
UNIT 18 – COMPUTATIONAL THINKING

Understanding and Applying Computational Calculations

⚙ 010101 110010

INDUCTION BOOKLET

Technology Company Work Experience Programmed

IT Department – Educational Materials Development

Student Name: _____
Course: BTEC IT
Unit: Unit 18 – Computational Thinking
Learning Aim: A
Teacher: _____
Submission Date: _____

TABLE OF CONTENTS	
Topic	Page no
1. Introduction • Overview of the Work Experience Program • Purpose of the Induction Booklet • Importance of Computational Thinking in IT	3
2. The Role of Computational Calculations in Problem-Solving • Why Calculations Matter in Computing • Examples of Computational Calculations in Real-World Scenarios	4-5
3. Understanding Number Systems in Computing • Natural Numbers, Integers, and Rational Numbers • Why Number Representation is Crucial in Computing	6-9
4. Binary and Hexadecimal Number Systems • Introduction to Binary (Base-2) and Its Role in Computing • Understanding Hexadecimal (Base-16) and Its Applications • Converting Between Decimal, Binary, and Hexadecimal	10-14
5. Power Notation and Its Computational Use • Definition and Purpose of Power Notation • Writing Numbers in Power Notation Including Powers of Two • Applications in Memory Storage, Processing, and Data Representation	15-16
6. Rounding Errors in Computational Calculations • What Are Rounding Errors and Why Do They Occur? • Impact of Rounding Errors on Data Accuracy • Methods for Identifying and Reducing Rounding Errors	17-18
7. 6-Bit Binary Representation • Understanding Unsigned and Signed Binary Representations • Two's Complement for Negative Numbers • Examples and Practical Applications	19
8. Converting Between Number Systems • Decimal to Binary Conversion (Step-by-Step) • Binary to Decimal Conversion • Decimal to Hexadecimal Conversion • Binary to Hexadecimal Conversion	20-22
9. Practice Questions & Solutions • Writing Numbers in Power Notation	23-28

• Binary, Decimal, and Hexadecimal Conversions • Identifying and Fixing Rounding Errors • Applying Two's Complement for Negative Numbers	
10. Glossary • Definitions of Important Computational Concepts	29-30
11. Conclusion • Key Takeaways from the Booklet • The Importance of Computational Thinking in IT Careers	31-32
12. Reference	33

1. Introduction

-

This induction booklet has been created as part of a work experience programme in

a technology company. The booklet is designed for students aged 14 to 15 who are new to computing. Its purpose is to introduce learners to computational thinking and explain how calculations are used in computing to solve problems.

- The booklet explains key ideas in a simple way so that new learners can understand how computers use numbers, logic, and step-by-step processes to complete tasks in real life.
- **Overview of the Work Experience Program**
- The work experience programme gives students the opportunity to understand how an IT department operates in a professional setting. During the programme, learners are introduced to key computing concepts, problem-solving techniques, and how real-world systems are designed and maintained. It helps students develop basic technical knowledge as well as logical thinking skills that are important in the technology industry.
- **Purpose of the Induction Booklet**
- The purpose of this induction booklet is to introduce junior learners to computational thinking and explain how calculations are used in computing. It provides clear explanations of number systems, binary representation, hexadecimal numbers, power notation, and rounding errors. The booklet aims to build a strong foundation so learners can understand how computers process information and solve problems.
- **Importance of Computational Thinking in IT**
- Computational thinking is important in IT because it helps people solve problems in a structured and logical way. In computing, problems are broken down into smaller steps, and calculations are used to find accurate solutions. Every digital system, from online shopping websites to mobile apps and banking software, depends on computational thinking to function correctly. Without it, computer systems would not be able to process data or make decisions efficiently.

2. The Role of Computational Calculations in Problem-Solving

- **Why Calculations Matter in Computing**

Computers do not “think” like humans. They follow instructions and perform calculations very fast.

Calculations help computers to:

- Add, subtract, multiply and divide numbers
- Compare values (greater than, less than)
- Store and process data
- Make logical decisions

For example:

If a student marks is 85, the system calculates:

If marks $\geq 80 \rightarrow$ Grade A

That is a comparison calculation.

Another example:

An online shop calculates:

Total price = item price $\times$ quantity

If price = £12

Quantity = 4

Total = $12 \times 4 = £48$

Without calculations, the system cannot give correct output.

- **Real-Life Examples of Computational Calculations**

Example 1: Banking System

Banks use calculations to:

- Add interest
- Deduct payments
- Calculate loan amounts

If a bank gives 5% interest on £1000:

Interest = $1000 \times 0.05 = £50$

Final amount = £1050

Example 2: social media

When you upload a picture:

- The system calculates file size
- It compresses data
- It calculates storage space

Example 3: GPS Navigation

Google Maps calculates:

- Distance
- Speed
- Estimated arrival time

If distance = 120 km

Speed = 60 km/h

Time = Distance ÷ Speed

Time = $120 \div 60 = 2$ hours

Example 4: Gaming

In games:

Score = previous score + new points

Health = health – damage

If health = 100

Damage = 25

New health = 75

This is all computational calculation.

3. Understanding Number Systems in Computing

Computers use different types of numbers. Each type has a purpose.

Natural Numbers

Natural numbers are counting numbers:

1, 2, 3, 4, 5, 6, ...

They are used in computing for:

- Counting objects
- Loop counters
- Index numbers

Example:

If a program prints numbers from 1 to 10:

for i = 1 to 10

Here, natural numbers control repetition.

Integers

Integers include positive and negative numbers:

..., -3, -2, -1, 0, 1, 2, 3, ...

They are used when values can go below zero.

Example:

Temperature system:

If temperature drops 5 degrees:

$10^{\circ}\text{C} \rightarrow 5^{\circ}\text{C}$

Or below zero:

$2^{\circ}\text{C} \rightarrow -3^{\circ}\text{C}$

In games:

Player score can become negative.

Bank balance:

If you spend more money than you have:

$\pounds 20 - \pounds 50 = -\pounds 30$

So integers are very important.

Rational Numbers

Rational numbers are fractions like:

$1/2$, $3/4$, $5/8$

They can also be decimals:

0.5, 0.25, 0.75

They are used in:

- Measurements
- Graphics scaling
- Screen resolution

Example:

Aspect ratio of screen:

16:9

This means $\text{width} \div \text{height} = 16/9$

If screen width = 1920

Height = 1080

$1920 \div 1080 = 16/9$

Binary Numbers

Binary only uses:

0 and 1

Computers use binary because electronic circuits have two states:

ON (1)

OFF (0)

Example:

Decimal 5 in binary:

$5 = 101$

Because:

$$\begin{aligned} &1 \times 4 + 0 \times 2 + 1 \times 1 \\ &= 4 + 0 + 1 \\ &= 5 \end{aligned}$$

Binary is used for:

- Data storage
- Memory
- Processing

Every letter, image, video becomes binary.

Hexadecimal Numbers

Hexadecimal uses:

0-9 and A-F

A = 10

B = 11

C = 12

D = 13

E = 14

F = 15

Hexadecimal is shorter than binary.

Example:

Binary: 11111111

Hexadecimal: FF

Much easier to read.

Used in:

- Memory addresses
- Web colour codes (#FF0000 = Red)
- Programming

Why Number Representation is Important

If numbers are stored wrongly:

- Calculations become incorrect

- Systems crash
- Data becomes corrupted

Correct number representation ensures:

- Accuracy
- Efficiency
- Speed

For example:

1 KB = 1024 bytes
Because $1024 = 2^{10}$

This is power of two.

4. Binary and Hexadecimal Number Systems

Introduction to Binary and Its Role in Computing

Binary is the main number system used in computers. It is called base-2 because it only uses two numbers:

0 and 1

Computers use binary because inside a computer there are electronic circuits. These circuits only understand two states:

- ON
- OFF

ON is written as 1

OFF is written as 0

Because of this, everything in a computer becomes 0s and 1s.

For example:

When you type your name on a keyboard, the computer does not store the letters directly. It changes them into binary code.

Even pictures, music, and videos are stored in binary form.

How Binary Works

Binary uses powers of 2.

Each position in a binary number has a value.

Example:

Binary number: 10110

From right to left:

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

Now calculate:

$$\begin{aligned}
 &1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 \\
 &= 16 + 4 + 2 \\
 &= 22
 \end{aligned}$$

So:

$$10110_2 = 22_{10}$$

This shows how binary numbers change into decimal numbers.

Why Binary Is Important

Binary is important because:

- It stores data
- It allows calculations
- It runs programs
- It controls hardware

For example, in an 8-bit system:

$$\begin{aligned}
 \text{Maximum number} &= 2^8 - 1 \\
 &= 255
 \end{aligned}$$

That is why colours in RGB go from 0 to 255.

Understanding Hexadecimal (Base-16) and Its Applications

Hexadecimal is base-16 number system.

It uses numbers:

0 1 2 3 4 5 6 7 8 9

And letters:

A B C D E F

Where:

A = 10

B = 11

C = 12

D = 13

E = 14

F = 15

Hexadecimal is easier to read than long binary numbers.

For example:

Binary: 11111111

Hexadecimal: FF

Hexadecimal is shorter and clearer.

Where Hexadecimal Is Used

Hexadecimal is used in:

- Memory addresses
- Programming
- Web colour codes
- Networking

Example in web design:

#FF0000 = Red

#00FF00 = Green

#0000FF = Blue

Each pair shows colour value in hexadecimal.

Binary and Hexadecimal Connection

Every 4 binary digits = 1 hexadecimal digit.

Example:

Binary: 1100

1100 = 12

12 in hex = C

So:

$1100_2 = C_{16}$

Another example:

Binary: 1011

1011 = 11

11 = B

So:

$1011_2 = B_{16}$

Converting Between Decimal, Binary, and Hexadecimal

Decimal to Binary (Step-by-Step)

Convert 74 to binary.

$$74 \div 2 = 37 \text{ remainder } 0$$

$$37 \div 2 = 18 \text{ remainder } 1$$

$$18 \div 2 = 9 \text{ remainder } 0$$

$$9 \div 2 = 4 \text{ remainder } 1$$

$$4 \div 2 = 2 \text{ remainder } 0$$

$$2 \div 2 = 1 \text{ remainder } 0$$

$$1 \div 2 = 0 \text{ remainder } 1$$

Write remainders from bottom to top:

1001010

So:

$$74_{10} = 1001010_2$$

Binary to Decimal

Convert 101101_2 to decimal.

$$1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$

$$= 32 + 8 + 4 + 1$$

$$= 45$$

So:

$$101101_2 = 45_{10}$$

Decimal to Hexadecimal

Convert 190 to hexadecimal.

$$190 \div 16 = 11 \text{ remainder } 14$$

$$14 = E$$

$$11 \div 16 = 0 \text{ remainder } 11$$

$$11 = B$$

So:

$$190_{10} = BE_{16}$$

Binary to Hexadecimal

Convert:

11101001

Group into 4 bits:

1110 1001

1110 = E

1001 = 9

So:

$11101001_2 = E9_{16}$

5. Power Notation and Its Computational Use

Definition and Purpose of Power Notation

Power notation is a way to write numbers using exponents.

Example:

2^3 means:

$$2 \times 2 \times 2 = 8$$

It helps write big numbers in shorter form.

Writing Numbers Using Powers of Two

Computers use powers of 2 because binary is base-2.

Important powers:

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

$$2^{10} = 1024$$

Example:

$$1024 = 2^{10}$$

$$2048 = 2^{11}$$

Scientific Notation Example

$$60000$$

$$= 6 \times 10^4$$

$$0.004$$

$$= 4 \times 10^{-3}$$

Applications in Memory and Data

Memory example:

$$8 \text{ bits} = 1 \text{ byte}$$

Maximum value:

$$2^8 - 1 = 255$$

16-bit system:

$$2^{16} - 1 = 65535$$

This explains data limits in systems.

6. Rounding Errors in Computational Calculations

What Are Rounding Errors?

Rounding error happens when computer cannot store a decimal number exactly.

Some numbers repeat forever in binary.

For example:

0.1 cannot be stored exactly in binary.

So computer stores a number very close to 0.1, but not exactly 0.1.

Why Do They Occur?

They happen because:

- Limited memory
- Floating-point representation
- Infinite binary fractions

Example of Rounding Error

In programming:

$0.1 + 0.2$

Sometimes result is:

0.30000000004

Not exactly 0.3

This is because of rounding error.

How to Reduce Rounding Errors

- Use higher precision data types
- Avoid exact equality comparison
- Use small tolerance value

Example:

Instead of:

if $a == b$

Use:

if $|a - b| < 0.0001$



Rounding Error

['raʊnd-ɪŋ 'er-ər]

A mathematical miscalculation or quantization error caused by altering a number to an integer or one with fewer decimals.

7. 6-Bit Binary Representation

Unsigned 6-bit Numbers

6 bits can represent:

$$2^6 = 64 \text{ numbers}$$

Range:

0 to 63

Example:

25 in 6-bit:

011001

Signed 6-bit Numbers

Signed 6-bit range:

-32 to +31

Two's Complement Example

Represent -13 in 6-bit.

Step 1: Convert 13 to binary:

001101

Step 2: Invert bits:

110010

Step 3: Add 1:

110011

So:

-13 = 110011

8. converting between number systems

Computers use different number systems to store and process data. Humans normally use decimal numbers, but computers use binary numbers. Because of this, sometimes we must convert numbers from one system to another.

This process is called converting between number systems.

Even though the number system changes, the value of the number stays the same.

Example 1: Decimal to Binary

Decimal number: 10

Convert to binary:

$$10 = 1010$$

So:

$$10_{10} = 1010_2$$

Both numbers mean the same value, but they are written in different number systems.

Example 2: Decimal to Hexadecimal

Decimal number: 26

Convert to hexadecimal:

$$26 = 1A$$

So:

$$26_{10} = 1A_{16}$$

Example 3: Binary to Decimal

Binary number: 1011

Convert to decimal:

$$\begin{aligned} &1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 \\ &= 8 + 2 + 1 \\ &= 11 \end{aligned}$$

So:

$$1011_2 = 11_{10}$$

Why Converting Number Systems is Important

Converting number systems is important in computing because:

- Computers store data in binary
- Programmers often read data in hexadecimal
- Humans understand numbers better in decimal

So conversions help us understand and work with computer data.

Real Life Example

For example, in web design a colour might look like this:

#FF5733

This is a hexadecimal colour code.

The computer converts this hexadecimal value into binary numbers so it can display the colour on the screen.

9. Practice Questions and Solutions

This section gives many practice questions. These questions help students understand number systems, power notation, rounding errors, and two's complement. Each question includes step-by-step solutions.

Part A – Writing Numbers in Power Notation

Question 1

Write 4096 as a power of 2.

Solution:

$$4096 = 2^{12}$$

Because:

$$2^{12} = 4096$$

Question 2

Write 32768 as a power of 2.

$$32768 = 2^{15}$$

Question 3

Write 0.0008 in scientific notation.

Move decimal 4 places:

$$0.0008 = 8 \times 10^{-4}$$

Question 4

Write 95000 in scientific notation.

Move decimal 4 places:

$$95000 = 9.5 \times 10^4$$

Part B – Decimal to Binary Conversion

Question 5

Convert 55 to binary.

$$55 \div 2 = 27 \text{ r}1$$

$$27 \div 2 = 13 \text{ r}1$$

$$13 \div 2 = 6 \text{ r}1$$

$$6 \div 2 = 3 \text{ r}0$$

$$3 \div 2 = 1 \text{ r}1$$

$$1 \div 2 = 0 \text{ r}1$$

Write from bottom:

110111

So:

$$55_{10} = 110111_2$$

Question 6

Convert 100 to binary.

$$100 \div 2 = 50 \text{ r}0$$

$$50 \div 2 = 25 \text{ r}0$$

$$25 \div 2 = 12 \text{ r}1$$

$$12 \div 2 = 6 \text{ r}0$$

$$6 \div 2 = 3 \text{ r}0$$

$$3 \div 2 = 1 \text{ r}1$$

$$1 \div 2 = 0 \text{ r}1$$

Answer:

1100100

So:

$$100_{10} = 1100100_2$$

Part C – Binary to Decimal

Question 7

Convert 101010_2 to decimal.

$$\begin{aligned} &1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 \\ &= 32 + 8 + 2 \\ &= 42 \end{aligned}$$

So:

$$101010_2 = 42_{10}$$

Question 8

Convert 111100_2 to decimal.

$$\begin{aligned} &1 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 \\ &= 32 + 16 + 8 + 4 \\ &= 60 \end{aligned}$$

So:

$$111100_2 = 60_{10}$$

Part D – Decimal to Hexadecimal

Question 9

Convert 220 to hexadecimal.

$$220 \div 16 = 13 \text{ r}12$$

$$12 = \text{C}$$

$$13 \div 16 = 0 \text{ r}13$$

$$13 = \text{D}$$

Answer:

DC

So:

$$220_{10} = DC_{16}$$

Question 10

Convert 300 to hexadecimal.

$$300 \div 16 = 18 \text{ r}12$$

$$12 = C$$

$$18 \div 16 = 1 \text{ r}2$$

$$1 \div 16 = 0 \text{ r}1$$

Answer:

12C

So:

$$300_{10} = 12C_{16}$$

Part E – Binary to Hexadecimal

Question 11

Convert 11001111_2 to hexadecimal.

Group into 4 bits:

1100 1111

$$1100 = C$$

$$1111 = F$$

Answer:

CF

Question 12

Convert 10110110_2 to hexadecimal.

1011 0110

$$1011 = B$$

$$0110 = 6$$

Answer:

B6

Part F – Two's Complement

Question 13

Represent -9 in 6-bit binary.

Step 1: Convert 9:

001001

Step 2: Invert bits:

110110

Step 3: Add 1:

110111

So:

-9 = 110111

Question 14

Represent -15 in 6-bit binary.

15 = 001111

Invert:

110000

Add 1:

110001

So:

-15 = 110001

Part G – Rounding Errors

Question 15

Why does $0.1 + 0.2$ not always equal 0.3?

Answer:

Because 0.1 cannot be stored exactly in binary. The computer stores an approximate value. When adding, small errors appear.

Question 16

If 5000 transactions lose £0.005 each because of rounding, how much total loss?

$$0.005 \times 5000 = 25$$

$$\text{Total loss} = \text{£}25$$

This shows how small rounding errors can become large problem.

Extra Practice (Advanced Level)**Question 17**

What is maximum value in 8-bit unsigned system?

$$2^8 - 1 = 255$$

Question 18

What is range of signed 8-bit system?

$$-128 \text{ to } +127$$

Because:

$$2^7 = 128$$

Question 19

Convert 255 to binary.

Answer:

$$11111111$$

Question 20

Convert FF_{16} to decimal.

$$\text{F} = 15$$

$$15 \times 16 + 15 = 255$$

10. Glossary

Term	Definition
Binary	Binary is a number system that uses only two digits, 0 and 1. It is the basic language used by computers. All types of data inside a computer, such as numbers, text, images, and programs, are finally stored and processed in binary form.
Hexadecimal	Hexadecimal is a base-16 number system. It uses the numbers 0–9 and the letters A–F. This system is often used in computing because it is shorter and easier to read than long binary numbers. It is commonly used in memory addresses, colour codes in web design, and programming.
Power Notation	Power notation is a way of writing numbers using exponents. In computing it is used to describe values based on powers of two. For example, $2^{10} = 1024$, which is about 1 kilobyte (KB) of data. This method helps represent large numbers more clearly.
Rounding Error	A rounding error is a small difference that happens when a number cannot be stored exactly in a computer system. Because computers have limited precision, some numbers are rounded to the closest possible value. This can sometimes cause small calculation mistakes.
Floating-Point	Floating-point is a method used by computers to store decimal numbers. It stores the number using a sign, exponent, and fraction (mantissa). Because of binary limitations, some decimal numbers cannot be stored exactly, which can lead to rounding errors.
Two's Complement	Two's complement is a method used in binary systems to represent negative numbers. In this system, the most significant bit (MSB) is used to show the sign of the number. This method allows computers to perform addition and subtraction easily with both positive and negative values.
Natural Numbers	Natural numbers are counting numbers such as 1, 2, 3, 4, and so on. In computing they are often used for counting operations, loops, and indexing positions in arrays or lists.
Integers	Integers are whole numbers that include positive numbers, negative numbers, and zero. In computing they are used in many situations, such as tracking positions, calculating balances, or representing temperature changes.
Rational Numbers	Rational numbers are numbers that can be written as a fraction of two integers, such as $1/2$ or $-3/4$. In computing they are useful when working with precise measurements, ratios, and graphical calculations like screen resolutions or aspect ratios.

Summary of Key Points

Power notation makes it easier to work with very large numbers in computing systems. Instead of writing long numbers, we can use exponents to show them in a simpler way.

Different number systems such as binary, decimal, and hexadecimal are very important in computing. They help computers store and process data correctly.

Rounding errors can happen when computers cannot represent numbers exactly because of limited memory precision. These small errors may affect results in fields like finance or scientific calculations.

Binary representation is the most important system in computing because computers store and process all information using 0s and 1s.

Two's complement is used in computers to represent negative numbers in binary form. This method makes it easier for computers to perform arithmetic operations.

Natural numbers are often used in computing when counting items, controlling loops in programs, or organizing data in lists and arrays.

Integers allow computers to represent both positive and negative values. This is useful in many situations such as financial systems, tracking movement, or measuring temperature changes.

Rational numbers are useful in computing when accurate fractions are needed. They are used in calculations related to measurements, graphics, and screen resolutions.

Hexadecimal numbers are commonly used in programming and digital systems. They provide a shorter way to represent binary numbers and are used in areas such as memory addressing, web colour codes, and digital forensics.

11. Conclusion

In this booklet, I explained how computational calculations are used in computing systems. I also explained binary and hexadecimal number systems, power notation, rounding errors, and two's complement representation.

Binary is the foundation of computing. All computers use binary to store and process data. Every number, letter, and image is converted into 0s and 1s.

Hexadecimal makes binary easier to read and understand. It is used in programming, memory addresses, and colour codes.

Power notation helps us understand memory sizes and system limits. For example, 2^8 explains why 8-bit systems go from 0 to 255.

Rounding errors show that computers are not perfect. Because computers use limited memory, some numbers cannot be stored exactly. This can cause small errors, especially in financial and scientific systems.

Two's complement allows computers to represent negative numbers correctly. Without two's complement, computers cannot handle signed values.

Key Takeaways from the Booklet

- Computers use binary as their main language.
- Hexadecimal is used to simplify binary.
- Power notation explains memory and data limits.
- Rounding errors can affect accuracy.
- Two's complement is used for negative numbers.
- Understanding number systems improves computational thinking.

Importance of Computational Thinking in IT Careers

Computational thinking is very important in IT jobs.

It helps in:

- Programming
- Cybersecurity
- Data science
- Artificial intelligence
- Web development
- Game design

IT professionals must understand how numbers are stored and calculated inside a system.

If a programmer does not understand binary, they may not understand memory problems.

If a developer does not understand rounding errors, financial systems may have mistakes.

If an engineer does not understand number limits, system may crash due to overflow.

Computational thinking also improves problem-solving skills. It teaches us to break big problems into small parts and solve them step by step.

This skill is useful not only in IT, but also in daily life.

Reference

Investopedia. *Rounding Error*. Available

at: <https://www.investopedia.com/terms/r/rounding-error.asp> (Accessed: 12 February 2026).

The Open University. *Squares, roots and powers: 4.2 Powers of 10*. Available

at: <https://www.open.edu/openlearn/science-maths-technology/mathematics-statistics/squares-roots-and-powers/content-section-4.2> (Accessed: 12 February 2026).

ScienceDirect. *Computational Mathematics*. Available

at: <https://www.sciencedirect.com/topics/computer-science/computational-mathematics> (Accessed: 12 March 2025).

Wikipedia. (2024). *Natural number*. Available

at: https://en.wikipedia.org/wiki/Natural_number (Accessed: 12 February 2026).

Brilliant. *Integers*. Available at: <https://brilliant.org/wiki/integers/> (Accessed: 12 February 2026).

MathsIsFun. *Rational Numbers*. Available at: <https://www.mathsisfun.com/rational-numbers.html> (Accessed: 12 February 2026).

Catalyst2. *Binary (Base-2) Numbers*. Available

at: <https://www.catalyst2.com/knowledgebase/dictionary/binary-base-2-numbers/> (Accessed: 12 February 2026).

Catalyst2. *Hexadecimal (Base-16) Numbers*. Available

at: <https://www.catalyst2.com/knowledgebase/dictionary/hexadecimal-base-16-numbers/> (Accessed: 12 February 2026).

Tony's Train Exchange. *Decimal-Binary-Hexadecimal, Conversion Chart*. Available

at: https://tonystrains.com/download/DCC_DecBiHex_Chart.pdf (Accessed: 12 February 2026).