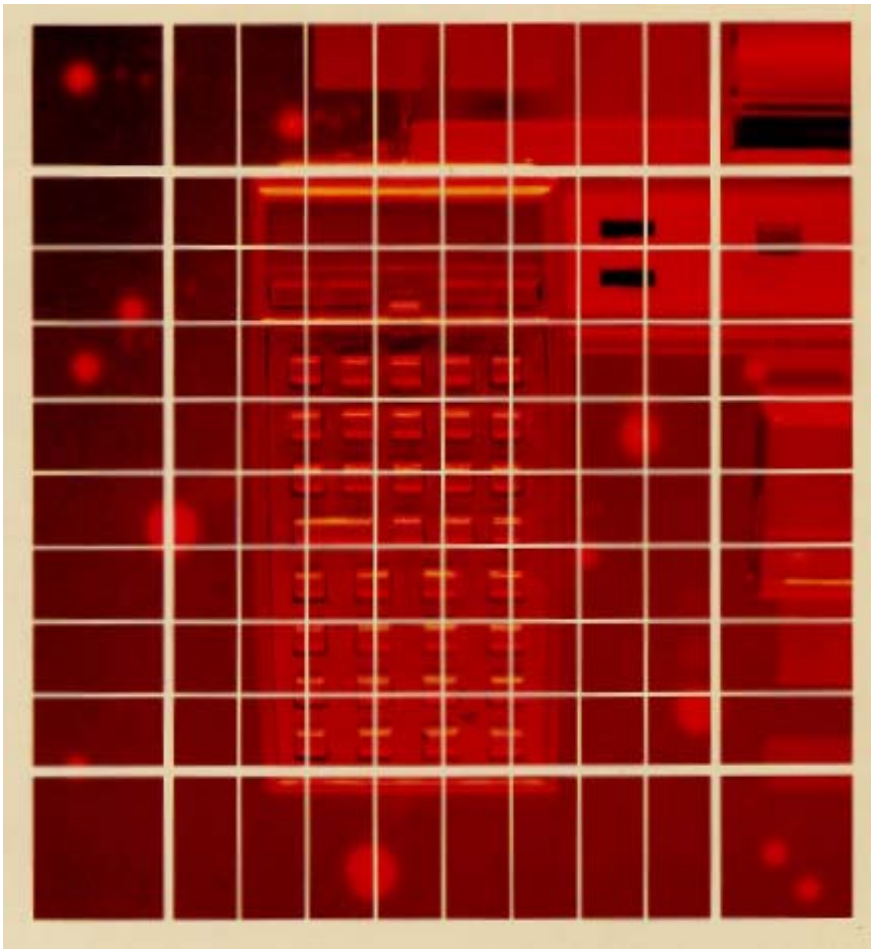
LIBRARY #4

HP-41 Extension Project

Development Notes and Usage Recommendations



Programmed by Ángel M. Martin

May 2012

This compilation revision 4.01.02

Screen captures taken from V41, Windows-based emulator developed by Warren Furlow.
See www.hp41.org

Acknowledgment.- The author wishes to thank the developers of the systems capable of housing the Library#4 project components:

- Diego Díaz – Clonix/NovRAM Modules;
- Meindert Kuipers, MLDL-2000; and
- Monte Dalrymple, 41-CL board.

Page#4 Project - Extensions to the HP-41

Table of Contents.

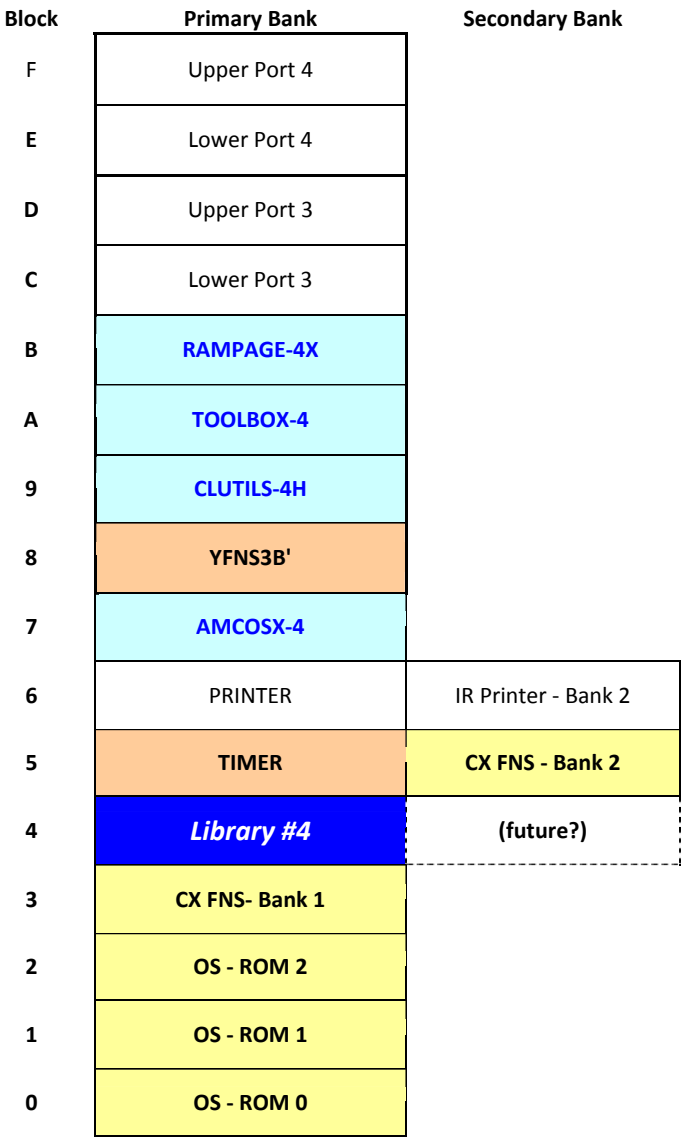


Figure 1.- Typical System Extension setup, corresponding to a 41CL where page#7 is used by the AMC OS/X instead of the HP-IL. Blue background cells denote Library#4-aware modules.

Library#4 Project - Extensions for the HP-41

1. Introduction.

Low and behold, 33+ years after the release of the HP41 (and more than 20 after its obsolescence!), after all the creativity, innovation, refinement and sheer amount of extensions and improvements over the original developed in those years; comes now yet another extensions project, cryptically named "**Library#4**" - no doubt you wonder what else can this be, and if there's no end to this and whether this platform can still be enhanced.

Well, the Library#4 project is not a radical extension, but more of an optimization of the architecture. Its name comes from the utilization of page#4 in the HP41 address bus as repository for a routine library module, to be accessed by many other ROMS – with the obvious room savings and additional consistency benefits derived form such an approach.

Page#4 is a system-reserved 4k block where only dedicated modules can be located. The 41 OS checks for the existence of any code at key locations within page#4 during critical process and events, like upon system cold/warm start-ups and each time there's a I/O interrupt. The original intent was to be used by diagnostics and other take-over modules, like HP's own service ROMS. Soon enough a few 3rd. parties found other usages for this, like the FORTH and LaitRAM ROMS – both exploiting the advanced capabilities that such privileged position would offer. These modules were written specifically to reside there, so they'll be conforming to the OS design.

Page#4 has a few more limitations though: it's very "invisible" to the OS in that there cannot be a FAT structure, nor does it support the polling points functionality – which starts with page#5, the Timer Module reserved one. So its practical use is somehow "limited", yet the challenge to take advantage of those almost-always empty 4k was impossible to resist!

From its inception, this idea has been the design goal of the project: to put all that empty space to a better use, gaining consistency and convenience on the way. Some notable mentions are:

- Same error handling and exception checking used for many functions
- Data input/output routines, common across multiple programs
- Lengthy code sections available where they don't take up vital space in the ROMs themselves.

The Library#4 module is not a take-over ROM, so the 41 OS is always in charge. It effectively deflects the OS polls and checking as though it weren't there – which for all practical purposes such is the case. It comes alive when routines are called from other modules – and that's the beauty of the thing: being always at the same, fixed locations it's a programmers dream to call them and access them, almost as if it was part of the OS itself.

A word of caution: *Page#4 project is not for the casual user.*

First off, it requires special hardware to house it – three examples are: Clonix/NoVRAM modules, the MLDL-2000, and the 41CL board. Second, its benefits are best realized when using the modules in combination with one another, even if there is no dependencies other than the presence of the Library#4 ROM itself always on-line.

Warning: Library#4 will conflict with a disabled HP-IL Printer (switched OFF in the HP-IL module). This is so because the OFF position changes the hard-coded address from page 6 to page 4, thus occupying the space already. Make sure this is not the case on your system!

2. Serving the needs of other ROMS

The main objective of any routine library is to be there when other functions or programs need them – and this also applies to the library#4 project. So just by itself it doesn't add any functionality at all, no configuration checks, no splash messages… it just stands there and does nothing. Deceivingly simple, but this of course changes as soon as you plug in any Library#4-aware module.

Common features of these are as follows:

1. Checking for the Library#4 presence upon system start-up, putting up a warning message if not there. At that point the user should NOT continue to use the specific module, or unpredictable results (of various consequences) will occur.

2. Also checking for the CX-OS version at start-up, which implies the Library#4 it's not compatible with CV or C machines (it uses CX addresses in multiple locations, to squeeze even more space and optimize things).

3. Relying on the library as storage place for many routines and even full function code segments, to the extent that in many instances all there is in the actual module is a FAT entry and a code snippet required to transfer the execution there.

So it's now clear that to capitalize on the Library#4 benefits the "client" module must be written with its design in mind. It comes without saying that dedicated versions of a few such modules have been prepared and are also part of the Library#4 project, such as:

1. **AMC_OSX.** Based on the OS Extensions from the CCD Module, this 4k ROM extends even further the original implementation. It incorporates the Multi-byte Assign function (ASG) and Prompt lengthener from the ML ROM; *it has full support of the extended CAT'4* (sorely missing in the CCD OS/X), adding extra support for five new file types! All this is possible within a 4k module only because of the intimate connection with the Library#4.

2. **TOOLBOX_4**. A new version of the classic, adding powerhouse HEX editor functions from the DISASM ROMs – with extended features to improve on its user interface, borrowed from the HEPAX HEXEDIT.

3. **RAMPAGE_4X**. Also a new version of the RAMPAGE module, with a truckload of RAM, Buffer and EM related functions that build upon its non-Library#4 already powerful predecessor. Don't miss the updated buffer catalog, or the GET/SAVE keys/buffer functions - a joy to behold.

4. **CLUTILS_4H**. More and improved CL-related functions offering a comprehensive integration with the CL board MMU and advanced features – with improved HEPAX configuration, additional system checks and catalogs - a double jeopardy for CL owners.

There are many functions included in these modules, with most areas covered providing access to multiple advanced features. The advantage of doing things now is one can pick and choose the best of all and even allow the luxury to improve on it – literally "getting on shoulders of giants".

The design criteria have been to avoid unnecessary redundancy, so a given function is only present in one module. This is strictly the case for the first 3 listed above, and most of the CLUTILS as well. **Appendix 2** has a complete listing of the functions, showing their distribution on each module by category and functionality group.

3. Identifying the Library.

Being such a silent companion, how can you tell whether the Library#4 is present on the system?

For one thing you'll have negative confirmation when it's not there, since each of the library-capable modules has a checking routine that runs upon the calculator ON event. Should the Library#4 be missing the message "NO LIBRARY" will be shown, and the initialization will be stopped – no further modules in the sequence.

Note: adding a module on V41 other than the HEPAX doesn't require re-initializing the calculator, so it's possible to bypass this protection. Always make sure you power-cycle the application to be safe.

The code below - called by the CALC_ON interrupt - shows the checking routine. Note that it is included in each module, as obviously cannot be place on the library itself. For the same reason there are no calls to the Library, so everything is done using calls to the standard OS.

LB4CHK	A56D	04E	C=0 ALL	
	A56E	15C	PT= 6	
	A56F	110	LD@PT- 4	*put "4000" in ADR field*
	A570	330	FETCH S&X	
	A571	106	A=C S&X	*put byte in A[S&X]*
	A572	130	LDI S&X	
	A573	***023***	CON:	*signature value*
	A574	366	?A#C S&X	*are they different?*
	A575	***3A0***	***?NC RTN***	*no, return.*
	A576	*3C1*	*?NC XQ*	*Enable & Clear Disp*
	A577	*0B0*	*->2CF0*	*[CLLCDE]*
	A578	*3BD*	*?NC XQ*	*Message Line*
	A579	*01C*	*->07EF*	*[MESSL]*
	A57A	*00E*	***"N"***	
	A57B	*00F*	***"O"***	
	A57C	*020*	***" "***	
	A57D	*00C*	***"L"***	
	A57E	*009*	***"I"***	
	A57F	*002*	***"B"***	
	A580	*012*	***"R"***	
	A581	*001*	***"A"***	
	A582	*012*	***"R"***	
	A583	*219*	***"Y"***	
aperex	A584	*3DD*	*?NC XQ*	*Left Justified format*
	A585	*0AC*	*->2BF7*	*[LEFTJ]*
	A586	108	SETF 8	
	A587	*201*	*?NC XQ*	
	A588	*070*	*->1C80*	*[MSG105]*
	A589	*3ED*	*?NC GO*	*HALT execution*
	A58A	*08A*	*-> 22FB*	*[ERR110]*

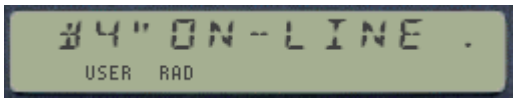
There's also another check for the CX-OS, this time performed within the Library code, and accessed from each module with just a simple ?NCXQ call. No news is good news, so there'll be no disruption when all is ok and the system is ready to operate properly.

In addition to this, there are a few functions available on the modules that either report its presence directly, or list it as part of the complete system configuration. The header functions are:

-RAMPAGE 4X' in the RAMPAGE4 module, XROM 17,00
-TOOLBOX4X' in the TOOLBOX4 module, XROM 13,00
-**CLUTILS 4H'** in the CLUTLS4H module, XROM 12,00

You'd notice it's always the first function in the FAT, thus easy enough to remember. To execute it you can use the extended **XEQ** function form the **AMC_OSX** module, followed by ENTER^, XEQ, xrom#, 00.

The output of these functions is a rather fancy display demo, originally written by Nelson F. Crowle and adapted to this project; make sure you don't miss it.



Note: for unknown reasons V41 hangs up during the execution of this function, so don't use it in the emulator. It is perfectly safe to run in the real environments.

The other functions that reveal the presence of the Library are:

BLCAT, the block catalog will clearly show page#4 with the unmistakable description:

ROMLST, the ROM List function produces a string with all the XROM ID's currently present on the system in sequential page order. Library#4 has XROM#35 (yes, that's beyond the allowed range which ends at 32, more on this later), which will be listed if present – like in the example below, listing the **X-Functions**, **Library#4**, **Timer**, and **TBOX4** modules:

Note: The output is limited to 24 characters, so only the last 8 ROM ID's will be shown (plugged in the higher addresses) – This means it's possible that for full configurations the lower-location modules are not shown.

PGREV – This function retrieves the page revision string from the last four bytes of the page. Input the location "04" to obtain the following:

Now you know how to check for the Library, which is well characterized in at least 4 different accounts - should any shadow of a doubt crops up or your system is acting up a little weird.

So how come is the Library#4 seen by these functions, and even given a "proper" name?

The answer is in the code written at the top of the page: the objective is to deflect the OS check, but it's been done so that it's compatible with the normal operation of the functions:

4000	023	JNC +04	signature= 023; xrom 35,:-)
4001	00B	JNC +01	DUMMY!
4002	007	JNC +00	Dummy Header
4003	00B	JNC +01	TITLE0
4004	3E0	RTN	deflect the call

From an execution standpoint, the (apparently arbitrary) jumps quickly return to the calling OS point (RTN instruction at location 0x4004) – yet the first 4 bytes can also be interpreted as XROM id# 35, *not conflicting by definition with *any* other module* – as number of functions (12), and proper address of the first function in a fictitious FAT, at location 0x470B. That's where the code for the header "**-LIBRARY 4**" resides. Clever perhaps, but simple when you think about it.

Note also that the return to the OS happens in every circumstance, regardless of whether it's called by a DEEP or LIGHT sleep event – which to the author's knowledge are the only two applicable instances. This is a safer approach than trying to use the DEEP sleep for other purposes, such as a COLD start initialization - and has the added advantage of being faster.

Here are the code snippets from ROM-0 in the OS:

BOOT1	0000	201	?NC GO	
	0001	006	->0180	[LSWKUP]
BOOT2	0002	2B5	?NC GO	
	0003	006	->01AD	[DSWKUP]

And their follow ups, the light-sleep wake-up:

LSWKUP	0180	001	?NC XQ	Take-Over
	0181	100	->4000	[RESET]
	0182	3D5	?NC XQ	
	0183	00C	->03F5	[PACH11]
WKUP10	0184	3CC	?KEY	
	0185	10F	JC +21	WKUP20

and the deep-sleep wake-up:

DSWKUP	01AD	001	?NC XQ	Take-Over
	01AE	100	->4000	[RESET]
	01AF	2E0	DSPOFF	
	01B0	3D5	?NC XQ	
	01B1	00C	->03F5	[PACH11]
	01B2	3CC	?KEY	
	01B3	03F	JC +07	WKUP25

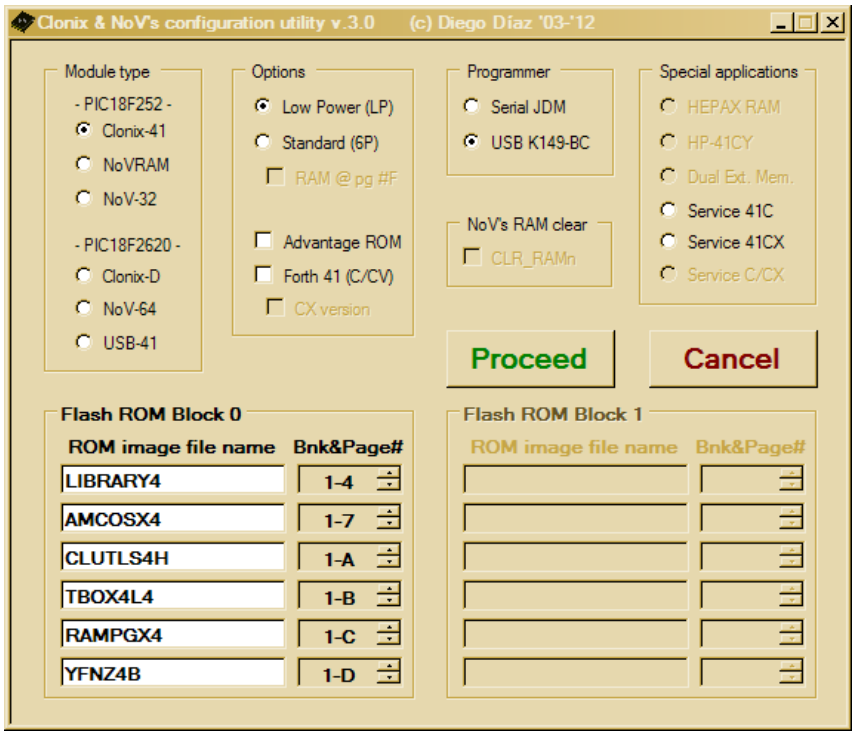
The net result is taking care of these hot locations without interfering with its normal operation, so from the HP-41's perspective there's nothing going on at page#4 – and the OS operation is not affected in any way whatsoever, assuring compatibility in all cases.

4. Mounting the Library – and its friends.

There's really not much involved into a proper configuration, apart from the obvious requirement that the module is configured to page#4. There are a few options available, as follows:

- CL Board - just map the modules using the appropriate MMU functions.
- MLDL2k - configure the settings registers using the MLSL2k manager s/w
- Clonix/NoVRAM – now an easy task using ClonixConfig V3.0

See the ClonixCongif screen capture below for an example *showing also the AMC_OS/X configured on page#7*. This is an optimal arrangement in that it's perfectly compatible and saves an additional outside page for other uses – remember however that it's not compatible with an HP-IL connected to the system.



Using the AMC_OS/X module is highly recommended given its additional capabilities, further extended on its Library#4 implementation. In fact both modules go hand in hand with one another, also leveraging from the X-Function routines in the CX /OS – in an all-new version of the classic powerhouse.

Note 1: To have a printer simulation you can use the USB-41, which only uses page#6 and thus has no conflicts with the set-up. Obviously it's also possible to use the 82143A non-HP-IL thermal printer as well.

Note 2:- If the HP-IL is needed just change the location of the AMC_OS/X module to a non-conflicting page, no matter which one since there are no location requirements and no dependencies exist. If you do, do NOT disable the usage of the HPIL printer (switch on the HPIL module), or its ROM will compete for page#4 usage with the Library#4 (!).

The programs below can be used to move the Clonix configuration programmed (burned) in the example above into the CL board – a convenient method that gives the best of both worlds, use it carefully (you'll likely need to modify the addresses in sRAM) but confidently. Note that pages #8 and #9 are not used, as it's assumed that's' where the current copies of YFNZ and CLUTILS reside.

```
01*  LBL "MAPON4"
02   TURBO50
03   "4-0>83F"
04   AVIEW
05   YMCPY
06   "7-0>83D"
07   AVIEW
08   YMCPY
09   "A-0>83E"
10   AVIEW
11   YMCPY
12   "B-0>814"
13   AVIEW
14   YMCPY
15   "C-0>815"
16   AVIEW
17   YMCPY
18   "D-0>807"
19   AVIEW
20   YMCPY
21   SF 11
22   OFF
23   "804040-883F"
24   YPOKE
25   "83E-RAM"
26   PLUG1U
27   "83D-RAM"
28   PLUGH
29   "814-RAM"
30   PLUG2L
31   "815-RAM"
32   PLUG2U
33   "DONE"
34   AVIEW
35   TONE 0
36   GTEND
37   END
```

Clonix configuration: expected (!)
Page #4 - Library4
Page #7 - AMC_OS/X
Page #A - CLUTLS4H
page #B - TBOX4L4
page #C - RAMPGX4'
page #D - YFNZ4B

will be copied to sRAM
0x83F - Library4
0x83E - CLUTLS4H
0x83D - AMC_OS/X
0x814 - TBOX4L4
0x815 - RAMPGX4
0x807 - YFNZ4B

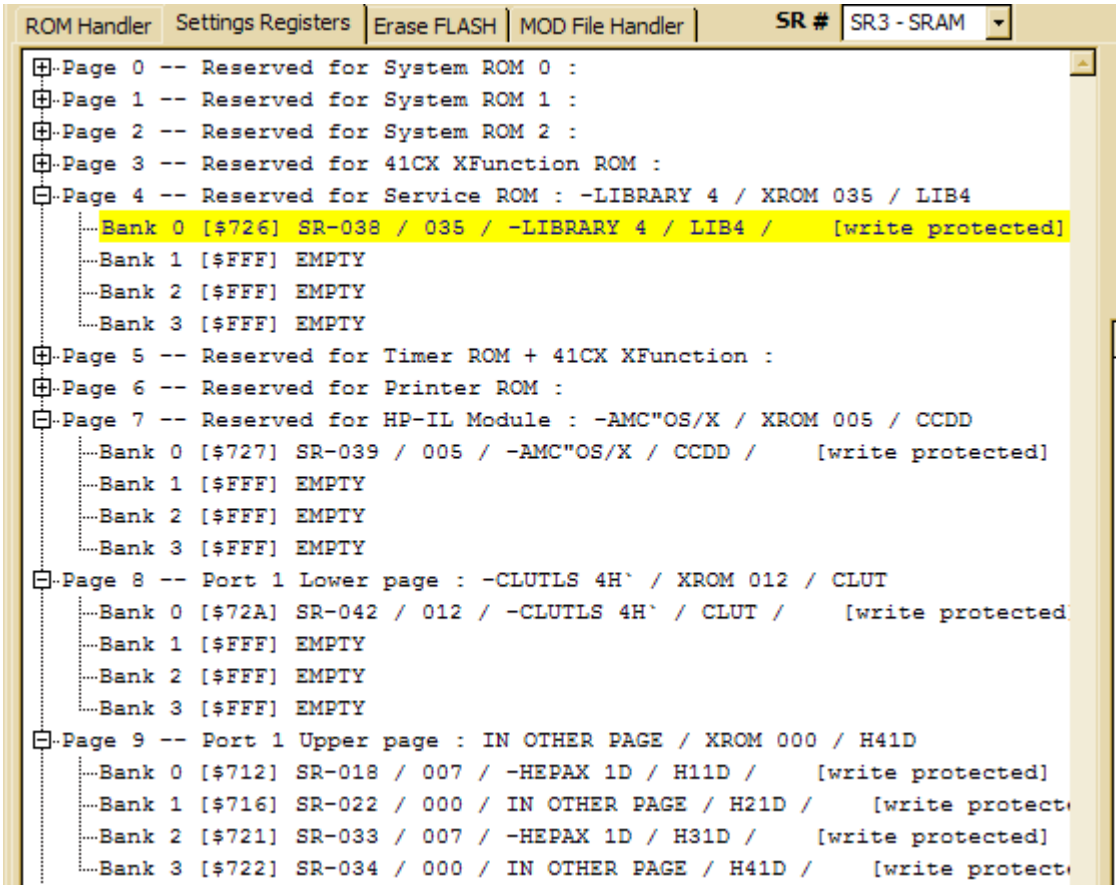
then the calculator will switch OFF. Remove the clonix and switch it ON to map the modules to their final destinations.

```
01*  LBL "MAPOFF4"
02   MMUDIS
03   "YFNZ"
04   PLUG1L
05   UPLUGH
06   UPLUG1U
07   UPLUG2
08   UPLUG3
09   "804040-0000"
10   YPOKE
11   MMUEN
12   END
```

Nothing fancy about the removal of the MMU entries. *You can use this prgram independently of the mapped ram blocks.*

Note 3. **Leave page#6 alone**. The 41 OS assumes it is reserved for the printer, and there are numerous calls to printer routines embedded in the OS ROMs. These will create havoc if you plug any other module in said page. The HEPAX module (in both its original form and as part of the NoVRAM modules) is the only one the author's aware of its compatibility with this location - basically achieved by having the section 0xpFFD to 0xpFF4 unused, as this is where all the extra printer polling-points are located. See the 41 OS VASMS for more details.

The figure below shows the status registers configuration on the MLDL2k manager corresponding to a very similar system set-up, including the HEPAX module as well (albeit not in page#6). Notice how it's all seamlessly supported – as long as there are no real modules connected to the same ports.



Final disclaimer: - It should be mentioned that running without the library would likely just have bogus results, consequence of the blank returns when calling non-existing code. However it may also hang up your system with various degrees of severity – none of them should cause any harm to the calculator, but *the author takes no responsibility and shall not be liable in any circumstance, regardless of how unlikely it might be. So when in doubt, just check for it!*

5. Let me count the times…

… that the same code snippet shows up in different functions, perhaps to perform a micro-operation (such as selecting chip0 and reading the T register), or to manage some house-keeping stuff. This is particularly manifest in math routines, where perhaps the original 41 OS had a lesser attention put into since it was a natural extension of older models, like the 65/67 MCODE.

So while not being glamorous in aspect or spectacular by nature, those common code segments amount to literally dozens of instances where the same code is written and must be maintained independently – adding to hundreds of bytes saved when put together.

Library#4 is therefore a bit of a "*personal revenge*": consider that it only takes 2 bytes to place a call to the fix locations where the routines reside, preserving the contents of C register, not needing to set the CPU in HEX mode… and you see why this approaches programmer's delight!

A couple of examples would illustrate these points:

1. Add one to the number in C

   A simple call to Lib#4 does the trick:

1E1	*?NC XQ*	*Increment C*
100	*->4078*	*[INCC10]*

Where:

INCC	**4076**	*361*	*?NC XQ*	*(this includes SETDEC)*
	4077	*050*	*->14D8*	*[CHK_NO_S]*
INCC10	4078	10E	A=C ALL	*holds sign and S&X*
	4079	02E	B=0 ALL	*clears it*
	407A	0FA	B<>C M	*holds 13-digit mant*
	407B	*001*	*?NC GO*	*13-bit form*
	407C	*062*	*->1800*	*[ADDONE]*

2. Dividing the number in C by 2

   Also a matter of a 2-byte spending and even with 13-digit precision:

3D9	*?NC XQ*	*Calculates C/2*
13C	*->4FF6*	*[DIVBY2]*

Where:

DIVTWO	**4FF3**	**02E**	**B=0 ALL**	*clears B*
	4FF4	0FA	B<>C M	*B holds 13-digit mant*
	4FF5	0AE	A<>C ALL	*A holds sign and S&X*
DIVBY2	**4FF6**	04E	C=0 ALL	
	4FF7	35C	PT=12	***build "2" in C***
	4FF8	090	LD@PT- 2	
	4FF9	*269*	*?NC GO*	
	4FFA	*062*	*->189A*	*[DV1-10]*

3. Calculating MOD(C,2) – handy to see it the number is odd or even.

Assuming there's a positive integer, it's just a matter of making the right call:

3CD	?NC XQ	Calculates C MOD 2
100	->40F3	[MOD2]

Where:

MOD2	40F3	361	?NC XQ	(includes SETDEC)
	40F4	050	->14D8	[CHK_NO_S]
	40F5	088	SETF 5	Take Integer part
	40F6	0ED	?NC XQ	
	40F7	064	->193B	[INTFRC]
	40F8	10E	A=C ALL	
	40F9	04E	C=0 ALL	
	40FA	35C	PT=12	**builds "2" in C**
	40FB	090	LD@PT- 2	
	40FC	070	N=C ALL	save it in N
	40FD	044	CLRF 4	
	40FE	171	?NC GO	Calculates MOD(A,C)
	40FF	066	->195C	[MOD10]

Here including alpha data checks and taking the integer part as well – a thorough job all behind the scenes.

Let's now see a useful example involving the TURBO settings in the CL board. For those (few) times when more speed is a problem, the simple solution is to disable turbo mode, do the displaying an then set it back to fast mode – as can be seen in the following sequence:

37D	?NC XQ	Cancels TURBO mode
13C	->4FDF	[TURBO0]
3A1	?NC XQ	wait a while
13C	->4FE8	[WAIT4]
38D	?NC XQ	Sets TURBO mode
13C	->4FE3	[TURB50]

Where:

TURBO0	4FDF	04E	C=0 ALL	
	4FE0	05C	PT= 4	cancels TURBO mode
	4FE1	210	LD@PT- 8	(so we see it well)
	4FE2	023	JNC +04	
TURB50	4FE3	04E	C=0 ALL	
	4FE4	05C	PT= 4	sets TURBO50
	4FE5	350	LD@R D	
	4FE6	1FC	WCMD	
	4FE7	3E0	RTN	
WAIT4	4FE8	046	C=0 S&X	
	4FE9	2A6	C=-C-1 S&X	Delay loop
	4FEA	266	C=C-1 S&X	
	4FEB	3FB	JNC -01	
	4FEC	3E0	RTN	

Many other examples abound, and really are at the core of this project. Not needing to deal with all the pedestrian code every single instance that is required translates into "lighter" modules, with available space for advanced features – read: function launchers, combined functionality, etc. A definite better position to be in when writing MCODE.

Appendix 1.- Documented Entry Points.

The tables below show the more relevant entry points available in the Library. As this project was born to address the needs of the other modules, it's somehow logical that the selection of the routines was determined by their actual contents - sometimes even in an "ad-hoc" fashion to overcome space constraints limitations, some other times to avoid redundancy of repeated code in multiple locations.

Broadly speaking, there are four main categories of code as follows:

1. Complete Functions – full code, ready to hook-up to from the client module which just has the function name and a transfer GOTO, plus of course a place for it in its FAT structure.
2. Error Messages – frequently accessed from many modules, provides consistency and saves bytes.
3. System Subroutines – subsets of code used in system extension functions.
4. Math Subroutines – subsets of code used in math functions.
5. Modified OS Routines – with small but required changes to make them work as subroutines, typically avoiding the last call to [NRFPU] or similar.

Complete or Quasi-Complete Functions:

ABSP	- 44B9	- Utils	- Alpha Back Space
AINT	- 4293	- Utils	- ARCL Integer Part
ALIST	- 44CF	- Math	- Used for Matrix and Polynomial editors
ANUMDL	- 4200	- Math	- ANUM with Deletion
ASG	- 475D	- Utils	- Multi-byte ASN
ASWAP	- 4272	- Utils	- Alpha swap around Comma (chr code# in N[S&X])
BST+	- 41C1	- Utils	- Auto BST
CDE	- 4480	- Utils	- NNN Code
CHKCFG	- 4143	- Utils	- Check ROM Configuration
CLEM	- 42DD	- Utils	- Clear Extended Memory
CLA-	- 42A7	- Utils	- CLA From Space chr.
CLA>	- 42AA	- Utils	- CLA from ">" chr.
CLAC	- 42AD	- Utils	- CLA from Comma
DCD	- 4497	- Utils	- NNN Decode
DTOA	- 42E6	- Utils	- Display to Alpha
DTST	- 430E	- Utils	- Display Test
MSGE	- 49CE	- Utils	-Mainframe Message
MOD2	- 40F3	- Math	- MOD(C,2)
NOPS	- 4920	- Utils	-Search for NOPS in page
POSA	- 40A7	- Utils	- Position in Alpha, without [NFRPU] call
PRD	- 4644	- Utils	- Print Display
READPG	- 45D5	- Utils	- Read Page from HPIL Disk
REGSORT	- 4887	- Math	- Register Sort
ROMCAT	- 433F	- Utils	- Alternative ROM Catalog
SPEED?	- 4994	- Utils	- CPU cycles
SST+	- 41C1	- Utils	- Auto SST
TF	- 444F	- Utils	- Toggle Flag by X
ΣDGT	- 432C	- Math	- Sum of Mantissa Digits
ST<>A	- 42C3	- Utils	- Swap Stack and Alpha
WRTPG	- 4584	- Utils	- Write page to HPIL Disk
Y/N?	- 41D1	- Utils	- Asks Yes/No input
E3/E+	- 401B	- Math	- Pointer builder
ZOUT	- 4029	- Math	- Output Complex Number

Error / Confirmation Messages:

CONFIG BAD	- 4129, w/ F9 Clear
CONFIG OK	- 4129, w/ F9 Set
DUP BUF	- 43CF
DUP XROM	- 4167
END OF BUF	- 43DA
NO _	- 43C8
NO CX/OS	- 4440
NO BUF	- 43D5
NO BUFFERS	- 43EA
NO HPIL	- 442F
NO KEYS	- 441F
NO PRINTER	- 461B
NO ROM	- 4126
NO XMEM	- 4400
NOT OK	- 4408
OK? Y/N	- 49E6
SPLASH	- 4717

NoVRAM-reserved area.

Sector 0x4100 to 0x410F is reserved for NoVRAM, NoV-32, and NoV-64 usage.

Other general comments.

Remember that page#4 does not offer any support for the polling points, thus all functionality associated to those events must be invoked directly by the client modules.

Code reuse is not only good because it saves room, but having it centralized in a single location allows for easier maintenance, enhancement, and immediate deployment to all functions that use it.

With about 2k still available, the Library ROM is not filled up yet. It's a living project and will have additions to the existing content, specially on the Math department – but there won't be changes to the current scheme to maintain compatibility.

Even in the unlikely event that it eventually fills up, there will be possible to use a bank-switched Library design for yet another 4k of memory, following the same BS conventions used in the Advantage Pac.

Appendix 2.- Function Inventory.

The following pages show all functions by group, with details of the module inclusion and other remarks. Refer to the specific module manual for additional details and instructios.

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ΣRG?			√		RAM	Finds Location of Stat Registers	none	Reg# in X		*Nelson Crowle*
7P<>S				√	RAM	Swap Primary and Secondary Regs	none	{R00-R10} <> {R11-R21}	*from Card Reader ROM*	*HP Co.*
A<>RG _ _				√	RAM	Swaps Alpha and Registers	First RG# in Prompt	Registers swapped		*Ángel Martin*
ARCLIP _ _				√	RAM	ARCL Integer Reg#	Reg# in prompt	Appends Integer part to Alpha		*Ángel Martin*
CLMM				√	RAM	Clear Main Memory	none	Main Memory Cleared		*Zengrange*
CLRAM				√	RAM	Clear RAM	none	All Ram cleared		*R. del Tondo*
NRCLX _ _ _				√	RAM	Non-Normalized RCL	Reg# in prompt	Register Content in X		*Sid Kelly*
PEEKR				√	RAM	Peek absolute register	address in X	content in X		*W&W GmbH*
POKER				√	RAM	Poke absolute register	data in ??	Stores data in register location		*W&W GmbH*
RAMED				√	RAM	RAM Editor	none	RAM Editor Active		*Zengrange*
RAMEDIT				√	RAM	RAM Editot	none	RAM Editor Active		*Håkan Thörngren*
RCLBM				√	RAM	RCL Byte by M	byte addr in M	Recalls byve value		*Mark Power*
ST<>Σ				√	RAM	Stat Reg and Stack swap	none	Registers swapped		*Nelson Crowle*
ST<>A				√	RAM	Alpha and Stack swap	none	Registers swapped		*Ken Emery*
ST<>RG _ _				√	RAM	Stack and Reg# swap	Reg# in prompt	Registers swapped		*Ángel Martin*
STOBM				√	RAM	STO Byte by M	byte addr in M	Stores byte		*Mark Power*
"VREG"				√	RAM	View Registers	bbb,eee in X	Shows Registers contents	*FOCAL program*	*Ángel Martin*
X<>BM				√	RAM	Swaps X and Byte by M	byte addr in M	Swaps bytes		*Mark Power*
X<>aNN				√	RAM	Swaps X and RG#	data in X and Register	Swaps registers contents	*in Y for PRGM*	*Nelson Crowle*
X<I>Y				√	RAM	Indirect Exchange X<>Y	R2# in Y, R1# in X	Registers exchanged		*Nelson Crowle*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ADR? _ _ _ _			√		MCODE	Address Encoder	Address in prompt	2-byte encoded string in X	*fom DISASM_4C ROM*	*Doug Wilder*
BCDBIN			√		MCODE	Converts BCD to BIN	Decimal number in X	Binary NNN in X		*Ken Emery*
BINBCD			√		MCODE	Binary to BCD	Binary NNN in X	Decimal number in X		*Ken Emery*
CGO _ _ _ _			√		MCODE	?C GO instruction encoder	Address in prompt	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
CHKROM _ _			√		MCODE	Check ROM	ROM id# in prompt	Checks valid checksum		*HP Co.*
CODE	√	√			MCODE	Codes Alpha to NNN	Digits in Alpha	Encoded NNN in X		*Ken Emery*
COMPILE			√		MCODE	Compile Program	Program LBL in Alpha	Jump distances compiled	*from MLROM module*	*Frits Ferwerda*
CPYPG _			√		MCODE	Copy Page	Source Page in X	Destination Page in prompt		*Ángel Martin*
CXQ _ _ _ _			√		MCODE	?C XQ Instruction Encoder	Address in prompt	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
DCD	√	√			MCODE	Decode X	NNN in X	Digits in Alpha		*W&W GmbH*
DCODE _ _ _			√		MCODE	Decode Register	Rg# in Prompt	Digits in Alpha	*from MLROM module*	*Frits Ferwerda*
"DISSST"			√		MCODE	SST Disassembler (HEPAX needed)	Address in prompt	Instructions disassembled (HEPAX)	*FOCAL program*	*VM Electronics*
FDATA _			√		MCODE	Function Data	FNAME in prompt	Shows Function Data	*FNAME in Alpha in PRGM*	*Klaus Huppertz*
GETW			√		MCODE	Get Word	Decimal addr in X	word value in X (NNN)		*Rafa Lorente*
HEX>VSM _			√		MCODE	Hex to Oct (VASM code)	Address in prompt	VASM code in display		*Clifford Stern*
HEXIN _			√		MCODE	HEX Input	none / Alpha text	Activates HEX keyboard		*Håkan Thörngren*
HXENTRY _			√		MCODE	HEX Input	none / Alpha text	Activates HEX keyboard		*Clifford Stern*
JC			√		MCODE	Jump if Carry encoder	jump distance	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
JNC			√		MCODE	Jump if NoCarry encoder	jump distance	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
NCGO _ _ _ _			√		MCODE	?NC GO Instruction Encoder	Address in prompt	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
NCXQ _ _ _ _			√		MCODE	?NC XQ Instruction Encoder	Address in prompt	Encoded instruction data	*fom DISASM_4C ROM*	*Doug Wilder*
NOPS			√		MCODE	NOPS location in ROM	Address RANGE in X	Finds location with >= 2 NOPS	*from MLROM module*	*Frits Ferwerda*
OSREV			√		MCODE	OS Roms revisions	none	Revisions in Display		*Nelson Crowle*
PC<>RTN			√		MCODE	PRGM and RTN Pointers swapped	none	Pointers swapped		*W&W GmbH*
PG?			√		MCODE	Page Info	Page# in prompt	1st. FNC in Alpha; NN in X, #FNS in Y		*W&W GmbH*
PGREV _ _	√		√		MCODE	Page Revision	Page# in prompt	Pg# rev in Alpha		*Ángel Martin*
REVPG?	√				MCODE					*Ángel Martin*
ROM? _ _			√		MCODE	ROM info	XROM id# in prompt	1st. FNC in Alpha; pg# in X, #FNS in Y	*from MLROM module*	*Frits Ferwerda*
ROMED _ _ _ _			√		MCODE	ROM Editor	Address in prompt	Hex Editor mode active	*uses: SST, BST, BA, ENTER^*	*Doug Wilder*
SUMPG _			√		MCODE	Sum Page	Page# in prompt	Checksum re-written	*RAM pages only*	*George Ioannou*
VSM>HEX _/_/_			√		MCODE	VASM to HEX	Data in prompts	HEX address in Display		*Clifford Stern*
X>$		√			MCODE	Number to Alpha Data	Number in X	NNN in X		*VM Electronics*
XQ>XR			√		MCODE	Convert XEQ to XROM	PRGM Name in Alpha	Changes WEX calls to XROM		*W&W GmbH*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ARCLCHR				√	XMEM	ARCL Character	FileName in Alpha	Recalls current chr to Alpha		*Håkan Thörngren*
CLEM	√	√			XMEM	Clear Extended Memory	none	Deleteds XMEM master register		*Håkan Thörngren*
CLXM				√	XMEM	Clear Extended Memory	none	Deleteds XMEM master register		*Zengrange*
FLHD				√	XMEM	File Head Location	FileName in Alpha	Header location addr. in X		*Ángel Martin*
FLTYPE				√	XMEM	File Type Finder	FileName in Alpha	Fily type in X		*Ángel Martin*
FLNG?				√	HPIL	HPIL Drive File Length	FileName in Alpha	File Length in X		*Ken Emery*
GETBUF				√	XMEM	Get buffer from XM	FileName in Alpha	Restores a buffer from XMEM		*Håkan Thörngren*
GETKA				√	XMEM	Get Keys from XM	File Name in Alpha	Key Assignments made		*Håkan Thörngren*
MRGKA				√	XMEM	Merge Keys from XM	File Name in Alpha	Key Assignments merged		*Håkan Thörngren*
READDF			√		HPIL	Read XM Data File from HPIL Drive	FileName in Alpha	Reads HPIL File to XMEM	*XM Data File must Exist!*	*R. del Tondo*
READPG	√	√			HPIL	Read HPIL Drive File into page#	FileName in Alpha, pg# in X	Reads File into pg# in X	*in X for PRGM*	*R. del Tondo*
READXM				√	HPIL	Read All XM content from HPIL Drive	FileName in Alpha	Reads File into X-Memory		*Ken Emery*
RENMFL				√	XMEM	Rename File	"OLD,NEW" in Alpha	File Renamed		*Ángel Martin*
RETPFL				√	XMEM	Retype File	new type in X	File re-typed		*Ángel Martin*
RSTCHK				√	XMEM	Reset CHKSUM Byte	FileName in Alpha	Resets Checksum Byte	*Program Files Only*	*Håkan Thörngren*
SAVEBUF				√	XMEM	Save Buffer in XM	id# in X, File Name in Alpha	Stores Buffer in X-MEM	*Includes the Header*	*Håkan Thörngren*
SAVEKA				√	XMEM	Save Keys in XM	File Name in Alpha	Key Assignments saved in XM		*Håkan Thörngren*
WRTDF			√		HPIL	Write XM File to HPIL Drive Data File	FileName in Alpha	Writes XMFile to HPIL	*HPIL Drive File must Exist!*	*R. del Tondo*
WRTPG	√	√			HPIL	Write Page# to HPIL Drive File	FileName in Alpha, pg# in X	Writes page in prompt to HPIL		*R. del Tondo*
WRTXM				√	HPIL	Write XM Contents to HPIL Drive	FileName in Alpha	Writes all XMEM to HPIL		*Ken Emery*
XXEQ				√	XMEM	Extended XEQ	FileName in Alpha	Executes Program	*Program Files Only*	*Ross Wentworth*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
?MMU	√				CLFNS	Is MMU active?	none	Yes/No, skips if false	Places 1/0 in X	*Monte Dalrymple*
ΣCL	√				CLFNS	CL Functions Main Launcher	Prompts "B:M:P:T:U"	Launches selected Launcher	BA to cancel out	*Ángel Martin*
ADRID	√				CLFNS	Address of ROM ID	Flash addr in Alpha	ROM ID in Alpha	Address as returned by YPOK	*Ángel Martin*
BAUD	√				CLFNS	BAUD functions launcher	Prompts "1:2:4:9"	Launches selected function	BA to cancel out	*Ángel Martin*
CLLIB	√				CLFNS	CL Library	Prompts "A-Z"	Starts listing at selected letter	hotkey: R/S, SST, SHIFT, ENTE	*Ángel Martin*
HEPINI _ / _	√				CLFNS	Hepax Initialization	Prompts for values	Iinitializes HEPAX File System	assigns same ROM ID# as pag	*Ángel Martin*
HEPX _	√				CLFNS	HEPAX Functions Launcher	Prompts "4:8:6:I:D"	Launches selected function	BA to cancel out	*Ángel Martin*
HPINYX	√				CLFNS	Hepax INI by X,Y	# pages in Y, first page in X	Iinitializes HEPAX File System	assigns same ROM ID# as pag	*Howard Owen*
HPX4	√				CLFNS	Configure 4k Hepax	none	RAM in Page F, ROM in pg. E		*Ángel Martin*
HPX8	√				CLFNS	Configure 8k Hepax	none	RAM in Pages E-F, ROM in pg. D		*Ángel Martin*
HPX16	√				CLFNS	Configure 16k Hepax	none	RAM in Pages C-F, ROM in pg. B		*Ángel Martin*
HPXX	√				CLFNS	Configure Hepax by X	flags selection (see manual)			*Ángel Martin*
MMU	√				CLFNS	MMU Functions Launcher	Prompts "C:D:E:?"	Launches selected function	BA to cancel out	*Ángel Martin*
MMUCAT	√				CLFNS	MMU Catalog	none	Sequential list of MMU Entries	hotkey: R/S, SST, SHIFT, ENTE	*Ángel Martin*
PPG#4	√				CLFNS	Plug Page #4	prompts menu choices	configures selected choice		*Ángel Martin*
PLUG	√				CLFNS	PLUG functions Launcher	ROM id# in Alpha	Plugs ROM in port / page		*Ángel Martin*
PLUGG _	√				CLFNS	Plug Page	ROM id# in Alpha	Plugs ROM in page#		*Ángel Martin*
PLUGG?	√				CLFNS	Plugged Info	prompts for page#	MMU entry for page		*Ángel Martin*
PLUGGX	√				CLFNS	Plug Page by X	page# in X	Plugs ROM in page	4k ROMS only	*Ángel Martin*
ROMLIB	√				CLFNS	ROM Library	none	Sequential list of ROM ID's	like CLLIB, starts with "A". Pr	*Ángel Martin*
SECURE	√				CLFNS	Activates Password Security	none	Sets SECURE mode ON	Password asked upon startup	*Nick Harmer*
TURBO	√				CLFNS	TURBO settings launcher	Prompts "X:2:5:1:0:,:?"	Launches selected function	BA to cancel out	*Ángel Martin*
UNLOCK	√				CLFNS	Deactivates Password security	asks for password	Sets Secure mode OFF	Calculator is switched off	*Ángel Martin*
UPPG4	√				CLFNS	Unplug Page #4	none	MMU entry cleared	Alters ALPHA	*Ángel Martin*
UPLGG _	√				CLFNS	Unplug Page	prompts for page# 1-14	MMU entry for page is cleared		*Ángel Martin*
UPLGGX	√				CLFNS	Unplug Page by X	page# in X	MMU entry for page is cleared		*Ángel Martin*
UPLUG _	√				CLFNS	UPLUG functions launcher	Prompts for location	Location is removed from MMU	Valid chrs: 1-4, L, U, BA	*Ángel Martin*
UPLUGA	√				CLFNS	Unplug by ALPHA	ROM rev string in Alpha	ROM unplugged if present	*shows NO MATCH if not*	*Ángel Martin*
XCAT _	√				CLFNS	Extended Catalog	Prompts: "B:F:H:L:M:R"	Launches selected function	BA to cancel out	*Ángel Martin*
XPASS	√				CLFNS	Change Password	asks old/new passwords	Password is changed	Security mode not changed	*Nick Harmer*
YCL>	√				CLFNS	Clear Alpha from ">"	string in ALPHA	Clears from ">" char to the right	Nothing if ">" is not present	*Ángel Martin*
YFNZ?	√				CLFNS	Location of YFNZ ROM	none	Location in MMU	Does stack lift.	*Monte Dalrymple*
YINPT _	√				CLFNS	Input String	prompts for characters	HEX entry plus control chrs	use "M" for "RAM, "K" for "1	*Ángel Martin*
YRALL	√				CLFNS	Flash Read-All	none	Reads Calculator/MMU from Flash	OK / OKALL in ALPHA	*Ángel Martin*
YSWAP>	√				CLFNS	Swap Alpha around ">"	string in ALPHA	Alpha swapped around ">"	Nothing if ">" is not present	*Ángel Martin*
YSWAP-	√				CLFNS	Swap Alpha around "-"	string in ALPHA	Alpha swapped around "-"	Nothing if "-" is not present	*Ángel Martin*
YWALL	√				CLFNS	Flash Write-All	none	Writes Calculator/MMU to Flash	*"OK" / "OKALL" in ALPHA*	*Ángel Martin*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ARCLBF _ _				√	KA/BUF	Buffer to Alpha	buf# in prompt	ALPHA recalled from Buffer	*24 chrs max*	*Ángel Martin*
ASG	√	√			KA/BUF	Multi-byte Assign Function	FCN code and Key code	Function assigned to key	*Supports IND (I)*	*Frits Ferwerda*
ASTOBF _ _				√	KA/BUF	Alpha to Buffer	buf# in prompt	ALPHA saved in Buffer		*Ángel Martin*
B?		√			KA/BUF	Buffer Existence	buf# in prompt	Yes/No Buffer test		*W&W GmbH*
BF>RGX _ _				√	KA/BUF	Buffer content to Registers	bbb,eee in X, buf# in prompt	Buffer content stored in Main Mem	*Header NOT included*	*Ángel Martin*
BF>ST _ _				√	KA/BUF	Buffer Content to Stack	buf# in prompt	Buffer content stored in Stack	*Header NOT included*	*Ángel Martin*
BFCAT	√			√	KA/BUF	Buffer Catalog	none	Enumerates all present buffers		*Ángel Martin*
BFHEAD				√	KA/BUF	Buffer Header	buf# in prompt	Decodes buffer register		*Ángel Martin*
BFLNG				√	KA/BUF	Buffer Length (Size)	buf# in prompt	Buffer Length in X		*Ángel Martin*
BFRCL _ _				√	KA/BUF	Restore Buffer from RG	id# in X; RCL in prompt	Restores buffer from Main Mem	*Includes the Header*	*Ángel Martin*
BFSTO _ _				√	KA/BUF	Store Buffer to RG	id# in X; RCL in prompt	Stores Buffer in Main Mem	*Includes the Header*	*Ángel Martin*
BFVIEW				√	KA/BUF	Buffer View	buf# in prompt	Views all Buffer registers		*Ángel Martin*
BLIST				√	KA/BUF	Buffer List	none	Shows list of present buffers		*David Yerka*
BUF?				√	KA/BUF	Buffer Existence	buf# in prompt	Yes/No Buffer test		*Ángel Martin*
BUFHD				√	KA/BUF	Buffer Header Location	buf# in prompt	Address of header reg (bottom)		*Ángel Martin*
CLB		√			KA/BUF	Delete Buffer	buf# in prompt	Deletes buffer		*W&W GmbH*
CLBUF				√	KA/BUF	Clear Buffer Contents	buf# in prompt	Buffer contents cleared		*Ángel Martin*
CRBUF				√	KA/BUF	Create Buffer	ID,00(SZ) in X	Creates Buffer		*Ángel Martin*
DELBUF				√	KA/BUF	Delete Buffer	buf# in prompt	Buffer Deleted		*Ángel Martin*
KACLR				√	KA/BUF	Clear KA/Buffer area	"ok"/"okall" in Alpha	Deletes KA / all IO area		*Hajo David*
KALNG				√	KA/BUF	KA area Length	none	Number of KARs used		*Hajo David*
KAPCK				√	KA/BUF	Pack KA registers	none	KA Registers packed		*Hajo David*
KYOFF				√	KA/BUF	Deactivate key assigment	Key pressed at prompt	Assignement deactivated	*from MLROM module*	*Frits Ferwerda*
LKAOFF				√	KA/BUF	Local KA disabled	none	Disables top 2 rows KA		*Ross Cooling*
LKAON				√	KA/BUF	Local KA Enabled	none	Re-enables all disabled KA		*Ross Cooling*
REIDBF				√	KA/BUF	Re-issue buffer id#	old,new id#'s in X	Buffer new id# is issued		*Ángel Martin*
RGX>BF _ _				√	KA/BUF	Registers to buffer Content	RG range in X, buf# in prompt	Fills Buffer with RG data	*Header NOT included*	*Ángel Martin*
RESZBF				√	KA/BUF	Re-size buffer	ID,00(SZ) in X	buffer has new size	*Increasing size loses content*	*Ángel Martin*
ST>BF _ _				√	KA/BUF	Stack to Buffer Content	buf# in prompt	Fills Buffer with Stack data	*Header NOT included*	*Ángel Martin*
VKEYS				√	KA/BUF	View Keys	none	Catalogs all assigned keys	*from PANAME module*	*PANAME*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ΣDGT		√			UTILS	Mantissa Digit Sum	number in X	Sum of mantissa digits in X		*Ángel Martin*
ABSP	√	√			UTILS	Alpha Back Space	none	Deletes rightmost chr in Alpha		*W&W GmbH*
ANUMDL	√	√			UTILS	ANUM with deletion	none	Gets number from Alpha to X	*Deletes it*	*HP Co.*
APPEND			√		UTILS	Append function	none	Turns Alpha ON ready to append	*from BLDROM module*	*Doug Wilder*
ARCLH		√			UTILS	ARCL Hex	Decimal number in X	Appends HEX number to Alpha		*W&W GmbH*
ARCLI	√	√			UTILS	ARCL Integer	Decimal number in X	Appends Integer part to Alpha		*W&W GmbH*
ASWAP		√			UTILS	Alpha Swap	"A,B" string in alpha	Swaps string to "B,A"		*Ángel Martin*
BLCAT	√		√		UTILS	Blck Catalog	none	Enumerates all Blocks (pages)		*VM Electronics*
BST^			√		UTILS	Auto BST	Pointer in PRGM	Continuous BST while pressed		*Nelson Crowle*
CAS		√			UTILS	Clear Auto-start	none	Auto-start flag cleared		*W&W GmbH*
CFX _ _	√		√		UTILS	Clear Flag	Flag in prompt	Flag cleared	*All 55 flags valid*	*Michael Katz*
CHKCFG	√		√		UTILS	Check Configuration	none	Checks for XROM id# conflicts		*Ángel Martin*
CLA-	√	√			UTILS	Clears Alpha from "-" chr	Text in Alpha	Deleted from "-"		*W&W GmbH*
CPRV			√		UTILS	Clear Private Status	Pointer in PRGM	Private removed		*Nelson Crowle*
CRTN?			√		UTILS	Curtain Location Finder	none	Curtain location to X		*Ken Emery*
CSST			√		UTILS	Continuous SST	Pointer in PRGM	Shows all PRGM lines		*Phil Trinh*
CVIEW			√		UTILS	Continuous View	Text in Alpha	Prints or Views Text		*Ken Emery*
DREG?			√		UTILS	Data Registers finder (SIZE)	none	SIZE to X		*Ken Emery*
DSP?			√		UTILS	Display Digits Finder	none	# of decimal digits to X		*Ángel Martin*
DTOA	√	√			UTILS	Display to Alpha	Text in Display	Text in Alpha		*Ángel Martin*
DTST	√	√			UTILS	Display Test	none	Shows Display Test		*Chris Dennis*
F/E		√			UTILS	Finx/Eng combined	none	sets both modes		*W&W GmbH*
FC?S _ _			√		UTILS	FC? And Set	flag# in prompt	Tests if Clear and sets it		*Ken Emery*
FNC?			√		UTILS	Function Data	XX,NN data in X	FNAME in Alpha		*W&W GmbH*
FS?S _ _			√		UTILS	FS? and Set	flag# in prompt	Tests if Set and sets it		*Ken Emery*
FREG?			√		UTILS	Free Registers finder	none	Available RAM registers		*Ken Emery*
GTADR		√			UTILS	GOTO Address	Address in prompt	Executes FNC at given location		*W&W GmbH*
GTEND			√		UTILS	GOTO .END.	none	Sets PRGM pointer to .END.		*Ken Emery*
LASTP			√		UTILS	Last Program	none	Sets PRGM pointer to .END.		*Zengrange*
MNFR _ _ _		√			UTILS	Mainframe Call	id# in prompt	executes function (see table)		*Clifford Stern*
MSGE		√			UTILS	Partial Message	addres in X	Message Text		*R. del Tondo*
NOP			√		UTILS	No Operation	none	Adds NOP line in PRGM		*Zengrange*
PLNG			√		UTILS	Program Length	Prompts for Name	Shows Program Length		*W&W GmbH*
PMTA _		√			UTILS	Prompt by Alpha	text in prompt	inputs text in Alpha		*W&W GmbH*
PMTH _ _		√			UTILS	Prompt HEX	Hex data in prompt	Decimal in X		*W&W GmbH*
PMTK _		√			UTILS	Prompt Key	Key range in Alpha	Prompts for Key		*W&W GmbH*
PRD		√			UTILS	Print Display	Text in Display	Text printed out	*from AECROM module*	*Nelson Crowle*
PTCAT _			√		UTILS	Port Catalog	Port# in prompt	Catalogs from that port on		*Clifford Stern*
RCLF			√		UTILS	RCL Flags	none	Puts NNN with all flags status in X		*Hajo David*
RNDM		√			UTILS	Random Number from seed	Seed in buffer	random number in X		*W&W GmbH*

Function	CL_Utils	-AMC"OS/X	-TOOLBOX	Rampage	Class	Description	Input	Output	*Comments*	Author
ROMCAT _ _	√		√		UTILS	ROM Catalog	XROM id# in prompt	ROM Functions listed		*J.D. Dodin*
ROMLST	√		√		UTILS	ROM List	none	Shows all on-line ROMS		*Ángel Martin*
RTN?		√			UTILS	Is RTN pending?	none	Yes/No, skips if false	*from BLDROM module*	*Doug Wilder*
SAS		√			UTILS	Set Auto-Start	none	Sets Auto-Start flag		*W&W GmbH*
SEED		√			UTILS	Stores Seed in Buffer	Seed in X	Stored in Buffer		*W&W GmbH*
SFX _ _	√		√		UTILS	Sets Flag	flag# in prompt	Flag Set	*All 55 flags valid*	*Michael Katz*
SPEED	√	√			UTILS	CPU Speed (usc/cycle)	none	result in X	*in micro-s per cycle*	*Doug Wilder*
SPLASH	√		√	√	UTILS	page#4 splash screen	none	shows splash test	*Easter Egg*	*Nelson Crowle*
SPRV			√		UTILS	Set Private	Pointer in PRGM	Sets Private Status		*Nelson Crowle*
SST^			√		UTILS	Auto SST	Pointer in PRGM	Continuous SST while pressed		*Nelson Crowle*
STOF			√		UTILS	STO Flags	NNN in X	Mass-update of all flags		*Hajo David*
TF _ _		√			UTILS	Toggle Flag	flag# in prompt	Status Toggled (any flag)	*in X for PRGM*	*Ken Emery*
TGFX _ _			√		UTILS	Toggle Flag	flag# in prompt	Status Toggled	*All 55 flags valid*	*Zengrange*
TGLC		√			UTILS	Toggle LowerCase	none	Lower Case Mode toggled	*for Alpha Input*	*W&W GmbH*
VIEWA		√			UTILS	View Alpha	text in Alpha	non-stopped View		*Ken Emery*
VIEWH		√			UTILS	View HEX	number in X	Shows HEX value in Display		*W&W GmbH*
WSIZE _ _		√			UTILS	Word Size	Size in prompt	Sets the word Size	*in X for PRGM*	*W&W GmbH*
XQ>GO			√		UTILS	Pop Address	none	Pops one RTN address		*Håkan Thörngren*
XROM _ _/_ _	√		√		UTILS	ROM Function Excute	XX,NN data in prompt	Executes function		*Clifford Stern*
XTOAH		√			UTILS	X to Alpha as HEX	number in X	appends char# to Alpha		*W&W GmbH*
Y/N?	√				UTILS	Yes/No Answer	pressed key		*from PANAME module*	PANAME

#	Name	Address	Description	Inputs	Output	Flags	Uses	RAM	RTN Levels	Calls
1	ABSP4	44B9	Deletes Alpha leftmost chr	text in Alpha	deleted chr	n/a	Alpha	Chip0	1	n/a
2	AINT4	4293	Append Integer to Alpha	value in X	int(x) appended	n/a	A,B, C,N,14(d)	Chip0	2	CHK$S, INT, AFORMT
3	AINT8	429A	Append	value in B	value appended	n/a	A,B, C,N,14(d)	Chip1	2	AFORMT
4	ALIST	44CF	Input List in Alpha	none	list in Alpha	??	everything!	chip0	3	NEXT1, ENCP00, APNDNW, ENLCD, BLINK1, CLRCLD, RSTSQ, NFRPU, MSGA
5	ANUMDL	4200	ANUM w/ Deletion	text in Alpha	num in X, deleted	2,3,9	everything!	chip0	3	FAHED, STBT10, GTBYTA, DIGENT, PTBYTA, INCADA, R^SUB, NOREG9, NFRPR
6	APERX4	400C	Display Msg and Halt RPN	msg prepared	shows custom msg	8	n/a	n/a	2	LEFTJ, MSG105
7	APNDBK	4025	Append Blank to Alpha	none	space appended	n/a	n/a	n/a	2	APND10
8	ASG	475C	ASG Function Header	ptompts for data	key asigned		everything!		4	ENCP00, NULTST, OFSHFT, XASN, BCDBIN, ERRNE, MASK, XASNRC,
9	ASG4	475D	ASG entry point	ptompts for data	key asigned		everything!		4	ENCP00, NULTST, OFSHFT, XASN, BCDBIN, ERRNE, MASK, XASNRC,
10	ASG#1	46B7	subroutine for ASG						1	n/a
11	ASG#2	46D1	subroutine for ASG						1	n/a
12	ASN4	467C	2-byte assignment routine	keycode in A(1:0); fnc in B(3:0)		1	A,B,C,8(P), 10(+)		4	ENCP00, TBITMA, GCPKC, SRBMAP, XASN05
13	BST+	41C1	Continuous BST	PRGM mode	BST actions	0,1, 6, 9	14(d), 15(e)		3	SSTBST, GETPC, NXLSST, GETLIN, PUTPC, DFRST8, ENCP00
14	CCDC4	469D	CCD subroutine	??	??	0	A,C,M,9(Q)	Chip0	3	ASRCH, ERRNE, ERROR, GETPC, FLINK, CPGM10
15	CDE4	4480	Code NNN	??	??	9	A,C,M,N	Chip0	3	n/a
16	CHECK0	41C8	subroutine for SST+			0, 13	C, 14(d)	Disp	2	ENCP00
17	CHKBFX	43A0	Check buffer w/ id# in X	buf id# in X(3)	buf adr in A[S&X]			buffer	2	BCDBIN, ERRDE
18	CHKBUF	43A9	Check buffer w/ id in C	buf id# in A[S&X]	buf adr in A[S&X]			buffer	1	ERRDE
19	CHKCFG	4143	Check Configuration	none	shows result msg(s)	9	A,B,C,N	Sleeper	1	APRMSG2, MESSL, APEREX, GENNUM
20	CHKEM	43F4	Check for EM presence	none	shows msg if not EM	n/a	A,C	n/a	1	n/a
21	CHKKEYS	4411	Check for KA entries	none	shows msg if none	n/a	A, C, 10(+), 15(e)	Chip0	1	n/a
22	CHKST2	4069	Check X,Y for AlphaData	values in X,Y		n/a	A,C	Chip0	2	CHK#S
23	CHKST3	4066	Check X,Y,Z for AlphaData	values in X,Y,Z		n/a	A,C	Chip0	2	CHK#S, CHK#S1
24	CHKST4	405E	Check X,Y,Z,T for AlphaData	values in X,Y,Z,T	Result in C	n/a	A,C	Chip0	2	CHK#S
25	CLA-	42A7	CLA from "-" chr	none	clears Alpha from chr#	n/a	A,B,C	Chip0	2	CLA
26	CLA>	42AA	CLA from ">" chr	none	clears Alpha from chr#	n/a	A,B,C	Chip0	2	CLA
27	CLAC	42AD	CLA from comma	none	clears Alpha from chr#	n/a	A,B,C	Chip0	2	CLA
28	CLACHR	42AF	CLA from chrcode in C[S&X]	chr# in C[S&X]	clears Alpha from chr#	n/a	A,B,C	Chip0	2	CLA
29	CLBF	4626	Subroutine for PRD	none	printer buffer cleared		A,B,C,	printer?	2	APEREX
30	CLEM4	42DD	Clear all X-Memory	none	all XM cleared	n/a	??	n/a	2	EMDIR
31	DCD4	4497	Decode NNN	??	??	9	A,B,C,8(P), 9(Q)	Chip0	1	n/a
32	DECC	407D	subtracts one from C	value in C		n/a	A,B,C	Chip0	2	CHK#S, SUBONE
33	DECC10	407F	ditto w/out [CHK#S]	value in C		n/a	A,B,C	Chip0	1	SUBONE
34	DIVTWO	4FF3	Divides c by 2	value in C		n/a	A,B,C,M	Chip0	1	DV1-10
35	DIVBY2	4FF6	Divides by 2 checking	value in C		n/a	A,B,C,M	Chip0	1	DV1-11
36	DTOA4	A2E6	Display to Alpha	data in display	data in Alpha		A,B,C	Chip0	3	ENCP00, CLA, ENLCD, APND10, CLRLCD, RSTKB
37	DTOHEX	4181	Enhanced [BCDBIN]	value in X[S&X]	value in C[S&X]		A,B,C	Chip0	2	ERRAD, ERROF, BCDBIN, GOTINT
38	DTOHXC	4182	Enhanced [BCDBIN]	value in C[S&X]			A,B,C	Chip0	2	ERRAD, ERROF, BCDBIN, GOTINT
39	DTST4	430E	Display Test	none	test string in dsp	??	??	Disp	2	CLLCDE, STMSGF
40	DUPBUF	43CF	Buffer error message	none	shows msg	n/a	n/a	Disp	2	APERMSG, APEREX
41	E3/	401B	Divides C by 1,000	value in C	Result in C, and {A,B}	n/a	A,C	Chip0	2	CHK#S,
42	E3/E+	401B	as above plus 1	value in C		n/a	A,B,C	Chip0	3	E3/, ADDONE
43	ENDOBF	43DA	Buffer error message	none	shows msg	n/a	n/a	Disp	2	APERMSG, APEREX
44	EXISTS	4386	Checks for REG existence	RG# in A[S&X]	adr checked	n/a	A,B,C, 13(c)	n/a	3	BCDBIN, ERRNE
45	GETRG#	435E	Gets postfix from next line	value in PRGM	value in A[S&X]	n/a	A,B,C,M		3	GETPC, NXBYT, NBYTAB, DECAD, PUTPCX, GOTINT
46	GETRM4	45D5	Read ROM from IL drive	Fname in Alpha	File read from Tape	n/a	everything!	HP-IL	2	FLSCH0, SEEKRN, SNDATA, UNT, RDDFRM, CSERK, BCDBIN, ERROF
47	HELLO4	4FC5	Welcome message	none	shows msg	n/a	A,C	n/a	2	MSGX
48	INCC	4076	adds one to C	value in C	Result in C, and {A,B}	n/a	A,B,C	Chip0	2	CHK#S, ADDONE
49	INCC10	4078	ditto w/out [CHK#S]	value in C	Result in C, and {A,B}	n/a	A,B,C	Chip0	1	ADDONE
50	ISOK?	4084	Checks if OK	OK/OKALL in alpha		1	A,B,C,M	Chip0	2	ERRDE, CLA
51	MOD2	40F3	Module (C,2)	value in C	Trnsfers execution	n/a	C,A	Chip0	2	CHK#S, INTFRC, MOD10

52	MSG4	49CE	Mainframe Message	value in X	shows msg (full/partial)	8	C,A	n/a	3	BCDBIN, MSGE, MSGX
53	NOBFFS	43EA	Buffer error message	none	prepares msg	n/a	A,C	n/a	3	APERMSG2, MESSL
54	NOBUFR	43D5	Buffer error message	none	prepares msg	n/a	A,C	n/a	3	APERMSG2, MESSL
55	NOCX	4436	No CX-OS error msg	none	prepares msg	n/a	A,C	n/a	3	CLLCLD, APERMSG2, MESSL
56	NOHPIL	4426	Checks for HPIL existence	none	prepares msg	n/a	A,C	n/a	1	APERMSG2, MESSL, APEREX
57	NOIL10	442F	No HP-IL error msg	none	prepares msg	n/a	A,C	n/a	2	APERMSG2, MESSL, APEREX
58	NOKEYS	441F	No KA entries messge	none	prepares msg	n/a	A,C	n/a	3	APERMSG2, MESSL, APEREX
59	NOMSG	43C8	"NO" prefix routine	msg text below	prepares msg	n/a	A,C	n/a	2	APERMSG2, MESSL, APEREX
60	NOPRT	461B	No Printer message	NONE	prepares msg	n/a	A,C	n/a	2	APERMSG2, MESSL, APEREX
61	NOPS4	4920	Locates NOPs in ROM	"BEG:END" NNN in x					4	SKP, APND10, BIN-D, XAVIEW, NFRPU,
62	NOROM	4126	No ROM message	none	prepares msg	n/a	A,C	n/a	1	APERMSG2, MESSL,
63	NOTOK	4408	Not OK error message	none	prepares msg	n/a	A,C	n/a	2	APERMSG, APEREX
64	NOXMEM	4400	NO XMEM msg	none	prepares msg	n/a	A,C	n/a	2	APERMSG2, MESSL, APEREX
65	NSWAP	4272	Swap Alpha around chr.	chr. In N	swaps texts	n/a	A,C,N, X(3),L(4)	Chip0	4	BIN--D, AROT, ABSP, POSA4, NFRX
66	ONEPMT	4113	Hex digit PROMPT	inputs for chr	appends it to prompt	3,4	A,C, 15(e)	Disp	2	BLINK, SKIP1,
67	PMASK0	4016	Mimics prompt from X	none	prompts if not PRGM	4, 13	A,C	n/a	2	BCDBIN
68	PMASK4	4012	Prompt Mask routine	VALUE IN x					2	BCDBIN
69	PMTFNM	428E	Prompt FNC Name	FNC adr in C[ADR]	prompts FNAME	n/a	C,M	n/a		CLLCDE, PROMF2
70	POSA4	40A7	Position in Alpha	char value in X	location in X(3)	1,2,	everything!	Chip0	3	X<256, FAHED, GTBYTA, INCADA, CNTBYT, BIN-D,
71	PRD4	4644	Print Display	Data in display	Data printed	5,6,7	everything!		3	ENLCD, ENCP00, PBYTCX, OUTPCT, APEREX
72	RAND4	4465	Random number math	seed in C	RND# in C	n/a	A,B,C	Chip0	2	MP2-10, AD1-10, INTFRC
73	RCL4	4337	Modified RCL w/o [NFRPU]	value in B	value in X, stack lifted	n/a	A,B,C,3(X)	Chip0	2	R^SUB
74	READNM	456D	Read FName from M	Text in M	Text in Q		A,B,C,M,Q	Chip0	1	n/a
75	RGSORT4	4887	Register SORT	reg range in ??	Registers Sorted		A,B,C,M	RAM	2	GTINDX
76	RGTHJ2	46F6	Right-Justify Display	display enabled	right-justified	n/a	A,B,C	Disp	1	n/a
77	RIGHTJ	46F4	Right-Justify Display	text in Disp	right-justified	n/a	A,B,C	Disp	2	ENLCD
78	RMLST4	49E1	ROMLIST code	none	list in Alpha	n/a	everything!	n/a	4	GENNUM, ENCP00, APNDDG+, APND10, XAVIEW
79	ROMCAT	433F	ROM Catalog	ROM id# in A[S&X]	Functions catalog		A,B,C	n/a	1	ERRNE, PTCNTR
80	SAVRM4	4584	Write ROM to IL Drive					Chip0	3	FLSCH, DUPFL, CRF14, SEEKN, SCMD, DDL0, PLERK, SDATAD, SCMD, UNL
81	SDGT4	432C	Sum of mantissa digits	number in X	result in X(3)	n/a	A,C	Chip0	1	n/a
82	SPEED	4994	CPU Cycles	none	result in X(3)		A,B,C	Chip0, Timer	2	ENTMR, BIN-D, DV2-10
83	SPLASH	4717	shows splash message	none	does the show			Disp	2	CLLCDE, ENCP00, STMSGF
84	SST+	41A4	Continuos SST	PRGM mode	SST actions	0,1, 6, 9	14(d), 15(e)		3	SSTBST, GETPC, NXLSST, GETLIN, PUTPC, DFRST8, ENCP00
85	ST<>A4	42C3	Stack / Alpha Swap	none	swaps registers	n/a	A,B,C	n/a	1	n/a
86	SURE?	496E	OK Y/N? Entry	prompts Y/N	1/0 in A[MS]	n/a	A,C	n/a	3	APRMSG2, MESSL, LEFTJ, BLINK1, ENCP00
87	SVGT4	460D	Checks X[S&X]<="F"	chr# in X		n/a	A,C	n/a	2	BCDBIN, ERROF
88	SVGTA	4611	Checks A[S&X]<="F"	chr# in A[S&X]		n/a	A,C	n/a	1	ERROF
89	TITLE0	470B	Library4 header	none	does the show			DSP	2	CLLCDE, ENCP00, STMSGF
90	TOGF4	444F	Toggle Flag	flag# in A[S&X]	sets/clear flag	n/a	A,C,14(d)		2	ERRNE, XCF, XSF
91	UCRUN	406E	User Code RUN	distance in C[S&X]	Trnsfers execution	n/a	A,C		2	XRM20
92	UCRUN2	4359	User Code RUN	addres in A(4:0)	Result in C, and {A,B}	n/a	A,C		2	XRM20
93	VALID	4395	Address Existence check	RG# in A[S&X]	checks for existence	n/a	A,C, 13(c)	Chip0	1	ERRNE
94	XAVEW?	44B5	XAVIEW if not in PRGM						2	XAVIEW
95	Y/N?	41D1	Yes/No answer	prompts for key	skips line if "N"	9	A,B,C, 14(d)		3	RSTKB, RSTSEQ, CLLCDE, RSTMS0, SKP, NOSKP, OFF
96	YESNO	4973	Yes/No entry			n/a			3	MESSL, LEFTJ, BLINK1, ENCP00
97	ZOUT	4029	Display Complex#	values in Y,X	data in Alpha	n/a	A,B,C,	Chip0	3	CLA, APN10, AFORMT, APND-, LDSSTO, XAVIEW, NFRPU
98	WAKEUP	4FB9	Wakeup splash	none	shows splash	n/a	C	n/a	3	CLLCDE, MESSL, ERR110
99	TURBO0	4FDF	Disables TURBO mode	none	clears turbo	n/a	C	n/a	1	n/a
100	TURB50	4FE3	Sets Turbo 50 mode	none	sets turbo50	n/a	C	n/a	1	n/a
101	WAIT4	4FE8	Waits a little	none	none	n/a	C	n/a	1	n/a