

MLDL- 2000 and HEPAX How-To

Howard Owen, hbo@egbok.com

The MLDL 2000ⁱ far surpasses any previous MLDL for the HP41 in every measure you can imagine. It has more memory, faster memory, and superior connectivity to the modern PC desktop. So why would you want to use an older MLDL system along with the MLDL2K? One reason is that the MLDL2K is operated exclusively from the desktop PC. All the HP-41 sees are ROM images in its address space. Older MLDL systems assumed that most of the work would be done on the calculator. So those systems provide tools for manipulating RAM in the ROM address space from the calculator itself. Another reason could be nostalgia. Much of the fun of the MLDL 2000 is in loading and playing with modules that are rare in their physical forms. The HEPAX module is one of the greatest of the “classic” MLDL systems for the HP41. This article will discuss how to get the HEPAX working on an MLDL2000/41C system.

Challenges

Playing around with the HEPAX is a tricky business on a number of counts. First, there's the challenge of getting the HEPAX module in a form usable by Mldl2k, the application that comes with the MLDL2000. Next, you must master the mapping of MLDL memory to HP41 memory. Finally, you have to format some of the MLDL 2000's memory to serve as RAM for the HEPAX.

Prerequisites

Before we jump in to those challenges, we should line up the tools we will need to use during the configuration of the HEPAX module. The first thing we need are images of the HEPAX module itself. Thanks to Warren Furlow, images of the HEPAX module are available at his great site, hp41.org, in the “Custom Peripherals” section.ⁱⁱ The module comes packaged in a zip file with two files, HEPAX.MOD and Hepax-1C.mod. The latter file is an earlier version of the HEPAX. The first file is version 1D, which is the one you will probably want to use.

Next, we need a tool to convert the .MOD formatted files, used by Warren's great V41 emulator, to .ROM format, which is understood by Mldl2k. The tool to do that is also available from Warren's site, in the “Utilities” section.ⁱⁱⁱ The page containing the download is labeled “Module File Utility,” and the file is called MODFile.zip. The zip archive contains three files, MODFile.exe, MODFileWin.exe and Readme.txt. The first is a DOS executable that does the work. The second is a GUI front end for the application.

Converting the MOD file to ROM format

After downloading and unpacking the above archives, the first step in bringing up the HEPAX is to convert the HEPAX.MOD file into four .ROM files, corresponding to the four bank-switched HEPAX pages. This is fairly easy to do. Start MODFileWin.exe and select File->Info. This will put you in a file browser, from which you can navigate to and select the HEPAX.MOD file. Once you have done this, selecting File->Extract ROMs will result in five files being created in the same directory as the one that HEPAX.MOD is located in. These will be:

HEPAX.txt
Hepax1-1D.ROM
Hepax2-1D.ROM
Hepax3-1D.ROM
Hepax4-1D.ROM

The text file is the MOD “header,” which contains information that allows V41 to load the contained raw .ROM images. The four .ROM images are the four (bank switched) pages of the HEPAX module, version 1D.

Mapping MLDL2000 to HP41 Memory

The MLDL2000 comes with a very large amount of RAM relative to the HP41. There are two types of memory in the MLDL2000, flash and static ram (SRAM.) There is a total of 2 MiB of flash and 512 KiB of SRAM, for a grand total of 1.5 MiB of usable memory. The MLDL2000 uses 16 bits to represent one 10 bit 41C word, so the memory capacity is effectively cut in half. But the total of .75 Mibi words of memory dwarfs the 41C's 64 Kibi words of ROM space. This allows the MLDL 2000 user to store many ROM images in the large memory, and map them in to the HP41's address space as needed. But then a method to perform the mapping is necessary. The MLDL2000 uses settings registers to accomplish this task.

MLDL 2000 Settings Registers

The settings registers consist of eight sets of sixteen ten bit words, four sets each for the two types of MLDL2000 memory. The sixteen words in each set correspond to the sixteen 4Kibi word pages of the 41C's ROM address space. The bits within each word encode various pieces of information about which page of MLDL2000 memory is mapped to the 41C page corresponding to the current status word. The Mldl2k application provides a convenient tool to perform this mapping. (see figure 1)

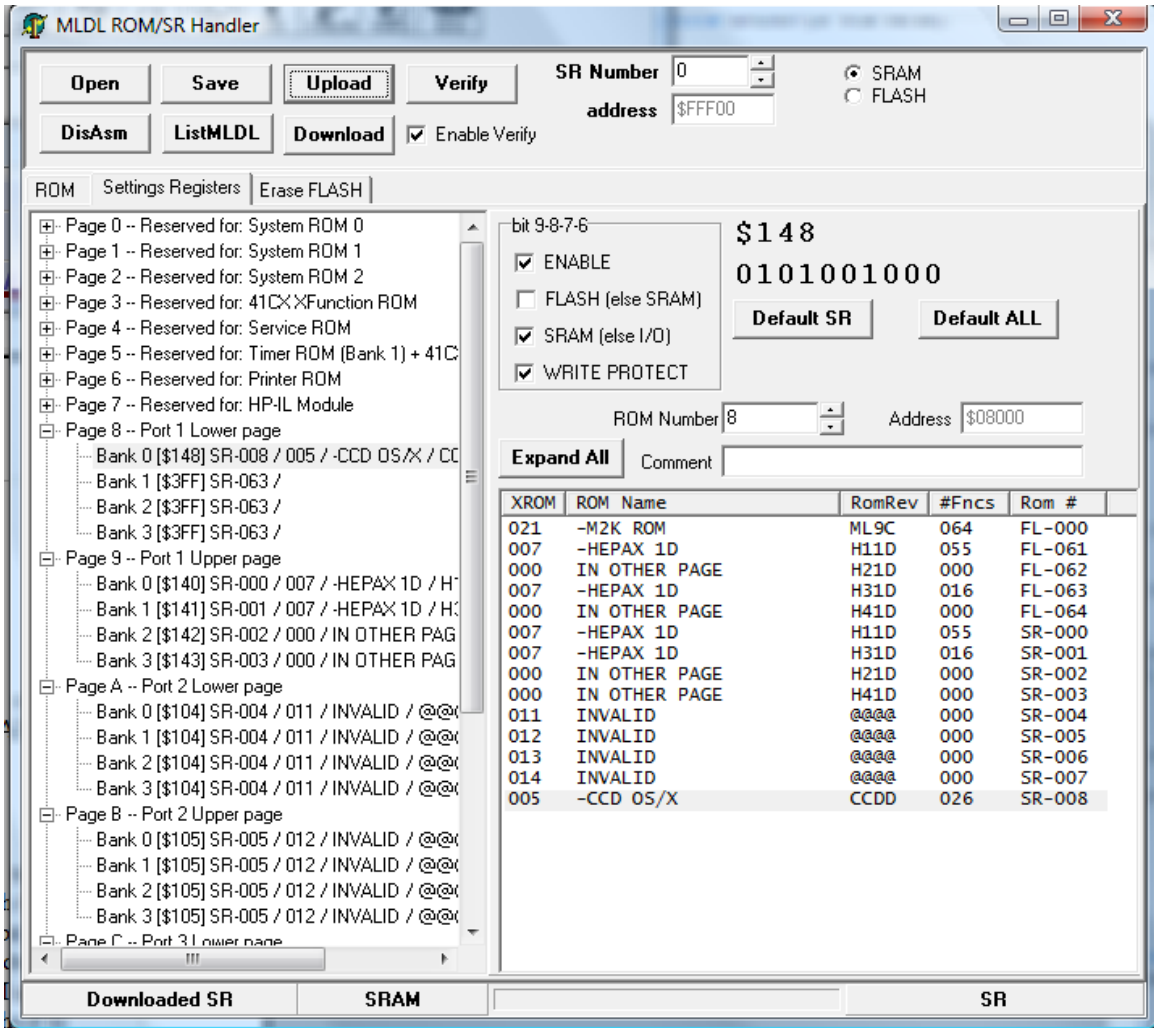


Figure 1. The Settings Register Handler

In this figure, the first bank (bank 0) of port 1 (the 8th page of 41C ROM space) is mapped to the 8th ROM image in the MLDL's SRAM. The list of MLDL memory at right shows that this SRAM clock contains the CCD OS/X ROM image. (The "Invalid" entries above are RAM page, which lack the HP41 ROM module checksum.)

The configuration shown in figure 1 is close to the one we will be implementing in this article. We will be returning to this figure later.

Loading the HEPAX into the MLDL2K

We need to load all four pages of the HEPAX somewhere in the MLDL2000's memory. This is done from the ROM tab of the ROM/SR handler. (figure 2)

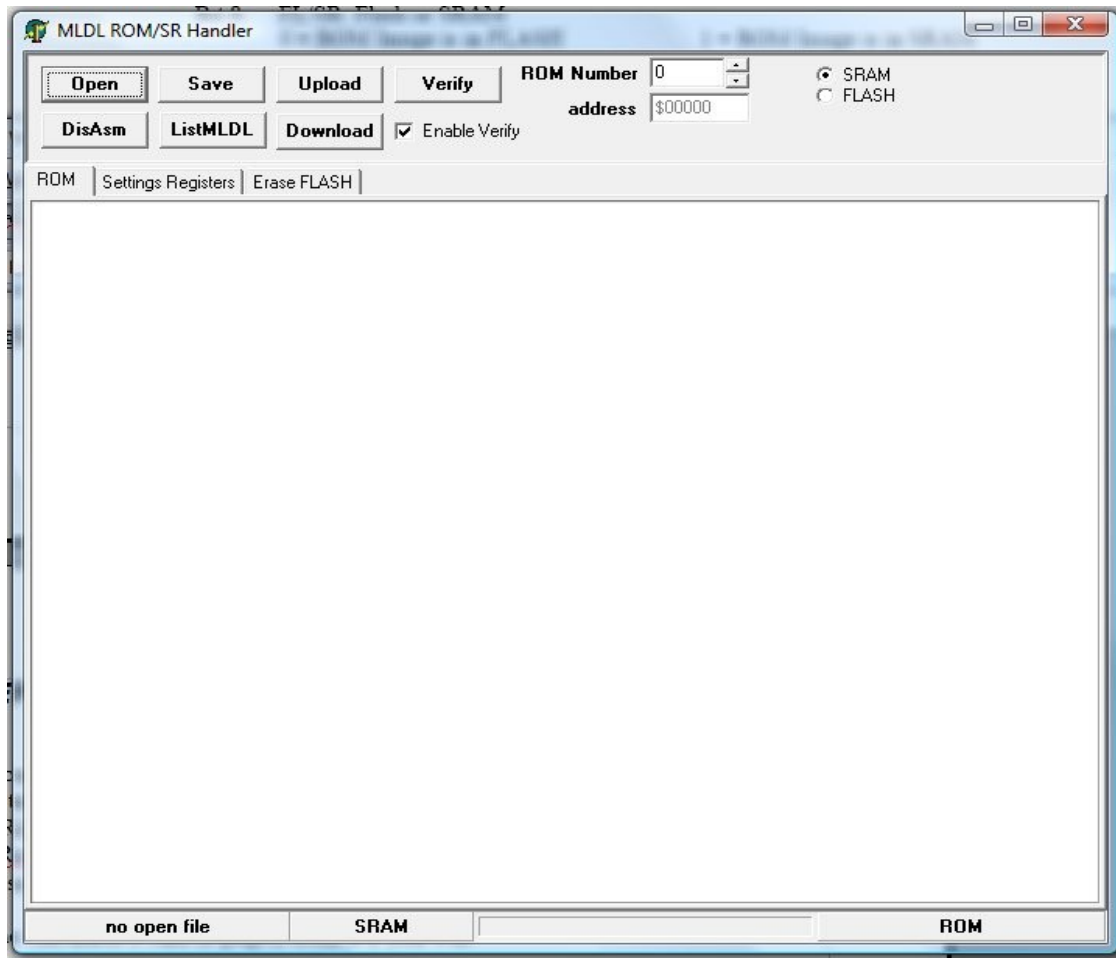


Figure 2. The ROM Handler

Proceeding from right to left in this screen, first select the type of memory you will load the image into. Here SRAM is selected. Next, select the ROM number. Here we have selected 0, for the first SRAM page. Next, click the "Open" button. This will open a file browser. Navigate to the ROM image you want to load, and press Open.

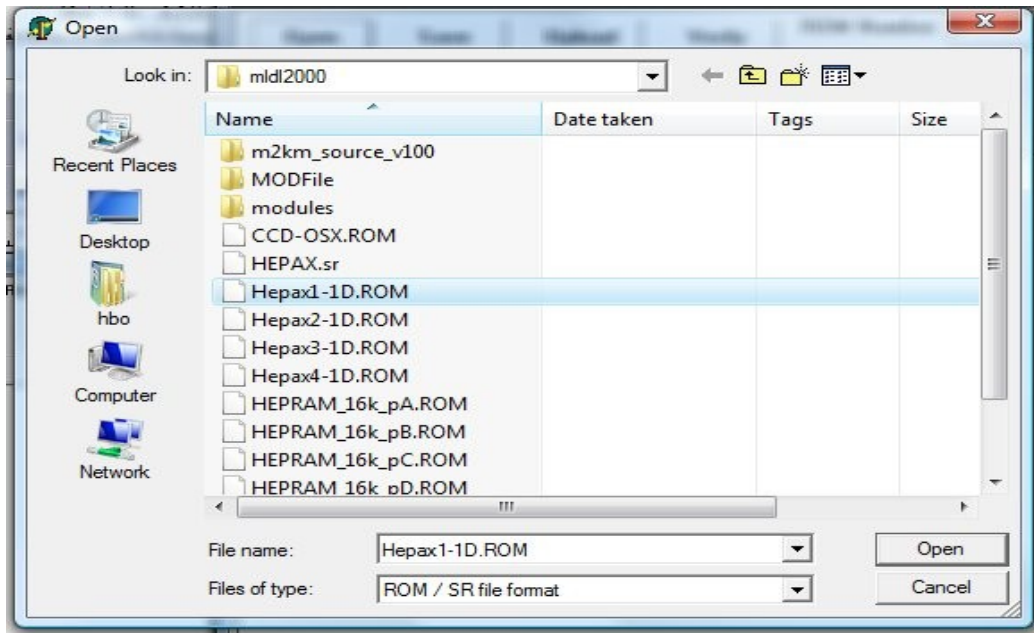


Figure 3. Selecting the ROM File to Load

Here we have selected the first of the four HEPAX images. After clicking on “Open” we get the following screen

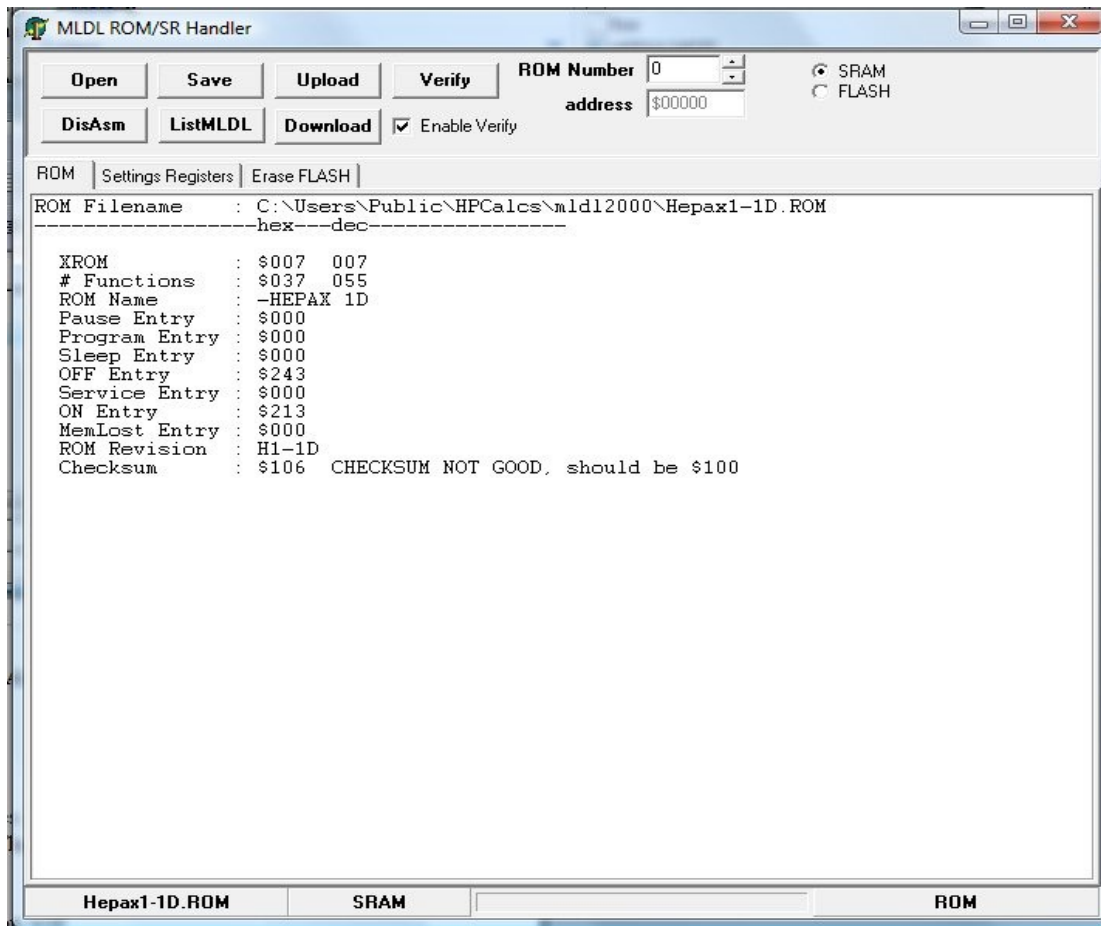


Figure 4. ROM File Loaded into MLDL2K

Up to this point, we have succeeded in loading the desired ROM image into the MLDL2K application. To write the ROM file to the MLDL 2000's memory, we must press the "Upload" button. After asking us if we really intend to do that, MLDL2K will do our bidding by writing the ROM image to the page indicated by the memory type and ROM number settings.

Tricky Bit #1

The first HEPAX ROM image has an incorrect checksum, as shown by the ROM handler. The other three ROM images have correct checksums. The incorrect checksum does not affect the operation of the HEPAX.

Tricky Bit #2

All four HEPAX pages have to be loaded into the same page in the 41C's memory. Access to these pages is accomplished using "bank switching," where a second, third or fourth page is substituted, or switched, into the address space of the primary page. The Original MLDL 2000 has a bug in the way it handles bank switching.¹ As a result, the HEPAX ROM images have to be loaded into the 41C's memory in the order 1, 3, 2, 4. There are two ways to accomplish this. First, you can load the HEPAX images onto the MLDL 2000 in the order given above. Then the mapping to 41C memory could use a normal, sequential order. On the other hand, you could load the HEPAX pages in 1, 2, 3, 4 order and switch the middle two in the mapping settings register. I've taken the first approach here.

XROM	ROM Name	RomRev	#Fncs	Rom #
021	-M2K ROM	ML9C	064	FL-000
007	-HEPAX 1D	H11D	055	FL-061
000	IN OTHER PAGE	H21D	000	FL-062
007	-HEPAX 1D	H31D	016	FL-063
000	IN OTHER PAGE	H41D	000	FL-064
007	-HEPAX 1D	H11D	055	SR-000
007	-HEPAX 1D	H31D	016	SR-001
000	IN OTHER PAGE	H21D	000	SR-002
000	IN OTHER PAGE	H41D	000	SR-003
011	INVALID	@@@	000	SR-004
012	INVALID	@@@	000	SR-005
013	INVALID	@@@	000	SR-006
014	INVALID	@@@	000	SR-007
005	-CCD OS/X	CCDD	026	SR-008

Figure 5. MLDL Layout Detail

In this detail from figure 1, we see that pages SR-000 through SR-003 are occupied with HEPAX ROM images. The "RomRev" column shows that the order of the images is 1, 3, 2, 4, as discussed above.

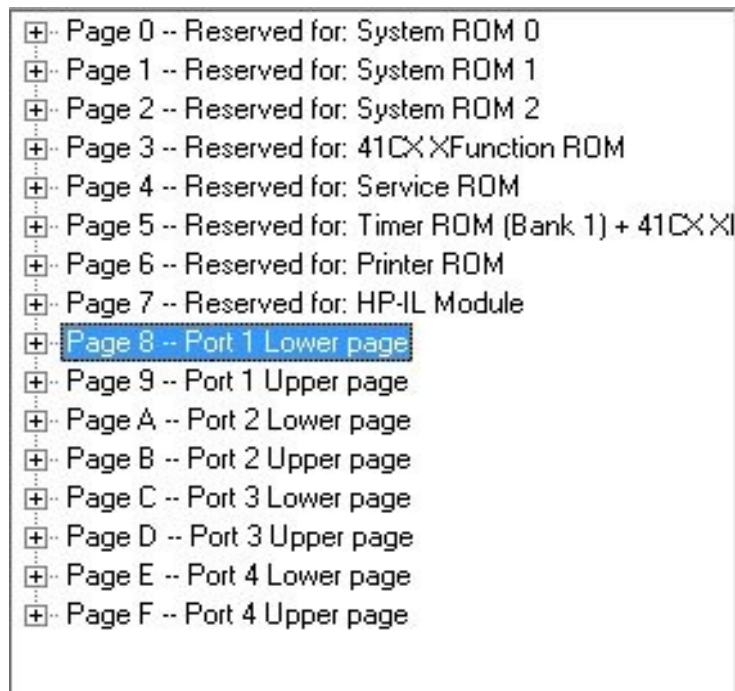
¹ AS of this writing, in July of 2007, Meindert Kuipers has updated firmware in development that will address this bug. If you have a "Mark II" MLDL 2000, just map the pages in their natural order, ignoring the special mapping in this section.

Mapping the ROMS into HP41 Address Space

The first thing we need to determine is where we want the HEPAX module to live in the HP41's address space. Since the HEPAX occupies only 4Kibi words in the 41's memory map by using bank switching, we only need to find one free page to load the module into. However, there is an additional complication we have to take account of at this point. The HEPAX uses some undocumented instructions of the Nut CPU to aid in its initialization routines. The MLDL 2000 doesn't support these instructions. So if the HEPAX is allowed to run its normal initialization, it will fail and the module will be unusable.

The solution to this problem is to load the HEPAX into an odd numbered page in the 41's address space. Since the HEPAX code assumes it is running from a real add-on module plugged in to one of the HP41's four ports, its page must be even. (Ports occupy two pages each of address space. The lower page always has an even page number.) The first thing the HEPAX does on initialization is to try and relocate itself to the lowest numbered port that doesn't already contain a module or system ROM. In order to do that, the HEPAX has to find itself in memory. It does that by looking for it's own ROM ID *in the even numbered pages only*. If we load the module into an *odd* numbered page, the HEPAX will not find itself, and will abort the initialization. This allows us to avoid the execution of those undocumented instructions, but it leaves us responsible for finishing the initialization that the HEPAX aborted out of.

So we are looking for a free, odd numbered port. We would also like the HEPAX to be situated so as not to interfere with plug-in modules we might have. And we would like to use the HP41's ROM address space efficiently, so we can get the most bang for our MLDL buck. Finally, we can't interfere with the operation of the 41C's system software. The MLDL2K SR handler gives us a nice overview of the 41C's ROM memory map:



+	Page 0	-- Reserved for: System ROM 0
+	Page 1	-- Reserved for: System ROM 1
+	Page 2	-- Reserved for: System ROM 2
+	Page 3	-- Reserved for: 41CXFunction ROM
+	Page 4	-- Reserved for: Service ROM
+	Page 5	-- Reserved for: Timer ROM (Bank 1) + 41CX XI
+	Page 6	-- Reserved for: Printer ROM
+	Page 7	-- Reserved for: HP-IL Module
+	Page 8	-- Port 1 Lower page
+	Page 9	-- Port 1 Upper page
+	Page A	-- Port 2 Lower page
+	Page B	-- Port 2 Upper page
+	Page C	-- Port 3 Lower page
+	Page D	-- Port 3 Upper page
+	Page E	-- Port 4 Lower page
+	Page F	-- Port 4 Upper page

Figure 6. 41C ROM Memory map

On all 41s, pages 0, 1 and 2 are reserved for system ROMs. On the 41CX, page three is occupied by the Xfunction ROM. Page 4 is reserved for the service ROM on all 41s, and page 5 is where the timer ROM of the 41CX goes, bank switched with more of the Xfunctions ROM. Ports 6 and 7 are reserved for the printer ROM, and the HPIL module. (The HPIL module uses port 6 if no other printer ROM is present, as long as the printer enable switch is set on the module.) If you don't plan to use HPIL or printers, then these ports are free for your use. Otherwise, the entire lower 32KiB of the 41Cs is off-limits to the MLDL2000 on a 41CX. Beginning with page 8, we encounter the pages used by plug-in ROMs (other than HPIL, Printer, Timer and Xfunctions.) Depending on what you have plugged in to the 41, some or all of these pages may be available for your use with the MLDL 2000.

My 41CX has no extra ROMs plugged in, and I intend to use printers and HPIL. So the first odd numbered page available to me is page 9. That's where I will load the HEPAX in this article. Since I have loaded the HEPAX images as described in earlier sections, I don't have to worry about the bank-switching work-around. I'll just map the RAM pages sequentially into the four banks of page 9:

XROM	ROM Name	RomRev	#Fncs	Rom #
021	-M2K ROM	ML9C	064	FL-000
007	-HEPAX 1D	H11D	055	FL-061
000	IN OTHER PAGE	H21D	000	FL-062

Figure 7. HEPAX Mapped to Page 9

To achieve the mapping, I first opened page 9 in the tree list at the left. I selected bank 0, and set the check boxes and ROM number as shown on the right. This resulted in a hex value of 0x140 for that settings register. Next, I selected bank 1 at left, and set the check boxes in same way, but selected ROM number 1. That resulted in a hex value of 0x141. I repeated these steps for each of the other two ROM images, mapping them in sequentially to the final two banks of port 9.

At this point, I can turn on my 41CX and XEQ HEPDIR. I'm rewarded with the message "H:NO FILESYS" in the 41C's display. This indicates that the basic module is working, but that no RAM pages were found. We will remedy that in the next section.

Creating HEPAX RAM

The next task is to provide some quantity of MLDL 2000 RAM to the HEPAX module to use for it's RAM. This memory must come from the SRAM of the MLDL 2000, since flash can't be written to merely by writing data to a memory location. Since we fooled the HEPAX module into skipping its initialization, the module won't have done an initial format of any RAM that may exist in the ROM address space. So we not only have to provide the SRAM pages to the 41, but we have to step in and provide the formatting the HEPAX skipped too.

The values used in this article to format the HEPAX file system were discovered by Luis C Viera and others, and published on the Museum of HP Calculators web site.^{iv} Luis used a variety of tools, but one in particular was very helpful. That was HP41X, the HP41 emulator by Hrastprogrammer for HP48, 49g and 50g calculators.^v

Appended to this article is an HP41 Mcode program that will do the formatting work for you.^{vi} ^{vii}But here's a brief description of what is needed to format pages for use as HEPAX RAM. Start with zero filled ROM images, (There are a couple of these in the HEPAX.MOD module. They will have been expanded along with the HEPAX ROM images when you ran MODFILE on the module.) Next, decide how many pages of RAM you want the HEPAX to use. Since I want the option to add other images to my 41CX, I opted to configure 16Kibi words, or four 41C pages for HEPAX RAM. This leaves one whole port free in pages E and F, plus one page at page 8. For four pages, you need four RAM images, so copy your zero filled page three times to create the four images. Next, you must decide where you want the RAM to reside in the 41Cs ROM memory space. I decided that I would use pages A through D for this.

Armed with this information, we can start editing the RAM images to provide the missing formatting. Use you favorite HEX editor. MLDL2K provides a nice one in Tools->Memory Editor. This requires you to load the RAM images into the MLDL 2000 for editing. There's also the HEXEDIT command in the HEPAX module itself, though that is a little harder to use. Starting with the first image, which will map into page A on the 41, edit the following locations with the indicated values:

Offset	Value	Comment
\$0000	\$00B	ROM ID
\$0FE7	\$000	Pointer to previous RAM page. This is the first page, so zero.
\$0FE8	\$00B	Pointer to next RAM page. This is page A, so next is B
\$0FE9	\$091	Fixed value for all pages. Function?
\$0FED	\$090	Ditto
\$0FEF	\$091	Ditto
\$0FF1	\$0E5	Ditto
\$0FF2	\$00F	Ditto
\$0FF3	\$200	Ditto

Table 1: Formatting Offsets and Values

A couple of notes. First, the 41Cs memory consists of 10 bit words, whereas the ROM file is a sequence of 8 bit bytes. The ROM file format uses 16 bits to represent the 10 bits of a 41C word. The values above are given as three hex characters, but the leftmost character is really only two bits wide. (4+4+2=10 bits) Second, subsequent pages, intended to load at pages B and C, the next and last pointers will both be non-zero. For example, the image that will load into page B will have \$00A as its last pointer and \$00C as its next pointer. The final page, loading at page D in this example, will have \$00C as its last pointer and \$000 as its next pointer. All other values are the same in each image.

If you have used an external hex editor to create the RAM images, the next step is to load the images into MLDL 2000 SRAM pages. This works exactly the same as for the HEPAX ROM images. Just pick some SRAM pages to load the RAM images. I suggest you load them into sequential pages to help keep track of which is which.

Mapping the RAM images to HP41C Pages

The mapping of HEPAX RAM images is similar to the corresponding operation on ROM images except for two differences. First, the Write Protect check box must be unchecked. Second, the RAM images must be mapped to all four banks in a given page. The following figure shows four RAM images mapped to four 41C pages:

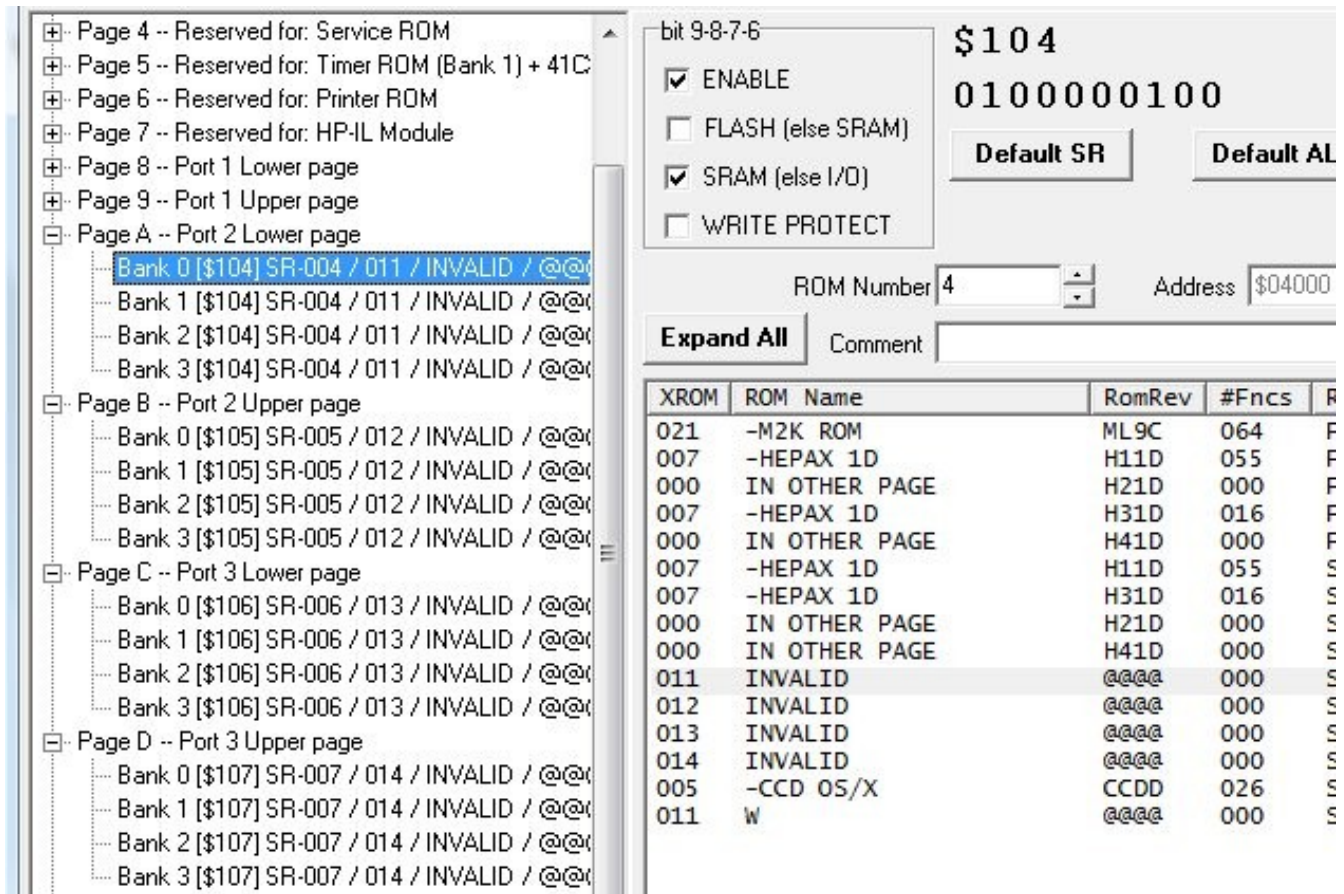


Figure 8. Mapping RAM Pages

The checksums on the RAM pages are invalid, This is normal and expected. Upload these settings to the MLDL2000 to activate them.

Putting it All Together

It is now time to repeat the test of the HEPAX HEPDIR command. This time, the calculator's display should read "H:DIR EMPTY". This means that the HEPAX module found a file system, but that it was empty. Clearing the display should show you the number of registers available in the RAM pages you just added. If you have been using the same values as the examples in this article, you should see 2,610 in the X register. Congratulations!

HEPINI - Initialize HEPAX File System Onboard the Calculator

```

#* HEPINI.SRC
* Assembled by A41
* Sat Jul 07 21:47:52 2007
;
; HEPINI - Initialize a HEPAX file system
; Copyright (C) 2007 Howard Owen
;
; This program is free software: you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation, either version 3 of the License, or
; (at your option) any later version.
;
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with this program. If not, see <http://www.gnu.org/licenses/>.
;
; Initialize the HEPAX file system beginning at the page number in X, and
; continuing for the number of pages in Y. The starting page value is checked to
; ensure that 7 < X < 16. The number of pages is checked to see that 0 < Y < 9.
; No checks are done to see that the memory written to is actually RAM, and actually writable.
;
;
; .TITLE "Initialize HEPAX RAM"
; .ZENCODE
0000 003 XROM 3
0001 001 FCNS 1
0002 00000A DEFR4K [HEPINI] 000A
; .NAME "HEPINI"
*0004 089 #089 ; "I"
*0005 00E #00E ; "N"
*0006 009 #009 ; "I"
*0007 010 #010 ; "P"
*0008 005 #005 ; "E"
*0009 008 #008 ; "H"
000A 260 [HEPINI] SETHEX
000B 04E C=0 ALL
000C 00E A=0 ALL
000D 02E B=0 ALL
000E 270 RAMSLCT ;Select chip 0, the HP41's user memory
000F 0B8 C=REG 2 ;Get the Y register, the number of pages
0010 38D008 NCXQ [BCDBIN] 02E3 ;Convert it to hex
0012 0E60C6 B=C X ;and store it in B *** (B=N)
0014 0F8 C=REG 3 ;Get the X register, the first page number
0015 38D008 NCXQ [BCDBIN] 02E3 ;Similarly
0017 106 A=C X ;But store the result in A *** (A=S)
0018 04E C=0 ALL ;load hex 10 for comparison with the start page# in A

```

```

0019 31C          PT=    1          ;
001A 050          LC      1          ;Results in (hex) 010 in C [1:0]
001B 306          ?A<C   X          ;Valid start page # is 7 < page < 16
001C 0B50A2       NCGO   [ERRDE] 282D    ;S >= 16, DATA ERROR
001E 31C          PT=    1          ;Now test the lower bound
001F 010          LC      0          ;Load hex 07 in C [1:0]
0020 1D0          LC      7          ;
0021 0A6          A<>C   X          ;*** (C=S)
0022 306          ?A<C   X          ;is 7 less than S?
0023 0B50A2       NCGO   [ERRDE] 282D    ;Nope, DATA ERROR
0025 0E6          B<>C   X          ;*** (B=S)
0026 106          A=C     X          ;*** (A=N)
0027 046          C=0    X          ;Valid nPages is 0 < N < 9
0028 39C          PT=    0          ;Load 09 into C [1:0]
0029 250          LC      9          ;
002A 306          ?A<C   X          ;Is N < 9?
002B 0B50A2       NCGO   [ERRDE] 282D    ;No, DATA ERROR
002D 346          ?A#0   X          ;Is N not equal to 0?
002E 0B50A2       NCGO   [ERRDE] 282D    ;No, DATA ERROR
0030 0C6          C=B     X          ;Start page in C(X)
0031 070          N=C     X          ;N is the lowest page number
0032 0A6106       C=A     X          ;Get the page count
0034 126          A=A+B   X          ;A(X)=start page + number of pages == highest page +1
0035 0A6          A<>C   X          ;C(X)=hp+1; A(X)=page count
0036 266          C=C-1  X          ;C(X)=hp
0037 158          M=C     X          ;M(X) is highest page number
0038 0A6          A<>C   X          ;A(X)=hp; C(X)=page count
0039 0E6          B<>C   X          ;B(X)=page count; C(X)=lowest page number
003A 0C6          [LOOP] C=B     X          ;B(X) is the count down index (page count)
003B 266          C=C-1  X          ;Underflow will set the carry bit
003C 067          JC      [OUT] +12 0048 ;No, end loop
003D 0E60C6       B=C     X          ;Yes, place decremented counter back in B(X)
003F 0B0          C=N     X          ;Calculate the current page from the base page number and the
counter
0040 106          A=C     X          ;base page number in A(X)
0041 0C6          C=B     X          ;Counter in C(X)
0042 146          A=A+C   X          ;Add them.
0043 37903C048    NCXQREL [INIPG] 0048    ;Initialize the current page
0046 3A3          JNC     [LOOP] -12 003A ;and loop
0047 3E0          [OUT]  RTN          ;Done
0048 04E          [INIPG] C=0    ALL        ;Prepare to load ROM address into C
0049 0A6106       C=A     X          ;Page number
004B 13C          RCR     8          ;Rotate page in C(X) into MSN of address field (C[6:3])
004C 0A6106       C=A     X          ;Fetch current page again
004E 226          C=C+1  X          ;This goes at the page base address
004F 040          WMLDL          ;Write X to page base address
0050 0A0          PT=P          ;C[6:3] is the address field
0051 01C          PT=    3          ;P points to the low nibble
0052 0E0          PT=Q          ;Q is the high nibble.
0053 15C          PT=    6          ;
0054 112          A=C     PQ         ;A(PQ)=page base address
0055 0B0          C=N          ;Save base address in N
0056 0B2112       C=A     PQ         ;

```

```

0058 070          N=C          ;N(PQ)=page base address;N(X)=lowest page number
0059 010          LC      0;
005A 3D03901D0   LC3      FE7          ;C(PQ)=page relative address of the last page pointer (lpp)
005D 15C          PT=       6          ;Set Q back from the LC3
005E 152          A=A+C    PQ          ;A(PQ)=absolute address of the lpp
005F 0B0          C=N          ;Get the lowest page number
0060 31C          PT=       1
0061 010          LC      0
0062 058          G=C          ;G= PCALC flag:lowest page number
0063 37903C0D2   NCXQREL [PCALC] 00D2 ;Set LPP and write it to ROM address
0066 0B0          C=N          ;Build the address of the next page pointer (NPP)
0067 09C          PT=       5          ;N had the base page in the right place.
0068 3D0390210   LC3      FE8          ;Now add the offset to the NPP
006B 15C          PT=       6          ;Reset Q from the LC3
006C 112          A=C      PQ          ;transfer the NPP address to A(PQ)
006D 198          C=M          ;Get the highest page number
006E 31C          PT=       1
006F 050          LC      1          ;This flag says we will subtract from the current page to get the
previous page
0070 058          G=C          ;G has '1P', where 'P' is the highest page number.
0071 37903C0D2   NCXQREL [PCALC] 00D2 ;Set NPP and write it to ROM address
0074 15C          PT=       6
0075 010          LC      0
0076 3D0390250   LC3      FE9          ;Load a series of constants into
0079 010250050   LC3      091          ;fixed offsets in the page. These (and preceding ones)
007C 0F20D2       B=C      PQ          ;were determined empirically through examination
007E 106          A=C      X          ;of HEPAX initialized RAM in HP41X, HrastProgrammer's
007F 37903C0C4   NCXQREL [LMLDL] 00C4 ;great HP41 emulator for the HP 48, 49 and 50. The
0082 010          LC      0          ;crucial values were discovered by Luis C. Viera and
0083 3D0390350   LC3      FED          ;published on the HP Calculator Museum web site. See in particular
0086 010250010   LC3      090          ;http://www.hpmuseum.org/cgi-
sys/cgiwrap/hpmuseum/archv015.cgi?read=83647
0089 0F20D2       B=C      PQ
008B 106          A=C      X
008C 37903C0C4   NCXQREL [LMLDL] 00C4
008F 010          LC      0
0090 3D03903D0   LC3      FEF
0093 010250050   LC3      091
0096 0F20D2       B=C      PQ
0098 106          A=C      X
0099 37903C0C4   NCXQREL [LMLDL] 00C4
009C 010          LC      0
009D 3D03D0050   LC3      FF1
00A0 010390150   LC3      0E5
00A3 0F20D2       B=C      PQ
00A5 106          A=C      X
00A6 37903C0C4   NCXQREL [LMLDL] 00C4
00A9 010          LC      0
00AA 3D03D0090   LC3      FF2
00AD 0100103D0   LC3      00F
00B0 0F20D2       B=C      PQ
00B2 106          A=C      X
00B3 37903C0C4   NCXQREL [LMLDL] 00C4

```

```

00B6 010          LC      0
00B7 3D03D00D0   LC3    FF3
00BA 090010010   LC3    200
00BD 0F20D2      B=C    PQ
00BF 106         A=C    X
00C0 37903C0C4   NCXQREL [LMLDL] 00C4
00C3 3E0         RTN
00C4 0B0         [LMLDL] C=N
00C5 09C         PT=    5
00C6 010010010   LC3    000
00C9 112         A=C    PQ
00CA 15C         PT=    6
00CB 132         A=A+B  PQ
00CC 0B2112      C=A    PQ
00CE 0A6106      C=A    X
00D0 040         WMLDL
00D1 3E0         RTN
00D2 04E         [PCALC] C=0    ALL
00D3 39C         PT=    0
00D4 098         C=G
00D5 31C         PT=    1
00D6 010         LC      0          ;C(X)=passed extreme (highest or lowest) page number
00D7 106         A=C    X          ;Save it in A(X)
00D8 0B0         C=N
00D9 17C         RCR    6          ;Rotate the high nibble into C(X)
00DA 21C         PT=    2          ;Mask off the nibbles to the left
00DB 010         LC      0
00DC 010         LC      0          ;what remains is the current page number
00DD 1C6         A=A-C  X          ;Difference with current page number
00DE 0A6         C<>A  X
00DF 2E6         ?C#0  X          ;Not the same?
00E0 03F         JC      [PNEQ] +7 00E7
00E1 15C         PT=    6          ;otherwise, we are on one of the extreme pages
00E2 010010010   LC3    000          ;The pointer to the next or previous page is
00E5 03C         RCR    3          ;therefore 0.
00E6 0BB         JNC    [PCONT] +23 00FD
00E7 39C         [PNEQ] PT=    0          ;We are in a page between the extremes
; Get the current page number from N again
00E8 0B0         C=N
00E9 17C         RCR    6          ;Rotate the high nibble into C(X)
00EA 21C         PT=    2
00EB 010         LC      0          ;Mask off the nibbles to the left
00EC 010         LC      0          ;what remains is the current page number
00ED 106         A=C    X          ;Store page number in A(X)
00EE 39C         PT=    0          ;Get the add/subtract flag
00EF 098         C=G
00F0 3CE         CSR    ALL
00F1 21C         PT=    2
00F2 010         LC      0
00F3 010         LC      0
00F4 2E6         ?C#0  X          ;Add/subtract flag
00F5 02F         JC      [ADD] +5 00FA

```

```

00F6 0A6106      C=A   X           ;Get the page number back in C(X)
00F8 266        C=C-1 X           ;the current page minus 1 is the previous page
00F9 023        JNC   [PCONT] +4 00FD
00FA 0A6106     [ADD] C=A   X
00FC 226        C=C+1 X           ;the current page plus 1 is the next page
00FD 15C        [PCONT] PT=   6
00FE 0B2112     C=A   PQ          ;Load the previous pointer address
0100 040        WMLDL                ;and write the computed value
0101 3E0        RTN

```

*

* GLOBAL SYMBOLS

* SYMBOL	VALUE	TYPE	REFERENCES
* [ADD]	00FA	REL	00F5
* [HEPINI]	000A	REL	0002
* [INIPG]	0048	REL	0043
* [LMLDL]	00C4	REL	007F 008C 0099 00A6 00B3 00C0
* [LOOP] 003A	REL	0046	
* [OUT]	0047	REL	003C
* [PCALC]	00D2	REL	0063 0071
* [PCONT]	00FD	REL	00E6 00F9
* [PNEQ] 00E7	REL	00E0	

*

* LOCAL SYMBOLS

* SYMBOL	VALUE	TYPE	REFERENCES
----------	-------	------	------------

*

* EXTERNAL REFERENCES

* SYMBOL	REFERENCED AT
----------	---------------

*

* MAINFRAME REFERENCES

* SYMBOL	VALUE	REFERENCES
* [BCDBIN]	02E3	0010 0015
* [ERRDE]	282D	001C 0023 002B 002E

*

* A41: 0 WARNINGS(S)

* A41: 0 ERROR(S)

* END

References:

- i [MLDL 2000 information: http://www.kuipers.to/hp41.htm](http://www.kuipers.to/hp41.htm)
- ii HP41.org “Custom Peripherals” section: <http://www.hp41.org/LibView.cfm?Command=List&CategoryID=75>
- iii HP41.org “Utilities” section: <http://www.hp41.org/LibView.cfm?Command=List&CategoryID=19>
- iv HEPAX file system formatting: see in particular <http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv015.cgi?read=83647>
- v HP-41X: <http://www.hrastprogrammer.com/>
- vi A Perl script that will create .ROM image files formatted for HEPAX is at <http://retrocalculator.com/hp41c/hepRAM.pl>
- vii HEPINI ROM image: <http://retrocalculator.com/hp41c/HEPINI.ROM>
HEPINI source: <http://retrocalculator.com/hp41c/HEPINI.SRC>