“Apple 2 Next”

Written by Paul Robson 11-12th- July 2023 (Revision 2)

This is a first draft specification of the “Apple II Next” the working (but unlikely to be final) name for the retro computer build on Olimex’s Neo6502 board.

The idea is conceptually the same as the Mega65 and Spectrum Next machines. The system is a more modern version of the original, but can run the software of its parent in compatibility mode.

Processor Hardware

The processor is a 65C02 rated at up to 8Mhz, connected to 64k of RAM memory, which is stored in the Raspberry PI Pico.

As with the standard machine, there is RAM memory from 0000-BFFF, and hardware i/o is between C000 and CFFF.

D000-FFFF contains 12k of “ROM” (RAM in practice) – Applesoft BASIC and the Monitor, and a further 16k of RAM bank switched as per Appendix D of the Apple Language Card Manual (using address range $C08X). The Autostart ROM also supercedes the upper 2k of this space replacing the default monitor.

Compatibility mode largely is identical save for the CPU speed. It would be nice to maximise the processor speed of the 65C02 but in compatibility mode this should run at 1.023Mhz

Keyboard

The keyboard generates ASCII keys and a strobe to $C000 which can be reset via $C010 software switch (ref Little pp 160)

Audio

The Apple II speaker toggles between high and low whenever location $C030 is accessed, so the pitch can be calculated from the cycles between.

Extension

The ability to simply specify the pitch of the sound by writing a value to <location>, which overrides the Apple behaviour. Off would be specified by value zero.

Display

The Apple II display is somewhat interesting and is described in detail in *Little*. However, in practice there are 3 modes, each of which has 2 areas of memory that can store the text/graphic data

- a 40x24 text display with upper case text (6 bit ASCII) that supports inverse and flashing using the other 2 bits.
- A 40x48 low res colour graphics display – each character is divided into an upper and lower half which defines the colour of that half.
- A 280x192 6 colour high res colour graphics display where pixels are encoded in 7 bits of 8 (the MSB is a colour selector) and operate in pairs.

Additionally the lower 32 scan lines can be used as 4 rows of 40 characters, rendered in the same way as the text display.

These are controlled by soft switches at $C050 … $C057.

As the screen itself is 640 and 480 pixels across and down, a 280 x 192 pixel graphics across would give a small border in the display as per the Vic 20 or Commodore 64 but much smaller.

Extension

The graphics system is extended with various “Modes”, the exact nature to be decided. Modes 0,1 and 2 would be the text, low-res and high-res modes as described previously, compatible with the Apple][

The system would have a user writeable base address register which would state where video memory started.

When a write occurs to any of $C050/1 (controlling text/graphics mode), $C054/5, (controlling which page is visible), $C052/3 (controlling the lower 4 lines being text/graphics) and $C056/7 (controlling lowres/highres mode) this would override both the mode setting (e.g. it would be set to 0,1 or 2) and the base page selection which would be set as per *Little.*

Any mode set using the alternative method detailed later can start at any address in memory the user desires.

Alternative modes

There are various options for modes. There is sufficient space for 256 of them, which should allow extension beyond the basics. Examples may include :

Pixels per line	Could be 280 (same as Apple][) or 320 (full screen width, or even 640 memory/RP2040 permitting. Possibly as low as 160.
Colours per line	2, 4, 16 or 256 colours per pixel could be used.
Palettes	For the first 2 a palletteised system should be used, or even multiple modes with the same resolution accessing different palettes.
Vertical Resolution	As a memory saver, vertical resolution could be halved (e.g. lines are repeated), thus allowing a 160 x 120 x 256 colour mode for example.
Apple II Modes	The high res Apple][modes or modes with a not dissimilar design could be used.
Normal text mode.	An Apple II text mode where flashing is replaced by lower case.

Disk System

@OliverSchmidt commented

“The Apple][+ firmware is extensible. The most important extension controls a mass storage device. Such a device offers an interface to read/write 512 byte blocks to/from memory. It's trivial to create a firmware implementing that interface that just hands the block number in question and the memory location in question to the RP2040 which performs the operation”

The slight downside with this is the Flash chip in the Neo6502 I think only erases 4k sectors. Against that people may only use this system in read-only mode (Game saving perhaps ?)

Memory Interface

Technically the memory space is all used, though looking at the extensions while the Apple][decodes for (say $C00X) software uses $C000 as $C001 is used in later machines.

The area $C080-$C0FF is available for hardware expansion. $C08x is used for the Language card.

Below is a suggested allocation.

$C0F0-$C0F1	Base address of graphics display
$C0F2	Set mode on write
$C0F3	Bit 7 use extended memory for display (1) use CPU memory (0)
$C0F4	Set CPU speed bit 7 : full speed (0) 1.023Mhz (1)
$C0F5	Set Sound frequency to <constant>/byte value, 0 = off

Software

I wouldn’t suggest we have what one would normally call a ROM memory. More that the open source firmware is loaded in at start up.

Through some method ; a BASIC command might be the simplest, providing a copy of the Applesoft ROM Image/Language ROM image is available (which I don’t think we could distribute !) the ROM images could be loaded into the upper 12k of RAM space, and the system rebooted.

References:

- Apple 2 Reference Manual
- Apple Language Card : Installation and Operation Manual.
- Inside the Apple Iic by Gary B Little.

Revisions:

- 12-July-23 : Revision following various suggestions.