1 Однострочный LCD-дисплей с параллельным интерфейсом

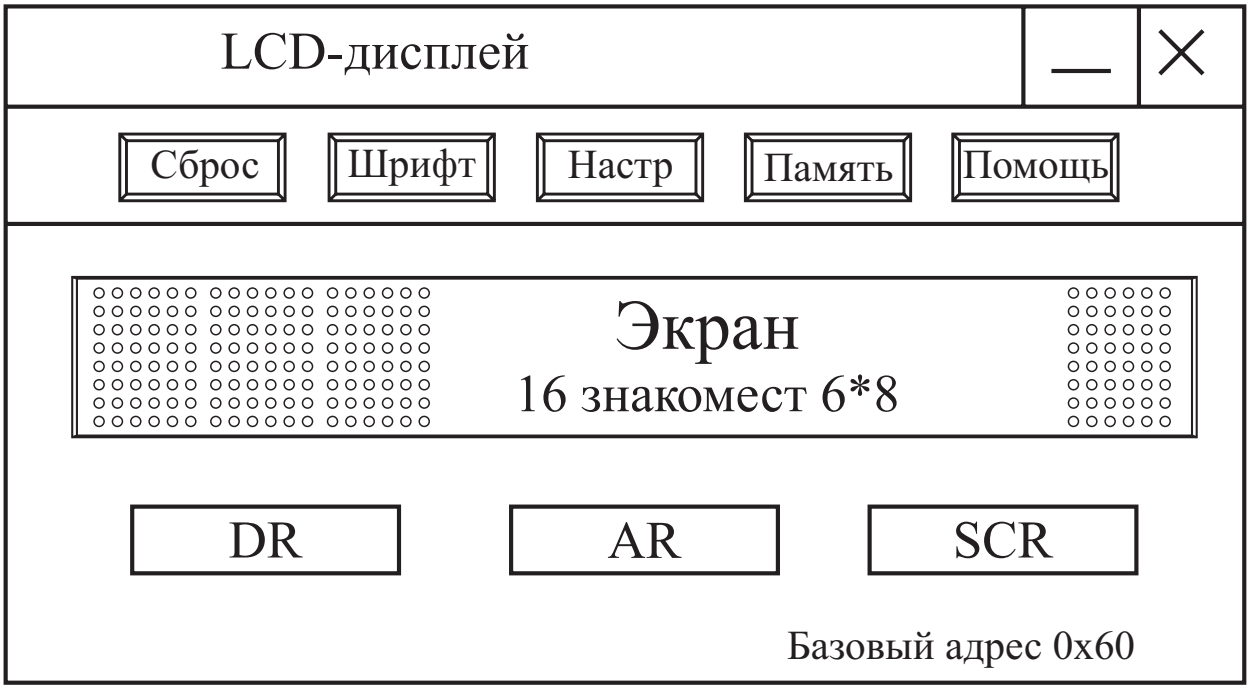


Рисунок 1. Окно обозревателя

Основные параметры:

- Количество строк — 1
- Количество знакомест в строке — 16
- Размер знакоместа — 6×8 пикселов
- Размер экрана — $16 \times 6 \times 8 = \mathbf{96} \times \mathbf{8} = 768$ пикселов
- Видеопамять — 16 байт
- Память экрана — $16 \times 6 = 96$ байт
- Память шрифта при использовании стандартной кодировки ASCII – 256 символов × 6 байт = 1536 байт

Основные требования:

⁕ По включению (и «Сброс» ?) в память шрифта загружается шрифт *По умолчанию* или выбирается один из нескольких предлагаемых текстовых файлов шрифтов, который будет загружен в память шрифта.

В видеопамять загружается 16 пробелов или курсор (_) и 15 пробелов.

⁕ Перезагрузка видеопамяти может осуществляться в одном из двух режимов:

a) нескольких байт в произвольном порядке;

b) только последовательно все 16 символов, начиная с крайнего левого в строке.

⁕ Любая перезагрузка видеопамяти должна сопровождаться полной (?) перезагрузкой памяти экрана.

⁕ Возможны два режима вывода:

1) «Бегущая строка», когда при выводе 17-го и далее символов осуществляется левый сдвиг и левые символы на экране (и в видеопамяти) теряются;
2) «Фиксированный размер», когда вывод 17-го символа блокируется и формируется флаг ошибки. В этом случае требуется очистка экрана (памяти) или переключение на режим «Бегущая строка».

Регистр управления/состояния

CSR[0]: E – включение [0]

CSR[1]: M – доступ в память [0]

$$M = \begin{cases} 0 - & \text{видеопамять} \\ 1 - & \text{память шрифта (здесь по каждому адресу отправлять 6 байт)} \end{cases}$$

CSR[2]: A – автоиндексация адреса [0]

CSR[3]: DR – направление сдвига

$$DR = \begin{cases} 0 - & \text{инкремент адреса (вправо)} \\ 1 - & \text{декремент адреса (влево)} \end{cases}$$

CSR[4]: SF – формат вывода строки [0]

$$SF = \begin{cases} 0 - & \text{Фиксированная строка} \\ 1 - & \text{Бегущая строка} \end{cases}$$

CSR[5]: R – тип сдвига [0]

$$R = \begin{cases} 0 - & \text{Обычный} \\ 1 - & \text{Циклический} \end{cases}$$

CSR[6]: Reset – сброс [0]

CSR[7]: Err – флаг ошибки [0]

Таким образом, разряды CSR[6:0] относятся к управляющим, а CSR[7] определяет состояние (ошибки) контроллера.

Контроллер должен реализовать следующие процедуры:

- Сброс
- Загрузка шрифта из выбранного текстового файла[1] (?)
- Выбор цветов фона и единичного пикселя[2]
- Редактирование видеопамяти[2]

Окно обозревателя LCD-дисплея

– Строка заголовка: «LCD-дисплей»

[1] В режиме настройки

[2] Программно на fN8

– Строка меню (кнопки)

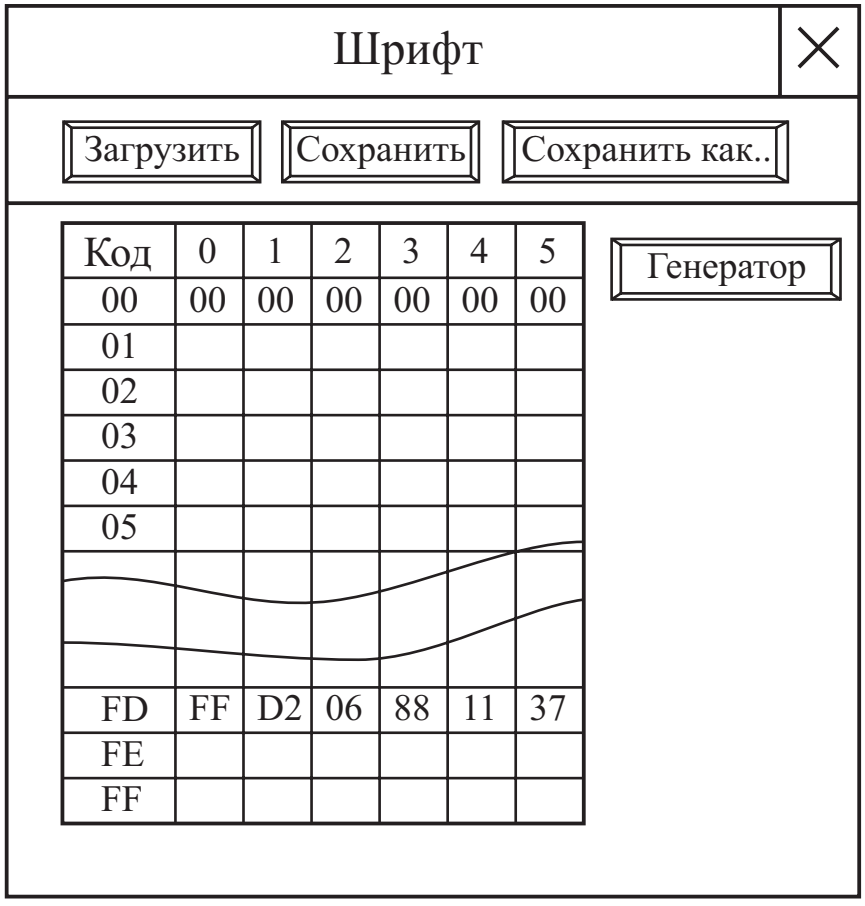
  - *Сброс*
  - *Шрифт* – открывает окно «Шрифт»
  - *Настройки* – открывает окно «Настройки»
  - *Память* – открывает окно «Видеопамять» – 16 ASCII-кодов (HEX)
  - *Помощь* – открывает окно «Помощь»

– Экран (16 знакомест по 6×8 пикселов)

– Три 8-разрядных (двоичных) регистра: (0)DR, (1)AR, (2)SCR

– Надпись «Базовый адрес – 0x[6]0»

Окно *Шрифт*



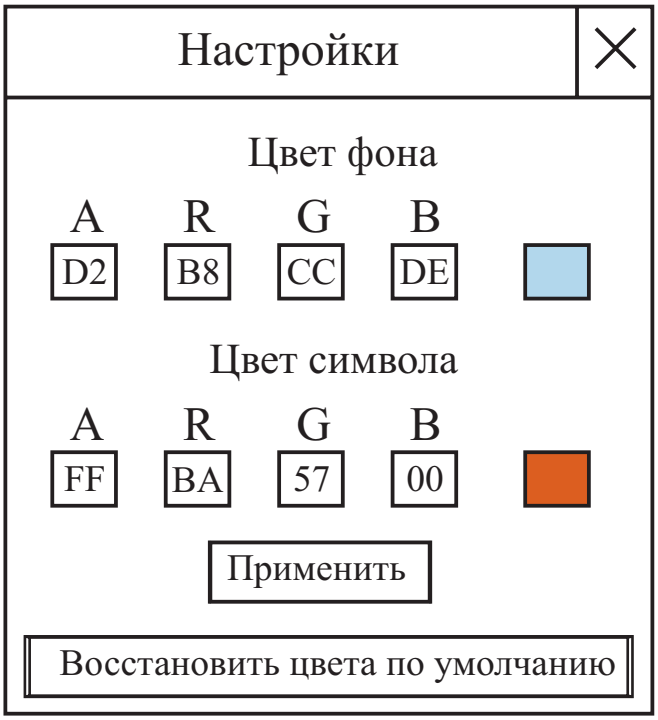
– Строка заголовка: «Шрифт»

– Строка меню (кнопки)

  - *Загрузить* – позволяет загрузить содержимое памяти шрифта из текстового файла
  - *Сохранить* – сохраняет текущее содержимое памяти шрифта под текущим именем
  - *Сохранить как* – допускает изменение имени при сохранении
  - *Генератор шрифта* – открывает программу генерации шрифта

– Дамп памяти шрифта – 256 строк по 6 двухразрядных шестнадцатеричных чисел. Строки пронумерованы от 00 до FF. Допускается ручное редактирование произвольных элементов строк и программное редактирование (fN8) только целых строк.

Окно *Настройки* позволяет выбрать цвета символов на экране и фона.



– Строка заголовка: «Настройки»
– Надпись «Цвет фона», 4 двухразрядных шестнадцатеричных числа, определяющих цвет (ARGB) и прямоугольник, закрашенный выбранным цветом
– То же для цвета символа.
– Кнопки *Применить* и *Восстановить цвета по умолчанию*

Генератор шрифта

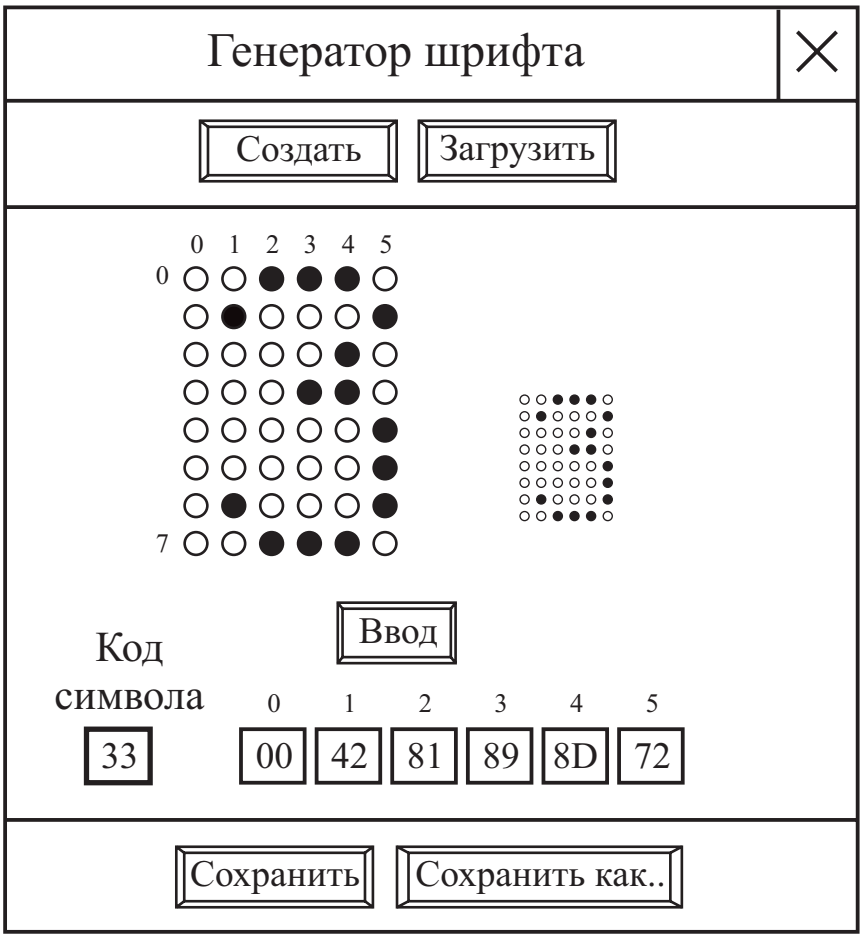
Генератор шрифта – автономная программа,позволяющая сгенерировать или отредактировать ASCII-коды символов для матрицы пикселей.

По кнопке *Создать* генерируется текстовый файл из 256 шестибайтовых «строк», представляющих собой изображение символа в матрице 6×8. При создании все байты всех строк равны 00.

Последовательно нажимая на кнопку *Ввод* выводим в поле *Код символа* ASCII-код очередного символа (номер строки текстового файла). В растре 6×8, представленном в увеличенном масштабе, щелчком мыши инвертируется произвольный элемент (пиксел). Одновременно получившееся изображение символа выводится в знакоместо в масштабе LCD-дисплея и отображается в форме шести байт (HEX), соответствующих значениям столбцов растра.

При нажатии *Ввод* текущий шестибайтовый код фиксируется в строке текстового фай-

ла, номер которой отображается в поле *Код символа* и в это поле выводится следующий номер. Допускается редактировать произвольные символы не подряд, а выборочно – для этого достаточно ввести ASCII-код символа в поле *Код символа*, отредактировать растр и нажать *Ввод*.



2 Контроллер последовательного обмена

Контроллер последовательного обмена (КПсО) реализован как полудуплексный[3]. Структурная схема контроллера и его связи с системной шиной и ВУ показаны на рис. 2

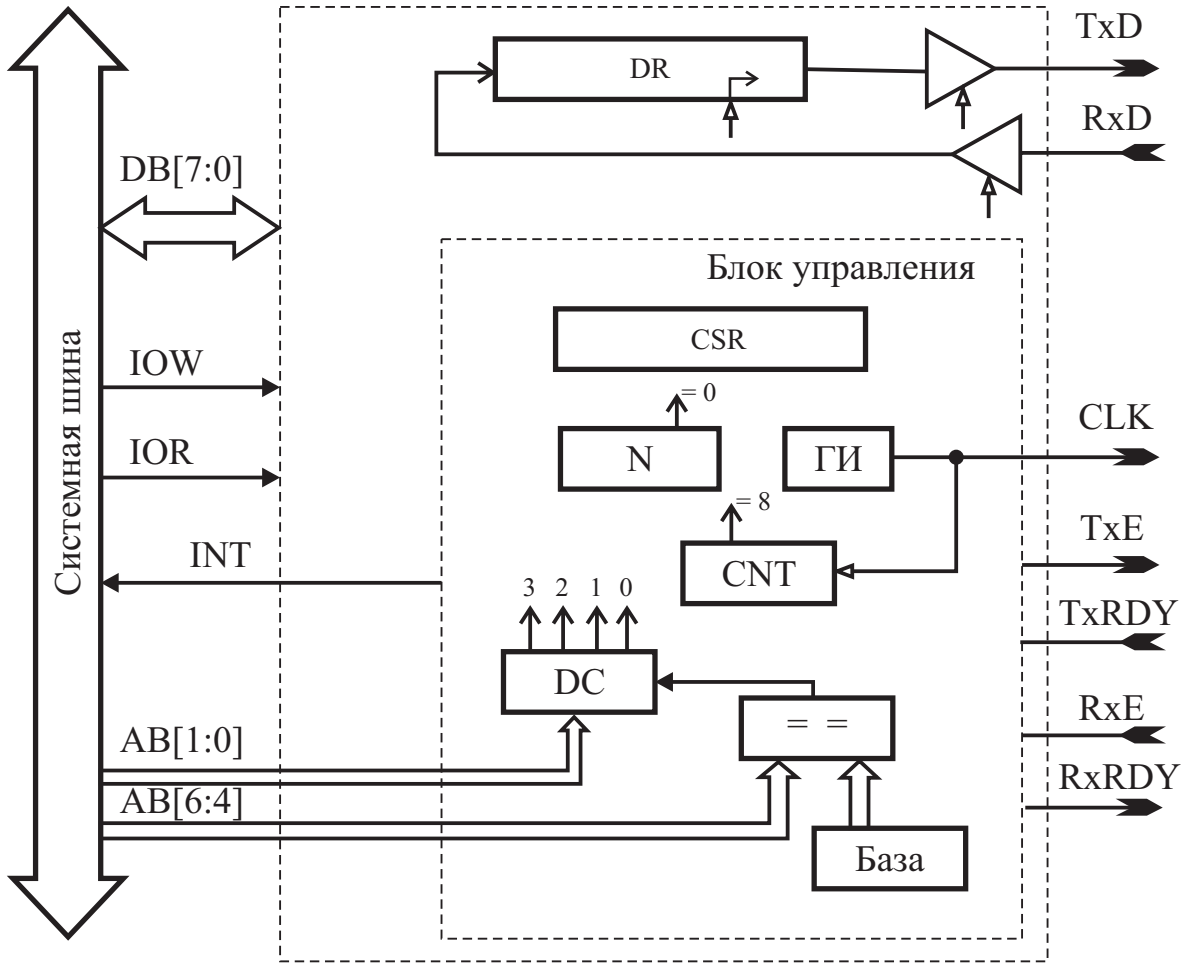


Рисунок 2. Структурная схема контроллера

Сигналы последовательного интерфейса «контроллер – ВУ»:
(В скобках – источник сигнала)

CLK – тактовый сигнал (контроллер)

TxD – данные передатчика (контроллер)

RxD – данные приёмника (ВУ)

TxE – разрешение передачи (контроллер)

RxE – разрешение приёма (ВУ)

TxRDY – готовность принять передачу (ВУ)

RxRDY – готовность начать приём (контроллер)

Временные диаграммы передачи и приёма приведены на рис. 3

[3]Режим, при котором передача ведётся в обоих направлениях, но с разделением по времени называют *полудуплексным*. В каждый момент времени передача ведётся только в одном направлении.

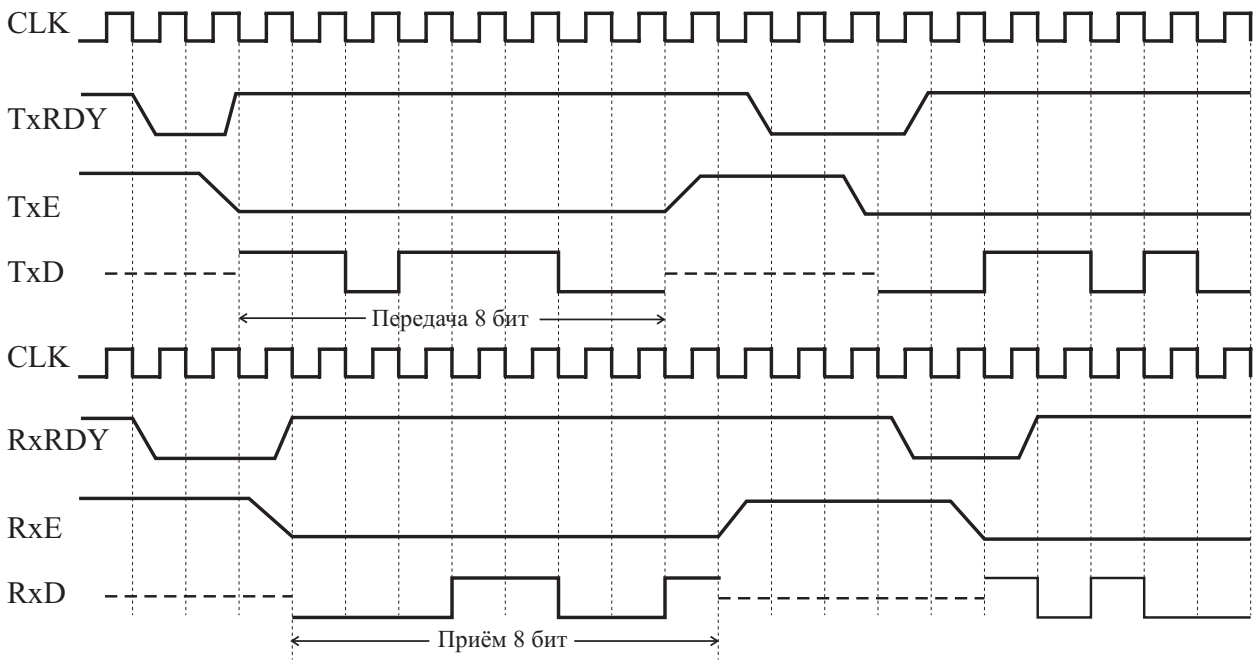


Рисунок 3. Временны́е диаграммы передачи и приёма

Контроллер содержит два програмно-доступных регистра (конечные точки): регистр данных приёмопередатчика DR и регистр управления/состояния CSR.

7	6	5	4	3	2	1	0
F	FR	FT	REI	TEI	S1	S0	E

Рисунок 4. Формат регистра CSR

- E – «1» - включение контроллера
- S[1:0] – частота генератора импульсов

  00 – OSC (частота задающего генератора)

  01 – OSC/4

  10 – OSC/16

  11 – OSC/64

- TEI – разрешить прерывание от передатчика
- REI – разрешить прерывание от приёмника
- FT – флаг завершения передачи байта
- FR – флаг завершения приёма байта
- F – тип транзакции («0» – передача, «1» – приём)

Разряды CSR[7:5] устанавливаются только контроллером. Независимо от значений DB[7:5] записываемого в CSR байта значения этих разрядов не меняется.

Блок управления контроллером (на рис. 2), помимо CSR, содержит генератор импульсов ГИ, частота которого задаётся полем CSR[2:1], счётчик битов CNT, счётчик байтов N, схему селекции конечных точек (регистр базового адреса, схема сравнения и дешифратор младших разрядов адреса).

Обмен данными между процессором fN8 и внешним устройством (ВУ) с последовательным интерфейсом реализуется с помощью **транзакций**. Различают два типа транзакций: *передача* (от процессора к ВУ) и *приём* (от ВУ в процессор).

Каждая транзакция начинается с передачи *управляющего байта транзакции* (УБТ), формат которого показан ниже:

F	N	A

где

F = УБТ[7] – тип транзакции («0» – передача, «1» – приём)

N = УБТ[6:2] – число передаваемых/принимаемых байт

A = УБТ[1:0] – адрес конечной точки ВУ.

Когда процессор помещает УБТ в DR контроллера, параметры F и N копируются в CSR[7] и счётчик байтов N соответственно, после чего УБТ передаётся в ВУ. Далее контроллер реализует N циклов передачи или приёма байтов (определяется значением CSR[7]), декрементируя после каждого цикла значение N. При N = 0 транзакция завершается и следующий байт, записанный процессором в DR воспринимается контроллером как УБТ и начало новой транзакции.

Алгоритм работы контроллера показан на рис. 5. Отметим, что в начале очередной транзакции флаги, управляющие сигналы и параметры имеют следующие значения:

$FT = 1, FR = 1$ – завершены передача и приём байтов;

$TxE = 1, RxE = 1, TxRDY = 1, RxRDY = 1$ – все управляющие сигналы в неактивном состоянии;

$F = 0$ $N = 0$ – включён режим передачи и нет передаваемых байтов. Эти параметры будут изменены при получении УБТ.

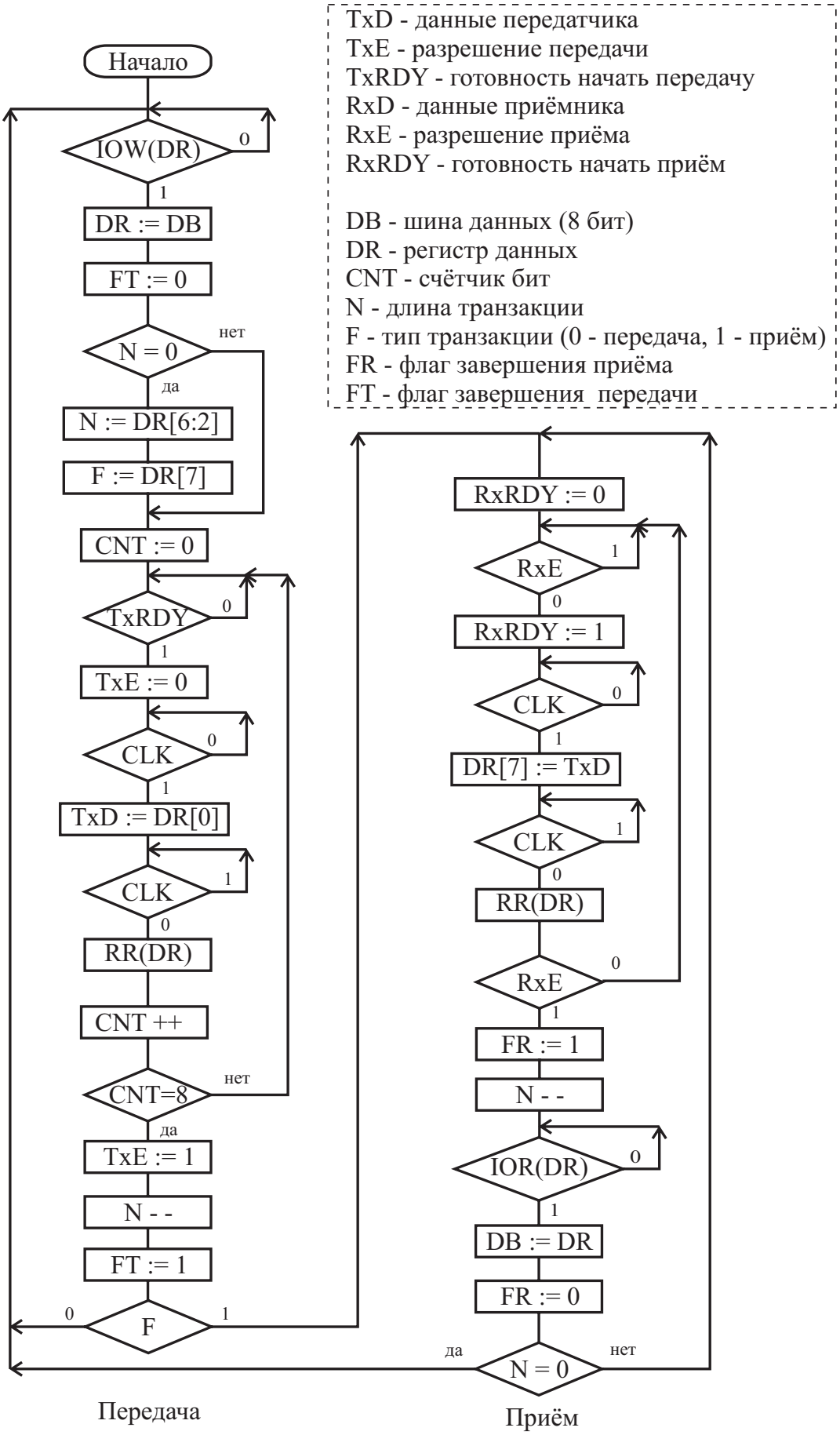


Рисунок 5. ГСА приёмопередатчика

3 Модуль сопряжения последовательного интерфейса с ВУ

В составе ВУ, которое обменивается с процессором по последовательному каналу, необходимо предусмотреть модуль сопряжения с контроллером последовательного обмена (см. раздел 2). Структура такого модуля представлена на рис. 6.

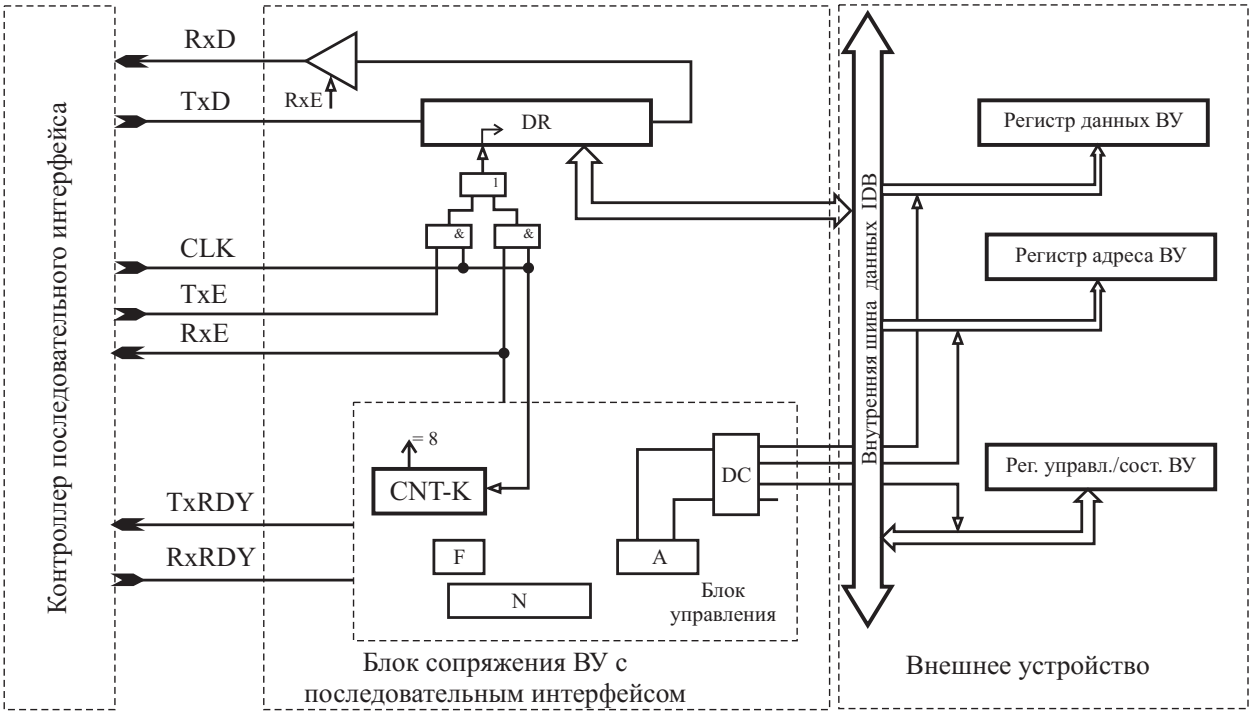


Рисунок 6. Модуль сопряжения

Модуль содержит сдвиговый регистр данных DR, через который осуществляется приём или передача последовательного кода и блок управления. Регистр DR имеет связь по внутренней 8-разрядной шине с конечными точками (программно-доступными регистрами) ВУ. Адрес конечной точки определяется полем A управляющего байта транзакции (см. стр. 9), а направление передачи данных – битом F.

Алгоритм последовательного обмена (со стороны модуля сопряжения) представлен на рис. 7. Обмен начинается с передачи процессором в ВУ управляющего байта транзакции, который помещается в служебный регистр блока управления. В рамках одной транзакции процессор связывается только с одной конечной точкой, причём транзакция передачи[4] может включать до 32 передаваемых байт, а транзакция приёма – только один байт (состояние ВУ)[5].

[4]Со стороны процессора

[5]Здесь речь идёт конкретно о ВУ – LCD-дисплей.

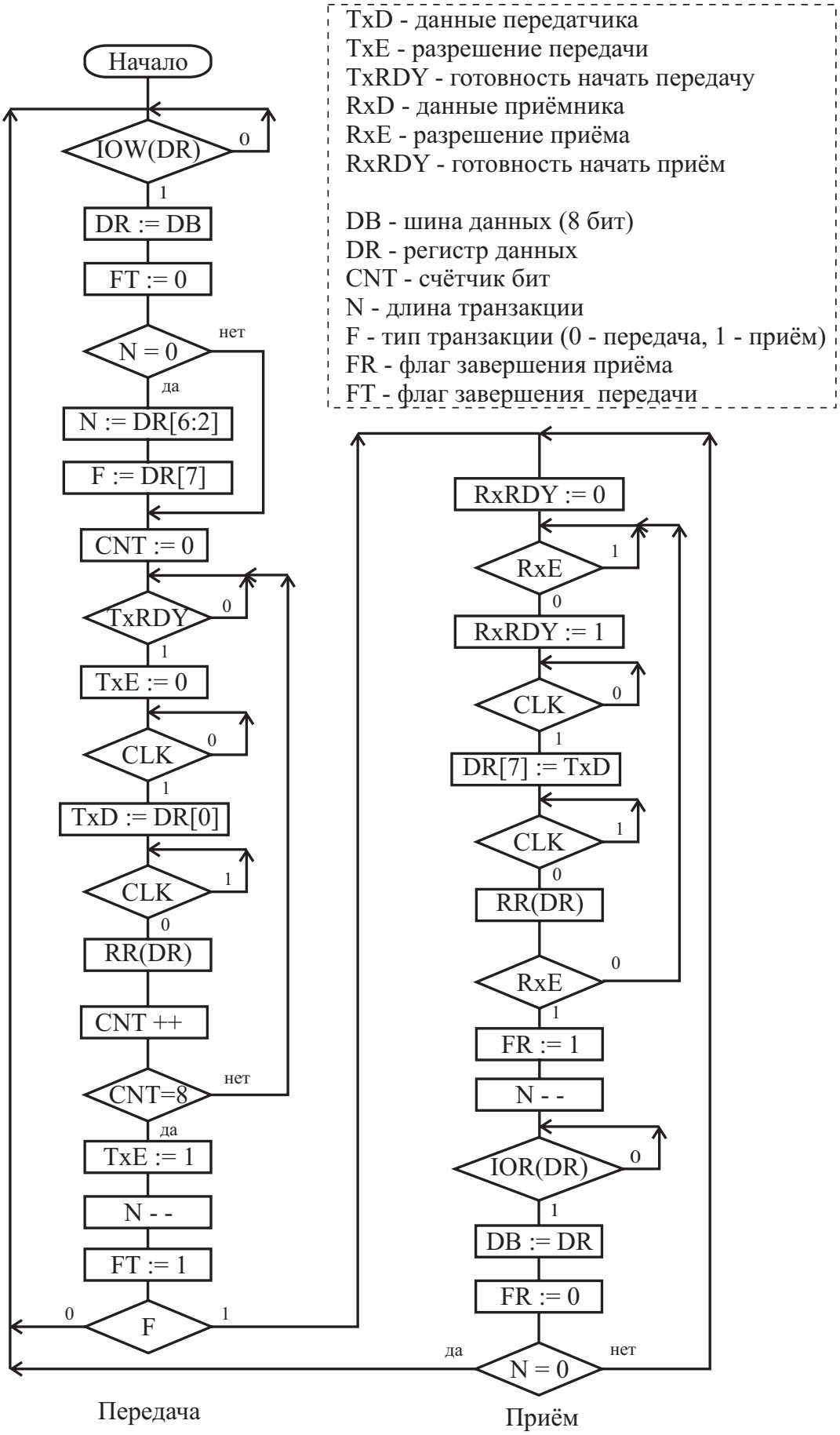


Рисунок 7. ГСА блока сопряжения