How to Write an Application Program

William B Jenkins
643 S Rte 83
Elmhurst IL 60126

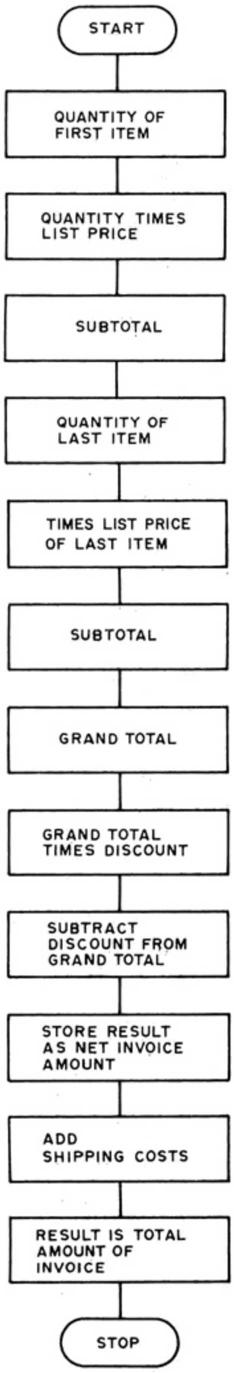

START
QUANTITY OF FIRST ITEM
QUANTITY TIMES LIST PRICE
SUBTOTAL
QUANTITY OF LAST ITEM
TIMES LIST PRICE OF LAST ITEM
SUBTOTAL
GRAND TOTAL
GRAND TOTAL TIMES DISCOUNT
SUBTRACT DISCOUNT FROM GRAND TOTAL
STORE RESULT AS NET INVOICE AMOUNT
ADD SHIPPING COSTS
RESULT IS TOTAL AMOUNT OF INVOICE
STOP

Figure 1: Basic flowchart outlining the procedure for completing an invoice. This is a first step in detailing precisely what the program should do.

The Texas Instruments SR-52 is a very impressive machine for its size. Several different libraries of programs are provided that include engineering, finance and games applications. The game programs do, of course, provide the easiest way to impress your friends with the power of the SR-52. This is no different from using game programs with any computer system. For personal use, though, games can quickly lose their glamour. When this happens, it is time to learn how to program.

As the owner of a small business, there are at least two tasks which consume a fair amount of my time. One is payroll and the other is invoice preparation. Of the two, invoice preparation is the easiest program to start with, although both programs offer good opportunities to learn many of the intricacies of programming. These techniques also provide a practical way of learning general programming methods. The result is the development of skills that include analyzing the problem and designing a logical way (flowchart) of identifying the best method for solving the problem. Next comes the conversion of the flowchart to the program language that your machine uses. To do this, let's start with the problem of developing a program to help prepare invoices.

Invoice Program

The first step is to think out what you would do to manually prepare an invoice. There are several ways of doing invoices. We are manufacturers of retail products and sell various quantities of many different items. Our invoices show the suggested list price for each item. After the items are totalled, a discount is calculated and subtracted from the total. The result is the net amount that the customer pays. There are other costs, such as shipping charges, which are also added. Figure 1 shows a flowchart for this first step.

Making a flowchart for this first step may seem to be unnecessary. What it does is give you a graphic description of the steps involved. From this, some of the next steps become more obvious. For example, in an invoice, we multiply one item by another and save the total. This basic step is repeated for as many items as required. This type of repetition is one of the things a computer does best. In the same way, it is the very thing that bores a human fastest.

The end steps require adding the individual totals and getting a grand total. The remaining steps are straightforward arithmetic.

The initial conclusion (one which we sort of knew from the start) is that this problem lends itself to solution by computer.

We will add one more set of calculations so that the final program can handle a second class of merchandise that does not have any discount applied. This addition is similar to the steps shown in figure 1, and we will include it in our next reiteration of the design.

SR-52 Features

What we have done so far has been independent of the system that we will be using. At this point, we have to learn a little about the specific hardware.

The SR-52 allows programs to be input using the calculator keyboard. The individual keys could be considered as a set of

Photo 1: SR-52 programmable calculator.

op codes for a high level language interpreter, with each key performing a set of fairly complex operations. The SR-52 provides several categories of key functions. It has the conventional arithmetic functions: +, –, x, ÷; and a set of memory functions that allow you to manipulate the 20 available memory areas. These functions include STORE, RCL (recall), SUM, PROD (product), and EXC (exchange). Trigonometric functions, Sin, Cos, Tan and other standard functions such as x^2, y^x, and $^1/x$ are also included. An important set of keys are those that allow the SR-52 program to make decisions. These can perform tests on program data in order to determine what the next program step might be. The SR-52 is shown in photo 1.

The SR-52 has a maximum of 224 program steps. To program the machine, you simply take the list of key sequences and push the buttons in that sequence until you are done. This is a little oversimplified, but it really isn't much more than that. If your program has been properly designed, the SR-52 will be able to do all of the finger crunching work when you put in the data. When you are satisfied with the program, you can save it on a magnetic card for future use.

Zeroing In

What we have done up to this point has given us a logical approach to the program solution of our problem, and an overall notion of how the SR-52 can do the job for us. Now we have to get a little closer to the SR-52 to know exactly how to translate our flowchart to key sequences.

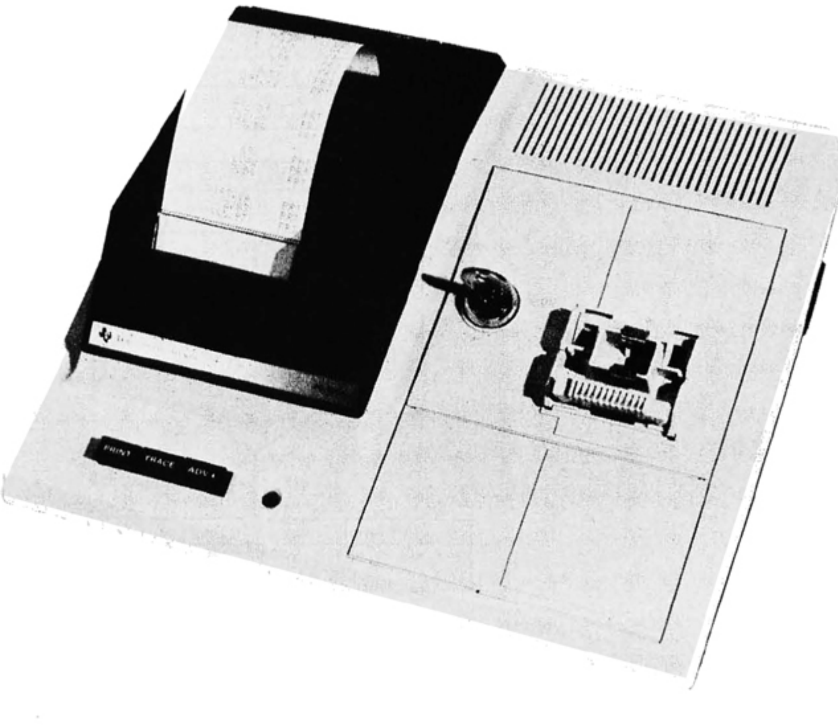
Our program should be easy for a nontechnical person to use. This means that entering the data should use very few steps. The program should do most of the work until the problem (invoice) is complete. To help accomplish this, the program will be written to use the optional SR-52 printer as a recorded output. The printer, shown in photo 2, in addition to providing a hard copy output, actually makes program development easier. By printing desired results as they occur we can avoid using too many of our precious 224 program steps to store and then later recall results. In some cases, programs that would be too long, more than 224 steps, without the printer can be handled on the SR-52. Additionally, since the printer can be operated under program control, some limited data formatting can be accomplished. Last of all, the printer allows the user to verify the accuracy of the data input.

Invoice Program Development

In the invoice problem we are working on, we can see from the initial flowchart that we are going to use several basic SR-52 functions. These are data storing, recalling, multiplying, subtracting, adding and printing.

We will now expand our initial flowchart

Photo 2: Printer attachment for SR-52.



to include the actual steps we will ask the SR-52 to do for us. In this expanded flowchart we have added the second category of merchandise we want to price. One other design consideration that we want to include is the manner in which we will have the user input the data. The SR-52 has a set of five buttons that are labeled A, B, C, D and E. These same buttons can be set for a second use which adds five more functions labeled A′, B′, C′, D′ and E′. These buttons are called "user defined keys." Depending on the way you write the program, a given user button will do whatever you want. For our use, we will want to use these buttons to define to the program what our data means. Table 1 shows the definition we have given to these keys for this program. Looking at the work the SR-52 does when a user defined key is pushed gives you an idea of the power of the machine.

To get the invoice data into the program, the user keys in the numbers on the keyboard and presses the appropriate user defined key.

Now, we can expand our initial flowchart to what we see in figure 2. The numbers in circles adjacent to various boxes are the order in which we will write the program sequence. We will refer to these numbers in the following discussion of the new flowchart.

1. This is some of the housekeeping that any program should have. It assures you that the initial condition of the calculator or computer is what you expect it to be. Here we want to CLR (clear), CMs (clear data memories), FIX 2 (fix the decimal point of the computations to two places) and HLT (halt). We want the machine to stop and wait for us to input data.
2. In this step we are going to define key C to mean discount. We are going to STO (store) it in data memory location 03, print what we entered, advance the paper one line and HLT again.
3. Key D is defined to be the shipping amount of the order; it is stored in data memory 04. We then halt. We could have printed this value, but I chose not to because it will be used at the end of the program where, if it is the wrong amount, not much harm is done.
4. Key A is the quantity of merchan-

Continued on page 152

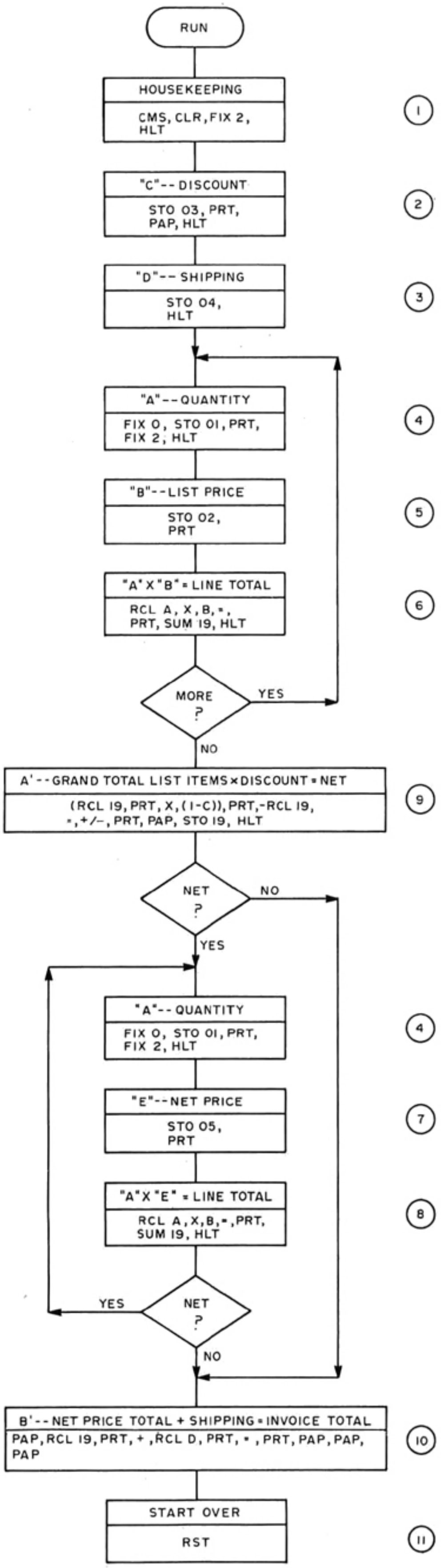

RUN
HOUSEKEEPING
CMS, CLR, FIX 2, HLT
1
"C"-- DISCOUNT
STO 03, PRT, PAP, HLT
2
"D"-- SHIPPING
STO 04, HLT
3
"A"-- QUANTITY
FIX 0, STO 01, PRT, FIX 2, HLT
4
"B"--LIST PRICE
STO 02, PRT
5
"A" X "B" = LINE TOTAL
RCL A, X, B, =, PRT, SUM 19, HLT
6
MORE ?
YES
NO
A'--GRAND TOTAL LIST ITEMS × DISCOUNT = NET
(RCL 19, PRT, X, (1-C)), PRT, -RCL 19, =, +/-, PRT, PAP, STO 19, HLT
9
NET ?
NO
YES
"A"-- QUANTITY
FIX 0, STO 01, PRT, FIX 2, HLT
4
"E"--NET PRICE
STO 05, PRT
7
"A" X "E" = LINE TOTAL
RCL A, X, B, =, PRT, SUM 19, HLT
8
YES
NET ?
NO
B'--NET PRICE TOTAL + SHIPPING = INVOICE TOTAL
PAP, RCL 19, PRT, +, RCL D, PRT, =, PRT, PAP, PAP, PAP
10
START OVER
RST
11

Figure 2: Expanded flowchart for invoice program includes net priced items. The actual programming steps for the SR-52 are also noted within the boxes. The circled numbers indicate the order in which the program was written and refer back to the text.

Continued from page 20

dise purchased. We temporarily FIX 0 to print the quantity as a nondecimal integer and make reading the tape easier. We store the quantity in data memory 01, print, restore the SR-52 to FIX 2 and HLT.

5. Key B is the list price of the product. It is stored in data memory 02 and printed. This time we did not use a HLT instruction, and we will see why in the next step.
6. Without a HLT in step 5, the SR-52 program counter automatically moves down to the next step. It encounters the command to multiply AxB, or more exactly that data we stored in memory 01 times the data we stored in memory 02. We print this result and then sum the result into data memory 19. Since we cleared the SR-52 memories in our initial housekeeping instruction, we know that the initial content of data memory 19 is zero. After all this we HLT.

At this point in our flowchart we are at a manual or human decision point as shown by the diamond. If we have more merchandise to price, we can go back and put in the next quantity A and price B. We would do this until we were done. Then we would go to the A′ button. In our flowchart this is labeled as step 9. Putting A′ near the end of the program may appear to be out of sequence, but the SR-52 doesn't care. It will automatically go looking for A′ program instructions whenever that button is pushed. My programming style says that I like to finish defining the basic user defined key functions before getting involved in the detailed program steps. Since we haven't defined key E yet, that will be the next step. Incidentally, claiming a programming style as a reason for doing something is an easy way of avoiding a lot of "no win" programming arguments. If it's a matter of taste, who can complain?

Since the flowchart represents the actual sequence the user follows, we see that the A′ button is pushed to complete and print the totaling and discount of the list price merchandise. The details of this block will be discussed below. At this point we are only concerned with what we do next. The diamond again indicates a manual decision. Do we have any net price items on this invoice? If so, we will go to A again. This time, button A will be associated with quantities of the net price items. If there are no net price items we will push button B′. This will complete the invoice computation. At this point in the flowchart we define button E.

7. Key E is the net price of the merchandise. We will store this price in data memory 05 and print the price. Again as in step 5, since we don't have a HLT instruction, the SR-52 will automatically proceed to step 8.
8. This step will multiply AxE, or the quantity in data memory 01 times the net price in data memory 05. The result is printed and summed into data memory 19. Incidentally, the previous value of data memory 19 is stored with the net price computation that resulted when button A′ was pushed. The diamond after this block is another manual decision. If there are more net price items go back to A; otherwise go on to B′.
9. This block, at first glance, appears to be complex. But it really isn't. It provides the following instructions:

   RCL 19 Recall the list price amount that was summed into data memory 19 and print (PRT) the amount.
   1–C Multiply the list price total by 1–C, the discount that was inputted, and stored originally with key C. Print the dollar amount of the discount.
   –RCL 19 Subtract the list price amount from the discount; "+/–"; change the sign since the result is a negative number and print the result. This is the net price of the merchandise.
   Pap Advance the paper one line for format clarity.
   STO 19 Store total net price in data memory 19 and HLT.

   Not so bad, was it?
10. We now define the program instructions for the B′ button:

   Pap One line of paper advance.
   RCL 19 Recall and print the total order net price that has been summed in previous

steps into data memory 19.

+RCL D Recall, add and print the shipping costs.

PRT= Total the entire order and print.

Pap, Pap, Pap Advance the paper three lines.

11. **RST** Resets the program counter back to 000. This program address is the first one we used in our housekeeping steps. The program will automatically execute those housekeeping instructions and then HLT. The SR-52 is now ready for a new invoice routine.

The revised flowchart is now complete and at the same time we have defined virtually every program instruction step. The total development of this program has been a very straightforward translation of what we want to accomplish into precise terms understandable to our computer.

The last programming step is to set the SR-52 to receive program instructions. A key marked LRN (learn) is provided for this. Several other programming keys are also provided which dramatically simplify putting a new program into the machine. At this point we will only need the LRN button. With the SR-52 in the LRN mode, we begin with the first instruction which is CMs (clear memories). We push that button on the keyboard. The program counter moves to 001 and we push the CLR button. The SR-52 stores the clear instruction and the program counter moves to 002. We continue pushing the buttons that we defined in our program until we reach the last instruction which is RST. Our program is now stored in the machine. We take the SR-52 out of the learn mode by pressing the LRN button again. The program counter is manually reset to 000. Listing 1 shows the entire program list.

We are now ready for the critical test. The RUN button is pressed and, if the display is not flashing, there is a good chance that our program might work. We test it by putting in a sample calculation. If it does work, our tape printout would look like listing 2. If there are problems, the printer will be a great help in finding the bug.

The first thing that can be done is to ask for a list of the program steps. This output list can be compared to the work sheet we used to enter the program. Analyzing the program list shown in listing 1 is made easier if we mark each program address line with the key function name. *[In listing 1 this is done with typeset notations, but in SR-52 practice this would be done with handwritten notations ...CH]* In addition to doing this, groups of instructions should be bracketed so that they can be related to the flowchart in figure 2. The first three digits in the program list are the program counter address. The next two digits are the op code or the key codes for the SR-52 keyboard. If the program doesn't work, you may have omitted an instruction. The SR-52 provides an easy way to get to the point of omission so that we can insert the proper instruction. In a similar way, an incorrect instruction can be omitted.

Listing 1: Program listing for the SR-52 invoice program. The program is broken into sections corresponding to the flowchart of figure 2.

```
                      PROGRAM  KEY
                      COUNTER  CODE
             CMS       000     47
             CLR       001     25
HSKPG        *FIX      002     57
 ①           2         003     02
             HLT       004     81
             *LBL      005     46
             C         006     13
             STO       007     42
DISCOUNT     0         008     00
 ②           3         009     03
             *prt      010     98
             *pap      011     99
             HLT       012     81
             *LBL      013     46
             D         014     14
SHIPPING     STO       015     42
 ③           0         016     00
             4         017     04
             HLT       018     81
             *LBL      019     46
             A         020     11
             *FIX      021     57
             0         022     00
QUANTITY     STO       023     42
 ④           0         024     00
             1         025     01
             *prt      026     98
             *FIX      027     57
             2         028     02
             HLT       029     81
             *LBL      030     46
             B         031     12
LIST         STO       032     42
PRICE        0         033     00
 ⑤           2         034     02
             *prt      035     98
             x         036     65
             RCL       037     43
             0         038     00
LINE         1         039     01
TOTAL        =         040     95
 ⑥           *prt      041     98
             SUM       042     44
             1         043     01
             9         044     09
             HLT       045     81
             *LBL      046     46
             E         047     15
             STO       048     42
NET          0         049     00
PRICE        5         050     05
 ⑦           *prt      051     98
             x         052     65
             RCL       053     43
             0         054     00
             1         055     01
LINE         =         056     95
TOTAL        *prt      057     98
 ⑧           SUM       058     44
             1         059     01
             9         060     09
             HLT       061     81
             *LBL      062     46
             *A'       063     16
             *pap      064     99
             (         065     53
             (         066     53
             RCL       067     43
             1         068     01
             9         069     09
             *prt      070     98
A'           x         071     65
LIST         (         072     53
PRICE        1         073     01
TOTAL        -         074     75
DISC AMT,    RCL       075     43
             0         076     00
NET          3         077     03
 ⑨           )         078     54
             )         079     54
             *prt      080     98
             -         081     75
             RCL       082     43
             1         083     01
             9         084     09
             )         085     54
             +/-       086     94
             *prt      087     98
             *pap      088     99
             STO       089     42
             1         090     01
             9         091     09
             HLT       092     81
             *LBL      093     46
             *B'       094     17
             *pap      095     99
             RCL       096     43
             1         097     01
             9         098     09
B'           *prt      099     98
TOTAL NET,   +         100     85
SHIPPING,    RCL       101     43
INVOICE      0         102     00
AMT          4         103     04
 ⑩           *prt      104     98
             =         105     95
             *prt      106     98
             *pap      107     99
             *pap      108     99
             *pap      109     99
 ⑪           *RESET    110     86
                       111     00
                       112     00
                       113     00
```

The most painful debugging problem is where your program logic just isn't right. The printer will allow you to trace and print the steps the program actually is following as it attempts to solve the problem. (In our case, calculate the invoice.) If you are not too far off, the SR-52 has the facilities

```
DISCOUNT      0.60    PRT

QUANTITY      12.     PRT
LIST PRICE    5.63    PRT
LINE TOTAL   67.56    PRT
QUANTITY      18.     PRT
LIST PRICE    1.87    PRT
LINE TOTAL   33.66    PRT
QUANTITY      34.     PRT
LIST PRICE    5.63    PRT
LINE TOTAL  191.42    PRT

TOTAL LIST  292.64    PRT
DISC AMT    117.06    PRT
NET TOTAL   175.58    PRT

QUANTITY      18.     PRT
NET PRICE     0.42    PRT
LINE TOTAL    7.56    PRT

TOTAL NET   183.14    PRT
SHIPPING      2.83    PRT
INV. AMT    185.97    PRT
```

Listing 2: Sample printout of the invoice program.

that allow you to easily implement your program fix.

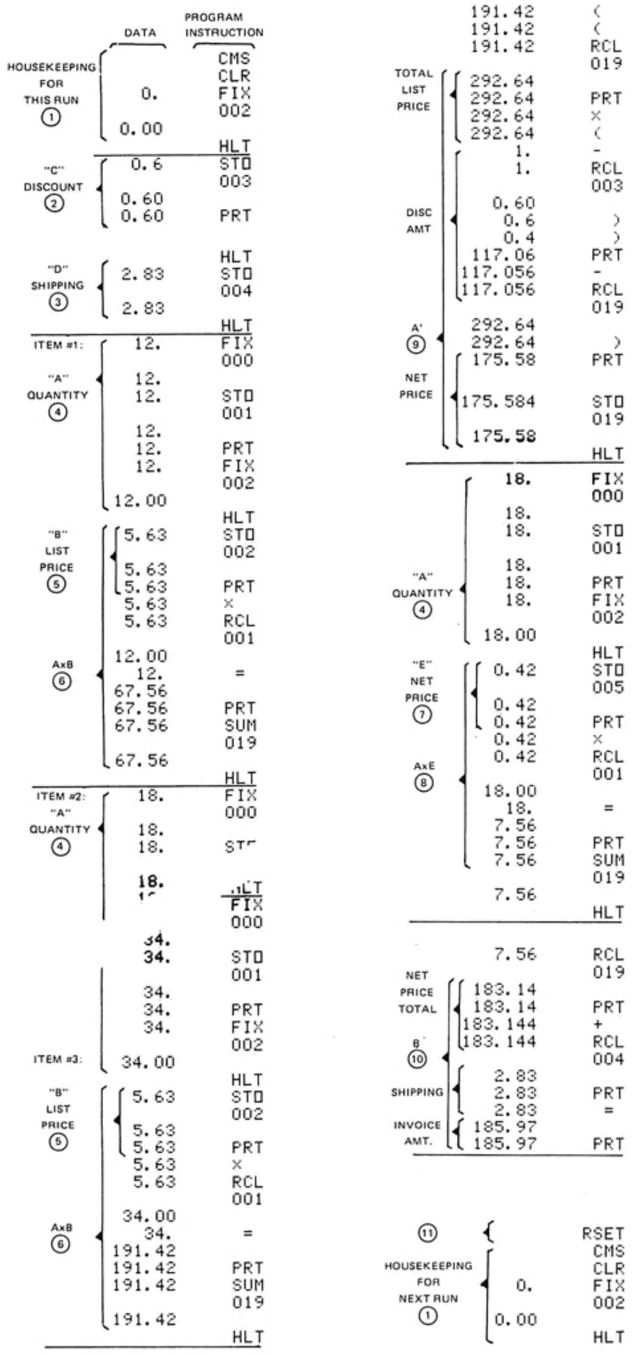
Listing 3 shows the printer output in the trace mode. Here the left digits are the printed equivalent of the SR-52 display. The right side data is the function the SR-52 is performing on the display data.

The data shown in listing 3 is the same data used for the printout shown in listing 2. Marking and grouping the trace data in the way shown will simplify comparing the trace output with the design flowchart of figure 2. Each step in the trace should match what you designed in the flowchart. If it doesn't, then the machine is being asked to do something incorrectly. If your flowchart logic is faulty, review the SR-52 instruction book before making corrections. If your flowchart design has an improper premise, then the calculator will indeed proceed with perfect logic to an improper result.

Somewhere between the two "dump" methods, list and trace, lies the identification to the program problem. Listing shows program loading mistakes and tracing will reveal program logic errors.

I could go out on a limb with the truth and say that my invoice program was up and running on the first pass. But seriously, with the debugging facilities already discussed, it wasn't too difficult to find and correct the problem areas.

After the program has been debugged, you should make a new program list. Any flowchart changes that were necessary should also be documented. If you don't do this documentation, you will find that your memory will fail you miserably if you want to modify the program sometime in the future. The up-to-date list will also let you reproduce the program in case the magnetic program card ever becomes unusable.


DATA
PROGRAM INSTRUCTION
HOUSEKEEPING FOR THIS RUN ①
"C" DISCOUNT ②
"D" SHIPPING ③
ITEM #1:
"A" QUANTITY ④
"B" LIST PRICE ⑤
AxB ⑥
ITEM #2:
"A" QUANTITY ④
ITEM #3:
"B" LIST PRICE ⑤
AxB ⑥
TOTAL LIST PRICE
DISC AMT
A' ⑨
NET PRICE
"A" QUANTITY ④
"E" NET PRICE ⑦
AxE ⑧
NET PRICE TOTAL
B' ⑩
SHIPPING
INVOICE AMT.
⑪
HOUSEKEEPING FOR NEXT RUN ①

```
            CMS
            CLR
    0.      FIX
            002
  0.00
            HLT
   0.6      STO
            003
  0.60
  0.60      PRT

            HLT
  2.83      STO
            004
  2.83
            HLT
   12.      FIX
            000
   12.
   12.      STO
            001
   12.
   12.      PRT
   12.      FIX
            002
 12.00
            HLT
  5.63      STO
            002
  5.63
  5.63      PRT
  5.63      ×
  5.63      RCL
            001
 12.00
   12.       =
 67.56
 67.56      PRT
 67.56      SUM
            019
 67.56
            HLT
   18.      FIX
            000
   18.
   18.      ST⌐
   18.      .ıLT
   1⌐       FIX
            000
   ɜ4.
   34.      STO
            001
   34.
   34.      PRT
   34.      FIX
            002
 34.00
            HLT
  5.63      STO
            002
  5.63
  5.63      PRT
  5.63      ×
  5.63      RCL
            001
 34.00
   34.       =
191.42
191.42      PRT
191.42      SUM
            019
191.42
            HLT
```

```
 191.42     (
 191.42     (
 191.42     RCL
            019
 292.64
 292.64     PRT
 292.64     ×
 292.64     (
     1.     -
     1.     RCL
            003
   0.60
    0.6       )
    0.4       )
 117.06     PRT
117.056     -
117.056     RCL
            019
 292.64
 292.64       )
 175.58     PRT

175.584     STO
            019
 175.58
            HLT
    18.     FIX
            000
    18.
    18.     STO
            001
    18.
    18.     PRT
    18.     FIX
            002
  18.00
            HLT
   0.42     STO
            005
   0.42
   0.42     PRT
   0.42     ×
   0.42     RCL
            001
  18.00
    18.      =
   7.56
   7.56     PRT
   7.56     SUM
            019
   7.56
            HLT

   7.56     RCL
            019
 183.14
 183.14     PRT
183.144     +
183.144     RCL
            004
   2.83
   2.83     PRT
   2.83      =
 185.97
 185.97     PRT


            RSET
            CMS
            CLR
     0.     FIX
            002
   0.00
            HLT
```

Listing 3: Another sample listing; this one is in trace mode. Trace mode outputs each program instruction and the data associated with it so you can check each individual program step. This is the SR-52's equivalent of a symbolic high level language in a larger machine.

Table 1: Functions of the user defined keys in the invoice program.

Key Name	Assignment when Button is Pushed
A	Data entered on key is the quantity of the merchandise purchased.
B	Data entered is the suggested list price of the merchandise. (Used with A.)
C	Data entered is the customers' discount.
D	Data entered is the shipping charge for the order.
E	Data entered is the net price (no discount) of the second category of merchandise. (Button A is also used with E.)
A'	This is the second function use of A. When pressed it tells the SR-52 to total the list price items, compute and subtract the discount and leave the net amount.
B'	This is the second function use of B. When pressed it tells the SR-52 to add the total of the net price items, if any, to the results obtained when A' was pushed. It then adds the shipping cost and results in the total for the order.

You should also write the user instructions at this time. Table 2 shows the user instructions for the invoice program. It is also helpful to the user if you attach a copy of a sample printout. The blank spaces on the magnetic program card should also be filled.

This invoice program, while being relatively simple, is certainly not a trivial program. It does involve many of the basic programming philosophies that would be applicable to writing a program for any system. With the prices of hand held calculators continuing to drop as they have in the past, the SR-52 and companion printer are not a bad way to get into the intricacies of computers. At the time of this writing, the combination can be bought for about $400. The fact that the SR-52 can be connected to the real life world (the printer), does hold a possible promise that a computer interface to the SR-52 might come along someday.

For those who have an SR-52 and need an invoice program, a listing of the program discussed in this article is provided in listing 1. The program uses 110 program steps, less than half of the total SR-52 capacity. This will allow space to modify the program to meet your specific needs.■

Step	Procedure	Press			Display
1	Load "A" side of card	CLR	2 nd	READ	Goes blank then 0
1a		run			
2	Enter discount (see note below)	C			Discount amt printed
3	Enter shipping	D			Shipping amt
4	Item quantity	A			Qty printed
5	Total list price of item Price printed Repeat 4 and 5 until all List price items entered	B			Individual and total price printed
6	Total of list items Prints list total Discount, net on list If there are net price items do 7 and 8 otherwise go to step 9	2 nd	A'		
7	Enter item quantity	A			Qty printed
8	Enter net item price Total net price printed Repeat 7 and 8 until done	E			Individual and total price printed
9	Net total printed, Then shipping and Invoice total printed	2 nd	B'		

Note: For 40% customers, enter .6;
For 50% customers, enter .5;
For 50+10+10, enter .405.

Table 2: User instructions for the invoice program. It is always necessary (for peace of mind) to write out these instructions, otherwise you may forget how to use a program.