МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования

Национальный исследовательский Томский политехнический университет

Институт кибернетики

Направление 230100 «Информатика и вычислительная техника»

Кафедра вычислительной техники

Курсовой проект

**«БИНАРНЫЕ ЧАСЫ С БУДИЛЬНИКОМ»**

по дисциплине «Схемотехника ЭВМ ч.2»

Вариант №33.1

Выполнили:

студент гр. 8В01 \_\_\_\_\_\_\_\_\_\_\_\_\_ Дудченко А.В.

\_\_\_.\_\_\_.\_\_\_\_\_\_

студент гр. 8В01 \_\_\_\_\_\_\_\_\_\_\_\_\_ Андреев С.А.

\_\_\_.\_\_\_.\_\_\_\_\_\_

Проверил:

доцент кафедры ВТ \_\_\_\_\_\_\_\_\_\_\_\_\_ Мальчуков А.Н.

\_\_\_.\_\_\_.\_\_\_\_\_\_

# Содержание

[Содержание 2](#_Toc373245233)

[Цель. 4](#_Toc373245234)

[Задание. 5](#_Toc373245235)

[Словесное описание устройства и его структурно-функциональная схема. 6](#_Toc373245236)

[Функциональная схема и ее описание (включая описание автоматов в виде графов). 8](#_Toc373245237)

[Блок Main 8](#_Toc373245238)

[Блок Receiver 12](#_Toc373245239)

[Блок TXD\_neg\_edge 14](#_Toc373245240)

[Блок LCD\_Main 16](#_Toc373245241)

[Блок LCD\_Command 19](#_Toc373245242)

[Блок LCD\_DRAM 21](#_Toc373245243)

[Блок LCD\_Loader 23](#_Toc373245244)

[Структурная схема 24](#_Toc373245245)

[Заключение. 26](#_Toc373245246)

[Список литературы. 27](#_Toc373245247)

[Приложения (руководство пользователя с фотографиями устройства и исходные коды модулей на языке описания аппаратуры). 28](#_Toc373245248)

[Руководство пользователя 28](#_Toc373245249)

[Общие сведения об устройстве 28](#_Toc373245250)

[Режим «Демонстрация времени» 29](#_Toc373245251)

[Режим «Установка времени» 29](#_Toc373245252)

[Режим «Установка будильника» 30](#_Toc373245253)

[Включение/выключение будильника 3 31](#_Toc373245254)

[Исходные коды модулей на языке Verilog 32](#_Toc373245255)

[Main 32](#_Toc373245256)

[Receiver 42](#_Toc373245257)

[TXD\_neg\_edge 44](#_Toc373245258)

[LCD\_Main 45](#_Toc373245259)

[LCD\_Comman 51](#_Toc373245260)

[LCD\_DRAM 52](#_Toc373245261)

[LCD\_Loader 54](#_Toc373245262)

# Цель.

Целью данной работы является создание устройства часы с будильником реализуемые на макете SDK 6.1.

Функции устройства должны в себя включать:

* установка текущего времени;
* установка времени срабатывания будильника;
* включение/отключение будильника;
* индикация срабатывание будильника.

# Задание.

В рамках курсового проекта необходимо

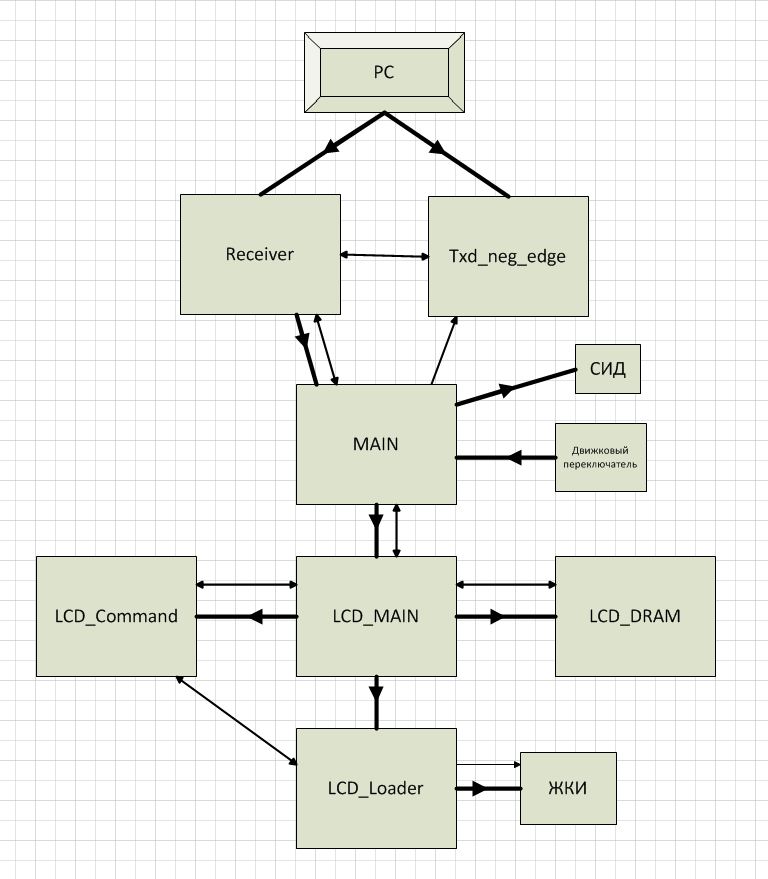
* Составить и согласовать техническое задание на разработку.
* Разработать структурно-функциональную схемы устройства. Спроектировать конфигурацию для ПЛИС и проверить работоспособность на макете SDK-6.1.
* Составить техническую документацию на разработку (пояснительная записка).
* Составить руководство пользователя.

Устройство должно удовлетворять следующим условиям:

1. отображение информации на ЖКИ макета SDK-6.1;
2. русский язык интерфейса;
3. разработанная прошивка ПЛИС устройства для макета SDK-6.1 должна позволять проверить корректность работы на произвольных данных.
4. Язык описания аппаратуры Verilog.

# Словесное описание устройства и его структурно-функциональная схема.

Структурно-функциональную схему устройства можно описать с помощью блоков: Main, Receiver, TXD\_neg\_edge, LCD\_Main, LCD\_Comman, LCD\_DRAM, LCD\_Loader.



Рисунок

На рисунке. 1 представлена структурно-функциональная схема устройства.

Прием данных начинается с блока main. Он отправляет блоку **Receiver** управляющий сигнал о переходе в режим считывания, после чего **Receiver** отправляет управляющий сигнал активации блока **Txd\_neg\_edge**. Блок **Txd\_neg\_edge** определяет первый задний фронт сигнала с **PC**, то есть начало старт бита и отсылает управляющий сигнал блоку **Receiver**. Блок **Receiver** последовательно получает 8 бит с **PC** и отправляет их в блок **Main** и на **СИД**. Чтобы активировать режим приема данных, необходимо подать управляющий сигнал блоку **Main** с помощью блока движковых переключателей.

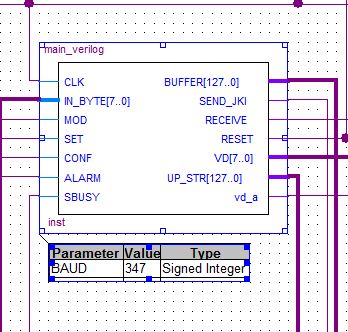
Блок **Main** передает в **LCD\_Main** данные и управляющий сигнал о начале записи данных. Из блока **LCD\_Main** данные передается в блок **LCD\_DRAM**, информация о задержке – в блок **LCD\_Command**, инструкции в блок **LCD\_Loader**. **LCD\_DRAM** разбивает полученные от **LCD\_Main** данные на пакеты. Полученные пакеты данных и управляющий сигнал далее передается в **LCD\_Loader**. **LCD\_Loader** посылает управляющий сигнал и данные на **ЖКИ**. Передача данных осуществляется по подаче управляющего сигнала, который может прийти либо от **LCD\_Command** либо от **LCD\_DRAM**. Из блока **LCD\_Main** информация о задержке поступает в блок **LCD\_Command**. Блок **LCD\_Command** формирует управляющие сигналы и отправляет их в **LCD\_Loader**. Управляющий сигнал содержит информацию о том будет передаваться в информационном пакете данные или инструкции.

Блоки связаны между собой управляющим сигналом который говорит о занятости или не занятости блока. Имеется синхросигнал. Он идет ко всем блокам от таймера (на схеме не отображен).

# Функциональная схема и ее описание (включая описание автоматов в виде графов).

## Блок Main

На рисунке 2 приведен блок Main.



Рисунок

**Входы**

* CLK: входной тактовый сигнал с частотой 40 МНг и периодом 25нс
* [7:0] IN\_BYTE: байт принятый по RS232
* MOD: сигнал переключения приема и отправки
* SET: сигнал переключения настройки будильника и часов
* CONF: сигнал подтверждения настройки
* ALARM: сигнал включения, отключения будильника
* SBUSY: сигнал занятости подчиненного устройства (0-свободен, 1-занят)

**Выходы**

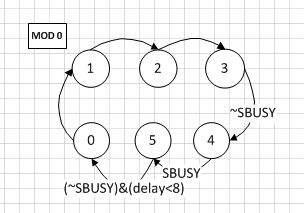
* [127:0] BUFFER: нижняя строка
* SEND\_JKI: управляющий сигнал вывода инфы на экран
* RECEIVE: управляющий сигнал - начать прием байт по RS232
* RESET: сброс ведомых устройств (для вывода их из режима ожидания байта по RS232)
* [7:0] VD: информация о состоянии передачи на светодиоды
* [127:0] UP\_STR: верхняя строка
* vd\_a : вывод о включенном будильнике

**Состояния**

**MOD 0:** режим часов

* 0: запись данных в строки для последующей передачи на ЖКИ;
* 1: после выдерживания задержки формирование из принятых данных двузначных чисел соответствующих времени;
* 2: формирование данных в буферы для вывода времени в бинарном формате;
* 3: отправка управляющего сигнала записи на ЖКИ;
* 4: ожидание освобождения устройства;
* 5: ждем освобождения устройства и 1 секунду.

На рисунке 3 приведен граф автомата.



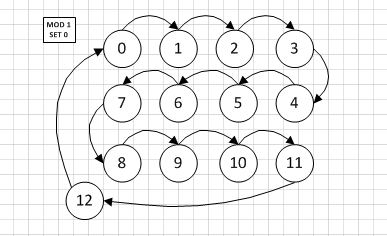
Рисунок

**MOD 1:** режим настройки

SET 0: настройка часов

* 0: устанавливаем начальные значения
* 1: если ЖКИ не занят, отправляем управляющий сигнал записи на ЖКИ
* 2: сброс управляющих сигналов, увеличение счётчика принятых байт
* 3: Если ЖКИ занят, установить RECEIVE в 0, выбрать следующее состояния в зависимости от получаемого пакета, иначе установить RECEIVE в 1.
* 4: Если управляемые устройства свободны, в зависимости от значения счетчика принятых байт записать на определенные позиции в буфер полученные данные и перейти в состояние 2, если это первый байт пакета, или в состояние 1, если это второй байт;
* 5: при нажатой кнопке переход в состояние 7, иначе принять следующие байты, записать их на соответствующие позиции и перейти в соответствующие состояния в зависимости от значения счетчика принятых байт. Если пакет четвертый, то увеличить светодиоды на 2;
* 6: если кнопка нажата, перейти в 7 состояние, иначе принять следующие байты, записать их на соответствующие позиции и перейти в соответствующие состояния в зависимости от значения счетчика принятых байт;
* 7: зажечь все диоды, сбросить счетчик секунд, записать время в строку для преобразований;
* 8: проверка и исправление правильности принятых секунд и минут в соответствии с форматом времени.
* 9: проверка и исправление правильности принятых часов;
* 10: запись времени в строку для передачи;
* 11: если подчиненные устройства свободны, отправки управляющего сигнала записи;
* 12: установка RESET в 1;

На рисунке 4 приведен граф автомата.

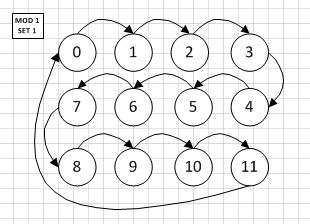


Рисунок

SET 1: настройка будильника

* 0: установка начальных значений;
* 1: отправка управляющего сигнала на ЖКИ
* 2: увеличение счетчика принятых байт
* 3: выбор состояния в зависимости от пакета
* 4: запись в буфер принятых данных в зависимости от значения счетчика принятых байт. Если счетчик равен двум, увеличить значения светодиодов
* 5:если кнопка нажата, перейти в 6 состояние, иначе записать на соответствующие позиции и перейти в соответствующие состояния в зависимости от счетчика принятых байт
* 6: зажечь светодиоды, записать принятые данные в переменную для редактирования
* 7: исправить время будильника в случае необходимости
* 8: исправить время будильника в случае необходимости для младших разрядов
* 9: установить диапазон работы будильника
* 10: если устройства свободны передать управляющий сигнал записи на светодиоды
* 11: установить RESET в 1

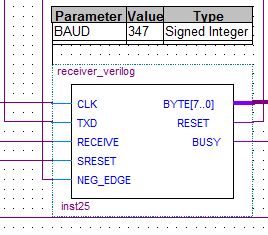
На рисунке 5 приведен граф автомата.



Рисунок

## Блок Receiver

На рисунке 6 приведен блок Receiver.



Рисунок

**Входы**

* CLK: входной тактовый сигнал с частотой 40 МГц и периодом 25нс
* TXD: линия приема данных c RS232
* RECEIVE: управляющий сигнал - начать прием байта по RS232
* SRESET: сигнал сброса ведомого устройства
* NEG\_EDGE: сигнал обнаружения среза (заднего фронта) сигнала на линии txd

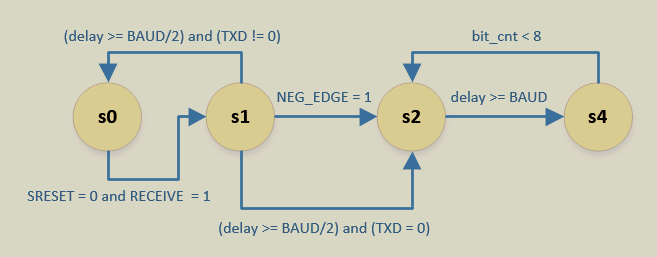
**Выходы**

* [7:0] BYTE: выдача принятого байта
* RESET: сброс ведомых устройств (для вывода их из режима ожидания байта по RS232)
* BUSY: сигнал занятости текущего устройства (0-свободен, 1-занят)

**Состояния**

* s0: сбрасывает сигнал занятости, сигнал сброса, регистр задержки и счетчик отсылаемых бит. Если получен управляющий сигнал приема байта и сброс не установлен, то переход на s1;
* s1: устанавливает флаг занятости, сбрасывает сигнал сброса. Если обнаружен срез на RS232, то на s2;
* s2: увеличиваем задержку, если она больше половины периода, то если на линии приема данных 0, то на s3 и сброс задержки, иначе на s0.;
* s3: увеличиваем задержку, если она больше периода, то сбрасываем её, увеличиваем счетчик отсылаемых бит, записываем полученную информацию и на s4;
* s4: увеличиваем задержку. Если бит меньше 8, то на s3. В другом случае, если задержка больше периода, то сбрасываем задержку, сбрасываем сигнал занятости.

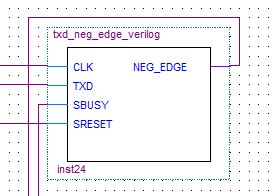
На рисунке 7 приведен граф автомата.



Рисунок

## Блок TXD\_neg\_edge

На рисунке 8 приведен блок Txd\_neg\_edge.



Рисунок

**Входы**

* CLK: входной тактовый сигнал с частотой 40МГц и периодом 25нс
* TXD: линия приема данных с RS232
* SBUSY: сигнал занятости подчиненного устройства (0-свободен, 1-занят)
* SRESET: управляющий сигнал сброса

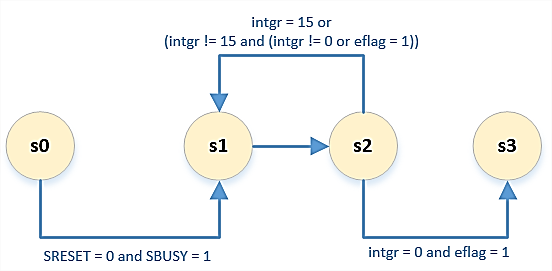
**Выходы**

* NEG\_EDGE: вырабатываемый сигнал обнаружения среза (заднего фронта) сигнала на линии TXD

**Состояния**

* s0: сбрасываем сигнал обнаружения среза, регистр среза и флаг единичного уровня;
* s1: если обнаружен единичный уровень TXD, то увеличиваем регистр среза, иначе уменьшаем, учитывая пределы – 0..15;
* s2: если зафиксировано 15 единичных уровней подряд, то устанавливаем флаг единичного уровня, переходим в s1. Если зафиксировано 15 нулевых уровней подряд, то на s3, иначе на s1;
* s3: устанавливаем сигнал обнаружения среза.

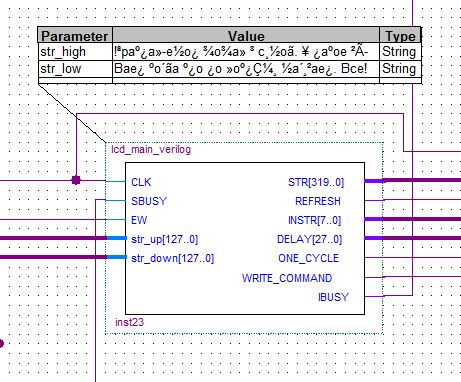
На рисунке 4 приведен граф автомата.



Рисунок

## Блок LCD\_Main

На рисунке 10 приведен блок LCD\_Main.



Рисунок

**Входы**

* CLK: 40Mhz 25ns
* SBUSY: (slave busy) сигнал занятости подчиненного устройства
* EW: можно писать
* [127:0] str\_up:
* [127:0] str\_down:

**Выходы**

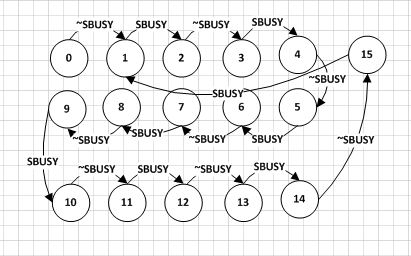
* [39:0] STR = 0: строка для вывода
* REFRESH = 0: обновление строки в LCD DRAM
* [7:0] INSTR = 0: инструкция LCD контроллеру
* [27:0] DELAY = 0: задержка между инструкциями
* ONE\_CYCLE = 0: 1) 4- или (0) 8-битная команда
* WRITE\_COMMAND = 0: запуск процесса загрузки команды в LCD
* IBUSY : занят -1, не занят 0. Можно ли посылать команду замены строки

**Состояния**

lcd\_state 0**:** инициализация

* 0-5: команды функциональных установок;
* 6-7: команда переключения в 4-битный режим;
* 8-9: команда определения количества строк(2) и шрифта дисплея(5\*8 точек);
* 10-11: команда включения дисплея, выключения курсора и его мигания;
* 12-13: команда очистки дисплея;
* 14-15: команда выбора режима сдвига курсора вправо при записи/чтении DDRAM, переход на выполнение команд.

На рисунке 11 приведен граф автомата.

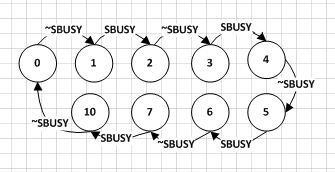


Рисунок

lcd\_state 1**:** выполнение команд

* 0-1: cмещение указателя RAM на верхнюю строку;
* 2-3: запись в lcd\_DRAM верхней строки;
* 4-5: смещение указателя RAM на нижню строку;
* 6-7: запись в lcd\_DRAM нижней строки;
* 10: ожидание сигнала от блока управления, если состояние кнопки изменилось переход в 0 состояние, запоминание состояния кнопки.

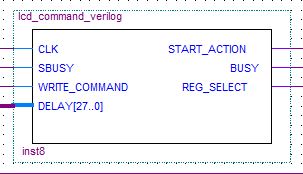
На рисунке 12 приведен граф автомата

****

Рисунок

## Блок LCD\_Command

На рисунке 13 приведен блок LCD\_Command.



Рисунок

**Входы**

* CLK: 40Mhz 25 ns
* SBUSY: (slave busy) сигнал занятости подчиненного устройства
* WRITE\_COMMAND: запуск процесса загрузки команды в LCD
* [27:0] DELAY: задержка между инструкциями

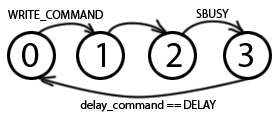
**Выходы**

* START\_ACTION = 0: сигнал запуска lcd\_loader
* BUSY = 0: запуск процесса загрузки команды в LCD
* REG\_SELECT = 0:

**Состояния**

* 0: Если необходимо загрузить команду в LCD, то начинает работу:
* 1: устанавливает флаг занятости, устанавливает флаг записи команды;
* 2: если не занят lcd\_loader\_verilog, то отправляет команду;
* 3: наращивает задержку, если задержка равна полученной, то переходит в начальное состояние, сбравсывает флаг занятости в 0 состоянии.

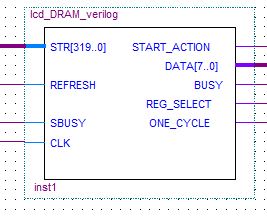
На рисунке 14 приведен граф автомата



Рисунок

## Блок LCD\_DRAM

На рисунке 15 приведен блок LCD\_DRAM.



Рисунок

**Входы**

* input [319:0] STR: строка для вывода
* input REFRESH: обновление строки в LCD DRAM в зависимости от POSITION
* input SBUSY: (slave busy) сигнал занятости подчиненного устройства
* input CLK: 10Mhz

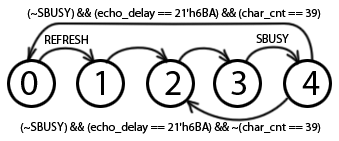
**Выходы**

* output reg START\_ACTION = 0: сигнал запуска lcd\_loader
* output reg [7:0] DATA = 0: инструкция LCD контроллеру
* output reg BUSY = 0: запуск процесса загрузки команды в LCD
* output reg REG\_SELECT = 0: флаг выбор регистра - переключает между записью команд и записью данных в память
* output reg ONE\_CYCLE = 0: 1) 4- или (0) 8-бит данных

**Состояния**

* 0: Если необходимо обновить строку в LCD DRAM, то начинает работу;
* 1: устанавливает флаг занятости, устанавливает флаг записи в память, устанавливает 8ми битные данные;
* 2: записывает информацию по 8 бит для передачи;
* 3: если не занят lcd\_loader\_verilog, то отправляет эти данные;
* 4: повторяет передачу 40 раз, то есть 320 бит информации передаст; по окончании сбрасывает флаг занятости в 0 состоянии.

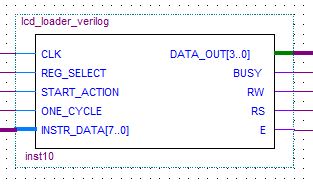
На рисунке 16 приведен граф автомата



Рисунок

## Блок LCD\_Loader

На рисунке 17 приведен блок LCD\_Loader.



Рисунок

**Входы**

* input CLK, // 10Mhz
* input REG\_SELECT, //флаг выбор регистра - переключает между записью команд и записью данных в память
* input START\_ACTION, //сигнал запуска lcd\_loader
* input ONE\_CYCLE, //(1) 4- или (0) 8-битная команда (данные)
* input [7:0] INSTR\_DATA, //инструкция/команда LCD контроллеру от вышестоящих устройств
* inout reg [3:0] DATA\_OUT, //4-х битный двунаправленный канал связи с контроллером LCD

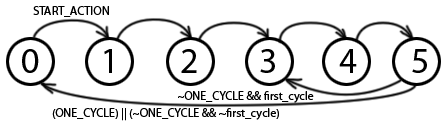
**Выходы**

* output reg BUSY = 0: запуск процесса загрузки команды в LCD
* output reg RW = 0: флаг чтения/записи из двунаправленного канала связи с контроллером LCD
* output reg RS = 0: флаг выбор регистра - переключает между записью команд и записью данных в память
* output reg E = 0: старт бит при пересылки команд и данных в контроллер

**Состояния**

* 0: Если есть сигнал запуска, то начинает работу:
* 1: устанавливает флаг занятости, устанавливает флаг первого цикла, переключается в режим записи команд/данных;
* 2: ожидание задержки, которая организует установку данных на выходных контактах;
* 3: установка старт бита и первой(второй) части инструкции/команды;
* 4: выдержка высокого уровня сигнала, сброс флага первого цикла, сброс старт бита;
* 5: выдержка низкого уровня сигнала;
* если 4х битная команда или две части 8ми битной отправлено, то на начало со сбросом флага занятости;
* если 8ми битная, то сброс флага первого цикла и отправка второй части, переход на 3.

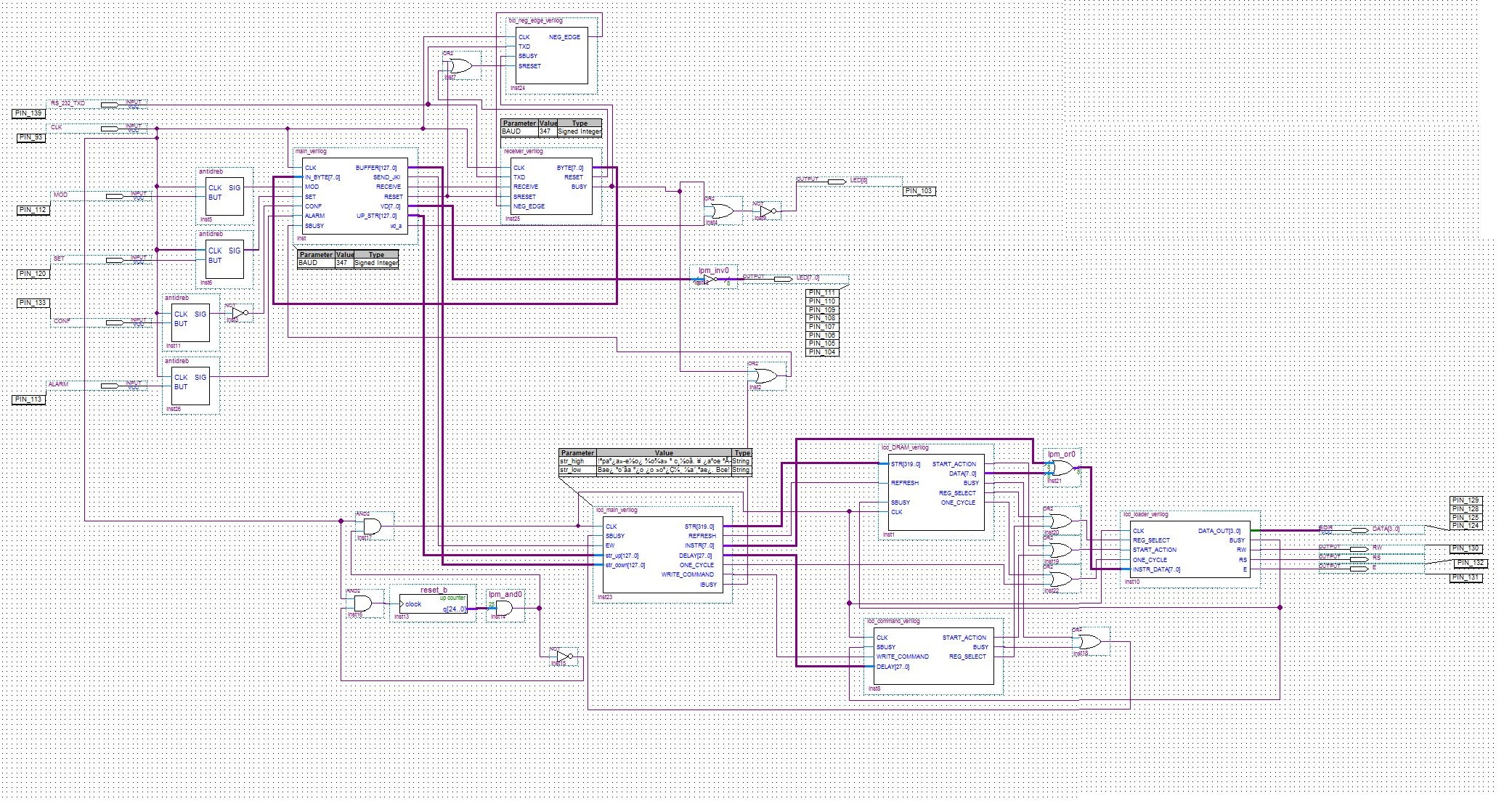
На рисунке 18 приведен граф автомата.



Рисунок

## Структурная схема

Полная структурная схема приведена на рисунке 19.



Рисунок

# Заключение.

В результате работы создано устройство бинарные часы с будильником. Устройство разработано и функционирует согласно техническому заданию.

# Список литературы.

1. Схемотехника ЭВМ / А.Н.Осокин, А.Н. Мальчуков; Издательство Томского политехнического университета. Томск, 2013. – 171с.
2. Стешенко В. Б. ПЛИС фирмы ALTERA: элементная база, система проектирования и языки описания аппаратуры. 3-е изд., стереотипное. – М. Издательский дом «ДодэкаXXI», 2007 – 573 c.
3. Угрюмов Е.П. Цифровая схемотехника: учеб. пособие для вузов. – 3-е изд., перераб. и доп. – СПб.: БХВ – Петербург, 2010. – 816 c.

# Приложения (руководство пользователя с фотографиями устройства и исходные коды модулей на языке описания аппаратуры).

# Руководство пользователя

## Общие сведения об устройстве

Устройство «Бинарные часы»

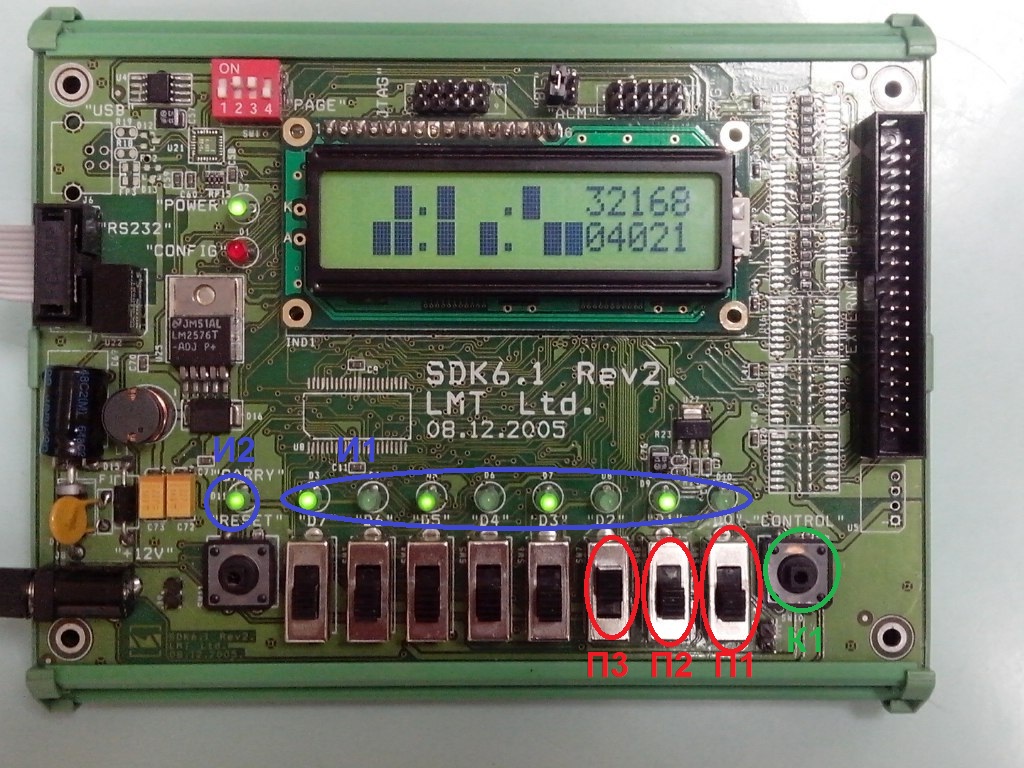


Рисунок 20

Устройство «Бинарные часы» имеют три режима работы:

1. Основной режим – Демонстрация времени
2. Установка времени
3. Установка будильника

## Режим «Демонстрация времени»

В этом режиме на дисплее устройства отображается текущее время в бинарном формате. (Рисунок 20)

Часы минуты и секунды отделены друг от друга двумя точками. Каждый составляющая времени (часы, минуты, секунды) отображается с помощью 6 индикаторов. Каждый индикатор имеет 2 состояния: горит, не горит. Каждый индикатор соответствует значению степени двойки от нулевой до пятой: 1, 2, 4, 8, 16, 32. Для перевода времени из бинарного формата в традиционный необходимо сложить значения соответствующие горящим индикаторам для каждой составляющей времени. Соответствие индикаторов и степеней двойки отображается на дисплее устройства.

## Режим «Установка времени»

Для активации режима «Установка времени» необходимо переключатель **П1**(Рисунок 21) установить в верхнее положение.

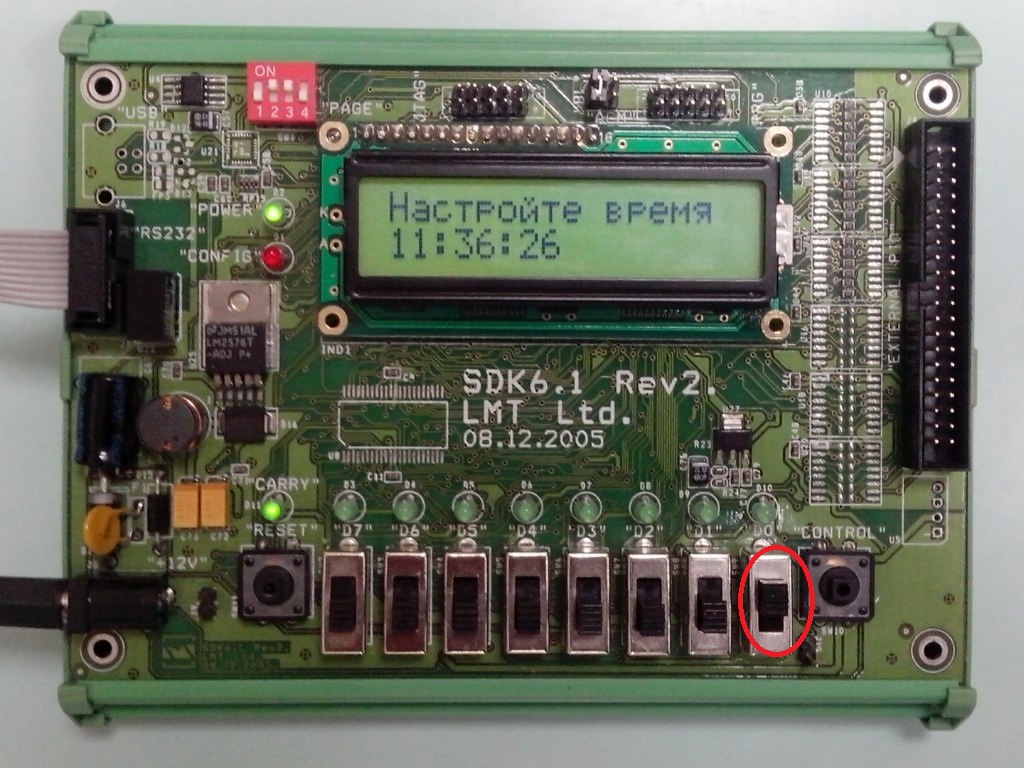


Рисунок 21

Установка времени осуществляется через интерфейс RS232 с помощью приложения *Term\_V1\_9b* или любого другого приложения позволяющего передавать данные через RS232 с характеристиками: baud rate 9600, data bits 8, stop bits 1.

Для установки времени необходимо отправлять на устройство через RS232 двоичные числа в последовательности часы, минуты, секунды. После передачи подтвердить ввод можно либо нажатием кнопки **К1,** либо выходом из режима установки переключением **П1** в нижнее положение.

Если время передано в неверном формате, оно автоматически приводится к правильному.

## Режим «Установка будильника»

Для активации режима «Установка будильника» необходимо переключатели **П1 и П2**  установить в верхнее положение. (Рисунок 22)

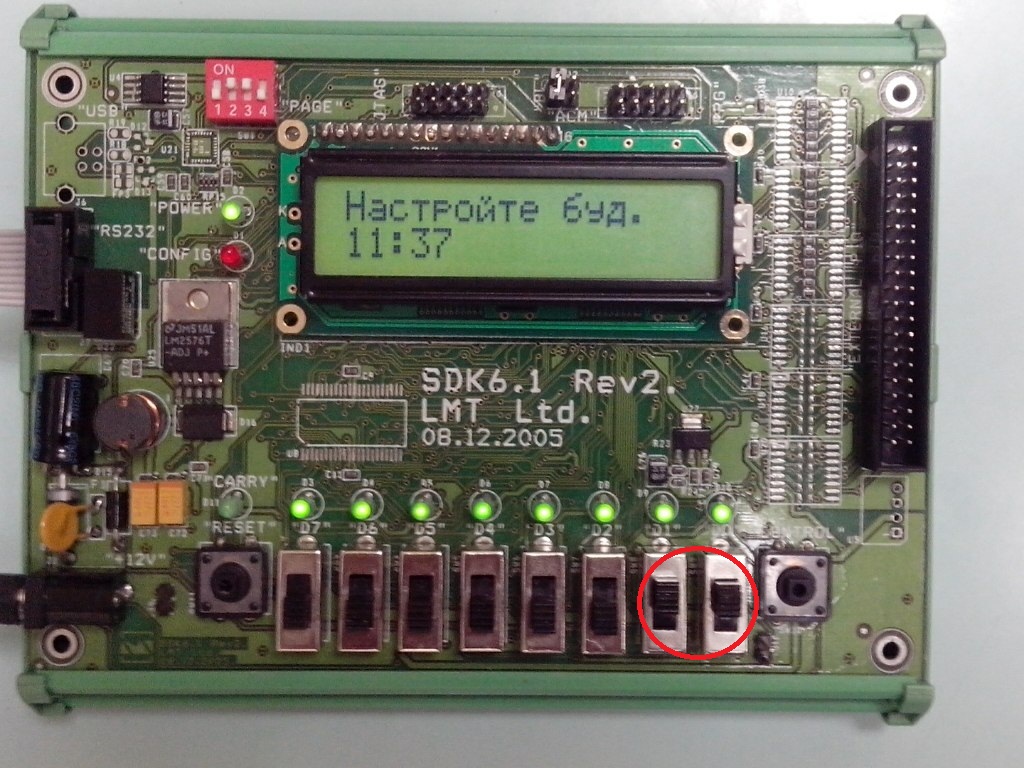


Рисунок 22

Установка времени срабатывания будильника осуществляется через интерфейс RS232 с помощью приложения *Term\_V1\_9b* или любого другого приложения позволяющего передавать данные через RS232 с характеристиками: baud rate 9600, data bits 8, stop bits 1.

Для установки времени срабатывания будильника необходимо отправлять на устройство через RS232 двузначные числа в последовательности часы, минуты.

Если время будильника передано в не верном формате, то оно автоматически будет приведено к верному.

## Включение/выключение будильника 3

Работа будильника в данном устройстве заключается в поочередном зажигании светодиодов группы **И1** (Рисунок 20) в установленное время.

Для того что бы будильник сработал в установленное время необходимо включить будильник. (Установка времени срабатывания будильника описывается в пункте **Режим «Установка будильника».**) Включение будильника осуществляется перемещением переключателя **П3** в верхнее положение. Горящий индикатор **И2** сообщает о включенном будильнике. Для выключения будильника необходимо переместить переключатель **П3** нижнее положение.

В случае если время срабатывания будильника не было установлено пользователем, оно устанавливается автоматически в значение 00:00.

Для прекращения работы будильника без его выключения необходимо нажать клавишу **К1**. Зажигание светодиодов будет прекращено, но при следующем совпадении времени срабатывания будильника с текущим временем будильник снова сработает.

Так же прекратить работу будильника можно выключив его (переключатель **П3** в нижнее положение). В таком случае, при следующем совпадении времени срабатывания будильника с текущим временем будильник не сработает, о чем сигнализирует не горящий светодиод **И2**.

# Исходные коды модулей на языке Verilog

## Main

module main\_verilog (

input CLK, // входной тактовый сигнал с частотой 40 МНг и периодом 25нс

input [7:0] IN\_BYTE, // байт принятый по RS232

input MOD, // сигнал переключения приема и отправки

input SET, //сигнал переключения настройки будильника и часов

input CONF, //сигнал подтверждения настройки

input ALARM, //сигнал включения, отключения будильника

input SBUSY, // сигнал занятости подчиненного устройства (0-свободен, 1-занят)

output [127:0] BUFFER,//нижняя строка

output reg SEND\_JKI, //управляющий сигнал вывода инфы на экран

output reg RECEIVE, // управляющий сигнал - начать прием байт по RS232

output reg RESET, // сброс ведомых устройств (для вывода их из режима ожидания байта по RS232)

output [7:0] VD, //информация о состоянии передачи на светодиоды

output [127:0] UP\_STR,//верхняя строка

output vd\_a //вывод о включенном будильнике

);

parameter BAUD = 347; // выбор делителя для скорости 115200

reg cl\_al=1;//сбрасывали ли за сутки будильник: 1-нет, 0-да

reg [127:0] up\_str; //буфер для хранения верхней строки

reg [127:0] buffer; // буфер для хранения нижней строки

reg [7:0] vd = 0; //блок инфы для светодиодов

reg [2:0] receive\_cnt = 0; // счетчик полученных байт

reg [4:0] r\_state\_al = 0; // автомат приема байт будильника

reg [4:0] r\_state\_ti = 0; // автомат приема байт таймера

reg [4:0] s\_state\_ti = 0; // автомат отправки данных о времени на ЖК

reg [4:0] t\_state = 0; //автомат приращения часов

reg [127:0] time1={"00:00:00 "}; // время в символьной форме

reg [127:0] alarm1={"00:00 "}; //запоминает будильник

reg [127:0] alarm2={"00:00:00 "}; //нижняя граница звона будильника

reg [127:0] alarm3={"00:00:59 "}; //верхняя граница звона будильника

reg [16:0] bin\_time; //время в числовой форме

reg [28:0] delay = 0;//задержка для прибавления секунды

assign BUFFER = buffer;

assign VD = vd;

assign UP\_STR = up\_str;

assign vd\_a = ALARM;

always @(posedge CLK)

begin

delay<=delay+29'd1;//тиканье часов - один такт 25 нс

if (delay>=40000000)

begin

delay<=0;

end

if (delay<8)

case (t\_state)//правильное приращение часов

0: begin time1[67:64]=time1[67:64]+4'd1; t\_state<=1; end //прибавили к первому разряду секунд

1: if (time1[67:64]>9) begin time1[75:72]= time1[75:72]+4'd1;time1[67:64]=0; t\_state<=2; end

else t\_state<=7;

2: if (time1[75:72]>5) begin time1[91:88]= time1[91:88]+4'd1;time1[75:72]=0; t\_state<=3; end

else t\_state<=7;

3: if (time1[91:88]>9) begin time1[99:96]= time1[99:96]+4'd1;time1[91:88]=0; t\_state<=4; end

else t\_state<=7;

4: if (time1[99:96]>5) begin time1[115:112]= time1[115:112]+4'd1;time1[99:96]=0;

if ((time1[115:112]>9)&&(time1[123:120]<2)) t\_state<=5; //если час 0 и 1 -старший и младший>9, то увеличиваем старший

else if ((time1[115:112]==4)&&(time1[123:120]==2)) t\_state<=6; end //если 24 часа, то все обнуляем

else t\_state<=7;

5: begin time1[123:120]= time1[123:120]+4'd1;time1[115:112]=0; t\_state<=7; end

6: begin time1[123:120]= 0;time1[115:112]=0; t\_state<=7; end

endcase

else t\_state<=0;

if ((alarm2>=time1)||(ALARM==0)) cl\_al=1;//будильник будет срабатывать

case (MOD)

//режим часов

0: begin

RECEIVE = 0;

RESET = 1;

r\_state\_al<= 0;

r\_state\_ti<= 0;

if (CONF==1) cl\_al=0; //если нажали кнопку, то перестают мегать диоды

//обработка будильника

if ((ALARM==1)&&(cl\_al==1)&&(time1>=alarm2)&&(time1<=alarm3)) begin

if (delay<=20000000) vd<=170;

else vd<=85; end

else vd<=0;

case (s\_state\_ti)//отправляем время на ЖКИ

0: begin up\_str = {" . . 32168"}; buffer={" . . 04021"}; s\_state\_ti <= 1; time1[63:0] = {" "};alarm1[63:0] = {" "}; end

1: if (delay>8) begin//формирования двузначных чисел времени из цифр

bin\_time[16:12]=time1[115:112]+time1[123:120]\*4'd10;

bin\_time[11:6]=time1[91:88]+time1[99:96]\*4'd10;

bin\_time[5:0]=time1[67:64]+time1[75:72]\*4'd10;

s\_state\_ti <= 2;

end

2: begin//формирование строк для вывода бинарного времени

//часы

if (bin\_time[16:16]) up\_str[119:112]=8'b11111111; else up\_str[119:112]=8'b00100000;

if (bin\_time[15:15]) up\_str[111:104]=8'b11111111; else up\_str[111:104]=8'b00100000;

if (bin\_time[14:14]) buffer[127:120]=8'b11111111; else buffer[127:120]=8'b00100000;

if (bin\_time[13:13]) buffer[119:112]=8'b11111111; else buffer[119:112]=8'b00100000;

if (bin\_time[12:12]) buffer[111:104]=8'b11111111; else buffer[111:104]=8'b00100000;

//минуты

if (bin\_time[11:11]) up\_str[95:88]=8'b11111111; else up\_str[95:88]=8'b00100000;

if (bin\_time[10:10]) up\_str[87:80]=8'b11111111; else up\_str[87:80]=8'b00100000;

if (bin\_time[9:9]) up\_str[79:72]=8'b11111111; else up\_str[79:72]=8'b00100000;

if (bin\_time[8:8]) buffer[95:88]=8'b11111111; else buffer[95:88]=8'b00100000;

if (bin\_time[7:7]) buffer[87:80]=8'b11111111; else buffer[87:80]=8'b00100000;

if (bin\_time[6:6]) buffer[79:72]=8'b11111111; else buffer[79:72]=8'b00100000;

//секунды

if (bin\_time[5:5]) up\_str[63:56]=8'b11111111; else up\_str[63:56]=8'b00100000;

if (bin\_time[4:4]) up\_str[55:48]=8'b11111111; else up\_str[55:48]=8'b00100000;

if (bin\_time[3:3]) up\_str[47:40]=8'b11111111; else up\_str[47:40]=8'b00100000;

if (bin\_time[2:2]) buffer[63:56]=8'b11111111; else buffer[63:56]=8'b00100000;

if (bin\_time[1:1]) buffer[55:48]=8'b11111111; else buffer[55:48]=8'b00100000;

if (bin\_time[0:0]) buffer[47:40]=8'b11111111; else buffer[47:40]=8'b00100000;

s\_state\_ti <= 3; end

3: begin//отправка управляющего сигнала записи на ЖКИ

if (SBUSY == 0) begin

SEND\_JKI = ~SEND\_JKI;

s\_state\_ti <= 4;

end

end

4: begin

if (SBUSY == 1) begin

s\_state\_ti <= 5;

end

end

5: if ((SBUSY == 0)&&(delay<8)) s\_state\_ti <= 1;//ждем пока ЖКИ освободится и новой секунды

default: s\_state\_ti <= 0;

endcase

end

//режим настройки

1: begin

case (SET)

0: begin

case (r\_state\_ti)//настройка часов

0: begin//задание начальных строк, которые надо вывести

up\_str = {"Hac",8'b10111111,"po",8'b10111001,8'b10111111,"e ",8'b10110011,"pe",8'b10111100,8'b11000111," "};

buffer=time1;//выводим текущее время

alarm1[63:0] = {" "};//стирание надписи "ОШИБКА"

receive\_cnt<=0;

r\_state\_ti<=1;

vd<=0;

r\_state\_al<= 0;

RESET = 1;//сбрасывает ожидание приема

s\_state\_ti<=0;

end

1: begin//отправка на ЖКИ

if (SBUSY == 0) begin//если не занят, то управляющий сигнал записи

SEND\_JKI =~SEND\_JKI;

r\_state\_ti<=2;

end

end

2: begin//увеличение принятых байт

RESET = 0;

RECEIVE = 0;

receive\_cnt = receive\_cnt + 3'd1;

r\_state\_ti <= 3;

end

3: begin//смотрим какой пакет получать данных

if (SBUSY == 1) begin

RECEIVE = 0;

case (vd)

0: begin r\_state\_ti <= 4; end

1: begin r\_state\_ti <= 5; end

3: begin r\_state\_ti <= 6; end

endcase

end

else RECEIVE = 1;

end

4: begin//первый пакет в буфер - часы

if (SBUSY == 0) begin

RESET = 1;

case (receive\_cnt)

1: begin

buffer[127:120] = IN\_BYTE;

r\_state\_ti <= 2;//без обновления ЖКИ

end

2: begin

buffer[119:112] = IN\_BYTE;

r\_state\_ti <= 1;//с обновлением ЖКИ

vd = vd+8'd1;

end

endcase

end

end

5: begin//второй пакет в буфер - минуты

if (CONF==1) begin r\_state\_ti<=7; end//если кнопку нажали, то сохраняем

else//иначе дальше принимаем

if (SBUSY == 0) begin

RESET = 1;

case (receive\_cnt)

3: begin

buffer[103:96] = IN\_BYTE;

r\_state\_ti <= 2;

end

4: begin

buffer[95:88] = IN\_BYTE;

r\_state\_ti <= 1;

vd = vd+8'd2;

end

endcase

end

end

6: begin//третий пакет в буфер - секунды

if (CONF==1) begin r\_state\_ti<=7; end

else

if (SBUSY == 0) begin

RESET = 1;

case (receive\_cnt)

5: begin

buffer[79:72] = IN\_BYTE;

r\_state\_ti <= 2;

end

6: begin

buffer[71:64] = IN\_BYTE;

r\_state\_ti <= 7;

end

endcase

end

end

7: begin//зажигаем светодиоды и из буффера записываем время

vd<=255;

time1<=buffer;

delay<=0; //сбросим счетчик секунд

r\_state\_ti<=8;

end

8: begin //проверка и исправление правильности секунд и минут

if (time1[71:64]>8'b00111001) time1[71:64]=8'b00111001;

if (time1[71:64]<8'b00110000) time1[71:64]=8'b00110000;

if (time1[79:72]>8'b00110101) time1[79:72]=8'b00110101; if (time1[79:72]<8'b00110000) time1[79:72]=8'b00110000; if (time1[95:88]>8'b00111001) time1[95:88]=8'b00111001; if (time1[95:88]<8'b00110000) time1[95:88]=8'b00110000; if (time1[103:96]>8'b00110101) time1[103:96]=8'b00110101; if (time1[103:96]<8'b00110000) time1[103:96]=8'b00110000; if (time1[127:120]>8'b00110010) time1[127:120]=8'b00110010; if (time1[127:120]<8'b00110000) time1[127:120]=8'b00110000;

r\_state\_ti<=9;

end

9: begin//проверка и исправление правильности часов

if (time1[127:120]==8'b00110010) begin

if (time1[119:112]>8'b00110011) time1[119:112]=8'b00110011; end

else begin

if (time1[119:112]>8'b00111001) time1[119:112]=8'b00111001; end

if (time1[119:112]<8'b00110000) time1[119:112]=8'b00110000;

r\_state\_ti<=10;

end

10: begin buffer=time1; r\_state\_ti<=11; end //запись времени в строку для передачи

11:if (SBUSY == 0)

begin //отправка управляющего сигнала на запись

SEND\_JKI =~SEND\_JKI;

r\_state\_ti<=12;

end

12: begin

RESET = 1;

end

default: r\_state\_ti<= 0;

endcase

end

1: begin

case (r\_state\_al)//настройка будильника

0: begin

up\_str = {"Hac",8'b10111111,"po",8'b10111001,8'b10111111,"e ",8'b10110010,"y",8'b11100011,". "};

time1[63:0] = {" "};//стирание надписи "ОШИБКА"

buffer=alarm1;//загружаем старый будильник

receive\_cnt<=0;

r\_state\_al<=1;

vd<=0;

r\_state\_ti<= 0;

RESET = 1;

s\_state\_ti<=0;

end

1: begin

if (SBUSY == 0) begin //отправка на жки

SEND\_JKI =~SEND\_JKI;

r\_state\_al<=2;

end

end

2: begin

RESET = 0;

RECEIVE = 0;

receive\_cnt = receive\_cnt + 3'd1;

r\_state\_al <= 3;

end

3: begin

if (SBUSY == 1) begin//выбор получаемого разряда(минуты, часы)

RECEIVE = 0;

case (vd)

0: begin

r\_state\_al <= 4;

end

1: begin

r\_state\_al <= 5;

end

endcase

end

else RECEIVE = 1;

end

4: begin//получение часов

if (SBUSY == 0) begin

RESET = 1;

case (receive\_cnt)

1: begin

buffer[127:120] = IN\_BYTE;

r\_state\_al <= 2;

end

2: begin

buffer[119:112] = IN\_BYTE;

r\_state\_al <= 1;

vd = vd+8'd1;

end

endcase

end

end

5: begin//получение минут

if (CONF==1) begin r\_state\_al<=6; end//подтверждение - конец приема

else

if (SBUSY == 0) begin

RESET = 1;

case (receive\_cnt)

3: begin

buffer[103:96] = IN\_BYTE;

r\_state\_al <= 2;

end

4: begin

buffer[95:88] = IN\_BYTE;

r\_state\_al <= 6;

end

endcase

end

end

6: begin

vd<=255;

alarm1=buffer;

r\_state\_al<=7;

end

7: begin

if (alarm1[95:88]>8'b00111001) alarm1[95:88]=8'b00111001;

if (alarm1[95:88]<8'b00110000) alarm1[95:88]=8'b00110000;

if (alarm1[103:96]>8'b00110101) alarm1[103:96]=8'b00110101;

if (alarm1[103:96]<8'b00110000) alarm1[103:96]=8'b00110000;

if (alarm1[127:120]>8'b00110010) alarm1[127:120]=8'b00110010;

if (alarm1[127:120]<8'b00110000) alarm1[127:120]=8'b00110000;

r\_state\_al<=8;

end

8: begin

if (alarm1[127:120]==8'b00110010) begin

if (alarm1[119:112]>8'b00110011) alarm1[119:112]=8'b00110011; end

else begin

if (alarm1[119:112]>8'b00111001) alarm1[119:112]=8'b00111001; end

if (alarm1[119:112]<8'b00110000) alarm1[119:112]=8'b00110000;

r\_state\_al<=9;

end

9: begin //перезапись время срабатывания будильника(00:00) в время работы будильника(00:00:00-59)

buffer=alarm1;

alarm2[127:88]=alarm1[127:88];

alarm3[127:88]=alarm1[127:88];

r\_state\_al<=10; end

10:if (SBUSY == 0)

begin

SEND\_JKI =~SEND\_JKI;

r\_state\_al<=11;

end

11: begin

RESET = 1;

end

default: r\_state\_al<= 0;

endcase

end

endcase

end

endcase

end

endmodule

## Receiver

module receiver\_verilog(

input CLK, // входной тактовый сигнал с частотой 40 МГц и периодом 25нс

input TXD, // линия приема данных c RS232

input RECEIVE, // управляющий сигнал - начать прием байта по RS232

input SRESET, // сигнал сброса ведомого устройства

input NEG\_EDGE, // сигнал обнаружения среза (заднего фронта) сигнала на линии txd

output [7:0] BYTE, // выдача принятого байта

output reg RESET, // сброс ведомых устройств (для вывода их из режима ожидания байта по RS232)

output reg BUSY // сигнал занятости текущего устройства (0-свободен, 1-занят)

);

parameter BAUD = 347; // выбор делителя частоты для скорости ~115273 (для скорости 115200)

reg [2:0] state = 0; // автомат приема байта по RS232

reg [8:0] delay = 0; // регистр задержки для выработки нужной частоты

reg [7:0] data = 0; // внутренний регистр для приема байта

reg [3:0] cnt\_bit = 0; // счетчик отсылаемых бит

assign BYTE = data; //связь выхода с внутренним регистром

always @(posedge CLK)

begin

if (SRESET == 1) state = 0;

//автомат приема байт по RS232

case (state)

0: begin

BUSY = 0;

RESET = 1;

delay = 0;

cnt\_bit = 0;

if (SRESET == 0 && RECEIVE == 1) state = 1;

end

1: begin

BUSY = 1;

RESET = 0;

if (NEG\_EDGE == 1) state = 2;

end

2: begin //проверка нулевого уровня старт-бита

if (delay >= BAUD/2)

if (TXD == 0) begin

delay = 0;

state = 3;

end

else state = 0;

else delay = delay + 9'd1;

end

3: begin

if (delay >= BAUD) begin

delay = 0;

cnt\_bit = cnt\_bit + 4'd1;

data = {TXD, data[7:1]};

state = 4;

end

else delay = delay + 9'd1;

end

4: begin

if (cnt\_bit < 8) state = 3;

else if (delay >= (BAUD)) begin //ожидаем стоп бита

delay = 0;

BUSY = 0;

end

else delay = delay + 9'd1;

end

endcase

end

endmodule

## TXD\_neg\_edge

module txd\_neg\_edge\_verilog(

input CLK, // входной тактовый сигнал с частотой 40МГц и периодом 25нс

input TXD, // линия приема данных с RS232

input SBUSY, // сигнал занятости подчиненного устройства (0-свободен, 1-занят)

input SRESET, // управляющий сигнал сброса

output reg NEG\_EDGE // вырабатываемый сигнал обнаружения среза (заднего фронта) сигнала на линии TXD);

reg [3:0] intgr = 0; // регистр для обнаружения среза (заднего фронта) сигнала посылаемого по RS232

reg [1:0] state = 0; // регистр для определения среза (заднего фронта) сигнала

reg eflag = 0; // флаг, показывающий был ли обнаружен единичный уровень на линии TXD

always @(posedge CLK)

begin

if (SRESET == 1) state = 0; //организация сброса

case (state)

0: begin

NEG\_EDGE = 0;

eflag = 0;

intgr = 0;

if (SRESET == 0 && SBUSY == 1) state = 1;

end

1: begin

if (TXD == 1 && intgr < 15) intgr = intgr + 4'd1;

if (TXD == 0 && intgr > 0) intgr = intgr - 4'd1;

state = 2;

end

2: begin

if (intgr == 15) begin

eflag = 1;

state = 1;

end

else if (intgr == 0 && eflag == 1) state = 3;

else state = 1;

end

3: begin NEG\_EDGE = 1;

end

default: state = 0;

endcase

end endmodule

## LCD\_Main

module lcd\_main\_verilog (

input CLK, // 40Mhz 25ns

input SBUSY, //(slave busy) сигнал занятости подчиненного устройства

input EW, //можно писать

input [127:0] str\_up,

input [127:0] str\_down,

output reg [319:0] STR = 0, //строка для вывода

output reg REFRESH = 0, //обновление строки в LCD DRAM

output reg [7:0] INSTR = 0, //инструкция LCD контроллеру

output reg [27:0] DELAY = 0,//задержка между инструкциями

output reg ONE\_CYCLE = 0, //(1) 4- или (0) 8-битная команда

output reg WRITE\_COMMAND = 0,//,//запуск процесса загрузки команды в LCD

output reg IBUSY //я занят -1, не занят 0. Можно ли посылать команду замены строки

);

reg lcd\_state = 0;

reg [3:0] init\_state = 0;

reg [3:0] exec\_state = 0;

reg prev\_but = 0;//для кнопки особо, чтоб не прыгали строки

always @(posedge CLK)

begin

case (lcd\_state)

0: // Инициализация дисплея

case (init\_state)

0: begin

if (~SBUSY) begin

INSTR = 8'b0011\_XXXX;

DELAY = 28'h280A2; //28 разрядный регистр, мин зад. 4.1мс, такт 25 нс, => 164000 такта = 280A0 +2 в запас

ONE\_CYCLE = 1'b1;

init\_state <= 1;

end

IBUSY<=1;

end

1: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 2;

end

else WRITE\_COMMAND = 1;

end

2: begin

if (~SBUSY) begin

INSTR = 8'b0011\_XXXX;

DELAY = 28'hFA2; //28 разрядный регистр, мин зад. 100мкс, такт 25 нс, => 4000 такта = FA0 +2 в запас

ONE\_CYCLE = 1'b1;

init\_state <= 3;

end

end

3: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 4;

end

else WRITE\_COMMAND = 1;

end

4: begin

if (~SBUSY) begin

INSTR = 8'b0011\_XXXX;

DELAY = 28'hFA2; //28 разрядный регистр, мин зад. 100мкс, такт 25 нс, => 4000 такта = FA0 +2 в запас

ONE\_CYCLE = 1'b1;

init\_state <= 5;

end

end

5: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 6;

end

else WRITE\_COMMAND = 1;

end

// переключаем в 4-битный режим

6: begin

if (~SBUSY) begin

INSTR = 8'b0010\_XXXX;

DELAY = 28'h620; //28 разрядный регистр, мин зад. 39мкс, такт 25 нс, => 1560 такта = 618 +2 в запас

ONE\_CYCLE = 1'b1;

init\_state <= 7;

end

end

7: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 8;

end

else WRITE\_COMMAND = 1;

end

// определяем количество строк(2) и шрифт дисплея(5х8 точек)

8: begin

if (~SBUSY) begin

INSTR = 8'b0010\_1000;

DELAY = 28'h620; //28 разрядный регистр, мин зад. 39мкс, такт 25 нс, => 1560 такта = 618 +2 в запас

ONE\_CYCLE = 1'b0;

init\_state <= 9;

end

end

9: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 10;

end

else WRITE\_COMMAND = 1;

end

// включаем дисплей, выключаем курсор и его мигание

10: begin

if (~SBUSY) begin

INSTR = 8'b0000\_1100;

DELAY = 28'h620; //28 разрядный регистр, мин зад. 39мкс, такт 25 нс, => 1560 такта = 618 +2 в запас

ONE\_CYCLE = 1'b0;

init\_state <= 11;

end

end

11: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 12;

end

else WRITE\_COMMAND = 1;

end

// очищаем дисплей

12: begin

if (~SBUSY) begin

INSTR = 8'b0000\_0001;

DELAY = 28'hEF12; //28 разрядный регистр, мин зад. 1.53мс, такт 25 нс, => 61200 такта = EF10 +2 в запас

ONE\_CYCLE = 1'b0;

init\_state <= 13;

end

end

13: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

init\_state <= 14;

end

else WRITE\_COMMAND = 1;

end

// выбор режим сдвига курсора вправо при записи в(чтении из) DDRAM

14: begin

if (~SBUSY) begin

INSTR = 8'b0000\_0110;

DELAY = 28'h620; //28 разрядный регистр, мин зад. 39мкс, такт 25 нс, => 1560 такта = 618 +2 в запас

ONE\_CYCLE = 1'b0;

init\_state <= 15;

end

end

15: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

lcd\_state <= 1;

end

else WRITE\_COMMAND = 1;

end

default: init\_state <= 0;

endcase

1: // EXECUTING COMMANDS

case (exec\_state)

// Смещаем указатель RAM на верхнюю строку

0: begin

if (~SBUSY) begin

INSTR = 8'b1000\_0000;

DELAY = 28'h620;

ONE\_CYCLE = 1'b0;

exec\_state <= 1;

end

IBUSY<=1;

end

1: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

exec\_state <= 2;

end

else WRITE\_COMMAND = 1;

end

// Запись в lcd\_DRAM верхней строки

2: begin

if (~SBUSY) begin

INSTR[7:0] = 8'h00;

STR[319:192] = str\_up;

REFRESH = 1;

exec\_state <= 3;

end

end

3: begin

if (SBUSY) begin

STR[319:192] = str\_up;

REFRESH = 0;

exec\_state <= 4;

end

else REFRESH = 1;

end

//Смещаем указатель RAM на нижнюю строку

4: begin

if (~SBUSY) begin

INSTR = 8'b1100\_0000;

DELAY = 28'h620; //28 ?ac?yaiue ?aaeno?, iei caa. 39ien, oaeo 25 in, => 1560 oaeoa = 618 +2 a caian

ONE\_CYCLE = 1'b0;

exec\_state <= 5;

end

end

5: begin

if (SBUSY) begin

WRITE\_COMMAND = 0;

exec\_state <= 6;

end

else WRITE\_COMMAND = 1;

end

// Запись lcd\_DRAM нижней строки

6: begin

if (~SBUSY) begin

INSTR[7:0] = 8'h00;

STR[319:192] = str\_down;

REFRESH = 0;

exec\_state <= 7;

end

end

7: begin

if (SBUSY) begin

STR[319:192] = str\_down;

REFRESH = 0;

exec\_state <= 10;

end

else REFRESH = 1;

end

10: begin//ждем сигнала от блока управления

if (~SBUSY) begin

exec\_state <= 10;

IBUSY<=0;

if (EW != prev\_but) begin

exec\_state <= 0;//если менять направление только, то на 8 состояние прыгать. Если строки, то 2.

end

prev\_but=EW;//запоминаем предыдущее состояние кнопки, чтобы не прыгало

end

end

default: exec\_state <= 0;

endcase

default: lcd\_state <= 0;

endcase

end

endmodule

## LCD\_Comman

module lcd\_command\_verilog (

input CLK, // 40Mhz 25 ns

input SBUSY, //сигнал занятости подчиненного устройства

input WRITE\_COMMAND, //запуск процесса загрузки команды в LCD

input [27:0] DELAY, //задержка между инструкциями

output reg START\_ACTION = 0,//сигнал запуска lcd\_loader

output reg BUSY = 0, //запуск процесса загрузки команды в LCD

output reg REG\_SELECT = 0 );

reg [2:0] command\_state;

reg [27:0] delay\_command;

always @(posedge CLK)

begin

case (command\_state)

0: begin

BUSY = 0;

START\_ACTION = 0;

delay\_command = 0;

if (WRITE\_COMMAND) command\_state <= 1;

end

1: begin

BUSY = 1;

REG\_SELECT = 0;

command\_state <= 2;

end

2: begin

if (~SBUSY) begin

START\_ACTION = 1;

end

else begin

START\_ACTION = 0;

command\_state <= 3;

end

end

3: begin

if (~SBUSY) begin

delay\_command = delay\_command + 1'b1;

if (delay\_command == DELAY) command\_state <= 0;

end

end

default: command\_state <= 0;

endcase

end endmodule

## LCD\_DRAM

module lcd\_DRAM\_verilog (

input [319:0] STR, //строка для вывода

input REFRESH, //обновление строки в LCD DRAM в зависимости от POSITION

input SBUSY, //(slave busy) сигнал занятости подчиненного устройства

input CLK, // 10Mhz

output reg START\_ACTION = 0,//сигнал запуска lcd\_loader

output reg [7:0] DATA = 0, //инструкция LCD контроллеру

output reg BUSY = 0, //запуск процесса загрузки команды в LCD

output reg REG\_SELECT = 0, //флаг выбор регистра - переключает между записью команд и записью данных в память

output reg ONE\_CYCLE = 0 //(1) 4- или (0) 8-бит данных

);

//Автомат записи строк в память контроллера LCD

reg [319:0] str\_temp = 0;

reg [319:0] str\_high = 0;

reg [319:0] str\_low = 0;

reg [3:0] echo\_state = 0;

reg [1:0] load\_state = 0;

reg [6:0] char\_cnt = 0;

reg [20:0] echo\_delay = 0;

always @(posedge CLK)

begin

case (echo\_state)

0: begin

REG\_SELECT = 0;

char\_cnt = 0;

echo\_delay = 0;

START\_ACTION = 0;

DATA[7:0] = 8'h00;

if (REFRESH) echo\_state <= 1;

end

1: begin

BUSY = 1;

REG\_SELECT = 1;

str\_temp = REFRESH ? str\_high : str\_low;

ONE\_CYCLE = 0;

echo\_state <= 2;

end

//начало рабочего цикла

2: begin

DATA[7:0] = str\_temp[319:312];

echo\_delay = 0;

echo\_state <= 3;

end

3: begin

if (~SBUSY) begin

START\_ACTION = 1;

end

else begin

str\_temp = {str\_temp[311:0], str\_temp[319:312]};

START\_ACTION = 0;

echo\_state <= 4;

end

end

4: begin

if (~SBUSY) begin

echo\_delay = echo\_delay + 1'b1;

if (echo\_delay == 21'h6BA) begin //задержка 43мкс, 1720, 6B8+2

if (char\_cnt == 39) begin

BUSY = 0;

echo\_state <= 0;

end

else begin

char\_cnt = char\_cnt + 1'b1;

echo\_state <= 2;

end

end

end

end

default: echo\_state <= 0;

endcase

case (load\_state)

0: begin

if (REFRESH) begin

str\_high = STR;

end

else str\_low = STR;

load\_state <= 0;

end

default: load\_state <= 0;

endcase

end endmodule

## LCD\_Loader

module lcd\_loader\_verilog (

input CLK, // 10Mhz

input REG\_SELECT, //флаг выбор регистра - переключает между записью команд и записью данных в память

input START\_ACTION, //сигнал запуска lcd\_loader

input ONE\_CYCLE, //(1) 4- или (0) 8-битная команда (данные)

input [7:0] INSTR\_DATA, //инструкция/команда LCD контроллеру от вышестоящих устройств

inout reg [3:0] DATA\_OUT, //4-х битный двунаправленный канал связи с контроллером LCD

output reg BUSY = 0, //запуск процесса загрузки команды в LCD

output reg RW = 0, //флаг чтения/записи из двунаправленного канала связи с контроллером LCD

output reg RS = 0, //флаг выбор регистра - переключает между записью команд и записью данных в память

output reg E = 0 //старт бит при пересылки команд и данных в контроллер );

//блок lcd\_loader

reg first\_cycle; //флаг первого цикла - если 1, то идёт загрузка первой части команды (данных)

reg [5:0] epw = 6'h00; //задержка, организующая нужную длительность сигнала Е

reg [2:0] proc\_state = 3'h0; //автомат - реализует логику работы блока

reg [5:0] delay\_proc = 6'h00; //задержка, организующая ожидания установки данных на выходных контактах (RS, RW)

always @(posedge CLK) begin

case (proc\_state)

0: begin

BUSY = 0;

RW = 1'b0; //всегда пишем и не читаем

epw = 6'h00;

RS = 0;

E = 0;

DATA\_OUT = 4'bZZZZ;

delay\_proc = 6'h00;

if (START\_ACTION) proc\_state <= 1;

end

1: begin

BUSY = 1;

first\_cycle = 1;

RS = REG\_SELECT;

proc\_state <= 2;

end

2: begin

delay\_proc = delay\_proc + 1'b1;

if (delay\_proc == 60) begin //задержка, организующая ожидания установки данных на выходных контактах (RS, RW)

// delay\_proc = 0;

proc\_state <= 3;

end

end

3: begin

E = 1;

DATA\_OUT[3:0] = (first\_cycle) ? INSTR\_DATA[7:4] : INSTR\_DATA[3:0];

proc\_state <= 4;

end

4: begin

//выдерживаем длительность высокого уровня сигнала Е = min 230 ns

epw = epw + 1'b1;

if (epw == 25) begin //625 ns

epw = 0;

E = 0;

proc\_state <= 5;

end

end

5: begin

//выдерживаем длительность низкого уровня сигнала Е = min 230 ns

epw = epw + 1'b1;

if (epw == 25) begin //625 ns

epw = 0;

if ((ONE\_CYCLE) || (~ONE\_CYCLE && ~first\_cycle)) proc\_state <= 0;

else if (~ONE\_CYCLE && first\_cycle) begin

first\_cycle = 0; //первая половина команды/данных отправлена

proc\_state <= 3;

end

end

end

default: proc\_state <= 0;

endcase

end endmodule