

Содержание

Глава 1

Адаптеры для персонального компьютера IBM PC

1. Адаптер ввода-вывода	3
2. Адаптер сети ETHERNET	20
3. Интерфейс шины ISA на ПЛИС	35
4. Адаптер COM-порта на ПЛИС	43
5. Адаптер LPT-порта на ПЛИС	62
6. Адаптер FLASH-диска DOC2000	76
7. Адаптер USB-COM	84
8. Адаптер USB-AVR	103
9. Адаптер USB-HUB	117
10. Адаптер USB-Serial	127
11. Адаптер для связи компьютеров через интерфейс USB	145
12. Адаптер FLASH-памяти	160

Глава 2

Устройства на контроллерах

13. Контроллер 8088	168
14. Минитерминал MCS	207
15. Контроллер с дистанционной модификацией программ	249
16. Универсальные цифровые часы на контроллере AVR	281
17. Музыкальная клавиатура	309
18. Автоматический корректор часов	327
19. PIC-контроллер в автомобильных часах	346

Глава 3

Тестовые устройства и программы

20. Плата диагностики POST	353
21. Эмулятор интерфейса ISA	372
22. Эмулятор ПЗУ	375
23. Тестирование LPT-порта	386
24. Тестирование COM-порта	399
25. Автоматизированный фильтр-удлинитель сетевого питания	404

Приложения 409

Приложение 1	
Система команд микроконтроллеров MCS-51	409
Приложение 2	
Система команд микроконтроллеров AVR	411
Приложение 3	
Система команд PIC микроконтроллеров семейства PIC16F84	414
Приложение 4	
Описание файлов к книге	415

Глава 1

Адаптеры для персонального компьютера IBM PC

1. Адаптер ввода-вывода

В этом разделе вниманию читателей предлагается устройство под названием «Адаптер ввода-вывода», который подключается к компьютеру через интерфейс ISA и имеет в своем составе регистры ввода и вывода, предназначенные для работы с различными периферийными устройствами (датчиками контроля, кнопками, индикаторами, исполнительными устройствами и т. п.).

Данный адаптер позволит познакомиться с принципами построения устройств расширения персонального компьютера через интерфейс ISA и повторить, при желании, эту конструкцию для расширения возможностей компьютера.

Адаптер позволяет подключить до 32-х устройств ввода и до 24-х устройств вывода. Например, при решении задачи охраны помещения, данный адаптер позволяет контролировать до 32-х устройств контроля (датчики открытия окон и дверей, кнопки пульта управления и т.д.) и управлять 24-я исполнительными устройствами (сиреной, аварийной лампой, электрозамком и пр.).

Несмотря на то что шина ISA постепенно вытесняется в современных компьютерах более производительной шиной PCI, она по-прежнему является шиной промышленного стандарта и зарекомендовала себя как надежная и недорогая шина. Большинство промышленных компьютеров имеет в своем составе шину ISA или ее аналог — PC104. Существуют также устройства расширения шины ISA, позволяющие совместно работать системам, включающим в себя от двух до двадцати устройств с шиной ISA. Рассмотрим структурную схему адаптера, приведенную на **Рис. 1.1**.

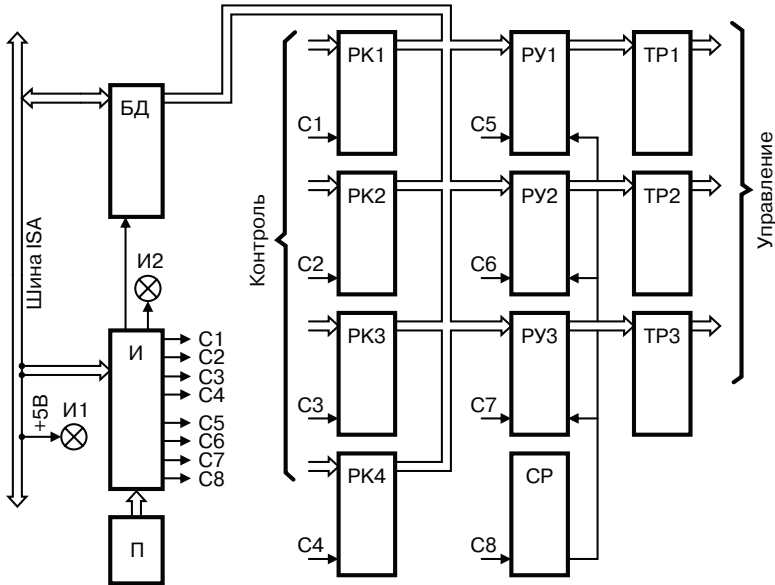


Рис. 1.1. Структурная схема адаптера

В состав адаптера входят следующие узлы:

И — интерфейс;

СР — схема разрешения;

БД — буфер данных;

ПК1...4 — регистры контроля;

П — поле перемычек;

РУ1...3 — регистры управления;

И1 — индикатор питания;

ТР1...3 — твердотельные реле.

И2 — индикатор обращения;

При включении адаптера схема разрешения (СР) по сигналу сброса от шины ISA принудительно переходит в состояние запрета выходов регистров управления (РУ), и последние переводят выходы управления в третье состояние, что обеспечивает выключение всех исполнительных устройств в начальный момент. Индикатор (И1) индицирует подачу питания на адаптер.

Поле перемычек «П» определяет базовый адрес модуля, при обращении к которому адаптер будет отзываться.

При совпадении адреса устройства ввода-вывода на шине ISA с базовым адресом модуля интерфейс И зажигает индикатор И2 и вырабатывает один из синхросигналов С1—С8, которые управляют чтением и записью регистров контроля (РК) и регистров управления (РУ).

Схема разрешения СР активизирует выходы регистров управления (РУ) и включает устройства управления, в управляющие разряды которых были записаны логические нули.

Твердотельные реле (ТР) обеспечивают согласование логических выходов регистров управления (РУ) с исполнительными устройствами.

Принципиальная электрическая схема адаптера приведена на **Рис. 1.2**.

Интерфейс И выполнен на микросхемах D2, D3, D4, D5.4 и наборе резисторов RN12. В качестве буфера данных используется микросхема D1 типа КР1533АП6 и набор резисторов RN11. Поле перемычек выполнено на элементе J1. Индикаторы построены на базе элементов HL1, HL2 и R1, R2. Схема разрешения (СР) выполнена на элементах D5.1—D5.3. Функции регистров контроля выполняют микросхемы D9—D12 и наборы резисторов RN1—RN4. Функции регистров управления выполняют микросхемы D6—D8 и наборы резисторов RN5—RN10. В качестве твердотельных реле применены элементы U1—U12.

В адаптеры были использованы следующие элементы: С1 — электролитический конденсатор типа К50-35-47 мкФ-6,3 В, С2—С25 — керамические конденсаторы типа К10-17Б-Н90-0,1 мкФ, микросхемы D1 — типа КР1533АП6, D2 — SN74НС688, D3 — КР1533АП5, D4 — КР1533ИД4, D5 — КР1533ЛА3, D6—D8 — КР1533ИР23, D9—D12 — КР1533ИР22, HL1, HL2 — светодиоды типа АЛС307АМ, J1 — любые перемычки или DIP-переключатель, наборы резисторов RN1—RN4, RN11, RN12 — типа НР1-4-9М-0,125-10 кОм, а RN5—RN10 — типа НР1-4-9М-0,125-510 Ом, R1, R2 — резисторы типа С2-23-0,125-1 кОм, твердотельные реле U1—U12 — типа 5П14.3Б, X2—X8 — соединители типа ВН-16. Вместо микросхем серии КР1533 можно применить микросхемы серии К555 или их зарубежные аналоги.

Допускается устанавливать не все регистры контроля и управления и связанные с ними элементы, а только количество, необходимое для решения поставленной задачи.

Адрес адаптера определяется с помощью установки перемычек J1. Обозначение контактов J1 приведено на **Рис. 1.3**.

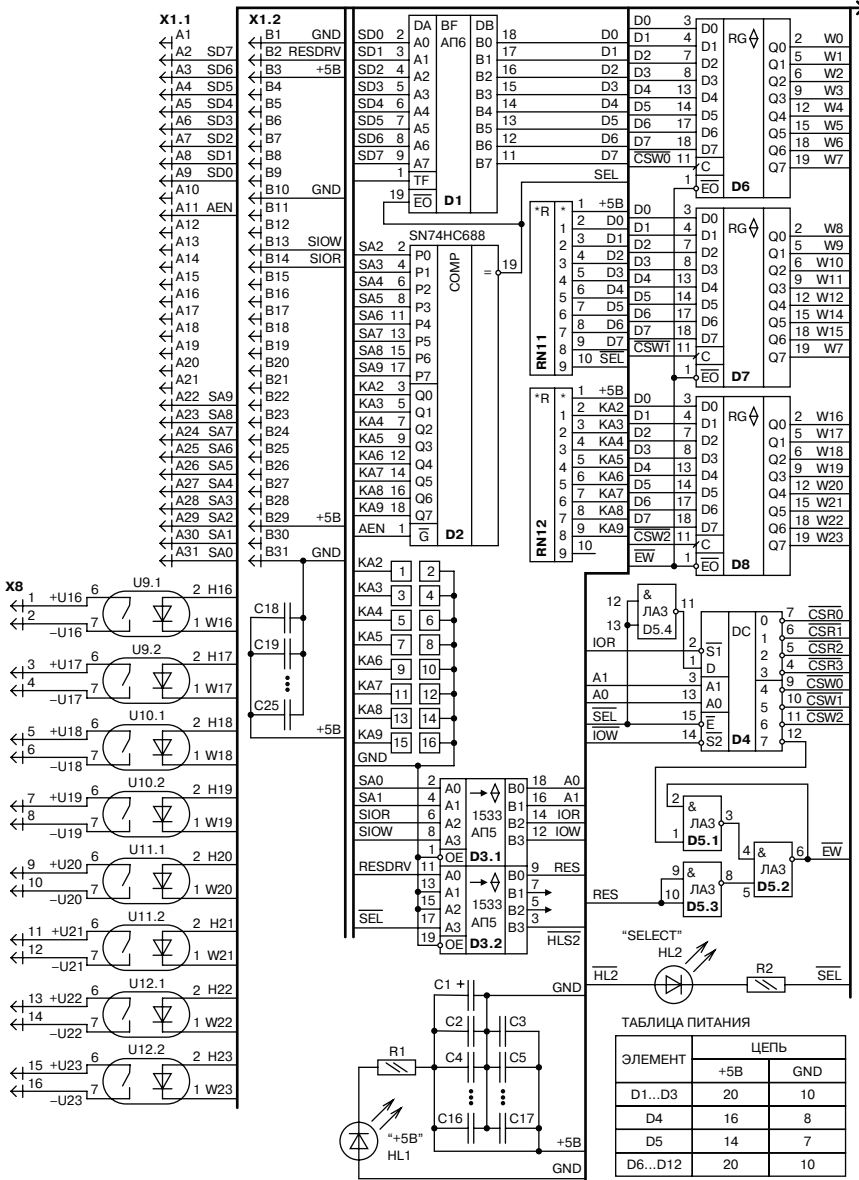


Рис. 1.2. Принципиальная электрическая схема адаптера

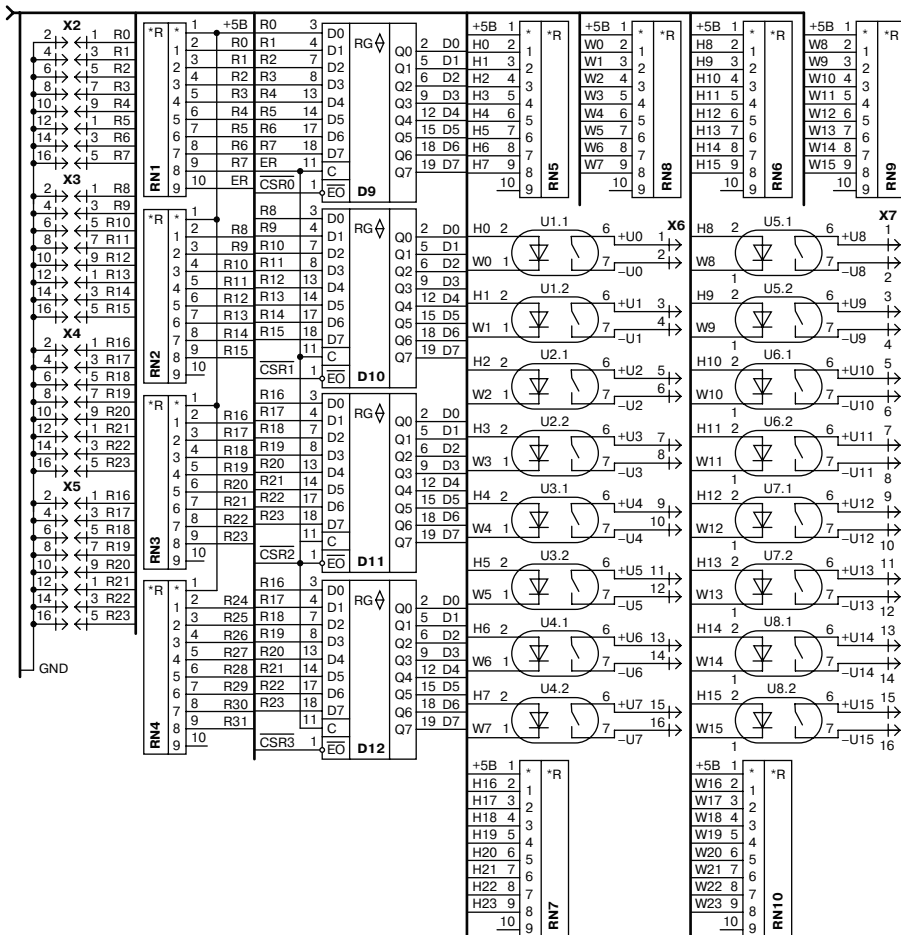


Рис. 1.2. Принципиальная электрическая схема адаптера (продолжение)

Пример установки базового адреса адаптера приведен в **Табл. 1.1**, программно-логическая модель адаптера в **Табл. 1.2**.

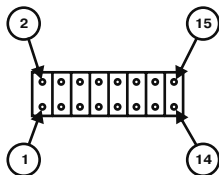


Рис. 1.3. Обозначение контактов J1

Таблица 1.1. Пример установки базового адреса адаптера

Адресные разряды	9	8	7	6	5	4	3	2	1	0	
Группа контактов	15 16	13 14	11 12	9 10	7 8	5 6	3 4	1 2	–	–	
Базовый адрес	1	0	1	1	1	1	0	0	0	0	2F0h
Состояние группы	P	3	P	P	P	P	3	3	–	–	
Базовый адрес	1	1	0	0	0	0	0	0	0	0	300h
Состояние группы	P	P	3	3	3	3	3	3	–	–	

Примечание. P — разомкнутые контакты, 3 — замкнутые контакты.

Таблица 1.2. Программно-логическая модель адаптера

Адрес порта	Операция	Назначение устройства
БА+0	Чтение	Регистр контроля 0
БА+1	Чтение	Регистр контроля 1
БА+2	Чтение	Регистр контроля 2
БА+3	Чтение	Регистр контроля 3
БА+0	Запись	Регистр управления 0
БА+1	Запись	Регистр управления 1
БА+2	Запись	Регистр управления 2
БА+3	Запись	Схема разрешения (запись любого значения по этому адресу переводит выходы всех регистров управления из третьего состояния в активное)

Примечание. БА (базовый адрес) адаптера выбирается пользователем из области свободных адресов компьютера и устанавливается перемычками на адаптере. Например, при выборе адреса 2F0h необходимо установить перемычки J1: 1-2, 3-4, 13-14.

Конструктивно адаптер выполнен на одной печатной плате с печатным соединителем X1 и шторкой-заглушкой. Внешний вид адаптера приведен на **Рис. 1.4**. Адаптер подключается к шине ISA через свободный слот и крепится к корпусу компьютера с помощью шторки. Топология печатной платы была получена в конструкторском пакете PCAD 4.5. Файл топологии находится на компакт-диске, прилагаемом к этой книге.

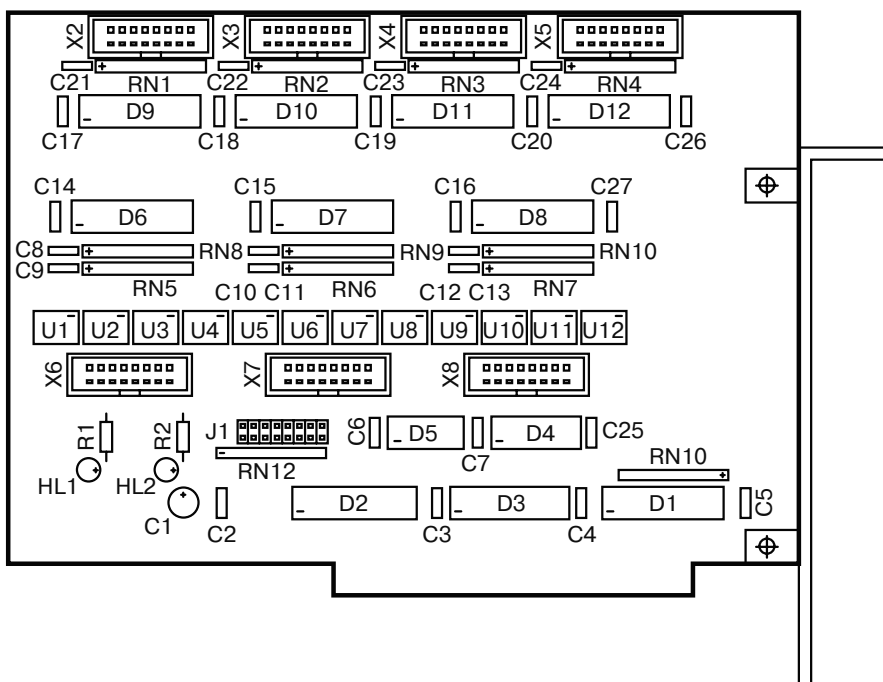


Рис. 1.4. Внешний вид адаптера

Для того чтобы не повредить компьютер неисправным адаптером, первоначальную отладку адаптера можно выполнить с помощью несложного эмулятора интерфейса ISA, описываемого в соответствующем разделе этой книги. После того, как адаптер заработает, его можно подключать к шине компьютера.

Для тестирования данного адаптера в составе компьютера автором книги был написан тест «test_avv» на языке программирования Си. Эта тестовая программа позволяет проверить работоспособность адаптера в составе компьютера и, кроме того, протестировать все подключенные к нему устройства. Программа запускается в режиме DOS и отображает на экране монитора состояние всех регистров адаптера. Окно работы программы приведено на **Рис. 1.5**. Справка по управлению программой выводится на самом экране.

```

Тест адаптера ввода вывода [Базов.адр. = 2F0h]

                З-А-П-И-С-Ь
Регистр RU0 (D6)  Регистр RU1 (D7)  Регистр RU2 (D8)  Регистр RU3 (D5)
Р а з р я д ы    Р а з р я д ы    Р а з р я д ы    ВКЛЮЧЕНИЯ ВЫХОДОВ
7-6-5-4-3-2-1-0  7-6-5-4-3-2-1-0  7-6-5-4-3-2-1-0  1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  Выходы выключ.

                Ч-Т-Е-Н-И-Е
Регистр RK0 (D9)  Регистр RK1 (D10)  Регистр RK2 (D11)  Регистр RK3 (D12)
Р а з р я д ы    Р а з р я д ы    Р а з р я д ы    Р а з р я д ы
7-6-5-4-3-2-1-0  7-6-5-4-3-2-1-0  7-6-5-4-3-2-1-0  7-6-5-4-3-2-1-0
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1

                П о м о щ ь :
<Esc>: выход <TAB>: циклич.переход м/д регистрами <Enter>: включение выходов
<Space>: dx=00 <F5>: dx=55h <F6>: dx=AAh <F7>: dx=FFh <1>...<8>: инверсия бит

```

Рис. 1.5. Окно работы программы

Полный исходный текст программы с комментариями приводится ниже. Он может послужить заготовкой для написания другой программы, работающей с адаптером при решении различных прикладных задач. Исполняемый код программы и текст программы на языке Си также записаны на компакт-диске, прилагаемом к этой книге.

Текст программы «test_avv»

```
/* =====*/
/* Программа «test_avv» тестирует адаптер ввода-вывода */
/* Режим работы программы:DOS */
/* Автор: Вальпа Олег */
/* Дата: 25.12.2001 */
/* =====*/

#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include <conio.h>

/* Скан коды клавиш */
#define ESC 0x011b
#define ENTER 0x1c0d
#define UP 0x4800
#define DOWN 0x5000
#define LEFT 0x4B00
#define RIGHT 0x4D00
#define TAB 0x0f09
#define SPACE 0x3920

#define K1 0x0231
#define K2 0x0332
#define K3 0x0433
#define K4 0x0534
#define K5 0x0635
#define K6 0x0736
#define K7 0x0837
#define K8 0x0938
#define K9 0x0a39
#define K0 0x0b30

#define KF1 0x3b00
#define KF2 0x3c00
#define KF3 0x3d00
#define KF4 0x3e00
#define KF5 0x3f00
#define KF6 0x4000
#define KF7 0x4100
#define KF8 0x4200
#define KF9 0x4300
#define KF10 0x4400

/* Координаты сообщений */
#define X 1
#define Y 1
```



```

printf («                                \r\n»);
printf («                Ч Т Е Н И Е                \r\n»);
printf (« Регистр RK0 (D9) Регистр RK1(D10) Регистр RK2(D11) Регистр RK3(D12) \r\n»);
printf (« Р а з р я д ы   Р а з р я д ы   Р а з р я д ы   Р а з р я д ы   \r\n»);
printf (« 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 \r\n»);
printf (« 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 \r\n»);
printf («                                \r\n»);
printf («                                \r\n»);
printf («                                \r\n»);
printf («                                \r\n»);
textcolor(YELLOW);
printf («                П о м о щ ь:                \r\n»);
printf (« <Esc>:выход <TAB>: циклич.переход м/д регистрами <Enter>:включение выходов \r\n»);
printf (« <Space>:dx=00 <F5>:dx=55h <F6>:dx=AAh <F7>:dx=FFh <1>...<8>:инверсия бит \r\n»);
printf («                                \r\n»);

textcolor(WHITE);
gotoxy (xr[mode] - 1, yr[mode] - 1);
printf (« 7 6 5 4 3 2 1 0 <<»);
gotoxy (xr[mode] - 1, yr[mode]);
printf (« <<»);
gotoxy (xr[mode] + 15, yr[mode]);
printf (« <<»);
gotoxy (xr[mode] - 1, yr[mode]+1);
printf (« <<»);

/*===== Инициализация портов и счетчиков =====*/

datrg[0] = 0xff;
datrg[1] = 0xff;
datrg[2] = 0xff;

outportb ( RU0 , datrg[0] ); /* Установить регистры в пассивное сост. */
outportb ( RU1 , datrg[1] ); /* Установить регистры в пассивное сост. */
outportb ( RU2 , datrg[2] ); /* Установить регистры в пассивное сост. */

/*===== Начало опросов и вывода =====*/
while(1)
{
if( bioskey(1) != 0 ) klav=bioskey(0);
else klav=0;

if(klav != 0)
{

if(klav==ESC)
{
/* putchar ( 0x07); */
setkursor ();
break;
}
}
}

```

```
if(klav==ENTER)
{
    textcolor(LIGHTRED);
    gotoxy (xr[3], yr[3]);
    cprintf («Выходы включены»);
    outportb ( RU3 , 0 );
}

if(klav==TAB)
{

    textcolor(LIGHTGREEN);
    gotoxy (xr[mode] - 1, yr[mode] - 1);
    cprintf (« 7 6 5 4 3 2 1 0 <<»);
    gotoxy (xr[mode] - 1, yr[mode]);
    cprintf (« <<»);
    gotoxy (xr[mode]+15, yr[mode]);
        cprintf (« <<»);
    gotoxy (xr[mode] - 1, yr[mode]+1);
    cprintf («          <<»);

    switch(mode)
    {
        case 0:
            mode = 1;
            adr_rg = RU1;
            break;
        case 1:
            mode = 2;
            adr_rg = RU2;
            break;
        case 2:
            mode = 0;
            adr_rg = RU3;
            break;
        default:
            break;
    }

    textcolor(WHITE);
    gotoxy (xr[mode] - 1, yr[mode] - 1);
    cprintf (« 7 6 5 4 3 2 1 0 <<»);
    gotoxy (xr[mode] - 1, yr[mode]);
    cprintf (« <<»);
    gotoxy (xr[mode]+15, yr[mode]);
        cprintf (« <<»);
    gotoxy (xr[mode] - 1, yr[mode]+1);
    cprintf («          <<»);

}

if (mode <= 2)
```

```
{
switch(klav)
{
case K1:
    datrg[mode]= datrg[mode] ^ 0x80 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K2:
    datrg[mode]= datrg[mode] ^ 0x40 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K3:
    datrg[mode]= datrg[mode] ^ 0x20 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K4:
    datrg[mode]= datrg[mode] ^ 0x10 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K5:
    datrg[mode]= datrg[mode] ^ 0x08 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K6:
    datrg[mode]= datrg[mode] ^ 0x04 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K7:
    datrg[mode]= datrg[mode] ^ 0x02 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case K8:
    datrg[mode]= datrg[mode] ^ 0x01 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case KF5:
    datrg[mode]= 0x55 ;
    outportb ( adr_rg , datrg[mode] );
    break;
case KF6:
    datrg[mode]= 0xAA ;
    outportb ( adr_rg , datrg[mode] );
    break;
case KF7:
    datrg[mode]= 0xFF ;
    outportb ( adr_rg , datrg[mode] );
    break;
case SPACE:
    datrg[mode]= 0x00 ;
    outportb ( adr_rg , datrg[mode] );
    break;
default:
```

```

        break;
    }
    for (i=0; i<=7; i++)
    {
        stp = 1;
        for (j=0; j<i; j++) stp = stp * 2;
        gotoxy (xr[mode] - i*2+14,yr[mode]);
        if ((datrg[mode] & stp) == 0) {textcolor(LIGHTRED); cprintf («0»);}
        else {textcolor(LIGHTBLUE); cprintf («1»);}
    }
}
textcolor(YELLOW);
textcolor(LIGHTCYAN);
continue;
}
/*===== Чтение регистров RK0-3 =====*/
data = inportb ( RK0 );
for (i=0; i<=7; i++)
{
    stp = 1;
    for (j=0; j<i; j++) stp = stp * 2;
    gotoxy (xr[ 4] - i*2+14,yr[ 4]);
    if ((data & stp) == 0) {textcolor(LIGHTRED); cprintf («0»);}
    else {textcolor(LIGHTBLUE); cprintf («1»);}
}

data = inportb ( RK1 );
for (i=0; i<=7; i++)
{
    stp = 1;
    for (j=0; j<i; j++) stp = stp * 2;
    gotoxy (xr[ 5] - i*2+14,yr[ 5]);
    if ((data & stp) == 0) {textcolor(LIGHTRED); cprintf («0»);}
    else {textcolor(LIGHTBLUE); cprintf («1»);}
}

data = inportb ( RK2 );
for (i=0; i<=7; i++)
{
    stp = 1;
    for (j=0; j<i; j++) stp = stp * 2;
    gotoxy (xr[ 6] - i*2+14,yr[ 6]);
    if ((data & stp) == 0) {textcolor(LIGHTRED); cprintf («0»);}
    else {textcolor(LIGHTBLUE); cprintf («1»);}
}

data = inportb ( RK3 );
for (i=0; i<=7; i++)
{
    stp = 1;
    for (j=0; j<i; j++) stp = stp * 2;
    gotoxy (xr[ 7] - i*2+14,yr[ 7]);

```