IO User Manual

V0.013 (US English)

Revision History

Revision	Date
V0.13 (Pre-release)	24 April 2024

CONTENTS

1 INTRODUCTION

1.1 Intended Audiences

IO was developed to be a tool which would prove useful to many individuals involved with computer technology, but particularly to two groups, computer engineers and students in computer related fields. Of course, its general-purpose features make it useful to anyone who uses a calculator regularly, but there may be more suitable alternatives for people not involved in computer related technologies; but then again, IO may have features that some such users may find particularly useful which are unavailable elsewhere.

Many of the subjects related to the areas where IO can prove useful are scientific in nature, and since all sciences rely heavily on mathematics as a foundation, to fully understand what IO does it is necessary to have a good understanding of many areas of mathematics. Although it is expected that most users have a sound mathematical education this will not always be the case, so this document attempts to provide enough background information on the areas of mathematics used so that any uncertainty or confusion felt when reading this document can hopefully be resolved by referring back to the related definitions of terms, explanation of notation, etc. which should be present in earlier parts of the manual. Many users should be able to skip over these parts of the manual if they already have a good knowledge of the subjects addressed.

The level of detail provided for the mathematical subjects covered is limited to what is required to understand the bulk of the contents of the manual. In some cases, this means that only brief coverage is provided (e.g., the descriptions of what numbers are) as a fully comprehensive treatment would require many more pages of text which would not be justified for the primary purpose of this document. Of course, if this manual sparks an interest in a particular area of mathematics, then the reader can always locate more in-depth information elsewhere; there are lots and lots of books about all manner of mathematical subjects.

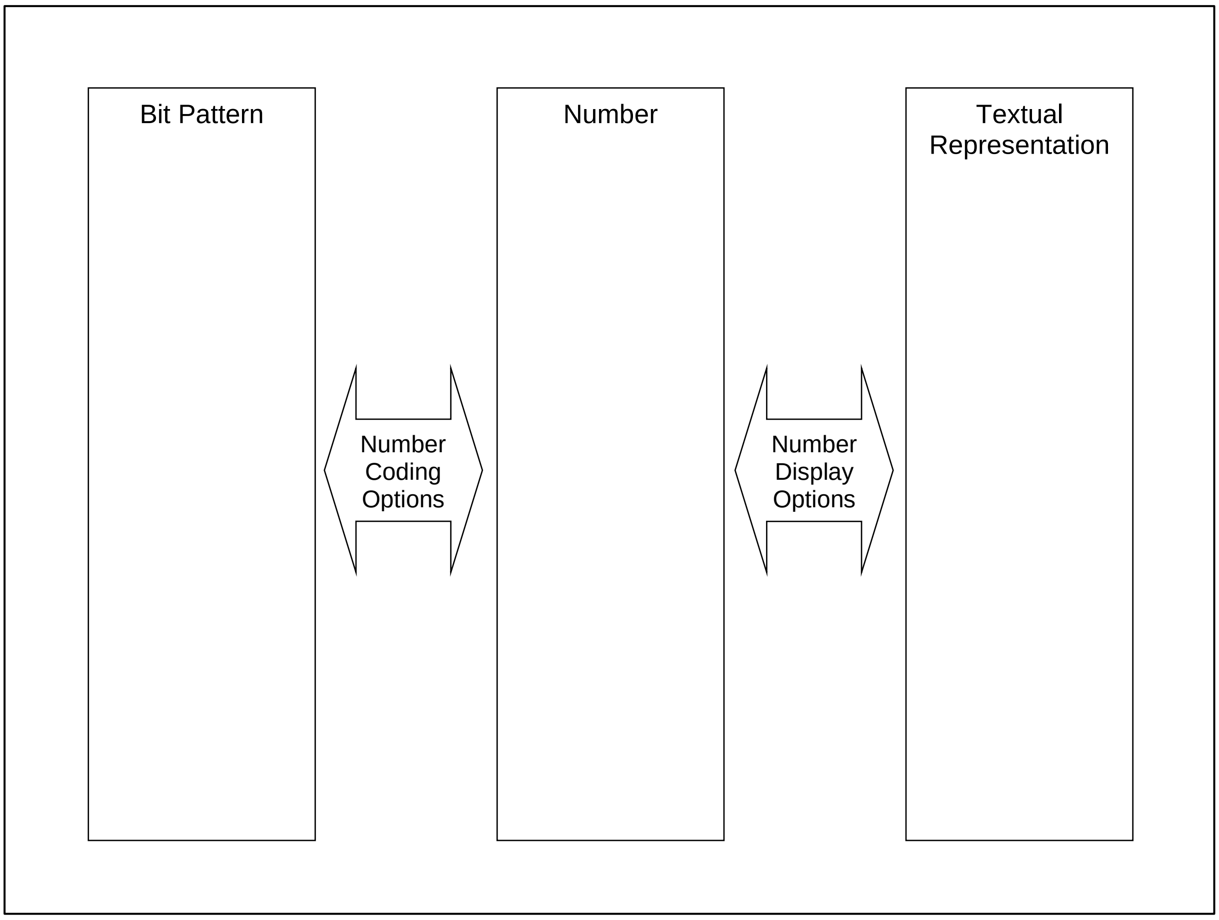
This document attempts to avoid some of the things that that can cause confusion and difficulty of understanding; primarily:

> Assumption of familiarity: many texts use terminology without defining it, assuming that the reader is already familiar with the term and shares the same understanding as the author. This document attempts to provide a definition for technical terms prior to using them.
>
> Consistent use of terminology: it is not uncommon for the same term to mean different things in different texts (and sometimes within the same text). This document attempts to always use previously defined terms, and no other, to refer to the object as defined. Alternative terminologies which the reader may encounter elsewhere are mentioned with the definitions, but are avoided in the main text.

1.2 Bit Patterns, Numbers and Textual Representations

Computer engineers/programmers are always working with numbers, the bit patterns used to represent those numbers inside the hardware, and the textual representation of those numbers displayed on an output device, printed on paper, or hand-written.

Usually, people think of all these things simply as numbers; however, they are three different types of things. It will be beneficial to get a good understanding of the relationships between these things as they will be referred to throughout this document.



The three types are:

Bit Pattern: an ordered sequence of bits, each bit will have one of two settings, usually denoted by 0 and 1.

Number: the abstract entity that everyone is familiar with, but is very complicated to define formally.

Textual Representation: made up of a string of digit characters, and possibly other characters such as sign(s), decimal point, exponent prefix, etc.

IO allows the mappings between a bit pattern and a number to be defined by the setting of Number Coding Options.

IO allows the mappings between a number and a textual representation to be defined by the setting of Number Display Options.

IO also provides textual representations of a Bit Pattern irrespective of the number it represents, but this is not shown on the diagram to keep things simple. For the same reason, the diagram does not show that there is a further mapping from Textual Representation to Display, which is defined by the setting of Number Display Options, Digit Set Options and Color Options.

For example, consider the number twelve.

The bit pattern used to represent the number twelve could be:

00001100 (8-bit unsigned integer coding)

000000010010 (12-bit Binary Coded Decimal integer coding)

01000001010000000000000000000000 (32-bit floating-point coding)

There are many ways of textually representing the number twelve, irrespective of the coding used, some being:

12 (Decimal)

14 (Octal)

C (Hexadecimal)

1.2e1 (Exponent Format)

12e00 (Engineering Format)

1100 (Binary)

1.1b3 (Binary Exponent Format)

It may sound strange if you've never thought about it, but there is no such thing as a decimal (or binary or hexadecimal) number, there are just decimal (or binary or hexadecimal) representations of a number. The base (or radix) is not an attribute of a number; it is an attribute of the textual representation of a number. It's a bit like the English words "The moon" and the Italian words "La luna", there are not different English and Italian moons, they are both referring to exactly the same thing, they are just using different words/letters to represent it in written form.

To hammer home the distinction between a number and its textual representation; the digit "4" is no more the number four than the words "the Empire State Building" are the actual Empire State Building; it is just much easier to make the distinction when the thing being represented is a physical object that you can visualize, something you cannot do with an abstract entity like a number.

This may all seem to be very pedantic, but it will aid understanding throughout this document if you keep in mind the distinctions between these three things rather than think of them all as just numbers.

2 THE BASICS

To fully understand and use the features provided by IO requires an understanding of several subjects, both mathematical and related to the operation of computers and other digital devices.

This chapter provides an introduction to these subjects and defines most of the terminology and notation used throughout the rest of this document

2.1 Sets

2.1.1 Introduction

Sets are probably the most fundamental of all the concepts used in mathematics, the areas of mathematics used in this document are built upon an understanding of sets. There now follows definitions of set theory terms and notation that will be used in this document, note that these are not complete as they only address terms and notation that is used in this document in order not to make things more complicated than they need to be.

2.1.2 Terminology and Definitions

A “set” is a collection of things which are referred to as the “elements” of the set. A set is said to “contain” all its elements. The elements of a set can be anything, and will be specified by the definition of the set. The number of elements in a set may be zero, limited or unlimited. An element can only be in a set once, in other words all the elements of a set are different. If sets contain exactly the same elements, then they are the same set.

A general set will be represented in this document by a serifed uppercase letter (e.g., S, T), possibly with a subscript.

The number of elements in a set is called the “cardinality” of the set.

A set S is said to be a “subset” of a set T if all of the elements in S are contained in T. This implies that a set is a subset of itself.

A set S is said to be a “proper subset” of a set T if it is a subset of T and is not the same set as T. This implies that T has at least one element which is not contained in S.

The “cartesian product” of two sets is the set of all ordered pairs of the elements of the two sets, the first element of the pair being from the first set and the second element of the pair being from the second set.

The “n-fold cartesian product” of n sets is the set of all ordered n-tuples of an element from each of the n sets.

A “mapping” from set S to set T is a rule that, for any element of set S identifies one or more elements of set T. Set T may or may not be the same set as S.

The “domain” of a mapping from set S to set T is the set S, i.e., the set containing all the elements for which the mapping is defined.

A “codomain” of a mapping from set S to set T is the set T, i.e., a set which contains all elements which are identified by the mapping for any element in its domain. A codomain may also contain elements which are not identified by the mapping for any element in its domain.

The “range” of a mapping to set T is the set containing all elements in T which are identified by the mapping for any element in its domain. The range of a mapping is a subset of the codomain of the mapping.

A “function” is a mapping that identifies a single element of its range for each element of its domain. Note that many mathematical texts do not make a distinction between “mapping” and “function”, but this document will only use the term “function” when a mapping satisfies this condition.

The “inverse function” of a function is a mapping that identifies a single element of the domain of the original function for each element of the range of the original function.

Some other commonly used terms related to sets include “Venn diagram”, “union”, “intersection” and “empty set”; as these terms are not used elsewhere in this document they are not defined here. There are plenty of educational resources available for readers to consult if they wish to learn more about sets.

2.1.3 Notation

$a \in S$ denotes that a is an element of set S.

$a \notin S$ denotes that a is not an element of set S.

The contents of any set may be denoted by identifying all its elements between a pair of “{“ and “}” characters, the elements being separated by commas, e.g., { a, b, c } represents the set containing a, b and c. “…” may be used to identify a number of elements when the meaning is obvious, e.g., { 0, 2, 4, … 96, 98 } represents the set of even integers (including zero) less than 100.

An alternative notation for the contents of a set is called “set-builder” notation, it looks like this:

{ “id” ∈ S : “condition(s)” }

where:

id a symbol used to refer to a general element of the set

S is a set to which all elements of the set being defined also belongs, i.e., the set is a subset of S

: is shorthand for “such that” or “where”, some texts use the character | instead

condition(s) is one or more relationships which refer to id, if more than one relationship is specified then they are separated by commas. The set contains only the elements of S for which all the specified relationships are true.

examples:

Note that $\mathbb{R}$ and $\mathbb{Q}$ which are used in the following examples are defined in the section about numbers.

$\{ x \in \mathbb{S} : x \notin \mathbb{T} \}$ is the set containing all the elements of set $\mathbb{S}$ which are not in set $\mathbb{T}$

$\{ x \in \mathbb{R} : x \neq 0 \}$ is the set of all non-zero real numbers

$\{ x \in \mathbb{R} : x > 0, x \notin \mathbb{Q} \}$ is the set of all positive irrational numbers

$|\mathbb{S}|$ denotes the cardinality of the set $\mathbb{S}$.

$\mathbb{S} \times \mathbb{T}$ denotes the cartesian product of the sets $\mathbb{S}$ and $\mathbb{T}$.

So,

$$\mathbb{S} \times \mathbb{T} = \{ (a, b) : a \in \mathbb{S}, b \in \mathbb{T} \}$$

$\mathbb{S}^n$ denotes the n-fold cartesian product of n sets, all of which are $\mathbb{S}$.

So,

$$\mathbb{S}^4 = \{ (a, b, c, d) : a \in \mathbb{S}, b \in \mathbb{S}, c \in \mathbb{S}, d \in \mathbb{S} \}$$

$m: \mathbb{S} \rightarrow \mathbb{T}$ denotes that *m* is a mapping (which may be a function) from set $\mathbb{S}$ to set $\mathbb{T}$, where $\mathbb{S}$ is the domain of the mapping and $\mathbb{T}$ is a codomain. The letter m is used here as an example, any other letter, symbol or sequence of characters may be used to identify the name of a mapping/function.

$f(x)$ denotes the element of the range of the function f identified by applying the function to the element x of its domain. The element x is referred to as the argument of the function. The element identified is often referred to as the element “returned” by the function. When the codomain of a function is a set of numbers then the returned element is commonly referred to as the “value” of the function for the argument.

$f^{-1}(x)$ denotes the element of the domain of the function f which identifies the element x of its range when the function is applied to it, so $f^{-1}(f(x)) = x$. f^{-1} is the inverse function of f.

Some other commonly used notation related to sets include $\subset$, $\subseteq$, $\supset$, $\supseteq$, $\bigcap$, $\bigcup$ and $\varnothing$; as these symbols are not used elsewhere in this document they are not defined here. There are plenty of educational resources available for readers to consult if they wish to learn more about sets.

2.2 Bit Patterns

2.2.1 Introduction

All information stored and processed by a computer (or any other digital device) is represented by bit patterns; hence they will frequently be referred to within this document. There now follows the definitions of terms and notation related to bit patterns that will be used throughout this document.

2.2.2 Terminology and Definitions

A “bit” is a unit of information that can take one of two settings; these two settings are referred to as zero and one. Although they share names with the numbers 0 and 1, bit settings are not numbers so will be denoted in this document by the symbols ⓪ and ① to emphasize the distinction. The symbol ⦿ will be used to denote a bit setting which may be either ⓪ or ①.

A “word” is a unit of storage that is capable of storing an ordered sequence of a specific number of bits. The “width” of a word is the number of bits contained within it.

A “field” is a sequence of consecutive bits within a word. Generally, fields within a word may or may not overlap (i.e., have common bits) but within this document it can be assumed that any fields referred to within a word do not overlap any other referenced fields in that word.

A “bit pattern” is a specific ordered sequence of bit settings that may be stored in a word or a field.

Note that the term “bitmap” is unrelated to a bit pattern, it is Windows terminology for a graphical object represented by a rectangular grid of colored pixels.

When a bit pattern is displayed or written in a document, the constituent bit settings are shown in the conceptual order in which they are stored. The first bit shown (on the left-hand side) is referred to as the “msb” (most significant bit) and the last bit shown (on the right-hand side) is referred to as the “lsb” (least significant bit).

Each bit in a bit pattern is assigned a ”bit number". The lsb has a bit number of zero and the msb has a bit number which is one less than the width of the word containing the bit pattern, the intervening bits are numbered in sequence. IO supports additional bit numbering schemes but this is the scheme used throughout this document.

A “bit value” is a numerical value associated with a bit setting. This value is defined by the function:

$$b: \{ ⓪, ① \} \to \mathbb{N}[0, 1]$$

where b(⓪) = 0 and b(①) = 1. Note that ℕ is defined in the section about natural numbers.

For the sake of brevity, the terms msb and lsb defined above are also used without qualification to refer to the bit values of the msb and lsb; the precise meaning of the term being used will be obvious from the context.

A “byte” is an area of storage which can contain eight bits, no more or less. So, a word or a field with a width of eight is often referred to as a byte.

If a word or fields has a width which is a multiple of eight bits then the first eight bits of the bit pattern it contains (on the left-hand side when displayed) is referred to as the “MSB” (most significant byte) and the last eight bits (on the right-hand side when displayed) is referred to as the “LSB” (least significant byte). Note the use of uppercase characters compared to the lowercase characters used for the most and least significant bits.

A “nibble” is an area of storage which can contain four bits, no more or less. Think of a nibble as being half a byte.

2.2.3 Notation

The notation B is used to denote the set of bit settings, i.e., $B = \{$ ⓪, ① $\}$.

This means that B^N is the set of all bit patterns that may be stored in a word of width N. As each of the N bits in the word may take one of two settings this means that $|B^N| = 2^N$.

The notation b_n is used to denote the bit value of the bit with bit number n within a bit pattern, e.g., the value of the least significant bit of a word is denoted by b_0 and the value of the most significant bit of a byte is denoted by b_7.

2.2.4 Data Representation within Hardware

All information/data stored by a piece of computer/digital hardware is represented within the hardware as words, as defined above. The storage areas will be referred to by different names depending on the type of device, e.g., registers refer to storage areas within CPU or peripheral interface devices, memory locations refer to storage areas within memory devices, etc.

Each bit of this data occupies a (usually) microscopic area (strictly speaking it is a volume) of space, and the setting of the bit is determined by the polarity/magnitude of some physical property of the matter contained within that area. The operation of the hardware is controlled by the laws of physics working on those settings.

Some examples of these properties are:

- The polarity/magnitude of an electrostatic charge stored within a cell of a memory device
- The magnetic polarity/magnitude of an area on the surface of a hard disc or piece of tape
- The reflective properties of an area on an inner surface of an optical disc
- The voltage level of an electrical conductor, e.g., wire within a cable, or track on a circuit board or within a chip

Since these properties are fundamentally analogue in nature, a particular hardware implementation will usually use two pairs of predefined threshold values to determine the setting of a bit. If the associated property for a bit lies between the pair of threshold values defined for ⓪ then the bit is considered to be set to ⓪, if the associated property for a bit lies between the pair of threshold values defined for ① then the bit is considered to be set to ①. So, each pair of threshold values defines a “band” of values for a bit setting.

Although digital devices operate very quickly as far as human comprehension is concerned, an analogue property cannot instantaneously change its value from being in one band to the other, it always takes a finite amount at time. Even though that period of time may be a fraction of a nanosecond this still means that when a bit setting changes there is a period of time when the value is between the threshold bands and so the bit setting is undefined. For this reason, the hardware must be designed so that every time a bit is accessed, or sampled, it is done at a time when the setting of the bit is well defined, not when it may be in the process of changing from one to another. This is achieved by the use of a “clock” which generates internal signals to indicate when it is safe to access a bit, i.e., when, if the hardware is working correctly, the property associated with the bit will lie within one of the two threshold bands. If the property does not lie within one of the threshold bands at that point in time, then the bit setting cannot be determined and it is common for the device to stop functioning as soon as this happens. Given that many digital devices access billions of bits per second without ever encountering this error it is testament to the quality of the work of the engineers involved in the production of our devices that we take this reliability for granted.

2.3 Numbers

2.3.1 Introduction

The formal mathematical definition of a number, and the different types of number, is much more complicated than many think, much more so than would be appropriate for it to be discussed in detail here. However, there now follows a simplified explanation of the four types of number referred to within this document. An understanding of complex numbers is not required.

Before that though, here is a brief discussion of something that is not a number but will often be referred to in the same context as a number; Infinity.

To reiterate: **Infinity is not a number.** It is not a member of any of the sets of numbers which are about to be described. It is a term used when there is no number large enough to define the result of a numerical calculation or satisfy a specified condition.

Infinity is represented by the symbol ∞, this symbol is often used in places in text where numerical digits would be valid but it never represents a numerical value, its use is shorthand to avoid having to use a relatively long-winded sequence of words, e.g., “$1 < x < \infty$“ is shorthand for “x is greater than 1 and has no upper limit”.

It is easy to think of infinity as being a number that is larger than every other number, but this definition does not make sense because: if ∞ is a number then by the definition of addition ($\infty + 1$) is also a number, ($\infty + 1$) is larger than ∞ by 1, therefore ∞ is not larger than every other number.

The term “an infinite number” is commonly used even though it appears to be self-contradictory. It is often used to indicate that the cardinality of some set is unlimited, e.g., “there are an infinite number of negative numbers”. In this document the more “correct” term “an unlimited number” is used instead.

Similarly, the floating-point codings, which are described in detail later in this document, use the term “Infinity Class” to refer to bit patterns used to identify any number which is larger than the largest number which can be represented using the coding. The term “Overflow Class” would have been more mathematically precise, but it’s much too late now to lobby for such a change.

Finally, you will often see notation like (see the next section for a description of summation notation)

$$\sum_{\mathrm{r}=0}^{\infty} \mathrm{T_r}$$

which implies that ∞ is an integer number, however this is just a commonly used shorthand notation for:

$$\lim_{N\to\infty} \sum_{\mathrm{r}=0}^{\mathrm{N}} \mathrm{T_r}$$

i.e. the limit of the sum no matter how large the value of N. Mathematical limits is one of those topics which is too complicated to be described in detail in a document like this; however, there are plenty of publications available which will provide this detail if you are interested.

2.3.2 Terminology, Notation and Definitions

The following descriptions of the different types of numbers start with an assumption of some knowledge of several mathematical concepts which nearly everyone has, but not the underlying mathematically rigorous definitions of these things. These concepts are: the numbers 0 and 1, the operations of addition, subtraction, multiplication, division and exponentiation (raising to a power).

When a division is to be performed the number being divided is called the "dividend" and the number it is being divided by is called the "divisor".

Note that throughout this document, the terms "number", "value" and "numerical value" are all used to mean the same thing.

Now is a good time to introduce some terminology and notation which will be used later in this document, as it is only used when dealing with numbers.

"summation notation" uses the Greek letter $\sum$ as follows:

$$\sum_{r=N_1}^{N_2} T_r$$

"r" can be any convenient letter as it is only used in the "T_r" part to denote an integer value in the specified range.

"N_1" can be either an integer value or $-\infty$. "N_2" can be either an integer value greater than or equal to N_1, or ∞, or $+\infty$.

"T_r" is a mathematical expression, which refers to r (or whatever letter is used), which is used to evaluate the "term" to be added.

Here are a couple of examples of its use:

$$\sum_{r=1}^{4} x^r = x + x^2 + x^3 + x^4$$

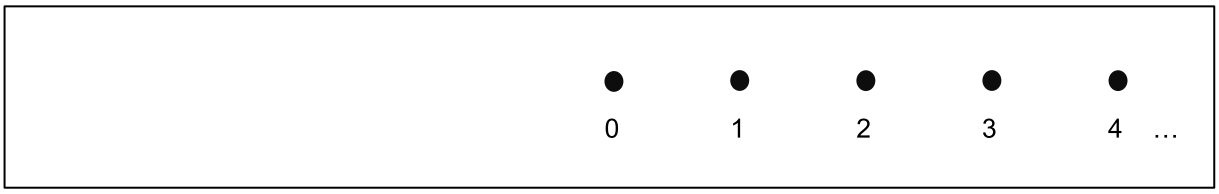
$$\sum_{n=0}^{\infty} \frac{(n+1)}{x^{2n}} = 1 + \frac{2}{x^2} + \frac{3}{x^4} + \frac{4}{x^6} + \cdots$$

A "polynomial" function, say P, is a function whose mapping is defined by:

$$P(x) = \sum_{n=0}^{N} c_n x^n = c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N$$

The value of N is called the "degree" of the polynomial, and the C_n values are called the "coefficients" of the terms. It can usually be assumed that C_N is not equal to zero. C_0 is called the "constant term". Values of *x* for which $P(x) = 0$ are called the "roots" of the polynomial P.

2.3.2.1 Natural Numbers



The set of natural numbers is denoted by the symbol ℕ.

It contains the elements 0 and 1, and others which are defined by adding 1 to previously defined elements, e.g., 2 is defined as (1 + 1), 3 is defined as (2 + 1), etc.

The above diagram depicts elements of ℕ, each element is depicted as a dot, and the dot for each element is positioned immediately to the left of the dot representing the element defined by adding 1 to it.

This sequence illustrates the following relationships between any two elements: the leftmost element of the two is “less” than the rightmost, and the rightmost of the two is “greater” than the leftmost. These definitions also apply to all the other types of number which are described in this chapter.

Elements next to each other in the diagram are said to be “consecutive” numbers. All pairs of consecutive natural numbers have no natural number(s) between them on the diagram. There are a finite number of natural numbers between any two different non-consecutive natural numbers.

There are an unlimited number of elements of ℕ.

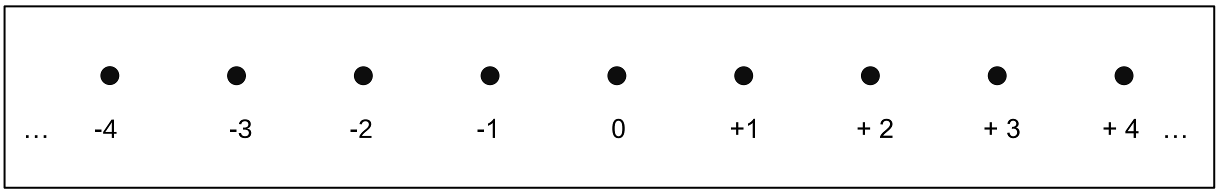
All elements of ℕ are greater than or equal to 0, hence 0 is the lower limit of the natural numbers.

Applying addition or multiplication to any two elements of ℕ always identifies an element of ℕ as the result.

Applying subtraction or division to any two elements of ℕ may or may not identify an element of ℕ as the result.

When the characters “N” or “n” are used to represent a number, it is usually, but not always, implied that the number is a natural number.

2.3.2.2 Integer Numbers



The set of integer numbers is denoted by the symbol ℤ.

The main difference between ℤ and ℕ is:

Applying subtraction to any two elements of ℕ always identifies an element of ℤ as the result.

Also:

ℕ is a proper subset of ℤ.

Unlike ℕ, ℤ has no lower limit.

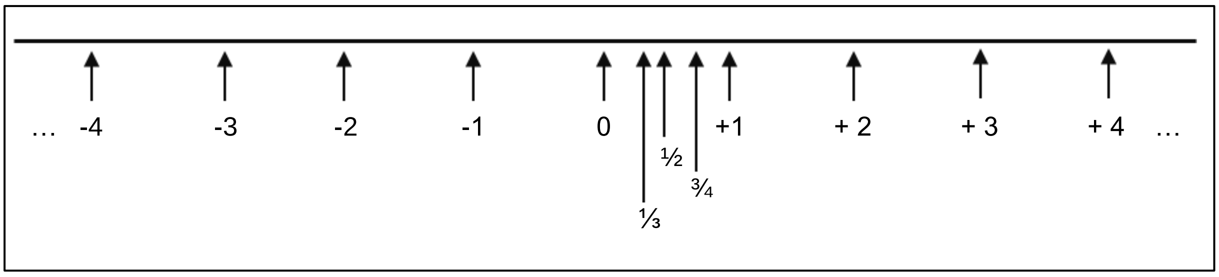
Applying addition, subtraction or multiplication to any two elements of ℤ always identifies an element of ℤ as the result.

Applying division to any two elements of ℤ may or may not identify an element of ℤ as the result.

The definition of consecutive numbers for natural numbers also applies to integer numbers.

The numbers to the right of 0 on the diagram are called "positive" numbers; the numbers to the left of 0 are called "negative" numbers. These definitions also apply to all the following types of number described in this chapter.

2.3.2.3 Rational Numbers



The set of rational numbers is denoted by the symbol ℚ.

The main difference between ℚ and ℤ is:
Applying division to any two elements of ℤ identifies an element of ℚ as the result if the divisor is not 0. Division by 0 is not defined.

Also:

ℤ is a proper subset of ℚ.

There are an unlimited number of elements of ℚ between any two different elements of ℚ.

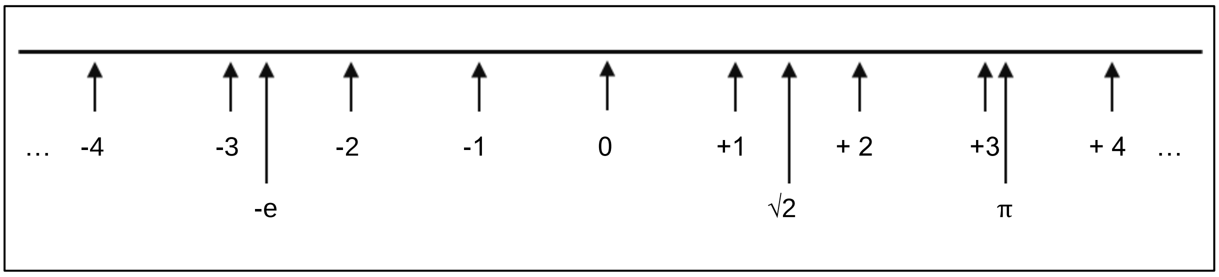
There are no consecutive elements of ℚ, any two different rational numbers always have an unlimited number of other rational numbers between them on the above diagram; this is why the rational numbers are represented by a horizontal line instead of a sequence of individual dots.

Each rational number is represented by a point on the number line; a point does not have a size; a point does have a distance from any other point on the line.

Applying addition, subtraction, multiplication or division to any two elements of ℚ identifies an element of ℚ as the result except in the case of division by 0.

Every element of ℚ is the result of dividing an element of ℤ by another element of ℤ, the divisor may be the same as the dividend, but will not be 0. The pair of elements of ℤ is not unique; there are an unlimited number of such pairs for each element of ℚ.

2.3.2.4 Real Numbers



The set of real numbers is denoted by the symbol ℝ.

The main difference between ℝ and ℚ is:
Not every element of ℝ is the result of dividing an element of ℤ by another element of ℤ.

Elements of ℝ which are not elements of ℚ are called “irrational” numbers.

ℚ is a proper subset of ℝ.

Applying addition, subtraction, multiplication or division to any two elements of ℝ identifies an element of ℝ as the result except in the case of division by 0.

If X is a real number and there does not exist any polynomial function with integer coefficients which has X as a root, then X is called a “transcendental” number. All transcendental numbers are irrational, but not all irrational numbers are transcendental.

In the above diagram, π and -e are transcendental numbers, √2 is not transcendental, all three are irrational.

2.3.3 <u>Notation</u>

In addition to the symbols defined above, this document will use the symbol ℙ to denote the set of positive and negative numbers whose magnitudes are sums of a finite number of powers (positive and/or negative) of 2. ℤ is a proper subset of ℙ, and ℙ is a proper subset of ℚ.

More precisely:

$$\mathbb{P} = \left\{ x \in \mathbb{Q} : x = \pm \sum_{n=N_1}^{N_2} b_n 2^n,\ b_n \in \mathbb{N}[0,1] \right\}$$

The following notation, called "interval notation", is used to represent proper subsets of the above sets of numbers which contain the following elements:

$\mathbb{S}(a,b)$: elements of $\mathbb{S}$ with a value greater than a and less than b

$\mathbb{S}[a,b)$: elements of $\mathbb{S}$ with a value greater than or equal to a and less than b

$\mathbb{S}(a,b]$: elements of $\mathbb{S}$ with a value greater than a and less than or equal to b

$\mathbb{S}[a,b]$: elements of $\mathbb{S}$ with a value greater than or equal to a and less than or equal to b

$(a$ can be replaced with $(-\infty$ to indicate that there is no lower limit

$b)$ can be replaced with $\infty)$ or $+\infty)$ to indicate that there is no upper limit

2.4 Textual Representations

2.4.1 Introduction

IO provides options to allow many different ways of representing numerical values in its display areas.

This section describes these and provides some background information.

In our everyday experience, practically all numbers are represented textually by an ordered sequence of characters, or character strings. How each character is interpreted depends on a number of different things which will be discussed shortly. One thing that all such character strings have in common is that each digit character conveys a value that is greater than the value conveyed by the next digit in the string, if any, by a fixed factor. The most common exception to this is the use of roman numerals on some clock faces, but as IO does not support roman numerals this will be ignored.

IO allows you to set a large number of Number Display Options which determine how a value will be displayed. More formally, they define a mapping from a set of numbers to a set of character strings. They are also used to define an inverse mapping which is used to identify the value represented by an entered sequence of keystrokes from the keypad/keyboard or from a character string pasted from the clipboard.

2.4.2 Terminology and Definitions

A “binary digit” is one of the characters “0” or “1”

An “octal digit” is one of the characters “0”, “1”, “2”, “3”, “4”, “5”, “6” or “7”

A “decimal digit” is one of the characters “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8” or “9”

A “hexadecimal digit” is one of the characters “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “a”, “b”, “c”, “d”, “e”, “f”, “A”, “B”, “C”, “D”, “E” or “F”

A “sign character” is one of the characters “+” or “-“

A “decimal point” and a “binary point” are each one of the characters “.”, “·” or “,”

An “exponent prefix” is one of the characters “e”, “E”, “b”, “B” or “ “

A “digit group separator” is one of the characters “,”, “ ’ ” or “ “

A “number base” is a natural number with a value greater than 1 which dictates how strings of digits are interpreted to determine the numerical value they represent. The term “radix” is also commonly used to refer to a number base.

A “digit” usually means the subset of the hexadecimal digits which are valid for the number base in use, but sometimes also includes the other characters defined above which may be present in a display area.

2.4.3 Integer Number Representations

Consider the character string “123”. Everyone knows that in everyday usage the character “1” does not represent the value 1, it actually represents the value 100. Similarly, the character “2” does not represent the value 2, it actually represents the value 20. And the whole string represents the value 100 + 20 + 3.

Expressing this in mathematical terms for a general string of digit characters "$C_{N-1}C_{N-2} \ldots C_1C_0$",

where N is the number of digit characters, and d_n is the unscaled value of the character C_n;

$$\text{Value} = \sum_{n=0}^{N-1} d_n 10^n$$

Note that any number raised to the power of zero is one.

Of course, this assumes that the value is being expressed in decimal, i.e., with a number base of ten.

This can easily be generalized for an arbitrary number base of b, as follows

$$\text{Value} = \sum_{n=0}^{N-1} d_n b^n$$

IO allows you to specify any number base from 2 to 16.

A negative value is represented by using the character string representing the magnitude of the value prepended with a "-" character.

The character string representing a positive value may optionally be prepended with a "+" character.

An important property of these representations is that every integer number, no matter how large, has a unique textual representation, notwithstanding the optional "+" prefix, for any particular number base.

Textual representations that use number bases of two, eight, ten and sixteen are called binary, octal, decimal and hexadecimal respectively. The other number bases also have such names, but they are rarely used.

Suffixes are sometimes used to identify the number base to be applied to a character string where clarification is useful. e.g., $151_{10} = 97_{16} = 10010111_2$. The number base is always expressed in decimal.

The alphabetic characters A, B, C, D, E and F are used as digit characters for number bases greater than ten to represent the digit values 10_{10}, 11_{10}, 12_{10}, 13_{10}, 14_{10} and 15_{10}. Uppercase or lowercase letters may be used.

Number bases which are a power of two (IO supports two, four, eight and sixteen) are particularly useful to computer engineers because when used to represent a value which is stored internally using an unsigned form of integer coding (this is described in detail in a later section) each digit character will identify the bit settings of a group of bits within the bit pattern.

2.4.4 Real Number Representations

To represent non-integer values a "point" character can be included in a character string to mark the boundary between integer digits (on the left of the point) and fractional digits (on the right of the point).

Our definition becomes:

For the general string of characters “$C_N C_{N-1} \ldots C_1 C_0 \, . \, C_{-1} C_{-2} \ldots C_{-M}$”,

where N is non-negative, M is positive, d_n is the unscaled value of the character C_n, and b is the number base;

$$\text{Value} = \sum_{n=-M}^{N} d_n b^n$$

Note that $x^{-y} = \frac{1}{x^y}$

IO supports number bases of two (binary) and ten (decimal) for representations of non-integer values. This restriction also applies to integer values stored using a non-integer coding. IO also supports a hexadecimal representation (defined in IEEE 754) which is described later as it does not conform to the general representation currently being described.

An important distinction from the integer case is that not all rational (and therefore not all real) numbers may be accurately represented by a binary or decimal textual representation.

If the value to be represented is too large or too small to be conveniently represented in this way, an alternative representation, called “exponent format”, is available.

It looks like this:

“$C_N C_{N-1} \ldots C_1 C_0 \, . \, C_{-1} C_{-2} \ldots C_{-M} \;\; P \;\; E_{T-1} \ldots E_0$”

P is an exponent prefix character, for which IO uses “E” or “e” for decimal representations, “B” or “b” for binary representations, or “ “ (space) for either. E_n are exponent characters.

The string is used to define two values, a “mantissa” value, and an “exponent” value, as follows;

where N, M and d_n are as before, T is the number of exponent characters, and e_n is the unscaled value of the character E_n;

$$\text{Mantissa Value} = \sum_{n=-M}^{N} d_n b^n \qquad \text{Exponent Value} = \sum_{n=0}^{T-1} e_n 10^n$$

The value being represented is: (Mantissa Value) x b $^{\text{(Exponent Value)}}$

where b is the number base. Note that the exponent value is always represented in decimal.

For any number there are an unlimited number of representations using the exponent format because the exponent value can be any integer value, the point just needs to be positioned so that the mantissa value results in the required value being represented.

A “normalized” exponent format representation is one where there is only one digit to the left of the point, and that digit is not 0. For any non-zero value there is only one normalized exponent format representation. IO will default to normalized exponent format displays of results of calculations, but allows the user to change the display to a non-normalized format.

The term “significand” is usually synonymous with mantissa; however, in this document it will only be used to refer to a mantissa of a binary exponent representation which has one binary digit to the left of the binary point.

2.4.4.1 Hexadecimal Significand Format

IO supports this format, but only for copying to/pasting from the clipboard, it does not use it for onscreen display. This allows for the transfer of these strings to/from source code of programming languages which support it.

Character strings in this format follow that described above for a base of sixteen, with the following differences:

> The character string is always prefixed with either 0x or 0X
>
> If present, an exponent value is prefixed with either p or P
>
> If present, an exponent value, which is always expressed in decimal, is an exponent of 2 (not 16) that the mantissa must be multiplied by to get the represented value, i.e., the exponent value specifies the number of bits by which the point is shifted, not the number of hexadecimal digits (each of which represents four bits).

If the value to be represented is an integer which can be stored in 64 bits then IO will generate a string with no point and no exponent, otherwise IO will generate a string with a mantissa starting with an integer part of 1 and with the remainder of the mantissa and exponent value as required to define the value.

<u>Examples</u>

1) 123 (decimal)
1111011 (binary)
0x7B (hexadecimal)

2) 5.208984375 (decimal)
101.001101011 (binary)
0x1.4d6p2 (hexadecimal)

3) -3.125e-2 (decimal)
-1b-05 (binary)
-0x1p-5 (hexadecimal)

2.4.4.2 Recurring Fractional Digits

As already mentioned, not all rational numbers may be able to be represented using the real number representation described above. Very often the result of a division would require an unlimited number of a recurring sequence of digits after the point to represent the required number. Some examples are:

1/3 = 0.333333333333333…

3/14 = 0.2142857142857142857…

1/10 (decimal) = 0.000110011001100110011001100110011001100110011001100110011… (binary)

These examples illustrate the following:

> Whether or not a number requires a recurring representation may depend on the base used, the value 1/10 in decimal is 0.1 which is not recurring. IO only supports binary and decimal for real numbers; for these, a number with a non-recurring binary representation will have a non-recurring decimal representation; the converse is not true as shown by this example.
>
> There may be a number of digits immediately after the point before digits start to recur.
>
> Which group of digits is considered to recur is not unique. In the 3/14 example you can consider the 142857 digits to recur after the first 2 digit, or the 285714 digits to recur after the 214 digits. To make the terminology refer to a unique set of digits we will always consider the case with the minimum number of initial digits after the point before recurring starts as the one which defines which digits recur. So, in the third example, the recurring digits are not the 1100 digits following the 000 digits, but the 0011 digits following the first fractional 0 digit.

In this document the notation to denote recurring fractional digits is to underline the first occurrence and not show those following, as follows:

$1/3 = 0.\underline{3}$

$3/14 = 0.2\underline{142857}$

$1/10$ (decimal) $= 0.0\underline{0011}$ (binary)

Note that many texts adopt a different notation, putting a dot above the first and last digits of a recurring sequence, e.g.,

$3/14 = 0.2\dot{1}4285\dot{7}$

Determining Equivalent Rational Number

Consider a fractional value represented in base b by the N recurring digits $d_1d_2..d_N$ immediately following the point, then

$$\textbf{Value} = 0\ .\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N \ldots$$

If we multiply this by b^N we get

$$\textbf{Value} \times b^N = d_1\,d_2\ldots d_N\ .\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N \ldots$$

Subtracting gives us

$$\begin{aligned}(\textbf{Value} \times b^N) - \textbf{Value} &= d_1\,d_2\ldots d_N\ .\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N \ldots\\ &\quad -\ 0\ .\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N\ d_1\,d_2\ldots d_N \ldots\end{aligned}$$

So,

$$\textbf{Value}\ (b^N - 1) = d_1\,d_2\ldots d_N$$

and,

$$\textbf{Value} = (d_1\,d_2\ldots d_N) / (b^N - 1)$$

Therefore,

$$\mathbf{0.\underline{d_1\,d_2\ldots d_N} = (d_1\,d_2\ldots d_N)\,/\,(b^N - 1)}$$

This result can be used to determine the equivalent rational number of any value with recurring fractional digits.

e.g.

$0.2\underline{142857}_{10}$ $= 0.2 + (\ 0.\underline{142857} / 10\ )$

$= (2 / 10) + (\ (\ 142857 / (10^6 - 1)\ )\ ) / 10\ )$ { by applying the above result }

$= (2 / 10) + (\ (142857 / 999999) / 10\ )$

$= (2 / 10) + (142857 / 9999990)$

$= (2 / 10) + (1 / 70)$ { you can use IO's HCF operator to help with this step }

$= (14 / 70) + (1 / 70)$

$= 15 / 70 = 3 / 14$

$0.0\underline{0011}_2$ $= 0.0_2 + (\ 0.\underline{0011}_2 / 10_2\ )$

$= (\ 0011_2 / (2^4 - 1)\ )\ ) / 2$ { by applying the above result }

$= (3 / 15) / 2$

$= 3 / 30 = 1 / 10$

$0.000\underline{1100}_2$ $= 0.000_2 + (\ 0.\underline{1100}_2 / 1000_2\ )$

$= (\ 1100_2 / (2^4 - 1)\ )\ ) / 8$ { by applying the above result }

$= (12 / 15) / 8$

$= 12 / 120 = 1 / 10$

2.4.5 Bit Patterns

By default, IO will display bit patterns as a string of 0 and 1 digit characters to represents bit settings of ⓪ and ① respectively. One digit will be displayed for each bit in the word, so there will be leading zero(s) if the msb of the bit pattern is set to ⓪.

IO also allows the use of bit pattern digit sets; this will replace the 0 and 1 digits with a pair of graphical representations of the ⓪ and ① bit settings.

3 NUMBER CODING

3.1 General Coding Principles

When the bit pattern contained in a word is used to represent a numerical value, the value represented by a bit pattern is defined by the “coding” being used. Hence, a coding is a mapping from the set of possible bit patterns to a set of numbers. The coding also contributes to the definition of the inverse mapping, i.e., from a set of numbers to a bit pattern; however, whereas the mapping identifies a single number for each bit pattern the inverse mapping may identify the same bit pattern for more than one number. Also, the inverse mapping for fixed/floating-point codings may be affected by the “Fixed/Floating-point Rounding Type” being used.

As the number of bits stored within any device is limited and the number of numbers is unlimited it is obvious that not all numbers may be represented using any particular coding, so the range of a coding will be a proper subset of a set of numbers.

Over the decades computers have used many types of coding, some of which are assumed by the low-level instructions of the central processing unit in order to perform numerical calculations, others require software to perform calculations if the instruction set of the CPU does not support the required calculations using the coding.

IO supports three types of number coding; integer, fixed-point and floating-point. IO supports a number of coding forms for each type, as follows:

Integer		
Coding Form	**Number of Codings**	**Total**
Unsigned	96	
2’s Complement	95	
Excess	95	
Binary Coded Decimal	84	
		370
Fixed-point		
Coding Form	**Number of Codings**	**Total**
Unsigned	63,680	
2’s Complement	41,580	
		105,260
Floating-point		
Coding Form	**Number of Codings**	**Total**
16-bit	5	
32-bit	5	
64-bit	5	
80-bit	4	
		19
Totals		
10		105,649

3.2 Integer Coding

3.2.1 Introduction

Integer codings map from the set of bit patterns that may be stored in a word to a proper subset of the integer numbers.

IO supports four forms of integer codings: Unsigned, 2’s Complement, Excess and Binary Coded Decimal (BCD).

The integer codings supported by IO have the following properties:

- Except for BCD codings, all bit patterns in the domain have a mapping defined
- The mapping is a function, i.e., different bit patterns map to different numbers
- The range consists of all the integer values between an upper and lower limit, i.e., there is no number between these limits which is not represented by a bit pattern in the domain
- The above implies that the inverse mapping is also a function, i.e., different values in the range will be represented by different bit patterns

3.2.2 Integer Unsigned Coding

3.2.2.1 Description

This is the simplest form of coding supported by IO. If a bit pattern’s bit values are written out as a string of 0’s and 1’s then that string interpreted as a binary number will be the number identified by unsigned form of coding applied to the bit pattern.

The unsigned form of integer coding is qualified by a single parameter, the width of the word, to fully define the coding.

Integer unsigned coding is supported for word widths from 1 to 96.

3.2.2.2 Mathematical Details

Where m is the integer unsigned coding mapping and N is the word width:

$$N \in \mathbb{N}[1, 96]$$

$$m: \mathbb{B}^N \rightarrow \mathbb{N}[0, 2^N)$$

m is a function, $\mathbb{B}^N$ is its domain, $\mathbb{N}[0, 2^N)$ is its range.

$$m^{-1}: \mathbb{N}[0, 2^N) \rightarrow \mathbb{B}^N$$

m^{-1} is a function, $\mathbb{N}[0, 2^N)$ is its domain, $\mathbb{B}^N$ is its range.

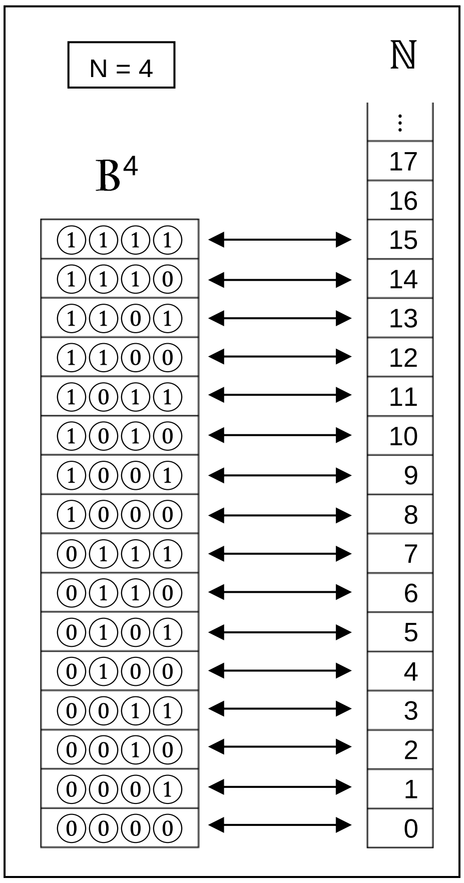
Where $p \in \mathbb{B}^N$ and b_n are the bit values of the bits within p:

$$m(p) = \sum_{n=0}^{N-1} b_n 2^n$$

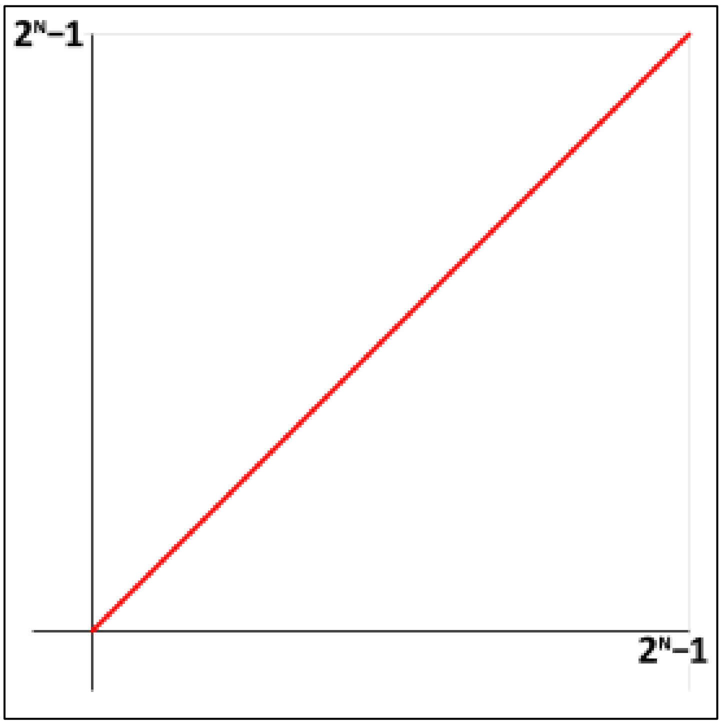
For example, for N = 8:

$$m(p) = b_0 + 2b_1 + 4b_2 + 8b_3 + 16b_4 + 32b_5 + 64b_6 + 128b_7$$

3.2.2.3 Illustrations



Example Mapping



Value mapping from unsigned integer coding values (for comparison with other codings)

3.2.3 Integer 2’s Complement Coding

3.2.3.1 Description

The 2’s complement form of integer coding is qualified by a single parameter, the width of the word, to fully define the coding.

Integer 2’s complement coding is supported for word widths from 2 to 96.

3.2.3.2 Mathematical Details

Where m is the integer 2’s complement mapping and N is the word width:

$$N \in \mathbb{N}[2, 96]$$

$$m: \mathbb{B}^N \rightarrow \mathbb{Z}[-2^{N-1}, 2^{N-1})$$

m is a function, $\mathbb{B}_N$ is its domain, $\mathbb{Z}[-2^{N-1}, 2^{N-1})$ is its range.

$$m^{-1}: \mathbb{Z}[-2^{N-1}, 2^{N-1}) \rightarrow \mathbb{B}^N$$

m^{-1} is a function, $\mathbb{Z}[-2^{N-1}, 2^{N-1})$ is its domain, $\mathbb{B}^N$ is its range.

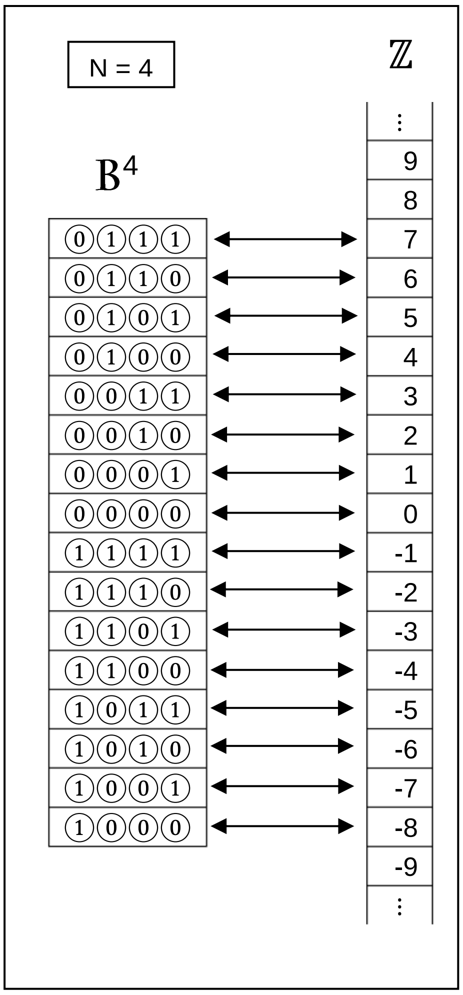
Where $p \in \mathbb{B}^N$ and b_n are the bit values of the bits within p:

$$m(p) = -2^{N-1} b_{N-1} + \sum_{n=0}^{N-2} b_n 2^n$$

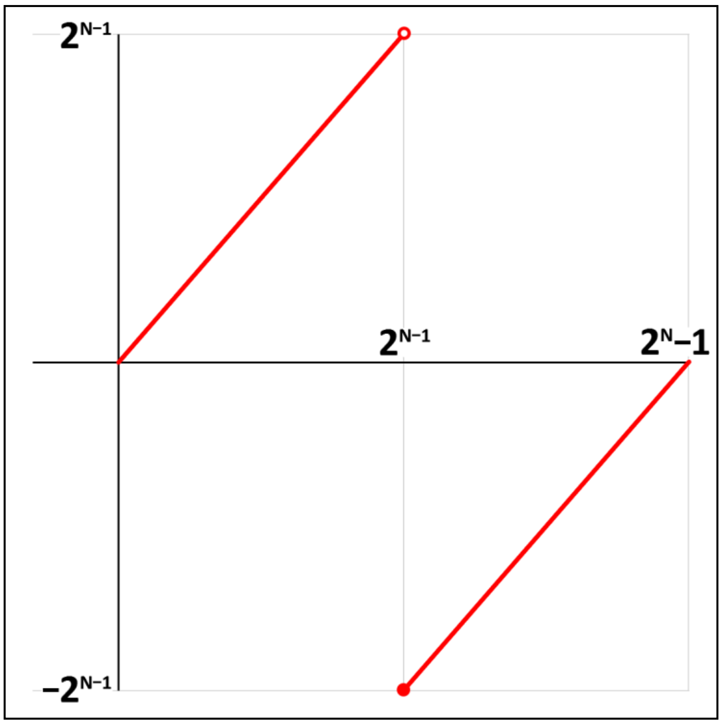
For example, for N = 8:

$$m(p) = (b_0 + 2b_1 + 4b_2 + 8b_3 + 16b_4 + 32b_5 + 64b_6) - 128b_7$$

3.2.3.3 Illustrations



Example mapping



Value mapping from unsigned integer coding values

3.2.4 Integer Excess Coding

3.2.4.1 Description

Excess coding is also referred to as biased or offset coding; for the sake of consistency only the term excess coding will be used in this document.

The excess form of integer coding is qualified by a single parameter, the width of the word, to fully define the coding.

Integer excess coding is supported for word widths from 2 to 96.

Excess coding is rarely, if ever, used by application software and is not supported by many, if any, programming languages; however, it is used to encode exponent values used by the floating-point codings so it is supported by IO to permit the display of these values and easily allow arithmetic to be performed on these values. See the sections on floating-point coding for further details.

Excess coding differs from all the other codings supported by IO in that it does not map a bit pattern containing all ⓪s to a value of zero.

3.2.4.2 Mathematical Details

For a word width of N, integer excess coding maps B^{N} to a proper subset of $\mathbb{Z}$.

Where m is the integer excess mapping and N is the word width:

$$N \in \mathbb{N}[2, 96]$$

$$m: \mathrm{B}^{N} \rightarrow \mathbb{Z}(-2^{N-1}, 2^{N-1}]$$

m is a function, B^{N} is its domain, $\mathbb{Z}(-2^{N-1}, 2^{N-1}]$ is its range.

$$m^{-1}: \mathbb{Z}(-2^{N-1}, 2^{N-1}] \rightarrow \mathrm{B}^{N}$$

m^{-1} is a function, $\mathbb{Z}(-2^{N-1}, 2^{N-1}]$ is its domain, B^{N} is its range.

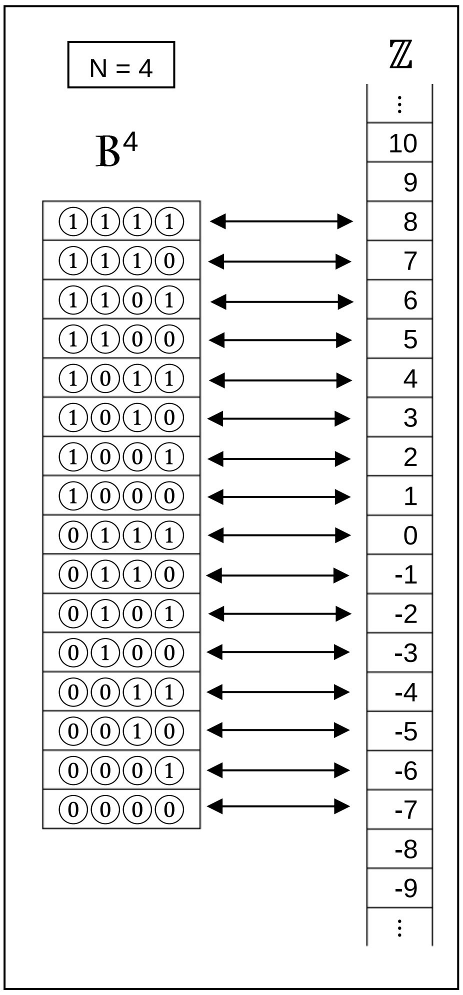
Where $p \in \mathrm{B}^{N}$ and b_n are the bit values of the bits within p:

$$\mathrm{m}(\mathrm{p}) = -(2^{\mathrm{N}-1} - 1) + \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{b}_{\mathrm{n}} 2^{\mathrm{n}}$$

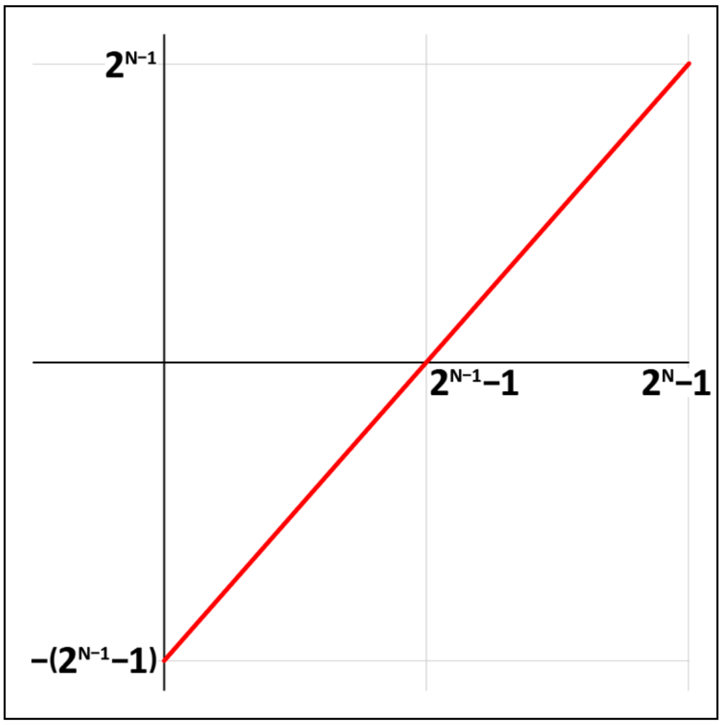
For example, for N = 8:

$$m(p) = (b_0 + 2b_1 + 4b_2 + 8b_3 + 16b_4 + 32b_5 + 64b_6 + 128b_7) - 127$$

3.2.4.3 Illustrations



Example mapping



Value mapping from unsigned integer coding values

3.2.5 Integer Binary Coded Decimal (BCD) Coding

3.2.5.1 Description

The BCD form of integer coding is qualified by two parameters, the width of the word and the number of bits per digit, to fully define the coding.

Integer BCD coding is supported for word widths from 4 to 96, and numbers of bits per digit of 4, 5, 6, 7 and 8. The word width must be a multiple of the number of bits per digit.

3.2.5.2 Mathematical Details

Where m is the integer BCD mapping, N is the word width and d is the number of bits per digit:

Each bit pattern in $\mathbb{B}^N$ is partitioned into non-overlapping fields of width d, with no gaps between the fields.

There is no mapping defined for a bit pattern if the least significant four bits in any field are set to ①⓪①⓪, ①⓪①①, ①①⓪⓪, ①①⓪①, ①①①⓪ or ①①①①.

There is no mapping defined for a bit pattern if d> 4 and any field does not have all its bits outside the least significant four all set to ⓪

$d \in \mathbb{N}[4, 8]$

$N \in \{ n \in \mathbb{N}[4, 96] : (n/d) \in \mathbb{N} \}$

$m: \mathbb{B}^N \rightarrow \mathbb{N}[0, 10^{N/d})$

m is a function, its domain is a proper subset of $\mathbb{B}^N$, $\mathbb{N}[0, 10^{N/d})$ is its range.

$m^{-1}: \mathbb{N}[0, 10^{N/d}) \rightarrow \mathbb{B}^N$

m^{-1} is a function, its domain is $\mathbb{N}[0, 10^{N/d})$, its range is a proper subset of $\mathbb{B}^N$.

Where $p \in \mathbb{B}^N$, and there is a mapping defined for p (as described above) and b_n are the bit values of the bits within p:

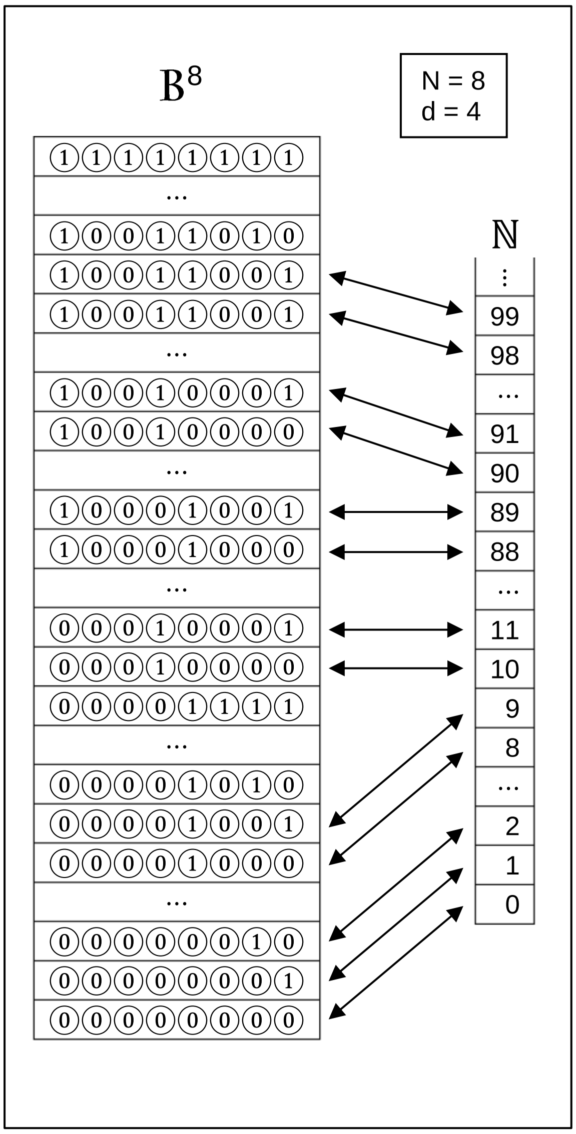
$$m(p) = \sum_{n=0}^{\frac{N}{d}-1} (b_{dn} + 2b_{dn+1} + 4b_{dn+2} + 8b_{dn+3})10^n$$

Note that dn represents “d multiplied by n”, n is not being used as a subscript of d.

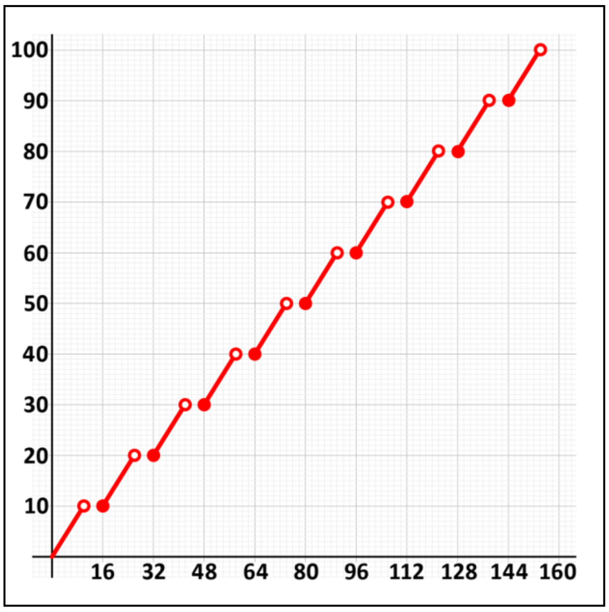
For example, for N = 8, d = 4:

$$m(p) = (b_0 + 2b_1 + 4b_2 + 8b_3) + 10(b_4 + 2b_5 + 4b_6 + 8b_7)$$

3.2.5.3 Illustrations



Example mapping



Value mapping from unsigned integer coding values (N = 8, d = 4)

3.3 Fixed-Point Coding

3.3.1 Introduction

Fixed-point coding is a relatively simple way of representing non-integer values. It can also be used to represent larger integer values than are made available by the integer codings.

The basic idea is that the value represented by a bit pattern is the value it represents using the corresponding integer coding, multiplied by a power of 2. The power of 2 used is specified as a second qualifier, in addition to the word width, called the “precision” of the coding. The precision will be positive if it is desired to represent non-integer values and negative if it is desired to represent larger integer values. You can specify a precision of zero, but that will result in an identical coding to the corresponding integer coding.

The reason it is named fixed-point is because the precision fixes the position of a notional binary point within (or outside) the bit pattern.

Although they are rarely used nowadays, their simplicity meant that they were commonly used in earlier computer systems where non-integer values within a well-defined range were required, e.g., telemetry systems.

IO supports two forms of fixed-point codings, Unsigned and 2’s Complement.

3.3.2 Fixed-Point Unsigned Coding

3.3.2.1 Description

The unsigned form of fixed-point coding is qualified by two parameters, the width of the word and the precision, to fully define the coding.

Fixed-point unsigned coding is supported for word widths from 1 to 64 and precisions from -99 to +99.

3.3.2.2 Mathematical Details

Where m is the fixed-point unsigned mapping, N is the word width and r is the precision:

$$N \in \mathbb{N}[1, 64]$$

$$r \in \mathbb{Z}[-99, 99]$$

$$m: \mathbb{B}^N \rightarrow \mathbb{P}[0, 2^{N-r})$$

m is a function, its domain is $\mathbb{B}^N$, its range is a proper subset of $\mathbb{P}[0, 2^{N-r})$

$$m^{-1}: \mathbb{R}[0, 2^{N-r}) \rightarrow \mathbb{B}^N$$

m^{-1} is a function, its domain is $\mathbb{R}[0, 2^{N-r})$, its range is $\mathbb{B}^N$

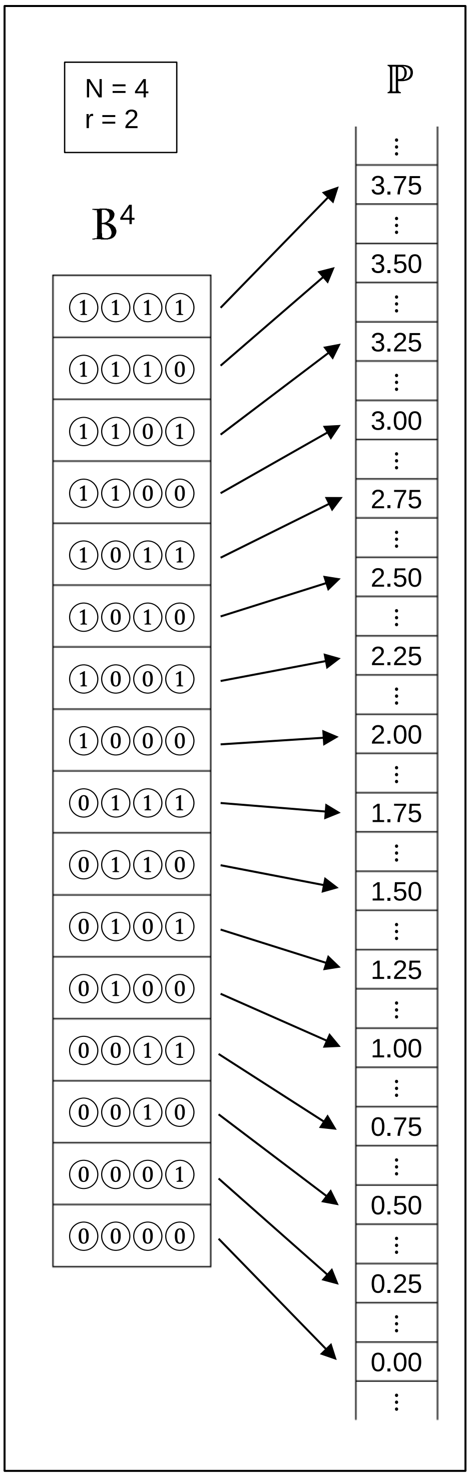
Where $p \in \mathbb{B}^N$ and b_n are the bit values of the bits within p:

$$m(p) = 2^{-r} \sum_{n=0}^{N-1} b_n 2n^n$$

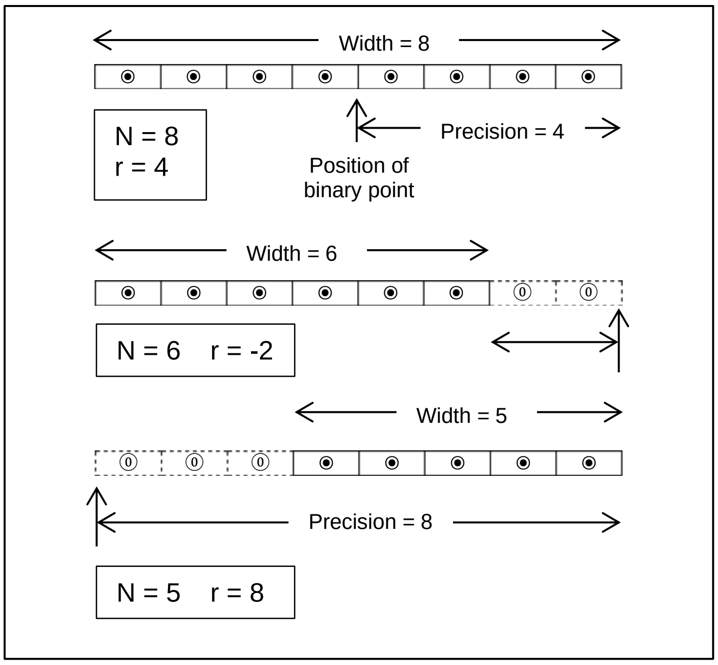
For example, for N = 8, r = 4:

$$m(p) = (b_0 + 2b_1 + 4b_2 + 8b_3 + 16b_4 + 32b_5 + 64b_6 + 128b_7) / 16$$

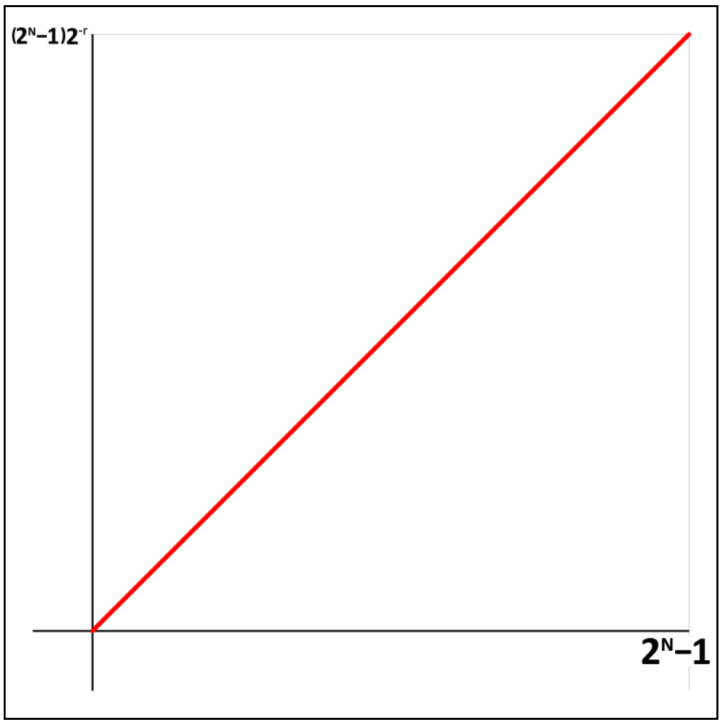
3.3.2.3 Illustrations



Example mapping



Example Positions of Binary Point



Value mapping from unsigned integer coding values

3.3.3 Fixed-Point 2’s Complement Coding

3.3.3.1 Description

The 2’s complement form of fixed-point coding is qualified by two parameters, the width of the word and the precision, to fully define the coding.

Fixed-point 2’s complement coding is supported for word widths from 2 to 64 and precisions from -99 to +99.

3.3.3.2 Mathematical Details

Where m is the fixed-point 2’s complement mapping, N is the word width and r is the precision:

$$N \in \mathbb{N}[2, 64]$$

$$r \in \mathbb{Z}[-99, 99]$$

$$m: \mathbb{B}^N \to \mathbb{P}[-2^{N-1-r}, 2^{N-1-r})$$

m is a function, its domain is $\mathbb{B}^N$, its range is a proper subset of $\mathbb{P}[-2^{N-1-r}, 2^{N-1-r})$

$$m^{-1}: \mathbb{R}[-2^{N-1-r}, 2^{N-1-r}) \to \mathbb{B}^N$$

m^{-1} is a function, its domain is $\mathbb{R}[-2^{N-1-r}, 2^{N-1-r})$, its range is $\mathbb{B}^N$

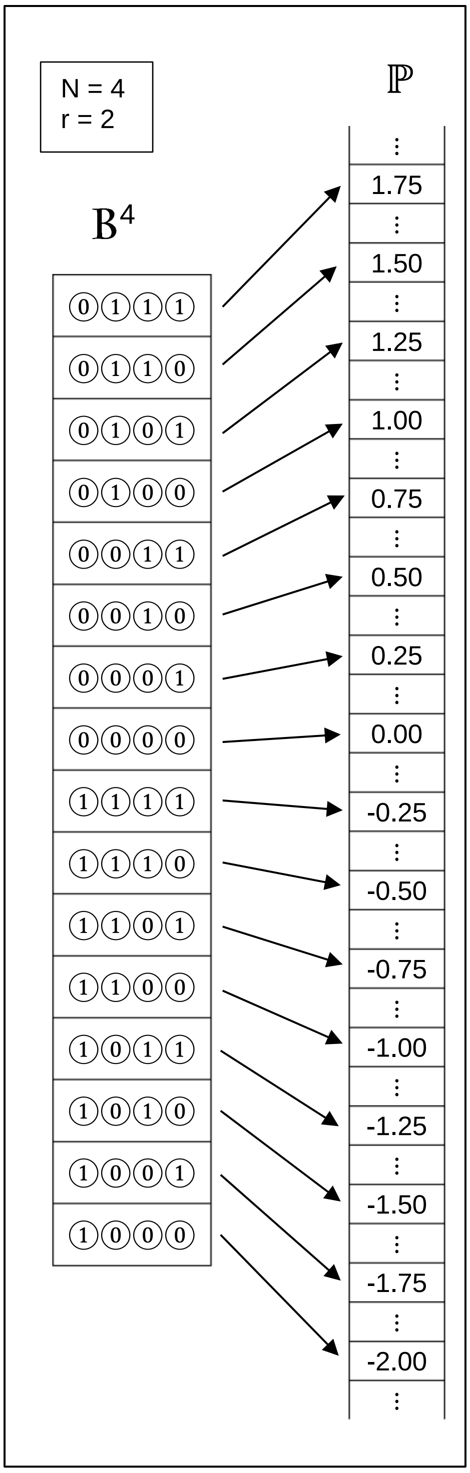
Where $p \in \mathbb{B}^N$ and b_n are the bit values of the bits within p:

$$m(p) = 2^{-r}\left(-2^{N-1}b_{N-1} + \sum_{n=0}^{N-2} b_n 2^n\right)$$

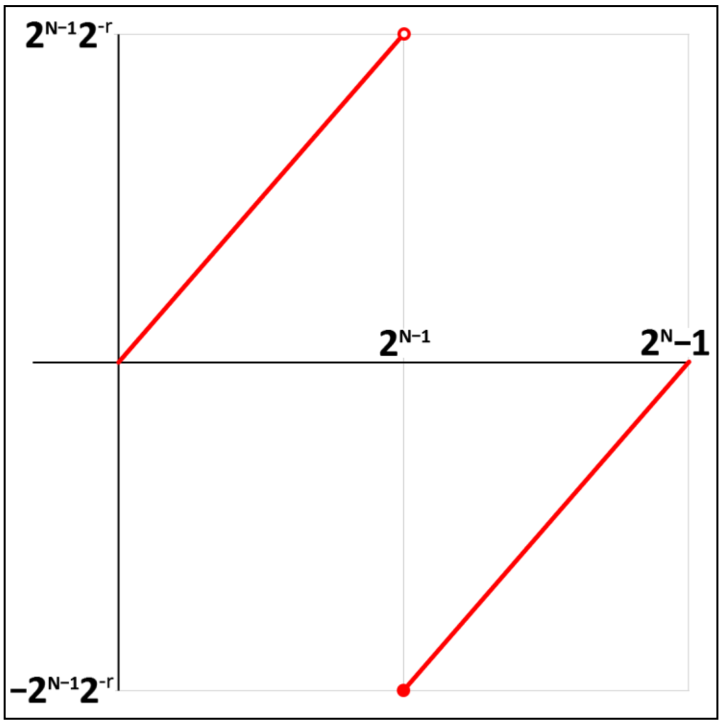
For example, for N = 8, r = 4:

$$m(p) = ((b_0 + 2b_1 + 4b_2 + 8b_3 + 16b_4 + 32b_5 + 64b_6) - 128\, b_7) / 16$$

3.3.3.3 Illustrations



Example mapping



Value mapping from unsigned integer coding values

3.4 Floating-Point Coding

3.4.1 Introduction

As we have seen, fixed-point codings can be used to represent non-integer values, but they have a limitation that makes them unsuitable for many programming situations. This limitation being that the accuracy of a stored value is determined by the precision of the coding and is absolute, i.e., fixed for all values that can be represented.

For example, an unsigned eight-bit fixed-point coding with a precision of three has a range of 0 to 31.875 with an accuracy of 0.125, which means that if the result of a calculation has an exact value of 10π ≈ 31.415926… the nearest this coding can store is 31.375 which is an error of about 0.13%, but if a calculation has an exact value of π/8 ≈ 0.3926990… the nearest the same coding can store is 0.375 which is an error of about 4.5%.

What would be more useful is a coding whose accuracy is not an absolute value fixed for all represented values but is a factor of the value being represented, e.g., a percentage, or a number of significant digits. So, how do we come up with a coding that can do this?

Think about the normalized exponential form of textual representations of a number whose value cannot be represented exactly, say 100π. It consists of a mantissa and exponent, the number of digits specified in the mantissa determines the accuracy of the representation, so

3.14×10^2 specifies the value with an accuracy of three significant digits

3.14159×10^2 specifies the value with an accuracy of six significant digits

We could define a coding which stored each digit of the mantissa and each digit of the exponent, the number of digits stored for the mantissa and for the exponent would be part of the definition of the coding. The number of digits stored for the mantissa would determine the accuracy of the represented value, and the number of digits stored for the exponent would determine the range of values that could be represented.

Of course, this example is shown in decimal, a coding based upon this principle which uses binary is more useful for our purposes. This is what the floating-point codings do, they effectively define a number of bits used to store a binary mantissa which determines the accuracy, a number of bits used to store a binary exponential value which determines the range of representable values and another bit to indicate the sign of the represented value. Unfortunately, this approach gives rise to some complications so things aren’t as simple as we would like. All this will now be described in more detail.

3.4.2 IEEE 754

The IEEE, which stands for The Institute of Electrical and Electronic Engineers, is an organization which, among other things, is responsible for the administration of many technical standards. One of these standards goes by the name of IEEE 754; this standard includes the definitions of the majority of the floating-point codings used by virtually every computer/digital device which needs to perform floating-point computations. IEEE 754 defines more floating-point codings than those supported by IO, and also uses multiple names for each coding it defines, but this document will simply use the names 64-bit, 32-bit and 16-bit. IO also supports the 80-bit floating-point format (extended precision) used by Intel floating-point hardware; this format is not defined by IEEE 754 although the standard does identify the concept of extended formats.

The floating-point codings are used to represent exact and approximate values of real numbers. If a desired real value cannot be exactly represented by a floating-point coding, then the bit pattern used will represent an approximated value. As will be seen, the exact non-zero values which may be represented by the floating-point codings are all sums of powers (positive and/or negative) of 2, so a floating-point coding for a word of width N will map $\mathbb{B}^N$ to a proper subset of $\mathbb{P}$.

3.4.3 Common Structure

The three IEEE 754 floating-point codings supported by IO have a common structure. The differences between the codings are the widths of two of the three constituent fields, which in turn determine the width of the word containing the bit pattern.

The msb of the bit pattern forms the “Sign Field”, the least significant bits of the bit pattern form the “Significand Fraction Field”, and the intervening bits form the “Exponent Field”. The significand fraction field is often called the “Trailing Significand Field”, in particular by the IEEE 754 specification. The widths of the Exponent and Significand Fraction fields are defined by the individual codings.

So, where N is the word width, NE is the width of the Exponent field and NF is the width of the Significand Fraction field, and b_n represents bit n of the bit pattern:

the word is partitioned into three fields, the Sign field consists of: b_{N-1}, the Exponent field consists of: b_{N-2}, ··· , b_{NF}, and the Significand Fraction field consists of: b_{NF-1}, ··· , b_0, as shown below

b_{N-1}	b_{N-2}	···	b_{NF}	b_{NF-1}	···	b_0
Sign	Exponent (NE bits)			Significand Fraction (NF bits)		

3.4.4 Floating-Point Classes

Each of the 2^N possible bit patterns that may be stored in a word belongs to one of five “classes”. The class to which a particular bit pattern belongs is determined by the first row, from the top, in the following table which matches the contents of the Exponent and Significand Fraction fields of the bit pattern. Note that the class of a bit pattern is independent of the contents of the sign field.

Exponent	Significand Fraction	Class
①①···①①	⓪⓪···⓪⓪	Infinity
①①···①①	◉◉···◉◉	NaN
⓪⓪···⓪⓪	⓪⓪···⓪⓪	Zero
⓪⓪···⓪⓪	◉◉···◉◉	Denormal
◉◉···◉◉	◉◉···◉◉	Normal

The denormal class is also commonly called “subnormal”.

The class of a bit pattern determines the mapping used to identify the number (if any) represented by the bit pattern.

So, to determine the value represented by a particular bit pattern the Exponent and Significand Fraction fields are examined to determine to which class the bit pattern belongs, and then the mapping for that class is applied to the bit pattern to identify the value represented.

The following notation is used to refer to the proper subsets of $\mathbb{B}^N$ which are of a particular class. Each element of $\mathbb{B}^N$ belongs to one, and only one, of the following sets.

$\mathbb{I}_N$ denotes the set of bit patterns of class Infinity; this set always has a cardinality of 2

$\mathbb{A}_N$ denotes the set of bit patterns of class NaN

$\mathbb{Z}_N$ denotes the set of bit patterns of class Zero; this set always has a cardinality of 2

$\mathbb{D}_N$ denotes the set of bit patterns of class Denormal

$\mathbb{N}_N$ denotes the set of bit patterns of class Normal

3.4.4.1 Normal Class

The vast majority of the possible bit patterns for any of the floating-point codings belong to this class; more precisely, all bit patterns whose exponent field contains a bit pattern which is neither all ⓪s nor all ①s are "normal".

The mapping from a number to the normal bit pattern used to represent it is illustrated by the following example:

Say we want to represent the value 12.125 using a floating-point coding:

The first thing to do is express the value in binary rather than decimal,

$12.125_{10} = 1100.001_2$

Now we express this in a normalized exponential format (see the section on textual representations if you don't know what this means)

$1100.001 = 1.100001x2^3$

To obtain the required bit pattern:

Firstly, we take the fractional part of the significand and set the bits of the significand fraction field to reflect these bit values, with the msb of the field representing the most significant fraction bit, and fill any bits to the right of the least significant bit set to a ① with ⓪s.

So, the significand fraction field will contain

①⓪⓪⓪⓪⓪①⓪⓪...

Then we take the exponent value, i.e., 3, and set the exponent field to the bit pattern that represents the exponent value using integer excess coding. This bit pattern will be different for different widths of the exponent field

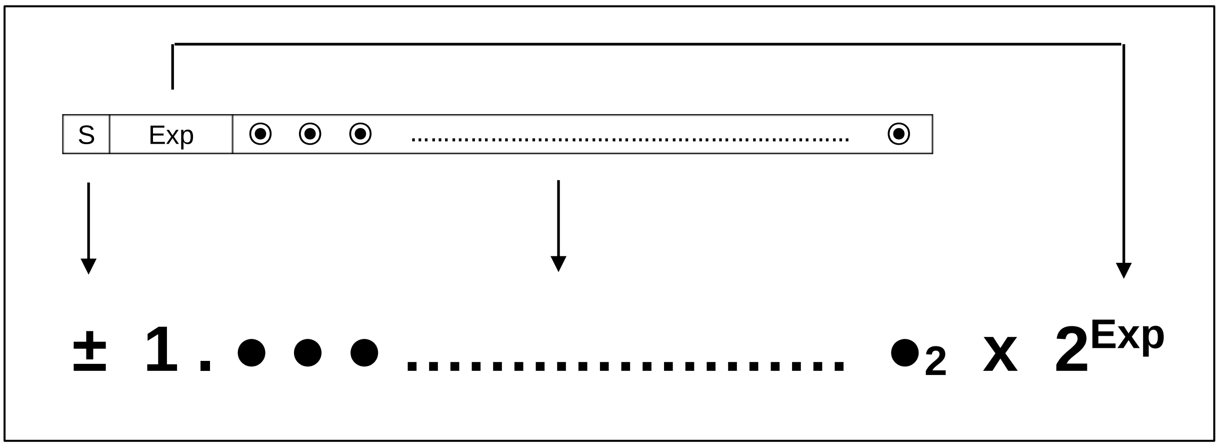
e.g.
if NE = 5 then the exponent field will contain ①⓪⓪①⓪,
if NE = 11 then the exponent field will contain ①⓪⓪⓪⓪⓪⓪⓪⓪①⓪

Finally, we set the sign filed/bit to ⓪ as the value is positive, we would set it to ① if the value was negative.

Note the following:

> As the integer part of the significand is always 1 for a non-zero value, we do not need to waste a bit of the bit pattern by storing it, we just need to store the fractional bits of the significand. However, this means that a value of zero cannot be represented using the normal class. The integer 1 bit value is not stored for the IEEE 754 codings, but is for the Intel 80-bit coding which will be described later.
>
> The normal class cannot be used to represent a value whose exponent value is outside of the range than can be represented in the floating-point coding's exponent field using integer excess coding, or whose exponent value would require the exponent field to contain either all ⓪s or all ①s.

The mapping from a bit pattern encoded using an IEEE 754 floating-point format whose exponent field does not contain all ⓪s nor all ①s, i.e., is of class normal, to the number it represents is illustrated below. Exp represents the value encoded in the exponent field using excess coding.



Say the bit pattern contains a sign field/bit set to ⓪, an exponent field containing the bit pattern representing the value +4 using integer excess coding, and a significand fraction field containing the bit pattern ①①⓪⓪⓪①⓪⓪…

Firstly, we construct a significand value with an integer part of 1 and a fractional part containing the bit values of the significand fraction field,

> 1.110001_2

Then we multiply this value by two to the power of the value contained in the exponent field,

> $1.110001 \mathrm{x} 2^4 = 11100.01$

If the sign field/bit was set to ① then we would negate this value, but it isn't so we don't,

We now have the represented value,

> $11100.01_2 = 28.25_{10}$

Mathematical Details

Where m is a supported IEEE floating-point mapping and N is the word width:

$$m: \mathrm{N}_{\mathrm{N}} \to \mathbb{P}$$

m is a function, its domain is N_{N}, its range is a proper subset of $\mathbb{P}$

Where $\mathrm{p} \in \mathrm{N}_{\mathrm{N}}$ and b_n are the bit values of the bits within p:

and

$$\mathrm{s} = 1 - 2\mathrm{b}_{\mathrm{N}-1}$$

and

$$\mathrm{e} = -(2^{\mathrm{NE}-1} - 1) + \sum_{\mathrm{n}=0}^{\mathrm{NE}-1} \mathrm{b}_{(\mathrm{NF}+\mathrm{n})} 2^{\mathrm{n}}$$

$$\mathrm{m(p)} = \mathrm{s}2^{\mathrm{e}} \left(1 + 2^{-\mathrm{NF}} \sum_{\mathrm{n}=0}^{\mathrm{NF}-1} \mathrm{b}_{\mathrm{n}} 2^{\mathrm{n}} \right)$$

3.4.4.2 Denormal Class

You may be wondering why the normal class does not include bit patterns whose exponent field contains all ⓪s. The problem would be that because of the implied value of 1 for the integer bit of the significand, it would be impossible to represent a value of zero. So, the zero class of bit patterns is defined (see later) to allow the representation of a zero value. Bit patterns of the zero class have an exponent field containing all ⓪s, and the significand fraction field also containing all ⓪s. This leaves a set of bit patterns available to represent values other than zero and those that can be represented by normal bit patterns. These bit patterns constitute the denormal class, and are used to represent non-zero values that are smaller than any that can be represented by a normal bit pattern.

Bit patterns whose exponent field contains all ⓪s and whose significand fraction field does not contain all ⓪s are “denormal”.

If it is required to represent a number which would require an exponent value which is lower than that available for the normal class, it may be possible to represent it using the denormal class.

The mapping from a number to the denormal bit pattern used to represent it is similar to that for the normal class, with the following differences:

> The integer bit, which like the normal class is not stored in the bit pattern, is a ⓪, not a ①.
>
> The exponent value is not the value represented by the bit pattern in the exponent field (which would be that for all ⓪s), but is the value that would be represented if the exponent field contained the bit pattern ⓪…⓪①, i.e., the lowest exponent value that could be stored for the normal class.

For example, say we are using a floating-point coding which has an exponent field width of five (NE = 5), and we want to represent the number $1.01_2 \times 2^{-22}$.

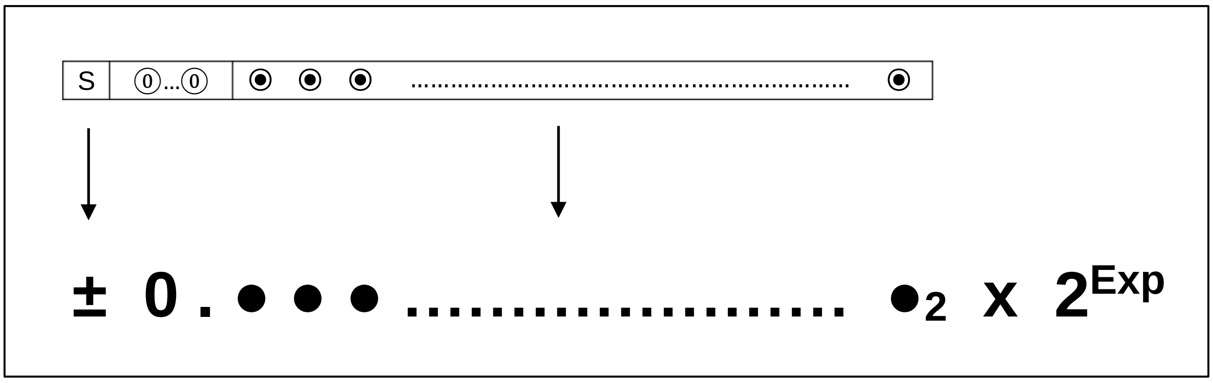
The lowest exponent value that can be represented in the exponent field for the normal class is that represented by ⓪⓪⓪⓪①, which is -14, so we cannot use the normal class to represent the required number.

What we do is change our textual representation from the normalized format shown above to a denormalized format with the exponent value being -14,

$$1.01_2 \times 2^{-22} = 0.0000000101_2 \times 2^{-14}$$

So, the required bit pattern has the sign field/bit set to ⓪ (for a positive value), the exponent field set to ⓪⓪⓪⓪⓪ (to indicate the denormal class), and the significant fraction field set to ⓪⓪⓪⓪⓪⓪⓪①⓪①. If the significand fraction field for the coding being used is wider than 10 bits then it will be padded with ⓪'s on the right-hand end as for the normal class. If the significand fraction field for the coding being used is narrower than 10 bits then the value cannot be represented exactly using the coding.

The mapping from a denormal bit pattern to the represented number is illustrated below. Exp represents the lowest (negative) value that can be encoded in the exponent field using excess coding for the normal class, i.e., that represented by a bit pattern of ⓪⓪......⓪① in the exponent field.



Say we are using a floating-point coding which has an exponent field width of eight (NE = 8), and the bit pattern is ① ⓪⓪⓪⓪⓪⓪⓪⓪ ⓪⓪⓪①①⓪⓪…

The exponent field contains all ⓪'s, so we have a denormal bit pattern, the sign field/bit contains ① so the value represented is negative. The exponent value to be used is that represented by the 8-bit bit pattern ⓪⓪⓪⓪⓪⓪⓪①, which is -126.

So, we construct the required mantissa using the un-stored integer bit value of 0 and the stored significant fraction bits to get

$$0.00011_2 \times 2^{-126}$$

Incorporating the sign and adjusting to a normalized textual representation gives us a represented value of:

$$-1.1_2 \times 2^{-130}$$

Mathematical Details

Where m is a supported IEEE floating-point mapping and N is the word width:

$m: \mathrm{D}_N \rightarrow \mathbb{P}$

m is a function, its domain is D_N, its range is a proper subset of $\mathbb{P}$

Where $p \in \mathrm{D}_N$ and b_n are the bit values of the bits within p:

and

$$s = 1 - 2b_{N-1}$$

and

$$e = -(2^{NE-1} - 2)$$

$$m(p) = s2^{e-NF} \sum_{n=0}^{NF-1} b_n 2^n$$

3.4.4.3 Zero Class

There are two elements of B^N which are of class zero, ⓪⓪⓪···⓪⓪ and ①⓪⓪···⓪⓪.

⓪⓪⓪···⓪⓪ is used to represent a value of exact-zero, and is also used to represent a positive number that is less than the smallest number which can be represented by the coding (positive underflow). This bit pattern is usually denoted simply by 0. In the case of positive underflow, it can be referred to as positive (or plus) zero and is sometimes denoted by +0.

①⓪⓪···⓪⓪ may be used to represent a negative number whose magnitude is less than the smallest number which can be represented by the coding (negative underflow). This bit pattern is referred to as negative (or minus) zero and is denoted by -0.

Bit patterns of class zero can also be thought of as being an extension of the denormal class as the methods used to convert between denormal bit patterns and values will give correct results if applied to bit patterns of class zero.

Mathematical Details

Where m is a supported IEEE floating-point mapping and N is the word width:

$m: \mathbb{Z}_N \rightarrow \{0\}$

Where $p \in \mathbb{Z}_N$:

$m(p) = 0$

3.4.4.4 Infinity Class

There are two elements of B^N which are of class infinity, ⓪①⋯①⓪⋯⓪ and ①①⋯①⓪⋯⓪, where all the bits in the Exponent field are set to ① and all the bits in the Significand Fraction field are set to ⓪. Neither of these bit patterns represent any particular number.

⓪①⋯①⓪⋯⓪ is used when it is desired to represent a specific positive number, but that number cannot be represented because it is greater than the largest number which can be represented by the coding. This condition is called “positive overflow”. This bit pattern is referred to as positive (or plus) infinity.

①①⋯①⓪⋯⓪ is used when it is desired to represent a specific negative number, but that number cannot be represented because it is greater in magnitude than the largest negative number which can be represented by the coding. This condition is called “negative overflow”. This bit pattern is referred to as negative (or minus) infinity.

There is no IEEE floating-point mapping defined for these bit patterns as there is no element of any set of numbers which is specifically represented by these bit patterns.

3.4.4.5 NaN Class

NaN is short for “Not a Number”, these bit patterns do not represent any number.

There are three types of NaN. Which type of NaN a particular bit pattern is, is determined by the first row, from the top, in the following table which matches the contents of the Sign and Significand Fraction fields.

Sign	Significand Fraction	Type
①	①⓪⋯⓪⓪	QNaN FP Indefinite
⦿	①⦿⋯⦿⦿	QNaN
⦿	⓪⦿⋯⦿⦿	SNaN

The QNaN Floating-point Indefinite bit pattern is generated in response to a requested operation for which there is no defined result, e.g., a NaN is specified as an operand, the square root or logarithm of a negative number, etc.

QNaN stands for “Quiet NaN”, if such a bit pattern is encountered during processing, then the instruction stream continues execution without interruption.

SNaN stands for “Signaling NaN”, if such a bit pattern is encountered during processing, then a processor exception is signaled and will be handled by an asynchronous handler if such handling is enabled.

IO will not generate a Signaling NaN or Quiet NaN bit patterns, apart from the QNaN FP Indefinite bit pattern, as the result of any floating-point operation; however, it is possible for the X register to contain such a bit pattern as the result of a conversion from another coding when the “Preserve Bit Pattern” Calculator Operation option is selected and is applicable for the conversion.

There is no IEEE floating-point mapping defined for these bit patterns as there is no element of any set of numbers which is represented by these bit patterns.

3.4.5 Summary

The following table summarizes the above information for a general IEEE 754 floating-point coding:

Sign Field	Exponent Field	Implied Integer Bit	Significand Fraction Field	Class / NaN Type	(Pseudo) Sign	Number of bit patterns
①	①①⋯①①	N/A	①①⋯①① : ①⓪⋯⓪①	QNaN	None	$2^{(NF-1)} - 1$
①	①①⋯①①	N/A	①⓪⋯⓪⓪	QNaN (FPI)		1
①	①①⋯①①	N/A	⓪①⋯①① : ⓪⓪⋯⓪①	SNaN		$2^{(NF-1)} - 1$
①	①①⋯①①	N/A	⓪⓪⋯⓪⓪	Infinity	Negative	1
①	①①⋯①⓪ : ⓪⓪⋯⓪①	①	①①⋯①① : ⓪⓪⋯⓪⓪	Normal		$(2^{NE} – 2)2^{NF}$
①	⓪⓪⋯⓪⓪	⓪	①①⋯①① : ⓪⓪⋯⓪①	Denormal		$2^{NF} - 1$
①	⓪⓪⋯⓪⓪	⓪	⓪⓪⋯⓪⓪	Zero		1
⓪	⓪⓪⋯⓪⓪	⓪	⓪⓪⋯⓪⓪	Zero	Positive	1
⓪	⓪⓪⋯⓪⓪	⓪	⓪⓪⋯⓪① : ①①⋯①①	Denormal		$2^{NF} - 1$
⓪	⓪⓪⋯⓪① : ①①⋯①⓪	①	⓪⓪⋯⓪⓪ : ①①⋯①①	Normal		$(2^{NE} – 2)2^{NF}$
⓪	①①⋯①①	N/A	⓪⓪⋯⓪⓪	Infinity		1
⓪	①①⋯①①	N/A	⓪⓪⋯⓪① : ⓪①⋯①①	SNaN	None	$2^{(NF-1)} - 1$

⓪	①①⋯①①	N/A	①⓪⋯⓪⓪ : ①①⋯①①	QNaN		$2^{(NF-1)}$

3.4.6 Floating-Point IEEE 16-Bit Coding

3.4.6.1 Description

This coding is also referred to as “binary16” or “half precision”.

This coding is rarely used in application software because of its very limited range and accuracy. However, because it adheres to the general structure of the IEEE 754 codings described above, it is very useful for providing examples to demonstrate these principles which are much easier to follow than the “larger” codings due to the smaller bit patterns and numbers required. It will be used frequently in this document for this reason.

3.4.6.2 Summary

Number of Bits; word (sign:exp:sigdf)	16 (1:5:10)	
Minimum Denormal Value	$2^{-24} \approx 5.960\text{x}10^{-8}$	
Minimum Normal Value	$2^{-14} \approx 6.104\text{x}10^{-5}$	
Epsilon Value	$2^{-10} \approx 9.766\text{x}10^{-4}$	
Maximum Consecutive Integer Value	$2^{11} = 2{,}048$	
Maximum Normal Value	$(2^{16} - 2^{5}) = 65{,}504$	
Exponent		
Excess	15	
Value for Denormal	-14	
Minimum Value for Normal	-14	
Maximum Value for Normal	15	
Numbers of Class Bit Patterns		
All	65,536	100.000%
Zero	2	~0.003%
Denormal	2,046	~3.122%
Normal	61,440	~93.750%
Infinity	2	~0.003%
QNaN	1,024	~1.563%
SNaN	1,022	~1.559%

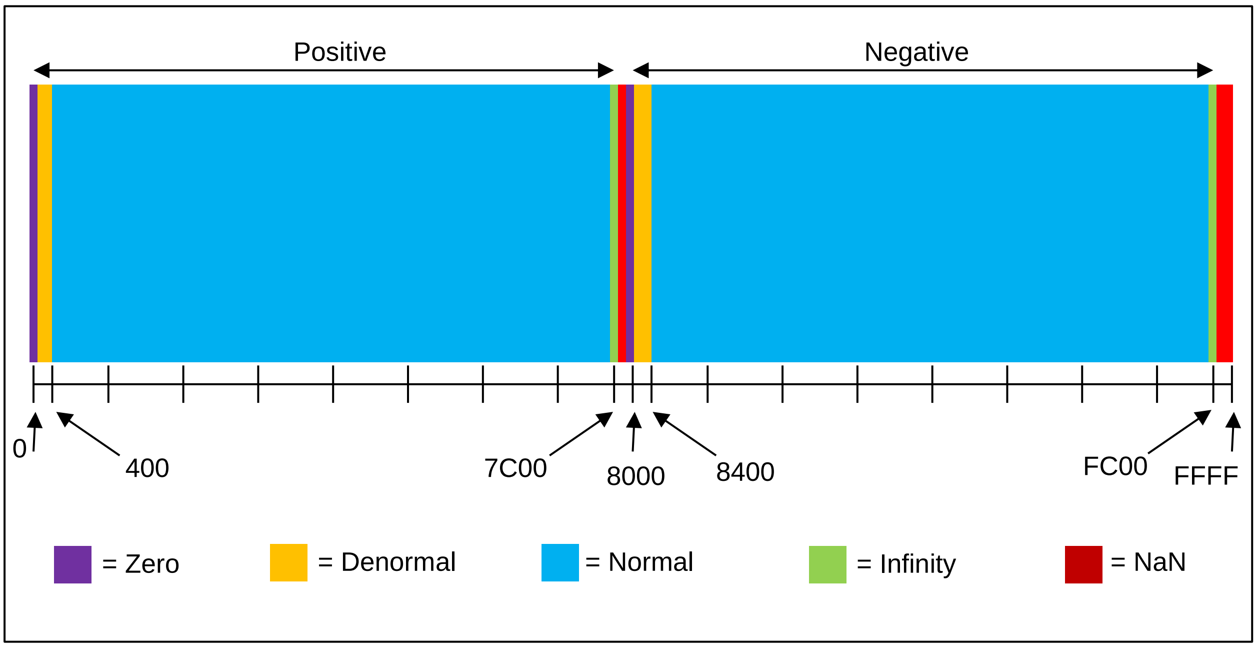
3.4.6.3 Details

The 16-bit word used to contain the bit pattern is partitioned as follows:

b_{15}	b_{14}	⋯	b_{10}	b_{9}	⋯	b_{0}

Sign	Exponent (5 bits)	Significand Fraction (10 bits)

3.4.6.4 Illustrations



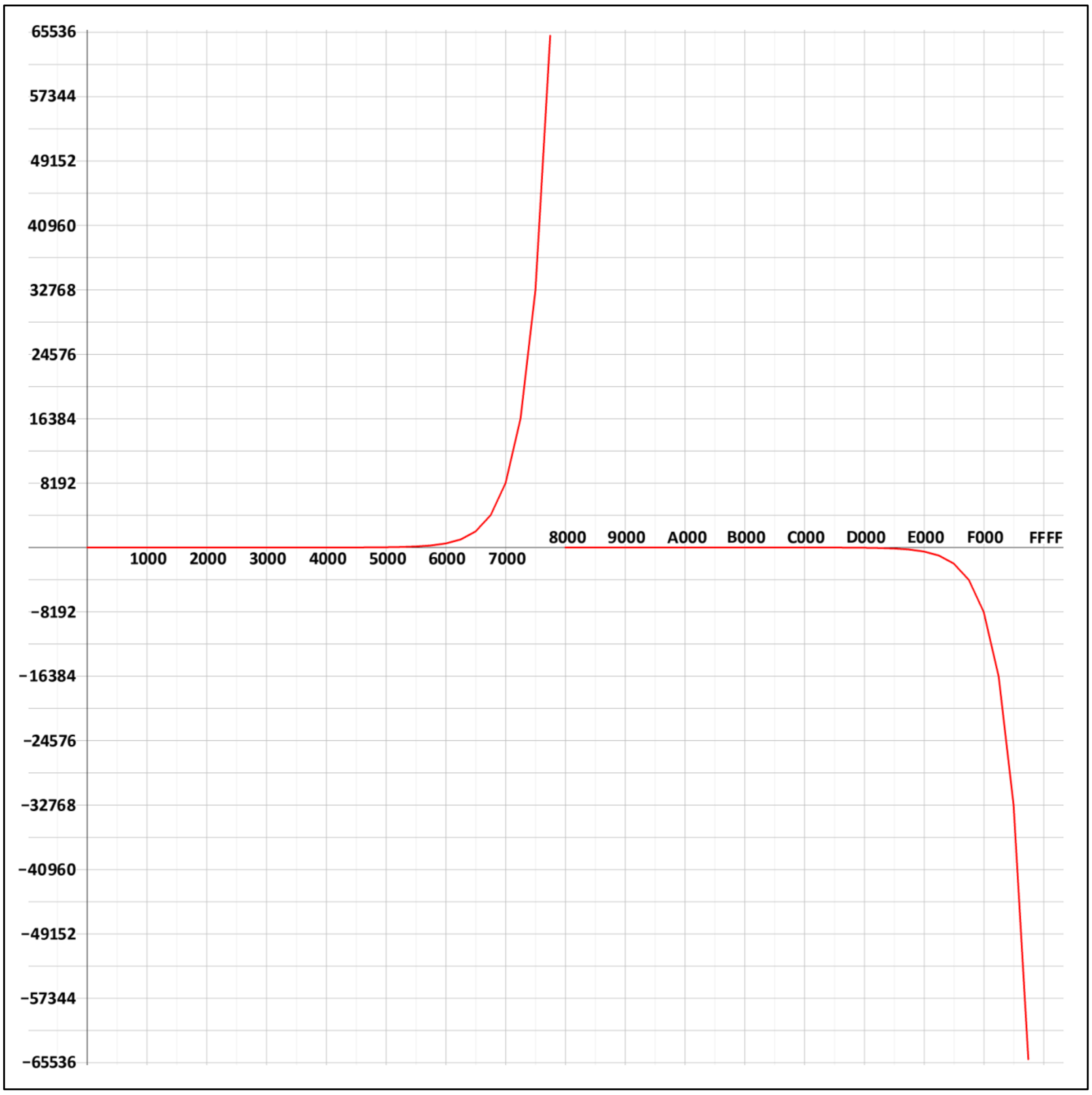
Class Distribution for Floating-point IEEE 16-Bit Coding

The diagram above shows the distribution of the different classes for the possible 16-bit bit patterns. The horizontal scale represents the bit patterns, ordered by their values when interpreted using 16-bit unsigned integer coding, annotated in hexadecimal.

The width of the areas denoting the Zero and Infinity classes are to a different scale to the other classes because they would be too thin to be clearly visible otherwise.

Corresponding diagrams for the other floating-point codings were not included as the areas representing the non-normal classes for those codings would be too thin to be clearly visible.

The next graph shows the values represented by the 16-bit bit patterns. The X axis is labeled as in the class distribution diagram.



16-bit Floating-point Value mapping from unsigned integer coding values

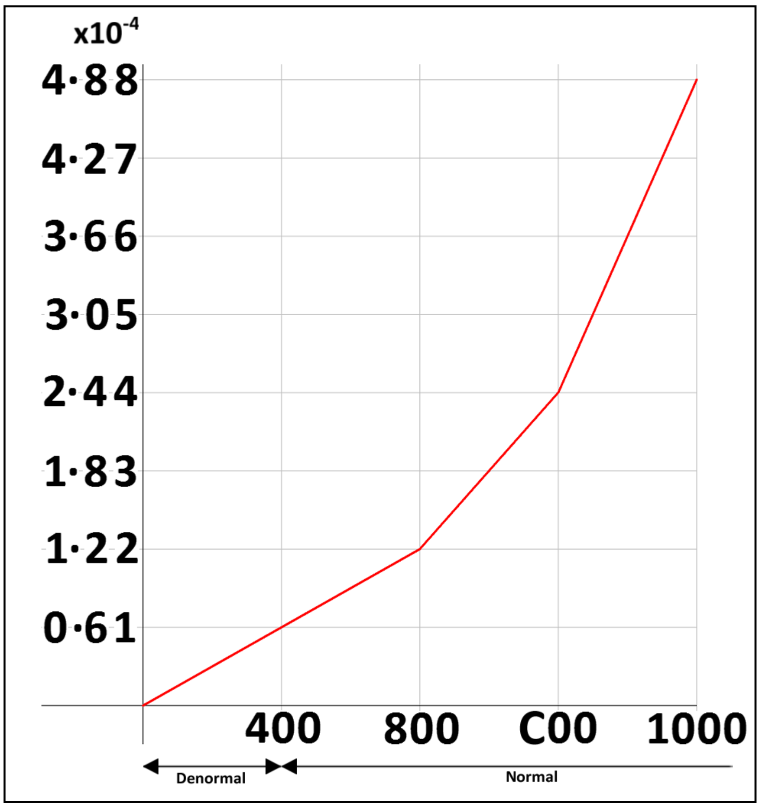
Although this graph shows the entire range of representable values, due to the linear scale of the Y axis most of the values appear indistinguishable from zero.

The graph may look like a pair of curves, but in reality, it consists of sequences of straight lines with gradually increasing magnitudes of gradient.

Corresponding graphs for the other floating-point codings are not included as they would have a very similar shape, but the axes would be labeled with much larger values.

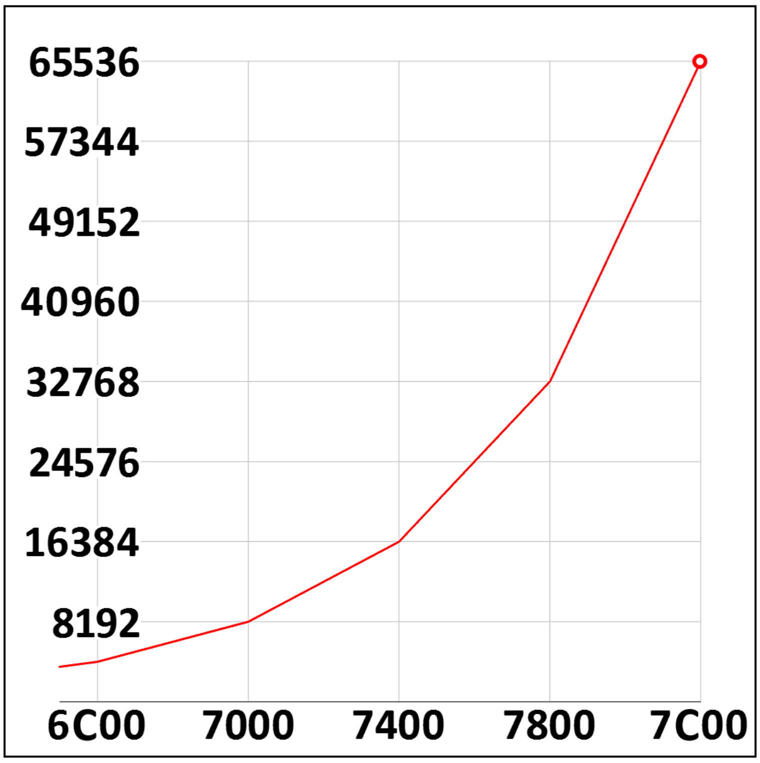
The following graphs show a zoomed-in view of two areas of the above graph to show more detail of the values represented.

Again, corresponding graphs for the other floating-point codings are not included as they would have a very similar shape, but the axes would be labeled with much smaller/larger values.



This graph shows the values represented by the initial 16-bit bit patterns. Notice that as the bit patterns transition from the denormal bit patterns to normal the gradient of the graph remains the same because the exponent value used to determine the value remains the same and the implied integer bit changes from 0 to 1, so the weight of the lsb of the significand remains the same.

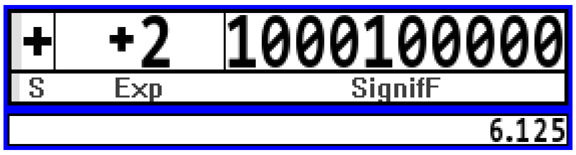
Subsequent transitions from one exponent value to the next double the gradient of the graph because the weight of the lsb of the significand doubles as a consequence.



This graph shows the values represented by the 16-bit bit patterns for the largest positive normal bit patterns. The doubling of the gradient for each increase of the exponent value continues, and by now the gradient is considerably steeper, as can be seen by comparing the scales used for the Y axis compared to the preceding graph.

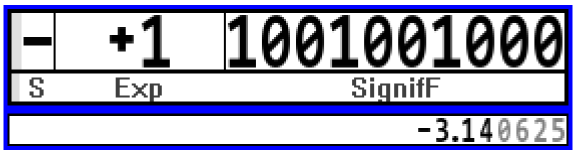
Note that the bit pattern for 7C00 is that for positive overflow (the positive infinity class), so does not represent a specific numerical value, hence the “hole” to indicate that a value of 65536 is not specifically represented.

Here are some examples as displayed by IO:



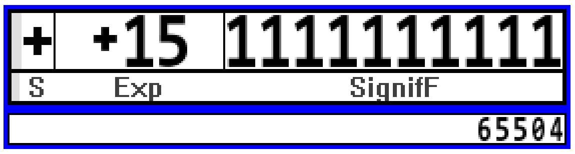
+ +2 1000100000

6.125



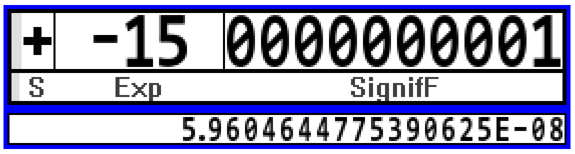
- +1 1001001000

-3.140625



+ +15 1111111111

65504



+ -15 0000000001

5.9604644775390625E-08

3.4.7 Floating-Point IEEE 32-Bit Coding

3.4.7.1 Description

This coding is not as commonly used as 64-bit coding, as most floating operations are performed by floating-point hardware these days, it is not any faster to use a 32-bit coding than a 64-bit coding so you might as well get the extra accuracy provided by 64-bit coding as it does not incur a performance overhead.

The most common use of 32-bit coding is in situations where a very large number of values need to be stored at the same time.

Good examples of this are programming interfaces for 3D graphics rendering. Typically, internal models of 3D objects contain definitions of small triangles in 3D space which approximate the surface of the object being modelled. To provide a good model these definitions can consist of thousands of triangles, each triangle is defined by the 3D coordinates of each of its three vertices, so each vertex requires three coordinates. Each coordinate usually needs to be stored as a floating-point value as an integer value would not be flexible enough to allow such processing as arbitrary scaling, etc. The accuracy provided by 32-bit coding is usually quite sufficient to generate good results, so it is commonly used in preference to 64-bit coding as the storage required to store the thousands of coordinates is halved in comparison.

3.4.7.2 Summary

Number of Bits; word (sign:exp:sigdf)	32 (1:8:23)	
Minimum Denormal Value	$2^{-149} \approx 1.4013\text{x}10^{-45}$	
Minimum Normal Value	$2^{-126} \approx 1.17549\text{x}10^{-38}$	
Epsilon Value	$2^{-23} \approx 1.19209\text{x}10^{-7}$	
Maximum Consecutive Integer Value	$2^{24} = 16,777,216$	
Maximum Normal Value	$(2^{128} - 2^{104}) \approx 3.40282\text{x}10^{38}$	
Exponent		
Excess	127	
Value for Denormal	-126	
Minimum Value for Normal	-126	
Maximum Value for Normal	127	
Numbers of Class Bit Patterns		
All	4,294,967,296	100.000%

Zero	2	~0.000%
Denormal	16,777,214	~0.391%
Normal	4,261,412,864	~99.219%
Infinity	2	~0.000%
QNaN	8,388,608	~0.195%
SNaN	8,388,606	~0.195%

3.4.7.3 Details

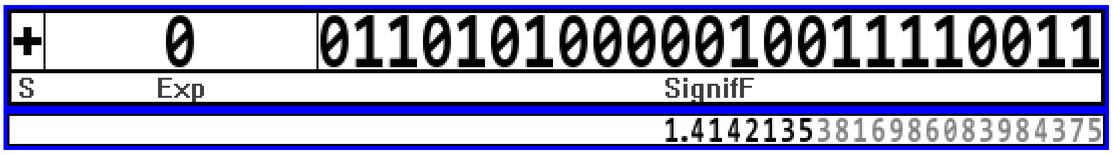
This coding is also referred to as “binary32” or “single precision”.

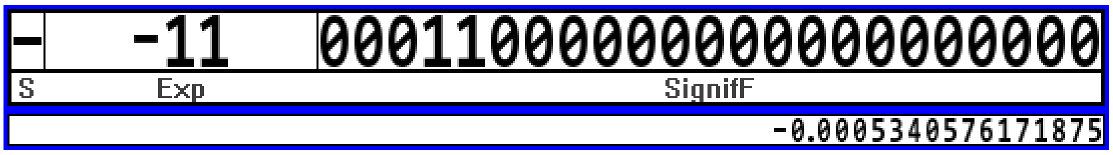
The 32-bit word used to contain the bit pattern is partitioned as follows:

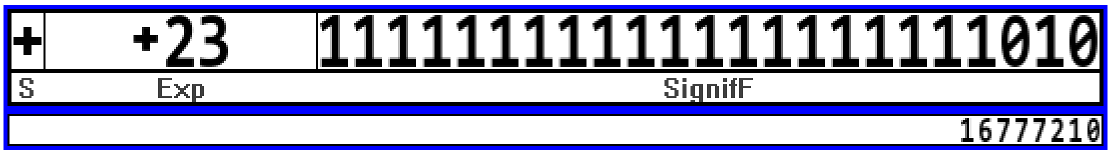
b_{31}	b_{30}	…	b_{23}	b_{22}	…	b_{0}
Sign	Exponent (8 bits)			Significand Fraction (23 bits)		

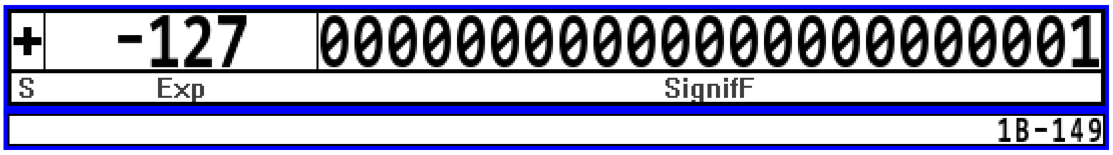
3.4.7.4 Illustrations

Here are some examples as displayed by IO:


```
+  0  01101010000010011110011
S  Exp  SignifF
1.41421353816986083984375
```


```
-  -11  00011000000000000000000
S  Exp  SignifF
-0.0005340576171875
```


```
+  +23  11111111111111111111010
S  Exp  SignifF
16777210
```


```
+  -127  00000000000000000000001
S  Exp  SignifF
1B-149
```

3.4.8 Floating-Point IEEE 64-Bit Coding

3.4.8.1 Description

This coding is the most commonly used of the floating-point codings; it is the default coding used for floating-point variables by many high-level computer languages.

3.4.8.2 Summary

Number of Bits; word (sign:exp:sigdf)	64 (1:11:52)	
Minimum Denormal Value	$2^{-1074} \approx 4.94065645841247 \times 10^{-324}$	
Minimum Normal Value	$2^{-1022} \approx 2.2250738585072 \times 10^{-308}$	
Epsilon Value	$2^{-52} \approx 2.22044604925031 \times 10^{-16}$	
Maximum Consecutive Integer Value	2^{53} = 9,007,199,254,740,992	
Maximum Normal Value	$(2^{1024} - 2^{971}) \approx 1.79769313486232 \times 10^{308}$	
Exponent		
Excess	1023	
Value for Denormal	-1022	
Minimum Value for Normal	-1022	
Maximum Value for Normal	1023	
Numbers of Class Bit Patterns		
All	18,446,744,073,709,551,616	100.000%
Zero	2	~0.000%
Denormal	9,007,199,254,740,990	~0.049%
Normal	18,428,729,675,200,069,632	~99.902%
Infinity	2	~0.000%
QNaN	4,503,599,627,370,496	~0.024%
SNaN	4,503,599,627,370,494	~0.024%

3.4.8.3 Details

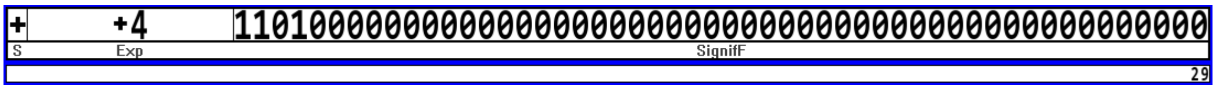
This coding is also referred to as "binary64" or "double precision".

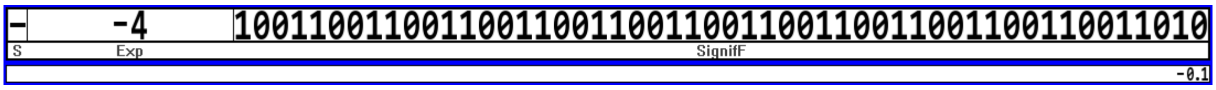
The 64-bit word used to contain the bit pattern is partitioned as follows:

b_{63}	b_{62}	…	b_{52}	b_{51}	…	b_0
Sign	Exponent (11 bits)			Significand Fraction (52 bits)		

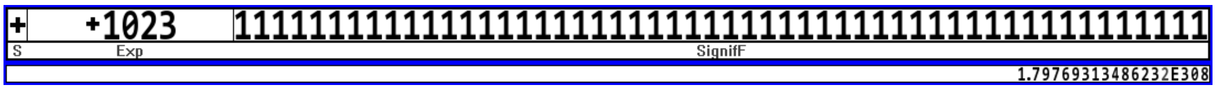
3.4.8.4 Illustrations

Here are some examples as displayed by IO:

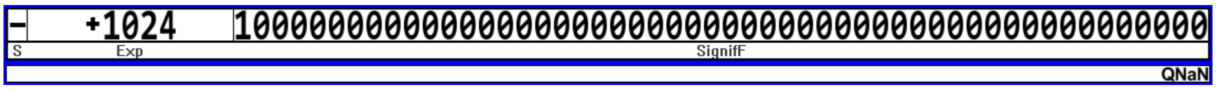

+ +4 1101000000000000000000000000000000000000000000000000
S Exp SignifF
29



```
-  -4  1001100110011001100110011001100110011001100110011010
S  Exp  SignifF
-0.1
```



```
+  +1023  1111111111111111111111111111111111111111111111111111
S  Exp  SignifF
1.79769313486232E308
```



```
-  +1024  1000000000000000000000000000000000000000000000000000
S  Exp  SignifF
QNaN
```

3.4.9 Floating-Point Intel 80-Bit Coding

3.4.9.1 Description

This coding is not defined by IEEE 754 but was developed by Intel and implemented in its 8087 floating-point coprocessors in 1980, this predated IEEE 754. This coding has been supported by subsequent iterations of Intel floating-point hardware since then, and by any floating-point hardware produced by other vendors which claims to be compatible. It has been a very long time since many mainstream computers you could buy which run a version of the Microsoft Windows operating system did not have hardware support for this coding.

This coding is based upon the general structure previously described for the supported IEEE 754 codings, but with a very important difference. Whereas the setting of the integer bit of the significand was implied by the contents of the exponent field for denormal and normal bit patterns, in this coding the integer bit is stored along with the fractional bits.

One of the consequences of this difference is that it is possible to construct bit patterns where the setting of the integer bit is not that implied by the contents of the exponent field. This gives rise to an additional set of classes of bit pattern; these are referred to as pseudo classes. Unfortunately, this results in about half of the available 2^{80} bit patterns being effectively wasted; but as there are over 65,000 times as many bit patterns available compared to 64-bit coding this loss was apparently considered to be acceptable.

3.4.9.2 Summary

Number of Bits; word (sign:exp:sigd)	80 (1:15:64)	
Minimum Denormal Value	$2^{-16445} \approx 3.6451995318824746\text{x}10^{-4951}$	
Minimum Normal Value	$2^{-16382} \approx 3.3621031431120935\text{x}10^{-4932}$	
Epsilon Value	$2^{-63} \approx 1.08420217248550443\text{x}10^{-19}$	
Maximum Consecutive Integer Value	2^{64} = 18,446,744,073,709,551,616	
Maximum Normal Value	$(2^{16384} – 2^{16320}) \approx 1.18973149535723\text{x}10^{4932}$	
Exponent		
Excess	16383	
Value for Denormal	-16382	
Minimum Value for Normal	-16382	
Maximum Value for Normal	16383	
Numbers of Class Bit Patterns		
All	1,208,925,819,614,629,174,706,176	100.000%
Zero	2	~0.000%
Denormal	18,446,744,073,709,551,614	~0.002%

Normal	604,426,016,319,167,168,249,856	~49.997%
Infinity	2	~0.000%
QNaN	9,223,372,036,854,775,808	~0.001%
SNaN	9,223,372,036,854,775,806	~0.001%
Pseudo Denormal	18,446,744,073,709,551,616	~0.002%
Unnormal	604,426,016,319,167,168,249,856	~49.997%
Pseudo Infinity	2	~0.000%
Pseudo QNaN	9,223,372,036,854,775,808	~0.001%
Pseudo SNaN	9,223,372,036,854,775,806	~0.001%

3.4.9.3 Details

This coding is also referred to as “extended precision”.

The 80-bit word used to contain the bit pattern is partitioned as follows:

b_{79}	b_{78}	···	b_{64}	b_{63}	···	b_0
Sign	Exponent (15 bits)			Significand (64 bits)		

The bit patterns are classified as shown in the table for the IEEE 754 codings if the significand integer bit (b_{63}) is set to the implied integer bit setting shown in the table, if it is not then it is classified as one of the pseudo classes shown in the following table.

Sign Field	**Exponent Field**	**Significand Integer Bit (b_{63})**	**Significand Fraction Bits**	**Pseudo Class / NaN Type**
①	①①···①①	⓪	①①···①① : ①⓪···⓪⓪	Pseudo QNaN
①	①①···①①	⓪	⓪①···①① : ⓪⓪···⓪①	Pseudo SNaN
①	①①···①①	⓪	⓪⓪···⓪⓪	Pseudo Infinity
①	①①···①⓪ : ⓪⓪···⓪①	⓪	①①···①① : ⓪⓪···⓪⓪	Unnormal
①	⓪⓪···⓪⓪	①	①①···①① : ⓪⓪···⓪⓪	Pseudo Denormal

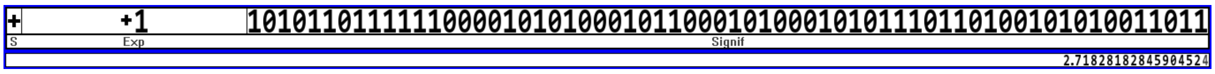
⓪	⓪⓪⋯⓪⓪	①	⓪⓪⋯⓪⓪ : ①①⋯①①	Pseudo Denormal
⓪	⓪⓪⋯⓪① : ①①⋯①⓪	⓪	⓪⓪⋯⓪⓪ : ①①⋯①①	Unnormal
⓪	①①⋯①①	⓪	⓪⓪⋯⓪⓪	Pseudo Infinity
⓪	①①⋯①①	⓪	⓪⓪⋯⓪① : ⓪①⋯①①	Pseudo SNaN
⓪	①①⋯①①	⓪	①⓪⋯⓪⓪ : ①①⋯①①	Pseudo QNaN

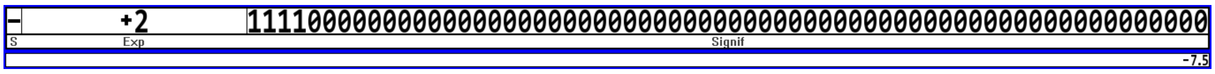
Note that the algorithm used to determine the numerical value for Normal and Denormal bit patterns, i.e., shift the binary point in the significand to a position determined by the contents of the exponent field, can also be applied to Unnormal and Pseudo Denormal bit patterns to obtain a numerical value. However, these mappings are not guaranteed to be supported by all implementations so should be avoided. If the X register contains such a bit pattern and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected then IO will not permit any floating-point operations to be performed on the X register contents.

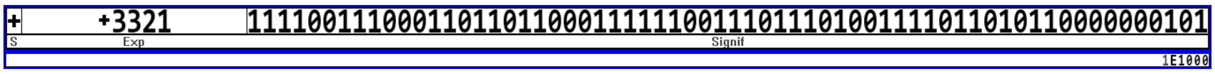
IO will not generate a pseudo class bit pattern as the result of any floating-point operation; however, it is possible for the X register to contain such a bit pattern as the result of a conversion from another coding when the “Preserve Bit Pattern” Calculator Operation option is selected and is applicable for the conversion.

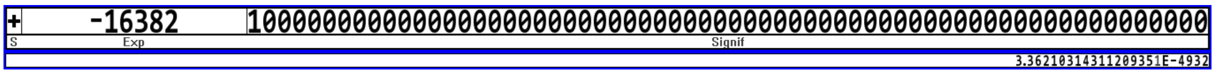
3.4.9.4 Illustrations

Here are some examples as displayed by IO:


```
+       +1       1010110111111000010101000101100010100010101110110100101010011011
S       Exp                                  Signif
                                                                2.71828182845904524
```


```
-       +2       1111000000000000000000000000000000000000000000000000000000000000
S       Exp                                  Signif
                                                                            -7.5
```


```
+       +3321    1111001110001101101100011111100111011101001111011010110000000101
S       Exp                                  Signif
                                                                          1E1000
```


```
+       -16382   1000000000000000000000000000000000000000000000000000000000000000
S       Exp                                  Signif
                                                       3.36210314311209351E-4932
```

3.5 Fixed/Floating-Point Rounding Types

The preceding descriptions of the fixed and floating-point codings identified the mappings from bit patterns to numbers. The inverse mappings are more complicated than for the integer codings because there is not a unique number identified by any particular bit pattern as is the case for integer codings.

The fixed and floating-point codings have an additional qualifier, the rounding type, which completes the definition of the inverse mapping.

3.5.1 Illustrations

The following diagrams illustrate the inverse mappings defined by each of the five supported rounding types.

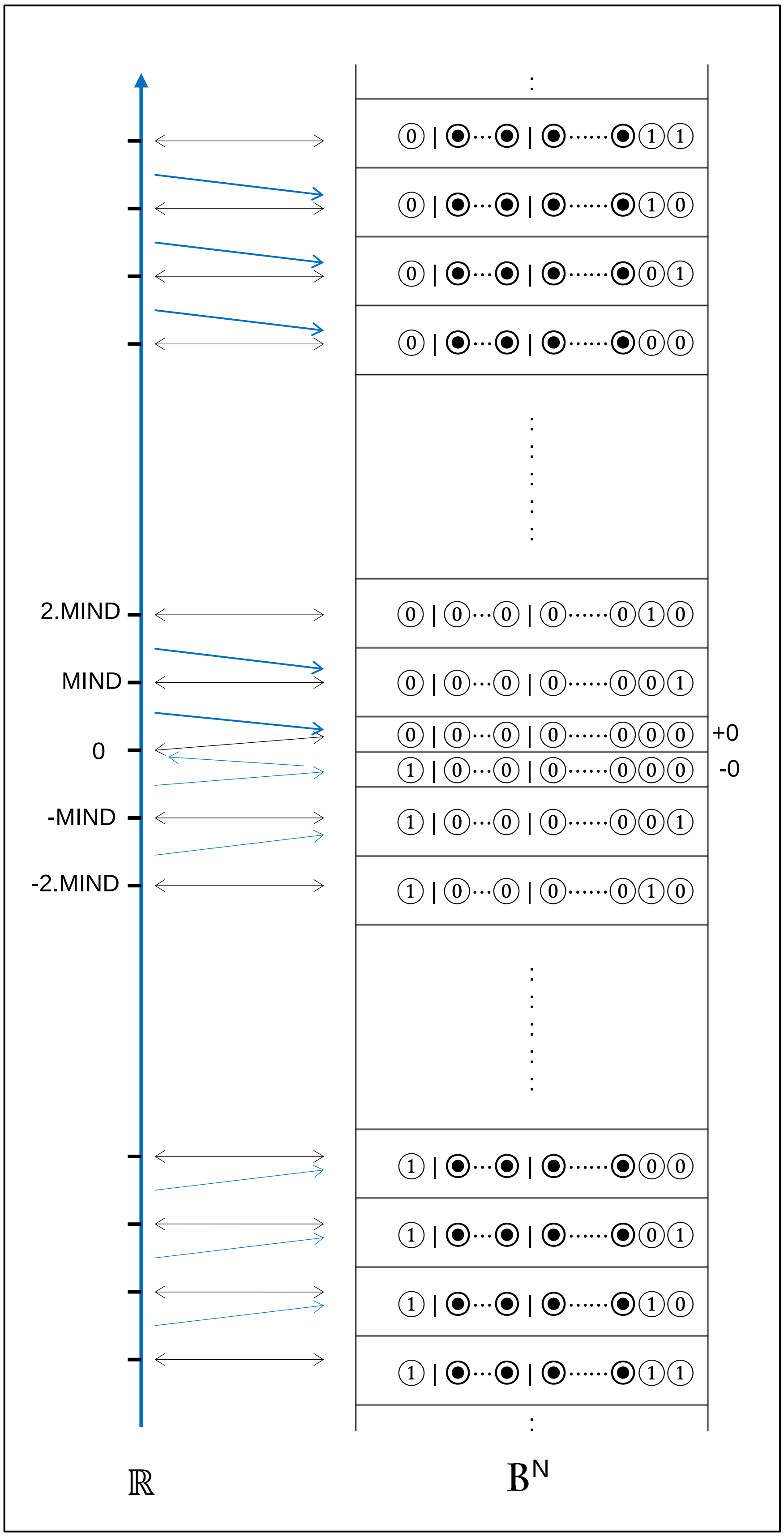
Remember that the result of a fixed/floating-point operation is a real number which may or may not be able to be represented exactly using the current coding. The rounding type is only applicable when this cannot be done, and defines the bit pattern chosen to represent the result.

The depicted bit patterns are for a general floating-point coding supported by IO (N = coding width). The vertical bars indicate the position of the boundaries between the three fields. The corresponding diagrams for fixed-point codings are not included; they would be very similar, but a little simpler.

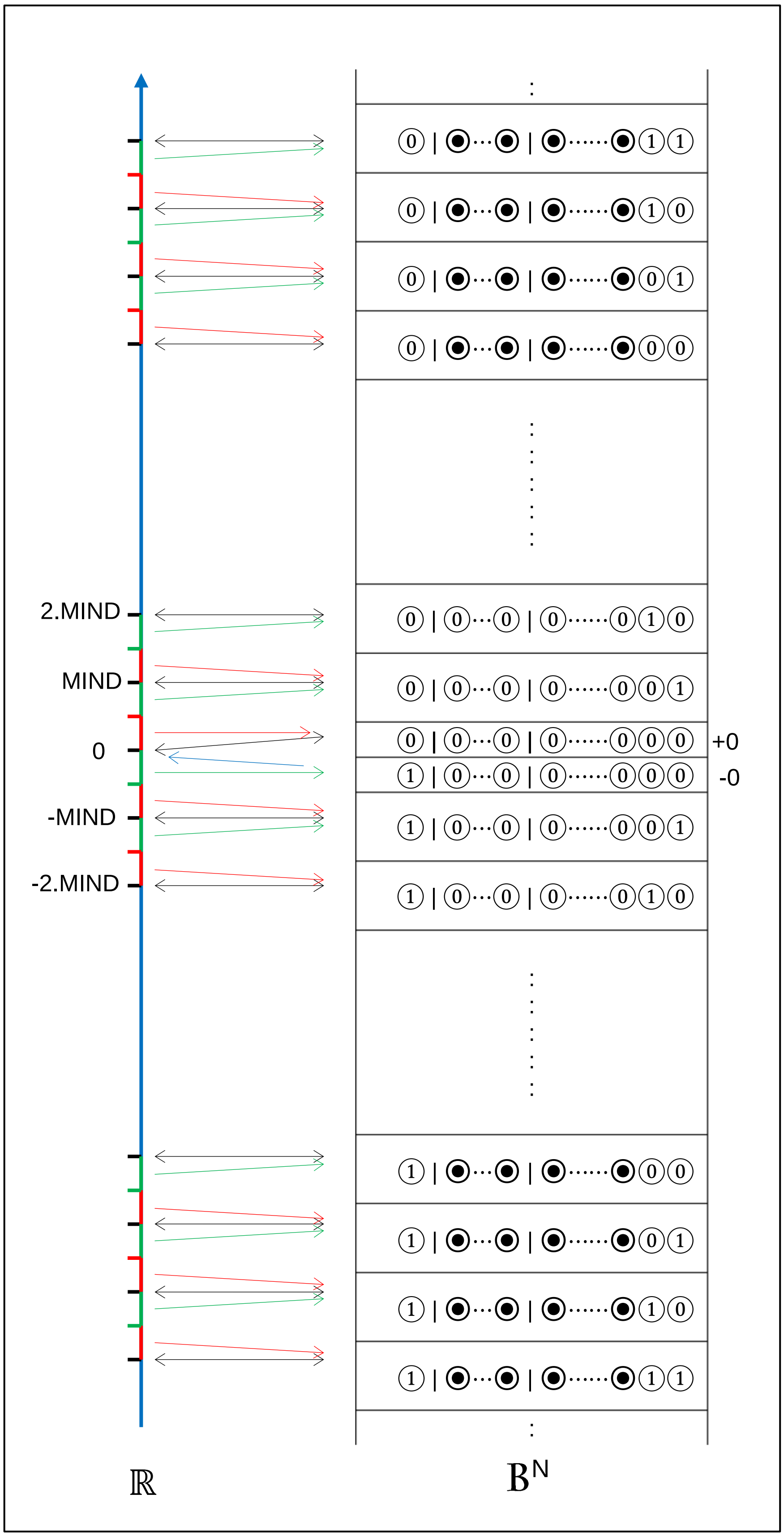
The black markers on the real number line and the black double arrows denote values and mappings for values that may be represented exactly. The other colored markers/sections of the real number line and single arrows denote values that cannot be represented exactly and the mapping to the bit pattern as defined by the rounding type. MIND denotes the smallest value that can be represented by a denormal bit pattern for the coding.

Note how things get more complicated around a value of zero due to the existence of the positive-zero and negative-zero bit patterns.

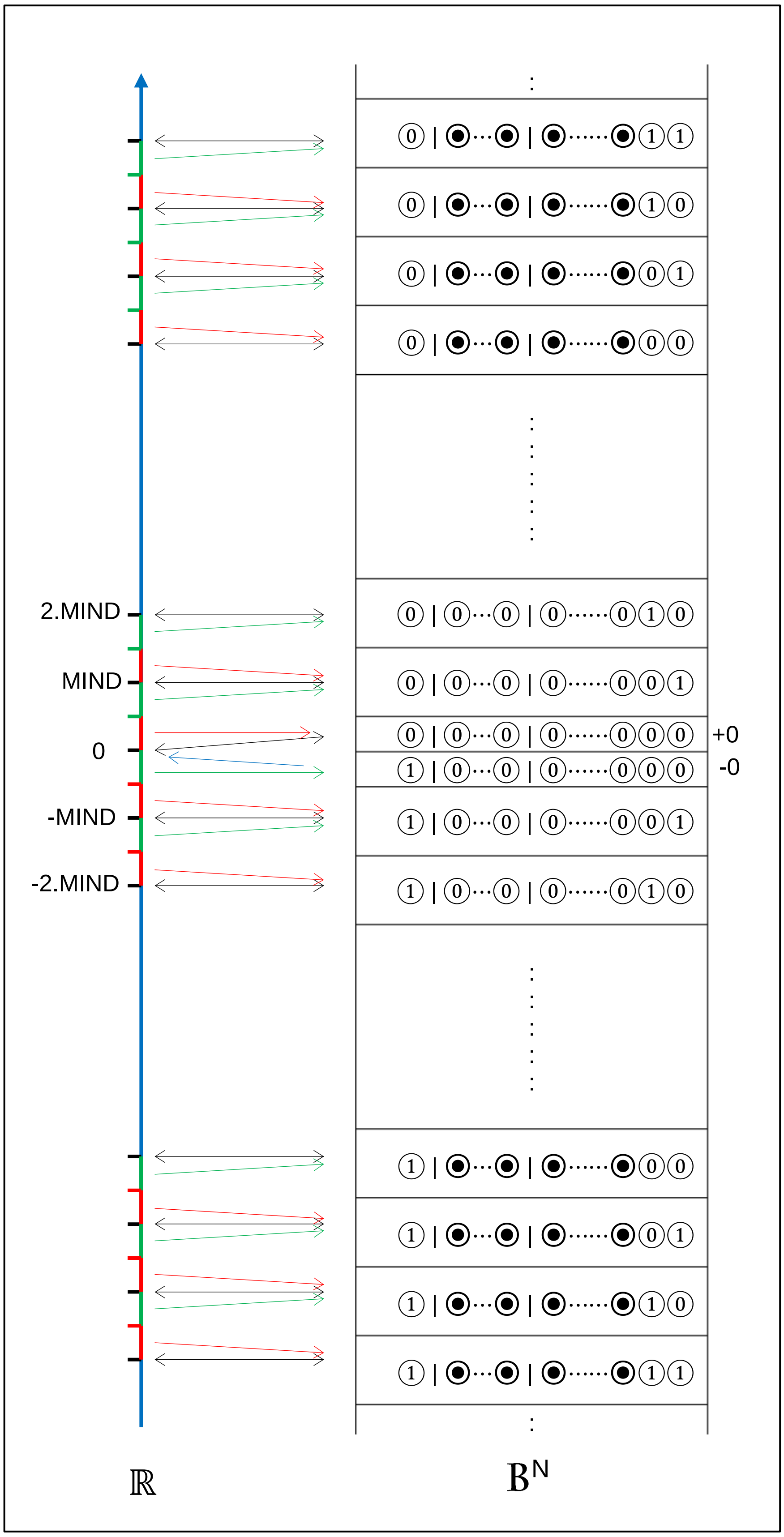
Also note that the only difference between the illustrations for the “To Nearest (Even)” and “To Nearest (Away From Zero)” rounding types is the color of one in four of the notches on the real number line. They only map to different bit patterns for one half of the values that lay exactly half way between exactly representable values.



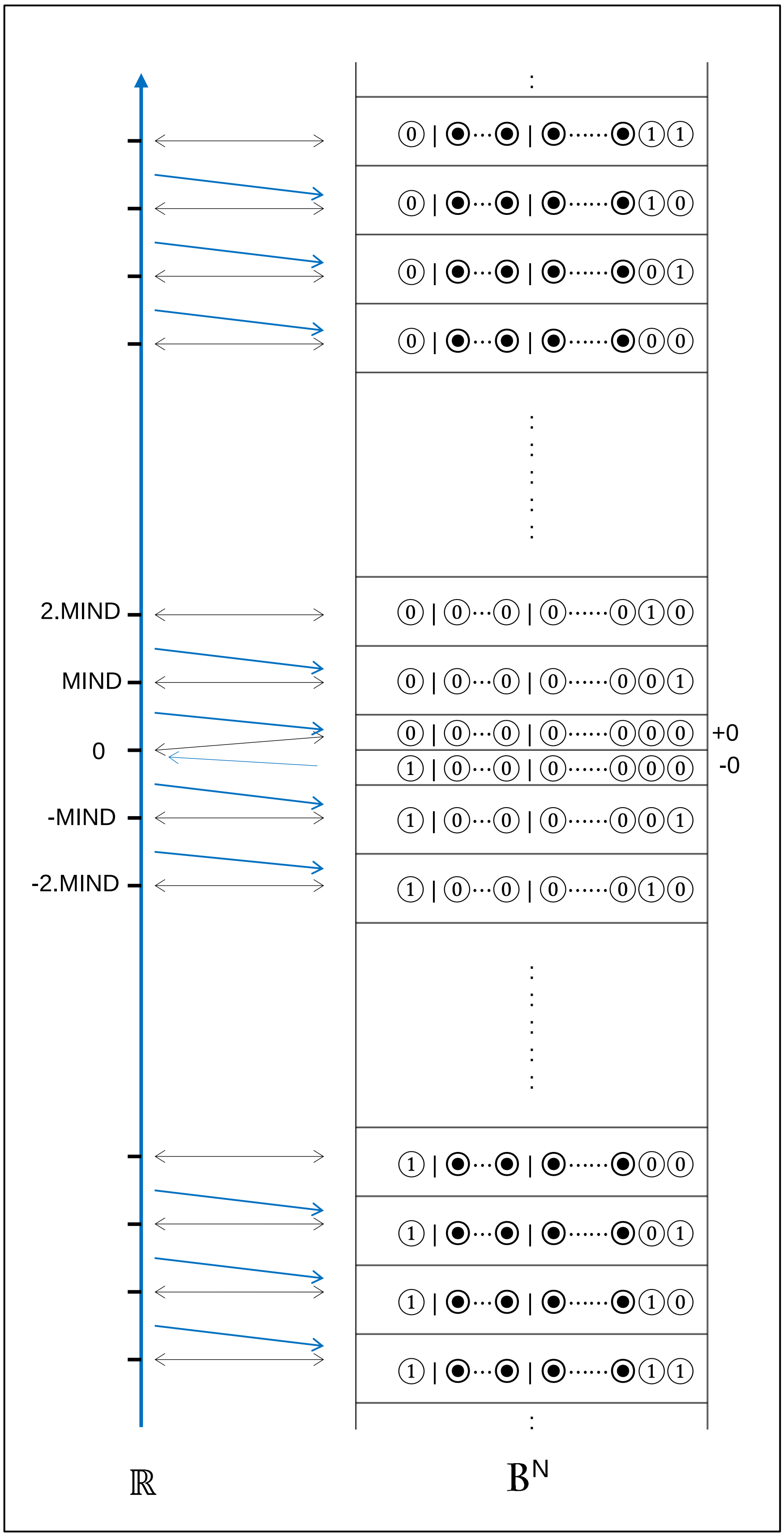
Floating-Point Codings Inverse Mapping for To Zero (Truncate) Rounding Type



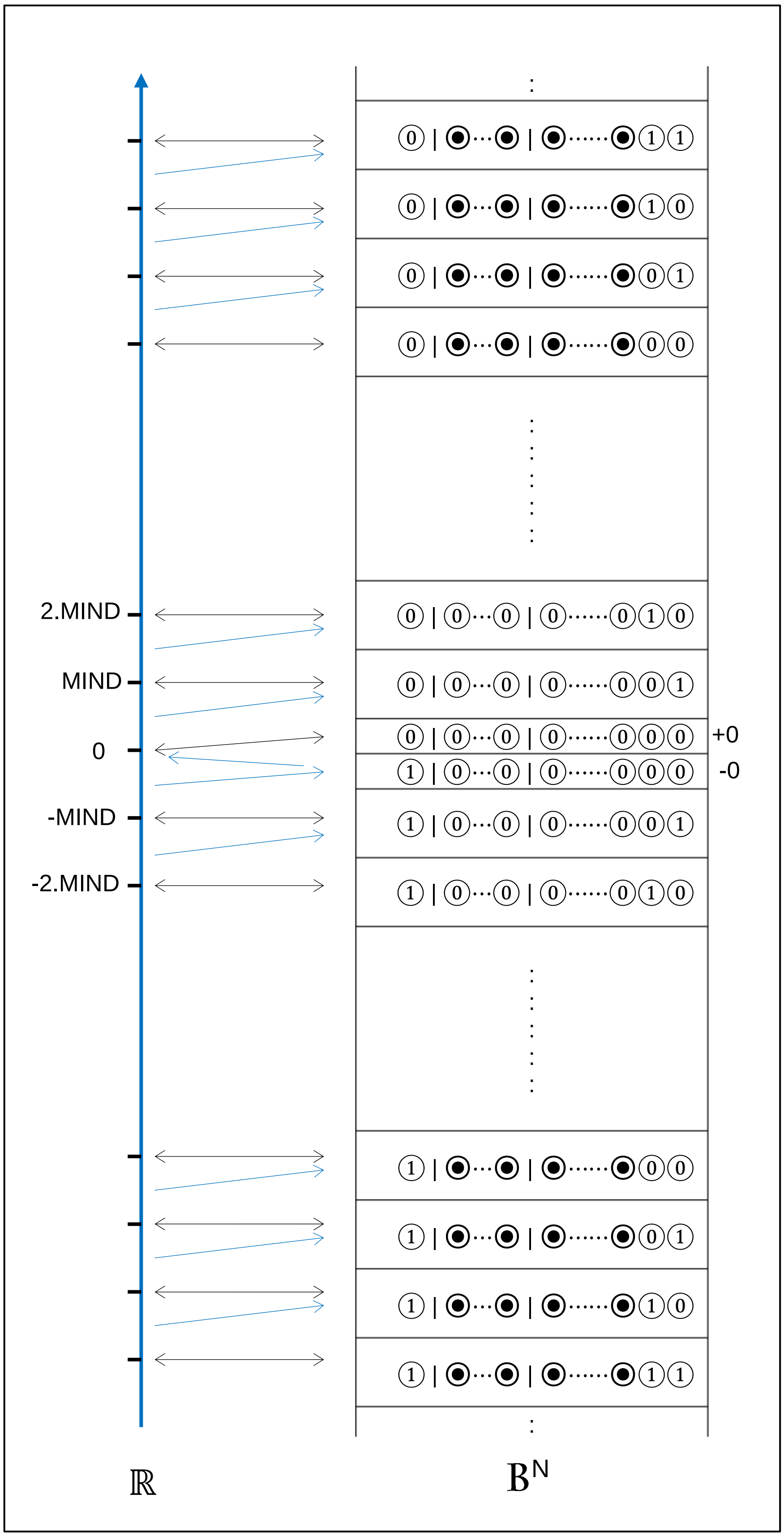
Floating-Point Codings Inverse Mapping for To Nearest (Even) Rounding Type



Floating-Point Codings Inverse Mapping for To Nearest (Away From Zero) Rounding Type



Floating-Point Codings Inverse Mapping for To Negative Infinity (Down) Rounding Type



Floating-Point Codings Inverse Mapping for To Positive Infinity (Up) Rounding Type

4 DISPLAY MODES

The user may select one of ten display modes supported by IO. Which display modes are available at any time is determined by the current number coding type, as shown in the following table:

Number Coding Type	**Display Mode Type**			
	Word Value	**Bit Pattern**	**Fields**	**Components**
Integer	Integer Word Value	Integer Word Bit Pattern Integer Fields Bit Pattern	Integer Fields Values	
Fixed-point	Fixed-point Word Value	Fixed-point Word Bit Pattern		
Floating-point	Floating-point Word Value	Floating-point Fields Bit Pattern	Floating-point Fields	Floating-point Components

The display mode is not part of the configuration so is not stored in configuration files. When IO starts up the Word Value display mode for the initial number coding type is selected.

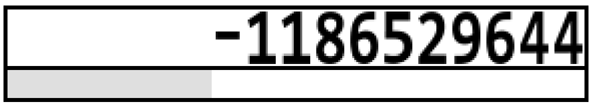
Display modes are selected by clicking the appropriate keypad key.

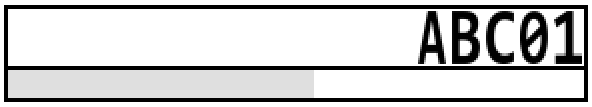
The current display mode is used for the main number display, both in the keypad and the detached display, and can also affect the number displays in the memories display window and the stack display window.

As well as the main value/bit pattern display, the display mode also affects the content of the markers display.

4.1 Integer Display Modes

4.1.1 Integer Word Value

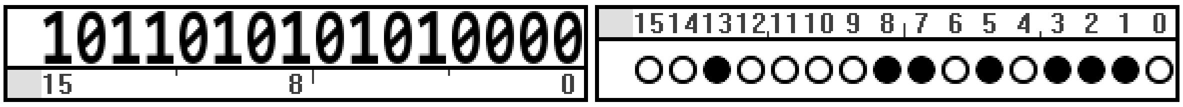

-1186529644


ABC01

The integer value for the bit pattern stored in the register, as defined by the current coding options, is displayed using the current number display options.

The markers area will indicate the area of the main value display which may be used to contain digits, which gives an idea of how close the current value is to the limits for the current coding. The width of the indicated area is affected by a number of things, including: the current coding, the current number base, whether or not a sign is displayed, whether or not digit group separators may be displayed, etc.

4.1.2 Integer Word Bit Pattern



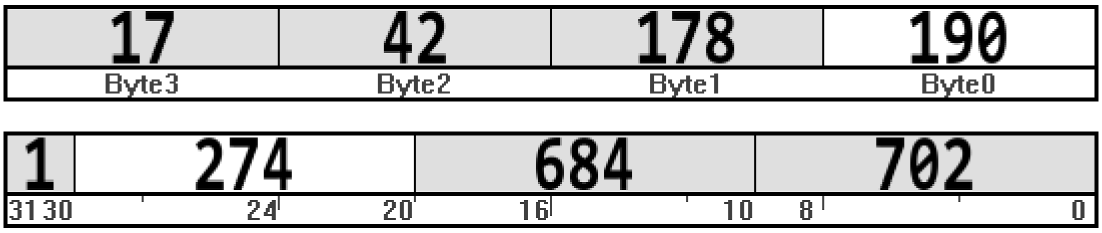
The setting of each bit of the bit pattern stored in the register is displayed using the digit set determined by the current digit set options.

The markers area is as for the Integer Word Value display mode but with additional content in the used area, as follows:

Each nibble boundary will be marked with a short notch
Each byte boundary will be marked with a medium height notch
Each 16-bit boundary will be marked with a long notch

If a medium or large digit set is being used then bit numbers will be displayed as determined by the Bit Numbering and Bit Number Display options in the Number display options

4.1.3 Integer Fields Values



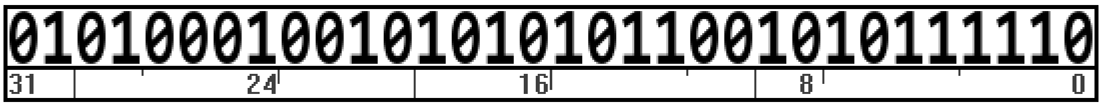
The bit pattern stored in the register is partitioned into fields as defined by the currently selected field layout, and the content of each field is displayed as determined by the settings for that field in the definition of the currently selected field layout. The location of the fields in the display will coincide with the location of the field within the word bit pattern. The currently selected field is identified by a different background color, as set on the Color options.

Depending of the current Number Display options, the markers area will contain either:

The labels for each field as set in the currently selected field layout definition

Notches and bit numbers for the entire word, as for the Integer Word Bit Pattern display mode, possibly with additional bit numbers displayed for field boundaries

4.1.4 Integer Fields Bit Pattern

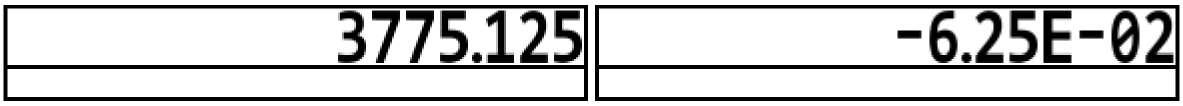


This is the same as the Integer Word Bit Pattern display mode but the markers area has additional notches, full height, to indicate the locations of field boundaries.

The Integer Fields Bit Pattern display mode is provided as a convenient way of quickly matching the bit patterns contained the fields with the values represented by those bit patterns. The contents of the X register cannot be modified in this display mode; hence the vast majority of IO's keys will be disabled.

4.2 Fixed-Point Display Modes

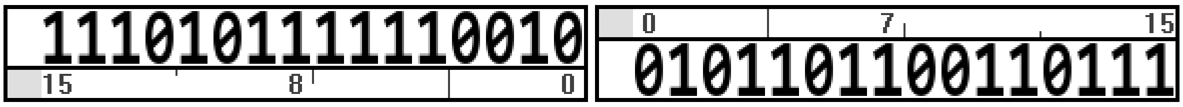
4.2.1 Fixed-Point Word Value



The value for the bit pattern stored in the register, as defined by the current coding options, is displayed using the current number display options.

The markers area will be empty.

4.2.2 Fixed-Point Word Bit Pattern

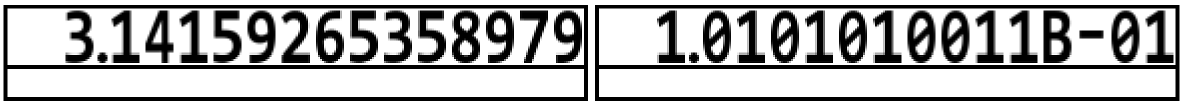


The setting of each bit of the bit pattern stored in the register is displayed in the same way as for the Integer Word Bit Pattern display mode.

The markers area will also be as for the Integer Bit Pattern display mode, but with the addition of a full height notch indicating the location of the binary point within the bit pattern, provided that it lies within the bit pattern.

4.3 Floating-Point Display Modes

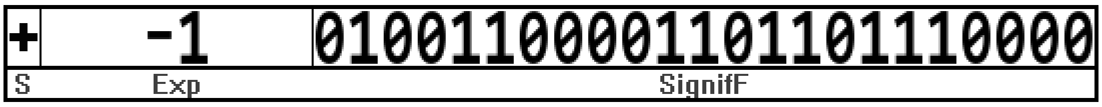
4.3.1 Floating-Point Word Value

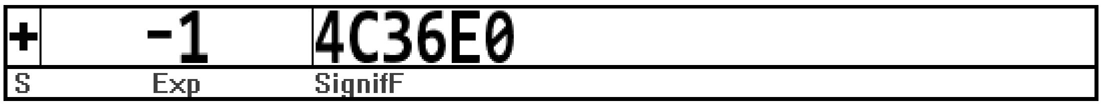


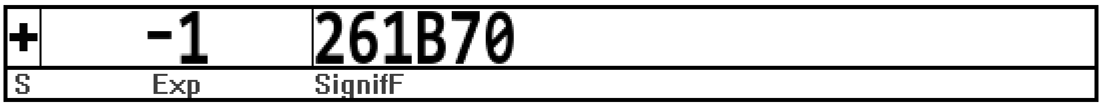
The value for the bit pattern stored in the register, as defined by the current coding options, is displayed using the current number display options.

The markers area will be empty.

4.3.2 Floating-Point Fields





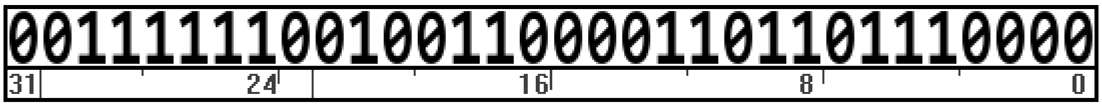


The bit pattern stored in the register is partitioned into fields as defined by the currently selected floating-point coding, and the content of each field is displayed as determined by the “Floating-point Fields Display Format” Number Display options. The location of the fields in the display will coincide with the location of the field within the word bit pattern

The markers area will contain the labels for each field, “S” for the sign field, “Exp” for the exponent field, “SignifF” for the significand fraction field of the IEEE 754 floating-point codings and “Signif” for the significand field of the Intel 80-bit floating-point coding.

The number entry keys will be disabled in this display mode, so may not be used to modify the contents of the X register.

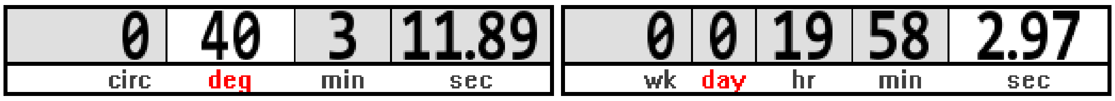
4.3.3 Floating-Point Fields Bit Pattern



This is the same as the Integer Word Bit Pattern display mode but the markers area has additional notches, full height, to indicate the locations of field boundaries.

The number entry keys will be disabled in this display mode, so may not be used to modify the contents of the X register.

4.3.4 Floating-Point Components



This display format only affects the main number display in the keypad/detached display window.

The value stored in the X register is partitioned into components as defined by the currently selected composite. The most significant component is displayed as a signed decimal integer value, the least significant (base) component is displayed with the number of decimal places set in the definition of the current composite and the intervening fields (if any) are displayed as unsigned decimal integer values. The width of a component in the display is determined by the maximum value that can be displayed for that component.

The currently selected component is identified by a different background color, as set on the Color options.

If the current digit size is not small then the markers area will contain the labels for each component. The current unit component is usually displayed in a different color from the others, as set in the Color options.

5 CALCULATOR ARCHITECTURE

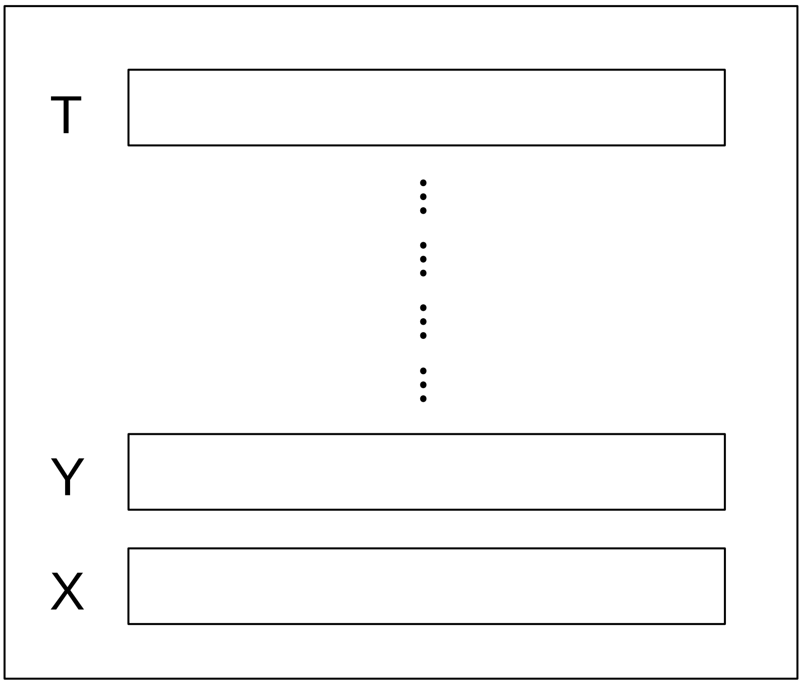
5.1 Registers

IO contains a number of storage areas it uses to contain the data on which it operates; these are called "registers". Each register should be thought of as a word with a width equal to that defined by the current Number Coding options, which contains a bit pattern. As the current Number Coding options will (nearly always) identify a numerical value which is mapped to by whatever the stored bit pattern may be, the register can also be considered to contain this value; although this is not strictly true it is easier for people to think in terms of the numbers being represented rather than the bit pattern stored, hence this document will regularly refer to the "value" stored in a register when it means the value represented by the bit pattern stored in the register using the current Number Coding options. This same distinction applies to all numerical data stored in any computer/digital device.

All well as a bit pattern, a register also contains information about its stored bit pattern which cannot be determined simply by inspecting the bit pattern.

The best example of the additional information is when the stored bit pattern was the result of a fixed/floating-point operation; the register contains an indication of whether or not the bit pattern represents the exact value of the result of the operation, or a value close to the exact value that can be represented using the current Number Coding options, the currently configured fixed/floating-point rounding type will determine which bit pattern will be used. So, if two registers contain the same bit pattern but different exact value indicators, then they contain different values; in the inexact case the exact value being represented is unknown but it is different from the exact value.

5.1.1 Register Stack



IO's register stack contains the registers which contain the data operated on by the majority of IO operations. The constituent registers are organized in a stack structure, which means that they are considered to be in a particular order, from bottom of the stack to top of the stack. The register at the bottom of the stack is referred to as the X register and is used as an input and output for many IO operations. The next register is referred to as the Y register and is used as an input to IO's operator operations. The register at the top of the stack is referred to as the T register. The registers between the Y and T registers do not have specific identifiers.

The number of registers in the register stack is determined by the Calculator Operation options. If the Calculator Type option is set to “Algebraic Logic” then the register stack will contain twenty registers. If the Calculator Type option is set to “Reverse Polish Notation” then the number of registers will be defined by the setting of the “Register Stack Size” option, which can be between four and twenty.

The contents of the bottom four registers of the register stack can be displayed in the Stack Display window. The content of the X register is displayed in the main display area of the keypad and detached display windows. The content of the X or Y registers can be displayed in the secondary number display area of the keypad and detached display windows.

5.1.2 Last X Register

Each time an operation changes the contents of the X register, the original content is stored in the Last X register. The contents of the X register and the Last X register can be exchanged using the Replace X With Last X operation, this can be useful if you accidently overwrite/replace the contents of the X register.

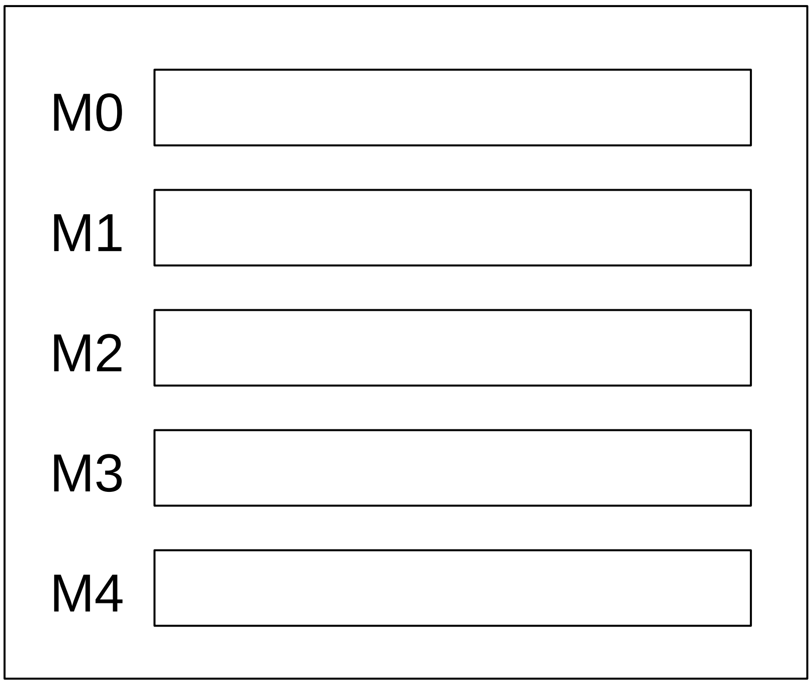
5.1.3 Algebraic Logic Previous Operation 2nd Operand Register

This register is only used by the Algebraic Logic calculator type, and then only when the “= key repeats previous operation” Calculator Operation option is selected.

If the = key is used when there are no pending operations, then the contents of this register are used as the 2nd operand of the repeated operation, if there is one.

The content of the stored previous operation 2nd operand, if any, can be displayed in the Stack Display window.

Memory Contents Registers



There are five memory registers, named M0, M1, M2, M3 and M4. At any time, one of these will be set to be the current memory register. All memory operations, with the exception of the polynomial and clear all memories operations, will only use the current memory register. The current memory register can be selected using the Memories Display window. An indication of which is the current memory register can be displayed in the secondary number display area of the keypad and detached display windows.

The current contents of all the memory registers can be displayed in the Memories Display window.

5.1.4 Memory Previous Contents Registers

Each memory register has an associated previous contents register. Each time an operation changes the contents of a memory register, the original content is stored in its previous contents register. The contents of a memory register and its previous contents can be exchanged using the Memories Display window, this can be useful if you accidently overwrite/replace the contents of the current memory register.

5.2 Stored Algebraic Logic Operators

These storage areas are only used by the Algebraic Logic calculator type.

5.2.1 Algebraic Logic Operator Stack

This stack will contain each operator which needs to be applied to perform the entered calculation, but cannot be applied yet due to the precedences of the entered operators or due to the use of the open-bracket key.

Each entry will contain either a pending operator or an indication of the use of the open-bracket key. The maintenance of the algebraic logic operator stack is performed automatically by IO when using the Algebraic Logic calculator type. The user does not need to be concerned with the operation of this stack; it is all handled by IO in response to the operations requested by the user.

The contents of the bottom four entries of the algebraic logic operator stack can be displayed in the Stack Display window.

5.2.2 Algebraic Logic Previous Operator

If the = key is used when there are no pending operations, then this identifies the operation that will be repeated, if there is one.

The stored previous operator, if any, can be displayed in the Stack Display window.

5.3 Calculator Flags

IO maintains a set of seven flags analogous to some of the status flags maintained by the majority of CPUs. Each flag is represented internally by a single bit which may be set to either ⓪ or ①.

The current settings of the flags are displayed in the Flags Status Display, if a flag is set to ⓪ then its identifying letter is not displayed, if a flag is set to ① then its identifying letter is displayed.

Some flag settings are determined by the current contents of the X register, the others are determined by the processing performed by the last operation.

Not all flags are defined for all number coding types. Their settings are not displayed for number coding types for which they are not defined.

5.3.1 Carry (C)

The setting of the Carry flag is determined by the last operation performed by IO. The setting of the Carry flag is defined for the integer and fixed-point number coding types, not for floating-point.

The Carry flag can be thought of as an extra bit of the X register, located immediately to the left of the msb of the word contained in the X register, but not part of it.

For addition and subtraction types of operation using integer unsigned coding, if the correct result can be represented using the current coding, then the Carry flag will be set to ⓪, if not then it will be set to ① to provide the required setting of the msb for the correct result if the word was one bit wider.

The Carry flag can be used by software to implement arithmetic operations on values which are represented by more than one word in memory.

5.3.2 Zero (Z)

The setting of the Zero flag is determined by the contents of the X register. The setting of the Zero flag is defined for all number coding types.

The setting of the Zero flag is controlled by the current number coding type and the setting of the associated “Z and N Flag Setting Control” Calculator Operation option. See the description of this option for an explanation of how the flag setting is defined.

5.3.3 Negative (N)

The setting of the Negative flag is determined by the contents of the X register. The setting of the Negative flag is defined for all number coding types.

The setting of the Negative flag is controlled by the current number coding type and the setting of the associated “Z and N Flag Setting Control” Calculator Operation option. See the description of this option for an explanation of how the flag setting is defined.

5.3.4 Overflow (V)

The setting of the Overflow flag is determined by the last operation performed by IO

The Overflow flag is only affected when using any of the following codings:

Integer Unsigned
Integer 2’s Complement
Fixed-point Unsigned
Fixed-point 2’s Complement

Operations performed using other coding will set the Overflow flag to ⓪.

And, it is only affected by the following operations:

Addition
Subtraction
Add With Carry
Subtract With Carry/Borrow
Increment
Decrement

Other operations will set the Overflow flag to ⓪.

Just about all CPUs which can perform integer arithmetic have an overflow flag, usually denoted by V. Which CPU instructions modify the overflow flag differs for different CPU architectures, but all will set it to the required setting for integer addition and subtraction instructions. No distinction is made between unsigned and 2’s complement coding, the same CPU instructions are used as the processing of the bit patterns used as operands is identical. This flag does not provide any useful information when using unsigned coding; it only accurately indicates an overflow when using 2’s complement coding.

To mirror this behavior, IO sets the Overflow flag to ⓪ if the bit pattern resulting from the operation represents the correct value if the operand(s) and result are interpreted using 2’s Complement coding, irrespective of the currently selected number coding, otherwise, IO sets the Overflow flag to ①.

The Value Overflow flag is used to indicate an overflow for all integer and fixed-point codings, and for these and other operations.

5.3.5 Denormal (D)

The setting of the Denormal flag is determined by the contents of the X register. The setting of the Denormal flag is defined for the floating-point number coding type, not for integer and fixed-point.

If the bit pattern stored in the X register is of class Denormal then this flag is set to ①, otherwise it is set to ⓪.

5.3.6 Exact (E)

The setting of the Exact flag is determined by the last operation performed by IO. The setting of the Exact flag is defined for the fixed-point and floating-point number coding types, not for integer.

If the last operation produced an exact result, then this flag is set to ①, otherwise it is set to ⓪.

5.3.7 Value Overflow (O)

The setting of the Value Overflow flag is determined by the last operation performed by IO. The setting of the Value Overflow flag is defined for the integer and fixed-point number coding types, not for floating-point.

If the correct result of an operation can be represented using the current coding, then the Overflow flag will be set to ⓪, if not then it will be set to ①. If the Overflow flag (V) is set to ① by an operation then the Value Overflow flag will also be set to ①, in this case V will be displayed in the Flags Status Display but O will not.

For the floating-point number coding type, an overflow generated by the last operation is indicated by the resulting bit pattern of class Infinity being displayed as “+Inf” or “-Inf” in the main number display.

5.3.8 Summary

The following table shows the type of each flag and identifies the number coding type(s) for which it is defined.

Flag		Type	Number Coding Type		
			Integer	Fixed-point	Floating-point
C	Carry	Operation	✔	✔	✖
Z	Zero	Content	✔	✔	✔

N	Negative	Content	✔	✔	✔
V	Overflow	Operation	✔	✔	✖
D	Denormal	Content	✖	✖	✔
E	Exact	Operation	✖	✔	✔
O	Value Overflow	Operation	✔	✔	✖

5.4 Operation Processing

5.4.1 Terminology

Note that in the following text, and throughout this document, “IO” is used to refer to the calculator; it is never used as an abbreviation for “Input/Output” as it is in most other computer related documentation.

Although not completely mathematically accurate, for the sake of simplicity and ease of understanding, the following terminology will be used in this document:

An “IO Operation” is some processing performed by IO which is initiated by the user, usually by clicking on a key or using a keyboard shortcut, though there are also other methods. Some IO operations use current register contents as inputs, some modify register contents as their output, some modify option settings, and most will modify the content of one or more of IO’s window(s).

A ”Bit Pattern Operation” is an IO operation which uses the bit pattern(s) contained in registers as inputs and generates a bit pattern as an output. The coding form of the current number coding options does not affect these operations. e.g., bitwise and, shift left, reverse bytes.

A ”Value Operation” is an IO operation which uses the value(s) represented by the bit pattern(s) contained in registers as inputs and generates a bit pattern representing the resulting value as an output. The values used and generated will be determined by the current number coding options. e.g., add, multiply, sine.

An “IO Function” is an IO operation that is applied to the contents of the X register, which is then replaced with the result of the operation, e.g., Square Root, Cosine, Reverse Bytes, etc. The original contents of the X register are referred to as the “argument” of the function.

An “IO Operator” is an IO operation that is applied to the contents of the Y and X registers, e.g., Add, Root, etc. The contents of the Y and X registers when the operator is applied are referred to as the first and second “operands” of the operator respectively.

5.4.2 General IO Operation Processing

IO operations which require a calculation to be performed, using the contents of one or more registers as inputs, will, follow these general procedures:

Bit Pattern Operations

- All inputs will be bit patterns of the word width selected for the current number coding type
- The specified bit pattern operation, and any intermediate bit pattern operations which are required, will be performed using this word width
- The resulting bit pattern will be of this width

e.g., for a number coding of integer unsigned 8-bit, if the Y register contains a value of 14 and the X register contains a value of 60, and the bitwise bit clear operator is applied, then the 8-bit bit pattern stored in the X register, i.e.,⓪⓪①①①①⓪⓪, is complemented to give the 8-bit bit pattern ①①⓪⓪⓪⓪①①, which is then bitwise and'ed with the 8-bit bit pattern stored in the Y register, i.e.,⓪⓪⓪⓪①①①⓪, to give the 8-bit bit pattern of ⓪⓪⓪⓪⓪⓪①⓪ as the result of the operation.

Integer Value Operations

- All input values will be converted to integer 128-bit 2's complement coding
- The specified value operation, and any intermediate value operations which are required, will be performed using this coding
- The resulting value will be converted to the current integer coding, truncating or saturating the result if required by the current configuration, to provide the final result

Fixed/Floating-Point Value Operations

- All input values will be converted to Intel 80-bit floating-point coding
- The specified value operation, and any intermediate value operations which are required, will be performed using this coding
- The resulting value will be converted to the current fixed/floating-point coding, using the currently configured rounding type if required, to provide the final result

e.g., for a number coding of IEEE 32-bit, if the cosecant function is applied, then the 32-bit floating-point value stored in the X register converted to 80bit floating-point coding, the sine of this value is obtained as an 80-bit floating-point value, this value is then divided into an 80-bit floating-point value of 1 to give the 80-bit floating-point result, and this result is then converted to 32-bit floating-point coding, using the configured rounding type if required, to provide the final result of the operation.

5.4.3 IO Function Processing

The user requests the processing of a function by clicking on the appropriate key (or using a keyboard shortcut) after having set up the required argument in the X register.

A function is always processed immediately, i.e., the contents of the X register is overwritten with the result of the function applied to the original contents, no other registers are affected.

The flags may be modified to reflect the result and/or the processing performed.

5.4.4 IO Operator Processing

The processing performed for an IO Operator will depend on the Calculator Type currently selected in the Calculator Operation options. The following descriptions can be observed in action by having the Stack Display window visible while the processing is being performed.

5.4.4.1 Reverse Polish Notation Calculator Type

The user requests the processing of an operator by clicking on the appropriate key (or using a keyboard shortcut) after having set up the required operands in the Y and X registers. If the entry of a number was in progress at this point in time, then that number entry is now deemed to have been completed.

The requested operator is immediately applied to the contents of the Y and X registers, the contents of the Y register are overwritten with the result and the register stack is then popped so that the result resides in the X register.

The flags may be modified to reflect the result and/or the processing performed.

5.4.4.2 Algebraic Logic Calculator Type

The user requests the processing of an operator by clicking on the appropriate key (or using a keyboard shortcut) after having set up the required first operand in the X register. If the entry of a number was in progress at this point in time, then that number entry is now deemed to have been completed.

If operator precedence is enabled in the Calculator Operation options then any higher priority operations at the bottom of the operator stack that can now be applied are processed.

Then the current contents of the X register are pushed onto the register stack again so that the X and Y registers both contain this value, and the requested operator is pushed onto the operator stack.

Note that the requested operator cannot be applied yet as the second operand has not yet been supplied, this will normally be entered next and overwrite the duplicated contents in the X register.

The flags may be modified to reflect the result and/or the processing performed by the last operator applied.

6 OPTIONS

The operation and appearance of IO is controlled by the settings of a large number of options.

The options can be changed by the user in a number of ways:

- All options can be set by using the appropriate window displayed by selecting the required item in the Options menu, or by shift-double-clicking on an associated status display
- Some options can be set by clicking on a designated key
- Some options can be set by using the keyboard to enter a keyboard shortcut
- Some options can be set by using a keypad popup window usually displayed by double-clicking on a status display

6.1 Option Sets

Due to the large number of options, they are partitioned into option sets. Every option belongs to one, and only one, option set. Each option set has a window which can be used to set any of its constituent options.

There are eight option sets:

- Number Coding
- Number Display
- Calculator Operation
- Digit Set
- Key Display
- Window
- Clipboard
- Color

6.1.1 Option Set Windows

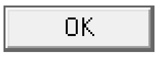
Each option set has an associated window which can be used to modify the settings of the options contained within it.

An option set window can be opened by selecting the appropriate item from the Options menu; some may be displayed by shift-double-clicking on an associated status display.

6.1.1.1 Common Features

Each of these windows contains a number of controls which can be used to make the required option settings. However, there are a number of features which are common to all option set windows.

All option set windows contain the following six buttons:



Sets all the options to the settings currently defined by the window controls, and then close the window. Exactly the same will be done if the “enter” keyboard key is pressed. If any of the control settings are invalid then a message identifying the error will be displayed and the current option settings will remain unchanged.

Closes the window leaving all the options as they were when the window was opened if the Apply button has not been used since then, or as they were when the Apply button was last used if it has. Exactly the same will be done if the Close button (the X at the top right corner of the window) is clicked, or the “esc” keyboard key is pressed.

Sets all the window controls to reflect the default option settings from the default configuration.

Sets all the window controls to reflect the options as they were when the window was opened.

Sets all the options to the settings currently defined by the window controls, and leave the window displayed. If any of the control settings are invalid then a message identifying the error will be displayed and the current option settings will remain unchanged.

Allows you to select a configuration file, and the window controls will then be set to reflect the option settings stored in that file. Alternatively, a configuration file can be selected by dragging it from an Explorer window and dropping it onto this button.

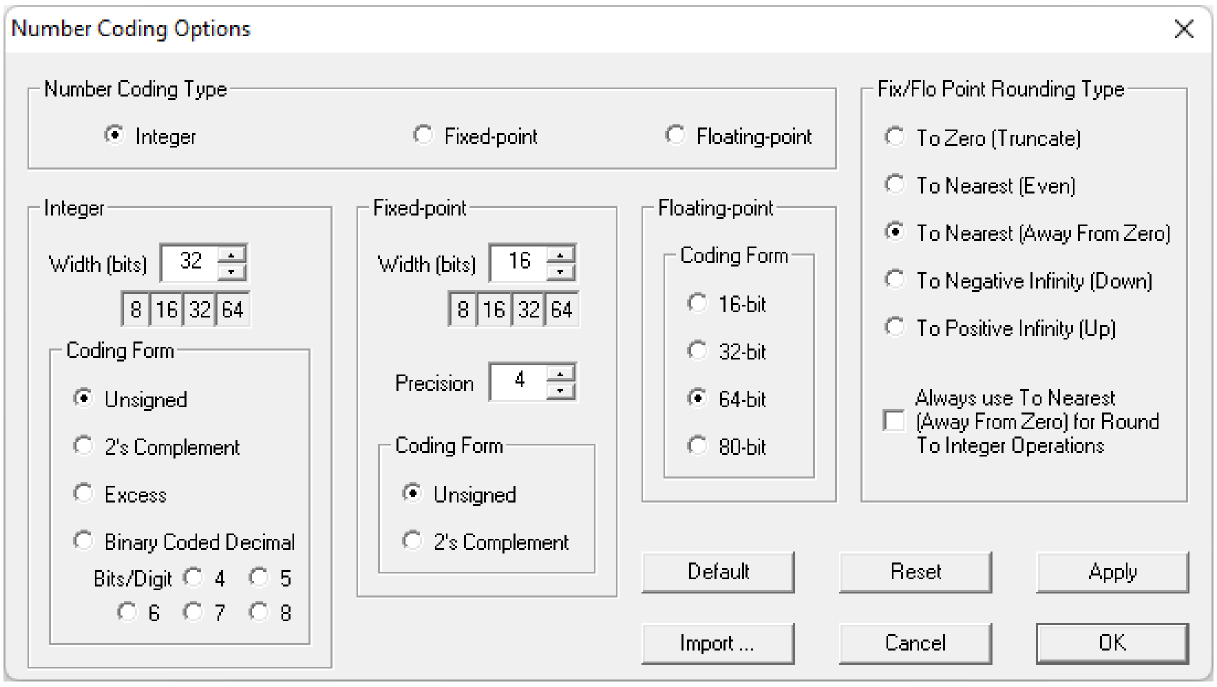
To aid understanding of the relationships between options within an option set, the controls to modify options which are not used with the current settings of other options within the options set, are disabled and will be grayed out.

Note that the appearance, e.g., colors, of your windows may differ from those shown in the following sections depending on the version of Windows being used, and the chosen Windows display settings.

6.2 Number Coding Options

These options control how IO maps the bit patterns stored in registers to/from the numbers they represent.

6.2.1 Option Set Window


Number Coding Options

Number Coding Type: [✓ Integer] Fixed-point Floating-point

Integer: Width (bits) 32 (8, 16, 32, 64); Coding Form: [✓ Unsigned] 2's Complement Excess Binary Coded Decimal; Bits/Digit 4 5 6 7 8

Fixed-point: Width (bits) 16 (8, 16, 32, 64); Precision 4; Coding Form: [✓ Unsigned] 2's Complement

Floating-point: Coding Form: 16-bit 32-bit [✓ 64-bit] 80-bit

Fix/Flo Point Rounding Type: To Zero (Truncate) To Nearest (Even) [✓ To Nearest (Away From Zero)] To Negative Infinity (Down) To Positive Infinity (Up)

☐ Always use To Nearest (Away From Zero) for Round To Integer Operations

The number coding option set window may be displayed by selecting the “Number Coding …” item from the Options menu, or by shift-double-clicking on the number coding status display.

6.2.1.1 Number Coding Type

These controls select the Number Coding Type to be used.

Number Coding Type

These buttons select one of the three number coding types and enables the controls for setting the options specific to that number coding type.

This option can take one of three settings:

• Integer

Selects the Integer number coding type and enables controls to allow the selection of the width and form of coding.

• Fixed-point

Selects the Fixed-point number coding type and enables controls to allow the selection of the width, precision, form of coding and rounding type.

• Floating-point

Selects the floating-point number coding type and enables controls to allow the selection of the width/coding combination and rounding type.

6.2.1.2 Integer

These controls select the Integer Number Coding Type options.

Width (bits)

Defines the number of bits contained in a stored bit pattern used to represent an integer value. The value entered must be in the range 1 to 96.

Clicking the numbered controls allow you to quickly select a width of 8, 16, 32 or 64 bits.

Coding Form

These controls select the form of Integer Coding to be used.

This option can take one of four settings:

• Unsigned

Selects unsigned integer coding. If a width of 1 is specified then unsigned coding is automatically selected.

• 2’s Complement

Selects 2's complement signed integer coding.

• Excess

Selects excess signed integer coding. If the excess value for the selected width is not too large it is displayed here.

• Binary Coded Decimal

Selects binary coded decimal unsigned integer coding, and enables the Bits/Digit controls.

Bits/Digit

Selects the number of bits for each decimal digit when binary coded decimal unsigned integer coding is selected. Only values in the range four to eight which are valid for the currently selected integer width, i.e., divide exactly into the width, will be available for selection.

6.2.1.3 Fixed-point

These controls select the Fixed-point Coding Type options.

Width (bits)

Selects the width of the bit pattern to be used to represent fixed-point values. The value entered must be in the range 1 to 64.

Clicking the numbered controls allow you to quickly select a width of 8, 16, 32 or 64 bits.

Precision

Selects the precision of the bit pattern to be used to represent fixed-point values. The value entered defines the position of the binary point within or outside the bit pattern. A value of 0 positions the binary point immediately to the right of the least significant bit of the bit pattern, a positive value positions it to the left of the 0 position by that many bits, a negative value positions it to the right of the 0 position by N bits, where N is the modulus of the value entered. The value entered must be in the range -99 to +99.

Coding Form

These controls select the form of Fixed-point Coding to be used.

This option can take one of two settings:

• Unsigned

Selects unsigned fixed-point coding. If a width of 1 is specified then unsigned coding is automatically selected.

• 2’s Complement

Selects 2's complement signed fixed-point coding

6.2.1.4 Floating-point

These controls select the Floating-point Coding Type options. The width of the bit pattern to be used to represent floating-point values is dictated by the selected Coding Form option.

Coding Form

These controls select the form of floating-point coding to be used.

This option can take one of four settings:

• 16-bit

Selects IEEE 754 Half Precision coding (binary 16)

• 32-bit

Selects IEEE 754 Single Precision coding (binary 32)

• 64-bit

Selects IEEE 754 Double Precision coding (binary 64)

• 80-bit

Selects Intel Extended Precision coding

6.2.1.5 Fix/Flo-point Rounding Type

This option determines the type of rounding used to store a result in the X register when the exact result cannot be stored using the current fixed/floating-point coding. If an exact result is stored then this option has no effect.

Rounding Type

This option can take one of five settings:

• To Zero (Truncate)

The value stored is the largest (magnitude) representable value of the same sign which is closer to zero than the exact result.

• To Nearest (Even)

The value stored is the representable value which is closest to the exact result. If there are two such values then that with an lsb of zero is used.

• To Nearest (Away From Zero)

The value stored is the representable value which is closest to the exact result. If there are two such values then that with the larger magnitude is used.

• To Negative Infinity (Down)

The value stored is the highest representable value which is less than the exact result.

• To Positive Infinity (Up)

The value stored is the lowest representable value which is greater than the exact result.

Always use To Nearest (Away From Zero) for Round To Integer Ops

Setting this option causes the "To Nearest (Away From Zero)" rounding type to be used for fixed-point or floating-point Round To Integer operations, otherwise the rounding type selected above will be used

6.2.1.6 Error Messages

The following error messages may be displayed after the “Apply” or “OK” buttons are used to indicate that the control settings would result in an invalid set of options. The current option settings will remain unchanged.

<u>Invalid Integer Number Width</u>

The Integer “Width (bits)” control contains a value outside the valid range of 1 to 96.

Invalid number of bits per BCD Digit

The Integer Coding Binary Coded Decimal “Bits/Digit” control contains a value outside the valid range of 4 to 8.

Invalid Integer Number Width for specified BCD Coding

The Integer Binary Coded Decimal Coding is selected and the Integer “Width (bits)” control contains a value which is not a multiple of the value specified in the “Bits/Digit” control.

Invalid Fixed-point Number Width

The Fixed-point “Width (bits)” control contains a value outside the valid range of 1 to 64.

Invalid Fixed-point Precision

The Fixed-point “Precision” control contains a value outside the valid range of -99 to +99.

Number Coding Option Change Handling options prevent this change for current X register content

The current setting in the Calculator Operation Options of the Number Coding Option Change Handling Incompatible Register Contents Handling option for changes from the current number coding type to the selected number coding type is set to “Prevent Change” and the X register contains a bit pattern/value (depending on the current setting of the associated Register Contents Preservation option) which cannot be preserved in a change to the coding specified by the control settings.

Number Coding Option Change Handling options prevent this change for current stack registers contents

The current setting in the Calculator Operation Options of the Number Coding Option Change Handling Incompatible Register Contents Handling option for changes from the current number coding type to the selected number coding type is set to “Prevent Change” and there is a stack register, other than the X register, which contains a bit pattern/value (depending on the current setting of the associated Register Contents Preservation option) which cannot be preserved in a change to the coding specified by the control settings.

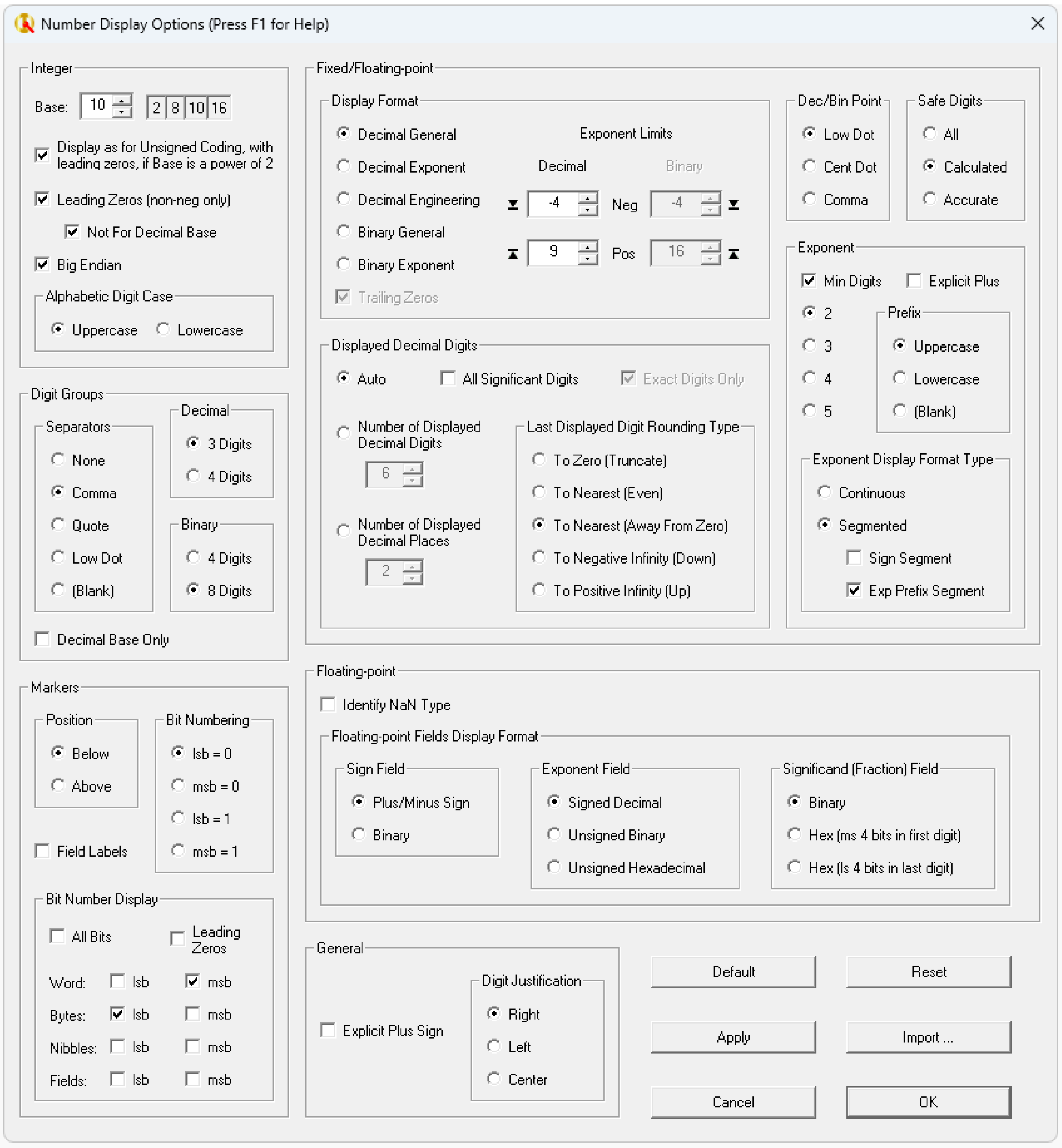
Number Coding Option Change Handling options prevent this change for current memory registers contents

The current setting in the Calculator Operation Options of the Number Coding Option Change Handling Incompatible Register Contents Handling option for changes from the current number coding type to the selected number coding type is set to “Prevent Change” and there is a memory register which contains a bit pattern/value (depending on the current setting of the associated Register Contents Preservation option) which cannot be preserved in a change to the coding specified by the control settings.

6.3 Number Display Options

These options control how IO displays the values represented by the bit patterns stored in registers.

6.3.1 Option Set Window


Number Display Options (Press F1 for Help)

Integer
Base: 10 | 2 | 8 | 10 | 16
[✓ Display as for Unsigned Coding, with leading zeros, if Base is a power of 2]
[✓ Leading Zeros (non-neg only)]
[✓ Not For Decimal Base]
[✓ Big Endian]
Alphabetic Digit Case: [✓ Uppercase] Lowercase

Digit Groups
Separators: None [✓ Comma] Quote Low Dot (Blank)
Decimal: [✓ 3 Digits] 4 Digits
Binary: 4 Digits [✓ 8 Digits]
Decimal Base Only

Markers
Position: [✓ Below] Above
Bit Numbering: [✓ lsb = 0] msb = 0 lsb = 1 msb = 1
Field Labels
Bit Number Display: All Bits Leading Zeros
Word: lsb [✓ msb]
Bytes: [✓ lsb] msb
Nibbles: lsb msb
Fields: lsb msb

Fixed/Floating-point
Display Format: [✓ Decimal General] Decimal Exponent Decimal Engineering Binary General Binary Exponent [✓ Trailing Zeros]
Exponent Limits: Decimal Neg -4, Pos 9; Binary Neg -4, Pos 16
Displayed Decimal Digits: [✓ Auto] All Significant Digits [✓ Exact Digits Only]
Number of Displayed Decimal Digits 6
Number of Displayed Decimal Places 2
Last Displayed Digit Rounding Type: To Zero (Truncate) To Nearest (Even) [✓ To Nearest (Away From Zero)] To Negative Infinity (Down) To Positive Infinity (Up)
Dec/Bin Point: [✓ Low Dot] Cent Dot Comma
Safe Digits: All [✓ Calculated] Accurate
Exponent: [✓ Min Digits] Explicit Plus
[✓ 2] 3 4 5
Prefix: [✓ Uppercase] Lowercase (Blank)
Exponent Display Format Type: Continuous [✓ Segmented]
Sign Segment [✓ Exp Prefix Segment]

Floating-point
Identify NaN Type
Floating-point Fields Display Format
Sign Field: [✓ Plus/Minus Sign] Binary
Exponent Field: [✓ Signed Decimal] Unsigned Binary Unsigned Hexadecimal
Significand (Fraction) Field: [✓ Binary] Hex (ms 4 bits in first digit) Hex (ls 4 bits in last digit)

General
Explicit Plus Sign
Digit Justification: [✓ Right] Left Center

The number display option set window may be displayed by selecting the “Number Display …” item from the Options menu, or by shift-double-clicking on the number display status display or the exponent limits status display.

6.3.1.1 Integer

These options apply only to the integer number coding type.

Base

Defines the number base used to display integer values.

The following number bases are available:

Base Value (in decimal)	Name(s)
2	Binary
3	Ternary, Trinary
4	Quaternary
5	Quinary, Pental
6	Senary
7	Septenary
8	Octal, Octonary, Octonal, Octimal
9	Nonary
10	Decimal, Denary
11	Unidecimal
12	Duodecimal
13	Tridecimal
14	Tetradecimal
15	Pentadecimal
16	Hexadecimal, Sexadecimal

Clicking the numbered controls allow you to quickly select a base of two (binary), eight (octal), ten (decimal) or sixteen (hexadecimal).

Display as for Unsigned Coding, with leading zeros, if Base is a power of 2

If this option is selected when the “Base” option is set to two, four, eight or sixteen, then the value defined by the unsigned form of integer coding will be displayed, with leading zeros, irrespective of the form selected in the Number Coding options. This means that each displayed digit will represent the contents of a sequence of bits in the stored bit pattern, which would not always be the case for other selected forms of coding.

Leading Zeros (non-neg only)

Setting this option will enable the display of leading zero digits in locations where a valid digit can appear for the current combination of Number Coding and Number Display Options. Leading zeros will not be displayed for negative values.

Not For Decimal Base

Setting this option will prevent the display of leading zeros if a number base of ten is selected.

Big Endian

Setting this option specifies that the first character displayed/entered via the Character Data Status Display is the one stored in the most significant byte of the stored bit pattern, otherwise the first character is stored in the least significant byte.

Alphabetic Digit Case

This option defines the case of letters used to display digits greater than nine when a number base greater than ten is selected. It also affects the labels for the corresponding digit entry keys.

This option can take one of two settings:

• Upper

Selects uppercase letters

• Lower

Selects lowercase letters

6.3.1.2 Fixed/Floating-Point

These options apply to fixed-point and floating-point number coding types.

Display Format

This option can take one of five settings:

• Decimal General

If the decimal exponent value lies between the selected decimal exponent limits, then the value is displayed in the traditional format, i.e., integer digits to the left of a decimal point (if any) and fractional digits (if any) to the right. Otherwise, the decimal exponent format is used.

• Decimal Exponent

The decimal exponent format is used, there will be a mantissa containing one integer decimal digit, followed by a decimal exponent.

• Decimal Engineering

Similar to the decimal exponent format, but the number of integer digits contained in the mantissa will be one, two or three so that the exponent value is a multiple of three.

• Binary General

If the binary exponent value lies between the selected binary exponent limits, then the value is displayed in binary floating-point format, i.e., integer bits to the left of a binary point (if any) and fractional bits (if any) to the right. Otherwise, the binary exponent format is used.

• Binary Exponent

The binary exponent format is used, there will be a mantissa containing one integer bit, followed by a binary exponent expressed in decimal

---

Trailing Zeros

Specifies that trailing zeros are to be displayed for binary display formats. The last trailing zero displayed (if any) will correspond to the least significant bit of the bit pattern for the current coding. This option is ignored for 2’s complement fixed-point codings.

Decimal Negative Exponent Limit

Specifies the negative exponent limit used to determine if the decimal general format should use the decimal exponent format. The decimal exponent format will be used if the exponent value is less than this limit.

Decimal Positive Exponent Limit

Specifies the positive exponent limit used to determine if the decimal general format should use the decimal exponent format. The decimal exponent format will be used if the exponent value is greater than this limit.

Binary Negative Exponent Limit

Specifies the negative exponent limit used to determine if the binary general format should use the binary exponent format. The binary exponent format will be used if the exponent value is less than this limit.

Binary Positive Exponent Limit

Specifies the positive exponent limit used to determine if the binary general format should use the binary exponent format. The binary exponent format will be used if the exponent value is greater than this limit.

⤓ These set the decimal/binary negative exponent limit to the lowest permitted value

⤒ These set the decimal/binary positive exponent limit to the highest permitted value

---

Displayed Decimal Digits

These options define what decimal digits are displayed. All bits are always displayed for binary display formats.

This option can take one of three settings:

• Auto

Displays a number of significant digits equal to the number of safe digits for the current coding. This may be increased if the “All Significant Digits” option is selected.

• Number of Displayed Decimal Digits

The selected number of decimal digits will be displayed. Some calculators refer to this as specifying the number of displayed significant digits, but that is misleading because if this setting causes trailing zero digits to be displayed after the decimal point, then those digits are not significant.

• Number of Displayed Decimal Places

The selected number of digits will be displayed after the decimal point.

All Significant Digits

If the Auto option is selected, then setting this option causes all significant digits to be displayed. Note that for very small or very large values this can result in thousands of displayed digits.

Exact Digits Only

If the “Auto” and the “All Significant Digits” options are selected, then setting this option will cause all significant digits to only be displayed for exact values.

Number of Displayed Decimal Digits

Specifies the number of decimal digits to be displayed if the “Num Disp Decimal Digits” option is selected.

Number of Displayed Decimal Places

Specifies the number of digits displayed after the decimal point if the “Num Disp Decimal Places” option is selected.

Last Displayed Digit Rounding Type

This option determines the type of rounding applied to the last displayed decimal digit when the full value cannot be displayed using the current Number Display Options.

This option can take one of five settings:

• To Zero (Truncate)

The value displayed is the largest displayable value of the same sign which is closer to zero than the full value.

• To Nearest (Even)

The value displayed is the displayable value which is closest to the full value. If there are two such values then the one with an even least significant digit is displayed.

• To Nearest (Away From Zero)

The value displayed is the displayable value which is closest to the full value. If there are two such values then the larger magnitude one is displayed.

• To Negative Infinity (Down)

The value displayed is the greatest displayable value which is less than the full value.

• To Positive Infinity (Up)

The value displayed is the least displayable value which is greater than the full value.

Dec/Bin Point

This option determines how decimal and binary points are displayed.

This option can take one of three settings:

• Low Dot

A dot character near the bottom of the digits is used.

• Cent Dot

A dot character near the middle of the digits is used.

• Comma

A comma character near the bottom of the digits is used.

If an option is selected that clashes with the currently selected digit group separators, then a different digit group separator is selected to avoid the clash.

Safe Digits

This option determines how the safe digits of inexact results are displayed.

This option can take one of three settings:

• All

All digits are displayed as safe.

• Calculated

A calculated number of digits are displayed as safe. This number is determined by the current fixed/floating-point coding, and the number of significant bits for denormal bit patterns.

• Accurate

The accurate number of digits are displayed as safe. This number is determined by the number of common initial digits between the lower and upper limits for the value.

Safe and unsafe digits are distinguished by setting different colors for them in the color options.

Min Digits

Setting this option causes at least the selected number of digits will be displayed for exponent values.

Minimum number of Exponent Digits

This option sets the minimum number of displayed exponent digits of the “Min Digits” option is selected.

This option can take one of four settings:

• 2

Displays at least two digits for exponent values.

• 3

Displays at least three digits for exponent values.

• 4

Displays at least four digits for exponent values.

• 5

Displays at least five digits for exponent values.

Explicit Plus

Setting this option will display positive exponent values prefixed with a "+".

Exponent Prefix

This option can take one of three settings:

• Uppercase

Exponent values will be preceded by an uppercase "E" (decimal) or "B" (binary).

• Lowercase

Exponent values will be preceded by a lowercase "e" (decimal) or "b" (binary).

• (Blank)

Exponent values will be preceded by a blank digit cell.

Exponent Display Format Type

These options determine how exponent number display formats are displayed

This option can take one of two settings:

• Continuous

A continuous sequence of digits is displayed.

• Segmented

The digits are segmented in the display. The mantissa and exponent values have their own segments. This allows long mantissas to be scrolled independently; the other segments will always be displayed in full.

Sign Segment

If this option is selected then a sign segment will be displayed to the left of the mantissa segment and a sign will not be displayed in the mantissa segment.

Exp Prefix Segment

If this option is selected then an exponent prefix segment will be displayed between the mantissa and exponent value segments.

6.3.1.3 Floating-Point

These options apply only to the floating-point number coding type.

Identify NaN Type

If this option is selected then the type of NaN will be displayed (QNaN for Quiet NaN, SNaN for Signaling NaN). If not selected then just NaN will be displayed for both.

The following options define how field contents are displayed in Floating-point Fields Display Mode

Sign Field

This option controls the display of the sign field.

This option can take one of two settings:

• Plus/Minus Sign

The sign field is displayed as a + or – character.

• Binary

The sign field is displayed as a 0 or 1 digit,

Exponent Field

This option controls the display of the exponent field.

This option can take one of three settings:

• Signed Decimal

The signed exponent value is displayed in decimal.

• Unsigned Binary

The exponent field is displayed as unsigned binary with leading zeros, effectively displaying the field bit pattern.

• Unsigned Hexadecimal

The exponent field is displayed as unsigned hexadecimal with leading zeros.

Note that exponent values are only displayed for Normal floating-point values. For other classes the bit pattern contained in the exponent field will be displayed, this will be either all 0s or all 1s.

Significand (Fraction) Field

This option controls the display of the significand/significand fraction field. The field contents will always be interpreted as an unsigned integer value and displayed with leading zeros.

This option can take one of three settings:

• Binary

The significand/significand fraction field is displayed in binary; the digits will fill the field.

• Hex (ms 4 bits in first digit)

The significand/significand fraction field is displayed in hexadecimal, with the significand value shifted left by the required number of bits so that the most significant four bits are represented by the most significant displayed digit. The digits, and the label, will be left justified within the field,

• Hex (ls bits in last digit)

The significand/significand fraction field is displayed in hexadecimal, so the least significant four bits are represented by the least significant displayed digit. The digits, and the label, will be left justified within the field,

---

6.3.1.4 Digit Groups

These options control the display of separators of groups of digits in displayed values. Only digits in the integer part of displayed values will be grouped (for displays containing an exponent this means digits to the left of the decimal/binary point). For a number base other than ten or two each group will contain four digits.

Separators

This option can take one of five settings:

• None

No group separators will be displayed.

• Comma

Groups will be separated by a comma character near the bottom of the digits.

• Quote

Groups will be separated by a single quote character near the top of the digits.

• Low Dot

Groups will be separated by a dot character near the bottom of the digits.

• (Blank)

Groups will be separated by a narrow blank area.

If an option is selected that clashes with the currently selected decimal/binary point then a different decimal/binary point is selected to avoid the clash.

Decimal Base Only

Setting this option causes group separators to only be displayed for a number base of ten (for all number coding types).

Decimal

This option sets the number of digits in each group when the base is ten.

This option can take one of two settings:

• 3 Digits

There will be three digits in each group.

• 4 Digits

There will be four digits in each group.

Binary

This option sets the number of digits in each group when the base is two.

This option can take one of two settings:

• 4 Digits

There will be four digits in each group.

• 8 Digits

There will be eight digits in each group.

6.3.1.5 Markers

These options control what is displayed in the markers region, which annotates the contents of the main number display.

Position

This option controls the position of the markers relative to the main number display.

This option can take one of two settings:

• Below

The markers are displayed below the main number display.

• Above

The markers are displayed above the main number display.

Field Labels

Setting this option will display the field labels instead of bit numbers in Integer Fields Values Display Mode.

Bit Numbering

This option controls how bits are numbered in the markers; they will always be in numerical sequence from one end of the bit pattern to the other.

This option can take one of four settings:

• lsb = 0

The least significant bit is numbered 0.

• msb = 0

The most significant bit is numbered 0.

• lsb = 1

The least significant bit is numbered 1.

• msb = 1

The most significant bit is numbered 1.

Bit Number Display

These options control which bit numbers are displayed in the markers, and how. Bit numbers will only be displayed in the Integer Fields Values Display Mode and the Bit Pattern Display Modes.

All Bits

Setting this option will display the bit numbers for all bits in the bit pattern.

Leading Zeros

Setting this option will display bit numbers with leading zeros.

The following option settings identify which bit numbers are displayed; they will be disabled if the “All Bits” options is set. You may set any combination of the least and/or most significant bits of four types of sequences of bits within the stored bit pattern.

Word: lsb

Displays the bit number of the lsb of the entire stored bit pattern.

Word: msb

Displays the bit number of the msb of the entire stored bit pattern.

Bytes: lsb

Displays the bit numbers of the lsb of each group of eight bits, starting at the lsb.

Bytes: msb

Displays the bit numbers of the msb of each group of eight bits, starting at the lsb.

Nibbles: lsb

Displays the bit numbers of the lsb of each group of four bits, starting at the lsb.

Nibbles: msb

Displays the bit numbers of the msb of each group of four bits, starting at the lsb.

Fields: lsb

In Integer Fields Values Display Mode, displays the bit numbers of the lsb of each field in the selected field layout.

Fields: msb

In Integer Fields Values Display Mode, displays the bit numbers of the msb of each field in the selected field layout.

6.3.1.6 General

These options apply to all number coding types.

Explicit Plus Sign

Setting this option will display positive values prefixed with a "+".

Digit Justification

This option determines how digits are justified within display areas in Word Value display modes.

This option can take one of two settings:

• Right

Digits will be displayed on the right-hand side of display areas.

• Left

Digits will be displayed on the left-hand side of display areas.

• Center

Digits will be displayed in the center of display areas.

6.3.1.7 Error Messages

The following error messages may be displayed after the “Apply” or “OK” buttons are used to indicate that the control settings would result in an invalid set of options. The current option settings will remain unchanged.

<u>Invalid Number Base</u>

The Integer “Base” control contains a value outside the valid range of 2 to 16.

<u>Invalid Exponent Limit</u>

Either

> The Fixed/Floating-point Decimal Negative Exponent Limit control contains a value outside the valid range of -8 to -1

or

> The Fixed/Floating-point Decimal Positive Exponent Limit control contains a value outside the valid range of 1 to 4950

or

> The Fixed/Floating-point Binary Negative Exponent Limit control contains a value outside the valid range of -8 to -1

or

> The Fixed/Floating-point Binary Positive Exponent Limit control contains a value outside the valid range of 1 to 16400

<u>Invalid Number of Displayed Decimal Digits</u>

The Fixed/Floating-point Num Disp Decimal Digits control contains a value outside the valid range of 1 to 900

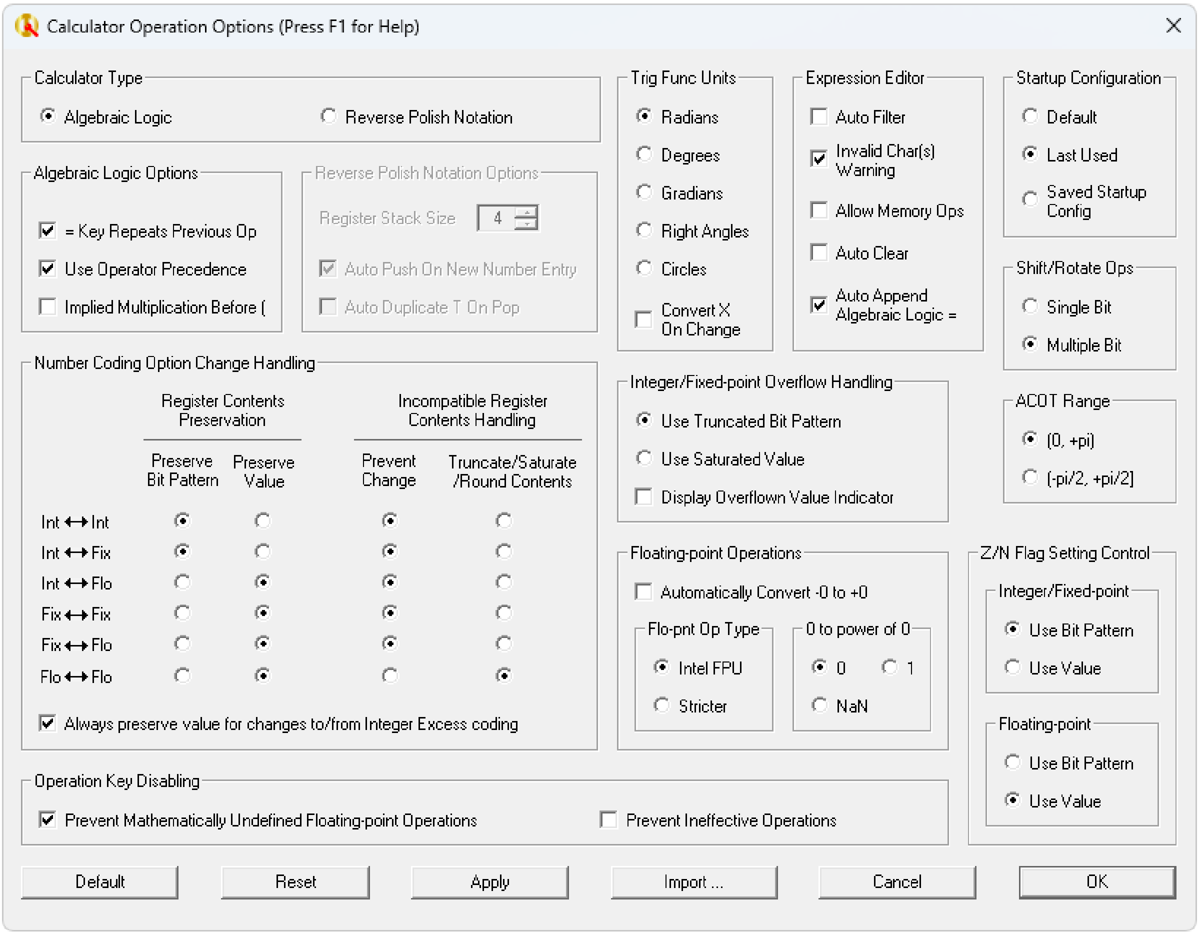
Invalid Number of Displayed Decimal Places

The Fixed/Floating-point Num Disp Decimal Places control contains a value outside the valid range of 0 to 32

6.4 Calculator Operation Options

These options control how IO operates when performing its various operations.

6.4.1 Option Set Window



Calculator Operation Options (Press F1 for Help)

Calculator Type: [✓ Algebraic Logic] Reverse Polish Notation

Algebraic Logic Options:
- [✓ = Key Repeats Previous Op]
- [✓ Use Operator Precedence]
- Implied Multiplication Before (

Reverse Polish Notation Options:
- Register Stack Size: 4
- [✓ Auto Push On New Number Entry]
- Auto Duplicate T On Pop

Trig Func Units: [✓ Radians] Degrees Gradians Right Angles Circles; Convert X On Change

Expression Editor:
- Auto Filter
- [✓ Invalid Char(s) Warning]
- Allow Memory Ops
- Auto Clear
- [✓ Auto Append Algebraic Logic =]

Startup Configuration: Default [✓ Last Used] Saved Startup Config

Shift/Rotate Ops: Single Bit [✓ Multiple Bit]

Number Coding Option Change Handling:

	Register Contents Preservation		Incompatible Register Contents Handling	
	Preserve Bit Pattern	Preserve Value	Prevent Change	Truncate/Saturate /Round Contents
Int ⟷ Int	●	○	●	○
Int ⟷ Fix	●	○	●	○
Int ⟷ Flo	○	●	●	○
Fix ⟷ Fix	○	●	●	○
Fix ⟷ Flo	○	●	●	○
Flo ⟷ Flo	○	●	○	●

[✓ Always preserve value for changes to/from Integer Excess coding]

Integer/Fixed-point Overflow Handling: [✓ Use Truncated Bit Pattern] Use Saturated Value; Display Overflown Value Indicator

ACOT Range: [✓ (0, +pi)] (-pi/2, +pi/2]

Floating-point Operations:
- Automatically Convert -0 to +0
- Flo-pnt Op Type: [✓ Intel FPU] Stricter
- 0 to power of 0: [✓ 0] 1 NaN

Z/N Flag Setting Control:
- Integer/Fixed-point: [✓ Use Bit Pattern] Use Value
- Floating-point: Use Bit Pattern [✓ Use Value]

Operation Key Disabling: [✓ Prevent Mathematically Undefined Floating-point Operations] Prevent Ineffective Operations

The calculator operation option set window may be displayed by selecting the “Calculator Operation …” item from the Options menu, or by shift-double-clicking on any of: the flags status display, the trigonometric units status display, the integer/fixed-point overflow handling status display, or the number coding option change handling status display.

6.4.1.1 Calculator Type

This option determines when IO processes operators in response to user input.

Calculator Type

This option can take one of two setting:

• Algebraic Logic

Uses Algebraic Logic to perform calculations, using brackets entered by the user and (optionally) operator precedence to determine the order in which operators are applied.

• Reverse Polish Notation

Uses Reverse Polish Notation to perform calculations; the user has complete control over when operators are applied, and of the stack of values/bit patterns to which they are applied.

6.4.1.2 Algebraic Logic Options

These options control aspects of IO operation when a Calculator Type of Algebraic Logic is selected.

= Key Repeats Previous Op

Setting this option causes clicking on the = key when there are no pending operations to repeat the previous two-operand operation (if there is one) using the current X register contents as the first operand and the same second operand as the previous operation. The previous operation is displayed in the Stack Window.

Use Operator Precedence

Setting this option causes operators to be applied in an order dependent on their relative precedence. This order can be overridden by using brackets. Consecutive operators of the same precedence, without any intervening brackets, will be applied in the order they are entered.

If this option is not set then operators are applied in the order they are entered by the user, though this can also be overridden by using brackets.

The following tables show the relative operator precedences for each number coding type:

<u>Integer Number Coding Type</u>

Precedence	Operator Keys
Highest	Perm $^{y}P_{x}$ $_{y}P_{x}$ Comb $^{Y}C_{x}$ $_{y}C_{x}$ $\binom{y}{x}$ HCF (x,y) LCM [x,y]
	× ∗ ÷ / mod %
	+ − ADDC +C SUBC SUBB −C
	SHL ⇤ << ASHR ⇥ >> LSHR ⇥ >>> ROL ⇆ ⟳ ROR ⇆ ⟳ (Multiple bit shift/rotate operations only)
	AND & ⊐D− NAND ~& ⊐D○− BIC &~ ⊐D−

Precedence	Operator Keys
	XOR EOR ^ XNOR NXOR ENOR NEOR ~^
Lowest	OR | NOR ~|

Fixed-Point Number Coding Type

Precedence	Operator Keys
Highest	y^x **
	× * ÷ /
	+ −
	SHL << ASHR >> LSHR >>> ROL ⟳ ROR ⟳ (Multiple bit shift/rotate operations only)
	AND & NAND ~& BIC &~ (Fixed-point Word Bit Pattern display mode only)
	XOR EOR ^ XNOR NXOR ENOR NEOR ~^ (Fixed-point Word Bit Pattern display mode only)
Lowest	OR | NOR ~| (Fixed-point Word Bit Pattern display mode only)

Floating-Point Number Coding Type

Precedence	Operator Keys
Highest	LOG_xY
	y^x ** $\sqrt[x]{y}$ $y^{1/x}$ ROOT
	× * ÷ /
Lowest	+ −

Implied Multiplication Before (

Setting this option causes a multiplication operator to automatically be requested before an opening bracket is entered if no operator has been explicitly entered.

6.4.1.3 Reverse Polish Notation Options

These options control aspects of IO operation when a Calculator Type of Reverse Polish Notation is selected.

Register Stack Size

Sets the size of the register stack. This must be in the range four to twenty.

Auto Push on New Number Entry

Setting this option will cause the register stack contents to automatically be pushed up when the entry of a new number is started. If not then a newly entered number will overwrite the value/bit pattern at the bottom of the stack.

Auto Duplicate T on Pop

Setting this option will cause the value/bit pattern originally stored at the top of the register stack to be duplicated in that location after a pop operation.

6.4.1.4 Number Coding Option Change Handling

These options control how IO modifies register contents when a change is made to the number coding options.

Register Contents Preservation

There are six of these options:

Int↔Int, Int↔Fix, Int↔Flo, Fix↔Fix, Fix↔Flo, Flo↔Flo

These options determine what register contents are preserved when a change between the indicated number coding types is made. Each can each take one of two settings:

• Preserve Bit Pattern

The bit pattern stored in a register will remain unchanged and the value it represents will change in accordance with the newly selected coding.

• Preserve Value

The bit pattern stored in a register will be modified so that the value it represents remains unchanged for the newly selected coding.

Incompatible Register Contents Handling

There are six of these options:

Int↔Int, Int↔Fix, Int↔Flo, Fix↔Fix, Fix↔Flo, Flo↔Flo

These options determine what happens when the contents of a register cannot be preserved as desired by the corresponding Register Contents Preservation option, i.e., the register contents are incompatible with the required preservation type. Each can take one of two settings

• Prevent Change

Keys which would select Number Coding options which are not compatible with all current register contents will be disabled. An attempt to select such Number Coding options using the Number Coding options window or a Number Coding popup window will display an error message and the change will not be made.

• Truncate/Saturate/Round Contents

Incompatible register content will be truncated, saturated or rounded as described below, though some incompatible content will still prevent a change, e.g., a negative value will not be converted to an unsigned or binary coded decimal coding, a NaN will not be converted to a non-floating-point coding.

If "Preserve Bit Pattern” is selected, the original word width is wider than the target word width, and a register contains a bit pattern with a bit set to ① which is to the left of the bit which corresponds to the most significant bit of the target word, then the bit pattern will be truncated, i.e., all bits to the left of the bit which corresponds to the most significant bit of the target word will be lost.

If “Preserve Value” is selected and the value originally stored in a register cannot be represented exactly using the target number coding, then the value indicated in the following table will be written to the register.

	To:	Integer		Fixed-Point		Floating-Point
From	Original Value	Unsigned BCD	2s Compl Excess	Unsigned	2s Compl	
Integer	> MAX	MAX	MAX	MAX	MAX	+INF
	< MIN	Prevent	MIN	Prevent	MIN	-INF
	Not representable	N/A	N/A	FP Round	FP Round	FP Round
Fixed-Point	> MAX	MAX	MAX	MAX	MAX	+INF
	< MIN	Prevent	MIN	Prevent	MIN	-INF
	Not representable	Int Round	Int Round	FP Round	FP Round	FP Round
Floating-Point	QNaN SNaN	Prevent	Prevent	Prevent	Prevent	QNaN SNaN
	+INF	MAX	MAX	MAX	MAX	+INF
	-INF	Prevent	MIN	Prevent	MIN	-INF
	> MAX	MAX	MAX	MAX	MAX	+INF
	< MIN	Prevent	MIN	Prevent	MIN	-INF
	Not representable	Int Round	Int Round	FP Round	FP Round	FP Round

The following terms are used in the table:

> “MAX” denotes the highest value that can be represented using the target coding, which will always be positive.

“MIN” denotes the lowest value that can be represented using the target coding. For unsigned and binary coded decimal codings this will be zero, otherwise, it will be negative.

“Not representable” denotes an original value that lies between the upper and lower value limits for the target coding, but still cannot be represented exactly using the target coding.

“Prevent” denotes that IO will prevent attempts to select the target coding, as described above.

“N/A” denotes a condition that cannot occur.

“FP Round” denotes the value obtained by applying the “Fixed/Floating-Point Rounding Type” selected in the target number coding options to the original value.

“Int Round” denotes the value obtained by applying the To Nearest (Away From Zero) rounding type to the original value if the “Always use To Nearest (Away From Zero) for Round To Integer Operations” number coding option is set for the target number coding, otherwise, the same value as for “FP Round” is used.

See the chapter on Floating-Point Codings for details of “+INF”, “-INF”, “QNaN” and “SNaN”.

Always preserve value for changes to/from Integer Excess coding

Setting this option causes the bit pattern stored in a register to be modified so that the value it represents remains unchanged if a change to/from Integer Excess coding is made, irrespective of the above settings. This can avoid undesirable behavior when “Preserve Bit Pattern” is selected due to the fact that a bit pattern of all ⓪s does not represent a value of zero in Integer Excess coding.

6.4.1.5 Trigonometric Function Units

These options determine what units the arguments of the trigonometric functions are interpreted as, and in what units the results of the inverse trigonometric functions are measured.

Trigonometric Function Units

This option can take one of five settings:

• Radians

Selects radians; there are 2π radians in a circle. An arc of a circle which has a length equal to the radius subtends an angle of one radian at the center of the circle.

• Degrees

Selects degrees; there are 360 degrees in a circle. The notation x° is used to denote an angle of x degrees. One degree can be subdivided into 60 “minutes”, and one minute can be further subdivided into 60 “seconds”. In this context, the terms minutes and seconds are sometimes followed by the words “of arc” to distinguish them from the units of time with the same names.

• Gradians

Selects gradians; there are 400 gradians in a circle. A gradian is also referred to as a grade, grad, or gon. Very uncommonly you may see a gradian referred to as a gradient.

• Right angles

Selects right angles; there are 4 right angles in a circle

• Circles

Selects circles. Circles, in this context, are sometimes referred to as “turns”.

Convert X On Change

Setting this option will cause the contents of the X register to be converted from the previous units to newly selected units. Only the X register is converted, no others are modified.

6.4.1.6 Expression Editor

These options control the operation of the expression editor.

Auto Filter

Setting this option causes the filtering of invalid characters as they are entered; otherwise, they are filtered just before the expression is executed.

Invalid Char(s) Warning

Setting this option causes a warning message to be displayed if an attempt is made to execute an expression which contains any invalid characters. You will be given the options of filtering out the invalid characters and then executing the expression, or cancelling the execution.

Allow Memory Ops

Setting this option allows S and R commands to be used in the expression to store/recall to/from the current memory during expression evaluation.

Auto Clear

Setting this option causes the expression string to be automatically cleared following execution.

Auto Append Algebraic Logic =

Setting this option causes a "=" character to be automatically appended to the expression string just before execution. This will only be done if Algebraic Logic is selected for the Calculator Type option and the last character in the expression is not already a "=".

6.4.1.7 Integer/Fixed-Point Overflow Handling

These options control how IO deals with overflow conditions generated by integer and fixed-point operations. Note that these options are not used when changing number coding options.

Integer/Fixed-point Overflow Handling

This option can take one of two settings:

• Use Truncated Bit Pattern

This causes the X register to be set to the truncated bit pattern of the result. This is the usual behavior for general purpose CPUs.

• Use Saturated Value

This causes the X register to be set to the largest (positive or negative) value for the current coding. This is optional behavior for many digital signal processing CPUs.

Note that the setting of this option does not affect the “Lowest Common Multiple”, “Permutations of X from Y” and “Combinations of X from Y” integer operations, these always return a saturated result in the case of overflow.

Display indicator on Overflow

Setting this option causes "Overflow" to be displayed in the main number display area when an overflow condition is generated. This will be more difficult to miss than just the setting of the V and/or O flags. The indicator can be removed by clicking the edit key to reveal the resulting value/bit pattern.

6.4.1.8 Floating-Point Operations

These options control how IO behaves for certain floating-point operations.

Automatically convert -0 to +0

Setting this option causes any -0 bit pattern generated to be automatically converted to the +0 bit pattern.

Flo-pnt Op Type

This option determines how certain floating-point operations are performed.

This option can take one of three settings:

• Intel FPU

Operations will be performed as by an Intel Floating-point Unit.

• Stricter

Operations will be performed in a stricter, more consistent manner.

0 to power of 0

This option allows you to specify what should result if this calculation is attempted.

This option can take one of three settings:

• 0

This causes a result value of 0 to be generated.

• 1

This causes a result value of 1 to be generated.

• NaN

This causes a NaN bit pattern to be generated.

6.4.1.9 Startup Configuration

This option identifies what configuration IO uses the next time it is started.

Startup Configuration

This option can take one of three settings:

• Default

IO will start up using the default configuration.

• Last Used

IO will start up using the configuration in use the last time that IO terminated cleanly.

• Saved Startup Config

IO will start up using the configuration saved the last time the "File" menu "Save Startup Config" item was used.

6.4.1.10 Shift/Rotate Operations

This option sets whether the shift and rotate operations are processed as functions which shift/rotate the X register contents by one bit or as operators which shift/rotate by a specified number of bits. The rotate-through-carry operations are not affected by this option, they will always rotate by a single bit.

Shift/Rotate Ops

This option can take one of two settings:

• Single Bit

This causes the X register to be shifted or rotated by a single bit as soon as the operation key is clicked.

• Multiple Bit

This causes the X register to be set to the original contents of the Y register shifted/rotated by the number of bits identified by the value originally contained in the X register. If the X register originally contained a negative value, then no shift/rotation is performed.

6.4.1.11 ACOT Range

This option allows you to specify the range of results returned by the arccotangent function. See the description of the ACOT key for a fuller discussion of this issue.

(0, +pi)

This causes result values > 0 and < π to be returned. The arccotangent function is continuous when this range is used.

(-pi/2, +pi/2]

This causes result values > -(π /2) and <= (π /2) to be returned. The arccotangent function has a discontinuity at zero when this range is used; however, this range is the same as for the arctangent function.

6.4.1.12 Z/N Flag Setting Control

These options control how the settings for the Zero flag (Z) and Negative flag (N) are determined by the contents of the X register.

Integer/Fixed-point

This option controls how the settings are determined for integer and fixed-point number coding types.

This option can take one of two settings:

• Use Bit Pattern

The Z flag is set if, and only if, all the bits in the bit pattern are set to ⓪.

The N flag is set if, and only if, the most significant bit the bit pattern is set to ①.

• Use Value

The Z flag is set if, and only if, the value represented by the bit pattern is 0.

The N flag is set if, and only if, the value represented by the bit pattern is negative.

Floating-point

This option controls how the settings are determined for the floating-point number coding type.

This option can take one of two settings:

• Use Bit Pattern

The Z flag is set if, and only if, all the bits in the bit pattern are set to ⓪.

The N flag is set if, and only if, the most significant bit the bit pattern is set to ①.

• Use Value

The Z flag is set if, and only if, the value represented by the bit pattern is +0 or -0.

The N flag is set if, and only if, the value represented by the bit pattern is negative.

6.4.1.13 Operation Key Disabling

These options control the disabling of keys in addition to those which are illegal.

Prevent Mathematically Undefined Floating-point Operations

Setting this option causes the disabling of keys which would produce a mathematically undefined result for floating-point codings. If these keys are not disabled and used then they will generate a bit pattern with a class of NaN or Infinity.

Prevent Ineffective Operations

Setting this option causes the disabling of keys which would not modify the contents of any register or affect a stack.

6.4.1.14 Error Messages

The following error messages may be displayed after the “Apply” or “OK” buttons are used to indicate that the control settings would result in an invalid set of options. The current option settings will remain unchanged.

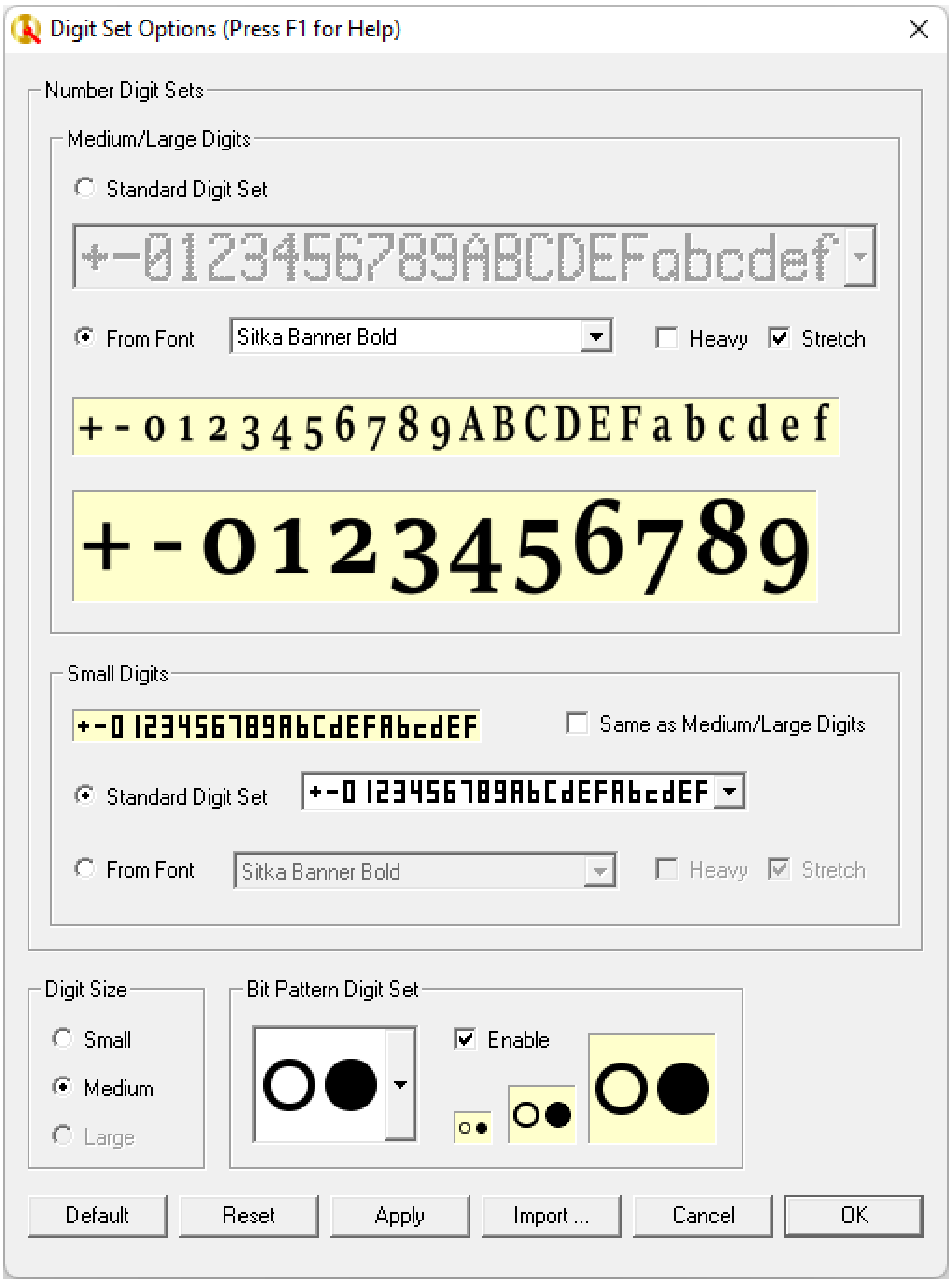
<u>Invalid Register Stack Size</u>

The Register Stack Size control contains a value outside the valid range of 4 to 20.

6.5 Digit Set Options

These options control the digit sets used by IO to display bit patterns stored in registers and the values they represent.

6.5.1 Option Set Window



Medium/Large Digits: Standard Digit Set [✓ From Font] Sitka Banner Bold; Heavy ☐; Stretch ☑

Small Digits: Same as Medium/Large Digits ☐; [✓ Standard Digit Set] From Font

Digit Size: Small [✓ Medium] Large

Bit Pattern Digit Set: Enable ☑

The digit set option set window may be displayed by selecting the “Digit Set …” item from the Options menu.

6.5.1.1 Number Digit Sets

Medium/Large Digits

Medium/Large Digit Set Source

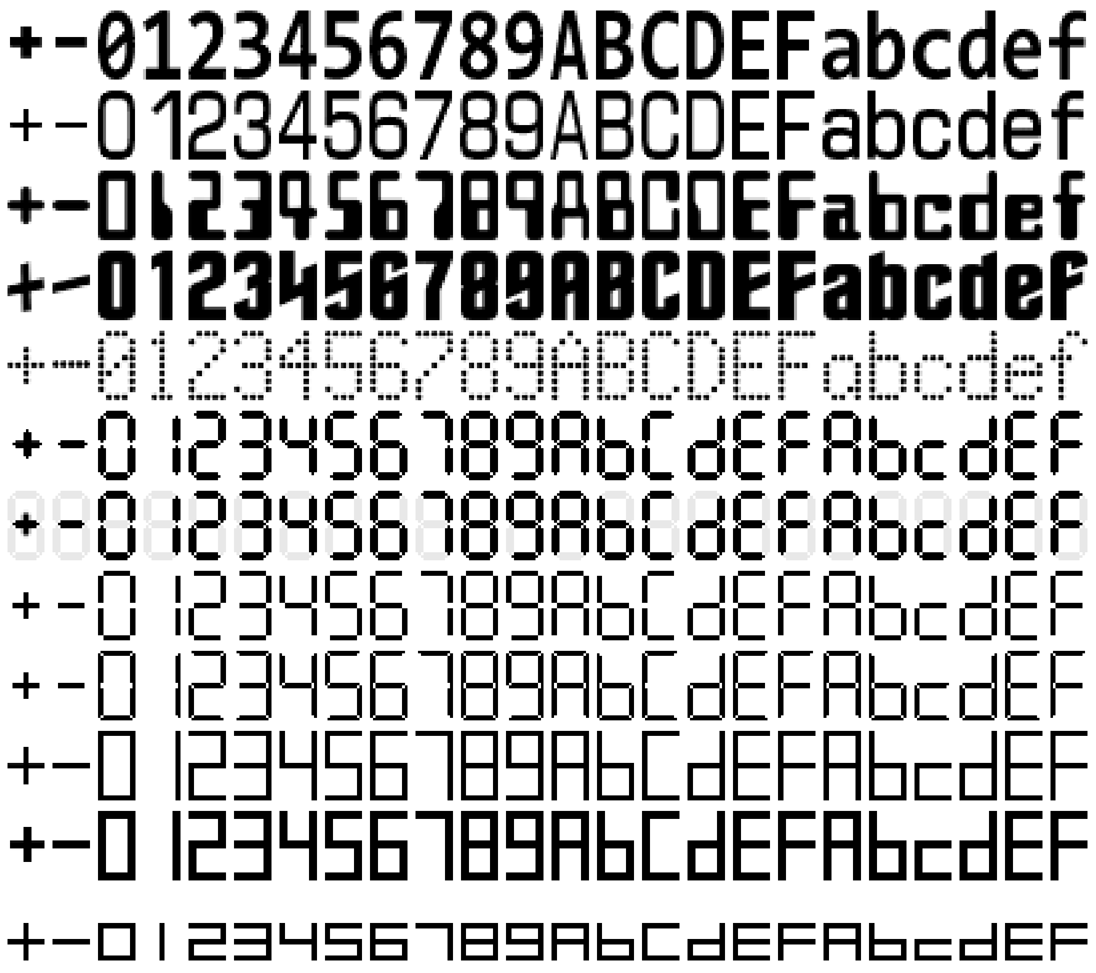
This option can take one of two settings:

• Standard Digit Set

Selects the use of one of IO's standard medium/large digit sets. The associated combo box will be enabled.

The combo box may be used to select one of IO's standard medium/large digit sets from the dropdown list displayed by clicking on the combo box. The combo box will display the selected standard medium/large digit set as black digits on a white background.

The following standard medium/large digit sets are available:



• From Font

Selects the use of a medium/large digit set based on one of the fonts installed on your computer. The associated combo box and checkboxes will be enabled.

This combo box may be used to select one of the fonts installed on your computer in order to generate a medium/large digit set based on it. See later for details of which fonts will be made available for selection.

Heavy

Setting this option will generate a heavier version of the selected font. This button will be disabled if the selected font is the heaviest available for the typeface.

Stretch

Setting this option will stretch each digit by the same independent horizontal and vertical factors so that the widest digit has the maximum width for a medium/large digit, and the tallest digit has the maximum height for a medium/large digit.

Selected Medium/Large Digit Set Digit Displays

The first display shows the currently selected medium digit set in the digit and digit background colors currently selected in the Color Options. Sign and hexadecimal digits (uppercase and lowercase) are displayed.

The second display shows the currently selected large digit set in the digit and digit background colors currently selected in the Color Options. Only the sign and decimal digits are displayed.

"Problem with Font" will be displayed here if the font selected in the Medium/Large Digits font selection combo box control could not be used to generate a digit set for some reason, probably because the font would not generate a valid character bitmap for a required scaling due to it not being implemented well enough.

Small Digits

Same as Medium/Large Digits

Ticking this checkbox causes all the settings for the medium/large digits to also be used for small digits. All the other small digit set controls will be disabled.

Selected Small Digit Set Digit Display

Displays the currently selected small digit set in the digit and digit background colors currently selected in the Color Options.

"Problem with Font" will be displayed here if the font selected in the Small Digits font selection combo box control could not be used to generate a digit set for some reason, probably because the font would not generate a valid character bitmap for a required scaling due to it not being implemented well enough.

Small Digit Set Source

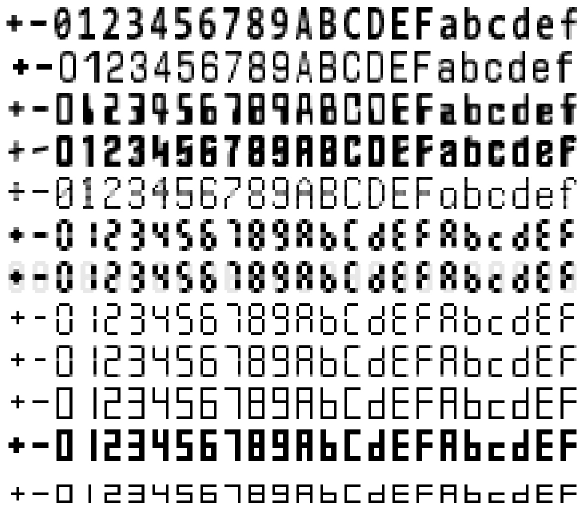
This option can take one of two settings:

• Standard Digit Set

Selects the use of one of IO's standard small digit sets. The associated combo box will be enabled.

The combo box may be used to select one of IO's standard small digit sets from the dropdown list displayed by clicking on the combo box. The combo box will display the selected standard small digit set as black digits on a white background.

The following standard small digit sets are available:



• From Font

Selects the use of a small digit set based on one of the fonts installed on your computer. The associated combo box and checkboxes will be enabled.

This combo box may be used to select one of the fonts installed on your computer in order to generate a small digit set based on it. See later for details of which fonts will be made available for selection.

Heavy

Setting this option will generate a heavier version of the selected font. This button will be disabled if the selected font is the heaviest available for the typeface.

Stretch

Setting this option will stretch each digit by the same independent horizontal and vertical factors so that the widest digit has the maximum width for a small digit, and the tallest digit has the maximum height for a small digit.

6.5.1.2 Digit Size

Size of displayed Digits

This option identifies the size of digits used in the main number displays of the keypad and detached display windows.

This option can take one of three settings

• Small

Small digits will be displayed

• Medium

Medium digits will be displayed

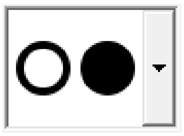
• Large

Large digits will be displayed. This may only be selected for the detached display window.

6.5.1.3 Bit Pattern Digit Set

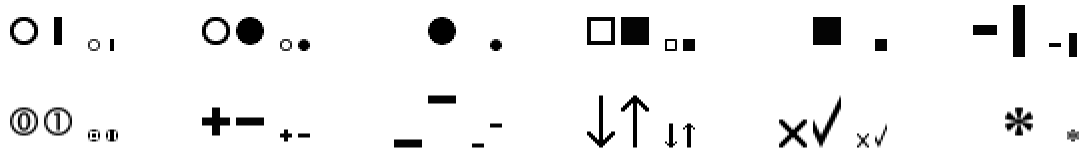
Enable

Setting this option will enable the use of the selected bit pattern digit sets in Bit Pattern Display Mode; otherwise, the 0 and 1 digits of the selected number digit sets will be used. The associated combo box will be enabled.



This combo box may be used to select one of the bit pattern digit sets from the dropdown list displayed by clicking on it. Any selection made will apply to all digit set sizes. The combo box will display the selected bit pattern digit sets as black digits on a white background.

The following bit pattern digit sets are available:



Selected Bit Pattern Digit Set Displays

These display the currently selected bit pattern digit sets in the digit and digit background colors currently selected in the Color Options.

6.5.1.4 Error Messages

The following error messages may be displayed after the “Apply” or “OK” buttons are used to indicate that the control settings would result in an invalid set of options. The current option settings will remain unchanged.

Unable to use selected Font for Medium/Large Digit Set

The font selected in the Medium/Large Digits font selection combo box control could not be used to generate a digit set for some reason, probably because the font would not generate a valid character bitmap for a required scaling due to it not being implemented well enough.

Unable to use selected Font for Small Digit Set

The font selected in the Small Digits font selection combo box control could not be used to generate a digit set for some reason, probably because the font would not generate a valid character bitmap for a required scaling due to it not being implemented well enough.

6.5.2 “From Font” Digit Sets

The font(s) selected by the combo box controls are the only IO options that are not completely defined within IO, but are largely effected by the implementations of the fonts as created by their developers. Consequently, some issues can arise which are beyond IO’s control.

If a configuration is loaded from a file, typically at startup, and a font specified in the Digit Set options of the configuration is not installed then the default Digit Set options will be used.

If the “Import” button is used to load a set of Digit Set options from a file and those options specify Font(s) which are not installed then the combo box controls, which would normally be initialized to select those fonts, will be initialized to select the first font(s) in the list.

It is very unlikely that all the fonts installed on your computer will be made available for selection using the combo box controls. Only fonts which satisfy all of the following conditions will be made available:

The font is a TrueType font

The font contains an ASCII character set

The font is not designed to generate vertical text

The font claims to contain definitions for all of the following characters:
+ - 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f . , ‘

Implementing a TrueType font is a very involved process and some developers do not take as much care as others to get all the details right, this is especially true of free fonts. Some of these shortcomings can cause IO problems when it attempts to generate a digit set from such a font. Some of these issues are:

The font does not generate correct bitmaps for all scales required to produce the digit set, this results in the “Problem with Font” displays and error messages described above.

The font claims to contain definitions for characters that it does not draw as would be expected, e.g., the characters appear as rectangles. A relatively common example of this issue is when a font draws the “+” character as something that looks nothing like a “+”.

Even with a well implemented font there are some issues that can make a font an unsatisfactory candidate for generating a digit set. Specifically:

Some fonts, especially decorative ones intended to be rendered at larger sizes than usual, e.g., for use on posters, can contain fine details which will get lost when “shrunk” to the size of the digit cells used by IO. This is especially true for small digit sets; hence the option to make independent selections for the small and medium/large digit sets.

The “+” and “–” characters are handled differently from the others in the following ways:

They are aligned vertically so that their vertical centers are aligned with the vertical center of the “0” digit, irrespective of their vertical positions defined in the font. This results in a consistent appearance in the number displays. Many fonts position the “–“ character to align with the centers of most lowercase alphabetic characters so that it looks good when used as a hyphen, this is usually lower than ideal when used as a minus sign.

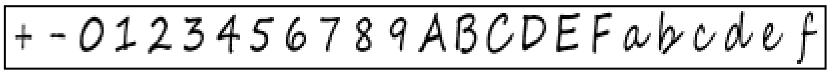
They are not stretched when the stretch option is selected. This prevents distortion of the relative lengths of the horizontal and vertical lines in the “+” character, which are usually equal in length.

Although they are not shown in the digit set displays, the dot, comma and single-quote characters in the font are used to generate the various types of decimal/binary points and digit group separators, as selected in the Number Display options. A centered decimal/binary point is generated using the dot character from the font, so the font does not need to contain a centered dot character.

The digit sets generated and used by IO will vary depending upon the current combination of a number of option settings and display mode so as to use as much of the digit cells as possible while being able to correctly display all possibly required characters. This will involve different scalings being applied to the digits for different combinations, but IO will always preserve the vertical alignments of the characters as they are defined in the font.

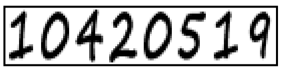
That was probably as clear as mud, so here are some examples to demonstrate this feature.

Suppose the following digit set digits are generated from the selected font:

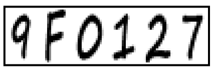


Notice that the uppercase alphabetic digits are a bit taller than the decimal digits, and that the lowercase “f” descends below the baseline of the other digits.

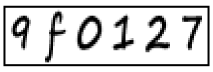
Now compare the following displays of the same value for different number display option settings:



Decimal



Hex Uppercase



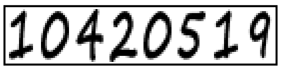
Hex Lowercase

Notice:

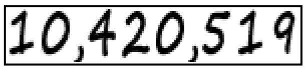
The decimal digits in the second display are a bit shorter than in the first in order to accommodate the extra height required by the “F” digit.

The decimal digits in the third display are shorter still in order to accommodate the extra height required by the “f” digit and to maintain the required vertical alignment of all the digits.

Similarly;


10420519

No Digit Group Separators


10,420,519

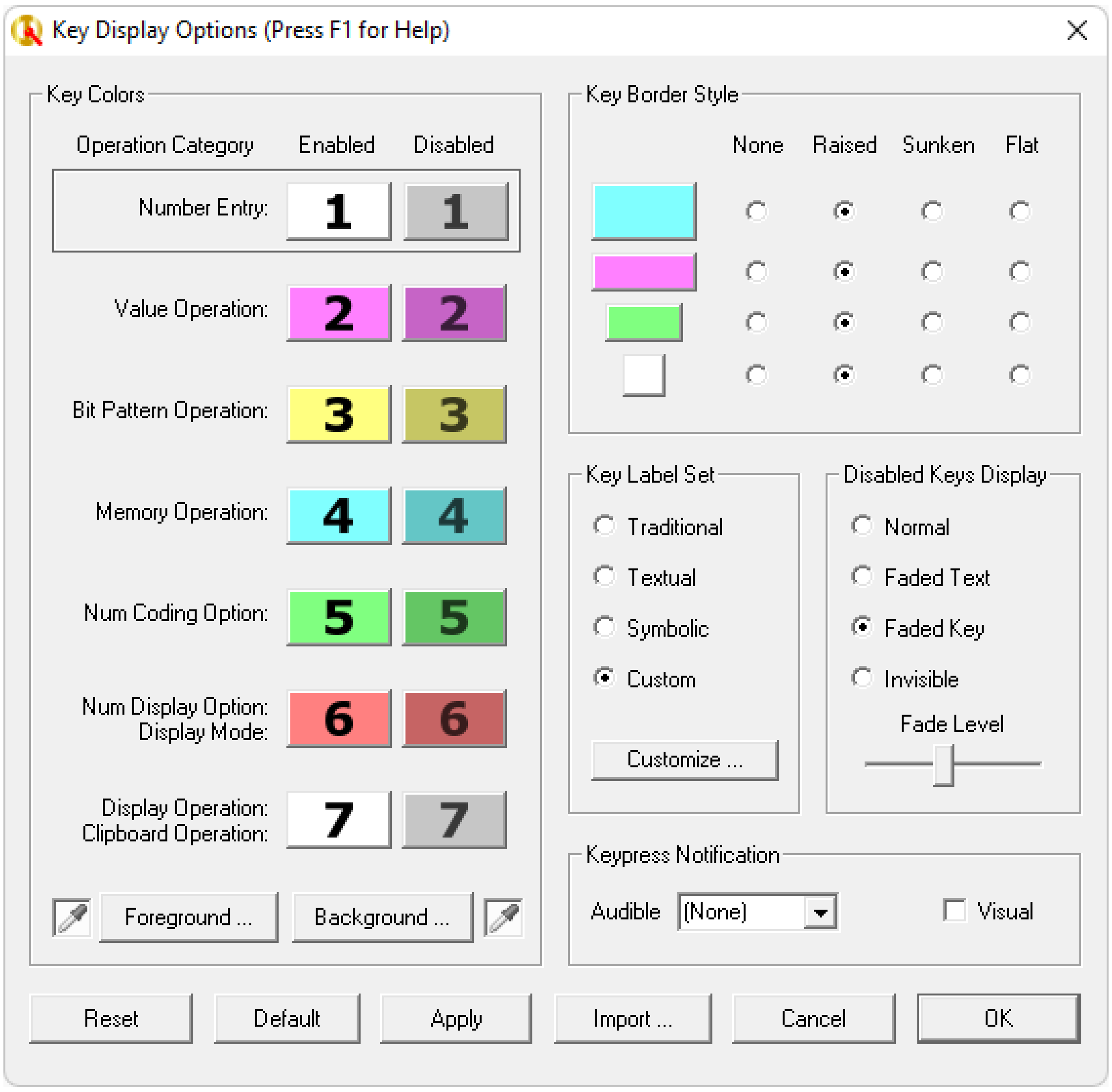
Comma Digit Group Separators

Notice that the digits in the second display are a bit shorter than in the first in order to accommodate the extra height required by the comma characters while maintaining the required vertical alignment of all the characters.

6.6 Key Display Options

These options control how IO displays its keys.

6.6.1 Option Set Window


Key Display Options (Press F1 for Help)

Key Colors

Operation Category	Enabled	Disabled
Number Entry:	1	1
Value Operation:	2	2
Bit Pattern Operation:	3	3
Memory Operation:	4	4
Num Coding Option:	5	5
Num Display Option: Display Mode:	6	6
Display Operation: Clipboard Operation:	7	7

Key Border Style

	None	Raised	Sunken	Flat
🟦	○	●	○	○
🟪	○	●	○	○
🟩	○	●	○	○
⬜	○	●	○	○

Key Label Set: Traditional [✓ Custom] (Textual, Symbolic)

Disabled Keys Display: Normal, Faded Text, [✓ Faded Key], Invisible

Fade Level

Keypress Notification

Audible (None)

☐ Visual

The key display option set window may be displayed by selecting the “Key Display …” item from the Options menu.

It can take some time to set up all the key colors and/or custom key label set selections to what you want if you feel so inclined. In which case, it is a good idea to save a configuration containing your settings to a file, then, if for any reason you lose these settings in your configuration, e.g., by loading the default configuration, you can quickly reinstate your preferred Key Display options by using the “Import …” button to load them from your saved configuration file.

6.6.1.1 Key Colors

These options define the foreground and background colors for each of the seven key categories. Each key category is identified by its name to the left of two displays, the first illustrates the currently selected foreground and background color for the key category and the second illustrates how a disabled key in the category will be displayed.

At least one, and possibly more, key categories will be selected at any time. The currently selected key category(ies) will be identified by enclosing frames.

The mouse is used to modify the current selection by clicking on the name of a category or one of the displays associated with it as follows:

Left Click

The clicked category will be selected and any other currently selected categories will be deselected.

Ctrl + Left Click

If the clicked category was not currently selected then it is added the current selection.

If the clicked category is currently selected then it is removed from the current selection provided it is not the only category selected, in which case nothing changes.

Note that the displayed example keys for each category are all of the large size, but a key category does not determine the size of all keys in the category.

The following key categories are identified:

Number Entry:	This key category comprises keys used to enter numerical values
Value Operation:	This key category comprises keys used to perform operations using the values represented by the bit patterns stored in registers
Bit Pattern Operation:	This key category comprises keys used to perform operations using the bit patterns stored in registers
Memory Operation:	This key category comprises keys used for operations involving the memory registers
Num Coding Option:	This key category comprises keys used to change the Number Coding Options
Num Display Option: Display Mode:	This key category comprises keys used to change the Number Display Options or the Display Mode
Display Operation: Clipboard Operation:	This key category comprises keys located near number display areas

Some keys can be considered to fall into more than one category, the category to which IO defines a key to belong to can be deduced from the colors used to display it.

Displays a color selection window to allow the selection of a new foreground color for the currently selected key category(ies).

Background ...

Displays a color selection window to allow the selection of a new background color for the currently selected key category(ies).

Changes the mouse pointer to an eyedropper shape, the next time this is clicked/released (almost) anywhere on the screen the color of the pixel clicked upon will become the new foreground/background color for the currently selected key category(ies).

See the description of the similar button in the color options set window for a detailed description of the use of this feature.

6.6.1.2 Key Border Style

This option determines the appearance of the border around the key labels.

The style of border can be specified independently for each of the four key sizes.

Key Border Style

The option for each of the four key sizes can take one of four settings:

• None

No borders are displayed.

• Raised

3D borders which make keys look raised are displayed.

• Sunken

3D borders which make keys look sunken are displayed.

• Flat

Simple 2D borders are displayed.

The selected styles for each key size are illustrated in a display of that size to the left of the corresponding radio buttons

The selected style for large keys is used for the displays in the Key Colors area.

The borders use standard Windows colors, so how clearly they appear will be affected by the configured background colors of the labels and parent window.

6.6.1.3 Key Label Set

This option determines the set of key labels that will be used to identify the operations performed by keys.

Key Label Set

This option can take one of four settings:

• Traditional

This will use the default set, which is a collection of labels traditionally used on calculators.

• Textual

This will use a set of labels which are primarily textual in appearance.

• Symbolic

This will use a set of labels which are primarily symbolic in appearance.

• Custom

This will allow you to independently select the label used for each key from a set of available labels specific for each operation associated with a key. The Customize button will be enabled.

If the Custom option is selected then clicking this button will open a window which will allow you to change the key label to be displayed for each associated operation.

6.6.1.4 Disabled Key Display

These options define how disabled keys are displayed.

Disabled Key Display

This option can take one of four settings:

• Normal

Disabled keys are displayed using the foreground and background colors for the key's category. The key borders may look slightly different from those of enabled keys.

• Faded Text

Disabled keys are displayed using a faded variant of the foreground text color for the key's category. The Fade Level slider will be enabled.

• Faded Key

Disabled keys are displayed using faded variants of the foreground and background colors for the key's category. The Fade Level slider will be enabled.

• Invisible

Disabled keys will be invisible.

Fade Level

This slider can be used to change the degree of fade applied to faded keys.

6.6.1.5 Keypress Notification

These options define how the user is notified of an operation key being pressed.

Audible

This combo box selects the sound, if any, to be played when the user clicks on an enabled key, or presses a corresponding keyboard key.

▶

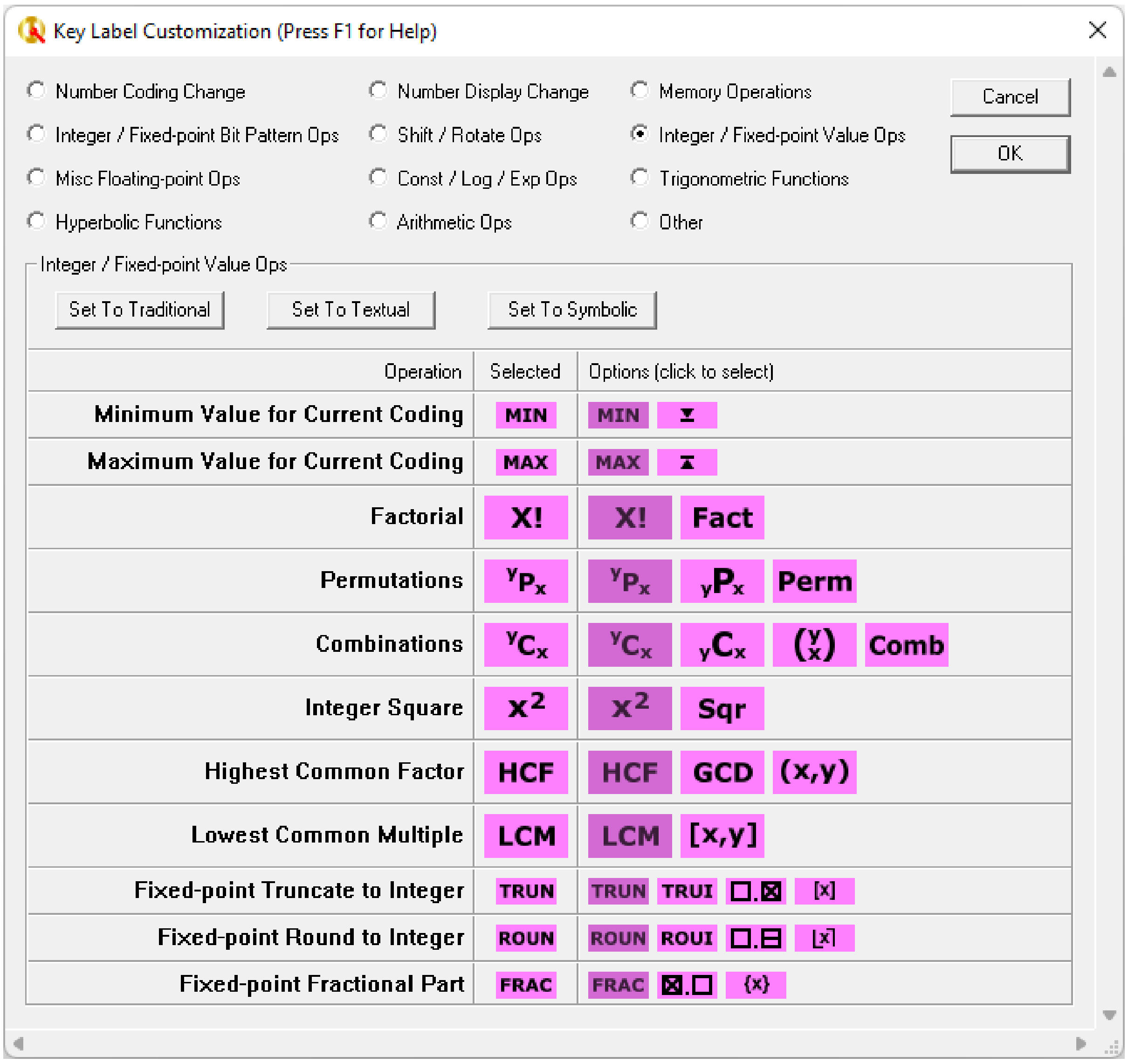
Plays the sound currently selected in the combo box.

Visual

Setting this option will cause the label of an enabled key to be briefly displayed with inverted colors when it is clicked, or the corresponding keyboard key is pressed.

6.6.2 Key Label Customization Window

This window allows the selection of the custom set of key labels from all those available.



Key Label Customization (Press F1 for Help)

○ Number Coding Change	○ Number Display Change	○ Memory Operations	Cancel
○ Integer / Fixed-point Bit Pattern Ops	○ Shift / Rotate Ops	[✓ Integer / Fixed-point Value Ops]	OK
○ Misc Floating-point Ops	○ Const / Log / Exp Ops	○ Trigonometric Functions	
○ Hyperbolic Functions	○ Arithmetic Ops	○ Other	

Integer / Fixed-point Value Ops

Set To Traditional | Set To Textual | Set To Symbolic

Operation	Selected	Options (click to select)			
Minimum Value for Current Coding	MIN	MIN	⤓		
Maximum Value for Current Coding	MAX	MAX	⤒		
Factorial	X!	X!	Fact		
Permutations	${}^{y}P_{x}$	${}^{y}P_{x}$	${}_{y}P_{x}$	Perm	
Combinations	${}^{y}C_{x}$	${}^{y}C_{x}$	${}_{y}C_{x}$	$\binom{y}{x}$	Comb
Integer Square	x^2	x^2	Sqr		
Highest Common Factor	HCF	HCF	GCD	(x,y)	
Lowest Common Multiple	LCM	LCM	[x,y]		
Fixed-point Truncate to Integer	TRUN	TRUN	TRUI	□.⊠	[x]
Fixed-point Round to Integer	ROUN	ROUN	ROUI	□.⊟	⌊x⌉
Fixed-point Fractional Part	FRAC	FRAC	⊠.□	{x}	

The key label customization window may be displayed by clicking the “Customize” button in the key display option set window while the “Custom” Key Label Style is selected.

6.6.2.1 Edit Buttons

Cancel

Discards any changes to the current custom set selection and close the window.

OK

Stores any changes to the current custom set selection and close the window.

6.6.2.2 Operation Group

Clicking one of the radio buttons displays the labels for the operation keys in that group in the main area of the window, from where the required label for each key may be selected.

6.6.2.3 Operation Buttons

Selects the traditional labels for all the operation keys in the selected group.

Selects the textual labels for all the operation keys in the selected group.

Selects the symbolic labels for all the operation keys in the selected group.

6.6.2.4 Main Area Columns

Operation

This contains a short description of the operation performed by the key.

Selected

This contains an image of the currently selected key label for the operation. It is displayed in the colors currently selected for its category in the parent Key Display Options window.

Options (click to select)

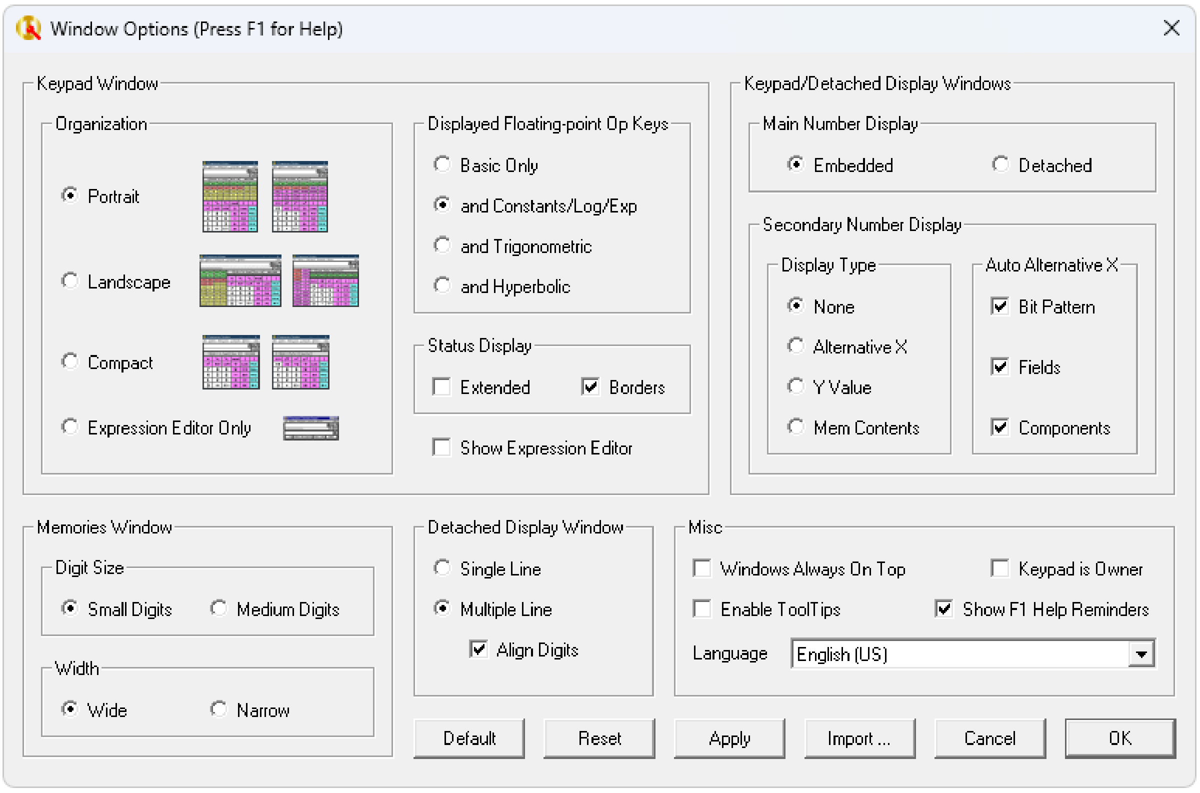
This contains an image of each of the available key labels for the operation. The currently selected label is displayed in the colors currently selected for its category in the parent Key Display Options window faded by the currently selected fade level. The unselected key labels are displayed in the unfaded colors.

Clicking on one of the unselected key labels will select it.

6.7 Window Options

These options control how IO displays its various windows.

6.7.1 Option Set Window



Window Options (Press F1 for Help)

Keypad Window
- Organization: [✓ Portrait] Landscape Compact Expression Editor Only
- Displayed Floating-point Op Keys: Basic Only [✓ and Constants/Log/Exp] and Trigonometric and Hyperbolic
- Status Display: [ ] Extended [✓] Borders
- [ ] Show Expression Editor

Keypad/Detached Display Windows
- Main Number Display: [✓ Embedded] Detached
- Secondary Number Display
  - Display Type: [✓ None] Alternative X Y Value Mem Contents
  - Auto Alternative X: [✓] Bit Pattern [✓] Fields [✓] Components

Memories Window
- Digit Size: [✓ Small Digits] Medium Digits
- Width: [✓ Wide] Narrow

Detached Display Window
- Single Line [✓ Multiple Line]
- [✓] Align Digits

Misc
- [ ] Windows Always On Top [ ] Keypad is Owner
- [ ] Enable ToolTips [✓] Show F1 Help Reminders
- Language: English (US)

The window option set window may be displayed by selecting the “Window …” item from the Options menu, or by shift-double-clicking on the secondary number status display.

6.7.1.1 Keypad Window

Organization

This option allows the keypad window to be arranged in the most convenient way for a particular environment, e.g., to minimize the obscuring of other windows, or just to satisfy the user's personal preferences.

This option can take one of four settings:

• Portrait

This will position the operation key region above the main number entry key region, resulting in a tall window.

• Landscape

This will position the operation key region to the left of the main number entry key region, resulting in a wider window allowing more digits to be displayed in the main number display.

• Compact

This will not display the operation key region, resulting is a smaller window which will obscure less of the desktop.

• Expression Editor Only

This will not display the operation key region or the main number entry key region, but will display the expression editor to allow use of that user interface.

These option settings can be quickly cycled forwards and backwards when this window is not displayed by using Ctrl+O and Shift+Ctrl+O.

Displayed Floating-point Operations

This option determines which floating-point operation keys are displayed, not displaying keys which you are not going to use will reduce the size of the keypad window.

This option can take one of four settings:

• Basic Only

This will only display the basic floating-point operation keys. These will always be displayed for portrait and landscape organizations for floating-point codings.

• and Constants/Log/Exp

This will display the floating-point constant keys, the logarithmic function keys and exponential function keys in addition to the basic operation keys.

• and Trigonometric

This will display the trigonometric and inverse trigonometric function keys in addition to all those specified above.

• and Hyperbolic

This will display the hyperbolic and inverse hyperbolic function keys in addition to all those specified above.

These option settings can be quickly cycled forwards and backwards when this window is not displayed by using Ctrl+F and Shift+Ctrl+F.

Status Display

These options determine how status displays are shown in the keypad.

Extended

If this option is not set then only the basic status displays are shown, these are the "Flags Status", the "Number Coding Status", the "Number Display Status" and the character display and entry/trigonometric units Display.

If this option is set then in addition to the basic status displays, the "Integer/Fixed-point Overflow Handling Status" and "Number Coding Option Change Status" displays will also be shown. The Dropbox, which allows quick loading of Configuration, Field Layouts Definition and Composites Definition files, will also be shown

Ctrl+S can be used to quickly switch this option setting when this window is not displayed.

Borders

Setting this option causes the status displays to be displayed with a border. This also affects the "Secondary Number Display Status" display in the Detached Display window.

Show Expression Editor

Setting this option will display the expression editor which may be used in addition to the main number key entry keys.

Ctrl+E can be used to toggle this option setting when this option set window is not displayed.

6.7.1.2 Keypad/Detached Display Windows

Main Number Display

This option determines the location of the main number display.

This option can take one of two settings:

• Embedded

The main number display will be embedded in the keypad window.

• Detached

The main number display will be displayed in a detached window which may be independently resized to display many more digits if required.

Ctrl+D can be used to quickly switch between these option settings when this option set window is not displayed.

Secondary Number Display

These options control the appearance and contents of the secondary number display which, if it is displayed, will appear immediately beneath the main number display.

Display Type

This option controls the secondary number display for all display modes if it is not overridden by the Auto Alternative X options.

This option can take one of four settings:

• None

No secondary number display will be shown.

• Alternative X

The secondary number display will show the contents of the X register using an alternative set of display mode/number display options.

• Y Value

The secondary number display will show the contents of the Y register.

• Current Memory Contents

The secondary number display will show the contents of the current memory register.

These option settings can be quickly cycled forwards and backwards when this window is not displayed by using Ctrl+N and Shift+Ctrl+N.

Auto Alternative X

These options control the automatic display of a secondary number display of the Alternative X type for different display modes, irrespective of the setting of the Display Type option above.

Bit Pattern

Setting this option causes the automatic display for Bit Pattern Display Modes.

Fields

Setting this option causes the automatic display for Fields Display Modes.

Components

Setting this option causes the automatic display for Floating-point Components Display Mode.

6.7.1.3 Memories Window

These options control the size of memory register displays in the memories window.

Digit Size

This option can take one of two settings:

• Small Digits

Small digits will be used to display the memory register contents.

• Medium Digits

Medium digits will be used to display the memory register contents.

Width

This option can take one of two settings:

• Wide

Wide display areas will be used to display the memory register contents.

• Narrow

Narrow display areas will be used to display the memory register contents.

6.7.1.4 Detached Display Window

Detached Display Type

This option controls which type of detached display is used. The setting of this option can be toggled by clicking on the maximize button of the detached display window (it is next to the close button).

• Single Line

Displays a single line, similar in appearance to the display in the keypad window. The width of the window can be changed by dragging a boundary or corner of the window, but not the height.

• Multiple Line

Displays multiple lines; this can be useful when there are more digits to be displayed than can be conveniently displayed on a single line. Both the width and height of the window can be changed by dragging a boundary or corner of the window.

Align Digits

If a Multiple Line display is selected then also selecting this option will ensure that digits are aligned with those on lines above and below. This will not otherwise be the case if any points and/or separators are displayed. This option only has any effect if all digits are displayed, i.e. the display is not scrollable.

Note that for Field and Component display modes, and segmented exponent displays, a single line format will always be used; this will be embedded within a multiple line display if that option is chosen.

Markers are not shown in multiple line displays unless an embedded single line format is used as described above.

6.7.1.5 Misc

Windows always on top

Setting this option causes IO's windows to always be displayed on top of other windows.

Keypad is Owner

Setting this option causes the Keypad window to own the Detached Display, Memories Display, Stack Display and X Register Details Display windows. Changes to this setting only come into effect after IO is restarted. The new setting must be contained in the configuration used by IO at startup; see the description of the “Startup Configuration” Calculator Operation option for details of which configuration is used at startup.

<u>When this option is selected:</u>

The Keypad window will always be behind the other IO widows, i.e., Detached Display window, Memories Display window, Stack Display window and X Register Details window. When the Keypad window is minimized the other visible IO windows will also be minimized to a single Taskbar icon. All minimized IO windows will be restored when the Taskbar icon is clicked. No IO window can be minimized independently of the others.

<u>When this option is not selected:</u>

The Keypad window may be positioned in front of or behind the other IO windows. The Keypad and Detached Display windows may be independently minimized and restored; each will have their own Taskbar icon. The Memories Display, Stack Display and X Register Details Display windows cannot be minimized.

Enable ToolTips

Setting this option causes the display of a tooltip when the mouse pointer hovers over a key or button; it will contain a brief description of the operation performed by the key or button.

Show F1 Help Reminders

Setting this option causes the display of a reminder that pressing the F1 key will display a Help window for the current window in the caption of the window. No reminder is displayed for the keypad window.

Language

This combo box allows the selection of the language used when text is displayed in a window, control, menu, or tooltip.

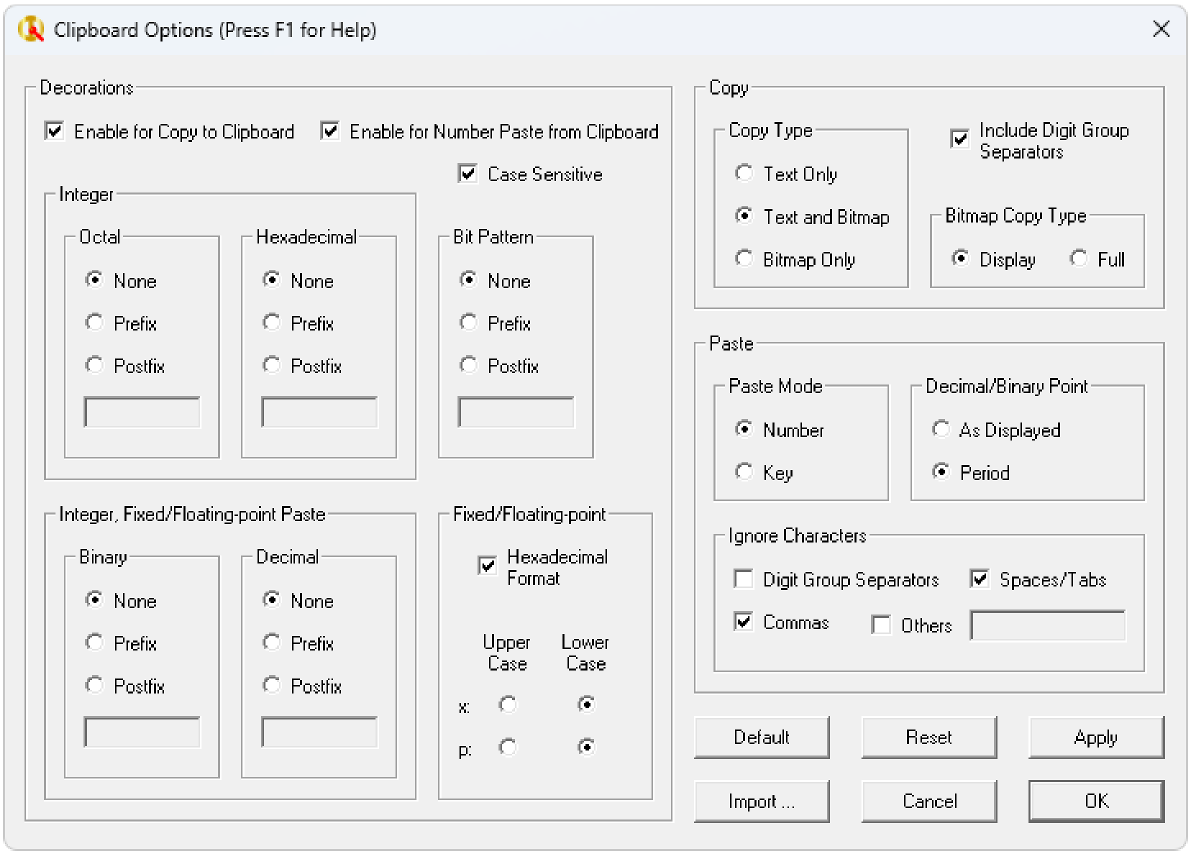
The following languages are available:

- US English
- UK English

6.8 Clipboard Options

These options control how IO interacts with the windows clipboard. Note that these options only affect clipboard operations initiated by the copy and paste buttons (and their associated keyboard shortcuts) in the keypad window, detached number display window and memories display window; they do not affect copy and paste operations from/to edit controls contained in option set windows, popup windows or the expression editor.

6.8.1 Option Set Window



Clipboard Options (Press F1 for Help)

Decorations
- [✓ Enable for Copy to Clipboard]
- [✓ Enable for Number Paste from Clipboard]
- [✓ Case Sensitive]

Integer
- Octal: [✓ None] Prefix Postfix
- Hexadecimal: [✓ None] Prefix Postfix
- Bit Pattern: [✓ None] Prefix Postfix

Integer, Fixed/Floating-point Paste
- Binary: [✓ None] Prefix Postfix
- Decimal: [✓ None] Prefix Postfix

Fixed/Floating-point
- [✓ Hexadecimal Format]
- x: Upper Case [✓ Lower Case]
- p: Upper Case [✓ Lower Case]

Copy
- Copy Type: Text Only [✓ Text and Bitmap] Bitmap Only
- [✓ Include Digit Group Separators]
- Bitmap Copy Type: [✓ Display] Full

Paste
- Paste Mode: [✓ Number] Key
- Decimal/Binary Point: As Displayed [✓ Period]
- Ignore Characters: Digit Group Separators [✓ Spaces/Tabs] [✓ Commas] Others

The clipboard option set window may be displayed by selecting the “Clipboard …” item from the Options menu.

6.8.1.1 Number Decorations

These options allow you to specify short character strings which will be added/removed to/from digit strings copied/pasted to/from the clipboard when using an integer number coding type.

Decoration (prefix or postfix) strings can be up to six characters long

Enable for Copy to Clipboard

If this option is selected and the “Base” Number Display option is set to two, eight, ten or sixteen then any prefixes/postfixes defined here for that number base will be prepended/appended to digits copied to the clipboard.

Enable for Number Paste from Clipboard

If this option is selected and the “Paste Mode” option is set to “Number”, then if any of the prefixes/postfixes defined above are present in the string being pasted they are removed and the identified number base is used to convert the pasted digits to a value irrespective of the setting of the “Base” Number Display option.

Case Sensitive

If this option is selected and the "Enable for Number Paste from Clipboard" option is selected, then a prefix/postfix will only match (and be removed) if the cases of all pairs of matched alphabetic characters (if any) are the same.

Octal

These options can be used when copying to the clipboard if the “Base” Number Display option is set to eight. This option can be used when pasting a number from the clipboard to identify that octal digits are being pasted, even if a base other than eight is selected in the Number Display Options. These options only apply to the integer number coding type.

This option can take one of three settings:

• None

No prefix or postfix characters are used to identify a string of octal digits.

• Prefix

The specified prefix characters are used to identify a following string of octal digits.

• Postfix

The specified postfix characters are used to identify a preceding string of octal digits.

Hexadecimal

These options can be used when copying to the clipboard if the “Base” Number Display option is set to sixteen. These options can be used when pasting a number from the clipboard to identify that hexadecimal digits are being pasted, even if a base other than sixteen is selected in the Number Display Options. These options only apply to the integer number coding type.

This option can take one of three settings:

• None

No prefix or postfix characters are used to identify a string of hexadecimal digits.

• Prefix

The specified prefix characters are used to identify a following string of hexadecimal digits.

• Postfix

The specified postfix characters are used to identify a preceding string of hexadecimal digits.

Binary

These options can be used when copying to the clipboard if either:

> The “Number Type” Number Coding option is set to “Integer” and the “Base” Number Display option is set to two

or

> The “Number Type” Number Coding option is set to “Fixed-point” or “Floating-point” and the “Display Format” Number Display option is set to “Binary General” or “Binary Exponent”

This option can be used when pasting a number from the clipboard to identify that binary digits are being pasted, even if a base other than two is selected in the Number Display Options. These options apply to all number coding types for paste operations and the integer number coding type only for copy operations.

This option can take one of three settings:

• None

No prefix or postfix characters are used to identify a string of binary digits.

• Prefix

The specified prefix characters are used to identify a following string of binary digits.

• Postfix

The specified postfix characters are used to identify a preceding string of binary digits.

Decimal

These options can be used when copying to the clipboard if either:

> The “Number Type” Number Coding option is set to “Integer” and the “Base” Number Display option is set to ten

or

> The “Number Type” Number Coding option is set to “Fixed-point” or “Floating-point” and the “Display Format” Number Display option is set to “Decimal General”, “Binary Exponent” or “Binary Engineering”

These options can be used when pasting a number from the clipboard to identify that decimal digits are being pasted, even if a base other than ten is selected in the Number Display Options. These options apply to all number coding types for paste operations and the integer number coding type only for copy operations.

This option can take one of three settings:

• None

No prefix or postfix characters are used to identify a string of decimal digits.

• Prefix

The specified prefix characters are used to identify a following string of decimal digits.

• Postfix

The specified postfix characters are used to identify a preceding string of decimal digits.

Bit Pattern

When copying to the clipboard these options can be used in a Bit Pattern display mode. When pasting a number from the clipboard these options can be used to identify that bit settings are being pasted, even if a Bit Pattern display mode is not selected. These options apply to all number coding types.

This option can take one of three settings:

• None

No prefix or postfix characters are used to identify a string of bit settings.

• Prefix

The specified prefix characters are used to identify a following string of bit settings.

• Postfix

The specified postfix characters are used to identify a preceding string of bit settings.

Fixed/Floating-Point

These options, if selected, apply to fixed-point and floating-point number coding types and will override any binary or decimal decorations specified above.

If the Hexadecimal Format option is set then the hexadecimal format for floating-point defined in IEEE 754 will be used for fixed-point or floating-point number coding types.

Hexadecimal Format

If this option is set then strings in hexadecimal floating-point format to be copied to/pasted from the clipboard.

Prefix Case

This option can take one of two settings:

• Uppercase

"0X" will be used as a prefix to the hexadecimal digits, and "P" will be used as a prefix to the exponent value.

• Lowercase

"0x" will be used as a prefix to the hexadecimal digits, and "p" will be used as a prefix to the exponent value.

The case of the generated hexadecimal digits will be determined by the current setting of the “Alphabetic Digit Case” Number Display option.

6.8.1.2 Copy

These options affect what is copied to the clipboard.

Copy Type

This option controls what type(s) of content is copied to the clipboard when a copy operation is performed.

This option can take one of three settings:

• Text Only

Only a text string representing the current register contents is copied to the clipboard.

• Text and Bitmap

Both a text string and a graphical display bitmap representing the current register contents are copied to the clipboard.

• Bitmap Only

Only a graphical display bitmap representing the current register contents is copied to the clipboard. This can be useful when you want to paste a graphic into an application window which can accept both text and graphics, and prioritizes text, e.g., a word processor

Include Group Separators

Setting this option causes a matching character for digit group separators selected in the Number Display Options, if any, to be included in the text string copied to the clipboard. The matching characters are as follows:

Digit Group Separator	**Matching Character**
None	None
Comma	,

Quote	'
Low Dot	.
(Blank)	(space)

Bitmap Copy Type

This option determines the contents of the graphical display bitmap copied to the clipboard.

This option can take one of two settings:

• Display

The current display contents are copied to the clipboard. This will include borders and, for the main number display, the markers area.

• Full

What would be the contents of the display if it was exactly the right size to display the whole of the current register contents are copied to the clipboard. This will not include borders or, for the main number display, the markers area. This will include the vertical separators for fields and components when using those Display Modes that would display them.

6.8.1.3 Paste

These options affect what is pasted from the clipboard.

Paste Mode

This option controls how characters pasted from the clipboard are interpreted.

This option can take one of two settings:

• Number

The pasted characters are used to define a numerical value to be stored in the register. Each character will be interpreted as a digit of the number base selected in the Number Display Options unless one of the currently defined prefixes/postfixes is used, in which case the identified number base is used to determine the value.

• Key

Each pasted character is interpreted as if that character had been entered from the keyboard; this means that as well as entering numerical values using digit keys, operations for which keyboard shortcuts are defined can also be performed.

Decimal/Binary Point

This option controls which character pasted from the clipboard is interpreted as a decimal/binary point in Number Paste Mode.

This option can take one of two settings:

• As Displayed

If decimal/binary points are displayed as a low comma (as specified in the Number Display Options) then a comma character will be interpreted as a decimal/binary point, otherwise a dot character will be interpreted as a decimal/binary point.

• Period

A dot character will be interpreted as a decimal/binary point irrespective of how a decimal/binary point is displayed.

The selected character will never be ignored, irrespective of the settings of the following options

---

Ignore Characters

These options identify characters which will be ignored in Number Paste Mode.

Pasted characters will only be processed if all characters are: valid for the number base in use, or part of an enabled number decoration, or one of the characters specified by these options, Otherwise, a Number Paste operation will do nothing.

Digit Group Separators

Setting this option causes all matching characters for the digit group separators selected in the Number Display Options, if any, to be ignored.

Spaces/Tabs

Setting this option causes space and tab characters to be ignored.

Commas

Setting this option causes comma characters to be ignored.

Others

Setting this option causes all the specified characters to be ignored. Up to eight characters may be specified.

The decimal/binary point character will never be ignored, irrespective of the settings of the above options.

6.8.1.4 Error Messages

The following error messages may be displayed after the “Apply” or “OK” buttons are used to indicate that the control settings would result in an invalid set of options. The current option settings will remain unchanged. The control containing the error will be selected/highlighted.

Prefix text not specified
Postfix text not specified

A decoration of the indicated type for a number base is selected but no decoration text has been supplied for that base.

Invalid prefix text

The text specified for a prefix decoration is invalid. A prefix cannot end with a valid digit character for the associated base. There is one exception; a single character prefix of “0” is permitted.

Invalid postfix text

The text specified for a postfix decoration is invalid. A postfix cannot start with a valid digit character for the associated base. Also, a binary postfix cannot start with a decimal digit.

Duplicate prefix text specified

The same prefix text has been specified for more than one decoration; this is not permitted as it would not be possible to determine the required base.

Duplicate postfix text specified

The same postfix text has been specified for more than one decoration; this is not permitted as it would not be possible to determine the required base.

No Other Ignore Characters specified

Other Number Paste Ignore Characters has been selected but no other characters have been supplied.

Invalid Other Ignore Character(s) specified

An illegal Other Number Paste Ignore Character has been specified, one (or more) of:
+ - 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

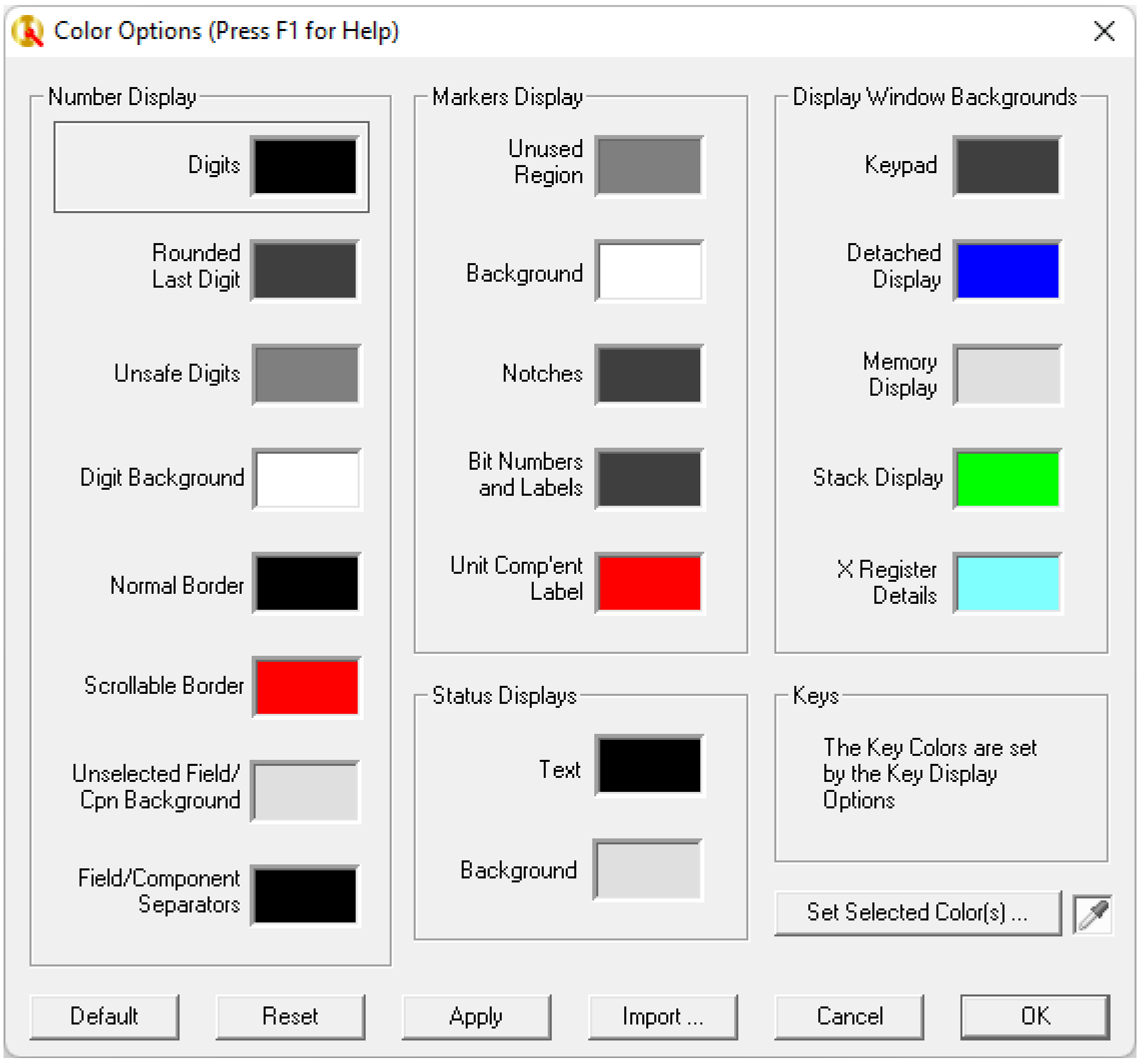
Decimal/Binary Point cannot be ignored

The specified decimal/binary point character has been specified as a character to be ignored. This is not permitted.

6.9 Color Options

These options control the colors used by IO for the various parts of its windows.

6.9.1 Option Set Window



The Key Colors are set by the Key Display Options

The color option set window may be displayed by selecting the “Color …” item from the Options menu.

It can take some time to set up all the colors to what you want if you feel so inclined. In which case, it is a good idea to save a configuration containing your settings to a file, then, if for any reason you lose these settings in your configuration, e.g., by loading the default configuration, you can quickly reinstate your preferred Color options by using the “Import …” button to load them from your saved configuration file.

Each color is identified by the name of the item immediately to the left of a block of the currently selected color for that item. At least one, and possibly more, item colors will be selected at any time. The currently selected item color(s) will be identified by enclosing frames.

The mouse is used to modify the current selection by clicking on either the name or color block as follows:

Left Click

The clicked item color will be selected and any other currently selected item colors will be deselected.

Left Double Click

The first click will select the single item color as described above, and the second click will display a color selection window for the item color.

Ctrl + Left Click

If the clicked item color was not currently selected then it is added the current selection.

If the clicked item color is currently selected then it is removed from the current selection provided it is not the only item color selected, in which case nothing changes.

Item colors can be modified by either:

1. Selecting the item(s) and then either:
    a. Clicking the "Set Selected Color(s) ..." button to display a color selection window
    b. Clicking the eyedropper button to change the mouse pointer to a color selection cursor
2. Double-clicking on the item name or its color block to display a color selection window.

6.9.1.1 Number Display

These colors are used in all number displays. These include the main and secondary number displays in the Keypad and Detached Display, the memory register contents displayed in the Memories display window, and the stack register contents displayed in the Stack display window.

Digits:	This is the color in which safe digits will be displayed. Safe digits are all integer number type digits and accurate fixed/floating-point number type digits.
Rounded Last Digit:	This is the color in which a rounded last digit will be displayed. This will only be used for the last displayed fixed/floating-point number type digit if it has been rounded up due of the Last Digit Rounding Type currently selected in the Number Display options.
Unsafe Digits:	This is the color in which unsafe digits will be displayed. Unsafe digits are fixed/floating-point number type digits which are not guaranteed to be accurate. See the Number Display section for a full discussion of safe/unsafe digits.
Digit Background:	This is the color for the background of number displays. For the Field and Component Display Modes this color will only be used for the selected Field/Component.

Normal Border:	This is the color of the border of number displays when all the digits of the number are displayed, i.e., the size of the display area is large enough to accommodate them all.
Scrollable Border:	This is the color of the border of number displays when not all the digits of the number are displayed, i.e., the size of the display area is not large enough to accommodate them all. This indicates that the display contents can be scrolled by dragging the inside of the display area with the mouse, or by using the keyboard keys associated with scrolling.
Unselected Field/Cpn Background:	This is the color for the background to digit displays for unselected Fields/Components in the Field/Component Display Modes.
Field/Component Separators:	This is the color of the vertical lines used to separate the Fields/Components in the Field/Component Display Modes.

6.9.1.2 Markers Display

These colors are used in the markers region of the main number displays in the Keypad and Detached Display.

Unused Region:This color is used to indicate the unused area of the main display, i.e., the area that cannot contain any digits for the current combination of Number Coding Options, Number Display Options and Display Mode.

Background:	This is the color for the background to the markers region under the used area of the main display.
Notches:	This is the color of the notches used to mark the boundaries between groups of four bits in Bit Pattern and Field Display Modes.
Bit Numbers and Labels:	This is the color of bit numbers and labels displayed in Bit Pattern, Field and Component Display Modes. In Component Display Mode it is not used for the unit component label.
Unit Comp’ent Label:	This is the color of the unit component label in Component Display Mode.

6.9.1.3 Display Window Backgrounds

These colors are used for the backgrounds to IO's windows.

Keypad:	This is the background color for the Keypad window.
Detached Display:	This is the background color for the Detached Display window.
Memory Display:	This is the background color for the Memories Display window.
Stack Display:	This is the background color for the Stack Display window.

6.9.1.4 Status Displays

These colors are used in all status displays, including the extended status displays.

Text:	This is the color of the text in the status displays.
Background:	This is the background color of the status displays.

6.9.1.5 Control Buttons

Set Selected Color(s) ...

Displays a color selection window to allow the selection of a new color for the currently selected item(s).

Changes the mouse pointer to an eyedropper shape, this may then be used to select a pixel (almost) anywhere on the screen. The color of the selected pixel will become the new color for the currently selected item(s).

This operation may be performed in one of two ways:

a) Press the left mouse button down while the mouse pointer is over the button, move the mouse pointer to the desired color on the screen, release the left mouse button.

b) Press and release the left mouse button while the mouse pointer is over the button, move the mouse pointer to the desired color on the screen, press and release the left mouse button. To cancel the operation, press and release the right mouse button.

Using method b), the mouse pointer will only maintain the eyedropper shape while it is over one of IO's windows, when moved into another window it will assume the shape normally used for that (part of the) window. This does not affect the operation, but can make it more difficult to identify the screen pixel whose color is selected.

When the left mouse button is clicked over a window using method b), the click may be processed by the window as normal, as well as the pixel color being selected by IO. If this is a problem, use method a).

Very occasionally, properties of (part of) a clicked window will not permit the color of the pixel to be determined; if this happens the selected color will remain unchanged.

6.9.1.6 Error Messages

The following error messages may be displayed after the "Apply" or "OK" buttons are used to indicate that the control settings would result in a display element being invisible. The current option settings will remain unchanged.

Digits Color cannot be the same as Digit Background Color

Rounded Last Digit Color cannot be the same as Digit Background Color

Unsafe Digits Color cannot be the same as Digit Background Color

Digits Color cannot be the same as Unselected Field/Component Background Color

Notches Color cannot be the same as Markers Background Color

Bit Numbers/Labels Color cannot be the same as Markers Background Color

Unit Component Label Color cannot be the same as Markers Background Color

Status Text Color cannot be the same as Status Background Color

If for some reason that is what you want then you can make the colors differ by 1 in one of their components, the resulting difference should be imperceptible to the human eye.

7 CONFIGURATIONS

A configuration is a collection of specific settings for all options.

A configuration can be stored in a configuration file from where it can be loaded.

There are option settings for controlling the configuration used when IO starts up.

7.1 Default Configuration

There is a built-in configuration called the default configuration which can be loaded quickly by selecting the “Load Defaults” item from the “Options” Menu, and specified as the start-up configuration in the “Calculator Operation” options.

The default configuration contains the following option settings:

Number Coding Option Set	
Number Coding Type	Floating-point
Integer Width	32
Integer Coding Form	2’s Complement
Number of bits per BCD Digit	4
Fixed-point Width	16
Fixed-point Precision	4
Fixed-point Coding Form	Unsigned
Floating-point Coding	64-bit
Fixed/Floating-point Rounding Type	To Nearest (Even)
Always use To Nearest (Away From Zero) for Round To Integer Ops	Yes
Number Display Option Set	
Integer Base	10
Display as for Unsigned Coding if Base is a power of 2	Yes
Leading Zeros (non-neg only)	No
Not For Decimal Base	Yes
Big Endian	Yes
Alphabetic Digit Case	Upper
Identify NaN Type	No
Floating-point Sign Field Display Format	Plus/Minus Sign
Floating-point Exponent Field Display Format	Signed Decimal
Floating-point Significand Field Display Format	Binary
Fixed/Floating-point Display Format	Decimal General
Fixed/Floating-point Trailing Zeros	Yes
Decimal Negative Exponent Limit	-4
Decimal Positive Exponent Limit	9
Binary Negative Exponent Limit	-4
Binary Positive Exponent Limit	16
Displayed Decimal Digits	Auto
All Significant Digits	No
Exact Digits Only	Yes
Number of Displayed Decimal Digits	6
Number of Displayed Decimal Places	2

Last Displayed Digit Rounding Type	To Nearest (Away From Zero)
Decimal/Binary Point	Low Dot
Safe Digits	Calculated
Exponent Style	Uppercase
Exponent Explicit Plus	No
Min Digits	Yes
Minimum Number of Exponent Digits	2
Exponent Display Format Type	Continuous
Sign Segment	No
Exponent Prefix Segment	Yes
Digit Group Separators	None
Digit Group Separators for Decimal Base only	Yes
Decimal Group Size	3 Digits
Binary Group Size	8 Digits
Markers Position	Below
Bit Numbering	lsb = 0
All Bits	No
Bit Number Leading Zeros	No
Display Field Labels	No
Displayed Bit Numbers	Word msb, Byte lsb
Explicit Plus Sign	No
Digit Justification	Right
Calculator Operation Option Set	
Calculator Type	Algebraic Logic
= Key Repeats Previous Op	Yes
Use Operator Precedence	Yes
Implied Multiplication Before (	No
Register Stack Size	4
Auto Push on New Number Entry	Yes
Auto Duplicate T on Pop	No
Int↔Int Register Contents Preservation	Bit Pattern
Int↔Fix Register Contents Preservation	Bit Pattern
Int↔Flo Register Contents Preservation	Value
Fix↔Fix Register Contents Preservation	Value
Fix↔Flo Register Contents Preservation	Value
Flo↔Flo Register Contents Preservation	Value
Int↔Int Incompatible Register Contents Handling	Prevent Change
Int↔Fix Incompatible Register Contents Handling	Prevent Change
Int↔Flo Incompatible Register Contents Handling	Prevent Change
Fix↔Fix Incompatible Register Contents Handling	Prevent Change
Fix↔Flo Incompatible Register Contents Handling	Prevent Change
Flo↔Flo Incompatible Register Contents Handling	Round Contents
Always preserve value for changes to/from Excess coding	Yes
Trigonometric Function Units	Radians
Convert X On Change	No
Expression Editor Auto Filter	No
Expression Editor Invalid Char(s) Warning	Yes
Expression Editor Allow Memory Ops	No
Expression Editor Auto Clear	No

Expression Editor Auto Append Algebraic Logic =	Yes
Integer/Fixed-point Overflow Handling	Truncate Bit Pattern
Display indicator on Overflow	No
Automatically convert -0 to +0	No
Floating-point Operation Type	Intel FPU
Value of Zero to the power of Zero	0
Startup Configuration	Last Used
Shift/Rotate Ops	Multiple Bit
ACOT Range	(0, +pi)
Integer/Fixed-point Z and N Flag Setting Control	Bit Pattern
Floating-point Z and N Flag Setting Control	Value
Prevent Mathematically Undefined Floating-point Operations	Yes
Prevent Ineffective Operations	No
Digit Set Option Set	
Medium/Large Digit Set Source	Standard
Selected Medium/Large Standard Digit Set	(first)
Selected Font for Medium/Large Digit Set	(none)
Heavy Font for Medium/Large Digit Set	No
Stretch Font for Medium/Large Digit Set	No
Small Digit Set Source	Standard
Selected Small Standard Digit Set	(first)
Selected Font for Small Digit Set	(none)
Heavy Font for Small Digit Set	No
Stretch Font for Small Digit Set	No
Digit Size	Medium
Enable Bit Pattern Digit Set	No
Selected Bit Pattern Digit Set	(first)
Key Display Option Set	
Number Entry Key Foreground Color	Black
Number Entry Key Background Color	White
Value Operation Key Foreground Color	Black
Value Operation Key Background Color	Light Magenta
Bit Pattern Operation Key Foreground Color	Black
Bit Pattern Operation Key Background Color	Light Yellow
Memory Operation Key Foreground Color	Black
Memory Operation Key Background Color	Light Cyan
Number Coding Option Key Foreground Color	Black
Number Coding Option Key Background Color	Light Green
Number Display Option/Display Mode Key Foreground Color	Black
Number Display Option/Display Mode Key Background Color	Light Red
Display Operation/Clipboard Operation Key Foreground Color	Black
Display Operation/Clipboard Operation Key Background Color	White
Key Border Style	Raised
Key Label Style	Traditional
Disabled Keys Display	Faded Key
Fade Level	4/8
Audible Keypress Notification	None
Visual Keypress Notification	No
Custom Key Labels	Same as Traditional

Window Option Set	
Organization	Portrait
Displayed Floating-point Operations	and Const/Log/Exp
Extended Status Displays	No
Status Display Borders	Yes
Show Expression Editor	No
Main Number Display	Embedded
Secondary Number Display	None
Auto Alternative X Display for Bit Pattern Display Modes	Yes
Auto Alternative X Display for Fields Display Modes	Yes
Auto Alternative X Display for Components Display Mode	Yes
Detached Display Window	Single Line
Align Digits	Yes
Memories Window Digit Size	Small
Memories Window Width	Wide
Windows always on top	No
Keypad is Owner	No
Enable ToolTips	No
Show F1 Help Reminders	Yes
Language	US English
Clipboard Option Set	
Decorations Enabled for Copy to Clipboard	No
Decorations Enabled for Paste from Clipboard	No
Case Sensitive Decorations	Yes
Binary Prefix Decoration Enabled	No
Binary Postfix Decoration Enabled	No
Binary Prefix/Postfix Characters	(none)
Octal Prefix Decoration Enabled	No
Octal Postfix Decoration Enabled	No
Octal Prefix/Postfix Characters	(none)
Decimal Prefix Decoration Enabled	No
Decimal Postfix Decoration Enabled	No
Decimal Prefix/Postfix Characters	(none)
Hexadecimal Prefix Decoration Enabled	No
Hexadecimal Postfix Decoration Enabled	No
Hexadecimal Prefix/Postfix Characters	(none)
Hexadecimal Format Fixed/Floating-point	No
Case of Hexadecimal Format Fixed/Floating-point prefixes	Lower
Copy Type	Text Only
Include Digit Group Separators	Yes
Bitmap Copy Type	Display
Paste Mode	Number
Decimal/Binary Point Character	Period
Ignore Digit Group Separators	No
Ignore Spaces/Tabs	Yes
Ignore Commas	Yes
Ignore Other Characters	No
Other Ignored Characters	(none)
Color Option Set	

Digits	Black
Unsafe Digits	Mid Gray
Digit Background	White
Normal Display Area Border	Black
Scrollable Display Area Border	Red
Unselected Field/Component Background	Light Gray
Field/Component Separators	Black
Markers Unused Region	Light Gray
Markers Background	White
Marker Notches	Dark Gray
Bit Numbers and Labels	Dark Gray
Unit Component Label	Red
Keypad Background	Dark Gray
Detached Display Background	Blue
Memory Display Background	Light Gray
Stack Display Background	Green
Status Display Text	Black
Status Display Background	Light Gray

7.2 Configuration Files

A configuration can be stored in a configuration file. Configuration files have a default filename extension of “.ioc”.

A current configuration can be stored in a configuration file by selecting the “Save Config…” item from the File menu.

The current configuration can be set to that stored in a configuration file by selecting the “Load Config…” item from the File menu, or by dragging the file from a Windows Explorer window and dropping it onto the Dropbox.

A configuration file to be automatically loaded upon startup, if desired, can be identified by selecting the “Save Startup Config” item from the File menu.

The controls in an option set window can be set to reflect the options for that option set stored in a configuration file by clicking on the “Import…” button in the window, or by dragging the file from a Windows Explorer window and dropping it onto the “Import…” button.

When IO terminates cleanly the current configuration is automatically stored in a “Last Used Configuration” file.

Note that only option settings are stored in a configuration file; the following are NOT stored:

- Display mode
- Register stack contents
- Operator stack contents
- Last X register contents
- Previous operation operator and register contents
- Memory registers contents
- Visibility of Help, Memories, Stack and X Register Details windows
- Window positions
- Detached Display Window width
- Multi-line Detached Display Window height

- Expression editor expression

The previous Number Coding option settings to be used by the “Use Previous Number Coding” operation are saved in addition to the current settings.

7.3 Startup Configuration

When IO starts up the initial configuration is determined by the following algorithm:

```
IF there is a stored last used configuration available
    Load the stored last used configuration
ELSE
    Load the default configuration
ENDIF
then
IF the “Startup Configuration” option is set to “Default”
    Load the default configuration
ELSE IF the “Startup Configuration” option is set to “Saved Startup”
    IF there is a saved startup configuration available
        Load the configuration saved by the last use of the “Save Startup Config” File menu
        item
    ENDIF
ENDIF
```

8 KEYPAD WINDOW

8.1 Organizations

The organization of the keypad window is determined by the following criteria:

The Keypad Window Organization option setting currently selected in the window options. This setting can be cycled using the Ctrl+O keyboard shortcut.

The Keypad Window Displayed Floating-point Operation Keys option setting currently selected in the window options. This setting can be cycled using the Ctrl+F keyboard shortcut.

The Keypad/Detached Main Number Display option setting currently selected in the window options. This setting can be toggled using the Ctrl+D keyboard shortcut.

The Keypad Window Status Display option setting currently selected in the window options. This setting can be toggled using the Ctrl+S keyboard shortcut.

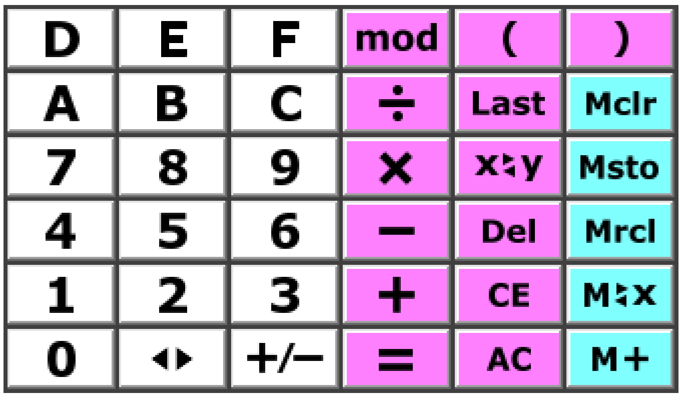
The Keypad/Detached Secondary Number Display Type option setting currently selected in the window options. This setting can be cycled using the Ctrl+N keyboard shortcut.

The Keypad Window Show Expression Editor option setting currently selected in the window options. This setting can be toggled using the Ctrl+E keyboard shortcut.

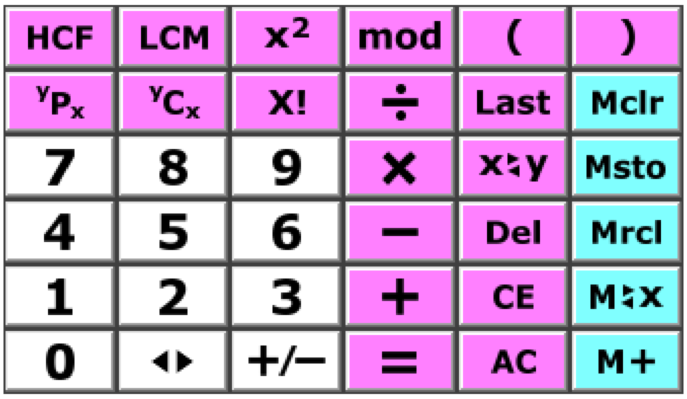
8.2 Keypad Regions

The keypad window is divided into six regions:

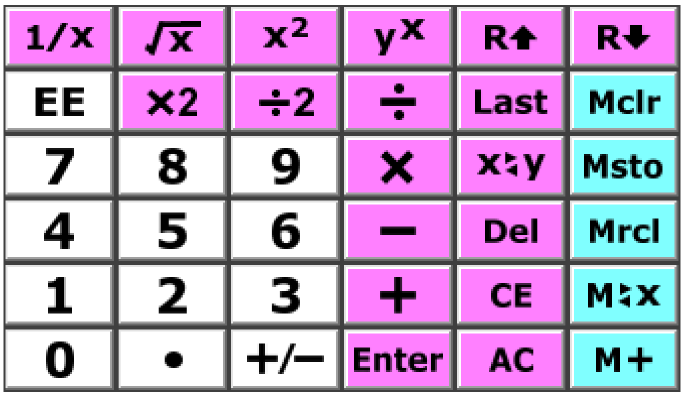
• Main Keys Region



Integer, Hexadecimal, Algebraic Logic



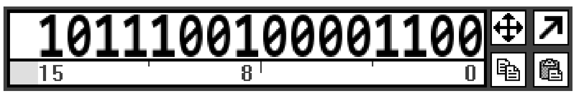
Integer, Decimal, Algebraic Logic

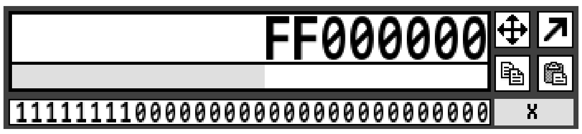


Fixed/Floating-Point Reverse Polish Notation

The main keys region contains all the large keypad keys; these comprise keys for number entry, the basic operators, the basic memory operations and other basic operations.

• Display Region

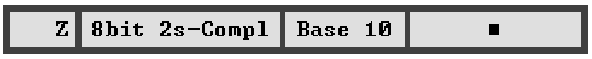




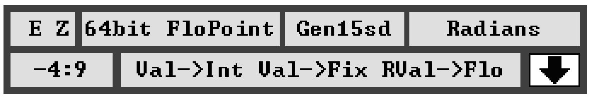
The display region is located at the top of the keypad window. It contains IO’s main number display which usually shows the current contents of the X register; it can optionally contain a secondary number display and its associated status display. This display region is removed from the keypad when the detached number display is made visible.

The display region contains square keys for digit size selection, detaching the display region from the keypad window and the clipboard operations,

• Status Display Region



Basic

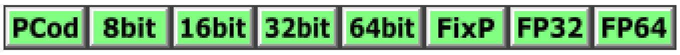


Extended

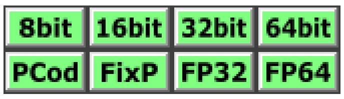
The status display region can be configured to contain one (Basic) or two (Extended) rows of status displays.

The status display region does not contain any keys. See the Status Displays section for a description of its contents.

• Number Coding Keys Region



Portrait/Compact, Floating-Point Landscape

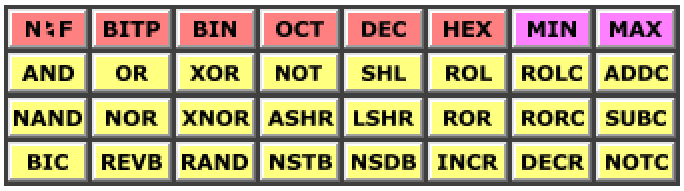


Integer/Fixed-Point Landscape

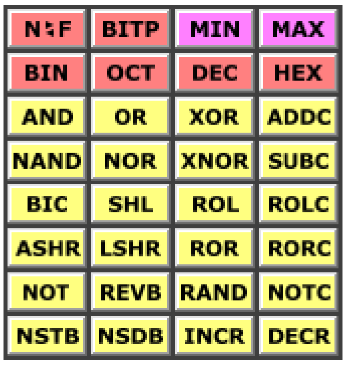
The number coding keys region is not present in the compact and expression editor only keypad organizations.

The number coding keys are replaced by two combo box controls when either the Integer Fields Values or the Floating-point Components display mode are entered by using the appropriate key. They are re-displayed when the same key is subsequently used to return to the original display mode.

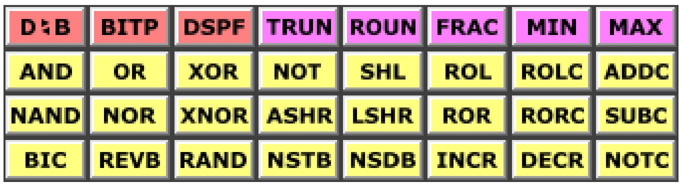
• Operation Keys Region



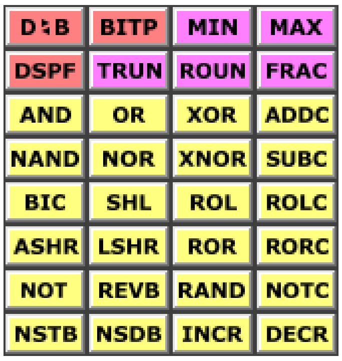
Integer, Portrait/Compact



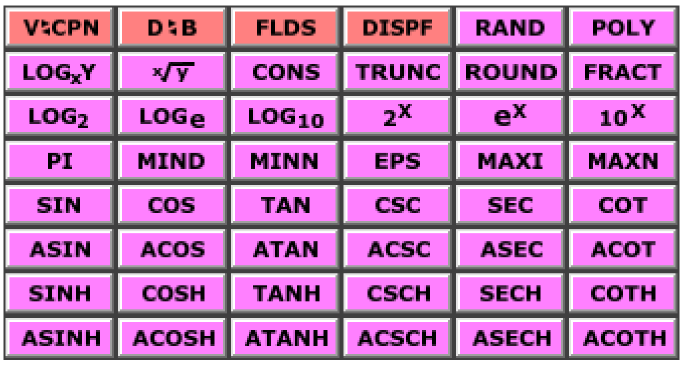
Integer, Landscape


D⇆B BITP DSPF TRUN ROUN FRAC MIN MAX
AND OR XOR NOT SHL ROL ROLC ADDC
NAND NOR XNOR ASHR LSHR ROR RORC SUBC
BIC REVB RAND NSTB NSDB INCR DECR NOTC

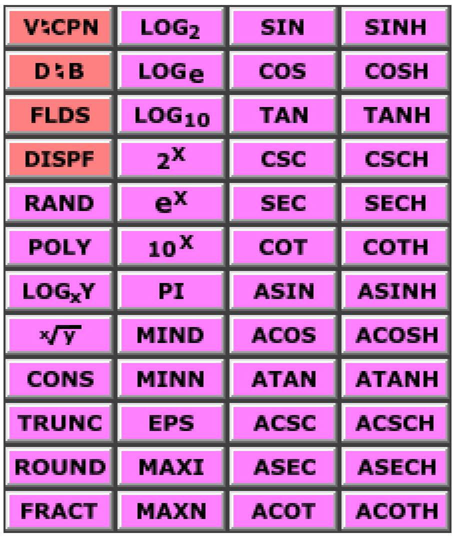
Fixed-point, Portrait/Compact



Fixed-point, Landscape


V⇆CPN D⇆B FLDS DISPF RAND POLY
LOGxY x√y CONS TRUNC ROUND FRACT
LOG2 LOGe LOG10 2X eX 10X
PI MIND MINN EPS MAXI MAXN
SIN COS TAN CSC SEC COT
ASIN ACOS ATAN ACSC ASEC ACOT
SINH COSH TANH CSCH SECH COTH
ASINH ACOSH ATANH ACSCH ASECH ACOTH

Floating-point, Portrait/Compact, all functions



Floating-point. Landscape, all functions

The operation keys region is not present in the “compact” or “expression editor only” keypad organizations.

• Expression Editor Region

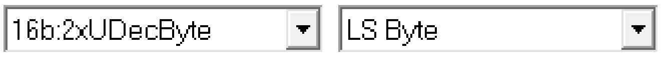
The expression editor region is only present if the “Show Expression Editor” window option is set, or if “Expression Editor Only” is selected for the “Organization” window option. The “Show Expression Editor” window option setting can be toggled by using the Ctrl+E keyboard shortcut.

The expression editor region contains the expression editor edit control and the Execute Expression button.

8.3 Combo Boxes

The number coding keys are replaced by two combo box controls when either the Integer Fields Values or the Floating-point Components display mode are entered by using the appropriate key.

8.3.1 Integer Fields Display Mode


16b:2xUDecByte
LS Byte

The first combo box is used to select a field layout from those currently available. It always displays the name of the currently selected field layout.

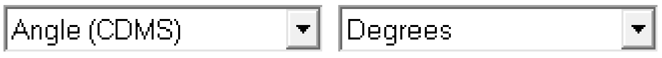
Field layouts which are wider than the width of the current integer coding will not be available for selection; neither will field layouts which are not wide enough to accommodate the current full bit pattern.

The second combo box may be used to select a field within the currently selected field layout. It always displays the name of the currently selected field, irrespective of how the field was selected.

Both combo boxes will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- A register on the register stack, other than the X register, contains a bit pattern which is not all ⓪s

8.3.2 Floating-Point Components Display Mode


Angle (CDMS)
Degrees

The first combo box is used to select a composite from those currently available. It always displays the name of the currently selected composite. It will be disabled if any register on the register stack contains a non-zero value.

The second combo box may be used to select a component of the currently selected composite. It always displays the name of the currently selected component, irrespective of how the component was selected.

The first combo box will be disabled under any of the following circumstances:

- A register on the register stack contains a value which is not exact-zero
- The algebraic logic operator stack is not empty, i.e., there is a pending operator to be applied

The second combo box will be disabled under any of the following circumstances:

- A register on the register stack, other than the X register, contains a value which is not exact-zero
- The algebraic logic operator stack is not empty, i.e., there is a pending operator to be applied

8.4 Keys

The following descriptions of the keypad keys are grouped as follows:

In the following descriptions of the operations performed by IO’s keys:

All key labels which can be selected for the operation are shown; many of these are only available if you use a custom key label set.

The “Flags Affected” tables indicate how each of the flags is/may be affected by the operation:

- indicates that the flag will remain unchanged

⓪ indicates that the flag will be set to ⓪

① indicates that the flag will be set to ①

◉ indicates that the flag may or may not be changed, depending on the outcome of the operation

Note that only the flags that are displayed in the Flags Status display when the key is available are shown, e.g., the Denormal and Exact flags are not shown for keys only available for the integer number coding type.

Also note that the settings of the Z flag shown will not be correct for Integer Excess coding if the “Integer/Fixed-point Z/N Flag Setting Control” Calculator Operation option is set to “Use Bit Pattern”.

8.4.1 Main Keys Region

8.4.1.1 All Number Coding Types/Calculator Types

Decimal Digits

0 1 2 3 4 5 6 7 8 9

Keyboard Shortcuts: 0 1 2 3 4 5 6 7 8 9 nkp0 nkp1 nkp2 nkp3 nkp4 nkp5 nkp6 nkp7 nkp8 nkp9

If the entry of a number is in progress, then the clicked digit is appended to the digits already entered, if an exponent is being entered then it will be appended to the exponent digits already entered. Otherwise, the entry of a new number will commence and the clicked digit will be the first digit of the new number being entered.

One or more of these keys may be disabled for the current display mode as follows:

Display Mode	**Disabled Digit Keys**
Integer Fields Bit Pattern Floating-Point Fields Floating-Point Fields Bit Pattern	All
Integer Word Bit Pattern Fixed-point Word Bit Pattern	2 ...... 9 0, 1 if bit settings for every bit in the word have already been entered, i.e., msb is set to ①
Integer Word Value Integer Fields Values	In Integer Field Values display mode, if the current field is displayed as a bit pattern, then the keys described in the row above are disabled, where the condition is applied to the current field instead of the entire word Otherwise, those which, if used, would result in a value being entered which is outside of the range that may be represented using the current integer number coding
Fixed-point Word Value Floating-Point Word Value	If a mantissa is being entered then those which, if used, would result in a mantissa value being entered which is outside of the range that may be represented using the current number coding If an exponent value is being entered then those which, if used, would result in a value being entered which is outside of the range that may be represented using the current number coding
Floating-Point Components	Those which, if used, would result in a component value being entered which is outside of the range for the current component

Flags Affected:

C	Z	N	V	D	E	O

-	◉	◉	⓪	◉	◉	⓪

Change Sign

+/− **CHS**

Keyboard Shortcut: #

The operation performed by this key will depend on circumstances as follows:

Circumstances	Operation
Integer number entry	Changes the sign of the currently entered integer value
Fixed-point mantissa entry Floating-point mantissa entry	Changes the sign of the currently entered value
Fixed-point exponent entry Floating-point exponent entry	If the exponent value is not zero then changes the sign of the exponent value If the exponent value is zero then prepares for the entry of a negative exponent value
Number entry not in progress	Changes the sign of the value currently contained in the X register, i.e., it behaves as a function. If the X register contains a Floating-point Infinity class bit pattern, then it will switch between the bit patterns for positive infinity and negative infinity.

There are circumstances in which this key will be disabled, as follows:

Circumstances	Key Disabled
Integer Fields Bit Pattern, Floating-Point Fields or Floating-Point Fields Bit Pattern display mode	Always
Number coding is Integer Unsigned, Integer Binary Coded Decimal or Fixed-point Unsigned	Always
Integer number entry	If changing the sign of the currently entered number would result in a value which is outside of the range that may be represented using the current integer number coding
Fixed-point mantissa entry	If changing the sign of the currently entered mantissa value would result in a mantissa value which is outside of the range that may be represented using the current fixed-point number coding
Fixed-point exponent entry	If changing the sign of the currently entered exponent value would result in a value which is outside of the range that may be represented using the current fixed-point number coding
Floating-point exponent entry	If changing the sign of the currently entered exponent value would result in a value which is outside of the range that may be represented using the current floating-point number coding

Number entry not in progress	If changing the sign of the value currently contained in the X register would result in a value which is outside of the range that may be represented using the current number coding

Flags Affected:

C	Z	N	V	D	E	O
-	-	◉	⓪	-	-	⓪

Add

+

Keyboard Shortcuts: + kp+ nkp+

When this operator is applied the values in the Y and X registers are added together to provide the result.

Floating-point Special Cases:

		X					
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN	-Inf	-Inf	-Inf	-Inf	QNaN
	-0	QNaN	-Inf	-0	-0	QNaN	+Inf
Y	0	QNaN	-Inf	-0	0	+0	+Inf
	+0	QNaN	-Inf	QNaN	+0	+0	+Inf
	+Inf	QNaN	QNaN	+Inf	+Inf	+Inf	+Inf

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain floating-point Infinity class bit patterns of different signs
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain inexact zeros of different signs

Flags affected when operator applied:

C	Z	N	V	D	E	O
◉	◉	◉	◉	◉	◉	◉

Subtract

−

Keyboard Shortcuts: - kp- nkp-

When this operator is applied the value in the X register is subtracted from the value in the Y register to provide the result.

Floating-point Special Cases:

		X					
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN	QNaN	-Inf	-Inf	-Inf	-Inf
Y	-0	QNaN	+Inf	QNaN	-0	-0	-Inf
	0	QNaN	+Inf	+0	0	-0	-Inf
	+0	QNaN	+Inf	+0	+0	QNaN	-Inf
	+Inf	QNaN	+Inf	+Inf	+Inf	+Inf	QNaN

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain floating-point Infinity class bit patterns of the same sign
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain inexact zeros of the same sign

Flags affected when operator applied:

C	Z	N	V	D	E	O
●	●	●	●	●	●	●

Multiply

× ∗

Keyboard Shortcuts: * kp* nkp*

When this operator is applied the values in the Y and X registers are multiplied together to provide the result.

Floating-point Special Cases:

		X					
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
Y	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN	+Inf	QNaN	0	QNaN	-Inf
	-0	QNaN	QNaN	+0	0	-0	QNaN
	0	QNaN	0	0	0	0	0
	+0	QNaN	QNaN	-0	0	+0	QNaN
	+Inf	QNaN	-Inf	QNaN	0	QNaN	+Inf

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain a value of inexact-zero and a floating-point Infinity class bit pattern

Flags affected when operator applied:

C	Z	N	V	D	E	O
⦿	⦿	⦿	⓪	⦿	⦿	⦿

Divide

÷ /

Keyboard Shortcuts: / kp/ nkp/

When this operator is applied the value in the Y register is divided by the value in the X register to provide the result.

Floating-point Special Cases:

		X					
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
Y	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN	QNaN	+Inf	QNaN	-Inf	QNaN
	-0	QNaN	+0	QNaN	QNaN	QNaN	-0
	0	QNaN	0	0	QNaN	0	0

+0	QNaN	-0	QNaN	QNaN	QNaN	+0
+Inf	QNaN	QNaN	-Inf	QNaN	+Inf	QNaN

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation”, the “Number Coding Type” Number Coding option is set to “Integer” or “Fixed-point”, and the X register contains a value of zero
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X register contains a value of exact-zero
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain floating-point Infinity class bit patterns
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X and Y registers contain values of inexact-zero

Flags affected when operator applied:

C	Z	N	V	D	E	O
◉	◉	◉	⓪	◉	◉	◉

Replace X with Last X

Last **◄ x**

The contents of the X and Last X registers are exchanged.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

Swap X with Y

x⇆y **Swap**

The contents of the X and Y registers are exchanged.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- Algebraic logic Y register contents undefined, i.e., there are no operators on the operator stack other than open-brackets

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

Delete Digit

Del ▶ ◀

Keyboard Shortcut: Backspace

If an exponent is being entered then the most recently entered exponent digit is removed; otherwise, if a mantissa is being entered then the most recently entered mantissa digit or the decimal/binary point is removed; otherwise, the most recently entered number digit is removed.

This key will be disabled under any of the following circumstances:

- Number entry is not in progress
- The X register contains a floating-point NaN, Infinity or Pseudo class bit pattern
- The "Number Coding Type" Number Coding option is set to "Integer", the "Prevent Ineffective Operations" Calculator Operation option is selected, and the X register contains a value of zero
- The "Number Coding Type" Number Coding option is set to "Fixed-point" or "Floating-point", the "Prevent Ineffective Operations" Calculator Operation option is selected, and no non-zero digits entered

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	-	◉	⓪

Clear Entry

CE 0▸X

Keyboard Shortcuts: Del kp.

The contents of the X register are overwritten with a value of zero. For Bit Pattern display modes it is overwritten with a bit pattern of all ⓪s, this will only make a difference for integer excess coding.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- The “Number Coding Type” Number Coding option is set to “Integer”, the “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains a value of zero
- If the “Number Coding Type” Number Coding option is not set to “Integer” and the “Prevent Ineffective Operations” Calculator Operation option is selected, then if number entry is in progress and no non-zero digits have been entered

Flags Affected:

C	Z	N	V	D	E	O
-	①	⓪	⓪	⓪	①	⓪

All Clear

AC **C** **CA** **0**

Keyboard Shortcut: Esc

The contents of all stack registers, the Last X register and the previous operation operand register are all overwritten with a value of zero. The operand stack is emptied. The previous operation operator is cleared. Memory registers are not affected.

For Bit Pattern display mode types, the registers will be overwritten with a bit pattern of all ⓪s. This will only make a difference if integer excess coding is selected.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode

Flags Affected:

C	Z	N	V	D	E	O
-	①	⓪	⓪	⓪	①	⓪

Clear Memory

Mclr **0▸M** **MC**

Keyboard Shortcuts: z Z

If the current memory register does not contain a value of zero then the contents of its previous contents register are overwritten with the contents of the current memory register and the contents of the current memory register are overwritten with a value of zero. For Bit Pattern display modes the current memory register is overwritten with a bit pattern of all ⓪s, this will only make a difference for integer excess coding.

This key will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register contains a value of zero/bit pattern

No flags are affected.

Store X to Memory

Msto **X▸M** **MS** **STO** **M in**

Keyboard Shortcuts: s S

If the current memory register does not have the same contents as the X register then the contents of its previous contents register are overwritten with the contents of the current memory register and the contents of the current memory register are overwritten with the contents of the X register.

This key will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register has the same content as the X register

No flags are affected.

Recall X from Memory

Mrcl **M▸X** **MR** **RCL**

Keyboard Shortcuts: m M

The contents of the X register are overwritten with the contents of the current memory register.

This key will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register has the same content as the X register

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

Swap X with Memory

M⇄X Msw Mex

Keyboard Shortcuts: x X

If the current memory register does not have the same contents as the X register then the contents of its previous contents register and the X register are overwritten with the contents of the current memory register and the contents of the current memory register are overwritten with the original contents of the X register.

This key will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register has the same content as the X register

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

Add X to Memory

M+ Madd

Keyboard Shortcuts: p P

The values contained in the X register and the current memory register are added together and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents of the current memory register are written to its previous contents register.

This key will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains a value of exact-zero

No flags are affected.

8.4.1.2 Integer Number Coding Type

All Number Bases

Edit Digits

◀▶ Edit

The displayed digits are treated as if they had just been entered by the user, this means that the delete key may then be used to remove the rightmost digit and the digit keys may be used to append further digits.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Number entry is in progress
- IO has just started up
- Either the All Clear key or the Clear Entry key has just been used

Flags Affected:

C	Z	N	V	O
-	-	-	⓪	⓪

Modulo

mod %

Keyboard Shortcut: %

When this operator is applied the value in the Y register is divided by the value in the X register and the remainder provides the result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X register contains a value of zero

Flags affected when operator applied:

C	Z	N	V	O
-	◉	◉	⓪	◉

Number Bases Greater Than 10 and Not Bit Pattern Display Mode Type

---

Non-Decimal Digits

A	B	C	D	E	F
a	b	c	d	e	f

Keyboard Shortcuts: a b c d e f A B C D E F

If the entry of a number is in progress, then the clicked digit is appended to the digits already entered. Otherwise, the entry of a new number will commence and the clicked digit will become the first digit of the number being entered.

Those keys which, if used, would result in a value being entered which is outside of the range that may be represented using the current integer number coding will be disabled.

Flags Affected:

C	Z	N	V	O
-	⓪	-	⓪	⓪

Number Bases Of 10 Or Less or Bit Pattern Display Mode

Highest Common Factor

HCF	GCD	(x,y)

When this operator is applied the result is the Highest Common Factor of X and Y, i.e., the largest integer value that divides exactly into both X and Y.

This is also referred to as the Greatest Common Divisor.

If this operator is applied when the “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the X register does not contain a positive value, then the result will be the value zero and the Value Overflow flag will be set.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The X register does not contain a positive value
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register does not contain a positive value

Flags affected when operator applied:

C	Z	N	V	O
-	◉	⓪	⓪	◉

Lowest Common Multiple

LCM **[x,y]**

When this operator is applied the result is the Lowest Common Multiple of X and Y, i.e., the smallest integer value that divides exactly by both X and Y.

Calculation:

$$[\mathrm{X}, \mathrm{Y}] = \frac{\mathrm{X}}{(\mathrm{X}, \mathrm{Y})} \mathrm{Y}$$

If this operator is applied when the “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the X register does not contain a positive value, then the result will be the value zero and the Value Overflow flag will be set.

If the result is too large to be represented using the current coding, then a saturated result, i.e., the largest value that can be represented by the current coding, is returned and the Value Overflow flag will be set. The setting of the “Integer/Fixed-point Overflow Handling” Calculator Operation option does not affect this behavior.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The X register does not contain a positive value
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register does not contain a positive value

Flags affected when operator applied:

C	Z	N	V	O
-	◉	⓪	⓪	◉

Integer Square of X

x² **Sqr**

Keyboard Shortcut: “

The value in the X register is replaced with its square, i.e., the original value multiplied by itself.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	O
-	-	⓪	⓪	◉

Permutations of X from Y

$^{Y}P_{x}$ $_{Y}P_{x}$ **Perm**

When this operator is applied the result is the number of possible permutations of X items from a collection of Y items.

Calculation:

$$^{Y}P_{X} = \frac{Y!}{(Y - X)!}$$

If this operator is applied when the “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the X register does not contain a positive value then the result will be the value zero and the Value Overflow flag will be set.

If the result is too large to be represented using the current coding, then a saturated result, i.e., the largest value that can be represented by the current coding, is returned and the Value Overflow flag will be set. The setting of the “Integer/Fixed-point Overflow Handling” Calculator Operation option does not affect this behavior.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The X register does not contain a positive value
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register does not contain a positive value

Flags affected when operator applied:

C	Z	N	V	O
-	◉	⓪	⓪	◉

Combinations of X from Y

$^{Y}C_{x}$ $_{Y}C_{x}$ $\binom{Y}{X}$ **Comb**

When this operator is applied the result is the number of possible combinations of X items from a collection of Y items.

Calculation:

$$^{Y}C_{X} = \frac{Y!}{X!\,(Y - X)!}$$

If this operator is applied when the “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the X register does not contain a positive value, then the result will be the value zero and the Value Overflow flag will be set.

If the result is too large to be represented using the current coding, then a saturated result, i.e., the largest value that can be represented by the current coding, is returned and the Value Overflow flag will be set. The setting of the “Integer/Fixed-point Overflow Handling” Calculator Operation option does not affect this behavior.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The X register does not contain a positive value
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register does not contain a positive value

Flags affected when operator applied:

C	Z	N	V	O
-	⦿	⓪	⓪	⦿

Factorial of X

X! **Fact**

Keyboard Shortcut: !

The value in the X register is replaced with its factorial, i.e., the product of itself and each positive integer value below it for positive values of X, or 1 if the X register contains a value of zero.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The X register contains a negative value
- The X register contains a value whose factorial is greater than the largest number that can be represented by the current integer coding

Flags Affected:

C	Z	N	V	O
-	⓪	⓪	⓪	⓪

8.4.1.3 Fixed/Floating-Point Number Coding Types

Decimal/Binary Point

●

Keyboard Shortcuts: . nkp.

This key enters a decimal point into the entered floating-point value for decimal display formats, or a binary point for binary display formats.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode and base component not selected
- Floating-point Components display mode, base component selected, and base component defined with zero decimal places
- Fixed/Floating-point mantissa being entered and binary/decimal point already entered
- Fixed/Floating-point exponent being entered

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

Enter Decimal Exponent

EE $x10^n$ E e

Keyboard Shortcuts: e E

This key switches between mantissa and exponent value entry for a number base of ten.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- Number entry not in progress
- Mantissa is being entered and mantissa has exact-zero or negative-zero value
- Exponent is being entered and exponent value is not zero

No flags are affected.

Enter Binary Exponent

EE $x2^n$ B b

Keyboard Shortcuts: b B

This key switches between mantissa and exponent value entry for a number base of two.

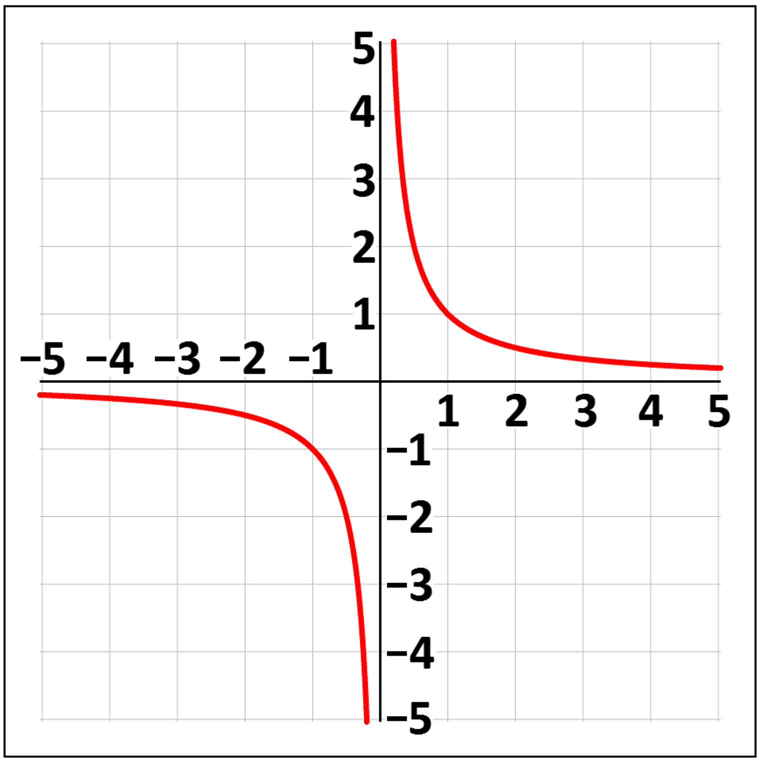
This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- Number entry not in progress
- Mantissa is being entered and mantissa has exact-zero or negative-zero value
- Exponent is being entered and exponent value is not zero

No flags are affected.

Reciprocal of X

1/x **x^{-1}** **Recip**



The value in the X register is replaced with its reciprocal, i.e one divided by its original value.

Calculation:

$$\text{recip}(X) = \frac{1}{X}$$

Domain: { x ∈ ℝ : x ≠ 0 } Range: { x ∈ ℝ : x ≠ 0 }

Exact Results:

Many, most notably:

$$\frac{1}{1} = 1, \quad \frac{1}{-1} = -1, \quad \frac{1}{2} = 0.5, \quad \frac{1}{-2} = -0.5, \quad \frac{1}{0.5} = 2, \quad \frac{1}{-0.5} = -2$$

Floating-point Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	-Inf	QNaN	+Inf	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Number Coding Type” Number Coding option is set to “Fixed-point” and the X register contains a value of zero

- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of +1 or -1

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

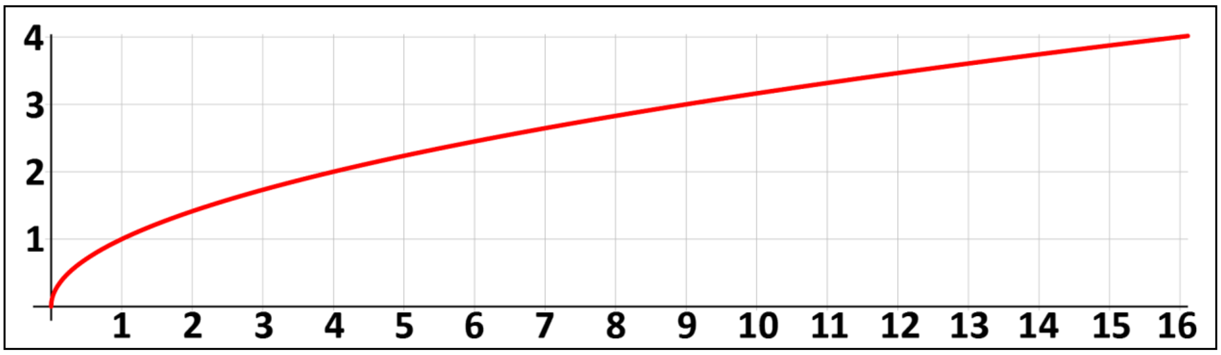
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value of exact-zero

Flags Affected:

C	Z	N	V	D	E	O
-	⓪	◉	⓪	◉	◉	◉

Positive Square Root of X

$\sqrt{x}$ $x^{1/2}$ Sqrt



The value in the X register is replaced with its non-negative square root.

Domain: ℝ[0, ∞)Range: ℝ[0, ∞)

Exact Results:

Many, most notably: $\sqrt{0} = 0,\ \sqrt{0.25} = 0.5,\ \sqrt{1} = 1,\ \sqrt{4} = 2$

Floating-point Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	0	+0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Number Coding Type” Number Coding option is set to “Fixed-point” and the X register contains a negative value
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero or +1

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

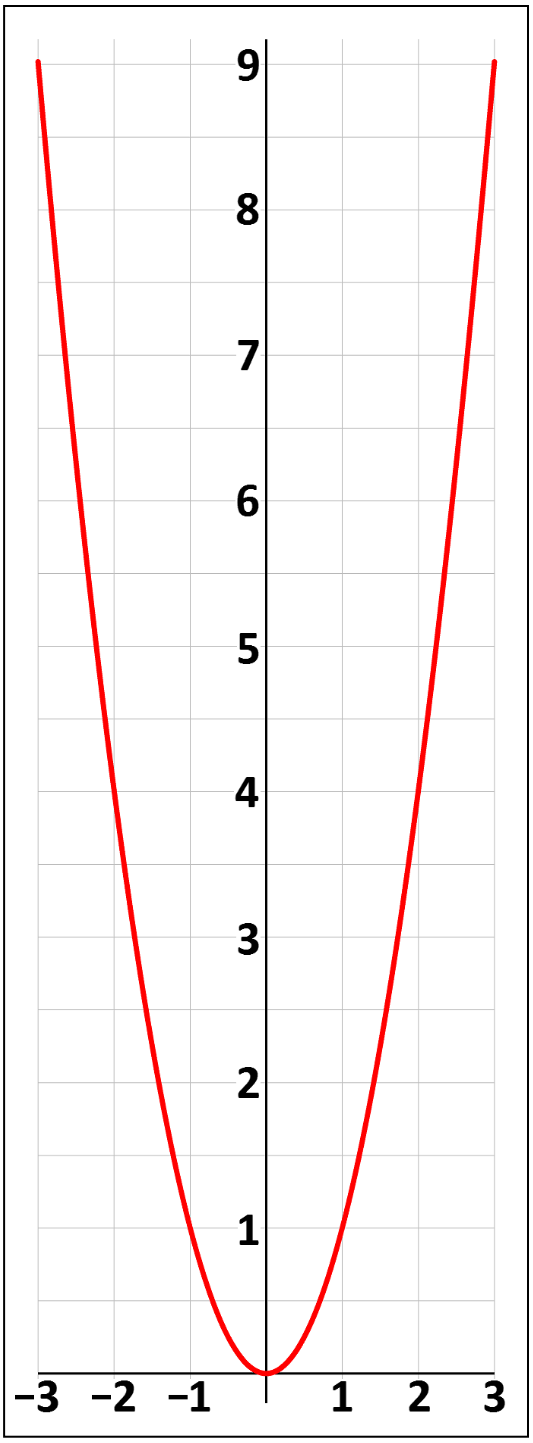
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a negative value

Flags Affected:

C	Z	N	V	D	E	O
-	-	⓪	⓪	◉	◉	◉

Fixed/Floating-point Square of X

x^2 Sqr



The value in the X register is replaced with its square, i.e., the original value multiplied by itself.

Calculation:

$$\mathrm{sqr}(X) = X \times X$$

Domain: ℝ

Range: ℝ[0, ∞)

Exact Results:

Many, most notably:

$$0^2 = 0, \quad 0.5^2 = 0.25, \quad 1^2 = 1,$$
$$(-0.5)^2 = 0.25, \quad (-1)^2 = 1$$

Floating-point Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+Inf	+0	0	+0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero or +1

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under the following circumstances:

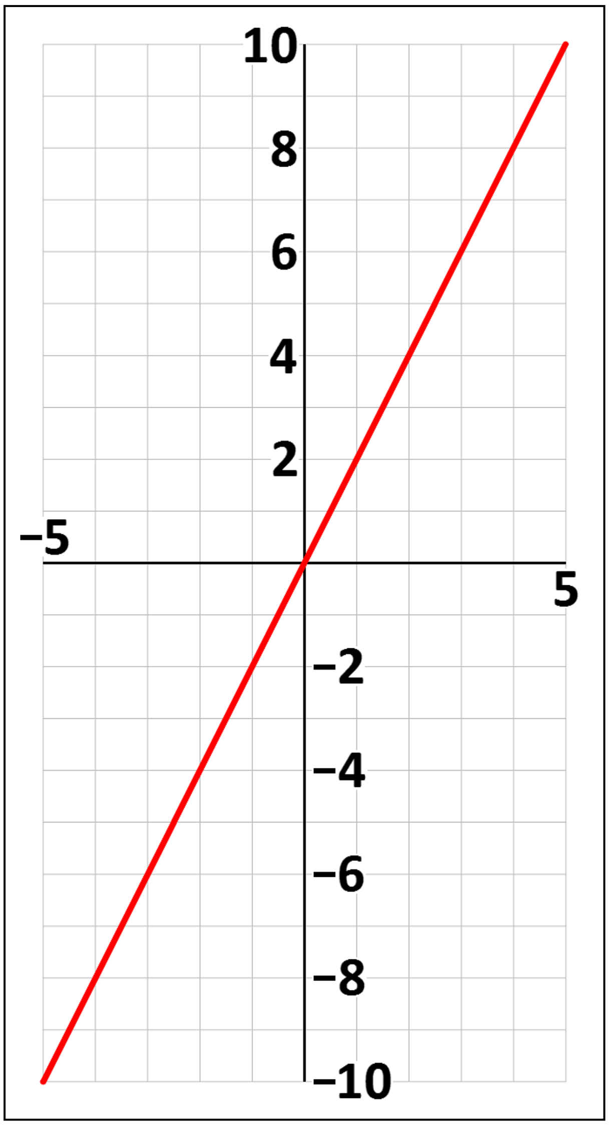
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

C	Z	N	V	D	E	O
-	-	⓪	⓪	◉	◉	◉

Multiply X by 2

×2 | ∗2 | Mul2 | 2x



The value in the X register is multiplied by two.

Calculation:

$$\mathrm{mul2}(\mathrm{X}) = 2 \times \mathrm{X}$$

Domain: $\mathbb{R}$

Range: $\mathbb{R}$

Exact Results:

Many, most notably:

$$0 \times 2 = 0, \qquad 1 \times 2 = 2,$$
$$0.5 \times 2 = 1, \qquad (-1) \times 2 = -2$$

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern

Floating-point Special Cases:

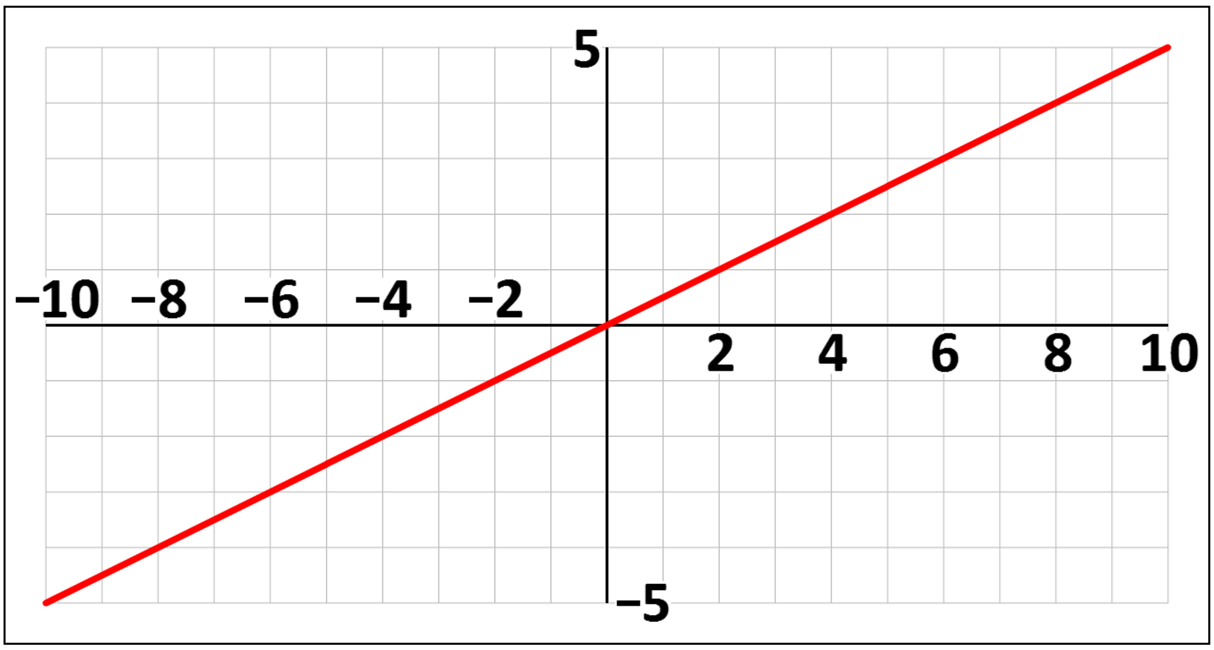
X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-Inf	-0	0	+0	+Inf

Flags Affected:

C	Z	N	V	D	E	O
-	-	-	⓪	◉	◉	◉

Divide X by 2

÷2 /2 Div2 x/2



The value in the X register is divided by two.

Calculation:

$$\operatorname{div2}(X) = \frac{X}{2}$$

Domain: $\mathbb{R}$ Range: $\mathbb{R}$

Exact Results:

Many, most notably: $0 \div 2 = 0,\ 1 \div 2 = 0.5,\ 2 \div 2 = 1,\ (-1) \div 2 = -0.5$

Floating-point Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-Inf	-0	0	+0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

C	Z	N	V	D	E	O
-	-	-	⓪	⦿	⦿	⦿

Y to the Power of X

y^x **∗∗** **^** **↑**

Keyboard Shortcut: ‘

When this operator is applied the value in the Y register is raised to the power of the value in the X register to provide the result.

Calculation:

$$Y^X = 2^{X \log_2(Y)}, \quad Y > 0$$

Floating-point Special Cases:

				X			
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
Y	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN	QNaN	QNaN	+1	QNaN	QNaN
	-0	QNaN	QNaN	QNaN	+1	QNaN	QNaN
	0	QNaN	QNaN	QNaN	0/+1/QNaN	0	0
	+0	QNaN	+Inf	+ie1	+1	+ie1	+0
	+Inf	QNaN	+0	QNaN	+1	QNaN	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Number Coding Type” Number Coding option is set to “Floating-point” and the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register contains a NaN or Pseudo class bit pattern

Flags affected when operator applied:

C	Z	N	V	D	E	O
-	-	⦿	⓪	⦿	⦿	⦿

8.4.1.5 Algebraic Logic Calculator Type

Open Bracket

([

Keyboard Shortcuts: ([

If the “Implied Multiplication Before (“ Calculator Operation option is selected and as a consequence an implied multiplication is required then the processing for the multiply key is performed.

Then, an open bracket indicator is pushed onto the operator stack, and any operators currently on the operator stack will not be applied until this open bracket has been matched by a close bracket, or the = key is used.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Implied Multiplication Before (“ Calculator Operation option is selected and the algebraic logic operator stack is full when the multiplication operator needs to be pushed onto it
- The algebraic logic operator stack is full when the open bracket indicator needs to be pushed onto it
- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Implied Multiplication Before (“ Calculator Operation option is not selected, number entry is in progress, and the last key used was not another Open Bracket

No flags are affected if the “Implied Multiplication Before (“ Calculator Operation option is not selected

Flags Affected if the “Implied Multiplication Before (“ Calculator Operation option is selected :

C	Z	N	V	D	E	O
◉	◉	◉	⓪	◉	◉	⓪

Close Bracket

)]

Keyboard Shortcuts:)]

All operators pushed onto the operator stack after the most recently pushed open bracket are applied and the open bracket indicator popped off the operator stack.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- There are no open bracket indicators on the operator stack

Flags Affected:

C	Z	N	V	D	E	O
⦿	⦿	⦿	⓪	⦿	⦿	⓪

Equals

=

Keyboard Shortcuts: = Enter kpEnter nkpEnter

If the algebraic logic operator stack is not empty then all pending operators on the stack are applied. Any unmatched open brackets are effectively supplied with a matching close bracket.

If the “= Key Reats Previous Op” calculator operation option is set, and the algebraic logic operator stack is empty and there are a stored algebraic logic previous operator and 2nd operand, then the stored operator is used with the current X register contents as the first operand and the stored operand as the second.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Prevent Ineffective Operations” Calculator Operation option is selected and IO has just started up or the AC key has just been used

Flags Affected:

C	Z	N	V	D	E	O
⦿	⦿	⦿	⓪	⦿	⦿	⦿

8.4.1.7 Reverse Polish Notation Calculator Type

RPN Enter

Enter **Ent⬆**

Keyboard Shortcuts: = Enter kpEnter nkpEnter

The current contents of the register stack are pushed up so that the Y register then has the same contents as the X register and the original contents of the register at the top of the stack are lost. The contents of the X register remain unchanged so is effectively duplicated at the bottom of the stack.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern

No flags are affected.

RPN Roll Up

R⬆ **RollU**

The contents of the register stack are rolled upwards, which is equivalent to performing an Enter and then replacing the contents of the X register with the original contents of the T register, i.e. the register at the top of the stack.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

RPN Roll Down

R⬇ **RollD**

The contents of the register stack are rolled downwards, which is equivalent to copying the original contents of each register on the stack to the register below it, so the original contents of the Y register is copied to the X register, finally the original contents of the X register is copied to the T register, i.e. the register at the top of the stack.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Number Coding Type” Number Coding option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern

Flags Affected:

C	Z	N	V	D	E	O
-	⦿	⦿	⓪	⦿	⦿	⓪

8.4.2 Operation Keys Region

8.4.2.1 Integer/Fixed-Point Number Coding Types

Integer/Fixed-point Bit Pattern Display Switch

BITP N⇅BP ⓪①

Switches between “Integer Word Value” / “Fixed-point Word Value” and “Integer Bit Pattern” / “Fixed-point Bit Pattern” display modes.

No flags are affected.

Minimum Value for Current Coding

MIN ⤓

This function sets the contents of the X register to the bit pattern used to represent the lowest number that may be represented using the current coding. For integer unsigned and BCD codings and fixed-point unsigned coding this will be zero, for integer 2’s complement and excess codings and fixed-point 2’s complement coding this will be the largest possible negative value.

This key will be disabled in Integer Fields Bit Pattern display mode.

Some examples for integer coding types:

	2’s Complement	**Excess**
8 bits	-128	-127
16 bits	-32,768	-32,767
32 bits	-2,147,483,648	-2,147,483,647
64 bits	-9,223,372,036,854,775,808	-9,223,372,036,854,775,807
80 bits	-604,462,909,807,314,587,353,088	-604,462,909,807,314,587,353,087
96 bits	-39,614,081,257,132,168,796,771,975,168	-39,614,081,257,132,168,796,771,975,167

Flags Affected:

C	Z	N	V	E	O
-	⦿	⦿	⓪	①	⓪

Maximum Value for Current Coding

MAX ⤒

This function sets the contents of the X register to the bit pattern used to represent the highest number that may be represented using the current coding. This will always be a positive value.

This key will be disabled in Integer Fields Bit Pattern display mode.

Some examples for integer coding types:

	2’s Complement	Excess
8 bits	127	128
16 bits	32,767	32,768
32 bits	2,147,483,647	2,147,483,648
64 bits	9,223,372,036,854,775,807	9,223,372,036,854,775,808
80 bits	604,462,909,807,314,587,353,087	604,462,909,807,314,587,353,088
96 bits	39,614,081,257,132,168,796,771,975,167	39,614,081,257,132,168,796,771,975,168

	Unsigned	BCD (4 bits per digit)
8 bits	255	99
16 bits	65,535	9,999
32 bits	4,294,967,295	99,999,999
64 bits	18,446,744,073,709,551,615	9,999,999,999,999,999
80 bits	1,208,925,819,614,629,174,706,175	99,999,999,999,999,999,999
96 bits	79,228,162,514,264,337,593,543,950,335	999,999,999,999,999,999,999,999

Flags Affected:

C	Z	N	V	E	O
-	⓪	⓪	⓪	①	⓪

Bitwise AND

AND &

Keyboard Shortcut: &

When this operator is applied the bit patterns in the Y and X registers are biwise-and’ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise and of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	⓪	⓪
	①	⓪	①

This operation is commonly used to clear a selection of bits within a word to ⓪ but leave all the other bits as they were. In this case the bit pattern in the second operand (X register) is called the mask; the bits in the mask which are set to ① identify the bits in the first operand (Y register) which are to have their settings preserved in the result, and the bits in the mask which are set to ⓪ identify the bits which are to be set to ⓪ in the result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Inclusive OR

OR I

Keyboard Shortcut: |

When this operator is applied the bit patterns in the Y and X registers are biwise-inclusive-or'ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise inclusive or of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	⓪	①
	①	①	①

This operation is commonly used to set a selection of bits within a word to ① but leave all the other bits as they were. In this case the bit pattern in the second operand (X register) is called the mask; the bits in the mask which are set to ① identify the bits in the first operand (Y register) which are to have their settings set to ① in the result, and the bits in the mask which are set to ⓪ identify the bits which are to have their settings preserved in the result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Exclusive OR

XOR EOR ^

Keyboard Shortcut: ^

When this operator is applied the bit patterns in the Y and X registers are biwise-exclusive-or'ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise exclusive or of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	⓪	①
	①	①	⓪

This operation is commonly used to invert a selection of bits within a word but leave all the other bits as they were. In this case the bit pattern in the second operand (X register) is called the mask; the bits in the mask which are set to ① identify the bits in the first operand (Y register) which are to have their settings inverted in the result, and the bits in the mask which are set to ⓪ identify the bits which are to have their settings preserved in the result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Complemented AND

NAND ~&

When this operator is applied the bit patterns in the Y and X registers are biwise-complemented-and'ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise complement of the bitwise and of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	①	①
	①	①	⓪

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Complemented Inclusive OR

NOR ~I

When this operator is applied the bit patterns in the Y and X registers are biwise-complemented-inclusive-or'ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise complement of the bitwise inclusive or of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	①	⓪
	①	⓪	⓪

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O

⓪	◉	◉	⓪	①	⓪

Bitwise Complemented Exclusive OR

NXOR XNOR NEOR ENOR ~^

When this operator is applied the bit patterns in the Y and X registers are biwise-complemented-exclusive-or’ed together to provide the result.

This generates a bit pattern in which each bit is set to the bitwise complement of the bitwise exclusive or of the corresponding bits in the Y and X registers. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	①	⓪
	①	⓪	①

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Bit Clear

BIC &~

Keyboard Shortcut: $

When this operator is applied the bit pattern in the X register is biwise-bit-cleared with the bit pattern in the Y register to provide the result.

This generates a bit pattern in which each bit is set to the bitwise and of the corresponding bit in the Y register and the bitwise complement of the corresponding bit in the X register. The table on the right shows the resulting bit settings for all combinations of initial bit settings.

		x	
		⓪	①
y	⓪	⓪	⓪
	①	①	⓪

This operation is commonly used to clear a selection of bits within a word to ⓪ but leave all the other bits as they were. In this case the bit pattern in the second operand (X register) is called the mask; the bits in the mask which are set to ⓪ identify the bits in the first operand (Y register) which are to have their settings preserved in the result, and the bits in the mask which are set to ① identify the bits which are to be set to ⓪ in the result.

Note that unlike the six bitwise operators described above this operation can, and usually will, give a different result if the X and Y registers are swapped, i.e., this is not a commutative operation.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode

- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⓪	◉	◉	⓪	①	⓪

Bitwise Complement X

NOT CPL ~ —▷○— ¬

Keyboard Shortcut: ~

The setting of each bit in the bit pattern stored in the X register is inverted.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	E	O
-	◉	◉	⓪	①	⓪

Shift Y Left by X Bits

SHL ⇤ <<

Keyboard Shortcut: <

When this operator is applied the bit pattern in the Y register is shifted left by the number of bits identified by the value in the X register to provide the result. A ⓪ is shifted into the least significant end for each shift performed. The carry flag is set to the setting of the last bit shifted out of the most significant end.

This operator is only available when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options.

If the X register contains a negative value, then no shift is performed.

For fixed-point codings:

> This operator is not available if the coding cannot represent any positive integer values
>
> If the X register contains a non-integer value then the truncated integer part of the value will be used as the shift count

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- Fixed-point coding and coding cannot represent any positive integer values
- The algebraic logic operator stack is full when the operator needs to be pushed onto it
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

Flags affected when operator applied:

C	Z	N	V	E	O
⦿	⦿	⦿	⓪	①	⓪

Arithmetic Shift Y Right by X Bits

ASHR ⊏⇥ >>

Keyboard Shortcut: >

When this operator is applied the bit pattern in the Y register is shifted right arithmetically by the number of bits identified by the value in the X register to provide the result. The setting of the most significant bits remains unchanged for each shift performed. The carry flag is set to the setting of the last bit shifted out of the least significant end.

This operator is only available when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options.

If the X register contains a negative value, then no shift is performed.

For fixed-point codings:

> This operator is not available if the coding cannot represent any positive integer values
>
> If the X register contains a non-integer value then the truncated integer part of the value will be used as the shift count

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- Fixed-point coding and coding cannot represent any positive integer values
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
⦿	⦿	-	⓪	①	⓪

Logical Shift Y Right by X Bits

LSHR ⟹ >>>

Keyboard Shortcut: \

When this operator is applied the bit pattern in the Y register is shifted right logically by the number of bits identified by the value in the X register to provide the result. The most significant bit is set to ⓪ for each shift performed. The carry flag is set to the setting of the last bit shifted out of the least significant end.

This operator is only available when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options.

If the X register contains a negative value, then no shift is performed.

For fixed-point codings:

> This operator is not available if the coding cannot represent any positive integer values
>
> If the X register contains a non-integer value, then the truncated integer part of the value will be used as the shift count

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- Fixed-point coding and coding cannot represent any positive integer values
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
◉	◉	⓪	⓪	①	⓪

Rotate Y Left by X Bits

ROL

When this operator is applied the bit pattern in the Y register is rotated left by the number of bits identified by the value in the X register to provide the result. The carry flag and the least significant bit are set to the original setting of the last bit shifted out of the most significant end.

This operator is only available when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options.

If the X register contains a negative value, then no rotation is performed.

For fixed-point codings:

> This operator is not available if the coding cannot represent any positive integer values
>
> If the X register contains a non-integer value then the truncated integer part of the value will be used as the rotation count

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- Fixed-point coding and coding cannot represent any positive integer values
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
◉	-	◉	⓪	①	⓪

Rotate Y Right by X Bits

ROR

When this operator is applied the bit pattern in the Y register is rotated right by the number of bits identified by the value in the X register to provide the result. The carry flag and the most significant bit are set to the original setting of the last bit shifted out of the least significant end.

This operator is only available when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options.

If the X register contains a negative value, then no rotation is performed.

For fixed-point codings:

> This operator is not available if the coding cannot represent any positive integer values
>
> If the X register contains a non-integer value, then the truncated integer part of the value will be used as the rotation count

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point Value display mode
- Fixed-point coding and coding cannot represent any positive integer values
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	E	O
◉	-	◉	⓪	①	⓪

Shift X Left by 1 Bit

SHL1 << 1

Keyboard Shortcut: <

The bit pattern in the X register is replaced with the original bit pattern shifted left by one bit. The least significant bit is set to ⓪. The carry flag is set to the original setting of the most significant bit.

This function is only available when the “Shift/Rotate Operations” option is set to “Single Bit” in the Calculator Operation Options.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
◉	◉	◉	⓪	①	⓪

Arithmetic Shift X Right by 1 Bit

ASR1 >> 1

Keyboard Shortcut: >

The bit pattern in the X register is replaced with the original bit pattern shifted right arithmetically by one bit. The setting of the most significant bit remains unchanged. The carry flag is set to the original setting of the least significant bit.

This function is only available when the “Shift/Rotate Operations” option is set to “Single Bit” in the Calculator Operation Options.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
◉	◉	-	⓪	①	⓪

Logical Shift X Right by 1 Bit

LSR1 >>> 1

Keyboard Shortcut: \

The bit pattern in the X register is replaced with the original bit pattern shifted right logically by one bit. The most significant bit is set to ⓪. The carry flag is set to the original setting of the least significant bit.

This function is only available when the “Shift/Rotate Operations” option is set to “Single Bit” in the Calculator Operation Options.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
◉	◉	⓪	⓪	①	⓪

Rotate X Left by 1 Bit

ROL1

The bit pattern in the X register is replaced with the original bit pattern rotated left by one bit. The carry flag and the least significant bit are set to the original setting of the most significant bit.

This function is only available when the “Shift/Rotate Operations” option is set to “Single Bit” in the Calculator Operation Options.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
◉	-	◉	⓪	①	⓪

Rotate X Right by 1 Bit

ROR1

The bit pattern in the X register is replaced with the original bit pattern rotated right by one bit. The carry flag and the most significant bit are set to the original setting of the least significant bit.

This function is only available when the “Shift/Rotate Operations” option is set to “Single Bit” in the Calculator Operation Options.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
◉	-	◉	⓪	①	⓪

Rotate X Left through Carry by 1 Bit

ROLC

The bit pattern in the X register is replaced with the original bit pattern rotated left through the carry flag by one bit. The carry flag is set to the original setting of the most significant bit. The least significant bit is set to the original setting of the carry flag.

This function can also be used to perform a Shift Left by 1 Bit operation when the “Shift/Rotate Operations” option is set to “Multiple Bit” in the Calculator Operation Options, provided you are careful. You must check that the carry flag is not set beforehand, to prevent a ① being shifted into the lsb instead of the required ⓪; if it is set then you can clear it by using the Complement Carry operation.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The "Prevent Ineffective Operations" Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
⦿	⦿	⦿	⓪	①	⓪

Rotate X Right through Carry by 1 Bit

RORC

The bit pattern in the X register is replaced with the original bit pattern rotated right through the carry flag by one bit. The carry flag is set to the original setting of the least significant bit. The most significant bit is set to the original setting of the carry flag.

This function can also be used to perform a Shift Right by 1 Bit operation when the "Shift/Rotate Operations" option is set to "Multiple Bit" in the Calculator Operation Options, provided you are careful.

To perform a Logical Shift Right by 1 Bit you must check that the carry flag is not set beforehand, to prevent a ① being shifted into the msb instead of the required ⓪; if it is set then you can clear it by using the Complement Carry operation.

To perform an Arithmetic Shift Right by 1 Bit you must check that the carry flag is set as required beforehand, to prevent an incorrect bit being shifted into the msb. If the msb in the X register is set to ⓪ then the carry flag must be clear. If the msb in the X register is set to ① then the carry flag must be set. If the carry flag is not set as required then you can modify it by using the Complement Carry operation.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The "Prevent Ineffective Operations" Calculator Operation option is selected and performing the operation would not modify either the X register contents or the setting of the Carry flag

Flags Affected:

C	Z	N	V	E	O
⦿	⦿	⦿	⓪	①	⓪

Add with Carry

ADDC +C

When this operator is applied the values in the Y and X registers and the carry flag are added together to provide the result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point number coding type
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	O
◉	◉	◉	◉	◉

Subtract with Carry/Borrow

SUBC SUBB −C

When this operator is applied the carry flag is added to the value in the X register, and the result is subtracted from the value in the Y register to provide the final result.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- Fixed-point number coding type
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

Flags affected when operator applied:

C	Z	N	V	O
◉	◉	◉	◉	◉

Reverse Bytes in X

REVB

This function is only available if the word width is a multiple of eight, i.e., each register contains a whole number of bytes. Where N is the number of bytes in the current word width, the contents of byte n (0 ⩽ n < N) is swapped with the contents of byte ((N-1) – n). This can be used for switching the endianness of a value stored in a sequence of bytes in memory.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Number Coding Type” Number Coding option is set to “Integer” and the “Integer Width” Number Coding option is not a multiple of eight
- The “Number Coding Type” Number Coding option is set to “Fixed-point” and the “Fixed-point Width” Number Coding option is not a multiple of eight
- The “Prevent Ineffective Operations” Calculator Operation option is selected and performing the operation would not modify the X register contents

Flags Affected:

C	Z	N	V	E	O
-	-	◉	⓪	①	⓪

Random Bit Pattern

RAND ???

Keyboard Shortcut: ?

This function sets each bit of the bit pattern stored in the X register to a random setting. Each bit will have a 50% probability of being set to a ⓪ and a 50% probability of being set to a ①.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	E	O
-	◉	◉	⓪	①	⓪

Number of Significant Bits

NSTB

The bit pattern in the X register is replaced with the value of the number of significant bits in the original contents.

The value returned is the width of the smallest field within the word that contains the most significant bit that is set to ① and the least significant bit.

If no bit is set to ① then the result is zero.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The current number coding cannot represent the value of the width of the coding, which is the largest value that may be the result

Flags Affected:

C	Z	N	V	E	O
-	◉	⓪	⓪	①	⓪

Number of Significand Bits

NSDB

The bit pattern in the X register is replaced with the value of the number of significand bits required to contain the original bit pattern.

The value returned is the width of the smallest field within the word that contains the least and most significant bits that are set to ①.

This value can be used to determine whether or not the original value can be exactly represented using a particular floating-point coding or unsigned fixed-point coding.

If no bit is set to ① then the result is zero.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The current number coding cannot represent the value of the width of the coding, which is the largest value that may be the result

Flags Affected:

C	Z	N	V	E	O
-	◉	⓪	⓪	①	⓪

Increment X

INCR +1

Keyboard Shortcut: ;

The bit pattern in the X register is incremented; this is equivalent to interpreting the bit pattern using unsigned integer coding and adding 1 to it.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	E	O
◉	◉	◉	◉	①	◉

Decrement X

DECR −1

Keyboard Shortcut: :

The bit pattern in the X register is decremented; this is equivalent to interpreting the bit pattern using unsigned integer coding and subtracting 1 from it.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	E	O
◉	◉	◉	◉	①	◉

Complement Carry

NOTC ~C

This operation inverts the setting of the carry flag.

This key will be disabled in Integer Fields Bit Pattern display mode.

Flags Affected:

C	Z	N	V	E	O
◉	-	-	⓪	-	⓪

8.4.2.2 Integer Number Coding Type

<u>All Display Modes</u>

Number/Fields Display Switch

This key switches the display mode from either Integer Word Value or Integer Word Bit Pattern to Integer Fields Values, or from Integer Fields Values to the display mode in use when it was switched to Integer Fields Values.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- Integer Word Value display mode and any register on the register stack other than the X register contains a bit pattern which has at least one bit set to ①
- Integer Word Value display mode and there are no currently loaded field layouts which are both not wider than the current word width and wide enough to contain the bit pattern currently contained in the X register
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation”, Integer Fields Values display mode, and any register on the register stack other than the X register contains a non-zero value

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	⓪

Word Value/Bit Pattern Display Modes

Binary Display

BIN N_2

Sets the “Integer Base” Number Display option to two.

This key will be disabled under any of the following circumstances:

- Integer Word Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Width” Number Coding option is set to one
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Base” Number Display option is set to two

No flags are affected.

Octal Display

OCT N_8

Sets the “Integer Base” Number Display option to eight.

This key will be disabled under any of the following circumstances:

- Integer Word Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Width” Number Coding option is set to one
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Base” Number Display option is set to eight

No flags are affected.

Decimal Display

DEC N_{10}

Sets the “Integer Base” Number Display option to ten.

This key will be disabled under any of the following circumstances:

- Integer Word Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Width” Number Coding option is set to one
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Base” Number Display option is set to ten

No flags are affected.

Hexadecimal Display

HEX N_{16}

Sets the “Integer Base” Number Display option to sixteen.

This key will be disabled under any of the following circumstances:

- Integer Word Bit Pattern display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Width” Number Coding option is set to one
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the “Integer Base” Number Display option is set to sixteen

No flags are affected.

Fields Display Mode

Field Labels Switch

LABS

This toggles the selection of the “Markers Field Labels” Number Display option, which will switch between the display of bit numbers and field labels in the markers area.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Digit Size” Digit Set option is set to small

No flags are affected.

Select Previous Field

◄FLD

This makes the field immediately to the left of the currently selected field the new selected field.

If the leftmost field is currently selected then the rightmost field becomes the new selected field.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	⓪

Select Next Field

FLD►

Selects the field immediately to the right of the currently selected field as the new selected field.

If the rightmost field is currently selected then the leftmost field becomes the new selected field.

This key will be disabled under any of the following circumstances:

- Integer Fields Bit Pattern display mode
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	⓪

Clear Unselected Fields

CUSF 0X0

Sets all fields except for the selected field to a bit pattern of all ⓪s. The selected field, and hence the X register, are unaffected.

This key will be disabled in Integer Fields Bit Pattern display mode

No flags are affected.

8.4.2.3 Fixed-Point Number Coding Type

Fixed-point Base Display Change

D↔B 10↔2 N10↔2

This changes the “Fixed/Floating-point Display Format” Number Display option between decimal and binary as follows:

From	To
Decimal General	Binary General
Decimal Exponent	Binary Exponent
Decimal Engineering	Binary Exponent
Binary General	Decimal General
Binary Exponent	Decimal Exponent

No flags are affected.

Fixed/Floating-point Display Format Change

DSPF

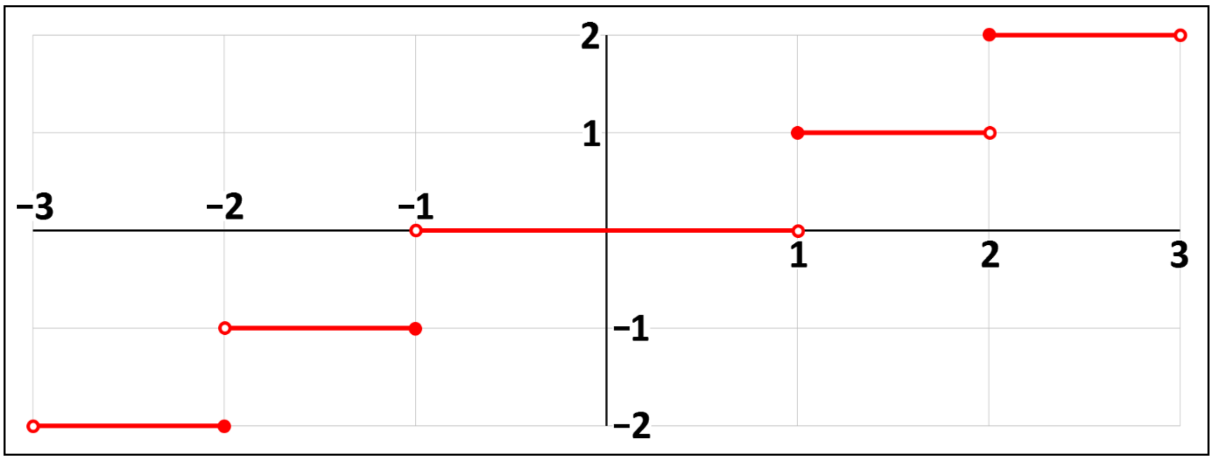
This changes the “Fixed/Floating-point Display Format” Number Display option between general, exponential and engineering as follows:

From	To
Decimal General	Decimal Exponent
Decimal Exponent	Decimal Engineering
Decimal Engineering	Decimal General
Binary General	Binary Exponent
Binary Exponent	Binary General

No flags are affected.

Fixed-point Truncate to Integer

TRUN TRUI [x] □.⊠



The value in the X register is replaced with its truncated integer part.

Domain: ℝ Range: ℤ

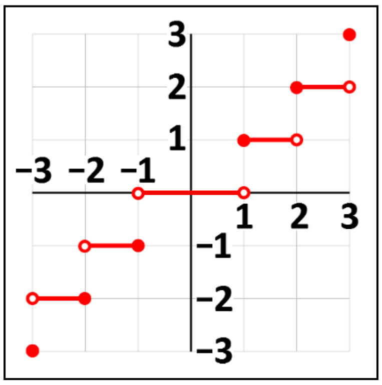
This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains an integer value.

Flags Affected:

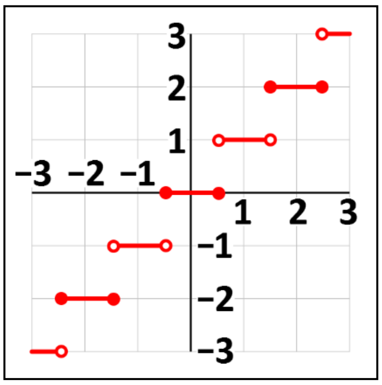
C	Z	N	V	E	O
-	◉	◉	⓪	①	⓪

Fixed-point Round to Integer

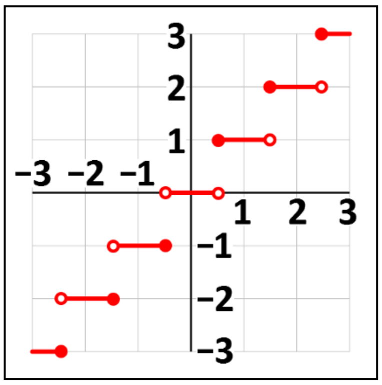
ROUN ROUI ⌊x⌉ □.⊟



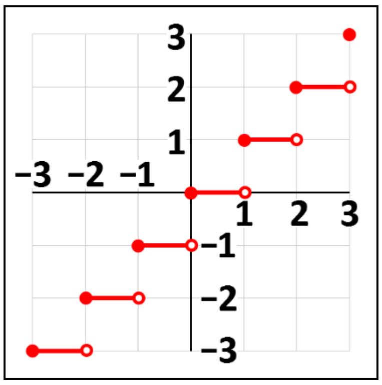
To Zero (Truncate)



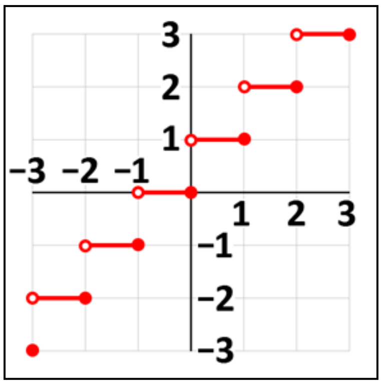
To Nearest (Even)



To Nearest (Away From Zero)



To Negative Infinity (Down)



To Positive Infinity (Up)

If the “Always use To Nearest (Away From Zero) for Round To Integer Ops” Number Coding option is selected then the value in the X register is replaced with the integer value obtained by applying the To Nearest (Away From Zero) rounding to its original value.

Otherwise, the value in the X register is replaced with the rounded integer value of its original value, which is obtained by applying the currently selected “Fixed/Floating-point Rounding Type” Number Coding option.

Domain: ℝ

Range: ℤ

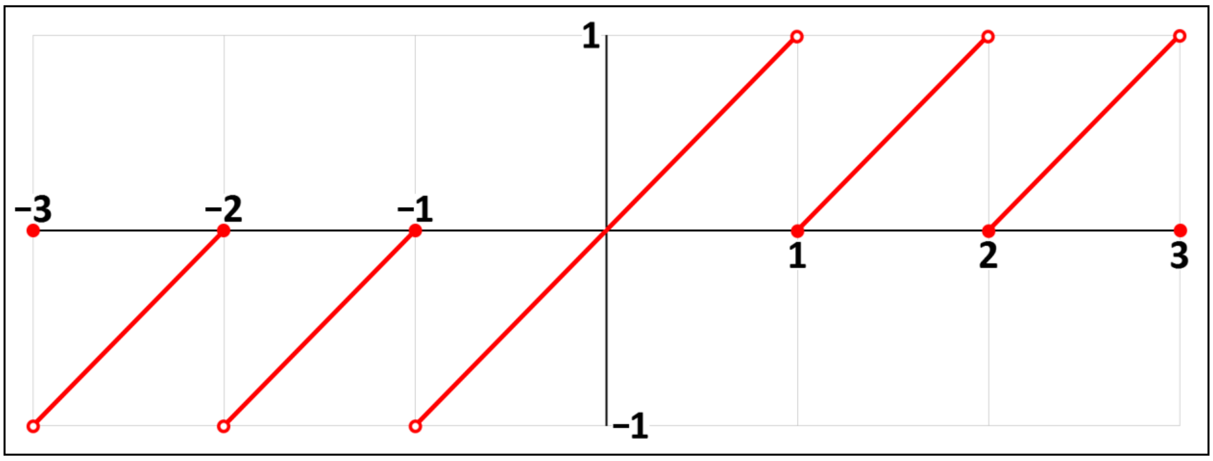
This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains an integer value.

Flags Affected:

C	Z	N	V	E	O
-	◉	◉	⓪	①	◉

Fixed-point Fractional Part of X

FRAC {x} ⊠.□



The value in the X register is replaced with its fractional part.

Domain: ℝ Range: ℝ(-1, 1)

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value with a magnitude less than one.

Flags Affected:

C	Z	N	V	E	O
-	◉	◉	⓪	①	⓪

8.4.2.4 Floating-Point Number Coding Type

<u>All Display Modes</u>

Basic Operations

Value/Components Display Switch

V⇄CPN

This key toggles the display mode between Floating-point Word Value and Floating-point Components.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The X register contains a floating-point NaN or Pseudo class bit pattern
- The X register contains a floating-point Infinity class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- Any register on the register stack other than the X register contains a value which is not exact-zero
- There are currently no loaded composites

Flags Affected:

Z	N	D	E
◉	-	◉	◉

Random Value [0, 1)

RAND 0.???

Keyboard Shortcut: ?

The value in the X register is replaced with a random value in the range [0, 1). The result will be an exact value.

This key will be disabled under any of the following circumstances:

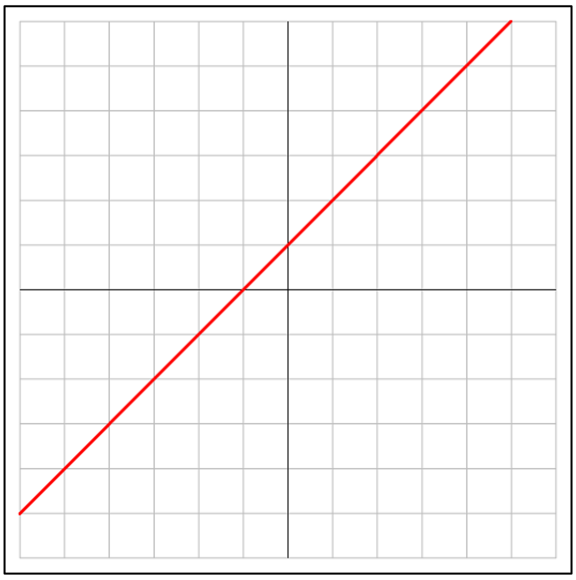
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

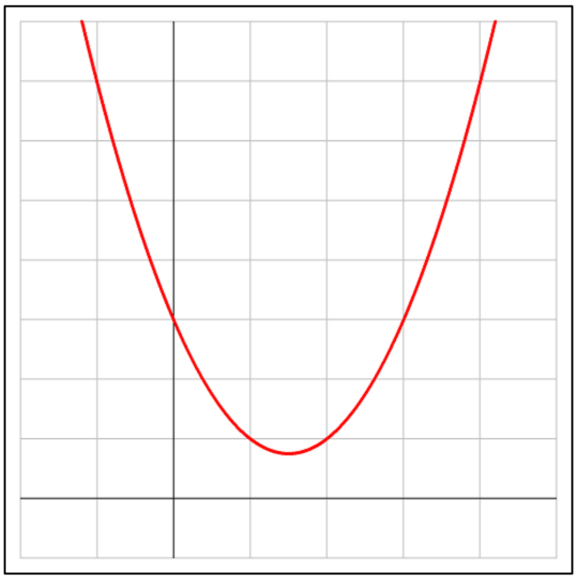
Flags Affected:

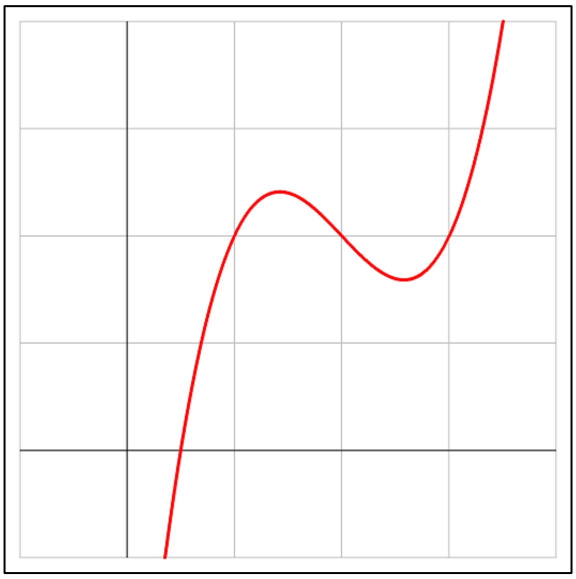
Z	N	D	E
◉	⓪	◉	①

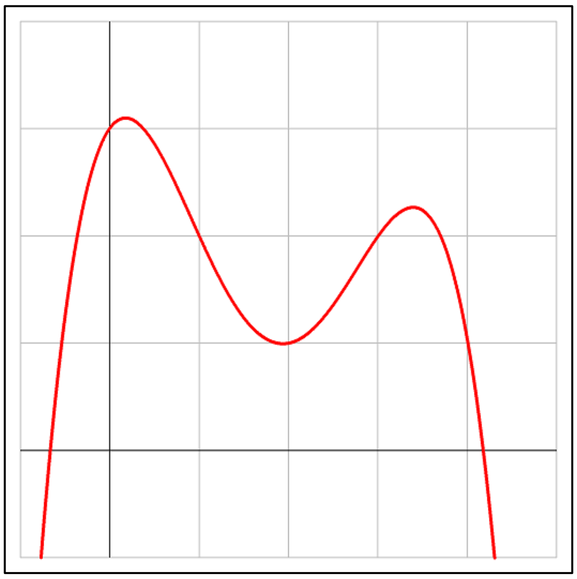
Polynomial of X

POLY $\Sigma M_n X^n$


$M_0 = 1 : M_1 = 1 : M_2$, M3, $M_4 = 0$


$M_0 = 3 : M_1 = -3 : M_2 = 1 : M_3, M_4 = 0$


$M_0 = -4.4 : M_1 = 11.733 : M_2 = -6.4 :$
$M_3 = 1.066 : M_4 = 0$


$M_0 = 3 : M_1 = 1.166 : M_2, = -3.75 :$
$M_3, = 1.833 : M_4 = -0.25$

Where M_n is the value currently stored in memory register n,

this function evaluates the polynomial ($M_4X^4 + M_3X^3 + M_2X^2 + M_1X + M_0$) and stores the resulting value in the X register.

This is the only IO operation that uses the contents of memory registers other than the currently selected memory register. Which memory register is currently selected does not affect this operation.

Note that the setting of the “Value of Zero to the power of Zero” calculator operation option is ignored during this calculation. If the X register contains a value of zero then the result will be the value stored in M_0, which is equivalent to always treating 0^0 as 1.

Domain: ℝ

Range: ℝ (a subset of ℝ if the highest index with a non-zero coefficient is even)

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and any memory register contains a floating-point NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Logarithm to Base X of Y

LOG$_X$Y

When this operator is applied the result is the power to which the value in the X register needs to be raised to in order to obtain the value in the Y register. In other words, the value originally contained in the X register defines a function (the logarithm with a base of that value), and the result is obtained by applying that function to the value originally contained in the Y register.

Base (original value in X register)	**Domain (valid original values in Y register)**	**Range**	**Notes**
$X > 1$, $X \in \mathbb{R}(1, +\infty)$	$Y > 0$, $Y \in \mathbb{R}(0, +\infty)$	ℝ	
$X = 1$	{ 1 }	{ 1 }	(designated result)
$0 < X < 1$, $X \in \mathbb{R}(0, 1)$	$Y > 0$, $Y \in \mathbb{R}(0, +\infty)$	ℝ	

$X = 0$, $0^0 \neq 1$
$0^0 = 1$
	{ 0 }
	{ 0, 1 }
		{ 1 }
		{ 0, 1 }
			$f(0) = 1$ (designated result)
			$f(0) = 1$, $f(1) = 0$

$-1 < X < 0$, $X \in \mathbb{R}(-1, 0)$	$Y \in \{ y : y = X^n, n \in \mathbb{Z} \}$	ℤ	
$X = -1$	{ -1, 1 }	{ 0, 1 }	$f(-1) = 1$ (designated result) $f(1) = 0$ (designated result)
$X < -1$, $X \in \mathbb{R}(-\infty, -1)$	$Y \in \{ y : y = X^n, n \in \mathbb{Z} \}$	ℤ	

Properties of Logarithm Function defined by Base (original value in X register)

A designated result is the value returned by IO when there are an unlimited number of “correct” results. In other words, the logarithmic mapping is not a true function, specifying a designated result makes it a function. The current setting of the “Value of 0 to the power of 0” Calculator Operation option determines whether or not $0^0 = 1$.

Calculation:

$$\log_X(Y) = \frac{\log_2(Y)}{\log_2(X)}, \quad X > 0, Y > 0$$

In the following graph:

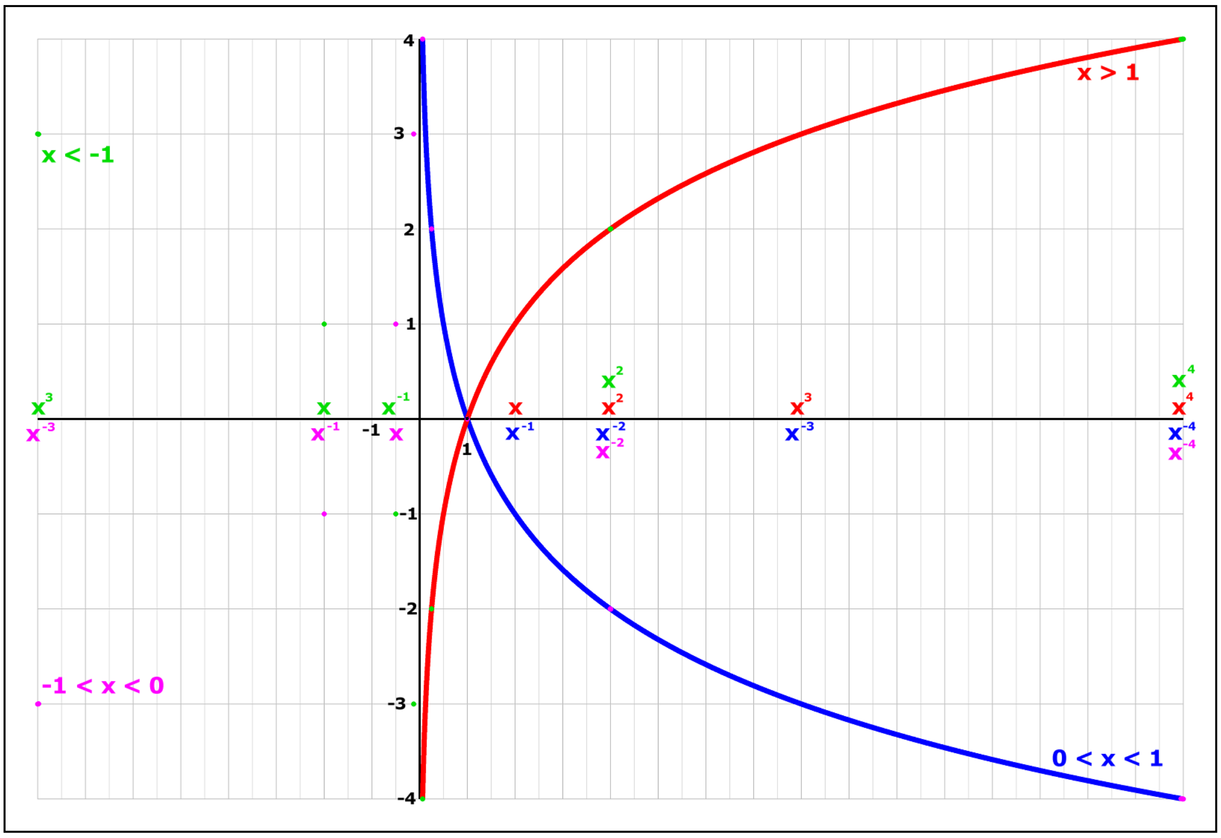
The colors of the plots identify the results for different ranges of the logarithm base, i.e., the value originally stored in the X register.

The horizontal axis identifies the value originally stored in the Y register, i.e., the value of the argument of the logarithm function. This is labeled in terms of the base of the logarithm function, i.e., the value originally stored in the X register.

The vertical axis identifies the result of the operation, i.e., the value finally stored in the X register.

The vertical grid lines shown are as they would be for bases (original contents of the X register) of ±2 or ±0.5.

The results for bases of 0, 1 and -1 are not shown.



Note the following:

If the Y register originally contains a value of 1 then the result is always 0, irrespective of the value of the base, with the following exception: if the value of 0 to the power of 0 is not configured as 1 in the Calculator Operation options then $\log_0(1)$ will be a NaN.

Special Cases:

X

		NaN/Pseudo	-Inf	-0	0	+0	+Inf
	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN					
	-0	QNaN					
Y	0	QNaN					
	+0	QNaN					
	+Inf	QNaN					

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The algebraic logic operator stack is full when the operator needs to be pushed onto it

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value of exact-zero

Flags affected when operator applied:

Z	N	D	E
◉	◉	◉	◉

Xth Root of Y

$\sqrt[x]{y}$ $y^{1/x}$ ROOT

When this operator is applied the value in the Y register is raised to the power of the reciprocal of the value in the X register to provide the result.

Calculation:

$$\sqrt[X]{Y} = 2^{\frac{\log_2(Y)}{X}}, \quad X \neq 0, Y > 0$$

Special Cases:

		X					
		NaN/Pseudo	-Inf	-0	0	+0	+Inf
	NaN/Pseudo	QNaN	QNaN	QNaN	QNaN	QNaN	QNaN
	-Inf	QNaN					
	-0	QNaN					
Y	0	QNaN					
	+0	QNaN					
	+Inf	QNaN					

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation”, the Y Register contains a negative value and the X register contains a value which is non-integer or even
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the X register contains a negative even integer value

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and the Y register contains a NaN or Pseudo class bit pattern

Flags affected when operator applied:

Z	N	D	E
-	-	◉	◉

Consolidate

CONS

The value in the X register is replaced with the value represented by the currently displayed digits.

Note that if the value in the X register is an inexact result but the value displayed can be represented exactly then the value in the X register will be replaced with this exact value and the Exact flag will be set to ①.

This key will be disabled under any of the following circumstances:

- Floating-point Word Value display mode and the “Display Format” Number Display option is set to “Binary General” or “Binary Exponent”
- Floating-point Word Bit Pattern display mode
- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- Number entry is in progress

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

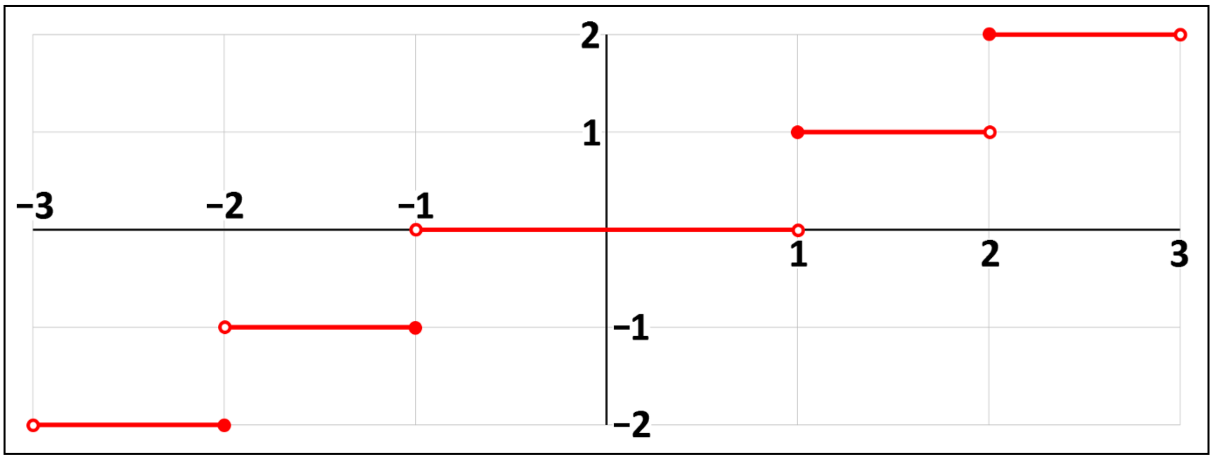
- The X register contains a NaN, Infinity or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
◉	-	◉	◉

Floating-point Truncate to Integer

TRUNC TRUNI [x] □.⊠



The value in the X register is replaced with its truncated integer part.

Domain: ℝ Range: ℤ

Exact Results:

Many

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-Inf	0	0	0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains an integer value

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

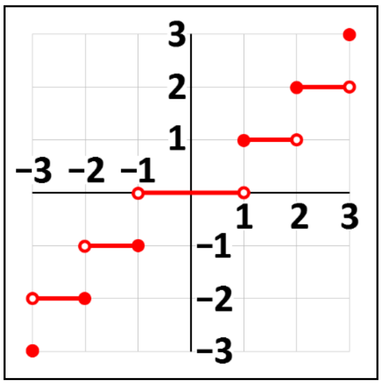
- The X register contains a NaN, Infinity or Pseudo class bit pattern

Flags Affected:

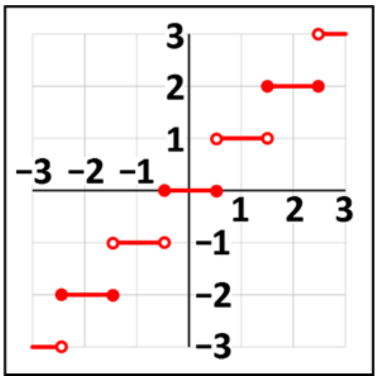
Z	N	D	E
◉	◉	◉	-

Floating-point Round to Integer

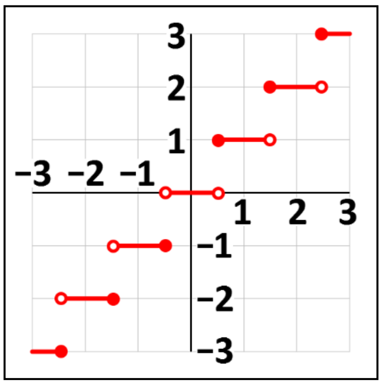
ROUND ROUNI ⌊x⌉ ▭.⊟



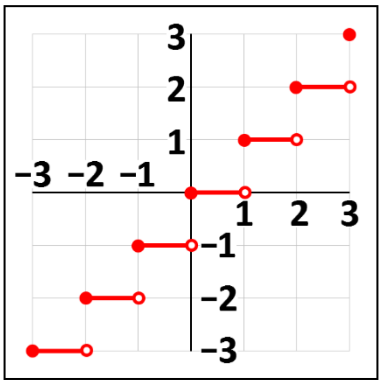
To Zero (Truncate)



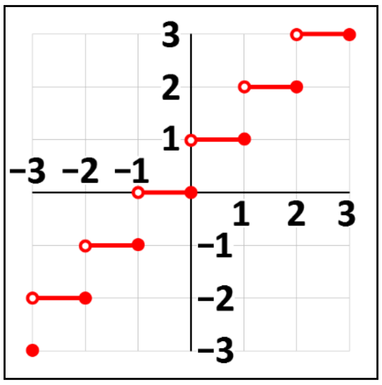
To Nearest (Even)



To Nearest (Away From Zero)



To Negative Infinity (Down)



To Positive Infinity (Up)

If the “Always use To Nearest (Away From Zero) for Round To Integer Ops” Number Coding option is selected then the value in the X register is replaced with the integer value obtained by applying the To Nearest (Away From Zero) rounding to its original value.

Otherwise, the value in the X register is replaced with the rounded integer value of its original value, which is obtained by applying the currently selected Fixed/Floating-point Rounding Type in the Number Coding options.

Domain: ℝ Range: ℤ

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains an integer value

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN, Infinity or Pseudo class bit pattern

Special Cases:

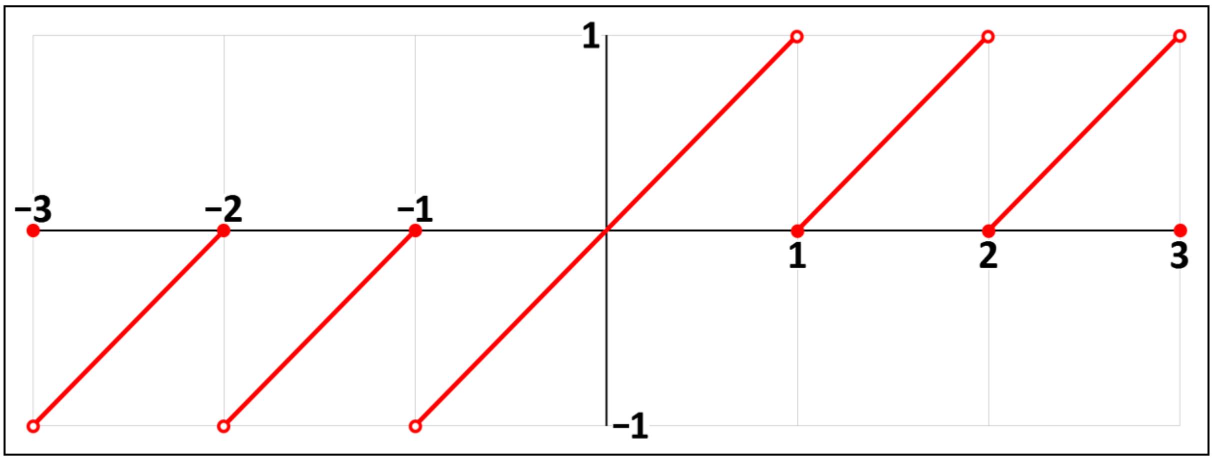
X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result for Round Up	QNaN	-Inf	0	0	+1	+Inf
Result for Round Down	QNaN	-Inf	-1	0	0	+Inf
Result otherwise	QNaN	-Inf	0	0	0	+Inf

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	-

Floating-point Fractional Part

FRACT {x} ⊠.▭



The value in the X register is replaced with its fractional part.

Domain: ℝ Range: ℝ(-1, 1)

Exact Results:

Many

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-0	0	+0	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value with a magnitude less than one.

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN, Infinity or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
◉	◉	-	-

Constant Operations

Although these can be thought of as functions as they identify a value for any value currently contained in the X register, the value identified is the same for all values of X so graphs are not provided as they would all simply be a horizontal line through the value of the constant on the Y axis.

Pi

`PI` `pi` `π` `π`

The value in the X register is replaced with the value closest to π possible using the current floating-point coding and Fixed/Floating-point Rounding Type currently selected in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	⓪

Minimum Denormal Value

`MIND` `⊻⊻`

The value in the X register is replaced with the smallest positive denormal value possible for the current number coding.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	①	①

Minimum Normal Value

`MINN` `⊻`

The value in the X register is replaced with the smallest positive normal value possible for the current number coding.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	①

Epsilon Value

EPS **ε**

The value in the X register is replaced with the epsilon value for the current number coding.

The epsilon value is the smallest positive value that can be added to a value of one that produces an exact result.

Multiplying an inexact value by the epsilon value for the number coding being used gives a very good approximation to the maximum difference from the exact result, i.e., the value represented by the lsb of the stored significand.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	①

Maximum Consecutive Integer Value

MAXI **⊼**

The value in the X register is replaced with the integer value n where ℕ[1, n] is the largest set of positive integer values which may be represented exactly using the current number coding, i.e., all positive integer values less than or equal to n can be represented exactly, but (n + 1) cannot. No non-integer value greater than n will be able to be represented exactly.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	①

Maximum Normal Value

MAXN

The value in the X register is replaced with the largest positive normal value possible for the current number coding.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	①

Logarithmic and Exponential Operations

The logarithm to a particular base of a number is the power to which the value of the base must be raised to obtain the number. Note that the term “base” used in this context has a different meaning to the term “base” described in the context of the textual representation of a number; they just use the same word.

A logarithmic function “$\log_b$“ maps a value to its logarithm to base b.

So,

$$b^{\log_b(x)} = x$$

IO provides the logarithm functions for base values of 2, e and 10. It also provides a logarithm operator which allows you to specify a more general value for the base. For positive bases the logarithm functions are only defined for positive argument values.

The logarithm to base 10 is often called the “common logarithm”.

The logarithm to base e is often called the “natural logarithm” and has the notation “ln” as well as “$\log_e$”.

Each logarithmic function is paired with an exponential function which provides the inverse mapping.

An exponential function “$\exp_b$“ maps a value to b raised to the power of the value.

So,

$$\exp_b(x) = b^x$$

and,

$$\log_b(\ \exp_b(x)\ ) = x, \qquad \exp_b(\ \log_b(x)\ ) = x$$

The value of an exponential function applied to an argument value is sometimes referred to as the “antilogarithm” or “antilog” of the argument value.

IO provides the exponential functions for base values of 2, e and 10. It also provides the operator y^x which allows you to specify a general value for the base.

When some texts refer to “the exponential function”, abbreviated to “exp”, they assume that the base is e, i.e., the value of the function for an argument of x is e^x.

The reason e is important enough to warrant its own special terminology stems from some results in calculus.

In differential calculus,

$$\frac{d}{dx}\, e^x = e^x$$

In integral calculus,

$$\int_1^x \frac{1}{t}\,dt = \log_e(x)$$

These results have many implications which cause e to turn up all over the place in mathematics.

The definitions of the hyperbolic functions, which are described later, are all related to e.

The following power series can be used to evaluate the $\log_e$ and $\exp_e$ functions:

$$\log_e(1+x) = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots$$

where $x \in \mathbb{R}(-1, 1]$

and,

$$\exp_e(x) = \sum_{n=0}^{\infty}\frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

The basic definitions of the logarithm functions lead to a number of logarithmic identities, i.e., formulas which apply to all values for which the functions are defined. e.g.

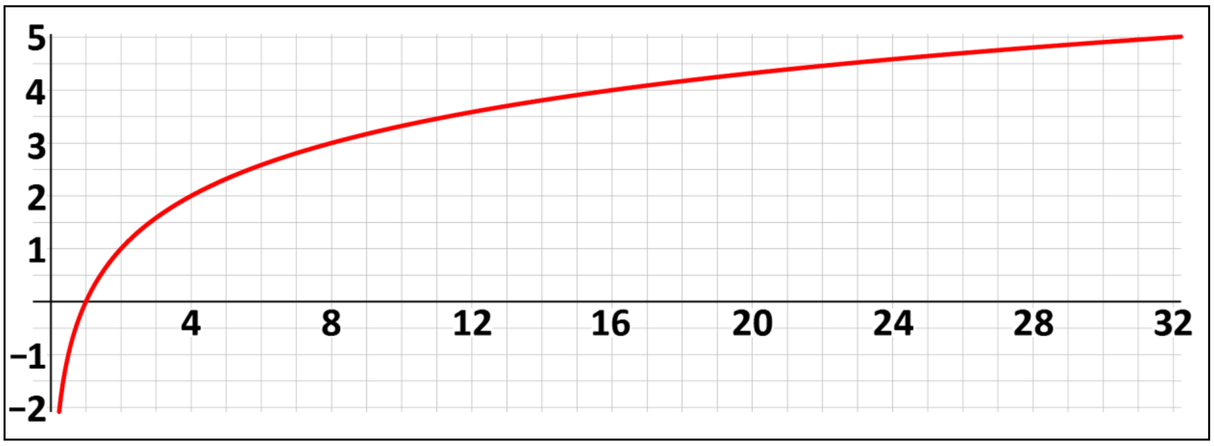
$$\log_b(xy) = \log_b(x) + \log_b(y) \qquad \log_b(x/y) = \log_b(x) - \log_b(y)$$

$$\log_b(x^y) = y\log_b(x) \qquad \log_b\left(\sqrt[y]{x}\right) = \frac{\log_b(x)}{y}$$

In the days before calculators were commonplace these identities were used together with printed tables of logarithms (usually to base 10) and exponentials (usually called anitlogs in those days) to perform calculations that would have been extremely time-consuming to perform otherwise.
For example, to calculate the 6th root of 13 you would get $\log_{10}(13)$ using the logarithm table, divide this value by 6, then look up the result of the division in the antilog table to get the final answer.

Logarithm to Base 2 of X

LOG₂



The value in the X register is replaced with its logarithm to base 2.

Domain: ℝ(0, ∞) Range: ℝ

Exact Results:

$\log_2(1) = 0$
16-bit Floating-point: $\log_2(2^n)$, n ∈ ℤ[-24, +15]
32-bit Floating-point: $\log_2(2^n)$, n ∈ ℤ[-149, +127]
64-bit Floating-point: $\log_2(2^n)$, n ∈ ℤ[-1074, +1023]
80-bit Floating-point: $\log_2(2^n)$, n ∈ ℤ[-16445, +16383]

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	QNaN	-Inf	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

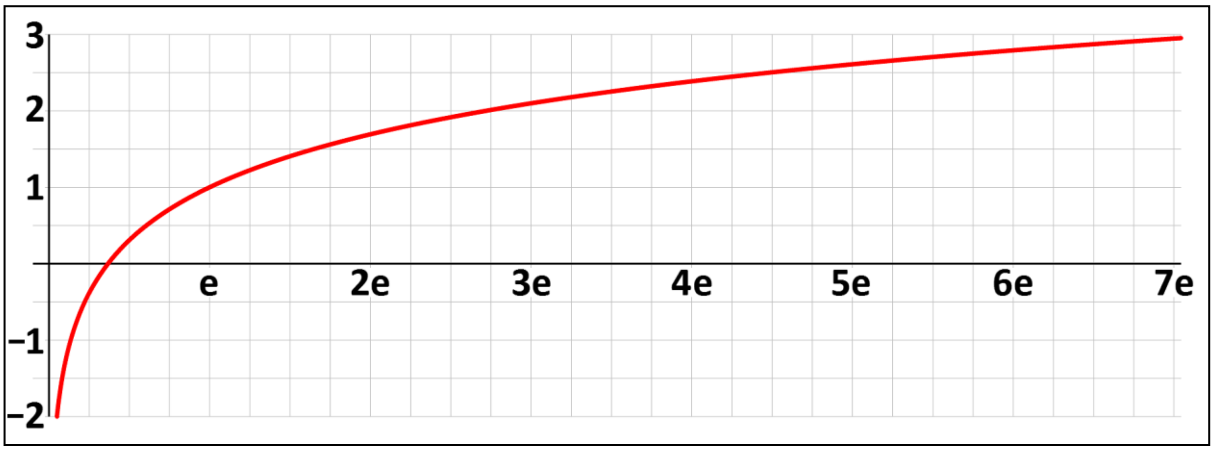
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a negative value or a value of exact-zero

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Logarithm to Base e of X

LOGe LN ln



The value in the X register is replaced with its logarithm to base e.

Calculation:

$$\log_e(X) = \frac{\log_2(X)}{\log_2(e)}$$

Domain: ℝ(0, ∞) Range: ℝ

Exact Results:

$\log_e(1) = 0$

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	QNaN	-Inf	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

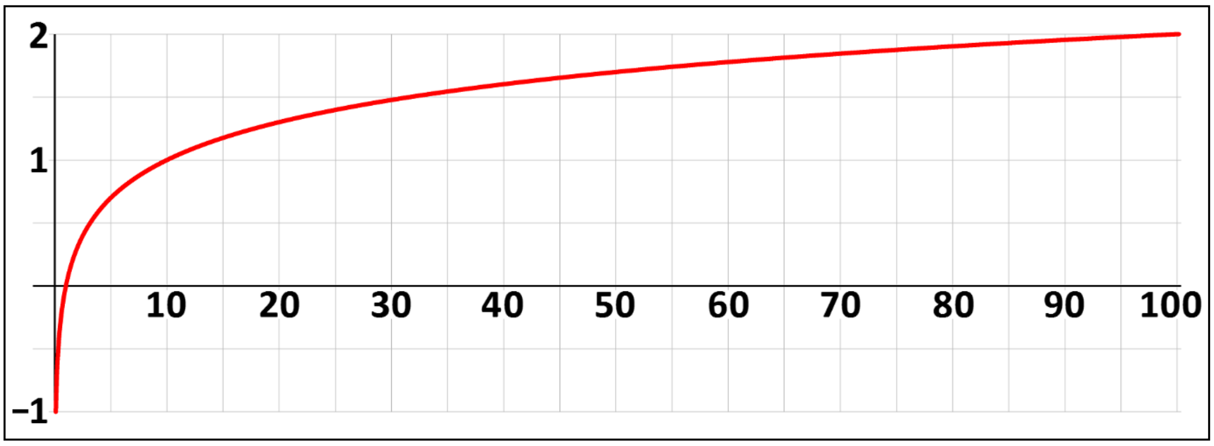
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a negative value or a value of exact-zero

Flags Affected:

Z	N	D	E
◉	◉	◉	◉

Logarithm to Base 10 of X

LOG10



The value in the X register is replaced with its logarithm to base 10.

Calculation:

$$\log_e(\mathrm{X}) = \frac{\log_2(\mathrm{X})}{\log_2(10)}$$

Domain: ℝ(0, ∞) Range: ℝ

Exact Results:

$\log_{10}(1) = 0$
16-bit Floating-point: $\log_{10}(10^n)$, n ∈ ℕ[0, 4]
32-bit Floating-point: $\log_{10}(10^n)$, n ∈ ℕ[0, 10]
64-bit Floating-point: $\log_{10}(10^n)$, n ∈ ℕ[0, 22]
80-bit Floating-point: $\log_{10}(10^n)$, n ∈ ℕ[0, 27]

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	QNaN	-Inf	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

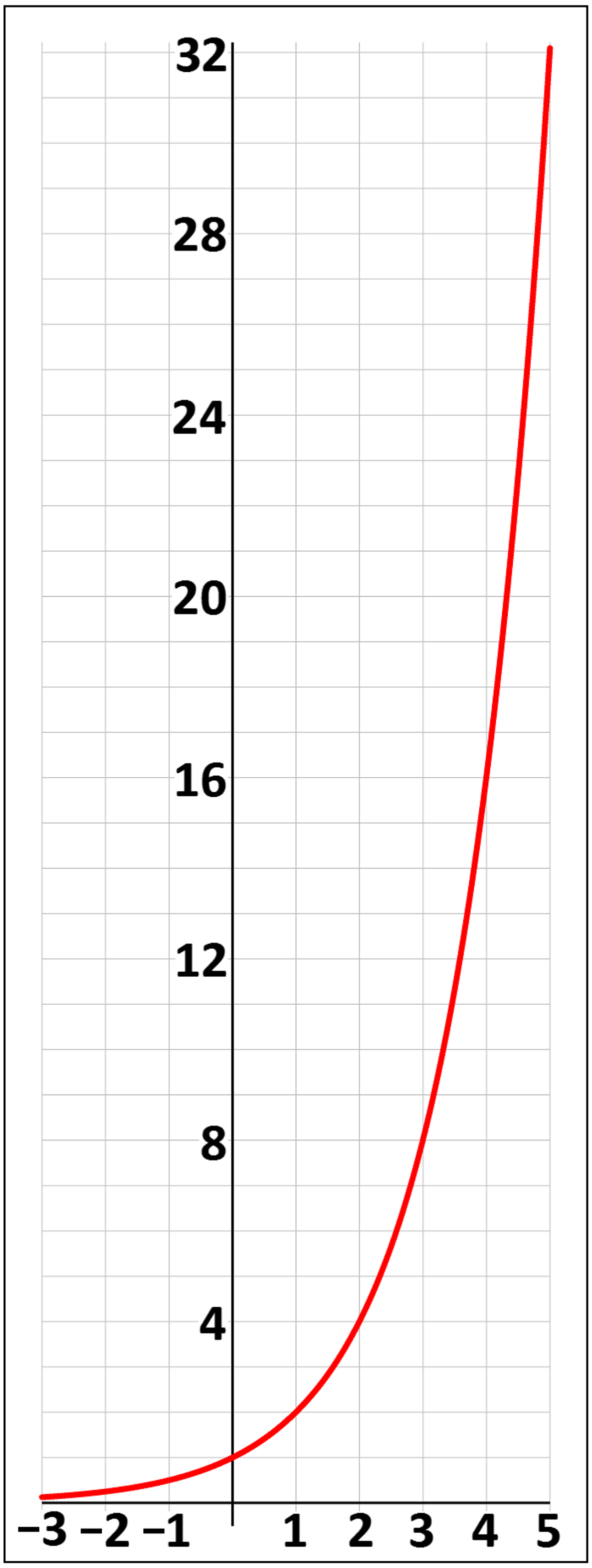
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a negative value or a value of exact-zero

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

2 to the Power of X

2^X | EXP$_2$ | ALOG$_2$



The value in the X register is replaced with the value of 2 raised to the power of its original value.

Domain: ℝ

Range: ℝ(0, ∞)

Exact Results:

$2^0 = 1$
16-bit Floating-point: 2^n, n ∊ ℤ[-24, +15]
32-bit Floating-point: 2^n, n ∊ ℤ[-149, +127]
64-bit Floating-point: 2^n, n ∊ ℤ[-1074, +1023]
80-bit Floating-point: 2^n, n ∊ ℤ[-16445, +16383]

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+0	+ie1	+1	+ie1	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

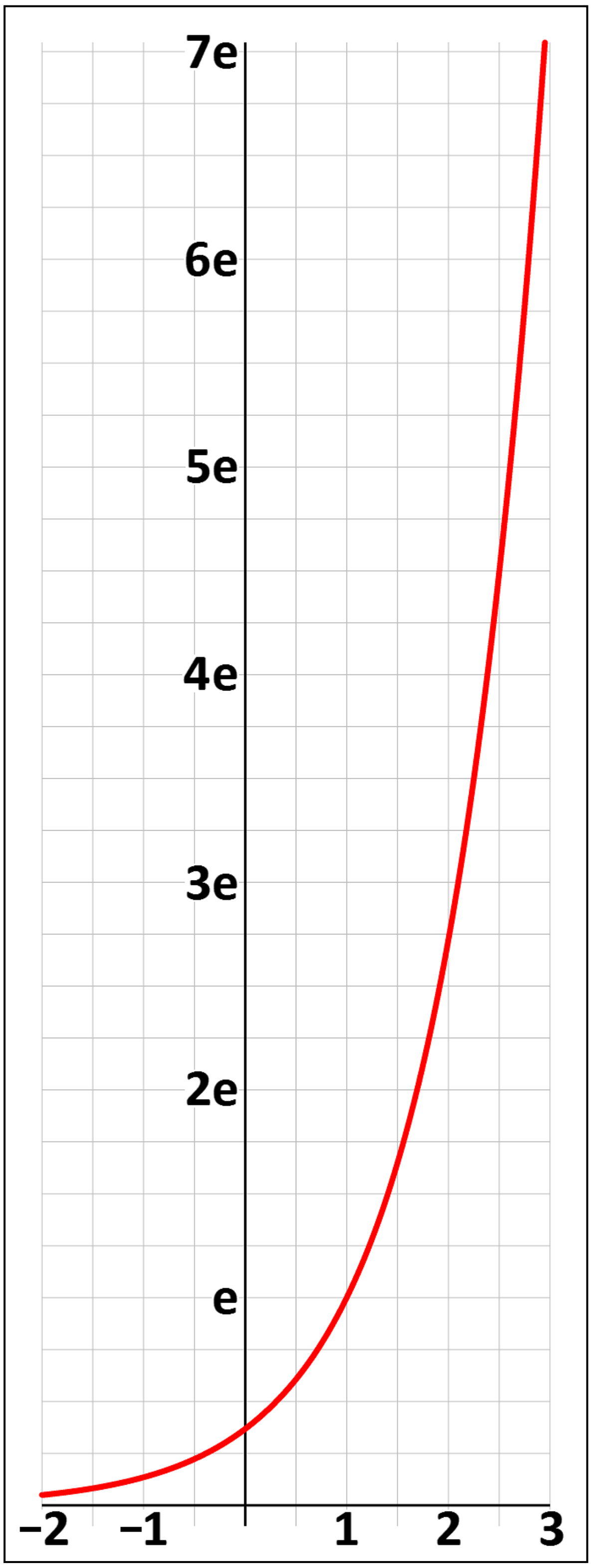
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⓪	⓪	◉	◉

e to the Power of X

e^X EXPe ALOGe



The value in the X register is replaced with the value of e raised to the power of its original value.

Domain: ℝ

Range: ℝ(0, ∞)

Exact Results:

$e^0 = 1$

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+0	+ie1	+1	+ie1	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

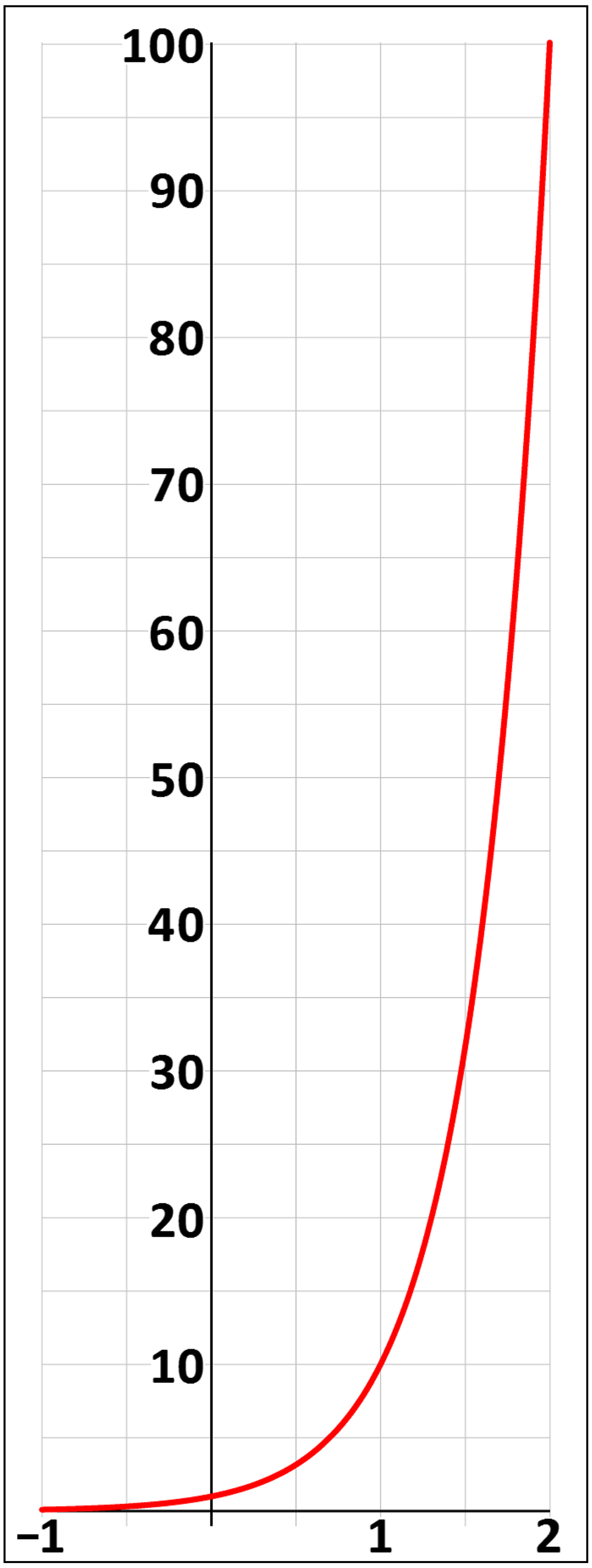
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⓪	⓪	◉	◉

10 to the Power of X

10^X | EXP_{10} | $ALOG_{10}$



The value in the X register is replaced with the value of 10 raised to the power of its original value.

Domain: ℝ

Range: ℝ(0, ∞)

Exact Results:

$10^0 = 1$
16-bit Floating-point: 10^n, n ∈ ℕ[0, 4]
32-bit Floating-point: 10^n, n ∈ ℕ[0, 10]
64-bit Floating-point: 10^n, n ∈ ℕ[0, 22]
80-bit Floating-point: 10^n, n ∈ ℕ[0, 27]

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+0	+ie1	+1	+ie1	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern

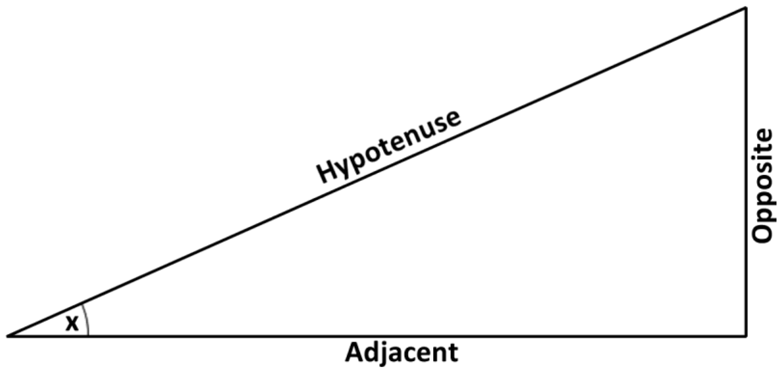
Flags Affected:

Z	N	D	E
⓪	⓪	◉	◉

Trigonometric Operations

The trigonometric functions are a family of functions that are commonly encountered in many areas of mathematics and engineering. IO has a key for each of the twelve supported trigonometric functions.

When students are introduced to the trigonometric functions, they are usually defined in a geometric context as the ratios of sides of a right-angled triangle where the argument of the function is one of the other two interior angles of the triangle.



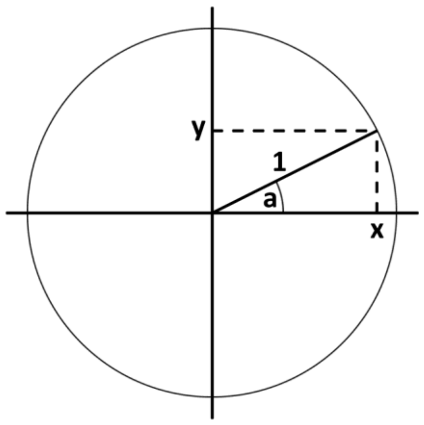
The sine function applied to a value x is defined as the ratio of the lengths of “Opposite” to “Hypotenuse” (Opposite / Hypotenuse) as depicted above. This ratio is the same irrespective of the size of the triangle. Similarly, the cosine function is defined as the ratio of “Adjacent” to “Hypotenuse” and the tangent function is defined as the ratio of “Opposite” to “Adjacent”.

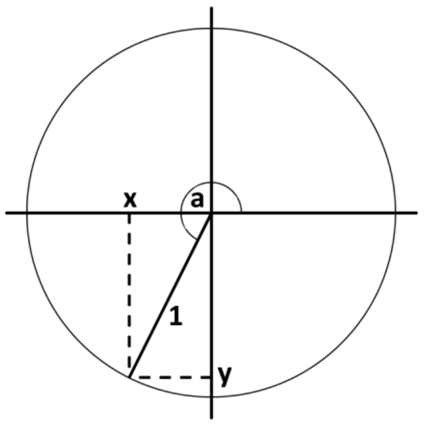
These definitions imply that the sine of x is equal to the cosine of (90˚ – x) and the cosine of x is equal to the sine of (90˚ – x), simply by considering the other interior angle of the triangle, and that the tangent of x is equal to the sine of x divided by the cosine of x.

The functions sine, cosine and tangent are usually abbreviated to sin, cos and tan, and their arguments are not usually enclosed in parentheses.

Note that a trigonometric function is different for different units of measurement of x, e.g., sin 1 where the 1 represents 1 degree is a different value from sin 1 where the 1 represents 1 gradian, therefore they are different functions, i.e., different mappings from ℝ to ℝ, but they share the same name.

Although these definitions are easy to understand, they do have the major restriction that they can only be applied to values of x in the range ℝ(0, 90), assuming that x is measured in degrees.





To overcome this restriction, consider a circle with a radius of 1 drawn over X and Y axes with its center at the origin. Now draw a straight-line segment from the origin and at an angle of *a* with the X axis (*a* may be positive or negative) to the point where it intersects the circle. The sine of *a* is now defined as the y coordinate of where the line intersects the circle. The cosine of *a* is now defined as the x coordinate of where the line intersects the circle. The tangent is now defined as the y coordinate divided by the x coordinate. As *a* can be any value, the functions are defined for any value (except those that would require a division by 0, e.g., tan 90˚).

The first diagram above illustrates that when *a* is in the first quadrant (0 < a < 90˚), x and y are both positive so the sine, cosine and tangent of *a* are all positive. The second diagram illustrates that when *a* is in the third quadrant (180˚ < a < 270˚), x any y are both negative so the sine and cosine of *a* are negative and the tangent of *a* is positive.

The trigonometric functions are sometimes referred to as the “circular functions”, this is because the points defined by the coordinates (cos *a*, sin *a*) trace out a circle as the value of *a* increases from 0.

It is obvious that as *a* increases past 360˚ the values of the trigonometric functions start repeating their values from what they are for 0. i.e., sin/cos/tan *a* = sin/cos/tan(*a* mod 360˚).

Unfortunately, these definitions do not help us to calculate the values of any of the trigonometric functions for any particular argument value. Drawing a diagram, measuring the lengths of lines with a ruler, and then maybe dividing the lengths is not really good enough. Fortunately, it can be proved (but not here because the level of mathematics required is beyond the scope of this document) that:

$$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

and

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

If the argument X in the above definitions is interpreted as the angle *a* in one of the above diagrams, then the angle must be measured in radians. This will be equal to the length of the arc of the unit circle shown in the diagrams subtended by the angle *a*.

These definitions can be used to calculate the numerical values of any of the trigonometric functions for any argument value.

For each of the three trigonometric functions already identified, there is an additional trigonometric function defined by taking the reciprocal of the value for the corresponding function. These functions are called cosecant, secant and cotangent.

The cosecant of x is defined as the reciprocal of the sine of x, and is abbreviated to csc (or cosec).

The secant of x is defined as the reciprocal of the cosine of x, and is abbreviated to sec.

The cotangent of x is defined as the reciprocal of the tangent of x, and is abbreviated to cot.

Note the relationship between the names; pairs of functions with and without the “co” prefix are related by the condition that the value of one when applied to a value x is equal to the value of the other when applied to a value of (90˚ – x) or (π /2 – x) radians. In the geometric definitions, the value of one of the pair applied to on angle of a right-angled triangle is the value of the other applied to the other angle.

Each of these six trigonometric functions has an inverse function which maps the value of one of the functions for a particular argument back to the value of the original argument. Because the six trigonometric functions repeat their values for each 360˚ (2π radians) the inverse functions have a range specified so that a unique value is defined as the value of the function for a particular argument.

The inverse trigonometric functions are often named by prefixing "arc" to the name of the corresponding trigonometric function, e.g., the inverse function of the tangent function is usually called arctangent, and is abbreviated to arctan or just atan, the notation $\tan^{-1}$ is also commonly used. These conventions are used for all the inverse trigonometric functions.

Another notation commonly used with the trigonometric functions (we will use sin in our example) is $\sin^n x$, where $n \in \mathbb{N}[2, \infty)$ to mean $(\sin x)^n$. Note that this is inconsistent with the $^{-1}$ notation used to denote an inverse function.

In summary:

Function	Equivalences	
$\sin(x)$	$\cos\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\csc(x)}$
$\cos(x)$	$\sin\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\sec(x)}$
$\tan(x)$	$\cot\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\cot(x)}$
$\csc(x)$	$\sec\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\sin(x)}$
$\sec(x)$	$\csc\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\cos(x)}$
$\cot(x)$	$\tan\left(\frac{\pi}{2} - x\right)$,	$\frac{1}{\tan(x)}$

The definitions of the trigonometric functions lead to the existence of many trigonometric "identities" (formulas which are true for all values), the most important being:

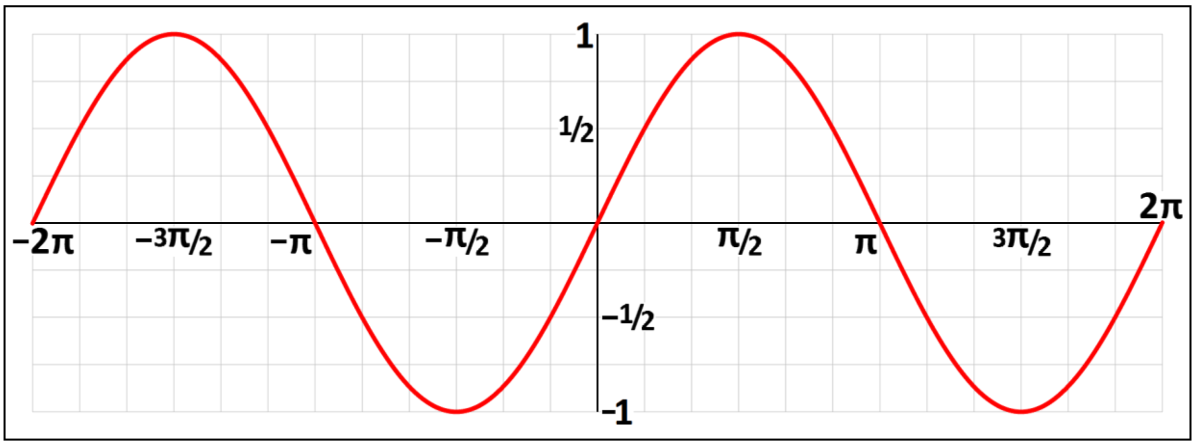
$$\sin^2 x + \cos^2 x = 1$$

In the following graphs for the trigonometric functions the X-axis values are labeled in units of radians, as are Y-axis values for the inverse trigonometric functions. Similarly, the domains of the trigonometric functions and the ranges of the inverse trigonometric functions are specified in units of radians.

Note that IO can only evaluate the results of the trigonometric functions for argument values in the range $(-2^{63}, +2^{63})$ radians. Argument values outside this range will return a QNaN result, even though the mathematical function may be defined for these values.

Sine of X

SIN



The value in the X register is replaced with its sine. The original value in the X register is interpreted as a number of units as currently defined by the “Trig Func Units” Calculator Operation option.

Geometric Definition: sin(x) = Opposite / Hypotenuse

Domain: ℝ Range: ℝ[-1, 1]

Exact Results:

All Trig Units
sin(0) = 0

Degrees

if (x / 180) ∊ ℤ,	then sin(x) = 0
if ((x – 30) / 360) ∊ ℤ,	then sin(x) = 0.5
if ((x – 90) / 360) ∊ ℤ,	then sin(x) = 1
if ((x – 150) / 360) ∊ ℤ,	then sin(x) = 0.5
if ((x – 210) / 360) ∊ ℤ,	then sin(x) = -0.5
if ((x – 270) / 360) ∊ ℤ,	then sin(x) = -1
if ((x – 330) / 360) ∊ ℤ,	then sin(x) = -0.5

Gradians

if (x / 200) ∊ ℤ,	then sin(x) = 0
if ((x – 100) / 400) ∊ ℤ,	then sin(x) = 1
if ((x – 300) / 400) ∊ ℤ,	then sin(x) = -1

Right Angles

if (x / 2) ∊ ℤ,	then sin(x) = 0
if ((x – 1) / 4) ∊ ℤ,	then sin(x) = 1
if ((x – 3) / 4) ∊ ℤ,	then sin(x) = -1

Circles

if 2x ∊ ℤ,	then sin(x) = 0
if (x – 0.25) ∊ ℤ,	then sin(x) = 1
if (x – 0.75) ∊ ℤ,	then sin(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-0	0	+0	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

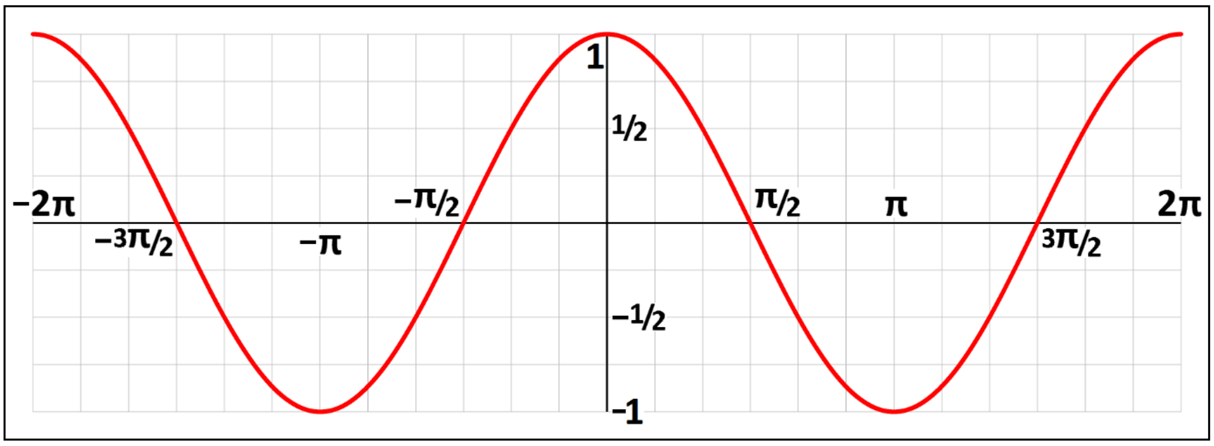
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value, when converted to radians, which is not in the range (-2^{63}, $+2^{63}$)

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Cosine of X

cos



The value in the X register is replaced with its cosine. The original value in the X register is interpreted as a number of units as currently defined by the “Trig Func Units” Calculator Operation option.

Geometric Definition: cos(x) = Adjacent / Hypotenuse

Domain: ℝ   Range: ℝ[-1, 1]

Exact Results:

<u>All Trig Units</u>
cos(0) = 1

<u>Degrees</u>

if (x / 360) ∈ ℤ,	then cos(x) = 1
if ((x – 60) / 360) ∈ ℤ,	then cos(x) = 0.5
if ((x – 90) / 180) ∈ ℤ,	then cos(x) = 0
if ((x – 120) / 360) ∈ ℤ,	then cos(x) = -0.5
if ((x – 180) / 360) ∈ ℤ,	then cos(x) = -1
if ((x – 240) / 360) ∈ ℤ,	then cos(x) = -0.5
if ((x – 300) / 360) ∈ ℤ,	then cos(x) = 0.5

<u>Gradians</u>

if (x / 400) ∈ ℤ,	then cos(x) = 1
if ((x – 100) / 200) ∈ ℤ,	then cos(x) = 0
if ((x – 200) / 400) ∈ ℤ,	then cos(x) = -1

<u>Right Angles</u>

if (x / 4) ∈ ℤ,	then cos(x) = 1
if ((x – 1) / 2) ∈ ℤ,	then cos(x) = 0
if ((x – 2) / 4) ∈ ℤ,	then cos(x) = -1

<u>Circles</u>

if x ∈ ℤ,	then cos(x) = 1
if ((x – 0.25) / 2) ∈ ℤ,	then cos(x) = 0
if (x – 0.5) ∈ ℤ,	then cos(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	+ie1	+1	+ie1	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

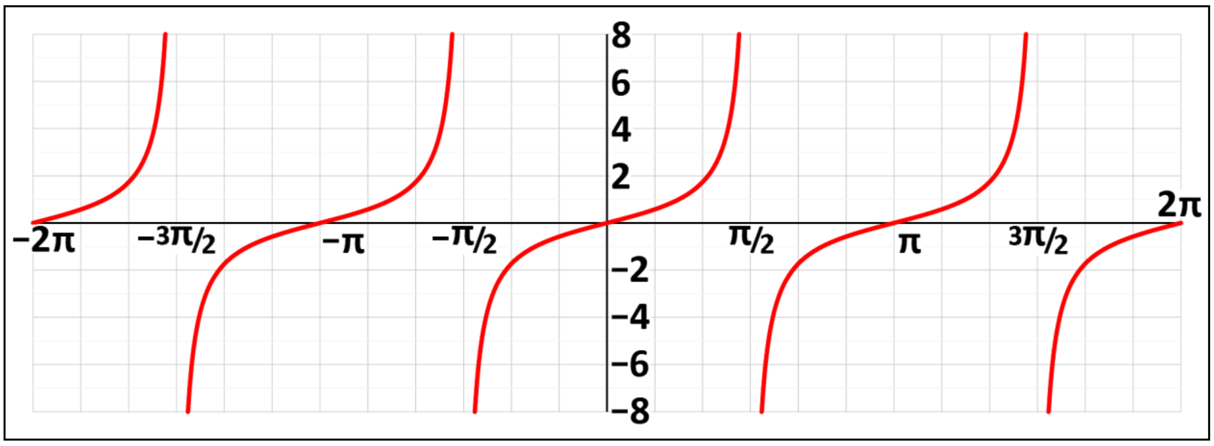
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value, when converted to radians, which is not in the range (-2^{63}, $+2^{63}$)

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Tangent of X

TAN



The value in the X register is replaced with its tangent. The original value in the X register is interpreted as a number of units as currently defined by the “Trig Func Units” Calculator Operation option.

Geometric Definition: tan(x) = Opposite / Adjacent

Domain: $\{ x \in \mathbb{R} : ((x - \pi/2) / \pi) \notin \mathbb{Z} \}$ Range: $\mathbb{R}$

Exact Results:

<u>All Trig Units</u>
tan(0) = 0

<u>Degrees</u>
if $(x / 180) \in \mathbb{Z}$, then tan(x) = 0
if $((x - 45) / 180) \in \mathbb{Z}$, then tan(x) = 1
if $((x - 135) / 180) \in \mathbb{Z}$, then tan(x) = -1

<u>Gradians</u>
if $(x / 200) \in \mathbb{Z}$, then tan(x) = 0
if $((x - 50) / 200) \in \mathbb{Z}$, then tan(x) = 1
if $((x - 150) / 200) \in \mathbb{Z}$, then tan(x) = -1

<u>Right Angles</u>
if $(x / 2) \in \mathbb{Z}$, then tan(x) = 0
if $((x - 0.5) / 2) \in \mathbb{Z}$, then tan(x) = 1
if $((x - 1.5) / 2) \in \mathbb{Z}$, then tan(x) = -1

<u>Circles</u>
if $2x \in \mathbb{Z}$, then tan(x) = 0
if $2(x - 0.125) \in \mathbb{Z}$, then tan(x) = 1
if $2(x - 0.375) \in \mathbb{Z}$, then tan(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-0	0	+0	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

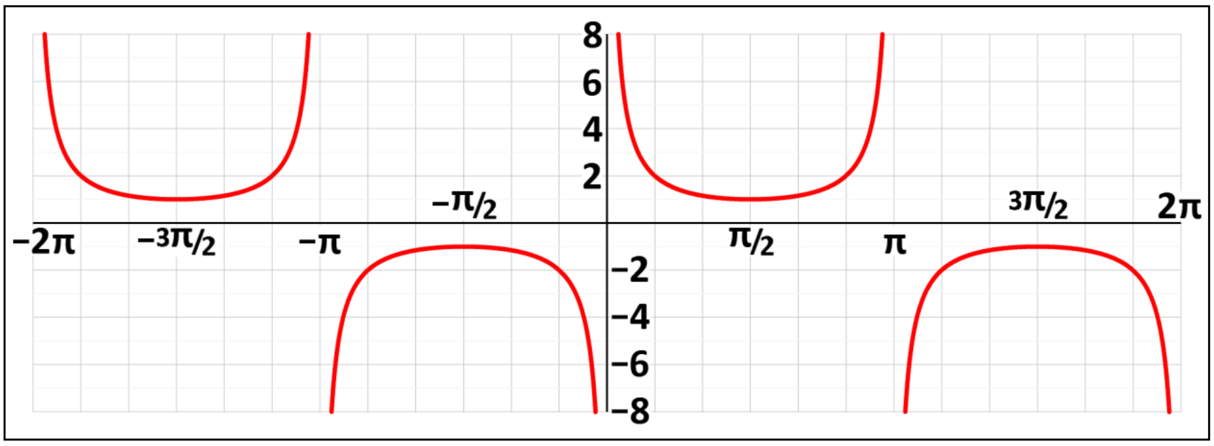
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value, when converted to radians, which is not in the range (-2^{63}, $+2^{63}$)
- The “Trig Funcs Units” Calculator Operation option is set to “Degrees”, and the X register contains a value of $(180n + 90)$ where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Gradians”, and the X register contains a value of $(200n + 100)$ where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Right Angles”, and the X register contains a value of $(2n + 1)$ where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Circles”, and the X register contains a value of $(2n + 1)/4$ where $n \in \mathbb{Z}$

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Cosecant of X

CSC



The value in the X register is replaced with its cosecant. The original value in the X register is interpreted as a number of units as currently defined by the “Trig Func Units” Calculator Operation option.

Geometric Definition: csc(x) = Hypotenuse / Opposite

Calculation:

$$\csc(X) = \frac{1}{\sin(X)}$$

Domain: { x ∈ ℝ : (x / π) ∉ ℤ } Range: { x ∈ ℝ : x ∉ ℝ(-1, 1) }

Exact Results:

Degrees
if ((x – 30) / 360) ∈ ℤ, then csc(x) = 2
if ((x – 90) / 360) ∈ ℤ, then csc(x) = 1
if ((x – 150) / 360) ∈ ℤ, then csc(x) = 2
if ((x – 210) / 360) ∈ ℤ, then csc(x) = -2
if ((x – 270) / 360) ∈ ℤ, then csc(x) = -1
if ((x – 330) / 360) ∈ ℤ, then csc(x) = -2

Gradians
if ((x – 100) / 400) ∈ ℤ, then csc(x) = 1
if ((x – 300) / 400) ∈ ℤ, then csc(x) = -1

Right Angles
if ((x – 1) / 4) ∈ ℤ, then csc(x) = 1
if ((x – 3) / 4) ∈ ℤ, then csc(x) = -1

Circles
if (x – 0.25) ∈ ℤ, then csc(x) = 1
if (x – 0.75) ∈ ℤ, then csc(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-Inf	QNaN	+Inf	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

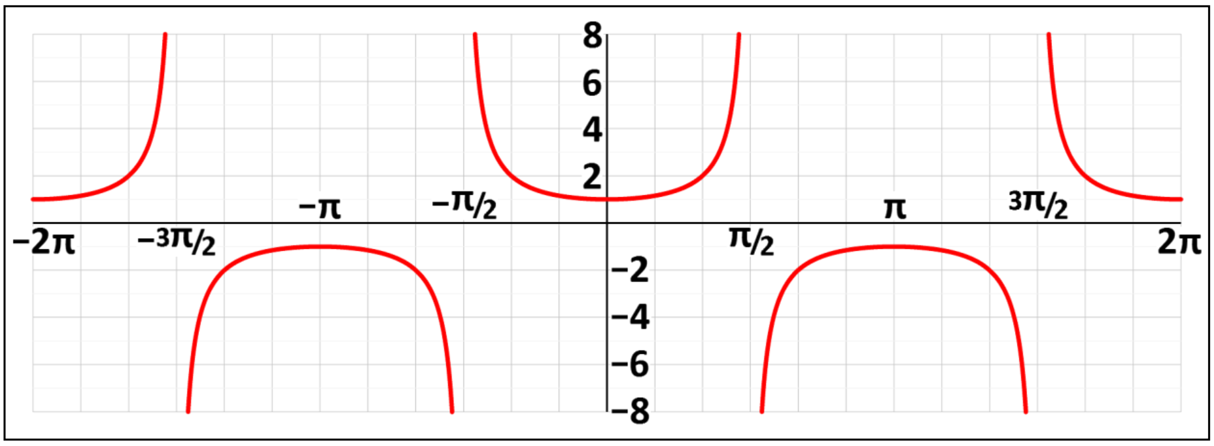
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value of exact-zero
- The X register contains a value, when converted to radians, which is not in the range (-2^{63}, $+2^{63}$)
- The “Trig Funcs Units” Calculator Operation option is set to “Degrees”, and the X register contains a value of 180n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Gradians”, and the X register contains a value of 200n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Right Angles”, and the X register contains a value of 2n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Circles”, and the X register contains a value of n/2 where $n \in \mathbb{Z}$

Flags Affected:

Z	N	D	E
⓪	-	⓪	◉

Secant of X

SEC



The value in the X register is replaced with its secant. The original value in the X register is interpreted as a number of units as currently defined by the “Trig Func Units” Calculator Operation option.

Geometric Definition: sec(x) = Hypotenuse / Adjacent

Calculation:

$$\sec(X) = \frac{1}{\cos(X)}$$

Domain: { x ∈ ℝ : ((x – π /2) / π) ∉ ℤ } Range: { x ∈ ℝ : x ∉ ℝ(-1, 1) }

Exact Results:

<u>All Trig Units</u>
sec(0) = 1

<u>Degrees</u>
if (x / 360) ∈ ℤ, then sec(x) = 1
if ((x – 60) / 360) ∈ ℤ, then sec(x) = 2
if ((x – 120) / 360) ∈ ℤ, then sec(x) = -2
if ((x – 180) / 360) ∈ ℤ, then sec(x) = -1
if ((x – 240) / 360) ∈ ℤ, then sec(x) = -2
if ((x – 300) / 360) ∈ ℤ, then sec(x) = 2

<u>Gradians</u>
if (x / 400) ∈ ℤ, then sec(x) = 1
if ((x – 200) / 400) ∈ ℤ, then sec(x) = -1

<u>Right Angles</u>
if (x / 2) ∈ ℤ, then sec(x) = 1
if ((x – 2) / 4) ∈ ℤ, then sec(x) = -1

<u>Circles</u>
if x ∈ ℤ, then sec(x) = 1
if (x – 0.5) ∈ ℤ, then sec(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	+ie1	+1	+ie1	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

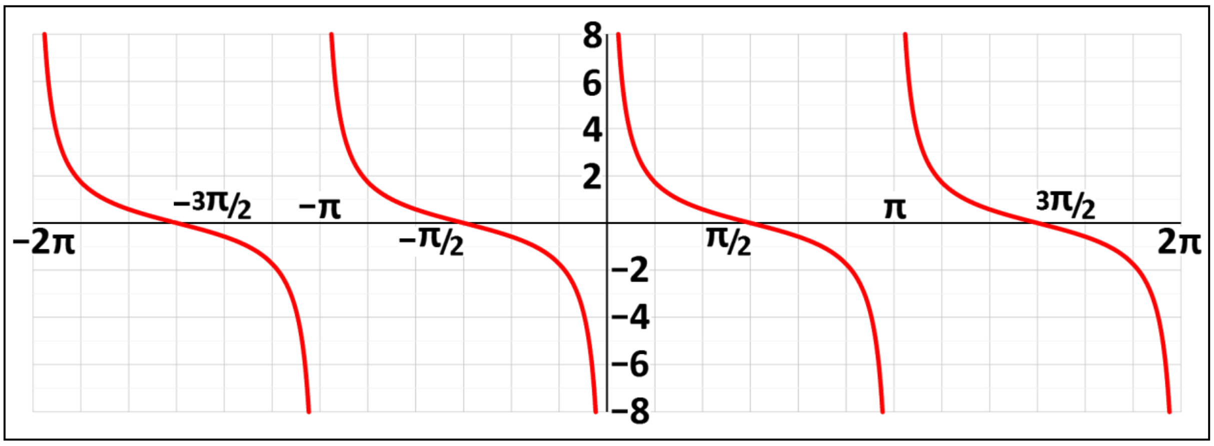
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value, when converted to radians, which is not in the range (-2^{63}, $+2^{63}$)
- The “Trig Funcs Units” Calculator Operation option is set to “Degrees”, and the X register contains a value of (180n + 90) where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Gradians”, and the X register contains a value of (200n + 100) where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Right Angles”, and the X register contains a value of (2n + 1) where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Circles”, and the X register contains a value of (2n + 1)/4 where $n \in \mathbb{Z}$

Flags Affected:

Z	N	D	E
⓪	⦿	⓪	⦿

Cotangent of X

COT



The value in the X register is replaced with its cotangent. The original value in the X register is interpreted as a number of units as currently defined by the "Trig Func Units" Calculator Operation option.

Geometric Definition: cot(x) = Adjacent / Opposite

Calculation:

$$\cot(X) = \frac{1}{\tan(X)}$$

Domain: { x ∈ ℝ : (x / π) ∉ ℤ }　　　　Range: ℝ

Exact Results:

Degrees	Gradians
if ((x – 45) / 180) ∈ ℤ, then cot(x) = 1	if ((x – 50) / 200) ∈ ℤ, then cot(x) = 1
if ((x – 90) / 180) ∈ ℤ, then cot(x) = 0	if ((x – 100) / 200) ∈ ℤ, then cot(x) = 0
if ((x – 135) / 180) ∈ ℤ, then cot(x) = -1	if ((x – 150) / 200) ∈ ℤ, then cot(x) = -1

Right Angles	Circles
if ((x – 0.5) / 2) ∈ ℤ, then cot(x) = 1	if 2(x – 0.125) ∈ ℤ, then cot(x) = 1
if ((x – 1) / 2) ∈ ℤ, then cot(x) = 0	if 2(x – 0.25) ∈ ℤ, then cot(x) = 0
if ((x – 1.5) / 2) ∈ ℤ, then cot(x) = -1	if 2(x – 0.375) ∈ ℤ, then cot(x) = -1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-Inf	QNaN	+Inf	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the "Prevent Mathematically Undefined Floating-point Operations" Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN, Infinity or Pseudo class bit pattern

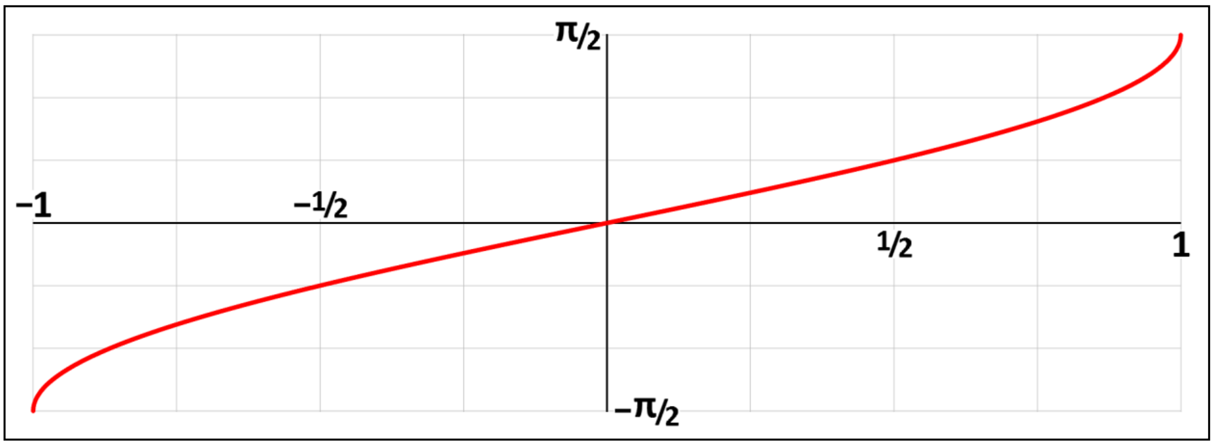
- The X register contains a value of exact-zero
- The X register contains a value, when converted to radians, which is not in the range $(-2^{63}, +2^{63})$
- The “Trig Funcs Units” Calculator Operation option is set to “Degrees”, and the X register contains a value of 180n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Gradians”, and the X register contains a value of 200n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Right Angles”, and the X register contains a value of 2n where $n \in \mathbb{Z}$
- The “Trig Funcs Units” Calculator Operation option is set to “Circles”, and the X register contains a value of n/2 where $n \in \mathbb{Z}$

Flags Affected:

Z	N	D	E
⦿	⦿	⦿	⦿

Arcsine of X

ASIN SIN⁻¹



The value in the X register is replaced with its arcsine. The result is returned in units as currently defined by the “Trig Func Units” Calculator Operation option.

Calculation:

$$\operatorname{asin}(X) = \operatorname{atan}\left(\frac{X}{\sqrt{1 - X^2}}\right)$$

Domain: ℝ[-1, 1] Range: ℝ[-π /2, π /2]

Exact Results:

All Trig Units	Degrees	Gradians	Right Angles	Circles
asin(0) = 0	asin(-1) = -90	asin(-1) = -100	asin(-1) = -1	asin(-1) = -0.25
	asin(-0.5) = -30	asin(1) = 100	asin(1) = 1	asin(1) = 0.25
	asin(0.5) = 30			
	asin(1) = 90			

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-0	0	+0	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value with a magnitude greater than one

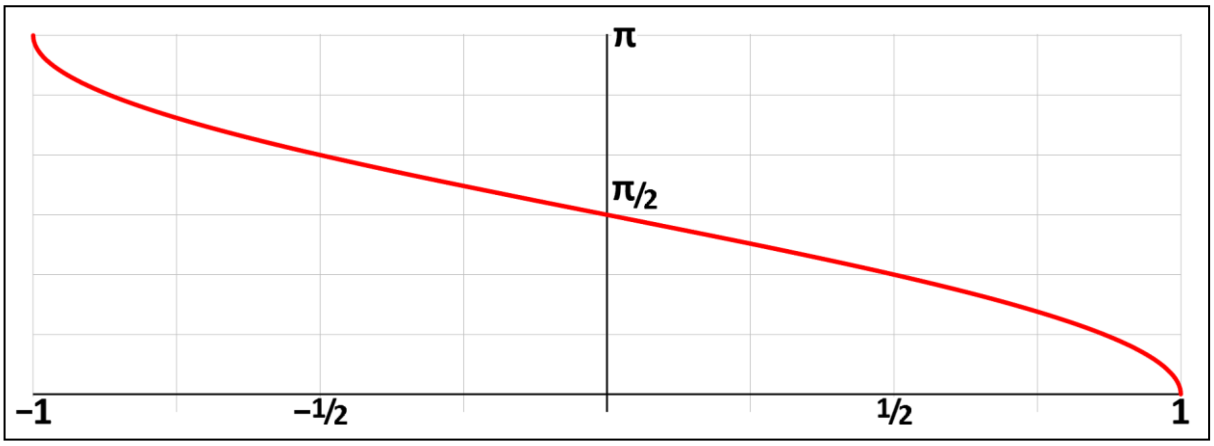
Flags Affected:

Z	N	D	E

-	-	◉	◉

Arccosine of X

ACOS COS⁻¹



The value in the X register is replaced with its arccosine. The result is returned in units as currently defined by the “Trig Func Units” Calculator Operation option.

Calculation:

$$\operatorname{acos}(X) = \operatorname{atan}\left(\frac{\sqrt{1 - X^2}}{X}\right), \quad X \neq 0$$

Domain: ℝ[-1, 1] Range: ℝ[0, π]

Exact Results:

All Trig Units	Degrees	Gradians	Right Angles	Circles
acos(1) = 0	acos(-1) = 180	acos(-1) = 200	acos(-1) = 2	acos(-1) = 0.5
	acos(-0.5) = 120	acos(0) = 100	acos(0) = 1	acos(0) = 0.25
	acos(0) = 90			
	acos(0.5) = 60			

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	+π/2	+π/2	+π/2	QNaN

Values shown in the table are in radians and will be converted to the configured Trigonometric Function Units if required.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

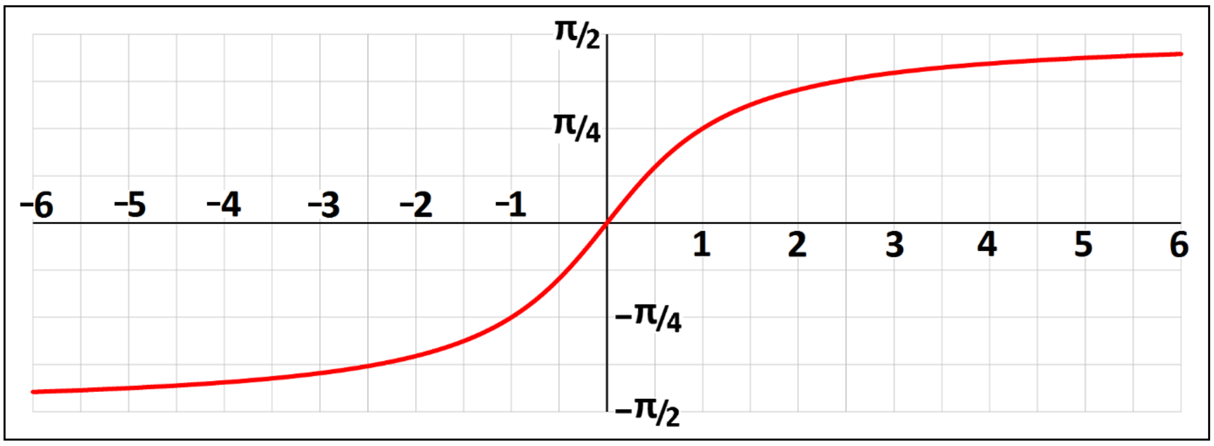
- The X register contains a NaN, Infinity or Pseudo class bit pattern
- The X register contains a value with a magnitude greater than one

Flags Affected:

Z	N	D	E
◉	⓪	◉	◉

Arctangent of X

ATAN TAN^{-1}



The value in the X register is replaced with its arctangent. The result is returned in units as currently defined by the “Trig Func Units” Calculator Operation option.

Domain: ℝ Range: ℝ(-π /2, π /2)

Exact Results:

All Trig Units	Degrees	Gradians	Right Angles	Circles
atan(0) = 0	atan(-1) = -45	atan(-1) = -50	atan(-1) = -0.5	atan(-1) = -0.125
	atan(1) = 45	atan(1) = 50	atan(1) = 0.5	atan(1) = 0.125

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	- π/2	-0	0	+0	+π/2

Values shown in the table are in radians and will be converted to the configured Trigonometric Function Units if required.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

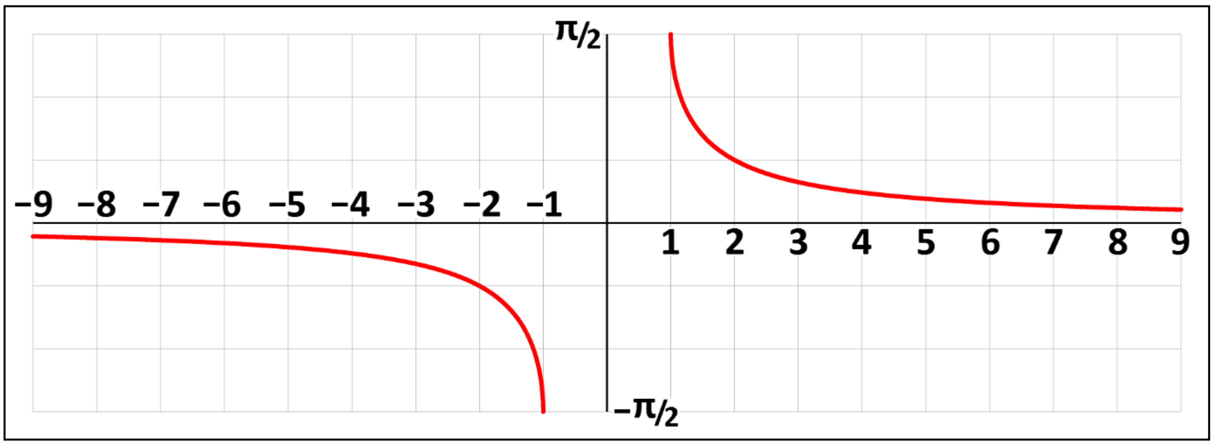
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
-	-	⦿	⦿

Arccosecant of X

ACSC CSC⁻¹



The value in the X register is replaced with its arccosecant. The result is returned in units as currently defined by the "Trig Func Units" Calculator Operation option.

Calculation:

$$\mathrm{acsc}(\mathrm{X}) = \mathrm{atan}\left(\frac{1}{\mathrm{X} \times \sqrt{1 - \frac{1}{\mathrm{X}^2}}}\right)$$

Domain: { $x \in \mathbb{R} : x \notin \mathbb{R}(-1, 1)$ } Range: { $x \in \mathbb{R}[-\pi/2, \pi/2] : x \neq 0$ }

Exact Results:

Degrees	Gradians	Right Angles	Circles
acsc(-2) = -30	acsc(-1) = -100	acsc(-1) = -1	acsc(-1) = -0.25
acsc(-1) = -90	acsc(1) = 100	acsc(1) = 1	acsc(1) = 0.25
acsc(1) = 90			
acsc(2) = 30			

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	QNaN	QNaN	QNaN	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the "Prevent Mathematically Undefined Floating-point Operations" Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

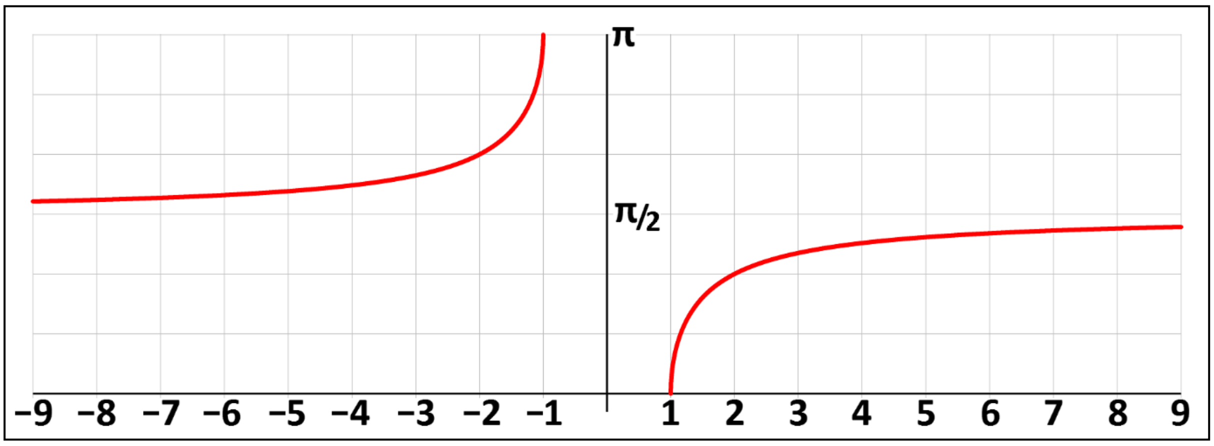
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value with a magnitude less than one

Flags Affected:

Z	N	D	E
⓪	-	◉	◉

Arcsecant of X

ASEC SEC⁻¹



The value in the X register is replaced with its arcsecant. The result is returned in units as currently defined by the “Trig Func Units” Calculator Operation option.

Calculation:

$$\mathrm{asec}(X) = \mathrm{atan}\left(X \times \sqrt{1 - \frac{1}{X^2}}\right)$$

Domain: { x ∈ ℝ : x ∉ ℝ(-1, 1) } Range: { x ∈ ℝ[0, π] : x ≠ (π /2) }

Exact Results:

<u>All Trig Units</u>	<u>Degrees</u>	<u>Gradians</u>	<u>Right Angles</u>	<u>Circles</u>
asec(1) = 0	asec(-2) = 120	asec(-1) = 200	asec(-1) = 2	asec(-1) = 0.5
	asec(-1) = 180			
	asec(2) = 60			

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+π/2	QNaN	QNaN	QNaN	+π/2

Values shown in the table are in radians and will be converted to the configured Trigonometric Function Units if required.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value with a magnitude less than one

Flags Affected:

Z	N	D	E
⦿	⓪	⦿	⦿

Arccotangent of X

ACOT COT⁻¹

The value in the X register is replaced with its arccotangent. The result is returned in units as currently defined by the “Trig Func Units” Calculator Operation option.

Normally, the range of an inverse trigonometric function is the same as that of its inverse reciprocal function, e.g., arccosecant has the same range as arcsine and arcsecant has the same range as arccosine. However, defining the range of the arccotangent function to be the same as that of the arctangent function, i.e., ℝ(-π /2, π /2), produces a discontinuity at x = 0, this can be avoided if the range is defined as ℝ(0, π). The “ACOT Range” Calculator Operation option allows you to choose which range to use.

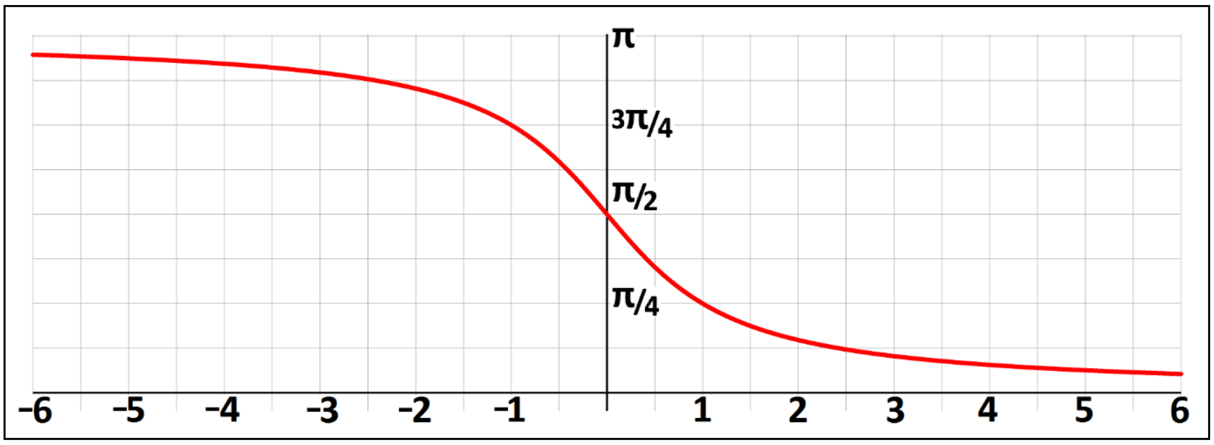
This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern

“ACOT Range” Calculator Operation Option set to (0, +pi)



Calculation:

$$\text{if } X > 0, \quad \operatorname{acot}(X) = \operatorname{atan}\left(\frac{1}{X}\right); \qquad \text{if } X < 0, \quad \operatorname{acot}(X) = \operatorname{atan}\left(\frac{1}{X}\right) + \pi$$

Domain: ℝ   Range: ℝ(0, π)

Exact Results:

Degrees	Gradians	Right Angles	Circles
acot(-1) = 135	acot(-1) = 150	acot(-1) = 1.5	acot(-1) = 0.375
acot(0) = 90	acot(0) = 100	acot(0) = 1	acot(0) = 0.25
acot(1) = 45	acot(1) = 50	acot(1) = 0.5	acot(1) = 0.125

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf

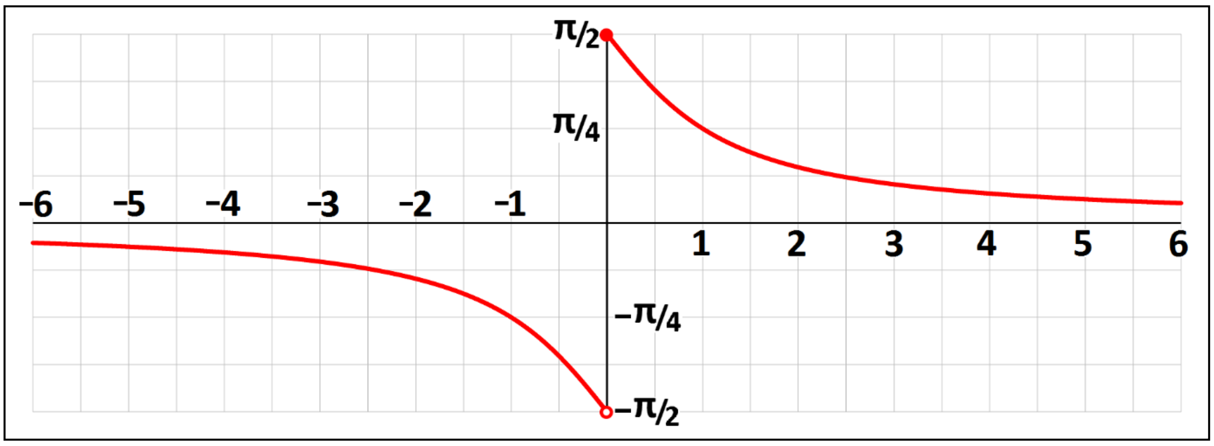
Result	QNaN	+π	+π/2	+π/2	+π/2	+0

Values shown in the table are in radians and will be converted to the configured Trigonometric Function Units if required.

Flags Affected:

Z	N	D	E
⓪	⓪	◉	◉

“ACOT Range” Calculator Operation Option set to (-pi/2, +pi/2]



Calculation:

$$\operatorname{acot}(X) = \operatorname{atan}\left(\frac{1}{X}\right), \quad X \neq 0$$

Domain: ℝ Range: { x ∈ ℝ(-π /2, π /2] : x ≠ 0 }

Exact Results:

Degrees	Gradians	Right Angles	Circles
acot(-1) = -45	acot(-1) = -50	acot(-1) = -0.5	acot(-1) = -0.125
acot(0) = 90	acot(0) = 100	acot(0) = 1	acot(0) = 0.25
acot(1) = 45	acot(1) = 50	acot(1) = 0.5	acot(1) = 0.125

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	-π/2	+π/2	+π/2	+0

Values shown in the table are in radians and will be converted to the configured Trigonometric Function Units if required.

Flags Affected:

Z	N	D	E
⓪	-	◉	◉

Hyperbolic Operations

Like the trigonometric functions, the hyperbolic functions are a family of functions that are commonly encountered in many areas of mathematics and engineering, but not as often as the trigonometric functions.

For each of the trigonometric functions there is an associated hyperbolic function. The name of a hyperbolic function is the same as that of its associated trigonometric function, but prefixed with the word “hyperbolic”. The abbreviation used for a hyperbolic function is the same as that for its associated trigonometric function, but suffixed with the letter “h”, e.g., the abbreviation for the hyperbolic tangent function is tanh".

There is no universally accepted way of pronouncing these abbreviations, just use whatever you are comfortable with.

The basic hyperbolic functions have the following definitions:

$$\sinh(x) = \frac{e^x - e^{-x}}{2}, \qquad \cosh(x) = \frac{e^x + e^{-x}}{2}, \qquad \tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

The relationships of the other hyperbolic functions, and their inverse functions, to these basic functions are the same as those between the associated trigonometric functions.

These definitions appear to have nothing in common with the trigonometric functions, a view that is reinforced when you compare the graphs of associated functions. The hyperbolic functions have no geometric interpretation and they are not periodic; so why are the associations made?

The reason is that, just as the definitions of the trigonometric functions result in many trigonometric identities, so do the hyperbolic functions, and many of them bear an uncanny resemblance to each other considering the fundamentally different definitions, though sometimes a sign is different.

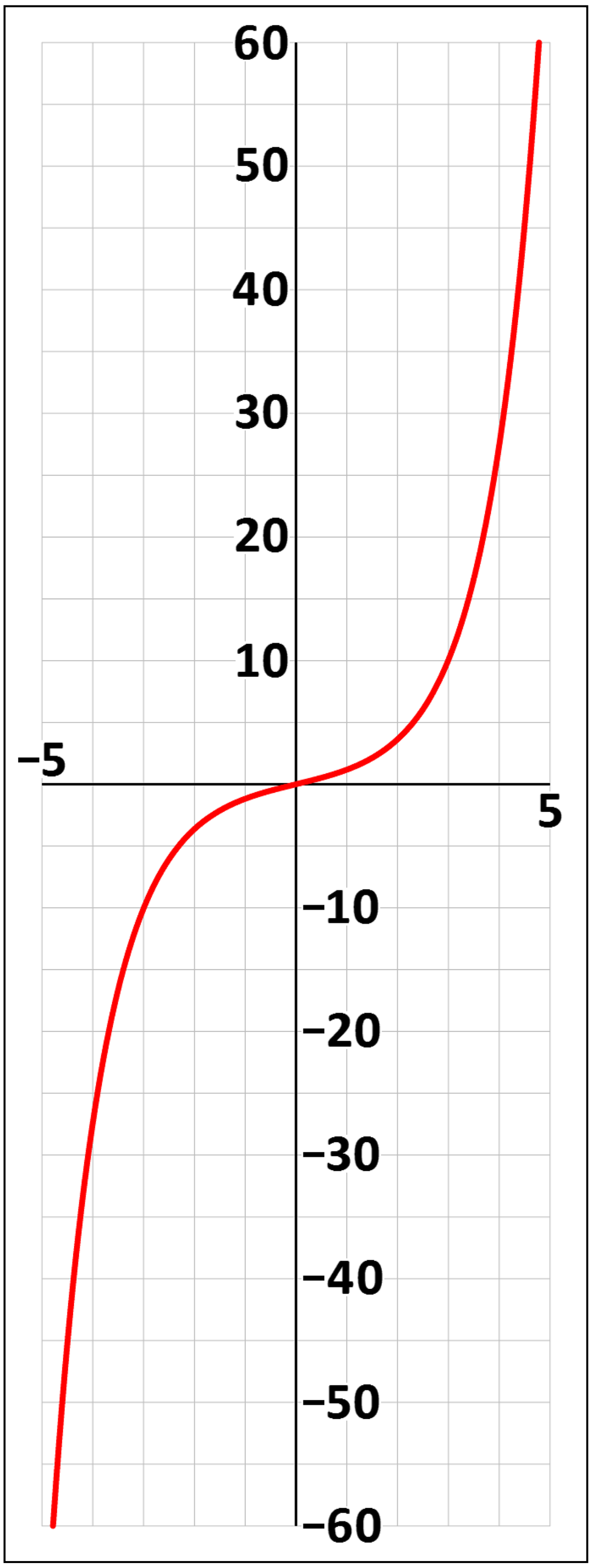
Some examples are:

Trigonometric	**Hyperbolic**
$\cos^2 x + \sin^2 x = 1$	$\cosh^2 x - \sinh^2 x = 1$
$\sin(x + y) = \sin x \cos y + \cos x \sin y$	$\sinh(x + y) = \sinh x \cosh y + \cosh x \sinh y$
$1 + \tan^2 x = \sec^2 x$	$1 - \tanh^2 x = \operatorname{sech}^2 x$
$\frac{d}{dx} \sin x = \cos x$	$\frac{d}{dx} \sinh x = \cosh x$

The name “hyperbolic” comes from the fact that the points defined by the coordinates (cosh(x), sinh(x)) trace out a hyperbola in a similar way to how the trigonometric (circular) functions trace out a circle.

Hyperbolic Sine of X

SINH



The value in the X register is replaced with its hyperbolic sine.

Calculation:

$$\sinh(\mathrm{X}) = \frac{\mathrm{e}^{\mathrm{X}} - \mathrm{e}^{-\mathrm{X}}}{2}$$

Domain: $\mathbb{R}$

Range: $\mathbb{R}$

Exact Results:

sinh(0) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-Inf	-0	0	+0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

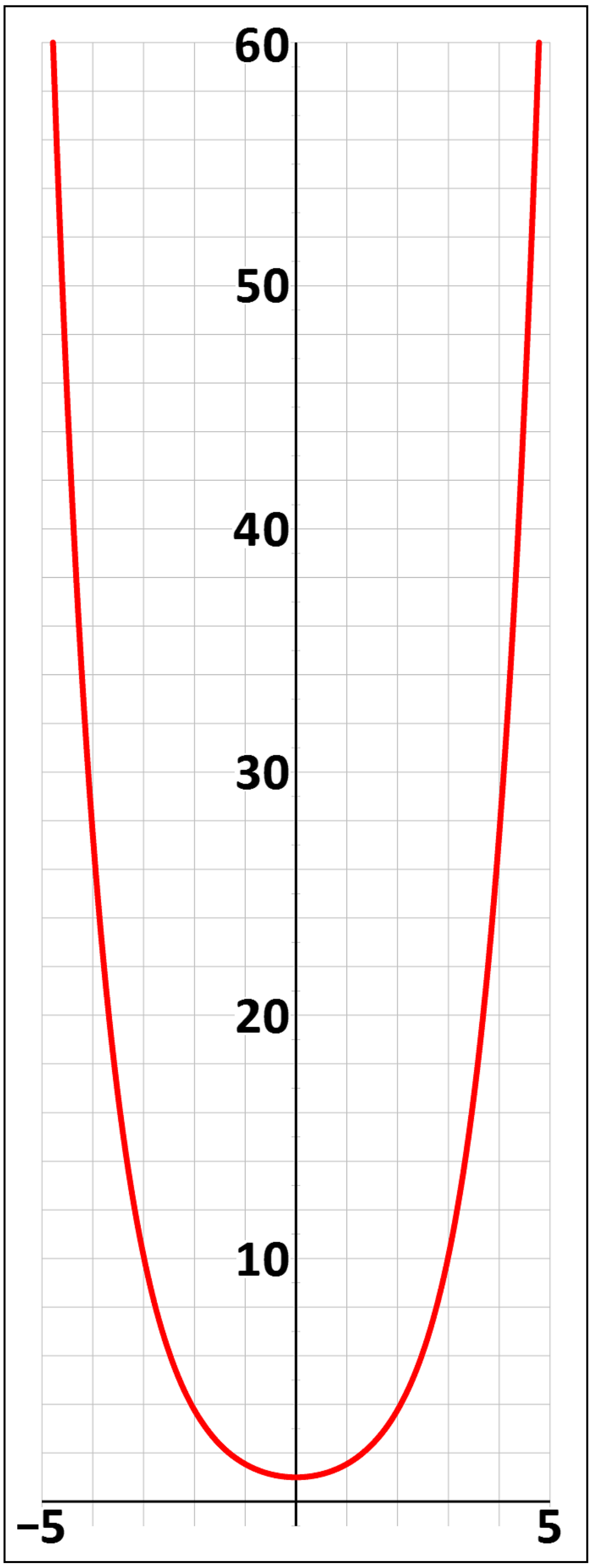
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
-	-	◉	◉

Hyperbolic Cosine of X

COSH



The value in the X register is replaced with its hyperbolic cosine.

Calculation:

$$\cosh(X) = \frac{e^X + e^{-X}}{2}$$

Domain: ℝ

Range: ℝ[1, ∞)

Exact Results:

cosh(0) = 1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+Inf	+ie1	+1	+ie1	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

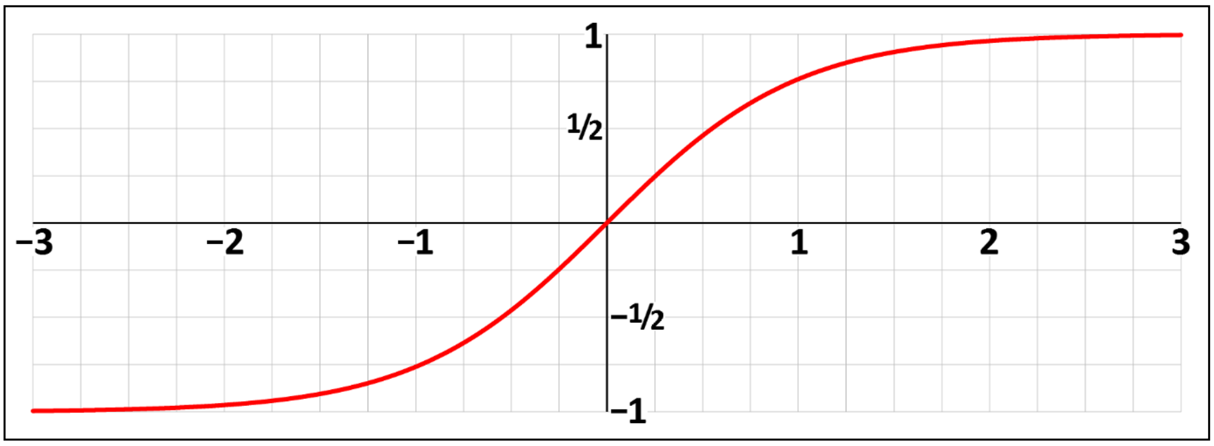
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⓪	⓪	⓪	◉

Hyperbolic Tangent of X

TANH



The value in the X register is replaced with its hyperbolic tangent.

Calculation:

$$\tanh(X) = \frac{e^{2X} - 1}{e^{2X} + 1}$$

Domain: ℝ Range: ℝ(-1, 1)

Exact Results:

tanh(0) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-ie1	-0	0	+0	+ie1

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

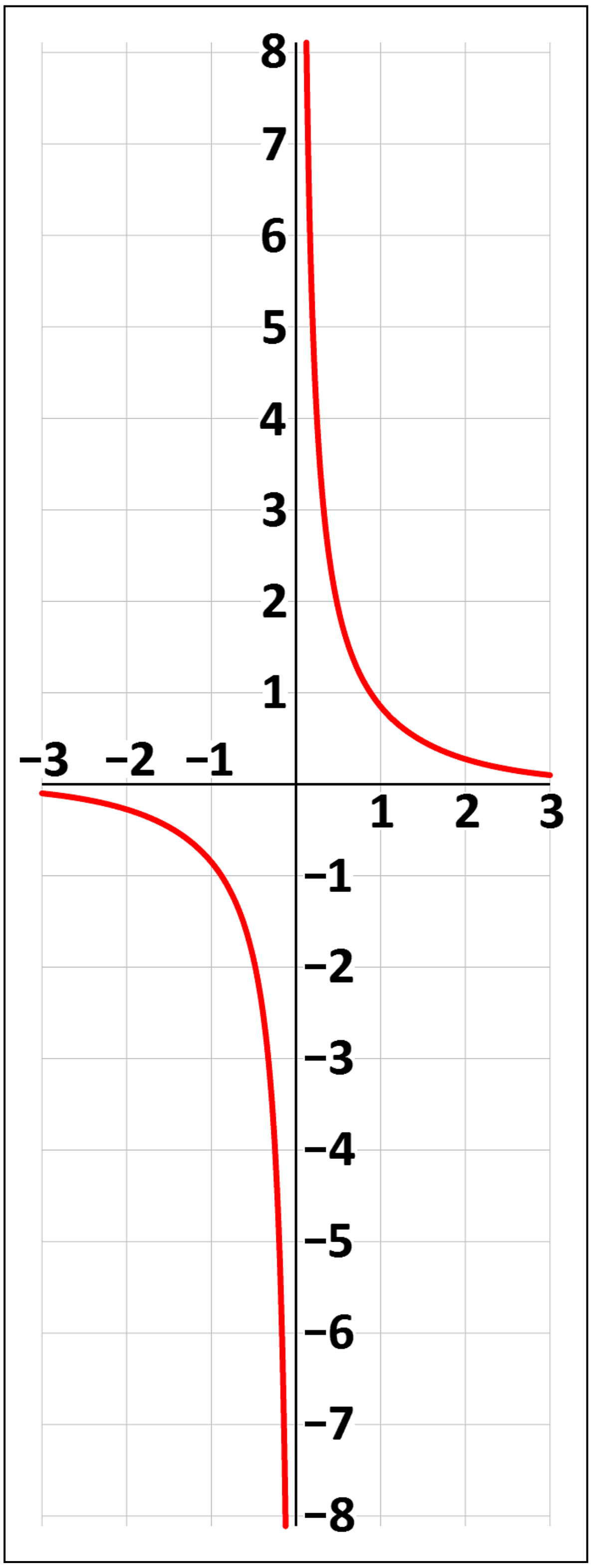
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
-	-	⦿	⦿

Hyperbolic Cosecant of X

CSCH



The value in the X register is replaced with its hyperbolic cosecant.

Calculation:

$$\operatorname{csch}(X) = \frac{2}{e^{X} - e^{-X}}$$

Domain: { x ∈ ℝ : x ≠ 0 }

Range: { x ∈ ℝ : x ≠ 0 }

Exact Results: None

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	-Inf	QNaN	+Inf	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a value of exact-zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

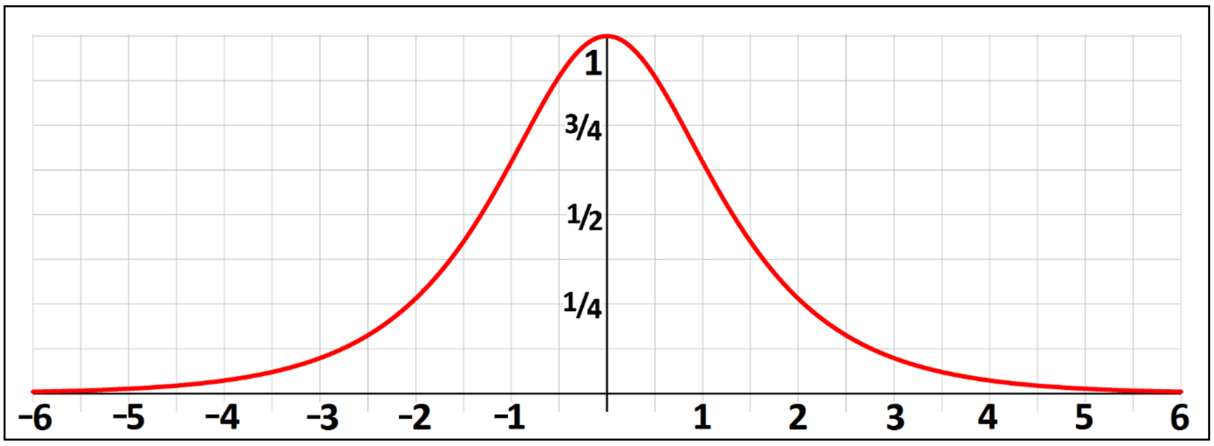
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⓪	-	◉	⓪

Hyperbolic Secant of X

SECH



The value in the X register is replaced with its hyperbolic secant.

Calculation:

$$\operatorname{sech}(X) = \frac{2}{e^{X} + e^{-X}}$$

Domain: ℝ Range: ℝ(0, 1]

Exact Results:

sech(0) = 1

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	+0	+ie1	+1	+ie1	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

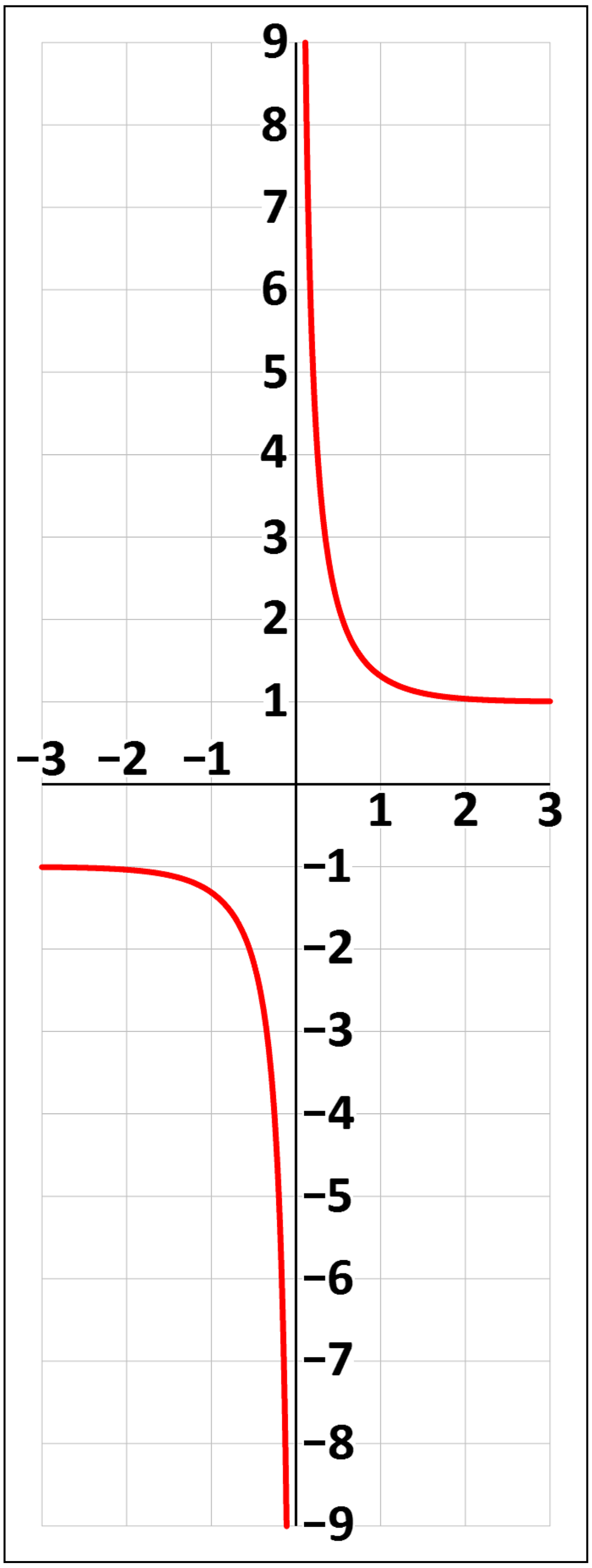
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
⓪	⓪	◉	◉

Hyperbolic Cotangent of X

COTH



The value in the X register is replaced with its hyperbolic cotangent.

Calculation:

$$\coth(X) = \frac{e^{2X} + 1}{e^{2X} - 1}$$

Domain: { x ∈ ℝ : x ≠ 0 }

Range: { x ∈ ℝ : x ∉ ℝ[-1, 1] }

Exact Results: None

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-ie1	-Inf	QNaN	+Inf	+ie1

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

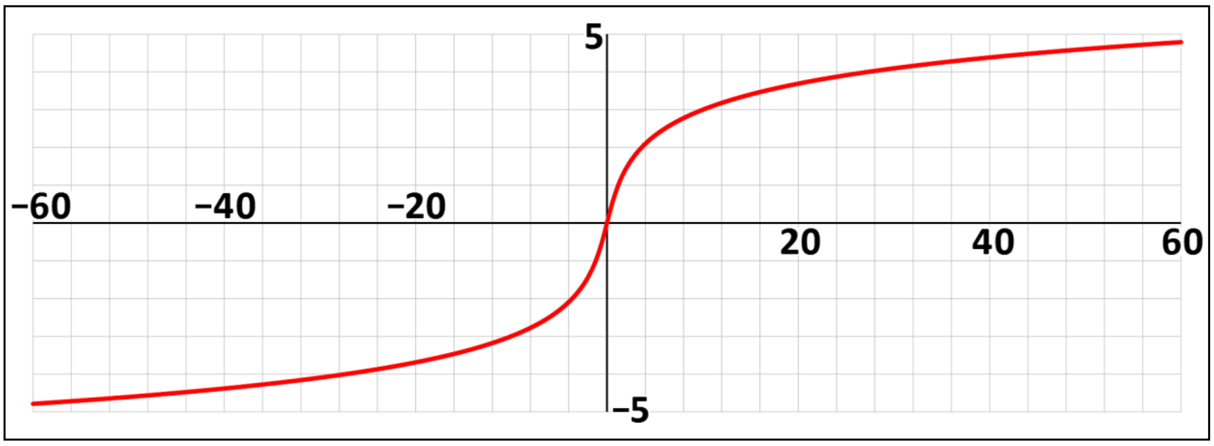
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value of exact-zero

Flags Affected:

Z	N	D	E
⓪	-	⓪	⓪

Inverse Hyperbolic Sine of X

ASINH SINH⁻¹



The value in the X register is replaced with its inverse hyperbolic sine.

Calculation:

$$\text{asinh}(\text{X}) = \log_e\left(\text{X} + \sqrt{X^2 + 1}\right)$$

Domain: ℝ Range: ℝ

Exact Results:

asinh(0) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-Inf	-0	0	+0	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

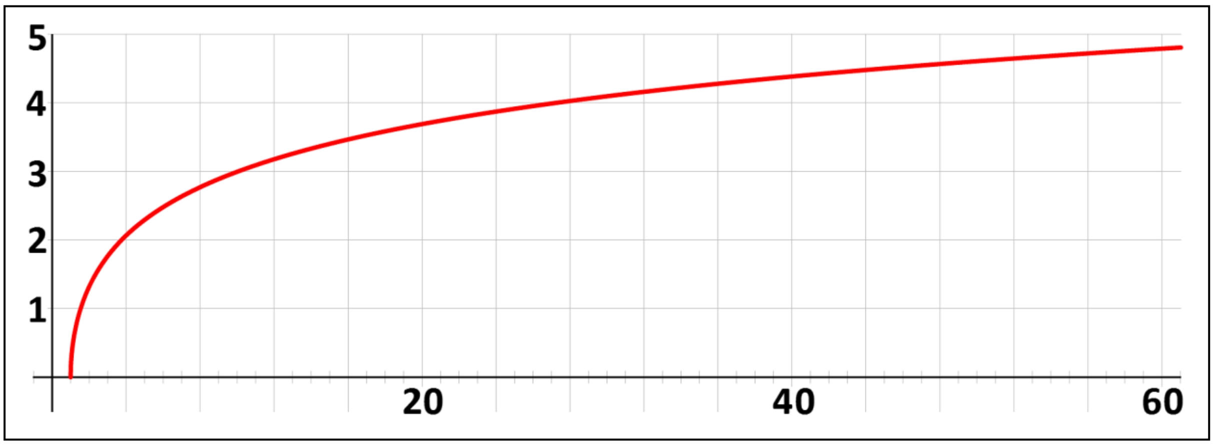
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
-	-	⦿	⦿

Inverse Hyperbolic Cosine of X

ACOSH COSH^{-1}



The value in the X register is replaced with its inverse hyperbolic cosine.

Calculation:

$$\text{acosh}(X) = \log_e\left(X + \sqrt{X^2 - 1}\right)$$

Domain: ℝ[1, ∞)　　　　Range: ℝ[0, ∞)

Exact Results:

acosh(1) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	QNaN	QNaN	+Inf

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

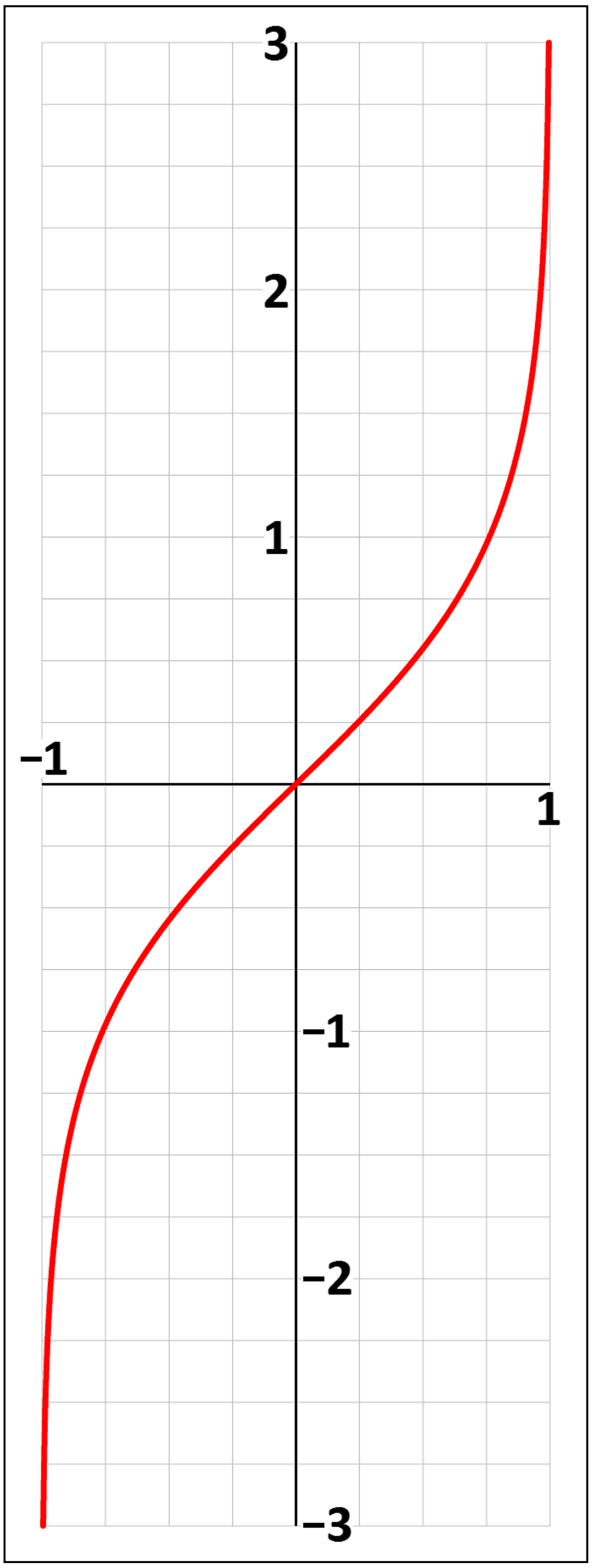
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value less than one

Flags Affected:

Z	N	D	E
◉	⓪	◉	◉

Inverse Hyperbolic Tangent of X

ATANH TANH^{-1}



The value in the X register is replaced with its inverse hyperbolic tangent.

Calculation:

$$\operatorname{atanh}(X) = \frac{\log_e\left(\frac{1+X}{1-X}\right)}{2}$$

Domain: ℝ(-1,1)

Range: ℝ

Exact Results:

atanh(0) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	-0	0	+0	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode
- The “Prevent Ineffective Operations” Calculator Operation option is selected and the X register contains a value of zero

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

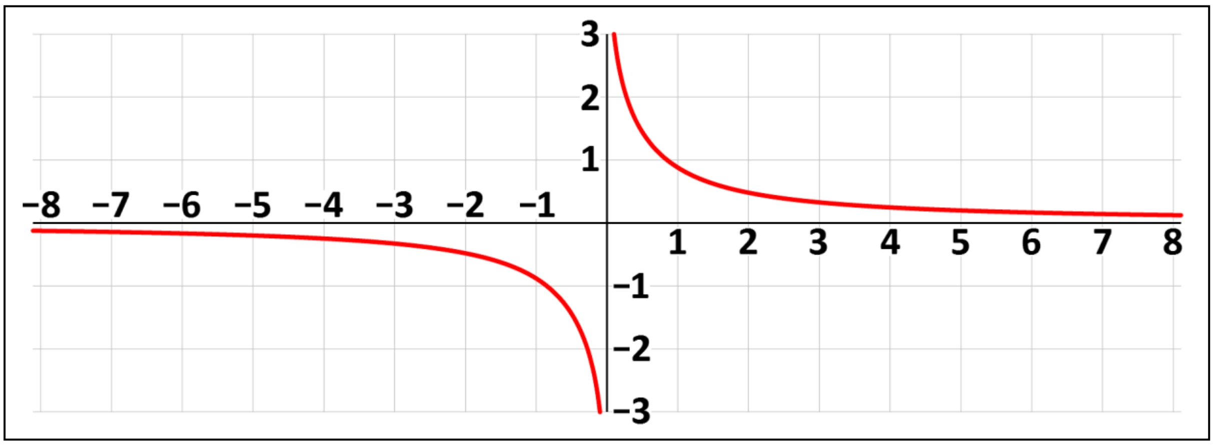
- The X register contains a NaN or Pseudo class bit pattern

Flags Affected:

Z	N	D	E
-	-	⦿	⦿

Inverse Hyperbolic Cosecant of X

ACSCH CSCH^{-1}



The value in the X register is replaced with its inverse hyperbolic cosecant.

Calculation:

$$\operatorname{acsch}(X) = \log_e\left(\frac{1}{X} + \frac{\sqrt{X^2+1}}{|X|}\right)$$

Domain: { x ∈ ℝ : x ≠ 0 } Range: { x ∈ ℝ : x ≠ 0 }

Exact Results: None

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	-Inf	QNaN	+Inf	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

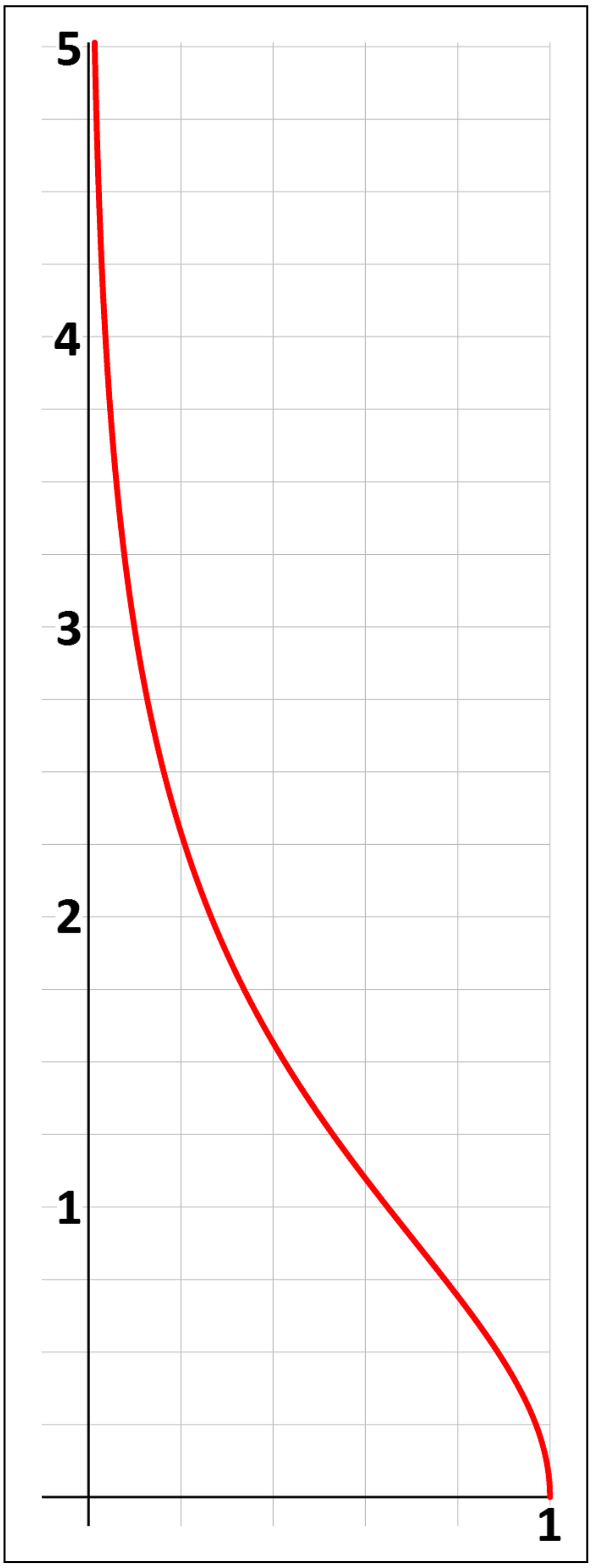
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value of exact-zero

Flags Affected:

Z	N	D	E
⓪	-	◉	⓪

Inverse Hyperbolic Secant of X

ASECH $SECH^{-1}$



The value in the X register is replaced with its inverse hyperbolic secant.

Calculation:

$$\operatorname{asech}(X) = \log_e\left(\frac{1+\sqrt{1-X^2}}{X}\right)$$

Domain: ℝ(0,1]

Range: ℝ[0, ∞)

Exact Results:

asech(1) = 0

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	QNaN	QNaN	QNaN	+Inf	QNaN

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

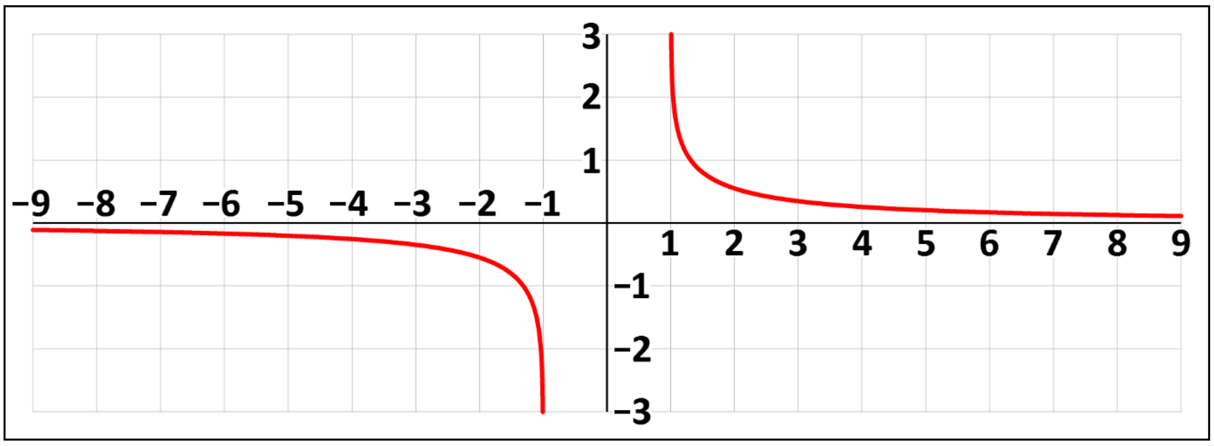
- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a value of exact-zero
- The X register contains a negative value
- The X register contains a value greater than one

Flags Affected:

Z	N	D	E
●	⓪	●	●

Inverse Hyperbolic Cotangent of X

ACOTH COTH^{-1}



The value in the X register is replaced with its inverse hyperbolic cotangent.

Calculation:

$$\mathrm{acoth}(\mathrm{X}) = \frac{\log_e\left(\frac{\mathrm{X}-1}{\mathrm{X}+1}\right)}{2}$$

Domain: { x ∈ ℝ : x ∉ ℝ[-1, 1] } Range: { x ∈ ℝ : x ≠ 0 }

Exact Results: None

Special Cases:

X	NaN/Pseudo	-Inf	-0	0	+0	+Inf
Result	QNaN	-0	QNaN	QNaN	QNaN	+0

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode

If the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, then this key will be disabled under any of the following circumstances:

- The X register contains a NaN or Pseudo class bit pattern
- The X register contains a negative value or a value with a magnitude less than or equal to one

Flags Affected:

Z	N	D	E
⓪	-	◉	⓪

Floating-Point Word Value Display Mode

Floating-point Base Display Change

D⇆B 10⇆2 $N_{10⇆2}$

This switches the “Fixed/Floating-point Display Format” Number Display option between decimal and binary as follows:

From	To
Decimal General	Binary General
Decimal Exponent	Binary Exponent
Decimal Engineering	
Binary General	Decimal General
Binary Exponent	Decimal Exponent

No flags are affected.

Floating-Point Word Value/Fields/Fields Bit Pattern Display Modes

Floating-point Fields Display Switch

FLDS

This switches between Floating-point Value and Floating-point Fields display mode.

This key will be disabled in Floating-point Fields Bit Pattern display mode.

No flags are affected.

Fixed/Floating-point Display Format Change

DISPF

This cycles the “Fixed/Floating-point Display Format” Number Display option between general, exponential and engineering as follows:

From	To
Decimal General	Decimal Exponent
Decimal Exponent	Decimal Engineering

Decimal Engineering	Decimal General
Binary General	Binary Exponent
Binary Exponent	Binary General

This key will be disabled in Floating-point Fields and Floating-point Fields Bit Pattern display modes.

No flags are affected.

Floating-Point Fields/Fields Bit Pattern Display Modes

Floating-point Bit Pattern Display Switch

BITP | F⇆BP | ⓪①

This switches between Floating-point Fields and Floating-point Fields Bit Pattern display mode.

No flags are affected.

Floating-Point Components Display Mode

Select Previous Component

◀CPN | ⇤▯▯

This makes the component immediately to the left of the currently selected component as the new selected component.

If the leftmost component is currently selected then the rightmost component becomes the new selected component.

No flags are affected.

Select Next Component

CPN▶ | ▯▯⇥

Selects the component immediately to the right of the currently selected component as the new selected component.

If the rightmost component is currently selected then the leftmost component becomes the new selected component.

No flags are affected.

Set Unit Component

UNIT =1

This makes the currently selected component the unit component

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the currently selected component is the unit component.

No flags are affected.

8.4.3 Number Coding Keys Region

Use Previous Number Coding

PCod

Keyboard Shortcut: Spacebar

The Number Coding options are set to the settings they had before the most recent change, including if they were changed by the use of this key. This allows you to quickly switch back and forth between two sets of Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of the “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for the previous number coding

Flags Affected:

C	Z	N	V	D	E	O
-	⦿	⦿	⓪	⦿	⦿	⦿

8-bit Integer Coding

8bit ◄ 8 ►

Sets the “Number Coding Type” option to Integer and the “Integer Width” option to 8 bits in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for integer codings

Flags Affected:

C	Z	N	V	O
-	⦿	⦿	⓪	⦿

16-bit Integer Coding

16bit ◂16▸

Sets the “Number Coding Type” option to Integer and the “Integer Width option” to 16 bits in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for integer codings

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	◉

32-bit Integer Coding

32bit ◂32▸

Sets the “Number Coding Type” option to Integer and the “Integer Width” option to 32 bits in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for integer codings

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	◉

64-bit Integer Coding

64bit ◂64▸

Sets the “Number Coding Type” option to Integer and the “Integer Width” option to 64 bits in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for integer codings

Flags Affected:

C	Z	N	V	O
-	◉	◉	⓪	◉

Fixed-point Coding

FixP

Sets the “Number Coding Type” option to Fixed-point in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for fixed-point codings

Flags Affected:

C	Z	N	V	E	O
-	◉	◉	⓪	◉	◉

32-bit Floating-point Coding

FP32

Sets the “Number Coding Type” option to Floating-point and the “Floating-point Coding” option to 32-bit in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for floating-point codings

Flags Affected:

C	Z	N	D	E
-	-	-	⦿	⦿

64-bit Floating-point Coding

FP64

Sets the “Number Coding Type” option to Floating-point and the “Floating-point Coding” option to 64-bit in the Number Coding options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields display mode
- Floating-point Fields Bit Pattern display mode
- The current combination of “Number Coding Option Change Handling” options in the Calculator Operation options, the Number Coding options, and the register (stack and memories) content prevent the change from being made
- The “Calculator Type” option in the Calculator Operation options is set to Algebraic Logic and the operator stack contains an operator that is not available for floating-point codings

Flags Affected:

C	Z	N	D	E
-	-	-	⦿	⦿

8.4.4 Expression Editor Region

Execute Expression

▶

Evaluates the expression currently contained in the expression editor edit control.

If the Expression Editor “Auto Filter” option in the Calculator Operation options is not set then any invalid characters in the expression editor edit control are removed prior to evaluation.

Evaluating an expression performs the same processing as that which would occur if each character contained in the edit control was entered at the keyboard.

Flags Affected:

C	Z	N	V	D	E	O
⦿	⦿	⦿	⦿	⦿	⦿	⦿

8.4.5 Display Region

Switch Main Number Display Digit Size

Toggles the “Digit Size” Digit Set option between “Small” and “Medium”. Small digits allow more digits to be displayed in the main number display. Medium digits allow more information to be displayed in the markers area.

No flags are affected.

Detach Number Display

Keyboard Shortcut: Ctrl+D

Switches the “Main Number Display” Window option from “Embedded” to “Detached”. The main number display will be detached from the keypad and displayed it in its own window.

No flags are affected.

Copy to Clipboard

Keyboard Shortcut: Ctrl+C

Copies the contents of the main number display to the clipboard. The format of the data copied to the clipboard is controlled by the Clipboard options.

This key will be disabled in Floating-point Components display mode.

No flags are affected.

Paste from Clipboard

Keyboard Shortcut: Ctrl+V

Pastes the text contents of the clipboard to the main number display and sets the contents of the X register accordingly. The interpretation of the text pasted from the clipboard is controlled by the Clipboard options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode

- Floating-point Components display mode and the “Paste Mode” Clipboard option is set to “Number”

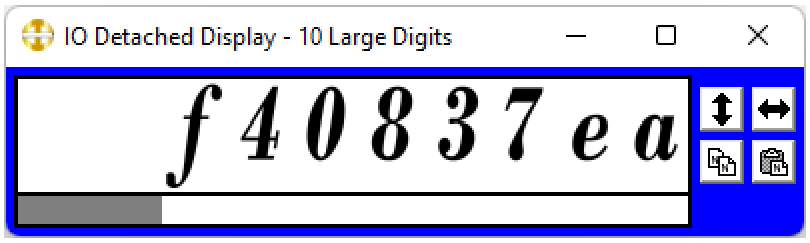
Flags Affected:

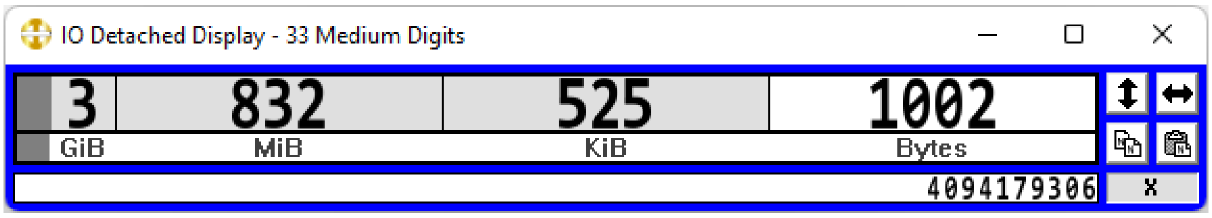
C	Z	N	V	D	E	O
-	⦿	⦿	⓪	⦿	⦿	⓪

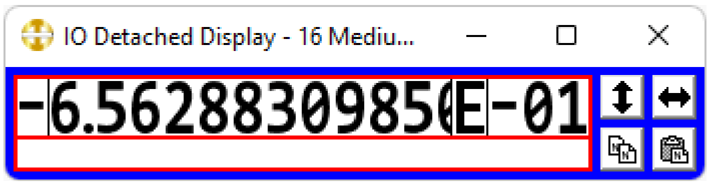
9 DETACHED NUMBER DISPLAY WINDOW

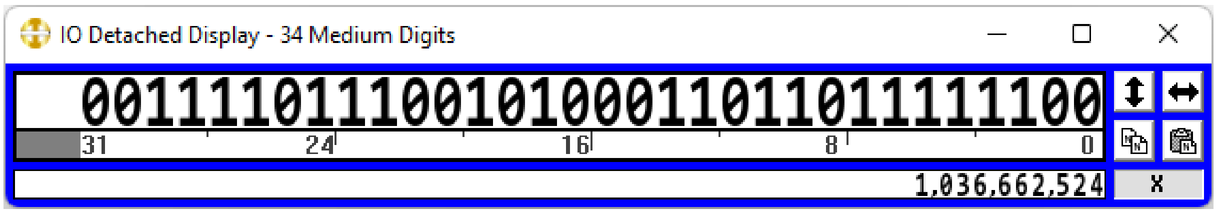
This window contains the main number display and the currently configured secondary number display, if any. It can be displayed by selecting the "Detached Number Display…" entry of the "Windows" menu, clicking on the keypad ↗ button, or by using Ctrl+D keyboard shortcut.

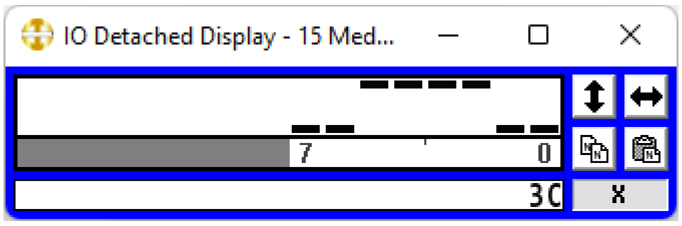
9.1 Organizations

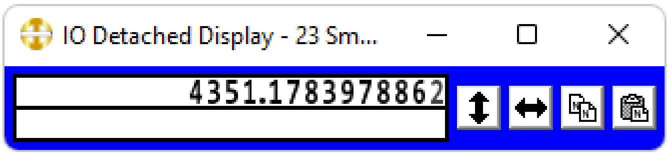


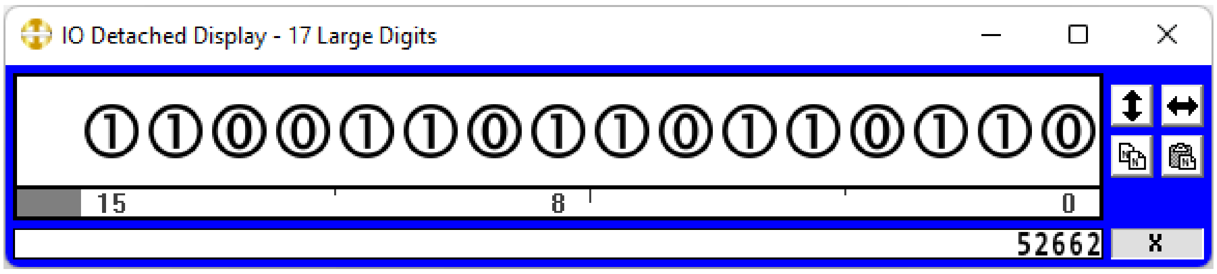




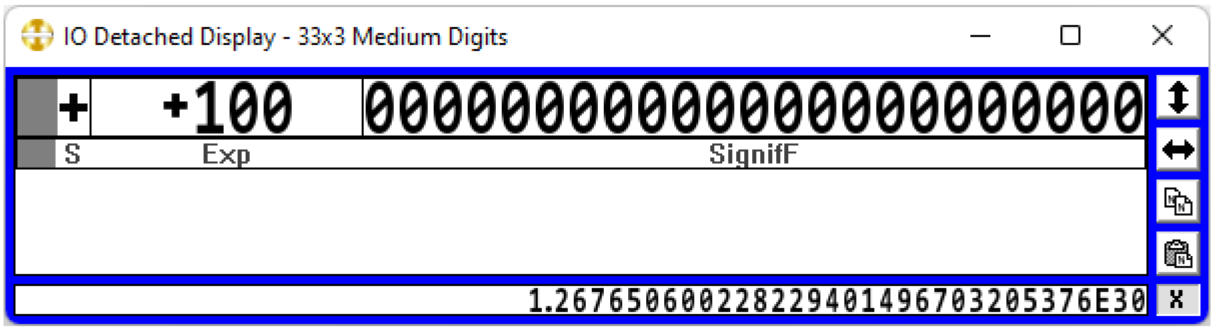




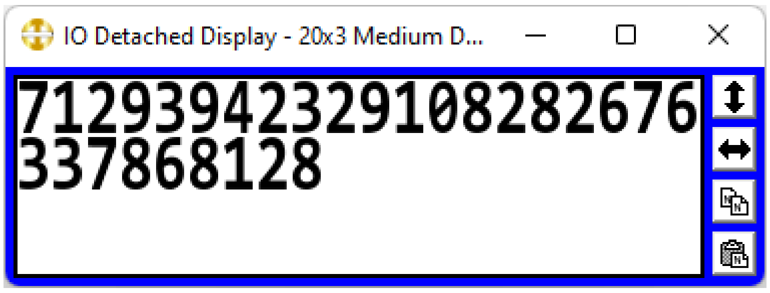



IO Detached Display - 17 Large Digits

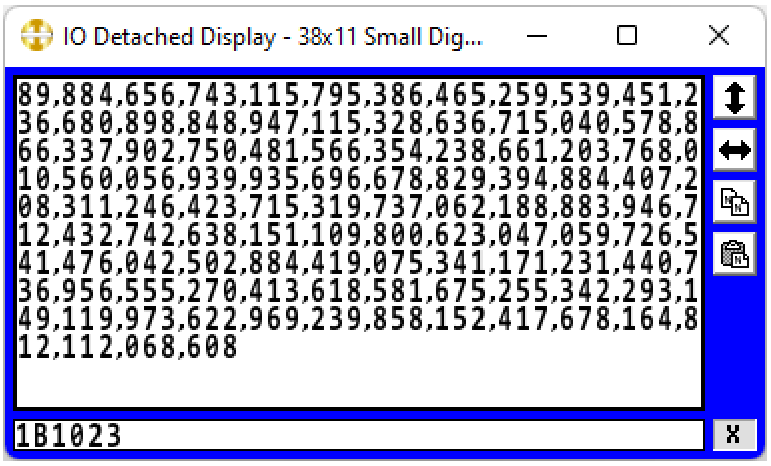
```
①①⓪⓪①①⓪①①⓪①①⓪①①⓪
15            8            0
52662
```


IO Detached Display - 33x3 Medium Digits

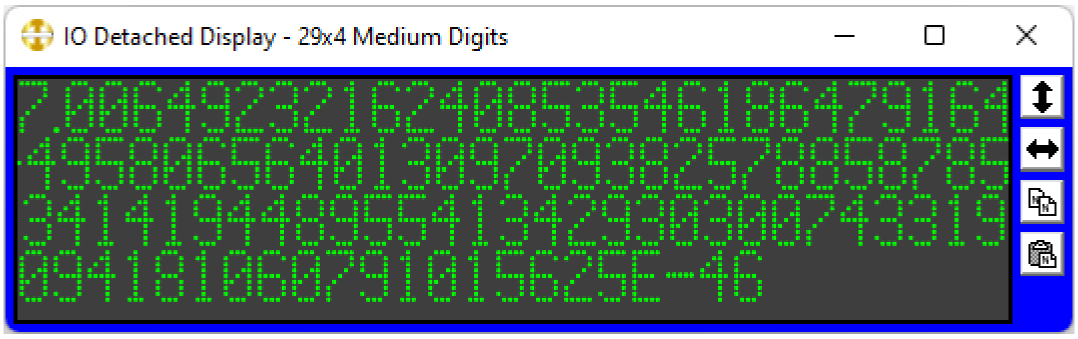
```
+  +100  00000000000000000000000000
S   Exp            SignifF
1.267650600228229401496703205376E30
```


IO Detached Display - 20x3 Medium D...

```
71293942329108282676
337868128
```


IO Detached Display - 38x11 Small Dig...

```
89,884,656,743,115,795,386,465,259,539,451,2
36,680,898,848,947,115,328,636,715,040,578,8
66,337,902,750,481,566,354,238,661,203,768,0
10,560,056,939,935,696,678,829,394,884,407,2
08,311,246,423,715,319,737,062,188,883,946,7
12,432,742,638,151,109,800,623,047,059,726,5
41,476,042,502,884,419,075,341,171,231,440,7
36,956,555,270,413,618,581,675,255,342,293,1
49,119,973,622,969,239,858,152,417,678,164,8
12,112,068,608
1B1023
```


IO Detached Display - 29x4 Medium Digits

```
7.0064923216240853546186479164
49580656401309709382578858785
34141944895541342930300743319
094181060791015625E-46
```

The organization of the detached number display window is determined by the following criteria:

The Detached Display Type currently selected in the window options. This setting may be toggled by clicking this window's maximize button which is immediately to the left of the close button.

The Digit Size currently selected in the digit set options. This setting can be cycled by clicking on the up/down arrow key in this window.

Whether or not the Secondary Display Type in the window options is currently set to None.

The width of this window may be modified by dragging the left or right edge of the window with the mouse.

If the Multiple Line setting is currently selected for the "Detached Display Type" Window option then: the height of this window may be modified by dragging the top or bottom edge of the window with the mouse, the width and height of this window may be modified by dragging one of the corners of the window with the mouse

9.2 Keys

Cycle Number Display Digit Size

Cycles the "Digit Size" Digit Set option through the sequence Small -> Medium -> Large -> Small. Shift-clicking will cycle in the reverse order.

Larger digits allow more information to be displayed in the markers area and secondary number display. The width of the detached display window is modified to maintain the same number of digits displayed (per row for multiline display).

No flags are affected.

Change Number Display Width

Cycle the number of digits displayed in the main number display through the sequence
9 -> 16 -> 24 -> 32 -> 48 -> 64 -> 80 -> 96 -> 9. Shift-clicking will cycle in the reverse order.

For an un-shifted click the number chosen will be 9 if the current number of displayed digits is 96 or more, otherwise it will be the smallest number in the list which is greater than the current number of displayed digits.

For a shifted click the number chosen will be 96 if the current number of displayed digits is 9 or less, otherwise it will be the largest number in the list which is less than the current number of displayed digits.

No flags are affected.

Copy to Clipboard

Keyboard Shortcut: Ctrl+C

Copies the contents of the main number display to the clipboard. The format of the data copied to the clipboard is controlled by the Clipboard options.

This key will be disabled in Floating-point Components display mode.

No flags are affected.

Paste from Clipboard

Keyboard Shortcut: Ctrl+V

Pastes the text contents of the clipboard to the main number display and sets the contents of the X register accordingly. The interpretation of the text pasted from the clipboard is controlled by the Clipboard options.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- Floating-point Components display mode and the “Paste Mode” Clipboard option is set to “Number”

Flags Affected:

C	Z	N	V	D	E	O
-	◉	◉	⓪	◉	◉	⓪

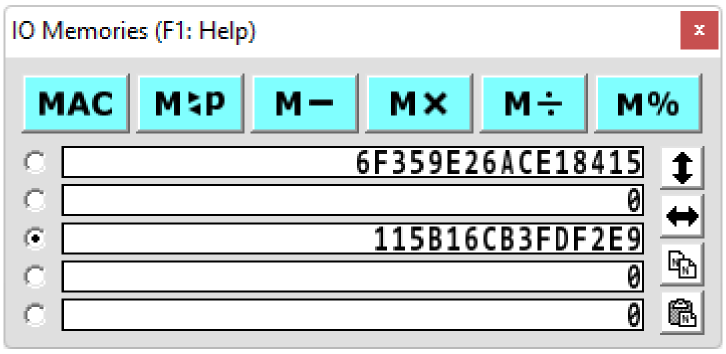
---

10 MEMORIES DISPLAY WINDOW

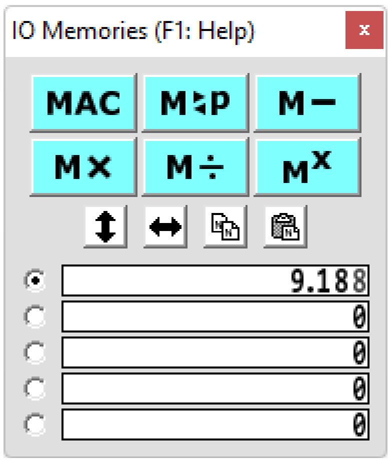
This window displays the current contents of all the memory registers. It contains buttons for additional operations related to the memory registers, including the selection of the current memory register. It can be displayed by selecting the “Memories…” entry of the “Windows” menu.

This window will not be displayed in Integer Fields Values, Integer Fields Bit Pattern or Floating-point Components display modes. If it is visible when Integer Fields Values or Floating-point Components display mode is selected then it will be closed.

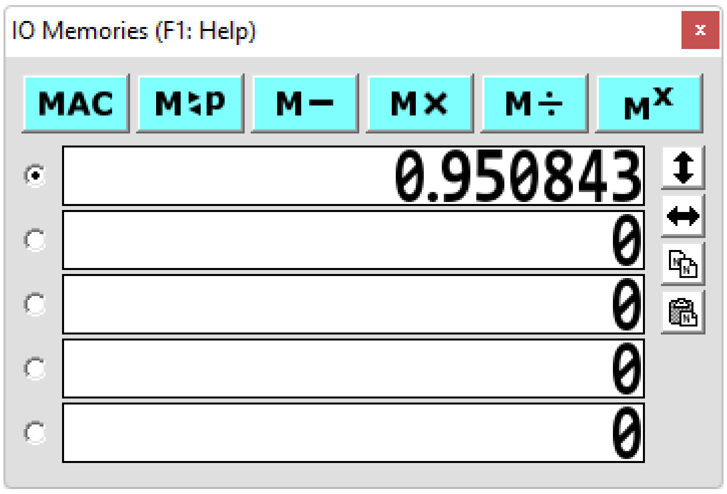
10.1 Organizations



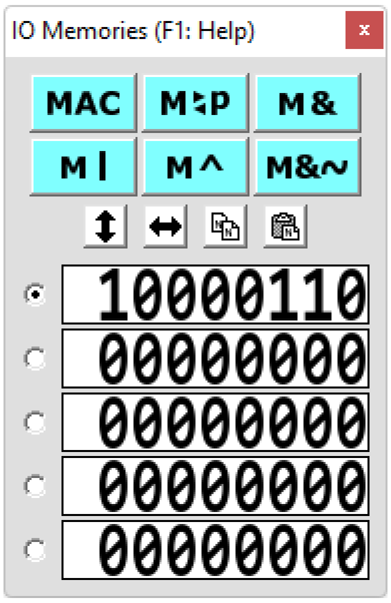
Wide, small digits, integer



Narrow, small digits, fixed/floating-point



Wide, medium digits, fixed/floating-point



Narrow, medium digits, bit pattern

The organization of the memories display window is determined by the following criteria:

The Memories Window Digit Size option setting currently selected in the window options. This setting can be toggled by clicking on the up/down arrow key in this window.

The Memories Window Width option setting currently selected in the window options. This setting can be toggled by clicking on the left/right arrow key in this window.

10.2 Current Memory Selection Buttons

The radio buttons are used to select the current memory register. All memory operation keys will use the selected memory register, with the exception of the Memory All Clear and Polynomial operations which use all of them.

The Polynomial operation uses the first displayed memory register as the constant term, and the following, in order, for the coefficients of increasing powers of the value stored in the X register.

10.3 Memory Register Displays

These display the current contents of the memory registers. They will be formatted according to the current Number Coding and Number Display options and the current Display Mode.

Double-clicking on one of these displays will perform a Memory Recall operation from the associated memory register to the X register, irrespective of which memory register is currently selected.

10.4 Keys

Clear All Memories

MAC **MCA** **M⇔0**

All the memory registers and previous contents registers are overwritten with a value of zero. For Bit Pattern display modes, the registers are overwritten with bit patterns of all ⓪s, this will only make a difference for integer excess coding.

No flags are affected.

Swap Memory with Previous Contents

M⇄P **Mspr**

The contents of the current memory register and its previous contents register are exchanged.

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the current memory register has the same contents as its previous contents register.

No flags are affected.

Subtract X from Memory

M− **Msub**

The value contained in the X register is subtracted from the value contained in the current memory register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is not displayed for a Bit Pattern Display Mode.

This key will be disabled under any of the following circumstances:

- Floating-point Fields Bit Pattern display mode
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains a value of exact-zero

No flags are affected.

Multiply Memory by X

M× **M∗** **Mmul**

The values contained in the X register and the current memory register are multiplied together and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is not displayed for a Bit Pattern Display Mode.

This key will be disabled under any of the following circumstances:

- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Number Coding Type” Number Coding Operation option is set to “Integer”, the “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains a value of one
- The “Number Coding Type” Number Coding Operation option is set to “Fixed-point” or “Floating-point”, the “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains an exact value of one
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register contains a value of zero

No flags are affected.

Divide Memory by X

M÷ **M/** **Mdiv**

The value contained in the current memory register is divided by the value contained in the X register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is not displayed for a Bit Pattern Display Mode.

This key will be disabled under any of the following circumstances:

- The “Number Coding Type” Number Coding Operation option is set to “Integer” or “Fixed-point”, and the X register contains a value of zero

- The “Number Coding Type” Number Coding Operation option is set to “Floating-point”, the “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected, and the X register contains a value of exact-zero
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern
- The “Number Coding Type” Number Coding Operation option is set to “Integer”, the “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains a value of one
- The “Number Coding Type” Number Coding Operation option is set to “Fixed-point” or “Floating-point”, the “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains an exact value of one
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register contains a value of zero

No flags are affected.

Modulo Memory by X

M% **Mmod**

The value contained in the current memory register is divided by the value contained in the X register and the remainder written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed for Integer Word Value display mode.

This key will be disabled under any of the following circumstances:

- The X register contains a value of zero
- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the current memory register contains a value of which is the same as the remainder when it is divided by the value contained in the X register, e.g., it is less than the value contained in the X register

No flags are affected.

Raise Memory to Power of X

M^X **Mpow**

The value contained in the current memory register is raised to the power of the value contained in the X register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed if the “Number Coding Type” Number Coding option is set to “Fixed-point” or “Floating-point”, but not in a Bit Pattern Display Mode.

This key will be disabled under any of the following circumstances:

- Floating-point Components display mode
- The “Prevent Mathematically Undefined Floating-point Operations” Calculator Operation option is selected and the X register contains a floating-point NaN or Pseudo class bit pattern

- The “Prevent Ineffective Operations” Calculator Operation option is selected, and the X register contains an exact value of one
- The “Prevent Ineffective Operations” Calculator Operation option is selected, the “Value of 0 to the power of 0” Calculator Operation option is set to “0”, the X register contains a value of exact-zero, and the current memory register contains a value of zero

No flags are affected.

Bitwise AND X to Memory

M& **Mand**

A “bitwise and” operation is performed on the bit patterns contained in the X register and the current memory register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed in Integer Word Bit Pattern and Fixed-point Word Bit Pattern display modes.

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the current memory register contains a bit pattern which will remain unchanged if “bitwise and”ed with the bit pattern contained in the X register.

No flags are affected.

Bitwise Inclusive OR X to Memory

M| **Mor**

A “bitwise inclusive or” operation is performed on the bit patterns contained in the X register and the current memory register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed in Integer Word Bit Pattern and Fixed-point Word Bit Pattern display modes.

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the current memory register contains a bit pattern which will remain unchanged if “bitwise inclusive or”ed with the bit pattern contained in the X register.

No flags are affected.

Bitwise Exclusive OR X to Memory

M^ **Mxor** **Meor**

A “bitwise exclusive or” operation is performed on the bit patterns contained in the X register and the current memory register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed in Integer Word Bit Pattern and Fixed-point Word Bit Pattern display modes.

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the current memory register contains a bit pattern which will remain unchanged if “bitwise exclusive or”ed with the bit pattern contained in the X register.

No flags are affected.

Bitwise Bit Clear X to Memory

M&∾ Mbic

A “bitwise bit clear” operation is performed on the bit pattern contained in the current memory register with the bit pattern mask contained in the X register and the result written to the current memory register. If this results in a change to the contents of the current memory register then the original contents are written to its previous contents register.

This key is only displayed in Integer Word Bit Pattern and Fixed-point Word Bit Pattern display modes.

This key will be disabled if the “Prevent Ineffective Operations” Calculator Operation option is selected and the current memory register contains a bit pattern which will remain unchanged if “bitwise bit clear”ed with the bit pattern contained in the X register.

No flags are affected.

Switch Memory Display Digit Size

⬍ ↕

The digit size used in the memory displays is switched between small and medium.

No flags are affected.

Switch Memory Display Width

⬌ ↔

The width of the memories window is switched between narrow and wide.

No flags are affected.

Copy current Memory to Clipboard

Keyboard Shortcut: Ctrl+C

The contents of the current memory register are copied to the clipboard. The format of the text copied to the clipboard is determined by how the contents are currently displayed and the Clipboard options.

No flags are affected.

Paste current Memory from Clipboard

Keyboard Shortcut: Ctrl+V

The contents of the clipboard are pasted to the current memory register. The interpretation of the text pasted from the clipboard is determined by the Number Display options, the Clipboard options and the Display Mode.

No flags are affected.

---

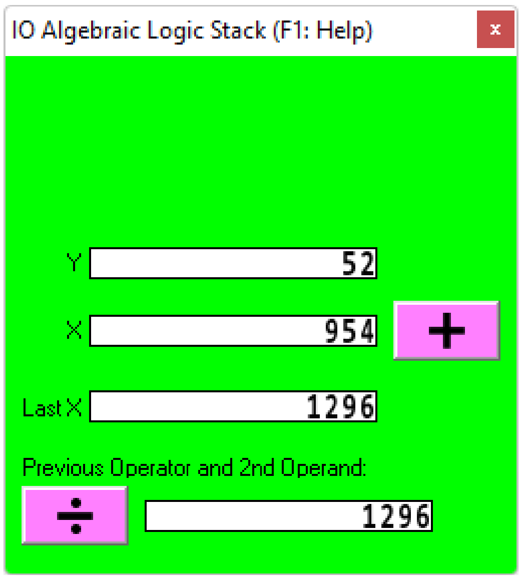
11 STACK DISPLAY WINDOW

This window displays the current contents of the Registers Stack. It can be displayed by selecting the “Stack…” entry of the “Windows” menu.

For the Algebraic Logic calculator type it also displays the current contents of the operator stack and shows the operation and 2nd operand used when repeated by using the = key.

The Stack Display Window does not have any active keys so it cannot be used to initiate any IO operations.

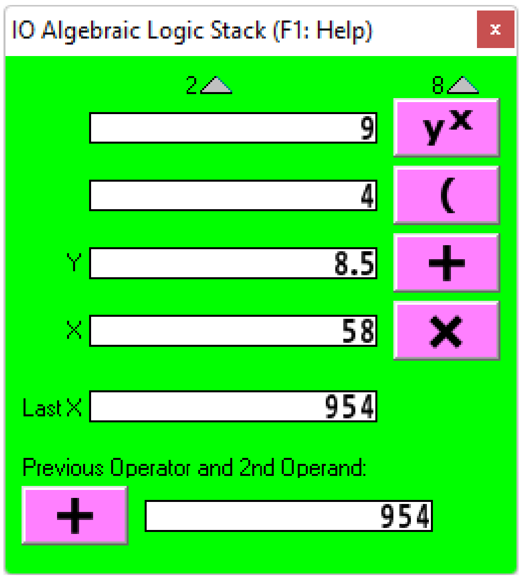
11.1 Organizations



IO Algebraic Logic Stack (F1: Help)

Y 52
X 954 +
Last X 1296
Previous Operator and 2nd Operand:
÷ 1296

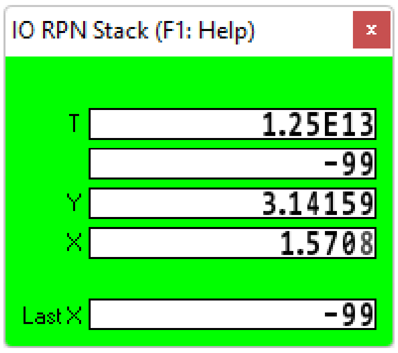
Algebraic Logic, All Entries



IO Algebraic Logic Stack (F1: Help)

2△ 8△
9 y^x
4 (
Y 8.5 +
X 58 ×
Last X 954
Previous Operator and 2nd Operand:
+ 954

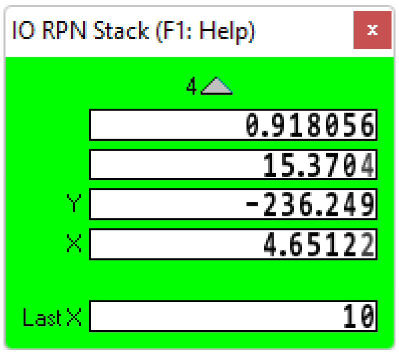
Algebraic Logic, Most Recent Entries



IO RPN Stack (F1: Help)

T 1.25E13
-99
Y 3.14159
X 1.5708
Last X -99

RPN, Four Stack Registers



IO RPN Stack (F1: Help)

4△
0.918056
15.3704
Y -236.249
X 4.65122
Last X 10

RPN, More Than Four Stack Registers

The organization of the stack display window is determined by the setting of the “Calculator Type” Calculator Operation option.

11.2 Register Stack Contents

The register stack is displayed for both calculator types. For the Algebraic Logic calculator type it is displayed on the left-hand side of the display window.

The most recently pushed register contents are referred to as being at the bottom of the stack and are depicted at the bottom of the diagrammatic representation of the stack in this display.

The register at the bottom of the stack is the X register, and is labeled as such in this display. The next register on the stack is the Y register, which is also labeled if displayed.

For the Algebraic Logic calculator type, only the X register and registers which have been pushed onto the stack are displayed. If there are no pending operators then only the X register will be displayed. If there are more than three registers pushed onto the stack in addition to the X register, then a numbered up-pointing arrow will be displayed, the number identifying how many un-displayed stack registers there are above the topmost displayed register.

There is a convention for the register at the top of the allocated stack registers for calculators which use Reverse Polish Notation to be referred to as the T register. If the register stack has four or fewer registers allocated in the Calculator Operation Options for a Calculator Type of Reverse Polish Notation then the T register will also be labeled in this display.

If there are more than four stack registers for the Reverse Polish Notation calculator type, as specified in the Calculator Operation options, then a numbered up-pointing arrow will be displayed, the number identifying how many un-displayed stack registers there are above the topmost displayed register.

11.3 Operator Stack Contents

The operator stack is only used, and hence displayed here, for the Algebraic Logic calculator type. It is displayed on the right-hand side of the display window.

Each entry in the operator stack identifies a pending operation which has yet to be performed due to, a) the relative precedences of the operators (assuming that the use of operator precedences has been enabled in the Calculator Operation Options), b) the use of one or more as yet unmatched open-brackets, or c) the incomplete entry of the final operand.

If there are more than four operators on the stack, then a numbered up-pointing arrow will be displayed, the number identifying how many un-displayed operators there are above the topmost displayed operator.

11.4 Last X Register Contents

The Last X register is displayed beneath the register stack.

It contains the most recent different value, or more accurately bit pattern, that was stored in the X register.

Its contents are swapped with that of the X register by the "Last X" operation.

11.5 Previous Operator and 2nd Operand

The Previous Operator and 2nd Operand are only used, and hence displayed here, if the “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the “= Key Repeats Previous Op” Calculator Operation option is selected. It is displayed beneath the Last X register.

It consists of the most recently processed operator and its second operand following the use of the = key.

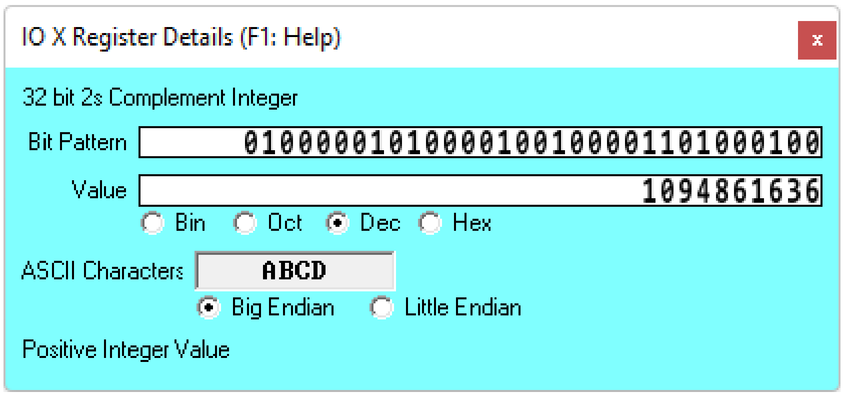
This operation can be repeated by clicking the = key again after entering a new first operand, without entering any other operator.

12 X REGISTER DETAILS WINDOW

This window displays details of the current contents of the X register. It can be displayed by selecting the “X Register Details…” entry of the “Windows” menu, or by double-clicking on any of:

- The main number display in the keypad window
- The main number display in the detached display window
- The X register display in the stack window

12.1 Organizations



IO X Register Details (F1: Help)

32 bit 2s Complement Integer

Bit Pattern 01000001010000100100001101000100

Value 1094861636

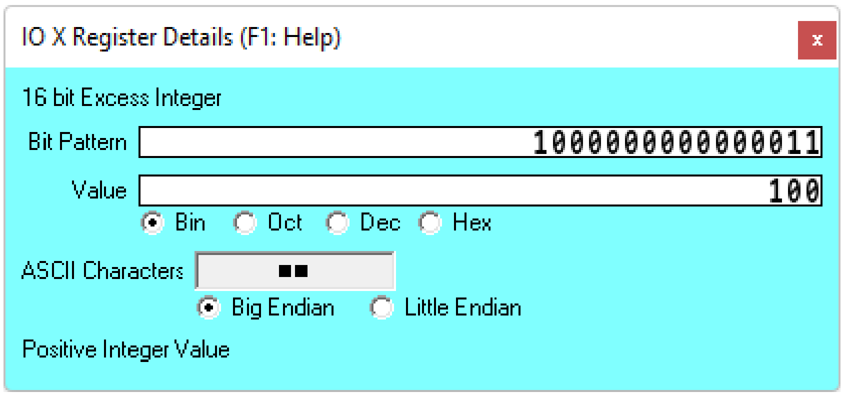
Bin Oct [✓ Dec] Hex

ASCII Characters ABCD

[✓ Big Endian] Little Endian

Positive Integer Value

Integer, Multiple of 8 bits



IO X Register Details (F1: Help)

16 bit Excess Integer

Bit Pattern 1000000000000011

Value 100

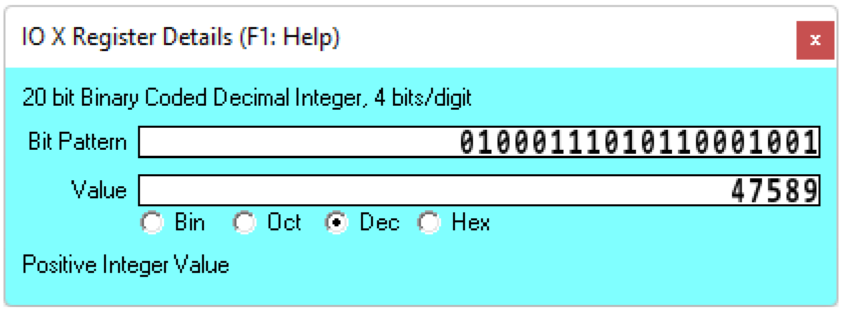
[✓ Bin] Oct Dec Hex

ASCII Characters ■■

[✓ Big Endian] Little Endian

Positive Integer Value

Integer, Multiple of 8 bits



IO X Register Details (F1: Help)

20 bit Binary Coded Decimal Integer, 4 bits/digit

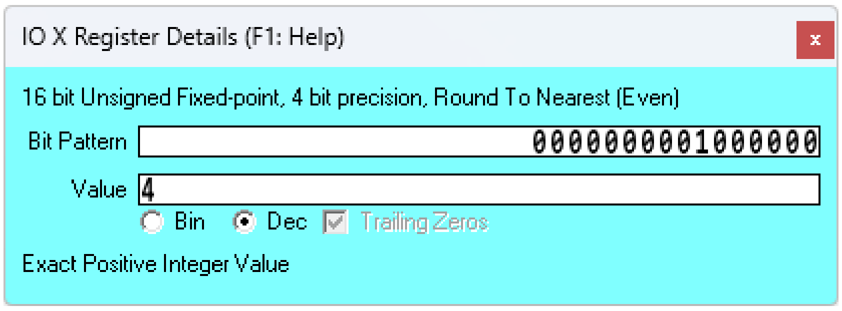
Bit Pattern 01000111010110001001

Value 47589

Bin Oct [✓ Dec] Hex

Positive Integer Value

Integer, Not Multiple of 8 bits


IO X Register Details (F1: Help)

16 bit Unsigned Fixed-point, 4 bit precision, Round To Nearest (Even)

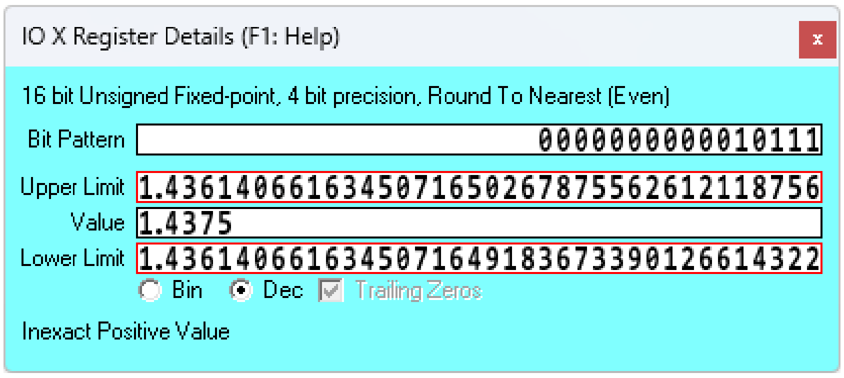
Bit Pattern `0000000001000000`

Value `4`

( ) Bin (•) Dec [✓] Trailing Zeros

Exact Positive Integer Value

Fixed-point, Exact Value


IO X Register Details (F1: Help)

16 bit Unsigned Fixed-point, 4 bit precision, Round To Nearest (Even)

Bit Pattern `0000000000010111`

Upper Limit `1.4361406616345071650267875562612118756`

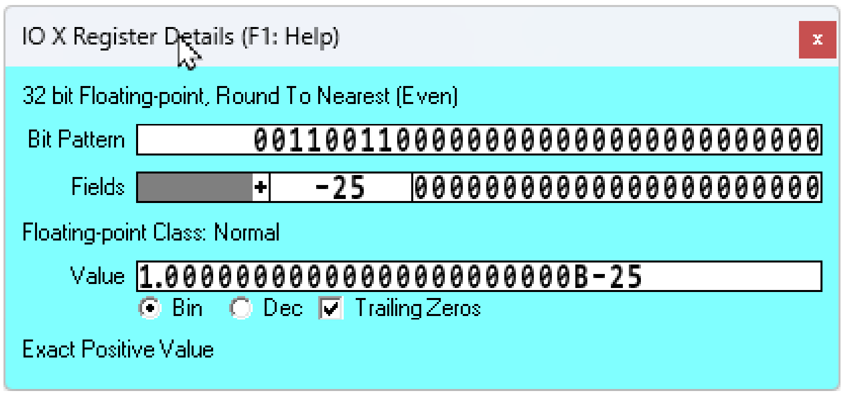
Value `1.4375`

Lower Limit `1.4361406616345071649183673390126614322`

( ) Bin (•) Dec [✓] Trailing Zeros

Inexact Positive Value

Fixed-point, Inexact Value


IO X Register Details (F1: Help)

32 bit Floating-point, Round To Nearest (Even)

Bit Pattern `00110011000000000000000000000000`

Fields `+` `-25` `00000000000000000000000`

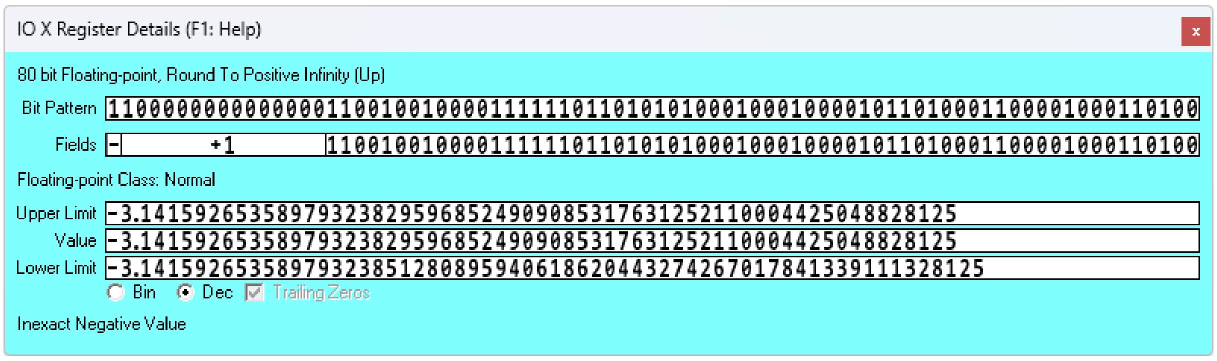
Floating-point Class: Normal

Value `1.00000000000000000000000B-25`

(•) Bin ( ) Dec [✓] Trailing Zeros

Exact Positive Value

Floating-point, Exact Value


IO X Register Details (F1: Help)

80 bit Floating-point, Round To Positive Infinity (Up)

Bit Pattern `11000000000000001100100100001111110110101010001000100001011010001100001000110100`

Fields `-` `+1` `1100100100001111110110101010001000100001011010001100001000110100`

Floating-point Class: Normal

Upper Limit `-3.14159265358979323829596852490908531763125211000442504882812 5`

Value `-3.141592653589793238295968524909085317631252110004425048828125`

Lower Limit `-3.141592653589793238512808959406186204432742670178413391113281 25`

( ) Bin (•) Dec [✓] Trailing Zeros

Inexact Negative Value

Floating-point, Inexact Value

The organization of the X Register Details window is determined by the following criteria:

The setting of the “Number Coding Type” Number Coding option

If the “Number Coding Type” Number Coding option is set to “Integer”, then whether or not the “Integer Width” Number Coding option is currently set to a multiple of eight

If the “Number Coding Type” Number Coding option is set to “Fixed-point” or “Floating-point”, then whether or not the X register currently contains an exact value

The width of this window may be modified by dragging the left or right edge of the window with the mouse.

12.2 Display Elements

The X Register Details window contains a number of display elements. Some are always displayed, which others are displayed depends on a combination of the Number Coding options and the current contents of the X register.

12.2.1 Coding

The details of the current Number Coding options are always displayed. These are used to determine the values displayed.

12.2.2 Bit Pattern

The bit pattern currently contained in the X register is always displayed.

12.2.3 Fields

Fields are only displayed for floating-point number coding types. The field layout used will be determined by the current floating-point coding.

12.2.4 Floating-point Class

The floating-point class is only displayed for floating-point number coding types.

The following floating-point classes may be reported for any floating-point coding:

Zero
Normal
Denormal
Positive Infinity
Negative Infinity
Quiet Not a Number
Signaling Not a Number

The following floating-point classes may be reported for 80-bit floating-point coding:

Unnormal
Pseudo Denormal
Pseudo Positive Infinity
Pseudo Negative Infinity
Pseudo Quiet Not a Number
Pseudo Signaling Not a Number

12.2.5 Value

The value determined by the current coding applied to the bit pattern currently contained in the X register, both of which are displayed above, is always displayed.

The format used is as currently defined by the current Number Display options, with the following exceptions:

All significant digits are always displayed for fixed/floating-point codings

The number base used is determined by the currently selected radio button below

The digits are always left justified within the display area for fixed/floating-point codings

The current Number Coding options are always used to determine the displayed vale irrespective of the current setting of the “Display as for Unsigned Coding if Base is a power of 2” Number Display option

12.2.6 Upper/Lower Limit

The upper limit and lower limit values are only displayed for fixed/floating-point codings, and only when an inexact value is currently contained in the X register.

The exact value, which cannot be represented using the current coding, is less than the displayed upper limit and greater than the displayed lower limit.

The limit values always use 80-bit floating-point coding irrespective of the current coding options, so may or may not be exact values depending on whether or not the exact limit can be represented exactly using 80-bit floating-point coding.

The formatting used is the same as for the X register value, as described above.

Note that, depending on the current coding (including the rounding type), the displayed value contained in the X register may not be between these limits. This appears to be just wrong, but is a very common occurrence; this issue is discussed in detail elsewhere in this document.

12.2.7 Number Base

These radio button controls are used to select the number base used for the displayed value and limits. They override the settings in the Number Display options.

For integer number coding types, the available options are binary, octal, decimal and hexadecimal.

For fixed/floating-point codings the only available options are binary and decimal.

12.2.8 ASCII Characters

ASCII characters are only displayed for the integer number coding type, and then only if the width of the bit pattern is a multiple of eight bits.

12.2.9 ASCII Characters Endianness

These radio button controls are used to determine the order in which the characters for each byte of the bit pattern are displayed. This order will be with the first character displayed being that for the byte which will be stored in the lowest addressed byte in memory for the selected endianness.

If Big Endian is selected then the first character displayed will be that for the most significant byte of the bit pattern.

If Little Endian is selected then the first character displayed will be that for the least significant byte of the bit pattern.

12.2.10 Value Type

The type of value currently stored in the X register is always displayed.

The display will contain one or more of the following items for the value currently contained in the X register:

No Value	Floating-point NaN or Infinity Class
Exact	Fixed/floating-point exact value
Inexact	Fixed/floating-point inexact value
Zero	Zero value
Positive	Positive value
Negative	Negative value
Integer	Integer value (all number coding types)

13 HELP WINDOWS

Apart from the popup windows, pressing the F1 key on your keyboard when any of IO's windows has the keyboard focus (or if it is displaying an error message window) will display the help window for that window. This will contain a description for each control in the parent window. The help window can be moved, resized and scrolled. The help window will automatically be closed if the parent window is closed.

The controls in a parent option set window may still be used while the help window is displayed. Although the controls in the help window may respond to user interaction, e.g., clicking on a radio button will de-select other radio buttons in its group, they only affect the appearance of the help window, they do not affect IO's operation.

There are two help windows which are not associated with a specific IO window, these are the Keypad Menu help window and the Keyboard Shortcuts help window.

Several help windows may be displayed simultaneously if desired.

The types of help window are:

- Keypad Menu Help Window
- Keyboard Shortcut Help Window
- Option Set Window Help Windows
- Keypad Help Window
- Other IO Window Help Windows

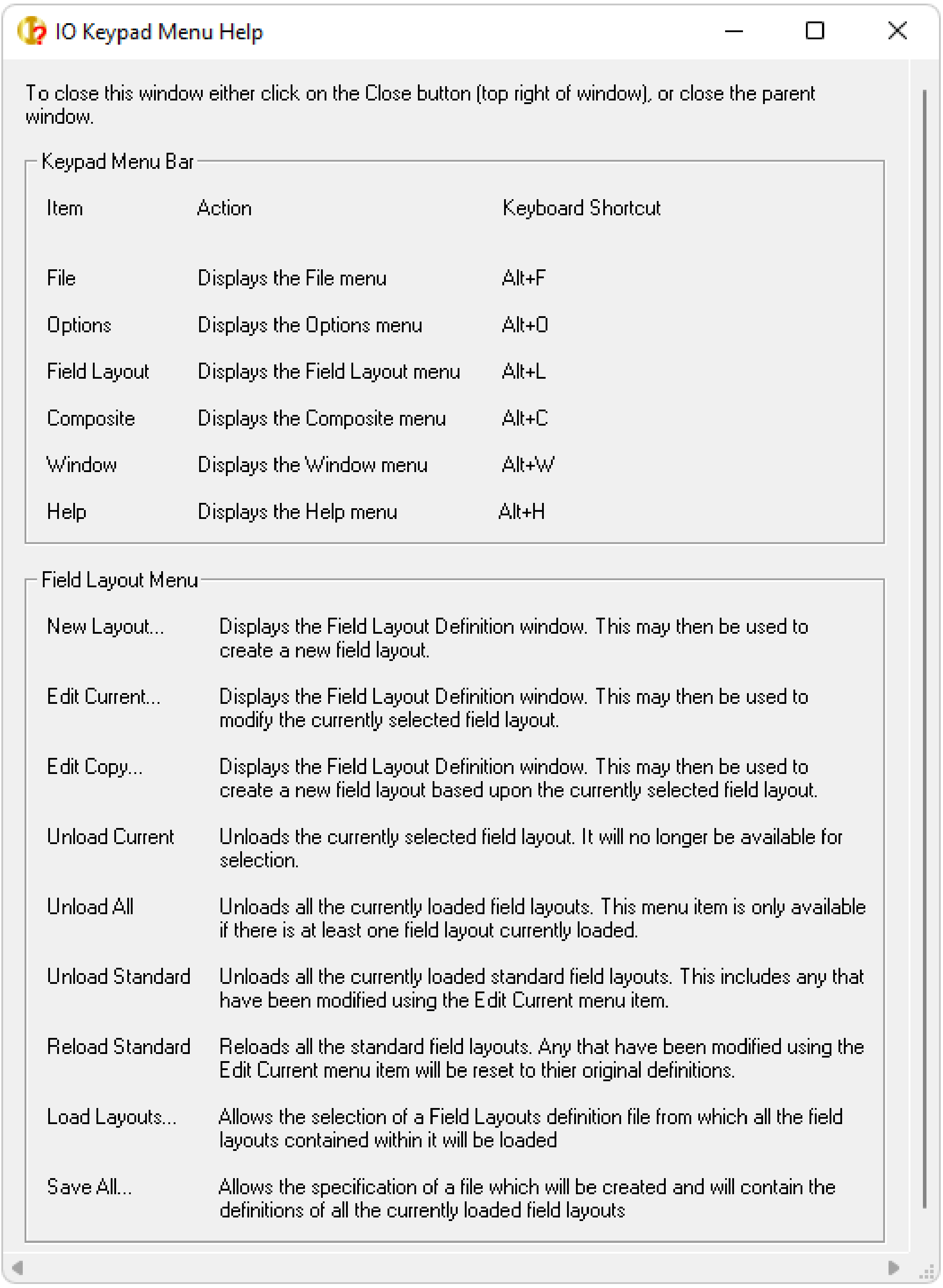
Although they are similar there are some differences.

13.1 Keypad Menu Help Window

This is displayed by displaying a keypad submenu and, moving the mouse pointer into the displayed submenu, and then pressing the F1 key on your keyboard. The submenu may belong to one of the keypad window menu-bar entries, or may be the expression editor popup menu.

The contents of this window are dynamic and change if you select different submenus. Only help information for the currently displayed submenu is displayed

The contents of this window may be scrolled by using the Scroll Bar controls, the mouse wheel, the Home, End, Page Up, Page Down, Up Arrow, Down Arrow, Left Arrow and Right Arrow keyboard keys.


IO Keypad Menu Help

To close this window either click on the Close button (top right of window), or close the parent window.

Keypad Menu Bar

Item	Action	Keyboard Shortcut
File	Displays the File menu	Alt+F
Options	Displays the Options menu	Alt+O
Field Layout	Displays the Field Layout menu	Alt+L
Composite	Displays the Composite menu	Alt+C
Window	Displays the Window menu	Alt+W
Help	Displays the Help menu	Alt+H

Field Layout Menu

New Layout...	Displays the Field Layout Definition window. This may then be used to create a new field layout.
Edit Current...	Displays the Field Layout Definition window. This may then be used to modify the currently selected field layout.
Edit Copy...	Displays the Field Layout Definition window. This may then be used to create a new field layout based upon the currently selected field layout.
Unload Current	Unloads the currently selected field layout. It will no longer be available for selection.
Unload All	Unloads all the currently loaded field layouts. This menu item is only available if there is at least one field layout currently loaded.
Unload Standard	Unloads all the currently loaded standard field layouts. This includes any that have been modified using the Edit Current menu item.
Reload Standard	Reloads all the standard field layouts. Any that have been modified using the Edit Current menu item will be reset to thier original definitions.
Load Layouts...	Allows the selection of a Field Layouts definition file from which all the field layouts contained within it will be loaded
Save All...	Allows the specification of a file which will be created and will contain the definitions of all the currently loaded field layouts

Example: Keypad Menu Help Window when Field Layout Menu is displayed

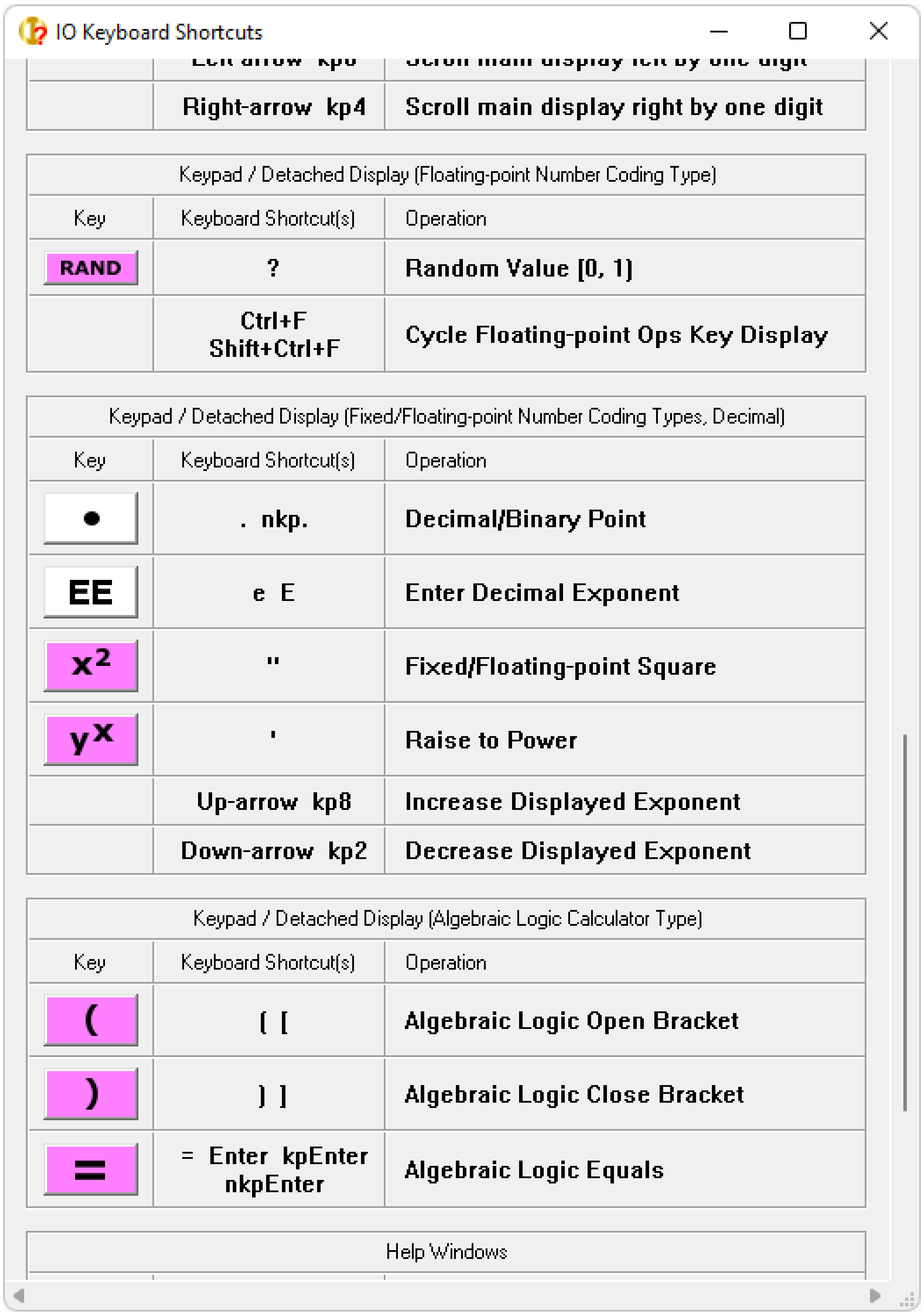
13.2 Keyboard Shortcut Help Window

This is displayed by selecting the “Keyboard Shortcuts” entry in the Help menu.

The contents of this window are dynamic and change while displayed so that only currently available keyboard shortcuts are displayed. Which keyboard shortcuts are available are determined by a number of things, including the current number coding type and which windows are displayed.

Keys displayed in in this window will have the same label and colors as the associated key in the parent window. These are also dynamic and change if you change the labels/colors in the Key Display options.

The contents of this window may be scrolled by using the Scroll Bar controls, the mouse wheel, the Home, End, Page Up, Page Down, Up Arrow, Down Arrow, Left Arrow and Right Arrow keyboard keys.



IO Keyboard Shortcuts

	Right-arrow kp4	Scroll main display right by one digit

Keypad / Detached Display (Floating-point Number Coding Type)

Key	Keyboard Shortcut(s)	Operation
RAND	?	Random Value [0, 1)
	Ctrl+F Shift+Ctrl+F	Cycle Floating-point Ops Key Display

Keypad / Detached Display (Fixed/Floating-point Number Coding Types, Decimal)

Key	Keyboard Shortcut(s)	Operation
●	. nkp.	Decimal/Binary Point
EE	e E	Enter Decimal Exponent
x^2	"	Fixed/Floating-point Square
y^x	'	Raise to Power
	Up-arrow kp8	Increase Displayed Exponent
	Down-arrow kp2	Decrease Displayed Exponent

Keypad / Detached Display (Algebraic Logic Calculator Type)

Key	Keyboard Shortcut(s)	Operation
(	([	Algebraic Logic Open Bracket
)	)]	Algebraic Logic Close Bracket
=	= Enter kpEnter nkpEnter	Algebraic Logic Equals

Help Windows

Example: Section of Keyboard Shortcuts Help Window

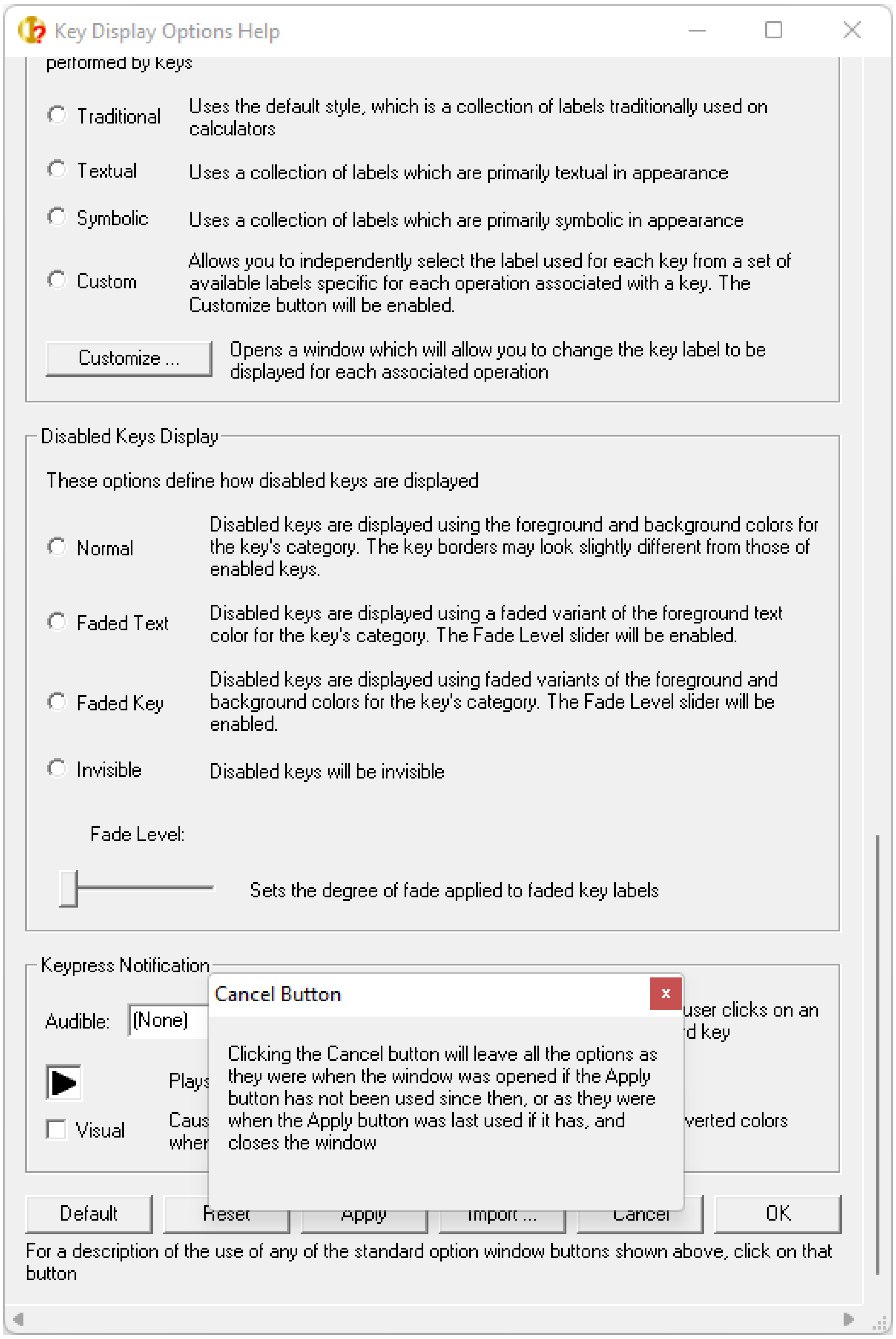
13.3 Option Set Window Help Windows

As the name implies, these are displayed when you press F1 while using an Option Set Window.

As the visibility of the controls in the parent window do not change so the contents of its help window do not change.

As all Option Set windows contain the six standard buttons, they are displayed at the bottom of the help window, and the help information for each button can be displayed in a popup window by clicking on the button in the help window.

The contents of these help windows may be scrolled by using the Scroll Bar controls or the mouse wheel.


Key Display Options Help

performed by keys

Traditional — Uses the default style, which is a collection of labels traditionally used on calculators

Textual — Uses a collection of labels which are primarily textual in appearance

Symbolic — Uses a collection of labels which are primarily symbolic in appearance

Custom — Allows you to independently select the label used for each key from a set of available labels specific for each operation associated with a key. The Customize button will be enabled.

Customize ... — Opens a window which will allow you to change the key label to be displayed for each associated operation

Disabled Keys Display

These options define how disabled keys are displayed

Normal — Disabled keys are displayed using the foreground and background colors for the key's category. The key borders may look slightly different from those of enabled keys.

Faded Text — Disabled keys are displayed using a faded variant of the foreground text color for the key's category. The Fade Level slider will be enabled.

Faded Key — Disabled keys are displayed using faded variants of the foreground and background colors for the key's category. The Fade Level slider will be enabled.

Invisible — Disabled keys will be invisible

Fade Level: — Sets the degree of fade applied to faded key labels

Keypress Notification

Audible: (None) … user clicks on an … d key

Plays

Visual — Caus… wher… verted colors

Cancel Button

Clicking the Cancel button will leave all the options as they were when the window was opened if the Apply button has not been used since then, or as they were when the Apply button was last used if it has, and closes the window

Default | Reset | Apply | Import ... | Cancel | OK

For a description of the use of any of the standard option window buttons shown above, click on that button

Example: Section of Help Window for Key Display Option Set Window, showing Button Help Popup Window

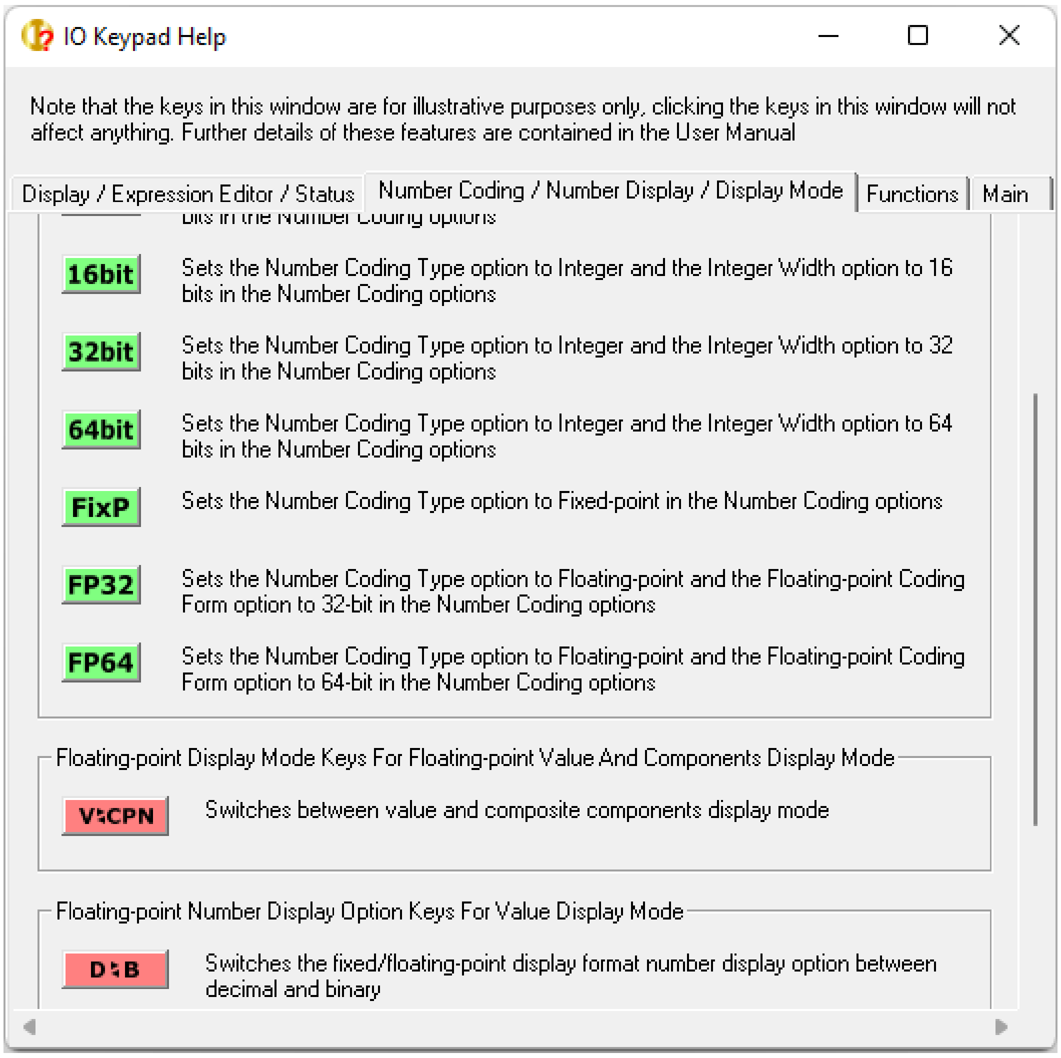
13.4 Keypad Help Window

Due to the large number of keys that can be visible in the Keypad its help window is subdivided into four pages, only one of which is displayed at any one time. Which page is displayed is selected by clicking on the corresponding tab near the top of the help window.

The keys, and other controls, which are visible in the Keypad change as you use it, the contents of the help pages are dynamic and change while displayed so that they usually only display help for currently visible keys and controls in the Keypad window.

Any keys displayed in a help page will have the same label and colors as the associated key in the Keypad window. These are also dynamic and change if you change the labels/colors in the Key Display options.

As well as the Scroll Bar controls and the mouse wheel, the contents of a help page may also be scrolled by using the Home, End, Page Up, Page Down, Up Arrow, Down Arrow, Left Arrow and Right Arrow keyboard keys.



IO Keypad Help

Note that the keys in this window are for illustrative purposes only, clicking the keys in this window will not affect anything. Further details of these features are contained in the User Manual

Display / Expression Editor / Status
Number Coding / Number Display / Display Mode
Functions
Main

16bit	Sets the Number Coding Type option to Integer and the Integer Width option to 16 bits in the Number Coding options
32bit	Sets the Number Coding Type option to Integer and the Integer Width option to 32 bits in the Number Coding options
64bit	Sets the Number Coding Type option to Integer and the Integer Width option to 64 bits in the Number Coding options
FixP	Sets the Number Coding Type option to Fixed-point in the Number Coding options
FP32	Sets the Number Coding Type option to Floating-point and the Floating-point Coding Form option to 32-bit in the Number Coding options
FP64	Sets the Number Coding Type option to Floating-point and the Floating-point Coding Form option to 64-bit in the Number Coding options

Floating-point Display Mode Keys For Floating-point Value And Components Display Mode

V⁝CPN	Switches between value and composite components display mode

Floating-point Number Display Option Keys For Value Display Mode

D⁝B	Switches the fixed/floating-point display format number display option between decimal and binary

Example: Section of Keypad Help Window

The four pages are:

Display / Expression Editor / Status

This contains help for the currently visible keys and other controls in the Display, Expression Editor and Status regions of the Keypad.

Number Coding / Number Display / Display Mode

This contains help for the currently visible keys and other controls in the Number Coding region and the currently visible keys in the Functions region which are related to the Number Display options and Display Mode.

Functions

This contains help for the currently available function and operator keys in the Functions region of the Keypad.

Main

This contains help for the currently available keys in the Main region of the Keypad.

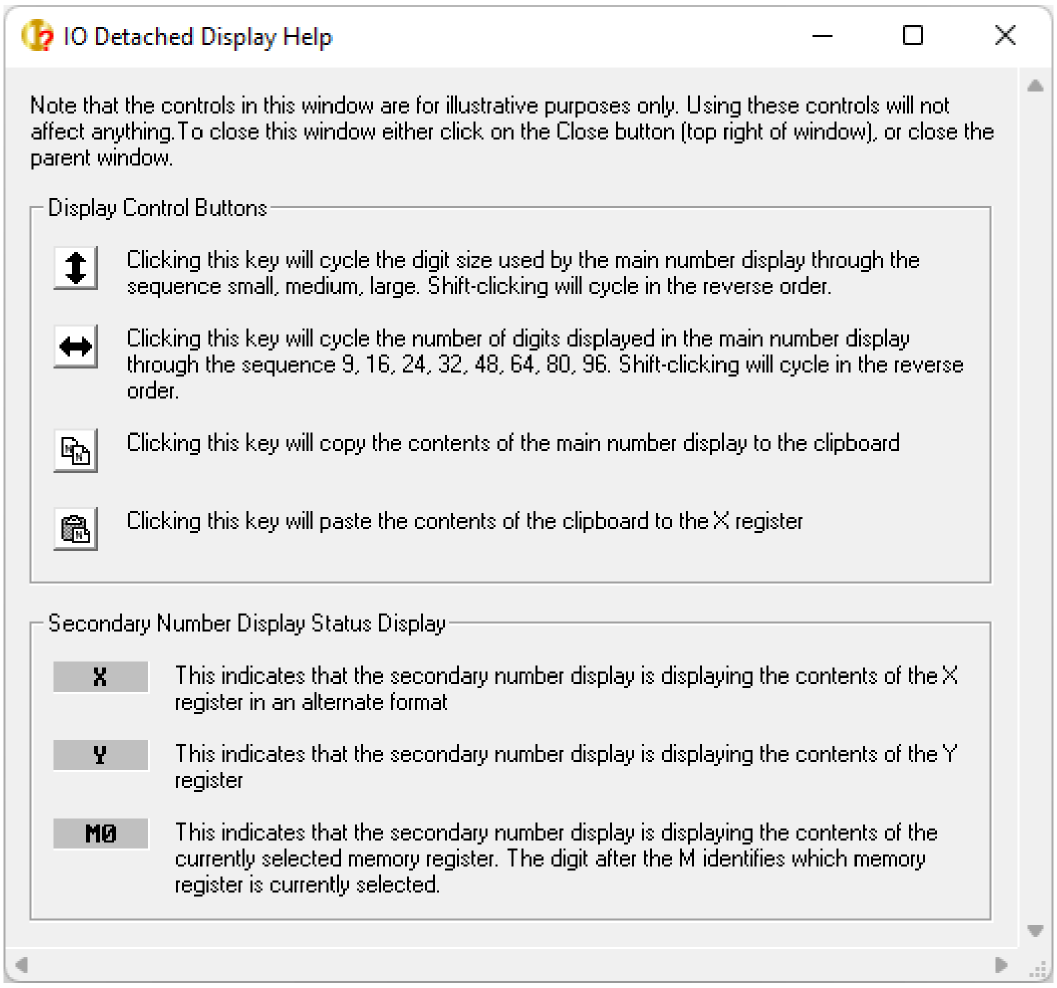
13.5 Other Window Help Windows

The following IO windows each have their own Help window:

- Detached Number Display
- Memories Display
- Stack Display
- X Register Details
- Field Layout
- Composite Components

Most of these are dynamic like the Keypad help pages and can change in response to changes in the parent window.

Also like the Keypad help pages, the contents of these help windows may be scrolled by using the Home, End, Page Up, Page Down, Up Arrow, Down Arrow, Left Arrow and Right Arrow keyboard keys.



Note that the controls in this window are for illustrative purposes only. Using these controls will not affect anything.To close this window either click on the Close button (top right of window), or close the parent window.

Display Control Buttons

Clicking this key will cycle the digit size used by the main number display through the sequence small, medium, large. Shift-clicking will cycle in the reverse order.

Clicking this key will cycle the number of digits displayed in the main number display through the sequence 9, 16, 24, 32, 48, 64, 80, 96. Shift-clicking will cycle in the reverse order.

Clicking this key will copy the contents of the main number display to the clipboard

Clicking this key will paste the contents of the clipboard to the X register

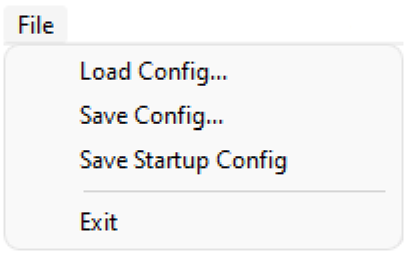
Secondary Number Display Status Display

X	This indicates that the secondary number display is displaying the contents of the X register in an alternate format
Y	This indicates that the secondary number display is displaying the contents of the Y register
M0	This indicates that the secondary number display is displaying the contents of the currently selected memory register. The digit after the M identifies which memory register is currently selected.

Example: Detached Display Help Window

14 MENUS

14.1 File Menu



This is displayed by clicking on “File” in the keypad menu, or by using the Alt+F keyboard shortcut.

14.1.1 Load Config...

Selecting this item will allow the selection of a configuration file which will then be used to set all option settings.

This menu item will be disabled under any of the following circumstances:

- Display mode type is not Word Value
- IO has not just been started and the “All Clear” key has not just been used

14.1.2 Save Config...

Selecting this item will save all option settings to a specified configuration file.

This menu item will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Components display mode

14.1.3 Save Startup Config

Selecting this item will initially set the “Startup Config” Calculator Operation option to “Saved Startup” if it is not already, and then store the current configuration so that it will be automatically loaded when the “Startup Config” Calculator Operation option is set to “Saved Startup”, i.e., not subsequently changed to something different.

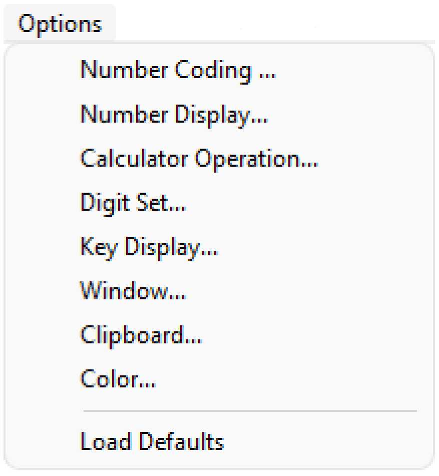
This menu item will be disabled under any of the following circumstances:

- Integer Fields Values display mode
- Integer Fields Bit Pattern display mode
- Floating-point Components display mode

14.1.4 Exit

Selecting this item will terminate IO, as will clicking on the keypad window’s close button. The current configuration will be saved as the “last used configuration” before termination.

14.2 Options Menu



This is displayed by clicking on “Options” in the keypad menu, or by using the Alt+O keyboard shortcut.

14.2.1 Number Coding…

Selecting this item will display the Number Coding Option Set window.

This menu item will be disabled if the display mode type is not Word Value.

14.2.2 Number Display…

Selecting this item will display the Number Display Option Set window.

This menu item will be disabled if the display mode type is not Word Value.

14.2.3 Calculator Operation…

Selecting this item will display the Calculator Operation Option Set window.

14.2.4 Digit Set…

Selecting this item will display the Digit Set Option Set window.

This menu item will be disabled if the display mode type is not Word Value.

14.2.5 Key Display…

Selecting this item will display the Key Display Option Set window.

14.2.6 Window…

Selecting this item will display the Window Option Set window.

14.2.7 Clipboard…

Selecting this item will display the Clipboard Option Set window.

14.2.8 Color…

Selecting this item will display the Color Option Set window.

14.2.9 Load Defaults

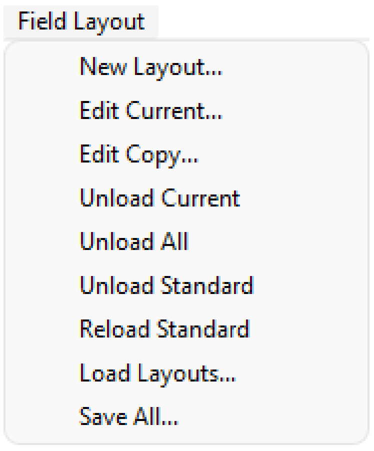
Selecting this item will load the default configuration.

This menu item will be disabled under any of the following circumstances:

- Display mode type is not Word Value
- IO has not just been started and the All Clear key has not just been used
- The contents of a register on the register stack or a memory register prevents a change to the default Number Coding options due to the current settings of the “Number Coding Option Change Handling” Calculator Operation options

---

14.3 Field Layout Menu



This is displayed by clicking on “Field Layout” in the keypad menu, or by using the Alt+L keyboard shortcut.

14.3.1 New Layout...

Selecting this item will display the Field Layout Definition window. This may then be used to create a new field layout.

14.3.2 Edit Current...

Selecting this item will display the Field Layout Definition window. This may then be used to modify the currently selected field layout.

This menu item will be disabled under any of the following circumstances:

- Display mode is not Integer Fields Values
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a bit pattern with at least one bit set to ①

14.3.3 Edit Copy...

Selecting this item will display the Field Layout Definition window. This may then be used to create a new field layout based upon the currently selected field layout.

This menu item will be disabled if the display mode is not Integer Fields Values or Integer Fields Bit Pattern.

14.3.4 Unload Current

Selecting this item will unload the currently selected field layout. It will no longer be available for selection.

This menu item will be disabled under any of the following circumstances:

- Display mode is not Integer Fields Values or Integer Fields Bit Pattern
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

14.3.5 Unload All

Selecting this item will unload all the currently loaded field layouts.

This menu item will be disabled under any of the following circumstances:

- There are no field layouts currently loaded
- Display mode is Integer Fields Values or Integer Fields Bit Pattern, the “Calculator Type” Calculator Operation option is set to “Algebraic Logic”, and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- Display mode is Integer Fields Values or Integer Fields Bit Pattern, the “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

14.3.6 Unload Standard

Selecting this item will unload all the currently loaded standard field layouts. This includes any that have been modified using the Edit Current menu item.

14.3.7 Reload Standard

Selecting this item will reload all the standard field layouts. Any that have been modified using the Edit Current menu item will be reset to their original definitions.

14.3.8 Load Layouts…

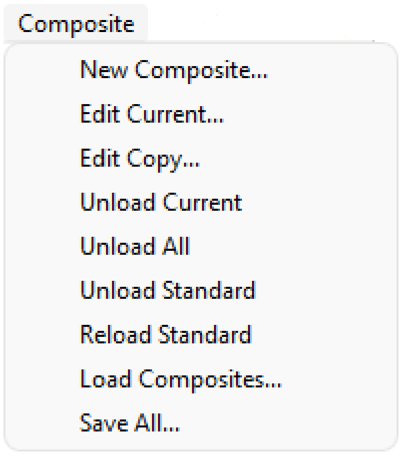
Selecting this item will allow the selection of a Field Layouts definition file from which all the field layouts contained within it will be loaded.

14.3.9 Save All…

Selecting this item will allow the specification of a file which will be created and will contain the definitions of all the currently loaded field layouts.

This menu item will be disabled if there are no field layouts currently loaded.

14.4 Composite Menu



This is displayed by clicking on “Composite” in the keypad menu, or by using the Alt+C keyboard shortcut.

14.4.1 New Composite...

Selecting this item will display the Composite Definition window. This may then be used to create a new composite.

14.4.2 Edit Current...

Selecting this item will display the Composite Definition window. This may then be used to modify the currently selected composite.

This menu item will be disabled under any of the following circumstances:

- Display mode is not Floating-point Components
- Any register on the register stack contains a non-zero value
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator

14.4.3 Edit Copy...

Selecting this item will display the Composite Definition window. This may then be used to create a new composite based upon the currently selected composite.

This menu item will be disabled if the display mode is not Floating-point Components.

14.4.4 Unload Current

Selecting this item will unload the currently selected composite. It will no longer be available for selection.

This menu item will be disabled under any of the following circumstances:

- Display mode is not Floating-point Components
- The “Calculator Type” Calculator Operation option is set to “Algebraic Logic” and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- The “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

14.4.5 Unload All

Selecting this item will unload all the currently loaded composites.

This menu item will be disabled under any of the following circumstances:

- There are no composites currently loaded
- Display mode is Floating-point Components, the “Calculator Type” Calculator Operation option is set to “Algebraic Logic”, and the operator stack is not empty, i.e., it contains at least one pending operator or open bracket indicator
- Display mode is Floating-point Components, the “Calculator Type” Calculator Operation option is set to “Reverse Polish Notation” and any register on the register stack other than the X register contains a non-zero value

14.4.6 Unload Standard

Selecting this item will unload all the currently loaded standard composites. This includes any that have been modified using the Edit Current menu item.

14.4.7 Reload Standard

Selecting this item will reload all the standard composites. Any that have been modified using the Edit Current menu item will be reset to their original definitions.

14.4.8 Load Composites...

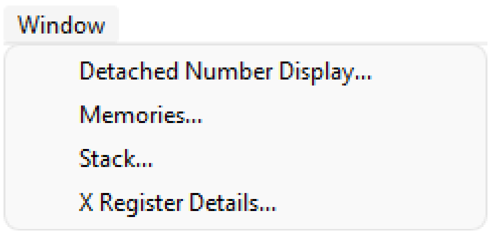
Selecting this item will allow the selection of a Composites definition file from which all the composites contained within it will be loaded.

14.4.9 Save All...

Selecting this item will allow the specification of a file which will be created and will contain the definitions of all the currently loaded composites.

This menu item will be disabled if there are no composites currently loaded.

14.5 Window Menu



This is displayed by clicking on “Window” in the keypad menu, or by using the Alt+W keyboard shortcut.

14.5.1 Detached Number Display...

Selecting this item will detach the main number display from the keypad and display it in the Detached Number Display window.

This menu item will be disabled if the detached number display window is currently displayed.

14.5.2 Memories…

Selecting this item will display the Memories Display window.

This menu item will be disabled under any of the following circumstances:

- Display mode is Integer Fields Values, Integer Fields Bit Pattern or Floating-point Components
- The memories display window is currently displayed

14.5.3 Stack…

Selecting this item will display the Stack Display window.

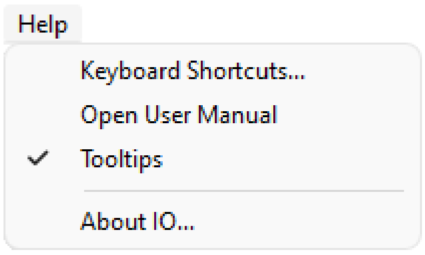
This menu item will be disabled if the stack display window is currently displayed.

14.5.4 X Register Details…

Selecting this item will display the X Register Details Display window.

This menu item will be disabled if the X register details display window is currently displayed.

14.6 Help Menu



This is displayed by clicking on "Help" in the keypad menu, or by using the Alt+H keyboard shortcut.

14.6.1 Keyboard Shortcuts…

Selecting this item will display the Keyboard Shortcuts Help window which displays all the currently available keyboard shortcuts.

This menu item will be disabled if the keyboard shortcuts help window is currently displayed.

14.6.2 Open User Manual

Selecting this item will display this user manual. You must have a default PDF Reader application installed on your computer for this to work.

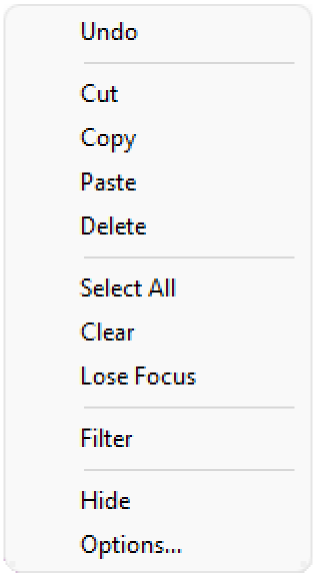
14.6.3 Tooltips

Clicking this item will toggle the setting of the "Enable Tooltips" Window option. A checkmark will be displayed if tooltips are enabled when the menu is displayed.

14.6.4 About IO…

Selecting this item will display the "About IO" window which contains information about the version of IO in use.

14.7 Expression Editor Menu



This is displayed by right-clicking on the expression editor edit control.

14.7.1 Undo

Selecting this item undoes the last edit operation. This operation may also be performed by using the Ctrl+Z keyboard shortcut. This menu item is not available if there no edits which can be undone.

14.7.2 Cut

Selecting this item cuts selected text to the clipboard. This operation may also be performed by using the Ctrl+X keyboard shortcut. This menu item is not available if no text is selected.

14.7.3 Copy

Selecting this item copies selected text to the clipboard. This operation may also be performed by using the Ctrl+C keyboard shortcut. This menu item is not available if no text is selected.

14.7.4 Paste

Selecting this item pastes text from the clipboard. This operation may also be performed by using the Ctrl+V keyboard shortcut. This menu item is not available if no text is available on the clipboard.

14.7.5 Delete

Selecting this item deletes selected text. This operation may also be performed by using the Del keyboard shortcut. This menu item is not available if no text is selected.

14.7.6 Select All

Selecting this item selects all text. This operation may also be performed by using the Ctrl+A keyboard shortcut.

14.7.7 Clear

Selecting this item deletes all text. This operation may also be performed by using the Ctrl+A and Del keyboard shortcuts.

14.7.8 Lose Focus

Selecting this item removes the keyboard focus from the edit control. This will hide the text cursor and allow the keyboard cursor movement keys to be used to scroll the main display.

14.7.9 Filter

Selecting this item deletes all invalid characters. This menu item is not available if the “Auto Filter” Calculator Operation option is selected.

14.7.10 Hide

Selecting this item removes the expression editor. This operation may also be performed by using the Ctrl+E keyboard shortcut. This menu item is not available if the “Organization” Window option is set to “Expression Editor Only”.

14.7.11 Options …

Selecting this item displays the Expression Editor Options popup window.

---

15 MAIN DISPLAY

The main display is located near the top of the keypad window or in the detached display window, depending on the setting of the “Main Number Display” Window option, and consists of two parts, the main value/bit pattern display and the markers display. These are located above/below each other depending in the setting of the “Markers Position” Number Display option.

15.1 Main Value/Bit Pattern Display

The content of the main number display is dictated by the current contents of the X register and the currently selected Display Mode.

The maximum number of digits that can be displayed is dependent on several things, as follows:

Display Type	Small Digits	Medium Digits	Large Digits
Keypad Portrait, Compact, Expression Editor Only organizations	32	17	
Keypad Landscape organization, Integer/Fixed-point	51	27	
Keypad Landscape organization, Basic only floating-point functions	38	20	
Keypad Landscape organization, and Constants/Log/Exp floating-point functions	45	23	
Keypad Landscape organization, and Trigonometric floating-point functions	51	27	
Keypad Landscape organization, and Hyperbolic floating-point functions	57	30	
Detached Single line, using display width key	96	96	96
Detached Single line, dragging window left/right edge	Limited by desktop size		
Detached Multiple line, using display width key and dragging top/bottom edge	96 x 79	96 x 39	96 x 19
Detached Multiple line, dragging window left/right edge or corners	Limited by desktop size		

The specified values are for completely displayed digits, there may be part of an additional digit visible.

15.1.1 Word Value Display Modes

A string of digit characters (and possibly other characters) is generated from the number currently stored in the X register in accordance with the current settings of the Number Display Options. If the “Exponent Display Format Type” Number Display option is set to “Segmented” then a string will be generated for each segment to be displayed. The string(s) are then displayed in the main value display in accordance with the current settings of the Digit Set and Color options.

15.1.1.1 Invalid Bit Pattern Status Displays

If the X register contains a bit pattern which cannot be mapped to a number by the current coding, then a textual status string is displayed instead of a numerical value.

The following status strings can be displayed:

Status String	Codings	Meaning
BadBCD	Integer BCD	Invalid BCD Digit Bit Pattern
+Inf	Floating-point	Positive Overflow
-Inf	Floating-point	Negative Overflow
QNaN	Floating-point	Quiet Not-a-number
SNaN	Floating-point	Signaling Not-a-number
Unnormal	Floating-point 80-bit	Pseudo Normal
PDenormal	Floating-point 80-bit	Pseudo Denormal
+PInf	Floating-point 80-bit	Pseudo Positive Overflow
-PInf	Floating-point 80-bit	Pseudo Negative Overflow
PQNaN	Floating-point 80-bit	Pseudo Quiet Not-a-number
PSNaN	Floating-point 80-bit	Pseudo Signaling Not-a-number

15.1.1.2 Exponent Adjustment

The keyboard up and down arrow keys can be used to perform the following operations in certain circumstances.

Increase Exponent

This operation is performed by pressing the keyboard/numeric keypad Up Arrow key when the Display Mode is Fixed-point Word Value or Floating-point Word Value.

<u>For Binary/Decimal General Display Format</u>

If an exponent value is not displayed, the display is switched to an exponent format with an exponent value of one and the mantissa adjusted to reflect the current value.

If an exponent value of minus one is displayed, the display is switched to a non-exponent format.

<u>For Binary/Decimal Exponent Display Format</u>

The displayed exponent value is increased by one and the displayed mantissa adjusted to reflect the current value.

For Decimal Engineering Display Format

The displayed exponent value is increased by three and the displayed mantissa adjusted to reflect the current value.

No flags are affected.

Decrease Exponent

This operation is performed by pressing the keyboard/numeric keypad Down Arrow key when the Display Mode is Fixed-point Word Value or Floating-point Word Value.

For Binary/Decimal General Display Format

If an exponent value is not displayed, the display is switched to an exponent format with an exponent value of minus one and the mantissa adjusted to reflect the current value.

If an exponent value of one is displayed, the display is switched to a non-exponent format.

For Binary/Decimal Exponent Display Format

The displayed exponent value is decreased by one and the displayed mantissa adjusted to reflect the current value.

For Decimal Engineering Display Format

The displayed exponent value is decreased by three and the displayed mantissa adjusted to reflect the current value.

No flags are affected.

15.1.2 Bit Pattern Display Modes

A string of 0 and 1 digit characters is generated from the bit pattern currently stored in the X register, and this is then displayed in the main bit pattern display in accordance with the current settings of the Digit Set and Color Options.

15.1.3 Fields Display Modes

15.1.4 Components Display Mode

15.2 Markers Display

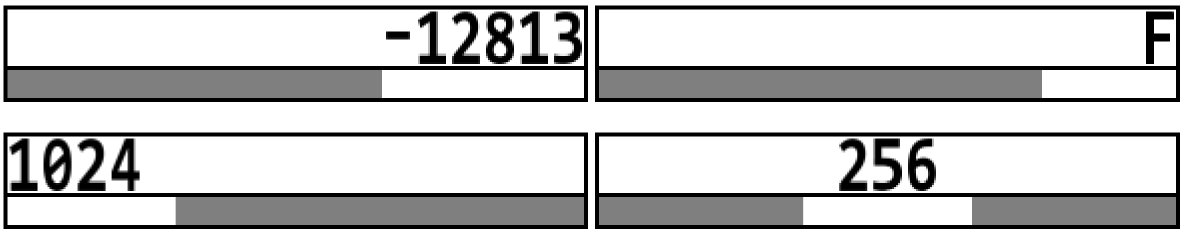
This area is located below or above the Main Value/Bit Pattern display depending on the setting of the “Markers Position” Number Display option, the default is below.

The type of content displayed in the markers area is primarily determined by the current display mode. Which bit numbers are displayed, and how, are controlled by the “Bit Numbering” and “Bit Number Display” Number Display options.

The colors of all the parts of the markers display are controlled by the Color options.

All the following examples are as for the default configuration.

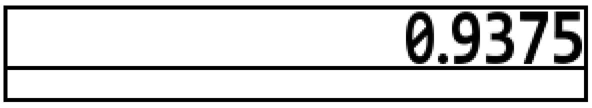
15.2.1 Integer Word Value Display Mode Markers



The markers area is divided into a used area and one or two unused areas. The unused area(s) identifies the area(s) of the main value display which cannot contain any digits for the current configuration, and the used area identifies the area of the main value display which can.

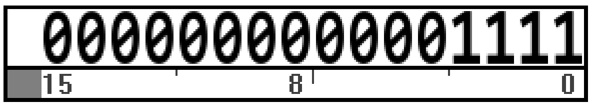
The relative positions of these areas and the number of unused areas will depend on the setting of the “Digit Justification” Number Display option.

15.2.2 Fixed/Floating-Point Word Value Display Mode Markers



The markers area will be empty.

15.2.3 Integer Word Bit Pattern Display Mode Markers



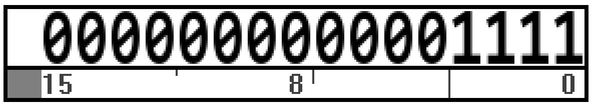
The markers area is divided into unused and used areas as for Integer Word Value display mode.

The used area will contain notches and bit numbers which identify the locations of the bits displayed in the main bit pattern display within the word.

A half-height notch denotes byte boundaries, quarter-height notches denote nibble boundaries.

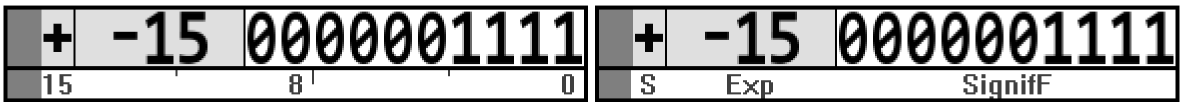
Bit numbers are not displayed if the “Digit Size” Digit Set option is set to “Small”.

15.2.4 Fixed-Point Word Bit Pattern Display Mode Markers



The markers are the same as for the Integer Word Bit Pattern display mode but will contain an additional full-height notch which denotes the position of the binary point if it lies within the word.

15.2.5 Integer Fields Values/Floating-Point Fields Display Modes Markers



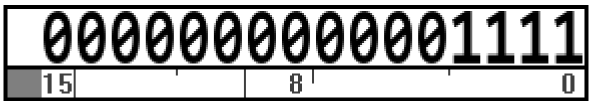
If the “Field Labels” Number Display option is not selected then the markers are the same as for the Integer Word Bit Pattern display mode.

If the “Field Labels” Number Display option is selected then the used area will contain the field labels for the displayed fields.

For the Integer Fields Values display mode bit numbers and labels are not displayed if the “Digit Size” Digit Set option is set to “Small”.

For the Floating-point Fields display mode bit numbers are not displayed if the “Digit Size” Digit Set option is set to “Small”, but labels are.

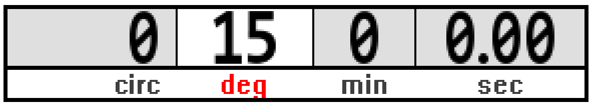
15.2.6 Integer/Floating-Point Fields Bit Pattern Display Modes Markers



The markers are the same as for the Integer Word Bit Pattern display mode but will contain additional full-height notches which denote the field boundaries.

Bit numbers are not displayed if the “Digit Size” Digit Set option is set to “Small”.

15.2.7 Floating-Point Components Display Mode Markers



The markers area will contain the component labels for the displayed components.

Labels are not displayed if the “Digit Size” Digit Set option is set to “Small”.

15.3 Secondary Number Display

The type of secondary number display to be displayed, if any, is determined by the current settings of the “Secondary Number Display” Window option as follows:

Initially set the required display type to that specified by the “Secondary Number Display Type” Window option

If the current display mode is of a bit pattern type and the “Bit Pattern Auto Alternative X” Window option is selected then set the required display type to Alternative X

Otherwise, if the current display mode is of a fields type and the “Fields Auto Alternative X” Window option is selected then set the required display type to Alternative X

Otherwise, if the current display mode is Floating-point Components and the “Components Auto Alternative X” Window option is selected then set the required display type to Alternative X

If the required display type is not set to “None” then the Secondary Number Display will be displayed below the Main Value/Bit Pattern Display, either in the Keypad window or the Detached Display widow.

The content of the Secondary Number Display is determined by the required display type as follows:

15.3.1 Alternative X

The following table shows how the current content of the X register will be displayed in the Secondary Number Display for various combinations of display mode, number base and floating-point class.

Display Mode	Base = 10	Base = 2	Base = other	NaN/Inf/Pseudo
Integer Word Value	Bit Pattern	Decimal Value	Decimal Value	-
Integer Word Bit Pattern	Value	Value	Value	-
Integer Fields Values	Full Bit Pattern Value	Full Bit Pattern Value	Full Bit Pattern Value	-
Integer Fields Bit Pattern	Full Bit Pattern Value	Full Bit Pattern Value	Full Bit Pattern Value	-
Fixed-point Word Value	Binary Value	Decimal Value	-	-
Fixed-point Word Bit Pattern	Value	Value	-	-
Floating-point Word Value	Binary Value	Decimal Value	-	Bit Pattern
Floating-point Fields	Value	Value	-	Bit Pattern
Floating-point Fields Bit Pattern	Value	Value	-	Bit Pattern
Floating-point Components	Composite Value	Composite Value	-	-

Notes:

Strictly speaking, for Integer Fields Values and Integer Fields Bit Pattern display modes what is displayed is not what is currently contained in the X register, which will be the contents of the currently selected field, but is what would be contained in the X register if a switch is made back to Integer Word Value display mode.

Unless otherwise specified, values will be displayed using the number base currently configured in the Number Display options for the current number coding type.

15.3.2 Y Value

The current content of the Y register is displayed using the current Number Display options.

The display will be blank in Floating-point Components display mode.

15.3.3 Current Memory Contents

The current content of the currently selected memory register is displayed using the current Number Display options. The status display immediately to the right of the Secondary Number display will also indicate which memory register is currently selected.

The display will be blank in any field or components display mode.

16 STATUS DISPLAYS

16.1 Basic Status Displays

There are five basic status displays, three of which are always visible.

16.1.1 Flags Status Display

This is always visible. It indicates the current settings of the calculator flags. Each flag has an identifying letter; this letter is displayed here if the flag is set to ① and not displayed if it is set to ⓪.

The identifying letters are:

C – Carry Flag
D – Denormal Flag
E – Exact Flag
N – Negative Flag
O – Value Overflow Flag
V – Overflow Flag
Z – Zero Flag

16.1.2 Number Coding Status Display

This is always visible. It displays a summary of the current Number Coding options, formatted as follows:

16.1.2.1 Integer Number Coding Type

Integer Word Value And Bit Pattern Display Modes

<Word Width>bit <gap> <Coding Form>

<Word Width> is the currently configured word width

<Coding Form> is one of:

Unsigned for integer unsigned coding

2s-Compl for integer 2s-complement coding

Excess for integer excess coding

BCD nbpd for binary coded decimal coding, n is the number of bits per digit

Integer Fields Values Display Mode

<Field Width> <gap> <Coding Form>

<Field Width> is Nbit where N is the width of the currently selected field

<Coding Form> is one of:

Unsigned for integer unsigned coding

2s-Compl for integer 2s-complement coding

Excess for integer excess coding

16.1.2.2 Fixed-Point Number Coding Type

Fix <Coding Form> [<Word Width> : <Precision>] { <Rounding Type> }

<Coding Form> is one of

Uns for fixed-point unsigned coding

2sC for fixed-point 2s-complement coding

<Word Width> is N where N is the currently configured word width

<Precision> is N where N is the currently configured precision

<Rounding Type> is one of:

T for To Zero (Truncate)

E for To Nearest (Even)

A for To Nearest (Away From Zero)

D for To Negative Infinity (Down)

U for To Positive Infinity (Up)

The rounding type will not be shown if the word width and precision values collectively require more than four characters to be displayed.

16.1.2.3 Floating-Point Number Coding Type

<Word Width>bit <gap> FloPnt <gap> <Rounding Type>

<Word Width> is the currently configured word width

<Rounding Type> is one of:

T for To Zero (Truncate)

E for To Nearest (Even)

A for To Nearest (Away From Zero)

D for To Negative Infinity (Down)

U for To Positive Infinity (Up)

16.1.3 Number Display Status Display

This is always visible. It displays a summary of the current Number Display options.

16.1.3.1 Integer Number Coding Type

Integer Word Value Display Mode

Base <gap> <Base Value>

<Base Value> is the value currently configured integer number base, expressed in decimal

Integer Bit Pattern Display Modes

BitPatt

No additional information is displayed

Integer Fields Values Display Mode (Non-Bit Pattern Current Field)

Base <gap> <Base Value>

<Base Value> is the value of the number base for the currently selected field, expressed in decimal

Integer Fields Values Display Mode (Bit Pattern Current Field)

BitPatt

No additional information is displayed

16.1.3.2 Fixed/Floating-Point Number Coding Types

Fixed/Floating-Point Word Value Display Modes

{ <Base> } <Display Format> { <Displayed Digits> }

<Base> is Bin for binary display formats; nothing is displayed for decimal display formats

<Display Format> is one of:

Gen for general display formats

Exp for exponential display formats

Eng for decimal engineering display format

<Displayed Digits> is one of:

All for all digits

Ndd for the indicated number of displayed decimal digits

Ndp for the indicated number of displayed decimal places

Nsd for the indicated number of automatically chosen significant decimal digits

NsdE for the indicated number of automatically chosen significant decimal digits for inexact values and all digits for exact values

Displayed Digits are only shown for decimal display formats.

Fixed/Floating-Point Bit Pattern Display Modes

BitPatt

No additional information is displayed

Floating-Point Fields Display Mode

Flds <Sign Format> <Exponent Format> <Significand Format>

<Sign Format> is one of:

S: the sign field is displayed as a + or -

B: the sign field is displayed as a 1 or 0

<Exponent Format> is one of:

D: the exponent value is displayed in decimal

B: the exponent value is displayed as unsigned binary with leading zeros

H: the exponent value is displayed as unsigned hexadecimal with leading zeros

Note that exponent values are only displayed for Normal floating-point values. For other classes the bit pattern contained in the exponent field will be displayed, this will be either all 0s or all 1s.

<Significand Format> is one of:

B: the significand is displayed in binary

HM: the significand is displayed in hexadecimal with the most significant 4 bits in the first digit

HL: the significand is displayed in hexadecimal with the least significant 4 bits in the first digit

Floating-Point Components Display Mode

Base Component Selected

<N>dp { <Rounded> }

<N> is the number of displayed decimal places

<Rounded> is R if the least significant displayed decimal place is rounded, nothing is displayed if not

Non-base Component Selected

<N>x<Label>

<N> is how many of the next component there are in one of the current component

<Label> is as many characters of the label for the next component that will fit in the display

16.1.4 ASCII Characters Status Display

This is only displayed for the integer number coding type, and only if the word width is a multiple of eight. This occupies the same area of the keypad as the trigonometric units status display.

One character is displayed for each byte of the bit pattern currently contained in the X register. If the Integer “Big Endian” Number Display option is currently selected then the first character displayed will be for the most significant byte of the bit pattern, if not then it will be for the least significant byte.

If a byte contains the ASCII code for a printable character, then that character will be displayed, if not then the block character ■ will be displayed.

Double-clicking on this status display will display the Character Data Entry popup window.

The following printable characters can be displayed:

MS 5 Bits	Least Significant 3 Bits							
	⓪⓪⓪	⓪⓪①	⓪①⓪	⓪①①	①⓪⓪	①⓪①	①①⓪	①①①
⓪⓪①⓪⓪		!	“	#	$	%	&	‘
⓪⓪①⓪①	(	)	*	+	,	-	.	/
⓪⓪①①⓪	0	1	2	3	4	5	6	7
⓪⓪①①①	8	9	:	;	<	=	>	?
⓪①⓪⓪⓪	@	A	B	C	D	E	F	G
⓪①⓪⓪①	H	I	J	K	L	M	N	O
⓪①⓪①⓪	P	Q	R	S	T	U	V	W
⓪①⓪①①	X	Y	Z	[	\	]	^	_
⓪①①⓪⓪	`	a	b	c	d	e	f	g
⓪①①⓪①	h	i	j	k	l	m	n	o
⓪①①①⓪	p	q	r	s	t	u	v	w
⓪①①①①	x	y	z	{	|	}	~	■

Notes:

⓪⓪①⓪⓪⓪⓪⓪ is the space character
⓪⓪①⓪⓪①①① is the single quote character
⓪①①⓪⓪⓪⓪⓪ is the grave accent character
⓪①①①①①①① is not a printable character

16.1.5 Trigonometric Units Status Display

This is only displayed for the “Number Coding Type” Number Coding option is set to “Floating-point”, and only if the trigonometric function keys are displayed. This occupies the same area of the keypad as the ASCII characters status display.

Displays the current setting of the “Trigonometric Units” Calculator Operation option.

One of the following will be displayed:

“Radians”, “Degrees”, “Gradians”, “Right Angles” or “Circles”

16.2 Secondary Number Status Display

The Secondary Number Status Display is only visible if the “Secondary Number Display Type” Window option is set to anything other than “None”. The setting of this option can be cycled by using the Ctrl+N keyboard shortcut.

When shown, it will be one of the following to indicate what is being displayed in the Secondary Number Display:

“X” for the contents of the X register in an alternative format

“Y” for the contents of the Y register

“Mn” for the contents of the currently selected memory register; n identifies which memory register is currently selected

16.3 Extended Status Displays

The Extended Status Displays are only visible if the “Status Display” Window option is set to “Extended”. The setting of this option can be toggled by using the Ctrl+S keyboard shortcut. There are thee extended status displays, two of which are displayed when the “Status Display” Window option is set to “Extended”.

16.3.1 Integer/Fixed-Point Overflow Handling Status Display

This is only displayed for integer and fixed-point number coding types and indicates the setting of the corresponding Calculator Operation option.

One of the following will be displayed:

“Trun” if the truncated bit pattern of the result is used

“Sat” if the saturated value of the result is used

16.3.2 Exponent Limits Status Display

This is only displayed if the number coding type is floating-point, and the decimal or binary general display format is selected and indicates the Exponent Limits set in the Number Display options for the current number base.

The format is:

<Neg Limit> : <Pos Limit>

The two numerical values for the limits for the current number base are displayed.

16.3.3 Number Coding Option Change Handling Status Display

This is displayed for all number coding types and indicates the settings of the Number Coding Option Change Handling options in the Calculator Operation options for changes from the current number coding type to each of the three types.

The format is:

<Pres> ->Int <gap> <Pres> ->Fix <gap> <Pres> ->Flo

Pres is one of:

> "BP" indicates that the bit pattern will be preserved and changes from incompatible register contents will be prevented
>
> "TBP" indicates that the bit pattern will be preserved and incompatible bit patterns will be truncated
>
> "Val" indicates that the value will be preserved and changes from incompatible register contents will be prevented
>
> "RVal" indicates that the value will be preserved and incompatible values will be rounded

16.3.4 Dropbox

The dropbox is not a status display but as it is located in the extended area of the Status Display region it is described here.

To use the dropbox select one or more files in a Windows Explorer window, then drag and drop them onto the Dropbox arrow. Dropped files of the following types will be loaded by IO.

<u>Configuration Files (.cfg)</u>

The configuration data in the file is loaded as described for the Load Config… item in the File menu.

<u>Field Layouts Files (.iof)</u>

The field layout definitions in the file are loaded as described for the Load Layouts… item in the Field Layout menu.

<u>Composite Files (.ioc)</u>

The composite definitions in the file are loaded as described for the Load Composites… item in the Composite menu.

Other types of file are ignored.

The order in which the files are loaded is not defined, so you should not drop more than one configuration file at a time as you cannot be sure which one will be loaded last and hence define the configuration.

17 EXPRESSION EDITOR

17.1 Introduction

The expression editor consists of an edit control and an associated “execute expression” button. These will be located immediately below the main display if it is embedded in the keypad, or at the top of the keypad if not. The button is located on the right-hand side and the edit control occupies the rest of the width of the keypad. Whether or not the expression editor is visible is determined by two window options.

If the Keypad Window Organization is set to “Expression Editor Only” then the keypad window will contain the expression editor, the status displays and the main number display if the “Embedded” option is selected. No keys will be visible below the expression editor.

17.2 Usage

The expression editor can be used to enter and edit a mathematical expression in textual form which can then be evaluated when required. Only operations with printable keyboard shortcuts can be included in the expression, so which are available is dependent on the current configuration and display mode.

The expression is evaluated by either clicking on the ▶ button, or by pressing the keyboard “enter” key while the edit control is active. Evaluating an expression performs the same processing as that which would occur if each character contained in the edit control was entered at the keyboard.

Note that the expression editor cannot be used to enter fixed/floating-point values in binary exponent format.

For example:

If the edit control contains

(12-8)*(24-9)=

If the current coding is integer 32-bit unsigned and the number base is set to decimal then executing the expression with an Algebraic Logic calculator will generate a result of 60. Note that the final “)” is optional and is included for clarity.

With a Reverse Polish Notation calculator, the following contents of the edit control will perform the same calculation

12=8-=24=9-*

If the number base is then set to hexadecimal, re-executing the expression will generate a result of 10E, a different result because the "12" and "24" will represent different values to be used in the calculations.

The expression in the edit control can be edited using the keyboard and clipboard in the same way as in a typical text editor. Note that the clipboard options do not affect clipboard operations used in the expression editor.

There are five Calculator Operation options which affect the operation of the expression editor. The expression editor edit control has a popup menu, displayed by right-clicking on the edit control, which provides access to editing and other operations on the edit control contents. Details of these options and menu entries are contained in the chapters on the Calculator Operation options and menus.

17.2.1 Expression Filtering

As not all characters that may be entered into the edit control are valid keyboard shortcuts, any invalid characters will be filtered, i.e., removed, prior to the expression being executed. Sometimes you may wish to paste some text into the edit control which you then want to edit to leave the expression to be evaluated. Sometimes you may prefer to have all invalid characters filtered out as soon as they are entered and sometimes you may prefer to manually remove them yourself as you edit the expression; there is an option that lets you choose which behavior you want.

17.2.2 Memory Operations

By default, only non-alphabetic keyboard shortcuts are valid, but optionally you can enable the processing of memory operations, which have alphabetic keyboard shortcuts, during expression evaluation if required.

For example:

If you wish to evaluate X^2/(X+1) for several values of X, and you are using the Algebraic Logic calculator type with a floating-point coding.

Select the "Allow Memory Ops" Calculator Operation option, then enter

```
s"/(m+1)=
```

into the edit control.

You can then enter the required value into the X register, and execute the expression, for each required value of X.

Executing the expression will perform the following operations:

s	Stores the current value of X in the current memory
"	Squares the value of X
/	Pushes X (the square of the original value of X) onto the register stack Pushes the division operator onto the operator stack
(	Pushes an open-bracket onto the operator stack
m	Recalls the original value of X from the current memory
+	Pushes the original value of X onto the register stack Pushes the addition operator onto the operator stack
1	Sets X register to a value of 1
)	Performs addition of 1 to original value of X
=	Completes calculation by processing the stacked division operator, leaving the final result in the X register

18 POPUP WINDOWS

The popups provide a quick method of modifying many options without having to display an option set window. A popup is also available to permit the entry of a value made up of a sequence of ASCII character codes.

They are displayed by double clicking on a related status display. Which popup is displayed is context sensitive; it is determined by a combination of which status display is double-clicked, the current configuration and the current display mode.

The controls displayed are a subset of those contained in the window for the related option set.

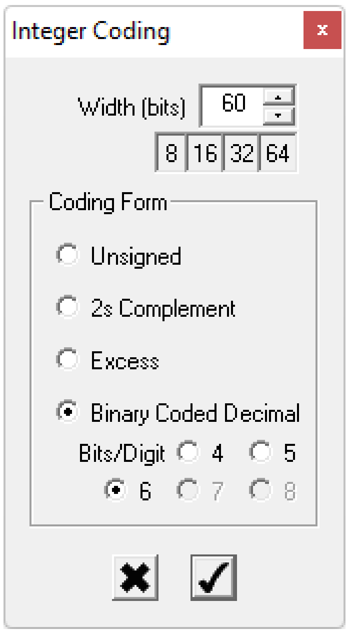
All popups contain the following two buttons:

✖ Clicking on this button is equivalent to clicking the “Cancel” button in the options set window. The same processing will be performed if the “esc” keyboard key is pressed, or the window’s close button is clicked.

✓ Clicking on this button is equivalent to clicking the “OK” button in the options set window. The same processing will be performed if the “enter” keyboard/keypad key is pressed.

Popups do not have their own Help windows; help for the controls contained in a popup is contained in the Help window for the window for the related option set.

18.1 Integer Coding


Integer Coding

Width (bits) 60

8 | 16 | 32 | 64

Coding Form

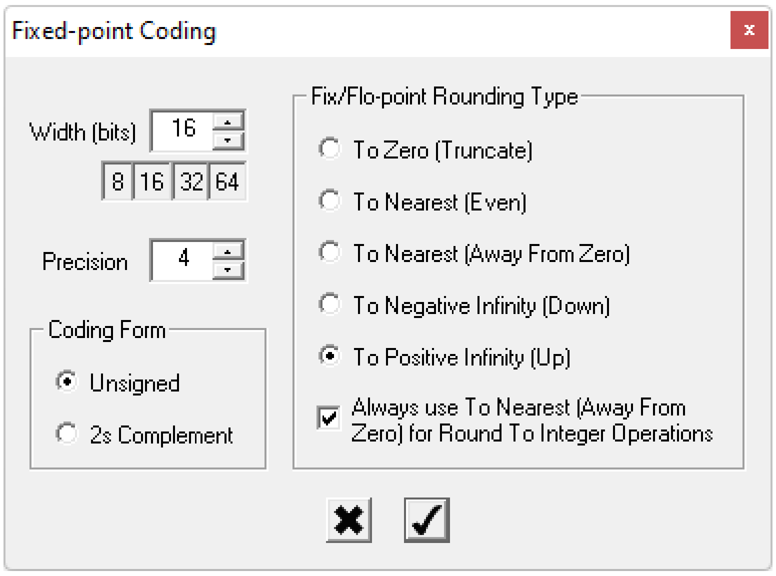
Unsigned [ ]
2s Complement [ ]
Excess [ ]
[✓ Binary Coded Decimal]

Bits/Digit: 4 5 [✓ 6] 7 8

This popup is displayed by double-clicking on the number coding status display when the number coding type is integer and the display mode type is not fields.

See the section on the number coding option set window for a description of how to use these controls and possible error messages.

18.2 Fixed-Point Coding


Fixed-point Coding

Width (bits) 16

8 | 16 | 32 | 64

Precision 4

Coding Form
- [✓ Unsigned]
- 2s Complement

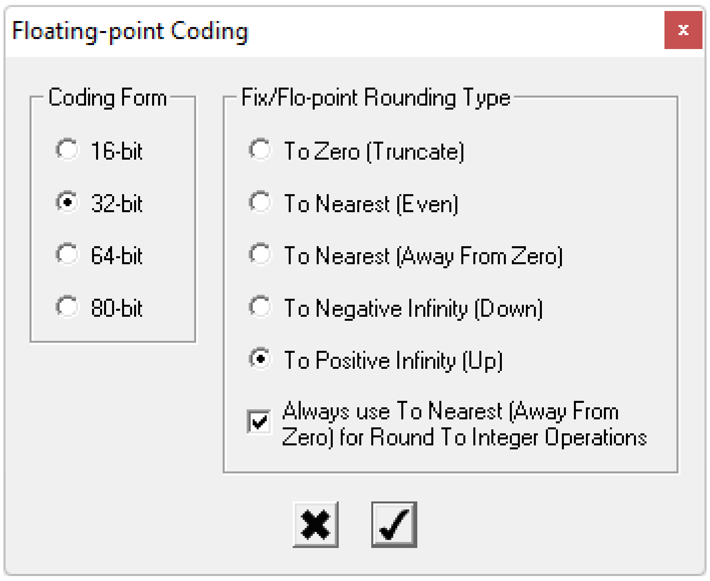
Fix/Flo-point Rounding Type
- To Zero (Truncate)
- To Nearest (Even)
- To Nearest (Away From Zero)
- To Negative Infinity (Down)
- [✓ To Positive Infinity (Up)]

[✓ Always use To Nearest (Away From Zero) for Round To Integer Operations]

This popup is displayed by double-clicking on the number coding status display when the number coding type is fixed-point.

See the section on the number coding option set window for a description of how to use these controls and possible error messages.

18.3 Floating-Point Coding


Floating-point Coding

Coding Form
- 16-bit
- [✓ 32-bit]
- 64-bit
- 80-bit

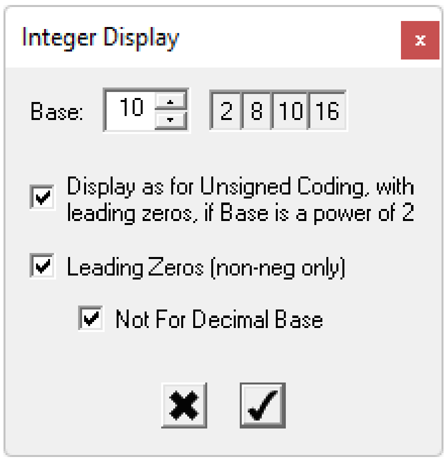
Fix/Flo-point Rounding Type
- To Zero (Truncate)
- To Nearest (Even)
- To Nearest (Away From Zero)
- To Negative Infinity (Down)
- [✓ To Positive Infinity (Up)]

[✓ Always use To Nearest (Away From Zero) for Round To Integer Operations]

This popup is displayed by double-clicking on the number coding status display when the number coding type is floating-point and the display mode type is not Fields.

See the section on the number coding option set window for a description of how to use these controls.

18.4 Integer Display


Integer Display

Base: 10 | 2 | 8 | 10 | 16

[✓] Display as for Unsigned Coding, with leading zeros, if Base is a power of 2

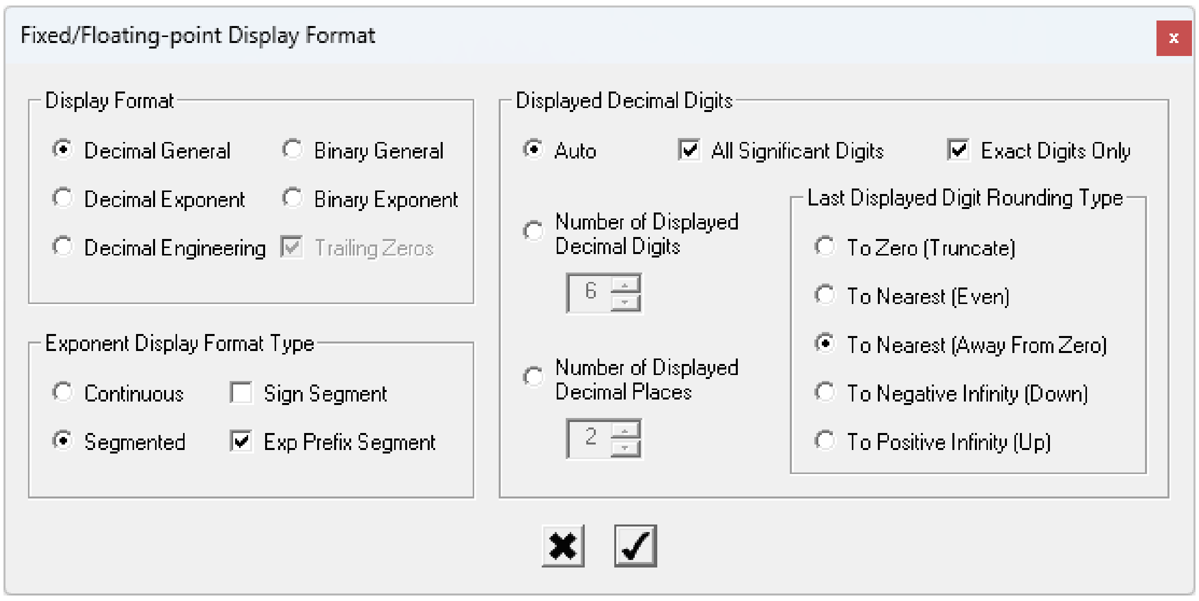
[✓] Leading Zeros (non-neg only)

[✓] Not For Decimal Base

This popup is displayed by double-clicking on the number display status display when the number coding type is integer and the display mode type is Word Value.

See the section on the number display option set window for a description of how to use these controls and possible error messages.

18.5 Fixed/Floating-Point Display Format


Fixed/Floating-point Display Format

Display Format
- [✓ Decimal General]
- Binary General
- Decimal Exponent
- Binary Exponent
- Decimal Engineering
- [✓] Trailing Zeros

Exponent Display Format Type
- Continuous
- [ ] Sign Segment
- [✓ Segmented]
- [✓] Exp Prefix Segment

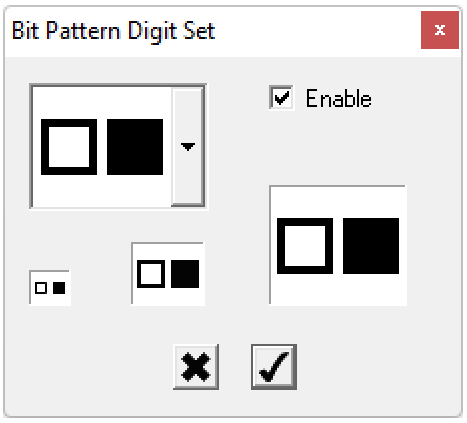
Displayed Decimal Digits
- [✓ Auto]
- [✓] All Significant Digits
- [✓] Exact Digits Only
- Number of Displayed Decimal Digits: 6
- Number of Displayed Decimal Places: 2

Last Displayed Digit Rounding Type
- To Zero (Truncate)
- To Nearest (Even)
- [✓ To Nearest (Away From Zero)]
- To Negative Infinity (Down)
- To Positive Infinity (Up)

This popup is displayed by double-clicking on the number display status display when the number coding type is fixed-point or floating-point and the display mode type is Word Value.

See the section on the number display option set window for a description of how to use these controls and possible error messages.

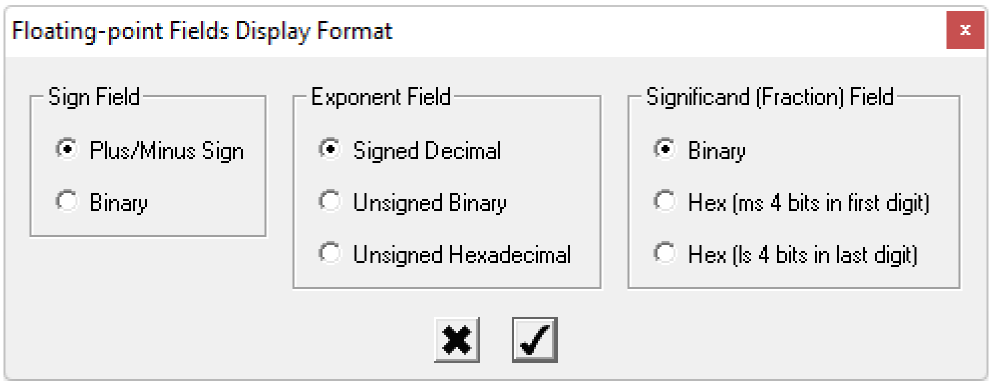
18.6 Bit Pattern Digit Set



This popup is displayed by double-clicking on the number display status display in a Bit Pattern display mode type.

See the section on the digit set option set window for a description of how to use these controls.

18.7 Floating-Point Fields Display Format



Sign Field: (●) Plus/Minus Sign, ( ) Binary

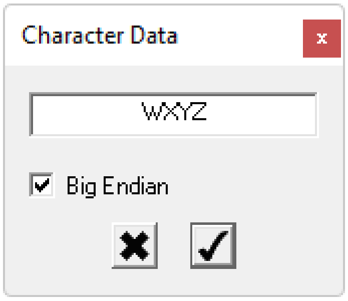
Exponent Field: (●) Signed Decimal, ( ) Unsigned Binary, ( ) Unsigned Hexadecimal

Significand (Fraction) Field: (●) Binary, ( ) Hex (ms 4 bits in first digit), ( ) Hex (ls 4 bits in last digit)

This popup is displayed by double-clicking on the number display status display in the floating-point fields display mode.

See the section on the number display option set window for a description of how to use these controls.

18.8 Character Data Entry



WXYZ

☑ Big Endian

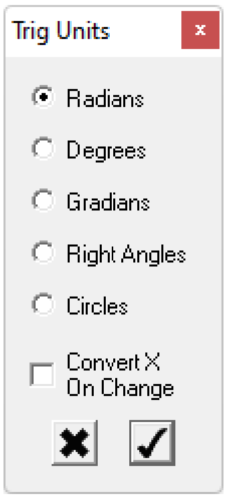
This popup is displayed by double-clicking on the ASCII characters status display.

See the section on the number display option set window for a description of how to use the “Big Endian” control.

If every byte of the current contents of the X register contains the ASCII code of a printable character, then these characters will be displayed in the edit control ready for editing. The order in which the characters are displayed will depend on the setting of the “Big Endian” option when this popup is displayed.

If the tick (ok) button is used then the ASCII codes of the characters in the edit control will be written to the bytes of the X register. Which bytes the character codes are written to will be determined by the setting of the "Big Endian" option which may be modified by using this popup.

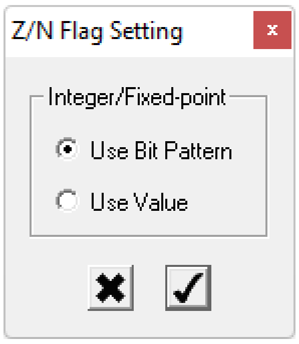
18.9 Trigonometric Function Units



This popup is displayed by double-clicking on the trigonometric units status display.

See the section on the calculator operation option set window for a description of how to use these controls.

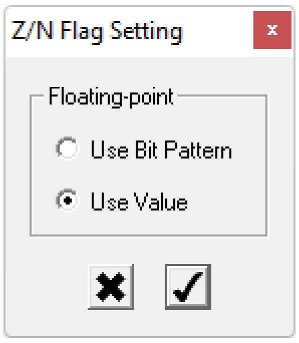
18.10 Integer/Fixed-Point N/Z Flag Setting Control



This popup is displayed by double-clicking on the flags status display when the number coding type is integer or fixed-point.

See the section on the calculator operation option set window for a description of how to use these controls.

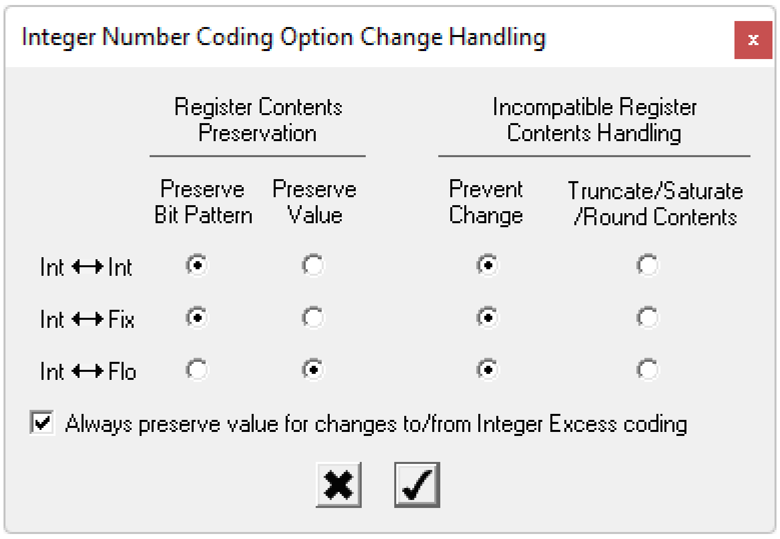
18.11 Floating-Point N/Z Flag Setting Control



This popup is displayed by double-clicking on the flags status display when the number coding type is floating-point.

See the section on the calculator operation option set window for a description of how to use these controls.

18.12 Integer Number Coding Option Change Handling



Integer Number Coding Option Change Handling

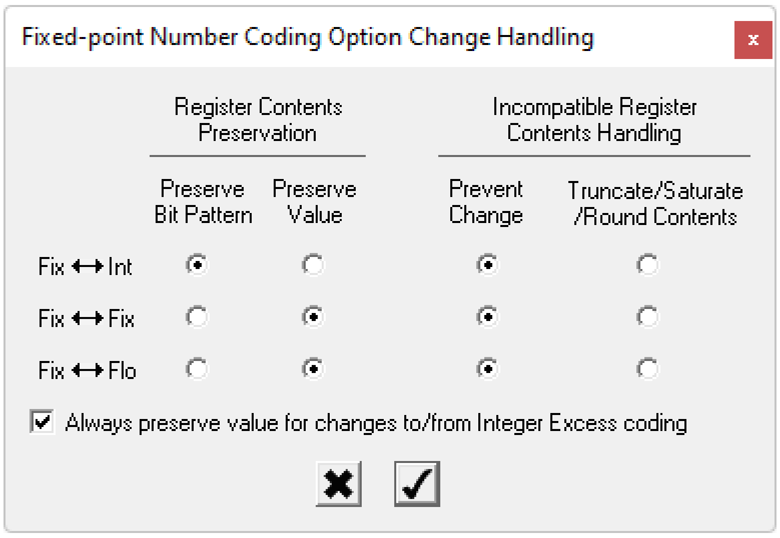
	Register Contents Preservation		Incompatible Register Contents Handling	
	Preserve Bit Pattern	Preserve Value	Prevent Change	Truncate/Saturate /Round Contents
Int ↔ Int	●	○	●	○
Int ↔ Fix	●	○	●	○
Int ↔ Flo	○	●	●	○

☑ Always preserve value for changes to/from Integer Excess coding

This popup is displayed by double-clicking on the number coding option change status display when the number coding type is integer.

See the section on the calculator operation option set window for a description of how to use these controls.

18.13 Fixed-Point Number Coding Option Change Handling



Fixed-point Number Coding Option Change Handling

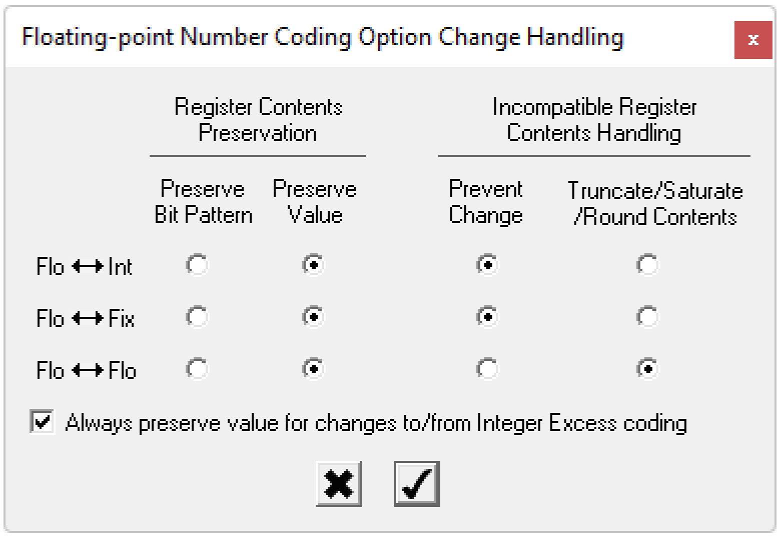
	Register Contents Preservation		Incompatible Register Contents Handling	
	Preserve Bit Pattern	Preserve Value	Prevent Change	Truncate/Saturate /Round Contents
Fix ↔ Int	●	○	●	○
Fix ↔ Fix	○	●	●	○
Fix ↔ Flo	○	●	●	○

☑ Always preserve value for changes to/from Integer Excess coding

This popup is displayed by double-clicking on the number coding option change status display when the number coding type is fixed-point.

See the section on the calculator operation option set window for a description of how to use these controls.

18.14 Floating-Point Number Coding Option Change Handling



Floating-point Number Coding Option Change Handling

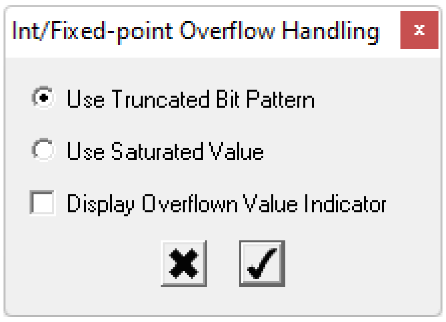
	Register Contents Preservation		Incompatible Register Contents Handling	
	Preserve Bit Pattern	Preserve Value	Prevent Change	Truncate/Saturate /Round Contents
Flo ↔ Int	○	●	●	○
Flo ↔ Fix	○	●	●	○
Flo ↔ Flo	○	●	○	●

[✓] Always preserve value for changes to/from Integer Excess coding

This popup is displayed by double-clicking on the number coding option change status display when the number coding type is floating-point.

See the section on the calculator operation option set window for a description of how to use these controls.

18.15 Integer/Fixed-Point Overflow Handling



Int/Fixed-point Overflow Handling

● Use Truncated Bit Pattern

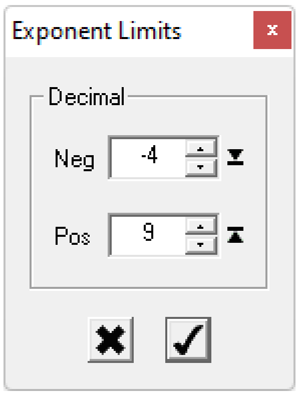
○ Use Saturated Value

[ ] Display Overflown Value Indicator

This popup is displayed by double-clicking on the integer/fixed-point overflow handling status display.

See the section on the calculator operation option set window for a description of how to use these controls.

18.16 Decimal General Display Format Exponent Limits



Exponent Limits

Decimal

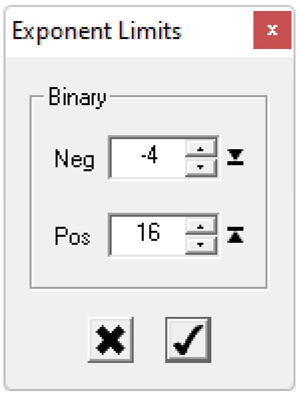
Neg -4

Pos 9

This popup is displayed by double-clicking on the exponent limits status display when the display format is decimal general.

See the section on the number display option set window for a description of how to use these controls and possible error messages.

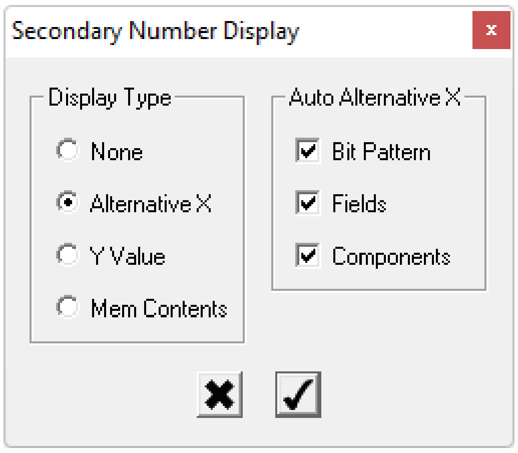
18.17 Binary General Display Format Exponent Limits



This popup is displayed by double-clicking on the exponent limits status display when the display format is binary general.

See the section on the number display option set window for a description of how to use these controls and possible error messages.

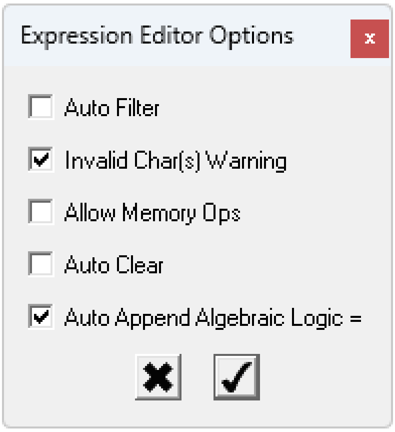
18.18 Secondary Number Display Type



This popup is displayed by double-clicking on the secondary number status display. This status display will only be displayed when the secondary number display type is not “None”, it will be displayed to the right of the secondary number display, which may be in the detached display window.

See the section on the window option set window for a description of how to use these controls.

18.19 Expression Editor Options

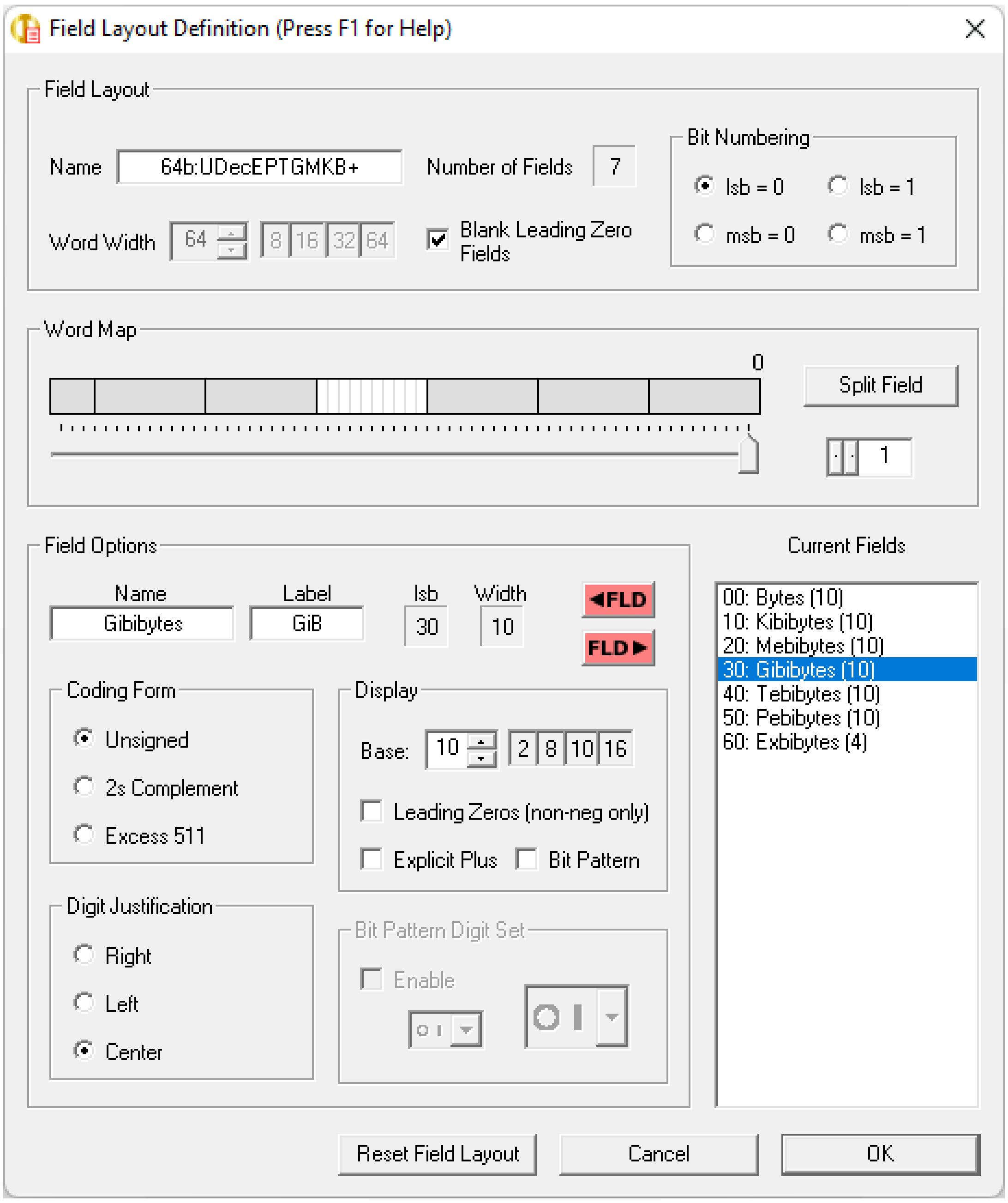


This popup is displayed by selecting the “Options…” item from the menu displayed when right-clicking on the Expression Editor edit control.

See the section on the calculator operation option set window for a description of how to use these controls.

19 FIELD LAYOUTS

19.1 Definition Window


Field Layout Definition (Press F1 for Help)

Field Layout

Name: 64b:UDecEPTGMKB+

Number of Fields: 7

Word Width: 64 (8 | 16 | 32 | 64)

[✓] Blank Leading Zero Fields

Bit Numbering: [● lsb = 0] ○ lsb = 1 ○ msb = 0 ○ msb = 1

Word Map: 0 — Split Field — 1

Field Options

Name	Label	lsb	Width
Gibibytes	GiB	30	10

◄FLD FLD►

Coding Form: [● Unsigned] ○ 2s Complement ○ Excess 511

Display: Base: 10 (2 | 8 | 10 | 16)

☐ Leading Zeros (non-neg only)

☐ Explicit Plus ☐ Bit Pattern

Bit Pattern Digit Set: ☐ Enable

Digit Justification: ○ Right ○ Left [● Center]

Current Fields

00: Bytes (10)
10: Kibibytes (10)
20: Mebibytes (10)
30: Gibibytes (10)
40: Tebibytes (10)
50: Pebibytes (10)
60: Exbibytes (4)

Reset Field Layout | Cancel | OK

The Field Layout Definition window is displayed when any of the Field Layout->New Layout… , Field Layout->Edit Current… or Field Layout->Edit Copy… menu items are selected. For a “New Layout” it will initially be populated to define a default field layout, the controls can then be used to modify the definition as required. For an “Edit Current” or “Edit Copy” it will initially be populated to reflect the definition of the currently selected field layout, the controls can then be used to modify the definition as required. For an “Edit Current” the field layout name cannot be edited. For an “Edit Copy” a modified name will initially be displayed and this may be edited as desired.

19.1.1 Edit Buttons

Either stores the current definition as a new field layout or uses it as the new definition for the currently selected field layout, depending on which menu option was used to open the window. The Window will then be closed. Note that the newly defined field layout may then be used, but it will not be saved for use by a subsequent IO session until the Field Layout->Save All… menu item is selected. If any of the control settings are invalid then a message identifying the error will be displayed and no field layout definition will be added or modified.

Discards the current definition and close the window.

Sets all the controls in the window to what they were when the window was opened.

19.1.2 Field Layout

Name

Defines the name of the field layout. A field layout name must be entered and may have up to 19 characters. The name cannot be modified when editing the currently selected field layout.

Word Width

Defines the number of bits contained in the word used to contain all the fields in the field layout. The value entered must be in the range 8 to 64. The word width cannot be modified when editing the currently selected field layout.

Clicking the numbered controls allow you to quickly select a word width of 8, 16, 32 or 64 bits.

No. Fields

Displays the number of fields currently defined for the field layout.

Blank Leading Zero Fields

Setting this option causes all the most significant fields which contain zero to be displayed as blank rather than displaying zeros.

Bit Numbering

Controls how bits are numbered in the markers for this field layout, they will always be in numerical sequence from one end of the bit pattern to the other. This setting will be used when this field layout is selected in Fields Display Mode irrespective of the setting of the “bit numbering” Number Display option.

This option can take one of four settings:

• lsb = 0

The least significant bit is numbered 0.

• msb = 0

The most significant bit is numbered 0.

• lsb = 1

The least significant bit is numbered 1.

• msb = 1

The most significant bit is numbered 1.

19.1.3 Word Map

The Word Map displays a diagrammatical representation of the currently defined fields within the word, together with the location of bit boundaries within the word.

The slider beneath the Word Map indicates the location of the current bit boundary. It may be dragged with the mouse to select a new current bit boundary. The Word Map and slider will be re-drawn to reflect changes in the word width, field definitions or current bit boundary. A field may be selected by clicking on it in the Word Map.

---

msb

The number displayed above the word map on the left-hand side is the bit number of the most significant bit of the word if bit numbering starts from that end, otherwise it is not displayed

---

lsb

The number displayed above the word map on the right-hand side is the bit number of the least significant bit of the word if bit numbering starts from that end, otherwise it is not displayed

---

Bit Boundary

Displays the bit number of the bit immediately to the left of the current bit boundary. It can be modified by clicking on the left/right arrows or entering a new bit number using the keyboard as well as by dragging the slider

---

Split Field

If the current bit boundary is not on a field boundary, then clicking this button will split the field containing the bit boundary into two fields at that position, otherwise the button will not be displayed.

Merge Fields

If the current bit boundary is on a field boundary, then clicking this button will merge the fields either side of the bit boundary into a single field, otherwise the button will not be displayed.

19.1.4 Field Options

These controls can be used to modify the options for the currently selected field.

Name

Defines the name of the selected field. A field name may have up to 19 characters.

Label

Defines the selected field's label. A field label may have up to 7 characters.

---

lsb

Displays the bit number of the least significant bit of the selected field. This may not be used to change it, it may only be changed using the Word Map.

Width

Displays the width of the selected field. This may not be used to change it; it may only be changed using the Word Map.

◄FLD ⇐▯▯

Causes the field to the left of the currently selected field to become the newly selected field. If the currently selected field is the leftmost field, then the rightmost field will become the newly selected field.

FLD► ▯▯⇒

Causes the field to the right of the currently selected field to become the newly selected field. If the currently selected field is the rightmost field, then the leftmost field will become the newly selected field.

Coding Form

These controls select the form of integer coding to be used for the selected field.

This option can take one of three settings:

• Unsigned

Selects unsigned integer coding. If a width of 1 is specified then unsigned coding is automatically selected.

• 2’s Complement

Selects 2's complement signed integer coding.

• Excess

Selects excess signed integer coding. If the excess value for the selected width is not too large it is displayed here.

Digit Justification

This option determines how the displayed digits for the field are justified within it.

This option can take one of three settings:

• Center

The digits will be center justified within the field.

• Right

The digits will be right justified within the field.

• Left

The digits will be left justified within the field.

Base

Defines the number base used to display the field contents. The number entered must be in the range two to sixteen.

Clicking the numbered controls allow you to quickly select a base of two (binary), eight (octal), ten (decimal) or sixteen (hexadecimal).

Explicit Plus

Setting this option will display a positive field value with a "+" prefix.

Leading Zeros (non-neg only)

Setting this option will display leading zeros for the field value. Leading zeros will not be displayed for negative values.

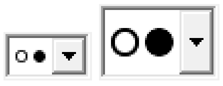
Bit Pattern

Setting this option will display the bit pattern stored in the field instead of its value. The Bit Pattern Digit Set controls for the field will be enabled.

Bit Pattern Digit Set

Enable

Setting this option will enable the use of the selected bit pattern digit sets for the field; otherwise, the 0 and 1 digits of the number digit sets selected in the digit sets options will be used. The associated combo boxes will be enabled.



These combo boxes may be used to select one of the bit pattern digit sets from the dropdown list displayed by clicking on one of the combo boxes. Any selection made will apply to all digit set sizes. The combo boxes will display the selected bit pattern digit sets as black digits on a white background.

19.1.5 Current Fields

This lists the fields currently defined for the field layout. The first field listed includes the least significant bit of the word; it is followed by the others in order of rising significance. A field may be selected by clicking on it in this list as well as by using the Word Map or the previous/next field buttons.

19.1.6 Error Messages

The following error messages may be displayed after the "OK" button is used to indicate that the control settings would result in an invalid field layout definition. No field layout definition will be added or modified.

Field Layout Name not entered

A name has not been entered for the field layout; this is required.

Field Name not entered

A name has not been entered for at least one of the fields; all fields must be given a name.

Field Layout with specified name currently loaded

A field layout with the specified name is currently loaded; no two loaded field layouts may have the same name,

19.2 Standard Field Layouts

IO starts up with twenty-one standard field layouts loaded.

The definitions of the standard field layouts may be modified using the "Edit Current…" item of the Field Layout menu, but these changes will not be saved so the next time IO starts up they will have their initial definitions. Modified versions that you wish to use again can be created using the "Edit Copy…" and "Save All…" items of the Field Layout menu.

The standard field layouts may be unloaded by using the “Unload Standard” item of the Field Layout menu. Any other loaded field layouts will remain loaded.

The standard field layouts may be reloaded after they have been unloaded or modified by using the “Reload Standard” item of the Field Layout menu.

The standard field layouts are:

19.2.1 Unsigned Decimal Byte Fields

16b:2xUDecByte

16-bit word, two 8-bit fields, each field displayed as unsigned decimal.

32b:4xUDecByte

32-bit word, four 8-bit fields, each field displayed as unsigned decimal.

64b:8xUDecByte

64-bit word, eight 8-bit fields, each field displayed as unsigned decimal.

96b:12xUDecByte

96-bit word, twelve 8-bit fields, each field displayed as unsigned decimal.

19.2.2 Unsigned Decimal 16-bit Fields

32b:2xUDec16b

32-bit word, two 16-bit fields, each field displayed as unsigned decimal.

64b:4xUDec16b

64-bit word, four 16-bit fields, each field displayed as unsigned decimal.

96b:6xUDec16b

96-bit word, six 16-bit fields, each field displayed as unsigned decimal.

19.2.3 Unsigned Decimal 32-bit Fields

64b:2xUDec32b

64-bit word, two 32-bit fields, each field displayed as unsigned decimal.

96b:3xUDec32b

96-bit word, three 32-bit fields, each field displayed as unsigned decimal.

19.2.4 Unsigned Decimal 10-bit Fields

16b:UDecKB

16-bit word.

One 10-bit field at least significant end of word, displayed as unsigned decimal.

One 6-bit field at most significant end of word, displayed as unsigned decimal.

32b:UDecGMKB

32-bit word.

Three 10-bit fields at least significant end of word, each displayed as unsigned decimal.

One 2-bit field at most significant end of word, displayed as unsigned decimal.

64b:UDecEPTGMKB

64-bit word.

Six 10-bit fields at least significant end of word, each displayed as unsigned decimal.

One 4-bit field at most significant end of word, displayed as unsigned decimal.

96b:UDecYZEPTGMKB

96-bit word.

Nine 10-bit fields at least significant end of word, each displayed as unsigned decimal.

One 6-bit field at most significant end of word, displayed as unsigned decimal.

19.2.5 Floating-Point Format Fields

16b:FloPnt(Bin)

16-bit word.

One 10-bit field at least significant end of word, displayed in binary with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 5-bit field located between other fields, displayed as excess 15 decimal.

16b:FloPnt(Hex)

16-bit word.

One 10-bit field at least significant end of word, displayed in hexadecimal with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 5-bit field located between other fields, displayed as excess 15 decimal.

32b:FloPnt(Bin)

32-bit word.

One 23-bit field at least significant end of word, displayed in binary with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 8-bit field located between other fields, displayed as excess 127 decimal.

32b:FloPnt(Hex)

32-bit word.

One 23-bit field at least significant end of word, displayed in hexadecimal with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 8-bit field located between other fields, displayed as excess 127 decimal.

64b:FloPnt(Bin)

64-bit word.

One 52-bit field at least significant end of word, displayed in binary with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 11-bit field located between other fields, displayed as excess 1023 decimal.

64b:FloPnt(Hex)

64-bit word.

One 52-bit field at least significant end of word, displayed in hexadecimal with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 11-bit field located between other fields, displayed as excess 1023 decimal.

80b:FloPnt(Bin)

80-bit word.

One 64-bit field at least significant end of word, displayed in binary with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 15-bit field located between other fields, displayed as excess 16383 decimal.

80b:FloPnt(Hex)

80-bit word.

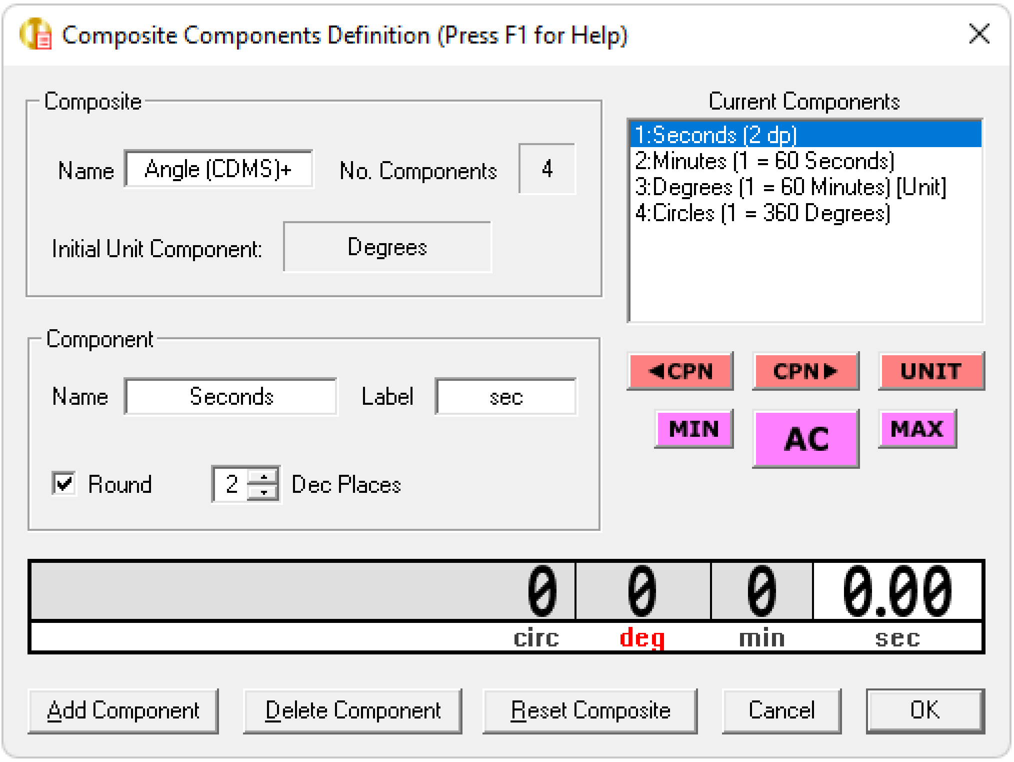
One 64-bit field at least significant end of word, displayed in hexadecimal with leading zeros.

One 1-bit field at most significant end of word, displayed using “+-” bit pattern digit set.

One 15-bit field located between other fields, displayed as excess 16383 decimal.

20 COMPOSITE VALUES

20.1 Definition Window


Composite Components Definition (Press F1 for Help)

Composite

Name: Angle (CDMS)+ No. Components: 4

Initial Unit Component: Degrees

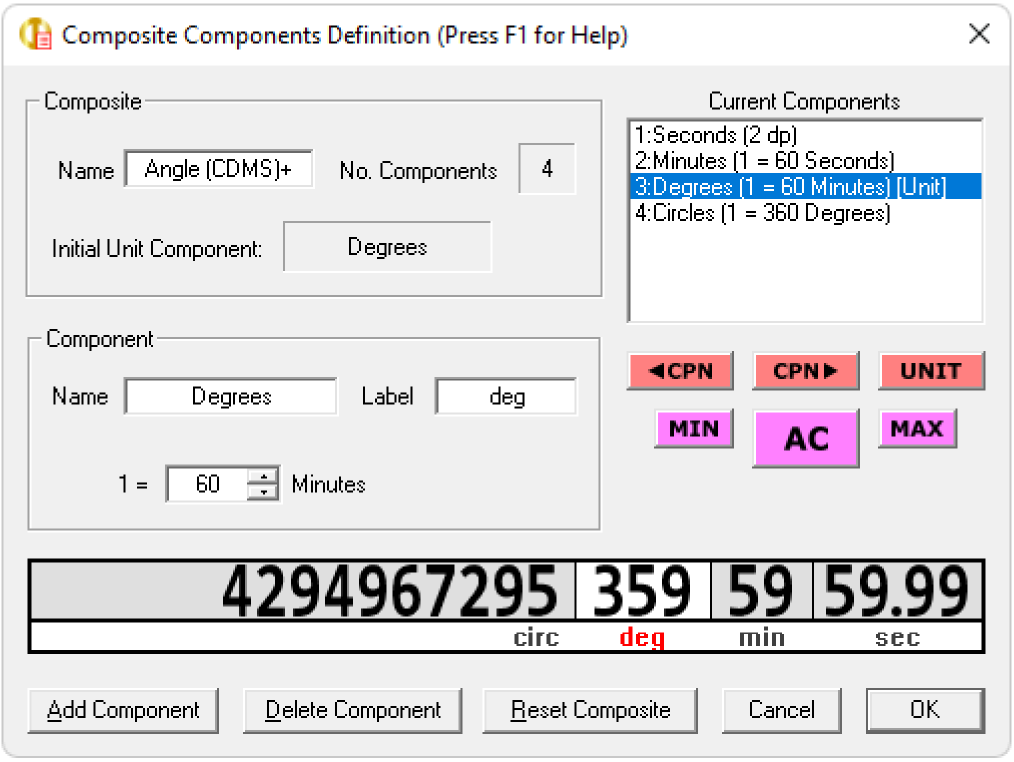
Current Components
1:Seconds (2 dp)
2:Minutes (1 = 60 Seconds)
3:Degrees (1 = 60 Minutes) [Unit]
4:Circles (1 = 360 Degrees)

Component

Name: Seconds Label: sec

[✓] Round 2 Dec Places

```
0      0     0     0.00
circ   deg   min   sec
```


Composite Components Definition (Press F1 for Help)

Composite

Name: Angle (CDMS)+ No. Components: 4

Initial Unit Component: Degrees

Current Components
1:Seconds (2 dp)
2:Minutes (1 = 60 Seconds)
3:Degrees (1 = 60 Minutes) [Unit]
4:Circles (1 = 360 Degrees)

Component

Name: Degrees Label: deg

1 = 60 Minutes

```
4294967295   359   59    59.99
circ         deg   min   sec
```

The Composite Components Definition window is displayed when any of the Composite ->New Composite… , Composite ->Edit Current… or Composite->Edit Copy… menu items are selected. For a "New Composite" it will initially be populated to define a default composite, the controls can then be used to modify the definition as required. For an "Edit Current" or "Edit Copy" it will initially be populated to reflect the definition of the currently selected composite, the controls can then be used to modify the definition as required. For an "Edit Current" the composite name cannot be edited. For an "Edit Copy" a modified name will initially be displayed and this may be edited as desired.

20.1.1 Composite

Name

Defines the name of the composite. A composite name must be entered and may have up to nineteen characters. The name cannot be modified when editing the currently selected composite.

No. Components

Displays the number of components currently defined for the composite. A composite must have at least two components but may not have more than six.

Initial Unit Component

Displays the name of the component currently selected as the Initial Unit Component for the composite. This component will be selected as the current and unit component the first time this composite is selected after IO starts up.

20.1.2 Component

These controls define the properties of the currently selected component.

Name

Defines the name of the component. A component name must be entered and may have up to nineteen characters.

Label

Defines the component label. A component label is optional and may have up to seven characters.

Round

Setting this property causes the displayed value to be rounded using the fixed/floating-point rounding type selected in the number coding options. It is only displayed for the base component.

Dec Places

Sets the number of decimal places displayed for the base component. It is only displayed for the base component. This value may not be more than six.

---

1 = [] (Previous Component)

Defines how many of the previous component there are in one of the current component. It is not displayed for the base component. This value must be at least two and may not be more than four thousand.

20.1.3 Current Components

This lists the components currently defined for the composite. The first component listed is the base component, which is followed by the others in order of rising significance. A component may be selected by clicking on it in this list.

20.1.4 Control Buttons

◄CPN ⇤▯▯

The component to the left of the currently selected component will become the newly selected component. If the currently selected component is the leftmost component, then the base component will become the newly selected component.

CPN► ▯▯⇥

The component to the right of the currently selected component will become the newly selected component. If the currently selected component is the base component, then the leftmost component will become the newly selected component.

UNIT =1

The currently selected component will be the unit component when the composite is initially selected

---

MIN ⊻

The component values for the largest possible negative composite value will be displayed in the Composite Display

---

AC C CA 0

Zero values will be displayed in the Composite Display for all components

---

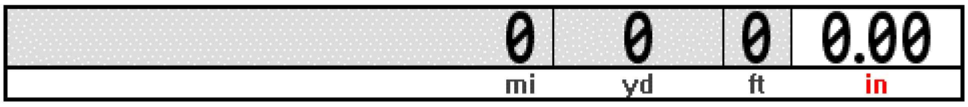
MAX

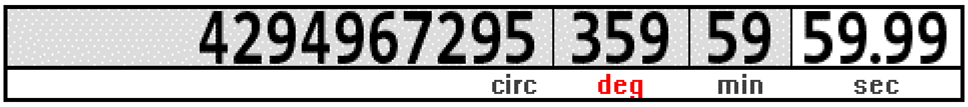
The component values for the largest possible positive composite value will be displayed in the Composite Display

---

The key labels and colors used for these buttons will be determined by the current configuration

20.1.5 Composite Display





This display reflects the current settings for the composite, and shows how it would appear in the main number display area using the current configuration. The component values displayed will depend on which of the MIN, AC and MAX buttons in the window was the last clicked.

A component may be selected by clicking on it or on the corresponding area of the markers section in this display

20.1.6 Edit Buttons

Add Component

Adds a new component to the composite. Only a new most significant component may be added. The new component will initially have the name "New", no label, and contain two of the previous most significant component. These may then be changed to the required settings. This button will be disabled if there are currently six components defined as this is the maximum number permitted.

Delete Component

Removes the most significant component from the composite. This button will be disabled if there are currently two components defined as this is the minimum number permitted.

Reset Composite

Sets all the controls in the window as they were when the window was opened.

Cancel

Discard the current definition and close the window.

Either stores the current definition as a new composite or uses it as the new definition for the currently selected composite, depending on which menu option was used to open the window. The Window will then be closed. Note that the newly defined composite may then be used, but it will not be saved for use by a subsequent IO session until the Composite ->Save All… menu item is selected. If any of the control settings are invalid then a message identifying the error will be displayed and no composite definition will be added or modified.

20.1.7 Error Messages

Current composite definition is invalid

This is displayed beneath the control buttons if the composite defined by the current control settings is invalid. In this situation the Composite Display will not be visible. The reason for the error must be remedied before the “OK” button will work, if not then clicking on the “OK” button will display a message describing the reason for the error.

The following error messages may be displayed after the “OK” button is used to indicate that the control settings would result in an invalid composite definition. No composite definition will be added or modified.

Composite Name not entered

A name has not been entered for the composite; this is required.

Component Name not entered

A name has not been entered for at least one of the components; all components must be given a name.

Composite with specified name currently loaded

A composite with the specified name is currently loaded; no two loaded composites may have the same name,

Invalid composite name

A composite name has not been entered or has more than the permitted maximum of 19 characters. It is not normally possible to enter such a name.

Invalid number of decimal places

The specified number of decimal places for the base component is not in the valid range of 0 to 6. It is not normally possible to enter such a value.

Invalid number of components

The number of components is outside of the valid range of 2 to 6. It is not normally possible to define such a number of components.

Invalid unit component

The selected unit component is invalid for the defined components. It is not normally possible to make such a selection.

Invalid component name

A component name has more than the permitted maximum of 19 characters. It is not normally possible to enter such a name.

Invalid component label

A component label has more than the permitted maximum of 7 characters. It is not normally possible to enter such a label.

Invalid multiple value

A component has been defined as being a multiple of the preceding component which is not in the valid range of 2 to 4000.

Combination of multiples and decimal places too large

The specified combination of multiple values and number of decimal places for the base component are too large to be accommodated.

20.2 Standard Composites

IO starts up with six standard composite definitions loaded.

The definitions of the standard composites may be modified using the “Edit Current…” item of the Composite menu, but these changes will not be saved so the next time IO starts up they will have their initial definitions. Modified versions that you wish to use again can be created using the “Edit Copy…” and “Save All…” items of the Composite menu.

The standard composites may be unloaded by using the “Unload Standard” item of the Composite menu. Any other loaded composites will remain loaded.

The standard composites may be reloaded after they have been unloaded or modified by using the “Reload Standard” item of the Composite menu.

The standard composites are:

20.2.1 Angle (CDMS)

The components are:

Seconds:	Displayed with 2 decimal places
Minutes:	There are 60 seconds in a minute
Degrees:	There are 60 minutes in a degree
Circles:	There are 360 degrees in a circle

The initial unit component is Degrees.

20.2.2 Length (MYFI)

The components are:

Inches:	Displayed with 2 decimal places
Feet:	There are 12 inches in a foot

Yards:	There are 3 feet in a yard
Miles:	There are 1760 yards in a mile

The initial unit component is Inches.

20.2.3 Mass (THSPO)

The components are:

Ounces:	Displayed with 2 decimal places
Pounds:	There are 16 ounces in a pound
Stones:	There are 14 pounds in a stone
Hundredweights:	There are 8 stones in a hundredweight
Tons:	There are 20 hundredweights in a ton

The initial unit component is Pounds.

20.2.4 Mass US (THPO)

The components are:

Ounces:	Displayed with 2 decimal places
Pounds:	There are 16 ounces in a pound
Hundredweights:	There are 100 pounds in a US hundredweight
Tons:	There are 20 US hundredweights in a US ton

The initial unit component is Pounds.

20.2.5 Time (WDHMS)

The components are:

Seconds:	Displayed with 2 decimal places
Minutes:	There are 60 seconds in a minute
Hours:	There are 60 minutes in an hour
Days:	There are 24 hours in a day
Weeks:	There are 7 days in a week

The initial unit component is Seconds.

20.2.6 Volume (GPF)

The components are:

Fluid Ounces:	Displayed with 2 decimal places
Pints:	There are 20 fluid ounces in a pint
Gallons:	There are 8 pints in a gallon

The initial unit component is Gallons.

21 KEYBOARD SHORTCUTS

Which of IO's keyboard shortcuts are available at any particular time is determined by which of IO's windows is currently activated. A window can usually be activated by clicking on it. The currently activated window is usually identified by it using different text and/or background colors in its title bar.

If a keyboard shortcut for a disabled key is used then it will be ignored.

In the following tables:

kp denotes a numeric keypad key when Num Lock is off
nkp denotes a numeric keypad key when Num Lock is on
Ctrl+ denotes a key pressed while the Ctrl key is held down
Shift+ denotes a key pressed while the Shift key is held down

21.1 Keypad & Detached Display

The following keyboard shortcuts are available when either the keypad window or the detached display window is the active window, unless the keyboard focus is in the expression editor edit control or the character data entry popup window edit control.

The printable characters identified here are recognized by the expression editor; any other characters contained in an entered expression will be filtered out at the configured time.

21.1.1 Always Available

Keyboard Key(s)	Operation	IO Key(s)
0 1 2 3 4 5 6 7 8 9 nkp0 nkp1 nkp2 nkp3 nkp4 nkp5 nkp6 nkp7 nkp8 nkp9	Digit Entry for base less than or equal to ten	0 1 2 3 4 5 6 7 8 9
#	Change Sign	+/− CHS
Esc	All Clear	AC C CA ‹0›
Backspace	Delete Digit	Del ▶ ◀
Del kp.	Clear Entry	CE 0▸x
+ kp+ nkp+	Add	+
- kp- nkp-	Subtract	−
* kp* nkp*	Multiply	× ✱

Keyboard Key(s)	**Operation**	**IO Key(s)**
/ kp/ nkp/	Divide	÷ /
Spacebar	Use Previous Number Coding	PCod ▤
z Z	Memory Clear	Mclr 0▸M MC
s S	Memory Store	Msto X▸M MS STO M in
m M	Memory Recall	Mrcl M▸X MR RCL
x X	Memory Swap	M⇄X Msw Mex
p P	Memory Add	M+ Madd
Ctrl+O Shift+Ctrl+O	Cycle though keypad organizations	
Ctrl+N Shift+Ctrl+N	Cycle/hide secondary number display	
Ctrl+S	Show/hide extended status displays	
Ctrl+E	Show/hide expression editor (no visible effect in Expression Editor Only organization)	
Home kp7	Scroll main display to left-hand end	
End kp1	Scroll main display to right-hand end	
Page-up kp9	Scroll main display towards left-hand end by width of display	
Page-down kp3	Scroll main display towards right-hand end by width of display	
Left-arrow kp6	Scroll main display towards right-hand end by one digit	
Right-arrow kp4	Scroll main display towards left-hand end by one digit	

21.1.2 Available For Embedded Number Display

Keyboard Key(s)	**Operation**	**IO Key(s)**
Ctrl+D	Detach Display	↗ ↗

Keyboard Key(s)	Operation	IO Key(s)
Ctrl+C	Copy to Clipboard	⧉ ⧉
Ctrl+V	Paste from Clipboard	📋 📋

21.1.3 Available For Detached Number Display

Keyboard Key(s)	Operation	IO Key(s)
Ctrl+C	Copy to Clipboard	⧉ ⧉
Ctrl+V	Paste from Clipboard	📋 📋
Ctrl+D	Close Detached Display	

21.1.4 Available For Integer Number Coding Type (Base Greater Than 10)

Keyboard Key(s)	Operation	IO Key(s)
a b c d e f A B C D E F	Digit Entry	A B C D E F a b c d e f
%	Modulo	mod %

21.1.5 Available For Integer Number Coding Type (Base Not Greater Than 10)

Keyboard Key(s)	Operation	IO Key(s)
“	Square	x^2 Sqr
%	Modulo	mod %
!	Factorial	X! Fact

21.1.6 Available For Integer/Fixed-Point Number Coding Types

Keyboard Key(s)	Operation	IO Key(s)
;	Increment	INCR +1
:	Decrement	DECR −1
~	Bitwise Complement	NOT CPL ~ ▷o ¬
&	Bitwise AND	AND & ⊐D
|	Bitwise Inclusive OR	OR I ⊐D
^	Bitwise Exclusive OR	XOR EOR ^ ⊐D
$	Bitwise Bit Clear	BIC &~ ⊐D
?	Random Bit Pattern	RAND ???
<	Shift Left	SHL ⇤ << SHL1 ← << 1
>	Arithmetic Shift Right	ASHR ⇥ >> ASR1 → >> 1
\	Logical Shift Right	LSHR ⇥ >>> LSR1 → >>> 1

21.1.7 Available For Floating-Point Number Coding Type

Keyboard Key(s)	Operation	IO Key(s)
?	Random Value	RAND 0.???
Ctrl+F Shift+Ctrl+F	Cycle through floating-point operation key displays	

21.1.8 <u>Available For Fixed/Floating-Point Number Coding Types</u>

Keyboard Key(s)	Operation	IO Key(s)
. nkp.	Decimal/Binary Point	•
“	Square	x^2 Sqr
‘	Raise to power	y^x ** ^ ↑
Up-arrow kp8	Increase displayed exponent value	
Down-arrow kp2	Decrease displayed exponent value	

21.1.9 <u>Available For Fixed/Floating-Point Number Coding Types (Decimal)</u>

Keyboard Key(s)	Operation	IO Key(s)
e E	Enter Decimal Exponent	EE $x10^n$ E e

21.1.10 <u>Available For Fixed/Floating-Point Number Coding Types (Binary)</u>

Keyboard Key(s)	Operation	IO Key(s)
b B	Enter Binary Exponent	EE $x2^n$ B b

21.1.11 <u>Available For Algebraic Logic Calculator Type</u>

Keyboard Key(s)	Operation	IO Key(s)
([	Open Bracket	([
)]	Close Bracket	)]

Keyboard Key(s)	Operation	IO Key(s)
= Enter kpEnter nkpEnter	Equals	=

21.1.12 Available For Reverse Polish Notation Calculator Type

Keyboard Key(s)	Operation	IO Key(s)
= Enter kpEnter nkpEnter	RPN Enter	Enter Ent⬆

21.2 Memories Display

The following keyboard shortcuts are available when the memories display window is the active window.

Keyboard Key(s)	Operation	IO Key(s)
z Z	Memory Clear	Mclr 0▸M MC
s S	Memory Store	Msto X▸M MS STO M in
m M	Memory Recall	Mrcl M▸X MR RCL
x X	Memory Swap	M⇄X Msw Mex
p P	Memory Add	M+ Madd
Ctrl+C	Copy current memory display to Clipboard	📋 📋
Ctrl+V	Paste from Clipboard to current memory display	📋 📋
Home kp7	Scroll current memory display to left-hand end	
End kp1	Scroll current memory display to right-hand end	
Page-up kp9	Scroll current memory display towards left-hand end by width of display	

Keyboard Key(s)	Operation	IO Key(s)
Page-down kp3	Scroll current memory display towards right -hand end by width of display	
Left-arrow kp6	Scroll current memory display towards right-hand end by one digit	
Right-arrow kp4	Scroll current memory display towards left-hand end by one digit	

21.3 Stack Display

The following keyboard shortcuts are available when the stack display window is the active window.

Keyboard Key(s)	Operation
Home kp7	Scroll X register display to left-hand end
End kp1	Scroll X register display to right-hand end
Page-up kp9	Scroll X register display towards left-hand end by width of display
Page-down kp3	Scroll X register display towards right-hand end by width of display
Left-arrow kp6	Scroll X register display towards right-hand end by one digit
Right-arrow kp4	Scroll X register display towards left-hand end by one digit

21.4 X Register Details Display

The following keyboard shortcuts are available when the X Register Details display window is the active window.

Keyboard Key(s)	Operation
Home kp7	Scroll value and limits displays to left-hand end
End kp1	Scroll value and limits displays to right-hand end
Page-up kp9	Scroll value and limits displays towards left-hand end by width of display
Page-down kp3	Scroll value and limits displays towards right-hand end by width of display
Left-arrow kp6	Scroll value and limits displays towards right-hand end by one digit
Right-arrow kp4	Scroll value and limits displays towards left-hand end by one digit

21.5 Help Windows

The following keyboard shortcuts are available when a help window other than an Option Set Window Help Window is the active window.

Keyboard Key(s)	Operation
Home kp7	Scroll to top left of help content
End kp1	Scroll to bottom right of help content
Page-up kp9	Scroll up help content by height of window
Page-down kp3	Scroll down help content by height of window
Left-arrow kp6	Scroll to left of help content by a small amount
Right-arrow kp4	Scroll to right of help content by a small amount
Up-arrow kp8	Scroll up help content by small amount
Down-arrow kp2	Scroll down help content by small amount

22 INDEX