How to build
a real Kernal Cartridge

by Thomas 'skoe' Giesel

Version 0.2 – work in progress
not for distribution

1 Introduction

1.1 What a Kernal Cartridge is needed for

The Kernal contains the operating system of the C64, for example the drivers used to access the serial bus and therefore the disk drives. Often users want to replace the slow original disk drive routines with faster ones. But there may also be other reasons why it is wanted to replace the Kernal or parts of it. If these functions are to be replaced by alternative implementations, the best approach is to replace the Kernal.

There are also other ways to replace these functions, for example by changing jump vectors in RAM which are used by the Kernal. But this mechanism has its limits: Only functions which are called through these vectors can be replaced. Also one has to be aware that these substitutions have to reside somewhere in the C64's address space, which potentially collides with memory requirements of other software.

This background makes clear why cartridges like Action Replay or The Final Cartridge III make a good job, but cannot accelerate all disk operations: They do not replace the Kernal.

One well-known Kernal replacement is Jiffy DOS. To use Jiffy DOS, the Kernal of the host computer (e.g. C64) and the operating system of each disk drive have to be replaced. Drives made by CMD and the sd2iec/uIEC come with Jiffy support as standard.

Most disk drives have their operating system in a ROM which is plugged into a socket, so it can be swapped easily. Unfortunately most C64s have their Kernal ROM soldered on the PCB, which makes it difficult to replace it.

The simplest solution for the users is a cartridge which actually substitutes the internal Kernal and behaves exactly like it from the software's perspective.

1.2 The Challenge to make a Kernal Cartridge

The Kernal can be found in the C64 address space at $E000 to $FFFF, it is 8 KiB in size. But the Kernal is not the only memory in this address space. There is also RAM below the Kernal. Programs can decide if they want to read the Kernal ROM or the RAM there by setting the HIRAM bit in the I/O register $01 of the 6510 CPU. Fig 1 shows the memory map of the C64 with no cartridge attached.

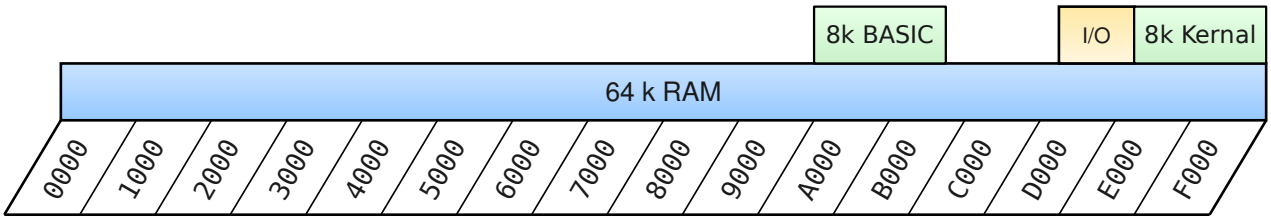

Fig 1: Memory map of the C64 with no cartridge attached

Write accesses from the CPU are always done to the RAM below the Kernal. The VIC-II always reads from that RAM but not from the Kernal ROM. But CPU read accesses depend from the signal

/HIRAM, which is set in in bit 1 of the I/O register $01. If /HIRAM is low (0), RAM is read. If it is high (1), Kernal ROM is read. Fig 2 shows this mechanism.

Remark: This is valid for C64 mode. In Ultimax mode there's a different behavior which is not shown here.

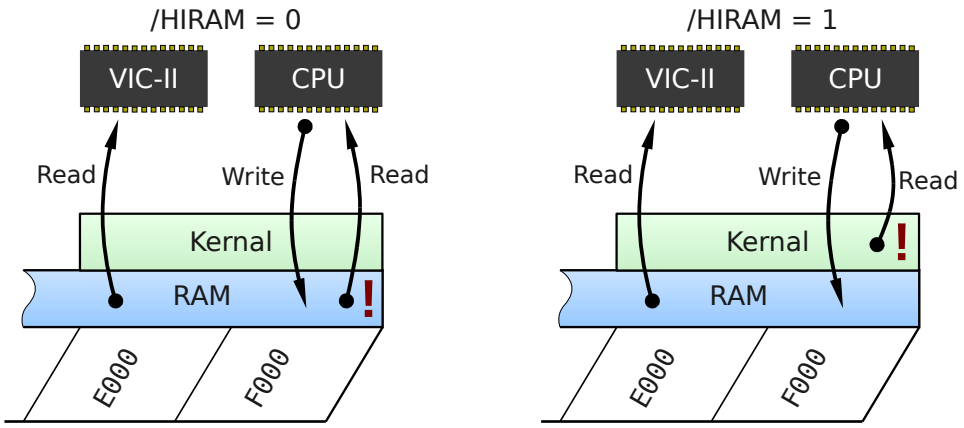

Fig 2: HIRAM controls access to Kernal memory space

A Kernal cartridge which is fully compatible to the internal Kernal must implement this behavior. Otherwise programs which use $01 to hide the Kernal ROM will fail, because they cannot use the RAM between $E000 and $FFFF. The problem is that the /HIRAM signal is not available on the Expansion Port. That is why many Kernal cartridges either cannot provide access to this RAM area or have to use a wire which must be connected to the /HIRAM signal inside the C64.

For a fully compatible Kernal cartridge with no additional wire, another way must be used to find out the state of /HIRAM. One obvious idea is to sniff the bus traffic and keep track on write accesses to the address $0001. Unfortunately this does not work, since these accesses are kept inside the 6510 CPU and are not visible on the bus.

Not only the 6510 I/O port can change the memory configuration, but also two lines which are connected to the expansion port: /GAME and /EXROM.

These two lines are pulled up with resistors inside the C64. This means when there is no cartridge attached both of them are high (1). When a cartridge is attached it can pull one or both of these lines low (0) to change the memory configuration. When a cartridge is attached, the memory configuration can still be configured with the 6510 I/O port, e.g. to hide the cartridge ROM.

There are two more control lines on the Expansion Port[1]: /ROML and /ROMH. Whenever the C64 – the address decoder in the PLA to be exact – decides that data from a cartridge must be read, it pulls down one of these lines. Traditionally they are connected directly to two 8 KiB ROM ICs in the cartridge.

An example: A cartridge pulls down /GAME and /EXROM to tell the C64 that there is a 16 KiB cartridge attached. The program which is running has not disabled any ROM (i.e. /HIRAM = 1 and /LORAM = 1). The CPU wants to read a byte at $A123. The address decoder in the PLA finds out that this byte must be read from the high ROM chip of the cartridge. That's why it pulls down /ROMH.

1 In fact there are even more, but these two are interesting now.

Table 1 shows the configurations which can be set by a cartridge using /GAME and /EXROM.

/GAME - /EXROM	Meaning
1 – 1	No cartridge attached Memory map unchanged
1 – 0	8 KiB cartridge /ROML mapped to $8000..$9FFF
0 – 0	16 KiB cartridge /ROML mapped to $8000..$9FFF, /ROMH mapped to $A000..$BFFF
0 – 1	Compatibility mode to Commodore MAX, so-called Ultimax mode /ROML mapped to $8000..$9FFF, /ROMH mapped to $E000..$FFFF

Table 1: Configurations of /GAME and /EXROM

In the *Commodore 64 Programmer's Reference Guide* several different memory configurations are described. Some interesting cases are shown in Table 2. In columns (1), (2) and (3) no cartridge is attached, i.e. /GAME and /EXROM are 1. Only /HIRAM controls whether there is RAM or Kernal ROM visible to the CPU at $E000 in this these cases.

There is also a memory area controlled by HIRAM which results in a control signal visible at the Expansion Port: When a 16 KiB cartridge is attached to the Expansion Port, /HIRAM is used to select whether the /ROMH part of the cartridge or the internal RAM is mapped to $A000. This is shown in columns (4), (5) and (6).

This is a valuable indicator to find out the state of HIRAM. Whenever a 16 KiB cartridge is attached to the Expansion Port and an address in the range of $A000 to $BFFF is on the address bus, the PLA sets the signal /ROMH directly depending from HIRAM. As /ROMH is connected to the Expansion Port it can be captured by the cartridge.

	/LORAM 1 /HIRAM 1 /GAME 1 /EXROM 1 (1)	/LORAM x /HIRAM 0 /GAME 1 /EXROM 1 (2)	/LORAM 0 /HIRAM 1 /GAME 1 /EXROM 1 (3)	/LORAM x /HIRAM 1 /GAME 0 /EXROM 0 (4)	/LORAM 0 /HIRAM 0 /GAME 0 /EXROM 0 (5)	/LORAM 1 /HIRAM 0 /GAME 0 /EXROM 0 (6)	/LORAM x /HIRAM x /GAME 0 /EXROM 1 (7)
E000-FFFF	Kernal	RAM	Kernal	Kernal	Kernal	RAM	ROMH
A000-BFFF	BASIC	RAM	RAM	ROMH	RAM	RAM	---

Table 2: Memory configurations

Now we got the first things a Kernal cartridge has to do:

- Set /GAME and /EXROM high (memory map: no cartridge attached)
- Wait until Phi2 is high (CPU cycle) and address bus is stable
- If it is a read access and the address is in Kernal space $E000..$FFFF:
  - How can we find out HIRAM now?

There's still an open point: The Kernal cartridge wants to know the state of HIRAM when the CPU reads from the Kernal address range of $E000 to $FFFF, but /ROMH is only set as described above for addresses in the range of $A000 to $BFFF.

For this reason the Kernal cartridge must manipulate the address bus in some way. When the CPU wants to read something, it sets its address outputs to the address to be read. This address cannot be altered by a cartridge normally, because it would fight against the CPU's pad drivers and possibly destroy them.

The 6510 CPU has a control input called *Address Enable Control* (AEC). If this input is low, the CPU turns off its address and data line drivers. This is exactly what we need, because then we can change the address bus to $A000 to $BFFF.

Unfortunately this line is not connected to the Expansion Port directly. But it is driven by the /DMA signal, which is available on the Expansion Port. /DMA has the downside that it also changes the state of another CPU control line: *Ready* (RDY). With this line the CPU can be halted. By the way that is the line which is also controlled by the VIC-II when it needs to fetch additional data which results in the so-called bad lines.

A look into the 6500 CPU family data-sheet reveals some details about RDY. Good news for the Kernal cartridge: If RDY is asserted during a Phi2 cycle and released before the cycle ends, it is ignored and does not stop the CPU. This also means that this line must be released before the Phi2 cycle ends if the CPU must not halt.

More about the timings of the mechanisms described here will be discussed in chapter 1.3.

This mechanism disturbs the procedure which normally take place when the CPU reads from the internal DRAM from the C64. When HIRAM is set and the CPU really wants to read RAM below the Kernal, the VIC-II is responsible to latch the upper and lower part of the address bus to the multiplexed address lines of the DRAM chips. As the external Kernal cartridge changes the address bus during the Phi2 phase the VIC-II will latch wrong addresses to the DRAM.

This means that the cartridge must also be able to put the right data from the RAM below the Kernal on the data bus. Therefore the cartridge simply mirrors the content of these 8 KiB of RAM in its own memory. This can be done very easily: Every write access to this address space is also written to the mirror RAM. If the cartridge finds out that the RAM below the Kernal is read by the CPU, it can send data from the mirror RAM. The VIC-II may also want to read from this space to fetch display data. The cartridge lets it read the internal DRAM in the C64 because it does only act on CPU read accesses.

With these findings we can add some more steps to our cookbook:

- Set /GAME and /EXROM high (memory map: no cartridge attached)
- Wait until Phi2 is high (CPU cycle) and address bus is stable
- If it is a read access and the address is in Kernal space $E000..$FFFF:
  - Assert the /DMA line, wait
  - Set the address bus to any address in the range 0xA000..0xBFFF, set /GAME and /EXROM to 0 to act like a 16 KiB cartridge, wait
  - Look at /ROMH to see whether ROM or RAM is selected by HIRAM, release /DMA to make the CPU set the real address on the bus again
    How to send the right byte to the CPU?

Finally we have to make sure the PLA selects no other chip but the cartridge and put the right byte onto the data bus. With another look at the memory maps we can find configuration (7) in Table 2. This is the so-called Ultimax mode. In this mode the /ROMH memory of the cartridge is selected

for any address in the Kernal address space. To get in this mode the cartridge has to continue to pull down /GAME (0) and to release /EXROM (1).

When the cartridge is finally selected with /ROMH, it can put the byte from the external Kernal or from the external RAM mirror on the bus. Which one is used depends from the state of HIRAM determined in the step ago. When the Phi2 cycle ends, the cartridge releases all lines and returns to idle state.

- Set /GAME and /EXROM high (memory map: no cartridge attached)
- Wait until Phi2 is high (CPU cycle) and address bus is stable
- If it is a read access and the address is in Kernal space $E000..$FFFF:
  - Assert the /DMA line, wait
  - Set the address bus to any address in the range 0xA000..0xBFFF, set /GAME and /EXROM to 0 to act like a 16 KiB cartridge, wait
  - Look at /ROMH to see whether ROM or RAM is selected by HIRAM, release /DMA to make the CPU set the real address on the bus again
  - Release /EXROM to go to Ultimax mode
  - When /ROMH is asserted, put the data byte from Kernal ROM or mirror RAM on the bus
  - Release all lines after Phi2 cycle has ended

1.3 Timing Considerations

Some of the steps needed for the external Kernal implementation have to consider timing requirements of the components of the C64. Most of them can be found in data-sheets.

- Wait until Phi2 is high and address bus is stable

When Phi2 goes high the VIC sets AEC high when it wants to allow a CPU cycle. After this signal reached the CPU, it takes up to T_{AES} = 75 ns according to the 6510 data-sheet until the address bus shows the final value. However, measurements done on different types of C64s have shown that the time between the rising edge of Phi2 and a stable address bus is significantly smaller. Examples are shown in Fig 3 and Fig 4.

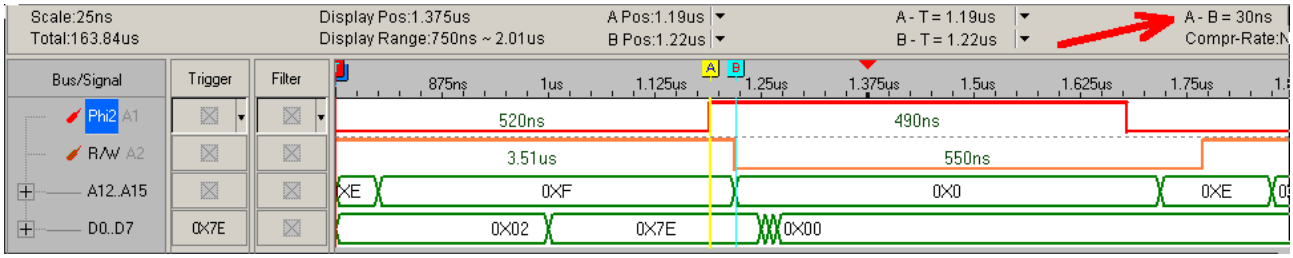

Fig 3: Address Bus after H-Edge of Phi2, Board 250407

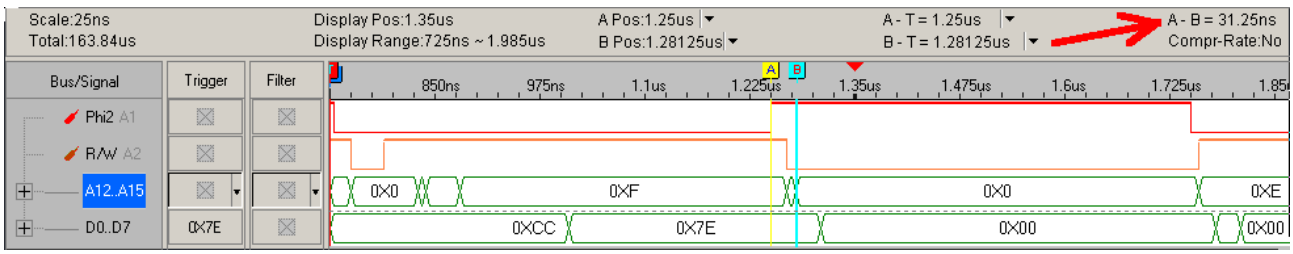

Fig 4: Address Bus after H-Edge of Phi2, Board 250469

To be on the safe side the external Kernal Cartridge will sample the address bus at t >= 80 ns, where t = 0 is the rising edge of Phi2.

- Assert the /DMA line, wait

When the Kernal cartridge pulls down /DMA at t >= 80 ns, the CPU will tristate its address output. The 6510 data-sheet states that this takes up to T_{AED} = 120 ns. We will modify the address bus after this time.

- Look at /ROMH to see whether ROM or RAM is selected by HIRAM

The data-sheet of the Signetics 82S100 PLA specifies a typical propagation delay T_{PD} of 35 ns and a maximum delay of 80 ns. The Sharp custom IC used in newer C64s is most probably as least as fast as the old PLAs.

We will read the result 80 ns after having set all input conditions. This is also the time when we release /DMA to allow the CPU to restore the address bus.

Less than T_{AES} = 75 ns later the address lines are stable again, the flash or RAM memory chip can select the right data byte. As soon as the PLA pulls down /ROMH again. /OE of the right memory chip is pulled down and the cartridge can put the data byte onto the bus. When a 55 ns flash is used, these delays will be together less than about 130 ns.

The remaining time is the Data Stability Time Period (T_{DSU} min 100 ns) for the 6510 data lines.

All times to measured by the cartridge can be implemeted as multiple of 40 ns. Therefore a CPLD with a 25 MHz clock may be used to build the cartridge. As the C64 signals come from another clock domain, the accuracy is also only 40 ns. Fig 5 shows the expected timing.

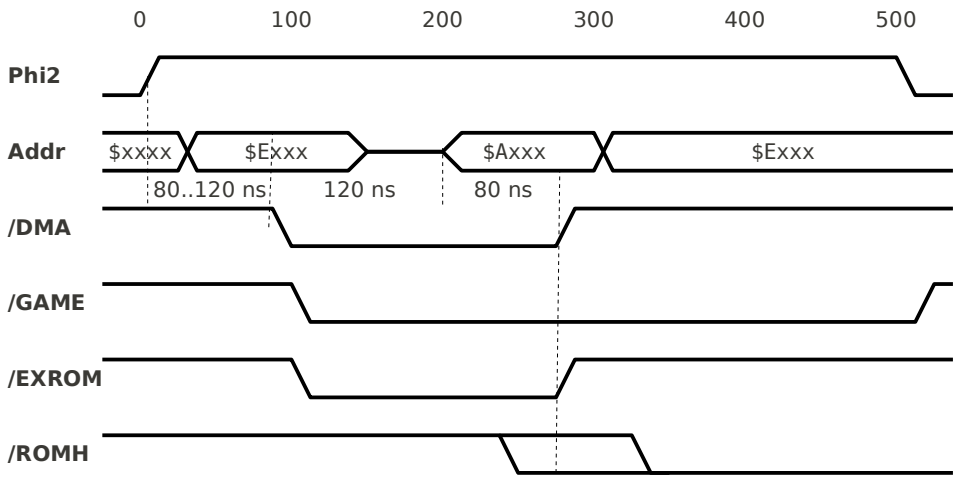

Fig 5: Timing of the External Kernal Cartridge

Timing limits have been taken from following data-sheets:

http://archive.6502.org/datasheets/mos_6500_mpu_nov_1985.pdf

http://matthieu.benoit.free.fr/cross/data_sheets/82S100.pdf

http://www.nxp.com/documents/data_sheet/PLS100X.pdf

1.4 First Results and Next Steps

A first prototype of the external Kernal cartridge has been implemented but still with incorrect timing. It was tested with several different C64 versions and worked as expected.

The description above shows that the PLA has to perform additional state transistions. Nevertheless the prototype did not cause it to get a higher temperature than without a cartridge or with traditional cartridges.

The next step will be to design a hardware which will act under the timing constrains explained in this document. It will have to be tested on different C64 revisions.