Министерство Российской Федерации по атомной энергии

НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ
имени А.П. Александрова

УТВЕРЖДАЮ
Директор НИТИ
В.А. Василенко
___.___ . 2000

Тренажёр “Диана-Барс”
ЭВМ БЭСМ-6

Эмулятор накопителей на магнитных дисках

Программное обеспечение

Техническое описание
99.213.70.000 ТО

Начальник отделения 4
А.И. Колесников
___.___ . 2000

Начальник отдела 44
Ю.А. Морозов
___.___ . 2000

Начальник группы 43.02.2
В.И. Иконников
___.___ . 2000

Ведущий инженер отдела 42
В.В. Маличев
___.___ . 2000

Сосновый Бор
2000

Содержание

1 Введение

В соответствии с техническим предложением "Эмулятор накопителей на магнитных дисках ЭВМ БЭСМ-6 комплексного тренажёра Диана-Барс" № 4-08/43 от 24.03.2000 разработаны, реализованы и сформированы два комплекта аппаратно-программных средств - "Эмуляторов НМД" (ЭНМД-3 и ЭНМД-4) на базе двух ПЭВМ PC/AT-486 с параметрами:

− Тактовая частота - 66 МГц,
− Объём оперативной памяти - 16 Мбайт,
− Объём памяти на "жёстком" диске (винчестере) - 540 Мбайт,
− Объём памяти на "гибком" диске - 1,44 Мбайт,
− Монитор - 15".

Для обеспечения высокой степени сохранности информации каждая ПЭВМ доукомплектована вторым съёмным винчестером ёмкостью 540 Мбайт.

"Эмулятор НМД" на базе одной из таких ПЭВМ образует для ЭВМ БЭСМ-6 поле накопителей дисковой памяти, эквивалентное ёмкости комплекта из 8-ми типовых пакетов магнитных дисков НМД и трём резервным копиям каждого комплекта.

Описание специальных аппаратных средств, разработанных и реализованных с целью подключения двух ПЭВМ PC/AT-486 к ЭВМ БЭСМ-6 приведены в документах:

− "Средства сопряжения с Эмулятором НМД" 99.213.71.000 ТО,
− "Интерфейсный модуль МС1304" 99.213.72.000 РЭ.

В настоящем документе приведено описание специального программного обеспечения, разработанного и включённого в состав системных программных средств ПЭВМ PC/AT-486 с целью реализации функционирования ЭНМД вместо типовых комплектов НМД ЭВМ БЭСМ-6 тренажёра "Диана-Барс".

Системное математическое обеспечение ЭВМ БЭСМ-6 не требует внесения изменений или дополнений при использовании "Эмулятора НМД" вместо типовых комплектов НМД. При этом никаких ограничений на функционирование ЭВМ БЭСМ-6 не накладывается.

2 Принципы размещения и формирования информации в ЭНМД

В качестве основных носителей информации в ЭНМД используются области памяти на "жёстком" диске (винчестере) ПЭВМ PC/AT-486. В поле памяти винчестера программно формируются 8 виртуальных дисковых устройств, каждому из которых соотнесён определённый "файловый пакет" (или его копия), эквивалентный по содержанию информации типового пакета магнитных дисков НМД со своим номером.

Один файл содержит информацию одной зоны физического пакета и имеет имя, соответствующее номеру физической зоны.

Для реализации обращения к выбранному виртуальному устройству, выбранному "файловому пакету" и выбранному файлу - эти участки информационного поля винчестера именуются посредством специальной операции (разметка).

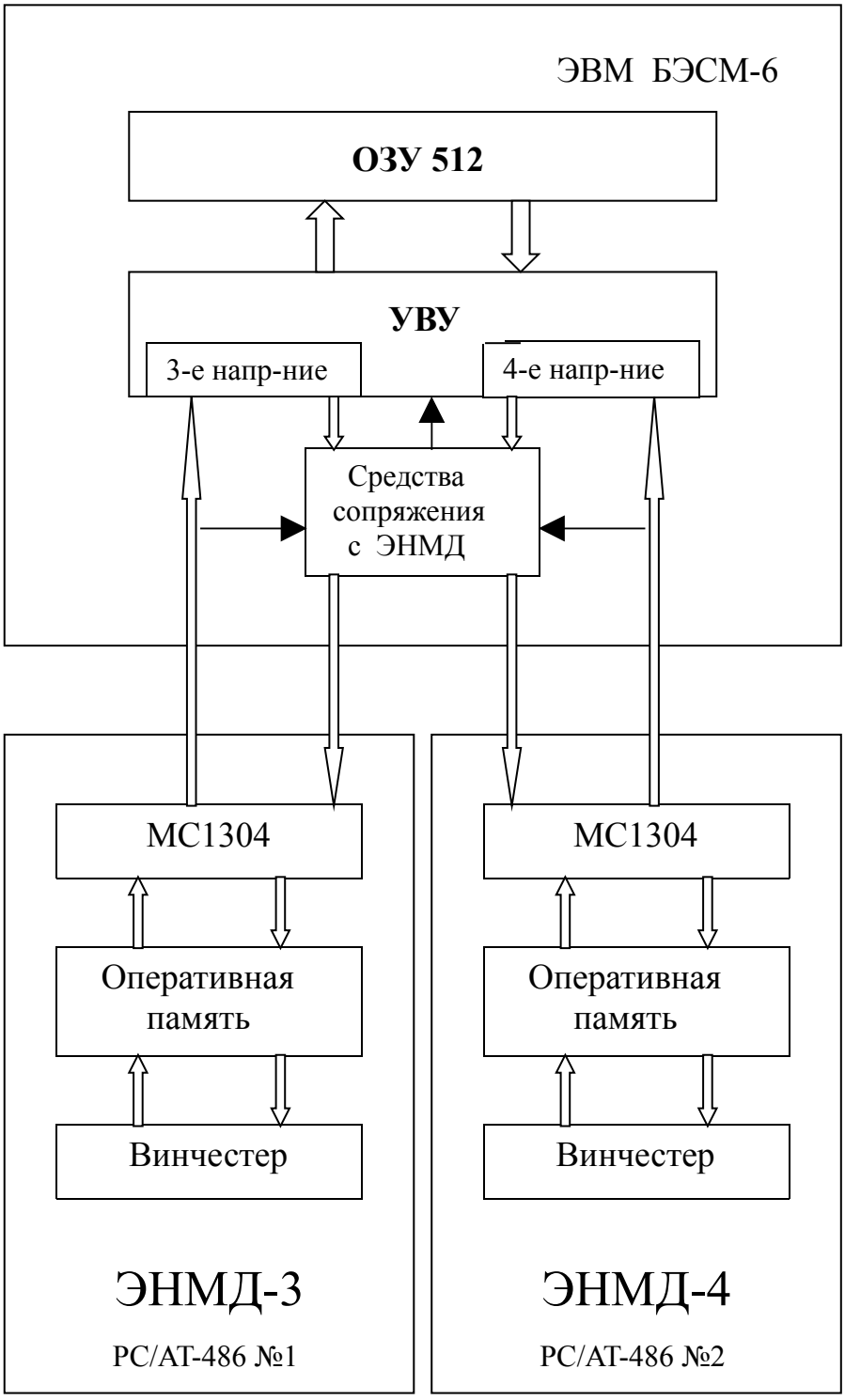

Рис 1 Блок-схема обмена информацией между ЭВМ БЭСМ-6 и ЭНМД 3-го и 4-го направлений

3 Последовательность перемещения информации при обмене между ЭВМ БЭСМ-6 и ЭНМД

Блок-схема обмена информацией между ЭВМ БЭСМ-6 и 3-го направления (ЭНМД-3) и 4-го направления (ЭНМД-4) приведена на рис.1.

3.1 Передача информации из ЭНМД в БЭСМ-6

В процессе выполнения обмена по "записи" в ОЗУ ЭВМ БЭСМ-6 заданный файл (или половина файла) из выбранного пакета "готового" виртуального устройства на винчестере ПЭВМ сначала переписывается в буфер обмена оперативной памяти (ОП) ПЭВМ, а затем передаётся через модуль связи МС1304 в соответствующее (3-е или 4-е) направление УВУ в заданный лист (половину листа) ОЗУ ЭВМ БЭСМ-6.

3.2 Передача информации из БЭСМ-6 в ЭНМД

В режиме "чтения" информация из заданного листа (половины листа) ОЗУ ЭВМ БЭСМ-6 передаётся через соответствующие направления УВУ, БСУ и МС1304 в буфер обмена ОП ПЭВМ и далее - в заданный файл выбранного файлового пакета "готового" виртуального устройства на винчестере ПЭВМ.

50-разрядное слово ЭВМ БЭСМ-6 при обмене с ПЭВМ размещается в пяти шестнадцатиразрядных ячейках памяти ПЭВМ (занимая младшие 10 разрядов каждой ячейки).

4 Структура ПО и его составных частей

4.1 Головная программа

Головная программа рабочего варианта ЭНМД выполняет следующие функции:

− Приём команд из ЭВМ БЭСМ-6 через входные порты модуля МС1304.

− Дешифрация принятых команд с запоминанием кода заданной операции.

− Выполнение простых операций: запрос, выбор модуля, опрос регистра состояния, гашение регистра состояния, подвод к заданной зоне, сброс на 0-й цилиндр и освобождение ЭНМД по окончании сеанса связи с ЭВМ БЭСМ-6.

− Для выполнения операций обмена: запись, чтение и сравнение - вызывается соответствующий драйвер и передаётся ему управление на время сеанса обмена одной зоной между БЭСМ-6 и ЭНМД.

− Выдача ответных асинхронных сигналов об окончании выполнения операции в выходные порты модуля МС1304. Первый из них, названный "clnPKA", даёт разрешение процессору ЭВМ БЭСМ-6 на выполнение следующей команды. Второй сигнал, названный "setPRER", сигнализирует об окончании операции обмена.

Процессор ПЭВМ при функционировании ЭНМД в комплексе ЭВМ БЭСМ-6 призван выполнять две программные задачи:

− управление процессами диалога и обмена информацией с ЭВМ БЭСМ-6;

− управление процессами отображения на экране монитора ПЭВМ.

Первая задача имеет безусловный приоритет, в противном случае возможны миллисекундные задержки реакции ПЭВМ на обращения со стороны ЭВМ БЭСМ-6 в диалоговом режиме работы и, как следствие, недопустимые задержки в обработке собственных временных прерываний ЭВМ БЭСМ-6. Поэтому предусмотрено, что задача управления процессами отображения на экране монитора инициируется только в промежутке времени, когда системное обеспечение ЭВМ БЭСМ-6 заведомо не формирует команды обращения к ЭНМД:

− от момента отработки запроса на "захват" ЭНМД до формирования в ЭНМД сигнала прерывания, направляемого ЭВМ БЭСМ-6 в качестве разрешения на проведение сеанса диалога ЭНМД;

− от момента получения исполнительной команды задания на обмен до момента фактического начала обмена массивом информации через УВУ БЭСМ-6;

− от момента окончания фактического обмена массивом информации до формирования в ЭНМД сигнала прерывания "Конец обмена".

4.2 Драйверы обмена

Драйверы записи, чтения и сравнения выполняют обмен слогами между ОП ПЭВМ и ОЗУ ЭВМ БЭСМ-6 по схеме, описанной в разделе 3, синхронно с тактирующими сигналами из стойки УВУ через порты модуля МС1304.

Период следования синхронизирующих сигналов, программно формирующихся ПЭВМ и поступающих УВУ ЭВМ БЭСМ-6 при обмене массивами информации с ЭНМД, составляет 11мксек (с типовыми устройствами НМД - 10 мксек).

Среднее время ожидания окончания обмена ЭВМ БЭСМ-6 с ЭНМД составляет 75 мсек (с типовыми устройствами НМД - 200 мсек).

Общий выигрыш во времени ожидания окончания обмена при использовании ЭНМД получается за счёт меньших затрат времени на подвод головок записи/чтения винчестера ПЭВМ к заданной дорожке (зоне) и на их подход к её началу по сравнению со значениями этого типа затрат при использовании типовых устройств НМД.

4.3 Программы автономного режима

Головная программа автономного режима, фиксируя факт нажатия клавиши на клавиатуре ПЭВМ, включает в работу программный дешифратор принятого кода. Если была нажата клавиша <F1>, на экране ПЭВМ появляется меню в виде списка функций, выполняемых в автономном режиме:

− разметка пакета,

− перепись зон с пакета на пакет,

− просмотр зоны в различных форматах,

− роспись зоны заданным кодом.

С помощью клавиш управления курсором (<↓>, <↑>) выбирается нужная строка меню и нажимается клавиша <Enter>. Далее работа продолжается в режиме диалога, по окончании которого программа вызывает соответствующую функцию (подпрограмму) и передает ей управление. После выполнения задания функция возвращает управление головной программе. Для выполнения следующего задания надо нажать клавишу <F1>.

5 Порядок работы с ЭНМД

5.1 Комплексный режим

Включение ПЭВМ с ЭНМД-3 или ЭНМД-4 производится только после подачи питания на ЭВМ БЭСМ-6. Процесс вызова операционной системы MS DOS заканчивается вызовом в работу исполнительного файла ЭНМД, который вызывает появление на экране монитора условного изображения виртуальных устройств с указанием номеров "установленных" на них пакетов. После этого начинается работа с пульта ЭВМ БЭСМ-6 с полем дисковой памяти данного ЭНМД.

Выключение ПЭВМ с ЭНМД следует производить после выключения питания ЭВМ БЭСМ-6.

Поскольку на обоих ЭНМД "установлены" одинаковые виртуальные пакеты, то в работу всегда отдаётся только одна из ПЭВМ: с ЭНМД-3 или с ЭНМД-4.

Запускаемая программа EMD_W.EXE

5.2 Автономный режим

Для перехода в автономный режим работы ЭНМД надо не менее 8-ми секунд удерживать нажатой кнопку "УСТ 0 ОБЩ" на пульте УУ ЭВМ БЭСМ-6. Это время подобрано экспериментально для того, чтобы принудительно завершить работу ПО ЭНМД и выйти в режим работы с панелями "Norton Commander". После этого на клавиатуре ПЭВМ надо нажать клавишу <F2>. Из появившегося на экране меню следует выбрать строку с названием "Автономный вариант ЭНМД" и нажать клавишу <Enter>. Далее - продолжать работу согласно с указаниями подраздела 4.3.

Запускаемая программа EMD_A.EXE

5.3 Работа с тестом УВУ

После включения ПЭВМ с ЭНМД согласно указаниям в подразделе 5.1 можно переходить к работе с тестом УВУ по стандартной схеме - как со стойкой КМД (см. заводскую инструкцию по работе с тестом УВУ).

Приложение А

(обязательное)

Тексты программ ЭНМД

(на языке Microsoft C)

```
/*г=================================================================¬
    "emd.c"   -  Emulator of MD for BESM-6
     Release  -  22.08'99 ... 19.07'2000
  L=================================================================-*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <bios.h>
#include <time.h>
#include <conio.h>
#include <graph.h>
#include <ctype.h>

#include "emd.h"

void main( int n, char *v[], char **envp)
{
 UI i, nf=0;
 UI rsya=0, rsoa=0; // added to rsy, rso
 char TST[16] = "Режим теста УВУ";
   /* Выборка параметра - имя exe/файла */
   strcpy(cpfn,v[0]);  // "Exe_File_Name"
   if(cpfn[0]=='D') l1[0]='E';
/* man(); */ if(krdf_e(1)) { printf("\n\t Excuse me please...");
         _getch(); goto E; }
   mdcheck(); // опознание пакетов на устройствах
   drtime(7); // '7' - font size
 for(;;) {
 G: // Ожидание <ПКА>:
  pus = _inpw(P_R2); GM = getGM;
  if(GM) // ГМ - 5-th bit from Ikonnikov
   { clnPKA;   // гашение <ПКА>
     clnHBSL;  // гашение <ХБСл>
     lin=pus=portinp=dir=pmd=mod=kom=0; drtime(7);
     if(g) { g=nf=0; GD=r49; mdcheck(); } // Гашение тестового режима
     nf++; if(nf>1000) goto E; // 26.05'2000
           else        goto G;
   }
 if(getPKA) { portinp = getKUS; // <~KUSj> (1-12) bits
  if(!dir) { if((portinp&077)==050) { clnPKA; goto G; }
             else // if(portinp&0400) // запрос ?
              { dir = 1; clnPKA; nf = 0; drtime(7);
scan = pimp = 0; // Проверка клавиатуры:
  if(_kbhit()) { pimp = _getch(); // Service ?
  if(pimp==0x00||pimp==0xe0) { pimp=_getch(); scan=1; }
               } else goto P;
   if(!scan) {
   if(pimp=='U'||pimp=='u') { // if(lin) goto P;
 B: CDBW1; stp(20,34); // Включение/Выключение устройства
  _outtext("Задайте N линейки (0,1): ");
   scanf("%u",&lin);     if(lin>1) { _outtext("Ошибка: N > 1");
  _outtext(ent); _getch(); goto B; } stp(20,36);
  _outtext("Задайте N устройства (0-7): ");
   scanf("%u",&mod);     if(mod>7) { _outtext("Ошибка: N > 7");
  _outtext(ent); _getch(); goto B; }
        unit(); CDBW1;
      if(!opros) mdcheck(); } // опознание пакетов
            } // end of "if(!scan)"
P:   if((portinp&07400)==01400) lin = portinp&03;
     setPRER; } // "if(portinp&0400)"
   goto G; } // "if(!dir)"
```

```
else { // "if(dir)"
       // опрос РС (1-12)р.р.:
  if(portinp==011)
   { if(!lin) { if(pmd&rsy) setB(pmd|GD|rsya); else setB(rsya); }
     else { if(pmd&rsy1) setB(pmd|GD|rsya); else setB(rsya); } }
       // опрос РС (13-24)р.р.:
  if(portinp==031) setB(rso|rsoa);
  if(portinp==014) { // Режим т.УВУ
      setIM; orsy1=orsy=rsy1=rsy=GD=0;
       if(!g) { g=1; dr_un(0);
     _setcolor(12); _moveto(64*8,5*8); _outgtext(TST);
              }    }
        clnPKA; // гашение <ПКА>
 switch(portinp&06000) // bits 12,11
  {
   case 02000:  // гашение РМ либо выбор ...
                // номера устройства - позиционно:
               pmd = portinp&0377; break;
   case 00000: kom = portinp&077; // прием типа команды
    switch(kom)
     {
      case  01: // сброс на 0-й цилиндр
      case  02: // подвод
      case  03: // чтение
      case  04: // запись
      case  06: // сравнение
                break;
      case  07: // чтение заголовка:
                for(i=0;i<10;i++) drtime(7);
                rdhd(); setPRER; goto G;
      case 010: // гашение РС
                rsoa = rsya = 0; break;
      case 050: // освобождение направления
                dir=0; setPRER; goto G;
       default: break;
     }
          break;
   case 04000: // прием непрерывного...
   case 06000: // ...адреса дорожки.
       adr = portinp&03777;
    if(adr>2029) { rsoa=02000; rsya=04000; // ОшБ и АВ
                   break; }
               for(i=0; i<5; i++) drtime(7); // для сл/слов
    switch(kom)
     {
      case 00: // тоже подвод !
      case 02: // подвод у нас всегда окончен
               mod_n(); // определяем N модуля
              _itoa(adr>>1,zone,10);
               if(!lin) { strcpy(zones[mod],zone); mcol[mod] = 14; }
               else { strcpy(zones1[mod],zone); mcol1[mod] = 14; }
                dr_un(mod+1);
               setPRER; break;
      case 03: // адрес для чтения
               rdmd(); setPRER; break; // вызов функции чтения
      case 04: // адрес для записи
               wrmd(); setPRER; break; // вызов функции записи
      case 06: // адрес для команды сравнения
               cmpr(); setPRER; break; // вызов функции сравнения
      default: setB(0); rsoa=04000; rsya=04000; // ОшПК и АВ
sprintf(buf,"Принят адрес %.4o для команды %.3o...",adr,kom);
    stp(20,58); _outtext(buf); break; //  докладываю: "ложняк" !
     }
```

```
     default: break;
    } // "switch(portinp&06400)"
       } // "if(dir)"
            } // "if(getPKA)"
          } // end of "for(;;)"
 E: _clearscreen(_GCLEARSCREEN); _fcloseall();
    _setvideomode(_DEFAULTMODE);
} // "main" end
```

```
/*г=========================================================¬
  * "emd.h" - Header for "emd.c" - 31.08'99 ... 21.08'2000 *
  L=========================================================-*/

#define TWO 2 // - for two lines


#define stp(x,y) _settextposition(y,x)
#define CDBW {_setcolor(0);_rectangle(_GFILLINTERIOR,60,250,580,480);}
#define CDBW1 {_setcolor(0);_rectangle(_GFILLINTERIOR,0,250,640,310);}
#define WCLN {_setcolor(0);_rectangle(_GFILLINTERIOR,0,0,640,480);}
#define UC unsigned char

 typedef unsigned int UI;

#ifdef TWO
 UC l1[10] = "d:\\emd\\"; // 1-st line
#else
 UC l1[10] = "c:\\emd1\\"; // 1-st line
#endif
 UC cpfn[30]; // name of "exe.file"
 UC ent[21] = ". Нажмите <Enter>...";
 UC disk[8][5] = {"0","0","0","0","0","0","0","0",};  // 0-th line
 UC zones[8][5] = {"4","4","4","4","4","4","4","4",}; // нач.зона
 UI mcol[8] = {15,15,15,15,15,15,15,15,}; // цвет зоны по устр-вам
 UC disk1[8][5] = {"0","0","0","0","0","0","0","0",}; // 1-sl line
 UC zones1[8][5] = {"4","4","4","4","4","4","4","4",}; // нач.зона
 UI mcol1[8] = {15,15,15,15,15,15,15,15,}; // цвет зоны по устр-вам
 UI pimp,scan; // - для опроса клавиатуры в "main()"
 UI pack; // - номер пакета при разметке в авт. режиме
 UI dir;  // - был запрос; гасится при освобождении - '050'
 UI lin;  // - N линейки
 UI mod;  // - N модуля
 UI pmd;  // - N модуля позиционный
 UI kom;  // - тип команды
 UI adr;  // - непрерывный адрес дорожки
 UI portinp,portout; // see in "getUS()"
 UI rsy;  // 1-12 p.p. PC: 1-8 bits - КП(0-7), 9-й - ГД.
 UI rsy1;  // 1-12 p.p. PC: 1-8 bits - КП(0-7), 9-й - ГД.
 UI rso = 01500; // 12-24 p.p. PC: 22-НВМ, 21-ВКЛ, 19-1рРЦГ(?)
 UC opros; // Check all packs, if == '1'
 UC GM; // <ГМ> = '1' от  'У0'(^^^)
 UI GD = 0400; // 9-th bit in "rsy" - ГД

#define ZN 1015 // Число физ.зон на пакете
#define TR 2580 // Число слогов на дорожке
```

```
 UI DB[TR*2]; // Data_Buffer_for_Read of a zone
 UI DW[TR*2+2]; // Data_Buffer_for_Write of a zone
 UI pus; // для опроса входного порта
 UI g; // Test YBY flag
 UI r49 = 0400; // 49-й контр.разряд
 UI key = 0170707; // ключ разметки
 UI slim,d,s,t,cnts; // for all cases
 UC pr[5],pw[5],zone[5],nmod[3],fz[18];
 UC slesh[2]="\\",nu[2]="0";
 UC buf[80],buft[12];

/* for "hdr_emd()" in "emd.fns" */
#define XU 3
#define YU 16
#define SU 9
#define SL 20
#define DU 30

 FILE *fptr;

#include <dos.h> // for "man()"

/* For connect with YBY - 05.09'99 ... 09.06'2000 */

#define P_R1 0x360 // <KYCj> - R;
#define P_W6 0x360 // уст-ка реж.<ИМ> - W;
#define P_W1 0x36A // <Бj> - W.
#define P_R2 0x362 // <ПКА>,<ХБСл>,<ОП>,<~РБУС>,<ГМ>: - R;
#define P_W2 0x362 // гашение <ПКА>  - W.
#define P_W3 0x364 // гашение <ХБСл> - W.
#define P_W4 0x366 // <СИМД> - W (~0.2мксек).
#define P_W5 0x368 // <Прер> - W (~0.2мксек).

#define getPKA  pus&001
#define getHBSL _inpw(P_R2)&002
#define getOP   _inpw(P_R2)&004
#define getRBUS _inpw(P_R2)&010
#define getGM   pus&020
#define getKUS (~_inpw(P_R1))&07777
#define setB(j) _outpw(P_W1,(j)&07777)
#define setSIMD _outpw(P_W4,1)
#define setPRER _outpw(P_W5,1)
#define clnPKA  _outpw(P_W2,0)
#define clnHBSL _outpw(P_W3,0)
#define setIM   _outpw(P_W6,1)

#include "autonom.fns"
#include "emd.fns"
#include "emd.cmd"
```

```
/*г=========================================================¬
   "autonom.fns" - Вспомогатальные служебные функции
    "int ug(void)" - в верхнем регистре _getche()
    "void unit(void)" - Вкл/выкл пакет на устройстве
    "void dr_un(int m)"  - Засветка N пакетов и тек.зон
    "void greb(int u)" - 'гребёнка' -> в слоги УВУ/БЭСМ
   Realease - 30.08'99 ... 19.07'2000
  L=========================================================-*/

int ug(void){ d=_getche(); if(isalpha(d))
 { s = (isupper(d)) ? d : _toupper(d); return s; } else return d; }


UI orsy,orsy1;
void unit()
{
#define NU (01<<mod)
 B: stp(20,38);
 if(!lin) { if( (!(rsy&NU) && !(orsy&NU)) )
    { _outtext("Пакета на устройстве нет");
      _outtext(ent); _getch(); return; } // pack is not
   if(rsy&NU) _outtext("Выключить "); else _outtext("Включить ");
  goto U; }
 else   { if( (!(rsy1&NU) && !(orsy1&NU)) )
    { _outtext("Пакета на устройстве нет");
      _outtext(ent); _getch(); return; } // pack is not
   if(rsy1&NU) _outtext("Выключить "); else _outtext("Включить ");
        }
 U: sprintf(buf,"устройство N°%u%u ? (Y/N) : ",lin,mod);
  _outtext(buf); d=ug(); if(d=='N') return; if(d!='Y') goto B;
 if(!lin) { if(rsy&NU) {rsy&=~NU; orsy|=NU;}
           else {rsy|=NU; orsy&=~NU;}     }
 else   { if(rsy1&NU) {rsy1&=~NU; orsy1|=NU;}
          else {rsy1|=NU; orsy1&=~NU;}      }
   opros=0;
} // "unit()"


void dr_un(int m)
{
 register i,j;
 if(!m) { WCLN;
 for(i=0; i<8; i++) // для 0-й линейки
  {
   if((rsy>>i)&01) { _setcolor(3); _rectangle(_GFILLINTERIOR,
    (XU+i*SU)*8+2,(YU)*8+2,(XU+(i+1)*SU-1)*8-2,(YU+SU)*8-2); }
        _setcolor(15); _moveto((XU+i*SU+2)*8+4,(YU-2)*8);
    j = atoi(disk[i]);  if(j<2048) goto M;
   _outgtext(disk[i]); _moveto((XU+i*SU+2)*8+4,(YU+SU+2)*8);
   j = atoi(zones[i]); sprintf(zone,"%.4o",(j-4)&07777);
   _setcolor(mcol[i]); _outgtext(zone);
  M:    _setcolor(14); _moveto((XU+SU*2)*8, (YU-5)*8);
   _outgtext("П А К Е Т Ы   Н А   Д И С К О В О Д А Х :");
   _moveto((XU+i*SU+1)*8,(YU+SU/2-2)*8); _outgtext("Устр-во");
   _moveto((XU+i*SU+2)*8,(YU+SU/2+1)*8);
    sprintf(buf," 0.%i",i); _outgtext(buf); _rectangle(_GBORDER,
           (XU+i*SU)*8, (YU)*8, (XU+(i+1)*SU-1)*8, (YU+SU)*8);
  }   _moveto((XU+i*SU-1)*8,(YU+SU+2)*8); _outgtext("<зоны>");

 for(i=0; i<8; i++) // для 1-й линейки
  {
   if((rsy1>>i)&01) { _setcolor(3); _rectangle(_GFILLINTERIOR,
    (XU+i*SU)*8+2,(YU+DU)*8+2,(XU+(i+1)*SU-1)*8-2,(YU+DU+SU)*8-2); }
```

```
        _setcolor(15); _moveto((XU+i*SU+2)*8+4,(YU+DU-2)*8);
    j = atoi(disk1[i]);  if(j<2048) goto M1;
   _outgtext(disk1[i]); _moveto((XU+i*SU+2)*8+4,(YU+DU+SU+2)*8);
   j = atoi(zones1[i]); sprintf(zone,"%.4o",(j-4)&07777);
   _setcolor(mcol1[i]); _outgtext(zone);
  M1:   _setcolor(14); _moveto((XU+SU*2)*8, (YU+DU-5)*8);
   _outgtext("П А К Е Т Ы   Н А   Д И С К О В О Д А Х :");
   _moveto((XU+i*SU+1)*8,(YU+DU+SU/2-2)*8); _outgtext("Устр-во");
   _moveto((XU+i*SU+2)*8,(YU+DU+SU/2+1)*8);
    sprintf(buf," 1.%i",i); _outgtext(buf); _rectangle(_GBORDER,
           (XU+i*SU)*8, (YU+DU)*8, (XU+(i+1)*SU-1)*8, (YU+DU+SU)*8);
  }   _moveto((XU+i*SU-1)*8,(YU+DU+SU+2)*8); _outgtext("<зоны>");
       } // "if(!m)"
  else   {
   if(!lin) { // для 0-й линейки
    if((m>8)||(m<0)) return; i=m-1; j=atoi(zones[i]);
     _setcolor(0);  _rectangle(_GFILLINTERIOR,
   (XU+i*SU+2)*8+4,(YU+SU+1)*8,(XU+i*SU+2+5)*8,(YU+SU+4)*8);
   _setcolor(mcol[i]); _moveto((XU+i*SU+2)*8+4,(YU+SU+2)*8);
    sprintf(zone,"%.4o",(j-4)&07777); _outgtext(zone);
   _setcolor(15); _moveto((XU+i*SU+2)*8+4,(YU-2)*8); _outgtext(disk[i]);
            } else { // для 1-й линейки
    if((m>8)||(m<0)) return; i=m-1; j=atoi(zones1[i]);
     _setcolor(0);  _rectangle(_GFILLINTERIOR,
   (XU+i*SU+2)*8+4,(YU+DU+SU+1)*8,(XU+i*SU+2+5)*8,(YU+DU+SU+4)*8);
   _setcolor(mcol1[i]); _moveto((XU+i*SU+2)*8+4,(YU+DU+SU+2)*8);
    sprintf(zone,"%.4o",(j-4)&07777); _outgtext(zone);
   _setcolor(15); _moveto((XU+i*SU+2)*8+4,(YU+DU-2)*8);
_outgtext(disk1[i]);
                  } // "else"
         } // "else"
} // "dr_un()"



/* Функция "гребёнка" для подыгрывания УВУ.
 * 'u' = 0 - // БЭСМ -> УВУ
 * 'u' = 1 - // УВУ -> БЭСМ
 * Release - 26.08'99 ... 13.10'99 */
void greb(int u)
{register i,j;
 UI c[5],d[5];
 for(i=0; i<TR*2/5; i++) // по словам БЭСМ
 { // Формирование 5-ти слогов для каждого слова:
  for(j=0;j<5;j++) {c[j]=0; d[j]=DB[i*5+j];}
  for(j=0;j<5;j++) // по слогам
   {
 if(!u)  // БЭСМ -> УВУ
  {c[j]|=((d[0]>>(9-j))&01)<<9; c[j]|=((d[0]>>(4-j))&01)<<8;
   c[j]|=((d[1]>>(9-j))&01)<<7; c[j]|=((d[1]>>(4-j))&01)<<6;
   c[j]|=((d[2]>>(9-j))&01)<<5; c[j]|=((d[2]>>(4-j))&01)<<4;
   c[j]|=((d[3]>>(9-j))&01)<<3; c[j]|=((d[3]>>(4-j))&01)<<2;
   c[j]|=((d[4]>>(9-j))&01)<<1; c[j]|=((d[4]>>(4-j))&01)<<0;
  } else // УВУ -> БЭСМ
  {c[0]|=((d[j]>>9)&01)<<(9-j); c[0]|=((d[j]>>8)&01)<<(4-j);
   c[1]|=((d[j]>>7)&01)<<(9-j); c[1]|=((d[j]>>6)&01)<<(4-j);
   c[2]|=((d[j]>>5)&01)<<(9-j); c[2]|=((d[j]>>4)&01)<<(4-j);
   c[3]|=((d[j]>>3)&01)<<(9-j); c[3]|=((d[j]>>2)&01)<<(4-j);
   c[4]|=((d[j]>>1)&01)<<(9-j); c[4]|=((d[j]>>0)&01)<<(4-j);
  }} // "for(j)"
 for(j=0;j<5;j++) DB[i*5+j]=c[j]; // new structure fixup
 } // "for(i)"
} // "greb()"
```

```
/*г=================================================================¬
   "emd.fns" - Служебные функции для "main()":
   "int how_file(char *f_n,char *mode)" - File open
    "int video(int nf)"  - Fonts setup
    "int pal_file(void)" - Palette setup
    "int krdf_e(int nf)" - Graph_mode set from "main()"
    "void gotoxy (char x, char y)" - set text position
    "void hdr_emd(void)" - Заставка продукта "ЭМУЛЯТОР"
    "void drtime(int w)" - Draw_of_Time_value & "БЭСМ"
    "void mdcheck(void)" - опознание пакетов на устр-вах
    Release - 15.06'99 ... 19.07'2000
  L=================================================================-*/


/* Открытие файла "*f_n" в режиме "*mode" */
int how_file(char *f_n, char *mode)
{if((fptr=fopen(f_n,mode))==NULL) return 1; return 0;};

int video(int nf)
{
// unsigned char _far *option = "t'tms rmn'h5w5b";
 UC *option[6]={"n1","n2","n3","n4","n5","n6"};
 UC *fondir = "*.fon";
 _setvideomoderows(_VRES16COLOR,25);
 if( _registerfonts(fondir) <= 0 )
  { printf("\n%s: No such file or directory \n",fondir); goto R; }
 if ( _setfont(option[nf]) < 0 )
  { printf("\n Нет шрифта \"%.18s\" в файле \"%s\" \n",option[nf],fondir);
 R:  _getch(); return 1; }
 _setvideomoderows(_VRES16COLOR,60); return 0;
}; // "video()"


int pal_file()
{
 # define MAXCOL 16
 long pal[MAXCOL];
 UC spalsim[13]="dtk_sim5.pal"; // Файл с палитрой
 UC n; // Чтение палитры из файла:
  if(how_file(spalsim,"r")) { perror(spalsim); return 1; }
  for(n=0; n<MAXCOL; n++) fscanf(fptr,"%lx ",&pal[n]);
  _remapallpalette(pal); fclose(fptr); return 0;
}; // "pal_file()"


int krdf_e(int nf)
{
 int h=0;
  h  =  video(nf); // Чтение файла с графич. фонтами
  h += pal_file(); // Чтение файла с палитрой
  return h;
}; // "krdf()"
```

```
/* Set text cursor position - Kalinin */
void gotoxy (char x, char y)
{
 _asm {
 mov dh,y
 mov dl,x
 mov bh,0
 mov ah,2
 int 10H
      }
}; // "gotoxy()"



void hdr_emd(void)
{static UI clr=10;
 _setcolor(clr); clr++; if(clr>15) clr=0;
 _moveto((XU+SU*2-4)*8+4, (YU-11)*8);
 _outgtext("Э М У Л Я Т О Р   М А Г Н И Т Н Ы Х   Д И С К О В");
 _setcolor(clr+1); _rectangle(_GBORDER,
  (XU+SU*2-5)*8,(YU-13)*8+6,(XU+SU*2+40)*8+4,(YU-8)*8-4 );
 _moveto(8,8); _outgtext(cpfn);
} // "hdr_emd()"



// Draw of time_field
void drtime(int w) {
#define PT 35 // X_time_position_at_screen
 static char sbuft,buft[12],w1=8;
 _strtime(buft); if(w) w1=w;  if(buft[7] != sbuft)
 {_setcolor(3); _rectangle(_GFILLINTERIOR,PT*8-1,8-1,(PT+w)*8,8*2+3);
  _moveto(PT*8,8);  _setcolor(14);  sbuft = buft[7]; _outgtext(buft);
   hdr_emd(); // Look at me please how I am wounderfull !
 } if(GM) _setcolor(12); else _setcolor(10);
   _moveto(597,8); _outgtext("БЭСМ");
} // "drtime()"



/* Опознание пакетов по принципу БЭСМ - по 4-й зоне */
void mdcheck(void)
{register i,j;
 UC fz1[4];
 UC z4[7]="\\0004";
           rsy = 0;
 for(j=0;j<8;j++) {_itoa(j,fz,10); strcat(fz,z4);
  if(how_file(fz,"rb")) continue;
  fread(DB,2,TR*2,fptr); fclose(fptr); greb(1); // to BESM-codes
  i=(DB[1*5+3]&01777)>>2; i|=(DB[1*5+2]&017)<<8; // N pack
 _itoa(i,disk[j],10); if(!g) rsy|=01<<j; } rsy &= ~orsy;
           rsy1 = 0;
 for(j=0;j<8;j++) {_itoa(j,fz1,10); strcpy(fz,l1);
  strcat(fz,fz1);  strcat(fz,z4);
  if(how_file(fz,"rb")) continue;
  fread(DB,2,TR*2,fptr); fclose(fptr); greb(1); // to BESM-codes
  i=(DB[1*5+3]&01777)>>2; i|=(DB[1*5+2]&017)<<8; // N pack
 _itoa(i,disk1[j],10); if(!g) rsy1|=01<<j; } rsy1 &= ~orsy1;
  opros=1; if(!g) dr_un(0);
} // "mdcheck()"
```

```
/*г=======================================================¬
  *  "emd.cmd" - Real mode of 03,04,06,07 - commands
      "void nozone(void)" - 'Не найден файл для зоны'
      "void mod_n(void)" - позиц. N модуля -> в цифровой
      "int zon_p(char*)" - zone_path for 'rdmd/wrmd'
      "void rdmd(void)" - read with MD
      "void wrmd(void)" - write on MD
      "void cmpr(void)" - zones compare
      "void rdhd(void)" - head/dor_read
  *  Release - 24.08'99 ... 19.07'2000
  L=======================================================-*/


void nozone(int z){ stp(20,55); _outtext("Не найден файл зоны ");
 sprintf(buft,"%.4o",z); _outtext(buft); _outtext(ent); _getch(); }


void mod_n(void)
{ // определяем N модуля
 UI i;
 UC md[8]={0,1,2,3, 4, 5, 6,  7};
 UC pm[8]={1,2,4,8,16,32,64,128};
     mod=0; for(i=0; i<8; i++)
   { if(pmd==pm[i]) { mod=md[i]; break; } else continue; }
}


int zon_p(char *m)
{
 UI i;
 mod_n();
  _itoa(mod,nmod,10);
   if(!lin) strcpy(fz,nmod); else { strcpy(fz,l1); strcat(fz,nmod); }
   strcat(fz,slesh); i=adr>>01;
  if(!(i/1000)) strcat(fz,nu); if(!(i/100)) strcat(fz,nu);
  if(!(i/10)) strcat(fz,nu); _itoa(i,zone,10); strcat(fz,zone);
  if(how_file(fz,m)) { nozone(i); return 1; }; return 0;
}

void rdmd(void)/* Драйвер чтения зоны - операция '03'*/
{
 register i;
  if(zon_p("rb")) goto R;
  fread(DB,2,TR*2,fptr); fclose(fptr);
 if(!lin) { strcpy(zones[mod],zone); mcol[mod]=10; }
 else { strcpy(zones1[mod],zone); mcol1[mod]=10; } dr_un(mod+1);
  t=0; if(adr&01) t=TR; // 2-nd half of zone
 B: for(i=0; i<TR; i++)
   { setB(DB[t+i]&01777); setSIMD;
 W:  if(getHBSL) clnHBSL; // гаш. <ХБСл>
     else goto W;  // wait of HBSL
    if(!(getRBUS)) goto R; // сл/слова
   } // "for(i)"
  if( (getRBUS) ) // (RBUS&&(!(OP))
    if(!t) { t=TR; goto B; } // to 2-nd half
 R: return;
}; // "rdmd()"




void wrmd(void) /* Драйвер записи зоны - операция '04'*/
```

```
{
 register i;
  if(zon_p("wb")) goto R;
   t=0; if(adr&01) t=TR; // 2-nd half of zone. Приём 0-го слога:
 A: for(i=0; i<TR; i++) { setSIMD;
 W: if(getHBSL==0) goto W;
    DW[t+i]=getKUS&01777; // прием i-го слога из УВУ
     clnHBSL;
    if(!(getRBUS)) goto Q; // sl/words (?)
                          } // "for(i)"
   if(getRBUS) { if(!t) { t=TR; goto A; } } // 2-nd half
   else {
 Q:  fwrite(DW,2,TR*2,fptr); fclose(fptr);
   if(!lin) { strcpy(zones[mod],zone); mcol[mod]=12; }
   else { strcpy(zones1[mod],zone); mcol1[mod]=12; } dr_un(mod+1);
        }
  R: return;
}; // "wrmd()"


/* Драйвер команды сравнения - '06' */
void cmpr(void)
{
 register i;
   if(zon_p("rb")) goto R;
   fread(DB,2,TR*2,fptr); fclose(fptr);
   t=0; if(adr&01) t=TR; // 2-nd half of zone only
 A: for(i=0; i<TR; i++) { setSIMD;
 W: if(getHBSL==0) goto W;
    DW[t+i]=getKUS&01777; // прием слога из УВУ
    clnHBSL; if(!(getRBUS)) goto Q; // sl/words
                          } // "for(i)"
  if(getRBUS) { if(!t) { t=TR; goto A; } } // 2-nd half
  else {
 Q:     i=adr>>1; _itoa(i,zone,10);
    if(!lin) { strcpy(zones[mod],zone); mcol[mod]=13; }
    else { strcpy(zones1[mod],zone); mcol1[mod]=13; } dr_un(mod+1);
       }
 R: return;
}; // "cmpr()"


/*  Формирование заголовков дорожек и выдача их в УВУ для операции '07' */
void rdhd(void)
{
 register i,ks,ch=0;
 UI nam=0404; // Addr_Marker for normal tracks
 UI ram=0406; // Addr_Marker for reserve tracks
 UI ksn=07777; // к.с.а. для 0-й дорожки
 UI ksr=07730; // к.с.а. для 200-й дорожки и далее
#define NA 2000 // Max_Address of normal tracks
 UI HDR[SL];    // Slogs for YBY

  for(i=0; i<SL; i++) HDR[i] = 0;
 // 1-st address:
  HDR[0] = (adr<NA) ? nam : ram; // slog 0
  ch = (adr%10) | ((adr/10)<<4); // N cyl/head
  HDR[1] = (ch>>3) & 0777;
  HDR[2] = (ch<<6) & 0777;
  ks = (adr<NA) ? ksn : ksr;
  ks ^= (ch>>8); ks ^= ((ch&0377)<<4);
  HDR[2] |= (ks>>10)&03;
  HDR[3]  = (ks>>01)&0777;
```

```
  HDR[4]  = (ks&01)<<8;
 // 2-nd address: miss 42-8=34 null bits (= 9*3 + 7 bits)
 // and then doubling with shift 1-st address
  for(i=0; i<5; i++) {
   HDR[i+8] |= (HDR[i]>>7)&03;
   HDR[i+9]  = (HDR[i]<<2)&0777;
                     }
 // 20 slogs are ready for transmission to YBY
   for(i=0; i<SL; i++)
    { setB(HDR[i]&0777); setSIMD;
 W:  if(getHBSL) clnHBSL; // гаш. <ХБСл>
      else goto W;  // wait of HBSL
     if(!(getRBUS)) goto R;
    } // "for(i)"
 R: drtime(7); setPRER;
}; // "rdhd()"
```