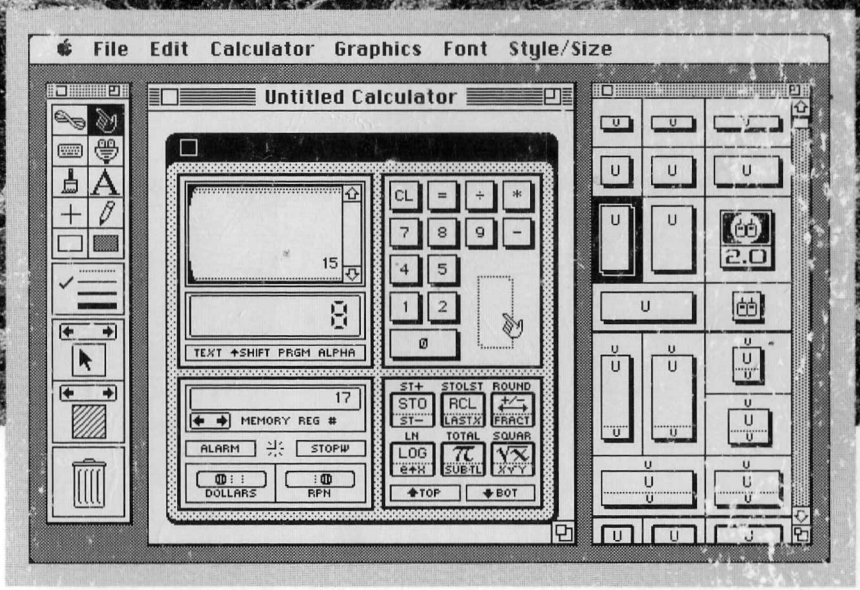
2.0 Power Version!

CALCULATOR

CONSTRUCTION SET



User's Guide

DUBL-CLICK

Calculator Construction Set
2.0 Power Version

REGISTRATION CARD *(Please print so we can read it— Thanks!)*

NAME

ADDRESS

CITY, STATE & ZIP COUNTRY

DAYTIME PHONE NUMBER EVENING PHONE NUMBER

Where did you buy this software?

- ☐ MacConnection
- ☐ MacWarehouse
- ☐ MacZone
- ☐ Programs Plus
- ☐ B.Dalton
- ☐ Egghead
- ☐ ComputerWare
- ☐ Computer store
- ☐ Software store...
- ☐ Book store...
- ☐ As a Gift...
- ☐ Other...

STORE NAME

CITY, STATE & ZIP

What kind of Macintosh System do you use?

Computer:
- ☐ Macintosh (older)
- ☐ MacPlus
- ☐ Macintosh SE
- ☐ Macintosh II

Printer:
- ☐ ImageWriter I or II
- ☐ ImageWriter LQ
- ☐ LaserWriter
- ☐ Other _____

Other:
- ☐ Memory: _____
- ☐ #floppy drives: _____
- ☐ #hard disks: _____

What magazines do you regularly read?

- ☐ MacUser
- ☐ MacWorld
- ☐ MacTutor
- ☐ MacTech Quarterly
- ☐ MacWeek
- ☐ InfoWorld
- ☐ Byte
- ☐ Publish
- ☐ Personal Publishing

What changes would you like in a future version of CCS?

What type of products would you like to see developed?

PLACE STAMP HERE

Dubl-Click Software
9316 Deering Avenue
Chatsworth, CA 91311

Attn: Warranty Department

Pr'td 1989

Blackwater 16 April 2017 (581)

User's Guide

Software designed and written by:
Cliff Joyce and Austin Durbin

Manual written by:
Cliff Joyce, Paul Thomas, and Austin Durbin

Manual edited by:
Karen Moss

Waltham and Austin fonts drawn by:
Mary Martinez, J. Doug Robertson, and Cliff Joyce

Dubl-Click Software, Inc.
9316 Deering Avenue / Chatsworth, CA 91311
(818) 700-9525

9316 Deering Avenue
Chatsworth, CA 91311
(818) 700-9525

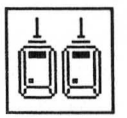
Table of Contents

Disk Exchanges

Exchanging 800K disks for 400K disks

CCS is distributed on 800K media. If your Macintosh cannot read 800K media, you may return the 800K disk(s), and copies of the address labels below, and we will rush you 400K disks at no charge. We apologize for any inconvenience this may cause.

Important To exchange your disks, mail them to this address:



Dubl-Click Software
9316 Deering Avenue
Chatsworth, CA 91311

Please complete this label and mail it with your disks (we will use it for the return mailing label address):

NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

COUNTRY _____

Defective or Erased Disks

Should your original disk(s) become unreadible (it does happen), return the offending disk(s) and copies of the address labels above, and we will replace the disk(s).

Back up your disks!

Although the original disks are not likely to fail, it is a good idea to make back-up copies to avoid accidental erasure. CCS is not copy-protected, so you can make a *Finder* copy of the disks. Be sure you lock the original disks before making backups. If you are unsure of how to make backups using the *Finder*, you should review your *Macintosh User's Manual*.

Forward

Welcome to Calculator Construction Set

Congratulations on your purchase of ***Calculator Construction Set*** (CCS for short). CCS provides you with the ability to design and create powerful, fully programmable calculators that may contain a host of higher functions. The finished calculators you build using CCS may be saved as stand-alone applications, or as desk accessories (DAs).

In designing CCS 2.0, we have tried to include the best of what the calculator world had to offer. We looked long and hard at handheld calculators from Casio, Hewlett-Packard, Sharp, Texas Instruments and others. We studied other Macintosh calculator software packages, and critiqued our own 1.0 version. And we listened to our best advisors— our CCS 1.0 users.

The versatile CCS can build calculators as simple or as powerful as you want. You can improve upon the basic *Calculator* desk accessory that came with your Macintosh, or build calculators that emulate (and even rival!) the most sophisticated calculators. If you don't want to design calculators at all, included with CCS are a number of pre-built calculators.

The beauty CCS is that it can be tailored to fit your individual needs. Perhaps you're looking for an extended 10-key calculator that has a Paper tape. Maybe you'd like a calculator that has financial functions plus the ability to save numeric output to a text file.

CCS can please most of the people, most of the time.

About CCS Technical Support

CCS is an easy application to learn. In fact, seasoned Macintosh users can probably figure most of it out without this manual. Novices will learn CCS quickly by reading the step-by-step tutorials in Chapters 1 through 3.

Occasionally even the most experienced Mac user requires some assistance. This manual has been carefully written to help you make the most of CCS. Chances are the help you seek can be found right here. Please, before contacting us for support, double-check this manual's Index. Perhaps the answer you seek is in a section where you might not have expected it.

After scanning the manual, if you are still unable to find the solution to your problem, please feel free to contact us. The best way is through the mail. A well written letter, together with all information pertinent to your problem (like what kind of Macintosh, *System/Finder* you are using, and perhaps a copy of your calculator work file), provides us with the necessary information and time to research an in-depth reply.

Write to us at:
Dubl-Click Software, Inc.
CCS2 Technical Support
9316 Deering Avenue
Chatsworth, CA 91311

Or, you may contact us via AppleLink. Our account name is:
`DUBL.CLICK`

For technical support via telephone, gather all information pertinent to the problem, and ask for CCS 2.0 Technical Support:
(818) 700-9525, weekdays: 9AM to 5PM, Pacific Std. Time

Our programmers are also available as consultants. If you need a script or an external function written for you, contact our programmers with your specifications and leave the coding to us. For more information on our capabilities and rates, contact CCS Technical Support.

What should I read next?
A Road Map to this Manual

With the exception of the first three chapters, we wrote this as a ***Random Access Manual***. In other words, you don't need to read it cover to cover. Instead, refer to the ***Index*** (in the back) and read only about the topic in question.

Depending upon your knowledge of calculators, the Macintosh, and mathematics— you may want to read at least some of the manual before using CCS.

Read the following to learn:

- **How to Save a Pre-Built Calculator** (Chapters 1,2 & 3).

  Chapter 1 will take you step by step through the process of opening and saving a pre-built calculator.
  Chapter 2 will help you install the finished calculator.
  Chapter 3 will show you how to use the finished calculator.

- **How to Build or Modify a Calculator** (Chapters 1 & 4).

  Chapter 1 has a tutorial that will teach you the basics of building a calculator.
  Chapter 4 is a reference guide which details every window, tool, menu item and calculator part CCS has to offer.

- **How to Use a Finished Calculator** (Chapters 3 & 5).
  Once you've got a calculator built:

  Chapter 3 is a tutorial designed to provide you with the basics of using the calculator.
  Chapter 5 provides an in-depth view of calculator anatomy.

Once again, welcome to CCS!

Important Before starting, please make back up copies of your *CCS 2.0* disk(s), and work from the back up copies.

Chapter 1

Tutorial 1:
Building Calculators

Calculator Construction Set (CCS) is an application in which you may build and save ***finished calculators***. You may save these calculators as stand-alone applications or as desk accessories.

If you have a problem while you are doing the tutorials, go back and reread the step-by-step instructions (it is possible that you skipped a step). If feel that you need more detailed explanation of a particular step or function, see Chapter 4 which covers the same ground (and more) in a reference manual format.

This chapter describes several ways to build a finished calculator using *Calculator Construction Set*:

- **Use a Pre-Built Calculator.** This is the fastest way to build a calculator. If you are anxious to start using a finished calculator and do not want to design one first, then this method is for you. CCS comes with many calculators which are ***pre-built*** or already designed. They are in documents called CCS work files. The next page explains how to save work files as finished calculators.

- **Modify a Pre-Built Calculator.** You can save a lot of time by opening one of the pre-built calculators and modifying it to suit your needs. In order to do this, however, you need to how to design a calculator from scratch (see below).

- **Design a Calculator from Scratch.** The rest of this chapter descibes how to use the CCS tools to decorate and add functions to your calculators, which we think you'll find suprisingly easy.

Saving Pre-Built Calculators

Start by *opening* the Calculator Construction Set application from the *Finder* or *Multifinder*.

Open an application by clicking on its icon, then pull down Open from the **File** *menu. A shortcut method is to* ***double-click*** *the application's icon (clicking twice very fast).*

After a moment, CCS will display its three windows and all its menus which we call the ***desktop***. Your screen will look something like this:

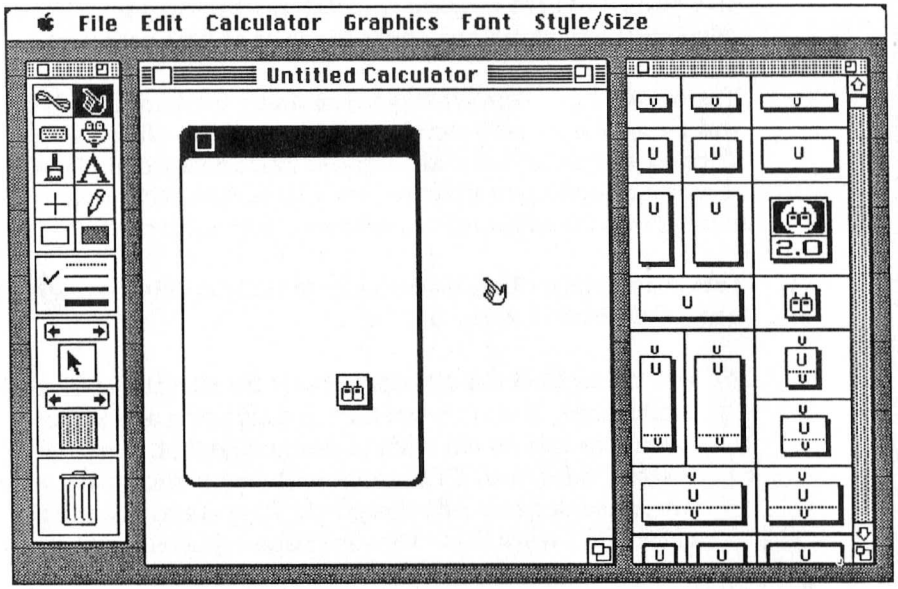


Figure 1-1 The CCS desktop as seen on a Mac Plus

Pull down **Close** from the **File** menu. This will close the middle window which we call the ***Work window***. Since CCS can only edit a single calculator at a time, it is necessary to close this window before opening a document.

Pull down **Open...** from the **File** menu. This will display the ***dialog box*** shown in Figure 1-2. (A dialog box is a window with buttons in it).

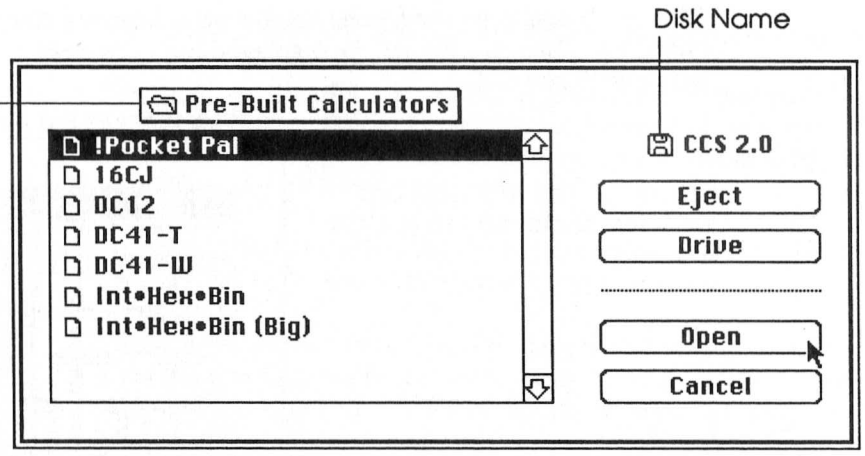


Figure 1-2 Standard "GetFile" dialog box

This window is called the ***Getfile*** dialog box. You use this dialog to open work files. We want to open a work file called *!Pocket Pal* which is located in the *Pre-Built Calculators* folder on the *CCS 2.0* disk.

If the ***Disk Name*** (above the ***Eject*** button) does not say *CCS 2.0*, click the ***Drive*** button, or insert the *CCS 2.0* disk into the Macintosh's disk drive.

If you are using 800K disks, you may have to open the *Pre-Built Calculators* folder to locate the *!Pocket Pal* file. To open a folder, select its name (in the big box with the scrollbar) and click the **Open** button.

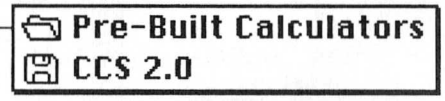


Figure 1-3 Pop-up Disk name/Folder name menu

To return to a previously opened folder, click on the pop-up disk name/folder name menu which is located at the top left of the dialog (see detail in Figure 1-3). Just select the name of a folder or disk to view their contents.

File and folder names are displayed in the large box with the scrollbar in it. To open a document, click on the file's name then click the **Open** button (or double-click the file's name).

After you've selected the *!Pocket Pal* file name from the *GetFile* dialog box, the work file will be displayed in CCS's Work window (Figure 1-4).

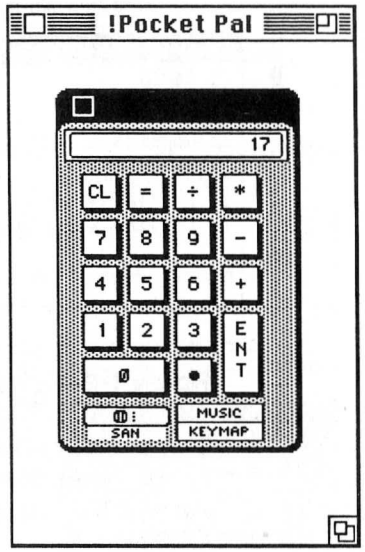


Figure 1.4 The !Pocket Pal work file displayed in the CCS work window

The next step is to save *!Pocket Pal* as a finished calculator. Pull down **Save As...** from the **File** menu. The ***CCS Savefile*** dialog will appear (see Figure 1-5).

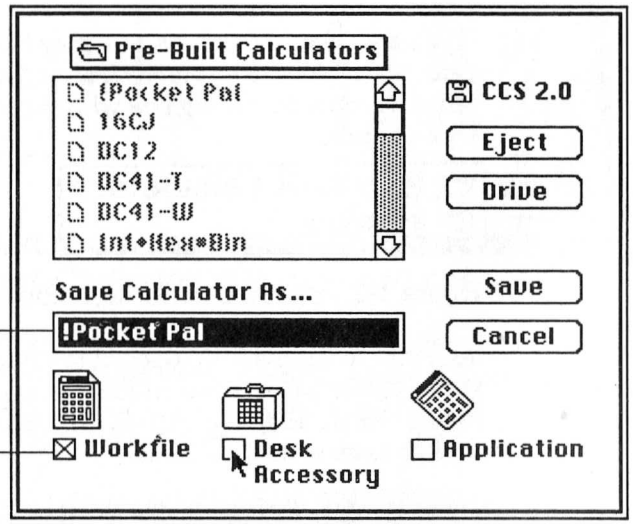


Figure 1-5 The CCS Savefile dialog box

The *SaveFile* dialog works just like the *GetFile* dialog with a few differences. Remember that we are *saving* file(s), not *opening* file(s). All existing file names will be grayed out so that you cannot select one. Folder names are not grayed out, and you may open them. Click the ***Disk*** button to choose the disk you want to save your file(s) on.

At the middle left is a box containing the name of the file you are saving. You may change this name by typing a new one.

The icons at the bottom of the dialog represent the three different kinds of files CCS saves. For now, check them all by clicking on the checkboxes or icons. We'll be saving these three file types:

- **Work file.** This is the only kind of file CCS can open and edit. All pre-built calculators on the *CCS 2.0* disk come in work files. Every time you save a file, a work file will always be saved. If you ***Save As...*** a work file using its original name, you will be asked if you wish to replace the original file. For this example, answer ***OK***.

- **Desk Accessory.** This is a finished calculator whose name will (eventually) appear in the menu (just like the *Calculator* desk accessory that came with your Macintosh). Desk accessories must be installed before they can be used—more on the installation of these in a minute.

- **Application.** This is a finished calculator application, which can be opened from the *Finder* or *Multifinder* the same way you opened CCS at the start of this chapter.

Click the ***Save*** button. That's it! If all went well, you should have three new files on your disk: *!Pocket Pal* (a work file), *!Pocket Pal•CALC* (a desk accessory file) and *!Pocket Pal•SELF* (an application). If an error occurs while saving a file, CCS will let you know what type of error has occurred. For example, if there is not enough room on your disk to save the file(s), CCS will tell you so. You should then use a different disk to save your file(s) on, and try saving the files again.

Pull down **Quit** from the **File** menu to leave CCS.

Opening the Finished Calculators

To start *!Pocket Pal•SELF*, just open it by double-clicking its icon from the *Finder*.

Before you can open *Pocket Pal•CALC*, you need to install it. You'll need to read Chapter 2 for instructions on installing desk accessories. But don't go away yet! Read a little more...

Using the Finished Calculators— Where do I go next?

Using a calculator like *!Pocket Pal* doesn't require much instruction. If you figured out how to use the *Calculator* desk accessory that came with your Macintosh, you have probably already guessed what most of *!Pocket Pal's* keys do.

But we suspect you didn't buy CCS just to replace the standard *Calculator* with *!Pocket Pal*. As you explore the other pre-built calculators, you may find it more difficult to guess what each function does.

Here's where to go next to learn about:

- **Groups of Functions** (Chapters 3 or 5). If you want to know how to use a specific group of functions, like Memory Registers for instance, go to Chapter 3 for a step-by-step tutorial. Chapter 5 presents the same information, but as an in-depth reference manual format.

- **Specific Functions** (Chapter 6). If you've used sophisticated handheld calculators and are familiar with concepts like Modes, Flags, Memory, and Stack Registers, just browse Chapter 6 which is presented in a reference manual format.

- **Building CCS Calculators** (Chapters 1 or 4). To learn more about calculator parts and features unique to CCS (like mapping onscreen keys to keystrokes), keep reading this chapter for a step-by-step tutorial. If you are an experienced Macintosh user (or have no patience for tutorials), you may browse Chapter 4 for the same infomation in reference manual format.

Modifying Pre-Built Calculators and Building Calculators from Scratch

Both these approaches require the same skills, but a different start...

To modify a pre-built calculator, we'll start by opening the pre-built calculator's work file as explained earlier in this chapter (starting on page 1-2). In addition to the work files in the *Pre-Built Calculators* folder, there is a folder called ***Calculator Templates*** containing more work files. The template files are not finished calculators; they just contain the most common keys used on ten-key pads.

To build a calculator from scratch, we'll start with an almost empty calculator shell. The easiest way to get one is to pull down **Close** from the **File** menu to close the Work window (if it was open). Then pull down **New Calculator** from the **File** menu to display an untitled Work window with only a single key on it.

In addition to the Work window, CCS has two ***palette*** windows: the ***Tools palette*** (shown at left in Figure 1-1), and the ***Parts palette*** (shown at right in Figure 1-1).

In general, you build a calculator by dragging ***calculator parts*** (located in the Parts palette) to the Work window. Using the tools in the Tools palette, you can:

Assign functions to parts (a process we call ***wiring***)

Assign keyboard equivalents to keys (what we call ***mapping***)

Decorate parts in the Work window

Wiring a Key

Let's start by wiring (assigning a function to) a key. Click on a square key in the Parts palette (Figure 1-6) and hold the mouse button down. The selected key will be highlighted (surrounded by a black square). Drag the part into the Work window and let go of the mouse button. A copy of the key will then drop into the Work window. Note that the original key is still in the Parts palette, so you can drag as many copies of the key as you want.

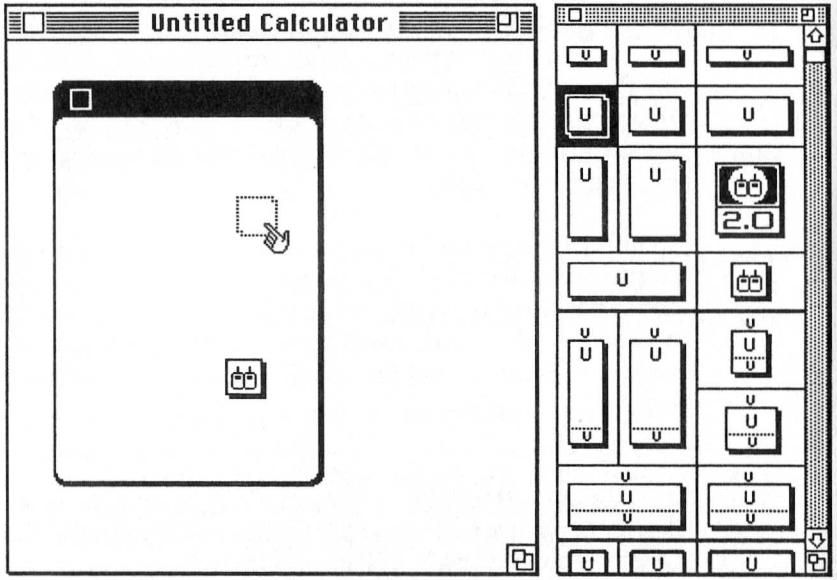


Figure 1-6 Dragging a part into the Work window

This new part does not as yet have any functions assigned to it, and so it has the letter U drawn on it (which means the key is *unwired*). Let's wire a function to it. Select the *Plug* tool by clicking on its icon in the Tools palette .

Figure 1-7
Selecting the Plug tool from the Tools palette

Move your mouse into the Work window, and you will see that the cursor now looks like the plug tool.

Click on the new part in the Work window with the plug. The *Wiring* dialog box will appear (Figure 1-8; on the next page).

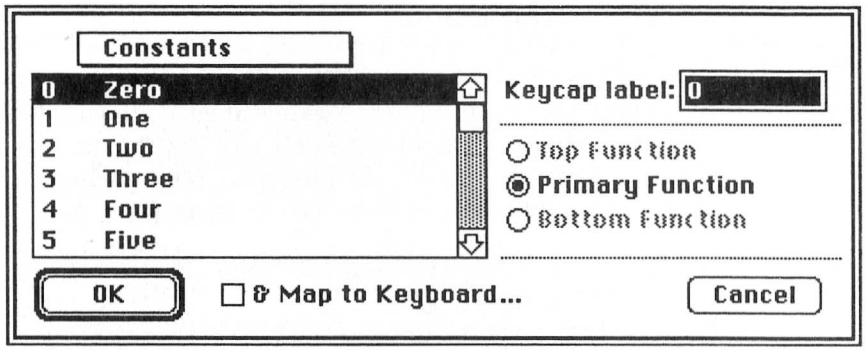


Figure 1-8 The Wiring dialog box (for wiring keys)

This dialog may look different depending upon your Mac's configuration. For now, don't worry if it does look different.

A list of available functions appears in the left side of the *Wiring* dialog (with a scrollbar attached to the list). Select a function by clicking on it, and it will invert (white letters on a black background). In Figure 1-8, the constant **Zero** is selected. Click on the constant **One** to select it.

Remember the U on the unwired square key? The U is the ***keycap label***. When you select a new function in the wiring dialog, the editable text to the right of the caption **Keycap label:** changes. This is just a default label; you are not stuck with it. To change the label, type in the label you want.

Because the square key we are wiring is a ***single function key***, the *Wiring* dialog's **Top Function** and **Bottom Function** radio buttons are grayed out. As you may have guessed, you may wire up to three functions on a multi function key. An unwired multi function key is shown in the margin at left.

Click the **OK** button to wire the key we clicked with the Plug tool, and the *Wiring* dialog will go away. Had you clicked the **Cancel** button, it would be as though you had never clicked on the key with the Plug tool.

Wiring a Multi function key

This time, we'll wire three functions to a key: –, + and =. Drag a multi function key (like the one shown at left) into your Work window, and click on it with the Plug tool. Again, the *Wiring* dialog will appear (Figure 1-8), except the **Top Function** and **Bottom Function** radio buttons will not be grayed out.

Important If your wiring dialog looks like the one in Figure 1-8, skip ahead to the next page.
If your wiring dialog looks like the one in Figure 1-9 (below), you should read this page.

If you are using an older Macintosh or older version of the System file, your *Wiring* dialog will look like the one below. This is because the technology to create pop-up menus is not available on your Macintosh.

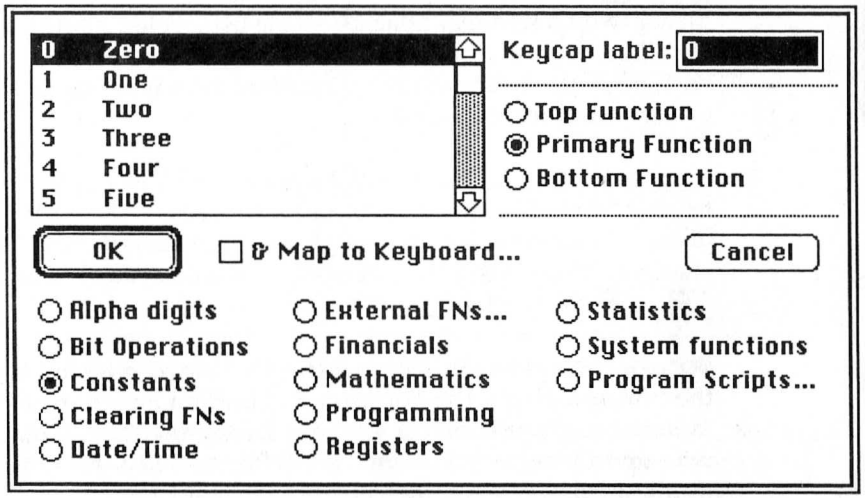


Figure 1-9 The Wiring dialog on an older Macintosh

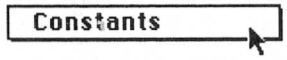
The **Constants** radio button at the bottom of the dialog is selected (with a black dot in the middle of it). This is because the functions in the scrolling box are all from a general group called Constants. Click on the **Mathematics** radio button (to the immediate right of the **Constants** button). Now all the functions from the Mathematics group are displayed.

You may skip the next page as it explains how to change function groups when using the Figure 1-8 *Wiring* dialog.

Important If your Wiring dialog looks like the one shown in Figure 1-9 on the previous page, you don't need to read this page.

If your Wiring dialog looks like the one shown in Figure 1-8, then you should read this page.

The word **Constants** appears in the top left of the *Wiring* dialog because the functions in the scrolling box are all from the general group of Constants.



Click inside the box that says **Constants**. A menu will pop-up as shown in Figure 1-10. Select **Mathematics** from this menu and let go of the mouse button. Now all the functions in the Mathematics group are displayed, and the box at the top of the *Wiring* dialog says **Mathematics** instead of **Constants**.

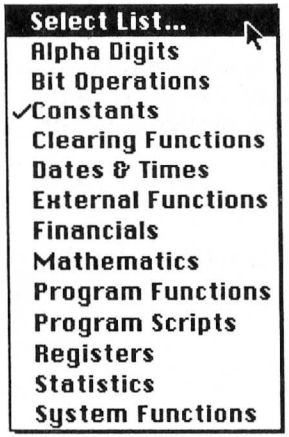
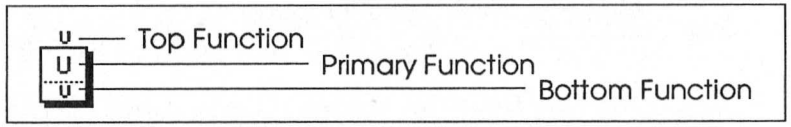


Figure 1-10 Selecting a new group of functions from the pop-up menu in the Wiring dialog

Select the **Subtraction** function from the scrolling list. The ***primary*** function of the multi function key is now wired as subtraction. This is the function in the center of the multi function key, as shown below.



Click the **Top Function** radio button, and select the **Addition** function from the scrolling list. The top function is now wired.

Click the **Bottom Function** radio button. We want to wire the bottom function wired as =. The = function is in the group of System functions; so select that group of functions. When the group of System functions is displayed in the scrolling list, select the **Equals** function. The bottom function of the multi function key is now wired. Click the **OK** button to close the ***Wiring*** dialog.

Important We have divided our functions into groups so there won't be so many to scroll through when wiring. Appendix 1 lists all individual functions and the groups in which you may find them in the Wiring dialog.

Wiring Pushbutton Switches

Let's wire some pushbutton switches next (a sample pushbutton switch appears in the margin). Scroll the Parts palette so you can see the pushbutton switches, then drag one into the Work window. Click on the pushbutton with the Plug tool, and the ***Wiring*** dialog shown in Figure 1-11 will appear.

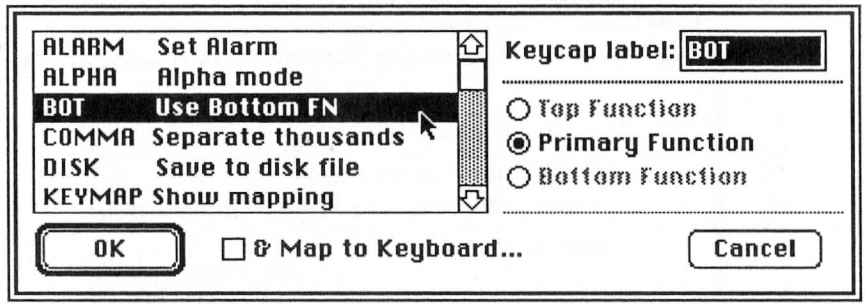


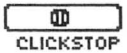
Figure 1-11 Wiring a Pushbutton Switch

Because pushbuttons only have one function, the **Top Function** and **Bottom Function** radio buttons are grayed out. There is only one group of pushbutton functions, so there is no need to change groups.

Select the **Use Bottom FN** item and click the **OK** button.

Drag another pushbutton into the Work window and wire it as the **Use Top FN** item in the ***Wiring*** dialog.

Wiring A Clickstop Switch



Let's wire a clickstop switch to control the format in which the LED will display its results.

Drag a clickstop switch from the Parts palette into the Work window and click on it with the Plug tool. The ***Clickstop Switch Wiring*** dialog will appear, prompting you to choose the kind of switch you want. (Figure 1-12).

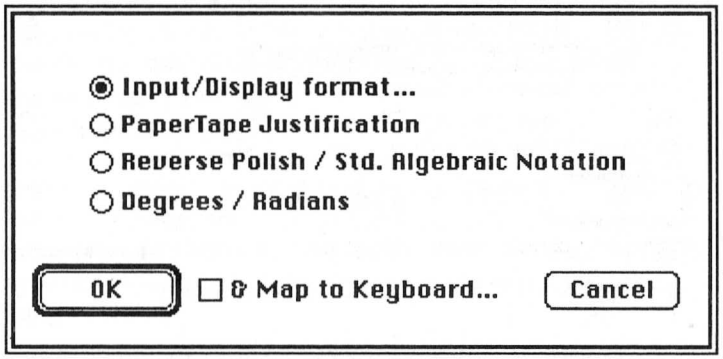


Figure 1-12 Clickstop Switch wiring dialog

Click on the **Input/Display Format...** radio button, then click the **OK** button, and a second dialog box will appear (Figure 1-13). If you selected any other kind of clickstop switch, the second dialog box would not appear, and your switch would be wired.

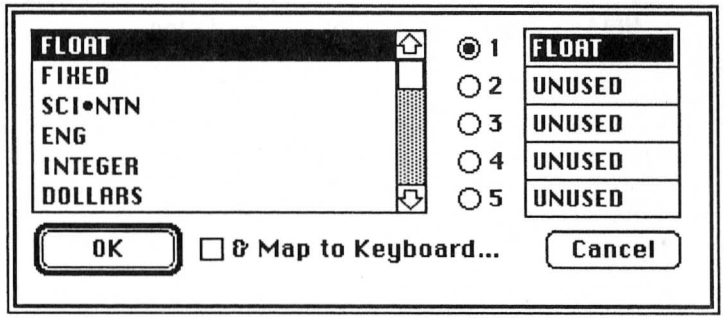


Figure 1-13 Input/Display Format clickstop wiring dialog

The radio buttons numbered 1 through 5 correspond to each clickstop on the switch. Because button number 1 is selected, we are wiring the first clickstop on the switch as the FLOAT format.

The box with the scrollbar on it contains all the display formats that can be wired to a clickstop switch. Select one. Notice how the text to the right of radio button number 1 changes to match the format you just selected. This text is the clickstop label and will draw underneath the switch. Clickstop labels, like keycap labels, may be edited. Select the FLOAT format again.

Select radio button number 2. Click on a format in the list to wire to clickstop number 2. You do not need to wire all the clickstops on the switch. In fact, you may wire as few as one.

Click the **OK** button to complete the wiring process. At any time during construction, you may rewire the switch (or almost any part for that matter), by clicking on it with the Plug tool.

Wiring Display Parts

Scroll the Parts palette so you can see the ***Standard LED*** (shown in Figure 1-14). Drag the Standard LED onto the Work window.

You do not need to wire the Standard LED as it comes *prewired*. Clicking on the Standard LED with the wiring tool will cause a dialog box to appear, telling you that this part is prewired.

There are four other prewired parts in the Parts Palette. All five are shown in Figure 1-14 (below).

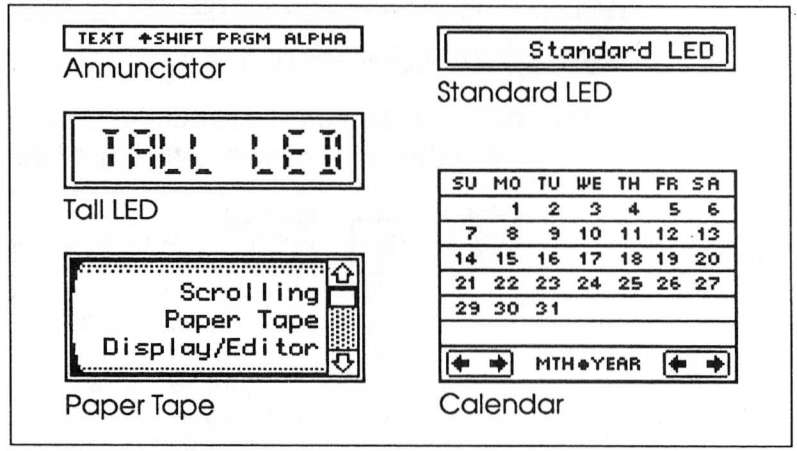


Figure 1-14 Pre-wired Display Parts:

Wiring the Logo Key

By now you may be wondering about the key with the Dubl-Click logo on it. This is the *logo* key. When this key is pressed in a finished calculator, a dialog box will display a message which you wired during construction.

For instance, wire the logo key's text as a reminder to yourself, a message for another user, or the date your calculator was constructed. You can think of the logo key's dialog as an "about" or "help" dialog box.

Click on the logo key with the Plug tool. The dialog box in Figure 1-15 will appear. Type your message in this dialog box. Click the radio buttons at the top of the dialog box to control the way the dialog will draw the text.

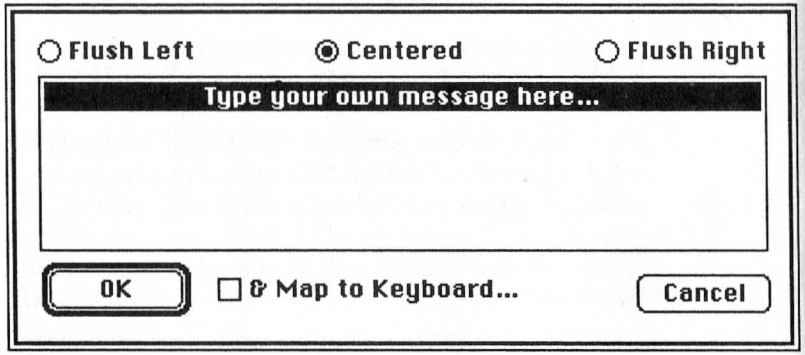


Figure 1-15 Wiring the logo key

The Parts palette has four different logo key shapes. All four shapes function exactly the same and are shown below.

Cleaning Up Your Act

We now have enough parts in our Work window to start making a mess. In order to later click on a part in the finished calculator it must be visible within the Work window's *Calcshell* (shown in Figure 1-16).

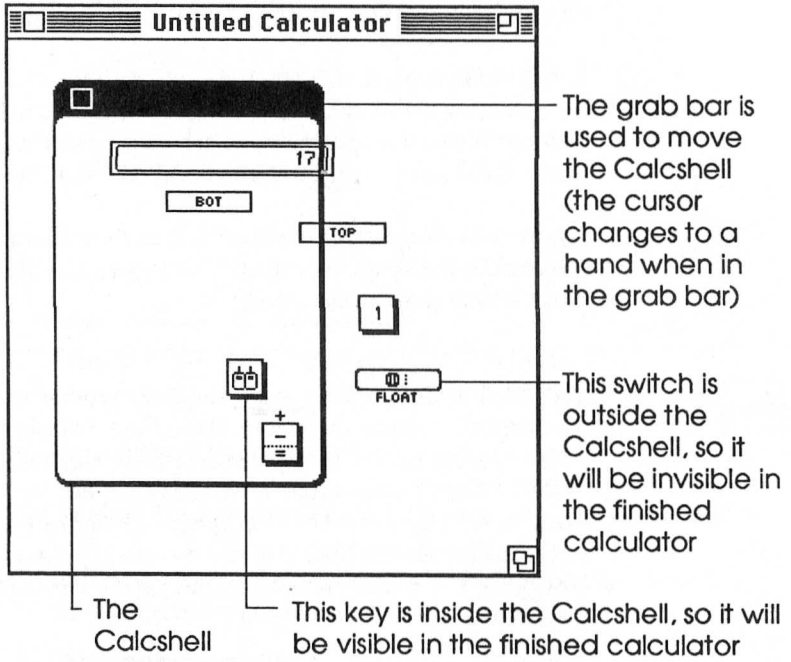
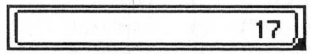


Figure 1-16 The Calcshell in the Work window

There are several ways to make sure all the parts are inside the Calcshell.

- **Move each part one at a time.** Use the *Magic Hand* tool (shown at left) to drag each part inside the Calcshell. If you attempt to drag one part on top of another part, CCS will zoom the part back to its origin, because calculator parts may not overlap. Clicking on a part in the Work window with the Magic Hand will *select* the part. The part will be drawn in some special way to indicate that it is part of the selection. Selected parts are shown in Figure 4-11.

For now, click on the LED in the Work window with the Magic Hand tool. See that little black rectangle in the lower right corner of the LED? We call that a ***knob***.



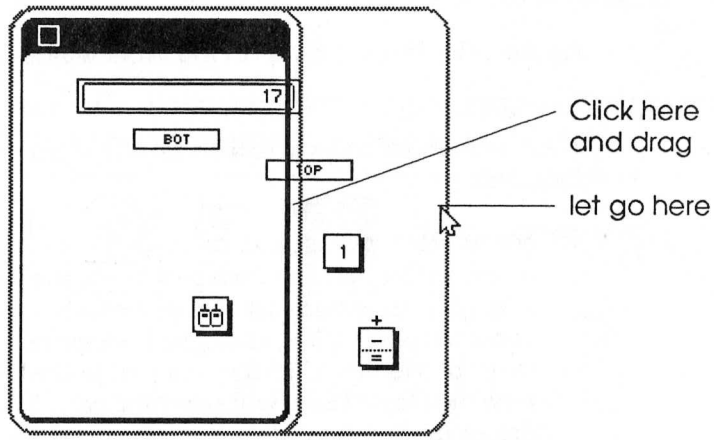
A selected LED with a knob in the bottom right corner

The Magic Hand has one other trick up its sleeve. Clicking on a part's knob will allow you to stretch the part.

Try stretching the LED now (you will only be able to stretch its width, by the way). Notice the number in the LED. This number indicates how many characters fit in the display when the display is used in the finished calculator.

Figure 4-11 shows each kind of part as they would appear if selected in the Work window. This figure also lists those parts which may be stretched.

❖ **Stretch the Calcshell.** Select the ***Rubberband*** tool (shown in margin). A band is drawn around the Calcshell. Click inside this band, and drag to resize the Calcshell as shown below. When your mouse is inside the band, the cursor will draw as a hollow arrow. Dragging a corner of the band stretches the Calcshell in two directions simultaneously. Dragging any other part of the band stretches it in one direction only.



❖ **Auto-Resize the Calc Shell.** Pull down this item from the **Calculator** menu to shrink or expand the Calcshell to the smallest size that will enclose every part in the Work window. Auto-resizing transformed the calculator shown in Figure 1-16 to that is shown below in Figure 1-17.

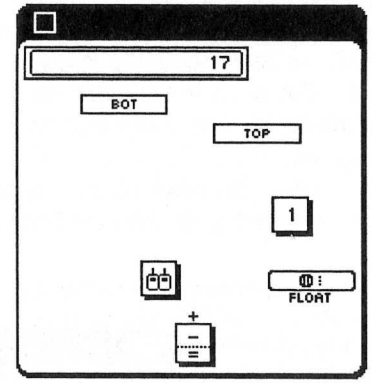


Figure 1-17 The Calcshell after Auto-Resizing

❖ **Radically Cleanup All Parts.** Pull down this option from the **Calculator** menu to move all parts in the Work window as close together as possible. The Calcshell is then auto-resized and moved to the top left corner of the Work window. Figure 1-18 (below) shows the result of a radical cleanup on the calculator shown in Figure 1-16.

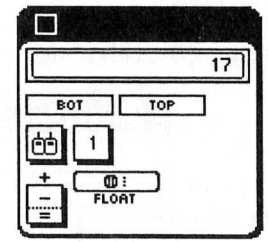


Figure 1-18 The Calcshell after Radical Cleanup

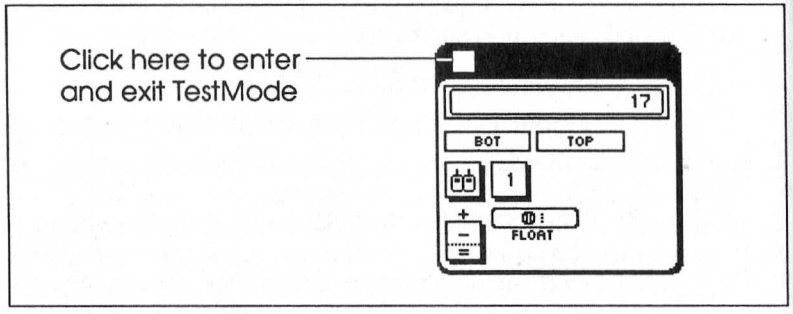
When one or more parts is selected, this menu item will say **Radically Cleanup Selected**, which will move only the selected parts close to each other, without affecting the Calcshell or the other parts in the Work window.

Testing your Calculator

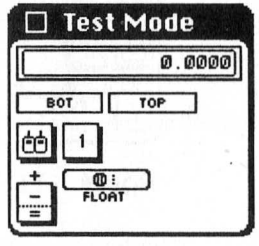
While your calculator is in the Work window, all its parts are simply objects which can be moved, wired, and decorated. You cannot use them to calculate anything.

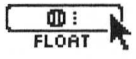
At any time during the construction process, you may enter what we call ***Test mode.*** In Test mode, the Work, Tools, and Parts windows fall away, the Calcshell is transformed into a calculator window, and the calculator's parts will function.

There are two ways to enter and exit Test mode: pull down **Test mode** from the **Calculator** menu, or click the "closebox" on the Calcshell.



Go into Test mode. Look at the LED; it no longer displays `17` (the number of digits that will fit in the display), but now displays `0.0000`. It is displaying the amount of zero because we have not yet keyed-in a number. There are four zeroes to the right of the decimal point because the calculator's ***input display format*** is set to `FIXED` format.





Look at the clickstop switch. It looks the same as it did in the Work window. Now click on the switch. The text under the clickstop switch changed, and is now inverted (white type on black background). This indicates that the clickstop switch now matches the calculator's input display format. Each time you click the switch, the display format changes. Click the switch again so that the current input display format is `FLOAT`.

Click on the `1` key. `1` appears in the LED. Type the number `1` on your keyboard. `11` appears in the LED. This is because the `1` key has been ***mapped*** to a key on the keyboard (more on key mapping in a moment).

Now click the multi function key. Which function was invoked? If you guessed the primary one you are correct; we just invoked the minus function.

Press the `1` key again, and the LED will display a single `1`. So far, our calculator is trying to solve the problem `11 - 1`. To get the result, we need to use the equals function which is also wired to the multi function key.

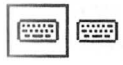
Click on the `BOT` pushbutton. The pushbutton will invert (draw as white type on a black background). This tells us that the `BOT` shift mode is active. Press our multi function key again. The equals (bottom) function is executed, and the `BOT` shift mode is cleared. The LED displays our result: `10`.

The `TOP` pushbutton works the same way as the `BOT` does. Now exit Test mode, and let's learn more about key mapping...

Mapping Onscreen Keys to the Mac's Keyboard

As we've seen, there are two ways to execute a key's function, both of which this manual will refer to as ***pressing*** the key:

(1) by ***clicking*** the onscreen key with the mouse;
(2) by ***typing*** the keystroke which the key was mapped to during construction.



Click on the logo key with the ***Keyboard*** tool (shown in margin). The ***Map to Keyboard*** dialog box will appear (Figure 1-19).

The logo key was not mapped when we clicked on it, so in the *Map to Keyboard* dialog box, it is **Currently mapped to:** `NUL`. That is also why the **Map to Key** and **Unmap Key** buttons are grayed out.

Any key which draws as shaded in this dialog cannot be mapped to the keyboard because it is either unmappable or already mapped to another key.

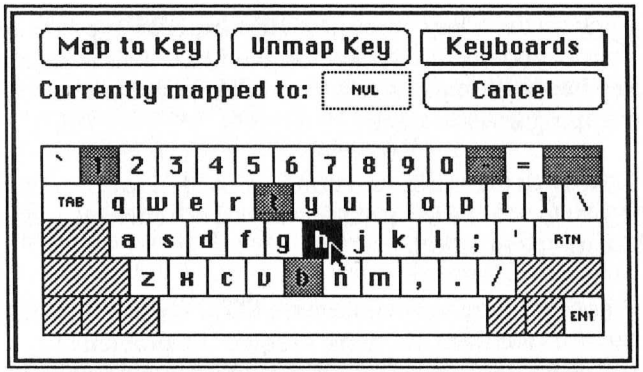


Figure 1-19 The Map to Keyboard dialog box

Click on the h key with your mouse, or type an h from the keyboard. The h key is inverted in the dialog box. Click the **Map to Key** button which maps the logo key to the letter h.

You may have noticed the **& Map to Keyboard...** checkbox in some of the *Wiring* dialog boxes. If this box is checked when you click the *Wiring* dialog's **OK** button, the *Map to Keyboard* dialog will appear immediately afterward.

Drag the logo key off the Calcshell using the Magic Hand tool. Go back into Test mode. In Test mode, the logo key is invisible because it is outside the Calcshell. But we can still press the logo key by typing the letter h, because we mapped the logo key to the letter h. Press the h now. The logo dialog appears with the message we wired earlier. Click in the dialog to make it go away.

Mapping keystrokes to invisible parts works best when the keystroke is easy to remember. For instance, commonly used functions like +, −, *, +, = and constants 0 through 9 are very easy to remember. You may also find it easy to remember that the TOP and BOT pushbuttons are mapped to letters T and B.

If you can't remember the character you mapped to a particular part, just click on the part with the Keyboard tool; the *Map to Keyboard* dialog will display the current mapping. To view all mappings in the Work window, pull down **Show Key Mapping** from the Calculator menu, and the key will redraw with the mapped character on its keycap. In a finished calculator, you can wire a pushbutton switch called KEYMAP to do the same thing.

Interior Decorating

The great thing about CCS is that you are in control— over the way your calculators function *and* look. Decorating a calculator may at first seem a bit frivolous, but it can actually help you remember how to use your calculator.

For instance, imagine how difficult it would be to use a calculator that had no keycap labels! This is a bit extreme, but a calculator with cryptic keycap labels can be just as difficult to operate as one with *no* labels at all.

CCS lets you put text or graphics on the calculator shell itself, and most of the keys the calculator contains.

Changing Keycap Labels

We've already seen one way to change keycap labels: via the *Wiring* dialog boxes. But you don't have to rewire keys just to change their keycap labels. CCS comes with a ***Text*** tool which can do the job faster than a *Wiring* dialog can.

Select the Text tool in the Tools palette. When your mouse is in the Work window, the cursor will draw as the "I-beam" (shown in the left margin).

Click on the multi function key in the Work window with the Text tool. The ***Change Keycap Label*** dialog box will appear (Figure 1-20). Simply type the new label(s) you want into the dialog box. Use the TAB key to advance from label to label.

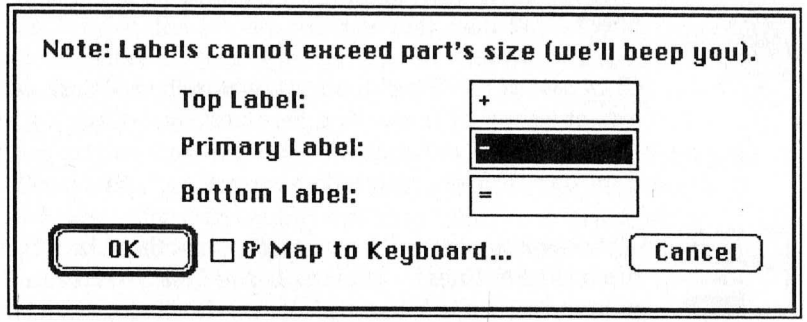


Figure 1-20 Changing a Keycap's Label(s)

Creating Captions with the Text tool (graphic parts)

The Text tool can also be used to create ***captions***, which are a form of ***graphic part***.

> *Graphic parts are simply decoration. You do not wire them to functions, or map them to the keyboard. You can drag graphic parts with the Magic Hand tool. Any part may be positioned on top of a graphic part.*

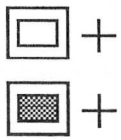
Select the Text tool and click it anywhere in the Work window—except on top of an existing part. A blinking insertion point will appear and you may start typing your caption. Type `Caption`.

Now select the Magic Hand tool and try dragging the caption around. When the caption is selected, you can change the font, size and style the caption draws in via the standard **Font**, **Size** and **Style** menus.

Select the Text tool again and click in the caption. While the blinking insertion point is present, you may edit the caption's text. To stop the editing process, just select a different tool, move the Calcshell or start typing a new caption.

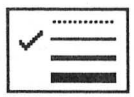
Creating Lines and Boxes (graphic parts)

Drawing boxes and lines around keys and displays can really help you find groups of functions in a hurry. This is especially true in calculators that contain many functions.



To draw boxes, just select either the **Framed Box** or **Filled Box** tools from the Tool palette (shown in margin along with the crosshairs cursor they use).

Click one of the "Box" tools anywhere in the Work window—except on top of an existing part, and start dragging your mouse. As you drag, a rectangle is drawn. When you let go of the mouse button, the rectangle becomes a graphic part.



The new box part will be framed with a thin black line. You may draw thicker lines by clicking in the ***Line Selector*** in the Tool palette (shown in left margin), and the cursor associated with the "Box" tools will redraw accordingly. Select the "dotted" line weight to draw boxes without frames (with Filled Box tool only).

To change the line weight of an existing box part, select the box part and pull down **Use √'ed Line Weight** from the **Graphics** menu, or select the box part and click the Line Weight selector in the Parts palette.

If you used the Filled Box tool to create your box part, the interior of the box will be filled with a pattern. The ***Pattern Selector*** (Figure 1-21) in the Parts palette determines which pattern the Filled Box tool will use.

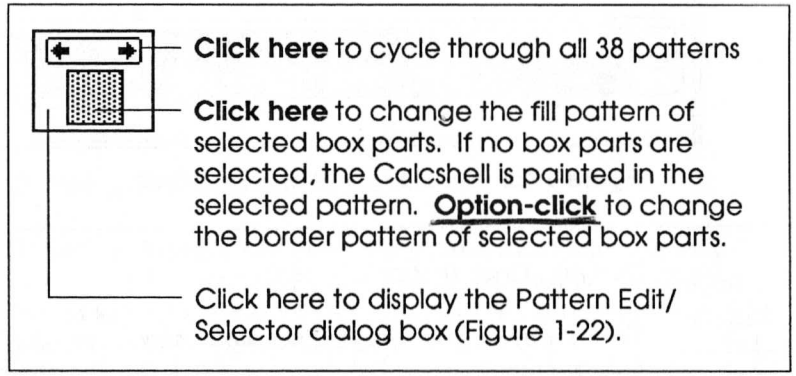


Figure 1-21 The Pattern Selector (in the Parts Palette)

To change the pattern of an existing box part, select the box part and pull down **Fill with Pattern** from the **Graphics** menu, or select the box part and click on the pattern in the Pattern Selector. If you hold down the option key while clicking in the Pattern Selector, the box's outline will be drawn using the selected pattern.

You can change the pattern displayed in the Pattern Selector by clicking on the arrows shown in Figure 1-21, or by pulling down **Edit Pattern...** from the **Edit** menu. The ***Edit Pattern*** dialog box will appear (Figure 1-22).

In the *Edit Pattern* dialog box... Edit the pattern by clicking on the "fatbits" in the upper left corner, and by clicking the 12 editing buttons to the right of the fatbits. You can select a different pattern by clicking on one in the bottom of the dialog.

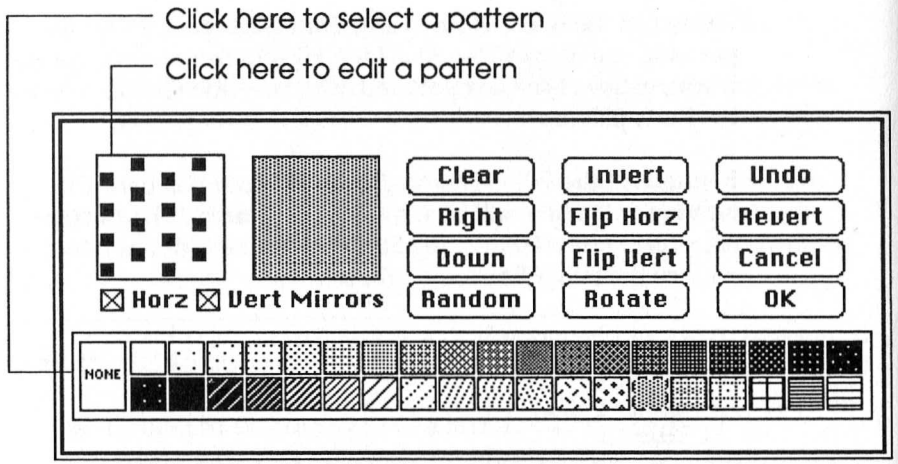
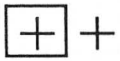


Figure 1-22 The Edit Pattern dialog box

Drawing Lines (graphic parts)



To draw lines, just select the **Crosshairs** tool from the Tool palette (shown in the left margin). Click the Crosshairs tool anywhere in the Work window (except on top of an existing part) and start dragging your mouse. As you drag, a line is drawn either vertically or horizontally depending upon which direction you started moving first. When you let go of the mouse button, the line becomes a graphic part.

Importing Graphic Parts

Not only can you create graphic parts in CCS, but you can import them from other applications. Start by copying your text or picture from the *Notepad*, *Scrapbook*, or *ArtRoundup* desk accessories, or from another application you may be running in *Multifinder*. After you copy your text or picture, put the desk accessories away, or if in *Multifinder*, go back to CCS.

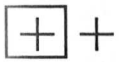
Make sure there are no parts selected in the Work window. You can do this by selecting the Magic Hand tool and clicking once in the Work window— anywhere except on a part. Pull down **Paste** from the **Edit** menu, and the text or picture will become a graphic part.

Pasting Graphics onto Key Caps

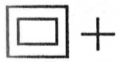
You can also replace a key's normal keycap label with a picture. Start by copying the picture to the clipboard as we did on the previous page. Then select a key in the Work window with the Magic Hand tool. Pull down **Paste** from the **Edit** menu. The picture will paste on to the keycap, and you may reposition the picture by clicking on it and dragging it around. As soon as you click outside the keycap, the picture is permanently positioned on the key.

CCS comes with *MacPaint* and *Scrapbook* files that contain many pictures you might want to paste onto your keycaps.

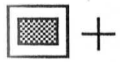
Drawing Graphics onto Keys



Using the ***Brush***, ***Crosshairs***, ***Pencil*** and ***Box*** tools, you may draw on any wired key. Select one of these tools and click on a wired key. The graphics you draw will not become new graphic parts, but will instead decorate the wired key.



Drawing Graphics onto Graphic Parts



You can also draw on box and text parts, or graphic parts you pasted into your calculator. Select one of the graphic tools and click on a graphic part. If the graphic part has not been edited before, you will be asked to confirm your choice. A graphic part which has been edited is stored as a *MacPaint*-style ***bitmapped*** graphic.

Note Often, bitmapped graphic parts take up much more memory and disk space than unedited graphic parts do.

Where do I go from here?

You have just learned the basics— most of what you should know to start ***building a calculator***. This tutorial did not cover everything CCS can do; just enough to get you going.

Browse Chapter 4 for an ***in-depth tour of each of CCS's windows, parts, tools and menus.***

To learn how to ***install a finished calculator***, go to Chapter 2 (on the next page).

To learn the basics of ***using finished calculators***, go to Chapter 3.

Chapter 2

Tutorial 2: **Installing Finished Calculators** (and fonts)

A Word on System versions

As of this writing, Apple has announced a new version of the Macintosh System: ***System 7.0***. Software developers (like us) have not yet received copies of the new System software, so we do not know exactly how it works. We do know that the way you install desk accessories and fonts will be radically different from the instructions on the following pages.

Important If you are using a System older than 7.0 (a lower number), you may skip this page.

If you are using System 7.0 or newer (a higher number), you should read only this page of this Chapter.

Installing Finished Calculators using System 7 (or newer)

Before you can use your finished calculators, you'll need to install them by following the instructions that will accompany *System 7*.

If you saved your calculator as a desk accessory, our guess is that you can simply drag the calculator desk accessory file into your *System Folder* to add it to the menu.

It is also rumored that *System 7* will allow you to add calculator application names to the menu as well. We have no specific information on how to accomplish this, so be sure to consult the *System 7* documentation for more.

Installing Finished Calculators and Fonts using System 6 (or older)

Before fonts and finished calculators (saved as desk accessories) can be used, they must be installed into the ***System file***. The ***Font/DA Mover*** is an application used for moving (installing) fonts and desk accessories into ***System*** and other files.

The ***Font*** and menus will always contain the names of all the fonts and desk accessories contained in your ***System file***.

Important You must use the *Font/DA Mover* to install fonts, or finished calculators (saved as desk accessories) before using them.

If you saved your calculator as an application, you do not need to install it. Simply double-click its icon to open it.

1 Start the *Font/DA Mover* by double-clicking its icon (shown at left). The screen shown in figure 2-1 will appear.

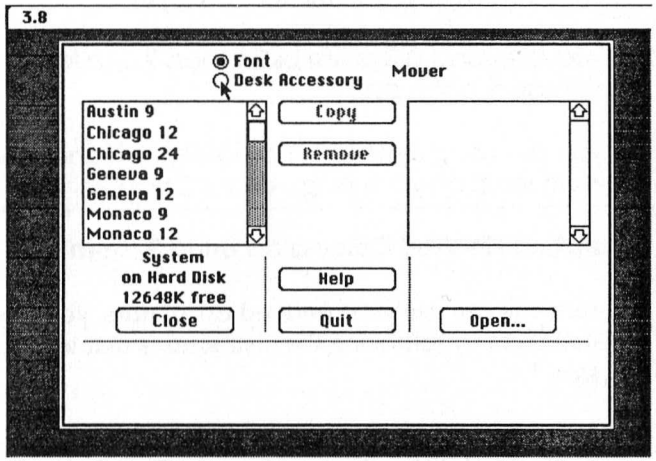


Figure 2-1 Clicking the desk accessory radio button

2 When *Font/DA Mover* is first started, all fonts installed in the current ***System file*** are displayed in the scrolling list on the left side of the screen. The name of the System file appears below the list of fonts.

If you want to install desk accessories, Click the ***Desk Accessory*** radio button (at top of figure 2-1), to display the list of desk accessories instead of the fonts.

3 Click the ***Open...*** button (bottom right of figure 2-2) to open a font or desk accessory file.

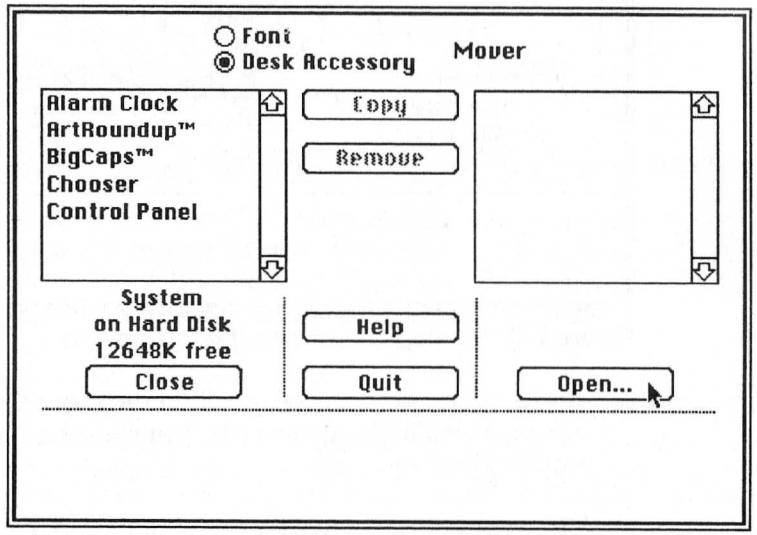


Figure 2-2 Clicking Font/DA Mover's Open button

4 The standard ***Get File*** dialog box will appear (Figure 1-2). Locate and ***Open*** the file containing the fonts or desk accessories you wish to install. Note that the *Austin* and *Waltham* fonts are both in the ***CCS Fonts*** file in the ***Utilities*** Folder on your CCS disk.

If you do not know how to use the standard ***GetFile*** dialog, refer to Chapter 1 for complete instructions.

5 After the standard ***GetFile*** dialog box has disappeared, the name of the fonts or desk accessories available for installation are displayed in the scrolling list on the right side of the screen (Figure 2-3).

Click on the names of the fonts or desk accessories you wish to install. The ***Copy*** button will no longer appear grayed-out. Click the ***<<Copy<<*** button.

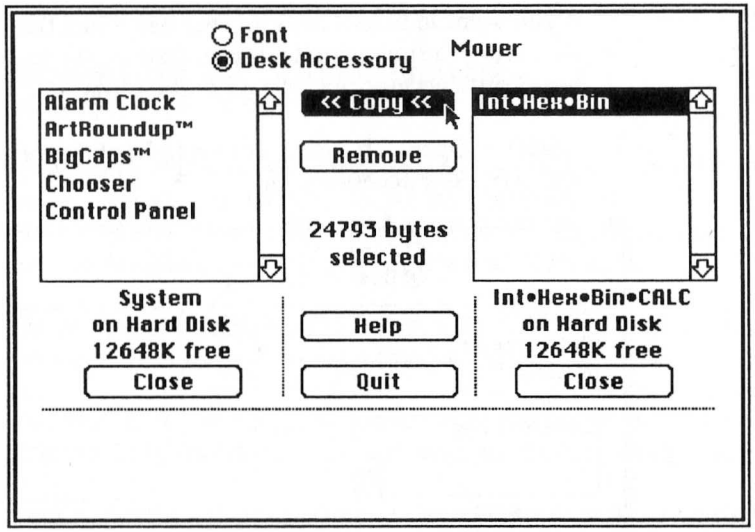


Figure 2-3 Moving a font into the System file

6 Congratulations! Your fonts and/or desk accessories are now installed in the ***System file***. Just click the ***Quit*** button to exit the *Font/DA Mover*.

The names of all the fonts installed in the ***System file*** will appear on the font menus in most applications. The names of all the desk accessories installed in the ***System file*** will appear on the menus in most applications.

Important Installing a font or desk accessory is a one-time operation; after it is installed, you should never have to re-install it into the same System file.

Chapter 3

Tutorial 3: **Using Finished Calculators**

In this tutorial, you will learn enough about your finished calculator to start using it. This chapter will make reference to many ***tutorial calculators***. These are CCS work files located in a folder called *Tutorial Calculators* on the *CCS 2.0* disk.

Important You should already know how to open and save a finished calculator, and how to get into CCS's Testmode. If you do not, please read Chapters 1 and 2 before continuing.

Most CCS functions have names called ***function names***. When this chapter refers to a function, it will shown the function's function name in the `Courier` typeface. The function name will not agree with the label drawn on the key if the function name won't fit on the keycap.

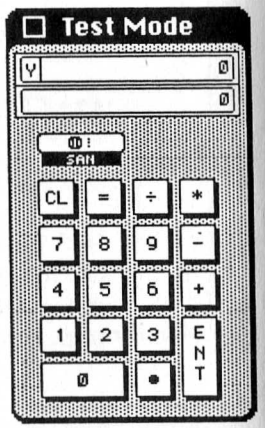
First Day of School.

You will get the most out of this chapter if you follow right along with the tutorial and do the calculations on your Macintosh, as they are done in the manual.

Make sure the *Calculator Construction Set* is running on your computer. If it is not, open the CCS application now. Close the **Untitled Calculator** window (which we will call the ***Work window*** for the rest of this chapter). Now turn the page to begin the first lesson...

Lesson One: Reverse Polish Notation (RPN) Standard Algebraic Notation (SAN)

Don't let these fancy names fool you. This is one of the easiest concepts to grasp in CCS.



Open a work file called *01_Tutorial*. Go into Test mode. Your Test mode window should look like the picture at right.

Look at the clickstop switch just below the two LED displays. The switch's label, SAN, is highlighted. This is because the calculator is in SAN (***Standard Alebgraic Notation***) mode.

You may be surprised to learn that you already know how to use Standard Algebraic Notation! We all learned it in grade school. Let's do that classic SAN calculation: 1+2=3.

Press the 1 key and the ***MainLED*** displays 1.

> *The MainLED is either the pre-wired Standard LED, or the Tall LED part. There can be only one MainLED per calculator (although calculators can have no MainLED as well). You can tell the MainLED from a SLED because SLEDs always have a single digit label (in a box) on the left side. The display with the "Y" on it is a SLED wired to display the Y-Register.*

Press the + key. The displays do not change. Press the 2 key. Notice how the original 1 moves into the SLED marked with a Y (called the ***Y-Register***), and the 2 goes into the MainLED.

Press the = key. The 1 and the 2 are added together, and the result of 3 is displayed in the MainLED.

Using Reverse Polish Notation

Now let's try the same thing in ***Reverse Polish Notation***. First, press the clickstop switch so the text below it says RPN. Note that any time you toggle between SAN and RPN, the MainLED and the Y-Register are cleared to zero. You'll learn why in Lesson Two.

Before we begin our exercise, a short (optional) history lesson...

> *SAN places math operators between variables (numbers) when evaluating algebraic expressions, as in 1+2=3. Polish logician Jan Lukasiewicz (1878—1956) developed a mathematical logic which specifies the operators before the variables. This became known as "Polish Logic." A variation of this logic places the operators after the variables. Hence, the term "Reverse Polish Notation."* [1]

In RPN, we need to rewrite our equation a little. Instead of entering the numbers with operators between them, we enter the numbers, *then* the operators. Here's how:

Press the 1 key. The MainLED displays 1. Press the ENTER key. The 1 moves into the Y-Register. Press the 2 key, which moves the 2 into the MainLED.

Now that we have entered our variables, we can enter the operator. Press the + key. The 1 and the 2 are added together, and the result of 3 is displayed in the MainLED. In RPN, the equation is evaluated immediately after we enter the operator, so there is no need to press the = key to end the equation. By the way, for the rest of this manual, we will refer to operators as functions.

[1] Hewlett-Packard, *HP-41CX Owner's Manual*, (Corvallis: Hewlett-Packard Company, 1984), p. 17.

Lesson Two: The Stack Registers

Most of the data used for immediate calculations is stored in an area of the calculator's memory called the ***Stack Registers***. Each register can hold one number. CCS allows you to configure from four to ten stack registers via the **Memory Allocation** item on the **Calculator** menu (Chapter 4 has more on this).

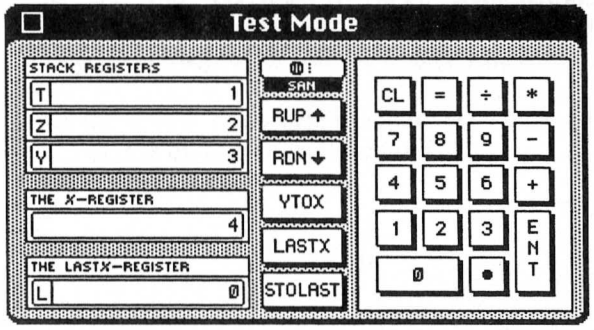
You already met the Y-Register in Lesson One. Now meet the whole family: `X`, `Y`, `Z`, `T`, `S5`, `S6`, `S7`, `S8`, `S9` and `S10`. In addition, there is cousin called the ***LastX-Register***. If you only configure four registers, `S5` through `S10` will not be present.

Exit Test mode, and close the *1_Tutorial* calculator. Open the *2_Tutorial* calculator. Go back into Test mode.

The *2_Tutorial* calculator has only 4 Stack Registers. We have wired ***SLEDs*** (supplemental LEDs) to display the `Y`, `Z`, `T`, and `LastX` Registers. Since the MainLED displays the X-Register, we don't need a SLED for the X-Register.

Lifting the Stack

Let's fill the Stack Registers up to get a feel for how they operate. Key-in: `1`, `ENTER`, `2`, `ENTER`, `3`, `ENTER`, `4`. Notice how each time you key-in a number, it goes into the X-Register. When you press the `ENTER` key, each register moves up to the next highest register. This is called automatic ***stack lift***. Press the `ENTER` key one more time. Notice how the T-Register gets replaced with the Z-Register, and the old value for `T` is lost.



Rolling the Stack

We can ***roll*** the stack manually, too. Press the roll up (`RUP`) key. See how all the registers move up to the next register? Notice how the highest register, `T`, moves back down to `X`. This is different from stack lift, where `T` was lost. Press the roll down (`RDN`) key. The registers now contain their original values; `X` moves out of the bottom and replaces `T` at the top. Rolling the stack is said to be ***non-destructive***— we don't loose any values.

Calculations Requiring Two Numbers

If you feel more comfortable in `SAN` rather than `RPN` mode, this is about all you need to know about Stack Registers: how to get numbers into the X- and Y-Registers. Many of the functions in CCS operate using two numbers. With some, you'll use the `ENTER` function to key-in a value for `Y`. Wiring SLEDs to display these registers keeps it simple— you won't be flying blind.

For instance, let's calculate the cube of 10. Remember from math class? Right— that's 10^3, or 10 to the power of 3. CCS has a function that's tailor-made for this calculation: ***Y To The Power Of X*** (`YTOX`). Chapter 6 describes `YTOX` like this:

> *Raises the number in the Y-Register to the power of the number in the X-Register. The stack is dropped, and the result is returned in the X-Register.*

Now we just need to get the number 10 into the Y-Register, and the number 3 into the X-Register.

Key in the number `10`; the X-Register contains `10`. Press the `ENTER` key; the stack lifts, so now the Y-Register contains `10`. Key in the number `3`; the X-Register contains `3`. We did it! Press the `YTOX` key. The result (`1,000`) is displayed in the X-Register. The stack drops (`T` moves to `Z`; `Z` moves to `Y`).

The LastX-Register

One final twist... In general, when you execute a function, the X-Register is automatically copied to the LastX-Register *before* the function is executed. In our YTOX example, the LastX-Register contains 3 because the X-Register contained 3 before we executed the function. Often, it is useful to recall the previous value for X. Use the function LASTX to do just that.

Let's fill the stack again. Key-in: 10, ENTER, 20, ENTER, 30, ENTER, 40. Press the LASTX key. The stack lifts, and the LastX-Register is copied into the X-Register— which now contains 3.

You can also manually store a value to the LastX-Register at any time by using the function STOLAST. Key 123 into the X-Register. Press the STOLAST key, and the LastX-Register contains 123.

If you don't want to learn about RPN, you can skip to Lesson Three.

Stack Drop in RPN

Press the clickstop swich to get into RPN mode. The entire stack clears. This is so the calculator does not get confused when switching notations; recall that SAN expects functions to be placed between numbers and RPN expects functions to follow the numbers.

Key-in: 1, ENTER, 2, ENTER, 3, ENTER, 4. Now press the + key repeatedly. Watch what happens each time you press the + key: the number in X goes into LastX. Y gets added to X. T drops to Z, and Z drops to Y. If we press the + key enough times, we eventually keep adding 1 to X. This is because the number in T is propagated to the lower registers each time the stack ***drops***. Stack *dropping*, like stack lifting is said to be ***destructive***. That is, we always loose a value at one end of the stack.

Lesson Three: The Memory Registers

In many ways, Memory Registers are like Stack Registers. They can both hold a single numeric value. Just as Stack Registers are work areas for the calculator's use, Memory Registers are work areas for your use. Memory Registers are places to store intermediate results— freeing up the Stack Registers for further calculations.

Memory Registers, like Stack Registers, have names. They are numbered from 0 to 254. During construction, you may configure from 10 to 255 Memory Registers.

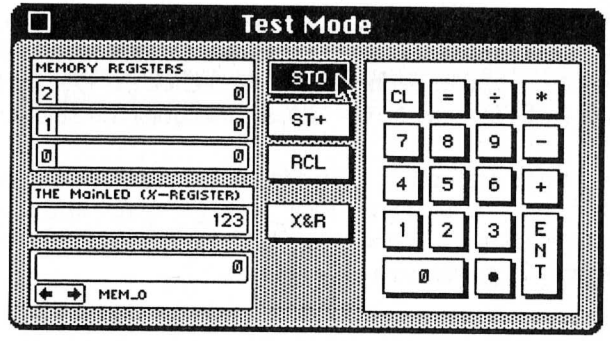
> *This manual often uses an abbreviation to refer to memory registers, like this:* MEM_0, *which means Memory Register #0.*

Unlike Stack Registers, Memory Registers do not roll, lift or drop. In fact, with very few exceptions, they don't change automatically— *you* must execute a function to store numbers to them, or recall numbers from them.

Exit Test mode, and close the 2_*Tutorial* calculator. Open the 3_*Tutorial* calculator. Go back into Test mode.

Storing data to a Memory Register using STO

Key-in: 123. The MainLED (X-Register) displays 123. Let's store the value in the X-Register to MEM_0.

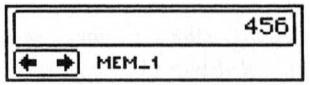


Press the STO key. A dialog box appears on top of the MainLED, displaying a ***prompt*** message (next page).

The STO Prompt Dialog

The left side of this display is the name of the function which is *prompting* you for input, in this case it's the function STO.

A Memory Register number is in the middle part of the prompt dialog. Click the **OK** button, to accept register number 0. The prompt dialog will go away. MEM_0 now contains 123. Just above the MainLED is a SLED wired to display MEM_0, which now shows the value 123. The ***Selector LED*** at the bottom of the calculator is also wired to display a Memory Register, and is showing the contents of MEM_0, which is 123.



Let's store a value to a different Memory Register. Key-in 456. Press the STO key. Again, the prompt dialog appears with register number 0 selected. Type in 1. Note that instead of clicking the **OK** button, you may type the return key on your keyboard.

The SLED wired to display MEM_1 now shows 456, as does the Selector LED. Notice how the text next to the arrows reads MEM_1. This is because the Selector SLED is wired to display the ***Default Memory Register.***

The Default Memory Register is the Memory Register you most recently used. Because we most recently stored to MEM_1, Memory Register #1 is now the Default Memory Register.

Key-in: 789. Press the STO key. Now the prompt dialog assumes you want to store to MEM_1, because it is the Default Memory Register. Type in the number 2, and click the **OK** button. The MEM_2 SLED now shows 789, and so does the Selector LED.

Click once on the left arrow in the Selector LED. See how the SLED changes to display MEM_1. Click again, and the SLED displays MEM_0. The Selector LED designates a new default Memory Register each time you click it.

Recalling data from a Memory Register using RCL

Now let's recall some data from a register. Press the RCL key. The prompt dialog appears again, but this time saying RCL instead of STO. Because we changed the Default Memory Register number using the Selector LED, the prompt is pre-filled with 0. Click the **OK** button, and the MainLED (X-Register) now displays 123; the value stored in MEM_0.

Swapping the X-Register & a Memory Register with XSWAPR

We've learned to store the X-Register to a Memory Register, and recall a Memory Register value to the X-Register. We can also swap the value in the X-Register with the value in a Memory Register. Here's how:

Press the XSWAPR key. You will be prompted for a register number. Type in 2. The X-Register now contains 789, and MEM_2 contains 123. Press the XSWAPR key againto restore the registers, and the X-Register will again contain 123; MEM_2 will contain 789.

Memory Register Arithmetic with STO+, STO-, STO*, STO/

You can also do simple arithmetic using the values in the X-Register and a Memory Register. Let's add the number in the X-Register (123) to the number in MEM_2 (789). Just press the STO+ key. The X-Register does not change, but MEM_2 now contains the result of the addition: 912.

Using three other functions, STO-, STO* and STO/, you can perform subtraction, multiplication, and division on values stored in Memory Registers.

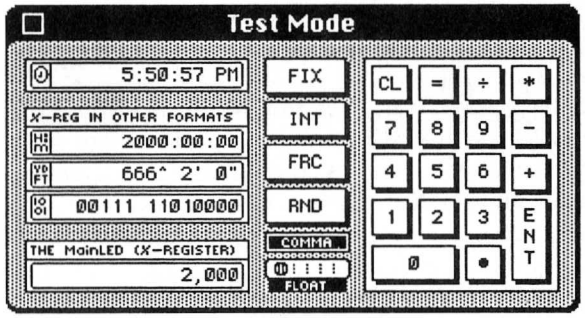
4-Tutorial

Lesson Four: Display Formatting

CCS calculators can display the same number in many different views without changing the underlying number. We call these different views ***display formats***. You may wire SLEDs to display the X-Register in the various display formats.

Exit Test mode, and close the *3_Tutorial* calculator. Open the *4_Tutorial* calculator. Go back into Test mode.

Key in the number: `2000`. The MainLED displays `2,000`. The other displays are not showing the same number. Let's see why:



The SLED at the top of the calculator is wired to display the current time— and does not display the number in the X-Register at all.

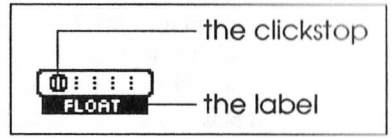
The next SLED is wired to display the X-Register in `HRMIN` format: `2000` hours, `0` minutes and `0` seconds.

The next SLED is wired to display the X-Register in `YDFEET` format: 2000 feet is displayed as `666` yards, `2` feet and `0` inches.

The last SLED is wired to display the X-Register in `BINARY` format: 2000 is displayed in base two (binary) as `111 11010000`. (`BINARY` format inserts a space every 8 characters to make the display more readible. This is because each `1` or `0` represents a bit, and there are 8 bits to a byte).

Input/Display Format

You can also control the format of the MainLED, which is currently displayed in `FLOAT` format. Notice that we have a clickstop switch labeled `FLOAT`. This clickstop switch controls the ***input/display format***, which is the format in which the MainLED displays the X-Register.



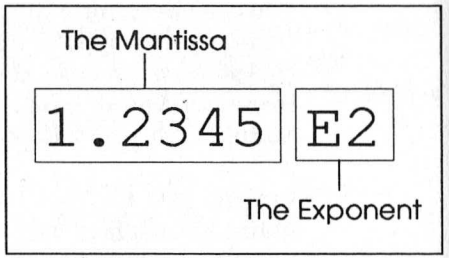
Key-in: `123.456789`. The MainLED displays the number just the way you entered it. Press the clickstop switch. The clickstop advances one position to the right, the label below it changes, and the input/display mode changes to `INTEGER`.

The MainLED redisplays in `INTEGER` format, displaying the number as `123`; no fractional portion. The `.456789` is still there. It just doesn't get displayed in `INTEGER` format.

Press the clickstop switch again. Now we're in `DOLLARS` format, and the MainLED displays the number as `$123.46`. Because the number is being displayed as dollars and cents, we only see two digits to the right of the decimal point. Notice that in the display, the number has been rounded up to `.46` from `.456`.

Press the clickstop switch again, to set the input/display format to `FIXED`. This format displays a fixed number of digits to the right of the decimal point— four digits in this case. So the MainLED displays the number as `123.4568`. Again the last digit has been rounded up in this display format. You can control the number of digits to the right of the decimal point with a function called `FIX`. Press the `FIX` key. A prompt dialog appears, with `4` indicating the current number of decimal places. Type in `9` and click the **OK** button. Now the `FIXED` format will display `9` digits to the right of the decimal point. The MainLED now displays the number as `123.456789000`.

Press the clickstop switch again, and we are in SCI format (Scientific Notation). The MainLED displays the number as 1.2345 E2. In this format, everything to the left of the E is called the *mantissa*, and everything to the right of the E is called the *exponent*. In Scientific Notation, the mantissa is always displayed with no more than four digits to the right of the decimal point. SCI is a sort of short-hand way to display numbers with lots of decimal places. The exponent tells you how many places to move the decimal point. In this case, the exponent says to move the decimal point 2 digits to the right, which would make our number 123.45. If the exponent is negative, you would move the decimal point to the left instead of the right.



Press the clickstop switch back to FLOAT format (Floating Decimal). This format provides you with the clearest view of the underlying number. Most simple handheld calculators display their data in floating decimal format.

How Input/Display Format affects numbers you Key-in

Not only does the input/display format control the way the MainLED is displayed, it also controls which digits are considered valid input.

Press the clickstop switch again, so that we are in INTEGER format. Try to enter 10.5. The calculator beeps when press the decimal point key, because INTEGER format only accepts whole numbers as valid input.

Other input/display formats have similar restrictions. BINARY format only accepts the digits 0 and 1. INTEGER, OCTAL and HEX formats only accept whole numbers. DEGREES, HRSMIN and TIMEDSP formats accept colons as separators (just as DOLLARS, ENG, FIXED, FLOAT, POUNDS and YEN formats accept decimal points as separators between whole and fraction numbers). Chapter 5, and the example calculations in Chapter 7 discuss these restrictions in greater detail.

Changing the Displayed Number (with INT, RND and FRC)

Sometimes, rather than view a number in a different format, you may want to change the underlying number itself.

In FLOAT input/display format, key in: 1234.56. That number in INTEGER format would display as 1234 in the LED, but the underlying number is still 1234.56.

Stay in FLOAT input/display format. Press the INT key. The number is redisplayed as 1234 because the INT function actually changed the underlying number to an integer by throwing away the fractional portion of it.

Key in: 1234.56 again. This time, press the RND key. A dialog will appear in the MainLED, prompting you for the number of decimal places to round to. Enter 0 and click the **OK** button. Rounding to zero decimal places is the same as rounding to an integer value. 1234.56 will be rounded to 1235.

Key in: 1234.56 again. Press the FRC key. This function does the opposite of the INT function. It throws away the integer portion of the number and keeps only the fractional part. The MainLED will show .56.

Turning Commas ON and OFF

Press the clickstop switch so that we are back in FLOAT input/display format. Key-in the number 2000. Notice how the MainLED displays 2,000 even though we keyed-in 2000 (no comma). Click the COMMAS pushbutton. The MainLED now displays 2000; the comma (thousands separator) has disappeared. Whenever the COMMAS pushbutton is highlighted, a comma will be inserted between every third digit to the left of the decimal point. The comma will only be inserted in the "decimal formats:" DOLLARS, ENG, FIXED, FLOAT, INTEGER, POUNDS, SCI, and YEN.

Lesson Five: The Paper Tape

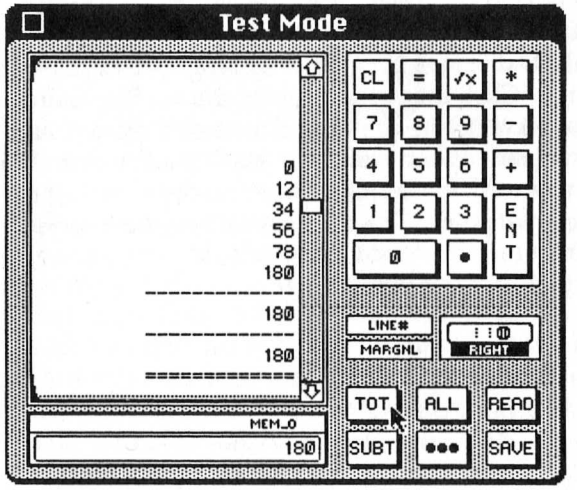
In the previous lessons, we saw how calculation results are usually returned in the X-Register— which is displayed in the MainLED. As you key-in new calculations, the MainLED display changes, and you can no longer view the old results. The Paper tape displays all the data you enter, and can also display the results of your calculations.

Exit Test mode, and close the *4_Tutorial* calculator. Open the *5_Tutorial* calculator. Go back into Test mode. Note that the *5_Tutorial* calculator is configured to do calculations using SAN.

Key-in: 12 + 34 + 56 + 78. Notice how the Paper tape displays each number as it is entered. But unlike the MainLED, the Paper tape doesn't display a running total when an arithmetic operator (+, –, *, ÷) is used in the calculation.

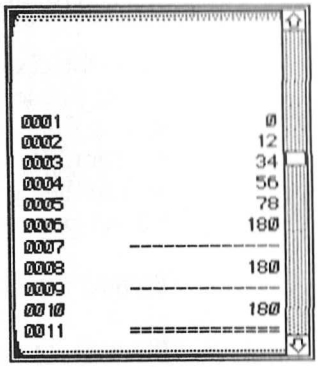
Displaying Totals on the Paper Tape

Press the = key, and a total displays on the last line of the tape. Press the SUBT key; a single line is drawn before the total is displayed. Press the TOT key; a single line, the total, and a double-line are displayed. To insert a visual separator (without a total), press the ••• key— which adds a line of "bullet" characters to the last line of the tape.



Counting a Column of Numbers

Sometimes the Paper tape's lack of totals can be very useful. For instance, when adding a column of numbers, you might want to count how many numbers have been input to make sure they all have been keyed-in. It is easier to count without the subtotals getting in the way.

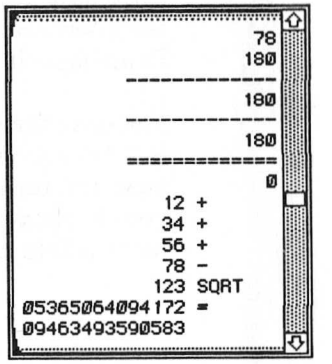


There's an even easier way to count your numbers. Press the LINE# pushbutton. Line numbers are displayed on the left side of the tape. Press the LINE# pushbutton again to turn off the display of line numbers.

Displaying Function names in the Margins

Sometimes, it is useful to have a record of the numbers that were input, ***and*** the functions that were used in the calcuations.

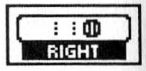
Press the CL key, and a zero appears on the tape. Press the MARGNL pushbutton. Key-in our example calculation again: 12+34+56+78. Notice how a plus sign appears in the right margin next to each number. We call this a ***marginal***. Now key-in –123. A minus sign appears as the marginal.



Press the √x key (to the right of the = key). XTO2 appears as the marginal because that is the function name for square root (Appendix 2 lists all the function names). If the Paper tape was any narrower than 22 characters, the letter F (for function) would appear instead of the function name.

Notice that, unlike the arithmetic operators (+, –, *, ÷), the XTO2 function updates the tape with a subtotal. You can see the entire result of the XTO2 function in the MainLED.

But because we have marginals turned on, we can only see the right portion of the result. There are several solutions to this: always view the results in the MainLED; turn MARGNLS off; stretch the Paper tape a little wider; or change the justification of the text on the tape.



Press the clickstop switch with the label RIGHT under it. Now the switch says LEFT, and the tape's text is drawn flush left— allowing us to view the left side of the XTO2 function's result. Press the switch again, and the text is centered within the tape. A final press on the switch will return us to the RIGHT position.

Editing the Paper Tape: TEXT mode

Now click your mouse on the text in the Paper tape. The cursor changes to the I-beam (shown at left). A blinking insertion point appears in the tape. The label TEXT appears on the Annunciator display (between the tape and the MainLED). We are in TEXT mode.

In TEXT mode, the keyboard is "unmapped" from the keys. That is, typing a keystroke will not invoke the function to which it is normally mapped. For instance, try keying-in: 12+34+56+78=. No calculations are performed as we are in TEXT mode. To return to normal calculation mode, click on the MainLED, or the calculator case. TEXT no longer appears in the Annunciator, and the insertion point has dissappeared. If your cursor is in the Paper tape, it will no longer be drawn as an I-beam.

You can change, cut, copy and paste text to/from the tape while in TEXT mode. Press the ALL key. This selects all the text on the tape, and turns TEXT mode ON. Type the backspace key on your keyboard to clear all the selected text. Click on the MainLED to turn TEXT mode OFF.

Saving and Reading Text Files

You can save the entire contents of the Paper tape to a text file by pressing the SAVEAS key (shown here as SAVE). A standard *SaveFile* dialog box (Figure 1-5) will appear, allowing you to name your file.

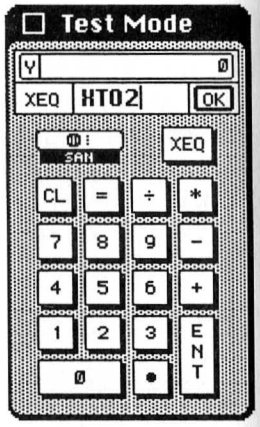
You can also read a text file using the GETAS key (shown here as READ). A standard *GetFile* dialog box (Figure 1-2) will appear, allowing you to open a text file. Once the text file has been read, its contents are displayed on the Paper tape.

6-Tutorial

Lesson Six: The XEQ key
(the most powerful key CCS has to offer)

You do not have to wire functions to keys in order to use the functions. If you know a ***function name***, you can access its function via the XEQ key. Appendix 2 lists the all the XEQable function names.

Exit Test mode, and close the *5_Tutorial* calculator. Open the *6_Tutorial* calculator. Go back into Test mode.



Key in: 10. Press the XEQ key. A prompt dialog will appear on top of the MainLED. Respond to the prompt by typing XTO2, which is the function name for the square of X. After you click the **OK** button, the MainLED will display 100, which is the square of 10.

Press the XEQ key again. This time, type SQRT into the prompt dialog. After you click the **OK** button, the square root function is executed, and the MainLED displays 10 (the square root of 100).

You can also use the XEQ key in lieu of a clickstop switch. Press XEQ, type in RPN, and click the **OK** button. The calculator is switched into RPN mode. You can tell because the clickstop switch has been redrawn to match the RPN input mode.

Key in: 123. Press the XEQ key, respond with BINARY, and click the **OK** button. The MainLED now displays 01111011 because we are in BINARY input/display format. Press the XEQ key again, and respond with FLOAT. We are back in floating decimal format, and 123 is displayed in the MainLED.

and
OK

School's Out!

You've now learned the basics of operating a calculator. Here's where to go next to learn about:

- **Calculator Anatomy** (Chapter 5) provides more detailed information on RPN/SAN input modes, Stack Registers, Memory Registers, display formats, and calculator modes.
- **Calculator Functions** (Chapter 6) is a comprehensive description of all the functions available, along with their function names.
- **Example Calculations** (Chapter 7) picks up where this tutorial ends, by using calculator functions in real-life situations to solve common problems.
- **Program Scripts** (Chapter 8) introduces you to the powerful world of scripting. This chapter assumes no prior programming knowledge and provides its own tutorial for building a script, along with some example scripts.

Chapter 4

About Calculator Construction Set

Iconology: what are all these files for?

CCS provides you with the ability to design and create powerful, fully programmable calculators that may contain a host of higher functions.

CCS is an application that allows you to build custom calculators and save them in three different kind of files:

- ***Work files*** can only be opened and edited by CCS. If you ever want to change a calculator, you will need its work file to open and edit. Whenever you save your calculator as any kind of file, a work file is also saved automatically. We have included many work files with CCS. They contain "pre-built" calculators which may be saved as ***finished calculators*** and used immediately, or use them as a starting point and alter to suit your taste.

- ***Font/DA Mover files*** are ***finished calculators*** which are ready to be installed in your System file using the *Font/DA Mover* application. Sometimes referred to as "suitcase" files, they cannot be edited by CCS. You may open these files directly from *Suitcase* or *Font/DA Juggler* if you own either of these utilities.

- ***Calculator Application files*** are ***finished calculators*** which do not have to be installed at all. To use these calculators, simply double-click their icons from the *Finder*, or from *Multifinder*. Finished calculator application files cannot be edited by CCS.

Files created by finished CCS calculators:

The finished calculators created by CCS can themselves create several kinds of files:

- ***Text Files*** can be created, opened, and edited by CCS calculators in CCS's Test mode, and by most word processors, or text editors. Because *TeachText* is an application that comes with the Macintosh, CCS calculators save their text files as *TeachText* documents. If you double-click the text file from the *Finder*, the *TeachText* application will open and display the contents of the document. See Chapter 6 for more on opening and saving text files using finished calculators.

- ***Script files*** can only be created, opened, and edited by finished CCS calculators, and in CCS's Test mode. A script contains a list of function names (or instructions) which the calculator will execute. See Chapter 8 for more on calculator scripts.

- ***Memory Register files*** can only be created and read by finished CCS calculators, and in CCS's testmode. These files contain the contents of a calculator's Memory Registers (up to 255). See Chapter 5 for more on Memory Registers.

Important We urge you to make backup copies of your *Calculator Construction Set* disk(s) before using the software. Please store your original disk(s) away, and work off the copies.

The CCS desktop

Open CCS by double-clicking the *CCS2* application's icon from the *Finder* or from *Multifinder*. You may also double-click a CCS work file, which will open the work file along with the *CCS2* application.

After CCS is open, you will see the CCS desktop (which may look different from this depending upon what kind of screen you are using):

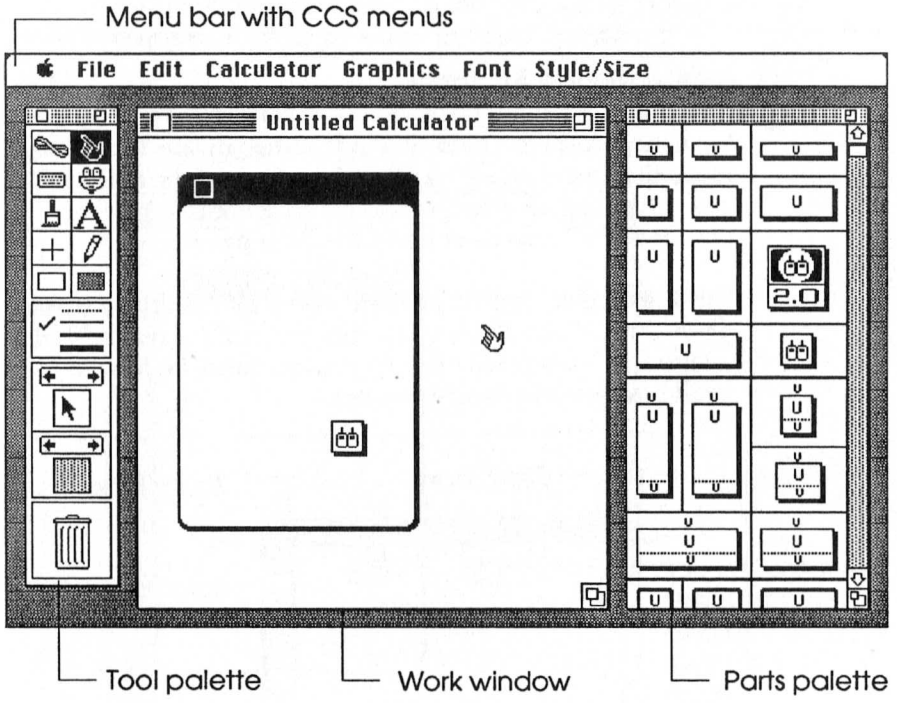


Figure 4-1 The CCS desktop as seen on a Macintosh Plus

Note You may drag CCS windows by their title bar. Or hold down the command key (⌘), and drag the window by clicking on any part of the window.

What are the windows for?

CCS has three windows: the ***Tools*** palette, the ***Work*** window, and the ***Parts*** palette (see Figure 4-1).

You construct a calculator by dragging ***part shapes*** from the Parts palette to the Work window. Use the tools in the Tools palette to assign functions and keystrokes to parts, and to stretch, label and decorate the parts.

You save calculators using the ***Save As...*** item in the ***File*** menu.

See the tutorial in Chapter 1 for complete, step-by-step instructions on building a calculator from scratch.

Palette Window Anatomy

A palette is a special kind of window that always that "floats" above document windows. Its titlebar, grow box, and scrollbar look different than other windows. CCS has two palette windows: the Tools palette and the Parts palette.

Clicking a palette's close box will hide a palette (show it again using items on the ***File*** menu). See the sections on the Tools palette and Parts palette for more information on how to use the zoom box, scroll bar and grow box.

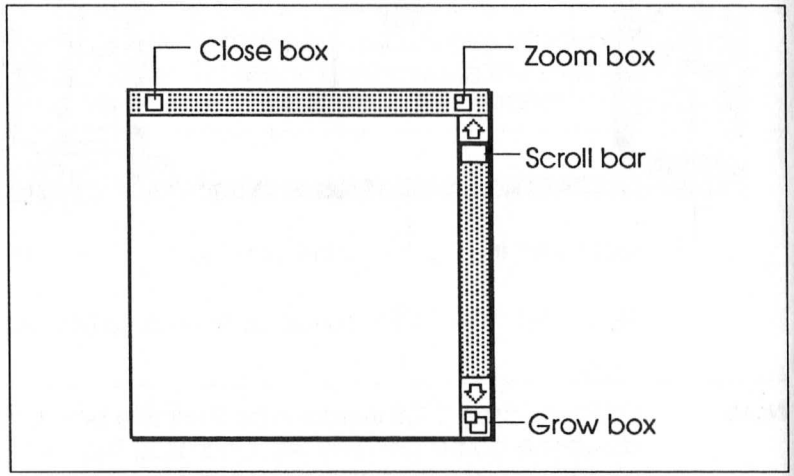


Figure 4-2 Features of a palette window

The Work window

The ***Work window*** is where the calculator you are building appears. To add a part to your calculator, drag the part from the Parts palette to the Work window.

The ***Calcshell*** (see Figure 4-3) is the size that the finished calculator's window will be when saved. Parts may be placed anywhere within the Work window, but only those parts placed on the Calcshell will actually draw in the finished calculator. We refer to parts outside the Calcshell as ***invisible parts***. If the invisible parts are mapped to a keystroke, you can invoke the part's function even though the part will be invisible (keystroke mapping is explained later in this chapter).

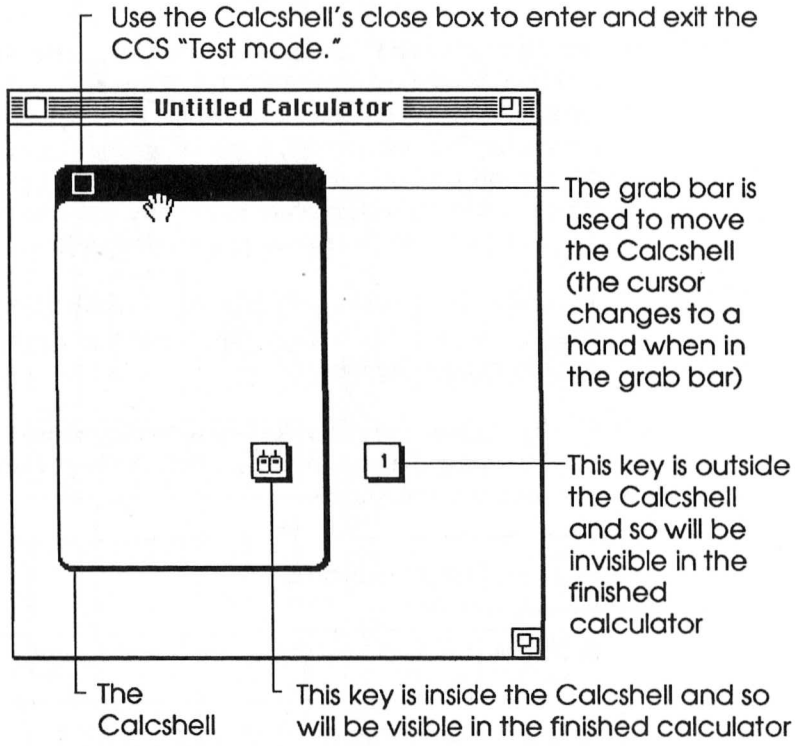


Figure 4-3 The CCS Work window (and Calcshell)

The Parts palette

Zooming and resizing only changes the Part palette's height.

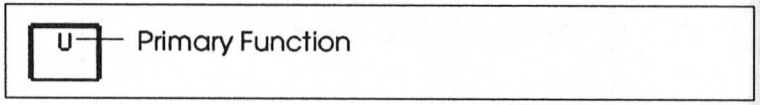
Most CCS parts reside in the Parts palette, and are referred to as ***calculator parts***. To add a calculator part to your calculator, drag the part from the Parts palette to the Work window. Note that calculator parts may not overlap each other in the Work window.

Most parts in the Parts palette are ***unwired***— they have no function assigned to them yet. Use the Plug tool to ***wire*** (assign functions to) the parts. The Plug tool is explained in the Tools palette section of this chapter. Figures 4-4 and 4-5 show all the part shapes contained in the Parts palette. Here is a description of each...

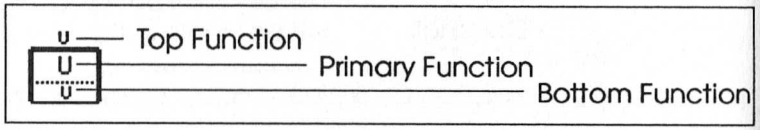
- **Single and Multi Function Keys.** These are the most common kind of part used in CCS. Keys are used to hold functions, and even program scripts. During construction, after a key has been wired, it can be ***mapped*** to the Macintosh keyboard using the Keyboard tool, or even ***decorated*** using the graphics tools. You can also paste a graphic onto a key by selecting a single key first.

  In finished calculators, you generally invoke the key's function in one of two ways, which this manual will later refer to as ***pressing*** the key:

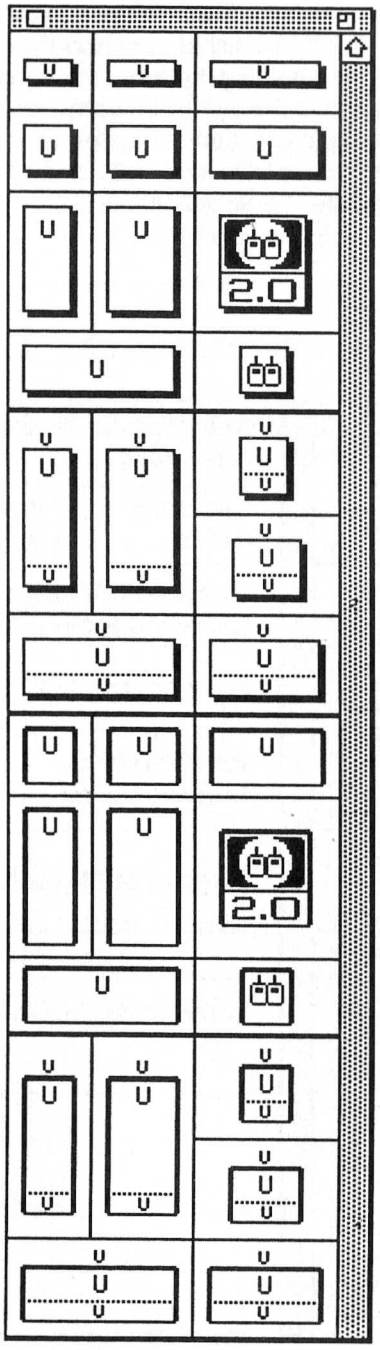
  (1) by ***clicking*** the onscreen key with the mouse
  (2) by ***typing*** the keystroke to which the key was mapped during construction.



A Single Function key (holds only one function)



A Multi Function key (holds up to three functions)



Single function keys may only be wired to a single function.

This section contains single function keys with **square corners** (and heavy drop-shadows).

Multi function keys may be wired with up to three functions on a single key.

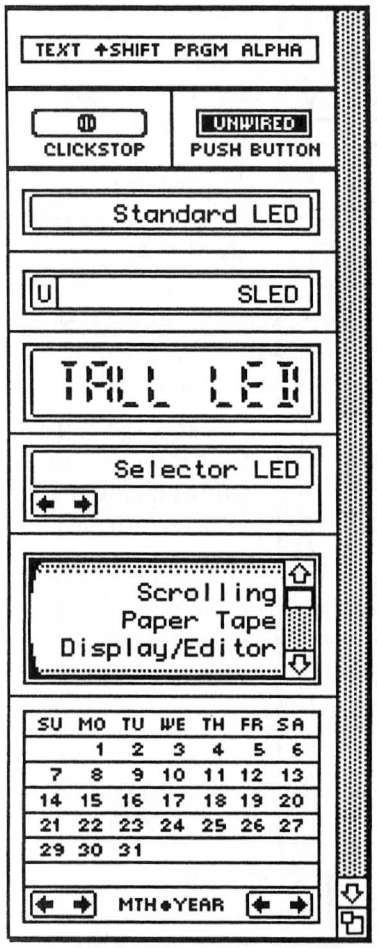
This section contains multi function keys with **square corners.**

This section contains single function keys with **round corners** (and no drop shadows).

This section contains multi function keys with **round corners.**

Figure 4-4 Unwired keys in the Parts palette

There are two different kind of keys: ***Single function*** keys and ***Multi function*** keys (discussed earlier in this chapter). Multi function keys can hold up to three functions in the same key. You must turn the TOP or BOT shift modes *ON* in order to access any key's function other than the PRIMARY one (the one in the center of the key). TOP and BOT shift modes are explained in Chapters 5 and 6.



Annunciator display (prewired)

Clickstop Switch | **Pushbutton** Switch

Standard LED display (prewired Main LED)

Supplemental (SLED) display

Tall LED display (prewired Main LED)

Selector SLED display

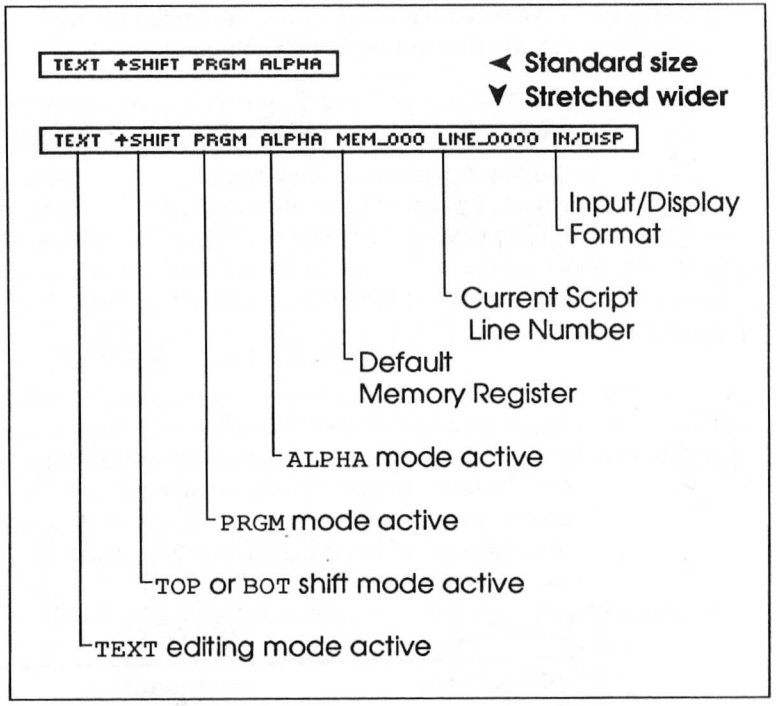
Paper tape display (prewired)

Calendar display (prewired)

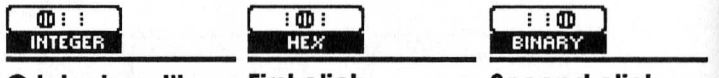
Figure 4-5 Displays and switches in the Parts palette

❖ **Annunciator display** helps you keep track of calculator modes and settings. The Annunciator will display information about TEXT, SHIFT, PRGM, ALPHA and Input/Display formats, as well as the numbers of the ***Default Memory Register*** and the ***Current Line number.***

Modes and Memory Registers are all explained in Chapter 5. The Current Line Number is explained in Chapter 8.



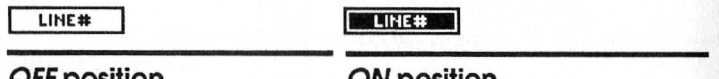
❖ **Clickstop switches** control calculator modes which have more than a simple *ON/OFF* state (see Chapter 6 for a list of all clickstop switch functions). Pressing the switch will advance the position of the clickstop one position to the right (Figure 4-6). Pressing a switch set at the rightmost position resets the clickstop to the first position. Each clickstop position displays a different label beneath the switch, which may be edited using the Text tool. The label is inverted (white letters on black background) when the switch setting matches the mode it controls. Switches may be mapped to keystrokes during construction.



Original position	**First click**	**Second click**
Third click returns to first stop	advances to second stop	advances to third stop

Figure 4-6 A Clickstop switch in Action

❖ **Pushbutton switches** are a special kind of key used to toggle modes *ON* and *OFF*. When a mode is *ON*, the pushbutton key will appear inverted (see Figure 4-7). Pushbutton switches can be mapped to keystrokes during construction. Pushbuttons can be wired for a variety of purposes, all of which are listed in Chapter 6.



***OFF* position**	***ON* position**
The Line Numbers are turned *OFF*	The Line Numbers are turned *ON*

Figure 4-7 A Pushbutton switch in Action

Note While in the Work window, stretchable display parts are drawn with a numeral on the right side of the display. This numeral represents the maximum number of characters that will fit in the display of a finished calculator.

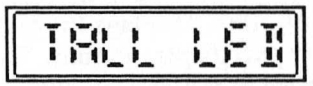
❖ **LED Displays** are used to display data, which is stored in registers (see Chapter 5 for more on registers). We use the term *LED* because the early handheld calculators used components called *Light Emitting Diodes* as their displays.

There are several kinds of LEDs in the CCS Parts palette. See figure 4-5 for pictures of each.

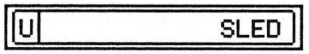
❑ *MainLED* is generally used to display the result of a given calculation. In ALPHA mode, the MainLED will display the contents of the Alpha Register. If a function needs to prompt you for more data, the prompt will appear in the MainLED. If an error occurs during an operation, the error message will also be displayed in the MainLED.

There can only be one MainLED per calculator, but there are two kinds of MainLED shapes available: the ***Standard*** and ***Tall LEDs*** (explained below).

❖ **Standard LED** always draws text in the 9 point Monaco font, and can hold from 17 to 24 characters.



❖ **Tall LED** draws text in the font/size/style you select during construction (discussed later in this chapter), and can hold from 1 to 24 characters. If you do not stretch the Tall LED wide enough, you may not be able to see the entire result of a calculation or error message. It is best to stretch the Tall LED so it will display a minimum of 17 characters.



- **SLED (Supplemental LED)** These can be distingished from standard LED shapes by the small box on the left side of the display (you may set the character in this box to any single character that you wish, or use the graphics tools to draw within this small box). SLEDs can be wired for a variety of purposes including:

  - ❑ *Display of Stack Registers** The format of the displayed Stack Register mirrors the currently active format of the number displayed in the MainLED.

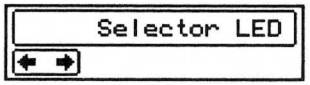
  - ❑ *Display of the Alpha Register** If the SLED does is not wide enough to display all 24 characters of the Alpha Register, you will only see the rightmost portion of the Alpha Register.

  - ❑ *Display of MainLED in alternative format.* The number displayed in the Alternative Format SLED will be rounded as appropriate for the format selected. However, the number as actually stored in the calculator will not be changed. If the MainLED contains a Prompt or is displaying the Alpha Register*, the Alternative display SLED will still display the number contained in the X-Register.

  - ❑ *Display of a Memory Register** You must wire this SLED to indicate which Memory Register it will always display. Memory Register SLEDs display their contents using the same display format as the MainLED.

  - ❑ *Display of the current time.* Displays the current time despite the current input/display format. The time will be displayed in the format of the country for which your System has been *localized* (set up for). During construction, you may select a time zone for this SLED to display. Other CCS Date/Time functions will not affect this display.

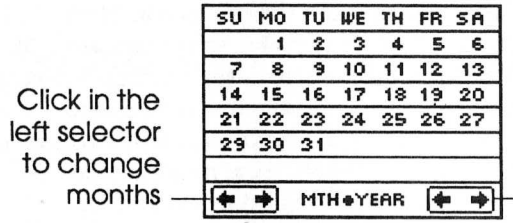
* *Registers are explained in Chapter 5.*



- **Selector LEDs** have arrows below their displays which allow the display to be scrolled. Selector LEDs can be wired for a variety of purposes including:

  - ➡ *Memory Register Display* allows you to scroll through all the Memory Registers, one at a time. Each time you select a new Memory Register, it becomes the *Default Memory Register* (See Chapter 5 for more).

  - ➡ *Binary Display* allows you to scroll through all 4 bytes of the X-Register as displayed in binary format. Since binary words take up 18 characters in the MainLED, you would not be able to view an entire long word unless your MainLED was stretched to display 36 characters.



- **Calendar Display** displays a monthly calendar. In finished calculators (or in Test mode), click the arrows below the calendar to select the month and year you wish to view. Click the `MTH•YEAR` text between the arrows to display the current month and year.

❖ **Paper tape Display** is used for many things. You can think of the Paper tape as a window within the finished calculator's window. Normally, the Paper tape displays a list of all the calculation results which were displayed in the MainLED.

If you click in a finished calculator's Paper tape, you may edit the text displayed there. We call this TEXT mode (see Chapter 5 for more on modes). In TEXT mode, a blinking insertion point will appear. Whenever the mouse is in the tape, the cursor changes to an I-beam (shown in margin). Click anywhere in the calculator's window (except the paper tape) to turn TEXT mode *OFF*.

In PRGM (program script recording) mode, the Paper tape displays a listing of the script you are editing. Click anywhere in the calculator's window (except the Paper tape, or a function part which reacts to mouse clicks) to turn PRGM mode *OFF*. A click on a part wired to a function will insert the part's function name into the tape. See Chapter 8 for more on this mode and script editing.

You can read a text file in from disk to display the file's contents on the Paper tape, where you may edit it. You may save the Paper tape's contents to a disk file as well. The Paper tape can also be printed out on your printer.

The Tools palette

The top section of this palette contains tools. The middle sections contain selectors. The bottom section is a trash can. Click on a tool icon to select that tool. There is always one selected tool, and its icon is inverted (white icon on black background). Figure 4-8 pictures the Magic Hand as the selected tool. When the mouse is in the Work window, the cursor's shape will correspond to the selected tool.

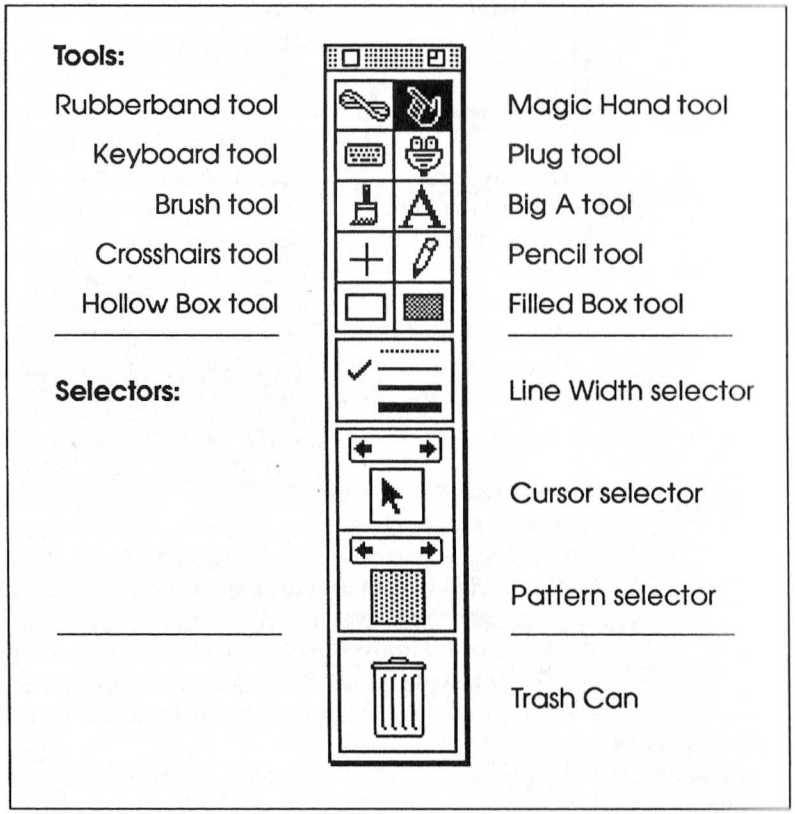


Figure 4-8 The Tools Palette

The Tools palette does not have a scroll bar, or grow box. Clicking the zoom box will toggle the size of the Tools palette between the size shown in Figure 4-8, and a smaller size (which hides the Cursor selector, Pattern selector and Trash Can).

❖ **Rubberband tool** (Calcshell Stretching) allows you to stretch the Calcshell larger or smaller (Figure 4-3). When you first select the Rubberband tool, a band is drawn around the Calcshell. Click inside this band to stretch the Calcshell.

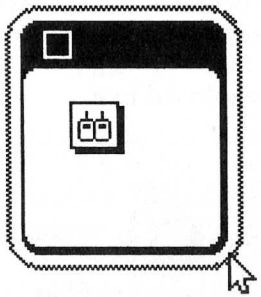


Figure 4-9 The cursor changes to a hollow arrow when the cursor is within the band which surrounds the Calcshell. You can stretch only while the cursor appears as the hollow arrow.

Dragging a corner of the band stretches the band in two directions at the same time. Dragging any other part of the band only stretches in a single direction.

Double-clicking the Rubberband tool icon in the Tools palette will shrink the Calcshell to the smallest size that will contain all parts in your calculator, then position the calculator in the upper left corner of the Work window.

❖ **Magic Hand tool** (Part Selecting, Dragging and Stretching) To *select* a part, click on it. To select another part, shift-click on it. Toggle part selection by shift-clicking. Unselect all parts by clicking on no parts at all. Select a group of parts by clicking on no parts at all and then dragging; a box will start drawing (see Figure 4-10). Any part intersected by the box will be selected (Figure 4-11 shows selected parts). *Double-clicking* the Magic Hand's icon will select all parts.

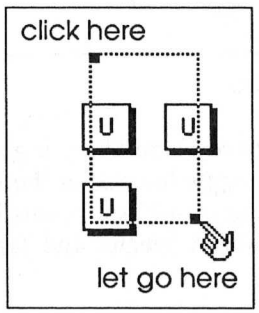


Figure 4-10 Selecting more than one part. Click on no parts at all and start dragging without letting go of the mouse button. The Magic Hand draws a rectangle. Any part intersected by the rectangle will be selected when you let go of the mouse button.

Magic Hand *continued*

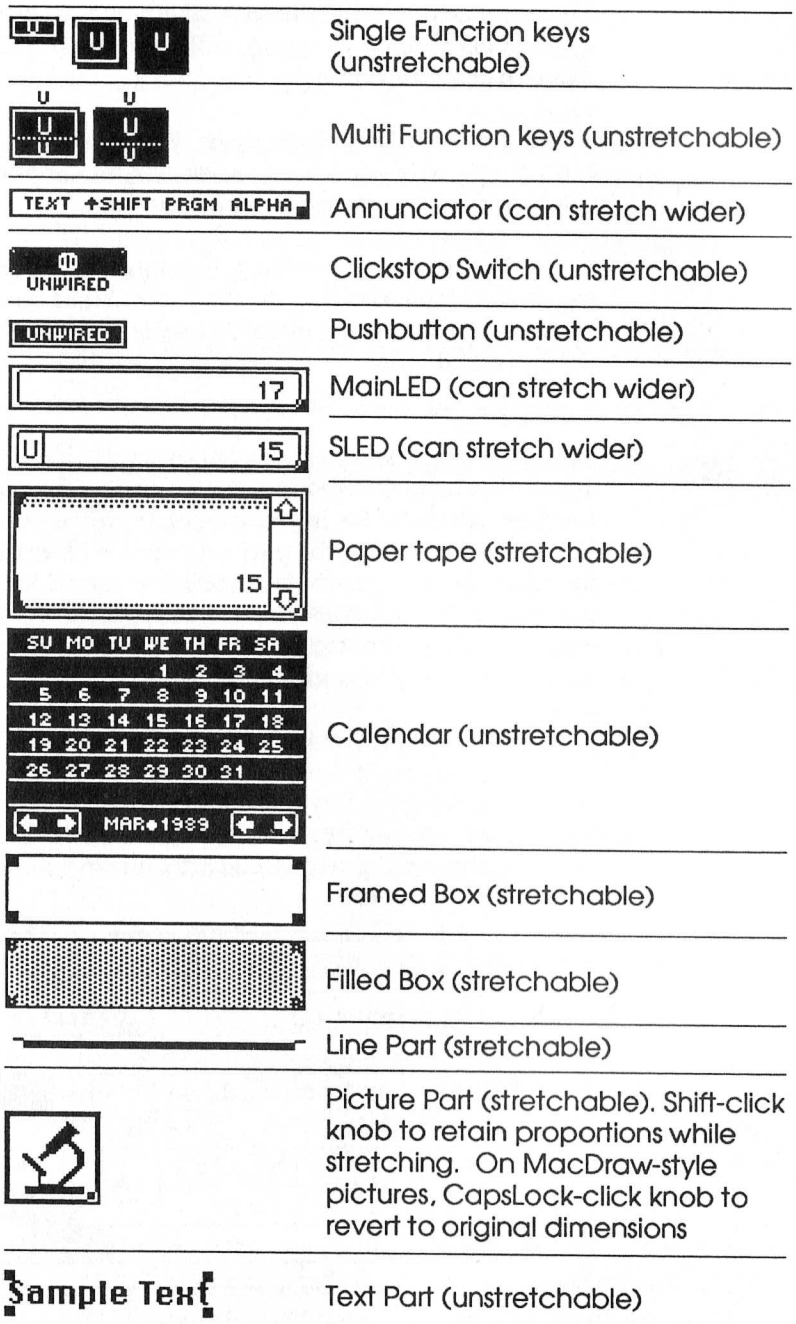


Figure 4-11 Parts as they would appear in the Work window if selected

Magic Hand *continued*

Click on a selected part and drag the mouse to ***drag*** all selected parts. Shift-dragging will constrain the drag to move in one direction only.

Not all parts can be ***stretched*** (see Figure 4-11 to see which can). Stretchable parts have "knobs" in one or more corners. Click on a knob and drag to stetch the part.

Power Users: hold down the option key to temporarily toggle the Magic Hand to the Plug tool. Hold down the option+shift keys to temporarily toggle the Magic Hand to the Keyboard tool.

❖ **Keyboard tool** (key mapping) allows you to assign keystrokes to an onscreen calculator part. When using your finished calculator (or in Test mode), typing that part's keystroke will invoke the part's function. Note that multi function keys have a single keystroke mapped to them. The TOP and BOT modes determine which function will be executed when a keystroke is typed. (See Chapters 5 and 6 for more on the TOP and BOT modes).

Figure 4-12 The unmapped icon on a key

Double-clicking the Keyboard tool's icon will redraw all the parts using mapped keystrokes instead of the usual keycap decor. Unmapped parts will be drawn using the unmapped icon (see Figure 4-12).

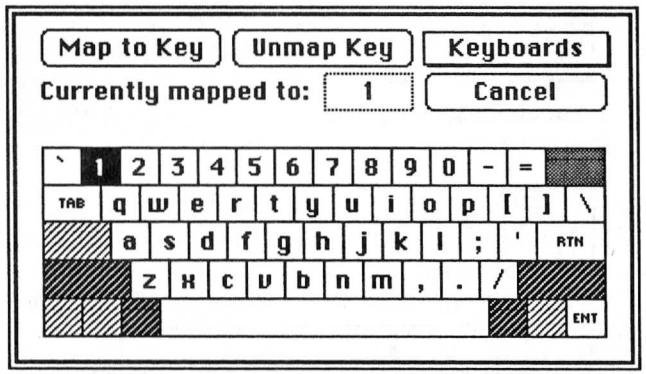


Figure 4-13 The Map to Keyboard dialog box

Keyboard tool *continued*

To map keystroke to a part click on the part using the Keyboard tool. The ***Map to Keyboard*** dialog box (Figure 4-13) will appear.

The ***Map to Keyboard*** dialog box draws keys in a unique way to indicate whether the key is mappable, unmappable, already mapped, or the current selection (see Figure 4-14). To map a keystroke to the calculator part, either click a key with the mouse, or type a character with the keyboard.

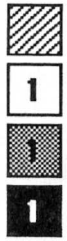


Figure 4-14 Map to Keyboard pattern legend

- ❑ The ***Map to Key*** button will map the current selection to the calculator part and close the dialog box.
- ❑ The ***Unmap Key*** button will map no keystroke to the calculator part and close the dialog box.
- ❑ The ***Cancel*** button will close the dialog box without changing the keystroke mapped to the calculator part.
- ❑ The ***Currently Mapped To*** text displays the keystroke the calculator part was mapped to when the dialog appeared.
- ❑ The box titled ***Keyboards*** is a pop-up menu unless you are using System version 4.0 or earlier (lower number). This menu (if present) allows you to select a picture of a different keyboard. You must have the *Key Layout* file in your *System Folder* to activate this menu. The *Key Layout* file can be found in the *Utilities* folder of the *CCS 2.0* disk.

The first time this dialog box appears, the dialog will draw a picture of the keyboard you have connected to your Macintosh. Otherwise, a generic keyboard will be drawn.

❖ **Plug Tool** (wiring) is used to assign functions to parts. We call this task *wiring*. Some parts cannot be wired: text, framed boxes, filled boxes, and picture parts. Other parts come pre-wired: MainLEDs, Annunciators, and Calendars.

To wire a part, click on the part with the Plug tool. Depending upon the shape of the part you clicked, a different dialog box will appear...

Logo Key Wiring

If you wire a *logo key* (a key with the Dubl-Click logo on it), the dialog box in Figure 4-15 will appear. This allows you to set the message that appears in a dialog box when the logo key is pressed in a finished calculator.

There are four different logo key shapes (shown at left). Each wired logo key, regardless of its shape, will have a unique message.

The *Flush Left*, *Centered*, and *Flush Right* radio buttons determine how the text will be drawn.

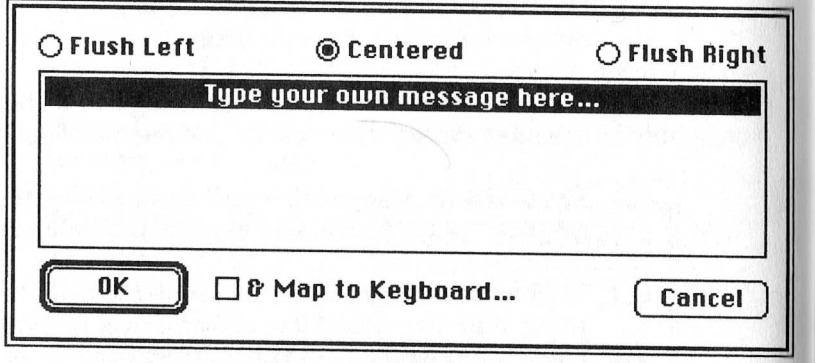


Figure 4-15 Logo Key Message Wiring dialog box

Plug tool
continued

Single and Multi Function Key Wiring

If you wire a ***single or multi function key*** (examples in the margin below), the dialog in Figure 4-16 will appear.

Select the desired function from the list on the left side of the dialog box. ***To change lists***, click on the pop-up menu above the list (shown in Figure 4-11 as "Constants"). If you are using an older *System* file, this dialog box will not have a pop-up menu—instead you will find a group of radio buttons at the bottom of the dialog box. Click on one of these radio buttons to change lists.

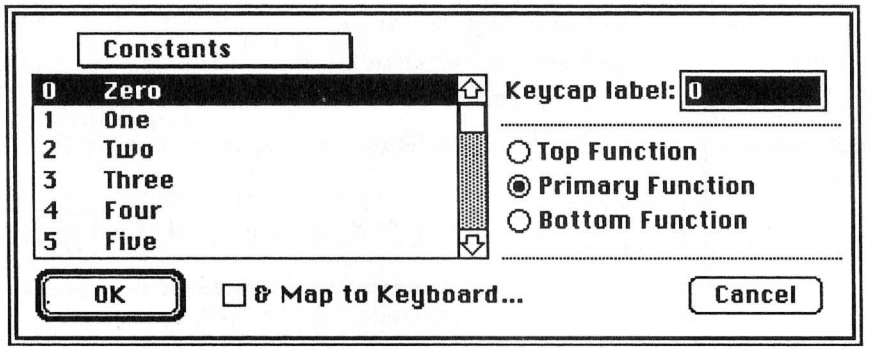


Figure 4-16 Single & Multi function key wiring dialog

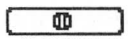
On single function keys, the ***Top Function*** and ***Bottom Function*** radio buttons will be grayed out. On multi function keys, when you click on the ***Top Function***, ***Primary Function*** or ***Bottom Function*** radio buttons, the function currently wired to that position of the key will be highlighted. If that position has not been wired yet, the last function selected will remain selected.

You may also type in the ***Keycap label*** that will be drawn on the key. Later, if you wish to change the keycap label, you may do so via this dialog box, or by clicking the key with the Text tool.

Clicking the ***Cancel*** button will not change the key's current wirings. Clicking the ***OK*** button will rewire the key to the settings you changed in this dialog box. If the ***& Map to Keyboard...*** box is checked, clicking ***OK*** will subsequently bring up the ***Map to Keyboard*** dialog box (Figure 4-8).

Clickstop Switch Wiring

Plug tool
continued



If you wire a ***clickstop switch*** (shown in margin), the dialog in Figure 4-17 will appear. Click a radio button to select the type of clickstop switch you want to wire.

Clicking the ***Cancel*** button will not wire the switch. Clicking the ***OK*** button will wire the switch as the type you selected. If the ***& Map to Keyboard...*** box is checked, clicking ***OK*** will next display the ***Map to Keyboard*** dialog box (Figure 4-13).

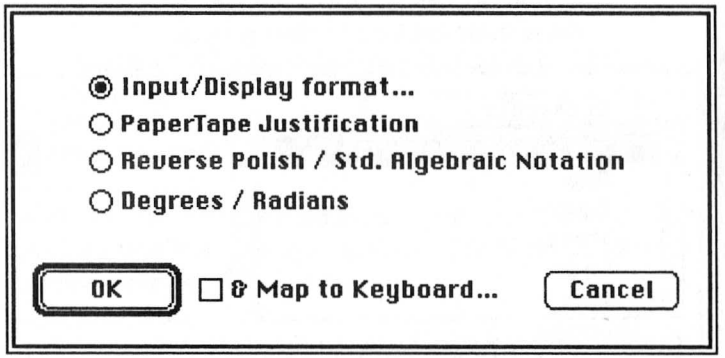


Figure 4-17 Clickstop Switch wiring dialog

If you selected ***Input/Display format...*** as the switch type, a second dialog box will immediately appear (Figure 4-18) allowing you to wire individual "clickstops." The radio buttons (labeled 1 through 5) select which clickstop you are wiring. Select the clickstop setting from the list on the left side of the dialog box. The text to the right of the clickstop radio button will appear beneath the clickstop switch in your calculator.

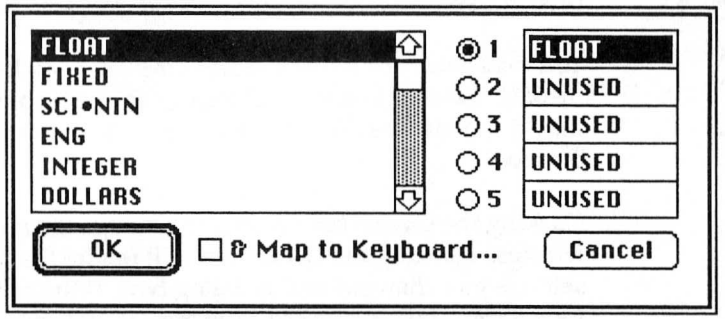


Figure 4-18 Input/Display Format clickstop wiring dialog

Pushbutton Switch Wiring

Plug tool
continued

If you wire a ***Pushbutton switch*** (shown in margin), the dialog in Figure 4-18 will appear. The ***Pushbutton Switch Wiring*** dialog works very much like the dialog shown in Figure 4-16. However, because there is only one list of pushbutton functions, there is no menu of lists in this dialog.

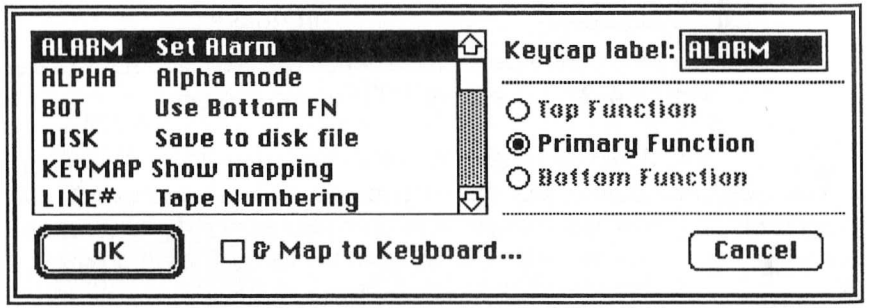
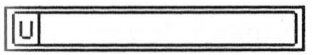


Figure 4-18 Pushbutton Switch Wiring dialog

SLED Display Wiring



If you wire a ***SLED*** (shown above), the dialog in Figure 4-19 will appear. If you wire the SLED to be either a Continuous Time display or a Memory Register display, a subsequent dialog box will appear (Figures 4-20 and 4-21) to prompt you for either a time zone or register number, respectively.

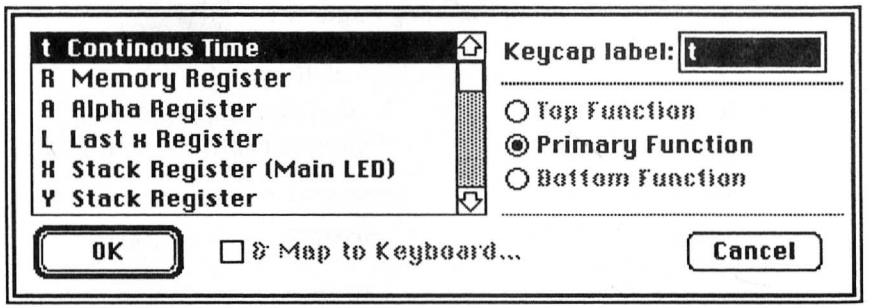


Figure 4-19 SLED wiring dialog

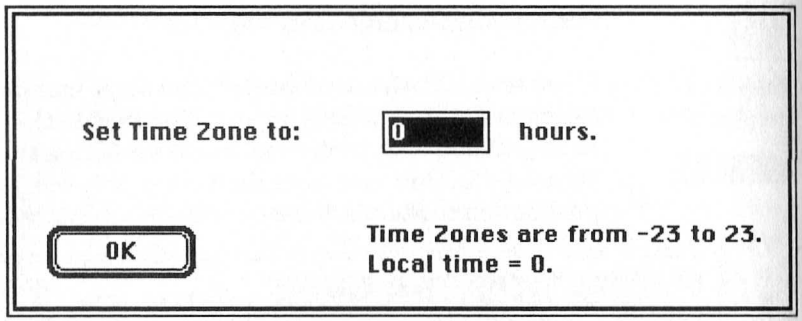


Figure 4-20 Time Zone SLED wiring dialog

Plug tool
continued

When you *rewire* either a Continuous Time display or a Memory Register display, the dialogs (Figures 4-20 and 4-21) will appear instead of the SLED wiring dialog (Figure 4-19). You must use the Text tool to change the keycap label for these two kinds of SLEDs.

Time Zone #0 is what we call *local time*, or the current time in your local time zone. All other zones are calculated in relation to local time. For example, a time zone of 3 will display the time three hours later than local time.

Memory Registers are described in Chapter 5. You may wire up to 256 Memory Registers, which are numbered as `MEM_0` through `MEM_255`.

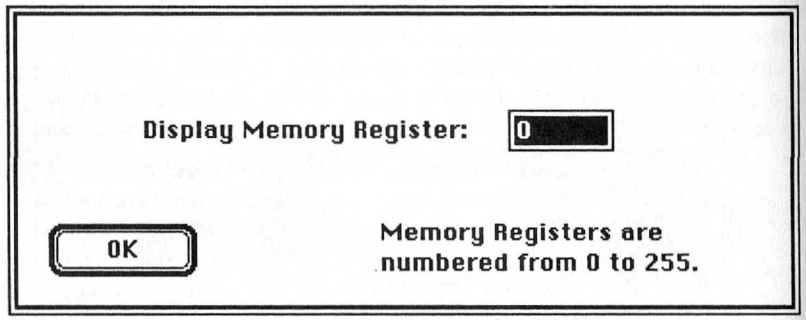


Figure 4-21 Memory Register SLED wiring dialog

❖ **Brush tool** (painting) is used to edit pictures on keycaps, and *graphic* parts (graphic parts are pictures that you paste into the calculator using the **Edit** menu, or create with the Text, Framed Box or Filled Box tools). The brush paints using the currently selected pattern (see the Pattern Selector later in this section). Hold down the option key to always use a white pattern; effectively turning the brush into an eraser.

Double-click the Brush tool's icon to edit/select paint brushes using the ***Edit/Select Brush*** dialog (Figure 4-22).

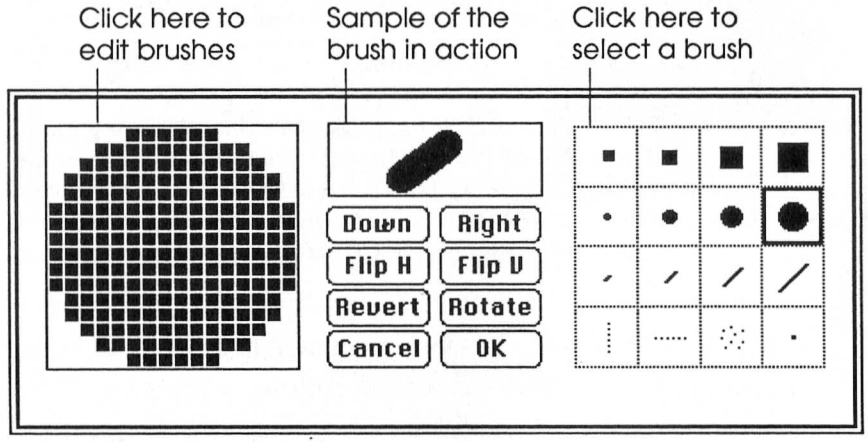


Figure 4-22 Edit/Select Brush dialog box

❖ **Text tool** (typing) can be used to create graphic parts, to type on graphic parts, or to change keycap labels.

If you click the text tool on a key, clickstop switch, pushbutton switch, or SLED in the Work window, the ***Change Keycap Label*** dialog box will appear (Figure 4-23). Changing the keycap label in this dialog box will not affect the function(s) wired to the part.

If you click the text tool on an ***Input/Display Format*** clickstop switch, the ***Input/Display Format*** clickstop wiring dialog will appear (Figure 4-18). It is possible to change both clickstop wiring and the switch labels in this dialog box.

Text tool
continued

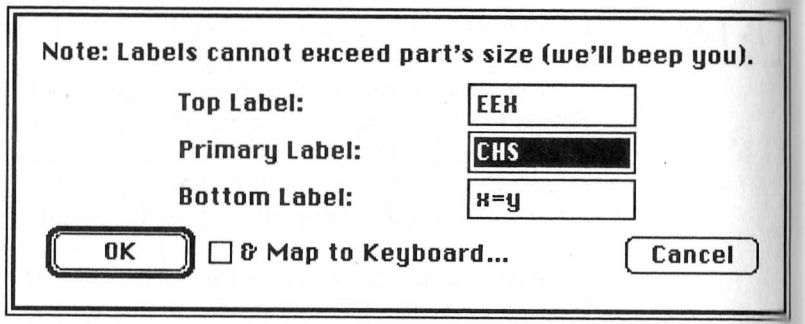
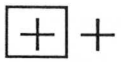


Figure 4-23 Change keycap label dialog box



❖ **Crosshairs tool** (line drawing) can be used to create graphic parts, or to draw on keycaps and other graphic parts. To create new graphic parts, start dragging this tool in the Work window. You may later use the Magic Hand to move and stretch graphic parts. To draw on existing parts, click the crosshairs cursor inside the existing part before dragging.

❖ **Pencil tool** (drawing and fatbitting) can be used to draw on keycaps and other graphic parts.

Double-clicking the Pencil tool's icon toggles the ***Fatbits*** menu setting (in the ***Graphics*** menu). If ***Fatbits*** is checked, clicking the pencil tool on a part will display the ***FatBits*** dialog box (figure 4-24). Holding down the ⌘ key and clicking on a part with the Pencil tool also displays the ***Fatbits*** dialog box.

The ***Fatbits*** dialog box has two tools of its own: a copy of the pencil tool, and a grabber hand tool. Click on the tool icons to change tools. Or, if you prefer, hold down the option key to temporily toggle between the two tools.

While in fatbits, the grabber hand tool will allow you to scroll around the fatbits image should the entire image not fit within the dialog box.

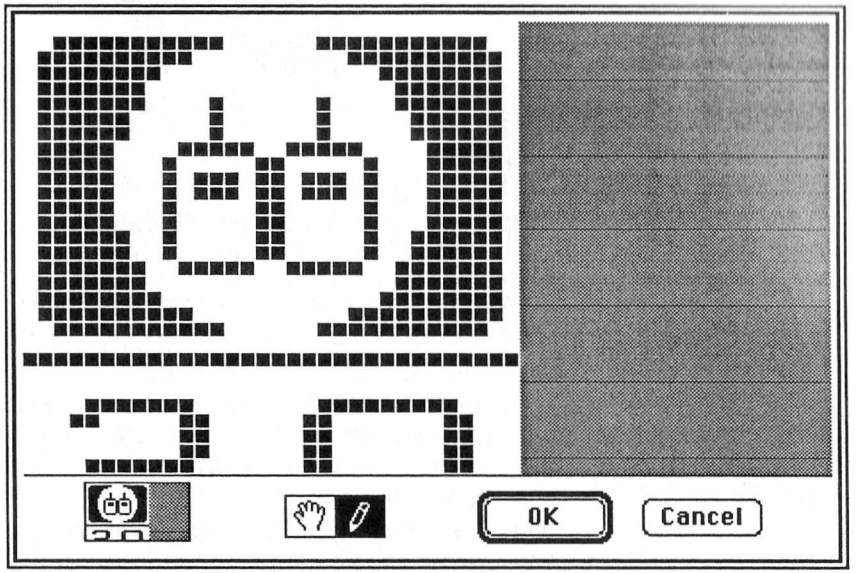
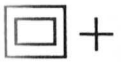
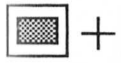


Figure 4-24 The Fatbits editing dialog box



❖ **Framed Box tool** (box drawing) can be used to create graphic parts, or to draw on keycaps and other graphic parts. To create new graphic parts, start dragging this tool in the Work window. You may later use the Magic Hand to move and stretch these parts. To draw on existing parts, click inside the existing part. This tool does ***not*** fill the boxes it creates with a pattern.



❖ **Filled Box tool** (box drawing) is exactly the same tool as the Framed Box tool, except that it fills the boxes it draws using the selected pattern (see the ***Pattern Selector*** explained later in this chapter). Both box drawing tools also draw their borders using the selected line weight (see the ***Line Weight Selector*** explained later in this chapter).

Double-clicking either box drawing tool icon will display the ***Select Roundness*** dialog box (Figure 4-25).

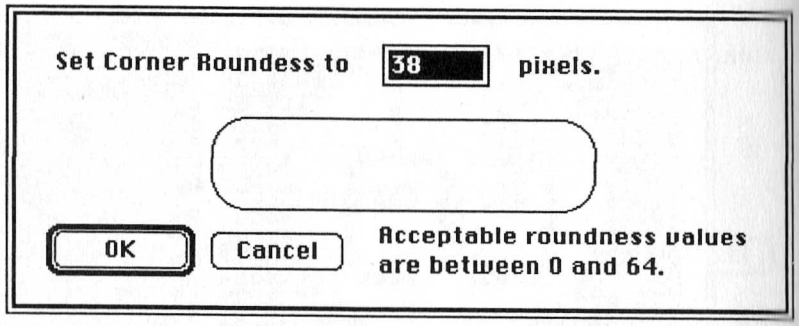


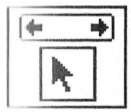
Figure 4-25 The Corner Roundness dialog box

Box tools
continued

Use the ***Corner Roundess*** dialog box to draw boxes with (guess what?) round corners. A value of zero will yield a square cornered box (so, we lied a little). When you click the *OK* button, any selected boxes in the Calcshell will immediately redraw using the new corner roundess setting. You may pull down the ***Graphics*** menu to inspect the current value set for corner roundness without bringing up this dialog box.

Using Selectors

❖ **Line Width Selector** controls the line weight of graphic parts created by the Crosshairs, Framed Box and Filled Box tools. Click on the weight desired— if any line or box parts are selected, they will immediately change to the newly selected weight. The weight represented by a dotted line is no weight at all (valid for filled boxes only). Any new lines and boxes will be formed using the selected weight.



❖ **Cursor Selector** lets you cycle through eight different cursors by clicking the arrows at the top of the selector box. Click on the cursor to edit it. The ***Edit/Select Cursor*** dialog box will appear (Figure 4-26).

The selected cursor will be saved with the finished calculator and will be used whenever the mouse is in the installed calculator's window. All eight cursors will be saved with work files.

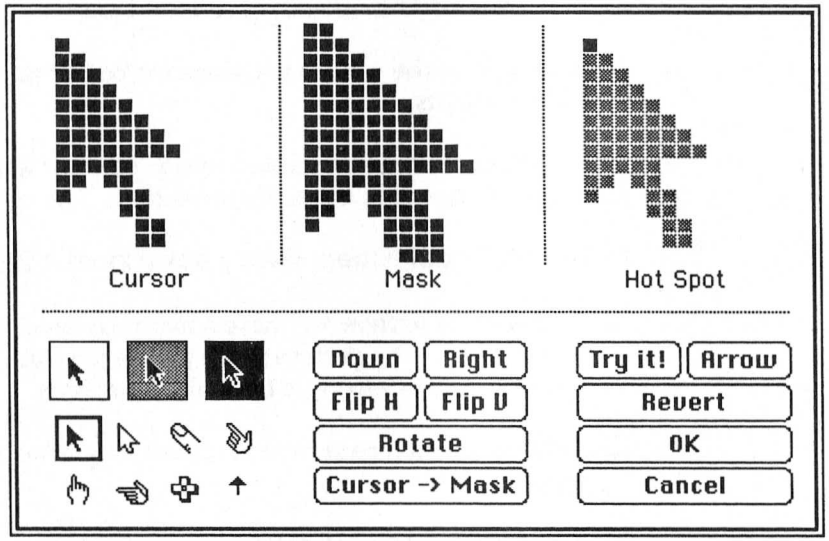
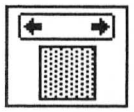


Figure 4-26 The Edit/Select Cursor dialog box



❖ **Pattern Selector** lets you cycle through 38 different patterns by clicking the arrows at the top of the selector box. When you save a calculator work file, all 38 patterns are saved in the work file.

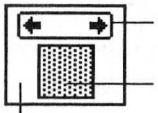


Figure 4-27 The Pattern Selector in detail

Click on the pattern to change the fill patterns of any selected graphic parts. If no graphic part is selected, the Calcshell will be painted with the selected pattern.

Option-click on the pattern to change the border patterns of any selected parts.

Click outside the selected pattern to edit patterns in the *Edit/Select Pattern* dialog box (Figure 4-28).

In the Edit/Select Pattern dialog box (Figure 4-28):

❑ The ***selected pattern*** will have a marquee dancing around it. Click a pattern in the palette to select it, or type the `tab` key to cycle through all patterns.

❑ Edit the selected pattern in the ***fatbits-style box*** in the upper left corner.

❑ Use the ***Clear*** through ***OK*** buttons to further manipulate the pattern.

☒ The ***Horz***, and ***Vert Mirrors*** checkboxes govern the symmetry of a random pattern, created using the ***Random*** button.

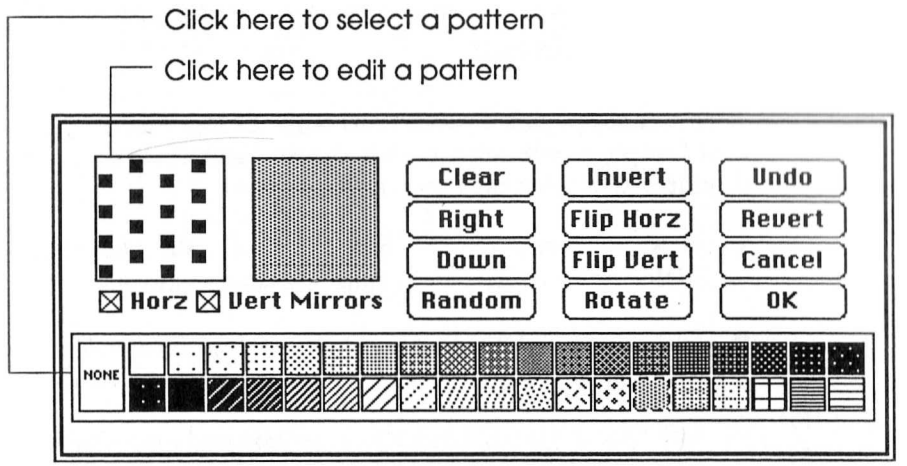


Figure 4-28 The Pattern Edit/Selector dialog box

❖ **Trash Can** is where you drag parts you wish to remove from your calculator entirely. If you prefer using the keyboard, you may press the ***backspace*** key to delete any selected parts instead of dragging them to the trash.

The CCS Menus

There are seven pull-down menus in the CCS menubar. The next few pages will cover each in detail. Here is a brief description of what each menu contains:

- **Apple Menu** Use this menu to display the *About CCS* dialog and to open any installed desk accessories.
- **File Menu** Use this menu to open, close and save files; load and save patterns; manage your windows and quit CCS.
- **Edit Menu** Use this menu to paste text and graphics onto your calculator and to display several "editing" dialogs.
- **Calculator Menu** Use this menu to control Calculator settings and manage your calculator parts.
- **Graphics Menu** Use this menu to manage your graphic parts and selector/graphic settings.
- **Font Menu** Use this menu to select the fonts you'll use for text typing and for the Tall LED.
- **Size/Style Menu** Use this menu to select the size and style you'll use for text typing and for the Tall LED.

Note Any CCS2 menu item that ends with elipsis (...) means that a dialog box will appear after selecting the menu item.

Any CCS2 menu item with a command-letter combination (like ⌘-L) at the right of the menu means that you may type the ⌘-key combination instead of pulling down the menu. This is called the menu item's keyboard equivalent.

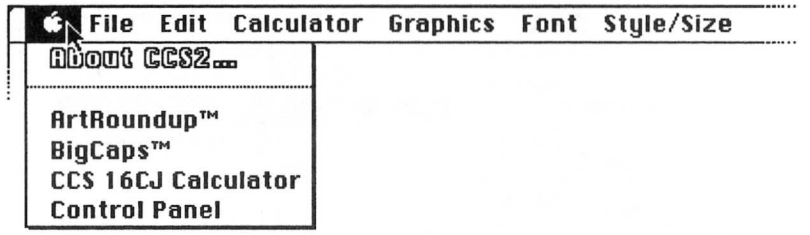


Figure 4-29 The Apple Menu

- **About CCS2...** displays the *About Calculator Construction Set* dialog box (Figure 4-30). Close this dialog box by clicking on it, or typing the `return` or `enter` keys on your keyboard. This dialog box contains the version number of the CCS you are using.
- **All other items** are the names of desk accessories currently installed. Select a desk accessory's name to open the D.A., or if it is already opened, to make it the front window.

Figure 4-30 The About CCS2 dialog box

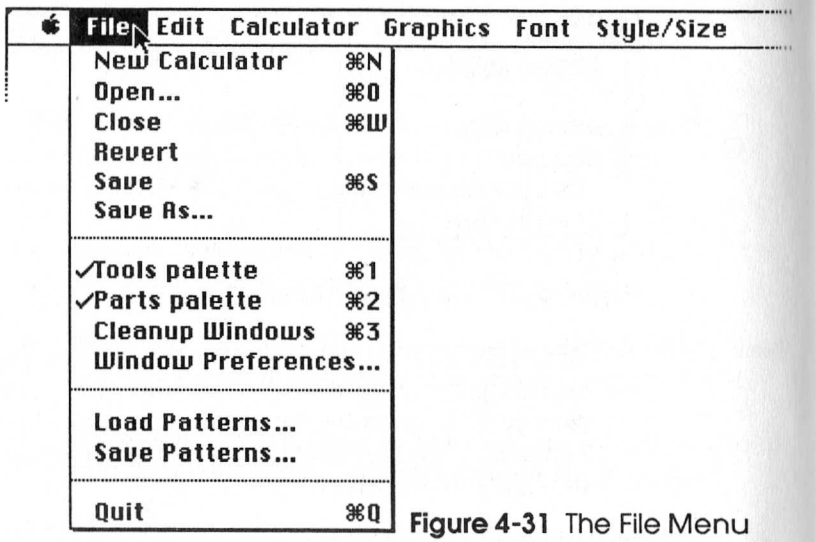


Figure 4-31 The File Menu

- **New Calculator (⌘N)** will display a new, untitled Work window with only a single logo key on it. When the Work window is open, this item will be grayed out.

- **Open... (⌘O)** displays a *GetFile* dialog box (Figure 4-32). Instructions for using this dialog box are on the next page. This dialog box will only display the names of folders and CCS2 work files. After you select a work file and click the *Open* button, the Work window will appear, displaying the calculator contained in the work file. When the Work window is open, this item will be grayed out.

- **Close (⌘W)** closes the Work window. If any changes were made to the calculator since you opened it, an alert box will appear and ask if you wish to save the changes. This item is grayed out when the Work window is closed.

- **Revert** will revert the calculator displayed in the Work window back to the last version saved to your disk. If you are working on an untitled calculator, or if the Work window is closed, this item will be grayed out.

- **Save (⌘S)** will save the calculator displayed in the Work window to the existing CCS2 work file on your disk. If you are working on an untitled calculator, or if the Work window is closed, this item will be grayed out.

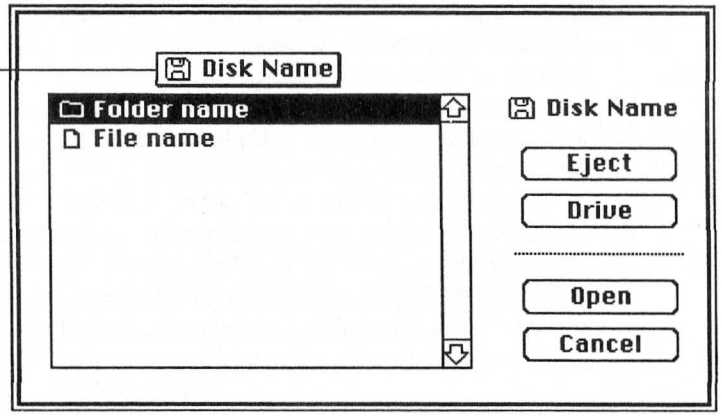


Figure 4-32 Standard "Get File" dialog box

CCS2 displays this dialog box when you pull down *Open...* or *Load Patterns...* from the **File** menu. The document names displayed here will either be CCS2 work file names or *MacPaint* file names. Here's how the dialog works:

If the ***Disk Name*** above the ***Eject*** button is not the name of the disk containing the file you wish to read, click the ***Drive*** button, or insert your disk into the Macintosh's disk drive.

If you are using 800K disks, you may have to open a folder or two to locate the ***File name*** you wish to open. To open a folder, select the ***Folder name***, then click the ***Open*** button.

Figure 4-33 Pop-up Disk name/Folder name menu

To return to a previously opened folder, click on the pop-up disk name/folder name menu which is located above the large box containing the ***Folder*** and ***File names*** (see detail in Figure 4-33). Just select a ***Folder name*** to view its contents.

To open a document, click on a ***File name*** and then click the ***Open*** button (or simply double-click a ***File name***).

❖ **Save As...** will save the calculator in the Work window to a file you create using the ***SaveFile*** dialog box (Figure 4-34).

☐ The ***pop-up menu*** above the list of folder names, the ***Eject*** and ***Drive*** buttons work just like those in the *GetFile* dialog box (Figure 4-33).

☒ The *SaveFile* dialog box allows you to save calculators in three different kinds of files (described in detail on the first page of this chapter). Click on the file icons or checkboxes at the bottom of the dialog to select which kind of files you wish to save.

If a file exists by the same name, you will be asked if you wish to replace it.

Work files are saved with the name you typed into the *Save File* dialog box. ***Desk Accessory*** files will be saved with •CALC added to the end of their names. Calculator ***Application*** files will be saved with •SELF added to the end of their names. We add suffixes to finished work file names so as not to replace your work files.

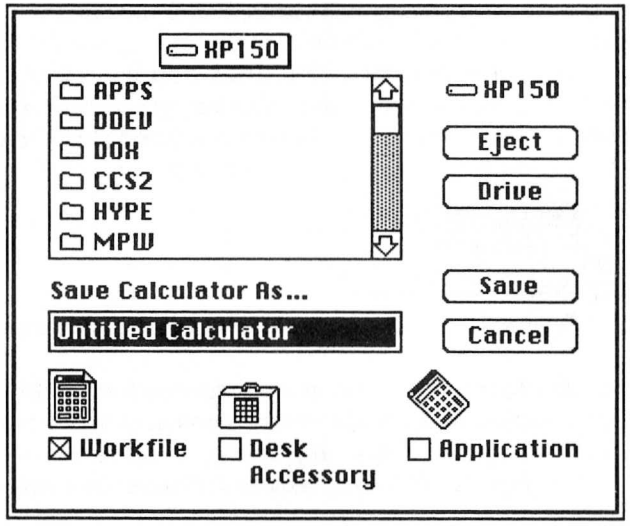


Figure 4-34 The CCS2 "Save File" dialog box.

❖ **Tools palette (⌘1)** will hide the Tools palette if it is visible, or it will show the Tools palette if it is invisible. Clicking the closebox in the Tools palette will also hide it.

❖ **Parts palette (⌘2)** will hide the Parts palette if it is visible, or it will show the Parts palette if it is invisible. Clicking the closebox in the Parts palette will also hide it.

❖ **Cleanup Windows (⌘3)** will rearrange the three CCS windows to position them as in Figure 4-1.

❖ **Window Preferences...** will display the dialog box shown in Figure 4-35.

☒ Checking the three ***Remember...*** boxes will tell CCS to save the positions of the CCS windows when you next quit CCS.

☒ If the ***Make changes permanent*** box is checked, CCS will remember to save the window positions every time you quit.

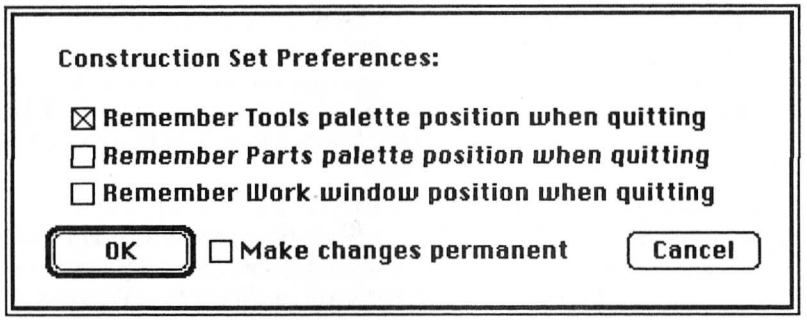


Figure 4-35 The Window Preferences dialog box

❖ **Load Patterns...** will display a *GetFile* dialog box (Figure 4-32) with a list of *MacPaint* file names from which to read the patterns from. The newly loaded patterns will replace all the patterns displayed in the ***Pattern Selector*** (Figure 4-28).

❖ **Save Patterns...** will display a *GetFile* dialog box (Figure 4-32) with a list of existing *MacPaint* file names to which you may save all the patterns displayed in the ***Pattern Selector***. Saving patterns to a *MacPaint* file will not disturb the pictures contained in the file.

❖ **Quit (⌘Q)** will quit *Calculator Construction Set*.

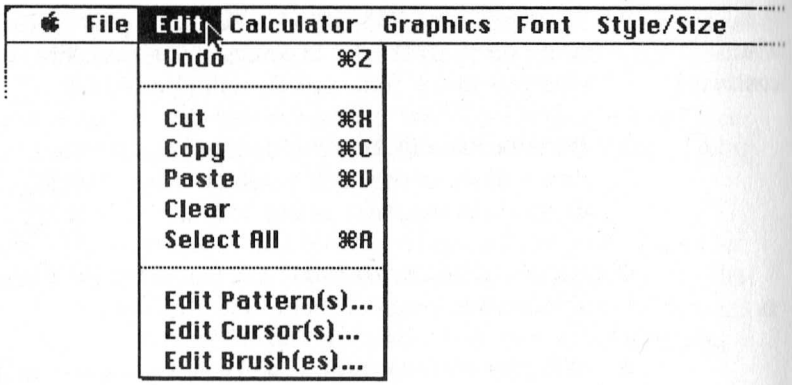


Figure 4-36 The Edit Menu

- **Undo, Cut, Copy, Paste, Clear** will always work in *Test mode* (see *System* functions in Chapter 6) or when a desk accessory is the frontmost window. Otherwise, *Paste* will allow you to add a new graphic part to your calculator (if not in Test mode). If a key is selected when you pull-down *Paste*, the graphic on the clipboard will be pasted onto the key as decoration.

- **Select All (⌘A)** will select all the parts in the Work window. Double-clicking the Magic Hand tool in the Tools palette will also *Select All* parts.

- **Edit Pattern(s)...** will display the *Edit/Select Pattern* dialog box (Figure 4-28). Clicking the selected pattern in the Tools palette will also display this dialog (see Figure 4-27).

- **Edit Cursor(s)...** will display the *Edit/Select Cursor* dialog box (Figure 4-26). Clicking the selected cursor in the Tools palette will also display this dialog.

- **Edit Brush(es)...** will display the *Edit/Select Brush* dialog box (Figure 4-22). Double-clicking the Brush tool in the Tools palette will also display this dialog.

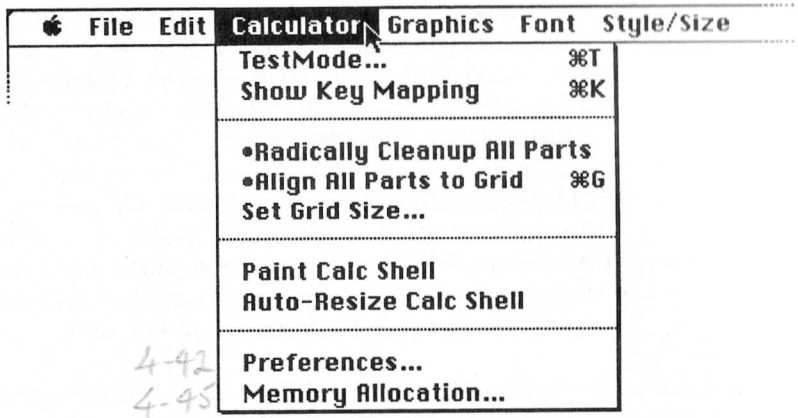


Figure 4-37 The Calculator Menu

- **Test Mode... (⌘T)** will start or exit Test mode. Test mode is a way to test your calculator as though it were a finished calculator— without leaving CCS to do so.

  When you start Test mode, the Tool palette, Parts palette, and Work window are hidden, despite the settings on the **File** menu. When you exit Test mode, the windows are shown again, unless they were hidden before starting Test mode.

  The Calcshell will become a real window. In Test mode, your calculator will function as though it were finished. All calculator functions work in test mode except the *Save Environment* preference (see Figure 4-39).

  An alternate method to start and exit Test mode is to click the close box on the Calcshell (see Figure 4-3).

- **Show/Hide Key Mapping (⌘K).** *Show Keymapping* will redraw all the parts in the Work window using mapped keystrokes instead of the usual keycap. Unmapped parts will be drawn using the unmapped icon (Figure 4-12). *Hide Key Mapping* will redraw all the parts in their normal state.

  Double-clicking the Keyboard tool in the Tools palette will also *Show/Hide Key Mapping*. Wire a `KEYMAP` pushbutton to *Show/Hide Key Mapping* in finished calculators.

❖ **Radically Clean Up Parts.** If no parts are selected, this menu item will read •*Radically Cleanup All Parts*, and will affect every part in the Work window. Otherwise, this menu item will read ◆*Radically Cleanup Selected* and will only affect the selected parts.

Clean up aligns parts to an invisible grid, moving parts as close to each other as possible. If cleaning up all the parts, the Calcshell will be shrunk to the smallest size that will contain all parts in your calculator, and positioned in the upper left corner of the Work window.

You may set the size of the invisible grid using the *Set Grid Size* dialog box (Figure 4-38).

❖ **Align Parts to Grid (⌘G).** If no parts are selected, this menu item will read •*Align All Parts to Grid*, and will affect every part in the Work window. Otherwise, this menu item will read ◆*Align Selected Parts to Grid* and will only affect the selected parts.

This function differs from the *Radical Cleanup* in that parts will not be moved as close to each other as possible. Instead they will be moved to the closest grid coordinate.

You may set the size of the invisible grid using the *Grid Settings* dialog box (Figure 4-38).

❖ **Set Grid Size...** displays a dialog box (Figure 4-38) in which you may choose settings that affect the invisible grid used by *Radical Cleanup* and *Align Parts* functions (see above).

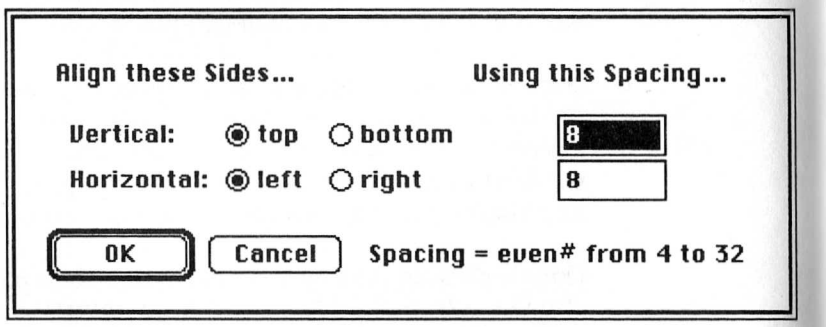


Figure 4-38 The Grid Settings dialog box

❖ **Paint Calcshell** paints the Calcshell with the selected pattern which is displayed in the Tool palette's Pattern Selector (Figure 4-27).

❖ **Auto Resize Calcshell** shrinks the Calcshell to the smallest size that will contain all parts in your calculator. Your calculator's parts will not be moved by this operation.

❖ **Preferences...** displays the *Calculator Preferences* dialog box (Figure 4-39). Use this dialog to set default settings for your finished calculator:

◉ *Save Environment?* If set to *Yes*, each time you close your finished calculator, the data in the Stack, Alpha and Memory Registers will be remembered (the first closing will take longer than subsequent closings). Note that finished calculators always remember their windows' screen position regardless of the *Save Environment* setting.

◉ *Case Sensitive Keys?* If set to *No*, finished calculators will accept either upper case or lower case keystrokes as the keystrokes mapped to a key. If you mapped a lower case x to a key, then typed the uppercase X, the calculator would execute the function mapped to the lower case x. If you actually mapped a function to the uppercase X, that function would be executed instead.

◉ *Paper Tape line #s?* If set to *Yes*, the Paper tape will display line numbers next to each line of text in the tape. Note that in *program record* mode (PRGM), line numbers are always displayed in the Paper tape regardless of this setting. If you wire and press a LINE#S pushbutton, the pushbutton will override this setting. However this setting will always determine the initial position of the pushbutton.

◉ *Paper Tape marginals?* If set to *Yes*, each time you invoke a function which affects the X-Register, the Paper tape will display a *marginal* next to the input or result. Normally, the marginal will be the *function name* (see Appendix 1). However, if the Paper tape is not wide enough to display the entire marginal, the letter F will be used as the marginal. If you wire and click a MARGNL pushbutton, the pushbutton will override this setting. However this setting will always determine the initial position of the pushbutton.

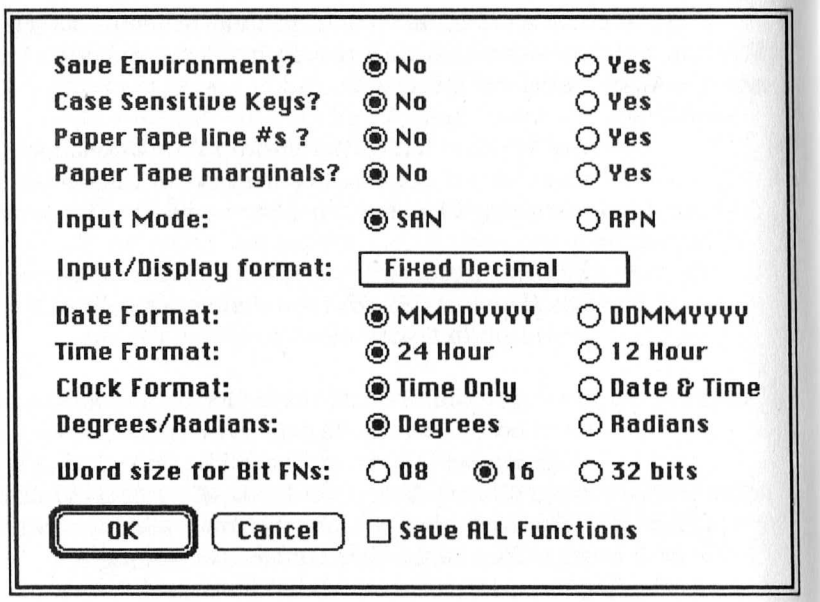


Figure 4-39 The Calculator Preferences dialog box

- *Input Mode* There are two different orders in which numbers and functions may be entered into a CCS calculator.

  Standard Algebraic Notation (SAN) works like the *Calculator* desk accessory that came with your Macintosh. In SAN, you enter equations the way they appear on paper. For instance: 2+3=5. Note that in SAN, you must always key-in the equals sign at the end of the equation before the result is displayed in the MainLED.

  Stack-based Reverse Polish Notation (RPN) works much like the Hewlett-Packard line of calculators. In RPN, you enter data into Stack Registers first, then invoke the operand (function). For instance: 2 ENTER 3 +. Note that in RPN, the equation is evaluated as soon as you invoke a function (in this case, plus). The answer 5 would automatically be displayed in the X-Register without the need to end the equation with an equals sign. See Chapters 3 or 5 for more on the Stack Registers.

  Unlike the MainLED, the Paper tape will not display answers to arithmentic operators (+, −, *, +) without pressing the equals, subtotal, or total keys.

- The *Input Mode* radio buttons determine whether you will enter equations in Standard or Reverse Polish Notation. If you wire and press an RPN/SAN clickstop switch, the switch will override this setting. However this setting will always determine the initial position of the switch. ***Note that each time you change between RPN and SAN using a clickstop switch, the Stack Registers will be cleared.***

- *Input/Display format* Use this pop-up menu to select the display/input format for your finished calculator. (See Chapter 5 for more on input/display formats). If you are using an older version of the System file, you will not be able to select the default input/display format— the default will be FIXED (with 4 decimal places).

  If you wire and press an input/display format clickstop switch, the switch will override this setting. However, this setting will always determine the initial position of the clickstop switch.

- *Date Format* There are two different formats for entering decimal dates. Note that leading and trailing zeroes may be omitted if you desire.

  - MMDDYYYY where January 24, 1984 is entered as `1.241984`
  - DDMMYYYY where January 24, 1984 is entered as `24.011984`

  If you wire and press either the MDY or DMY keys, the keys will override this setting. However this setting will always be used until you override with the keys.

- *Time Format* You can choose the format used by time displays (with the exception of the Continuous Time SLEDs).

  - 12 hour, where 2:30 in the afternoon would be: `02:30:00 PM`
  - 24 hour, where 2:30 in the afternoon would be: `14:30:00`

  If you wire and press either the CLOCK12 or CLOCK24 keys, the keys will override this setting. However this setting will always be used until you override with the keys.

◉ ***Clock Format*** You can choose the format used by time/date displays (with the exception of the Continuous Time SLEDs).

- Time Only (where only the time is displayed)
- Date & Time (where date is appended to the time display)

If you wire and press either the `DATE` or `TIME` keys, the keys will override this setting. However this setting will always be used until you override with the keys.

◉ ***Degrees/Radians*** This angular setting determines whether trigonometric functions expect input and display results as either degrees or radians. See the ***Math Functions*** section in Chapter 6 for more.

If you wire and press a `DEGREES/RADIANS` clickstop switch, the switch will override this setting. However this setting will always determine the initial position of the switch.

◉ ***Word Size*** These settings determine the length of a bit operation (such as `AND`, `OR`, `XOR`). See the ***Bit Operations*** section of Chapter 6 for more.

If you wire and press an `SWS` (set word size) key, the key will allow you to override this setting. However this setting will remain in effect until you override it. Wire and press a `WS?` (word size status) key to view the current setting.

☒ ***Save All Functions*** When saving the calculator as a desk accessory file, CCS only saves those functions which you have specifically wired to parts— to keep the size of the desk accessory as small as possible. Wiring the `XEQ` or `PRGM` functions will cause all functions to be saved— regardless of this checkbox's setting. When saving the calculator as an application file, all functions are saved— regardless of this checkbox's setting.

Checking this box will force CCS to save all functions. If you wired a script to a key, the script may not run properly unless all functions are saved in the calculator. If, while running your script from your finished calculator D.A., you get a `Script Error` message, try checking the *Save All Functions* box and saving your calculator again.

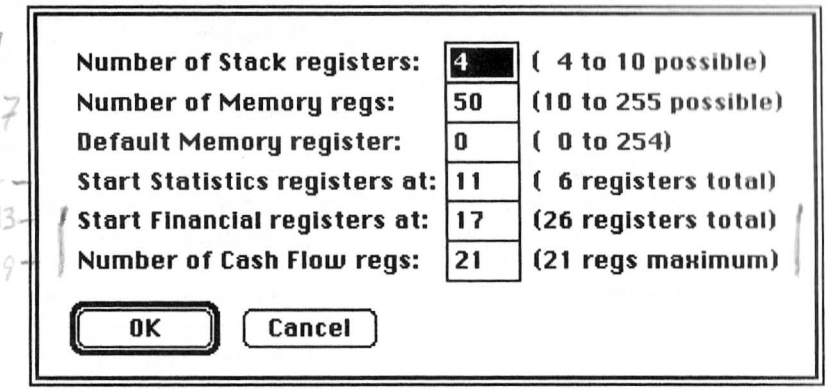


Figure 4-40 The Calculator Memory Allocation dialog box

❖ **Memory Allocation...** displays the ***Calculator Memory Allocation*** dialog box (Figure 4-40). Use this dialog to set the number of Stack and Memory Registers to save with your finished calculator. See Chapter 5 for more specific information on registers.

❑ ***Number of Stack Registers:*** you may wire from 4 to 10 Stack Registers, named: `X`, `Y`, `Z`, `T`, `S5`, `S6`, `S7`, `S8`, `S9`, `S10`.

❑ ***Number of Memory Registers:*** you may wire from 10 to 255 Memory Registers, named `0` through `254`.

❑ ***Default Memory Register:*** this is the Memory Register which program scripts will store and save to if you do not specify one in your script. See Chapter 5 for more.

❑ ***Start Statistics Registers at:*** Enter the number of the first Statistics Register. If you are wiring statistics functions, you will need to allocate a block of 6 Memory Registers for the exclusive use of the statistics functions. Otherwise, this setting is ignored (see Chapter 5).

❑ ***Start Financial Registers at:*** Enter the number of the first Financial Register. If wiring financial functions, allocate a block of 26 Memory Registers. If not using financial functions, this setting has no meaning (see Chapter 5).

❑ ***Number of Cash Flows regs:*** Enter the number of cash flow registers to be used in financial calculations (see Chapter 8).

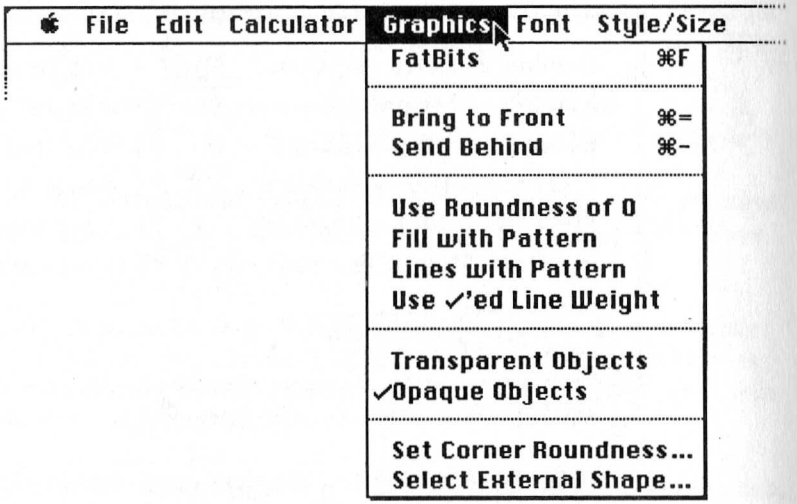


Figure 4-41 The Graphics Menu

- **Fatbits (⌘F)** If this item is checked, clicking a part with the Pencil tool displays the ***Fatbits*** editing dialog (Figure 4-24).

- **Bring to Front (⌘=)** will bring the selected graphic part(s) to the front—they will draw in front of other graphic parts. Graphic parts always draw behind keys, switches and displays. This item is grayed out when there are no graphic parts selected.

- **Send Behind (⌘-)** will send the selected graphic part(s) to the back—they will draw behind other graphic parts. Graphic parts always draw behind keys, switches, and displays. This item is grayed out when there are no graphic parts selected.

- **Use Roundess of x** will set the roundness factor of any selected box parts, where x = the roundess factor. Set the roundness factor with the ***Set Corner Roundness...*** dialog. This item is grayed out when there are no graphic parts selected.

- **Fill with Pattern** will fill any selected box parts with the selected pattern (as displayed in the Tools palette). This item is grayed out when there are no graphic parts selected.

- **Lines with Pattern** will paint the border of any selected box or line parts with the selected pattern (as displayed in the Tools palette). This item is grayed out when there are no graphic parts selected.

- **Use Checked Line Weight** will set the thickness of the border of any selected box or line parts with the selected line weight (as displayed in the Tools palette). Use the ***Line Weight Selector*** in the Tools palette to set the line weight (see Figure 4-28). This item is grayed out when there are no graphic parts selected.

- **Transparent/Opaque Objects** will change the opaqueness of an selected graphic part. If a part is opaque, any parts behind it will be hidden by the opaque part. If a graphic part is transparent, any parts behind it can be seen through the transparent part.

- **Set Corner Roundess...** displays the dialog shown in Figure 4-25, which allows you to draw box parts with rounded corners.

- **Select External Shape...** displays the dialog shown in Figure 4-43, which allows you choose an external shape definition for the selected part. Valid shape numbers are from `0` to `31`. If your copy of CCS does not contain a `'XSHP'` resource of the ID number you choose, the shape will be drawn like this:

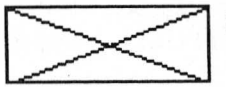


Figure 4-42 A part set to draw with an `'XSHP'` resource which is not present.

Experienced programmers may write their own `'XSHP'` segments in C, Pascal, or Assembly languages. Your `XSHP` code will then be called by the calculator and CCS to handle the drawing of the part. For more information on writing external shapes and external functions, see Appendix 7.

Release Notes:
External Shape files
Not supplied
floppy disks No space
Send for $5

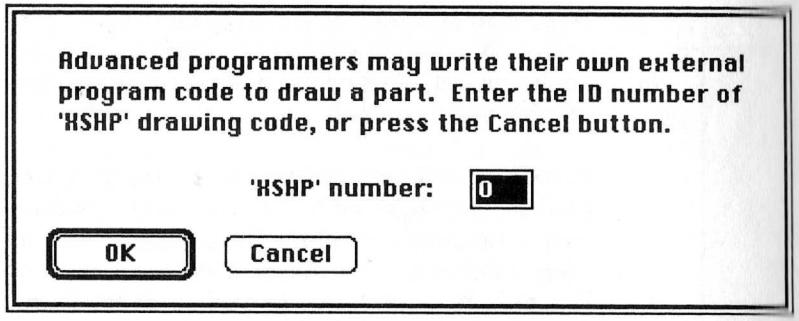


Figure 4-43 External Shape Selection dialog box

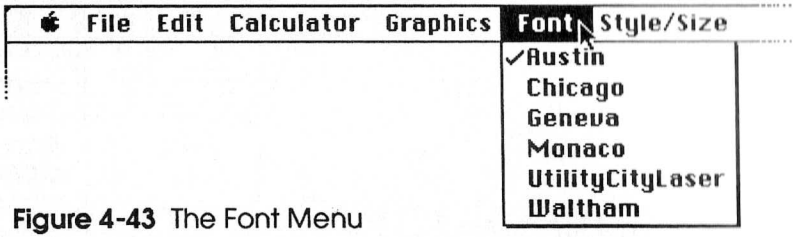


Figure 4-43 The Font Menu

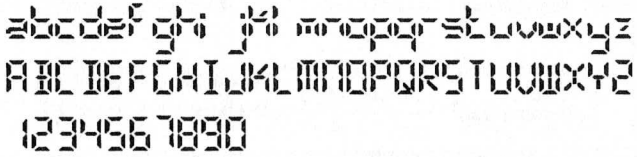
This menu will contain the names of all installed fonts. When CCS is first opened, it will select the ***Austin*** font— if you have installed it. Otherwise, the default application font (usually *Geneva*) will be selected.

The *Austin* and *Waltham* fonts are in a *Font/DA Mover* file on the CCS2 disk. Font selection affects Text parts and the Tall LED. Select these parts with the Magic Hand, then choose a font from the ***Font*** menu. The parts will draw using the font you choose.

The Austin 9 point Font
drawn by Cliff Joyce, and useful for text parts

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890

The Waltham Font
drawn by J. Douglas Robertson and Mary Martinez, useful for Tall LED displays



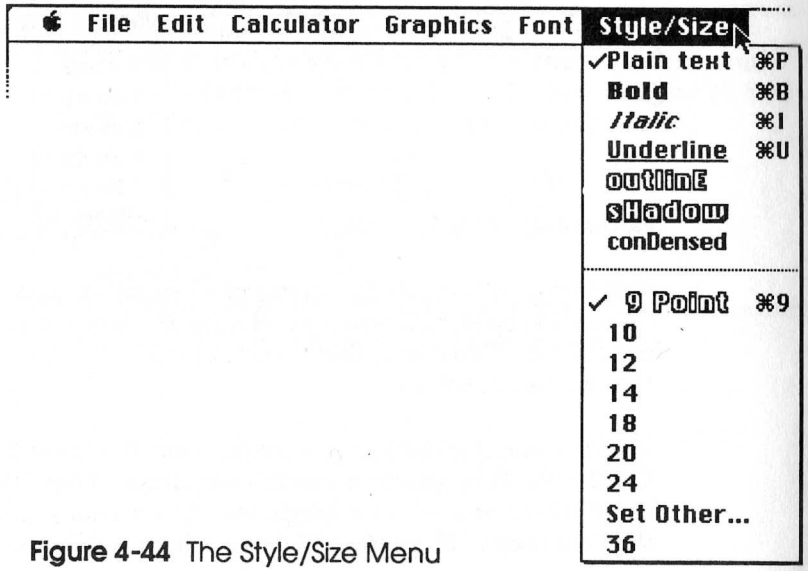


Figure 4-44 The Style/Size Menu

As with font selection— size, and style selection only affects Text parts and the Tall LED. Select these parts with the Magic Hand and choose a style or size from this menu.

Set Other... displays a dialog which lets you choose a size which might not be listed on the menu (Figure 4-45). The "other" size appears below the *Set Other...* menu item.

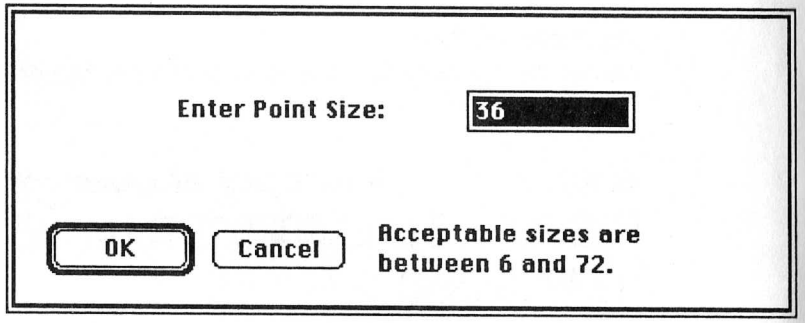


Figure 4-45 The "Set Other..." Point Size dialog box

Chapter 5

Finished Calculators: Modes and Registers

This chapter describes built-in components of a finished calculator (we assume you have wired every function available). Unlike the parts that control and display them, *Modes* and *Registers* are built into every finished calculator— without having to specifically drag them onto a Calcshell.

An analogy would be the way the Macintosh ***clipboard*** works. When you pull down *Copy* from the *Edit* menu, you take it on faith that what you are copying ends up on the clipboard. You usually never ***see*** the clipboard— until you pull down *Paste* from the *Edit* menu. Some applications (like the *Finder*) let you see the clipboard's contents via a window—but you don't ***have to see*** the clipboard to be able to use it.

The same idea aplies to modes and registers. Often, you can wire a part which will display a mode or register— but this is not a requirement. The next few pages will provide specifics on:

- **Display Formats** control how LEDs display their contents.
- **Modes** affect the way the calculator will operate.
- **Stack Registers** a block of memory which calculator functions use to hold and manipulate data.
- **Memory Registers** a block of memory into which you may store and retrieve data.
- **Flags** a block of memory into which you may store `TRUE` or `FALSE` settings; each setting is referred to as a *flag*.

Display Formats

Display Formats govern how displays will draw their contents. You may wire a SLED to display the X-Register in any display format. The possible display formats are:

BINARY	Base 2 arithmetic
DEGREES	Shown as Degrees:Minutes:Seconds
DOLLARS	2 decimal places, preceeded by $
DOZENAL	Base 12 arithmetic
ENG *(Engineering)*	Like SCI (see below), except the exponent is always a multiple of 3
FIXED	0 to 9 decimal places (see FIX function)
FLOAT	Floating Decimal
FTINCH	Shown as Feet' Inches"
HEX	Base 16 (hexadecimal) arithmetic
HRSMIN	Shown as Hours:Minutes:Seconds
INTEGER	Whole numbers only
OCTAL	Base 8 arithmetic
POUNDS	2 decimal places, preceeded by £
SCI *(Scientific Notation)*	Where 1.23 E4 represents 12,300. Left of the E is the ***mantissa***. Right of the E is the ***exponent***. Exponents determine how many places to shift the mantissa's decimal point left (if exponent is negative) or right (if exponent is positive). When numbers contain too many significant digits to fit in a display using the appropriate display format, the calculator may automatically display the number in Scientific Notation.
TIMEDSP	Shown as Days:Hours:Minutes:Seconds
YDFEET	Shown as Yards^Feet'Inches"
YEN	2 decimal places, preceeded by ¥

The Input/Display Format is the display format in which the calculator performs its operations. There is only one input/display format active at any time. The MainLED, the Paper tape, and various SLEDs all display their contents using the input/display format. The input/display format also controls what kind of data you can key-in, and how it will be interpreted.

The default input/display format may be set during construction via the ***Calculator Preferences*** dialog box (Figure 4-39). The input/display format may be changed in finished calculators by pressing an ***Input/Display*** format clickstop switch (see Chapter 6), or by pressing the XEQ key, then responding to the prompt dialog with the format's name.

Here's how the input/display formats restrict and interpret the data you key-in:

- **BINARY** (Base 2 format). You may only enter the digits 0 or 1, which are interpreted as single bits. Doing any calculations in this format will always truncate the answer to a whole number.

- **DEGREES** (Degrees:Hours:Seconds format). Digits 0 through 9 are valid in this format. Separate degrees, hours and minutes by entering a colon (:) between them. If you only enter one colon, your input is interpreted as Degrees:Hours. If you do not enter any colon, your input is interpreted as Degrees.

- **DOLLARS** (A fixed decimal format). Digits 0 through 9 are valid in this format. Use the decimal point (.) to separate whole numbers from fractional numbers. You may enter more than 2 digits to the right of the decimal point, even though the underlying number will be rounded in the display to 2 decimal places. You do not enter the leading $ yourself; it will be displayed automatically. If you do not enter a decimal point, your input is interpreted as an integer.

- **DOZENAL** (Base 12 format). Digits 0 through 9, A and B are valid in this format. Use the decimal point (.) to separate whole numbers from fractional numbers. If you do not enter a decimal point, your input is interpreted as an integer.

- **ENG** (Engineering). Same as SCI (below) except the exponent will always be displayed as a multiple of 3— even if you did not enter it that way.

- **FIXED** (Fixed decimal format). Digits 0 through 9 are valid in this format. Use the decimal point (.) to separate whole numbers from fractional numbers. Execute the FIX function to set the number of digits to the right of the decimal point (from 0 to 9); FIXED format always rounds the display to this number of digits, but does not round the underlying number. You may enter more digits to the right of the decimal point than FIXED format will display. If you do not enter a decimal point, your input is interpreted as an integer.

- **FLOAT** (Floating decimal format). Digits 0 through 9 are valid in this format. Use the decimal point (.) to separate whole numbers from fractional numbers. If the number being displayed is an integer, FLOAT will not display a decimal point. If you do not enter a decimal point, your input is interpreted as an integer.

- **FTINCH** (Feet and Inches format). Digits 0 through 9 are valid in this format. Use the single quote (') to separate feet from inches. If you do not enter a quote mark, your input is interpreted as Feet.

- **HEX** (Hexadecimal or Base 16 format). Digits 0 through 9, and A through F are valid in this format. Doing any calculations in this format will always truncate the answer to a whole number.

- **HRSMIN** (Hours:Minutes:Seconds format). Digits 0 through 9 are valid in this format. Separate hours, minutes and seconds by entering a colon (:) between them. If you only enter one colon, your input is interpreted as Hours:Minutes. If you do not enter any colon, your input is interpreted as Hours.

- **INTEGER** Digits 0 through 9 are valid in this format. Doing any calculations in this format will always truncate the answer to a whole number.

- **OCTAL** (Base 8 format). Digits 0 through 8 are valid in this format. Doing any calculations in this format will always truncate the answer to a whole number.

- **POUNDS** (A fixed decimal format). Same as DOLLARS format, except using a £ symbol instead of a $ symbol.

- **SCI** (Scientic Notation; a fixed decimal format). Digits 0 through 9 are valid in this format. Use the decimal point (.) to separate whole numbers from fractional numbers. Use the EEX (enter exponent) function to separate the mantissa from the exponent. If you press the CHS (change sign) function before the exponent has been entered, the entire number will be negative. Pressing CHS after the exponent has been entered will treat the exponent as negative.

- **TIMEDSP** (Days, Hours, Minutes, Seconds format). Digits 0 through 9 are valid in this format. Separate days, hours, minutes, and seconds by entering a colon (:) between them. If you don't enter colons, your input is interpreted as Days.
  One colon = Days:Hours.
  Two colons = Days:Hours:Minutes.
  Three colons = Days:Hours:Minutes:Seconds.

- **YDFEET** (Yard, Feet and Inches format). Digits 0 through 9 are valid in this format. Use the caret (^) to separate yards from feet. Use the single quote (') to separate feet from inches. If you do not enter either separator, your input is interpreted as Feet. If you enter the single quote, but not the caret, your input is interpreted as feet and inches— but will be redisplayed as yards, feet and inches. For instance, if you enter: 20'6, it will be displayed as: 6^2'6". If you only want to display feet and inches, use the FTINCH format.

- **YEN** (A fixed decimal format). Same as DOLLARS format, except using a ¥ symbol instead of a $ symbol.

Calculator Modes

A ***mode*** is an aspect of a finished calculator that affects the way the calculator and it's functions will operate.

❖ **Input Modes** control what kind of task you want to use the calculator for. The input modes are:

➤ ***Normal Calculation mode*** used to do calculations. The MainLED's input/display format determines which characters are valid input when in this mode. Display formats may be viewed/controlled by wiring an ***Input/Display*** format clickstop switch.

➤ ***RPN/SAN Mode*** There are two different orders in which you may enter numbers and execute functions into a CCS calculator while in normal calculation mode:

- ***Standard Algebraic Notation*** (SAN) works like the *Calculator* desk accessory that came with your Macintosh. In SAN, you enter equations the way they appear on paper. For instance: 2+3=5. Note that in SAN, you must always key in the equals sign at the end of the equation before the result is displayed in the MainLED.

- ***Stack-based Reverse Polish Notation*** (RPN) works much like the Hewlett-Packard line of calculators do. In RPN, you enter data into stack registers first, then invoke the operand (function). for instance: 2 ENTER 3+. Note that in RPN, the equation is evaluated as soon as you invoke a function (in this case, plus). The answer 5 would automatically be displayed in the X-Register without the need to end the equation with an equals sign. Stack Registers are explained later in this chapter.

  Unlike the MainLED, the Paper tape will not display answers to arithmetic operators (+, −, *, +) without pressing the equals, subtotal or total keys.

➤ ***ALPHA Mode*** is used to enter text into the 24 character Alpha Register. Press the ALPHA pushbutton switch to enter/exit ALPHA mode. When in ALPHA mode, the MainLED will display the contents of the Alpha Register, and the Annunciator display will say ALPHA. In general, calculations cannot be performed while in ALPHA mode. However, the following functions do work in ALPHA mode:

ADATE	ASTO	CLC	XASA
APPEND	ATIME	COPY	XTOA
ARCL	ATIME24	CUT	
AROT	CL	PASTE	
ASHF	CLA	SST	

➤ ***TEXT Mode*** is used to edit the text contained on a Paper tape. Enter TEXT mode by clicking on the Paper tape or via the TEXTON function. A blinking vertical bar will mark the insertion point. Exit TEXT mode by clicking anywhere in the calculator except in the Paper tape, or via the TEXTOFF function. When in TEXT mode, the Annunciator display will say TEXT. In general, calculations cannot be performed while in TEXT mode. However, the following functions do work in TEXT mode:

CENTER	CUT	LINEOFF	SELALL
CL	GETAS	PASTE	SST
CLC	LEFT	RIGHT	TEXTOFF
COPY	LINEON	SAVEAS	TPRINT

➤ ***PRGM (Program Script Recording) Mode*** is used to create, edit and display program scripts using the Paper tape. Enter and exit PRGM mode by pressing the PRGM pushbutton (complete details are in Chapter 8). When in PRGM mode, the Annunciator display will say PRGM.

❖ **Shift Modes** There are two of these: TOP and BOT shift. They determine which function will be invoked on multi function keys.

A single click on the TOP or BOT pushbutton will temporarily turn the TOP or BOT mode *ON*. As soon as another key is pressed, the shift mode turns itself *OFF*.

⌘-click either the TOP or BOT pushbuttons to "lock" them. They remain locked in the *ON* position until you click the TOP or BOT pushbuttons again.

❖ **Degrees/Radians Angular Mode** determines whether trigonometric functions expect input and display results as either degrees or radians. See the Mathematic Functions in Chapter 6 for more. Press the ***Degrees/Radians*** clickstop switch to change the setting of the Degrees/Radians mode.

Stack Registers

Stack Registers are a block of memory used to contain the data on which CCS functions will generally perform calculations (there are some functions which perform operations directly on data contained in Memory Registers).

CCS2 calculators are, therefore, referred to as ***stack-based***. You may choose between 4 to 10 stack registers during construction via the ***Calculator Preferences*** dialog box (see Figure 4-39). The stack registers (levels) are labeled:

X, Y, Z, T, S5, S6, S7, S8, S9 and S10

The LastX-Register. Every CCS2 calculator also contains a "bonus" Stack Register called the ***LastX-Register***. Many calculator functions copy the number in the X-Register to the LastX-Register before executing. This means that you can use the Last X-Register as a kind of **Undo** command. Press the LASTX key to move the contents of the LastX-Register into the X-Register. See Appendix 1 for a list of functions which change the LastX-Register.

In normal calculation mode, the MainLED always displays the contents of the X-Register. During construction, you may wire SLEDs to display the contents the Stack Registers. However, the Stack Registers are present in your calculator whether or not you wire SLEDs to display their contents.

See Chapter 6 to understand which stack register(s) a function will act upon. Not every function relies solely upon an amount in the X-Register. Many require values in both the X-Register and the Y-Register. Some, like financial calculations, also rely on data contained in specific Memory Registers (see Chapter 7).

The contents of the Stack Registers will be preserved when you close your calculator if you selected the ***Save Environment*** preference during construction (see Figure 4-39).

Important If you are using Standard Algebraic Notation, the workings of the Stack Registers will be transparent to you, and you may skip this section.

3-3
5-6

Stack-Based RPN Calculations

Figure 5-1 (below) shows what happens to each Stack Register during each step of an RPN calculation, using the sample equation: $32 \div (5^2 - 9)$. This example assumes a 4-level stack. For brevity, the T-Register is not shown.

Z-Register	Z 0	**Key in:** 32
Y-Register	Y 0	the stack lifts
Main LED	32	32 goes into X-
LastX Reg	L 0	Register

Z-Register	Z 0	**Key in:** ENTER
Y-Register	Y 32	the stack lifts
X-Register	32	
LastX reg	L 0	

Z-Register	Z 0	**Key in:** 5
Y-Register	Y 32	
X-Register	5	replaces X-Register
LastX reg	L 0	

Z-Register	Z 0	**Key in:** XTO2
Y-Register	Y 32	
X-Register	25	result in X-Register
LastX reg	L 5	original X to LastX

Z-Register	Z 32	**Key in:** 9
Y-Register	Y 25	the stack lifts
X-Register	9	9 goes into X-Register
LastX reg	L 5	

Z-Register	Z 0	**Key in:** -
Y-Register	Y 32	the stack drops
X-Register	16	
LastX reg	L 9	old X to LastX-Register

Z-Register	Z 0	**Key in:** /
Y-Register	Y 0	the stack drops
X-Register	2	
LastX reg	L 16	old X to LastX-Register

Figure 5-1 Stack Lift/Drop Example $(32 \div (5^2 - 9))$

Stack Lift and Stack Drop

These terms describe the automatic movements of stack contents either up (as in stack lift) or down (as in stack drop).

The ***Stack Lifts*** when a new number is keyed into the X-Register. (like 32 and 9 in our example). The number in each register moves ***up*** to the next register (Y-Register moves to Z-Register, and Z-Register moves to T-Register). The number in the highest register (here, the T-Register) is lost.

The ***Stack Drops*** when a function combines the numbers in the X-register and Y-register (like - and / in our example). The numbers in all registers move ***down*** to the next register (T-Register moves to Z-Register , Z-Register moves to Y-Register, X-Register moves to the LastX-Register). The number in the lowest register—which is always the LastX-Register is lost. The result of the function is then moved to the X-Register.

Pressing ENTER separates two sequentially-keyed numbers (like 32 and 5 in our example). When you press ENTER, the stack lifts, copying the number in the X-Register (32 in our example) to the Y-Register. But then, the ENTER function ***disables stack lift*** long enough for you to key a second number into the X-Register (5 in the example).

Memory Registers

Memory Registers are storage areas in which you, or a program script you write, may place important, although temporary data. The data may be numeric, or text (alpha data). CCS calculators do not use Memory Registers to store program scripts. Some Financial and Statistical functions also use Memory Registers as work areas.

Memory Register contents may be displayed in SLEDs, Selector SLEDs, or recalled to the MainLED. Memory Registers are present in your calculator whether or not you wire SLEDs to display their contents.

The contents of the Memory Registers can be manipulated using memory functions (see Chapter 6). Memory Registers, although temporary storage, can be saved to and recalled from permanent disk files. If you choose the ***Save Environment*** preference during construction, all Memory Registers will be remembered each time you close your calculator.

During construction, you may wire a contiguous block of up to 255 Memory Registers, named from `0` through `254`. We sometimes abbreviate Memory Register 24, for example, as `MEM_24`. Each Memory Register you wire increases the amount of memory used by your finished calculator by 12 bytes.

If you will be using Financial or Statistical functions in your calculator, you will need to wire a block of Memory Registers for the exclusive use of those functions:

- **Financial Registers** 26 of the Memory Registers are designated as Financial Registers. The financial functions rely upon these registers being wired, and may not function at all if you have not allocated the Financial Registers. Financial Registers are normally assigned to be Memory Registers 17 through 42, but may be moved to any contiguous block of available Memory Registers via the ***Memory Allocation*** dialog box (figure 4-40), or the `MOVEF` function (described in Chapter 6).

Here are the Financial Registers, which are always assigned in the following order:

Financial Register name:	**Memory Register number:** (unless you move them)
N	17
i	18
PV	19
Pmt	20
FV	21
FØ thru F2Ø	22 thru 42 (Optional)

Figure 5-2 Financial Registers

If your calculator has no financial functions wired, registers 17 through 42 are available for use as normal Memory Registers. When financial functions have been wired, care should be taken when using Memory Registers for general purpose `STO` and `RCL` functions. For example, if you `STO` something to `MEM_17`, you will replace the financial functions' value of `N` that may have previously stored.

If you wire less than 41 Memory Registers at construction time, the full compliment of Financial Registers will not be available unless you assign them to a lower range.

Note that the Cash Flow registers (`F0-F20`) are optional. See the ***Memory Allocation*** dialog box (Figure 4-40) and the Financial Registers section of Chapter 6 for more.

- **Statistics Registers** Six of the Memory Registers are designated as Statistics Registers. The statistics functions rely upon these registers being wired, and may not function at all if you have not allocated the Statistics Registers. Statistics Registers are normally assigned to be Memory Registers 11 through 16, but may be moved to any contiguous block of available Memory Registers via the ***Memory Allocation*** dialog box (figure 4-40) or the `ΣREG` function (Chapter 6).

Here are the Statistics Registers, which are always assigned in the following order:

Statistics Register name:	MemReg #: (if not moved)
Σx summation of x-values	11
Σx^2 summation of squared x-values	12
Σy summation of y-values	13
Σy^2 summation of squared y-values	14
Σxy summation of products of x-values and y-values	15
ΣN # of data pairs accumulated	16

Figure 5-3 Statistics Registers

If your calculator has no statistics functions wired, registers 11 through 16 are available for use as normal Memory Registers. When statistics functions have been wired, care should be taken when using Memory Registers for general purpose STO and RCL functions, so as not to overwrite one of the Statistics Registers by accident.

If you wire less than 16 Memory Registers at construction time, the full compliment of Statistics Registers will not be available unless you assign them to a lower range.

Default Memory Register

This is a term we use to describe the Memory Register that is currently in use. Generally, your CCS calculator always keeps track of the ***Default Memory Register*** for you. The Annunciator Display (if stretched wide enough) will always display MEM_nnn, where nnn is the number of the Default Memory Register. The ***Memory Register*** Selector SLED is used to display/change the Default Memory Register.

Consult Appendix 1 for a list of memory functions which change the Default Memory Register.

The Alpha Register

This is a special register which can contain alphanumeric data. The ***Alpha Register*** can hold up to 24 characters. This is the only register which allows text input, however you may copy the contents of the Alpha Register to any valid Memory Register (see the ***Functions*** Chapter).

Pressing the ALPHA pushbutton will display the Alpha Register in the MainLED, and turn ALPHA mode *ON*. In ALPHA mode, the Macintosh keyboard is ***unmapped***. That is to say, typing a key does not invoke a function normally mapped to the key. Instead, a letter will be placed in the Alpha Register.

If you press a key just after turning ALPHA mode *ON*, that first keystroke will clear the contents of the Alpha Register before adding the new letter. If you do not want to clear the existing contents of the Alpha Register, press the APPEND function before typing the first keystroke, and the new letter will be added to the right side of the Alpha Register.

When the Alpha Register is full, the leftmost character will be shifted out of the register and lost.

You may wire a SLED to constantly display the Alpha Register. However, the Alpha Register is built into your calculator whether or not you wire a SLED to display it in.

The Alpha Register will be preserved when your calculator is closed, by wiring the ***Save Environment*** preference during construction.

Status Flags

A Flag is small area of memory into which a TRUE or FALSE setting is stored. A flag that is *set* is interpreted as TRUE. A flag that is clear is interpreted as FALSE.

CCS has 80 Status Flags. Many have the same meaning as the Status Flags found on the HP-41 series of calculators, and are included for program script compatability. Some HP flags are not supported in CCS. Unsupported "read only" flags will always test as clear. Unsupported "read-write" flags can be used as additional user flags.

All flags will be saved with the environment if your finished calculator was configured to do so.

READ/WRITE FLAGS (0 thru 29)

You may directly set/clear flags 0 thru 29 using the SF (set flag) and CF (clear flag) functions. You may also test them using the functions: FC?, F?C, FS? and F?S (see Chapter 6).

- **User controlled bit flags** (0 thru 10)
  Set/clear/test user flags for your own purposes—usually in calculator scripts.

- **Automatic Startup Script Execution** (11)
  From a finished calculator, set a permanent script as the current script (using the PRGM dialog box), Set Flag 11, and close the calculator. The permanent script will automatically execute the next time the finished calculator is opened. Clear Flag 11 and close the calculator to stop the script from automatically executing. Flag 11 only has meaning in finished calculators configured to save their environments and does not work at all in Test mode.

- **The Carry Flag** (15)
  This flag is used by several of the ***Bit Operations*** functions. See Chapter 7 for more on bit operations which rotate using the *carry flag*.

- **External Device Control** (12-20, except 15)
  These flags are not supported in CCS. They will always test clear unless you set them.

- **Printer Enable** (21)
  This flag is not supported in CCS. It will always test clear unless you set it.

- **Data Input Flags** (22-23)
  These flags are used by scripts which prompt users for input to help if and where the user's response can be found. *Flag 22* is set when numbers are keyed into the X-Register. *Flag 23* is set when characters are keyed into the Alpha Register. Your program script should clear these two flags just before the PROMPT statement, as the calculator never clears them for you.

- **Error Flags** (24-25)
  These flags are not supported in CCS. They will always test clear unless you set them.

- **Audio Enable** (26)
  When this flag is set, any function which would normally produce a beep or note will do so. You can mute these functions by clearing this flag.

- **User Keyboard** (27)
  This flag is not supported in CCS. It will always test clear unless you set it.

- **Punctuation Control**(28-29)
  In decimal input/display formats, the displays are formatted using two punctuation marks: a Radix (decimal point) and Thousands Separator (a comma in the United States). *Flag 28* will swap these two punctuation marks each time you set or clear it.
  Flag 29 will force or inhibit the thousands separator. The commas pushbutton will also toggle flag 29.

READ ONLY FLAGS (30 thru 80)

Some CCS flags are "read only" meaning that you cannot set or clear the flags— only test them using the functions: `FC?` and `FS?`. Generally, these flags are set by the Calculator itself.

- **Catalog Control** (30)
  This flag is not supported in CCS. It will always test clear.

- **Date Format** (31)
  Interrogate this flag to determine the current Date format. Flag 31 tests clear if in `MDY` format, and will be set if in `DMY` format.

- **Internal Flags** (32-35)
  These flags are not supported and will always test clear.

- **Number of Decimal Places** (36-39)
  These 4 flags comprise a nibble which represents the number of decimal places the `FIXED` input/display format will use.

- **Input/Display Format** (40-41)
  Both flags clear = `SCI` input/display format.
  Flag 41 set only = `ENG` input/display format.
  Flag 40 set only = `FIXED` input/display format.
  Both flags set = Other format (see flags 64-71).

- **Angular Format** (42-43)
  Both flags clear = Degrees format.
  Flag 43 set only = Radians format.
  Flag 42 set only = Gradients format (not supported).

- **Continuous ON** (44)
  This flag is not supported but will always be set.

- **Alpha Mode Flags** (48)
  This flag will be set when the calculator is in `ALPHA` mode, and cleared when not.

- **Message in MainLED** (49)
  This flag will be set when the calculator has a message displayed in the MainLED, and cleared when not.

- **Printer Exists** (55)
  This flag is set when the calculator has a print buffer open.

- **All other Internal Flags** (between 44and 54)
  These flags are not supported and will always test clear.

CCS2 FLAGS (56 thru 79) Not found in HP-41C

These read-only flags are not found in an HP-41C and may conflict with newer HP models which support more than the standard 55 number of HP-41 flags. But we thought you might find them useful in CCS.

- **Alarm Set** (56)
  This flag is set when there is a Macintosh alarm pending (the alarm was either set by the calculator or the ***Alarm Clock*** desk accessory).

- **Spooling to Disk** (57)
  This flag is set when the calculator has a disk buffer open.

- **Line Numbering** (58)
  This flag is set when the calculator is set to display line numbers on the Paper tape or disk/print buffers.

- **Marginals** (59)
  This flag is set when the calculator is set to display marginals on the Paper tape or disk/print buffers.

- **Music ON** (60)
  This flag is set when the calculator is set to play a tone with each key press.

- **Stop Watch Running** (61)
  This flag is set when the calculator is running a stopwatch timer.

- **RPN/SAN** (62)
  This flag is set when the calculator is in Reverse Polish Notation (`RPN`) mode, and cleared when the calculator is in Standard Algebraic Notation (`SAN`) mode.

- **Spare** (63)
  This flag is not currently used and will always test clear.

- **CCS Input/Display Mode** (64-71)
  These 8 flags form a byte which represents a value from 1 through 17 where:

1 = INTEGER	7 = SCI	13 = HRSMIN
2 = NORMAL	8 = DEGREES	14 = TIME
3 = FIXED	9 = HEX	15 = FTINCH
4 = DOLLARS	10 = OCTAL	16 = YDFEET
5 = POUNDS	11 = BINARY	17 = ENG
6 = YEN	12 = DOZENAL	

- **More Spares** (72-79)
  These flags are not currently used and will always test clear.

Chapter 6

Finished Calculators: Functions

This chapter describes all the functions that may be incorporated into any calculator. To see a particular function used in context, read the ***Example Calculations*** chapter.

Our functions are organized in groups. These are the same groupings as you will find in the *Wiring* dialog boxes (see Figures 4-15 through 4-21):

- **Alpha Digits** used to enter data into the Alpha Register
- **Bit Operations** manipulating numbers at the bit level
- **Constants** used to key-in constant data
- **Clearing** used to clear various registers and flags
- **Date/Time** used to add and subtract dates and times
- **External** custom functions written by other programmers
- **Financials** Annuities, Bonds, Present and Future values
- **Mathematics** Math and Trigonometric functions
- **Programming** used to control program scripts
- **Register** used to manipulate registers
- **Scripts** are explained in Chapter 8
- **Statistics** deviations and averages
- **System** functions control the calculator's operation
- **Clickstop Switches** control modes with multiple settings
- **Pushbuttons** toggle simple *ON/OFF* modes
- **SLEDs** used to display registers
- **Selector SLEDs** used to display registers
- **Calendars** used to display dates

- **Prompt Functions** which require additional arguments
- **Unwireable Functions** which are used only in scripts

Alpha Digits

Alpha digits are any ASCII character which can be typed via the Macintosh keyboard. Alpha digits are only accepted as valid input in ALPHA mode. While in ALPHA mode, the keyboard is ***unmapped*** (typing keystrokes will not invoke the functions the keystrokes are normally mapped to). However, alpha digits may be wired to keys.

Constants

Constants are used to enter data, and may be wired to keys. Basic constants include:

- **0-9** used in all input/display formats except OCTAL (allows only 0 through 7) and BINARY (allows only 0 and 1).
- **A-F** used in two input/display formats: HEX (allows all) and DOZENAL (allows A and B).
- π = 3.14159265358979323
- **e** = 2.71828182845904523

Internally, CCS always represents a number with 18 significant digits of accuracy. If you enter a number with more than 18 significant digits, the additional digits will be treated as zeroes. For instance, key in the number: 1234567890123456789 and press the ENTER key. The number will be changed to 123456789012345678**0**. The last digit, a 9, was beyond the 18 significant digits, so it became a 0.

Separators

Separators are also included in our ***Constants*** grouping. Separators are used while entering digits and include:

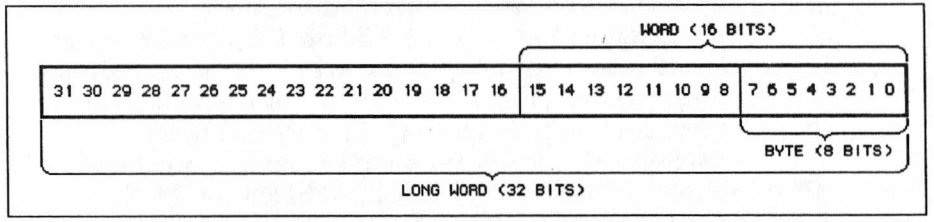
- **Left and Right parenthesis** control the precedence of math operations only when in Standard Algebraic Notation (SAN). For example: 3+(2*5)=13. Without the parentheses the equation is calculated from left to right as the numbers and operators are typed in: 3+2*5=25. There is no limit to the number of times you ***nest*** (put parenthesis inside parenthesis). Here is an example of nested parenthesis: ((8+(3*2))+2)=7. You will not be able to see the parentheses in the MainLED as you enter them.
- **Colon** (:) is used to separate hours, minutes, and seconds in several input/display formats: TIME, HRSMIN, DEGREES.
- **Radix** (or decimal point) is used to separate the whole part of a number from the fractional part. In the U.S., the radix character is a period. In other countries, it is often a comma. See the flags section of Chapter 5 for more information.
- **Caret** (^) is used to separate yards from feet in two input/display formats: YDFEET and FTINCH.
- **Single Quote** (') is used to separate feet from inches in two input/display formats: YDFEET and FTINCH.

Changing display formats will not change the numbers stored in your registers. For instance, if you key in 123.456 while in FLOAT display mode, then change to INTEGER display format, the MainLED will show 123 even though the calculator is actually remembering 123.456. To verify, change back to FLOAT mode.

If you enter a constant or separator which is not consistent with the current input display format, your calculator will beep. For instance, the only valid constant input in BINARY mode would be either a one or a zero.

Bit Operations

This section assumes you know something of binary formatted numbers. Bits are numbered from 0 to 31 (see below). Each of these operations will be affected by the ***word size*** parameter (see `WSIZE` and `WSTAT` functions).



We suggest that in addition to the MainLED, you wire a SLED to display the X-Register in `BINARY` format. That way, you can see exactly which bits these operations affect, while still being able to enter data in `INTEGER` or `HEX` format.

❖ **AND** Logical AND (programmable)

Combines the numbers in the X- and Y-Registers, leaving 1-bits only in the positions where both operands had 1s. The Stack is dropped, and the result returns in the X-Register.

❖ **ASL** Arithmetic Shift Left (programmable)

Shifts the number in the X-Register, one bit to the left, placing the result in the X-Register. The bit being shifted in from the right is always a zero bit.

❖ **ASR** Arithmetic Shift Right (programmable)

Shifts the number in the X-Register, one bit to the right, placing the result in the X-Register. The bits shifted in from the left are copies of the original sign (high order) bit.

❖ **BCHG** Bit Change (programmable)

Toggles a specified bit of the number in the Y-Register. The bit is specified by the number in the X-Register. The Stack is dropped, and the result returns in the X-Register.

❖ **BCLR** Bit Clear (programmable)

Operates like `BCHG`, except `BCLR` clears the specified bit.

❖ **BITS** Bit Count (programmable)

Sums the number of bits set in the X-Register. The bit count is returned in the X-Register, and the original bit pattern (number) is moved to the LastX-Register.

❖ **BSET** Bit Set (programmable)

Operates like `BCHG`, except `BCLR` sets the specified bit.

❖ **BTST** Bit Test (programmable)

Tests a specified bit of the number in the Y-Register. The bit is specified by the number in the X-Register. After the test, the stack will drop, leaving the number being tested in the X-Register. The MainLED will display either `TRUE` or `FALSE`. Use the `CLD` (clear display) function to clear the result. `BTST` may be used in program scripts to control conditional branching.

❖ **EXT** Sign Extend (programmable)

Extends the sign bit. If the word size is 8 bits, bit 7 (the sign bit) is extended through bit 15. If in word size is 16 bits, bit 15 (the sign bit) is extended through bit 31. `EXT` is useful when entering values in `HEX` format: `EXT` extends the sign bit so an `INTEGER` format view of the value could be seen as a negative number.

For example, with word size set to 8 bits, key-in `FF` while in `HEX` input/display format. `FF` displays as `255` in `INTEGER` format. After pressing `EXT`, the number would be displayed as `-1` if in `INTEGER` format.

❖ **LJX** Left Justify Word in X-Reg (programmable)

Shifts the number in the X-Register left, filling with zero bits, until the high order bit is set. The stack is lifted, leaving the result in the Y-Register. The X-Register will contain the number of shifts used to justify the number. An attempt to left justify zero will result in an error.

❖ **LSL** Logical Shift Left (programmable)

This is the same function as `ASL`, and is included so `LSR` does not get lonely.

❖ **LSR** Logical Shift Right (programmable)

Shifts the number in the X-Register, one bit to the right, placing the result in the X-Register. Unlike `ASR`, the bits shifted in from the left are always zeros.

❖ **MASKL** and **MASKR**
Mask Left and Mask Right (programmable)

These masking functions will work in any of the input/display formats, and create masks which can be used in conjunction with `AND`, `OR`, and `XOR` functions.

Each `MASK` function uses the number in the X-Register as the ***bit count***; the number of bits to be set in the resulting mask. The Mask is returned in the X-Register, overwriting the original bit count.

MASKL Mask Left (programmable)

Creates a mask which is left justified within the set word size (`BYTE`, `WORD`, `LONG`). For instance, with a word size of 16 bits (`WORD`), and a bit count of 5, `MASKL` will return: `11111000 00000000`.

MASKR Mask Right (programmable)

Creates a mask which is right justified. For instance, with a word size of 16 bits (`WORD`), and a bit count of 5, `MASKR` will return: `00000000 00011111`.

❖ **NOT** Logical NOT (programmable)

Inverts every bit of the number in the X-Register: each 1-bit becomes a 0-bit, and vice versa. The stack does not move.

❖ **OR** Logical OR (programmable)

Combines the numbers in the X- and Y-Registers, leaving 1-bits in the positions where either operand had 1s. The stack is dropped, and the result returns in the X-Register.

❖ **RLC** Rotate Left through Carry (programmable)

Rotates the number in the X-Register one bit to the left. The carry bit is shifted in from the right, and the leftmost bit is shifted out to the carry bit. See the Status Flags section of Chapter 5 for more about the carry bit.

❖ **RLCN** Multiple RLC (programmable)

Rotates (using `RLC`) the number in the Y-Register left by the number of bits specified in the X-Register.

❖ **RLN** Multiple Rotate Left (programmable)

Rotates (using `ROL`) the number in the Y-Register left by the number of bits specified in the X-Register.

❖ **ROL** Rotate Left (programmable)

Rotates the number in the X-Register left one bit. The bit being shifted out on the left side is shifted back in from the right side. The stack does not move.

❖ **ROR** Rotate Right (programmable)

Rotates the number in the X-Register right one bit. The bit being shifted out on the right side is shifted back in from the left side. The stack does not move.

❖ **RRC** Rotate Right through Carry (programmable)

Rotates the number in the X-Register one bit to the right. The carry bit is shifted in from the left, and the rightmost bit is shifted out to the carry bit.

❖ **RRCN** Multiple RRC (programmable)

Rotates (using RRC) the number in the Y-Register right by the number of bits specified in the X-Register.

❖ **RRN** Multiple Rotate Right (programmable)

Rotates (using ROR) the number in the Y-Register right by the number of bits specified in the X-Register.

❖ **WSIZE** Set Word Size (programmable)

Prompts you for the word size that the bit operations will operate on. Word size is specified as the number of bits: `8`, `16` or `32`. Invalid input will not change the word size. During construction, you can set the default word size using the **Preferences** item from the **Calculator** menu.

❖ **WSTAT** Word Size Status (programmable)

Displays the current word size in the MainLED as: BYTE (8 bits), WORD (16 bits) or LONG (32 bits). Use the CLD (clear display) function to clear the result. This function does not affect any registers.

❖ **XOR** Logical Exclusive OR (programmable)

Combines the numbers in the X- and Y-Registers, inverting bits only in the positions where the two operand bits are different. The stack is dropped, and the result returns in the X-Register.

Another view of this function is XORing with `0` leaves a bit unchanged, while XORing with 1 inverts the bit. For example, with `10010101` in the Y-Register and `10001101` in the X-Register, XOR will leave `00011000` in the X-Register.

Clearing Functions

❖ **CL** "Smart" Clear (programmable)

Smart Clear is the functionally the same as pulling down *Clear* from the *Edit* menu. In normal calculating mode, CL clears the X-Register. While entering a number (before another function has been invoked), CL is the same as CE (clear entry). In TEXT and PRGM modes, CL clears the selection on the Paper tape. In ALPHA mode, CL clears the Alpha Register.

❖ **CE** Clear Entry (not programmable)

In normal calculating mode, CE clears the X-Register but saves the pending operation. For instance, if you had typed:

```
10 + 2 CE 3 =
```

the answer would be 13. By contrast, typing:

```
10 + 2 CLX 3 =
```

would leave an answer of 3, with 10 still in the Y-Register.

❖ **CLA** Clear Alpha Register (programmable)

Clears all 24 characters in the Alpha Register.

❖ **CLC** Clear Calculator (programmable)

Clears all Stack Registers, all Memory Registers, the Alpha Register, and all pending operations.

❖ **CLD** Clear Display of Message (programmable)

Clears the display of an *error* or *status message* which appears in the MainLED . An error message is the result of any error that occurs (like divison by zero, for example). A status message is the result of a test operation (like X>0?) which returns TRUE or FALSE. These messages do not affect the Stack Registers, but need to be cleared before you can view the contents of the MainLED again. Clicking on no parts at all is the same as pressing CLD. Pressing any part will clear the display, but then the function will immediately be executed.

❖ **CF** Clear Flag nn (programmable)

Clears the flag you specify in answer to the prompt in the MainLED . Flags 30-80 cannot be set directly; they are set by calculator functions.

CLF ❖ **CLRF** Clear Financial Regs (programmable)

Clears those Memory Registers which were allocated as Financial Registers.

CLM ❖ **CLRG** Clear Memory Registers (programmable)

Clears all Memory Registers.

❖ **CLP** Clear Prefix (not programmable)

Clears BOT and TOP shift modes.

CLS ❖ **CLST** Clear Stack Registers (programmable)

Clears all Stack Registers, and any pending operations.

❖ **CLX** Clear X-Register (programmable)

Clears the X-Register and any pending operation.

❖ **CLΣ** Clear Summations (programmable)

Clears those Memory Registers which were allocated as Statistics Registers.

Date/Time Functions

All *Date* functions use either the DMY or MDY decimal date formats. Note that leading and trailing zeroes may be omitted if you desire. When working in DMY or MDY formats, we recommend using the FIXED input/display format with the number of decimal places set to 6 (use the FIX function to set the number of decimal places).

- MM.DDYYYY where January 24, 1984 is entered as 1.241984
- DD.MMYYYY where January 24, 1984 is entered as 24.011984

Most *Time* functions accept input in any input/display format. For instance, in FLOAT format, 1.5 decimal hours is the same as 01:30:00 in TIME format. You do not have to add and subtract times using the HMS format (see below)— it is included here as a convenience for Hewlett-Packard calculator users.

---

1 aDT ❖ **ADATE** Alpha Date (programmable)

Appends the number in the X-Register (assumed to be in DMY or MDY format) to the Alpha Register, where it will be in the form: DD/MM/YY or MM/DD/YY.

---

2 aTM ❖ **ATIME** Alpha Time 12-Hour format (programmable)

Appends the number in the X-Register to the Alpha Register, where it will be in the form: HR:MIN:SEC AM/PM or 24HR:MIN:SEC.

---

3 a24 ❖ **ATIME24** Alpha Time 24-Hour format (programmable)

Appends the number in the X-Register to the Alpha Register, where it will be in the form: 24HR:MIN:SEC regardless of the CLK12 and CLK24 settings. This is useful when you want to display elapsed times and do not want the AM or PM displayed.

---

4 C12 ❖ **CLK12** Clock 12-Hour Format (programmable)

Sets a 12-hour clock display format. This format does not affect how time data should be entered, just the format of the CLOCK display, and ATIME functions.

---

5 C24 ❖ **CLK24** Clock 24-Hour Format (programmable)

Sets a 24-hour clock display format. This format does not affect how time data should be entered, just the format of the CLOCK display, and ATIME functions.

---

6 CLK ❖ **CLKT** Set Clock to show Time only (programmable)

Sets the CLOCK to show the time only.

---

7 CTD ❖ **CLKTD** Set Clock to show Time & Date (programmable)

Sets the CLOCK to show the time and the date.

---

8 DCK ❖ **CLOCK** Display Clock (programmable)

Will display the clock in the MainLED, with time and/or date, depending on the ***clock display setting***. Use CLKT and CLKTD to change the clock display setting. Use the CLD (clear display) function to clear the result. This function does not affect any registers.

---

9 DAT ❖ **DATE** Get Date only (programmable)

Moves the current date to the X-Register (in DMY or MDY format).

---

10 DT+ ❖ **DATE+** Add Number of Days to Date (programmable)

Adds the integer value representing the number of days, in the X-Register, to the date in the Y-Register (assumed to be in DMY or MDY format). The stack is dropped, and the result is returned in the X-Register.

❖ **DDAYS** Delta Days (programmable)

Subtracts the date in the Y-Register from the date in the X-Register (both assumed to be in DMY or MDY format). The stack is dropped, and the difference (in number of days) is returned in the X-Register. The date which was in the X-Register is saved in the LastX-Register.

❖ **DMY** Day-Month-Year Format (programmable)

Sets the decimal date input format to DDMMYYYY, where January 24, 1984 is entered as `24.011984`

❖ **DOW** Get Day of Week (programmable)

Converts the date in the X-Register (assumed to be in DMY or MDY format) to a value representing the day of the week the date would fall on (Monday = 0). The date is saved in the LastX-Register, and the stack does not move.

DOW only works for dates between January 1, 1904 and February 6, 2040.

❖ **GETIME** Return Current Time (programmable)

Lifts the stack and moves the current time (in decimal hours) into the X-Register. To get the current time in HMS format, use the function TIME instead.

❖ **HMS** Decimal hrs to hr-min-sec (programmable)

Converts the number in the X-Register, in decimal hours format, to Hours-Minutes-Seconds format, abbreviated as: HH.MMSS, where HH is hours, MM is minutes, and SS is seconds.

❖ **HMS+** hr-min-sec Plus (programmable)

Adds the number in the X-Register to the number in the Y-Register (both assumed to be in HH.MMSS format). The stack is dropped, and the result is returned in the X-Register.

To add times which are in a decimal hours format, use the regular ***Addition*** function (see the Mathematics section of this chapter). It is less confusing to add decimal hours while in TIME input/display format.

❖ **HMS-** hr-min-sec Minus (programmable)

Subtracts the number in the X-Register from the number in the Y-Register (both assumed to be in HH.MMSS format). The stack is dropped, and the result is returned in the X-Register.

To subtract times which are in a decimal hours format, use the regular ***Subtraction*** function (see the Mathematics section of this chapter). It is less confusing to subtract decimal hours while in TIME input/display format.

❖ **HR** hr-min-sec to Decimal Hours (programmable)

Converts the number in the X-Register (assumed to be in Hours-Minutes-Seconds format) to decimal hours format.

❖ **MDY** Month-Day-Year Format (programmable)

Sets the decimal date input format to MMDDYYYY, where January 24, 1984 is entered as `1.241984`

❖ **SETDATE** Set Clock Date (programmable)

Sets the Macintosh date (as viewed in the ***Alarm Clock*** and ***Control Panel*** desk accessories) to the decimal date (see MDY and DMY for date formats) in the X-Register. The stack does not move.

21 STM ❖ **SETIME** Set Clock Time (programmable)

Sets the Macintosh time (as viewed in the ***Alarm Clock*** and ***Control Panel*** desk accessories) to the decimal time in the X-Register, where `1.5` in `FLOAT` display format would be `01:30:00` in `TIME` format. The stack does not move.

---

22 TIM ❖ **TIME** Return Current Time (programmable)

Lifts the stack and moves the current time (in `HMS` format) into the X-Register. To get the current time in decimal hours, use the function `GETIME` instead.

External Functions

This wiring category allows you to wire ***external*** functions to keys. Most external functions do not come with CCS; they are written by third party programmers in Pascal, C, or Assembly languages, and then installed into CCS by the third party. If you are an experienced programmer and interested in writing your own external function, consult Appendix 7 for more information.

If your copy of CCS contains any external functions, their function names will appear in the wiring dialog. Wire them to a key as you would any other function. CCS comes with one external function called `CUBE` (which is described below).

External functions can be accessed in scripts via their function names. In order to build a script properly, both the calculator being using to build the script, and the one being used to run the script must contain keys wired to the external functions which are referenced within the script (except in Test mode).

❖ **CUBE** X Cubed (programable)

Calculates the cube of the number in X-Register, and returns the result in the X-Register.

The cube of a number is the number times itself three times. For example, the `CUBE` of `10` is: `1000` (which is `10*10*10`).

also: SWATCH 'Racing Seconds Stop Watch'
RANDOM
A5-3 REGMOVE
REGSWAP

CCS2 Release Notes -- Read Me

Financial Functions:

Many of the financial functions described here work in conjunction with other financial functions. To fully understand how the financials function as a group, we strongly recommend that you read the Financial Calculations section of Chapter 7, which uses the functions in context. Many financials make use of a special group of Memory Registers called ***Financial Registers*** (see Figure 5-2).

❖ **12×** Get 12 Times X (programmable)

Multiplies the number in the X-Register by 12 and copies the product into the financial register N.

❖ **12/** Get X Divided by 12 (programmable)

Divides the number in the X-Register by 12 and copies the quotient into the financial register I.

❖ **AMORT** Amortization (programmable)

Using the number of periods in the X-Register, amortizes x number of periods using values stored in PMT, I, PV Financial Registers. Updates values in registers PV and N.

❖ **BEG** Set Payment Mode to Begin (programmable)

Sets payment mode to Begin for compound interest calculations involving payments.

❖ **CFJ** Cash Flow J (programmable)

Stores all cash flows except the initial one for discounted cash flow problems.

❖ **CFO** Initial Cash Flow (programmable)

Stores the initial cash flow for discounted cash flow problems.

❖ **DB** Declining-Balance Depreciation (programmable)

Calculates depreciation and the remaining depreciable value using the declining balance method of depreciation. Before calculating the amount of depreciation, the following amounts must be entered:

The original cost using PV,

The salvage value using FV, enter 0 using FV for assets with no salvage value,

The expected useful life in years using N, and

The declining balance factor (as a percentage) using I.

Once the above amounts have been entered, then enter the year for which depreciation is to be calculated and press DB. The calculated amount of depreciation is displayed in the MainLED. To view the remaining depreciable value, press XSWAPY to swap the X-Register with the Y-Register.

❖ **ENDP** Set Payment Mode to End (programmable)

Sets payment mode to End for compound interest calculations involving payments.

❖ **FV** Future Value (programmable)

Stores or computes the future value for financial calculations.

❖ **I** Interest (programmable)

Stores or computes the interest rate per compounding period for financial calculations.

❖ **INST** Simple Interest (programmable)

Calculates simple interest on both 365- and 360- day bases. INST requires amounts in the N,I, & PV Financial Registers. 360-day simple interest is returned in the X-Register. 365-day simple interest is returned in the Z-Register. Principal amount is returned in the Y-Register.

❖ **IRR** Internal Rate of Return (programmable)

Calculates Internal Rate of Return for up to 20 uneven cash flows, and intitial investment using values entered using CFO, CFJ, and NJ.

❖ **N** Number of Periods (programmable)

Stores or computes the number of periods for financial calculations.

❖ **NJ** Store Number of Periods of Cashflow (programmable)

Stores the number of times (maximum of 99) a cash flow occurs for NPV and IRR calculations.

❖ **NPV** Net Present Value (programmable)

Calculates the Net Present Value for a series of cash flows using values stored with CFO, CFJ, and NJ.

❖ **PMT** Payment (programmable)

Stores or computes the periodic payment for financial calculations.

❖ **PV** Present Value (programmable)

Stores or computes the present value for financial calculations.

❖ **SL** Calc Straight Line Depreciation (programmable)

Calculates depreciation and the remaining depreciable value using the straight line method of depreciation. Before calculating the amount of depreciation, the following amounts must be entered:

Calculates depreciation and the remaining depreciable value The original cost using PV, and

The salvage value using FV, enter 0 using FV for assets with no salvage value, and

The expected useful life in years using N.

Once the above amounts have been entered, then enter the year for which depreciation is to be calculated and press SOYD. The calculated amount of depreciation is displayed in the MainLED. To view the remaining depreciable value, press XSWAPY to swap the X-Register with the Y-Register.

❖ **SOYD** Sum Of Years Depreciation (programmable)

Calculates depreciation and the remaining depreciable value using the sum-of-the-years method of depreciation. Before calculating the amount of depreciation, the following amounts must be entered:

The original cost using PV,

The salvage value using FV, enter 0 using FV for assets with no salvage value, and

The expected useful life in years using N.

Once the above amounts have been entered, then enter the year for which depreciation is to be calculated and press SOYD. The calculated amount of depreciation is displayed in the MainLED. To view the remaining depreciable value, press XSWAPY to swap the X-Register with the Y-Register.

❖ **FMOVE** Allocate Financial Registers (programmable)

Moves the block of 26 Financial Registers. The first Financial Register will be the Memory Register number you specify when prompted. In a script, you must supply the Memory Register number on the same line as the `FMOVE` instruction with a space between the two. If you do not supply a Memory Register number, the *Default Memory Register* will be assumed.

❖ **FREG?** Locate Financial Registers (programmable)

Lifts the stack and moves the number representing the first Financial Memory Register into the X-Register.

Important We strongly recommend that you read the Financial Calculations section of Chapter 7 for a clear understanding of the Financial functions.

Math Functions

❖ * Multiplication (programmable)

Multiplies the number in the Y-Register by the number in the X-Register. The stack is dropped, and the result is returned in the X-Register.

❖ + Addition (programmable)

Adds the number in the X-Register to the number in the Y-Register. The stack is dropped, and the result is returned in the X-Register.

❖ - Subtraction (programmable)

Subtracts the number in the X-Register from the number in the Y-Register. The stack is dropped, and the result is returned in the X-Register.

❖ / Division (programmable)

Divides the number in the Y-Register by the number in the X-Register. The stack is dropped, and the result is returned in the X-Register.

❖ **%** Percent (programmable)

Computes X% of Y and retains the Y value in the Y-Register. For example, if the Y-Register contains `200` and the X-Register contains `25`, the `%` function will return an answer of `50` (because 25% of 200 = 50).

❖ **%CH** Percent Change (programmable)

Computes the percent of change between the number in the Y-Register and the number in the X-Register. For example, if the Y-Register contains `200` and the X-Register contains `25`, the `%CH` function will return an answer of `-87.5`.

❖ **%T** X Percent of Number (programmable)

Computes the percent that X is of the number in the Y-Register. For example, if the Y-Register contains `200` and the X-Register contains `25`, the `%T` function will return an answer of `12.5` (because 25 = 12.5% of 200).

❖ **1/X** Reciprocal (programmable)

Computes the reciprocal of the number in the X-Register and returns the result in the X-Register. If you think of the number in the X-Register as a fraction, reciprocal swaps the numerator with the demoninator. For example, the reciprocal of 2 is one half.

❖ **10TOX** Common Exponential (programmable)

Calculates 10 rasied to the power of the number in the X-Register and returns the result in the X-Register.

❖ **ABS** Absolute Value (programmable)

Calculates the absolute value of the number in the X-Register, and returns the result in the X-Register. Absolute value has the affect of stripping of a negative sign.

❖ **ACOS** Arc Cosine (programmable)

Calculates the Arc Cosine of the number in the X-Register, and returns the result in the X-Register.

❖ **ASIN** Arc Sine (programmable)

Calculates the Arc Sine of the number in the X-Register, and returns the result in the X-Register.

❖ **ATAN** Arc Tangent (programmable)

Calculates the Arc Tangent of the number in the X-Register, and returns the result in the X-Register.

❖ **CHS** Change Sign (programmable)

Changes the sign of the number in the X-Register. If you pressed `EEX` while entering digits, `CHS` changes the sign of the exponent instead of the entire number. `CHS` will not terminate digit entry or cause the stack to move.

In scripts, the `CHS` function is abbreviated as `-` when used within a number, like this: `-1.234`. A negative exponent would be entered as `1.234 E-56`. The minus sign can appear anywhere on the line. If it appears before the `E`, the entire number will be negative. If the minus sign appears after the `E`, the exponent will be negative.

❖ **COS** Cosine (programmable)

Calculates the Cosine of the number in the X-Register, and returns the result in the X-Register.

❖ **CSC** Cosecant (programmable)

Calculates the Cosecant of the number in the X-Register, and returns the result in the X-Register.

❖ **CTN** Cotangent (programmable)

Calculates the Cotangent of the number in the X-Register, and returns the result in the X-Register.

❖ **D-R** Degrees To Radians (programmable)

Converts the number in the X-Register which expresses an angle in degrees to a number that expresses the same angle in radians. The result is returned in the X-Register.

❖ **ETOX** Natural Exponential (programmable)

Calculates e raised to the power of the number in the X-Register, and returns the result in the X-Register.

❖ **FACT** N! Factorial (programmable)

Calculates the value of N! Factorial using the number contained in the X-Register. The value for N must be an integer in the range $0 < N < 99$. The result is returned in the X-Register.

❖ **FRC** Fractional Part of X (programmable)

Extracts the fractional part of the number in the X-Register, and displays the result in the X-Register. For example: the fractional part of `1234.5678` is `.5678`.

❖ **INT** Integer Part of X (programmable)

Extracts the integer part of the number in the X-Register, and displays the result in the X-Register. For example: the integer part of `1234.5678` is `1234`.

❖ **LN** Natural Logarithm (programmable)

Calculates the natural logarithm (base *e*) of the number in the X-Register, and returns the result in the X-Register.

❖ **LOG** Common Logarithm (programmable)

Calculates the common logarithm (base 10) of the number in the X-Register, and returns the result in the X-Register.

❖ **P-R** Polar To Rectangular Coordinates (programmable)

A point on a plane can be described by either ***polar*** or ***rectangular*** coordinates. Polar coordinates are expressed as: r (magnitude) and θ (angle). Rectangular coordinates are expressed as: x (horizontal) and y (vertical).

Using polar coordinates r in the X-Register and θ in the Y-Register, `P-R` calculates the rectanglular coordinates for the same point and stores x in the X-Register and y in the Y-Register. The vertical coordinate y will take on the sign of the Polar angle θ. The LastX-Register will contain r.

❖ **R-D** Radians To Degrees (programmable)

Converts the number in the X-Register which expresses an angle in radians to a number that expresses the same angle in degrees. The result is returned in the X-Register.

❖ **R-P** Rectangular To Polar Coordinates (programmable)

Using rectangular coordinates x in the X-Register and y in the Y-Register, `R-P` calculates the polar coordinates for the same point and stores r in the X-Register and θ in the Y-Register. The Polar angle θ will take on the sign of the vertical coordinate y. The LastX-Register will contain x.

See `P-R` (above) for an explanation of x, y, r and θ.

❖ **RND** Round Mantissa (programmable)

Rounds the number in the X-Register to the number of decimal places you supply when prompted. For example, `123.456789` will be rounded to `123.46` if you supply `2` as the number of decimal places when prompted. In a script, you must supply the number of places on the same line as the `RND` instruction with a space between.

Note Unlike `RND`, input/display formats do not actually change any of the numbers contained in the Stack or Memory Registers. Displaying a number in `INTEGER` format does not change the actual number in the X-Register.

❖ **SEC** Secant (programmable)

Calculates the Secant of the number in the X-Register, and returns the result in the X-Register.

❖ **SIGN** Sign of X (programmable)

Tests the X-Register and returns `-1` if `X` is negative, or `1` if `X` is positive or zero. The answer is returned in the X-Register, and the original X-Register is saved in the LastX Register.

❖ **SIN** Sine (programmable)

Calculates the Sine of the number in the X-Register, and returns the result in the X-Register.

❖ **SQRT** Square Root of X (programmable)

Calculates the square root of the number in the X-Register, and returns the result in the X-Register. Taking the square root of a negative number will generate a DATA ERROR.

❖ **TAN** Tangent (programmable)

Calculates the Tangent of the number in the X-Register, and returns the result in the X-Register.

❖ **XROOTY** Xth Root of Y (programmable)

Computes the *x*th root of the number in the Y-Register, using the number in the X-Register as *x*. The stack is dropped, and the result is returned in the X-Register.

❖ **XTO2** X Squared (programmable)

Calculates the Square of the number in the X-Register, and returns the result in the X-Register.

❖ **YTOX** Y To The X Power (programmable)

Raises the number in the Y-Register, to the power of the number in the X-Register. The stack is dropped, and the result is returned in the X-Register.

Programming (Scripting) Functions

❖ **DSE** Decrement & skip if ≤ (programmable)

Used to direct program script execution. By use of a control number stored in a Memory Register you specify, DSE allows a script to loop a specific number of times. The format of this control number is `iiiii.fffcc`, where:

`iiiii`	***is the Current Counter Value*** can be any positive number up to 5 digits; defaults to `0` if not specified
	is a decimal point
`fff`	***is the final counter value*** must be a positive 3-digit number; defaults to `0` if not specified
`cc`	***is the increment (decrement) value*** must be a positive 2-digit number; defaults to `1` if not specified

DSE takes the number `iiiii.fffcc` and decrements the value `iiiii` by cc. The Memory Register you specify will be updated with the new `iiiii` value. DSE then tests the value `iiiii` against the value `fff`. If `(iiiii-cc)≤fff`, the script line following the DSE instruction is skipped.

When invoked while in normal calculation mode, you will be prompted for a Memory Register number, and DSE will test the value `iiiii` against the value `fff`, returning a TRUE or FALSE message in the MainLED, which may be cleared using the CLD (clear display) function.

In a script, you must supply the Memory Register number on the same line as the DSE instruction with a space between the two. If you do not supply a Memory Register number, the *Default Memory Register* will be assumed.

❖ **FC?** Flag nn clear? (programmable)

Used to direct program script execution. Tests a flag which you specify. If the flag is not ***clear***, the script line immediately following the FC? instruction will be skipped, otherwise it will be executed.

When invoked while in normal calculation mode, you will be prompted for a flag number, and FC? will return a TRUE or FALSE message in the MainLED, which may be cleared using the CLD (clear display) function.

In a script, you must supply the flag number on the same line as the FC? instruction with a space between the two. If you do not supply a flag number, the script line immediately following the FC? instruction will always be executed.

❖ **FC?C** Flag nn clear? (programmable)

The same instruction as FC? except that the flag you specify will always be ***cleared*** after it has been tested.

❖ **FS?** Flag nn set? (programmable)

Tests a flag which you specify. If the flag is not ***set***, the script line immediately following the FS? instruction will be skipped. All other aspects of FS? are the same as FC?.

❖ **FS?C** Flag nn set? (programmable)

The same instruction as FS?, except that the flag you specify will always be ***cleared*** after it has been tested.

❖ **GTO** Go To (programmable)

Unconditionally branches to the label you specify. In normal calculation mode, the calculator will prompt you for the label, then immediately execute the script at the label you specify. In a script, you must supply the label on the same line as the GTO instruction with a space between the two. See Chapter 8 for a detailed discussion of script flow to see how the RTN and END instructions will affect the GTO instruction.

❖ **ISG** Increment & Skip if Greater Than (programmable)

Exactly the same function as DSE, except that ISG takes the number iiiii.fffcc and increments the value iiiii by cc. If (iiiii+cc)>fff, the script line following the ISG instruction is skipped.

❖ **RTN** Return from Subroutine (programmable)

Returns from the subroutine called by a XEQ instruction in a program script. When invoked outside a script, RTN does nothing; it is wireable merely to make script editing easier.

❖ **R/S** Run/Stop Script (not programmable)

Stops script execution if clicked. From the keyboard, hold down the ⌘ and period keys instead. Although this function is not programmable, the equivalent function name is STOP.

If no scripts are running, R/S executes the current script, pointed to by the ***current script pointer*** (explained in the ***Building Program Scripts*** chapter). If there are no scripts in memory, R/S returns an error message in the MainLED.

❖ **SF** Set Flag nn (programmable)

Sets a flag which you specify. When invoked while in normal calculation mode, you will be prompted for a flag number which must be between 0 and 79 inclusive. In a script, you must supply the flag number on the same line as the SF instruction with a space between the two. If you do not supply a flag number, SF does nothing. If you supply a flag number which is out of bounds, SF returns a Non Existent error message in the MainLED.

❖ **SST** Single Step through Script (not programmable)

Executes a single line of the current script, pointed to by the ***current script pointer*** (explained in the ***Building Program Scripts*** chapter).

❖ **XEQ** Execute Function or Script (programmable)

This is perhaps the most powerful key you can build into your calculator. The XEQ key can be used to access almost any CCS function by typing in its ***function name*** when prompted.

When XEQ is invoked from a wired key, the calculator will prompt you for a function name or script label.

If you supply a function name, the function is immediately executed. Note that wiring an XEQ key during construction will trigger CCS to save all functions, despite the setting of the "Save All Functions" option in the ***Preferences*** dialog box (Figure 4-39).

If you supply XEQ with the name of a script already in memory, the script is executed starting at line one. If you supply a label within the current script, the current script is executed starting with the line containing the label.

When XEQ is used within a program script, the calculator will only execute a script label—not a function.

In program scripts, each function name should appear on its own line, and is not preceeded by an XEQ instruction.

Supply the label on the same line as the XEQ instruction with a space between the two. The label is treated as a subroutine. See the ***Building Calculator Scripts*** chapter for a detailed discussion of script flow to see how the RTN and END instructions will affect the XEQ instruction.

❖ **X=0?** Test: X = 0?
X≠0? Test: X ≠ 0?
X<0? Test: X < 0?
X>0? Test: X > 0?
X<=0? Test: X ≤ 0?

X=Y? Test: X = Y?
X≠Y? Test: X ≠ Y?
X<Y? Test: X < Y?
X>Y? Test: X > Y?
X<=Y? Test: X ≤ Y?

Compare Instructions (programmable)

These ***compare*** functions all test a specific condition by comparing two values, and return either TRUE or FALSE as a result.

When not running a script, the result will be displayed as a message in the MainLED. Use the CLD (clear display) function to clear the message.

In a program script, you may use these instructions to divert the script to another line number. The script line immediately following the conditional test will be skipped if the result of the test is FALSE, or will be executed if the result of the test is TRUE. See the ***Building Calculator Scripts*** chapter for a detailed discussion of script flow and conditional branching.

X=NN? Test: X = Memory Register nn?
X≠NN? Test: X ≠ Memory Register nn?
X<NN? Test: X < Memory Register nn?
X>NN? Test: X > Memory Register nn?
X<=NN? Test: X ≤ Memory Register nn?
X>=NN? Test: X ≥ Memory Register nn?

X=R? Test: X = Memory Register nn?
X≠R? Test: X ≠ Memory Register nn?
X<R? Test: X < Memory Register nn?
X>R? Test: X > Memory Register nn?
X<=R? Test: X ≤ Memory Register nn?
X>=R? Test: X ≥ Memory Register nn?

Compare Instructions (programmable)

The *compare* instructions which end with NN? compare the number in the X-Register with the number in a Memory Register. Place the Memory Register's number (NN) in the Y-Register before executing these compares.

The *compare* functions which end with R? test against a value contained in a Memory Register which you specify. When not running a script, the *prompt function* will cause the calculator to prompt you for a register number, which you must type in before continuing. In a script, you must supply the Memory Register number on the same line as the *compare* instruction with a space between the two. If you do not supply a Memory Register number, the *Default Memory Register* will be assumed.

A=R? Alpha Register = Memory Register nn?

Compares the contents of the Alpha Register with the contents of a specified Memory Register. This is a prompt function (like the ones described immediately above).

Comparing strings of different lengths will result in a FALSE result. Remember that the Memory Registers can contain a maximum of six alpha characters.

Register Functions

Any function which calls for you to specify a Memory Register is termed a ***prompt*** function. When not running a script, you will be prompted for a Memory Register number. In a script, you must supply the Memory Register number on the same line as the prompt instruction with a space between the two. If you do not supply a Memory Register number, the *Default Memory Register* will be assumed.

❖ **ALENG** Alpha Length (programmable)

The number of non-blank characters contained in the Alpha Register is returned in the X-Register, and the stack is lifted.

❖ **ANUM** Alpha Number (programmable)

ANUM scans the Alpha Register for any series of digits that could be considered the representation of a number. ANUM then lifts the stack, converts the string to a number, and moves the number to the X-Register. The Alpha Register is unchanged. If ANUM could not extract a valid number based upon the Alpha Register, zero will be moved to the X-Register.

ANUM scans from the left side of the Alpha Register, looking for the first "numeric" character. The evaluation of numeric data depends upon the input/display format (for instance, in BINARY mode, only 0s and 1s are considered to be numeric characters). ANUM completes the number when it encounters the first non-numeric character.

❖ **APPEND** Append to Alpha Register (programmable)

Normally when entering ALPHA mode, the first character typed will replace the previous contents of the Alpha Register. To prevent this, use the APPEND function just before keying in the first alpha digit.

In a script, APPEND should immediately follow AON, followed by the alpha data (enclosed it quotes).

❖ **ARCL** Alpha Recall (programmable)

Recalls alpha data from a Memory Register you specify, appending the alpha data to the Alpha Register. The MainLED will display `Not Alpha Data` if you specify a Memory Register that does not contain alpha data, and no operation will be performed.

❖ **AROT** Alpha Rotate (programmable)

Shifts all characters in the Alpha Register left by the number of places you specify. Characters shifted out the left are appended to the right side of the Alpha Register. When not running a script, you will be prompted for the number of places. In scripts, the number must appear on the same line as the `AROT` instruction with a space between the two.

❖ **ASHF** Alpha Shift 6 chars Left (programmable)

Shifts all characters in the Alpha Register left six places, effectively removing the leftmost 6 characters.

❖ **ASTO** Alpha Store (programmable)

Stores the leftmost 6 characters of the Alpha Register in a Memory Register which you specify. Memory Registers can hold up to 6 alpha characters.

The MainLED will display `Alpha Data` if you attempt to `RCL` (recall) from a Memory Register into which you moved ***alpha data*** using `ASTO`. This is because `RCL` can only recall a ***number*** from a Memory Register to the X-Register.

❖ **ATOX** Alpha to X-Register (programmable)

Lifts the stack, shifts the leftmost character out of the Alpha Register and places the character's corresponding ASCII code into the X-Register. (See Appendix 6 for a table of Macintosh ASCII values). `ATOX` moves `0` to the X-Register if the Alpha Register contained no characters.

❖ **AVIEW** Alpha View (programmable)

Displays the contents of the Alpha Register in the MainLED until you press `CLD` (clear display) or any other function. This is quite useful in program scripts when you want to prompt the user for input, or display a result as a text rather than a number.

❖ **GETR** Get Memory Registers from saved file (programmable)

`GETR` displays a standard ***GetFile*** dialog box (Figure 4-32) with the names of all the Memory Register files. Open one of the files, and the registers contained in the file will replace those in the calculator.

❖ **LASTX** Recall from Lastx Register (programmable)

Copies the number in the LastX-Register into the X-Register. You can think of `LASTX` as a kind of ***Undo*** function.

❖ **RCL** Recall from Memory Register nn (programmable)

Lifts the stack, then copies a number from a Memory Register which you specify into the X-Register. The MainLED will display `Alpha Data` if you attempt to `RCL` alpha data from a Memory Register into which you moved alpha data using `ASTO` (Alpha Store).

❖ **SAVER** Save Memory Registers in a file (programmable)

`SAVER` displays a standard ***PutFile*** dialog box (Figure 4-34). Name your file and click the ***Save*** button. All the calculator's Memory Registers will be saved in the new file. A sample Memory Register file icon is shown at left.

❖ **STOLAST** Store X to LastX Register (programmable)

Copies the number in the X-Register to the LastX-Register.

❖ **STO** Store X to Memory Register nn (programmable)

Copies the number in the X-Register to a Memory Register which you specify. The contents of the Memory Register will be replaced by the new number.

❖ **ST*** Multiply Memory Register nn by X (programmable)

Multiplies a number in the Memory Register you specify by the number in the X-Register, storing the product in the specified Memory Register. The number in the X-Register is not changed, and the stack does not move.

❖ **ST+** Add X to Memory Register nn (programmable)

Adds the number in the X-Register to a number in the Memory Register you specify, storing the sum in the specified Memory Register. The number in the X-Register is not changed, and the stack does not move.

❖ **ST-** Sub X from Memory Register nn (programmable)

Subtracts the number in the X-Register from a number in the Memory Register you specify, storing the difference in the specified Memory Register. The number in the X-Register is not changed, and the stack does not move.

❖ **ST/** Divide Memory Register nn by X (programable)

Divides the number in the X-Register by a number in the Memory Register you specify, storing the quotient in the specified Memory Register. The number in the X-Register is not changed, and the stack does not move.

❖ **VIEW** Display Memory Register nn in LED (programmable)

Temporarily displays the contents of a Memory Register you specify in the MainLED until you press `CLD` (clear display) or any other function. This function can be used in lieu of wiring a Memory Register Selector SLED.

❖ **XASA** Convert X-Register to Alpha (programmable)

Appends the digits in the X-Register to the Alpha Register as a string of characters. If there is not enough room to append, the Alpha Register will be shifted left enough places to append the entire string of digits.

Recall that the Alpha Register can hold a maximum of 24 characters. Appending additional characters will push the leftmost character out of the Alpha Register and into oblivion.

❖ **XSWAPF** Exchange X-Register and Flags 0 through 7 (programmable)

Exchanges the number in the X-Register with the first 8 user flags. You may then use the Bit Operation functions to test and set particular bits. The stack does not move.

❖ **XSWAPR** Swap X-reg and Memory Register nn (programmable)

Exchanges the number in the X-Register with a number from a Memory Register which you specify. The MainLED will display `Alpha Data` if you attempt to exchange the X-Register with alpha data from a Memory Register which you previously moved alpha data into using `ASTO`.

❖ **XSWAPY** Exchange X- and Y-Registers (programmable)

Exchanges the number in the X-Register with the number in the Y-Register. The stack does not move.

❖ **XTOA** Convert X-Register to Alpha (programmable)

Appends a character that is the ASCII equivalent of the number in the X-Register to the Alpha Register. The number in the X-Register must be an integer from 0 to 255. See Appendix 6 for a table of Macintosh ASCII values.

If the number in the X-Register is greater than 255 or less than 0, the MainLED will display `Data Error`.

If the number in the X-Register is 8, the last character in the Alpha Register will be backspaced (removed) since ASCII value 8 is the backspace character.

Scripts

This wiring category allows you to wire program scripts to calculator keys. Once a script is wired to a key, it becomes a permanent part of the calculator.

You may create your own scripts, or load scripts which were created by others.

Please refer to Chapter 8 for an in-depth discussion of scripting and wiring scripts to keys.

Statistical Functions

❖ **MEAN** Means of Accumulated X & Y Values (programmable)

Calculates the mean (arithmetic average) of the accumulated X- and Y- values. The mean of the X-values is displayed after MEAN is pressed. To display the mean of the Y-values, press XSWAPY to swap the X- and Y-Registers. See Σ+ and Σ− for a description of how to enter the X- and Y- values.

❖ **SDEV** Standard Deviation (programmable)

Calculates the standard deviation of the accumulated X- and Y- values. The standard deviation of the X-values is displayed after the SDEV function is executed. To display the standard deviation of the y-values, press XSWAPY to swap the X- and Y-Registers. See Σ+ and Σ− for a description of how to enter the X- and Y- values.

❖ **Σ+** Summation Plus (programmable)

Accumulates X- and Y- data pairs for statistical calculations. After updating the the statistical registers, Σ+ displays the total accumulated number of data pairs in the X-Register.

❖ **Σ-** Summation Minus (programmable)

Corrects accumulated X- and Y- data pairs for statistical calculations. After updating the the statistical registers, Σ− displays the total accumulated number of data pairs in the X-Register.

Entering statistical data: Prior to accumulating new statistical data, you should clear the statistics registers by pressing CLΣ. In one-value statistical calculations, to enter each data value, key the number into the display and press the Σ+ key.

In two-value statistical calcuations, enter each data pair as follows:

① Key the Y-value into the display and press ENTER
② Key the X-value into the display and press Σ+

Each time the Σ+ key is pressed the calculator does the following:

① Increases the number in R1 by 1 and displays the result.
② Adds the X-value to the number in R2.
③ The square of the X-value is added to the number in R3.
④ Adds the Y-value to the number in R4.
⑤ The square of the Y-value is added to the number in R5.
⑥ The product of the X- and Y-values is added to the number in R6.

The following table shows where the accumulated X- and Y- data pairs are stored in statistics registers R1 through R6:

R1 ΣX summation of X-values.
R2 ΣX2 summation of squares of X-values.
R3 ΣY summation of Y-values.
R4 ΣY2 summation of squares of Y-values.
R5 ΣXY summation of product of X- andY-values.
R6 ΣN number of data points accumulated.

❖ **ΣREG** Allocate Statistical Registers (programmable)

Moves the block of 6 Statistical Registers. The first Statistical Register will be the Memory Register number you specify. In normal calculation mode, you will be prompted for a Memory Register number. In a script, you must supply the Memory Register number on the same line as the ΣMOVE instruction with a space between the two. If you do not supply a Memory Register number, the *Default Memory Register* will be assumed.

❖ **ΣREG?** Locate Statistical Registers (programmable)

Lifts the stack and moves the number representing the first Statistical Memory Register into the X-Register.

❖ **WTDAVG** Weighted Average (programmable)

This is a two-value statistical function. Key-in data pairs using the ENTER and Σ+ functions. The numbers keyed into the Y-Register are the values. The numbers keyed into the X-Register are the weights.

WTDAVG calculates the weighted average of Y-values (in the ΣY register) and X-weights (in the ΣX register). The weighted average is returned in the X-Register.

Chapter 7 contains an example WTDAVG calculation.

Important We strongly recommend that you read the Statistical Calculations section of Chapter 7 for a clear understanding of the Statistical functions.

System Functions

❖ ◄- Backspace (not programmable)

While keying-in digits, backspace removes the last digit entered. When not keying-in digits, backspace will clear the X-Register, except in ALPHA mode where backspace will clear the Alpha Register.

❖ = Equals (programmable)

In SAN mode, = completes any pending operations, and displays a total in the MainLED and the Paper tape. In RPN mode, the total just gets repeated since RPN always displays a running total for you.

❖ **ADV** Insert Blank Line on Paper tape (programmable)

Inserts a blank line on the Paper tape.

❖ **BEEP** Beep Once (programmable)

Beeps the Macintosh speaker one time. Most useful when executed from a program script.

❖ **COPY** Copy to Clipboard (programmable)

This is a "smart" copy function, which is functionally the same as pulling down ***Copy*** from the ***Edit*** menu. In normal calculation mode, COPY copies the X-Register to the Macintosh clipboard. In TEXT and PRGM modes, COPY copies the selection on the Paper tape (if any). In ALPHA mode, COPY copies the Alpha Register.

Please refer to the note which follows the CUT function for more on the way the COPY and CUT functions format text on the Macintosh clipboard.

❖ **CUT** Cut to Clipboard (programmable)

This is a "smart" cut function, which is functionally the same as pulling down ***Cut*** from the ***Edit*** menu. In normal calculation mode, CUT copies the X-Register to the Macintosh clipboard, then clears the X-Register. In TEXT and PRGM modes, CUT copies then clears the selection on the Paper tape (if any). In ALPHA mode, CUT copies then clears the Alpha Register.

Note Hold down the option key when invoking the CUT and COPY functions to prevent "thousands separators" (usually commas), dollar signs and spaces from being copied from the X-Register. Only the digits 0 thru 9, A thru F and the radix (usually a decimal point; period) will be copied.

❖ **DEG** Degrees Mode (programmable)

Selects degrees angular mode for a trigonometric functions. Neither the display or the stack is immediately affected by this function.

❖ **DRAW•** Draw Bullets on Paper tape (programmable)

Inserts a line of bullet characters (•) on the Paper tape.

❖ **ENTER** Y, Enter X (programmable)

Separates X and Y values. Input Y first, press ENTER, then input X. Y will end up in the Y-Register and X will end up in the X-Register. In ***Standard Algebraic Notation*** (SAN), the only time you will need to use the ENTER key is for functions which require two values to be input (such as YTOX). In ***Reverse Polish Notation*** (RPN), use the ENTER function to fill the stack with values before executing functions.

The ENTER function disables stack lift, allowing you to input another number into the X-Register. Normally the act of entering a number into the X-Register first lifts the stack.

❖ **EEX** Enter Exponent (programmable)

While entering numbers in any decimal input/display format, EEX allows you to enter the exponent after you have entered the mantissa. Note that before you press EEX, CHS will change the sign of the entire number. After you press EEX, CHS will change the sign of the exponent only.

In scripts, the EEX function is abbreviated as E when used within a number, like this: `1.234 E56`. A negative exponent would be entered as `1.234 E-56`. The minus sign can appear anywhere on the line. If it appears before the E, the entire number will be negative. If the minus sign appears after the E, the exponent will be negative.

❖ **FIX** Set Number of decimal places (programmable)

FIX prompts you to enter the number of decimal places to display while in FIXED input/display format. Note: FIX does ***not*** subsquently set the input/display format as it does on the HP-41C. Use the CCS2 function FIXED to change the input/display format.

❖ **MEM** Memory Map (programmable)

Displays the ***Memory Map*** dialog box (Figure 6-1), which provides you with information about Memory and Stack Registers.

Number of Stack Registers:	**4**
Number of Memory Registers:	**50**
Default Memory Register:	**0**
Number of Financial Registers:	**26**
Financial Registers start at:	**17**
Number of Statistical Registers:	**6**
Statistical Registers start at:	**11**

Figure 6-1 The Memory Map dialog box

❖ **GETAS** Read ASCII Text File (programmable)

Displays a standard ***GetFile*** dialog box (Figure 4-32) containing the names of text files and folders. Click on the name of the file you wish to read, then click the ***Open*** button. GETAS will read the first 32,000 characters from the file and display them on the Paper tape where you may edit them. The original file remains intact. You must wire a Paper tape in order to use GETAS.

❖ **PASTE** Paste from Clipboard (programmable)

This is a "smart" paste function, which is functionally the same as pulling down ***Paste*** from the ***Edit*** menu. In TEXT and PRGM modes, PASTE replaces the selection on the Paper tape (if any) with the text on the clipboard. If there is no selection range, the text is inserted into the Paper tape.

In ALPHA mode, PASTE "types" each character on the clipboard into the Alpha Register. If the clipboard contains more than 24 characters, only the last 24 pasted will end up in the Alpha Register.

In normal calculation mode, PASTE "types" each character on the clipboard into the X-Register. Any character which is not considered a valid constant in the current input/display format will generate a beep, and will not be pasted into the X-Register.

❖ **RAD** Set Radians Mode (programmable)

Selects radians angular mode for a trigonometric functions. Neither the display or the stack is immediately affected by this function.

❖ **RDN** Roll Stack Down (programmable)

Shifts all stack contents down one register without losing any data. Each Stack Register moves its data to the Register below it. The X-Register (the bottom most register) moves its data to the highest register. The LastX-Register is not changed by RDN.

❖ **RUP** Roll Stack Up (programmable)

Shifts all stack contents up one register without losing any data. Each Stack Register moves its data to the Register above it. The top most register moves its data to the X-Register (the lowest register). The LastX-Register is not changed by RUP.

❖ **SAVEAS** Save Ascii Text File (programmable)

Displays a standard ***SaveFile*** dialog box (Figure 4-34). Type in the name of the file you wish to save and click the ***Save*** button. The contents of the Paper tape will be saved in the *Teachtext* file you create (sample icon is shown at left). You must wire a Paper tape in order to use SAVEAS.

❖ **SELALL** Select All Text on Paper tape (programmable)

Selects all text on the Paper tape and turns TEXT mode *ON*.

❖ **SUBT** Paper tape Subtotal (programmable)

Draws a row of minus signs (a single line) on the Paper tape then displays a total below this row. The Paper tape does not display a running total of addition, subtraction, multiplication and division operations, so it is necessary to press the SUBT, TOT or = (equals) keys to display the total.

❖ **TONE** Tone Using Prompted Value (programmable)

Plays a note that you specify on the Macintosh speaker. This note must be a whole number from 1 to 84. Notes are organized in octaves of 12 notes each, starting 3 octaves below middle C. In DOZENAL input/display format, it is easy to tell which octave is being played. In a two-digit, base 12 number, the 12s digit equals the octave, and the ones digit equals the note within the octave. You may have to turn up the speaker volume using the ***Control Panel*** desk accessory to hear very low notes. In standard calculation mode, you will prompted for a note value. In a script, you must supply the note value on the same line as the TONE instruction with a space between the two. If you do not supply a value for the note, no note will be played.

❖ **TONEX** Tone Using Value in X-Register (programmable)

The same function as TONE, except TONEX uses a rounded copy of the number in the X-Register as the note to play instead of prompting for the value of the tone to play. TONEX does not change the X-Register.

❖ **TOT** Paper tape Total (programmable)

Same as SUBT (subtotal), except TOT draws a row of equals signs (a double line) before drawing the total below the row.

❖ **TPRINT** Print Paper tape (programmable)

Prints all text contained on the Paper tape to your Macintosh's printer. A number of dialog boxes will appear depending on which printer you have selected using the *Chooser* desk accessory.

Unlike the PRINT pushbutton switch, you must have a Paper tape in order to use TPRINT. Also, unlike the PRINT pushbutton, TPRINT does not have to start building a buffer; the Paper tape *is* the buffer.

For more, see the PRINT pushbutton towards the end of this chapter.

Clickstop Switches

Clickstop functions can be accessed via their function names either in response to the XEQ key's prompt, or within a script.

Because it is possible to wire more than one clickstop switch of the same type, a switch will be highlighted (the switch's label will be drawn as white type on a black background) when it is *active* (Figure 6-2). An active switch's displayed setting matches the current state of the mode or format it controls. For instance, an INTEGER switch will be active if the input/display format really is INTEGER. Press a switch to activate it. The last switch pressed will, therefore, always be active.

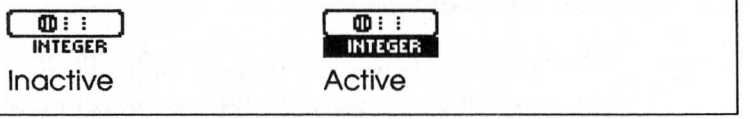


Figure 6-2 Inactive and Active Clickstop Switches

❖ **Input/Display Formats**

Chapter 5 describes these formats in great detail. To access more than five display formats in a single calculator you may wire multiple clickstop switches. If the Annunciator display is stretched wide enough, it will display the function name of the input/display format which can be:

BINARY	FIXED	OCTAL
DEGREES	FTINCH	POUNDS
DOLLARS	HEX	SCI
DOZENAL	HRSMIN	TIMEDSP
ENG	INTEGER	YDFEET
FLOAT	NORMAL	YEN

Note: CCS2 functions ENG, FIXED and SCI do not prompt for the number of decimal places as they do on the HP-41C. Use the FIX function to set the number of decimal places to be displayed in FIXED format.

❖ **Paper Tape Justification**

This switch controls the way text is drawn on the Paper tape. The three options are: ***Left***, ***Centered*** and ***Right***. In PRGM recording mode, text is always flush left— regardless of this switch's setting. Function names are:

```
LEFT      CENTER    RIGHT
```

❖ **RPN/SAN** (Reverse Polish/Standard Algebraic Notation)

Chapters 3 and 5 describe these modes in great detail. The equivalent function names are:

```
RPN           SAN
```

❖ **Degrees/Radians**

Determines whether trigonometric functions expect input and display results as either degrees or radians. See the ***Mathematics Functions*** in this chapter for more. Function names are:

```
DEG           RAD
```


Pushbutton Switches

Pushbutton switches are ***not programmable***; you cannot access them from within a program script. However, some do have equivalent function names which can be used within scripts. We refer to an inverted pushbutton (white letters on black background) as one that is ***ON***. an uninverted pushbutton (black letters on white background) is ***OFF*** (Figure 6-3).

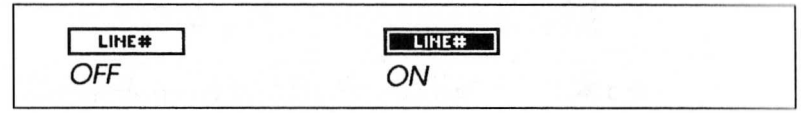


Figure 6-3 Pushbuttons in *OFF* and *ON* states

❖ **ALARM** Set/Clear Alarm (not programmable)

This pushbutton can be used as an alternative to the ***Alarm Clock*** desk accessory. ALARM controls the Macintosh alarm. When an alarm is set, the ALARM pushbutton will be *ON*. When an alarm goes off, the menu will either blink or toggle between the icon and the alarm icon (shown in the margin).

Press an ALARM pushbutton that's already *ON* to turn the both the pushbutton and the alarm *OFF*. The menu will stop blinking when you turn an alarm *OFF*. You may also use the ***Alarm Clock*** desk accessory to turn off an alarm which was set by the ALARM pushbutton.

Press an ALARM pushbutton that's *OFF* to turn both the pushbutton and the alarm *ON*. The alarm time will be the value in the X-Register. For instance, if the X-Register contains `1.5`, then the alarm will be set to go off at 1:30 AM. Although the ALARM can be set while in any display format, it is less confusing if you switch into TIME display format before entering the ALARM time into the X-Register.

❖ **ALPHA** Alpha ModeToggle (programmable)

Toggles ALPHA mode *ON* or *OFF*. While in ALPHA mode, the Annunciator will display ALPHA. All functions will be unmapped from their keystrokes. In other words, you may type normally with your Macintosh keyboard, without invoking any functions. The keystrokes will instead be placed into the Alpha Register, the contents of which will be displayed in the MainLED.

There are two equivalent function names which may be used in calculator scripts: AON and AOFF.

❖ **BOT** Use Bottom FN (not programmable)

Controls the BOT ***shift*** mode. Shift modes determine which function will be invoked on multi function keys. Shift modes are also referred to as ***prefixes***.

Press the BOT pushbutton to temporarily turn the BOT shift mode *ON*. As soon as another key is pressed, the shift mode turns itself *OFF*.

⌘-click the BOT pushbutton to ***lock*** it in the *ON* position. The BOT shift mode remains locked until you press either the TOP or BOT pushbuttons again. You may also press the CLP (clear prefix) key to unlock shift mode pushbuttons.

❖ **DISK** Save to disk file (programmable)

Pressing a DISK pushbutton which is *OFF*, will turn it *ON* and open a ***buffer***. Any subsequent changes to the X-Register will be saved in the buffer.

Pressing a DISK pushbutton which is *ON*, will turn it *OFF* and close the buffer. A standard ***SaveFile*** dialog box (Figure 4-34) will appear, allowing you to save the contents of the buffer into a *Teachtext* file (sample icon is shown at left). Note that the DISK and PRINT pushbuttons each use their own buffers, so it is possible to have each function independently.

Equivalent function name for this function is DISK.

❖ **COMMAS** Thousands Separators (programmable)

Toggles the display of commas between every third digit to the left of the decimal point. The function name is COMMAS.

❖ **KEYMAP** Show mapping (programmable)

Toggles the display of keystroke characters on keystroke-mappable parts. This works very much like double-clicking the Keyboard tool in the Tools palette during construction (Figure 4-13). The function name is KEYMAP.

❖ **LINE#** Tape Numbering (programmable)

Toggles the display of line numbers on the left of the Paper tape and the DISK and PRINT buffers. In PRGM mode, the Paper tape always displays line numbers, despite the setting of this pushbutton. Function names are: LINEON, LINEOFF.

❖ **PRINT** Print on Printer (programmable)

Same as the DISK pushbutton, except instead of displaying a ***SaveFile*** dialog box after the buffer is closed, the PRINT pushbutton displays print dialog boxes. The number and configuration of these dialog boxes depends upon the print driver you have chosen (with the ***Chooser*** desk accessory). The result will always be to print the text flush right on your printer. The equivalent function name is PRINT.

❖ **PRGM** Program Management (not programmable)

Chapter 8 describes this pushbutton in great detail.

❖ **MARGNL** Tape Marginals (programmable)

When this pushbutton is *ON*, each time you invoke a function which affects the X-Register, the Paper tape, DISK buffer and PRINT buffer will display a short description of the function next to the input or result. Normally, the marginal will be the function name. However, if the Paper tape is not wide enough to display the entire marginal, the letter F will be used as a proxy (on the Paper tape only). Equivalent function names are: MARGON and MARGOFF.

❖ **MUSIC** with keypress (programmable)

When MUSIC is *ON*, each press on a key will play a note on the Macintosh speaker (if you have the speaker volume set loud enough; set the speaker volume using the ***Control Panel*** desk accessory). MUSIC is primarily to provide you with audible feedback that the key you pressed was processed. MUSIC will not affect the results of calculations. Equivalent function names are: MUZON and MUZOFF.

❖ **STOPW** Stopwatch (programmable)

Pressing the STOPW pushbutton *ON* will reset the stopwatch timer to zero and start the timer counting again.

Pressing the STOPW pushbutton *OFF* will display the number of elapsed seconds (since you pressed STOPW pushbutton *ON*) in the X-Register. The result will be displayed in the current display format. The equivalent function name is: SW.

❖ **TOP** Use Top FN (not programmable)

Controls the TOP ***shift*** mode. Shift modes determine which function will be invoked on multi function keys. Shift modes are also referred to as ***prefixes***.

Press the TOP pushbutton to temporarily turn *ON* the TOP shift mode. As soon as another key is pressed, the shift mode turns itself *OFF*.

⌘-click the TOP pushbutton to ***lock*** it in the *ON* position. The TOP shift mode remains locked until you press either the TOP or BOT pushbuttons again. You may also press the CLP (clear prefix) key to unlock shift mode pushbuttons.

SLEDs (Supplemental LEDs)

SLEDs are not ***programmable***— in fact, clicking on them does nothing more than allow you to drag the finished calculator around the screen. During construction, SLEDs are not mappable to keystrokes, and have no equivalent function names.

❖ Display of the **Stack Registers**

The format of the displayed Stack Register mirrors the input/display mode (controlled by clickstop switches). Unlike the MainLED, SLEDs will always display the contents of the Stack Registers, no matter the calculator's current mode.

If you wire a SLED to display a non-existent Stack Register (like register 5 when you only wired 4 Stack Registers), the SLED will display No Register.

❖ Display of the **Alpha Register**

Unlike the Main LED, an Alpha SLED will always display the contents of the Alpha Register. If the SLED has not been stretched wide enough to display all 24 characters of the Alpha Register, only the leftmost portion will be seen.

❖ Display of the **Main LED in Alternate Format**

This is actually a display of the X-Register in an alternate display format. Unlike the Stack Register SLEDs, these SLEDs will be unaffected by the input/display format. This kind of SLED is useful in calculators that convert between several different number bases (like an INTEGER, HEX, BINARY calculator).

❖ Display of a **Memory Register**

Memory Register SLEDs always display their contents using the same display format as the MainLED.

If you wire a SLED to display a non-existent Memory Register (like `MEM_50` when you only wired 10 Memory Registers), the SLED will always display `No Register`.

❖ Display of the **Current Time**

You can think of this display as a continuously updating clock display. This SLED displays the current time despite the input/display format. The time will be displayed in the format of the country for which your Macintosh System has been localized (set up for). During construction, you may select a time zone for this SLED to display. A time zone of zero is considered local time. Other CCS Date/Time functions will not affect this display.

4-13

Selector SLEDs (the ones with the arrows)

Selector SLEDs are not programmable. During constuction, they are not mappable to keystrokes, and have no equivalent function names. Clicking on their arrows will change their displays.

❖ **Memory Register Display**

Displays the contents of the ***Default Memory Register.*** Clicking this display's arrows allows you to scroll through the calculator's Memory Registers, one at a time. Clicking the Left arrow scrolls to the previous register; the Right arrow scrolls to the next register. Each time a new Memory Register is selected, it becomes the *Default Memory Register* (See Chapter 5 for more).

This Selector SLED always displays its contents using the same display format as the MainLED.

❖ **X-Register as a Binary Number**

Displays the contents of the X-Register in `BINARY` display format. Clicking this display's arrows allows you to scroll through all bytes of the X-Register as displayed in `BINARY` format. Since binary bytes take up 9 characters in the MainLED (because we insert spaces between each word), you would not be able to view an entire long word unless your MainLED was stretched to display 36 characters. This SLED is a nice alternative to such a wide display.

The Calendar Display (not programmable)

During constuction, the ***Calendar*** display cannot be mapped to keystrokes, and has no equivalent function name. Clicking on the Calendar's arrows will change the Calendar's display.

The Calendar displays a month at a time. Click the arrows below the calendar to select the month and year you wish to view. The Left arrows scroll to the previous month or year; the Right arrows scroll to the next month or year. Click the MTH•YEAR text between the arrow selectors to display the current month and year.

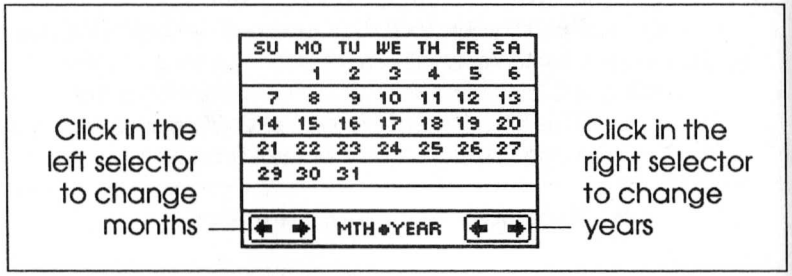


Figure 6-4 The Calendar Display

Note The dates displayed in the Calendar display do not affect the X-Register, or work in conjunction with any of the CCS date/time functions.

Prompt Functions

Any calculator function which prompts you for an answer is called a ***prompt function.*** A complete list of prompt functions appears in Appendix 4. In general, these are the functions which were described earlier in this chapter with phrases like "in the Memory Register which you specify." This page will tell you how to respond to prompts.

If your calculator has a MainLED wired, you'll answer the prompt in the MainLED (Figure 6-5), otherwise the prompt will appear in a dialog box (Figure 6-6). Type your reply to the prompt, then click the ***OK*** button (or type the `return/enter` keys on your Macintosh keyboard).

Figure 6-5 The MainLED prompting for the `STO` function

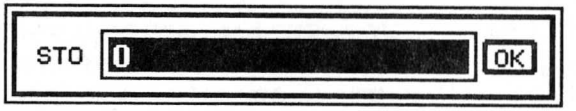


Figure 6-6 Answering the `STO` function in the Prompt dialog

Some prompt functions will supply a default answer; hit the ***OK*** button to accept the answer. For instance, `STO` supplies the Default Memory Register as the answer, while `FIX` supplies the current `FIXED` mode setting.

If you want to cancel a prompt, just backspace out the entire answer and press the ***OK*** button. Cancelling a prompt is equivalent to cancelling the function itself. It is as though you never invoked the function.

Responding to Prompts

There are two ways to respond to a prompt: directly and indirectly. For example, say you want to store a value in Memory Register #1 using the STO function...

- **Direct Reply.** Press STO, and respond by typing a 1. The X-Register would then be copied to Memory Register #1.

- **Indirect Addressing.** Press STO, and respond by typing IND 2. If Memory Register #2 contains the value 1, the X-Register would then be copied to Memory Register #1. This is called ***indirect addressing*** because you do not respond directly with the answer, but indirectly with the *address* of the Register which contains the answer.

In CCS2 calculators, you may use indirect addressing to respond to any prompt function. However indirect addressing is usually done from within a script. See Chapter 8 for many creative uses of indirect addressing in scripts.

Important As with function names, IND is a reserved key word. Take care that your scripts do not include any reserved key words in their titles, or you may not be able to XEQ your script.

Using Prompt Functions in Scripts

When a prompt function is encountered in a script, the script does not stop execution and wait for you to input the answer. Instead, the answer should be on the same script line as the prompt function, with a space between the function name and the answer.

Unwireable Functions

These functions are available only from program scripts— they may not be wired to a part, except as part of a permanent script. See Chapter 8 for more on permanent scripts.

- **END** End Statement

  The last line in a program script (see Chapter 8).

- **LBL** Label Statement

  A script label (see Chapter 8).

- **PROMPT** Prompt user for input from within a script

  Displays the Alpha Register in the MainLED and halts script execution. Using PROMPT in scripts is a good way to get users to input data by displaying a message which instructs the user to enter data. Do so by first storing a message in the Alpha Register, then invoking the PROMPT function. PROMPT should not be confused with ***prompt functions*** (explained on the previous two pages).

- **PSE** Pause script execution

  Stops script execution for 2 seconds, then resumes execution.

- **STOP** Stop script execution

  Stops script execution. Resume by pressing the R/S key.

- **TALKOFF & TALKON** Turn Talk OFF and ON

  Normally, all displays are updated while a script is running. If you set TALKOFF, the displays will not be updated until the script execution is stopped. This is useful in situations where you do not wish to display temporary calculations used within the script— just the end result. Setting TALKOFF will also make your scripts run much faster.

Chapter 7

Example Calculations

This chapter provides examples of common calculations and functions. Seeing how a function is used "in context" often makes it easier to understand what the function actually does. See Chapter 6 for a comprehensive description of each function.

We have provided a calculator work file for each example in this chapter. You will get the most out of this chapter by opening the calculator work files in CCS, going into Test mode, and following along with the examples. The work files are located in the *Chapter 7 Calculators* folder, inside the *Tutorial Calculators* folder, on the *CCS 2.0* disk.

The examples covered in this chapter are:

3 7-8 7-21

- **Date/Time calculations**
  Arithmetic operations using dates and times, finding days between dates, adding days to dates.

2 7-5

- **Yard/Feet/Inch calculations**
  Arithmetic operations using yards, feet and inches. Figuring square footage and square yardage.

4 7-10

- **Percentage calculations**
  Figuring percentages using `%`, `%CH`, and `%T`; markup, discount, and percent of total.

5 7-14

- **Statistical calculations**
  Finding the mean, weighted average, and standard deviation for groups of data pairs.

6 7-18

- **Financial calculations**
  Calculating annuities, investment analysis, amortization schedules, and depreciations.

Time Calculations

These functions provide you with the necessary tools to do almost any type of date/time calculation or conversion. To help you understand how these functions work, we've put together a few sample calculations...

Figure 7-1 (at right) shows an employee's time card for one week of work. In order to pay the employee, we need to calculate how many decimal hours have been worked. This is easy in CCS...

There are two ways to calculate the number of hours. The first is to set the input/display format to HRSMIN, and then use the normal subtraction function. The second way is to key-in the hours in the HMS format and use the HMS- function.

The "AM/PM" column of Figure 7-1 contains the times the employee wrote down.

The "24-HR" column of Figure 7-1 contains the same times in 24-hour format. The AM times match the employee's times, but the PM times are 12 hours greater than the employee's times.

Time Card	AM/PM Time	24-HR Time
Monday		
Punch IN	08:45	08:45
OUT to Lunch	12:15	12:15
IN from Lunch	12:50	12:50
Punch OUT	05:19	17:19
Tuesday		
Punch IN	08:32	08:32
OUT to Lunch	11:57	11:57
IN from Lunch	12:38	12:38
Punch OUT	05:38	17:38
Wednesday		
Punch IN	09:03	09:03
OUT to Lunch	12:11	12:11
IN from Lunch	01:15	13:15
Punch OUT	05:52	17:52
Thursday		
Punch IN	08:18	08:18
OUT to Lunch	11:57	11:57
IN from Lunch	01:15	13:15
Punch OUT	06:48	18:48
Friday		
Punch IN	07:38	07:38
OUT to Lunch	11:22	11:22
IN from Lunch	12:17	12:17
Punch OUT	04:42	16:42

Figure 7-1 Employee's Time Card

Time Card Calculation using HRSMIN Input/Display Format

Calculator Required: 1_TGIF

Problem: We're going to calculate the number of hours and minutes between the time the employee punches in and out. Each time we'll add the number of hours to a cumulative total which we'll keep in Memory Register #0.

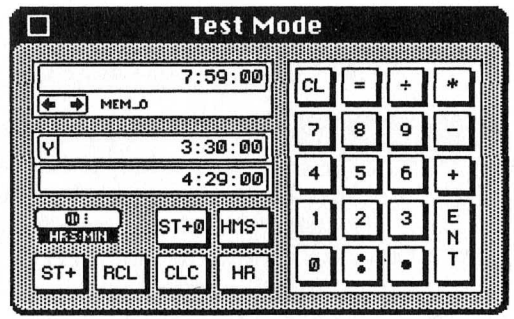
Solution: The calculator will be in HRSMIN format, and we will be working in SAN mode.

Key-in the first punch-OUT time which is 12:15.

> *Note that in HRSMIN format, the colon is used to separate hours:minutes:seconds. If you only enter one colon, it is assumed that you entered hours:minutes only.*

Press the – key. Key-in the first punch-IN time which is 8:45. Press the = key. The answer, 3:30, is displayed in the MainLED.

Now let's store the result. Press the ST+ key, and respond to the prompt dialog box with 0 (for Memory Register #0), then press the **OK** button. Our result is stored in MEM_0, and displayed in HRSMIN format, since that's the calculator's input/display format.



Key-in the next punch-OUT time of 17:19. Press the – key. Key-in the next punch-IN time of 12:50.

Instead of pressing the = key, followed by the ST+ key, press the ST+0 key. The ST+0 key is wired as a script of just two lines:

```
0001  =
0002  ST+ 0
```

Using this script saves us time. We don't have to press the = key, the ST+ key, or respond to ST+'s prompt dialog for each calculation.

After repeating this process for each pair of punch-OUT and punch-IN times (Tuesday through Friday), MEM_0 will display the total number of hours and minutes as 41:30:00.

Press the clickstop switch so the calculator is in FIXED input/display format. MEM_0 will now display our result as decimal hours: 41.5000. Press the RCL key, and respond to the prompt with 0, then press the **OK** button. 41.5000 is displayed in the MainLED.

All that's left is to multiply the number of hours by the rate of pay. We'll assume the employee makes $7.50 an hour. Press the * key, key-in 7.50, then press the = key. Our employee will earn $311.25 for the week.

Time Card Calculation using the HMS- function

Solution: We'll be working in SAN mode. The calculator should be in FIXED format; if it is not, press the clickstop switch until it says FIXED. Press the CLC key to clear the entire calculator (Stack Registers and Memory Registers).

This example works just like previous one did, except that each pair of punch-OUT and punch-IN times are entered like this:

Key-in the punch-OUT time (in HMS format) as: 12.15. That's right— use a decimal point instead of a colon. Press the ENTER key. Key-in the punch-IN time as: 8.45 (again, a decimal point instead of a colon). Press the HMS- key. The answer, 3.3000, is displayed in the MainLED. Press the ST+0 key, and the answer is added to MEM_0.

After calculating the difference between each pair of punch-OUT and punch-IN times, MEM_0 will display 41.3000. Press the RCL key to recall the result to the MainLED. This number is in HMS format. To convert from HMS format to decimal hours, press the HR key. The answer is now displayed as 41.5000 hours. Now multiply the number of hours by the rate of pay, just as we did in the previous example.

Yard/Feet/Inch Calculations

In FLOAT and FIXED display formats, the decimal point can be thought of as a ***separator***. The decimal point separates the integer from the fractional part of a number.

FTINCH format has a separator called the single quote (') which is used to separate feet from inches.

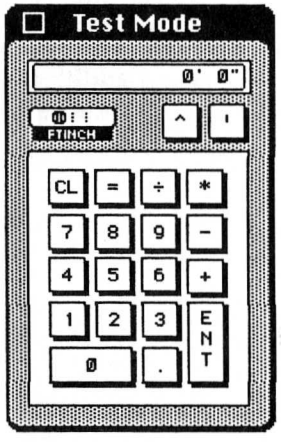
YDFEET format has two separators: the ^ (caret) separates yards from feet, and the ' (single quote) separates feet from inches.

For example, in YDFEET format you would key-in 8 yards, 5 feet, and 6 inches as: 8^5'6. An inches symbol (") is automatically appended to the last digit after you've finished keying-in your number. Yards, feet, and inches must be entered as integers; fractional inches are not allowed in YDFEET or FTINCH formats.

If you do not key-in a single-quote mark in FTINCH format, your input is interpreted as feet.

If you do not key-in either separator in YDFEET format, your input is interpreted as feet. If you enter the single quote, but not the caret, your input is interpreted as feet and inches— but will be redisplayed as yards, feet and inches. For instance, if you enter: 20'6, it will be displayed as: 6^2'6". If you only want to display feet and inches, use the FTINCH format.

Calculator required: *2_Yard Sale.*



Converting Inches to Feet, and to Yards/Feet

Problem: You measure a narrow wall using a tape measure that only has inch marks. The wall is 63 inches wide. You want to know the width of your wall in feet, and in yards/feet.

Solution: The calculator should be in FTINCH format already. Key-in the number of inches, by first pressing the ' key, then the inches: 63. Press the = key. Your wall is 5'3" wide. To find the number of yards/feet, press the clickstop switch, so it is in YDFEET format. The wall is 1^2'3" wide.

Feet/Inch Calculation Example

Problem: You are building a picture frame. You have measured the width (2 feet, 3 inches) and height (3 feet, 11 inches) of the picture, and now want to calculate how much wood will be required to build the frame.

Solution: The calculator should be in FTINCH format. Our calculations will be done in SAN mode.

Key-in the width: 2'3 (be sure to press the ' key between the two numbers). Press the + key. Key-in the height: 3'11. Press the = key. The height plus the width is: 6'2". This is half the material we need. Press the * key. Key-in: 2, and press the = key. Our picture frame requires 12'4" of material to surround the picture.

Note: when building frames, professionals always figure an additional 8" to 10" of material for the mitered corners. To be on the safe side, press the + key, and key-in: '10 (ten inches). Press the = key. We should really buy 13'2" of material.

Calculating Areas (Square Footage and Square Yardage)

Problem: You just added a new rumpus room to your house, and are ready to purchase the flooring. You aren't sure if you want to use tile or carpet, but since you've just spent a fortune on the room, you will go with the least expensive choice. In pricing the flooring, you find that tile is sold by the square foot, while carpet is sold by the square yard. Your rumpus room is 12 feet 6 inches, by 20 feet, 8 inches. The tile is $12 per square foot, and the carpet is $26 per square yard. What will each kind of floor covering cost, and which one will be cheaper?

Solution: Let's figure the square footage first. Make sure the calculator is in FTINCH format. Multiply the room dimensions to figure square footage. Key-in: 12'6, press the * key. Then key-in: 20'8, and press the = key. Your room is just over 258 square feet.

To figure the total tile cost, we multiply square feet by price per square foot. Press the clickstop switch to FLOAT format which displays the square footage as 258.3333. Press the * key, and key-in the price per foot: 12. Press the = key. Tiling the rumpus room will cost $3,100.

Let's figure the price for carpeting. Get into YDFEET format. Figure the square yardage by multiplying 12'6 by 20'8 which equals 86^0'4". Go into FIXED format, which again displays the square footage of 258.3333. Divide this by 3 to get 86.1111 square yards. Multiply the square yardage by the cost per yard: 26. It will cost $2,238.88 to carpet the rumpus room.

Now you've got another problem— what color carpet buy? Sorry, CCS can't help you there.

Date Calculations

In CCS, date calculations are as simple as adding numbers together using the two available date formats: MDY (MM.DDYYYY) and DMY (DD.MMYYYY).

Adding Days to a Date

Calculator Required: 3_The Dating Game.

Problem: Today we're thinking about buying a 90-day certificate of deposit (CD) and we would like to know when this CD will mature. We will be using MDY format, and our calculator is in FIXED input/display format.

Solution: Because date formats require 6 decimal places, we need to press the FIX key, and respond to the prompt with 6, then press the **OK** button. The MainLED will display 0.000000 (zero to six decimal places).

Place today's date into the X-Register by pressing the DATE key. The MainLED displays today's date in MDY format, like this: 7.041989 which represents July 4, 1989.

Press the ENTER key to move the date into the Y-Register. Key-in the number of days to be added to the date: 90. Press the DATE+ key, which adds the number of days (in the X-Register) to the date (in the Y-Register). The stack drops, leaving the answer in the X-Register (the MainLED).

Our 90-day CD would mature on 10.021989 (October 2, 1989) if purchased on 7.041989 (July 4, 1989).

Calculating Days between Dates

Calculator Required: 3_The Dating Game.

Problem: Suppose your employer requested that you prepare a detailed report to be turned in by August 11, 1989. You were given this assignment on June 22, 1989. How many days would you have to prepare? What day of the week is August 11th?

Solution: Again, make sure the calculator is set to display 6 decimal places (see previous example).

Start by keying-in the starting date: 6.221989. Press the ENTER key to move the start date into the Y-Register. Key-in the due date: 8.111989. Press the DDAYS key. The stack will drop, the number of days between the two dates will return in the X-Register, and the MainLED will display 50.000000. You have 50 days to complete your project.

If you had transposed the dates (entering the due date first), the number of days between the dates would be negative.

To calculate the day of the week this of particular due date, key-in the due date again: 8.111989. Press the DOW key. FRI displays in the MainLED. Click on the MainLED (or press the CLP key), and FRI changes to the number 5.000000. This is because Friday is the 5th day of the week (1 = Monday; 7 = Sunday).

Percentage Calculations

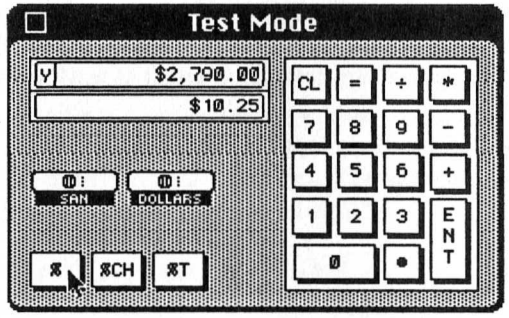
CCS has three different percentage functions: % (Percent), %CH (Percent Change), and %T (Percent of Total). These three functions operate very differently from one another. You must also be aware of whether you are in RPN or SAN when using percentage functions in conjunction with addition and subtraction.

Calculator Required: 4_The Price is Right

Calculating Simple Percent

Problem: A computer salesperson has just sold $2790.00 worth of equipment. Assuming that she makes 10.25% commission on every sale, how much money has she just made?

Solution: Key-in the sale price of 2790. Press the ENTER key; the sale price is now in the Y-Register. Key-in the commission rate: 10.25. Press the % key. The salesperson just made $285.98 on the sale, which is displayed in the MainLED.



Subtracting Simple Percent (discount)

Problem: A $150.00 office chair is on sale for 20% off. How much is the sale price of the chair?

SAN Solution: Make sure the calculator is in SAN mode. Key-in the retail price: 150. Press the – key. Key-in the percent of discount: 20. The Y-Register contains the retail price, the MainLED shows: 20. Press the % key. The MainLED displays the discount: $30.00. Press the = key. The discount is subtracted from the retail price, and the sale price is displayed in the MainLED: $120.00.

RPN Solution: Make sure the calculator is in RPN mode. Key-in the retail price: 150. Press the ENTER key. The Y-Register displays $150.00. Key-in the percent of discount: 20. Press the % key. The discount is displayed in the X-Register: $30.00. Press the – key. The discount is subtracted from the retail price, and the sale price is returned in the X-Register: $120.00.

Adding Simple Percent (markup)

Problem: Calculator Construction Set retails for $89.95. How much will a retail customer pay after the 6.5% California state sales tax has been added?

SAN Solution: Make sure the calculator is in SAN mode. Key-in the retail price: 89.95. Press the + key. Key-in the sales tax percentage: 6.5. The Y-Register now contains the retail price of $89.95, and the MainLED displays 6.5. Press the % key, and the MainLED displays the sales tax: $5.85. Press the = key, and the sales tax is added to the retail price, displaying the sales price of $95.80 in the MainLED.

RPN Solution: Make sure the calculator is in RPN mode. Key-in the retail price: 89.95. Press the ENTER key; the Y-Register now contains the retail price. Key-in the sales tax percentage: 6.5. Press the % key, and the X-Register displays the sales tax: $5.85. Press the + key, and the sales tax is added to the retail price, displaying the sales price of $95.80 in the X-Register.

Percent of Change Discount (points off)

Problem: A software product you are interested in buying has a suggested retail price of $199.95, but a mail order company will sell you this same product for $129.00. How many percentage points *off* is the mail order company offering?

Solution: Press the clickstop switch so the calculator will display in `FIXED` format. Key-in the suggested retail price: `199.95`. Press the `ENTER` key, placing `199.9500` in the Y-Register. Key-in the discounted price: `129`. Press the `%CH` key. The answer is `-35.4839%` off retail.

Percent of Change (percentage increase)

Problem: A computer system was priced at $2495. The sudden drop in the value of the dollar made RAM chips more expensive, and the price of the computer system climbed to $2895. How much did the price increase, expressed as a percentage?

Solution: Make sure the calculator is in `FIXED` input/display format. Key-in the starting price: `2495`. Press the `ENTER` key, placing `2495.0000` in the Y-Register. Key-in the changed price: `2895`. Press the `%CH` key. The price went up `16.0321%`.

Percent of Total Example #1:

Problem: In Calculating your weekly expenditures, you find that you spend $75 a week on groceries and dining. If your weekly take home pay is $450, what percentage of your paycheck is spent on food?

Solution: Make sure the calculator is in `FIXED` input/display format. Key-in your take-home pay: `450`. Press the `ENTER` key to move the take-home pay to the Y-Register. Key-in the amount spent on groceries: `75`. Press the `%T` key. You spend `16.6667%` of your paycheck on food.

Percent of Total Example #2:

Problem: An advertising agency annually spends $134,000 on personal computers. Of that, they spend $98,000 on Apple Macintosh® computers. What percentage of their total computer budget is spent on Macs?

Solution: Make sure the calculator is in `FIXED` input/display format. Key-in the total number annual budget: `134000`. Press the `ENTER` key to move the total budget to the Y-Register. Key-in the amount spent on Macintoshes: `98000`. Press the `%T` key. `73.1343%` of the annual computer budget was allocated for the purchase of Macintoshes.

Statistical Calculations

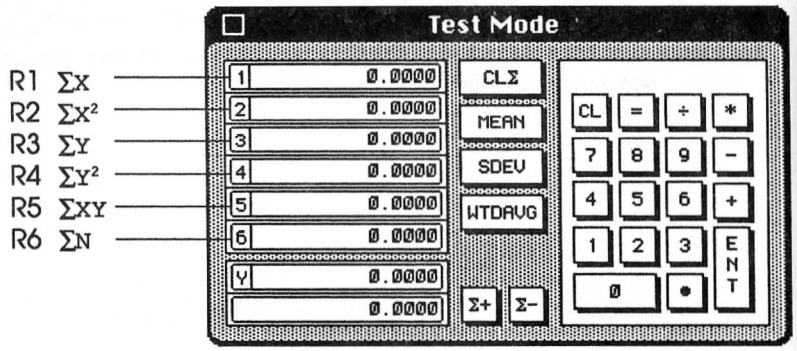
Most CCS functions require you to enter data into the Stack Registers, where the result is returned.

Statistical functions still require you to enter data in the X- and Y-Registers, but statistical functions return their results in a special block of 6 Memory Registers, called the ***Statistical Registers***.

Important Statistical functions store much of their data in the Statistics Registers. Before starting any of the examples, please refer to Figure 5-3, which shows the Statistics Registers.

You can use the function ΣREG? to locate the number of the first Statistical Register. You can use the function ΣMOVE to move all six registers starting with the register you specify. See Chapter 6 for more on these two functions.

Calculator Required: *5_Stats.* Our calculator has the Statistical Registers assigned to their default locations. ΣX (R1) is the first register and is assigned to MEM_11.



Single-Value Statistical Calculations

Single-value calculations are so-called because they require only a single set of accumulated values. Each number in the set is keyed-in to the X-Register one at a time. The Σ+ function is then used to accumulate the X-values into the ΣX Statistical Register.

Problem: You are a hardworking student. You have just finished mid-terms in your six classes. You want to calculate your Grade Point Average (GPA) to impress your friends and solicit more money from your parents. Here are your grades:

class *(number of values entered)*	grade *(X-value)*
1	75
2	98
3	89
4	92
5	94
6	83

Solution: The calculator will be in FIXED input/display format, and we will be working in SAN mode. Start by making sure the Statistics Registers are cleared, by pressing the CLΣ key.

Key-in the first grade: 75. Press the Σ+ key. The X-value is accumulated into the ΣX register (R1), while the ΣN register (R6) and the X-Register both contain the number of X-values accumulated so far.

Key-in each remaining grade, followed by a press on the Σ+ key. Note that in this calculator, the Σ+ key is mapped to the return key on the keyboard to speed up data entry. If you make a mistake and key-in the wrong X-value, simply key the erroneous number into the X-Register and press the Σ− key. This will remove the value from the accumulations.

When all six grades have been entered, press the MEAN key. This will calculate the arithmetic mean (average) of all the grades you entered. The X-Register contains the result: 88.5. That's a solid B+ and good enough for a $200 bonus from Mom and Dad!

Double-Value Statistical Calculations (using data pairs)

Double-value calculations are so-called because they require two sets of accumulated values. You key numbers into both the X- and Y-Registers, then use Σ+ to accumulate X-values into the ΣX Statistical Register, and Y-values into the ΣY Statistical Register. We refer to each X and Y set of values as a ***data pair***.

Problem: Find the hourly productivity of a group of Macintosh programmers during a typical work day. Here is the raw data:

Programmer *(data pair #)*	Daily hours worked *(X-value)*	Lines of code written *(Y-value)*
1	8.0	1025
2	8.5	1145
3	10.0	980
4	7.5	1250
5	8.0	740
6	11.0	1075
7	8.0	925
8	9.0	1560

Solution: The calculator will be in FIXED input/display format, and we will be working in SAN mode. Start by making sure the Statistics Registers are cleared, by pressing the CLΣ key.

Let's key-in the first data pair: lines of code written (Y-value), then hours worked (X-value). Key-in lines of code: 1025. Press the ENTER key to put lines of code into the Y-Register. Key hours worked (8) into the X-Register.

Press the Σ+ key. The data pair is accumulated into the Statistics Registers. The ΣX register (R1) contains 8, the ΣY register (R3) contains 1025, and the ΣN register (R6) contains 1. To confirm the accumulation, the number of data pairs accumulated (the same as in R6) returns in the X-Register. This is useful when there is no SLED wired to display the ΣN register (R6).

Enter all eight data pairs. Press the MEAN key. The X-Register contains the average hours worked per day by a programmer in our group. The Y-Register contains the average lines of code produced per day by a single programmer.

Press the SDEV key. The Y-Register will contain the ***Standard Deviation*** for all the Y-values, and likewise, the X-Register will contain the *Standard Deviation* for all the X-values.

> *In layman's terms, Standard Deviation is a way to gauge how far apart the values are from each other. The lower the number, the closer together the values are. The higher the number, the larger the descrepancy between the values.*

In our example, the Standard Deviation in the X-Register is a relatively low number. So, we can see that our programmers are all working about the same number of hours.

But the Standard Deviation in the Y-Register is a fairly high number, which could mean a number of things. Perhaps a few of the programmers are gung-ho— handily producing more than the others. A few programmers might be taking too many coffee breaks. (Or maybe it's just not valid to measure a programmer's performance in terms of lines of code per day).

Weighted Mean (Weighted Average)

Problem: The weighted mean (WTDAVG) differs from the arithmetic mean (MEAN). You can calculate MEAN using either single-values or data pairs. But WTDAVG only works properly with data pairs. Numbers keyed into the Y-Register are the ***values***. Numbers keyed into the X-Register are the ***weights***. When calculating WTDAVG, each value's importance in the overall group is determined by its corresponding weight.

In the previous example, we calculated the average amount of code written in one day by our group of programmers. But everybody worked for a different length of time. WTDAVG will provide a more accurate representation of their productivity than did MEAN.

Solution: In the previous example, we already entered each data pair. Our accumulated Y-values are the lines of code written. The weighting factor is the daily hours worked. Press the WTDAVG key. The weighted average returns in the X-Register. Our programming group can produce an average of 1089.89 lines of code per day.

Financial Calculations

If you are an experienced financial calculator user: CCS implements all of the financial functions found in the Hewlett-Packard 12C with the exception of `PRICE` (calculate bond price) and `YTM` (yield to maturity). CCS also follows the same sign convention (positive for money received, negative for money paid) as the HP-12C.

The rest of this chapter has been written for those who have had little or no experience with any type of financial calculator.

CCS contains a complete set of functions to help you solve your financial problems. The following two sections describe financial functions: the first covers information you'll eventually need to fully utilize all the functions; the second explains basic principals with examples. In each example, every new function used is explained in detail.

Financial Functions—Section One

Most CCS functions expect you to enter data into the Stack Registers, then return their results in the Stack Registers.

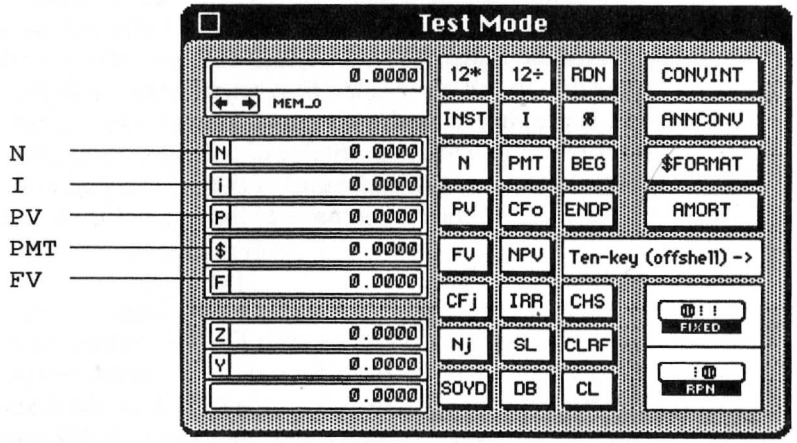
Financial functions (like Statistical functions) still require you to enter data in the X- and Y-Registers, their results are returned to a special block of Memory Registers. The financial functions use a block of 26 Memory Registers called the ***Financial Registers***.

As explained in Chapter 5, there are 26 designated Financial Registers. Registers 17 through 21 are dedicated to the ***Annuity Functions*** (explained below).

17	`N`	Number of compounding periods,
18	`I`	Interest per compounding period,
19	`PV`	Present value, or principal,
20	`PMT`	Monthly payment,
21	`FV`	Future value, or an amount after interest is added.

You can use the function `FREG?` to locate the number of the first Financial Register. The `FMOVE` function us used to move all 26 registers beginning at the specified register location. See Chapter 6 for more on these two functions.

Calculator Required: *6_Down to Business.* Our calculator has Financial Registers assigned to their default locations (the first Financial Register `N` is assigned to `MEM_17`). The first 5 registers (`N`, `I`, `PV`, `PMT`, `FV`) are displayed in SLEDs. Use the Selector LED to view the optional Cash Flow Registers (`F0` through `F20`, wired as `MEM_22` through `MEM_42`). The ten-key pad is located off the Calcshell to make room for the financial functions. Type numbers and arithmetic operators from your keyboard.



Annuity vs Compound Interest

An annuity is a series of equal cash flows of money paid or received. That is, when you make or receive regular payments at regular times, you've got an annuity. If the bank loans you money and you make regular payments, you've got an annuity. If you own rental property and receive monthly payments, then you've got an annuity. Different equations are used depending on whether your problem is an annuity or a compound interest problem. CCS distinguishes between these two cases by examining the contents of the `PMT` (payment) register. If the `PMT` register contains zero, it's a compound interest problem. If the `PMT` register contains a non-zero value, then you have an annuity problem. The five basic financial functions are sometimes refered to as the Annuity functions.

Payment Modes

The payment mode is set according to when each payment is made. When payments are made at the beginning of the period, you must set the payment mode with BEG. Conversely, when the payment is made at the end of each pay period, you must set the payment mode to ENDP.

According to Dr. Elbert Greynolds of Southern Methodist University, payments made at the end of each payment period are called ***ordinary annuities***.[1] Most loans are considered ordinary annuities. Payments made at the beginning of each pay period are called ***annuities due***. Examples of annuities due are payments made in expectation of a service or as part of an investment program. Most leases are considered annuity due situations. When you're not sure of the payment mode, a good question to ask is: was the first payment made at the end or at the beginning of the first payment period? In the examples that follow, we will show examples of ordinary annuities (when the payments are made at the end of the period) and annuities due (when the payments are made at the beginning of each period).

Sign Convention

The sign convention followed by CCS is: money paid out is considered negative, money received is considered positive. In the first example (on the next page), $5000 is considered positive cash flow because you received the money. In the second example, $2000 is negative since it was paid out to your friend. The annuity functions (N, I, PMT, PV, FV) assume that if PMT is positive, PV is negative. The opposite also holds true: if PMT is negative, PV is positive. AMORT (the amortization function) requires that you key-in the principal borrowed (PV) as a positive number. The monthly payments (PMT) must be negative since it's money being paid out.

[1] Dr. Elbert Greynolds, *BA•II Guidebook*, (Dallas: Texas Instruments, Inc., 1986), pp. 3-9.

The financial functions are listed below. Each financial operation is grouped according to the type of analysis it performs.

Basic Annuity Functions

- 12* Multiplies X by 12 and stores product in N register.
- 12/ Divides X by 12 and stores the quotient in I register.
- AMORT Amortization. Returns the amounts applied toward principal and interest on one or multiple payments.
- BEG Beginning payment mode.
- ENDP Ending payment mode.
- FV Future value, or amount after interest has been compounded.
- I Interest per compounding period.
- INST Simple interest on a 360- and 365-day basis.
- N Number of compounding periods.
- PMT Payment.
- PV Present value, or principal.

Grouped Cash Flow

- CFO Initial cash flow.
- CFJ Cash flow j.
- NJ Stores number of times a cash flow is repeated.
- NPV Net Present Value. Uses the values stored in CFO, CFJ, and NJ.
- IRR Internal Rate of Return.

Depreciation

- DB Declining Balance Depreciation.
- SL Straight line method depreciation.
- SOYD Sum-of-the-years-digits depreciation.

Other

- FMOVE Moves the entire block of Financial Registers. You will be prompted for location of the first Financial Register.
- FREG? Locate Financial Registers. Returns the register number of the first Financial Register to the X-Register.

Executing an Annuity function will do one of two things:

- **Store a number in a particular Financial Register.**
  Key the number into the X-Register. Press the financial function representing the value you wish to store.
  For example, to store 13 into the I register, and 4 into the N register. Key-in: 13 and press the I key. Key-in: 4 and press the N key.

  To recall a number from a Financial Register, press the RCL key, and type in: N, I, PV, PMT, or FV.

- **Calculate a value for the Annuity function executed.**
  To solve for a particular annuity function, you must first store three known values in the appropriate Financial Registers. Figure 7-2 shows the required arguments.

N	=		I	PV[2]	PMT[2]	FV[2]
I	=	N		PV[2]	PMT[2]	FV[2]
PV	=	N	I		PMT[1]	FV[1]
PMT	=	N	I	PV[1]		FV[1]
FV	=	N	I	PV[1]	PMT[1]	

Figure 7-2 Financial function known values table
[1] :at least one of these two functions required
[2] :at least two of these three functions required

After storing the appropriate values, simply execute the function that you want to find.

Financial Calculations Example 1

Problem: You've borrowed $5000 at 14% interest compounded monthly. If your monthly payments are $150, how many payments will you have to make, and for how many years?

Solution: Our calculator is in FIXED input/display format. Press the FIX key, and type in 2, to always display two digits to the right of the decimal point. It's always a good idea to clear all of the Financial Registers before starting any calculations, so press the CLRF key (clear Financial Registers).

We want to solve for the number of payments, which is N. We can see in Figure 7-2, that to solve for N, we must always enter values for I, and two of the remaining three: PV, PMT, or PV. Let's key-in our known values...

Interest is 14%. Key-in: 14, and press the 12/ key— which divides the value in the X-Register by twelve, and stores the quotient in the I register. We divided the ***annual*** interest rate by 12 because the value stored in the I register represents the interest per compounding period. In this example, interest is compounded monthly.

Principal, or Present Value is $5000. Key-in: 5000 and press the PV key. Key-in the monthly payment: 150. Since the payment represents negative cash flow, the payment value should be negative. Press the CHS key, and the X-Register will now contain -150. Press the PMT key to store the Payment in the PMT register. Since the payments will most likely be made at the end of each month, set the payment mode by pressing ENDP. Now that the three known values have been stored to the various registers, press the N key to solve for N. The MainLED will display 43.00. You will make 43 monthly payments.

It's easy to view the number of payments as number of years. Press the clickstop switch labeled FIXED, which will advance the clickstop to DOZENAL. In DOZENAL format, the decimal number 43 is displayed in Base 12 as 37. Because the 12's digit is 3, and the 1's digit is 7, we can say that our 43 monthly payments will span a period of three years and seven months. Press the clickstop switch again to return the calculator to FIXED input/display format. Let's do another example...

Financial Calculations Example 2

Problem: You agree to loan a friend $2000 at 11% interest, compounded monthly. Your friend agrees to pay you back the principal ($2000) plus interest in three years. How much will his monthly payments be?

Solution: In this example, we want to solve for PMT (amount of payment), and we know three of the four variables:
N (number of payments; three year's worth),
I (interest rate of 11%), and PV (present value of $2000).

Let's enter a value for N. Key-in: 3 (3 years) and press the 12* key. The number in the X-Register (3 years) is multiplied by 12 (12 months per year) and stores the product (number of months) in the N register.

Now we'll store the interest rate as we did in the previous example. Key-in: 11 and press the 12/ key. Recall that the 12/ function divides the number in the X-Register (annual interest) by 12 and stores the result in the I register (monthly interest).

Key-in: 2000 (the principal you are loaning), and press the CHS key, followed by the PV key. Because we are loaning money out, our ***cash flow is negative***, so we must enter the present value as a negative number.

Set the payment mode to the end of each month by pressing the ENDP key. Now that we've keyed-in the three known values, press the PMT key to solve for payment amount. The number returned to the X-Register is 65.48. This amount is the monthly payment required to pay off this debt. The number returned is positive since it represents ***positive cash flow***— money you receive each month.

Money has a time-related value. There are four things you need to sort out when dealing with interest related problems.

1. The amount of money involved. The initial loan or investment is refered to as the principal.
2. The interest rate. How often is the interest compounded?
3. The length of time that the money is loaned out or borrowed.
4. The type of interest, or how the interest is computed.

With these four criterion in mind, we will now discuss the various ways that interest is computed...

Simple Interest Example 1

Simple Interest is computed by multiplying the principal, with the interest rate, and the length of time. In this case, no financial functions are necessary.

Problem: You've loaned a company $2000 at 8.5% interest for two years. Assuming you're calculating simple interest, how much money will you have at the end of the second year?

Solution: Key-in: 2000 (the principal) and press the ENTER key. Now key-in: 8.5 (the interest rate) and press the % (percent) key. The MainLED will display 170. Key-in: 2 (number of years), and press the * (multiplication) key. The MainLED displays 340, which is the profit made on the principal. Add this number to the principal by pressing the + (addition) key. The MainLED displays 2340.00, the amount of money that you'll have at the end of two years.

Simple Interest Example 2

The INST function calculates the simple interest on both a 360- and 365-day basis, returning two results. To use this function you must store values for N, I, and PV in Financial Registers.

Problem: You decide to loan another company $5000 for 120 days at 12% annual interest. How much money will you have made at the end of the 120 days?

Solution: Key-in: 5000 (the principal), press the CHS key, then the PV key. Recall that money paid out is always keyed-in as a negative number. Key-in: 120 (the length of the loan in days), and press the N key (number of periods). Key-in 12 (annual interest rate) and click I.

Each of the known quantities are now stored in its appropriate Financial Register. Press the INST key. The number returned to the X-Register, 200.00, represents the simple interest based on a 360-day basis. The number returned to the Z-Register, 197.26, is the interest calculated on a 365-day basis. Press the RDN key twice to roll the stack down, displaying the Z-Register in the MainLED.

Note: When using INST to calculate simple interest, it is assumed that the value stored in N (number of periods) is expressed as the number of days. If you want to solve a simple interest problem using years instead of days, follow the same procedure outlined in ***Simple Interest Example 1.***

Compound Interest Example 1

Most problems involving interest deal with interest compounded in one form or another. Banks usually refer to their interest rates as an ***APR*** (Annual Percent Interest Rate), but often compound their interest at different time periods. Compounded interest problems are solved by CCS using the following equation:

$$PV(1+i)^{N} = FV$$

Problem: You have $5000 that you wish to place in an interest bearing savings account for three years. Bank A quotes you an APR of 8.5% compounded monthly, while Bank B says their interest rate is 9.5% compounded quarterly. Which bank will give you the better deal?

You're solving for a future value (FV) since you want to know how much money you'll have after a certain time period (at a future date). Recall from ***Section One***, that you need to key-in at least three of four basic financial values when solving for the fifth. ***Note:*** The five basic financial functions, or annuity functions are: N, I, FV, PMT, PV.

Solution: Let's calculate Bank A's return. Key-in: 5000 (our principal or present value), press the CHS key, and the PV key. This number is negative since you pay the bank this amount— it represents money paid out. Because Bank A compounds the interest on a monthly basis, we can use the 12* function. Key-in: 3 (number of years) and press the 12* key to compute the number of months in 3 years. The 12* function automatically stores the result (36) in the N register. Key-in: 8.5 (the annual interest rate), and press the 12/ key to compute the monthly interest rate, storing the result in the I register.

We now have three known values stored in their respective registers. Press the FV key to compute the future value of your principal. Bank A gives you $6446.51 after three years with 8.5% interest compounded monthly.

Now let's calculate Bank B's return. Press the CLRF key to clear the old financial data. Once again, key-in the deposit: 5000, and then press the CHS and PV keys. Recall that N represents the number of compounding periods, and that I represents the interest per compounding period.

Key-in: 3 (number of years), and press the ENTER key. Key-in: 4 (periods per year), and press the * (multiplication) key. Because Bank B compounds interest quarterly, we will calculate N by multiplying the number of years (3) by the number of periods per year (4). Press the N key to store the result (12) in the N register.

Key-in: 9.5 (the APR), and press the ENTER key. Key-in: 4 (periods per year), and press the / (division) key. To compute the interest per period, we have divided the APR (9.5) by the number of periods per year (4). Press the I key to store the result (2.38) in the I register.

We have now placed the three known values in their respective registers. Press the FV key to compute future value. Bank B gives you $6626.69 after three years with 9.5% interest compounded quarterly.

The results show that even though Bank B compounds interest four times a year, the higher interest rate gives you $180.18 more than Bank A.

Simple Interest Example 3

Interest can also be compounded on a continuous basis. However, CCS financial functions calculate compound interest on a discrete basis. To compare continuous interest rates with discrete rates, we have to convert from continuous to discrete using the following formula:

$$i = e^{r} - 1$$

To convert from one to the other, we'll just key-in the formula. But what if you always work with interest rates that are compounded continuously? Typing in the extra numbers and functions can become a real chore. CCS to the rescue! We can write a simple program script that does this conversion for us. In addition, we can wire this program script to a key in our calculator. After that, the conversion is just a click away.

Problem: You have $5000 that you'd like to deposit for 3 years in a savings account with 8.5% interest. How much more money will you earn if the interest on the account is compounded continuously, as opposed to what you might earn if the interest were compounded monthly?

Solution: First let's calculate the money earned when the interest is compounded monthly. This should be easy for you by now. Key-in and store the known values in their respective registers:

```
5000, CHS, PV, 8.5, 12/, 3, 12*, FV
```

The answer returned is 6446.51.

Now, let's see just how easy this conversion business is with CCS. Type in the continuous rate in decimal form and run through the equation:

```
0.085, ENTER, ETOX, 1, -, 100, *
```

Store this number in the I register by pressing the I key. Now key-in:

5000, CHS, PV, 3, N, FV

The money you earn when interest is compounded continuously for three years is 6452.31. You earn $5.80 more when the interest is compounded continually.

Script Solution: This problem needs to be solved only once. You can then write a program script that will always solve the problem for you. This program script takes the decimal equivalent of the continuous interest rate from the X-Register, converts it to a discrete rate, and then stores the correct number in the I register. Here's our script. We'll call it CONVINT:

```
ETOX   ;Trust us folks-- this works!
1      ;
-      ;This script is based upon the
100    ;formula: i = e^r -1
*      ;
I      ;
```

We have already wired this script to a key in the *6_Down to Business* calculator. We have also provided the script in a file called *CONVINT.Script*, located in the same folder as the calculator.

Don't forget to enter the number in decimal form, i.e. if the continuous interest rate is 12.5%, you would divide by 100 to get 0.125 before you ran the conversion script.

Let's solve the same problem with our program script:

0.085, CONVINT, 5000, CHS, PV, 3, N, FV

Easy, right?

Annuities

As you will recall, annuities are defined as a series of equal payments. We also explained that CCS distinguishes between compound interest problems and annuity problems by the contents of the PMT register. If PMT = 0, then it's a compound interest problem. If PMT ≠ 0, then it's an annuity problem. You'll notice in all of the previous compound interest problems; we did not use PMT at all. Some tutorials for different handheld financial calculators suggest that zero be placed in the PMT register when doing compound interest problems just to be sure.

When dealing with annuity problems, which are regular payments at equal intervals, you also need to be aware of when the payments are made to set the appropriate payment mode. The next two examples illustrate how to solve annuity problems.

Annuity Example 1

Problem: You decide to give a low interest loan to a friend in the amount of $1500 at 5% annual interest, compounded monthly. You'd like the principal and interest due to be paid back to you in two years. How much should your friend's monthly payments be?

Solution: Since this is a loan, it is considered an ordinary annuity with payments made at the end of each period. In this case, the payment period is monthly. Therefore, the payment mode will be ENDP for payments made at the end of each month. We will be solving for PMT, the amount of the monthly payment. Following the same procedure as always, we will store the three known values to their appropriate registers, and then solve for the unknown quantity by pressing the corresponding keys:

```
ENDP, 1500, CHS, PV, 5, 12/, 2, 12*, PMT
```

The answer returned is 65.81. Your friend should you pay this amount every month if he wants to remain your friend. The principal is negative since it represents money paid out. Recall that when PV is positive, PMT will be negative and vice versa. If you had keyed-in the principal as a positive number, the amount for PMT would have been negative. Let's try this and see:

```
ENDP, 1500, PV, 5, 12/, 2, 12*, PMT
```

This time your calculator returned -65.81.

Annuity Example 2

Problem: You decide to tighten your belt and save money on a regular basis. You deposit $500 at the beginning of each month in a bank that lists an APR of 8.5%. Recall that the APR is an annual interest rate compounded monthly. What will your balance be at the end of 18 months?

Solution: The first thing to realize is that these deposits are made at the beginning of each month. The payment mode is therefore set to BEG. We want to solve for the future value (FV) of the account since we want to know how much will we'll have at a future date. By now, you should be able to solve this problem with your eyes closed. We've listed the appropriate key strokes in case you have to peek:

```
BEG, 500, CHS, PMT, 8.5, 12/, 18, N, FV
```

After 18 months you'll have $9630.64. N represents the number of compounding periods. In this example we knew exactly how many periods to calculate for, 18, so we stored that number in the N register.

What difference would it have made if you had deposited your money at the end of each month (the end of each payment period) instead of at the beginning of each month? Let's find out. The only difference in the keystrokes used will be the payment mode:

```
ENDP, 500, CHS, PMT, 8.5, 12/, 18, N, FV
```

You'd only have $9562.90 if you made your deposits at the end of each month instead of at the beginning of the month. That's a $67.74 difference. The choice is yours.

Savings Example 1

Problem: You'd like to make one deposit in a savings account and come back at some future date to collect the goods. The principal is $1500. The bank gives you 8.5% APR (compounded monthly). What you want to know is, how long will it take your money to grow to $5000?

Solution: This problem does not involve any regular payments, so right away you know it's a compound interest problem. The known values are: PV, I, and FV. We are going to solve for N. Don't forget to key-in the initial deposit as a negative number. We'll store zero into the PMT register just to let the calculator know that we're dealing with a compound interest problem. Here we go:

```
0, PMT, 1500, CHS, PV, 8.5, 12/, 5000, FV, N
```

The result, in the X-Register, is 171 months. Divide 171 by 12 for a result of 14.25 years (14 years and 3 months).

"What about the payment mode?" you ask. Our reply is that this is a compound interest problem. You don't need to set the payment mode when you're solving for a compound interest problem.

Savings Example 2

Problem: You draw up a savings plan which involves making regular monthly deposits at the beginning of each month. (Remember how much more money you earned when the money was deposited at the beginning of the month as opposed to the end of the month?) You decide to deposit $250 at the beginning of each month for five years. The bank representative tells you that they provide 8.5% interest compounded continuously. How much money will you have at the end of your five-year plan?

Solution: This time, you need to set the appropriate payment mode due to the regular payment schedule. Each deposit is made at the beginning of the month. Set the payment mode to BEG. The known quanities are: PMT, I, and N. The unknown quantity is FV. We will solve this problem two ways. The first way will be done using the necessary keystrokes. In the second part, a program script which converts the continuous rate to its equivalent discrete rate will be used.

Using keystrokes (the long way): The equation used to convert a continuous interest rate on an annuity to a discrete rate is:

$$i = e^{r/f} - 1$$

The new factor, f, represents the number of payments per year. We will first convert the continuous rate to its equivalent discrete rate, and then solve for the future value. The continuous interest rate must be keyed-in in its equivalent decimal format.

```
BEG, 0.085, ENTER, 12, /, ETOX, 1, 100, CHS, *,
I, 250, CHS, PMT, 5, 12*, FV
```

After five years you will have 18,757.70.

Using a script (the short way): Let's write a script that converts the continuous interest rate to its discrete equivalent, and then stores the converted discrete rate to the I register. We can then wire our program script to a key where it will always be at our disposal. Our script looks for the continuous interest rate in decimal form in the X-Register. The name of the program script will be ANNCONV. Here's the script:

```
12      ;Trust us again, folks!
/       ;
ETOX    ;This script works.
1       ;
-       ;
100     ;
*       ;
I       ;
```

We have already wired this script to a key in the *6_Down to Business* calculator. We have also provided the script in a file called *ANNCONV.Script*, located in the same folder as the calculator. Let's solve the same problem using ANNCONV.

```
BEG, 0.085, ANNCONV, 250, CHS, PMT, 5, 12*, FV
```

We get the same answer, 18,757.70, using fewer key strokes.

Real Estate Example 1

Problem: You're buying a house and shopping for loans. You want a 30-year loan for $175,000 at 13.5% interest, compounded monthly. The bank charges you 3 points (ouch) to cover their expenses. What will the monthly payments be? If you take the points (loan fee) into consideration, what is the actual interest rate the bank is charging you?

Solution: To find the monthly payments, store the known quantities in the various registers. This is a loan, or an ordinary annuity, with payments made at the end of the payment period (each month) so you will need to set the payment mode to END. The loan amount, or principal, will be keyed-in as a positive number since it represents money received. This means that the number returned for the amount of each payment will be negative. Key-in our known quantities:

```
ENDP, 175000, PV, 30, 12*, 13.5, 12/, PMT
```

Your calculator tells you that your monthly payments will be -2,004.47. This number is negative since it represents money paid out.

A point is 1% of the loan amount. When the bank charges you 3 points, you actually pay the bank a loan fee which is 3% of $175,000. Calculate the loan fee like this:

```
175000, ENTER, 3, %
```

By charging you three points, the bank is paid 5,250.00 in addition to the interest on the loan. The bank says that the interest charged on your mortgage is 13.5%. However, they charged you an additional 3% of the principal. How much interest are you really being charged when you include points that the bank charged you?

We want to find the interest I. We need to store at least three known values. To find the interest we will use the answers found in the previous two questions. We will use the monthly payment amount (PMT), and the principal received from the bank, minus the amount that the bank charged as a loan fee ($5250). The third value to store is the duration of the loan.

With these three values stored, we will solve for the true interest charged by pressing the I key. However, I represents the interest per compounding period. To convert to the annual interest rate just multiply the number returned by 12. Don't forget to set the payment mode to ENDP:

```
ENDP, 30, 12*, 2004.47, CHS, PMT, 169750, PV, I,
12, *
```

We multiplied the monthly interest rate by 12 instead of using the 12* function because we don't want to store the product in the N register. The true annual interest rate (adjusted for the points) is 13.949%. The MainLED displays 13.95 because we are in FIXED format, with FIX set to 2 decimal places.

Real Estate Example 2

Problem: You've found a house that you want to buy. After shopping around you've also found a great interest rate. The house costs $210,000. You make a 10% down payment. The bank will loan you the balance (after the down payment) at 12% compounded monthly for 30 years. How much will the monthly payments on the loan be? If taxes and insurance are roughly 30% of your monthly payment, what will your total monthly payments be?

Solution: Let's break this problem into separate parts. First, we'll figure the amount of the loan we need. The house costs $21,000 and you are putting 10% down:

```
210000, ENTER, 10, %, -
```

This means that you'll need a loan of 18,900.00, which is displayed in the X-Register.

To solve for the monthly payments, store what you know: PV (the principal borrowed), I, and the length of the loan. This is a loan, or an ordinary annuity, with payments made at the end of each period. The payment mode must therefore be set to ENDP. Please remember that the principal is positive (money received):

```
ENDP, 189000, PV, 12, 12/, 30, 12*, PMT
```

The monthly payments toward the loan will be 1,944.08. However, this amount excludes taxes and insurance.

You know that taxes and insurance will be 30% of your monthly payments. Calculate your total monthly payments to the bank like this:

```
1944.08, ENTER, 30, %, +
```

Your total monthly payment will be 2,527.30.

Real Estate Example 3

Problem: Jack takes out a $250,000 mortgage to buy a house at an annual interest rate of 12.5%. He can make monthly payments of $1600. He wants to pay off the entire loan in 15 years. What will his final balloon payment be?

Solution: A balloon payment is a payment that's larger than what you would normally pay every month— the payment "balloons" to a larger amount. In this case, the last payment will be larger than the normal payment in order to pay off the entire debt.

Please note that we have four known values in this problem, instead of the required three. If we didn't know how much Jack's monthly payments were, we would be computing the monthly payments on a 15-year loan. However, since we know how much Jack pays every month, we can calculate how much Jack would owe after paying $2500.00 every month for 15 years. Since this is a loan, set the payment mode to ENDP. We will clear the Financial Registers with CLRF prior to solving this problem:

```
CLRF, ENDP, 15, 12*, 12.5, 12/, 250000, PV,
2500, CHS, PMT, FV
```

The final "balloon" payment will be 304,578.84.

$3050
cleared 18½ years
used
spread sheets
8 Dec 2020
10:00 pm

Personal Loan Example

Problem: A friend of yours borrows $2500 from you. You charge him 14% interest compounded monthly. With a monthly payment of $250, how long will it be before he finishes paying you back?

Solution: The known values are PV, I, and PMT. You are solving for N (number of payments). Since this is a loan, the payment mode should be set to ENDP. Here we go:

```
CLRF, ENDP, 2500, CHS, PV, 14, 12/, 250, PMT, N
```

The MainLED displays 11, meaning that after 11 months he will have completed all of his payments.

Born to Shop Example

Problem: You and "the significant other" go shopping for a couch ensemble for your living room. After deciding on the make and model, the salesman at one store tells you that he can sell you the set for $1200. The salesman at another store tells you that he can sell you the exact same living room set on an installment of $55 a month for 3 years at 15% interest. Which is the better deal?

Solution: What you want to know is how much the living room set costs at the second store, or the present value (PV) of the furniture. Simply key-in what you know. The payments are $55 a month, keyed-in as a negative number. The interest rate is 15% compounded monthly. You'll be making payments for three years. Don't forget that loans are ordinary annuities requiring ENDP payment mode. Go for it:

```
CLRF, ENDP, 55, CHS, PMT, 15, 12/, 3, 12*, PV
```

The second couch costs 1,586.60. If you were to take the offer at the second furniture store you'd end up paying $386.60 more for the same merchandise.

Amortization

CCS has a powerful amortization function called AMORT. Before you obtain an amortization schedule, you need to store the:

principal in PV,
interest per compounding period in I, and
payment value in PMT.

Before you press the AMORT key, key-in the number of payments to be amortized in the X-Register.

AMORT returns the following values:

X-Register = the amount applied toward interest,
Y-Register = the amount applied toward the principal,
Z-Register = the number of payments just amortized.

AMORT also updates the following Financial Registers:

PV = the new remaining balance of the loan,
N = the total number of payments amortized.

The PV and N registers are updated after pressing the AMORT key. If you need to come up with a new amortization schedule starting with the very first payment, store the original value of the principal in PV, and reset N to zero. AMORT (the amortization function) requires that you key-in the principal borrowed (PV) as a positive number. The monthly payments (PMT) must be negative since it's money paid out.

If a new amortization schedule is needed and you don't know the value for PMT, calculate PMT as described in the previous examples and solve for PMT.

These points will all be covered in the examples that follow.

Amortize: a mortgage or debt by instalments

Amortization Example 1

Problem: You have taken out a 30-year, $210,000 mortgage at an APR of 12.5%. Prepare a monthly amortization schedule for the first three months of the loan, and an annual amortization schedule for the first three years of the loan.

Solution: Before you can get an amortization schedule, you need to know the values for PV, PMT, and I. In this example, PMT is unknown. Therefore, the first thing we'll solve for is PMT. As you will recall from the previous examples, to find PMT, we need to have at least three values. In this case, we know PV, N, and I. Let's calculate PMT.

The mortgage amount is keyed-in as a positive number since it's money that you receive. We're going to make the monthly payments at the end of each month, so we'll set the payment mode to ENDP.

```
ENDP, 210000, PV, 30, 12*, 12.5, 12/, PMT
```

The monthly payments will be 2,241.24.

We now have a value for the monthly payments. AMORT places the total number of payments amortized in the N register. We'll have to reset the N register to zero since we used it to calculate the monthly payment, like this:

```
0, N
```

We now have values for PMT, PV, and I. The payments are monthly. Therefore, the number of payments to amortize must be entered in monthly units. The payment mode has already been set to ENDP, so we don't need to change it.

Let's amoritize the first month...

Key-in the number of periods: `1`, and press the AMORT key to compute the interest. The MainLED shows `-2,187.50`. This is the amount of the first monthly payment that will go toward paying the interest. The amount of money of the first payment that goes toward paying off the principal is found in the Y-Register: `-53.74`.

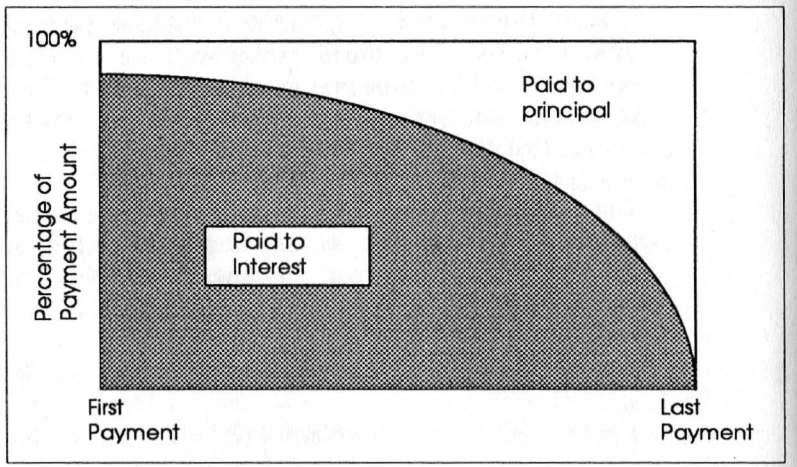


Figure 7-3 Amortization Graph

As our Amortization Graph shows, the bulk of the first payments go toward paying the interest on the loan. Recall that the PV and N registers are updated after every press on the AMORT key. Viewing the contents of the PV register shows that you have `209,946.26` remaining in the balance. Viewing the contents of the N register shows that we have amortized only one payment.

Let's repeat this procedure to find the amortization schedule for the second and third months...

2nd month. Key-in: `1` and press the AMORT key...

Interest:	`2,186.94`	(payment toward interest)
Principal:	`54.30`	(payment toward principal)
PV register:	`209,891.96`	(remaining balance of principal)
N register:	`2.00`	(total payments made)

3rd month. Key-in: `1` and press the AMORT key...

Interest:	`2,186.37`	(payment toward interest)
Principal:	`54.87`	(payment toward principal)
PV register:	`209,837.09`	(remaining balance of principal)
N register:	`3.00`	(total payments made)

The next thing that we need to do is prepare an annual amortization schedule for the first three years of the loan. Recall that the N register contains the total number of payments amortized. Each time AMORT is pressed, the PV register, which contains the amount of the principal, is updated to reflect the amount paid toward the principal. The amount of the monthly payments, contained in the PMT register, remains constant after each call to AMORT.

To find the amortization schedule for the first year, or the first 12 months, we must reset the N register to zero, and store the original loan amount in the PV register. I and PMT do not need to be changed since their contents remain unchanged:

```
0, N, 210000, PV
```

Now let's find the amortization for the first three years...

1st year. Key-in: 12 and press AMORT.

Interest:	26,211.73	(payments toward interest)
Principal:	683.15	(payments toward principal)
PV register:	209,316.85	(remaining balance of principal)
N register:	12.00	(total payments made)

2nd year. Key-in: 12 and press AMORT.

Interest:	26,121.29	(payments toward interest)
Principal:	773.59	(payments toward principal)
PV register:	208,543.26	(remaining balance of principal)
N register:	24.00	(total payments made)

3rd year. Key-in: 12 and press AMORT..

Interest:	26,018.84	(payments toward interest)
Principal:	876.04	(payments toward principal)
PV register:	207,667.22	(remaining balance of principal)
N register:	36.00	(total payments made)

Deferred Mortgage Example

Problem: Calculate the annual amortization schedule for the first four years on a deferred mortgage of $150,000 with an APR of 12.5%. The monthly payments are $1963.42. The mortgage was closed on January 1, 1989. The first payment is due November 1, 1989.

Solution: We have all the information needed to calculate the amortization schedule. The only wrinkle to this problem is the deferred payment schedule. The first thing we will do is store the known values in the appropriate registers with the exception of the monthly payment amount. We will first find out how much the principal grows during the period when no payments are made. We will then key-in the amount of the monthly payments and proceed with the amortization schedule.

Store the known values in the Financial Registers...

The payment mode is set to ENDP because this is a mortgage, or an ordinary annuity.

Payment mode (loan):	ENDP
Principal borrowed:	150000, PV
Interest rate:	12.5, 12/
Deferred payment amount:	0, PMT
reset the N register:	0, N

Type in the number of defered periods (10) and press the AMORT key. There are ten full months between January first and November first.

Interest:	16,378.16
Principal:	16,378.16
PV register:	166,378.16
N register:	10.00

The value returned for interest shows how much interest accrued during the ten-month period that no payments were made.

The number in the Y-Register tells us that the interest accrued was added to original amount borrowed. This is reflected in the contents of the PV register.

The interest accrued was added to the principal ($150,000). As you will recall, AMORT updates the PV register everytime it's called. The N register lets us know that a total number of 10 payments have been amortized.

We will now compute the amortization schedule for the first year. This is done by storing the payment amount in the PMT register, and keying in the remaining months of the first year. Since there are two full months remaining in the year, this is the number that is to be typed into the X-Register.

Monthly payment: `1963.42, CHS, PMT`

1st year. Key-in: 2 and press the AMORT key.

Interest:	3,463.82	(payment toward interest)
Principal:	463.02	(payment toward principal)
PV register:	165,915.14	(remaining balance of principal)
N register:	12.00	(total payments made)

2nd year. Key-in: 12 and press the AMORT key.

Interest:	20,571.99	(payment toward interest)
Principal:	2,989.05	(payment toward principal)
PV register:	162,926.09	(remaining balance of principal)
N register:	24.00	(total payments made)

3rd year. Key-in: 12 and press the AMORT key.

Interest:	20,176.20	(payment toward interest)
Principal:	3,384.84	(payment toward principal)
PV register:	159,541.25	(remaining balance of principal)
N register:	36.00	(total payments made)

4th year. Key-in: 12 and press the AMORT key.

Interest:	19,727.97	(payment toward interest)
Principal:	3,833.07	(payment toward principal)
PV register:	155,708.18	(remaining balance of principal)
N register:	48.00	(total payments made)

Net Present Value and the Internal Rate of Return

NPV (Net Present Value) and IRR (Internal Rate of Return) are two functions used to evaluate the potential profitability of an investment requiring a group of variable payments. By using NPV you can determine how much money you might make, or lose, on an investment requiring a group of payments made on a regular basis. These payments do not have to be equal. IRR is the interest rate earned on the series of payments made (also referred to as cash flows). These definitions are not exact or precise: they are used here only to give you an idea of their use. For an exact definition, see the financial section in Chapter 6.

CCS calculators assume that each cash flow occurs at the end of each period. You must set the payment mode to ENDP whenever you are calculating values for NPV or IRR.

The keys used with NPV and IRR are: CFO, CFJ, and NJ. CFO is used to key-in the initial cash outlay.

CCS Calculators will store up to 20 variable cash flow values, 20 grouped cash flow values, or a combination of variable and grouped cash flows totalling 20. You must indicate whether the cash flow is either positive or negative by using the appropriate sign convention (positive for money received, negative for money paid out).

CFJ and NJ are used together to key-in the different payments made. To enter a cash flow amount that is different from the preceding value, key-in the new value and press CFJ. When the next cash flow value is keyed-in, CCS will automatically enter 1 for NJ.

Let's say that your company pays installments on an investment in the amount of $2000 for 5 periods, and $5000 for 8 periods. To key-in these cash flow values, you would use the following keystrokes:

```
2000, CFJ, 5, NJ
5000, CFJ, 8, NJ
```

The number keyed-in prior to pressing NJ tells the calculator the number of times that the cash amount is repeated. The highest value that NJ accepts is 99. If you have a cash flow amount that is repeated more than 99 times you must break up the amount. For example, if Company X has made 125 payments of $3,000 on an investment, you would key-in the amounts as:

```
3000, CFJ, 99, NJ
3000, CFJ, 26, NJ
```

Once all your cash flow values have been keyed-in to your calculator, you're ready to use NPV and IRR. The next two examples will help to illustrate what NPV and IRR represent and the arguments required to find the net present value and the internal rate of return. Both examples were taken from a book called *Practical Real Estate Financial Analysis: Using the HP-12C Calculator*, written by Dr. Elbert B. Greynolds Jr. and Julius S. Aronofsky.[2]

[2] Julius S. Aronofsky and Elbert B. Greynolds, Jr., *Practical Real Estate Financial Analysis Using the HP-12C Calculator*, (Chicago: Real Estate Education Company, 1983), pp. 196-200.

Take my Mortgage, Please Example

Problem: An opportunity comes to your attention: you can purchase a mortgage with a current balance of $60,721.33. It has an APR of 12%, compounded monthly. There are 180 payments of $661.98 remaining on the mortgage. In addition, the mortgage has a final balloon payment of $30,053.85 due in addition to the final payment. The holder of this mortgage will sell it to you for $43,783.30. However, you are also considering another investment of the same amount of money, with a 15% return. Which is the better deal?

Solution: We want to compare the two investments and see which one will provide the higher return. Here is the information on the mortgage:

Number of payments left (N):	180.00
Amount of payments (PMT):	661.98
Interest (I):	12.00 %
Future value (FV):	30,053.85

The other investment provides a return of 15%.

How much is the mortgage worth today? We will compare the present value of the mortgage to its asking price. The resulting number will show if this is a good investment. We can compare both plans by using the interest rate of the second deal with all other values of the first. Another way of stating this would be: we are looking for a return of at least 15% on the mortgage deal to see how it compares to the second investment. Since all of the payments are the same amount, we can use the annuity keys to solve for the future value of the mortgage:

```
ENDP, 180, N, 15, 12/, 661.98, PMT, 30053.85,
FV, PV
```

The MainLED displays: -50,510.37. This means that the mortgage is worth $50,510.37 today if it had a 15% APR.

We will now compute the NPV of the mortgage investment. Keep in mind that the payment amounts are keyed-in per period. In this example, there will be 179 payments of $661.98. The last payment will include the balloon payment of $30,053.85. This means that the last payment keyed-in will be $30,715.83. In solving for the NPV we will use the target interest rate of 15%. The initial cash outlay will be stored using the CFO function:

```
CLRF, ENDP, 15, 12/, 43783.30, CHS, CFO, 661.98,
CFJ, 99, NJ, 661.98, CFJ, 80, NJ, 30715.83, CFJ,
NPV
```

The value returned is: 6,727.07. Recall that the future value of the mortgage cash flows was calculated to be $50,510.37. This was the equivalent value of the future cash flows using the targeted interest rate. If we subtract the asking price of the mortgage ($43,783.30) from the future value of the mortgage, we get $6,727.07. This number represents the net present value of the series of future cash flows that you'll receive if you buy the mortgage. Your profit is $6,727.07 for this deal.

There may be situations when the number returned for NPV will be zero. The number may even be negative. When the value for NPV is zero, you break even. When the NPV is negative, you'd be better off putting your money somewhere else.

The IRR (internal rate of return) on this investment represents the increase of the money invested. Stated another way: the IRR is the interest rate that makes the future cash flows equal to the initial cash outlay of the mortgage ($43,783.30). To calculate the IRR of the mortgage investment we will key-in the same values using the variable cash flow keys, with the exception of the target interest rate. We won't key-in a value for the interest since it is the unknown:

```
CLRF, ENDP, 43783.30, CHS, CFO, 661.98, CFJ, 99,
NJ, 661.98, CFJ, 80, NJ, 30715.83, CFJ, IRR
```


The number returned, 1.48, represents the monthly interest rate for this series of cash flows. To find the annual return, multiply 1.48 by 12. The annual return on this investment is 17.72%. To check the answer we can use the annuity keys and solve for I. We can use the annuity keys since all the payments are equal:

```
CLRF, 180, N, 43783.30, CHS, PV, 661.98, PMT,
30053.85, FV, I
```

Notice that when we use the annuity keys we don't have to add the future value (the last balloon payment) to the amount of the last payment.

For Sure, Dudes!

Problem: I.L.B. Sure, Inc. is evaluating an investment proposal that involves the following quarterly after-tax cash flows:

Quarter	*Cash Flow*
0	-1,225,400
1	10,000
2	15,000
3	20,000
4	30,000
5	30,000
6	30,000
7	40,000
8	1,540,000

The Chief Financial Officer recommends that the investment provide a 14% return on the investment. Does this investment meet the required return? Will the NPV and IRR satisfy the share holders? Will the Chief Financial Officer lose his job?

Solution: The first step is to input all the data and leave the calculating to CCS. Remember, all NPV and IRR problems require that the payment mode be set to ENDP. The desired return on the investment is divided by four since the cash flows are quarterly, or four times a year. Use the following keystrokes to input the data:

```
CLRF, ENDP, 14, ENTER, 4, /, I, 12254000, CHS,
CFO, 10000, CFJ, 150000, CFJ, 20000, CFJ, 30000,
CFJ, 3, NJ, 40000, CFJ, 1540000, CFJ, NPV
```

The net present value for this series of cash flows is 93,044.27. This represents the value of this investment program. Pressing the IRR key tells you that the rate of return on this investment per period, every three months in this case, is 4.5%. Multiplying this number by four shows a yearly return of 18%.

Please note: When calculating the IRR, CCS calculators assume that the cash flows are reinvested at the same rate.

Calculating IRR may take a while. This is because IRR is solved by a series of iterations. If you do not want to wait for the series of calculations to be completed, hold down the ⌘ and period keys on your Macintosh keyboard.

Depending on the magnitudes and signs of the cash flows, the calculator may find that the value of the IRR may entail one number, multiple numbers, a negative number, or no number at all. You are the final judge as to the "correctness" of the result.

Depreciation

CCS calculates the annual depreciation and the depreciable value of a piece of equipment using three standard methods: Straight-line, Sum-of-the-years-digits, and Declining balance. All three methods require the same information for their calculations:

1. Asset cost stored to the PV register.
2. Salvage cost stored to the FV register.
3. Expected useful life of the asset, in years, stored to the N register.

Before you hit the appropriate depreciation key, you must key-in the year for which the depreciation is being calculated. In addition, when using the declining balance method, you must key-in the declining balance factor (expressed as a percentage) in the I register.

Run-of-the-Mill Example

Problem: A milling machine, purchased for $30,000 has a salvage value of $2,000 and a 15 year life. Find the depreciable value of the mill for the first three years of the machine's life using the straight-line method.

Solution A: Store the required data into the Financial Registers:

```
CLRF, 30000, PV, 2000, FV, 15, N,
```

Now key-in: 1 and press the SL key:

```
1, SL
```

The X-Register displays: 1,866.67. This is the depreciation of the mill for the first year of its life. The Y-Register displays: 26,133.33. This is the remaining depreciable value of the mill. The remaining depreciable value is simply the book value of the asset minus its salvage value. Since the Financial Registers now contain the required information, we key-in the life year of the machine and press the SL key.

2nd Year. Key-in: 2 and press the SL key:

Depreciation:	1,866.66	(X-Register)
Depreciable value:	24,266.67	(Y-Register)

3rd Year. Key-in: 3 and press the SL key:

Depreciation:	1,866.67	(X-Register)
Depreciable value:	22,400.00	(Y-Register)

Solution B: Let's do the same problem using the Sum-of-the-years-digits method. We'll need to clear the Financial Registers and restore the values into their registers:

```
CLRF, 30000, PV, 2000, FV, 15, N, 1, SOYD
```

Depreciation:	3,500.00	(X-Register)
Depreciable value:	24,500.00	(Y-Register)

2nd Year. Key-in: 2 and press the SOYD key:

Depreciation:	3,266.67	(X-Register)
Depreciable value:	21,233.33	(Y-Register)

3rd Year. Key-in: 3 and press the SOYD key:

Depreciation:	3,033.33	(X-Register)
Depreciable value:	18,200.00	(Y-Register)

Solution C: This time we'll use the Declining Balance method to calculate the depreciation of the mill using the same values. Recall that the declining balance method requires that you store the declining balance factor as a percentage to the I register. In this example, the declining balance factor is 1.25 times the straight-line rate. This means that the DB factor is 125%:

```
CLRG, 30000, PV, 2000, FV, 15, N, 125, I, 1, DB
```

Depreciation:	2,500.00	(X-Register)
Depreciable value:	25,500.00	(Y-Register)

All the values stored to the Financial Registers are still intact. We now repeat the same procedure to find the depreciation values for the second and third years of the mill's life.

2nd Year. Key-in: 2 and press the DB key:

Depreciation:	2,291.67	(X-Register)
Depreciable value:	23,208.33	(Y-Register)

3rd Year. Key-in: 3 and press the DB key:

Depreciation:	2,100.69	(X-Register)
Depreciable value:	21,107.64	(Y-Register)

Chapter 8

Building and Using Program Scripts

What is a Program Script?

At its simplest, you can think of a CCS ***program script*** as a recordins of a series of CCS functions strung together. Scripts can be as simple as a constant you use over and over, or as complicated as a loan amortization schedule. Program scripts are a powerful way to extend CCS's set of basic built-in functions. They are also a convenient way to repeat a calculation you might do frequently— such as adding sales tax to a retail price, then displaying a total on the Paper tape.

You create, view, and edit program scripts using the Paper tape and the PRGM pushbutton. Scripts may be printed on your printer, and saved into files on floppy or hard disks. You may also save scripts as a permanent part of your calculator by wiring them to a key. We call these scripts ***Permanent Scripts***.

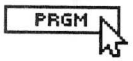
CCS program scripts use a similar syntax (set of key words to represent functions) as the Hewlett-Packard 41C calculator. In fact, many of the programs written for the HP-41C should run unaltered on CCS calculators. (See Appendix 5 for differences between the HP programming and CCS script syntax).

This chapter talks about:

- **Creating a new script**
- **Managing existing scripts**
- **Saving scripts permanently in calculators**
- **Program Script Anatomy**
- **Executing (running) Scripts**

Creating a Program Script

Program Scripts are created in Test mode or by a finished calculator. In either case, the calculator must contain both a PRGM pushbutton and a (preferably tall) Paper tape.



To create a script, click the PRGM pushbutton. The main PGRM dialog box will appear (Figure 8-1).

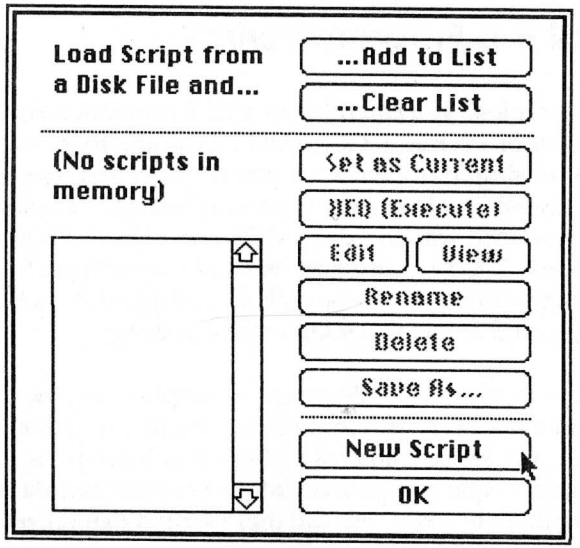


Figure 8-1 Clicking the New Script button in the main PRGM dialog box

All the buttons in this dialog box will be explained in a moment. For now, click the *New Script* button. Another dialog box will appear, prompting you to name the new script you are about to create (Figure 8-2).

Important Function names are reserved key words. Take care that your scripts do not include any reserved key words in their titles, or you may not be able to execute your scripts.

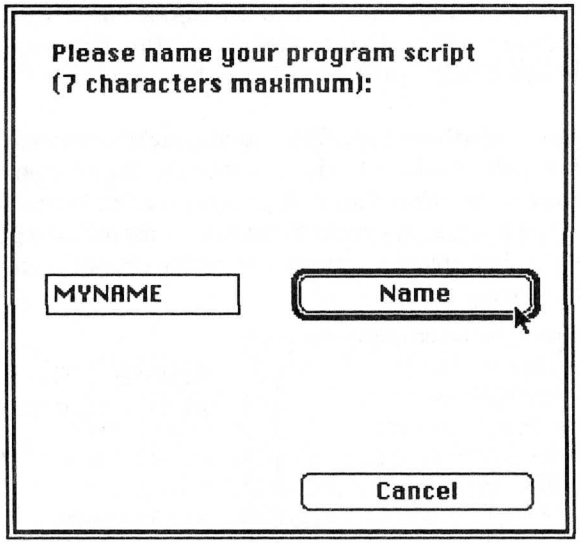


Figure 8-2 **The Script Name dialog box**

Type a one to seven character script name. Click the *Name* button. Both dialog boxes will go away, and the calculator will be in PRGM (program recording) mode. The PRGM pushbutton will be inverted (white letters on a black background). The Annunciator display will say PRGM. The Paper tape will be empty, except for line numbers as in Figure 8-3. ***Don't worry!*** When you exit PRGM mode, the Paper tape will return to normal. You will not lose any data that was previously on the Paper tape.

In PRGM mode, the Paper tape always has line numbers, despite the LINE# pushbutton's setting. In PRGM mode, the Paper tape always left-justifies text, despite the setting of the Justification clickstop switch or default setting.

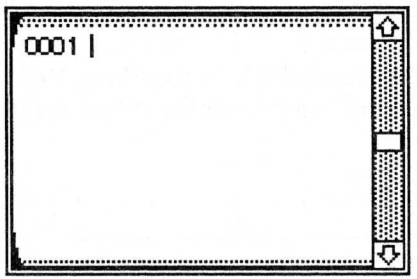


Figure 8-3 The Paper tape ready for you to input a new script

The Macintosh keyboard is now ***unmapped***, and typing a key on the keyboard will no longer execute the function mapped to that key. Instead, text can be typed into the Paper tape.

Click on a calculator key. The name which represents the key's function will be added to the script in the Paper tape. We call this name a ***Function Name***. A program script is really nothing more than a series of function names— entered one per line. Appendix 1 contains a complete list of the function names of all CCS functions.

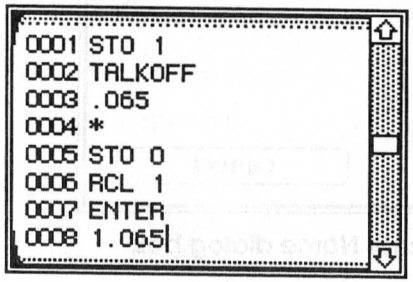


Figure 8-4 The Paper tape with a Script displayed

You can type function names into the script yourself, but clicking on a key automatically adds its function name for you. This is a good way to the learn function names— or a Mac-style shortcut if you do ***not*** want to learn function names!

When you are finished adding function names to your script, click the PRGM pushbutton again. This will turn PRGM mode *OFF*, and return you to normal calculation mode. The PRGM pushbutton will no longer be inverted. The Annunciator will refresh. Your Paper tape will redraw with its previous contents. Your new script will be in memory, unless you did not enter any function names.

You may also turn PRGM mode *OFF* by clicking any any non-key part (like an LED, a graphics part, or the calculator window).

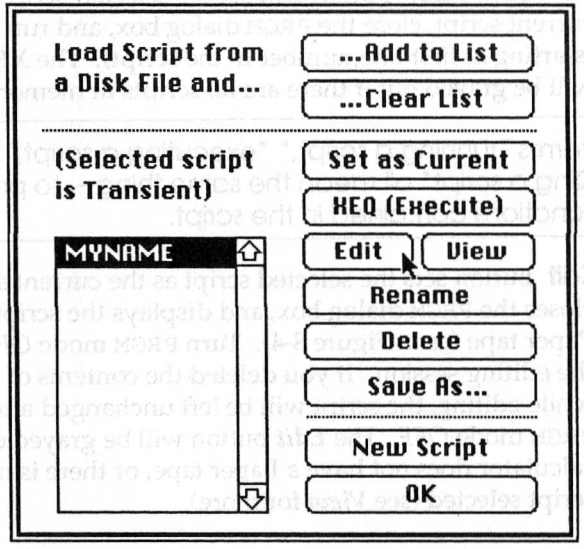
Management of Existing Scripts

Click on the PRGM pushbutton again to bring up the PRGM dialog box (Figure 8-5, below).

The top section of this dialog box allows you to load existing scripts which were previously saved into program script files (a sample file icon is shown in the margin).

- **...Add to List** button will display a standard *GetFile* dialog box (Figure 4-32) with a list of script file names. Select a name, and click the ***Open*** button. The script will be loaded and its name will be added to the list of scripts in memory. If the name of the script being loaded conflicts with the name of a script already in memory, you will be asked to rename the new script before continuing. The name for this function is GETSUB, which can be used in scripts.

- **...Clear List** button does exactly the same thing as the ***...Add to List*** button, except the script being loaded will replace all other transient scripts, clearing the list of scripts in memory. This function's name is GETP, which can be used in scripts.



Clr List

Figure 8-5 PRGM dialog box with a transient script in memory

- The **Scrolling List** box (bottom left of dialog) contains a catalog of all scripts in memory. It will scroll if there are enough scripts in memory to fill the box. Before using many of the other buttons in the PRGM dialog, you must select a script from this list. Click on a script name to select it (selected script names will be inverted).

  The text above the scrolling list will tell you what kind of script is selected:

  Transient Scripts have not been wired to calculator keys.

  Permanent Scripts have been wired to calculator keys.

  Locked Scripts are permanent scripts which have been locked during wiring (Figure 8-7)

- **Set as Current** button will designate the selected script as the ***current script***, and close the PRGM dialog box. Back in the calculator, pressing the R/S (run/stop) key will begin running the current script, starting at the ***current line number*** (explained later in this chapter). This button will be grayed out if there are no scripts in memory.

- **XEQ (Execute)** button will set the selected script as the current script, close the PRGM dialog box, and run the script (starting at first line number in the script). The *XEQ* button will be grayed out if there are no scripts in memory.

Note The terms "running a script," "executing a script," and "XEQing a script" all mean the same thing— to process the functions contained in the script.

- **Edit** button sets the selected script as the current script, closes the PRGM dialog box, and displays the script in the Paper tape (as in Figure 8-4). Turn PRGM mode *OFF* to finish the editing session. If you deleted the contents of the script while editing, the script will be left unchanged after turning PRGM mode *OFF*. The ***Edit*** button will be grayed out if the calculator does not have a Paper tape, or there is no transient script selected (see ***View*** for more).

- **View** works like the ***Edit*** button, except a script cannot be altered while viewing it in the Paper tape. Permanent scripts can be viewed, but not edited (except in Test mode). Locked scripts cannot be viewed or edited (except in Test mode).

- **Rename** button will display the dialog box shown in Figure 8-2. Click the ***Cancel*** button if you do not wish to rename the script. Otherwise, type in the new name and click the ***Name*** button. The ***Rename*** button will be grayed out if there are no scripts in memory.

- **Delete** button will completely remove the selected script from memory. Before doing so, an alert box will appear to confirm your choice (Figure 8-6). If you saved your script to a file before clicking the ***Delete*** button, the copy of the script contained in the file will not be affected by this function. The ***Delete*** button will be grayed out if there are no scripts in memory, or if the selected script is a permanent or locked script (see the next section for more on permanent scripts).

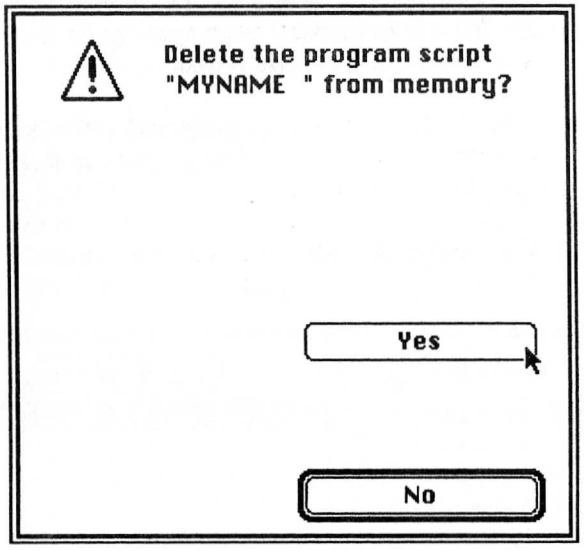


Figure 8-6 Confirm Script Deletion alert box

- **Save As...** button saves the selected script into a script file. A *SaveFile* dialog box appears (Figure 4-34). You may name the file anything you want— it will not rename the script contained in the file. The ***Save As...*** button will be grayed out if there are no scripts in memory, or if the selected script is locked (except in Test mode).

Saving Scripts in Text Files.

Scripts are stored in a compressed format when saved as CCS script files— which may only be read by CCS and CCS calculators. If you prefer to save your script in a "human-readible" text file instead of the CCS script file, you may...

Start by editing the script, which will display it on the Paper tape. Select all the text on the tape. Pull down ***Copy*** from the ***Edit*** menu. Turn PRGM mode *OFF*. Click in the Paper tape (turning TEXT mode *ON*). Pull down ***Paste*** from the ***Edit*** menu, which will paste the text into the tape. Press the SAVEAS key, and save the Paper tape's contents into a *Teachtext* file. *Teachtext* is an application that reads text files, and comes on the *CCS 2.0* disk.

- **New Script** button was explained at the beginning of this chapter. This button will be grayed out if there is no Paper tape in the calculator.

- **OK** button closes the PRGM dialog box.

Saving Scripts Permanently in Calculators

As seen in the previous section of this chapter, scripts may be saved to and loaded from disk files by the calculator. But during construction, you may also build a script into the calculator permanently. We call these scripts ***permanent scripts***.

Permanent scripts are ***wired*** to calculator keys, and will be saved with the calculator in work files, desk accessory files, and calculator application files. In finished calculators, pressing a key wired to script will execute the script.

To wire a permanent script, start in CCS's Test mode. Using the PRGM dialog, load a script which you wish to save as a permanent part of the calculator. (The previous section of this chapter tells you how to load and create scripts).

Exit CCS Test mode. Select the ***Plug*** tool from the Tool palette. Click the plug cursor on the key to which you want to wire the script. The dialog shown in Figure 8-7 will appear.

Select the ***Program Scripts...*** list if it is not yet selected. To change lists, click on ***the pop-up menu*** at the top left of the dialog box. If you are using an older System file, this dialog box will not have a pop-up menu— instead you will find a group of radio buttons at the bottom of the dialog box. Click on one of these radio buttons to change lists.

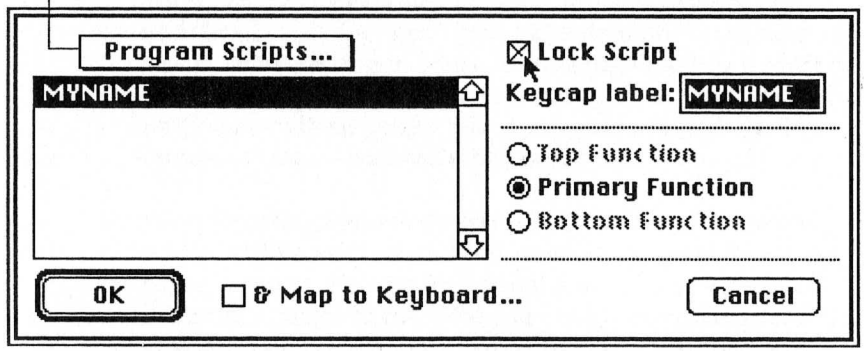


Figure 8-7 Permanently Wiring a Script to a Key and Locking the Script

Click on the name of the script you wish to permanently wire. If you change the keycap label, the script name will not be changed.

Click the **Lock Script** checkbox to lock or unlock the selected script. ***Locked scripts*** are permanent scripts which cannot be edited, viewed, deleted, or saved to disk files. This is a way to protect your source code from prying eyes. To unlock a script, click on the script's key with the plug tool, and uncheck the *Lock Script* checkbox.

This *Wiring* dialog box will not display the names of any permanent scripts which have been wired to other keys— only the names of transient scripts (if any) and the script previously wired to this key (if there was one).

Permanent scripts may not be deleted using the PRGM dialog. To convert a permanent script back to a transient script, you must unwire it during construction, or delete the key to which the script is wired. You may then delete the transient script using the PRGM dialog box.

Program Script Anatomy

Where scripts are stored

Unlike the HP-41C, CCS calculators do not store scripts in Memory Registers. Instead, CCS program scripts are stored dynamically in the Macintosh's RAM. Each script is limited to 32K— a very large script! Should your script exceed 32K, you may break it into subroutines (small scripts), as one script can GOTO (branch to) or XEQ (call) a subroutine in another script.

Elements of Program Scripts

All CCS program scripts consist of three elements:

- **A Global Label or Script Name**
  Global Labels are entered into scripts as "LBL NAME"; LBL is the script function name for label, and NAME is an up-to-seven character label name you type using capital letters. There should be a space after the LBL instruction. A *Script Name* is the name you provide the PRGM dialog box (Figure 8-2) when you create your script. If you do not implicitly type an LBL instruction at the start of your script, the script name will be inserted automatically when you finish editing the script.

- **The body of the script**
  This consists of *Function Names* — entered one per line, each in all capital letters. Type more than one numeral on a line to enter a number. Hexadecimal digits A thru F must be entered as capital letters. Lines starting with a semicolon (;) will be considered comments. Lines starting with double quotes (") are considered alpha input.

  When entering *Prompt Function Names* (function names which require additional data), enter the prompt function name, a space, then the additional data. For example: to store the number in the X-Register to Memory Register #123, you would type STO 123. Appendices 1 and 2 contains a list of all script function names. Appendix 3 contains a list of all prompt functions and their expected prompts.

- **A Script END statement**
  This is a special script function name which acts as an end-of-script marker. If you do not implicitly type an END instruction at the end of your script, one will be inserted automatically when you finish editing the script.

The LBL and END statements are important because they provide an identifier and boundary for the script. In general, a CCS script is executed from a label to its END statement.

To illustrate these script elements, consider a simple program to calculate the area of a circle using the formula:
$A = \pi r^2$. We call this the AREA script (Figure 8-8).

```
01  LBL AREA      ◄ Script label
02  X2            ◄ Start of script body
03  π
04  *
05  END           ◄ Script END statement
```

Figure 8-8 The AREA Script

The above script will square the number in the X-Register and multiply it by the constant π. If the original value contained in the X-Register is a radius, then this program will calculate the area of a circle.

The Current Script Pointer

An internal ***Current Script Pointer*** is maintained within the calculator which always points to the next script line to be executed. Normally, when you are not running a script, this script pointer will point to line number 0001 of the ***current script***. The current script is the last script to be executed or the one you ***Set as Current*** using the PRGM dialog box (Figure 8-5).

Some functions change the script pointer. For example, if a PSE instruction is used to halt script execution, the script pointer will point to the next script line following the PSE instruction. The SST (single step) function will execute the line currently pointed to, and then advance the script pointer to the next line in the script. In general, each time a script function is executed, the script pointer advances to the next line.

Executing (running) Scripts

To ***execute*** (or run) a script, the calculator must be in normal calculation mode. The basic steps for running a program are:

① Make sure the script is in memory.
② Enter or store any data that the script needs before it starts.
③ Start the script, which may be done any of these ways:
- Press the XEQ key and type in the name of your script.
- Press the PRGM pushbutton, select your script and click the XEQ button.
- Press a Permanent Program Script key (which you wired during construction).
- Press the R/S key, which will execute the script starting at the current line.

To resume execution of the current script at the current line, press the R/S (run/stop) key.

Stopping Script Execution

To stop a script's execution, hold down the ⌘ and period keys on your keyboard. A script will also stop executing when it encounters a STOP or PROMPT or END instruction (written into the script).

Scripting in Detail

Indirect Addressing

Indirect Addressing can be a powerful tool when used within a script. To recap from Chapter 6, there are two ways to respond to any prompt function: directly and indirectly. For example, say you want to store a value to Memory Register #1 using the STO function...

- **Direct Reply.** Press STO, and respond by typing a 1. The X-Register would then be copied to Memory Register #1.

- **Indirect Addressing.** Press STO, and respond by typing IND 2. If Memory Register #2 contains the value 1, the X-Register would then be copied to Memory Register #1. This is called *indirect addressing* because you do not respond directly with the answer, but indirectly with the *address* of the register which contains the answer.

To use indirect addressing in a script, simply follow any prompt function name with a space, the keyword IND, and a register number. The example above would appear in a script as the line:

```
STO IND 2    ;Store X-Register in the Memory
             ;Register whose number is stored
             ;in Memory Register 2
```

Here's another example: To check a flag stored in Memory Register #01 using register #15 as the indirect register, you would insert the following line into your script:

```
FS? IND 15   ;Test the flag whose number is
             ;stored in Memory Register 15
```

On the next few pages you will discover how this simple technique can be used to control program flow by creating what programmers call a *loop* . We will explain loops in a minute.

Script Labels

Script Labels can be from one to seven characters long. In addition, script labels may not conflict with CCS key words which we call *function names*. A function name represents a calculator function. For instance, in the AREA script (Figure 8-8), XTO2 is the function name for the square of X.

Local and Global labels (and XEQ, GTO, LBL functions)

Unlike the HP-41C, CCS scripts do not use local and global labels per sé. Instead, any label can be either a global or local label, depending upon where you type a colon after a GTO or XEQ instruction in your script. Note that the GTO and XEQ instructions must be immediately followed by at least one space.

There are the four main forms of the GTO (or XEQ) instructions:

`GTO Label`	Look for this label in the current script.
`GTO :Label`	Look for this label in the current script. If not found, look in other scripts.
`GTO Script:Label`	Look for this label in the script named "Script."
`GTO Script:`	Look for the script named "Script." Treat line 0001 as the label.

These instructions are treated as errors:

`GTO :`	Generates a script error
`GTO`	Generates a script error

Branching and Subroutines

Branching is a term used to describe an instruction which redirects the flow of a script. The script is said to ***jump*** to a line other than the next line in the script (which is normally executed). When a GOTO (label) or XEQ (label) instruction is executed, the search begins for a matching label. If one is found, control is transferred to the line containing that label. There are two basic kinds of branches: unconditional and conditional.

Unconditonal branches always divert the script to another label or line number. CCS scripts have two unconditional branch instructions: GOTO and XEQ.

Conditional branches divert depending on whether a tested condition is TRUE or FALSE. The script line immediately following the conditional test will be skipped if the result of the test is FALSE. The next line in the script will be executed if the result is TRUE. Conditional branch instructions include compares of the X-Register to zero, the X-Register to the Y-Register, and the X-Register to a specified Memory Register, and the Alpha Register to a specified Memory Register. See Chapter 6 for more on each conditional test.

Executing a Subroutine is accomplished with the XEQ instruction. XEQ and GOTO are similar in that they both transfer script control to a specified label. But XEQ is different. When the next RTN or END instruction is encountered, script control returns to the line number immediately following the XEQ instruction.

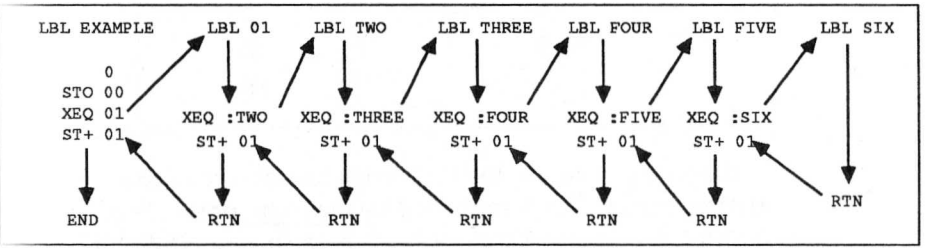
Loops are created by branching to a script label. Consider the loop in this next script fragment:

```
LBL LOOPER   ;Loop back here
-1           ;Subtract 1 from X-Register
=            ;In case we're in SAN
X=0?         ;Does X=0?
GTO LOOPER   ;If TRUE, loop some more
END          ;If FALSE, terminate the loop
```


The Subroutine Return Stack

Do not confuse the *subroutine stack* with the *Stack Registers* explained in Chapter 5. If you never write program scripts, you can ignore this section.

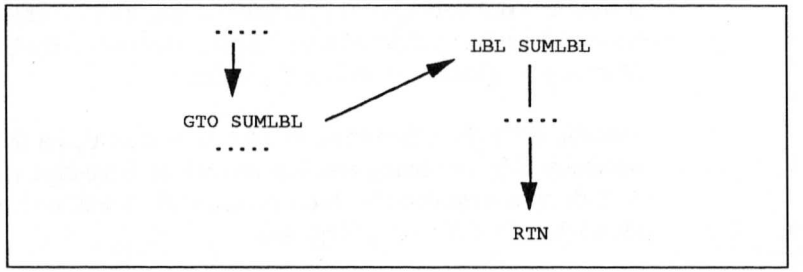
When an XEQ instruction is encountered in a script, the calculator remembers the location of the XEQ instruction. We call this location the ***return address***. While the subroutine is being executed, the return address is stored in an area called the ***subroutine stack***. The subroutine stack is large enough to store up to six return addresses.



New return addresses are added to the bottom of the subroutine stack. When a script encounters either an RTN or END instruction, the last return address pushed onto the subroutine stack is used to return to the XEQ instruction which called the subroutine. The last return address is then popped off the subroutine stack. If there are no entries in the subroutine stack, the script stops running. Here's how all this works in practice...

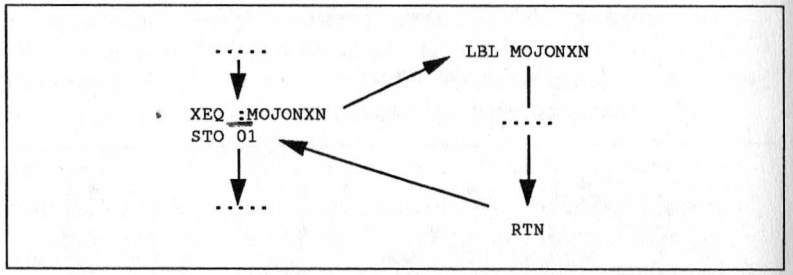
Directing Program Script Flow

GTO and XEQ both alter the flow of script execution, although in different ways. GTO will permanently branch to a different subroutine, as shown in the following diagram.



GTO SUMBLB will divert script exection to the routine listed under LBL SUMLBL. Note that the label is local to the same script as the calling function. Script execution will stop at the first RTN or END in the subroutine SUMLBL.

XEQ also diverts script flow to the named subroutine. However, once the subroutine has finished executing, script execution will return to the line following XEQ.



In the example above, the first script branched to a subroutine in another script. This is indicated by the colon preceeding the label MOJONXN. After MOJONXN finishes executing, control will be transferred back to the line following the XEQ instruction.

GTO and XEQ can also take indirect arguments within scripts. Both functions will accept IND followed by a valid Memory Register number, as in:

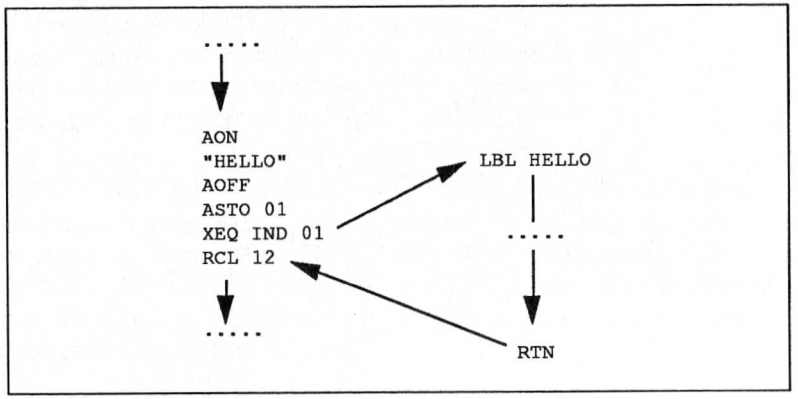
```
XEQ IND 44   ;Execute the label stored in
             ;Memory Register 44

GTO IND 1    ;Branch to the label stored in
             ;Memory Register 1
```

The Memory Register being used as the index may contain either alpha or numeric data. Note that the fractional portion of a number is not considered part of the label name. This is important when using DSE and ISG instructions that affect the Memory Register containing the index.

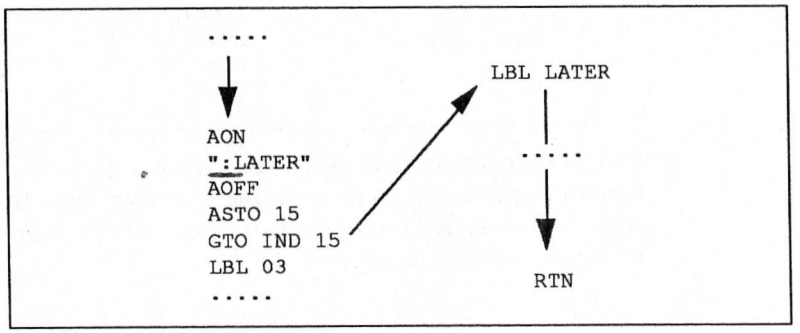
Use the ASTO function to store LBL names as alpha data in the Memory Register being used as an index. Take care that your LBL does not exceed six characters, as ASTO will only store six characters in a Memory Register.

When using indirect addressing, it is much more convenient to use numbers as LBLs, instead of letters. It takes only one script line to STO a number to a register, in comparison with several lines to turn AON, type a label into the Alpha Register, turn AOFF, then finally ASTO the label into a Memory Register.

The diagram below shows a script segment illustrating how XEQ can be used with indirect addressing.



Note that since the label HELLO is not preceeded by a colon, only the current script would be searched for this label. A colon counts as another character when ASTOred in the Memory Register. Therefore, labels must not exceed five characters when being called from another script. Here is an example demonstrating a branch to a subroutine in another script using indirect addressing with GTO:



This particular label takes up the maximum six characters in the Alpha Register. If LATER is not found in the current script, then all scripts in the calculator will be searched until LATER is found. If LATER is not found, script execution will halt, and Non Existent will be displayed in the MainLED.

There are two ways to branch to a numeric label using indirection with GTO and XEQ. The first method allows for branching to labels within the same script. To branch to a numeric label in the same script, the label number is placed in an indirect register and the call is made to execute that subroutine. For example, in the following script, the label number is stored into Memory Register #0. This register is then passed indirectly to XEQ, which executes LBL 01.

```
.....
1                          LBL 01
STO 00                     AON
XEQ IND 00                 "Dubl-Click Software"
CLA                        AOFF
.....                      AVIEW
                           PSE
                           RTN
```

The only way to branch to a numeric label in a separate script is to store the numeric label as an alpha string (with the colon appended to the first digit). For example, in this script:

```
.....
AON                        LBL 05
:05                        AON
AOFF                       "That's all folks!"
ASTO 01                    AOFF
GTO IND 01                 AVIEW
.....                      PSE
                           RTN
```

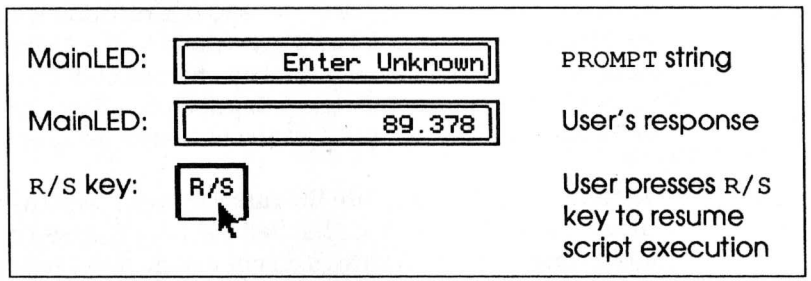
05 has been saved as an alpha label. This is the only way to branch to a numeric label in a separate script. Note that since LBL 05 was called by GTO, script execution will halt after LBL 05 has finished running.

Using the PROMPT function

It is often very helpful to use the PROMPT function to query the user when input is required for a given script. For example, in the script segment shown, Enter Unknown will appear in the MainLED and script execution will stop.

```
AON                    ;Turn alpha mode on
"Enter Unknown"        ;the prompt string
AOFF                   ;turn alpha mode off
PROMPT                 ;stop script & show string
STO 15                 ;save user's answer
```

PROMPT is a combination of the functions AVIEW (which displays the contents of the Alpha Register in the MainLED) and STOP. When script execution stops because of PROMPT, simply enter a number and then press the R/S key. R/S will resume script execution at the line following PROMPT. When the script shown above is run, the illustration below shows how to interact with the prompt using R/S.



You may decide to prompt for character input instead of numeric input. The following script shows how this is done.

```
AON             ;Turn ALPHA mode ON & clear reg
"Y"             ;Put "Y" in the Alpha Register
ASTO 1          ;Store "Y" in Memory Register 1
CLA             ;Clear the Alpha Register
"y"             ;Put "y" in the Alpha Register
ASTO 2          ;Store "y" in Memory Register 2
CLA             ;Clear the Alpha Register
"Continue Y/N?"    ;The prompt string
PROMPT          ;Note: ALPHA mode still ON
AOFF            ;Turn off after the prompt
A=R? 01         ;Did user type "Y" ?
GTO SEEYA       ;Yes, branch !
A=R? 02         ;Did user type "y" ?
GTO SEEYA       ;Yes, branch !
GTO LATER       ;Nope, exit somewhere else
```

Both upper and lower case y are stored away in Memory Registers. ALPHA mode is turned *ON* and the prompt string is drawn in the Alpha Register. The PROMPT function is placed in the script after the string has been written to the Alpha Register— ALPHA mode has not been turned *OFF* prior to PROMPT. This allows the user to input alpha characters instead of numbers, then press the R/S key to resume script execution.

To test the user's input, use the function A=R? which compares the Alpha Register with a specified Memory Register containing alpha data. If the two strings do not match, A=R? returns FALSE and the next script line is skipped. The specified Memory Register number must be on the same line as A=R? with a space between the two.

Loop Control Functions (DSE & ISG)

DSE (Decrement and Skip if Equal or less than) and ISG (Increment and Skip if Greater than) are two powerful looping functions that can provide you with enormous control over script execution.

As discussed in Chapter 6, both DSE and ISG require a control number stored in a Memory Register. This Memory Register number is specified on the same line as the DSE or ISG instructions, separated by a space. Used with GTO or XEQ, DSE or ISG allow loops and jumps to other subroutines depending on whether the looping condition is TRUE or FALSE. (Conditional branching will be covered in a minute). This example of which makes use of the ISG instruction:

```
LBL CUBEME   ;Optional
3.02703      ;Control number for ISG
STO 00       ;Store in Memory Register
;------      ;End of initialization
LBL 01       ;Local label for looping
RCL 00       ;Fetch control number
INT          ;Toss out fractional part
3            ;Input "3"; stack will lift
YTOX         ;Cube Y (X = power of 3)
PSE          ;Display results for 2 seconds
ISG 00       ;Increment control number
GTO 01       ;Loop if <=27
END          ;Optional
```

The control number 3.02703 instructs ISG to start counting with 3, to finish counting when the counter is greater than 27, and to increment by 3. ISG and DSE will cause the next line in the script to be skipped when the test condition proves TRUE. You'll notice that the line following ISG is a GTO instruction which causes script execution to return to LBL 01. Once the counter in the control number is greater than 27, the line following ISG 00 will be skipped, which will end this script.

It may be helpful at times to take advantage of the defaults assumed in the control number. You will recall that if no value is specified for the final counter value (fff), zero is assumed. If no increment value (cc) is specified, one is assumed.

Both of these are nice defaults for the DSE instruction. Our next example demonstrates this technique.

Ex (2) This example calculates the yearly profit on an initial investment. The yearly percent increase is added to the initial investment. This script also illustrates how to prompt for and store numeric input. Recall that after entering the requested input at the prompt, the user must press R/S to resume script execution.

```
LBL INTEREST          ;Optional script name
RPN                   ;Use Reverse Polish Notation
AON
"Initial Investment?" ;1st prompt string
AOFF
PROMPT                ;Wait for user input
STO 00                ;Store once for compounding
STO 01                ;Store again for initial value
AON
"Yearly return (%)?"  ;2nd prompt string
AOFF
PROMPT                ;Wait for user input
STO 02                ;Store user input
AON
"Number of years?"
AOFF
PROMPT                ;Wait for user input
STO 03                ;Store user input
;-----
LBL 01                ;The Loop Begins
RCL 01                ;Initial amount invested
RCL 02                ;Percent growth of investment
%                     ;Calculate it
ST+ 01                ;Yearly profit + initial inv.
DSE 03                ;Loop
GTO 01                ;Skip if control number = 0
FIXED                 ;Change display mode
FIX 2                 ;Two decimal places
RCL 01                ;Initial amount + profits
RCL 00                ;Initial amount only
-                     ;Calculate PURE profits
AON
"Profit = $"          ;Prefix results with string
AOFF
XASA                  ;Append results to string
AVIEW                 ;Display results
END                   ;Optional
```


Note that the number entered for Number of years? is stored as the control number for DSE. Since this number represents the number of years to compound, this value is used as our control number. CCS will correctly assume that we wish to stop at zero, and count down by one. XASA appends the number in the X-register to the contents of the Alpha Register.

(3) These two looping functions (DSE and ISG) can be further extended using indirect addressing, as in this example:

```
LBL DOIT2IT ;Optional
1.005       ;Set up control number
STO 00      ;Store it in register 0
;-----
LBL 01      ;LOOP starts here
AON
"X"         ;X marks the spot
AOFF
RCL 00      ;Fetch control number
INT         ;  as an integer
XASA        ;Append control integer to "X"
AON
APPEND      ;Don't overwrite Alpha Register
"?"         ;Finish the sentence
AOFF
PROMPT      ;Ask the user for a number
STO IND 00  ;Store user's input as control #
ISG 00      ;Increment counter
GTO 01      ;Loop if counter <= 5
END         ;Optional
```

The control number (1.005 —stored in Memory Register #0) not only controls the number of times the instructions in the loop are executed, it also acts as an address pointer. The counter value acts as the address where the data will be stored. Each time through the loop, its value is incremented by one, causing the input number to be stored to a different location. The same instruction is used to store the number: STO IND 00.

Conditional Branching Instructions

Programming functions with question marks in their names are known as ***conditional branching instructions***. Each conditional branch instruction does its particular test and then executes the next line in the script if the test returns `TRUE`. If the test returns `FALSE`, the next line is skipped.

In the example script shown below, the conditional branch instruction used is `X=0?`. If the X-Register contains `0`, we exit the loop. Otherwise, the next line is skipped (`GTO EXIT`) which causes `DSE` to test and decrement the counter value and continue looping back to `LBL 00`.

```
LBL COMPARE ;Optional Script name
RPN         ;Use Reverse Polish Notation
25.020      ;Control number
STO 00      ;Store control number
;-----      ;End of initialization
LBL 00      ;The main LOOP
RCL 00      ;Fetch control number
INT         ;Toss fractional part
22          ;Enter 22
-           ;Control number - 22
X=0?        ;Is it zero?
GTO EXIT    ;Yep-- goto the exit
DSE 00      ;Nope- decrement the counter
GTO 00      ;  and loop
;-----      ;End of LOOP
LBL EXIT    ;End of the line
END         ;End of the script (optional)
```


Program Scripts for Sale

If you have a problem that requires a special script, there is a good chance that Dubl-Click Software has the script you're looking for. Our extensive CCS program script library contains scripts that are applicable to many situations. These scripts are sold in ***Script Packs***. Call CCS Technical Support for details about pricing and script availability.

Calling All Scripts

If you think your program script has got the "right stuff" then Dubl-Click would like to hear from you. We pay royalties on any program script that we accept, and publish in a script pack.

Submitted scripts must be original. We will not publish any program scripts which can be found in the public domain, or that have been taken from a commercial source, such as the *HP-41 User Group Library.*

Chapter 9

Trouble Shooting: Q&A

Q: Why won't my file(s) open from the *Finder*?

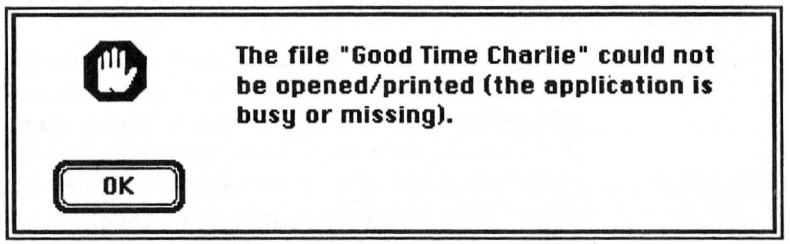


Figure 9-1 Application busy/missing dialog

A: If the dialog box in Figure 9-1 appears when you double-click a file with a suitcase icon (shown at left) from the *Finder*, the problem is that your disk does not contain the *Font/DA Mover*— the application that is used to install fonts and desk accessories. Read Chapter 2 for instructions on installing fonts and desk accessories.

If the dialog box appears when you double-click a CCS work file, you do not have CCS on your disk.

If the dialog box appears when you double-click any other file, you are simply missing the application that created the file you double-clicked.

Q: What is the *Finder*?

A: *The Finder* is the application that has little icons (pictures) of all your disks and files scattered about the screen. It is most easily recognized by the Trash Can icon in the lower right corner of the screen. The *Finder* is also sometimes referred to as the *Macintosh Desktop*.

Q: When I insert the *CCS 2.0* disk, this dialog box appears. What's wrong?

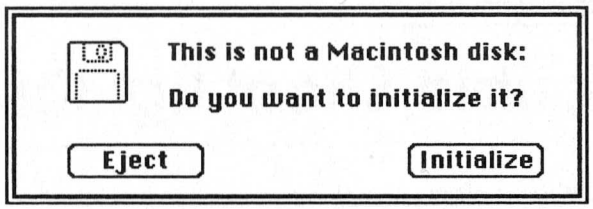


Figure 9-2 Initialize disk dialog. ***Please eject the disk!***

A: Two possibilities: ① You are attempting to read an 800K disk on a 400K drive. If you don't have an 800K drive, read page *v* regarding 400K disk exchanges, ② You're reading the *CCS 2.0* disk on an 800K drive, but it is connected to a Macintosh which doesn't know how to read hierarchical disks. Make sure you are using **System 3.2** or newer and that the **Hard Disk 20** file is in the **System** folder on your startup disk. (Get both from your Apple dealer).


Hard Disk 20

Q: When I open the *CCS 2.0* disk in the *Finder*, the only thing on the disk is an empty folder. Why?

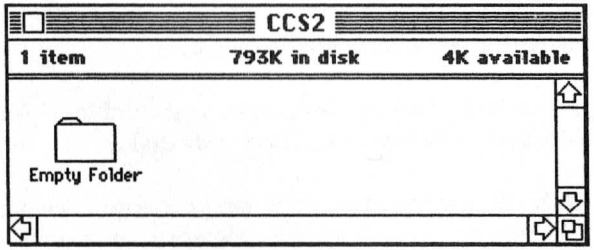


Figure 9-3 HFS disk viewed as flat directory

A: Your startup disk is using an older version of the **System** file which does not contain the information required to read hierarchical disks. Update your startup disk with the latest **System** file and *Finder*. If your startup disk is single-sided (400K), you will have to create a new double-sided (800K) disk first, and copy the files from the old startup disk and the current **System** file and *Finder* onto the newly created 800K startup disk. Your local authorized Apple dealer has the latest **System** file and *Finder*.

Q: Why doesn't the cursor change to a plug tool when I hold down the `OPTION` key?

A: This power-user shortcut only works while the Magic hand is selected. If the Magic Hand is selected, and the `CapsLock` key is down, this shortcut won't work.

Q: Why doesn't the cursor change to a keyboard tool when I hold down the `OPTION+SHIFT` keys?

A: This power-user shortcut only works while the Magic hand is selected. If the Magic Hand is selected, and the `CapsLock` key is down, this shortcut does not work either.

Q: Why does CCS beep at me when I try to drag a Tall LED, Standard LED, or Paper tape from the Parts palette?

A: Each of these parts has a limit of one per calculator. If CCS is beeping at you, it is because you are attempting to exceed this limit. The Standard LED and the Tall LED are both considered to be a MainLED— you cannot have both on your calculator.

Q: Why can't I drag a part onto the calculator? Everytime I try to drag a part onto the Calcshell nothing happens: I don't see the part in the Calcshell.

A: CCS will not allow you to drop a part on top of a calculator part, although you can drop a part on top of a graphic part (like a picture, rectangle, line, or caption). Note that you cannot drop a part on the text at the top of a multi function key either.

Q: How can I drag both a caption (graphic part that I've typed in using the text tool) and a SLED?

A: Use the `SHIFT` key to select both parts, then let go of the `SHIFT` key and drag one part. When you drag one selected part, all the selected parts will move along with it.

Q: I've got a calculator that takes up most of my screen (on a standard MacPlus or MacSE). I'd like to place a few keys off of the calculator shell, but I can't move the Work window or the CalcShell.

A: You don't have to drag the Work window by its title bar. You can click anywhere in the Work window and drag it around by first holding down the ⌘ key (so that the four-way arrow cursor appears— shown in margin).

Q: Why can't I move the Tools or Parts palettes with my mouse using the four-way arrow cursor? I press the ⌘ key and the cursor still looks like an arrow.

A: The CapsLock key is down.

Q: I've just used the rectangle tool to draw a box in my calculator. Why can't I see the box?

A: Your box is filled with a white pattern, and has no border frame. Try selecting all the parts in your Calcshell, then select a line weight of at least one pixel, using the Line Weight Selector in the Parts palette. Your box should then be visible.

Q: Everytime I click on the pattern in the Tools palette the entire calculator redraws. I want the pattern to go in the box that I've made with the rectangle tool. How can I do this?

A: You are painting the Calcshell, not the box. You must first select your box using the Magic Hand tool. Then you may then click on the pattern selector in the Tools palette.

Q: How do I get the menu items in the Graphics menu to work? They are all grayed out.

A: Many items in the **Graphics** menu only work on selected graphic parts. Select a graphic part first using the Magic Hand tool.

Q: "Select External Shape..." from the Graphics menu is grayed-out. How can I use it?

A: Select any part first using the Magic Hand tool.

Q: Why can't I edit the top function label of my multi function key in the *FatBits* dialog?

A: It is not considered part of the keycap, and is only a text label. Edit the TOP function's label with the Text tool.

Q: Why can't I get into the *FatBits* dialog by clicking the Pencil tool on a key?

A: The **FatBits** item on the **Graphics** menu must be checked first. You can also get into FatBits by holding down the ⌘ key and clicking the Pencil tool on a key.

Q: I'm in the *Wiring* dialog. Why can't I assign the function I want to my multi function key? Instead I get a function that I already wired to another position of my multi function key.

A: You are selecting the function first, then clicking on the **Top, Primary,** or **Bottom** radio button. This is backwards. You *first* need to select the position by clicking the radio buttons, *then* select the function itself from the scrolling list.

Q: Why doesn't the picture in the *Map to Keyboard* dialog box look like my keyboard?

A: Either CCS cannot find a picture of your keyboard in the *Key Layout* file, or CCS cannot locate the *Key Layout* file itself. Make sure the *Key Layout* file is in the System Folder on the disk where CCS resides. The *Key Layout* file does not include layouts for many non-Apple keyboards. If you have a non-Apple keyboard whose picture is not drawn in the *Map to Keyboard* dialog box, please contact us. We will try to work with the manufacturer to produce a keyboard layout for you.

Q: I am using an Extended Keyboard with function keys across the top of the keyboard. Why can't I map any of these keys to my calculator's keys?

A: According to Apple Computer, these keys are reserved for use with macro programs such as *Macro Maker* (which comes with the *System* and *Finder*). Without getting too technical, these keys all produce the same ASCII character. If CCS permitted you to map to these keys, they would all be mapped to the same function.

Q: Why do the some of the keys in my ten-key pad show up as arrows in the *Map to Keyboard* dialog box?

A: Because those keys are arrows according to the keyboard layout you are using in the ***Map to Keyboard*** dialog. With some keyboard layouts, you must press the SHIFT key to turn the arrows into arithmetic operators (like +, -, *, /). You might be happier using a different keyboard layout which you may select using the pop-up menu in the top right of the ***Map to Keyboard*** dialog box.

Q: Why do some of my keys have little rectangles drawn on their keycaps after I pull down *Show Keyboard Mapping* from the menu?

A: You have mapped the keystroke to a non-alpha numeric key (like the RETURN, ENTER, CLEAR, TAB, HELP, HOME, or arrow keys). There is no font character which corresponds to the mapped character, so the font draws a rectangle instead.

Q: Why can't I select any tools or parts, even though the Parts and Tools palettes are visible? My cursor looks like a four-way arrow.

A: You do not have a Work window open.

Q: Why doesn't anything happen when I click in the Parts palette's zoom box?

A: The Parts palette is already zoomed as large is it will zoom. If you want it to zoom to another size, resize the window using the window's grow box. The next time you click the zoom box, the window will toggle between these two sizes.

Q: Why isn't there a scroll bar in my Parts palette?

A: There really is a scroll bar, but it is not highlighted because the Work window is not open, or because a desk accessory is the frontmost window.

Q: Why do my Tools and Parts palettes dissappear sometimes in *MultiFinder*?

A: When CCS is not the frontmost application, it automatically hides its palette windows to get them out of your way. CCS leaves the Work window visible. Note: under *MultiFinder*, the palettes will not be automatically hidden if a desk accessory is the frontmost window.

Q: I changed one pattern in the Tools palette. How can I get the unchanged patterns palette to come back?

A: Pull down the **Load Patterns** item from the **File** menu. You can read patterns in from another CCS work file, or a *MacPaint* file. You can also quit CCS then reopen it.

Q: How can I edit and save cursors?

A: Edit them just as you would a pattern; click on the cursor displayed in the Cursor Selector (in the Tools palette). A dialog box will appear, allowing you to edit the cursor. Patterns and cursors are always saved with each work file.

Q: When I decide to save my calculator as a DA or as an application, I get a dialog box that asks me if I want to replace the current version of my calculator work file. Why does this happen?

A: This is normal. We want to make sure you don't lose any of your work. Pulling down **Save As...** always causes a new work file to be saved. If you do not change the name of the calculator being saved, the new work file will replace the old work file.

Q: Why are two unnamed, 9-point fonts installed each time I use my calculator, even though I remove them with the *Font/DA Mover*?

A: CCS calculators use two private fonts to draw themselves. When you use a calculator, the calculator looks for these private CCS fonts. If the fonts have not been installed, the calculator installs a new pair of fonts. When you remove the fonts, the calculator replaces them the next time you use the calculator. By the way, the two fonts take up very little memory.

Q: My friend gave me a calculator work file that he created. Why does the lettering that he placed in the calculator draw funny?

A: You don't have the same fonts installed that he does. Try selecting the part that "draws funny" and changing it by selecting a new font.

Q: Why won't my function execute when I click on the key its wired to? I am in CCS.

A: You're not in Test mode. You can tell if you are in Test mode because the Work window falls away, turning the Calcshell into a window titled "Test mode." The **Test mode** item in the **Calculator** menu will have a check next to it, while most the other menu items will be grayed-out.

Q: How do I get the top and bottom functions on my calculator keys to work?

A: Prefix a press on your multi function key with a press on either the TOP or BOT pushbuttons. Make sure your calculator is wired with a TOP and/or BOT pushbutton. If the TOP or BOT pushbuttons are off the Calcshell, make sure they are mapped to keyboard characters so you can access the TOP and BOT shift modes.

Q: When I press keys wired as arithmetic operators (+, −, *, /), nothing happens. Why?

A: You are trying to key-in SAN equations, but your calculator is in RPN mode. For example, if you key-in: 2+3= while in RPN, all you will see is 2 in the Y-Register and 3 in the X-Register.

Q: When I key-in numbers, they appear on the Paper tape, but not in the MainLED. Also, when I press a key wired as a function (like +), the calculator beeps. Why?

A: Your calculator is in TEXT mode, which means that you are editing the contents of the Paper tape. You cannot perform calculations in TEXT mode. To exit TEXT mode, and resume normal calculation mode, click on the MainLED.

Q: How do I get out of ALPHA mode?

A: Press the ALPHA pushbutton. If you didn't wire one, you will not be able to get out of ALPHA mode.

Q: I'm storing data to Memory Registers. Why does my data change after I do a few calculations?

A: You executed a financial or statistical function which uses a block of Memory Registers to store cumulative data. See Figures 5-2 and 5-3 for a list of which registers the functions use. The solution is to use a different set of Memory Registers when storing your data.

Q: Why do I keep getting a `Data Error` message whenever I try to solve for one of the financial functions such as `N`, `I`, `PV`, `PMT`, `FV`?

A: You have not input enough known values. You ***must*** input at least three known values to solve for a financial function.

Q: Why do I get beeped when I use `DDAY` or `DATE+`?

A: One or both of your arguments are not in `DMY` or `MDY` format. Consult Chapter 6 for more on `DDAY` and `DATE+`.

Q: How can I do a quick conversion between `HEX`, `BINARY`, `DECIMAL`, and `OCTAL` without going through all of the different conversion functions, or pressing clickstop switches?

A: Wire SLEDs to display the X-Register in `HEX`, `BINARY`, `DECIMAL`, and `OCTAL` formats. CCS comes with a work file called *INT/HEX/BIN* which has SLEDs wired in this way.

Q: Why are only 24 characters printed on the Paper tape?

A: You are in `ALPHA` mode. The Alpha Register only holds 24 characters, and in `ALPHA` mode the Paper tape just reflects what's in the Alpha Register.

Q: Why do the numbers on my Paper tape disappear past the left edge of the Paper tape display?

A: Your Paper tape is not stretched wide enough to display all the characters on the line. Perhaps you have the `MARGNLS` pushbutton in the *ON* position (marginals can take up an additional 8 characters on the Paper tape). You can either stretch your Paper tape wider, or wire a clickstop switch to change the justification of the text on the Paper tape— in effect, allowing you to scroll the tape horizontally.

Q: Why won't my Alpha Register LED display all 24 characters?

A: It is not stetched wide enough. Unlike the HP-41C, CCS calculators do not scroll LEDs when there are more characters in the Alpha Register than will fit on the LED. Instead, you can simply stretch the LED to display the full Alpha Register. During construction, a right-justified number is drawn on each LED. This number represents the maximum number of characters LED can display.

Q: Why aren't the `LINE#` and "Input/Display Format" displayed in my Annunciator.

A: The Annunciator display is not stretched wide enough to display them.

Q: Why do I get a `Non Existent` error when I try to `STO` or `RCL` to `S5` (stack register 5)?

A: Chances are you only wired four Stack Registers. To see how many registers have been wired, you can press the `MEM` key. To wire more registers, use the ***Memory Allocation*** dialog box.

Q: Why do I get a `Non Existent` error when I try to `STO` or `RCL` a Memory Register?

A: Chances are you are trying to use Memory Register which does not exist. To see how many registers have been wired, you can press the `MEM` key. To wire more registers, use the ***Memory Allocation*** dialog box.

Q: How can I type the Σ (Sigma) symbol?

A: Hold down the `OPTION` key and type a lowercase `w` on your keyboard.

Q: How can I get the Annunciator to display MEM_0 instead of the Memory Register number that is there now?

A: Change the Default Memory Register via the Selector SLED. You can also change the Default Memory Register by executing a function that uses a Memory Register (such as STO or RCL).

Q: Why is my calculation's result displayed in Scientific Notation when the input/display format is not SCI?

A: The MainLED is not stretched wide enough to display all the significant digits of your result. This is particularly true if you are using a Tall LED as the MainLED, or if you are in FLOAT input/display format.

Q: Why does the result of my calculation have so many digits after the decimal point?

A: You are in FLOAT input/display format. Go into FIXED or DOLLARS formats.

Q: How can I speed up the execution of my script? All the displays flash while the script is running.

A: Make the first line of your script the TALKOFF instruction, which will prevent displays from being updated while a script is running. This will dramatically speed up script execution.

Q: I've put TALKOFF in my script, but it seems like my script slows down after every PROMPT, PSE, or AVIEW.

A: Insert a TALKOFF instruction after every PROMPT, PSE, or AVIEW function in your script.

Q: Why won't my script work when I type in its name into the XEQ dialog box?

A: Perhaps you didn't spell the script properly in the XEQ dialog box. Your script name might conflict with a reserved keyword, like a function name (see Appendices 2 and 3 for a list of keywords). Your script may not be in memory. It is often helpful to XEQ scripts from the PRGM dialog box where you can see if the script is in memory. You can execute the script by clicking on its name instead of having to spell it.

Q: Why does the PRGM dialog box show scripts that belong to another calculator? I didn't create them in this calculator.

A: During construction, CCS doesn't delete scripts from memory when you close a Work window. If you want to delete the scripts that came from a previous calculator, go into Test mode and delete them via the PRGM dialog box.

Q: Why doesn't my script's entire name appear on the keycap to which it was wired?

A: Your key is not wide enough to display the entire name. Wire the script to a wider key.

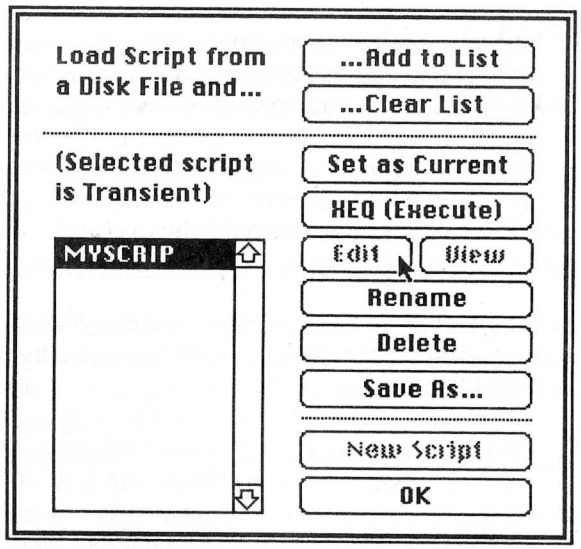
Q: How do I unwire a script once I've wired it to a key?

A: Delete the key by dragging it into the trash can in the Tools palette, or by selecting the key and typing the backspace on your keyboard.

Q: Why does my script name dissappear from the Wiring dialog after I wire my script to a key?

A: Each script can only be wired to a single key. Once a script has been wired to a key, its name no longer appears in the list of scripts available for wiring.

Q: Why are the *Edit*, *View*, and *New Script* buttons grayed out in the *PRGM* dialog box?



A: ***There are two possibilities:***

Your calculator does not contain a Paper tape. Without a Paper tape, there is no way to display the script, ***or...***

The script you are trying to edit or view is a locked. If the script is locked, the words "Selected Script is Locked" will appear above the selected script name. You may not edit or view locked scripts in finished calculators.

Q: My script runs OK in Test mode, but I keep getting `Script Error` messages when I run my script from the calculator desk accessory. Why?

A: You are running a script which has been wired to a key. The functions referenced in the script may not be saved in the calculator desk accessory. To remedy this, load your calculator work file into CCS. Pull down the ***Preferences*** dialog box from the **Calculator** menu and check the box titled **Save All Functions**. Then save your calculator desk accessory again.

Q: My script has stopped running because of a problem. How can I find out which line caused the problem?

A: The number of the next line to be executed is highlighted when you edit or view a script on the Paper tape.

Q: How can I set my script to start executing from a specific line number?

A: Edit your script in the Paper tape, and click on the line number where you want execution to begin. The line number will be highlighted (as in the illustration at left). This technique is sometimes helpful when debugging a script: click on the line where you want to begin testing and single step through the script.

Appendix 1

All Functions Listed Alphabetically

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
'	Const	N	N	N	N	Y	N	5-4
"	Const	N	N	N	N	Y	N	8-11
(	Const	N	N	N	N	Y	N	6-3
)	Const	N	N	N	N	Y	N	6-3
*	Math	E	Y	N	N	Y	Y	6-23
+	Math	E	Y	N	N	Y	Y	6-23
−	Math	E	Y	N	N	Y	Y	6-23
.	Const	N	N	N	N	Y	N	6-3
/	Math	E	Y	N	N	Y	Y	6-23
:	Const	N	N	N	N	Y	N	5-4
;	Alphas	N	N	N	N	Y	N	8-11
^	Const	N	N	N	N	Y	N	5-5
=	System	N	N	N	N	Y	Y	6-45
%	Math	E	Y	N	N	Y	Y	6-23
%CH	Math	E	Y	N	N	Y	Y	6-23
%T	Math	E	Y	N	N	Y	Y	6-24
0 - 9	Const	N	N	N	N	Y	N	6-2
1/X	Math	E	Y	N	N	Y	Y	6-24
10TOX	Math	E	Y	N	N	Y	Y	6-24
12*	Finan	N	Y	N	Y	Y	Y	6-18
12/	Finan	N	Y	N	Y	Y	Y	6-18
A - F	Const	N	N	N	N	Y	N	6-2
A=R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
ABS	Math	E	Y	N	N	Y	Y	6-24
ACOS	Math	E	Y	N	N	Y	Y	6-24
ADATE	Date...	E	N	N	N	Y	Y	6-12
ADV	System	E	N	N	N	Y	Y	6-45
ALARM	PushBtn	N	N	N	N	N	N	6-53

E = Enable, D = Disable, C = Clear, Y = Yes, N = No

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
ALENG	Regs	E	N	N	N	Y	Y	6-35
AMORT	Finan	E	N	N	N	Y	Y	6-18
AND	BitOps	E	Y	N	N	Y	Y	6-4
ANUM	Regs	E	N	N	N	Y	Y	6-35
any digit	Alphas	N	N	N	N	Y	N	6-2
AOFF	PushBtn	N	N	N	N	Y	Y	6-54
AON	PushBtn	N	N	N	N	Y	Y	6-54
APPEND	Regs	N	N	N	N	Y	Y	6-35
ARCL	Regs	E	N	Y	Y	Y	Y	6-36
AROT	Regs	E	N	Y	N	Y	Y	6-36
ASHF	Regs	E	N	N	N	Y	Y	6-36
ASIN	Math	E	Y	N	N	Y	Y	6-24
ASL	BitOps	E	Y	N	N	Y	Y	6-4
ASR	BitOps	E	Y	N	N	Y	Y	6-4
ASTO	Regs	E	N	Y	Y	Y	Y	6-36
ATAN	Math	E	Y	N	N	Y	Y	6-24
ATIME	Date...	E	N	N	N	Y	Y	6-12
ATIME24	Date...	E	N	N	N	Y	Y	6-12
ATOX	Regs	E	N	N	N	Y	Y	6-36
AVIEW	Regs	E	N	N	N	Y	Y	6-37
backsp.	System	D	N	N	N	N	N	6-45
BCHG	BitOps	E	Y	Y	N	Y	Y	6-4
BCLR	BitOps	E	Y	N	N	Y	Y	6-5
BEEP	System	N	N	N	N	Y	Y	6-45
BEG	Finan	E	N	N	N	Y	Y	6-18
BINARY	ClikSW	N	N	N	N	Y	Y	5-3
BITS	BitOps	E	Y	Y	N	Y	Y	6-5
BOT	PushBtn	N	N	N	N	N	N	6-54
BSET	BitOps	E	Y	N	N	Y	Y	6-5
BTST	BitOps	E	N	N	N	Y	Y	6-5
CE	Clear	N	N	N	N	N	N	6-10
CENTER	ClikSW	N	N	N	N	Y	Y	6-52
CF	Clear	E	N	Y	N	Y	Y	6-11
CFJ	Finan	E	N	N	N	Y	Y	6-18
CFO	Finan	E	N	N	N	Y	Y	6-18
CHS	Math	N	N	N	N	Y	Y	6-25
CL	Clear	D	N	N	N	Y	Y	6-10
CLA	Clear	E	N	N	N	Y	Y	6-10
CLC	Clear	C	C	N	N	Y	Y	6-10
CLD	Clear	E	N	N	N	Y	Y	6-10

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
CLK12	Date...	E	N	N	N	Y	Y	6-13
CLK24	Date...	E	N	N	N	Y	Y	6-13
CLKT	Date...	E	N	N	N	Y	Y	6-13
CLKTD	Date...	E	N	N	N	Y	Y	6-13
CLOCK	Date...	E	N	N	N	Y	Y	6-13
CLP	Clear	N	N	N	N	N	N	6-11
CLRF	Clear	E	N	N	N	Y	Y	6-11
CLRG	Clear	E	N	N	N	Y	Y	6-11
CLST	Clear	C	C	N	N	Y	Y	6-11
CLX	Clear	D	N	N	N	Y	Y	6-11
CLΣ	Clear	E	N	N	N	Y	Y	6-11
COMMAS	PushBtn	N	N	N	N	Y	Y	6-55
COPY	System	E	N	N	N	Y	Y	6-45
COS	Math	E	Y	N	N	Y	Y	6-25
CSC	Math	E	Y	N	N	Y	Y	6-25
CTN	Math	E	Y	N	N	Y	Y	6-25
CUT	System	D	N	N	N	Y	Y	6-46
D-R	Math	E	Y	N	N	Y	Y	6-25
DATE	Date...	E	Y	N	N	Y	Y	6-13
DATE+	Date...	E	Y	N	N	Y	Y	6-13
DB	Finan	E	N	N	N	Y	Y	6-19
DDAYS	Date...	E	Y	N	N	Y	Y	6-14
DEG	System	N	N	N	N	Y	Y	6-46
DEGREES	ClikSW	N	N	N	N	Y	Y	5-3
DISK	PushBtn	N	N	N	N	Y	Y	6-54
DMY	Date...	E	N	N	N	Y	Y	6-14
DOLLARS	ClikSW	N	N	N	N	Y	Y	5-3
DOW	Date...	E	Y	N	N	Y	Y	6-14
DOZENAL	ClikSW	N	N	N	N	Y	Y	5-3
DRAW•	System	E	N	N	N	Y	Y	6-46
DSE	PgmFNs	N	N	Y	N	Y	Y	6-29
EEX	System	N	N	N	N	Y	N	6-47
END	N	N	N	N	N	Y	N	6-63
ENDP	Finan	E	N	N	N	Y	Y	6-19
ENG	ClikSW	N	N	N	N	Y	Y	5-4
ENTER	System	D	N	N	N	Y	Y	6-46
ETO1	Const	E	Y	N	N	Y	Y	6-2
ETOX	Math	E	Y	N	N	Y	Y	6-25
EXT	BitOps	E	Y	N	N	Y	Y	6-5
FACT	Math	E	Y	N	N	Y	Y	6-26

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
FC?	PgmFNs	E	N	Y	N	Y	Y	6-30
FC?C	PgmFNs	E	N	Y	N	Y	Y	6-30
FIX	System	N	N	Y	N	Y	Y	6-47
FIXED	ClikSW	N	N	N	N	Y	Y	5-4
FLOAT	ClikSW	N	N	N	N	Y	Y	5-4
FMOVE	Finan	E	N	N	Y	Y	Y	6-22
FRC	Math	E	Y	N	N	Y	Y	6-26
FREG?	Finan	E	N	Y	N	Y	Y	6-22
FS?	PgmFNs	E	N	Y	N	Y	Y	6-30
FS?C	PgmFNs	E	N	Y	N	Y	Y	6-30
FTINCH	ClikSW	N	N	N	N	Y	Y	5-4
FV	Finan	N	N	N	N	Y	Y	6-19
GETAS	System	N	N	N	N	Y	Y	6-48
GETIME	Date...	N	N	N	N	Y	Y	6-14
GETP	PrgmCat	N	N	N	N	Y	Y	8-5
GETR	Regs	N	N	N	N	Y	Y	6-37
GETSUB	PrgmCat	N	N	N	N	Y	Y	8-5
GTO	PgmFNs	N	N	Y	N	Y	Y	6-30
HEX	ClikSW	N	N	N	N	Y	Y	5-4
HMS	Date...	E	Y	N	N	Y	Y	6-14
HMS+	Date...	E	Y	N	N	Y	Y	6-15
HMS-	Date...	E	Y	N	N	Y	Y	6-15
HR	Date...	E	Y	N	N	Y	Y	6-15
HRSMIN	ClikSW	N	N	N	N	Y	Y	5-4
I	Finan	N	N	N	Y	Y	Y	6-19
IND	N	N	N	N	N	Y	N	6-62
INST	Finan	E	N	N	N	Y	Y	6-20
INT	Math	E	Y	N	N	Y	Y	6-26
INTEGER	ClikSW	N	N	N	N	Y	Y	5-4
IRR	Finan	N	N	N	N	Y	Y	6-20
ISG	PgmFNs	N	N	Y	N	Y	Y	6-31
KEYMAP	PushBtn	N	N	N	N	Y	Y	6-55
LASTX	Regs	E	N	N	N	Y	Y	6-37
LBL	N	N	N	N	N	Y	Y	8-11
LEFT	ClikSW	N	N	N	N	Y	Y	6-52
LINEOFF	PushBtn	N	N	N	N	Y	Y	6-55
LINEON	PushBtn	N	N	N	N	Y	Y	6-55
LJX	BitOps	E	Y	N	N	Y	Y	6-6
LN	Math	E	Y	N	N	Y	Y	6-26
LOG	Math	E	Y	N	N	Y	Y	6-26

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
LSL	BitOps	E	Y	N	N	Y	Y	6-6
LSR	BitOps	E	Y	N	N	Y	Y	6-6
MARGOFF	PushBtn	N	N	N	N	Y	Y	6-55
MARGON	PushBtn	N	N	N	N	Y	Y	6-55
MASKL	BitOps	E	Y	N	N	Y	Y	6-6
MASKR	BitOps	E	Y	N	N	Y	Y	6-6
MDY	Date...	E	N	N	N	Y	Y	6-15
MEAN	Stats	E	Y	N	N	Y	Y	6-42
MEM	System	E	N	N	N	Y	Y	6-47
MUZOFF	PushBtn	N	N	N	N	Y	Y	6-56
MUZON	PushBtn	N	N	N	N	Y	Y	6-56
N	Math	N	N	N	Y	Y	Y	6-20
NJ	Finan	E	N	N	N	Y	Y	6-20
NOT	BitOps	E	Y	N	N	Y	Y	6-7
NPV	Finan	E	N	N	N	Y	Y	6-20
OCTAL	ClikSW	N	N	N	N	Y	Y	5-5
OR	BitOps	E	Y	N	N	Y	Y	6-7
P-R	Math	E	Y	N	N	Y	Y	6-26
PASTE	System	E	N	N	N	Y	Y	6-48
PGRM	PushBtn	N	N	N	N	N	N	8-2
PI	Const	E	Y	N	N	Y	Y	6-2
PMT	Finan	N	Y	N	Y	Y	Y	6-20
POUNDS	ClikSW	N	N	N	N	Y	Y	5-5
PRINT	PushBtn	N	N	N	N	Y	Y	6-55
PROMPT	N	N	N	N	N	Y	Y	6-63
PSE	N	N	N	N	N	Y	Y	6-63
PV	Finan	N	N	N	Y	Y	Y	6-20
R-D	Math	E	Y	N	N	Y	Y	6-27
R-P	Math	E	Y	N	N	Y	Y	6-27
R/S	PgmFNs	N	N	N	N	N	N	6-31
RAD	System	N	N	N	N	Y	Y	6-48
RCL	Regs	E	N	Y	Y	Y	Y	6-37
RDN	System	E	Y	N	N	Y	Y	6-48
RIGHT	ClikSW	N	N	N	N	Y	Y	6-52
RLC	BitOps	E	Y	N	N	Y	Y	6-7
RLCN	BitOps	E	Y	N	N	Y	Y	6-7
RLN	BitOps	E	Y	N	N	Y	Y	6-7
RND	System	E	Y	Y	N	Y	Y	6-27
ROL	BitOps	E	Y	N	N	Y	Y	6-7
ROR	BitOps	E	Y	N	N	Y	Y	6-8

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
RPN	ClikSW	C	C	N	N	Y	Y	5-6
RRC	BitOps	E	Y	N	N	Y	Y	6-8
RRCN	BitOps	E	Y	N	N	Y	Y	6-8
RRN	BitOps	E	Y	N	N	Y	Y	6-8
RTN	PgmFNs	N	N	N	N	Y	Y	6-31
RUP	System	E	N	N	N	Y	Y	6-49
Σ+	Stats	N	Y	N	N	Y	Y	6-42
Σ-	Stats	N	Y	N	N	Y	Y	6-42
ΣREG	Stats	E	N	Y	Y	Y	Y	6-43
ΣREG?	Stats	E	N	N	N	Y	Y	6-43
SAN	ClikSW	C	C	N	N	Y	Y	5-6
SAVEAS	System	N	N	N	N	Y	Y	6-49
SAVER	Regs	N	N	N	N	Y	Y	6-37
SCI	ClikSW	N	N	N	N	Y	Y	5-5
SDEV	Stats	E	Y	N	N	Y	Y	6-42
SEC	Math	E	Y	N	N	Y	Y	6-27
SELALL	System	N	N	N	N	Y	Y	6-49
SETDATE	Date...	E	N	N	N	Y	Y	6-15
SETIME	Date...	E	N	N	N	Y	Y	6-16
SF	PgmFNs	E	N	Y	N	Y	Y	6-31
SIGN	Math	E	N	N	N	Y	Y	6-27
SIN	Math	E	Y	N	N	Y	Y	6-28
SL	Finan	E	N	N	N	Y	Y	6-21
SOYD	Finan	E	N	N	N	Y	Y	6-21
SQRT	Math	E	Y	N	N	Y	Y	6-28
SST	PgmFNs	N	N	N	N	N	N	6-31
ST*	Regs	E	N	Y	Y	Y	Y	6-38
ST+	Regs	E	N	Y	Y	Y	Y	6-38
ST-	Regs	E	N	Y	Y	Y	Y	6-38
ST/	Regs	E	N	Y	Y	Y	Y	6-38
STO	Regs	E	N	Y	Y	Y	Y	6-38
STOLAST	Regs	E	Y	N	N	Y	Y	6-37
STOP	N	N	N	N	N	Y	Y	8-13
SUBT	System	N	N	N	N	Y	Y	6-49
SW	PushBtn	N	N	N	N	Y	Y	6-56
TALKOFF	N	N	N	N	N	Y	Y	6-63
TALKON	N	N	N	N	N	Y	Y	6-63
TAN	Math	E	Y	N	N	Y	Y	6-28
TEXTOFF	PprTape	N	N	N	N	Y	Y	5-7
TEXTON	PprTape	N	N	N	N	Y	Y	5-7

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
TIME	Date...	E	N	N	N	Y	Y	6-16
TIMEDSP	ClikSW	E	Y	N	N	Y	Y	5-5
TONE	System	N	N	Y	N	Y	Y	6-49
TONEX	System	N	N	N	N	Y	Y	6-50
TOP	PushBtn	N	N	N	N	Y	Y	6-56
TOT	System	N	N	N	N	Y	Y	6-50
TPRINT	System	N	N	N	N	Y	Y	6-50
VIEW	Regs	E	N	Y	Y	Y	Y	6-39
WSIZE	BitOps	E	N	Y	N	Y	Y	6-8
WSTAT	BitOps	E	N	N	N	Y	Y	6-8
WTDAVG	Stats	E	N	N	N	Y	Y	6-44
XASA	Regs	E	N	N	N	Y	Y	6-39
XEQ	PgmFNs	E	N	Y	N	Y	N	6-32
XOR	BitOps	E	Y	N	N	Y	Y	6-8
XROOTY	Math	E	Y	N	N	Y	Y	6-28
XSWAPF	Regs	E	N	N	N	Y	Y	6-39
XSWAPR	Regs	E	N	Y	Y	Y	Y	6-39
XSWAPY	Regs	E	N	N	N	Y	Y	6-39
XTO2	Math	E	Y	N	N	Y	Y	6-28
XTOA	Regs	E	N	N	N	Y	Y	6-39
X<0?	PgmFNs	E	N	N	N	Y	Y	6-33
X<=0?	PgmFNs	E	N	N	N	Y	Y	6-33
X<=NN?	PgmFNs	E	N	N	N	Y	Y	6-34
X<=R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X<=Y?	PgmFNs	E	N	N	N	Y	Y	6-33
X<NN?	PgmFNs	E	N	N	N	Y	Y	6-34
X<R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X<Y?	PgmFNs	E	N	N	N	Y	Y	6-33
X=0?	PgmFNs	E	N	N	N	Y	Y	6-33
X=NN?	PgmFNs	E	N	N	N	Y	Y	6-34
X=R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X=Y?	PgmFNs	E	N	N	N	Y	Y	6-33
X>0?	PgmFNs	E	N	N	N	Y	Y	6-33
X>=NN?	PgmFNs	E	N	N	N	Y	Y	6-34
X>=R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X>NN?	PgmFNs	E	N	N	N	Y	Y	6-34
X>R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X>Y?	PgmFNs	E	N	N	N	Y	Y	6-33
X≠0?	PgmFNs	E	N	N	N	Y	Y	6-33
X≠NN?	PgmFNs	E	N	N	N	Y	Y	6-34

Function Name	Wire From:	Stack Lift?	LastX Save?	Prompt FN?	Default MEM?	OK in Script?	OK to XEQ?	See Page:
X≠R?	PgmFNs	E	N	Y	Y	Y	Y	6-34
X≠Y?	PgmFNs	E	N	N	N	Y	Y	6-33
YDFEET	ClikSW	N	N	N	N	Y	Y	5-5
YEN	ClikSW	N	N	N	N	Y	Y	5-5
YTOX	Math	E	Y	N	N	Y	Y	6-28

E = Enable, D = Disable, C = Clear, Y = Yes, N = No

Appendix 2

Executable (XEQable) Function Names

This table lists all the function names which are valid input in response to an XEQ prompt.

Function Name	Function Description
*	Multiplication
+	Addition
-	Subtraction
/	Division
=	Equals
%	Percent
%CH	Percent Change
%T	x Percent Of Number
1/X	Reciprocal
10TOX	Common Exponential
12*	Multiply x by 12 and store in Financial Register
12/	Divide x by 12 and store in Financial Register
A=R?	Compare Alpha Register and Memory Register
ABS	Absolute Value
ACOS	Arc Cosine
ADATE	Alpha Date
ADV	Advance Paper tape
ALENG	Alpha Length
AMORT	Amortization
AND	Binary AND
ANUM	Alpha Number
AOFF	Alpha Mode *OFF*
AON	Alpha Mode *ON*
APPEND	Append to Alpha Register
ARCL	Alpha Recall
AROT	Alpha Rotate

Function Name	Function Description
ASHF	Alpha Shift
ASIN	Arc Sine
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
ASTO	Alpha Store
ATAN	Arc Tangent
ATIME	Alpha Time
ATIME24	Alpha Time 24-Hour
ATOX	Alpha To X-Register
AVIEW	Alpha View
BCHG	Bit Change
BCLR	Bit Clear
BEEP	Beep
BEG	Beginning Payment
BINARY	Input/Display Format
BITS	Bit Count
BSET	Set Bit
BTST	Bit Test
CENTER	Center Text in Paper Tape
CF	Clear Flag NN
CFJ	Cash Flow J
CFO	Initial Cash Flow
CHS	Change Sign
CL	Clear (smart clear)
CLA	Clear Alpha
CLC	Clear Calculator (Memory, Stack, Alpha Registers)
CLD	Clear Temporary Display in MainLED
CLK12	Clock 12-Hour Format
CLK24	Clock 24-Hour Format
CLKT	Clock Time
CLKTD	Clock Time/Date
CLOCK	Display Clock
CLRF	Clear Finanacial Registers
CLRG	Clear Memory Registers
CLST	Clear Stack Registers
CLX	Clear X-Register
CLΣ	Clear Statistics Registers
COMMAS	Toggle Commas *ON/OFF*
COPY	Copy To Clipboard
COS	Cosine

Function Name	Function Description
CSC	Cosecant
CTN	Contangent
CUT	Cut To Clipboard
D-R	Degrees To Radians
DATE	Get Date
DATE+	Add Days To Date
DB	Declining Balance Depreciation
DDAYS	Delta Days
DEG	Degrees
DEGREES	Input/Display Format
DISK	Save Paper tape To Disk
DMY	Day Month Year
DOLLARS	Input/Display Format
DOW	Day Of Week
DOZENAL	Input/Display Format
DRAW•	Draw Bullets On Paper tape
DSE	Decrement And Skip
ENDP	Set Payment Mode To End
ENG	Input/Display Format
ENTER	Enter Value
ETO1	Constant Of *e* (2.71828182845904523)
ETOX	Natural Exponential
EXT	Sign extend (bit operation)
FACT	N! Factorial
FC?	Skip If Flag NN Is not Clear
FC?C	Skip If Flag NN Is not Clear, and Clear Flag
FIX	Set Number of Decimal Places
FIXED	Input/Display Format
FLOAT	Input/Display Format
FMOVE	Move Financial Registers
FRC	Extract Fractional Part of X
FREG?	Locate First Financial Register
FS?	Skip If Flag NN is not Set
FS?C	Skip If Flag NN is not Set, and Clear Flag
FTINCH	Input/Display Format
FV	Future Value
GETAS	Read Text File and Display on Paper tape
GETIME	Return Current Time in decimal hours format
GETP	Load Program File
GETR	Get Memory Registers from file

Function Name	Function Description
GETSUB	Load Subroutine
GTO	Go To...
HEX	Input/Display Format
HMS	Decimal Hours To Hours Minutes Seconds
HMS+	Add Times (using HH.MMSS format)
HMS-	Subtract Times (using HH.MMSS format)
HR	Convert from HH.MMSS format to decimal hours
HRSMIN	Input/Display Format
I	Interest
INST	Simple Interest
INT	Extract Integer Part of X
INTEGER	Input/Display Format
IRR	Internal Rate Of Return
ISG	Increment And Skip
KEYMAP	Show/Hide Keymapping
LASTX	Recall from LastX-Register
LEFT	Left Justify Text in Paper tape
LINEOFF	Line Numbering *OFF*
LINEON	Line Numbering *ON*
LJX	Left Justify Word In X-Register
LN	Natural Logarithm
LOG	Common Logarithm
LSL	Logical Shift Left
LSR	Logical Shift Right
MARGOFF	Marginals *OFF*
MARGON	Marginals *ON*
MASKL	Mask Left
MASKR	Mask Right
MDY	Month Day Year Format
MEAN	Means Of Accumulated X & Y Values
MEM	Display Memory Map
MUZOFF	Music with Keypress *OFF*
MUZON	Music with Keypress *ON*
N	Number Of Periods
NJ	Store Number Of Periods Of Cash Flow
NOT	Binary NOT
NPV	Net Present Value
OCTAL	Input/Display Format
OR	Binary OR
P-R	Polar To Rectangular Coordinates

Function Name	Function Description
PASTE	Paste To Clipboard
PI	Value Of π (3.14159265358979323)
PMT	Payment
POUNDS	Input/Display Format
PRINT	Print The Buffer
PSE	Pause Script Execution
PV	Present Value
R-D	Radians To Degrees
R-P	Rectangular To Polar Coordinates
RAD	Radians
RCL	Recall From Memory Register to X-Register
RDN	Roll Stack Down
RIGHT	Right Justify Text in Paper tape
RLC	Rotate Left Thru Carry
RLCN	Multiple Of RLC
RLN	Multiple Rotate Left
RND	Round X to 0 through 9 places
ROL	Rotate Left
ROR	Rotate Right
RPN	Reverse Polish Notation (and clear stack)
RRC	Rotate Right Thru Carry
RRCN	Multiple Of RRC
RRN	Multiple Rotate Right
RTN	Return From Subroutine
RUP	Roll Stack Up
Σ+	Summation Plus
Σ-	Summation Minus
ΣREG	Move Statistical Regs
ΣREG?	Locate Statistical Regs
SAN	Standard Algebraic Notation (and clear stack)
SAVEAS	Save Paper tape In Text File
SAVER	Save Memory Registers in file
SCI	Input/Display Format
SDEV	Standard Deviation
SEC	Secant
SELALL	Select All (Paper tape)
SETDATE	Set Clock Date
SETIME	Set Clock Time
SF	Set Flag NN
SIGN	Return Sign of X

Function Name	Function Description
SIN	Sine
SL	Calc Straight Line Depreciation
SOYD	Sum Of Years Depreciation
SQRT	Square Root
ST*	Store X-Register times MemReg to MemReg
ST+	Store X-Register plus MemReg to MemReg
ST-	Store X-Register minus MemReg to MemReg
ST/	Store X-Register divided by MemReg to MemReg
STO	Store X-Register To Memory Register
STOLAST	Store X-Register To LastX-Register
STOP	Stop Program Script
SUBT	Paper tape Subtotal
SW	Clear Stopwatch / Display Elapsed Time
TAN	Tangent
TEXTOFF	Turn TEXT mode *OFF*
TEXTON	Turn TEXT mode *ON*
TIME	Return Current Time in HMS format
TIMEDSP	Input/Display Format
TONE	Play A Tone Of Value NN
TONEX	Play A Tone Of Value X
TOT	Paper tape Total
TPRINT	Print Paper tape On Printer
VIEW	View Memory Register In MainLED
WSIZE	Set Word Size
WSTAT	Word Size Status
WTDAVG	Weighted Average
XASA	Append X-Register to Alpha Register
XOR	Binary Exclusive OR
XROOTY	Xth Root Of Y
XSWAPF	Exchange X-Register With Flags 0 through 7
XSWAPR	Exchange X-Register With Memory Register
XSWAPY	Exchange X-Register and Y-Register
XTO2	Square Of X
XTOA	Convert X To ASCII
X<0?	Boolean Test
X<=0?	Boolean Test
X<=NN?	Boolean Test
X<=R?	Boolean Test
X<=Y?	Boolean Test
X<NN?	Boolean Test

Function Name	Function Description
X<R?	Boolean Test
X<Y?	Boolean Test
X=0?	Boolean Test
X=NN?	Boolean Test
X=R?	Boolean Test
X=Y?	Boolean Test
X>0?	Boolean Test
X>=NN?	Boolean Test
X>=R?	Boolean Test
X>NN?	Boolean Test
X>R?	Boolean Test
X>Y?	Boolean Test
X≠0?	Boolean Test
X≠NN?	Boolean Test
X≠R?	Boolean Test
X≠Y?	Boolean Test
YDFEET	Input/Display Format
YEN	Input/Display Format
YTOX	Y To The X Power

Appendix 3

Script Function Names

In additon to the Table of ***Executable Function Names*** (listed in Appendix 2), this table contains function names which are also valid program script function names. Note: in program scripts, all function and label names must be entered in capital letters.

Script FN Name	Function Description
Any digit	In ALPHA mode, this is valid input
0 thru 9	Constants 0 through 9 (depends on input format)
A thru F	Constants A through F (in HEX input format)
.	Radix or Decimal point (might be a comma)
:	Colon (depends on input format)
^	Caret (depends on input format)
'	Single Quote (depends on input format)
"	Double Quote (surrounds alpha data)
(	Left Parenthesis. Indicates function precedence
)	Right Parenthesis. Indicates function precedence
;	Semicolon. Remaining text on the line is comment
EEX	Enter Exponent (name is E if imbedded in number)
END	End of Script Marker
ENTER	Y Enter X
LBL	Label. Remaining text on the line is the label name
PROMPT	Display Alpha Register and Stop Script Execution
PSE	Stop Script Execution for 2 seconds, then resume
STOP	Stop Script Execution
TALKON	Turn Displays *ON*
TALKOFF	Turn Displays *OFF*
XEQ	Execute another Script. Remaining text on the line is the label name

Appendix 4 - Prompts

This table lists all functions which display a prompt in the MainLED, and a description of the data they expect in response.

Function	*Expected Response*	*Description of Function*
A=R?	MemReg number	Alpha Register = MemReg?
ARCL	MemReg number	Alpha Recall
AROT	Number of digits	Alpha Rotate
ASTO	MemReg number	Alpha Store
CF	Flag number	Clear Flag NN
DSE	iiii.fffcc	Decrement And Skip
FC?	Flag number	Skip if flag nn is clear
FC?C	Flag number	Skip & clear if flag NN clear
FS?	Flag number	Skip if flag NN is set
FS?C	Flag number	Skip & set if flag NN is set
FIX	Number of places	Set # of decimal places
FMOVE	MemReg number	Move Financial Regs to…
GTO	Script label	Go To...
ISG	iiii.fffcc	Increment And Skip
RCL	MemReg number	X-Reg = MemReg
RND	Number of places	Round X
ΣREG	MemReg number	Move Statistical Regs to…
SF	Flag number	Set Flag NN
ST*	MemReg number	MemReg = MemReg * X
ST+	MemReg number	MemReg = MemReg + X
ST-	MemReg number	MemReg = MemReg - X
ST/	MemReg number	MemReg = MemReg ÷ X
STO	MemReg number	MemReg = X-Register
TONE	Tone Value	Play a Tone of value NN
VIEW	MemReg number	View MemReg in MainLED
WSIZE	Number of bits	Set Word Size (8,16,32 bits)
XSWAPR	MemReg number	Swap MemReg & X
XEQ	Script label or FNname	Execute…

MemReg = *Memory Register nn,* ***X-Reg*** = *the X-Register,*
FN name = *Function Name,* ***Label*** = *script label*

Function	Expected Response	Description of Function
X<R?	MemReg number	X-Reg < MemReg?
X=R?	MemReg number	X-Reg = MemReg?
X=<R?	MemReg number	X-Reg ≤ toMemReg?
X>R?	MemReg number	X-Reg > MemReg?
X>=R?	Memreg number	X-Reg ≥ MemReg?

MemReg = *Memory Register nn,* **X-Reg** = *the X-Register,*
FN name = *Function Name,* **Label** = *script label*

Appendix 5

Differences between CCS and HP-41 Calculators

You'll find CCS2's financial functions similar to those on the HP-12C, and the TI Business Analyst II.™ CCS2 bit operations are similar to those in the HP-16C. The CCS2 scripting capabilities emulate the HP-41CX. The interface and file handling procedures are pure Macintosh.

While CCS 2.0 includes features found in most the calculators mentioned above, CCS2 best captures the spirit of the Hewlett-Packard 41CX. This is largely because our users told us they enjoyed programming the 41C more than any other calculator—and they liked the idea of running all their 41C program strips on their Macintoshes.

The purpose of this Appendix is largely to help the 41C user convert HP-41C program strips to run on CCS2 calculators. We also want to help 41C users write more powerful scripts by calling attention to the many CCS2 capabilities which transcend those of the 41C. ***After reading this Appendix, we strongly suggest that you read Chapter 8, if you haven't done so already.***

Function Names

Some CCS2 function names are spelled differently than their HP-41C counterparts. For instance, because the arrow is not a standard Macintosh character, the CCS function XTO2 is equivalent to the HP function X↑2.

HP-41CX Functions not found in CCS 2.0

HP Alpha FN Name	HP Alpha Function Description Functions not included in CCS2
ALMCAT	Alarm Catalog
ALMNOW	Activates oldest past-due alarm
APPCHR	Append Alpha Register to end of current record
APPREC	Append Alpha Register as new record
ARCLREC	Append current record to Alpha Register
ASN	Assign function to User keyboard
ASROOM	Returns number of bytes free in current file
BST	Displays preceeding program line
CAT	Executes catalog n (use MEMMAP instead)
CLALMA	Clear first alarm whose message matches Alpha
CLALMX	Clear the nth alarm as specified by X-Register
CLFL	Clears named in the Alpha Register
CLKEYS	Clear all assignments on User keyboard
CLP	Clear program (use PGRM dialog instead)
CLALMS	Clear all alarms
CLRGX	Clear iith register, as specified by X-Register
CORRECT	Sets time and adjusts accuracy factor
CRFLAS	Create text file using name in Alpha Register
CRFLD	Create data file
DEC	Decimal format (use FLOAT format instead)
DEL	Deletes nn program lines (use PGRM pushbutton)
DELCHR	Deletes nn chars from record
DELREC	Deletes record
ED	Text editing mode (use TEXTON and TEXTOFF)
EMDIR	Lists directory of exteneded memory
EMDIRX	Finds nth file listed in EMDIR
EMROOM	Returns # of registers available for new file
E^X-1	Natural Exponential (arguments close to zero)
FLSIZE	Returns number of registers in file
GETKEY	Waits 10 seconds & returns key code
GETKEYX	Wait ss seconds & returns key code
GETREC	Copies record starting at ptr to Alpha Register
GETRX	Copies registers using data in X-Register
GETX	Copies current register into X-Register
GRAD	Selects Gradients angular mode
INSCHR	Inserts contents of Alpha Register

HP Alpha FN Name	HP Function Name Functions not included in CCS2
INSREC	Inserts contents of Alpha Register as new record
LN1+X	Natural Logarithm (arguments close to zero)
MOD	Y Modulo X (remainder)
OCT	Decimal to Octal (use OCTAL format instead)
OFF	Turns calculator off (use closebox instead)
ON	Selects continuous on (use menu instead)
PACK	Packs program memory (automatic in CCS2)
PASN	Assign alpha label to user keyboard
PCLPS	Clears programs (use PGRM pushbutton)
POSA	Searches Alpha Register for target in X-Register
POSFL	Searches file for target in Alpha Register
PSIZE	Allocates Memory Registers for program storage
PURFL	Purges file named in Alpha Register (use *Finder*)
RCLAF	Recalls clock accuracy factor
RCLALM	Recalls alarm parameters
RCLFLAG	Recalls status of flags 00 thru 43
RCLPT	Returns pointer value for current file
RCLPTA	Returns pointer value
RCLSW	Returns stopwatch time
REGMOVE	Copies block of registers to another block
REGSWAP	Swaps two blocks of registers
RESZFL	Resizes current file
RUNSW	Runs stop watch (use pushbutton instead)
SAVEP	Saves program in file (use PGRM pushbutton)
SAVERX	Copies some Memory Registers to a file
SAVEX	Copies X-Register to a file
SEEKPT	Sets Pointer for file using X-Register
SEEKPTA	Sets Pointer to number in X-Register
SETAF	Set clock accuracy factor
SETSW	Set stopwatch to starting time
SIZE	Allocates Memory Registers for main storage
SIZE?	Returns # of MemRegs (use MEMMAP)
STOFLAG	Returns status of flags 00 thru 43 using x
STOPSW	Stops running stop watch (use pushbutton)
SWPT	Sets stop watch store and recall pointers
T+X	Adjusts clock time by increment in X-Register
USER	Activates/Deactivates USER keyboard
XYZALM	Sets alarm time user values in Stack Registers

Comparison of Features: CCS2 & HP-41C

To remain consistent with the original 1.0 version of CCS, and the realities of the Macintosh, some terms used in Hewlett-Packard's documentation are slightly different those used in CCS2 documentation. Here's a sample:

HP terminology	*CCS2 terminology*
data register	Memory Register
program strip	program script
the display	MainLED, SLED, or Paper tape
gold shift	TOP shift
blue shift	BOT shift
program mode	PRGM mode
ED mode	TEXT mode

Stack Registers

The HP-41C/CV/CX has four standard Stack Registers: X,Y,Z,T, and the LastX-Register. The LastX-Register contains the X-operand from the last numeric function performed (with the exception of CHS).

CCS 2.0 allows from four to ten Stack Registers, plus the LastX-Register. Data can be stored and retrieved from the various Stack Registers through direct and indirect addressing. Unlike the HP-41C, CCS2 does not allow alpha data to be stored to any Stack Register. CCS2 Stack Registers are named: L (LastX), X, Y, Z, T, S5, S6, S7, S8, S9, and S10.

These names are used when storing or recalling data from the Stack Registers. On the HP-41, the names must be preceeded by a decimal point. For instance, to store the X-Register value in the Z-Register, you press the STO key, the decimal point key, a key with either X, Y, Z or T on it. In CCS2, you respond to the STO function's prompt with the Stack Register's name , or to address a register indirectly: with a Stack Register's name preceeded by the keyword IND. Upper and lowercase characters are valid.

The LastX-register contains the X-operand from the last numeric function performed, just like the HP-41. (See Appendix 1 for a list of functions that save the X-Register to the LastX-Register.) In addition, CCS2 provides the user with an additional function, STOLAST, which acts like a STO to the LastX-Register.

CCS2 also allows the X-Register to be swapped with any valid Stack Register using XSWAPR. RUP, RDN, STO, RCL, and register arithmetic is fully supported with any valid Stack Register.

Memory Registers

The HP-41CX comes standard with 319 registers in main memory. Only the first 100 data registers can be accessed directly. Any data register greater than 99 can only be accessed through indirection. With the additional 124 extended data registers, the HP-41CX provides 443 total data registers. However, this memory is shared between data, program strips (in CCS2 referred to as *program scripts*), alarms, and user keyboard information.

CCS2 can be configured with 10 to 255 Memory Registers, all of which are for the exclusive storage of alpha or numeric data. All Memory Registers can accessed either directly or indirectly. CCS2 program scripts are not stored in Memory Registers, but are allocated dynamically. This means that the size of an individual script and the number of scripts installed in any CCS2 calculator are limited only by the amount of free Macintosh RAM. HP-41CX alarm functions are not supported in CCS2—instead CCS2 has a Macintosh-style alarm function: the ALARM pushbutton. Because the user can configure the CCS calculator during construction, there is no USER keyboard mode in CCS2, and hence Memory Registers will never contain any USER keyboard data.

CCS2 does not allow for operations on blocks of Memory Registers, such as: CLRGX, REGMOVE and REGSWAP. But CCS2 does have functions to read/write a calculator's entire block of Memory Registers from/to disk files (GETR and SAVER).

As with the Stack Registers, CCS2 Memory Registers have names which are used to respond to functions which prompt for them (such as STO and RCL, for instance). Again, the keyword IND may prefix the register name in order to indirectly address another register. The valid Memory Register names are:

- **Numbers 0 through 255** depending upon how many registers the calculator has been configured with during construction.
- **Financial Register Names:** N, I, FV, PMT, PV. Both upper and lower case characters are accepted.
- **Financial Register Names:** ΣX, ΣX2, ΣY, ΣY2, ΣXY. Both upper and lower case characters are accepted.

Flags

The HP-41 control flags (11-29) and system flags (30-55) not only reflect the current status of the calculator, but by changing these flags various settings are subsequently changed.

In CCS2, only four flags can be truly be called *control* flags. They are: auto start (11), audio enable (26), and punctuation control (28-29). All other flags in the user/system flag range (11-80) only reflect current settings; changing them directly will ***not*** change the various calculator settings they reflect.

The additional flags (56-80) reflect the status of functions that are unique to CCS2. Again, these flags only reflect current settings; changing them directly will ***not*** change the various calculator settings they reflect.

CCS2 provides 12 additional user flags, because CCS2 does not support the various HP hardware control flags. See Chapter 5 and Appendix 8 for detailed information on Status Flags.

User Mode

On the HP-41CX, USER mode assigns both built-in functions and program strips to nearly any key on the HP keyboard. CCS2 accomplishes the same thing during construction via the Plug and Keyboard tools. CCS2 is essentially always in USER mode. Any function, script, or external function can be mapped to almost any key on your Macintosh keyboard.

Annunciator Display

The CCS2 annunciator display goes above and beyond the Annunciator found on the HP-41CX. In addition to the standard ALPHA, PRGM, and SHIFT annunciators, CCS2 displays the current ***Default Memory Register***, the next line number to be executed in the current script, the current input/display format, and the state of the TEXT mode (called ED on the HP-41). CCS2's Annunciator does not display the angular mode. The DEGREES/RADIANS clickstop switch displays the angular mode (note that the GRAD function is not supported). CCS2's Annunciator does not display the status of flags 00-04.

Numerical Accuracy

CCS2 is accurate to 18 significant digits as opposed to the HP-41C's 10 significant digits. If you feel that the number displayed suffers from rounding errors due to too much accuracy, switch over to the FIXED input/display format, and FIX the number of significant digits to 9 or less.

File Handling

The HP-41CX provides the capability for creating, saving, and changing text (ASCII) files, data files, and script files. CCS2 has the same capabilities, but with a distinct Macintosh flavor...

- **Data Files.** CCS2 does not support or implement pointer operations on data files. The contents of all Memory Registers in a particular CCS2 calculator can be saved to a data file with the function SAVER. Memory Registers are restored from data files using the function GETR. Data files can be read by any CCS2 calculator— not just the one which created the file.

  For example, all 50 Memory Registers from calculator "A" could be saved to data file "X." Calculator "B" with 100 Memory Registers would restore the first 50 of its registers from data file "X."

  Conversely, all 100 Memory Registers from calculator "B" could be saved to data file "Z." Calculator "A" could restore all 50 of its Memory Registers with the first 50 of registers from data file "Z."

- **Text Files.** CCS2 differs from the way the HP-41CX allocates, creates, and manipulates text files in three ways:

  1) CCS2 provides a stretchable, scrolling Paper tape where all text appears, and can be edited. Edit the tape by clicking in it, or using the function TEXTON. Stop the editing by clicking in the MainLED or by using the function TEXTOFF. (The HP-41 has the function ED).

  2) CCS2 allows the user to open a DISK file buffer and a PRINT file buffer simultaneously, to which all output from the MainLED is echoed. Another function, TPRINT, immediately prints the contents of the Paper tape.

  3) The contents of the Paper tape can be saved/read to/from text files using the CCS2 functions SAVEAS and GETAS. You will be prompted for file names via the familiar Macintosh *Save File/Get File* dialog boxes (Figures 4-34 and 4-33).

Note that text files and text file functions are only available when a Paper tape is included in a calculator. CCS2 does not support text character functions, text record functions, or text pointer functions found in the HP-41CX. CCS2 does not implement any text file search routines.

The Alpha Register and ALPHA Mode

CCS2 implements all the standard HP-41CX Alpha Register functions. Like the HP-41, CCS2's Alpha Register can hold up to 24 characters. In addition, the CCS2 function XASA appends the contents of the X-Register to the Alpha Register. Since it is possible to wire the Alpha Register to a SLED, all 24 characters can be viewed at once— so CCS2 does not scroll the Alpha Register display as the HP-41 does. All characters are written to the Paper tape while in ALPHA mode.

CCS2 allows you to type in the full Macintosh ASCII character set, including diacritics (shown in Appendix 6). In ALPHA mode, the following functions may be executed:

ADATE	ASTO	CLC	XASA
APPEND	ATIME	COPY	XTOA
ARCL	ATIME24	CUT	
AROT	CL	PASTE	
ASHF	CLA	SST	

Note that these functions must be wired to keys (or invoked from within a script) to work in ALPHA mode since the keyboard will be "unmapped" and all keyboard input ends up in the Alpha Register.

HP Strip / CCS2 Script Compatability

Any script that can run on an HP-41C can run on a CCS2 calculator with the following changes:

Alpha Data

Lines containing alpha strings should be surrounded by quotes, and bracketed by AON and AOFF. For example, the HP-41C line:

```
†X=?
```

in a CCS2 script should be written as:

```
AON
"X=?"
AOFF
```

The HP-41C allows comparison of alpha data stored in the X- and Y-Registers using the functions X=Y? and X≠Y? However, CCS2 does not allow alpha data in Stack Registers. CCS2 *does* allow comparison of the Alpha Register and any valid Memory Register using the CCS2 prompting function A=R?. When used in a script, the register number must follow the function with a space between the two. Remember that Memory Registers can contain a maximum of 6 alpha characters, while the Alpha Register can hold up to 24 characters. Comparing strings of different lengths will return FALSE.

Local and Global Labels

CCS2 does not distinguish between local and global labels the same way the HP-41C does. Refer to Chapter 8 for a detailed description of the CCS2 syntax for GTO and XEQ instructions.

Synthetic Programming

CCS2 (or Hewlett-Packard for that matter), does not support any synthetic programming instructions.

artificially made

Miscellanous Scripting Information

There is no need to PACK a CCS2 script to remove null bytes. This is done for you.

There is no need to GTO.. to place an END statement in the script; CCS2 always places an invisible END statement on the last line of the script.

There is no need to provide a global script label at the beginning of a script. You provide the script name when creating a new script via the PRGM dialog (Figure 8-2).

To examine a particular line in a script, press the PRGM pushbutton to display the PRGM dialog box (Figure 8-2). Select the script you wish to view, and click either the **Edit** or **View** buttons. The entire contents of a script will appear on the Paper tape (which must be included in a calculator if you wish to edit or view scripts).

If there is an error in your script which halts script execution, Script Error will display in the MainLED, and the number of the next line to be executed will be displayed in the Annunciator, this number will also be highlighted in the script itself when displayed in the Paper tape.

Using the instruction TALKOFF at the beginning of your script can greatly speed script execution by inhibiting the updating of all displays with the exception of the Annunciator. See Chapter 6 for more on TALKON and TALKOFF.

If building more than one calculator during a single CCS2 session, you may notice that scripts created in one calculator show up in other calculators while in Test mode. Delete any unwanted scripts by clicking the **Delete** button in the PRGM dialog box.

CCS2 scripts feature something not found in HP-41C strips: the ability to add comments anywhere in a script by inserting a semi-colon (;) before the comment. In other words, anything following a semi-colon on a script line is considered comment.

Appendix 6

Standard ASCII Values

Use this chart to convert X-Register numbers to/from Alpha Register characters.

#	Char	#	Char	#	Char	#	Char	#	Char
0		26		52	4	78	N	104	h
1		27		53	5	79	O	105	i
2		28		54	6	80	P	106	j
3		29		55	7	81	Q	107	k
4		30		56	8	82	R	108	l
5		31		57	9	83	S	109	m
6		32		58	:	84	T	110	n
7		33	!	59	;	85	U	111	o
8		34	"	60	<	86	V	112	p
9		35	#	61	=	87	W	113	q
10		36	$	62	>	88	X	114	r
11		37	%	63	?	89	Y	115	s
12		38	&	64	@	90	Z	116	t
13		39	'	65	A	91	[	117	u
14		40	(	66	B	92	\	118	v
15		41	)	67	C	93	]	119	w
16		42	*	68	D	94	^	120	x
17		43	+	69	E	95	_	121	y
18		44	,	70	F	96	`	122	z
19		45	-	71	G	97	a	123	{
20		46	.	72	H	98	b	124	|
21		47	/	73	I	99	c	125	}
22		48	0	74	J	100	d	126	~
23		49	1	75	K	101	e	127	
24	1	50	2	76	L	102	f		
25		51	3	77	M	103	g		

Macintosh Extended "ASCII " Values

These values are not part of the ASCII standard, but they are the extended values that are standard on a Macintosh. You may see different characters in these positions depending on the font with which you view them.

#	Char	#	Char	#	Char	#	Char	#	Char
128	Ä	154	ö	180	¥	206	Œ	232	Ë
129	Å	155	õ	181	µ	207	œ	233	È
130	Ç	156	ú	182	∂	208	–	234	Í
131	É	157	ù	183	Σ	209	—	235	Î
132	Ñ	158	û	184	Π	210	“	236	Ï
133	Ö	159	ü	185	π	211	”	237	Ì
134	Ü	160	†	186	∫	212	‘	238	Ó
135	á	161	°	187	ª	213	’	239	Ô
136	à	162	¢	188	º	214	÷	240	
137	â	163	£	189	Ω	215	◊	241	Ò
138	ä	164	§	190	æ	216	ÿ	242	Ú
139	ã	165	•	191	ø	217	Ÿ	243	Û
140	å	166	¶	192	¿	218	⁄	244	Ù
141	ç	167	ß	193	¡	219	¤	245	ı
142	é	168	®	194	¬	220	‹	246	ˆ
143	è	169	©	195	√	221	›	247	˜
144	ê	170	™	196	ƒ	222	fi	248	¯
145	ë	171	´	197	≈	223	fl	249	˘
146	í	172	¨	198	Δ	224	‡	250	˙
147	ì	173	≠	199	«	225	·	251	˚
148	î	174	Æ	200	»	226	‚	252	¸
149	ï	175	Ø	201	…	227	„	253	˝
150	ñ	176	∞	202		228	‰	254	˛
151	ó	177	±	203	À	229	Â	255	ˇ
152	ò	178	≤	204	Ã	230	Ê		
153	ô	179	≥	205	Õ	231	Á		

Appendix 7

Writing External CCS Code Segments

This Appendix assumes you know how to program in C, Pascal, or Assembly language. ***If you are not a programmer, this Appendix will be of little use to you.***

Overview of CCS Code Segments

This appendix explains how to write two different kinds of code segments: ***External Shapes*** and ***External Functions***.

- **External Shape** segments (resource type `'XSHP'`) are called when the calculator receives an update event. Any CCS2 part can be drawn using an external shape, but may not function properly when drawn that way. For instance, a Paper tape may not update properly if CCS leaves the drawing to your segment.

- **External Function** segments (resource type `'XCSC'`) are called when the user uses your external function's name in a script, or presses a key wired to your external function.

External shapes and functions have access to all the finished calculator's registers, flags, modes and display devices. External functions may also display their own dialog boxes.

Contrary to *Inside Macintosh*, desk accessories ***can*** be segmented— segments can be loaded by the desk accessory itself, without any help from the Segment Loader. After you have written your external segments, you install them into CCS's resource fork using *ResEdit*. Segment resources must be numbered between `0` through `31`, so *Font/DA Mover* will treat them as owned resources when installing the finished calculator desk accessory.

After your segments have been moved into CCS, they may be "attached" to calculator parts by the user during construction. External shapes are attached via the ***Select External Shape*** item on CCS's ***Graphics*** menu (Figure 4-16). Attach external functions by wiring them to keys— clicking on the key with the plug tool, then selecting ***External Functions*** from the pop-up menu in the ***Wiring*** dialog box (Figure 4-43).

When the user saves a finished calculator, your segments will be saved along with the calculator (if they were attached as described in the above paragraph). The segments will be renumbered appropriately in the saved calculator so that the *Font/DA Mover* will treat them as owned resources.

Example Source Code

The *External Shapes & FNs* folder on the *CCS 2.0* disk contains heavily commented example source code and equates files. Use this code to create `XCSC` and `XSHP` resources. We have provided examples written in:

- MDS Assembler
- MPW Assembler
- TML Pascal (MPW version)
- Lightspeed C

Appendix 8

Status Flags

CCS has 80 Status Flags. Many have the same meaning as the Status Flags found on the HP-41 series of calculators, and are included for program script compatability. Some HP flags are not supported in CCS. Unsupported "read only" flags will always test as clear. Unsupported "read-write" flags can be used as additional user flags.

There is a detailed description of each flag in Chapter 5.

READ/WRITE FLAGS		POSSIBLE SETTINGS
00-10	User controlled bit flags	(clear)
11	Automatic Script Execution	(0=N, 1=Y)
12-20	External Device Control	(not supported)
15	The "Carry bit"	(variable)
21	Printer Enable	(not supported)
22	Numeric Data Input	(0=N, 1=Y)
23	Alpha Data Input	(0=N, 1=Y)
24	Range Error Ignore	(not supported)
25	Error Ignore	(not supported)
26	Audio Enable (beep or not)	(0=N, 1=Y)
27	User Keyboard	(not supported)
28	Swap Radix & 1,000's Separator	(0=Other, 1=USA)
29	Display 1,000's Separators	(0=N, 1=Y)

READ ONLY FLAGS		POSSIBLE SETTINGS
30	Catalog Control Flag	(not supported)
31	Date Format	(0=MDY, 1=DMY)
32-35	Internal Flags	(not supported)
36-39	Number of Decimal Places	(a nibble)
40-41	Display Format	(2 bits: 0=SCI, 1=ENG, 2=FIX, 3=Other)
42-43	Angular Format	(2 bits: 0=DEG, 1=RAD ,2=GRAD)
44	Continuous ON	(always set)
45	System Data Entry	(not supported)
46	Partial Key Sequence	(not supported)
47	TOP or BOT shift	(not supported)
48	Alpha Mode Flag	(0=N, 1=Y)
49	Battery Low	(not supported)
50	Message in MainLED	(0=N, 1=Y)
51	SST flag	(not supported)
52	PGRM mode flag	(not supported)
53	I/O ready flag	(not supported)
54	PAUSE in progress	(not supported)
55	Printer Exists	(0=N, 1=Y)

CCS2 FLAGS (NOT IN HP-41C)		POSSIBLE SETTINGS
56	Alarm Set	(0=N, 1=Y)
57	Spooling to Disk	(0=N, 1=Y)
58	Line Numbers	(0=N, 1=Y)
59	Marginals	(0=N, 1=Y)
60	MUSIC On	(0=N, 1=Y)
61	StopWatch running	(0=N, 1=Y)
62	RPN/SAN	(0=SAN, 1=RPN)
63	Reserved for Future Use	(always clear)
64-71	CCS Display Format	(8 bits)

1 = INTEGER	7 = SCI	13 = HRSMIN
2 = NORMAL	8 = DEGREES	14 = TIME
3 = FIXED	9 = HEX	15 = FTINCH
4 = DOLLARS	10 = OCTAL	16 = YDFEET
5 = POUNDS	11 = BINARY	17 = ENG
6 = YEN	12 = DOZENAL	

72-79	Reserved for Future Use	(8 bits: always clear)

Appendix 9

Technical Specifications

SPECIFICATION	MIN	MAX
Number of Parts in a Calculator	0	no limit
Number of MainLEDs in a Calculator	0	1
Number of Paper tapes in a Calculator	0	1
Number of Stack Registers in a Calculator	4	10
Number of Memory Registers in a Calculator	10	255
Number of Characters in LED/SLED displays	14	24
Number of Significant Digits	n/a	18
Number of digits to right of decimal using FIX	0	9
Largest/Smallest number possible	1.0 -4932	1.0 E4932
Levels of Nested Subroutines in Scripts	n/a	6
Levels of Nested Parenthesis	0	no limit

Glossary of CCS Terms

Alpha Function a function which affects the Alpha Register.

Alpha Mode a mode in a finished calculator in which text data may be entered into the Alpha Register.

Alpha Register a memory location within a finished calculator which can hold up to 24 characters.

Annunciator a calculator part which displays several modes and settings.

Bitmap a picture made up of pixels (screen dots) which may be toggled ON/OFF individually.

CCS and ***CCS2*** abbreviations for *Calculator Construction Set*, version 2.0.

Calcshell the case in which parts are drawn in the Work window. When a work file is saved as finished calculator, the Calcshell would become the finished calculator's window.

Calculator Part a part which is added to the calculator by dragging it from the Parts palette. Note that calculator parts may not overlap each other.

Check Box a control used in dialog boxes which allows you to select more than one choice (shown above).

Clickstop switch a calculator part which controls a mode which has more than one setting.

Command Key the key on the Macintosh keyboard with the ⌘ symbol on it. This symbol is sometimes called a *cloverleaf*, a *heraldic knot*, or the *camp key* symbol. This is an international symbol for "special feature" except in Sweden where it is used on roadsigns to indicate that a camping area is ahead.

Default Memory Register the Memory Register on which an operation will be performed if no register is specified.

Display format controls the way data is displayed.

Double-click clicking the mouse button twice in rapid succession; a powerful shortcut to selecting-then-clicking.

FN an abbreviation for *function*.

Fatbits a drawing environment which is 8 times normal size.

Financial Registers a block of Memory Registers used to hold financial data (used by financial functions).

Finder an application used to copy, delete, and rename files. The Finder draws a trash can icon in the bottom right corner of the screen.

Finished Calculator a calculator which has been saved as either a desk accessory or an application. A calculator used in Test mode is functionally equivalent to a finished calculator.

Function Name a name used to refer to a function. Many functions may be executed by typing their names after pressing the XEQ key. In program scripts, function names are used to refer to functions.

GetFile a dialog box that displays a list of files to be opened.

Grab bar the section of the Calcshell that would be a title bar on a window. Use the grab bar to move the Calcshell.

Graphic Part a part which is added to the calculator by creating it with tools from the Tools palette, or by pasting text and graphics into the Work window using the **Edit** menu. Graphic parts are always drawn behind calculator parts, and may overlap each other.

Grayed out refers to an inactive menu item or dialog button item. Usually, selecting something will activate these items.

Input /Display Format is the display format in which the calculator performs its operations.

Keycap the section of a calculator part which contains text or graphics— used to indicate the function wired to the part.

Keystroke the character generated by the act of typing a key on the Macintosh keyboard.

LED a calculator part which displays data, text or messages. The term *L.E.D.* is an abbreviation for *Light Emitting Diode*; the device used as displays on the first handheld calculators.

LastX-Register a Stack Register to which most CCS functions copy X to just before the functions execute.

Locked Script a permanent script that's been locked via the *Wiring* dialog. Locked scripst can't be edited, viewed, or deleted.

Magic Hand a tool used to select, drag, and stretch parts in the Work window.

MainLED the special LED part which usually displays the contents of the X-Register.

Mapping the act of assigning a keystroke to a calculator part.

Palette a kind of window that "floats" above the Work window. See Chapter 4 for details.

Paper tape a calculator part which displays editable text.

Part a single object in the calculator. See also *calculator part* and *graphic part*.

Parts palette the window containing unwired calculator parts.

Permanent Script a script that has been wired to a key. Permanent scripts are saved with the calculator.

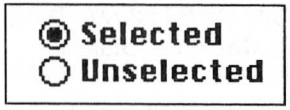
PGRM record mode a mode in finished calculators where program scripts may be edited.

Pressing means either *clicking* a calculator part with the mouse, or *typing* a keystroke which has been mapped to a part.

Program Script a series of functions which may be permanently saved/read to/from a disk file, and repeated.

Pushbutton switch a calculator part which controls a mode which has only *ON/OFF* settings.

RPN *Reverse Polish Notation;* an input mode where data is first moved into Stack Registers, then operations are keyed-in.



Radio Button a control used in dialog boxes which allows you to select only one of several choices (shown above).

Rubberband an object which draws around the Calcshell to allow stretching of the Calcshell.

SAN *Standard Algebraic Notation;* an input mode where data is entered in the order it would be written as an equation.

SLED *Supplemental LED;* a variation on the LED used to display data other than the contents of the X-Register.

SaveFile a dialog box that allows you to name a new file, and select the disk and folder in which the new file will reside.

Script see *program script*.

Stack Registers a block of memory (seperate from memory registers) used to hold and manipulate data by functions.

Standard LED a MainLED which always displays its contents using the 9 point *Monaco* font.

Statistical Registers a block of Memory Registers used to hold statistical data (used by Statistics functions).

Tall LED a MainLED which displays its contents using a font size and style which you choose.

Test mode a mode in the CCS application which allows use of the calculator-under-construction as though it were a finished, installed calculator.

TEXT mode a mode in finished calculators where the text on the Paper tape may be edited.

Tools palette the CCS window containing tools and selectors.

Transient Script a script that has not been wired to a key.

Wiring the act of assigning a function to a calculator part.

Work file a CCS document file containing a calculator which can be edited by CCS.

Work Window the window which displays the calculator being constructed. Calculators displayed in this window are treated as a collection of parts that may be manipulated, not as functions wihch perform calculations.

XEQ abbreviation for *execute*. There is a CCS function called XEQ which allows you to enter a function name, script name, or script label to execute.

X-Register a Stack Register used by most functions as a work area. You key values into the X-Register. Functions usually return results in the X-Register.

Y-Register a Stack Register used by many two-value functions as a work area. You first key a value into the X-Register, then use the ENTER function to place the value in the Y-Register.

Bibliography

Adams, Thomas. *Programming The HP-41C/CV/CX*. 2nd ed. Photocopy. 1989.

Apple Computer. *Macintosh™ Utilities User's Guide*. Cupertino: Apple Computer, Inc., 1987.

Apple Computer. *Macintosh Plus Owner's Guide*. Cupertino: Apple Computer, Inc., 1987.

Apple Computer. *Macintosh SE*. Cupertino: Apple Computer, Inc., 1987.

Apple Computer. *Macintosh II*. Cupertino: Apple Computer, Inc., 1986.

Apple Computer. *Macintosh System Software User's Guide*, Version 6.0. Cupertino: Apple Computer, Inc., 1988.

Apple Computer. *LaserWriter and LaserWriter Plus*. Cupertino: Apple Computer, Inc., 1986.

Apple Computer. *Imagewriter User's Manual*, Part I: Reference. Cupertino: Apple Computer, Inc., 1984.

Ayres, Frank, Jr. *Mathematics of Finance*. (Schaum's Outline Series) New York: McGraw-Hill, 1963.

Beyer, William, H.,ed. *CRC Standard Mathematical Tables*. 27th ed. Boca Raton: CRC Press, 1984.

Dodin, Jean-Daniel. *Inside The HP-41*. Trans. Mary-Denise Dodin and John Vandenabbeele. Ed. Wilson Holes. Berkeley: Synthetix, 1985.

EduCALC. *Portable Computing Catalog #44*. Laguna Nigel: EduCALC, Inc., 1989.

Greynolds, Elbert B.,Jr. *BA•II Executive Caculator Guidebook.* Dallas: Texas Instruments, Inc., 1986.

Greynolds, Elbert B., Julius Aronofsky, and Robert Frame. *Financial Analysis Using Calculators: Time Value of Money.* New York: McGraw-Hill, 1980.

Greynolds, Elbert B., and Julius Aronofsky. *Practical Real Estate Financial Analysis: Using The HP-12C Calculator.* Chicago: Real Estate Education Company, 1983.

Hewlett-Packard. *Vol. 2: Operation In Detail. HP-41CX Owner's Manual.* Singapore: Hewlett-Packard Company, 1983.

Hewlett-Packard. *HP-12C Owner's Handbook and Problem Solving Guide.* Corvallis: Hewlett-Packard Company, 1982.

Hewlett-Packard. *HP-16C Computer Scientist Owner's Handbook.* Corvallis: Hewlett-Packard Company, 1984.

Hewlett-Packard. *The HP-41C/41CV Alphanumeric Full Performance Programmable Calculator, Owner's Handbook and Programming Guide.* Corvallis: Hewlett-Packard Company, 1980.

Jarett, Keith. HP-41 *Extended Functions Made Easy.* Berkeley: Synthetix, 1983.

Jarett, Keith. HP-41 *Synthetic Programming Made Easy.* Berkeley: Synthetix, 1982.

Kazmier, Leonard, J. *Business Statistics.* (Schaum's Outline Series) New York: McGraw-Hill, 1976.

McCornack, Alan, and Keith Jarett. *HP-41 Advanced Programming Tips.* Berkeley: Synthetix, 1987.

Mier-Jedrzejowicz, W.A.C. *Extend Your HP-41.* Berkeley: Synthetix, 1985.

McCarty, George. *Calculator Calculus.* Long Beach: EduCALC, 1976.

Sharp Electronics. *Applications In Finance: Instructions and Applications For The Sharp EL-533 Financial Calculator.* Osaka: Sharp Electronics Corporation, 1985.

Wadman, Ted, and Chris Coffin. *An Easy Course in Programming The HP-41.* Corvallis: Grapevine, 1983.

Wickes, W. C. *Synthetic Programming On The HP-41C.* Corvallis: Larken, March, 1982.

Index

Control Number 6-29
IIIII. 55Fcc

E

F

G

S

T