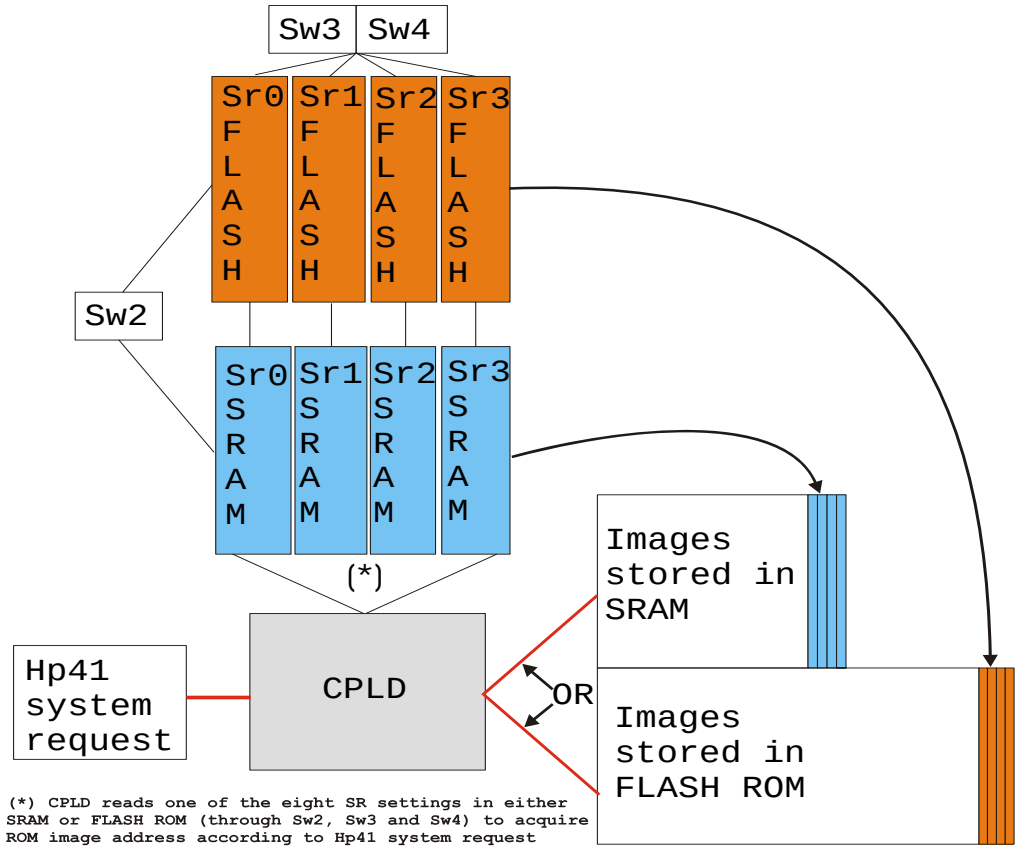
BRIEF EXPLANATIONS ON THE MLDL2000 INTERNAL ADDRESSING

By L.C.Vieira (MLDL2059, MLDL2018)

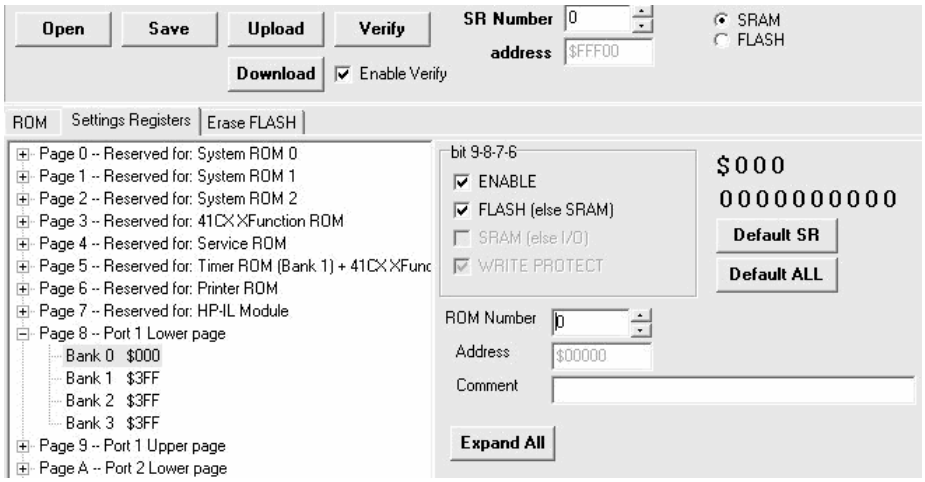
The basic operating principle over Meindert´s MLDL2000 is based on the fact that it has eight groups of the so called *setting registers* (SR), and that the contents of each SR´s logically connect each page in the HP41 to any of the existing ROM images you upload to the MLDL2000 memory.

Each setting register, any of the eight ones, has sixteen of what I call 'ROM image references' (pointers?), which can be changed at any time (SR contents). These references point to a valid SRAM or FLASH address inside the MLDL2000 memory. Each of these references is 'hardwired' to what I call a 'fixed origin point', and these fixed points are the HP41 system pages. Pages 0, 1 and 2 are designed to system ROM, page 3 is not used in the HP41C/CV while holds CX' X-functions ROM, page 4 is designed to the testing ROM and etc.

Back to the MLDL2000 internals, the active setting register is the one you select through switches SW2 (SR in SRAM of FLASH), SW3 and SW4 (SR0 to SR3). Once the SR is selected, any system request from the HP41 for any of the 16 pages is analyzed by MLDL´s CPLD and compared to the corresponding reference in the selected setting register (in fact, the active SR has its contents added to the system request to compose a valid MLDL memory address). As an example, with the default SR setting the CPLD analyses each system request in the HP41 bus and if it is not pointing to page #8, the HP41 remains quiet because all of the other SR data actually have a dummy address (255) to FLASH ROM. This is the only address in FLASH that cannot contain a ROM image because it actually holds FLASH setting registers. The picture below tries to illustrate this procedure. Note that the slightly bold red lines show actual data flow.



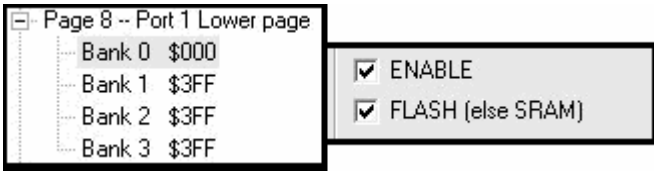
To the address definition through SW2, SW3, SW4 and SR contents are also added the image status, and this is done through ROM/SR Handler. See below.



In this picture we see that:

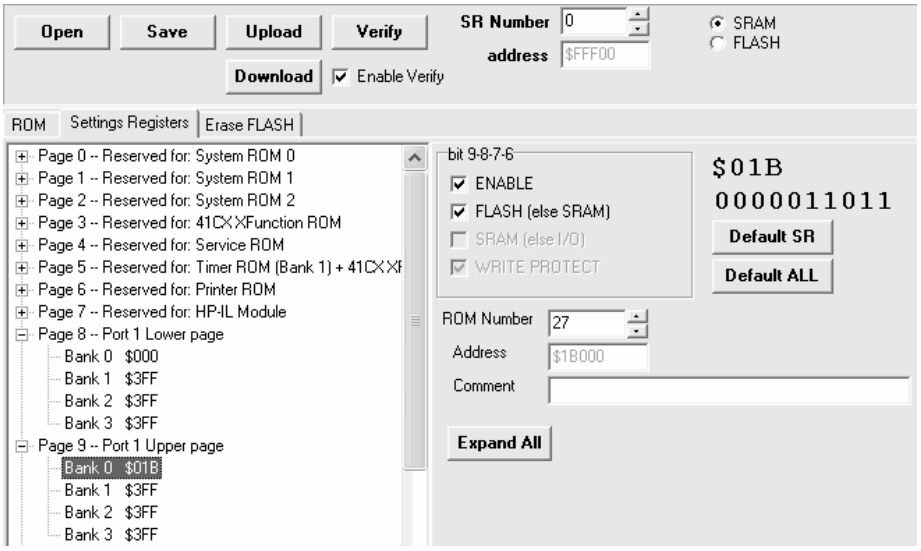
SRAM setting register 0 contents is accessed:

If this is the active SR (SW2=SW3=SW4=OFF), any system request to page 8 is addressed to the ROM image stored in FLASH h'000, and this image is enabled (system 'sees' its contents; otherwise, if ENABLE is not set, system sees nothing)



Now we need to load the necessary ROM images in the correct memory 'washers', and we use the ROM tab in the ROM/SR Handler to do so. As you have already done that, just make sure you have it stored in the correct place. I decided to keep M2K ROM in the same address it was, so the setting registers kept the same initial organization.

I use a sort of 'memory mapping follower' that I showed to Diego Dias and he liked the way I organized it. Maybe it will be useful for you, too, so I attached it to this e-mail and you can evaluate it and use it as you wish (a MSword DOC file). If you look at my own memory organization, consider that I want to make the HP41 to 'see' the Extended Functions/Memory module in page #9. By observing my map, this ROM is stored in FLASH memory, sector 6 at h'1B000 (or 27d). So I should provide the following information to the SR handler:



After saving this information (Upload), every time you select FLASH setting registers SR0 (SW2=SW3=SW4=OFF), you'll see -M2K ROM and EXT FCN IC through CAT 2.

Now it is time to go ahead filling the MLDL2000 with ROM images. If the ROM image is stable (error free), it can be stored in FLASH ROM because these ROM images are supposed to work well and do not need service/correction. Meindert´s M2K ROM is an example of that. What is actually needed by now is the following:

1 – How and why to plan the ROM images distribution in MLDL2000 memory
2 – What ROM images do you have that you want to upload to the MLDL2000
3 – How to perform the upload

Prior to do anything, have you already planned the actual distribution? I mean, where will you load the existing, available ROM images? The other two documents I sent you in one of my first e-mails are actually dedicated to this organization. If you have them in hands (or if you can have a look at them in the computer), you'll see that I organized the ROM images in MLDL2000 according to their primary function and relation. By doing so and printing the final Memory Map, it is easier to change any of the SR settings in order to have access to any of these ROM images.

As we know, each valid ROM image is composed of a single 4KByte data block or groups of 4KByte data blocks. It is not difficult to find some 8KByte modules, like the HP original Petroleum ROM and Navigation ROM, or the famous thirdy part CCD ROM, AECROM and others. Some more advanced modules may use more than 2 data blocks, like the HP Advantage (3×4KByte blocks) or HEPAX (4×4KByte blocks). In order to occupy one single page and have 4×4KByte blocks addressed through it, the HP41 operating system + hardware allow bank switching (four banks per page). Bank switching will be discussed later, though.

The HP41 system addressing is mainly composed of sixteen possible 4KByte data blocks. Each of these sixteen possible blocks are named 'pages', and they are 'numbered' (identified) from 0_h to F_h ($_h$ stands hexadecimal). The first eight valid system pages are mainly designed to the operational system (P#0 to P#2) and extensions (printer, X-functions, HPIL, etc.). They are not usually addressed otherwise, but we can go there and have some extra modules addressed to these lower, system-related pages. The upper

eight pages (P#8 to P#F) are actually ‘hardwired’ to ports 1 to 4, the ones accessed through the small caps below the LCD. In brief:

Port	Pages	Port	Pages
1	8h, 9h	2	Ah, Bh
3	Ch, Dh	4	Eh, Fh

One important fact: because of some hardware related features, some modules that I call ‘address dependent’, or ‘hardware independent’, will always be addressed the same, whatever port you plug them in. The HPIL is one of them: it is always addressed to page #7 whatever port you plug it in. The peripheral printer HP82143A, the HPIL Printer functions and the 82242A IR printer module are other examples of such ‘address dependant’ modules: whatever port you plug any of these in, they will be ‘seen’ by the HP41 system through page #5. That’s one of the three main reasons (the other two are coming later) they cannot share the same HP41 at the same time… Address conflict! The other modules I call ‘address independent’, or ‘hardware dependant’. They will be seen by the HP41 through the address ‘wired’ through the port they are plugged in. Some of them are the HP Wand (82153A), all of the application ROM modules (MATH ROM, STAT ROM, etc.) and some others (AECROM, CCD ROM, etc.).

The MLDL2000 has two blocks of memory: a FLASH ROM memory area capable of storing up to 255 ROM images, and the SRAM memory area, capable of storing up to 127 ROM images or any sort of data. The FLASH ROM may also store any sort of data, but the HP41 cannot edit FLASH ROM contents, it can only be done through the USB connection and a PC. SRAM can have its contents changed, though… And if the MLDL2000 is kept disconnected for too long, data stored in SRAM may (will) be lost, while FLASH ROM contents remain.

What we need to do prior to go ahead filling the MLDL2000 with ROM images is to define where to store them, i.e., in which addresses. Actually, an organized way to do so. I myself prefer storing ROM images in FLASH ROM because these ROM images are supposed to work well and do not need service/correction (or, at least, should not). Meindert´s M2K ROM is an example of that: it is stored in FLASH ROM #0000. The SRAM memory I actually leave for experimenting, although I did not find the time to go deeper…yet!

In order to help organizing ROM images in memory, I wrote the following DOC files to help me. I’m attaching to this e-mail the ROM_IMAGE_MAPPING_V04.DOC file. Please, have a look at it and feel free changing its contents.

The first page is a general ‘distribution’ according to the way the FLASH ROM is addressed through the CPLD. The other five pages (2 to 6) actually have the references to the ROM images actually stored (or planned to be stored) in FLASH ROM. The other three pages (7 to 9) refer to data stored (or to be stored) in SRAM. First, leftmost column is a reference to the sector, already defined by the CPLD. The second column is a sequential number that refers to each 4KByte FLASH ROM block. The last possible address in each area (FLASH ROM = 255, SRAM = 127) refers to the SR Settings address, or the address where the SR definition is stored. These addresses cannot be filled with anything else.

Once we define ‘what’ goes ‘where’, we can prepare both, computer and HP41+MLDL2000, to be connected. At this point, Dr. Jüergen, it would be easy to both of us if I know where did you choose to store the ROM images. Please, let me know if you agree with the distribution I made or if you plan to do something else. I prefer reading from

you prior to go ahead. If you believe my proposal is fine for you, then just let me know. I know you wrote you'd be "following my guidance", but it does not exclude the possibility of you finding something I did not see that would be better.

The next steps will be:

1. reading a ROM image to the M2KM (MLDL2000 Manager) buffer (PC);
2. setting where to store the ROM image already in the M2KM buffer (PC);
3. uploading the ROM image in the M2KM buffer to the MLDL2000 memory;
4. reading the MLDL2000 memory in the selecting position to confirm if uploading was successful;
5. repeat from step #1 till all needed/available ROM images are stored;
6. configure SR settings in order to 'connect' desired ROM images to any available page in the HP41 (PC);
7. uploading SR setting to the MLDL2000
8. disconnect after all is done;
9. test if system is OK after configuration.

I'm confident we'll succeed configuring your entire MLDL2000 this very weekend.

O.K.! So, fasten your seat belts and grasp the chair arms… First, please, make sure you can easily identify all ROM images in your computer, i.e., in which directory they are actually stored. Now let's plan a bit.

I am not exactly sure about which area do you actually intend to use your HP41 with, and I am also not sure if you have an HP41C/CV or an HP41CX. I'll try a configuration that suits any model. We can discuss later a better, more efficient proposal, so let's walk through the basics. Let's first upload the following images to the MLDL2000 FLASH ROM:

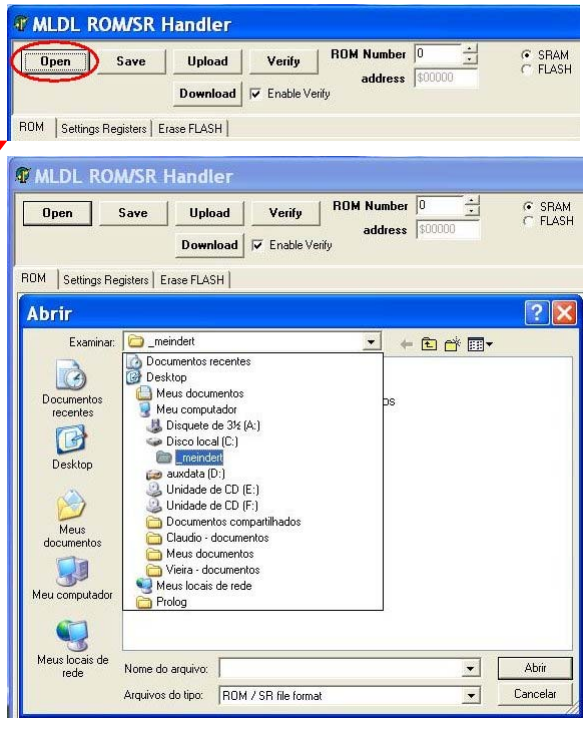
MATH 1D(*) (if possible, all of the versions: 1A, 1B, 1C and 1D)
STATISTICS 1B(*)
SURVEY 1B
FINANCE 1D(*) (if possible, all of the versions: 1B and 1D)
STANDARD 1C (this one can be replaced, though)
CIRCUIT 1A
STRUCTURAL L 1B
STRESS 1A

(*)These are the ones we are actually going to use now; anyway, you can load all of these and any others you wish

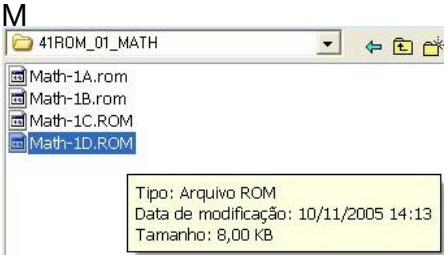
This would fill FLASH ROM sectors 30 and 33, according to our map. Then we would set SRAM SR#0 to hold the following definition:

Page	ROM image
8	M2K ROM (already done)
9	MATH 1D
A	STATISTICS 1B
B	FINANCE 1D

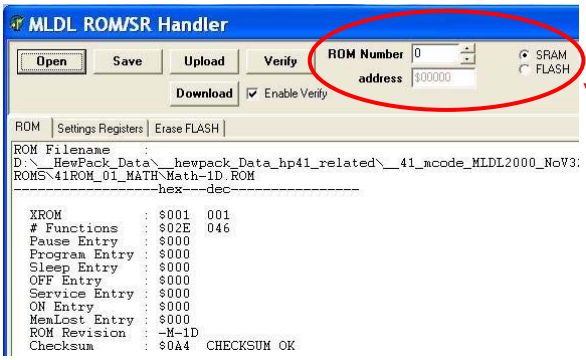
If we agree with, let's connect the HP41+MLDL2000 to the computer and run MLDL2K.EXE. The first step is loading the ROM images in MLDL2000 FLASH ROM, according to our maps. Right after connecting, open the ROM/SR Handling tool and make sure it is ready to handle ROM images.



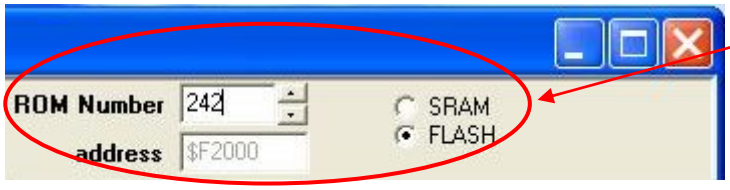
Now click in [OPEN] and you'll see the same [OPEN] folder you find at Windows Explorer. (mine is in Portuguese, and 'Abrir' means 'To Open'). Please, find the directory where you loaded the ROM images and locate the MATH1D.ROM



Now, confirm with [OPEN] in your Windows Explorer folder. You should see this in your computer:

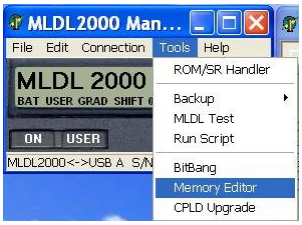


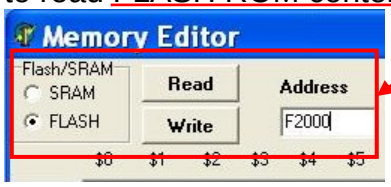
This image tells us that Math-1D.ROM is loaded in the computer buffer (MLDL2K Manager RAM buffer), and that we can upload it to the MLDL2000 anytime we want to. At this moment the computer is set to do so at ROM number 0, address 00000_h$ in SRAM. We do not want to do that, because according to our map, this ROM image is intended to be stored in FLASH ROM, address $F2000, ROM Number 242 (see page 6 of the DOC file I sent you, ROM mapping)., so we must change these data.



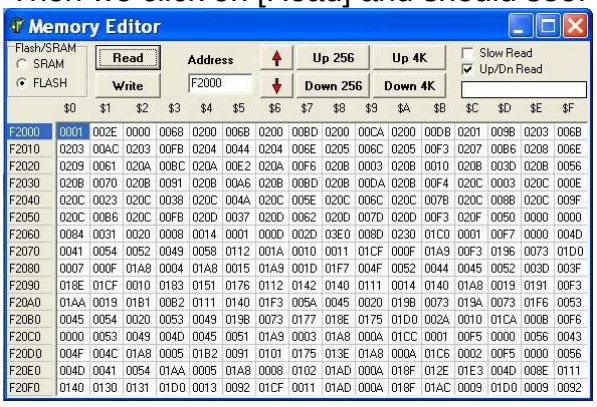
Now we just press [Upload], confirm, and wait for less than one second. A process evolution indicator shows how it goes…

As for safety, we can (optionally) open the Memory Editor tool and have a look at the memory contents. First we confirm that we want to read FLASH ROM contents at address $F2000$_h$.



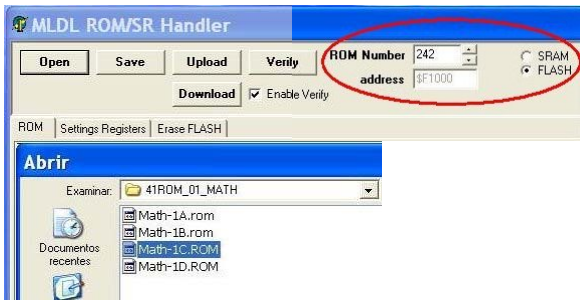


Then we click on [Read] and should see:

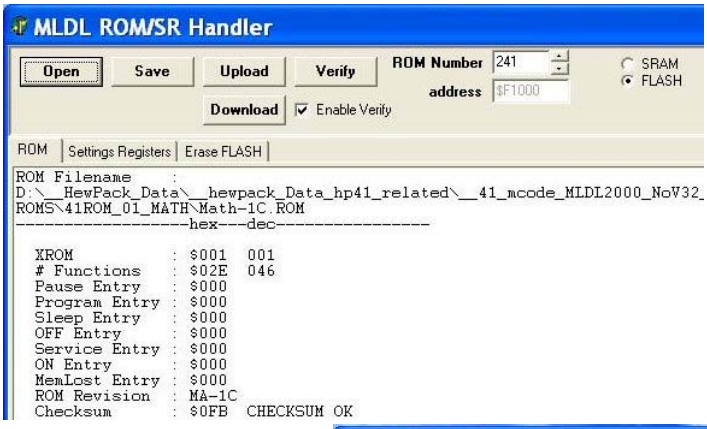


What have we done so far? So far we loaded the ROM image of the MATH 1D module into the MLDL2000's FLASH ROM, address $F2000, equivalent to ROM number 242, sector 33. What should be done now? Now we tell the MLDL2000 CPLD that every possible fetch to page #9 should be intercepted and be given, as a return, the equivalent contents of the newly recorded ROM, namely MATH 1D. This is done through SR contents, and we already know which SR we want to use: SR0 in SRAM. If you feel confident to do so (I myself prefer to), just load every possible ROM image to the MLDL2000 FLASH ROM, then you'll only need to change SR contents whenever you want. Remember that there are four possible SR in SRAM and another four in FLASH ROM, so at least eight ready-to-activate SR may be kept in the MLDL2000, and each of these will give you a fully configured HP41 8^) Pretty neat, ahn?

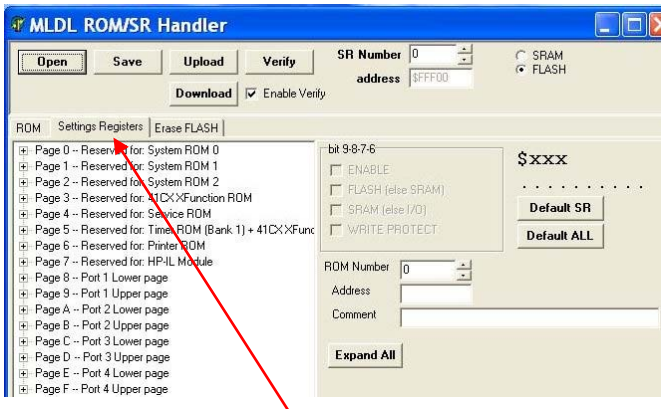
As an example, to load MATH1C.ROM to the MLDL2000 FLASH ROM, address $F1000h, or ROM Number 241, first we go back to ROM handling in the RM/SR Handler window, open the MATH1C.ROM (load it in the handler buffer). Note that data from previous ROM image uploading is kept



Then we update the necessary fields, as shown below.



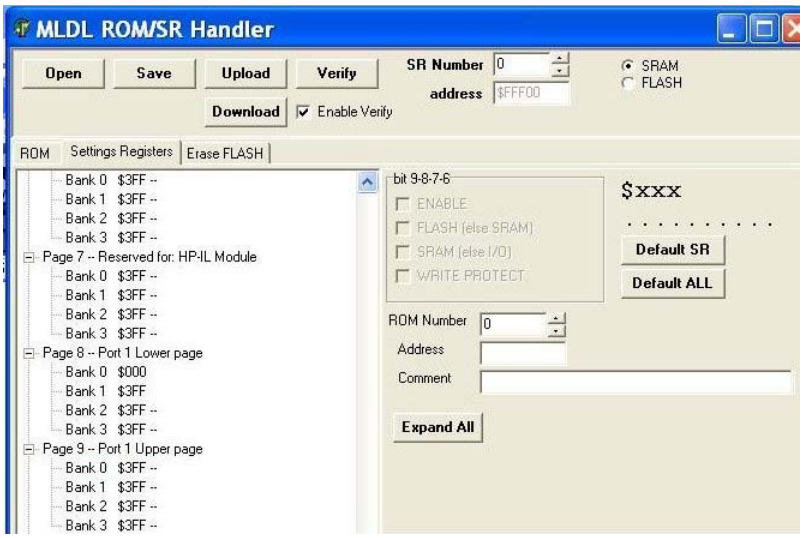
Then just click on [Upload], confirm, and check the process evolution indicator. In order to complete the whole example, we should load at least STATISTICS 1B and FINANCE 1D.



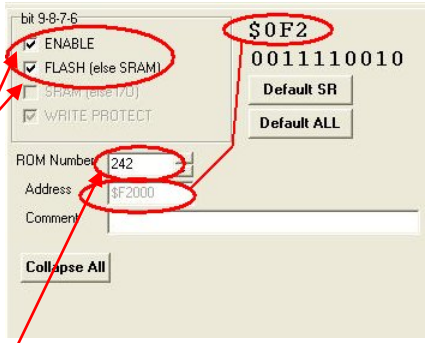
Now let's deal with the SR data. Consider that the ROM images we need are already loaded. It is too important to know WHERE they are, because we can only use the ROM images in the MLDL2000 if we correctly point them through the SR contents. The SR contents are the link telling the CPLD how to react to any possible system call. Now, click in the Setting Registers tab of the ROM/SR handler.

I prefer handling the SR registers after [Expand All], because I like to see what is going on. The next images will consider it so, but you may just expand one single assignment by clicking in the small [+] at the left side of it.

When you first open the SR handler, you'll see the default SR contents. In order to know what is actually stored in the SR you want to handle (change, update), you should [Download] its contents. To download SRAM SR0 contents, set the SR header accordingly and press [Download]. Confirm it and you'll see:



These are the actual contents of the SRAM SR0. As expected, the only programmed assignment is the one for page #8 (Port 1, lower page) and its contents 'point' to $000. Page #9 has a default value and points to nowhere, so we will set it to point to MATH 1R ROM image, stored in FLASH ROM at address $F2000h, or ROM number 242. Now click in Bank 0 of Page 9 and you see that its corresponding default configuration is loaded. Then, what we need to set is shown in the next image:

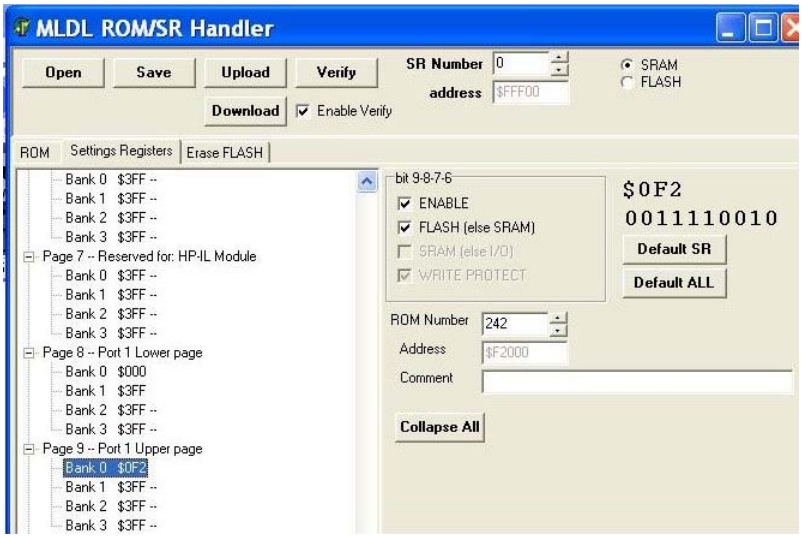


The best order to alter these fields is:

Choose FLASH;
Then choose ENABLE;
Then specify ROM number (key it in or use the up/down arrows).

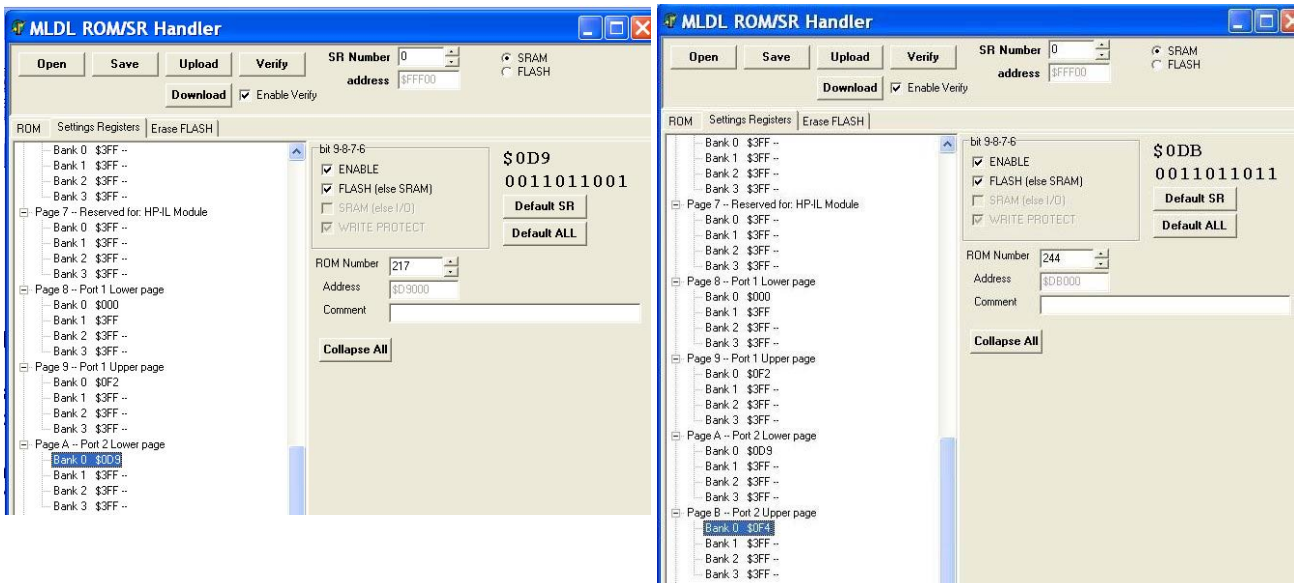
As you are setting the new information, the corresponding SR field is updated accordingly. You may also note that there is a 'bit decoder' showing the actual binary code corresponding to the ROM address. I usually left Comments filed blank, never thought

about using it. Might be handy to inform what ROM image (or purpose) is being pointed, but I actually never tried it out. At this point you should see:



One important note: any SR, either in SRAM or in FLASH ROM, may point to any memory area, that may also be either in SRAM or in FLASH ROM. SR contents defines that, not SR type. And we see this here: SR in SRAM pointing to ROM images stored in FLASH ROM.

Prior to upload the newly changed SR, let's go ahead and set Page #A and Page #B pointers:



Page #A now points to ROM Number 217 (in our map: STATISTICS 1B, sector 30), while Page #B now points to ROM number 244 (FINANCE 1D, Sector 33). So, having all points being set, all we need to do now is to upload the updated SRAM SR0 back to the MLDL2000. As all settings in the header match, simply click in [Upload] and confirm.

Now we can close the MLDL2K manager, disconnect the calculator from the computer and test if everything is fine through [CAT] 2. If yours is a 41CX, you'll see the headers for the new ROM images in sequence. If you have either a C or a CV, you can alternatively press and hold [ENTER] while turning your calculator [ON]. As we have already set M2K ROM, the [ON][ENTER] sequence starts a check that shows all ROM images in their pages. If it is O.K., you'll see:

8:-M2K ROM
9: MATH 1D
A:STAT 1D
B:FINANCE 1D

SUGGESTED MEMORY MAPPING FOR THE MLDL2000

This is a suggestion for memory mapping the MLDL2000 ROM images. The goal is to keep track with ROM images already stored while having a better idea of where to store any new ones. This is not based on the whole bunch of existing HP41 ROM images and their versions, so this map may not be applicable when storing them all.

GENERAL MAPPING (BLOCKS)

Sct#	#ROM	Start	ROM contents
0	2	**00000**	(originally-M2K ROM)
1	1	**02000**	
2	1	**03000**	
3	4	**04000**	
4	8	**08000**	HP mainframe ROM's (CX)
5	8	**10000**	HP mainframe ROM's (CX)
6	8	**18000**	HP mainframe ROM's (C/CV)
7	8	**20000**	HP mainframe ROM's (C/CV)
8	8	**28000**	
9	8	**30000**	
10	8	**38000**	
11	8	**40000**	
12	8	**48000**	
13	8	**50000**	
14	8	**58000**	
15	8	**60000**	
16	8	**68000**	MCODE development
17	8	**70000**	MCODE development
18	8	**78000**	thirdy part system extensions (devices)
19	8	**80000**	thirdy part system extensions (devices)
20	8	**88000**	thirdy part system extension (functions)
21	8	**90000**	thirdy part system extension (functions)
22	8	**98000**	thirdy part application programs/functions
23	8	**A0000**	thirdy part application programs/functions
24	8	**A8000**	HP other modules
25	8	**B0000**	HP other modules
26	8	**B8000**	HP system extensions (devices)
27	8	**C0000**	HP system extensions (devices)
28	8	**C8000**	HP system extensions (functions)
29	8	**D0000**	HP system extensions (functions)
30	8	**D8000**	HP application programs
31	8	**E0000**	HP application programs
32	8	**E8000**	HP application programs
33	8	**F0000**	HP application programs
34	7	**F8000**	
34	1	**FF000**	Setting Registers

FLASH MAPPING

Sct#	ROM#	Start	ROM ID, Version	XROM/OBS
0	0	**00000**	**-M2K ROM**	**XROM 21** (copied in 7A000)
	1	**01000**		
1	2	**02000**		
2	3	**03000**		
3	4	**04000**		
	5	**05000**		
	6	**06000**		
	7	**07000**		
4	8	**08000**	**CXROM0 1P**	**CX mainframe**
	9	**09000**	**CXROM1 1P**	
	10	**0A000**	**CXROM2**	
	11	**0B000**	**CX XFCN L 2D**	**XROM 25**
	12	**0C000**	**CX XFCN U**	**XROM 25**
	13	**0D000**	**CX TIMER 2C**	**XROM 26**
	14	**0E000**		
	15	**0F000**		
5	16	**10000**		
	17	**11000**		
	18	**12000**		
	19	**13000**		
	20	**14000**		
	21	**15000**		
	22	**16000**		
	23	**17000**		
6	24	**18000**	**C/CVROM0 1P**	**C/CV Mainframe**
	25	**19000**	**C/CVROM1 1P**	
	26	**1A000**	**C/CVROM2**	
	27	**1B000**	**X FUNCTIONS 1C**	**XROM 25**
	28	**1C000**		
	29	**1D000**		
	30	**1E000**		
	31	**1F000**		
7	32	**20000**		
	33	**21000**		
	34	**22000**		
	35	**23000**		
	36	**24000**		
	37	**25000**		
	38	**26000**		
	39	**27000**		
8	40	**28000**		
	41	**29000**		
	42	**2A000**		
	43	**2B000**		
	44	**2C000**		
	45	**2D000**		
	46	**2E000**		
	47	**2F000**		
9	48	**30000**		
	49	**31000**		
	50	**32000**		

	51 52 53 54 55	**33000** **34000** **35000** **36000** **37000**		
10	56 57 58 59 60 61 62 63	**38000** **39000** **3A000** **3B000** **3C000** **3D000** **3E000** **3F000**		
11	64 65 66 67 68 69 70 71	**40000** **41000** **42000** **43000** **44000** **45000** **46000** **47000**		
12	72 73 74 75 76 77 78 79	**48000** **49000** **4A000** **4B000** **4C000** **4D000** **4E000** **4F000**		
13	80 81 82 83 84 85 86 87	**50000** **51000** **52000** **53000** **54000** **55000** **56000** **57000**		
14	88 89 90 91 92 93 94 95	**58000** **59000** **5A000** **5B000** **5C000** **5D000** **5E000** **5F000**		
15	96 97 98 99 100 101 102 103	**60000** **61000** **62000** **63000** **64000** **65000** **66000** **67000**		
16	104	**68000**	**DAVID ASSM 2C**	**XROM 02**

	105	69000	DAVID ASSM LABELS	
	106	6A000	MASS 1H	XROM 28
	107	6B000	ZENROM	XROM 05
	108	6C000	HEPAX H1 1D	XROM 07(EARLY MAPPING)
	109	6D000	HEPAX H2 1D	(EARLY MAPPING)
	110	6E000	HEPAX H3 1D	XROM 07(EARLY MAPPING)
	111	6F000	HEPAX H4 1D	(EARLY MAPPING)
17	112	70000	BLDROM 1B	XROM 05
	113	71000	ALPHA ROM 41	XROM 06
	114	72000	SANDBOX L 3D	XROM 08
	115	73000	SANDBOX U 3D	XROM 13
	116	74000	TOOLBOX 4U	XROM 13
	117	75000	NFCROM 1B	XROM 17
	118	76000		
	119	77000		
18	120	78000	MCEPROM 1C	XROM 15
	121	79000	MLDL ERAMCO 9B	XROM 21
	122	7A000	-M2K ROM 9C	XROM 21 (copied in 00000)
	123	7B000	ESRSU1A L DF	XROM 04
	124	7C000	ESRSU1A U AF	XROM 06
	125	7D000	ESMLOS 7B	XROM 10
	126	7E000		
	127	7F000		
19	128	80000		
	129	81000		
	130	82000		
	131	83000		
	132	84000		
	133	85000		
	134	86000		
	135	87000		
20	136	88000	HEPAXADVH1 1C	XROM 07
	137	89000	HEPAXADVH2 1C	
	138	8A000	HEPAXADVH3 1C	XROM 07
	139	8B000	HEPAXADVH4 1C	
	140	8C000	HEPAX H1 1D	XROM 07
	141	8D000	HEPAX H2 1D	
	142	8E000	HEPAX H3 1D	XROM 07
	143	8F000	HEPAX H4 1D	
21	144	90000	CCDL 1B	XROM 09
	145	91000	CCDU 2B	XROM 11
	146	92000	CCD OSX	XROM 05
	147	93000	CCD+ L 1B	XROM 09
	148	94000	CCD+ U 2B	XROM 11
	149	95000	PPCL	XROM 10
	150	96000	PPCU	XROM 20
	151	97000		
22	152	98000	41Z 4Z	XROM 01
	153	99000	LANDNAV 1A	XROM 01
	154	9A000	AECROML 1A	XROM 18
	155	9B000	AECROMU 1A	XROM 18
	156	9C000	SANDMATH III L	XROM 03
	157	9D000	SANDMATH III U	XROM 02
	158	9E000		

	159	**9F000**		
23	160	**A0000**	**FORTH H4 L 05**	
	161	**A1000**	**FORTH H5 H 06**	**XROM 09**
	162	**A2000**		
	163	**A3000**		
	164	**A4000**		
	165	**A5000**		
	166	**A6000**		
	167	**A7000**		
24	168	**A8000**		
	169	**A9000**		
	170	**AA000**		
	171	**AB000**		
	172	**AC000**		
	173	**AD000**		
	174	**AE000**		
	175	**AF000**		
25	176	**B0000**	**PLOTTER L 1A**	**XROM 17**
	177	**B1000**	**PLOTTER U 2A**	**XROM 18**
	178	**B2000**	**DATA ACQ L 1B**	**XROM 21**
	179	**B3000**	**DATA ACQ U 1B**	**XROM 31**
	180	**B4000**		
	181	**B5000**		
	182	**B6000**		
	183	**B7000**		
26	184	**B8000**	**HPIL DEVLP L 1A**	**XROM 24**
	185	**B9000**	**HPIL DEVLP U 1B**	**XROM 22**
	186	**BA000**	**EXTENDED IO 1A**	**XROM 23**
	187	**BB000**	**HPIL 1H**	**XROM 28**
	188	**BC000**		
	189	**BD000**		
	190	**BE000**		
	191	**BF000**	**AUTOSTART 1A**	**XROM 10**
27	192	**C0000**	**WAND 1F**	**XROM 27**
	193	**C1000**	**PRINTER 1E**	**XROM 29**
	194	**C2000**	**PRINTER IL 2E**	**XROM 29**
	195	**C3000**	**PRINTER IR 3C**	**XROM 29**
	196	**C4000**	**CARD READER 1F**	**XROM 30**
	197	**C5000**	**CARD READER 1G**	**XROM 30**
	198	**C6000**		
	199	**C7000**		
28	200	**C8000**	**ADVANTAGE L1 1B**	**XROM 22**
	201	**C9000**	**ADVANTAGE U1 1B**	**XROM 24**
	202	**CA000**	**ADVANTAGE U2 1B**	
	203	**CB000**	**ADVANTAGE APP 1A**	**XROM 19**
	204	**CC000**		
	205	**CD000**		
	206	**CE000**		
	207	**CF000**		
29	208	**D0000**		
	209	**D1000**		
	210	**D2000**		
	211	**D3000**		
	212	**D4000**		

	213 214 215	**D5000** **D6000** **D7000**		
30	216 217 218 219 220 221 222 223	**D8000** **D9000** **DA000** **DB000** **DC000** **DD000** **DE000** **DF000**	**MATH 1A** **STATISTICS 1B** **SURVEY 1B** **FINANCE 1B** **STANDARD 1C** **CIRCUIT 1A** **STRUCTURAL L 1B** **STRESS 1A**	**XROM 01** **XROM 02** **XROM 03** **XROM 04** **XROM 05** **XROM 06** **XROM 07** **XROM 08**
31	224 225 226 227 228 229 230 231	**E0000** **E1000** **E2000** **E3000** **E4000** **E5000** **E6000** **E7000**	**HOME MANAGEMENT 1A** **GAMES 1A** **REAL ESTATE L 1A** **REAL ESTATE U 1A** **MACHINE 1A** **THERM/TRANSPORT 1A** **NAVIGATION L 1B** **NAVIGATION U 1B**	**XROM 09** **XROM 10** **XROM 11** **XROM 11** **XROM 12** **XROM 13** **XROM 14** **XROM 14**
32	232 233 234 235 236 237 238 239	**E8000** **E9000** **EA000** **EB000** **EC000** **ED000** **EE000** **EF000**	**PETROLEUM L 1A** **PETROLEUM U 2A** **AVIATION 1A** **CLINICAL LAB 1A** **SECURITIES 1A** **STRUCTURAL U 1A**	**XROM 15** **XROM 16** **XROM 19** **XROM 19** **XROM 19** **XROM 19**
33	240 241 242 243 244 245 246 247	**F0000** **F1000** **F2000** **F3000** **F4000** **F5000** **F6000** **F7000**	**MATH 1B** **MATH 1C** **MATH 1D** **FINANCE 1C** **FINANCE 1D**	
34	248 249 250 251 252 253 254 (SR)	**F8000** **F9000** **FA000** **FB000** **FC000** **FD000** **FE000** **FF000**		

Setting Register Layout									
Bit 9	***Bit 8***	***Bit 7***	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
EN	0=SRAM	A19	A18	A17	A16	A15	A14	A13	A12
EN	1=FLASH	IO	WP	A17	A16	A15	A14	A13	A12

SRAM mapping

Sct#	ROM#	Start	ROM ID, Version	XROM/obs
0	0 1	**00000** **01000**		
1	2	**02000**		
2	3	**03000**		
3	4 5 6 7	**04000** **05000** **06000** **07000**		
4	8 9 10 11 12 13 14 15	**08000** **09000** **0A000** **0B000** **0C000** **0D000** **0E000** **0F000**		
5	16 17 18 19 20 21 22 23	**10000** **11000** **12000** **13000** **14000** **15000** **16000** **17000**		
6	24 25 26 27 28 29 30 31	**18000** **19000** **1A000** **1B000** **1C000** **1D000** **1E000** **1F000**		
7	32 33 34 35 36 37 38 39	**20000** **21000** **22000** **23000** **24000** **25000** **26000** **27000**		
8	40 41 42 43 44 45 46 47	**28000** **29000** **2A000** **2B000** **2C000** **2D000** **2E000** **2F000**	**RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM** **RAM MCODE DAVID ASSM**	
9	48	**30000**	**HEPAX 16KRAM BL1**	

	49	**31000**	**HEPAX 16KRAM BL1**	
	50	**32000**	**HEPAX 16KRAM BL1**	
	51	**33000**	**HEPAX 16KRAM BL1**	
	52	**34000**	**HEPAX 16KRAM BL2**	
	53	**35000**	**HEPAX 16KRAM BL2**	
	54	**36000**	**HEPAX 16KRAM BL2**	
	55	**37000**	**HEPAX 16KRAM BL2**	
10	56	**38000**	**HEPAX 32KRAM**	
	57	**39000**	**HEPAX 32KRAM**	
	58	**3A000**	**HEPAX 32KRAM**	
	59	**3B000**	**HEPAX 32KRAM**	
	60	**3C000**	**HEPAX 32KRAM**	
	61	**3D000**	**HEPAX 32KRAM**	
	62	**3E000**	**HEPAX 32KRAM**	
	63	**3F000**	**HEPAX 32KRAM**	
11	64	**40000**		
	65	**41000**		
	66	**42000**		
	67	**43000**		
	68	**44000**		
	69	**45000**		
	70	**46000**		
	71	**47000**		
12	72	**48000**		
	73	**49000**		
	74	**4A000**		
	75	**4B000**		
	76	**4C000**		
	77	**4D000**		
	78	**4E000**		
	79	**4F000**		
13	80	**50000**		
	81	**51000**		
	82	**52000**		
	83	**53000**		
	84	**54000**		
	85	**55000**		
	86	**56000**		
	87	**57000**		
14	88	**58000**		
	89	**59000**		
	90	**5A000**		
	91	**5B000**		
	92	**5C000**		
	93	**5D000**		
	94	**5E000**		
	95	**5F000**		
15	96	**60000**		
	97	**61000**		
	98	**62000**		
	99	**63000**		
	100	**64000**		
	101	**65000**		
	102	**66000**		

	103	**67000**		
16	104 105 106 107 108 109 110 111	**68000** **69000** **6A000** **6B000** **6C000** **6D000** **6E000** **6F000**		
17	112 113 114 115 116 117 118 119	**70000** **71000** **72000** **73000** **74000** **75000** **76000** **77000**		
18	120 121 122 123 124 125 126 (SR)	**78000** **79000** **7A000** **7B000** **7C000** **7D000** **7E000** **7F000**		

Setting Registers

Setting Register Layout									
Bit 9	***Bit 8***	***Bit 7***	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
EN	0=SRAM	A19	A18	A17	A16	A15	A14	A13	A12
EN	1=FLASH	IO	WP	A17	A16	A15	A14	A13	A12

Bits	Value	Meaning
9	0 1	FLASH/SRAM/IO Bank is disabled (disregard others) FLASH/SRAM/IO Bank is enabled (consider others)
8	0	ROM Image is in FLASH
7 to 0	Any valid	A19 to A12 address ROM images in FLASH memory
8	1	ROM Image is in SRAM
7	0 1	Bank is SRAM Bank is I/O
6	0 1	Write Enabled (SRAM only) Write Protected (SRAM only)
5 to 0	Any valid	A17 to A12 address ROM images in SRAM memory

SR mapping (FLASH)

These setting are intended to daily use and are not necessarily meant to be changed constantly.

	SR0		SR1	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	X-FUNCTIONS (25)	27 #1B
6				
7			ADV_APP 1A (19)	203 #CB
8 (1L)			ADVANTG 1B (22)	200 #C8
9 (1H)			ADVANTG 1B (24)	201 #C9 202 #CA
A (2L)			CCDL 1B (09)	144 #90
B (2H)			CCDH 2B (11)	145 #91
C (3L)			PPC	149 #95
D (3H)			PPC	150 #96
E (4L)			AECROM 1A (18)	154 #9A
F (4H)			AECROM 1A (18)	155 #9B

	SR2		SR3	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	X-FUNCTIONS (25)	27 #1B
6				
7				
8 (1L)	SANDMATH 3A (03)	158 #9E	CCDL 1B (09)	144 #90
9 (1H)	SANDMATH 2B (02)	159 #9F	CCDH 2B (11)	145 #91
A (2L)	PETROL 1A (15)	232 #E8	MATH 1D (01)	242 #F2
B (2H)	PETROL 2A (16)	233 #E9	STAT 1B (02)	217 #D9
C (3L)	STRUCT 1B (07)	222 #DE	SURVEY 1B (03)	218 #DA
D (3H)	STRUCT 1A (19)	237 #ED	CIRCUIT 1A (06)	221 #DD
E (4L)	STRESS 1A (08)	223 #DF	MACHINE 1A (12)	228 #E4
F (4H)	AVIATION 1A (19)	234 #EA	THERM 1A (13)	229 #E5

Configuration for ADVANTAGE 1B

Files (ROM) – ADVL1 IB.ROM, ADVU1 IB.ROM, ADVU2 IB.ROM,

Bank	Port low	(address)	Port high	(address)
0	ADVL1 1B	200 #C8	ADVU1 1B	201 #C9
1	none	255 #FF	none	255 #FF
2	none	255 #FF	ADVU2 1B	202 #CA
3	none	255 #FF	none	255 #FF

SR mapping (SRAM)

	SR0		SR1	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	MLDL ERAMCO (21)	121 #79
6				
7	TOOLBOX 4U (13)	116 #74	NFCROM 1B (17)	117 #75
8 (1L)	DAVIDASM 2C (02)	104 #68	SANDBOX 3D (08)	114 #72
9 (1H)	DAVID LABELS	105 #69	SANDBOX 3D (13)	115 #73
A (2L)	SRAM MLDL	40 #28	SRAM MLDL	40 #28
B (2H)	SRAM MLDL	41 #29	SRAM MLDL	41 #29
C (3L)	HEPAX 4kRAM	48(#30)/56(#38)	SRAM MLDL	42 #2A
D (3H)	HEPAX 4kRAM	49(#31)/57(#39)	SRAM MLDL	43 #2B
E (4L)	HEPAX 4kRAM	50(#32)/58(#3A)	SRAM MLDL	44(#2C)/46(#2E)
F (4H)	HEPAX 4kRAM	51(#33)/59(#3B)	SRAM MLDL	45(#2D)/47(#2F)

	SR2		SR3	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	HEPAX 1D (07)	108/110/109/111
6				
7	TOOLBOX 4U (13)	116 #74	DAVIDASM 2C (02)	104 #68
8 (1L)	DAVIDASM 2C (02)	104 #68	HEPAX 4kRAM	48(#30)/56(#38)
9 (1H)	DAVID LABELS	105 #69	HEPAX 4kRAM	49(#31)/57(#39)
A (2L)	SRAM MLDL	42 #2A	HEPAX 4kRAM	50(#32)/58(#3A)
B (2H)	SRAM MLDL	43 #2B	HEPAX 4kRAM	51(#33)/59(#3B)
C (3L)	HEPAX 4kRAM	52(#34)/5A(#3C)	HEPAX 4kRAM	52(#34)/5A(#3C)
D (3H)	HEPAX 4kRAM	53(#35)/5B(#3D)	HEPAX 4kRAM	53(#35)/5B(#3D)
E (4L)	HEPAX 4kRAM	54(#36)/5C(#3E)	HEPAX 4kRAM	54(#36)/5C(#3E)
F (4H)	HEPAX 4kRAM	55(#37)/5D(#3F)	HEPAX 4kRAM	55(#37)/5D(#3F)

Configuration for HEPAX 1D

Files (ROM) – HEPAXH1.ROM, HEPAXH2.ROM, HEPAXH3.ROM, HEPAXH4.ROM
Page: 07 (HPIL)

Bank	P. #7 (address)	Bank	P. #7 (address)
0	HEPAXH1 108 #6C000	1	HEPAXH3 110 #6E000
2	HEPAXH2 109 #6D000	3	HEPAXH4 111 #6F000

HEPAX RAM FILES:

All banks on each page must be addressed to the related RAM location. Addresses for HEPAX RAM pages are shown below.

32KRAM scheme:

PAGE	RAM addr	PAGE	RAM addr	PAGE	RAM addr	PAGE	RAM addr
8		9		A		B	
C		D		E		F	

16KRAM scheme:

PAGE	RAM addr	PAGE	RAM addr	PAGE	RAM addr	PAGE	RAM addr
8		9		A		B	

MCODE DEVELOPMENT LAB
SRAM setting registers
ONLY 41C/CV – USES PAGE #5

	SR0		SR1	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	MLDL ERAMCO (21)	121 #79
6				
7	TOOLBOX 4U (13)	116 #74	NFCROM 1B (17)	117 #75
8 (1L)	DAVIDASM 2C (02)	104 #68	CCDL 1B (09)	144 #90
9 (1H)	DAVID LABELS	105 #69	CCDH 2B (11)	145 #91
A (2L)	SRAM MLDL	40 #28	SRAM MLDL	40 #28
B (2H)	SRAM MLDL	41 #29	SRAM MLDL	41 #29
C (3L)	HEPAX 4kRAM	48(#30)/56(#38)	SRAM MLDL	42 #2A
D (3H)	HEPAX 4kRAM	49(#31)/57(#39)	SRAM MLDL	43 #2B
E (4L)	HEPAX 4kRAM	50(#32)/58(#3A)	SRAM MLDL	44(#2C)/46(#2E)
F (4H)	HEPAX 4kRAM	51(#33)/59(#3B)	SRAM MLDL	45(#2D)/47(#2F)

	SR2		SR3	
Page	ROM ID	#ADDR	ROM ID	#ADDR
5	M2K ROM 9C (21)	0 #00	HEPAX 1D (07)	108/110/109/111
6				
7	TOOLBOX 4U (13)	116 #74	DAVIDASM 2C (02)	104 #68
8 (1L)	DAVIDASM 2C (02)	104 #68	HEPAX 4kRAM	48(#30)/56(#38)
9 (1H)	DAVID LABELS	105 #69	HEPAX 4kRAM	49(#31)/57(#39)
A (2L)	SRAM MLDL	42 #2A	HEPAX 4kRAM	50(#32)/58(#3A)
B (2H)	SRAM MLDL	43 #2B	HEPAX 4kRAM	51(#33)/59(#3B)
C (3L)	HEPAX 4kRAM	52(#34)/5A(#3C)	HEPAX 4kRAM	52(#34)/5A(#3C)
D (3H)	HEPAX 4kRAM	53(#35)/5B(#3D)	HEPAX 4kRAM	53(#35)/5B(#3D)
E (4L)	HEPAX 4kRAM	54(#36)/5C(#3E)	HEPAX 4kRAM	54(#36)/5C(#3E)
F (4H)	HEPAX 4kRAM	55(#37)/5D(#3F)	HEPAX 4kRAM	55(#37)/5D(#3F)

The arrangement shown above allows as many data handling possibilities as I could think of, with the memory editing ROM images I have (in time: does anybody have a ZENROM image to share? QRG for ZENROM and Angel´s TOOLBOX are welcome, too). Please, observe that all addresses pointed in HADDR column refer to my own MLDL2059 unit, they are from my own FLASH and SRAM mapping, so yours will be surely different.

While working with SR3 configuration, all 32KRAM are available to HEPAX and DAVID ASSM (there are two 32K RAM blocks in MLDL2000 already set to work in the specified SRAM addresses). At any moment, in order to work with HEPAX data (either transfer to or from another development environment), SR0 and SR2 offer the same ROM images to work with (M2K ROM, Angel´s TOOLBOX, and DAVID Assembler), although addressing two different HEPAX RAM area: SR0 with the lower 16KRAM and SR2 with the upper 16KRAM. At the same time, two 4K RAM blocks in SRAM (buffers, one for SR0 and another for SR2) are addressed, so data can be transferred without problems. SR1 allows simultaneous access to data in both SRAM blocks available in SR0 and SR2, and also offers a third set of two 4K SRAM blocks.

This is the first arrangement I thought about. I'd like to read about others ideas, experiences.

Setting HEPAX RAM contents (reference: Meindert's MLDL2000)

Contents Table

#ADR	X000	XFE7	XFE8	XFE9	XFED	XFEF	XFF1	XFF2	XFF3
CONT	**#ref**	**prev**	**next**	091	090	091	0E5	00F	200

X is any value from #8h to #Fh, according to the port occupied by the RAM block in calculator's memory. Contents in shaded areas vary according to their meaning (below).

#ref a unique reference for this RAM contents (CAT **2** sees XROM data). Usually starts with #B.
prev reference to the previous RAM block in chain, as system ports (8, 9, etc.). If first, set to 0. If sequenced chain, use (X-1)
nex reference to the next RAM block in chain, as system ports (8, 9, etc.) If last, set to 0. If sequenced chain, use (X+1)

Example: The references below indicate the contents for some known arrangements. Shaded values indicate first and last RAM blocks in chain. In all cases, the following contents are repeated (X= port #)

```
XFE9: 091          XFEF: 091          XFF2: 00F
XFED: 090          XFF1: 0E5          XFF3: 200
```

16K on ports #8 to #B.

```
8000: 00B          9000: 00C          A000: 00D          B000: 00E
8FE7: 000          9FE7: 008          AFE7: 009          BFE7: 00A
8FE8: 009          9FE8: 00A          AFE8: 00B          BFE8: 000
```

32K on ports #8 to #F.

```
8000: 00B          9000: 00C          A000: 00D          B000: 00E
8FE7: 000          9FE7: 008          AFE7: 009          BFE7: 00A
8FE8: 009          9FE8: 00A          AFE8: 00B          BFE8: 00C

C000: 00F          D000: 010          E000: 011          F000: 012
CFE7: 00B          DFE7: 00C          EFE7: 00D          FFE7: 00E
CFE8: 00D          DFE8: 00E          EFE8: 00F          FFE8: 000
```

Obs: it is possible to configure to any number of blocks in any chain sequence. For example, if you want only two blocks of 4KRAM, each one in ports #B and #F:

```
B000: 00B          BFE8: 00F          FFE7: 00B
BFE7: 000          F000: 00C          FFE8: 000
```