



RAMBLING

RAMBLING is a place for the editor to discuss points of interest, news bits and the contents of the current issue of News80s. Since this is the initial issue of the publication, I thought I'd start this RAMBLING with some background information on myself and the publication. If you're not interested in that, then please GOTO this issue's feature article on the HP Assembler ROM on page 2.

My name is Dale Flanagan. I've been involved in systems design for around 12 years. Before that, I was a contributing editor for a hobby magazine. I have an MBA from UCLA in marketing and information systems, and I've also worked as a marketing manager for a major motorcycle importer, as well as several years as a systems and procedures manager for a major automobile importer. For the past 5 years I've run my own consulting firm that specializes in computer systems design, with clients both here in the U.S. and in Japan. You should also know that I'm a founder of Joseki Computer Corporation, a company that designs and markets microcomputer software (because of this, News80s has a policy of soliciting independent software reviews of Series 80 software, and any Joseki software will always be reviewed by an independent reviewer).

I bought my first microcomputer in 1977. It was an Apple II board, and the user had to supply his/her own keyboard, case and power supply! At first the micro was as much a toy as a serious business machine, because there wasn't much software to make it do "serious" work. But as software improved, and I became more involved in writing software for it, the business uses of microcomputers grew until I now use 2 Apple II's, a Radio Shack, an IBM PC, and an HP—85A. The HP was bought in early 1981, and it's become one of my favorite micros. My personal "ideal" micro hasn't been invented yet, but when it is, it will have many of the features found on the HP—85.

One frustrating aspect of owning a Series 80 microcomputer is the relative lack of information on the Series 80. Even in the early days, Apple users had many groups, newsletters and magazines devoted to spreading knowledge about the Apple II and its users. Today, there are 5 or 6 complete magazines devoted exclusively to the Apple, plus numerous club newsletters. Because they tend to be business and professional people, Series 80 users may not be club oriented. However, they're certainly information oriented.

HP's Basic Exchange newsletter is excellent, but it can't provide all the information Series 80 users need and want. Since Basic Exchange is an official HP publication, it's also obviously going to concentrate on HP's own products when it's doing reviews and such for the Series 80. The HP User's Library serves another useful purpose, but it's oriented towards software distribution and not information distribution. This is why News80s was born.

News80s has an orientation towards business and professional users of HP-83 and HP-85 microcomputers, but we'd welcome any contributions that help spread Series 80 information to our readers. We're especially looking for information that can help either the advanced or beginning Series 80 user, as well as short programs, hints or subroutines which might be of general interest. If you'd like to review a product or piece of software you've been using, then please send for our review guidelines.

Right now we can only offer editorial help and gratitude for your contributions, but by sharing your knowledge you'll be helping the Series 80 user base to grow in their understanding and use of their computers. In the long run, that can only help us all.

In this issue, we have a review of the HP Assembler ROM, the “For/Next” beginner’s column, and several “quickies” (hints). In future issues, we’re planning articles on systems design, translating Basic programs into HP Basic, and several other topics that should be of interest.

News80s will publish 4 times a year, in February, April, August and November. Series 80’s clubs and users’ groups may receive a 50 word mention in News80s by simply writing and telling us about your group. Classified ad rates are listed in the newsletter, and commercial ad rates are available on request.

If you wish to subscribe to News80s, a one year subscription (4 issues) is available for \$10 from Dale Flanagan, News80s, P.O. Box 1329, Redondo Beach, CA 90278. We hope you enjoy this sample!

HEWLETT PACKARD’S ASSEMBLER ROM

Sooner or later it’s going to happen. You’re going to be writing a program where you want to do something that Basic doesn’t let you do, or where Basic is too slow. What do you do?

Well, there are at least three answers to that. One, you don’t do it! Two, you get “clever” and find a way to program around it. Three, you use machine language. Machine language is the “natural” language of your computer. It lets you do anything your imagination and the physical constraints of the hardware allow. It works as fast as your computer works, and it gives you total control over data structures. So why don’t we all use machine language? Because it’s usually a tremendous amount of work when compared to a language like Basic. With all the control comes more responsibility for handling every little thing that the computer is supposed to do, and you often have to roll up your sleeves to slug it out at the bit and byte level.

Still, to handle specific problems, machine language is often the only way. An assembler is a program/tool to help you write machine language programs (called Binary programs in the HP Series 80 programming manual). The assembler doesn’t make binary program creation a snap, but it does ease the burden on the programmer considerably. With an assembler, you can express things in more human terms (called “mnemonics”), so if you want to load a particular value at a byte in memory, you can write “LDB” instead of “240”. The assembler also keeps track of physical locations in memory for subroutines and jumps (which are like GOTO’s in Basic), and allows you to use label names (instead of line numbers or memory location values) to indicate sections of code. It also allows much easier editing and modifications to the code you’re writing. When you’ve completed the program, the assembler will then “assemble” the binary program, substituting the proper machine language codes for the assembler’s mnemonics and labels.

The Hewlett Packard Series 80 assembler is in a Read Only Memory (ROM) chip. The Assembler goes for \$295 retail (plus the cost of a ROM drawer, if you don’t already have one). For your money you get the Assembler ROM; a large manual; sample programs; and a “Global” file, on both tape and disk, that contains various system labels and subroutine locations.

HP’s approach to assembly language programming in the Series 80 is rather unique. Instead of using special commands (like PEEK, POKE, CALL and USRs that other versions of Basic use to access machine language routines, HP actually allows you to expand the “vocabulary” of Basic, to include new reserved words that execute machine language routines.

Because a Series 80 binary program expands the Basic reserved word set, you also have to learn how Basic parses (or interprets) a program. Although some things are relatively straightforward to program, some other types of binary programs require you to write a parsing routine along with the machine language routine that actually does the work. An interesting aspect of this is that the binary program must be in memory before you can use any of the new words you've defined (or you'll get a syntax error), because the computer won't know how to parse the new reserved word properly. Since the computer searches the binary program before it searches the internal HP parsing routines, it's also possible for you to redefine existing HP Basic reserved words, so they'll operate in a new manner. For instance, you could redefine MIN(1,3) to return the sum of the two numbers, instead of the smaller of the two numbers. This may not have a lot of practical use, but it could lead to listings that would drive friends (and would—be software pirates) crazy.

The key to using the HP assembler is the manual. The manual is quite large but, in my opinion, it has several curious shortcomings.

The manual assumes that you already know about assembly language programming. With all the other material that has to be covered, this is a reasonable approach, although a beginner's tutorial never hurts. If you don't know about assembly language programming, trying to learn from the HP manual would be difficult, at best, and you should probably learn assembly language concepts by studying a beginner's book. Unfortunately, this will have to be a book written for some other microprocessor, because to my knowledge the HP manual is "it" when it comes to information on the Series 80 microprocessor. One book that I can recommend is "How to Program Microcomputers" by William Barden, Jr. (Howard W. Sams & Co., publishers), which covers 8080, 6800, and 6502 microprocessors, as well as general concepts that will also apply to the Series 80.

The manual covers Assembler statements and operation, an overview of the Series 80 microprocessor, Assembler instructions, Series 80 systems architecture, writing binary programs (for ROM and RAM), and built-in HP Series 80 routines which might be useful in writing your own binary programs. A short chapter also covers the HP—82928A System Monitor ROM, which is NOT necessary for Assembler ROM operations.

A good deal of time is spent on writing binary programs for ROM's. This information is mixed in with information on writing RAM based binary programs that are loaded in from tape or disk. Unfortunately, at this time there's no way for the Series 80 user to make his/her own ROMs, so all this material on ROM creation is intrusive and not too useful. The ROM material should have been put in a separate chapter, or placed in the instruction manual that will have to be written when, and if, HP ever releases a ROM creation accessory for the Series 80.

Tape and disk data formats and program formats are also missing from the manual, making any binary program manipulation of both harder. The thought occurred to me that this might have been done to make binary programs that pirate protected software harder to write, but this is just a supposition on my part.

The manual has only 6 complete examples of binary programs. To my taste, this isn't nearly enough, because I usually learn a great deal by studying program examples.

The manual also has several little cryptic statements and omissions. They're really not important, but to the curious they raise all sorts of questions with out providing answers. For instance, in the manual we learn that with the Series80, you can have one Basic program,

one binary (machine language) program, and several somewhat mysterious subprograms. The manual isn't exactly clear what a subprogram is, but apparently they're ROM resident programs of some type. Another thing we never learn is what kind of microprocessor chip is inside the Series 80. Evidently it's a custom HP chip, but we're told everything about the chip but what it's called. Of course, you can live a long and happy life without knowing the name of the Series 80 chip, but to the professionally curious, it is a little puzzle.

Most of these gripes are admittedly small, and to offset them the manual has many strong points. The description of addressing modes available, for instance, is extremely clear. The manual also has a complete description of how the Series 80 operating system works when parsing, running a Basic program and when running in the calculator mode.

Also included in the manual is a complete description of many of the internal ROM routines found in the HP 80. This is a veritable gold mine for the assembly language programmer, chock full of good stuff that otherwise you'd have to write for yourself. This section alone is worth the price of the Assembler ROM, if you do a large amount of assembly language programming.

The actual operation of the Assembler is very much like writing a Basic program. The lines of code have line numbers (yes, the AUTO and REN commands work), and all the HP's powerful editing keys can be used.

The Assembler also offers several powerful new features that can make life a little easier for the programmer. For instance, you can find any label or reference in the assembly program with the new FREF and FLABEL commands. HP even redefines the "K2" and "K3" keys to display the new commands for you! You can translate from octal to decimal or vice-versa with new commands, show or not show remarks (at your option), link source files, refer to a "Global" file that has all the system addresses of the HP routines mentioned earlier, and use a variety of pseudo opcodes (or assembler mnemonics) that control all sorts of options. Some of the more interesting pseudo opcodes are ones that allow conditional assembly of sections of code, based on the status of flags set by the programmer. Sorry, no assembler macro facility is provided, but I guess every thing can't be crammed into one little ROM. With the Assembler ROM, you can also examine sections of memory (directly or indirectly), and change memory if desired. With the HP Monitor ROM, you can set breakpoints and do some other things, but with the Assembler ROM's memory examine and change features, I can see why HP says the Monitor is not necessary for Assembler operations.

The HP Assembler ROM is currently the only assembly language tool available for the Series 80. Fortunately, it's a very good assembler and well worth the price for the programmer that wants to expand out of Basic into machine language programs.

QUICKIE #1 — INITIALIZING VALUES

When initializing values in a program, you can save a few bytes of memory by initializing the values as shown in lines 50 and 60, instead of the method in line 10.

```
10 X=0 @ Y=0           80 DISP X,Y,Z
20 !                   90 DISP A$, B$
50 X,Y,Z=0             100 END
60 A$, B$="HI"
```

PROGRAMS FOR FOR/NEXT COLUMN

```

10 ! PROGRAM 1
20 !
30 ! FIRST SET X START TIME
40 X=TIME
50 FOR I=1 TO 100
60 DISP I
70 NEXT I
80 Y=TIME - X
90 ! SET Y = ELAPSED LOOP TIME
100 !
110 ! NOW TRY 'GOTO' COUNTING
120 X=TIME
130 I=0
140 I=I+1
150 DISP I
160 IF I=100 THEN 180
170 GOTO 140
180 Z=TIME - X
190 !
200 DISP "GOTO LOOP TIME = "; Z
210 DISP "FOR...NEXT LOOP TIME = "; Y
220 DISP "FOR/NEXT LOOP IS "; Z - Y; " FASTER"

```

```

10 ! PROGRAM 2
20 !
30 X=TIME ! SET START TIME
40 FOR I TO 1000
50 NEXT I
60 !
70 Y = TIME - X ! SET Y TO ELAPSED TIME FOR
LOOP
80 DISP "LOOP 1 TIME IS ";Y
90 !
100 ! (NOW 25 REMARK LINES)
110 !
----
----
330 !
340 X = TIME ! SET START TIME AGAIN
350 FOR I=1 TO 1000
360 NEXT I
370 !
380 Y=TIME - X ! SET Y TO ELAPSED TIME FOR
LOOP
390 DISP "LOOP 2 TIME IS" ;Y
400 END

```

```

10 ! PROGRAM 3
20 !
30 DISP "LOOP IS SET TO GO TO 10"
40 DISP
50 DISP "INPUT A STARTING VALUE"
60 INPUT X
70 !
80 FOR I=X TO 10
90 DISP "I DID THE LOOP!"
100 NEXT I
110 !
120 DISP "DID I DO THE LOOP?"
130 END

```

```

10 ! PROGRAM 4
20 FOR I=1 TO 12.3 STEP .4
30 DISP I
40 NEXT I
50 DISP "ENDING VALUE WAS ";I
60 END

```

```

10 ! PROGRAM 5
20 FOR I=10 TO I STEP - 1
30 DISP I
40 NEXT I
50 DISP "ENDING VALUE WAS ";I
60 END

```

FOR/NEXT

FOR/NEXT is aimed at the beginner. Its usual format is to concentrate on a particular Basic command or programming technique, and it will try to amplify your understanding of the topic under discussion.

Before we continue, it's probably appropriate to lay down some ground rules so you'll know what we're trying to accomplish. For/Next isn't a Basic language primer. There are many good beginner's texts available, including the one that came with your HP-85 or HP-83. For some reason, some people assume that because they got it "free" with their computer, it can't be very good. For/Next assumes you've read one of these primers, to get background concepts on programming and the Basic topic under discussion. Specifically, For/Next will often refer you to the appropriate section of the HP Series 80 Owner's Manual and Programming Guide, so you'll know exactly where to look.

We believe that programming is a craft learned by experience. If you haven't tried the examples in the HP Guide, then you've short changed yourself in the learning experience. By the same token, we believe you should try the programs given as examples in For/Next. They'll almost always be short, and they'll hopefully illustrate a point about HP Basic that might be useful in your own programming.

In this column, we don't pretend to be a programming guru. In fact, like old time pilots and landings ("If you can walk away from it, it's a good one"), we believe that any program that works properly is a "good" one. Obviously, some programs are better than others, but one of the first things you should learn is that there are a tremendous number of ways you can program something to accomplish a task. In For/Next, we give you a variety of bits and pieces that, hopefully, will help you write programs that are "better" instead of merely good.

This time we'd like to start with the column's namesake, the For/Next loop. As Program 1 illustrates, a GOTO loop can be used to accomplish the same thing as a For/Next loop. Both the GOTO loop and the For/Next loop will display a series of numbers from 1 to 100 on your HP's screen. However, most people find the For/Next loop easier to read and understand. In addition, in this case the For/Next loop is slightly more efficient in terms of program space and speed. Finally, the For/Next loop is easier to set—up because there's no initial value to set (line 130) or test to be made (line 160), as with the GOTO loop.

Program 1, by the way, illustrates a timing technique that you can use in your own programs. The HP Series 80 has several built-in timers and a clock, and these can be used to time various events. The clock is read by the TIME function. Unless it's been set, the clock won't return the "real" time to you, but for this program we're only interested in elapsed time, so it doesn't matter. In line 40 we set X equal to the start time, and in line 80 we subtract the start time from the ending time to get the elapsed time in seconds. This same technique can be used to time sections of code in your own programs.

Before I go on, I guess I should also caution you not to go crazy over elapsed times. Most programs follow the old 80/20 rule (80% of the time is spent executing 20% of the code). If programs are really time sensitive, this technique can be used to discover where your program is spending it's time, and you can work on speeding up those sections of code. But before you do that, make sure your time is going to be productively spent as you "fine tune" the program. If you spend 4 hours speeding up a once a month job so it runs 8 minutes instead of 12 minutes, it will take you five years to offset your programming time with faster run times!

Program 2 illustrates an interesting property of HP Basic. This program has two For/Next loops separated by 25 REMark lines (omitted from the listing for clarity and space considerations). At the start of each loop and end of each loop, the internal HP TIME function is used to see how long it takes for each loop to run. If you copy this simple program and RUN it, you'll see that it takes the second For/Next loop about as long to run as the first one. So what?

Well, if you ran the same program on an Apple II computer (and added a clock, because they don't have a built-in one like the HP), you'd find that the second loop would run slower than the first. Because of the way the Basic in the Apple (among others) works, the speed of a For/Next loop is sensitive to the loop's placement in the program.

With HP Basic, you can place For/Next loops where they fall logically in a program, instead of resorting to fancy program structures to move speed sensitive loops to the front of the program; a very desirable feature!

Program 3 illustrates another property of HP Basic. If you run this program and enter a starting value greater than 10, the loop will not be executed (some Basics go through every loop at least once, no matter what the starting value). This is a useful piece of information to keep in mind. If you write a program where you count on the loop being executed at least once, be sure that you can't enter the loop with a starting value greater than the ending value.

Most loops increase by "1" each time, but in Program 4 we illustrate that this doesn't necessarily have to be so. We can explicitly tell the HP to count by a value other than "1" with the STEP command. In fact, as program 5 shows, we can even tell the HP to count backwards, if we want to.

Nesting of loops is covered very well in the HP Guide (Section 7), so we won't repeat that here.

The For/Next command is a powerful and useful one that is frequently used in almost all types of programs. I hope the sample programs and discussion illustrates some of its properties, and in the next column we'll look at another Basic command or technique and explore it together.

QUICKIE #2 — YES OR NO?

One of the most common questions asked of a computer operator is a "Yes" or "No" response to various questions asked in a program. The following subroutine shows one way to handle this situation. It can be called from appropriate parts of the program, and the "Q9" variable returns the response of the operator to the main program so appropriate action can be taken.

```
10 DISP "WANT TO TRY THIS          500 ! *** Y/N SUBROUTINE
AGAIN (Y/N)";                     510 Q9=3 FIRST SET ERROR FLAG
20 GOSUB 500                       520 INPUT K9$ ! NOW GET A KEY
30 ON Q9 GOTO 80,610,10           530 IF K9$="" THEN 580 ' CHECK FOR NULL STRING
40 ! 1ST NUMBER IS 'YES'          540 K9$=K9$[1,1] ' JUST GET FIRST CHARACTER
RESPONSE                          550 K9$=UPC$(K9$) ! CONVERT TO UPPERCASE
50 ! 2ND NUMBER IS 'NO'          560 IF K9$="Y" THEN Q9=1 ! YES
RESPONSE                          570 IF K9$="N" THEN Q9=2 NOPE
60 ! 3RD NUMBER IS ERROR -       580 IF Q9=3 THEN BEEP DISP ">>> 'Y' OR 'N' ONLY!"
RETRY INPUT                       590 ! (TELL 'EM WE HAVE A PROBLEM IF Q9 WASN'T
70 !                               RESET)
80 DISP "OKAY, ONE MORE          600 RETURN
TIME !"                            610 END
90 GOTO 10
```

SUBSCRIPTION AND ADVERTISING RATES

SUBSCRIPTIONS to News80s are available for \$10 for four issues. News80s is published quarterly, in February, April, August and November. All subscriptions start with the next published issue, after receipt of payment. Please make checks or money orders out to Dale Flanagan, and send subscriptions to News80s, P.O. Box 1329, Redondo Beach, CA 90278.

CLASSIFIED ADVERTISEMENTS are available for \$ 5.00 per issue (maximum of 50 words, please). Ads must be received by the 10th of January, March, July and October, for inclusion in the next month's issue.

COMMERCIAL AD RATES are available on request, for full page, half page and quarter page advertisements.

SERIES 80 CLUBS AND USER GROUPS may receive a free 50 word notice by writing News80s a note with information about your group. Please include information on how interested News80s readers can contact your group!

NEWS 80S NEWSLETTER
P.O. BOX 1329
Redondo Beach, CA 90278

Sample Issue

FIRST CLASS

n@w s80 s

The Microcomputer Journal For HP Series 80

**News80s ran for 12 issues between 1982 and 1984
(#1, #2, Special Issue and #3 thru #11).**

**It was an independent newsletter edited by Dale Flanagan for:
HP-83, HP-85, HP86 and HP87 Personal Computer users.**

**These are used with the permission of Dale Flanagan, who retains the
copyright.**

Scanned and converted by M. A. Cragg