Системы счисления

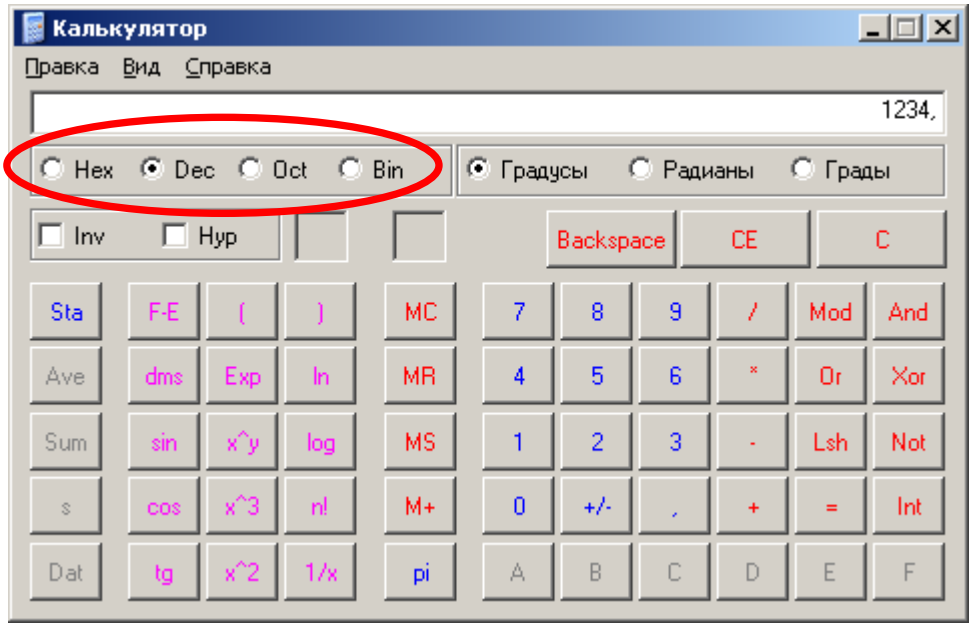
Ну что же, раз мы будем изучать счётчик, так давайте уж вспомним, как вообще происходит счёт. Начнём с привычной нам десятичной системы счисления. Там всё просто (для нас). Есть 10 цифр – от 0 до 9. Вот их мы и перебираем. Давайте попробуем… 0, 1, 2, 3, 4, 5, 6, 7, 8, 9… И что дальше? А дальше тоже просто: Если цифра уже и так максимальная, то её надо занулить, а ту, что слева – увеличить. То есть, 9 меняется на 10. Потом идут 11, 12, 13, 14, 15, 16, 17, 18, 19, ну, и наконец, 20, 21, 22… 97, 98, 99… Здесь увеличиваем крайнюю цифру – получаем правило «Занули и увеличь ту, что левее». Увеличиваем ту, что левее – срабатывает то же правило, так 99 превращается в 100.

Пока ничего нового. Самое смешное, что нового и дальше не будет, просто оно не очевидно на первый взгляд. Давайте представим себе, что к нам прилетели инопланетяне и украли цифры 8 и 9. У нас осталось всего восемь цифр – от 0 до 7. К слову сказать, в очень распространённых ЭВМ 70х-80х годов PDP-11 (они же СМ ЭВМ, они же ДВК, они же УКНЦ, они же БК-001Х) так и было, так что пример взят не с потолка. Итак, давайте посчитаем с такими цифрами… 0, 1, 2, 3, 4, 5, 6**, 7, 10,** 11, 12, 13, 14, 15, 16**, 17, 20,** 21, …, 66**, 67, 70,** 71, 72, 73, 74, 75, 76**, 77, 100**, 101… Как видим, по-прежнему, ничего нового нет. Хоть в десятичной, хоть в восьмеричной системе мы считаем по тому же правилу: Увеличивай знак, а если оно максимально возможное – занули и увеличь знак слева от текущего (по тому же правилу). Просто в одном случае максимумом была девятка, а в другом – семёрка.

Ну вот, аналогично можно ввести чисто гипотетическую четверичную систему счисления (практического смысла этой системы мне не известно, но я просто стараюсь понижать основание системы плавно, а не резко). Там счёт будет примерно такой: 0, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33, 100, 101, 102, 103, 110… Логично? По-моему, да. Если не согласны – попрактикуйтесь ещё.

И, наконец, давайте понизим основание системы счисления до двух. Здесь бывают только две цифры – 0 и 1. То есть, достигнув единицы, мы уже получаем то самое правило: «Занули и увеличь слева». Ну что поделаешь, если такая система счисления (по числу логических уровней в проводе). Что получаем: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, … Как видим, всё то же самое, просто если перейти сразу от десяти чисел к двум, оно не так бросается в глаза, если же понижать сразу, к двоичной системе, глаз уже намётан.

Как переводить числа из системы в систему? Существует красивая теория, которой в наше время грузили первокурсников. Все числа переводятся, но на это тратится море времени. Интересно, кто-нибудь из «загруженных» помнит эту теорию? Лично я не сразу и вспомню, так как вывел свою (потом узнал, что точно так же работают регистры последовательного приближения ☺), но она требует запоминания ряда констант, поэтому не буду грузить и ей. Для частных случаев (восьмеричная<->двоичная и шестнадцатеричная<->двоичная) есть элементарные правила, я потом их по мере надобности приведу… А пока – запомним, что калькулятор в Windows, будучи переведённым в инженерный вид, поможет нам перевести числа из любой системы в любую (имеются в виду, системы, имеющие практическое значение). Достаточно выбрать исходную систему, ввести число, а затем – задать целевую систему. И число само преобразуется…



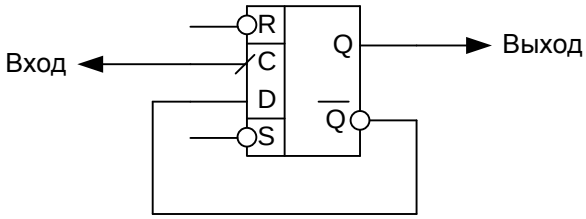
Уж коли разрядность чисел в электронике фиксированная (помните 16-разрядные компьютеры, 32-разрядные компьютеры, 64-разрядные компьютеры? Это число проводов в шине данных), то значения должны иметь одну и ту же ширину. Например, значение 1 в четырёхразрядном виде будет выглядеть, как 0001. Собственно, механические электросчётчики, либо счётчики воды – они ведь тоже имеют фиксированную разрядность и исходно занулены… Так что пойдёте домой – гляньте на лестничной клетке образец. Поэтому запишем увеличивающееся четырёхразрядное двоичное число по этим правилам:

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

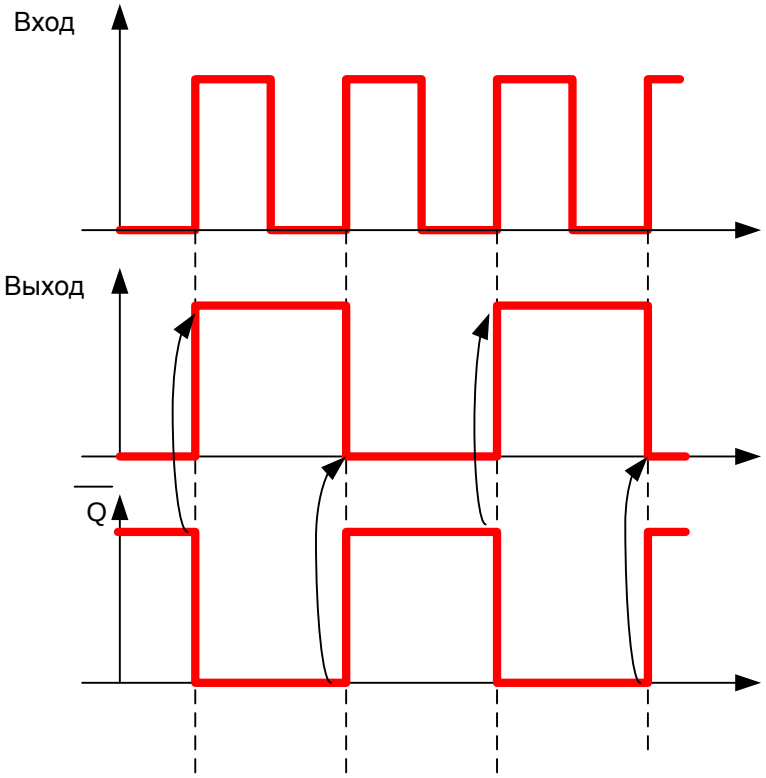
Ну, собственно, пока и всё. Как говорится, если на сцене в первом акте висит ружьё, то в последнем оно выстрелит. Вот этот столбец мы оставим в первом акте, а пока перейдём кто второму.

T-триггер

А давайте включим D-триггер, изученный в прошлый раз, вот в таком виде:



Что получаем? А получаем мы довольно забавную вещь. Пусть исходно на прямом выходе Q у нас ноль, тогда на инверсном – единица. Если мы подадим положительный перепад на вход C, эта единица у нас запрыгнет в прямой выход, а на инверсном появится ноль. При следующем положительном перепаде на входе C, этот ноль прыгнет в выход… И так далее. Графически это можно представить так:

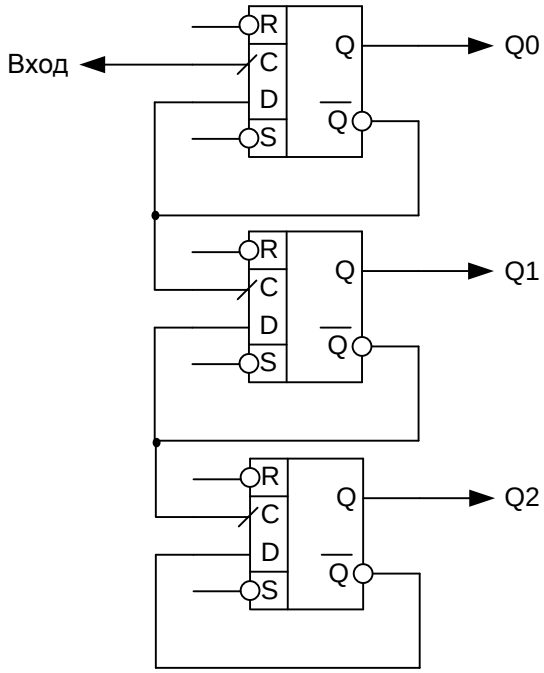


Итак. Мы видим два важных свойства этого включения D-триггера. Во-первых, он меняет состояние на выходе по каждому положительному перепаду на входе. Во-вторых (это следует из первого), частота импульсов на выходе ровно в два раза ниже, чем на входе (то есть, на нём легко делаются делители частоты, но о них мы поговорим в практической части статьи).

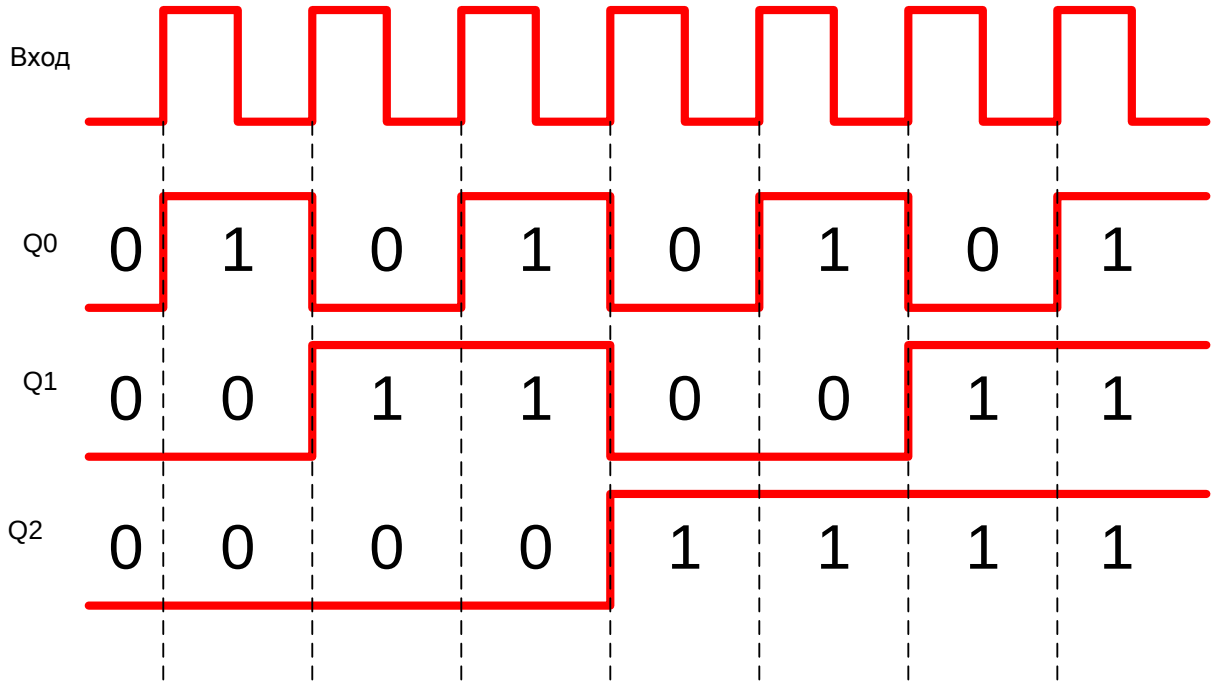
Такой триггер называется счётным триггером, или T-триггером.

Асинхронный счётчик

Ну вот. А теперь давайте возьмём не один, а три T-триггера. Можно было бы взять и больше, но графики будут слишком скомканные, так что ограничимся тремя. И включим мы их вот так:



К ранее сказанному, на словах можно добавить только то, что следующий триггер получит положительный перепад только тогда, когда на инверсном выходе сменится 0 на 1, а на прямом – 1 на 0. Ничего не напоминает? Тогда поднимаем глаза к правилу счёта: Если меняем максимальное число (а у нас это 1) на 0, то увеличим следующую цифру. Вот как раз это и есть сигнал следующему Т-триггеру: «Давай, перебрасывайся». Для пущей убедительности зарисуем работу наших триггеров графически (чтобы не загромождать рисунок, я не стал приводить инверсных выходов, достройте их мысленно или в качестве самостоятельной работы на бумаге):



Выпишем значения Q2 Q1 Q0 (именно в такой последовательности) для каждого такта:

000

001
010
011
100
101
110
111

Возвращаемся к ружью в первом акте, и понимаем, что оно выстрелило. Мы сделали такую конструкцию, которая САМА СЧИТАЕТ! И называется она – Асинхронный Счётчик. Где могут использоваться счётчики? Ну, возьмём, к примеру, счётчик адреса команд в любом процессоре. По сигналу Reset он сбрасывается (заводим сигнал Reset от смывного бачка на все входы R наших триггеров, они и сбросятся), а дальше – по тактовым импульсам – считает, перебирая наши команды в памяти. В Новгородском офисе многие помнят, как я рассказывал о таймерах (правда, тогда почти никто почти ничего не понял, но сейчас уже понятнее?). Ну, и так далее. Мы ещё со счётчиками будем часто встречаться.

Правда, предупрежу, что асинхронные счётчики были популярны в XX веке. Тогда частоты были небольшие, а микросхемы – слабые. При чём тут частота? Обратите внимание, что переброска триггеров идёт один за другим. Пусть у нас огромное число, и надо перебросить его от 1111111111111111 к 10000000000000000. Сначала перебросится триггер 0, затем – он пошлёт сигнал на триггер 1. Далее, перебросится триггер 1 и пошлёт сигнал на триггер 2. Далее перебросится триггер 2 и пошлёт сигнал на триггер 3. Ну, и так далее. Мы помним, что любой логический вентиль вносит задержку в сигнал. Мы также помним, что каждый D-триггер состоит из большого количества логических вентилей. А значит – задержа будет значительная. А если у нас число 32 битное, то она увеличится в 32 раза. А если частота высокая (скажем, 200 МГц)? Тогда пока сороконожка в лице наших триггеров передёрнет все лапки, сигнал на входе C уже перекинется ещё раз. И первые триггеры уже начнут новую жизнь. И мы получим не значение, а чёрте что. Вот так.

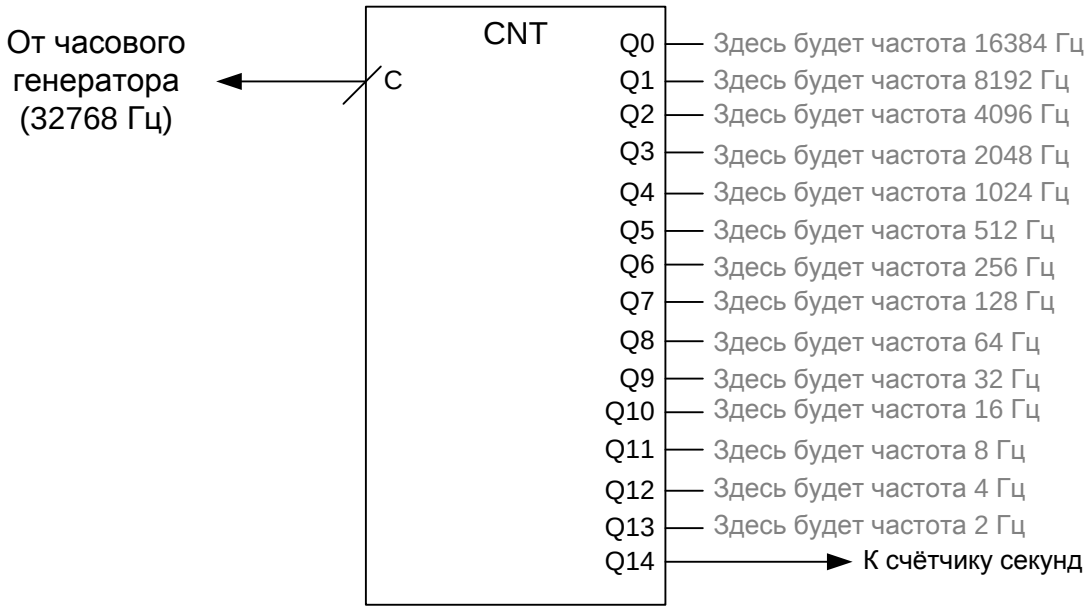
Именно поэтому сейчас переходят на синхронные счётчики, у которых при дёргании входа, перебросятся сразу все триггеры. Однако, такие счётчики достаточно сложны (поэтому мы их пока не рассматриваем, хотя со временем, как доберёмся до сумматоров, один из них краешком глаза глянем). Ну, и в силу сложности, они приходят в нашу жизнь только сейчас, когда число транзисторов в микросхемах достигло комфортного значения. Но тем не менее, на приемлемых частотах (единицы мегагерц) и при приемлемой разрядности (ну, скажем, 8), лично я периодически до сих пор использую такие счётчики (если условия того требуют, разумеется).

Полупрактический пример использования счётчиков

Уффф. Давайте теперь сделаем что-нибудь полезное из счётчиков. Скажу сразу, статья пишется в гостинице, поэтому не обещаю, что всё будет работать. Но по крайней мере, идея – точно верна, а остальное можно списать на «предлагается отладить самостоятельно». Итак, давайте-ка мы сделаем простенькие электронные часы. Вернее, ту часть часов, которая основана на тех функциях, которые мы уже знаем.

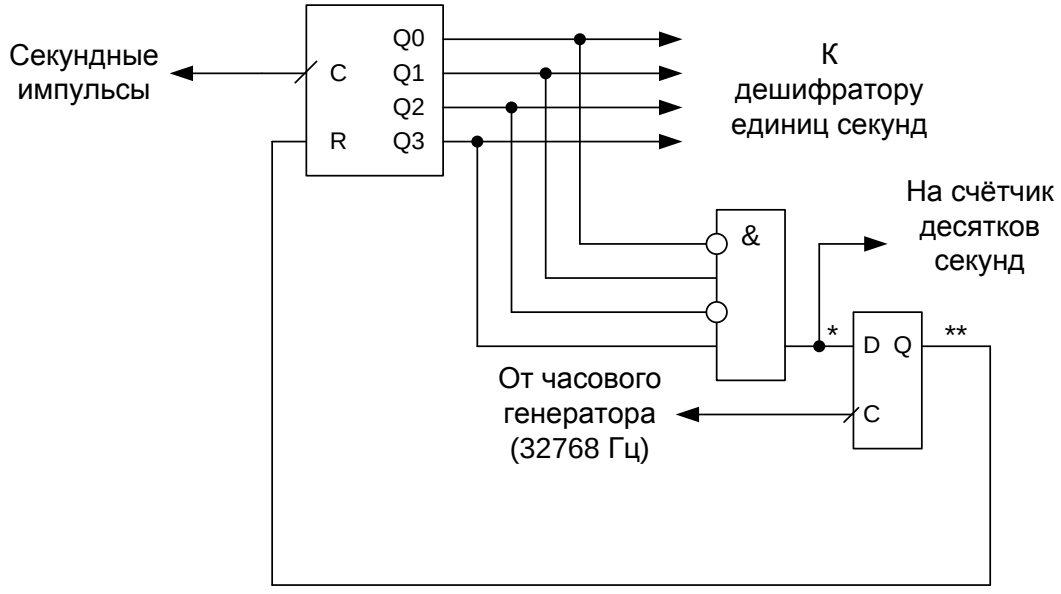
В любом магазине радиодеталей можно купить часовой кварц. Из него можно сделать генератор, который вырабатывает 32768 импульсов в секунду (то есть, частота

этого генератора равна 32768 герц, или 32.768 килогерц, кому как нравится, все записи означают одно и то же). Нам в часах необходимо получить для начала секундные импульсы. Давайте вспомним, что один Т-триггер делит частоту на 2. Следующий уже поделит на 4, дальше – на 8, на 16, на 32, на 64, на 128… И так потихоньку дойдём до 32768 (именно отсюда такая странная частота и взята, что она прекрасно получается последовательными делениями на 2). Итак. Блок формирования секундных импульсов делается на 15-разрядном счётчике, и будет выглядеть так:



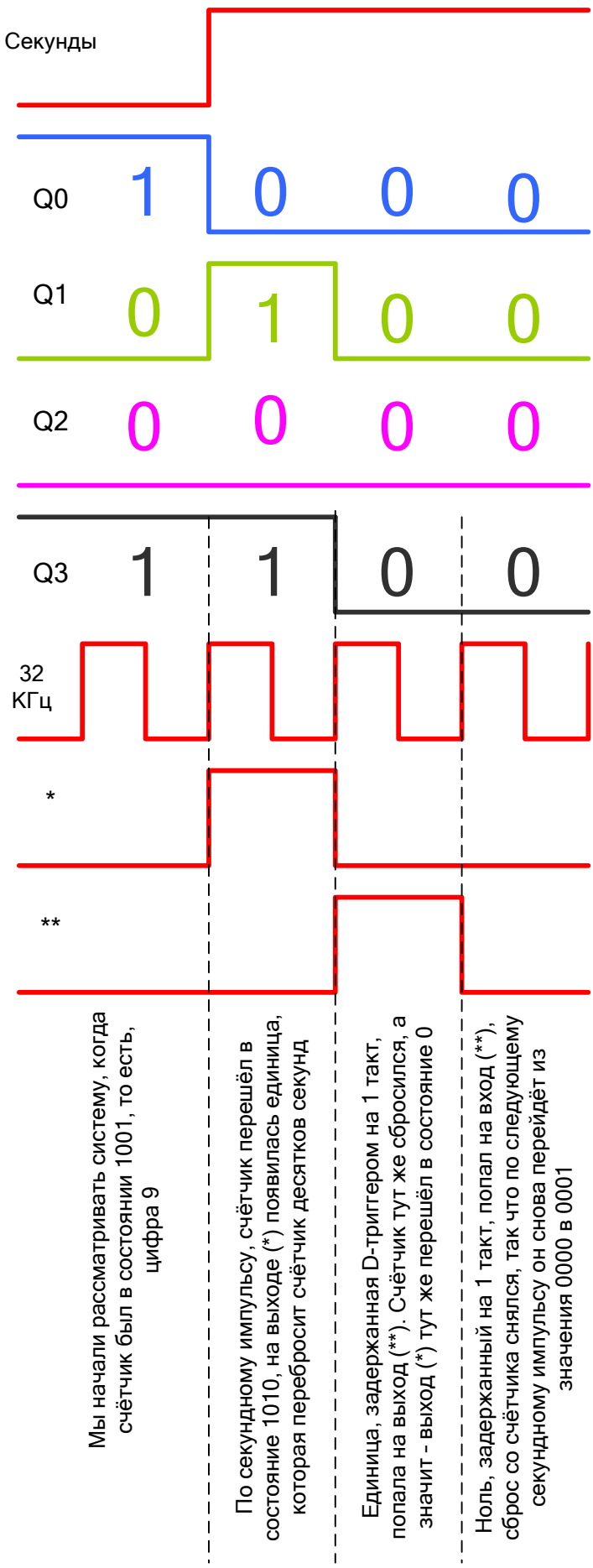
Итак. Зная, что каждый следующий выход счётчика перебрасывается вдвое реже, мы нашли тот выход, с которого можно взять импульсы с частотой 1 Гц, то есть, тикающие ровно 1 раз в секунду.

Ну вот. Теперь давайте потикаем секундами. Единицы секунд тикают от 0 до 9. Как только дотикали до 9, пора перебрасывать десятки секунд, а единицы – сбрасывать. Девятке соответсвует двоичный код 1001 (это я на калькуляторе подсмотрел), а после неё идёт число 1010 (это мы правила суммирования вспомнили). Ну и прекрасно, сделаем счётчик, который при достижении значения 1010 сначала посылает сигнал «Перебрасывайся» дальше, а затем – сбрасывает сам себя.



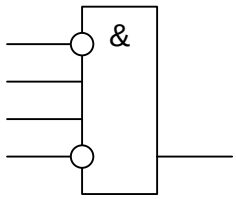
Как работает этот счётчик? Медленно и непринуждённо (1 раз в секунду) он проходит через состояния 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. После состояния 9 он перекидывается в состояние 10 (двоичное 1010). В это время открывается логический элемент 4И (с частично инверсными входами). На выходе (*) появляется логическая единица. По этому сигналу перебросится счётчик десятков секунд. А по следующему такту часового генератора (который тикает с бешеной скоростью), единица перескочит из точки (*) в точку (**). Из этой точки она перейдёт на вход сброса счётчика, и счётчик перейдёт в состояние 0000, то есть, ноль. В кино кадры меняются с частотой 25 герц, здесь – 32 тысячи герц. Так что этой микрозадержки никто и не заметит. А задержка нам нужна, чтобы триггеры в счётчике гарантировано сбросились.

Графически, этот процесс можно представить следующим образом:

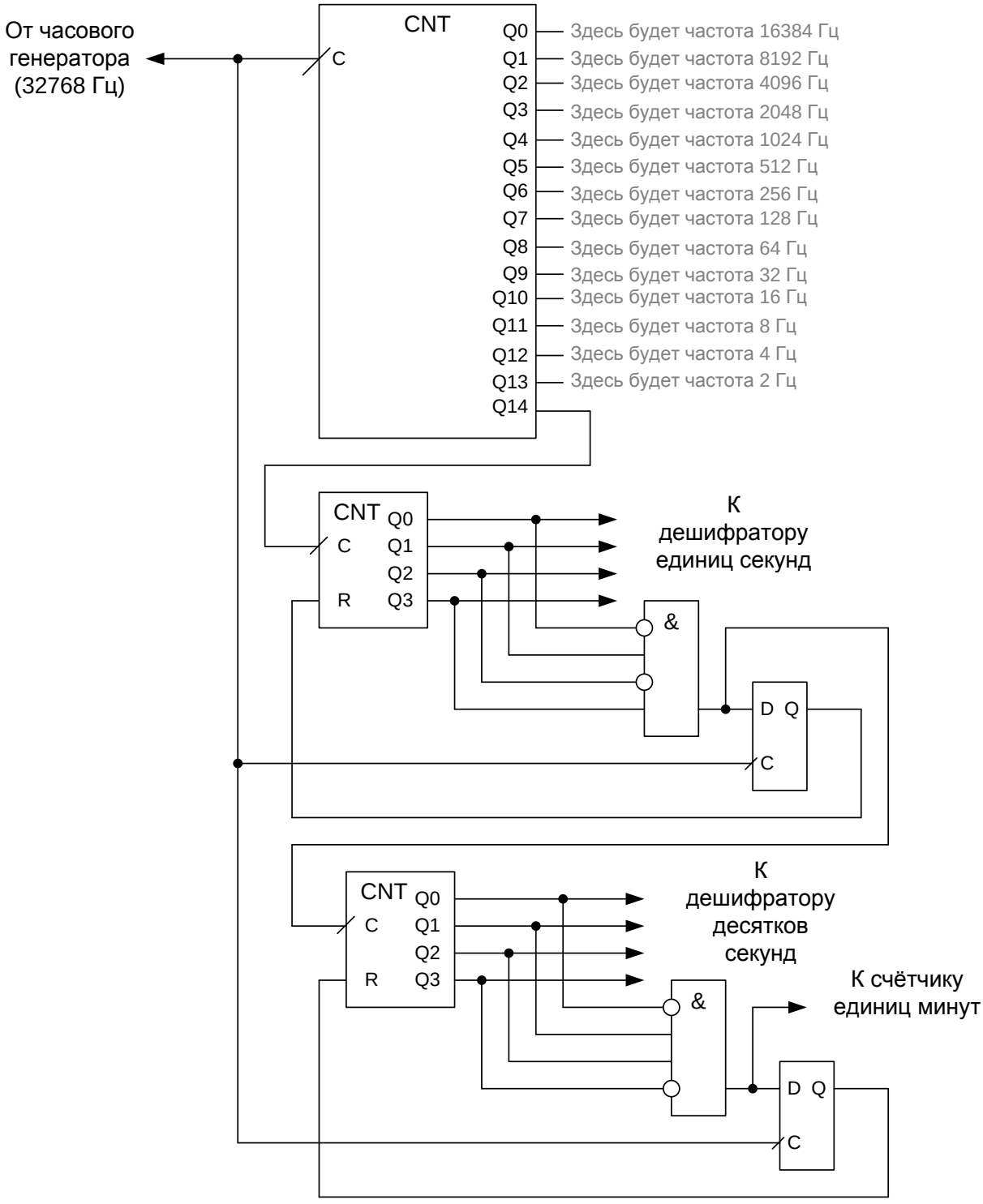

Секунды
Q0
1
0
0
0
Q1
0
1
0
0
Q2
0
0
0
0
Q3
1
1
0
0
32
КГц
*
**
Мы начали рассматривать систему, когда счётчик был в состоянии 1001, то есть, цифра 9
По секундному импульсу, счётчик перешёл в состояние 1010, на выходе (*) появилась единица, которая перебросит счётчик десятков секунд
Единица, задержанная D-триггером на 1 такт, попала на выход (**). Счётчик тут же сбросился, а значит - выход (*) тут же перешёл в состояние 0
Ноль, задержанный на 1 такт, попал на вход (**), сброс со счётчика снялся, так что по следующему секундному импульсу он снова перейдёт из значения 0000 в 0001

Что такое дешифратор – мы рассмотрим в одной из следующих статей. Он преобразует двоичный код в значения, отображаемые на 7-сегментных индикаторах.

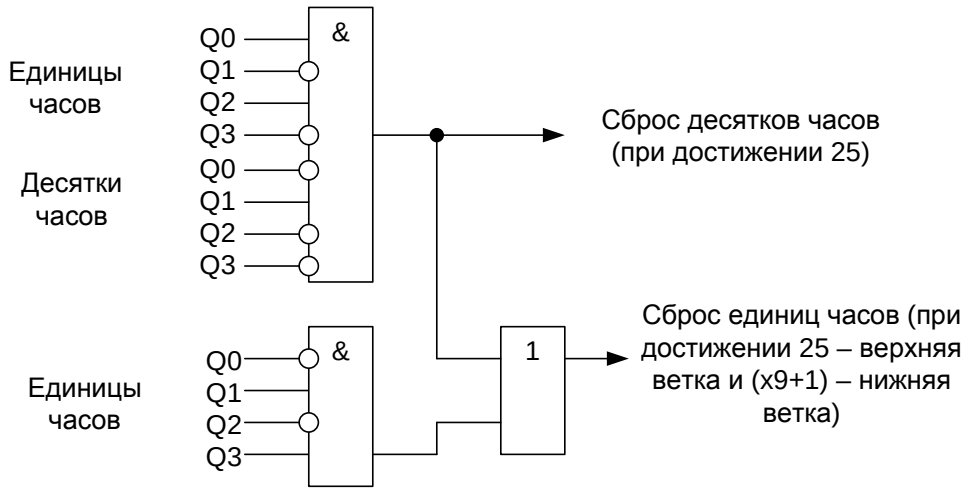
Для десятков секунд, надо сделать точно такую же схему, но они должны тикать не до 9, а до 5. То есть, счётчик надо сбросить, когда его значение будет равно 6. С помошью калькулятора выясняем, что в двоичном виде это будет 0110, а значит, единственное, что надо поменять – это тип элемента, формирующего сигнал (*). Он будет таким:



Ну, то есть, общая составленная схема часов станет такой:



Минуты тикают аналогично секундам, а у часов надо чуть иначе сделать схему сброса – они должны сбрасываться, когда часы равны 25 (0010 для десятков и 0101 для единиц). Соответственно, единицы часов могут сбрасываться как при достижении значения «десять» (1010), так и при достижении суммарного значения «25» (0010-0101). Но тут нам поможет элемент 2ИЛИ, который объединит эти сигналы, чтобы при появлении любого из них на вход единиц часов приходил сброс. В общем, что-то вроде этого:



. Так как наша задача – не любой ценой спаять часы, а просто понять, как счётчики применяются на практике, не будем загромождать статью дальнейшими подробностями.

Схема великовата? Вы на свои программы посмотрите, после того, как они уже завершены. После этого, станет понятно, что схема весьма даже и компактна. А главное – в ней нет ничего страшного, если знать, какой квадратик что означает, и рисовать всё последовательно… Вот примерно так всё и делается.

Ну, и чтобы закончить с триггерами и перейти к сложным логическим функциям (среди которых и дешифратор, о котором писалось немного выше), рассмотрим два вида регистров. Правда, сделаем это уже в следующий раз. Если ни один из самолётов, на которых мне предстоит лететь, не упадёт…