Chapter 19
Numbers in Different Bases

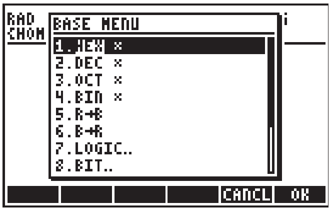
In this Chapter we present examples of calculations of number in bases other than the decimal basis.

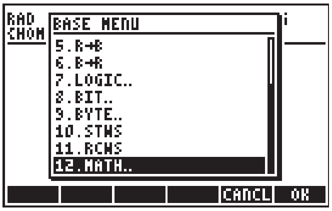
Definitions

The number system used for everyday arithmetic is known as the *decimal* system for it uses 10 (Latin, deca) digits, namely 0-9, to write out any real number. Computers, on the other hand, use a system that is based on two possible states, or *binary* system. These two states are represented by 0 and 1, ON and OFF, or high-voltage and low-voltage. Computers also use number systems based on eight digits (0-7) or *octal* system, and sixteen digits (0-9, A-F) or *hexadecimal*. As in the decimal system, the relative position of digits determines its value. In general, a number n in base b can be written as a series of digits n = $(a_1a_2 \ldots a_n.c_1c_2 \ldots c_m)_b$. The "point" separates *n* "integer" digits from m "decimal" digits. The value of the number, converted to our customary decimal system, is calculated by using $n = a_1 \cdot bn^{-1} + a_2 \cdot b^{n-2} + \ldots + a_nb^0 + c_1 \cdot b^{-1} + c_2 \cdot b^{-2} + \ldots + c_m \cdot b^{-m}$. For example, $(15.234)_{10} = 1 \cdot 10^1 + 5 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2} + 4 \cdot 10^{-3}$, and $(101.111)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$

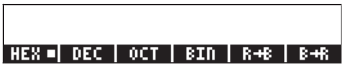
The BASE menu

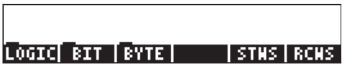
While the calculator would typically be operated using the decimal system, you can produce calculations using the binary, octal, or hexadecimal system. Many of the functions for manipulating number systems other than the decimal system are available in the BASE menu, accessible through ⟨→⟩ BASE (the ⟨3⟩ key). With system flag 117 set to CHOOSE boxes, the BASE menu shows the following entries:





With system flag 117 set to SOFT menus, the BASE menu shows the following:





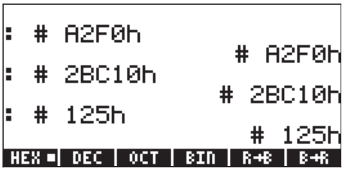
With this format, it is evident that the LOGIC, BIT, and BYTE entries within the BASE menu are themselves sub-menus. These menus are discussed later in this Chapter.

Functions HEX, DEC, OCT, and BIN

Numbers in non-decimal systems are written preceded by the # symbol in the calculator. The symbol # is readily available as ↰ #__ (the 3 key). To select which number system (current base) will be used for numbers preceded by #, select one of the following functions in the first BASE menu, i.e., HEX(adecimal), DEC(imal), OCT(al), or BIN(ary). For example, if HEX■ is selected, any number written in the calculator that starts with # will be a hexadecimal number. Thus, you can write numbers such as #53, #A5B, etc. in this system. As different systems are selected, the numbers will be automatically converted to the new current base.

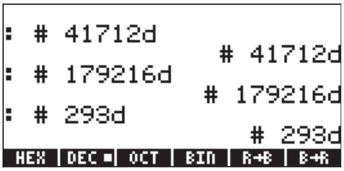
The following examples show the same three numbers written with the # symbol for different current bases:

HEX



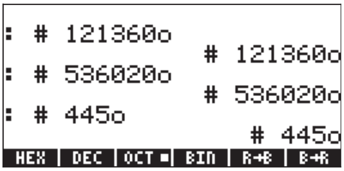
```
: # A2F0h
                # A2F0h
: # 2BC10h
               # 2BC10h
: # 125h
                 # 125h
```

DEC



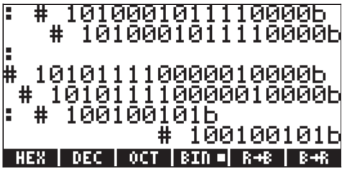
```
: # 41712d
                # 41712d
: # 179216d
               # 179216d
: # 293d
                  # 293d
```

OCT



```
: # 121360o
               # 121360o
: # 536020o
               # 536020o
: # 445o
                  # 445o
```

BIN

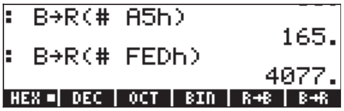


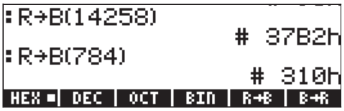
```
: # 1010001011110000b
  # 1010001011110000b
:
# 101011110000010000b
 # 101011110000010000b
: # 100100101b
           # 100100101b
```

As the decimal (DEC) system has 10 digits (0,1,2,3,4,5,6,7,8,9), the hexadecimal (HEX) system has 16 digits (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F), the octal (OCT) system has 8 digits (0,1,2,3,4,5,6,7), and the binary (BIN) system has only 2 digits (0,1).

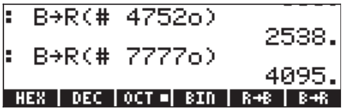
Conversion between number systems

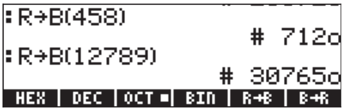
Whatever the number system selected, it is referred to as the binary system for the purpose of using the functions R→B and B→R. For example, if HEX■ is selected, the function B→R will convert any hexadecimal number (preceded by #) into a decimal number, while the function R→B works in the opposite direction. Try the following exercises, HEX is the current base:


```
: B→R(# A5h)
                165.
: B→R(# FEDh)
               4077.
```

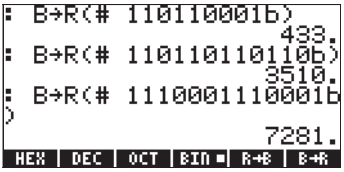

```
:R→B(14258)
            # 37B2h
:R→B(784)
             # 310h
```

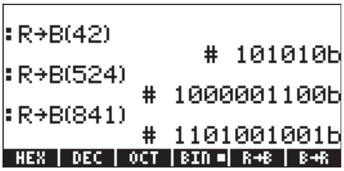
The following examples show conversions when the base is the octal system:


```
: B→R(# 4752o)
               2538.
: B→R(# 7777o)
               4095.
```


```
:R→B(458)
             # 712o
:R→B(12789)
           # 30765o
```

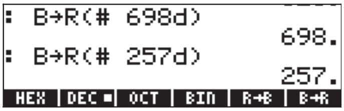
We also present transformations using the binary system as the current base:

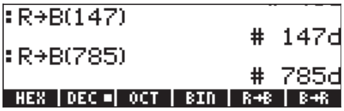

```
: B→R(# 110110001b)
                433.
: B→R(# 110110110110b)
               3510.
: B→R(# 1110001110001b
)
               7281.
```


```
:R→B(42)
          # 101010b
:R→B(524)
      # 1000001100b
:R→B(841)
      # 1101001001b
```

Notice that every time you enter a number starting with #, you get as the entry the number you entered preceded by # and followed by the letter h, o, or b (hexadecimal, octal, or binary). The type of letter used as suffix depends on which non-decimal number system has been selected, i.e., HEX, OCT, or BIN.

To see what happens if you select the DEC setting, try the following conversions:


```
: B→R(# 698d)
                  698.
: B→R(# 257d)
                  257.
```


```
:R→B(147)
              # 147d
:R→B(785)
              # 785d
```

The only effect of selecting the DECimal system is that decimal numbers, when started with the symbol #, are written with the suffix d.

Wordsize

The wordsize is the number of bits in a binary object. By default, the wordsize is 64 bites. Function RCWS (ReCall WordSize) shows the current wordsize. Function STWS (SeT the WordSize) allows the user to reset the wordsize to any number between 0 and 64.

Changing the wordsize will affect the way that binary integer operations are performed. For example, if a binary integer exceeds the current wordsize, the leading bits will be dropped before any operation can be performed on such number.

Operations with binary integers

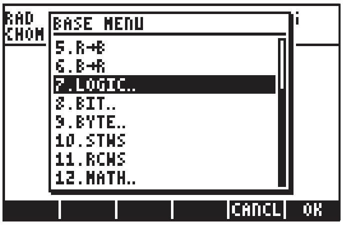
The operations of addition, subtraction, change of sign, multiplication, and division are defined for binary integers. Some examples, of addition and subtraction, are shown below, for different current bases:

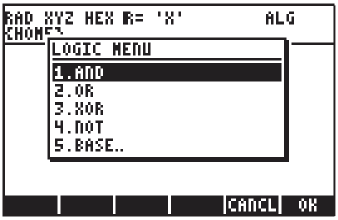
```
#A02h + #12Ah = #B2Ch
#2562d + #298d = #2860d
#5002o + #452o = #5454o
#101000000010b + #100101010b = #101100101100b
```

```
#A02h - #12Ah = #8D8h
#2562d - #298d = #2264d
#5002o - #452o = #4330o
#101000000010b - #100101010b = #100011011000b
```

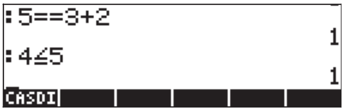
The LOGIC menu

The LOGIC menu, available through the BASE (BASE) provides the following functions:

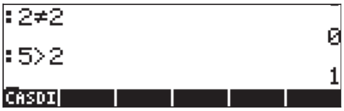




The functions AND, OR, XOR (exclusive OR), and NOT are logical functions. The input to these functions are two values or expressions (one in the case of NOT) that can be expressed as binary logical results, i.e., 0 or 1. Comparisons of numbers through the comparison operators =, ≠, >, <, ≤, and ≥, are logical statements that can be either true (1) or false (0). Some examples of logical statements are shown below:



```
:5==3+2
                    1
:4≤5
                    1
```



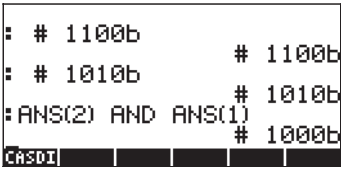
```
:2≠2
                    0
:5>2
                    1
```

Functions AND, OR, XOR, and NOT can be applied to comparison statements under the following rules:

1 AND 1 = 1	1 AND 0 = 0	0 AND 1 = 0	0 AND 0 = 0
1 OR 1 = 1	1 OR 0 = 1	0 OR 1 = 1	0 OR 0 = 0
1 XOR 1 = 0	1 XOR 0 = 1	0 XOR 1 = 1	0 XOR 0 = 0
NOT(1) = 0	NOT(0) = 1		

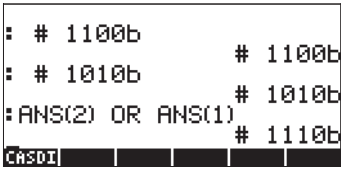
These functions can be used to build logical statements for programming purposes. In the context of this Chapter, they will by used to provide the result of bit-by-bit operations along the lines of the rules provided above. In the following examples, the base number system is indicated in parentheses:

AND (BIN)



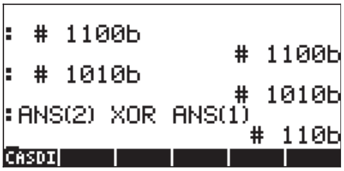
```
: # 1100b
                # 1100b
: # 1010b
                # 1010b
:ANS(2) AND ANS(1)
                # 1000b
CASDI
```

OR (BIN)



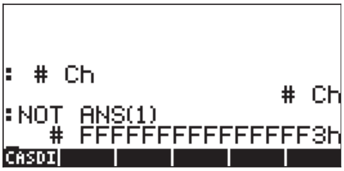
```
: # 1100b
                # 1100b
: # 1010b
                # 1010b
:ANS(2) OR ANS(1)
                # 1110b
CASDI
```

XOR (BIN)



```
: # 1100b
                # 1100b
: # 1010b
                # 1010b
:ANS(2) XOR ANS(1)
                 # 110b
CASDI
```

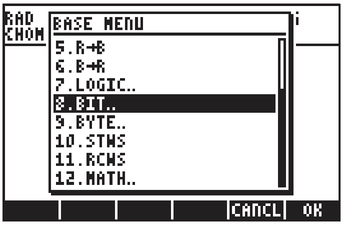
NOT (HEX)



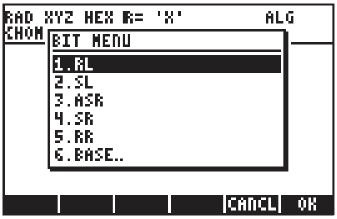
```
: # Ch
                   # Ch
:NOT ANS(1)
  # FFFFFFFFFFFFFFF3h
CASDI
```

The BIT menu

The BIT menu, available through the BASE (BASE) provides the following functions:



```
RAD  BASE MENU
XHOM 5.R→B
     6.B→R
     7.LOGIC..
     8.BIT..
     9.BYTE..
     10.STWS
     11.RCWS
     12.MATH..
                    CANCL  OK
```



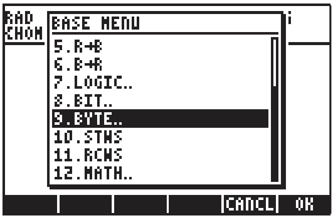
```
RAD XYZ HEX R= 'X'        ALG
XHOM  BIT MENU
      1.RL
      2.SL
      3.ASR
      4.SR
      5.RR
      6.BASE..
                    CANCL  OK
```

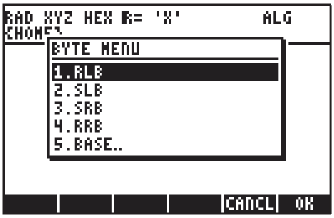
Functions RL, SL, ASR, SR, RR, contained in the BIT menu, are used to manipulate bits in a binary integer. The definition of these functions are shown below:

RL: Rotate Left one bit, e.g., #1100b → #11000b
SL: Shift Left one bit, e.g., #1101b → #11010b
ASR: Arithmetic Shift Right one bit, e.g., #1100010b → #110001b
SR: Shift Right one bit, e.g., #11011b →#1101b
RR: Rotate Right one bit, e.g., #1101b → #1000000000000000000000000000000000000000000000000000000000001b

The BYTE menu

The BYTE menu, available through the BASE (BASE) provides the following functions:





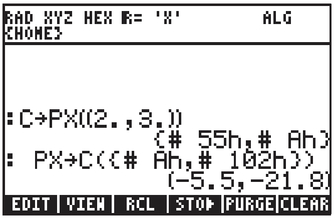
Functions RLB, SLB, SRB, RRB, contained in the BIT menu, are used to manipulate bits in a binary integer. The definition of these functions are shown below:

RLB: Rotate Left one byte, e.g., #1100b → #110000000000b
SLB: Shift Left one byte, e.g., #1101b → #110100000000b
SRB: Shift Right one byte, e.g., #11011b →#0b
RRB: Rotate Right one byte, e.g., #1101b →
#11010000000000000000000000000000000000000000000000000
00000000000b

Hexadecimal numbers for pixel references

Many plot option specifications use pixel references as input, e.g., { #332h #A23h } #Ah 0. 360. ARC, to draw an arc of a circle. We use functions C→PX and PX→C to convert quickly between user-unit coordinates and pixel references. These functions can be found through the command catalog (CAT).

Some examples are shown below:



```
RAD XYZ HEX R= 'X'          ALG
{HOME}

:C→PX((2.,3.))
              {# 55h,# Ah}
: PX→C({# Ah,# 102h})
               (-5.5,-21.8)
```