OWNER'S MANUAL

ProtoSYSTEM ™

 PROTOTECH, INC.

P. O. BOX 12104

BOULDER, CO. 80303

T A B L E   O F   C O N T E N T S

L I S T   O F   P R O G R A M S

LIST OF TABLES

ProtoSYSTEM OVERVIEW

The Prototech, Inc. ProtoSYSTEM is a flexible and expandable interface between the HP 41 C/V calculator and various peripheral and memory devices. The modular design of the ProtoSYSTEM allows the user to start with the ProtoSYSTEM INTERFACE and one peripheral board, and expand the power of his system as his needs increase.

The ProtoSYSTEM INTERFACE is the initial device required to allow the user to add on any of the peripheral boards. It plugs directly into any of the ports of the calculator. By adding peripheral boards to the system the user can:

* Plug in Hewlett Packard Memory and Application modules and switch them on or off from the calculator under program control (ProtoROM);
* Plug in blocks of 4096 words of memory that can be used to emulate Hewlett Packard modules and EPROMs and can contain user language and/or microcode routines (ProtoCODER);
* Plug in semi-permanent Hewlett Packard-format EPROMs (Erasable Programmable Read Only Memories) containing user language or microcode routines (ProtoEPROM);
* Plug in various other memory and peripheral devices.

The ProtoSYSTEM INTERFACE provides control, address, and data signals for all peripheral boards. It also contains a battery to maintain the contents of HP Memory modules and the memory of the ProtoCODER when the ProtoSYSTEM is not plugged in to the calculator. All boards can be programmed from the keyboard or under program control by a sequence of three operations:

1) Enter the data to be programmed into ALPHA as a series of hexadecimal characters,
2) Convert these hex characters to binary in the X register using "HN" or "CODE", and
3) Execute SIGN.

The ProtoSYSTEM will program one of its boards depending on the contents of the X register.

CONNECTING AND DISCONNECTING THE ProtoSYSTEM

The ProtoSYSTEM INTERFACE cable plugs directly into any of the four ports of the HP 41. When connecting or disconnecting the ProtoSYSTEM from the calculator first turn off the calculator. If you do not, you may damage the ProtoSYSTEM or the HP 41. To insert the cable into the calculator, line up the plug with any port with the flat surface of the plug upwards, then gently push the plug into the port of the calculator until it snaps in place. If it does not go in easily, check for obstructions -- DO NOT FORCE IT. To remove the ProtoSYSTEM from the calculator, grasp the plug by the sides and pull straight away from the calculator.

When the ProtoSYSTEM is connected to the HP 41, it uses the calculator's battery. If you have HP Memory modules or ProtoCODER data that you wish to retain when you disconnect the ProtoSYSTEM, be sure that the battery is installed.

CONNECTING PERIPHERAL BOARDS TO THE ProtoSYSTEM

The ProtoSYSTEM has a 25-pin bus connector that passes signals between the boxes containing ProtoSYSTEM boards. To plug a box onto the ProtoSYSTEM, check that all pins protruding from the box are straight, then place the box directly over the INTERFACE box (or the top box). Line up the pins to go into the slot of the lower box. Gently press the top box down until it rests on the lower box. DO NOT FORCE IT. If you have trouble pressing the boxes together, look for obstructions or bent pins. The peripheral boards can be plugged in in any order, however, ProtoROM boards containing HP Memory modules should be as close to the ProtoSYSTEM INTERFACE as possible. The greater the distance from the calculator, the more likely that data will not be transmitted properly because of increased capacitance of the wires and traces.

POWER SUPPLY

The ProtoSYSTEM contains a Duracell PX-30 3 volt battery (or equivalent) to maintain HP Memory module and ProtoCODER memory. To install it, remove the top cover of the ProtoSYSTEM INTERFACE box by removing the two screws at the upper corners of the box. Lift off the cover and remove the old battery by raising the boards slightly and sliding the battery out between the boards. Insert the new battery with the + side upwards, and put the box back together.

DEVICE SELECTING

The ProtoSYSTEM has its own addressing system so that several boards of the same type can be connected simultaneously and used independently. For example, two (or more) ProtoROM boards can be used at the same time by giving them different DEVICE SELECT addresses which are set by switches on each board. Up to 16 devices can be addressed directly using addresses 0 - F in hexadecimal (0000 - 1111 in binary). Each address specifies a different board. The device select information is specified by the user, and is contained in the X register that the user sets up each time he wants to program a board.

SETTING UP THE PROGRAMMING DATA

The user must set up data in the X register to program the ProtoSYSTEM peripheral boards. This data depends on the board to be programmed.

The X register (and all calculator data registers) consists of 56 "bits" of information. Each bit can be either 1 or 0 (on or off, set or clear). To save space when writing this data, these 56 bits are grouped in 14 blocks of 4 bits, each called a nybble or digit. Each nybble is represented by a decimal digit (0 - 9) or a letter (A - F) in the hexadecimal ("hex") numbering system. Table 1 lists the conversions from hex to binary. By convention the bits are numbered from left (high order) to right (low order) from 55 to 0. The digits or nybbles are numbered from left to right from 13 to 0. It is necessary to understand the bit structure of the X register for the user to be able to program the ProtoSYSTEM properly. Table 2 lists the bits of the X register that are used by all ProtoSYSTEM devices during programming.

EXAMPLE:

The X register contains the following bits of data:
0000 0010 1001 1111 1110 1010 0111 1011 0001 0100 0110 1101 0011 0101

Referring to Table 1, this can be written in hex as: 029FEA7B146D35. In this example, nybble 9 is E in hex, which means that bits 39, 38, and 37 are 1 (on) and bit 36 is 0 (off).

TABLE 1: CONVERTING HEXADECIMAL TO BINARY

BINARY	HEX	BINARY	HEX	BINARY	HEX	BINARY	HEX
0000	0	0001	1	0010	2	0011	3
0100	4	0101	5	0110	6	0111	7
1000	8	1001	9	1010	A	1011	B
1100	C	1101	D	1110	E	1111	F

TABLE 2: DATA BITS USED TO PROGRAM THE ProtoSYSTEM

BITS	CONTENTS
55-54	These bits must both be 1. This signals the ProtoSYSTEM to program one of the boards. Normally both of these bits will not be set, so that the SIGN function will operate as it usually would.
53-4	These bits contain the data, address, and other control information. To determine the data to be coded here, consult the Appendix for the device you are programming. For example, bits 53-44 contain the address for the ProtoCODER, but for the ProtoROM they contain the port number and on/off code.
3-0	These bits contain the DEVICE SELECT address. This is a hex digit from 0 to F. Each board has its own address, set by switches on the board, so that the user can tell the ProtoSYSTEM which board he wishes to program.

WRITING THE DATA TO THE ProtoSYSTEM

To program any ProtoSYSTEM board, the user needs a program to convert hexadecimal digits in ALPHA into a binary number in the X register. No external EPROM is needed. There are three steps to write an item of data to a ProtoSYSTEM board:

1) Determine the appropriate data using the Appendix for the board to be programmed and convert to hex using Table 1. Enter this string of hex digits into ALPHA. For example, to enter "029FEA7B146D35" into ALPHA:
(ALPHA) (SHIFT) 0 (SHIFT) 2 (SHIFT) 9 F E A (SHIFT) 7 B (SHIFT) 1 (SHIFT) 4 (SHIFT) 6 D (SHIFT) 3 (SHIFT) 5 (ALPHA).
2) Execute a program to convert the hex data in ALPHA into binary data in X. Several programs are available to do this:
"CODE" in Synthetic Programming on the HP 41C by William Wickes,
"HN" in the PPC ROM (see Appendix 4),
"CODE" in the PPC EPROM 2 (see Appendix 4),
"CODE" in the JIMROM written by Jim DeArras,
"CODE" in the Prototech, Inc. NFCROM.
The purpose of this program is to convert the data in ALPHA character by character into nybbles in the X register.
3) Execute the SIGN function. This signals the ProtoSYSTEM to program one of its boards. It will examine the X register and pick out the data needed and pass it along the 25-pin bus to the appropriate board, which will use this data to program itself.

PPC ROM COMPATABILITY

According to Keith Jarrett, the PPC ROM uses the SIGN function on non-normalised numbers in two of its routines. "CD" run with more than 20 characters in ALPHA could cause a write to the ProtoSYSTEM. This would happen only when the 21st character from the right end of the ALPHA string is hex C0 or greater. A write to the ProtoSYSTEM may also occur when using "NH" with the left-most nybble of the X register being C, D, E, or F. In these two cases, care must be taken when using both the ProtoSYSTEM and the PPC ROM.

EFFECTS OF MASTER CLEAR

When a master clear is executed (see the Hewlett Packard Owners Handbook), any HP Memory and Extended Memory modules that are switched to be on-line to the calculator will be cleared. If the Memory module is plugged into the ProtoSYSTEM but is not switched on-line, then it will not be cleared. No other ProtoSYSTEM boards are affected by a master clear.

The HP 41 calculator can address up to 65536 (64K) 10-bit words of information in ROMs (Read Only Memories). This includes the system ROMs, HP Extension and Application modules, and EPROMs (Erasable Programmable Read Only Memories). These 64K words are separated into 16 "pages" of 4096 (4K) words each, numbered from 0 to F in hexadecimal (see Table 1). The 4096 words on each page are numbered in hex from 000 to FFF. The ROM addresses are specified as PWWW where P is the page number and WWW is the word number on the page. Some of these 16 pages are preassigned for system use and cannot normally be used to contain user programs. Table 3 shows which pages of the ROM space are preassigned.

Any page from 5 to F that is unused by normal HP peripherals can be used to contain a ROM, EPROM, or a page of ProtoCODER memory. Be careful to not have more than one device addressed at a given page. If you do, you will probably crash your calculator. If you crash, remove the modules from the calculator then remove the battery pack for a few seconds. This will usually fix a crash.

TABLE 3: HEWLETT PACKARD ROM PREASSIGNED PAGES

PAGE	CONTENTS
0,1,2	These ROMs contain the HP 41 operating system. They tell the calculator how to act like an HP 41.
3	Currently unused by any HP ROMs, and not normally addressed by the HP operating system. Future HP calculators may use it. It can contain user programs, routines and data if the user jumps into this page with a Class 1 microcode instruction.
4	Currently used by the HP Diagnostic ROM which helps the HP service people diagnose problems with calculators and modules. This could contain an alternate operating system. When ON is pressed the contents of this ROM (if it is plugged in to the system) are executed.
5	Used by the HP Time Module.
6	Used by either the 82143 or 82162 HP Printer.
7	Used by the HP- IL module.
8,A,C,E	These pages are normally used by HP modules. The page number is determined by the port where the module is plugged in. For example, if you have the GAMES module plugged in to port 3, it will be addressed at page C. Port 1 is addressed at page 8, port 2 at page A, port 3 at page C, and port 4 at page E.
9,B,D,F	These pages are not normally used by HP modules unless the module is 8K(like REAL ESTATE). If a module is 4K (most are) and plugged into port 1, then it is addressed at page 8, and page 9 is left unused. When an 8K module is plugged in, it is addressed at two consecutive pages (8-9, A-B, C-D, E-F) depending on the port.

CHAPTER 4: ProtoSYSTEM PRODUCT INFORMATION

The following boards are now available from Prototech, Inc.

* ProtoSYSTEM INTERFACE - The main control board that plugs into the HP 41. It is necessary to provide data, control, and address signals for the other peripheral boards.

* ProtoROM - Allows the user to plug in up to 4 HP modules including Memory, Extension, and Application modules. Each module can be individually switched on or off, and told what port to be "strapped" to. The switching can be done from the keyboard or under program control.

* ProtoEPROM - Allows the user to plug in one Hewlett Packard-format EPROM set containing user language programs and/or microcode.

* ProtoCODER - Provides the user with 4096 (4K) words of memory which is programmable in microcode (the machine language of the microprocessor in the calculator). No external EPROM is needed to program the ProtoCODER.

Several other boards are under development. Inquire for more details.

CHAPTER 5: WARRANTY, SERVICE, ASSISTANCE

LIMITED WARRANTY

The ProtoSYSTEM INTERFACE and all ProtoSYSTEM peripheral boards are warranted against defects in materials and workmanship for a period of 90 days from the date shipped from Prototech, Inc. Within this warranty period, Prototech, Inc. will rapair or at its option replace a defective part at no charge to the owner, provided that Prototech, Inc. is contacted within the warranty period for shipping instructions. There will be a charge for repairs after the warranty period has expired. Prototech, Inc. assumes no responsibility for damages, either direct or consequential, from the use of its products. Prototech, Inc. will have no obligation to modify or update products after sale. This warranty does not apply to products damaged by accident or misuse, or to products that have been modified by anyone other than Prototech, Inc. This warranty is made in lieu of all other warranties, either express or implied.

SERVICE

If your ProtoSYSTEM INTERFACE or any ProtoSYSTEM peripheral board requires service, contact Prototech, Inc. for instructions.

ASSISTANCE

If you need technical or applications assistance relating to the use of the ProtoSYSTEM, contact Prototech, Inc. at (303)-447-9883 (no collect calls), or write to:

PROTOTECH, INC.
P. O. BOX 12104
BOULDER, CO. 80303

ProtoROM PURPOSE

The ProtoROM expands the number of available ports for the user to plug in HP modules. Each ProtoROM board attached allows the user to plug in four additional HP modules. Each module can be switched on to or off from the calculator under program or keyboard control, in addition to being told into which port the calculator will think it is plugged. Although the calculator can only have up to four Applications modules on-line at one time (see Chapter 3), up to 64 modules can be plugged in with the ProtoROM boards, and then switched on only when they are needed. The ProtoROM can also be used to increase the register memory available to the calculator using a technique called "page switching". More information on this technique is available in the PPC ROM Users Manual (see Appendix 4).

PLUGGING IN HP MODULES

To insert a module, turn off the calculator, then place the module into one of the slots in the side of the ProtoROM box, with the printing on the module upright, and the label on the ProtoROM box upwards. Gently push the module all the way in. Do not force it. If it does not go in easily, remove and look for obstructions. To remove a module, turn off the calculator, then grasp the handle on the module and pull straight out.

SETTING THE DEVICE SELECT SWITCHES

Each ProtoROM board has a set of four switches inside so that the ProtoSYSTEM can tell the boards apart when more than one is connected. These switches set the device select address (see Chapter 2). To set the switches in your ProtoROM, you need to remove the ProtoROM from the ProtoSYSTEM and remove the top of the ProtoROM box. Remove the screws from the base of the box then turn the box upright and lift the cover off. Inside on the circuit board you will see a small set of four switches. Determine an unused device select address (one that none of the other ProtoSYSTEM boards is using) which will be a hexadecimal number from 0 to F. Convert this to binary using Table 1. Orient the ProtoROM box so that the 25-pin bus is away from you, then set the switches from left to right for each binary digit (0=off, 1=on). On the left side of the block of switches is an arrow that shows the direction to push the switch for on or off. After setting the switches properly, put the top of the ProtoROM box on, and attach the ProtoROM to the ProtoSYSTEM. Write the device select address of the board on a label and place it on the side of the box for future reference.

PROGRAMMING THE ProtoROM

Before programming the ProtoROM, reread Chapter 2 which explains how to write data to the ProtoROM board. The format of the data word in the X register to program the ProtoROM board is (in hexadecimal):

Cp 00 00 00 00 00 ms

"p" tells the ProtoROM the port number and whether to switch the ROM on or off(see Table 4);
"C" activates the ProtoSYSTEM;
"m" tells the ProtoROM which slot of the four to activate (see Table 5);
"s" tells the ProtoROM the device select address of the ProtoROM board to be programmed.

The ProtoROM board will retain its programming until it is reprogrammed or until the ProtoSYSTEM battery is removed.

TABLE 4: "p" VALUES FOR PROGRAMMING THE ProtoROM

"p"	PORT NUMBER TO PLACE ROM	ON/OFF
0	1	OFF
1	1	ON
2	2	OFF
3	2	ON
4	3	OFF
5	3	ON
6	4	OFF
7	4	ON

TABLE 5: "m" VALUES FOR PROGRAMMING THE ProtoROM

ProtoROM			
SLOT 1 "m"=0	SLOT 2 "m"=1	SLOT 3 "m"=2	SLOT 4 "m"=3

EXAMPLE:

You have the ProtoROM with the following Application modules plugged in:

SECURITIES	MATH	GAMES	STATISTICS
SLOT 1	SLOT 2	SLOT 3	SLOT 4

and you have the device select switches set to 0000 (all off).

You wish to use the program BONDS in the SECURITIES module. To place this module ("m" = 0) in port 1, set up the X register as:

C1 00 00 00 00 00 00

by putting the string "C1000000000000" in ALPHA and executing "CODE" or "HN". Then execute SIGN to switch on the SECURITIES module. Run a CAT 2 to verify that the SECURITIES module is now on-line.

Now you want to play CRAPS on the GAMES module, and you want the GAMES module in port 1 instead of the SECURITIES module. To remove the SECURITIES module, set up the X register as:

C0 00 00 00 00 00 00

then execute SIGN. Now set up the X register as:

C1 00 00 00 00 00 20

and execute SIGN to turn on the GAMES module ("m" = 2) in port 1 ("p" = 1). Run a CAT 2 to verify that GAMES is now on-line.

Now you want to use STATISTICS along with GAMES to find out why you are losing at CRAPS. Set up X as:

C3 00 00 00 00 00 30

and execute SIGN to put the STATISTICS module ("m" = 3) into port 2 ("p" = 3). Run a CAT 2 to verify that both GAMES and STATISTICS are on-line.

Now you just acquired an 8K REAL ESTATE module, but all the ports of the ProtoROM are occupied. You plug in a second ProtoROM board and set the device select switches to 0001. Since GAMES is still in port 1 and STATISTICS is in port 2, you decide to put REAL ESTATE in port 3. Set up the X register as:

C5 00 00 00 00 00 21

and execute SIGN. This assumes that you put REAL ESTATE in slot 3 ("m" = 2) of the second ProtoROM board. "s" = 1 to tell the ProtoSYSTEM the device select address of the second ProtoROM board, and "p" = 5 to switch the selected module ON into port 3.

ProtoEPROM PURPOSE

The ProtoEPROM allows the user to plug in standard Hewlett Packard format EPROM sets to the HP 41 and use them just like HP Application modules. EPROM sets may contain user language programs, assembly language (microcode) routines, and/or data tables. EPROMs provide an inexpensive (about half the cost of HP ROMs) means for the user to have his programs available without using HP 41 memory. In addition, microcode routines provide the user with options not normally available. For example, you can recall registers without normalisation, or pack up to 11 alphabetic letters into one data register. Microcode generally executes substantially faster than user language programs. Up to 13 EPROM sets can be plugged in to the calculator at one time. (see Chapter 3)

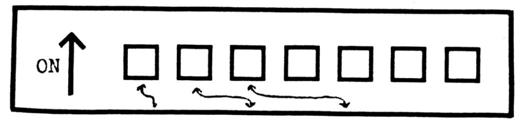
PLUGGING IN EPROMS

The ProtoEPROM board will accept one standard HP-format EPROM set which can contain 2 or 3 EPROM chips. To insert or remove EPROMs, you must remove the Proto-EPROM from the ProtoSYSTEM and remove the top of the ProtoEPROM box. Remove the screws from the base of the box then turn the box so the label is facing up, and lift the cover off. Turn the cranks on the ends of the three large sockets on the circuit board until they point upward. This unlocks the pins of the EPROMs from the sockets. If you already have some EPROMs in the sockets, you may now remove them. At the end of the circuit board that is opposite the 25-pin connector and adjacent to the EPROM sockets are markings on the circuit board that identify which EPROM is to be placed in which socket. The markings are "L2", "L1", and "U", from left to right. These are the same markings that are provided on PPC EPROMs (see Appendix 4). If you have a 2-EPROM set, then socket L2 will remain empty. If you have 24-pin EPROMs, they go in the 28-pin socket with the four extra holes in the sockets nearest to the 25-pin connector. With the ProtoEPROM board oriented so that the 25-pin connector is away from you, the EPROMs are inserted so that pin 1 (identified by a notch in the end of the chip, or a dot at pin 1) is to the upper left. After determining the proper socket and orientation for the EPROMs, set them in the sockets and rotate the crank on each socket until it is horizontal. This locks the EPROMs into the sockets. Now you need to set the address select and specification switches.

SETTING THE ADDRESS SELECT AND SPECIFICATION SWITCHES

Review Chapter 3 to determine which page you want to use as the address for the EPROM. Examining the ProtoEPROM board you will see a small set of seven switches. Orient the ProtoEPROM so that the 25-pin connector is away from you. Then set the right-most four switches to specify the EPROM address (consult Table 1 to convert the hexadecimal address to binary). From left to right set each switch on for 1 or off for 0. On the left side of the block of switches is an arrow which shows the direction to push the switch for on or off. For an 8K EPROM set, the right-most switch is ignored so that the EPROM will be on an 8K boundary occupying pages 4-5, 6-7, 8-9, A-B, C-D, or E-F. For a 16K EPROM set the right-most two switches are ignored so that the EPROM will be on a 16K boundary occupying pages 4-5-6-7, 8-9-A-B, or C-D-E-F.

After setting the address select switches, you need to tell the ProtoEPROM what types and sizes of EPROMs you are using. The remaining three switches do this. The left-most switch tells the ProtoEPROM whether you have 2 or 3 EPROMs in your EPROM set. The next switch tells the ProtoEPROM if your EPROM set is 4K or larger than 4K. The third switch from the left specifies if your EPROM set is 16K or smaller than 16K. Set these switches according to the following diagram:



Set off for:	2 chips	4K	16K
Set on for:	3 chips	8/16K	4/8K

EXAMPLE:

You have the PPC EPROM 2 set which is a 4K EPROM containing various microcode routines written by Jim DeArras. To set up this EPROM set so that it is addressed at page 8, set the switches from left to right to 0011000 (off-off-on-on-off-off-off). If you want this EPROM set in page E, set the switches to 0011110. If you have the 2-chip PPC EPROM 1 which is 8K and you want to address it at pages 8-9 then set the switches to 011100x where x can be either 0 or 1. If you have the 3-chip PPC EPROM 1 set you would set the switches to 111100x.

ProtoCODER PURPOSE

The ProtoCODER allows the user to use the assembly language of the HP 41. This is the internal code interpreted by the microprocessor. By convention this code is referred to as "microcode" although this term is technically incorrect. The ProtoCODER contains 4096 10-bit words of programmable memory. After programming, it appears to the calculator to be an HP Applications or Extension module. Although microcode programming is a relatively new frontier for HP 41 users, the advantages of using it are enormous. It will allow you to write routines to do specific things that cannot be done with regular keyboard programming. For example, it will allow you to pack up to eleven alphabetic characters (A - Z plus any 6 other special characters) into one register as compared to the standard six per register. It will also run substantially (up to 100 times) faster than user language programs, while doing the same thing. The user can write and debug microcode routines in the ProtoCODER then make them more permanent by burning them into an EPROM. The ProtoCODER is easy to program and will retain its programming until reprogrammed or until the battery is removed from the ProtoSYSTEM.

BATTERY AND POWER REQUIREMENTS

The RAM (Random Access Memory) chips in the ProtoCODER will retain their contents as long as they are provided with from 2 to 6 volts. The battery in the ProtoSYSTEM will keep this memory alive when the ProtoSYSTEM is not plugged in to the calculator. When the ProtoSYSTEM is plugged in to the calculator, it runs from the calculator's battery. If the BAT indicator is on, you will not be able to program the ProtoCODER until you recharge or replace the battery, however, the ProtoCODER will not lose its contents unless the voltage drops below 2 volts.

INITIAL SETUP

When you first plug in your ProtoCODER to the ProtoSYSTEM, you should have the address select switches set to 3 hexadecimal. This is necessary since the RAM chips can contain garbage (anything) upon power-up. Page 3 is not currently used by the HP 41 and is completely ignored by the operating system. If you set it in a higher page and run a CAT 2 or if the contents of words FF4 - FFA in the ProtoCODER contain anything other than zeroes, you will crash the calculator. To recover from this crash, unplug the ProtoSYSTEM from the HP 41, set the address select switches on the ProtoCODER to 3 hex, then remove the batteries from the calculator for a few seconds and replace them. An alternative recovery which will usually work is to leave the ProtoSYSTEM plugged in to the HP 41, set the address select switches to 3, wait for a few seconds, then toggle the "ON" switch off and on until you get a regular display. You should leave the address select switches set to 3 until you have a valid ROM image in the ProtoCODER. It is not necessary to clear any words that you do not use as long as the catalog table at the beginning of the ROM image is correct. When experimenting with microcode, it is a good idea to reserve the first 64 words to contain catalog information and start your programs at the 65th word. This will allow you to have up to 31 functions in the ProtoCODER without having to move blocks of memory around when you want to add another program.

SETTING THE ADDRESS SELECT AND DEVICE SELECT SWITCHES

Each ProtoCODER board has two sets of four switches accessable from the top of the ProtoCODER box so that the ProtoSYSTEM can tell the boards apart when more than one is connected, and so that the calculator knows where the ProtoCODER is located in the ROM address space(see Chapter 3). To set the device select and address select switches, orient the ProtoCODER sothat the 25-pin bus connector is away from you, then identify the switches according to the ProtoCODER label. Determine an unused device select address (one that none of the other ProtoSYSTEM devices currently use) which will be a hexadecimal number from 0 to F. Convert this to binary using table 1. Set the switches from left to right for each binary digit (0 or 1). Set the switch on for 1 or off for 0. On the left side of the block of switches is an arrow that shows the direction to push the switch for on or off.. Then determine which page you want the calculator to address the ProtoCODER in using data from Chapter 3. Convert this to binary then set the corresponding address select switches.

PROGRAMMING THE ProtoCODER

The first step in programming the ProtoCODER is to have a list of code to be entered or a program such as in the examples later in this Appendix. Unless you are using the interrupt-jump locations, they must all be set to zero. These are the addresses from FF4 to FFA in the ProtoCODER. If you use a program such as BOOT (listed below) you can load the ProtoCODER in blocks of words, otherwise each word to be programmed into the ProtoCODER is loaded separately. Before continuing, read Chapter 2 which tells you how to write data to the ProtoCODER board. The format of the data word (contained in the X register) to write data to the ProtoCODER is:

DD D0 00 00 AA A0 0S

To find DDD, add hex C00 to the 10 bits of data to be written to the ProtoCODER. AAA contains the hex address (000 - FFF) where the data will be written. S is the device select address. To write this data to the ProtoCODER put the string of 14 hex characters in ALPHA, then execute CODE or HN then execute SIGN. Full instructions on how to do this are in Chapter 2.

EXAMPLE:

You have written and entered into the ProtoCODER a routine called CLY which clears the Y register to zero. You execute it and it crashes the calculator. Looking back through the listing you find:

ADR	DATA	INSTRUCTION
C100	04E	C=0 ALL
C101	270	RAM SLCT
C102	0A8	WRIT Y

followed by some unknown code. Notice that the last three digits of the ADR specify the location in the ProtoCODER and the "C" specifies the page number that is set on the ProtoCODER switches. Examining the listing you notice that you forgot to end the routine with a RTN (3E0) to get back to the operating system after completing clearing Y. To make this routine work, you need to add 3E0 into the ProtoCODER at address C103. Assuming that you have the device select switches set to 2 hex in the ProtoCODER, set up X as:

FE 00 00 00 10 30 02

by entering "FE000000103002" into ALPHA, then execute CODE. The FE0 portion contains the instruction and the two bits to signal the ProtoSYSTEM to program a board. The 103 specifies the address within the ProtoCODER, and the 2 specifies the device select. After executing CODE then execute SIGN to write it to the ProtoCODER. Now try executing CLY again. It should work this time.

USING THE ProtoCODER AFTER PROGRAMMING

After programming a valid ROM image into the ProtoCODER, it will function transparent to the ProtoSYSTEM and without user intervention. It will appear to be an HP Applications or Extension module to the calculator. However, if the ROM image is not correct, crashes or unpredictable results may occur. To have a correct ROM image, the catalog linkage at the beginning of the ROM (starting at address 000) must be set up correctly, and the interrupt-jump words (FF4 - FFA) must either be zero or be used carefully. For proper ROM image setup, see page 16

MICROCODE INSTRUCTIONS

The HP 41's brain (microprocessor) defines what can and cannot be done with the calculator by having a specific set of instructions. These instructions are commonly referred to as "microcode". They are stored in ROMs (Read Only Memories) or anything that looks like a ROM to the calculator (such as a ProtoCODER or EPROM). The sequence of these instructions determines what the calculator will actually do. It gives the calculator its personality that makes it act like an HP 41. The calculator will function just as well with some other operating system or language and could be changed to a completely different personality just by changing these ROMs. By using a disassembler program (such as DISASM, listed on page 15 you can list the contents of pre-existing ROMs to get a general idea of how things are done in microcode.

The processor of the HP 41 has several internal registers that it does operations on and with. Registers A, B, and C (a different register from the c register in the system stack) are 56 bits long -- the same size as one user data register. These registers are used for arithmetic and input/output (I/O) operations. Registers M and N are also 56 bits long but can only be used to store or recall data. The G register is 8 bits long and can be used to store and recall data. Registers P and Q are 4-bit long registers used as digit pointers. They specify the field in A, B, or C where a specific operation is to be performed. Normally only one of these pointers is active at one time. The selected pointer is call PT. The PC register contains the program counter which specifies the address of the next ROM word to be executed. PC is normally increased by one each time an instruction is executed, but can be modified by a GOTO, GOSUB or jump instruction. PC is 16 bits long which means that 2^{16} (65536) addresses are available. Registers ST, F (or T) and KB are each 8 bits long. ST contains <u>system</u> flags 7 - 0. The other <u>system</u> flags (13 - 8) are only accessable individually. These 14 flags are different from the user flags (55 - 0). Flags 13 - 10 are generally reserved for system use. Flag 13 is set if a program is running. Flag 12 is set to indicate a PRIVATE program. Flag 11 is set to enable a stack lift at the end of an instruction. Flag 10 is set to indicate that the program pointer in user stack register b is a ROM address. The HP 41 also has a C flag (Carry or Condition) which is set when a test is true or when a carry occurs. The C flag is cleared one instruction after being set. It is also set when the calculator is first turned on.

The T register (or F register) is used by the HP 41 to control the bender which makes the beeps. The KB register is used to contain a keycode when a key is pressed.

All 56 bit registers are separated into several fields. Depending on the operation, the user can select the field that is affected. The part of the register outside that field is not affected. Named fields are:

```
NYBBLE(DIGIT)  13 12 11 10  9  8  7  6  5  4  3  2  1  0
               -S --------------M-------------- XS  -EXP
                                  ---ADR----- ---S&X--
                                        -KB-
```

There are four classes of microcode instructions: 0, 1, 2, 3. The class is determined by the right-most two bits of the 10-bit instruction. The following tables list both the Hewlett Packard mnemonics for microcode instructions and the mnemonics first published by Steve Jacobs (PPC #5358) in <u>PPC ROM LISTINGS 2</u>. The HP mnemonic is listed first followed in parenthesis by Steve Jacobs' mnemonic.

CLASS 0 INSTRUCTIONS

Class 0 instructions can be separated into two types: Parameter and Special instructions. Table 6 lists the mnemonics for Parameter instructions. Table 7 lists the hex instruction code. Table 8 lists the mnemonics and hex instruction codes for Class 0 Special Instructions. Some of the instructions in these tables that are listed as NOP or UNUSED are used by the HP-IL. HP also has several mnemonics for the same instruction sometimes. For example, RABCR creates the same hex code as C=REGN 14 but is used to read from the display. HP also has some "macros" which are mnemonics that are replaced by one or more instructions. For example, "C=A" is not a normal instruction. It is a macro which is replaced by the sequence "AC EX and A=C".

TABLE 6: CLASS 0 PARAMETER INSTRUCTION MNEMONICS

MNEMONIC	OPERATION
NOP (NOP)	NO OPERATION
Sp=0 (CLRF p)	CLEAR SYSTEM FLAG p
Sp=1 (SETF p)	SET SYSTEM FLAG p
?Sp=1 (?FSET p)	SET C FLAG IF SYSTEM FLAG p SET
LC p (LD@R- p)	LOAD p INTO C AT PT AND DECREMENT PT
?PT=p (?R=p)	SET C FLAG IF POINTER EQUALS p
PT=p (R=p)	SET POINTER TO p
(SELP p)	TRANSFER CONTROL TO A PERIPHERAL
REGN=C p (WRIT p)	WRITE C TO RAM OR DEVICE
?Fp=1 (?FI p)	SET C FLAG IF PERIPHERAL FLAG SET
C=REGN p (READ p)	READ C FROM RAM OR DEVICE
RCR p (RCR p)	ROTATE C RIGHT BY p DIGITS

TABLE 7: CLASS 0 PARAMETER INSTRUCTION HEX CODES

INSTRUCTION MNEMONIC	p=0 (T)	1 (Z)	2 (Y)	3 (X)	4 (L)	5 (M)	6 (N)	7 (O)
NOP	000	040	080	0C0	100	140	180	1C0
Sp=0	384	304	204	004	044	084	144	284
Sp=1	388	308	208	008	048	088	148	288
?Sp=1	38C	30C	20C	00C	04C	08C	14C	28C
LC p	010	050	090	0D0	110	150	190	1D0
?PT=p	394	314	214	014	054	094	154	294
PT=p	39C	31C	21C	01C	05C	09C	15C	29C
SELP p	3A4	324	224	024	064	0A4	164	2A4
REGN=C p	028	068	0A8	0E8	128	168	1A8	1E8
?Fp=1	3AC	32C	22C	02C	06C	0AC	16C	2AC
C=REGN p	038	078	0B8	0F8	138	178	1B8	1F8
RCR p	3BC	33C	23C	03C	07C	0BC	17C	2BC

INSTRUCTION MNEMONIC	p=8 (P)	9 (Q)	10 (⊢)	11 (a)	12 (b)	13 (c)	14 (d)	15 (e)
NOP	200	240	280	2C0	300	340	380	3C0
Sp=0	104	244	0C4	184	344	2C4	-	-
Sp=1	108	248	0C8	188	348	2C8	-	-
?Sp=1	10C	24C	0CC	18C	34C	2CC	-	-
LC p	210	250	290	2D0	310	350	390	3D0
?PT=p	114	254	0D4	194	354	2D4	-	-
PT=p	11C	25C	0DC	19C	35C	2DC	-	-
SELP p	124	264	0E4	1A4	364	2E4	1E4	3E4
REGN=C p	228	268	2A8	2E8	328	368	3A8	3E8
?Fp=1	12C	26C	0EC	1AC	36C	2EC	1EC	3EC
C=REGN p	238	278	2B8	2F8	338	378	3B8	3F8
RCR p	13C	27C	0FC	1BC	37C	2FC	1FC	3FC

TABLE 8: CLASS 0 SPECIAL INSTRUCTION HEX CODES

HEX	MNEMONIC	OPERATION
x34		UNUSED
x74		UNUSED
xB4		UNUSED
xF4		UNUSED
3C4	CLR ST (ST=0)	CLEARS ST AND FLAGS 0 - 7
3C8	RST KB (CLRKEY)	CLEARS "KEY PRESSED" FLAG
3CC	CHK KB (?KEY)	SET C FLAG IF KEY PRESSED
3D4	DEC PT (R=R-1)	DECREMENT CURRENT POINTER
3DC	INC PT (R=R+1)	INCREMENT CURRENT POINTER
018		UNUSED
058	G=C (G=C @R,+)	COPY DIGITS R, R+1 FROM C TO G
098	C=G (C=G @R,+)	COPY G INTO C DIGITS R, R+1
0D8	CG EX (C<>G @R,+)	EXCHANGE G AND C
118		UNUSED
158	M=C (M=C ALL)	COPY C INTO M
198	C=M (C=M ALL)	COPY M INTO C
1D8	MC EX (C<>M ALL)	EXCHANGE M WITH C
218		UNUSED
258	F=ST (T=ST)	COPY ST INTO F
298	ST=F (ST=T)	COPY F INTO ST
2D8	FST EX (ST<>T)	EXCHANGE F WITH ST
318		UNUSED
358	ST=C (ST=C X)	COPY DIGITS 1, 0 FROM C TO ST
398	C=ST (C=ST X)	COPY ST INTO DIGITS 1, 0 OF C
3D8	CST EX (C<>ST)	EXCHANGE ST WITH C DIGITS 1, 0
020	SPOPND (XQ→GO)	DROP STACK TO CONVERT XQ TO GO
060	POWOFF (POWOFF)	GO TO STANDBY MODE
0A0	SEL P (SLCT P)	SELECT P AS THE ACTIVE POINTER
0E0	SEL Q (SLCT Q)	SELECT Q AS THE ACTIVE POINTER
120	?P=Q (?P=Q)	SET C FLAG IF POINTERS ARE EQUAL
160	?LLD (?LOWBAT)	SET C FLAG IF LOW BATTERY
1A0	CLRABC (A=B=C=0)	CLEAR REGISTERS A, B, C
1E0	GOTOC (GOTO ADR)	COPY C DIGITS 6-3 INTO PC
220	C=KEYS (C=KEY [illegible])	COPY KEY REGISTER INTO C(4:3)
260	SETHEX (SETHEX)	USE HEXADECIMAL ARITHMETIC
2A0	SETDEC (SETDEC)	USE DECIMAL ARITHMETIC
2E0	DISOFF (DSPOFF)	TURN OFF DISPLAY
320	DISTOG (DSPTOG)	TOGGLE DISPLAY
360	RTN C (?C RTN)	IF C SET THEN SUBROUTINE RETURN
3A0	RTN NC (?NC RTN)	IF C CLEAR, DO SUBROUTINE RETURN
3E0	RTN (RTN)	POP STACK INTO PC FOR RETURN
030		UNUSED
070	N=C (N=C ALL)	COPY C INTO N
0B0	C=N (C=N ALL)	COPY N INTO C
0F0	NC EX (C<>N ALL)	EXCHANGE C WITH N
130	LDI (LDI S&X)	LOAD NEXT ROM WORD INTO C(2:0)
170	STK=C (PUSH ADR)	PUSH ADR FROM C ONTO STACK
1B0	C=STK (POP ADR)	POP ADR FROM STACK INTO C
1F0		UNUSED
230	(GOTO KEY)	LOAD KB INTO LOWER 8 BITS OF PC
270	DADD=C (RAM SLCT)	SET RAM ADDRESS TO C(2:0)
2B0		UNUSED
2F0	DATA=C (WRITE DATA)	WRITE C TO RAM OR DEVICE
330	CXISA (FETCH S&X)	LOAD C(2:0) FROM ROM ADR C(6:3)
370	C=C OR A	LOGICAL OR OF C WITH A, BIT BY BIT
3B0	C=C AND A	LOGICAL AND OF C WITH A
3F0	PFAD=C (PRPH SLCT)	SET PERIPHERAL ADDRESS TO C(2:0)

CLASS 1 INSTRUCTIONS

Class 1 instructions are absolute GOTOs and EXECUTEs. They consist of two consecutive ROM words of the format:

A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 0 1

A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 p p

A_{15}-A_0 is the 16-bit address to branch to. The pp field of the second word determines what type of instruction it is. Table 9 shows values for pp.

EXAMPLE: ?NC GO 0232 which jumps to the MEMORY LOST routine is coded as:

0011 0010 01 = 0C9 as first word
0000 0010 10 = 00A as second word

TABLE 9: CLASS 1 INSTRUCTIONS

pp	MNEMONIC	OPERATION
00	GOSUB (?NC XQ)	IF C CLEAR, EXECUTE SUBROUTINE
01	GOSC (?C XQ)	IF C SET, EXECUTE SUBROUTINE
10	GOLNC (?NC GO)	IF C CLEAR, GOTO ROM ADDRESS
11	GOLC (?C GO)	IF C SET, GOTO ROM ADDRESS

TABLE 11: CLASS 2 FIELDS OF OPERATION

FIELD	AREA OF OPERATION
PT (@R)	AT DIGIT SPECIFIED BY CURRENT POINTER
X (S&X)	AT EXPONENT - DIGITS 2, 1, 0
WPT (R←)	UP TO POINTER FROM THE RIGHT
(ALL)	ALL DIGITS. HP USUALLY LEAVES THIS FIELD BLANK
PQ (P-Q)	FROM POINTER P, LEFT UP TO Q OR DIGIT 13
XS (XS)	EXPONENT SIGN - DIGIT 2
M (M)	MANTISSA - DIGITS 12 - 3
S (MS)	MANTISSA SIGN - DIGIT 13

CLASS 2 INSTRUCTIONS

The set of Class 2 microcode instructions perform arithmetic and logical operations. Arithmetic operations are performed in hexadecimal or decimal depending on the last mode operation executed (SETHEX or SETDEC).

TABLE 10: CLASS 2 INSTRUCTION HEX CODES

MNEMONIC	OPERATION	CODE FOR EACH FIELD OF OPERATION							
		PT	X	WPT	ALL	PQ	XS	M	S
A=0 (A=0)	CLEAR A	002	006	00A	00E	012	016	01A	01E
B=0 (B=0)	CLEAR B	022	026	02A	02E	032	036	03A	03E
C=0 (C=0)	CLEAR C	042	046	04A	04E	052	056	05A	05E
AB EX (A<>B)	EXCHANGE A WITH B	062	066	06A	06E	072	076	07A	07E
B=A (B=A)	COPY A INTO B	082	086	08A	08E	092	096	09A	09E
AC EX (A<>C)	EXCHANGE A WITH C	0A2	0A6	0AA	0AE	0B2	0B6	0BA	0BE
C=B (C=B)	COPY B INTO C	0C2	0C6	0CA	0CE	0D2	0D6	0DA	0DE
BC EX (B<>C)	EXCHANGE B WITH C	0E2	0E6	0EA	0EE	0F2	0F6	0FA	0FE
A=C (A=C)	COPY C INTO A	102	106	10A	10E	112	116	11A	11E
A=A+B(A=A+B)	ADD B INTO A	122	126	12A	12E	132	136	13A	13E
A=A+C(A=A+C)	ADD C INTO A	142	146	14A	14E	152	156	15A	15E
A=A+1(A=A+1)	INCREMENT A	162	166	16A	16E	172	176	17A	17E
A=A-B(A=A-B)	SUBTRACT B FROM A	182	186	18A	18E	192	196	19A	19E
A=A-1(A=A-1)	DECREMENT A	1A2	1A6	1AA	1AE	1B2	1B6	1BA	1BE
A=A-C(A=A-C)	SUBTRACT C FROM A	1C2	1C6	1CA	1CE	1D2	1D6	1DA	1DE
C=C+C(C=C+C)	DOUBLE C	1E2	1E6	1EA	1EE	1F2	1F6	1FA	1FE
C=A+C(C=A+C)	ADD A INTO C	202	206	20A	20E	212	216	21A	21E
C=C+1(C=C+1)	INCREMENT C	222	226	22A	22E	232	236	23A	23E
C=A-C(C=A-C)	A-C INTO C	242	246	24A	24E	252	256	25A	25E
C=C-1(C=C-1)	DECREMENT C	262	266	26A	26E	272	276	27A	27E
C=-C (C=0-C)	COMPLEMENT C	282	286	28A	28E	292	296	29A	29E
C=-C-1	NINES COMPLEMENT C	2A2	2A6	2AA	2AE	2B2	2B6	2BA	2BE
?B#0 (?B#0)	SET C FLAG IF B≠0	2C2	2C6	2CA	2CE	2D2	2D6	2DA	2DE
?C#0 (?C#0)	SET C FLAG IF C≠0	2E2	2E6	2EA	2EE	2F2	2F6	2FA	2FE
?A<C (?A<C)	SET C FLAG IF A<C	302	306	30A	30E	312	316	31A	31E
?A<B (?A<B)	SET C FLAG IF A<B	322	326	32A	32E	332	336	33A	33E
?A#0 (?A#0)	SET C FLAG IF A≠0	342	346	34A	34E	352	356	35A	35E
?A#C (?A#C)	SET C FLAG IF A≠C	362	366	36A	36E	372	376	37A	37E
A SR (RSHFA)	SHIFT A RIGHT 1 DIG	382	386	38A	38E	392	396	39A	39E
B SR (RSHFB)	SHIFT B RIGHT 1 DIG	3A2	3A6	3AA	3AE	3B2	3B6	3BA	3BE
C SR (RSHFC)	SHIFT C RIGHT 1 DIG	3C2	3C6	3CA	3CE	3D2	3D6	3DA	3DE
A SL (LSHFA)	SHIFT A LEFT 1 DIG	3E2	3E6	3EA	3EE	3F2	3F6	3FA	3FE

CLASS 3 INSTRUCTIONS

Class 3 instructions allow the program to jump up to 63 words forward or backward from its present location. The mnemonics are GONC or GOTO (JNC) and GOC (JC). In Hewlett Packard listings, the GOTO is followed by a label. In disassembled listings, the GOTO is followed by "*+pp" or "*-pp" which indicates a jump relative to the current instruction address (*).

TABLE 12: CLASS 3 INSTRUCTIONS

DIST ANCE	JNC *-	JC *-	JNC *+	JC *+	DIST ANCE	JNC *-	JC *-	JNC *+	JC *+
*±01	3FB	3FF	00B	00F	*±02	3F3	3F7	013	017
*±03	3EB	3EF	01B	01F	*±04	3E3	3E7	023	027
*±05	3DB	3DF	02B	02F	*±06	3D3	3D7	033	037
*±07	3CB	3CF	03B	03F	*±08	3C3	3C7	043	047
*±09	3BB	3BF	04B	04F	*±0A	3B3	3B7	053	057
*±0B	3AB	3AF	05B	05F	*±0C	3A3	3A7	063	067
*±0D	39B	39F	06B	06F	*±0E	393	397	073	077
*±0F	38B	38F	07B	07F	*±10	383	387	083	087
*±11	37B	37F	08B	08F	*±12	373	377	093	097
*±13	36B	36F	09B	09F	*±14	363	367	0A3	0A7
*±15	35B	35F	0AB	0AF	*±16	353	357	0B3	0B7
*±17	34B	34F	0BB	0BF	*±18	343	347	0C3	0C7
*±19	33B	33F	0CB	0CF	*±1A	333	337	0D3	0D7
*±1B	32B	32F	0DB	0DF	*±1C	323	327	0E3	0E7
*±1D	31B	31F	0EB	0EF	*±1E	313	317	0F3	0F7
*±1F	30B	30F	0FB	0FF	*±20	303	307	103	107
*±21	2FB	2FF	10B	10F	*±22	2F3	2F7	113	117
*±23	2EB	2EF	11B	11F	*±24	2E3	2E7	123	127
*±25	2DB	2DF	12B	12F	*±26	2D3	2D7	133	137
*±27	2CB	2CF	13B	13F	*±28	2C3	2C7	143	147
*±29	2BB	2BF	14B	14F	*±2A	2B3	2B7	153	157
*±2B	2AB	2AF	15B	15F	*±2C	2A3	2A7	163	167
*±2D	29B	29F	16B	16F	*±2E	293	297	173	177
*±2F	28B	28F	17B	17F	*±30	283	287	183	187
*±31	27B	27F	18B	18F	*±32	273	277	193	197
*±33	26B	26F	19B	19F	*±34	263	267	1A3	1A7
*±35	25B	25F	1AB	1AF	*±36	253	257	1B3	1B7
*±37	24B	24F	1BB	1BF	*±38	243	247	1C3	1C7
*±39	23B	23F	1CB	1CF	*±3A	233	237	1D3	1D7
*±3B	22B	22F	1DB	1DF	*±3C	223	227	1E3	1E7
*±3D	21B	21F	1EB	1EF	*±3E	213	217	1F3	1F7
*±3F	20B	20F	1FB	1FF	*±40	203	207	---	---

MICROCODE ADDITIONAL NOTES

SYSTEM STACK

The microprocessor in the HP 41 uses an address stack to keep track of subroutine calls. This stack will hold 4 address entries. Each time an EXECUTE occurs, the address of the second word of the EXECUTE instruction is "pushed" onto the stack -- it becomes the lowest entry and the other entries are moved up by one. If there were already four addresses in the stack, the top one is lost. Whenever a RTN occurs, the bottom entry of the stack is copied into the PC register and all other entries are moved down by one. A zero is moved into the top stack position. When a SPOPND occurs, the stack is dropped by one. When a C=STK occurs, the stack is dropped by one and the address that was in the bottom stack entry is copied into the address field of C (6:3). When a STK=C occurs, the stack is lifted by one, and the address field of c is copied into the bottom stack location.

SYSTEM STATUS

There are three major modes that the HP 41 can be in: sleep, standby, and active. In sleep mode, the calculator is turned off. In standby mode, the calculator is turned on but is not executing any microcode. In active mode, the calculator is running microcode. The system ROMs (pages 0 - 2) contain code to differentiate between sleep and standby by setting the C flag when the ON key is pressed.

USING THE DISPLAY

The display functions independently from the HP 41 processor. It has its own processors. For an in depth coverage of how the display works, consult Paul Lind's article published in the PPC Calculator Journal V9N2P15 and in PPC Technical Notes #10 P21.

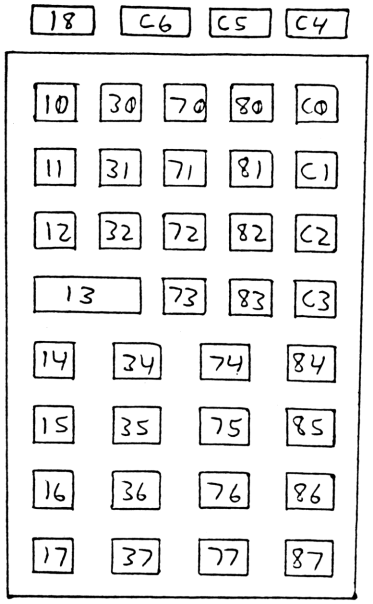
SETHEX VS. SETDEC

Upon execution of either of these operations, it remains in effect until another of these operations is performed. In DEC mode, all arithmetic operations are performed on digits from 0 to 9. In HEX mode, all arithmetic operations are performed on digits from 0 to F. The C flag is set if the operation performed causes the result to exceed 9 (DEC) or F(HEX) or if the result is less than 0. This is how HP does their normalisation: In DEC mode, add 1 (which sets the C flag and converts the result to decimal) then subtract 1 to return the original result. If the original word was already normalised, it is not affected by adding and subtracting 1.

RELOCATION

When you write a microcode routine to be contained in an external ROM (or EPROM or ProtoCODER), you cannot use Class 1 GOTOs or EXECUTEs if you want the contents to be able to work when plugged in any port or addressed to any page. However, there are several routines in the operating system pages 0 - 2 to allow you to write absolute GOTOs and EXECUTEs.

KEYCODES RETURNED BY "C=KEYS" IN MICROCODE



The above hexadecimal keycodes are returned in C(4:3) when C=KEYS is executed. If no key was pressed, 00 is returned. All other digits of C are unaffected.

HP 41 TONES IN MICROCODE

The HP 41 uses a short microcode routine located at address 16DD to control the bender for all TONE operations. Both the frequency and the duration of the tone are software controlled and are predictable given the cycle time of the calculator. The system routine accepts 2 digits of data to specify the tone. The system chops off the left-most bit and interprets it to mean INDIRECT if it is 1. TONE instructions appear in memory as 9F ab where a is normally 0 and b is 0 - 9 unless created syntheticly. The duration of the tone is determined by the contents of ROM word 16F2 + ab. This value is decremented in a loop as the tone is being heard until it reaches zero, which terminates the tone. The frequency is determined by b. Hewlett Packard intended only ten tones to be used, but the TONE routine will look up ROM data for all 128 tones. This explains why some of the synthetic tones changed in duration when HP came out with a ROM update.

To use the bender, store 00 in the F (or T) register and store FF in the ST register. Tones are created by turning F on and off, that is, by swapping F and ST. The number of swaps defines the length of duration of the tone. The number of instructions between swaps defines the frequency of the tone. The duration and frequency will also vary depending on the cycle time of your calculator. Non-speeded-up calculators generally have a cycle time of 155 - 158 microseconds per microcode instruction.

EXAMPLE:

TONE 9 (9F 09) has a period of 6 processor cycles. Given the cycle time of 158 microseconds, the period would be .000158 * 6 = .000948 seconds. Then the frequency is 1 / .000948 = 1055 hertz. To determine the duration, convert the ROM data word at 16F2 + 09 = 16FB which is 215 hex, to decimal, then add one since the looper decrements this number until it is LESS than zero. This number (= 534 decimal) is the number of times that the bender is flopped using the F register. The duration of TONE 9 is 534 loops * 3 cycles per loop * .000158 seconds per cycle = .253 seconds.

SAMPLE MICROCODE ROUTINES

The following routines provide examples of using microcode on the HP 41. Each routine includes the name which is coded backwards from the first word to be executed. The catalog table at the beginning of each ROM has a list of addresses of the routines. The address in the catalog table points to the first word of executable code for microcode routines.

"X=1?"

This routine sets up a 1 in the C register then branches to the system ROM routine that makes the comparison.

```
EE00 06F ? JC      *+17
EE01 031 1
EE02 03D = ?C  XQ 0F0C
EE03 018 X UNUSED
EE04 04E Σ C=0     ALL
EE05 270 ⊢ RAM     SLCT
EE06 35C ν R=      C<12>
EE07 050 ⊢ LD@R-   1
EE08 055 e
EE09 05A ß ?NC GO 1615
```

"+1"

This routine is a good example of the speed of microcode compared to user code. When you execute "+1", the calculator starts counting from zero, incrementing by 1 each loop until any key is pressed. It returns the resulting total in X.

```
E170 0B1 1
E171 02B + ?C  GO 0A2C
E172 04E Σ C=0     ALL
E173 2A0   SETDEC
E174 270 ⊢ RAM     SLCT
E175 23A : C=C+1   M
E176 3CC ν ?KEY
E177 3F3 c JNC     *-02
E178 130 0 LDI     S&X
E179 009 I =D9
E17A 10E N A=C     ALL
E17B 35C ν R=      C<12>
E17C 1A6 & A=A-1   S&X
E17D 3FA ß LSHFA   M
E17E 342 b ?A≠0    @R
E17F 3EB ⁎ JNC     *-03
E180 0AE . A<>C    ALL
E181 0E8 Γ WRIT    3<X>
E182 3C8 Γ CLRKEY
E183 3CC ν ?KEY
E184 3F7 ' JC      *-02
E185 3E0 ⊢ RTN
```

"GOOSE"

This routine appends a left-facing goose to the display. For a demonstration of something that you can not do with user code, enter GOOSE as a program line in a program that is displaying a goose when running.

```
E0C1 085 E
E0C2 013 S ?C  GO 0421
E0C3 00F O JC      *+01
E0C4 00F O JC      *+01
E0C5 007 G JC      *+00
E0C6 3C1 a
E0C7 0B0 0 ?NC XQ 2CF0
E0C8 130 0 LDI     S&X
E0C9 02C , =D44
E0CA 3A8 ( WRIT    14<d>
E0CB 14D ≠
E0CC 024 $ ?NC XQ 0953
E0CD 3E0 ⊢ RTN
```

"CODE"

This routine encodes up to 14 characters from ALPHA into X. The 14 characters in ALPHA are character representations of the hex code to be entered into X. Each character can be 0 - F. If less than 14 characters are to be encoded into X, leading zeroes are assumed.

```
E14B 085 E
E14C 004 D ?NC XQ 0121
E14D 00F O JC      *+01
E14E 003 C JNC     *+00
E14F 04E Σ C=0     ALL
E150 270 ⊢ RAM     SLCT
E151 260 ⊢ SETHEX
E152 248 Γ SETF    9
E153 10E N A=C     ALL
E154 1B8 8 READ    6<N>
E155 0EE Σ C<>B    ALL
E156 130 0 LDI     S&X
E157 006 F =D6
E158 0EE Σ C<>B    ALL
E159 37C ν RCR     C<12>
E15A 3EE Σ LSHFA   ALL
E15B 358 Γ ST=C    X
E15C 39C \ R=      0
E15D 102 B A=C     @R
E15E 250 ⊢ LD@R-   9
E15F 39C \ R=      0
E160 14C ν ?FSET   6
E161 013 S JNC     *+02
E162 142 b A=A+C   @R
E163 0EE Σ C<>B    ALL
E164 266 ♦ C=C-1   S&X
E165 01F _ JC      *+03
E166 0EE Σ C<>B    ALL
E167 393 S JNC     *-0E
E168 24C ν ?FSET   9
E169 023 # JNC     *+04
E16A 244 d CLRF    9
E16B 178 Γ READ    5<M>
E16C 34B ⁎ JNC     *-17
E16D 0AE . A<>C    ALL
E16E 0E8 Γ WRIT    3<X>
E16F 3E0 ⊢ RTN
```

"LOAD"

This is a user language program to assist the user in loading ROM data into the ProtoCODER. To use, execute LOAD and enter the starting ProtoCODER address to be loaded into ALPHA at the prompt, then press R/S. Then at each successive prompt, enter the hex characters to be entered in the next word. After each entry, LOAD will recall what it just loaded and display it to verify that it is correct. LOAD uses the Extended Functions module and PPC EPROM 2.

```
01♦LBL "LOAD"
02 "1000"
03 CODE
04 STO 01
05 "START"
06 AON
07 PROMPT
08 "├000"
09 CODE
10 STO 00
11 "DATA"
12♦LBL 01
13 AON
14 PROMPT
15 AOFF
16 ATOX
17 19
18 +
19 XTOA
20 -1
21 AROT
22 "├0000000000"
23 CODE
24 0
25 NRCL
26 X+Y
27 SIGN
28 0
29 NRCL
30 RXR
31 RXR
32 RXR
33 X<>ROM
34 DECODE
35 ASHF
36 0
37 NRCL
38 1
39 NRCL
40 X+Y
41 STO 00
42 GTO 01
43 END
```

"DISASM"

This routine is used to list the contents of a ROM. To use it, put the ROM address in X using CODE or "HN" in the PPC ROM. Then execute DISASM. The ROM code is returned in ALPHA in the format "AAAA CCC R " where AAAA is the ROM address, CCC is the word of ROM code, and R is the character representation of the ROM code. X is also increased by 1, and the result of the CXISA is returned in Y.

```
E0CE 08D M
E0CF 013 S ?C  GO 0423
E0D0 001 A
E0D1 013 S ?C  GO 0400
E0D2 009 I
E0D3 004 D ?NC XQ 0102
E0D4 260 ├ SETHEX
E0D5 345 e
E0D6 040 ├ ?NC XQ 10D1
E0D7 130 0 LDI    S&X
E0D8 006 F =D6
E0D9 1BC < RCR    B<11>
E0DA 130 0 LDI    S&X
E0DB 009 I =D9
E0DC 10E N A=C    ALL
E0DD 0F8 Γ READ   3<X>
E0DE 22E . C=C+1  ALL
E0DF 0E8 Γ WRIT   3<X>
E0E0 26E Σ C=C-1  ALL
E0E1 1BC < RCR    B<11>
E0E2 330 0 FETCH  S&X
E0E3 0A8 ( WRIT   2<Y>
E0E4 2BC < RCR    7
E0E5 130 0 LDI    S&X
E0E6 003 C =D3
E0E7 2FC µ RCR    D<13>
E0E8 39C \ R=     0
E0E9 302 B ?A<C   @R
E0EA 02B + JNC    *+05
E0EB 242 b C=A-C  @R
E0EC 282 B C=0-C  @R
E0ED 31C \ R=     1
E0EE 222 " C=C+1  @R
E0EF 0EE Σ C<>B   ALL
E0F0 178 Γ READ   5<M>
E0F1 37C µ RCR    C<12>
E0F2 0E6 ♦ C<>B   S&X
E0F3 0F6 ♦ C<>B   XS
E0F4 168 Γ WRIT   5<M>
E0F5 0EE Σ C<>B   ALL
E0F6 18A : A=A-1  M
E0F7 373 c JNC    *-12
E0F8 178 Γ READ   5<M>
E0F9 10E N A=C    ALL
E0FA 09C \ R=     5
E0FB 04A ß C=0    R<-
E0FC 17C µ RCR    6
E0FD 1A8 ( WRIT   6<N>
E0FE 0AE . A<>C   ALL
E0FF 13C < RCR    8
E100 090 P LD@R-  2
E101 010 P LD@R-  0
E102 090 P LD@R-  2
E103 010 P LD@R-  0
E104 090 P LD@R-  2
E105 010 P LD@R-  0
E106 2DC µ R=     D<13>
E107 090 P LD@R-  2
E108 010 P LD@R-  0
E109 168 Γ WRIT   5<M>
E10A 0B8 8 READ   2<Y>
E10B 358 Γ ST=C   X
E10C 284 D CLRF   7
E10D 398 X C=ST   X
E10E 14C µ ?FSET  6
E10F 043 c JNC    *+08
E110 01C \ R=     3
E111 090 P LD@R-  2
E112 310 P LD@R-  C
E113 010 P LD@R-  0
E114 1BC < RCR    B<11>
E115 330 0 FETCH  S&X
E116 02B + JNC    *+05
E117 08C L ?FSET  5
E118 01F _ JC     *+03
E119 148 Γ SETF   6
E11A 398 X C=ST   X
E11B 10E N A=C    ALL
E11C 178 Γ READ   5<M>
E11D 23C < RCR    2
E11E 0A6 & A<>C   S&X
E11F 0B6 6 A<>C   XS
E120 37C µ RCR    C<12>
E121 168 Γ WRIT   5<M>
E122 2CC µ ?FSET  D<13>
E123 360 ├ ?C     RTN
E124 2C9 α
E125 042 b ?NC GO 10B2
```

XROM WORD FORMAT

Each 4K ROM contains mostly routines and programs, however, several words are used by the system to keep track of where these routines are located, and when to execute them. The first block of words in the ROM contains the XROM number, the number of routines, and a catalog linkage table. Near the end of the ROM there is a table containing interrupt-jump points, the ROM name, and a checksum. XROM contents are listed in Table 13.

TABLE 13: XROM CONTENTS

ADDRESS	CONTENTS
x000	Hex XROM number, i. e., the Printer (29) is 01D.
x001	Hex number of routines. For 82143 printer with 25 routines, this is 019. This includes the XROM label (-PRINTER-). XROMs do not need a name but if you provide one, it should be followed by a RTN in case it is executed.
nnn-nnn+1	Starting with words 002-003, these pairs contain the address of the routine. For microcode routines this address points to the first executable word. The first word is 00a. The second is 0bc. This points to address xabc. For user language programs, the first word is 20a. xabc points to the first word of the LBL.
kkk-kkk+1	Following n+1 pairs of words for an XROM with n functions, are two words containing 000. This signals the system that the end of the catalog table has been reached.
FF4-FFA	These words contain a single instruction to be executed at designated interrupt times. In most cases these will be zero. At the specific times that these interrupts are generated, the system looks through all XROMs to find any non-zero values. If a non-zero value is found, then the instruction (usually a GOTO) found there is executed.
FF4	Interrupts during PSE loop.
FF5	Interrupts after each program line.
FF6	Wakeup with no key down.
FF7	Interrupts when turned off.
FF8	Interrupts when peripheral flag is set.
FF9	Wakeup with ON key.
FFA	Interrupts after MEMORY LOST.
FFB-FFE	Contains the ROM name used by the HP Service module. This can contain anything.
FFF	ROM checksum. This is used to verify that the contents of the ROM are correct. The HP Service module adds all ROM words together and compares the result to the checksum. If they match, then the ROM is declared to be "OK".

APPENDIX 4: PPC INFORMATION

PPC is the Personal Programming Center which is an organization of users dedicated to personal computing. It is the oldest personal computing group in the world. PPC publishes the PPC Calculator Journal which disemenates information and programs for HP calculators. For information on membership, obtaining back issues of the PPCCJ, and information about the PPC ROM or PPC EPROMs, send a 9x12" envelope with 2oz of postage or equivalent international postal coupons to:

PPC
2545 W. Camden Place
Santa Ana, Ca. 92704

PPC Technical Notes is a publication of the Melbourne Chapter of the PPC. For subscription information, send a self addressed envelope and international postal coupons to:

PPCTN
R.M. Eades
P. O. Box 15
Hampton, Victoria 3188
Australia

For information on ordering PPC EPROMs, send a SASE to:

Logical Systems Associates
P. O. Box 1023
Garden Grove, Ca. 92642

or consult PPCCJ V9N1P27.

PROTOSYSTEM OWNERS MANUAL ADDENDUM

PROTOTECH, INC.
P. O. BOX 12104
BOULDER, CO. 80303

T A B L E O F C O N T E N T S

SECTION 1: CORRECTIONS TO MANUAL

Page 8, line 6 in "Programming the ProtoCODER" - BOOT is not listed anywhere in the manual, however, it is listed in this addendum.
Page 11, lines 7 and 8 from the bottom - the < signs were left out twice in each line.
Page 12, in Table 12 - the hex code for JC *-25 should be 2DF, not 2BF.

SECTION 2: HP AND PPC MNEMONICS

In the tables of microcode mnemonics, both the standard HP mnemonics and the PPC adopted mnemonics (of Steve Jacobs) are listed. The HP mnemonic is listed first followed by the PPC mnemonic in parenthesis.

SECTION 3: ABOUT WRITING DATA TO THE ProtoSYSTEM

There is no need to erase a particular location in the ProtoCODER. When you write new data to that location, it writes over (replaces) whatever was there.

As suggested by David Spear, it is easier and faster to use an F (or A - E) instead of 0 for digits CODEd in X that are ignored by the device being programmed. For example, to write a C=0 instruction (0AE) to location 236 in the ProtoCODER (with the device select switches set to 0), both of the following will perform exactly the same:

```
     CA E0 00 00 23 60 00
  or CA #F FF FF 23 6F F0
```

but the second one is much faster to enter in ALPHA.

SECTION 4: APPLICATIONS PROGRAMS

This section contains two programs to assist in copying ROM contents to mass storage (eg tape) and to load ROM data from mass storage or data registers.

BOOT is used to sequentially load the ProtoCODER with 5 ROM words contained in sequential user registers. The user registers can be loaded from the cassette tape or any other source. It loads words until it has loaded the contents of absolute register at address 1FF, then stops. It requires a HP 41CV or C with Quad RAM. Boot must be run as a program line, e.g.:

```
LBL "AA"
BOOT
STOP
```

To use, set up the contents of the registers to contain the 50 bits each of 5 ROM words, right justified, with 000100 as the leftmost 6 bits. For example, to clear the interrupt jump locations (FF4-FFA), SIZE to 003 then store 0 in registers 1 and 2. You must load the highest numbered registers with data since BOOT starts at the register address you input and continues running until it reaches the highest register number. Before running the above program, you need to set up the stack as:

- Z contains the CODEd absolute address (000-FFF) of the first ProtoCODER location to load
- Y contains 0
- T and X do not matter
- L contains the CODEd absolute address of the register immediately preceding the one to start loading data from

For the above example,

```
"FF4" CODE ENTER 0 ENTER "1FD" CODE STO .L
```

then run the above program.
Here "FF4" is the first address to load (the first interrupt jump location)

and "1FD" is the absolute address of register 0 which immediately precedes register 1 containing the first data to be loaded.
BOOT is listed on the next page.

DUMP can be used to save microcode routines on tape (etc) or to copy data from a ROM to subsequently load into the ProtoCODER. DUMP takes 5 ROM words starting with the absolute address CODEd into X, and packs them into one register as an ALPHA string and stores this at the absolute register address CODEd in Y. It also increments Y and adds 5 to X. For example, to copy the interrupt jump locations from the ROM in port 1 (addresses 8FF4 - 8FFD) into registers 1 and 2 (to be loaded with BOOT as above), SIZE to 003, then:

"1FE" CODE ENTER "8FF4" CODE DUMP DUMP

DUMP is listed on the next page.

SECTION 5: EPROM AVAILABILITY

Logical Systems Associates will not be accepting any more orders for EPROM copies. However, Joe Bell will have EPROM copies available. For more information and pricing, write to:

JOE BELL
SURVEY CALCULATIONS JOURNAL
P O BOX 6674
SAN BERNARDINO, CA 92412

SECTION 6: NFCROM

If you have an EPROM box or ProtoEPROM, then NFCROM may be very useful to you. It contains routines to load the ProtoCODER quickly, and to disassemble (with mnemonics) any ROM. Copies of NFCROM with routine descriptions can be obtained from:

Prototech, Inc.
P. O. Box 12104
Boulder, Co. 80303

The price is $35 per NFCROM set which includes the routine descriptions and shipping. Following is a list of the routines in NFCROM:

NFCROM-1A displays message	BOOT as above
DUMP as above	LEFT rotates display to left
, appends left goose to display	DISASM microcode disassembler
CL clears system flag 12	RCLA non-normalised recall
CODE hex to non-normalised number	DECODE non-normalised number to hex
+1 microcode speed comparator	AK hex code key assignments
. appends right goose to display	DIS appends any char to display
ROM? displays ROM 0,1,2 revisions	DISTST display test
LOAD loads a word into ProtoCODER	X=1? comparison
PROMT usiversal hexadecimal prompting	LODE loads sequential ProtoCODER words
CAT lists on-line ROMs and other things	POW2 extended precision powers of 2
LODB hex byte loader (unfinished)	MNEM mnemonics for disassembler
MANT returns mantissa of X to X	ROMSUM computes ROM checksum
M10INT returns INT(X) MOD 10 to X	STA non-normalised store
BJUMP byte jumper	TOGF toggles user flag
DEC-HEX 12 digits decimal to hex	HEX-DEC 8 digits hex to decimal
<> exchanges IND X with IND Y	INIT initializes ProtoCODER with CAT

```
EC44 094 T
EC45 00F O
EC46 00F O
EC47 002 B
EC48 260 ⊢ SETHEX
EC49 04E Σ C=0    ALL
EC4A 270 ⊢ DADD=C
EC4B 0B8 8 C=REGN 2(Y)
EC4C 2E6 ♦ ?C#0   X
EC4D 0AF / GOC    *+15
EC4E 138 8 C=REGN 4(L)
EC4F 226 & C=C+1  X
EC50 128 ( REGN=C 4(L)
EC51 33C < RCR    1
EC52 358 Γ ST=C
EC53 08C L ?S=1   5
EC54 360 ⊢ RTN C
EC55 000 @ NOP    3
EC56 130 0 LDI
EC57 004 D CON    0004
EC58 0A8 ( REGN=C 2(Y)
EC59 138 8 C=REGN 4(L)
EC5A 270 ⊢ DADD=C
EC5B 038 8 C=REGN 0(T)
EC5C 10E N A=C    ALL
EC5D 04E Σ C=0    ALL
EC5E 270 ⊢ DADD=C
EC5F 0AE . AC EX  ALL
EC60 028 ( REGN=C 0(T)
EC61 033 3 GONC   *+06
EC62 266 ♦ C=C-1  X
EC63 0A8 ( REGN=C 2(Y)
EC64 04E Σ C=0    ALL
EC65 270 ⊢ DADD=C
EC66 038 8 C=REGN 0(T)
EC67 1EE Σ C=C+C  ALL
EC68 1EE Σ C=C+C  ALL
EC69 2DC μ PT=    13
EC6A 0D0 ⊢ LC     3
EC6B 37C μ RCR    12
EC6C 028 ( REGN=C 0(T)
EC6D 23C < RCR    2
EC6E 1EE Σ C=C+C  ALL
EC6F 1EE Σ C=C+C  ALL
EC70 1BC < RCR    11
EC71 05A β C=0    M
EC72 05E Σ C=0    S
EC73 17C μ RCR    6
EC74 10E N A=C    ALL
EC75 078 Γ C=REGN 1(Z)
EC76 106 F A=C    X
EC77 226 & C=C+1  X
EC78 068 Γ REGN=C 1(Z)
EC79 138 8 C=REGN 4(L)
EC7A 0E8 Γ REGN=C 3(X)
EC7B 06E Σ AB EX  ALL
EC7C 338 8 C=REGN 12(b)
EC7D 10E N A=C    ALL
EC7E 35D ≠
EC7F 000 @ GOSUB  00D7
EC80 03C < RCR    3
EC81 130 0 LDI
EC82 3CE Σ CON    0974
EC83 1BC < RCR    11
EC84 01C \ PT=    3
EC85 1E0 ⊢ GOTOC
EC86 000 @ NOP    3

E3CE 150 ⊢ LC     5
E3CF 01C \ PT=    3
E3D0 0CC μ ?S=1   10
E3D1 023 # GONC   *+04
E3D2 1A6 & A=A-1  X
E3D3 1A6 & A=A-1  X
E3D4 043 c GONC   *+08
E3D5 302 B ?A<C   PT
E3D6 027 ' GOC    *+04
E3D7 1C2 b A=A-C  PT
E3D8 166 ♦ A=A+1  X
E3D9 01B [ GONC   *+03
E3DA 162 b A=A+1  PT
E3DB 162 b A=A+1  PT
E3DC 0AE . AC EX  ALL
E3DD 328 ( REGN=C 12(b)
E3DE 0EE Σ BC EX  ALL
E3DF 1BC < RCR    11
E3E0 35C μ PT=    12
E3E1 3D9 α
E3E2 03E > GOLONG 0FF6

E088 090 P
E089 00D M
E08A 015 U
E08B 004 D
E08C 04E Σ C=0    ALL
E08D 260 ⊢ SETHEX
E08E 270 ⊢ DADD=C
E08F 070 ⊢ N=C
E090 0F8 Γ C=REGN 3(X)
E091 1BC < RCR    11
E092 2DC μ PT=    13
E093 110 P LC     4
E094 330 0 CXISA
E095 10E N A=C    ALL
E096 0B0 0 C=N
E097 1BC < RCR    11
E098 0A6 & AC EX  X
E099 1E6 ♦ C=C+C  X
E09A 1E6 ♦ C=C+C  X
E09B 1EE Σ C=C+C  ALL
E09C 1EE Σ C=C+C  ALL
E09D 33C < RCR    1
E09E 070 ⊢ N=C
E09F 0AE . AC EX  ALL
E0A0 23A : C=C+1  M
E0A1 27E Σ C=C-1  S
E0A2 38B K GONC   *-0F
E0A3 05E Σ C=0    S
E0A4 046 ♦ C=0    X
E0A5 03C < RCR    3
E0A6 0E8 Γ REGN=C 3(X)
E0A7 0B8 8 C=REGN 2(Y)
E0A8 22E . C=C+1  ALL
E0A9 0A8 ( REGN=C 2(Y)
E0AA 26E Σ C=C-1  ALL
E0AB 270 ⊢ DADD=C
E0AC 0B0 0 C=N
E0AD 2DC μ PT=    13
E0AE 050 ⊢ LC     1
E0AF 2F0 ⊢ DATA=C
E0B0 3E0 ⊢ RTN
```

SECTION 7: ProtoROM SPECIFICATION CHANGES

The ProtoROM specification now reads "Each ProtoROM board lets the user plug in up to 4 HP Application Modules (ROMs). Each module can be individually switched on or off of the HP 41C/V bus under keyboard or program control."

This should be changed in your manual at the following locations:
Page 1 under ProtoSYSTEM Overview, lines 12-14
Page 2 replace the Effects of Master Clear paragraph with:
"No ProtoSYSTEM boards are affected by MASTER CLEAR."
Page 4 lines 7-9
Page 6 delete lines 11-15 under ProtoROM Purpose starting with "The ProtoROM can also . . ."

SECTION 8: ROM CHARACTER TABLE, FUNCTION NAMES, PROMPTING, NON-PROGRAMMABILITY

The HP41 recognises 2 distinct character sets: modified ASCII (listed in the HP41 hex tables) and the ROM character set. The ROM character set is used for most internal operations including coding the ROM function names.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	⊢	a	b	c	d	e	‾	ᵀ	Ⱦ	⊼	大	丰	μ	≠	Σ	∡

Note: the colon(3A) displays as a boxed star. The comma(2C) is also the left facing goose when used in a function name or display, and the period(2E) is also the right facing goose.

When a function is executed, the operating system checks the ROM words containing the first two characters of the function name and the two words immediately following. The catalog table entry for a microcode function (both mainframe and XROM functions) points to the first word of executable code. The function name is listed in reverse order immediately preceding the first word of executable code. For example, CLA(hex 87) has a catalog entry at 1487 of 0D1 which means that the first executable word of CLA is at 10D1. The ROM listing for CLA is:

```
10CE 081 A
10CF 00C L
10D0 003 C
10D1 04E   C=0
10D2 168   REGN=C 5(M)
10D3 1A8   REGN=C 6(N)
10D4 1E8   REGN=C 7(O)
10D5 228   REGN=C 8(P)
10D6 3E0   RTN
```

This shows how the function name is listed in reverse order. To tell the operating system that the end of the function name has been reached, add 080 hex to the final character. For CLA, add 080 to A(001) to get 081 at location 10CE. CLA requires no prompt. To provide a prompt, set the top two bits in the first two characters of the function name by adding the hex constants in the following table:

ADD HEX TO		PROMPT TYPE ACCEPTED						EXAMPLE
1ST CHAR	2ND CHAR	NONE	NULL ALPHA	ALPHA	# DIG	IND & IND ST	ST	
000	(any)	X						CLA
100	000		X	X				CLP
100	100				3,4			SIZE
100	200			X				
100	300				1	X		CAT
200	000				2	X	X	STO
200	100				2	X	X	RCL
200	200				2	X		FS?
200	300				2	X		
300	000			X	2			LBL
300	100			X	2	X		XEQ
300	200			X	2			
300	300			X	2	X	X(.ddd)	GTO

Although STO (2,0) and RCL (2,1) appear to be the same, they are not. If you use the STO combination, the calculator will also accept + - * or / to change the instruction to ST+, etc. Your intended instruction will change to ST+ if you use this combination, and will not execute as you expect. This will also happen for the LBL, XEQ, and GTO combinations. Examples:

```
12D2 097 W      1105 099 Y      12CC 085 E
12D3 005 E      1106 010 P      12CD 00E N
12D4 109 I      1107 00F O      12CE 30F O
12D5 216 V      1108 103 C      12CF 114 T
```

The operating system examines these ROM bits and executes a prompt (if the appropriate bits are set) before the function is executed. If the prompt accepts an ALPHA string, the input data is loaded into the Q register, right justified, in reverse order, in ASCII. For example, ASN "COPY" loads 00 00 00 59 50 4F 4C into Q before ASN is executed. If the prompt is numeric, the input data is loaded into the A register in binary. A numeric input of 55 returns 00 00 00 00 00 00 37 in A. A numeric input of IND 55 returns 00 00 00 00 00 00 B7 in A.

Two other ROM words of a microcode function are examined by the operating system. The first executable word, if a NOP (000), indicates that the function is non-programmable. This means that if you execute the function in program mode, it executes rather than being entered as a program line. SIZE, ASN, and CLP are non-programmable functions. If the first two executable words of an XROM function are both zero, then the function is both non-programmable and executes immediately. This means that no function name is displayed and that the function will not NULL. The function is executed when the key is pressed rather than when the key is released. PRGM, shift, and back-arrow are non-programmable, immediately executable functions. Note that unless your routine checks for key release, and the key to which your function is assigned is held down, the function will be executed repeatedly until the key is released. These two words affect the function operation only if the calculator is in PRGM mode. In RUN mode, they are ignored.

SECTION 9: EXAMPLE OF INITIAL SETUP OF ProtoCODER

More recent ProtoCODERs are shipped containing a single-entry catalog, with all other words cleared. By following the steps below, you can initialize words 000-005, 00D-010, and FF4-FFA to contain a single (dummy) function "ABC".

First, set the address select switches to 3 hex (0011 - off off on on) and set the device select switches to 0 (off off off off).

Second, enter each of the following strings in ALPHA and execute CODE and SIGN after each:

```
"C12FFFFF000FF0"    CODE    SIGN
"C01FFFFF001FF0"    CODE    SIGN
"C00FFFFF002FF0"    CODE    SIGN
"C10FFFFF003FF0"    CODE    SIGN
"C00FFFFF004FF0"    CODE    SIGN
"C00FFFFF005FF0"    CODE    SIGN
"C83FFFFF00DFF0"    CODE    SIGN
"C02FFFFF00EFF0"    CODE    SIGN
"C01FFFFF00FFF0"    CODE    SIGN
"FE0FFFFF010FF0"    CODE    SIGN
"C00FFFFFFF4FF0"    CODE    SIGN
"C00FFFFFFF5FF0"    CODE    SIGN
"C00FFFFFFF6FF0"    CODE    SIGN
"C00FFFFFFF7FF0"    CODE    SIGN
"C00FFFFFFF8FF0"    CODE    SIGN
"C00FFFFFFF9FF0"    CODE    SIGN
"C00FFFFFFFAFF0"    CODE    SIGN
```

Finally, set the address switches to F (on on on on) and run a CAT 2. You should see ABC as the final entry in the catalog listing.

SECTION 10: CALCULATION OF ROM CHECKSUM

If you wish to copy your microcode into an EPROM for permanance, you should calculate the checksum to store in word FFF of your EPROM. To do this, you can use ROMSUM in the NFCROM (see Section 6) or you can write a routine to do it yourself.

The checksum is computed by adding all 10 bit words together. Each time a carry occurs from the leftmost bit, a wraparound carry is performed: add 1 back in to the sum. After adding all 4096 words together, subtract 1. If the result is 000, then the checksum is correct. Note that the checksum is only used by the HP Service Module to verify that an XROM is not damaged or has not had any data altered. The HP41 operating system ignores (and does not check) the checksum.

There are several ways to take care of the 10-bit carry. One way (used by ROMSUM, the HP Service Module, and the HP-IL Monitor Module) is:

Set up A=C00   B=C01   C=x000000 where x is ROM page number to be added then:

```
PT=5
CXISA
A=A+C X
GONC *+2
A=A+B X
C=C+1 WPT
GONC *-5
A=A-B X
```

This leaves the resulting sum in A(2:0).

SECTION 11: USER LANGUAGE VS. MICROCODE TIMING

Microcode can be used to greatly increase the execution speed of some program segments. As an example, run "+1" in the Owners Manual or in NFCROM against an equivalent user language program:

```
LBL 01
+
GTO 01
```

Before execution, set up the stack to contain all 1s. Depending on your reflexes, +1 should count about 125 times as fast as the user language program. By replacing blocks of simple user code instructions with microcode routines, the program as a whole will run much faster. As an example, the sequence of user code:

```
08 RCL 01
09 STO 04
10 CLX
11 STO 09
12 1
13 CLA
```

will execute in about 139 ms (23+20+10+20+57+9) according to PPCTN #6 p3-5. This sequence could be replaced by the microcode routine CLR:

```
08 CLR
```

which will execute in about 38 ms. CLR is the following routine:

```
C=0
DADD=C
PFAD=C
REGN=C 5(M)
REGN=C 6(N)
REGN=C 7(O)
REGN=C 8(P)
C=C+1 S
RCR 1
REGN=C 3(X)
C=REGN 13(c)
RCR 3
C=C+1 X
DADD=C
AC EX
C=DATA
AC EX
C=C+1 X
C=C+1 X
C=C+1 X
DADD=C
AC EX
DATA=C
LDI
CON 005
C=A+C X
DADD=C
C=0
DATA=C
RTN
```

SECTION 12: BRIEF OVERVIEW OF HP 41 DISPLAY PROGRAMMING

Most of the information below was first published by Paul Lind in PPCTN #10.

To operate the display on the HP 41, you must select the display and deselect the RAM. To do this, either:

```
LDI
CON OFD
PFAD=C
LDI
CON 010
DADD=C
```

or execute the system routine to do this (GOSUB 07F6).
After selecting the display, you can write data from the C register into the display, read data from the display into the C register, and write data from the C register into the annunciators.

Each of the 12 character positions of the display is coded with 9 bits. The leftmost bit, if set, specifies that bits 3-0 contain a special character in row 4 of the table in section 8. If the leftmost bit (bit 8) is set, bits 5-4 should be zero, otherwise a space will be displayed. Bits 7-6 define the punctuation field of the character:

00 = no punctuation
01 = period
10 = colon
11 = comma

Bits 5-4 specify which row (0-3) the displayed character is in (see the table in Section 8), and bits 3-0 specify the character within the row.

Data can be read or written to the left or right end of the display. Data is pushed onto the display when written: the rest of the characters are shifted to make room for the incoming data. When data is read, it is pulled off of the end of the display and rotated back in the other end.

Various fields can be written or read from the display:

All 9 bits
Bits 7-0
Bits 7-4
Bits 3-0
Bit 9

A read or write on a specified field affects only that field, so that a write into bits 3-0 rotates only bits 3-0 of each character over, without affecting the other bits (8-4) at all. Data can be read or written in blocks of 1, 4, 6, or 12 characters at one time. When 4 or 6 characters are read or written, each character is moved in one at a time, then rotated, then the next, etc. For a read, the first character becomes the least significant in C, etc. For a write, the rightmost character in C is written first, etc. The table on the next page shows all display read/write instructions and what they do.

The annunciators are read (with C=REGN 5(M)) and written (with DATA=C) to/from the C register, digits 2-0. Bits have the following significance:

Bit 11 = BAT	Bit 10 = USER	Bit 9 = G	Bit 8 = RAD
Bit 7 = SHIFT	Bit 6 = 0	Bit 5 = 1	Bit 4 = 2
Bit 3 = 3	Bit 2 = 4	Bit 1 = PRGM	Bit 0 = ALPHA

HP 41 DISPLAY INSTRUCTIONS

MNEMONIC	HEX	READ/ WRITE	NUMBER OF CHARACTERS	BITS AFFECTED	NUMBER DIGITS OF C TO DEFINE EACH CHARACTER	ROTATION DIRECTION
DATA=C	2F0	WRITE	ALL ANNUNCIATORS		1 bit per annunciator	NONE
C=REGN T	038	READ	12	3-0	1	LEFT
REGN=C T	028	WRITE	12	3-0	1	RIGHT
C=REGN Z	078	READ	12	7-4	1	LEFT
REGN=C Z	068	WRITE	12	7-4	1	RIGHT
C=REGN Y	0B8	READ	12	8	1	LEFT
REGN=C Y	0A8	WRITE	12	8	1	RIGHT
C=REGN X	0F8	READ	6	7-0	2	LEFT
REGN=C X	0E8	WRITE	6	7-0	2	RIGHT
C=REGN L	138	READ	4	8-0	3	LEFT
REGN=C L	128	WRITE	4	8-0	3	RIGHT
C=REGN M	178	READ	ALL ANNUNCIATORS		1 bit per annunciator	NONE
REGN=C M	168	WRITE	6	7-0	2	LEFT
C=REGN N	1B8	?????				
REGN=C N	1A8	WRITE	4	8-0	3	LEFT
C=REGN O	1F8	READ	1	3-0	1	RIGHT
REGN=C O	1E8	WRITE	1	3-0	1	RIGHT
C=REGN P	238	READ	1	7-4	1	RIGHT
REGN=C P	228	WRITE	1	7-4	1	RIGHT
C=REGN Q	278	READ	1	8	1	RIGHT
REGN=C Q	268	WRITE	1	8	1	RIGHT
C=REGN ⊢	2B8	READ	1	3-0	1	LEFT
REGN=C ⊢	2A8	WRITE	1	3-0	1	LEFT
C=REGN a	2F8	READ	1	7-4	1	LEFT
REGN=C a	2E8	WRITE	1	7-4	1	LEFT
C=REGN b	338	READ	1	7-0	2	RIGHT
REGN=C b	328	WRITE	1	7-0	2	RIGHT
C=REGN c	378	READ	1	7-0	2	LEFT
REGN=C c	368	WRITE	1	7-0	2	LEFT
C=REGN d	3B8	READ	1	8-0	3	RIGHT
REGN=C d	3A8	WRITE	1	8-0	3	RIGHT
C=REGN e	3F8	READ	1	8-0	3	LEFT
REGN=C e	3E8	WRITE	1	8-0	3	LEFT