Making the 41CL User-friendly

Monte J. Dalrymple

The 41CL is an upgrade to the venerable HP-41, created by replacing the CPU board in a 41C/CV/CX fullnut with the PC board shown in Figure 1. This paper will not discuss the hardware [(1)] but instead will concentrate on the software required to take advantage of the new features of the design.

The NEWT (Nut, Expanded With Turbo) microprocessor [(2)] used in the 41CL provides a number of new features surrounding a clone of the CPU originally used in the 41C series. These new features, particularly the Memory Management Unit (MMU), are probably more complex than the typical 41C user is used to dealing with.

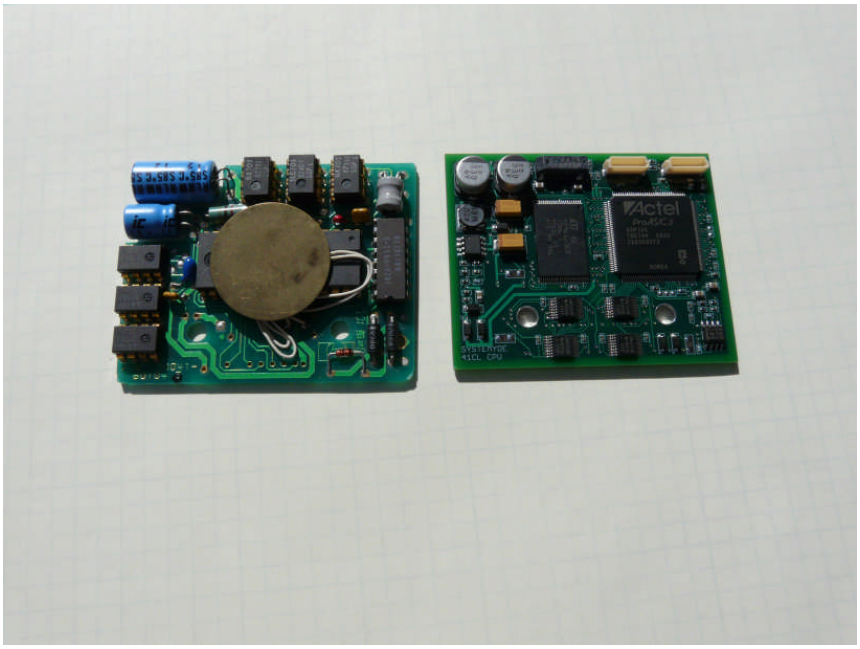

Figure 1 - Old and new CPU board

The 41CL attempts to hide this complexity from the typical user, while still providing full control over the hardware to the advanced user. This is accomplished by including a special set of functions, much like the Extended Functions present in the 41CX.

These functions, which I call Y-Functions and which Catalog as YFNS -1A, aim to provide an intuitive way for the user to manage the module images present in the Flash memory on the new CPU board as well as control the new Turbo operating speed modes. In addition, for the advanced user, there are functions to PEEK and POKE memory, move entire 4k blocks of memory, access the on-chip peripheral registers, and do PUT and GET operations with the on-board serial port.

Probably the most significant new feature provided by the 41CL is the wide range of module images available in the on-board Flash memory. These images can be "plugged into a Port" just as a physical module would be plugged into a Port. First, the four-character module identifier is entered into the Alpha register and the appropriate PLUG command is executed from the keyboard. Twelve PLUG commands are available: PLUG1, PLUG1L and PLUG1U control Port 1 on the calculator, with corresponding commands for the other three ports.

The PLUG commands parse the module identifier and verify that the selection is valid for the Port selected. Note that only the first and last characters of the module identifier are parsed by the software to increase performance. Just like the original physical modules, some module images can only be used in particular ports, and if an invalid configuration is attempted a DATA ERROR message will result. Once the selection is validated, the software then automatically programs the MMU so that accesses for addresses in that Port will be redirected to the module image in the Flash memory. At this point a Catalog

will verify that the module is "plugged into the Port" and available for use. No power-down of the calculator is required. While a module image is plugged into a port the physical port is not accessed by the CPU, so physical modules may be present in the Port without any electrical conflict.

In addition to the module images in Flash memory, blocks of RAM memory may also be plugged into a Port. This allows the on-board RAM memory to be used as HEPAX memory or MLDL memory, for example. An experienced user may want to copy a module image to RAM, modify it, and then plug this modified image into a Port. I had to do this while debugging the design, as I had some errors in several of the new functions that I had to patch before they could be used.

Unplugging a module image from a Port is just as simple. There are twelve UNPLUG commands available: UPLUG1, UPLUG1L and UPLUG1U control Port 1, with corresponding commands for the other three ports. Unlike the PLUG commands, no checking for configuration is performed. The software merely invalidates the appropriate MMU entries so that the CPU resumes access of the physical Port. The user is responsible for not unplugging half of a module image! Table 1 shows the module images available and the corresponding module identifiers.

Module Image	Mnemonic	Module Image	Mnemonic	Module Image	Mnemonic
Advantage Pac 1B	A41P	ESMLDL 7B	ESML	Plotter Pac 1A	PLOT
ADVENTURE #1	ADV1	Extended I/O 1A	EXIO	PPC Module	PPCM
ADVENTURE #2	ADV2	Financial Pac	FINA	Real Estate Pac 1A	REAL
AECROM	AECR	FUNSTUFF ROM	FUNS	SANDMATH-7	SANA
Autofinance	AFIN	Games Pac 1A	GAME	Securities Pac 1A	SECY
ALPHA ROM	ALPH	GMAC 2	GMAS	SGS GAS	SGSG
Assembler 3	ASMB	GMAC 3	GMAT	SIM Module	SIMM
ASTRO-2010	ASTT	HEPAX 1D	HEPX	SKWID ROM	SKWD
HP Autostart	AUTO	Home Management	HOME	Simplex Module	SMPL
Aviation Pac 1A	AVIA	JMB-MATH	JMAT	Statistics Pac 1B	STAT
Boeing-B52	B52B	JMB-MATRIX	JMTX	Stress Analysis	STRE
Buffer ROM	BUFR	LABELS ROM	LBLS	Structural Analysis	STRU
CCD Module 1B	CCDR	LANDNAV	LAND	SUP-R-ROM	SUPR
Chemistry User	CHEM	Math Pac 1D	MATH	Survey Pac 1B	SURV
CHESS ROM	CHES	Machine Design Pac	MCHN	Thermal Pac 1A	THER
Circuit Analysis Pac	CIRC	Melbourne ROM	MELB	Toolbox ROM	TOOL
Clinical Lab Pac	CLIN	Mil Engineering	MILE	TREKKIES ROM	TREK
CurveFit ROM	CURV	MLROM	MLRM	UNITCONV ROM	UNIT
Data Acquisition	DACQ	Navigation Pac 1B	NAVI	USPS Module	USPS
David Assembler	DAVA	NFCROM	NFCR	alternate YFNS	YFNS
HP-IL Development	DEVI	NAVCOM 2	NVCM	default YFNS	YFNZ
HP- IL Diagnostic	DIIL	P3B/C Aviation Pac	P3BC	HP41Z ROM	Z41Z
Diamond ROM	DMND	PCODER 1A	PCOD		
ES41 Module	E41S	Petroleum Pac 1A	PETR		

Table 1 - Module images and mnemonics

Finding a place for these new functions in the address space of the CPU was a challenge, because I wanted to do it in the most compatible fashion. At power-up the MMU is disabled and all MMU entries are cleared. Rather than reserve a specific Port for the Y-Functions, while the MMU is disabled they are mapped to Page 7 of the CPU address space. This is the page reserved for HP-IL functions, so HP-IL will not be available to the user until the MMU is enabled.

Before enabling the MMU however, the user must PLUG the Y-Functions into a Port. Then the MMUEN command must be executed. At this point the 41CL is ready to go. If you enable the MMU before plugging the Y-Functions into a Port you have a plain 41CV/CX with none of the new features available.

There are three other MMU commands available besides MMUEN. The MMUDIS command disables the MMU and restores the 41CL to the power-on condition (as far as the memory is concerned). MMUCLR clears all MMU entries, and so should only be executed while the MMU is disabled. Otherwise the Y-Functions will no longer be available. Finally, the MMU? command tests the status of the MMU, returning 0 in the X register if the MMU is disabled and 1 in the X register if the MMU is enabled.

Control of the various Turbo modes is similar. The TURBO2, TURBO5, TURBO10, TURBO20 and TURBO50 commands select the corresponding Turbo clock speed. The TURBOX command disables the Turbo mode and returns to 1x operation. The TURBO? command returns the current value of the Turbo mode in the X register. All of the module images, as well as the majority of the operating system software run at the selected speed. However, there are several timing loops in the operating system that will always automatically run at 1x.

The commands above will probably be sufficient for the vast majority of users, and that's why I made them as simple as possible to use. But advanced (or adventurous) users can take advantage of the remainder of the Y-Functions and the features of the NEWT microprocessor itself.

Most of these advanced commands require one or more memory addresses, and in designing the command operation I chose to use the Alpha register for the user to input this information (because it is hexadecimal). The command is then executed, and if data is returned the value in the Alpha register is modified appropriately. Some commands also return a status in the X register (0 for success and 1 for failure). Most of the commands allow both a logical address, which is the 16-bit HP-41 address plus the bank select information, or a physical address, which is the native NEWT 24-bit address. See Figure 2 for the requirements for the various commands.

Command	Address format
YMCLR	PPPPPP-DDDD or LLLL-B-DDDD
YMCPY	PPP>PPP or PPP>L-B or L-B>PPP or L-B>L-B
YPEEK & YPOKE	PPPPPP-DDDD or LLLL-B-DDDD or R---DDDD
YFERASE	PPPPPP
YFWR	PPP-PPP

Table 2 - Command address format

Four basic commands for dealing with memory are available. The YMCLR command initializes a 4k block of memory to a specified value, while the YMCPY command copies from one 4k block of memory to another. As I mentioned previously, this command is handy for moving a module image to RAM for editing. This copy command accepts either logical (what the CPU uses) or physical memory addresses for both source or destination. These commands do not check for an attempt to write to Flash memory though, so be careful.

The YPEEK and YPOKE commands read or write a specific word to or from memory (remember that the NEWT memory is 16-bits wide) or an on-chip peripheral port. The YPOKE command must be used with care, however, because the entire address space of the NEWT microprocessor is available. This means that a user can write directly to a word in the X-, Y-, Z- or T register, or to the variables used by the operating system or directly to an MMU register.

The last two memory commands deal with the Flash memory, and I hesitated including them because they require even more care by the user. Because they operate on the Flash memory, the code must be relocated to RAM before the commands can be used. Both commands check to make sure that they are running from RAM before executing, and will return an error message without executing if they are located in the Flash memory. Both commands will also return an error message without executing if the user attempts to erase or program the area of Flash that holds the operating system. Allowing either operation on the operating system area would turn the 41CL into a brick.

The YFERASE command erases a 32kx16 block of Flash to the default state of 0xFFFF for every word. The YFWR copies a 4kx16 block of RAM to a 4kx16 block of Flash. An attempt to copy from Flash will return an error message without executing, because the only operation allowed during a Flash write is the writes themselves. Both of these Flash-related commands actually control the Turbo mode during execution. The YFERASE command executes in 1x mode because it has to wait 1.6s for the erase operation to conclude before returning to the operating system. The YFWR command executes in 50x mode for maximum speed. Both commands return to the previous Turbo mode upon completion.

The final commands, which I am still working on, deal with the on-board RS-232 port. Since there is no existing way for the serial signals to exit the calculator these functions may be rarely used, but who knows. Currently I have commands to set the baud rate: BAUD96, BAUD48, BAUD24 and BAUD12 select 9600, 4800, 2400 and 1200 baud respectively. There will also be PUT and GET commands to transfer individual bytes. I also plan to include block transfer commands, and am open to suggestions on how they should work.

Although it has taken forever, the 41CL is almost ready for prime time. In retrospect, I underestimated the effort required to both design the PC board and write the software to control the new features on the NEWT microprocessor. But I think that the result has been worth the effort (and wait).

References

(1) Dalrymple, Monte. "Calculator Brain Transplant." Circuit Cellar #243. October 2010.
(2) http://www.systemyde.com/pdf/newt.pdf