NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INVESTIGATION OF CARD PROGRAMMABLE AND
CHIP PROGRAMMABLE POCKET CALCULATORS
AND CALCULATOR SYSTEMS FOR USE AT
NAVAL POSTGRADUATE SCHOOL AND
IN THE NAVAL ESTABLISHMENT

by

Harry Rudolph Kruse
and
Hugh Alan Burkett

March 1977

Advisors: H.J. Larson
          R.H. Shudde

Approved for public release; distribution unlimited.

T178061

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Investigation of Card Programmable and Chip Programmable Pocket Calculators and Calculator Systems for Use at Naval Postgraduate School and in the Naval Establishment		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1977
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Harry Rudolph Kruse Hugh Alan Burkett		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1977
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Hand-held calculators
Card programmable calculators
HP-67   TI-59
HP-97   NS-7100
SR-52

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis investigates the usefulness of card programmable pocket calculators in the Management curricula of the Naval Postgraduate School and in the fleet, based upon manufacturer-provided information on the HP-67, HP-97, SR-52, TI-59, and NS-7100 calculators; NPS classroom experimentation; "hands on" programming of the HP-67 and SR-52; interviews; and the literature. All aspects of calculator

(20. ABSTRACT Continued)

functions, programming and programmability are surveyed
with particular emphasis on educational and practical
applications. Thus, this is a baseline document for study
by potential purchasers and users. This study concludes
that these machines provide significant advantages in
teaching or learning mathematical concepts and that the
pocket calculator is a potentially important management
and tactical support tool navy-wide. In addition,
"thinking process transmutation," discovered during this
study, is concluded to be an inevitable and important
by-product of calculator programming which significantly
improves the user's overall analytic capacity.

Investigation of Card Programmable and
Chip Programmable Pocket Calculators
and Calculator Systems for Use at
Naval Postgraduate School and
in the Naval Establishment

by

Harry Rudolph Kruse
Lieutenant Commander, United States Navy
B.S., University of Arizona, 1960
L.L.B., LaSalle University, 1970

and

Hugh Alan Burkett
Lieutenant Commander, Civil Engineer Corps, United States Navy
B.S., University of Oklahoma, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL

March 1977

ABSTRACT

This thesis investigates the usefulness of card pro-
grammable pocket calculators in the Management curricula
of the Naval Postgraduate School and in the fleet, based
upon manufacturer-provided information on the HP-67,
HP-97, SR-52, TI-59, and NS-7100 calculators; NPS classroom
experimentation; "hands on" programming of the HP-67 and
SR-52; interviews; and the literature. All aspects of
calculator functions, programming and programmability are
surveyed with particular emphasis on educational and prac-
tical applications. Thus, this is a baseline document for
study by potential purchasers and users. This study con-
cludes that these machines provide significant advantages
in teaching or learning mathematical concepts and that the
pocket calculator is a potentially important management and
tactical support tool navy-wide. In addition, "thinking
process transmutation", discovered during this study, is
concluded to be an inevitable and important by-product of
calculator programming which significantly improves the
user's overall analytic capacity.

TABLE OF CONTENTS

5

-

ERRATA

Page	Line #/Item	Should Read
14	21 - after "document"	(Page 37)
23	21 - after "often"	do not
25	3 - along	among
26	2 - after "materials"	[8]
44	16 - IV.E(9)C.(8)	IV.E(9)c(9) on page 108
45	18 - +7 = & 7+	÷ 7 = & 7 ÷
47	9	delete AOS
58	4 - DEC DEG	DEC → DEG
	8 - DEG DEC	DEG → DEC
	10 - DEG RAD	DEG → RAD
63	6 - X Y	X ⇄ Y
76	"user keyed labels" blocks a, c, e	add N'; r; n
77	Step 1 key entry	fLBLA
	Step 65 key entry	RCL4
81	"user keyed labels" diagram	draw <u>lines</u> <u>only</u> to match <u>page</u> 76
89	"user keyed labels" (E')	$\pm$ Δ i → DO NPV
	Flags: Block 0	insert "if $n < N$"
	Step 1 - call	<u>call</u>→
	Step 6 -	$\underline{\Delta i}$→
90	Step 37 - +PVn REG 69	+PV → REG 69
	Step 54 - if n = N	if n = N GTO STEP 193
91	Step 100-/i_t/REG	/i_t/ → REG 98

Page	Line #/Item	Should Read
92	Step 131 - key entry .∅	9
	note· Δi REG	Δi → REG 96
93	Step 182 - REG GO	REG 60
	Step 191 note (Clr) REG 99	(CLR) → REG 99
94	"user keyed labels" - e Δi Do	± Δi → DO NPV
	Step 18 key entry	hR↑
	Step 20 note	← G TO LBL 1
95	Step 44 note (#n) REG	#n → REG
	Step 46 note (N+1) REG	N+1) → REG
96	Step 83 note LBL C	LBL c
	Step 96 key entry h R	h R↑
	Step 98 key entry h R↑	h R↓
	Step 118 note i REG	i → REG
97	Step 135 note -11) REG	-11) → REG
	Step 148 note it) REG	it) → REG
	Step 166 key entry	h ST I
	Step 175 note -26) REG	-26) → REG
99	Add title top of page	(Initial Data Entry Operations
102	7 - (i_t stored)	(+i_t stored)
	15 - (n· 0) & (n 20)	(n < o) and (n > 20)
106	5 - Display column	0.0639
	13 - Press key column	B

2

Page	Line #/Item	Should Read
109	21 - after program	program card
113	23 - IV.C	IV D
121	1 - on	or
	13 - Figure 2	Figure 3
123	9 - one equally	are equally
	13 - Figure 2	Figure 3
125	5 - Figure 5	Figure 6
128	5 - 5.	6.
135	10 - Figure 6	Figure 7
137	2 - Figure 6	Figure 7
138	2 - after "Explanation" add	of Figure 7 (page 142)
141	2 - Figure 7	Figure 8
142	add 3rd line (above C')	Card 1
	"user keyed labels" block D aij REG	aij $\rightarrow$ REG
	Step 4 add note	(Cj= 90+j $\rightarrow$ REG 99)
	Step 26-29 note	bracket 1, 11, 91, 84 RF
143	Step 71 - Run	Run $\rightarrow$
	Step 79 - aij	aij $\rightarrow$
	Steps 96, 100, 104- cost	cost $\rightarrow$
145	add title (above C')	Card 2
	"user keyed labels (C')" - REG ØØ	REG $\rightarrow$ 00
	(D) K PR	K $\rightarrow$ PR
	(E) K PR ROW K-PR	K $\rightarrow$ PR $\rightarrow$ ROW K $\bullet$ PR

Page	line#/item	should read
145 (con't)	Step 7 - under K *note K PR*	$\dfrac{K}{K \to PR} \to$
	Step 11 - P_{ROW}	$P_{ROW} \longrightarrow$
146	Step 36 - RUN	RUN $\longrightarrow$
	Step 66 - $\overline{P_{ROW}}$	$P_{ROW} \longrightarrow$
148	Step 1 - key entry	g^{LBLa}
149	Step 32 key entry	g^{FRAC}
154	15 - (Table 3)	(Figure 3)

4

I. INTRODUCTION

A. GENERAL

Card programmable electronic pocket calculators offer portable computational power and versatility heretofore unknown at an astonishingly low cost. Most problems which require the analysis of numbers can be programmed into and automatically solved by this new breed of machine at the touch of a few buttons. This new ability to execute complex computations, on demand, in the field, and in seconds can enable one to significantly improve his professional efficiency in almost any profession. In the opinion of the authors, the next generation of card programmable calculators is going to have a greater ultimate effect on the modern world than did the computer itself in recent years.

The invention of the logarithm by Napier in 1614 [1] simplified and speeded mathematical computations by mankind. The calculator revolution of today is comparable to the revolution in mathematics brought about by the logarithm. The non-programmable electronic calculator has already all but eliminated the market for the sliderule while providing greater accuracy at a lower cost. On the other hand, sophisticated calculators are now beginning to compete with larger computers. The first card programmable calculator, the Hewlett-Packard HP-65, was introduced in 1974. In January of 1976, Texas Instruments Incorporated (TI)

9

introduced the SR-52 card programmable calculator in competition with the HP-65. Hewlett-Packard (HP) answered the challenge in June 1976 with the HP-67 which provided more than twice the capability of the HP-65. National Semiconductor intends to introduce their Model 7100 before summer of 1977. It will have 480 user programmable steps plus 4000 library steps. In June 1977, TI will replace the SR-52 with the TI-59, which will have as many as 960 user programmable steps plus a 5000 step library cartridge. This relatively rapid evolution in hand-held calculators is the result of high consumer demand, and market competition. It is believed by the authors that hand-held computers with up to 20K of step processing capability will be available within the next decade.

B. PURPOSE

The beginning for this thesis started with the purchase of programmable calculators (HP-25's) for use by the authors in course work at the Naval Postgraduate School (NPS). Having gained the ability to program the purchased calculators, it was soon determined that more capability was desired in order to solve more complex problems. This phenomenon is described by Thompson [2] as the "fill-up" principle where the user tends to "fill-up" the capability of the machine and thus, outgrow it.

In 1976, NPS purchased more than sixty SR-52 machines for use in a pilot project designed to determine if such

machines should be formally incorporated into the various curricula. Additionally, the NPS faculty questioned whether or not the use of these calculators in the fleet could improve the capability of managers, i.e., if rapid and concise analysis of data could enable managers to arrive at better decisions which might lead to improved operational readiness.

It is, therefore, the purpose of this study to investigate the potential of card programmable pocket calculators as a tool for both educational endeavors and management support in conjunction with the NPS pilot project.

C. STUDY METHODOLOGY

(1) The Management Science curricula of the Naval Postgraduate School (NPS) was used for educational program guidelines. NPS Management Science students come from diverse educational backgrounds; many of the students have minimal formal mathematical training. The Management Science curriculum encompasses the following disciplines that include numerical computation:

      a. Mathematics

      b. Probability

      c. Statistical Methods

      d. Financial Accounting

      e. Management Accounting

      * Particular courses are listed in Appendix A.

11

(2) The Hewlett Packard HP-67 and the Texas Instruments
SR-52 were studied "hands on" and compared on the basis
of:

        a. Machine language

        b. Programming and programmability

        c. Machine functions and capabilities

        d. Use in the educational process

        e. Use at NPS and in the Fleet.

Comparable functions and capabilities of the NS Model
7100 and the TI 59 are listed in Appendix B. Samples of
these two machines were not available during the time frame
of this study; therefore, listed data is informational
only, as provided by the manufacturers. The authors had
opportunity only to observe factory emulators which could
accomplish most of the functions predicted for the actual
production models of these machines.

(3) Additional information was collected through:

        a. Interviews and surveys of users

        b. A search of the literature

        c. Telephonic and written interview of
           researchers and educators

        d. Review of written information provided by
           educators and researchers

        e. Review of information provided by manufacturers.

(4) There is little in the literature to indicate
the impact of card programmable calculators in the educational

and fleet environments. This thesis is written with the expectation that it will be disseminated to readers who have various levels of familiarity and needs concerning these calculators. These underlying factors necessitated a baseline study that includes technical information, as well as non-technical, as deemed important to the various audiences. Therefore, the reader may extract information as considered appropriate.

II. CARD-PROGRAMMABLE CALCULATORS AND EDUCATION

A. THE CALCULATOR AS A TEACHING AID?

Whether or not the calculator is usable as a teaching aid is a multi-faceted question.

Rogers [3] believes that a good teaching aid must be enduring if it is to be of value in the educational process; i.e., as enduring as the blackboard. She lists four features that seem to separate enduring teaching aids from others:

1. Inexpensive and/or durable

2. Controllable by learner

3. Does what the learner wants done

4. Flexible usability

The question of expense and durability is simply one of value analysis. Such analysis must necessarily evaluate (a) initial incremental cost, (b) initial support cost, and (c) recurring cost. Support costs include publications and peripheral equipment required to take full advantage of the system. Recurring cost of calculators is primarily a function of machine failure rates. A suggested method of value analysis on this subject is located elsewhere in this document.

A hand held calculator is certainly controllable by the student, particularly when the student is allowed to take it home with him. Equally obvious is that the machines surveyed in this thesis are usable in almost any course of instruction

14

that requires numerical computation and are capable of doing what the student wants them to do. In addition, these instruments can be used to great advantage in the fleet after graduation. In short, hand held calculators have an advantage that even the blackboard cannot compete with: portability.

Bell [4] points out that there is considerable agreement that calculators should play an important role in the educational process as a result of their availability and use outside the world of schools. He also reports that almost invariably there is high initial interest which persists over a long time provided students are given interesting things to do with the calculators.

Suydam [5] lists the two fundamental arguments regarding the use of calculators in the educational process in general:

Proponents argue:

"The hand-held calculator is the tool used in society today for calculations. Schools are 'burying their heads in the sand' if calculators are not recognized and used as the calculational tool that they are."

Opponents argue:

"The principal objectives of mathematics instruction (at least in grades K-9) are that children learn the basic facts and paper-and-pencil algorithms. Such learning will not occur if calculators are made available in schools."

The opposing view is also argued at the college level, although calculators are not forbidden in most college classrooms and numerous schools have taught a basic sliderule

course. It is agreed that a certain amount of computational skill must be required before calculators can be used entirely effectively. But the authors do not see much distinction between "paper-and-pencil" and calculators, and even less between sliderules and calculators. I.e., what is the difference between using paper-and-pencil and using calculators in the learning process? At best, it would seem to be a moot point.

Available literature indicates to the authors that the benefits of using calculators strongly outweigh not using calculators. Interestingly, no evidence has been found that calculators or sliderules negate the learning process. Paper and pencils are merely different forms of tools. Any tool that facilitates learning should be accepted on its relative merits rather than being eliminated through emotional recrimination. To put it simply, correct answers instill confidence and it is inescapable that learning will take place while using calculators. The degree of learning depends upon the student, the teacher and how well the tools available are used. Calculators will not replace the thinking process but will enhance its capabilities to accomplish more, as did the sliderule for the engineering profession.

The authors find no substantial reason against using calculators as an educational tool per se and fully support the proponents viewpoint.

16

Peripherally, an additional aid is on the market for use by instructors — the "Edu-Calc." It is manufactured by Educational Calculator Devices, Inc., in Laguna Beach, California. It is an electronic repeater unit coupled with a calculator, produced as an integral unit. With a calculator on the top of the unit for operation by the instructor, the display faces the students for them to follow.

This integral unit is supplied in a brief case, which makes it easy for the instructor to carry it with him from class to class.

At present, the unit is manufactured using only Hewlett-Packard calculators. When queried as to the possible incorporation of other manufacturers calculators, Mr. George Schultz, Manager, Academic Sales, stated that such an eventuality would not be considered at this time due to machine reliability factors. No "Edu-Calc" machines have been returned for repair during the life of their production (approximately one year). He provided a rough estimate that an HP-67 model "Edu-Calc" could be made available for approximately $1200 [6].

B. THE NPS PILOT PROJECT: CLASSROOM USE OF CARD
    PROGRAMMABLE CALCULATORS

The project began with the issuance of SR-52 card programmable calculators to fifteen beginning students in the Naval Intelligence curriculum. The calculators were incorporated in the course "Mathematics for Naval Intelligence" (MA 2310); this course is described as [7]:

"A review of linear, logarithmic, sinusoidal and exponential functions, with graphical emphasis; differentiation and integration with both analytical and numerical procedures, continuation to include introductory treatments of Fourier analysis; the Fourier integral, spectral analysis, differential equations, and the Laplace transformation."

The course was completed during the October-December 1976 quarter.

In addition to the calculators, students were issued a text [8] that had been adapted by the instructor to take best advantage of the calculators. The students received 5 hours of classroom lectures and 3 laboratory hours each week for 11 weeks.

During the first two weeks of the quarter, the class concentrated on learning machine capabilities and programming. The balance of the quarter was spent learning mathematics.

The students completed survey forms during the fourth and twelfth weeks of the quarter and were interviewed throughout the quarter. These students will continue to use the calculators in follow-on courses.

Following is an analysis of available information:

a. Ages of students: 29 ± 5

b. Previous degrees: BA/BS   87%

c. Highest level math previously taken:

       college algebra   40%

       calculus   46%

       differential
       equations   14%

d. No student had previous experience in programming calculators or computers.

e. Average use of calculators per student over 11 week period: 97.08 hours (37.8 hrs without programming; 59.28 hrs programming and using programs).

f. 100% of the students believe that the calculator oriented course was a worthwhile educational endeavor that has enabled them to acquire an ability useful in further education.

g. 93% of the students foresee the use of programmable calculators in future billets (versus 67% in the 4th week) and are generally willing to buy a calculator (versus 73% in the 4th week). However, that willingness is predicated on (a) billet requirements and (b) the belief that if the government benefits, the government should purchase them.

h. 93% of the students believe that using a programmable calculator and programming has helped them to understand mathematical concepts (as compared to 53% at the 4th week).

i. 100% of the students rate handout materials better (each student ranked them 4 on a scale of 1 to 5) than manufacturers publications. They also believe that these materials were of significant help in learning mathematical concepts.

j. 20% of the class experienced machine failures
    (machine would not print cards or would not accept
    manual program input). Not only did these students
    rate programming lower as a learning mode, but they
    also received the lowest grades in the class.

k. 87% of the students believe a printer would be
    useful in programming.

l. Machine features that the students particularly
    liked included its card-programmability; 11 particu-
    larly disliked its inability to exponentiate
    negative numbers (Y^X) without indicating an error
    condition.

m. It is noted that most answers concerning questions
    about future applications were guarded. Students
    stated that they did not know future course
    requirements, therefore, were not sure of particular
    uses.

n. It is clear that the students perceive a strong
    potential for the application of programmable
    calculators in future billets.

o. Comparison of survey/interview results at the fourth
    week with those of the twelfth week indicates a
    strong increase in enthusiasm and confidence in
    the ability to carry out computations of increasing
    complexity. One particularly common statement
    among the students was that they were able to

compute much faster and solve many more problems, and, therefore, were able to devote more time to studying theory. All students have the desire to learn more about the machine, programming and applications. Some students are already writing programs for future uses.

Each student interviewed indicated that he now views algorithms in a different way; in the way of a flowchart or the way it could be programmed most efficiently. In addition, each confirms that he now tends to think about problems outside mathematics in a similar manner.

p. This class began the quarter with the same text material as covered by previous classes (except for adaptation of the material to the calculator-teaching mode). The instructor, after reviewing student records, initially estimated that this particular class would have difficulty in completing the text (records of this class in prior mathematics courses indicated an overall grade point average of 2.0 or C). However, he found that the class had completed the text during the ninth week and he was able to include additional materials that would prove very useful in future courses. He observed that students gained a great deal of insight and intuition through writing programs and solving

repetitive computations. He stated that the major advantages of calculator usage included "breaking the ice", students obtained correct answers, developed confidence and learned mathematics extremely well. Not only were the students able to cover significantly more material (15% more depth), they were also able to complete a final examination that the instructor judged to be 20% more difficult (complex) than any previous examination in this course.

q. The class of the previous quarter, without benefit of card programmable calculators, attained an overall grade point average of 3.62 ± .43. This class attained an overall grade point average of 3.60 ± .44.

r. As a result of the success experienced by this class, the instructor began the following quarter (second quarter for the students) by teaching the students to compute cumulative/inverse, cumulative normal and binomial distribution values rather than referring to tables. He has also taught the Runge-Kutta method for solving differential equations, carrying out fifty cycles during a class period. Normally these methods are too time consuming to be demonstrated past the "exposure" level. Accordingly, it certainly appears that card programmable calculators have provided a significant advantage to this class.

s.   The mathematics courses in the Naval Intelligence
     curriculum are similar to those in the Management
     Science curriculum. Therefore, the information
     collected from this class is entirely adaptable to
     the following discussion concerning the Management
     Science curricula.

C.   PROPOSED USE OF PROGRAMMABLE CALCULATORS IN THE NPS
     MANAGEMENT SCIENCE CURRICULA

Appendix A lists the courses that are considered to be
logical candidates for the incorporation of card programmable
calculators in the teaching process. The investigators
believe that each of these courses could be taught in a much
more efficient manner by adapting the computational portions
to algorithms, demonstrating the algorithms to the students
and having the students program them, and, finally, having
the students carry out several exercises for practice and
application.

The authors believe that the essence of the educational
process lies in being able to apply knowledge once gained.
Unfortunately, most students seem to do well in the class-
room but all too often not really know how to apply their
new capabilities.

It is axiomatic that an acquired skill will become
"rusty" or will be lost, if not exercised. However, that
tendency is ameliorated when a student is made aware of
when and how to use a particular algorithm to solve a

particular problem. In other words, teach the student how to recognize a problematic situation, show him how to break it down into its components, analyze each component and reconstruct the situation to efficiently accomplish the task.

The NPS Management Science curriculum includes a wide variety of disciplines, each with its own procedures and techniques. The generally accepted method of instruction combines classroom lectures concerning theory with several hours of homework "crunching numbers." Realistically, little time is left available in the classroom to review homwork problems or theory. Consequently, if the student does not understand some aspect of the course, he is left substantially to his own devices to completely comprehend theory, numerical manipulation or both. A student who misses a critical point early in the quarter may suffer the balance of the quarter by not understanding an important aspect in the progression of course material. Given that the student is carrying more than one course, this effect may take place in several areas. The effect can then become pryamidal with the student demonstrating less than that of which he is capable. Concomitantly, some students simply cannot manipulate numbers rapidly, in spite of the fact that they fully comprehend the material. These students appear to be "poor students" at examination time, yet may be as capable as most any student in the long run.

This investigation has led the authors to believe that all of these effects could be ameliorated to a great degree, with significant gains along all students, through incorporation of the card programmable calculator in NPS Management Science curricula as well as other NPS curricula. In addition, the student could take several programs (recorded on cards) with him to the fleet for use in his next billet.

D. A RECOMMENDED COURSE STRUCTURE

Based upon the pilot project, the authors believe unequivocally that a course of instruction in card programmable calculators can lead to a significantly improved Naval Postgraduate School product and that a beneficial effect would carry over into the fleet.

Such an initial course should be structured in the following manner:

a. Offered during the first quarter of study in the undergraduate portion of the curriculum.

b. The course should be no less than two two-hour classroom periods per week for four weeks, to become proficient with the calculator, followed by two one-hour classroom periods during the remaining seven weeks for applications.

c. The course should be designed to include the use of all functions of the calculator coupled with flow charting and programming.

d. It should be provided as an initial course in mathematics.

Professor Gaskell, NPS Mathematics Department, has already developed text materials that could provide a basis for the accomplishment of these goals. Basically, his programmed text incorporates the teaching of algorithms commensurate with teaching applications and the full capability of the machine.

Obtaining this education early in the curriculum would enable the student to allay the fear and drudgery of numerical analysis. He would be much better prepared for advanced mathematics and the early computer programming courses. CDR Gibfreid, Chairman, NPS Computer Science Department [9], believes that a course in programmable calculators would enable the doubling of the length of time now available for teaching management information systems (MIS) applications.

E. THINKING PROCESS TRANSMUTATION: AN IMPORTANT BY-PRODUCT

Another important effect which would be developed through the recommended course, however, is considered to be even more powerful and far-reaching: This is the effect of thought process transmutation which occurs as an almost inescapable by-product during some four to six weeks of such study.

The process discovered during this research, occurs in the following manner:

(1) Initially, the neophyte programmer is merely impressed with the capabilities of the machine; he attempts to rush into programming without sufficiently studying the

operators manual. His initial success is thus limited, so he begins to conscientiously study the manual to determine how the machine operates.

(2) As his ability to use the machine improves, he discovers the various methods of flow charting. He now has a visual aid that he developed, enabling him to see the program in graphic form.

(3) Once he has developed some amount of expertise in flow charting and programming, he begins to visualize formulas and processes in a new way. For all practical purposes, his analytical thinking processes take on the characteristics of a flow chart or program. As he gains even more expertise in the art, he takes on the challenge of improving the efficiency of his programs through redefinition and more complex methods for eliminating unnecessary steps.

He has inescapably, but subtly, reorganized his thinking processes. He now has developed a new capability to organize his thoughts concerning any particular task, to graphically display the steps necessary to the task, to make logical comparisons and distinctions between facts and hypothesis, and to then develop the most efficient procedure to accomplish the task.

This subtle transmutation seems to take place whether or not the student recognizes it. The transmutation results in the enhancement of personal organization and efficiency of thinking that cannot help but carry over into other areas of the individual's life.

This process alone has the potential of becoming a powerful tool for every individual that experiences it. The process cannot be experienced through the use of a non-programmable calculator; the same process was not experienced by the authors while using the HP-25 (a lower capacity machine with 49 step capability).

Not only was the HP-25 rapidly outgrown by the authors but keying in a program each time limited the efficiency of programming, particularly when more machine capacity was required. With the lower capacity machine, the only mental effects encountered seemed to be those associated with tailoring a program to fit the machine, if it would fit at all.

In short, the individual's mental faculties were not taxed to the degree necessary for transmutation to take place. Lower capacity machines, however, may be entirely sufficient to teach programming and/or transmutation at lower educational or experience levels [10].

Conversely, a 224 step machine has sufficient capacity for the vast majority of problems (as indicated in the appendices), while longer problems e.g., those used in linear regression, queuing, etc., tend to tax the mental comprehension of the individual while constructing programs.

As a result of programming the HP-67 and the SR-52, and experiencing the transmutation process, it is suspected that machines of greater capacity (greater than 224 steps)

28

may not result in equal or greater transmutation capability. It is feared that students would simply write programs that fit within the capacity of the machine, without regard for further efficiency. Programming would be sloppy because programming space would allow sloppiness. Hence, although a student who begins with a larger capacity machine may develop intricate programming and thinking capabilities, nothing forces the process. Thus, if the aim is to increase overall analytical ability as quickly as practicable, students ought to begin with machines of no greater capacity than the SR-52 or the HP-67. It is suspected that greater machine capacity will lead to a longer time for transmutation to take place.

It is strongly believed by the authors that the refinements and complexities developed through the described process carry over into other fields of endeavor and, especially, that the more refined and more complex the thinking process change becomes, the greater the analytical capacity of the user.

Thinking process transmutation is a most desirable effect to be created as early as practicable in the educational program of a student. The student could be expected to be much better prepared to meet the challenge of later courses. It is axiomatic that educators within the above described process could surpass all previous records in improving the ability of their students. More importantly,

29

graduates would be significantly more valuable to their employers.

F. CONCLUSIONS

The drudgery and fear of manipulating complex formulas is no longer necessary. An educator can now provide the student with the algorithm to solve a problem, have him program it and carry out several computations, and then be able to spend a greater share of instructional time teaching theory and applications. There is little question that the student will gain greater intuition and understanding of mathematical concepts by following this procedure.

The card-programmable calculator itself is clearly an enduring teaching aid. Its portability is a great advantage, enabling the student to carry exceptional computing power in his pocket.

Educators should have no fear of this device. If used properly, much more can be accomplished in the classroom, in both quantity and depth of coverage of material. Accordingly, educators can significantly advance their teaching capabilities and improve the quality of graduates thereby.

Thinking process transmutation is a phenomenon that ought to be studied in more detail. At this time, it can be said that such transmutation occurs during four to six weeks of calculator usage. The phenomenon has occurred with every person interviewed who has had four to six weeks experience in programming (card-programmable) calculators.

30

It is clear to the investigators that before providing more machine capacity (more than 224 step capability) to the student, the phenomenon should be studied further in order to assure that it occurs at the earliest time in the curriculum. In addition, more work needs to be done to determine an appropriate measure of the phenomenon. It may be that thinking process transmutation can be enhanced by "stacking" machines of increasing capacity in the educational process, i.e., 224 step, 500 step, 1000 step, etc.

When selecting a calculator for an educational program, machine reliability should be a strong consideration. Although it cannot be said with certainty, machine failure seems to have been a strong factor with the students who received the lowest grades in the pilot project. At the very least, a failed machine is of little use and could create an administrative burden to get machines repaired.

Ideally, sponsors should provide students with calculators upon matriculation for use during their educational program and after graduation. Instructors should be issued calculators with printing capability, both for the benefit of students and instructors. Additionally, an aid, such as the "Edu-Calc", should be provided for classroom instruction.

Unfortunately, far too many educators are not yet aware of the potential of this most valuable tool. The authors are convinced that, provided exposure and knowledge is given, the vast majority of people soon begin to realize

the diverse applications of the instrument and the new abilities that they command. Conversely, a failure to exploit the advantages of the card-programmable pocket calculator in the educational process could be contradictive of the tenets of the process itself.

III. FLEET USE OF CARD-PROGRAMMABLE CALCULATORS

A. GENERAL

The use of card-programmable calculators in fleet operations is certainly a feasible eventuality considering their usability and adaptability. A dramatic example of fleet usefulness is that portrayed by Commander, Patrol Wings U.S. Pacific Fleet and Commander, Patrol Wings U.S. Atlantic Fleet who have been using several HP-67's for approximately one year for airborne search detection problems [11]. An HP-65 (no longer in production), a predecessor of the HP-67, was used as a backup for the onboard computer system on the Apollo-Soyuz linkup mission [12]. Several civilian ships navigators use card-programmable calculators for speed and accuracy of calculations [13]. Appendix C is a short list of some of the places where card-programmable calculators should be used in the Navy and is by no means a complete list. Specific examples of usefulness are listed in Appendix D. In short, appendices C and D indicate that practical and feasible application in the fleet within innumerable disciplines is limited only by imagination.

The simple fact that a great deal of time can be saved while analyzing data, coupled with unparalleled accuracy at a relatively low cost, is a strong reason for implementing the use of these calculators throughout the fleet. This

33

fact alone would enable managers to make more timely decisions based upon thoroughly analyzed and accurate data.

It is obvious that decisions supported by accurate information are usually better decisions. Better (and faster!) decisions would form the very heart of improved management that could lead to improved operational readiness throughout the Navy.

B. MACHINE CAPACITY REQUIREMENTS

Based upon this study, the capacity of machines currently in production (the SR-52 and HP-67) is probably sufficient for the vast majority of fleet applications. Lieutenant Commander Harvey states [14] specifically that he believes machines of greater capacity would not provide significant gains in tactical capability. Comparisons of appendices C and D and the NPS Pilot project [supra] support that belief. It is therefore clear that 224 step capacity is adequate.

If a particular problem is beyond the capacity of currently available machines, it might be more efficient to use available computers. Alternatively, a great deal of management information, no longer available due to computer operation cost, could be regained by fleet users, e.g., daily maintenance material management (3M) summaries could be produced in abbreviated form by 3M analysts.

C. ADDITIONAL ADVANTAGES OF FLEET USE

As an additional advantage, fleet exchange of programs and programmed cards could be an excellent vehicle for the

transfer of technology among fleet users at a very low cost.
LCDR Harvey [supra] indicates that the publication of 700
copies of one tactical program costs approximately $1000,
including programmed cards and all supporting publications.

There is little doubt that fleet exchange of programs
would lead to a certain amount of standardization in
programming. However, directed standardization of all
programming would not be in the best interests of the Navy
if programmable calculators become a standard Navy item.
Specifically, strict standardization of programming would
probably stifle individual creativity and investigation.
This aspect is considered vital in order to obtain the
benefit of all fleet talent available. Put simply, the
Navy should take advantage of the natural curiosity and
investigation that will occur among users.

D. MACHINE RELIABILITY

If card programmable calculators are purchased for use
in the fleet, machine reliability should be a prime consider-
ation. COMPATWINGSPAC experience shows that HP-67's operated
on aircraft electrical systems (115 VAC, 400 Hz) have incurred
less than a five percent failure rate although operating
temperatures are eight degrees higher than in normal (115 VAC,
60 Hz) usage. Unfortunately, no data has been found for the
SR-52 under similar conditions.

The above mentioned degree of reliability, or better,
is necessary, if such a machine is to be used to make

35

tactical decisions, even if the cost of returning machines to the factory for repair is not considered.

E. CONCLUSIONS

The potential impact of the card-programmable calculator upon fleet operations must be considered as enormous. Such a machine can provide significant advantages in tactical employment, management efficiency and the transfer of technology. Although this broad range of applications has been shown, many specific uses are yet to be defined.

The authors will attempt to define further specific uses upon returning to the fleet after graduation.

Although these machines are not a panacea for solving all management problems in the fleet, the Navy should take every advantage of their computing power.

IV. PROGRAMMING AND PROGRAMMABILITY

A. GENERAL

Programming and programmability of card programmable calculators is the most important area to be studied when attempting to decide on the kind of system to use. In simplistic terms, programming is only a matter of putting an algorithm in a particular form for the machine to accept it and then compute a correct answer. On the other hand, programmability depends upon several factors.

Programmability is defined here as a combination of:

    (1)   machine language

    (2)   merging of key strokes in program steps

    (3)   type and number of program steps available

    (4)   ease of programming.

Each of these factors must be compared on its own merits. Then, the user must decide which combination will best suit his needs within his budget. The aforementioned interrelated factors are discussed further in following subsections concerning value analysis, basic and advanced programming techniques.

B. A METHOD OF VALUE ANALYSIS

Once the determination has been made as to the system (or systems) that will satisfy the particular need, the question of value analysis arises. Certainly, a part of that value analysis was determining the need in the first

place. However, here to be discussed is the primary
question of cost.

Cost(s) of a card programmable calculator should be
viewed as _system_(s) cost. Accordingly, cost(s) to be con-
sidered are more than just the purchase of a calculator.
They include a) initial machine cost, b) cost of additional
support such as program card libraries to be purchased or
published, text materials, replenishment items, etc.,
c) recurring cost, i.e., replacement and/or repair cost
(this is primarily a question of machine failure rates).

The decisionmaker must determine (or make the best
estimate) each of these costs for the various systems on
the market, and then, compare them as to the need to be
satisfied and the money he is willing to spend.

It is up to the decision maker to decide the weight to
be given to each particular item of the analysis. For
instance, initial machine cost may be determined to be more
important than reliability or vice versa. As an aid in
making these many judgements, the decision maker ought to
obtain the benefit of the knowledge of as many users as
practicable.

Necessarily, the end use of the system(s) will likely
be the major consideration in the analysis of costs and
benefits. There are no hard and fast rules concerning
any particular item. Again, it is up to the decision
maker to determine which system(s) will satisfy the needs
and the money he is willing to spend to obtain the best value.

However, the decision maker must also consider at least these two factors:

(1) Increased simplicity, speed, or convenience (which can be gained from the new calculator) not available under the current procedures.

(2) Additional capabilities which can be expected to accrue as expertise on the new calculator(s) occurs.

Most purchasing decisions appear to be based more upon the former than the latter. In the opinion of the authors, the latter is much more important. Thus decision makers should assume such benefits will be available to higher levels if more capable, more convenient calculators are purchased.

Manufacturer's suggested retail prices, as of this writing (March 1977) are:

SR-52	$249.95
HP-67	$450.00
HP-97 (Includes Printer Capability)	$750.00
NS-7100 (Projected)	$400.00
TI-59 (Projected)	$300.00
PC-100A Printer for SR-52 or TI-59:	$199.95
TI-58	$125.00

C. ALGEBRAIC VS REVERSE POLISH NOTATION

1. Description

The machines compared herein operate either in an Algebraic Operating System (AOS) (SR-52) or in the Reverse

Polish Notation (RPN) (HP-67) system.

The AOS enthusiast states, "I put the problem into the machine as it is normally written down", while the RPN enthusiast states, "I put the problem into the machine as it is normally calculated." Neither statement is entirely correct. For instance, consider the following formula:

$$Q = \sqrt{2KD/k_c}$$

For computation, the following (conceptual) sequences must normally be followed on the two machines: (The machines have no actual commas; commas are inserted below to group concepts only.)

AOS	RPN
2, XK, XD, $\div k_c$, =, $\sqrt{\ }$	2 ENTER, KX, DX,
or	$k_c \div$, $\sqrt{\ }$
(2, XK, XD, $\div k_c$), $\sqrt{\ }$	

The reader should note that the formula had to be rewritten in all cases to enable machine computations. Neither AOS nor RPN allow entry of the square-root function operator prior to completion of all other operations. This problem must be <u>worked</u> from the inside to the outside in all cases: i.e., in all cases the machine must be told to compute the

result of $(2KD/k_c)$ and then be told to find the square

root of that result. It cannot, as one concept, be told to

compute the square root of $(2KD/k_c)$. The same argument holds

for sin x, cos x, tan x, ln x, log x, x!, x^2, and 1/x

operations on both machines, as well as to the Y^x and $^X\sqrt{Y}$

two function operations. Obviously, then, the AOS user

uses a considerable amount of RPN notation.

The AOS enthusiast also argues that he doesn't have

to learn a new language. Again, there is disparity — in

addition to the examples of the previous paragraph. Returning

to the generalized formula, $Q = \sqrt{2KD/k_c}$, note that there are

neither any parentheses nor any internal equality signs on

the right-hand side of the equation as were required for

the AOS example above. General formulas do not always have

parentheses, and rarely, if ever, show implied internal

equality signs. Instead, it is assumed that the user will

intuitively group variables and mathematical operators as

if these implied symbols were, in fact, present. This being

the case, the AOS user must usually stop to remember where

the implied symbols go and insert them. On the other hand,

the RPN user must remember to always enter the operator

after the variable. Hence, it is clear that for either AOS

or for RPN, the user must, to some extent, "learn a new

language". The RPN enthusiast usually argues that he must

only enter the variable(s) and then "tell the calculator

what to do with them". Truly, the RPN user must remember

41

only one rule: The operator(s) is(are) always entered
after the variable(s).

Only two registers are required to solve a simple
problem such as W + X + Y + Z in either AOS or RPN. In
AOS, additional registers will allow varied grouping of the
variables with parentheses, but will not permit additional
forms to solve the same problem; i.e., regardless of paren-
theses, each operator is entered before the variable it
will ultimately operate on. In RPN, additional registers
will allow the problem to be structured in varied ways,
all of which separate variables and their operators to
different degrees. For example:

```
              AOS                            RPN

(Base Case) W + X + Y + Z =      W Enter X + Y + Z +
    or      W + X + (Y + Z) =     W Enter X + Enter Y Enter Z ++ *
    or      W + (X + Y + Z) =     W Enter X Enter Y + Z + +
    or      W + (X + (Y + Z)) =   W Enter X Enter Y Enter Z + + +
    or      (W + X) + (Y + Z) =   W Enter X + Enter Y Enter Z + + *
                                  * or:
                                  W Enter X + Y Enter Z + +
```

In short, an RPN machine gives flexibility that is not
available in an AOS machine. This feature becomes a powerful
benefit when writing programs in RPN. It also saves the time
of rewriting an equation [begin inside, work out, operator
after variable], an option the A0S user usually does not
have if he wants the most efficiency [3].

Using Hewlett-Packard terms [4], the automatic
stack operates as follows:

42

Data is entered:

5 ENTER 6 ENTER 7 ENTER 8

The stack registers then can be visualized as:

5.00	T	
6.00	Z	
7.00	Y	
8.	X	(DISPLAY)

Pushing the "roll-down" key once results in the following change:

8.00	T	
5.00	Z	
6.00	Y	
7.00	X	(DISPLAY)

Now, pushing the "roll-up" key twice changes the display to:

6.00	T	
7.00	Z	
8.00	Y	
5.00	X	(DISPLAY)

Adding 5 to 5 in display by pressing 5 + changes the stack to:

7.00	T	
7.00	Z	
8.00	Y	
10.00	X	(DISPLAY)

As noted above, keying in 5 pushed 6 out of the stack. When + was pressed, the two 5's were added to put 10 in display and the stack dropped. The 7 in T is duplicated in Z. The dropping and lifting of the stack enables the user to position intermediate results in long calculations without the necessity of reentering the numbers. The stack coupled with RPN creates an efficiency of the language that is not within the capability of AOS. On the other hand, AOS' programs are easier to "translate" because of the almost direct algebraic notation used therein.

2. Accuracy

Accuracy of the last several digits displayed by the SR-52 or HP-67 subsequent to executing calculations varies slightly between the machines. For example, refer to the "Internal Rate of Return" (IRR) nine place results calculated in Section IV.E(9)C.(8). The accuracy of either calculator is adequate for all but the most esoteric uses.

3. Execution Time

Execution time is generally slightly less on the SR-52 than on the HP-67 for identical functions. The SR-52 executes exponentiation to positive powers much faster than the HP-67; conversely, the SR-52 creates an error condition with the correct, but unsigned, number in the display when attempting to exponentiate to negative powers. Thus the SR-52 programmer must include three steps such as IF ERR, A; +/- or five steps such as IF ERR, 123; +/- (to ensure that the program will continue) if the program can accept negative

44

exponentiation values. Even when such steps are required,
execution time is less on the SR-52 than in the normal
execution time on the HP-67. Then the HP-67 runs slower,
but uses less program space for negative exponentiation.

D. BASIC CALCULATOR PROGRAMMING CONCEPTS

Basic programming is nothing more than routinization
of repetitively occurring mathematical equations in terms
of equation variables as stored by the user into particular
storage registers. For example, consider the calculation
of $\sqrt{9 \times 26/7}$. One routine mental (or pencil-and-paper)
approach to this calculation is to multiply 9 X 26 yielding
the product of these two terms, dividing this product by 7
yielding a quotient and, finally, extracting the square root
of the quotient. This process can be manually keyed into
SR-52 or HP-67 calculators as follows:

SR-52	HP-67
9 X 26	9 Enter 26 X
+ 7 =	7 +
2nd $\sqrt{x}$	f $\sqrt{x}$

Resulting in: 5.78174467   5.781744670

Naturally, a program written specifically to generate this
unique result would never be written unless one had some
reason to constantly be reminded that
$\sqrt{9 \times 26/7}$ = 5.78174467(0). Conversely, a program written

45

to generate $\sqrt{A \times B/C}$ for any given value of the variables
might prove most useful. Such a program is readily accom-
plished merely by storing the value of A in Storage Register
(Reg) #1, the value of B in Reg #2, the value of C in Reg
#3, and defining the calculation process in program memory.
This translation process is best begun by rethinking
$\sqrt{A \times B/C}$ in language the calculator can understand, i.e.,
as $\sqrt{(\text{Reg \#1 Contents}) \times (\text{Reg \#2 Contents}) \div (\text{Reg \#3 Contents})}$.
For calculator keying, the Recall (RCL) instruction orders
the proper storage register contents into operation. (In
SR-52, the primary storage registers are numbered 00 through
19; in HP-67 as 0 through 9 and Secondary (S) 0 through S9.)
Hence, $\sqrt{A \times B/C}$ translates in <u>concept</u> as:

<u>SR-52</u>   <u>HP-67</u>

$\sqrt{\text{RCL 01} \times \text{RCL 02} \div \text{RCL 03}}$ $\sqrt{\text{RCL 1} \times \text{RCL 2} \div \text{RCL 3}}$

The manual keystrokes to store the variable values used
above into appropriate storage registers would be:

<u>SR-52</u>   <u>HP-67</u>

9 STO 01   9 STO 1
26 STO 02   26 STO 2
7 STO 03   7 STO 3

The manual keystrokes to accomplish the previous example
would then be:

	SR-52		HP-67
	RCL 01 X RCL 02		RCL 1 ENTER RCL 2 X
	÷ RCL 03 =		RCL 3 ÷
	2nd √x̄		f √x̄

In order to convert each of the above keystroke sequences
into a program, it is only necessary to assign each sub-
routine a beginning point, or Label (LBL) and a stopping
point. Labeling requires three keystrokes in either the
AOS HP-67 or the SR-52: the stopping point requires only
one keystroke. For a Label named "A", these keystrokes
would be:

		SR-52	HP-67
LABELING:		2nd LBL A, normally abbreviated: *LBL A	f LBL A
ENDING:		HLT (Halt)	R/S (Run/Stop)

[Program space actually required is less on the HP-67:
see section IV.E(6).]

The respective Labels can be used either for storing the
variables in desired registers or for operating on the
stored variables. Hence, an entire program for the
previous example might be:

SR-52	HP-67
*LBL A STO 01 HLT	f LBL A STO 1 R/S
*LBL B STO 02 HLT	f LBL B STO 2 R/S
*LBL C STO 03 HLT	f LBL C STO 3 R/S
*LBL D	f LBL D
RCL 01 X RCL 02	RCL 1 ENTER RCL 2 X
÷ RCL 03 =	RCL 3 ÷
* √x̄	f √x̄
HLT	R/S

This program can be manually keyed into program memory exactly as listed above. The manual keystroke sequence to calculate the prior example, on either an SR-52 or HP-67 calculator, is now reduced to eight keystrokes as follows: (Commas are never keystrokes, but are used herein to clarify operations): 9 A, 26 B, 7 C, D. The calculator will now respond almost immediately with the answer, 5.78174467(0). More importantly, the calculator will not "forget" the contents of any storage register unless those contents are changed by the user. Thus, in order to calculate $\sqrt{9 \ X \ 32.5 + 7}$ (immediately following calculation of the previous example) it is only necessary to change the value of "B" from 26 to 32.5 and rerun the calculation process defined by Label D. This is accomplished by the keystrokes 32.5 B, D. The calculator will this time respond almost immediately with 6.464187055 (AOS) or 6.464187056 (RPN). Similarly, a completely new problem, such as $\sqrt{44.3 \ X \ 13.6 \div 89.66667}$ can be entered as 44.3 A, 13.6 B, 89.66667 C, D--yielding 2.59212414(0), etc.

The programs developed above are quite basic and thus are not particularly efficient. Both can be substantially refined. A more complete discussion of advanced programming follows.

E. ADVANCED PROGRAMMING TECHNIQUES FOR SR-52 OR HP-67

   1. Introduction

Basic calculator programming concepts are introduced in the previous subsection but an in-depth discussion of basic and intermediate programming techniques is not provided herein since such discussion is readily available in manufacturer's handbooks, manufacturer's promotional literature, user club publications and other sources listed in the bibliography. Instead, the discussion which follows is designed to clarify the difference between AOS and RPN advanced programming techniques, regardless of the reader's prior familiarity with either system. Of course, since the following discussion constantly compares the two systems, readers already familiar with either AOS or RPN are likely to find this discussion to be lighter reading. On the other hand, this information is also designed to help readers who are presently unfamiliar with either system to determine which system can best meet their personal or organizational needs. In this respect, the information provided is especially designed to enable managers, staff analysts and procurement officers to determine which system is optimal for the organizational requirements being encountered. As far as

can be determined by the authors, the only currently available literature which significantly facilitates such management and procurement decisions is published by the respective manufacturers. Accordingly the following discussion attempts to create an unbiased comparison which can be used for managerial/procurement decisions.

Data obtained from the various manufacturers by the authors — sometimes on a non-disclosure basis concerning specifics — indicates, in general, that the comparisons provided herein are likely to stand inviolate for at least ten years. The introduction of new machines within the next decade, as currently envisioned by the manufacturers, will do little to change the analysis which follows. Thus, readers in management positions are especially encouraged to study the following analysis in detail. On the other hand, persons unfamiliar with either AOS or RPN should be able to determine which system best fits their individual idiosyncracies and personal preferences after studying the material which follows. Hence, this material should greatly enhance personal purchasing decisions as well as organizational purchasing decisions. This information should be especially useful to students, educators, or others considering the purchase of a personal card-programmable calculator.

3. Color Coding

In order to minimize the number of keys required on the machines, all manufacturers use most keys for at least

50

two purposes — and sometimes three or four. So, most keys

have a primary function and one or more secondary function(s).

Typically, the primary function is centered in one color

upon the flat upper surface of the key itself, whereas

each secondary function is listed in a different color above

the key, below the key, or upon an angled face of the key.

Therefore, a key may have associated functions listed in

one, two, three, or four different colors and the primary

colors of the keys themselves vary according to purpose.

In order to enable the calculator to determine which

listed key function is desired when a key is pressed, color-

coded, dedicated, "Second-Function" keys are pressed _first_

to designate secondary key functions similarly colored.

Typically, the primary function listed upon a key is exe-

cuted simply by pressing that key; the secondary function

is executed by _first_ pressing a second-function key (which

matches color with the color of the desired secondary func-

tion) and _then_ pressing the key; the alternate secondary

function, if any, is executed by _first_ pressing an alternate

(third) second-function key (which matches color with the

color of the third function) and _then_ pressing the key, etc.

AOS systems such as the SR-52 or TI-59 have one second

function key (Yellow: 2nd) whereas the HP-67 has three

(Yellow: f), (Blue: g), (Black: h). The AOS notation "2nd"

corresponds directly with the thought process involved, i.e.,

use the matching-color 2nd function listed — whereas the HP-67

51

notation corresponds indirectly by using standard symbols for mathematical functions (f, g, h) such that the user can easily modify the thought process to: "Use the function of the key (f of key, g of key, h of key) by first pressing the color matched and appropriately labelled second function key (f, g, h)." The reader should note that the color coding greatly simplifies actual usage and complicates only the reading of documented programs by novice users, e.g., a novice user would tend to look at the keyboard, see what color a desired function is and then push appropriate keys to execute the function without difficulty. The novice is not required to memorize that $\sqrt{x}$ is always preceded by 2nd or f since the color coding, in actual usage, readily prompts appropriate action. On the other hand, the documentation of programs normally includes all keys which must be pressed, such that $\sqrt{x}$ becomes 2nd $\sqrt{x}$ or * $\sqrt{x}$ or $f\sqrt{x}$, which may take a little getting used to. However, new or different does not necessarily equate to difficult. Thus the often-heard cry of novice users that documented programs are "difficult to read" appears to be unfounded.

3. Programming Language

Fear of having to learn a "new language" in order to operate a card-programmable calculator is, in general, unfounded. True, the user may be required to supply implied parentheses or equality signs (AOS) or be required to always (RPN) or sometimes (AOS) enter the operator after the mathematical variables, and be required to become familiar with

52

a long list of abbreviations — but none of these requirements
can be considered to be a new language. In fact, the
"language" required by either AOS or RPN is best defined as
merely a simple abbreviation system for normal mathematical
language, similar to abbreviations that most high-school
graduates are already familiar with. Thus, learning the
language of the calculator is simply a matter of adapting
prior personal abbreviation habits to the abbreviations
listed on the calculator keys. Once the novice has learned
to recognize the listed abbreviations, he is ready to oper-
ate or to program. Further, since many abbreviations are
common to both AOS and RPN, individuals can quickly master
a second system. Additionally, many of the abbreviations
are phonetic or otherwise quite straightforward. Hence, it
is feasible to learn the great majority of these abbrevia-
tions, adequate for almost all calculators on the market,
in less than half a day.

Table 1 lists most abbreviations commonly found,
as well as a brief definition of each abbreviation. (For
full description of each function, the reader is referred
to handbooks published by the various manufacturers.)

The first column in Table 1 lists abbreviations which
are used by the authors to amplify documented programs in
the remainder of this work, especially in the remarks sec-
tions of the programs themselves. This notation largely
duplicates keystroke abbreviations found on the various
calculators or amplifies such abbreviations. The reader

can, therefore, more easily translate any of the programs
which follow — on either machine — by studying this consis-
tent notational system in lieu of studying the particular
keystrokes actually required. In other words, instead of
learning two systems, the reader can concentrate on the one
"translation system" which is provided throughout. Viewed in
another way, this notational system is also a first step
toward a universal calculator language; the average reader
will find it useful for annotating his own future programs.

The reader is warned not to equate the charted list
of key abbreviations to functional capabilities (see Appendix
B for functional capabilities). Many functions require
sequenced keystrokes and thus are not in Table 1; e.g.,
on the SR-52, the arc sine is executed by pressing INV SIN
and e^x is executed by pressing INV Lnx, but neither ARC SIN
nor e^x appear, on the SR-52, as key abbreviations.

The second column of Table 1 defines the abbrevia-
tions as used by the authors. Such use usually, but not
always, corresponds to the designed primary use of the same
abbreviation by the calculators. The abbreviations duplicate
notation actually found on the machines to the maximum
extent feasible, adding or changing only where necessary
to achieve the desired consistent set of abbreviations which
can then be used to amplify either HP-67 or SR-52 programs
as discussed previously.

The final columns show the relationship between
the definitions and the labeling used on the machines.

54

Direct correlation of both the abbreviation and its meaning
to a key found on one of the calculators is indicated either
by "P" (Primary function of one key) or by "*" (Secondary
function of one key) or by S (Manual Switch) as listed under
each calculator. Additionally, alternate abbreviations used
on the calculators are listed opposite the appropriate defini-
tion. Finally, the entry "(OU)" in these columns means that
the same abbreviation is used for some other use on the
calculator in each case the other use can be found elsewhere
in the same column. For example, "(OU)" is located in the
right column opposite the left-column abbreviation "DEG",
which is defined as "Degrees" for use in this study. Several
entries below the "(OU)" entry in the same (right) column,
"DEG" appears, opposite "DEG MODE" in the left column,
which is the abbreviation used for "Degrees Trigonometric
Mode" throughout this study. Other entries on this same
latter line of Table 1 indicate both what the abbreviation
"DEG" designates as used by the particular calculators and
what "DEG MODE" designates as used by the authors, i.e.,
that the abbreviation "DEG" appears as a second-function
key designation (*) on the HP-67 designating "Degrees
Trigonometric Mode" and as a switch designation (S) on
the SR-52 designating "Degrees Trigonometric Mode."

Hence, Table 1 defines how the authors use particu-
lar abbreviations throughout the remainder of this work;
whether such usage corresponds directly to usage on the

55

particular machines, and (if the correspondence is not direct) how the same abbreviations are used on the calculators or what different abbreviations are used on the calculators.

Novice calculator users are advised to concentrate only upon the two left columns of Table 1. More advanced readers can use Table 1 to determine if a specific abbreviation used by this study also appears as an abbreviation on the SR-52 or HP-67. However, all readers are again warned that Table 1 lists abbreviations, not functional capabilities. After achieving basic familiarity, readers can restudy Table 1 and Appendix B, together with the discussion which follows, in order to better determine which calculator's system appears to best fit personal or organizational purchase parameters.

4. <u>Key Code Systems</u>

Both the HP-67 and SR-52 use a key code system which closely corresponds (for primary functions) to the location of each key in an imaginary superimposed, second quadrant, xy matrix. The y-value is read first, neglecting the minus sign; i.e., row, column. Thus the primary function of the key at topmost, leftmost (the A key) is coded 11 the primary function of the key at fourth row <u>down</u>, third column to the <u>right</u> is 43, etc. As one exception to this pattern, on both machines, numeric keys are directly coded 00, 01, ..., 09. This exception causes no confusion both because it is direct and because there is no "zero row" of keys.

TABLE I. KEY ABBREVIATIONS

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
A	User-defined Label	P		P	
A'	User-defined Label	*			
a	User-defined Label			*	
B	User-defined Label	P		P	
B'	User-defined Label	*			
b	User-defined Label			*	
BST	Backstep (Program Mode)	*		*	
C	User-defined Label	P		P	
C'	User-defined Label	*		*	
c	User-defined Label	*		*	
CHS	Change Sign of Display	P	+/-	P	
CLR	Clear	P	(OU)	P	
CLR	Clear Entry or Arrest Flashing Error	P:	CE	P	CLX
CLR DSP	Clear Display	P	CLR	P	CLX
CLR FLAG	Clear Flag			*	ċF
CLR PRGM	Clear Program			*	CLPRGM
CLR REG	Clear (Primary) Registers			*	CLREG
COORD	Coordinates				
COS	Cosine	P		*	
$\cos^{-1}$	Arc Cosine			*	
D	User-defined Label	P		P	
D'	User-defined Label	*			
d	User-defined Label			*	

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
DEC	Decimal				
DEC DEG	Convert DEC DEG(HR) DSP - Format to DEG(HR), MIN, SEC, DSP - Format			*	H..MS
DECR	Decrement by 1				
DEL	Delete Step (PRGM MODE)	*		*	
DEG	Degrees				(OU)
DEG DEC	Convert DEG(HR), MIN, SEC DSP-Format to DEC DEG(HR) DSP-Format	*	D.MS	*	H
DEG MODE	Degrees Trigonometric Mode	S	D	*	DEG
DEG RAD	Convert Degrees to Radians	S	R	*	R
DO	Compute or Execute				
DSP	Display or X-Register				(OU)
DSZ	Decrement DSP (or I-Reg) by 1, Skip Step if Result is Zero	*			
DSZ(i)	DSZ (Indirectly in REG Addressed by I-REG)			*	
E	User-defined Label	P		P	
E'	User-defined Label	*			
e	User-defined Label			*	
EEX	Enter Exponent	P	EE	P	
ENG DSP	Engineering Display Format				ENG
ENTER	Lift STACK; Enter DSP into STACK Y-REG			P	
ERR	Error			*	
e^x	Xth Power of e			*	
EXCH	Exchange	*	EXC	*	

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
EXP	Exponent				
f	2nd-Function Call-Key	P	2nd	P	
FORMAT	Set Display Format				
FUNC	Function				
FIX DSP	Fixed Decimal Display Format	*	FIX	P	DSP
FLG	Flag				
FLOAT PT	Floating Decimal Display Format				
FRAC	Fractional Portion			*	
g	2nd-Function Call-Key	P	2nd	P	
GRAD	GRADS TRIG MODE			*	GRD
GTO	Go To	P		P	
h	2nd-Function Call-Key	P	2nd	P	
(i)	I-Register			P	(OU)
IF ERR	If Error is Displayed (Flashing)	*			
IF FLAG	If Flag is Set	*	if flg	*	F?
IF x=y	If Display = Y-register			*	x=y
IF x≤y	If Display ≤ Y-register			*	x≤y
IF x≠y	If Display ≠ Y-register			*	x≠y
IF x>y	If Display > Y-register			*	x>y
IF x=0	If Display = zero	*	if zro	*	x=0
IF x≥0	If Display ≥ zero	*	if pos		
IF x≠0	If Display ≠ zero			*	x≠0
IF x<0	If Display < zero			*	x<0
IF x>0	If Display > zero			*	x>0

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
INCR	Increment by 1				
IND	Indirect	*		P	(i)
INS	Insert Step (Program Mode)	P			
INT	Integer Portion			*	
INV	Inverse Function of ...	P			
ISZ	Increment DSP (or I-Reg) by 1; Skip Step if result is zero	*		*	
ISZ(i)	ISZ (In REG Addressed by I-REG)			*	
LBL	Label	*		*	
LBLf	Label			*	
LASTx	Entry Prior to Last Operation			*	LSTx
LIST	List Program	*			
LN	Logarithm, Base e	P	ln x	*	
LOG	Logarithm, Base 10	*		*	
LRN	Learn (Shift to Program Mode)	P		S	W/PRGM
MEM	Memory				
MERGE	Merge Steps (Program Mode			*	
ON/OFF	Switch Machine ON/OFF	S		S	
OPS	Operations				
OW	Otherwise				
PAPER	Advance Paper	*	pap		
PAUSE	Pause from Operation (for data entry)			*	
P→R	Convert Polar Coordinates to Rectangular	P/R		R	

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
PRI	Primary				
PRGM	Program				
PRGM MODE	Program Mode	P	LEARN	S	W/PRGM
PROD	Product (Multiplication in Register)	*			
P⇄S	Primary/Secondary Register Exchange			*	
R↓	Roll Down Stack Registers			*	
R↑	Roll Up Stack Registers			*	
RAD	Radians				(OU)
RAD→DEG	Convert Radians to Degrees			*	D
RAD MODE	Radians Trigonometric Mode	S	R	*	RAD
RCL	Recall	P		P	
RCL(i)	RCL (Indirectly from Register Addressed by I-REG)				
READ	Read Card	*			
RECT	Rectangular Coordinates				
REG/ REG	Register/Store in REG				(OU)
R→P	Convert Rectangular Coordinates to Polar	*			
REVU REG	Review Primary Register			*	REG
REVU STACK	Review Stack Register			*	STK
RND	Round Off			*	
RSET	Reset Counter to 000	*			
RTN	Return Control to Calling Routine (or Keyboard)	*		*	
RUN	Run Routine (Program)	P		P	R/S

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
s	Sample Standard Deviation			*	
SBR	Subroutine	P			
SCI DSP	Scientific Notation Display Format			*	
S_n	Secondary Register n (n=1,...,9)				
SIN	Sine	P		*	
SIN^{-1}	Arc Sine			*	
SPACE	No Operation (Skip Step or Space)			*	
SST	Single-Step (Program Mode)	P		P	
STACK	X,Y,Z,T Register Group				
SET FLAG	Set Flag (Turn On)	*	ST FLG	*	SF
STO	Store	P		P	
STO(i)	Store (Indirectly, in Register Addressed by I-REG)				
STOP	Stop (Halt)	P	HLT	P	R/S
SUM	Sum to (Add to Register)	P			
TAN	Tangent	P		*	
TAN^{-1}	Arc Tangent			*	
THRU	Through				
W/	With				
WRITE	Write Data Onto Card			*	W/DATA
X	Times, Multiplication Operator	P		P	
x	X-Register or Display				
$\bar{x}$	Arithmetic Mean			*	
X!	Factorial	*		*	N!

TABLE I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52		HP-67	
-x-	Flash Display (x-register)			*	
|x|	Absolute Value of x			*	ABS
$\sqrt{x}$	Square root of x	*		*	
x y	Exchange x and y register			*	
$x\sqrt{y}$	xth root of y	P			
x^2	Square of x-register (Display) value	P			
y	y-register	*			
y DEG +	Add y-register DEG,MIN,SEC to Display			*	
y^x	xth power of y	*		*	
z	z-register				
*	2nd-Function Key	P	2nd	P	f,g,h
+	Addition Operator	P		P	
.	Decimal Point	P		P	
÷	Division Operator	P		P	
=	Equality Operator	P			
=>	Implies				
(	Parenthesis, left	P			
)	Parenthesis, right	P			
%	Percentage			*	
%CH	Percent Change			*	
π	Pi (3.141492654)	*		*	
#	Number				
-	Subtraction Operator	P		P	

Table I. (Continued)

Abbreviation	Meaning as Used by Authors	SR-52	HP-67
Σ+	<u>SUM</u>: x to S4, x^2 to S5, y to S6, y^2 to S7, xy to S8, +1 to S9		P
Σ-	<u>SUM</u>: -x to S4, $-x^2$ to S5, -y to S6, $-y^2$ to S7, -xy to S8, -1 to S9		*
0	Zero	P	P
1	One	P	P
1/x	Reciprocal of Display	*	*
2	Two	P	P
3	Three	P	P
4	Four	P	P
5	Five	P	P
6	Six	P	P
7	Seven	P	P
8	Eight	P	P
9	Nine	P	P
10^x	x^{th} power of 10		*

Notes: P = Primary function of one key found on the calculator

* = Secondary function of one key found on the calculator

S = Manual Switch; not a key

The machines differ only concerning the coding system for secondary functions. Although the HP-67 can handle one, two, or three two-digit numbers in its coding displays when in programming mode, the SR-52 can handle only one two-digit number in its coding display. Thus, secondary functions are coded on the HP-67 exactly like primary functions, the only difference being that two or three two digit numbers are displayed for secondary function entries which require two or three keystrokes, respectively. Thus f LOG on the HP-67 is displayed as 31 53 and f LBL A is displayed as 31 25 11, etc. Conversely, on the SR-52, the coding system itself is modified to handle secondary functions. Since the calculator is only 5 keys wide, the column code for all primary functions is defined by the digits 1 through 5, leaving 6 through 9 and 0 available. Further, all secondary functions on the SR-52 are printed above the key. Hence, by mentally folding the remaining available digits over the first five digits in each row, each secondary function is assigned a number of its own. As an example, the coding for the third and fourth rows of the SR-52 is as follows:

Third Row:	Secondary Functions:	36	37	38	39	30
	Primary Functions:	31	32	33	34	35
Fourth Row:	Secondary Functions:	46	47	48	49	40
	Primary Functions:	41	42	43	44	45

Although the need for ten codes per row puts the last secondary code out of sequence in each row, the system does have logic and is easily learned: higher locations of abbreviations on the keyboard itself correspond to higher-numbered keycodes (with exception of the right column secondary functions). The above method solves both the coding for secondary functions and the two-digit display limitation. Since the secondary-function code cannot be created without pressing the "2nd" key, there is no need to ever display the key code for the "2nd" key — as is required to distinguish between the three available second keys on the HP-67. Hence, "2nd LOG" becomes 28, period, on the SR-52: "2nd LBL A" becomes 46 (2nd LBL) as one step and 11 (A) as another step; etc.

5. Key Coding Formats

Key coding displays, on both machines, naturally include the program step number. Thus, if the examples above begin at step #136, the complete formats would be:

SR-52		HP-67	
Keystrokes	Display	Keystrokes	Display
2nd LOG	136 28	f LOG	136 31 53
2nd LBL	137 46	f LBL A	137 31 25 11
A	138 11		

Such displays are extremely useful for verifying or editing programs. The user can always determine what key instruction is stored as a given step number by merely counting (down

and over) the rows and columns indicated by the respective keycode system.

6. HP-67 and SR-52 Program Storage Capacities

It is often assumed that since the HP-67 can store up to three keycodes per step of program memory whereas the SR-52 stores only one, the effective program capacity of the HP-67 must be three times as large. This is simply untrue, as evidenced above, because the third HP-67 keycode is never used for anything other than to designate one of the second-function keys, a designation that is never separately required at all on the SR-52. The HP-67 program capacity is slightly larger than the SR-52 capacity, but the reason is the way keystrokes are merged per step on the HP-67, not because three keycodes are displayed per step. This merging applies primarily to all addressing and labeling. Thus, returning to step 137 in the above example, the SR-52 code "46" relays the same amount of information as do the HP-67 codes "31 25". But the HP-67 program capacity is increased because the "A" requires an additional step for its code on the SR-52, whereas the "A" is merged as "11" in the codes for step 137 on the HP-67. Similarly, recalling or storing data in a given register requires three steps on the SR-52 but only one step on the HP-67. Hence, the maximum ratio of obtainable program capacities, ever encountered by the authors is approximately 2.4:1, with a more common long-term ratio of between 1.3:1 and 1.6:1, in favor of the HP-67,

for advanced programming. Conversely, novice programmers commonly encounter a ratio slightly higher than 2:1. As an example, Figures 2 and 3 (Single or Bivariate Statistical Analysis) are written with identical algorithms and program sequencing on each machine, disregarding special functions available on each machine which could be used to significantly shorten the programs (HP-67 in particular). In other words, these programs are intentionally written inefficiently in order to achieve a typical comparison of the number of program steps actually required to place identical programs in either machine. The result is 2.196:1, for these particular programs. The ratio would favor the HP-67 even more if special functions, such as the Σ+key available only on the HP-67, are included; conversely, program optimization techniques as addressed later in this study will do more to reduce SR-52 steps (for the common functions) than to reduce HP-67 steps, because the HP-67 steps are more compact in the first place. In summary, many programs do not require extensive use of special functions found on only one of the machines. In such cases, the program capacity ratio appears to be between 1.3:1 and 1.6 to one in favor of the HP-67, but certainly is not 3:1 except in unique cases.

7. Magnetic Card Formats

     Standard SR-52 and HP-67 magnetic cards are reproduced (actual size) below. The lower reproduction is a HP-67 card placed upon the backside of a SR-52 to better show relative sizes.

Figure 1

Single or Bivariate Statistical Analysis (SR-52)

(Sheet 1 of 7)

USER KEYED LABELS									
A'	N'	B'		C'	Calculate r	D'		D'	n
A	X	B	Y	C	Calculate Regression	D	X*/Y	E	Y*/X

FLAGS: OFF/ON | OTHER LABELS | NOTES

	OFF	ON
0		X
1		X
2		X
3		X
4		X

NOTES

See Sheets 6 and 7.

REGISTERS

0	
1	LAST X
2	Σx_i
3	Σx_i^2
4	n
5	LAST Y
6	Σy_i
7	Σy_i^2
8	$\Sigma x_i y_i$
9	$ns^2 y$
0	$\bar{x}$
1	$\bar{y}$
2	r
3	$ns xy$
4	$S \dot{x}$
5	Sy
6	$\hat{b}$
7	ns^2
8	$\hat{a}$
9	

SET STATUS

DISPLAY   TRIG

☐ SCI	☒ DEG
☐ ENG	☐ RAD
☐ FIX	☐ GRAD
☒ FLOATING PT	

STEP	KEY CODE	KEY ENTRY
000	46	*LBL
1	11	A
2	42	STO
3	00	0
4	01	1
5	44	SUM
6	00	0
7	02	2
8	40	*x²
9	44	SUM
010	00	0
1	03	3
2	01	1
3	44	SUM
4	00	0
5	04	4
6	43	RCL
7	00	0
8	02	2
9	55	÷
020	10	*E'
1	95	=
2	42	STO
3	01	1
4	00	0
5	40	*x²
6	65	x
7	10	*E'
8	95	=
9	94	+/-

COMPUTE X VALUES

69

STEP	KEY CODE	KEY ENTRY	NOTES
030	85	+	
1	43	RCL	
2	00	0	
3	03	3	
4	95	=	
5	42	STO	
6	01	1	
7	07	7	
8	55	÷	
9	10	*E'	
040	95	=	
1	30	*$\sqrt{x}$	
2	42	STO	
3	01	1	
4	04	4	
5	10	*E'	
6	56	*RTN	
7	46	*LBL	
8	10	*E'	n counter
9	43	RCL	
050	00	0	
1	04	4	
2	56	*RTN	
3	46	*LBL	
4	12	B	compute Y values
5	42	STO	
6	00	0	
7	05	5	
8	44	SUM	
9	00	0	
060	06	6	
1	40	*x^2	
2	44	SUM	
3	00	0	
4	07	7	
5	53	(	
6	43	RCL	
7	00	0	
8	06	6	
9	55	÷	
070	10	*E'	
1	95	=	
2	42	STO	
3	01	1	
4	01	1	
5	40	*x^2	
6	65	x	
7	10	*E'	
8	95	=	
9	94	+/-	

STEP	KEY CODE	KEY ENTRY	NOTES
080	85	+	
1	43	RCL	
2	00	0	
3	07	7	
4	95	=	
5	42	STO	
6	00	0	
7	09	9	
8	55	÷	
9	10	*E'	
090	95	=	
1	30	*√x̄	
2	42	STO	
3	01	1	
4	05	5	
5	43	RCL	
6	00	0	
7	01	1	
8	65	X	
9	43	RCL	
100	00	0	
1	05	5	
2	95	=	
3	44	sum	
4	00	0	
5	08	8	
6	56	*RTN	
7	46	*LBL	
8	13	C	compute Regression
9	43	RCL	
110	00	0	
1	08	8	
2	75	-	
3	53	(	
4	53	(	
5	43	RCL	
6	01	1	
7	01	1	
8	65	X	
9	43	RCL	
120	01	1	
1	00	0	
2	54	)	
3	65	X	
4	10	*E'	
5	54	)	
6	95	=	
7	42	STO	
8	01	1	
9	03	3	

STEP	KEY CODE	KEY ENTRY	NOTES
130	55	÷	
1	43	RCL	
2	01	1	
3	07	7	
4	95	=	
5	42	STO	
6	01	1	
7	06	6	
8	65	X	
9	43	RCL	
140	01	1	
1	00	0	
2	95	=	
3	94	+/-	
4	85	+	
5	43	RCL	
6	01	1	
7	01	1	
8	95	=	
9	42	STO	
150	01	1	
1	08	8	
2	56	*RTN	
3	46	*LBL	
4	16	*A'	N'
5	02	2	
6	65	X	
7	43	RCL	
8	01	1	
9	04	4	
160	95	=	
1	55	÷	
2	53	(	
3	93	.	
4	00	0	
5	02	2	
6	05	5	
7	65	X	
8	43	RCL	
9	01	1	
170	00	0	
1	54	)	
2	95	=	
3	81	HLT	
4	46	*LBL	
5	15	E	Y*/X
6	65	X	
7	43	RCL	
8	01	1	
9	06	6	

STEP	KEY CODE	KEY ENTRY	NOTES
180	95	=	
1	85	+	
2	43	RCL	
3	01	1	
4	08	8	
5	95	=	
6	81	HLT	
7	46	*LBL	
8	14	D	X*/Y
9	75	−	
190	43	RCL	
1	01	1	
2	08	8	
3	95	=	
4	55	÷	
5	43	RCL	
6	01	1	
7	06	6	
8	95	=	
9	81	HLT	
200	46	*LBL	
1	18	*C	Compute r
2	43	RCL	
3	01	1	
4	03	3	
5	55	÷	
6	53	(	
7	10	*E	
8	65	X	
9	53	(	
210	43	RCL	
1	01	1	
2	04	4	
3	65	X	
4	43	RCL	
5	01	1	
6	05	5	
7	54	)	
8	54	)	
9	95	=	
220	42	STO	
1	01	1	
2	02	2	
3	81	HLT	END OF PROGRAMMABLE MEMORY
4			
5			
6			
7			
8			
9			

FIGURE 1

Single or Bivariate Statistical Analysis (SR-52)

Program Description, Equations, Variables, etc.

This program provides $\bar{x}$, s_x, $\bar{y}$, s_y, b, a, r, x* given y, y* given x
for bivariate samples. In addition, N' (optimum sample size with
95% confidence) may be determined using data from x_i calculation.

a) N' is computed with the formula

$$(\frac{\sigma^2 x_i}{.025 \, \bar{x}_i})^2$$

described in Chapter 12 of the Production Handbook; Carson, Bolz
and Young; Ronald Press, N.Y.; 1972.

b) The balance of the program is based upon Chapters 9 and 10,
Introductory Statistics; Zehna; Prindle, Weber & Schmidt, Inc,
Boston; 1974.

c) Single variable samples can be analyzed for $\bar{x}$, s_x and N'

d) Confidence level for N' may be changed by inserting a different $\alpha/2$
in steps 164-166.

Note: *E' is used for a subroutine in the program to recall n.
Pressing *E' will place n in display.

74

Step	User Instructions	Input Data/Units	Press	Output Data/Units
1.	Record program and initialize:			
2.	a) for single variable sample	x_i		x data
	(n will be in display after each x_i)			
	b) for bivariate sample	x_i	A	x data
	(last $(x_i)(y_i)$ will be in display)	y_i	B	y data
	c) after x_i and y_i are entered		C	
	linear regression is calculated			
	d) estimated correlation coefficient		*C'	r
	can now be calculated			
	e) N' is calculated from x_i data (in A)		*A'	N'
	f) X* may be calculated by placing a Y in display and pressing D	Y	D	X*
	g) Y* may be calculated by placing an X in display and pressing E	X	E	Y*
	h) balance of data may be recalled using RCL and appropriate register number.			

Figure 2

Single or Bivariate Statistical Analysis (HP-67)

USER KEYED LABELS				
a / A Xi	b / B Yi	c / C Calculate	d / D X*/Y	e / E Y*/X

FLAGS: OFF/ON

	OFF	ON
0		X
1		X
2		X
3		X

OTHER LABELS

NOTES

See Sheets 6 and 7

REGISTERS

0	Σx_i
1	$\bar{x}$
2	S_x
3	Σx^2_i
4	$\bar{y}$
5	S_y
6	Σy_i
7	Σy_i^2
8	$\Sigma x_i y_i$
9	n
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	nS_x^2
B	nS_y^2
C	nS_{xy}
D	Last x/$\hat{b}$
E	Last y/$\hat{a}$

SET STATUS

DISPLAY

SCI	
ENG	
FIX	
FLOATING PT	

TRIG

DEG	
RAD	
GRAD	

STEP	KEY CODE	KEY ENTRY	
0			
1	31 25 11	CLBL A	CALCULATE X values
2	33 14	STO D	
3	33 61 00	STO +0	
4	32 54	gx^2	
5	33 61 03	STO +3	
6	01	1	
7	33 61 09	STO +9	
8	34 00	RCL 0	
9	34 09	RCL 9	
010	81	÷	
1	33 01	STO 1	
2	32 54	gx^2	
3	34	09	
4	71	X	
5	42	CHS	
6	34 03	RCL 3	
7	61	+	
8	33 11	STO A	
9	34 09	RCL 9	
020	81	÷	
1	31 54	f√X	
2	33 02	STO 2	
3	34 09	RCL 9	
4	84	R/S	
5	31 25 12	fLBL B	CALCULATE Y values
6	33 15	STO E	
7	33 61 06	STO +6	
8	32 54	gx^2	
9	33 61 07	STO +7	

STEP	KEY CODE	KEY ENTRY	NOTES
030	34 06	RCL 6	
1	34 09	RCL 9	
2	81	÷	
3	33 04	STO 4	
4	32 54	gx^2	
5	34 09	RCL 9	
6	71	X	
7	42	CHS	
8	34 07	RCL 7	
9	61	+	
040	33 12	STO B	
1	34 09	RCL 9	
2	81	÷	
3	31 54	f√X	
4	33 05	STO 5	
5	34 14	RCL D	
6	34 15	RCL E	
7	71	X	
8	33 61 08	STO +8	
9	84	R/S	
050	31 25 13	fLBL C	CALCULATE Regression
1	34 08	RCL 8	
2	34 09	RCL 9	
3	34 01	RCL 1	
4	34 04	RCL 4	
5	71	X	
6	71	X	
7	51	-	
8	33 13	STO C	
9	34 11	RCL A	
060	81	÷	
1	33 14	STO D	
2	34 01	RCL 1	
3	71	X	
4	42	CHS	
5	34 04	$CL 4	
6	61	+	
7	33 15	STO E	
8	84	R/S	
9	32 25 11	gLBL a	CALCULATE N'
070	34 02	RCL 2	
1	02	2	
2	71	X	
3	83	.	
4	00	0	
5	02	2	
6	05	5	
7	34 01	RCL 1	
8	71	X	
9	81	÷	

77

STEP	KEY CODE	KEY ENTRY	NOTES
080	32 54	gX²	
1	84	R/S	
2	32 25 13	gLBL c	CALCULATE r
3	34 13	RCL C	
4	34 02	RCL 2	
5	34 05	RCL 5	
6	34 09	RCL 9	
7	71	X	
8	71	X	
9	81	÷	
090	84	R/S	
1	31 25 14	fLBL D	CALCULATE X*/Y
2	34 15	RCL E	
3	51	–	
4	34 14	RCL D	
5	81	÷	
6	84	R/S	
7	31 25 15	CLBL E	CALCULATE Y*/X
8	34 14	RCL D	
9	71	X	
100	34 15	RCL E	
1	61	+	
2	84	R/S	
3	32 25 15	fLBL e	RECALL n
4	34 09	RCL 9	
105	84	R/S	
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

END OF PROGRAM.
(End of Programmable Memory at Step #224)

FIGURE 2
Single or Bivariate Statistical Analysis (HP-67)

Program Description, Equations, Variables, etc.

1. This program provides $\overline{X}$, Sx, $\overline{Y}$, Sy, $\hat{b}$, $\hat{a}$, r, X* given Y, Y(given X

    for bivariate samples. In addition, N' (optimum sample size with

    95% confidence) may be determined using data from X_i calculation.

    a) N' is computed with the formula

    $$(\frac{2\sigma_{x_i}'}{.025\overline{x}_i})^2$$

    described in Chapter 12 of Production Handbook; Carson, Bolt

    and Young; Ronald Press, N.Y., 1972.

    b) The balance of the program is based upon Chapters 9 and 10,

    Introductory Statistics; Zehna; Prindle, Weber & Schmidt,

    Inc., Boston; 1974.

    c) Single variable samples can be analyzed for $\overline{X}$, Sx and N'.

    d) Confidence level may be changed by inserting a different $\alpha/2$

    in Steps 73-76.

Step	User Instructions	Input Data/Units	Press	Output Data/Units
1.	Record program and initialize			
	a) for single variable sample (n will be in display after each X_i)	X_i	A	X data
	b) for bivariate sample (Last $(x_i y_i)$ will be in display)	X_i	A	X data
		Y_i	B	Y data
	c) After X_i and Y_i are entered linear regression is calculated		C	
	d) Estimated correlation coefficient can now be calculated		c	r
	e) N' is now calculated from X_i data (in A)		a	N'
	f) X* may be calculated by placing a Y in display and pressing D		Y	X*/Y
	g) Y* may be calculated by placing an X in display and pressing E		X	Y*/X
	h) balance of data may be recalled using RCL and appropriate register number.			

SR-52
STANDARD
CARD
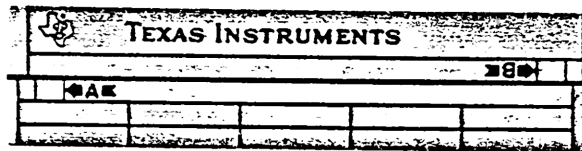
HP-67
STANDARD
CARD

HP-67 CARD
ON
SR-52 CARD

Both the SR-52 and HP-67 have 5 user-definable keys as the top row of keys, labeled A through E. The second functions of each of these are also user-definable, labeled A' through E' on the SR-52 or labeled a through e on the HP-67. The functions defined for each label are written on the card as desired by the programmer. The following notation is used throughout this work to represent the 10 user keyed labels, as those labels would actually be completed upon the card itself:

USER KEYED LABELS				
— A	— B	— C	— D	— E

The labels for the top row, of course, are completed to correspond to the keys on the machine under discussion in each case.

8. Recording or Reading Magnetic Cards

After keying a program into memory, the program can be permanently stored on a magnetic card by passing the card through the card reader. Initializing the card reader to record requires the proper setting of a switch on the HP-67 or several keystrokes on the SR-52. Data from storage registers can also be permanently stored on cards. Initializing the card reader for data requires the setting of a switch and two keystrokes on the HP-67; a special prior program card must be used together with blank data card on the SR-52, using two keystrokes with each card.

Stored programs or data remains on the cards until the information in intentionally altered by the user; i.e., the cards are reusable or can be permanently filed for dedicated purposes. Various systems are used on the different calculators to prevent accidental altering of cards, or to abort protections used to prevent accidental destruction of card information in order to reuse "permanent" cards. All of these systems work well, hence, these differences will not be addressed herein.

Reading previously programmed magnetic cards requires only the proper setting of a switch on the HP-67; two or three keystrokes per card side are required on the SR-52. Additional machine functions which are automatically executed when a card is read differ between calculators. This subject is addressed in Section IV.E.10.h.

9. <u>Advanced Programmability Comparisons, SR-52 and HP-67</u>

The authors believe that this work is relatively unbiased toward these two machines. Both have unique capacities and constraints which can apply to a given situation, but full discussion of each would unnecessarily triple the length of this section. Thus, programming concepts in this section are presented on whatever machine appears to be most pertinent for that particular concept, with no attempt to "yes but" the alternate machines alternate capabilities for some other concept. Further, since more readers are expected to already be familiar with algebra than with reverse polish notation — and for uniformity — SR-52 programs are normally discussed first. This is not intended to degrade the HP-67 or HP-97 in any manner.

a. General — Internal Rate of Return (IRR) As A Model

In order to provide a problem simple enough to be readily understood by the majority of readers, yet intricate enough to necessitate complex programming techniques, an Internal Rate of Return program is developed herein, upon the HP-67 and SR-52, as a vehicle for clarifying differences between these machines and programming systems. These programs are also compared with literature (programs) available from the manufacturers in order to yield additional insights concerning both available programs and programming potentials and limitations.

b. Internal Rate of Return (IRR) Concepts

IRR programs incorporate concepts involving the
time value of money; specifically, the net present value
(NPV) of a series of n future (periodic) cas flows (CF's)
CF_1, CF_2, ..., CF_n, after the incurrence of some initial
investment. The discrete interest rate (i) which will make
the sum of discounted cash flows equal the initial investment
is defined as the IRR, and reflects the effective discrete
interest rate that will be earned upon the investment. For
a series of twenty future cash flows, using the above notation,
the formula for NPV is:

$$NPV = \sum_{n=1}^{20} \frac{CF_n}{(1+i)^n} - INVEST$$

A negative NPV indicates that, at the chosen interest rate,
a shortfall (below the amount which would be expected at
that interest rate) equal to the magnitude of the negative
NPV will occur; a positive NPV indicates that, at the chosen
interest rate, an additional gain (above the amount which
would be expected at that interest rate) equal to the
magnitude of the NPV will occur; i.e., the defined interest
on the original investment has been exceeded by the positive
NPV amount.

The quantity $\frac{CF_n}{(1+i)^n}$ is defined as the discounted
present value (PV) of each flow n, written PV_n. In order to
determine IRR, the above formula must be solved for the
value of i which makes the difference zero; i.e.,

84

$$0 \quad = \quad \sum_{n=1}^{20} \frac{CF_n}{(1+i)^n} \quad - \quad \text{INVEST}$$

or

$$0 \quad = \quad \sum_{n=1}^{20} PV_n \quad - \quad \text{INVEST}$$

Graphically, the cash flow process can be pictured on a time line as follows:

+$ CF_1 CF_2 CF_3 CF_4 . . . . CF_{18} CF_{19} CF_{20}
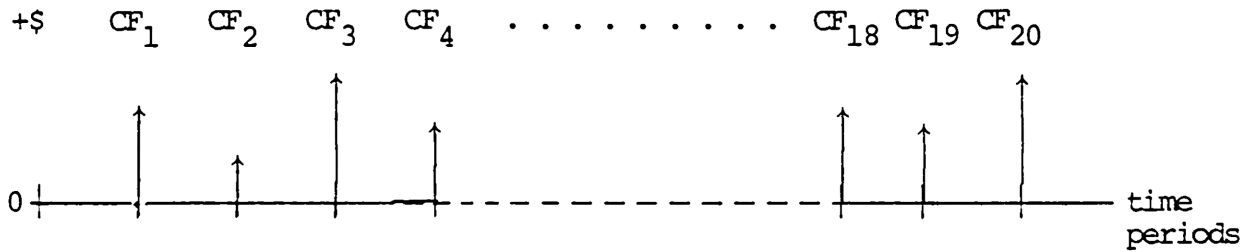
-$ INVEST

The future cash flows may also include zero or negative (additional future investment) values. Hence, a more complex case can be pictured on a time line as:

CF_1 CF_2 CF_3 CF_4 CF_5 CF_6 ... CF_{15} CF_{16} CF_{17} CF_{18} CF_{19} CF_{20}
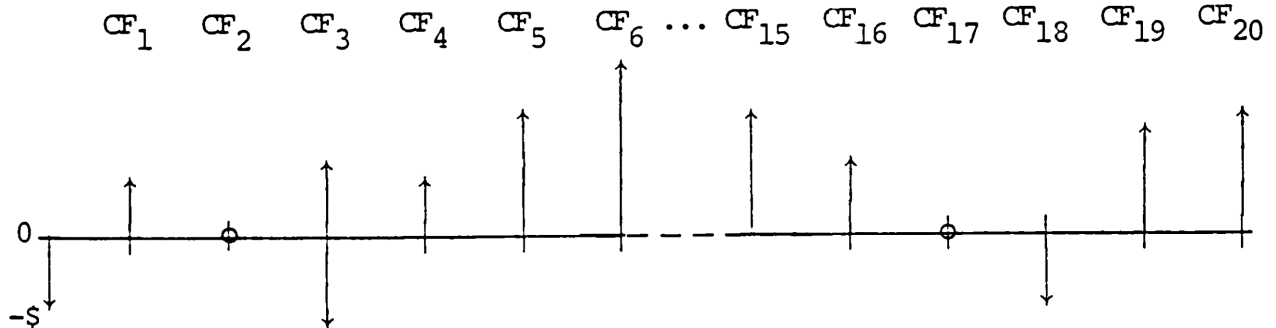
c. IRR Program Capabilities

(1) The programs developed herein will provide
the discrete IRR for any conceivable periodic (equally-
spaced) set of 21 positive, zero or negative cash flows.
The programs solve for two decimal places beyond the integer
percentage value (i.e. 7000%, .62% or 6.39% or 0.03%) unless
instructed by the user to proceed further; accuracy up to
the machine display capability (10 significant digits), if
desired, may be initially requested by the user. The programs
will also solve for a negative IRR, i.e., where net undis-
counted income is less than net undiscounted investments.

Either program will permanently store an
initial (positive, zero, or negative) "investment" and
twenty additional future positive, zero, or negative periodic
cash flows. In addition to the IRR, either program will
compute the NPV of these flows at any positive or negative
trial interest rate; the initial value (investment) or the
trial interest rate may each be individually modified by
the user at any time; the trial interest rate may also be
incremented or decremented by the user at any time (in lieu
of reentering a many-digit rate where only the last digit
or last several digit sequence is to be changed); the number
of cash flows (data entry values exclusive of the value
normally used for the initial investment) is automatically
accumulated and stored for possible later recall by the user
and each value can be individually modified or deleted

(changed to zero) at any time; previously unused data points from the twenty available may be _automatically_ added (on the end) at any time without recall of the number of entries. used previously; and the manual entry of long sequences of zero values may be avoided by instructing the calculator to jump over them in its automatic accumulation. Most of the above processes are executed _exclusively on the 10 user-defined keys_. Accordingly, this program is satisfactory for IRR, NPV or Profit analysis of any periodic sequence of 21 unique values or for an Investment followed by either 20 unique values or by 20 identical (annuity) values. The HP-67 program will also calculate the undiscounted breakeven point for the series of values but this subroutine is omitted from the SR-52 for lack of program memory space. In summary, these programs are satisfactory for rather complete analyses of any 21 flows spaced by uniform intervals — days, weeks, 12 day periods, months, quarters, years or whatever. Each of these programs will perform the function of at least three of the programs published by the respective manufacturers — using only user defined keys. Additional manufacturer programs can be accomplished by including limited manual use of other keys.

Functionally similar subroutines are arranged in a different sequence for each calculator in order to take advantage of the program search patterns unique to each calculator and thus minimize execution time for each. Since these different arrangements make direct

87

comparison of the programs difficult, applicable functional subroutines are individually reproduced within the analysis which follows, in addition to the complete listings provided for each program. Meanwhile, the reader should concentrate primarily upon the first page of each program.

(1) Step Ratios. Because the following programs are advanced and optimized programs, the SR-52/HP-67 step ratio is approximately 1.32, i.e., the number of steps required on the HP-67 to accomplish all of the functions which can be accomplished by the 208 step SR-52 program is 158. The remaining 50 HP-67 steps are used for additional functional, informational, and user-prompting subroutines not programmable on the SR-52 because of program memory space constraints. (The SR-52 program does not use 16 steps in order to make available two extra data registers. Thus the HP-67 program additionally uses these steps for additional routines not programmable on the SR-52.)

(2) Initial Data Entry Operations. Typical operations which can be performed by either of these single-card programs are: (p 99)

88

FIGURE 3

INTERNAL RATE OF RETURN (SR-52)

USER KEYED LABELS

A' CLR REG	B' SET n=DSP	C' [USED]	D' DO IRR	E' i DO
A $ INVEST	B STO CF's	C [RCL N]	D STO TRIAL i	E DO NPV

FLAGS: OFF/ON

0		X
1	X	
2	X	
3	X	
4	X	

OTHER LABELS (empty)

NOTES

i = INTEREST RATE

i_t = TRIAL i

Δi = INCREMENT OR DECREMENT TO i_t

n = CF# (PERIOD #)

PV = PRESENT VALUE

NPV = NET PV

REGISTERS

00	Cash Flow 1
01	CF #2
02	CF #3
03	CF #4
04	CF #5
05	CF #6
06	CF #7
07	CF #8
08	CF #9
09	CF #10
10	CF #11
11	CF #12
12	CF #13
13	CF #14
14	CF #15
15	CF #16
16	CF #17
17	CF #18
18	CF #19
19	CF #20
68	n
69	ΣPV
96	Δi or (n-1)
97	N=MAX n
98	TRIAL i/IRR
99	INVEST

SET STATUS

DISPLAY

	SCI		DEG	X
	ENG		RAD	
	FIX		GRAD	
X	FLOATING PT			

TRIG: X DEG

	STEP	KEY CODE	KEY ENTRY		
PRGM	000	B1	HLT	STOP	
CALL	001	46	*LBL	RCL Δi	
USER	002	18	*C'	& DO BELOW	
OR	003	43	RCL		
PRGM	004	09	9		
GIVE	005	06	6	RCL Δi	
Δi	006	46	*LBL	ADD Δi TO TRIAL i	
	007	10	*E'	& DO BELOW	
USER	008	44	SUM		
OR	009	09	9		
PRGM	010	08	8	+Δi → REG 98	
CALL	011	46	*LBL	DO NPV	
	012	15	E	(@ TRIAL i)	
	013	25	CLR	CLR DSP & CLR	
	014	01	1	REGs 60-69	
	015	44	SUM		
	016	09	9		
	017	08	8	+1 → REG 98	
	018	43	RCL		
	019	09	9		
	020	08	8	RCL (1 + TRIAL i)	
	021	45	yˣ		
	022	53	(		
	023	43	RCL		
	024	06	6		
	025	08	8	RCL(n-1) ADD 1 = n	
	026	85	+		000
	027	01	1		018 122
	028	54	)		066 192
	029	94	+/-	DO $(1+i_t)^{-n}$	

RCL (1 + TRIAL i)

$$\left. \begin{matrix} RCL(n-1) \\ ADD \\ 1 \end{matrix} \right\} = n$$

DO $(1+i_t)^{-n}$

89

STEP	KEY CODE	KEY ENTRY	NOTES
030	65	X	
1	36	*IND	
2	43	RCL	
3	06	6	
4	08	8	
5	95	=	$(1 + i_t)^{-n} \times (CF \#n) = PV_n$
6	44	SUM	
7	06	6	$+PV_n$ REG 69 = ΣPV_n
8	09	9	
039	01	1	
040	44	SUM	
1	06	6	
2	08	8	INCREMENT n → REG 68
3	43	RCL	
4	06	6	
5	08	8	
6	75	–	
7	43	RCL	
8	09	9	
9	07	7	
050	95	=	COMPUTE (n – N)
1	90	*IFZRO	
2	01	1	
3	09	9	
054	0	3	IF n = N GTO STEP 193
5	36	*IND	
6	43	RCL	
7	06	6	
8	08	8	IF n ≠ N, RCL CF#n
9	90	*IFZRO	
060	00	0	
1	03	3	
2	09	9	IF CF#n = 0, GTO STEP 039
3	41	GTO	
4	00	0	
5	01	1	IF FV$_n$≠0, GTO STEP 018
066	08	8	
7	46	*LBL	SET n = DSP(x)
8	17	*B'	
9	75	–	
070	01	1	
1	95	=	
2	48	*EXC	
3	09	9	
4	07	7	(n – 1) → REG 97; RCL PRIOR N
5	42	STO	
6	09	9	
7	06	6	PRIOR N REG 96
8	94	+/–	
9	85	+	

USER GIVE n

STEP	KEY CODE	KEY ENTRY	NOTES
080	01	1	
1	85	+	
2	13	C	RCL REG 97 → © (DO LBL)
3	95	=	DO(-PRIOR N + 1 + (n-1))
4	80	*IF POS	
5	13	C	IF n > PRIOR N, GTO C
6	50	*ST FLG	
7	00	Ø	IF n < PRIOR N, SET FLAG Ø
8	46	*LBL	RCL N
9	13	C	(N = MAX n)
090	43	RCL	
1	09	9	
2	07	7	
3	56	*RTN	STOP (OR RTN TO CALLING SBR)
4	46	*LBL	STO TRIAL i
5	14	D	
6	40	*x²	
7	30	*√X̄	DO ABSOLUTE VALUE OF i_t
8	42	STO	
9	09	.9	
100	08	8	$\lvert i_t \rvert$ REG 98
1	56	*RTN	STOP (DSP i_t)
2	46	*LBL	STO CF's
3	12	B	(FUTURE CASH FLOWS)
4	36	*IND	
5	42	STO	
6	09	9	
7	07	7	STO FV_n → REG (n-1)
8	01	1	
9	44	SUM	
110	09	9	
1	07	7	INCREMENT n
2	22	INV	
3	60	*IF FLG	
4	00	Ø	
5	13	C	IF FLAG Ø NOT SET, GTO C
6	43	RCL	
7	09	9	
8	06	6	IF FLAG Ø SET, RCL PRIOR N
9	48	*EXC	
120	09	9	
1	07	7	STO PRIOR N → REG 97; DSP n
122	86	*RSET	CLR FLAG Ø; GTO STEP 000
3	46	*LBL	DO IRR
4	19	*D'	
5	57	*FIX	
6	08	8	FORMAT 8 DEC DSP
7	93	.	
8	00	Ø	
9	02	2	0.02 = Δi

Left margin labels: USER GIVE i_t (at steps ~091–094); USER GIVE CF_n (at steps ~100–102); USER CALL (at step 122–3)

STEP	KEY CODE	KEY ENTRY	NOTES
130	42	STO	
1	09	0	Δi REG 96
2	06	6	
133	18	*C'	DO NPV → *C' (DO LBL)
4	80	*IF POS	
5	01	1	
6	03	3	
7	03	3	IF NPV NOT NEG, GTO STEP 133
8	02	2	
9	94	+/−	
140	22	INV	
1	49	*PROD	IF NPV NEGATIVE:
2	09	9	−(Δi÷2) = Δi' → REG 96
3	06	6	
144	18	*C'	DO NPV *C' (DO LBL)
5	22	INV	
6	80	*IF POS	
7	01	1	
8	04	4	
9	04	4	IF NPV STILL NEG, GTO STEP 144
150	93	.	
1	02	2	
2	94	+/−	
3	49	*PROD	
4	09	9	IF NPV POS OR 0:
5	06	6	(−0.2)XΔi = Δi" REG 96
6	43	RCL	
7	09	9	
8	06	6	(Δi" = NEW Δi)
9	75	−	
160	02	2	
1	52	EE	
162	04	4	
3	94	+/−	(LOW LIMIT = 0.0002)
4	95	=	
5	22	INV	
6	52	EE	DO (Δi − LOW LIMIT)
7	80	*IF POS	
8	01	1	
9	03	3	
170	03	3	IF Δi > LOW LIMIT, GTO STEP 133
1	43	RCL	
2	09	9	IF Δi < LOW LIMIT:
3	08	8	RCL IRR
4	22	INV	
5	57	*FIX	FORMAT FLOATING PT DSP
6	56	*RTN	STOP (DSP IRR)
7	46	*LBL	CLR REG
8	16	*A'	
9	47	CMS	CLR REG 00 THROUGH 19

*

USER CALL

STEP	KEY CODE	KEY ENTRY	NOTES
180	01	1	
1	17	*B'	CLR REG 97 (*8') (DO LBL)
2	25	CLR	CLR DSP & REG GO THRU 69
3	14	D	CLR REG 98 (D) (DO LBL)
4	42	STO	
5	09	9	
6	06	6	CLR REG 96
7	46	*LBL	INVEST $
8	11	A	
9	42	STO	
190	09	9	
1	09	9	INVEST (CLR) REG 99
192	86	*RSET	CLR ALL FLAGS; GTO 000 & STOP
193	01	1	(LBL E (NPV SBR) CONTINUANCE)
4	22	INV	
5	44	SUM	
6	09	9	
7	08	8	$(-1) \rightarrow$ REG 98 $\Rightarrow$ TRIAL i
8	43	RCL	
9	06	6	
200	09	9	
1	75	-	
2	43	RCL	
3	09	9	
4	09	9	
5	05	=	ΣPV - INVEST = NPV @ i_t
206	56	*RTN	STOP (DSP NPV @ i_t)
7	86	*RSET	GTO 000 IF RUN BUTTON IS
208			
9			
210			
1			
2			
3			
4		.	
5			
216			
7			
8			
9			
220			
1			
2			
223	END OF MEM		
4			
5			
6			
7			
8			
9			

STEPS 208 THRU 215 APPROPRIATED FOR USE AS REG #96

STEPS 216 THRU 223 APPROPRIATED FOR USE AS REG #97

MISTAKENLY HIT BY USER WHILE PRGRM HALTED AT STEP 206. THIS SAFETY FEATURE EXCLUDES FALLING THRU STEPS 208-223 AND THUS PROTECTS ALL OTHER STORED DATA FROM BEING ALTERED BY REG 96/97 "INSTRUCTIONS".

FIGURE 4

Internal Rate of Return (HP-67)

USER KEYED LABELS					
a CLR REG	b SET n=DSP	c DO PROFIT	d DO IRR	e ±Δi DO	
A $ INVEST	B STO CF's	C DO B.E. PT	D STO TRIAL i	E DO NPV	

FLAGS:	OFF/ON		OTHER LABELS		NOTES
0	X	0 BLANK	(FOR ERROR DSP)	n = CF # (PERIOD #)	
1	X	1	END SET n SBR b	n.m = NON-INT ENTRY	
2 RESET n	X	2 FIND	MAX n W/CF=Ø	(.m = USER MISTAKE	
3	X	3	END CFs SBR B	i = INTEREST RATE	
		4	MINUS NEXT CF	i_t = TRIAL i	
		5 CYCLE	NEXT DEC (IRR)	Δi = INCREMENT OR	
		6	SKIP Ø CF SBR	DECREMENT TO i_t	
		7 TEST Δi	LIM(IRR SBR)	PV = PRESENT VALUE	
		8	END IRR SBR d	NPV = NET PRESENT VALUE	
		9	END NPV SBR E	B.E. = BREAK-EVEN	

REGISTERS

0	CASH FLOW 1				
1	CF #2				
2	CF #3				
3	CF #4	USER			
4	CF #5	GIVE	STEP	KEY CODE	KEY ENTRY
5	CF #6	$INV	///Ø/	///////////	///////////

		GIVE	STEP	KEY CODE	KEY ENTRY		
6	CF #7		001	31 85 11	f LBL A	STO INVEST	
7	CF #8	USER	002	33 11	STO A	INVEST → REG A	
8	CF #9	GIVE	003	23 02	DSP 2	FORMAT 2 DEC DSP	
9	CF #10	n.m	004	35 22	h RTN	STOP. (DSP $ INVEST)	
S0	CF #11		005	32 25 12	g LBL b	SET n = DSP(x)	
S1	CF #12		006	31 83	f INT	DSP INT n	
S2	CF #13		007	35 82	h LST x	DSP INPUT VALUE (x=n.m)	
S3	CF #14		008	23 09	DSP 9	FORMAT 9 DEC DSP	
S4	CF #15		009	32 61	g x≠y	IF INPUT VALUE ≠ INT	
S5	CF #16	ERR	010	22 00	GTO Ø	DSP ERR(n.m) STOP.	
S6	CF #17		011	23 00	DSP Ø	OW, FORMAT INT DSP	
S7	CF #18		012	02	2		
S8	CF #19		013	00	0		
S9	CF #20		014	35 52	h x↔y		
			015	32 81	g x>y	IF n > 20	
A	INVEST	ERR	016	22 00	GTO Ø	DSP ERR (n) STOP.	
B	N = MAX n		017	01	1		
C	Δi or BE Pt		018	35 54	h R		
D	TRIAL i/IRR		019	32 51	g x=y	IF n = 1	
E	NPV		020	22 01	GTO 1	GTO LBL 1	
		ERR	021	32 71	g x<y	IF n < 1	
I	N MAX/PRIOR		022	22 00	GTO Ø	DSP ERR (n) STOP.	
			023	31 25 01	f LBL 1	END SET n SBR b	

SET STATUS

		024	35 53	h R↓	If 1< (n=x)<20
DISPLAY	TRIG	025	51	-	(n-1) = FAKE PRIOR n
		026	35 33	h ST I	→ REG I
SCI	X DEG	027	34 12	RCL B	RCL PRIOR N
ENG	RAD	028	31 84	f -x-	FLASH PRIOR N
X FIX	GRAD	029	35 54	h R↑	RCL n; (n=x)
FLOATING PT					

94

STEP	KEY CODE	KEY ENTRY	NOTES
030	32 81	g x>y	IF n >PRIOR N
1	33 12	STO B	STO n (=New N) → REG B
2	35 51 02	h SF 2	EITHER WAY, SET FLAG 2
3	35 22	h RTN	DSP n & STOP.
4	31 25 12	f LBL B	STO CF's
5	23 00	DSP Ø	FORMAT INT DSP
6	02	2	
7	01	1	
8	35 34	h RC I	
9	01	1	
040	61	+	
1	32 51	g x=y	IF PRIOR N = 20
2	22 00	GTO Ø	DSP ERR (21) STOP.
3	35 54	h R↑	
4	33 24	STO (i)	OW, STO CF #n REG #(n-1)
5	35 53	h R↓	
6	35 33	h ST I	STO(PRIOR N+1) REG I
7	35 71 02	h F? 2	IF FAKE PRIOR N WAS USED
8	35 71 02	h F? 2	SKIP NEXT STEP
9	22 03	GTO 3	OW, GTO LBL 3
050	31 84	f -x-	FLASH FAKE PRIOR N USED
1	34 12	RCL B	RCL ACTUAL PRIOR N
2	32 81	g x>y	IF PRIOR N > FAKE PRIOR N
3	35 83	h ST I	STO PRIOR N → REG I
4	23 02	DSP 2	EITHER WAY, FORMAT 2 DEC DSP
5	35 54	h R	
6	35 72	h PAUSE	FLASH CF #n ($)
7	23 00	DSP Ø	REFORMAT INT DSP
8	31 25 02	f LBL 2	FIND MAX n W/CF = Ø
9	35 34	h RC I	RCL MAX n
060	31 51	f x=Ø	IF MAX n = Ø
1	22 03	GTO 3	GTO LBL 3
2	31 33	f DSZ	OW, DECREMENT n
3	35 84	h SPACE	
4	34 24	RCL (i)	RCL CF #n
5	31 54	f x=Ø	IF CF #n = Ø
6	22 02	GTO 2	GTO STEP Ø58
7	31 34	f ISZ	OW, INCREMENT n
8	31 25 03	f LBL 3	END CFs SBR B
9	35 34	h RC I	RCL n (=NEW N)
070	33 12	STO B	STO N → REG B
1	35 22	h RTN	STOP.(DSP N)
2	31 25 13	f LBL C	DO B.E. POINT
3	00	Ø	
4	35 33	h ST I	STO(Ø = n-1) → REG I
5	34 11	RCL A	RCL INVEST (=INVEST - CFØ)
066	31 25 04	f LBL 4	MINUS NEXT CF
7	34 24	RCL (i)	RCL CF #n (=CF_n)
8	51	-	DSP ((INVEST - ΣCF_{n-1}) - CF_n) 066
9	35 54	h R↑	STO (INVEST - ΣCF_n) → STACK 087

95

STEP	KEY CODE	KEY ENTRY	NOTES
080	34 12	RCL 8	RCL N
1	35 34	h RC I	RCL (n-1)
2	32 51	g x=y	IF (n-1) = N
3	22 31 13	GTO f c	GTO LBL C (FINDS ERR)
4	31 34	f ISZ	OW, STO n→ REG I
5	35 54	h R↑	DSP (INVEST - CF_n) = BALANCE
6	31 81	f X>Ø	IF STILL POS
087	22 04	GTO 4	GTO LBL 4
8	35 82	h LSTx	OW,ADD BACK CF_n TO BALANCE
9	61	+	= INVEST - ΣCF_{n-1}
090	35 82	h LST x	= POS CASH BALANCE
1	81	÷	÷CF_n = FRAC PART OF YEAR
2	31 33	f DSZ	LAST FULL YR USED REG I
3	35 34	h RC I	IF LFYU ≠ Ø, RCL LFYU
4	61	+	ADD LFYU OR Ø TO FRAC
5	33 13	STO C	STO B.E. POINT REG C
6	35 54	h R	RCL PRIOR N
7	35 33	h ST I	STO PRIOR N REG I
8	35 53	h R↑	RCL B.E. POINT
9	23 06	DSP 6	FORMAT 6 DEC DSP
100	35 22	h RTN	STOP (DSP B.E. POINT)
1	32 25 13	g LBL c	DO PROFIT
2	00	Ø	
3	33 14	STO D	
4	31 22 15	f GSB E	(DO NPV @ i=Ø) (E) (DO LBL)
5	31 81	f x>Ø	IF PROFIT > Ø STOP
6	35 22	h RTN	IF PROFIT < Ø CONTINUE
7	31 25 14	f LBL D	STO TRIAL i
8	31 71	f X<Ø	IF TRIAL i < Ø
9	22 00	GTO Ø	DSP ERR (-TRIAL i)
110	33 14	STO D	OW, STO TRIAL i→ REG D
1	23 04	DSP 4	FORMAT 4 DEC DSP
2	35 22	h RTN	STOP (DSP TRIAL i)
3	32 25 14	g LBL d	DO IRR
4	83	.	
5	00	Ø	
6	02	2	0.02 = i
117	31 25 05	f LBL 5	CYCLE NEXT DEG (IRR)
8	33 13	STO C	STO i REG C
119	34 13	RCL C	RCL i
120	32 22 15	g GSB e	(DO NPV SBR) (e) (DO LBL
1	07	7	
2	42	CHS	
3	35 33	h ST I	STO(-7)→ REG i
4	35 53	h R↓	
5	31 81	f X>Ø	IF NPV POS
6	22.24	GTO (i)	GTO STEP 119
127	34 13	RCL C	IF NPV NEG, RCL Δi
8	02		
9	81	÷	

Left margin labels:
USER OR PRGM CALL

USER OR PRGM GIVE i_t

ERR

USER CALL ONLY

96

	STEP	KEY CODE	KEY ENTRY	NOTES
	130	42	CHS	$(-\Delta i \div 2) = \Delta i'$
ⓔ	1	32 22 15	g GSB e	(DO NPV SBR) ⓔ (DO LBL)
AND	2	01	1	
Ⓔ	3	01	1	
MAY	4	42	CHS	
BE	5	35 33	h ST I	STO (-11) REG I
PRGM	6	35 53	h R↓	RCL NPV @ TRIAL i
OR	7	31 71	f X< Ø	IF NPV NEG
USER	138	22 24	GTO (i)	GTO STEP 127
CALL	139	22 07	GTO 7	IF NPV POS, GTO STEP 184
	140	32 25 15	g LBL e	±Δi → DO BELOW
	1	34 14	RCL D	
	2	61	+	
	3	33 14	STO D	+Δi → REG D
	4	31 25 15	f LBL E	DO NPV @ TRIAL i
	5	34 14	RCL D	RCL TRIAL i = i_t
	6	01	1	
	7	61	+	
	8	33 14	STO D	STO $(1 + i_t)$ REG D
	9	00	Ø	
	150	33 15	STO E	STO (n=Ø) → REG E
	151	35 24	h X⇄I	STO n → REG I; DSP (-26) OR Ø
	2	34 14	RCL D	RCL $(1 + i_t)$
	3	35 34	h RC I	RCL (n-1)
	4	01	1	ADD
	5	61	+	1
	6	42	CHS	
	7	35 63	h y^x	DO $(1 + i_t)^{-n}$
	8	34 24	RCL (i)	
	9	71	X	$(1 + i_t)^{-n}$ X (CF #n) = PV_n
	160	34 15	RCL E	
	1	61	+	
	2	33 15	STO E	+PV_n → REG E = ΣPV_n
	3	35 34	h RC I	RCL n
	164	01	1	
	5	61	+	
	6	35 33	h SI I	INCREMENT n → REG I
	7	34 12	RCL B	
	8	32 51	g x=y	IF N=n
	169	22 09	GTO 9	GTO STEP 201
	170	34 24	RCL B	IF N≠n, RCL CF #n
	1	31 51	g x=y	IF CF #n = Ø
	2	22 06	GTO 6	GTO LBL 6
	3	02	2	IF CF #n ≠ Ø
	4	06	6	
	5	42	CHS	STO (-26) REG I; DSP n
	6	35 24	h x⇄I	GTO STEP #151
	7	22 24	GTO (i)	
	178	31 25 06	f LBL 6	SKIP Ø-CF (SBR E)
	9	01	1	

STEP	KEY CODE	KEY ENTRY	NOTES
180	09	9	
1	42	CHS	
2	35 24	h x⇄I	STO (-19) → REG I; DSP n
183	22 24	GTO i	GTO STEP 164
184	31 25 07	f LBL 7	TEST i LIM (IRR SBR)
5	02	2	
6	43	EEX	
* 7	05	5	
8	42	CHS	LOW LIMIT = 0.00002
9	34 13	RCL C	
0	83	.	
1	01	1	
2	71	X	$\Delta i \div 10 = \Delta i'$
3	32 81	g x>y	IF $\Delta i'$ > LOW LIMIT, GTO STEP 117
194	22 05	GTO 5	OW, CONTINUE BELOW
195	31 25 08	f LBL 8	END IRR SBR (d)
6	34 12	RCL B	
7	35 33	h STI	STO N → REG I
8	23 04	DSP 4	FORMAT 4 DEC DSP
9	34 14	RCL D	DSP IRR
200	35 22	h RTN	STOP.
201	31 25 09	f LBL 9	END NPV SBR (E)
2	34 14	RCL D	RCL(1 + TRIAL i) = (1 + IRR)
3	01	1	
4	51	-	
5	33 14	STO D	STO IRR → REG D
6	34 15	RCL E	
7	34 11	RCL A	
8	51	-	ΣPV_n - INVEST = NPV
9	33 15	STO E	STO NPV → REG E
* 210	23 02	DSP 2	FORMAT 2 DEC DSP → $
* 211	31 24	f RND	ROUND NPV TO NEAREST CENT
2	31 51	f x=Ø	IF ROUNDED NPV = Ø
213	22 08	GTO 8	GTO LBL 8
* 214	35 82	h LSTx	OW, RCL ACTUAL NPV
5	35 22	RTN	STOP. (RTN TO CALLING SBR)
6	32 25 11	g LBL a	CLR REG
7	61	÷	STO DSP → REG LSTx
8	31 43	f CLREG	CLR REGs Ø THRU 9
9	31 42	f P⇄S	EXCH PRI/SEC REGs
220	31 43	f CLREG	CLR REGs SØ THRU S9
1	23 09	DSP 9	FORMAT 9 DEC DSP
2	44	CLx	CLR DSP
3	35 61 02	h CF 2	CLR FLAG 2
4	35 22	h RTN	STOP.
///5			
///6			
///7			
///8			
///9			

DATA		ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
INVESTMENT:	SR 52:	INVEST	A	INVEST	(As entered)
	HP 67:	INVEST	A	INVEST	($ and ¢)
CASH FLOWS:		CF#1	B	1.	
		CF#2	B	2.	
		⋮	⋮	⋮	
		CF#n	B	n.	
		⋮	⋮	⋮	
		CF#20	B	20.	
DECIMAL TRIAL INTEREST RATE (i)					
	SR-52	i_t	D	i_t	(As entered)
	HP-67	i_t	D	i_t	(n.wxyz)
EXAMPLES:	i_t=10%	.1	D	0.1000	
	i_t=7654.321%	76.54321	D	76.5432	(All digits stored)

(3) Data Modification Operations.

		ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
CHANGE INVESTMENT		NEW INVEST	A	NEW INVEST	
CHANGE INTEREST RATE		NEW i_t	D	NEW i_t	
CHANGE CASH FLOW FOR PERIOD n (CF_n)					
	SR-52:	n	*B	m.	(n-1)
		CF_n	B	n.	
	HP-67	n	fb	N.	(flashes 4X)
				n.	
		CF_n	B	n.	(flashes 4X)
				CF_n	(flashes 1X)
				N.	(Highest n used*) [see *note next page]

*NOTE: If n = N <u>and</u> CF_n = 0 is entered, the HP-67 program will automatically redefine N as the highest period for which a non-zero cash flow is stored in memory. Accurate definition of N speeds program operations but is not required to obtain the solutions. Redefinition on the SR-52, if desired subsequent to the above entries, is accomplished by:

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
		*B	M.	(N-1)
		*RSET	N.	(=Prior N-1)

INCREMENT i_t by ±Δi AND AUTOMATICALLY RUN NPV

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
SR-52:	±Δi	*E	NPV	(10 DIGITS)
HP-67:	±Δi	fe	NPV	($ and ¢)

CLEAR ALL DATA FOR NEW CASE:

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
SR-52		*A	0.	
HP-67:		fa	0.000000000	

(4) Undiscounted Subroutines.

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
PROFIT MARGIN				
HP-67:		fc	PROFIT	($ and ¢; DESTROYS i_t)
SR-52:	0	D	0.	(i_t)
		E	PROFIT	(DIGITS APPLICABLE TO CF's USED)

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
OTHER (HP-67 ONLY)				
FIRST BREAK EVEN POINT:		C	B.E.PT	(6 Decimal Places)
ESTIMATE i_t:		C	B.E.PT	
		fc	PROFIT	
		RCL C	B.E.PT	
		RCL A	INVEST	
		X ÷	i_t	(2 Decimal Places)
		DSP 4	i_t	(4 Decimal Places)
		f RND	i_t	
		D	i_t	(STORE D)

(5) Time-Valued Subroutines.

		ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
NPV @ i_t:	SR-52		E	NPV	(10 DIGITS)
	HP-67		E	NPV	($ and ¢)
IRR :	SR-52		*D	IRR	(10 DIGITS)
	HP-67		fd	IRR	(4 Decimal Places)
INCREMENT/DECREMENT i_t & RUN NPV					Discussed Above

(6) <u>Error Protections</u>. (When User attempts to store invalid values)

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
NEGATIVE TRIAL INTEREST USING (STO Trial i) Key RATE (i_t):				
SR-52:	$-i_t$	D	$+i_t$	(i_t Stored)
HP-67:	$-i_t$	D	Error	
		CLx	$-i_t$	(Not stored)
(IF NEGATIVE i_t DESIRED)				
SR-52	0	D	0.	
	$-i_t$	*E'	NPV	($-i_t$ stored)
HP-67	0	D	0.0000	
	$-i_t$	fe	NPV	($-i_t$ stored)
NEGATIVE (n 0) or TOO LARGE (n 20) USING (SET n=DSP) KEY.				
SR-52:	n	*B'	M	(n-1)

Although the display is normal, no data has yet been stored. Alert user will note minus signs and/or increments of n in wrong directions or beyond 0 to 20 limits. However, some data <u>must</u> be entered against key B in the usual manner in order to avoid undesired redefinition of N, i.e.,

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
	Any Value	B	n	(no data stored)
HP-67:	n	fb	Error	
		CLx	n	

	ENTER	PRESS KEYS	DISPLAY	FORMAT & NOTES
NON-INTEGER PERIOD NUMBER (±n.p) USING (SET n=DSP) KEY,				
SR-52:	±n.p	*B'	±m.p	((n.p)-1)
Same comments and process as above				
	Any Value	B	±n.p	(No data stored)
HP-67:	±n.p	fb	Error	
		CLx	±n.p	
21st CASH FLOW WHEN USING (STO CF's) KEY:				
SR-52:	CF#21	B	21.	(No data stored)
	CF#22,etc.	B	22.	(No data stored)
Error Noted by User. To correct:				(Redefine N)
		*B'	21.	
		*B'	20.	
		*RESET	20.	
HP-67:	CF#21	B	Error	
		CLx	21.	(No data stored)

(7)   Sample Calculations, SR-52 IRR Program.

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
INVEST	$1000	A	1000.	0.8
CF#1	$ 500	B	1.	1.2
CF#2	$ 600	B	2.	1.2
TRIAL i (ANNUAL)	.1	D	0.1	0.8
DO NPV		E	-49.58677686	3.5
DECR. i,DO NPV	-.04	*E'	5.69597721	3.8
*DECR. i,DO NPV	-.0111	*E'	22.04961262	3.8
*DO IRR		*D'	0.0639	32.5
				47.6

*NOTE: LOWEST RUN TIME NORMALLY OCCURS WITH (i_t = IRR - 0.0111)

e.g., WITH i_t = 0.0528, RUN TIME FOR ABOVE IRR IS

22.8 SECONDS.

CHANGE INVEST	$1200.75	A	1200.75	0.8
ADD CF#3	$ 100	B	3.	1.2
CHANGE CF#1	1	*B'	0.	1.8
	$ 555	B	1.	1.0
ADD CF#4	$ 200	B	4.	1.2 ·
ADD CF#15	15	*B'	14.	1.8
	$ 352.16	B	15.	1.4
ADD CF#16	$ 100	B	16.	1.2
DELETE CF#16	(16)	*B'	15.	1.8
	0	B	16.	1.0
RCL N		C	16.	0.3

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
REDEFINE N		*B'	15.	1.8
		*RSET	15.	--
TRIAL i	.1	D	0.1	0.8
DO NPV		E	95.7016984	14.0
INCR i, DO NPV	.05	*E'	-41.07430984	14.5
DECR i, DO NPV	-.025	*E'	21.92961966	14.5
DO IRR		*D'	0.1332	179.0
				285.7
				(4'46")

NOTE: WITH i_t = 0.1221 (VICE .1250) RUN TIME FOR
IRR is 64 SECONDS (VICE 179).

(8)   Sample Calculations, HP-67 IRR Program.

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
INVEST	$1000	A	1000.00	0.8
CF#1	$ 500	B	1.	1.9
CF#2	$ 600	B	2.	1.9
DO B.E.PT(YRS)		C	1.833333	2.8
DO PROFIT		fc	100.00	5.0
TRIAL i (ANNUAL)	.1	D	0.1000	0.8
DO NPV		E	-49.59	6.1
(DISPLAY 9)			(-49.58677686)	

105

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
DECR i, DO NPV	-.04	fE	5.70	6.3
*DECR i, DO NPV	-.011	fE	22.05	6.6
*DO IRR		fD		53.8
				86.0

*NOTE: LOWEST RUN TIME NORMALLY OCCURS WITH (i_t = IRR-0.0111) e.g., WITH i_t = 0.0528, RUN TIME FOR ABOVE IS 37.0 SECONDS.

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
CHANGE INVEST	$1200.75	A	1200.75	0.8
ADD CF#3	$ 100	B		1.9
CHANGE CF#1	1	fb	3. (Flashes 4X)	
			1.	7.1
	$ 555		1. (Flashes 4X)	
			555.00 (Flashes 1X)	
			3.	2.2
ADD CF #4	200	B	4.	1.9
ADD CF#15	15	fb	4. (Flashes 4X)	
			15.	7.1
	$ 352.16	B	15. (Flashes 4X)	
			352.16 (Flashes 1X)	
			15.	8.2
ADD CF#16	$ 100.00	B	16.	1.9
DELETE CF#16	(16)	fb	16. (Flashes 4X)	
			16.	7.1
			16.	
			16. (Flashes 4X)	

	ENTER	PRESS KEYS	DISPLAY	RUN TIME IN SECONDS
	0	B	0.00 (Flashes 1X)	
			15.	9.2
DO B.E. PT (YRS)		C	2.457500	3.9
DO PROFIT		fc	606.41	17.2
ESTIMATE i_t				
		RCL C		
		RCL A		
		X÷		
		DSP 4	0.2055	NA
TRIAL i		f RND	0.2055	
	(.2055)	D	0.2055	0.8
DO NPV		E	-154.36	19.4
DECR i, DO NPV	-.08		20.57	19.7
DO IRR		fd	0.1332	260.0
				460.4
				(7'20")

NOTE: WITH i_t = 0.1221 (VICE .1255) RUN TIME FOR
IRR IS 123 SECONDS (VICE 260).

(9) Redefining IRR Program Decimal Accuracy
Limits. In order to carry the IRR calculations to a digit
limit other than four (or to reset to four digits
subsequently) the procedures are:

107

ENTER		PRESS KEYS	DISPLAY

SR-52

ENTER		PRESS KEYS	DISPLAY
Go to Step 162		GTO 1,6,2	(0.1332)
Shift to Learn Mode		LRN	162 04
In same step, Reset Decimal Digit			
Limit (1 through 9):		9	163 94
Backstep to Review Change		*BST	162 09
Shift to Run Mode		LRN	(0.1332)
Enter Trial i	0.1221	D	0.1221
DO IRR:		*D'	0.133286694

Run time for above IRR is 7 minutes and 38 seconds for

9 decimal digits. Roundoff error (amount NPV misses zero

at IRR) decreases from +22.371990 at IRR = 0.1332 to

+0.0000130 at IRR = 0.133286694. Repeat above to reset

number of digits again; do not reread card if same set of

data needed for further analysis — data will be altered

thusly in this program because it uses some program steps as

data registers.

HR-67

ENTER	PRESS KEYS	DISPLAY		
Shift switch to Program Mode	W/PRGM	--		
Delete Step 214	GTO.214	214	35	82
	h DEL	213	22	08
Delete Step 211	GTO.211	211	31	24
	h DEL	210	23	15

ENTER	PRESS KEYS	DISPLAY
Delete Step 210	h DEL	209 23 15
Delete Step 187	GTO.187	187 05
	h DEL	186 43
Mentally add 1 to Decimal Limit Desired		
Enter limit (2 through 10)	1	187 01
	0	188 00
Shift switch to Run Mode	RUN	(0.1332)
Enter Trial i 0.1221	D	0.1221
DO IRR:	fd	0.1333
	DSP 9	0.133286694

Run time for above IRR is 12.0 minutes for 9 decimal digits.
Roundoff Error (Amount NPV misses zero at IRR) decreases from
+22.372000 ¢ at IRR = 0.1332 to 0.0000000 ¢ at IRR = 0.133286694,
i.e., beyond the machine capability to determine. To reset
number of digits to 4, merely reread card. To reset to some
other number of digits, reread card and then repeat above
process. No data will be altered.

(10) Comparison With Manufacturer Programs for IRR.
The Hewlett-Packard "Internal Rate of Return" program and
(BD1-01A) yields the same interest rate for the previous
problem in 116.3 seconds (vice 260 seconds with the program
developed above). However, this card uses 197 program memory
steps to do nothing but IRR and always requires manual entry
of all values, i.e.:

INVEST	$1200.75 → A	CF 8	0 → C
CF1	555 → C	CF 9	0 → C
CF2	600 → C	CF10	0 → C
CF3	100 → C	CF11	0 → C
CF4	200 → C	CF12	0 → C
CF5	0 → C	CF13	0 → C
CF6	0 → C	CF14	0 → C
CF7	0 → C	CF15	352.16 → C
		D	→ 13.32866940

If the user keys in a mistake, the entire data entry process must be begun anew. This program will not accept a negative or zero Investment but will accept up to 44 positive cash flows, limited to five digits each. However, for more than 22 cash flows, accuracy decreases to ± 0.01 % (.0001 decimal). If there are negative as well as positive cash flows, the program accepts up to 22 cash flows. This program may sometimes halt prematurely with ERROR in the display if the actual IRR is greater than 100% or if the sign of the cash flows is reversed more than once. Additionally, since more than one interest rate is considered correct in the mathematical sense when the sign of the cash flows is reversed more than once, the user has no way of determining which rate this program has found. (Conversely, the program developed by the authors of this thesis, if given a trial interest rate of zero, will always find the IRR (positive or negative) closest to zero.) Finally, this program will not warn the user of improper data entries or

110

results, and stored data cannot be used for other purposes but must be reentered against other programs to determine break-even point, profit, net present value, etc.

In summary, then, card BD1-01A is considered inferior to the program developed herein unless IRR is required for more than 20 positive cash flows (each limited to five digits) and no other information or analysis is desired. Otherwise, the card BDI-01A user must be satisfied with a program that may more quickly provide one of the possible solutions or may provide no solution at all, or may not provide the best solution. The BD1-01A user must also be very careful to avoid input errors since stored data is very difficult to review or to change.

The Texas Instruments "Variable Cash Flow (Present Value)" program card (FI1-23) will only accept ten cash flow values and hence will not work the previous problem. Card FI1-23 uses 204 program steps whereas the SR-52 card developed herein uses 224 to accomplish all of the previously discussed capabilities. Thus Card FI1-23 is considered to represent particularly inefficient programming; the card has little practical utility. It does work quickly and well for 10 values, and data can be individually reviewed or altered easily. This card, FI1-23, does yield rates dependent upon the trial interest rate, similar to the program developed by the authors. The five-cash-flow data set listed on page 131 in the Texas Instruments Finance Library for this card requires 151 seconds to run, as below, with accuracy

111

specified to nearest cent of NPV. Similar data runs on the
program developed herein in 133 seconds without specifying
accuracy, but additional accuracy (using the same trial
interest rate) requires progressively longer times as
shown below.

```
INVEST       $40,200
CF #1            400
CF #2          9,200
CF #3         11,560
CF #4         17,048
CF #5         45,484
Trial i      15%
Accuracy     .01 (Specified for FI1-23 only)
```

IRR Results:	Solution	Seconds	Amount NPV misses 0 @IRR
(FI1-23) :	0.1978193283	151	+ $0.0095657
Program	0.1978	133	+ $2.5701892
Herein:	0.19781	149	+ $1.2452525
	0.197819	196	+ $0.0528593
	0.1978193	223	+ $0.0131138
	0.19781939	265	+ $0.0011901
	0.197819398	321	+ $0.0001301
With Trial i = 0.1867:	0.197819398	235	+ $0.0001301

A shrewd user of the program developed herein, of course,
would normally run the data once to obtain 0.1978 and then,
if desired, modify the program and use a trial i of 0.0111
less as mentioned earlier; i.e., 0.1867, to obtain the value
0.197819398, as shown above, with a total running time of
133 + 235 = 368 seconds.

112

It should be obvious from the above that experienced programmers can, in many cases, develop programs more subtle or more efficient than the program found in the libraries published by any manufacturer. The manufacturers are motivated to sell calculators and calculator cards, not to create particularly efficient programs. Thus the typical approach appears to be that of including generally useful, but simplistic, programs in published libraries, so long as the programs function without error for most input possibilities.

On the other hand, the usefulness of the published programs to less experienced programmers is immense because the cards allow calculations such users might have no idea how to (mathematically) approach, much less program.

10. Advanced Programming Optimization Techniques

The previously developed IRR programs (Figures 3 and 4) and additional programs are used below to clarify subtle differences required when programming the SR-52 or HP-67.

a. Labeled, Direct, and Indirect Relative Addressing.

The simplest way to call a subroutine is merely to give it a label, as discussed previously in Section IV.C., and call the label as required. Equally simple direct addressing (not available on the HP-67) typically uses a "GTO XYZ" statement in lieu of a label. Thus "GTO 123", stored as program steps on the SR-52, will branch the program

113

to step 123 whenever this calling instruction is encoun-
tered. But this branch is an unconditional transfer; the
program does not return to the calling point. For true
subroutines, the similar instruction of "SBR 123" is used
instead; this call returns the program counter to the step
immediately after the "3" when the program encounters a
"RTN" instruction subsequent to step 123.

Indirect relative addressing is used on the
HP-67, for unconditional branching, by first placing a
negative integer in the I-Register such that when the instruc-
tion "GTO (i)" is next encountered during program execution,
the program will halt, note its own current step number,
backstep the number of steps specified by the current (nega-
tive) number in the I-register and there resume program
execution. Similarly, f GSB (i) is used to call and execute
a subroutine and then return to the step after the call.
These features each require two or three extra steps, com-
pared to the SR-52, of program memory for each branch thus
defined; these features are not included or normally needed
on machines such as the SR-52 which have direct addressing
capabilities.

Examples of these types of addressing are ex-
tracted from Figures 3 and 4 (IRR Programs) and reproduced
below, as Figure 5.

FIGURE 5

TYPES OF CALLS

SR-52 MAIN ROUTINE (IRR)

STEP	KEY ENTRY
125	*LBL
	*D'
	*FIX
	8
	.
	0
	2
130	STO
	9
	6
133	*C'
	*IF POS
	1
	3
	3
	2
	+/-
140	INV
	*PROD
	9
	6
144	*C'
	INV
	*IF POS
	1
	4
	4
150	.
	2
	+/-
	*PROD
	9
	6
	RCL
	9
	6
	-
160	2
	EE
	4
	+/-
	=
	INV
	EE
	*IF POS
	1
	3
170	3
	RCL
172	9

STEP	KEY ENTRY
173	8
	INV
	*FIX
176	RTN

** DIRECT ABSOLUTE ADDRESS CALL (steps 133)

** DIRECT ABSOLUTE ADDRESS CALL (steps 144)

DIRECT ABSOLUTE ADDRESS CALL (steps 170)

HP-67 MAIN ROUTINE (IRR)

STEP	KEY ENTRY
113	g LBL d
	.
	0
	2
	f LBL 5
	STO C
	RCL C
120	g GSB e
	7
	CHS
	h ST I
	h R↓
	f X>0
126	GTO i
127	RCLC
	2
	÷
130	CHS
	g GSB e
	1
	1
	CHS
	h ST I
	h R↓
	f X<0
138	GTO (i)
139	GTO 7
//////////////	
184	f LBL 7
	2
	EEX
	5
	CHS
	RCL C
190	.
	1
	X
	g X>Y
194	GTO 5
	f LBL 8
	RCL B
	b ST I
	DSP 4
	RCL D
200	b RTN

* INDIRECT RELATIVE CALL (120)

LABEL CALL ** (g GSB e)

INDIRECT RELATIVE CALL (138)

LABEL CALL

LABEL CALL (194)

**LABEL CALLS; ARROWS OMITTED IF REQUIRED FOR CLARITY OF OTHER CALLS; CALLED LABEL NOT SHOWN HEREON.

115

b. Label Search Mechanisms and Subroutine Locations.

The call for a labeled subroutine on the SR-52 causes the program step counter to immediately reset to 000 and then begin a downward search looking for "LBL". Each label thus encountered is then further tested to determine if it is the requested label. If so, execution of that label begins; if not, the downward search is resumed. Thus SR-52 label-location times are directly proportional both to the distance between 000 and the called label and to the number of intervening labels, but the step number of the call itself has little, if any, effect upon location time.

The call for a labeled subroutine on the HP-67 causes a search to begin downward from the point of call in a manner otherwise similar to the SR-52. Thus, HP-67 label location times are directly proportional to the downward distance between the step number of the call and the label, and to the number of intervening labels. (The HP-67 "falls through" its last step, 224, into step 001 if required during this process.)

For subroutines which are infrequently called by the program, the length of label location time is relatively unimportant. But for frequently called or for iterative routines, subroutine location often becomes the most critical factor in optimizing program run times.

Run time is minimized on the SR-52 by placing the most frequently called labeled-subroutines near the

beginning of the program or, as an alternative, placing a series of labeled GTO XYZ (Step Number) statements at or near the beginning; e.g., *LBL A GTO 046, *LBL B GTO 113, *LBL ⁵C' GTO 214 defines three labeled subroutines (A, B, C') which, respectively, begin execution at steps 046, 113, and 214. In these cases the label names are merely moved, but the GTO XYZ instructions require four additional steps per label. Thus the method is only applicable where extra step space is available.

    c.   <u>Nesting and Stacking Labels or Subroutines</u>.

        Program execution normally stops only when a "HLT" or "R/S" instruction is encountered, or when an error condition is created (such as dividing by zero or branching to a non-existent label) or when the "RTN" instruction is encountered in the <u>primary</u> routine being executed. Conversely, labels function only to identify the starting point of a <u>called</u> subroutine. Thus, encountering an <u>uncalled</u> label during program execution has no effect at all; the label is merely ignored. For this reason, labels can sometimes be <u>nested</u> such that the same single step number ends every subroutine in the nest. For example, consider the following SR-52 subroutine (assume any non-zero value, j, is stored in REG 00):

STEP	KEY ENTRY	LABEL CALLED	VALUE CALCULATED & STORED IN REG 99
001	*LBL		
2	C	C	$7 + 73 + 10 + j$ = $90 + j$
3	7		
4	+		
5	*LBL		
6	*D	*D	$73 + 10 + j$ = $83 + j$
7	7		
8	3		
9	+		
010	*LBL		
1	B	B	$10 + j$ = $10 + j$
2	1		
3	*LBL		
4	A	A	$0 + j$ = j
5	0		
6	+		
7	RCL		
8	0		
9	0		
020	INV		
1	*IFZRO	If $j \neq 0$	
2	0		
3	2		
4	7		
5	+		
6	1		
027	=		
8	STO		
9	9		
030	9		
031	*RTN	STOP	

(j assumed to be needed for other purposes; hence, INV*DSZ features available on calculator not used.)

Similarly, if j=0:

C $7 + 73 + 10 + 0 + 1$ = 91

D $73 + 10 + 0 + 1$ = 84

B $10 + 0 + 1$ = 11

A $0 + 1$ = 1

If the above four labels are each individually written, 77 (vice 31) steps are required as listed below. (Assume Program Begins at Step 001.)

*LBL   C : 90 + RCL 00 , INV * IFZRO 015,+1= , STO 99, *RTN.

*LBL *D' : 83 + RCL 00 , INV * IFZRO 036,+1= , STO 99, *RTN.

*LBL   B : 10 + RCL 00 , INV * IFZRO 056,+1= , STO 99, *RTN.

*LBL   A :   RCL 00 , INV * IFZRO 074,+1= , STO 99, *RTN.

If the common steps are combined into an unlabeled subroutine (beginning at step 035) 50 steps are required, i.e.,:

*LBL   C : 90 , SBR 035 , *RTN.

*LBL *D' : 83 , SBR 035 , *RTN.

*LBL   B : 10 , SBR 035 , *RTN.

*LBL   A :   SBR 035 , *RTN.

Subroutine: + RCL 00 , INV * IFZRO 046,+1= , STO 99 , *RTN.

If the above subroutine is labeled A and used as a terminus vice as a subroutine 39 steps are required, i.e.:

*LBL   C : 90 , + , GTO A

*LBL *D' : 83 , + , GTO A

*LBL   B : 10 , + , GTO A

*LBL   A : RCL 00 , INV * IFZRO 034,+1= , STO 99 , *RTN.

Clearly, the most efficient use of memory space is demonstrated by the initial nesting method. Such nesting can also be shown to be the most efficient method on RPN calculators. Nesting always saves steps on either type of system simply because nesting completely avoids using any calls whatsoever for any subroutine properly sequenced in the nest, and also avoids repeating identical instructions within several subroutines. For many examples of complex nesting with RPN, see Figure 4. This program has 19 labels but only 8 RTN instructions; Labels 1, 2, 3, 4, 5, 8, and E are nested under other labels. In the comparable SR-52 program, Figure 3, labels *E', E, and A are nested under other labels.

Stacking is the process of minimizing program execution time by avoiding lengthy label searches, rather than a process designed to minimize program storage space. Stacking costs steps on the SR-52, but does not cost steps on the HP-67. On the SR-52, labels are stacked followed by GTO xyz statements as discussed in Section IV.E.(10)a above. On the HP-67, entire subroutines are stacked immediately after the subroutine(s) which call the stacked subroutines most often, in order of expected call frequencies.

Returning to Figure 4, Labels (Subroutines) 6, 7, 8, and 9 are stacked in order of expected frequency to accomplish such minimum average label-search times on the HP-67 during IRR calculations. Stacked subroutines may sometimes also be nested, as is Subroutine 8 in Figure 4.

d. Appropriating Program Steps on Registers
    for Data Memory

The SR-52 has 22 memory registers nominally available: Registers 0 through 19, 98 and 99. (Some of the Registers 60 through 69 may also be available, depending upon the number of pending operations being stored.) An additional 18 Registers (80 through 97) may be "purchased" by trading eight steps of program memory per extra register desired, working backwards from the end of the program. Thus trading of steps 216 through 223 "buys" register 97 and trading of steps 208 through 215 buys register 96, etc., down to the purchase of register 80 with steps 80 through 87. (This technique is demonstrated in Figure 2 to create two extra registers.) In this manner, a program which can be limited to 80 steps (000 through 079) can use at least 40 storage registers on the SR-52; a program limited to 112 steps (one side of a card) can use at least 36 registers, including 0 through 19 and 84 through 99.

The availability of so many storage registers allows the user to approach arrayed data with straight-forward indexing techniques. For example, using the latter allocation above, some of the 36 available registers may be arbitrarily assigned to the following array (two-digit numbers equate to register address numbers):

Column No.:	1	2	3	4	5	6	7
ROW *D' :	84	85	86	87	88	89	90
ROW A :	01	02	03	04	05	06	07
ROW B :	11	12	13	14	15	16	17
ROW C :	91	92	93	94	95	96	97

Combining this array with the optimized nested routines of subsection (c) above then allows programmed indexing by column and row number. For example, if 3 (the column index) happens to be stored in REG 00 when LBL B (the row index) happens to be called, the nested program routines will generate the number 13 (10 + j = 10 + 3 = 13), the same number located in column 3, Row B of the array. A subsequent call for LBL C would generate 93, the number located in Column 3, Row C, etc. Thus, each register number listed in Column 3 could be generated, in succession, by the subroutine: 3 STO 00 , *D' , A , B , C . Similarly, if the number stored in the column indexer (REG 00) happens to next be changed from 3 to 4, a call for LBL A would generate 04, the number located in Column 4, Row A, etc.

Note, within the nested programs introduced earlier, that each call, in addition to generating the desired number, actually stores the generated number in REG 99. Thus REG 99 can be used as an addresser for the arrayed registers, in each case, by some other portion of the program. Additionally, the column number can

automatically incremented in REG 00 by still another portion of the program. In this manner, each of the values actually stored in the registers listed in a particular row of the array may be sequentially recalled and used by the program. Similarly, each register listed in any column of the array may be sequentially called prior to again changing the column indexer. These row and column call techniques and the above array are used in Subsection (h) below to facilitate a linear-programming problem, but one equally applicable to any other problems which must address a four by eight array of data.

A simple example of appropriated SR-52 program steps is shown in Figure 2, where only two extra registers were required. In summary, the user should always remember that useful extra registers are available on the SR-52 anytime the applicable program steps are not used.

An alternate method, packing registers in order to store two values per register, can be used on the HP-67 to create an additional register (or registers). This method is discussed in Section IV.E (10) h. below. The only limitation of this method appears to be that only 10 digits may be handled between both numbers, uniformly split as desired, e.g., 5/5 or 6/4, etc.

A different, but equally unique, feature of the HP-67 is its ability to store many "words" in its storage registers. Any word formed with the letters a, B, C, d, E, g, I, l, O, o, r, S, Y and Z may be normally, but not always

be created. For example, the following 11-space word
groups (including mandatory period) can be stored in
registers (not in program memory)

YES SIr .	IdEa rEadY.
I dEClarE .	I do SaY .
I IS rEadY .	I do AgrEE.
BOSS SaY .	o o o o .
good IdEaS .	gollY gee .
I do EaSY .	IdEa CraZY.
god BlESS .	alSO SOrrY.
go BaBY go .	I dISagrEE.

Cards containing 32 phrases may be obtained via the HP-65
Users Club [19] by club members, or from club members who
have already obtained them; there is no copyright involved.
There is no other published source at this time.

Only extremely advanced programmers can create
words; the process involves understanding design architecture
of the machine. Conversely, the words are transferred to the
storage registers by merely reading the data card, once
obtained.

e. Structuring Loop Controls and Counters

Loops are normally structured to begin with a
given value and iteratively change that value until some
specified limit is reached, at which point the program will
exit from the loop. The given value may be either the

variable which is to be worked upon inside the loop or a simple (additional) counter which is incremented or decremented by the program during each pass through the loop. An example of the former (no counter), extracted from the IRR programs of Figures 3 and 4, is duplicated as Figure 5 below. In this example, the program begins with a given value of 0.02 (2%) as an increment for the trial interest rate supplied by the user. The increment is added to the trial interest rate and the program falls into the first loop, where it is immediately sent elsewhere to compute the net present value (NPV) of the Investment and cash flows stored. Upon returning, the newly calculated NPV is tested to determine if it is positive — if so, the program loops back to add the increment again and repeat the loop process. This continues until the NPV goes negative, at which point the program falls out of the first loop, divides the increment by minus two creating a decrement, and falls into the second loop. This loop cycles similarly until the NPV goes positive again, at which point the program falls out of the second loop, changes the decrement back in to the previous increment by multiplying by minus two, and tests to see if the increment has yet been reduced to the specified limit (0.0002). If not, the increment is divided by ten to move the decimal one place and the program loops back to the beginning of the first loop. (The HP-67 program changes the increment before the limit test and thus tests against 0.00002 instead.) The entire process above then repeats. When the SR-52 program

FIGURE 6
LOOP CONTROL BY SIGN OF VARIABLE

SR-52 IRR SUBROUTINE

STEP	KEY ENTRY
127	.
8	0
9	2
130	STO
1	9
2	6
3	*C'
4	*IF POS
5	1
6	3
7	3
8	2
9	+/-
140	INV
1	*PROD
2	9
3	6
4	*C'
5	INV
6	*IF POS
7	1
8	4
9	4
150	.
1	2
2	+/-
3	*PROD
4	9
5	6
6	RCL
7	9
8	6
9	-
160	2
1	EE
2	4
3	+/-
4	=
5	INV
6	EE
7	*IF POS
8	1
9	3
170	3
1	RCL
2	9
3	8
4	INV
5	*FIX
176	*RTN

IF (i - .0002) IS ZERO OR POSITIVE

HP-67 IRR SUBROUTINE

STEP	KEY ENTRY
114	.
5	0
6	2
7	f LBL 5
8	STO C
9	RCL C
120	g GSB e
1	7
2	CHS
3	h ST I
4	h R↓
5	f X>0
6	GTO (i)
7	RCL C
8	2
9	÷
130	CHS
1	g GSB e
2	1
3	1
4	CHS
5	h ST I
6	h R↓
7	f x<0
8	GTO (i)
9	GTO 7
////////////	
184	f LBL 7
5	2
6	EEX
7	5
8	CHS
9	RCL C
190	.
1	1
2	X
3	g x>y
4	GTO 5
5	f LBL 8
6	RCL B
7	h ST I
8	DSP 4
9	RCL D
200	h RTN

IF NPV IS POSITIVE

IF NPV IS NEGATIVE

IF Δi > .00002

For details, See Figures 3 and 4

has completed the inner loops using the specified limit as the increment, or when the HP-67 has completed the inner loops using ten times the specified limit for the increment, the program stops with the IRR in the display.

The loop control method discussed above, which is actually a modified bisection search pattern using defined vice variable increments, is useful anytime the programmer needs to structure loops based upon a sign change of the variable being addressed within the program. This method also allows limiting accuracy to a specified number of decimal places to avoid unnecessary iterations. Differences between the machines for these loops are the order of increment division and test as already mentioned and the loop-back commands themselves. The SR-52 requires programming the step number of the beginning of the loop immediately below each sign test; the HP-67 requires either programming (prior to the sign test) the relative (minus) number of steps to get back to the beginning of the loop from the GTO (i) instruction which follows the sign test, or the programming of extra labels which can be used as addresses after the sign tests. Extra labels can also be used on the SR-52, but absolute numeric addresses are executed quicker. (On the HP-67, the relative (minus-step) addresses are quicker than label addresses for short distances.)

Both calculators have a register (the zero register) which can be automatically decremented by one and tested for zero; the program counter will skip one

address after the test if the register value has reached zero (the DSZ function). The HP-67 also has an ISZ function, and can also branch to an instruction, vice an address, in either case, if desired. (The user of the SR-52 can only increment negative numbers, and can only branch to an address.) The HP-67 can also use the DSZ or ISZ function against the value in the display; the SR-52 cannot.

Any storage register may be used as a counter and decremented or incremented by the program itself on either calculator, but this method is only used when the zero register is needed to store other data such that the automatic DSZ or ISZ cannot be used. Of course, this use of other registers as counters also requires that the stored value be recalled and tested by the program at the end of each loop iteration, similar to the process in Figure 5. Additional details are available in the manufacturer's handbooks.

f. Multiple (Alternate) Uses of One Subroutine

One example of multiple uses of one subroutine was provided in subsection (c) above, using nested labels. Another example is at step 104 of Figure 4, which is duplicated below:

Step	Key Entry	Explanation
101	g LBL c	Begins "Do Profit" Routine
102	0	
103	STO D	Sets Trial i = 0
104	f GSB E	Run NPV routine
105	f x > 0	If Profit > 0, Stop
106	h RTN	

128

This routine uses the fact that the NPV of a series of
cash flows at an interest rate of zero is the same as the
undiscounted profit margin. Hence, instead of writing a
long separate routine to calculate the profit margin, the
authors merely set i = 0 and branch to the NPV routine
from the short routine above. This routine continues:

107	f LBL D	Begins "Do TRIAL i" Routine
8	f x< 0	If TRIAL i < 0
9	GTO 0	DSP ERR (-TRIAL i)
110	STO D	OW, STO TRIAL i in REG D
1	DSP 4	FORMAT 4 DEC DSP
112	h RTN	STOP.(DSP TRIAL i)

During the breakeven point routine, not shown above, exe-
cution is branched to step 101 if a situation arises where
all cash flows have been subtracted from the investment but
the balance is still negative. In this case, the profit
routine (which, in turn, uses the NPV routine) is used to
recalculate the negative profit. This value then falls
through step 107 to hit step 108, a step actually designed
to catch negative input errors by the user for the trial
interest rate. In this case, however, the value tested is
the negative profit, and the program branches to 0, a non-
existent label in this program. This generates an error
condition and halts program execution with "Error" in the
display. If the user then presses any key on the keyboard,
the key will not execute. Instead, the negative profit,
with minus sign, will appear in the display. Thus the

129

routine has a third use — calculating negative profits. Of course, the negative profit was already in the stack when the program branched to step 101, so why was it recalculated? Simply because getting it out of the stack, halting execution, and displaying the error would require considerable program steps which are not available. Instead, then, the single step (#083, GTO f c), accomplished all that was required, since the steps 101 through 112 already existed for other purposes. Similar multiple use of subroutines can be designed into most programs with a little thought.

g. Error or User-Prompting Routines

In addition to the above example, there are four other places in Figure 4 where the program is ordered to branch to the unused label zero, creating an error condition. In each case, the value which will be displayed when the user pushes any key to clear the error display provides information concerning what error was made, only because this information is built into the program structure:

Step	Info Built into program	Error Which will be Displayed
010	None; User's error returned	Non-integer Period Number
016	None; User's error returned	Period No. Greater Than 20
022	None; User's error returned	Negative Period Number
042	21	21 (User attempted to store more than 20 Cash Flows)
109	None; User's error returned	Negative interest rate
109	Profit Routine as above	Negative profit margin for Break-even routine

130

Instead of generating an error, the SR-52 program (Figure 3) avoids negative inputs by taking the absolute value of inputs which can be incorrectly entered as negative. This is better than no protection, but program space in the IRR routine does not allow warnings similar to the above. In other programs with more space, similar devices could be used. The SR-52 flashes the error value automatically, instead of flashing "Error" once and halting with error in the display as does the HP-67. Either method is useful.

User prompting routines are more difficult to structure on the SR-52 since (unlike the HP-67) the SR-52 will not automatically halt in execution, flash a value, or accept input, and then automatically continue. Conversely, for the HP-67 IRR program (Figure 4), whenever the user decides to shift to another period number and change the cash flow for that period, and thus use key b (Set n = DSP(x)), the calculator announces the maximum period number already used (flashes it four times) and then displays the requested period number. This is designed to say to the user: "OK, you have N values in here now, now go change the one you asked for." Similarly, when the user enters the cash flow for the period requested, the HP-67 announces the period number which the cash flow was stored under (flashes it four times), the amount stored (flashes it once) and halts displaying the maximum period number which has a cash flow stored against it, after the change. This is designed to say to the user: "OK, you just changed the cash flow for

the nth period.... to the amount of $.... and you now have N values stored in the machine." (If the value zero is stored against the maximum period, the HP-67 program automatically searches out and displays the period number for the highest non-zero cash flow stored, as the third part of the above sequence.)

On the SR-52, in lieu of the above, the IRR program (Figure 3) provides a user key (C) that can be pressed to display the maximum N stored, prior to using the "Set n = DSP(x)" key (B'). Then, when the user enters the desired period number and presses the B' Key, the display responds with the desired period number minus one. Note that during automatic accumulation of cash flows using the "STO CF's" user-defined label (see Figure 3 or 4) on either machine, the period number is displayed for each cash flow stored in sequence. Thus the user is accustomed to having the previous period number in display immediately before each new cash flow is entered. The SR-52 IRR routine thus capitalizes on this familiarity when the period number is abruptly changed by the user via the "Set n = DSP(x)" key. This prompting is designed to say to the user: "OK, I (the machine) have switched to the customary cash flow accumulation mode, thus the period prior to the one you requrest — and that you will next enter a cash flow into — is in my display; now enter the desired cash flow." The user then enters the desired cash flow and presses the "STO CF's" key

This time, the calculator responds with the number of the period used (as originally requested by the user in the previous routine). The user can then check the total number of cash flows stored, if desired, by pressing C. Since the SR-52 program does not (for lack of program space only) automatically reset N if 0 is stored in N, this check will alert the user to reset N, if required, by pressing *B, *RSET. (See Section IV.E.(9)c(3).) Clearly, without the minus one routine discussed above, the user would be faced with the same period number both before and after entering the cash flow, which could more easily lead to user's forgetting which operation was just accomplished, especially if the user is interrupted during this process. Conversely, the two routines are tied together by the display itself with the above method. Similar display prompts can be designed into most programs.

Finally, the HP-67 use of "words" in storage registers (see Section IV.E.(10)d.) can also be used for user prompting at end of executing or user prompting of intermediate errors. Setting this process up, however, is somewhat difficult, because the user must use both a data card which contains the word-phrases and the program card prior to commencing operations. Additionally, one storage register per phrase desired must be available and otherwise unused by the program at any time, and the program must be structured to have the result (or error), normally to be displayed after the words, stored in the proper STACK REG.

The process will then work similar to the HP-67 flashing displays above. Thus a three phrase message might be transferred into the STACK in the event of an error such that the message, "IdEa CraZY. alSO SOrrY. I dISagrEE. -123.4567890" is flashed at the user for the result (or error) valued -123.4567890. If the sequence of desired phrases is longer, register review vice STACK review can be used similarly, except that all intervening values stored in the registers 1 through 9 and 20 through 25 would also be flashed. Alternately, a sequence which individually recalls each phrase, pauses to display it, and then repeats this cycle for the next phrase may also be programmed.

h. Multiple Card Operations

Either calculator may be programmed to read additional program or data cards in operation. Since this process is somewhat more difficult with the SR-52, the SR-52 will be used in this section in all examples. The HP-67 process is simpler to program because the HP-67 "smart" card reader does not require that the programs be structured to ensure that read instructions on one card appear in the same step number as do halt instructions on the next card. The HP-67 will also automatically distinguish between program and data, and will automatically reset all flags and the display mode as specified on the new card. Once programmed, however, the systems work similarly: A program card is read into the calculator and the user begins using

the user defined keys for entry of variables and/or to run
calculations. Once calculations have begun, the user slips
the leading edge of a second card into the card reader.
This card will not be read until the program reaches a
point specified in the program, at which time the second
card is pulled through the card reader, modifying the program
or supplying additional data, and execution continues. As
an example, the authors designed a program which can be used
by students being introduced to basic linear programming
methods (Figure 6). The program will optimize an objective
function subject to three equations containing seven variables
each. The example below is subject to only two equations
of five variables each, but is solved the same way. Thus,
as an example, suppose the student is asked to:

Maximize: $2x_1 + 5x_2 + 7x_3$

Subject to: $3x_1 + x_2 + 2x_3 + x_4 + 0x_5 = 150$

$\qquad\qquad\qquad x_1 + 3x_2 + 4x_3 + 0x_4 + x_5 = 250$

(An explanation of linear programming is beyond the scope of
this work [18]. Thus the following analysis is most useful
to those readers already familiar with the technique.
Remember, however, that the method of this example is a
simplistic method for beginning students, not the most
advanced method that can be programmed.)

The student is given the following instructions:

(1) Construct an expanded Simplex Tableau as shown below.
(2) Compute the $(c_j - z_j)$ row.
(3) Mark the maximum value in the $(c_j - z_j)$ row with one asterisk (*).
(4) Compute the ratio of each constraint to the value in the pivot (*) column.
(5) Mark the minimum ratio obtained with two asterisks (**) to define the pivot row.
(6) Mark the intersection of the pivot row and pivot column with the symbol (@) to define the pivot element.
(7) Multiply the pivot element, and every other element in the pivot row, by the reciprocal of the pivot element; enter this new row in the next section of the tableau.
(8) Multiply the new row by whatever factor is required such that when the multiplied row is added to a remaining row, in the previous tableau, the element in the pivot column becomes zero; bring down this zero (∅) and the resulting sum for each other element in the same previous row into the new tableau.
(9) Continue with new tableaus until all elements in the $(c_j - z_j)$ row become zero or negative or some other informational condition is reached.

If required to work the example problem with the above instructions, the result might look like this:

COST	ROW	VARIABLE	x_1	x_2	x_3	x_4	x_5	x_6	
			2	5	7	0	0	P	
$C_4 = 0$	A	x_4	3	1	2	1	0	150	
$C_5 = 0$	B	x_5	1	3	4@	0	1	250	**
		$(c_j - z_j)$	2	5	7*	0	0	(0)	
$C_4 = 0$	A	x_4	2.5@	-0.5	∅	1	-0.5	25	**
$C_5^4 = 7$	B	x_3	0.25	0.75	1	0	0.25	62.5	
		$(c_j - z_j)$	0.25*	-0.25	0	0	-1.75	(437.5)	
$C_1 = 2$	A	x_1	1	-0.2	0	0.4	-0.2	10	
$C_3^1 = 7$	B	x_3	∅	0.8	1	-0.1	0.3	60	
		$(c_j - z_j)$	0	-0.2	0	-0.1	-1.7	(440)	

OPTIMAL.

136

As a teaching aid to the above process, and to avoid mathematical errors, the Program in Figure 6 can be used by the student, in lieu of pencil and paper algorithms, in the exact manner shown above. (This program does the computations one line at a time the same way the student would. The student must only learn to operate the program properly and to relate to the matrix developed in Section IV.E.(10)d., as reproduced below; the student must compute ratios and pick his own pivots in the usual manner — hence these steps do not appear below.)

Column Number:	1	2	3	4	5	6	7	8
Objective Row: *D'	84	85	86	87	88	89	90	
ROW A:	01	02	03	04	05	06	07	08
ROW B:	11	12	13	14	15	16	17	18
ROW C:	91	92	93	94	95	96	97	98

Although this matrix differs from the standard matrix typically learned for the Simplex method (a_{11},a_{12},etc.) it is similar and rather easily learned by the student; the subscripts normally learned do not happen to all be available as register numbers on the SR-52.

The process to use this two card program (one side programmed per card) is:

Explanation	Sample Entry	Press	Display

Read Card 1   *READ

Set up OBJECTIVE ROW   *D'   84.
Set up row counter   D   84.
Enter OBJECTIVE ROW values (Column 1)   2   RUN   84.
                           (Column 2)   5   RUN   85.
                           (Column 3)   7   RUN   86.

Set up ROW A   A   1.
Set up row counter   D   1.
Enter ROW A values (Column 1 term)   3   RUN   1.
                   (Column 2 term)   1   RUN   2.
                   (Column 3 term)   2   RUN   3.
                   (Column 4 term)   1   RUN   4.
                   (Column 5 term)   0   RUN   5.
                   (Column 6 term)   150   RUN   6.

Set up ROW B   B   11.
Set up row counter   D   11.
Enter ROW B values (Column 1 term)   1   RUN   11.
                   (Column 2 term)   3   RUN   12.
                   (Column 3 term)   4   RUN   13.
                   (Column 4 term)   0   RUN   14.
                   (Column 5 term)   1   RUN   15.
                   (Column 6 term)   250   RUN   16.
(Same process for ROW C when required)

Set up for $(c_j - z_j)$ calculations   *RSET   (16.)
Enter highest column number used   6   E   0.
    (Values for each)   (5th)   RUN   0.
    (successively)   (4th)   RUN   0.
    (smaller column)   (3rd)   RUN   7.
    (compute in   )   (2nd)   RUN   5.
    (sequence   )   (1st)   RUN   2.

Alternately, or to recheck a value,
merely enter column number: 3   E   7.
                                              1   E   2.
                                            etc.

Read Card 2   *E'   ---
                                                               HLT   (2.)

Multiply ROW B by (1/4 = .25)
    and Relist ROW B.   .25   D   0.25
                                                               B   11.
    (Column 1 term)   RUN   0.25
    (Column 2 term)   RUN   0.75
    (Column 3 term)   RUN   1.
    (Column 4 term)   RUN   0.
    (Column 5 term)   RUN   0.25
    (Column 6 term)   RUN   62.5

Explanation	Sample Entry	Press	Display
Multiply new ROW B by -2 and	-2.	E	-2.
add to previous ROW A;		B	11.
list new ROW A		A	1.
(Column 1 term)		RUN	2.5
(Column 2 term)		RUN	-0.5
(Column 3 term)		RUN	0.
(Column 4 term)		RUN	1.
(Column 5 term)		RUN	-0.5
(Column 6 term)		RUN	25.
Reread Card 1 and Reset Counters		*E'	---
		HLT	(25.)
		*RSET	(25.)
Set up Cost Routine		*C'	(25.)
Enter Cost of ROW A	0	RUN	0.
Enter Cost of ROW B	7	RUN	7.
(Same process for ROW C when required)			
Set up for $(c_j - z_j)$ calculations		*RSET	(7.)
Enter highest column number used and			
compute as in previous tableau:	6	E	-437.5
(Column 5 term)		RUN	-1.75
(Column 4 term)		RUN	0.
(Column 3 term)		RUN	0.
(Column 2 term)		RUN	-0.25
(Column 1 term)		RUN	0.25
Reread Card 2		*E'	---
		HLT	(0.25)
Multiply ROW A by $(1/2.4 = .4)$ and			
relist	.4	D	0.4
		A	1.
(Column 1 term)		RUN	1.
(Column 2 term)		RUN	-0.2
(Column 3 term)		RUN	0.
(Column 4 term)		RUN	0.4
(Column 5 term)		RUN	-0.2
(Column 6 term)		RUN	10.
Multiply new ROW A by -1/4 and add to			
previous ROW B; list new ROW B	-.25	E	-0.25
		A	1.
		B	11.
(Column 1 term)		RUN	0.
(Column 2 term)		RUN	0.8
(Column 3 term)		RUN	1.
(Column 4 term)		RUN	-0.1
(Column 5 term)		RUN	0.3
(Column 6 term)		RUN	60.

Explanation	Sample Entry	Press	Display
Reread Card 1 and Reset Counters		*E'	---
		HLT	(60.)
		*RSET	(60.)
Set up Cost Routine		*C'	(60.)
Enter Cost of Row A	2	RUN	2.
Enter Cost of Row B	7	RUN	7.
Set up for ($c_j - z_j$) calculations		*RSET	(7.)
Enter highest column number used and			
compute as in previous tableaus	6	E	-440.
(Column 5 term)		RUN	-1.7
(Column 4 term)		RUN	-0.1
(Column 3 term)		RUN	0.
(Column 2 term)		RUN	-0.2
(Column 1 term)		RUN	0.
Problem complete. Recall maximum:	6	E	-440.

Result is $440.00.

If the reader is wondering why this program uses
two cards but only one side per card, the answer is that
only the first 112 program steps are used as program. The
remainder are used to create the storage registers 84
through 97. Of course, the two 112 step programs could be
put on alternate sides of the same card anyway, but since
the two programs use different user-defined labels, the use
of one card for both purposes would be confusing.

On the HP-67, the necessary number of data
registers can only be created by packing (storing values on
opposite sides of the decimal) at least 20 of the 26 available
registers. This limits inputs to values having no more than
5 significant digits, which is quite satisfactory for the
above example. This entire program can, also, be listed on
only one HP-67 card. Further, it is somewhat simpler to use

than the above SR-52 method. For interested readers, such
a program is provided at Figure 7.

In a manner almost identical to the above,
programs can be created which do not require secondary input
from the user, other than start commands after initial input.
Programs can even be created on loops of magnetic mylar
instead of cards, loops which require only occasional starts
by the user.

i. Program Space Versus Execution Time

In programs which compute solutions based upon
non-iterative algorithms, or with minimal use of iterative
(loop) algorithms, execution speed is not normally a
consideration because SR-52 and HP-67 can both execute an
average program of 224 instructions in less than 15 seconds
(unless numerous exponentiations are required in the case
of the HP-67). However, when numerous iterations of
algorithmic loops are necessary, execution time becomes a
critical factor. For example, note that the IRR examples
discussed in Sections IV.E.(9)c(7) and (8) require 4 minutes
46 seconds on the SR-52 and 7 minutes 20 seconds on the
HP-67. If these programs were not optimized for execution
speed, execution could require more than two times these
respective amounts of time. Extremely long program execution
times are both inconvenient to the user and unnecessarily
wasteful of battery power. Hence complex or iterative main
routines should normally be designed to minimize execution
time.

FIGURE 7
LINEAR PROGRAMMING AID (SR-52)

USER KEYED LABELS					
	B' (USED)	C' INIT. COST: A,B,C	D' SELECT OBJ. ROW	E' READ CARD 2	
A SELECT ROW A	B SELECT ROW B	C SELECT ROW C	D a_{ij} REG$_{ij}$	E $j \to (c_j - z_j)$	

FLAGS: OFF/ON

0		X		
1		X		
2		X		
3		X		
		X		

REGISTERS

Reg	Content		Other Labels
00	j Index	(COL)	
01	a SUS. A1		
02	a SUS. A2		
03	a SUS. A3		
04	a SUS. A4		
05	a SUS. A5		
06	a SUS. A6		
07	a SUS. A7		
08	b SUS. A8		
09	Row A COST		
10	Row B COST		
11	a SUS. B1		
12	a SUS. B2		
13	a SUS. B3		
14	a SUS. B4		
15	a SUS. B5		
16	a SUS. B6		
17	a SUS. B7		
18	b SUS. B8		
19	Row C COST		
91	a SUS. C1		
92	a SUS. C2		
93	a SUS. C3		
94	a SUS. C4		
95	a SUS. C5		
96	a SUS. C6		
97	a SUS. C7		
98	b SUS. C8		
99	a SUS. ij		

OTHER LABELS

84	COEF. x SUS. 1
85	COEF. x SUS. 2
86	COEF. x SUS. 3
87	COEF. x SUS. 4
88	COEF. x SUS. 5
89	COEF. x SUS. 6
90	COEF. x SUS. 7
68	P SUS. j
69	$(c_j - z_j)$

NOTES

SUS. = Subscript

COEF. = Coefficient

TO INITIALIZE:

PRESS HLT*RSET.

INSTRUCTIONS IN THESIS.

SET STATUS

DISPLAY   TRIG

☐ SCI		X DEG	
☐ ENG		☐ RAD	
☐ FIX		☐ GRAD	
X FLOATING PT			

STEP	KEY CODE	KEY ENTRY	
000	10	*E'	
1	46	*LBL	
2	13	C	SELECT ROW C
3	07	7	
4	85	+	
5	46	*LBL	
6	19	*D'	SELECT OBJECTIVE ROW
7	07	7	
8	03	3	$(D_j = 83 + j \to REG\ 99)$
9	85	+	
010	46	*LBL	SELECT ROW B
1	12	B	
2	01	1	$(B_j = 10 + j \to REG\ 99)$
3	46	*LBL	SELECT ROW A
4	11	A	
5	00	Ø	$(A_j = Ø + j \to REG\ 99)$
6	85	+	
7	43	RCL	
8	00	Ø	
9	00	Ø	
020	22	INV	
1	90	*IFZRO	IF j ≠ 0
2	00	Ø	
3	02	2	
4	07	7	GTO STEP 027
5	85	+	IF j = 0, As Above,
6	01	1	$A_j = 1$
027	95	=	$B_j = 11$
8	42	STO	$C_j = 91$ $\to$ REG 99
9	09	9	$D_j = 84$

142

STEP	KEY CODE	KEY ENTRY	NOTES
030	09	9	
1	56	*RTN	
2	46	*LBL	$j \rightarrow (C_j - Z_j)$
3	15	E	
4	42	STO	
5	00	Ø	
6	00	Ø	
037	19	*D'	⊛D'
8	25	CLR	
9	01	1	
040	34	+/-	
1	17	*B'	⊛B'
2	11	A	Ⓐ
3	43	RCL	
4	00	Ø	
5	09	9	
6	17	*B'	⊛B'
7	12	B	Ⓑ
8	43	RCL	
9	01	1	
050	00	Ø	
1	17	*B'	⊛B'
2	13	C	Ⓒ
3	43	RCL	
4	01	1	
5	09	9	
6	46	*LBL	(USED)
7	17	*B'	
8	65	X	
9	36	*IND	
060	43	RCL	
1	09	9	
2	09	9	
3	95	=	
4	22	INV	
5	44	SUM	
6	06	6	
7	09	9	
8	43	RCL	
9	06	6	
070	09	9	
1	56	*RTN	STOP.
2	58	DSZ	
3	00	Ø	
4	03	3	
5	07	7	IF j ≠ 0, GTO STEP 037
076	56	*RTN	
///7	CONTINUED		
///8	ON NEXT		
///9	PAGE		

USER PRESS RUN

143

STEP	KEY CODE	KEY ENTRY	NOTES
///0			
///1			
///2			
///3			
///4			
///5			
///6			
077	46	*LBL	$a_{ij} \rightarrow REG_{ij}$
8	14	D	
9	81	HLT	STOP.
080	36	*IND	
1	42	STO	
2	09	9	
3	09	9	
4	43	RCL	
5	09	9	
6	09	9	
7	85	+	
8	01	1	
9	85	=	
090	48	*EXC	
1	09	9	
2	09	9	
3	14	D	
4	46	*LBL	INIT. COST: A,B,C
5	18	*C'	
6	81	HLT	STOP.
7	42	STO	
8	00	0	
9	09	9	
100	81	HLT	STOP.
1	42	STO	
2	01	1	
3	00	0	
4	81	HLT	STOP.
5	42	STO	
6	01	1	
7	09	9	
8	56	*RTN	
9	46	*LBL	READ CARD 2
110	10	*E'	
1	96	*READ	END OF CARD 1.
///2	SEE		
///3	NEXT		STEPS 112-223
///4	PAGE		USED AS STORAGE
///5	FOR		REGISTERS 84-97.
///6	CARD		
///7	NUMBER		
///8	TWO		
///9			

Row labels (left margin):
- PRGM OR USER GIVE a_{ij} — steps 077–079
- USER GIVE: ROW A COST — steps 094–096
- ROW B COST — steps 099–100
- ROW C COST — steps 103–104

144

| A | SELECT ROW A | B | SELECT ROW B | C' STO K REG ØØ / C SELECT ROW C | D | K·PR / K·PR IN PR | E' REREAD CARD 1 / E K PR ROW / K·PR SUM IN ROW |

FLAGS: OFF/ON

0		X	
1	IF		X
2	IF		X
3		X	
		X	

REGISTERS

00	j INDEX
01	a SUS. A1
02	a SUS. A2
03	a SUS. A3
04	a SUS. A4
05	a SUS. A5
06	a SUS. A6
07	a SUS. A7
08	b SUS. A8
09	ROW A COST
10	ROW B COST
11	a SUS. B1
12	a SUS. B2
13	a SUS. B3
14	a SUS. B4
15	a SUS. B5
16	a SUS. B6
17	a SUS. B7
18	b SUS. B8
19	ROW C COST
91	a SUS. C1
92	a SUS. C2
93	a SUS. C3
94	a SUS. C4
95	a SUS. C5
96	a SUS. C6
97	a SUS. C7
98	b SUS. C8
99	a SUS. ij

(COL)

USER GIVE K

USER GIVE P ROW

SET STATUS

DISPLAY   TRIG

	SCI			X	DEG	
	ENG				RAD	
	FIX				GRAD	
X	FLOATING PT					

OTHER REGISTERS

84	COEF. x SUS. 1
85	COEF. x SUS. 2
86	COEF. x SUS. 3
87	COEF. x SUS. 4
88	COEF. x SUS. 5
89	COEF. x SUS. 6
90	COEF. x SUS. 7
68	P SUS. j
69	$(C_j - Z_j)$

STEP	KEY CODE	KEY ENTRY
000	96	*READ
1	46	*LBL
2	18	*C'
3	42	STO
4	00	Ø
5	00	Ø
6	56	*RTN
7	46	*LBL
8	14	D
9	18	*C'
010	81	HLT
1	46	*LBL
2	13	C
3	08	8
4	00	0
5	85	+
6	46	*LBL
7	12	B
8	01	1
9	46	*LBL
020	11	A
1	01	1
2	95	+
3	42	STO
4	09	9
5	09	9
6	60	*IF FLG
7	01	1
8	00	Ø
9	06	6

NOTES

SUS. = SUBSCRIPT
COEF. = COEFFICIENT
PR   = PIVOT ROW

TO INITIALIZE:

press hlt.

STO K   REG ØØ

K PR
K TIMES PR IN PR

SELECT ROW C
(Set ij = 91+K)

SELECT ROW B
(Set ij = 11+K)

SELECT ROW A
(Set ij = 1+K)

STEP	KEY CODE	KEY ENTRY	NOTES
030	07	7	IF FLG 1 SET, GTO STEP 067
1	60	*IF FLG	
2	02	2	
3	00	Ø	
4	07	7	
5	05	5	IF FLG 2 SET, GTO STEP 075
6	56	*RTN	
7	43	RCL	
8	00	Ø	
9	00	Ø	
040	36	*IND	
1	49	*PROD	
2	09	9	
3	09	9	
4	36	*IND	
5	43	RCL	
6	09	9	
7	09	9	
8	81	HLT	
9	01	1	
050	44	SUM	
1	09	9	
2	09	9	
3	41	GTO	
4	00	Ø	
5	03	3	
6	07	7	
7	56	*RTN	STOP OR RTN TO CALLING SBR
8	46	*LBL	READ CARD 2
9	10	*E'	
0	86	*RSET	
1	46	*LBL	K → PR → ROW
2	15	E	K TIMES PR SUM IN ROW
3	18	*C'	
4	50	*ST FLG	
5	01	1	
6	81	HLT	
067	22	INV	
8	50	*ST FLG	
9	01	1	
070	42	STO	
1	06	6	
2	08	8	
3	50	*ST FLG	
4	02	2	
075	81	HLT	
///6	CONTINUED		
///7	ON		
///8	NEXT		
///9	PAGE		

Left margin labels:
- USER PRESS RUN
- PRGM OR USER CALL
- USER GIVE K
- USER GIVE P ROW
- PRGM OR USER GIVE ROW

STEP	KEY CODE	KEY ENTRY	NOTES
///0			
///1			
///2			
///3			
///4			
///5			
076	22	INV	
7	50	*ST FLG	
8	02	2	
079	43	RCL	
080	00	Ø	
1	00	Ø	
2	65	X	
3	36	IND	
4	43	RCL	
5	06	6	
6	08	8	
7	95	=	
8	36	*IND	
9	44	SUM	
090	09	9	
1	09	9	
2	36	*IND	
3	43	RCL	
4	09	9	
5	09	9	
6	81	HLT	STOP.
7	01	1	
8	44	SUM	
9	09	9	
100	09	9	
1	44	SUM	
2	06	6	
3	08	8	
4	41	STO	
5	00	Ø	
6	07	7	
7	09	9	
8	56	*RTN	STOP OR GTO STEP 079
9	00	/////////	
110	00	/////////	
111	10	*E'	END OF CARD 2.
///2			
///3			
///4			
///5			
///6			
///7			
///8			
///9			

PRGM
OR
USER
GIVE
ROW

USER
PRESS
RUN

STEPS 112-223
USED AS STORAGE
REGISTERS 84-97.

147

FIGURE 8
LINEAR PROGRAMMING AID (HP-67)

USER KEYED LABELS:

a INITIALIZE LOADING	b	c INITIALIZE CONVERSION	d INITIALIZE PIVOT SEQ.	e INIT. Z_j SEQUENCE
A ROW A	B ROW B	C ROW C	D ROW D	E Z_j

FLAGS: OFF/ON

0			
1			
2			
3			

REGISTERS

0	
P1	STO ROWS
2	A & B
3	
4	
5	
6	
7	
8	
9	
0	
S1	STO ROWS
2	C & D
3	
4	
5	
6	
7	
8	
9	
A	STORE
B	COSTS
C	
D	
E	PIVOT VALUE

SET STATUS

DISPLAY / TRIG

DISPLAY	TRIG
SCI	DEG
ENG	RAD
FIX	GRAD
FLOATING PT	

OTHER LABELS

@	POSITIONING
1	CONTROLS
2	
3	
4	
5	
6	
7	

NOTES

Any row may be used as the Objective Row.

Costs of each row are manually stored as STO A, STO B, STO C, etc.

Pivot value manually stored as STO E.

STEP	KEY CODE	KEY ENTRY	
001	32 25 11	a LBL a	INITIALIZE LOADING
2	01	1	
3	36 33	h STO I	
4	84	R/S	
5	31 25 06	f LBL 6	
6	31 42	f P⇄S	
7	31 25 05	f LBL 5	
8	84	R/S	
9	31 25 13	f LBL C	LOAD ROW C
010	35 51 02	SF 2	
011	31 42	f P⇄S	
2	31 25 11	f LBL A	LOAD ROW A
3	43 ·	EEX	
4	02	2	
5	71	X	
6	31 83	f INT	
7	33 61 24	ST +i	
8	35 34	h RCL I	
9	01	1	
020	61	+	
1	35 33	h STO I	
2	35 71 02	F2?	
3	22 06	GTO 6	
4	22 05	GTO 5	
5	31 25 14	f LBL D	LOAD ROW D
6	35 51 02	SF 2	
7	31 42	f P⇄S	
8	31 25 12	f LBL B	LOAD ROW B
9	43	EEX	

STEP	KEY CODE	KEY ENTRY	NOTES
030	03	3	
1	81	÷	
2	32 83	gf RAC	
3	33 61 24	ST +i	
4	35 34	h RCL I	
5	01	1	
6	61	+	
7	35 33	h STO I	
8	35 71 02	F2?	
9	22 06	GTO 6	
040	22 05	GTO 5	
1	32 25 14	g LBL d	INITIALIZE PIVOT SEQUENCE
2	01	1	
3	35 33	h STO i	STO PIVOT VALUE
4	84	R/S	IN E
5	31 25 03	f LBL 3	
6	31 42	P ⇄ S	
7	31 25 04	f LBL 4	
8	84	R/S	
9	31 25 13	f LBL C	PIVOT ON ROW C
050	31 42	f P ⇄ S	
1	35 51 02	SF 2	
2	31 25 11	f LBL A	PIVOT ON ROW A
3	34 24	RCL i	
4	31 83	f INT	
5	34 15	RCL E	
6	43	EEX	
7	02	2	
8	71	X	
9	81	÷	
060	31 84	f -X-	
1	43	EEX	
2	02	2	
3	71	X	
4	31 83	f INT	
5	34 24	RCL i	
6	32 83	g FRAC	
7	61	+	
8	33 24	STO i	
9	35 34	h RCL I	
070	01	1	
1	61	+	
2	35 33	h STO I	
3	01	1	
4	51	-	
5	35 71 02	F2?	
6	22 03	GTO 3	
7	22 04	GTO 4	
8	31 25 14	f LBL D	PIVOT ON ROW D
9	35 51 02	SF2	

STEP	KEY CODE	KEY ENTRY	NOTES
080	31 42	f P⇄S	
1	31 25 12	f LBL B	PIVOT ON ROW B
2	34 24	RCL i	
3	32 83	g FRAC	
4	34 15	RCL E	
5	43	EEX	
6	03	3	
7	81	÷	
8	81	÷	
9	31 84	f -X-	
090	43	EEX	
1	03	3	
2	81	÷	
3	32 83	g FRAC	
4	34 24	RCL i	
5	31 83	f INT	
6	61	+	
7	33 24	STO i	
8	35 34	h RCL I	
9	01	1	
100	61	+	
1	35 33	h STO I	
2	01	1	
3	51	-	
4	35 71 02	f2?	
5	22 03	GTO 3	
6	22 04	GTO 4	
7	32 25 13	g LBL C	INITIALIZE CONVERSION
8	01	1	
9	35 33	h STO I	
110	84	R/S	
1	31 25 07	f LBL 7	
2	31 42	f P⇄S	
3	35 61 00	CF0	
4	31 25 00	f LBL 0	
5	84	R/S	
6	31 25 13	f LBL C	PIVOT/CONV. ROW C
7	35 51 00	SF0	
8	31 42	f P⇄S	
9	31 25 11	f LBL A	PIVOT/CONV. ROW A
120	34 24	RCL i	
1	31 83	f INT	
2	43	EEX	
3	02	2	
4	81	÷	
5	35 71 02	h F?2	
6	22 02	GTO 2	
7	22 01	GTO 1	
8	31 25 14	f LBL D	PIVOT/CONV. ROW D
9	35 51 00	h SF 0	

FOLLOWING LABELS:
(1) PRESS LBL FOR PIVOT ROW
(2) PRESS LBL FOR ROW
     BEING CONVERTED

STEP	KEY CODE	KEY ENTRY	NOTES
130	31 42	f P⇄S	
1	31 25 12	f LBL B	PIVOT/CONV. ROW B
2	34 24	RCL i	
3	32 83	g FRAC	
4	43	EEX	
5	03	3	
6	71	X	
7	35 71 02	h F?2	
8	22 02	GTO 2	
9	22 01	GTO 1	
140	31 25 01	f LBL 1	
1	33 13	STO C	
2	35 51 02	h SF 2	
3	35 71 00	h F? 0	
4	22 07	GTO 7	
5	22 00	GTO 0	
6	31 25 02	f LBL 2	
7	34 13	RCL C	
8	34 15	RCL E	
9	71	X	
150	51	−	
1	31 84	f -X-	
2	35 34	h RCL I	
3	01	1	
4	61	+	
5	35 33	h STO I	
6	35 71 00	h F? 0	
7	22 07	GTO 7	
8	22 00	GTO 0	
9	32 25 15	g LBL e	INITIALIZE Z_j SEQUENCE
160	01	1	STO ROW COST VALUES
1	35 33	h STO I	IN REGs A,B,C AND D
2	84	R/S	AS APPLICABLE
3	31 25 15	f LBL E	CALCULATE Z
4	44	CLX	
5	34 24	RCL i	
6	31 83	f INT	
7	43	EEX	
8	02	2	
9	81	÷	
170	34 11	RCL A	
1	71	X	
2	34 24	RCL i	
3	32 83	g FRAC	
4	43	EEX	
5	03	3	
6	71	X	
7	34 12	RCL B	
8	71	X	
9	61	+	

STEP	KEY CODE	KEY ENTRY	NOTES
180	31 42	f P⇄S	
1	34 24	RCL i	
2	31 83	f INT	
3	43	EEX	
4	02	2	
5	81	÷	
6	34 13	RCL C	
7	71	X	
8	61	+	
9	34 24	RCL i	
190	32 83	g FRAC	
1	43	EEX	
2	03	3	
3	71	X	
4	34 14	RCL D	
5	71	X	
6	61	+	
7	31 84	f -X-	
8	35 34	h RCL I	
9	01	1	
200	61	+	
1	35 33	h STO I	
2	31 42	f P≠S	
203	84	R/S	END OF THIS PROGRAM
4			
5			
6			
7			
8			
9			
210			
1			
2			
3			
4			
5			
6			
7			
8			
9			
220			
1			
2			
3			
224			END OF PROGRAMMABLE MEMORY
///5			
///6			
///7			
///8			
///9			

Attempts to minimize execution times often conflict with attempts to minimize program space. Because of the search patterns used internally by either calculator, the shortest possible program is not normally the fastest. Thus the user must usually consider eliminating steps and tolerating slower execution speeds in order to squeeze in the primary program itself, or in order to add additional user-defined routines to the program, as opposed to minimizing execution speeds. The best choice is normally to minimize, to the maximum extent possible, the number of steps required by all secondary routines which are never, or infrequently, used by the primary routine(s). Conversely, the best choice for the primary routine(s) is normally to minimize steps only to the extent that execution time is not adversely affected by the exclusion of addressing techniques which require more steps, but which execute faster. These kinds of tradeoffs are incorporated in the IRR routines discussed earlier, for example. Nesting, stacking and even separating the parts of routines can often be used to increase execution speed, without requiring additional steps. Thus, the location of each routine within the program may also be analyzed by the programmer with regard to desires concerning program space or execution times.

In summary, the optimization of calculator programs requires tradeoffs which are based upon experience and published literature, whereas basic, less efficient (but useful) calculator programs can be created even by the novice user.

11.   Machines of the Future

       a.   The National SemiConductor Model 7100 (NS-7100)

            The projected capabilities of this calculator

(Appendix B) are immense, exceeding the SR-52 and HP-67 by

at least one order of magnitude. Its projected cost is

$400. Its 4000-step basic library is structured as a

semiconductor cartridge, comparable (theoretically) to about

18 full SR-52 or HP-67 cards. The non-volatile memory thus

requires no drive motor in order to be read. Then, its

keyable memory and user cartridge give another 240 steps,

each. This totals an equivalent of 20 cards. Its projected

hardwired functions are extensive.

            It is difficult to comprehend the capacity of

this calculator. For example, consider the IRR program

developed earlier (Table 3) and multiply the capacity of

that program by 20, or consider 20 equally complex programs

all in your hand at one time and all usable in whatever

sequence is desired, either in total or as subroutines

of a user-keyed or user-stored additional program; subroutines

of indexed library programs may also be called from the

keyboard. This calculator does not have user-definable

keys. Instead, multiple-digit call numbers, as specified on

the library index, must be used to call a routine. This

system is designed to correspond closely to normal usage of

mathematical subscripting, but requires more keystrokes than

do systems which use user-defined keys.

154

The 240 step cartridge programmable by the user
is much bulkier and much more expensive ($15 vice 50¢ each)
than a card of 224 steps. This may be considered, in some
applications, as the price for security: unlike cards,
semiconductor cartridges cannot be accidentally altered by
an external magnetic field.

National Semiconductor is not planning to
manufacture a printer for the NS-7100; another (independent)
company is planning to do this.

Experience with the NS-7100 may show that most
problems can be solved without much additional programming
by the user, simply by using the extensively indexed
routines available from each library cartridge. The authors
of this work are biased, however, toward high use of
individualized, permanently stored programs. Such an
accumulation of programs on the NS-7100 user cartridges
would be extremely expensive. It thus appears that the
calculator may be better suited to users who do not mind
keying in programs manually when required and who are
otherwise satisfied to depend heavily upon preprogrammed
routines, rather than generating large numbers of
individualized programs.

    b.   The Texas Instruments Programmable
           Calculators 59 and 58 (TI-59; TI-58)

The TI-59 machine is the highest capacity hand-
held calculator projected to be available in June 1977. The
projected cost is $300. When combined with the PC-100A

printer it provides extensive alphanumeric formating (63 digits, letters, mathematical and Greek symbols) as well as discrete-point curve plotting (printable coordinate points 20 by n). This entire system will fit in a brief-case with room to spare. The TI-59 has extensive hardwired (semiconductor) functions, additional interchangeable 5000-step library semiconductor chips plug into the calculator's backside, and 480 step magnetic cards pass through its card reader for additional capacity. Only one card is required for the optional mode of 480 steps and 60 registers but an additional card may be used to store data in registers or to redefine step/register ratios; i.e., optional modes of 960 steps with no registers or 720 steps with 30 registers, etc. (The tradeoff is 10 registers per 80 steps, except that no more than 100 data-storage registers are available.) The calculator will also read cards during operation, as desired.

Key codes are double-merged, whereas they are unmerged on the SR-52 and triple merged on the HP-67 or NS-7100. Because of machine differences, the number of double merged TI-59 program steps required to accomplish programs equivalent to programs of triple-merged systems appear to yield a step ratio of 1.2 to 1. Hence its nominal mode of 60 registers and 480 card-steps (+5000 chip steps) roughly equates to 4567 NS-7100 steps or HP-67 steps. Of course, neither the NS-7100 or HP-67 has either this many steps or 60 storage registers.

Projected execution speeds are faster on the
TI-59 than on the SR-52.

To introduce the reader to the data range
available on the TI-59, several tape outputs of this system,
obtained from Texas Instruments emulators, are reproduced
below as Figures 9, 10, 11, and 12. These tapes show that
this calculator retains all of the good features of its
predecessor (the SR-52) but, evidently, few of the dis-
advantages of the SR-52. (See also Appendix B.)

(The HP-67 will also accomplish some types of
graphical output by using the limit of each line printed,
or figures within each line printed, to outline curves.
An example of an HP-97 sine curve generated in this manner
is provided as Figure 13 below for comparison with the
TI-59 sine curves duplicated in Figure 11.)

The TI-59 is likely to be marketed longer than
has been the case for many calculators. Instead of pursuing
new machine designs, the manufacturers appear to be re-
searching improvements to the interchangeable semiconducter
chips which clip into the machine. Thus even more capacity,
on the same machine, may become available in the future.
Meanwhile, the TI-58 is projected for introduction at the
same time as the TI-59, at a lower ($125) cost. The TI-58
is identical to the TI-59 except that it has no motor and
doesn't read cards. In any event, until National Semiconductor
provides a printing capability for the NS-7100, the TI-59
combined with the PC-100A printer is projected to be the most
compact system available of equal capabilities.

FIGURE 9

TI-59 AVAILABLE ALPHANUMERICS

Char	Code		Char	Code
0	01		T	37
1	02		.	40
2	03		U	41
3	04		V	42
4	05		W	43
5	06		X	44
6	07		Y	45
7	10		Z	46
8	11		+	47
9	12		×	50
A	13		≠	51
B	14		Γ	52
C	15		π	53
D	16		⊕	54
E	17		(	55
-	20		)	56
F	21		,	57
G	22		↑	60
H	23		%	61
I	24		‡	62
J	25		/	63
K	26		=	64
L	27		"	65
M	30		x	66
N	31		x̄	67
O	32		?	70
P	33		?	71
Q	34		÷	72
R	35		⊕	73
S	36		II	74
			^	75
			π	76
			Σ	77

FIGURE 10

TI-59 KEY CODE LISTING (CODES 00-99)
(Output Tape Width Cut Off Examples)

000	00	0	045	43	RCL	090	00	00
001	01	1	046	00	00	091	73	RC*
002	02	2	047	44	SUM	092	00	00
003	03	3	048	00	00	093	74	SM*
004	04	4	049	45	Y×	094	00	00
005	05	5	050	46	INS	095	75	-
006	06	6	051	47	CMS	096	76	LBL
007	07	7	052	48	EXC	097	00	0
008	08	8	053	00	00	098	77	GE
009	09	9	054	49	PRD	099	00	00
010	10	E'	055	00	00	100	00	00
011	11	A	056	50	I×I	101	78	Σ+
012	12	B	057	51	BST	102	79	x̄
013	13	C	058	52	EE	103	80	GRD
014	14	D	059	53	(	104	81	RST
015	15	E	060	54	)	105	82	HIR
016	16	A'	061	55	÷	106	00	00
017	17	B'	062	56	DEL	107	83	GO*
018	18	C'	063	57	ENG	108	00	00
019	19	D'	064	58	FIX	109	84	OP*
020	20	CLR	065	00	00	110	00	00
021	21	2ND	066	59	INT	111	85	+
022	22	INV	067	60	DEG	112	86	STF
023	23	LNX	068	61	GTO	113	00	00
024	24	CE	069	00	00	114	87	IFF
025	25	CLR	070	00	00	115	00	00
026	26	2ND	071	62	PG*	116	00	00
027	27	INV	072	00	00	117	00	00
028	28	LOG	073	63	EX*	118	88	DMS
029	29	CP	074	00	00	119	89	π
030	30	TAN	075	64	PD*	120	90	LST
031	31	LRN	076	00	00	121	91	R/S
032	32	X?T	077	65	×	122	92	RTN
033	33	X²	078	66	PAU	123	93	.
034	34	┌X	079	67	EQ	124	94	+/-
035	35	1/X	080	00	00	125	95	=
036	36	PGM	081	00	00	126	96	WRT
037	00	00	082	68	NOP	127	97	DSZ
038	37	P/R	083	69	OP	128	00	00
039	38	SIN	084	00	00	129	00	00
040	39	COS	085	70	RAD	130	00	00
041	40	IND	086	71	SBR	131	98	ADV
042	41	SST	087	00	00	132	99	PRT
043	42	STO	088	00	00	133	00	0
044	00	00	089	72	ST*			

TI-59 SINE CURVE PROGRAM AND OUTPUTS (ACTUAL SIZES)

PROGRAM

```
000  76  LBL
001  11   A
002  42  STO
003  00   00
004  92  RTN
005  76  LBL
006  12   B
007  42  STO
008  01   01
009  92  RTN
010  76  LBL
011  13   C
012  43  RCL
013  01   01
014  38  SIN
015  65   x
016  09   9
017  85   +
018  01   1
019  00   0
020  95   =
021  69  OP
022  07   07
023  43  RCL
024  00   00
025  44  SUM
026  01   01
027  61  GTO
028  13   C
029  00   0
```

STANDARD SPACING
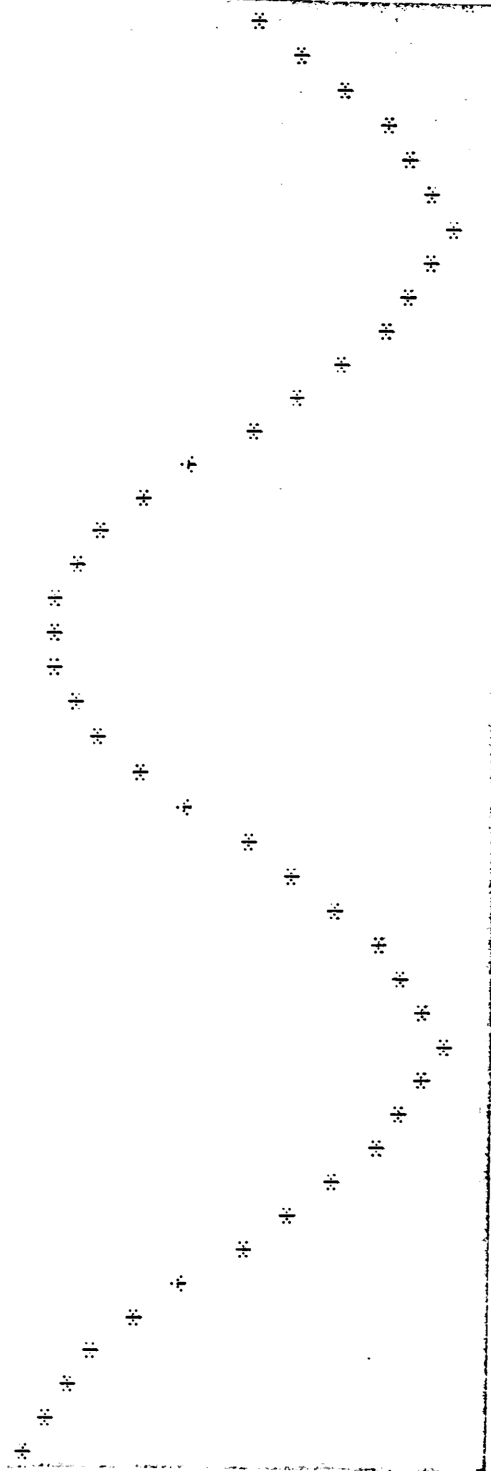
COMPRESSED SPACING
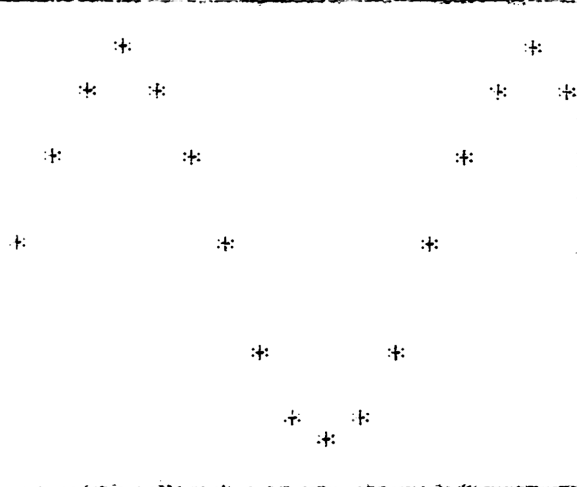
FIGURE 12
TI-59 ALPHANUMERIC CALENDAR OUTPUT
(Program Not Shown)

```
TEXAS INSTRUMENTS


        1978.


                           JAN
    S    M    T    W    T    F    S
    1    2    3    4    5    6    7
    8    9   10   11   12   13   14
   15   16   17   18   19   20   21
   22   23   24   25   26   27   28
   29   30   31



                           FEB
    S    M    T    W    T    F    S
                   1    2    3    4
    5    6    7    8    9   10   11
   12   13   14   15   16   17   18
   19   20   21   22   23   24   25
   26   27   28



                         MARCH
    S    M    T    W    T    F    S
                   1    2    3    4
    5    6    7    8    9   10   11
   12   13   14   15   16   17   18
   19   20   21   22   23   24   25
   26   27   28   29   30   31



                         APRIL
    S    M    T    W    T    F    S
                                  1
    2    3    4    5    6    7    8
    9   10   11   12   13   14   15
   16   17   18   19   20   21   22
   23   24   25   26   27   28   29
   30



                           MAY
    S    M    T    W    T    F    S
         1    2    3    4    5    6
    7    8    9   10   11   12   13
   14   15   16   17   18   19   20
   21   22   23   24   25   26   27
   28   29   30   31
```

FIGURE 13

HP-97 SINE CURVE PROGRAM AND OUTPUT
(Actual Sizes)

Note: X-axis labeling in 10 degree increments printed with curve display.

F. PROGRAMMABILITY CONCLUSIONS

Programming calculators, as discussed in this work, is the art of structuring algorithms such that the calculator executes those algorithms most efficiently. On the other hand, programmability relates to machine langauge; types and numbers of keystrokes required; merging of keystrokes in program steps; number of program steps available; functions available; program and data storage capacity; speed of execution; and ease of programming. All of these items have been discussed in detail. In the opinion of the authors, conclusions are:

(1) Programming calculators is certainly within the capability of the average college student and probably within the capability of most people at even lower levels of education or experience.

(2) Pragmatically, the differences in learning algebraic, AOS, or RPN languages are rather insignificant. However, RPN does add programming flexibility, once learned.

(3) Type and number of keystrokes required depends primarily upon functions available on the keyboard and, secondarily, upon the frequency that parentheses are used (in lieu of algebraic hierarchy) on non-RPN systems.

(4) Excluding use of the HP-67 "Σ+" key, the average number of merged program steps available on the HP-67 as compared to the SR-52 approaches a maximum ratio of 3 to 1, but the average ratio can be reduced to 1.3 to 1 for routine programming, dependent, of course, on abilities of the programmer.

(5) The number of steps available depends upon sub-paragraphs (2), (3) and (4), supra.

(6) Functional comparisons are as provided in Appendix B.

(7) The SR-52, compared to the HP-67, lacks an equivalent amount of program storage capacity. The SR-52 has easier access to extra registers for data storage whenever program steps can be sacrificed.

(8) Speed of execution is faster on the SR-52, and is particularly faster when exponentiation is used.

(9) The ease with which either calculator can be programmed depends more upon the experience and expertise of the programmer than upon the machine. However, the HP-67 is easier to use for merging programs or multiple card operations. Conversely, the direct addressing system used by the SR-52 is easier to use than the relative addressing used on the HP-67.

(10) The projected new machines (NG-7100 and TI-59) portend quantum improvements in capacity and capability. Questions concerning ease of use remain to be answered by actual usage; projections known to the authors are that Texas Instruments, Inc., intends to stop SR-52 production and start TI-59 production almost simultaneously. Both the projected NS-7100 and the projected TI-59 and TI-58 represent new levels of sophistication, but can only be theoretically compared to each other, at the time of this writing, since production models are not available to the authors.

Theoretically, none of these machines have serious competitors in the current market, with the possible exception of the HP-97 for some limited applications. Most information made available to the authors concerning these machines is represented by previous discussion and appendices hereto. In any case, such calculators indicate the beginning of a new generation of pocket calculators.

In summary, the prospective user should determine which of the above capacities, constraints, and (as yet) unproven manufacturer's projections, when compared to the somewhat different prices of these current and projected calculators, best fit his own parameters and yield the best value for the specific situation.

APPENDIX A

EDUCATIONAL GUIDELINES: CANDIDATE COURSES IN THE MANAGEMENT
SCIENCE CURRICULA, NAVAL POSTGRADUATE SCHOOL [15]

The following listed required courses are considered to
be readily adaptable for incorporation of the hand held
card programmable calculator as an educational tool:

Mathematics

MA 2040 - Matrix Algebra

MA 2305/2306 - Calculus I-II

MA 2045 - Computational Matrix Algebra

MA 1100 - Calculus and Vector Analysis

Accounting

MN 2150 - Financial Accounting

MN 3161 - Managerial Accounting

MN 4151 - Internal Control and Auditing

MN 4152 - Decision Making for Financial
         Management

MN 4154 - Financial Management in the Navy

MN 4161 - Controllership

MN 4162 - Cost Accounting

Probability and Statistics

PS 3005 - Probability

MN 3011/3012 - Probability and Statistics
for Management I-II

MN 3211/3212 - Operations Analysis for
Management I-II

OS 3062 - Intelligence Data Analysis

OS 3201/3202 - Fundamentals of Operations
and Systems Analysis

OS 32-3 - Survey of Operations and Systems
Analysis

In addition to the foregoing, use of hand-held card programmable calculators in the required curricula is considered to be of importance in preparation for and/or utilization in the following courses:

CS 0110 - Fortran Programming

CS 0113 - COBOL Programming

MN 3183 - Management Information Systems
and the Computer

MN 3214 - Operations Research Methodology

MN 3645 - Investigative Methods of Economics

MN 4145 - Systems Analysis

MN 4181 - Applications of Management Information Systems

OA 3604 - Linear Programming

OA 3620 - Inventory I

OA 3704 - Stochastic Processes

OA 4510 - Selected Topics of Probability and Statistics

OA 4614 - Methods and Practices of Systems Analysis

OA 4633 - Networks, Flows and Graphs

OA 4634 - Games of Strategy

OA 4651 - Search Theory and Detection

OA 4654 - Combat Models

APPENDIX B

COMPARISON OF CALCULATOR FUNCTIONS

|	SR 52	TI* 59	HP 67	NS* 7100
Logic System	AOS	AOS	RPN	ALG
Memory				
Parens	9	9		9
Automatic four memory stack			X	
Last X			X	
Program	224	480(1)	224	480(2)
Pending Operations	10	8	4	10
Addressable Memory (dedicated)	20	60(1)	26	26(2)
Algebraic hierarchy	X	X		X
Positioning Operations				
Stack roll down			X	
Stack roll up			X	
X, Y exchange			X	X
X, I exchange			X	X
X, Register nn exchange	X			
Fixed notation	X	X	X	X
Scientific notation	X	X	X	X
Engineering notation		X	X	X
Auto-overflow into scientific	X	X	X	(3)
Enter exponent	X	X	X	X
Change sign	X	X	X	X
Improper op and low battery indicator	X	X	X	X
X, T exchange		X		X
Programming Features				
Program review - back step/ single step	X	X	X	X
Insert/delete	X	X	X	X
Overwrite	X	X	X	
Relative Step number addressing			X	X
Relative Branching			X	X
Direct Branching Label	X	X	X	X
Direct Branching to Step no.	X	X		X
Pause		X	X	X
Condition Tests	5	12(4)	8	9
Flags	5	10	4	8
DSZ	X	X	X	X
ISZ (looping)		X	X	

169

	SR 52	TI* 59	HP 67	NS* 7100
Subroutine levels	2	6	3	4
File reader				
Stores programs and data		X	X	
Merges programs and data			X	
Automatic prompting	X		X	
Labels	72	72	20(5)	64
User-definable keys	10	10(6)	10	
Indirect control of:				
Data Storage and Recall	X	X	X	X
Storage arithmetic	X	X	X	X
Unconditional branching	X	X	X	X
Subroutine branching	2	6	3	9
DSZ	X	X	X	X
ISZ		X	X	
Display			X	X
Clearing options				
Clear entry	X	X	X	X
Clear T		X		
Clear all	X(a)			X
Clear registers	X	X	X	X
Clear program	X	X	X	X
Clear register nn	X(b)	X(b)		X
Built in Statistical Functions				
Mean, Standard deviation				
(no. variables)		2	2	1
Factorial	X		X	X
Summation $(n,\Sigma x,\Sigma x^2,\Sigma y,\Sigma y^2,\Sigma xy)$		X	X	
$(n,\Sigma x,\Sigma x^2)$				X
Built in Scientific/Mathematical Functions				
Trigonometric				
Decimal degrees	X(c)	X	X	X
Radians	X	X	X	X
Grads		X	X	X
Sin, Cos, Tan (plus inverses)	X	X	X	X
Rectangular/Polar conversion	X	X	X	X
Decimal angle time/degree angle time (H.M.S.)	X	X	X	X
Degree/Radian conversion	X		X	X
Conversion any angular measure to any other				X
Logarithmic	X	X	X	X
Log x, 10^x	X	X	X	X
Ln x, e^x	X	X	X	X
Exponentiate negative number	(d)	(d)	X	X

|	SR 52	TI* 59	HP 67	NS* 7100
Other Built in Functions				
Y^X , x , x^2 , Pi , 1/x	X	X	X	X
+ , - , x , ÷	X	X	X	X
X Root of Y		X		X
%			X	X
% change			X	X
Absolute value		X	X	X
Integer/fraction truncation		X	X	X
Rounding key			X	
Merge programs			X	
Conversions: in/cm, gal/liter Kg/lb, Newtons/lb force, Deg C/Deg F, BTU/Foot lb force				X
Printing Features [7]				
Print X	X	X	X	
List stack registers			X	
List data		X	X	
List labels		X	X	
Paper advance	X	X	X	
Variable print modes	X	X	X	
Print space	X	X	X	
List program	X	X	X	
List crom program [8]		X		

NOTES:

*Based solely on manufacturers claims. NS7100 uses chip vice card to record programs.

(1) 480 steps nominal (mainframe) with 60 registers. Alternate levels: 160 steps with up to 98 registers or 960 steps with 0 registers; plus 5000 step library CROM.

(2) 240 mainframe and 240 permanent file cartridge. Mainframe retains program plus data as long as batteries are charged; or 240 mainframe plus 4000 + library cartridge. Can retrieve particular numbers from pending operations (total of 37 registers.)

(3) Rounds as ordered.

(4) Compare X to T (4 tests) plus compare X to registers (8 tests).

(5) Labels are reusable.

(6) 10 user definable functions plus call label.

(7) SR52 and TI59 plug into a separate printing unit. HP67 cards must be printed with an HP97. All functions of HP67 and HP97 are the same except for the integral printing unit. No printer presently available for NS 7100 (see 8 below).

(8) The TI59 CROM (library chip) is coded by "page." A complete CROM has 10 "pages" with 500 steps per "page." NS 7100 printer to be manufactured by separate manufacturer. In addition NS 7100 will have full input/output capability with other systems. Also, NS 7100 displays GTO, label and shift codes.

(a) except registers 20-99

(b) only registers 60-69 all at once

(c) activated by a switch vice a button

(d) will work but gives error flash; will halt a running program
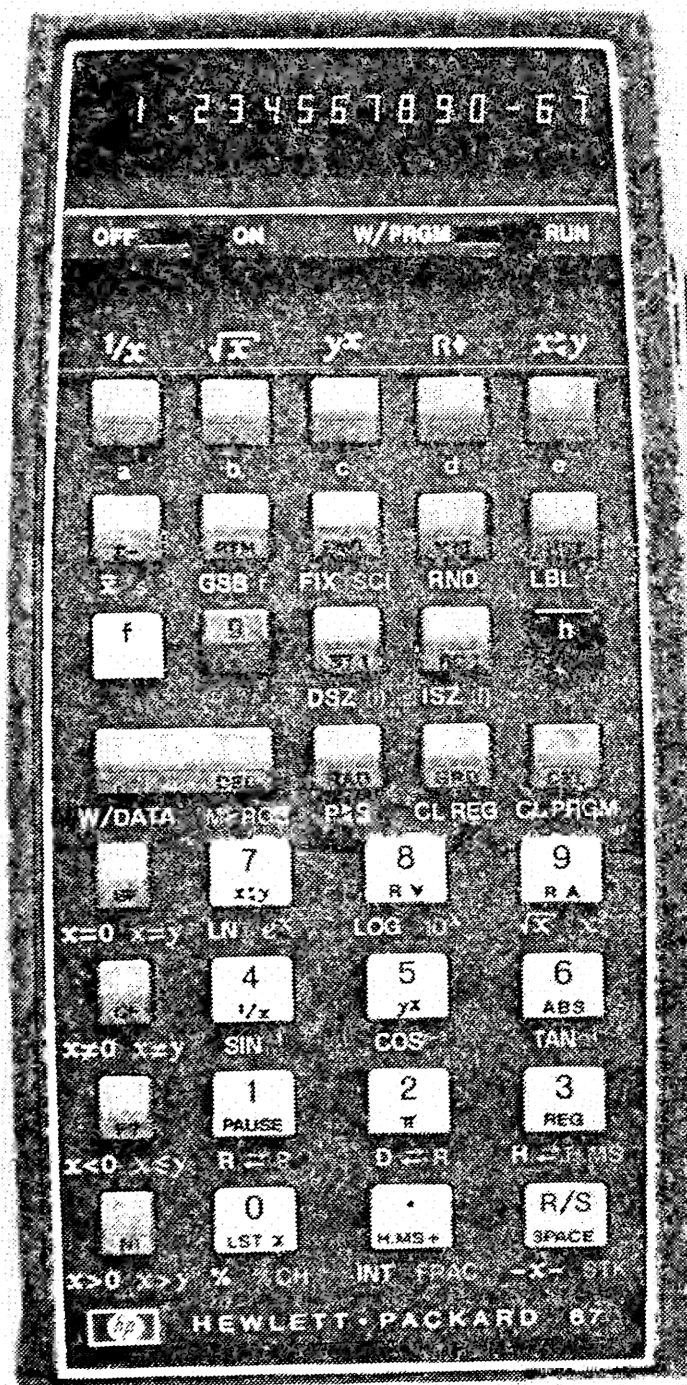
HP-67
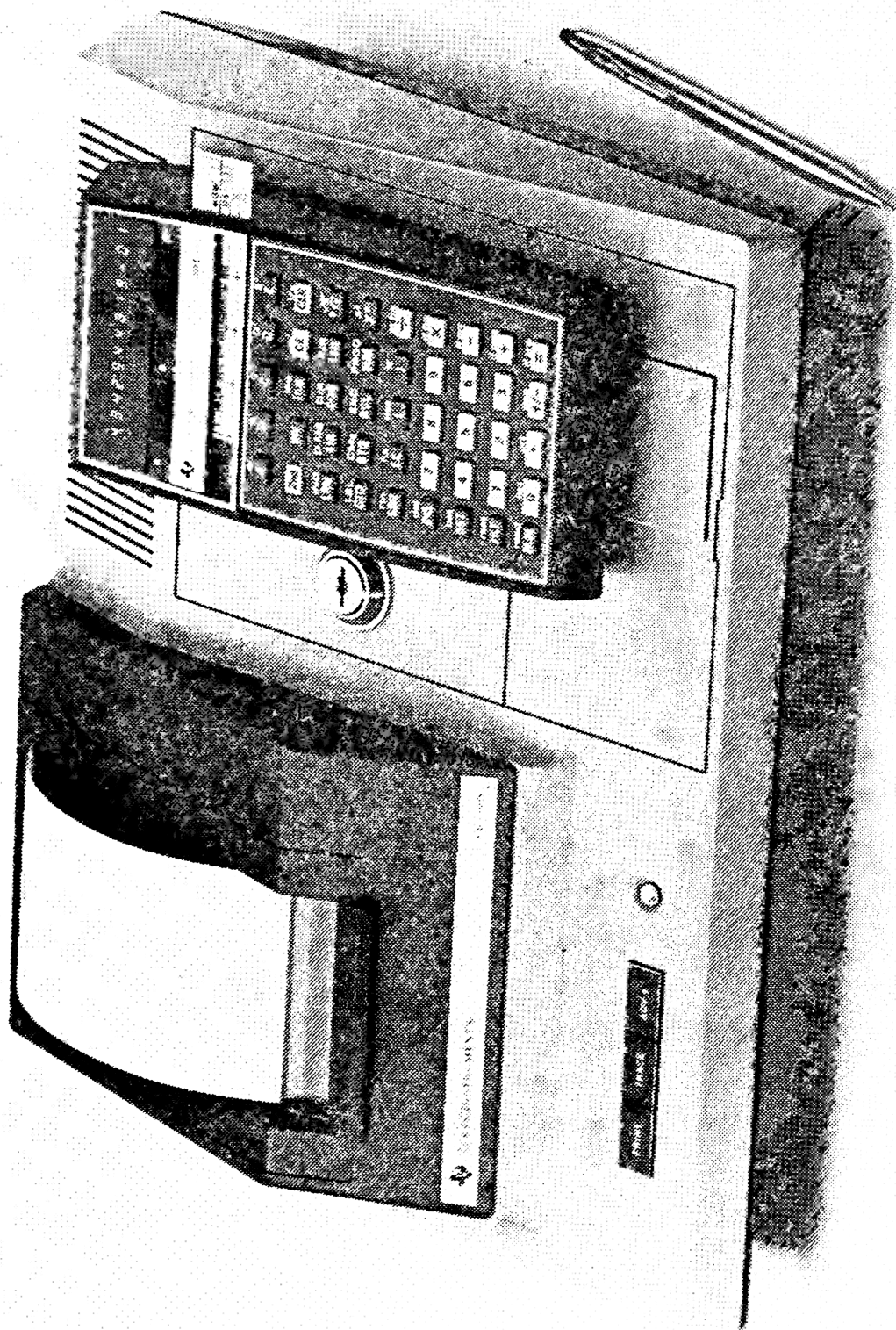
SR-52 Mounted on PC-100

HP-97
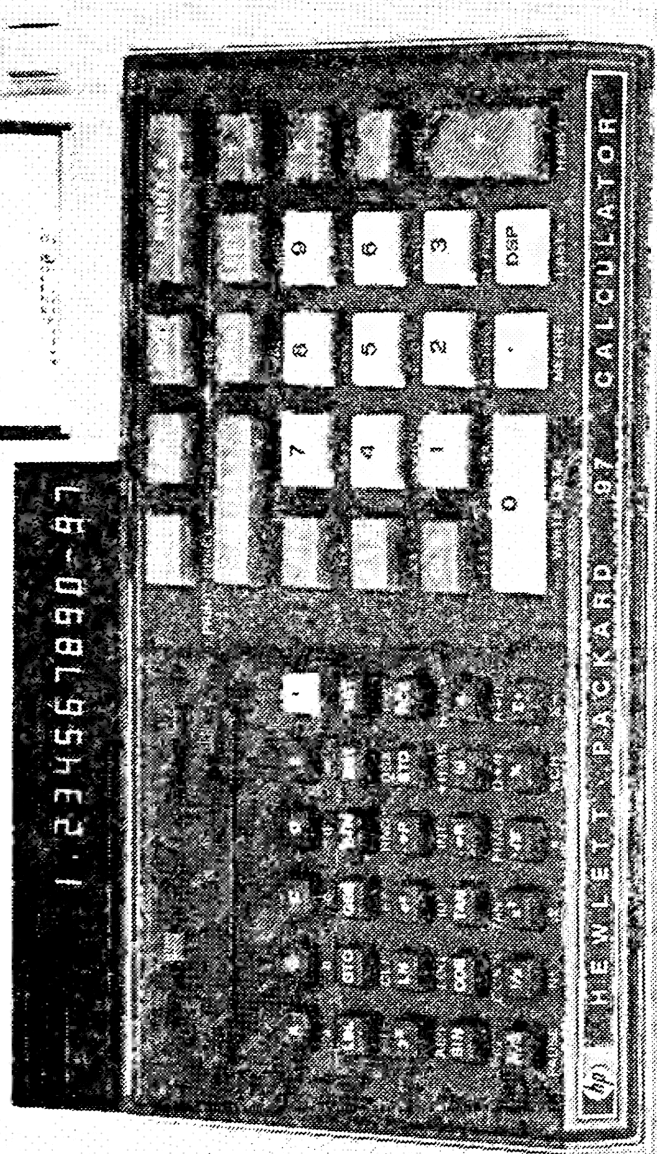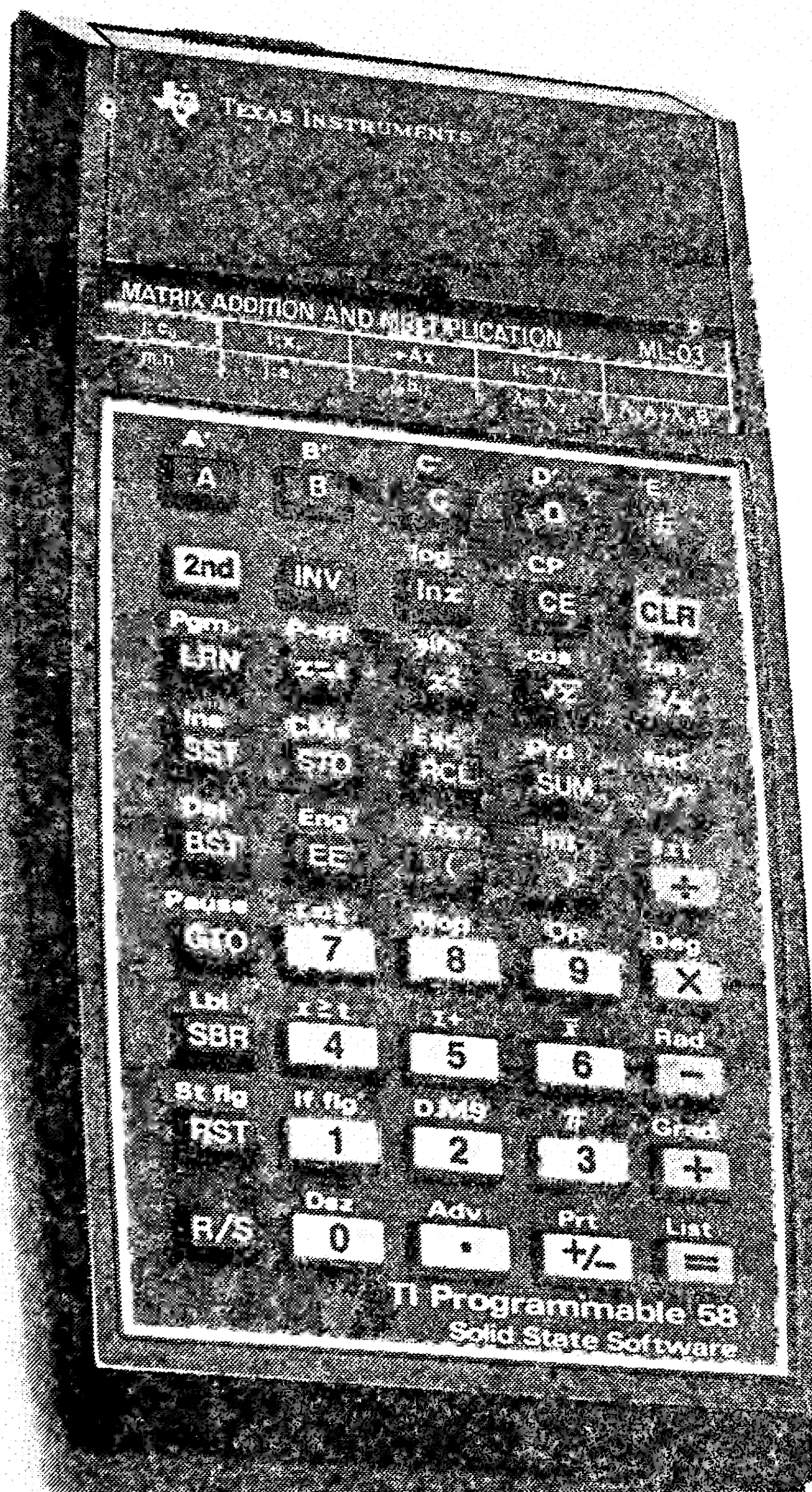
TI-58
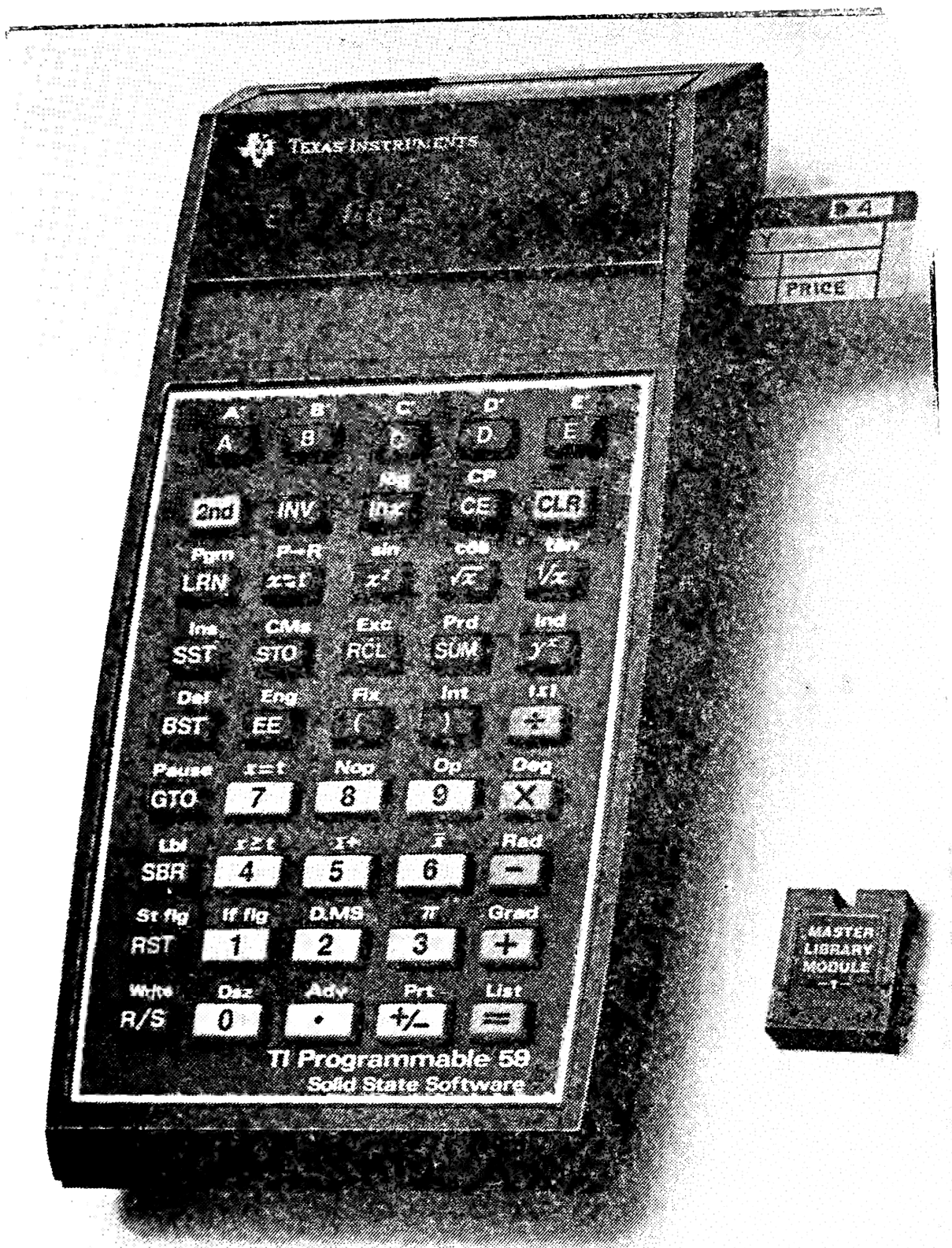
TI-59
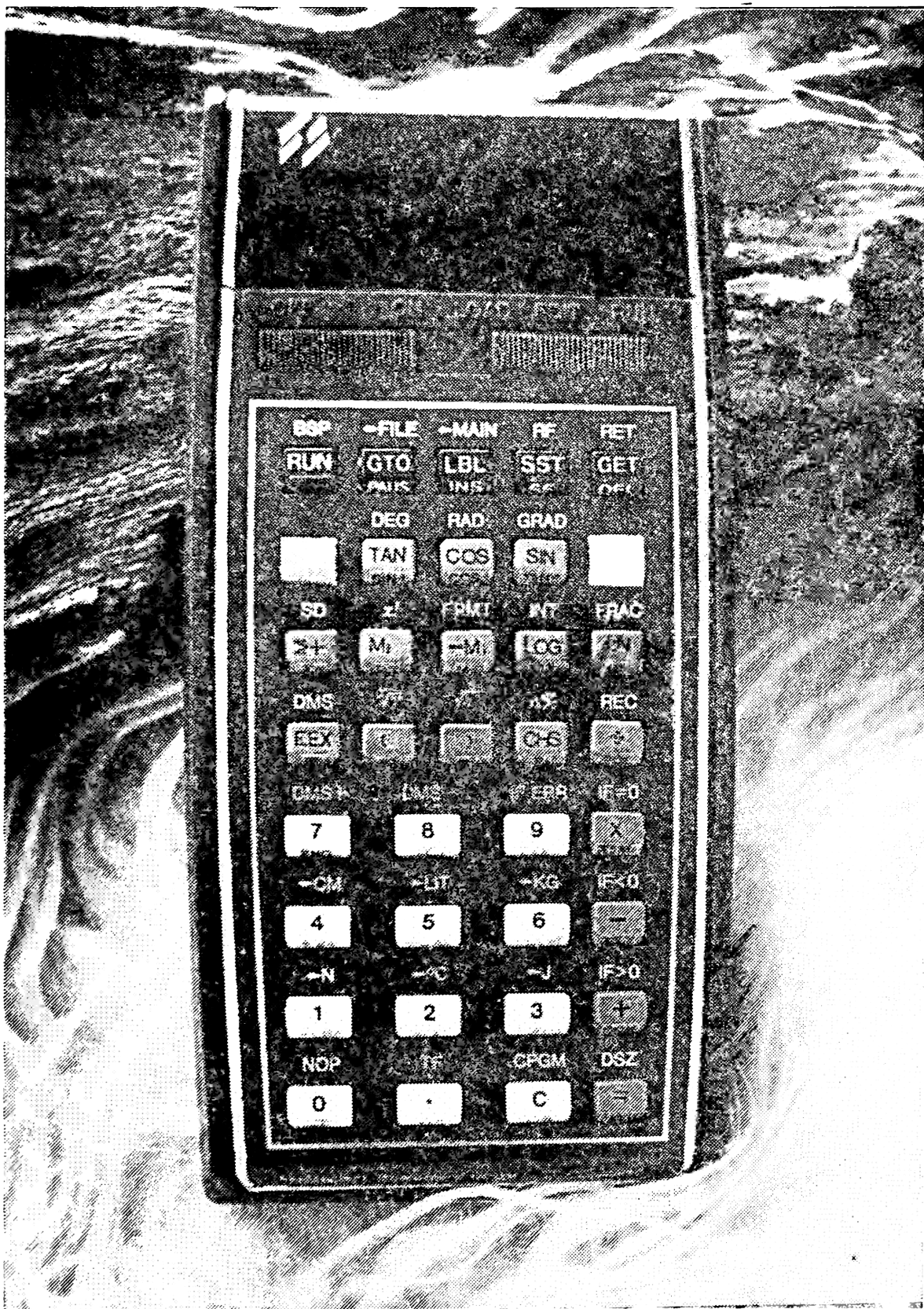
NS Model 7100

APPENDIX C

FLEET USES

The following non-exhaustive list of fleet functions indicates the broad range of applications for programmable calculators:

Aviation:

Tactics

Navigation

Flight profiles

Weight and balance (particularly cargo)

Analysis of Maintenance Material Management Data

Trend analysis (quality assurance)

Surface:

Tactics

Navigation and plot

Analysis of Planned Maintenance System Data

Trend Analysis (quality assurance)

Damage Control

Ships Engineering

Civil Engineer Corps:

Surveying

Structural design

Roadway design

Construction

Financial Analysis

Medical Corps:

Biological Analysis

Pathological Analysis

Meteorology:

Analysis of Meteorological data

Prediction

Research and Development:

All forms of weapons system design, research and
development analysis.

All of the following kinds of officers should be issued card-
programmable calculators:

Aeronautical Engineering Duty

Aviation Maintenance Duty

Weapons Engineering Duty

Civil Engineer Corps

Supply Corps

P-coded Operations analysis and engineering officer
billets

P-coded Managerial billets

180

APPENDIX D

EXAMPLES OF USER-SUBMITTED PROGRAMS [16,17]
(Extracted from some 5000 available programs)

BUSINESS

Experience curve for manufacturing cost

Summation of Ledger Columns

Amortization Schedule

New Product Growth Factor - Gompertz Method

Multi-Family Land Use Evaluation

Manufacturing Learning Curve - Unit and Cumulative Cost

Pert Estimating

Universal Rate of Return

Multivariate Corporate Failure Prediction Model

MATHEMATICS

4 x 4 Determinant and Simultaneous Equations

Complex Arithmetic

Complex Functions

Radar Range-height Calculation

Function and Derivatives

Maxima and Minima

La Grange Polynomial Interpolation

Numerical Integration

Differential Equations

ENGINEERING

    Phase Shift Oscillator Design

    Rectangular Waveguide Calculations

    Transmittal Laser Pulse Energy

    Aircraft Flyby Look - Angles and Rates

    Bode of Transfer Function with Eighth Order Polynomial

    Phase Locked Loop Design, Acoustic Horn Evaluation

    Ballistic Missile, Range, Elevation Angle

    Biomechanics

    Two-Instrument Radial Survey

COMPUTER SCIENCE

    Binary Coded Decimal with Parity to Decimal Conversion

    Control Data Computer Octal Dump Decoding

    Decimal to IBM 370 Floating Point Hexadecimal Conversion

    Octal Debug Aid

    Optimum Disk File Blocking

    Timesharing Wait Model

    Sentential Logic

PROBABILITY AND STATISTICS

    Moments, Skewness and Kurtosis

    Permutations and Combinations

    Two-State Markov Chain Matrix

    Five Variable Regression Analysis

    Chi Square Proportion Difference

    Biserial Correlation Coefficient

    The Cochran Q Test

QUALITY ASSURANCE/RELIABILITY

 Redundant System Reliability

 Aerhenius Chemical Reaction Rate

 X Bar and R Control Charts

 Correlation: Reliability and Validity

NATURAL SCIENCES

 Environmental Noise Levels

 Acid-base Balance

 Creatinine Clearance

 Enzyme Kinetics

 ECG Data Optimization

 Blood Acid-Base Status

 Fick Cardiac Output

 Oxygen Saturation and Content

 Tumor Growth

 Absorption Spectroscopy Calibration

 Orbital Mechanics

GAMES

 Casino Game Model for Study of Behavior

 Simulation Wargame

 Combat Odds

 Space Ship Landing Simulator

 Underwater Submarine Hunt

 Biorhythms

 Space Battle

 Space Docking

AIR NAVIGATION

Flight, Plan and Verification

Predict Freezing Level

Dead Reckoning

Rhumbline Navigation

Great Circle Flying

Position and Navigation by One VOR

Weight and Balance

Moon Sight Reduction

MARINE NAVIGATION

Course Made Good from Three Bearings

Map Initialization

Running Fix from One Object

Planet Location

Sextant Correction

Storm Avoidance

Distance and Bearing to the Mark Sun Sight Reduction

LIST OF REFERENCES

1. Roberts, E.M., _Fingertip Math_, Texas Instruments, Inc. 1974.

2. Naval Research Laboratory Report 2938, _Programmable Calculator — Mini Computer Tradeoffs_, by K.P. Thompson, December 1974.

3. Rogers, Joy J., "The Electronic Calculator — Another Teaching Aid?" _The Arithmetic Teacher_, Vol. 23, no. 7, November 1976, pp. 527-530.

4. Bell, Max S., "Calculators in Elementary Schools? Some Tentative Guidelines and Questions Based on Classroom Experience," 3 _Ibid_, pp. 502-509.

5. National Science Foundation Final Report. Grant no. EPP 75-16157, _Electronic Hand Calculators: The Implications for Pre-College Education_, by Marilyn N. Snydam, February 1976.

6. Interview of Mr. George Shultz, Manager, Academic Sales, Educational Calculator Devices, Inc., Laguna Beach, California.

7. Naval Postgraduate School Catalog 1976-77.

8. Gaskell, R.E., _Turning On Your Mathematics_, Naval Postgraduate School, September 1976.

9. Interview of Commander C. Gibfreid, Chairman, Computer Science Department, Naval Postgraduate School.

10. Lewart, Cass R. and Hal Brown, _65 Notes_, January 1977, Vol. 4, No. 1, pp. 1 and 12.

11. Interview of Lieutenant Commander P.I. Harvey, Tactical Development and Evaluation Officer; Commander, Patrol Wings U.S. Pacific Fleet.

12. _What to Look for Before You Buy an Advanced Calculator_, Approach 13-30 Corporation, 1976.

13. Interview of Mr. Ron Eldrid, Hewlett Packard Company, Santa Clara, California.

14. 11 _Ibid_.

15. 7 _Ibid_.

16. *Catalog of Contributed Programs*, HP65/67/97, Hewlett Packard Company, July 1976.

17. *Software Catalog (PPX-52)*, Texas Instruments, Inc. November 1976.

18. Hadley, G., *Linear Programming*, Addison-Wesley, 1963.

19. 10 *Ibid*.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Documentation Center   2
    Cameron Station
    Alexandria, Virginia 22314

2. Library, Code 0212   2
    Naval Postgraduate School
    Monterey, California 93940

3. Dr. Carl R. Jones   2
    Department Chairman, Code 54Js
    Administrative Sciences Department
    Naval Postgraduate School
    Monterey, California 93940

4. Professor Harold J. Larson, Code 55La   1
    Operations Research Department
    Naval Postgraduate School
    Monterey, California 93940

5. Assoc. Professor Rex. H. Shudde, Code 55Su   10
    Operations Research Department
    Naval Postgraduate School
    Monterey, California 93940

6. Lieutenant Commander Harry R. Kruse, USN   5
    Attack Carrier Air Wing Eleven
    FPO, San Francisco 96601

7. Lieutenant Commander Alan Burkett, CEC, USN   5
    COMRNCF REP West
    Suite 203
    1220 Pacific Hiway
    San Diego, California 93132

8. Chief of Naval Operations   1
    ATTN: OP0942D21
    Washington, D.C. 20350

9. Chief of Civil Engineers   1
    200 Stovall Street
    Alexandria, Virginia 22332

10. Mr. D.S. Hurst   5
    Naval Air Systems Command
    ATTN: AIR340C
    Washington, D.C. 20361

187

11. Capt. G. Dowd, USN   1
    Naval Air Systems Command
    ATTN: PMA 270
    Washington, D.C. 20361

12. LCDR P.I. Harvey, USN   1
    Commander, Patrol Wings   ATTN: Code 51
      U.S. Pacific Fleet
    NAS Moffett Field, California 94035

13. Mr. Richard Nelson   1
    HP 65 Users Club
    2541 W. Camden Place
    Santa Ana, California 92704

14. Mr. Ron Eldrid   3
    Advanced Products Division
    Hewlett Packard Company
    3003 Scott Blvd
    Santa Clara, California 95050

15. Mr. James Chumbley   3
    National Semiconductor Corp.
    1177 Kern Avenue
    Sunnyvale, California 94086

16. Mr. Stavro E. Prodromou   3
    Calculator Division
    Texas Instruments, Inc.
    P.O. Box 5012, MS5
    Dallas, Texas 75222

17. Mr. Marvin Johnson   1
    Vice President
    University of Arizona
    Tucson, Arizona

18. Mr. George Schultz   1
    Manager, Academic Sales
    Educational Calculator Devices, Inc.
    P.O. Box 974
    Laguna Beach, California 92652

19. National Council of Teachers of Mathematics   1
    1906 Association Drive
    Reston, Virginia 22091

20. Dr. Marilyn Suydam   1
    Associate Professor
    Ohio State University
    1200 Chambers Rd., 310
    Columbus, Ohio 43212

21. Dr. Jacob T. Schwartz, Chairman   1
    Department of Computer Science
    New York University
    251 Mercer Street
    New York, New York 10012

22. Mr. Robert A. Sulit   1
    Head, Operations Research Division
    ATTN: Code 1860
    Department of the Navy Naval Ship Research
     and Development Center
    Bethesda, Maryland 20084

23. Mr. Andre A. Pugin   1
    Naval Weapons Engineering Support
     Activity
    ATTN: Code ESA 83
    Washington, D.C. 20390

24. PPX-52   1
    Texas Instruments, Inc.
    P.O. Box 22283
    Dallas, Texas 75222

25. Mr. R.C. Vanderburgh   1
    52 Notes
    9459 Taylorsville Road
    Dayton, Ohio 45424

26. Mr. Jim Grant,   1
    Head, Programmable Calculator Support Group
    ATTN: PROCAL
    Naval Electronics Laboratory Center
    271 Catalina Blvd.
    San Diego, California 92152

27. HP Key Notes   1
    Hewlett-Packard Company
    Users Library
    1000 N.E. Circle Boulevard
    Corvallis, Oregon 97330

28. U.S. Naval Academy   1
    ATTN: Administrative Science Dept.
    Annapolis, Maryland 21402

29. U.S. Coast Guard Academy   1
    ATTN: Dept. of Mathematics
    New London, Connecticut 06320

30. U.S. Air Force Academy   1
    ATTN: Administrative Science Dept.
    Colorado Springs, Colorado 80840

31. U.S. Military Academy   1
    ATTN: Administrative Science Department
    West Point, New York   10996

32. Capt. J.M. Barron, USN   1
    ATTN: Code 03
    Director of Programs
    Naval Postgraduate School
    Monterey, California   93940

33. Professor R.E. Gaskell, Code 53Gl   1
    Mathematics Department
    Naval Postgraduate School
    Monterey, California   93940

34. Assoc. Prof. Rudolph Panholzer, Code 62Pz   1
    Electrical Engineering Department
    Naval Postgraduate School
    Monterey, California   93940

35. Assoc. Prof. A.W. McMasters, Code 54Mg   1
    Administrative Sciences Department
    Naval Postgraduate School
    Monterey, California   93940

36. Adjunct Professor R. W. Hamming, Code 52Hg   1
    Computer Science Department
    Naval Postgraduate School
    Monterey, California   93940

37. Mr. Alvin Andrus   1
    Office of Naval Research
    ATTN: Code 230
    800 North Quincy Street
    Arlington, Virginia   22217

38. Mr. Dean Lampman   1
    National 7100 Users Club
    5440 Cooper Road
    Cincinnati, Ohio   45245

39. R.N. Forrest, Code 55Fo   1
    Operations Research Department
    Naval Postgraduate School
    Monterey, California   93940