ИНСТРУКЦИЯ К ПРОГЕ imitator.exe версия: 128k v.1.2 **16.02.2023**.

СОДЕРЖАНИЕ

0. Об имитаторе Д3-28, системные требования, запуск

Благодарности:

Благодарю – моего коллегу Сергея Александровича Р., предоставившего свои магнитофонные кассеты с программами для Д3-28, а также фотографии плат (а затем и сами платы) согласующих устройств;

– моего коллегу Анатолия Владимировича Б., подготавливавшего рукописные материалы по Фортрану-85 и Выстре и кассеты с ПО;

– Виталия Васильевича Колесника – за многочисленные труды: создание сайта Все о Д3-28 (с on-line дизассемблером!), поиск разнообразной информации, сканирование раритетных текстов с описанием программ, технической документации, схем Д3-28, и за оцифровку уникальной коллекции кассет с ПО (в том числе ОС ВТ-МХТИ-128, – это стало главным стимулом к разработке Д3-28-имитатора с расширенным от 32к до 128к ОЗУ);

– участницу форума «Полигон Призраков» Наталию с ником «**xlat**» благодарю особенно: уважаемая Наталия – единственный человек в стране, сумевший в наши дни восстановить реальную Д3-28 до работоспособного состояния. Спасибо, Наталия, за предложение помощи и проведение опытов на реальной машине Д3-28, в которых выявились особенности команд этой ЭВМ! А также спасибо Вам и «**maddev**» за онлайновый эмулятор ЭВМ Д3-28 ;

– участников форума «Полигон Призраков» за информацию в теме Электроника Д3-28 (Вопросы по этому "Калькулятору с кассетой") ; особенно – участника с ником «**sanders**» за предоставление кассет для оцифровки, «**shattered**», пояснившего специфику работы фрязинского дисплея 15ИЭ-00-013 при передаче блоков, и «**xoiss**» – за столь необходимую моральную поддержку, интерес к Д3-28, множество идей, и за восстановление схемы МР УСО.

– «**old_hippie**» и других участников форума РТ 20, поделившихся интересными и полезными воспоминаниями о Д3-28;

– авторов сайтов www.leningrad.su/museum/, www.pc-history.com и retropc.org – с замечательной библиотекой технической документации, обеспечивших свободный доступ к фото Д3-28 и докам по Д3-28. (В данном «Руководстве пользователя имитатора Д3-28» на стр. 151-153 в Приложении 4 показаны в уменьшенном виде фото с копирайтами авторов сайтов; эти фото после ретуши и ещё некоторых преобразований были применены в имитаторе Д3-28).

Системные требования: прога запускается под управлением 32-разрядных ОС Win ME, XP, Vista, Win 7 – эти варианты мной опробованы. Автор сайта d3-28.ru сообщил также об успешной работе проги в Linux с Wine.

Важно: драйвер видеосистемы компьютера должен поддерживать OpenGL. (Не являюсь спецом в таких вопросах, но, например, попытки запуска проги в виртуальной Win XP в VM Virtual Box на ноутбуке с Win 7 показали мне, что прога

там дико тормозит – даже невозможно дождаться начального диалога с Бейсиком… Хотя на том же ноутбуке прямо под Win 7 прога работает нормально.)

Запуск: после распаковки (в режиме «с сохранением подкаталогов») zip-архива в любое место на любом диске запускайте прогу из файла imitator.exe. Установка (инсталляция) проги не требуется. Проге не нужен интернет. Она ничего не пишет в реестр, хотя сама Windows, видимо, отмечает в реестре запуск проги, так как она запоминает, из какой папки открываются файлы.

В результате распаковки zip-архива в папке с файлом imitator.exe должна образоваться папка [u_soft], папка [txt], и папка [bmp] – с картинками шрифта и Д3-28. Папку [bmp] и картинки нельзя переименовывать / удалять – без них прога не запустится. А с [u_soft] и [txt] можно поступать как угодно. В [txt] хранятся файлы, имитирующие магнитофонные записи с ПО для Д3-28; в [u_soft] – ПО для создания wav-файлов с ПО; в обе папки можно будет добавлять и своё творчество.

Прога имитирует работу старинного вычислительного комплекса *"Электроника Д3-28 с 32-килобайтным адресным пространством и расширенным до 128 килобайт ОЗУ, с терминалом 15ИЭ-00-013, подключенным к машине через согласующее устройство ПЕЛ2.240.001, с текстовым печатающим устройством (ТПУ) типа термопечатающего устройства 15ВВП80-002"*. Такой комплекс по своим возможностям близок к «Радио-86РК», но появился он примерно на пять лет раньше, где-то на рубеже 1980-х годов. Информация выводится в комплексе на алфавитно-цифровой дисплей терминала (или на алфавитно-цифровое ТПУ и на цифровое табло машины). «Графика», «псевдографика» и встроенная операционная система в этом комплексе отсутствуют.

Для ясности договоримся о терминах:

ПК – персональный компьютер, на котором запускается прога имитатора Д3-28.
ПК-клава – клавиатура персонального компьютера. (Ещё и мышь пригодится).
Монитор – экран монитора персонального компьютера. Это всё, что надо проге.

Машина – изображение Д3-28 на мониторе или воображаемая Д3-28 в имитаторе.
Пульт (и табло) – изображение пульта управления (и табло) Д3-28 в имитаторе.
Дисплей – окно проги в режиме имитации экрана терминала 15ИЭ-00-013.
Клава – воображаемая или изображённая клавиатура терминала 15ИЭ-00-013.
КД – воображаемый контроллер дисплея, согласующий дисплей с машиной.
ТПУ – воображаемое термопечатающее устройство, подключенное к Д3-28.
НМЛ – изображённый или воображаемый кассетный магнитофон машины.
Программ(к)а – последовательность команд для машины Д3-28.
Прога – наш имитатор.

Нажать клавишу – означает «нажать и отпустить». О необходимости удерживать клавишу нажатой будет сказано явно, а иначе – надо обязательно отпускать.

1. Быстрое ознакомление с имитатором

1. Запустите на исполнение imitator.exe. На мониторе появится окно проги с изображением машины Д3-28. Изображение машины и клавы строится, к сожалению, не целиком, а из шести текстур – рисунков 256x256 px; поэтому на разных ПК возможны разные «краевые» дефекты изображения в виде тонких вертикальных и горизонтальных полос, разделяющих текстуры. Если они очень заметны, то для их устранения пробуйте потянуть мышью вертикальные или горизонтальные края окна, меняя его размер. Изменять размер окна допускается.

Отсутствие крышки «НМЛ» и отсутствие имени txt-файла в верхней строке окна (то есть в строке с именем проги), служат признаками того, что «в НМЛ не вставлена кассета с магнитофонной записью» какой-либо программы.

2. В меню **File** выберите **Open txt-file as Tape**. В открывшемся стандартном окне обзора файлов найдите нашу папку [txt] и выберите из неё для открытия файл Basic_D3-28_v3A__KP-157107__N-11343.txt. Тем самым вы «поставили в машину кассету» с Бейсиком-157107. Признаки присутствия кассеты: НМЛ теперь изображается с закрытой крышкой, а вверху окна проги присутствует имя открытого txt-файла, имитирующего магнитную ленту.

Пояснение: слово tape означает «магнитофонная лента». Вообще-то я не знаю английского как следует, но решил для всех текстов в проге использовать только латинский шрифт, поскольку для Windows это родной системный шрифт, и с ним не должно быть проблем. Кириллица же при неумелом программировании может отображаться на экранах компьютеров в форме непонятных «крюкозяблов». Поэтому тексты в меню проги и в окнах сообщений напечатаны мной на некоем «квази-английском» :-), ведь я не то что неумелый программист, а вообще не программист, увы.

3. Нажмите на ПК-клаве клавишу <$C_{лат}$> – это имитация кнопки С на пульте машины, т. е. команда «Сброс». Затем нажмите <L> – это имитация кнопки СЛ на пульте, команда «читать с ленты». При этом мигнёт окошко в крышке НМЛ – здесь это признак того, что файл «загрузился с ленты в ОЗУ машины Д3-28».

4. Нажмите <$K_{лат}$> – это имитация кнопки КП (Контроль Программы) на пульте машины, т. е. команда «вычислить контрольную сумму» файла, находящегося в ОЗУ машины. В нижнем табло машины будут видны цифры контрольной суммы 157107, точка и нули.

5. Нажмите опять <C>, чтобы программный счетчик сбросился к нулевому шагу. И затем <S> – это имитация нажатия кнопки S на пульте, т. е. команда «Start», – команда запуска программы из ОЗУ машины с текущего шага. Машина перейдёт из режима «останов с индикацией» в режим «работа по программе»; в нашем примере это означает, что начнёт работать Бейсик-157107. Слева рядом с нижним табло загорится индикатор ожидания машиной ответа от терминала.

6. В меню **View** выберите **Display**. Поначалу прога имитирует терминал, отключенный от машины, поэтому на дисплее нет текста (кроме служебной строки с цифрами); программа в машине ожидает сигналов от терминала. Нажмите на ПК-клаве клавишу <F5> – это имитация включения ДУП и ЛИН на клаве терминала. Бейсик выведет на дисплей фразы своего «начального диалога».

7. Ответьте на вопросы в диалоге, причём в ответ на СНИМИТЕ КАССЕТУ! и НОМЕРА ВНЕШНИХ ПОДПРОГРАММ? нажимайте <Enter> на ПК-клаве. Вообще, для ввода цифр можно нажимать на ПК-клаве обычные клавиши с цифрами или клавиши дополнительной цифровой клавиатуры (Numpad, если она есть в вашем ПК); точка на Numpad при вводе дробных частей десятичных чисел тоже будет работать. Ввод завершайте ПК-клавишей <Enter> – она имитирует клавишу ПС (перевод строки) на клаве терминала, или ПК-клавишей <Backspace> – она имитирует клавишу ВК (возврат каретки). Вместо ввода числа «ноль» во многих больших программах для Д3-28, в том числе в Бейсике, допускается просто нажимать ПС.

8. После того, как Бейсик выведет на дисплей слово ГОТОВ и двоеточие, нажмите (и отпустите) на ПК-клаве клавишу <Ctrl> – тем самым клава терминала переводится из режима РУС в режим ЛАТ; это необходимо для правильного ввода команд Бейсика. Когда потребуется режим РУС, надо будет снова нажать <Ctrl>.

Чтобы посмотреть текущее состояние клавы, нажмите <F1> (а для возврата – ещё раз <F1>). Это работает, когда в меню **View** выбран **Display**. На мониторе изобразится клава терминала (а также знак «Выкл / Вкл ТПУ»), и можно будет посмотреть, как меняется состояние при нажатиях следующих ПК-клавиш:

<F5> – включает / выключает ДУП-ЛИН.
<Ctrl> – переключает РУС / ЛАТ,
<Shift> – переключает ВР / НР («верхний регистр» / «нижний регистр»),

Менее важные клавиши, работают так только после <F1> в режиме **View**>**Display**:

<4> – вызывает окошко с информацией о «бите b4» контроллера дисплея; см. раздел 5,

<V> – показывает «версию имитатора» – дату компиляции exe-файла,
<M> – показывает режим имитации ОЗУ: 128k или 32k, возможен выбор;
<S> – показывает номера сегментов, подключенных к страницам ОЗУ, см. раздел 6,

<F> – вызывает диалог об индикации оценки FPS («кадров в секунду»),
<D> – вызывает диалог о включении замедления; см. раздел 8,

<L> – вызывает диалог о команде LOAD; см. раздел 3, стр. 30,
<C> – вызывает диалог об «отсветке» при OUTOWC; см. раздел 3, стр. 32,

<E> – вызывает диалог «показывать ли на табло порядок E числа, если E = –0»,
<K> – вызывает диалог об индикации обнаружения «редкостных» команд.

<O> – позволяет обнаружить команды с Ошибкой Программы.
<R> – позволяет включить обнаружение шагов с командой RTS (возврат из подпрограмм), а затем и перейти к пошаговому режиму. Всё это помогает исследовать программы.
<A> – запуск встроенного в имитатор дизассемблера; см. раздел 4.
<P> – печатать или не печатать маленькие буквы в распечатках на ТПУ.
<U> – вкл. "Устройства" для создания wav-файлов; см. руководство в pdf в папке u_soft. (В интернете уже добавилась папка u_soft_2.zip в папке **D3-28** . Ссылки, в том числе на текущие выполняемые мной обновления и доки, см. там в файле AAA_READ_ME.)

О включении ТПУ см. раздел 4. Для управления ТПУ используются ПК-клавиши <Home> и <End>. Прочие клавиши не следует нажимать в режиме просмотра клавы. Соответствие между ПК-клавой и клавой терминала в режиме набора текстов на экране дисплея описано в разделе 2.

9. Выберите **File** > **Close txt-file as Tape** – так вынимается кассета из НМЛ. Затем: **Open txt-file as Tape**, и откройте файл game_1WAR_bas.txt; тем самым в НМЛ ставится кассета с игровой тест-программой «1WAR» на языке Бейсик.

10. Наберите команду LOAD и нажмите ПС, т. е. <Enter> на ПК-клаве. Через короткое время успешное выполнение команды обозначится двоеточием в новой строке дисплея. При наборе команд и любых слов смотрите только на латинские буквы на клавишах ПК-клавиатуры. Ведь ПК-клава в имитаторе не может воспроизвести расположение и ЛАТ- и РУС-букв с клавиш старинного терминала, да ещё и кнопок на пульте машины, поэтому в имитаторе приняты некие правила соответствия клавиатур; о них идёт речь в разделе 2.

11. Чтобы посмотреть, есть ли имя у загруженного Бейсиком файла, наберите PRINTOPEN и нажмите ПС; имя выведется вслед за буквой P. Вообще, в Бейсике любое действие подтверждайте нажатием ПС, т. е. <Enter> на ПК-клаве.

12. Для запуска программы подайте команду RUN. Описание игры 1WAR приводилось в другом pdf. Но в ней и так всё понятно: надо уничтожить летающей точкой-«истребителем» вражескую «эскадрилью» квадратиков. Для управления в этой игре назначены цифровые клавиши (удобнее нажимать их на Numpad):

<1>, <2>, <3> – выбор горизонтальной составляющей скорости точки;
<7>, <8> – налево, направо; <9>, <6> – вверх, вниз; <0> – выход из игры.
При выходе из игры на экран будет выведено количество шагов («time») и штрафные очки («shocks» – количество столкновений точки с краями игрового поля). Подразумевается, что чем с меньшими «time» и «shocks» уничтожены все квадратики, тем лучше игрок справился с задачей :-)

(Однако эта бейсик-программка – демо-новодел; на реальной Д3-28 она жутко бы «тормозила». На реальной Д3-28 хорошо работала аналогичная фортранная программа; см. раздел 5.)

13. Любую бейсик-программу можно остановить клавишей АР1 на клаве терминала; на ПК-клаве в имитаторе АР1 это <F3>. (Но срабатывает это не всегда с первого раза). Другой способ подачи из терминала команды АР1 – нажатие

«клавиши СУ вместе с клавишей $Р_{лат}$». В имитаторе нажатие СУ имитируется нажатием (без удержания) клавиши <CapsLock>. Отображается это светодиодом «CapsLock» самой ПК-клавы (если он есть в вашем ПК), а также надписью «СУ нажата!» над клавишей СУ на изображении клавиатуры терминала при его просмотре командой <F1> в режиме **View** > **Display**. Будьте внимательны: если нажали <CapsLock>, то не забудьте в дальнейшем нажать её ещё раз, чтобы выйти из имитации СУ и вернуть ПК-клавиатуру в исходное состояние.

14. После того, как бейсик-программа остановлена (или до того, как запущена), можно посмотреть на дисплее её листинг, набрав команду LIST. После вывода «одного экрана» Бейсик приостанавливает вывод; для продолжения листинга нажимайте ПС; если нажать <Пробел>, то листинг прервётся. Чтобы вывести лишь несколько строк, надо указать первый и последний номер строк для вывода, например LIST 200,270 (и, как обычно, нажать ПС, т. е. <Enter>).

15. Командой CLEARP 1,7999 очищаем память Бейсика от всех строк текущей бейсик-программы, подготавливаясь к вводу с НМЛ или к набору с клавы новой бейсик-программы. Очистив так память, для примера наберём и запустим командой RUN (с последующим ПС, т. е. <Enter>) ритуальную программку:

```
1 PRINT'HELLO WORLD'
```

Если повторить такой опыт с РУС-буквами, то при исполнении программки или при её листинге Бейсик-157107, увы, не выведет на дисплей русских букв:

Видно, что здесь есть проблема с выводом РУС-букв. О её решении см. в разделе 5 про «бит b4». Есть также более простое, но лишь частичное решение, – пригодное в тех случаях, когда программка должна выводить одни только русские буквы, без латинских: после набора RUN, но до нажатия ПС можно перевести дисплей в режим РУС (нажав на ПК-клаве клавишу <Ctrl>). Для примера проведите опыт: наберите вот такую программку с маленькими латинскими буквами в кавычках оператора PRINT

```
1 PRINT'vopa'
```

В новой строке наберите RUN, нажмите <Ctrl> и только после этого <Enter>.

16. Клавиша <Escape> на ПК-клаве вызывает «ресет» имитатора; это почти то же самое, что закрыть прогу имитатора и открыть её заново (при ресете запоминаются только текущий «бит b4», вывод FPS, печать маленьких букв, а также значение параметра замедления, если он был задан). Для повторного запуска ресет бывает более удобным, чем закрытие имитатора и новый его запуск. Ресет имитатора удобен, если что-то пошло не так. Однако не перепутайте клавишу <F1> с <Escape>: ресет, как и закрытие имитатора, действует немедленно – прога ничего не переспрашивает и не сохраняет результатов работы; и в ней не предусмотрена отмена (undo) ни для каких действий.

2. Управление дисплеем

<F1> – эта ПК-клавиша вызывает (только при **View** > **Display)** изображение клавы терминала с индикаторами текущего состояния; повторное нажатие <F1> – возврат к экрану дисплея.

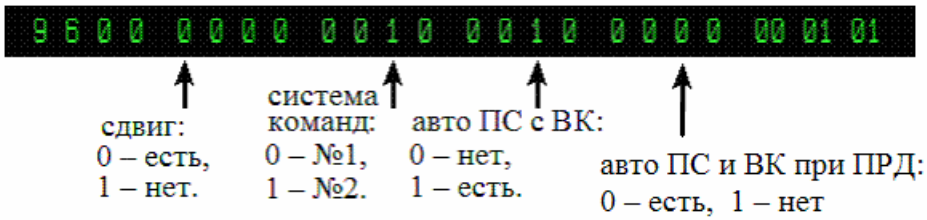
Некоторые функциональные клавиши в имитаторе, как и на реальном терминале, генерируют байт-код для передачи через «Линию» в машину; ниже коды указаны в 16-ричной системе счисления в тетрадно-десятичной форме:

<F2> – имитирует клавишу АР2 на клавиатуре терминала (код 0111),
<F3> – имитирует клавишу АР1 на клавиатуре терминала (код 0100),
<F4> – имитирует клавишу С1 на клавиатуре терминала, (код 0001),
<F12> – имитирует клавишу СБР на клавиатуре терминала (код 0012); в системе команд №1 очищает ОЗУ дисплея, в с. к. №2 действует как ПС.

Другие функциональные клавиши меняют состояние терминала, но кодов не дают:

<F5> – имитирует включение / выключение ЛИН и одновременно ДУП,
<F6> – при включённой ЛИН выключает / включает ДУП,
<F7> – выключает / включает РЕД.

<F8> – включает / выключает настройку служебной строки, т. е. имитирует СДВ. Прога пока ещё не умеет имитировать работу дисплея точно. В частности, в служебной строке доступно изменение пока только четырёх параметров:



В режиме СДВ ПК-клавиша <F9> сбрасывает их в 0. Установка в 1 или 0 – ПК-клавишей <↓>, маркер надо ставить слева от нужного разряда, двигая его вдоль служебной строки ПК-клавишами <→> или <←>.

<F9> – ОЧС экрана (при не СДВ). Но такая очистка у нас не сбрасывает дисплей в начальное состояние после включения питания, а устанавливает показанную выше настройку служебной строки (при «b4 = 0»), и задаёт клаве терминала режим РУС (русские буквы) и ВР (верхний регистр, т. е. большие буквы). Так сделано для того, чтобы меньше нажимать всяких клавиш при запуске Бейсика-157107 вслед за запуском имитатора. То есть считается, что работа с Бейсиком-157107 это основной начальный режим, а работа с другими программами и режимами – занятие «для знатоков», которые сумеют сами выбрать нужную им настройку дисплея и клавы.

При «b4 = 1» очистка экрана клавишей <F9> и ресет клавишей <Escape> задают параметры служебной строки так: 9600 0000 0000 0000 0000 … .

Клавиша <F10> принадлежит самой ОС Windows, а клавиша <F11> пока находится в резерве: она не задействована в имитаторе.

<CapsLock> – при каждом нажатии имитирует прижимание / отпускание СУ.

<Shift> – имитирует переключение ВР / НР, не генерирует кодов для «Линии».

<Ctrl> – переключает РУС / ЛАТ и генерирует коды: РУС – 0014, ЛАТ – 0015.

Другие клавиши, применяемые для имитации клавиатуры терминала, тоже генерируют коды:

<Tab> – генерирует код ТАБ (0108); ничего другого эта кнопка у нас не делает.

<Delete> – имитирует клавишу ЗБ, т. е. «забой» (код 0715).

<Backspace> – имитирует нажатие ВК, т. е. «возврат каретки» (код 0013).

<Enter> – имитирует нажатие ПС, т. е. «перевод строки» (код 0010).

Клавиши со стрелками ↑ ↓ ← → на ПК-клавиатуре при **View** > **Display** (но не при СДВ) имитируют аналогичные клавиши со стрелками клавы терминала – с кодами 0112, 0113, 0110, 0109 соответственно. Эти коды отрабатываются дисплеем в системе команд № 1, а в системе команд № 2 они не отрабатываются.

У терминала 15ИЭ-00-013 есть и другие специальные клавиши формирования текста на экране. Их коды, как и вообще все служебные коды (от 0000 до 0115) можно генерировать нажатием СУ с некоторой алфавитно-цифровой клавишей: в её семибитном байт-коде (с нулевым 8-м битом) биты 7-ой и 6-ой при нажатой СУ сбрасываются в ноль. В имитаторе именно так можно получать коды отсутствующих клавиш терминала; роль СУ выполняет <CapsLock>. Например, нажав после <CapsLock> клавишу < I > (код 0409), имитируем нажатие ГТ (код 0009).

Чтобы знать возможности терминала и осознанно ими пользоваться, следует ознакомиться с его режимами и с его двумя системами команд по документам:

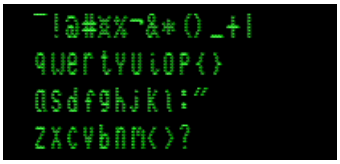
«Дисплей алфавитно-цифровой 15-ИЭ-00-13:
Папка № 1. Техническое описание. ЩЦМ 3.778.012 ТО,
Папка № 2. Инструкция по эксплуатации. ЩЦМ 3.778.012 ИЭ»,
(см. https://d3-28.ru/displej-alfavitno-tsifrovoj-15-ie-00-13/ или
http://emuverse.ru/wiki/15%D0%98%D0%AD-00-013)

При наборе текстов важны клавиши ВР / НР – они влияют на выбор шрифта, но кодов в машину не посылают, и клавиши РУС / ЛАТ – они переключают русские и латинские буквы и при этом посылают свои коды в машину. Как уже пояснялось, их имитация такова:

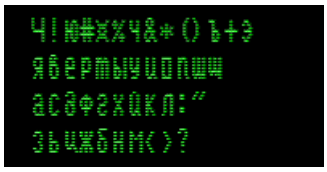
<Ctrl> – переключает РУС / ЛАТ,
<Shift> – переключает ВР / НР, то есть «верхний регистр» / «нижний регистр».

В имитаторе в режиме ВР, т. е. при выключенном НР (индикатор НР клавы терминала при этом погашен) на дисплее набираются цифры и большие буквы.

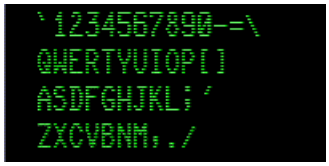
При выключенном ВР, т. е. при включённом НР (индикатор НР при этом горит) набираются, как правило, маленькие буквы и следующие символы в режиме ЛАТ:



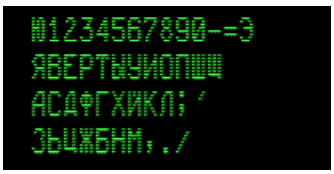
Нажатие тех же ПК-клавиш в режиме НР РУС даёт:



Нажатие тех же ПК-клавиш в режиме ВР ЛАТ даёт:



Нажатие тех же ПК-клавиш в режиме ВР РУС даёт:



Для набора русских букв (разумеется, в режиме РУС) не надо смотреть на расположение русских букв на ПК-клаве; надо смотреть только на латинские буквы на ПК-клаве и нажимать латинские буквы по правилу соответствия:

$А_{рус} – А_{лат}$, $Б_{рус} – В_{лат}$, $В_{рус} – W_{лат}$, $Г_{рус} – G_{лат}$, $Д_{рус} – D_{лат}$, и т. д.

При этом роль «хелпа» может выполнять изображение клавы терминала, которое включается и выключается ПК-клавишей <F1> как раз в режиме **View** > **Display**, – там каждая пара соответствующих друг другу русских и латинских букв изображена на одной и той же клавише терминала.

Исключение в имитаторе составляют буквы Ч и Ю: см. картинки выше. Так вышло потому, что на клаве терминала буква Ч попала на клавишу со шляпкой ^, буква Ю попала на клавишу с собакой @, тогда как на ПК-клаве шляпка ^ и собака @ находятся на клавишах с цифрами.

Цифры и десятичную точку удобно набирать на дополнительной цифровой ПК-клаве (Numpad), если она имеется; а можно и на основной ПК-клаве набирать.

Коды клавиш, посылаемые в имитаторе терминалом в машину, – 7-битные, с равным нулю битом «паритета» (чётности). Разбор имеющихся программ для Д3-28 показал, что в них бит паритета у принятого кода всегда сбрасывается в ноль, поэтому я не стал пока формировать бит паритета, чтобы не усложнять и без того громоздкую процедуру анализа клавишных кодов. Увидеть коды клавиш, посылаемые в имитаторе из терминала в машину, можно с помощью программки display_codes_KP-1689_N-106.txt. (Считывание: С, СЛ; запуск: С, S). Она при нажатии каждой клавиши выводит на экран символ, если клавиша символьная, и четыре цифры байт-кода клавиши в тетрадно-десятичной форме; коды команд отрабатываются, а затем выводятся на экран. Вывод идёт столбиком.

Потренироваться набирать буквы и символы, а также управлять дисплеем при разных параметрах служебной строки можно, когда он отключен от «линии» связи с машиной, то есть находится «в автономном режиме». Как уже пояснялось, в проге подключение терминала к линии имитируется ПК-клавишей <F5>.

Когда машина находится в программном режиме, выключение ДУП-ЛИН (клавишей <F5>) проявляется как пауза в выводе программой данных на экран; так можно вызывать паузу специально, когда захочется. В паузе набор символов с клавиатуры изменит картинку на экране без влияния на дальнейшую работу программы после включения ДУП-ЛИН (если не включать режим блочной передачи, см. ниже). Выйти из программного режима, т. е. перевести машину в состояние «останов с индикацией», можно кнопкой С или Ш на её пульте (<$С_{лат}$> или <Enter> в имитаторе, см. раздел 7). При выключенных ДУП и ЛИН терминал находится в автономном режиме и отрабатывает на экране дисплея нажатия клавиш. Такое состояние с остановленной машиной реализуется также сразу после запуска или ресета имитатора.

Если машина не работает по программе, то после включения ДУП и ЛИН (ПК-клавишей <F5>) печатание символов не отрабатывается дисплеем, потому что их коды никакой программой не принимаются и не отправляются обратно в дисплей. Если теперь выключить ДУП (в имитаторе это доступно при включённой ЛИН, роль ДУП выполняет <F6>), то коды

клавиш будут поступать не только в «линию», но и в дисплей, и, следовательно, отобразятся на экране. Если же машина выполняет программу, которая принимает и отправляет коды клавиш в терминал, то при выключенной ДУП они отобразятся на экране дисплея дважды (а если программа не отправляет коды, то – один раз).

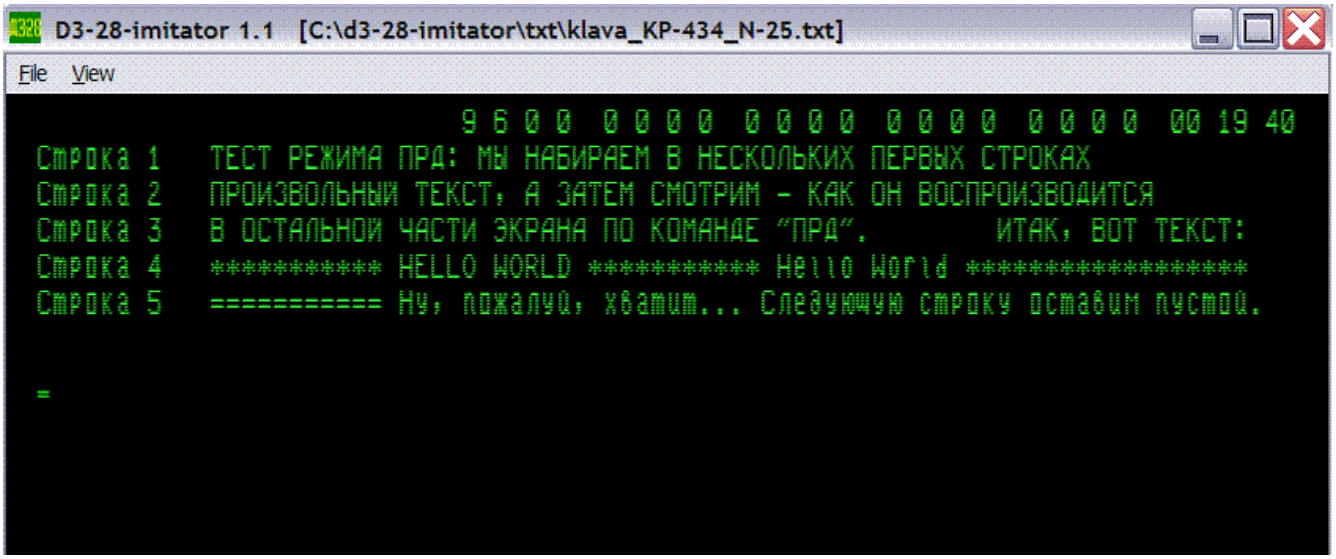
Всё это можно пронаблюдать и осмыслить с помощью коротенькой программки klava_KP-434_N-25.txt (считывание: С, СЛ, запуск: С, S). Программка активирует контроллер дисплея передачей в него его «вектора» 1515 при состоянии шины УПР 1507 и затем ожидает поступления байт-кода из терминала. Нажимая клавишу, мы посылаем в «линию» её код. Приняв этот код, программка отправляет его по «линии» обратно в терминал и ожидает поступления следующего байта; и так далее. Поэтому при включенных кнопках ДУП-ЛИН и работающей программке наши нажатия клавиш отрабатываются дисплеем так же, как они отрабатывались бы с выключенными ДУП-ЛИН (т. е. в автономном режиме, в том числе и без программки). Если теперь выключить ДУП, то коды будут отрабатываться дисплеем дважды.

В версиях имитатора до октября 2019 г. имитация ОЗУ дисплея была упрощённой: имитировалась только одна экранная страница (24 строки, по 80 знакомест в строке). В октябрьской 2019 г. версии этот недостаток устранён. Теперь имитируются две страницы ОЗУ: видимый экран и «невидимый» экран, образующие «кольцевой буфер». Невидимая область сдвигается в видимую командами сдвига текста вниз (код 0106, т. е. СУ + V или клавиша ↧) и сдвига текста вверх (код 0102, т. е. СУ + R или клавиша ↥); они отрабатываются в системе команд №1. «Художественный» вывод на дисплей иллюстрирует программка demo_15ie-00-013__KP-50120_N-3546.txt; её описание см. в этом же txt-файле.

Блочная передача

Начиная с осени 2018, в порядке испытания в имитаторе стали доступны режимы блочной передачи кодов из терминала: ПРД – передача экранной страницы (24 строки), ПРС – передача одной строки. Эти режимы кратко описаны в упомянутой выше документации к терминалу; о деталях см. https://www.phantom.sannata.org/viewtopic.php?p=421275#p421275 Работу ПРД легко пояснить с помощью упомянутой выше программки klava_KP-434_N-25.txt.

Сначала, запустив имитатор и эту программку, наберём несколько (например, пять) строк произвольного текста:



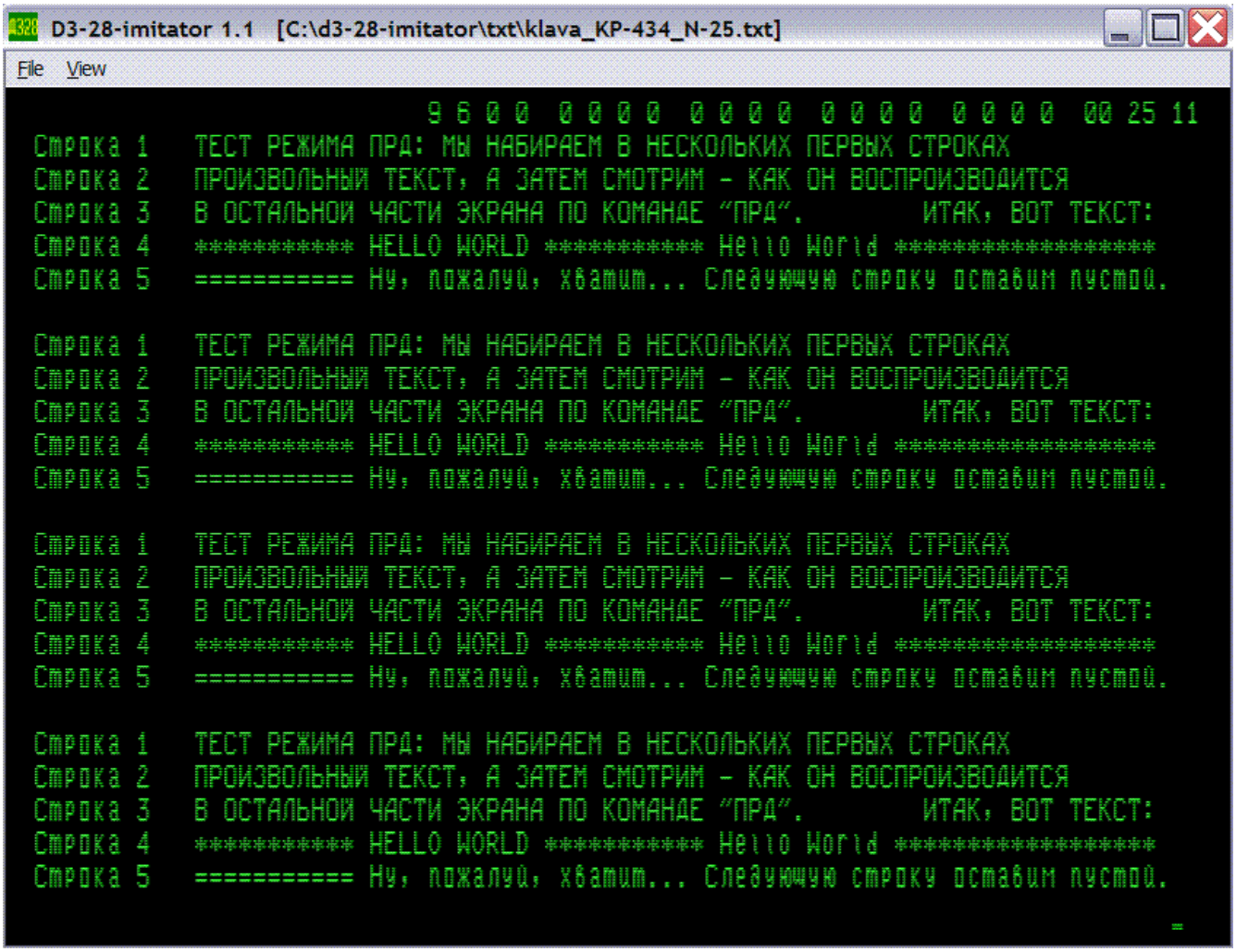
Здесь набор текста можно делать, в том числе, с отключенными ДУП-ЛИН. Чтобы работали команды перемещения курсора в произвольное место экрана клавишами со стрелками (удобные для редактирования в процессе набора текста), в служебной строке должна быть указана система команд №1. Затем ставим курсор в желаемое знакоместо, с которого наша программка должна будет выводить на экран символы, получаемые с экрана в блочной передаче. В режиме ПРД передача начнётся с первого знакоместа в 1-й строке; в нашем примере принимающая программка будет выводить принятые символы, начиная с первого знакоместа в 7-й строке.

Кнопка «ПРД» в имитаторе отсутствует. Чтобы подать команду ПРД, мы, после «нажатия СУ», в режиме просмотра дисплея «нажимаем Ч», т. е. – ПК-клавишу <~> после того, как зажгли индикатор НР и погасили ЛАТ. При этом клавиши ДУП-ЛИН тоже должны быть заранее нажаты. Код ПРД это 0114, он исполняется в системе команд №1. При нажатой СУ такой код образуется из 0714 – это код «Ч» регистра РУС, либо из 0514 – это код «ч» в регистре РУС или ^ в регистре ЛАТ, либо из 0314 – это код > в любом из регистров РУС, ЛАТ. Результат зависит от выбора РУС / ЛАТ, так как от него зависит начальное состояние имитации.

Если в процессе начавшейся блочной передачи нажать <F1>, то увидим, что индикатор ПРД горит (или время от времени мигает). Вот соответствующий скриншот, на нем видно также то положение функциональных клавиш терминала, при котором подавалась команда ПРД:

В нашем примере после подачи команды ПРД сначала происходит передача первых 6 строк, которые программка отображает в строках 7 – 12. Затем этот блок символов повторяется в следующих шести строках, и так далее. Действительно, поскольку на экране помещается 24 строки (не считая служебной строки), и 24 строки должны передаваться при ПРД, то под первыми шестью строками, набранными вручную, помещается три таких блока. А затем после передачи каждой очередной строки происходит скроллинг, изменяющий картину на экране,

поэтому принятый последний блок имеет уже иной вид. Результат, образовавшийся до скроллинга, показан на следующем скриншоте:



```
                                   9 6 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  00 25 11
 Строка 1   ТЕСТ РЕЖИМА ПРД: МЫ НАБИРАЕМ В НЕСКОЛЬКИХ ПЕРВЫХ СТРОКАХ
 Строка 2   ПРОИЗВОЛЬНЫЙ ТЕКСТ, А ЗАТЕМ СМОТРИМ - КАК ОН ВОСПРОИЗВОДИТСЯ
 Строка 3   В ОСТАЛЬНОЙ ЧАСТИ ЭКРАНА ПО КОМАНДЕ "ПРД".        ИТАК, ВОТ ТЕКСТ:
 Строка 4   *********** HELLO WORLD *********** Hello World ******************
 Строка 5   =========== Ну, пожалуй, хватит... Следующую строку оставим пустой.

 Строка 1   ТЕСТ РЕЖИМА ПРД: МЫ НАБИРАЕМ В НЕСКОЛЬКИХ ПЕРВЫХ СТРОКАХ
 Строка 2   ПРОИЗВОЛЬНЫЙ ТЕКСТ, А ЗАТЕМ СМОТРИМ - КАК ОН ВОСПРОИЗВОДИТСЯ
 Строка 3   В ОСТАЛЬНОЙ ЧАСТИ ЭКРАНА ПО КОМАНДЕ "ПРД".        ИТАК, ВОТ ТЕКСТ:
 Строка 4   *********** HELLO WORLD *********** Hello World ******************
 Строка 5   =========== Ну, пожалуй, хватит... Следующую строку оставим пустой.

 Строка 1   ТЕСТ РЕЖИМА ПРД: МЫ НАБИРАЕМ В НЕСКОЛЬКИХ ПЕРВЫХ СТРОКАХ
 Строка 2   ПРОИЗВОЛЬНЫЙ ТЕКСТ, А ЗАТЕМ СМОТРИМ - КАК ОН ВОСПРОИЗВОДИТСЯ
 Строка 3   В ОСТАЛЬНОЙ ЧАСТИ ЭКРАНА ПО КОМАНДЕ "ПРД".        ИТАК, ВОТ ТЕКСТ:
 Строка 4   *********** HELLO WORLD *********** Hello World ******************
 Строка 5   =========== Ну, пожалуй, хватит... Следующую строку оставим пустой.

 Строка 1   ТЕСТ РЕЖИМА ПРД: МЫ НАБИРАЕМ В НЕСКОЛЬКИХ ПЕРВЫХ СТРОКАХ
 Строка 2   ПРОИЗВОЛЬНЫЙ ТЕКСТ, А ЗАТЕМ СМОТРИМ - КАК ОН ВОСПРОИЗВОДИТСЯ
 Строка 3   В ОСТАЛЬНОЙ ЧАСТИ ЭКРАНА ПО КОМАНДЕ "ПРД".        ИТАК, ВОТ ТЕКСТ:
 Строка 4   *********** HELLO WORLD *********** Hello World ******************
 Строка 5   =========== Ну, пожалуй, хватит... Следующую строку оставим пустой.
```

Реальный терминал ведёт блочную передачу с фиксированной скоростью, указанной в служебной строке (в данном примере – 9600 бит в секунду), и после того, как передаст в линию связи с внешним устройством символы из всех 24 строк экрана, он выходит из режима ПРД вне зависимости от того, успела или нет принимающая программа во внешнем устройстве обработать этот поток данных.

Однако, в имитаторе нет должного тайминга, поэтому выход из ПРД реализован иначе. Используется то обстоятельство, что имитатор при работе с любой программой циклически выбирает очередную команду программы из ОЗУ машины и выполняет её. Если программа в течение 1000 таких циклов не примет байт из буфера «линии», передаваемый дисплеем в режиме ПРД, то он заменяется следующим байтом и теряется. (Если же программа быстро забирает байт из буфера, то блочная передача осуществляется довольно быстро и успешно). Когда таким образом будут переданы в «линию» все подлежащие передаче байты символов с экрана, режим ПРД выключается. С остановленной машиной ПРД в имитаторе не включается.

Выйти из ПРД досрочно в имитаторе можно несколькими способами: клавишей <F5> выключить ДУП-ЛИН (затем можно снова включить, это самый безобидный способ); либо – клавишей <F9> выполнить «очистку дисплея» (при этом на экране всё сотрётся); либо – перейти к просмотру машины и остановить её клавишей <С$_{\text{лат}}$> или <Enter>. (Кроме того, в имитаторе

сделана попытка реализации команд «прекратить передачу» и «возобновить передачу», но её работоспособность ещё не проверена.) Полезно также знать, что в свёрнутом состоянии и при просмотре клавиатуры терминала (клавишей <F1>) работа имитатора резко замедляется, и тогда многошаговые операции, в том числе ПРД, занимают много времени.

К особенностям ПРД в имитаторе, по-видимому, надо отнести и передачу команд ЛАТ (0015) или РУС (0014). Каждый раз, когда должен быть передан символ с изменившимся признаком РУС/ЛАТ по сравнению с предыдущим символом, имитатор сначала передает соответственно код ЛАТ или РУС и лишь затем код символа. Эти команды не учитываются при подсчёте количества переданных знакомест. Также не влияют на подсчёт и команды ПС с ВК, передаваемые в конце каждой строки (они передаются, если в служебной строке не установлена «1» в разряде «авто ПС и ВК при передаче»). Пробелы в конце строк имитатор старается не передавать при ПРД, но, конечно, включает их в число переданных символов.

В том же сеансе работы с программкой klava_KP-434_N-25.txt можно посмотреть, как выполняется команда ПРС (её код: 0006). Для этого устанавливаем курсор в начало какой-нибудь строки с текстом, и после «нажатия СУ» в режиме просмотра дисплея нажимаем <F> (без СУ эта клавиша даёт код 0606 в регистре РУС или 0406 в регистре ЛАТ). По этой команде символы строки, в которой установлен курсор, поочерёдно посылаются дисплеем в машину, а наша программка в машине возвращает их обратно в дисплей. В результате мы видим, что строка на экране не изменяется, а вдоль неё лишь перемещается курсор; в конце строки он останавливается (так как при ПРС не передаются ПС и ВК в конце строки). Если перед выполнением ПРС курсор установить в произвольное место строки, то по команде «ПРС» начало строки передастся в это место, и строка может измениться.

Выполнение блока 4 в тесте «061»

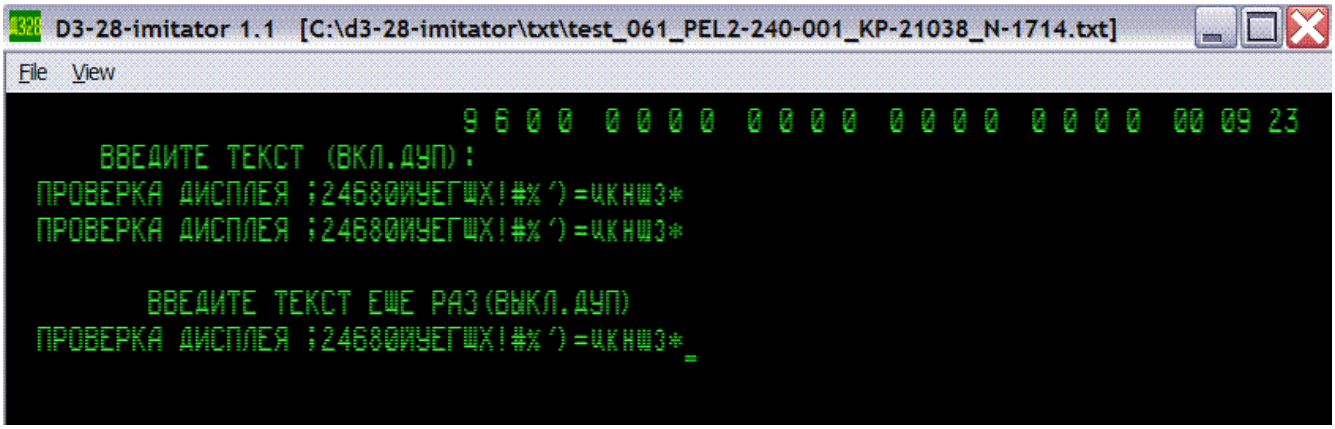
Блок 4 в тест-программе 0.073.061 называется «проверка взаимодействия Д3-28 и дисплея 15ИЭ-00-013». Сканы описания этой тест-программы есть на сайте **Виталия К** : https://d3-28.ru/programmy/test-programma-0-073-061/ и в библиотеке сайта **Alexey17** : http://retropc.org/index.html?action=w_razdela&id_razdel=31 (находятся поиском по 0.073.061). Выполнение этого теста можно рассматривать как проверку имитатора и одновременно хорошее упражнение по управлению дисплеем, в том числе – клавишами ДУП и РЕД.

В процессе работы эта программа попытается вести отсчёт интервала времени 5 сек с помощью машинной команды PAUSER, поэтому в имитаторе следует включить зависимость этой команды от содержимого регистра R_{10} – в режиме просмотра дисплея надо нажать <F1>, затем <D>, ответить «нет», и затем на вопрос «change PAUSER?» ответить «да» (т. е. включить опцию «PAUSER depends on R10»; см. раздел 8). Следует также включить ТПУ нажатием <Home>; до открытия файла с программой через меню «File» ТПУ создает свой txt-файл в той папке, в которой находится exe-файл имитатора, см. раздел 4. И заодно следует заранее включить ДУП-ЛИН. Параметры служебной строки надо сбросить в ноль: нажимаем <F8>, <F9>, <F8>. Затем переходим к просмотру машины, и командами С, СЛ, С загружаем открытый через меню «File» файл test_061_PEL2-240-001_KP-21038_N-1714.txt – это модификация тест-программы с изменениями, указанными на стр. 38 описания теста «для случая подключения дисплеев к Д3-28 через устройство согласующее ПЕЛ2.240.001». Запускаем тест и выбираем в нём 4-й блок: <$Т_{лат}$>, <S>, <4>, <S>.

Вернувшись в режим просмотра дисплея, набираем строку символов по образцу на экране. Описание теста подсказывает, что при этом должны быть нажаты ЛИН, ДУП, РЕД, причём набор символов до X включительно ведётся с погашенными индикаторами ЛАТ и НР,

а дальше – с включённым НР; однако из-за специфики нашей имитации клавиатуры символы ' и = в имитаторе набираются с погашенным НР. Закончив набор строки, гасим НР и нажимаем ПС.

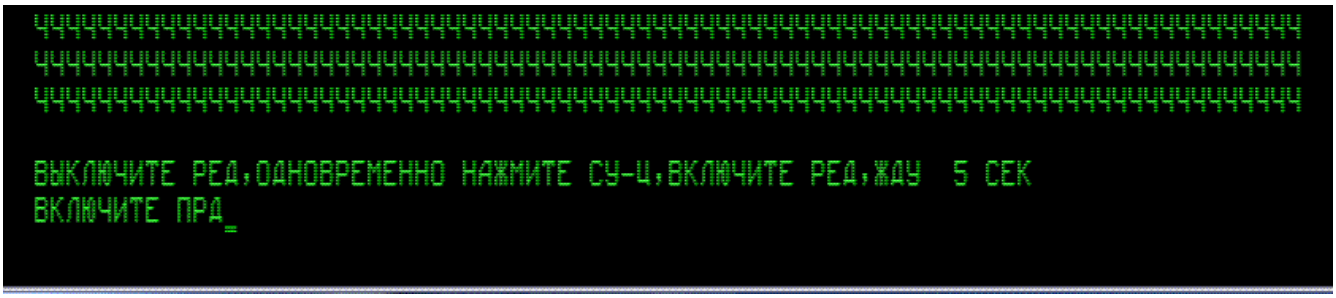
На этом этапе программка предлагает выключить ДУП (т. е. надо нажать <F6>) и повторить набор этих же символов ещё раз. С включённым ДУП набирать пришлось бы «вслепую», так как на этом этапе программа не возвращает принятые коды обратно в дисплей. С выключенным же ДУП коды набираемых символов из клавиатуры терминала поступают и в программу в машине через «линию», и в дисплей; вот так выглядит результат:



Гасим НР и нажимаем ПС. Если ошибок не было, то программка очищает экран и выводит 20 строк с символами Ч (по 80 символов в строке, всего 1600 символов), и затем предлагает в течение 5 секунд выключить РЕД, нажать СУ и Ц, включить РЕД. Это надо проделать быстро. Если промедлить, то мы не увидим на экране дальнейшей реакции, а перейдя к просмотру машины, увидим на табло машины финишную заставку теста:

Машина при этом оказывается остановленной (в чём легко убедиться, нажимая <V> или <W>, – будет виден переход в режим «Ввод»; нажатие <R> – возврат в режим «Работа»). ТПУ печатает сообщение о невыполнении теста и номер ошибки. В данном случае, вероятнее всего, это будет «ОШИБКА 1», означающая отсутствие ответа терминала в интервале времени ожидания. Тест-программу придётся запускать заново (<C$_{лат}$>, <T$_{лат}$>, <S>, <4>, <S>).

При удачном выполнении предписания «выкл. РЕД, СУ-Ц, вкл. РЕД» на экране появляется строка с текстом «ВКЛЮЧИТЕ ПРД»; вот скриншот, показывающий нижнюю часть экрана на этой стадии (с тремя последними строками, заполненными символом Ч):



```
ЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧ
ЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧ
ЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧЧ


ВЫКЛЮЧИТЕ РЕД,ОДНОВРЕМЕННО НАЖМИТЕ СУ-Ц,ВКЛЮЧИТЕ РЕД,ЖДУ  5 СЕК
ВКЛЮЧИТЕ ПРД_
```

Включать ПРД программа нас не торопит, поэтому можно спокойно вдуматься в смысл предыдущего предписания. Как поясняется в документации дисплея, при выключенной клавише РЕД (т. е. в режиме «не РЕД») дисплей не отрабатывает команды, он лишь заносит в своё ОЗУ их коды в виде латинских букв. Эти буквы должны мерцать с частотой 5 Гц в режиме «не РЕД», а в режиме с включённой РЕД они не видны – на их месте изображаются пробелы. В данном примере выключив РЕД (клавишей <F7>), мы должны увидеть в начале строки под последней строкой с буквами Ч мерцающую латинскую букву С; это символическое изображение байт-кода 0003. Считается, что код 0003 означает команду «КТ – конец текста». Но здесь это не важно: фрязинский дисплей не отрабатывает код 0003, и авторы теста могли бы вместо него выбрать другой байт, так как далее в тесте проверяется лишь, что этот байт записался в ОЗУ дисплея на знакоместо после 1600-го символа Ч.

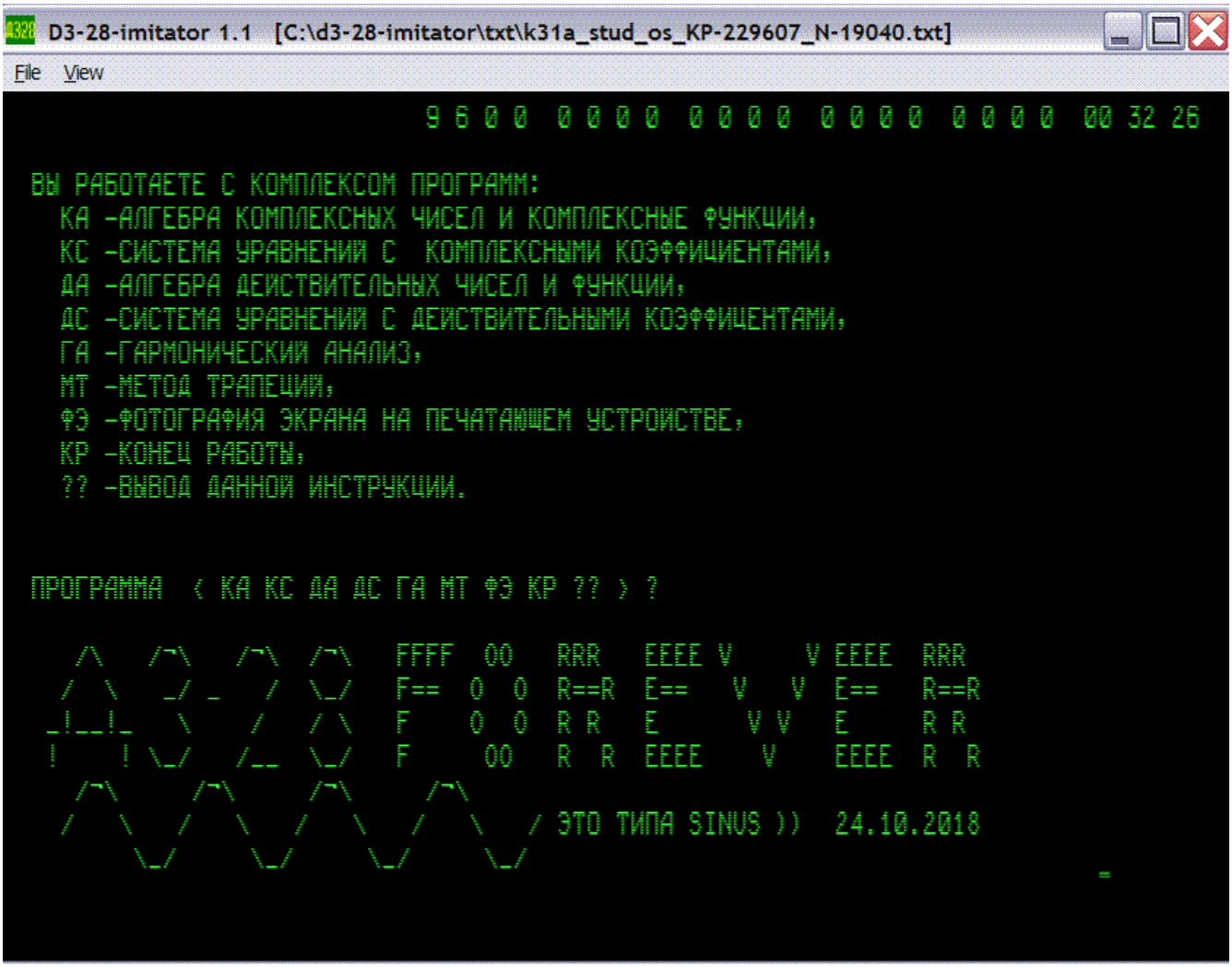
Подать команду ПРД здесь можно так же, как уже описывалось выше (с тем только отличием, что теперь клавиша ДУП остаётся выключенной) – с «нажатой СУ» и с включенным НР нажимаем ПК-клавишу <~>. Выключенная ДУП здесь существенна, так как на этом этапе тест-программа не отправляет обратно в дисплей принятые с клавиатуры коды. Дисплей, получив код 0114 команды ПРД, начинает блочную передачу символов с экрана в «линию», а тест-программа, получив код 0114, начинает принимать и подсчитывать поступающие из «линии» байты. При этом она проверяет, что поступают только коды символа Ч и коды ПС и ВК, которыми завершаются строки, и, наконец, что после ПС и ВК, завершающих строку с 1600-м символом Ч, приходит байт 0003. Длится это довольно долго, никак не проявляясь на экране; затем экран очищается, машина останавливается и индицирует на своём табло финишную заставку. В txt-файле ТПУ появляется сообщение об успешном выполнении 4-го блока теста:

```
ПРОГРАММА ПРОВЕРКИ 4

   4    ТЕСТ ВЫПОЛНЕН ПРАВИЛЬНО

ВЫПОЛНЕНИЕ ТЕСТА ЗАКОНЧЕНО
КОНТРОЛЕР (ФИО,ПОДПИСЬ,ШТАМП)
```

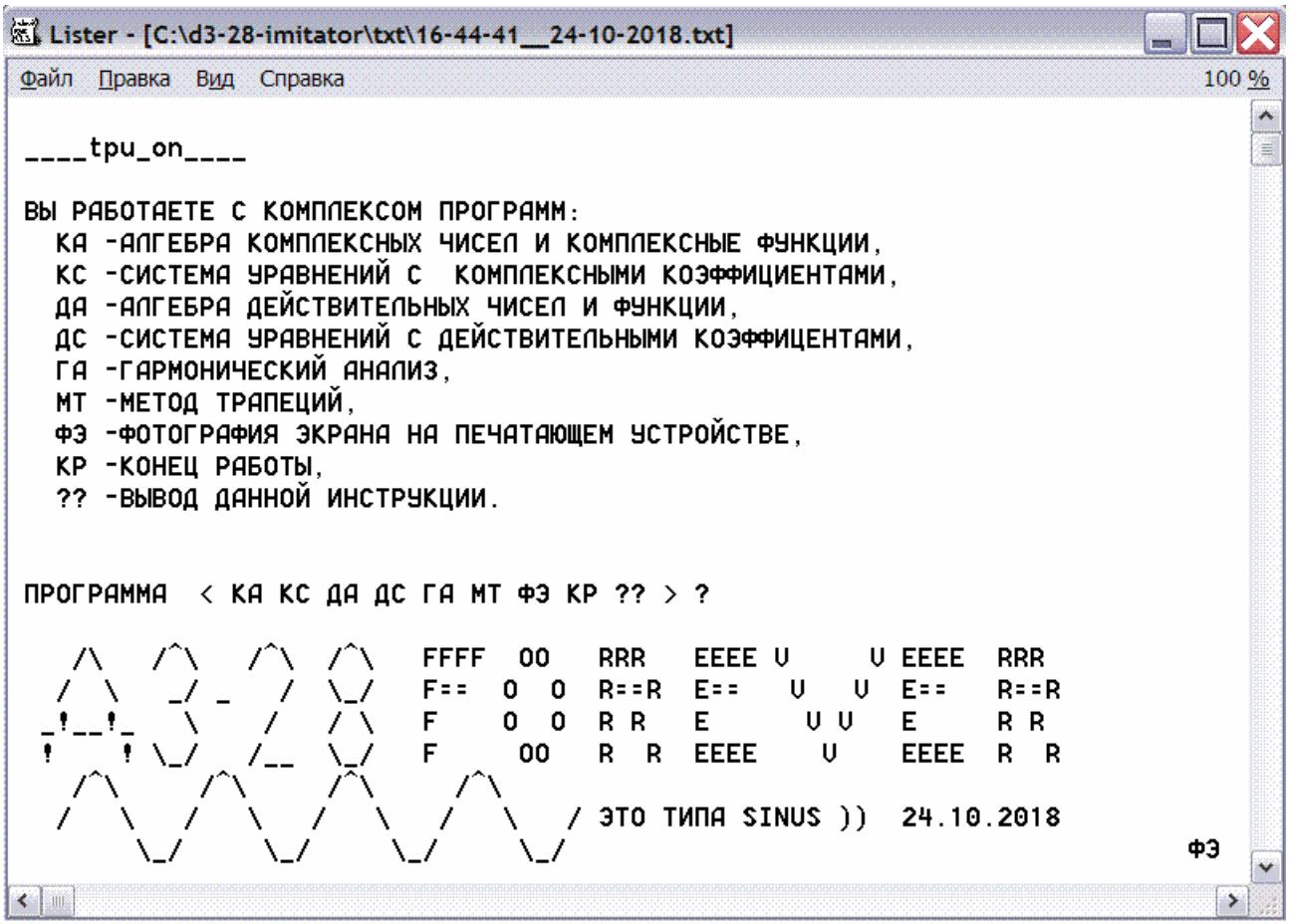
«Фотография экрана» блочной передачей с экрана на ТПУ

Ещё одна программа, позволяющая проверить и продемонстрировать работу блочной передачи в нашем имитаторе – k31a_stud_os_KP-229607_N-19040.txt. Оцифровку этой программы из своей коллекции кассет выполнил и любезно предоставил **Виталий К**, автор сайта d3-28.ru. Загрузка (разумеется, в свежезапущенный или освежённый ресетом имитатор, как и в предыдущих примерах): СЛ, КП (необязательная проверка контрольной суммы), С. Запуск: S, переходим к просмотру дисплея, сбрасываем в ноль разряд служебной строки «авто ПС с ВК»: <F8>, →, →, ↓, <F8>, включаем ТПУ: <Home> (при этом ТПУ создаёт свой txt-файл в той же папке, из которой открывался txt-файл с программкой), включаем ДУП-ЛИН: <F5>. Программка выводит на экран своё «меню».

В этом меню есть пункт: ФЭ –ФОТОГРАФИЯ ЭКРАНА НА ПЕЧАТАЮЩЕМ УСТРОЙСТВЕ. В конце, ниже приглашения ПРОГРАММА < КА КС ДА ДС ГА МТ ФЭ КР ?? > ? мы попробуем набрать свой произвольный текст и затем директивой ФЭ вывести его вместе с исходным меню на ТПУ. С этой целью выключаем ДУП-ЛИН (чтобы перевести дисплей в автономный режим), и в служебной строке сбрасываем в ноль параметр системы команд (чтобы перейти в систему команд №1 и тем самым получить доступ к дисплейным командам редактирования экрана): <F8>, ←, ↓, <F8>. Затем пишем или «рисуем» что-нибудь; закончив, устанавливаем курсор где-нибудь в правой нижней части экрана – там, где собираемся напечатать команду ФЭ:



Перед подачей команды ФЭ – возвращаем дисплей в систему команд №2: <F8>, ↓, <F8>; проверяем (просмотром клавиатуры терминала по нажатию <F1>) индикаторы ЛАТ и НР, а также «нажатие СУ» – гасим их, если они горят; включаем ДУП-ЛИН; и, вернувшись к изображению экрана, набираем ФЭ. Программа, по-видимому, сама включает с. к. №1 для блочной передачи, инициирует блочную передачу всего экрана, и в новой строке внизу экрана выводит очередное приглашение ПРОГРАММА < КА КС ДА ДС ГА МТ ФЭ КР ?? > ? При этом в txt-файле, имитирующем распечатку на ТПУ, появляется желаемая «фотография экрана»:



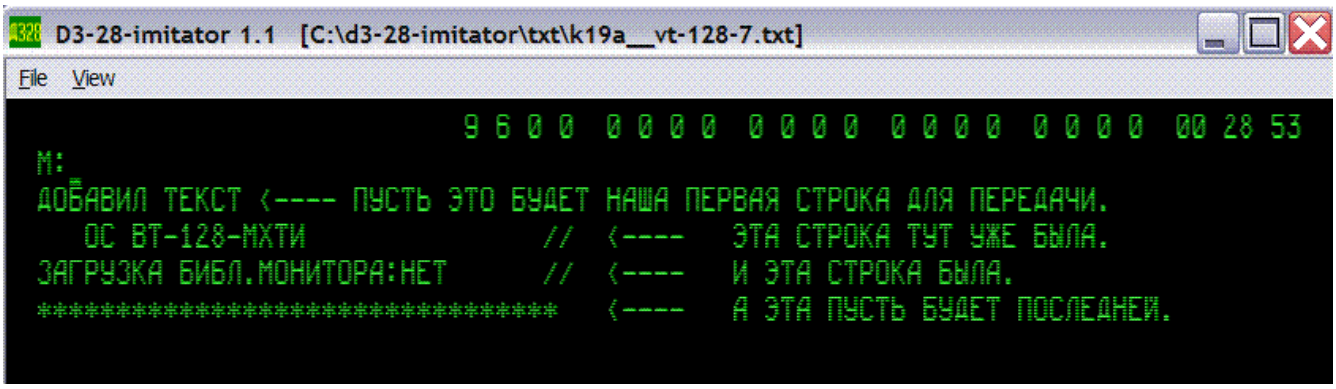
Блочная передача содержимого экрана на ТПУ предусмотрена и в программах «ОС ВТ-128-МХТИ». В описании, полученном из оцифровки кассеты номер 20 в коллекции уважаемого Виталия К, говорится, как действовать после того, как информация на экране будет подготовлена к передаче на ТПУ:

1. директивой МОС: «ПРД» курсор устанавливается на верхней строке экрана.
2. клавишами «курсор вниз» и «курсор вверх» производится установка курсора на первую распечатываемую строку, что фиксируется клавишей «ВК».
3. в ответ на запрос СТРОК: вводится количество распечатываемых строк экрана дисплея и «ВК».

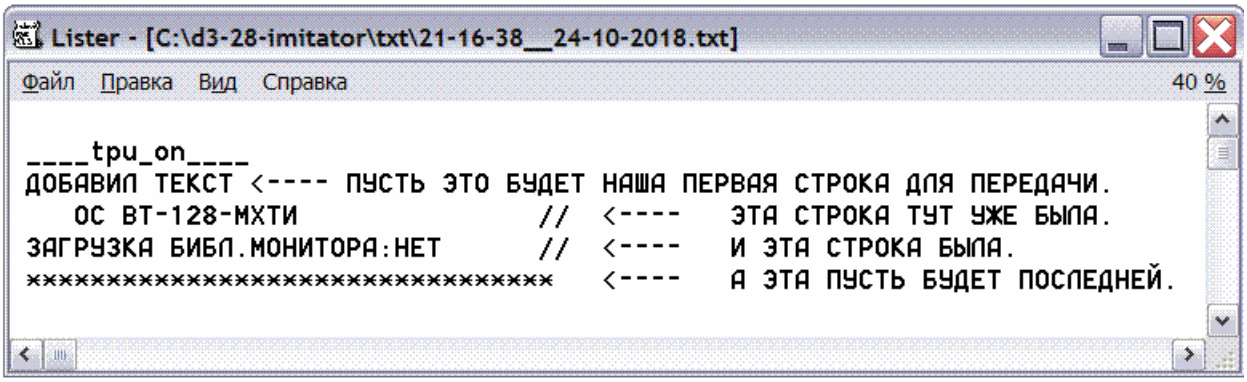
По-видимому, здесь блочная передача реализуется в режиме ПРС, а команда ПРД программой принимается, но в дисплей не возвращается (это предположение, в потроха программы я не углублялся).

Посмотрим, как это работает на примере «ОС ВТ-128-МХТИ с кассеты 19» – файл с этой версией системы k19a__vt-128-7.txt доступен на сайте Виталия К. d3-28.ru: https://d3-28.ru/programmy/vt-128-mhti-operatsionnaya-sistema/

Загрузка и запуск производятся уже хорошо известными нам командами на пульте машины: СЛ, КП (необязательная проверка контрольной суммы загрузчика: 1433), С, S. Переходим к изображению экрана дисплея, сбрасываем в ноль параметры служебной строки: <F8>, <F9>, <F8>, включаем ТПУ: <Home>, включаем ДУП-ЛИН: <F5>. Программа выводит на экран своё имя и вопрос о «загрузке библиотеки монитора»; отвечаем <N>, то есть «нет». В верхней строке появляется приглашение к вводу директив: «М:». Выключаем ДУП-ЛИН и печатаем на экране что-нибудь для передачи в ТПУ. Для краткости в этом примере не станем разрисовывать весь экран (хотя можно было бы :-), лишь добавим несколько строк произвольного текста к тому, что уже есть. Поскольку ТПУ не печатает маленькие буквы, пользуемся только большими буквами:



Включаем ДУП-ЛИН; и даём команду ПРД, т. е. нажимаем <CapsLock>, <Shift>, <~>, что означает «нажатие СУ и Ч». Программа выводит в позиции курсора слово ПУСК и устанавливает курсор в начало экрана. Нажимаем <CapsLock>, <Shift>, чтобы выключить СУ и НР. Клавишей ↓ опускаем курсор в начало следующей строки и нажимаем ВК, т. е. <Backspace>. Программа выводит в верхней строке запрос СТРОК:: Вводим число 4, так как у нас приготовлено 4 строки для передачи, и опять нажимаем ВК, т. е. <Backspace>. Курсор бежит сверху вниз по начальным позициям в наших четырёх строках, затем возвращается в верхнюю строку, и программа выводит очередное приглашение к вводу директив «М:». При этом в txt-файле, имитирующем распечатку на ТПУ, появляются переданные с экрана строки:



Реализация режима ПРД, имеющаяся сейчас в имитаторе, успешно проверена также в двух версиях программы, разработанной в 1980-х годах Виталием К., которая представляет собой экранный редактор документов. (Окончательная версия этого редактора документов ещё не оцифрована с кассет, поэтому здесь пока не будем рассматривать эту программу.)

Экранное редактирование бэйсик-программ

Рассмотрим иллюстративный пример одного из наиболее важных применений блочной передачи – экранное редактирование текста пользовательской программки на языке «Бэйсик Д3-28, вариант 3А». Обычное редактирование сводится, чаще всего, к удалению строки или к новому набору с самого начала строки, подлежащей изменению. Удалять-то строку легко – достаточно набрать её номер и нажать ПС; а вот повторный набор – дело скучное и часто сопровождающееся новыми опечатками, исправление которых опять-таки сводится к очередной попытке набора злополучной строки…. Поэтому более удобным представляется редактирование текста на экране дисплея в автономном режиме.

Возьмём знакомую нам версию Бэйсика – с контрольной суммой 157107. Чтобы работать с РУС- и ЛАТ-буквами, надо до запуска Basic_D3-28_v3A__KP-157107__N-11343.txt установить «бит4=1» (см. раздел 5: в режиме просмотра дисплея нажимаем <F1>, <4>, на вопрос «change b4?» отвечаем «да», и затем «нет»). Параметры служебной строки дисплея сбрасываем в ноль: <F8>, <F9>, <F8>. После загрузки и запуска Бэйсика (С, СЛ, С, S) включаем ДУП-ЛИН, и после прохождения начального диалога набираем какую-нибудь программку, которую будем редактировать. Вот пример текста бейсик-программки до редактирования, выведенный на экран командой LIST, и результат исполнения этой программки командой RUN:

```
:LIST

10 REM ЭТО ПРИМЕР ДЛЯ ЭКРАННОГО РЕДАКТИРОВАНИЯ
20 PRINT'TEST'
30 PRINT'*************** ЗДРАВСТВУЙТЕ, ТОВАРИЩИ !   HELLO WORLD,   ПРИВЕТ РЕБЯТ
А,  И ТАК ДАЛЕЕ И ТОМУ ПОДОБНОЕ.'
40 PRINT'КОНЕЦ ПРОГРАММКИ'

:RUN
TEST
*************** ЗДРАВСТВУЙТЕ, ТОВАРИЩИ !   HELLO WORLD,   ПРИВЕТ РЕБЯТА,  И ТАК
 ДАЛЕЕ И ТОМУ ПОДОБНОЕ.
КОНЕЦ ПРОГРАММКИ
ОСТАНОВ В СТРОКЕ  40
:_
```

Для примера, попробуем редактировать, начиная со строки 20 (тем самым проверим, что с предыдущей строкой не случится ничего дурного). План изменений, допустим, такой:

– в короткой строке 20 изменим весь текст;

– в длинной строке 30 попробуем изменить только начало и конец, не перепечатывая всю строку; например, ближе к концу строки добавим спецификатор @1, чтобы последние слова печатались маленькими буквами;

– чтобы в текстах в следующих строках вернуться к большим буквам, добавим строку 35 с оператором PRINT @0;

– добавим новую строку, 45, в которой будут определены новые переменные и какое-нибудь вычисление с ними: тем самым мы проверим возможность добавления методом экранного редактирования сложной строки, которую Бэйсик должен будет не только принять в свою память, но и вставить в ней операторы LET;

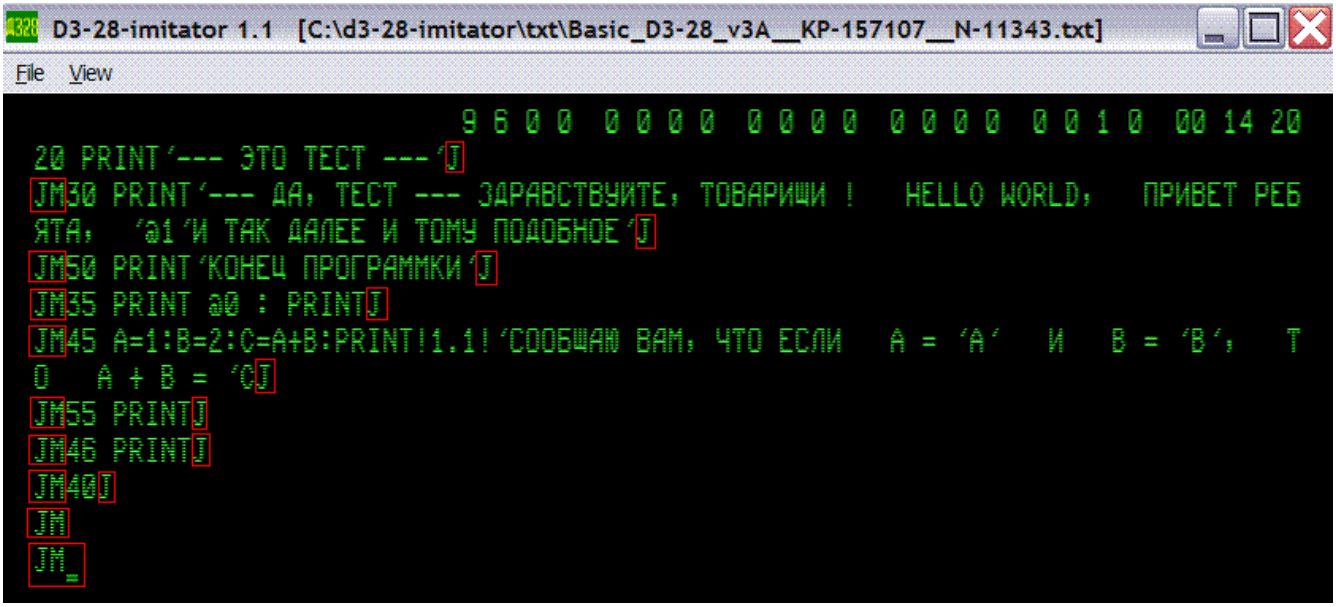
– добавим также строки с операторами PRINT, чтобы при выводе на экран появились пробелы между строками с текстом (такой «разреженный» вывод приятнее читается);

– в строке 40 изменим её номер на 50. Это будет проверка возможности изменять только лишь нумерацию строк, без изменения их содержимого. При этом старую строку удалим, указав лишь её номер.

Для редактирования сначала подаём команду LIST с указанием начального и конечного номеров строк нужного фрагмента программы. Затем выключаем ДУП-ЛИН. Разряд «авто ПС и ВК при ПРД» устанавливаем в «1» – этим действием отключается автоматическая передача команд ПС и ВК в конце каждой 80-символьной строки; так надо делать для передачи длинных строк (содержащих более 80 символов; Бэйсик допускает строки длиной 120 символов).

Затем путём ввода пустых строк или командой смещения текста вверх сдвигаем текст на экране так, чтобы первая редактируемая строка оказалась самой верхней (см. следующий скриншот). И добиваемся желаемых изменений текста. Для смещения какой-либо строки или её части вправо от курсора пользуемся командой «размыкание» (код 0103): включив <CapsLock>, нажимаем <S>. Аналогично, сдвиг влево реализуется командой «смыкание» (код 0104): с включенной <CapsLock> нажимаем <$Т_{лат}$>.

С выключенной (отжатой) клавишей РЕД (в имитаторе это <F7>) нажимаем <Enter> и <Backspace> так, чтобы в конце каждой строки появился мерцающий символ перевода строки J, и чтобы в начале следующей строки появились мерцающие символы перевода строки и возврата каретки: JM. Те символы, которые должны мерцать в режиме «не РЕД», на скриншоте обведены красной рамкой:



Закончив всё это, включаем РЕД. При этом на месте мерцавших символов появятся пробелы. Включаем ДУП-ЛИН; и даём в режиме РУС команду ПРД, т. е.: включаем <CapsLock> – это означает «нажатие СУ», включаем режим НР и РУС, и нажимаем <~> (это – «нажатие Ч»).

Бейсик ругается – выдает непонятный текст и множество сообщений об ошибках:

```
  45 A=1:B=2:C=A+B:PRINT!1.1!'СООБЩАЮ ВАМ, ЧТО ЕСЛИ   A = 'A'   И   B = 'B',   T
O   A + B = 'C
  55 PRINT
  46 PRINT
  40

:'-     'E'
:5 '
:m
:'И И Оц
:m
:m
:0
 :
ОШИБКА   2  В СТРОКЕ   0

ОШИБКА   2  В СТРОКЕ   0

ОШИБКА   2  В СТРОКЕ   0
```

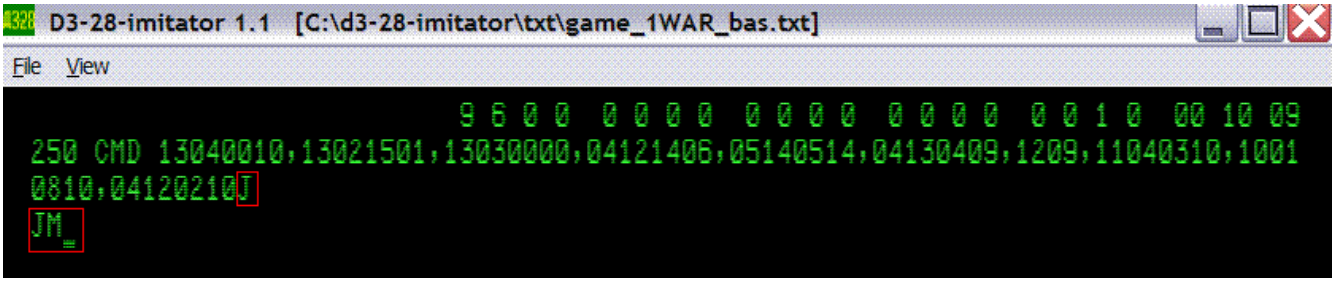
Однако команды LIST и RUN показывают, что желаемое изменение программки состоялось:

```
:LIST

10 REM ЭТО ПРИМЕР ДЛЯ ЭКРАННОГО РЕДАКТИРОВАНИЯ
20 PRINT'--- ЭТО ТЕСТ ---'
30 PRINT'--- ДА, ТЕСТ --- ЗДРАВСТВУЙТЕ, ТОВАРИЩИ !   HELLO WORLD,   ПРИВЕТ РЕБЯТ
А,  '@1'И ТАК ДАЛЕЕ И ТОМУ ПОДОБНОЕ'
35 PRINT @0 : PRINT
45 LETA=1:LETB=2:LETC=A+B:PRINT!1.1!'СООБЩАЮ ВАМ, ЧТО ЕСЛИ   A = 'A'   И   B = '
B',   ТО   A + B = 'C
46 PRINT
50 PRINT'КОНЕЦ ПРОГРАММКИ'
55 PRINT

:RUN
--- ЭТО ТЕСТ ---
--- ДА, ТЕСТ --- ЗДРАВСТВУЙТЕ, ТОВАРИЩИ !   HELLO WORLD,   ПРИВЕТ РЕБЯТА,  и так
 далее и тому подобное

СООБЩАЮ ВАМ, ЧТО ЕСЛИ   A =  1.0   И   B =  2.0,   ТО   A + B =  3.0

КОНЕЦ ПРОГРАММКИ

ОСТАНОВ В СТРОКЕ  55
:_
```

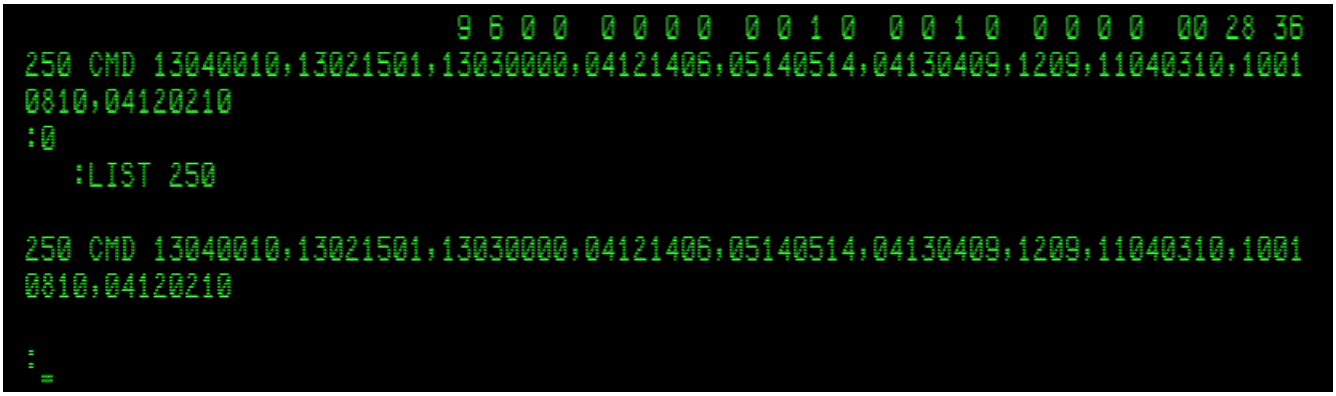
Ещё один пример экранного редактирования бейсик-программы – подбор в тест-игре game_1WAR_bas.txt наиболее приемлемой (для конкретного пользователя на конкретном ПК) скорости «точки-истребителя». С этой целью изменим несколько цифр в начале строки 250.

Строка 250 предписывает машине опрашивать терминал командой 0412 1406, ожидающей сигнал только в конечном интервале времени (команда INPOWS; см. «*Справочник программиста 3.857.100 Д3*»). Если «синхроимпульс периферии» (СИП), подтверждающий передачу байт-кода из клавиатуры терминала, приходит в заданном промежутке времени от начала исполнения INPOWS, то программка анализирует принятый код клавиши и управляет ходом игры. Если же СИП не поступает, то «точка-истребитель» выводится в очередную позицию на прежней траектории. Таким образом, чем больше время ожидания СИП, тем медленнее движется «точка-истребитель». Время ожидания СИП определяется 2-байтовым числом в регистре R_{10}, состоящем из регистров S_4 (старший байт) и S_5 (младший байт). При ненулевом S_4 значение S_5 уже не сильно влияет на темп игры. Поэтому в программке темп игры управляется только числом в S_4: командой 1304 0106 в начале строки 250 в регистр S_4 засылается число 0106, подобранное опытным путём при наладке программки.

Попробуем заменить указанное значение 0106 меньшим числом: 0010. В данной программке нет РУС-букв, поэтому все действия можно выполнить в режиме с «b4=0»; он включается сам при запуске имитатора. Как загрузить Бейсик-157107 и game_1WAR_bas.txt рассказывалось в разделе 1. Затем подаем команду LIST 250. Выключаем ДУП-ЛИН, задаём параметры служебной строки в соответствии с показанным выше скриншотом (см. также скриншот ниже), и выполняем желаемое редактирование, как уже пояснялось:



Затем включаем РЕД, ДУП-ЛИН; и даём команду ПРД в режиме ЛАТ, т. е. включаем <CapsLock>, включаем режим НР (и ЛАТ, если он был выключен), и нажимаем <^>. На этот раз Бэйсик ведёт себя смирно – просто выводит нолик и двоеточия. Выключив <CapsLock>, НР, и вернувшись к исходным параметрам служебной строки, убеждаемся командой LIST 250, что желаемое изменение начала строки 250 состоялось:



«Истребитель» теперь будет летать быстрее, но менее послушно, так как уменьшилось время ожидания кода управления. Повторяя редактирование, можно нащупать какой-то оптимум.

Такие опыты показывают, что экранное редактирование бэйсик-программок не всегда получаются на 100% успешным; итог иногда зависит от содержания редактируемых строк. Иногда теряется символ в длинной строке с частым чередованием букв и пробелов (затем такую строку можно исправить повторным экранным редактированием). Удаление строки иногда сопровождается потерей следующей строки; поэтому, удалять строки лучше в обычном режиме, это, к тому же, проще. Почему Бэйсик выдаёт череду сообщений об ошибках – пока тоже мне непонятно. (В описании Бэйсика есть замечание: «4.3. При использовании блока последовательного интерфейса для дисплея 15ИЭ-00-013 режим передачи данных /ПРД/ не поддерживается интерпретатором». Может быть, как-то это обстоятельство проявляется, хотя имитатор и не имитирует работу КД совершенно точно.) Но, в общем, результат позитивный: так ли, сяк ли, экранное редактирование бэйсик-программок в имитаторе с введением имитации режима блочной передачи стало доступным.

Байт-коды дисплейных команд и символов

Ниже для справочных целей приведены байт-коды дисплейных команд и символов в десятичной и в тетрадно-десятичной 16-ричной записи (по описанию из указанной выше документации к дисплею; в документации байт-коды даны в восьмеричной системе счисления).

Десятичное значение кода	16-ричное значение кода	Обозначение команды	Как отрабатывается в системе команд №1	в системе команд №2
0	0000	нет команды	не отрабатывается	
1	0001	С1	не отрабатывается	
2	0002	↓	не отрабатывается	
3	0003		не отрабатывается (притом кое-где упоминается как код «КТ»)	
4	0004		не отрабатывается	
5	0005		в имитаторе не отрабатывается	
6	0006	ПРС	№1: передача строки,	№2: не отрабатывается
7	0007	звонок	в имитаторе не отрабатывается	
8	0008	↖	№1: курсор в начало страницы,	№2: курсор на шаг влево
9	0009	ГТ	№1: не отрабатывается,	№2: горизонтальная табуляция
10	0010	ПС	перевод строки	
11	0011	СТС	№1: стирание от курсора до конца строки,	№2: как ПС
12	0012	СБР	№1: очистка ЗУ дисплея,	№2: как ПС
13	0013	ВК	возврат «каретки»	
14	0014	РУС	переход к рус-символам, т.е. к кодам набора 1 КОИ-7	
15	0015	ЛАТ	переход к лат-символам, т.е. к кодам набора 2 КОИ-7	
16	0100	АР1	не отрабатывается	
17	0101	ПРМ	команда «возобновить передачу текста»	
18	0102	⇧	№1: сдвиг текста вверх на одну строку,	№2: не отрабатывается
19	0103	↦	№1: размыкание (сдвиг текста на шаг вправо от курсора) или прекратить передачу, если она шла.	№2: «прекратить передачу»
20	0104	↤	№1: смыкание (сдвиг на шаг влево до курсора),	№2: не отрабат.
21	0105	↰	№1: как ПС и затем ВК,	№2: не отрабатывается
22	0106	⇩	№1: сдвиг текста вниз на одну строку,	№2: не отрабатывается
23	0107		№1: переход к системе команд «номер 2»,	№2: не отрабат.
24	0108	ТАБ	в имитаторе не отрабатывается	
25	0109	→	№1: курсор на шаг вправо,	№2: не отрабатывается
26	0110	←	№1: курсор на шаг влево,	№2: не отрабатывается
27	0111	АР2	№1: не отрабатывается, №2: как первый байт команды «АР2 …»	
28	0112	↑	№1: курсор на одну строку вверх,	№2: не отрабатывается
29	0113	↓	№1: курсор на одну строку вниз,	№2: не отрабатывается
30	0114	ПРД	№1: передача страницы,	№2: не отрабатывается
31	0115	СТР	№1: стирание от курсора до конца страницы,	№2: не отрабат.

Десятичное значение кода	16-ричное значение кода	Символ
32	0200	пробел
33	0201	!
34	0202	“
35	0203	#
36	0204	¤
37	0205	%
38	0206	&
39	0207	‘
40	0208	(
41	0209	)
42	0210	*
43	0211	+
44	0212	,
45	0213	–
46	0214	.
47	0215	/

Десятичное значение кода	16-ричное значение кода	Символ
48	0300	0
49	0301	1
50	0302	2
51	0303	3
52	0304	4
53	0305	5
54	0306	6
55	0307	7
56	0308	8
57	0309	9
58	0310	:
59	0311	;
60	0312	<
61	0313	=
62	0314	>
63	0315	?

Десятичное значение кода	16-ричное значение кода	Символ в наборе 0	Символ в наборе 1
64	0400	@	*ю*
65	0401	A	*а*
66	0402	B	*б*
67	0403	C	*ц*
68	0404	D	*д*
69	0405	E	*е*
70	0406	F	*ф*
71	0407	G	*г*
72	0408	H	*х*
73	0409	I	*и*
74	0410	J	*й*
75	0411	K	*к*
76	0412	L	*л*
77	0413	M	*м*
78	0414	N	*н*
79	0415	O	*о*
80	0500	P	*п*
81	0501	Q	*я*
82	0502	R	*р*
83	0503	S	*с*
84	0504	T	*т*
85	0505	U	*у*
86	0506	V	*ж*
87	0507	W	*в*
88	0508	X	*ь*
89	0509	Y	*ы*
90	0510	Z	*з*
91	0511	[	*ш*
92	0512	\	*э*
93	0513	]	*щ*
94	0514	^	*ч*
95	0515	_	*ъ*

Десятичное значение кода	16-ричное значение кода	Символ в наборе 0	Символ в наборе 1
96	0600	`	Ю
97	0601	*a*	А
98	0602	*b*	Б
99	0603	*c*	Ц
100	0604	*d*	Д
101	0605	*e*	Е
102	0606	*f*	Ф
103	0607	*g*	Г
104	0608	*h*	Х
105	0609	*i*	И
106	0610	*j*	Й
107	0611	*k*	К
108	0612	*l*	Л
109	0613	*m*	М
110	0614	*n*	Н
111	0615	*o*	О
112	0700	*p*	П
113	0701	*q*	Я
114	0702	*r*	Р
115	0703	*s*	С
116	0704	*t*	Т
117	0705	*u*	У
118	0706	*v*	Ж
119	0707	*w*	В
120	0708	*x*	Ь
121	0709	*y*	Ы
122	0710	*z*	З
123	0711	{	Ш
124	0712	|	Э
125	0713	}	Щ
126	0714	‾	Ч
127	0715	■ (забой)	■ (забой)

2-байтные командные последовательности дисплея 15ИЭ-00-013
(отрабатываются только в системе команд №2)

16-ричный код 2 2-байтный код	Символы команды	Как отрабатывается
0111 0401	АР2 A	Перемещение курсора на одну строку вверх.
0111 0402	АР2 B	Перемещение курсора на одну строку вниз.
0111 0403	АР2 C	Перемещение курсора на один шаг вправо.
0111 0404	АР2 D	Перемещение курсора на один шаг влево.
0111 0405	АР2 E	Переход к системе команд №1.
0111 0408	АР2 H	Перемещение курсора в начало страницы.
0111 0409	АР2 I	Перемещение курсора на одну строку вверх. Если курсор уже в верхней строке, то текст сдвигается вниз на одну строку.
0111 0410	АР2 J	Стирание текста от позиции курсора до конца страницы.
0111 0411	АР2 K	Стирание текста от позиции курсора до конца строки.
0111 0509	АР2 Y	«Прямая адресация курсора»: курсор перемещается в позицию, задаваемую следующими за АР2 Y двумя байтами. Десятичными значениями этих двух байтов должны быть 32+y, 32+x, где y – номер строки (0,1, …, 23), x – номер позиции в строке (0,1, …, 79).

3. Работа с НМЛ

Все действия с меню доступны в любом режиме работы имитатора – при имитации «фрязинского дисплея» (т. е. когда в меню **View** выбран пункт **Display**) и при имитации «машины Д3-28» (когда в меню **View** выбран пункт **Machine**), как в режиме «останова с индикацией», так и в «режиме работы по программе». Сразу после запуска или ресета имитатора (а также после нажатия кнопок С или Ш на пульте машины, т. е. <$С_{лат}$> или <Enter> в имитаторе) машина находится в состоянии «останова с индикацией».

Меню имитатора даёт доступ к следующим действиям с НМЛ:

File > **Open txt-file as Tape –** открытие txt-файла (если он не открыт), имитирующее установку кассеты в НМЛ. Признаком присутствия в НМЛ кассеты является наличие имени txt-файла в верхней строке окна имитатора; а также – закрытая крышка НМЛ на изображении машины (появляющемся при выборе **View** > **Machine**).

File > **Close txt-file as Tape** – так мы «вынимаем кассету из НМЛ».

ПК-клавиша со стрелкой ← действует (если машина находится в состоянии останова и выбран режим **View** > **Machine**) как кнопка «перемотка назад»; лента при этом перематывается в самое начало.

ПК-клавиша со стрелкой → аналогично имитирует «перемотку вперёд», но «перематывает» не до конца, а на одну программную единицу, заканчивающуюся командой END, т. е. кодом 0512. Блочные записи типа бейсик-программ тоже

перематываются целиком (при необходимости эту опцию можно отключить): в имитаторе перемотка вперёд стремится «пропускать одну запись». Если в txt-файле есть всего одна программа, то это то же самое, что и перемотка до конца.

Перемотка отображается в имитаторе миганием окна в крышке НМЛ (ну надо же было хоть как-то отобразить перемотку :-) . Если кассета в НМЛ отсутствует, то перемотка не включается, и нажатие ПК-клавиш со стрелками никак не отображается. Если кассета в НМЛ есть, но с неё уже всё считано или перемотано вперёд до конца, то попытка ещё раз перемотать вперёд отображается в имитаторе зажиганием индикатора ОМ – ошибка магнитофона (это верхняя лампочка рядом с Y-табло машины). Назад можно перематывать сколько угодно.

Если машина находится в программном режиме, то перемотки (и вообще большинство действий с пульта машины) в имитаторе не работают, блокируются; в реальности это не так, но в имитаторе сделано так, чтобы не усложнять имитатор прерыванием имитации программного режима машины без серьёзного повода.

<L> – на пульте машины (т. е. в режиме **View** > **Machine**) эта ПК-клавиша имитирует кнопку СЛ: если не идёт «работа по программе», то происходит считывание программы «с ленты» (из открытого в имитаторе txt-файла); оно также отображается миганием окна в крышке НМЛ. Если лента кончилась, то загорится индикатор ОМ; обычно в этом случае чтобы считать с ленты программу, надо нажать <$C_{лат}$> (индикатор ошибки погаснет), перемотать назад, и только затем подать команду <L>. Разумеется, если искомая программка не первая на ленте, то после перемотки назад надо нужное число раз сделать перемотку вперёд. <$C_{лат}$> устанавливает нулевой шаг загрузки в ОЗУ; для загрузки на ненулевой шаг следует выполнять специальные действия, указанные в описании программки.

<$K_{лат}$> – эта ПК-клавиша имитирует кнопку КП на пульте машины: команда вычисления контрольной суммы байтов в ОЗУ машины с нулевого адреса до байта с кодом команды END. Если команды END в ОЗУ машины нет, то загорится индикатор ОП – ошибка программы, и будет мигать X-табло. При успешном вычислении контрольная сумма индицируется на X-табло машины (о табло и о кнопках пульта машины рассказано в разделе 7).

После выполнения команды КП программный счетчик машины указывает не на начало ОЗУ, а на ячейку ОЗУ с кодом команды END. Поэтому перед запуском считанной программы или перед записью программы с нулевого шага ОЗУ на ленту (см. ниже) надо подать команду «Сброс» – нажать <C> на ПК-клаве; тогда программный счетчик машины будет указывать на начальный шаг в ОЗУ.

<Z> – эта ПК-клавиша имитирует кнопку ЗЛ на пульте машины: команда записи на ленту содержимого ОЗУ с нулевого шага и вплоть до кода END. Если такого кода в ОЗУ машины нет или не поставлена кассета в НМЛ, то загорится

индикатор ОМ (так сделано в имитаторе, а как было в реальности – уже не помню).

Надо быть внимательным, чтобы не испортить командой ЗЛ какую-нибудь уникальную «ленту». Желательно для записи заранее заготовить «чистую ленту»: создать «Блокнотом» пустой txt-файл с желаемым именем и поместить его в нашу папку [txt], чтобы перед записью можно было бы командой меню **File** > **Open txt-file as Tape** открыть именно его. (Для первых опытов в папке [txt] в архиве с имитатором уже есть пустой txt-файл, его имя: rabochaja_kasseta.txt.)

Разновидности команд записи и считывания файлов машиной описаны в «*Справочнике программиста*» для Д3-28. Здесь в подробности не вдаюсь, напомню только, что Д3-28 может работать с записями двух типов. Один из них (записывается и считывается командами SAVEX, LOADX, LOADP) – сплошной массив байтов, отделённых друг от друга контрольным битом чётности. Такой файл обязательно завершается байтом 0512 (это код машинной команды END) и его битом чётности 0. Имитатору биты чётности не нужны; запись на ленте имитируется txt-файлом, в котором байты программы напечатаны обычными десятичными цифрами и отделены друг от друга обязательным кодом перевода строки (в hex-нотации это байт 0A). Допускается присутствие в txt-файле также кода возврата каретки (hex-код 0D) и пробелов (hex-код 20).

Второй тип файла (записывается и считывается командами SAVER, LOADR) – последовательность 256-байтных блоков, без битов чётности и без обязательного кода END; на реальной МЛ блоки отделены друг от друга паузой в несколько миллисекунд. В частности, так записываются бейсик-программки. В имитаторе такая «запись на ленте» отличается от предыдущей только тем, что паузы в начале и в конце каждого 256-байтного блока имитируются цепочкой из 15 символов нижнего подчёркивания (hex-код 5F) с обязательным кодом перевода строки в конце цепочки. Именно по нижним подчёркиваниям в txt-файле имитатор различает файлы первого и второго типа.

Можно создавать txt-файлы, которыми будут имитироваться «магнитные ленты» с записями многих программ или их дублей. Для этого надо попеременно считывать программы из одних txt-файлов, записывать их друг за другом в другой txt-файл, не забывая при этом должным образом «менять кассеты» в НМЛ и делать нужное количество перемоток вперёд. (Правда, в этом мало смысла, так как всегда можно копировать txt-файлы и редактировать их содержимое более удобными программами в Windows :-)

После открытия файла командой **File** > **Open txt-file as Tape** имитатор перед его чтением проверяет наличие расширения txt в имени файла и смотрит начало файла, чтобы убедиться, что в нём записаны числа, представляющие байты программы. Если в файле что-то не то, то при чтении загорится индикатор ОМ,

прога не станет читать такой файл. При записи же прога проверяет только расширение, т. к. файл для записи может быть пустым. (И поэтому легко обмануть прогу, нарочно заменив расширение не текстового файла на расширение txt, так что имитатор произведёт-таки в него запись. Таким путём можно командой ЗЛ нарочно испортить любой файл; но я снимаю с себя ответственность за подобную порчу файлов, поскольку только что пояснил, что так не надо поступать).

В программах высокого уровня предусмотрены специальные команды работы с НМЛ; ими и надо пользоваться, когда машина находится в режиме «работа по программе». Например, в Бейсике-157107 работают команды REWIND – перемотка назад, SKIP N F – пропустить N файлов, SAVE – запись на ленту, LOAD – загрузка с ленты. В Фортране и в Выстре также есть команды работы с НМЛ. О формировании «библиотек записей» рассказывается в разделе 5 (и в документации к языкам высокого уровня, таким как Бейсик, Фортран и др.).

Процессы записи и чтения программ на МЛ составляли важный и волнующий этап работы с реальной машиной Д3-28. Постепенно на лентах образовывались смеси записей разного формата – программки для Бейсика (256-байтные блоки без контроля чётности), для Фортрана (многофайловые записи с кодами END, содержащие тройные копии заголовков и самих программок), для Выстры (её записи – с двойными заголовками, тоже с кодами END). Этот ностальгический аспект частично воспроизводится в имитаторе, хотя, конечно, загрузка программ в имитатор не полностью подобна реальному считыванию файлов с магнитной ленты. Отличие (наряду с тем, что имитатор читает и записывает txt-файлы намного быстрее, чем реальная Д3-28 работала с МЛ) происходит от того, что кассету в машину Д3-28 можно ставить и снимать с уже частично перемотанной лентой – в положении, сохранившемся после предыдущих действий с ней, – а txt-файл не обладает такой «памятью»: каждый раз командой **File** > **Open txt-file as Tape** txt-файл открывается с самого начала.

Чтобы имитатор мог перематывать последовательность 256-байтных блоков как одно целое, введена опция «пропускать LOAD-R блоки». При этом Бейсик, Фортран и Выстра могут успешно считывать свои программки из смеси, пропуская чужие. (Этой опции нет в ранних версиях имитатора. Если она чему-то помешает или приведёт к ошибкам, то её можно отключить через диалог, вызываемый клавишей <L> после нажатия <F1> в режиме **View** > **Display**).

Для иллюстрации чтения записей разного типа из смеси «на одной МЛ» в папке [txt] есть файл mix_25-09-2019.txt, содержащий 17 программ. Их краткое описание дано в конце того же txt-файла и приведено ниже. (Перечень вообще всех программ в txt-папке, прилагаемой к данной версии, с их кратким описанием см. в разделе 10.)

1. Первая программка в файле mix_25-09-2019.txt работает без дисплея и загружается без языков высокого уровня, просто командами <C>, <L>; <K> (проверка контрольной суммы КП=1258). Запуск: <C>, < T >, <S>. На X- и Y-табло машины «побегут» цифры, изображающие 2018 год, и будет мигать индикатор слева от X-табло – индикатор ожидания СИП (синхроимпульса периферии). Для останова программки можно нажать <C>. В дальнейшем эту программу можно использовать для контроля опции «замедление имитатора», см. раздел 8.

2. Вторая программа в файле mix_25-09-2019.txt работает тоже без языков высокого уровня, но уже с дисплеем. Это текстовая игра «23 спички». Она получена компиляцией программного текста, также имеющегося в файле mix_25-09-2019.txt, на языке Выстра. Загрузка и запуск: <L>, <K> (проверка контрольной суммы КП=22571), <C>, <S>. Переходим к **View** > **Display**, нажимаем <F5>. Далее действуем по подсказкам, появляющимся на дисплее, подтверждая ввод чисел нажатием <Enter>. Для останова переходим к **View** > **Machine**, нажимаем <C>; повторный старт: <S> и переход к **View** > **Display**, чтобы видеть текст на дисплее и реагировать на него.

3. Игра «23 спички» на языке Бейсик. Сначала надо загрузить сам Бейсик-157107 из файла Basic_D3-28_v3A__KP-157107__N-11343.txt (а перед этим, как и вообще перед загрузкой любой программы высокого уровня, следует «освежить» сам имитатор – перезапустить его клавишей <Escape>). О запуске Бейсика рассказано в разделах 1 и 5. Затем «ставим в НМЛ» mix_25-09-2019.txt и загружаем бейсик-программку командой LOAD или командой с именем программки: LOAD'23SP BAS-VERSION'. Запускается она командой RUN, как и любая бейсик-программа. Для останова бейсик-программы можно в ответ на запрос числа вводить нечисловой символ < / >; тогда Бейсик выдаст сообщение об «ошибке 13», и будет готов принимать новые команды.

Чтобы получить список всех имеющихся в файле mix_25-09-2019.txt бейсик-программ, надо загрузить командой LOAD'CTLG' и запустить бейсик-программку CTLG. Она выводит на дисплей записанные в ней имена бейсик-программ и их краткое описание. Программа CTLG находится почти в самом конце файла mix_25-09-2010.txt, поэтому она <u>загружается не мгновенно</u>, и пугаться подобной задержки не надо. Затем, исполнив REWIND, можно загрузить по имени и запустить любую программку из полученного списка. Эти бейсиковские программки в файле mix_25-09-2019.txt чередуются с программами на языках Фортран и Выстра.

Чтобы получить список фортран-программок надо загрузить (после ресета имитатора клавишей <Escape>) Фортран-5М из файла Fortran-5m_KP-138177_N-10414.txt (или Фортран-85 из файла Fortran-85_p1-p2.txt, см. раздел 5), запустить Фортран, нажав <T> и ещё раз <T> «на пульте машины», «подключить дисплей» (т. е. **View** > **Display** и нажать <F5>, причём для работы в Фортране надо сбросить в ноль разряды служебной строки дисплея), войти в операционную систему Фортрана: <S>, «поставить в НМЛ» mix_25-09-2019.txt, и подать команду просмотра «каталога магнитной ленты»: МЛ К. Прочитав программку CTLG и выполнив её листинг командами Фортрана (см. раздел 5), увидим строки с кратким описанием фортран-программок. Выполнив командой МЛ Н перемотку назад, можно загрузить <u>по имени и номеру версии</u> любую фортран-программку из полученного списка.

Чтобы получить список выстра-программ надо загрузить (после ресета имитатора) программу Выстра из файла D3-28_Vystra__KP-113478__N-9478.txt; запустить её, нажав <S> «на пульте машины», «подключить дисплей», «поставить в НМЛ» mix_25-09-2019.txt, и подать команду чтения с ленты программки CTLG (С, <Enter>, *имя_прогр*, <Enter>; см. раздел 5). Отыскивая запись CTLG, Выстра выведет на дисплей имена всех попадающихся ей выстра-программок. Загрузив таким образом выстра-текст CTLG, можно подать команду К (компиляция), и затем Е (экзекуция); на дисплее появится описание выстра-программ, содержащееся в CTLG. В Выстре нет команд перемотки ленты; вместо перемотки назад в имитаторе можно вынимать из НМЛ и снова ставить в НМЛ файл mix_25-09-2019.txt. И затем – загружать по имени желаемую выстра-программу из полученного списка, но перед загрузкой надо не забывать удалять из памяти Выстры предыдущую программу командой Ж <Enter>.

Таким образом, интересный момент здесь в том, что каждая система высокого уровня – Бейсик, Фортран, и Выстра, – читает из смеси только «свои» записи, а чужие пропускает. Вот дальнейшее перечисление программ в порядке их расположения в файле mix_25-09-2019.txt:

4. «23SP» – выстра-текст игры «23 спички», загружается в Выстре. Можно посмотреть её листинг (см. раздел 5) и пытаться её усовершенствовать редактированием; и записать в пустой txt-файл как новый выстра-текст; либо компилировать и исполнить. Либо – компилировать и затем, уничтожив исходный текст командой Ж и саму Выстру командой Г, записать в пустой txt-файл как самостоятельную программу (подобную второй программке в файле mix_25-09-2019.txt); для такой записи надо перейти к **View** > **Machine**, «зарядить в НМЛ» новый txt-файл, нажать <C> и <Z>. (Всё это просто для упражнения в действиях с имитатором).

5. «HALL» – бейсик-программа для учебной лабораторной работы по исследованию эффекта Холла. Загрузка и запуск – в Бейсике-157107, причём перед загрузкой Бейсика надо установить в имитаторе «бит b4 =1» (см. раздел 5). Эта программа управляла измерительной установкой, которая подключалась к Д3-28 через УСО (устройство связи с объектами) АЦСКС-1024-001. Программа определяет, подключена ли установка, и если нет, то предлагает измерения имитировать (для нашего имитатора подходит именно этот режим). Чтобы успешно выполнить работу и получить от ЭВМ положительную оценку, надо правильно задать диапазон температур

(пример: начальная T = 77 K (температура жидкого азота), конечная T = 350 K, приращение = 30 K), и после имитации измерений надо суметь ответить на вопросы учебного теста :-). Интересная программка.

6. Фортранная «исходная программа», загружается в Фортране-5М (или в Фортране-85) как ИП с именем SINU версия 1. После трансляции (командой T K1) она выводит на дисплей график синуса. Вывод этот довольно быстрый, так как он реализован вставкой в машинных кодах. Программка предназначена для иллюстрации преимущества Фортрана перед Бейсиком в быстродействии.

7. «DATA-SAVE TEST» – бейсик-программа, в которой разрабатывалась запись и чтение именованных массивов данных на МЛ (этот способ затем применялся в других программах для записи на МЛ результатов лабораторных измерений и обработки данных). Иллюстрирует работу с именованными массивами – символы имени сохраняются в разрядах десятичной переменной с помощью вставки в машинных кодах. Инструкция по работе с этой программой есть в самой программе. Для записи / чтения тест-массивов надо «поставить в НМЛ» вместо mix_25-09-2019.txt новый txt-файл или файл test_data-save_bas__blocks.txt, имеющийся в папке [txt].

8. «JUL1» – выстра-текст первой версии программы построения «фрактала JULIA». Загружается в Выстре (напоминание: если это не первая загрузка в Выстре, то перед новой загрузкой надо удалить из ОЗУ предыдущую выстра-программку командой Ж <Enter>, а иначе новая загрузка подстыкуется к концу предыдущего выстра-текста. Затем подаём выстра-команды: C <Enter> JUL1 <Enter>). После компиляции и запуска командами Выстры K <Enter> E <Enter> программа запрашивает параметры для расчёта «фрактала». Для примера можно ввести: начало Re = –2, Im = 1 (это координаты начальной точки на комплексной плоскости; в более продвинутой программке FRMJ они обозначаются как X и Y), D = 0.1, P = 38, S = 20, N = 24, R = 3, ReC = –0.75, ImC = 0.4. Эта программа примитивная: в ней нет возможности изменения «раскраски фрактала» (т. е. нет появившихся в FRMJ опций «Вид 4» и «Вид 8» с параметром M), и нет вывода результата на ТПУ.

9. «SIMPLE SINUS» – бейсик-программа, выводит на дисплей график синуса оператором PRINTTAB.

10. «KLAV-DEC-COD» – выстра-текст простенькой программки, показывающей клавишные коды; её описание см. в разделе 12 «Приложение 2: команды Выстры».

11. «CTLG» – выстра-текст программки с перечнем выстровских программок в файле mix_25-09-2019.txt

12, 13, 14, 15 – версии 5, 2, 3 и 4 фортранной ИП с именем «TSIF», означающим «тест оператора IF». Это очень короткие программки, запускать их не надо, достаточно посмотреть листинги. Несмотря на то, что они очень похожи, версия 5 в старых версиях имитатора вызывала ошибку, а версии 2, 3, 4 – нет. С 2.07.2019 соответствующий баг имитатора устранён (ура!), листинг всех тест-программок TSIF проходит без ошибки. Они сохранены здесь на всякий случай, для проверки отсутствия такого бага в дальнейших версиях имитатора.

16 и 17 – программы «CTLG» в формате Бэйсика (и затем бэйсик-запись END) и Фортрана (с завершающей записью КБ).

Клавишей <C> после нажатия <F1> в режиме **View** > **Display** отключается или включается недокументированная опция «отсветка» – индикация регистров X и Y при выполнении машинной команды OUTOWC. В некоторых программах отсветка используется для индикации загрузки 256-байтных блоков. Поскольку отсветка может в других случаях тормозить выполнение самой OUTOWC, в имитаторе предусмотрено отключение отсветки. Пример: вывод бегущей строки в первой программке в файле mix_25-09-2019.txt сделан указанной отсветкой; если её отключить перед запуском программки, то индикаторы X и Y останутся погашенными.

4. Работа с ТПУ

<Home> – первым нажатием этой ПК-клавиши включается возможность имитации ТПУ – термопечатающего устройства; это «включение ТПУ». Прога создаёт txt-файл с уникальным именем типа «цифры__цифры», где первая группа цифр это *час-минута-сек*. создания файла, а вторая группа цифр представляет дату создания файла: *день-месяц-год*. Сразу после создания файла в нём есть только одна строка: с текстом tpu_on. Такой файл – как бы рулон бумаги, заряженной в ТПУ; в него пойдёт вывод букв, цифр и прочих символов, предусмотренный в программах.

Windows сама выбирает место для этого файла: если в имитаторе уже открывались txt-файлы из какой-то папки, то файл помещается в эту папку; если же мы включили ТПУ сразу после запуска имитатора, то файл создаётся в той папке, где находится exe-файл с имитатором.

Дальнейшие нажатия <Home> имитируют прокрутку бумаги в ТПУ: каждое нажатие добавляет в txt-файл пустую строку. Таким путём можно отделять друг от друга блоки текста, выведенные в один и тот же txt-файл по ходу работы с ТПУ.

<End> – первым нажатием этой клавиши имитируется, если ТПУ уже было включено, отрывание бумаги: созданный при включении ТПУ txt-файл закрывается, и с этого момента его можно куда угодно переместить и / или переименовать. При этом создаётся новый txt-файл, пустой.

Если сразу же нажать <End> второй раз, то ТПУ выключается; в пустой файл печатается текст tpu_off, файл этот закрывается, а новый файл не создаётся. Такой файл-обрывок в дальнейшем надо не забыть удалить, он никому не нужен.

Если же после первого нажатия <End> выполнить какие-либо другие действия, то пустой txt-файл играет роль бумаги, оставшейся в ТПУ; в него пойдёт вывод, а после нажатия <End> он тоже будет «оторван» (и при немедленном повторном нажатии <End> «ТПУ выключится»; чтобы его снова включить, надо будет опять нажать <Home>. И т. д.).

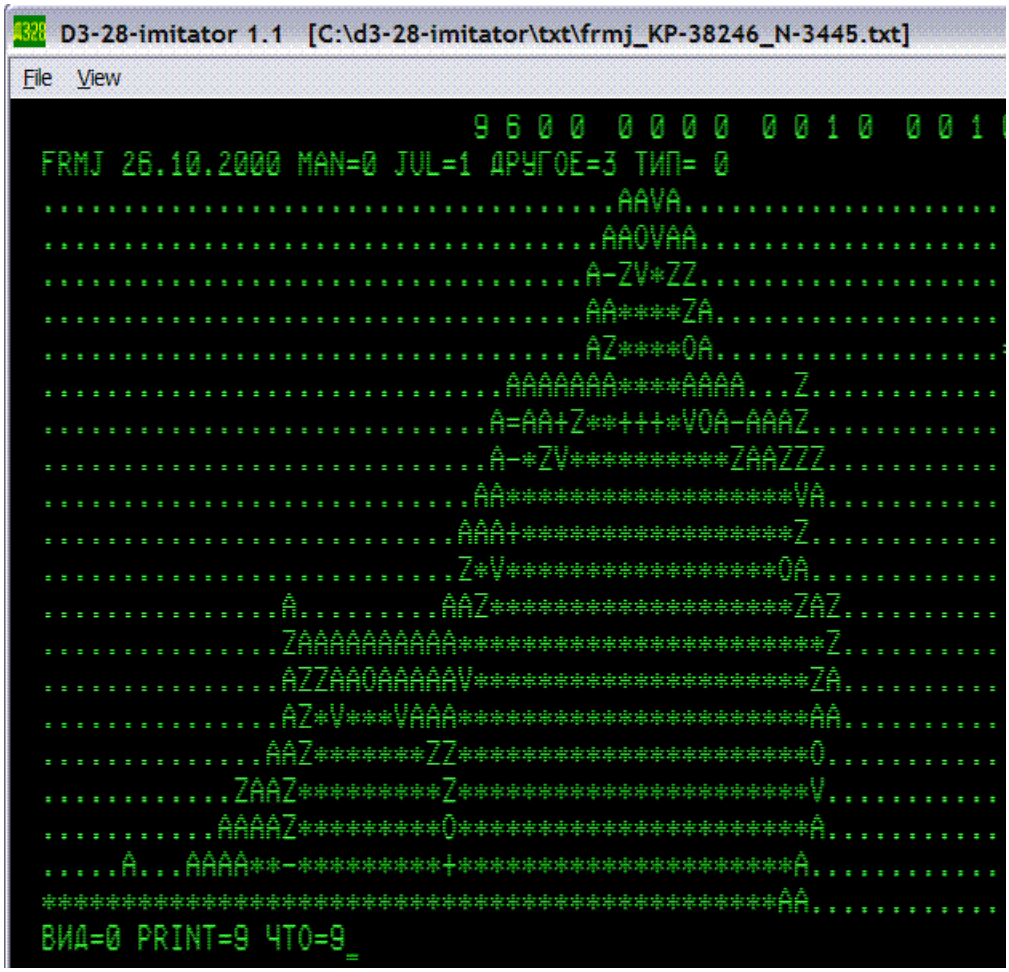
В самодельных программах вывод на печать надо программировать детально, по шагам (пример: см. raspechatchik_D3-28__KP-935__N-67.txt). В готовых же программах, таких как Бейсик, Фортран и Выстра, операции вывода на печать уже встроены; они описаны в документации к этим программам.

Например, в Бейсике-157107 листинг программ выводится на печать командой LIST#1. Имитатор выполняет такую команду в один присест. Целиком листинг делается и в Выстре. В Фортране же листинг печатается частями; после каждой части надо нажимать клавишу <Пробел>.

Реальное ТПУ умеет печатать только большие буквы. Имитатор в исходном режиме – тоже; при этом коды больших латинских и русских букв, а также цифр и прочих символов для вывода в txt-файл берутся из кодировочной таблицы “Windows Cyrillic (1251)”. Однако при работе имитатора с некоторыми версиями ОС ВТ-МХТИ, рассчитанными на применение более совершенных принтеров, нежели ТПУ, встречается необходимость распечатывать важные пояснительные тексты с маленькими буквами. Для этого достаточно добавить маленькие буквы из таблицы КОИ-8 – такой тип печати включается и выключается в имитаторе клавишей <P> после нажатия <F1> (в режиме **View** > **Display)** и выбора затем

опции «mhti-128c mode». Для распечатки экрана программкой print_screen__KP-2033_N-127.txt следует выбирать опцию «print-screen mode»; пояснения к этой программке приведены в конце указанного txt-файла.

Имитатор не нарушает форматирования текста, если оно предусмотрено в программке для Д3-28. Однако есть нюанс, связанный с выбором шрифта в той программе, которая будет применяться для просмотра txt-файла, имитирующего распечатку на ТПУ. Проиллюстрирую это примером. Вот часть скриншота экрана дисплея с расчётом «фрактала» программой FRMJ в нашем имитаторе Д3-28:



Здесь видно, что в ответ на вопрос «ЧТО=» (имеется ввиду «что делать дальше?») был выбран вариант «PRINT».

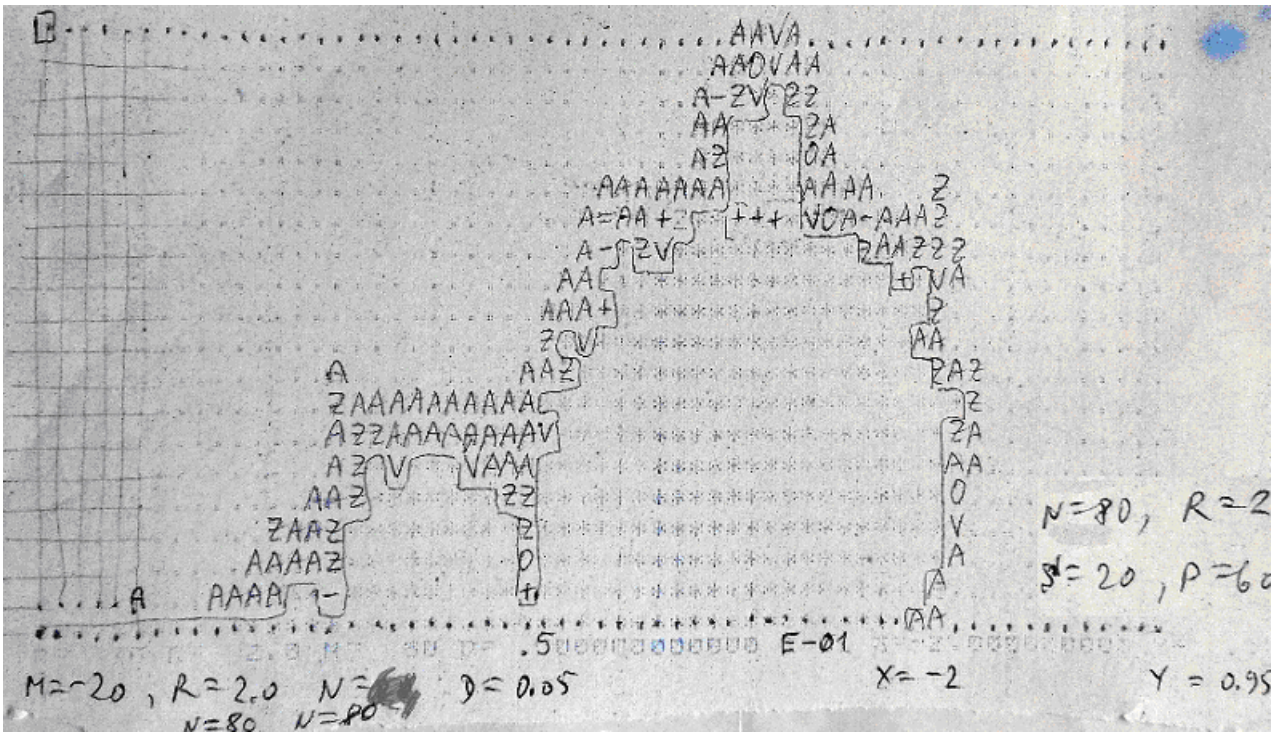
Имитатор с включенным ТПУ создал txt-файл с именем 17-41-57__23-6-2019.txt. Ниже изображено содержимое этого txt-файла в том виде, как его показывает программа Lister в составе файлового менеджера типа Total Commander (Lister запускается нажатием <F3> или «Просмотр», в его меню «Вид» следует выбрать «кодировку Windows»; в настройках был выбран шрифт Fixedsys):

Lister - [c:\d3-28-imitator\txt\17-41-57__23-6-2019.txt]

Файл Правка Вид Справка

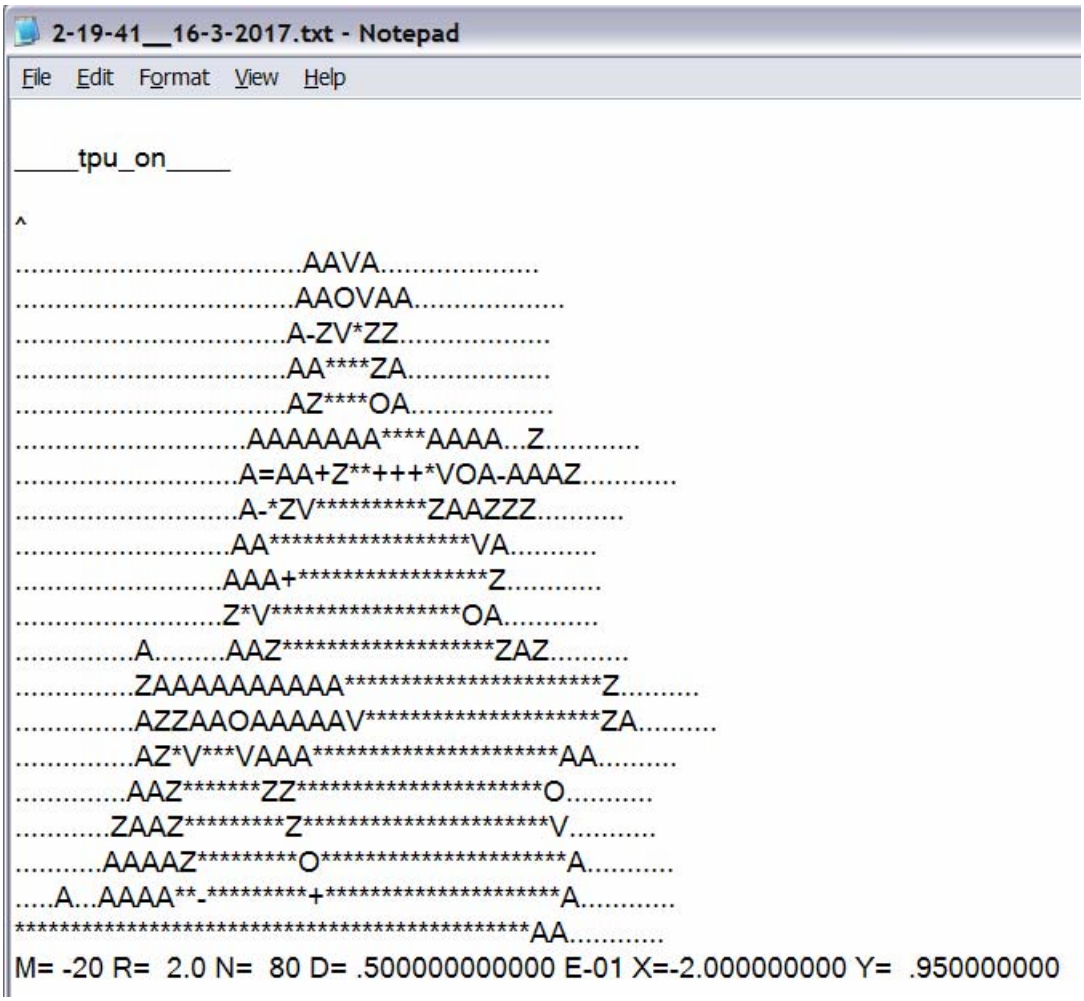
```
____tpu_on____
^
.....................................AAVA....................
....................................AAOVAA...................
...................................A-ZV×ZZ...................
...................................AA××××ZA..................
...................................AZ××××OA..................
..............................AAAAAAA××××AAAA...Z............
.............................A=AA+Z××+++×VOA-AAAZ............
.............................A-×ZV××××××××××ZAAZZZ...........
...........................AA×××××××××××××××××VA...........
..........................AAA+×××××××××××××××××Z............
..........................Z×V×××××××××××××××××OA............
...............A.........AAZ×××××××××××××××××××ZAZ..........
...............ZAAAAAAAAAA×××××××××××××××××××××Z..........
...............AZZAAOAAAAAV×××××××××××××××××××××ZA.........
...............AZ×V×××VAAA×××××××××××××××××××××AA.........
..............AAZ×××××××ZZ××××××××××××××××××××O...........
............ZAAZ×××××××××Z××××××××××××××××××××V...........
...........AAAAZ×××××××××O××××××××××××××××××××A...........
.....A...AAAA××-×××××××××+××××××××××××××××××××A............
×××××××××××××××××××××××××××××××××××××××××××××AA.........
M= -20 R=  2.0 N=  80 D= .500000000000 E-01 X=-2.000000000 Y=  .950000000
```

Сравним эту картину со старинной распечаткой на реальном ТПУ:



Старинная распечатка плохо сохранилась, обесцветилась, так что пришлось на ней бледные буквы подправить ручкой, но главное видно: имитатор распечатал в txt-файле всё правильно, а Lister всё правильно показал.

Но вот как выглядит содержимое того же txt-файла, если его открыть в «Блокноте» с другим шрифтом, в нашем примере – со шрифтом Microsoft Sans Serif:



```
2-19-41__16-3-2017.txt - Notepad
File  Edit  Format  View  Help

____tpu_on____

^
.................................AAVA...................
.................................AAOVAA..................
...............................A-ZV*ZZ..................
...............................AA****ZA.................
...............................AZ****OA.................
.........................AAAAAAA****AAAA...Z............
.........................A=AA+Z**+++*VOA-AAAZ............
.........................A-*ZV**********ZAAZZZ...........
.........................AA*****************VA...........
.........................AAA+****************Z............
.........................Z*V****************OA............
..............A.........AAZ******************ZAZ..........
..............ZAAAAAAAAAA**********************Z..........
..............AZZAAOAAAAAV********************ZA..........
..............AZ*V***VAAA*********************AA..........
..............AAZ*******ZZ*********************O...........
............ZAAZ*********Z*********************V...........
...........AAAAZ*********O*********************A...........
.....A...AAAA**-*********+********************A............
*********************************************AA............
M= -20 R=  2.0 N=  80 D= .500000000000 E-01 X=-2.000000000 Y=  .950000000
```

Видно, что картина искажена, поскольку ширина «знакоместа» у разных символов здесь разная, пробел вообще очень узкий.

Значит, если требуется воспроизвести формат, то следует для просмотра txt-файла, имитирующего ТПУ-распечатку, выбирать «моноширинный» шрифт – с постоянной шириной знакоместа (как это было в реальном ТПУ). Например, вполне подойдёт хорошо известный и достаточно распространенный шрифт **`Courier`**.

Работа с дизассемблером

В имитатор встроен дизассемблер байт-кодов из txt-файла или из имитируемого ОЗУ ЭВМ Д3-28 в стандартную для Д3-28 мнемонику команд, указанную в заводском "Справочнике программиста". Отступление от стандарта только в том что: а) между тетрадами байтов у нас не ставится пробел; б) перед номером десятичной ячейки ставится буква С; в) перед байт-кодами ставятся символы, если данные байты могут быть интерпретированы как символьные, – такой способ обнаружения текстовых фрагментов я заимствовал из онлайнового дизассемблера уважаемого Виталия Васильевича Колесника на его сайте d3-28.ru.

Вызов дизассемблера в имитаторе: **View > Display**, нажимаем клавишу <**F1**> и затем клавишу <A>. Кнопкой ДА (Yes) в окошке с вопросом "To print mnemocodes (disassembling) of the program ?" можно согласиться работать с дизассемблером, либо кнопкой НЕТ (No) – отказаться, если раздумали. Если не отказались, то на вопрос о типе дизассемблинга отвечаем "ДА", если хотим дизассемблировать байт-коды из текстового файла, либо – "НЕТ", если хотим дизассемблировать всю рабочую зону ОЗУ.

Дизассемблируются байт-коды с того шага (с того значения программного счётчика PC), который виден на X-табло (или Y-табло) в режиме "Ввод"; обычно это 00000, но можно задать любое нужное значение. Результат зависит от начального шага и может быть неправильным, если начальный шаг оказался вторым байтом 2-байтовой команды. Символьные коды, таблицы адресов, и прочие служебные данные тоже не распознаются – они дизассемблируются как команды.

Txt-файл для дизассемблирования не обязательно загружать в "ОЗУ Д3-28", достаточно открыть его командой **File > Open txt-file as Tape**. Если txt-файл не открыт или открыт, но не содержит в первых своих строках программу не блочного типа для Д3-28, то появится лишь диагностическое сообщение, без совершения дальнейших действий. Дизассемблируется только подходящий по содержанию txt-файл – программка (или её фрагмент) с кодом END (0512) в конце; по положению этого кода дизассемблер определяет, где ему остановиться. Если надо дизассемблировать лишь часть txt-программки, то следует в её копии в конце интересующего участка вставить байт-коды 0000 0512 и открыть в имитаторе такую копию. Если открыт файл со смесью программок, и первая запись в смеси является обычной программкой, то она будет воспринята дизассемблером как пригодная. Для работы с остальными обычными программками из смеси можно их переписать в отдельные txt-файлы (см. раздел 3 – о работе с НМЛ).

Блочные записи (т. е. последовательности 256-байтных блоков) содержат номера блоков, контрольные суммы и прочие служебные коды, не являющиеся машинными командами, поэтому такие записи не считаются пригодными для дизассемблинга. Однако блочная запись может быть загружена в ОЗУ командами загрузки в той программке, в которой она создавалась (т. е. командами Бейсика или ОС ВТ МХТИ и т. п.). Затем можно дизассемблировать всю рабочую зону ОЗУ и попытаться обнаружить там интересующие фрагменты.

Если перед вызовом дизассемблера мы включили имитацию ТПУ, то тем самым создали txt-файл для "ТПУ-распечаток", – в него будет выводиться листинг. При выключенном ТПУ дизассемблер для каждого своего вывода создаёт свой текстовый файл с временем-датой в качестве имени и с расширением .diz (в той папке, из которой открывались файлы для имитатора).

5. Запуск Бейсика, Фортрана, Выстры, и др. «Бит b4» КД

В этом разделе перечисляются нюансы, важные для правильного запуска некоторых высокоуровневых программ и для дальнейшей работы с ними.

Бейсик-157107

Сразу после открытия проги имитатора параметр «бит b4» сброшен в ноль; на что он влияет – рассмотрим ниже, пока не заботьтесь об этом. И установлена комфортная для запуска «Бейсика-157107» (т. е. Бейсика с контрольной суммой 157107) настройка дисплея:

служебная строка: 9 6 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 …
(т. е. система команд №2 и режим «ВК с ПС», так что каждая из команд ВК или ПС производит две операции: «возврат каретки» и «перевод строки»),

режим клавиатуры терминала: ВР РУС.

Если «бит b4» мы не меняли, то команда ОЧС (очистка экрана дисплея ПК-клавишей <F9>) или «ресет» имитатора (ПК-клавишей <Escape>) возвращает дисплей к этой же настройке. Действия для загрузки и запуска Бейсика-157107:

File > **Open txt-file as Tape** ,
в папке [txt] открываем Basic_D3-28_v3A__KP-157107__N-11343.txt.

Запуск производится с пульта машины; если мы смотрели экран дисплея, то необходимо перейти к изображению машины: **View** > **Machine**, чтобы ПК-клава стала управлять машиной, а не дисплеем. Затем на ПК-клаве нажимаем клавиши с латинскими буквами:

<C> – сброс счётчика команд (правда, поначалу он и так уже сброшен).
<L> – команда СЛ: загрузка программы в Д3-28 с магнитной ленты.
<K> – команда КП: проверка контрольной суммы. (Имитатор пока ещё не давал сбоев, а в реальности-то проверяли даже номер шага с кодом команды END (0512); в имитаторе для этого нажимаем <V> или <W> и смотрим на X-табло, там должно быть: 11343 05 12. Посмотрев, возвращаемся в режим «Работа» нажатием <R>).

<C> – сброс.
<S> – старт; в этот момент машина переходит в «программный режим» (а до этого машина была в состоянии «останова с индикацией»).

View > **Display** – переход от изображения машины к дисплею; при этом переходе в имитаторе автоматически меняется назначение ПК-клавиш.

<F5> – подключаем дисплей к «линии» связи с Д3-28, это вкл. ДУП-ЛИН, если до этого он был отключен от линии. Повторные нажатия <F5> отключают и включают ДУП-ЛИН; можно посмотреть текущее состояние, нажав <F1>; выход из просмотра – ещё раз <F1>.

Бейсик выводит на дисплей свой начальный диалог. Отвечая на СНИМИТЕ КАССЕТУ! и НОМЕРА ВНЕШНИХ ПОДПРОГРАММ?, а также вместо ввода нуля, нажимаем на ПК-клаве <Enter> – это команда ПС; конечно, можно и ноль вводить, когда надо. Включение ТПУ будет работать и в том случае, если в начальном диалоге мы не подтвердим наличие печатающих устройств.

Пример работы Бэйсика с внешними подпрограммами дан в программке call0_mpss-3-60-1988__bas.txt. Пояснения приведены в конце этого же txt-файла.

<Ctrl> – перевод клавы терминала в режим ЛАТ; делаем его после того, как Бейсик-157107 выведет слово ГОТОВ и двоеточие.

Слова языка «Бэйсик» (1970–80х годов) достаточно известны из различной справочной документации. Конкретно про Бейсик-157107 для Д3-28 см.:

Бэйсик-интерпретатор (Вариант 3А) Руководство программиста 1.419.001 Д58
«Бэйсик-интерпретатор (Вариант 3) Руководство программиста 1.419.001 Д23»
(есть в библиотеке по адресу http://retropc.org/Biblioteka_r_31.html#c254).

Описание вариантов «Бэйсика» с примерами программ научной направленности имеется в книге «*Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ» В.П. Дьяконов, М. "Наука", 1987*; существует также допечатка тиража этой книги, вышедшая в 1989 г. Большинство программ из этой книги без изменения должны работать и в имитаторе; см. примеры в diakonov-87_bas.txt, в конце этого же txt-файла дано описание приведённых там программок.

Здесь не стану вдаваться в подробности; напомню только некоторые из самых важных команд Бейсика:

LOAD – команда загрузки бейсик-программы пользователя с ленты. Важно не забыть перед этой командой поставить в НМЛ нужную кассету:

File > **Close txt-file as Tape** – «вынимаем кассету» с Бейсиком,
File > **Open txt-file as Tape** – «ставим кассету» с бейсик-программами.

REWIND – перемотка ленты назад.

LIST (или LIST#0, если до этого был LIST#1) – листинг с выводом на дисплей.

LIST#1 – распечатка листинга. До этого надо включить ТПУ, т. е. нажать <Home>.

SAVE – запись бейсик-программы на ленту. (При этом важно не путать кассеты!)

RUN – запуск бейсик-программы пользователя.

CLEARP 1,7999 – удаление бейсик-программы пользователя из ОЗУ (обычно перед набором или перед загрузкой с ленты новой бейсик-программы).

Каждая команда в Бейсике выполняется после ПС, т. е. <Enter> на ПК-клаве.

В имитаторе работают все разновидности команд записи и чтения (подробно описанные в документации к Бейсику), – такие как запись программы с именем, чтение программы по имени с ленты со многими программами (учитываются первые 6 символов имени), запись и чтение массивов, пропуск файлов на ленте.

Можно сформировать «библиотеку записей» – txt-файл, имитирующий ленту со многими файлами бейсик-программок. Это делается аналогично имитации ленты со многими записями программ в машинных кодах: попеременно набираем программы или считываем их командой LOAD из txt-файлов, и записываем их друг за другом командой SAVE'*имя_программы*' в файл-«библиотеку», не забывая при этом должным образом «менять кассеты» и выполнять REWIND перед оператором SKIP N F с нужным количеством N. Если не предполагается в дальнейшем изменять содержимое библиотеки, то можно в роли последней программки записать список имён имеющихся в ней программ, условившись называть такую программку-список каким-то «стандартным» именем, например 'CTLG' или 'КАТАЛОГ'. Тем самым появляется возможность сначала командой LOAD'CTLG' загрузить и узнать список имён программ на данной «ленте», а затем выполнить REWIND и загрузить нужную программу по её имени или по очерёдности её расположения.

В самом конце «ленты» следует делать завершающую запись: SAVE END. Тогда при неосторожных попытках повторного чтения последней программы без перемотки назад Бейсик не будет «зависать», а просто выведет на экран двоеточие – признак готовности к работе.

Если пытаться читать оператором LOAD «окончившуюся ленту», в которой нет завершающей записи, сделанной оператором SAVE END, то «Бейсик зависнет»; машина Д3-28 останется при этом в программном режиме. В этом случае надо перейти к изображению пульта машины и остановить машину нажатием <$С_{лат}$>; затем можно снова войти в Бейсик нажатием <$Т_{лат}$>, <$М_{лат}$>.

Запущенную командой RUN программу (или слишком длинный листинг) можно остановить клавишей АР1, т. е. нажатием <F3> на ПК-клаве. При этом Бейсик продолжает работать: машина находится в «программном режиме».

Чтобы остановить машину, мы должны перейти к управлению с её пульта, выбрав **View** > **Machine**, и «нажать на пульте кнопку С или Ш», т. е. нажать на ПК-клаве <$С_{лат}$> или <Enter>. Разница в том, что при нажатии «С» программный счетчик устанавливается на начальный шаг (а также происходит ресет указателя стека, сброс масок прерываний и т. п.), а при нажатии «Ш» счётчик после останова машины просто указывает на следующую команду из программы; её можно посмотреть в режиме «Ввод», нажав <V> или <W>.

Возобновить работу Бейсика после останова машины можно попытаться нажатием <S> (сначала вернувшись из режима «Ввод» в режим «Работа» клавишей <R>), если счетчик команд указывает на упомянутую невыполненную команду. Либо, если останов произведён клавишей <C>, – поочерёдным нажатием двух ПК-клавиш: <$Т_{лат}$> и <$М_{лат}$> в режиме «Работа». Это имитация нажатия кнопок со значком «треугольник» и «М» на пульте машины; тем самым подаётся команда «поиск метки» для «горячего старта Бейсика».

В папке [txt] примером «ленты с библиотекой бейсик-программ» наряду с mix_25-09-2019.txt и diakonov-87_bas.txt служит файл bega__5prg__bas.txt. Вид экрана в начале работы с ним:

```
                    9 6 0 0  0 0 0 0  0 0 1 0  0 0 1 0

БЭЙСИК Д3-28, ВАРИАНТ 3А
СНИМИТЕ КАССЕТУ!
В КОМПЛЕКС ВХОДИТ ПЕЧАТАЮЩЕЕ УСТРОЙСТВО ( 0 - НЕТ, 1 - ДА ) ?0
НОМЕРА ВНЕШНИХ ПОДПРОГРАММ?


ГОТОВ
:LOAD'КАТАЛОГ'
:RUN

BEGA-1
BEGA-2
SINUS
XAOC-1
katalog

ostanov u stroke  60
:REWIND:LOAD'XAOC-1'
:PRINTOPEN
P XAOC-1
```

Чтобы избежать проблем с РУС-буквами, желательно имена программ (да и все тексты в программах) набирать только ЛАТ-буквами. Так, в приведённом примере имена всех четырёх программ, составляющих «библиотеку», образованы латинскими буквами, только слово КАТАЛОГ при записи и чтении набиралось русскими буквами.

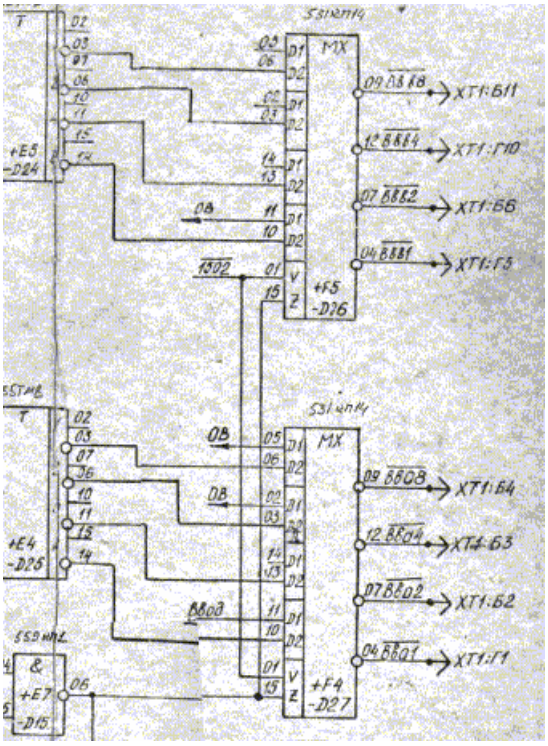
«Бит b4» контроллера дисплея

Прога имитатора позволяет экспериментировать с одним из параметров имитации контроллера дисплея – «битом b4», который, как оказалось, сильнейшим образом влияет на работу Бейсика-157107 (и Бейсика ТДМ; влияния же на работу других программ я не заметил). Поясню, о чём идёт речь:

Ввод данных в реальную машину из контроллера дисплея (КД) происходит байтами по 8-проводной шине «Ввод»; биты этой шины обозначим так:

b8, b4, b2, b1, a8, a4, a2, a1

Назначение байта на шине «Ввод» определяется байт-кодом, который машина выставляет на другой 8-проводной шине – шине «Управления» (УПР). Существенны три значения управляющего байт-кода: при УПР = 1507 контроллер дисплея выдаёт в шину «Ввод» байт 1515 – это признак того, что машина будет работать именно с КД, а не с каким-то другим периферийным устройством; при УПР = 1501 КД выдаёт в шину «Ввод» тот байт, который пришёл с клавиатуры терминала (при этом КД формирует для машины ещё и сигнал прерывания Пр8, но сейчас нам будет важно не это); при УПР = 1502 КД должен выдавать в шину «Ввод» байт, который несёт в машину информацию о состоянии дисплея и о типе дисплея – так называемое «слово состояния» КД.



И вот, именно со «словом состояния» КД имеется неясность. Можно заметить, что на схеме КД четыре из восьми входов мультиплексоров, которые формируют байт для шины «Ввод», висят в воздухе. Причём, как раз эти входы важны при появлении сигнала, соответствующего УПР =1502.

Схема эта вообще нарисована плохо; из справочников по мультиплексорам выяснилось, что соответствие между номерами входов и выходов должно быть вот таким (при УПР = 1502):

Выход: b8, b4, b2, b1, a8, a4, a2, a1
Вход: 11, 14, 05, 02, 11, 14, 05, 02

(красным цветом отметил висящие в воздухе «пустышки»).

«Методом тыка», с учётом также и указанных на схеме соединений, удалось найти один из работоспособных вариантов формирования слова-состояния КД в имитаторе (при УПР = 1502):

Выход на шину «ввод» : b8, b4, b2, b1, a8, a4, a2, a1
Сигнал на входе: 0, 0, 1, 1, Пр8, 1, 0, 0

В этом варианте, как видим, В-тетрада равна $0011_2 = 3_{10}$; в ней бит b4 = 0. Этот вариант по умолчанию устанавливается при запуске имитатора. Приведённое выше описание работы Бейсика-157107 относится именно к этому варианту

(и в этом варианте у Бейсика-157107 возникает проблема с выводом русских букв, о которой шла речь в разделе 1).

Позже выяснилось, что работа Бейсика-157107 существенно меняется, если выбрать для B-тетрады значение $0111_2 = 7_{10}$; т. е. установить бит b4 = 1. В этом случае резко усугубляется один из недостатков проги: уменьшается и без того невысокая скорость вывода на дисплей. Но появляются и преимущества – в этом варианте Бейсик-157107 правильно выводит на дисплей русские и латинские буквы, и позволяет менять размер шрифта при выводе на дисплей (точно так, как это указано в документации к Бейсику): оператор PRINT@1 выводит маленькие буквы, оператор PRINT@0 выводит большие буквы; пример:

```
:LIST

1 PRINT'HELLO WORLD, ПРИВЕТ РЕБЯТА!'
2 PRINT@1'HELLO WORLD, ПРИВЕТ РЕБЯТА!'
3 PRINT@0'СНОВА БОЛЬШИЕ БУКВЫ'
4 PRINT'БЕЗ СОБАКИ 1 ТАК И БУДУТ БОЛЬШИМИ'

:RUN
HELLO WORLD, ПРИВЕТ РЕБЯТА!
hello world, привет ребята!
СНОВА БОЛЬШИЕ БУКВЫ
БЕЗ СОБАКИ 1 ТАК И БУДУТ БОЛЬШИМИ
ОСТАНОВ В СТРОКЕ   4
:
```

Таким образом, приходится заключить, что <u>вариант с b4 = 1 – правильный</u>!

(Вероятно, входы мультиплексоров внутри самих этих микросхем «подтянуты» к $+E_п$, поэтому разработчики КД посчитали возможным оставить «в воздухе» те входы, на которые должен подаваться положительный потенциал логической единицы.)

Для переключения имитатора из режима «b4 = 0» в режим «b4 = 1» или обратно надо нажать ПК-клавиши <F1> и <4> в режиме **View** > **Display** с остановленной машиной. Ответ «Да» в окошке с названием-вопросом «Change b4?» меняет значение бита b4 с нуля на единицу и обратно. После того, как вариант будет выбран, надо ответить «Нет»; тогда окошко исчезнет.

В режиме с «b4 = 1» Бейсик в своем начальном диалоге, а также при исполнении операций LIST и PRINT, сам выполняет дополнительный перевод строки. Поэтому в настройке дисплея следует отключить «авто ПС=ВК» (об установке параметров служебной строки см. в разделе 2):

9 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 …

В режиме с «b4 = 1» очистка дисплея (ОЧС) ПК-клавишей <F9> или ресет имитатора ПК-клавишей <Escape> автоматически устанавливает служебную строку в указанное выше состояние – с отключенным «авто ПС=ВК» и с системой команд №1 (она не является обязательной, можно перейти и к системе №2).

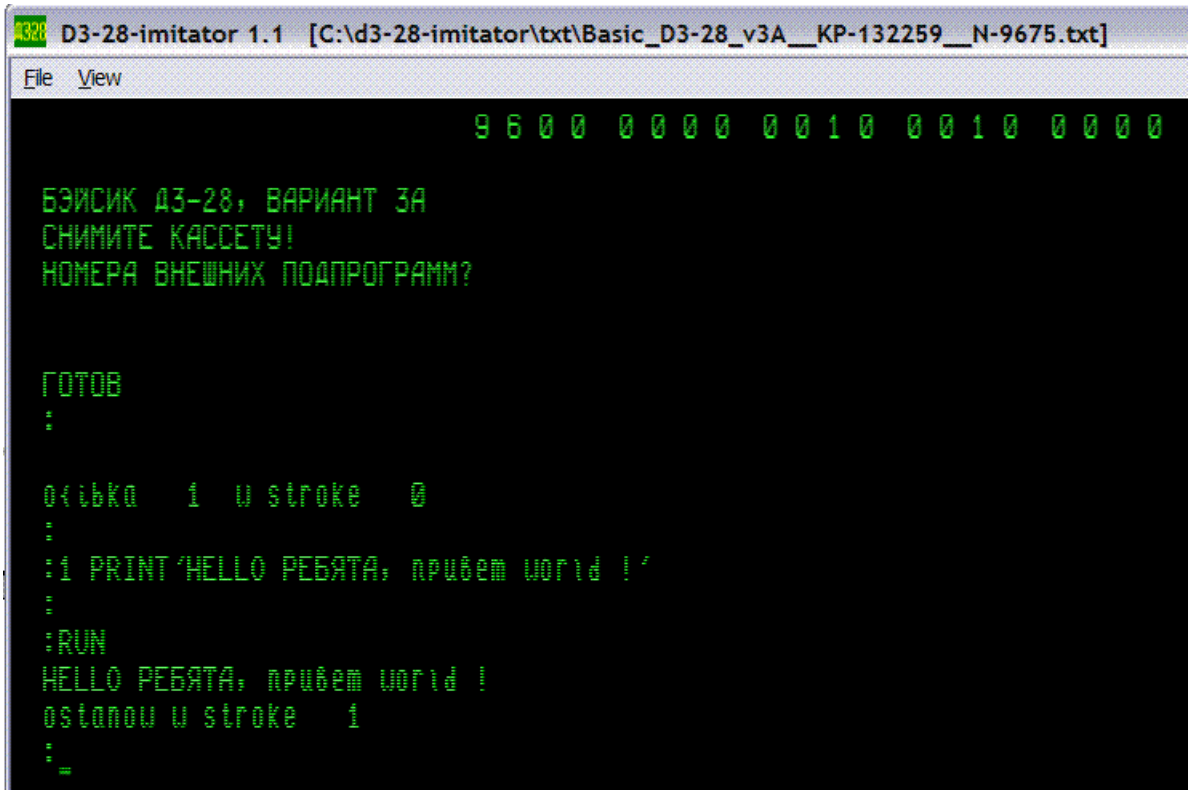
Наверное, надо бы вообще убрать из имитатора режим «b4 = 0», раз уж он «неправильный»; но я успел привыкнуть к нему, и, к тому же, на моём стареньком ПК он более быстрый, так что пока я его по-прежнему считаю основным :-)

Бейсик-132259

Файл Basic_D3-28_v3A__KP-132259__N-9675.txt содержит байт-коды из оцифровки имеющейся у меня кассеты с более ранней версией Бейсика – с контрольной суммой 132259. Оказывается, бит b4 на работу Бейсика-132259 не влияет, поэтому достаточно запускать его только в исходном режиме имитатора: с «b4 = 0». Бейсик-132259 запускается в имитаторе так же, как и Бейсик-157107 при b4 = 0, с той же самой настройкой дисплея «по умолчанию»:

служебная строка: 9 6 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 …
режим клавиатуры терминала: ВР РУС.

Особенностью этого Бэйсика является то, что нажатие <Ctrl> для перевода клавы терминала в состояние ЛАТ каждый раз Бейсик-132259 воспринимает как ошибку; но это не препятствует набору программок с ЛАТ-буквами:



В этом Бейсике не работает LIST#1. И команды @1 и @0 в операторе PRINT не работают. Однако они и не нужны, потому что Бейсик-132259 и без них умеет выводить оператором PRINT маленькие буквы, притом различая ЛАТ и РУС. Ещё особенность: в Бэйсике-132259, похоже, не работает экранное редактирование.

Интересно также, что программка, набранная и записная в Бейсике-132259, при запуске в Бейсике-157107 может правильно выводить на экран РУС-буквы даже при «b4 = 0».

Пример: программы BEGA-1, SINUS и ХАОС-1 в упоминавшемся выше файле bega__5prg__bas.txt, а также все бейсик-программки, кроме HALL, в mix_25-09-2019.txt, отредактированы и записаны Бейсиком-132259. Поэтому текст из них правильно выводится на дисплей в любом из двух Бейсиков и при любой настройке «бита b4» (однако листинг может выглядеть на экране странно). С этой же целью и некоторые другие программки, имеющиеся в папке [txt], отредактированы Бейсиком-132259. Программа же BEGA-2 редактировалась Бейсиком-157107, и каталог набирался в Бейсике-157107. Текст из них правильно выводится на дисплей именно в Бейсике-157107: либо при «b4=1», либо (если «b4=0») при нажатии <Ctrl> после набора RUN, но до нажатия ПС. Чтобы указанный каталог прочитать с ленты по его имени в Бейсике-132259, надо его имя набирать латинскими маленькими буквами. Вся эта ситуация связана с тем, что Бейсик-157107 записывает только коды букв и цифр, а Бейсик-132259 записывает ещё и коды команд 0015 и 0014, которые переключают РУС / ЛАТ.

Для Д3-28 были разработаны и другие версии «Бэйсика»:

Известный знаток и коллекционер вычислительной техники **Сергей Фролов** любезно предоставил wav-файл с оцифровкой Бейсика с контрольной суммой 157057: http://rt20.mybb2.ru/viewtopic.php?p=1902249#p1902249 Txt-файл с байт-кодами этого Бэйсика (и с комментарием о приоритете в конце файла) включен в папку [txt] под именем Basic_D3-28_v3A__KP-157057__N-11343.txt. Эта версия Бэйсика, по-видимому, очень похожа на Бейсик-157107 (см. краткое обсуждение на форуме РТ20 по приведённой выше ссылке).

k35b_Basic_plan_KP-192147_N-12621.txt – некий «Бейсик План»; оцифрован с «кассеты 35 Б» из архива программ https://d3-28.ru/kassety-s-programmami-d3-28/ и любезно предоставлен уважаемым **Виталием Колесником**. Записи бейсик-программок для «Бейсика План» не читаются обычными Бейсиками; в папке [txt] есть пример: test-basic-plan__2prg__plan-bas.txt; пояснения приведены в конце этого же txt-файла, составленного для простейшей проверки.

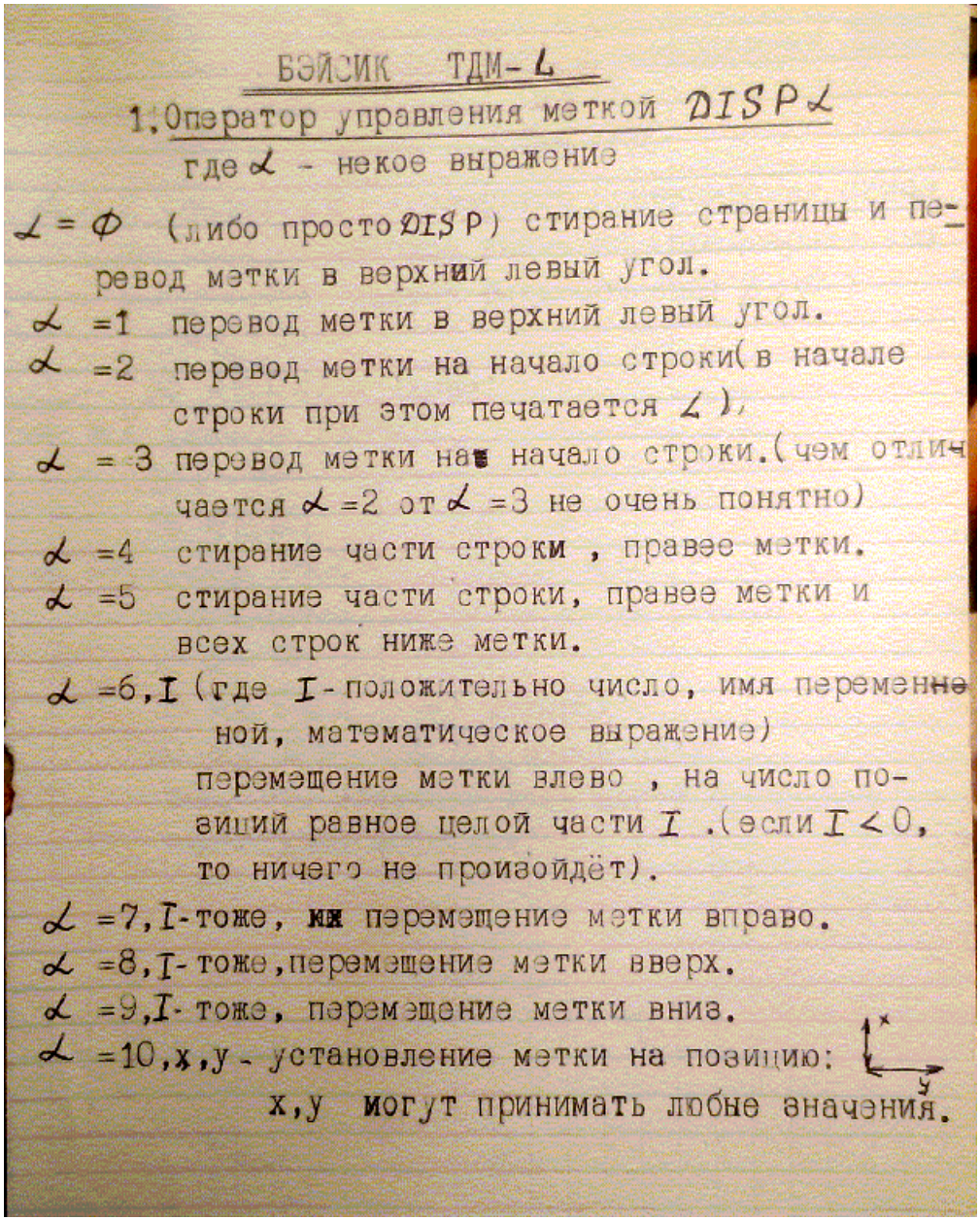
Бэйсик ТДМ

Одна из оцифровок с имеющихся у меня кассет, которую я поначалу принял за копию оцифровки Бейсика-157107, после раскодировки оказалась неким «Бэйсиком ТДМ Л»; в папке [txt] это файл Basic3A-tdm-l__KP-224538_N-15856.txt. Он загружается «одним куском», как и файлы упомянутых выше Бейсиков.

Существует ещё и «Бэйсик ТДМ М». Он записан 256-байтными блоками и загружается специальным загрузчиком с контрольной суммой 568. В папке [txt] загрузчик, блоки «Бэйсика ТДМ М» и комментарий о приоритете находятся в файле s3_side1__basic-tdm.txt. Чтение и запуск: <C>, <L>, <K> (проверка КП=568), <C>, <S>. Кассету с «Бэйсиком ТДМ М» любезно предоставил уважаемый участник форума «Полигон Призраков» **sanders**.

Документации к Бэйсику ТДМ не имею. Упоминание о нём в связи с операциями из алгебры матриц и ввода символьных переменных есть в книге В. П. Дьяконова «*Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ» 1987*, см. стр. 40, 41. О матричных операциях см. также в книге В. П. Дьяконова *«Применение персональных ЭВМ и программирование на языке Бейсик» 1989*, стр. 56 и далее; Бейсик-3А-TDM там причислен к «развитым версиям языка Бейсик». Плюс к этому – пара тетрадных листков (их фотографии любезно предоставил мой коллега Сергей Александрович Р.; они приведены ниже).

Опыты показывают, что начальный диалог выглядит так же, как в Бэйсике-157107, с тем только отличием, что Бэйсик ТДМ пишет слово ЖДУ вместо более привычного ГОТОВ (а также ОСТАНОВ СТРОКА вместо ОСТАНОВ В СТРОКЕ). Если в имитаторе установлен «бит b4 = 1», то регистры РУС / ЛАТ переключаются корректно – после вывода сообщения русскими буквами Бэйсик ТДМ сразу готов принять команду в регистре ЛАТ, включать для этого ЛАТ вручную не приходится. Вывод на экран маленьких букв с помощью спецификатора @1 в команде PRINT работает в Бэйсике ТДМ так же, как в Бэйсике-157107.



БЭЙСИК ТДМ-L

1. Оператор управления меткой DISP L

где L - некое выражение

L = Ф (либо просто DISP) стирание страницы и перевод метки в верхний левый угол.

L =1 перевод метки в верхний левый угол.

L =2 перевод метки на начало строки (в начале строки при этом печатается L),

L = 3 перевод метки на начало строки. (чем отличается L =2 от L =3 не очень понятно)

L =4 стирание части строки, правее метки.

L =5 стирание части строки, правее метки и всех строк ниже метки.

L =6,I (где I - положительно число, имя переменной, математическое выражение) перемещение метки влево, на число позиций равное целой части I. (если I < 0, то ничего не произойдёт).

L =7,I - тоже, но перемещение метки вправо.

L =8,I - тоже, перемещение метки вверх.

L =9,I - тоже, перемещение метки вниз.

L =10,x,y - установление метки на позицию: x,y могут принимать любые значения.

2. Оператор работы с символьными ~~RED~~ переменными RED

Оператор имеет вид REDL

L = L (аналог LET) оператор присваивания символьной переменной значения (предварительно переменной должна быть описана: L=Ф)

```
10 L=Ф
15 REDLL = 'кино'
```

L ±I (аналог INPUT)

```
10 L=Ф
15 REDI  '   'L
```

L =P(аналог PRINT). После REDP строка не переводится. Символ перевода строки /

```
10 L = Ф
15 REDLL    =' четырёх'
20 REDP   L 'ступенчатый'/
```

элементы массивов тоже могут принимать символьные значения.

Опытным путём (в имитаторе) выяснилось также, что команда STOP в Бэйсике ТДМ не работает; при чтении с ленты её код воспринимается как слово SKIP. Вместо команды CLEARP работает CLEAR. Бэйсик ТДМ позволяет вводить с клавиатуры и выводить на экран текстовые переменные; пример программки с текстовым вводом:

```
10 A = 1
20 REDI A
30 PRINT¤A
```

После запуска (командой RUN) эта программка ожидает в строке 20 ввода символов с клавиатуры. По ходу ввода символы отображаются в строке на экране. После окончания ввода, т. е. после нажатии ПС (<Enter> в имитаторе) строка символов выводится оператором PRINT¤ из переменной A на экран. В байт-кодах Бэйсика ТДМ видны отсутствующие в Бэйсике-157107 команды: DISP и MAT. Команда MAT нужна для вычислений с матрицами и работает так, как рассказано на стр. 59 в книге В. П. Дьяконова *«Применение персональных ЭВМ и программирование на языке Бейсик», 1989*. Опыт показал также, что в имитаторе при установленном «бит b4=1» и системе дисплейных команд №1 команда DISP 1 перемещает

курсор в начало первой строки экрана, а команда DISP 0 ещё и очищает экран, в согласии с информацией из «тетрадных листков». В байт-кодах Бэйсика ТДМ видна также машинная команда ONSEGM, переключающая сегменты 128к-памяти; исходя из этого можно предположить, что Бэйсик ТДМ способен неким образом обращаться к расширенной до 128к памяти, но как именно и зачем – не знаю. Если эти строки прочтут знающие люди, просьба к вам: поделитесь, пожалуйста, более полной информацией о Бэйсике ТДМ.

Редактирование бейсик-программок

В режиме с «бит b4 = 1», если при этом для дисплея выбрана система команд №1, в Бейсике-157107 и в Бейсике ТДМ доступно редактирование набираемой строки. Точнее говоря, до нажатия ПС, которым завершается набор строки, доступно исправление символов, набранных подряд, – при каждом нажатии ЗБ (в имитаторе это <Delete>) курсор в набираемой строке сдвигается на один шаг влево, а выведенный там символ стирается.

Таким способом можно удалить несколько символов, идущих подряд в конце набираемой строки, и заменить их новыми. Это очень полезное свойство, так как опечатки при наборе строки возникают часто, особенно – из-за необходимости часто менять регистр ВР / НР при наборе скобок, знаков *, +, =, и т. п. в громоздких математических выражениях.

Если перед запуском Бейсика был выбран режим с «бит b4 = 0», то указанное редактирование в системе команд №1 тоже работает, но выглядит оно иначе: при каждом нажатии <Delete> на экран выводятся буквы Н Н (с пробелом посередине) справа от последних выведенных символов в строке. Если теперь завершить набор строки, как полагается, нажатием <Enter>, и выполнить листинг, то эти буквы Н Н не появятся, а будет видно, что стёрлись набиравшиеся перед ними символы, притом в количестве, равном количеству нажатий <Delete>. Если вслед за нажатиями <Delete> выполнялся набор каких-то новых символов, то они появятся на месте стёртых символов, как и должно быть при исправлении ошибок.

В Бейсике-132259 указанная картина (появление Н Н при нажатии <Delete>, и при этом всё-таки выполнение редактирования) наблюдается вне зависимости от того, какой был выбран режим: «бит b4 = 0» или «бит b4 = 1».

Если для дисплея выбрана система команд №2, то вне зависимости от того, какой выбран режим КД, «бит b4 = 0» или «бит b4 = 1», ни в одном из Бейсиков указанное редактирование не работает (а курсор прыгает в начало страницы при нажатии <Delete>). Во всяком случае, так обстоят дела в имитаторе.

Для того чтобы отредактировать (изменить) уже готовую строку в бейсик-программке, т. е. строку, подтверждённую нажатием ПС в набираемой программке или в считанной с ленты, её обычно набирают заново, с желаемыми поправками. Так же поступают и для изменения номера строки. Для удаления ненужной строки достаточно набрать её номер и нажать <Enter> (т. е. ПС в случае работы на реальном терминале).

Однако такой способ – новый набор целой строки – неудобен в применении к длинным строкам. Длинные строки удобнее исправлять методом «экранного редактирования» (см. раздел 2); оно может оказаться полезным также при наборе фрагментов бейсик-программ со многими строками. Но, как показали опыты в имитаторе, в Бейсике-132259 экранное редактирование не работает.

Следует избегать завершения строки двоеточием. Опыт (в имитаторе) показывает, что эта ошибка может нарушить нормальную нумерацию строк и тогда она уже не поддастся исправлению.

Номера ошибок, сообщаемые Бейсиком-157107

0 – переполнение памяти, отведённой пользователю;
1 – недопустимый оператор;
2 – переполнение строки ввода (больше 120 символов);
3 – недопустимый ограничитель в строке;
4 – недопустимый номер строки;
5 – несоответствие кавычек в предложении;
6 – отсутствие открывающей скобки перед аргументом функции;
10 – неправильная запись индексов;
11 – неправильная запись индекса, массив не определён;
12 – несоответствие скобок в выражении;
13 – недопустимый элемент выражения;
14 – функция пользователя не определена;
15 – неправильное имя переменной;
20 – неправильная операцияотношения;
21 – недопустимый оператор IF;
22 – неправильный DIM, COM;
23 – недостаточно места для массива;
24 – неправильный DEF;
25 – нет данных для READ;
26 – недопустимый оператор DATA;
27 – неправильный формат команд в CMD;
30 – неправильный формат FOR – NEXT;
31 – нет NEXT;
32 – не было FOR;
33 – переполнение стека FOR – NEXT;
34 – нулевой шаг FOR;
35 – неправильный формат PRINT;
36 – неправильно задан формат печати;
37 – недопустимое выражение в TAB;
38 – отсутствие открывающей записи в буфере МЛ;
40 – НС2 < НС1;
41 – превышение уровня подпрограмм;
42 – RETURN без GOSUB;
43 – нет строки для перехода по GOSUB или GOTO;
44 – нет внешней подпрограммы с указанным номером;
50 – неправильное предложение с операторами обслуживания МЛ и перфоленты;
52 – сбой структуры файла;
53 – отсутствие в ЗУ массива при приеме с МЛ и перфоленты;
54 – не считан очередной блок данных с МЛ;

55 – считанный с МЛ блок не помещается в ОЗУ;
59 – при загрузке или записи программы с МЛ указан только один номер строки;
60 – нет ответа печатающего устройства;
61 – нет ответа считывателя;
62 – нет ответа перфоратора;
63 – нет ответа накопителя при обмене данными;
64 – нет ответа накопителя при приеме регистра состояния;
65 – ошибка по контрольному коду;
66 – ошибка по нечетности;
67 – ошибка при поиске;
68 – нет ответа на выбор накопителя;
69 – нет ответа на выбор команды;
70 – недопустимый номер накопителя;
71 – неправильный формат;
72 – недопустимый номер диска;
73 – недопустимый номер сектора или дорожки;
121 – недопустимые знаки при вводе по INPUT;
122 – недостаточно данных для INPUT;
123 – несуществующая переменная;
124 – слишком много данных для INPUT;
128 – некорректная операция (ОП) в процессе вычисления.

FORTRAN-85

Аналогично тому, как это делалось для загрузки Бейсика (см. выше), в свежезапущенный или освежённый ресетом имитатор «ставим кассету» Fortran-85_p1-p2.txt – в этом файле содержатся обе части Фортрана-85. Нажатием ПК-клавиши <L> в режиме просмотра пульта машины загружаем первую часть. Нажатием <K> проверяем контрольную сумму: 45669. Делаем сброс нажатием клавиши <C>.

Запуск производим нажатием ПК-клавиши <T> и ещё раз <T>. Зажигается индикатор «ожидание ответа ПУ».

Повторю, чтобы не возникало недоразумений: при запуске Фортрана не нажимайте <S> . Надо два раза нажать <T>. Это относится и к Фортрану-5М.

Затем: **View** > **Display**, так мы переходим от изображения машины к дисплею. Нужный регистр клавиатуры терминала Фортран в начальном диалоге будет задавать сам, а мы должны перед диалогом установить нули в служебной строке дисплея; для этого нажимаем <F8>, <F9>, и опять <F8>. Нажатием <F5> включаем ДУП-ЛИН. Фортран-85 задает вопрос:

ТРАНСЛЯТОР ЗАГРУЖАТЬ? (Д/Н)

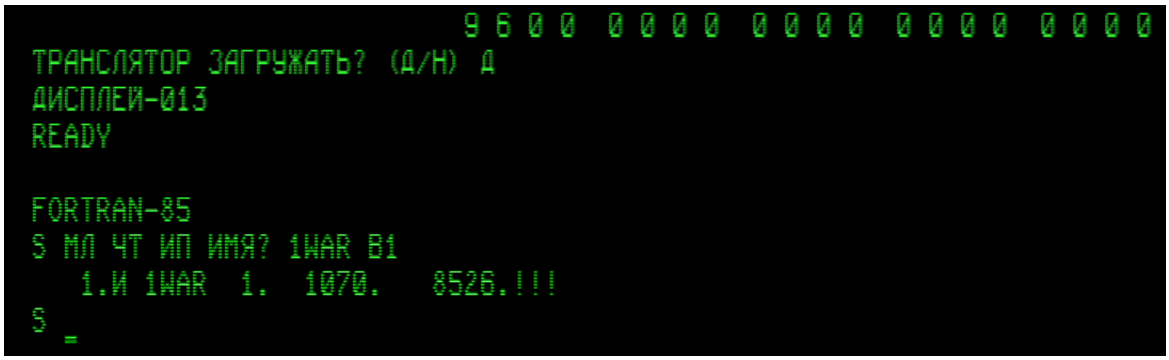
Нажимаем <D>, т. е. в ответ говорим «Да». Фортран сам загружает «с кассеты» вторую часть и спрашивает, какой у нас дисплей:

ДИСПЛЕЙ-01

Нажимаем <3>, так как у нас «013-й» дисплей. Фортран выводит слово READY и ждёт дальнейших директив. Нажимаем <Пробел>. На экран выводится строка FORTRAN-85, и программа ждёт новых указаний.

Если мы не собираемся набирать новую программу, а хотим вводить готовые программки «с магнитной ленты», то нажимаем <S> – это переход в «операционную систему» Фортрана. Затем нажимаем <M>, на экране эта команда сама допечатывается до МЛ, клава терминала остаётся в режиме РУС. Чтобы вызвать «Чтение с ленты», вводим Ч, т. е. нажимаем на ПК-клаве <Shift>, <~> и опять <Shift>; на экране появляется ЧТ. Нажимаем < I > – это означает: «исходная программа». Появляется запрос: ИП ИМЯ? Надо правильно ввести первые четыре буквы имени программы (а иначе Фортран «зависнет»). Важно перед такими делами не забыть сменить кассету.

Через меню **File** откроем, например, 1WAR__2WAR__frtrn5m.txt и введём имя: 1WAR. В ответ на запрос номера версии в виде буквы В нажимаем <1>. Вместо привычной <Enter> нажимаем <Пробел>. Это характерная черта Фортрана – в нём большинство команд подтверждается нажатием клавиши <Пробел>. Фортран читает три копии программы с ленты и обозначает успех тремя восклицательными знаками. Вот так выглядит фрагмент экрана на этой стадии:



Далее мы можем смотреть листинг программы (для этого надо нажать <L>), и пытаться редактировать программу (см. Приложение 1). Либо – транслировать её и запустить: в этом случае сначала нажимаем <T>, на экране появляется запрос «канала трансляции»: ТП К. Нажимаем <1> и <Пробел>. Появляется запрос Л?, т. е. с листингом или без? Если мы согласны увидеть не только адреса команд, но и текст команд, то нажимаем <L> (но выполняться листинг будет долго и с остановками; для продолжения надо нажимать <Пробел>). Быстрее будет без листинга, поэтому нажимаем не <L>, а <Пробел>.

После вывода BD нажимаем <Пробел>. Появляется буква S, означающая приглашение к вводу очередной директивы для «операционной системы». Нажимаем ПК-клавишу с косой чертой: < / >. На экран выводится слово READY. Нажимаем <R>, эта буква сама превращается в слово RUN, и сразу же происходит запуск транслированной программы 1WAR.

1WAR это игровая тест-программа, которая у нас тестирует взаимодействие машины с дисплеем и с клавиатурой; см. описание её бейсиковской версии в разделе 1. В фортранном исполнении эта игра работает динамичнее, заметно приятнее, чем на Бэйсике. Управление в ней (ПК-клавишами):

<1>, <2>, <3> – выбор горизонтальной составляющей скорости точки;
<7>, <8> – налево, направо; <9>, <6> – вверх, вниз;
<0> – выход из игры.

После выхода из программки Фортран-85 пишет: READY. Если мы нажмём <R>, то программка запустится снова. И т. д.

Если после выхода из программки, т. е. после того, как Фортран-85 напишет READY, нажать <Пробел>, то на дисплей выведется строка FORTRAN-85. Затем Нажатием <S> попадаем в «операционную систему», и тогда можно подавать новые указания. Например, можем распечатать на ТПУ листинг программки:

<Home> – этой клавишей «включается ТПУ»; затем <A> – на дисплей выводится слово АЦПУ; и <L> – запускается печатание; оно происходит частями, после появления каждого двоеточия следует нажимать <Пробел>.

Команды набора программ и редактирования приведены в pdf с конспектом описания Фортрана-85 для Д3-28, а также – в Приложении 1. В конспекте, однако, не очень ясно изложен способ работы с библиотеками фортранных записей. Рассмотрим этот существенный аспект подробнее.

Упомянутый файл 1WAR__2WAR__frtrn5m.txt служит примером имитации ленты, содержащей смесь записей. В начале этой «ленты» записан FORTRAN-5M (о нём речь пойдёт ниже), а затем – фортранная библиотека: исходные тексты и рабочие (транслированные в машинные коды) фортранные программки со своими заголовочными файлами. В конце есть завершающая запись «КБ».

Чтобы увидеть каталог фортранной библиотеки надо (находясь в «операционной системе» S) подать команду МЛ К, т. е. нажать <$М_{лат}$> и <$К_{лат}$>. Результат:

```
READY

FORTRAN-85
S МЛ К
  НЗ Т ИМЯ   В     Б      КС  123
   1.И 1WAR  1.  1070.   8526.!!!
   2.И 2WAR  1.  2664.  21488.!!!
   3.Р 1WAR  1.  4306.  47581.!!!
   4.Р 2WAR  1.  5626.  63418.!!!
   5.И SF-T  1.   199.   1650.!!!
   6.И SF-T  2.   255.   2218.!!!¤
S _
```

Обозначения столбцов в этой таблице: НЗ – номер записи, Т – тип (И – исходный текст программы на Фортране, Р – рабочая программа), ИМЯ – имя программы, В – номер версии программы, Б – размер программы в байтах, КС – контрольная сумма, 123 – номера копий на ленте для индикации качества чтения: удачно прочитанная копия отмечается восклицательным знаком, сбойная копия – знаком минус. Символ ¤ в конце каталога означает наличие завершающей записи «Конец Библиотеки». В каталог попадают только те программы, которые записаны командами Фортрана и поэтому имеют специальную заголовочную часть; сам Фортран и прочие программы, если они есть на ленте, при выводе каталога не обнаруживаются.

Можно продолжить формирование библиотеки, добавляя в неё новые фортранные программки или удаляя лишние. Для добавления новой исходной программки, имеющейся в ОЗУ, т. е. уже набранной или прочитанной из какого-либо txt-файла, следует «поставить кассету» с библиотекой, выполнить указанным выше образом просмотр каталога программ в ней, и подать команду записи: МЛ ЗП И (т. е. нажать <M>, затем <Z> и затем < I >). Добавить в библиотеку на МЛ рабочую программку удастся только <u>сразу после трансляции</u> и просмотра каталога МЛ; при этом вместо И надо подать команду Р. Фортран не спрашивает имя записываемой программы, так как он уже его знает; он запрашивает лишь номер версии, который мы желаем присвоить новой записи на МЛ. Вводим желаемый номер версии и нажимаем <Пробел>. Фортран дописывает программку в библиотеку после уже имеющихся в ней программ и присваивает ей следующий по порядку номер записи. Завершаем эти действия командой записи признака «Конец Библиотеки»: МЛ ЗП КБ (слово КБ набирается нажатием <$К_{лат}$>).

Удаление лишних записей лучше делать сразу после запуска Фортрана. Сначала смотрим каталог (МЛ К), затем набираем команду типа МЛ БНН2 3 5 0, указывая в ней номера записей, которые хотим сохранить. Команда набирается нажатием <$М_{лат}$> и <$В_{лат}$>. Здесь для примера сохраняем записи 2, 3 и 5 (этот учебный опыт проводим на резервной копии файла 1WAR_2WAR__frtrn5m.txt). Список сохраняемых номеров завершаем номером «ноль» – это признак конца списка. И нажимаем <Пробел>. Фортран переписывает указанные программы с начала ленты (при этом «небиблиотечная» программа окажется затёртой) и сам дописывает в конце признак КБ, показывая результат так:

```
S МЛ БНН2 3 5 0
  НЗ Т ИМЯ   В     Б      КС  123
   2.И 2WAR  1.  2664.  21488.
   3.Р 1WAR  1.  4306.  47581.
   5.И SF-T  1.   199.   1650.kb
```

Если теперь снова посмотреть каталог, то увидим, что Фортран перенумеровал записи – присвоил им естественные порядковые номера, начиная с единицы:

```
S МЛ К
  НЗ Т ИМЯ   В     Б      КС  123
   1.И 2WAR  1.  2664.  21488.!!!
   2.Р 1WAR  1.  4306.  47581.!!!
   3.И SF-T  1.   199.   1650.!!!x
```

Чтобы положить начало новой библиотеке, надо иметь в ОЗУ набранную или считанную из какого-либо файла фортранную программку. У неё обязательно будет какое-то исходное имя. «Ставим в НМЛ» созданный заранее пустой txt-файл с желаемым названием. В операционной системе Фортрана подаём команду МЛ ЗП (т. е. нажимаем <М>, <Z>) и нажимаем <В>. В ответ на запрос Б Н вводим число 0 – к нему Фортран прибавит единицу, и результат станет «библиотечным номером» первой записи. Завершаем командную строку нажатием клавиши <Пробел>. В новой строке снова подаём команду МЛ ЗП, указываем тип записи: < R > для рабочей программы или < I > для исходной. На запрос номера версии «В» вводим желаемое число, и нажимаем < Пробел >; происходит запись программы с исходным именем. И затем не забываем подать команду записи «Конец Библиотеки»: МЛ ЗП КБ.

Если Фортран вместо очередного чтения файла выводит значок ¤, то надо «перемотать ленту назад» командой МЛ Н (буква Н вводится нажатием <N>). Все упражнения (с чтением, записью, трансляцией, запуском программок) можно выполнять неоднократно, возвращаясь в операционную систему Фортрана, пока не произойдёт что-то нежелательное и непоправимое; в последнем случае придётся клавишей <Escape> запустить имитатор с самого начала.

Рабочую программу, созданную в Фортране-85, почему-то не удаётся запустить при считывании с ленты в Фортране: идёт череда сбоев. Однако такая программа может быть запущена без загрузки Фортрана, а прямо с пульта машины, как и всякая программа в машинных кодах. Надо только её «найти на ленте». Заголовочная часть у фортранной программки состоит из четырёх файлов, а основную часть программки Фортран записывает в трёх копиях, так что запись одной программки это 7 файлов.

Например, чтобы загрузить с «ленты» 1WAR__2WAR__frtrn5m.txt без загрузки Фортрана основную часть рабочей программы 1WAR (это 3-я программа в «каталоге», но в начале этой ленты есть ещё не замечаемый каталогом файл Фортрана-5М), надо перемотать вперёд 1+7+7+4=19 файлов. Т. е. открываем файл 1WAR__2WAR__frtrn5m.txt и в режиме **View** > **Machine** нажимаем 19 раз ПК-клавишу <→ >. Затем: <L> (команда СЛ – «считывание с ленты»), <K> (команда КП – проверка контрольной суммы, должно получиться число, которое раньше мы видели в «каталоге МЛ»: 47581), <C> (сброс счётчика команд в ноль).

После этого можно либо сразу запустить считанную указанным образом программу, либо сначала «поставить в НМЛ чистую кассету» и записать программу, нажав < Z > (это команда ЗЛ – запись на ленту) в отдельный txt-файл, чтобы в дальнейшем не извлекать её так из «библиотеки» каждый раз.

Считанная указанным образом (т. е. без Фортрана) рабочая программка запускается обычным образом: <C>, <S>. Переходим к **View** > **Display**, включаем ДУП-ЛИН нажатием <F5>, если «линия» была отключена. На экране появляется буква R. Дело в том, что программка 1WAR была записана не в Фортране-85, а в Фортране-5М, который обозначает своё состояние готовности не словом READY, а одной буквой R. Нажимаем <S> (потому что в Фортране-5М в состоянии R это команда «Счёт по программе»), и программа начинает работать. Если бы мы аналогичным образом имели дело с программой, транслированной в машинные коды и записанной как «рабочая программа» в Фортране-85, то при аналогичном её запуске с пульта машины на экране появилось бы слово READY; тогда для старта надо было бы нажать <R> – это команда RUN в Фортране-85.

Фортранная реализация игры 1WAR работает шустрее и гораздо чётче откликается на клавиши управления, чем бейсиковская версия. (На реальной Д3-28 бейсиковская версия 1WAR практически вообще не работает – возникает просто жуткий тормоз. Фортранная же версия работает очень хорошо.)

При исполнении рабочей фортранной программки может произойти нечто на первый взгляд странное: на экран выведутся некие числа с минусом, и программка остановится. Это номера шагов, где Фортран обнаруживает ошибки программы (такие как деление на ноль, засылка в 2-байтовую переменную числа с абсолютным значением превышающим 32767, и т. п.)

FORTRAN-5M

FORTRAN-5M – версия Фортрана в одном файле (в отличие от Фортрана-85, состоящего из двух частей). Похоже, это более новая версия, поскольку она менее «глючная»: в имитаторе она успешно запускает в том числе и свои рабочие программки, считанные с ленты. В папке [txt] Фортран-5М находится на «кассете» Fortran-5m_KP-138177_N-10414.txt, а также – в начале упоминавшейся выше «библиотеки записей» 1WAR__2WAR__frtrn5m.txt.

Работа с Фортраном-5М мало отличается от работы с Фортраном-85. В свежезапущенный или освежённый ресетом имитатор «ставим кассету» 1WAR__2WAR__frtrn5m.txt, нажимаем: <L>, <K> (контрольная сумма равна 138177, код END на шаге 10414), <C>, <T> и ещё раз <T> (латинские).

Затем: **View** > **Display**, <F8>, <F9> и опять <F8>. Нажатием <F5> включаем ДУП-ЛИН. На экран выводится буква R, означающая готовность Фортрана-5М к работе (Ready). Нажимаем <T> – это вход в «транслятор», на экране появляется строка FORTRAN-5M. Затем для входа в «операционную систему» Фортрана нажимаем <S>. Далее можно выполнять те же действия с файлами фортранных программок, которые рассматривались выше. Например, для просмотра каталога уже «стоящей в НМЛ кассеты» подаём команду МЛ К. Результат выглядит так:

```
                          9 6 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0

RT
FORTRAN-5M
S МЛ К
  nz t ιmq   U     b      kS  123
   1.ι 1WAR  1.  1070.   8526.!!!
   2.ι 2WAR  1.  2664.  21488.!!!
   3.r 1WAR  1.  4306.  47581.!!!
   4.r 2WAR  1.  5626.  63418.!!!
   5.ι SF-T  1.   199.   1650.!!!
   6.ι SF-T  2.   255.   2218.!!!x
S _
```

Здесь маленькие латинские буквы вместо больших русских – не глюк имитатора; такая же картина возникала и в реальности. Буквой *i* обозначены исходные программы (ИП), буквой *r* – рабочие программы (РП).

Две версии программки SF-T позволяют здесь заметить, что оба Фортрана слегка по-разному работают со шрифтами: версия 1 написана в Фортране-85, при трансляции и запуске в Фортране-5М она в финальных сообщениях выводит маленькие русские буквы вместо больших латинских; в SF-T версии 2 формат сообщений отредактирован в Фортране-5М (это отмечено комментарием, посмотрите листинг программы, нажав <L>), и она выводит текст финальных сообщений правильно. Опыт показывает, что выводить тексты маленькими буквами умеют оба Фортрана.

В Фортране-5М, в отличие от Фортрана-85, вполне удаётся запускать транслированные им самим и записанные на ленту рабочие программы (РП). Например, загрузим из файла 1WAR__2WAR__frtrn5m.txt рабочую программу 2WAR командами

S МЛ ЧТ РП ИМЯ? 2WAR B1 <Пробел>

Фортран-5М выведет на экран параметры загруженной программы, и напечатает букву S – признак ожидания директив для «операционной системы». Нажимаем <S>, появляется буква R – признак того, что мы вышли из «операционной системы» и вернулись к начальному меню Фортрана-5М, в состояние Ready. Ещё раз нажимаем <S> – теперь (т. е. после R) это означает команду «Счёт по программе». Программа 2WAR начинает работать.

Заодно опишу, что происходит в этой игре 2WAR.

Игра 2WAR начинается с запроса «СКОРОСТЬ B = ». В ответ надо ввести число не большее 5; оно может быть дробным; для начала можно попробовать ввести 1, а в следующих попытках уменьшить или увеличить его, в зависимости от быстродействия компьютера. Оно задаёт скорость американского «бомбардировщика» и тем самым задаёт темп игры на данном компьютере. Бомбардировщик изображается буквой «B», взлетающей из правого нижнего угла экрана – из «USA»; он летит бомбить атомной бомбой Советский Союз, изображённый в левом нижнем углу экрана как «SU» – Soviet Union (эта игра сочинялась мной в начале 1980-х годов; угроза ядерной войны в те времена представлялась вполне реальной). Сверху чертится горизонтальная линия с цифрами; это условная «шкала дальности».

Оказавшись над SU, вражеский «бомбардировщик» сбрасывает бомбу, и она падает вертикально вниз… Экран очищается и появляется финальный текст: «ВЫ УНИЧТОЖЕНЫ». Таков неудачный исход игры. Чтобы этого не случилось, надо в подходящий момент времени, пока «B» ещё в полёте к SU, нажать клавишу <Пробел>. Тогда на экране появляется запрос горизонтальной и вертикальной составляющих вектора скорости V ракеты, которая должна будет сбить вражеский бомбардировщик «B»:

–V =
!V =

Ввод отрицательного числа вызывает новый старт игры – так сделано для большей динамичности: чтобы в случае неудачного выбора момента времени для пуска ракеты не дожидаться заведомо неудовлетворительного финала.

Подбор положительных значений для этих составляющих скорости – один из немногих творческих моментов в игре, как и выбор момента для ввода вектора скорости и тем самым запуска нашей ракеты. «Шкала дальности» на экране дисплея облегчает эту задачу, позволяя нам примечать раз от раза положение «B» и положение ракеты-точки при их сближении.

Вернее говоря, удачное начало игровых действий должно быть вот каким: как только «B» взлетел из USA, надо пустить ракету с прицелом на территорию «USA», – ведь необходимо прежде всего уничтожить военные базы противника, а иначе оттуда будут вновь и вновь взлетать бомбардировщики, даже если мы наловчимся их сбивать без промаха. Затем, второй (или более) ракетой следует сбить «B»:

```
                        9 6 0 0  0 0 0 0  0 0 1 0  0 0 0 0  0 0 0 0  00 10 03

ВРЕМЯ ВОЙНЫ =     1.
 -V = 4

 !V = 3
 ...... ..1.........2.........3.........4.........5.........6.........7.......

                                                                  B
                              .   .   .   .
                         .                   .
                     .                           .
                 .                                   .

             .                                           .
         .                                                   .



       .

    .


.
SU                                                                      USA
```

```
 ...... . 1.........2.........3.........4.........5.........6.........7.......

                                  .   *
                         .   .   .   .   .
                     .   .                   .
                     .                           .
                 .                                   .
             .
             .                                           .
         .                                                   .

       .           ПОБЕДА !!!   П О Б Е Д А !!!   ПОБЕДА !!!    .
       .                                                           .

    .                                                                .



.                                                                    .
SU                                                                   ***
```

Для того чтобы остановить машину с работающей фортранной программкой и затем вернуться в работавший до этого Фортран, можно на пульте машины (т. е. в режиме **View > Machine**) нажать клавишу сброса <$С_{лат}$> и затем попробовать нажать <$Т_{лат}$> и <S> (если до этого работали с ИП), или <$Т_{лат}$> и <$Т_{лат}$> (если работала загруженная с ленты РП). Перейдя к **View > Display**, увидим в случае удачи, что Фортран снова работает – находится в состоянии Ready. Далее можно действовать так, как рассказывалось выше; ИП или РП, имевшаяся в памяти машины, обычно там остаётся невредимой.

Для того чтобы начать набирать новую ИП, надо из состояния Ready войти в транслятор Фортрана-5М нажатием <T> (или нажатием <Пробел> в Фортране-85), вслед за появлением названия Фортрана нажать <Пробел>, и затем нажать <M>. На экране появятся слова MAIN PROGRAM, это приглашение к вводу желаемого имени новой ИП. Ввод имени и строк ИП следует завершать клавишей ВК, т. е. <Backspace> в имитаторе. В каждой строке есть три фиксированных позиции – для номера операции, для метки, и для самой операции; в каждой строке допускается описывать только один оператор. Переход от 1-й позиции ко 2-й и к 3-й – клавишей <Пробел>. Фортран нумерует строки ИП автоматически. Дальнейшие подробности см. в разделе 11. Приложение 1: команды Фортрана-85 и -5М.

ОС ВТ-МХТИ 32к, версия ЛФТИ

Кассету с «ОС ВТ-МХТИ 32к» любезно предоставил участник форума «Полигон Призраков» **sanders**. Эта система записана 256-байтными блоками, она загружается специальным загрузчиком (с контрольной суммой КП = 1168). Байт-коды загрузчика и самой системы, полученные из оцифровки кассеты, находятся в папке [txt] в файле s4_side2_os-mhti_32k.txt.

Загрузка в имитатор: <L>, <K> (проверка КП загрузчика), <C>, <S>. Затем: **View** > **Display**, <F5> – с этого момента начинается загрузка блоков, она отображается на экране дисплея строкой символов +1 и +2.

Чтобы на экране дисплея было меньше пустых строк, надо в служебной строке дисплея сбросить в ноль параметр «авто ПС=ВК» (как пояснялось в разделе 2).

Окончив загрузку, система предлагает создать свою копию. Если вздумали согласиться, то надо «вставить чистую кассету», а иначе будет затёрта исходная запись системы. Запись производится 256-блоками с 1-, 2- или 3-кратными копиями блоков. Чтобы отказаться от копирования системы (это обычный, нормальный шаг), надо в ответ на запрос ДУБЛЕЙ- ввести 0. На запрос ПЕЧАТЬ (и затем ТПУ/УВВ-ПЧ) отвечаем нажатием цифры < 1 >. Затем в диалоге нажимаем <Пробел>, пока не появится буква М с двоеточием, означающая приглашение к вводу директив для «Монитора» ОС МХТИ.

Ниже на скриншоте показан этап загрузки ОС МХТИ 32к и ввод с клавиатуры простейшей исходной программки с именем TEST. Действия при наборе программки – такие же, какие были бы в Фортране-85 или -5М (только с тем отличием, что в Фортране-85 и -5М нет оператора TEXT):

```
D3-28-imitator 1.1   [C:\d3-28-imitator\txt\s4_side2_os-mhti_32k.txt]
File  View
                                9 6 0 0  0 0 0 0  0 0 1 0  0 0 0 0  0 0 0 0
+1+1+1+1+1+1+1+1+1+1+1+1+1+1+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2
+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2
КОПИЯ:ДУБЛЕИ-0
ПЕЧАТЬ:  ТПУ/УВВ-ПЧ:
ОС МХТИ ВЕРСИЯ ЛФТИ:
АДРЕС РП :  3165
АДРЕС ИП : 13000

   ТАБЛИЦЫ:
МАКС.РАБ.ПР.: 21562 Б

М:Л Т:Д
М:В:П
       MAIN PROGRAM TEST
     1     TEXT('      ПРИВЕТ, РЕБЯТА!   HELLO WORLD! ')
     2     END
:
М:КОМП Т:

BD =  3248:
М:С
Р:С
      ПРИВЕТ, РЕБЯТА!   HELLO WORLD!
Р:_
```

Пояснения к этой картине: опытным путём выяснилось, что перед набором программки следует указать – «листинг» или «компиляция». Для компиляции директива выглядела бы так: КОМП Т : Д; при этом набор строк программки сопровождался бы выводом на экран адресов сразу скомпилированных операторов программки. На картинке показан выбор не компиляции, а «листинга» с выводом на дисплей: Л Т : Д. Вот ещё вариант, он более правильный: <W>, <N>, т. е. В : Н, что означает «ввод новой программы»; при этом система объявляет себя как ВТ-9Р-МХТИ и ожидает нажатия клавиш <Пробел>, <М>. (Первый вариант необходим для продолжения ввода ранее набранной программки.)

Затем была дана директива ввода программы: В : П. Курсор перешёл в начало следующей строки. После нажатия клавиши <Пробел> нажимаем <М>; система выводит слова MAIN PROGRAM и ожидает ввода имени программки, в нашем примере это TEST. Ввод имени и вообще ввод каждой строки программки надо завершать нажатием клавиши ВК на клавиатуре терминала, т. е. <Backspace> на ПК-клаве.

Набранная программка скомпилирована по команде КОМП Т : <Пробел>. После того, как система вывела значение BD, был нажат <Пробел>. Затем: <S>

и ещё раз <S> – для запуска программки (как в Фортране-5М). Видно, что система исполнила программку и осталась в режиме «Р» (режим «Работа»), так что повторными нажатиями клавиши <S> программка исполняется снова:

```
D3-28-imitator 1.1   [C:\d3-28-imitator\txt\saved_in_os-mhti-32k.txt]
File  View
                          9 6 0 0  0 0 0 0  0 0 1 0  0 0 0 0  0 0 0 0
P:C
      ПРИВЕТ, РЕБЯТА!   HELLO WORLD!
P:C
      ПРИВЕТ, РЕБЯТА!   HELLO WORLD!
P:M
M:БИБЛ.МЛ:ОСТ КФ:0 + NN:0
M:БИБЛ.МЛ:ЗП:И V:1
   1 ИП TEST   1    59    469
M:БИБЛ.МЛ:ЗП:КБ
M:БИБЛ.МЛ:К Т:Д
  N3  Т  ИМЯ   V  БАИТ    КС  123
   1 ИП TEST   1    59    469 !!!?
M:БИБЛ.МЛ:ОСТ КФ:1 + NN:0
M:КОМП Т:

BD =  3248:
M:БИБЛ.МЛ:ЗП:Р К:1  V:1
   2 РП TEST   1  3247  45116
M:БИБЛ.МЛ:ЗП:КБ
M:БИБЛ.МЛ:К Т:Д
  N3  Т  ИМЯ   V  БАИТ    КС  123
   1 ИП TEST   1    59    469 !!!
   2 РП TEST   1  3247  45116 !!!?
M:_
```

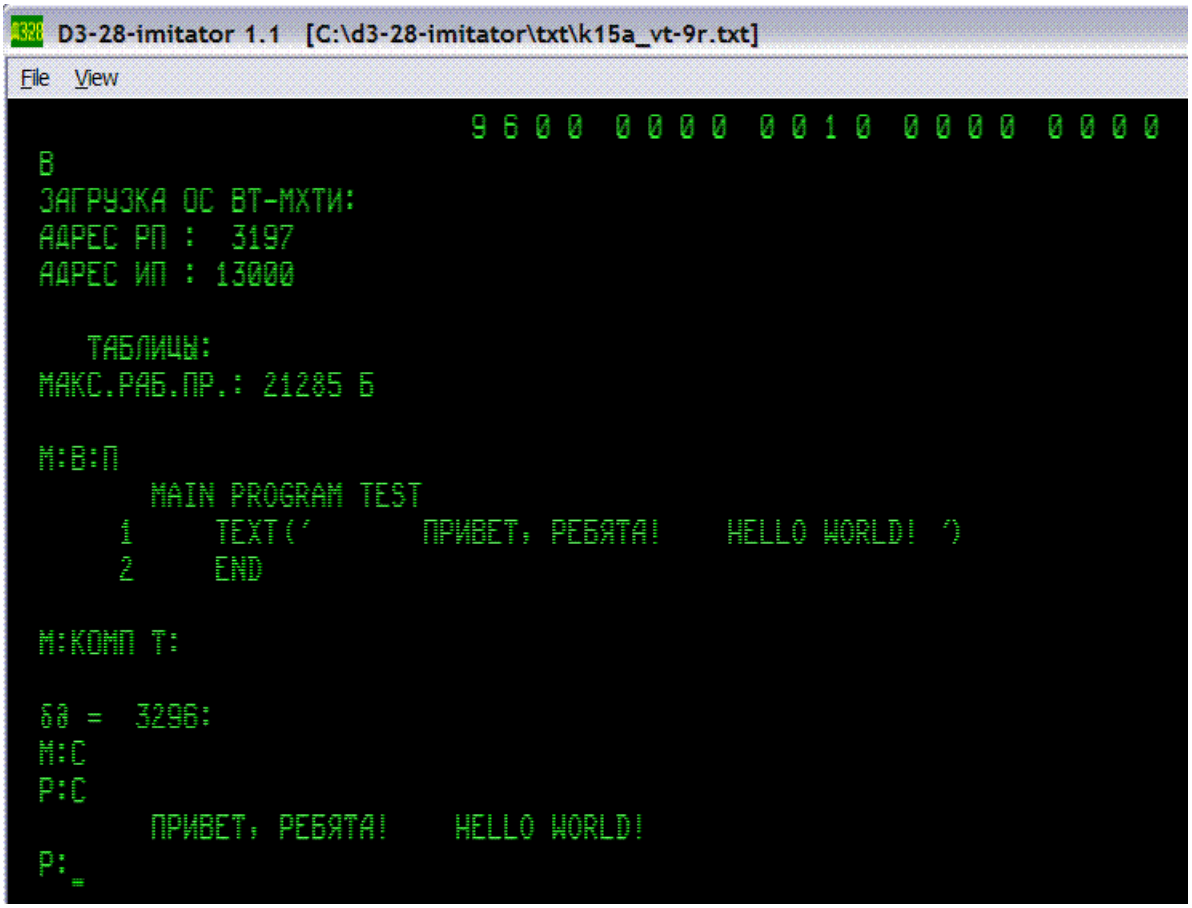
Нажатием <М> возвращаемся из режима «Р» в режим «М», т. е. в Монитор». Далее на скриншоте видны очень важные команды – запись программок из ОЗУ в «библиотеку на магнитной ленте». В общих чертах всё похоже на Фортран-85 или -5М; но есть нюансы. Работа с МЛ начинается командой БИБЛ.МЛ (нажатием Б, т. е. <$В_{лат}$>). Чтобы избежать порчи записей в «библиотеке», надо указывать количество оставляемых файлов – параметр ОСТ КФ; для чистой ленты указываем 0, а для ленты с записями – количество сохраняемых программ, и нажимаем <Пробел>. Параметру NN даём значение 0, и затем – <Пробел>. Таким путём удаётся записывать в «библиотеку на МЛ» исходные и рабочие программки. При этом для записи рабочей программы особенно важна последовательность действий: сначала смотрим каталог ленты: БИБЛ.МЛ : К Т : Д, затем задаем БИБЛ.МЛ : ОСТ КФ, затем компилируем исходную программку: КОМП Т, и записываем рабочую: БИБЛ.МЛ : ЗП : Р К : 1 с желаемым номером версии V.

Параметр K : 1, вероятно, задаёт «номер канала» для рабочей программы, так что единица означает «дисплей»; если указать другое значение, то программка в имитаторе зависнет: она будет ожидать ответа (СИП) от какого-то другого периферийного устройства.

Есть и другие компилирующие системы программирования, разработанные в МХТИ для машины Д3-28 с памятью 32к, которые можно охарактеризовать как фортрано-подобные. По-видимому, наиболее близка к Фортрану-5М (и -85) система «ВТ-9Р МХТИ»:

ВТ-9Р МХТИ

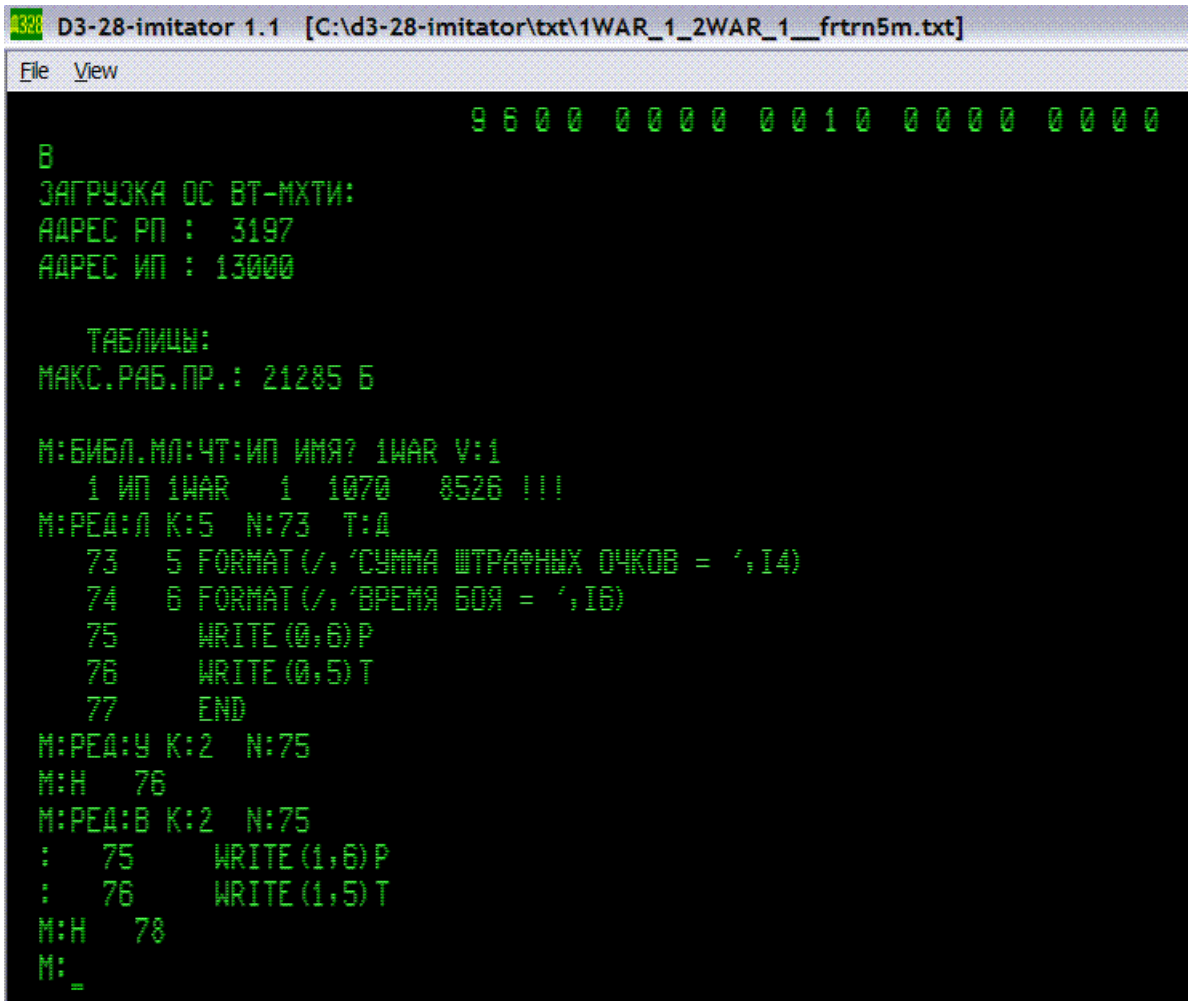
Оцифровку системы ВТ-9Р МХТИ из своей уникальной коллекции кассет с программным обеспечением ЭВМ «Электроника Д3-28» выполнил и любезно предоставил **Виталий Колесник**, автор сайта d3-28.ru. Эта система загружается 256-байтными блоками загрузчиком с КП=610. В папке [txt] байт-коды системы находятся в файле k15a_vt-9r.txt. В имитаторе перед загрузкой следует сбросить в ноль параметры служебной строки дисплея (нажатием <F8>, <F9>, <F8>); программа сама установит систему команд № 2. Загрузка – <L>, <K> <C>, <S>; номера загружаемых блоков индицируются отсветкой на табло X. После <F5> в начальном диалоге – <Пробелы>. Вот МХТИ-аналог примера с версией ЛФТИ:



Команды работы с МЛ в этой системе такие же, как и в ОС версии ЛФТИ. Чтобы записать существующую в ОЗУ ИП, подаём команду БИБЛ.МЛ : КАТАЛОГ, затем БИБЛ.МЛ : ОСТ КФ : *число* <Пробел> +NN : <Пробел> , где *число* указывает, сколько файлов мы хотим оставить на МЛ в начале ленты. ИП командой БИБЛ.МЛ : ЗП : И добавится к ним в конец. Есть и другой режим записи: после NN перечисляем дальнейшие выборочные номера оставляемых файлов, подтверждая каждый номер <Пробелом>; завершается список номеров также <Пробелом>. Система в процессе отбора этих номеров затирает имеющуюся в ОЗУ программку, поэтому должна быть набрана новая ИП. Для записи РП надо подавать те же подготовительные команды, затем компилировать ИП, и сразу подавать команду записи: БИБЛ.МЛ : ЗП : Р k : 1 V : *номер_версии*. После каждой записи следует записывать КБ.

Сходство с Фортраном-5М видно, в частности, в том, что записи Фортрана-5М успешно читаются системой ВТ-9Р МХТИ. Можно смотреть их листинг; и даже компилировать и запускать, если отредактировать различие в нумерации «каналов»: в Фортранах дисплей считается «каналом 0», а в системах ВТ-МХТИ – «каналом 1». Редактирование номера канала для вывода на дисплей поясняется следующим примером:

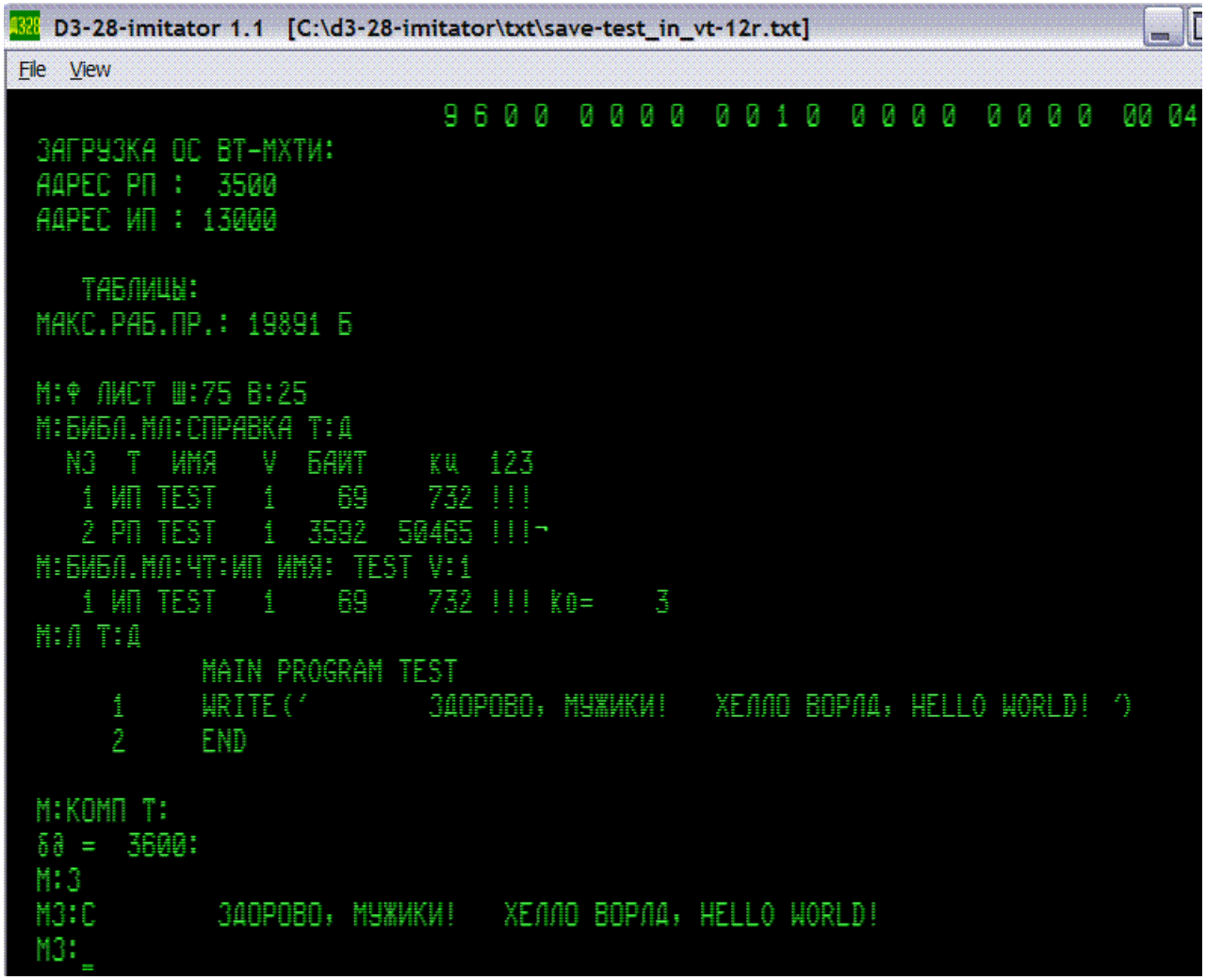
Скриншот иллюстрирует действия по редактированию команд WRITE в игровой тест-программке 1WAR, написанной в Фортране-5М. В верхней части картинки виден начальный диалог системы ВТ-9Р МХТИ, затем – команда чтения исходной программы «1WAR версия 1» из файла 1WAR_1_2WAR_1__frtrn5m.txt:



Команда РЕД : Л выводит на листинг К штук строк, начиная с номера N. Видим, что надо изменить строки 75 и 76 – заменить в списке параметров операторов WRITE номер канала 0 на 1. Для этого мы удаляем 2 строки, начиная с 75-ой строки, командой РЕД : У К : 2 N : 75. Команда Н проверяет нумерацию строк и показывает их количество (с учётом нулевой строки с именем программы). Затем мы командой РЕД : В К : 2 N : 75 аналогично требуем вставить 2 строки; система в ответ позволяет заново набрать строки 75 и 76, что мы и делаем. Команда Н показывает, что обновлённые строки добавились в программу. Редактирование выполнено. Этого достаточно для работы программы 1WAR в системе ВТ-9Р МХТИ, но – не в ОС ВТ-МХТИ 32к версии ЛФТИ (там адреса переменных типа INTEGER иные). В Фортране-5М и -85 работают практически такие же приёмы редактирования программок.

Ещё одна фортраноподобная (по внешней форме команд) система – ВТ-12Р МХТИ; её оцифровку выполнил и любезно предоставил **Виталий К**. В папке [txt] она находится в файле k17a_vt-12r.txt. Загрузчик с КП = 612.

Запускается она так же, как ВТ-9Р, но в работе проявляются отличия. Перед вводом программы приходится командой Ф увеличивать «ширину листа» (см. скриншот ниже), т. к. длина строки, установленная по умолчанию, слишком коротка. При записи на МЛ вместо команды ОСТ КФ нужна команда РЕД : ОСТ КФ, а версия обозначается буквой *ж* вместо v. Вместо *кс* почему-то *кц*. Вместо команды КАТАЛОГ – команда СПРАВКА.



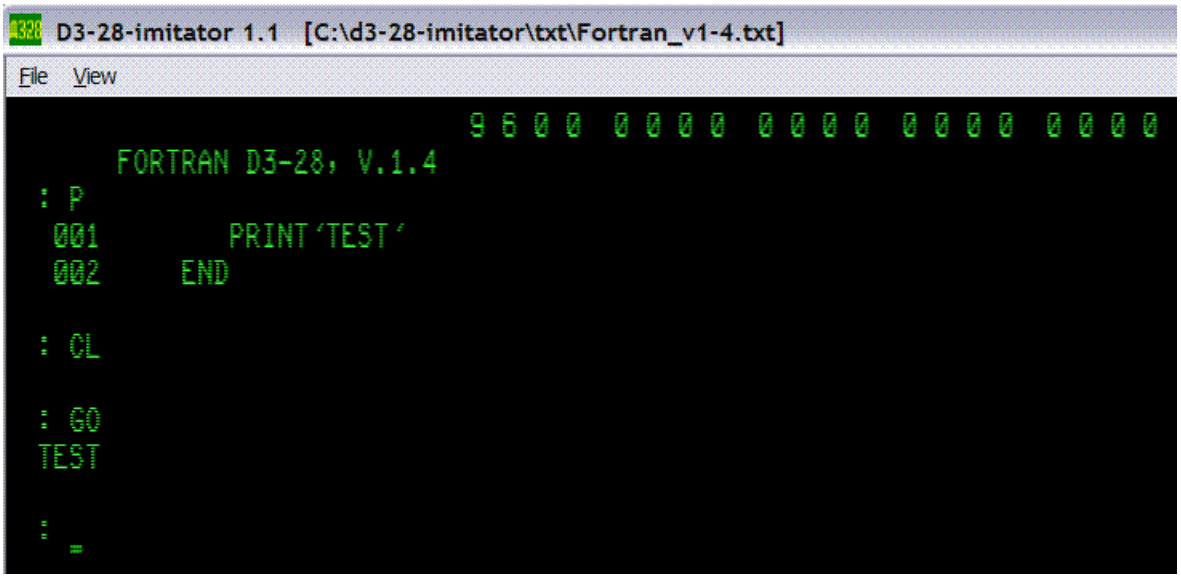
На скриншоте показан начальный диалог ВТ-12Р МХТИ, коррекция формата – «ширины и высоты листа», просмотр каталога записанной в этой системе «ленты» с ИП и РП «TEST», чтение ИП «TEST», листинг с выводом на дисплей, компиляция без вывода на дисплей, и запуск. Командой к запуску служит нажатие <Z>; при этом приглашение на экране принимает вид МЗ : , а для собственно «счёта по программе» следует затем нажать <S> (всё это в устанавливаемом самой системой регистре РУС).

Ввод новой программы с клавиатуры терминала инициируется командой В : Н, т. е. нажатием <W>, <N>. Система объявляет себя как ВТ-12С-МХТИ (в отличие от предыдущей системы, которая после команды В : Н объявляет себя как ВТ-9Р-МХТИ). Затем надо нажать <Пробел>, <М> для ввода MAIN PROGRAM или <S> для ввода SUBROUTINE, и набрать имя программы. Этот ввод, как и дальнейший ввод строк завершается нажатием ВК, т. е. <Backspace> (что характерно для фортраноподобных систем, работающих на машине Д3-28). В этой системе нет оператора TEXT, его заменяет WRITE.

Странное отличие этой версии ВТ-МХТИ от предыдущих – несовместимость сделанных ими записей. Попытка листинга «чужой» программки ведёт к выводу на экран какого-то «мусора». Но трудно сказать, является ли это свойством систем или здесь проявляется несовершенство имитатора, поскольку у нас нет подробной информации о работе этих программ на реальной Д3-28.

Фортран Кочеткова

Создатель сайта https://d3-28.ru/ **Виталий К**. предоставил также описание https://d3-28.ru/programmy/fortran-d3-28-kochetkova-v-m-liizht/ и оцифровку версии Фортрана, заметно отличающейся от всех упомянутых выше версий.



Приведённый выше скриншот иллюстрирует простейший тест этой системы в имитаторе. В папке [txt] байт-коды «Фортрана Кочеткова» находятся в файле Fortran_v1-4.txt. Загрузка: <C>, <L>, <K> (проверка контрольной суммы загрузчика КП = 5000), <C>, <S>. Перед включением ДУП-ЛИН клавишей <F5> в режиме **View** > **Display** следует включить регистр ЛАТ клавишей <Ctrl>.

Здесь же перечислю ряд других системных программ, для которых результаты оцифровки и сканирования документации любезно предоставлены уважаемым **Виталием К**.

СПП-8 МХТИ – в папке [txt] это файл spp-8___KP-65890_N-5297.txt.
Пояснения: https://d3-28.ru/programmy/spp-8-sistema-podgotovki-programm/ ,
см. также текст в конце файла spp-8_mnemocodes.txt.
Загрузка: <L>, <K> (проверка КП = 65890). Запуск: <C>, <S>, после включения ДУП-ЛИН клавишей <F5> в режиме **View** > **Display** система выводит на экран строчку «СПП-8 МХТИ», и Д3-28 останавливается. Переходим к **View** > **Machine**, нажимаем снова <S>; в режиме **View** > **Display** на экране появляется двоеточие – приглашение к вводу директив. Пример исполнения директив приведён в pdf с дизассемблингом «spp-8_diz.pdf» по ссылке выше на сайте d3-28.ru.

ЭТ-10 – «операционная система Цейтлина ЭТ-10»,
в папке [txt] это файлы k12b_et-10__KP-3550.txt и k12b_et-10__red-28213.txt (версия с отредактированным мной шагом 28213 для устранения ошибки в дизассемблере).
Описание и команды системы ЭТ-10 – в брошюре А. С. Цейтлина:
https://d3-28.ru/programmy/et-10-tsejtlina-a-s-moskovskij-vniiz/ Обсуждение и простые примеры:
https://d3-28.ru/forum/topic/operatsionnaya-sistema-et-10/

МАШЯЗ – в папке [txt] это файлы: mashyaz___KP-108069_N-8130.txt или
k7a__mashyaz_KP-108090_N-8130.txt
Документация: https://d3-28.ru/dokumentatsiya-d3-28/mashyaz/ .
Перед загрузкой в имитатор надо установить «бит b4 = 1», сбросить в ноль параметры служебной строки дисплея: <F8>, <F9>, <F8> при **View** > **Display**. Загрузка и запуск: **View** > **Machine**, <C>, <L>, <K> (проверка КП), <C>, <S>, включение ДУП-ЛИН клавишей <F5> в режиме **View** > **Display**. Эта программа может работать с ОЗУ 128к, более подробное пояснение см. в разделе 6.

МИНИМОНИТОР – в папке [txt] это файл k7b_minimonitor.txt.
Порядок запуска описан в конце этого же txt-файла.
Документация: https://d3-28.ru/dokumentatsiya-d3-28/minimonitor/

Версии ОС ВТ-128-МХТИ – фортраноподобные системы, работающие с ОЗУ 128к. На данный момент из оцифровок Виталия получены байт-коды трёх версий: 128R, 128Д, 128С с распечатками больших пояснительных текстов. Мы решили не втискивать этот обширный материал в архив с имитатором. Файлы с байт-кодами версий ОС ВТ-128-МХТИ, скриншоты и пояснения размещены на сайте https://d3-28.ru/ . Там же на форуме предполагается проводить обсуждение этих систем:
https://d3-28.ru/forum/topic/os-vt-128-mhti-s-lenty-19-a-ispytanie-v-imitatore/
https://d3-28.ru/forum/topic/os-vt-mhti-128s-primery-raboty-v-imitatore/
(см. также краткое обсуждение на форуме РТ20:
http://rt20.mybb2.ru/viewtopic.php?p=1979982#p1979982).

Выстра

«Выстра» это – «вычислительная система-транслятор». Как с ней работать, здесь не рассказываю полностью, всё изложено в Приложении 2; (вернее, не всё, а кое-что из самого важного; многое приходится выяснять экспериментально, и занятие это довольно интересное: язык Выстры красив своей краткостью). Рассмотрим запуск Выстры и некоторые первые шаги.

Выстра запускается в имитаторе с его настройкой по умолчанию: «бит b4=0», служебная строка дисплея: 9 6 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 …

В свежезапущенном или освежённом ресетом имитаторе открываем командой **File** > **Open txt-file as Tape** файл D3-28_Vystra__KP-113478__N-9478.txt. Загружаем его в режиме **View** > **Machine** нажатием <L>. Проверяем нажатием <$K_{лат}$> контрольную сумму: КП = 113478. Затем делаем сброс счётчика команд ПК-клавишей <C>, и даём команду «старт» ПК-клавишей <S>.

Переходим к **View** > **Display,** и включаем ДУП-ЛИН ПК-клавишей <F5>. На экране дисплея появится приглашение к заданию функции: буква Ф.

Например, наберём простейшую программку. Для этого нажимаем < I > и подтверждаем командой ПС, т. е. нажимаем <Enter>. Маркер переходит в следующую строку. Набираем три ноля 000 и подтверждаем нажатием <Enter>. Маркер сам опускается в нижнюю строку экрана дисплея, там выводятся три ноля и пробел. Такова специфика набора программы в Выстре – автоматически выводится номер строки и за ним пробел, причём заполнение экрана идёт снизу вверх, так что при наборе каждой новой строки прежние строки заново выводятся над набираемой (увы долго, для ускорения пробуйте сместить окно имитатора максимально направо, чтобы его б*о*льшая часть была не видна на мониторе ПК). Кроме того, Выстра всё время норовит сама переключать регистр РУС –> ЛАТ; поэтому его надо часто проверять, нажимая <F1>, и менять на нужный клавишей <Ctrl>. Допустим, мы набрали вот такой текст (номера строк и пробелы после номеров строк Выстра делает сама, их не надо набирать):

```
000 'DC';
001 'IO'KT,ПРИВЕТ,KC;
002 'EN';
003 #
```

Здесь только слово ПРИВЕТ набирается в РУС-регистре; остальные символы в ЛАТ-регистре. Набор каждой строки, кроме последней, завершается командой ПС, т. е. нажатием <Enter>. Набор последней строки, с обязательным значком #, означающим конец программы, надо завершать командой ВК, т. е. нажатием <Backspace>; тогда Выстра в следующей строке выведет букву Ф.

Нажмём <K> и ПС, это означает команду компиляции. После компиляции, сопровождающейся выводом на экран номеров строк и операторов, в конце снова увидим букву Ф. Нажимаем <E> и ПС – это команда к выполнению (экзекуции) скомпилированной программки, и наблюдаем результат. Вот как это выглядит вместе с набором программки:

```
000 'DC';
001 'IO'KT,ПРИВЕТ,KC;
002 'EN';
003 #
Ф K

000   00512
001   00516
002   00531
003
  10112  00537  000004Ф E
ПРИВЕТ
Ф _
```

Поскольку текст программки ещё имеется в ОЗУ, его листинг можно распечатать. Для этого нажимаем <Home>, т. е. «включаем ТПУ», переводим клаву в режим РУС нажатием <Ctrl>, нажимаем ПК-клавишу <$Р_{лат}$> (при этом на экран выводится буква П, означающая команду «печатать»), и нажимаем <Enter>, т. е. ПС. В Выстре почти все действия надо подтверждать нажатием ПС.

Можно текст программки сохранить «на ленте». С этой целью следует придумать и запомнить для будущих считываний имя программы. Назовём её, например, TEST. Для записи исходного кода нашей программки «на ленту» надо не забыть вынуть кассету с Выстрой (**File** > **Close txt-file as Tape**), заготовить в нашей папке [txt] новый txt-файл с каким-то именем, «вставить его в НМЛ» (**File** > **Open txt-file as Tape**), и подать Выстре команды:

<A> ПС TEST ПС.

По командам ПС маркер переходит в новую строку. В результате последнего ПС образуется желаемая запись в txt-файле. Чтобы убедиться, что запись получилась, можно перезапустить имитатор клавишей <Escape>, снова загрузить и запустить Выстру, «вынуть» её, «поставить кассету» с проверяемой записью и считать её выстровской командой чтения с ленты:

<C> ПС TEST ПС. Здесь, как и при записи, слово TEST это имя программы.

В случае удачного считывания имя программы появляется ещё раз, в следующей строке. Компиляция нажатием <K> ПС и экзекуция нажатием <E> ПС покажут, что всё получилось нормально.

Опыты можно продолжить следующим образом. Переведём клаву в режим РУС, нажав <Ctrl>, и нажмём <V> ПС – это команда Ж, означающая уничтожение текста программки в памяти Выстры; при этом скомпилированная программка в ОЗУ машины сохраняется. Снова включим РУС и нажмём <G> ПС – это команда Г, означающая уничтожение самой Выстры. С этого момента машина уже не работает по программе, а находится в состоянии «останова с индикацией».

Скомпилированную программку теперь можно запускать командами с пульта машины: <C> – сброс счётчика команд, и затем <S> – старт. Переходя к просмотру экрана дисплея, увидим, что после каждого такого запуска на экране появляется слово ПРИВЕТ, т. е. откомпилированная программка работает самостоятельно, без Выстры. Нажав «на пульте машины» <$C_{лат}$> и затем <Z>, можно сохранить её «на ленту» – на новую, или на ту же; во втором случае программка запишется как четвёртый файл, потому что первые три файла это две части заголовка и сам выстровский текст нашей программки. Скомпилированная в машинные коды и записанная машинной командой ЗЛ программка считывается с ленты и запускается без Выстры: «перемотка ленты», если надо, <C>, <L>, <C>, <S>.

Для иллюстрации работы Выстры с НМЛ в папке [txt] есть пример «ленты с библиотекой выстра-программ»: frmj_inpx_inpu_outx_test__vystra-text.txt; в конце этого txt-файла даны подробные пояснения. В начале записана не библиотечная (не имеющая специального заголовка) программа «FRMJ» в машинных кодах; она запускается, как обычно (<C>, <S>), и строит на экране дисплея «фракталы» типа «Mandelbrot» или «Julia». Эта программа скомпилирована Выстрой в имитаторе из исходной программы FRMJ, которая записана в «библиотеку» Выстрой в формате выстра-текста: две части заголовка плюс основной блок байт-кодов, содержащий выстровский текст программы FRMJ. Будем считать, что исходный текст FRMJ есть первая запись в собственно выстровской «библиотеке». В этой программе имеется очень полезная для всех выстровских программ процедура INP(X) – ввод числа с клавиатуры дисплея; в самой Выстре нет такого ввода с клавиатуры, встроен только ввод чисел с пульта машины оператором INPUT(X). Выстра позволяет выделять полезные процедуры из исходных текстов – записывать их в виде отдельных программ.

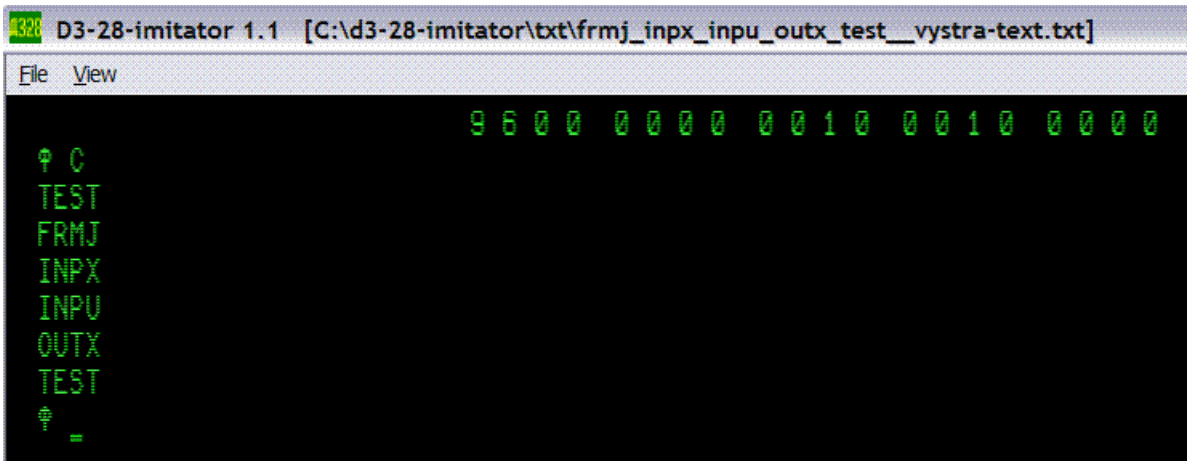
Вторая запись в «библиотеке», с именем INPX, это как раз упомянутая полезная процедура. К голой процедуре INP(X), заканчивающейся в строке 033, там добавлено ещё семь строк – чтобы с ними запись стала программкой, которую можно после считывания в Выстре скомпилировать (командой К ПС) и запустить (командой Е ПС) для контроля. Для применения же INP(X) в новых программах эти лишние семь строк надо будет удалить.

Третья запись в библиотеке – программка с именем INPU, иллюстрирующая один из способов применения встроенного в Выстру оператора INPUT(X) для ввода чисел с пульта машины через регистр X. Понять этот пример поможет листинг в конце данного txt-файла.

Четвёртая запись в библиотеке – процедура OUT(X). Эта процедура выводит свой числовой аргумент X на экран в формате "мантисса и порядок числа": один знак мантиссы до десятичной точки, девять знаков мантиссы после десятичной точки, затем символ порядка "E" и два знака порядка числа. Она даёт возможность выводить на экран числа с большой абсолютной величиной порядка (до 98 включительно).

Пятая запись в библиотеке – рассмотренная выше программка TEST. Здесь она играет роль метки «конец библиотеки», так как в Выстре нет аналога фортрановской команды КБ. Если условиться всегда заканчивать библиотеку программкой с именем TEST, то список имён выстровских программ на ленте можно получать командой чтения программы с именем TEST,

потому что Выстра, отыскивая на ленте заданную программу, выводит на экран имена всех библиотечных программ, попадающихся перед заданной. В нашем примере это выглядит так: после буквы Ф подаем команду С, подтверждаем её нажатием ПС (т. е. <Enter>), набираем имя TEST, подтверждаем нажатием ПС. Выстра читает и выводит на экран имена попадающихся на ленте выстровских программ, пока не найдёт и не загрузит в ОЗУ заданную программу:



Допустим, мы решили теперь загрузить FRMJ. Важны два нюанса. Во-первых: надо удалить из ОЗУ уже ненужную программку (т. е. TEST), <u>в противном случае новая загрузка подстыкуется к концу предыдущей</u>. Команда удаления программки из ОЗУ: Ж ПС. Во-вторых, надо сделать перемотку назад. В имитаторе для этого проще всего «вынуть» и снова «вставить кассету». Есть и другой способ. В Выстре нет команды перемотки назад, но зато <u>Выстра допускает многократную остановку и запуск</u>. Поэтому можно переходить к **View** > **Machine**, и нажимать <Enter> (это имитация нажатия кнопки Ш на пульте машины) – так в имитаторе останавливается Выстра. Затем нажимаем ← – это имитация кнопки «перемотка назад». И нажимаем <C>, <S> – это запуск Выстры; при этом она снова выводит на дисплей букву Ф, так что на экране будут две буквы Ф рядом. Наконец, подаём команду считывания с ленты: С ПС FRMJ ПС. Выстра ищет и загружает программу FRMJ, выводит на экран в новой строке её имя, и выводит очередное приглашение Ф:

Допустим, теперь мы хотим вывести на экран листинг этой программы. Команда листинга: И ПС *трёхзначный_номер_строки* ПС. Выстра выведет столько строк, сколько поместится на экране, с номерами меньшими заданного, <u>включая и строку с номером равным заданному – такую строку условимся называть *текущей*</u>. Однако если задан номер строки, больший номера последней строки в программе, то выведется лишь строка с номером 000. Значит, желательно

заранее знать номер последней строки. Для этого полезно в нулевую строку помещать комментарий с указанием номера последней строки; тогда, сделав сначала листинг нулевой строки, мы узнаем допустимые номера строк.

Для примера посмотрим листинг первых трёх строк в FRMJ:

```
000 ¤ FRMJ 141 СТРОКА ;'PR'INP(X);
001 'DC'B,W[6];'KO'405,W,404,B,413,1312;B+=1;'IF'(B/2-INT(B/2+.1))#0;B+=1;
002 'FI';B=16*INT(B/2+.1);'KO'405,B,413,1200,1100,1200,1104,5,703,701,413,
```

Комментарием является текст между символом ¤ и точкой с запятой; из него видно, что номер последней строки в FRMJ есть 141.

Обратим внимание: маркер находится в начале текущей строки. Если начать набирать символы, то начало текущей строки будет редактироваться – таким путём можно добавить туда комментарий или, например, метку. Если нажать ВК (т. е. <Backspace>), то появится признак ожидания новой команды: буква Ф. Если теперь подать команду **)** ПС, то из программы будут удалены все строки перед текущей строкой, а если подать команду **(** ПС, то будут удалены все строки после текущей строки. Так при должном усердии можно нарезать программу на части, записать части на ленту или на разные ленты, а затем считывать их, подстыковывая к концу набираемого текста. После каждой подстыковки следует подавать команду перенумеровки: <9> ПС; она восстанавливает нумерацию строк в порядке возрастания номеров.

Пока маркер находится после листинга в начале текущей строки, можно эту строку отредактировать всю целиком, удалив в ней весь текст нажатиями клавиши «Забой», т. е. <Delete> в имитаторе. Затем можно ввести новый текст и нажать ВК. Чтобы редактировать не всю текущую строку, а только некоторое место в ней, маркер можно подвести к этому месту командами АР2 С (это команда «маркер вправо»), АР2 D (команда «маркер влево»). По командам АР2 А («маркер вверх») или АР2 В («маркер вниз») происходит новый вывод строк на экран, так что новой текущей строкой окажется предыдущая или следующая. Завершение редактирования и переход к запросу очередной функции (Ф) производится нажатием ВК.

FRMJ служит примером программы, требующей много памяти. После её компиляции следует перед её запуском удалить из ОЗУ исходный текст командой Ж ПС и удалить саму Выстру командой Г ПС. После этого FRMJ запускается с пульта машины: <C>, <S>.

В папке [txt] есть также файл frmj_KP-38246_N-3445.txt – это раскодировка оцифровки реальной рабочей программы FRMJ, скомпилированной в давние времена на реальной машине Д3-28. Может возникнуть вопрос: почему у скомпилированной в имитаторе FRMJ другая контрольная сумма: КП = 38206? Оказывается, значение КП здесь зависит от того, запускалась ли (и как) программка FRMJ перед подсчётом контрольной суммы. На реальной Д3-28 перед записью программка была подробно опробована. Если мы и в скомпилированной в имитаторе программке до её «записи на МЛ» выполним счёт с параметрами ТИП=0, R=2, N=24, D=0.15, X=–3, Y=1.2, P=38, S=18, затем ЧТО=1, M=–4, затем «включим ТПУ» и выберем ЧТО=9, затем снова ЧТО=9, дождёмся вывода матрицы чисел «А», затем ЧТО=15 (не забывая подтверждать все вводы нажатием ПС), затем на пульте машины нажмём <C>, <K>, то увидим как раз КП =38246. Полученную так программку с КП = 38246 мы можем записать в новый txt-файл (командами <C>, <Z>), и побайтно сравнить байт-коды программки в этом txt-файле с байт-кодами раскодировки реальной давней записи frmj_KP-38246_N-3445.txt. Оказывается, они совпадают. Таким образом, можно утверждать, что имитатор имитирует работу компилятора Выстра точно (по крайней мере – в данном примере :-).

6. Управление расширенной до 128к памятью

Объём адресного пространства ОЗУ Д3-28, с которым работают машинные команды, составляет 32 килобайта. Он разбит на 4 страницы (по 8 килобайт); в имитаторе страницы условно обозначены как P0, P1, P2, P3:

Страница ОЗУ	16-ричные значения адресов	10-тичные значения адресов
P0	0000 0000 – 0115 1515	0 – 8191
P1	0200 0000 – 0315 1515	8192 – 16383
P2	0400 0000 – 0515 1515	16384 – 24575
P3	0600 0000 – 0715 1515	24576 – 32767

Физическое ОЗУ Д3-28 состоит из 16 сегментов (по 8 килобайт) с условными номерами 00, 01, 02, … , 12, 13, 14, 15. К четырём страницам адресного пространства могут быть «подключены» любые четыре сегмента в любом порядке. Допускается даже подключение одного и того же сегмента к разным страницам, так что разные адреса будут указывать на одни и те же физические ячейки памяти.

Подключением сегментов к страницам управляет двухбайтовый «регистр сегментов»: его четыре тетрады B1 A1 B2 A2 (с обозначением B1 A1 – первый байт, B2 A2 – второй байт) задают и хранят номера сегментов, подключенных к страницам:

страницы адресов ОЗУ: P3 P2 P1 P0
тетрады регистра сегментов: B1 A1 B2 A2

При включении питания машины B1 = 15, A1 = 14, B2 = 13, A2 = 12, т. е. к страницам P3, P2, P1, P0 подключены сегменты 15, 14, 13, 12 соответственно. В старых Д3-28 с физическим ОЗУ 32к других подключений быть не может. Так же обстоит дело в имитаторе в режиме «32к» (его можно включить выбором режима памяти, открывающимся нажатием <M> после **View** > **Display** , <F1>).

В Д3-28 с физическим ОЗУ 128к машинная команда ONSEGM (её код: 0412 1506) посылает в регистр сегментов значения из двухбайтового регистра R_{10}: байт B1 A1 регистра сегментов приравнивается старшему байту регистра R_{10}, байт B2 A2 регистра сегментов приравнивается младшему байту регистра R_{10}. (Напоминание: старшему байту двухбайтовых регистров служебной зоны ОЗУ соответствует меньший адрес. Служебная зона ОЗУ занимает в адресном пространстве адреса 32512 – 32767, т. е. она находится в конце страницы P3). ONSEGM – единственная команда в системе команд машины Д3-28, управляющая подключением сегментов к страницам. По её завершении в R_{10} автоматически заносится 0000 0000.

Имитатор после запуска или ресета работает в режиме «128к». Поскольку эта версия имитатора получена доработкой 32к-версии имитатора, в ней машинные команды Д3-28 обращаются к фиксированному массиву ячеек памяти, имитирующему ОЗУ объёмом 32 килобайта (наследство прежних версий). Наряду с этим вспомогательным «32к-массивом» в имитаторе есть массив, имитирующий 16 сегментов объёмом 128 килобайт, но до первого вызова команды ONSEGM он заполнен «мусором» 1515 и не используется; при этом «регистр сегментов» в имитаторе хранит для страниц следующие значения номеров сегментов: P3 = 15, P2 = 14, P1 = 13, P0 = 12.

При исполнении команды ONSEGM в имитаторе прежде всего текущее содержимое вспомогательной памяти «32к» записывается в сегменты «128к-массива», соответствующие номерам сегментов из регистра сегментов P3, P2, P1, P0. Затем в P3, P2, P1, P0 заносятся значения тетрад регистра R_{10}, так что P3 P2 = старшему байту, P2 P0 = младшему байту регистра R_{10}). Затем во вспомогательный «32к-массив» памяти (с которым будут работать машинные команды) переписывается содержимое сегментов «128к-массива» соответственно текущим номерам сегментов из P3, P2, P1, P0. И затем в регистр R_{10} заносится ноль.

Текущее подключение сегментов к страницам в имитаторе можно посмотреть в режиме **View** > **Display** нажатием <S> после <F1>.

Для проверки ОЗУ и системы команд Д3-28 предназначен «028-тест» из комплекта стандартного ПО. Есть две версии этого теста: старая версия для 32к-машин (с контрольной сумой КП =60000), и новая версия (с контрольной суммой КП = 73370). Тест состоит из четырёх частей; выполнение каждой части из первых трёх индицируется на табло машины цифрами, изображающими номер части теста. После выполнения четвёртой части старый тест выводит на табло машины цифры, символизирующие «60-летие Октября». Новый 028-тест после четвёртой части выводит на табло номер теста, номер прохода (тест работает повторно, пока машина не будет остановлена), и обнаруженный объём физической памяти. В папке [txt] магнитную ленту со старой версией теста имитирует файл:

D3-28_test_028__KP-60000_N-5999.txt

Новая версия 028-теста (пригодная для машин с ОЗУ 128к) оцифрована с МЛ **Виталием К**; она находится в файле

028-test_KP-73370_N-8921.txt

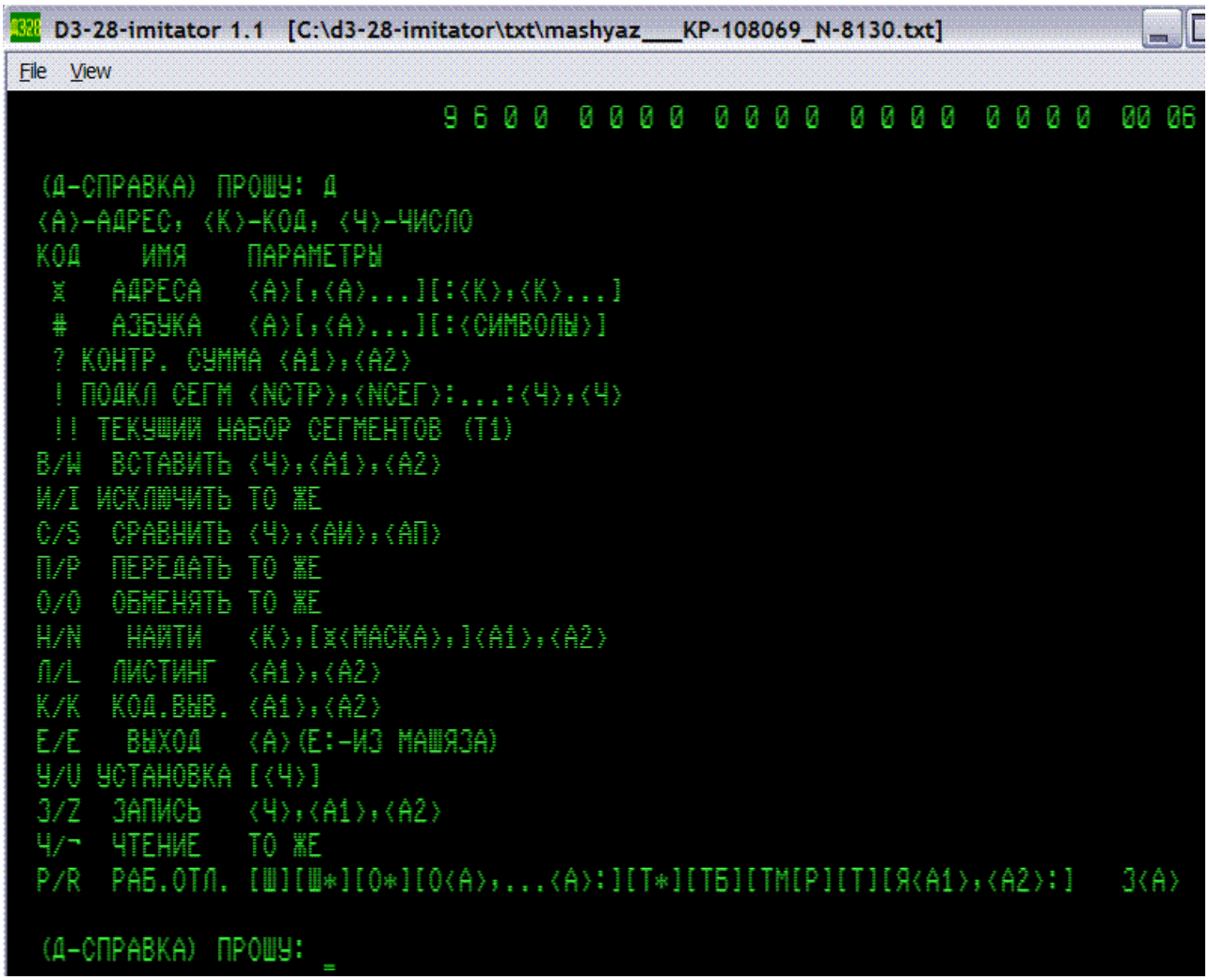
Загрузка 028-теста: <C>, <L>, <K> (проверка контрольной суммы). Запуск: <C>, <T>, <S>.

В тесте каждое обнаружение ошибки выполнения машинных команд ведёт к останову машины. Проверка арифметических команд в тесте основана на сравнении погрешности многократных вычислений с заданным в тесте ожидаемым значением погрешности.

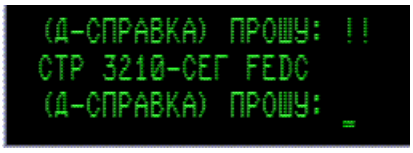
Имитатор не умеет имитировать микропрограммные алгоритмы арифметики Д3-28. В нём используются более точные математические подпрограммы из системной библиотеки Windows, и поэтому он выдает погрешности, немного меньшие ожидаемых в тесте. Из-за этого работа теста останавливается; для продолжения мы должны нажать <S>. В 1-й части нового 028-теста арифметика («пи-тест») проверяется 14 раз – при различных подключениях сегментов ОЗУ; так что приходится 14 раз нажимать <S>, прежде чем 1-я часть теста будет пройдена. Во 2-й части тоже есть проверка арифметики (та же, что и в старом 028-тесте); приходится нажать <S> два раза, прежде чем на табло обозначится завершение 2-й части. 3-я и 4-я части теста проходят в имитаторе без остановок, и затем всё повторяется.

Таким образом, закрывая глаза на ожидаемую неудачу с проверкой арифметики, можно сказать, что 028-тест не находит ошибок имитации команд и ошибок в работе «128к ОЗУ».

Опыт показывает однако, что работой 028-теста ещё не гарантируется правильность имитации всех команд. Более серьёзным подтверждением (но опять-таки не доказательством) успешной имитации может служить работа имитатора со сложными системными программами. Одна из таких программ – МАШЯЗ. О её загрузке на нулевой адрес и запуске рассказано на стр. 66. Вот скриншот с картиной после выполнения директивы <D>, <Enter> (в этой программе ввод директив и продолжение вывода на экран надо подтверждать нажатием ПС, т. е. <Enter>):



В описании программы (см. документацию https://d3-28.ru/dokumentatsiya-d3-28/mashyaz/) предлагается перед работой с программой занести номера подключенных сегментов ОЗУ в служебный регистр T_1. В имитаторе для удобства это сделано уже при запуске имитатора, поэтому номера сегментов можно посмотреть директивой «!!» сразу после запуска МАШЯЗа (если после запуска имитатора не изменялись T_1 и подключение сегментов):



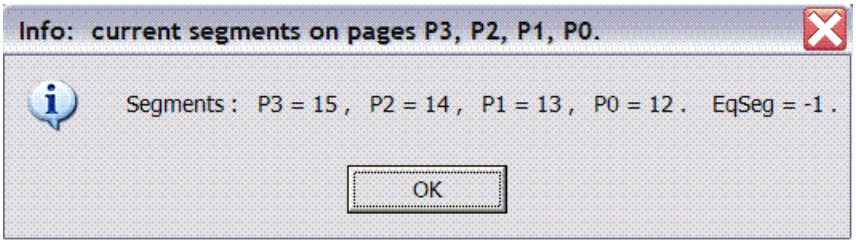
Видно, что программа вывела на экран номера страниц 3 2 1 0 и номера подключенных к ним сегментов в hex-нотации: F E D C. В десятичной записи это 15 14 13 12; так и должно быть сразу после включения машины.

Попробуем директивой «!» подключить к странице P2 сегмент 2, а к странице 1 – сегмент 1 (страницы P3 и P0 не изменяем, т. к. в них находятся служебные регистры, в том числе счётчик команд, и сам МАШЯЗ, загруженный командами <C>, <L> в страницу P0). И затем посмотрим результат опять директивой «!!»:

```
(Д-СПРАВКА) ПРОШУ: !2,2:1,1

(Д-СПРАВКА) ПРОШУ: !!
СТР 3210-СЕГ F21C
(Д-СПРАВКА) ПРОШУ: _
```

Видно, что сегменты 2 и 1 подключились успешно.

Проведём более интересное испытание. Освежим имитатор клавишей <Escape>, или запустим его заново и установим «бит b4 = 1». При **View** > **Display** после <F1> (когда в окне имитатора изображается клавиатура терминала) нажмём <S>; в информационном окошке будут видны номера сегментов, подключенных к страницам P3, P2, P1, P0:



На пульте машины (т. е. при **View** > **Machine**) нажатием <V> или <W> перейдём в режим «Ввод», и наберём простенькую программку, всего семь байт, для подключения сегментов с номерами 03, 02, 01, 00:

Номер шага	Байт-коды команд	Мнемокод и содержание операций
00000	1304 0302	MOV #0302, S04; байт 03 02 заносится в регистр S_4
00002	1305 0100	MOV #0100, S05; байт 01 00 заносится в регистр S_5; S_4 и S_5 это R_{10}.
00004	0412 1506	ONSEGM ; подключение сегментов с номерами из тетрад R_{10}.
00006	0512	END ; конец программы

(Набирать надо каждую цифру байта, ввод каждого байта завершается обязательным нажатием <Enter>. Содержимое ОЗУ в направлении б*о*льших адресов просматривается нажатием <Enter> без ввода цифр, просмотр байтов с меньшими адресами – нажатием <Backspace>.)

Переходим в режим «Работа» нажатием <R>. Нажатием <C> устанавливаем программный счётчик в начало программы (шаг 00000 на странице P0), и запускаем программу нажатием <S>. Подключаются новые сегменты, и выполнение программы не может завершиться без индикации ошибки, т. к. в подключившемся к странице P0 сегменте 00 программы нет: там пока только «мусор» в виде байтов 1515. Тем не менее, задуманное подключение сегментов произошло. Нажимаем <C>, чтобы машина остановилась.

Поскольку к странице P3, в конце которой находится служебная зона ОЗУ, тоже подключился новый сегмент, заполненный мусором 1515, вручную обнулим двухбайтовый регистр BP. Для этого нажимаем <V>, клавишей <Backspace> или набором номера шага –00095 после нажатия <N> устанавливаем адрес в ОЗУ 0715 1000, и вводим друг за другом два нулевых байта: 0000 <Enter> и 0000 <Enter>. Заодно можно инициализировать и следующую пару байтовых ячеек в ОЗУ, введя в них друг за другом байты 0004 и 0000; это регистр BD. Указатель стека SP можно не менять, т. к. он инициализировался сам при нажатии <C>. (Для понимания того, что мы видим на табло машины в режиме «Ввод», следует помнить, что верхнее табло показывает текущее значение программного счётчика = 16-ричный адрес в регистре PC, нижнее табло показывает пятизначный десятичный «номер шага» для этого адреса, и байт в ячейке с этим адресом. Значения PC, BP и «номер шага» всегда связаны формулой: PC = «номер шага» + BP.)

Теперь заполним начало сегмента 00 байт-кодами какой-нибудь программки. Например, «загрузим с МЛ» программу вычисления cos(X), текст которой приведён на рис. 11 в «Инструкции по эксплуатации 3.857.100 ИЭ» для Д3-28, стр. 97:

```
00000  04 04  00 03   MOV X,003
00002  07 13          QRT
00003  07 11          NEG X
00004  04 04  00 01   MOV X,001
00006  07 01          DIG 1
00007  04 04  00 00   MOV X,000
00009  04 04  00 02   MOV X,002
00011  04 13  11 00   MOV X,R00
00013  06 04          MOV X,Y
00014  04 13  04 00   MOVD R00,X
00016  04 03  00 02   DIV X,002
00018  10 00  01 00   ADD #01,R00
00020  04 13  04 00   MOVD R00,X
00022  04 03  00 02   DIV X,002
00024  10 00  01 00   ADD #01,R00
00026  04 05  00 01   MOV 001,X
00028  04 02  00 02   MUL X,002
00030  04 14  00 04   MOV Y,004
00032  04 05  00 02   MOV 002,X
00034  06 00          ADD X,Y
00035  07 14          RES
00036  06 00          ADD X,Y
00037  04 05  00 04   MOV 004,X
00039  05 09          BEQ Y,X 00042
00040  14 02  01 11   BR 00014
00042  04 05  00 03   MOV 003,X
00044  08 03          COS
00045  05 15          STOP
```

Рис.11. Пример вычисления

$$\cos X = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \frac{X^8}{8!} - \frac{X^{10}}{10!} + \ldots$$

Эта программка, дополненная на шаге 00046 кодом 0512, есть в папке [txt] в файле cos_ris-11_KP-386_N-46.txt. Открываем его и загружаем на нулевой начальный адрес нажатиями ПК-клавиш <R> (это переход в режим «Работа»), <C>, <L>.

«Вынимаем кассету» с этой программкой и «ставим кассету» с программой МАШЯЗ – открываем в имитаторе файл mashyaz___KP-108069_N-8130.txt. Будем загружать его на страницу P2, т. е. в сегмент 02. Для этого нажимаем <V> или <W> (переход в режим «Ввод»), нажимаем <N> и набираем номер шага 16384; на этот начальный адрес мы собираемся загрузить МАШЯЗ. Переходим нажатием <R> в режим «Работа», и, не нажимая <C> (чтобы не сбросить в ноль начальный адрес), загружаем МАШЯЗ командой <L>. И опять-таки без нажатия <C> запускаем МАШЯЗ командой <S>. Переходим к **View** > **Display**, сбрасываем в ноль служебную строку дисплея клавишами <F8>, <F9>, <F8>, и включаем ДУП-ЛИН клавишей <F5>. МАШЯЗ выводит своё вежливое приглашение. Можем ввести директиву <D>, <Enter>, чтобы иметь на экране справку о директивах.

В регистре T_1 пока ещё мусор 1515, так что если дать директиву «!!» для просмотра номеров сегментов средствами МАШЯЗа, то увидим FFFF. Попробуем занести в T_1 правильные номера сегментов 03, 02, 01, 00 директивой МАШЯЗа «Адреса». Для этого клавишей <Ctrl> включим режим ЛАТ, и пользуясь клавишей <Shift> для переключений ВР / НР, наберём строку

¤7F21:32 и нажмём <Enter>.

Машина выводит указанный нами 16-ричный адрес в hex-нотации 7F21 первого байта регистра T_1, затем после косой черты она выводит этот же адрес в десятичной форме, затем старое значение байта в ячейке по этому адресу, две звёздочки, и новое значение байта; и ждёт дальнейшей команды. В той же строке наберём:

¤7F31:10 и нажмём <Enter>. Т. е. просим занести байт 10 (это hex-нотация) во второй байт регистра T_1, имеющий на 16 десятичных единиц больший адрес, чем адрес первого байта.

Машина выполняет задание и опять ждёт дальнейших команд. Чтобы выйти из режима «Адреса», нажимаем точку и <Enter>. После этого машина готова принять от нас новую директиву. Даём директиву «!!» и убеждаемся, что МАШЯЗ запомнил номера подключенных сегментов правильно, и может правильно показывать их привязку к страницам ОЗУ. Вот так выглядит всё это упражнение:

```
(Д-СПРАВКА) ПРОШУ: ж7F21:32

7F21 /32545   FF**32 ж7F31:10

7F31 /32561   FF**10 .

(Д-СПРАВКА) ПРОШУ: !!
СТР 3210-СЕГ 3210
(Д-СПРАВКА) ПРОШУ: _
```

Проверим работу директивы «Листинг»: попробуем вывести на экран и на печать коды программки «cos(X)», расположенной в адресах 0 – 46 (это десятичные значения), т. е. – на странице ОЗУ P0 в сегменте 00. Для этого клавишей <Home> включаем ТПУ, набираем: <L> / 0, / 50 <CapsLock>, <P>, <CapsLock>, <D>, 0, <Enter>. Косая черта указывает, что адрес мы задаём в десятичной форме. Машина начинает выводить на экран листинг и заодно дизассемблинг (приостанавливаясь после примерно 20 строк; для продолжения надо нажимать <Enter>). При этом МАШЯЗ изображает байт-коды программки в hex-виде. Вот как выглядит начало этого листинга:

```
(Д-СПРАВКА) ПРОШУ: Л/0,/50Д0

0000 /00000   4403   MOV X,(10*00+03)
0002 /00002   7D     QRT
0003 /00003   7B     NEG X
0004 /00004   4401   MOV X,(10*00+01)
0006 /00006   71     DIG 01
0007 /00007   4400   MOV X,(10*00+00)
0009 /00009   4402   MOV X,(10*00+02)
000B /00011   4DB0   MOV X,R00
000D /00013   64     MOV X,Y
000E /00014   4D40   MOVD R00,X
0010 /00016   4302   DIV X,(10*00+02)_
```

Одновременно в папке [txt] образуется txt-файл, имитирующий «ТПУ-распечатку» этого же самого листинга (его интересно сравнить с листингом на приведённом на стр. 76 фото из «Инструкции по эксплуатации»):

```
0000 /00000   4403   MOV X,(10*00+03)
0002 /00002   7D     QRT
0003 /00003   7B     NEG X
0004 /00004   4401   MOV X,(10*00+01)
0006 /00006   71     DIG 01
0007 /00007   4400   MOV X,(10*00+00)
0009 /00009   4402   MOV X,(10*00+02)
000B /00011   4DB0   MOV X,R00
000D /00013   64     MOV X,Y
000E /00014   4D40   MOVD R00,X
0010 /00016   4302   DIV X,(10*00+02)
0012 /00018   A010   ADD #01,R00
```

```
0014 /00020   4D40   MOVD R00,X
0016 /00022   4302   DIV X,(10*00+02)
0018 /00024   A010   ADD #01,R00
001A /00026   4501   MOV (10*00+01),X
001C /00028   4202   MUL X,(10*00+02)
001E /00030   4E04   MOV Y,(10*00+04)
0020 /00032   4502   MOV (10*00+02),X
0022 /00034   60     ADD X,Y
0023 /00035   7E     RES
0024 /00036   60     ADD X,Y
0025 /00037   4504   MOV (10*00+04),X
0027 /00039   59     BEQ Y,X .+003 ;НА 002A
0028 /00040   E21B   BR .-026 ;НА 000E
002A /00042   4503   MOV (10*00+03),X
002C /00044   83     COS
002D /00045   5F     STOP
002E /00046   5C     END
002F /00047   FFFF   OUTARV FF
```

Закончив вывод этого листинга, МАШЯЗ ждёт новых директив. Командой ! 0, /12 <Enter> подключаем к странице P0 сегмент 12 (это десятичный номер), затем проверяем успешность подключения, и просим вывести на экран листинг первых десяти байт в этом сегменте. Как и должно быть, там сохранились байт-коды той самой первой программки, которую мы набирали для смены сегментов, установленных по умолчанию при включении машины:

```
(Д-СПРАВКА) ПРОШУ: !0,12
?N СЕГМЕНТА? ?НЕ ПОНИМАЮ ВАС!

(Д-СПРАВКА) ПРОШУ: !0,/12

(Д-СПРАВКА) ПРОШУ: !!
СТР 3210-СЕГ 321C
(Д-СПРАВКА) ПРОШУ: Л0,/10

0000 /00000   D432   MOV #32,S04
0002 /00002   D510   MOV #10,S05
0004 /00004   4CF6   ONSEGM
0006 /00006   5C     END
0007 /00007   FFFF   OUTARV FF
0009 /00009   FFFF   OUTARV FF
(Д-СПРАВКА) ПРОШУ: _
```

(При вводе номера сегмента «12» я сначала забыл ввести косую черту / , указывающую систему счисления для номера, и МАШЯЗ очень вежливо обратил на это моё внимание. Кстати, для исправления ошибочно введённых символов в строке с директивой до нажатия <Enter> можно использовать «Забой», т. е. нажимать ПК-клавишу <Delete>; этот удобный способ редактирования командной строки в МАШЯЗе работает хорошо.)

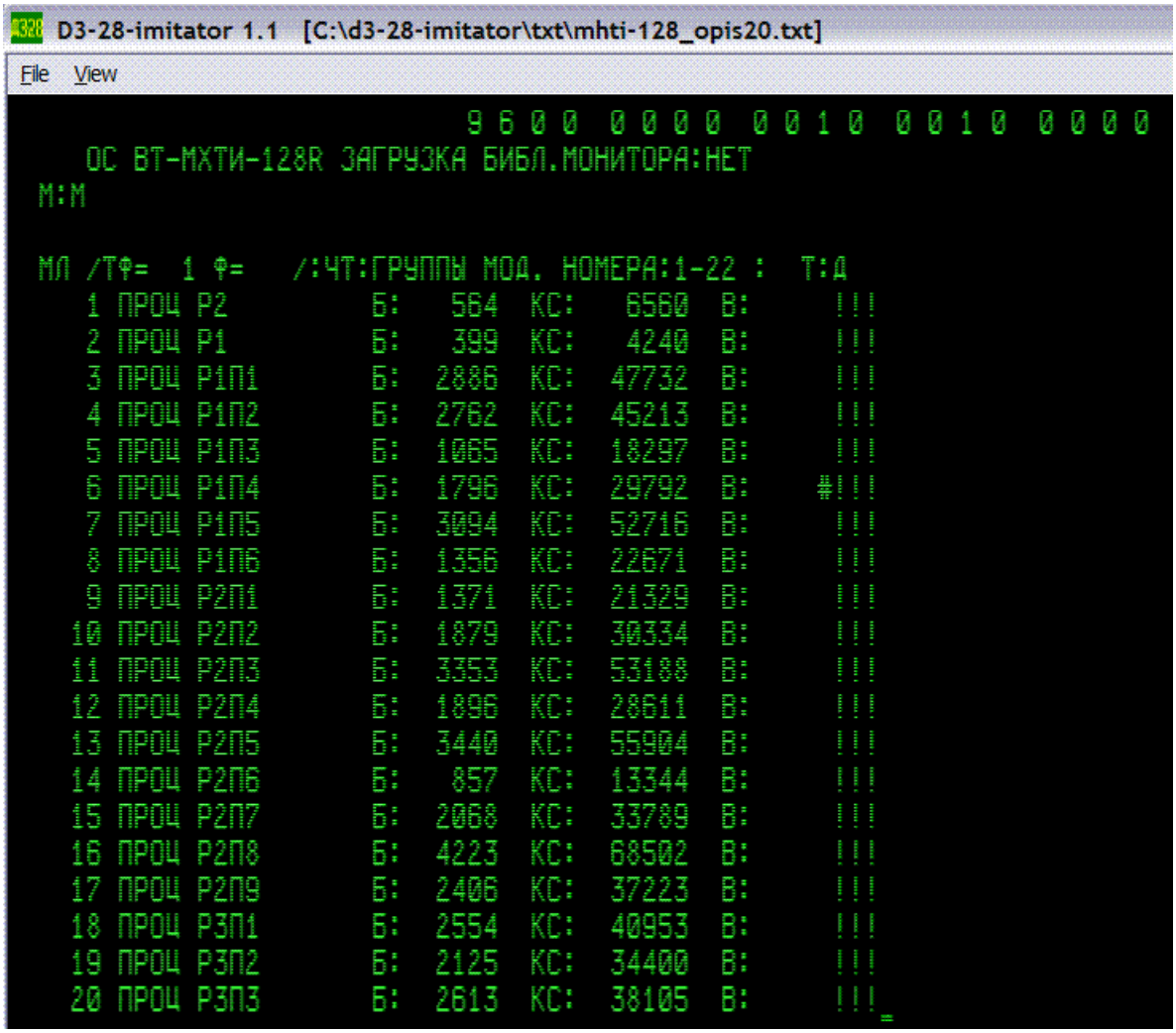
Таким образом, складывается впечатление (во всяком случае на данном этапе), что имитатор успешно работает с программой МАШЯЗ, использующей 128к-память Д3-28.

Управление 128к-памятью применяется также в Бейсике-План и в программах ОС ВТ-128-МХТИ. Кратко описать разные версии ОС ВТ-128-МХТИ затруднительно, поэтому они не включены в данный архив.

(Txt-файлы с байт-кодами версий ОС ВТ-128-МХТИ и документацию к ним можно найти на сайте d3-28.ru. Автор сайта уважаемый **Виталий К**. сохранил большую коллекцию кассет, выполнил оцифровку многих из них, и любезно предоставил свободный доступ ко всем раритетным материалам.)

Примечательное свойство ОС ВТ-128-МХТИ – возможность загрузки в ОЗУ Д3-28 сразу группы программ. Программы помещаются в «библиотеку монитора ОС» в ОЗУ машины. Для библиотеки в ОЗУ, согласно документации к этой ОС, отводится 7 сегментов (со 2-го по 8-й). т. е. 56 килобайт памяти. Ниже скриншоты иллюстрируют способность имитатора работать с такими системами управления 128к-памятью.

Вот начальный диалог и выполнение директивы загрузки с МЛ группы из 22 программных модулей в системе ВТ-МХТИ-128R:



Продолжение:

```
  21 ПРОЦ Р3П4        Б:  2164  КС:  35318  В:     !!!"

  22 ПРОЦ Р4П1        Б:  2827  КС:  45710  В:     !!!
М:М

МЛ /ТФ= 23 Ф= 22/:Н
М:М

МЛ /ТФ=  1 Ф= 22/:ЧТ:ГРУППЫ МОД. НОМЕРА:1-22 :  Т:Д
   1 ПРОЦ Р2          Б:   564  КС:   6560  В:     !!!
   2 ПРОЦ Р1          Б:   399  КС:   4240  В:     !!!
   3 ПРОЦ Р1П1        Б:  2886  КС:  47732  В:     !!!
   4 ПРОЦ Р1П2        Б:  2762  КС:  45213  В:     !!!
   5 ПРОЦ Р1П3        Б:  1065  КС:  18297  В:     !!!
   6 ПРОЦ Р1П4        Б:  1796  КС:  29792  В:    # КОНЕЦ БИБЛ: НБ=  2: КМ=  27:

М:_
```

Здесь виден результат опыта, позволяющего оценить объём «библиотеки монитора»: после того как 22 программки загрузились в память машины, была сделана перемотка назад и дана повторная директива загрузки тех же программок. При повторной загрузке система сумела загрузить полностью только 5 программок, а затем выдала сообщение КОНЕЦ БИБЛ. Размер каждой программки виден (это параметр Б); подсчёт суммарного количества байт в загрузившихся программках даёт в итоге 54 килобайта, что близко к ожидаемым 56 килобайтам.

Просмотр «каталога библиотеки» в ОЗУ показывает, что все читавшиеся с МЛ программки должным образом числятся в библиотеке монитора системы (середина списка на этом скриншоте для краткости пропущена):

```
М:БИБЛ:КАТАЛОГ
   1 ПРОЦ Р2          Б:   564  В:   1
   2 ПРОЦ Р1          Б:   399  В:   1
   3 ПРОЦ Р1П1        Б:  2886  В:   1
   4 ПРОЦ Р1П2        Б:  2762  В:   1
   5 ПРОЦ Р1П3        Б:  1065  В:   1
   6 ПРОЦ Р1П4        Б:  1796  В:   1#
   7 ПРОЦ Р1П5        Б:  3094  В:   1
   8 ПРОЦ Р1П6        Б:  1356  В:   1
   9 ПРОЦ Р2П1        Б:  1371  В:   1
```

. . .

И так далее: перечисляются те же имена программ, что и при загрузке с МЛ . . .

```
  21 ПРОЦ Р3П4        Б:  2164  В:   1
  22 ПРОЦ Р4П1        Б:  2827  В:   1 "

  23 ПРОЦ Р2          Б:   564  В:   2
  24 ПРОЦ Р1          Б:   399  В:   2
  25 ПРОЦ Р1П1        Б:  2886  В:   2
  26 ПРОЦ Р1П2        Б:  2762  В:   2
  27 ПРОЦ Р1П3        Б:  1065  В:   2
М:_
```

Повторно прочитанным с МЛ программкам система присвоила номер версии 2. Для полной уверенности в том, что информация действительно присутствует в ОЗУ, можно посмотреть листинг любой программки из «библиотеки монитора». Вот, например, листинг программки, присутствующей в библиотеке монитора под номером 23:

```
                          9 6 0 0  0 0 0 0  0 0 1 0  0 0 0 0  0 0 0 0  02 33 44
M:ЛИСТ:ГРУППЫ МОД. НОМЕРА:23 :  Т:Д  23 ПРОЦ. Р2
                                                  ВТ-МХТИ-128RL
          PROCEDURE P2
     1    /:
     2    /:                              РАЗДЕЛ II
     3    /:
     4    /:                ДИРЕКТИВЫ МОНИТОРА ОПЕРАЦИОННОЙ СИСТЕМЫ
     5    /:
     6    /:                     1...ОБЩАЯ ХАРАКТЕРИСТИКА
     7    /:                     2...БИБЛИОТЕКА МОНИТОРА
     8    /:                     3...ВВОД ИСХОДНЫХ ПРОГРАММ
     9    /:                     4...ЛИСТИНГ ПРОГРАММ
    10    /:                     5...СБОРКА ИСХОДНЫХ ПРОГРАММ
    11    /:                     6...КОМПИЛЯЦИЯ ИСХОДНЫХ ПРОГРАММ
    12    /:                     7...ИСПОЛНЕНИЕ ПРОГРАММ
    13    /:                     8...БИБЛИОТЕКА НА МАГНИТНОЙ ЛЕНТЕ
    14    /:                     9...ДОПОЛНИТЕЛЬНЫЕ ДИРЕКТИВЫ МОНИТОРА
    15    /:
           M:_
```

В этом конкретном примере программка оказалась «текстом» из комплекта документации к системам ВТ-МХТИ-128. Она имеет фортраноподобный вид, причём в каждой строке содержится только «оператор комментария» **/ :** . Так же обстоит дело и с остальными программками в данной «группе модулей». Оказывается, с такими текстами лучше работать (редактировать их и распечатывать) в другой версии системы ВТ-МХТИ-128; но это уже отдельная история.

7. Работа с машиной без дисплея

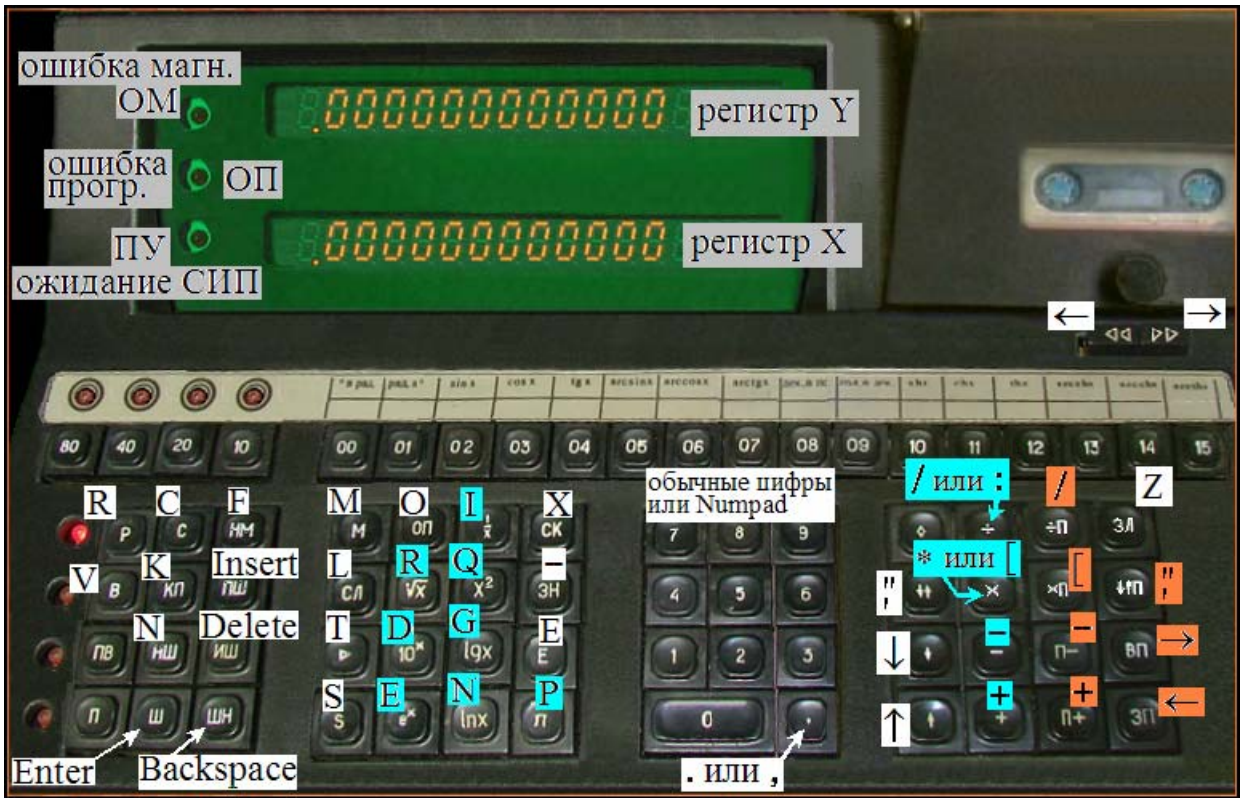
В этом разделе речь идёт об имитации собственно Д3-28, без дисплея. Конечно, наиболее интересна работа с дисплеем и с высокоуровневыми программами, такими как Бейсик, Фортран, Выстра, ОС ВТ-МХТИ, и др. Однако машина Д3-28 и сама по себе вызывает много ностальгических воспоминаний – о довольно интересном программировании в машинных кодах. Ниже описываются, в основном, правила соответствия между ПК-клавишами в имитаторе и кнопками пульта Д3-28. Для осознанной работы с имитатором по этим правилам необходимы знания о пульте машины и о её системе команд из документации, поставлявшейся с Д3-28:

«*Инструкция по эксплуатации 3.857.100 ИЭ*»,
«*Справочник программиста 3.857.100 Д3*».

(https://d3-28.ru/dokumentatsiya-d3-28/instruktsiya-po-ekspluatatsii-d3-28/
https://d3-28.ru/dokumentatsiya-d3-28/spravochnik-programmista/
http://retropc.org/Biblioteka_r_31.html#c254 см. там поиском в «Библиотеке».)

Машина сразу после запуска имитатора находится в режиме «Работа» – горит индикатор слева от кнопки Р; на табло X и Y светятся нули, означающие, что регистры X и Y содержат нулевые значения. Это ещё не «работа по программе», а лишь состояние «останова с индикацией». Режимы ПВ (печать программы при её вводе) и П (печать программы) в имитаторе отсутствуют; печать каждого байта при вводе представляется ненужной в имитаторе, а отсутствие печати программы командой П у нас компенсируется возможностью распечатки программы «на ТПУ» специальной программой-распечатчиком, загружаемой в любую свободную область ОЗУ (см. раздел 10).

Не имитируется и кнопка с ромбиком – печать числа из регистра X. Остальные кнопки имитируются ПК-клавишами; правда, имитация некоторых команд неполная и неточная. Вот схема соответствия кнопок пульта Д3-28 и ПК-клавиш:



На этой схеме надписи на зелёном табло – пояснения, где какой регистр, и что означают три индикатора, расположенные слева от табло регистров X и Y; эти надписи (у них серый фон) не относятся к кнопкам.

Белый цвет фона надписей на кнопках условно означает, что указанные в надписях ПК-клавиши работают <u>без прижимания других клавиш</u>. Напомню назначение тех из них, которые действуют в режиме «Работа»:

<R> / <V> – переключают режимы «Работа» / «Ввод». Вместо <V> годится <W>.
<$C_{лат}$> – сброс счётчика команд и масок прерываний, ресет указателя стека.

<$K_{лат}$> – команда вычисления контрольной суммы программы в ОЗУ машины.
<L> – считывание с ленты.
<Z> – запись на ленту.
< ← > – перемотка назад.
< → > – перемотка вперёд.

<S> – старт работы по программе (команда GO, её код: 0514).

< $T_{лат}$ > – начало команды «найти метку и запустить работу по программе с этой метки». В имитаторе табло X гаснет, а мы должны ввести второй байт команды – либо цифрами, либо одной из букв: <T>, <S>, <M>, <X> (это зависит от структуры программы; способ её запуска должен указываться в документации к программе). Номер метки в виде четырёх цифр в имитаторе вводится с постоянно нажатой клавишей <Ctrl>.

< F > – начало команды «найти (find) метку». В имитаторе табло X гаснет, а мы должны ввести второй байт команды: номер метки в виде четырёх цифр; он вводится с постоянно нажатой клавишей <Ctrl>. В имитаторе эта команда отличается от <T> тем, что она не запускает программу. Перейдя нажатием <V> или <W> в режим «Ввод», мы увидим команду в ОЗУ машины, следующую за найденной меткой; саму метку мы увидим, выполнив шаг назад клавишей <Backspace>. Такой поиск меток удобен для изучения или отладки сложных программ.

<$O_{лат}$> – гашение индикатора ОП. (Сброс, т. е. <C> тоже гасит индикаторы.)

<$X_{лат}$> – гашение табло X; может применяться перед вводом в X нового числа.

<$E_{лат}$> – сброс порядка у числа, находящегося в X; обычно применяется перед вводом порядка числа X.
< минус > – изменение знака на противоположный у числа в X, или изменение знака порядка у числа в X, если перед эти нажималась <E>.
Клавиши с цифрами – в режиме «Работа» служат для ввода числа в регистр X (или для ввода номеров меток и кодов команд, если нажата <Ctrl>).
Клавиши с точкой или с запятой – ввод десятичной точки для числа в X.
< ↑ > – засылка в регистр Y того же числа, которое имеется в X.
< ↓ > – засылка в регистр X того же числа, которое имеется в Y.
Клавиша с кавычками (“ и ‘) – обмен числами между регистрами X и Y.

<Enter> – останавливает «работу машины по программе», если она шла; тем самым машина переводится в состояние «останова с индикацией».

При этом счетчик команд указывает на ближайшую невыполненную команду. (Сброс <C> тоже останавливает работу машины по программе, но при этом счетчик команд сбрасывается к нулевому шагу).

Из режима «Работа» можно произвести запуск «работы по программе», если программа присутствует в ОЗУ машины. Чтобы в ОЗУ появилась программа, её надо туда загрузить «с магнитной ленты» (см. раздел 3) или ввести руками байт за байтом с пульта машины в режиме «Ввод»:

Режим «Ввод» включается нажатием <V> или <W>.
Нажатие <R> – переход в режим «Работа».

<u>В режиме «Ввод»</u> действие клавиш отличается от того, что было при «Работе»:

Пятизначный номер на X-табло это номер шага. Если содержимое регистра BP равно нулю, то номер шага равен содержимому регистра PC, т. е. программному счетчику; в общем же случае номер шага = PC – BP . Табло Y показывает двухбайтное значение PC в 16-ричной системе счисления; байты разбиты на тетрады; каждая тетрада принимает значения в диапазоне 00, …, 15.

Рядом с номером шага на X-табло высвечивается байт содержимого ОЗУ на этом шаге, т. е. содержимое байтовой ячейки ОЗУ машины с адресом, равным содержимому регистра PC (этот адрес выведен в 16-ричном виде на табло Y).

Нажатие <Enter> увеличивает на единицу номер шага, а нажатие <Backspace> уменьшает его на единицу. Таким путём можно байт за байтом просматривать содержимое ОЗУ машины и её регистров в служебной зоне ОЗУ. Описание служебной зоны ОЗУ см. в «*Справочнике программиста*» для Д3-28.

<N> – нажатием этой клавиши номер шага гасится, и машина ожидает <u>ввода пятизначного числа</u> – десятичного значения номера шага, на котором мы хотим посмотреть или изменить байт в ячейке ОЗУ. Мы должны нажать пять раз цифровые клавиши; тогда содержимое соответствующей ячейки ОЗУ изобразится рядом с введенным номером шага.

Если в режиме «Ввод» набирать цифры без предварительного нажатия <N>, то машина воспримет их как ввод нового байт-кода в данную ячейку ОЗУ. Чтобы отказаться от начатого, но незавершённого ввода байта, надо нажать <Enter> или <Backspace>. Чтобы завершить ввод байта, надо ввести именно четыре цифры (по две цифры в каждой из двух тетрад байта) и обязательно нажать <Enter>. Чтобы отказаться от завершения ввода после того, как уже введены четыре цифры (например, мы ошиблись при их вводе или передумали), надо нажать <Backspace>. В любом случае можно исправить ошибку при наборе повторным вводом нужного байта в ячейку с нужным адресом. Таким путём в имитаторе можно вручную ввести в ОЗУ машины программу, написанную «в машинных кодах».

Ввод байтов в ОЗУ с реального пульта реальной машины производится кнопками прямого кодирования: старшая тетрада каждого байта вводится в коде 8-4-2-1 кнопками, расположенными под четырьмя индикаторами, а младшая тетрада вводится одной из кнопок в ряду 00, 01, …, 15. В имитаторе процедура ввода байтов в ОЗУ упрощена; кнопки прямого кодирования отсутствуют; допускается ввод знака «минус» для набора отрицательного номера шага (при не равном нулю ВР).

Для отладки небольших программ и для их модификации могут быть полезными операции «Исключить Шаг» и «Прибавить Шаг»; на ПК-клавиатуре в имитаторе им назначены клавиши <Delete> и <Insert>. При исключении шага байт-код на текущем шаге теряется, на его место переписывается байт из следующей ячейки, и т. д. Прибавление шага заключается в том, что байты в ОЗУ, начиная с текущего, переписываются в ячейки с б*о*льшим на единицу адресом, а на текущий шаг записывается байт-код команды GO, т. е. 0514. Затем можно его заменить любым желаемым байт-кодом, вводя новые четыре цифры. Следует помнить, что такая модификация не испортит работоспособность программы только в том случае, когда она не сбивает адреса ветвлений и переходов, имеющихся в программе.

Простейший пример ввода программы с пульта машины

Введите указанным выше путём с нулевого шага программку, содержащую всего 9 байт:

0700 0604 0701 0600 0412 0615 1402 0004 0512

Смысл этих команд прост:

0700 – ввод в регистр X числа 0,
0604 – пересылка содержимого X в Y,
0701 – ввод в регистр X числа 1,
0600 – сложение X+Y и засылка результата в Y,
0412 0615 – двухбайтовая команда «Пауза», для индикации X и Y на табло,
1402 0004 – двухбайтовая команда перехода к шагу «00003»,
т. е. к сложению X+Y и к засылке результата снова в Y.
0512 – команда END. Она необходима для того, чтобы можно было записать эту программку «на ленту». (На всякий случай в нашей папке [txt] эта программка уже есть – с именем y-test__KP-88_N-8.txt.)

Это упражнение показывает, как вводить программу с пульта машины. В данном примере запрограммирован простой счётчик: после каждой паузы с индикацией число в регистре Y будет увеличиваться на единицу. Чтобы запустить программку, надо перейти нажатием <R> в режим «Работа», нажать <C> и затем <S>. Для останова можно нажать <Enter>; машина закончит выполнять текущую команду и прекратит счёт. После останова мы сможем клавишей <V> или <W> перейти в режим «Ввод» (пока велась работа по программе, такой переход был заблокирован; проверка этого факта служит способом узнать в имитаторе, находится машина в «программном режиме» или уже остановилась, если режим её работы неочевиден).

В качестве примера модификации программы установим номер шага равным 3, выполним «Прибавить Шаг», и введём вместо 0514 код 0700. Тем самым мы запрограммировали занесение в регистр X числа 10. Запустив снова программку, мы увидим, что теперь счёт в регистре Y идёт с шагом 10.

Подобным же образом легко модифицировать программку так, чтобы в ней X и Y перемножались: для этого байт-код 0600 надо заменить на 0602; чтобы получался не ноль, надо изменить и команды засылки чисел в X. Если запрограммировать засылку в X числа с порядком 10 или больше, то при выполнении программы будет быстро достигнуто «превышение разрядной сетки», так как в Д3-28 порядок числа задаётся всего двумя десятичными разрядами, и, значит, не может превышать 99. При превышении разрядной сетки имитатор может глючить и выводить на табло что-то бестолковое (потому что в имитаторе имитация команд Д3-28 для простоты сделана во многих местах без проверок корректности действий пользователя, и не в соответствии с алгоритмами «арифметики Д3-28»).

Набирать программки бывает удобнее в текстовом редакторе, например, в «Блокноте», и сохранять результат как txt-файл для последующей загрузки в имитатор через его «магнитофон». Байт-коды следует набирать столбиком (по одному байту в каждой строке), и после каждого байта нажимать <Enter>, – такой формат txt-файла предусмотрен в имитаторе для «чтения с ленты».

Примером простой демо-программки служит также 7-11-2017_tablo-XY-test__KP-637_N-36.txt. Ещё один пример «демо» на табло Д3-28 – первая программка в файле mix_25-09-2010.txt (для её работы на реальной машине Д3-28, вероятно, придётся вручную подобрать число в регистре R_{10}, определяющее время ожидания СИП в машинной команде OUTOWC и тем самым регулирующее частоту смены цифр на табло). Обе программки запускаются (после <C>) командой поиска метки с кодом 0514, т. е. клавишами <T>, <S>.

Учебный пример – программка cos_ris-11_KP-386_N-46.txt из «*Инструкции по эксплуатации*» (см. там стр. 96-97, рис.11). После загрузки (клавишами <C>, <L>) набираем в регистре X какое-нибудь число, например X=1.23456789, и нажимаем <S>. Результат: в регистре Y будет значение cos(X), вычисленное программкой по формуле ряда Тейлора, а в регистре X – значение cos(X), вычисленное по машинной команде с кодом 0808. В нашем примере это 0.329929069890; чтобы продолжать, надо нажать <C> и только после этого вводить новое X. В программке важную роль играет машинная команда RES – вычисление "остатка". Правда, в имитаторе эта команда пока ещё не сделана, она заменена простенькой заглушкой. Тем не менее, программка работает.

Ещё один учебный пример – q-eq_ris-9_KP-543_KP-7765_KP-34321.txt. В этом txt-файле приведены три версии программы вычисления корней квадратного уравнения. В конце файла напечатаны подробные пояснения. Первая версия набрана прямо по листингу из «*Инструкции по эксплуатации*», стр. 90-93, рис.9. Вторая и третья версии – результат трансляции в машинные коды программ с аналогичным алгоритмом вычисления и ввода-вывода без дисплейного терминала, составленных в системах программирования Выстра и Фортран-5М.

sip_sip-wyw-auto_sip-upr-auto__vystra-texts.txt – здесь первые 3 программки работают без дисплея. Это как бы уже не учебный, а серьёзный пример, – программки предназначены для выявления адресов подключенных к Д3-28 периферийных устройств, способных обмениваться информацией с Д3-28 (в том числе и через УСО АЦСКС-1024-001). Подробные пояснения есть в конце этого txt-файла.

В папке [txt] есть также «стандартные» (входившие в комплект поставки машины Д3-28) программы, работающие без дисплейного терминала. Следуя нынешней моде, их можно назвать «легендарными»:

statistika__KP-7074__N-806.txt – пакет подпрограмм «Статистика». В конце txt-файла, после байт-кодов напечатаны пояснения о загрузке «Статистики» и о работе с подпрограммами пакета в имитаторе. Текст подпрограмм, а также полный набор правил и примеров работы с ними см. в брошюрке «*Руководство оператора 3.857.100 Д12*» из комплекта документации к машине Д3-28.

D3-28_test_028__KP-60000_N-5999.txt – старый тест системы команд,
D3-28_test_017__KP-5895_N-527.txt – тест НМЛ,
028-test_KP-73370_N-8921.txt – новый тест ОЗУ и системы команд.

О работе этих «стандартных» тестов рассказано в «*Инструкции по эксплуатации*» для Д3-28. См. также пояснения ниже – в разделе 13.

Выполнение одиночных команд с пульта машины

Машина Д3-28 допускает выполнение команд, подаваемых с пульта кнопками прямого кодирования: старшая тетрада байта вводится в коде 8-4-2-1 кнопками, расположенными под четырьмя индикаторами, а младшая тетрада вводится одной из кнопок в ряду 00, 01, …, 15.

В имитаторе тоже возможна подобная подача команд, хотя на ПК-клаве нет кнопок прямого кодирования. В имитаторе коды команд для непосредственного исполнения можно набирать цифровыми ПК-клавишами, удерживая постоянно нажатой клавишу <Ctrl>. Отпускание <Ctrl> до окончания набора команды имитатор воспринимает как отказ от команды. Если команда однобайтовая, то после набора её байта перед набором кода следующей команды можно отпустить <Ctrl>. Если команда двухбайтовая, и мы желаем её выполнить, а не отказываться, то отпустить <Ctrl> можно будет только после набора двух байтов.

Индикаторы 8-4-2-1 на изображении пульта машины отображают набор старшей тетрады байта, а при наборе второй тетрады они гаснут – так в имитаторе индицируется факт подачи команды (на реальной же ЭВМ Д3-28 они продолжают гореть; программист на реальной Д3-28 ощущает подачу команды своими пальцами, поскольку реально нажимает одну из кнопок в ряду 00, 01, …, 15).

Пример: удерживая <Ctrl> прижатой, наберём цифры 0701; это код команды DIG 1. Результат будет такой же, как при вводе в регистр X числа 1 клавишей <1> без <Ctrl>. Затем наберём, удерживая <Ctrl> прижатой, цифры 0700; результат такой же, как если бы мы продолжили ввод числа в X нажатием <0> без <Ctrl>, т. е. на X-табло высветится число 10, если до этого там были одни нули.

Есть нюанс: перейдя в режим «Ввод», мы увидим, что в имитаторе каждая подача команд с пульта машины сопровождалась увеличением номера шага (увеличением PC). В реальности такого увеличения PC не должно быть; но я не стал изменять программу имитатора для устранения этого недостатка имитации. В руководствах по Д3-28 вообще не рекомендуется подача с пульта команд, следствием которых должны быть переходы или ветвления, т. е. вполне определённые изменения PC. Так что, одиночные команды вообще не слишком-то полезны на практике. Примерно из этих же соображений я поленился сделать имитацию пошагового исполнения программы (как в реальной Д3-28 кнопкой Ш).

Некоторые одиночные команды в реальной Д3-28 могут быть исполнены нажатием специальных кнопок в правой части пульта, таких как xП, П-, П+, и др. – это команды арифметических действий с участием десятичных ячеек памяти (см. руководства по работе с Д3-28). В имитаторе им соответствуют ПК-клавиши, которые на приведённой выше схеме (см. стр. 82) показаны на красном фоне – их следует нажимать, удерживая постоянно нажатой клавишу <Ctrl>.

ПК-клавиши, показанные на синем фоне (см. стр. 82), следует нажимать, удерживая постоянно нажатой клавишу <Shift>, – тогда им соответствуют математические функции, обозначенные на кнопках пульта Д3-28.

Пример: введём в X цифровой клавишей <1> число 1, клавишей < ↑ > скопируем его в Y, введём в X клавишей <2> число 2 (всё это делается без прижимания других клавиш) и, наконец, удерживая прижатой <Shift>, нажмём клавишу <+>. На табло регистра Y появится результат сложения чисел из регистров X и Y.

Ещё пример: нажатием <$P_{лат}$> с постоянно прижатой <Shift> мы вводим в регистр X число π. Аналогичным образом, если мы введём в X число 1 и нажмём с постоянно прижатой <Shift> клавишу <E>, то получим в регистре X результат вычисления $e^1 = e$; это число Непера, т. е. основание натуральных логарифмов: 2,71828182846.

Такие вычисления можно комбинировать с кодами команд математических операций, которые подаются с постоянно прижатой <Ctrl>. Например, после того, как указанным выше способом в X было введено число *e*, наберём цифровыми клавишами с постоянно прижатой <Ctrl> команду 0802. Тогда на табло X появится результат вычисления sin(X); в данном примере это $\sin(e) = 0{,}410781290502$.

В имитаторе отсутствует возможность выполнения команды RES (вместо неё в имитаторе действует некая подделка); в реальной Д3-28 эта команда выводит на табло «остаток» арифметического вычисления и тем самым увеличивает точность вычисления свыше 12 десятичных разрядов. В имитаторе вообще нет точной эмуляции алгоритмов математики Д3-28; поэтому при исполнении примеров из руководств возможны отличия (в одном или нескольких последних разрядах) от результатов, указанных в документации к Д3-28.

8. Регулировка быстродействия имитатора

Имитатор не воспроизводит динамику работы реальной машины Д3-28. Хотя на современном ПК вычисления выполняются во много раз быстрее, чем в Д3-28, в имитаторе вывод текстов на дисплей (и, в частности, вывод символов оператором PRINTTAB при построении графиков в Бейсике) происходит заметно медленнее, чем следовало бы. Скорость имитации такого вывода зависит от быстродействия конкретного ПК, от работы его видеосистемы.

Регулировать быстродействие имитатора в сторону увеличения невозможно. А замедлить работу имитатора можно – введением задержки между обращениями к интерпретатору машинных команд.

Такая задержка может потребоваться, в основном, в двух случаях:

1) если ваш ПК очень шустрый, так что, например, «полёт точки-ракеты» в тест-игре 2WAR происходит слишком быстро;

2) если исследуется программа для Д3-28, в которой нет частого вывода информации на дисплей или на табло. Рассмотрим простой пример. В разобранной на стр. 85 программке «y-test» заменим двухбайтовую команду 0412 0615, выводившую «счётчик Y» на табло Y, двумя командами 0514 («пустыми» командами GO, они просто передают управление дальнейшим командам в программе):

0700 0604 0701 0600 0514 0514 1402 0004 0512

Откроем в Windows «Диспетчер задач» на вкладке «Процессы» (или «Монитор ресурсов», или какую-то аналогичную программу, способную показывать процент занятости центрального процессора ПК). Запустим имитатор Д3-28. Будет видно, что он практически не потребляет ресурсов компьютера, пока в нём не запущена программка. Однако, если мы запустим указанную выше модифицированную программку «y-test», то увидим, что нагрузка процессора возрастает почти до максимально возможной; например, у меня на стареньком ПК с Windows XP занятость процессора подскакивает до 90% и более! Процессор без передышки крутит цикл с нашей программкой, через некоторое время температура его заметно повышается, вентилятор начинает сильнее шуметь. (Остановив машину нажатием <Enter>, увидим, какое большое число циклов успел накрутить наш счётчик за время работы программки без вывода Y на табло).

Желательно чередовать работу процессора ПК над вычислениями в имитаторе с более спокойной работой в операционной системе ПК. Имитатор стремится обеспечить такое чередование автоматически: он отдаёт несколько миллисекунд из своих «квантов времени» операционной системе после каждого вывода «кадра» – вывода изображения машины, клавиатуры или экрана дисплея на монитор ПК. Но если обновление кадра не предусмотрено в программке (или происходит редко), то процессор работает без передышки, как в рассмотренном выше примере.

В подобных ситуациях можно применить задержку вычислений (Delay), которая командой Sleep(1) будет отдавать операционной системе 1 мс времени каждый раз после заданного количества циклов работы имитатора. В имитаторе это количество циклов задаётся параметром с именем dwN_delay. (А если он равен нулю, то это означает «задержки нет».)

При запуске имитатора параметр dwN_delay равен нулю. При ресете сохраняется его текущее значение. Чтобы его изменить, надо в режиме **View** > **Display** нажать <F1> и затем <D>. Появится окошко с названием-вопросом «Change delay?» и с информацией о текущем значении dwN_delay (также там будет информация о работе GetTickCount на данном ПК – сколько раз выполняется Sleep(10) на «интервале времени 100 мс», заданном через команду GetTickCount; на эту информацию можно не обращать внимания). Чтобы ввести новое значение задержки, нажимаем «Да»; в появившемся окне вводим желаемое значение задержки (или 0, если хотим выключить задержку); на повторный вопрос «Change delay?» отвечаем «Нет», чтобы перейти ко второй части диалога и затем там выйти из состояния настройки.

Во второй части диалога о задержке спрашивается, надо ли изменить имитацию команды PAUSER. Дело в том, что в программах эта команда обычно используется для приостановки при управлении электромагнитом и мотором НМЛ, чтобы дать время инерционной механике перейти из одного состояния в другое. Длительность такой паузы определяется числом в регистре R_{10}. В имитаторе подобные паузы увеличивали бы время считывания (и записи) блоков с ленты, поэтому «по умолчанию» такая зависимость паузы от содержимого R_{10} отключена (PAUSER выполняется просто как Sleep(4)). Но бывают и другие программы – в них PAUSER используется для предоставления времени пользователю на выполнение определённых действий; в таких случаях в имитаторе надо включать зависимость паузы от R_{10}, а иначе мы не успеем выполнить требуемые действия (пример см. в разделе 2: тест «061» в блоке 4 предписывает пользователю выполнить ряд нажатий клавиш за 5 секунд).

В реальной машине Д3-28 один такт длится одну-две микросекунды, исполнение одной машинной команды занимает несколько тактов. Поэтому для оценки (очень грубой) значения dwN_delay, хоть как-то приближающего динамику имитатора к динамике реальной Д3-28, можно считать, что за 1 мс имитатор должен выполнять порядка 1000 циклов. Т. е. можно для начала задать dwN_delay = 1000, а затем корректировать это значение. При значении 1000 у меня на стареньком ПК с Windows XP «занятость процессора» в рассмотренном выше

модифицированном «у-тесте» падает уже до 1%. При значении dwN_delay порядка 100 и менее имитатор может так сильно замедлиться, что он будет казаться «зависающим» в имитации работы с Бейсиком. (Но для заметного замедления «бегущей строки 2018» в первой программке из mix_25-09-2019.txt этого может быть ещё недостаточно; потребуется dwN_delay порядка 10.) При увеличении dwN_delay свыше нескольких тысяч наличие задержки будет, наоборот, всё менее заметным.

В ноутбуке с Windows 7 монитор ресурсов показывает «использование процессора» и ещё «частоту процессора» в процентах от максимальной. Как выяснилось, во время вывода имитатором текстов на дисплей процент использования процессора невелик, но «частота» может подниматься до 90% и более, и при этом температура тоже повышается. Однако есть возможность ограничить повышение «частоты» и тем самым препятствовать нагреву системы при работе всех программ, не только имитатора. Для этого открываем: Панель управления > Электропитание > Настройка плана электропитания > Изменить дополнительные параметры питания > Управление питанием процессора > Максимальное состояние процессора; и вместо указанных там 100% задаём, например, 60% (при питании от сети и от батареи).

В имитаторе можно включить индикацию частоты вывода «кадров» – параметр FPS (frames per second). Он оценивается в проге очень грубо – с помощью функции GetTickCount. Но всё-таки такая оценка помогает заметить, как та или иная программка распоряжается командами вывода информации. Если вывод очень редкий, то кадр будет обновляться, в основном, с частотой мигания курсора дисплея (примерно 5 раз в секунду), и значение FPS получится близким к 005. Если вывод частый, то в имитаторе сработает ограничение частоты обновления кадра (предусмотренное, чтобы избежать перегрева видеочипа), так что значение FPS, скорее всего, не превысит 060 … 070. Значение FPS изображается белым шрифтом в начале служебной строки дисплея. Чтобы увидеть FPS или затем скрыть его, надо в режиме **View** > **Display** нажать <F1>, <F>, и ответить на вопрос «Show FPS?».

9. Об улучшениях, о недостатках имитатора, о тестировании

В октябре 2019 года уважаемая Наталия **xlat** – участница форума «Полигон Призраков», – предложила помощь в изучении ЭВМ «Электроника Д3-28» и выполнила более сорока опытов на реальной машине Д3-28, выявивших особенности ряда команд этой ЭВМ! PDF с отчётом об этих опытах есть здесь:
https://yadi.sk/i/qDCS9FMF74RQUA .

В феврале 2021 **xlat & maddev** (Наталия и Роман) предъявили свою разработку онлайнового эмулятора Д3-28: https://xlat8086.com/d3-28/ ;
подробное сообщение на "Полигоне Призраков" здесь: ссылка.

Результаты опытов на реальной машине, а также результаты ставших доступными экспериментов на онлайновом эмуляторе Д3-28, я теперь учитываю при обнаружении и устранении ошибок в своём имитаторе.

В данной версии имитатора (16.02.2023) исправлена остававшаяся долгое время незамеченной ошибка имитации команды CMD (0413 02 R_j): там недосчитывался на 1 адрес перехода при исполнении через CMD 1-байтовой команды ветвления (например 0508 BLT Y,X при Y>=X) в том случае, когда ветвление должно идти не на +3 шага, а на +1. (Для этого случая в прежних версиях имитатора ошибочно не учитывалось, что

счетчик команд заранее увеличивается на 1 при исполнении второго байта команды CMD. Так как раньше этот случай в программах не встречался, ошибка не проявлялась. Она обнаружилась лишь недавно – в результате раскодирования и попыток запуска программки форматирования текстов в "Бейсике-План".)

Какие ещё обнаруживались ошибки имитации, как они проявлялись? Вот, например, неожиданно вскрывшийся «глюк» в версиях имитатора до 2020 года: в Бейсике-157107 при выводе на экран оператором PRINT отрицательных чисел не в формате «по умолчанию», а, например, в формате !2.0!, печаталась ерунда типа такой: –49 вместо –1. Это пример трудно обнаружимой ошибки, так как имитатор сами-то вычисления делал верно с любыми числами, положительными и отрицательными, и вывод результатов в формате «по умолчанию» (т. е. !F1.9!) для любых чисел тоже выполнялся верно. Ошибка обнаружилась чисто случайно, и затем исправилась сама собой после переделки имитации ряда команд с учётом информации, полученной из указанных выше опытов **xlat** на реальной Д3-28.

К сожалению, в документации Д3-28 команды описаны бегло, примеры приведены лишь самые простые, поэтому с самого начала при попытках имитации команд встретилось много неясностей, особенно – в различных условиях учёта бита знака двухбайтовых регистров, необходимости нормализации десятичных регистров, в действиях с «нечисловым» их содержимым. Обусловленные такими неясностями «глюки» обнаруживались не сразу, случайно.

Вот ещё несколько примеров (более полное перечисление «глюков» с хронологией попыток улучшения имитатора есть в файле info_*дата*_.txt в папке [Debug] в zip-архиве с «сорцами» имитатора src_d3-28_imitator_*дата*.zip). В версиях имитатора до июля 2019 г. программа МАШЯЗ при листинге выводила в номере шага 00008 вместо восьмёрки ноль. От этого «бага» удалось избавиться, изменив в имитаторе обработку «отрицательного нуля» в регистрах R. С учётом различия «–0» и «+0» в команде условного перехода 1410 B2A2 устранилась ошибка имитации оператора IF в Фортранах 85 и 5М, которая тоже относилась к «трудным»: локализовать её причину долго не удавалось, так как эта ошибка появлялась не при каждом обращении к IF. Изменив обработку бита знака в команде вычитания константы из регистров R, внезапно удалось запустить транслятор ВТ-МХТИ-128С, который до этого в имитаторе не работал и поэтому казался мне неисправным. Каждый раз после исправления очередной ошибки возникает надежда на то, что наш имитатор, возможно, уже достиг некоего совершенства – хотя бы в приближённой симуляции Д3-28. (Анализ принципиальной схемы машины и прошивок ПЗУ с построением в итоге полноценного *эмулятора* – вот это действительно надёжное решение; но такое сложное дело выше моих возможностей.)

Можно сказать заранее, на чём имитатор обязательно «споткнётся». Вот четыре типа недостатков имитатора:

1. В имитаторе не реализованы строго те же алгоритмы «арифметики», которые заложены в микрокоде в ПЗУ Д3-28. В доках сказано, что микрокод использует ячейки ОЗУ Д3-28, но не объясняется детально, как именно. Если программист, знающий такие детали, применит их в программе, то имитатор с такой программой работать не сможет. (Но это, наверное, экзотика. На практике в естественных задачах такие программы, в которых играли бы роль недокументированные данные из служебных ячеек, мне не встречались.)

Приведу искусственный пример этого типа. У меня сохранились результаты теста с выводом недесятичного содержимого оператором PRINT в Бейсике-157107 на реальной машине Д3-28 1980-х годов. В два первых байта мантиссы десятичной переменной (скажем, A с начальным значением 0) командой CMD засылались недопустимые или допустимые значения. Затем A выводилась на экран оператором PRINT !F1.9! A. Вот пример с недопустимым значением 10 первой тетрады первого байта (оно на 1 больше допустимого значения 09):

первый байт = **10 08**, второй байт = **07 06**

В этом случае в реальной Д3-28 Бейсик вывел на экран число 1.087600000. Оказывается, с этим примером теперешний имитатор справляется успешно – выводит на экран такое же число 1.087600000 (в версиях до 2020 года имитатор во всех примерах с некорректными тетрадами выдавал ответы, отличающиеся от ответов Бейсика в реальной Д3-28). Но вот пример, в котором некорректные значения имеют обе тетрады первого байта:

первый байт = **1514**, второй байт = **0706**

В этом случае реальная Д3-28 с Бейсиком-157107 вывела на экран фрязинского дисплея комбинацию символов 0.>76000000E–01, содержащую нечисловой знак > . Но имитатор в этом примере выдаёт число 1.476000000E–01.

Дело в том, что Бейсик в реальной Д3-28 выполняет с переменными действия по зашитым в ПЗУ Д3-28 алгоритмам, которые и превращают байт «1514» в «0.>». В имитаторе же сначала из тетрад десятичной переменной формируется число в компьютерном формате double с плавающей запятой, затем с ним выполняются действия по алгоритмам математической библиотеки Windows, и получившееся число преобразуется обратно к 12-разрядному тетрадному формату десятичных переменных, свойственному Д3-28. Таким образом, имитатор всегда вычислит какое-то число, даже в случаях с некорректным содержимым тетрад десятичной переменной; выдать нечисловой ответ для десятичной переменной имитатор не может в принципе.

По аналогичной причине имитатор не выполняет должным образом команду RES (код 0714). В реальной Д3-28 операция RES выводит в регистр X остаток арифметических действий из служебных десятичных ячеек памяти, что при необходимости позволяет увеличить точность вычислений добавлением к результату ещё 12 разрядов. В имитаторе не реализовано такое же самое накопление остатка, как в реальной Д3-28, а просто в роли остатка берутся последние разряды числа в формате double, не попавшие в первые 12 разрядов результата. Такая «заглушка» вместо операции RES оказалась достаточной для работы имеющихся программ – Бейсика, Фортрана, Выстры и др.

Результаты имитаторных вычислений (по алгоритмам математической библиотеки Windows для чисел формата double) и вычислений на реальной Д3-28 (по алгоритмам из ПЗУ Д3-28) могут различаться в последних знаках 12-разрядной десятичной мантиссы. В прикладных программах для задач физики, где часто достаточной является точность лишь с тремя–четырьмя разрядами, такое различие не играет роли. Но оно препятствует нормальному прохождению тест-программ, в которых погрешность вычислений специальным образом накапливается и сравнивается с эталонным значением, зафиксированным в тест-программе; подробнее об этом сказано ниже, в разделе 13 «Приложение 3: дополнительные пояснения. Работа тест-программ в имитаторе».

2. Имитатор не может имитировать реальную динамику (тайминг) выполнения команд Д3-28; в том числе, имитатор не воспроизводит динамику работы Д3-28 с магнитофонной лентой. В нём не смогут правильно выполняться программы записи, чтения, перемотки, в которых используется инерционность ЛПМ или применяются команды обнаружения ракорда либо сигнала с магнитной ленты, поиск паузы с заданной длительностью.

К счастью, такие команды относятся к редко применяемым; Бейсик, Фортран и Выстра работают с МЛ без них, так что работа этих программ с МЛ имитируется успешно, как уже пояснялось в разделах 3 и 5.

Имитация работы с МЛ в ОС ВТ-МХТИ-128 – менее успешная. В описании к этой системе говорится, что после просмотра содержимого МЛ директивой «Каталог» система оказывается подготовленной к записи новой программы на МЛ, которая продолжит имеющуюся на МЛ библиотеку записей (как это было бы и в Фортране). Но в имитаторе с ОС ВТ-МХТИ-128 нельзя так записать продолжение библиотеки: имитатор произведёт запись некорректно – почти в самое начало «ленты» (txt-файла, имитирующего МЛ), на место за первым заголовочным файлом на этой «ленте». Насколько я сумел понять, дело здесь в том, что в реальности система ВТ-МХТИ-128, прочитав в процессе выполнения директивы «Каталог» запись-метку «Конец Библиотеки» (КБ), делает короткую по времени перемотку назад и затем исполняет команду загрузки LOAD X, чтобы, обнаружив код end в последней библиотечной программке, остановиться перед меткой КБ. Т. е. система ВТ-МХТИ-128 использует инерционность ЛПМ при выполнении машинной команды перемотки назад (REW). В имитаторе же указатель позиции в txt-файле, имитирующем МЛ, командой REW устанавливается сразу в начало txt-файла.

Эта конкретная проблема легко решается. В имитаторе с ОС ВТ-МХТИ-128 для подготовки к продолжению записей «в библиотеку на МЛ» можно считать с МЛ в ОЗУ (в «библиотеку Монитора») ту программку, которую мы желаем сохранить на МЛ как последнюю запись перед записью будущей новой программки; затем, если она в данном сеансе не нужна, её можно удалить из ОЗУ директивой «Библ. Чистка». Тогда новая запись на МЛ пойдёт вслед за ней.

С чтением и записью программок в режиме «Монитор Рабочих Программ» пока мне не удалось разобраться. Поэтому не исключаю, что такие операции в имитаторе с ОС ВТ-МХТИ-128 в режиме МРП не работают; однако, надо продолжить испытания, прежде чем делать окончательные выводы.

3. Имитатор не обладает способностью, свойственной реальной машине Д3-28 (с её многоконтактным разъёмом «Ввод-Вывод»), управлять любыми периферийными устройствами.

Имитируется только управление печатающим устройством «ТПУ 15ВВП80-002» и взаимодействие с «терминалом 15ИЭ-00-013» через «контроллер дисплея ПЕЛ2.240.001». Поэтому вывод информации на ТПУ и / или дисплей, если он предусмотрен в программе для Д3-28, должен нормально работать в имитаторе, а обращение программы к другим периферийным устройствам командами с неограниченным временем ожидания СИП (синхроимпульса периферии) будет выглядеть в имитаторе как «зависание» Д3-28 в состоянии с зажжённым индикатором ожидания ответа от периферии. (Сам имитатор при этом не зависает: всегда можно перевести Д3-28 в состояние останова нажатием на её пульте кнопки С или Ш т. е. <C> или <Enter> в имитаторе).

4. Наконец, имеются, условно говоря, «косметические» недостатки имитатора – состояние машины может отображаться на X- и Y-табло в имитаторе не точно так, как в реальности; и, кроме того, – кнопки пульта машины изображаются, но не нажимаются :-). Или, что хуже, – действие некоторых кнопок вообще не имитируется).

Например, если в реальной Д3-28 после выполнения в программе команды гашения CLR X и затем останова табло X выглядит погашенным, то в имитаторе на табло X в разрядах мантиссы изображаются нули. Это различие связано с тем, что команда CLR X очищает регистр X в ОЗУ машины числом 0, имитатор выводит число из регистра X на табло, а в реальной Д3-28 есть ещё и дешифратор, который может (наверное) гасить табло, когда в ОЗУ в регистр X уже занесено число 0. Такое различие не влияет на вычисления, поэтому его допустимо считать чисто «косметическим».

Ещё пример: в имитаторе в режиме «Ввод» допускается ввод знака «минус» для набора отрицательного номера шага при не равном нулю ВР, тогда как в реальной машине Д3-28 такой удобный способ набора отрицательного номера шага не предусмотрен (тут речь идёт скорее о преимуществе, а не о недостатке имитатора).

К «косметическим недостаткам» имитатора можно отнести также отсутствие в имитаторе режимов ПВ (печать программы при её вводе) и П (печать программы), и отсутствие имитации «кнопки с ромбиком» (печать числа из регистра X). Кроме того, отсутствует доступное в реальной Д3-28 пошаговое исполнение программок нажатиями кнопки Ш. Упомянутые режимы печати можно было бы сделать, но представляется, что они будут засорять компьютер неудобными для просмотра и / или ненужными txt-файлами. Компактно оформленные «дампы ОЗУ Д3-28» и аутентично выглядящие листинги можно получать с помощью загружаемых в имитатор программок. Пошаговое слежение за работой программки в имитаторе становится возможным после включения обнаружения возвратов из подпрограмм (клавишами <F1> и <R> в режиме **View > Display**).

Некоторые способы тестирования имитатора

Перечисляемые ниже испытания могут: а) быть неплохим упражнением для восстановления былых либо для приобретения новых навыков работы с Д3-28 :-), б) помочь в проверке имитатора под иными версиями ОС Windows, нежели ME, XP, 7, в) помочь при «портировании» / написании симуляторов или эмуляторов Д3-28 под другие системы, более современные, чем уже устаревшие 32-разрядные Windows, г) дать пользователю конкретную информацию о возможностях имеющегося имитатора .

. Начальный этап проверок, самый простой, – ввод чисел в регистр X разными способами, следуя примерам из *Справочника программиста (3.857.100 Д3)*, стр. 37. Затем – выполнение команд из примеров, приведённых там на стр. 37–40. Для разбора примеров с порядком нормализованных чисел, равным –0, в имитаторе можно включить «косметическую» опцию «to show E = –0»: переходим к **View > Display**, нажимаем <F1>, <E>, и в диалоговом окне с отметкой OFF на вопрос «Change?» отвечаем «Да (Yes)», после чего для продолжения опытов возвращаемся к **View > Machine**. Выключается индикация порядка –0 таким же способом (при наличии в диалоговом окне отметки ON).

. В имитаторе можно выполнить примеры из раздела 3 «Проверка правильности вывода информации на индикацию» в документе *Тест-программа 0.073.016*, этот документ есть здесь:

https://d3-28.ru/programmy/test-programma-0-073-016/ и здесь:
http://retropc.org/Biblioteka_r_31.html#c254 (djvu-файл в «Библиотеке»).

Правда, в последних разрядах 12-разрядной мантиссы в имитаторе наблюдается некоторое отличие от указанных в документе результатов; это следствие упоминавшейся выше разницы математических алгоритмов имитатора и ЭВМ Д3-28.

. Простые примеры ввода данных и вычислений есть также в *Инструкции по эксплуатации (3.857.100 ИЭ)*, стр. 37 – 43. (В таблице 8 с примерами значений тригонометрических функций, по-видимому, попадаются опечатки: для команды с кодом 0806 там приведён неверный ответ; для команды с кодом 0808 приведён ответ при начальных значениях Y = 5 и X = 2, а не при указанных в тексте Y = 3 и X = 2.)

На стр. 96–97 (рис. 11) *Инструкции по эксплуатации* – листинг программки (в машинных кодах) вычисления cos(X); а на стр. 90–93 (и на рис. 9) описывается программка вычисления корней квадратного уравнения. В папке [txt] из zip-архива с имитатором эти программки есть в txt-файлах cos_ris-11_KP-386_N-46.txt и q-eq_ris-9_KP-543_KP-7765_KP-34321.txt (в конце этого txt-файла даны подробные пояснения). Эти простые программки подходят для начального тестирования программного режима работы имитатора. (Если бы не работали они, то уж заведомо не работали бы такие сложные программы, как Бейсик, Фортран, Выстра, Машяз, ОС ВТ-МХТИ-128, проверку работоспособности которых в имитаторе можно рассматривать с этой точки зрения как «высокоуровневый тест»).

. Много числовых результатов приведено в документе *Руководство оператора 3.857.100 Д12*

https://d3-28.ru/dokumentatsiya-d3-28/rukovodstvo-operatora-d3-28/ , см. также:
http://retropc.org/Biblioteka_r_31.html#c254 (djvu-файл в «Библиотеке»).

Это пакет подпрограмм «Статистика». В папке [txt] из zip-архива с имитатором он находится в txt-файле statistika__KP-7074_N-806.txt; в конце там есть пояснения по работе с ним.

. Целенаправленная «проверка системы команд и ОЗУ» осуществляется «028-тестом»; о его работе в ЭВМ Д3-28 говорится на стр. 54–55 в *Инструкции по эксплуатации (3.857.100 ИЭ)*, он там называется «тест-программа И5М0.073.028». У нас есть две разные раскодировки 028-теста с м/ф кассет – старый тест, не определяющий объём ОЗУ, и более новая его версия. Об их работе в имитаторе рассказано ниже в данном «Руководстве к имитатору» в разделе 13 «Приложение 3: дополнительные пояснения. Работа тест-программ в имитаторе».

. Более чувствительным к недостаткам имитатора «высокоуровневым» тестом могут служить разнообразные вычисления в пользовательских программах на языке Бейсик. В том числе – из книги В. П. Дьяконова «*Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ» 1987*; имитатор справляется с такими вычислениями и, значит, его повседневное применение пользователями может «продлить жизнь» этой хорошей книге :-)

Для иллюстрации несколько первых примеров из этой книги приведены у нас в папке [txt] в файле diakonov-87_bas.txt; в конце этого файла даны подробные пояснения. Чтобы продемонстрировать возможность применения имитатора с Бейсиком-157107 для не самых уж элементарных «повседневных» расчетов пользователю (конечно, имеющему ретро-компьютер с 32-разрядной ОС Windows :-) в его «хобби» или учебных занятиях, в этот файл я добавил два своих примера, в которых применяются алгоритмы из указанной книги В. П. Дьяконова. В первом из них методом Симпсона вычисляются интегралы, встречающиеся

в радиотехнической задаче о расчёте активной и реактивной частей импеданса дипольной антенны. Второй пример тоже имеет «студенческий» уровень сложности: численное решение методом Рунге-Кутта уравнения Шрёдингера для волновых функций стационарных состояний $\psi(x)$ гармонического одномерного осциллятора в квантовой механике. Поясню его подробнее.

В примере с квантовым осциллятором речь идёт о дифференциальном уравнении

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + \frac{1}{2}m\omega^2x^2\psi = \varepsilon\psi$$

где: $\hbar$ – постоянная Планка, m – масса частицы, ε – энергия частицы, ω – частота колебаний этой частицы вблизи начала координат $x = 0$, говоря языком классической механики, под действием возвращающей силы $F(x) = -dV/dx = -m\omega^2 x$, где $V(x) = m\omega^2 x^2/2$ – потенциальная энергия; наконец, $\psi(x)$ – подлежащая исследованию волновая функция частицы. Согласно принципам квантовой механики, $|\psi(x)|^2$ есть плотность вероятности обнаружить частицу в окрестности точки x. По физическому смыслу этой задачи, вероятность обнаружить частицу в бесконечном удалении от начала координат равна нулю, поэтому искомые решения $\psi(x)$ подчиняются граничному условию: $\psi(x) \to 0$ при $x \to \pm\infty$.

На лекциях и в учебниках по основам квантовой механики студентам даётся вывод результата: такие решения («убывающие на бесконечности», как принято говорить) существуют только при дискретных значениях энергии: $\varepsilon = \hbar\omega N + \hbar\omega/2$, где $N = 0, 1, 2, 3, 4, \ldots$. Другими словами: подставив это выражение для ε в уравнение Шрёдингера, затем разделив все члены уравнения на так называемую энергию нулевых колебаний $\hbar\omega/2$, обозначив $x^2/(\hbar/(m\omega)) = X^2$ (так что X – безразмерная координата, для которой единицей длины служит так называемая амплитуда нулевых колебаний $(\hbar/(m\omega))^{1/2}$), обозначив функцию $\psi(x)$ с координатой x, выраженной через X, как функцию $Y(X)$, и обозначив двумя штрихами её вторую производную $d^2Y/dX^2 = Y''$, получаем уравнение Шрёдингера в более простой форме

$$Y'' = (X^2 - 2N - 1)\,Y,$$

вот для этого уравнения и доказывается в курсах лекций, что решения с граничными условиями $Y(X) \to 0$ при $X \to \pm\infty$ существуют только при дискретных значениях $0, 1, 2, 3, 4, \ldots$ параметра N. Такие решения $Y_N(X)$, как выясняется в курсах лекций, представляются через известные элементарные функции, а именно – через экспоненту $\exp(-X^2/2)$, умноженную на полиномы Эрмита по степеням переменной X со старшей степенью, равной N.

У студентов, забывающих в ходе обильных выкладок роль условия $Y(\pm\infty) = 0$, часто возникает вопрос: «куда пропали» решения при произвольных значениях параметра N? Как выглядят решения $Y(X)$, если не «квантовать руками» параметр N, а позволить ему изменяться непрерывно, – это допускает «чистая» математика (не ссылающаяся на «физический смысл»)?

Оказывается, при значениях N, не равных $0, 1, 2, 3, 4, \ldots$, решения $Y(X)$ уже не выражаются через элементарные функции. Однако их можно изучить путём численного интегрирования дифференциального уравнения для $Y(X)$, и убедиться, что такие решения $Y(X)$ возрастают, а не убывают с ростом X. Это и делается (при произвольном в разумных пределах N, включая целочисленные значения) в бейсик-программке QUANTUM OSCILLATOR из библиотеки программок в txt-файле diakonov-87_bas.txt.

Подробности о работе с этой программой изложены в конце файла diakonov-87_bas.txt, а здесь посмотрим на её результаты с точки зрения тестирования имитатора. Распечатки значений $Y(X)$, полученных численным интегрированием дифференциального уравнения при нецелочисленных значениях N, сравнить не с чем, но при $N = 0, 1, 2, 3, 4, \ldots$ есть возможность сравнения с расчётом на ПК по явным формулам для $Y_N(X)$, например, в Маткаде.

Берём, полученную в имитаторе в программке QUANTUM OSCILLATOR распечатку столбца со значениями $Y(X)$ при $N = 0$, удаляем в ней нечисловые строки (получившийся файл у меня назывался «19-19-57__26-3-2020_quantu_N-0.txt») и загружаем в документ Маткада, как массив с именем Y0:

```
file0 := "c:\d3-28-imitator\txt\19-19-57__26-3-2020_quantu_N-0.txt"
Y0 := READPRN(file0)
```

В таком массиве содержится 65 значений $Y(X)$, для X от 0 до 4 с шагом $dX = 4/64 = 0.0625$. Создаём массив с этими дискретными значениями X_n , и по формуле с элементарной функцией $\exp(-X_n^2/2)$ получаем для сравнения с Y0 «эталонный» массив; обозначаем его как P0:

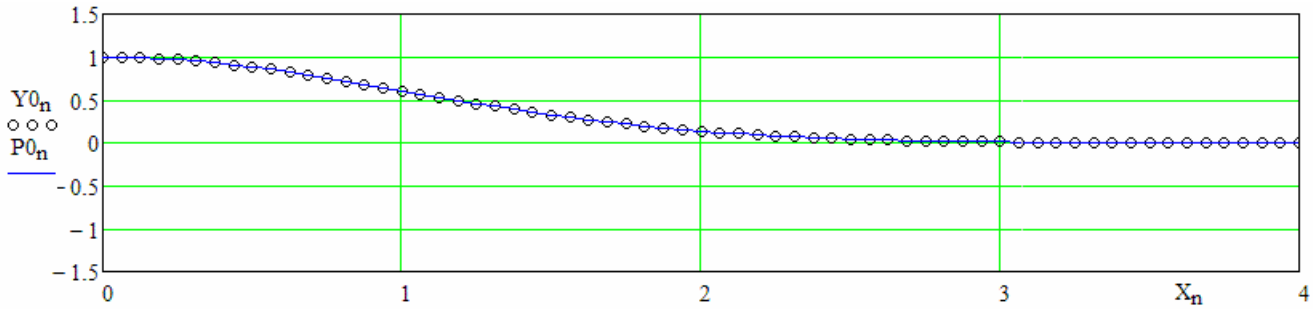
$$dX := \frac{4}{64} \qquad dX = 0.0625$$

$n := 0, 1 .. 64$ // начало цикла по n от 0 до 64 с шагом 1

$$X_n := dX \cdot n$$

$$P0_n := \exp\left[\frac{-(X_n)^2}{2}\right]$$

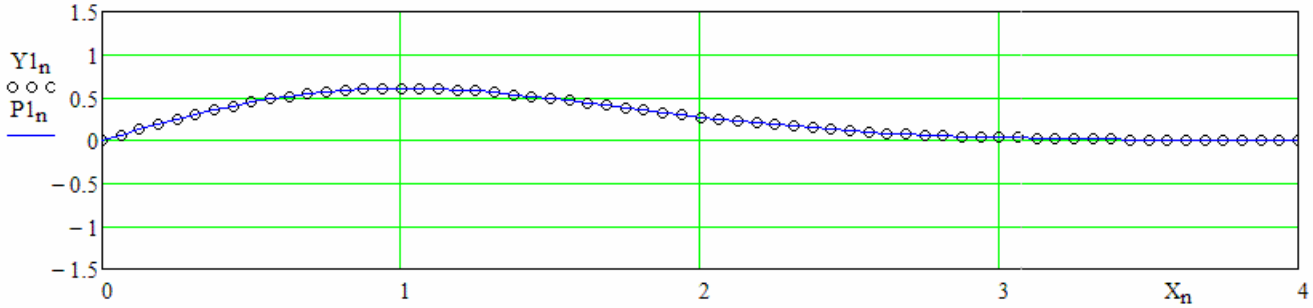
И с помощью графика сравниваем вычисленные в имитаторе в бейсик-программке значения Y0 с эталонными значениями P0, найденными по точной формуле в Маткаде. На графике значения Y0 изображены кружочками, а по значениям P0 проведена тонкая плавная линия:



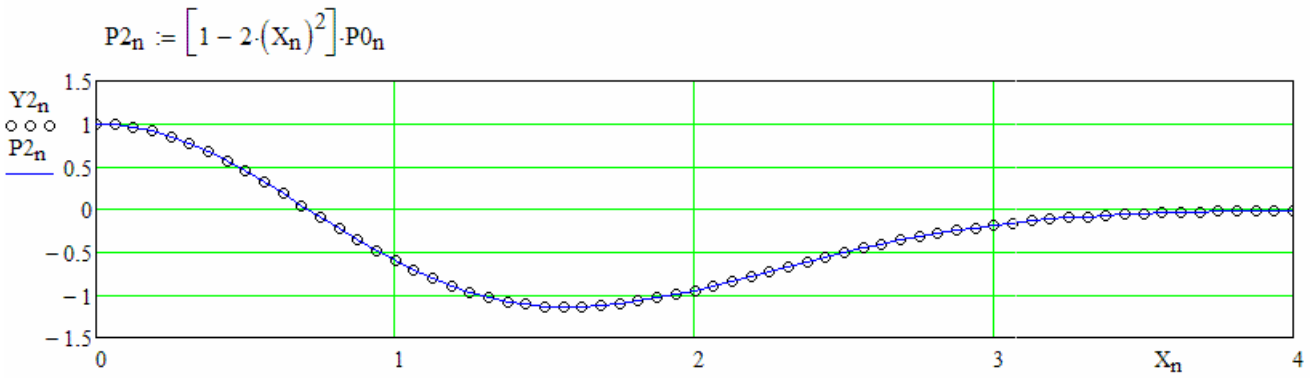
Видно, что кружки лежат на линии, т. е. имитатор справился с задачей. Точность совпадения Y0 с P0, оцениваемая по формуле $(P0_n - Y0_n)/P0_n$, в области $X < 3.5$ лучше одного процента. Вблизи края с $X \approx 4$ значения волновой функции становятся малыми, и точность ухудшается до 18 %, но её достаточно для качественных выводов о поведении волновой функции.

Аналогичное сравнение имитаторного расчета $Y(X)$ при $N = 1$ с вычислением по точной формуле в Маткаде также радует глаз (причём, чем больше N, тем лучше точность совпадения):

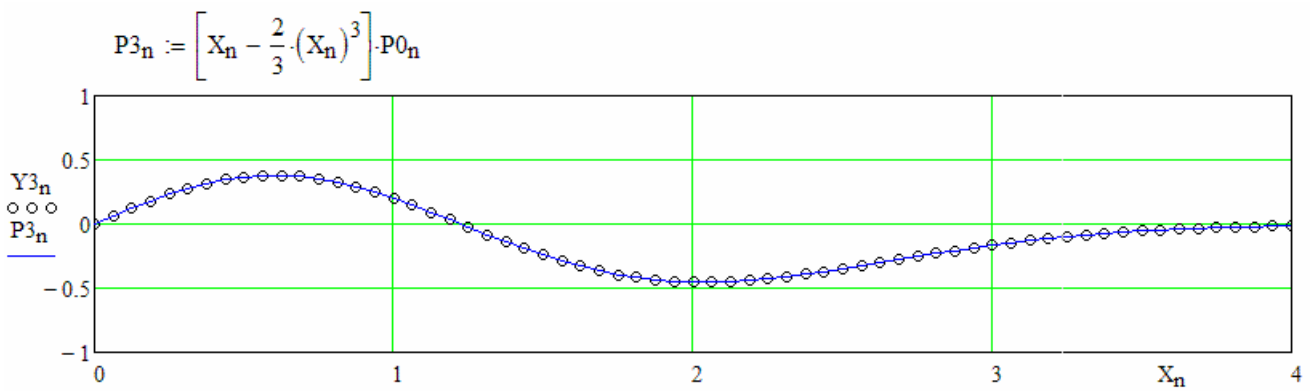
$$P1_n := X_n \cdot P0_n$$



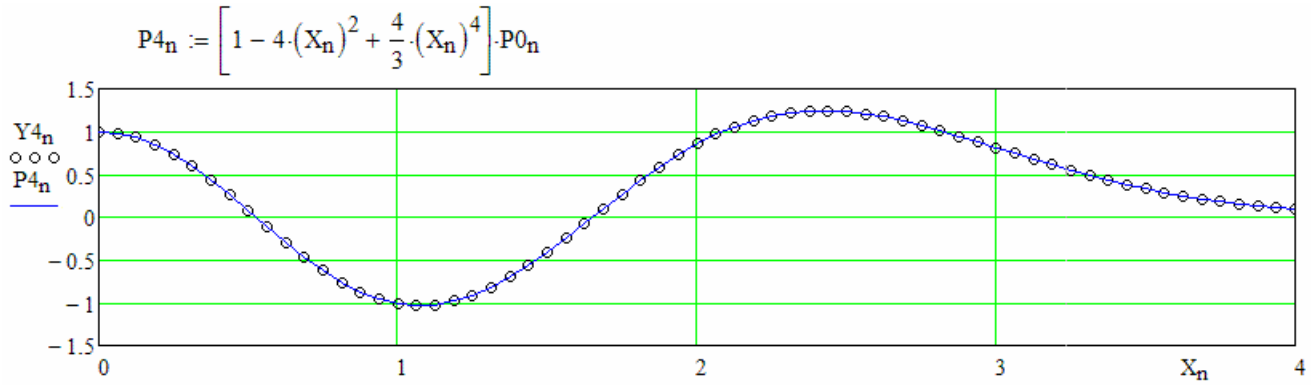
Картина при $N = 2$:



Картина при $N = 3$:



Картина при $N = 4$:



И так далее.

На примере этой учебной задачи видно, что в имитаторе с Бейсиком-157107 можно выполнять довольно интересные вычисления – прямо чуть ли не как в Маткаде :-). Нет, конечно, не говорю, что имитатор способен 100-процентно заменить Маткад. В частности, графические возможности алфавитно-цифрового дисплея – очень уж скромные. Но всё-таки они не нулевые. Например, вот как в программке QUANTUM OSCILLATOR выглядит график волновой функции, аналогичный приведённой выше маткадной картинке при $N = 4$:

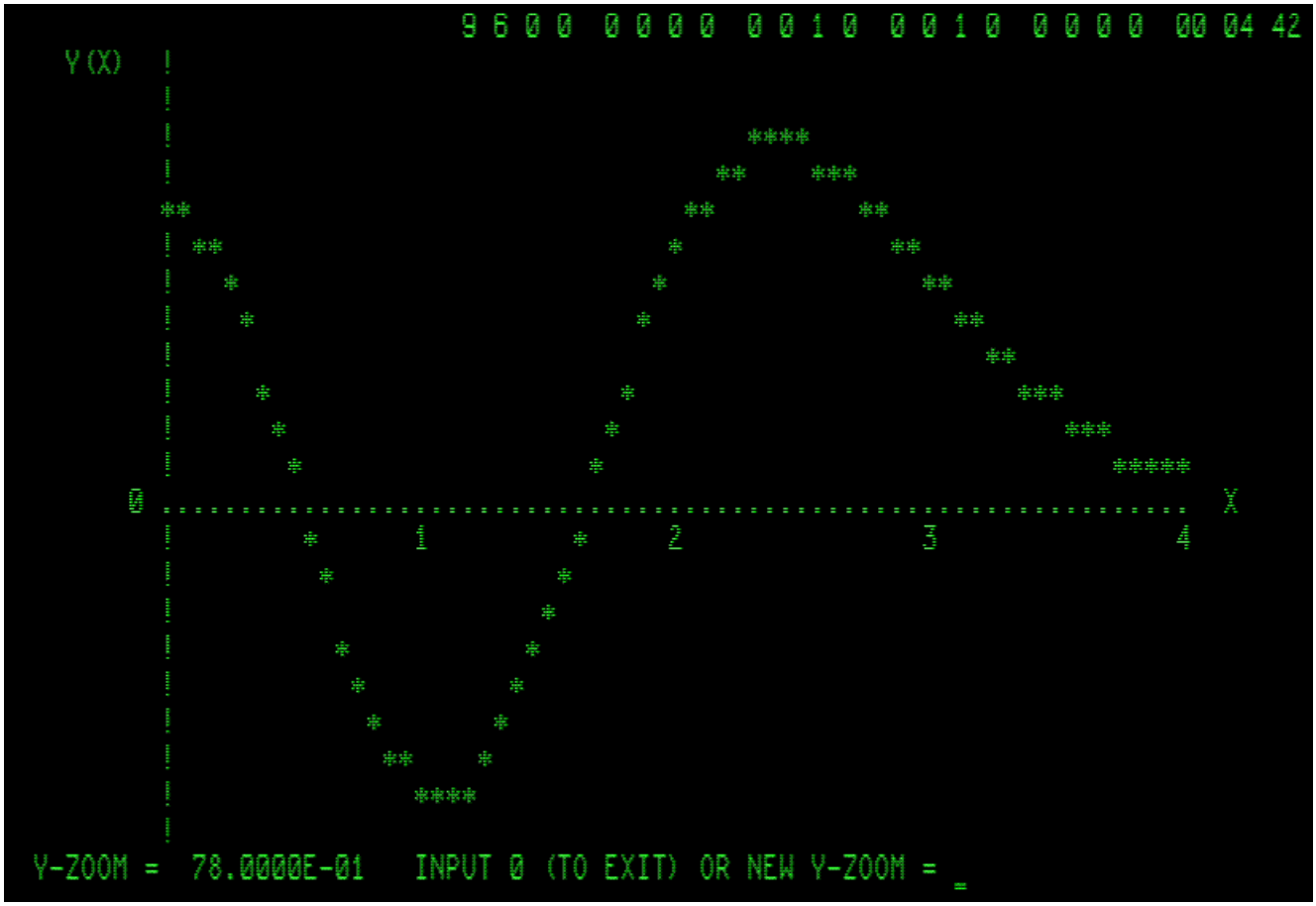
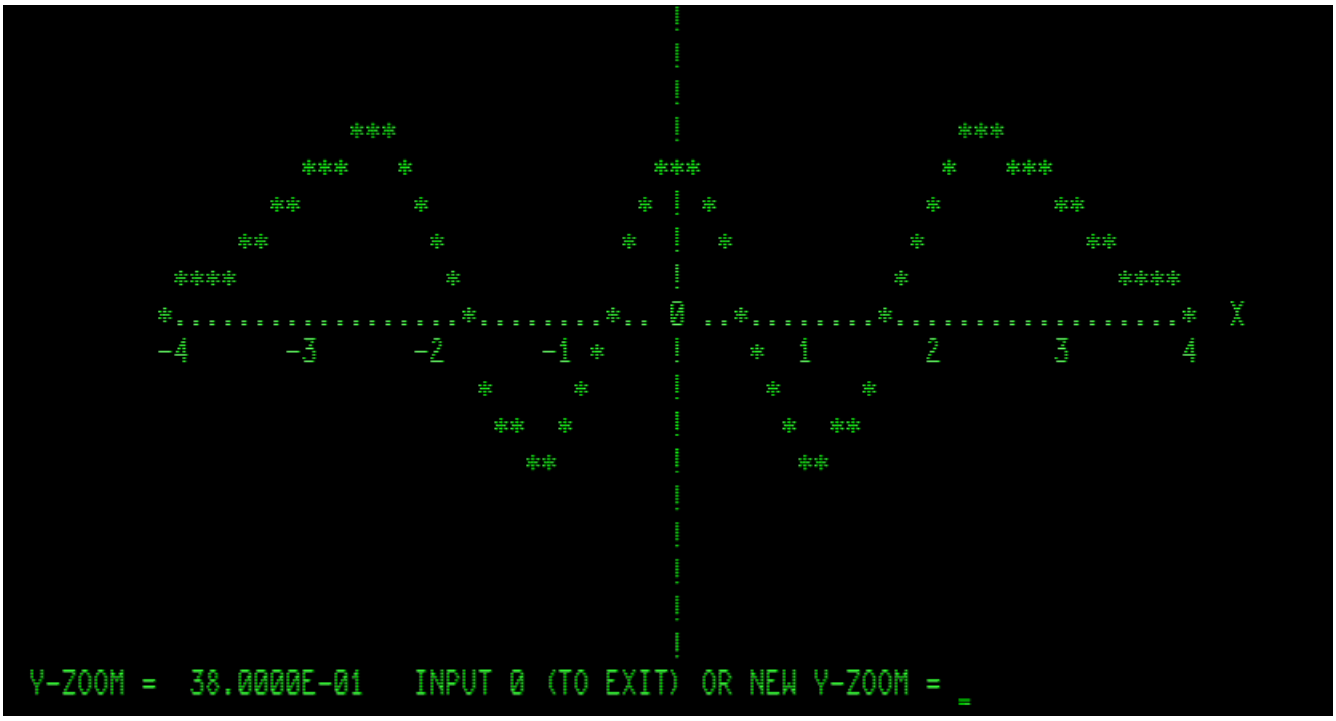
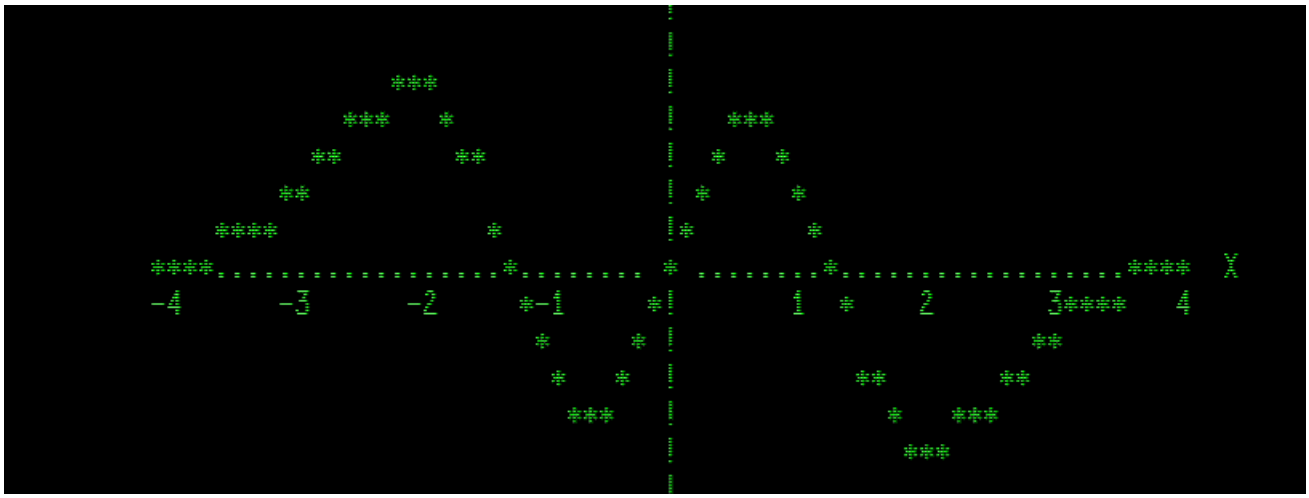


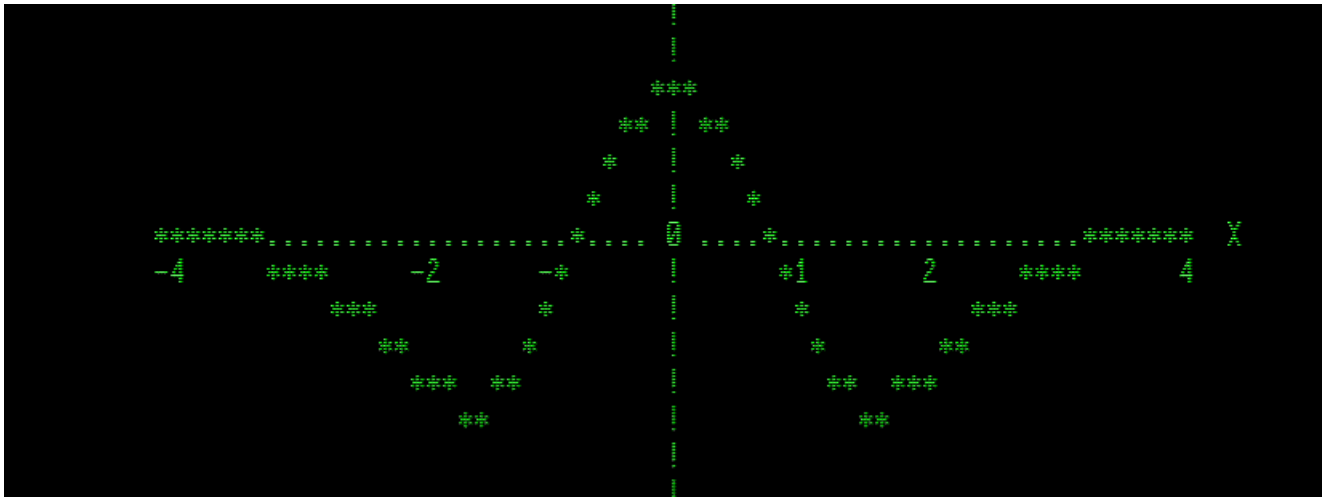
График этой же волновой функции ($N = 4$, чётное решение) в интервале $-4 < X < 4$:



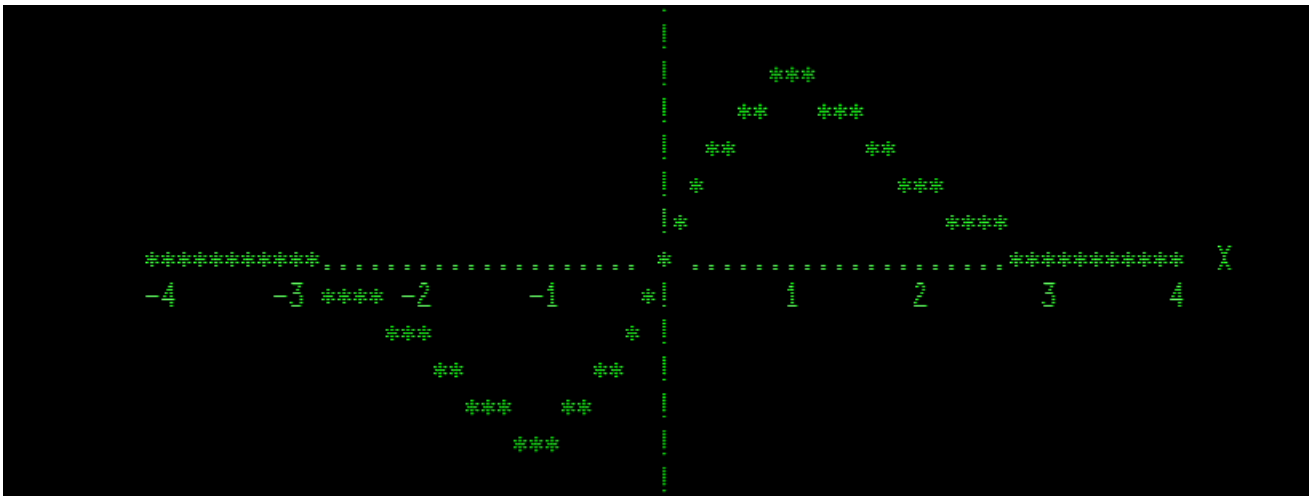
Заодно можно взглянуть на аналогичные графики волновых функций для предшествующих уровней энергии квантового осциллятора. Картина для стационарного состояния с $N = 3$:



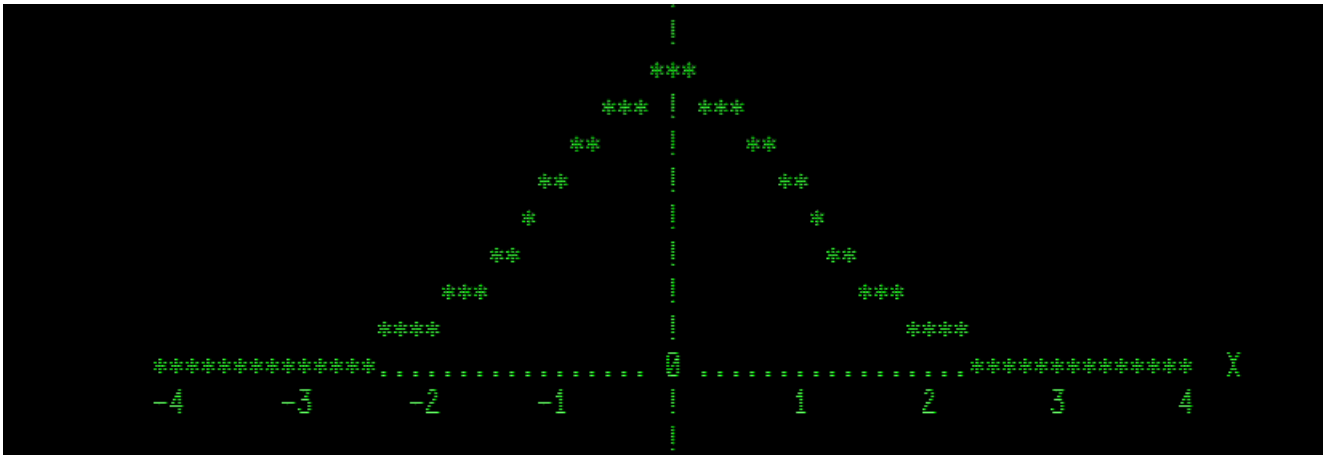
Волновая функция стационарного состояния с $N = 2$ (чётное решение):



Волновая функция стационарного состояния с $N = 1$ (нечётное решение):

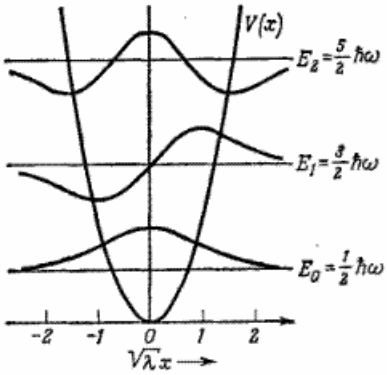


И, наконец, график волновой функции так называемого основного состояния осциллятора (т. е. состояния на самом нижнем энергетическом уровне), $N = 0$, чётное решение, без узлов:



В учебниках по квантовой механике можно встретить качественно такие же графики волновых функций стационарных состояний осциллятора, рассчитанных по формуле $H_N(x)\exp(-x^2/2)$, где $H_N(x)$ – полиномы Эрмита. Для примера вот фрагмент текста с рисунком со стр. 82 из тома 1 известной книги Флюгге «Задачи по квантовой механике» М.: Мир, 1974:

Легко видеть, что собственные функции обладают определенной четностью. В самом деле, в силу симметрии потенциала, $V(-x) = V(x)$, функция $u_n(-x)$ является решением дифференциального уравнения наряду с функцией $u_n(x)$ и принадлежит тому же самому собственному значению (30.8). Так как вырождение отсутствует, то оба решения могут различаться лишь постоянным множителем f, причем в силу условия нормировки $|f|^2 = 1$. При повторном изменении знака x мы возвращаемся к исходному решению, поэтому $f^2 = 1$ или $f = \pm 1$, следовательно, всякая собственная функция должна быть либо четной, либо нечетной.



Фиг. 19. Потенциал, энергетические уровни и собственные функции гармонического осциллятора.

Таким образом, численный метод решения уравнения Шредингера (с граничными условиями $Y(0)=1$, $Y'(0) = 0$ для чётной функции $Y(X)$, или $Y(0)=0$, $Y'(0) = 1$ для нечётной $Y(X)$) в имитаторе Д3-28 с программкой на Бейсике, использующей алгоритм Рунге-Кутта из *Справочника* В. П. Дьяконова, даёт верный результат.

. Следующий тест тоже включает расчёты на Бейсике – в этом плане он принципиально не отличается от тестирования сравнением с результатами учебных расчётов из книг или из Маткада, – но однако в нём численное сравнение ведётся с реальными данными одной научной работы, которые количественно обрабатывались в бейсик-программе на реальной машине Д3-28 и затем распечатывались у нас на ТПУ (в 1980-х годах). Такой тест имитатора не является необходимым (но для меня он важен, в том числе, из-за ностальгической составляющей :-)

Чтобы тому, кто вдруг это читает, было понятнее и интереснее, поясню, что и как делалось в той работе. На самодельной экспериментальной установке (создававшейся в течение ≈ 5 лет, это была целая эпопея) мы измеряли теплоёмкость $C(T)$ вещества $Pb_{1-x}TeTl_x$ при очень низкой температуре T: 1–2 К. Такая температура получалась откачкой паров жидкого гелия, который при этом становился сверхтекучим, норовил ползти вверх по стенкам сосуда и быстро испаряться, и поэтому сосуд с ним, а также – держатель с образцом, нагревателем, термометром и механическим тепловым ключом, – всё размещалось в сложном многослойном криостате с охлаждающими азотными / гелиевыми и теплоизолирующими вакуумными «рубашками». Цель опытов – изучение сверхпроводимости $Pb_{1-x}TeTl_x$, обнаруженной нами в 1981 году. (Кратко об этой физике говорится в обзоре в УФН т. 145 1985 г., стр. 74–76. В суровые 1990-е дорогостоящий жидкий гелий стал платным, нам недоступным. В начале 2000-х мой замечательный научный руководитель И. А. Черник – основной мотор нашей деятельности – всё сильнее болел, и дни его оказались сочтены ... Преподавательская нагрузка моя росла, утяжелилась курсом квантовой механики; его я затем вёл для студентов радиофизического факультета 18 лет, пока окончательно не ушёл на пенсию, а экспериментальную работу не смог продолжить. Конец истории: через четверть века после нашего обнаружения сверхпроводимости $Pb_{1-x}TeTl_x$ тему подняли и развили сотрудники американского физика Джебелла, даже придумали свою теоретическую модель.)

Опыты, включая приготовление образцов, были сложными в исполнении, со многими методическими нюансами. Но сама идея измерения $C(T)$ простая. В образец вещества (≈ 140 г, это примерно половина моля), охлаждённый до ≈ 1 К, подавался импульс энергии $t \cdot I \cdot U$, где $t = 10$ сек – длительность импульса электрического тока I в нагревателе, U – напряжение на выводах нагревателя. Измеряли: значения I, U (приборами ЦУИП), и с особой точностью (по записи на диаграммной ленте в самописце КСП-4 с калиброванной отсечкой «постоянной составляющей» потенциометром Р306 и усилением «полезного сигнала» фотоэлектрическим усилителем Ф116) V_{T1} – напряжение на специальном резисторе-термометре до подачи импульса, и V_{T2} – после; ток J в термометр задавался стабильным источником, он был заранее известен. Эти данные вручную вводились в машину Д3-28. Машина по программе с довольно громоздкой формулой (градуировкой) для $T(R)$ вычисляла через два значения сопротивления термометра $R_1 = V_{T1}/J$ и $R_2 = V_{T2}/J$ значения температуры образца T_1 и T_2 до и после импульса, среднюю температуру $T = (T_1 + T_2)/2$, и теплоёмкость $t \cdot I \cdot U / (T_2 - T_1)$. Из этой теплоёмкости машина вычитала вычисленную по аппроксимирующему полиному теплоёмкость держателя с термометром и нагревателем, умножала результат на корректирующую «аппаратную функцию» $K(T)$ (уточняемую время от времени так, чтобы при проверочных измерениях теплоёмкости чистой меди получать эталонный результат в соответствии с известным из литературы «Copper Reference Equation» – CRE; это было необходимо потому, что градуировка термометра немного «плыла» со временем из-за многократных циклов охлаждения-нагревания) и умножала на молярный коэффициент образца K_M – чтобы получить теплоёмкость $C(T)$ одного моля вещества. Так или примерно так измеряется одна температурная «точка»; затем всё это повторяется при получившейся более высокой температуре T – тем самым измеряется следующая температурная «точка» $C(T)$, и так далее.

Такие измерения мы (притом вместе со студентами) проводили в течение многих лет – изучали по поведению $C(T)$ физику возникновения сверхпроводимости в зависимости от состава, дополнительных примесей и т.п. на различных сериях образцов веществ типа PbTe<Tl>. Студенты у нас были все хорошие, помогали и с приготовлением образцов, и в ходе измерений, и при обработке записей из тетрадей с результатами.

В ходе измерений приходилось напрягаться – безошибочно, но быстро (т. к. из-за паразитного теплоподвода образец довольно быстро нагревался даже без наших 10-секундных импульсов энергии) записывать в тетрадь номера «точек» и соответствующие показания ЦУИПов I и U, обдумывая одновременно, какие надо готовить (на эквиваленте нагревателя) значения I для следующих точек. В 1986 году наш очень хороший студент С. Г. Кобышев сумел «покончить с этим безобразием» (хоть и частично, но это очень облегчило работу): он нашёл в лаборатории валявшиеся без дела транскриптор Ф5033 и печатающее устройство ЭУМ-23, разработал, отмакетировал и спаял две платы для сопряжения с цифровым выходом ЦУИП-ов и запуска печати с необходимой задержкой от «реле времени», подававшего ток в нагреватель

образца. Эта техника избавила нас от необходимости (чреватой ошибками, для исправления которых приходилось повторно измерять весь ход $C(T)$) записывать I и U руками: ЭУМ-23 автоматически распечатывала номера точек и I, U; вручную надо было только добавлять в распечатку значения V_{T1} и V_{T2} с диаграммной ленты и затем такой блок I, U, V_{T1}, V_{T2} вводить с клавиатуры фрязинского терминала в программу в Д3-28.

Сохранилась (правда, почти совсем обесцветившаяся) ТПУ-распечатка краткой версии бейсик-программы, вычислявшей $C(T)$. В её начальных строках видны массивы «аппаратных» чисел – вспомогательные константы и коэффициенты полиномов для градуировки термометра, теплоёмкости держателя, CRE:

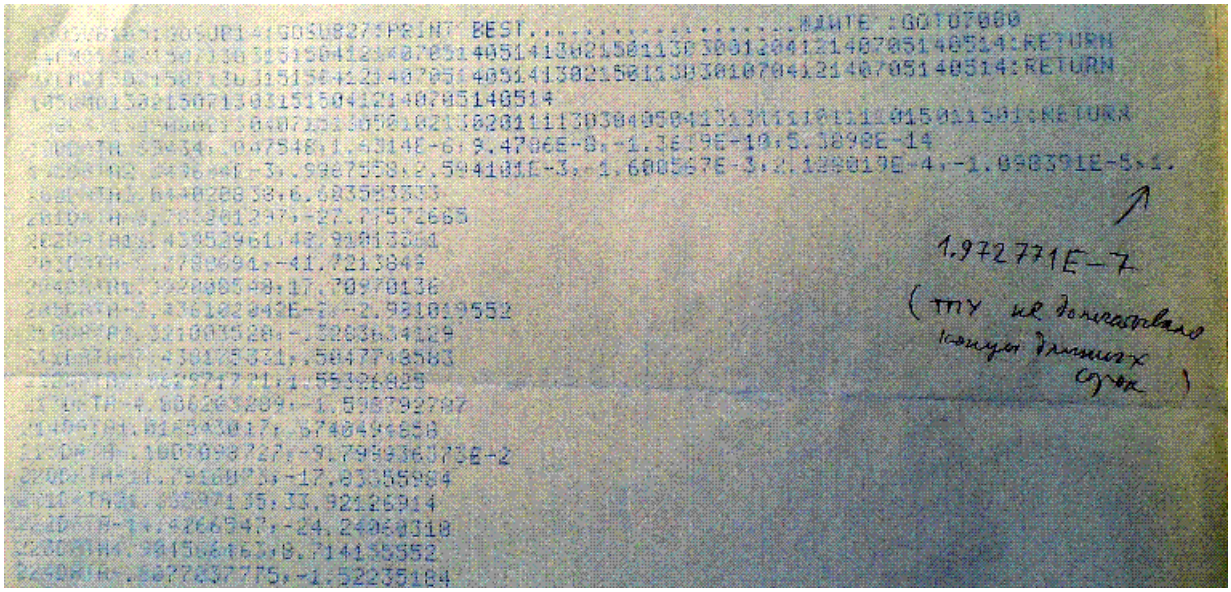
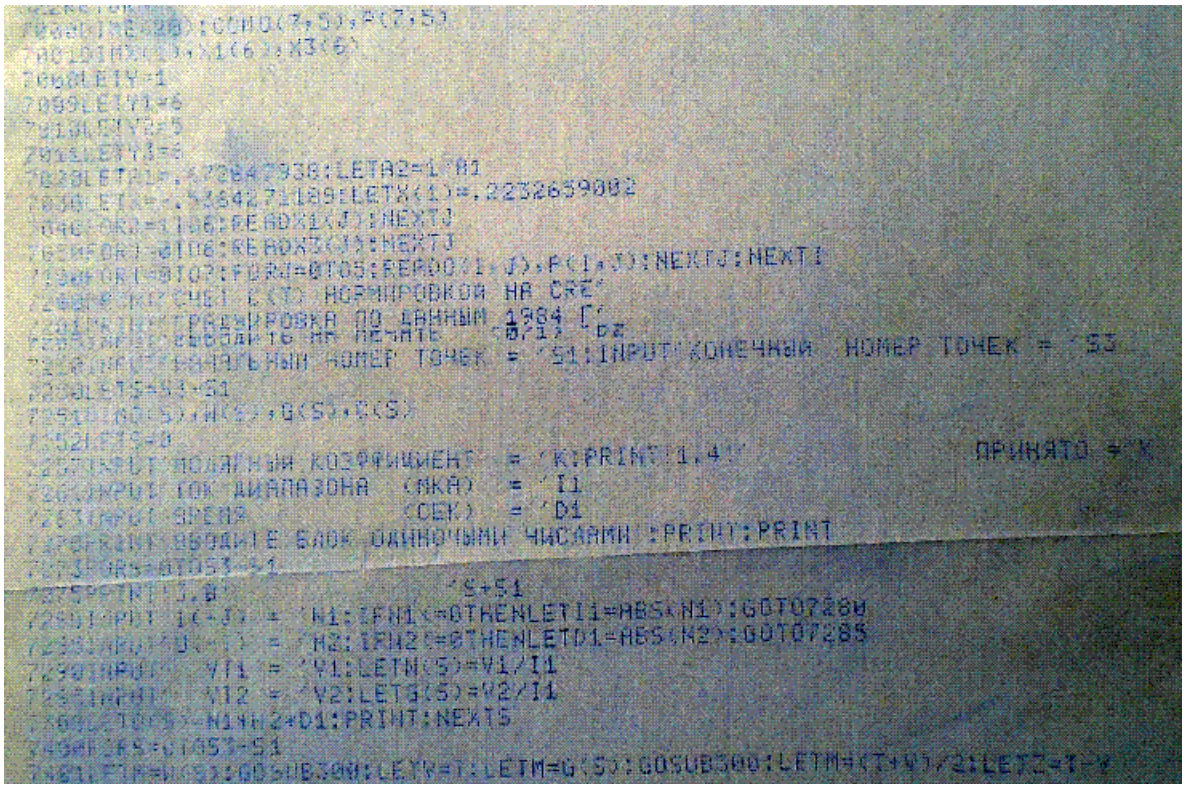
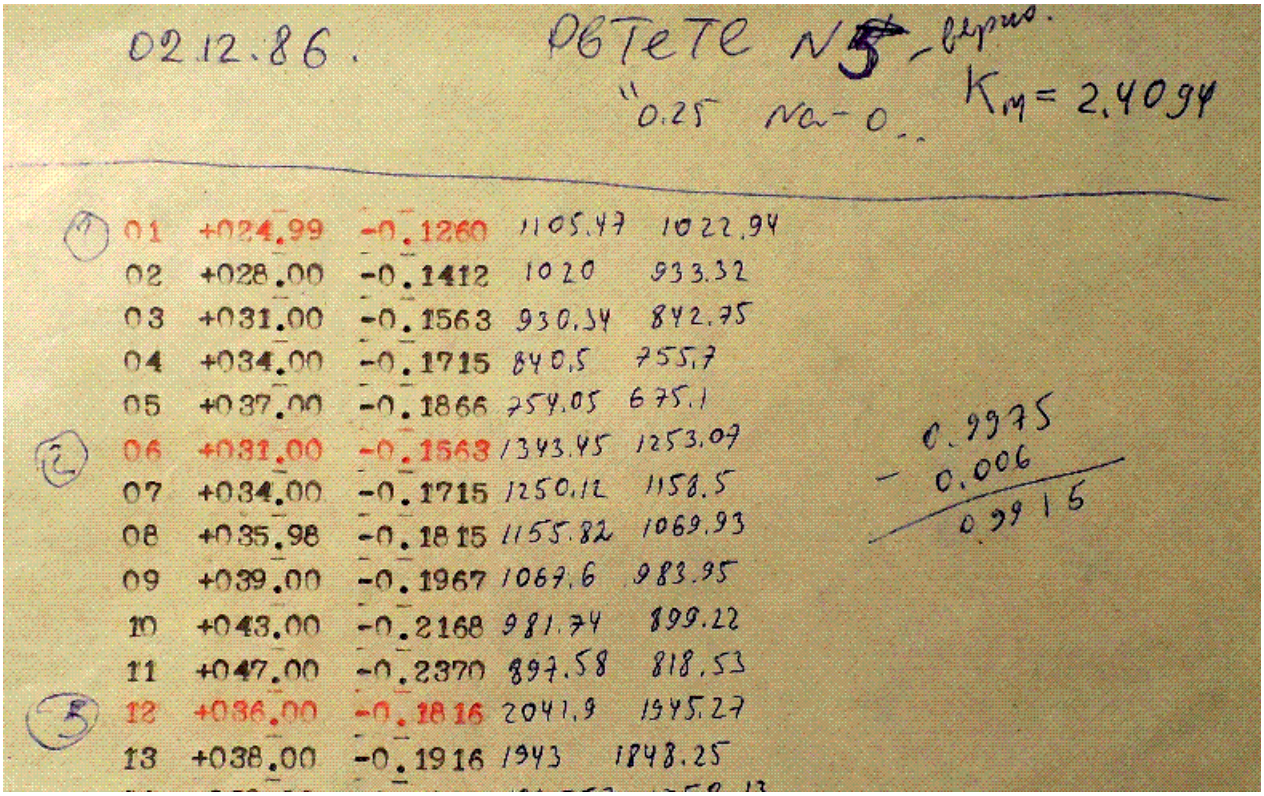


Фото фрагмента с инициализацией переменных и начальным диалогом:



По распечатке можно набрать программу в имитаторе, хотя трудно набирать такое множество чисел, не ошибаясь. Однако эта программа нашлась и в оцифровке одной из кассет; её раскодировка, пригодная для запуска в имитаторе с Бейсиком-157107 теперь имеется в папке [txt] в txt-файле ok__heat-capacity__bas.txt, она там вторая от начала файла (см. пояснения в конце указанного txt-файла). В конце txt-файла есть листинг этой программки; по нему видно, какие выполняются математические действия: наряду с подсчётом значений полиномов присутствует вычисление логарифма, взятие нецелой степени числа, деление.

Сохранились также некоторые результаты измерений, печатавшиеся на ЭУМ-23. Для примера вот фото первого десятка точек из такого протокола измерений для одного из образцов:



01	+024.99	-0.1260	1105.47	1022.94
02	+028.00	-0.1412	1020	933.32
03	+031.00	-0.1563	930.34	842.75
04	+034.00	-0.1715	840.5	755.7
05	+037.00	-0.1866	754.05	675.1
06	+031.00	-0.1563	1343.45	1253.07
07	+034.00	-0.1715	1250.12	1158.5
08	+035.98	-0.1815	1155.82	1069.93
09	+039.00	-0.1967	1069.6	983.95
10	+043.00	-0.2168	981.74	899.22
11	+047.00	-0.2370	897.58	818.53
12	+036.00	-0.1816	2041.9	1945.27
13	+038.00	-0.1916	1943	1848.25

Здесь цифры слева, обведённые кружками, это значения тока в термометре *J*. Дело в том, что градуировка термометра была сильно нелинейная; сопротивление термометра экспоненциально резко уменьшалось с повышением температуры. Поэтому весь температурный интервал был разбит на несколько диапазонов; ток термометра надо было увеличивать от диапазона к диапазону: 1 мкА, 2 мкА, 5 мкА, 10 мкА, и так далее.

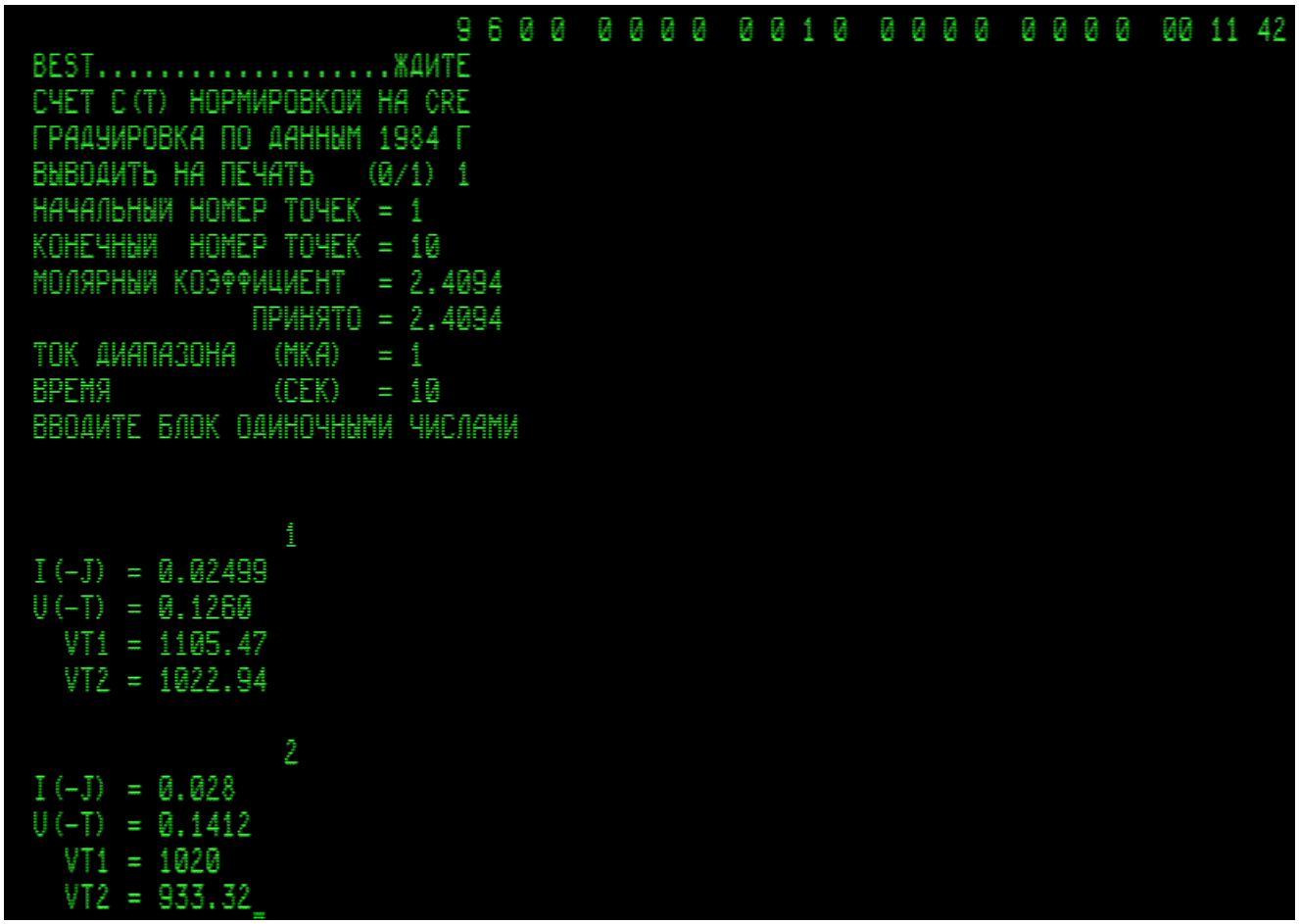
На фото видно, что первая «точка» (строчка числовых данных) в каждом диапазоне печаталась красным цветом; остальные точки печатались чёрным цветом. Значит, в ЭУМ-23 было две печатающих ленты, красная и чёрная; но как достигалось их переключение, – автоматически с помощью транскриптора или вручную, – уже не вспомню. В программе для смены «тока диапазона» его значение надо было вводить с минусом вместо *I* (и, аналогично, для смены длительности импульса его новое значение надо было вводить с минусом вместо *U*).

Затем в распечатке слева направо идут столбцы: номеров точек, значений *I* (например, 0.02499 мА), значений *U* (например, 0.1260 В), и затем – вписанные вручную значения V_{T1} и V_{T2}.

Наконец, самое главное с точки зрения «тестирования имитатора» с помощью этих старинных материалов: сохранилась реальная ТПУ-распечатка результатов работы бейсик-программы, вычислявшей $C(T)$ на реальной Д3-28 (см. фото ниже, после скриншотов имитатора). Программа распечатывала также значения $C(T) / T^3$ и $C(T) / T$ – такие функции были нужны в ходе дальнейшего анализа и физической интерпретации результатов. (Физику здесь не обсуждаю. Отмечу только, что научные результаты С. Г. Кобышева составили содержание его очень хорошей выпускной дипломной работы; за неё весной 1987 года он получил от государственной экзаменационной комиссии оценку «отлично».)

Собственно «тестирование имитатора» в этом примере заключается в том, чтобы вводить данные из указанного выше реального протокола измерений в программу вычисления $C(T)$ в имитаторе и сравнивать имитаторный результат с реальной старинной ТПУ-распечаткой.

Ниже показан скриншот начального диалога программы в имитаторе с Бейсиком-157107. После диалога там видны строки с набором первых двух точек из распечатанного на ЭУМ-23 протокола измерений. В этом примере, как видно из начального диалога, выполняется расчёт $C(T)$ для первых десяти точек:

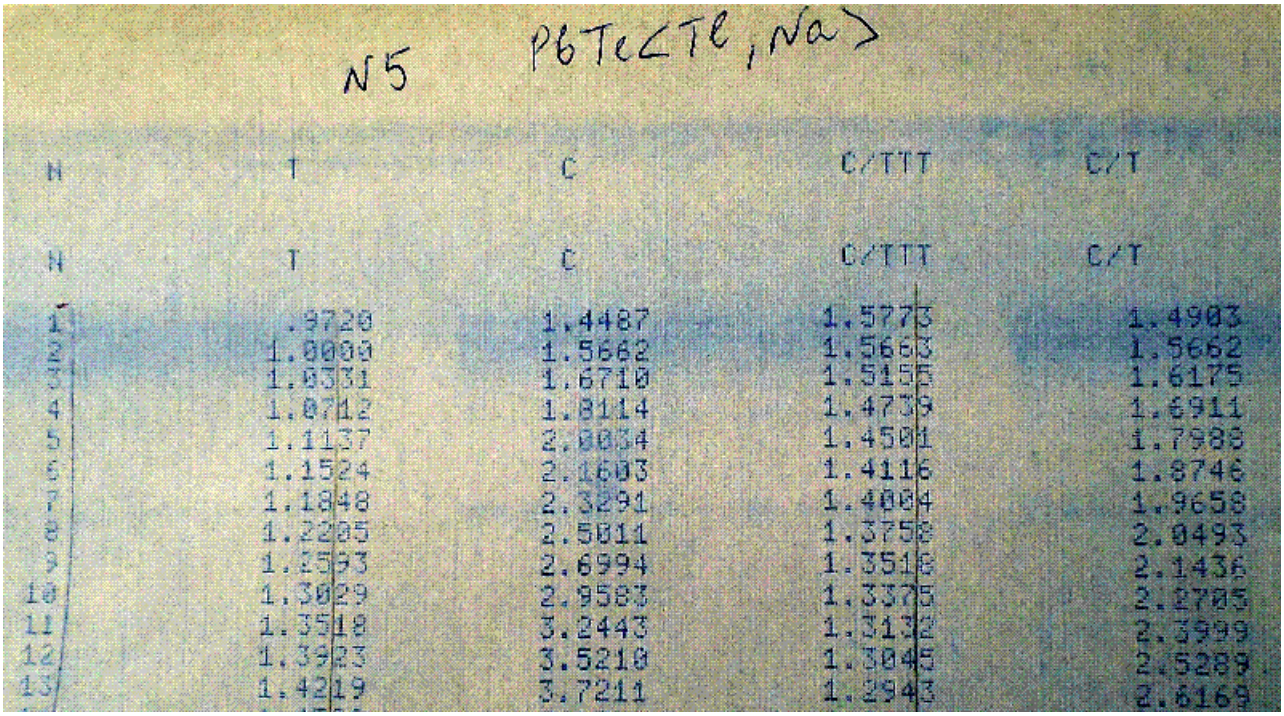


И так далее. В соответствии с протоколом измерений при вводе строчки с номером 6 производится изменение «тока диапазона»: для параметра I вводим отрицательное значение –2. Его абсолютную величину, т. е. 2 программа присваивает параметру «ток диапазона» и повторяет запрос I для ввода значений I, U, V_{T1}, V_{T2} с номером 6.

Скриншот с окончанием ввода десяти точек и последующим выводом результатов имитаторного расчёта T в градусах Кельвина, $C(T)$ в единицах мДж / (моль·К), а также $C(T) / T^3$ и $C(T) / T$:

```
                                   9 6 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  00 29 14
I(-J) = 0.039
V(-T) = 0.1967
  VT1 = 1067.6
  VT2 = 983.95

                10
I(-J) = 0.043
V(-T) = 0.2168
  VT1 = 981.74
  VT2 = 899.22

   1  T =      .9720  C =     1.4487  C/TTT =     1.5773  C/T =     1.4903
   2  T =     1.0000  C =     1.5662  C/TTT =     1.5663  C/T =     1.5662
   3  T =     1.0331  C =     1.6710  C/TTT =     1.5155  C/T =     1.6175
   4  T =     1.0712  C =     1.8114  C/TTT =     1.4739  C/T =     1.6911
   5  T =     1.1137  C =     2.0034  C/TTT =     1.4501  C/T =     1.7988
   6  T =     1.1524  C =     2.1603  C/TTT =     1.4116  C/T =     1.8746
   7  T =     1.1848  C =     2.3291  C/TTT =     1.4004  C/T =     1.9658
   8  T =     1.2205  C =     2.5011  C/TTT =     1.3758  C/T =     2.0493
   9  T =     1.2593  C =     2.6994  C/TTT =     1.3518  C/T =     2.1436
  10  T =     1.3029  C =     2.9583  C/TTT =     1.3375  C/T =     2.2705

ОСТАНОВ В СТРОКЕ7999
:_
```

Эту таблицу сравниваем с распечаткой результатов аналогичного расчёта на реальной Д3-28:



N5 PbTe<Te, Na>

N	T	C	C/TTT	C/T
N	T	C	C/TTT	C/T
1	.9720	1.4487	1.5773	1.4903
2	1.0000	1.5662	1.5663	1.5662
3	1.0331	1.6710	1.5155	1.6175
4	1.0712	1.8114	1.4739	1.6911
5	1.1137	2.0034	1.4501	1.7988
6	1.1524	2.1603	1.4116	1.8746
7	1.1848	2.3291	1.4004	1.9658
8	1.2205	2.5011	1.3758	2.0493
9	1.2593	2.6994	1.3518	2.1436
10	1.3029	2.9583	1.3375	2.2705
11	1.3518	3.2443	1.3132	2.3999
12	1.3923	3.5210	1.3045	2.5289
13	1.4219	3.7211	1.2943	2.6169

Видно, что имитатор выдержал этот тест: имитаторные и реальные результаты совпадают. (Столбцы в реальной ТПУ-распечатке оформлены немного иначе, но это неважно: просто расчёт на реальной Д3-28 вёлся в другой версии нашей пользовательской бейсик-программы «BEST». Разные версии различались вспомогательными опциями, такими как возможность записи блока на МЛ, выбор способа ввода данных – с МЛ, или точка вручную и сразу расчёт, или вручную весь блок точек, а потом расчёт блока. Вычислительное «ядро» во всех версиях было одним и тем же.)

. Выполнен и ещё один тест имитатора, может быть, самый серьёзный. Поскольку я не владею программистскими знаниями на должном уровне, то не представляю толком, как работают компиляторы. Поэтому мне думается, что компиляторы – очень сложный вид программ. Если этот так, то очень серьёзным тестом имитатора является сравнение результатов работы компиляторов в имитаторе и в реальной машине Д3-28.

Для такого сравнения был предпринят поиск в имеющихся оцифровках кассет пар записей, представляющих собой пары типа «Исходная Программа» → «Рабочая Программа», которые записывались на реальной Д3-28. Удалось найти четыре такие пары: две из них получены в системе программирования «Выстра», одна – в «ОС МХТИ 32к, версия ЛФТИ», и одна – в «Fortran-5M».

Файлы wav и txt-раскодировки с этими парами программ помещены в папку D3-28_wav на Яндекс-диске. Там же находится pdf с очень подробным описанием действий с ними по тестированию имитатора, и результатов действий: info_5__prg_dla_compil__1-03-2020.pdf.

Говоря кратко, итог состоит в том, что не «наскоком», а при определённом старании в имитаторе удаётся воспроизвести полученные на реальной машине Д3-28 результаты компиляции во всех четырёх примерах точно – с побайтным совпадением РП.

10. Перечень программ в txt-папке, прилагаемой к данной версии

Basic_D3-28_v3A__KP-157107__N-11343.txt,
Basic_D3-28_v3A__KP-132259__N-9675.txt,
Basic3A-tdm-l__KP-224538_N-15856.txt,
Basic_D3-28_v3A__KP-157057__N-11343.txt (предоставлен **Сергеем Фроловым**),
s3_side1__basic-tdm.txt (из коллекции форумчанина с ником **sanders**).

k31a_basic_KP-177107_N-12820.txt (из коллекции **Виталия Колесника**),
Это версии системы программирования на языке «Бейсик для Д3-28, вариант 3А».
Запуск: <C>, <S>; в имитаторе всегда используются только латинские клавиши. О настройке служебной строки дисплея и другие подробности см. в разделе 5.

k35b_Basic_plan_KP-192147_N-12621.txt (тоже из коллекции **Виталия К**. По записям на МЛ этот Бейсик несовместим с предыдущими Бейсиками; см. test-basic-plan__2prg__plan-bas.txt).

k35b_1-44_formater__plan-bas.txt – программа форматирования текстов. Работает под управлением Бейсика-План; см. пояснения в конце этого же txt-файла.

Fortran-85_p1-p2.txt – система программирования «Fortran-85 для Д3-28», (с моей кассеты).
Запуск: <C>, <T>, <T>. Подробности см. в разделе 5, а также в разделе 11: Приложение 1.

Fortran-5m_KP-138177_N-10414.txt – система программирования «Fortran-5M для Д3-28».
Запуск: <C>, <T>, <T>. Подробности см. в разделах 5 и 11. Это вариант, очень близкий к Fortran-85, но состоящий из одного файла. Работает лучше, чем Fortran-85; и скомпилированный в нём код работает лучше.

D3-28_Vystra__KP-113478__N-9478.txt – система программирования «Выстра».
Запуск: <C>, <S>. Подробности см. в разделе 5 и в разделе 12: Приложение 2.

---

s4_side2_os-mhti_32k.txt (из коллекции **sanders**) – ОС МХТИ, версия ЛФТИ; см. раздел 5.

Ещё ряд программ из коллекции **Виталия К**.:

k15a_vt-9r.txt, k31b_mhti-10r_2x__ks_610_3677_610.txt ,
k17a_vt-12r.txt. Это близкие к Фортрану-5М (и -85) системы «ОС ВТ-МХТИ».

Fortran_v1-4.txt – «Фортран Кочеткова» (ЛИИЖТ, кафедра ТОЭ).
Подробнее см. в разделе 5 и по ссылке:
https://d3-28.ru/programmy/fortran-d3-28-kochetkova-v-m-liizht/

Эти фортраноподобные программы для Д3-28 работают с памятью объёмом только 32 килобайта.

---

k111b__d8__KP-33920_N-2909_tpu_v1.txt – «Программа распечатки Д8», отредактированная мной для работы с ТПУ. Руководство по работе с ней приведено в конце этого же txt-файла.

spp-8___KP-65890_N-5297.txt – система подготовки программ, МХТИ. Пояснения см. в разделе 5, в spp-8_mnemocodes.txt, и по ссылке: https://d3-28.ru/programmy/spp-8-sistema-podgotovki-programm/

---

k12b_et-10__KP-3550.txt – «операционная система ЭТ-10», см.:
https://d3-28.ru/programmy/et-10-tsejtlina-a-s-moskovskij-vniiz/

k12b_et-10__red-28213.txt – версия с, как я думаю, исправленной мной ошибкой дизассемблера; пояснения в конце этого же txt-файла.

---

mashyaz___KP-108069_N-8130.txt и k7a__mashyaz_KP-108090_N-8130.txt

Это система подготовки и отладки программ, содержит реассемблер команд Д3-28, может работать с ОЗУ 128 килобайт. Пояснения см. в разделе 5 и по ссылке
https://d3-28.ru/dokumentatsiya-d3-28/mashyaz/

k7b_minimonitor.txt. Порядок запуска описан в конце этого же txt-файла.
Документацию см. по ссылке: https://d3-28.ru/dokumentatsiya-d3-28/minimonitor/

---

k31a_stud_os_KP-229607_N-19040.txt по-видимому, это обучающий «мат. пакет» для студентов.

Ещё некоторые программы, собранные (в том числе и составленные) мной по разным источникам:

call0_mpss-3-60-1988__bas.txt – программки, демонстрирующие работу Бейсика с внешними подпрограммами. Все необходимые пояснения приведены конце этого txt-файла.

---

demo_15ie-00-013__KP-50120_N-3546.txt – демонстрация «художественного» вывода текста на дисплей с использованием управляющих кодов дисплея в системе команд №1. В конце этого же txt-файла приведены все пояснения, в том числе о запуске.

---

print_screen__KP-2033_N-127.txt – программка, распечатывающая на ТПУ изображение экрана дисплея. В конце этого же txt-файла рассказано, как с этой программкой работать.

---

mix_25-09-2019.txt – смешанная «библиотека программок» в машинных кодах и на языках Бейсик, Фортран, Выстра. В ней 17 программок (см. пояснения в разделе 3), в том числе игра «23 спички». Эта библиотека служит для демонстрации работы имитатора с НМЛ. Наиболее серьёзные программы в ней – бейсиковские «HALL» и «DATA-SAVE TEST», работавшие в 1980-е годы на реальной Д3-28.

test_data-save_bas__blocks.txt – тест-файл для работы под управлением «DATA-SAVE TEST».

---

bega__5prg__bas.txt – «библиотека программок» на языке Бейсик. В ней 5 программок. В том числе – квази-игра «тараканьи бега»; она может использоваться для выявления особенностей вывода символов в разных режимах имитатора, в том числе для сравнительной оценки скорости вывода символов. В файле содержатся две записи программы «тараканьи бега». Первая запись называется BEGA-1, она сделана в Бейсике-132259. Вторая запись сделана в Бейсике-157107, она называется BEGA-2. Кроме того в этой «библиотеке» содержатся бейсик-программы: SINUS – построение графика синуса с задаваемым количеством точек (может использоваться для измерения условной скорости работы имитатора на разных ПК), ХАОС-1 – построение графика хаотической динамики (внутри программы есть инструкция; это математическая программка), КАТАЛОГ – список бейсик-программ на данной «ленте».

Запуск – только под управлением Бейсика: загрузка командой LOAD, старт командой RUN. По умолчанию подразумевается запуск в Бейсике-157107 при b4=0. Оказывается, при этом в программе BEGA-1 нормально воспроизводятся РУС-буквы. При запуске же программы BEGA-2 надо после набора RUN, но перед нажатием <Enter> нажать <Ctrl>, тогда текст выведется в регистре ВР РУС (либо перезапустить Бейсик-157107 в режиме с b4=1). Пояснения см. в разделе 5.

---

day_of_week__bas.txt – определяет день недели по задаваемой дате; пояснение есть в конце этого txt-файла
Запуск под управлением Бейсика.

---

search_sip__bas.txt – обнаруживает Синхро-Импульс Периферии (СИП), пояснения есть в конце этого txt-файла.
Запуск под управлением Бейсика

SF-test_bas.txt – тест С. Фролова. См. пояснения в Приложении 3. Запуск под управлением Бейсика.

---

test-basic-plan__2prg__plan-bas.txt – здесь 2 простейшие программки, записанные программой «Basic plan» для тестирования (см. пояснения в конце этого файла). ВНИМАНИЕ: эти 2 записи несовместимы с обычными Бэйсиками и не читаются ими.

---

game_1WAR_bas.txt – демонстрационная тест-программа с игровыми элементами для наглядной проверки взаимодействия машины, дисплея и клавиатуры в имитаторе; иллюстрирует некоторые приёмы применения машинных кодов в бейсик-программах пользователя. Пояснения см. в разделе 1. (Это «новодел», для имитатора; на реальной Д3-28 не покатит :-)

Запуск – только под управлением Бейсика: загрузка командой LOAD, старт командой RUN. В разных Бэйсиках быстродействие может быть разным. (На реальной Д3-28 будет жуткий тормоз ☺)

---

disp-1_bas.txt – программка, с помощью которой можно изучать вывод символов на дисплей путём применения машинных кодов в бейсик-программах. Пояснения см. в Приложении 3. Запуск только под управлением Бейсика.

---

ok__heat-capacity__bas.txt – две программки на «Бейсике», см. пояснения в конце этого файла.
ok – печатает цифрами «стерео»-картинку, которая изображает слово, состоящее всего из двух латинских букв.
best-ждите – вычисляет теплоемкость $C(T)$, см. раздел 9.

---

diakonov-87_bas.txt – здесь 9 программок для запуска под управлением Бейсика-157107: семь программок из книги В.П. Дьяконова "Справочник по алгоритмам и программам на языке бейсик для персональных ЭВМ" 1987, плюс пара самодельных программок на основе алгоритмов из этой книги. См. очень подробные пояснения в конце самого этого txt-файла.

---

game_luna__bas.txt – «Посадка на Луну», текстовая игровая программа, набрана по листингу в документации к Бейсику. Из числа «легендарных» программ :-) game_luna__red18-08-2022_bas.txt – – та же игра с более аккуратно набранными пробелами в текстовых сообщениях.

game_corolevstvo__bas.txt – «Королевство Эйфория», текстовая игровая программа.

game_casino__bas.txt – «Казино», текстовая игровая программа.

game_vojna__bas.txt – «Война», игровая программа; сочинена студентами под впечатлением от занятий на военной кафедре; похоже, чепуха полная :-)

Запуск: все эти игровые программы работают только под управлением Бейсика (пробуйте в первую очередь Бейсик с контрольной суммой 157107 или 132259).

---

bioritmy__bas.txt – тоже «легендарная» для пользователей Д3-28 программа – из «*Руководства программиста*» на Бейсике. Интересна она, конечно, не якобы биоритмами, а тем, как в ней организовано построение одновременно трёх синусоид на экране алфавитно-цифрового дисплея. Запуск – только под управлением Бейсика: загрузка командой LOAD, старт командой RUN.

Для правильного вывода графиков на экран здесь необходимо воспользоваться Бейсиком-157107 с b4=1 (см. раздел 5) и задать систему команд №1, сбросив в ноль разряды служебной строки дисплея нажатием клавиш <F8>, <F9>, <F8>. Тогда правильно отображается «нулевая линия» на графиках биоритмов, которая строится из символов I. Сначала печатается символ I, а затем он заменяется символом графика, если график должен пересечься с «нулевой линией». Интересно, что если бит-4 не установлен в единицу, то такой замены символов не происходит, и «нулевая линия» местами ломается: смещается направо.

1WAR_1_2WAR_1__frtrn5m.txt – файл, имитирующий «ленту» с библиотекой игровых демонстрационных фортранных программ и с тестом С. Фролова на Фортране.

В начале файла записан сам FORTRAN-5M; загрузка: <C>, <L>, <K> (КП=138177, код 0512 на шаге N=10414), запуск: <C>, <T>, <T> (без нажатий <S>).

Далее на этой «ленте» идут исходные фортранные тексты программ 1WAR и 2WAR, рабочие (т. е. транслированные в машинные коды) программы 1WAR и 2WAR, причём у всех этих программ номер версии равен 1. И затем – исходные тексты SF-T версий 1 и 2. В конце есть запись «конец библиотеки».

Запуск исходных фортранных программ (ИП) – под управлением Фортрана-5М (или Фортрана-85); они могут быть исполнены только после трансляции в машинные коды.

Запуск рабочих фортранных программ (РП): под управлением Фортрана-5М, либо – запуск с пульта машины (запускается основной блок из записи РП). Подробности см. в разделе 5.

---

frmj_inpx_inpu_outx_test__vystra-text.txt – здесь 6 программ: одна «в машинных кодах» (работает без загрузки языков высокого уровня), пять программ – исходные тексты на языке «Выстра» (запускаются эти исходные тексты только под управлением Выстры). Очень подробные пояснения см. в конце этого txt-файла. .

frmj_KP-38246_N-3445.txt – рабочая программа расчёта «фракталов» FRMJ, скомпилированная в Выстре на реальной машине Д3-28 в давние времена. Рабочие программы служат примером автономно работающих программ, созданных в Выстре. Запуск: <C>, <S> на пульте машины; загрузка Выстры не требуется. См. разделы 5 и 12, а также: info_5__prg_dla_compil__1-03-2020.pdf.

---

sip_sip-wyw-auto_sip-upr-auto__vystra-texts.txt – здесь 6 записей: 3 рабочие программки, полученные компиляцией в Выстре исходных программ, и 3 исходные программки на языке «Выстра». См. очень подробные пояснения в конце этого же txt-файла.

---

q-eq_ris-9_KP-543_KP-7765_KP-34321.txt – разные реализации программы вычисления корней квадратного уравнения по алгоритму из "Инструкции по эксплуатации" для Д3-28, см. там стр. 90-93, рис.9. Очень подробные пояснения приведены в конце этого txt-файла. (Вычисление корней квадратного уравнения это как бы «Hello World» для ЭВМ :)

---

display_codes_KP-1689_N-106.txt – программа выводит на дисплей байт-код, поступающий из терминала при каждом нажатии клавиши на клавиатуре терминала. Выводится символ (если клавиша символьная), и за ним байт-код. Если клавиша подаёт управляющий код, а не код символа, то этот код исполняется и выводится. Каждый код выводится в новой строке. Чтобы не было пустых строк, можно выключить в настройке служебной строки режим «авто ПС=ВК»:

9600 0000 0010 0000 0000 …

Запуск: <C>, <S> «на пульте машины»; затем переходим к просмотру дисплея.

---

klava_KP-434_N-25.txt – простейшая программка связи дисплея с машиной; см. раздел 2.

test_061_PEL2-240-001_KP-21038_N-1714.txt – в блоке 4 тестируется дисплей, см. раздел 2.

028-test_KP-73370_N-8921.txt – новый тест системы команд и ОЗУ (оцифровка **Виталия К**),
D3-28_test_028__KP-60000_N-5999.txt – старый тест системы команд и ОЗУ,
D3-28_test_017__KP-5895_N-527.txt – тест НМЛ, см. «*Инструкцию по эксплуатации*» Д3-28,
а также ниже: Приложение 3, раздел «Работа тест-программ в имитаторе».

---

y-test__KP-88_N-8.txt – демо-программка, поясняющая программирование в машинных кодах Д3-28; см. раздел 7. Запуск: <C>, <S>. Смотреть на экран дисплея не нужно; программка всего лишь выводит последовательные значения целых чисел на Y-табло машины. Для останова надо нажать <Enter> или <C>.

countdown_wang_ks-533_n-36.txt – аналогичное демо из репертуара Wang; см. пояснение в конце файла.

7-11-2017_tablo-XY-test__KP-637_N-36.txt – ещё одна демо-программка; она тоже лишь выводит цифры на табло машины. Запуск здесь иной: <C>, <T>, <S>. Останов обычный: <Enter> или <C> в режиме просмотра пульта машины.

cos_ris-11_KP-386_N-46.txt – учебная программка, выводит на табло машины значения cos(X), вычисленные двумя способами. См. в «*Инструкции по эксплуатации*» Д3-28 стр. 96-97. Запуск: <C>, набираем в регистре X желаемое число, нажимаем <S>, смотрим на табло результат вычисления cos(X). Затем <C>, набираем новое X, <S>, и т. д.

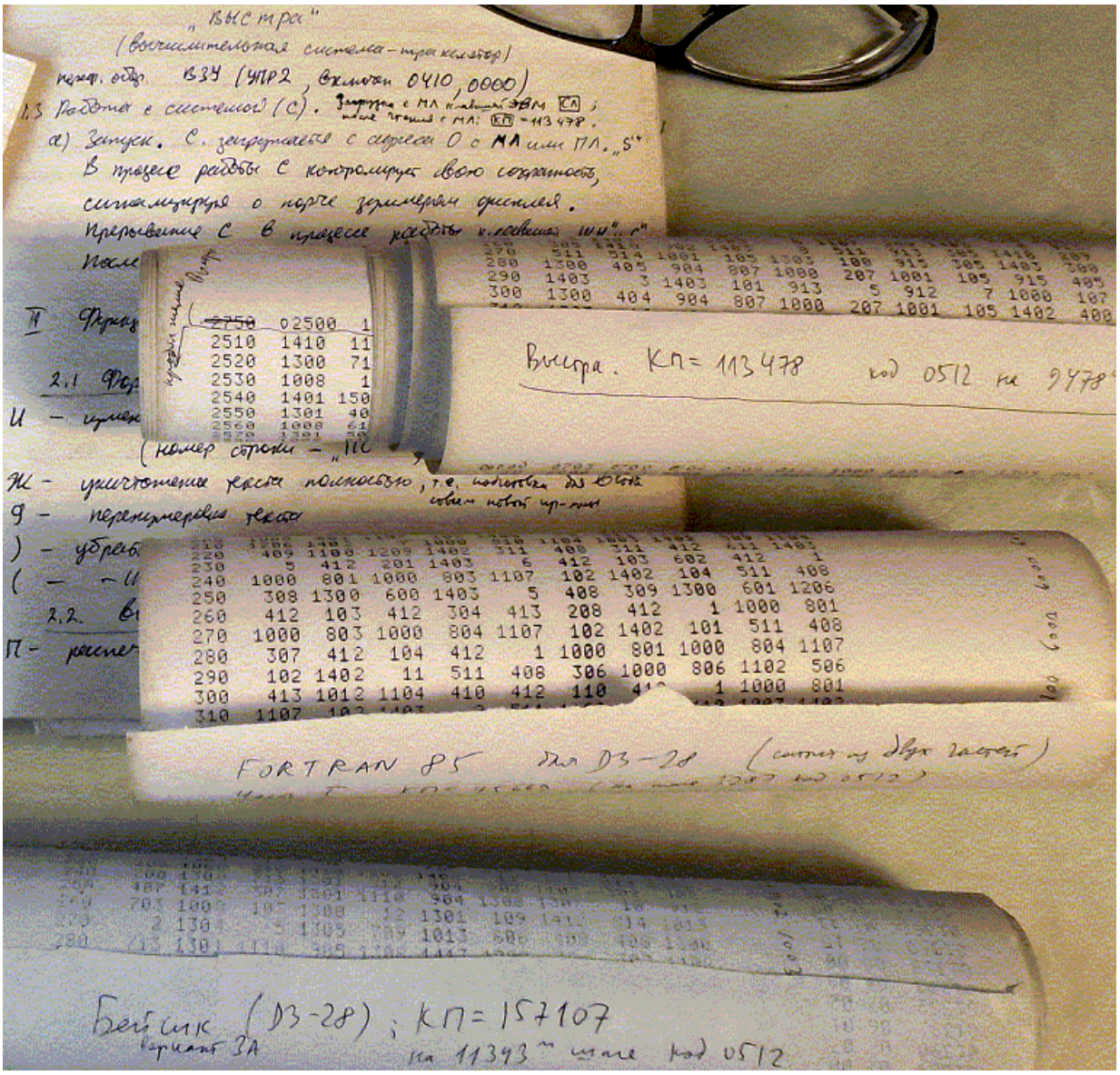
statistika__KP-7074__N-806.txt – пакет подпрограмм «Статистика» из комплекта ПО к Д3-28. Пояснения о запуске и о работе в имитаторе см. в конце этого же txt-файла. Подробности см. в «*Руководстве оператора 3.857.100 Д12*» (этот документ есть по ссылкам: http://retropc.org/Biblioteka_r_31.html#c254 (djvu-файл в «Библиотеке»), https://d3-28.ru/dokumentatsiya-d3-28/rukovodstvo-operatora-d3-28/)

---

rabochaja_kasseta.txt – пустой файл, имитирующий «рабочую кассету» для теста НМЛ; см. Приложение 3. Его можно использовать и для опытов с командами записи на НМЛ в других случаях. Конечно, подобных файлов (с желаемыми именами) можно и нужно создавать сколько угодно, по мере надобности.

---

raspechatchik_D3-28__KP-935__N-67.txt – программа для распечатки номеров шагов и байт-кодов программы из ОЗУ машины. Она работала с реальной машиной Д3-28 и позволила получить на ТПУ распечатки важных системных программ – Бейсика, Фортрана и Выстры:



В имитаторе распечатчик может пригодиться, например, для получения дампов сегментов ОЗУ с такими сложными системами, как ОС ВТ-МХТИ (см. также print_column__КР-782_N-57.txt).

Работа с распечатчиком (это один из нескольких возможных вариантов действий):

0. Включаем ТПУ (в имитаторе «ТПУ включается» ПК-клавишей <Home>).

1. Вводим в ОЗУ с ленты (или с пульта Д3-28 вручную, если это не слишком долго) ту программу, из которой надо распечатать часть или её всю.

2. Кнопкой КП проверяем контрольную сумму. Здесь и далее речь о кнопках на пульте Д3-28; их соответствие с ПК-клавишами в имитаторе указано в разделе 7.

3. Кнопкой В переходим в режим «Ввод», при этом индикатор покажет номер шага с кодом 0512 (END) программы в ОЗУ. И дописываем на следующих шагах код 0514 (это «пустая» команда GO) сколько-то раз – чтобы получился «круглый» номер шага, удобный для запоминания, с которого позже мы введём в ОЗУ машины свой распечатчик. Этот номер шага следует запомнить.

4. Кнопкой Р переходим в режим «Работа», и с пульта Д3-28 кнопками прямого кодирования подаём двухбайтовую команду 0413 0415; происходит засылка в регистр X начального адреса дальнейшей загрузки. (Напомню, что в имитаторе цифры 0413 0415, поскольку они представляют собой байт-коды команды, надо набирать с постоянно нажатой ПК-клавишей <Ctrl>.)

5. Ставим магнитную ленту с распечатчиком и загружаем его в ОЗУ командой 1202 с пульта машины. (В имитаторе цифры 1202 надо набирать с постоянно нажатой ПК-клавишей <Ctrl>.)

6. Прямым кодированием заносим в регистры R0 и R1 соответственно начальный и конечный адрес того участка ОЗУ, который желаем распечатать. С этой целью поочерёдно пользуемся режимами В и Р (в имитаторе – клавишами <V> либо <W> и <R>):

В: нажимаем НШ и вводим пять цифр (старшие из них могут быть нулями), это желаемый начальный адрес в десятичной системе; он сам собой оказывается в программном счётчике PC. (В имитаторе НШ это <N>).

Р: подаём двухбайтовую команду 1104 1500, это засылка PC в регистр R0 (в имитаторе цифры 1104 1500 набираем с постоянно нажатой ПК-клавишей <Ctrl>).

В: нажимаем НШ и вводим пять цифр = желаемый конечный адрес.

Р: подаём команду 1104 1501, тем самым конечный адрес засылается в R1. (в имитаторе цифры 1104 1501 набираем с постоянно нажатой ПК-клавишей <Ctrl>).

7. В: нажимаем НШ и вводим номер шага, на котором находится начало нашего распечатчика (этот номер мы запомнили раньше); тем самым мы подготовили стартовый адрес для запуска распечатчика.

8. Р: нажимаем клавишу S – старт; в имитаторе это клавиша <S>. Всё.

Распечатчик создаёт txt-файл, в котором левый столбец представляет десятичные значения адресов в ОЗУ с шагом 10, а в каждой полной строке содержатся 10 байт-кодов. Не следует пугаться пробелов, встречающихся в таком файле: дело в том, что имитатор, как и реальное ТПУ, не печатает нули, если они находятся в начале числа (так что, если, например, байт-код или номер состоит из одних нулей, то в ТПУ-распечатке он выглядит сплошным пробелом).

print_column__KP-782_N-57.txt – ещё один распечатчик, он выводит на ТПУ столбиком байты из ОЗУ (дамп). Загружается на свободное место в ОЗУ, запуск с этого места: S. Цифра 1 на X-табло приглашает набрать в X начальный адрес дампа, набираем, нажимаем S. Цифра 2 приглашает набрать конечный адрес дампа, набираем, нажимаем S. ТПУ надо включать до того.

---

11. Приложение 1: команды Фортрана-85 и -5М

ОПИСАНИЕ ТРАНСЛЯТОРА FORTRAN-85
ДЛЯ МИКРО-ЭВМ «Д3-28»

Это не сам официальный документ, а пересказ рукописного конспекта (из тетради бывшего студента, а затем уважаемого сотрудника, нашего коллеги А. В. Березина) некоего документа, составленного в те давние времена; так что, здесь могут быть пропуски, опечатки и неточности…

1. *Запуск транслятора*

Транслятор FORTRAN-85 состоит из двух частей (блоков), записанных на МЛ. 1-я часть содержит системную библиотеку и драйверы работы со стандартными периферийными устройствами. 2-я часть – собственно транслятор. Для работы с ним необходимо:

1.1 Загрузить с МЛ первую часть. Проверить КП и нажать клавишу С (Сброс) на пульте Д3-28.
1.2 Нажать два раза клавишу ▷ на пульте Д3-28.
1.3 ЭВМ выведет на экран дисплея: «транслятор загружать (Д / Н) ?» Если вторая часть не нужна, то надо нажать Н на клавиатуре дисплейного терминала. Если вторая часть нужна, то нажать Д; при этом ЭВМ сама ищет на магнитной ленте вторую часть и загружает её в память.
1.4 Далее ЭВМ запрашивает тип дисплея: «Дисплей – 01 ». Необходимо ввести цифру 3 для дисплея 15-ИЭ-00-013 (Фрязино) или цифру 7 для дисплея 017 (Винница).
1.5 ЭВМ выведет на дисплей: READY.
1.6 Нажать пробел, машина выведет: FORTRAN-85.
1.7 Для ввода программы нажать пробел; ЭВМ напечатает SEGMENT, далее нажать M(AIN PROGRAM) для набора программы или S(UBROUTINE) для подпрограммы. Для выхода на операционную систему транслятора нажать S в крайней левой позиции курсора.
1.8 RUN набранное после READY до загрузки программы вызывает начальный диалог, в котором задаются спецификации Фортрана:

2. *Начальный диалог по RUN*

В начальном диалоге можно получить «каталог МЛ», а также изменить следующие параметры (в скобках указаны значения по умолчанию. Если параметр не надо изменять, то следует нажать пробел):

ЛИСТ количество строк в листе (24)
СТРОКА количество символов в строке (75)

АИП адрес исходной программы (14000)
СЕГМентация разрешена (1)
ТАБ если 0 – начало, 1 – дальше (имеется ввиду продолжение этого диалога)
МЕТ количество меток (50)
GOTO количество операторов перехода (50)
ИД количество идентификаторов (100)
DO количество вложений циклов (8)
SUB количество подпрограмм (8)
АТР адрес транслятора распечатывается после прохождения диалога.

Вывод информации на экран дисплея приостанавливается после вывода 24 строк: печатается двоеточие; для продолжения вывода следует нажимать пробел.

3. *Работа с транслятором*

Одним из трёх режимов работы транслятора является операционная система (ОС) транслятора. Её признаком служит символ S (от слова System) в крайней левой позиции экрана. Выход в ОС делается автоматически после окончания набора программы, или принудительно – в последнем случае надо напечатать символ S в позиции 1-го останова. ОС транслятора обеспечивает:

– получение листинга программы на экране дисплея или АЦПУ,
– перенумеровку операторов,
– продолжение ввода программы,
– трансляцию программы,
– работу с МЛ,
– переход в режим редактора программы,
– передачу управления на начало транслятора.

3.1 Получение листинга: S Л

После нажатия русской Л распечатывается программа из памяти, на «языке» транслятора. Если листинг затребован одновременно с трансляцией, то дополнительно выводится информация о распределении операторов программы в памяти ЭВМ и указывается базовый адрес данных (BD).

Программа на языке транслятора, как правило, сохраняется в памяти ЭВМ и после трансляции. Поэтому при сбое, который не привёл к выключению ЭВМ, можно получить листинг исходной программы и продолжить её ввод, нажав русскую «В» при положении курсора в крайней левой позиции на экране дисплея. Как правило, листинг необходим и после коррекции, т.к. во время листинга операторы получают новые номера и анализируются идентификаторы переменных и массивов. Для листинга (в т.ч. при трансляции) на АЦПУ надо подать директиву А. Пример:
S АЦПУ S Л – печать листинга (разумеется, при этом ТПУ надо не забыть включить :).
S ТР К1 Л? Л – текст программы выводится на экран во время трансляции.

Если Фортран при листинге обнаруживает ошибку в программе, то он прекращает листинг и переходит в режим редактора (обозначается звёздочкой *). Можно получить полный листинг (в том числе программы с ошибками) или частичный, если в режиме АЦПУ включить режим Редактор. Команды:

S АЦПУ S Р*Л К *количество_строк* Н *с_какого_номера_начать_(можно_с_нуля))*

3.2 Перенумеровка операторов

После коррекции в пределах одной программной единицы может образоваться несколько операторов с одним номером. Чтобы номера операторов шли в порядке возрастания, надо выполнить команду: S H

После её завершения печатается общее количество операторов, имеющихся в программе.

3.3 Продолжение ввода программы: S B

Продолжение ввода после сбоя возможно без чтения с МЛ, если ЭВМ не выключалась. Следует провести листинг, выйти на ОС транслятора, и после символа S нажать русскую В.

В режиме ввода применимы дополнительные команды:

I в позиции 1-го останова – вызывает таблицу определённых к данному моменту переменных и массивов.

A – позволяет узнать адрес набираемого оператора в памяти ЭВМ.

K – коррекция набранного оператора после сообщения об ошибке (ERR).

3.4 Трансляция программы: S TP KN Л

где N – «номер канала», с которым будет работать ЭВМ; он записывается на шаге с номером 00022, и при необходимости его можно будет изменить, остановив ЭВМ.

Если N = 0, то после принудительного останова или сбоя в работе программы и повторном её запуске выход будет производиться на начало рабочей программы.

Если N не равно 0, то после повторного запуска выход будет производиться в режим READY, откуда можно войти в ОС транслятора для коррекции исходной программы.

Если нужен листинг, то после запроса Л: вводится символ Л и нажимается пробел. Если листинг не нужен, то сразу нажимается пробел, после чего начинается трансляция. В ходе трансляции на дисплей выводятся сообщения об ошибках и адреса, на которые транслируются операторы исходной программы.

Если в транслируемой основной программе есть обращения к подпрограммам, то об этом выдаются сообщения «SUBROUTINE имя» в порядке объявления подпрограмм в основной программе. Если порядок расположения подпрограмм отличается от порядка их объявления, то имена подпрограмм распечатываются в соответствии с их объявлением в основной программе, и в целом рабочая программа работает правильно. Если при трансляции требуемых подпрограмм нет в памяти ЭВМ, то их нужно ввести либо с терминала дисплея, либо с МЛ.

Признак окончания трансляции – печать указателя служебной области памяти BD. При необходимости можно ввести другое значение BD и нажать пробел. Если изменять его не нужно, то просто нажимается пробел, и ЭВМ автоматически выходит на ОС транслятора. Исходную программу и оттранслированную программу можно записать на МЛ; последнее позволяет в дальнейшем пользоваться программой без транслятора.

3.5 Работа с МЛ

S МЛ B – перемотка вперёд
S МЛ H – перемотка назад

Если надо записать программные единицы на новую ленту, то используются команды:

S МЛ ЗЛ БН0

И
П
S МЛ ЗП Р BN
Ф
С

где:

И – запись исходной программы
П – запись подпрограммы
Р – запись программы в кодах (рабочей)
Ф – запись фрагмента программы
С – запись сегмента

Команда «S МЛ ЗЛ БН0» указывает на то, что библиотека записей начнётся с нуля. Буква «И, П, Р, С» выбирается в зависимости от вида записываемой программной единицы. Ответом ЭВМ на эти действия будет запрос В, на который надо ввести номер «N» версии – любое число от 1 до 99 (подразумевается, что под одним и тем же именем могут храниться различные версии программ и подпрограмм с разными номерами версий. Имя это 4 символа, но важны только первые 2). Запись на МЛ начнётся после нажатия пробела; ЭВМ делает трёхкратную запись.

Можно записать на МЛ фрагмент программы, т. е. часть программы, не имеющую оператора END. При считывании с МЛ такой фрагмент подстыковывается к концу вводимой программы. Это позволяет часто используемые куски программ не вводить каждый раз с клавиатуры, а считывать с МЛ.

Возможна запись на МЛ уже оттранслированной программы как «сегмента», что позволяет в дальнейшем работать без загрузки второй части FORTRANa. Запись программы в виде сегмента возможна только сразу после её трансляции, до её запуска. Если различные сегменты должны работать с одним и тем же массивом данных, то они должны быть записаны с одинаковым указателем BD.

Чтобы программа могла работать в ЭВМ автономно, её следует записать как «рабочую»; при этом на МЛ сначала запишется 1-я (системная) часть FORTRANa, а затем оттранслированная программа.

Подпрограмма может быть записана на МЛ только в том случае, если она создаётся отдельно от основной программы. Если подпрограмма вводится сразу после основной программы, то она будет записана на МЛ вместе с основной программой.

После окончания записи программной единицы выводится количество шагов в ней и контрольная сумма; по этим параметрам при необходимости можно найти программу на МЛ без помощи транслятора.

Если на МЛ уже есть записи, которые следует сохранить, то в первой команде указывается число оставляемых на МЛ записей. В этом случае запрос номера версии записываемой программы поступит только после пропуска указанного числа записей.

Если после записи программных единиц работа с МЛ больше не планируется, то следует ввести команду:

S МЛ ЗП КБ

При этом на МЛ будет записана метка (три раза код 0000 0512), означающая «Конец Библиотеки».

Команды чтения программ или подпрограмм с МЛ:

И
П
S МЛ ЧТ Р имя BN
Ф

где N – номер версии программной единицы с указанным именем; после нажатия пробела начинается её поиск. Признаком того, что нужная запись найдена, является печать её параметров, а также символов:

! – если запись считана правильно,
- – если при чтении был сбой,

и после этого ЭВМ выходит на ОС транслятора. Если все три экземпляра записи считаны со сбоем (напечатались три символа -) то следует попытаться повторить чтение. Если напечатался хотя бы один знак !, то программа есть в памяти ЭВМ и с ней можно начинать работать.

Прочитать с МЛ программу в рабочих кодах можно либо с использованием транслятора, либо без него. Первый способ возможен, если программа была записана как «сегмент». В этом случае надо загрузить 1-ю часть транслятора; в ответ на запрос о загрузке второй части нажать русскую букву Н; после того, как ЭВМ выведет «READY», нажать S(EGMENT); и на запрос В указать номер версии программы. ЭВМ ищет программу с начала ленты и в случае правильного считывания автоматически запускает её. Если искомый сегмент не удалось считать, то лента перематывается в начало и читается повторно, и т.д., причём выйти из такого режима можно будет только принудительным остановом ЭВМ.

Печать каталога записей:

S МЛ К – этой командой включается просмотр ленты от начала и до метки «конец библиотеки» (если она есть) или до конца ленты (в последнем случае просмотр МЛ остаётся незавершённым; этой ситуации надо стараться избегать, т.е. не забывать записывать метку КБ).

Ещё одна команда формирования библиотеки записей на МЛ:

S МЛ Б НН M N K …

где M, N, K, … – номера записей, которые требуется сохранить на МЛ. Последний номер – ноль. По этой команде указанные записи переписываются последовательно с начала ленты.

3.6 Выход в режим «редактор»:

S Р(ЕДАКТОР)
*

Указанная в скобках часть команды допечатывается машиной автоматически. Кроме того здесь происходит перевод строки и печать символа *. Выход в редактор происходит и при листинге программы, содержащей ошибки. Для возврата в ОС транслятора надо нажать любую неслужебную клавишу. Команды редактора:

* Л KN1 HN2 – листинг части программы, где N1 – количество операторов,
N2 – номер оператора, с которого следует печатать листинг.

При таком листинге не ведётся синтаксический контроль. (Полезный факт: в редактор можно зайти после команды АЦПУ, и тогда можно будет получить распечатку листинга программы, даже если в ней есть ошибки (препятствующие листингу вне режима редактора)).

* ОП HN1 – коррекция оператора с номером N1. Эта команда автоматически распечатывает метку и служебное слово, если они есть в корректируемом операторе. Просмотр оператора ведётся клавишей → (вперед) и ← (назад).

Ненужные символы удаляются клавишей: ←| дисплея 013 и < дисплея 017.

Чтобы вставить символы, нужно сначала раздвинуть текст клавишей:
|→ дисплея 013 и > дисплея 017.

Заканчивается коррекция оператора либо его просмотром клавишей → до конца, либо клавишей «ПС» из любого места.

Изменить номер метки или служебное слово можно только полным удалением данного оператора с последующим повторным вводом нужного слова и метки. После удаления или ввода оператора необходимо делать перенумеровку операторов.

Если в программе нет указываемого номера оператора, то ЭВМ запрашивает номер оператора повторно.

* У KN1 HN2 – удаление N1 операторов, начиная с оператора с номером N2. После этого нужно выйти в ОС транслятора для перенумеровки. (Полезный факт: оператор с номером 0 это строка с именем программы. Удаляя её и заново вставляя, можем имя программы изменять.)

* В KN1 HN2 – вставка N1 операторов, N2 – номер оператора, перед которым делается вставка. (Т.е. если надо изменить оператор, то какой номер удалили, такой же и вставляем.)

Вставка операторов производится аналогично вводу программы. Ввод каждого оператора завершается нажатием клавиши «ВК». Если допущена ошибка, то можно повторить ввод (до нажатия «ВК») нажав «ПС».

После ввода требуемого количества операторов нужно выйти в ОС транслятора и произвести перенумеровку. Если обнаружится неправильное служебное слово, то вставка отменяется, вновь набранные операторы игнорируются, и нужно повторить всю команду вставки операторов.

3.7 Передача управления в начало транслятора.

Для перехода из ОС транслятора в его начало надо нажать клавишу / , при этом будет выведено сообщение «READY». Отсюда можно запустить оттранслированную программу командой RUN, или вернуться в ОС транслятора, нажав пробел и затем S.

В любом месте работы транслятора можно принудительно остановить ЭВМ и повторно её запустить (клавишей сброс С и S) с пульта ЭВМ. При этом указатель ВР = 0. Если регистр ВР не сброшен в ноль, то его можно обнулить следующими командами с пульта ЭВМ Д3-28:

```
0413  1000    (CLR R00)
0413  0600    (MOV R00, BP)
```

Если нарушилась связь ЭВМ с дисплеем, то следует с пульта ЭВМ подать команды выбора дисплея:

```
1302  1507    (MOV 1507, S02)
1303  1515    (MOV 1515, S03)
0412  1407    (OUTOWS)
```

4. *Набор программы в трансляторе*

Для набора каждого оператора предусмотрены три позиции останова курсора:

1-й останов, крайнее левое положение курсора. Здесь возможен выход в ОС транслятора клавишей S. Если выход не нужен, то нажимаем пробел; ЭВМ напечатает номер строки и остановится в позиции второго останова.

2-й останов – позиция для ввода метки или символа комментария С. Если это не нужно, то нажимаем пробел, и ЭВМ остановится в позиции третьего останова.

3-й останов – для печатания операторов. При использовании операторов цикла выводится столько символов ^ , сколько имеется уровней вложения циклов. При этом в любом случае для набора оператора присваивания нужно сначала сделать ещё один пробел.

Имя основной программы либо подпрограммы должно состоять из четырёх или более символов. Однако ЭВМ будет различать эти имена только по двум первым символам.

5. *Типы переменных*

Имеется два типа переменных: двухбайтовые целые со знаком, и восьмибайтовые вещественные со знаком. Допускается следующее неявное описание переменных:

I, J, K, L, M, N – переменные целого типа (INTEGER,м аксимальное значение абсолютной величины переменной типа INTEGER не должно превышать 32767). Переменные, имена которых начинаются с остальных букв, т. е. A, B, C, … , P, R, … – вещественные (REAL).

Для явного описания типа переменных служат операторы INTEGER и REAL.

6. *Описание массивов*

Пример описания массивов:

DIMENSION A[1,2,K], B[M]

Максимальная размерность массива: 3. Элементы массива располагаются в памяти так, что сначала изменяется правый индекс; в случае двумерного массива это означает, что элементы матрицы располагаются по строкам. Допустимо выражать граничные значения индексов переменными целого типа (M, K, …), но тогда им предварительно должны быть присвоены числовые значения. Массивы могут быть также описаны операторами INTEGER, REAL, COMMON

Имена массивов, как и имена программ ЭВМ различает по первым двум символам.

7. *Стандартные функции FORTRANa для Д3-28*

SIN( ) синус
COS( ) косинус
TAN( ) тангенс
ASN( ) арксинус
ACN( ) арккосинус
ATN( ) арктангенс
HSN( ) гиперб. Синусё
HCN( ) гиперб. Косинус
EXT( ) показательная
SQR( ) квадрат
QRT( ) корень квадратный
INV( ) обратная величина
NEG( ) изменение знака

HTN( ) гиперб. тангенс
AHT( )
AHC( )
RAD( ) радианы
DEG( ) градусы
LOG( ) логарифм десятичный
LGT( ) логарифм натуральный
EXP( ) экспонента
ABS( ) модуль
CPI( ) const pi
RND( ) генератор случ. чисел от 0 до 1
SEG( ) вызов сегмента с МЛ

8. *Ключевые слова FORTRANa для Д3-28*

CA(LL) вызвать
COM(MON) общий
CON(TINUE) продолжить
DI(MENSION) размерность
DO выполнить
EN(D) конец
EL(SE) иначе
F(ORMAT) формат
G(O TO) перейти к
IF если
IN(TEGER) целое
M(AIN PROGRAM) основная программа
READ читать
REAL вещественное
RET(URN) вернуться
S(UBROUTINE) подпрограмма
WR(ITE) написать

K(:) оператор «код»
C(:) комментарий

9. *Оператор безусловного перехода*

GOTO M где M – метка оператора, которому передаётся управление. Меткой служит целое число от 1 до 999 без знака или идентификатор целочисленной переменной (его желательно объявить в списке INTEGER).

Оператор GOTO M работает только внутри основной программы и не работает внутри подпрограммы. Но он может использоваться для возврата из подпрограммы в основную программу: если затем в основной программе встречается оператор RETURN, то он возвращает управление в программе на следующий шаг после обращения к подпрограмме (CALL …)

Метки локализованы в тех программных единицах, где они установлены; поэтому в программе и подпрограмме метки могут повторяться.

Метками не могут быть числа 13 (это в десятичной форме код «ВК») и 92 (это в десятичной форме код 0512 команды END)

10. *Оператор присваивания*

P = арифметич. выражение или число,

где P – переменная. Перед именем переменной надо обязательно набрать пробел; он служит оператором присваивания и набирается, как и все операторы, в 3-ей позиции. Все переменные, которые используются в правой стороне равенства, должны быть предварительно определены. Если встретится ранее не указанная переменная, то транслятор сообщит об ошибке и не примет такого ошибочного оператора. Поэтому, если переменная ещё не имеет числового значения и ей не присваивается числовое значение данным оператором присваивания, то она обязательно должна быть описана в операторе задания типа переменных.

В арифметическое выражение могут входить обозначения стандартных функции с аргументом в круглых скобках.

11. *Оператор условного перехода арифметический*

IF(арифм. выражение)M1,M2,M3

Если выражение вскобках < 0, то переход на метку M1,
= 0, то переход на метку M2,
> 0, то переход на метку M3.

Все указанные в операторе метки должны обязательно встретиться в программе.

12. *Оператор условного перехода логический*

```
IF(Выражение1 Знак Выражение2)
……….
ELSE
……….
END
```

«Знак» набирается одной из клавиш > < = #

Если альтернатива оператору, указанному в строке под IF, не нужна, то ELSE может отсутствовать.

13. *Оператор цикла арифметический*

DO M I=K1,K2,K3

где: M – метка оператора CONTINUE,
I – переменная цикла,

K1 – начальное значение переменной цикла (или арифм. выр. для него),
K2 – конечное значение переменной цикла,

K3 – шаг изменения переменной цикла,
(если шаг равен единице, то K3 можно не указывать).

Конец тела цикла обозначается оператором CONTINUE со своей меткой.

Для каждого цикла, вложенного в цикл, должен присутствовать свой оператор CONTINUE со своей меткой.

Значения переменной цикла I не должны «промахнуться» мимо K2, а иначе цикл не прекратит работать. Переменные цикла локальны, т.е. они определены только в цикле и не существуют за его пределами. Если они нужны и вне цикла, то их необходимо описать в операторе описания переменных DIMENSION или COMMON.

14. *Оператор цикла логический*

```
WHILE(Выражение1 Знак Выражение2)
…………………………………………….
  Выражение1 = ……………………………
……………………………………………..
END
```

Пример:

```
   A=5
WHILE(A<10)
   A=A+0.1
…………
END
```

Оператор цикла логического существенно отличается от арифметического тем, что переменная логического цикла может быть типа REAL.

15. *Оператор ввода-вывода*

```
READ(K,M)A,B,C,…
WRITE(K,M) A,B,C,…
```

где: K – номер устройства,
M – метка оператора формата; K и M – целочисленные переменные.

Возможные значения K:
0 – дисплей, 1 – печатающее устройство, 2 – магнитофон,
3 – перфоратор, 4 – фотосчитыватель,
или K = десятичный номер регистра управления.

Признак окончания ввода – нажатие клавиши «ПС» или « , »

Если при вводе числа допущена ошибка, а клавиша «ПС» еще не нажата, то надо:
1) для дисплея 017 нажать клавишу / столько раз, сколько цифр необходимо затереть, чтобы ввести на их место новые цифры;
2) для дисплея 013 клавишей ← перевести курсор назад и ввести новые цифры.

Для вывода комментариев на печать необходимо указать переменную вывода, например:

```
      WRITE(0,M)A
M     FORMAT(/,'текст', M1)        переменная в формате M1 не печатается.
```

16. *Оператор формата*

```
M     FORMAT(список параметров формата, разделенных запятыми)
```

/ – перевод каретки,
^ – перевод строчки для печати,
IN – целочисленный формат (где N – количество выводимых разрядов числа),
FN,K – вещественный формат: N – количество разрядов перед запятой,
K – количество разрядов после запятой.
E – формат числа с указанием порядка,
M1 – машинный формат.

Для ввода числа лучше указывать формат E, т.е. предполагать число с порядком.

Для вывода текста его заключают в апострофы: '…'

Указатель кратности вывода перед выражением в списке параметров формата может быть целочисленной переменной, лучше объявить её в перечне INTEGER.

Для вывода массива по элементам строк возможна следующая организация оператора формата:

```
      …
      DIMENSION AB[2,3],K
        K=3
      WRITE(0,5)AB
  5   FORMAT(/,K,F1,0)
      END
```

Тогда массив AB распечатается следующим образом:

1.	2.	3.
4.	5.	6.

17. *Оператор END*

После ввода оператора END производится контроль программы и проверяется наличие меток переходов. При их отсутствии выводятся сообщения типа:

G 5
G 100 и так далее.

18. *Оператор подпрограммы*

SUBROUTINE имя (A,B,…)
…… операторы ……
END

где A,B,…– формальные параметры, входные и выходные.

Если формальным параметром является идентификатор массива, то размерность не пишется, а ставится знак # после имени. Такой массив можно описать в подпрограмме, например: SUBROUTINE(N,M0#,MS#).

Если в подпрограмме используются другие величины, то они должны быть описаны в основной программе, например COMMON(X[K],Y), и объявлены в том же порядке в подпрограмме оператором COMMON(R,Q) без указания размерности (в данном примере внутри подпрограммы идентификатор R обозначает массив X, а идентификатор Q – переменную Y).

Если ввод подпрограммы производится после набора основной программы, то после сообщения транслятора «SUBROUTINE имя» необходимо нажать «ВК», нажать S в позиции ключевых слов, допечатать заголовок подпрограммы и продолжить ее ввод.

Заканчивается подпрограмма оператором END. Оператор RETURN не обязателен.

19. *Оператор обращения к подпрограмме*

CALL имя (A,B,C,…)
где A,B,C,… – фактические параметры; фактические и формальные параметры должны соответствовать друг другу по типу, количеству, порядку расположения. Фактический параметр не может быть элементом массива.

20. *Оператор подпрограммы-функции*

Имя подпрограммы-функции состоит только из одной буквы; за ней следует список фактических переменных в скобках. Пример:

A=B+C*S(PP,TT,W3)

где S – имя подпрограммы-функции.

После окончания ввода основной программы ввод подпрограммы-функции запрашивается так же, как и ввод обычной подпрограммы.

Вычисленное значение подпрограммы-функции находится в регистре X ЭВМ, и для того чтобы его не потерять при дальнейших вычислениях, можно его присвоить какой-либо переменной или выходному параметру в списке фактических параметров.

21. *Оператор описания общих блоков*

COMMON A,B,C,…
где A,B,C,… – идентификаторы простых переменных или массивов.

Оператор COMMON используется в тех случаях, когда данные, необходимые для работы программной единицы, находятся в другой программной единице. Поэтому оператор COMMON должен присутствовать как в основной программе, так и в подпрограмме. При этом общие данные будут занимать одну и ту же область памяти.

Оператор COMMON может содержать информацию о размерности заново описываемого массива, но не должен содержать размерности ранее описанных массивов.

22. *Знаки операций*

+ сложение	** возведение в степень
– вычитание	
* умножение	& логическая операция «и»
: или / деление	! логическая операция «или»

Указанные логические операции выполняются над младшими байтами целочисленных переменных (типа INTEGER).

23. *Оператор «Код»*

Оператор «Код» позволяет вставлять последовательность машинных кодов Д3-28 в текст программы. Например:

K:C5 60, 11 01, …

где C5 = (1205) – внутренняя подпрограмма Фортрана для поиска массива с адреса 60 (полученного по директиве I в режиме ввода программы). 11 = (0101) – также внутренняя подпрограмма, она определяет размерность массива с записью в регистр R4 адреса элемента массива с номером 01 и в регистр R7 адреса конечного элемента массива.

Адреса переменных и массивов выводятся директивой I в режиме ввода программы; они начинаются с номеров (относительно BD):

20 – для переменных типа REAL (это десятичный номер первой восьмибайтовой десятичной ячейки в отведённой для вещественных переменных области памяти),

60 – для массивов,

80 – для переменных типа INTEGER.

Машинные коды разделяются пробелами или запятыми; любые другие символы воспринимаются как начало комментария.

24. *Распределение памяти для Фортрана в Д3-28*

Адрес (номер шага)	*Содержимое памяти*
0	Служебная библиотека
02601	Оттранслированная программа
(BD)	Служебная зона (80 байт)
(BD) + 80	40 ячеек INTEGER (80 байт)
(CD 20) (BD) + 160	40 ячеек REAL (320 байт)
(CD 60) (BD) + 480	40 определителей массивов (320 байт)
(BD) + 800	Массивы
14000	Исходная программа
ATP	Транслятор
	Стек
32512	Служебная зона ОЗУ Д3-28

25. Операции с матрицами

(но, по-видимому, они не реализованы в имеющемся у нас Фортране, … мне не удалось их испытать)

Матричные операторы работают только с массивами типа REAL.

A = имя массива – матричное присваивание.

B = A’ – операция транспонирования матрицы A.

A = A¤ – операция обращения матрицы.
C = A*B – операция умножения матрицы A на переменную B.

C = C + B – операция сложения матрицы A и матрицы B.

C = C – B – операция вычитания матрицы B из матрицы A.

26. *Операторные скобки*

BEGIN
……
END

Переменные и массивы, определенные в операторных скобках определены только внутри этой части программы. После ввода оператора END проводится контроль программы и проверяется наличие меток для переходов, которые также локализованы внутри операторных скобок и могут совпадать с метками в основной программе.

27. *Оператор комментария*

C – во второй позиции. После ввода этого символа можно вводить любой пояснительный текст (комментарий). При трансляции комментарий игнорируется и не занимает места в оттранслированной программе.

28. *Запись данных на магнитную ленту*

Запись имеет следующую структуру:

1	2	3	50 байт	1	2	3	41байт	
заголовок			разделитель	информация			буфер	

Заголовок содержит номер записи, размер в байтах, тип записи (M или S: массив или переменные). Номер записи при считывании и записи запоминается в регистре T2. Регистр T2 обнуляется только при загрузке транслятора. Если номер записи равен (T2)+1, то по команде WRITE(2,M)XX производится запись массива данных XX с текущего положения магнитной ленты по формату M. Если номер записи больше или меньше, чем (T2)+1, то буфер при записи не записывается, а считывается, поэтому для записи данных необходимо обеспечить такое положение ленты, что транслятор мог бы прочесть предыдущий номер записи.

Перемотка ленты происходит только по формату M1.

Если T2 не равно нулю, а лента новая, то надо ввести кодовое обнуление T2. Например:

```
      READ(0,5)J
    5 FORMAT(/,'Укажите номер записи на МЛ',E)
      IF(J=0)
      K: 4D A0 EC 02
      END
```

Пример записи данных:

```
      INTEGER (или REAL) X1[N],Y2[N],Z8[K]
      WRITE(2,10)X1,Y1,Z8
   10 FORMAT(2M5,M7)
```

При использовании в спецификации формата повторителя размерность массивов должна быть одинаковой. В данном примере массивы X1, Y2, Z8 получат номера записей на МЛ соответственно M5, M6 и M7.

29. *Счет по программе после трансляции*

READY (чтобы появилось это сообщение при работе в ОС, надо нажать /)
R(UN) – счет по программе

30. *Ввод-вывод на перфоратор*

Ввод-вывод на перфоратор аналогичен работе с МЛ; пример:

S ПЕР ЗП И В1

31. *Организация работы с прерываниями*

Фортран допускает работу с двумя уровнями внешних прерываний:

1-й уровень: ПР1, ПР2, ПР4;
2-й уровень: ПР8.

Разрешение внешних прерываний задается, как обычно, установкой соответствующих разрядов регистра маски. Номер метки, на которую передаётся управление при поступлении сигналов прерывания, равняется первому и второму оператору присваивания соответственно для 1-го и 2-го уровня прерывания.

Для правильной работы с прерываниями возврат из прерывающей программы должен осуществляться командой RTI (1211), с помощью кодовой вставки: K:CB

Выход из прерывающей программы осуществляется командой псевдовозврата RTII (1210), также при помощи кодовой вставки. При этом следует произвести запрет всех прерываний: K:CA, D0 00,4D 70

==========

Команды Фортрана-5М

Это не сам официальный документ, а перепечатка с моего очень краткого рукописного конспекта некоего документа; так что здесь могут быть пропуски, опечатки и неточности…

ОПИСАНИЕ ТРАНСЛЯТОРА FORTRAN-5M
ДЛЯ МИКРО-ЭВМ «Д3-28»

Обозначения: зелёные буквы – то, что выводит на дисплей ЭВМ Д3-28; обычные буквы – то, что вводит программист с клавиатуры терминала 15ИЭ-00-013.

Режимы РУС / ЛАТ машина устанавливает сама.

Запуск системы

После загрузки с МЛ (как обычно, командами С, СЛ на пульте машины, и после проверки контрольной суммы командами КП, С) – надо нажать два раза клавишу с треугольником:

1. ▷ ▷ – ЭВМ выведет на дисплей букву R и включит режим РУС.

2. RЗ – переход (необязательный) к начальному диалогу. Там можно задать длину листа, строки, сегментацию, адрес исходной программы, …

3. RТ – ЭВМ выведет FORTRAN-5M и сделает перевод строки. Если теперь нажать S, то попадём «в операционную систему»; и тогда можно загружать программку с МЛ, записывать на МЛ, редактировать, распечатывать программку – всё как уже рассказывалось для Фортран-85.

RС или RД – счёт по программе.

RМ *пробел* N *число* жмём ПС – происходит перемотка МЛ в начало, счёт по программе.

Счёт по программе осуществляется, если она уже есть в памяти машины и оттранслирована.

Ввод программы или подпрограммы:

RТ *пробел* MAIN PROGRAM *имя*

или:

RТ *пробел* SUBROUTINE *имя*

Имя должно содержать 4 символа, но идентификация будет осуществляться только по первым двум символам. При записи или чтении на МЛ будет запрашиваться также номер версии программы; имя и номер версии, введённый при записи на МЛ, желательно помнить, чтобы в дальнейшем указывать их для чтения с МЛ именно нужной нам программы.

Завершение ввода строки при наборе программ – клавишей ВК.
Отказ от набранной строки (например, из-за ошибки) – клавишей ПС (если не нажали ВК).

Текст программы в каждой строке размещается по трём позициям (т.е. всё, как в Fortran-85):

Номер_оператора. Метка (или пробел). Оператор.

Перемещение курсора в строке от позиции к позиции – клавишей «пробел».
В одной строке допускается только один оператор.
Для оператора END метка недопустима.

Перед строкой с оператором, исполняющим действия с переменными, они должны иметь числовые значения (как и в Бейсике).

Операторы:

1. Можно записать в строку комментарий:

   Номер_оператора. C : текст_комментария.

2. Описание переменных и массивов:

   Номер_оператора. . INTEGER A, B, C
   Номер_оператора. . REAL D, E, X[2,2], Y[3,B]

3. Оператор присваивания – пробел:

   Номер_оператора. . *пробел* A=B+C

4. Операторы арифметических действий обозначаются стандартно:

   + сложение
   – вычитание
   * умножение
   / или : деление
   ** или ! возведение в степень

5. Переходы: GOTO номер_метки – безусловный переход к строке с меткой.
   Метка это число от 1 до 999 (кроме 13), либо целочисленная переменная.

   Условный переход:

   IF (арифм_выражение) M_1 , M_2 , M_3

Здесь роль меток переходов такова: M_1 если арифм_выражение < 0
M_2 если арифм_выражение = 0
M_3 если арифм_выражение > 0

6. Операторные скобки:

```
BEGIN
описание_переменных
операторы
END

IF (выражение1) знак >, < или = (выражение2)
…
операторы, и, может быть, ELSE
…
END

WHILE (выр1  знак >, < , = или #  выр2)
…
операторы; они выполняются, пока выполнено условие в скобках
…
END
```

В операторных скобках переменным можно давать старые имена; они не перепутаются с аналогичными переменными в других местах (так ли это? я не проверял толком).

7. Операторы ввода-вывода:

```
READ(K,M)A,B,C,…
WRITE(K,M) A,B,C,…
```

где: K – номер устройства,
M – метка оператора формата; K и M – целочисленные.

Возможные значения K:
0 – дисплей, 1 – печатающее устройство, 2 – магнитофон (НМЛ),
3 – перфоратор, 4 – фотосчитыватель,

Задание формата:

```
Номер_оператора.   Метка        .  FORMAT(… , … , …).
```

Возможные указатели формата (в скобках оператора формата):

/ – перевод строки.
IN – целое число с количеством разрядов N.
FN,K – вещественное число (REAL) с N разрядами до десятичной запятой, K – после запятой.
E – число с указанием его десятичного порядка.
M1 – машинный формат (на НМЛ).
'текст' – сопровождающий текст

Ввод числа (в работающей программе) подтверждается клавишей ПС.

8. Цикл:

```
        DO  M  I=K1, K2, K3
        ^ оператор
        ^ оператор
        ^ ……————
M       ^ CONTINUE
```

где: M – метка оператора CONTINUE,
I – переменная цикла,
K1 – начальное значение переменной цикла (или арифм. выр. для него),
K2 – конечное значение переменной цикла,
K3 – шаг изменения переменной цикла,

9. Обращение к подпрограммам:

```
CALL имя_пп  (A, B, X[…, …], … )
```

В скобках перечисляются формальные параметры. Они отождествляются с переменными в описании подпрограммы в той последовательности, которая указана в скобках при описании подпрограммы, набранном при её вводе. (Если параметров нет, то скобки не надо набирать.)

Для ввода подпрограммы после приглашения системы «SUBROUTINE имя_пп» надо нажать пробел. Появится нумерованная строка, в ней в позиции 3-го останова надо нажать S:

SUBROUTINE имя_пп (C, D, Y#, …)
……
оператор
оператор
……
RETURN
END

10. Ввод программы завершается оператором END. При этом производится автоматический контроль программы и переход в «операционную систему». Если отсутствуют метки, указанные в операторах, то выводятся сообщения о таких метках:

G номер_отсутствующей_метки
G номер_отсутствующей_метки
…….

До ввода оператора END можно прочитать адреса переменных. Для этого надо в крайней левой позиции подать команду I. При этом на дисплей выведется таблица идентификаторов и адресов переменных. Это может потребоваться при вводе операторов с машинными кодами.

S B – команда перехода из «операционной системы» к продолжению ввода (если был END, то его надо удалить в режиме редактирования; а иначе продолжение ввода проигнорируется).

11. Ввод операторов в машинных кодах:

K : … шестнадцатеричные коды команд Д3-28, разделённые пробелами …

Коды вводятся в сокращённой записи (hex-запись):

00 вместо 0000
12 вместо 0102
…….
99 вместо 0909

При этом тетрады, содержащие 10 … 15 обозначаются буквами:

A вместо 10
B вместо 11
C вместо 12
D вместо 13
E вместо 14
F вместо 15

Например, команда 1501 1507 выглядит так: K: F1 F7

Действия в операционной системе:

S Л – листинг набираемой (или прочитанной с МЛ) программы. S АЦПУ Л – на ТПУ.

Редактирование:

S РЕД В KN_1 HN_2 – вставить N_1 операторов, начиная с номера N_2 .

S РЕД У KN_1 HN_2 – удалить N_1 операторов, начиная с номера N_2 .

S Н – нумерация операторов (т. е. строк в программе), необходимая после редактирования. На дисплей выводится количество строк в исходной программе (ИП), без листинга.

Для завершения редактирования необходимо сделать листинг (S Л), т. к. по ходу листинга делается автоматическая проверка соблюдения правил написания программы.

Возможны три режима записи на магнитную ленту (МЛ):
1) запись производится на МЛ последовательно, вне зависимости от чистоты ленты:

S МЛ ЗП Б N 0 – подготовка к записи «библиотеки», начиная с 1-ой программы,
S МЛ ЗП ИП В номер – запись исходной программы, версия с указанным номером, имя программы при этом не задается, т. к. оно уже было задано при наборе программы или при чтении её с ленты.

Для записи на МЛ рабочей программы (т. е. оттранслированной программы) или подпрограммы следует вместо ИП указать РП или ПП, соответственно.

...

К сожалению, на этом конспект обрывается… Однако, как нетрудно заметить, правила работы с FORTRAN-5M почти те же, что уже известные нам правила для Fortran-85 (см. выше). В частности, чтение с МЛ исходной программы с известным именем и номером версии производится командой:

S МЛ ЧТ ИП ИМЯ? имя_программы_4_символа В номер_версии

Список (каталог) фортранных записей, имеющихся на МЛ, можно получить командой:

S МЛ К

При редактировании (и не только) бывает полезным листинг части программы:

S РЕД Л KN1 HN2 – листинг части программы, где N1 – количество операторов (строк), которые надо посмотреть, N2 – номер оператора, с которого следует выводить листинг. При таком листинге не ведётся синтаксический контроль, и поэтому так можно просматривать, в том числе, строки с ошибками – прерывания листинга сообщением ERR здесь не возникнет.

(Эксперименты в имитаторе показывают, что редактирование командой ОП, как в Фортране-85, не работает в Fortran-5M и в фортраноподобных системах ВТ-9Р МХТИ, «ОС ВТ-МХТИ 32к версия ЛФТИ» и, может быть, в других.)

Трансляция исходной программы для получения пригодной к запуску рабочей программы (РП):

S ТП К1 Л? Пробел (а если нажать Л, то трансляция пойдёт с листингом).

Для перехода из ОС транслятора в его начало надо нажать клавишу С (<S> в имитаторе), при этом будет выведено R. Затем нажимаем либо Т, либо (для запуска готовой РП) ещё раз С.

Полезные факты излагаются также в разделах 5 и 13, и, наверное, ещё будут обнаруживаться опытным путём.

12. Приложение 2: команды Выстры

Это не официальный документ, а перепечатка рукописного конспекта некоего документа. Конспект в тетради (нашего коллеги А.В. Березина) был краткий; значит, здесь есть пропуски и могут быть опечатки, неточности.

КРАТКОЕ ОПИСАНИЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ
«ВЫСТРА» ДЛЯ «Д3-28»
(ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА–ТРАНСЛЯТОР)

Для работы с фрязинским дисплеем (15ИЭ-00-013) следует в служебной строке дисплея установить две единицы:

Служебная строка дисплея: 9600 0000 0010 0010 0000

1. *Запуск системы*

Система ВЫСТРА (далее просто «система») загружается в ОЗУ машины Д3-28 с магнитной ленты (МЛ) с начального адреса (номера шага) 00000. Запуск – клавишей «S» на пульте машины. В процессе работы система контролирует свою сохранность, сигнализируя о порче зуммером дисплея. Работу системы можно прервать, нажав на пульте машины поочередно клавиши «ШН» и «С». После этого запуск осуществляется снова клавишей «S».

После запуска системы в левом верхнем углу экрана появляется буква Ф. Это означает, что система ждёт указаний – какую функцию выполнять. Надо выбрать желаемую функцию, например И (см. об этом ниже). Выбор функции, как и любое действие, подтверждается на клавиатуре дисплея клавишей «ПС» (или комбинацией клавиш «СУ»+«J»), а окончание ввода и переход снова в режим Ф подтверждается клавишей «ВК».

2. *Функции системы*

<u>2.1 Формирование (набор и редактирование) исходных текстов программ</u>. Команды:

И – изменение текста. Например, пусть уже имеется текст многострочной программы. Нажимаем «И», «ПС», указываем трехзначный номер строки, и нажимаем «ПС»; система выведет на экран текст, начиная со строки 000 по указанную строку. Нажимаем «ВК»; система переходит в режим выбора Ф. Если теперь исполнить команду, обозначаемую скобкой **(** , то уничтожатся все строки программы с номерами, большими указанного после И. То есть:

(– убрать текст после текущей строки. Исполняется после нажатия ПС.

) – убрать текст перед текущей строкой. Исполняется после нажатия ПС.

9 – перенумеровка строк в порядке возрастания номеров строк. Исполняется после нажатия ПС.

Ж – уничтожение текста полностью (подготовка к вводу совсем новой программы). Исполняется после нажатия ПС.

Набор новой программы также начинается с И (или I), ПС, 000, ПС.

2.2 Ввод / вывод текста на носители. Команды:

П – распечатка программы на ТПУ.

А – запись на МЛ. Нажав «ПС» после «А», вводим имя программы (от 0 до 99 любых символов, лучше не длинное) и также подтверждаем нажатием «ПС».

С – считывание с МЛ. Например, для ввода с МЛ программы с именем нажимаем: «С», «ПС». *имя_программы*, «ПС». Если не указать имя программы, то система будет читать и выводить на экран имена всех программ подряд до конца ленты, не загружая программ. Вместо имени сбойной программы система выводит на экран символ *. При поиске программы с заданным именем всё происходит так же, но найденная программа загружается в память и дальнейшее чтение МЛ прекращается.

Если программа с нужным именем на МЛ не найдена или имя не было указано, то для выхода из режима чтения МЛ можно прервать работу системы, нажав на пульте машины клавишу «ШН» и затем «С» (в имитаторе <C> при **View>Machine**) После этого – запустить систему клавишей «S» для продолжения работы.

Л – вывод на перфоленту.

Ч – ввод с перфоленты.

Я – запись в «ящик» (в ВЗУ – внешнее запоминающее устройство).

Ы – вытащить из ВЗУ.

Примечание: выводится на носители всегда весь текст. При вводе с носителей вводимый текст добавляется в конец к имеющемуся в ОЗУ тексту; и тогда, если эта добавка должна стать продолжением имевшейся исходной программы, надо сделать перенумеровку командой «9».

2.3 Трансляция и выполнение программы.

К – компиляция (трансляция),

Э – экономичная трансляция,

Е – счет по программе (выполняется только после К или Э).

Г – уничтожение системы. При этом освобождается память, которую занимала система; транслированная программа может использовать всю память. Запускается транслированная программа клавишей «S» на пульте машины.

После записи транслированной программы на МЛ с ней можно работать как с обычной программой в машинных кодах, т. е. без загрузки системы «ВЫСТРА»: транслированная программа считывается с МЛ по команде «СЛ» на пульте машины, загрузка в ОЗУ производится на начальный адрес 0; запуск – по команде «S» на пульте машины.

Запись транслированной программы на МЛ после операций Ж и Г производится как обычно – по команде «ЗЛ» после «С» на пульте Д3-28. Просмотр контрольной суммы – «КП».

Пояснения к написанию текста программы

Строки программы нумеруются трёхзначными числами: 000, 001, 002, … . Допускаются пустые строки (содержащие только номер строки) – это будут резервные строки на случай редактирования текста программы. Операторы программы в каждой строке отделены от номера строки пробелом; он вставляется системой автоматически.

Текст программы необходимо заканчивать символом #. Этот символ, является признаком конца программы, он обязательно должен быть в последней строке программы. Завершать ввод строки с символом # надо клавишей ВК.

В тексте программы используются ключевые (кодовые) слова, специальные символы, идентификаторы переменных, стандартных функций, а также меток. В строке может быть несколько операторов, они разделяются точкой с запятой « **;** »

Оператором перехода к метке служит просто сам идентификатор этой метки, например, буква «М». В том месте программы, куда должен произойти переход, надо написать такой же идентификатор с двоеточием (в данном примере «М**:**»).

Идентификаторы начинаются с латинской или русской буквы, затем могут идти цифры или буквы, всего до 255 символов.

Ключевые (кодовые) слова пишутся в кавычках **'**…**'** :

'DC' … объявления переменных ; операторы … ; 'EN';
'DO' … 'TO' … 'OD';
'IF' … … 'FI';
'PR' имя процедуры ; переменные ; операторы … ; 'EP';
'EX' имя вызываемой процедуры ;
'KO' коды машинных команд через запятую ;
'IO' команды для ввода-вывода ;
'ST' … ;

Определяющими являются первые две буквы, за ними можно писать любые символы.

Например, можно писать 'STOP' вместо 'ST', или 'КОД' вместо 'KO', и так далее.

Подобным же образом, если в программе имеются несколько участков с определениями переменных, то их можно помещать для удобства в нумерованные кодовые «скобки»:
'DC1' … ; 'EN1'; затем (или внутри) 'DC2' … ; 'EN2'; и т. д.

Специальные символы:

¤ – символ, означающий начало комментария, например:

```
000 ¤ TEST   100 СТРОК;
. . .      . . .
099 #
```

[ ] , ( ) – скобки.

+ – * / – знаки операций (сложить, вычесть, умножить, разделить),
? возведение в степень,
= равно,
не равно,
> , < больше, меньше. Знаки ≥ и ≤ кодируются как «СУ»+«О» и «СУ»+«Е», но на экране не отображаются.
@ или,
& и,
¬ не,
; , " разделители.

Переменные, нумерующие элементы массива, перечисляются в квадратных скобках, через запятую. Например: `A[I,J]`.

Операции типа B=B+C допускается записывать в виде: B+=C ; аналогичным образом допустимы записи типа B–=C; B*=C; B/=C, где C может быть не только переменной, а и математическим выражением.

Количество процедур, меток, и т. п. в программе ограничено:

количество процедур …….....	не более 63
меток …….....	не более 192
индексов в массиве …….....	не более 10
ячеек прямой адресации …..	не более 159
символов в операторе ….....	не более 400
символов в идентификаторе..	не более 255

Стандартные функции:

EXP, LN, LG, ABS, SIN, COS, TG, SH, CH, TH, ASIN, ACOS, ATG, ATH, ACH,

INT – взятие целой части,
SQRT – корень квадратный,
SQRE – квадрат (работает быстрее, чем возведение в степень два: ?2),
NOF – возвращает свой аргумент без изменения,

INPUT – ввод числа с пульта Д3-28 (аргумент отображается на индикаторе регистра X, затем надо набрать вводимое число и нажать «S»).

Пример ввода массива посредством INPUT в двойном цикле:

```
'DC' A[5,7];
'DO' I 'TO' 5;
'DO' J 'TO' 7;
A[I,J] = INPUT(I+100*J);
'OD'; 'OD';
…… 'EN';
```

Заодно стоит отметить, что переменные циклов не надо объявлять в 'DC'… . При указанной здесь краткой форме цикла переменные цикла I и J пробегают значения от 1 до заданных значений с шагом 1.

Более удобен ввод чисел с клавиатуры дисплея, но для этого в программе должны быть строки со специальной процедурой INP(X) (как в программе FRMJ, см. раздел 5).

<u>Примеры команд вывода на экран дисплея</u>:

'IO' KC; – этот оператор переводит курсор в начало новой строки.

'IO' KT, ТЕКСТ; – выводится слово ТЕКСТ с текущей позиции курсора.

'IO' KT, "Т"Е"К"С"Т; – выводится слово Т Е К С Т с начальным пробелом в текущей позиции курсора. Пробелы выводятся вместо кавычек "

Пример вывода значения переменной, например, P:

'IO' KC, KT, … поясняющий текст … , КЧ, P, 3.3;

этот оператор переводит курсор на новую строку, выводит поясняющий текст, и выводит значение переменной P с тремя знаками до десятичной точки и с тремя знаками после. Буквы КЧ здесь могут быть русскими. Во всех примерах подразумевается, что операторы должны быть в пронумерованных строках.

Пример запуска простейшей программки:

1) Набираем простейшую программку:

```
000 'IO' KT, TEXT, KC;
001
002 #
003
```

Все буквы здесь набираются как латинские, в ЛАТ-режиме дисплея. KC в конце оператора вывода – для перевода курсора в новую строку после вывода слова TEXT.

Ввод программы завершаем клавишей «ВК».

2) Затем выполняем функцию К, т. е. компилируем. На экран выведется информация:

```
000  00512
001
002
   10096  00527  00002Ф
```

3) Выполняем функцию E. На экран будет выведен результат работы данной программы:

TEXT
Ф

Буква Ф является признаком того, что система ожидает дальнейших указаний.

К сожалению, в кратком конспекте было мало пояснений к применению разных операторов, условных переходов, вызова процедур и прочих деталей. Однако, «язык ВЫСТРА» лаконичен, читается легко, так что подробности легко увидеть в конкретных примерах. В качестве одного из простых примеров рассмотрим следующую небольшую программку:

```
000 'PR'INK(K);'KO'1304,715,1305,1515,1302,1507,1303,1515,412,1407,1402,3,
001 1302,1501,412,1406,1402,3,1302,0,413,409,404,K;'EP';
002 'DC'K;'IO'KC;M:;'EX'INK(K);'IO'KT,",КЧ,K,3.0;M;'EN';
003 #
```

Здесь в первых двух строках между словами `'PR'` и `'EP'` даётся определение нашей пользовательской процедуры `INK(K)`, которая описывается машинными кодами после слова `'KO'`:

1304 0715 – в регистр S_4 засылается число 0715,
1305 1515 – в регистр S_5 засылается число 1515; эти два однобайтовых регистра составляют регистр R_{10} .
1302 1507 – в регистр S_2 засылается число 1507, это будет код на шине УПР при активации дисплея.
1303 1515 – в регистр S_3 засылается число 1515, это «вектор» (номер) контроллера дисплея при его активации.
0412 1407 – команда OUTOWS: передача на шину ВЫВ одного байта из S_3 при состоянии шины УПР = S_2 .
Время ожидания ответа (СИП) ограничено, оно определяется числом в R_{10} . Если СИП нет, то:
1402 0003 – переход на 3+1=4 шага назад, т.е. опять к выполнению OUTOWS. Пропускается, если СИП поступил.
1302 1501 – в регистр S_2 засылается байт 1501, это будет код на шине УПР при обмене данными с дисплеем.
0412 1406 – команда INPOWS: приём с шины ВВОД одного байта в S_3 при УПР = S_2.
Время ожидания СИП ограничено, оно определяется числом в R_{10} . Если СИП не поступил, то:
1402 0003 – переход на 4 шага назад, т.е. опять к выполнению INPOWS. Пропускается, если СИП поступил.
1302 0000 – в регистр S_2 засылается байт 0000. Регистры S_2 и S_3 составляют регистр R_9 . Таким образом, в регистре R_9 хранится 16-ричное число, равное коду, принятому из дисплея командой INPOWS.
0413 0409 – число из R_9 переводится в десятичную систему счисления и засылается в регистр X.
0404 xxxx – Число из регистра X засылается в десятичную ячейку памяти с номером xxxx. В данной программе в роли номера xxxx выступает формальный параметр K, объявленный в заголовке процедуры.

В строке 002 между словами `'DC'` и `'EN'` задаётся собственно та последовательность действий, которую должна выполнять программка. Команда K предписывает компилятору назначить конкретный адрес десятичной ячейки памяти для переменной K. Команда `'IO'KC` означает вывод в дисплей кода ПС (с учётом того, что в служебной строке дисплея установлен режим «авто ПС с ВК», получится перевод курсора в начало следующей строки). Буква с двоеточием M**:** означает просто метку для следующих за ней команд. Следующими командами являются: `'EX'INK(K)` – исполнение процедуры ввода байта из дисплея с засылкой его десятичного значения в переменную с именем K; затем `'IO'KT … ;` – на экран выводится пробел и десятичное число K с тремя знаками до запятой и с нулём знаков после запятой; и затем – переход к метке M, т.е. опять к ожиданию ввода байта с дисплея, и т.д. Таким образом, эта программка выводит на экран коды нажатых клавиш в десятичной форме.

Набирать программку следует в режиме ЛАТ; лишь КЧ (или только Ч) набирается РУС-буквами. Результат компиляции и затем исполнения программки с нажатиями клавиш QWERTYUIOP[] выглядит так:

```
↑ K

000   00512
001   00516  00540
002   00543  00547  00551  00553  00562  00577  00579
003
  10256  00585  00006↑ E

   81   87   69   82   84   89   85   73   79   80   91   93_
```

Для продолжения учёбы можно перейти к **View > Machine**, остановить машину и снова войти в Выстру: <Enter>, <C>, <S>, **View > Display**. Командой Ф А ПС *имя_файла* ПС желательно записать выстровский текст набранной программки в новый txt-файл, например, с именем KLAV-DEC-COD (тогда считывать запись надо будет тоже с этим именем – командой Ф С ПС KLAV-DEC-COD ПС; букву Ф система выводит сама, это приглашение).

Затем, вызвав, командой Ф I ПС 002 ПС листинг с текущей строкой 002, можно двигать курсор вправо командой АР2 С, удалять клавишей ЗБ символ переменной К и печатать вместо него L, – так, чтобы отредактированная строка приняла следующий вид: **`002 'DC'L;'IO'KC;M:;'EX'INK(L);'IO'KT,",KЧ,L,3.0;M;'EN';`**
При необходимости подвинуть курсор влево пользуемся командой АР2 D. Закончив, нажимаем ВК. Компиляция и запуск изменённой так программки показывает, что она осталась работоспособной. Этот опыт говорит нам о том, что при вызове процедуры на исполнение можно в её аргументах указывать новое имя переменной.

Ещё пример редактирования: снова вызвав листинг с текущей строкой 002, изменим в ней некоторые символы так, что строка примет вид: **`002 'DC'L;M:;'EX'INK(L);'IO'KC,KЧ,L,3.0;M;'EN';`**
В таком варианте программка будет выводить десятичные коды клавиш не строчкой, а столбиком.

Пример сравнительно большой программы, проверенной в реале, – FRMJ (от слов Fractal, Mandelbrot и Julia). В частности, в ней есть процедура INP(X), позволяющая вводить числа с клавиатуры дисплея, а не с пульта Д3-28, как с INPUT. Работа FRMJ описана в frmj__7-04-2017.pdf: https://yadi.sk/d/BFXSop4Z3GmJRN , сама эта программа имеется у нас в папке [txt] в файле-библиотеке frmj_inpx_inpu_outx_test__vystra-text.txt.

Текст процедуры INP(X), позволяющей в любой выстра-программе вводить числа с клавиатуры терминала, целесообразно хранить на МЛ. При наборе новых программ можно пользоваться этим сохранённым текстом без каких-либо изменений в нём. О получении листинга и о распечатывании выстровских листингов на ТПУ рассказано в разделе 5, здесь на этом не останавливаюсь.

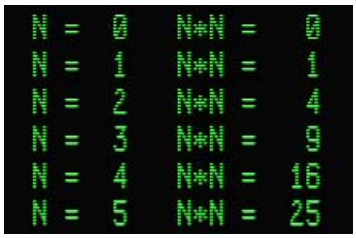
В txt-файле frmj_inpx_inpu_outx_test__vystra-text.txt есть и более простые примеры; о них тоже говорилось в разделе 5. В конце txt-файла приведены пояснения и листинги. Программка с именем INPU иллюстрирует применение встроенного в Выстру оператора INPUT(X) для ввода чисел с пульта машины через регистр X – без клавиатуры терминала. Процедура OUT(X) выводит аргумент X на экран в формате "мантисса и порядок числа", т. е. она позволяет выводить б*о*льшую абсолютную величину порядка, чем через опции оператора `'IO'KЧ, …` .

Детали языка «Выстра» желательно выяснять экспериментально. Например, опыт показывает, что для цикла допустима не только краткая форма записи, но и форма с указанием начального и конечного значений переменной цикла в отдельных переменных, в одной или в двух: «Длинный» оператор можно «размазать» по двум (а то и более) строкам – для удобства визуального восприятия и редактирования. Вот работоспособный пример:

Код исходной программки:

```
000 'DC'N1,N2,A[5];
001 N1=0;N2=5;'DO'LOOP=N1'TO'N2;
002 A[LOOP]=LOOP*LOOP;
003 'IO'KC,KT,N"=",KЧ,LOOP,1.0,
004 KT,"""N*N"=",KЧ,A[LOOP],2.0;
005 'OD';'IO'KC;'EN';
006 #
```

Результат:



В системе «Выстра» на комплексе «Д3-28 + дисплейный терминал» можно создавать и отлаживать программки для «голой» Д3-28, т. е. с вводом-выводом только через пульт и табло Д3-28. Ввод обеспечивается оператором INPUT(A). Вывод – оператором 'STOP'B,C; он переводит машину в состояние «останов с индикацией», и при этом значения переменных B и C отображаются соответственно на табло X и Y (имена переменных, конечно, могут быть другими). Простой пример есть в q-eq_ris-9_KP-543_KP-7765_KP-34321.txt; необходимые пояснения, с листингом, приведены в конце этого txt-файла.

Более серьёзные примеры выстровских программок – для Д3-28 без дисплейного терминала, но, может быть, с какими-то другими периферийными устройствами, в том числе с УСО АЦСКС-1024-001, – подробно разбираются в файле sip_sip-wyw-auto_sip-upr-auto__vystra-texts.txt. Там же демонстрируется удобный метод построения программы в виде последовательности вызовов заранее определённых процедур, с передаваемыми и возвращаемыми параметрами; причём, параметрами могут быть не только простые переменные, но и массивы.

13. Приложение 3: дополнительные пояснения

Вывод символов на дисплей в Бейсике с помощью машинных кодов

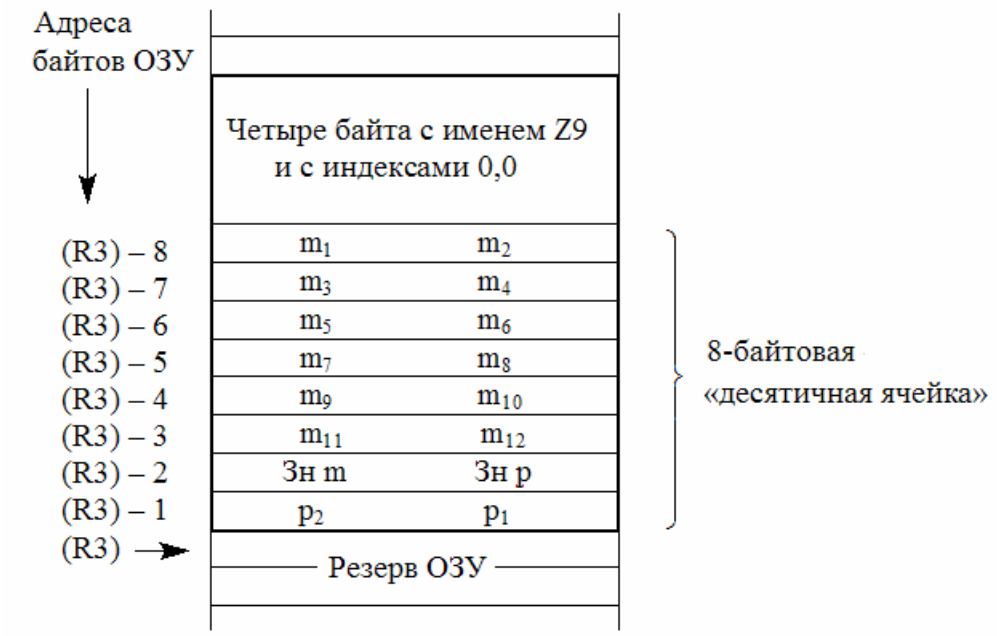
Рассмотрим вкратце работу программы DISP-1; эта программа задумана как тест для разработки и проверки способов вывода символов на экран дисплея подачей в дисплей машинных кодов с помощью команды Бейсика CMD. Сначала программа очищает экран и рисует символами * прямую линию, а затем запрашивает координаты X, Y и код для следующего символа. Координата X это номер позиции в строке, допустимые значения: 0 … 79. Координата Y = номер строки: 0 … 23. Байт-код символа следует вводить как десятичное число 16·B+A, где B – старшая тетрада, A – младшая тетрада байта. Байт-коды символов можно узнать в документации к дисплею, или из программы display_codes_KP-1689_N-106.txt. Например, десятичное число 32 соответствует пробелу, ч*и*сла от 33 до 47 соответствуют символам ! “ # и т. д. до / , числа от 48 до 57 соответствуют цифрам от 0 до 9, числа от 58 до 64 соответствуют символам **: ;** < = > ? @ , затем идут коды букв, квадратных скобок, и т. п. Наибольшее допустимое число равно 127, оно соответствует символу «забой».

Листинг программы «DISP-1»:

```
1 REM ' DISP - ТЕСТ-ПРОГРАММА ВЫВОДА БАЙТ-КОДОВ В ДИСПЛЕЙ '
2LETX=0:LETY=0:LETZ7=0
3LETI=0:LETJ=0:LETK=0:LETL=0
4 LETX1=0:LETY1=0
8 LETZ8=0:LETZ9=0:REM ' ЭТО БУФЕРЫ '
9 GOSUB200:GOSUB300:GOTO90
10LETZ9=32+Y:LETZ7=INT(Z9/16)
12LETZ9=4096*Z7+256*(Z9-Z7*16)+32+X
14CMD11040310,10010810,04120310,04131209,10000210,09040910
16CMD11040310,10010810,13020111,13030509,09040910
17CMD11040310,10011510,10010510,04120310,04131209,10001510,10000110,09120310
18CMD11040310,10010810,13090005,15011501
20RETURN
91LETK=10
92FORI=0TOK
94LETX=2*I
95LETY=I
96LETZ8=42:GOSUB10
98NEXTI
99 GOTO 122
100INPUT'X,Y='X,Y
102IFX<0GOTO7999
110 INPUT' БАЙТ = 'Z8
120GOSUB10
122LETX=0:LETY=20:LETZ8=32
123GOSUB10
130GOTO100
200 LETZ9=17
205 CMD11040310,10010810,13090001,15011501
210 RETURN
300CMD11040310,10010610,13020111,13030410,09040910,13030408,10010210,09040910,13090004,
15011501
305 RETURN
7999END
```

Вся идея основана на том, что переменные в Бейсике заносятся в память в порядке их определения в программе, а регистр R_3 хранит номер первого свободного байта в конце области памяти, занятой переменными. Поэтому мы можем заносить в ячейки переменных желаемые

коды команд для дисплея, и узнавать их адреса (номера) через регистр R_3. Например, пусть последняя определённая переменная есть Z9. Её имя занимает 4 байта, а затем идут 8 байт, хранящие в своих тетрадах 12 цифр мантиссы $m_1 \ldots m_{12}$, знак мантиссы, знак порядка, и две цифры порядка p_2 , p_1 :



Допустим, мы хотим, чтобы программа перевела дисплей в систему команд №2. В документации к дисплею сказано, что для этого надо передать в дисплей байт-код с тетрадами 01 07. С этой целью мы пишем в бейсик-программе: Z9 = 17. Тем самым задаются желаемые значения первых двух тетрад мантиссы: $m_1 = 1$, $m_2 = 7$. Затем оператором CMD исполняется последовательность машинных команд (составленная по информации из «*Справочника программиста*»):

1104 0310 – пересылка содержимого регистра R3 в регистр R10: R10 ← (R3),
1001 0810 – вычитание из (R10) числа 8, т. е.: R10 ← (R10) – 8,
1309 0001 – вводим в 1-байтовый регистр S9 число 1,
1501 1501 – команда OUTS: передача из машины в шину «Вывод» (S9) байт с начального адреса из регистра R10 при состоянии шины УПР = 1501 (такое УПР нужно при передаче в контроллер дисплея из машины кода символа или команды.

В программе все эти действия выполняет подпрограмма, расположенная в строках 200–210. Аналогично устроена и подпрограмма быстрой очистки экрана (см. строки 300–305). Для очистки экрана дисплея, работающего в системе команд №2, надо передать ему 4 байта: два байта 0111 0408 образуют команду AP2 H – перемещение маркера в начало страницы, два байта 0111 0410 составляют команду AP2 J – стирание текста от позиции маркера до конца страницы на экране. Присутствующие здесь тетрады 11 и 10 не могут быть цифрами мантиссы десятичного числа Z9, так как для разрядов мантиссы любого десятичного числа допустимы лишь цифры от 0 до 9. Поэтому Z9 здесь не задаётся как бейсиковская десятичная переменная, а применяется прямая пересылка желаемых байт-кодов в первые четыре байта 8-байтовой ячейки Z9 из регистра R9 с помощью машинных команд в операторе CMD. Подобным же образом подпрограмма в строках 10–20 формирует в байтовых ячейках ОЗУ, относящихся к переменным Z8 и Z9, код выводимого на экран символа и коды дисплейной команды AP2 Y – это «прямая адресация маркера» в позицию, которая задаётся следующими двумя байтами

(с десятичными значениями 32+Y, 32+X, где Y и X – номер строки и номер позиции в строке на экране алфавитно-цифрового дисплея).

Вывод на дисплей в Фортране с помощью машинных кодов

Игровая тест-программка 1WAR иллюстрирует способ вывода символов на дисплей 15 ИЭ-00-013 путём расчёта кода символов в «буферных» переменных Фортрана и прямой адресации курсора в желаемую позицию на экране дисплея.

Листинг программы 1WAR в Фортране-5М:

```
    .     .   MAIN PROGRAM 1WAR
   1.     .   INTEGER A,B,C,O,T,X,Y,M,Z9
   2.     .   REAL V,D,G,R
   3.     .   INTEGER P
   4.     .    A=5888
   5.     .   K:AC AD AC CD D4 00 D5 50 D9 01 F1 F1 AF CD AF AD
   6.     .    P=0
   7.     .    A=3867
   8.     .    B=18459
   9.     .    C=18944
  10.     .    M=2
  11.    1.   K:AC AD AC CD D4 00 D5 50 D9 05 F1 F1 AF CD AF AD
  12.     .   GOTO M
  13.    2.    A=7001
  14.     .    C=32512
  15.     .   DO 3 I=1,4,1
  16.     .  ^  X=36+I
  17.     .  ^ DO 4 J=1,4,1
  18.     .  ^^  Y=10+J
  19.     .  ^^  B=256*(32+Y)+32+X
  20.     .  ^^ K:AC AD AC CD D4 00 D5 50 D9 05 F1 F1 AF CD AF AD
  21.    4.  ^^ CONTINUE
  22.    3.  ^ CONTINUE
  23.     .    Y=0
  24.     .    X=0
  25.     .    C=11776
  26.     .    V=1
  27.     .    R=1
  28.     .    G=1
  29.     .    T=0
  30.   90.    B=X-76
  31.     .    Z9=B
  32.     .   IF(Z9)100,110,110
  33.  100.    Z9=X-3
  34.     .   IF(Z9)130,120,120
  35.  120.    Z9=Y-22
  36.     .   IF(Z9)140,150,150
  37.  140.    Z9=Y-1
  38.     .   IF(Z9)170,160,160
  39.  110.    R=-1
  40.     .    T=T+1
  41.     .   GOTO 120
  42.  130.    R=1
  43.     .    T=T+1
  44.     .   GOTO 120
  45.  150.    G=-1
```

```
46.     .    T=T+1
47.     .   GOTO 160
48. 170.     G=1
49.     .    T=T+1
50. 160.     X=X+V*R
51.     .    Y=Y+G
52.     .    B=256*(32+Y)+32+X
53.     .   K:AC AD AC CD D4 00 D5 50 D9 05 F1 F1 AF CD AF AD
54.     .   K:AC 9D D2 F1 D3 00 4C E6 5E 5E 4D 49 C9 44 21 AF 9D
55.     .    P=P+1
56.     .    D=-D-0.28976E5
57.     .   IF(D-0)180,190,191
58. 180.   GOTO 90
59. 191.   IF(D-3)200,200,210
60. 200.    V=D
61.     .   GOTO 90
62. 210.   IF(D-9)220,230,230
63. 220.   IF(D-7)250,260,270
64. 250.    G=1
65.     .   GOTO 90
66. 260.    R=-1
67.     .   GOTO 90
68. 270.    R=1
69.     .   GOTO 90
70. 230.    G=-1
71.     .   GOTO 90
72. 190.    T=T-4
73.    5.   FORMAT(/,'СУММА ШТРАФНЫХ ОЧКОВ = ',I4)
74.    6.   FORMAT(/,'ВРЕМЯ БОЯ = ',I6)
75.     .   WRITE(0,6)P
76.     .   WRITE(0,5)T
77.     .   END
```

Пояснения:

В начальных строках программы объявляются имена и типы переменных. Для дальнейшего важно, что целочисленные (двухбайтовые) переменные *A*, *B*, *C* объявлены в указанном порядке. Согласно описанию «Фортрана для Д3-28» их адреса в ОЗУ машины нам будут известны, и они смогут играть роль буферных ячеек памяти, в которых мы будем формировать байты кодов управления для вывода их в дисплей машинными командами по шинам «Ввод/Вывод» и «УПР».

Первым примером управления дисплеем служат строки 4, 5. Десятичное значение переменной *A* сформировано по следующему принципу (и аналогичный принцип будет применяться для переменных *B* и *C*):

$$A = 4096 \cdot b2 + 256 \cdot a2 + 16 \cdot b1 + 1 \cdot a1 ,$$

где: b2 и a2 – десятичные значения тетрад байта с адресом (BD)+80_{10},
b1 и a1 – десятичные значения тетрад следующего за ним байта.

За этими байтами в памяти машины аналогичным образом располагаются два байта переменной *B*, а за ними – два байта переменной *C*.

Десятичное значение *A* = 5888 согласно указанной формуле представляется в виде

$$5888_{10} = 4096 \cdot 01 + 256 \cdot 07 + 16 \cdot 00 + 1 \cdot 00 ,$$

то есть в шестнадцатеричной системе с десятичной записью тетрад это число представляет собой пару байтов вида 0107 и 0000. Здесь первый байт – код перехода к «системе команд №2» дисплея 15 ИЭ-00-013, а второй байт имеет нулевое значение и не будет играть роли при выводе в дисплей только первого байта.

Строка 5 программы содержит оператор K с машинными командами в hex-записи, предназначенными для вывода в дисплей первого байта из «буфера A». При десятичной записи эти команды имеют следующий вид (заодно поясняю их смысл):

1012 1013 – запись в стек содержимого регистра R_{10} (чтобы позже его восстановить),
1012 1213 – запись в стек содержимого регистра R_{12} ,
1304 0000 – занесли в регистр S_4 байт с нулями 0000,
1305 0500 – занесли в регистр S_5 байт 0500 = $(16\cdot05+1\cdot00)_{10} = 80_{10}$.
Пара байтовых регистров S_4 и S_5 образует двухбайтовый регистр R_{10}. Таким образом, в R_{10} мы занесли число 80, т. е. смещение относительно (BD) первого байта первой переменной типа INTEGER (у нас это переменная *A*).
1309 0001 – занесли в регистр S_9 байт 0001, т. е. число 1.
1501 1501 – команда вывода (OUTS) при состоянии шины управления УПР=1501. Этот вывод предназначен дисплею. Командой OUTS в шину «Выв» выводятся подряд S_9 байтов из памяти машины с начального адреса (R_{10})+ (BD). То есть в нашем случае будет выведен 1 байт – первый байт переменной *A*; он содержит код 0107 – код перевода дисплея «в систему команд №2» (см. описание дисплея).
1015 1213 – восстановили из стека содержимое регистра R_{12} ,
1015 1013 – восстановили из стека содержимое регистра R_{10} , так как эти и другие регистры в служебной зоне ОЗУ Д3-28 могут быть нужны для работы Фортрана.

В строке 11 программы аналогичным образом в дисплей выводятся 5 байтов, предварительно сформированных в трёх последовательно расположенных в памяти машины двухбайтовых переменных: *A* = 3867, *B* = 18459, *C* = 18944. В шестнадцатеричной системе эти значения переменных представляются байтами так:

Адрес = (BD)+80_{10}	Переменная	*A* =	0015 0111	– это код клавиши ЛАТ – код АР2
		B =	0408 0111	– код H – код АР2
		C =	0410 0000	– код J – этот байт не используется.

Т. е. сначала выводится байт 0015 – код перевода дисплея в «латинский регистр». Затем выводится пара байтов 0111 0408, представляющая собой код команды АР2 H – это дисплейная команда перемещения курсора в начало страницы. И затем выводится пара байтов 0111 0410 – код дисплейной команды АР2 J, которой стирается текст экрана от курсора и до конца страницы. Вся эта комбинация значений *A*, *B*, *C* вместе с машинными командами вывода 5 байтов составляет «команду очистки экрана» (она пригодна для применения также в других программах или в разных участках одной программы).

Строки 13–22 содержат двойной цикл; он строит на экране дисплея «эскадрилью» из 4 x 4 символов «забоя». В игре они выполняют роль вражеских самолётов, которые нужно как можно быстрее сбить). Значения переменных *A*, *B*, *C* здесь выбраны по следующим соображениям:

A = 7001_{10} = $(0111\ \ 0509)_{16}$ – это два первых байта четырёхбайтовой командной последовательности для дисплея АР2 Y … , она устанавливает курсор в заданную позицию на экране; следующие два байта в этой последовательности должны задавать номер строки Y и номер позиции X в строке на экране дисплея. Значения байтов, которыми задаётся Y, должны лежать в диапазоне от $0200_{16} = 32_{10}$ (для верхней строки) до $0307_{16} = 56_{10}$ (нижняя строка), а строк на экране может быть не больше 24. Следовательно, при естественной нумерации строк числами Y = 0 … 23_{10} байт информации о номере строки Y должен формироваться как число Y+32_{10} .

Значения байтов, которыми задаётся X, должны лежать в диапазоне от $0200_{16} = 32_{10}$ (крайнее левое положение курсора) до $0615_{16} = 111_{10}$ (крайнее правое положение курсора), а количество знакомест в строке равно 80. Значит, при нумерации знакомест в строке числами X = 0 … 79 байт информации о номере X должен формироваться как число X+32_{10} .Этими соображениями и определяется формула для двухбайтовой переменной *B*:

$B = 256\cdot(Y + 32) + X + 32$.

C = 32512_{10} = $(0715\ \ 0000)_{16}$ – здесь байт 0715 представляет код символа «забой».

Кодовая вставка в строке 20 (так же как и в строке 11) обеспечивает вывод 5 байтов из буферных переменных *A*, *B*, *C* на дисплей.

Основные «игровые» действия осуществляются в строках от 30-й (с меткой 90) и далее. Из стартовой позиции с X = 0 и Y = 0 вылетает «трассирующая» точка – символическое изображение истребителя, который должен таранить вражеские «самолёты», оставляя на их месте символ точки. Первоначальное направление движения точки определяется приращением X и Y с шагом 1. Вывод кода точки в текущую позицию X и Y организован в строках после метки 160 уже знакомым нам способом: кодовая вставка в строке 53 обеспечивает вывод 5 байтов из буферных переменных *A*, *B*, *C* в дисплей. Переменная *A* = 7001 содержит два байта кодов AP2 Y, переменная *B* содержат два байта информации о позиции курсора, переменная *C* содержит байт кода точки 0214: $C = 11776_{10} = (0214\ \ 0000)_{16}$

При достижении трассирующей точкой границ основной части экрана, она зеркально отражается от границы. Однако подобное движение «истребителя» не позволяет ему протаранить все 16 целей. Игрок должен управлять направлением полёта истребителя, добиваясь скорейшего уничтожения всех целей при минимальном количестве отражений от границ поля боя. Для этого в строке 54 имеются команды приёма байта с клавиатуры дисплейного терминала; эти команды имеют следующий вид (и смысл):

1012 0913 – запись в стек содержимого регистра R_9,
1302 1501 – занесли в регистр S_2 байт 1501, это будет значение УПР при приёме из дисплея,
1303 0000 – занесли в регистр S_3 байт 0000,
0412 1406 – команда INPOWS: попытка приёма одного байта в регистр S_3 из шины «Ввод» при состоянии шины УПР = (S_2) с ограниченным временем ожидания Синхро-Импульса Периферии (СИП). Если СИП не поступил (т.е. приём байта не состоялся), то машина выполняет две следующие команды, а если СИП поступил (т.е. клавиша на дисплейной клавиатуре была нажата и её код поступил в Д3-28), то машина переходит к выполнению команд через две.
0514 0514 – это две «пустые» команды GO, так что вне зависимости от того, была или не была нажата клавиша на клавиатуре дисплея, машина выполняет дальнейшие команды:
0413 0409 – в десятичный регистр X заносится содержимое регистра $R_9 = S_2\ S_3$. Это перевод в десятичную систему числа $(1501\ \ 0000)_{16}$, если код клавиш в S_3 не поступил, либо числа $(1501\ \ b1a1)_{16}$, если клавиша была нажата и её байт-код b1a1 поступил в S_3.
1209 – нормализация десятичного числа в регистре X, т.е. такой сдвиг мантиссы, чтобы она не начиналась с нулей; соответственно изменяется и порядок числа в регистре X.
0404 0201 – засылка числа из регистра X в десятичную ячейку в памяти машины с адресом относительно (BD), равным десятичному номеру $10 \cdot 02 + 01 = 21$. Согласно описанию распределения памяти в Фортране для Д3-28, номер первой переменной в списке REAL есть 20, так что номер 21 принадлежит второй переменной в перечне REAL, т.е. в нашей программе это переменная *D*.
1015 0913 – восстановили из стека содержимое регистра R_9.

Таким образом, если клавиатуру не нажимали, то *D* будет равно десятичному значению числа

$(1501\ \ 0000)_{16} = (11110001\ \ 00000000)_2$

Поскольку старший бит здесь равен 1, а не 0, то это число интерпретируется машиной как отрицательное число:

$-(01110001\ \ 00000000)_2 = -(0701\ \ 0000)_{16} = -28928_{10}$.

Если же, например, была нажата клавиша «0» (её код есть 0300_{16}) то *D* будет равно десятичному значению другого отрицательного числа:

$(1501\ \ 0300)_{16}$, т.е.: $-(0701\ \ 0300)_{16} = -28976_{10}$.

В строках программы 57, 58 реализовано сравнение числа $-D - 28976$ с нулём. Если это нуль, то, значит, нажималась клавиша «0», и тогда программа переходит к завершению – к строкам 72 … 77, где на экран выводится время боя и сумма штрафных очков. (Клавишу «0» следует нажимать сразу же, как только будет уничтожен последний значок забоя, а иначе продолжится напрасное увеличение «времени боя»).

Если число – D – 28976 меньше нуля, то, значит, никакая клавиша не нажималась, и тогда программа передаст управление на метку 90, т. е. полёты истребителя продолжатся в прежнем режиме.

Если число – D – 28976 больше нуля, то оно равно номеру, указанному на цифровой клавише; при этом работа программы продолжится в строках 59 … 71, где анализируется номер клавиши и в зависимости от него изменяется режим полёта точки-истребителя

Составлению подобных программ, в которых требуется знать адреса переменных (в нашем примере речь идёт о «буферных» переменных A, B, C), способствует имеющаяся в Фортране и в фортраноподобных системах программирования возможность вывода таблицы адресов на экран дисплейного терминала. Таблица адресов выводится на экран командой I в режиме ввода программы, и даёт информацию обо всех переменных, объявленных в программе к моменту подачи команды.

В программе 1WAR все переменные объявляются в первых трёх строках, поэтому для просмотра их адресов можно запросто повторить ввод таких же трёх строк в отдельной программке, чтобы получить доступ к команде I. Вот как это выглядит в «Фортране-5М» и в «ОС МХТИ 32к, версия ЛФТИ»:

```
D3-28-imitator 1.1   [C:\d3-28-imitator\txt\Fortran-5m_KP-138177_N-10414.txt]
File  View
                                9 6 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  00 03 30

RT
FORTRAN-5M
     . MAIN PROGRAM IDENTIFIERS
    1.       INTEGER A,B,C,O,T,X,Y,M,Z9
    2.       REAL V,D,G,R
    3.       INTEGER P
I        id tip Od   U
         A  IS  80.    .
         B  IS  82.    .
         C  IS  84.    .
         O  IS  86.    .
         T  IS  88.    .
         X  IS  90.    .
         Y  IS  94.    .
         M  IS  96.    .
         Z9 IS  98.    .
         V  RS  20.    .
         D  RS  21.    .
         G  RS  22.    .
         R  RS  23.    .
         P  IS 100.    .
=
```

```
D3-28-imitator 1.1   [C:\d3-28-imitator\txt\s4_side2_os-mhti_32k.txt]
File  View
                                9 6 0 0  0 0 0 0  0 0 0 0
M:Л T:Д
M:B:П
        MAIN PROGRAM IDENTIFIERS
      1     INTEGER A,B,C,O,T,X,Y,M,Z9
      2     REAL V,D,G,R
:     3     INTEGER P
I
A  ЦП  81
B  ЦП  83
C  ЦП  85
O  ЦП  87
T  ЦП  89
X  ЦП  91
Y  ЦП  93
M  ЦП  95
Z9 ЦП  97
V  ВП  20
D  ВП  21
G  ВП  22
R  ВП  23
P  ЦП  99
```

Видно, что в Фортране-5М адреса целочисленных (т. е. двухбайтовых) переменных начинаются со смещения 80_{10} (относительно BD) и идут с шагом 2; при этом смещение $92_{10} = 0512_{16}$ пропущено, т. к. оно совпадает с кодом команды END. В ОС МХТИ 32к сделано хитрее – адреса целочисленных переменных начинаются с нечётного смещения 81_{10}, поэтому пропуска нет.

Кстати, из этой картины становится понятно, что для запуска 1WAR под управлением ОС МХТИ 32к надо отредактировать в 1WAR строки 5, 11, 20 и 53 – вместо кода D4 00 D5 50, означающего команду занесения в регистр R_{10} числа $50_{16} = 80_{10}$, надо сделать там такой же код с единичкой на конце: D4 00 D5 51. И в строках 75, 76 заменить WRITE(0, …) на WRITE(1, …), потому что в ОС МХТИ 32к дисплей считается каналом 1, а не 0. Такую замену ноликов на единички удобно делать редактированием в Фортране-85 с помощью оператора ОП. Надо только аккуратно двигать маркер направо клавишей → и впечатывать 1 тогда, когда над маркером появляется знакоместо нолика без самого нолика. Отредактированную ИП следует тем же Фортраном-85 записать «на МЛ». Полученная так версия 1WAR будет успешно работать в ОС МХТИ 32к.

Работа тест-программ в имитаторе

D3-28_test_028__KP-60000_N-5999.txt – стандартная тест-программа «028», проверяющая систему команд и ОЗУ объёмом 32 килобайта машины Д3-28. Это старый вариант теста; наличие расширенной до 128к памяти в нём не определяется. Запуск: <C>, <T>, <S>. Для работы теста «028» дисплей не нужен.

Тест состоит из четырёх блоков, и в реальности должен работать непрерывно: после 4-го блока сам запускается снова 1-й блок. Но в имитаторе после 1-го блока каждый раз происходит останов, так как имитатор не воспроизводит ту погрешность вычислений в тесте, которая должна возникать на реальной Д3-28. При этом на табло мы видим:

Y = .414990264375–08
X = .888000000000–08

Нижнее число (т. е. X) – значение погрешности, которое должно накопиться в реальной Д3-28 при многократных вычислениях с алгоритмами разных математических функций. Верхнее число – погрешность, накапливающаяся в имитаторе. Она примерно вдвое меньше той, что ожидается в тесте (потому что в имитаторе ещё не сделана точная имитация 12-разрядной математики реальной машины Д3-28), и это служит причиной остановки теста.

Поскольку это «осознанная ошибка», то ничего страшного для нас в ней нет: просто надо после каждой такой остановки нажимать два раза <S>. Тогда запускается продолжение теста, и дальше тест идёт как надо, с автоматическим возвратом к 1-му блоку после 4-го. Прохождение каждого из первых трех блоков индицируется на табло выводом номера блока в виде «цифр через одну»; через короткое время эти цифры меняются местами с погашенными разрядами. После 4-го блока тест своеобразно напоминает нам об историческом событии 1917 года.

028-test_KP-73370_N-8921.txt – новая версия теста «028», в ней проверяется также и наличие сегментов 128к-памяти. В первой части нового 028-теста в 14 различных подключениях сегментов проводится «пи-тест»: в регистр X заносится машинной командой PI число пи = 3.14159265359, и затем с ним последовательно выполняются команды INV, SQR, EXT, EXP, LOG, LGT, QRT, SIN, ASN, COS. ACS, TAN, ATN, INV. Получившийся в X результат сравнивается с числом в Y-регистре; это ожидаемый тестом результат: Y = 3.14159265370. Если X = Y, то «пи-тест» считается выполненным успешно, и 028-тест работает дальше; в противном случае происходит останов с ндикацией Y = 3.14159265370 и номера ошибки X = 4.

В имитаторе после «пи-теста» происходит останов. Если заменить команду занесения в X номера ошибки 4 командой STOP, чтобы в X сохранился результат вычислений, то мы увидим, что у нас X = 3.14159265358.Т.е. имитатор выдаёт результат, на одну единицу в 11-ом знаке после запятой отличающийся от исходного значения «пи». (При идеальной точности вычислений должно было бы получиться точно исходное «пи»). В тесте же ожидается на порядок менее точный результат. Таким образом, опять ничего страшного для нас в такой остановке теста нет; важно то, что имитатор способен делать вычисления хорошо. Хотя и неприятно, что эта «осознанная ошибка» в тесте повторяется 14 раз; приходится 14 раз нажимать <S>, прежде чем индицируется завершение первой части 028-теста.

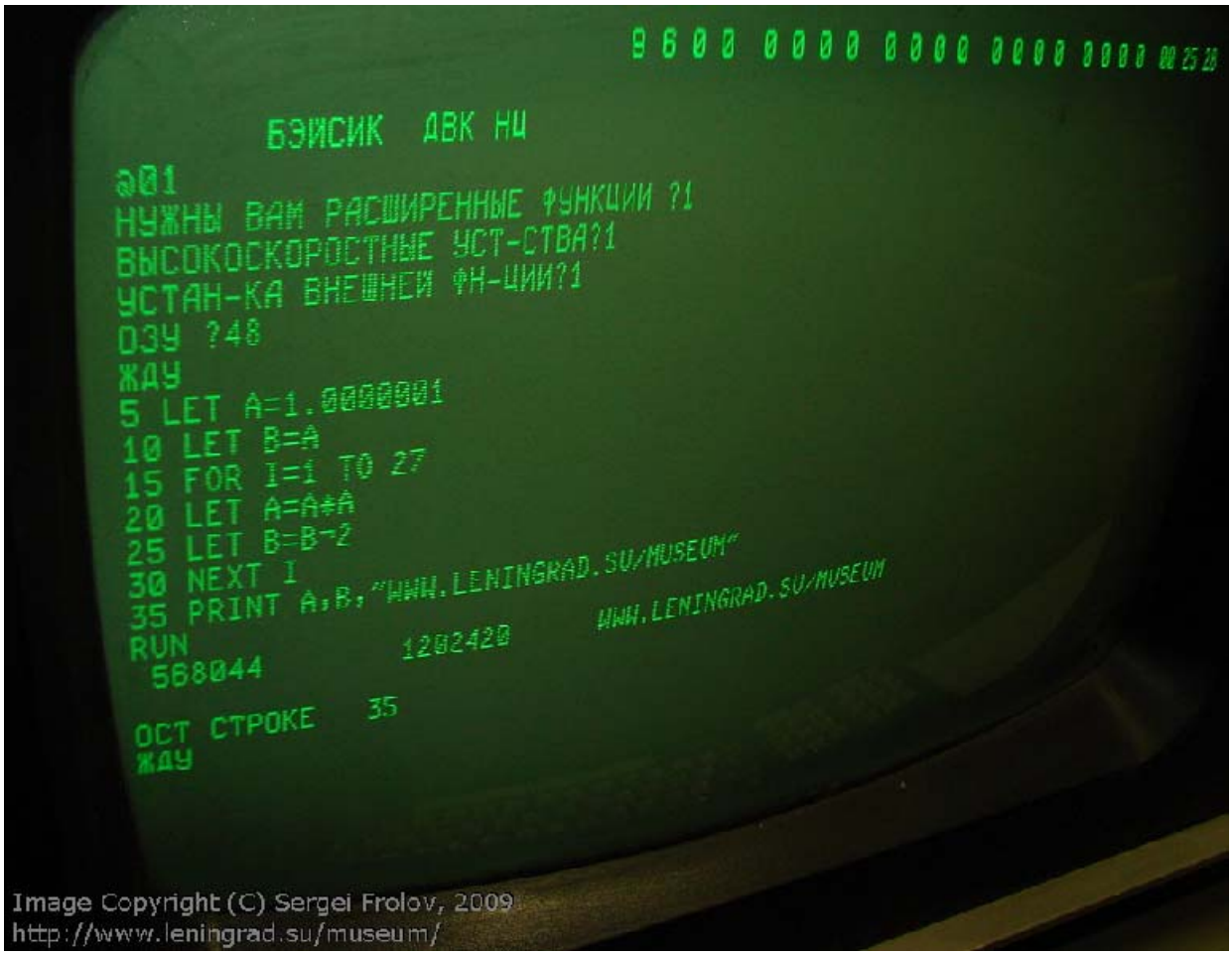
Во второй части нового 028-теста производится та же проверка арифметики, что и в старом 028-тесте; приходится нажать <S> два раза, прежде чем на табло будет индицировано завершение второй части. Третья и четвёртая части теста проходят в имитаторе без остановок (и затем всё повторяется заново). Этот тест в имитаторе правильно определяет объём памяти: «128к» по умолчанию. Если запустить имитатор в режиме «32к», то 028-тест видит, как и должно быть, 32 килобайта памяти. О работе этого теста рассказано в «*Инструкции по эксплуатации 3.857.100 ИЭ*» на стр. 54–55.

D3-28_test_017__KP-5895_N-527.txt – стандартная тест-программа, проверяющая работу НМЛ и пригодность магнитной ленты в «рабочей кассете»; см. стр. 54 «*Инструкции по эксплуатации 3.857.100 ИЭ*» Для работы этого теста дисплей не нужен. Запуск: <C>, <T>, <S>.

После запуска тест останавливается с индикацией числа 21 на обоих табло. Так программа напоминает нам о необходимости вынуть из НМЛ кассету с тестом и поставить чистую «рабочую кассету»! (Лучше это делать заранее, не дожидаясь напоминания).

После того как мы поставим «рабочую кассету», надо нажать <S>. Тест снова запускается и работает довольно долго. В имитаторе это проявляется как мигание НМЛ. В итоге на табло фиксируются числа: Y = 1, X = 0. В «Инструкции по эксплуатации» говорится, что такие числа означают полностью успешное прохождение теста.

SF-test_bas,txt – программа, позволяющая в имитаторе Д3-28 проверить вычислительную способность Бейсика (или, аналогично, – Фортрана либо Выстры, и т.д.) с помощью «теста Сергея Фролова». Листинг бейсик-версии виден на фотографии из статьи Википедии в Интернете о «фрязинском дисплее 15ИЭ-00-013»:



Число B приравнивается числу A = 1.0000001. Затем в цикле число A умножается 27 раз на себя (каждый раз результат заносится снова в A). Это эквивалентно возведению исходного значения в степень 134217728. Число B в том же цикле возводится в квадрат (каждый раз результат заносится снова в B). Оба способа должны были бы дать один и тот же результат. Однако, как видно на фотографии, первый способ дал 568044, а второй способ дал 1202420. (И оба эти результата грубо неверные, как показывает сравнение с приведённым ниже расчётом на современном ПК). Из начального диалога видно, что так печально обстоит дело в «Бэйсике ДВК НЦ»;

сокращение ДВК НЦ означает: Диалоговый Вычислительный Комплекс «Научный Центр» :-) (в этом комплексе работала машина Электроника-60, считавшаяся более совершенной, чем Д3-28).

Вот программка и современный результат такого же теста в «Маткаде» (на ПК с ОС Windows XP):

$$F(A) := \left| \begin{array}{l} B \leftarrow A \\ \text{for } J \in 1, 2 .. 27 \\ \quad \left| \begin{array}{l} A \leftarrow A \cdot A \\ B \leftarrow B^2 \end{array} \right. \\ X_0 \leftarrow A \\ X_1 \leftarrow B \\ X \end{array} \right.$$

$A := F(1.0000001)_0$ $\qquad$ $B := F(1.0000001)_1$

$A = 674530.4755$ $\qquad$ $B = 674530.4755$

Проверка: $(A)^{\frac{1}{134217728}} = 1.0000001$

Листинг и результат работы того же самого теста в имитаторе Д3-28:

```
ГОТОВ
:LOAD'SF-TEST'
:LIST

1 PRINT
5 LETA=1.0000001
10 LETB=A
15 FOR J=1 TO 27
20 LETA=A*A
25 LETB=B¬2
30 NEXT J
35 PRINT!6.4!'     'A'     'B
40 PRINT

:RUN

     674513.1185     674513.1185

ОСТАНОВ В СТРОКЕ  40
:_
```

Видно, что по сравнению с «Маткадом» «Бэйсик 3А для Д3-28» в имитаторе тоже врёт, но всё-таки первые 4 цифры выдаёт верные.

14. Приложение 4: фотографии

Электронная вычислительная машина (устройство специализированное управляющее вычислительное) «Электроника Д3-28»:

Image Copyright © Sergei Frolov, 2010
http://www.leningrad.su/museum/

Большое спасибо, **Sergei Frolov**! Эта фотография (и следующая тоже) существенно способствовала изготовлению программы-имитатора Д3-28.

В имитаторе изображение "машины Д3-28" не является фотографически точным. Оно построено из фрагментов фотографий, с ретушированием и с изменением пропорций – чтобы индикаторное табло выглядело покрупнее, и цифры на нём чтобы были видны подслеповатому пользователю (вроде меня) чётче.

Клавиатура дисплейного терминала «15ИЭ-00-013»:

Image Copyright © Sergei Frolov, 2009
http://www.leningrad.su/museum/

В имитаторе на этом фото я дорисовал часть корпуса блока логики терминала.

Ещё одна фотография ЭВМ «Электроника Д3-28»:

С этой фотографии для имитатора пригодилось изображение крышки НМЛ.

Ценный кадр – на этой фотографии машина Д3-28 изображена в режиме «Ввод»:

Фото https://pp.userapi.com/c314916/v314916604/8080/BcZZT7kK_xw.jpg из сообщения 24 августа 2017 https://www.phantom.sannata.org/viewtopic.php?p=246948#p246948
участника форума «Полигон Призраков» с ником **IdeaFix** (в дальнейшем Guest).

Нижнее табло показывает, что в ячейке с десятичным адресом 05684 записан байт 1515; верхнее табло изображает этот же адрес двумя байтами в тетрадно-десятичном виде: 0106 0304. (В hex-записи это означает, что по адресу 16 34 записан байт FF).

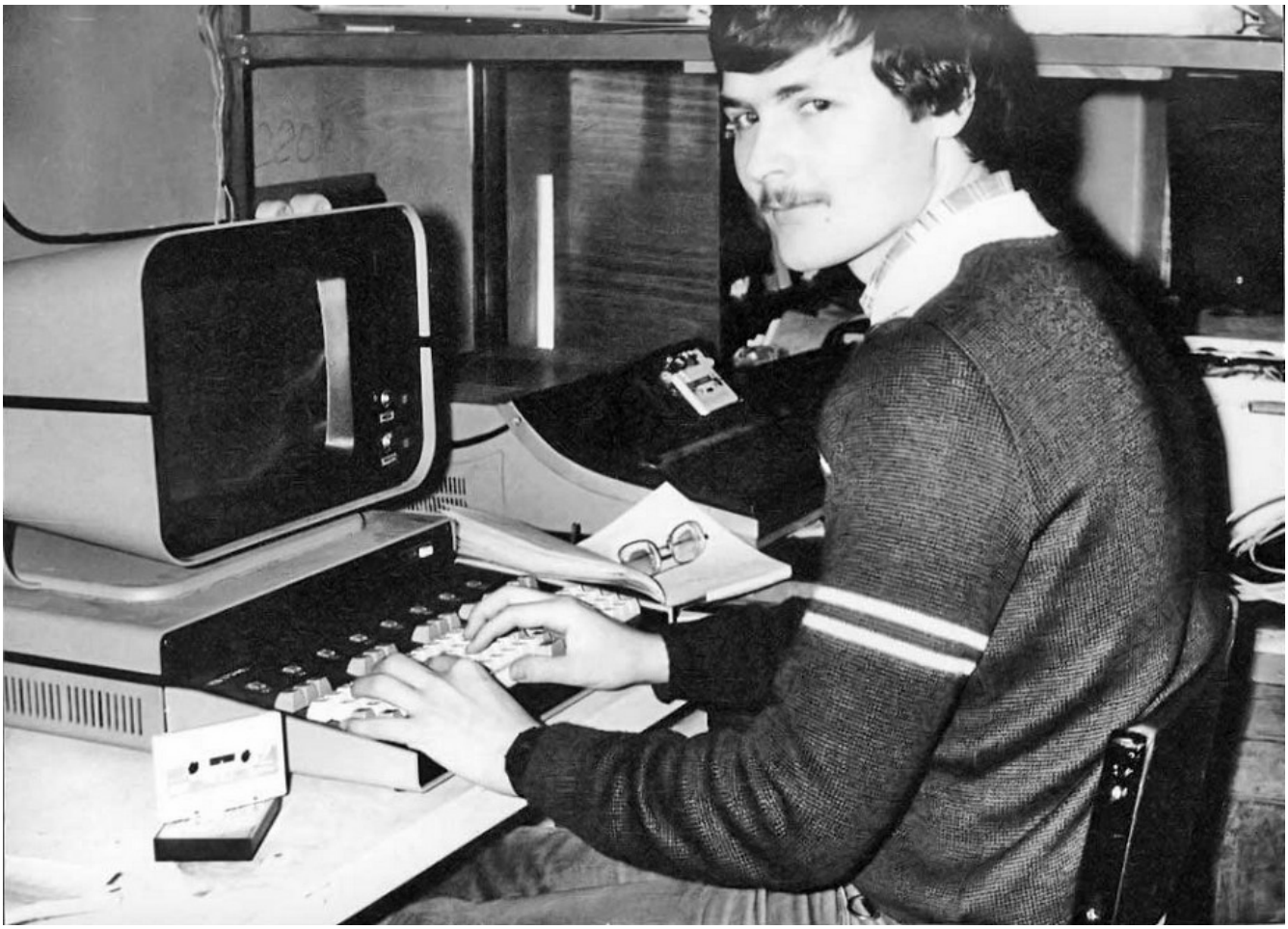
Вот такой комплекс пытается имитировать наш имитатор:

фото с сайта http://photo.sstu.ru/main.php?g2_itemId=14117&g2_imageViewsIndex=1

Слева – ТПУ (похожее на 15ВВП80-002, но я не уверен, что это именно оно; ниже будет фото с девушкой, вот там точно оно). Справа – клавиатура терминала 15ИЭ-00-013, за ней – «блок логики» терминала, на нём стоит ЭВМ Д3-28, а на ней – собственно «фрязинский дисплей».

В завершение – ещё две ностальгические фотографии из уже далёких 1980-х годов. Эти изображения не нашли применения в программе; но они передают атмосферу лаборатории политехнического вуза того времени. Обе фотографии найдены в сети в свободном доступе; авторам большое спасибо! (Сюда эти фотографии помещаю без разрешения авторов, поэтому не сопровождаю их подробным описанием и персональной информацией.)

Вот здесь мы видим комплекс «Д3-28 с терминалом 15ИЭ-00-013» не на музейной витрине, а в его «природной среде обитания»:



А здесь надо взглянуть ещё и на самый верх фотоснимка – на полке видно (жаль, что не полностью) ТПУ 15ВВП80-002. Это одна из немногих фотографий, найденных в сети, с довольно чётким изображением точно такого же ТПУ, какое мне запомнилось в те давние времена.

Зима 2023. Sinus

Река времён в своём стремленьи
Уносит все дела людей ...

Г. Р. Державин
