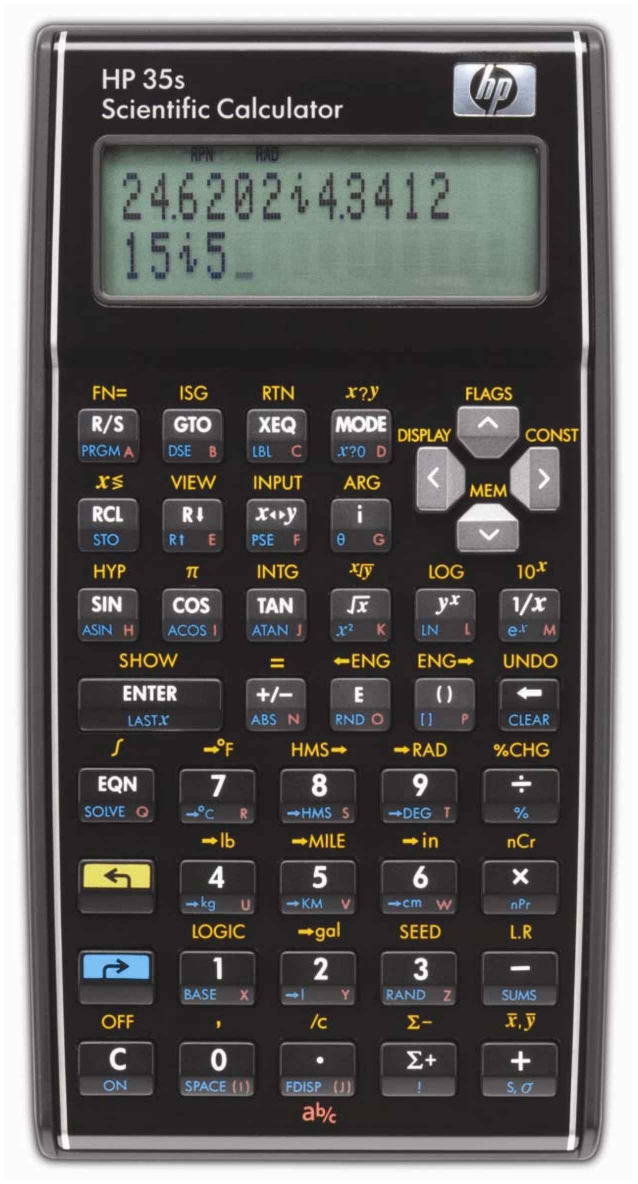
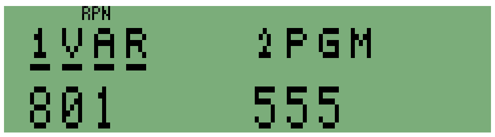
hp calculators

HP 35s Using the indirect registers



The HP 35s and indirect registers

The HP35s contains registers or variables that can be referenced directly or indirectly. Variables A through Z can be directly addressed, as in a ↱ STO A instruction. Indirect addressing uses two of these direct variables as indices that hold the location or address where an operation is to be performed. The two variables that are used this way are I and J. The indirect registers begin at address 0 and can go up to 800, if the user allocates that many. That is 801 additional storage registers compared to the earlier HP 33s calculator. Figure 1 shows the HP 35s display when a non-zero value has been stored into indirect register 800 (which would allocate 801 indirect registers). NOTE: the indirect registers are allocated by the highest-numbered, non-zero indirect register in use. If the highest used register is cleared or has a zero stored into it, the allocation will shrink down to the next highest non-zero indirect register.


```
RPN
1VAR     2PGM
801      555
```
Figure 1

Each allocated indirect register uses 37 bytes of program memory. This is to allow room for storing a real number, a complex number, or a 2-D or 3-D vector containing 2 or 3 real numbers. It is possible to use a short program to place 3 real numbers into each indirect register, reducing the memory required by nearly 2/3. See the Indirect Registers Data Packing module for more information.

It is also possible to address the direct variables and the statistics variables indirectly using addresses of -1 through -32. Address -1 would refer to the direct variable A, address -26 would refer to the direct variable Z, and -27 trough -32 would refer to the statistical summation registers. This is shown in a table on page 14-22 of the HP 35s user's guide.

The way indirect addressing works is to store the number corresponding to the register you wish to use in either I or J. Then you perform a ↱ STO (I) or ↱ STO (J) (or any other allowed operation). For example, if you wish to recall a value stored in direct register A, you can either press RCL A or store -1 into I by ↱ STO (I) and then perform a RCL (I). Both will recall the value stored in A.

As an example, suppose register I contains the number 5. If you have a number in the display you wish to store and you press ↱ STO (I), then the number displayed would be stored into indirect register 5, since that is the number that was in the I register. The J register works the same way. The address to be used is stored into either I or J, and then the special (I) or (J) part of an instruction tell the HP 35s to perform the operation on the register pointed to by the value in I or J. If you attempt to recall or use a value in an indirect storage location that is not allocated, you will get an error message in the HP 35s display.

This becomes very useful is when you need to work with a lot of numbers, often within a program, or when you may not be able to know in advance where the number you wish to use is stored.

Differences from the HP 33s

On the HP 33s, a special register was used as the index for indirect operations, the register i. On the HP 35s, two registers can be used as two different indices, the variables I and J.

On the HP 33s, the variables A through Z were referenced indirectly by values placed in register i of 1 to 26 in order. On the HP 35s, the variables A through Z are referenced by values in either I or J of -1 down to -26, with -1 referencing A and -26 referencing Z. Similarly, the statistics registers were referenced indirectly on the HP 33s using values of 28 through 33. On the HP 35s, they are referenced using an index of -27 through -32. On the HP 33s the index register was itself referenced by a value of 27. Since the index registers on the HP 35s are the variables I and J, there is no specialized index register.

On the HP 35s, the indirect registers are cleared using the CLVARx function, found in the ↱ CLEAR menu. Press ↱ CLEAR ∨ and the display below will be shown. CLVARx is menu choice 6.

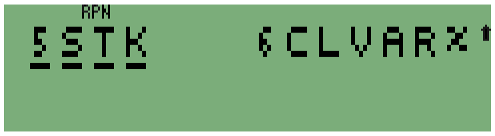

```
RPN
5STK     6CLVARx
```

Figure 1

The CLVARx function prompts for a 3-digit value. The function will clear all indirect registers larger than this value – they will be set to zero. Because the HP 35s allocates indirect registers based on the highest numbered register containing a non-zero value, this will change the number of allocated indirect registers.

For example, if there are 100 indirect registers allocated, CLVARx 050 will clear and deallocate the indirect registers 50 and higher, leaving 0 through 49 available for use. CLVARx 000 would clear indirect registers 1 and higher. To clear indirect register 0, store 0 into it.

The HP 35s indirect registers provides the user with many capabilities beyond what was available with the HP 33s.

Examples using the indirect registers

<u>Example 1:</u> Write a short program to load indirect registers 1 through 10 with random numbers.

<u>Solution:</u> Press GTO · · and then enter the program below. Press ↱ PRGM and then these keystrokes:

↱ LBL A 1 · 0 1 ↱ STO I ↱ RAND ↱ STO (I) ↰ ISG I GTO A 0 0 4 ↰ RTN

The program will look like the one below and will have the length as shown in Figure 2. The program checksum is C289.

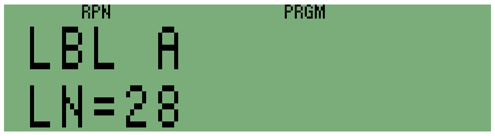

```
RPN              PRGM
LBL A
LN=28
```

Figure 2

```
A001  LBL A
A002  1.01
A003  STO I
A004  RANDOM
A005  STO (I)
A006  ISG I
A007  GTO A004
A008  RTN
```

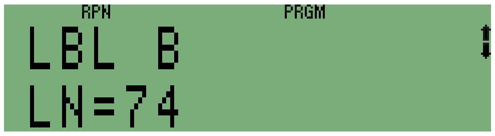
Answer: Run the program by pressing XEQ A ENTER. When the program completes, in RPN mode the last two random numbers generated will be shown while in algebraic mode, only the most recent random number will be shown in the display.

Example 2: Assuming that indirect registers 1 through 10 contain the random numbers generated in example 1, write a short program to sort indirect registers 1 through 10 with the smallest number found in register 1.

Solution: The program presented below will sort indirect registers 1 through 10 using the Bubble Sort. The program will run in either RPN mode only. Press GTO · · and then enter the program below. Press ↱ PRGM and then these keystrokes:

↱ LBL B 1 · 0 1 ↱ STO I 1 + ↱ STO J ↰ FLAGS 2 0 RCL (I) RCL (J) ↰
x?y 3 GTO B 0 1 8 ↰ ISG I ↰ ISG J GTO B 0 0 8 ↰ FLAGS 3 0 GTO
B 0 0 2 ↰ RTN ↰ FLAGS 1 0 ↱ STO (I) x↔y ↱ STO (J) x↔y
GTO B 0 1 2 ↰ RTN

The program will look like the one below and will have the length and checksum as shown in Figure 3. The program checksum is 5C10.



LBL B
LN=74

Figure 3

B001	LBL B	B013	ISG J
B002	1.01	B014	GTO B008
B003	STO I	B015	FS? 0
B004	1	B016	GTO B002
B005	+	B017	RTN
B006	STO J	B018	SF 0
B007	CF 0	B019	STO (I)
B008	RCL (I)	B020	X <> Y
B009	RCL (J)	B021	STO (J)
B010	X < Y?	B022	X <> Y
B011	GTO B018	B023	GTO B012
B012	ISG I		

Answer: Run the program by pressing XEQ B ENTER. As the program executes, the flag 0 annunciator will flash on and off as the program loops through the indirect registers making exchanges of adjacent registers when needed. When the program completes the smallest value will be in indirect register 1 and the largest value will be in indirect register 10.

Example 3: Jake wants a program that will loop through the numbered indirect registers to display the values stored within. Input is to be a number of the form bbb.eee, where bbb is the first indirect register to be viewed and eee is the highest to be viewed. If bbb.eee is positive, the program should PAUSE to display the contents. If bbb.eee is entered as a negative number, the program should STOP to display the value. Jake only wants a program that works in RPN mode. He also wants a program that displays the indirect register being displayed in Y and its contents shown in X.

Solution: Jake wrote the program presented below which will display the indirect registers. The program will run in RPN mode only. Press GTO · · and then enter the program below. Press ↱ PRGM and then these keystrokes:

↱ LBL C ↱ x?0 3 ↰ FLAGS 1 1 ↱ ABS ↱ STO I RCL I ↰ INTG 6 RCL (I) ↰ FLAGS 3 1 R/S ↰ FLAGS 3 1 GTO C 0 1 4 ↱ PSE ↰ ISG I GTO C 0 0 6 ↰ FLAGS 2 1 ↰ RTN

The program will look like the one below and will have the length and checksum as shown in Figures 4 and 5 below.

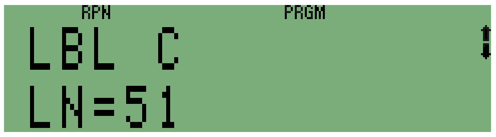

RPN PRGM
LBL C
LN=51

Figure 4

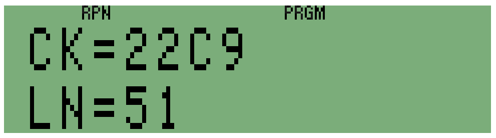

RPN PRGM
CK=22C9
LN=51

Figure 5

```
C001   LBL C
C002   X < 0?
C003   SF 1
C004   ABS
C005   STO I
C006   RCL I
C007   IP
C008   RCL(I)
C009   FS? 1
C010   STOP
C011   FS? 1
C012   GTO C014
C013   PSE
C014   ISG I
C015   GTO C006
C016   CF 1
C017   RTN
```

Answer: Run the program by pressing 1 · 0 1 XEQ C ENTER. Since the input was positive, the HP 35s will pause and display each value stored in indirect registers 1 through 10. To have the HP 35s stop and display the value, key in the input as a negative number. Jake now has the program he wanted. NOTE: If you specify a range of bbb.eee which goes beyond the presently allocated indirect registers, this program will produce an error message.