CONTENTS

1  INTRODUCTION
   WHAT IS 41UCC?

2  GETTING STARTED
   MAKING A BACKUP COPY  1
   UNDERSTANDING CP/M  3
   HOW TO USE 41UCC  4
   A REAL PROGRAM - LOWPASS  10
   LISTING OF LOWPASS  17
   ANOTHER EXAMPLE PROGRAM - SECANT  21
   INTERACTIVE MODE  32
   INDIRECT COMMAND MODE  32
   LISTING OF SECANT  36

3  INTRODUCTION TO SPECIAL FEATURES
   DEFINE BYTE  41
   END  41
   EQUATE  41
   STRING EQUATES  42
   EXPRESSIONS  44
   GLOBAL LABELS  46
   #INCLUDE  46
   KEY ASSIGNMENTS  46
   PAGE  47
   SET  47
   TITLE  47

4  41UCC COMMAND LINE PARAMETERS  49

APPENDIX A
   INSTRUCTIONS THAT DIFFER FROM THE HP-41  52

APPENDIX B
   SUMMARY OF ERROR MESSAGES  53

APPENDIX C
   SYNTHETIC INSTRUCTIONS  56

APPENDIX D
   PRP LISTINGS AND 41UCC  58

APPENDIX E
   PRINTING BARCODES  60

APPENDIX F
   MODIFYING 41UCC  63

APPENDIX G
   LISTINGS AND BARCODE FOR EXAMPLES  65
INTRODUCTION

WHAT IS 41UCC?

41UCC is a "User-Code cross Compiler". The "user-code" part means that it accepts normal, everyday programs just like you already write for your HP-41C/CV. The "cross" part means that it does not run on the HP-41 - it runs on any 48K or larger 8080/8085/Z80 CP/M 2.2 system. The "compiler" part means that 41UCC takes the programs that you have written and compiles them into the binary codes that the HP-41 understands.

WHAT WILL IT DO FOR ME?

Simply, it will allow you to write programs for your HP-41 in a fraction of the time previously required. Also, it will make documentation and modification of your programs a much simpler task.

HOW DOES IT DO THAT?

Your work is made easier in several ways:
(1) You can write your program using your favorite text editor.
(2) You can add comments anywhere you like - to improve program documentation.
(3) You can use meaningful names to refer to registers.
(4) You can make changes more easily - no more going through and changing every reference to register 01 to register 02 if you need to make a change - you need only to make one change at the start of the program.
(5) Symbolic expressions!
(6) And much more!

WILL IT ACCEPT SYNTHETIC CODES?

Yes, of course - if you want to use them.

WHERE CAN I GET IT?

From Hand Held Products Inc., 6201 Fair Valley Drive, Charlotte, N.C. 28211. 41UCC is available from stock. When ordering, specify 8" CP/M, IBM-PC w/Z-80 card, or Osborne 5 1/4" formats. Other formats (such as Heath/Zenith 5 1/4", Apple II w/Z-80 card, Avatar, Televideo, Xerox/Kaypro 5 1/4", or Superbrain) are available on special request. Please allow an extra two weeks for delivery if you request a special format. 41UCC requires an 8080/8085/Z-80 or similar CP/M system, 48K of memory, and at least one disk drive to run. More memory and two disk drives are recommended.

Copyright 1982 by Leslie Brooks.
This section will explain how to make a backup disk, what 41UCC is

*I am a novice — how do I use 41UCC?*

It is important to understand at least the basics of CP/M in order to effectively use 41UCC. In particular, you should know how to create a file with a text editor (such as CP/M's ED), how to get a directory (a listing of all the files on the disk), and how to make a backup copy of files or disks (with CP/M's command PIP). If you do not know how to do these things, a good book to start with is the *CP/M PRIMER* by Stephen Murtha and Mitchele Waite, published by Howard W. Sams & Co. Another good choice would be *USING CP/M - A Self-Teaching Guide* by Judi Fernandez and Ruth Ashley. This one is published by Wiley.

This entire manual also assumes that you know how to program an HP-41C. It is not necessary that you know synthetic programming, nor is it even helpful (unless your application requires it). Knowledge of any assembly language will be an asset in using 41UCC.

**THE FIRST STEP**

The first step in using 41UCC is to **MAKE A BACKUP COPY**. Should the power fail while you are using your working disk, or should your dog fetch it for you, or a child smear a banana into it, you will be very glad of a safe original disk sitting on the shelf. To make this backup, you need to put a freshly formatted (initialized) disk into drive B of your machine and your CP/M system disk in drive A. If you don't know how to format a disk, look in your system manual under FORMATTING or INITIALIZING A DISK. The example it gives should look something like this

```
A> format
FORMAT Version 1.5

Drive A or B? b
(S)ingle or (D)ouble Density? d

Now formatting drive B double density.
Formatting done.

A>
```

Here a few notes are in order - the 'A>' is CP/M's prompt, and the rest is what you typed. I will always put your entries in **bold face** so that you can distinguish them from the things the computer types. I will always assume (unless otherwise noted) that you hit the RETURN or ENTER key at the end of any line you type. This tells the computer that you are through with the line and it can now process it - in general, it ignores the command line until you press the RETURN or ENTER key. If I need to explicitly show that you hit the RETURN/ENTER key, I will use the '<CR>' symbol (RETURN is short for Carriage Return).
GETTING STARTED

Now we will copy PIP (a file copying program) and a system image (a copy of CP/M) onto it. To copy PIP to the new disk, type

A> pip b:=a:pip.com

If you get the response

PIP?

you do not have PIP on the disk, and need to get a disk which does have it.

Now you have told PIP to send a copy of itself to drive B; you need only to copy CP/M to your new disk in drive B and then we can start using it. This is not quite the same as copying a file with PIP, because CP/M is not a file - so Digital Research gave us a special program called SYSGEN to GENerate a new SYStem image. Running it involves typing its name, and then telling it to get the system (CP/M) from drive A and put it on drive B. It looks like this:

A> sysgen
SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)a
SOURCE ON A, THEN TYPE RETURN<CR>
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)b
DESTINATION ON B, THEN TYPE RETURN<CR>
FUNCTION COMPLETE
A>

Now in drive B we have a fresh disk with CP/M and PIP on it. Put this disk in drive A and type a control-c (hold down the CONTROL or CTRL key and press C). A common notation for control functions is the 'C' symbol. This symbol followed by a character means to hold down the control key and press the character. Thus ^C means to hold down the control key and press the 'C' key. There will be a slight pause followed by CP/M's prompt.

A>

If you do not get CP/M's prompt again you have an error. It could mean that you did not do the SYSGEN properly, or it could mean that you have a bad disk or a bad copy of SYSGEN. Try again until it works.

Your fresh disk is now in drive A. To make it into a usable disk with 41UCC on it (in addition to CP/M and PIP), place your 41UCC disk in drive B, and copy everything on it to drive A:

A> pip a:=b:*.*

You now have a backup copy of 41UCC. Put the original 41UCC disk in a safe place and use the copy for all of your work. If
GETTING STARTED

you damage the copy, you won't lose a week of work while I send you a new disk.

I HAVE MY BACKUP - WHAT NOW?

Now, it would be helpful if you understood a little bit of what 41UCC is intended to do, and how it interacts with programs such as ED, RDS and PBAR before you actually start using it.

We can't really understand how these programs relate to each other without understanding a bit about CP/M. Okay, so what is CP/M? Well, CP/M is just a program which allows you to do useful things on your computer. Let's look at an analogy. What does your HP-41C do if you push XEQ ALPHA "SIZE" ALPHA? It prompts you for the SIZE you want, right? But how did it 'know' that it should do that? The only reason it works that way is because there is a program running in it whenever it is on - but you never 'see' this program, you just see the results. The only reason it works the way it does is because HP programmed it that way. However, you don't have to be an expert on the intricacies of this program in order to use the calculator - you just push the right button and it works. CP/M can be thought of as being the program that runs the calculator. You don't have to understand all of it in order to be able to use it. Now, how did you know that your calculator would respond properly when you tried to set the size? Well, there was a number in the display (i.e. the calculator was turned on) and the PRGM enunciator was not turned on (the calculator was not running a program). In the same way, we can give CP/M a command whenever we see

A>

or

B>

This means that CP/M is ready to accept a command. If we do not see this prompt, or if we see a different prompt (such as * or ?) then some other program is running and we cannot use CP/M commands. If the PRGM was showing on your calculator you would not expect to be able to execute SIZE - programs do not understand things like SIZE or CATalog 1.

Speaking of CATalog 1, how do you find out what programs are on a disk? CP/M does have a command that corresponds to CAT 1; it is called DIR (DIREctory). A CAT 1 catalogs all of the programs that are in memory and ready to run; in the same way a DIR catalogs all of the programs that are on disk.

A>DIR (hit a Carriage Return after the R)
A: TST28 UCC : HEX UCC
A>

Now you can see that you have the files 41UCC.COM, RDS.COM,
FILTER.COM, SECANT.UCC, and others on your disk. (The file names are all given as eight letters plus the three letter type, and the dot in the name is not shown.) So now we have our CATalog 1, but there is a difference - all of the names you see in a CATalog 1 listing are programs which can be run, but not all of the names you see in a DIRectory listing can be run. Just as you can have data files in extended memory, CP/M allows data files on disk. Programs - anything that can be run - always have a name that ends in .COM (for COMmand). 41UCC's complete name is 41UCC.COM - but we rarely have to use the .COM part. If you push XEQ ALPHA "SIZE" ALPHA on your HP-41C you don't have to specify that SIZE is an executable (.COM in CP/M) program - it wouldn't make sense to try to execute anything else. In the same way, in CP/M you don't have to specify the .COM part in order to run a program - you just type the programs' name. If you wanted to run 41UCC, you would just type

A>41ucc

Notice that you did not type the 'A>' - CP/M did that. Also, just like you have to hit ALPHA at the end of a program name on the HP-41C, so you had to hit a RETURN (or CR or Carriage Return on some keyboards.) This tells CP/M that you have reached the end of the name - just as hitting ALPHA tells the HP-41C that you have reached the end of the program name.

Finally, if you push XEQ ALPHA "FOO" ALPHA you know that you should have a program called FOO in memory. If you do not you will get NONEXISTENT. If you did this to CP/M

A>FOO

CP/M would look on disk for the program FOO.COM. If the program did not exist, you would get

FOO?

which is CP/M's way of saying "FOO is not a command that I understand myself, and I can't find it on disk either."

So now you know what CP/M is, how to get a CATalog 1 listing out of it, how to XEQ a program, and what CP/M's version of NONEXISTENT looks like. What else do you need to know in order to effectively use 41UCC? The most important thing you need to know is exactly what you intend to accomplish.

YOUR GOALS

At this point you should have one or more of three goals. Take a look at figure 1 and think about which of these goals you have:

1) You have an existing program, on an HP-41C, that you wish to burn unchanged into EPROMs. This is path 1 in figure 1, and involves only RDS. You do not need 41UCC to accomplish this.
2) You have an existing program on an HP-41C and you would like to document it and/or make some changes to it before burning EPROMs. This is path 2 in figure 1, and involves 41UCC, ED or some other text editor, and a program called FILTER.COM. (FILTER.COM is used to convert an HP-41C PRP listing to a format that 41UCC can understand. When you need to do this, look in appendix D).

3) You would like to create an HP-41C program from scratch on your microcomputer, and download it into the HP-41C for testing. For this you will need a text editor (such as ED) and 41UCC. For downloading the program to the HP-41C you can go through RDS and burn EPROMs or you can print barcodes on a suitable printer. For information on burning EPROMs consult the RDS documentation. For information on printing barcodes see appendix E.

Now look at figure 1 again, with your goal in mind. Notice that there are two ways to get a program into RDS - you can upload it from the HP-41C using the HP-IL, or you can produce it through 41UCC. For getting programs back down to the HP-41C, you can go through RDS and burn EPROMS or you can go through PBAR and print barcode. Notice also that 41UCC has to have an input file, which you create with a text editor (or FILTER.COM), and 41UCC in turn produces three output files. The .LST or LiST file is human readable and contains a great deal of useful information about your program (including a cross reference of all of the flags, registers, and labels you have used). The .BIN or BINary file is used as input to RDS and cannot be printed. The .WND or WaND file contains barcode information which could be transferred to someone else or printed on your printer (such as an MX-80, MX-100, Trilog, Printronix, or daisywheel). You can also send the WaND file to your printer or screen and look at it. If you send it to your screen some of the letters may flash or look strange - this is normal.

Using 41UCC involves only two (hopefully) very easy steps. First take your favorite text editor and type in your HP-41C program, then save it to disk. This creates a file on disk which will be used as input to 41UCC. The second step will be to use 41UCC to produce all of the output files discussed above.

THE FIRST STEP

Let's pretend that your program looks something like this:

```
LBL 'TEST' ;MY FIRST TEST PROGRAM
BEEP ;TELL ME THAT IT RAN
END ;BUT DON'T DO MUCH ELSE
```

Now, admittedly, this is a very simple program, but it is a good start. Notice first of all that the label in your test program is in quotes. This will be true of all alpha labels in any program to be used with 41UCC. Secondly, notice the comments
Figure 1
GETTING STARTED

in the program - these are one of the prime advantages of 41UCC over programming on an HP-41C. Comments are preceded by a semi-colon, and may go anywhere in the program. I put the 'LBL' to the left of the other commands so that labels are easy to spot, but this is not required. In fact, there are no limitations on the format of lines - commands and labels may go in any column, and you may use spaces or tabs anywhere you like. If you wish, you may indent loops (a la Pascal) in order to make them more obvious. Finally, you should assure yourself that this really is a normal HP-41C program just like many that you have written. 41UCC supports many other enhancements (listed in alphabetical order in the next section "INTRODUCTION TO SPECIAL FEATURES"), but for now we do not need to worry about them. There are a few 41UCC instructions which do not look like their HP-41C counterparts; these are all listed in Appendix A.

Now we need to type in this program and save it as a file on disk so that 41UCC can work on it. Assuming that we use the CP/M text editor ED, typing in our program will go something like:

A> ed test.ucc
NEW FILE
: *I
  1: LBL 'TEST' ;MY FIRST TEST PROGRAM
  2: BEEP ;TELL ME THAT IT RAN
  3: END ;BUT DON'T DO MUCH ELSE
  4: ^Z
: *E
A>

Again, everything you typed is in bold face; everything the computer produced is in normal face. The '^Z' on line four means that you held down the CONTROL key and pressed the 'Z' key. This tells ED that you want to get out of insert mode. The 'E' on the following line means that you want to end your editing. ED will return to CP/M after saving what you typed in as the file TEST.UCC. If you do not know how to use your text editor please stop now and learn it. If you are using WordStar you should use it in non-document mode.

THE SECOND STEP

After you exit from ED your program will exist on disk as TEST.UCC. You will now want to run 41UCC on it; that looks like this:

A> 41UCC I=TEST.UCC
41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS.
DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED.
VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO
A>
GETTING STARTED

Had there been any errors in your program, they would have shown up here. The "I=" in the command line tells 41UCC what file to process. 41UCC has now read in your source file (TEST.UCC), checked it for errors, and compiled it to produce the files TEST.LST (the Listing file), TEST.BIN (Binary file), and TEST.WND (Wnd or barcode file). Take a look back at Figure 1 if you need a mental picture of what is happening at this step. The section titled 41UCC COMMAND LINE PARAMETERS explains how to turn off the generation of the Wnd or Binary files, should you not want them. Ready for something a bit more complex? Suppose we modify our test program so that it looks like this:

LBL 'TEST1' ; MODIFYING MY TEST PROGRAM A BIT
T 'HELLO' ; A TEXT STRING
;
; THIS IS TO BE EXECUTED THE FIRST TIME THE PROGRAM RUNS
;
APPEND 'WORLD' ; ADD THIS TO INCLUDE EVERYONE
AVIEW
PSE
XEQ 'TUNE' ; PLAY SOME MUSIC
END

To do that, we will need to use ED again.

A>ed test.ucc
: *OA
1: *STEST^ZTEST1^Z
1: *SMY FIRST TEST PROGRAM^ZMODIFYING MY TEST PROGRAM A BIT^Z
1: *LI
2: T 'HELLO' ; A TEXT STRING
3: 
4: ; THIS IS TO BE EXECUTED THE FIRST TIME THE PROGRAM RUNS
5: 
6: APPEND 'WORLD' ; ADD THIS TO INCLUDE EVERYONE
7: AVIEW
8: PSE
9: XEQ 'TUNE' ; PLAY SOME MUSIC
10: ^Z
11: *K
: *B#T
1: LBL 'TEST1' ; MODIFYING MY TEST PROGRAM A BIT
2: T 'HELLO' ; A TEXT STRING
3: 
4: ; THIS IS TO BE EXECUTED THE FIRST TIME THE PROGRAM RUNS
5: 
6: APPEND 'WORLD' ; ADD THIS TO INCLUDE EVERYONE
7: AVIEW
8: PSE
9: XEQ 'TUNE' ; PLAY SOME MUSIC
10: BEEP ; TELL ME THAT IT RAN
1: *E

This example gives you a few more things of interest such as
1) text strings are preceded by a "T"
2) having no proof reader's append mark I used "APPEND" for this function.

Let's compile this new file and see what we get. Before we do though, you should notice that this time I will refer to the file just as 'TEST' - not as 'TEST.UCC'. The '.UCC' is optional and 41UCC will assume that you mean it even if you leave it off.

A>41UCC I=TEST
41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS. DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED.
VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

O ERROR(S) IN PHASE ONE
O ERROR(S) IN PASS ONE
O ERROR(S) IN PASS TWO

THERE WERE REFERENCES TO ALPHA LABELS NOT DEFINED IN THIS PROGRAM.
IF THESE LABELS ARE NOT IN ANOTHER PROGRAM, YOU HAVE AN ERROR.
CHECK THE CROSS REFERENCE IN THE .LST FILE FOR DETAILS.

A>

In our modification of the test program we had an error - the label 'TUNE' was not defined, so 41UCC warned us about this. This is only a warning, it is not a fatal error. If you print a copy of TEST.LST, you will discover in the cross reference (at the end of your program) a page that looks like this:

<table>
<thead>
<tr>
<th>UNDEFINED ALPHA LABELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)</td>
</tr>
<tr>
<td>LABEL</td>
</tr>
<tr>
<td>NAME ON</td>
</tr>
<tr>
<td>TUNE</td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:

<table>
<thead>
<tr>
<th>MEANING</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOTO</td>
<td>G</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>X</td>
</tr>
</tbody>
</table>

This means that you referred to an alpha label TUNE on line 10 of your program, but you did not define the label anywhere in your program. If in fact you do have a label 'TUNE' in some other program currently in your HP-41C, then you can safely ignore this warning and continue. If you do not have a label 'TUNE' in any of the programs in your HP-41C, then attempting to run the program 'TEST1' that we just compiled will produce a 'NONEXISTENT' error message. All error messages and their meanings are listed in Appendix B.

Now suppose I were going to burn an EPROM with my program in it. I don't need the WaND file to burn an EPROM, and I don't like wasting space on disk for it, so I use

A>41UCC I=TEST,L=LST:,NW
GETTING STARTED

which tells 41UCC to compile the file "TEST.UCC", send the listing directly to the printer, and produce no wand file at all. These parameters (or options) may be specified in any order. A complete listing of command line parameters is given in the section titled 41UCC COMMAND LINE PARAMETERS.
Now let's look at a real problem, and develop a real program to solve it. Suppose we want to design a low-pass filter (for a CB antenna filter or for a stereo bypass). We could go to a handbook such as the "ARRL Amateur Radio Handbook", or "Basic Computer Programs in Science and Engineering" and get a formula for this type of filter. Page 196 of the Basic book shows us a schematic of a simple filter, and we can see from the formulas that we will need to specify

1) the terminating resistance (52 ohms for a CB, 8 ohms for a stereo)
2) the cutoff frequency of the filter

The filter contains one coil and two capacitors. The formulas for their values are

coil - \( \frac{R}{\pi F} \) [terminating resistance divided by the cutoff frequency times PI]

capacitor - \( \frac{1}{2\pi RF} \) [1 over 2 times PI times the terminating resistance times the cutoff frequency]

Now from this information we can write a program to prompt for input and produce the proper output. It would look something like this:

```
LBL 'LOWPASS' ;OUR LOW-PASS FILTER PROGRAM
T 'FREQUENCY=?' ;ASK FOR THE CUTOFF FREQUENCY
PROMPT
STO 00 ;SAVE IT
T 'R(TERM)=?' ;ASK FOR THE TERMINATING RESISTANCE
PROMPT
STO 01 ;SAVE THE RESISTANCE

;CALCULATE THE INDUCTOR (COIL) FIRST
; L = \( \frac{R}{\pi F} \)

RCL 00 ;GET THE FREQUENCY
PI
* ;PI * FREQUENCY
/
;DIVIDE INTO THE RESISTANCE
STO 02 ;SAVE IT FOR FUTURE USE

;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
T 'L=
ARCL X
AVIEW
PSE

;NOW CALCULATE THE CAPACITOR
; C = \( \frac{1}{2\pi RF} \)

RCL 01 ;GET THE RESISTANCE AGAIN
RCL 00 ;AND THE FREQUENCY
*
PI
*
STO+ X \;\text{DOUBLE IT}
STO 03 \;\text{SAVE FOR FUTURE USE}

;NOW DISPLAY THE CAPACITOR VALUE
T 'C= '
ARCL X
AVIEW
PSE
END

You can see from this that a 41UCC program really does look very much like a normal HP-41C program. The two most obvious differences in this example are the T that precedes a text line and the fact that comments can go anywhere in the program. There are a few 41UCC instructions which do not look like their HP-41C counterpart; these are all listed in Appendix B - 'Instructions Which Differ From the HP-41C.'

If you only use this filter program once a month or so, this version may be adequate, but suppose you use it very often, and also use HP's circuit analysis module. You will quickly get tired of having the module write over your stored values of capacitance and inductance for the filter. Now you would like to move the registers used by this program out of the way - say to 50-53. If the filter program were very long you would get very tired of looking for every occurrence of '0' and changing it to '50'. 41UCC has provided a way around this - we can give a register a name and then refer to it by name. Since we normally put the definitions of the names at the beginning of the program, we have only one place to look to change which registers we are using. Here is our filter program converted to use names for the registers.

;LOW-PASS FILTER PROGRAM
;WRITTEN BY LESLIE BROOKS
;NOVEMBER 17, 1982.

;REGISTER EQUATES
EQU FREQUENCY 00 \;\text{USE REGISTER 0 FOR THE FREQUENCY}
EQU RESISTANCE 01 \;\text{USE REGISTER 1 FOR THE RESISTANCE}
EQU CAPACITOR 02 \;\text{CAPACITOR VALUE}
EQU INDUCTOR 03 \;\text{COIL VALUE}

LBL 'LOWPASS' ;OUR LOW-PASS FILTER PROGRAM
T 'FREQUENCY=?' ;ASK FOR THE CUTOFF FREQUENCY
PROMPT
STO FREQUENCY ;SAVE IT
T 'R(TERM)=?' ;ASK FOR THE TERMINATING RESISTANCE
PROMPT
STO RESISTANCE ;SAVE THE RESISTANCE

;CALCULATE THE INDUCTOR (COIL) FIRST
; \quad L = R/(\pi \times \text{FREQUENCY})
A REAL PROGRAM

; RCL FREQUENCY ;GET THE FREQUENCY
PI * ;PI * FREQUENCY
/ ;DIVIDE INTO THE RESISTANCE
STO INDUCTOR ;SAVE IT FOR FUTURE USE
;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
T 'L=' ARCL X AVIEW PSE

;NOW CALCULATE THE CAPACITOR
; C = 1/(2 * PI * RESISTANCE * FREQUENCY)
RCL RESISTANCE ;GET THE RESISTANCE AGAIN
* ;AND THE FREQUENCY
PI *
STO+ X ;DOUBLE IT
STO CAPACITOR ;SAVE FOR FUTURE USE
;NOW DISPLAY THE CAPACITOR VALUE
T 'C=' ARCL X AVIEW PSE END

Again, it looks pretty much like a standard HP-41C program - except for calling registers by names. 41UCC will convert these names to the proper register numbers for the HP-41C. If you looked at this program on your calculator you would see the correct register numbers in place of the names - but you could put your 41UCC listing beside the calculator and see the names.

This is better than the first program, but we still have to change four lines in order move the registers we are using to 50-53. The four lines

EQU FREQUENCY 00 ;USE REGISTER 0 FOR THE FREQUENCY
EQU RESISTANCE 01 ;USE REGISTER 1 FOR THE RESISTANCE
EQU CAPACITOR 02 ;CAPACITOR VALUE
EQU INDUCTOR 03 ;COIL VALUE

would have to be changed to

EQU FREQUENCY 50 ;USE REGISTER 0 FOR THE FREQUENCY
EQU RESISTANCE 51 ;USE REGISTER 1 FOR THE RESISTANCE
EQU CAPACITOR 52 ;CAPACITOR VALUE
EQU INDUCTOR 53 ;COIL VALUE

and the rest of the program would be unchanged.

This isn't too difficult, but could it be easier? Suppose
that there were forty or fifty registers involved rather than just four? I wouldn't want to have to change forty or fifty register numbers! There is in fact an easier way to do this - we would change the same four lines to look like this

```
EQU BASE 50 ; USE 50 FOR THE BASE REGISTER
EQU FREQUENCY BASE+0 ; THE CUTOFF FREQUENCY
EQU RESISTANCE BASE+1 ; THE TERMINATING RESISTANCE
EQU CAPACITOR BASE+2 ; CAPACITOR VALUE
EQU INDUCTOR BASE+3 ; COIL VALUE
```

and we added a new line to define the base register. **Now** to change the registers we are using we only need to change one line! This is a big improvement over the original program where we had to go through every line making changes in order to change the register assignments. It is also much more readable than the original program - we don't have to remember what went in register 1 - we just save a resistance in RESISTANCE and recall it in exactly the same way.

Now, before you get too excited and run off naming every register in sight, you should remember that 41UCC only looks at the first seven letters in each name. If we tried to define a register (in the filter program) with the name RESISTABLE we would get an error when we ran 41UCC. The reason for this is that 41UCC would not be able to tell the difference between RESISTANCE and RESISTABLE, and would complain about it. 41UCC does not treat upper and lower case differently here, so either one would produce the same result. Also, you can't put any character you can think of in a name - just letters, numbers, dollar signs '$', and underlines '_'. One person who will remain nameless tried to use a name that was something like BASE-PAGE. It worked fine until he put a

```
STO BASE-PAGE
```

in his program and 41UCC tried to subtract PAGE from BASE to see what register he was using! NOT what the nice man had in mind, but exactly the sort of thing you will get if you try putting funny characters in register names. (Please remember that this is not true for alpha labels - 41UCC will accept anything for them.)

You should also be aware that the symbols R1, R2, R3, and R4 are special symbols and belong to 41UCC. You should not try to create your own symbols by these names. 41UCC uses them like this

```
A>41UCC I=LOWPASS,R1=50
```

41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS. DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED.
VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

0 ERROR(S) IN PHASE ONE
41UCC accepted the value of R1 as a parameter on the command line, and passed it to the program. If we modified LOWPASS to look like

```assembly
EQU BASE R1 ;USE R1 FOR THE BASE REGISTER
EQU FREQUENCY BASE+0 ;THE CUTOFF FREQUENCY
EQU RESISTANCE BASE+1 ;THE TERMINATING RESISTANCE
EQU CAPACITOR BASE+2 ;CAPACITOR VALUE
EQU INDUCTOR BASE+3 ;COIL VALUE
```

then we could change the registers LOWPASS uses simply by re-compiling the program with a new value for R1. There would be no need to go in and edit the program. If we don't give R1 a value on the command line it will get the default value of zero.

However, enough on register names, and let's get back to the program. The values that we are producing are in Henries and Farads - not common units of measure. Most people would be much happier if we divided the inductor value by 1000 to produce millihenries, and the capacitor by 1 million to produce microfarads. This change is very easy to make in our program - just put in the division right before storing and displaying the values. We would also want to label them, so (for the inductor) we get something like this:

```assembly
/ ;DIVIDE INTO THE RESISTANCE
1000 ;CONVERT TO MILLIHENRIES
/
STO INDUCTOR ;SAVE IT FOR FUTURE USE
;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
T 'L= '
ARCL X
APPEND ' MH' ;UNITS ARE MILLIHENRIES
AVIEW
PSE
```

We can see something new here - the text string append is APPEND for 41UCC. Most keyboards do not have an append mark, so this seems reasonable, and is certainly readable. So now our inductor value is labeled as being in millihenries, let's do the capacitor.

Here we run into a problem - MICROFARADS is too long to put on the display with the capacitor value. The usual notation for microfarads uses the Greek letter mu "µ". If the HP-41C had a lower case U we could use that. But wait - the HP-41C display has the Greek letter µ - but we can't get to it from the keyboard. Will 41UCC allow us to use it? Yes, 41UCC will but we will need to know a bit about the HP-41C instruction set.
A REAL PROGRAM

In the HP-41C the character code for a "mu" is 12 (0C hexadecimal.) How do we put this into a character string? Well, an append text string of three characters is encoded as 0F4H,07FH, followed by the three characters. If we change our program like this:

STO+ X ;DOUBLE IT
EQU TEXT4 0F4H ;APPEND TEXT STRING OF THREE CHRS.
EQU APPEND 07FH ;THE APPEND FUNCTION
EQU MU 12 ;GREEK LETTER MU
126 ;CONVERT TO MICROWATDS
/
STO CAPACITOR ;SAVE FOR FUTURE USE
;NOW DISPLAY THE CAPACITOR VALUE
T 'C= '
ARCL X
DB TEXT4,APPEND,' ','MU','F' ;LABEL IT AS MICROWATDS
AVIEW
PSE
END

it will cause the capacitor value to be properly labeled as being in microfarads. The DB instruction is not a standard HP-41C instruction. In fact it is not an HP-41C instruction at all, but what is called a pseudo-op. It is a pseudo HP-41C instruction called DEFINE BYTE, and it actually evaluates the rest of the line and passes the values it finds to the HP-41C unchanged. This is not something you will need to use in every program but is very handy to have when you do need it.

This seems to be about as much damage as we can do to such a simple program. If you don't understand something at this stage try going back, typing the program into your computer, and running it through 41UCC. Then feed it into your calculator and see what it looks like there. It will appear to be an old and familiar friend there, and you will be able to compare it to the 41UCC listing and see what was actually produced. Just as a passing note, if you want to make a direct comparison between the two programs you should add a '4L' on the command line for 41UCC.

A>41UCC I=LOWPASS,4L

This means 'use HP-41C Line numbers' - so the line numbers in 41UCC's LIST file and the line numbers you see on the HP-41C will be exactly the same. It makes the two programs much easier to compare. The barcode for this one is given in Appendix G - you might learn a good bit by reading it into your calculator, then comparing it to the listing.

There are two thing that we can still do to this program - if we use it a lot, we will always want to assign it to a key. In 41UCC this requires that we put a key assignment number after a label. To assign LOWPASS to the 'LN' key, we would modify our
program like this

```
LBL 'LOWPASS' : 15 ; ASSIGN TO THE 'LS' KEY
```

and the assignment would automatically be made for us when we scanned in the barcode for the program. RDC does not yet support automatic key assignments, but will in the next version.

Now let's take a look at the listing that 41UCC produced for our filter program. I will make a few notations on it to point out things of interest.
41UCC V 1.45, Copyright 1981 by Leslie Brooks. Distributed by Hand Held Products Incorporated.

PROGRAM COUNTERNUMBERS

0000 0000     ;LOW-PASS FILTER PROGRAM
0000 0000     ;WRITTEN BY LESLIE BROOKS
0000 0000     ;NOVEMBER 17, 1982.
0000 0000     ;SPECIAL EQUATES
0000 0000     ;
0000 0000     ;EQU TEXT4  OFGH     ;APPEND TEXT STRING OF THREE CHR.
0000 0000     ;EQU APPEND 07FH     ;THE APPEND FUNCTION
0000 0000     ;EQU NU    12     ;GREEK LETTER NU
0000 0000     ;REGISTER EQUATES
0000 0000     ;
0000 0000     ;EQU BASE    R1     ;USE R1 FOR THE BASE REGISTER
0000 0000     ;EQU FREQUENCY  BASE+0     ;THE CUTOFF FREQUENCY
0000 0000     ;EQU RESISTANCE  BASE+1     ;THE TERMINATING RESISTANCE
0000 0000     ;EQU CAPACITOR  BASE+2     ;CAPACITOR VALUE
0000 0000     ;EQU INDUCTOR  BASE+3     ;COIL VALUE
0000 0000     ;
0000 0000     ;C000F800 LBL 'LOWPASS'     ;OUR LOW-PASS FILTER PROGRAM
4CF5F50
415353
0000 0008     ;F8466245 T 'FREQUENCY=?'     ;ASK FOR THE CUTOFF FREQUENCY
5155454E
4359303F
0000 0017     ;PROMPT
0000 0018     ;STO  FREQUENCY     ;SAVE IT
0000 0019     ;F0522B54 T 'R(TERM)=?'     ;ASK FOR THE TERMINATING RESISTANCE
45524029
303F
0000 0023     ;PROMPT
0000 0024     ;STO  RESISTANCE     ;SAVE THE RESISTANCE
0000 0025     ;
0000 0025     ;CALCULATE THE INDUCTOR (COIL) FIRST
0000 0025     ;L = R/(PI * FREQUENCY)
0000 0025     ;
0000 0025     ;
0000 0025     ;RCL  FREQUENCY     ;GET THE FREQUENCY
0000 0026     ;2 PI
0000 0027     ;*     ;PI * FREQUENCY
0000 0028     ;/     ;DIVIDE INTO THE RESISTANCE
0000 0029     ;11101010 1000     ;CONVERT TO MILLIHENRIES
0000 002D     ;/     ;
0000 002E     ;33 STO  INDUCTOR     ;SAVE IT FOR FUTURE USE
0000 002F     ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
0000 002F     ;F34C3020 T 'L:'
0000 0033     ;9B73 ARCL  X
0000 0035     ;F47F204D APPEND ' MH'     ;UNITS ARE MILLIHENRIES
48
0000 003A     ;VIEW
0000 003B     ;99 PSE
0000 003C     ;33 STO  INDUCTOR     ;SAVE IT FOR FUTURE USE
21 0030 ; NOW DISPLAY THE CALCULATED INDUCTOR VALUE
21 0030 F34C3D20 T "L=
22 0041 9B73 ARCL X
23 0043 7E AVIEW
24 0044 89 PSE
25 0045 ;
25 0045 ; NOW CALCULATE THE CAPACITOR
25 0045 ; C = 1 / (2 * PI * RESISTANCE * FREQUENCY)
25 0045 ;
25 0045 ;
25 0045 21 RCL RESISTANCE ; GET THE RESISTANCE AGAIN
26 0046 20 RCL FREQUENCY ; AND THE FREQUENCY
27 0047 42 X
28 0048 72 PI
29 0049 42 X
30 004A 9273 STO+ X ; DOUBLE IT
31 004C 111B16 1E6 ; CONVERT TO MICROFARADS
32 004F 43 /
33 0050 32 STO CAPACITOR ; SAVE FOR FUTURE USE
34 0051 ; NOW DISPLAY THE CAPACITOR VALUE
34 0051 F3433D20 T "C=
35 0055 9B73 ARCL X
36 0057 F47F200C DB TEXT4,APPEND," µF" ; LABEL IT AS MICROFARADS
37 005C 7E AVIEW
38 005D 89 PSE
39 005E C0000D END
UNDEFINED ALPHA LABELS
(These are errors if not defined in another program)

<table>
<thead>
<tr>
<th>LABEL NAME</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
</table>

****** NO SYMBOLS WERE UNDEFINED ******

FLAG USAGE SUMMARY

FLAG & LINE NUMBERS OF REFERENCES TO THE FLAG

****** NO FLAGS WERE USED ******

NUMERIC LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL $</th>
<th>DEFINED ON</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL $ ON</th>
</tr>
</thead>
</table>

****** NO NUMERIC LABELS WERE USED ******
### REGISTER USAGE SUMMARY

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>LINE NUMBERS OF REFERENCES TO THE REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>4-S 8-R 26-R</td>
</tr>
<tr>
<td>001</td>
<td>7-S 25-R</td>
</tr>
<tr>
<td>002</td>
<td>33-S</td>
</tr>
<tr>
<td>003</td>
<td>14-S 20-S</td>
</tr>
<tr>
<td>X</td>
<td>16-R 22-R 30-S 35-R</td>
</tr>
</tbody>
</table>

**TAG MEANINGS ARE:**
- CF: CLEAR FLAG INDIRECT
- DI: DECREMENT INDIRECT AND SKIP IF EQUAL
- DS: DECREMENT AND SKIP IF EQUAL
- FC: FLAG CLEAR? INDIRECT
- FS: FLAG SET? INDIRECT
- GI: GOTO INDIRECT
- II: INCREMENT INDIRECT AND SKIP IF GREATER
- IS: INCREMENT AND SKIP IF GREATER
- R: RECALL
- RI: RECALL INDIRECT
- S: STORE
- SF: SET FLAG INDIRECT
- SI: STORE INDIRECT
- TC: FLAG TEST AND CLEAR INDIRECT
- XI: EXECUTE INDIRECT
- XC: EXCHANGE X AND R

---

### ALPHA LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOWPASS</td>
<td>1 0000</td>
<td></td>
</tr>
</tbody>
</table>

**TAG MEANINGS ARE:**
- GOTO
- EXECUTE
### INTEGER SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
<td>1 007F</td>
<td>36-</td>
</tr>
<tr>
<td>BASE</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>CAPACIT</td>
<td>1 0002</td>
<td>33-</td>
</tr>
<tr>
<td>FREQUEN</td>
<td>1 0000</td>
<td>4- 8- 26-</td>
</tr>
<tr>
<td>INDUCTO</td>
<td>1 0003</td>
<td>14- 20-</td>
</tr>
<tr>
<td>MU</td>
<td>1 000C</td>
<td>36-</td>
</tr>
<tr>
<td>R1</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>RESISTA</td>
<td>1 0001</td>
<td>7- 25-</td>
</tr>
<tr>
<td>TEXT4</td>
<td>1 00F4</td>
<td>36-</td>
</tr>
</tbody>
</table>

---

### STRING SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
</table>

**** NO STRING SYMBOLS WERE USED ****

---

### VARIABLE USAGE SUMMARY

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE VARIABLE</th>
</tr>
</thead>
</table>

**** NO VARIABLES WERE USED ****
Let's move on to another example program. This one will solve a common mathematical problem – finding zeros of a function. The method I will use is called the secant method; it is fairly fast and simple. A good explanation of the secant method, including a FORTRAN program example, is given in Elementary Numerical Analysis: An Algorithmic Approach by Conte and de Boor. For now all we need to know is the formula and how to use it. The secant method is described by

\[
X_{n+1} = X_n - \frac{X_n - X_{n-1}}{f(X_n) - f(X_{n-1})}
\]

It starts with a function \( f(X) \) and two guesses \((n, n-1)\) for a zero of the function; the guesses should be on either side of the actual zero. The formula above is then evaluated, and the result \((X_{n+1})\) becomes the new \( X_n \). The previous \( X_n \) becomes the new \( X_{n+1} \), and the old \( X_{n-1} \) is discarded. The formula is then reevaluated, and this continues until \( f(X) \) reaches zero (or very close to it). As an example, if we wanted to find the square root of 5, we would say that our function is

\[
f(X) = 5 - X^2
\]

because 5 minus \( X \) squared obviously equals zero if \( X \) is the square root of 5. We would also need to guess that the square root of 5 must lie between 1 and 5. From this information we could use the secant method to find the square root. By plugging our guesses 1 and 5 into the formula for the secant method we get

\[
X_{n+1} = 5 - \frac{5 - 1}{f(5) - f(1)}
\]

or

\[
X_{n+1} = 5 - \frac{4}{(-20) - 4}
\]

which gives

\[
X_{n+1} = 1.6666, \quad X_n = 5
\]

Plugging these guesses back into the formula will produce a new (and closer) guess, and so forth. Now let's write a program to do this.

```
TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'
LBL 'SC' ;ENTRY POINT TO THE PROGRAM
T 'GUESS 1?' ;ASK FOR GUESS 1
```
ANOTHER EXAMPLE PROGRAM

PROMPT
STO 00 ;SAVE X_{n-1}
T 'GUESS 2?' ;ASK FOR GUESS 2
STO 01 ;SAVE X_n
T 'FUNCTION NAME?' ;ASK FOR THE FUNCTION NAME
AON

PROMPT
ASTO 04 ;SAVE THE FUNCTION NAME
AOFF
.01
STO 02 ;LOOP 10 TIMES

;CALCULATE F(GUESS1) TO START THE PROGRAM

LBL 01
RCL 00 ;GET X_{n-1} BACK AGAIN
XEQ IND 04 ;EXECUTE THE FUNCTION
STO 05 ;AND SAVE F(X_{n-1})
RCL 01 ;GET X_n
XEQ IND 04 ;EVALUATE F(X_n)
RCL 01 ;GET X_n
RCL 00 ;AND X_{n-1}
- ;SUBTRACT THEM
X<>Y ;GET f(X_n) BACK
STO 2 ;SAVE IT AGAIN
RCL 05 ;AND GET f(X_{n-1})
- ;SUBTRACT THESE
/ ;(X_n - X_{n-1}) / (f(X_n) - f(X_{n-1}))
* ;MULTIPLY BY f(X_n)

;X now contains a correction factor to be added to X_n

RCL 01 ;GET X_n
X<>Y
-

;NOW WE HAVE X_{n+1} IN THE X REGISTER

X<> 01 ;EXCHANGE X_{n+1} WITH X_n
STO 00 ;X_n BECOMES THE NEW X_{n-1}
ISG 02 ;INCREMENT THE LOOP COUNTER
GTO 01 ;LOOP IF NOT DONE

;IF WE GET HERE, WE ARE DONE - DISPLAY THE RESULT
This program will certainly work, although it is hardly the best that we could do. However, there are several things we can learn from it. The first thing to notice here is the TITLE at the top - this is another pseudo instruction which causes a title to be printed at the top of every page of the listing. Other than this there is nothing of any importance in this version of the program. However, putting numbers directly into a program is very bad practice - they should always be equates so that they may be found and changed easily. Let's do that for this one.

TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'

;REGISTER EQUATES

EQU GUESSL 00 ;FIRST GUESS FOR X
EQU GUESS2 01 ;SECOND GUESS FOR X
EQU LOOP 02 ;LOOP COUNT
EQU FUNCTION 04 ;FUNCTION NAME
EQU SCRATCH 05 ;SCRATCH REGISTER (USUALLY HOLDS f(X_n-1)

;LABELS

EQU START 01 ;START OF THE MAIN LOOP

LBL 'SC' ;ENTRY POINT TO THE PROGRAM
T 'GUESS 1?' ;ASK FOR GUESS 1
PROMPT
STO GUESSL ;SAVE X_{n-1}
T 'GUESS 2?' ;ASK FOR GUESS 2
STO GUESS2 ;SAVE X_n
T 'FUNCTION NAME?' ;ASK FOR THE FUNCTION NAME
AON
PROMPT
ASTO FUNCTION ;SAVE THE FUNCTION NAME
AOFF
.01
STO LOOP ;LOOP 10 TIMES

;CALCULATE F(GUESSL) TO START THE PROGRAM

LBL START
RCL GUESSL ;GET X_{n-1} BACK AGAIN
ANOTHER EXAMPLE PROGRAM

XEQ IND FUNCTION ;EXECUTE THE FUNCTION
STO SCRATCH ;AND SAVE F(X_{n-1})
RCL GUESS2 ;GET X_n
XEQ IND FUNCTION ;EVALUATE F(X_n)
RCL GUESS2 ;GET X_n
RCL GUESS1 ;AND X_{n-1}
- ;SUBTRACT THEM
X<>Y ;GET f(X_n) BACK
STO Z ;SAVE IT AGAIN
RCL SCRATCH ;AND GET f(X_{n-1})
- ;SUBTRACT THESE
/ ;(X_n - X_{n-1}) / (f(X_n) - f(X_{n-1}))
* ;MULTIPLY BY f(X_n)

;X now contains a correction factor to be added to X_n

RCL GUESS2 ;GET X_n
X<>Y
-

;NOW WE HAVE X_{n+1} IN THE X REGISTER

X<> GUESS2 ;EXCHANGE X_{n+1} WITH X_n
STO GUESS1 ;X_n BECOMES THE NEW X_{n-1}
ISG LOOP ;INCREMENT THE LOOP COUNTER
GTO START ;LOOP IF NOT DONE

;IF WE GET HERE, WE ARE DONE - DISPLAY THE RESULT

RCL GUESS2 ;X_n
END

This is much easier to read than the original but it still has a constant embedded in it - the loop count (.01). If this were a large program the loop count might be referred to in many places, and we would want to be able to change it easily. In order to do this we would add another equate

EQU COUNT ' .01' ;MAXIMUM NUMBER OF TIMES THROUGH ;(DIVIDED BY 1000)

and the lines

.01
STO LOOP ;LOOP 10 TIMES

would become
COUNT
STO LOOP ; LOOP 10 TIMES

This may seem a bit unusual at first, but it really isn't hard to understand. The symbol COUNT has been given a string of characters - ' .01' - as its value. If we then put the word COUNT all by itself as the first symbol on a line, 41UCC will convert it to the equivalent string and evaluate the string. This same technique will also work after an XROM, but nowhere else. If you want to know more about this, look up STRING EQUATES in the section titled INTRODUCTION TO SPECIAL FEATURES.

Now if we have the need to put this loop count in several places throughout our program, someone else reading the program can immediately tell that this is the same loop count. If we had a different constant which also happened to be .01, we could give it a different name so that they would never be confused.

The next thing to do to our program is to allow a user to call it as a subroutine, which means we must skip the prompting for the initial guesses and function name. We can use flag 10 to tell us whether or not to prompt for this information - if flag 10 is set, we will skip the prompts and begin executing immediately. We will need to add

EQU NO_PROMPT 10 ; FLAG IS SET IF GUESSES ARE ALREADY ENTERED

LBL 'SC' ; ENTRY POINT TO THE PROGRAM
FS?C NO_PROMPT ; SET IF CALLED AS A SUBROUTINE
GTO START ; SKIP THE PROMPTING IF SET

to our program in order to allow this. Now our program looks like this

TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'

; REGISTER EQUATES

EQU GUESS1 00 ; FIRST GUESS FOR X
EQU GUESS2 01 ; SECOND GUESS FOR X
EQU LOOP 02 ; LOOP COUNT
EQU FUNCTION 04 ; FUNCTION NAME
EQU SCRATCH 05 ; SCRATCH REGISTER (USUALLY HOLDS f(x_{n-1})

; LABELS

EQU START 01 ; START OF THE MAIN LOOP

; FLAGS

26
EQU NO_PROMPT 10 ;FLAG IS SET IF GUESSES ARE ALREADY ENTERED

LBL 'SC' ;ENTRY POINT TO THE PROGRAM

;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME
FS?C NO_PROMPT ;SET IF CALLED AS A SUBROUTINE GTO START ;SKIP THE PROMPTING IF SET

;ELSE PROMPT NORMALLY
T 'GUESS 1?' ;ASK FOR GUESS 1 PROMPT STO GUESS1 ;SAVE X_{n-1} T 'GUESS 2?' ;ASK FOR GUESS 2 STO GUESS2 ;SAVE X_n T 'FUNCTION NAME?' ;ASK FOR THE FUNCTION NAME AON PROMPT ASTO FUNCTION ;SAVE THE FUNCTION NAME AOFF .01 STO LOOP ;LOOP 10 TIMES

;CALCULATE F(GUESS1) TO START THE PROGRAM

LBL START
RCL GUESS1 ;GET X_{n-1} BACK AGAIN XEQ IND FUNCTION ;EXECUTE THE FUNCTION STO SCRATCH ;AND SAVE F(X_{n-1}) RCL GUESS2 ;GET X_n XEQ IND FUNCTION ;EVALUATE F(X_n) RCL GUESS2 ;GET X_n RCL GUESS1 ;AND X_{n-1} - ;SUBTRACT THEM X<>Y ;GET f(X_n) BACK STO Z ;SAVE IT AGAIN RCL SCRATCH ;AND GET f(X_{n-1}) - ;SUBTRACT THESE /

; (X_n - X_{n-1}) / (f(X_n) - f(X_{n-1}))
* ;MULTIPLY BY \( f(x_n) \)

; \( x \) now contains a correction factor to be added to \( x_n \)

RCL GUESS2 ;GET \( x_n \)
X<>Y

; NOW WE HAVE \( x_{n+1} \) IN THE X REGISTER

X<> GUESS2 ;EXCHANGE \( x_{n+1} \) WITH \( x_n \)
STO GUESS1 ;\( x_n \) BECOMES THE NEW \( x_{n-1} \)
ISG LOOP ;INCREMENT THE LOOP COUNTER
GTO START ;LOOP IF NOT DONE

;IF WE GET HERE, WE ARE DONE - DISPLAY THE RESULT

RCL GUESS2 ;\( x_n \)
END

This seems to be about as much as we can do toward cleaning up the program as it is now - but perhaps we could generalize a few things. For example, the lines

T "FUNCTION NAME?" ;ASK FOR THE FUNCTION NAME
AON
PROMPT
ASTO FUNCTION ;SAVE THE FUNCTION NAME
AOFF

perform a function which could be used in many programs without change - is there any way we could actually do this? In fact there is - let's change these lines to a real routine:

EQU FUNC_NAME LBL_BASE ;CREATE A LABEL NUMBER FOR THIS ROUTINE
SET LBL_BASE LBL_BASE+1 ;CREATE A NEW LABEL BASE

LBL FUNC_NAME
T "FUNCTION NAME?" ;ASK FOR THE FUNCTION NAME
AON
PROMPT
ASTO FUNCTION ;SAVE THE FUNCTION NAME
AOFF

This creates a complete and useful function - but where did LBL_BASE come from, and what is this SET? The LBL_BASE is a symbol whose value comes from outside the function, and SET gives LBL_BASE a new value. If we come into this routine with LBL_BASE equal to 5, then FUNC_NAME will have the value 5, and our routine
ANOTHER EXAMPLE PROGRAM

will be labeled by label 5. The symbol LBL_BASE will get a new value - 6 - so that the next routine will be guaranteed to have a label that does not conflict with any other. Create this routine with your text editor and save it on disk as FUNCNAME.INC, we will use it in the next step.

Now we will change our secant program to look like this:

TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'

;SPECIAL EQUATES

EQU LBL_BASE 02 ;WE HAVE USED LABEL 1

;REGISTER EQUATES

EQU GUESS1 00 ;FIRST GUESS FOR X
EQU GUESS2 01 ;SECOND GUESS FOR X
EQU LOOP 02 ;LOOP COUNT
EQU FUNCTION 04 ;FUNCTION NAME
EQU SCRATCH 05 ;SCRATCH REGISTER (USUALLY Holds f(Xn-1)

;LABELS

EQU START 01 ;START OF THE MAIN LOOP

;FLAGS

EQU NO_PROMPT 10 ;FLAG IS SET IF GUESSES ARE ALREADY ENTERED

LBL 'SC' ;ENTRY POINT TO THE PROGRAM

;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME

FS>C NO_PROMPT ;SET IF CALLED AS A SUBROUTINE
GTO START ;SKIP THE PROMPTING IF SET

;ELSE PROMPT NORMALLY

T 'GUESS 1?' ;ASK FOR GUESS 1
PROMPT
Another Example Program

STO GUESS1 ;save X_{n-1}
T 'GUESS 2?' ;ask for GUESS 2
STO GUESS2 ;save X_n

;now prompt for the function name

#include funcname.inc

.01
STO LOOP ;loop 10 times

;calculate f(GUESS1) to start the program

lbl start

. .
. (This part of the program is unchanged.)
. .
end

Notice that there is now a definition for LBL_BASE, and
notice what has happened to the lines that used to ask for the
function name.

T 'FUNCTION NAME?' ;ask for the function name
AON
PROMPT
ASTO FUNCTION ;save the function name
AOFF

has been replaced by the single line

#include funcname.inc

When 41UCC sees this line it will go out to the disk, find
the file FUNCNAME.INC which we created, and insert it into the
program in place of the #INCLUDE line. Notice that the #INCLUDE
begins in the first column, that there is exactly one space
between the INCLUDE and the file name, and that there is nothing
else on the line. All of these things must be exactly so in
order for 41UCC to replace the line by the file. What this means
is that you may have a symbol called INCLUDE, and text strings
like

T '#INCLUDE'

and 41UCC will not do strange things with them behind your back.

30
Now when we look at the listing of our secant program, it will be similar to this

TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'

;SPECIAL EQUATES
SET LBL_BASE 02 ;WE HAVE USED LABEL 1

;REGISTER EQUATES
EQU GUESS1 00 ;FIRST GUESS FOR X
EQU GUESS2 01 ;SECOND GUESS FOR X
EQU LOOP 02 ;LOOP COUNT
EQU FUNCTION 04 ;FUNCTION NAME
EQU SCRATCH 05 ;SCRATCH REGISTER (USUALLY HOLDS f(X_{n-1})

;LABELS
EQU START 01 ;START OF THE MAIN LOOP

;FLAGS
EQU NO_PROMPT 10 ;FLAG IS SET IF GUESSES ARE ALREADY ENTERED

LBL 'sc' ;ENTRY POINT TO THE PROGRAM

;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME
FS?C NO_PROMPT ;SET IF CALLED AS A SUBROUTINE
GTO START ;SKIP THE PROMPTING IF SET

;ELSE PROMPT NORMALLY
T 'GUESS 1?' ;ASK FOR GUESS 1
PROMPT
STO GUESS1 ;SAVE X_{n-1}
T 'GUESS 2?' ;ASK FOR GUESS 2
STO GUESS2 ;SAVE X_n
ANOTHER EXAMPLE PROGRAM

; NOW PROMPT FOR THE FUNCTION NAME

EQU FUNC_NAME LBL_BASE  ; CREATE A LABEL NUMBER FOR THIS
               ; ROUTINE
SET LBL_BASE LBL_BASE+1 ; CREATE A NEW LABEL BASE

LBL FUNC_NAME
    T 'FUNCTION NAME?'  ; ASK FOR THE FUNCTION NAME
    AON
    PROMPT
    ASTO FUNCTION      ; SAVE THE FUNCTION NAME
    AOFF

    .01
    STO LOOP           ; LOOP 10 TIMES

; CALCULATE F(GUESS1) TO START THE PROGRAM

LBL START
    .
    .
    .
    (This part of the program is unchanged.)
    .
    .
END

and we can see that the function has been included exactly as we wished. Because we put a label on the function we can GTO or XEQ it from anywhere in the program, just as though we had typed it into the program rather than #INCLUDE'ing it. This is a very powerful technique, and can be used to build up libraries of useful functions which may then be used in many different programs. If a bug is discovered in one of your library routines you make the correction in only one place, and then recompile all of the affected programs - there is no need to edit each program that uses the routine.

As a final note you should notice that the same method used to guarantee that the label for our function was unique could be used to provide a unique register or group of registers for local storage. In the general case where we have a routine that needs two labels, three registers, and one flag all to itself, we would write it something like

; MY OWN LABELS

EQU LBL_1    LBL_BASE     ; MY FIRST LABEL
EQU LBL_2    LBL_BASE+1   ; MY SECOND LABEL
ANOTHER EXAMPLE PROGRAM

SET LBL_BASE LBL_BASE+2 ;CREATE A NEW LABEL BASE

;MY OWN REGISTERS

EQU REG_1 REG_BASE ;MY FIRST REGISTER
EQU REG_2 REG_BASE+1 ;MY SECOND REGISTER
EQU REG_3 REG_BASE+2 ;MY THIRD REGISTER

SET REG_BASE REG_BASE+3 ;CREATE A NEW REGISTER BASE

;MY OWN FLAG

EQU FLAG_1 FLG_BASE ;MY OWN PERSONAL FLAG

SET FLG_BASE FLG_BASE+1 ;CREATE A NEW FLAG BASE

Now let's suppose that we do make a change to a library routine and we want to update all of our programs that use this routine. A quick check of our documentation reveals that the programs affected are SECANT, GEAR, NEWTON, and PI. Rather than recompiling them with

A>41UCC I=SECANT

A>41UCC I=GEAR

A>41UCC I=NEWTON

and so on ad nauseum, why don't we just

A>41UCC

41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS. DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED. VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

?I=SECANT

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

?I=GEAR

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

?I=NEWTON

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

?I=PI
ANOTHER EXAMPLE PROGRAM

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

?^Z
A>

This is much quicker and easier than the first method, because 41UCC does not have to be reloaded from disk each time. An even easier method, if we are going to be making a number of changes to the same set of programs, would be to create a file

I=SECANT
I=GEAR
I=NEWTON
I=PI

and call it something like TEST.IND. We can now use it as an INDIRECT COMMAND FILE to pass instructions to 41UCC, simply by typing:

41UCC @TEST

After 41UCC has executed, the screen will look like this:

A>41UCC @TEST

41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS.
DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED.
VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

I=SECANT

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

I=GEAR

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

I=NEWTON

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO

I=PI

0 ERROR(S) IN PHASE ONE
0 ERROR(S) IN PASS ONE
0 ERROR(S) IN PASS TWO
ANOTHER EXAMPLE PROGRAM

41UCC has read the file TEST.IND one line at a time, and has executed those lines just as though they had been typed in at the console. This is a very powerful feature, and very advantageous whenever you are working with multiple programs at one time.

A REAL LISTING

Now let's take a look at the listing 41UCC produces from the program SECANT. I will make some notes on it to point out things of particular interest. A listing of TST28 and barcode for FILTER, SECANT, and TST28 is included in Appendix G. TST28 is a test program which contains an example of every instruction 41UCC understands, all in alphabetical order. If you have doubts about the form of a particular instruction, take a look at this listing and it may help.
TITLE "SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS"

;SPECIAL EQUATES
SET LBL_BASE 02 ;WE HAVE USED LABEL 1

;REGISTER EQUATES
EQU GUESS1 00 ;FIRST GUESS FOR X
EQU GUESS2 01 ;SECOND GUESS FOR X
EQU LOOP 02 ;LOOP COUNT
EQU FUNCTION 04 ;FUNCTION NAME
EQU SCRATCH 05 ;SCRATCH REGISTER (USUALLY HOLDS f(Xn-1)

;LABELS
EQU START 01 ;START OF THE MAIN LOOP

;FLAGS
EQU NO_PROMPT 10 ;FLAG IS SET IF GUESSES ARE ALREADY ENTERED

LBL 'SC' ;ENTRY POINT TO THE PROGRAM

C00F300 LBL 'SC' ;ENTRY POINT TO THE PROGRAM

5343

0006 ;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T
0006 ;PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME

0006 A00A FS?C NO_PROMPT ;SET IF CALLED AS A SUBROUTINE
0008 B200 GTO START ;SKIP THE PROMPTING IF SET

000A ;ELSE PROMPT NORMALLY
000A

000A F8475545 T "GUESS 1?" ;ASK FOR GUESS 1
53532031 3F

0013 8E PROMPT
0014 30 STO GUESS1 ;SAVE Xn-1
0015 F8475545 T "GUESS 2?" ;ASK FOR GUESS 2
53532032 3F

001E 31 STO GUESS2 ;SAVE Xn
001F ;NOW PROMPT FOR THE FUNCTION NAME
001F

001F EQU FUNC_NAME LBL_BASE ;CREATE A LABEL NUMBER FOR THIS
001F ;ROUTINE
001F SET LBL_BASE LBL_BASE+1 ;CREATE A NEW LABEL BASE
001F
SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS

LBL FUNC_NAME
T "FUNCTION NAME?" ;ASK FOR THE FUNCTION NAME
4354494F
4E204E41
4D453F

8C AON
PROMPT
ASTO FUNCTION ;SAVE THE FUNCTION NAME
ROFF

1A1011 .01
STO LOOP ;LOOP 10 TIMES

20 RCL GUESS1 ;GET Xn-1 BACK AGAIN
XEQ IND FUNCTION ;EXECUTE THE FUNCTION
STO SCRATCH ;AND SAVE F(Xn-1)

21 RCL GUESS2 ;GET Xn
XEQ IND FUNCTION ;EVALUATE F(Xn)

21 RCL GUESS1 ;AND Xn-1
- »SUBTRACT THEN
X<>Y ;GET F(Xn) BACK

9171 STO 2 ;SAVE IT AGAIN

25 RCL SCRATCH »AND GET f(Xn-1)
41 - »SUBTRACT THESE
/ ;( Xn - Xn-1 ) / ( f(Xn) - f(Xn-1))
X *MULTIPLY BY f(Xn)

21 RCL GUESS2 »GET Xn
- ;SUBTRACT AGAIN

401 X<> GUESS2 »EXCHANGE Xn+1 WITH Xn

30 STO GUESS1 ;Xn BECOMES THE NEW Xn-1

8200 »IF NOT DONE
STO GUESS2 ;NOW WE HAVE Xn+1 IN THE X REGISTER

401 X<> GUESS2 »EXCHANGE Xn+1 WITH Xn

9602 ISG LOOP ;INCREMENT THE LOOP COUNTER

8200 »LOOP IF NOT DONE

501 X<> GUESS2 »NOW WE HAVE Xn+1 IN THE X REGISTER

8200 »LOOP IF NOT DONE

CE01 X<> GUESS2 ;EXCHANGE Xn+1 WITH Xn

30 STO GUESS1 ;Xn BECOMES THE NEW Xn-1

9602 ISG LOOP ;INCREMENT THE LOOP COUNTER

B200 GTO START ;LOOP IF NOT DONE

B200 GTO START ;LOOP IF NOT DONE

21 RCL GUESS2 ;Xn

C0000D END
SECANT METHOD FOR \( F(x) \geq 0 \) BY LESLIE BROOKS

UNDEFINED ALPHA LABELS
(THese are errors if not defined in another program)

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>ON</td>
<td></td>
</tr>
</tbody>
</table>

**** NO SYMBOLS WERE UNDEFINED ****

FLAG USAGE SUMMARY

FLAG & LINE NUMBERS OF REFERENCES TO THE FLAG

010 2-TC

TAG MEANINGS ARE:
- CF: CLEAR FLAG
- FS: FLAG SET?
- FC: FLAG CLEAR?
- SF: SET FLAG
- TC: FLAG TEST AND CLEAR

NUMERIC LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>17</td>
<td>3-6 38-6</td>
</tr>
<tr>
<td>002</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- G: GOTO
- X: EXECUTE
SECANT METHOD FOR $F(X) \geq 0$. BY LESLIE BROOKS

REGISTER USAGE SUMMARY

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>LINE NUMBERS OF REFERENCES TO THE REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>6-S 18-R 24-R 36-S</td>
</tr>
<tr>
<td>001</td>
<td>8-S 21-R 23-R 32-R 35-XC 39-R</td>
</tr>
<tr>
<td>002</td>
<td>16-S 37-1S</td>
</tr>
<tr>
<td>004</td>
<td>13-S 19-XI 22-XI</td>
</tr>
<tr>
<td>005</td>
<td>20-S 28-R</td>
</tr>
<tr>
<td>z</td>
<td>21-S</td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:  
- CF: CLEAR FLAG INDIRECT
- DI: DECREMENT INDIRECT AND SKIP IF EQUAL
- DS: DECREMENT AND SKIP IF EQUAL
- FC: FLAG CLEAR? INDIRECT
- FS: FLAG SET? INDIRECT
- GI: GOTO INDIRECT
- II: INCREMENT INDIRECT AND SKIP IF GREATER
- IS: INCREMENT AND SKIP IF GREATER
- R: RECALL
- RI: RECALL INDIRECT
- S: STORE
- SF: SET FLAG INDIRECT
- SI: STORE INDIRECT
- TC: FLAG TEST AND CLEAR INDIRECT
- XI: EXECUTE INDIRECT
- XC: EXCHANGE X AND R

ALPHA LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>1 0000</td>
<td></td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:  
- G: GOTO
- X: EXECUTE
SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS

### INTEGER SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIO</td>
<td>1 0004</td>
<td>13- 19- 22-</td>
</tr>
<tr>
<td>FUNC_NA</td>
<td>9 0002</td>
<td>9-</td>
</tr>
<tr>
<td>GUESS1</td>
<td>1 0000</td>
<td>6- 18- 24- 36-</td>
</tr>
<tr>
<td>GUESS2</td>
<td>1 0001</td>
<td>8- 21- 23- 32- 35- 39-</td>
</tr>
<tr>
<td>LOOP</td>
<td>1 0002</td>
<td>16- 37-</td>
</tr>
<tr>
<td>NO_PROM</td>
<td>1 0000A</td>
<td>2-</td>
</tr>
<tr>
<td>R1</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>SCRATCH</td>
<td>1 0005</td>
<td>20- 28-</td>
</tr>
<tr>
<td>START</td>
<td>1 0001</td>
<td>3- 17- 38-</td>
</tr>
</tbody>
</table>

### STRING SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
</table>

***** NO STRING SYMBOLS WERE USED *****

### VARIABLE USAGE SUMMARY

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBL_BAS</td>
<td>1 0003</td>
<td>9-</td>
</tr>
</tbody>
</table>
This section will explain (in fairly dry and technical detail) some of the features available in 41UCC that are not found in the HP-41. Most of the examples of their uses are in the previous sections. These features can be very powerful.

Probably the first thing that people notice about an HP-41 program to be run through 41UCC is the comments. A comment may appear anywhere in the program, on any line, so long as it is preceded by a semicolon. Everything following a semicolon on a line is considered to be a comment and is ignored by 41UCC.

```
;this is a comment, because it is preceded by a semicolon
```

Comments in a program are extremely useful and important, particularly if you ever want to go back and modify a program, but that is not all that you get from using 41UCC.

**DEFINE BYTE**

One very useful feature included in 41UCC is the "DEFINE BYTE" pseudo-op. (A pseudo-op is an instruction for 41UCC rather than for the HP-41C). The DB pseudo-op allows you to define bytes which will be included in your program as is, with no questions asked. For example, suppose you wish to display a capacitor value in microfarads - the HP-41 has the Greek letter mu available, but it is not on the keyboard. The "DB" pseudo-op makes it very easy to include this character in a text string.

```
EQU append 07FH ;;41C APPEND FUNCTION
EQU MU 0CH ;;GREEK LETTER MU
EQU TEXT4 0F4H ;;41C TEXT STRING 4 FUNCTION

T 'C=' ;;SHOW THE CAPACITOR VALUE
ARCL X ;;VALUE IS IN X
DB TEXT4, APPEND,' ',MU,'F' ;;MICROFARADS
```

This will append "space, mu, F" to the capacitor value in the alpha register. Notice that "DB" may take multiple arguments (or operands), all separated by commas. These operands may in fact be completely general expressions - there is no inherent limit on what you may put in a "DB" statement, provided that it can be evaluated properly.

**END**

The "END" instruction informs 41UCC that the end of the program has been reached. No further lines will be processed, and you will get an error message if there are lines after the "END". The "END" is not mandatory.

**EQUATES**

An equate assigns a permanent value to a symbol. This symbol may then be used anywhere in the program. For example,
INTRODUCTION TO SPECIAL FEATURES

EQU FREQUENCY 00 ; INPUT FREQUENCY

assigns the value 0 to the symbol FREQUENCY, and you may then use
FREQUENCY freely throughout your program. Thus we have decided
to use register 0 for storing a frequency and we can now say

STO FREQUENCY ; SAVE THE INPUT FREQUENCY
STO+ FREQUENCY ; ADD AN OFFSET TO THE FREQUENCY
ST+ FREQUENCY+3 ; STORE IN REGISTER 3
RCL FREQUENCY ; RECALL THE LAST USED FREQUENCY

anywhere in our program. 41UCC will understand that we are
referring to register 0.

ONLY THE FIRST SEVEN CHARACTERS of a symbol are significant.
That is, FREQUENCY, FREQUENC, FREQUEN1, and FREQUENTLY will all
be treated as identical by 41UCC. Also, the only characters
allowed in a symbol are the letters A-Z, the digits 0-9, the
dollar sign '$', and the underline '_'. Lower case letters are
considered the same as upper case letters. All symbols must
begin with a letter.

The EQUate may go anywhere in your program, and may in some
cases be referred to before its definition, as here

STO TIME ; SAVE THE CURRENT TIME
EQU TIME 05 ; REGISTER FOR CURRENT TIME

It is usually a good idea to put equates at the beginning of
your program, so that they are easy to find, but there are excep-
tions. When we look at the SET and #INCLUDE psuedo-ops we will
see that it can be advantageous (or even essential) to put
certain EQUates elsewhere in the program. It would be good
practice to set them off in some special way so that they are
still easy to find - for example, by using PAGE to go to the top
of a new page before any embedded equates, or by putting several
comment lines ahead of them like this

;;; EQUATES FOR THE DATE MODULE
;
; THE DATE MODULE PERFORMS THE FOLLOWING FUNCTIONS:
; 1)......
; 2).....
; 3)....
;
; THE FOLLOWING SYMBOLS ARE DECLARED

EQU CDATE BASE + 1 ; CURRENT DATE
EQU SDATE BASE + 2 ; STAR DATE
EQU PDATE BASE + 3 ; PREVIOUS DATE OF INTEREST

STRING EQUATES

Another form of the equate is the string equate, so called
because it assigns a text string value to a symbol. String equates have the following form:

```
EQU MATRIX 'XROM 30,01' ;MATH PAC FUNCTION
EQU SIMEQ '30,02' ;SIMULTANEOUS EQUATIONS
EQU DET 'XROM 30,06 ;TAKE THE DETERMINANT'
EQU TEXT 'HELLO WORLD'
```

Probably the most common use for this will be to define XROM functions for later use in a program. To execute the MATRIX program we need only have a program such as the following:

```
EQU MATRIX 'XROM 30,01 ;HP MATH PAC MATRIX FUNCTION'
EQU SIMEQ '30,02'
LBL 'MATH1A';MATH PAC NAME
MATRIX
XROM SIMEQ
END
```

41UCC will recognize that the symbol MATRIX has the value 'XROM 30,01', and will expand it and evaluate the resulting XROM. The List file that is produced will look (in part) like this:

```
LBL 'MATH1A';MATH PAC NAME
XROM 30,01 ;HP MATH PAC MATRIX FUNCTION
XROM SIMEQ
```

Notice that any text string may be included in a string equate – it does not have to be used merely for XROM's. Also, you may include a comment within the string equate if you wish, and this comment will then appear on any line where you use the equated symbol as an operator. This is demonstrated by the MATRIX definition and use above. You should also notice that the two methods of defining an XROM produce different results in the listing – the first method (used for MATRIX) produces the XROM number in the listing, the second method (used for SIMEQ) produces the XROM name. This can be used to advantage, as we can see in this example:

```
EQU ACCURACY '2 E-90 ;MAXIMUM ERROR ALLOWED'
EQU CERROR 03 ;USE REGISTER 3 FOR THE CALCULATED ERROR
```

```
LBL 'ERROR'
ACCURACY
RCL CERROR ;GET THE CALCULATED ERROR
X<Y? ;IS THE ERROR LESS THAN THE LIMIT?
RTN ;RETURN IF LESS THAN
. ;ELSE KEEP GOING
```

After 41UCC has been run on this program, we can look at the listing and see (in part):

```
43
```
INTRODUCTION TO SPECIAL FEATURES

LBL 'ERROR'
2 E-90 ;MAXIMUM ERROR ALLOWED
RCL ERROR ;GET THE CALCULATED ERROR
.
.
.

Now it is easy to see that the symbol ACCURACY has been converted to its value - the number 2 E-90. This allows you to put frequently used constants at the beginning of the program where they are easy to find. If the constants change, it is very simple to change the constants in just one place - where they are defined - rather than going through the entire program to find and change all of them.

Symbols having string values may appear only as the first symbol on a line or as the operand of XROM.

EXPRESSIONS

Expressions may appear anywhere a numeric operand would be valid. The supported operators are:

+ addition
- subtraction
* multiplication
/ division
AND boolean product
EQ test for equality
GE test for greater than or equal to
GT test for greater than
LE test for less than or equal to
LT test for less than
MOD remainder after division
NE test for not equal
NOT unary one's complement
OR boolean sum
SHL shift a left b bits, end off, zero fill
SHR shift a right b bits, end off, zero fill
XOR boolean difference

Standard evaluation hierarchy is used, but nested parentheses may be used to force any order of evaluation desired. Constants may be either numeric or ASCII (ASCII constants must be in quotes). Numeric constants may have a post radix (B=binary, O,O=Octal, D=Decimal, and H=Hexadecimal). The default base is decimal. All numeric constants must start with a digit from 0 through 9. All of the following are valid constants:

OABH 10 525 10010011B 125Q 125O 525D

while these are invalid:

F5H - does not start with a digit
10010011 - is too large to be a valid decimal number
525E3 - not a valid decimal number. (This would be a
valid number for the HP-41C, if entered into a program like this:

525E3 ;constant offset
STO OFFSET ;save the initial value

but it is not valid in an expression that 41UCC must evaluate - such as:

TONE OFFSET AND 525E3

or

STO INDEX + 525E3

41UCC works with 16 bit quantities in expressions, and 525E3 just isn't valid.)

We could fix the invalid numbers above as follows:

0F5H - valid hex number
10010011B - valid binary number
25E3H - valid hex number

All arithmetic is sixteen bit integer only. Some examples of using expressions in 41UCC are:

SET BASE 10 ;SET THE BASE REGISTER TO 10
EQU FREQ BASE+00
EQU TIME FREQ+1
EQU TIME1 TIME+1
EQU COMPLEX TIME1+1 ;COMPLEX SUM
EQU ROTATION (COMPLEX+2)/FREQUENCY
SET BASE BASE+6 ;NEW BASE REGISTER

Now we can change where our registers are merely by changing one definition - for the base register.

GLOBAL LABELS

Global labels "A" (global labels being those that show up in catalog 1 listings) are perfectly understood by the HP-41, in spite of Hewlett-Packard's failure to provide a convenient means of producing them. 41UCC has several instructions that support single character global labels that would normally be local labels. The first instruction in this class is "GLBL", which forces a label to be global. For example:

GLBL 'A' ;PRODUCES A GLOBAL LABEL 'A'
GLBL 'FRED' ;HAS NO AFFECT ON 'FRED'
GLBL 'c' ;PRODUCES A GLOBAL 'c'

Now to access these labels, we need instructions which explicitly reference a global label. Because 41UCC has these instructions, a global "A" and a local "A" are not considered to be the same label and do not cause a double definition error.
INTRODUCTION TO SPECIAL FEATURES

GTOG 'A' ; PRODUCES A GOTO GLOBAL LABEL 'A'
GTOG 'FRED' ; PRODUCES A NORMAL GOTO 'FRED'
GTOG 'c' ; PRODUCES A GOTO GLOBAL 'c'

XEQG 'A' ; PRODUCES AN EXECUTE GLOBAL LABEL 'A'
XEQG 'FRED' ; PRODUCES A NORMAL XEQ 'FRED'
XEQG 'c' ; PRODUCES AN EXECUTE GLOBAL 'c'

INCLUDE

A very frequent problem in large programming projects is the passing back and forth (or sharing) of modules between programmers or between programs. 41UCC provides a very convenient mechanism to solve this problem - the #INCLUDE feature. Assuming that a file MATHIA.INC exists on drive A, and that it contains the definitions of all of the HP math module functions, these definitions may be inserted directly into your program by placing the following line at the appropriate place:

#INCLUDE MATHIA.INC

You should type the #INCLUDE exactly as it appears here, with nothing else on the line, and with just one space or tab between the end of the #INCLUDE and the name of the file to include. The #INCLUDE line will be replaced (in your program's listing) by the actual text of the file MATHIA.INC. This feature is not limited to including definitions; any text, including subroutines, may be inserted into a file this way. However, nested includes (an include within an included routine) are not allowed.

You may wish to put a PAGE instruction immediately before or after the #INCLUDE, so that the #INCLUDE'd text will be set off from the rest of your program. This makes for easier reading and easier recognition of text that was brought in from another file.

KEY ASSIGNMENTS

Key assignments are a great convenience, and are supported by 41UCC. For example, suppose you have a long program which you wish to compile under 41UCC, and you would like to have the main entry point assigned to a key for easy execution. You could assign the key by hand every time you download a new copy of your program to the HP-41C, but this would get very tiring after a while. 41UCC provides a means for you to put the key assignment into the program so that it is automatically assigned by the bar code reader whenever you read in the program.

LBL 'KEY' : 15 ;ASSIGN TO THE 'LN' KEY
LBL 'KEY2' : -15

This will assign KEY to the LN key (key code 15) and assign KEY2 to the shifted LN key. The key codes are exactly the same as the ones you would see on the HP-41 display if you assigned the keys.
INTRODUCTION TO SPECIAL FEATURES

manually. If you are not sure what keycode to use for a particular key, just pick up your HP-41C and assign some function to the key you wish to use. The calculator will display a keycode after the function name when you make the assignment. Use this same keycode after your label in your program, and 41UCC will automatically make the key assignment for you in the barcode.

PAGE

PAGE is a pseudo-op that tells 41UCC to go to the top of the next page before printing the next line of your LiST file. You may place the PAGE command anywhere in your program, and may have as many as you wish. It is very useful for formatting your program for ease of reading, and is frequently used to separate major routines from each other, or to separate #INCLUDE files from each other.

SET

The SET pseudo-op assigns a value to a variable. For example, you could define a base register using SET, and change it further down in your program with another SET instruction. Notice that a symbol defined with EQU may not have its value changed later - its value is permanent.

```
SET BASE 00 ;BASE REGISTER
EQU TIME BASE+00 ;CURRENT TIME
EQU ANGLE BASE+1 ;PHASE ANGLE
SET BASE BASE+2

#INCLUDE SUB1.UCC
```

Now the subroutine can be guaranteed to have non-conflicting register assignments if it looks like this:

```
;SUBROUTINE ONE
EQU THETA BASE+00
EQU GAMMA BASE+1
EQU DELTA BASE+2
SET BASE BASE+3 ;NEW BASE REGISTER
```

TITLE

The TITLE pseudo-op tells 41UCC that you would like a title to be printed at the top of every page of your LiST file. The TITLE pseudo-op is followed by the title (in quotes) that you wish to have printed.

```
TITLE 'MY PROGRAM TO SOLVE A PROBLEM. COPYRIGHT 1982. ' 
```

Now you will have a title printed at the top of every page of your listing, just as you have it within the quotes. You may have more than one title within a single program if you wish. Multiple TITLEs might be used to have major routines or sections
of the program clearly identified at the top of the page, or to have #INCLUDE files clearly identify themselves.
Parameters for 41UCC may be given in any order and are separated by commas. Only one is mandatory - the input file specification. The parameters specified are valid only for the line on which they are specified. That is, if you are in interactive mode, specifying PR on one line will give you private bar code for that one compilation, but the file compiled by the next line will have public bar code unless you use the PR command again. The only exceptions to this are

1) the LC= option (line count)
2) the BC= option (barcode length)
3) the TL= option (total lines on a page)

These three options, once changed, remain valid until you reload 41UCC from disk. The reason for this difference is that the paper size you are using (and therefore the Line Count per page) is not likely to change from one compilation to the next, while the other parameters may easily change.

4L specifies that you would like the listing line numbers to match exactly the line numbers for the same program on the HP-41. If you do not specify 4L, every line of the listing will have a unique line number.

B= specifies the binary file name. If no binary file name is given, it defaults to the input file name and type "BIN".

BC= specifies the barcode Bar Count. This is the number of unit width bars that your printer can print on one line. For narrow printers such as the Epson MX-80 (BC=250), 41UCC will generate only about 10 bytes of bar code per line. For wide printers such as the Printronix or Trilog, BC may be set to a higher value (about 400) and 41UCC will generate a full 16 bytes of barcode per line.

CO= specifies a console output name. If no output file name is given, it defaults to CON: (i.e. the physical console). If a filename is specified, all messages that would normally go the console will go to the file specified. This can be very handy when you want to go get a cup of coffee.

I= specifies the input file name. Type "UCC" is assumed and may not be overridden. This is the only required parameter.

L= specifies the listing file name. If no listing file name is given, it defaults to the input file name and type "LST".

LC= specifies the line count to be used in the listing file. This is the actual number of lines per page on which you wish printing to occur. Header lines and blank lines are included in this count. The line count may be specified in decimal, hex, octal, or even binary, provided that the proper post radix is used. If no post radix is given, decimal is assumed. This is the only parameter which carries over from one compilation to the
41UCC COMMAND LINE PARAMETERS

next when you are in interactive mode or indirect mode.

NB specifies that no binary file is to be generated.

NL specifies that no listing file is to be generated.

NW specifies that no wand file is to be generated.

NX specifies that no cross reference is to be generated in the List file

PR specifies that private bar code is desired.

Rl= specifies the value of the symbol Rl. If no Rl is given in the command line, Rl will assume the value zero. These symbols may currently be given only numeric values - not strings. They will accept any number which can be represented in 16 bits, signed or unsigned. Thus you could say

A>41UCC I=TEST,R1=25
A>41UCC I=TEST,R1=25H
A>41UCC I=TEST,R1=0010010111B
A>41UCC I=TEST,R1=-87H

and all of these would be valid.

R2=, R3=, R4= all work like Rl above

TL= specifies the total number of lines on a page. Thus if you had an 8 line per inch printer and 11 inch paper, you might wish to specify

TL=88, LC=80

on the command line. This would cause 41UCC to print on 80 of the 88 available lines.

W= specifies the wand file name. If no wand file name is given, it defaults to the input file name and type "WND".

Valid file names include standard CP/M file names, and also the logical device names (CON:, LST:, PUN:, RDR:, or NUL:). For example, the command line:

A>41UCC I=TEST1,L=LST:,BC=300,NB

will run 41UCC with TEST1 as the input file, send the listing file directly to the list device, generate bar code up to 300 unit widths long per line, and produce no binary file.

SYSTEM COMMANDS

50
System commands are not options or parameters on the command line; they must appear as the only command on a line. The supported system commands are:

/DIR d:afn
prints a directory to the console of all files on drive d: satisfying the specified file name. The default file name is *.* and the default drive is the current drive. Read/Only files will be preceded by a greater than sign (>) rather than a colon. System files will not be listed.

/DRIVE d:
specifies a new drive as the current drive.

/ERA d:afn
erases the specified file(s). If the file name *.* is specified, the user will be asked to confirm this before the files are erased.

/REN newfn=oldfn
renames a file. The file names specified must be unambiguous.

/RESET
makes all drives read/write again.

/SET afn $a
sets attribute "a" on all files satisfying the file name.
Legal attributes are:

- DIR make file(s) appear in the directory
- SYS do not list file(s) in the directory
- R/W make file(s) Read/Write
- R/O make file(s) Read/Only

/TYP e ufn
type the specified file to the console.

/USER n
sets the user number to n (0-15).

These commands are quite useful when 41UCC is used in the interactive mode - for instance, we could erase files and get a directory from within 41UCC, simply by typing the command in response to 41UCC's prompt.

A>41UCC I=TEST.UCC
41UCC - AN HP-41C USER CODE COMPILER. COPYRIGHT 1981 BY LESLIE BROOKS. DISTRIBUTED BY HAND HELD PRODUCTS INCORPORATED. VERSION 1.45 - NOVEMBER 8, 1982. Serial Number AC0002

?/dir b:*.*.ucc
will give us a directory of all files of type "UCC" on drive B.
A few instructions had to be changed from their familiar HP-41C form due to limitations of standard keyboards. A few are also allowed to have alternate forms. All of these are in the following list.

<table>
<thead>
<tr>
<th>HP-41 INSTRUCTION</th>
<th>41UCC FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 to the X POWER</td>
<td>10**X, 10^X</td>
</tr>
<tr>
<td>APPEND TEXT STRING</td>
<td>APPEND, APPND, APND</td>
</tr>
<tr>
<td>CLEAR THE SUMMATION REGISTERS</td>
<td>CLS, CLSIGMA</td>
</tr>
<tr>
<td>DEGREES TO RADIANS</td>
<td>D-R, D-&gt;R</td>
</tr>
<tr>
<td>e to the X POWER</td>
<td>E**X, E^X</td>
</tr>
<tr>
<td>e to the X POWER MINUS ONE</td>
<td>E**X-1, E^X-1</td>
</tr>
<tr>
<td>ENTER</td>
<td>ENTER, ENTER^</td>
</tr>
<tr>
<td>GTO</td>
<td>GTO, GOTO</td>
</tr>
<tr>
<td>POLAR TO RECTANGULAR</td>
<td>P-R, P-&gt;R</td>
</tr>
<tr>
<td>RADIANS TO DEGREES</td>
<td>R-D, R-&gt;D</td>
</tr>
<tr>
<td>RECTANGULAR TO POLAR</td>
<td>R-P, R-&gt;P</td>
</tr>
<tr>
<td>ROLL (STACK) UP</td>
<td>R^</td>
</tr>
<tr>
<td>SIGMA PLUS</td>
<td>S+, SIGMA+</td>
</tr>
<tr>
<td>SIGMA MINUS</td>
<td>S-, SIGMA-</td>
</tr>
<tr>
<td>STATISTICAL REGISTERS</td>
<td>SREG, SIGMAREG</td>
</tr>
<tr>
<td>STORE TIMES</td>
<td>ST*, STO*</td>
</tr>
<tr>
<td>STORE PLUS</td>
<td>ST+, STO+</td>
</tr>
<tr>
<td>STORE DIVIDE</td>
<td>ST/, STO/</td>
</tr>
<tr>
<td>STORE MINUS</td>
<td>ST-, STO-</td>
</tr>
<tr>
<td>TEXT STRING</td>
<td>T</td>
</tr>
<tr>
<td>X NOT EQUAL TO ZERO</td>
<td>X¹=0?, X#0?, X&lt;&gt;0?</td>
</tr>
<tr>
<td>X NOT EQUAL TO Y</td>
<td>X¹=Y?, X#Y?, X&lt;&gt;Y?</td>
</tr>
<tr>
<td>X SQUARED</td>
<td>X**2, X^2</td>
</tr>
<tr>
<td>Y to the X POWER</td>
<td>Y**X, Y^X</td>
</tr>
</tbody>
</table>
LISTING ERROR MESSAGES

A - Argument error - the wrong type of Argument was encountered when evaluating a line - for example, a quoted string was encountered following a "TONE" instruction.

C - An illegal Character was encountered in the operand. This could be caused by using WordStar to edit your program and forgetting to use non-document mode. It could also be caused by something obvious like trying to use a ~ in a symbol.

D - A symbol was Defined twice or more. This may be caused by accidentally giving two routines the same name, or by \#INCLUDE'ing a file which contains a name conflicting with one of your own. Only the first seven characters of a symbol are significant, so long symbols may also cause a doubly-defined error if sufficient care is not used in naming them. Look in the cross reference in the listing to find the conflicting definition. That will give you the line number and you can then go see exactly what 41UCC is complaining about. Trying to SET an EQUated symbol (or vice versa) will also produce this error.

E - An Expression was encountered which could not be properly evaluated. This could be caused by an improperly formed expression (e.g. 3+*5) or by an undefined symbol.

O - A symbol Overflowed the symbol buffer. This should be a very rare error, and would normally indicate something seriously wrong with 41UCC, your program or your system. If you have an extremely long line try breaking it up.

P - Phase error - the first and second passes of 41UCC did not agree on how many bytes were in your program. The most likely causes of this would be an undefined symbol or an illegal forward reference. Because of the problems people frequently have with this error, let's have an example:

```
0 0000 LBL 'TEST' ;TEST PROGRAM
1 U 0008 GTO 'A' ;'A' IS UNDEFINED
2 P 0008 LBL 'B' ;SECOND ENTRY POINT
END
```

Here, label 'B' got a phase error, because label 'A' was undefined. If you fix the "U" error on line 1, the "P" error on line 2 will disappear. A more difficult problem to catch is an illegal forward reference:

```
0 0000 EQU INPUT OUTPUT+1 ;USE FOR INPUT ROUTINE
1 0000 EQU OUTPUT 02 ;USE LABEL 2 FOR OUTPUT ROUTINE
2 0000
3 0000 LBL 'TEST2'
4 E 0008 GTO INPUT ;INPUT SOME DATA
5 0009 LBL 'B' ;SECOND ENTRY POINT
```

53
This error was caused because when we tried (in line 0) to find the value of INPUT we did not yet know the value of OUTPUT. If we reverse the order of the equates the problem will go away. As a general rule of thumb, unless there is a particular need for their being elsewhere, PUT ALL EQUATES BEFORE ANY REFERENCES TO THEM!.

U - A symbol was Undefined. The usual causes of this are a typographical error or forgetting to include the referenced routine.

OTHER ERROR MESSAGES

Argument Error on Command Line - an illegal or improperly formed command line was entered.

A>41UCC I=FRED,Q,W=2.5.3

would produce this message, because 41UCC does not have a "Q" option, and 2.5.3 is not a valid file name.

Disk Error on Write to File - 41UCC encountered an error in trying to write a file to disk. This is a fatal error and will cause processing to stop immediately. Possible errors are disk or directory full, or a bad disk.

Include Error - the #include was improperly formed, or the file was not found on the specified drive. Includes may not have any comment on the line with them - this and not having the file on the disk will probably be your most common mistakes. You should be able to do a 'DIR' - a disk directory of the current drive - and see the name of the include file. If it is not on the current drive, then you need to copy it there in order for the #include to work.

Symbol Table Overflow - 41UCC's internal symbol table overflowed. Deleting unused string equates will be the quickest way to fix this, as string equates require a good bit of room in the symbol table. If you have included comments inside the quoted part of the string equate you might consider removing them. Specifying NX on the command line (no Cross Reference desired) will also decrease the amount of table space needed. If you continue to get this message, you need more memory in the system, or you need to break your program up into more manageable pieces.

**** SYSTEM FAILURE **** - This is the worst error message you can get. It goes on to say that you may have a bad computer (memory, cpu, who knows), or a bad copy of 41UCC, or you have found a really bad bug. If you can get it to fail in the same way on two separate systems you have probably found a bug or have a bad copy of 41UCC - go ahead and give us a call. 41UCC is
actually capable of catching many disk, memory, cpu, and author (meaning me) errors - many of the routines check their own input for validity, even though it was passed to them by another routine. Thus, if a bit gets changed, the chances are very good that it will make the input to some routine invalid, at which point everything will come to a screeching halt and you will get this message.
APPENDIX C - SYNTHETIC INSTRUCTIONS

Synthetic instructions are those instructions which are perfectly understood by the HP-41C, but which HP did not provide access to. Thus, they have been synthesized from other instructions hitherto. Now, you can enter them as easily as any other instruction through the use of 41UCC. 41UCC supports instructions using the following registers as operands:

\[ a, b, c, d, e \text{ and } M, N, O, P, Q, R \]

in addition to the standard registers 0-99 and stack. The "R" register corresponds to the one that prints as an append mark. Thus the following instructions are completely acceptable to 41UCC:

\[
\begin{align*}
\text{STO} & \quad A \\
\text{STO} & \quad a \quad \text{; GENERATES THE SAME CODE}
\end{align*}
\]

\[
\begin{align*}
\text{STO} & \quad M \\
\text{ISG} & \quad P
\end{align*}
\]

In addition to these extra registers, 41UCC allows operands ranging from 0 to 111 for those HP-41 operators which normally allow only 0 to 99 (i.e. STO, RCL, DSE) with the exception of LBL, GTO, and XEQ, which may have operands only in the range 0 to 99. Thus the following instructions are valid:

\[
\begin{align*}
\text{STO} & \quad 111 \\
\text{STO} & \quad \text{IND} \quad 110 \\
\text{DSE} & \quad 100
\end{align*}
\]

while these are invalid:

\[
\begin{align*}
\text{LBL} & \quad 100 \\
\text{GTO} & \quad 100 \\
\text{XEQ} & \quad 100
\end{align*}
\]

Those instructions which normally have a range less than 99 are limited to their normal range with 41UCC, with the exception of the TONE instruction, which is allowed to have an operand in the range 0 to 127. The following instructions are legal:

\[
\begin{align*}
\text{TONE} & \quad 66 \\
\text{FIX} & \quad 9 \\
\text{ENG} & \quad 3
\end{align*}
\]

while these are still illegal:

\[
\begin{align*}
\text{ENG} & \quad 10 \\
\text{SF} & \quad 30 \\
\text{FS?C} & \quad 60
\end{align*}
\]

The reasons for these non-restrictions and restrictions are sometimes complex, but simply put, I have tried to provide the maximum number of meaningful functions within the limitation of the HP-41C instruction set, the way it works in practice, the
utility of and need for certain extensions, and what HP can reasonably be expected to support. If you really need to generate a "FIX 11", probably the best way to do it is with a DB statement such as this:

```plaintext
EQU FIX 9CH ;"FIX" PREFIX
EQU FIX11 'DB FIX,11' ;GENERATES A TRUE FIX 11
LBL 'MYPROG'
FIX11 ;MY OWN PSEUDO-FUNCTION
END
```

**IMPORTANT NOTE:**

In order to protect those users who do not know synthetic programming, and have no desire to learn the hard way, USER EQUATES OVERRIDE SYNTHETIC FUNCTIONS! This means that the following program segment:

```plaintext
EQU M 01 ;MY VARIABLE
STO M
```

will produce a store into register 01 - not into the M register. In this program, there is now no way to reference the M register without resorting to a DB statement.
It is often the case that you already have a very nice program on an HP-41C and you would like to take this program, document it, edit it, run it through 41UCC, and then move it back into the calculator for testing. In order to do this you will probably need some knowledge of the hardware of your computer - enough at least to hook the HP 82166 HP-IL to parallel converter (or, when it becomes available, the RS-232 converter) to a port on your computer. For example, suppose your computer has a parallel port for the printer, and the CP/M RDR device (paper tape reader) is implemented so that it reads from the printer port. On many machines you will need to assign the port your 82166 is hooked to to the RDR device. For example, on the Osborne, if your 82166 were hooked to the Centronics port you would need to assign the RDR to this port by using STAT as follows:

A>STAT RDR:=UR1:

This tells STAT to assign User Reader 1 (the Centronics port on the Osborne) to the logical device RDR.

You would then type

A>PIP MYFILE.41C=RDR:

This tells PIP to read from the paper tape reader (on your system, the printer port) and send whatever it reads to MYFILE.41C. Then, on the calculator, you would

XEQ ALPHA MANIO ALPHA

to go to manual (rather than automatic) I/O, then you would address the 82166 as a listener. If it were the second device in the loop, addressing it as a listener would be

2 ENTER
XEQ ALPHA LISTEN ALPHA

Now to actually send a program to the computer we will "print" it - but the 82166 will copy the entire listing to the computer.

XEQ ALPHA PRP ALPHA

When PRP prompts for the file name, you would give it the name of the file you wish to modify. When the file is completely "printed", type

26 ENTER
XEQ ALPHA ACCHR ALPHA

which sends an end-of-file to PIP. PIP will then finish writing the file on the disk, and return to CP/M.
There is only one thing left - to convert the PRP listing to something 41UCC will understand. There is a special program called FILTER.COM to do this, and all we have to do is type

A>FILTER MYFILE

Filter will find MYFILE.41C on the disk and convert it to MYFILE.UCC, which 41UCC will be able to understand. (If you have used synthetics in your program FILTER will not do quite all of the work - you will still need to work on the synthetic lines by hand.)

Should you wish to you can use PIP to do other things with information from your HP-41. For example you could say

A>PIP CON:=RDR:

and everything that the calculator sent out on the loop would be displayed on the screen of your computer. Or you could say

A>PIP LST:=RDR:

and everything would be sent to your printer (or your modem if you used a funny cable to hook it to your printer port....). Don't hook a modem to your Osbornes' printer port though - it would destroy the port at least. Other computers do not have this problem.
APPENDIX E - PRINTING BARCODES

41UCC produces a .WND (WaND) file which contains all of the information needed to produce a barcode listing of your program. Of course, you need a printer capable of producing barcode in the first place. Some that are (and for which I have already written the programs) are the Epson MX-80/MX-100 (without Grafrax) and the Trilog. I have also tried to use a Microline (Okidata) 80a, but the positioning was very poor and produced unreadable barcodes. Included on your 41UCC disk are several programs for printing barcodes, including source to a simple version. If you have a printer that is not already supported, take a look at these files (or get a friend to if you do not know 8080 assembly language). It should not be too hard to figure out how to make your printer work (if it is possible at all). Any daisywheel printer should be able to do it (with the proper type wheel), and many matrix printers can print barcodes.

Using My Standard Barcode Printing Programs

If you can use one of the programs that I have already written, or if you have modified one of them to work with your printer, then they all work alike. Using PMX (for the MX-80/MX-100 without Grafrax) as an example, you could print FILTER.WND by typing

A>PMX I=FILTER

Now that isn't too difficult is it? If you wish to print several files at once, use it just like 41UCC

A>PMX
?I=FILTER
?I=SECANT
?I=HEX
?Z

This will tell PMX to print the files FILTER.WND, SECANT.WND, and HEX.WND as barcode.

Finally, you could put all of the lines that you typed in above into a file (which I will call BAR.IND):

A>TYPE BAR.IND
I=FILTER
I=SECANT
I=HEX
^Z
A>

Now to print all of these files as barcode, just type

A>PMX @BAR

and they will be printed one at a time. It works just like 41UCC!
Format of the WaND File

The WaND file contains all of the sequencing, data, and checksum information for the barcode - the only thing it does not contain is the header and trailer bars. Each row of barcode is required to have two zero bars at the beginning and a one zero at the end (so that the wand knows in which direction you are scanning). It is the responsibility of your barcode printing routine to add these bars. If you need more information on the contents of a row of barcode, I suggest that you read the HP publication "Creating Your Own Barcode."

A typical row of barcode might look in part like this

041000C000F3415343....

Now, assuming that your barcode printing program has read this in, you need to:

1) Print the left header bars - two bars of zero.
2) Strip off the top bit.
3) Convert each character in order into binary.
4) For each bit, print a zero bar if the bit is zero, or a one bar if the bit is a one. Thus, the first character in the line is a zero. Converting this to binary yields 0000, so we print 4 zero bars. The next character is a one, which yields 0001, so we print 3 zero bars and a one bar.
5) When you reach the carriage return, line feed at the end of the row you must print the trailer bars - a one bar followed by a zero bar.

Now go to a new line on your printer, read another row of barcode, and start the whole process over.

Getting Fancy

There are a couple of things which may be done to produce better or more useable barcode.

First, you could print line numbers for each row of barcode. This is very easy to do, and makes the barcode a bit easier to use because the wand prompts for the line number it wants next.

Second, you could print the listing line numbers above the corresponding row of barcodes. Thus if row 3 of the barcode produces lines 7 through 10 of the listing, you could print (7-10) above the row. This involves counting the high bits that are set in the WaND file - each high bit that is set marks the beginning of a new line of the listing. The high bits were set for just this purpose.

Third, you could print the same row of barcode more than once. If your printer allows you to roll the paper less than a full line you can print tall barcodes that are easier to read
than very short ones. This can also make the difference between having to use a straightedge and not.

**LOOKING FORWARD**

If you write your own barcode printing program or modify one of mine you should make a few allowances for future changes.

First, you should ignore all spaces and null bytes in the WaND file - some printers require a space on a line in order for the carriage return (or linefeed) to work, and therefore these spaces will show up in the WaND file. Also, some printers require a null after the linefeed, and the nulls will be in the WaND file also.

Second, you should ignore any line which begins (after discarding any spaces or nulls) with a '$'. I intend to use the $ in the future to mark such things as a title and comments to be printed with the barcode.

Your 41UCC distribution disk contains the source code to portions of the barcode printing programs that I have written. Feel free to modify these to work with your own printer; I put them there for that purpose. You may also give away copies of the barcode printing programs (NOT 41UCC!) freely. You may not sell the barcode printing programs even if you modify the printer drivers.

If you intend to do much programming in 8080 assembly language I strongly recommend that you take a look at the assembler, linker, and I/O library from Mycroft Labs in Tallahassee. I use their package for all of my work (including 41UCC and the barcode printing programs) and for large projects I think it is the best available. (Far better than RMAC or M80.) However, plain old ASM, which came on your CP/M disk, will do for modifying the barcode printing programs.
APPENDIX F - MODIFYING 41UCC

There are portions of 41UCC that are set up to make it easy for you to customize them. The file called 41UCX.ASM on your 41UCC distribution disk is the source code to certain pointers and parameters that you may change. In particular it contains a string to initialize and de-initialize your printer, a string to do a line feed and a form feed, and pointers to a user customization area and to the command line parameters. If you modify this file, reassemble it, and then merge it back into 41UCC with SID or DDT, you will have a customized version.

For example, suppose your printer (an MX-80) needs only a carriage return in order to go to the next line. You would modify PLF: from

```
PLF: DB 2,CR,LF ;PRINTER STRING TO DO A LINE FEED
DB 0
DB 0,0,0,0
```

to

```
PLF: DB 1,CR ;PRINTER STRING TO DO A LINE FEED
DB 0,0
DB 0,0,0,0
```

and your MX-80 would now work correctly with 41UCC. The first byte of each of the strings is the number of bytes in the string. Note that you cannot use more bytes than I made available! Thus you may use at most 8 bytes for the line feed string.

Now reassemble this program

```
A>ASM 41UCX
.
.
.
```

and merge it with 41UCC

```
A>DDT 41UCC.COM
.
.
.
-141UCX.COM
-R
.
.
-*C
A>SAVE 120 41UCCX.COM
```

(this saves the result as 41UCCX.COM). Now test out your new version, make any more changes you wish, and you have your customized version. Be certain to keep an original copy of 41UCC so that you can undo any damage you cause to your working copy!
APPENDIX F - MODIFYING 41UCC

If you need to make more involved changes, such as having 41UCC produce no WaND file as the default you will need to use the EXTRA1 area that is set aside for such things. You may use EXTRA1, which contains 128 bytes, for anything you wish. EXTRA2 is reserved for my use - bug fixes and things like that. 41UCC contains a pointer to this extra segment. You can find this pointer by looking at the first three bytes of the program. This is a jump to START which then jumps around the pointers, jump vectors, constants, and strings.

In order to have no WaND file produced (by default) you would need to use JV3 - the jump vector called immediately after the command line. Change this to point to your routine in EXTRA1. The routine in EXTRA1 should set NW+3 to a non-zero value. This will tell 41UCC that the user put the NW option on the command line, and no WaND file will be produced. If you later wish to get a WaND file all you need to do is specify W=A: (or something similar) on the command line. This will override the NW option.

The jump vectors JV1 through JV4 are entirely yours to use as you wish.
APPENDIX G - PROGRAM LISTINGS AND BARCODE

41UCC V 1.45, Copyright 1981 by Leslie Brooks.
Distributed by Hand Held Products Incorporated.

0000; TEST 28 OF THE HP-41C CROSS ASSEMBLER
0000; FULL BLOWN TEST - APRIL 9, 1982
0000; This program is a test of 41UCC.
0000; It contains every legal 41UCC instruction in several possible forms.
0000; It also contains most of the allowed pseudo-ops.
0000; All of the instructions are in alphabetical order, so it provides a handy
0000; reference.
0000; IF YOUR COPY OF 41UCC CANNOT COMPILE THIS PROGRAM WITHOUT ERRORS YOU HAVE A
0000; PROBLEM! YOU COULD HAVE A BAD COPY, A BAD DISK, OR A BAD MACHINE.
0000
0000 EQU SCRATCH 05
0000 EQU MATRIX '30,01' ; HP MATH PAC MATRIX ROUTINE
0000
0000 LBL 'TST28'
54535432
0000 30
0009 4C
000A 4D
000B 42
000C 40
000D 41
000E 43
000F 60
0010 57
0011 57
0012 61
0013 5D
0014 8F
0015 88
0016 8C
0017 F2F41
0018 F7F3031
0019 F2F41
0020 F2F41
0021 9873
0022 9800
0023 9814
0024 9863
0025 9878
0026 9880
0027 9885
0028 9894
0029 9BE3
002A 9BF3
002B 9BFB
002C 88
002D 88
002E 88
002F 88
0030 9873
0031 9800
0032 9814
0033 9863
0034 9878
0035 9880
0036 9885
0037 9894
0038 9BE3
0039 9BF3
003A 9BFB
003B 88
003C 5C
003D 88
003E 88
003F 88
0040 9873
0041 9800
0042 9814
0043 9863
0044 9878
0045 9880
0046 9885
0047 9894
0048 9BE3
0049 9BF3
004A 9BFB
004B 88
004C 5C
004D 88
004E 88
004F 88
0050 9873
0051 9800
0052 9814
0053 9863
0054 9878
0055 9880
0056 9885
0057 9894
0058 9BE3
0059 9BF3
005A 9BFB
005B 88
005C 5C
005D 88
005E 88
005F 88
0060 9873
0061 9800
0062 9814
0063 9863
0064 9878
0065 9880
0066 9885
0067 9894
0068 9BE3
0069 9BF3
006A 9BFB
006B 88
006C 5C
006D 88
006E 88
006F 88
0070 9873
0071 9800
0072 9814
0073 9863
0074 9878
0075 9880
0076 9885
0077 9894
0078 9BE3
0079 9BF3
007A 9BFB
007B 88
007C 5C
007D 88
007E 88
007F 88
0080 9873
0081 9800
0082 9814
0083 9863
0084 9878
0085 9880
0086 9885
0087 9894
0088 9BE3
0089 9BF3
008A 9BFB
008B 88
008C 5C
008D 88
008E 88
008F 88
0090 9873
0091 9800
0092 9814
0093 9863
0094 9878
0095 9880
0096 9885
0097 9894
0098 9BE3
0099 9BF3
009A 9BFB
009B 88
009C 5C
009D 88
009E 88
009F 88
00A0 9873
00A1 9800
00A2 9814
00A3 9863
00A4 9878
00A5 9880
00A6 9885
00A7 9894
00A8 9BE3
00A9 9BF3
00AA 9BFB
00AB 88
00AC 5C
00AD 88
00AE 88
00AF 88
00B0 9873
00B1 9800
00B2 9814
00B3 9863
00B4 9878
00B5 9880
00B6 9885
00B7 9894
00B8 9BE3
00B9 9BF3
00BA 9BFB
00BB 88
00BC 5C
00BD 88
00BE 88
00BF 88
00C0 9873
00C1 9800
00C2 9814
00C3 9863
00C4 9878
00C5 9880
00C6 9885
00C7 9894
00C8 9BE3
00C9 9BF3
00CA 9BFB
00CB 88
49 004C 9A63  ASTO  99
50 004E 9A73  ASTO  X
51 0050 9A7C  ASTO  8
52 0052 9A80  ASTO  IND  00
53 0054 9A85  ASTO  IND  SCRATCH
54 0056 9A94  ASTO  IND  20
55 0058 9A9E  ASTO  IND  99
56 005A 9AF3  ASTO  IND  X
57 005C 9AFC  ASTO  IND  A
58 005E 5E  ATAN
59 005F 7E  APIVIEW
60 0060 86  BEEP
61 0061 A900  CF  00
62 0063 A91D  CF  29
63 0065 A9F3  CF  IND  X
64 0067 A9E3  CF  IND  99
65 0069 A980  CF  IND  00
66 006B A985  CF  IND  SCRATCH
67 006D A994  CF  IND  20
68 006F A9F9  CF  IND  Q
69 0071 54  CHS
70 0072 87  CLA
71 0073 87  CLD
72 0074 8A  CLR6
73 0075 70  CLS
74 0076 70  CLSIGMA
75 0077 73  CLST
76 0078 77  CLX
77 0079 5A  COS
78 007A 6A  D-R
79 007B 4B454C4C  DB  'HELLO WORLD',25+03H+000100008F20574F
                      524C442C
80 0087 444F4E27  DB  "DON'T GO AWAY" ;DOUBLE QUOTES ARE ALLOWED
                             5420474F
                      20415741
81 008F 5F  DEC
82 0095 80  DE6
83 0096 9700  DSE  00
84 0098 9714  DSE  20
85 009A 9763  DSE  99
86 009C 9773  DSE  X
87 009E 9775  DSE  M
88 00A0 9780  DSE  IND  00
89 00A2 9785  DSE  IND  SCRATCH
90 00A4 9794  DSE  IND  20
91 00A6 97E3  DSE  IND  99
92 00AA 97F3  DSE  IND  X
93 00AB 97F5  DSE  IND  M
94 00AC 55  E**X
95 00AB 55  E^X
147 0117 82  GRAD
148 0118 B100  GTO 00
149 011A D00063  GTO 99
150 011D AE00  GTO IN 00
151 011F AE14  GTO IN 20
152 0121 AE63  GTO IN 99
153 0123 AE73  GTO IN X
154 0125 AE78  GTO IN P
155 0127 AE7F  GTO IN E
156 0129 1DF4652 GOTO 'FRED'

4544
157 012F 1DF4652 GOTO 'FREQUENCY'
45515545
4E
158 0138 D00066  GOTO 'A'
159 013B B100  GTO 00
160 013D D00063  GTO 99
161 0140 AE00  GTO IN 00
162 0142 AE14  GTO IN 20
163 0144 AE63  GTO IN 99
164 0146 AE73  GTO IN X
165 0148 AE78  GTO IN P
166 014A AE7F  GTO IN E
167 014C 1DF4652 GTO 'FRED'
4544
168 0152 1DF4652 GOTO 'FREQUENCY'
45515545
4E
169 015B D00066  GTO 'A'
170 015E 1DF4652 GTO6 'FRED'
4544
171 0164 1DF411  GTO6 'A'
172 0167 6C  HMS
173 0168 49  HMS+
174 0169 4A  HMS-
175 016A 6D  HR
176 016B 68  INT
177 016C 9600  ISG 00
178 016E 9614  ISG 20
179 0170 9663  ISG 99
180 0172 9673  ISG X
181 0174 967F  ISG E
182 0176 9680  ISG IN 0
183 0178 9694  ISG IN 20
184 017A 96E3  ISG IN 99
185 017C 96F3  ISG IN X
186 017E 96F9  ISG IN Q
187 0180 76  LASTX
188 0181 01  LBL 00
189 0182 CF63  LBL 99
190 0184 CF66  LBL 'A'
191 0186 CF78  LBL 'a'
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>0188 CF7F</td>
<td>LBL &quot;e&quot;</td>
</tr>
<tr>
<td>193</td>
<td>018A C000F000</td>
<td>LBL &quot;FREQUENCY&quot;</td>
</tr>
<tr>
<td></td>
<td>46524551</td>
<td>55454E</td>
</tr>
<tr>
<td>194</td>
<td>0195 56</td>
<td>LOG</td>
</tr>
<tr>
<td>195</td>
<td>0196 50</td>
<td>LN</td>
</tr>
<tr>
<td>196</td>
<td>0197 65</td>
<td>LN(1+x)</td>
</tr>
<tr>
<td>197</td>
<td>0198 7C</td>
<td>MEAN</td>
</tr>
<tr>
<td>198</td>
<td>0199 4B</td>
<td>MOD</td>
</tr>
<tr>
<td>199</td>
<td>019A 6F</td>
<td>OCT</td>
</tr>
<tr>
<td>200</td>
<td>019B 8D</td>
<td>OFF</td>
</tr>
<tr>
<td>201</td>
<td>019C 4E</td>
<td>P-&gt;R</td>
</tr>
<tr>
<td>202</td>
<td>019D 4E</td>
<td>P-R</td>
</tr>
<tr>
<td>203</td>
<td>019E T2</td>
<td>PI</td>
</tr>
<tr>
<td>204</td>
<td>019F 8E</td>
<td>PROMPT</td>
</tr>
<tr>
<td>205</td>
<td>01A0 89</td>
<td>PSE</td>
</tr>
<tr>
<td>206</td>
<td>01A1 6B</td>
<td>R-D</td>
</tr>
<tr>
<td>207</td>
<td>01A2 4F</td>
<td>R-&gt;P</td>
</tr>
<tr>
<td>208</td>
<td>01A3 4F</td>
<td>R-P</td>
</tr>
<tr>
<td>209</td>
<td>01A4 81</td>
<td>RAD</td>
</tr>
<tr>
<td>210</td>
<td>01A5 20</td>
<td>RCL 00</td>
</tr>
<tr>
<td>211</td>
<td>01A6 2F</td>
<td>RCL 15</td>
</tr>
<tr>
<td>212</td>
<td>01A7 9010</td>
<td>RCL 16</td>
</tr>
<tr>
<td>213</td>
<td>01A8 9014</td>
<td>RCL 20</td>
</tr>
<tr>
<td>214</td>
<td>01A9 9063</td>
<td>RCL 99</td>
</tr>
<tr>
<td>215</td>
<td>01AA 9073</td>
<td>RCL X</td>
</tr>
<tr>
<td>216</td>
<td>01AB 907C</td>
<td>RCL B</td>
</tr>
<tr>
<td>217</td>
<td>01B1 9080</td>
<td>RCL IND 00</td>
</tr>
<tr>
<td>218</td>
<td>01B2 9094</td>
<td>RCL IND 20</td>
</tr>
<tr>
<td>219</td>
<td>01B3 90E3</td>
<td>RCL IND 99</td>
</tr>
<tr>
<td>220</td>
<td>01B4 90F3</td>
<td>RCL IND X</td>
</tr>
<tr>
<td>221</td>
<td>01B5 90F9</td>
<td>RCL IND Q</td>
</tr>
<tr>
<td>222</td>
<td>01B6 75</td>
<td>RDN</td>
</tr>
<tr>
<td>223</td>
<td>01B7 6E</td>
<td>RMD</td>
</tr>
<tr>
<td>224</td>
<td>01B8 05</td>
<td>RTN</td>
</tr>
<tr>
<td>225</td>
<td>01B9 74</td>
<td>R^</td>
</tr>
<tr>
<td>226</td>
<td>01B9 47</td>
<td>S+</td>
</tr>
<tr>
<td>227</td>
<td>01CA 48</td>
<td>S-</td>
</tr>
<tr>
<td>228</td>
<td>01CC 9D00</td>
<td>SCI 00</td>
</tr>
<tr>
<td>229</td>
<td>01CD 9D09</td>
<td>SCI 9</td>
</tr>
<tr>
<td>230</td>
<td>01CE 9DF3</td>
<td>SCI IND X</td>
</tr>
<tr>
<td>231</td>
<td>01CF 9D80</td>
<td>SCI IND 00</td>
</tr>
<tr>
<td>232</td>
<td>01D0 9D94</td>
<td>SCI IND 20</td>
</tr>
<tr>
<td>233</td>
<td>01D1 9DE3</td>
<td>SCI IND 99</td>
</tr>
<tr>
<td>234</td>
<td>01D2 9DF0</td>
<td>SCI IND C</td>
</tr>
<tr>
<td>235</td>
<td>01D3 7D</td>
<td>SDEV</td>
</tr>
<tr>
<td>236</td>
<td>01D4 A800</td>
<td>SF 00</td>
</tr>
<tr>
<td>237</td>
<td>01D5 A814</td>
<td>SF 20</td>
</tr>
<tr>
<td>238</td>
<td>01D6 A81D</td>
<td>SF 29</td>
</tr>
<tr>
<td>239</td>
<td>01D7 A880</td>
<td>SF IND 00</td>
</tr>
<tr>
<td>240</td>
<td>01D8 A894</td>
<td>SF IND 20</td>
</tr>
<tr>
<td>241</td>
<td>01D9 A8E3</td>
<td>SF IND 99</td>
</tr>
<tr>
<td>242</td>
<td>01DA ABF3</td>
<td>SF IND X</td>
</tr>
</tbody>
</table>
243 01DE ABFE SF IND D
244 01E0 ABF4 SF IND L
245 01E2 47 SIGMA+
246 01E3 48 SIGMA-
247 01E4 9900 SIGNAREG 00
248 01E6 TA SIGN
249 01E7 59 SIN
250 01E8 52 SQRT
251 01E9 9900 SREG 00
252 01EB 9914 SREG 20
253 01ED 9963 SREG 99
254 01EF 9980 SREG IND 00
255 01F1 9994 SREG IND 20
256 01F3 99E3 SREG IND 99
257 01F5 99F3 SREG IND X
258 01F7 99F1 SREG IND Z
259 01F9 99F8 SREG IND P
260 01FB 9400 STX 00
261 01FD 9414 STX 20
262 01FF 9463 STX 99
263 0201 9480 STX IND 00
264 0203 9494 STX IND 20
265 0205 94E3 STX IND 99
266 0207 94F3 STX IND X
267 0209 94F2 STX IND Y
268 020B 9200 ST+ 00
269 020D 9214 ST+ 20
270 020F 9263 ST+ 99
271 0211 9273 ST+ X
272 0213 9280 ST+ IND 00
273 0215 9294 ST+ IND 20
274 0217 92E3 ST+ IND 99
275 0219 92F3 ST+ IND X
276 021B 92FB ST+ IND A
277 021D 9300 STO- 00
278 021F 9314 STO- 20
279 0221 9363 STO- 99
280 0223 9380 STO- IND 00
281 0225 9394 STO- IND 20
282 0227 93E3 STO- IND 99
283 0229 93F3 STO- IND X
284 022B 93FF STO- IND E
285 022D 9500 STO/ 00
286 022F 9514 STO/ 20
287 0231 9563 STO/ 99
288 0233 9580 STO/ IND 00
289 0235 9594 STO/ IND 20
290 0237 95E3 STO/ IND 99
291 0239 95F3 STO/ IND X
292 023B 30 STO 00
293 023C 3E STO 14
294 023D 3F STO 15
295 023E 9110 STO 16
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Target Code</th>
<th>Target Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>296</td>
<td>0240</td>
<td>9163</td>
<td>STO 99</td>
</tr>
<tr>
<td>297</td>
<td>0242</td>
<td>9180</td>
<td>STO IND 00</td>
</tr>
<tr>
<td>298</td>
<td>0244</td>
<td>910F</td>
<td>STO IND 15</td>
</tr>
<tr>
<td>299</td>
<td>0246</td>
<td>91E3</td>
<td>STO IND 99</td>
</tr>
<tr>
<td>300</td>
<td>0248</td>
<td>91FC</td>
<td>STO IND B</td>
</tr>
<tr>
<td>301</td>
<td>024A</td>
<td>84</td>
<td>STOP</td>
</tr>
<tr>
<td>302</td>
<td>0248</td>
<td>'F5592741'</td>
<td>T &quot;HELLO&quot;</td>
</tr>
<tr>
<td>303</td>
<td>0251</td>
<td>'F5592741'</td>
<td>T &quot;Y'ALL&quot;</td>
</tr>
<tr>
<td>304</td>
<td>0257</td>
<td>'F5592741'</td>
<td>T &quot;Y''ALL&quot;</td>
</tr>
<tr>
<td>305</td>
<td>025D</td>
<td>5B</td>
<td>TAN</td>
</tr>
<tr>
<td>306</td>
<td>025E</td>
<td>9F00</td>
<td>TONE 0</td>
</tr>
<tr>
<td>307</td>
<td>0260</td>
<td>9F09</td>
<td>TONE 9</td>
</tr>
<tr>
<td>308</td>
<td>0262</td>
<td>9FFF</td>
<td>TONE 127</td>
</tr>
<tr>
<td>309</td>
<td>0264</td>
<td>9F80</td>
<td>TONE IND 00</td>
</tr>
<tr>
<td>310</td>
<td>0266</td>
<td>9F94</td>
<td>TONE IND 20</td>
</tr>
<tr>
<td>311</td>
<td>0268</td>
<td>9FE3</td>
<td>TONE IND 99</td>
</tr>
<tr>
<td>312</td>
<td>026A</td>
<td>9FF3</td>
<td>TONE IND X</td>
</tr>
<tr>
<td>313</td>
<td>026C</td>
<td>9800</td>
<td>VIEW 00</td>
</tr>
<tr>
<td>314</td>
<td>026E</td>
<td>9814</td>
<td>VIEW 20</td>
</tr>
<tr>
<td>315</td>
<td>0270</td>
<td>9863</td>
<td>VIEW 99</td>
</tr>
<tr>
<td>316</td>
<td>0272</td>
<td>9873</td>
<td>VIEW X</td>
</tr>
<tr>
<td>317</td>
<td>0274</td>
<td>9880</td>
<td>VIEW IND 00</td>
</tr>
<tr>
<td>318</td>
<td>0276</td>
<td>9894</td>
<td>VIEW IND 20</td>
</tr>
<tr>
<td>319</td>
<td>0278</td>
<td>98E3</td>
<td>VIEW IND 99</td>
</tr>
<tr>
<td>320</td>
<td>027A</td>
<td>98F3</td>
<td>VIEW IND X</td>
</tr>
<tr>
<td>321</td>
<td>027C</td>
<td>98F0</td>
<td>VIEW IND T</td>
</tr>
<tr>
<td>322</td>
<td>027E</td>
<td>63</td>
<td>X!=:0?</td>
</tr>
<tr>
<td>323</td>
<td>027F</td>
<td>63</td>
<td>X:&lt;:0?</td>
</tr>
<tr>
<td>324</td>
<td>0280</td>
<td>79</td>
<td>X!=:Y?</td>
</tr>
<tr>
<td>325</td>
<td>0281</td>
<td>79</td>
<td>X:&lt;:Y?</td>
</tr>
<tr>
<td>326</td>
<td>0282</td>
<td>51</td>
<td>X&lt;&lt;:2</td>
</tr>
<tr>
<td>327</td>
<td>0283</td>
<td>66</td>
<td>X:&lt;:0?</td>
</tr>
<tr>
<td>328</td>
<td>0284</td>
<td>78</td>
<td>X:&lt;:0?</td>
</tr>
<tr>
<td>329</td>
<td>0285</td>
<td>46</td>
<td>X:&lt;:Y?</td>
</tr>
<tr>
<td>330</td>
<td>0286</td>
<td>CE00</td>
<td>X&lt;&gt; 00</td>
</tr>
<tr>
<td>331</td>
<td>0288</td>
<td>CE14</td>
<td>X&lt;&gt; 20</td>
</tr>
<tr>
<td>332</td>
<td>028A</td>
<td>CE63</td>
<td>X&lt;&gt; 99</td>
</tr>
<tr>
<td>333</td>
<td>028C</td>
<td>CE70</td>
<td>X&lt;&gt; T</td>
</tr>
<tr>
<td>334</td>
<td>028E</td>
<td>CE80</td>
<td>X&lt;&gt; IND 00</td>
</tr>
<tr>
<td>335</td>
<td>0290</td>
<td>CE94</td>
<td>X&lt;&gt; IND 20</td>
</tr>
<tr>
<td>336</td>
<td>0292</td>
<td>CEE3</td>
<td>X&lt;&gt; IND 99</td>
</tr>
<tr>
<td>337</td>
<td>0294</td>
<td>CEF3</td>
<td>X&lt;&gt; IND X</td>
</tr>
<tr>
<td>338</td>
<td>0296</td>
<td>CEF5</td>
<td>X&lt;&gt; IND M</td>
</tr>
<tr>
<td>339</td>
<td>0298</td>
<td>71</td>
<td>X&lt;&gt;Y</td>
</tr>
<tr>
<td>340</td>
<td>0299</td>
<td>44</td>
<td>X&lt;Y?</td>
</tr>
<tr>
<td>341</td>
<td>029A</td>
<td>67</td>
<td>X:=0?</td>
</tr>
<tr>
<td>342</td>
<td>029B</td>
<td>78</td>
<td>X:=Y?</td>
</tr>
<tr>
<td>343</td>
<td>029C</td>
<td>64</td>
<td>X:&gt;0?</td>
</tr>
<tr>
<td>344</td>
<td>029D</td>
<td>45</td>
<td>X:&gt;Y?</td>
</tr>
<tr>
<td>345</td>
<td>029E</td>
<td>E00000</td>
<td>XEQ 00</td>
</tr>
</tbody>
</table>
346 02A1 E00063 XEQ 99
347 02A4 E00066 XEQ "A"
348 02A7 E0007B XEQ "a"
349 02A8 E0007F XEQ "e"
350 02AD AE80 XEQ IND 00
351 02AF AE94 XEQ IND 20
352 02B1 AEE3 XEQ IND 99
353 02B3 AEF3 XEQ IND X
354 02B5 E00066 XEQ "A"
355 02B8 1EF44652 XEQ "FRED"
4544
356 02BE 1EF44652 XEQ "FREQUENCY"
45515545
4E
357 02C7 1EF141 XEQG "A"
358 02CA 1EF44652 XEQG "FRED"
4544
359 02D0 A782 XROM 30,02
360 02D2 A781 XROM MATRIX
361 02D4 51 X^2
362 02D5 53 Y^X
363 02D6 53 Y^X
364 02D7 C0000D END
**UNDEFINED ALPHA LABELS**
(These are errors if not defined in another program)

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYmBol NAME</th>
<th>ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**** NO SYMBOLS WERE UNDEFINED ****

**FLAG USAGE SUMMARY**

<table>
<thead>
<tr>
<th>FLAG #</th>
<th>LINE NUMBERS OF REFERENCES TO THE FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>61-CF 107-FC 116-TC 131-FS 138-TC 236-SF</td>
</tr>
<tr>
<td>009</td>
<td>108-FC</td>
</tr>
<tr>
<td>015</td>
<td>109-FC</td>
</tr>
<tr>
<td>020</td>
<td>237-SF</td>
</tr>
<tr>
<td>029</td>
<td>62-CF 117-TC 139-TC 238-SF</td>
</tr>
<tr>
<td>055</td>
<td>110-FC 132-FS</td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- CF: CLEAR FLAG
- FS: FLAG SET?
- FC: FLAG CLEAR?
- SF: SET FLAG
- TC: FLAG TEST AND CLEAR

**NUMERIC LABEL USAGE SUMMARY**

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>108</td>
<td>148-6 159-6 345-X</td>
</tr>
<tr>
<td>099</td>
<td>109</td>
<td>149-6 160-6 346-X</td>
</tr>
<tr>
<td>$</td>
<td>190</td>
<td>158-6 169-6 347-X 354-X</td>
</tr>
<tr>
<td>$</td>
<td>191</td>
<td>348-X</td>
</tr>
<tr>
<td>$</td>
<td>192</td>
<td>349-X</td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- 6: GOTO
- X: EXECUTE
REGISTER USAGE SUMMARY

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>LINE NUMBERS OF REFERENCES TO THE REGISTER</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>35-R 39-RI 47-S 52-SI 65-CF 83-DS 88-DI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>114-FC 120-TC 135-FS 142-TC 150-6I 161-6I 177-IS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>182-II 210-R 217-RI 239-SF 260-S 263-SI 268-S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>330-XC 334-XC 350-XI</td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>40-RI 53-SI 66-CF 89-DI</td>
<td></td>
</tr>
<tr>
<td>014</td>
<td>293-S</td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>211-R 294-S 298-SI</td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>212-R 295-S</td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>36-R 41-RI 49-S 54-SI 67-CF 84-DS 90-DI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>115-FC 121-TC 136-FS 143-TC 151-6I 162-6I 178-IS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>183-II 213-R 218-RI 240-SF 261-S 264-SI 269-S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>273-SI 278-S 281-SI 286-S 289-SI 331-XC 335-XC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>351-XI</td>
<td></td>
</tr>
<tr>
<td>099</td>
<td>37-R 42-RI 49-S 55-SI 64-CF 85-DS 91-DI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>114-II 122-TC 137-FS 144-TC 152-6I 163-6I 179-IS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>184-II 214-R 219-RI 241-SF 262-S 265-SI 270-S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>274-SI 279-S 282-SI 287-S 290-SI 296-S 299-SI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>332-XC 336-XC 352-XI</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>333-XC</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>267-SI</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>34-R 43-RI 50-S 56-SI 63-CF 86-DS 92-DI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111-FC 118-TC 133-FS 140-TC 153-6I 164-6I 180-IS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>185-II 215-R 220-RI 242-SF 266-SI 271-S 275-SI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>283-SI 291-SI 337-XC 353-XI</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>244-SF</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>87-DS 93-DI 112-FC 119-TC 338-XC</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>134-FS</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>141-TC</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>154-6I 165-6I</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>68-CF 186-II 221-RI</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>38-R 44-RI 57-SI 276-SI</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>51-S 216-R 300-SI</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>243-SF</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>155-6I 166-6I 181-IS 284-SI</td>
<td></td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:

<table>
<thead>
<tr>
<th>CF</th>
<th>CLEAR FLAG INDIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI</td>
<td>DECREMENT INDIRECT AND SKIP IF EQUAL</td>
</tr>
<tr>
<td>DS</td>
<td>DECREMENT AND SKIP IF EQUAL</td>
</tr>
<tr>
<td>FC</td>
<td>FLAG CLEAR? INDIRECT</td>
</tr>
<tr>
<td>FS</td>
<td>FLAG SET? INDIRECT</td>
</tr>
<tr>
<td>GI</td>
<td>GOTO INDIRECT</td>
</tr>
<tr>
<td>II</td>
<td>INCREMENT INDIRECT AND SKIP IF GREATER</td>
</tr>
<tr>
<td>IS</td>
<td>INCREMENT AND SKIP IF GREATER</td>
</tr>
<tr>
<td>R</td>
<td>RECALL</td>
</tr>
<tr>
<td>RI</td>
<td>RECALL INDIRECT</td>
</tr>
<tr>
<td>S</td>
<td>STORE</td>
</tr>
<tr>
<td>SF</td>
<td>SET FLAG INDIRECT</td>
</tr>
<tr>
<td>SI</td>
<td>STORE INDIRECT</td>
</tr>
<tr>
<td>TC</td>
<td>FLAG TEST AND CLEAR INDIRECT</td>
</tr>
<tr>
<td>XI</td>
<td>EXECUTE INDIRECT</td>
</tr>
<tr>
<td>XC</td>
<td>EXCHANGE X AND R</td>
</tr>
</tbody>
</table>
### ALPHA LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>145 010A</td>
<td>171-6 357-X</td>
</tr>
<tr>
<td>FRED</td>
<td>146 010F</td>
<td>156-6 167-6 170-6 355-X 358-X</td>
</tr>
<tr>
<td>FREQUEN</td>
<td>193 010A</td>
<td>157-6 168-6 356-X</td>
</tr>
<tr>
<td>TST28</td>
<td>15 0000</td>
<td></td>
</tr>
</tbody>
</table>

Tag meanings are: 
G GOTO  
X EXECUTE

### INTEGER SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL NAME ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>SCRATCH</td>
<td>12 0005</td>
<td>40- 53- 66- 89-</td>
</tr>
</tbody>
</table>

### STRING SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL NAME ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATRIX</td>
<td>13 &quot;30,01&quot;</td>
<td>360-</td>
</tr>
</tbody>
</table>

### VARIABLE USAGE SUMMARY

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE VARIABLE NAME ON</th>
</tr>
</thead>
</table>

***** NO VARIABLES WERE USED *****
SLOU-PASS FILTER PROGRAM

;WRITTEN BY LESLIE BROOKS

;NOVEMBER 17, 1982.

;SPECIAL EQUATES

EQU TEXT4 0F4H ;APPEND TEXT STRING OF THREE CHRS.
EQU APPEND 07FH ;THE APPEND FUNCTION
EQU NU 12 ;GREEK LETTER NU

;REGISTER EQUATES

EQU BASE R1 ;USE R1 FOR THE BASE REGISTER
EQU FREQUENCY BASE+0 ;THE CUTOFF FREQUENCY
EQU RESISTANCE BASE+1 ;THE TERMINATING RESISTANCE
EQU CAPACITOR BASE+2 ;CAPACITOR VALUE
EQU INDUCTOR BASE+3 ;COIL VALUE

CO00F800 LBL "LOWPASS" ;OUR LOW-PASS FILTER PROGRAM
4C4F5750
415353
FB465245 T "FREQUENCY=?" ;ASK FOR THE CUTOFF FREQUENCY
5155454t
4359303F
8E PRONT
30 STO FREQUENCY ;SAVE IT
F9522854 T "R(TERM)=??" ;ASK FOR THE TERMINATING RESISTANCE
45524029
3D3F
27 0023 8E PROMPT
28 0024 31 STO RESISTANCE ;SAVE THE RESISTANCE
29 0025 ;
30 0025 ;CALCULATE THE INDUCTOR (COIL) FIRST
31 0025 ; L = R/(PI * FREQUENCY)
32 0025 ;
33 0025 20 RCL FREQUENCY ;GET THE FREQUENCY
34 0026 T2 PI
35 0027 42 * ;PI * FREQUENCY
36 0028 43 / ;DIVIDE INTO THE RESISTANCE
37 0029 11101010 1000 ;CONVERT TO MILLIHENRIES
38 002D 43 /
39 002E 33 STO INDUCTOR ;SAVE IT FOR FUTURE USE
40 002F ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
41 002F F34C3D20 T "L= "
42 0033 9B73 ARCL X
43 0035 F47F2040 APPEND " MH" ;UNITS ARE MILLIHENRIES
44 003A 7E AVIEW
45 003B 89 PSE
46 003C 33 STO INDUCTOR ;SAVE IT FOR FUTURE USE
47 0030 ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
48 0030 F34C3D20 T 'L = '
49 0041 9B73 ARCL X
50 0043 7E AVIEW
51 0044 89 PSE
52 0045 ;
53 0045 ;NOW CALCULATE THE CAPACITOR
54 0045 ; C = 1 / (2 * PI * RESISTANCE * FREQUENCY)
55 0045 ;
56 0045 21 RCL RESISTANCE ; GET THE RESISTANCE AGAIN
57 0046 20 RCL FREQUENCY ; AND THE FREQUENCY
58 0047 42 *
59 0048 72 PI
60 0049 42 *
61 004A 9273 STO+ X ; DOUBLE IT
62 004C 11B16 1E6 ; CONVERT TO MICROFARADS
63 004F 43 /
64 0050 32 STO CAPACITOR ; SAVE FOR FUTURE USE
65 0051 ;NOW DISPLAY THE CAPACITOR VALUE
66 0051 F343D20 T 'C = '
67 0055 9B73 ARCL X
68 0057 F47F200C DB TEXT4,APPEND,' ','MU','F' ; LABEL IT AS MICROFARADS
69 005C 7E AVIEW
70 005D 89 PSE
71 005E C00000 END
### UNDEFINED ALPHA LABELS

(These are errors if not defined in another program)

<table>
<thead>
<tr>
<th>Label Name</th>
<th>Defined Value</th>
<th>Line Numbers of References to the Symbol</th>
</tr>
</thead>
</table>

***** NO SYMBOLS WERE UNDEFINED *****

---

### FLAG USAGE SUMMARY

Flag # Line Numbers of References to the Flag

***** NO FLAGS WERE USED *****

---

### NUMERIC LABEL USAGE SUMMARY

Label # Defined Line Numbers of References to the Label

***** NO NUMERIC LABELS WERE USED *****
REGISTER USAGE SUMMARY

REGISTER   LINE NUMBERS OF REFERENCES TO THE REGISTER  

000  25-S  33-R  57-R
001  28-S  56-R
002  64-S
003  39-S  46-S  X  42-R  49-R  61-S  67-R

TAG MEANINGS ARE:  
CF  CLEAR FLAG INDIRECT
DI  DECREMENT INDIRECT AND SKIP IF EQUAL
DS  DECREMENT AND SKIP IF EQUAL
FC  FLAG CLEAR? INDIRECT
FS  FLAG SET? INDIRECT
GI  GOTO INDIRECT
II  INCREMENT INDIRECT AND SKIP IF GREATER
IS  INCREMENT AND SKIP IF GREATER
R  RECALL
RI  RECALL INDIRECT
S  STORE
SF  SET FLAG INDIRECT
SI  STORE INDIRECT
TC  FLAG TEST AND CLEAR INDIRECT
XI  EXECUTE INDIRECT
XC  EXCHANGE X AND R

ALPHA LABEL USAGE SUMMARY

LABEL  DEFINED VALUE  LINE NUMBERS OF REFERENCES TO THE LABEL ON

LOWPASS  22  0000

TAG MEANINGS ARE:  
G  GOTO
X  EXECUTE
## INTEGER SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
<td>10 007F</td>
<td>68-</td>
</tr>
<tr>
<td>BASE</td>
<td>15 0000</td>
<td></td>
</tr>
<tr>
<td>CAPACIT</td>
<td>19 0002</td>
<td>64-</td>
</tr>
<tr>
<td>FREQUEN</td>
<td>17 0000</td>
<td>25- 33- 57-</td>
</tr>
<tr>
<td>INDUCTO</td>
<td>20 0003</td>
<td>39- 46-</td>
</tr>
<tr>
<td>MU</td>
<td>11 000C</td>
<td>68-</td>
</tr>
<tr>
<td>R1</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>1 0000</td>
<td></td>
</tr>
<tr>
<td>RESISTA</td>
<td>10 0001</td>
<td>28- 56-</td>
</tr>
<tr>
<td>TEXT4</td>
<td>9 00F4</td>
<td>68-</td>
</tr>
</tbody>
</table>

## STRING SYMBOL USAGE SUMMARY

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
</table>

**** NO STRING SYMBOLS WERE USED ****

## VARIABLE USAGE SUMMARY

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE VARIABLE</th>
</tr>
</thead>
</table>

**** NO VARIABLES WERE USED ****
TITLE 'SECANT METHOD FOR F(X)=0. BY LESLIE BROOKS'

;SPECIAL EQUATES
SET LBL_BASE 02 ;WE HAVE USED LABEL 1

;REGISTER EQUATES
EQU GUESST 00 ;FIRST GUESS FOR X
EQU GUESS2 01 ;SECOND GUESS FOR X
EQU Loop 02 ;LOOP COUNT
EQU FUNCTION 04 ;FUNCTION NAME
EQU SCRATCH 05 ;SCRATCH REGISTER (USUALLY
;HOLDS f(Xn-1)

;LABELS
EQU START 01 ;START OF THE MAIN LOOP

;FLAGS
EQU NO_PROMPT 10 ;FLAG IS SET IF GUESSES ARE ALREADY
ENTERED

ENTRY POINT TO THE PROGRAM
5343

;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON’T
;PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME

;SET IF CALLED AS A SUBROUTINE
;SKIP THE PROMPTING IF SET

;ELSE PROMPT NORMALLY

;ASK FOR GUESS 1

;ASK FOR GUESS 2

;SAVE Xn

;NOW PROMPT FOR THE FUNCTION NAME

;CREATE A LABEL NUMBER FOR THIS

;CREATE A NEW LABEL BASE
SECANT METHOD FOR \( f(x) = 0 \)  BY LESLIE BROOKS

50 001F 03 LBL FUNC_NAME
51 0020 FE6554E T "FUNCTION NAME?" ;ASK FOR THE FUNCTION NAME
4354494F
4E204E41
4D453F
52 002F 8C AOM
53 0030 8E PROMPT
54 0031 9A04 ASTO FUNCTION ;SAVE THE FUNCTION NAME
55 0030 8B AOFF
56 0034
57 0034 1A1011 .01
58 0037 32 STO LOOP ;LOOP 10 TIMES
59 0038
60 0038 ;CALCULATE \( f(\text{GUESS1}) \) TO START THE PROGRAM
61 0038
62 0038 02 LBL START
63 0039 20 RCL GUESS1 ;GET \( x_{n-1} \) BACK AGAIN
64 0039 AE04 XEQ IND FUNCTION ;EXECUTE THE FUNCTION
65 003C 35 STO SCRATCH ;AND SAVE \( f(x_{n-1}) \)
66 003D 21 RCL GUESS2 ;GET \( x_n \)
67 003E AE04 XEQ IND FUNCTION ;EVALUATE \( f(x_n) \)
68 0040 21 RCL GUESS2 ;GET \( x_n \)
69 0041 20 RCL GUESS1 ;AND \( x_{n-1} \)
70 0042 41 - ;SUBTRACT THESE
71 0043 71 X<>Y ;GET \( f(x_n) \) BACK
72 0044 9171 STO Z ;SAVE IT AGAIN
73 0046 25 RCL SCRATCH ;AND GET \( f(x_{n-1}) \)
74 0047 41 - ;SUBTRACT THESE
75 0048 43 / ; \( \frac{(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \)
76 0049 42 \* ;MULTIPLY BY \( f(x_n) \)
77 004A
78 004A ;\( x \) now contains a correction factor to be added to \( x_n \)
79 004A
80 004A 21 RCL GUESS2 ;GET \( x_n \)
81 004B 71 X<>Y
82 004C 41 -
83 004D
84 004D ;NOW WE HAVE \( x_{n+1} \) IN THE \( x \) REGISTER
85 004D
86 004D CE01 X<> GUESS2 ;EXCHANGE \( x_{n+1} \) WITH \( x_n \)
87 004F 30 STO GUESS1 ;\( x_n \) BECOMES THE NEW \( x_{n-1} \)
88 0050 9602 ISG LOOP ;INCREMENT THE LOOP COUNTER
89 0052 B200 GTO START ;LOOP IF NOT DONE
90 0054
91 0054 ;IF WE GET HERE, WE ARE DONE - DISPLAY THE RESULT
92 0054
93 0054 21 RCL GUESS2 ;\( x_n \)
94 0055 C00000 END
SECANT METHOD FOR $F(X) = 0$. BY LESLIE BROOKS

UNDEFINED ALPHA LABELS
(These are errors if not defined in another program)

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td></td>
<td>ON</td>
</tr>
</tbody>
</table>

***** NO SYMBOLS WERE UNDEFINED *****

FLAG USAGE SUMMARY

FLAG # LINE NUMBERS OF REFERENCES TO THE FLAG

010 33-TC

TAG MEANINGS ARE:
- CF CLEAR FLAG
- FS FLAG SET?
- FC FLAG CLEAR?
- SF SET FLAG
- TC FLAG TEST AND CLEAR

NUMERIC LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL #</th>
<th>DEFINED</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 62</td>
<td>34-6</td>
<td>09-6</td>
</tr>
<tr>
<td>002 50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- 6 GOTO
- X EXECUTE
SECANT METHOD FOR $F(X) \geq 0$. BY LESLIE BROOKS

REGISTER USAGE SUMMARY

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>LINE NUMBERS OF REFERENCES TO THE REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>40-S 63-R 69-R 87-S</td>
</tr>
<tr>
<td>001</td>
<td>42-S 66-R 68-R 80-R 86-XC 93-R</td>
</tr>
<tr>
<td>002</td>
<td>58-S 88-IS</td>
</tr>
<tr>
<td>004</td>
<td>54-S 64-XI 67-XI</td>
</tr>
<tr>
<td>005</td>
<td>65-S 73-R</td>
</tr>
<tr>
<td>Z</td>
<td>72-S</td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- CF: CLEAR FLAG INDIRECT
- DI: DECREMENT INDIRECT AND SKIP IF EQUAL
- DS: DECREMENT AND SKIP IF EQUAL
- FC: FLAG CLEAR? INDIRECT
- FS: FLAG SET? INDIRECT
- GI: GOTO INDIRECT
- II: INCREMENT INDIRECT AND SKIP IF GREATER
- IS: INCREMENT AND SKIP IF GREATER
- R: RECALL
- RI: RECALL INDIRECT
- S: STORE
- SF: SET FLAG INDIRECT
- SI: STORE INDIRECT
- TC: FLAG TEST AND CLEAR INDIRECT
- XI: EXECUTE INDIRECT
- XC: EXCHANGE X AND R

ALPHA LABEL USAGE SUMMARY

<table>
<thead>
<tr>
<th>LABEL</th>
<th>DEFINED VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>28 0000</td>
<td></td>
</tr>
</tbody>
</table>

TAG MEANINGS ARE:
- G: GOTO
- X: EXECUTE
SECANT METHOD FOR \( F(x) \geq 0 \). BY LESLIE BROOKS

**INTEGER SYMBOL USAGE SUMMARY**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED</th>
<th>VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIO</td>
<td>15</td>
<td>0004</td>
<td>54- 64- 67-</td>
</tr>
<tr>
<td>FUNC_MA</td>
<td>46</td>
<td>0002</td>
<td>50-</td>
</tr>
<tr>
<td>GUESS1</td>
<td>12</td>
<td>0000</td>
<td>40- 63- 69- 87-</td>
</tr>
<tr>
<td>GUESS2</td>
<td>13</td>
<td>0001</td>
<td>42- 66- 68- 80- 86- 93-</td>
</tr>
<tr>
<td>LOOP</td>
<td>14</td>
<td>0002</td>
<td>58- 88-</td>
</tr>
<tr>
<td>NO_PROM</td>
<td>24</td>
<td>000A</td>
<td>33-</td>
</tr>
<tr>
<td>R1</td>
<td>1</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>1</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>1</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>SCRATCH</td>
<td>16</td>
<td>0005</td>
<td>65- 73-</td>
</tr>
<tr>
<td>START</td>
<td>20</td>
<td>0001</td>
<td>34- 62- 89-</td>
</tr>
</tbody>
</table>

**STRING SYMBOL USAGE SUMMARY**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINED</th>
<th>VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE SYMBOL</th>
</tr>
</thead>
</table>

******* NO STRING SYMBOLS WERE USED *******

**VARIABLE USAGE SUMMARY**

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DEFINED</th>
<th>VALUE</th>
<th>LINE NUMBERS OF REFERENCES TO THE VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBL_BAS</td>
<td>7</td>
<td>0003</td>
<td>48-</td>
</tr>
</tbody>
</table>