

# Concepts of Conversion of Base 10 Number to Base 2 Fixed Point Register Binary Number Representation

Sean Rodby, Luke Snyder, Autar Kaw  
University of South Florida  
United States of America  
kaw@eng.usf.edu

Website: <http://numericalmethods.eng.usf.edu>  
Version: *Mathematica 6*; Last Revised: August 28, 2008

## Initialization

---

Clearing the definitions of all symbols in the current context:

```
ClearAll [Evaluate [Context [] <> "*" ]]
```

## Introduction

---

The following worksheet illustrates how to convert a positive decimal number to a fixed point register binary number using loops and various conditional statements. The user inputs a decimal number in the *Input* section of the program, and then an equivalent binary number is given as an output.

## Section 1: Input Data

---

This is the only section where the user interacts with the program.

- Enter positive real number to be converted to binary number

```
decnum = 54.3;
```

- Enter number of bits to be used in *integer part*

```
m = 8;
```

- Enter number of bits to be used in *fractional part*

```
n = 6;
```

This is the end of the user section. All information must be entered before proceeding to the next section. **RE-EVALUATE THE NOTEBOOK.**

## Section 2: Procedure

---

Calculating the minimum and maximum possible base-10 values based on the number of bits specified by the user.

```

sum1 = 0;
Do[sum1 = sum1 + 2i, {i, 0, m - 1}]
sum2 = 0;
Do[sum2 = sum2 + 2-i, {i, n}]

maxval = N[sum1 + sum2, 10]

255.9843750

minval = N[2-n, 10]

0.01562500000

```

If the maximum value that can be produced given the user specified number of bits is smaller than the number to be represented, then more bits are needed. Similarly, if the minimum value that can be produced given the user specified number of bits is larger than the number to be represented, a change in bits is needed.

```

If[maxval < decnum,
  Print["Since |", decnum, "| > ", maxval, " the number of bits specified is not
    sufficient to represent this number in floating point representation.
    Either specify fewer mantissa bits, or more total bits for the worksheet."]]
If[minval > decnum, Print["Since ", minval, " > |", decnum,
  "|, the number of bits specified is not sufficient to represent
    this number in floating point representation. Either specify
    fewer mantissa bits, or more total bits for the worksheet."]]
If[minval ≤ decnum ∧ maxval ≥ decnum, Print["Since ", minval, " < |", decnum, "| < ",
  maxval, " the base-10 number can be represented in floating point binary format."]]

```

Since  $0.01562500000 < |54.3| < 255.9843750$   
the base-10 number can be represented in floating point binary format.

Using the *Floor* command to isolate each section of the decimal number to begin binary number conversion

```

beforedecimal = Floor[decnum]

54

afterdecimal = decnum - Floor[decnum]

0.3

```

Using a loop to perform the binary conversion repetitively until the necessary number of bits are used. Using the floor command to calculate the quotient of the decimal number. Using a combination of the *Ceiling* and *Floor* commands to determine the remainder, and inputting this value into an array. Then calculating the binary conversion of the value before the decimal. Here we use the variable *bold* to denote the value of the integer part of the decimal number.

```

bold = beforedecimal ;
j = 1 ;
Do [
  quotient = Floor [  $\frac{\mathbf{bold}}{2}$  ] ;
  Binaryi = Ceiling [  $\frac{\mathbf{bold}}{2}$  ] - Floor [  $\frac{\mathbf{bold}}{2}$  ] ;
  If [ quotient == 0  $\wedge$  beforedecimal  $\neq$  0 ,
    Binaryi = 1 ;
    Break [ ] ;
  If [ beforedecimal == 0 ,
    Binaryi = 0 ;
    Break [ ] ;
  bold = quotient ;
  j = j + 1 ,
  { i , m } ]

```

Using this method, the placement of the binary numbers in the array *Binary* are backwards and so must be flipped around to form the actual binary number.

```

Do [ BinaryBeforeDeci = Binaryj+1-i , { i , j } ]

```

Converting the *fractional part* of the decimal into a binary number. Here we initialize the sum using a new value, *boldA*, which represents the value after the decimal point.

```

boldA = afterdecimal ;
Do [
  Num = boldA * 2 ;
  If [ Num < 1 ,
    BinaryAfterDeci = 0 ] ;
  If [ Num  $\geq$  1 ,
    Num = Num - Floor [ Num ] ;
    BinaryAfterDeci = 1 ] ;
  boldA = Num ,
  { i , n } ]

```

Combining the binary numbers and determining the number of terms in the resulting base 2 number. Then displaying the results.

```

str1 = "" ;
Do [ str1 = StringJoin [ str1 , Tostring [ BinaryBeforeDeci ] ] , { i , j } ] ;
str2 = "" ;
Do [ str2 = StringJoin [ str2 , Tostring [ BinaryAfterDeci ] ] , { i , n } ] ;
FinalBinNum = StringJoin [ str1 , ".", str2 ]

110110.010011

```

Using a loop to sum values of the *integer* portion of the base-2 number. The loop variable *sumint* is used for summation and is initialized at 0.

```

intbin = Floor [ToExpression [FinalBinNum]] ;
strint = ToString [intbin] ;
m = StringLength [strint] ;
n = StringLength [FinalBinNum] ;

sumint = 0 ;
Do [
  bininti = ToExpression [StringTake [strint, {i}]] ;
  sumint = sumint + bininti * 2m-i,
  {i, m}]

```

Using a loop to sum values of the *fractional* portion of the base-2 number. The loop variable *sumfrac* is used for summation and is initialized at 0.

```

sumfrac = 0 ;
j = 1 ;
Do [
  binfracj = ToExpression [StringTake [FinalBinNum, {i}]] ;
  sumfrac = sumfrac + binfracj * 2-j ;
  j = j + 1,
  {i, m + 2, n}]

```

Adding the *fractional* portion of the base-2 number with the *integer* portion which yields the base-10 number.

```

totaldec = N [sumint + sumfrac, 10]

54.29687500

```

Finding the total relative percentage difference between the representation and the number to be represented.

```

totpercdiff =  $\frac{\text{decnum} - \text{totaldec}}{\text{decnum}} * 100$ 

0.00575506

```

## Conclusion

---

This worksheet illustrates the use of *Mathematica* to convert a base-10 number to a base-2 binary number. It is important to understand the binary system as it has numerous applications. Critical to this understanding is being able to convert binary numbers to base-10 numbers, and vice-versa.

## References

---

Binary Representation of Numbers

See: [http://numericalmethods.eng.usf.edu/nbm/gen/01aae/nbm\\_gen\\_aae\\_txt\\_binaryrepresentation.pdf](http://numericalmethods.eng.usf.edu/nbm/gen/01aae/nbm_gen_aae_txt_binaryrepresentation.pdf)

*Legal Notice: The copyright for this application is owned by the author(s). Neither Wolfram Research nor the author(s) are responsible for any errors contained within and are not liable for any damages resulting from the use of this material. This application is intended for non-commercial, non-profit use only. Contact the author for permission if you wish to use this application in for-profit activities.*