Cloning the 98228A ROM for the 9825T

This all started when I purchased a 9825T, I had not intended to buy a 9825, but this 9825T came up for sale at the same time as I had some extra cash from the sale of some other equipment, so I bought it. Once I received it and go it working, the next consideration was mass storage. The tape drive actually works in the one I got and the tire on the capstan did not disintegrate when I tried to use it, however I found it clumsy storing files on the tape, and I don't have any really good tape cartridges anyway, so I started looking at support for diskette drives. I discovered right away that the most common ROM for diskette support only supported the 9885 drive and there was a much rarer ROM that supported both the 9885 and 9895. Well I have what is functionally equivalent to a 9895 but did not have a 9885, so I knew that the 98228A ROM was the ROM I needed or I would need to clone a 9885. I have since purchased a 9885M and I am waiting for delivery of it, I already have the GPIO module for attaching a 9885.

Late summer 2016 people are preparing for the annual HHC meeting and a gentleman I correspond with mentions to me that fellow MoHPC member David Ramsey was bringing a 9825 with a 8" diskette drive to the HHC meeting. I right away sent an email off to David asking which 8" diskette unit he had and mentioned my search for the illusive 98228A. He sends me an email back "Oh, you mean this ROM? " with a picture of a 98228A. I emailed him back saying I would very much like to borrow it and promised to be careful and he replied that he could loan it to me after the HHC meeting.

While I waited for the ROM to become available I studied the little bit of information about the memory map that is in the service guide and also the schematic of the 98228A that Tony Duell created and I had a good idea how it worked and proceeded to create a setup using GPIO cards in a 9920 system to simulate the memory bus. I already had some ROM modules for 0925A and 9831 so I proceeded to use them to test my setup while I waited for the 98228A to be delivered.

I also made up a cable to extend the memory bus outside my 9825T with a connector that I could conveniently connect logic analyzer probes to. This cable was designed so that I could plug the ROM board into the end so that the ROMs would see the same interface. I also used the ROM board from my 9825T on the end of my 9920 based bus simulator.

When the 98228A arrived I plugged it into the 9825T and monitored the operation of the ROM using my logic analyzer. What I observed confirmed what I had figured out about the way the bank selection for this ROM worked, so now I was all set to dump the ROM so I connected the 9825 ROM board to my 9920 and proceeded to use the programs I had previously to dump the 8 pages from the ROM. The program that I used saved the words of data from the ROM in ACSII text format, so I then transferred the files to a Linux machines where I used a PERL script to convert the ASCII text to binary format. I then proceeded to dump all of the ROMs in my 9825T both OS and accessory ROMs. Now I was faced with verifying the ROM images. I knew of Ansgar Kuckes Utility to check 9835/45 ROM images and thought that I would give it a try and found that when used with 1K block size it did in fact appear to work, 'romcheck -plain -blocksize 1024 *filename* ', but the first page of the 98228A image failed and that made me nervous.

First prototype and trouble shooting.

My first prototype was wire wrapped on a prototype board that would not fit inside the 9825T but I could plug it into my buds extension which proved to be handy. Once I finished wiring it I plugged it in and it did not work. Carefully reviewing it I found that I had made a wiring error and once I corrected that, I was happy to find that the cat command now worked. I then opened the manual and started to work my way thought the commands and it was going good until I got to the data file functions none of them appeared to work properly and I thought back to the failed checksum. The logic analyzer I have has a function to de-multiplex a bus, I had never used it before but it seemed easy enough so I set it up and traced the operation of one of the failing command, and then compared it to a trace done with the original 98228A and found a single bit error that changed an instruction and caused some commands to end prematurely. I corrected the bit in the ROM image and rechecked the checksum and now it passed, which made me feel better. I burned a new EPROM and went back checking the commands and was now able to get though all the commands related to 9895, at that time I was unable to check 9885 specific commands, however I am now waiting for a 9885 and when I get it and assuming it is functional I will check 9885 operation and update this document. I also tried redumping the first page of the 98228A on my 9920 a few times but that one bit was always wrong, yet the ROM works correctly when installed in a 9825T.

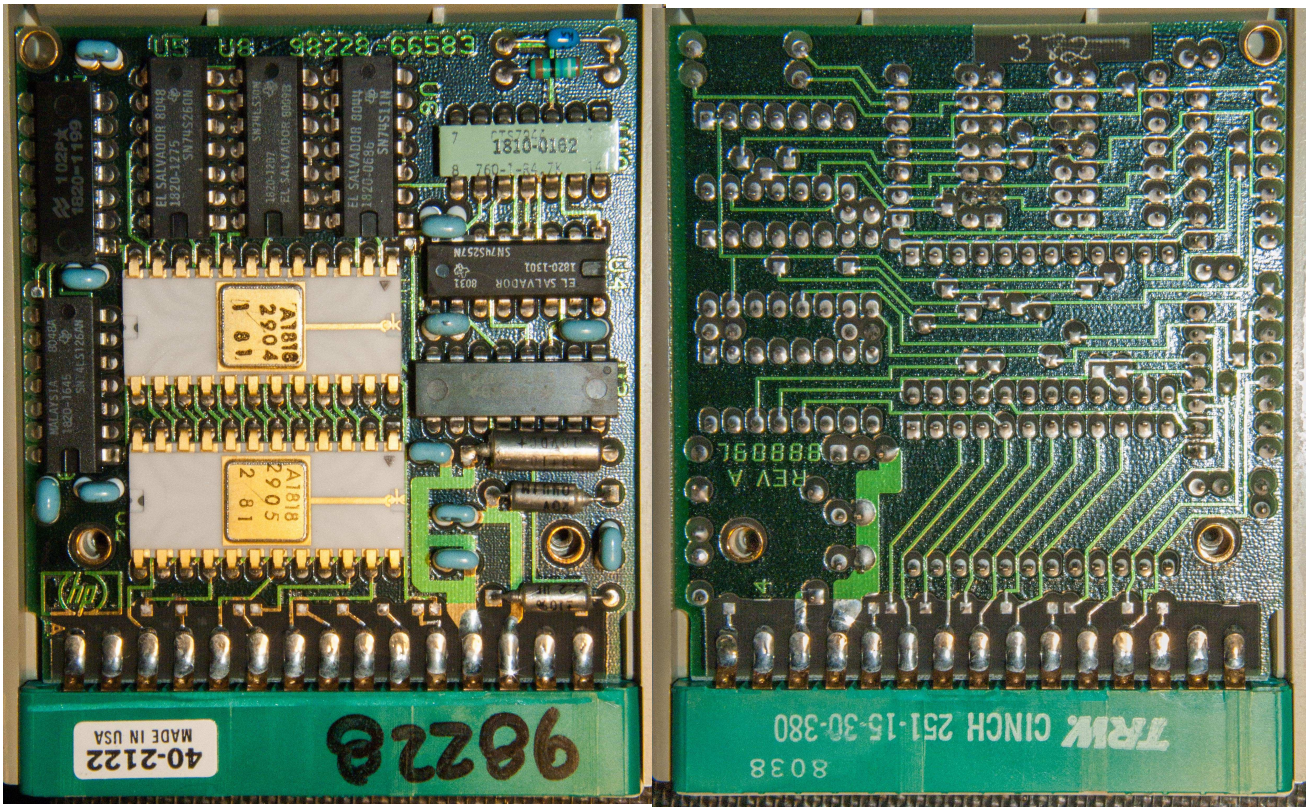
Second Prototype

In the 9825T there are transceiver chips on the ROM board that are driving the ROM slots. These transceivers have their Vcc connected to 7.5V and when they are driving the ROM slots the logic swing is from 0V to almost Vcc. The outputs from the EPROM in my first prototype where connected directly to the ROM bus and as such, when the CPU is outputting the address, the outputs of the EPROM are exposed to this 7.5V logic swing. The spec sheets for the EPROMs said the maximum voltage on any pin is Vcc (5V) +1. This made me wonder if in the long term this could lead to failure of the EPROM, even though when the pins are exposed to the higher voltage they are in the high impedance state, the spec sheet does not say it is ok to expose them to more than Vcc+1 at any time. TTL on the other hand is more robust so I decided to insert 74LS245 transceivers to isolate the EPROM from ROM bus. I modified my first prototype to add the 74LS245s and found that it worked fine.

The next step was to fit it all into a ROM shell. When I got my 9825T it came with a couple of ROM modules that are of no use in a 9825T since the equivalent ROM are built into the system, and I also found that one of the ROM modules had a defective chip, so its shell became the donor for my 98228A clone. The space inside the shell is pretty tight plus the posts for two of the fasteners that hold it closed, obstruct a bit of the board. The board is not centred in the shell, there is more space below the board that above so it works best if you build it component side down. There is also a ridge running down the centre of the bottom that means there is about 2mm less clearance in the centre than on the sides. I wanted to socket the EPROM, and a small PLD but clearance is a big issue. What I ended up doing was locating these components on the side where there is more clearance, and I put pins from a machined pin socket into enlarged holes to get the chips as low as possible while still in sockets, this was sufficient to fit the parts into the shell. I wired this by hand using wire wrap wire, and soldered connections, and about half way through I started thinking there was

a better way, but I did finish wiring it in this manner. What would have been easier would have been to wire it using some wire I have that has solder through insulation. I don't know if you can even get this wire any more but it eliminates cutting, and stripping wires it makes wiring prototypes a lot faster. The down side is the wire is a bit fragile, but in this case since I was fitting the end result into a plastic shell that would not have mattered.

How the ROM works



Both side of the original 98228A ROM.

There was some discussion on the classic computers list concerning the 98228A ROM, and the fact that it was bank selected, and appeared to be tied to the 9825T's state machine that allows for the overlap of RAM and ROM, it turns out it does because one area of the memory map that the ROM occupies falls into the area that would normally be RAM, so it is dependant on the state machine to differentiate between RAM and ROM. The bank selection mechanism is completely independent, and contained within the ROM module. Looking at the memory map shown in the 9825 service guide for the 9825B with option 201, which is a 9825T, we see that there are two slots for Mass Storage ROM one is at 3000_{16} or 30000_8 with the note saying it is reserved for part of 98228A or all of 98217A, and then again at the top of the ROM map at $5C00_{16}$ or 56000_8 there is another area reserved for part of the 98228A. This upper area is in an area that overlaps with RAM, and is therefore dependant on the state machine to operate. Another space to make note of is the area marked for Basepage ROM,

this is the only area in ROM that is not overlapped by RAM, and this is important to the operation of the ROM.

The ROM cartridge contains 8K words of ROM in two of HP's custom ROMs that are designed to operate directly from a multiplexed sixteen bit bus. As well as the ROM array, these ROMs also contain address decoding circuitry that is likely mask programmed the same way as the ROM array, so they will only respond to the range of addresses they are programmed for. The 8K words of ROM are accessed in eight 1K word chunks selected by bank selection circuitry built into the ROM cartridge.

The Bank selection works by writing a value between 0 and 7 to an address that falls within the space of this Basepage ROM, in fact the addresses are chosen so that the 3 least significant bits will be equal to the bank number in the ROM desired. Since this are is solely occupied by ROM, the write operations are ignored by the ROM. It turns out that when the ROM is addressed in the 1K word window at 3000_{16} you will always see bank 0 of the ROM however when addressed in the window at $5C00_{16}$ you will see the currently selected bank.

Refer to the schematic of the original ROM in the package with this document. U3 is the bank select register, this register clocks in the 3 least significant bits of the Memory/Address Bus (MAD) on the falling edge of ALE when the -WR signal is low and the output of U8 is low. U8 is the address decoder for U3, and the output will be low when MAD14, MAD13, MAD12, MAD11, and MAD10 are all low and MAD5 is high all other bits are don't care. This puts the address into the range of 20_{16} to $3FF_{16}$ which is the space occupied by the basepage ROM. The ALE signal goes through gates U6B and U6C and there is a 196 ohm resistor between the output of U6B and input pins 9 & 10 of U6C as well as a small capacitor between those pins and the output of U6C and the output of U6C is also pulled up by 2 4.7K ohm resistors in parallel. I am not sure what the purpose of the resistor R1 and capacitor C1 are but I did include them in my first prototype but dropped them in later version and it had no ill effect. The output of U6C is then inverted by U7E because U3 latches on the rising edge of its clock signal which is the falling edge of ALE. From what I observed using the logic analyzer at the time when ALE drops, the address is still on the bus. U4 is a 4 way 2 to 1 multiplexor that selects between inputs of all zeros or the output from the bank register, with the exception of the third "C" multiplexor which has its inputs connected to the outputs of of U5A and U6A. U4-Za drives A10, Zb drives A11, Zc drives A14 and Zd drives A13. U6A output will be high when MAD10, MAD11, and MAD14 are all low which would be in the 3000_{16} range. The output U5A will be high when MAD10, MAD11 and MAD14 are high which would be in the $5C00_{16}$ range. The active input is selected by the inverted MAD13 so the "0" inputs are active when MAD13 is high (3000_{16} range) and the "1" inputs when MAD13 is low ($5C00_{16}$ range). The end result of all this is that the address presented on the MAD bus get translated into two 4K word ranges that get presented to the two HP custom ROM chips inside the module. Addresses in the 3000_{16} to $33FF_{16}$ range get translated to 5000_{16} to $53FF_{16}$,addresses in the $5C00_{16}$ to $5FFF_{16}$ range get combined with the output of the bank select register since MAD13 is now low with the contents of the bank select register U3 providing AD10, AD11, and AD13 to the ROM and AD14 is the output of U5A which will high in this input address range, the addresses presented to the ROMs for each value of the bank select register is as follows:

0	5000_{16} to $53FF_{16}$
1	5400_{16} to $57FF_{16}$
2	5800_{16} to $5BFF_{16}$
3	$5C00_{16}$ to $5FFF_{16}$
4	7000_{16} to $73FF_{16}$
5	7400_{16} to $77FF_{16}$
6	7800_{16} to $7BFF_{16}$
7	$7C00_{16}$ to $7FFF_{16}$

Note that bank 0 in the $5C00_{16}$ range is the same bank as is accessed in the 3000_{16} range.

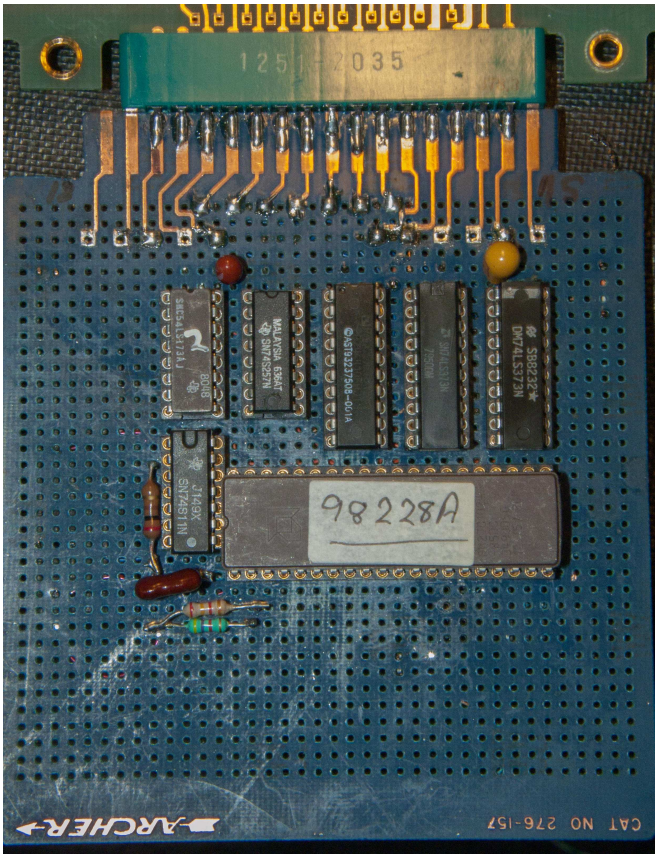
U5B ORs the -CART_OE signal with the output from the ROM that indicates it has been selected to enable tristate buffers on AD10, AD11, AD13 and AD14 and at the same time disable the outputs of U4. This is necessary since the HP custom ROM chips have a multiplexed address and data pins, and during address time we want the outputs from U4 driving these pins, but during data time we wants these pins to drive the MAD bus.

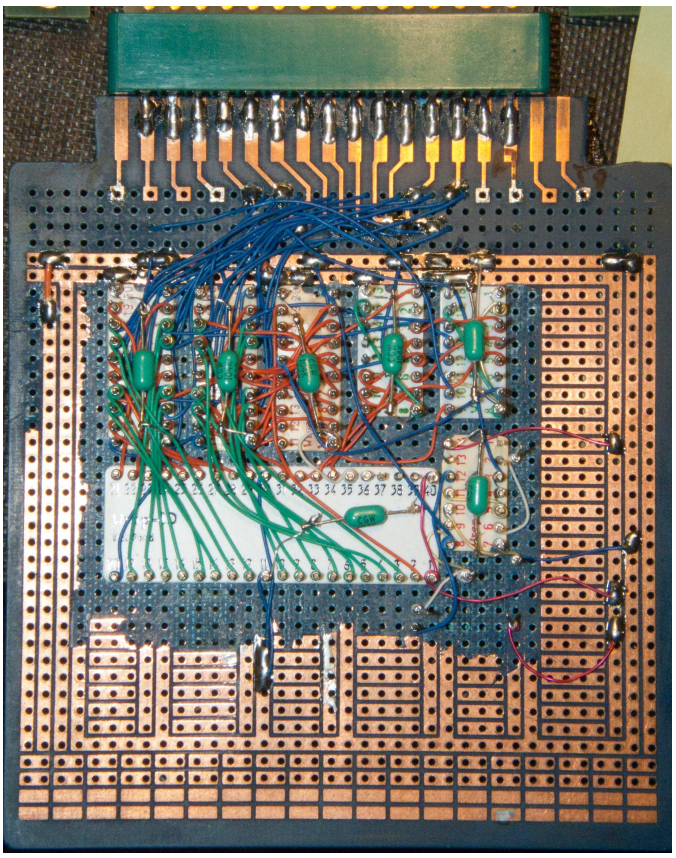
If you refer to the schematics sketches done by Tony Duell, be careful about how you interpret the pinouts of the ROM cartridge connectors connectors. The pinout included with his schematic of the 98228A ROM cartridge, which is the one I reproduced on my schematics is correct if looking into the connector on the cartridge. When facing the connectors on the ROM board in the 9825, pin 1, the -WR signal is the top right most contact. On his diagram of the ROM backplane the only way that pinout would be correct is if you where viewing the edge connectors from the back of the machine.

My Clones

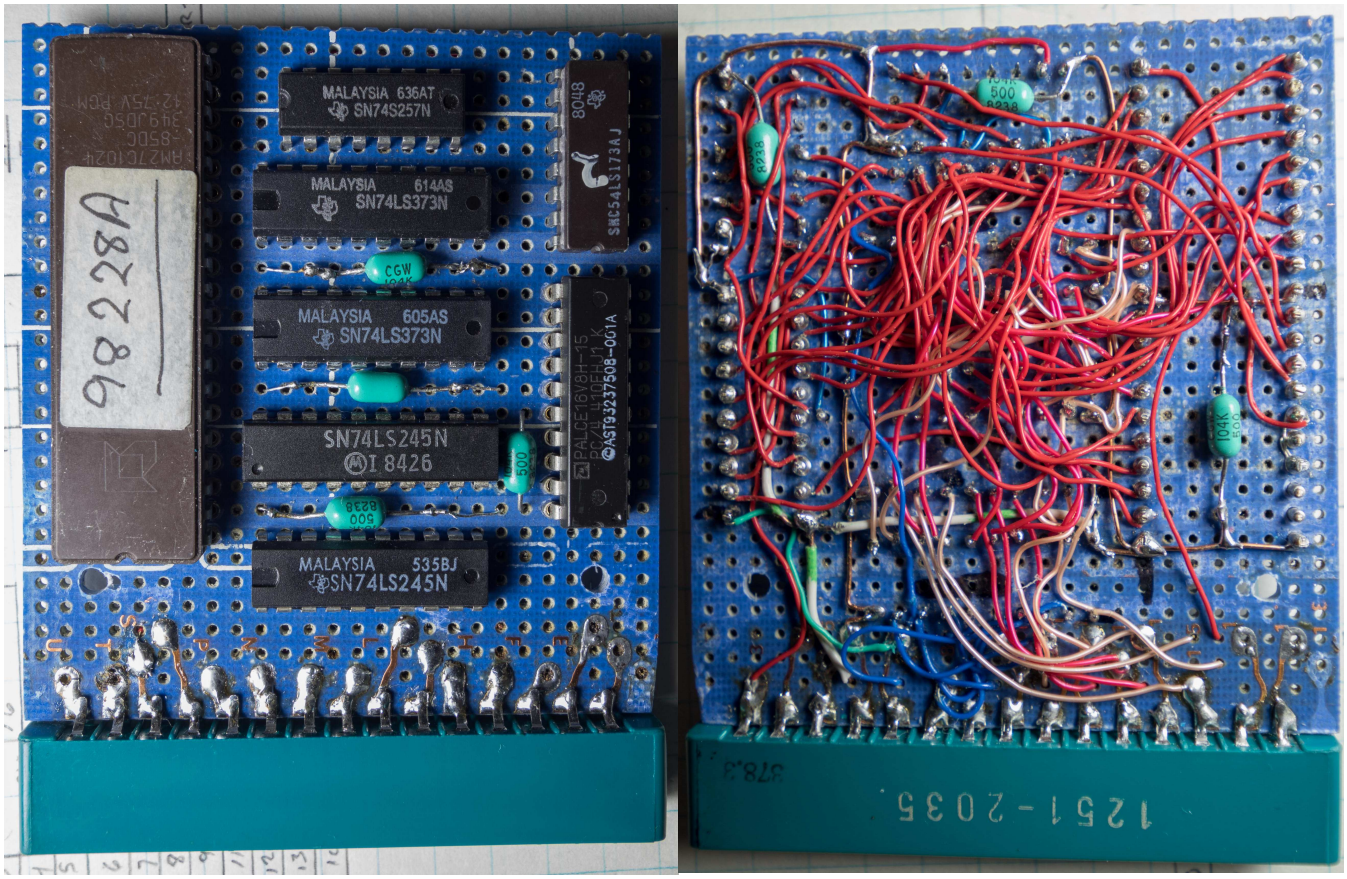
In the package with this document, there are schematics labelled clone1 and clone2 they are basically the same circuit except clone1 uses a 16bit EPROM and clone2 uses two 8 bit EPROMs. For clone1 I chose a 16 bit EPROM because the single 40 pin package takes less space that two 28 pin packages and space was at a premium inside a ROM cartridge shell. The down side of using the 16 bit EPROM is you waste a lot of space inside, the smallest one I know of is 64K x 16 you could fit all of the 9825T's ROMs into this EPROM including the 98228A and both plotter ROMs into it. Using two 8 bit EPROMs you can get a better fit using 8K EPROMs, but using all pin-in-hole components I don't think I could have squeezed in two 2764s, but thinking about it later I realized I could save space by using surface mount packages for some of the TTL instead of regular DIP packages. I eliminated the 74LS126 as it is no longer needed since the industry standard EPROMs has separate address and data pins. As mentioned earlier I decided to isolate the higher than normal voltage on the MAD bus by using 74LS245 transceivers because of concerns about exposing the EPROM to the MAD bus. The address is latched by a pair of 74LS373 and the gate level logic is replaced by a small PLD. The bank select latch and multiplexer are retained, however only three of the

four multiplexers in the 74S257 are used as all that is needed now is to select a 1K word bank, there is no longer a need to do address translation.





This is the front and back of my first prototype, the area on the wiring side that has had all the copper peeled off is roughly the area available inside the ROM cartridge. I had intended to trim this board down and use it for the version I would fit into the cartridge shell but it did not end up that way.



Both sides of the version I fit into the cartridge shell. This card is mounted component side down in the shell. The two holes on either side of the bottom 74LS245 are for fasteners that hold the shell together. The EPROM is actually too close to the hole and I had to shave down a post in the shell to get it to fit. The board is not centred inside the shell so when I soldered it onto the board I set the board and the connector into the bottom of the shell and bend the pins on the connector down to the board and soldered them down to the pads on the wiring side, I then took the card and connector out and did the same thing for the component side.

Taking the ROM cartridge shells apart can be a bit of a problem, the easiest way I found to do this is to drill a small hole through the bottom, and drive the fasteners out with a small punch, see picture to the right. For the fasteners towards the connector end of the cartridge, if you look closely you will see a faint dimple where the post is inside the cartridge, drill your holes in the centre of this dimple. The posts at the handle end are under the peak of the hand grip and to locate where to drill, on the top side measure in to the centre of the fastener to determine where to drill on the bottom.