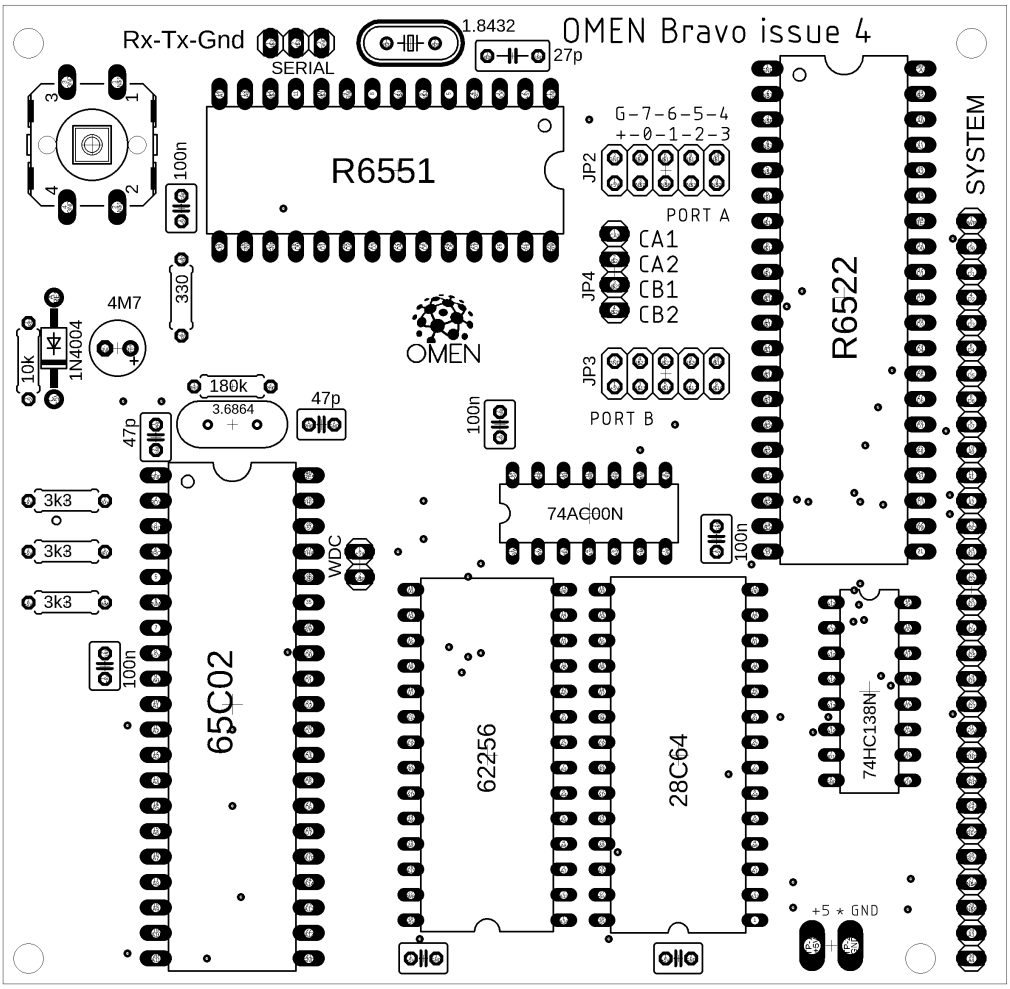
=======

The main software for OMEN Bravo issue 4 is the freeware C'Mon Serial Monitor by Bruce Clark.

You can use the OSI Basic with a minimal effort, as well as the lot of other software.

THE BOARD

========




Jumpers and pin headers

--------------------

SERIAL: Serial port. Pins are RxD, TxD, GND from left to right.

WDC: For Rockwell 65C02 leave open, connect for WDCs W65C02S.


## SYSTEM APPLICATION CONNECTOR

==========

This connector is on the right edge of board. Pin 1 is on the upper side, next to the SYSTEM label.

Pins:

| | | |
|---|---|---|
| 1 | /WR | |
| 2 | D0 | |
| 3 | D1 | |
| 4 | D2 | |
| 5 | D3 | |
| 6 | D4 | |
| 7 | D5 | |
| 8 | D6 | |
| 9 | D7 | |
| 10 | A0 | |
| 11 | A1 | |
| 12 | A2 | |
| 13 | /RD | |
| 14 | IO1 | --- 9000h - 93FFh used for VIA 6522 parallel interface |
| 15 | IO2 | --- 8800h - 8BFFh |
| 16 | IO3 | --- 9800h - 9BFFh |
| 17 | IO4 | --- 8400h - 87FFh IOx signals are decoded by 74138 |
| 18 | IO5 | --- 9400h - 97FFh |
| 19 | IO6 | --- 8C00h - 8FFFh |
| 20 | IO7 | --- 9C00h - 9FFFh |
| 21 | Vcc | |
| 22 | GND | |
| 23 | /IRQ | |
| 24 | PHI2 | |
| 25 | /RES | |
| 26 | RDY | |
| 27 | N/C | |
| 28 | N/C | |
| 29 | A3 | |
| 30 | A4 | |


## PERIPHERAL IC ADDRESSES

### ACIA 6551:

83FCh - Data Register

83FDh - Status Register

83FEh - Command Register

83FFh - Controll Register

### VIA 6522:

93F0h - ORB/IRB

93F1h - ORA/IRA

93F2h - DDRB

93F3h - DDRA

93F4h - T1C-L

93F5h - T1C-H

93F6h - T1L-L

93F7h - T1L-H

93F8h - T2C-L

93F9h - T2C-H

93FAh - SR

93FBh - ACR

93FCh - PCR

93FDh - IFR

93FEh - IER

93FFh - ORA/IRA


## MEMORY MAP
==========

0000h - 7FFFh - System RAM 32k

8000h - BFFFh - I/O space, see above

C000h - DFFFh - EEPROM 8k shadow

E000h - FFFFh - EEPROM 8k


C’Mon the Compact Monitor

by Bruce Clark

C'mon, the Compact monitor, is a very small monitor program for the 65C02.

Commands

---

cr - output a CRLF (and a dash prompt)

0-9A-F - accumulate hex digit

, - store 16-bit "byte" at address, increment address

@ - set address

G - go (call routine)

X - hex/ASCII dump 128 bytes

: - enter the HEX line to store into memory

All other characters are ignored (thus you can use spaces freely). Leading zeros can be omitted. Note that commands execute immediately, i.e. you will get a memory dump when you press X not when you press Enter.

Any routines you call with the G command should return with the D flag clear (they will be called with the D flag clear).


Examples

--------

1234 X - dump 128 bytes starting at address $1234

1234 G - call the routine at address $1234

1234 @ 11,22,33, - store $11, $22 and $33 at address $1234

Single step feature

-------------------

To single step, use the @ command to set the address, then use the $ (single step) command to single step that instruction. Use $ again to single step the next instruction. After each single step, the program counter (of the next instruction) and the A, S, X, Y, P (in hex), and P (again, but in binary) registers will be output.

If you set the break vector to BREAK, it will save the registers and set the address so that the next instruction (after the signature byte) can be single stepped. Example:

```
    ORG $2000

    CLC

    LDA #$1234

    ADC #$5678

    BRK

    .byte $0000 ; signature byte (its value is ignored)

    EOR #$ABCD
```


2000G can be used to execute up to the ADC instruction. A subsequent $ command will begin single stepping at the EOR instruction.

Since it isn't always convenient to use BRK, it is also possible to use JSR BRKPT instead of BRK (and its signature byte). Everything about the preceding example would be the same, except that the code would be:

```
    ORG $2000

    CLC

    LDA #$1234

    ADC #$5678

    JSR BRKPT

    EOR #$ABCD
```