hp calculators

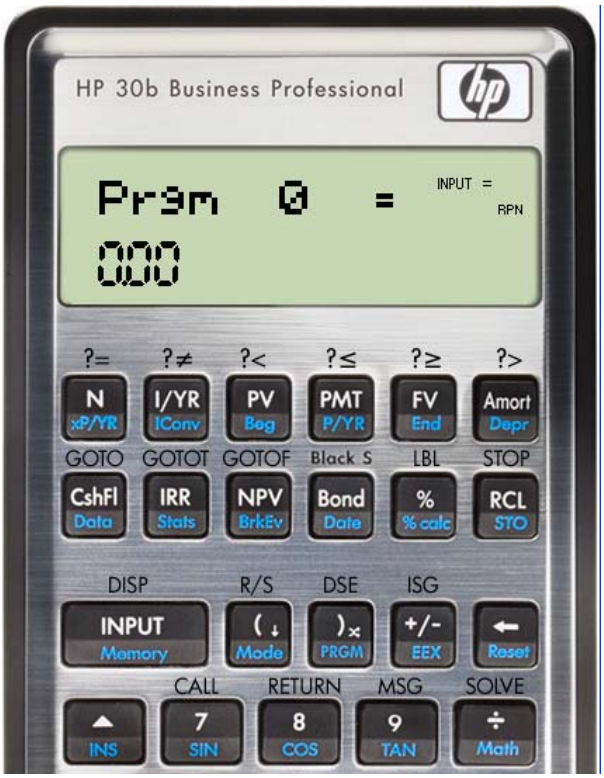
HP 30b Programming - reference

Programming on the HP 30b

The HP 30b Business Professional calculator includes a programming capability designed to help automate repetitive calculations and extend the usefulness of the built-in function set of the calculator. The capability includes the creation of up to 10 separate programs using up to 290 bytes of memory among them.

Programs record keystrokes, with each keystroke using one byte of memory, although some commands use more than one byte, as described later. In addition, many program-only functions are provided for conditional tests, conditional and unconditional "gotos", looping, displaying intermediate results and even calling other programs as subroutines.

This learning module will cover the HP 30b programming environment in some detail and provide an explanation of the program-only functions and other information needed as a reference for writing programs on the HP 30b. Other learning modules will show how to enter and edit programs, how to automate short, repetitive tasks, loop and call subroutines as well as showing several example programs to help get you started.

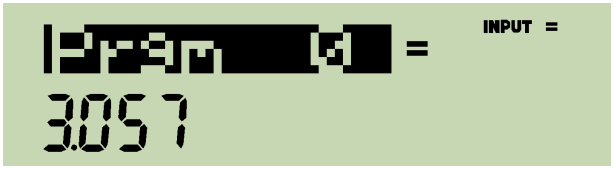


As shown in the picture at right, the HP 30b has additional functions assigned to the keys that are program-only functions. Other than the Black-Scholes function (shown as Black S), which is not a program function but a financial function, these functions are not printed or labeled on the actual HP 30b itself. However, an overlay is provided that lays over the top rows of keys that help indicate how these functions are mapped to the keys.

Each of these functions is inserted into a program by pressing the shift key and holding it down while pressing the key under which the program function is displayed. For example, to insert a LBL (label) command, press [shift] and, while holding it down, press [%]. In these learning modules describing programming, this will be shown as [shift] + [%]. Pressing that key combination will insert a LBL instruction into a program in program edit mode. Pressing that key combination in calculation mode will do nothing.

There are 10 numbered slots available for programs, numbered from 0 to 9. These are displayed in the program catalog which is viewed by pressing [shift] [PRGM]. In the image above, the program catalog is displayed, showing Prgm 0 or program 0. Pressing the [▲] or [▼] keys will scroll through the list of 10 programs. Pressing [INPUT] will enter the selected program, allowing you to view the program steps stored in that program slot or to change the program steps. To exit this program editing mode and return to the program catalog, press [shift] [PRGM] To exit the program catalog and return to calculation mode, press [ON/CE].

When a program is displayed, a number will be shown below it indicating how many bytes are used. If the program name is shown in reverse video, then the program has been assigned to a key and can be executed by pressing the appropriate key combination, even when in calculation mode. This is shown in the image at right. When viewing a program in the program catalog, pressing [shift] [Reset] [INPUT] will delete the presently displayed program and return you to the calculation environment. To delete all programs, press [shift] [Reset] [▲] [▲] [INPUT] [INPUT] while in calculation mode.



At different places within a program, you can insert a Label (LBL) command. A label defines a location to which program control may be transferred. The HP 30b can handle up to 100 labels within the entire program memory. These labels are a two-digit numeric value from 00 to 99. No label can be used more than once, which makes each label a “global” label and defined only once within the global program memory space. If you attempt to enter a label that has already been used, a message saying “Exists!” will be briefly displayed.

Programming functions on the HP 30b

The programming functions provided include 6 conditional tests, as shown in the table below. For conditionals, if the test returns a true result, a value of 1 is pushed to the stack in RPN mode. If a test returns a false result, a value of zero is pushed. This result can then be used by the “go to” commands described below to alter program flow based upon a test performed by the program.

Notes: 1) These commands test the value in Y against the value in X, which is the way HP’s RPL-based graphing calculators work but the opposite of the way conditional tests have worked on past models. It is very important to notice this difference. 2) In RPN mode, the original contents of stack levels Z, Y and X are pushed one level higher into the stack when the conditional test returns a 0 or 1 as a result. The original contents of stack level T are lost when a conditional test is executed. If stack level T is needed, be sure to save it in a memory register before executing a conditional test. 3) In algebraic and chain modes, an = is required after the X value to perform the comparison. This equal will end all pending operations, so be sure that a conditional test in algebraic or chain mode is not performed with any pending operations.

Conditional tests

Symbol	Description
?=	Y equal to X? In RPN mode, usage is Y X ?= to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?= X = to compare the value in Y to the value in X. If they are equal, pushes a 1 to X. Otherwise, pushes a 0 to X.
?≠	Y not equal to X? In RPN mode, usage is Y X ?≠ to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?≠ X = to compare the value in Y to the value in X, with the final = required to perform the comparison. If they are not equal, pushes a 1 to X. Otherwise, pushes a 0 to X.
?<	Y less than X? In RPN mode, usage is Y X ?< to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?< X = to compare the value in Y to the value in X, with the final = required to perform the comparison. If Y is less than X, pushes a 1 to X. Otherwise, pushes a 0 to X.
?≤	Y less than or equal to X? In RPN mode, usage is Y X ?≤ to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?≤ X = to compare the value in Y to the value in X, with the final = required to perform the comparison. If Y is less than or equal to X, pushes a 1 to X. Otherwise, pushes a 0 to X.
?≥	Y greater than or equal to X? In RPN mode, usage is Y X ?≥ to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?≥ X = to compare the value in Y to the value in X, with the final = required to perform the comparison. If Y is greater than or equal to X, pushes a 1 to X. Otherwise, pushes a 0 to X.
?>	Y greater than X? In RPN mode, usage is Y X ?> to compare the value in Y to the value in X. In algebraic or chain mode, usage is Y ?> X = to compare the value in Y to the value in X, with the final = required to perform the comparison. If Y is greater than X, pushes a 1 to X. Otherwise, pushes a 0 to X.

There are three “go to” commands provided for use by the HP 30b. A GOTO command simply transfers control to the specified label. The GOTOT and GOTOF commands evaluate the number on the stack to determine whether to transfer control to the specified label or to continue with the next program step. These commands are described in the table below.

NOTE: The GOTOT and GOTOF commands consume their arguments off the stack in RPN mode, whether the argument indicates true or false and whether the command actually transfers program execution to a label or not. If the stack were holding a 0 value, a program that executed a GOTOF command would not only jump to the specified label, but would drop the zero from the stack as well! If the stack were holding a non-zero value, a program that executed a GOTOT command would not only jump to the designated level, but would drop the non-zero value from the stack. The GOTO command does not consume an argument from the stack. In chain and algebraic mode, the argument is not removed.

Goto functions

Function	Description
GOTO	Go to the label specified. This is an unconditional jump to the label location. Displayed as Gto.
GOTOT	Go to the label specified if the value in X is non-zero. A conditional test returns a result of 1 if the conditional is true, but the GOTOT command jumps to the specified label for ANY non-zero value. Displayed as GT.
GOTOF	Go to the label specified if the value in X is zero. A zero is a false result from a conditional. Displayed as GF.

There are two commands that allow programs to loop or repeat a series of steps until a certain condition is met. These are Increment and Skip if Greater (ISG) and Decrement and Skip if Equal (DSE). These are described in the table shown below. Although not required, these looping functions are often used with a “go to” instruction in order to loop. It is also acceptable to place any other instruction after the “loop” function. That instruction will either be executed or skipped.

Looping functions

Function	Description
DSE	Decrement and Skip if Equal. Takes a memory register from 0 to 9 as an argument. Evaluates the value contained in the register for the form ccccc.eeeii, where ccccc is the value of the counter, eee is the ending value, and ii is the increment value. It decreases the value of ccccc by the value of ii, stores the result back into the memory register, and then compares it to the value eee. If the new value for ccccc is equal to or less than eee, the next program step is skipped. The default value for ii is 1. For example, if the value in memory 0 is 15.00802, a DSE 0 command would subtract 2 from integer 15 portion of memory 0 and store the result back into memory 0. It would then compare the integer portion in memory register 0 (which is now 13) to the value 8. Since 13 is not equal to or less than 8, the next program step would be executed. For another example, if memory register 0 held the value 5.00400, the default value used for ii to decrease the memory register’s contents would be 1. A DSE 0 command would subtract 1 from the contents of memory 0, store result into memory register 0, and then since 4 is equal to the .004 portion of memory 0, the next program step would be skipped.

Looping functions

Function	Description
ISG	Increment and Skip if Greater. Takes a memory register from 0 to 9 as an argument. Evaluates the value contained in the register for the form ccccc.eeeii. It increases the value of ccccc by the value of ii, stores the result back into the memory register, and then compares it to the value eee. If the new value for ccccc is greater than eee, the next program step is skipped. The default value for ii is 1. For example, if the value in memory 1 is 5.00804, an ISG 1 instruction would add 4 to the integer portion 5 of memory 1 and store the result back into memory 1. It would then compare the integer portion in memory register 1 (which is now 9) to the value 8. Since 9 is greater than 8, the next program step would be skipped. For another example, if memory register 0 held the value -3.00000, the default value used for ii to increase the memory register's contents would be 1. The ISG 0 command would add 1 to the integer portion of memory 0, -3, and store the result back into memory register 0, and then since -2 is not greater than the 000 portion of memory 0, the next program step would be not be skipped.

HP 30b programs can call and use subroutines from other program locations. A subroutine may be used if the series of program steps is repeated several times to save space or perhaps if a section of code has already been keyed in and tested, to allow for a saving of time by not having to key in the code again.

The HP 30b allows up to 4 levels of subroutines. Attempting to call a fifth subroutine level will result in an error. The two subroutine functions are described below.

Subroutine functions

Function	Description
CALL	Transfers control to the label specified as a subroutine. The subroutine label can be in the current program or in any other program. Uses 2 bytes of program memory.
RETURN	Returns control to the step after the CALL instruction.

The HP 30b provides two program functions to provide feedback to the user. These are described in the next table.

Feedback functions

Function	Description
DISP	Displays the number in the X register. Expects an argument from 0 to 9. Displays value and pauses program execution for a period of time equal to the argument multiplied by 200 milliseconds. A value of 0 does display the X register, but with no execution delay. If preceded by a MSG command, pauses to display the message for the same period of time. Former HP-41 users will find a similarity to the VIEW command from that model in this function.

Feedback functions

Function	Description
MSG	Allows for the entry of a text message to be displayed by a running program. Enters a character selection mode where the [▲] and [▼] keys move through the available character set. Pressing [INPUT] selects the presently displayed character and moves the cursor one location to the right to allow for another character to be entered. Once [INPUT] has been pressed to select the first letter, the next letter begins with lowercase a letter. Pressing [▬] + [▼] will skip ahead 10 characters and pressing [▬] + [▲] will skip backward 10 characters. Press [ON/CE] to stop entering text. To delete a character entered while in the process of entering letters, press [←]. To move the character position to a numeric digit as a letter in a message, press the corresponding numeric key. To move to an =, press [=]. Pressing [·] moves to the ? character. Pressing [+/-] moves the character position to a space. The arithmetic operators move the character position to the corresponding operator character. Press [INPUT] to select the displayed character and prepare to enter another character. Press [ON/CE] to select the displayed character and terminate character entry. A maximum of 8 characters may be entered as a message. Using a MSG immediately before a R/S instruction allows for a prompt to the user. A MSG might say “Rate?” and be a prompt to a user to enter an interest rate. If a MSG is present as the first line of a program, it is used as the program’s title and displayed in the program catalog. This may be useful as a memory aid. Note that this command uses 2 bytes of program memory plus one byte for each character in the message. When a MSG is present in the display, the small = annunciator is turned on in the display. This will help differentiate a MSG of the letters SIN from a program instruction SIN.

The HP 30b provides two functions which control program execution. One will terminate a running program completely while the other stops program execution, allowing the program to be resumed after user action, such as an input or calculation.

Program execution control functions

Function	Description
R/S	When encountered in a program, stops execution at that step. When pressed from the keyboard, resumes program execution (assuming that a program has previously stopped because a R/S was encountered in a program) at the step following the previous R/S instruction.
STOP	Terminates program execution immediately. Does not allow for the continuation of program execution using a keyboard supplied R/S. This step should be used in all programs to ensure a program terminates properly. It does not have to be the last step in a program, but it is the proper way to terminate execution of a program.

Certain HP 30b non-program keyboard functions operate differently when entered in a program. The following table explains their use and any special conditions that apply. Of special note, these functions are entered into a program using the shift-hold method.

Non-program keyboard functions

Key	Description
[ON/CE]	Exits program mode and returns to calculation mode. Performs this action whether viewing programs in the program catalog of if pressed while editing / viewing a program's steps. Note: To insert an [ON/CE] into a program as a program step, press [▬], hold it down and then press [ON/CE] at the same time. If inserted into a program, "ON" is displayed for the program step.
[▲]	While viewing the program catalog, moves up to the previous program. While editing a program, moves up to the previous program step. When pressed in RPN mode with no menu active, performs a stack roll up. Note: To insert an [▲] into a program as a program step, press [▬], hold it down and then press [▲] at the same time. This is often required to access an HP 30b function stored in a menu. If inserted into a program, "Up" is displayed for the program step.
[▼]	While viewing the program catalog, moves down to the next program. While editing a program, moves down to the next program step. When pressed in RPN mode with no menu active, performs a stack roll down. Note: To insert a [▼] into a program as a program step, press [▬], hold it down and then press [▼] at the same time. This is often required to access an HP 30b function stored in a menu. If inserted into a program, "Down" is displayed for the program step.
[←]	While viewing the program catalog, begins editing the displayed number, just as it would in calculation mode. While viewing program steps, deletes the presently displayed step. Note: To insert a [←] into a program as a program step, press [▬], hold it down and then press [←] at the same time.

Program instruction byte usage

Functions placed into a program generally use one byte each of the available memory. There are a few functions that use more than one, however. These are shown below.

Program instruction byte usage

Key	Description
[▬] [STO] [RCL]	Store and recall use two bytes each.
[▬] [STO] [▬] [Mode] [RCL] [▬] [Mode]	Store and recall mode use two bytes each.
[▬] + [)] [▬] + [+/-]	Decrement and skip if equal (DSE) and increment and skip if greater (ISG) both use two bytes.
[▬] + [INPUT]	Display a value (Disp) uses two bytes.

Program instruction byte usage

Key	Description
[↓] + [7]	Call (to call a subroutine) uses two bytes.
[↓] + [CshFl]	Unconditional go to instructions use two bytes.
[↓] + [IRR]	Go to if true instructions use two bytes.
[↓] + [NPV]	Go to if false instructions use two bytes.
[↓] + [%]	Each label uses two bytes.
[↓] [STO] [+] [↓] [STO] [−] [↓] [STO] [×] [↓] [STO] [÷] [RCL] [+] [RCL] [−] [RCL] [×] [RCL] [÷]	Storage register arithmetic and recall register arithmetic use three bytes each.

Store and recall MODE

The MODE menu contains settings which control the way in which the HP 30b operates. In addition to changing these settings in the menu itself, it is possible to recall these settings as a number as well as storing a different number which will change the MODE menu settings. It is possible to change from algebraic mode to RPN mode by storing a three-digit number, for example. This can often be faster than changing the setting in the MODE menu and is very useful within a program. It also allows a program to save the current MODE settings when it begins execution, change them if it needs to do so, and return the user's original settings when it finishes.

From a fully reset HP 30b, pressing [RCL] [↓] [Mode] will display the following:

These digits represent the default settings. Interpreted from right to left, these are:

2 decimal places shown, chain operating mode, degrees angle mode, 2 variable statistics, linear regression, mm.ddyyyy date mode, radix point of ".", thousands separator shown, actual day mode, annual mode, TVM standard mode.

This is explained in the following table.

Meaning of digits stored in MODE

Digit position	Values and description
1st and 2nd digits from right	These two digits indicate the display format chosen. 0-11 display 0-11 decimal places 12 displays in “best” or “floating” format
3rd digit from right	1 indicates Chain mode 2 indicates Algebraic mode 3 indicates RPN
4th digit from right	1 indicates radians mode. 2 indicates degrees mode.
5th digit from right	1 indicates 2 Var statistics mode. 2 indicates 1 Var statistics mode. 3 indicates 1 Freq statistics mode.
6th digit from right	Digits indicate regression model selected. 1 indicates a*X+b 2 indicates a*ln(X)+b 3 indicates b*e^(aX) 4 indicates b*X^a 5 indicates b*a^X 6 indicates a/X+b 7 indicates aX^2+bX+c
7th digit from right	1 indicates mm.ddyyyy date mode. 2 indicates dd.mmyyyy date mode.
8th digit from right	1 indicates radix point shown is “.” 2 indicates radix point shown is “,”
9th digit from right	1 indicates no thousands separator shown. 2 indicates thousands separator shown.
10th digit from right	1 indicates Actual day count mode. 2 indicates 360 day count mode.
11th digit from right	1 indicates Semi-Annual mode. 2 indicates Annual mode.
12th digit from right	1 indicates TVM Standard mode. 2 indicates TVM Canada mode.

Consider these examples of how to change the settings using ▬ STO ▬ Mode.

Example 1: Assuming the HP 30b is in chain or algebraic mode, set RPN mode and do not change the display format or any other settings. One could press:

3 1 4 ▬ STO ▬ Mode

The effect of this would be to place a 3 in the 10th digit from the left. This would set RPN mode. The 14 in the last two digits is an invalid value for those digit positions and no change to them would be made. If 300 were stored, it would change to RPN mode but would set FIX 0 as well. Storing 114 would change the mode to Chain and storing 214 would change the mode to Algebraic.

Example 2: Make sure the HP 30b is in chain mode, FIX 2 and ACTUAL day mode. Do not change any other modes. One could press:

1 9 9 9 9 9 1 0 2 ▬ STO ▬ Mode

The effect of this would be to place a 1 in the 4th position from the left, setting ACTUAL days mode, a 1 in the 10th digit from the left, setting chain mode and 02 in the last two digits, setting FIX 2. The values of 9 are invalid values for the corresponding MODE positions and no changes would be made to them.

Example 3: Extract the statistics mode (whether 2 Var, 1 Var or 1 Freq) and display it as a 1, 2 or 3.

RCL ▬ Mode 1 0 0 0 0 ÷ ▬ Math ▲ = 1 0 ×

This would place a 1, 2 or 3 in the display. This would be particularly helpful in a program to extract items from the Descriptive sub-menu within the STATS menu, since the number of ▼ key presses required to get to the mean of X, etc, will vary depending on the statistics mode due to the suppression of the calculated Y-values if the HP 30b is in 1 Var mode.

Making program key assignments

The HP 30b provides the ability to assign a program to a key. Once assigned to a key, pressing the key will automatically run the program. Suppose you wanted to assign a program to the ▬ RAND key position. To make this assignment, enter the program catalog by pressing ▬ PRGM.

Note: If you assign this program to the ▬ RAND key position, the random number function is ONLY available if you clear the program key assignment or if you press ▬ + 1 (shift-hold 1) while in the calculation environment (since the random number function is the shifted function of the 1 key). If the shift-hold RAND position (▬ + 1) has ALSO been assigned to a program, the original functionality of the key position will be unavailable until one of the key assignments has been cleared. To clear an assignment, press ← when program step 0 is displayed.

The program catalog is displayed showing program 0. Press INPUT to begin editing the program. Now press ▲. This takes the program listing to step 0, as shown below.

The HP 30b will automatically assign a program to the key represented by the step entered at step 0. With step 0 shown, press [shift] [RAND] to assign this program to that key position on the keyboard. Note that the program step location did not advance when the program assignment was entered – it stayed on step 0, as shown below.

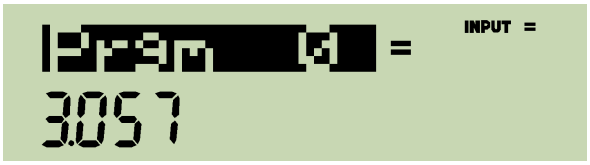
Now press [▼] [ON/CE] to exit the program editing environment. To run the program, press [shift] [RAND]. Programs can be assigned to primary key positions (so that the program executes when the key is pressed by itself), to shifted key positions (so that the program executes when you press shift and then the key) and to shift-hold key positions (so that the program executes when you press and hold shift and then press the key, with the following exceptions:

Every unshifted key is assignable, except: [ON/CE] [INPUT] [←] [▲] [▼]

Every shifted key is assignable, except: [shift] [Reset] and [shift] [PRGM]

Every shift-hold key is assignable.

When a program has been assigned to a key, the program name will appear in “reverse video” in the program catalog, as shown below.



Single-stepping a program

When a program is shown in the program catalog, pressing [=] will execute the program (if you have just entered a value as an input to the program, pressing [=] [=] will execute the program).

If you wish to execute the program one step at a time, press [shift] + [▼], where you press and hold the shift key and then press the down arrow key. If you do this quickly, the next program step is executed and control is returned to the user. If you hold the [▼] key down for a slightly longer time, the next step to be executed will be briefly displayed on the top row of the screen. This ability can be quite useful when trying to determine why a program is not executing exactly as expected.

If simply wish to view the next program step but not execute it yet, press [shift] + [▲], This ability can also be quite useful when you wish to see what the next program step executed will be.

Program checksums

Each program has a checksum, useful to confirm proper entry of a program. This checksum is displayed as a three digit decimal value after the byte count of the program when viewing the program in the program catalog. The checksum is

based upon the sum of the internal code for each program byte multiplied by one plus the remaining number of bytes in the program.

HP Solve

One function written on the HP 30b overlay and shown on the image of the HP 30b program functions earlier in this document is labeled SOLVE. This function is non-programmable but operates on a program to solve for one variable in a properly constructed program. This use of the HP Solve feature is quite powerful and is described in its own learning module.

Accessing Ans1, Ans2, Ans3 and Ans4 in chain and algebraic mode

The HP 30b has 4 registers that hold numbers in RPN mode, described as stack levels X, Y, Z and T. When in chain or algebraic mode, these registers are labeled Ans1, Ans2, Ans3, and Ans4. These registers contain the last 4 values computed when [=] was pressed. For example, from a cleared HP 30b in chain mode, press:

[1] [+] [2] [=]
[3] [×] [2] [=]
[4] [−] [2] [=]

At this point, Ans3 will contain the value of 3, Ans2 will contain the value of 6 and Ans1 will contain the value of 2. To use one of these previous results in another calculation, press the [▲] and [▼] keys to move to the location desired, press [RCL] [=] to recall the value stored in the result register, and then perform the calculation desired. For example, with the previous calculations just performed, to take the result of 1+2= and multiply it by the result of 4-2, press:

[▲] [▲] [RCL] [=]	This recalls the result in Ans3 (the result of 1 + 2 =)
[×]	Begins the multiplication
[▲] [▲] [RCL] [=]	This recalls the result in Ans1 (the result of 4 – 2 =)
[=]	And computes the result, showing a value of 6.

This capability allows for the entry of arguments into these registers before executing a program. If a program needs 3 values, key the values separated by [=], and then execute a program that pulls the values from the Ans registers as shown above.

Available program memory

As stated earlier, the HP 30b has 290 bytes available for programs. To view the amount used for programs, press [▬] [Memory] [▲] [▲]. The amount of available memory is the difference between 290 and the number shown. In the sample screen below, 274 of the available 290 bytes are being used for programs, leaving 16 bytes available.

Conclusion

Other learning modules explain how to enter and edit programs, how to automate repetitive tasks, and how to use looping and subroutines. Still another module provides examples of already written programs to illustrate how HP 30b programming works. The programming capability of the HP 30b can provide a very helpful extension of the already powerful function set built into the HP 30b.