Оглавление

Список используемых сокращений

LSB — Least Significant Bit (младший значащий бит)
MSB — Most Significant Bit (старший значащий бит)
Акк. — аккумулятор
АОТ — арифметика ограниченной точности
АПТ — арифметика с плавающей точкой
Арг. — аргумент
АФТ — арифметика с фиксированной точкой
Дв. — двоичный
Дес. — десятичный
ДЛ — делитель
ДМ — делимое
КТЦ — контрольно-тренировочный цикл
Ман. — мантисса
МК — микроконтроллер
МлБ — младший байт
ММ — множимое
МН — множитель
Пор. — порядок
ПР — произведение
ПТ — плавающая точка
Разр. — разряд
Рег. — регистр
Сл. — слагаемое
СрБ — средний байт
СтБ — старший байт
ФТ — фиксированная точка
Ч — частное
ЭО — экранная область

Глава 1. АРИФМЕТИКА С ПЛАВАЮЩЕЙ ТОЧКОЙ

1.1. Общие сведения о представлении чисел

1.1.1. Свойства позиционных систем счисления

Прежде чем приступить к рассмотрению программ, реализующих арифметику с плавающей точкой (наряду с этим термином в вычислительной математике и технике используется термин-синоним «фиксированная запятая»), необходимо ознакомиться с некоторыми общими сведениями о форматах представления двоичных и десятичных, целых и дробных, знаковых и беззнаковых чисел. Также полезно вспомнить методы, алгоритмы и программы выполнения арифметических операций над множеством этих чисел, обеспечивающие заданную точность и диапазон вычислений при определенных затратах памяти и времени работы микроконтроллера.

Изображение чисел в любой *позиционной системе счисления* с натуральным основанием R ($R > 1$) базируется на представлении их в виде произведения целочисленной степени R^m основания R на *полином* от этого основания:

$$A_R = \pm R^m \sum_{i=1}^{n} a_i R^{-i}, \qquad (1.1)$$

где $a_i \in \{0, 1, ..., R-1\}$ — цифры R-ичной системы счисления (R-ичные цифры);

n — количество разрядов *(разрядность)*, используемых для представления числа;

R^m — *характеристика числа*, причём показатель $m \in \{..., -2, -1, 0, +1, +2, ...\}$;

$R^m R^{-i} = R^{m-i}$ — *позиционный вес i-го разряда* числа.

В *десятичной системе* ($R = 10$) для представления чисел используются цифры $a_i \in (0, 1...9)$, в *двоичной* ($R = 2$) — цифры $a_i \in (0, 1)$, в *шестнадцатеричной* ($R = 16$) — цифры $a_i \in (0, 1...9, A, B, C, D, E, F)$, где прописным латинским буквам A...F эквивалентны соответственно числа 10...15 в десятичной системе.

Если $m = \text{const}$, то формула (1.1) определяет представление числа A_R в форме с *фиксированной точкой*. Это означает, что позиция, в которой точка фиксируется между разрядами числа, отделяя целую часть от дробной и определяя вес соответствующих разрядов, постоянна в процессе вычислений для всего множества используемых чисел и зависит от заранее установленного значения m. Если $m \le 0$, то формула (1.1) представляет *дробные числа* (правильные дроби), если $m \ge n$ — *целые числа*, если $0 \le m \le n$ — *смешанные числа* (неправильные дроби). Обычно для чисел с фиксированной точкой значение m ограничено $0 \le m \le n$, т. е. позиция точки в числе выбирается в рамках n разрядов, отведённых для изображения цифровой части числа.

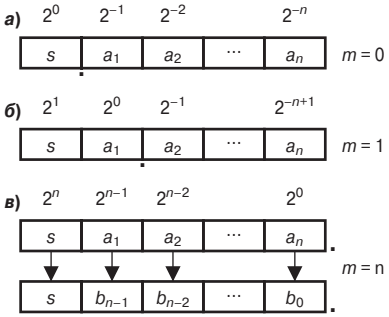

Рис. 1.1. Форматы двоичных чисел с фиксированной точкой:
а) дробные числа; *б*) смешанные числа; *в*) целые числа.

На **Рис. 1.1** приведены форматы двоичных ($R = 2$) чисел с фиксированной точкой: при $m = 0$ *(а)*, при $m = 1$ *(б)* и при $m = n$ *(в)*. Далее ограничимся рассмотрением только целых и дробных чисел, поскольку действия над смешанными числами могут быть сведены к отдельным операциям над их целыми и дробными частями. Дробные числа, в отличие от целых, будем рассматривать как числа с фиксированной точкой в узком смысле этого термина, и для их обозначения в формулах и программах будем использовать букву «ф».

В микроконтроллерах основой представления чисел является двоичная система счисления.

1.1.2. Представление целых и дробных двоичных чисел

Рассмотрим представление дробных и целых двоичных чисел.

При $R = 2$, $m = 0$ формула (1.1) имеет вид

$$A_{2\text{ф}} = \pm\sum_{i=1}^{n} a_i \cdot 2^{-i} = \pm 0.a_1 a_2 \ldots a_n, \qquad (1.2)$$

где 2^{-i} — вес i-го двоичного бита[1];

$a_i \in \{0, 1\}$ — двоичные цифры.

Формула (1.2) определяет ***дробные двоичные числа со знаком***. Левая её часть даёт *развернутую форму записи* числа в виде *полинома*, а правая — *свёрнутую форму записи* в виде последовательности цифр (коэффициентов полинома). Для изображения *знака числа S* (от лат. Signum — знак) отводится дополнительный *знаковый бит* (нулевой — с весом 2^0), а точка фиксируется перед старшим (первым — с весом 2^{-1}) цифровым битом числа (см. **Рис. 1.1*а***). Если заранее известно, что множество используемых чисел не содержит отрицательных, в формате числа бит знака можно опустить. Такие числа называют *беззнаковыми*. В R-битной сетке дробных беззнаковых чисел может быть точно представлено 2^n различных, в том числе $2^n - 1$ ненулевых, чисел, удовлетворяющих неравенству

$$A_{2\text{ф(min)}} = 2^{-n} \le A_{2\text{ф}} \le A_{2\text{ф(max)}} = 1 - 2^{-n}, \qquad (1.3)$$

где $A_{2\text{ф(min)}}$ — минимальное (ненулевое),

$A_{2\text{ф(max)}}$ — максимальное из представимых чисел.

В свёрнутой форме записи эти числа изображаются в виде $A_{2\text{ф(min)}} = 0.00...01$ и $A_{2\text{ф(max)}} = 0.11...11$.

В случае $R = 2$, $m = n$ формула (1.1) преобразуется к виду

$$A_2 = \pm\sum_{i=1}^{n} a_i \cdot 2^{n-i} = \pm\sum_{i=0}^{n-1} b_i \cdot 2^{i} = \pm b_{n-1} \ldots b_1 b_0, \qquad (1.4)$$

где $b_i = a_{n-i}$ для всех $i = 0, 1, ..., n - 1$;

$a_i, b_i \in \{0, 1\}$ — двоичные цифры.

Перекодировка индексов в формуле (1.4) даёт удобочитаемое соответствие веса бита 2^i индексу его цифры. Формула (1.4) определяет ***целые***

[1] В данной книге в тех случаях, когда речь идёт о представлении двоичных чисел, вместо термина «разряд» используется термин «бит». — *Примеч. науч. ред.*

двоичные числа со знаком. Под знак отводится дополнительный $(n + 1)$-й бит с весом 2^n, а точка фиксируется после младшего (нулевого — с весом 2^0) цифрового бита числа (**Рис. 1.1*в***). Для беззнаковых целых чисел бит знака S опускается. В n-битной сетке целых беззнаковых чисел, как и для дробных, может быть представлено множество 2^n целых, в том числе $2^n - 1$ ненулевых, чисел, удовлетворяющих неравенству

$$A_{2(\min)} = 2^0 = 1 \le A_2 \le A_{2(\max)} = 2^n - 1 . \qquad (1.5)$$

В свёрнутой форме эти числа изображаются такими же последовательностями цифр, как и дробные числа, отличаясь лишь положением точки: $A_{2ф(\min)} = 00...01$ и $A_{2ф(\max)} = 11...11$.

В формулах (1.2)...(1.5) двоичные числа представлены в виде *полинома*, коэффициенты которого выражены двоичными цифрами, а основание — десятичным числом, т. е. цифрой 2. Учитывая, что в любой позиционной системе счисления её основание однозначно представляется с помощью цифр этой системы в виде 10_R (по определению $R = (R - 1) + 1$, а такая операция даёт нуль в младшем разряде и единицу переноса в старшем разряде), формулы (1.2)...(1.5) можно обобщить на случай любой позиционной системы:

$$A_{R\text{ф}} = \pm \sum_{i=1}^{n} a_i \cdot 10_R^{-i} , \quad 10_R^{-n} \le |A_{R\text{ф}}| \le 1 - 10_R^{-n} , \qquad (1.6)$$

$$A_R = \pm \sum_{i=0}^{n=1} a_i \cdot 10_R^{i} , \quad 1 \le |A_R| \le 10_R^{n} - 1 , \qquad (1.7)$$

где обозначение модуля введено для знаковых чисел, а неравенства справедливы для всех ненулевых значений представляемых чисел.

Формула (1.6) определяет свойства *дробных*, а формула (1.7) — *целых чисел* с основанием 10_R. Эти формулы полезны, в частности, при рассмотрении чисел в десятичной и шестнадцатеричной системах счисления, широко используемых в микроконтроллерах наряду с двоичной системой.

1.1.3. Смешанные системы счисления

Для представления в микроконтроллерах систем счисления с основанием $R = 10$ и $R = 16$ используются *смешанные системы счисления*, в которых каждая цифра R-ичной системы изображается цифрами другой, Q-ичной ($Q < R$), в частности двоичной системы. Такая смешанная система на-

зывается Q—R-ичной. Для записи R-ичной цифры отводится одно и то же количество Q-ичных разрядов, минимально необходимое для представления любой R-ичной цифры.

В *двоично-десятичной системе* каждая десятичная цифра кодируется тетрадой двоичных цифр, причём между десятью (из возможных $2^4 = 16$) двоичными числами и десятичными цифрами устанавливается взаимно однозначное соответствие. Возможны различные варианты установления этого соответствия, порождающие различные *двоично-десятичные коды* с теми или иными свойствами. В программах настоящей книги используется один из наиболее распространенных кодов — код с естественными *весами* «8421», в котором десятичные цифры кодируются их естественными двоичными эквивалентами: (0 — 0000), (1 — 0001), ..., (9 — 1001). Этот код преимущественно применяется для ввода/вывода данных (при преобразованиях из двоичной системы в двоично-десятичную и наоборот), но может использоваться и непосредственно при обработке данных.

В частном случае смешанных систем, когда $R = Q^k$ (k — целое), например $R = 16$, $Q = 2$ ($k = 4$), запись числа в R-ичной системе является сокращенной записью представления этого числа в Q-ичной системе: цифра шестнадцатеричной системы заменяет значение четырёх цифр двоичной системы, что упрощает описание представлений двоичных чисел. Эта смешанная система широко используется в программах, приведённых в данной книге.

Известно, что арифметические операции над числами в любой позиционной системе счисления выполняются по тем же правилам, что и в десятичной арифметике, поскольку все они основываются на общих правилах выполнения операций над соответствующими полиномами. При этом используются те таблицы сложения (вычитания) и умножения, которые имеют место в конкретной системе счисления. В частности, для двоичной системы действуют правила:

$$\begin{array}{lll} 0+0=0, & 0-0=0, & 0\times 0=0, \\ 0+1=1, & 1-0=1, & 0\times 1=0, \\ 1+0=1, & 1-1=0, & 1\times 0=0, \\ 1+1=10, & 10-1=1, & 1\times 1=1. \end{array}$$

Эти правила не содержат операции деления двоичных цифр, поскольку её выполнение, в отличие от других операций, не может быть сведено к действиям над отдельными цифрами. Напомним, что при сложении в двоичной системе счисления двух единиц происходит *перенос* в следующий, более старший бит, а при вычитании единицы из нуля происходит *заём* в данный бит из более старшего бита, в результате чего данный бит устанавливается в единицу.

1.1.4. Представление знаковых чисел. Дополнительный и прямой код

При выполнении арифметических операций возникает проблема идентификации отрицательных чисел. Как отмечалось выше, для изображения *знака* двоичных знаковых чисел отводится специальный *знаковый бит S*. Изображение знака «+» в этом бите принято кодировать для двоичных чисел цифрой 0, а знака «–» — цифрой 1. При этом изображение числа со знаком содержит только двоичные цифры, но само число подразумевается состоящим из двух частей: знаковой и цифровой.

Если цифровая часть положительных и отрицательных чисел содержит всегда абсолютную величину числа, то такой способ представления знаковых чисел называют *прямым кодом*. Обработка чисел, представленных в прямом коде, требует отдельных операций над цифровой и знаковой частями, альтернативного выполнения операций сложения и вычитания, приводит к появлению двух *представлений нуля*: +0 и –0 (например, для дробных чисел +0 = 0.00...00; –0 = 1.00...00). Эти недостатки прямого кода не позволяют использовать его при обработке данных, и он находит применение преимущественно в операциях ввода/вывода (хотя известны примеры использования его в микроЭВМ для основного представления знаковых чисел). В микроконтроллерах для обработки знаковых чисел используются в основном *дополнительные коды* (в качестве промежуточных — также и *обратные коды)*.

Рассмотрим определения и свойства этих кодов. Пусть знаковое число с *фиксированной точкой* содержит $n + 1$ R-ичных разрядов, причём крайний левый разряд в формате числа — разряд знака, в котором «+» закодирован цифрой 0, а «–» — цифрой $R - 1$ (1 — для двоичной, 9 — для десятичной, F — для шестнадцатеричной системы счисления). Тогда дополнительный код числа определяется выражением

$$[A_R]_{\text{Д}} = \begin{cases} A_R, & \text{если } A_R \geq 0; \\ \Gamma + A_R, & \text{если } A_R < 0, \end{cases} \tag{1.8}$$

где Γ — *граница числа*, причём для дробных чисел $\Gamma = 10^1{}_R$ ($\Gamma = 2$ для $R = 2$, $\Gamma = 10$ для $R = 10$ и $\Gamma = 16$ для $R = 16$), а для целых чисел $\Gamma = 10_R^{n+1}$ ($\Gamma = 2^{n+1}$ для $R = 2$, $\Gamma = 10^{n+1}$ для $R = 10$ и $\Gamma = 16^{n+1}$ для $R = 16$). Иными словами, в дополнительном коде запись положительного числа идентична его записи в прямом коде, а запись отрицательного числа представляет собой результат операции вычитания модуля числа A_R из границы: $\Gamma - |A_R|$.

Поскольку граница Γ представляет собой фиксированную степень, легко показать, что указанная операция вычитания сводится к поразрядной операции дополнения каждой цифры числа до старшей цифры систе-

мы счисления (поиску взаимно обратной цифры $\overline{a}_i$, такой, что $a_i + \overline{a}_i = R - 1$) и суммированию полученного обратного кода с единицей младшего разряда числа, т. е. с величиной 10_R^{-n} для дробных и $10_R^n = 1$ для целых чисел. Например, для десятичной дроби 0.1234 её отрицательный эквивалент в дополнительном коде имеет вид 9.8765 + 0.0001 = 9.8766. Для проверки правильности преобразования сложим эти числа: 0.1234 + 9.8766 = = 10.0000. Результат равен границе десятичной дроби. Поскольку формат чисел содержит 5 разрядов (один знаковый и 4 цифровых), полученная при сложении чисел единица переноса выходит за рамки формата, и истинный результат равен 0.0000, что подтверждает правильность преобразования отрицательного числа в дополнительный код. Аналогичные действия используются для обратного перевода дополнительного кода отрицательного числа в прямой код, т. е. для получения модуля числа.

Смысл использования дополнительного кода заключается в том, что, во-первых, арифметические операции вычитания и сложения чисел в дополнительных кодах сводятся к операции алгебраического суммирования (вычитание заменяется сложением уменьшаемого с дополнительным кодом вычитаемого); во-вторых, обработка знаковой и цифровой частей чисел при сложении производится по одним и тем же правилам, причём правильный знак результата формируется автоматически. Заметим, что в дополнительном коде, в отличие от прямого и обратного кодов, существует единственное *представление нуля* (для дробных чисел 0 = 0.00...00), но вместе с тем имеется и одно особое отрицательное число вида 1.0...00, которое является своим собственным дополнением, т. е. это число не имеет дополнительного к нему положительного числа. Поэтому в дополнительных кодах область представления положительных и отрицательных чисел оказывается несимметричной относительно нуля.

1.1.5. Понятие об арифметике ограниченной точности. Переполнение и антипереполнение

В микроконтроллерах, как и в любых других технических устройствах, выполняющих автоматические вычисления, количество битов (разрядов) n, отводимых для представления чисел, ограниченно и фиксированно. Этот факт наряду с использованием для представления чисел двоичной системы приводит к существенным отличиям машинной арифметики от обычной арифметики, реализуемой при помощи карандаша и бумаги (такую арифметику принято именовать арифметикой неограниченных ресурсов). Назовем числа, представляемые в фиксированной разрядной сетке, *числами с ограниченной точностью*, а машинную арифметику — *арифметикой ограниченной точности* (АОТ). Эти термины подчеркивают принципи-

альную особенность машинной арифметики, которую программист постоянно должен иметь в виду.

Рассмотрим основные свойства АОТ. Как следует из формул (1.6), (1.7), числа $|A_R| > A_{R(\max)}$ и $|A_R| < A_{R(\min)}$ нельзя представить в n-разрядном формате, поскольку представляемые числа принадлежат множеству, или *диапазону*, $\{[-A_{R(\max)}, -A_{R(\min)}], 0, [+A_{R(\min)}, +A_{R(\max)}]\}$, см. **Рис. 1.2*а***. Для двоичных дробных и целых чисел эти крайние (пограничные) значения ненулевых представимых чисел определены формулами (1.3) и (1.5), см. **Рис. 1.2*б, в***. Если при выполнении арифметической операции появляется число $|A_R| > A_{R(\max)}$, это приводит к переполнению n-битного формата числа и, как правило, к ошибочному представлению результата операции (ошибка *переполнения*). Наоборот, если при выполнении операции возникает такое ненулевое число, что $|A_R| < A_{R(\min)}$, это приводит к антипереполнению (*потере значности*) формата и соответственно к ошибке *антипереполнения*, при которой все ненулевые числа $A_R \in \{-A_{R(\min)}, +A_{R(\min)}\}$ приходится представлять в виде нуля. Любое число из указанного интервала называют *машинным нулем*.

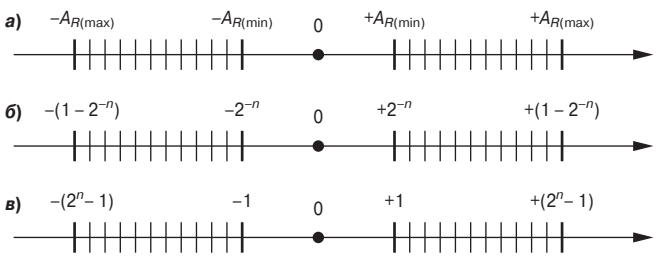

Рис. 1.2. Размещение чисел ограниченной точности на числовой оси:
а) R-ичные числа; *б*) двоичные дробные числа; *в*) двоичные целые числа.

Появление ошибок переполнения и машинного нуля в АОТ нарушает ряд аксиом, истинных в арифметике точных чисел, в частности *ассоциативный* $(a + (b - c) = (a + b) - c)$ и *дистрибутивный* $(a \times (b - c) = a \times b - a \times c)$ *законы*, что можно показать на примерах. Пусть дробные *беззнаковые* десятичные числа имеют в 4-разрядном формате значения $a = 0.2000$, $b = 0.9000$ и $c = 0.8999$. Тогда вычисление левой части выражения ассоциативного закона даёт значение $0.2000 + (0.9000 - 0.8999) = 0.2001$, а при вычислении правой части возникает переполнение: $(a + b) = 0.2000 + 0.9000 > 0.9999$. Аналогично при вычислении правой части выражения дистрибутивного закона получаем значение $0.2000 \times 0.9000 - 0.2000 \times 0.8999 = 0.1800 - 0.1799 = 0.0001$, а при вычислении левой части возникает машинный нуль: $a \times (b - c) = 0.2000 \times (0.9000 - 0.8999) = 0.2000 \times 0.0001 < 0.0001$. Поскольку значения правых и левых частей выражений не совпадают,

данные законы нарушаются, и, следовательно, в АОТ для получения правильного результата вычислений имеет значение *порядок операций*, позволяющий избежать ошибок переполнения и машинного нуля. Заметим, что порядок операций оказывает влияние и на *точность результата*. Выявление ошибок переполнения и антипереполнения является задачей, которую программисту (и только ему!) необходимо решать при разработке своих арифметических программ.

Диапазон $[A_{R(\min)}, A_{R(\max)}]$ точного представления в АОТ ненулевых дробных R-ичных чисел имеет дискретный характер, т. е. в пределах этого диапазона точно представляется лишь то множество $R^n - 1$ чисел, в R-ичной записи которых отличные от нуля цифры содержатся только в первых n после точки разрядах (остальные дополнительные разряды, если предположить их существование, будут содержать нули). На практике приходится представлять в n-разрядном формате любые дробные числа из указанного диапазона, в том числе и те (а их большинство), разряды которых, выходящие за рамки принятого формата, содержат ненулевые R-ичные цифры. Например, десятичная дробь 0.1 в двоичном формате представляется бесконечной периодической двоичной дробью 0.0(0011). При записи этой дроби в ограниченном формате приходится отбрасывать ряд ненулевых цифр и тем самым представлять её как приближённое число с ограниченной точностью.

Числа с ограниченной точностью порождаются не только в процессе их начального представления в ограниченном формате, создающем погрешность представления, но и при выполнении арифметических операций над точными или приближёнными исходными числами. Так, например, в ***арифметике целых чисел***, не имеющей погрешности представления исходных чисел, операция деления порождает неправильные дроби (говорят, что множество целых чисел незамкнуто относительно операции деления), которые в формате целых чисел записываются приближённо целыми числами ограниченной точности. Аналогичные погрешности операций могут иметь место при умножении и делении дробных чисел.

1.1.6. Округление чисел

Как погрешности представления, так и погрешности операций имеют одну природу, связанную с округлением чисел. Поэтому рассмотрим правила округления чисел и величины, характеризующие точность округлённых чисел. Округление точного числа A_R^*, содержащего $n + k$ ($k = 1, 2, ...$) R-ичных разрядов, заключается в ограничении его формата n разрядами. При выполнении этой операции желательно обеспечить наибольшую близость округлённого числа A_R (число в формате n) к округляемому числу A_R^* (число в формате $n + k$).

На практике обычно используют два способа округления: отбрасывание и симметричное округление. *Способ отбрасывания* определяется для дробных и целых R-ичных чисел формулами (1.6) и (1.7). Согласно этому способу дополнительные k разрядов округляемого числа A_R^* просто исключаются из формата без какой-либо коррекции части числа в разрядах n. *Симметричное округление* определяется следующими выражениями соответственно для дробных и целых R-ичных чисел:

$$A_{R\text{ф}} = \begin{cases} \pm\sum\limits_{n}^{n} a_i \cdot 10_R^{-1}, & \text{если } a_{n+1} < R/2; \\ \pm\sum\limits_{i=1}^{i=1} a_i \cdot 10_R^{-i} + 10_R^{-n}, & \text{если } a_{n+1} \geq R/2; \end{cases} \tag{1.9}$$

$$A_R = \begin{cases} \pm\sum\limits_{i=0}^{n-1} a_i \cdot 10_R^{i}, & \text{если } a_{-1} < R/2; \\ \pm\sum\limits_{i=0}^{n-1} a_i \cdot 10_R^{i} + 1, & \text{если } a_{-1} \geq R/2. \end{cases} \tag{1.10}$$

При симметричном округлении значение первой из отбрасываемых цифр округляемого числа A_R^* используется для решения вопроса о коррекции части числа в оставшихся разрядах n. На практике способ симметричного округления для случая $a_{n+1} = R/2$ ($A_{-1} = R/2$) иногда дополняют *правилом Гаусса*, в соответствии с которым коррекция оставшейся части числа производится, если цифра a_n (a_0) — чётная. Оценим точностные характеристики методов округления: абсолютную и относительную ошибки (погрешности).

Определим *абсолютную ошибку округления* ΔA_R как разность значений округляемого A_R^* и округлённого A_R чисел, а *относительную ошибку округления* δA_R как модуль отношения абсолютной ошибки к значению округляемого числа:

$$\Delta A_R = A_R^* - A_R;\ \delta A_R = \left|\Delta A_R / A_R^*\right|. \tag{1.11}$$

Поскольку в большинстве случаев точное значение округляемого числа A_R^* неизвестно (например, число получено с погрешностью в процессе измерения является иррациональным и т. п.), то неизвестны и точные величины ошибок в выражениях (1.11). Однако почти всегда имеется воз-

можность оценить граничные (предельные) значения ошибок. Определим *граничную абсолютную* $\Delta_г$ и *относительную* $\delta_г$ *ошибки округления* следующим образом:

$$|\Delta A_R| \le \Delta_г;\ \delta_г = \Delta_г / |A_R|. \qquad (1.12)$$

Замена $|A^*_R|$ из формулы (1.11) на $|A_R|$ в выражении (1.12) не вносит существенной погрешности в значение $\delta_г$, если $\Delta_г << |A_R|$. На практике используют по возможности минимальное значение граничной ошибки $\Delta_г$. В тех случаях, когда известна точная величина ошибки ΔA_R, она одновременно принимается и в качестве граничной. Очевидно, что граничная абсолютная ошибка любых R-ичных чисел, представленных в n-разрядном формате, имеет значения $\Delta_г = 10^{-n}{}_R$ и $\Delta_г = 1$ соответственно для дробных и целых чисел при способе округления с отбрасыванием и $\Delta_г = 10^{-n}_R\ /2$ и $\Delta_г = 1/2$ соответственно для дробных и целых чисел при симметричном округлении.

Способ отбрасывания порождает всегда округлённые числа «с недостатком», т. е. числа с положительной абсолютной ошибкой, поэтому этот способ называют ещё *несимметричным округлением*. Симметричный же способ, в зависимости от значения первой отбрасываемой цифры, порождает как числа «с недостатком», так и числа «с избытком» (числа с отрицательной абсолютной ошибкой), что, как правило, приводит в процессе многочисленных округлений при вычислениях к *компенсации ошибок* и повышению точности результата. Процессу компенсации ошибок способствует и вышеупомянутое правило Гаусса (оно основывается на предпосылке о равновероятности чётных и нечётных чисел). Способ *симметричного округления* применяется преимущественно в программах, выполняющих арифметическую обработку.

В *арифметике с фиксированной точкой* все числа представляются с одинаковой граничной абсолютной ошибкой. Величина же граничной относительной ошибки зависит от модуля округлённого числа и возрастает от максимального числа к минимальному:

$$\Delta_г / |A_R|_{\max} \le \delta_г \le \Delta_г / |A_R|_{\min}, \qquad (1.13)$$

$$2^{-n} \le \delta_{г2} \le 1, \qquad (1.14)$$

где $\delta_г$, $\delta_{г2}$ — ошибки R-ичных и двоичных чисел, использующих правило округления с отбрасыванием.

Из выражений (1.13), (1.14) следует, что малые числа, близкие к машинному нулю, могут иметь высокую *относительную ошибку* (до 100% при округлении с отбрасыванием и до 50% при симметричном округлении),

т. е. обладать низкой точностью. Если такие числа участвуют в промежуточных вычислениях, *точность результата* будет того же порядка. Поэтому при программировании вычислений важно обеспечить требуемую точность как для исходных чисел, так и для промежуточных результатов. Это достигается правильным выбором порядка операций и значности чисел.

1.1.7. Значность числа

Программисту для уяснения путей достижения требуемой точности вычислений необходимо иметь представление о том, что такое значность числа. *Значность* определяет не общее количество цифр в изображении числа, а только количество значащих цифр, т. е. всех верных цифр числа, кроме нулей, стоящих слева в изображении числа. Например, числа 12.3; 0.123; 0.000123 имеют по три значащие цифры. Цифры в записи приближённого числа принято считать верными, если граничная абсолютная ошибка представления этого числа не превосходит половины единицы (иногда целой единицы) его младшего разряда. Очевидно, что в силу этого определения все цифры округлённого числа (при точном округляемом числе) являются верными, хотя фактически могут не совпадать с соответствующими цифрами точного числа. Например, при *симметричном округлении* точного числа 0.17997 до четырёх цифр округлённое число имеет значение 0.1800, не совпадающее с точным в трёх разрядах, но тем не менее имеющее все верные цифры.

Значность чисел определяет *граничную относительную ошибку* их представления: чем выше значность, тем меньше эта ошибка и, следовательно, тем выше точность числа. Точному числу на числовой оси соответствует точка, а округлённому — интервал чисел. На **Рис. 1.3** показаны интервалы $0.05 < 0.1 < 0.15$ и $0.095 < 0.10 < 0.105$ округлённых чисел 0.1 и 0.01 соответственно с одной и двумя значащими цифрами. Чем выше значность, тем уже интервал неопределенности и тем ближе округлённое число к точному.

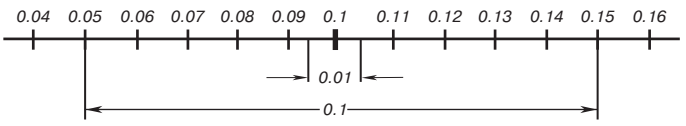

Рис. 1.3. Интервалы округлённых чисел различной значности на числовой оси.

Заметим, что в *арифметике с фиксированной точкой* (*АФТ*), в отличие от арифметики с плавающей точкой (об этом будет сказано далее), повышение значности приводит одновременно к существенному увеличению диа-

пазона представимых чисел, т. е. точность и диапазон чисел оказываются тесно связанными друг с другом. Повышение значности в 8-битных микроконтроллерах достигается, как правило, побайтным увеличением формата чисел (8, 16, 24 двоичных битов и т. д.), поскольку *байт* — 8-битное двоичное слово — является структурной единицей, требующей минимальных программных затрат для обработки (доступ же к отдельным битам байта связан с усложнением обработки). Программы сложения, вычитания, умножения и деления, рассматривавшиеся в томе 3 настоящего издания, оперируют с данными различного формата, что позволяет выбрать ту или иную программу в зависимости от требуемых точности вычислений и диапазона представления чисел.

При использовании арифметики с фиксированной точкой для выполнения сложных вычислений (последовательностей арифметических операций) возникает необходимость расчёта масштабов, обеспечивающих значения модулей исходных промежуточных и результирующих чисел в пределах заданного диапазона их представлений. В данной книге для вычислений сложных выражений (см. ниже) применяется *арифметика с плавающей точкой*, поэтому здесь вопросы расчёта масштабов не рассматриваются.

1.1.8. Граничные абсолютные ошибки в арифметических операциях с числами ограниченной точности

Операции сложения и вычитания точных чисел не вносят погрешности в результат. Свойства этих операций для чисел ограниченной точности определяются следующим правилом:

> ***граничная абсолютная ошибка суммы или разности чисел не превышает сумму граничных ошибок представления слагаемых:***

$$\Delta_{\text{г}}\left(\sum_{i=1}^{k}\pm X_i\right)\le\sum_{i=1}^{k}\Delta_{\text{г}}(X_i)\,. \tag{1.15}$$

Это правило учитывает наихудший случай распределения ошибок округления слагаемых: когда ошибки независимы и одного знака. При значительном числе слагаемых, полученных методом симметричного округления, происходит взаимная *компенсация ошибок* разного знака, и истинная абсолютная ошибка суммы (разности) лишь в исключительных случаях близка к граничной.

Граничная относительная ошибка суммы слагаемых одного знака находится между наименьшей и наибольшей из граничных относительных ошибок слагаемых, с учётом формулы (1.12):