МДК.01.02

Ремонт, регулировка, испытания и сдача систем автоматики

Тема 2.1. Математические основы электронно-вычислительной техники

Полный конспект лекции на 2 часа

Теория с примерами, схемами и контрольными вопросами

6-й разряд · 4 академических часа: 2 часа теории + 2 часа ЛПЗ

Основа: КУПД «Слесарь по КИПиА» от 21.04.2026 № УПД-15

Учебно-методический комплект · 2026

Содержание

1. Зачем слесарю КИПиА двоичная математика ····· 3
1.1. Информация, сигнал и код ····· 3
1.2. Бит, байт, слово и регистр ····· 3
2. Позиционные системы счисления ····· 4
2.1. Разряд и вес позиции ····· 4
2.2. Десятичная, двоичная и шестнадцатеричная системы ··· 5
3. Перевод чисел между системами ····· 6
3.1. Из двоичной в десятичную ····· 6
3.2. Из десятичной в двоичную ····· 6
3.3. Двоичная ↔ шестнадцатеричная ····· 7
4. Представление целых чисел в вычислительной системе ··· 8
4.1. Беззнаковые числа и диапазон ····· 8
4.2. Прямой, обратный и дополнительный код ····· 8
4.3. Переполнение и знаковый разряд ····· 9
5. Двоичная арифметика ····· 9
5.1. Сложение ····· 9
5.2. Вычитание ····· 10
5.3. Умножение и деление ····· 10
6. Фиксированная и плавающая точка ····· 10
6.1. Масштаб и разрешение ····· 10
6.2. Плавающая точка ····· 10
7. Биты, маски и слова состояния в автоматике ····· 11
7.1. Чтение отдельных битов ····· 11
7.2. Шестнадцатеричная запись регистров ····· 12
8. Практический пример измерительного канала ····· 12
9. Типичные ошибки и способы самопроверки ····· 12
10. Итоговый алгоритм и контрольные вопросы ····· 13

1. Зачем слесарю КИПиА двоичная математика

Современная система автоматики воспринимает физические величины через датчики, преобразует их в электрические сигналы, а затем - в цифровые коды. Даже если слесарь КИПиА не пишет программу контроллера, при диагностике ему приходится читать регистры, адреса, слова состояния, шестнадцатеричные коды ошибок и двоичные маски. Поэтому системы счисления - это не отвлечённая математика, а рабочий язык цифровой части автоматики.

На практике цифровой код встречается почти на каждом этапе: в показании АЦП, в параметре конфигурации, в адресе устройства, в слове состояния, в журнале обмена и в сообщении об ошибке. Один и тот же физический параметр может последовательно существовать как ток 4–20 мА, напряжение на входе АЦП, двоичный код, масштабированное целое число и отображаемое инженерное значение. Специалисту важно понимать переходы между этими представлениями, иначе трудно отличить неисправность датчика от ошибки масштаба, формата или передачи данных.

В этой теме не требуется становиться программистом. Требуется научиться читать машинное представление осознанно: определить основание системы счисления, разрядность, знак, масштаб и назначение отдельных битов. Эти пять вопросов образуют простой диагностический алгоритм, который будет использоваться во всех последующих темах МДК.01.02.

Практический смысл. Если на экране диагностической программы показано STATUS = 0x000B, специалист должен понимать, что это 0000 0000 0000 1011_2 и что установлены биты 0, 1 и 3. Без этого невозможно уверенно разобрать причины блокировки или срабатывания защиты.

1.1. Информация, сигнал и код

Сигнал - физическая форма передачи информации. В аналоговом канале значение может изменяться непрерывно, например 4-20 мА. В цифровом устройстве результат представляется конечным набором символов. Минимальный двоичный символ - бит со значением 0 или 1. Совокупность битов кодирует число, состояние или команду.

Следует различать физический уровень и логическое значение. Логическая единица не обязана буквально означать 1 В, а логический ноль - 0 В. Электрические уровни задаются конкретным интерфейсом и схемой. В рамках темы 2.1 мы работаем с абстрактными значениями 0 и 1, а электрические параметры цифровых интерфейсов будут рассматриваться позже.

Код сам по себе не содержит смысла. Например, 0000 0101_2 может означать число 5, комбинацию двух активных флагов, часть адреса, код команды или фрагмент измеренного значения. Значение возникает только вместе с описанием формата. Поэтому профессиональная привычка - сначала определить тип данных, а затем выполнять вычисления.

- 0 и 1 могут соответствовать двум устойчивым уровням напряжения;
- один бит может обозначать состояние «выключено/включено», «норма/авария», «нет/есть»;
- несколько битов образуют поле или число;
- смысл кода определяется документацией: один и тот же набор битов без описания регистра интерпретировать нельзя.

1.2. Бит, байт, слово и регистр

Бит - один двоичный разряд. Восемь битов образуют байт. Два байта часто образуют 16-разрядное слово. В контроллерах и цифровых преобразователях используются регистры

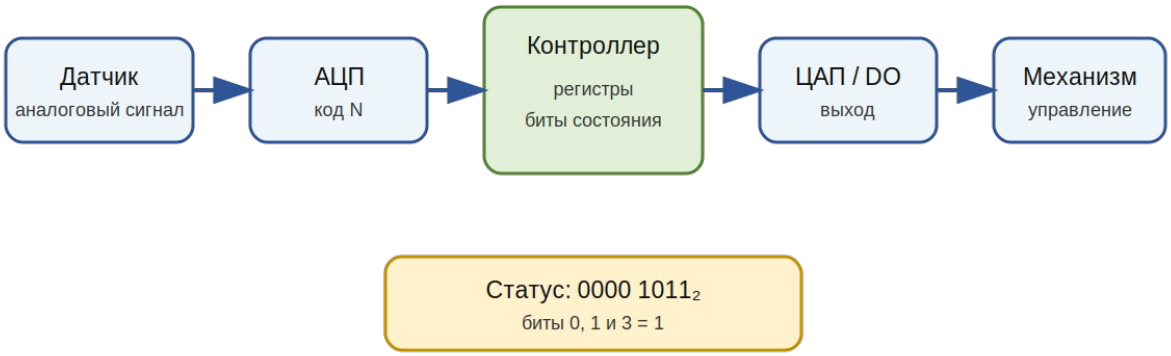
фиксированной разрядности: 8, 16, 32 бита и более. Разрядность определяет число возможных комбинаций и диапазон представимых целых чисел.

При чтении байта биты обычно нумеруют справа налево: младший бит имеет номер 0, затем 1, 2 и так далее. В 8-разрядном байте номера идут от 0 до 7. В 16-разрядном слове - от 0 до 15. Термины младший разряд (LSB) и старший разряд (MSB) описывают положение разряда по значимости. Это важно при чтении масок и при работе с последовательными интерфейсами.

Количество возможных комбинаций равно 2 в степени количества битов. У одного бита две комбинации, у двух - четыре, у восьми - 256. Однако число комбинаций и числовой диапазон - не одно и то же: если часть комбинаций используется для отрицательных значений, диапазон меняется.

Разрядность	Комбинаций	Беззнаковый диапазон
8 бит	$2^8 = 256$	0…255
16 бит	$2^{16} = 65\,536$	0…65 535
32 бита	2^{32}	0…4 294 967 295

Рисунок 1. Где двоичные данные встречаются в автоматике



Двоичный и шестнадцатеричный формат нужен для регистров, масок, статусов и диагностики.

Рисунок 1. Двоичные данные внутри измерительного и управляющего канала.

2. Позиционные системы счисления

2.1. Разряд и вес позиции

В позиционной системе значение цифры зависит от её положения. Вес разряда равен основанию системы в степени, соответствующей позиции. В десятичной системе основание 10, в двоичной - 2, в шестнадцатеричной - 16. Нумерацию целых разрядов удобно начинать справа с нуля.

Общая идея одинакова для любой позиционной системы: число является суммой произведений цифр на веса соответствующих разрядов. Поэтому привычное десятичное 347 означает $3\times10^2 + 4\times10^1 + 7\times10^0$. В двоичной системе меняется только основание: каждый следующий разряд слева имеет вес в два раза больше предыдущего.

Для самопроверки полезно оценивать порядок величины. Если в 8-разрядном числе установлен только старший бит, значение равно 128. Если дополнительно установлен младший - 129. Такая оценка помогает быстро замечать ошибку в весах разрядов.

Для двоичного числа 101101_2 веса справа налево равны 2^0, 2^1, 2^2, 2^3, 2^4, 2^5. Поэтому 101101_2 = 1×32 + 0×16 + 1×8 + 1×4 + 0×2 + 1×1 = 45_{10}.

Рисунок 2. Разрядные веса позиционной системы

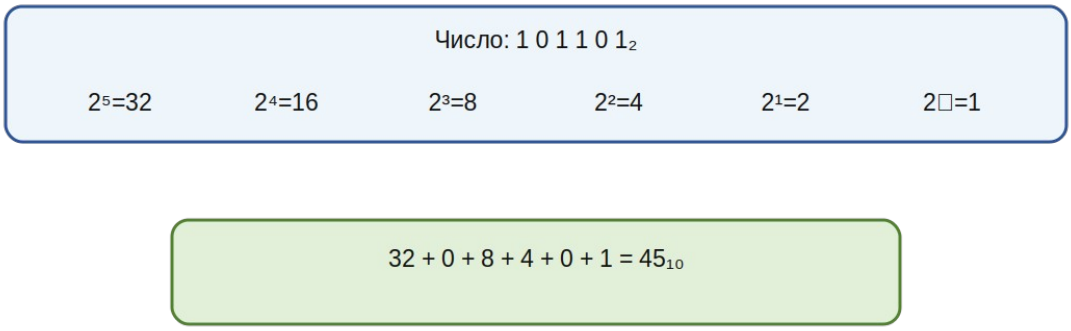


Рисунок 2. Разрядные веса двоичного числа.

2.2. Десятичная, двоичная и шестнадцатеричная системы

Система	Основание	Цифры	Пример
Десятичная	10	0…9	45_{10}
Двоичная	2	0, 1	101101_2
Шестнадцатеричная	16	0…9, A…F	$2D_{16}$

Шестнадцатеричная система удобна потому, что одна её цифра соответствует ровно четырём двоичным разрядам. Это делает длинные машинные слова компактными и легко проверяемыми.

После цифры 9 в hex используются символы A, B, C, D, E и F, соответствующие десятичным значениям 10…15. Запись 0x2D обычно читается как шестнадцатеричное число 2D. Индекс $_{16}$ и префикс 0x выполняют одну задачу - не дать перепутать основание.

Шестнадцатеричная запись особенно полезна для 8-, 16- и 32-разрядных данных. Байту соответствуют две hex-цифры, 16-разрядному слову - четыре, 32-разрядному - восемь. Поэтому значение 0x00FF сразу показывает структуру 16-разрядного слова лучше, чем десятичное 255.

Рисунок 3. Двоичный код и шестнадцатеричные тетрады

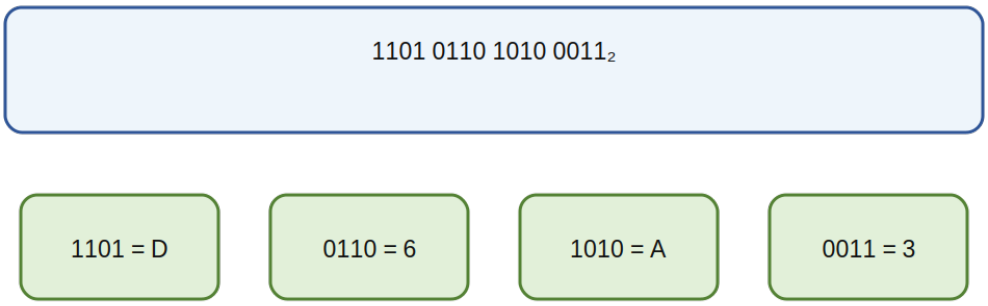


Рисунок 3. Группирование двоичного слова по четыре бита.

3. Перевод чисел между системами

3.1. Из двоичной в десятичную

Алгоритм: подписать веса разрядов, оставить веса только там, где стоит 1, и сложить их. Например, $11001010_2 = 128 + 64 + 8 + 2 = 202_{10}$. Такой способ особенно полезен при разборе масок и слов состояния.

Разберём ещё один пример: $0011\ 0101_2$. Единицы стоят в разрядах 5, 4, 2 и 0. Значение равно $32 + 16 + 4 + 1 = 53_{10}$. Нули можно не учитывать, но позиции пропускать нельзя: частая ошибка - сжимать запись после удаления нулей и тем самым менять веса.

Если число содержит много разрядов, его удобно сначала перевести в hex, а затем при необходимости - в десятичное. Например, $1111\ 0000_2 = F0_{16} = 15 \times 16 + 0 = 240_{10}$. Такой комбинированный способ снижает количество ручных операций.

3.2. Из десятичной в двоичную

Целое положительное число последовательно делят на 2. На каждом шаге записывают остаток 0 или 1. Двоичную запись получают чтением остатков снизу вверх. Для числа 45 остатки дают 101101_2.

Почему остатки читают снизу вверх? Первый остаток показывает младший разряд - чётность исходного числа. Следующий остаток относится к весу 2^1, затем 2^2 и так далее. Последнее деление формирует старший разряд. Именно поэтому порядок записи противоположен порядку выполнения делений.

Проверка результата обязательна: полученное двоичное число переводят обратно суммой весов. Для 101101_2 получаем $32+8+4+1=45$. Если обратный перевод не совпадает с исходным числом, ошибка обычно находится в одном из остатков или в их порядке.

Рисунок 4. Перевод десятичного числа в двоичное

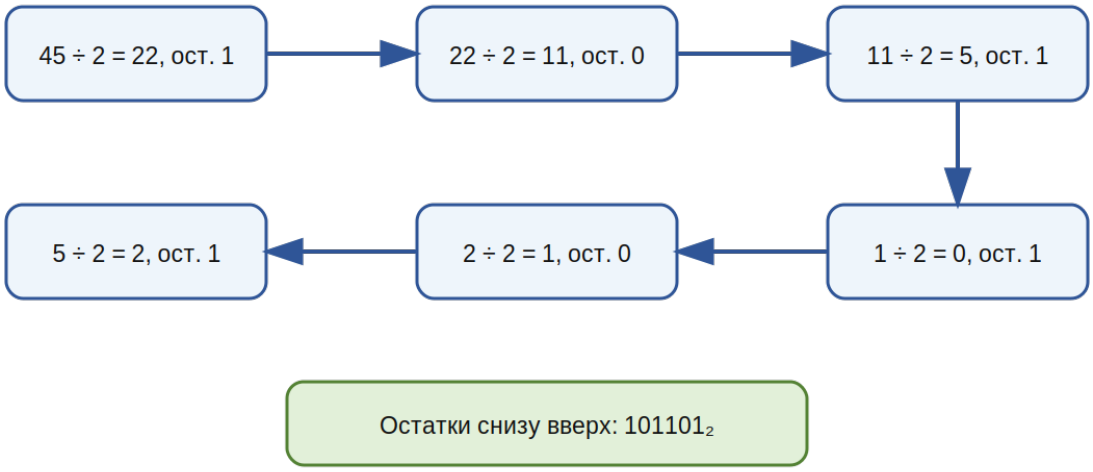


Рисунок 4. Метод последовательного деления на 2.

3.3. Двоичная ↔ шестнадцатеричная

Для перевода двоичного числа в шестнадцатеричное его разбивают справа налево на группы по четыре бита. При необходимости слева добавляют нули. Каждую тетраду заменяют символом 0…F. Обратный перевод выполняют заменой каждой шестнадцатеричной цифры четырьмя битами.

Пример: 10110110_2 делим на 1011 и 0110. $1011_2 = B_{16}$, $0110_2 = 6_{16}$, поэтому результат $B6_{16}$. Для обратного перевода $B6_{16}$ просто заменяем B на 1011, 6 на 0110 и получаем исходные восемь битов.

Для 16-разрядного слова 0xA305 тетрады читаются как A=1010, 3=0011, 0=0000, 5=0101. Получаем 1010 0011 0000 0101_2. Группировка пробелами по четыре бита повышает читаемость и снижает риск пропуска разряда.

Hex	Bin	Hex	Bin
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

4. Представление целых чисел в вычислительной системе

4.1. Беззнаковые числа и диапазон

Если все n разрядов используются для величины, диапазон беззнакового целого равен от 0 до 2^n-1. Например, 8 бит позволяют представить 0…255. При работе с регистром необходимо знать не только числовое значение, но и его тип: беззнаковый, знаковый, набор битов, счётчик, масштабированная величина и т. д.

Беззнаковый формат удобен для счётчиков, номеров, кодов состояния и величин, которые физически не бывают отрицательными. Но даже здесь необходим масштаб. Значение регистра 2530 может означать 2530 импульсов, 253,0 °C, 25,30 % или 2,530 МПа - всё зависит от описания параметра.

При сравнении двух устройств важно убедиться, что одинаковы не только значения регистров, но и их интерпретация. Совпадение числа без совпадения формата ничего не доказывает.

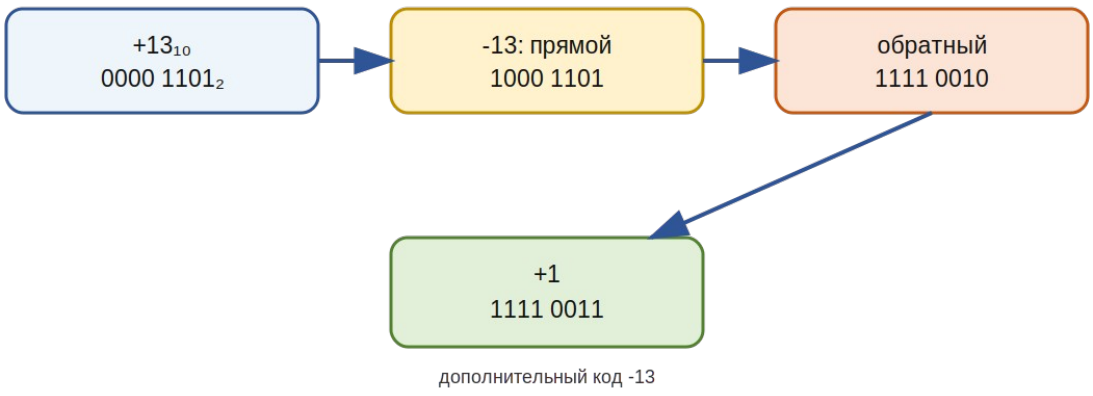
4.2. Прямой, обратный и дополнительный код

Для учебного понимания отрицательных чисел рассматривают прямой, обратный и дополнительный код. В прямом коде старший разряд используется как знак, а остальные разряды задают модуль. Обратный код отрицательного числа получают инверсией битов положительного числа. Дополнительный код получают прибавлением единицы к обратному коду. Именно дополнительный код широко применяется для знаковых целых, потому что упрощает арифметику.

Пример для −13 в 8 битах. Сначала +13 = 0000 1101_2. Инвертируем все восемь битов: 1111 0010_2. Прибавляем 1: 1111 0011_2. Проверка выполняется обратным путём: у отрицательного дополнительного кода инвертировать биты, прибавить 1 и получить модуль 13.

Не следует смешивать учебный прямой код со способом хранения в конкретном контроллере. Документация может объявлять параметр как unsigned, signed integer, BCD, float или специальный формат. Всегда исходят из указанного типа данных.

Рисунок 5. Прямой, обратный и дополнительный код



В реальных процессорах отрицательные целые обычно хранятся в дополнительном коде.

Рисунок 5. Последовательность получения дополнительного кода отрицательного числа.

Важно. Разрядность обязательна. Запись «дополнительный код −13» неполна без указания

8-, 16- или другой разрядности: набор битов зависит от ширины слова.

4.3. Переполнение и знаковый разряд

Для 8-разрядного знакового числа в дополнительном коде диапазон составляет −128…+127. Если результат выходит за этот диапазон, возникает переполнение. В автоматике это опасно тем, что неправильный тип или масштаб регистра может превратить ожидаемое положительное значение в отрицательное или вызвать резкий скачок отображаемой величины.

Например, математически 100+40=140. Но 140 не помещается в 8-разрядный signed-диапазон. Если программа не контролирует переполнение, набор битов может быть истолкован как отрицательное число. Для слесаря это диагностический признак: неожиданный отрицательный параметр может быть не неисправностью датчика, а ошибкой типа или преобразования.

Переполнение отличается от насыщения. Некоторые системы при выходе за диапазон специально удерживают максимальное или минимальное допустимое значение; другие выполняют машинное переполнение. Поведение нужно брать из алгоритма и документации.

5. Двоичная арифметика

5.1. Сложение

Базовые правила: 0+0=0; 0+1=1; 1+0=1; $1+1=10_2$; $1+1+1=11_2$. При 1+1 в текущем разряде записывается 0, а единица переносится в следующий старший разряд.

Пример: $0010\ 1101_2$ (45) + $0001\ 0111_2$ (23). Сложение выполняют справа налево с учётом переносов и получают $0100\ 0100_2$. Обратная проверка: 64+4=68, а 45+23=68. Такая двойная проверка особенно полезна на первых этапах обучения.

При сложении слов фиксированной разрядности необходимо следить за выходным переносом и признаком переполнения. Для беззнакового и знакового форматов критерии переполнения различаются, поэтому сначала определяют тип данных.

Рисунок 6. Двоичное сложение и перенос



Перенос между разрядами - ключевой механизм двоичного сложения.

Рисунок 6. Пример двоичного сложения с переносом.

5.2. Вычитание

Вычитание можно выполнять столбиком с заёмом либо через дополнительный код: A−B = A + (−B). Второй способ отражает реальную работу цифровой арифметики. При ручной проверке полезно перевести исходные числа и результат в десятичную систему и убедиться, что равенство сохраняется.

При вычитании столбиком, если из 0 нужно вычесть 1, выполняют заём из старшего разряда. В двоичной системе занятая единица старшего разряда соответствует 10_2 в текущем. Логика полностью аналогична десятичному вычитанию, только основание равно 2.

В дополнительном коде отдельного аппаратного вычитателя не требуется: отрицательное значение второго операнда складывается с первым. Поэтому понимание дополнительного кода непосредственно объясняет работу арифметического узла процессора.

5.3. Умножение и деление

Умножение двоичного числа на 2 соответствует сдвигу влево на один разряд, если не возникает переполнение. Деление беззнакового целого на 2 соответствует сдвигу вправо. Сдвиги активно применяются в микроконтроллерах, но их нельзя механически применять к любому типу данных без понимания знака и масштаба.

Обычное двоичное умножение строится из сдвигов и сложений: если очередной бит множителя равен 1, соответствующая сдвинутая копия множимого участвует в сумме. Деление аналогично по смыслу последовательному сравнению и вычитанию. Для слесаря важно понимать принцип, а не выполнять длинные машинные операции вручную.

Практический вывод: сдвиг - это операция над битами. Для поля флагов сдвиг используется для построения маски, а для числового значения может менять величину. Поэтому одинаковая операция имеет разный смысл в зависимости от типа данных.

6. Фиксированная и плавающая точка

6.1. Масштаб и разрешение

В системах с фиксированной точкой положение условного разделителя заранее известно. Например, контроллер может хранить температуру 253 как 25,3 °C при масштабе 0,1 °C/ед. Такой формат прост и предсказуем, но диапазон и разрешение связаны с разрядностью.

Масштабирование широко применяется в промышленных регистрах. Значение 0…10000 может соответствовать 0…100,00 %, а 0…27648 - номинальному диапазону аналогового входа конкретного контроллера. Число в регистре и инженерная величина связаны коэффициентом. Ошибка коэффициента выглядит как стабильная, но неверная величина.

Разрешение показывает минимальный шаг представления. Если 12-разрядный АЦП охватывает 0…10 В, существует 4096 кодовых состояний. Идеальный шаг составляет примерно 10/4095 ≈ 0,00244 В на код. Это не то же самое, что точность: реальная погрешность включает дополнительные составляющие.

6.2. Плавающая точка

Формат с плавающей точкой представляет число через знак, мантиссу и порядок. Он позволяет работать с очень большими и очень малыми величинами, но вводит эффекты округления. В промышленном обмене важно знать порядок байтов и формат данных, иначе корректные четыре байта могут быть интерпретированы как совсем другое число.

При чтении float через промышленный протокол недостаточно получить четыре байта. Нужно знать, в каком порядке передаются байты и слова, соответствует ли формат IEEE 754, и действительно ли выбранный адрес начинается с первого байта числа. Ошибка порядка байтов часто даёт «фантастическое» значение при полностью исправной линии связи.

Следовательно, при подозрении на неверное цифровое значение проверяют цепочку: физический сигнал → код АЦП → внутренний тип данных → масштаб → порядок байтов → отображение. Этот подход помогает не менять исправный датчик из-за программной ошибки интерпретации.

Рисунок 7. Фиксированная и плавающая точка

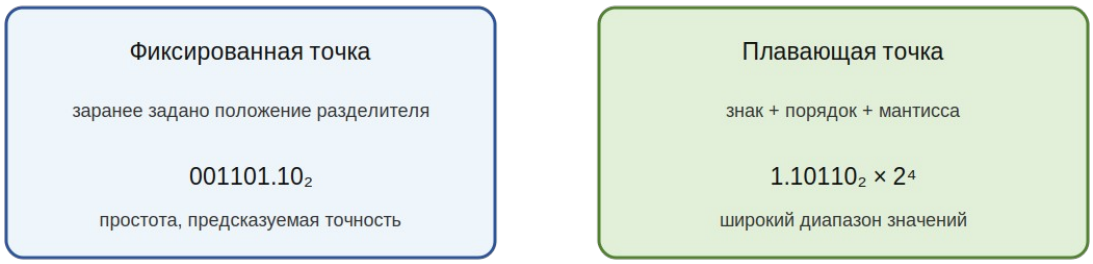


Рисунок 7. Сравнение двух способов представления чисел.

7. Биты, маски и слова состояния в автоматике

Часть регистров содержит не одно число, а набор независимых признаков. Каждый бит имеет собственный смысл: готовность, авария, дистанционный режим, разрешение пуска, состояние входа и т. п. Для выделения признаков применяют маски.

Битовая маска - число, в котором единицы стоят в интересующих позициях. Например, маска $0000\ 1000_2$ = 0x08 выделяет бит 3. Если операция проверки показывает, что соответствующий бит слова установлен, активен связанный с ним признак. В этой теме достаточно понимать смысл маски; конкретные логические операции подробно закрепляются тренажёром логики.

Поле может занимать несколько соседних битов, например биты 4…6 могут задавать код режима от 0 до 7. Тогда одного ответа «бит установлен» недостаточно: поле нужно выделить и интерпретировать как маленькое число. Это типичный приём в регистрах состояния и конфигурации.

7.1. Чтение отдельных битов

Пусть 8-разрядное слово состояния равно 00001011_2. Если нумеровать биты справа с нуля, единицы стоят в позициях 0, 1 и 3. Следовательно, активны функции, закреплённые документацией за этими битами. Нельзя называть их «Авария» или «Пуск» без таблицы регистра конкретного устройства.

Учебный пример: если карта регистра говорит «бит 0 - готов, бит 1 - работа, бит 3 - предупреждение», слово 00001011_2 означает одновременную готовность, работу и предупреждение. Если бит 4 обозначает аварию и он равен нулю, аварийный признак по этой карте не активен. На реальном объекте названия и логика могут быть другими.

Иногда логика инверсная: ноль может означать активное состояние. Поэтому даже правильно определив номер единичного бита, нельзя делать технологический вывод без описания. Документация остаётся источником значения.

7.2. Шестнадцатеричная запись регистров

То же слово можно записать 0x0B. Префикс 0x обычно указывает шестнадцатеричную форму. Для 16-разрядного регистра 0xA305 двоичное представление: 1010 0011 0000 0101_2. Такая запись удобна в диагностических программах, журналах обмена и документации протоколов.

Быстрая проверка выполняется по тетрадам: A=1010, 3=0011, 0=0000, 5=0101. Если диагностический журнал показывает изменение 0x0003 → 0x000B, сравнение в двоичной форме показывает, что дополнительно установился бит 3. Такой приём позволяет быстро локализовать изменившийся признак.

Не следует переводить каждое слово состояния в десятичную систему: десятичная запись часто скрывает структуру битов. Для числового параметра десятичная форма удобна, для флагов - двоичная или hex.

Маска	Двоично	Что проверяет
0x01	0000 0001	бит 0
0x02	0000 0010	бит 1
0x04	0000 0100	бит 2
0x08	0000 1000	бит 3

8. Практический пример измерительного канала

Предположим, 12-разрядный АЦП выдаёт код от 0 до 4095 для входного диапазона 0…10 В. При коде N=2048 напряжение примерно равно половине диапазона: U≈2048/4095×10≈5,00 В. Здесь двоичный код становится числовым представлением аналогового сигнала. В следующей теме МДК.01.02 этот принцип будет подробно рассмотрен для АЦП и ЦАП.

Рассмотрим диагностическую ситуацию. На клеммах аналогового входа измерено около 5 В, но на экране системы отображается 2,5 В. Если сырой код АЦП близок к 2048, сам АЦП, вероятно, видит сигнал правильно. Тогда следует проверить масштабирование и преобразование инженерных единиц. Если сырой код около 1024, искать нужно раньше по цепочке - входной делитель, конфигурацию диапазона или сам канал АЦП.

Другой пример - слово состояния. Устройство не разрешает пуск, а обмен с ним есть. Сравнение status до и после команды показывает изменение одного бита. По карте регистра этот бит может оказаться «нет готовности внешней системы». Двоичное представление помогает отделить коммуникационную неисправность от технологической блокировки.

Таким образом, математические основы используются не изолированно. Они позволяют разбить диагностическую задачу на уровни: физическая величина, электрический сигнал, сырой код, формат данных, алгоритм масштабирования и логические признаки.

Связь с последующими темами. Тема 2.1 даёт математический язык. Тема 2.2 использует его для АЦП/ЦАП, 2.3 - для микропроцессоров, 2.4 - для микроконтроллеров, 2.6 - для регистров и промышленных интерфейсов.

9. Типичные ошибки и способы самопроверки

Большинство ошибок темы 2.1 связано не со сложностью вычислений, а с потерей контекста: забыто основание, пропущен нулевой разряд, не указана разрядность или неверно выбран тип данных. Поэтому правильное оформление является частью решения, а не косметикой.

Ошибка	Почему возникает	Как проверить
Читают биты слева как 0,1,2…	Забывают правило младшего разряда	Подписать позиции справа налево, начиная с 0
Смешивают 1010_2 и 1010_{10}	Не указано основание	Всегда писать индекс или префикс
Hex A принимают за десятичное 10 без контекста	Не понимают тетраду	$A_{16} = 1010_2 = 10_{10}$
Получают неверный дополнительный код	Инвертируют не все разряды	Сначала зафиксировать разрядность
Игнорируют переполнение	Результат математически верен, но не помещается	Проверить допустимый диапазон n бит
Интерпретируют слово состояния как обычное число	Не посмотрели карту битов	Использовать таблицу регистра устройства

10. Итоговый алгоритм и контрольные вопросы

1. Определить основание и разрядность числа.
2. При переводе в десятичную подписать веса позиций.
3. При переводе из десятичной использовать деление на основание и остатки.
4. Для binary ↔ hex группировать по четыре бита.
5. Перед трактовкой отрицательного значения уточнить формат и разрядность.
6. При арифметике контролировать перенос, заём и переполнение.
7. Для регистров различать числовое значение и набор независимых флагов.
8. В автоматике всегда сверять смысл битов, масштаб и порядок байтов с документацией.

Контрольные вопросы

9. Почему шестнадцатеричная запись удобна при работе с регистрами?
10. Чему равно 101101_2 в десятичной системе?
11. Как перевести целое десятичное число в двоичное?
12. Сколько комбинаций имеет 8-разрядное слово?
13. Каков диапазон 8-разрядного беззнакового числа?
14. Как получить дополнительный код отрицательного числа?
15. Что такое переполнение?
16. Чем фиксированная точка отличается от плавающей?
17. Какие биты установлены в слове 00001011_2?
18. Почему нельзя трактовать бит регистра без документации?
19. Как связаны одна hex-цифра и четыре двоичных бита?
20. Где эти знания применяются при диагностике систем автоматики?

Итог. Двоичная математика нужна не ради ручных вычислений, а для осознанного чтения цифровых данных. Специалист должен понимать, как число получено, в каком формате хранится, где находится знак, какой масштаб применён и какие отдельные биты несут диагностические признаки.