

41CL Calculator



Every effort has been made to ensure the accuracy of the information contained herein. If you find errors or inconsistencies please bring them to our attention.

Copyright © 2011, Systemyde International Corporation. All rights reserved.

Notice:

“HP-41C”, “HP-41CV”, “HP-41CX” and “HP” are registered trademarks of Hewlett-Packard, Inc. All uses of these terms in this document are to be construed as adjectives, whether or not the noun “calculator”, “CPU” or “device” are actually present.

Acknowledgements:

This project never could have succeeded without Warren Furlow’s excellent Web site hp41.org. And even more important was his SDK41R6 software suite, for code development, and his V41 program for code debugging. Numerous people have answered my dumb questions on the Web site hpmuseum.org, and the book “Inside the HP-41” by Jean-Daniel Dodin was invaluable for getting a foothold on understanding the HP-41 operating system and register usage. Gene Wright was kind enough to be my voice at the HHC 2010 conference.

Table of Contents

Introduction	3
Can my calculator be upgraded?	4
Getting Started	7
Installing the 41CL Circuit Board	7
Initial Software Configuration	11
41CL Extra Functions	13
MMU Functions	13
Turbo Functions	14
Plug into Port/Unplug from Port Functions	16
Memory Block Functions	21
Memory/IO Read and Write Functions	24
Memory Buffer Functions	25
Flash Memory Functions	27
Serial Port Functions	29
Memory Management	35
The MMU and program addresses	35
The MMU and data addresses	37
Memory Organization	41
Flash memory organization	41
RAM memory organization	49
Using HEPAX	51
Using FORTH41	55
Optional Serial Connector	57
Installing the serial connector	58
Serial connector jack signals	60
Patching Code	61
41CL Extra Functions Summary	63
Error Messages	67

Revision History

Date	Changes	Pages
01/14/2011	Initial release	
01/15/2011	Misc typos	
01/19/2011	Added chapter on FORTH41, added rampage image	
01/23/2011	more typos	
01/25/2011	more typos	
02/10/2011	more typos, added pictures, serial connector section	
02/22/2011	more typos	
04/04/2011	Added paragraph on backing up 41C register memory Added "Patching Code" section	23 59 on
04/22/2011	Removed "current bank" as logical address option. Added clarification to serial functions. Miscellaneous clarifications to text. Added some new module images and mnemonics. Modified the "Patching Code" section for a different example.	
04/29/2011	Updates to Flash functions, color changes	
04/30/2011	Fixed some error messages (PLUGxx and YFERASE) Function Summary, Error Messages sections added	various 93 on
05/01/2011	Deleted ASTU module mnemonic. Included in ASTT.	18
05/03/2011	Spectral Analysis ROM added	
05/17/2011	Added Algebra, Sandmath II and Modified Advantage modules	
06/06/2011	Corrected XROM number for BCMW ROM. Added cautionary note about serial port connector.	18 59
06/11/2011	Added part list for serial connector	59-60
06/14/2011	Typo in register byte layout	38
06/16/2011	Added info on current drain Added info on Operating System usage of register address space	31 38-39
06/24/2011	Updated recover procedure for lost Y-functions.	12

Introduction

The 41CL takes advantage of modern technology to significantly add to the capabilities of the 41C system. In particular, the 41CL provides the following features:

- All features of an HP-41CX except for the Time Module. CX Time functions (the software) are included, but a Time module plugged into a Port is required for full timer functionality.
- Full 600-register Extended Memory is built in.
- Over 100 plug-in module images are built in. Functions are included to allow these images to be virtually plugged into a calculator Port and unplugged from a calculator Port.
- Turbo mode, which allows the calculator to run at up to 50X normal speed. Actual values available are 2X, 5X, 10X, 20X and 50X.
- 46 empty pages (4K in length) of Flash memory are available for non-volatile storage.
- 58 pages (4K in length) of RAM are available. All RAM is continuously powered.
- A sophisticated Memory Management Unit (MMU) allows full access to the large physical memory.
- Full bus compatibility for the Ports, allowing the use of any peripheral designed for the HP-41 system.
- A full-duplex serial port is available when the optional serial connector is used. This optional connector uses a 2.5mm stereo jack mounted in a blank port cover.

With these features, however, come some drawbacks:

- Power consumption is higher, at least while the calculator is off or in light sleep (between keystrokes). Where the original HP-41 required about 10uA while off, the 41CL requires about 110uA. This will lead to reduced battery life.

- The original HP-41 could retain memory contents for several minutes while the batteries were changed. Because of the higher current consumption, the 41CL only retains the memory contents for a few seconds while the batteries are out. For this reason, you should probably have an extra battery holder ready to go when changing batteries.
- The advanced technology used in the 41CL is a double-edged sword. The Flash memory, as well as the programmable logic devices used to implement the NEWT microprocessor, only guarantee data retention for 20 years.

Can my calculator be upgraded?

The 41CL is an upgrade created by replacing the CPU circuit board in a 41C/CV/CX with the 41CL circuit board. This replacement is only possible for calculators that actually have a CPU circuit board. The easiest way to tell if this is the case is to look at the HP-41 display. If the light part of the display has square corners, like those shown below, the calculator is a candidate for replacing the CPU circuit board.



Hewlett-Packard changed the display driver circuitry in the 41 series during production, and this change affected one component value on the CPU circuit board. The 41CL circuit board implements the component value used in later production units. Units with the correct display driver can be identified as follows:

- If your 41C (not CV or CX) has a serial number starting with “1954” or larger it uses the correct display driver.
- If your 41CV/CX has a serial number starting with “2003” or larger it uses the correct display driver.

It is theoretically possible to use the 41CL circuit board with the older display driver, but this requires soldering an extra capacitor to either the 41CL circuit board or to the calculator main board. If you really want to upgrade a calculator that uses the older display driver, please contact us directly for details.

In addition to the display driver change, Hewlett-Packard also experimented with a different method of connecting the CPU board to the main board. Unfortunately it is not possible to identify units that used this different connection method via the serial number. Identifying such a calculator is a step in the installation process covered in the next section.

Getting Started

The 41CL circuit board is designed to be a drop-in replacement for the original CPU circuit board, and the installation is not difficult. However, certain precautions must be taken to prevent damage to both old and new circuitry.

All integrated circuits are susceptible to damage from electrostatic discharge (ESD). If you have access to an electrostatically protected work area by all means use it. If not, make sure that you ground yourself immediately before starting the installation process. The best way to keep from generating a static charge is to not move around while working, so make sure you have everything required before starting the process. **Do not touch exposed conductors, and handle both the original CPU circuit board and the 41CL circuit board by the edges.** The 41CL circuit board has a 2mm space around the edges devoid of circuitry to facilitate handling in this manner.

Tools required for the installation are a small Philips-head screwdriver, an Xacto knife or similar, a pair of tweezers, and a small flashlight.

Installing the 41CL circuit board

Follow the steps below to perform the installation:

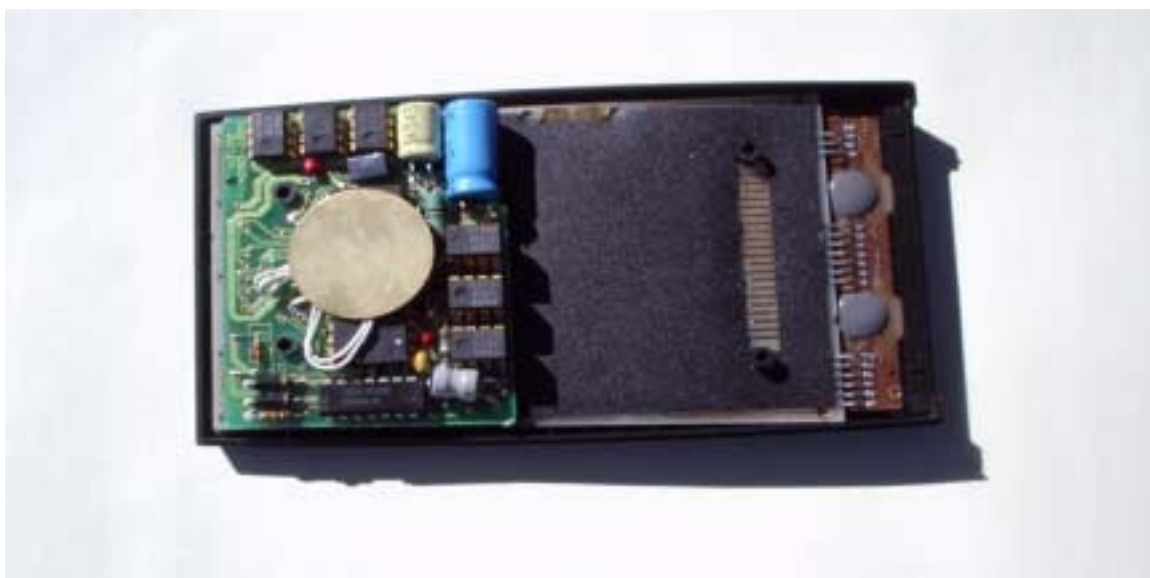
1. Read through all of these instructions before starting the installation process, to make sure that you understand each step. **We are not responsible if you damage or ruin your calculator or the 41CL circuit board while attempting this installation.**
2. Verify that your calculator is one that has a CPU circuit board. Only calculators with “square corners” on the LCD display panel (refer to the Introduction section for a picture) have CPU circuit boards.
3. Remove the battery case, by first sliding the case towards the top of the calculator until the bottom end of the battery case pops free. Now is a good time to install fresh batteries.
4. Carefully remove the four rubber feet, using a pointed knife to pry up one corner of a foot and a pair of tweezers to lift the foot from the case. Be careful not to damage the feet,

as replacements are difficult to find. They are attached to the calculator body using double-sided tape which can usually be reused.

5. Remove the four screws located in the recesses under the rubber feet. Be very careful not to lose them, as they are essentially impossible to find unless you want to buy 10,000 of them. The screws are all #2-28 trilobular thread-forming types. Those at the bottom of the case are 1/4" (they may be 3/8" if the calculator has been serviced), while those at the top of the case are 3/4"



6. Lift off the bottom case and the U-shaped center case section. Note the orientation (front-to-back) of the center case section, as it not symmetric.

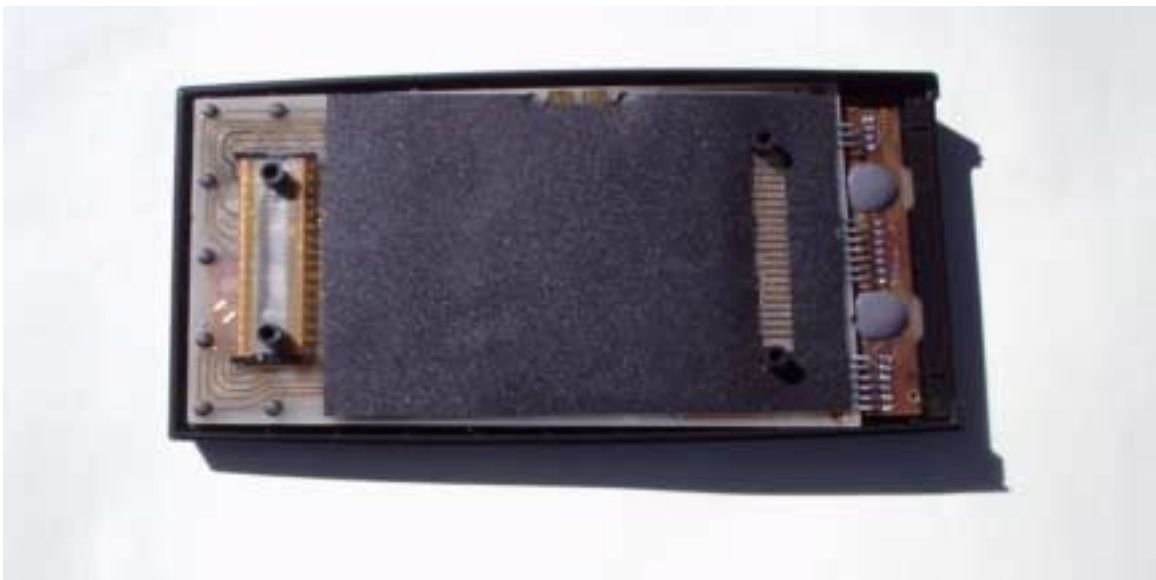


Don't worry if your old CPU looks slightly different from that shown in the picture. These boards went through several revisions during the life of the 41 series.

7. **STOP!** Before lifting the old CPU circuit board use a flashlight to look into the space between the CPU circuit board and the base circuit board. You will be able to see which type of connector is being used. The connector style in early production units will look like styrofoam. If this is the case, attempting to replace the CPU circuit board is not advised, as this type of connector material is not really reusable. Later production units use conductors rolled around a flexible plastic tubular form, which will appear shiny in the light from the flashlight. Being reusable, this type of conductor is suitable for use with the 41CL circuit board.

8. Before removing the old CPU circuit board, carefully inspect all four screw posts on the front case. If they are cracked or broken (a common problem) they will need to be repaired before re-assembling the calculator. Instructions for this repair can be found on the Museum of HP Calculators web site. Repair the screw posts before proceeding any further.

9. Using only the edges of the board, carefully lift the old CPU circuit board off of the base circuit board. The connectors will usually remain in place on the main circuit board. If by chance they do not, restore them to their proper location. The two halves of the connector are usually held together by a flexible piece of plastic that fits over the two screw posts just like the CPU circuit board



10. Using only the edges of the 41CL circuit board place this board on the main circuit board, with the two lower screw posts going through the holes in the 41CL circuit board

just as they did with the original CPU circuit board. Use the antistatic bag containing the 41CL circuit board to store the old CPU circuit board. Do not throw away the old CPU circuit board, as it may still have utility in the future. If you don't want to store the old CPU circuit board, send it to us. If you are installing the optional serial connector, this is the time to do it. Refer to the "Optional Serial Connector" section later in the document



11. Carefully fit the center case section (with the proper orientation) and bottom case back together with the remainder of the calculator body.

12. Re-install the four screws. The proper way to do this is to slowly turn the screw backwards until you feel the screw threads "click" into the threads in the post. Carefully tighten the screws. **Do not over-tighten the screws** as you risk cracking or shearing off the screw posts. It is best to hold the case sections tightly together with one hand while tightening the screws with the other hand, as this reduces the stress the screws place on the screw posts and case pieces.

13. Re-install the battery case (with the new batteries) and turn the calculator on. If there is no response the flexible connectors are not completely connecting the 41CL circuit board to the main circuit board, and you will need to either tighten the screws a little or reform the circular connector slightly. If garbage rather than the **MEMORY LOST** message appears, try removing the battery pack for 30 seconds and re-inserting it again. The display controller contains its own power-on-reset circuit, which sometimes does not properly synchronize with the CPU, leading to garbage in the display on power-up. This occasionally occurs even in the original design. In some cases it may be necessary to do this several times, with a longer time between battery pack insertions.

14. Once the 41CL circuit board installation is verified working, reinstall the calculator feet, and proceed to the initial configuration of the software.

Initial Software Configuration

The 41CL includes a set of functions that provide access to the new features of the NEWT microprocessor. When power is first applied or when the calculator is reset, resulting in the *MEMORY LOST* message, the 41CL Extra Functions are mapped to Page 7 to allow you to do the initial configuration of the calculator. This Page 7 mapping is enforced by the hardware for as long as the MMU is disabled.

During the initial configuration the 41CL Extra Functions must either be moved elsewhere so that Page 7 can be used by HP-IL Peripherals or the Page 7 entries in the MMU must be programmed to point to the 41CL Extra Functions after the MMU is enabled. Until the 41CL Extra Functions are moved from Page 7 HP-IL will not be available. Don't worry though, there will be no conflict with an 82160A HP-IL Module plugged into a Port on the calculator.

There are two images of the 41CL Extra Functions available, to prevent potential XROM conflicts. The default (Page 7) version uses XROM #15, while the second version uses XROM #31. The two copies are identical except for the XROM number. Only a few modules use XROM #15, so most users will not need to use the alternate version.

The minimum sequence for the initial configuration uses three 41CL Extra Functions (these functions are explained in the next section). If you don't need to use the advanced features of the 41CL this is sufficient for the initial configuration. This sequence is:

1. **XEQ ALPHA MMUCLR ALPHA** initializes all MMU entries in memory, making it safe to enable the MMU in the third step.
2. **ALPHA YFNZ ALPHA XEQ ALPHA PLUG1L ALPHA** plugs the XROM #15 version of 41CL Extra Functions into the lower half of Port 1 (which is Page 8). Since the MMU is still disabled this has no effect yet. Note that any port can be used for the 41CL Extra Functions. Plugging the 41CL Extra Functions into a Port allows the use of HP-IL.
2. (Alternate) **ALPHA 804070-8007 ALPHA XEQ ALPHA YPOKE ALPHA** programs the MMU for Page 7 to point at the XROM #15 version of 41CL Extra Functions. This saves half of a Port, but means that HP-IL will not be available. Since the MMU is still disabled this has no effect yet.
3. **XEQ ALPHA MMUEN ALPHA** enables the MMU, which starts the redirection of either Page 8 or Page 7 to the 41CL Extra Functions. When using Page 8 any

ROM module plugged into Port 1 will not be seen by the 41CL. However, Port 1 can still be used for modules with a fixed address such as the 82143A Printer, the 82153A Wand, the 82160A HP-IL Module, the 82182A Time Module or the 82242A IR Printer Module.

Enjoy your new 41CL calculator! The only thing to remember is that the 41CL Extra Functions **must** remain plugged into a Port for the new features to be available. If you inadvertently plug another module image into the Port used by the 41CL Extra Functions, taking the place of these functions as far as the OS is concerned, the only way to recover is by momentarily removing the battery pack.

41CL Extra Functions

The 41CL Extra Functions are required to provide access to the powerful new features available on the 41CL circuit board. Depending on your level of experience and how adventurous you are, you may or may not need all of these functions.

Casual users will only need the MMU (Memory Management Unit) Functions, the Turbo Functions, and the functions that allow you to Plug and Unplug module images to and from the calculator Ports.

If you are a HEPAX user you will need the Memory Block Functions and the Memory/IO Read and Write Functions. Refer to the Using HEPAX section for the details of how to do the initial setup of HEPAX memory.

Users interested in MCODE programming or building a custom module image will find the Memory Buffer Functions useful. The Flash Memory Functions are included for those users wishing to program the Flash memory on the 41CL circuit board beyond the initial programming. The Serial Port Functions allow users to control the serial port on the 41CL circuit board. Using the serial port will require installation of the optional serial port connector cable.

MMU Functions

The MMU Functions allow the user to clear, enable, disable, and test the status of the MMU. Normally only the **MMUCLR** and **MMUEN** commands will be necessary, to first initialize and enable the MMU after a *MEMORY LOST* condition.

MMUCLR (no arguments)

Executing **MMUCLR** clears the contents of all of the MMU registers in memory, by writing 0x0000 to these registers. This function should only be executed while the MMU is disabled, to prevent unpredictable results. Executing **MMUCLR** will result in all pages

(except for the Operating System, Extended Functions, Time Functions and 41CL Extra Functions) being fetched from the Ports rather than fetched from internal memory.

MMUDIS (no arguments)

Executing **MMUDIS** clears the global MMU enable bit inside the NEWT microprocessor. This automatically reassigns the 41CL Extra Functions to page 7, but only after the code returns to executing in one of the Operating System pages. This delayed switch allows the function to complete normally and return to the Operating System. This function does not affect the MMU contents.

MMUEN (no arguments)

Executing **MMUEN** sets the global MMU enable bit inside the NEWT microprocessor. This automatically disables mapping of the 41CL Extra Functions to page 7, but only after the code returns to executing in one of the Operating System pages. This delayed switch allows the function to complete normally and return to the Operating System. The 41CL Extra Functions must have been assigned to some other page prior to executing **MMUEN**, or the 41CL Extra Functions will no longer be available to the user.

MMU? (no arguments, returns with **0** (disabled) or **1** (enabled) in the X register)

Executing **MMU?** tests the state of the global MMU enable bit inside the NEWT microprocessor, returning with **0** in the X register if the MMU is disabled and **1** in the X register if the MMU is enabled. No stack lift occurs, so any data in the X register is lost.

Turbo Functions

The Turbo Functions give you complete control over the operating speed of the calculator. The performance in Turbo mode is not linear, because some operations must always occur at normal speed. For example, scanning the keyboard (which is done once per program line) always executes at normal speed. Similarly, accessing the display for any reason always executes at normal speed. All accesses of a physical Port occur at normal speed, along with several known timing loops in the Operating System.

Turbo modes increase the current consumption during normal operation, but have no effect during idle time (between keypress) or during the time when the calculator is off. The Turbo mode is preserved during idle time, but not when the calculator is turned off.

Like the Turbo mode performance, the current consumption as a result of Turbo mode is not linear. This is because only a fraction of the circuitry actually runs at a different speed during Turbo mode, in addition to the aforementioned normal special cases.

TURBOX (no arguments)

Executing **TURBOX** immediately disables any Turbo mode in effect. Note that any Turbo mode increases power consumption somewhat, so the Turbo modes should be used judiciously, and is really only appropriate when running programs.

TURBO2 (no arguments)

Executing **TURBO2** immediately enables the 2X Turbo mode.

TURBO5 (no arguments)

Executing **TURBO5** immediately enables the 5X Turbo mode.

TURBO10 (no arguments)

Executing **TURBO10** immediately enables the 10X Turbo mode.

TURBO20 (no arguments)

Executing **TURBO20** immediately enables the 20X Turbo mode.

TURBO50 (no arguments)

Executing **TURBO50** immediately enables the 50X Turbo mode.

TURBO? (no arguments, returns with the current Turbo mode in the X register)

Executing **TURBO?** tests the state of the Turbo control bits inside the NEWT microprocessor. The current Turbo speed is returned in the X register. The results returned will be one of **0**, **2**, **5**, **10**, **20** or **50**. No stack lift occurs, so any data in the X register is lost.

Plug into Port/Unplug from Port Functions

These functions allow the user to virtually plug and unplug module images from the calculator ports. The functions know the location in System Memory of the module images, providing an easy way for the user to control the configuration of the calculator without having to remember specific memory addresses. Refer to the Table below for the mnemonics for the different module images. No attempt will be made here to explain what the different module images do; it is assumed that the user has access to the documentation for any modules of interest.

Advanced users may enjoy exploring some of the poorly-documented images that are included in the 41CL. Advanced users can even build a new module image in memory and then virtually plug it into a port by specifying the relevant memory address for the Plug function.

PLUG1 (module image identifier ALPHA register)

Executing **PLUG1** inserts a module image into Port 1, which is pages 8 and 9 of the logical address space. The module identifier must be properly formatted in the ALPHA register (see below) or a **BAD ID** message will result. This function automatically programs the MMU registers for both pages 8 and 9 (all banks) as appropriate for the selected module image. If the module image cannot be used in pages 8 and 9 a **DATA ERROR** message will result. Modules that are only one page long will be loaded into the lower page and the upper page will be left empty. **PLUG2**, **PLUG3** and **PLUG4** are identical functions, except that they operate on Port 2 (pages A and B), Port 3 (pages C and D) and Port 4 (pages E and F) respectively.

PLUG1L (module image identifier ALPHA register)

The function **PLUG1L** is identical to **PLUG1** except that it only operates on the lower half of Port 1 (which is page 8 of the logical address space). This function can only be used with module images that are one page long. **PLUG2L**, **PLUG3L** and **PLUG4L** are identical functions, except that they operate on Port 2 (page A), Port 3 (page C) and Port 4 (page E) respectively.

PLUG1U (module image identifier ALPHA register)

The function **PLUG1U** is identical to **PLUG1** except that it only operates on the upper half of Port 1 (which is page 9 of the logical address space). This function can only be used with module images that are one page long. **PLUG2U**, **PLUG3U** and **PLUG4U** are

identical functions, except that they operate on Port 2 (page B), Port 3 (page D) and Port 4 (page F) respectively.

UPLUG1 (no arguments)

Executing **UPLUG1** removes the module image from Port 1, which is pages 8 and 9 of the logical address space. The function does this by clearing the MMU entries for these pages. **UPLUG2**, **UPLUG3** and **UPLUG4** are identical functions, except that they operate on Port 2, Port 3 and Port 4 respectively.

UPLUG1L (no arguments)

The function **UPLUG1L** is identical to **UPLUG1** except that it only operates on the lower half of Port 1 (which is page 8 of the logical address space). **UPLUG2L**, **UPLUG3L** and **UPLUG4L** are identical functions, except that they operate on Port 2, Port 3 and Port 4 respectively. Be careful using this function if you are planning on plugging a physical module into the corresponding Port, because the upper half of the Port will still be fetched from internal memory.

UPLUG1U (no arguments)

The function **UPLUG1U** is identical to **UPLUG1** except that it only operates on the upper half of Port 1 (which is page 9 of the logical address space). **UPLUG2U**, **UPLUG3U** and **UPLUG4U** are identical functions, except that they operate on Port 2, Port 3 and Port 4 respectively. Be careful using this function if you are planning on plugging a physical module into the corresponding Port, because the lower half of the Port will still be fetched from internal memory.

The table below shows the module images that are present in the Flash memory of the 41CL, along with the mnemonics for use with the various **PLUGxx** functions and any restrictions on module image placement. Four character mnemonics were chosen to allow easy-to-remember ALPHA contents, but only the first and last characters of the mnemonics are parsed by the **PLUGxx** functions.

Users should be careful of XROM conflicts when using module functions in programs. Refer to the original HP documentation for details. Advanced users can circumvent conflicts by copying a module image to RAM and then manually modifying the XROM number before plugging this modified image into a Port.

Mnemonic	Restrictions	Description	XROM number
A41P	full Port	HP Advantage Pac 1B	22, 24
AADV		Advantage Applications ROM	19
ADV1	Port 1 only	Adventure ROM, part 1	12
ADV2	Port 3 only	Adventure ROM, part 2	13
AECR	full Port	AECROM Module, page 1	18
AFDE	full Port	AFDC1 ROM	16, 17
AFDF	full Port	AFDC2 ROM	18, 19
AFIN		Autofinance Module	21
ALGG	full Port	Algebra ROM	6, 7
ALGY		Astrology ROM	31
ALPH		Alpha ROM	6
AOSX		AMC-OSX ROM	5
ASM4		Assembler 4	21
ASMB		Assembler 3	21
ASTT	Port 1 or 3 only	Astro-2010 ROM and Astro-2010 UI ROM	6, 8
AUTO		HP Autostart	10
AV1Q		Beechcraft ROM	31
AVIA		HP Aviation Pac 1A	19
B52B	full Port	Boeing-B52 Module	21, 31
BCMW		BCMW ROM	8
BESL	full Port	Bessel Functions ROM	2, 3
BLND		BufferLand ROM	8
BUFR	after Z41Z	Buffer ROM	8
CCDP	full Port	CCD Plus Module	9, 11
CCDR	full Port	CCD Module 1B	9, 11
CCDX		CCD OS/X	5
CHEM		Chemistry User Module	20
CHES	full Port	Chess ROM	8
CIRC		HP Circuit Analysis Pac 1A	6
CLIN		HP Clinical Lab & Nuclear Medicine Pac 1A	19
CURV	full port	CurveFit ROM	4, 5
CVPK	full Port	CVPAK ROM	21, 31
DA4C		Dasasm 4C	15
DACQ	full Port	HP Data Acquisition 1B	21, 31
DASM		Disasm 4D	15
DAVA		David Assembler 2C	2

DEVI	full Port	HP HP-IL Development Pac 1B	22, 24
DIIL		HP HP-IL Diagnostic	19
DMND		Diamond ROM	31
DYRK		Dyerka ROM	31
E41S	full Port	ES41 Module	4, 6
ESML		ESMLDL 7B Module	10
EXIO		HP Extended I/O 1A	23
EXTI		SKWID EXT-IL ROM	27
FINA		HP Financial Decisions Pac 1D	4
FUNS	Port 1 or 3 only	Funstuff ROM	10
GAME		HP Games Pac 1A	10
GMAS		GMAC 2 Module	31
GMAT	full Port	GMAC 3 Module	21, 31
HCMP		Hydracomp ROM	21
HEPR		HEPAX RAM Template	
HEPX		HEPAX 1D Module	7
HOME		HP Home Management Pac 1A	9
ICOD		ICODE ROM	19
ILBF		IL Buffer ROM	22
JMAT	full Port	JMB-Math ROM, Page 1	5, 6
JMTX	full Port	JMB-Matrix ROM	8
LBLs		Labels ROM	
LAND		LandNav Module	1
MADV	full Port	Modified Advantage Pac	24, 26
MATH		HP Math Pac 1D	1
MCHN		HP Machine Design Pac 1A	12
MDP1	full Port	MDP1 ROM	15, 16
MDP2	full Port	MDP2 RMOM	17, 18
MELB		Melbourne ROM	12
MILE	full Port	Military Engineering	21, 31
MLRM		MLROM	21
MTRX		Matrix ROM	7
MTST		MCTEST ROM	3
MUEC	full Port	Muecke ROM	21, 31
NAVI	full Port	HP Navigation Pac 1B	14
NCHP		NOV CHAP ROM	31
NFCR		NFCROM 1B	17
NPAC	full Port	Navpac ROM	14, 15
NVCM	full Port	NAVCOM 2	14, 15

OILW	full Port	Oilwell ROM	21, 31
P3BC	Port 1 or 3 only	Aviation Pac for P3B/C	9, 21, 31
PANA	full Port	Paname ROM	5, 9
PARI		ProtoPARIO ROM	14
PCDE		ProtoCODER2 ROM	16
PCOD		Pcoder 1A Module	16
PETR	full Port	HP Petroleum Fluids Pac 1A	15, 16
PLOT	full Port	HP Plotter Pac 1A	17, 18
PMLB		PPC-Melbourne ROM	12
POLY	full Port	Polynomial Functions ROM	6, 9
PPCM	full Port	PPC Module	10, 20
PRIQ	full Port	Pride ROM	21, 31
QUAT	full Port	Quaternions ROM	15, 16
RAMP		RAMpage ROM	15
REAL	full Port	HP Real Estate Pac 1A	11
ROAM		ROAM-0A ROM	5
ROMS		ROMSV01 ROM	9
SANA	Port 1 or 3 only	Sandmath-7 ROM (12K version)	2, 3, 6
SBOX	full Port	Sandbox ROM	8, 13
SECY	full Port	HP Securities Pac 1A	19
SGSG		SGS GAS Module	21
SIMM	Port 1 or 3 only	SIM Module	4, 10, 30, 31
SIMP		Simplex ROM	16
SKWD		SKWID ROM	8
SMCH	full Port	Speed Machine II ROM	21, 31
SMPL		Simplex Module	31
SMTS	full Port	Sandmath-7 ROM (8K version)	2, 3
SND2	full Port	Sandmath II ROM	2, 3
SPEC		Spectral Analysis ROM	8
STAN		HP Standard Applications Pac 1C	5
STAT		HP Statistics Pac 1B	2
STRE		HP Stress Analysis Pac 1A	8
STRU	full Port	HP Structural Analysis Pac 1B	7, 19
SUPR	full Port	SUP-R-ROM	21, 31
SURV		HP Survey Pac 1B	3
THER		HP Thermal & Transport Science Pac 1A	13
TOMS		TOMSROM	6
TOOL		Toolbox II ROM	13
TREK		Trekkies ROM	11

TRIH		83trinh ROM	9
UNIT		UnitConv ROM	10
USPS	full Port	USPS Module	21, 31
XXXA		4K user image at address 0x0C8000	
XXXB		4K user image at address 0x0D0000	
XXXC		4K user image at address 0x0D8000	
XXXD	full Port	8K user image at address 0x0E0000	
XXXE	full Port	8K user image at address 0x0E8000	
XXXF		16K user image (four banks) at address 0x0F0000	
YBFR		Y-Functions Buffer Area (RAM address 0x805000)	
YFNS		optional 41CL Extra Functions 1A (XROM #31)	31
YFNZ		default 41CL Extra Functions 1A (XROM #15)	15
Z41Z	full Port	HP41Z Complex Number ROM	1, 4
ZENR		ZENROM ROM	5
ZEPM		ZEPROM ROM	9
-RAM		4K block of memory (independent of bank)	
-16K		16K block of memory to Bank 1, 2, 3 and 4, in that order	

The **PLUGxx** functions can also be used to assign memory directly to the Ports. The upper three nibbles of the physical memory address (so that it begins on a 4K boundary) are specified, and the **PLUGxx** function assigns either one 4K block, independent of bank, or four contiguous 4K blocks in Bank 1, 2, 3, 4 order. When a full port **PLUGxx** function is used to assign memory directly only the lower half of the Port is assigned (just like a 4K module image).

To assign RAM memory to a Port, use the **-RAM** mnemonic for a bank-independent assignment (this is used for HEPAX RAM), or the **-16K** mnemonic for a bank-dependent assignment with the three most-significant digits of the physical address (since images always begin on 4K memory boundaries) in character positions 7-5 of the ALPHA register:

character							
	7	6	5	4	3	2	1
physical address	P5	P4	P3	-	R	A	M
physical address	P5	P4	P3	-	1	6	K

Memory Block Functions

The memory block functions allow the user to manipulate pages (4K blocks) of memory. In particular a page can be initialized to a user-selected value or copied to another location in memory. Note that these functions do not check whether the blocks are in Flash memory or RAM, and if you attempt to write to Flash memory the operation will appear to proceed, without any writes occurring.

Given that they are operating of 4096 memory locations, these functions take a fair amount of time to complete, especially at normal speed. For this reason, it is recommended that they always be executed while in Turbo mode.

YMCLR (hexadecimal address and data in ALPHA register)

Executing **YMCLR** writes the contents of the data field to a 4K block of memory starting at the address specified in the address field. The address and data must be properly formatted in the ALPHA register or a **DATA ERROR** message will result. The address field is truncated to create an address that is on a 4K boundary before the writes commence. Only memory addresses are valid for this function, and **DATA ERROR** will result if an I/O address is specified.

The table below shows the formatting required for the address and data in the ALPHA register for the **YMCLR** function. All characters, except for the required “-” characters, must be valid hex digits (0-9 or A-F). Any other character, or the absence of the “-” characters, will result in a “**DATA ERROR**” message.

character	11	10	9	8	7	6	5	4	3	2	1
physical address	P5	P4	P3	P2	P1	P0	-	D3	D2	D1	D0
logical address	L3	L2	L1	L0	-	B	-	D3	D2	D1	D0

All address characters (“P” or “L”) must be present, including leading zeros.

All data characters (“D”) must be present.

The bank identifier (“B”) in the logical address allows the bank to be directly specified (1, 2, 3 or 4), for those pages that support multiple banks. In addition, all banks (A) may be specified. For pages that do not support banks any of these options may be specified, because the field will be ignored.

Note that the **YMCLR** function cannot be used to write to physical modules. Also, this function executes at the current Turbo speed.

YMCPY (hexadecimal starting address pair in ALPHA register)

Executing **YMCPY** copies the contents of one 4K block of memory to another 4K block of memory. The source and destination address must be properly formatted in the ALPHA register or a **DATA ERROR** message will result. This function only allows copying block of memory that start on 4K boundaries and are 4K in length. Only memory addresses are valid for this function, and **DATA ERROR** will result if an I/O address is specified.

The table below shows the formatting required for the address and data in the ALPHA register for the **YMCPY** function. All characters, except for the required “>” character and optional “-” character must be valid hex digits (0-9 or A-F). Any other character, or the absence of the “>” characters, will result in a **DATA ERROR** message.

character	7	6	5	4	3	2	1
	source			>	destination		
physical address to physical address	P5	P4	P3	>	P5	P4	P3
physical address to logical address	P5	P4	P3	>	L3	-	B
logical address to physical address	L3	-	B	>	P5	P4	P3
logical address to logical address	L3	-	B	>	L3	-	B

All address characters (“P” or “L”) must be present, including leading zeros.

The bank identifier (“B”) in the logical address allows the bank to be directly specified (1, 2, 3 or 4), for those pages that support multiple banks. For pages that do not support banks any of these options may be specified, because the field will be ignored.

If the MMU is not enabled for a source logical address specified with the **YMCPY** function, the data is fetched from a physical module and copied to internal memory. In this case

the bank identifier is ignored, because the physical module will be in control of the bank select. This means that only Bank 1 of the module can be copied to memory.

Note that the **YMCPY** function can be used to write to a physical Port. In this case the function uses the WROM instruction, which only writes 10 bits. Also, this function executes at the current Turbo speed.

The **YMCPY** function is ideal for creating backups of system information such as the 41C register memory or the MMU contents. To backup the 41C register memory (including all user programs) simply copy the contents of memory starting at address 0x800000 to an available block of RAM. Use address 0x804000 to backup the MMU configuration.

Memory/IO Read and Write Functions

If you are not sure what you are doing, do not attempt to use these functions, particularly the memory write. The entire memory is accessible using these functions, which means that you can write directly to register memory, program the MMU, update the register address information or modify (i.e. corrupt) Operating System variables.

YPOKE (hexadecimal address and data in the ALPHA register)

Executing **YPOKE** writes directly to either memory or an internal I/O port. The address and data must be properly formatted in the ALPHA register (see below) or a **DATA ERROR** message will result. This function does not check the address except for proper formatting, so attempting to write to Flash memory is allowed, although it will be ignored because the function does not properly format the write for Flash memory.

Note that the peripheral version of this function only writes 12 bits to the peripheral port. This is okay because none of the peripheral write locations accept more than 12 bits.

YPEEK (hexadecimal address and data in the ALPHA register)

Executing **YPEEK** reads directly from either memory or the internal I/O port. The address and data must be properly formatted in the ALPHA register (see below) or a **DATA ERROR** message will result. The data field in the ALPHA register when the function is called is ignored, but is replaced with the actual data read from either the memory or the internal I/O port.

The table below shows the formatting required for the address and data in the ALPHA register for the **YPOKE** and **YPEEK** functions. All characters, except for the required “-”

characters, must be valid hex digits (0-9 or A-F). Any other character, or the absence of the “-” characters, will result in a **DATA ERROR** message.

character	11	10	9	8	7	6	5	4	3	2	1
physical address	P5	P4	P3	P2	P1	P0	-	D3	D2	D1	D0
logical address	L3	L2	L1	L0	-	B	-	D3	D2	D1	D0
port address				R	-	-	-	D3	D2	D1	D0

All address characters (“P”, “L” or “R”) must be present, including leading zeros.

All data characters (“D”) must be present, even for the **YPEEK** read function. The placeholder data characters will be replaced by the data read by the function.

The bank identifier (“B”) in the logical address allows the bank to be directly specified (1, 2, 3 or 4), for those pages that support multiple banks. In addition, for writes only, all banks (A) may be specified. For pages that do not support banks any of these options may be specified, because the field will be ignored. This means that only Bank 1 of a plug-in module can be accessed.

Memory Buffer Functions

The 41CL reserves one 4K block (one page) of System memory to be used as a buffer for assembling module images. The Y-Function Buffer Area is located at physical address 0x805000 - 0x805FFF, and has an associated Y-Functions Buffer Pointer stored at address 0x804010. The memory buffer functions provide a convenient way to move data to the buffer to assemble a module image without having to continuously specify the destination address. Instead, the lower twelve bits of the destination address are held in the Buffer Pointer, which is automatically incremented by the **YBUILD** function after use.

Thus, to assemble blocks of code the user merely initializes the Buffer Pointer to the start of the block, with either 0x000 if assembling a FAT, or 0x084 if assembling functions, and then copies blocks of memory, one after the other, to the buffer. The Buffer Pointer is updated to point at the next buffer location after each copy. Once an image is assembled, the FAT can be built using the regular **YPOKE** functions and the entire image moved to another location in memory using the **YMCOPY** function for use.

Of course all of the normal memory functions may be used with the Buffer Area, and indeed the region can also be used as normal memory when not being used as the buffer.

YBPNT (hexadecimal data in the ALPHA register, modifies the ALPHA register to return with the address and data of the Buffer Pointer in the ALPHA register)

Executing **YBPNT** writes data directly to the Y-Function Buffer Pointer. The data must be a valid four-digit hexadecimal number in the ALPHA register or a **DATA ERROR** message will result. Only the lower three digits of this value are used, and the most-significant digit is ignored and not changed by the buffer functions. The function returns with the normal **YPOKE** formatted physical address (see below) of the Buffer Pointer in the ALPHA register.

YBPNT? (no argument, returns with the address and data of Y-Function Buffer Pointer in ALPHA register)

Executing **YBPNT?** reads directly from Y-Function Buffer Pointer. The function returns with the normal **YPEEK** formatted physical address (see below) of the Buffer Pointer in the ALPHA register.

The table below shows the formatting for the address and data returned in the ALPHA register by the **YBPNT** and **YBPNT?** functions.

character	11	10	9	8	7	6	5	4	3	2	1
physical address	8	0	4	0	1	0	-	D3	D2	D1	D0

The data characters (“D”) are the contents of the Y-Function Buffer Pointer, which is stored at physical address 0x804010.

YBUILD (hexadecimal starting address and transfer length in the ALPHA register)

Executing **YBUILD** copies a block of data (up to 4096 words) from memory to the Y-Functions Buffer Area, starting at the location addressed by the Y-Functions Buffer Pointer. The source address and transfer length must be properly formatted in the ALPHA register or a **DATA ERROR** message will result. The Buffer Pointer is updated to point at the next Buffer Area location at the end of the transfer.

Care must be exercised because this function will wrap around the end of the Buffer Area, back to the beginning of the Buffer Area, if the transfer length specified so indicates.

The table below shows the formatting required for the address and data in the ALPHA register for the **YBUILD** function. All characters, except for the required “-” character, must be valid hex digits (0-9 or A-F). Any other character, or the absence of the “-” character, will result in a **DATA ERROR** message.

character	11	10	9	8	7	6	5	4	3	2	1
physical address	P5	P4	P3	P2	P1	P0	-	0	D2	D1	D0

All address characters (“P”) must be present, including leading zeros.

All transfer length characters (“D”) must be present, but D3 must be “0”, limiting the transfer length to 4096 or less. The number of words transferred is the transfer length. A transfer length of 4096 words is specified with transfer length of “000”.

The **YBUILD** function only supports physical addresses. This means that if you want to transfer data from a physical module to the Buffer Area the data must first be transferred to RAM memory so that a physical address can be specified.

Flash Memory Functions

If you are not absolutely sure of what you are doing, do not attempt to use these functions! While these functions do prevent you from corrupting the Operating System of the calculator, they still allow you to erase or modify the rest of the Flash memory. You must be familiar with how Flash memory operates before attempting to use these functions.

Flash memory has limited endurance, typically 100,000 write cycles, and is erased by sectors, which are 64K bytes (32K words, or eight pages) in the case of the 41CL. An erased Flash sector returns 0xFFFF in every location. Only 0’s can be written to any given location in Flash, which means that writes to Flash can only change a “1” to a “0” and never vice-versa.

During a Flash erase or write, no other accesses of the Flash memory are allowed. This means that these functions must be running out of RAM to work. Both functions check for this, and return with an error message if they are not running in RAM.

If you really want to use either of the Flash memory functions you must copy the entire 41CL Extra Functions image to RAM and then program the MMU to use this RAM copy of these functions

YFERASE (hexadecimal address in ALPHA register)

Executing **YFERASE** erases an entire sector (usually 32K words, or eight pages) of Flash memory. The address specified can lie anywhere within the sector and must be properly formatted in the ALPHA register or a **DATA ERROR** message will result.

This function checks that it is executing from RAM, and returns with a **CODE=ROM** error message if this is not the case. This function cannot be used to erase the Operating System area (the first sector in the Flash) and will return with the **OS AREA** error message if an address in the first sector of the Flash memory is specified.

The table below shows the formatting required for the address and data in the ALPHA register for the **YFERASE** function. All characters, must be valid hex digits (0-9 or A-F). Any other character will result in a **DATA ERROR** message.

character	6	5	4	3	2	1
physical address	P5	P4	P3	P2	P1	P0

All address characters (“P”) must be present, including leading zeros. An address that is in RAM (P5 is 8 or greater) will return with the **DST=RAM** error message.

The **YFERASE** function automatically includes a 1.6 second delay, because the Flash erase operation requires this much time to complete. The function will either return immediately with an error message, without executing, or wait for the entire 1.6 seconds before returning.

YFWR (hexadecimal starting address pair in ALPHA register)

Executing **YFWR** copies the contents of one 4K block (one page) of RAM memory to a 4K block of Flash memory. The source and destination address must be properly formatted in the ALPHA register or a **DATA ERROR** message will result. This function only allows copying block of memory that start on 4K boundaries and are 4K in length. Only physical memory addresses are valid for this function.

This function checks that it is executing from RAM, and returns with a **CODE=ROM** error message if this is not the case. This function cannot be used to write to the Operating System area (the first sector in the Flash) and will return with the **OS AREA** error message if an address in the first sector of the Flash memory is specified as the destination.

The table below shows the formatting required for the address and data in the ALPHA register for the **YFWR** function. All characters, except for the required “>” character, must be valid hex digits (0-9 or A-F). Any other character, or the absence of the “>” characters, will result in a **DATA ERROR** message.

character	7	6	5	4	3	2	1
	source			destination			
physical address to physical address	P5	P4	P3	>	P5	P4	P3

All address characters (“P”) must be present, including leading zeros. A destination address that is in RAM (destination P5 is 8 or greater) will return with the **DST=RAM** error message, and a source address that is in Flash (source P5 is 7 or less) will return with the **SRC=ROM** message.

The **YFWR** function executes at the current Turbo speed, with the approximate times shown in the table below. The function will either return immediately with an error message, without executing, or write the entire 4K block to Flash.

Speed	YFWR execution time
1x	~24 seconds
2x	~14 seconds
5x	~7 seconds
10x	~4 seconds
20x	~3 seconds
50x	~2 seconds

Serial Port Functions

The 41CL contains an RS-232 serial port, but unless you have the special PCB connector this functionality will not be available. However, these serial port functions are present on all 41CL circuit boards.

The serial port hardware is initialized and the baud rate is set to 1200 whenever the calculator is turned on. The serial port uses 8N1 format (eight bits of data, no parity, and one stop bit).

Depending on the baud rate, it may be advisable to run the 41CL in 50x Turbo mode when performing serial operations to make sure that the CPU has sufficient speed to keep up with the serial port. The serial port functions do not automatically increase the processor speed to 50x.

Even using the 50x Turbo mode, at higher baud rates there will be gaps between transmit characters and the receiver will need gaps between receive characters. This is because some instructions still run at 1x speed independent of the Turbo mode. Keep this restriction in mind when using the serial block transfer functions. If the source of serial data generates serial characters without any intervening idle time it will probably be necessary to use 1200 baud or lower to prevent receive overruns.

The serial functions only support physical addresses. This means that if you want to transfer data between a physical module and the serial port the data must be buffered in RAM memory before the final transfer to or from the physical module.

All of the serial data transfer functions contain a time-out feature to prevent locking up the machine in the case of an unavailable serial port. This time-out period is dependent on the Turbo mode, as shown in the table below:

Speed	Serial time-out period
1x	~950mS
2x	~635mS
5x	~475mS
10x	~475mS
20x	~475mS
50x	~475mS

In the absense of a valid RS-232 level on the serial receive input the RS-232 transceiver automatically powers down. But whenever there is a valid RS-232 level on the receive input the transceiver will be powered. This is a significant additon to the current drain on

the batteries, so the serial port should only be connected to a PC or other RS-232 equipment when actually using the serial port. The table below shows the typical current drain for the 41CL with and without the serial port powered up.

State	Serial port powered down	Serial port powered up
Off	110uA	2.8mA
Light Sleep (between keypresses)	4.2mA	6.9mA
Running (1x)	7.9mA	10.6mA
Running (50x)	11.3mA	14.0mA

SERINI (no arguments)

Executing **SERINI** initializes the serial port and sets the baud rate to 1200. Both the transmit and receive buffers are emptied and the receiver and transmitter are both set to the idle state. This command has no effect on the RS-232 driver.

BAUD96 (no arguments)

Executing **BAUD96** sets the baud rate for the serial port to 9600.

BAUD48 (no arguments)

Executing **BAUD48** sets the baud rate for the serial port to 4800.

BAUD24 (no arguments)

Executing **BAUD24** sets the baud rate for the serial port to 2400.

BAUD12 (no arguments)

Executing **BAUD12** sets the baud rate for the serial port to 1200. This is the default selection for the serial port, and is automatically selected when the calculator is turned on or when the **SERINI** command is issued.

YGETLB or **YGETUB** (hexadecimal address in the ALPHA register, returns **0** in X register to indicate success, **1** in X register to indicate failure, or **2** in X register to indicate overflow error)

Executing **YGETxB** reads a byte from the serial port and writes this byte to either upper byte (**YGETUB**) or the lower byte (**YGETLB**) of the memory location specified as the address. The address must be properly formatted in the ALPHA register (see below) or a “DATA ERROR” message will result. This function does not check the address except for proper formatting, so attempting to write to Flash memory is allowed, although it will be ignored unless you properly unlock the Flash write function.

The **YGETxB** functions will return after the time-out period if no receive byte is available. Upon exit, these functions place a **0** in the X register to indicate success, a **1** in the X register to indicate that a time-out occurred, and a **2** in the X register to indicate that an overflow occurred. No stack lift occurs, so the contents of the X register are lost. In the case of overflow the data in the receive buffer is discarded (since it is error anyway) and is not written to memory.

YPUTLB or **YPUTUB** (hexadecimal address in the ALPHA register, returns **0** in X register to indicate success or **1** in X register to indicate failure)

Executing **YPUTxB** reads a byte from the specified address and attempts to write it to the serial port. Either the upper byte (**YPUTUB**) or the lower byte (**YPUTLB**) of the memory location may be transmitted. The address must be properly formatted in the ALPHA register (see below) or a **DATA ERROR** message will result.

The **YPUTxB** functions will return after the time-out period if the transmitter is not able to accept a byte. Upon exit, these functions place a **0** in the X register to indicate success and a **1** in the X register to indicate that a time-out occurred. No stack lift occurs, so the previous contents of the X register are lost.

The table below shows the formatting required for the address in the ALPHA register for the **YGETxB** and **YPUTxB** functions. All characters must be valid hex digits (0-9 or A-F). Any other character will result in a **DATA ERROR** message.

character	<table border="1"> <tr> <td>6</td> <td>5</td> <td>4</td> <td>3</td> <td>2</td> <td>1</td> </tr> </table>						6	5	4	3	2	1
6	5	4	3	2	1							
physical address	P5	P4	P3	P2	P1	P0						

All address characters (“P”) must be present, including leading zeros.

YEXP (hexadecimal address and transfer length in the ALPHA register, returns **0** in X register to indicate success or **1** in X register to indicate failure)

Executing **YEXP** transfers an entire block of data (up to 4096 words) from internal memory to the serial port. The address and transfer length must be properly formatted in the ALPHA register (see below) or a **DATA ERROR** message will result. All addresses are valid for this function.

YIMP (hexadecimal address and transfer length in the ALPHA register, returns **0** in X register to indicate success, **1** in X register to indicate time-out, or **2** in X register to indicate overflow error.)

Executing **YIMP** transfers an entire block of data from the serial port into internal memory. The address and transfer length must be properly formatted in the ALPHA register (see below) or a **DATA ERROR** message will result. The address must be in RAM, because this function does not properly format writes for the Flash memory. Transferring data to a physical Port is not supported either.

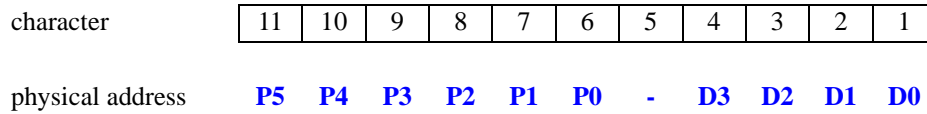
The **YIMP** and **YEXP** functions transfer words one byte at a time, in little-endian order (least significant byte first), from the lowest memory address (the one specified in the ALPHA register) to the highest memory address.

These functions will wait for the time-out period between bytes, but if this time limit is exceeded the function will immediately halt and return with an error. The **YIMP** function will also halt and return with an error if an overflow is detected. In the case of overflow the byte in error is discarded without being written to memory.

In any case the transfer length in the Alpha register will be updated to show the number of words remaining to be transferred. This allows the function to be started again without modifying the contents of the ALPHA register. Only the transfer of both bytes of a word counts as a successful transfer. If the transfer times out between the first and second byte of a word transfer, or the receiver overflows on either byte of a word transfer the entire transfer is deemed unsuccessful.

Successful completion will return **0** in the X register, while a time-out will return **1** in the X register. If an overflow occurs a **2** is returned in the X register. No stack lift occurs, so the previous contents of the X-register are lost.

The table below shows the formatting required for the address and data in the ALPHA register for the **YEXP** and **YIMP** functions. All characters must be valid hex digits (0-9 or A-F). Any other character will result in a **DATA ERROR** message.



All address characters (“P”) must be present, including leading zeros.

All transfer length characters (“D”) must be present, but D3 must be “0”, limiting the transfer length to 4096 or less. The number of words transferred is the transfer length plus one, allowing block transfers of from 1 to 4096 words (2 to 8192 bytes).

Memory Management

The original 41C system used dedicated ROM and RAM chips to implement the memory for the calculator, and the memory organization was mostly hidden from the user. The 41CL calculator replaces these custom memory chips with a pair of industry-standard memory devices, and besides the normal 41C view of memory, provides built-in functions that allow the user direct access to the physical memory.

The original ROM memories (including the plug-in ROMs in application pacs) are replaced with a single Flash (non-volatile, but re-programmable) memory, while the RAM chips are replaced with a single low-power RAM device. Given the advance of technology since the design of the original 41C system, these new memory devices provide significantly more storage than the original 41C system could even use. To take advantage of this increased storage capacity, the 41CL design includes a Memory Management Unit (MMU). Plug-in application pacs and peripherals are still supported, but separate application pacs are no longer really necessary.

The MMU takes the memory address, in either the program (ROM) address space or the data (RAM) address space, and translates this address into an address in either the Flash memory or the static RAM. This translation is completely automatic and transparent to the user.

Most users will never need to concern themselves with the operation of the MMU, as the new 41CL Extra Functions take care of programming the MMU in most cases. However, advanced users may need to understand how the MMU works and is programmed to take full advantage of some features of the 41CL calculator.

The MMU and program addresses

The 41C system uses a 16-bit (64K) program address, which is divided into sixteen pages of 4K each. For many of these pages, there can be up to four “banks”, which are selected under software control during program operation. This natural division of 4K pages is used by the MMU in the 41CL, so that each bank in each page can be mapped by the

MMU to a specific absolute address in the Flash memory or RAM on the 41CL circuit board.

To accomplish this mapping, the MMU takes the upper four bits of the program address (which selects the page), plus the two bits which select the bank, and forms a special memory address. The contents of this memory location, if the MMU is enabled, replaces the original upper four bits of the program address and forms a 24-bit address that is used to access the memory devices on the 41CL circuit board (the lower twelve bits are not modified by the MMU). Thus, in theory, any page can be mapped to any 4K page in the physical memory on the 41CL circuit board.

The 41CL does not allow user control of the mapping of some of the pages, to protect the Operating System (OS) of the calculator. So pages 0, 1, and 2 are never mapped by the MMU, because this is where the basic OS is stored. In addition, pages 3 and 5 are never mapped by the MMU, because this is where the X-Functions and Time Module functions for the calculator are located. But every other page can be mapped using the MMU.

The MMU entries are located in the 4K page of memory starting at address 0x804000, which is located in RAM. The actual address for an MMU entry is formed using this base address, with the page number replacing bits 7-4 and the bank number replacing bits 3-2. Note that the order of the banks follows the encoding used by the original 41C hardware, rather than conventional encoding. Since not all pages support banks only the following MMU entries are valid:

Physical Address	Contents
0x804040	MMU register for Page 4 (no banking supported)
0x804060	MMU register for Page 6 (no banking supported)
0x804070	MMU register for Page 7 (no banking supported)
0x804080	MMU register for Page 8, Bank 1
0x804084	MMU register for Page 8, Bank 3
0x804088	MMU register for Page 8, Bank 2
0x80408C	MMU register for Page 8, Bank 4
0x804090	MMU register for Page 9, Bank 1
0x804094	MMU register for Page 9, Bank 3
0x804098	MMU register for Page 9, Bank 2
0x80409C	MMU register for Page 9, Bank 4
.	.
.	.
0x8040F0	MMU register for Page F, Bank 1
0x8040F4	MMU register for Page F, Bank 3
0x8040F8	MMU register for Page F, Bank 2
0x8040FC	MMU register for Page F, Bank 4

Pages 8-F correspond to the Ports on the calculator, with pages 8-9 being Port 1, pages A-B being Port 2, and so on. The MMU entries for these pages are automatically handled by 41CL Extra Functions, so only the MMU entries for pages 4, 6, and 7 need to be manually programmed.

Page 4 is special to the OS, and only a few ROM images can be used in it. If you don't know what you are doing, don't try to use Page 4. Page 6 is normally used by a printer and Page 7 is used by HP-IL, so don't try to use either of these pages unless you don't need access to a printer or HP-IL.

The contents of an MMU memory locations are used as follows:

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EN	reserved			A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12

If the contents of bit 15, the Enable bit, are not set to one, the MMU entry is ignored and the corresponding page and bank will be fetched from a Port. Bits 14-12 are reserved for future use and must always be programmed with zeros.

The MMU and data addresses

Data addresses (“registers” in 41C parlance) are also translated by the MMU, but this translation is not programmable. Instead, registers are mapped to specific locations in the RAM on the 41CL board. This dedicated mapping is shown in the table below. Note that the 41C OS is not capable of addressing registers above 0x3FF, but space is reserved in the 41CL memory for register addresses up to 0xFFFF in case there are future enhancements to the OS code.

In the 41C system only the lower four bits of the register address can be specified by an instruction, and all of the upper register address bits are held in a dedicated register called (not surprisingly) the “register address”. In the 41CL this register address is stored in the RAM in a special location, at address 0x804000.

Like the original 41C, the 41CL preserves the data in RAM as long as power is applied. Unlike the original 41C, the 41CL allows RAM to be used to hold program data. All that is required for this type of operation is that the MMU point to a 4K block of RAM rather than a 4K block in the Flash portion of the address space. In the 41CL bit 23 of the physical memory address determines whether the address is in Flash (bit 23 is zero) or in RAM (bit 23 is one).

A seven-byte 41C register is stored in four successive memory locations as shown below:

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr lsb																
00	Byte 1								Byte 0							
01	Byte 3								Byte 2							
10	Byte 5								Byte 4							
11	unused								Byte 6							

The table below shows the organization of the 41C register memory in the physical memory of the 41CL circuit board. Because the OS manages the data in these locations, users are discouraged from attempting to modify any of these memory locations.

Physical Address	Contents	Operating System (OS) use
0x800000 - 0x800003	Register 000	T register
0x800004 - 0x800007	Register 001	Z register
0x800008 - 0x80000B	Register 002	Y register
0x80000C - 0x80000F	Register 003	X register
0x800010 - 0x800013	Register 004	LAST X register
0x800014 - 0x800017	Register 005	ALPHA register 1-7
0x800018 - 0x80001B	Register 006	ALPHA register 8-14
0x80001C - 0x80001F	Register 007	ALPHA register 15-21
0x800020 - 0x800023	Register 008	ALPHA register 22-28
0x800024 - 0x800027	Register 009	Temp ALPHA Scratch
0x800028 - 0x80002B	Register 00A	Unshifted Key Assign, OS status
0x80002C - 0x80002F	Register 00B	Program Return Stack
0x800030 - 0x800033	Register 00C	Program Return Stack, Program Pointer
0x800034 - 0x800037	Register 00D	Address Pointers
0x800038 - 0x80003B	Register 00E	FLAG register
0x80003C - 0x80003F	Register 00F	Shifted Key Assign, Program Line Number
0x800040 - 0x8000FF	Registers 10 - 3F	not visible to OS
0x800100 - 0x800103	Register 040	X memory
0x800104 - 0x800107	Register 041	X memory
.	.	X memory
0x8002F8 - 0x8002FB	Register 0BE	X memory

0x8002FC - 0x8002FF	Register 0BF	X memory
0x800300 - 0x800303	Register 0C0	Main memory
0x800304 - 0x800307	Register 0C1	Main memory
.	.	Main memory
0x8007F8 - 0x8007FB	Register 1FE	Main memory
0x8007FC - 0x8007FF	Register 1FF	Main memory
0x800800 - 0x800803	Register 200	not visible to OS
0x800804 - 0x800807	Register 201	X memory
.	.	X memory
0x800BBC - 0x800BBF	Register 2EF	X memory
0x800BC0 - 0x800C03	Registers 2F0 - 300	not visible to OS
0x800C04 - 0x800C07	Register 301	X memory
.	.	X memory
0x800FBC - 0x800FBF	Register 3EF	X memory
0x800FC0 - 0x800FFF	Registers 3F0 - 3FF	not visible to OS
0x801000 - 0x803FFF	Registers 400 - FFF	not currently utilized by OS
0x804000		Register Address buffer

Memory Organization

The tables below show the memory organization for both the Flash memory and RAM memory on the 41CL circuit board. Note that only the Operating System, Extended Functions and Time Function images are always accessible. While the MMU is disabled the SY41CL Extra functions are mapped to Page 7 to allow programming of the MMU. During this initial MMU setup the Extra Functions page should be assigned elsewhere so that Page 7 can be used by HP-IL Peripherals. Any of the other ROM images in Flash may be assigned to logical memory using the MMU. The MMU must be completely programmed (all pages and all banks) before issuing the Enable MMU command or unpredictable results may occur.

Flash memory organization

The initial Flash memory organization includes just about every module image imaginable, and should be sufficient for most users. However, the 41CL Extra Functions include functions for erasing and programming the Flash if necessary.

Starting Physical Address	Contents	Mnemonic	Description
0x000000	nut0-n		HP Operating System Rev N, page 0
0x001000	nut1-f		HP Operating System, Rev F, page 1
0x002000	nut2-l		HP Operating System, Rev L, page 2
0x003000	xfns3-2d		HP CX Extended Functions 2D
0x004000	service-2a	(note 1)	HP Service 2A
0x005000	xfns5-2d		HP CX Extended Functions 2D
0x006000	time-2c		HP CX Time Functions 2C
0x007000	yfns-1a	YFNZ	default 41CL Extra Functions 1A (XROM #15)
0x008000	adv11-1b	A41P	HP Advantage Pac 1B, page 1
0x009000	adv11-1b		HP Advantage Pac 1B, page 1
0x00A000	adv11-1b		HP Advantage Pac 1B, page 1

0x00B000	adv11-1b		HP Advantage Pac 1B, page 1
0x00C000	advu1-1b		HP Advantage Pac 1B, page 2, bank 1
0x00D000	advu2-1b		HP Advantage Pac 1B, page 2, bank 2
0x00E000	advu1-1b		HP Advantage Pac 1B, page 2, bank 1
0x00F000	advu1-1b		HP Advantage Pac 1B, page 2, bank 1
0x010000	aecroml	AECR	AECROM Module, page 1
0x011000	aecromu		AECROM Module, page 2
0x012000	autofinance	AFIN	Autofinance Module
0x013000	assembler3	ASMB	Assembler 3
0x014000	autostart	AUTO	HP Autostart
0x015000	aviation-1a	AVIA	HP Aviation Pac 1A
0x016000	boeing-b52l	B52B	Boeing B-52 Module, page 1
0x017000	boeing-b52u		Boeing B-52 Module, page 2
0x018000	ccd1-1b	CCDR	CCD Module 1B, page 1
0x019000	ccdu-2b		CCD Module 2B, page 2
0x01A000	chemuser	CHEM	Chemistry User Module
0x01B000	circuit-1a	CIRC	HP Circuit Analysis Pac 1A
0x01C000	clinical-1a	CLIN	HP Clinical Lab & Nuclear Medicine Pac 1A
0x01D000	dataacq1-1b	DACQ	HP Data Acquisition 1B, page 1
0x01E000	dataacqh-1b		HP Data Acquisition 1B, page 2
0x01F000	david-2c	DAVA	David Assembler 2C
0x020000	hpildev1-1b	DEVI	HP HP-IL Development Pac 1B, page 1
0x021000	hpildevu-1a		HP HP-IL Development Pac 1A, page 2
0x022000	hpil-diag	DIIL	HP HP- IL Diagnostic
0x023000	diamond	DMND	Diamond ROM
0x024000	es41l	E41S	ES41 Module, page 1
0x025000	es41u		ES41 Module, page 2
0x026000	esmdl1-7b	ESML	ESMLDL 7B Module
0x027000	extio-1a	EXIO	HP Extended I/O 1A
0x028000	finance-1d	FINA	HP Financial Decisions Pac 1D
0x029000	games-1a	GAME	HP Games Pac 1A
0x02A000	gmac2	GMAS	GMAC 2 Module
0x02B000	gmac3l	GMAT	GMAC 3 Module, page 1
0x02C000	gmac3u		GMAC 3 Module, page 2
0x02D000	hepax1-1d	HEPX	HEPAX 1D Module, bank 1
0x02E000	hepax2-1d	(note 2)	HEPAX 1D Module, bank 2
0x02F000	hepax3-1d		HEPAX 1D Module, bank 3
0x030000	hepax4-1d		HEPAX 1D Module, bank 4
0x031000	homemgt-1a	HOME	HP Home Management Pac 1A

0x032000	labels	LBL5	LABELS ROM
0x033000	landnav	LAND	LANDNAV Module
0x034000	math-1d	MATH	HP Math Pac 1D
0x035000	machine-1a	MCHN	HP Machine Design Pac 1A
0x036000	melbourne	MELB	Melbourne ROM
0x037000	mil-engl	MILE	Military Engineering, page 1
0x038000	mil-engu		Military Engineering, page 2
0x039000	mlrom	MLRM	MLROM
0x03A000	navl-1b	NAVI	HP Navigation Pac 1B, page 1
0x03B000	navu-1b		HP Navigation Pac 1B, page 2
0x03C000	nfcrom-1b	NFCR	NFCROM 1B
0x03D000	navcom2l	NVCM	NAVCOM 2, page 0
0x03E000	navcom2u		NAVCOM 2, page 1
0x03F000	apac1	P3BC	Aviation Pac for P3B/C, page 1
0x040000	apac2		Aviation Pac for P3B/C, page 2
0x041000	apac3		Aviation Pac for P3B/C, page 3
0x042000	apac4		Aviation Pac for P3B/C, page 4
0x043000	pcoder-1a	PCOD	Pcoder 1A Module
0x044000	petroleuml-1a	PETR	HP Petroleum Fluids Pac 1A, page 1
0x045000	petroleumu-2a		HP Petroleum Fluids Pac 2A, page 2
0x046000	plotterl-1a	PLOT	HP Plotter Pac 1A, page 1
0x047000	plotteru-2a		HP Plotter Pac 2A, page 2
0x048000	ppcl	PPCM	PPC Module, page 1
0x049000	ppcu		PPC Module, page 2
0x04A000	realestatel-1a	REAL	HP Real Estate Pac 1A, page 1
0x04B000	realestateu-1a		HP Real Estate Pac 1A, page 2
0x04C000	quater1	QUAT	Quaternion Module, page 1
0x04D000	quater2		Quaternion Module, page 2
0x04E000	securities-1a	SECY	HP Securities Pac 1A
0x04F000	standard-1c	STAN	HP Standard Applications Pac 1C
0x050000	sgs-gas	SGSG	SGS GAS Module
0x051000	sim12l	SIMM	SIM Module, page 1
0x052000	sim12u		SIM Module, page 2
0x053000	sim34l		SIM Module, page 3
0x054000	sim34u		SIM Module, page 4
0x055000	skwid	SKWD	SKWID ROM
0x056000	simplex	SMPL	Simplex Module
0x057000	stat-1b	STAT	HP Statistics Pac 1B
0x058000	stress-1a	STRE	HP Stress Analysis Pac 1A

0x059000	structl-1b	STRU	HP Structural Analysis Pac 1B, page 1
0x05A000	structu-1a		HP Structural Analysis Pac 1A, page 2
0x05B000	sup-r-roml	SUPR	SUP-R-ROM, page 1
0x05C000	sup-r-romu		SUP-R-ROM, page 2
0x05D000	survey-1b	SURV	HP Survey Pac 1B
0x05E000	thermal-1a	THER	HP Thermal & Transport Science Pac 1A
0x05F000	toolbox2	TOOL	Toolbox II ROM
0x060000	uspsl	USPS	USPS Module, page 1
0x061000	uspsu		USPS Module, page 2
0x062000	yfuns-1a	YFNS	alternate 41CL Extra Functions 1A (XROM #31)
0x063000	41zl	Z41Z	HP41Z Complex Number ROM, page 1
0x064000	41zu		HP41Z Complex Number ROM, page 2
0x065000	buffer	BUFR	Buffer ROM (used with HP41Z)
0x066000	alpha	ALPH	Alpha ROM
0x067000	astro-1	ASTT	Astro-2010 ROM, page 1
0x068000	astro-2		Astro-2010 ROM, page 2
0x069000	astro-3		Astro-2010 ROM, page 3
0x06A000	astr0-ui		Astro-2010 UI ROM
0x06B000	chess	CHES	Chess ROM, page 1
0x06C000	rubiks		Chess ROM, page 2
0x06D000	playground	FUNS	Funstuff ROM, page 1
0x06E000	puzzles		Funstuff ROM, page 2
0x06F000	casino		Funstuff ROM, page 3
0x070000	action		Funstuff ROM, page 4
0x071000	jmb-mathl	JMAT	JMB-Math ROM, page 1
0x072000	jmb-mathu		JMB-Math ROM, page 2
0x073000	amc-math	SANA (note 3)	SandMath-7 ROM, page 1
0x074000	hl-math		SandMath-7 ROM, page 2
0x075000	amc-mtrx		SandMath-7 ROM, page 3
0x076000			Blank (loaded as SandMath-7 page 4)
0x077000	trekkies	TREK	Trekkies ROM
0x078000	unitconv	UNIT	UNITCONV ROM
0x079000	dungeons	ADV1	Adventure ROM, Page 1
0x07A000	caves		Adventure ROM, Page 2
0x07B000	adventure		Adventure ROM, Page 3
0x07C000	castle		Adventure ROM, Page 4
0x07D000	bmf5	ADV2	Adventure ROM, Page 5
0x07E000	bmf6		Adventure ROM, Page 6

0x07F000	bmf7		Adventure ROM, Page 7
0x080000			Blank (loaded as Adventure ROM page 8)
0x081000	statistics	CURV	CurveFit ROM, Page 1
0x082000	curvefit		CurveFit ROM, Page 2
0x083000	matrix1	JMTX	JMB-Matrix ROM, Page 1
0x084000	matrix2		JMB Matrix ROM, Page 2
0x085000	disasm-4d	DASM	DisAsm 4D ROM
0x086000	ext-il	EXTI	SKWID EXT-IL ROM
0x087000	ccd-osx	CCDX	CCD-OSX ROM
0x088000	sandboxL-3d	SBOX	Sandbox ROM, page 1
0x089000	sandboxU-3d		Sandbox ROM, page 2
0x08A000	panameL-3c	PANA	Paname ROM, page 1
0x08B000	panameU-2a		Paname ROM, page 2
0x08C000	83trinh	TRIH	83trinh ROM
0x08D000	adv-app	AADV	Advantage Applications ROM
0x08E000	afdc1L	AFDE	AFDC1 ROM, page 1
0x08F000	afdc1U		AFDC1 ROM, page 2
0x090000	afdc2L	AFDF	AFDC2 ROM, page 1
0x091000	afdc2U		AFDC2 ROM, page 2
0x092000	amc-osx	AOSX	AMC-OSX ROM
0x093000	assembler4	ASM4	Assembler 4 ROM
0x094000	astrology	ALGY	Astrology ROM
0x095000	av1	AV1Q	AV1 ROM
0x096000	bcmw	BCMW	BMW ROM
0x097000	cvpak1	CVPK	CVPAK ROM, page 1
0x098000	cvpak2		CVPAK ROM, page 2
0x099000	dyerka	DYRK	Dyerka ROM
0x09A000	forth4	(note 4)	HP-41 FORTH ROM, page 1
0x09B000	forth5		HP-41 FORTH ROM, page 2
0x09C000	hydracmp	HCMP	Hydracomp ROM
0x09D000	icode	ICOD	Icode ROM
0x09E000	laitram	(note 1)	Laitram-XQ2 ROM
0x09F000	mctest	MTST	Mctest ROM
0x0A0000	mdp1L	MDP1	MDP1 ROM, page 1
0x0A1000	mdp1U		MDP1 ROM, page 2
0x0A2000	mdp2L	MDP2	MDP2 ROM, page 1
0x0A3000	mdp2U		MDP2 ROM, page 2
0x0A4000	mueckeL	MUEC	Muecke ROM, page 1
0x0A5000	mueckeU		Muecke ROM, page 2

0x0A6000	navpac2L	NPAC	Navpac ROM, page 1
0x0A7000	bw1a		Navpac ROM, page 2
0x0A8000	oilwell-L	OILW	Oilwell ROM, page 1
0x0A9000	oilwell-U		Oilwell ROM, page 2
0x0AA000	pario	PARI	Pario ROM
0x0AB000	pcoder-1a	PCDE	Pcoder ROM
0x0AC000	ppc-melb	PMLB	PPC-Melbourne ROM
0x0AD000	pride1	PRIQ	Pride ROM, page 1
0x0AE000	pride2		Pride ROM, page 2
0x0AF000	roam-0a	ROAM	ROAM ROM
0x0B0000	romsv01	ROMS	Romsv01 ROM
0x0B1000	simplex	SIMP	Simplex ROM
0x0B2000	spd-machine2L	SMCH	Speed Machine II ROM, page 1
0x0B3000	spd-machine2U		Speed Machine II ROM, page 2
0x0B4000	tomsrom	TOMS	Tomsrom ROM
0x0B5000	zenrom-3B	ZENR	Zenrom ROM
0x0B6000	zeprom	ZEPM	Zeprom ROM
0x0B7000	turbol	CCDP	CCD Plus ROM, page 1
0x0B8000	turbou		CCD Plus ROM, page 2
0x0B9000	hepram	HEPR	HEPAX RAM Template
0x0BA000	disasm4c	DA4C	DisAsm 4C ROM
0x0BB000	RAMpage	RAMP	RAMpage ROM
0x0BC000	spectral	SPEC	Spectral Analysis ROM
0x0BD000	algebra1	ALGG	Algebra ROM, page 1
0x0BE000	algebra2		Algebra ROM, page 2
0x0BF000	Empty		Reserved Flash
0x0C0000	bessel1	BESL	Bessel Functions ROM, page 1
0x0C1000	bessel2		Bessel Functions ROM, page 2
0x0C2000	polyn1	POLY	Polynomial Functions ROM, page 1
0x0C3000	polyn2		Polynomial Functions ROM, page 2
0x0C4000	ilbuffer	ILBF	IL Buffer ROM
0x0C5000	novch	NCHP	NOV CHAP ROM
0x0C6000	bufrland	BLND	BufferLand ROM
0x0C7000	matrix	MTRX	Matrix ROM
0x0C8000	Empty	XXXA	4K User ROM image
0x0C9000 - 0x0CF000	Empty		User Flash
0x0D0000	Empty	XXXB	4K User ROM image
0x0D1000 - 0x0D7000	Empty		User Flash
0x0D8000	Empty	XXXC	4K User ROM image

0x0D9000 - 0x0DF000	Empty		User Flash
0x0E0000	Empty	XXXXD	8K User ROM image, page 1
0x0E1000	Empty		8K User ROM image, page 2
0x0E2000 - 0x0E7000	Empty		User Flash
0x0E8000	Empty	XXXxE	8K User ROM image, page 1
0x0E9000	Empty		8K User ROM image, page 2
0x0EA000 - 0x0EF000	Empty		User Flash
0x0F0000	Empty	XXXxF	16K User ROM image, bank 1
0x0F1000	Empty		16K User ROM image, bank 2
0x0F2000	Empty		16K User ROM image, bank 3
0x0F3000	Empty		16K User ROM image, bank 4
0x0F4000	Empty		User Flash
0x0F5000	Empty		User Flash
0x0F6000	smath2L	SND2	Sandmath II ROM, page 1
0x0F7000	smath2U		Sandmath II ROM, page 2
0x0F8000	advml	MADV	Modified Advantage ROM, page 1
0x0F9000	advml		Modified Advantage ROM, page 1
0x0FA000	advml		Modified Advantage ROM, page 1
0x0FB000	advml		Modified Advantage ROM, page 1
0x0FC000	advmU1		Modified Advantage ROM, page 2, bank 1
0x0FD000	advmU2		Modified Advantage ROM, page 2, bank 2
0x0FE000	advmU1		Modified Advantage ROM, page 2, bank 1
0x0FF000	advmU1		Modified Advantage ROM, page 2, bank 1

Note 1: This is a Page 4 take-over ROM. This type of image can only be used by directly programming the MMU entry for page 4. However, once this is done the 41CL Extra Functions are no longer available, so the only way to undo this action is to either remove the batteries, or reset the calculator with **Backspace-ON**.

Note 2: The HEPAX ROM image in the 41CL does not include the function which automatically attempts to relocate the HEPAX ROM to the lowest available page, because the 41CL hardware does not support this operation. A by-product of this missing function is that the HEPAX RAM is not automatically initialized, so the HEPAX RAM must be manually initialized before use. A template for initializing the HEPAX RAM is included in the 41CL images. Refer to the “Using HEPAX” section for details.

Note 3: The SandMath ROM can be loaded as an 8K image if the matrix operations are not required. Use the mnemonic “SMTS” for this case.

Note 4: The FORTH ROM is hard coded to use Pages 4 and 7. This means that the MMU must be programmed directly if you want to use FORTH ROM. Refer to the “Using FORTH41” section for details.

RAM memory organization

The register memory, the register address and MMU contents are stored in RAM as indicated in the table below. All other RAM may be assigned to logical memory using the MMU. Do not attempt to modify the contents of register memory or the register address directly. These areas are managed by the Operating System. MMU contents should only be directly modified while the MMU is disabled, as unexpected results may occur otherwise. The **PLUGxx** and **UPLUGxx** commands may be used at any time, because they modify the MMU entries for all four banks at one time.

Physical Address	Contents
0x800000 - 0x800003	Register 000
0x800004 - 0x800007	Register 001
.	.
.	.
0x800038 - 0x80003B	Register 00E
0x80003C - 0x80003F	Register 00F
0x800040 - 0x8000FF	Registers 10 - 3F (not visible to OS)
0x800100 - 0x800103	Register 040 (X memory)
0x800104 - 0x800107	Register 041 (X memory)
.	.
.	.
0x8002F8 - 0x8002FB	Register 0BE (X memory)
0x8002FC - 0x8002FF	Register 0BF (X Memory)
0x800300 - 0x800303	Register 0C0
0x800304 - 0x800307	Register 0C1
.	.
.	.
0x8007F8 - 0x8007FB	Register 1FE
0x8007FC - 0x8007FF	Register 1FF
0x800800 - 0x800803	Register 200 (not visible to OS)
0x800804 - 0x800807	Register 201 (X memory)

.	.
0x800BBC - 0x800BBF	Register 2EF (X memory)
0x800BC0 - 0x800C03	Registers 2F0 - 300 (not visible to OS)
0x800C04 - 0x800C07	Register 301 (X memory)
.	.
0x800FBC - 0x800FBF	Register 3EF (X memory)
0x800FC0 - 0x800FFF	Registers 3F0 - 3FF (not visible to OS)
0x801000 - 0x803FFF	Registers 400 - FFF (not utilized by OS)
0x804000	Register Address storage location
0x804001 - 0x80400F	Reserved system memory
0x804010	Y-Functions Buffer Pointer storage location
0x804011 - 0x80403F	Reserved system memory
0x804040	MMU register for Page 4 (no banking supported)
0x804041 - 0x80405F	Reserved system memory
0x804060	MMU register for Page 6 (no banking supported)
0x804061 - 0x80406F	Reserved system memory
0x804070	MMU register for Page 7 (no banking supported)
0x804071 - 0x80407F	Reserved system memory
0x804080	MMU register for Page 8, Bank 1
0x804084	MMU register for Page 8, Bank 3
0x804088	MMU register for Page 8, Bank 2
0x80408C	MMU register for Page 8, Bank 4
.	.
0x8040F0	MMU register for Page F, Bank 1
0x8040F4	MMU register for Page F, Bank 3
0x8040F8	MMU register for Page F, Bank 2
0x8040FC	MMU register for Page F, Bank 4
0x804100 - 0x8043FF	Reserved system memory
0x804400 - 0x804FFF	Reserved system memory
0x805000 - 0x805FFF	Y-Functions Buffer Page
0x806000 - 0x83FFFF	User RAM (58 4Kx16 pages)

Using HEPAX

The HEPAX module was one of the most complex third-party 41C modules ever created. It had hardware and software that automatically relocated the module to the lowest unused page of memory, and provided RAM that resided in program memory for file storage. Not all of the features of the HEPAX module are supported by the 41CL calculator, and this section will discuss the details of how to use the HEPAX image included in the 41CL Flash memory.

The HEPAX module used all four banks of the page where the code was located. The 41CL only support banks in pages 5 and 8-F. Since page 5 is used by the Extended Function and Time module, the HEPAX image can only be loaded into pages 8-F, which correspond to the Ports. Pages 6 and 7 can still be used for HEPAX RAM, because RAM does not use multiple banks, but the HEPAX image will not work if loaded there.

The HEPAX module also supported write-protection for pages of HEPAX RAM, using an opcode that was ignored by the processor in the 41C. This opcode is also ignored by the NEWT processor in the 41CL, but the write-protect feature is not supported by the 41CL hardware. Keep this in mind if you attempt to write-protect pages of HEPAX memory.

Note that the 41CL Extra Functions make it easy to create a backup copy of RAM pages, using the YMPCY function to copy an entire page of memory to another location in physical memory. If you are doing work that might inadvertently corrupt HEPAX RAM, which is what the write-protect feature was for, try creating a copy of the HEPAX RAM page first. Since the HEPAX code does not “know” about physical memory, the copy will effectively be write-protected.

The main issue for HEPAX users is that the 41CL hardware does not support the automatic relocation of the HEPAX image. The MMU makes this function unnecessary, so the HEPAX image in the 41CL has been modified to eliminate the automatic relocation feature. Unfortunately a by-product of this modification is that the HEPAX RAM will not be automatically initialized at start-up. Instead, you will need to initialize any HEPAX RAM when the HEPAX image is first plugged into a port.

The 41CL provides a template for initializing HEPAX RAM pages, which simplifies the process considerably, but you will still need to initialize one or two locations in each RAM page.

The 41C code listing below shows the template for initializing HEPAX RAM. This template, which is stored at address 0x0B9000 in the 41CL Flash, should be copied to each 4K block of RAM that is going to be used for HEPAX RAM. The “fixed value” locations are checked by the HEPAX software, and values not matching those shown in the listing template will cause an error when the HEPAX code attempts to use the RAM.

Two locations in each HEPAX RAM page contain address pointers. These address pointers hold four-bit page numbers in the least-significant digit. The pointer at address 0xFE7 points to the previous page of HEPAX RAM, and a value of 0x000 here marks a page of HEPAX RAM as the first in the chain. The pointer at address 0xFE8 points to the next page of HEPAX RAM, and a value of 0x000 here marks a page of HEPAX RAM as the last in the chain.

```

;*****
  .TITLE      "HEPAX RAM"

  .HP

XROM 13

  .FILLTO    0FE6

#000        ; FE7 Previous page identifier
#000        ; FE8 Next page identifier
#091        ; FE9 fixed value
#000        ; FEA
#000        ; FEB
#000        ; FEC
#090        ; FED fixed value
#000        ; FEE
#091        ; FEF fixed value
#000        ; FF0
#0E5        ; FF1 fixed value
#00F        ; FF2 fixed value
#200        ; FF3 fixed value

  .FILLTO    0FFE

```

As an example, the sequence of commands listed below uses four pages of 41CL RAM (at addresses 0x808000, 0x809000, 0x80A000 and 0x80B000) as HEPAX RAM assigned to pages C through F (Ports 3 and 4).

First, the RAM is initialized by copying the HEPAX RAM template to these four pages of RAM memory:

```
ALPHA 0B9>808 ALPHA
XEQ ALPHA YMCPY ALPHA
```

```
ALPHA 0B9>809 ALPHA
XEQ ALPHA YMCPY ALPHA
```

```
ALPHA 0B9>80A ALPHA
XEQ ALPHA YMCPY ALPHA
```

```
ALPHA 0B9>80B ALPHA
XEQ ALPHA YMCPY ALPHA
```

Next, the HEPAX RAM pointers in these pages must be initialized. Note that the template loads 0x0000 into all of the pointers to start with, so only the non-zero pointers need to be written:

```
ALPHA 808FE8-000D ALPHA
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 809FE7-000C ALPHA
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 809FE8-000E ALPHA
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 80AFE7-000D ALPHA
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 80AFE8-000F ALPHA
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 80BFE7-000E ALPHA
XEQ ALPHA YPOKE ALPHA
```

Finally the RAM pages are plugged into the Ports:

```
ALPHA 808-RAM ALPHA
XEQ ALPHA PLUG3L ALPHA
```

```
ALPHA 809-RAM ALPHA
XEQ ALPHA PLUG3U ALPHA
```

```
ALPHA 80A-RAM ALPHA
XEQ ALPHA PLUG4L ALPHA
```

```
ALPHA 80B-RAM ALPHA
XEQ ALPHA PLUG4U ALPHA
```

At this point the HEPAX RAM is initialized to a point where the HEPAX code can recognize and use it. To verify that you've done everything correctly, try executing a HEPDIR command. The display should return *H:DIR EMPTY*, and clearing this from the display should show *2610*, which is the size of four pages of HEPAX RAM.

Using FORTH41

The regular PLUGxx functions cannot be used to insert the FORTH ROM into the 41CL logical memory because this ROM is hard-coded to use page 4 and page 7. Instead, you will need to program the MMU entries for pages 4 and 7 directly.

It's a little more complicated than just programming the two MMU entries though. The problem is that between programming the two MMU entries the 41C OS is going to check certain locations in program memory, including the start of page 4 and places near the end of pages 5 through F. This means that the OS may get confused with only half of the FORTH ROM visible. The way around this potential problem is to disable the MMU during programming. The sequence of commands shown below will enable FORTH41. None of the other MMU entries are affected by this programming, so the configuration of the calculator is retained.

First, the MMU is disabled. This has no effect on the contents of the MMU:

```
XEQ ALPHA MMUDIS ALPHA
```

Next, the MMU for pages 4 and 7 are programmed to point at the FORTH ROM image:

```
ALPHA 804040-809A ALPHA  
XEQ ALPHA YPOKE ALPHA
```

```
ALPHA 804070-809B ALPHA  
XEQ ALPHA YPOKE ALPHA
```

Finally, the MMU is re-enabled:

```
XEQ ALPHA MMUEN ALPHA
```

This method can be used whenever you want to use page 4, 6 or 7 for internal code. Be careful though, not to plug a real module that uses one of these pages into a Port while you

are using one of these pages for internal code, as this will create a conflict on the Port signals, and may damage either the 41CL or the modules, or both.

Disabling the FORTH ROM is just as simple. First, the MMU is disabled:

```
XEQ ALPHA MMUDIS ALPHA
```

Next, the MMU entries for pages 4 and 7 are cleared:

```
ALPHA 804040-0000 ALPHA  
XEQ ALPHA YPOKE ALPHA
```

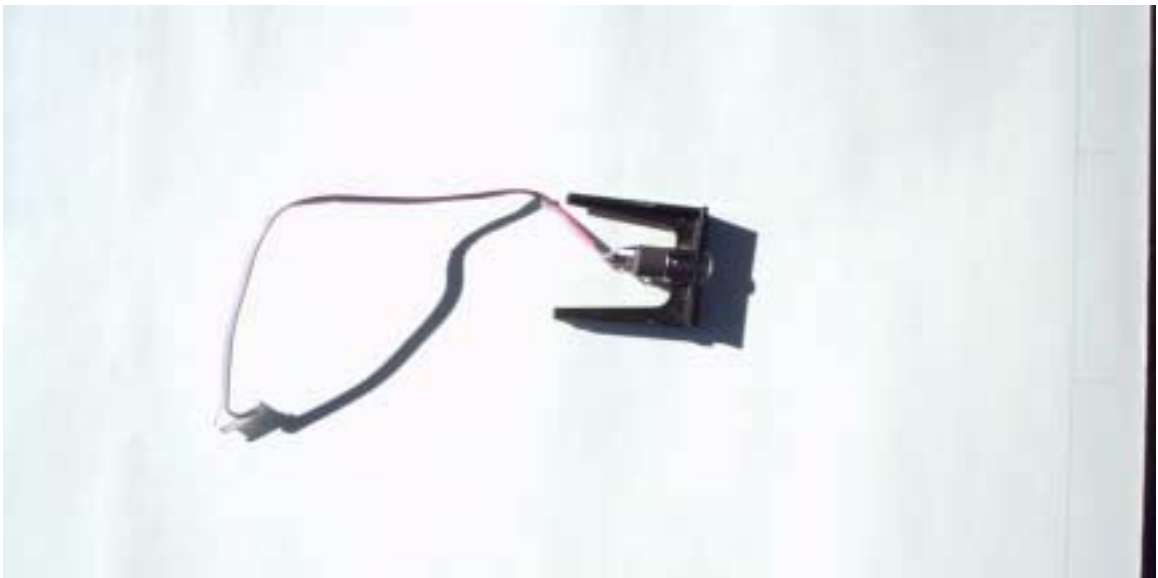
```
ALPHA 804070-0000 ALPHA  
XEQ ALPHA YPOKE ALPHA
```

Finally, the MMU is re-enabled:

```
XEQ ALPHA MMUEN ALPHA
```

Optional Serial Connector

The 41CL circuit board contains a simple RS-232C serial port. The Receive Data, Transmit Data and Ground signals for the serial port are present on the programming connector for the CPLD on the board. The optional serial connector contains a plug for this connector, connected through a cable to a 2.5mm stereo jack mounted in a blank Port cover. The serial connector is designed to occupy Port 1 on the calculator only. While it can be plugged into any other Port, doing so will interfere physically with the remaining Ports because of the cable.



Installing the serial connector

Follow the steps below to perform the installation:

1. Read through all of these instructions before starting the installation process, to make sure that you understand each step. **We are not responsible if you damage or ruin your calculator or the 41CL circuit board while attempting this installation.**

2. Insert the connector end of the serial connector through Port 1 and in between the space between the calculator body and the flexible Port connectors



3. Route the cable down the side of the calculator body, between the edge of the battery compartment and the outside of the case. It is helpful to use double-sided tape to hold the cable in place next to the battery compartment between the Port and the battery charger Port.



Note how the cable turns sharply near the bottom of the Port to take advantage of the space that will exist between the upper and lower halves of the case. It is easiest to route the cable with the center section of the case in place on the lower case half.

4. Carefully plug the connector on the cable into the connector jack on the 41CL circuit board with the “CP” label next to it. This is the programming connector for the CPLD. The connector used is very fragile and was not really designed for multiple insertions, so take your time, but make sure the plug is fully seated in the connector on the 41CL circuit board.

The two connectors for programming the CPLD and the FPGA on the board are identical, and plugging the serial port connector into the wrong one will damage the board. The 41CL circuit board is shipped with a blank plug in the FPGA connector to prevent accidentally inserting the serial connector in the wrong jack.

This step is where the double-sided tape holding the cable in place is useful, as it keeps the cable from popping out of the channel where it resides.



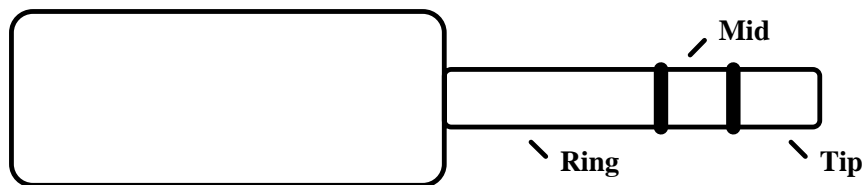
5. Fit the two case halves together, making sure not to pinch the serial cable.

CAUTION! The tension holding a 2.5mm plug in the serial connector jack is higher than the tension holding the blank port cover in the calculator body. This means that trying to pull out the plug will tend to pull the blank port cover out of the calculator, potentially damaging the internal connections to the serial connector jack. So always remember to hold the blank port cover in place when attempting to remove the serial port plug from the calculator.

Serial connector jack signals

The serial connector jack signals are assigned as follows:

- **Tip:** Transmit Data from the point of view of the 41CL calculator. This should connect to pin 2 of a female DB9 connector for use with a PC.
- **Mid:** Ground. This should connect to pin 5 of a female DB9 connector for use with a PC.
- **Ring:** Receive Data from the point of view of the 41CL calculator. This should connect to pin 3 of a female DB9 connector for use with a PC.



The type of cable required to connect the 41CL calculator to a PC is also used for older digital cameras and cell phones, so it can still be found. However, be aware that two different signal arrangements were used for these types of cables, depending on the manufacturer. Part number BC20213-6 from www.cableclub.com is compatible with the 41CL calculator.

If you want to try to construct a serial connector yourself the part numbers are listed below. Note that the cable comes with 10 conductors, so you will have to trim it down to three conductors. Refer to the 41CL schematic for the signal connections on the circuit board connector.

Plug for circuit board connector: 455-2189-ND from Digi-Key

Multi-conductor cable: MB20G-10-ND from Digi-Key

2.5mm Stereo Jack: 161-7000-EX from Mouser

Patching Code

The 41CL Extra Functions make it simple to patch software preloaded into the 41CL. Most of the software preloaded into the Flash memory can be copied to the RAM memory, patched, and then the MMU can be used to reference this patched code. Only pages that cannot be relocated by the MMU cannot be patched. This is pages 0-3 and 5 (which hold the Operating System, the Extended Functions and the Time Functions). As mentioned previously, these pages are protected from modification to prevent users from inadvertently turning the 41CL into a brick.

To illustrate what is required to patch code, go through the steps below to modify the ROM ID of the 41CL Extra Functions to avoid a conflict with other modules.

To patch code the ROM image must first be copied to an available page in RAM using the **YMCPY** function. Note that when running 1x the **YMCPY** function takes about 30 seconds to copy the entire 4k page. If you are running in 50x Turbo mode the copy will take about 8 seconds. In either case, because the function is machine code the calculator display will be completely blank during the copy.

```
ALPHA 007>80C ALPHA  
XEQ ALPHA YMCPY ALPHA
```

In this example just one location needs to be modified for proper operation. Note that whenever patches are specified only the 4K relative address will be given. Use the upper nibbles of the RAM address chosen to hold the patched code for the remainder of the address.

0x000 should be 0x0010 to change the ROM ID to 16 (which is 10 in hexadecimal)

Use the **YPOKE** function to write directly to the desired location in RAM memory. Since we are using the page starting at address 0x80C000 to hold the patched ROM image the following keystrokes are required to apply this patch:

```
ALPHA 80C000-0010 ALPHA  
XEQ ALPHA YPOKE ALPHA
```

Then we can use the **PLUG1L** function to insert the patched image into the lower half of Port 1 (or wherever the Y-Functions are, or will be, located):

```
ALPHA 80C-RAM ALPHA  
XEQ ALPHA PLUG1L ALPHA
```

It's that simple! The MMU in the 41CL, along with the ability to peek and poke memory, makes this machine a hacker's delight.

Function Summary

This appendix lists all of the 41CL Extra Functions, along with the arguments and return values.

Function	Arguments (in ALPHA)	Returns (X register)	Returns (ALPHA)	Notes
MMUCLR				
BAUD12				
BAUD24				
BAUD48				
BAUD96				
MMUCLR				
MMUDIS				
MMUEN				
MMU?		<i>0</i> <i>1</i>		MMU is disabled MMU is enabled
PLUG1	module ID			
PLUG1L	module ID			
PLUG1U	module ID			
PLUG2	module ID			
PLUG2L	module ID			
PLUG2U	module ID			
PLUG3	module ID			
PLUG3L	module ID			
PLUG3U	module ID			
PLUG4	module ID			
PLUG4L	module ID			
PLUG4U	module ID			
SERINI				
TURBOX				
TURBO2				

TURBO5				
TURBO10				
TURBO20				
TURBO50				
TURBO?		<i>0</i> <i>2</i> <i>5</i> <i>10</i> <i>20</i> <i>50</i>		Turbo mode disabled 2x Turbo mode enabled 5x Turbo mode enabled 10x Turbo mode enabled 20x Turbo mode enabled 50x Turbo mode enabled
UPLUG1				
UPLUG1L				
UPLUG1U				
UPLUG2				
UPLUG2L				
UPLUG2U				
UPLUG3				
UPLUG3L				
UPLUG3U				
UPLUG4				
UPLUG4L				
UPLUG4U				
YBPNT	data			
YBPNT?			address/data	Buffer Pointer value is in the data field
YBUILD	address/length			
YEXP	address/length	<i>0</i> <i>1</i>		Successful transfer Time-out
YFERASE	address			
YFWR	address			
YGETLB	address	<i>0</i> <i>1</i> <i>2</i>		Successful transfer Time-out Overflow
YGETUB	address	<i>0</i> <i>1</i> <i>2</i>		Successful transfer Time-out Overflow
YIMP	address/length	<i>0</i> <i>1</i> <i>2</i>		Successful transfer Time-out Overflow
YMCLR	address/data			4k word initialization
YMCPY	address pair			4k word transfer
YPOKE	address/data			

YPEEK	address/data		address/data	input data field is replaced with actual data
YPUTLB	address	<i>0</i> <i>1</i>		Successful transfer Time-out
YPUTUB	address	<i>0</i> <i>1</i>		Successfull transfer Time-out

Error Messages

This appendix lists all possible error messages returned by the 41CL Extra Functions, along with the meaning.

Error Message	Function	Meaning
<i>BAD ID</i>	PLUG1 PLUG1L PLUG1U PLUG2 PLUG2L PLUG2U PLUG3 PLUG3L PLUG3U PLUG4 PLUG4L PLUG4U	Invalid module ID in ALPHA
<i>CODE=ROM</i>	YFERASE YFWR	Trying to execute function from Flash
<i>DATA ERROR</i>	PLUG1 PLUG1L PLUG1U PLUG2 PLUG2L PLUG2U PLUG3 PLUG3L PLUG3U PLUG4	Invalid hexadecimal in ALPHA

	PLUG4L PLUG4U YBPNT YBUILD YEXP YFERASE YFWR YGETLB YGETUB YIMP YMCLR YMCPY YPOKE YPEEK YPUTLB YPUTUB	
<i>DST=RAM</i>	YFERASE YFWR	Attempting Flash operation on RAM
<i>OS AREA</i>	YFERASE YFWR	Attempting Flash operation on Operating System
<i>SRC=ROM</i>	YFWR	Trying to transfer from Flash to Flash