

# Языки описания схем

[mk.cs.msu.ru](http://mk.cs.msu.ru) → Лекционные курсы → Языки описания схем

## Блок К3

Кое-что ещё:  
схемная реализация приёмника UART

Лектор:  
**Подымов Владислав Васильевич**  
E-mail:  
**valdus@yandex.ru**

ВМК МГУ, 2024/2025, осенний семестр

# UART: как реализовать приёмник

Схема передатчика оказалась простой: всё, что она должна уметь —

- ▶ начинать передачу согласно заданному сигналу
- ▶ сохранять значение на входе в начале передачи
- ▶ поразрядно выдавать сообщение с заданным числом тактов на разряд

Приём сообщения устроен более сложно:

- ▶ частоты и фазы тактовых сигналов передатчика и приёмника независимы и **никак** не могут быть синхронизированы внутри приёмника
- ▶ сообщение должно корректно приниматься с учётом всех **погрешностей** и независимо от сочетания частот и фаз

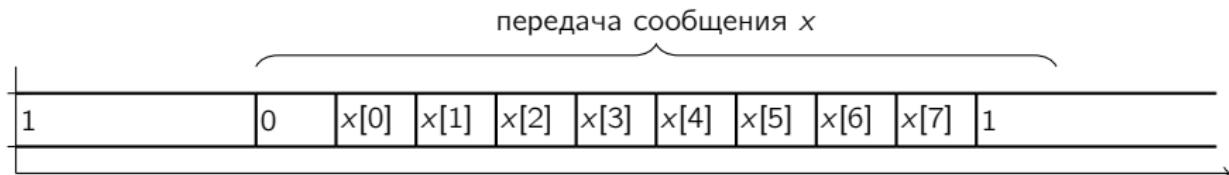
# UART: как реализовать приёмник

Рассмотрим простой вариант приёмника:

- ▶ Порты:
  - ▶  $clk$ ,  $rst$  — тактовый сигнал и асинхронный сброс
  - ▶ выход  $x$  ширины 8
  - ▶ выход  $busy$  ширины 1
  - ▶ вход  $rx$  ширины 1
    - ▶ **Receive Data**  $\xrightarrow{x?}$   $RxD \rightarrow Rx$   
Это устоявшееся обозначение провода приёма данных по UART
- ▶  $busy = 1 \Leftrightarrow$  сообщение принимается
- ▶ После приёма сообщения и до начала следующего приёма в  $x$  выводится принятое число

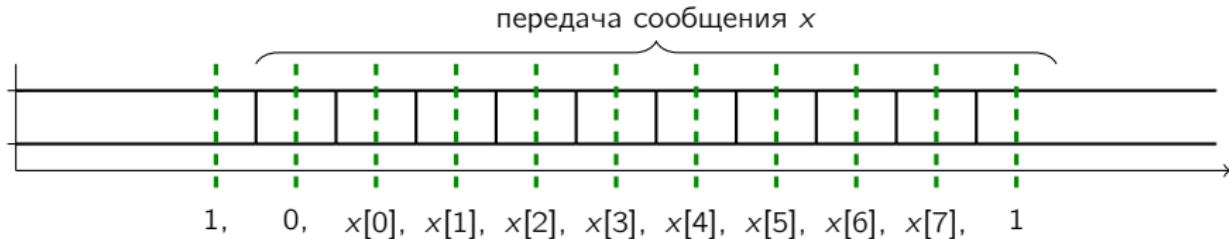
Попробуем придумать схему такого приёмника —  
может быть, не самую лучшую, но какую-нибудь попроще

# UART: как реализовать приёмник



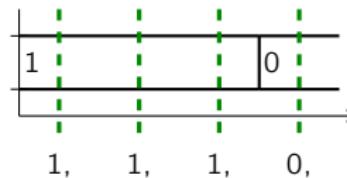
Приёмник, как и любая синхронная схема, может читать значения со входов и изменять состояние **только** в моменты выбранных фронтов тактового сигнала

Как следствие, в приёмнике должно выполняться подходящее **семплирование (дискретизация)** входа  $rx$ : замена непрерывного сигнала на дискретную последовательность цифровых значений этого сигнала в выбранные моменты времени



# UART: как реализовать приёмник

## Начало приёма сообщения



Пусть  $\mu$  — частота приёмника, и  $\tau = \frac{1}{\mu}$

Если приёмник считывал со входа тишину (1), и затем по очередному фронту тактового сигнала в момент  $t$  считал значение 0, то он может быть уверен в том, что

- ▶ передача сообщения началась, и
- ▶ момент начала передачи сообщения лежит в интервале  $[t - \tau, t]$

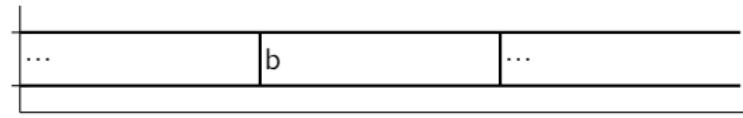
Назовём фронт, по которому считано такое значение 0, **нолевым**, и начнём с него отсчёт фронтов

Основная задача приёмника после нолевого фронта — «аккуратно» посчитать фронты для приёма всех битов данных

# UART: как реализовать приёмник

## Приём очередного бита

Рассмотрим идеальную пересылку одного бита сообщения ( $b$ ):



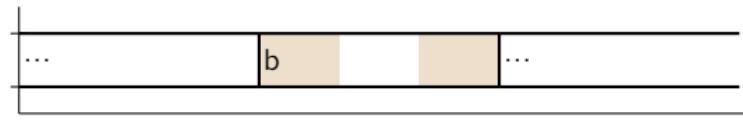
Предположим, что известно точное время  $t$  начала пересылки сообщения

Тогда можно точно вычислить идеальное время начала ( $\ell$ ) и конца ( $r$ ) пересылки бита  $b$ , отталкиваясь от  $t$  и частоты протокола

В реальности длительность пересылки каждого бита колеблется в пределах некоторой **погрешности**, и погрешность может накапливаться с каждым следующим битом

# UART: как реализовать приёмник

## Приём очередного бита



Из-за погрешностей в интервале пересылки каждого бита появляются «серые зоны»: интервалы  $[\ell, \ell + \Delta]$  и  $[r - \Delta, r]$ , где  $\Delta$  — наибольшая возможная накопленная погрешность времени пересылки бита

Попытка считать значение текущего бита из серой зоны может привести к тому, что из-за накопленной погрешности фактически считается значение предыдущего или следующего бита

Семплирование сообщения приёмником должно быть устроено так, чтобы значение каждого бита читалось в *идеальном* интервале, но **вне** серой зоны

# UART: как реализовать приёмник

## Приём очередного бита

Положим для примера, что частота приёмника в три раза выше частоты протокола

Тогда в каждом *идеальном* интервале пересылки каждого бита содержится **ровно три** фронта с такими граничными случаями их расположения (в зависимости от точного времени начала передачи сообщения):

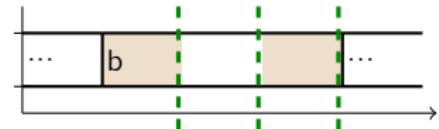
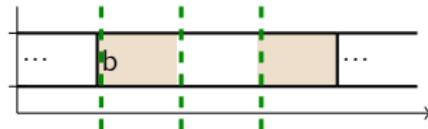


Если ширина серой зоны с каждой стороны меньше трети периода протокола, то средний фронт обязательно находится вне серой зоны

Этот средний фронт для бита  $x[i]$  имеет номер  $3 \cdot i + 4$

# UART: как реализовать приёмник

## Приём очередного бита



*Немного обобщим:* если частота приёмника  $\mu$  в  $N$  раз больше частоты протокола  $\nu$ , то фронт с номером  $N \cdot (i + 1) + \lfloor \frac{N}{2} \rfloor$  —

- ▶ либо самый близкий к середине интервала передачи  $x[i]$ , если  $N$  нечётно,
- ▶ либо один из двух фронтов, самых близких к середине этого интервала, если  $N$  чётно

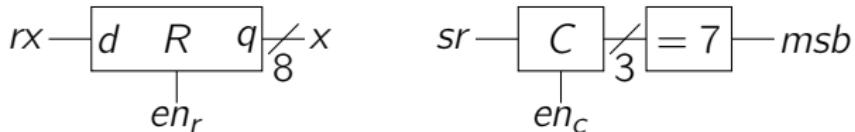
*Окончательно обобщим:* если  $\mu$  не делится нацело на  $\nu$ , то формула для номера среднего фронта бита  $x[i]$  устроена сложнее, но если частота  $\mu$  достаточно высока, то можно считать, что она *приблизительно* в  $\lfloor \frac{\mu}{\nu} \rfloor$  раз больше частоты  $\nu$ , и отнести *реальное* расхождение в погрешность

*Итог:* бит  $x[i]$  можно семплировать по фронту с номером

$$\lfloor \frac{\mu}{\nu} \rfloor \cdot (i + 1) + \lfloor \frac{\mu}{2 \cdot \nu} \rfloor$$

# UART: как реализовать приёмник

## Операционный автомат



$R$  — синхронный **сдвиговый регистр** ширины 8 с включением:

- ▶ на выход выдаётся сохранённое значение
- ▶  $en_r = 0 \Rightarrow$  сохранённое значение не изменяется
- ▶  $en_r = 1 \Rightarrow$  сохранённое значение сдвигается на один разряд в сторону младшего бита, и в старший бит записывается текущее значение  $d$

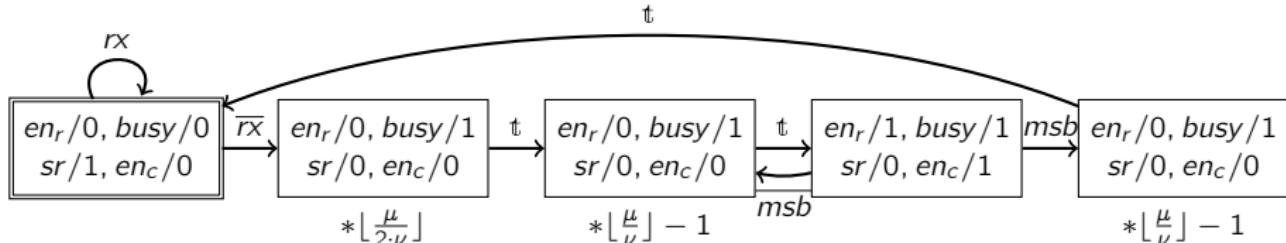
$C$  — синхронный **счётчик** ширины 3, отсчитывающий число принятых разрядов данных:

- ▶ на выход выдаётся сохранённое значение
- ▶  $sr = 1 \Rightarrow$  сохраняется значение 0
- ▶  $sr = 0, en_c = 1 \Rightarrow$  сохранённое значение увеличивается на 1

$\ll= 7\gg$  — комбинационная схема, выход которой равен 1  $\Leftrightarrow$  вход равен 7

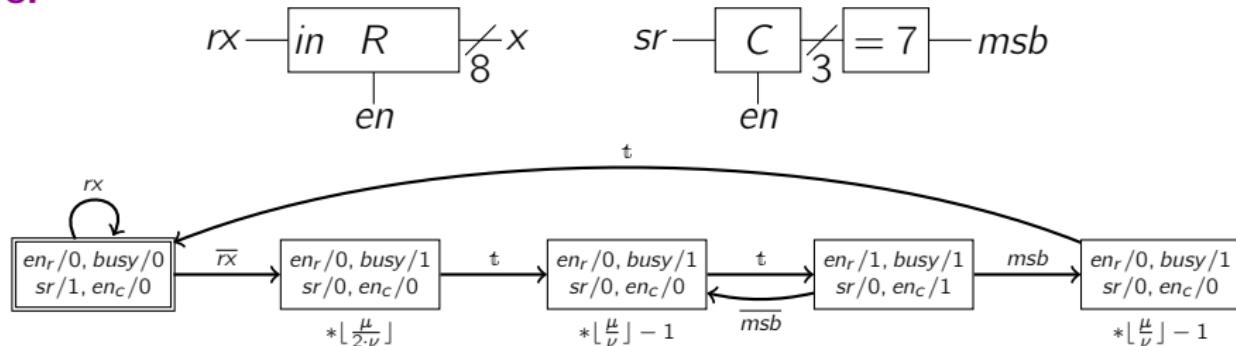
# UART: как реализовать приёмник

## Управляющий автомат



# UART: как реализовать приёмник

## Итог



Эту схему можно легко адаптировать и к другим вариантам протокола:

- ▶ Частота протокола: учтена в числе  $\frac{\mu}{\nu}$
- ▶ Ширина пересылаемого значения: поправить ширину  $R$  и  $C$  и число в подсхеме «=  $M$ »
- ▶ Порядок пересылки разрядов: подходящим образом соединить выход  $R$  с портом  $x$
- ▶ Проверка корректности передачи: добавить подсхему суммирования принятых битов, увеличить длительность правого состояния управляющего автомата