

*Болдырев А.В.,  
кандидат технических наук,  
доцент кафедры «Автоматизация производственных процессов»  
Донской Государственный Технический Университет  
Россия, г. Ростов-на-Дону*

## **ЛОГИЧЕСКИЙ ИНТЕРФЕЙС КЛАВИАТУРЫ ДЛЯ СТЕНДА НА ОСНОВЕ ПЛИС**

***Аннотация:** В данной статье рассматривается реализация интерфейса стандартной клавиатуры на ПЛИС учебного стенда, предназначенного для отладки и моделирования программируемых схем. Представлены особенности ввода данных через модифицированный порт PS/2, часть кода описания на языке Verilog схемы интерфейса клавиатуры и схема его тестирования. Полученные результаты подтверждают корректную работу интерфейса.*

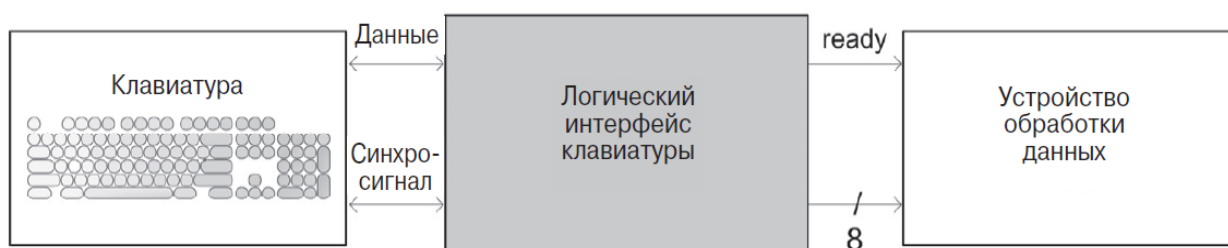
***Ключевые слова:** ПЛИС; учебный стенд; программирование; Cyclone; Intel/Altera; ASCII; Quartus II; Verilog.*

## **LOGICAL KEYBOARD INTERFACE FOR THE PLD-BASED STAND**

***Abstract:** This article discusses the implementation of a standard keyboard interface on an FPGA-based educational stand designed for debugging and simulation of programmable circuits. It presents features of data input through a modified PS/2 port, part of the code description in Verilog language for the keyboard interface circuit, and its testing scheme. The obtained results confirm correct operation of the interface.*

**Keywords:** *PLD; training stand; programming; Cyclone; Intel/Altera; ASCII; Quartus II; Verilog.*

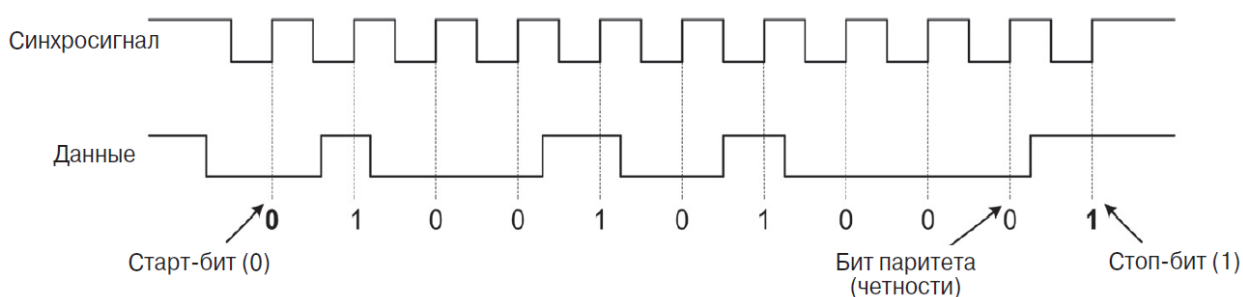
Для изучения основ проектирования электронных модулей на базе ПЛИС в российских вузах успешно применяется стенд SDK-6.1/E производства ООО «ЛМТ» (г. Санкт-Петербург) [1]. Стенд также может использоваться для создания прототипов встраиваемых систем автоматического управления и обработки сигналов, в т. ч. измерительных приборов и контроллеров. Наличие в составе стенда ПЛИС CYCLONE производства Intel/Altera и подсистемы памяти (FLASH, SRAM, EEPROM) позволяет создавать на его основе полнофункциональное вычислительное ядро. В связи с этим возникает задача подключения к стенду стандартной 104-клавишной клавиатуры. Одному из вариантов реализации интерфейса клавиатуры на ПЛИС стенда и посвящена данная работа. Особенностью предлагаемого решения является использование для ввода данных модифицированного порта PS/2.



**Рисунок 1. Логический интерфейс клавиатуры**

Как видно из рисунка 1 устройство интерфейса связано с клавиатурой с помощью двух двунаправленных сигналов [2]. При этом данные передаются в виде синхронизируемой последовательности импульсов. Нажатым клавишам ставится в соответствие 8-битный код ASCII, благодаря которому они становятся доступными для устройства обработки данных. Причем клавиатура посылает команды и коды клавиш, а устройство обработки данных – только команды.

Управление линиями данных и синхросигнала осуществляется с обеих сторон, в то время как синхронизация данных в каждом направлении обеспечивается синхросигналом клавиатуры. Когда нет передачи данных, обе линии находятся в высокоимпедансном состоянии. На рисунке 2 показана временная диаграмма передачи последовательных данных клавиатуры на примере кода 29h клавиши «пробел» [2]. Здесь же определены все 11 бит формата последовательных данных, синхронизация которых осуществляется по фронту синхросигнала.



**Рисунок 2. Передача последовательных данных клавиатуры**

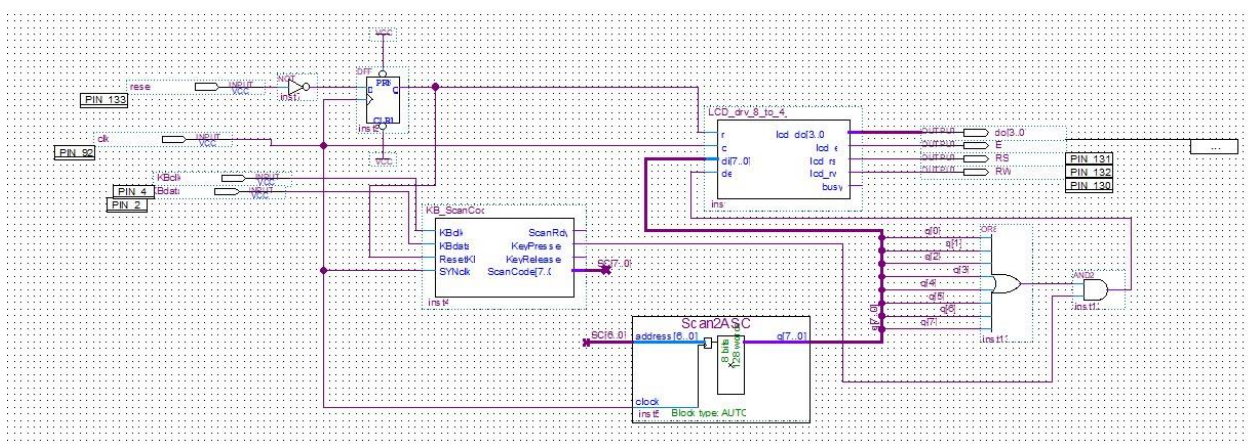
Из возможных режимов передачи данных клавиатурой описываемый логический интерфейс реализует только передачу данных к устройству их обработки (рисунок 1), а также режим ожидания в случае, если устройство блокирует линию синхросигнала.

В интерфейсе используется скан-код клавиатуры Scan Code 2, принимаемый обычно по умолчанию. Скан-коды клавиатуры содержат код нажатия клавиши Make и код отпускания клавиши Break, которые и идентифицируют состояния клавиши. Для большинства клавиш код Break имеет значение F0h, которое следует за кодом Make [2]. Соотношение кодов Make и Break дает возможность идентифицировать многократные нажатия клавиш и порядок, в котором они были нажаты.

Интерфейс читает последовательные данные из клавиатуры, когда нажимается клавиша, обнаруживает код Make и ищет этот код в таблице

преобразования ASCII. Для упрощения задачи таблица поиска обрабатывает только символы верхнего регистра.

Полный драйвер клавиатуры содержит описание на языке Verilog модуля KB\_ScanCode и память типа ПЗУ Scan2ASC для поиска кода ASCII. Схема драйвера клавиатуры с назначением выводов ПЛИС, созданная в пакете Quartus II для станда SDK-6.1/E, приведена на рисунке 3[3]. Здесь же показан драйвер жидкокристаллического (ЖК) дисплея LCD\_drv\_8\_to\_4, необходимый для тестирования интерфейса клавиатуры.



**Рисунок 3. Схема тестирования драйвера клавиатуры для станда SDK-6.1/E**

Входы KBdata и KBclk модуля KB\_ScanCode являются входами данных и синхронизации клавиатуры, а 8-битная шина ScanCode – главным выходом модуля.

Данная часть также использует быстрый сигнал синхронизации SYNclk и вход сброса клавиатуры ResetKB. Генератором сигнала SYNclk с частотой 40МГц и периодом 25нс является встроенный генератор станда SDK-6.1/E. Выходной сигнал ScanRdy модуля KB\_ScanCode подтверждает готовность скан-кода, а еще один выходной сигнал KeyRelease указывает, что нажатая клавиша была отпущена. Эти сигналы обеспечивают различие между состояниями Make и Break. Когда нажимается клавиша, на выходе ScanRdy

появляются три положительных импульса: один – для кода Make и два – для кода Break. Выход KeyRelease принимает значение 1 только один раз, когда все три кода уже переданы.

Часть кода описания схемы интерфейса показана на рисунках 4 и 5.

Память типа ПЗУ Scan2ASC, имеющая 8 линий адреса и 8-битные слова, реализуется с помощью мегафункции пакета Quartus II. Шестнадцатеричные адреса ячеек памяти ПЗУ от 0D до 66 определяются содержимым кодов ASCII для скан-кодов, которые соответствуют адресам памяти ПЗУ. Эта мегафункция определяется для использования файла инициализации памяти KbASCII.mif. Выход ScanCode подсоединяется к адресному входу памяти ПЗУ, а коды ASCII, соответствующие входным адресам, формируются на ее выходах. Реализованная как мегаблок эта поисковая память на рисунке 3 обозначена символом Scan2ASC.

```

module KB_ScanCode
(
  input KBclk,
  input KBdata,
  input ResetKB,
  input SYNclk,
  output ScanRdy, // одиночный строб при приёме любого скан-кода
  output KeyPressed, // одиночный строб, если не F0 и предыдущий не F0 (префикс «release»)
  output KeyReleased, // одиночный строб, если предыдущий F0 (клавиша отпущена)
  output reg [7:0] ScanCode
);
reg KBclk_r1, KBclk_r2; // KB Синхронизация
reg KBdata_r1, KBdata_r2;
reg [3:0] stt; // состояние: 0 - режим ожидания (ожидание «начальный бит»), 1...8 - бит
// данных, 9 - бит четности, 10 - стоповый бит
reg F0_prev; // предыдущий код F0 (префикс «релиз»)
wire KBclk_rise = !KBclk_r2 & KBclk_r1;
wire shift_en = KBclk_rise & (stt >= 1) & (stt <= 8);
wire is_F0 = (ScanCode == 8'hf0); // теперь F0 (префикс «релиз»)
wire last_ready_bit = KBclk_rise & (stt == 10) & (KBdata_r2 == 1); // is STOP_BIT == 1
assign ScanRdy = last_ready_bit;
assign KeyPressed = last_ready_bit & !F0_prev & !is_F0;
assign KeyReleased = last_ready_bit & F0_prev & !is_F0;
task reset_reg;
begin
  stt <= 0;
  F0_prev <= 0;
end
endtask
initial reset_reg();

```

#### ***Рисунок 4. Описание интерфейса клавиатуры***

Логическая схема на вентилях OR8 и AND2 (рисунок 3) блокирует нулевые коды на выходе модуля Scan2ASC, не допуская вывод на дисплей неизвестных памяти символов. Элементы DFF и NOT исключают помехи в цепи сброса клавиатуры при нажатии кнопки Reset.

Так как стенд SDK-6.1/E не имеет разъема PS/2, клавиатура подключается к стенду через разъем загрузки EEPROM автоконфигурации J1 “PRG” и порт дискретных входов/выходов J4 “EXTERNAL PIO”. Разведение

проводников клавиатуры показано в таблице 1, а подключение к стенду иллюстрирует рисунок 6.

```

always @(posedge SYNclk) begin // привязать внешние сигналы к внутренним часам
    KBclk_r1 <= KBclk;
    KBclk_r2 <= KBclk_r1;
    KBdata_r1 <= KBdata;
    KBdata_r2 <= KBdata_r1;
    if(ResetKB) begin
        reset_reg();
    end else begin
        if(shift_en) ScanCode <= {KBdata_r2, ScanCode[7:1]};
        if(KBclk_rise) begin
            case(stt)
                0: if(!KBdata_r2) stt <= 1; // если стартовый бит
                10: stt <= 0; // стоповый бит
                default: stt <= stt + 1; // следующее состояние
            endcase
            if(last_ready_bit) F0_prev <= is_F0;
        end
    end
end
endmodule

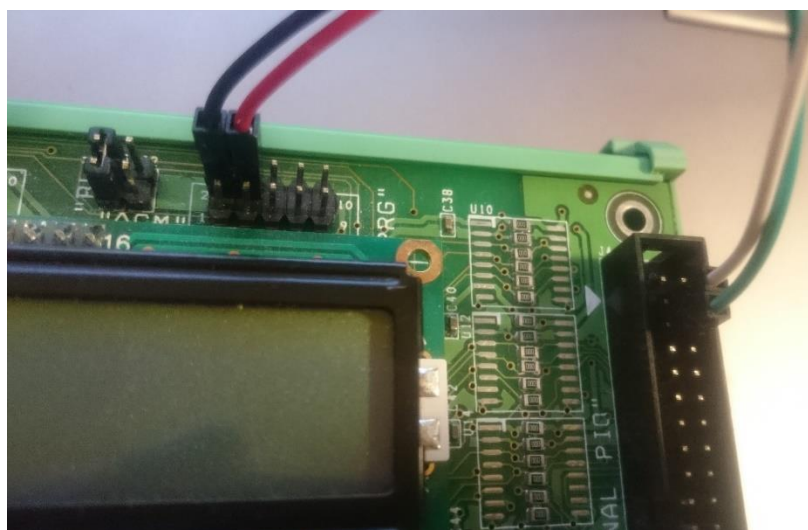
```

**Рисунок 5. Окончание описания интерфейса клавиатуры**

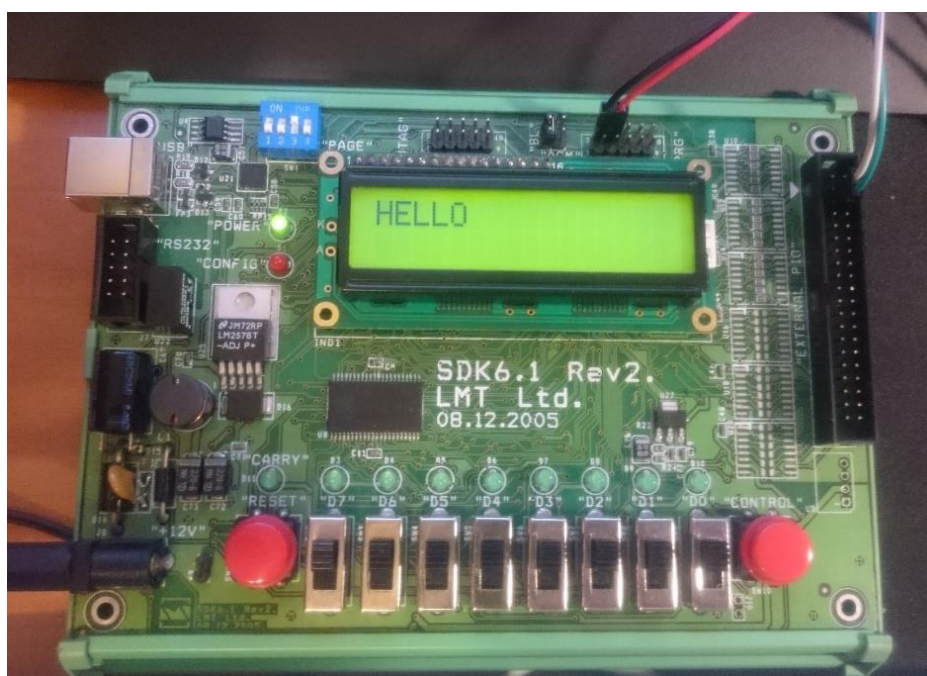
**Таблица 1.**

**Разведение проводников клавиатуры**

Наименование цепи	Цвет провода	Номер контакта PRG	Номер контакта PIO
V (VCC)	Красный	4	-
D (Data)	Белый	-	4
C (Clock)	Зеленый	-	6
G (Gnd)	Черный	2	-



*Рисунок 6. Подключение клавиатуры к стенду SDK-6.1/E*



*Рисунок 7. Демонстрация работы клавиатуры*

Работоспособность стандартной 104-клавишной клавиатуры совместно со стендом SDK-6.1/E демонстрирует рисунок 7.

#### **Используемые источники:**

1. Комплекс учебный лабораторный SDK-6.1. Руководство пользователя. – СПб.: ООО «ЛМТ», 2006.

2. Наваби З. Проектирование встраиваемых систем на ПЛИС / пер. с англ. Соловьева В. В. – М.: ДМК Пресс, 2016.

3. Болдырев А. В., Степаненко Д. Р. Особенности проектирования встраиваемых систем на ПЛИС [Электронный ресурс]: URL: [https://alley-science.ru/sovremennaya\\_nauka\\_i\\_ee\\_razvitiye\\_\\_5\\_32\\_\\_2019/](https://alley-science.ru/sovremennaya_nauka_i_ee_razvitiye__5_32__2019/)(дата обращения: 10.05.2021).

4. Принципиальная схема клавиатуры [Электронный ресурс]: URL: <https://tokzamer.ru/bez-rubriki/principialnaya-shema-klaviatury/>(дата обращения: 10.05.2021).

5. Интерфейс клавиатуры [Электронный ресурс]: URL: <https://intuit.ru/studies/courses/3460/702/lecture/14158?page=1/>(дата обращения: 10.05.2021).