USER CODE COMPILER

VERSION 1.45

REFERENCE MANUAL

HAND HELD PRODUCTS
6201 FAIR VALLEY DRIVE
CHARLOTTE, NORTH CAROLINA 28211
PHONE: (704) 377-3841

CONTENTS

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 m**eaningful 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 Super-brain) 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.

GETTING STARTED

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).

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 '^' 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

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: 41UCC    COM : RDS      COM : FILTER  COM : SECANT  UCC
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**

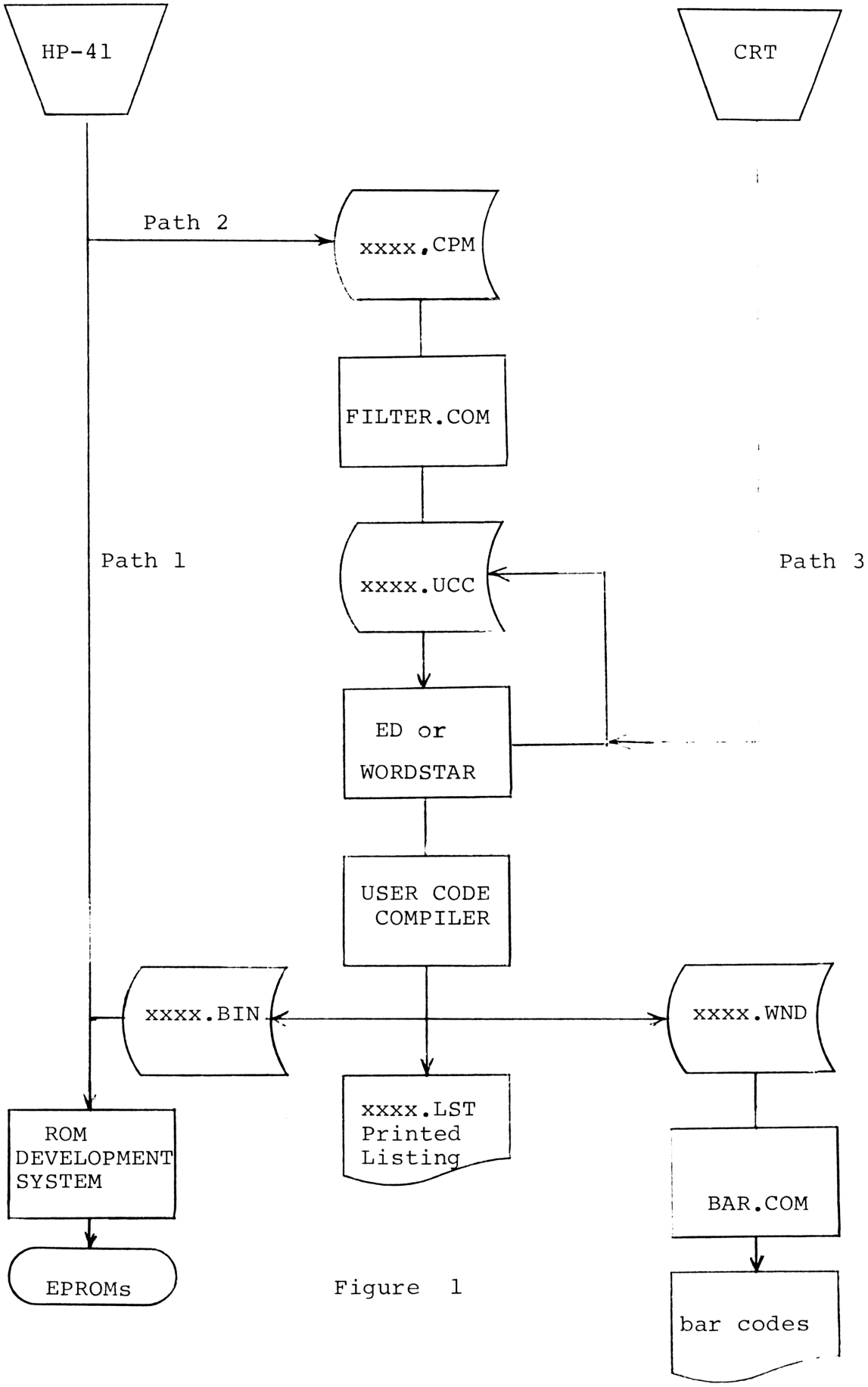

HP-41
CRT
Path 2
xxxx.CPM
FILTER.COM
Path 1
xxxx.UCC
Path 3
ED or
WORDSTAR
USER CODE
COMPILER
xxxx.BIN
xxxx.WND
xxxx.LST
Printed
Listing
ROM
DEVELOPMENT
SYSTEM
BAR.COM
EPROMs
bar codes

Figure 1

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 counter-parts; 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>
```

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 (WaND 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 WaND 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
     : *0A
    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

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

```
                        UNDEFINED ALPHA LABELS
          (THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)
LABEL    DEFINED VALUE   LINE NUMBERS OF REFERENCES TO THE SYMBOL
 NAME       ON

TUNE         10     0000       10-X

TAG MEANINGS ARE:          G        GOTO
                           X        EXECUTE
```

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
```

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.

A REAL PROGRAM

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 - R/(PI * F) [terminating resistance divided by the cutoff frequency times PI]

capacitor - 1 / (2 * PI * R * F) [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 = R/(PI * FREQUENCY)
;
      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 = 1 /(2 * PI * RESISTANCE * FREQUENCY)
;
      RCL         01             ;GET THE RESISTANCE AGAIN
      RCL         00             ;AND THE FREQUENCY
      *
```

```
     PI
     *
     STO+       X               ;DOUBLE IT
     STO        03              ;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          ;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

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
;    L = R/(PI * FREQUENCY)
```

```
;
     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
     RCL        FREQUENCY       ;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 site, 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
```

```
     0 ERROR(S) IN PASS ONE
     0 ERROR(S) IN PASS TWO
A>
```

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

```
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 micro-farads. 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:

```
     /                        ;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 "u". If the HP-41C had a lower case U we could use that. But wait - the HP-41C display **has** the Greek letter mu - but we can't get to it from the key-board. Will 41UCC allow us to use it? Yes, 41UCC will but we will need to know a bit about the HP-41C instruction set.

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

     1E6                         ;CONVERT TO MICROFARADS
     /
     STO        CAPACITOR        ;SAVE FOR FUTURE USE
;NOW DISPLAY THE CAPACITOR VALUE
     T          'C= '
     ARCL       X
     DB         TEXT4,APPEND,' ',MU,'F'   ;LABEL IT AS MICROFARADS
     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

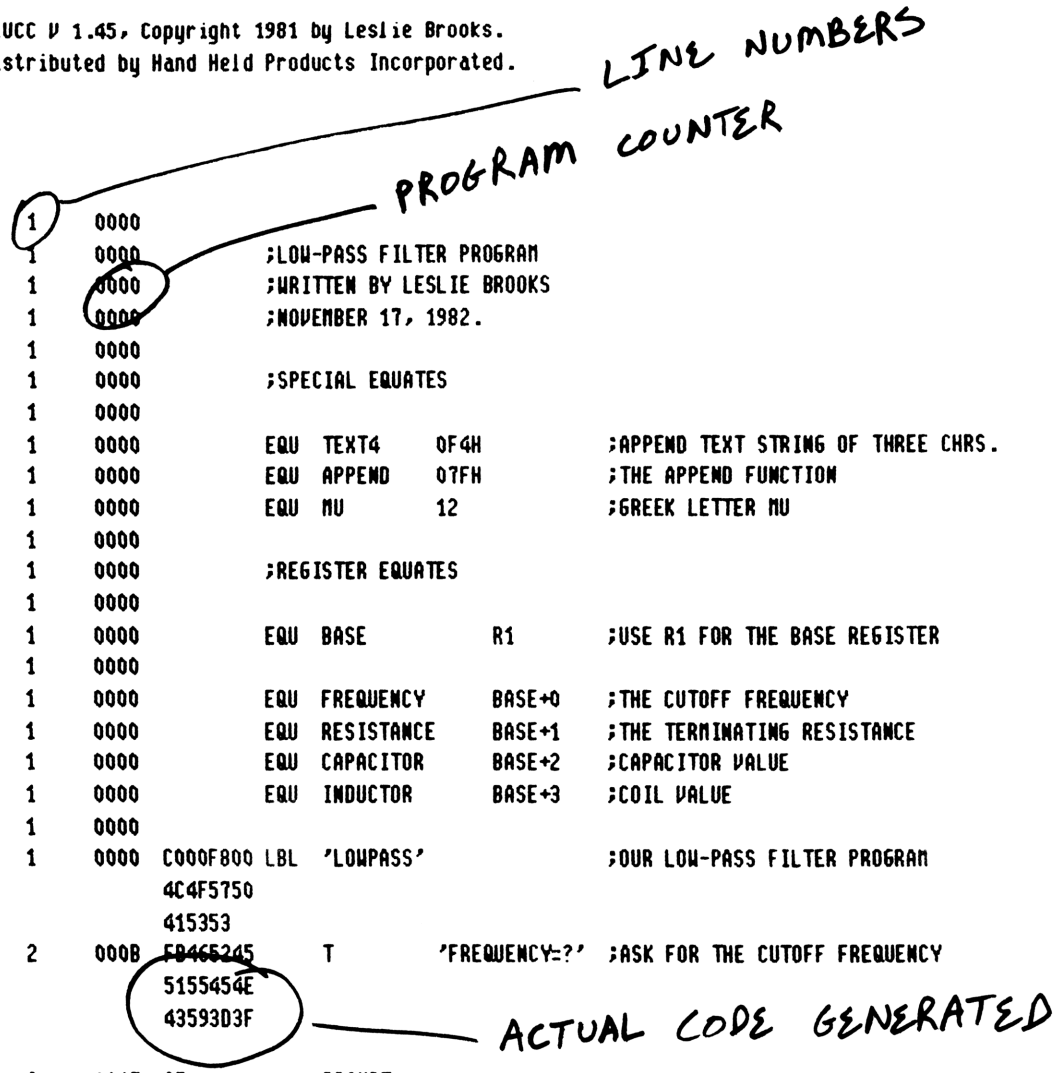
program like this

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

and the assignment would automatically be made for us when we scanned in the barcode for the program. RDS 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.
```




```
1     0000
1     0000          ;LOW-PASS FILTER PROGRAM
1     0000          ;WRITTEN BY LESLIE BROOKS
1     0000          ;NOVEMBER 17, 1982.
1     0000
1     0000          ;SPECIAL EQUATES
1     0000
1     0000          EQU  TEXT4     0F4H          ;APPEND TEXT STRING OF THREE CHRS.
1     0000          EQU  APPEND    07FH          ;THE APPEND FUNCTION
1     0000          EQU  MU        12            ;GREEK LETTER MU
1     0000
1     0000          ;REGISTER EQUATES
1     0000
1     0000          EQU  BASE           R1       ;USE R1 FOR THE BASE REGISTER
1     0000
1     0000          EQU  FREQUENCY      BASE+0   ;THE CUTOFF FREQUENCY
1     0000          EQU  RESISTANCE     BASE+1   ;THE TERMINATING RESISTANCE
1     0000          EQU  CAPACITOR      BASE+2   ;CAPACITOR VALUE
1     0000          EQU  INDUCTOR       BASE+3   ;COIL VALUE
1     0000
1     0000  C000F800 LBL  'LOWPASS'              ;OUR LOW-PASS FILTER PROGRAM
            4C4F5750
            415353
2     000B  F8465245      T        'FREQUENCY=?' ;ASK FOR THE CUTOFF FREQUENCY
            5155454E
            43593D3F
3     0017  8E            PROMPT
4     0018  30            STO      FREQUENCY     ;SAVE IT
5     0019  F9522854      T        'R(TERM)=?'   ;ASK FOR THE TERMINATING RESISTANCE
            45524D29
            3D3F
6     0023  8E            PROMPT
7     0024  31            STO      RESISTANCE    ;SAVE THE RESISTANCE
8     0025          ;
8     0025          ;CALCULATE THE INDUCTOR (COIL) FIRST
8     0025          ;    L = R/(PI * FREQUENCY)
8     0025          ;
8     0025  20            RCL      FREQUENCY     ;GET THE FREQUENCY
9     0026  72            PI
10    0027  42            *                      ;PI * FREQUENCY
11    0028  43            /                      ;DIVIDE INTO THE RESISTANCE
12    0029  11101010      1000                   ;CONVERT TO MILLIHENRIES
13    002D  43            /
14    002E  33            STO      INDUCTOR      ;SAVE IT FOR FUTURE USE
15    002F          ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
15    002F  F34C3D20      T        'L= '
16    0033  9B73          ARCL     X
17    0035  F47F204D      APPEND   ' MH'         ;UNITS ARE MILLIHENRIES
            48
18    003A  7E            AVIEW
19    003B  89            PSE
20    003C  33            STO      INDUCTOR      ;SAVE IT FOR FUTURE USE
```

```
21    003D          ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
21    003D  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  21            RCL       RESISTANCE      ;GET THE RESISTANCE AGAIN
26    0046  20            RCL       FREQUENCY       ;AND THE FREQUENCY
27    0047  42            *
28    0048  72            PI
29    0049  42            *
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,' ',MU,'F'  ;LABEL IT AS MICROFARADS
            46
37    005C  7E            AVIEW
38    005D  89            PSE
39    005E  C0000D        END
```

CROSS REFERENCE

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

UNDEFINED ALPHA LABELS
(THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)

LABEL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

FLAG USAGE SUMMARY

FLAG #	LINE NUMBERS OF REFERENCES TO THE FLAG

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

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

NUMERIC LABEL USAGE SUMMARY

LABEL #	DEFINED ON	LINE NUMBERS OF REFERENCES TO THE LABEL

***** NO NUMERIC LABELS WERE USED *****

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

REGISTER USAGE SUMMARY

REGISTER	LINE NUMBERS OF REFERENCES TO THE REGISTER #				
000	4-S	8-R	26-R		
001	7-S	25-R			
002	33-S				
003	14-S	20-S			
X	16-R	22-R	30-S	35-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

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

ALPHA LABEL USAGE SUMMARY

LABEL	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE LABEL
LOWPASS	1	0000	

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

INTEGER SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL		
APPEND	1	007F	36-		
BASE	1	0000			
CAPACIT	1	0002	33-		
FREQUEN	1	0000	4-	8-	26-
INDUCTO	1	0003	14-	20-	
MU	1	000C	36-		
R1	1	0000			
R2	1	0000			
R3	1	0000			
R4	1	0000			
RESISTA	1	0001	7-	25-	
TEXT4	1	00F4	36-		

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

STRING SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

VARIABLE USAGE SUMMARY

VARIABLE NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE VARIABLE

***** NO VARIABLES WERE USED *****

ANOTHER EXAMPLE PROGRAM

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

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

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

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

or

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

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
```

```
     PROMPT
     STO  00              ;SAVE X(n-1)
     T    'GUESS 2?'      ;ASK FOR GUESS 2
     STO  01              ;SAVE Xn
     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 Xn
     XEQ  IND 04          ;EVALUATE F(Xn)
     RCL  01              ;GET Xn
     RCL  00              ;AND X(n-1)
     -                    ;SUBTRACT THEM
     X<>Y                 ;GET f(Xn) BACK
     STO  Z               ;SAVE IT AGAIN
     RCL  05              ;AND GET f(X(n-1))
     -                    ;SUBTRACT THESE
     /                    ;( Xn - X(n-1) ) / ( f(Xn) - f(X(n-1)))
     *                    ;MULTIPLY BY f(Xn)

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

     RCL  01              ;GET Xn
     X<>Y
     -

;NOW WE HAVE X(n+1) IN THE X REGISTER

     X<>  01              ;EXCHANGE X(n+1) WITH Xn
     STO  00              ;Xn 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
```

```
     RCL  01              ;Xn
     END
```

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  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

LBL  'SC'                 ;ENTRY POINT TO THE PROGRAM
     T    'GUESS 1?'      ;ASK FOR GUESS 1
     PROMPT
     STO  GUESS1          ;SAVE Xn-1
     T    'GUESS 2?'      ;ASK FOR GUESS 2
     STO  GUESS2          ;SAVE Xn
     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 Xn-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 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
```

```
     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

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(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 Xn

;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.

ANOTHER EXAMPLE PROGRAM

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
```

```
;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
```

```
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
```

```
     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
```

```
A>
```

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.

```
1    0000
1    0000              TITLE     'SECANT METHOD FOR F(X)=0.  BY LESLIE BROOKS'
1    0000
1    0000          ;SPECIAL EQUATES
1    0000
1    0000              SET  LBL_BASE       02        ;WE HAVE USED LABEL 1
1    0000
1    0000
1    0000          ;REGISTER EQUATES
1    0000
1    0000              EQU  GUESS1         00        ;FIRST GUESS FOR X
1    0000              EQU  GUESS2         01        ;SECOND GUESS FOR X
1    0000              EQU  LOOP           02        ;LOOP COUNT
1    0000              EQU  FUNCTION       04        ;FUNCTION NAME
1    0000              EQU  SCRATCH       05        ;SCRATCH REGISTER (USUALLY
1    0000                                            ;HOLDS f(Xn-1)
1    0000          ;LABELS
1    0000
1    0000              EQU  START          01        ;START OF THE MAIN LOOP
1    0000
1    0000          ;FLAGS
1    0000
1    0000              EQU  NO_PROMPT      10        ;FLAG IS SET IF GUESSES ARE ALREADY
1    0000                                            ;ENTERED
1    0000
1    0000
1    0000  C000F300 LBL  'SC'                ;ENTRY POINT TO THE PROGRAM
           5343
2    0006
2    0006          ;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T
2    0006          ;PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME
2    0006
2    0006  AA0A          FS?C      NO_PROMPT      ;SET IF CALLED AS A SUBROUTINE
3    0008  B200          GTO       START          ;SKIP THE PROMPTING IF SET
4    000A
4    000A          ;ELSE PROMPT NORMALLY
4    000A
4    000A  F8475545      T    'GUESS 1?'     ;ASK FOR GUESS 1
           53532031
           3F
5    0013  8E            PROMPT
6    0014  30            STO  GUESS1         ;SAVE Xn-1
7    0015  F8475545      T    'GUESS 2?'     ;ASK FOR GUESS 2
           53532032
           3F
8    001E  31            STO  GUESS2         ;SAVE Xn
9    001F
9    001F          ;NOW PROMPT FOR THE FUNCTION NAME
9    001F
9    001F          EQU  FUNC_NAME LBL_BASE       ;CREATE A LABEL NUMBER FOR THIS
9    001F                                        ;ROUTINE
9    001F          SET  LBL_BASE  LBL_BASE+1     ;CREATE A NEW LABEL BASE
9    001F
```

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

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


    9     001F  03       LBL FUNC_NAME
   10     0020  FE46554E      T    'FUNCTION NAME?'    ;ASK FOR THE FUNCTION NAME
                4354494F
                4E204E41
                4D453F
   11     002F  8C            AON
   12     0030  8E            PROMPT
   13     0031  9A04          ASTO FUNCTION            ;SAVE THE FUNCTION NAME
   14     0033  8B            AOFF
   15     0034
   15     0034  1A1011        .01
   16     0037  32            STO  LOOP           ;LOOP 10 TIMES
   17     0038
   17     0038            ;CALCULATE F(GUESS1) TO START THE PROGRAM
   17     0038
   17     0038  02       LBL START
   18     0039  20            RCL  GUESS1         ;GET Xn-1 BACK AGAIN
   19     003A  AE84          XEQ  IND FUNCTION   ;EXECUTE THE FUNCTION
   20     003C  35            STO  SCRATCH        ;AND SAVE F(Xn-1)
   21     003D  21            RCL  GUESS2         ;GET Xn
   22     003E  AE84          XEQ  IND FUNCTION   ;EVALUATE F(Xn)
   23     0040  21            RCL  GUESS2         ;GET Xn
   24     0041  20            RCL  GUESS1         ;AND Xn-1
   25     0042  41            -                   ;SUBTRACT THEM
   26     0043  71            X<>Y                ;GET f(Xn) BACK
   27     0044  9171          STO  Z              ;SAVE IT AGAIN
   28     0046  25            RCL  SCRATCH        ;AND GET f(Xn-1)
   29     0047  41            -                   ;SUBTRACT THESE
   30     0048  43            /                   ;( Xn - Xn-1 ) / ( f(Xn) - f(Xn-1))
   31     0049  42            *                   ;MULTIPLY BY f(Xn)
   32     004A
   32     004A            ;X now contains a correction factor to be added to Xn
   32     004A
   32     004A  21            RCL  GUESS2         ;GET Xn
   33     004B  71            X<>Y
   34     004C  41            -
   35     004D
   35     004D            ;NOW WE HAVE Xn+1 IN THE X REGISTER
   35     004D
   35     004D  CE01          X<>  GUESS2         ;EXCHANGE Xn+1 WITH Xn
   36     004F  30            STO  GUESS1         ;Xn BECOMES THE NEW Xn-1
   37     0050  9602          ISG  LOOP           ;INCREMENT THE LOOP COUNTER
   38     0052  B200          GTO  START          ;LOOP IF NOT DONE
   39     0054
   39     0054            ;IF WE GET HERE, WE ARE DONE - DISPLAY THE RESULT
   39     0054
   39     0054  21            RCL  GUESS2         ;Xn
   40     0055  C0000D        END
```

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

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

UNDEFINED ALPHA LABELS
(THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)

LABEL NAME	DEFINED ON VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

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

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

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

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

NUMERIC LABEL USAGE SUMMARY

LABEL #	DEFINED ON	LINE NUMBERS OF REFERENCES TO THE LABEL
001	17	3-G 38-G
002	9	

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

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

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


                REGISTER USAGE SUMMARY

REGISTER   LINE NUMBERS OF REFERENCES TO THE REGISTER   #

  000       6-S     18-R     24-R     36-S
  001       8-S     21-R     23-R     32-R     35-XC    39-R
  002      16-S     37-IS
  004      13-S     19-XI    22-XI
  005      20-S     28-R
    Z      27-S

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
```

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

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


                ALPHA LABEL USAGE SUMMARY

 LABEL  DEFINED VALUE   LINE NUMBERS OF REFERENCES TO THE LABEL
          ON

SC          1    0000

TAG MEANINGS ARE:       G       GOTO
                        X       EXECUTE
```

INTEGER SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL					
FUNCTIO	1	0004	13-	19-	22-			
FUNC_NA	9	0002	9-					
GUESS1	1	0000	6-	18-	24-	36-		
GUESS2	1	0001	8-	21-	23-	32-	35-	39-
LOOP	1	0002	16-	37-				
NO_PROM	1	000A	2-					
R1	1	0000						
R2	1	0000						
R3	1	0000						
R4	1	0000						
SCRATCH	1	0005	20-	28-				
START	1	0001	3-	17-	38-			

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

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

STRING SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

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

VARIABLE USAGE SUMMARY

VARIABLE NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE VARIABLE
LBL_BAS	1	0003	9-

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,

```
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 exceptions. 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):

```
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,Q=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:

```
0ABH  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.

```
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 MATH1A.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 MATH1A.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 MATH1A.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

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 psuedo-op tells 41UCC that you would like a title to be printed at the top of every page of your LiST file. The TITLE psuedo-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

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.

R1= specifies the value of the symbol R1. If no R1 is given in the command line, R1 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 R1 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

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

/TYPE 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.

APPENDIX A - INSTRUCTIONS THAT DIFFER FROM THE HP-41 STANDARD

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.

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

APPENDIX B - SUMMARY OF ERROR MESSAGES

LISTING ERROR MESSAGES

A - **A**rgument 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 **C**haracter 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 **D**efined 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 **E**xpression 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 **O**verflowed 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 - **P**hase 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
```

```
6     0009
7     0009        END
```

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 **U**ndefined. 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 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:

```
STO     A
STO     a               ;GENERATES THE SAME CODE
STO     M
ISG     P
```

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:

```
STO     111
STO     IND     110
DSE     100
```

while these are invalid:

```
LBL     100
GTO     100
XEQ     100
```

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:

```
TONE    66
FIX     9
ENG     3
```

while these are still illegal:

```
ENG     10
SF      30
FS?C    60
```

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:

```
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:

```
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.

A>

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 Graftrax) 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 Graftrax) 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
  .
  .
  .
-I41UCX.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


```
 2    0000           ;TEST 28 OF THE HP-41C CROSS ASSEMBLER
 3    0000           ;FULL BLOWN TEST - APRIL 9, 1982
 4    0000           ;This program is a test of 41UCC.
 5    0000           ;It contains every legal 41UCC instruction in several possible forms.
 6    0000           ;It also contains most of the allowed pseudo-ops.
 7    0000           ;All of the instructions are in alphabetical order, so it provides a handy
 8    0000           ;reference.
 9    0000           ;IF YOUR COPY OF 41UCC CANNOT COMPILE THIS PROGRAM WITHOUT ERRORS YOU HAVE A
10    0000           ;PROBLEM! YOU COULD HAVE A BAD COPY, A BAD DISK, OR A BAD MACHINE.
11    0000
12    0000           EQU    SCRATCH 05
13    0000           EQU    MATRIX  '30,01'         ;HP MATH PAC MATRIX ROUTINE
14    0000
15    0000  C000F600 LBL    'TST28'
            54535432
            38
16    0009  4C               %
17    000A  4D               %CH
18    000B  42               *
19    000C  40               +
20    000D  41               -
21    000E  43               /
22    000F  60               1/X
23    0010  57               10**X
24    0011  57               10^X
25    0012  61               ABS
26    0013  5D               ACOS
27    0014  8F               ADV
28    0015  8B               AOFF
29    0016  8C               AON
30    0017  F27F41           APPEND  'A'
31    001A  FF7F3031         APPEND  '01234567891234'
            32333435
            36373839
            31323334

32    002A  F27F41           APPND   'A'
33    002D  F27F41           APND    'A'
34    0030  9B73             ARCL    X
35    0032  9B00             ARCL    00
36    0034  9B14             ARCL    20
37    0036  9B63             ARCL    99
38    0038  9B7B             ARCL    A
39    003A  9B80             ARCL    IND     00
40    003C  9B85             ARCL    IND     SCRATCH
41    003E  9B94             ARCL    IND     20
42    0040  9BE3             ARCL    IND     99
43    0042  9BF3             ARCL    IND     X
44    0044  9BFB             ARCL    IND     A
45    0046  88               ASHF
46    0047  5C               ASIN
47    0048  9A00             ASTO    00
48    004A  9A14             ASTO    20
```

```
49    004C  9A63          ASTO    99
50    004E  9A73          ASTO    X
51    0050  9A7C          ASTO    B
52    0052  9A80          ASTO    IND     00
53    0054  9A85          ASTO    IND     SCRATCH
54    0056  9A94          ASTO    IND     20
55    0058  9AE3          ASTO    IND     99
56    005A  9AF3          ASTO    IND     X
57    005C  9AFB          ASTO    IND     A
58    005E  5E            ATAN
59    005F  7E            AVIEW
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  7F            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  48454C4C      DB      'HELLO WORLD',25+03H+00010000B
            4F20574F
            524C442C

80    0087  444F4E27      DB      "DON'T GO AWAY"             ;DOUBLE QUOTES ARE ALLOWED
            5420474F
            20415741
            59
81    0094  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    00A8  97F3          DSE     IND     X
93    00AA  97F5          DSE     IND     M
94    00AC  55            E**X
95    00AD  55            E^X
```

```
 96    00AE  58            E**X-1
 97    00AF  58            E^X-1
 98    00B0  9E00          ENG     00
 99    00B2  9E09          ENG     9
100    00B4  9EF3          ENG     IND     X
101    00B6  9EE3          ENG     IND     99
102    00B8  9E94          ENG     IND     20
103    00BA  9EF5          ENG     IND     M
104    00BC  83            ENTER
105    00BD  83            ENTER^
106    00BE  62            FACT
107    00BF  AD00          FC?     00
108    00C1  AD09          FC?     9
109    00C3  AD0F          FC?     15
110    00C5  AD37          FC?     55
111    00C7  ADF3          FC?     IND     X
112    00C9  ADF5          FC?     IND     M
113    00CB  ADE3          FC?     IND     99
114    00CD  AD80          FC?     IND     0
115    00CF  AD94          FC?     IND     20
116    00D1  AB00          FC?C    00
117    00D3  AB1D          FC?C    29
118    00D5  ABF3          FC?C    IND     X
119    00D7  ABF5          FC?C    IND     M
120    00D9  AB80          FC?C    IND     0
121    00DB  AB94          FC?C    IND     20
122    00DD  ABE3          FC?C    IND     99
123    00DF  9C00          FIX     0
124    00E1  9C09          FIX     9
125    00E3  9CF3          FIX     IND     X
126    00E5  9CF5          FIX     IND     M
127    00E7  9C80          FIX     IND     0
128    00E9  9C94          FIX     IND     20
129    00EB  9CE3          FIX     IND     99
130    00ED  69            FRC
131    00EE  AC00          FS?     00
132    00F0  AC37          FS?     55
133    00F2  ACF3          FS?     IND     X
134    00F4  ACF6          FS?     IND     M
135    00F6  AC80          FS?     IND     0
136    00F8  AC94          FS?     IND     20
137    00FA  ACE3          FS?     IND     99
138    00FC  AA00          FS?C    00
139    00FE  AA1D          FS?C    29
140    0100  AAF3          FS?C    IND     X
141    0102  AAF7          FS?C    IND     0
142    0104  AA80          FS?C    IND     0
143    0106  AA94          FS?C    IND     20
144    0108  AAE3          FS?C    IND     99
145    010A  C000F200      GLBL    'A'
             41
146    010F  C000F500      GLBL    'FRED'
             46524544
```

```
147   0117  82          GRAD
148   0118  B100        GOTO    00
149   011A  D00063      GOTO    99
150   011D  AE00        GOTO    IND     00
151   011F  AE14        GOTO    IND     20
152   0121  AE63        GOTO    IND     99
153   0123  AE73        GOTO    IND     X
154   0125  AE78        GOTO    IND     P
155   0127  AE7F        GOTO    IND     E
156   0129  1DF44652    GOTO    'FRED'
                4544
157   012F  1DF74652    GOTO    'FREQUENCY'
                45515545
                4E
158   0138  D00066      GOTO    'A'
159   013B  B100        GTO     00
160   013D  D00063      GTO     99
161   0140  AE00        GTO     IND     00
162   0142  AE14        GTO     IND     20
163   0144  AE63        GTO     IND     99
164   0146  AE73        GTO     IND     X
165   0148  AE78        GTO     IND     P
166   014A  AE7F        GTO     IND     E
167   014C  1DF44652    GTO     'FRED'
                4544
168   0152  1DF74652    GTO     'FREQUENCY'
                45515545
                4E
169   015B  D00066      GTO     'A'
170   015E  1DF44652    GTO6    'FRED'
                4544
171   0164  1DF141      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     IND     Q
183   0178  9694        ISG     IND     20
184   017A  96E3        ISG     IND     99
185   017C  96F3        ISG     IND     X
186   017E  96F9        ISG     IND     Q
187   0180  76          LASTX
188   0181  01          LBL     00
189   0182  CF63        LBL     99
190   0184  CF66        LBL     'A'
191   0186  CF7B        LBL     'a'
```

```
192     0188  CF7F          LBL     'e'
193     018A  C000F800      LBL     'FREQUENCY'
              46524551
              55454E
194     0195  56            LOG
195     0196  50            LN
196     0197  65            LN1+X
197     0198  7C            MEAN
198     0199  4B            MOD
199     019A  6F            OCT
200     019B  8D            OFF
201     019C  4E            P->R
202     019D  4E            P-R
203     019E  72            PI
204     019F  8E            PROMPT
205     01A0  89            PSE
206     01A1  6B            R-D
207     01A2  4F            R->P
208     01A3  4F            R-P
209     01A4  81            RAD
210     01A5  20            RCL     00
211     01A6  2F            RCL     15
212     01A7  9010          RCL     16
213     01A9  9014          RCL     20
214     01AB  9063          RCL     99
215     01AD  9073          RCL     X
216     01AF  907C          RCL     B
217     01B1  9080          RCL     IND     00
218     01B3  9094          RCL     IND     20
219     01B5  90E3          RCL     IND     99
220     01B7  90F3          RCL     IND     X
221     01B9  90F9          RCL     IND     Q
222     01BB  75            RDN
223     01BC  6E            RND
224     01BD  85            RTN
225     01BE  74            R^
226     01BF  47            S+
227     01C0  48            S-
228     01C1  9D00          SCI     00
229     01C3  9D09          SCI     9
230     01C5  9DF3          SCI     IND     X
231     01C7  9D80          SCI     IND     00
232     01C9  9D94          SCI     IND     20
233     01CB  9DE3          SCI     IND     99
234     01CD  9DFD          SCI     IND     C
235     01CF  7D            SDEV
236     01D0  A800          SF      00
237     01D2  A814          SF      20
238     01D4  A81D          SF      29
239     01D6  A880          SF      IND     00
240     01D8  A894          SF      IND     20
241     01DA  A8E3          SF      IND     99
242     01DC  A8F3          SF      IND     X
```

```
243    01DE  A8FE          SF      IND     D
244    01E0  A8F4          SF      IND     L
245    01E2  47            SIGMA+
246    01E3  48            SIGMA-
247    01E4  9900          SIGMAREG 00
248    01E6  7A            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          ST*     00
261    01FD  9414          ST*     20
262    01FF  9463          ST*     99
263    0201  9480          ST*     IND     00
264    0203  9494          ST*     IND     20
265    0205  94E3          ST*     IND     99
266    0207  94F3          ST*     IND     X
267    0209  94F2          ST*     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
```

```
296   0240  9163          STO    99
297   0242  9180          STO    IND    00
298   0244  918F          STO    IND    15
299   0246  91E3          STO    IND    99
300   0248  91FC          STO    IND    B
301   024A  84            STOP
302   024B  F548454C      T      'HELLO'
            4C4F
303   0251  F5592741      T      "Y'ALL"
            4C4C
304   0257  F5592741      T      'Y''ALL'
            4C4C
305   025D  5B            TAN
306   025E  9F00          TONE   0
307   0260  9F09          TONE   9
308   0262  9F7F          TONE   127
309   0264  9F80          TONE   IND    00
310   0266  9F94          TONE   IND    20
311   0268  9FE3          TONE   IND    99
312   026A  9FF3          TONE   IND    X
313   026C  9800          VIEW   00
314   026E  9814          VIEW   20
315   0270  9863          VIEW   99
316   0272  9873          VIEW   X
317   0274  9880          VIEW   IND    00
318   0276  9894          VIEW   IND    20
319   0278  98E3          VIEW   IND    99
320   027A  98F3          VIEW   IND    X
321   027C  98F0          VIEW   IND    T
322   027E  63            X!=0?
323   027F  63            X≠0?
324   0280  79            X!=Y?
325   0281  79            X≠Y?
326   0282  51            X↑2
327   0283  66            X<0?
328   0284  7B            X<=0?
329   0285  46            X<=Y?
330   0286  CE00          X<>    00
331   0288  CE14          X<>    20
332   028A  CE63          X<>    99
333   028C  CE70          X<>    T
334   028E  CE80          X<>    IND    00
335   0290  CE94          X<>    IND    20
336   0292  CEE3          X<>    IND    99
337   0294  CEF3          X<>    IND    X
338   0296  CEF5          X<>    IND    M
339   0298  71            X<>Y
340   0299  44            X<Y?
341   029A  67            X=0?
342   029B  78            X=Y?
343   029C  64            X>0?
344   029D  45            X>Y?
345   029E  E00000        XEQ    00
```

```
346   02A1  E00063        XEQ   99
347   02A4  E00066        XEQ   'A'
348   02A7  E0007B        XEQ   'a'
349   02AA  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  1EF74652      XEQ   'FREQUENCY'
            45515545
            4E
357   02C7  1EF141        XEQ6  'A'
358   02CA  1EF44652      XEQ6  '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)

LABEL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

FLAG USAGE SUMMARY

FLAG #	LINE NUMBERS OF REFERENCES TO THE FLAG					
000	61-CF	107-FC	116-TC	131-FS	138-TC	236-SF
009	108-FC					
015	109-FC					
020	237-SF					
029	62-CF	117-TC	139-TC	238-SF		
055	110-FC	132-FS				

TAG MEANINGS ARE:

CF	CLEAR FLAG
FS	FLAG SET?
FC	FLAG CLEAR?
SF	SET FLAG
TC	FLAG TEST AND CLEAR

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

NUMERIC LABEL USAGE SUMMARY

LABEL #	DEFINED ON	LINE NUMBERS OF REFERENCES TO THE LABEL			
000	188	148-G	159-G	345-X	
099	189	149-G	160-G	346-X	
A	190	158-G	169-G	347-X	354-X
a	191	348-X			
e	192	349-X			

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

REGISTER USAGE SUMMARY

REGISTER	LINE NUMBERS OF REFERENCES TO THE REGISTER #						
000	35-R	39-RI	47-S	52-SI	65-CF	83-DS	88-DI
	114-FC	120-TC	135-FS	142-TC	150-GI	161-GI	177-IS
	182-II	210-R	217-RI	239-SF	260-S	263-SI	268-S
	272-SI	277-S	280-SI	285-S	288-SI	292-S	297-SI
	330-XC	334-XC	350-XI				
005	40-RI	53-SI	66-CF	89-DI			
014	293-S						
015	211-R	294-S	298-SI				
016	212-R	295-S					
020	36-R	41-RI	48-S	54-SI	67-CF	84-DS	90-DI
	115-FC	121-TC	136-FS	143-TC	151-GI	162-GI	178-IS
	183-II	213-R	218-RI	240-SF	261-S	264-SI	269-S
	273-SI	278-S	281-SI	286-S	289-SI	331-XC	335-XC
	351-XI						
099	37-R	42-RI	49-S	55-SI	64-CF	85-DS	91-DI
	113-FC	122-TC	137-FS	144-TC	152-GI	163-GI	179-IS
	184-II	214-R	219-RI	241-SF	262-S	265-SI	270-S
	274-SI	279-S	282-SI	287-S	290-SI	296-S	299-SI
	332-XC	336-XC	352-XI				
T	333-XC						
Y	267-SI						
X	34-R	43-RI	50-S	56-SI	63-CF	86-DS	92-DI
	111-FC	118-TC	133-FS	140-TC	153-GI	164-GI	180-IS
	185-II	215-R	220-RI	242-SF	266-SI	271-S	275-SI
	283-SI	291-SI	337-XC	353-XI			
L	244-SF						
M	87-DS	93-DI	112-FC	119-TC	338-XC		
N	134-FS						
O	141-TC						
P	154-GI	165-GI					
Q	68-CF	186-II	221-RI				
a	38-R	44-RI	57-SI	276-SI			
b	51-S	216-R	300-SI				
d	243-SF						
e	155-GI	166-GI	181-IS	284-SI			

TAG MEANINGS ARE:

CF	CLEAR FLAG INDIRECT	SI	STORE INDIRECT
DI	DECREMENT INDIRECT AND SKIP IF EQUAL	TC	FLAG TEST AND CLEAR INDIRECT
DS	DECREMENT AND SKIP IF EQUAL	XI	EXECUTE INDIRECT
FC	FLAG CLEAR? INDIRECT	XC	EXCHANGE X AND R
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		

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

ALPHA LABEL USAGE SUMMARY

LABEL	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE LABEL				
A	145	010A	171-G	357-X			
FRED	146	010F	156-G	167-G	170-G	355-X	358-X
FREQUEN	193	018A	157-G	168-G	356-X		
TST28	15	0000					

TAG MEANINGS ARE: G GOTO
X EXECUTE

INTEGER SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL			
R1	1	0000				
R2	1	0000				
R3	1	0000				
R4	1	0000				
SCRATCH	12	0005	40-	53-	66-	89-

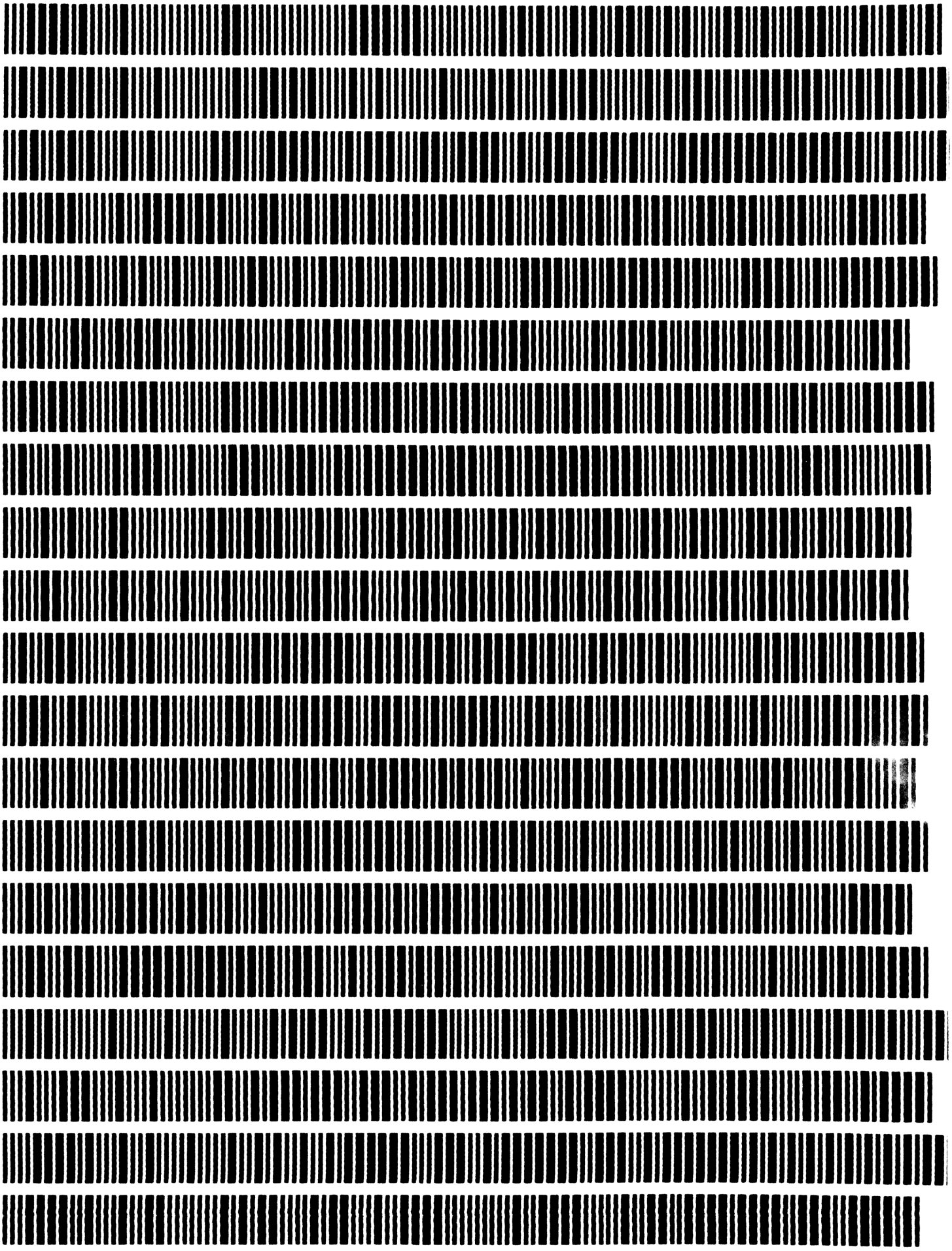
STRING SYMBOL USAGE SUMMARY

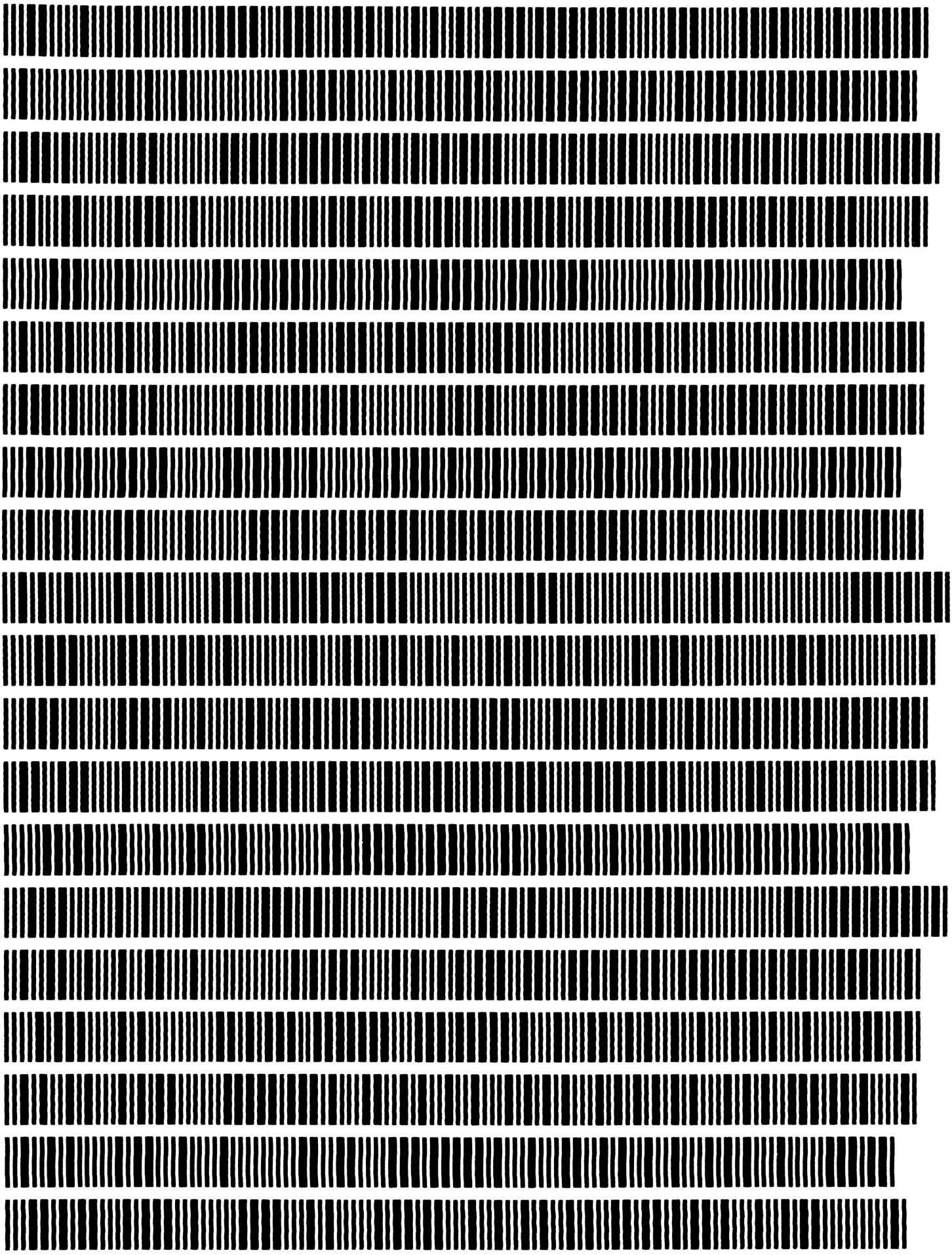
SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL
MATRIX	13	"30,01"	360-

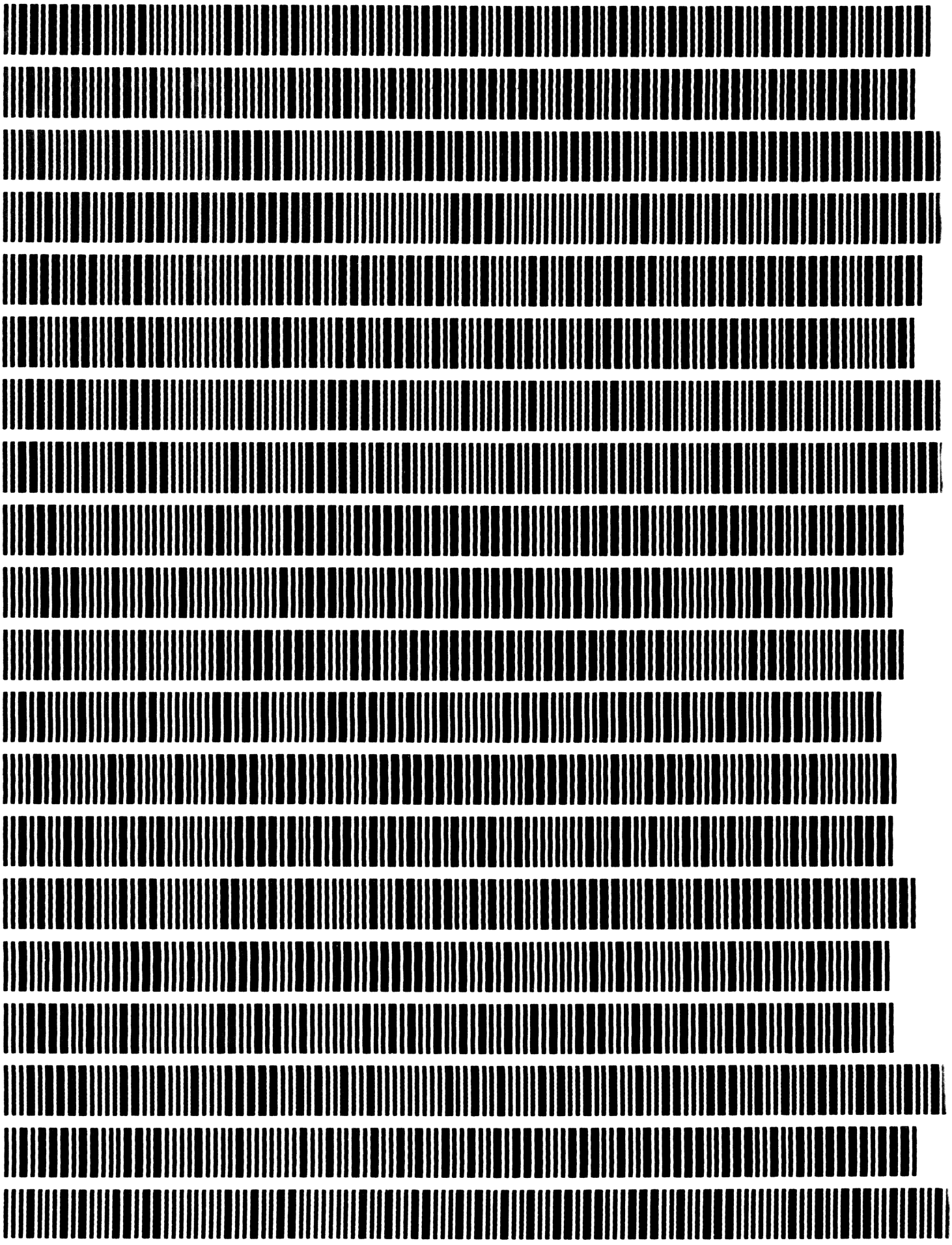
VARIABLE USAGE SUMMARY

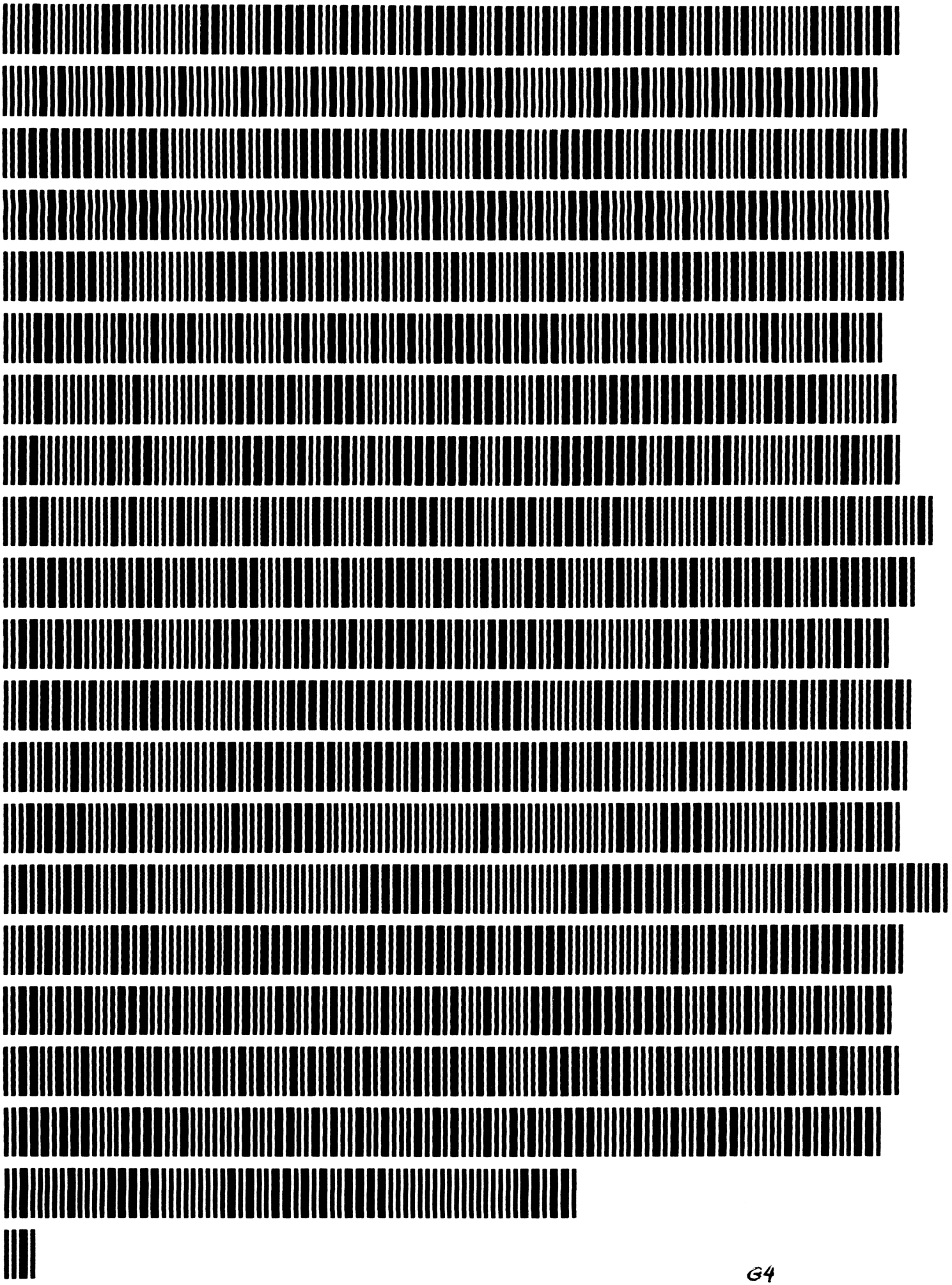
VARIABLE NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE VARIABLE

***** NO VARIABLES WERE USED *****









```
 2    0000
 3    0000          ;LOW-PASS FILTER PROGRAM
 4    0000          ;WRITTEN BY LESLIE BROOKS
 5    0000          ;NOVEMBER 17, 1982.
 6    0000
 7    0000          ;SPECIAL EQUATES
 8    0000
 9    0000          EQU  TEXT4     0F4H          ;APPEND TEXT STRING OF THREE CHRS.
10    0000          EQU  APPEND    07FH          ;THE APPEND FUNCTION
11    0000          EQU  MU        12            ;GREEK LETTER MU
12    0000
13    0000          ;REGISTER EQUATES
14    0000
15    0000          EQU  BASE           R1       ;USE R1 FOR THE BASE REGISTER
16    0000
17    0000          EQU  FREQUENCY      BASE+0   ;THE CUTOFF FREQUENCY
18    0000          EQU  RESISTANCE     BASE+1   ;THE TERMINATING RESISTANCE
19    0000          EQU  CAPACITOR      BASE+2   ;CAPACITOR VALUE
20    0000          EQU  INDUCTOR       BASE+3   ;COIL VALUE
21    0000
22    0000  C000F800 LBL  'LOWPASS'              ;OUR LOW-PASS FILTER PROGRAM
            4C4F5750
            415353
23    000B  FB465245      T        'FREQUENCY=?' ;ASK FOR THE CUTOFF FREQUENCY
            5155454E
            43593D3F

24    0017  8E            PROMPT
25    0018  30            STO      FREQUENCY     ;SAVE IT
26    0019  F9522854      T        'R(TERM)=?'   ;ASK FOR THE TERMINATING RESISTANCE
            45524D29
            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  72            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  F47F204D      APPEND   ' MH'         ;UNITS ARE MILLIHENRIES
            48
44    003A  7E            AVIEW
45    003B  89            PSE
46    003C  33            STO      INDUCTOR      ;SAVE IT FOR FUTURE USE
```

```
 47    003D           ;NOW DISPLAY THE CALCULATED INDUCTOR VALUE
 48    003D  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  111B16        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  F3433D20      T         'C= '
 67    0055  9B73          ARCL      X
 68    0057  F47F200C      DB        TEXT4,APPEND,' ',MU,'F'  ;LABEL IT AS MICROFARADS
             46
 69    005C  7E            AVIEW
 70    005D  89            PSE
 71    005E  C0000D        END
```

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

UNDEFINED ALPHA LABELS
(THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)

LABEL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

FLAG USAGE SUMMARY

FLAG #	LINE NUMBERS OF REFERENCES TO THE FLAG

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

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

NUMERIC LABEL USAGE SUMMARY

LABEL #	DEFINED ON	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

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

ALPHA LABEL USAGE SUMMARY

LABEL	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE LABEL
LOWPASS	22	0000	

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

INTEGER SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL		
APPEND	10	007F	68-		
BASE	15	0000			
CAPACIT	19	0002	64-		
FREQUEN	17	0000	25-	33-	57-
INDUCTO	20	0003	39-	46-	
MU	11	000C	68-		
R1	1	0000			
R2	1	0000			
R3	1	0000			
R4	1	0000			
RESISTA	18	0001	28-	56-	
TEXT4	9	00F4	68-		

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

STRING SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

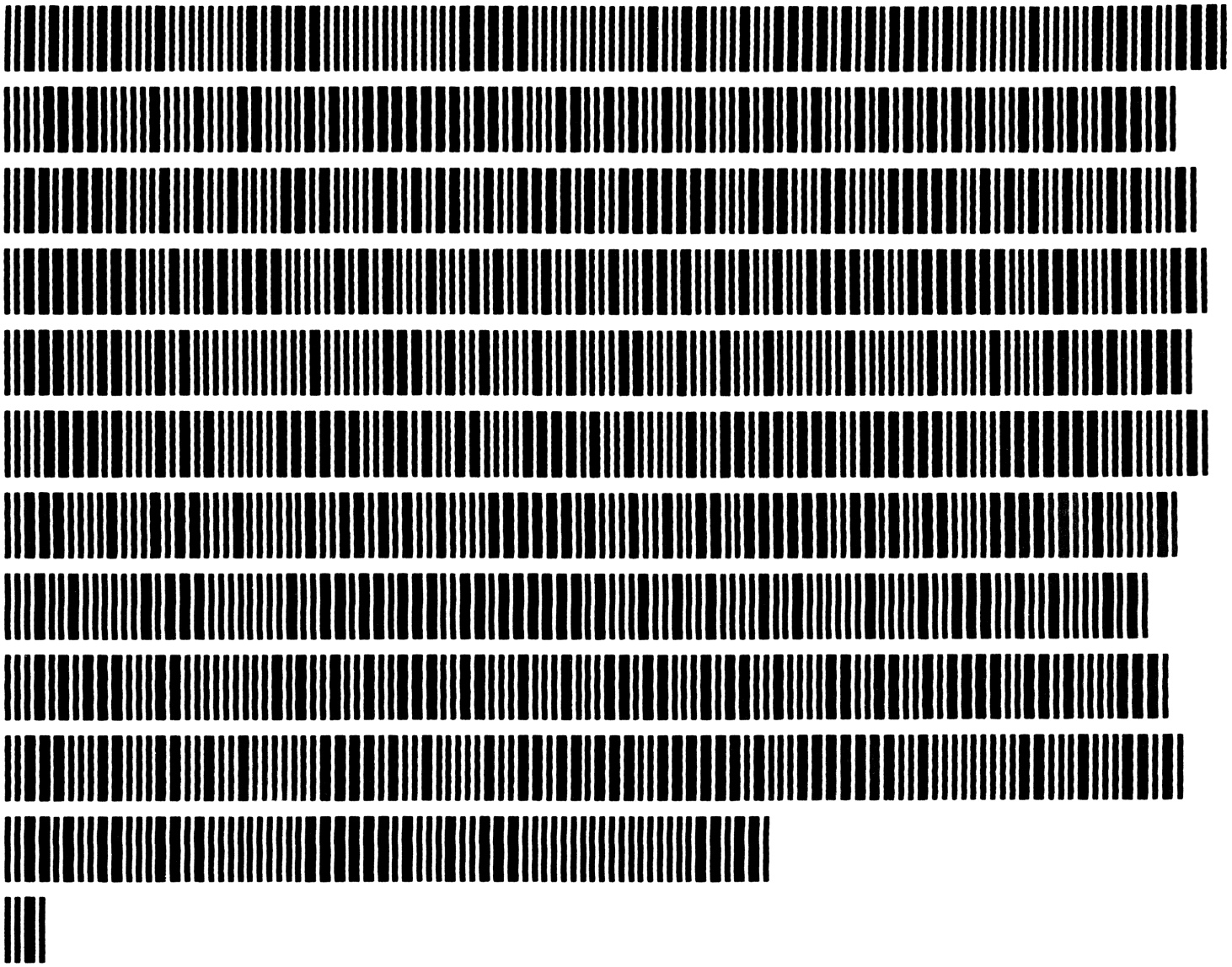
***** NO STRING SYMBOLS WERE USED *****

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

VARIABLE USAGE SUMMARY

VARIABLE NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE VARIABLE

***** NO VARIABLES WERE USED *****



```
 2    0000
 3    0000              TITLE     'SECANT METHOD FOR F(X)=0.  BY LESLIE BROOKS'
 4    0000
 5    0000          ;SPECIAL EQUATES
 6    0000
 7    0000              SET  LBL_BASE       02        ;WE HAVE USED LABEL 1
 8    0000
 9    0000
10    0000          ;REGISTER EQUATES
11    0000
12    0000              EQU  GUESS1         00        ;FIRST GUESS FOR X
13    0000              EQU  GUESS2         01        ;SECOND GUESS FOR X
14    0000              EQU  LOOP           02        ;LOOP COUNT
15    0000              EQU  FUNCTION       04        ;FUNCTION NAME
16    0000              EQU  SCRATCH        05       ;SCRATCH REGISTER (USUALLY
17    0000                                            ;HOLDS f(Xn-1)
18    0000          ;LABELS
19    0000
20    0000              EQU  START          01        ;START OF THE MAIN LOOP
21    0000
22    0000          ;FLAGS
23    0000
24    0000              EQU  NO_PROMPT      10        ;FLAG IS SET IF GUESSES ARE ALREADY
25    0000                                            ;ENTERED
26    0000
27    0000
28    0000  C000F300 LBL  'SC'               ;ENTRY POINT TO THE PROGRAM
            5343
29    0006
30    0006          ;BRANCH IF FLAG 10 IS SET - ACT LIKE A SUBROUTINE, DON'T
31    0006          ;PROMPT THE USER FOR THE INITIAL GUESSES OR FUNCTION NAME
32    0006
33    0006  AA0A          FS?C      NO_PROMPT      ;SET IF CALLED AS A SUBROUTINE
34    0008  B200          GTO       START          ;SKIP THE PROMPTING IF SET
35    000A
36    000A          ;ELSE PROMPT NORMALLY
37    000A
38    000A  F8475545      T    'GUESS 1?'     ;ASK FOR GUESS 1
            53532031
            3F
39    0013  8E            PROMPT
40    0014  30            STO  GUESS1         ;SAVE Xn-1
41    0015  F8475545      T    'GUESS 2?'     ;ASK FOR GUESS 2
            53532032
            3F
42    001E  31            STO  GUESS2         ;SAVE Xn
43    001F
44    001F          ;NOW PROMPT FOR THE FUNCTION NAME
45    001F
46    001F          EQU  FUNC_NAME LBL_BASE       ;CREATE A LABEL NUMBER FOR THIS
47    001F                                        ;ROUTINE
48    001F          SET  LBL_BASE  LBL_BASE+1     ;CREATE A NEW LABEL BASE
49    001F
```

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

```
   50     001F  03       LBL FUNC_NAME
   51     0020  FE46554E       T    'FUNCTION NAME?'    ;ASK FOR THE FUNCTION NAME
                4354494F
                4E204E41
                4D453F
   52     002F  8C             AON
   53     0030  8E             PROMPT
   54     0031  9A04           ASTO FUNCTION            ;SAVE THE FUNCTION NAME
   55     0033  8B             AOFF
   56     0034
   57     0034  1A1011         .01
   58     0037  32             STO  LOOP           ;LOOP 10 TIMES
   59     0038
   60     0038           ;CALCULATE F(GUESS1) TO START THE PROGRAM
   61     0038
   62     0038  02       LBL START
   63     0039  20             RCL  GUESS1         ;GET Xn-1 BACK AGAIN
   64     003A  AE84           XEQ  IND FUNCTION   ;EXECUTE THE FUNCTION
   65     003C  35             STO  SCRATCH        ;AND SAVE F(Xn-1)
   66     003D  21             RCL  GUESS2         ;GET Xn
   67     003E  AE84           XEQ  IND FUNCTION   ;EVALUATE F(Xn)
   68     0040  21             RCL  GUESS2         ;GET Xn
   69     0041  20             RCL  GUESS1         ;AND Xn-1
   70     0042  41             -                   ;SUBTRACT THEM
   71     0043  71             X<>Y                ;GET f(Xn) BACK
   72     0044  9171           STO  Z              ;SAVE IT AGAIN
   73     0046  25             RCL  SCRATCH        ;AND GET f(Xn-1)
   74     0047  41             -                   ;SUBTRACT THESE
   75     0048  43             /                   ;( Xn - Xn-1 ) / ( f(Xn) - f(Xn-1))
   76     0049  42             *                   ;MULTIPLY BY f(Xn)
   77     004A
   78     004A           ;X now contains a correction factor to be added to Xn
   79     004A
   80     004A  21             RCL  GUESS2         ;GET Xn
   81     004B  71             X<>Y
   82     004C  41             -
   83     004D
   84     004D           ;NOW WE HAVE Xn+1 IN THE X REGISTER
   85     004D
   86     004D  CE01           X<>  GUESS2         ;EXCHANGE Xn+1 WITH Xn
   87     004F  30             STO  GUESS1         ;Xn BECOMES THE NEW Xn-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         ;Xn
   94     0055  C0000D         END
```

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

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

UNDEFINED ALPHA LABELS
(THESE ARE ERRORS IF NOT DEFINED IN ANOTHER PROGRAM)

LABEL NAME	DEFINED ON VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

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

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

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

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

NUMERIC LABEL USAGE SUMMARY

LABEL #	DEFINED ON	LINE NUMBERS OF REFERENCES TO THE LABEL
001	62	34-G 89-G
002	50	

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

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

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

REGISTER USAGE SUMMARY

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

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

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

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

ALPHA LABEL USAGE SUMMARY

LABEL	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE LABEL
SC	28	0000	

TAG MEANINGS ARE:

G	GOTO
X	EXECUTE

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

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

INTEGER SYMBOL USAGE SUMMARY

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

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

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

STRING SYMBOL USAGE SUMMARY

SYMBOL NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE SYMBOL

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

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

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

VARIABLE USAGE SUMMARY

VARIABLE NAME	DEFINED ON	VALUE	LINE NUMBERS OF REFERENCES TO THE VARIABLE
LBL_BAS	7	0003	48-