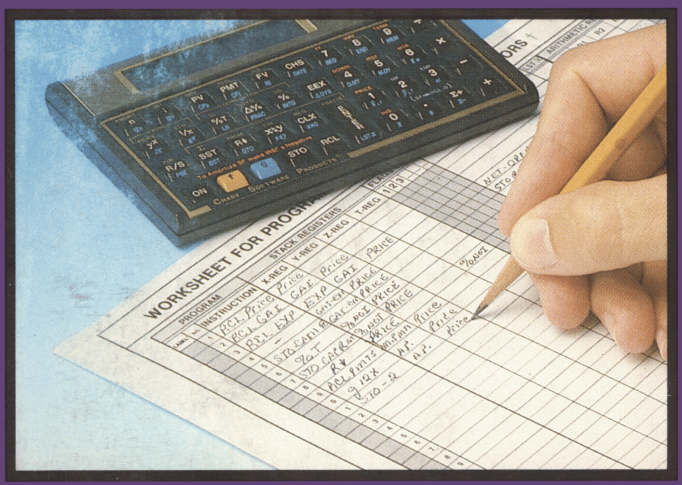
HP-12C

Programming Hints For Your HP-12C



A Step By Step Procedure For Easy Program Development.

CHARY

Chary Software Products, a division of Chary Corporation makes no warranties, either expressed or implied, with respect to the software described in this book, its quality, performance, merchantability, or fitness for any particular purpose, nor does Chary Software Products make any warranties, either expressed or implied, with respect to the accuracy of the contents of this book. Furthermore, Chary Software Products reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Chary Software Products to notify any person of such revisions or changes.

REGISTRATION CARD

This registration card entitles you to receive, free, any corrections made to the product(s) listed below. You will also be retained on CHARY's mailing list and will be informed of all new products, services and/or revisions as they become available.

NAME: _____

ADDRESS: _____

CITY: _____ STATE: _____ ZIP: _____

PRODUCT(s) PURCHASED: _____

DATE OF PURCHASE: _____

STORE NAME WHERE PURCHASED: _____

CHARY CORPORATION
21213-B HAWTHORNE BLVD. SUITE #5120, TORRANCE, CA 90509
(213) 545-7584

Place
Stamp
Here

CHARY CORPORATION
21213-B HAWTHORNE BLVD.
SUITE #5120
TORRANCE, CA 90509-2881

PROGRAMMING HINTS FOR YOUR HP-12C

Dennis W. Borgogno
Michael F. Aulert

CHARY SOFTWARE PRODUCTS
CHARY CORPORATION

PH-0995.7500 Rev. A 08/86 Printed in U.S.A.

Contents

Preface

This book contains two sections which are distinctly different in their purpose. In itself, this book is not meant to be a programming manual but an introduction to the basics of programming and to the ease of programming the HP-12C calculator.

The first section of this book is intended to give the reader a better awareness of the thought process involved in writing a program for the HP-12C. Hopefully, after reviewing this section the reader will be able to better utilize the information found in the Programming Section of the Hewlett-Packard Owner's Handbook. The first section is also intended to serve as a means of encouraging the reader to forge ahead and begin to realize the true potential of the HP-12C calculator through programming.

The second section of this book shares with the reader, who is familiar with programming the HP-12C, several points on how to minimize the amount of code used when writing a program. Again, this section is intended to be used in conjunction with the Hewlett-Packard Owner's Handbook and only after the reader has obtained a basic understanding of how to write programs for the HP-12C calculator.

Introduction

Programming is nothing more than storing away a sequence of keystrokes which are frequently used. Thus, programs are simply keystrokes stored in memory. Suppose you are required to determine either a cash flow or a monthly payment which will require a large number of keystrokes and suppose this requirement occurs often. It would then become advantageous to program your calculator with the required keystrokes. Then instead of pressing all the keystrokes each time you need an answer, you simply input the required data and press one key to start the program. The calculator does the rest automatically.

To assist the reader with an organized approach to develop such time saving programs, this book presents an easy five step approach to generating the required keystroke sequences. This approach is then used to generate, via an example, a comprehensive program from which the user will be able to see the time saving benefit of program utilization.

Upon completion of this book the novice will have a greater awareness of the ease of programming the HP-12C calculator. Both the novice and the more advanced programmer will benefit from the approach used in generating keystroke sequences for the HP-12C calculator presented herein.

The Basic Function of A Calculator Program

A calculator program is a set of precisely stated instructions which tell the calculator the order in which to execute a desired set of computations on a given set of data and where to store or "output" the results of those computations for later use and/or review by the user.

In the HP-12C calculator the process of reducing data is the same as in any other computer. However, like all other computers, the HP-12C is limited by its design and the instructions which have been implemented into that design.

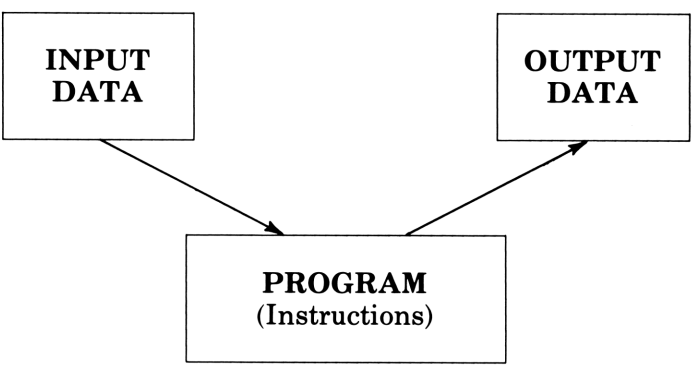


Figure 1 - The Program

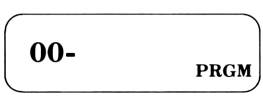
The program resides in memory and executes the specified computations on the Input Data to produce the Output Data.

Programming Features of The HP-12C

The **programming features** of the HP-12C discussed herein will be limited in scope with just a brief description of the "programming" keys and the available sets of registers and how each may be used. Greater detail can be obtained from your Owner's Handbook for the HP-12C.

First, a word about the two "modes of operation" associated with the calculator. The calculator in its everyday normal use functions in a "mode" which is referred to as the "calculator" mode. In this mode the calculator does the normal arithmetic functions, financial functions, percentages, etc. The second mode of operation associated with the HP-12C is referred to as the "programming" mode. It is in this mode the user "programs" the calculator, that is, the user keystrokes a series of instructions called a program into the calculator or makes modifications to an existing program. This mode is only used to load and modify programs. *You do not "run" or "execute" your programs in the programming mode.*

To place the calculator in the "programming" mode you press the gold **"f"** function key followed by the **"P/R"** key (P/R is labeled above the **R/S** key). When this is done the calculator will display:



Whenever you can observe the indicator "PRGM" in the display you are in the "programming" mode. To exit the programming mode you simply press the **"f P/R"** key sequence and you will return to the calculator mode of operation. You may also exit the programming mode by "cycling your calculator", that is, turn it **off** and then turn it back **on**. Every time the calculator is turned on it comes on in the calculator mode.

Instructions:

In addition to the "standard" key set on the HP-12C there are seven additional keys which are used exclusively for programming. These are:

R/S - The "RUN/STOP" Key - When this key is pressed it causes program execution to begin at the instruction the calculator is pointing to in memory. When program execution encounters this instruction in the program code, execution is stopped. This instruction is most often used to stop execution to allow the user to record the displayed results. It can also be used as a break point in debugging programs.

PSE - The PAUSE Key - When the program executes this instruction a momentary pause in the execution occurs (approximately 1 sec.) and the contents of the X-register are displayed. This instruction is most often used to display intermediate results in debugging programs.

GTO - The "GO TO" Instruction - This instruction is used to change program flow. It may be used by itself, however, most often it is used with a "conditional test" instruction such as $X \leq Y$ or $X = 0$ (i.e. if $X = 0$ then GTO XX).

X ≤ Y - The "X Less Than Or Equal To Y" Key - This is a "conditional test" instruction which asks the question: is the number in the X-Register less than or equal to the number in the Y-Register? If the answer to the question is yes (true) the calculator will then execute the instruction immediately following the $X \leq Y$ instruction. If the answer to the question is no (false) the calculator will skip the instruction immediately following the test and execute the next instruction in the sequence. This logic is considered to be "DO IF TRUE" logic (i.e. DO the next instruction IF the test is TRUE).

X = 0 - The "EQUAL ZERO" Test - A conditional test to see if the content of the X-Registers is zero. This instruction is also a "DO IF TRUE" logical test. It will "DO" (execute) the next instruction if the answer to the test is TRUE. If the answer is FALSE it will skip the instruction immediately following the test and execute the next instruction in the sequence.

SST, BST - The "SINGLE STEP" and "BACK STEP" Keys - These are not used as program instructions but are used only for program editing when you wish to "step" forward to the next program instruction or "step" backwards to the previous instructions. The **SST** key will also allow the execution of a program to be done one programming line at a time. Each time the **SST** key is pressed a program line of code is executed. This becomes valuable when "debugging" your program as discussed later.

Multiple Programs

The HP-12C can have up to 99 program lines of code. These 99 lines can be one large program or many small programs. However, the total number of program lines cannot exceed 99.

To have multiple programs in your calculator there are three very important things to do which are highly recommended. There may be others you want to do but the following should be a minimum:

1) Each program should end with a g GTO XX instruction which repositions the calculator to the beginning of the program (i.e. if program B begins on line 22 and ends on line 35 then the last instruction should be g GTO 22).

2) Keep a log as to what program begins on which program line. In order to run program "B" that begins on line 22 the calculator must be positioned to start instruction execution at line 22.

Example:

Program Name	Program Description	Program Start
A	Buy-vs-Sell	00
B	Lease Costs	22
C	Buy-vs-Rent	42

3) Keep a log or description of what each program does and how to run the program. You will need to know what inputs are required and where each of them is to be stored.

Available Registers

The HP-12C calculator is a very powerful calculator. It derives much of its power from the many functions which are "hardwired" into the basic unit. However, a great deal more power comes from utilizing the programming features that also come with the calculator.

The calculator has 30 registers which can be used in programming.

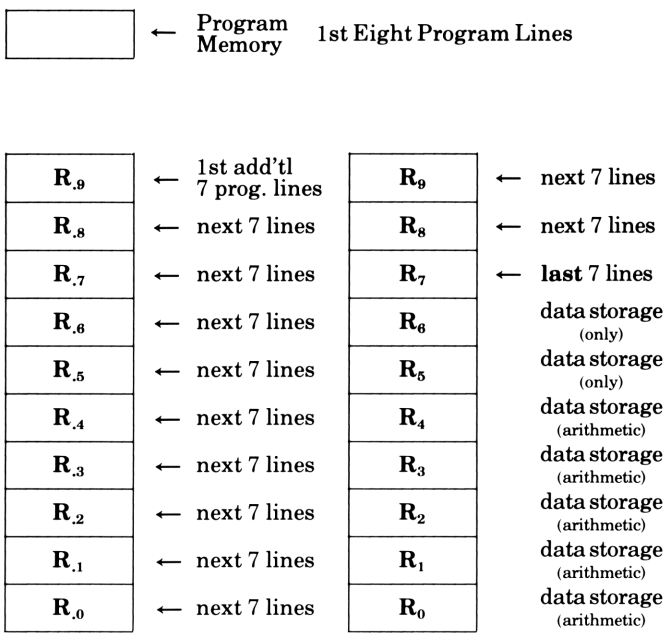
The registers are as follows:

Registers 0 through 9	(R_0 - R_9)	10
Registers .0 through .9	($R_{.0}$ - $R_{.9}$)	10
Financial Registers	(n, i, PV, PMT, FV)	5
Stack Registers	(X, Y, Z, T)	4
Last X Register	(LSTx)	1
	Total Registers	30

Utilization of the Registers

The usage of the various registers for **program storage** and **data storage** is given below.

For
Program Storage



There are 13 registers reserved for **program storage** in addition to PROGRAM MEMORY, they are: $\mathbf{R}_{.9}$ through $\mathbf{R}_{.0}$ plus $\mathbf{R}_9$, $\mathbf{R}_8$ and $\mathbf{R}_7$. The "unused" **program storage** registers, if any, can be used for **data storage**.

In addition to the twenty registers mentioned there are another ten (10) registers which can be used for **data storage** when programming your calculator. These are:

Registers Available for Data Storage

Financial Registers	5
Stack Registers	4
The Last X Register	1
Registers	10

With these ten registers and the seven registers $\mathbf{R_0}$ through $\mathbf{R_6}$ (listed above), there are a minimum of seventeen (17) registers "reserved" for **data storage**. In addition, any unused **program storage** registers can also be used (i.e. if your program only uses registers $R_{.9}$ through $R_{.0}$ then the 3 unused registers, R_9, R_8, and R_7, normally reserved for storage of program lines, can be used for data storage.)

Thus: the total number of **data storage** registers is:

17 + unused **program storage** registers

The Recommended Use of Registers

Since, for the most part, the "register sets" (financial, stack, and arithmetic register sets) each have different functions, it becomes advantageous to use them in the most efficient manner. Listed below is the recommended use of the various register sets for the HP-12C.

Input data is best stored in:

* The Financial Registers: **n, i, PV, PMT, FV**
  (Any data can be stored here, it need not be financial data.)

* Registers $\mathbf{R_5}$ and $\mathbf{R_6}$ and any of the unused registers reserved for program storage. (To determine the number of unused registers keystroke **g mem**. The number of unused registers will be displayed on the right side of the display while the total number of program lines used is displayed on the left.)

  Example:

  The above display indicates 71 programming lines have been used and 11 registers of the 20 registers R_0 through $R_{.9}$ are available for data storage. That is registers R_0 to $R_{.0}$ can be used for data storage.

Output data is best stored in:

* The Arithmetic Registers: $\mathbf{R_0}$, $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$
  (Good for output because of their arithmetic capability.)

* The Stack Registers: **X, Y, Z, T**
  (Good for ease of accessability.)

The best "*working*" registers are:

* The Stack Registers: **X, Y, Z, T**
* The Last X Register: **g LSTx**
* The Arithmetic Registers: $\mathbf{R_0}$, $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, $\mathbf{R_4}$

In diagram form we have:

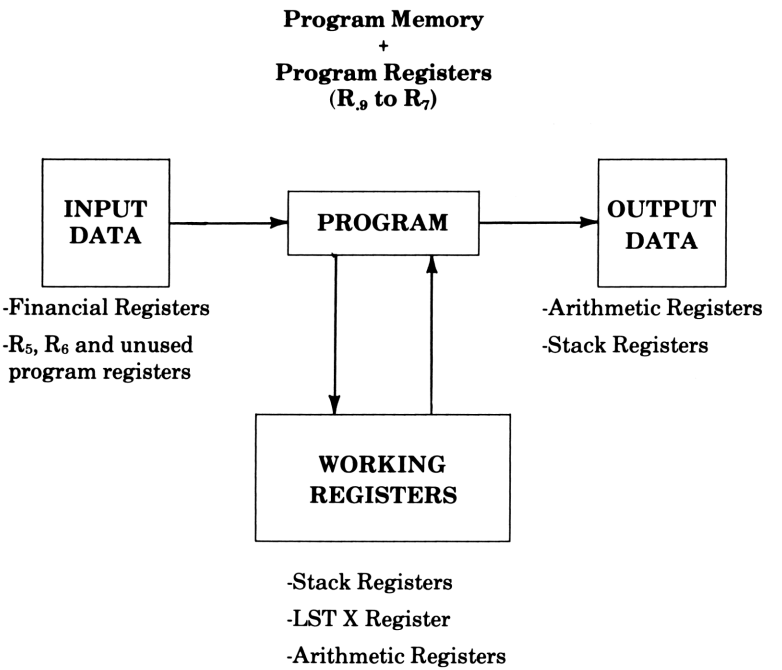


Figure 2 - Register Usage
This figure depicts the most effective usage of the HP-12C registers.

These are recommended usages and are not fixed. Note: There is an overlap on the usage of some registers and this overlap will depend on the user's experience and the user's program and how it is written. Experience, however, has shown the above usage to be most effective for minimizing program lines and for ease of running programs.

The Programming Process

In this section we discuss the *stages* of the programming process. They are:
(1) program development, (2) debugging the program, (3) running the program, and (4) editing the program.

The greatest amount of time will be devoted to the first stage; program development, since this is by far the most important and difficult part.

The *remaining* stages will be mentioned only for continuity and no in-depth discussion will be given. For an excellent presentation of these other stages of the programming process refer to your Hewlett-Packard Owner's Handbook.

Stage 1 - Program Development

Program development, besides being the most important, can be one of the most difficult stages of the programming process for a number of reasons. However, with an organized approach, this difficulty can be greatly reduced if not entirely removed. Often this aspect of the programming process is overlooked by many programming books and because of this PROGRAMMING HINTS FOR YOUR HP-12C was written. Presented below is an approach, which when used, will save time, aggravation and make program writing easy and enjoyable.

Five Steps to Easier Programming

Step 1) Write out a description of what you want the program to do. The more detail given in the description the less likely you will have to edit your program later.

Step 2) Write equations for what you want in terms of what you know by using easy to remember symbols.

Step 3) Assign registers to your desired output and to your known input symbols.

Step 4) Rewrite your equations using the assigned registers in place of the output and input symbols.

Step 5) With the aid of an example, input the known values into the assigned registers and execute the necessary steps outlined in your equations while recording each step. The recorded steps become the program instructions to your program.

Example:

Suppose you are a real estate agent and your clients are always asking you to determine the combined monthly payments for the mortgage, taxes, and insurance on the property they are about to purchase, a very likely and understandable request. Since this depends on many different things and because as soon as you tell them the amount for a 20% down payment you know they are going to ask about the amount for a 25% down payment and a different interest rate, you decide you are going to make it easy for yourself.

Since you would like to intelligently respond to your client's requests, you decide to program your HP-12C calculator to meet the challenge. Then by just keystroking a couple of numbers into your calculator and "pushing a button" you will have the answers to their questions in just seconds.

Although this is a simple situation it is one which will demonstrate the suggested approach to be taken when writing any size program for the HP-12C calculator.

Step 1) Write out a description of what you want the program to do. The more detail given in the description the less likely you will have to edit your program later.

Program Description

The program is to calculate the monthly payments made up of the following: (1) the payment on a new loan, (2) the payment for taxes (assumed to be a certain percentage of the purchase price), and (3) the payment for fire and homeowners insurance. It is to assume the down payment plus the loan will equal the purchase price (i.e. there will be no secondary financing). The program is to calculate the new loan amount when given the purchase price and the down payment and the user shall be able to view the loan amount by pressing the X ⇆ Y key after running the program.

The program is to be written in such a way as to allow the user to specify a different value for any INPUT and observe the new total monthly payment resulting from that new INPUT.

The only inputs required will be: (1) the purchase price, (2) the down payment, (3) the annual fire and homeowners insurance premium, and (4) the terms on the loan (i.e. the interest rate and the amortization period in years). Taxes will be calculated by the program.

Step 2) Write equations for what you want in terms of what you know by using easy to remember symbols.

EQUATION-1

$$\text{Total Mo. Pymts} = \underset{(1)}{\text{Mo. Pmt. on Loan}} + \underset{(2)}{\text{Mo. Pmt. for Taxes}} + \underset{(3)}{\text{Mo. Pmt. for Insurance}}$$

continued

LET: the following symbols be used.

(unknown)	**TMP**	=	Total Monthly Payment
(known)	**PP**	=	Purchase Price
	DP	=	Down Payment
	INS	=	Insurance premium (annual)
	PV	=	Present Value of the new loan amount
	i	=	Annual interest rate of the new loan
	n	=	Term of the loan in years

Also Define:

(calculated) **PMT** = Payment on the Loan (monthly)

then rewriting EQUATION-1 where:

(1) = **PMT** = A function of the loan amount (calculated in the program by taking the down payment from the purchase price), the interest rate, and the amortization period. This payment calculation will be made using the financial registers of the HP-12C by solving for PMT.

(2) = Mo. Pmt. for Taxes = **PP** × 1 % ÷ 12

(For this example we will assume the taxes to be 1% of the purchase price.)

(3) = Mo. Pmt. for Insurance = **INS** ÷ 12

And substituting the above equivalent representations for **(1), (2), and (3)**, into EQUATION-1 we obtain:

EQUATION-2

$$\mathbf{TMP} = (\mathbf{PMT}) + (\mathbf{PP} \times 1\% \div \mathbf{12}) + (\mathbf{INS} \div \mathbf{12})$$

At this point we have completed step 2 and we now want to assign the symbols in EQUATION-2 to the registers of the calculator.

Step 3)	Assign registers to your desired output and to your known input symbols.

For the **outputs** in this example:

assign:

Total Monthly Payment **TMP** to $\mathbf{R}_0$ (Register 0)

For the **inputs** in this example:

assign:

Purchase Price	**PP**	to $\mathbf{R}_1$	(Register 1)
Down Payment	**DP**	to $\mathbf{R}_2$	(Register 2)
INSurance	**INS**	to $\mathbf{R}_3$	(Register 3)
Payment Periods	**n**	to **g n**	(Financial Register)
Interest On Loan	**i**	to **g i**	(Financial Register)

The choice for register assignment is somewhat arbitrary except when a financial calculation using the financial registers is required. Since we will want our program to be able to amortize a loan for different loan amounts and different interest rates we assign the necessary loan data to the required financial registers (n, i).

At this point we have completed step 3 and we now want to rewrite EQUATION-2 using the registers of the calculator.

Step 4)	Rewrite your equations using the assigned registers in place of the input and output symbols.

Rewriting **EQUATION-2** in terms of the assigned registers we have:

EQUATION-3

$$\mathbf{R_0 = PMT + (R_1 \times 1\% \div 12) + (R_3 \div 12)}$$

The above step allows us to generate the keystrokes necessary to "solve" or "execute" this equation with the HP-12C.

At this point we have completed step 4 and we now want to use an example to generate us the necessary keystrokes.

Step 5)	With the aid of an example input the known values into the assigned registers and execute the necessary steps outlined in your equations while recording each step. The recorded steps become the program instructions to your program.

Suppose a property were to be purchased with the following terms:

Purchase Price	=	$183,250.00	
Down Payment	=	$ 35,250.00	
Insurance	=	$ 325.00	(annual premium)

continued

Loan Amount = (to be determined)

Loan interest rate is: 11.50%
Loan amortization term: 30.00 yrs.

Taxes are assumed to be 1% of the purchase price and will be calculated by the program.

First we want to keystroke the above data into the registers assigned in step 3.

Keystroke	Display	Comments
183,250 **STO** 1	183,250.00	Purchase Price
35,250 **STO** 2	35,250.00	Down Payment
325 **STO** 3	325.00	Annual Insurance
11.5 **g i**	0.96	Mo. Int. Rate
30 **g n**	360.00	# of Payments

Once all the required input data is keyed into the calculator we can begin to "key" and record the necessary steps required to bring this data to the X and Y registers (the "working" registers) where the necessary operations on the values can be performed. Note: Only five inputs are required in order to execute the program. Also, keep in mind we want to be able to change any input and run the program again for the new data.

The keystroking we do now will become our desired program.

A good understanding of how the stack registers and the financial registers work would be helpful. In the keystroking which follows it is assumed the reader has a working understanding of these registers. If the reader does not have a good understanding of the financial and stack registers and the ways in which to manipulate them, then a quick review of the Owner's Handbook is recommended.

The following keystrokes are required in order to bring the data from the storage location in which they are stored to the X and Y registers where the necessary operations will be performed on the data. Depending on what operation of our Equation (EQUATION-3) is performed first, one may create a different set of instructions. For simplicity we will work from left to right in the equation to generate the required keystrokes.

Advanced keystrokes and fancy register manipulations will be left as a topic for another article.

Keystroke	Display	Comments
0 **FV**	0.00	Future Value Of Loan At Term.
RCL 1	183,250.00	Purchase Price
RCL 2	35,000.00	Down Payment
—	148,000.00	Loan Amount
PV	148,000.00	Loads Loan Amount Into PV Register
PMT	148,000.00	Initializes The Financial Register
PMT (running)	-1,465.63	**Mo. Loan Pymt.**
RCL 1	183,250.00	Purchase Price
1 **%**	1,832.50	Yearly Taxes
RCL 3	325.00	Ins. Premium
+	2,157.50	Yearly Tax + Ins.
12 ÷	179.79	**Monthly Tax + Ins.**
RCL PMT	-1,465.63	Loan Payment
CHS	1,465.63	Changes Sign.
+	1,645.42	**Total Monthly Payment.**

Set up the program to display the loan amount by pressing **X ≶ Y**.

RCL PV	148,000.00	**New Loan Amount.**
X ≶ Y	1,645.42	**Total Monthly Pymt.**

When the program is through running the total monthly payment for the loan, taxes, and insurance will be showing in the display.

The above keystrokes represent a keystroke sequence necessary to bring the input data (from their storage locations) to the X and Y registers and perform the required calculations for computing the total monthly payment for the loan, the taxes, and the insurance.

Upon recording the above keystrokes we have completed the final step in the recommended approach for developing a keystroke sequence for a program.

Now that we have the program written we want to store the keystrokes of the program into the calculator's "memory". This process of "storing" the keystrokes in the calculator's memory is referred to as "loading the program".

LOADING THE PROGRAM

To load this program into your calculator:

(1) **Press f P/R** — Places calculator into the *program mode.*
(2) **Press f PRGM** — Clears Program Memory and Program Registers.
(3) Keystroke in the Program Instructions. When keystroking in the instructions, you should see the following:

Keystroke	Display	
0	01-	0
FV	02-	15
RCL 1	03-	45 1
RCL 2	04-	45 2
—	05-	30
PV	06-	13
PMT	07-	14
PMT	08-	14
RCL 1	09-	45 1
1	10-	1
%	11-	25
RCL 3	12-	45 3
+	13-	40
1	14-	1
2	15-	2
÷	16-	10
RCL PMT	17-	45 14
CHS	18-	16
+	19-	40
RCL PV	20-	45 13
X ≶ Y	21-	34
g GTO 00	22-	43,33 00

(4) **Press f P/R** (this returns the calculator to the *calculator mode*)

For a detailed explanation on programming your calculator refer to your HP-12C Owner's Handbook, Part II; Programming.

Your calculator should now be programmed. To execute the program, key the known input data into the assigned registers and press **R/S**.

The following is an example showing how to store the necessary input data and how to run the program you have just entered into your calculator.

Keystroke	Display	Comments
Storing the Input Data		
183,250 **STO** 1	183,250.00	Purchase Price
35,250 **STO** 2	35,250.00	Down Payment
325 **STO** 3	325.00	Annual Insurance
11.5 **g i**	0.96	Mo. Interest Rate
30 **g n**	360.00	# of Payments
Running the Program		
R/S (running)	1,645.42	Total Mo. Payment
Viewing the Loan Amount		
X ≶ Y	148,000.00	New Loan Amount

Changing the Inputs

Now when your clients ask you "what would our payments be if the down payment was . . ?" You simply key in the new down payment amount, store it in the appropriate register and press **R/S**. In just seconds you have the answer.

TO CHANGE:

The **Down Payment** - Key into the display the new down payment amount and **STO** the new value in Register 2.

The **Interest Rate** on the loan - Key into the display the new annual rate and press **g i**.

The **Term of The Loan** - Key into the display the new term of the loan in years and press **g n**.

The **Purchase Price** - Key into the display the new purchase price and **STO** the new value in Register 1.

The **Annual Insurance** amount - Key into the display the new insurance amount and **STO** the new value in Register 3.

To see the new monthly payment after any change is made press **R/S** and when "running" stops flashing in the display you will have the new total monthly payment displayed. To view any new loan amount press **X ≶ Y**.

In Table-1 below are some monthly payments derived by using this example program for different interest rates and down payments.

LOAN INTEREST RATE

DOWN PAYMENT	11.00%	11.50%	11.75%	12.00%
$35,250	1,589.23	1,645.42	1,673.72	1,702.14
$37,000	1,572.56	1,628.09	1,656.05	1,684.14
$40,000	1,543.99	1,598.38	1,625.77	1,653.28

Table-1
Monthly payments versus various down payments and interest rates for a fixed purchase price.

Table-1 above was derived with a minimum of keystrokes in a matter of just seconds per calculation using the program written for this example.

Stage 2 - Debugging The Program

There are three keystroke functions for the HP-12C which aid in debugging of programs; they are:

R/S
g PSE
SST

Both **R/S** and **g PSE** are used within the body of the program as "break points" to facilitate in the development of larger programs. Many times they are used in the final version of the program.

R/S - The Run/Stop instruction when executed within a program will stop the execution of the program. This is sometimes desired to check, change, or record an intermediate result of a program calculation. To resume program execution simply press **R/S**.

g PSE - The Pause instruction when executed will "flash" the contents of the X-register in the display for about a one second duration.

SST - The Single Step key allows the user to execute a program one program line at a time. This helps to determine where the errors are if the program does not properly run, by allowing the user to view the results of each executed program line.

For a greater detailed description of how each of these keys can best be utilized refer to your Owner's Handbook section on programming basics.

Stage 3 - Running The Program

To run what is called a "user written" program the calculator must be in the *calculator mode.* This is the mode you normally use every time you operate your calculator.

Three steps are involved in the running of every user written program:

> Step 1) You must input the required data into the proper storage locations required by the program you are about to run.
>
> Step 2) You must "position the calculator" to the beginning of the program you are about to run. If you only have one program in your calculator and it starts on program line #1 then "positioning the calculator" to the beginning of the program is done automatically for you whenever you turn on your calculator. If you have more than one program and you want to run a program which does not start at program line #1 you must "position the calculator" by pressing a **g GTO XX** where "XX" is the program line where your program begins.
>
> Step 3) The last step is to press the **R/S** key which starts the execution of the program.

Once the program has been written and stored it will remain in memory until "removed" by one of the following happenings:

1) While in "PRGM" mode you press **f PRGM**.
2) You reset memory by pressing **" — ON"** when the calculator is off.
3) A loss of power for an extended period of time.
4) Programs can be lost if you traumatize your calculator.

Stage 4 - Editing The Program

There are various reasons why you may want to modify a program you have stored in your calculator. It may be to correct an error, to insert new instructions, to delete instructions, or to change the program flow.

The instructions most frequently used in editing are:

SST g BST GTO

Those less frequently used are:

R/S PSE

SST - The **SINGLE STEP** Key - Pressing this key while in the *program mode* advances the calculator to the next line in program memory, then displays that line number and the keycode of the instruction stored there. If this key is pressed and held while in this mode the calculator displays, in sequence, all of the program lines stored in memory.

Pressing this key while in the *calculator mode* executes one line of program code. The line of code that is executed is the one where the calculator is "positioned" when this key is pressed. If you press and hold this key down while in the *calculator mode*, the calculator will display the program line and code for the instruction that will be executed when you release the key.

g BST - The **BACK STEP** Key - Pressing this key while the calculator is in the *program mode* sets the calculator back to the previous line in program memory, then displays the line number and the keycode of the instruction stored there. If pressed and held down the calculator will back space through all of the programming lines stored in memory.

Pressing this key while in the *calculator mode* will only reposition the calculator to the previous program line in memory. If pressed and held the previous program line will be displayed and when released the calculator will be positioned at this new program line.

GTO - The **GO TO** Key - This key is used to instruct the program or the calculator to go to a specified program line. This instruction can be used in both the *program mode* and the *calculator mode* by the user and also as an instruction by a program.

While in the *calculator mode* the user can direct the calculator to go to a specific program line by simply pressing **g GTO XX** where **XX** is the program line number. If the calculator is in the *program mode* and the user wishes to go to a different location the user must use a "." with this key, (i.e. **g GTO. XX**). The use of the period tells the calculator to reposition itself to the program line **XX** and not to store this as a program instruction. Without the use of the "." while in the *program mode* the calculator would store this keystroke as a program instruction.

Program editing is usually divided into three groups:

1) **Changing Instructions**
   - By overwriting an old instruction with a new instruction.

2) **Adding Instructions**
   - By appending new instructions to the end of the program.
   - By branching from within the body of the program to new code at the end.

3) **Deleting Instructions**
   - By replacing the instruction with a "no-op" instruction such as a digit or a series of digits.
   - By branching over the instructions.

The reader may wish to enhance the sample program given in this book to include seller financing, (i.e. the seller carries back some financing in addition to the lender's financing). This can be easily accomplished by executing the 5 steps outlined in this book.

SUMMARY

Although the example given is a simple example of a calculator program, its use as a vehicle to demonstrate the five-step approach to writing programs is adequate. Hopefully, upon the completion of this book the reader will have a clearer understanding of the process involved in writing a program for the HP-12C and will be encouraged, by the ease of so doing, to forge ahead and begin writing simple programs to utilize the much overlooked programming power of the HP-12C calculator.

Appendix

The purpose for this section is to show areas and ways in which programming lines can be minimized.

For every seven programming lines beyond the eight which use "program memory" an additional register (starting at $R_{.9}$ then going to $R_{.8}$ and continuing down through $R_{.0}$, and then R_9 through R_7) is required to store additional programming lines. Due to the way registers are allocated, it becomes very important to minimize program lines. Oftentimes the reduction of one line of the program will free a badly needed register.

For example: If a program has 23 programming lines then all of program memory is used plus three additional registers ($R_{.9}$, $R_{.8}$, $R_{.7}$) This would leave 17 registers (20 - 3 = 17) for data storage and in most cases would be more than enough. However; if you could reduce the number of program lines by just one, to 22, you would free a program storage register, which could then be used for data storage.

The allocation for the 23 programming lines above is as follows:

Program Memory	First 8 Lines
Register $R_{.9}$	Next 7 Lines
Register $R_{.8}$	Next 7 Lines
Register $R_{.7}$	Next 1 Line
	23 Lines

Since each register can store 7 programming lines, a program having 29 lines would require no more registers than a program having 23 lines. However, a program having 23 line requires one more register than a program having only 22 lines.

Thus one programming line can sometimes make a big difference if it provides an additional badly needed data storage register.

Minimizing Programming Lines

Remembering that sometimes just one line of code can save a register, the reader may want to study the code given below and experiment with other ways of accomplishing similar results.

For purposes of demonstration, comparisons and continuity it will be assumed the number one wishes to operate on is stored in register seven (R_7).

The g 12× and g 12÷ Keys

When you want to **Multiply** or **Divide** by 12:

A General Keystroke Sequence	An Improved Keystroke Sequence
RCL 7	RCL 7
1	g 12×
2	
×	
4 Lines	**2 Lines**

A General Keystroke Sequence	An Improved Keystroke Sequence
RCL 7	RCL 7
1	g 12÷
2	
÷	
4 Lines	**2 Lines**

Caution Note: When using the "**g 12×**" or the "**g 12 ÷** " key, the result is both stored in the respective **"n"** or **"i"** register and displayed in the display. It has not been **"ENTER"**"ed (i.e. it is not

in the Y-Register and *it will* "disappear" from the stack register set if another number is "keyed" into the display or if another number is **"RCL"**ed into the display. The result however is stored in the "n" or "i" and can be recalled for use when needed). You may store the resulting number in a register or use it in an arithmetic operation with the contents of the Y-Register or the contents of one of the Arithmetic Registers. Remember to be cautious with the displayed results when using the "g 12×" and the "g 12 ÷ " functions.

The previous code-saving technique is great if the contents of the **"n"** and **"i"** registers can be destroyed. But suppose you want to multiply the contents of R_7 by 12 and **not** destroy the contents of the **"n"** register.

A General Keystroke Sequence	**An Improved Keystroke Sequence**
RCL 7	RCL 7
1	i
2	RCL g 12÷
×	
4 Lines	**3 Lines**

Or suppose you wanted to divide the contents of R_7 and save the contents of the **"i"** register. Then:

A General Keystroke Sequence	**An Improved Keystroke Sequence**
RCL 7	RCL 7
1	n
2	RCL g 12×
÷	
4 Lines	**3 Lines**

Note: When using the **"RCL g"** with the **"12×"** and the **"12 ÷"** keys the reverse operation is performed.

Using the EEX Key

Many times you may be required to multiply, divide, add, or subtract with multiples of ten like 100, 1,000, or even 10,000. Since each digit requires a program line, a single 7 digit number would take an entire register.

With the **EEX** key programming lines can be minimized if you are dealing with these types of numbers. Consider:

Multiplying by 1000

A General Keystroke Sequence	An Improved Keystroke Sequence
RCL 7	RCL 7
1	EEX
0	3
0	×
0	
×	
6 Lines	**4 Lines**

Whenever you have a series of zeros you can save programming lines when you use the **EEX** key to designate the number of zeros.

Using the Percentage Keys

Many times with a little thought, the use of the Percentage keys will allow you to minimize some program code.

Dividing by 50

A General Keystroke Sequence	An Improved Keystroke Sequence
RCL 7	RCL 7
5	2
0	%
÷	
4 Lines	**3 Lines**

Subtracting 100 From a Number

A General Keystroke Sequence	An Improved Keystroke Sequence
RCL 7	RCL 7
1	RCL 7
0	%
0	Δ %
—	
5 Lines	**4 Lines**

Note: If the Y-Register is **non-zero**, then:

RCL 7
%
Δ %

3 Lines

Others may exist and certainly these do not always have a use in a program. The important thing here is that with a little creative effort you may be able to minimize some code and write yourself a program that does more in a smaller amount of space.

Conditional Testing and Branching

When writing a program it becomes necessary at times to change the flow of the program because of some condition (i.e. you may want to reduce the price on an item if the quantity ordered exceeds a certain limit).

Program flow is changed by "branching". Two types of branching exist in programming. These are **unconditional** and **conditional** branching.

Unconditional or direct, as it is sometimes called, is done with a **"g GTO XX"** command where "XX" is the program line number of the next instruction to be executed.

Conditional branching is a direct branch which is based upon a condition. The decision to branch is made by "testing" a condition resulting from comparing one number with another.

The HP-12C calculator has two instructions for which to "test" a number. The number tested is always the number in the X-Register. These instructions are:

X ≤ Y	Is X less than or equal to Y?
X = 0	Is X equal to zero?

Branching decisions in programming have usually been done by testing for one of the following six conditions which can exist between any two real numbers.

The question is asked:

is the number in the **X-Register**,

1) = equal to — **(E)**
2) X ≤ Y less than or equal to — **(LTE)**
3) < less than — **(LT)**
4) X ≥ Y greater than or equal to — **(GTE)**
5) > greater than — **(GT)**
6) ≠ not equal to — **(NE)**

the number in the **Y-Register** ?

Following are sets of program keystrokes necessary to generate the six conditional tests mentioned above. The code given is for testing the contents of the X-Register while maintaining its integrity.

In all cases if the "condition" tests "true" the program flow will go to program line #30. If the "condition" tests "false" the program flow will go to program line "XX". Line "XX" can be anywhere in the program.

Keystrokes for Conditional Testing

#1- Condition Test: X = Y ?

	Keystroke	Comments
*		
*		
*		
25	g X ≤ Y	is X LTE to Y ?
26	X ⇆ Y	exchange X for Y Do if last test is True
27	g X < Y	is X LTE to Y ? Do if last test is T or F
28	g GTO 30	Do if last test is True
29	g GTO XX	Do if last test is False
30	- - - - - - - -	any statement
*		
*		

Or another way for the same test as above.

	Keystroke	Comments
*		
*		
*		
26	—	subtract X from Y if = then the display = 0
27	g X = 0	is X = 0 (is the display = 0 ?)
28	g GTO 30	Do if last test is True
29	g GTO XX	Do if last test is False
30	g LSTx	Restores original value to X
*		
*		

Keystrokes for Conditional Testing *Cont'd.*

#2- Condition Test: X LTE Y ?

	Keystroke	**Comments**
*		
*		
*		
27	g X ≤ Y	is X LTE to Y
28	g GTO 30	Do if True
29	g GTO XX	Do if False
30	--------	any statement
*		
*		

#3- Condition Test: X LT Y ?

	Keystroke	**Comments**
*		
*		
*		
26	X ⇆ Y	exchange X for Y
27	g X ≤ Y	is X′ LTE Y′ ? (i.e. is Y GT X ?)
28	g GTO XX	Do if True
29	X ⇆ Y	Do if False - restores X to original value
30	--------	any statement
*		
*		

Keystrokes for Conditional Testing *Cont'd.*

#4- Condition Test: X GTE Y ?

	Keystroke	Comments
*		
*		
*		
26	X ⇆ Y	exchange X for Y
27	g X ≤ Y	is X' LTE Y' ? (i.e. is X GTE Y ?)
28	g GTO 30	Do if True
29	g GTO XX	Do if False
30	X ⇆ Y	Restores X to the original value
*		
*		

#5- Condition Test: X GT Y ?

	Keystroke	Comments
*		
*		
*		
28	g X ≤ Y	is X LTE Y ?
29	g GTO XX	Do if True
30	--------	Do if False (any statement)
*		
*		

Keystrokes for Conditional Testing *Cont'd.*

#6- Condition Test: X NE Y

	Keystroke	Comments
*		
*		
*		
27	-	subtract X from Y if = then display = 0
28	g X = 0	is X = 0 ? (is the display = 0 ?)
29	g GTO XX	Do if True
30	g LSTx	Do if False - restores original value to X
*		
*		

If the contents of the X-Register can be altered (not preserved) then the keystroking for the above testing could be simplified in a couple of cases.

The use of conditional branching allows a program to do many different things depending on the results of testing the relationship between two numbers or the result of some arithmetic operation performed between two numbers. In essence, a program having conditional branching can be viewed as a program having a series of smaller subprograms. Then, based on some "condition test" the main program will determine which "subprogram" is to be run.

$9.95

CHARY

21213-B Hawthorne Blvd.
Suite 5120
Torrance, CA 90509

"A leader in development of calculator software."

PH-0995.7500 Rev. A 08/86

Printed in U.S.A.