Загрузка ПО в ЭВМ Д3-28 через шину ВВОД-ВЫВОД
(Пока это ещё неопробованный проект. Версия 2.08.2022 **)**

СОДЕРЖАНИЕ

0.	О чём вообще речь (постановка задачи)	1
1.	Загрузка программ в Д3-28 методом "m-code"	4
2.	Интерфейсное устройство для создания "m-code" из "s-code"	9
3.	Как wav-файлы создаются в имитаторе Д3-28 версии 1.2	13
4.	Инструкция по изготовлению wav-файлов диалоговыми программками	40
5.	Загрузка блочных программ на примере ОС ЛФТИ 32к и ВТ МХТИ 128С	48
6.	Загрузчик m-code. Тестирование wav-файлов с m-code и s-code в имитаторе	52
7.	Приложение 1: принципиальная схема интерфейсного устройства	60
8.	Приложение 2: схема двухканальной записи на МЛ	64

0. О чём вообще речь (постановка задачи)

Речь идёт о загрузке программного обеспечения (ПО) в оперативную память (ОЗУ) ЭВМ "Электроника Д3-28" с последующей записью этого ПО на магнитофонные кассеты, подобные МК-60, имеющимся в Д3-28 накопителем на магнитной ленте (НМЛ). Подразумеваются те полностью работоспособные экземпляры Д3-28, для которых их "родные" кассеты не сохранились или изначально отсутствовали. Они могут таким образом обзавестись ПО.

Работу с магнитными дисками и перфолентами не рассматриваю, так как она требует внешнего оборудования более сложного, чем то, о котором у нас пойдёт речь. Без дополнительного оборудования Д3-28 приспособлена к работе с магнитной лентой – на пульте Д3-28 имеются кнопки ЗЛ и СЛ, которыми запускаются запись программ из ОЗУ на МЛ и чтение с МЛ в ОЗУ. Действия с встроенным в Д3-28 НМЛ включены в систему команд Д3-28 и, соответственно, – во многие "легендарные" программы (Бейсик, Фортран и др.), как их неотъемлемая часть. Именно поэтому даже с учётом, может быть, сложности восстановления НМЛ, нежелателен полный отказ от НМЛ при изучении старинного ПО на реальных экземплярах Д3-28.

ПО для Д3-28, существовавшее на МЛ, теперь в доступном виде хранится в текстовых файлах, полученных мной раскодированием оцифровок МЛ. Для ясности и полноты картины, сложившейся к данному моменту, перечислю эти оцифровки. Множество оцифровок выполнил **Виталий Васильевич Колесник**; о первоначальной технике оцифровки он рассказал на страницах созданного им сайта d3-28.ru: Оцифровка магнитофонных кассет с программами Д3-28. В дальнейшем Виталий Васильевич приблизил схему оцифровки к схеме чтения МЛ в Д3-28 – применил в роли усилителя воспроизведения операционный усилитель без разделительных конденсаторов и без стандартных для бытовых магнитофонов (м/ф) цепей коррекции амплитудно-частотной характеристики; см. его статью. Ряд программ (версии Бейсика, Фортран, Выстра и др., см. папку D3-28_wav) я оцифровал на м/ф "Романтик М-309С-1" со стандартной АЧХ. Аналогично выглядят оцифровки Бейсика в wav-файле, предоставленном **Sergei Frolov**, и оцифровки программ для Д3-28 с 16-килобайтным ОЗУ (15ВСМ-0.01-001), выполненные форумчанином **mig_25**; см. также здесь.

Возможен вопрос: почему бы не записать обратно на кассеты бытовыми магнитофонами прямо эти оцифровки, если ставится цель загружать данное ПО с кассет? Ответ: форма сигналов в оцифровках не подходит для такой цели.

На рис. 1 показаны графики желаемой (и реализованной в Д3-28) зависимости от времени для тока в катушках записывающей двухканальной (стерео) головки. Биты записываются просто переключением полярности тока. Током порождается магнитное поле в зазоре головки, оно намагничивает ленту, поэтому намагниченность двух дорожек МЛ приобретает такую же форму, как и у токов канала нулей и канала единиц. При чтении те участки МЛ, на которых уровень намагниченности постоянен, не вызывают появления сигналов в катушках магнитной головки. Сигнал (напряжение) читающих катушек пропорционален производной магнитного поля по времени – пропорционален скорости изменения намагниченности МЛ, движущейся вблизи зазора головки в процессе чтения. Т. е. нужные для схемы чтения импульсы напряжения нулей и единиц создаются только резкими перепадами намагниченности:

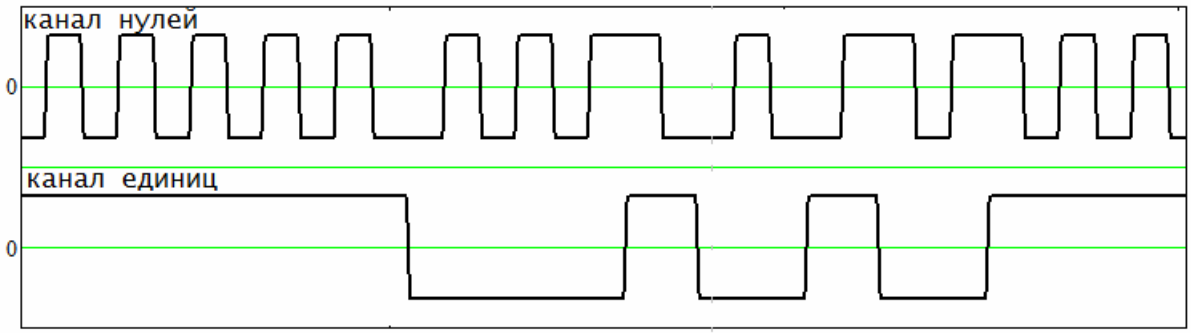


Рис. 1 Изменение во времени токов магнитной головки в Д3-28 в режиме записи. В результате, так же скачкообразно изменяется направление намагниченности на дорожках МЛ.

На рис. 2 показан соответствующий фрагмент реальной оцифровки, полученной на бытовом м/ф (пример взят из wav-файла с оцифровкой программы Fortran-5M). Видно, что не все перепады получились одинаковыми по величине, – потому что длительные ненулевые постоянные уровни не воспроизводятся (и не записываются) бытовым м/ф:

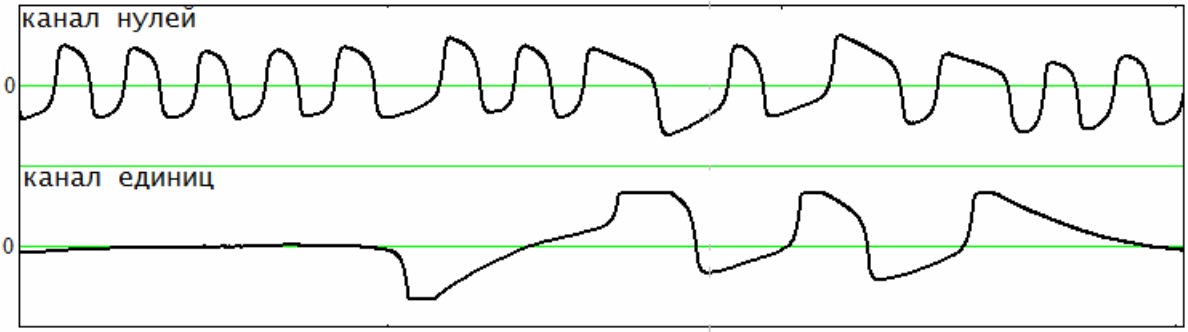


Рис. 2 Графики сигнала из оцифровки бытовым м/ф.

При оцифровке без коррекции АЧХ усилителя воспроизведения форма сигнала ещё больше отличается от показанной на рис.1: сигнал имеет вид Λ-образных импульсов, расположенных на месте перепадов намагниченности МЛ. Его график выглядит примерно так, как на рис. 3, где показана производная сигнала с рисунка 2:

Рис. 3 Примерно это даёт оцифровка через ОУ без коррекции АЧХ (т. е. с плоской АЧХ).

Усилитель записи бытового м/ф со стандартной АЧХ поднимает низкие и заваливает высокие частоты. Значит, если подавать на линейный вход записывающего м/ф просто напряжение из оцифровок, то мы получим сглаженные перепады; т. е. форма тока записи (и намагниченности МЛ) будет отличаться и от показанной на рис. 2 или 3, и не примет желаемого вида типа рис. 1.

Сигналы идеальной формы (типа рис. 1) можно синтезировать в компьютере. Если затем записать их на МЛ бытовым м/ф, то прочтётся ли такая запись машиной Д3-28? Этот вопрос рассмотрен на форуме "Полигон Призраков" в теме Свисток для Д3-28 (программа для создания WAV-файлов в формате Д3-28). Не пересказывая всё, процитирую итог обсуждения, высказанный автором темы **xoiss**: "с обычным магнитофоном – тупик". Проблема связана с неспособностью м/ф записывать постоянные ненулевые уровни. Из-за этого необходимые перепады сигнала не имеют стабильной достаточно большой величины. Кроме того, на форму записанного сигнала влияет автоматическая установка уровня записи, система шумопонижения (если она есть), наклон зазора магнитной головки, – факторы, индивидуальные в разных м/ф. К нечитаемости МЛ в Д3-28 может привести также различие скоростей движения ленты в Д3-28 и в бытовых м/ф или путаница в нумерации каналов.

От перечисленных недостатков свободна запись программ из ОЗУ машины Д3-28 на МЛ посредством её "родного" НМЛ. (Предварительно исправность НМЛ легко проверить записью короткой программки, набираемой вручную с пульта Д3-28.) В этом варианте надо только решить задачу первоначальной загрузки ПО в ОЗУ Д3-28. Одно из возможных решений – с условным названием "метод m-code" (и его реализация в форме "s-code") – подробно рассматривается ниже.

Другой вариант – запись синтезированного сигнала (типа рис. 1) из плеера или компьютера на МЛ в отдельном записывающем "кассетнике" или лентопротяжном механизме (ЛПМ) со стерео-головкой, включённой так же, как в Д3-28. На таком кассетнике можно тиражировать кассеты с ПО; однако их читаемость в НМЛ разных экземпляров Д3-28 не гарантируется.

В обоих вариантах источником ПО служит, как уже упоминалось, доступный архив текстовых (txt) файлов с раскодировками оцифровок МЛ . (О деталях процесса раскодировки можно прочитать в материалах на сайте d3-28.ru. Первые отчёты (frmj_tape_reading.pdf и basic-texts_tape_reading_7-04-2017.pdf) с описанием раскодирования есть на "Полигоне" в этом и в следующем сообщениях.) Txt-файлы с раскодировками оцифровок В. В. Колесника доступны в Архиве программ Д3-28 на сайте d3-28.ru; некоторые из них включены мной в zip-архив с имитатором Д3-28 и продублированы в папке D3-28 на яндекс-диске. В указанной папке также есть txt-файлы с остальными раскодировками – версии Бейсика, Фортран, Выстра, входящие в архив с имитатором, и др.

Таким образом, ставятся две взаимосвязанные задачи – "железячная" и "софтовая":

- создать не слишком сложную схему первоначального ввода в Д3-28 сигнала, содержащего информацию из того или иного txt-файла с ПО для Д3-28;
- создать программу для преобразования txt-файла в wav-файл с указанным сигналом на персональных компьютерах (ПК) пользователей, заинтересовавшихся этой темой.

Такая постановка задач не нова, открытия тут я не сделал. Варианты решения 1-й задачи обсуждались на "Полигоне" в теме Д3-28, устройства для загрузки ПО (способы, идеи, ... конструкции). Вариант (и критика) решения 2-й задачи дан форумчанином **xoiss** в созданной им теме Свисток для Д3-28 (программа для создания WAV-файлов в формате Д3-28) – программа "MK60" на Питоне. Насколько мне известно, с реальными машинами Д3-28 ничего из этого не проверялось.

Ниже речь идёт о следующих предполагаемых решениях указанных двух задач:

- об одной конкретной схеме загрузки ПО из wav-файлов в машину Д3-28 **через её разъём ВВОД / ВЫВОД**, предназначенный для работы с периферийными устройствами; и:
- о преобразовании txt-файлов с ПО для Д3-28 в wav-файлы специальными командами **в имитаторе** (симуляторе Д3-28) **версии 1.2**.

1. Загрузка программ в Д3-28 методом "m-code"

Что такое "m-code" (м-код)? В технической литературе для сигналов этого типа существуют строгие определения, но здесь я пользуюсь, для краткости, придуманными мной условными названиями. Буква **м** в "м-код" напоминает нам о терминах "**м**анчестерский код", "**м**анипуляция фазой", "**м**еандр".

Параметры одноканального сигнала типа "м-код" и их обозначения (см. рис. 4 и 5):

Частота дискретизации ***Fd*** (то же, что *sample rate*) – количество отсчётов (сэмплов) цифрового сигнала, передаваемое за 1 секунду. Величина 1 / *Fd* есть длительность отсчёта.

Период – последовательность отсчётов, передающая один бит. Период характеризуется количеством отсчётов, обозначаю его как ***dM***. Длительность периода есть *dM* / *Fd*.

Перепад (то же, что фронт или спад импульса в сигнале) – переход в сигнале от высокого уровня к низкому или наоборот. Количество отсчётов в перепаде обозначаю как ***dF***.

Постоянный уровень – характеризуется количеством отсчётов ***dC*** и своей величиной. Если постоянный уровень не равен нулю, то его величину называем амплитудой, ***Ampl***.

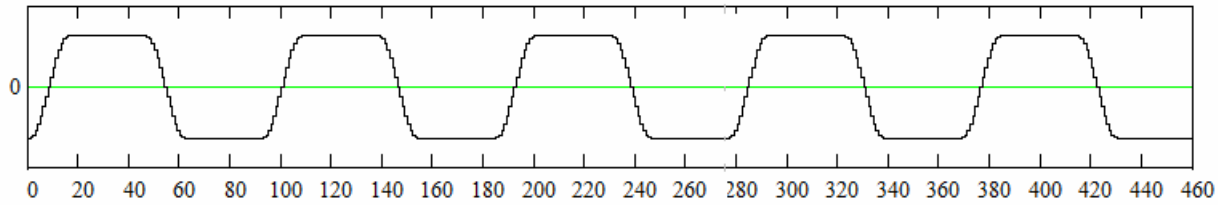


Рис. 4. Пример: пять периодов с битами "0" в начале сигнала. Здесь каждый период содержит *dM* = 92 отсчёта, каждый перепад содержит *dF* = 16 отсчётов, каждый участок с постоянным уровнем содержит *dC* = 30 отсчётов. Номера отсчётов указаны на горизонтальной оси графика.

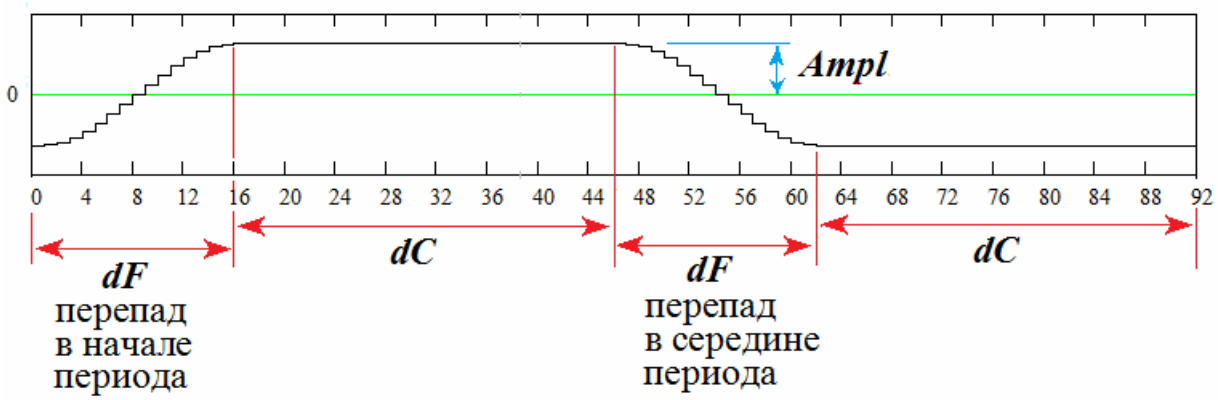


Рис. 5. Растянутое изображение одного периода, содержащего *dM* = 92 отсчёта.
В сигнале типа m-code параметры периода *dM*, *dF* и *dC* связаны соотношением:
dM* = 2·*dF* + 2·*dC.

Картина типа рис. 4 может встретиться и в одном из каналов обычного 2-канального сигнала Д3-28. Например, так может выглядеть серия 10 бит "0" в канале нулей; при этом в канале единиц будет всё время удерживаться постоянный ненулевой уровень. Так же может выглядеть и серия 10 бит "1" в канале единиц; при этом уровень будет постоянным в канале нулей. Важно, что в обычном 2-канальном сигнале Д3-28 информацию о бите несёт каждый перепад. При этом показанный на рис. 4 пример, где центры соседних перепадов разделены 46 отсчётами, будет характерен для оцифровок кассетными м/ф с их стандартной скоростью движения МЛ и со стандартной в аудио-файлах частотой дискретизации $Fd = 44100$ Гц. Такая картина соответствует времени передачи одного бита $46 / Fd \approx 0.001$ сек; т. е. скорость передачи ≈ 1000 бит / сек, – значение, по порядку величины близкое к 100 байт / сек.

Явных данных о скорости движения МЛ в Д3-28 я не видел в документации. Вот данные из "И5М3.857.100 ТО": скорость чтения-записи – около 445 байт/сек, объём информации на одной стороне кассеты типа МК-60 – до 200 Кбайт. Из этого можно заключить, что в Д3-28 время чтения одной стороны МЛ примерно в 4 раза меньше, чем характерные для бытового кассетника 30 минут; т. е. скорость протяжки ленты в Д3-28 примерно в 4 раза больше, чем в обычном ЛПМ. (Такое различие может оказаться важным в опытах по тиражированию кассет с ПО для Д3-28 на внешнем пишущем кассетнике; и особенно – при эмуляции НМЛ).

Различие между "м-кодом" и "2-канальным обычным сигналом Д3-28" (так условимся называть картину типа рис. 1) можно пояснить следующим образом:

1. В случае сигнала с м-кодом, помимо того, что он 1-канальный, только один перепад в периоде несёт информацию о бите – это перепад в середине периода. Условимся, что перепад (в середине периода) от верхнего уровня к нижнему, т. е. от значения $Ampl > 0$ к $(-Ampl)$ на рис. 5, означает бит "0". Перепаду же в середине периода от нижнего уровня к верхнему соответствует бит "1". Кроме того, для понимания работы программок синтеза и чтения м-кода полезно следующее соображение.

На стыках периодов, т. е. в начале очередного периода, перепад может быть, а может его и не быть. Дело в том, что если очередной период несёт такой же бит, как бит в предыдущем периоде, то для установки нужного исходного уровня необходим начальный перепад, а если биты разные, то в начале очередного периода на участке dF просто продолжится конечный уровень предыдущего периода. Поскольку информационным мы считаем перепад в середине периода, то перепад в начале периода, если он есть, информационным не является.

На рис. 6 для примера м-кода показан фрагмент записи с программкой y-test__KP-88_N-8:

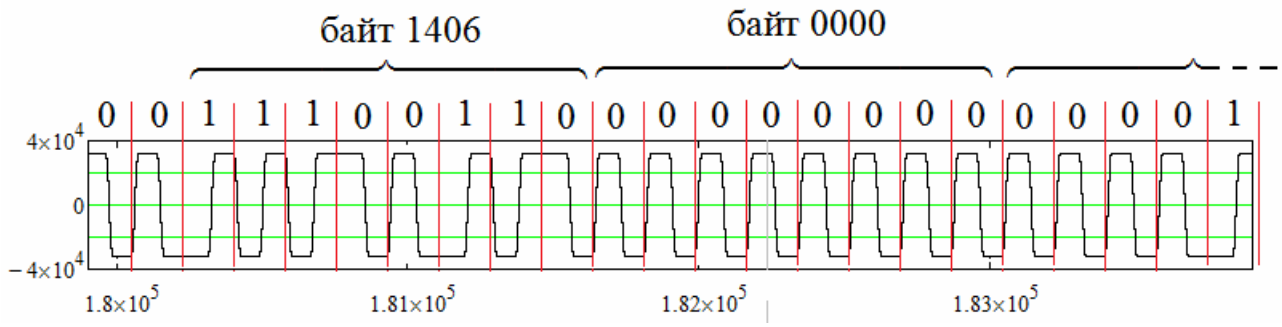


Рис. 6. Пример м-кода. Красными чёрточками показаны границы периодов. Над перепадами в середине периодов показана их интерпретация – биты 0 или 1. Сигнал начинается серией 128 нулевых байт; на рисунке видны два бита 0 из этой серии. Затем идёт синхробайт 1406 (14 06 в тетрадно-десятичной записи, принятой в документации к Д3-28; его hex-запись: E6; двоичная запись: 11100110_2). Затем идут два байта (на рисунке поместился только первый из них) 0000 и 0008 – это номер шага с командой END при нумерации шагов с нуля, так что 8 + 1 = 9 есть количество байт в данной программке. И затем в сигнале аналогичным образом кодируются байты самой программки. Так сигнал строится "по умолчанию".

2. Поскольку сигнал с м-кодом будет обрабатываться пользовательской программой-загрузчиком в ОЗУ Д3-28 (а не микропрограммой чтения МЛ, зашитой в ПЗУ Д3-28), то скорость загрузки ПО из сигнала с м-кодом будет ниже, чем при обычной загрузке с МЛ; она определяется количеством и временем выполнения используемых в загрузчике команд.

В частности, оценка для загрузчика "loader_m-code_addr-x_kp-1812_n-138.txt" показывает, что предполагаемое время приёма одного бита $\approx$ 0.004 сек. Это оценка длительности одного периода. Тогда при выборе $Fd = 44100$ Гц количество отсчётов в периоде dM следует выбрать равным $0.004 \cdot Fd \approx 176$. В частности, $dM = 176$ получается при $dF = 23$, $dC = 65$. Эти значения параметров задаются по умолчанию в нашем виртуальном "Устройстве синтеза wav-файлов", включённом в имитатор Д3-28 версии 1.2. Их изменение доступно в программках для имитатора Д3-28 версии 1.2, которые пользователь может составлять с целью управления указанным "Устройством", как периферийным устройством, якобы подключенным к разъёму ВВОД / ВЫВОД имитируемой машины. В архиве с имитатором версии 1.2 в папке u_soft есть готовые примеры таких программок; они рассматриваются ниже, в разделах 3 и 4.

3. Двухполярный сигнал с м-кодом (как в примере на рис. 6) практически не содержит постоянной составляющей и поэтому он не требует от тракта передачи способности передавать сколь угодно длительный ненулевой уровень. В этом и состоит преимущество такого сигнала. Однако здесь нужно сделать оговорку. Если на практике окажется, что Д3-28 сможет принимать м-код лишь с очень малой скоростью, то нам придётся выбрать большую длительность dM и, значит, большую длительность участков dC. В таком случае, хотя в среднем по времени у сигнала по-прежнему не будет постоянной составляющей, будут присутствовать чередующиеся длительные участки уровня $Ampl$ и уровня $-Ampl$, и проблема их передачи через аудио-тракт плеера или компьютера снова возникнет. О решении этой проблемы с помощью "s-code" речь пойдёт в следующем разделе, а пока продолжаю описывать сигнал с м-кодом.

Упомянутый синтезатор wav-файлов записывает отсчёты сигнала как двухбайтовые целые числа со знаком. Поскольку двухбайтовое целое число со знаком не может превышать 32767, то этим значением определяется допустимая максимальная величина $Ampl$. Например, у показанной на рис. 6 записи $Ampl = 32000$. В упомянутых программках управления синтезом wav-файлов значение $Ampl$ (т. е. "громкость" сигнала), как и значения остальных перечисленных выше параметров, можно в разумных пределах задавать на своё усмотрение.

О форме перепада: почему перепад выбран плавным, а не резким? Спектр частот резких перепадов (т. е. спектр сигналов с прямоугольными, ступенчатыми импульсами) очень широкий, а полоса пропускания любых аудио-трактов (в плеерах, компьютерах, м/ф) ограничена не только снизу, но и сверху. Так что, слишком широкополосный сигнал всё равно исказится аудио-трактом: перепады будут как-то завалены, испорчены. То же самое другими словами: для фильтров, имеющихся в аудио-трактах, резкий перепад сигнала это "удар", возбуждающий в тракте переходные процессы, колебания, – они искажают форму сигнала. С этой точки зрения плавные сигналы предпочтительнее.

В упомянутом выше виртуальном "Устройстве", предназначенном для синтеза wav-файлов, доступно программирование формы перепада пользователем. Перепад задаётся для середины периода формулой типа $Ampl \cdot f(n)$, где $n = 0, 1, 2, \ldots, dF-1$. Пользователь в своей "Д3-28-программке" описывает функцию $f(n)$ и программирует вычисление её значений и их вывод в "Устройство" в цикле по n. В случае м-кода перепад в начале периода, если он есть, "Устройство" затем вычисляет как $-Ampl \cdot f(n)$. Таким путём можно для экспериментов задать и резкий перепад, выбрав ступенчатую функцию; например: $f(n) = 1$ при $n < dF / 2$ и $f(n) = -1$ при остальных значениях $n < dF$. На приведённых выше рисунках перепад даётся функцией $f(n) = \cos(\pi \cdot n / dF)$. Для s-кода, о котором речь идёт в разделе 2, будет $f(n) = \sin(2 \cdot \pi \cdot n / dF)$.

Принцип загрузки программы из сигнала с м-кодом:

ЭВМ Д3-28 на своей задней стенке имеет разъём с байтовой шиной ВВОД. На 8 линий этой шины можно из периферийного устройства (ПУ) подавать однополярные сигналы ТТЛ и принимать их в ОЗУ Д3-28 командами ввода, доступными в пользовательских программках. В этом разъёме есть контакты и других шин (ВЫВОД, УПР, линии синхросигналов Вв, СИП, СИМ, разумеется есть и "земля" 0В), но пока сосредоточимся только на подаче в Д3-28 сигнала по одной линии (и 0В) – по линии младшего бита шины ВВОД; он обозначается как "Вва1". Точнее говоря, сигналы на контактах разъёма ВВОД-ВЫВОД должны быть логически инвертированными, инверсия обозначается чертой над наименованием сигнала; см., например, краткий пересказ сведений о "Вводе-выводе" с цитатами из документации; этот нюанс мы учтём в схеме интерфейсного устройства, а пока не будем обращать на него внимания.

Из мобильного плеера (или из ПК и т. п.), проигрывающего wav-файл, через выход на наушники сигнал с м-кодом следует подать сначала на самодельное интерфейсное устройство (далее обозначаю его как ИУ). Амплитуда сигнала на выходе плеера зависит от типа плеера, от регулировки громкости в нём, и, скорее всего, не превышает 1 вольта. ИУ должно преобразовать двухполярный сигнал плеера в однополярный сигнал ТТЛ – в сигнал для транзисторно-транзисторной логики с напряжением питания +5 В, типа микросхем серии К155: уровень логического "0" не более 0.4 В, уровень логической "1" не менее 2.4 В. Такое напряжение с выхода ИУ следует подать в машину Д3-28 на её вход "Вва1" (с "землёй" 0В).

Предположим, для определённости, что значению *–Ampl* из записи с м-кодом плеер с ИУ на ТТЛ-выходе ИУ сопоставляет логический 0, а значению *Ampl* сопоставляет логическую 1.

Структуру программы-загрузчика задаёт пользователь. Рассмотрим, не вникая в детали, основные действия загрузчика, выполняющего последовательно три задачи: обнаружение синхробайта, чтение количества байт для загрузки, и загрузку этих байт в ОЗУ с начального адреса 0. Синхробайт необходим для правильной разбивки битового потока на байты.

После запуска загрузчика, на стадии инициализации, команда INPO в нём настраивается на приём с шины ВВОД одного байта в однобайтовый регистр (в ячейку памяти в служебной зоне ОЗУ) с именем, например, S3. В регистр S0 записывается двоичный код 00000001, это маска для выделения младшего бита в S3 командой (S0 AND S3) → S3. Регистр S1 очищается нулями: 00000000; в него загрузчик будет добавлять один за другим принимаемые биты. В регистр S7 будет на временное хранение посылаться содержимое S3.

Допустим, плеер теперь воспроизводит начало wav-файла, там закодированы начальные нули. В некоторый момент времени загрузчик принимает в S3 байт ВВОД, и после выделения младшего бита, т. е. после очистки ненужных разрядов нулями, посылает значение Вва1 в S7. Затем быстренько в S3 ещё раз принимается Вва1, и загрузчик сравнивает S3 с S7. Если S3 = S7, то, перепад ещё не пришёл; тогда загрузчик ещё раз аналогично принимает Вва1 в S3 и сравнивает с S7. Загрузчик продолжает это делать раз за разом, пока перепада нет.

В какой-то момент времени приходит перепад, уровень на линии Вва1 меняется, и загрузчик это обнаруживает – свежепринятое значение бита Вва1 в S3 оказывается не равным ранее сохранённому в S7. Загрузчик не знает, является ли обнаруженный перепад информационным, но на всякий случай считает его информационным и посылает принятый младший бит S3 в копилку S1. В принципе, эта посылка должна бы выглядеть так: командой умножения на 2 содержимое S1 сдвигаем влево (младший бит в S1 при этом становится равным нулю) и добавляем младший бит S3 командой (S1 OR S3) → S1. На практике всё немного сложнее, так как в Д3-28 нет команд умножения для однобайтовых регистров, есть для двухбайтовых; но это уже не принципиальный нюанс. Существенно однако, что из-за подобных нюансов время выполнения нужных нам действий в Д3-28, увы, довольно велико.

Затем изменившееся таким образом содержимое S1 загрузчик сравнивает с синхробайтом 11100110_2; если совпадения нет, то сравнивает и с инвертированным синхробайтом 00011001. Если совпадений нет (их и не должно быть, пока в плеере звучат начальные нули), то загрузчик переходит к повторению указанной выше процедуры обнаружения перепада и добавления очередного бита в копилку S1. Но перед этим должно пройти время, равное длительности трёх четвертей периода: $(3/4)\cdot dM / Fd$. Такая задержка необходима потому, что обнаруженный перепад загрузчик считает информационным, и, значит, возможный перепад через полпериода необходимо игнорировать как неинформационный. Вот поэтому только после задержки более чем на полпериода (но меньше чем на период) загрузчик обновляет S7 и ищет новый перепад. (В Д3-28, наверное, придётся поэкспериментировать с циклом задержки, так как и без него значительная задержка набирается из-за медленности выполнения команд.)

Рассмотрим случай, когда первый обнаруженный загрузчиком перепад оказался перепадом на стыке периодов с начальными нулями; т. е. это неинформационный перепад, а загрузчик посчитал его информационным перепадом от логического 0 к логической 1. Тогда в копилку S1 занесётся эта логическая 1. Пока звучат начальные нули, каждый следующий обнаруживаемый перепад будет таким же, и в копилке раз за разом будет накапливаться байт 11111111 вместо, казалось бы, должного 00000000. Страшна ли эта "ошибка загрузчика"?

Вернёмся к рис. 6. Вот загрузчик добирается до перепада, отмеченного на рисунке левой красной чертой, принимает его за 1, и через 3/4 периода снова начинает принимать Вва1. Видно, что он сколько-то раз получит равенство S3 = S7 = 0. А затем он обнаружит перепад к логической единице, который действительно является информационным: это старшая единица из синхробайта 11100110. Значит, дальнейшие принимаемые перепады тоже будут информационными, и таким образом загрузчик накопит в регистре-копилке весь синхробайт. Легко проследить, что такой же результат будет и в том случае, если с самого начала загрузчик обнаружит перепады от 1 к 0 и тем самым интерпретирует звучание начальных нулей именно как нулевые байты. Таким образом, "ошибка" в поиске синхробайта не возникает.

Наконец, аналогично легко рассмотреть случай, когда в плеере с ИУ происходила инверсия полярности сигнала, так что на выходе ИУ логические 0 и 1 поменялись местами. В этом случае загрузчик примет инвертированный синхробайт. Тогда в регистр S6, инициализированный нулём 00000000_2, загрузчик посылает байт 11111111_2 и затем загрузчик будет инвертировать каждый принятый байт в накопителе байта S1 командой (S6 XOR S1) → S1. Если же инверсии сигнала нет, так что значение S6 остаётся равным нулю, то эта команда не изменяет S1. Таким способом удаётся избавиться от необходимости следить за возможной инверсией сигнала где-либо в тракте передачи – метод м-кода оказывается нечувствительным к инверсии полярности.

После обнаружения синхробайта в загрузчике снимается признак необходимости искать синхробайт, и начинает работать счётчик принимаемых бит, чтобы выявлять факт накопления в S1 целого байта. Следующие принятые за синхробайтом два байта загрузчик интерпретирует как данные о количестве байт, подлежащих загрузке в ОЗУ, – эти данные поступают в двухбайтовый регистр-счётчик для цикла загрузки. Затем загрузчик выполняет этот цикл – загружает принимаемые байты подряд в ячейки ОЗУ, начиная с нулевого адреса.

Так работает программка "loader_m-code_addr-x_kp-1812_n-138.txt", загружающая в Д3-28 информацию из wav-файлов, структура которых в нашем "Устройстве" синтеза wav-файлов задана по умолчанию. "Устройство" синтеза wav-файлов позволяет строить файлы с м-кодом и по-другому: пользователь может изменить количество начальных нулей, задать другой синхробайт, добавить адреса посадки в ОЗУ (например, вместо количества байт), контрольную сумму и любую иную информацию. В этом случае Устройство запишет в wav-файл содержимое Буфера как есть, без "добавок по умолчанию". При этом пользователь может соответственно модифицировать указанную программку-загрузчик или заново сочинить свою собственную.

2. Интерфейсное устройство для создания "m-code" из "s-code"

Интерфейсное устройство (ИУ), описываемое ниже, формирует нужный для Д3-28 однополярный ТТЛ-сигнал с м-кодом из проигрываемого плеером сигнала с "s-code". Буква **s** в термине "s-code" напоминает нам об импуль**с**ной, **s**inus-образной форме сигнала на участках, которые в м-коде мы называли перепадами.

Сигнал с s-кодом хорош тем, что он вообще не содержит ненулевых постоянных уровней, и поэтому он может без проблем передаваться любым аудио-трактом даже с очень малой скоростью передачи. Соответствие между м-кодом и s-кодом пояснено на рис. 7:

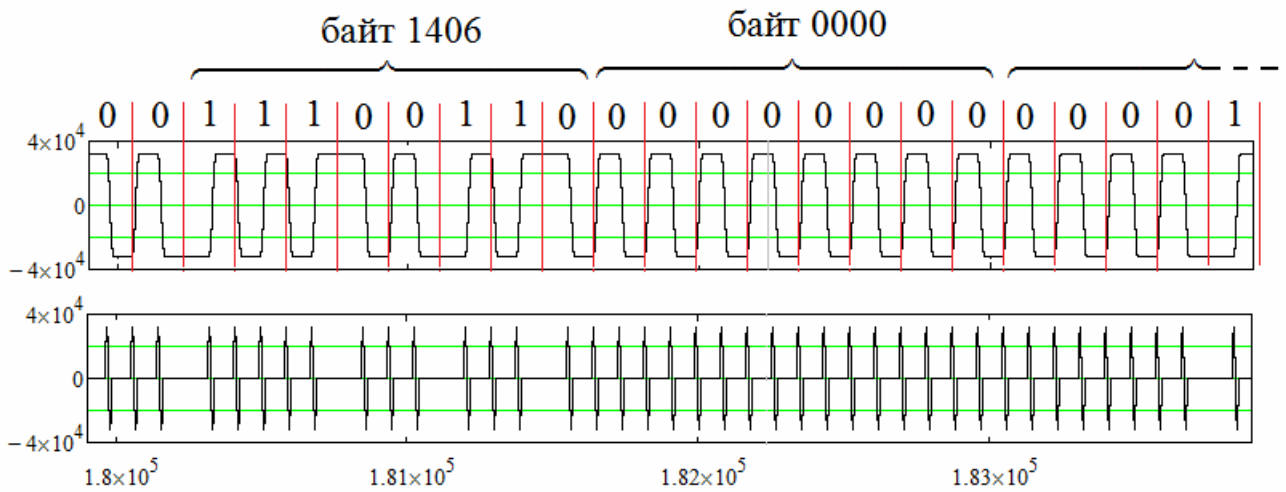


Рис. 7. На верхнем графике показан тот же фрагмент wav-данных с м-кодом, что и на рис. 6. На нижнем графике – соответствующая wav-запись с s-кодом. Здесь (а также на рис. 8) видно, что на месте каждого перепада от одного уровня к другому в сигнале с s-кодом располагается синусообразный импульс, причём постоянный уровень между импульсами – нулевой.

Для параметров одноканальной записи с s-кодом оставим те же обозначения (и термины), которые в предыдущем разделе были введены для м-кода; они поясняются рисунком 8:

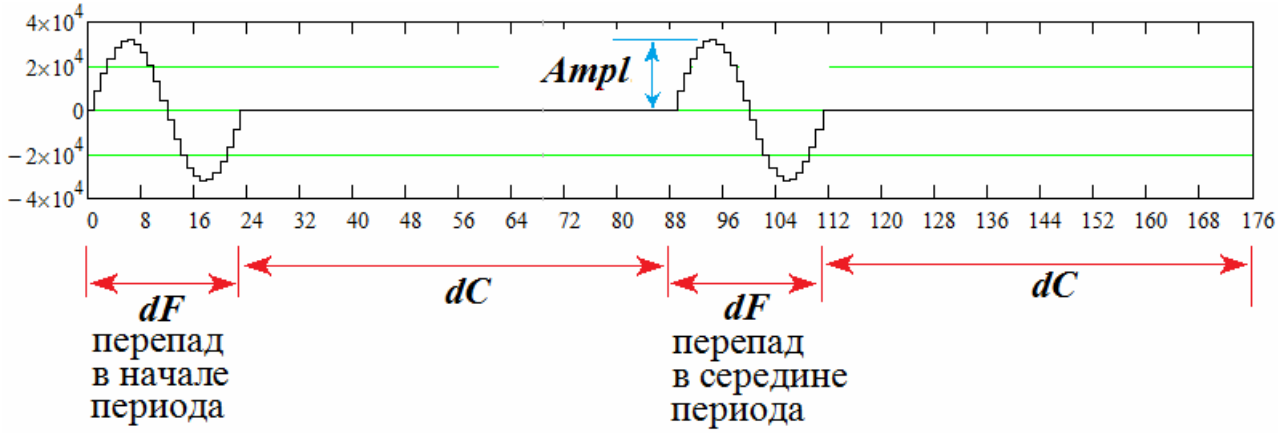


Рис. 8. Растянутое изображение одного периода s-code, содержащего $dM = 176$ отсчётов.

В сигнале типа s-code, как и в m-code, параметры периода dM, dF и dC связаны соотношением $\boldsymbol{dM = 2{\cdot}dF + 2{\cdot}dC}$. В данном примере $dF = 23$, $dC = 65$. Отсчёты в импульсах (в "перепадах") вычислены по формуле $Ampl{\cdot}f(n)$, где $n = 0, 1, 2, \ldots, dF-1$, функция $f(n)$ задана формулой $f(n) = \sin(2{\cdot}\,\pi{\cdot}n\,/\,dF)$.

Из рисунка 7 ясно, что s-code можно преобразовать в ТТЛ-сигнал с m-code просто D-триггером, если для синхровхода триггера сформировать однополярные импульсы из двухполярных синусообразных импульсов s-code. Формирование синхроимпульсов сводится к усилению

сигнала с s-кодом из плеера и затем к устранению участков отрицательной полярности выпрямителем. Структура такого ИУ схематично изображена на рис. 9:

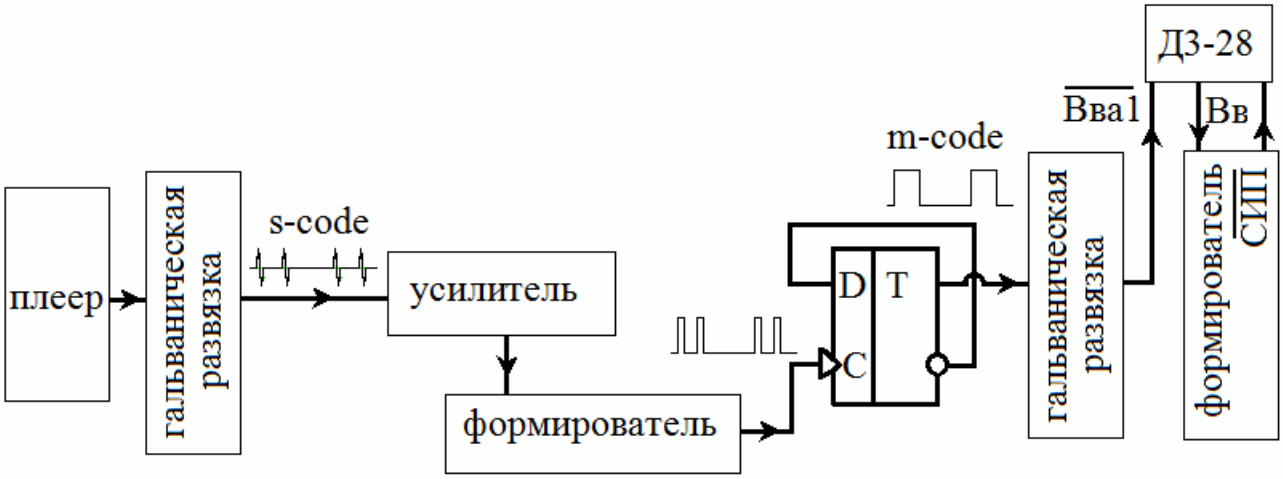


Рис. 9. Структурная схема предлагаемого интерфейсного устройства

Здесь учтено, что плеер (им может быть мобильное устройство или ПК), ИУ и Д3-28 в общем случае подключаются через собственные блоки питания к силовой электросети и поэтому они должны быть по "землям" и сигнальным цепям гальванически развязаны друг от друга.

Вероятно, можно придумать много разных вариантов реализации структуры подобного типа. Например, если ПК (или микроконтроллер с достаточно ёмким устройством памяти) имеет порт вывода ТТЛ-сигналов, который можно безбоязненно соединить с контактами разъёма "Ввод-вывод" Д3-28, то через него удалось бы загружать ПО в Д3-28 непосредственно из файлов с м-кодом – преобразовывать s-code в m-code не требуется в таком варианте. Для самых больших Д3-28-программок размер wav-файлов с указанными выше параметрами "по умолчанию" – несколько десятков мегабайт. Если нет памяти с таким объёмом, сопряжённой с упомянутым контроллером, то тогда роль памяти сможет играть плеер, и для передачи данных через его аудио-выход в плеере должен быть файл с s-кодом, а контроллер преобразует s-код в м-код программно. Возможно, приёмником-усилителем аудио-сигнала из плеера в этом варианте может служить АЦП в составе контроллера. Гальваническую развязку с Д3-28 при необходимости могут обеспечивать схемы с оптронами.

Теперь расскажу о своей реализации ИУ типа рис. 9 (ещё не проверенной на реальной Д3-28; не имею этой ЭВМ). К сожалению, у меня нет знаний о микроконтроллерах. Мои познания и элементная база – из радиолюбительства 1960-80-х; да прибавились старческие проблемы со зрением. Поэтому ничего не стал покупать, а из уже имеющихся б / у старинных радиодеталей спаял, как позволило зрение, макет "на картонке"; точнее, на гетинаксе, навесным монтажом. Подробная схема приведена в Приложении 1; вот её краткое описание:

- Плеер – старый китайский "Explay C300" с его зарядным устройством в роли блока питания.
- Гальв. развязка на входе в ИУ – согласующий трансформатор от УНЧ радиоприёмника.
- Усилитель – один каскад на биполярном кремниевом транзисторе n-p-n (типа КТ315).
- Формирователь синхроимпульсов для триггера – два КТ315 с двумя диодами Д223Б.
- Триггер – один из двух триггеров в составе микросхемы К555 ТМ2.
- Гальв. развязка между выходом триггера и входом Ввa1 в Д3-28 – ВЧ-трансформатор.

Вот подробности о ВЧ-гальванической развязке и о формирователе СИП:

Поскольку оптронов (и опыта работы с ними) у меня нет, то пришла в голову мысль заменить оптическую передачу данных радиочастотной передачей; благо, имею радиолюбительский опыт построения детекторных приёмников и гетеродинов – маломощных ВЧ-генераторов для радиоприёмников. См. рис. 10:

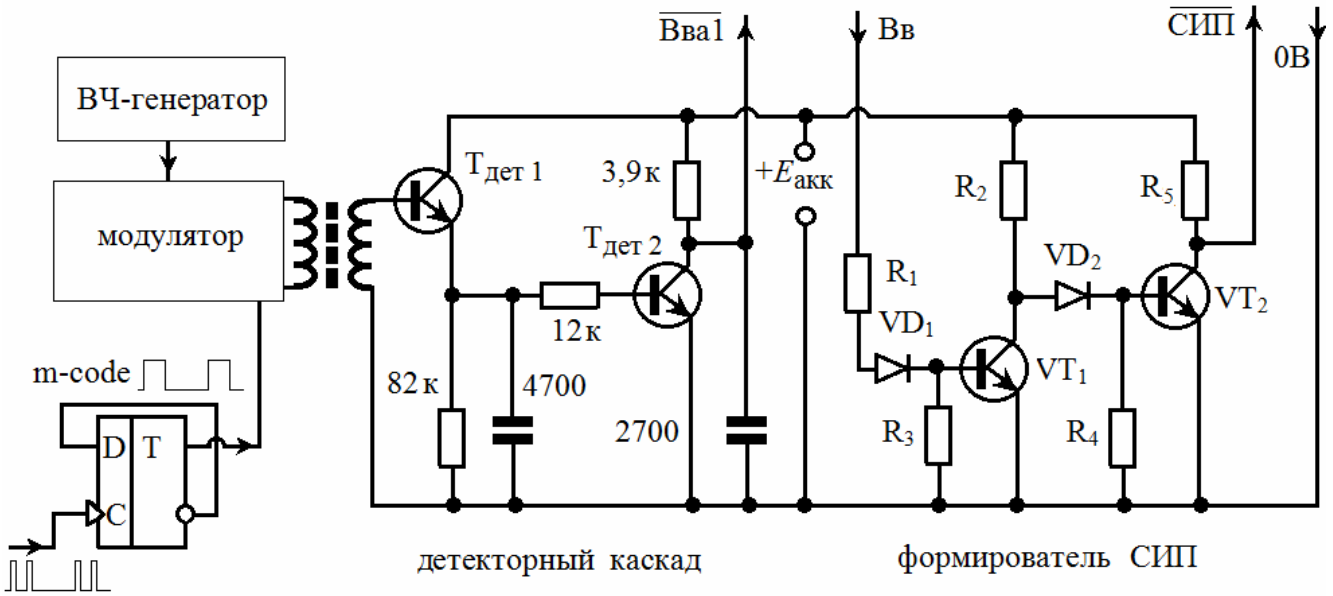


Рис. 10. Подключение выхода ИУ к Д3-28 без гальванической связи с остальной частью ИУ. (Все транзисторы – типа КТ315, диоды – Д223Б или Д220Б, $R_1 = R_2 = 18$ кОм, $R_3 = R_4 = 82$ кОм, $R_5 = 3.9$ кОм. Трансформатор: 2 по 40 витков на ферритовом кольце.)

Высокочастотное (~ 6 МГц) напряжение из маломощного генератора поступает в модулятор. Напряжение с выхода триггера управляет модулятором: если на выходе триггера логический 0, то модулятор "закрыт" – не пропускает на свой выход ВЧ-напряжение, а когда на выходе триггера логическая 1, – модулятор "открыт", пропускает ВЧ-сигнал в первичную обмотку трансформатора на своём выходе. Схема генератора и модулятора приведена в Приложении 1.

Переход "база-эмиттер" 1-го транзистора детекторного каскада ($T_{дет\,1}$) играет роль диода – вместе с RC-нагрузкой он выделяет огибающую ВЧ-сигнала, поступающего со вторичной обмотки трансформатора: когда сигнала нет, напряжение между эмиттером $T_{дет\,1}$ и "общим выводом" (0В) близко к нулю, и транзистор $T_{дет\,2}$ закрыт, а когда сигнал есть, то оно близко к 2 вольтам, и транзистор $T_{дет\,2}$ открыт. Показанные на рис. 10 транзисторы могут питаться от батареи из 4 аккумуляторов: $+E_{акк} \approx 4 \cdot 1.2$ В = 4.8 В $\approx$ 5 В. Когда $T_{дет\,2}$ открыт, напряжение на его коллекторе близко к нулю, а когда он закрыт, напряжение на его коллекторе через резистор 3.9 килоома подтянуто к $+E_{акк}$. Таким образом, напряжение на коллекторе $T_{дет\,2}$ служит инвертированным сигналом с м-кодом ТТЛ-формата – это сигнал $\overline{\text{Ввa1}}$ для Д3-28.

О начале выполнения команды INPO машина сообщает интерфейсному устройству синхросигналом $\text{Вв} \rightarrow 0$. Выполнение INPO завершается тогда, когда ИУ выдаёт в машину ответный сигнал $\overline{\text{СИП}} \rightarrow 0$. По этому сигналу машина восстанавливает высокий ТТЛ-уровень на линии Вв, и тогда ИУ должно "снять $\overline{\text{СИП}} = 0$" – восстановить высокий ТТЛ-уровень на этой линии. В формирователе СИП на рис. 10 такая причинно-следственная связь $\overline{\text{СИП}}$ с Вв реализована двумя инверторами на транзисторах VT_1 и VT_2. В нашем случае нет надобности синхронизировать $\overline{\text{СИП}}$ с $\overline{\text{Ввa1}}$, поэтому у нас формирование $\overline{\text{СИП}}$ никак не связано с остальными частями ИУ; желательно только, чтобы по ходу поиска перепадов сигнала $\overline{\text{Ввa1}}$ машина повторяла выполнение команды INPO как можно чаще.

В экспериментах может выясниться, что схема на рис. 10 допускает упрощения (ну или наоборот: необходимо усложнение…). Вот три варианта (А, Б и В) предполагаемых упрощений:

А) Самым простым формирователем СИП, наверное, может быть просто провод, соединяющий контакты Вв и $\overline{\text{СИП}}$ на разъёме ВВОД-ВЫВОД машины Д3-28. Кроме того, если вход $\overline{\text{Вва1}}$ уже подтянут к высокому ТТЛ-уровню в машине Д3-28, то можно попробовать обойтись без его подтяжки к $+E_{\text{акк}}$ в ИУ. И тогда, чтобы вообще избавиться от аккумуляторов, можно попробовать питать $T_{\text{дет 1}}$ напряжением высокого ТТЛ-уровня с выхода $\overline{\text{СИМ}}$, имеющегося в разъёме ВВОД-ВЫВОД машины Д3-28:

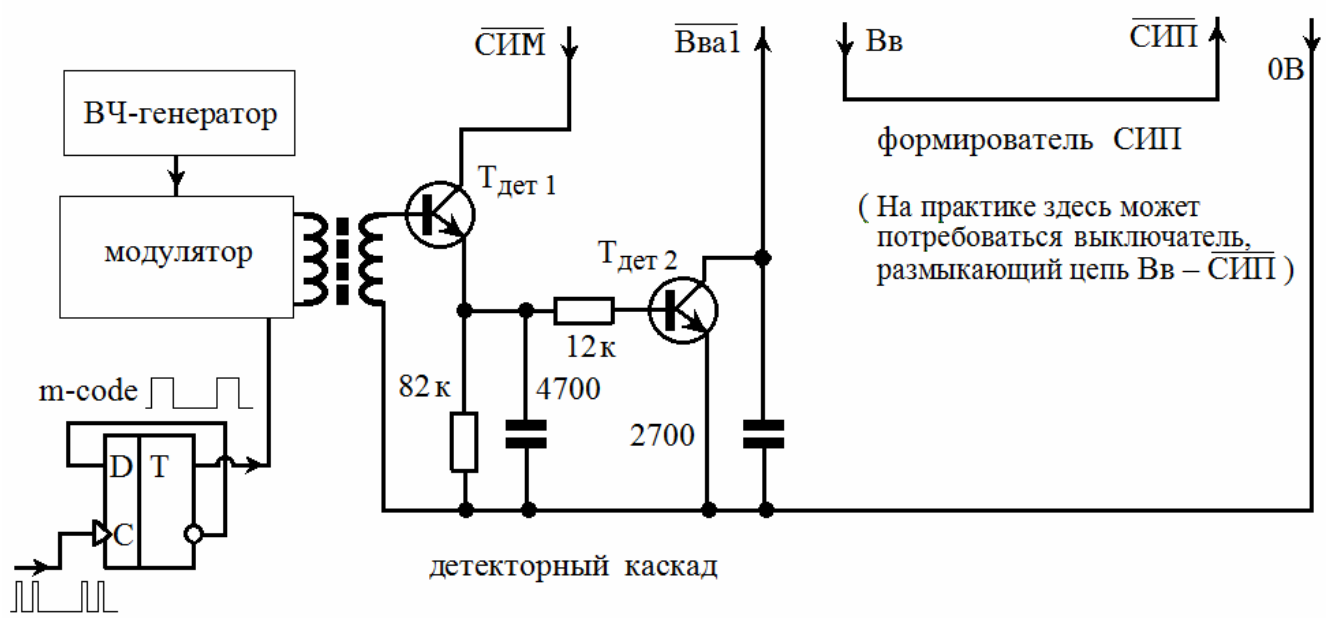


Рис. 11. Вариант А схемы типа показанной на рис. 10.

Б) Если же контакты Вв и $\overline{\text{СИП}}$ на разъёме ВВОД-ВЫВОД машины Д3-28 соединять нельзя (не знаю, что тому могло бы быть причиной… ну, вдруг нужна некая задержка от Вв до $\overline{\text{СИП}}$), то транзисторный формирователь СИП не выкидываем. Чтобы всё-таки избавиться от аккумуляторов, запитываем от $\overline{\text{СИМ}}$ не только $T_{\text{дет 1}}$, но и VT_1, предполагая при этом, что входы $\overline{\text{Вва1}}$ и $\overline{\text{СИП}}$ внутри машины Д3-28 уже подтянуты к высокому ТТЛ-уровню:

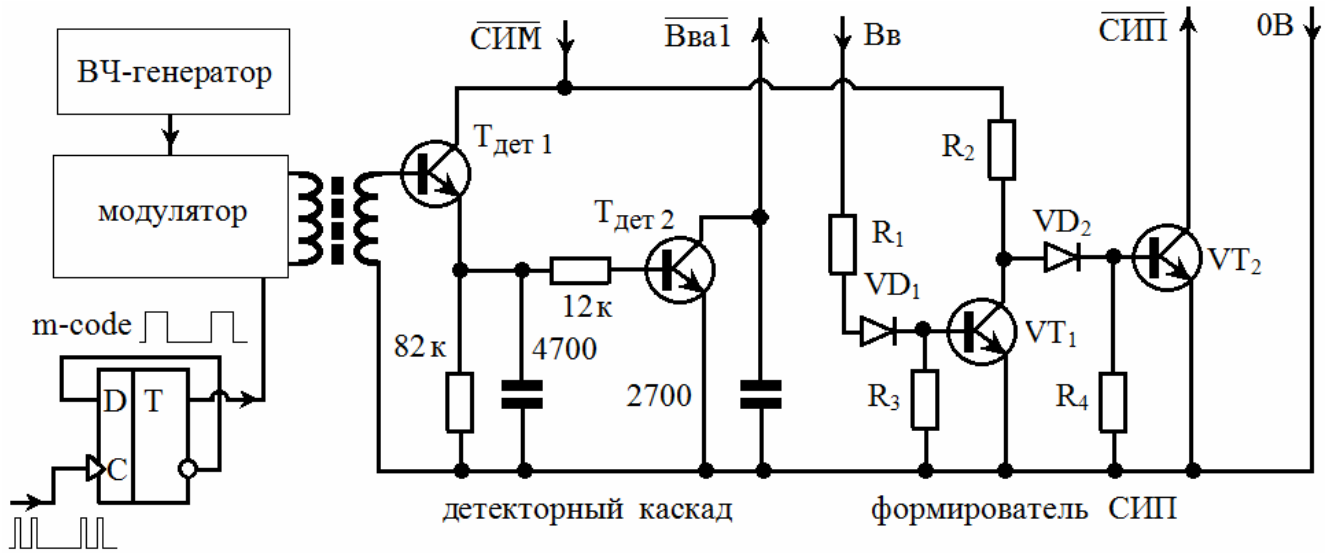


Рис. 12. Вариант Б схемы типа показанной на рис. 10.

В) Если же подтяжка входа $\overline{\text{Вва1}}$ к высокому ТТЛ-уровню необходима, то аккумуляторную батарею не выкидываем; при этом $\overline{\text{СИМ}}$ можем не использовать. Формирователь СИП заменяем проводом, если это допустимо:

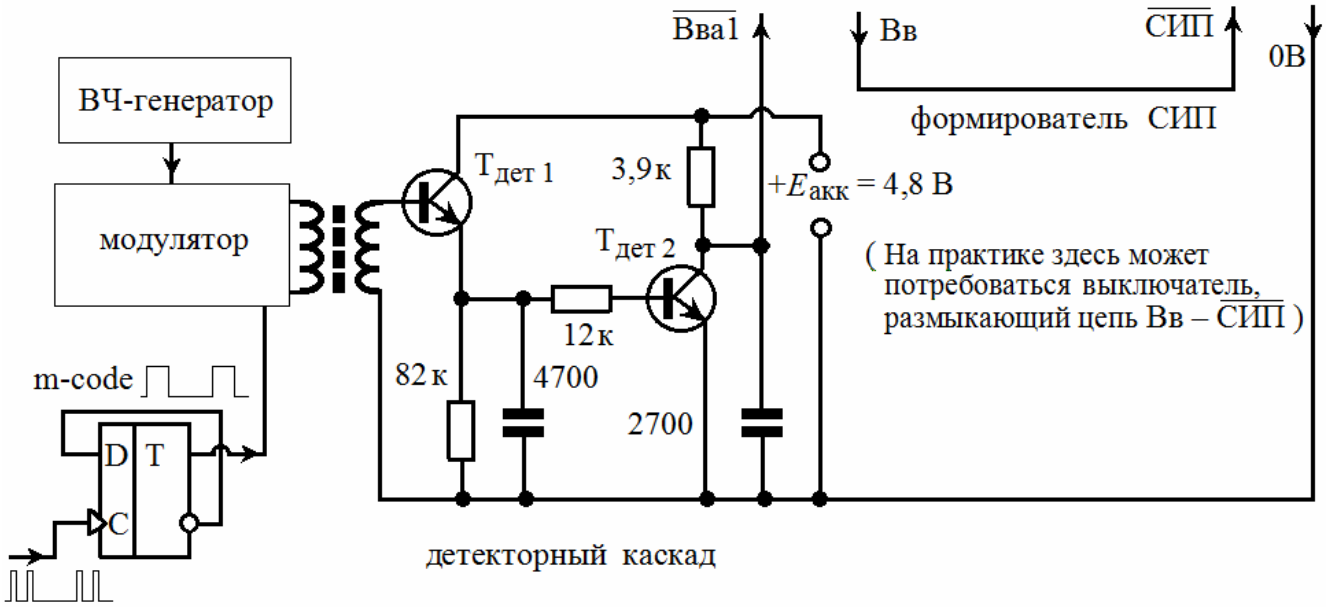


Рис. 13. Вариант В схемы типа показанной на рис. 10.

Г) Наконец, вариантом "Г" назовём исходную схему, показанную на рис. 10.

Таким образом: вариант А – самый простой; варианты А и Б – не требуют выходного источника питания (аккумуляторов), но и не подтягивают входы машины к "+"; в вариантах А и В нет сложного формирователя СИП (его роль играет просто проволочная перемычка); в вариантах В и Г не используется выход СИМ, так как для питания и подтяжки применена аккумуляторная батарея; вариант Г (рис. 10) – среди названных вариантов самый сложный, но и наименее зависимый от предположений. В моём макете ИУ варианты выбираются перемычками.

Описание "софта":

3. Как wav-файлы создаются в имитаторе Д3-28 версии 1.2

Под управлением ОС Windows экспериментировать с производством wav-файлов из txt-программок с не самодельным софтом удаётся, например, в не бесплатной программе Mathcad. Однако работа в ней требует навыков, продумывания и программирования всех деталей до мелочей (включая разложение каждого тетрадно-десятичного числа из txt-файла на биты и их кодирование последовательностями сэмплов). Не у каждого пользователя / владельца Д3-28 найдётся время и желание углубляться в такие детали и притом инсталлировать Mathcad.

Можно было бы сделать утилиту – "экзешник для создания wav-файлов", – упрятать туда программистские нюансы, а пользователю предоставить диалоговый интерфейс с меню для выбора типа и параметров wav-файла. В широком круге экспериментов с загрузкой ПО в Д3-28 различным пользователям могут понадобиться следующие типы wav-файлов:

- 1-канальные с s-кодом (для загрузки из плеера через ИУ, преобразующее s-код в м-код);
- 1-канальные с м-кодом (для загрузки из ПК через некий порт с постоянными уровнями);
- структура этих wav-ов может выбираться "по умолчанию" либо составляться пользователем;
- 2-канальные в обычном формате Д3-28 (для записи на внешнем кассетнике по схеме Д3-28);
- 2-канальные с необычной формой импульсов, выбираемой пользователем ("синус", "пила").

С учётом множества параметров меню получается многовариантным, а хочется ещё и дать возможность пользователю-экспериментатору программировать форму импульсов (перепадов) по своему желанию. И поскольку в конкретном опыте все варианты не нужны одновременно, то, значит, лучше бы для выбранного варианта компилировать свой экзешник, с простым меню.

В итоге пришла в голову мысль: задействовать в роли синтезатора wav-файлов уже имеющуюся прогу (раздаваемую мной свободно, не требующую инсталляции, и как раз предназначенную любителям Д3-28) – "имитатор Д3-28" с некоторой доработкой. На данный момент это версия 1.2. Имитатор Д3-28 предоставляет вполне гибкий интерфейс: в нём легко пользоваться простыми для изучения языками программирования Д3-28 – Бейсиком или Фортраном (Fortran-5M может компилировать рабочие программки для Д3-28), или составлять управляющие wav-синтезом программки прямо в машинных кодах Д3-28.

В детали синтеза wav-файлов пользователю вникать нет необходимости; я спрятал их в виртуальное периферийное "Устройство", как бы подключенное к шине "Ввод-вывод" имитируемой машины Д3-28. Это Устройство имеет свою память ("Буфер") и исполняет команды – пользователь им управляет из своей программки в имитируемом ОЗУ Д3-28. В реальной жизни для управления реальными ПУ пригодно всё множество разных операций ввода-вывода в системе команд Д3-28. Однако в имитаторе, чтобы его не усложнять, я сделал доступными для связи с Устройством только две двухбайтовые команды – INPO УПР (её код: 1400 B2A2) и OUTO УПР (её код: 1401 B2A2) с пятью специальными значениями второго байта УПР = B2A2, адресующими пять однобайтовых регистров в Устройстве. "Архитектура" такой системы в процессе создания ею файла 028-test.wav из файла 028-test.txt (это выдуманные для рисунка имена файлов) схематично показана на рис. 14:

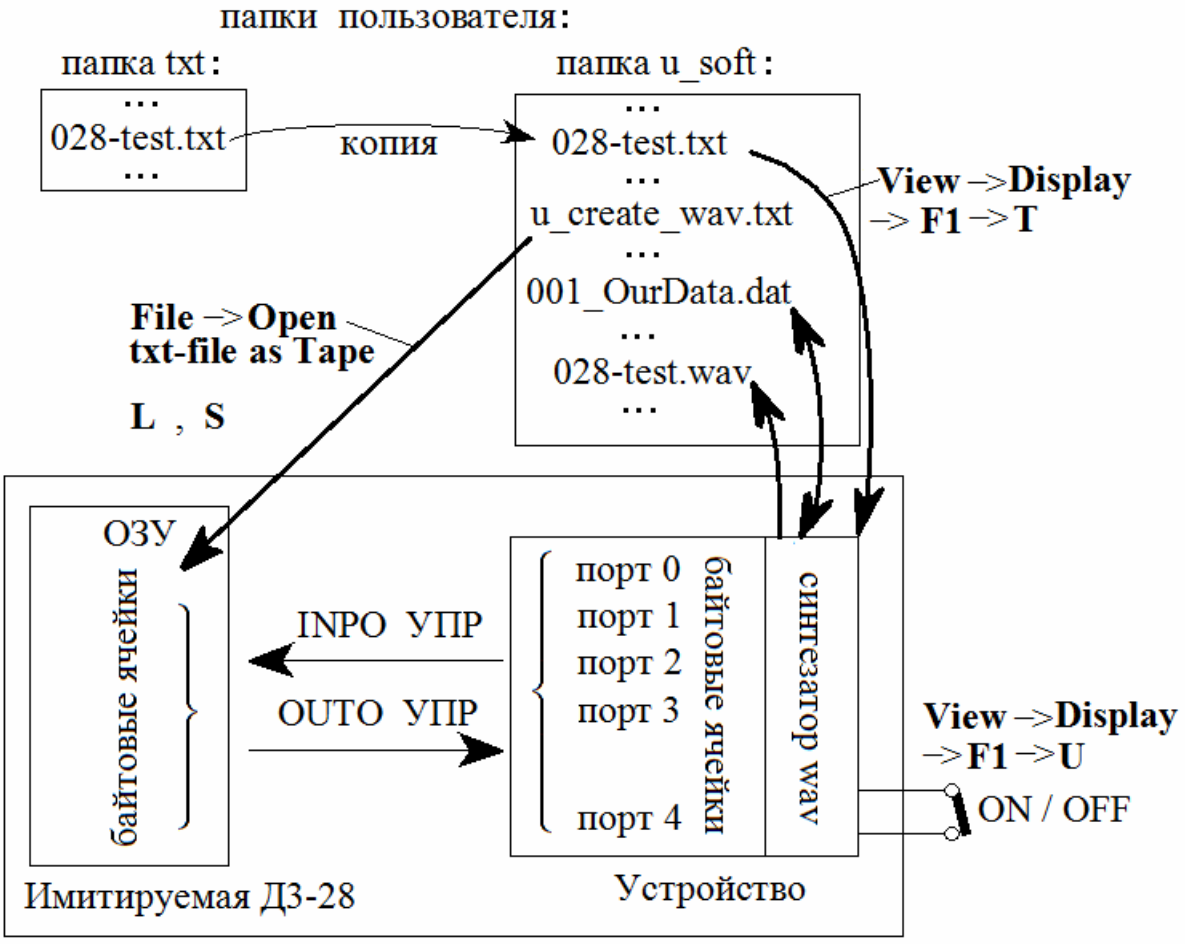


Рис. 14. Создание wav-файла специальным Устройством в составе имитатора Д3-28 версии 1.2. Для вкл / выкл Устройства надо в меню выбрать **View** > **Display** и нажать на клавиатуре **F1**, **U**.

Четыре однобайтовых регистра в Устройстве с условными именами *порт* 0, *порт* 1, *порт* 2, *порт* 3 применяются для обмена данными, а пятый регистр – с именем *порт* 4 – имеет особое назначение: в него командой OUTO УПР = 1014 (её код: 1401 1014) пользовательская программка посылает номер макрокоманды для Устройства, и Устройство эту макрокоманду исполняет. Все макрокоманды Устройства перечислены ниже в таблицах 1 и 2.

В общих чертах (к нюансам вернёмся потом) работа Устройства выглядит так. Устройство, после того как мы его активировали через пункт меню имитатора **View** > **Display** клавишами **F1**, **U**, создаёт "Буфер" – временный файл с именем 001_OurData.dat, записывает туда данные из того txt-файла, который мы уже открыли через**View** > **Display**, **F1**, $\mathbf{T}_{\text{лат}}$ для преобразования в wav, и затем оно выполняет с этими данными действия, предусмотренные программкой пользователя. Эта программка, например, с именем типа u_create_wav.txt открывается в имитаторе обычным образом, через меню **File** > **Open txt-file as Tape**, загружается в имитируемое ОЗУ Д3-28, как обычно клавишей **L**, и запускается на исполнение клавишей **S**. Тип сигнала в wav-файле, параметры, форма перепадов, – всё задаётся в этой программке. Созданный wav-файл Устройство поместит в ту же папку, из которой мы уже открывали файлы, и само удалит там впоследствии ненужный "Буфер" 001_OurData.dat.

Нюанс: если пользоваться разными папками ПК, то может случиться так, что Устройство в конце своей работы не сможет само удалить "Буфер" – не найдёт его; и тогда нам придётся удалять его вручную файловым менеджером ПК. Поэтому при работе с имитатором рекомендуется файлы открывать из какой-то одной папки, например, из папки u_soft, скопировав туда желаемый исходный txt-файл перед началом работы с Устройством.

Ниже везде, где речь идёт о Буфере, подразумевается именно указанный Буфер Устройства (а не буфер обмена Windows в ПК).

Табл. 1. Доступные значения УПР в команде INPO УПР имитатора Д3-28 для ввода байта из Устройства в ОЗУ Д3-28 по адресу (R_{10}) + BD

Байт-коды команды:	Действие:
1400 1010	ввод байта из порта 0 Устройства
1400 1011	ввод байта из порта 1 Устройства
1400 1012	ввод байта из порта 2 Устройства
1400 1013	ввод байта из порта 3 Устройства
1400 1014	Это особая команда: она имитирует аналогичную команду в загрузчике м-кода и предназначается для тестирования загрузок методом м-кода: По адресу (R_{10}) + BD в имитируемое ОЗУ Д3-28 вводится байт 00000001_2, если в Буфере Устройства число с текущим номером *N* больше нуля; а иначе (так бывает, когда в Буфере – wav-данные) вводится байт 00000000. Вычисляется новый текущий номер по формуле *N* + *Шаг1* + *Дельта*, где *Дельта* инициализировалась нулём. По умолчанию *Шаг1* = 2. Это шаг по числам в Буфере для поиска перепада в wav-данных с м-кодом. Если в Буфере число с номером *N* + *Шаг1* имеет прежний знак, то полагается *Дельта* = 0. А иначе *Дельта* = *Шаг2*, где *Шаг2* = (3/4)·*dM*; это послужит задержкой на 3/4 периода после обнаружения перепада.

Табл. 2. Доступные значения УПР в команде OUTO УПР имитатора Д3-28 для вывода байта в Устройство из ОЗУ Д3-28 с адреса (R_{10}) + BD

Байт-коды команды:	Действие:
1401 1010	вывод байта в порт 0 Устройства
1401 1011	вывод байта в порт 1 Устройства
1401 1012	вывод байта в порт 2 Устройства
1401 1013	вывод байта в порт 3 Устройства
1401 1014	это особая команда: вывод байта в порт 4 Устройства и исполнение Устройством макрокоманды с номером, равным значению этого байта. **Список макрокоманд для Устройства при OUTO 1014**: 1. Если (порт 4) = 1, то параметру *dF* присваивается новое значение (если оно не равно нулю, а иначе текущие значения параметров не изменяются) : *dF* = 256·(порт 0) + (порт 1); то есть (порт 0) и (порт 1) интерпретируются как одно 2-байтовое число > 0. С новым *dF* корректируется параметр периода: *dM* = 2·*dF* + 2·*dC*. Устройство показывает значения параметров в окошке сообщений и спрашивает: конвертировать ли данные с s-кодом в Буфере в м-код? Если ответ "нет", то выполнение команды завершается, а если "да", то данные в Буфере, интерпретируемые как s-код, преобразуются в м-код (в соответствии с рис. 7 в разделе 2). Смысл этой макрокоманды: это подготовка к тестированию загрузки загрузчиком м-кода из ранее синтезированного wav-файла с s-кодом. Для создания Буфера с wav-данными выбираем (включив Устройство) **View** > **Display**, **F1**, **W** и открываем нужный wav-файл. Значения его параметров *dF* и *dM* должны быть известны, вот их-то и надо ввести в Устройство в ходе подготовки к тестированию. 2. Если (порт 4) = 2, то параметру *dM* присваивается новое значение (если оно не меньше, чем 2·*dF*, а иначе текущие значения параметров не изменяются) *dM* = 256·(порт 0) + (порт 1); затем: корректируется по *dM* параметр постоянства *dC* = Int(½·(*dM* – 2·*dF*)) (здесь Int означает целую часть), и под значения *dC* и *dF* окончательно подстраивается параметр периода: *dM* = 2·*dF* + 2·*dC*. Выполнение этой команды Устройство завершает окошком с параметрами и кнопкой ОК. 3. Если (порт 4) = 3, то параметру *Шаг1* присваивается новое значение (если оно не равно нулю, а иначе текущее значение *Шаг1* не изменяется) : *Шаг1* = 256·(порт 0) + (порт 1). Этот параметр актуален в команде INPO 1014. Затем в порты вносятся текущие значения *dF* и *dC* согласно формулам: 256·(порт 0) + (порт 1) = *dF*, 256·(порт 2) + (порт 3) = *dC* .

Перечисленные выше действия предназначены, в основном, для настройки тестирования загрузчиков м-кода и получающихся wav-файлов.

Следующие три макрокоманды (строго в указанной последовательности!) подготавливают создание wav-файла, так что в итоге wav-файл появляется:

4. Если (порт 4) = 4, то частоте дискретизации *Fd* (то же, что sample rate) присваивается новое значение (если оно больше 9, а иначе остаётся прежнее):

   Fd = 256·(порт 0) + (порт 1). Затем:

   Для добавочной области в конце Буфера задаётся начальный *номер* отсчётов, которые будут описывать перепад ("буфер для перепада").

   Сбрасывается в ноль счётчик ($n = 0$) будущих обращений к команде вывода в "буфер перепада" значений перепада *Ampl:f*(*n*) (см. раздел 1).

   Задаётся тип wav-файла: при (порт 4) = 4 это "s-code из txt-программки". Т. е. подразумевается, что в Буфер загружены из txt-файла байт-коды собственно программки, а Устройство само добавит в wav-файл полагающиеся по умолчанию 128 начальных нулевых байт, синхробайт 1406, и два байта, указывающие номер последнего байта в программке (при нумерации шагов программки с нуля).

5. Если (порт 4) = 5, то в "буфер перепада" на позицию с текущим *номером* вносится число со знаком (интерпретируемое как *Ampl:f*(*n*)). Это внесение может сопровождаться модификацией значения в порте 0:

   Если (порт 0) > 127 , то число ((порт 0) – 128) вносится в тот же порт 0 вместо первоначального, и переменной *k* присваивается значение –1; а иначе (порт 0) не модифицируется, и *k* получает значение +1. Затем:

   Буфер [*номер*] = *k*·(256·(порт 0) + (порт 1)).

   Таким образом, здесь (порт 0) и (порт 1) интерпретируются как одно 2-байтовое число со знаком; на знак указывает старший бит в (порте 0).

   Затем *номер* позиции в буфере перепада и счётчик *n* вхождений в эту команду увеличиваются на единицу. Подразумевается, что эта команда будет выполняться в цикле с $n = 0, 1, 2, \ldots, dF - 1$ для передачи в Устройство отсчётов импульса *Ampl:f*(*n*), вычисляемых в управляющей программке пользователя по формуле, составленной пользователем.

6. Если (порт 4) = 6, то параметру *dC* присваивается новое значение (если оно не равно нулю, а иначе текущее значение *dC* не изменяется) :

   dC = 256·(порт 0) + (порт 1). Затем:

   Количеству отсчётов в перепаде *dF* присваивается последнее значение счётчика *n*, сформировавшееся в макрокоманде "5": $dF = n_{\text{конечное}}$.

   По значениям *dC* и *dF* определяется период: $dM = 2 \cdot dF + 2 \cdot dC$.

   Устройство показывает значения параметров в окошке и запрашивает согласие на создание wav-файла с этими параметрами. При ответе "нет" Устройство остановит управляющую программку; а если "да", то оно создаст wav-файл с подходящим именем и доложит о выполнении в окошке с кнопкой OK.

7. Если (порт 4) = 7, то в порты 0, 1, 2, 3 как в одно 4-байтовое число без знака вносится количество чисел в Буфере (или 0, если Буфер не создан).

   Это количество чисел Устройство определяет и сообщает в окошке при открытии пользователем исходного файла с данными через меню **View** > **Display**, **F1**, **W** (в случае wav-файла, тогда количество чисел есть количество отсчётов); или $\mathbf{T}_{лат}$ (в случае txt-файла с байт-кодами, тогда количество чисел есть количество тетрадно-десятичных кодов); или $\mathbf{B}_{лат}$ (в случае bin-файла, тогда количество чисел есть просто размер файла, т. е. количество байт в нём).

Следующие пять макрокоманд это варианты макрокоманды "4", отличающиеся только заданием типа wav-файла. Любая из них должна применяться только вместо "4", перед макрокомандами "5" и "6".

8. Если (порт 4) = 8, то создастся wav-файл типа: m-code из txt-программки, а не s-code; (остальные действия будут те же, что и в макрокоманде "4").

9. Если (порт 4) = 9, то создастся wav-файл типа: 2-канальный обычный Д3-28, типа показанного на рис. 1. (Остальные действия те же, что в макрокоманде "4").

10. Если (порт 4) = 10, то создастся wav-файл типа: 2-канальный с необычной формой импульсов, $Ampl{\cdot}f(n) = \sin(2{\cdot}\,\pi{\cdot}n\,/\,dF)$ или др. (Остальные действия будут те же что и в макрокоманде "4").

Для 2-канальных wav-файлов роль параметра "период" берёт на себя вместо dM вдвое меньшая величина: $dm = dF + dC$.

11. Если (порт 4) = 11, то создастся wav-файл типа: m-code из чисел в Буфере без каких-либо "добавлений по умолчанию" (т. е. без добавления начальных нулей, синхробайта, и т. п.) Остальное – как и в макрокоманде "4".

12. Если (порт 4) = 12, то создастся wav-файл типа: s-code из чисел в Буфере без каких-либо "добавлений по умолчанию" (т. е. без добавления начальных нулей, синхробайта, и т. п.) Остальное – как и в макрокоманде "4".

    В макрокомандах 11 и 12 считается, что пользователь открыл в Буфер файл (txt или bin), в котором всё нужное уже заранее добавлено.

Следующие две макрокоманды позволяют для контроля сохранять на диске (в ту же папку, из которой открывались файлы для Устройства) данные из Буфера в файл в формате txt или bin. Они годятся и для преобразования txt ⇔ bin. При этом стоит помнить, что большие файлы здесь обрабатываются долго.

13. Если (порт 4) = 13, то числа из Буфера (подразумевается, что все они целые десятичные) распечатываются столбиком в txt-файл с именем 001_Our_buffer.txt. Так можно распечатать и сэмплы из wav-файла, но это может оказаться долгим делом; притом txt-файл может получиться очень большим.

14. Если (порт 4) = 14, то числа из Буфера распечатываются в bin-файл с именем 001_Our_buffer.bin. Здесь подразумевается, что числа в Буфере представляют собой тетрадно-десятичную запись байтов и поэтому они переводятся в десятичный целочисленный формат со значениями от 0 до 255, т.е. с hex-значениями от 00 до FF в bin-файле.

	Наконец, вот последние (в данной версии) две макрокоманды: они предназначены для регулирования второстепенных параметров 2-канальных wav-файлов, вероятно, не влияющих радикально на исход экспериментов. Речь идёт о возможности программно переставить каналы 0 и 1 (лучше это делать аппаратно, если обнаружится такая необходимость), а также об изменении начальных знаков уровней ± *Ampl* и об изменении длительности паузы в начале и в конце потока информационных перепадов: 15. Если (порт 4) = 15, то трёхзначному десятичному параметру *знаки* присваивается десятичное значение из порта 3, т. е. *знаки* = (порт 3). Разряды $A_3A_2A_1$ параметра *знаки* следующим образом влияют на формирование 2-канального wav-файла: если $A_1 = 0$, то начальный знак канала wav_1 есть +1 (так и по умолчанию), а иначе начальный знак канала wav_1 будет –1; если $A_2 = 0$, то начальный знак канала wav_0 есть +1 (так и по умолчанию), а иначе начальный знак канала wav_0 будет –1; если $A_3 = 0$, то канал wav_0 есть канал нулей, канал wav_1 есть канал единиц (так и по умолчанию); а иначе wav_0 есть канал единиц, wav_1 есть канал нулей. Примеры: при (порт 3) = 321 каналы будут переставлены, начальные знаки будут – и –, а при (порт 3) = 21 перестановки каналов не будет, и только установятся знаки – и –. При (порт 3) = 1 (или 2, или 3, … , 9) перестановки каналов не будет, знаки: $+_0, -_1$. При (порт 3) = 0 установится настройка "по умолчанию": без перестановки, знаки +, +. 16. Если (порт 4) = 16, то количеству отсчётов *dP* в паузах перед началом и после потока информационных перепадов для 2-канального wav-файла будет присваиваться новое значение (если (порт 2) > 0, а иначе будет работать формула по умолчанию: $dP = 20 \cdot dm$) по формуле: $dP = (\text{порт } 2) \cdot dm$. В окошке сообщений будет показано старое и новое *dP*.

Пояснения с примерами создания wav-файлов:

Описание шагов работы с "дружественными к пользователю" диалоговыми программками дано в следующем разделе, а здесь выполнение макрокоманд проиллюстрирую программками "в машинных кодах". Предполагается, что читатель знаком с основными сведениями из "*Справочника программиста 3.857.100 Д3*" о пульте с кнопками и индикаторами, о регистрах ОЗУ, и о системе команд ЭВМ Д3-28. Об имитации работы с пультом Д3-28 в имитаторе рассказано в разделе 7 в "Руководстве к имитатору".

После запуска или ресета имитатора включаем Устройство: **View** > **Display** , **F1**, **U**; окошко покажет состояние Устройства – выключено (OFF), – и предложит его изменить на ON (включено). Кстати, так можно проверять состояние Устройства по ходу работы. Выключается оно аналогично, при этом все его настройки сбрасываются к настройкам "по умолчанию", а память освобождается, т. е. удаляется Буфер, – так можно делать ресет Устройства.

Следующий за включением шаг – вводим исходные данные в Устройство: **View** > **Display**, **F1**, затем нажимаем либо **T** $_{\text{лат}}$, либо **B** $_{\text{лат}}$, либо **W**. Стандартное окно для открытия файла позволит нам выбрать файл в соответствии с нажатой клавишей (и в соответствии с задачей):

$T_{\text{лат}}$ – для выбора txt-файла со столбцом тетрадно-десятичных байт-кодов программки; имя txt-файла обязательно должно иметь расширение .txt.

$B_{лат}$ – для выбора "бинарного" файла, т.е. файла с байт-кодами программки в hex-формате; в имени bin-файла обязательно должно быть расширение .bin. (Эта опция для Д3-28 не нужна. Она введена здесь только потому, что Устройство может создавать wav-файлы для любых компов, принимающих м-код, – типа РК86 или ЭВМ Синуса; в интернете программки для таких компов чаще всего есть именно в hex-формате.)

W – для выбора wav-файла с м-кодом или с s-кодом с целью тестирования его загрузки и для теста загрузчиков. Файл должен быть 1-канальным и иметь расширение .wav.

Дальнейшие шаги разбираю на простейшем примере. Допустим, в папке u_soft имелась бы программка (назовём её pi_kp-15_n-1.txt) с всего двумя байт-кодами: 0609 и 0512. (Это команда PI для вывода в регистр X числа "пи" = 3.14159265359 и команда END. Конечно, загружать 2 байта через зад Д3-28 реального смысла нет; этот пример чисто иллюстративный.) После ввода её в Устройство командой **View** > **Display**, **F1**, $\mathbf{T}_{лат}$ окошко по квази-английски сообщает нам, что в свежесозданном Буфере количество данных = 2; так мы видим, что Устройство не ошиблось (и мы тоже; в имени программки число n есть номер шага с командой END при нумерации шагов с нуля, так что количество байт-кодов равно n + 1). Вообще, хотя "дуракоустойчивость" Устройства слабая, оно способно выдавать некоторые проверочные сообщения; далее для краткости я о них подробно не пишу.

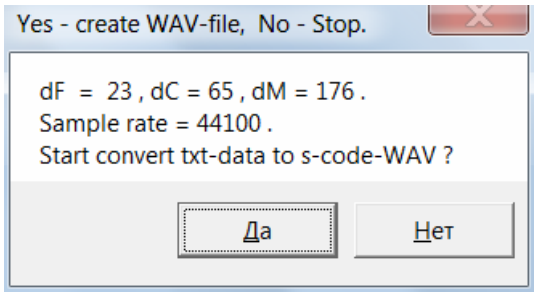
1) *Пример создания wav-файла типа* "s-code из txt-программки":

Командой меню **File** > **Open txt-file as Tape** открываем из папки u_soft программку u_create_s-code_wav_kp-2226_n-182.txt, в режиме **View** > **Machine** загружаем её в имитируемое ОЗУ Д3-28 на адрес 0 клавишами **C**, **L**, и запускаем клавишей **S**; (это обычные действия при запуске программок в имитаторе, о них рассказано в "Руководстве к имитатору", поэтому далее уже не буду их так подробно перечислять).

На Y-табло появляется цифра 4 – это приглашение к вводу в X параметра для макрокоманды с номером 4, т. е. приглашение к вводу значения частоты дискретизации *Fd*. Работа программки при этом останавливается. Можно набрать цифровыми клавишами желаемое значение; либо – ничего не набирать и тем самым согласиться с имеющимся нулём на X-табло, чтобы сохранилось значение *Fd* = 44100 сек $^{-1}$, заданное по умолчанию; этот вариант для примера и выберем. Затем нажимаем **S** для продолжения работы по программе.

На Y-табло появляется цифра 5 – приглашение к вводу значения длины перепада *dF*. Оно требуется самой программке для счётчика в цикле автоматического ввода в Устройство значений перепада по макрокоманде 5, поэтому значение *dF* надо в любом случае набрать на X-табло, даже если выбираем значение по умолчанию: 23. Затем нажимаем **S**.

На Y-табло появляется цифра 6 – приглашение к вводу в X значения длины *dC* постоянного уровня между перепадами; это параметр в макрокоманде с номером 6. Если согласиться с имеющимся на X-табло нулём, то сохранится прежнее значение *dC*, по умолчанию равное 65. Нажимаем **S** для продолжения работы. Появляется окно со значениями параметров и вопросом: "начать конвертировать txt-данные в s-code-wav?" Отвечаем "Да":



Появляется стандартное окно "Сохранить как", в котором можно задать желаемое имя будущему wav-файлу. В нашем примере Устройство по умолчанию предлагает имя

001_Our_44100_s-code_dM-176_dF-23_N-1.wav ,

в нём перечисляются значения параметров *Fd*, *dM*, *dF*, тип сигнала и номер N шага с кодом END из исходной программки. Нажимаем в этом окне кнопку "Сохранить". Устройство быстро исполняет макрокоманду 6 и сообщает в своём окошке: Done, ok. Нажимаем ОК. Вот и всё – wav-файл должен обнаружиться в папке u_soft (если мы по ходу дела не лазили в другие папки).

Размер получившегося wav-файла: 380206 байт. Он такой потому, что в начале подготовки к его созданию применялась макрокоманда 4 – она предполагает автоматическое добавление 128 нулевых байт, синхробайта 1406, двух байт с номером N последнего шага, затем пишутся N + 1 байт собственно txt-программки, и на всякий случай ещё 2 нулевых байта в конце. Итого в нашем примере в s-код конвертировалось 128+1+2+2+2 = 135 байт. В каждом байте 8 бит, каждый бит кодируется *dM* = 176 отсчётами, и каждый отсчёт в wav-файле записывается двумя байтами. Таким образом, имеем в wav-файле 135·8·176 = 190080 отсчётов и вдвое больше байт, т. е.: 380160 байт с отсчётами. И, кроме того, в начало wav-файла записывается стандартная заголовочная часть размером 46 байт; вот так и получается размер wav-файла 380206 байт. Форму получившегося сигнала рассмотрим ниже, после разбора других примеров.

Рассмотренный пример – основной в задаче создания wav-файлов для ввода txt-программок в Д3-28 через интерфейсное устройство, преобразующее сигнал с s-кодом в сигнал с м-кодом.

2) *Пример создания wav-файла типа* "s-code из чисел в Буфере":

Допустим, нас не устраивает указанное выше автоматическое построение wav-файла. Например, мы хотим иметь 2 начальных нулевых байта, синхробайт пусть остаётся прежний (если не хотим изменять загрузчик), а в конце вместо двух нулевых байт пусть идут 3 байта, тетрадами которых изображаются цифры десятичного значения контрольной суммы txt-программки; в нашем примере у программки (0609 0512) контрольная сумма равна 15.

Тогда, прежде всего, текстовым редактором (например, обычным виндовским Блокнотом) надо добавить в исходный txt-файл желаемые байты, не забыв указать и новую информацию о количестве байт, подлежащих загрузке в Д3-28. В нашем примере загрузке подлежат 2 + 3 = 5 байт, поэтому указать надо число 5 – 1 = 4. Т. е. в нашем примере вместо собственно программки (0609 0512) в txt-файле надо приготовить столбик следующих байт-кодов:

0000 0000 1406 0000 0004 0609 0512 0000 0000 0105

Эти новые txt-данные (назовём их pi_buff_10.txt) следует ввести в Устройство после его ресета, либо просто вместо старых данных; во втором случае Устройство по команде **View** > **Display**, **F1**, $\mathbf{T}_{\text{лат}}$ предложит сначала закрыть старый txt-файл с данными (кстати, так можно смотреть, что было введено в Буфер до того), затем надо будет снова нажать $\mathbf{T}_{\text{лат}}$.

Вместо u_create_s-code_wav_kp-2226_n-182.txt загружаем в имитируемое ОЗУ Д3-28 программку u_buff_to_s-code_wav_kp-2200_n-182.txt и запускаем её. В отличие от *Примера 1* она приглашает вводить желаемую частоту дискретизации как параметр макрокоманды с номером 12 (а не 4). Все остальные наши действия – те же, что и в *Примере 1*; причём, нужно внимательно выбирать имя сохраняемого wav-файла, чтобы не перезаписать какой-нибудь нужный старый, и чтобы как-то отобразить специфику нового. В нашем примере я добавил в конце имени метку buff-10 вместо предлагаемой по умолчанию метки N-9:

001_Our_44100_s-code_dM-176_dF-23_buff-10.wav

Если делаем много разных wav-файлов, то бывает удобным менять и номер файла: 002_ и т. д.

3) *Пример создания wav-файла типа* "m-code из чисел в Буфере":

Заодно, пока в Буфере Устройства находятся байт-коды из указанного выше иллюстративного txt-файла pi_buff_10.txt, создадим wav-файл с м-кодом; (wav-файлы с м-кодом в принципе могут пригодиться для экспериментов с загрузкой ПО в Д3-28 какой-то аппаратурой с портом, поддерживающим передачу данных ТТЛ-уровнями в шину "Ввод-вывод" напрямую из сигнала с м-кодом, без s-кода). Если в Буфере находится что-то другое, то для данного примера открываем туда pi_buff_10.txt, как пояснялось выше.

Загружаем в имитируемое ОЗУ программку u_buff_to_m-code_wav_kp-2220_n-182.txt и запускаем её. Она приглашает набирать желаемую частоту дискретизации как параметр макрокоманды с номером 11. Все остальные наши действия – те же, что и в предыдущих двух примерах. Результат в этом примере я сохранил себе с именем:

001_Our_44100_m-code_dM-176_dF-23_buff-10.wav

Размер wav-файла в этом и предыдущем примерах: $10 \cdot 8 \cdot 176 \cdot 2 + 46 = 28206$ байт.

4) *Пример создания wav-файла типа* "m-code из txt-программки":

Вводим в Устройство txt-данные из иллюстративного файла pi_kp-15_n-1.txt, загружаем в имитируемое ОЗУ Д3-28 программку u_create_m-code_wav_kp-2256_n-182.txt, запускаем её. Она приглашает вводить желаемую частоту дискретизации как параметр макрокоманды с номером 8. Все остальные наши действия – те же, что и в предыдущих примерах. Результат в этом примере я сохранил себе с именем, которое Устройство предложило по умолчанию:

001_Our_44100_m-code_dM-176_dF-23_N-1.wav

Размер этого wav-файла такой же, как в *Примере 1*; с тем же объяснением.

Для контроля взглянем теперь, какова
форма сигналов в получившихся одноканальных wav-файлах:

Имитатор Д3-28 не умеет изображать графики, поэтому придётся воспользоваться каким-то сторонним софтом (например, аудио-редактором, или Маткадом), способным изображать "волновую форму" (waveform) сигнала в wav-файле.

Если есть сторонняя программа, способная построить график по столбцу чисел из txt-файла, то можно макрокомандой 13 Устройства записать данные из wav-файла как столбец десятичных чисел в txt-файл (нюанс: это медленная процедура). Для этого через команду **View** > **Display**, **F1**, **W** открываем в Буфер интересующий нас одноканальный wav-файл; (нюанс: после окончания действий с ним надо через ту же команду его закрыть, а иначе Устройство не будет реагировать на попытки открыть txt-файлы; ну или надо сделать ресет Устройства). В имитируемое ОЗУ загружаем программку для подачи макрокоманд u_outo_kp-434_n-31.txt, и запускаем её клавишей **S**. Внешне её запуск незаметен: на X- и Y-табло по-прежнему видны нули; это ожидание набора номера макрокоманды. Набираем 13, нажимаем клавишу **S** – номер 13 переходит на Y-табло, ещё раз нажимаем **S**, и тогда через некоторое время окошко с кнопкой ОК сообщит, что файл 001_Our_buffer.txt записан.

Ниже приведены графики сигналов из наших иллюстративных wav-файлов, построенные программой Маткад. Качество этих графиков, конечно, не полиграфическое, корявенькие они, но характер сигналов по ним понять можно. Иллюстративная txt-программка pi_kp-15_n-1.txt такая короткая, что вся поместилась на рис. 15 и рис. 16 вместе с автоматически добавленными байтами (кроме 128 начальных нулевых байт – они не поместились на рисунках):

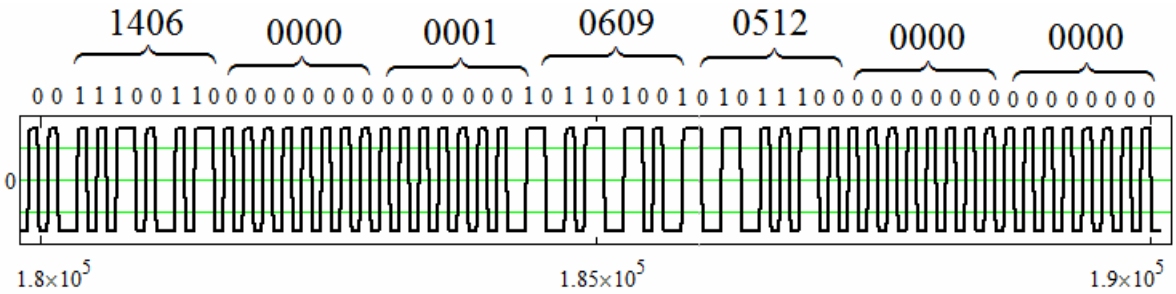


Рис. 15. Сигнал с м-кодом в файле 001_Our_44100_m-code_dM-176_dF-23_N-1.wav, полученный в *Примере 4* как "m-code из txt-программки" для pi_kp-15_n-1.txt.

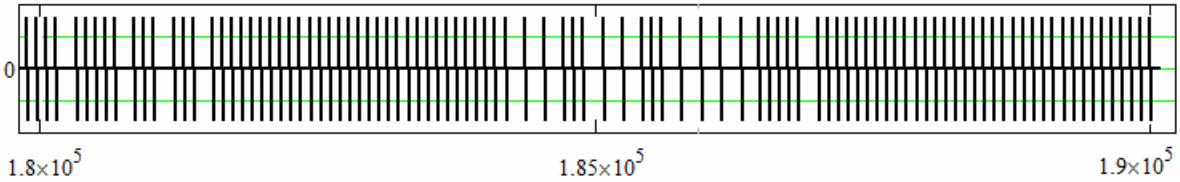


Рис. 16. Сигнал с s-кодом в файле 001_Our_44100_s-code_dM-176_dF-23_N-1.wav, полученный в *Примере 1* как "s-code из txt-программки" для pi_kp-15_n-1.txt.

Аналогично рис. 15 и рис. 16 выглядят содержательные части сигналов в *Примере 3* с "m-code из чисел в Буфере" и в *Примере 2* с "s-code из чисел в Буфере"; только в них вместо байта 0001 закодирован байт 0004, в конце после двух нулевых байт есть ещё байт 0105, а в начале вместо 128 присутствуют лишь 2 нулевых байта.

Форма "перепадов" в сигналах, показанных на рис. 15 и рис. 16, лучше видна на растянутых по горизонтали графиках. Вот так выглядят перепады в m- и s-кодах байта 0609:

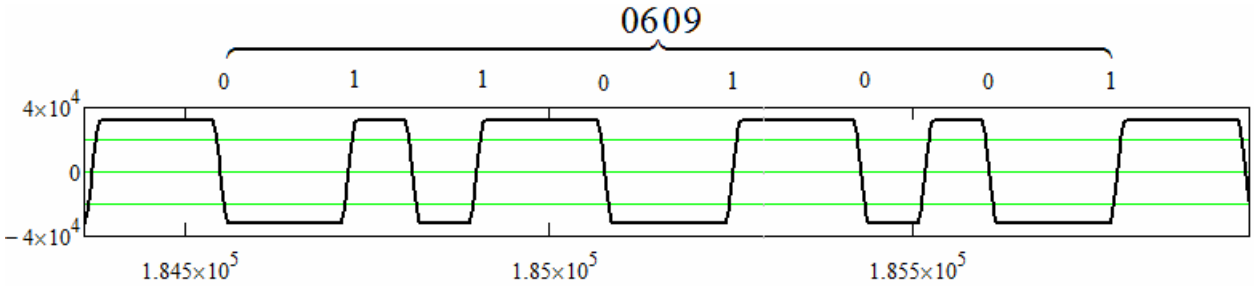


Рис. 17. Растянутое изображение м-кода из сигнала, показанного на рис. 15. Перепады задавались функцией $f(n) = \cos(\pi \cdot n / dF)$, где $n = 0, 1, 2, \ldots, dF-1$.

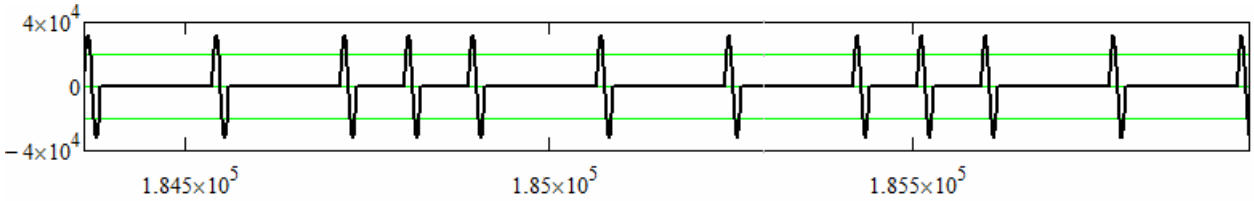


Рис. 18. Растянутое изображение s-кода из сигнала, показанного на рис. 16. Перепады задавались функцией $f(n) = \sin(2 \cdot \pi \cdot n / dF)$, где $n = 0, 1, 2, \ldots, dF-1$. (Ещё более растянутое изображение одного периода было приведено на рис. 8.)

Как пояснялось в разделе 1, значения перепада определяются для середины периода формулой типа $Ampl \cdot f(n)$, где $n = 0, 1, 2, \ldots, dF-1$, причём функция $f(n)$ задаётся пользователем в программке, управляющей Устройством при создании wav-файла. В папке u_soft есть готовые примеры программок с разными формами перепада; взглянем на эти примеры:

5) *Пример wav-файла типа* "m-code из txt-программки" *с перепадами экспоненциальной формы*

Если вместо программки u_create_m-code_wav_kp-2256_n-182.txt, которая создаёт м-код с показанными на рис. 17 перепадами, точно так же воспользоваться программкой (тоже создающей wav-файлы с м-кодом) u_create_m-code-exp_wav_kp-2626_n-206.txt, то перепады будут иметь вид экспоненциально заваленных фронтов и спадов:

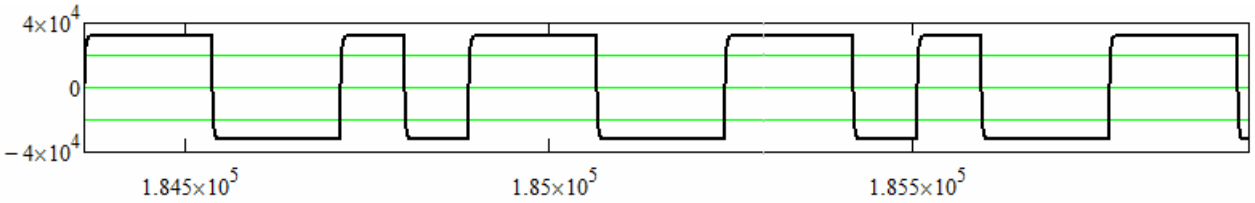


Рис. 19. Сигнал с м-кодом при выборе $f(n) = 2\cdot\exp(-0.75\cdot n) - 1$.
Коэффициент 0.75 можно изменять в программке вручную,
он влияет на крутизну перепадов.

6) *Пример wav-файла типа* "m-code из txt-программки" *с перепадами ступенчатой формы*

Если аналогично применить программку u_create_m-code-step_wav_kp-2594_n-206.txt, то получим wav-файл со ступенчатыми, т. е. предельно крутыми перепадами:

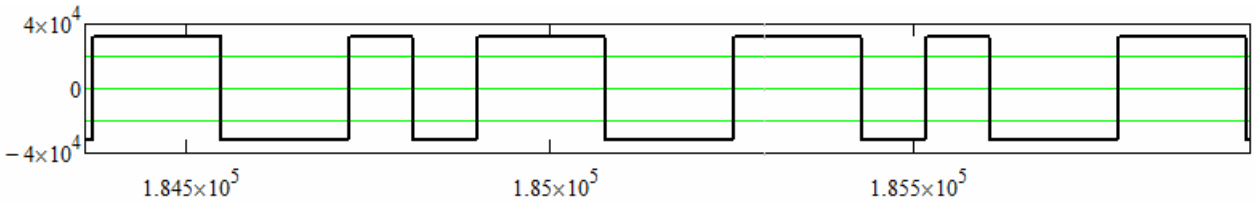


Рис. 20. Сигнал с м-кодом при выборе $f(n) = 1$ для $n < dF / 2$
и $f(n) = -1$ при остальных значениях $n < dF$.

Информацию об уровнях $\pm Ampl$ для м-кода Устройство получает из начального значения перепада $Ampl{\cdot}f(0)$, поэтому при выборе формулы для $f(n)$ в случае с м-кодом важно позаботиться о выполнении условия $f(0) = 1$.

7) *Пример wav-файла типа* "s-code из txt-программки" *с резкими перепадами ступенчатой формы*

Если аналогично применить программку u_create_s-code-step_wav_kp-2586_n-206.txt, то получим wav-файл с s-кодом со ступенчатыми перепадами (вместо синусообразных):

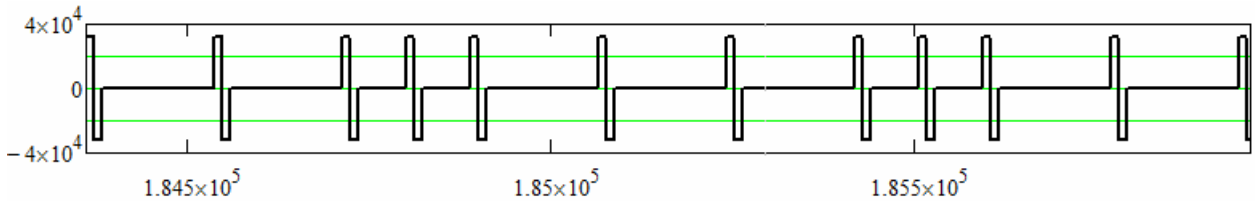


Рис. 21. Сигнал с s-кодом при выборе $f(n) = 1$ для $n < dF / 2$
и $f(n) = -1$ при остальных значениях $n < dF$.

В каждой управляющей программке можно вручную изменить знак параметра *Ampl* – это даёт инверсию полярности сигнала в wav-файле. И можно вручную изменить (уменьшить) величину *Ampl* – изменить "громкость" сигнала. (Вероятно, всё это не скажется принципиально на результатах опытов; тем не менее, такая степень свободы в управлении wav-синтезом у нас есть.)

Одноканальные wav-файлы нужны для опытов с загрузкой ПО в ЭВМ Д3-28 через её разъём "Ввод-вывод". Основной, разработанный вариант, – загрузка s-кода с преобразованием в м-код интерфейсным устройством типа показанного на рис. 9 и в Приложении 1.

Виртуальное Устройство в имитаторе может создавать и 2-канальные wav-файлы – для другого варианта опытов – для записи "обычных-Д3-28-сигналов" на МЛ отдельным ЛПМ со стерео-головкой, включённой так же, как в Д3-28 (см. Приложение 2), или даже бытовым м/ф при необычной форме импульсов. Опыты в этом направлении мной до конца не додуманы; неизвестно, могут ли они быть успешными, так что примеры создания 2-канальных wav-файлов привожу просто на всякий случай, "вдруг когда-нибудь пригодятся":

8) *Пример создания wav-файла типа* "2-канальный обычный Д3-28"

На роль исходных данных возьмём опять pi_kp-15_n-1.txt. Начальные действия – такие же как в предыдущих примерах, но в качестве программки, создающей wav-файл, теперь загружаем в имитатор и запускаем u_create_2ch-d3-28_wav_kp-2258_n-182.txt (метка 2ch в имени программки означает 2 channels – два канала).

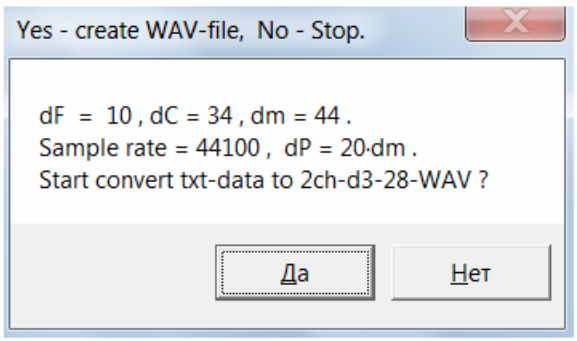
Эта программка приглашает вводить желаемую частоту дискретизации как параметр команды с номером 9. Нажатием клавиши **S** соглашаемся со значением по умолчанию, 44100 Гц.

Для следующих параметров значения по умолчанию вряд ли подойдут, нужна предварительная их оценка. Допустим, мы хотим, чтобы в кассетник с "бытовой" скоростью движения ленты скорость передачи информации из wav-файла составила 1 бит за 0.001 сек. Тогда 1 бит должен передаваться количеством отсчётов dm = (0.001 сек)·(44100 Гц) ≈ 44. Под это количество надо подобрать слагаемые в формуле $dm = dF + dC$. Например, можно попробовать взять dF = 10 и dC = 34.

(В дальнейшем, изменяя dC с шагом 1 в ту или иную сторону, можно в новых wav-файлах пытаться тонко подобрать оптимальную скорость передачи данных. Подбор значения dF тоже может оказаться важным: чем меньше dF, т. е. чем резче перепад, тем больше будет амплитуда импульса в читающей головке Д3-28, но и тем меньше станет его длительность. Знак *Ampl* для "2-канального обычного Д3-28" сигнала изменять нельзя: при $Ampl < 0$ получится ерунда.)

Цифра 5 на Y-табло – приглашение к вводу длины перепада dF. Набираем 10 и нажимаем **S**. На Y-табло появляется цифра 6 – приглашение к вводу dC. Набираем 34 и нажимаем **S**.

Появляется окно со значениями параметров и вопросом: "начать конвертировать txt-данные в обычный двухканальный для Д3-28 wav-файл?" Отвечаем "Да":



В стандартном окне "Сохранить как" можно согласиться с предлагаемым по умолчанию именем файла 001_Our_44100_2ch-d3-28_dm-44_dF-10_N-1.wav (либо изменить его желаемым образом).

В этом примере синтезированный Устройством 2-канальный сигнал выглядит так:

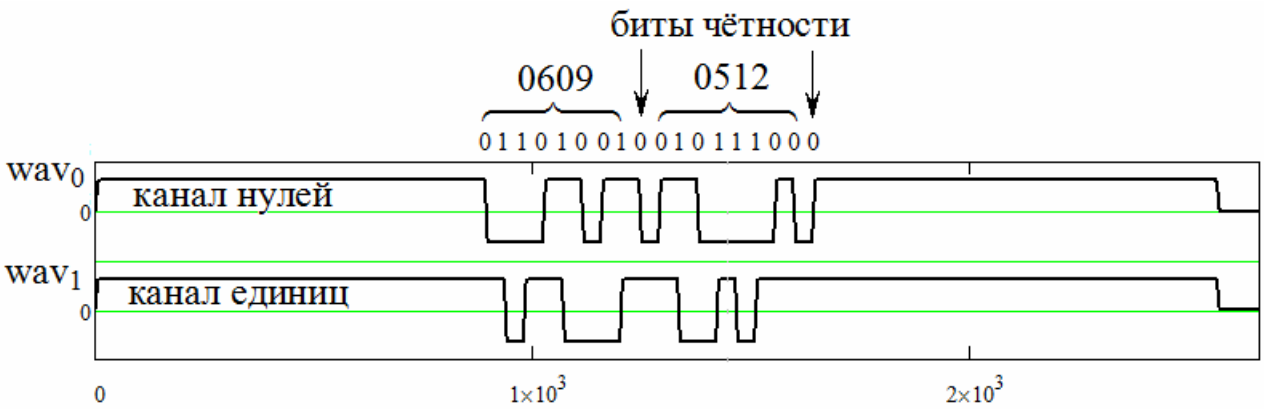


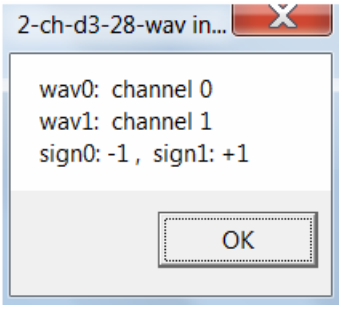
Рис. 22. Двухканальный сигнал в файле 001_Our_44100_2ch-d3-28_dm-44_dF-10_N-1, полученный в *Примере 8* как "2-канальный обычный Д3-28" для pi_kp-15_n-1.txt. В начале и в конце присутствуют паузы длиной $dP = 20 \cdot dm$ (так "по умолчанию"). Каналы не переставлены. Знаки начальных уровней: + в каналах нулей и единиц.

9) *Пример создания wav-файла типа* "2-канальный обычный Д3-28" *с изменённым знаком начального уровня в канале нулей*:

Начало действий в этом примере такое же, как в предыдущем примере. Но перед запуском программки u_create_2ch-d3-28_wav_kp-2258_n-182.txt надо макрокомандой "15" передать в порт 3 Устройства параметр *знаки* = $A_3A_2A_1$ с не равной нулю цифрой A_2; например, *знаки* = 10.

Для этого куда-нибудь в старшие адреса (например, на шаг 20000) загружаем уже упоминавшуюся выше программку подачи макрокоманд u_outo_kp-434_n-31.txt. В своём исходном виде она перед передачей Устройству номера макрокоманды передаёт части числа (укладывающегося в 2 байта) из регистра X в порт 0 и в порт 1. Чтобы она смогла передать параметр *знаки* в порт 3, надо перед её запуском вручную заменить в ней байт УПР = 1011 байтом УПР = 1013; он находится на шаге 00021 от начала, т. е. на шаге 20021, если программка загружена на шаг 20000. (Как изменять байты в программках, рассказано в разделе 7 "Руководства к имитатору".) Затем устанавливаем номер шага на начало этой программки, т. е. в нашем примере на шаг 20000, переводим имитируемую машину из режима "Ввод" в режим "Работа" и запускаем программку клавишей **S**.

Внешне её запуск незаметен: на X- и Y-табло по-прежнему видны нули – это ожидание набора номера макрокоманды. Набираем 15, нажимаем клавишу **S**, номер 15 переходит на Y-табло, набираем желаемое значение параметра, т. е. 10 в нашем примере, и ещё раз нажимаем **S**, макрокоманда исполняется, и окошко сообщает результат настройки:



Нажимаем **C** (сброс) для возврата к шагу 0 – к началу u_create_2ch-d3-28_wav_kp-2258_n-182.txt, и запускаем эту программку нажатием **S**. Дальнейшие действия – такие же, как в предыдущем примере. Результат:

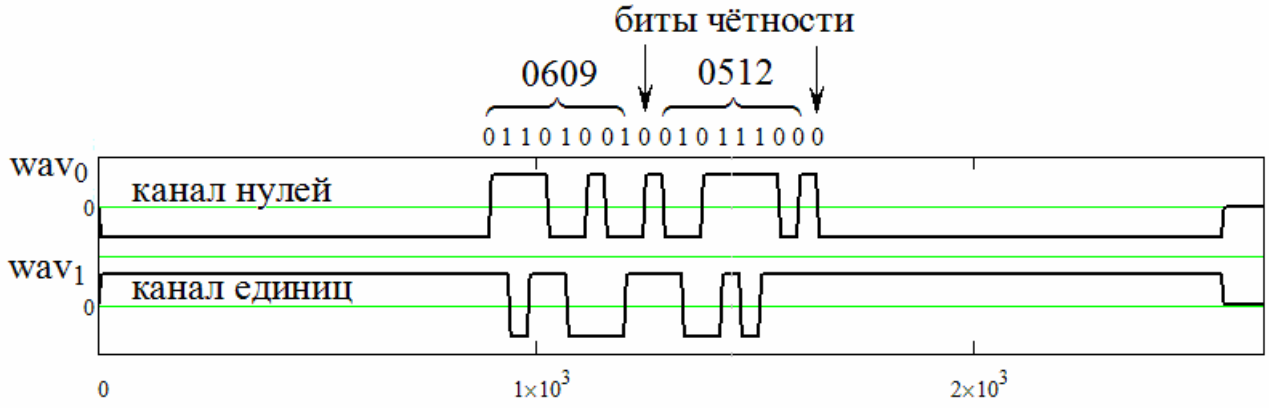


Рис. 23. Двухканальный сигнал, полученный в *Примере 9* как "2-канальный обычный Д3-28" для pi_kp-15_n-1.txt. Отличие от сигнала на рис. 22: здесь изменён знак начального уровня в канале нулей: имеем "–" вместо "+".

10) *Пример создания wav-файла типа* "2-канальный обычный Д3-28" *с переставленными каналами нулей и единиц*:

Для получения 2-канального wav-файла с переставленными каналами надо перед запуском программки u_create_2ch-d3-28_wav_kp-2258_n-182.txt передать макрокомандой "15" в порт 3 Устройства параметр *знаки* = $_3A_2A_1$ с не равной нулю цифрой $_3A$ Например, чтобы получить сигнал с такими же знаками начальных уровней в каналах нулей и единиц, как на рис. 23, но с переставленными каналами, можно задать значение *знаки* = 101. Здесь мы выбрали начальный знак "–" в сигнале wav_1, потому что теперь в сигнале wav_1 расположится канал нулей, и мы хотим получить начальный знак "–" именно в нём. Результат:

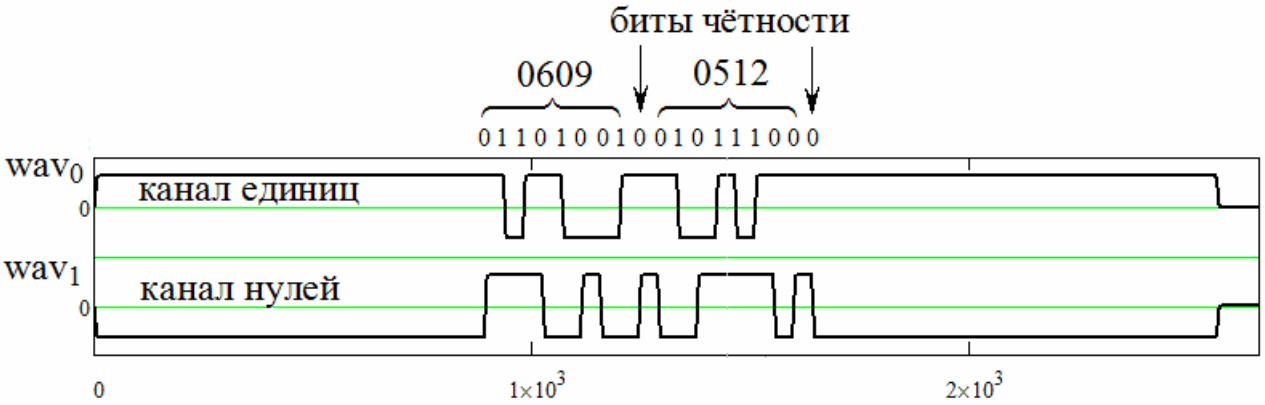


Рис. 24. Двухканальный сигнал, полученный в *Примере 10* как "2-канальный обычный Д3-28" для pi_kp-15_n-1.txt. Отличие от сигнала на рис. 23: здесь каналы нулей и единиц поменялись местами.

Явного примера с изменением длины пред- и после-информационной паузы *dP* не привожу. Всё просто: *dP* изменяется тоже программкой u_outo_kp-434_n-31.txt, но перед её запуском надо вручную заменить байт УПР = 1011 байтом УПР = 1012 (а не 1013, как было в предыдущем примере). После её запуска набираем номер макрокоманды 16, нажимаем **S**, набираем желаемый коэффициент (для передачи в порт 2, см. пункт 16 в приведённой выше таблице 2), нажимаем **S**. Затем указанным выше образом создаём 2-канальные wav-файлы.

Приведу пару примеров создания 2-канальных wav-файлов с "необычной" формой импульсов (см. пункт 10 в табл. 2).

11) *Пример создания wav-файла типа* "2-канальный с sin-образными импульсами".

Программка u_create_2ch-s-pulse_wav_kp-2196_n-182.txt на месте каждого перепада обычного 2-канального сигнала Д3-28 создаёт синусообразный импульс (по аналогии с тем, как в одноканальном сигнале с s-кодом sin-импульсы сопоставляются перепадам м-кода). Наше "Устройство" до ресета помнит предыдущую настройку; и помнит содержимое Буфера до ввода в него новых данных. Например, если перед этим создавался сигнал, показанный на рис. 24, то для получения аналогичного сигнала с sin-импульсами достаточно загрузить в имитируемое ОЗУ Д3-28 и запустить вместо u_create_2ch-d3-28_wav_kp-2258_n-182.txt программку u_create_2ch-s-pulse_wav_kp-2196_n-182.txt. Действуя затем, как и в предыдущем примере, т. е. по аналогии с *Примером 8*, получим результат, показанный на рис. 25:

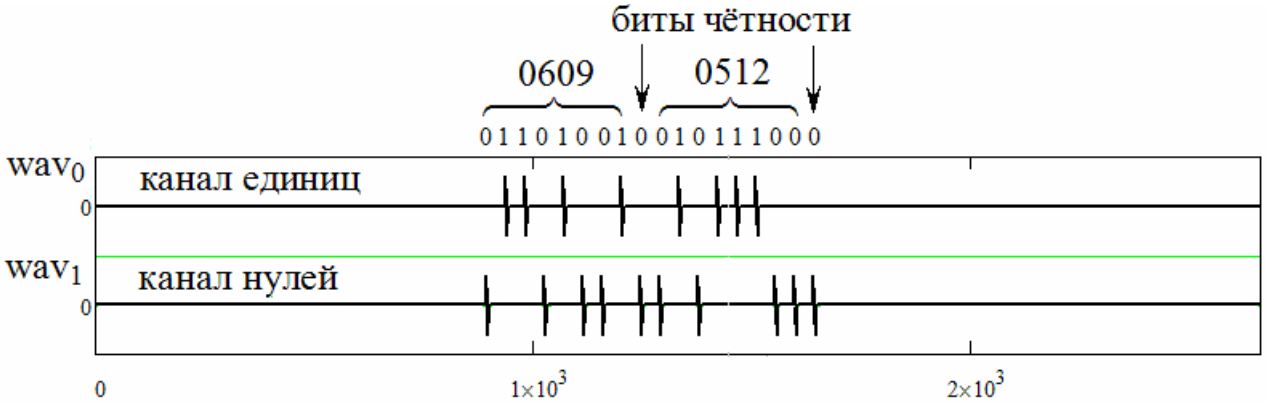


Рис. 25. 2-канальный сигнал в файле 001_Our_44100_swap2ch-s-pulse_dm-44_dF-10_N-1.wav, полученный в *Примере 11* как "2-канальный с sin-импульсами" для pi_kp-15_n-1.txt. В начале и в конце присутствуют паузы длиной $dP = 20 \cdot dm$ (так "по умолчанию"). Каналы переставлены (так как их перестановка была задана в настройке раньше). Чтобы отключить перестановку, надо макрокомандой 15 задать (порт 3) = 0 ⟶ *знаки*.

Отключив перестановку каналов макрокомандой 15 с помощью программки, которая после выполнения *Примера 9* всё ещё находится в ОЗУ на шаге 20000, можно снова запустить находящуюся на шаге 00000 программку u_create_2ch-s-pulse_wav_kp-2196_n-182.txt, чтобы получить версию wav-файла без перестановки каналов. Синусообразная форма импульсов отчётливее видна на растянутом по горизонтали изображении:

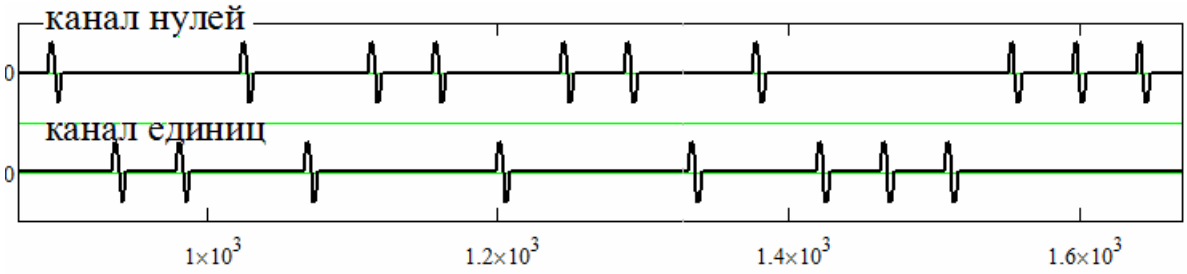


Рис. 26. 2-канальный сигнал в файле 001_Our_44100_2ch-s-pulse_dm-44_dF-10_N-1.wav, полученный как "2-канальный с sin-импульсами" для pi_kp-15_n-1.txt без перестановки каналов. Изображена в растянутом виде только информативная часть сигнала, а начальная и конечная паузы здесь не показаны.

Команда изменения знаков начальных уровней не влияет на такие сигналы; потому что у них постоянный уровень между импульсами нулевой. Перестановка каналов, разумеется, влияет.

12) *Пример создания wav-файлов типа* "2-канальный с пилобразными импульсами".

Продолжим упражняться. Загрузим теперь на нулевой адрес "ОЗУ Д3-28" программку u_create_2ch-saw-pulse_wav_kp-2623_n-226.txt. Она получена на скорую руку переделкой u_create_2ch-s-pulse_wav_kp-2196_n-182.txt, поэтому в работе с ней есть особенности.

Допустим, мы хотим иметь периоды длиной в 44 сэмпла, как в предыдущих примерах двухканальных wav-файлов (см. также рис. 27 и 28 ниже). Такой период можно составить из начальной пологой части "вниз" длиной 19 сэмплов, крутой части "вверх" длиной 5 сэмплов (меньшее значение программка не пропустит из-за защиты от ошибок при вводе, эту защиту можно снять вручную, но пока пусть будет так), после крутой части пусть идёт вторая пологая часть "вниз" длиной 19 сэмплов, и в конце – один сэмпл со значением 0; итого: 19 + 5 + 19 + 1 = 44.

Поскольку регулировать длину такого периода можно двумя параметрами – длинами крутой и пологой частей, то в данной программке для ввода значений этих параметров используется интерфейс ввода двух параметров из предыдущей программки. А именно: длину крутой части пилы вводим в ответ на приглашение к макрокоманде 5 (как если бы это была длина перепада), длины пологих частей вводим в виде их суммы, т. е. в нашем примере 19 + 19 = 38, в ответ на приглашение к макрокоманде 6 (как если бы это была длина постоянного уровня).

При этом программка подаёт Устройству только команду 5, и в трёх циклах друг за другом передаёт значения сэмплов для пилы – начальную пологую часть, крутую часть, и вторую пологую часть. Устройство "думает", что ему передают значения перепада, поэтому в итоге оно насчитывает dF = 19+5+19 = 43 значения; на самом же деле это длина зуба пилы. Затем программка автоматически командой 6 передаёт в Устройство параметр dC = 1; меньше (т. е. ноль) передавать нельзя, иначе сохранится какое-то предыдущее значение dC. Устройство показывает окошко со значением $dm = dF + dC$ = 44, и просит нашего согласия на синтез wav-файла. Соглашаемся. Вот результат:

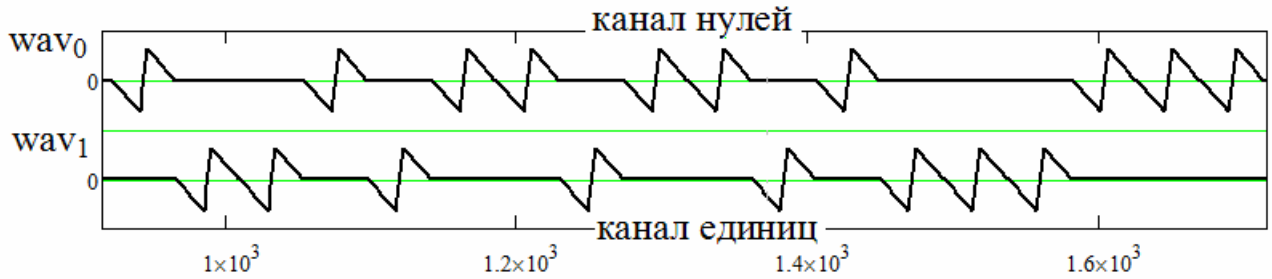


Рис. 27. 2-канальный сигнал с пилообразными импульсами для pi_kp-15_n-1.txt.

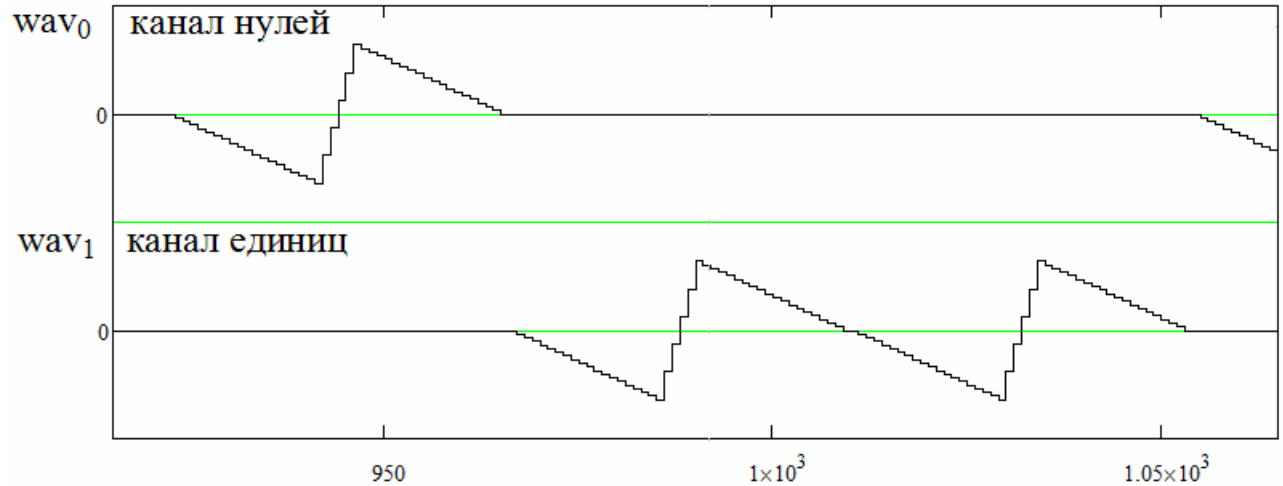


Рис. 28. Увеличенное изображение нескольких первых импульсов из рис. 27.

Напомню, что 2-канальный пилообразный сигнал предназначен для опытов с загрузкой ПО в реальную машину Д3-28 через её НМЛ с магнитной ленты, записанной на бытовом м/ф; этот вариант опытов обсуждался на форуме "Полигон Призраков" в теме Д3-28, устройства для загрузки ПО (способы, идеи, ... конструкции). Плюс такого варианта: ему не надо никакого ИУ.

Идея этого варианта в том, что пологие участки пилы должны давать слабый сигнал в катушках читающей головки, и машина их проигнорирует; а крутыми участками пилы наведутся в читающей головке большие импульсы, как от перепадов в обычном 2-канальном сигнале Д3-28, и машина их воспримет. Предполагается, что такого положения дел можно добиться подбором "громкости" записи при достаточно большом отношении производных (т. е. наклонов) на крутом и пологом участках пилы. В рассмотренном выше примере оценка для наклона крутого участка есть $2 \cdot Ampl / 5$, а на пологом: $Ampl / 19$; их отношение: $2 \cdot 19 / 5 = 7.6$.

Это отношение можно увеличить с помощью прежних упоминавшихся программок для управления Устройством и для синтеза 2-канальных wav-файлов с пилообразными импульсами при небольшой коррекции вручную. Как и в предыдущем примере, загрузим на нулевой адрес в имитируемое ОЗУ Д3-28 программку u_create_2ch-saw-pulse_wav_kp-2623_n-226.txt. В режиме "Ввод" в имитируемой машине скорректируем два байта: на шаге 00071 введём 0514 (вместо 0705; тем самым снимется защита от задания слишком малых значений параметра в команде "5"), и введём 0515 на шаге 00195 – для останова перед автоматическим выполнением команды "6". Заодно загрузим на адрес 20000 программку подачи команд для Устройства u_outo_kp-434_n-31.txt.

Допустим, как и в предыдущем примере, мы хотим иметь периоды длиной в 44 сэмпла. 44 можно составить из пологого + крутого + пологого участков пилы так: $44 = 21 + 2 + 21$. Длину = 2 крутой части пилы набираем в ответ на приглашение к макрокоманде "5", длины пологих частей набираем в виде их суммы = 42 в ответ на приглашение к макрокоманде "6". Перед шагом 00196 происходит останов. Тогда через режим "Ввод" задаём шаг 20000 и запускаем (разумеется, перейдя в режим "Работа") u_outo_kp-434_n-31.txt. Набираем номер команды "2", затем набираем какое-нибудь значение dM для этой команды, например, 46, – просто чтобы посмотреть, какое текущее значение dF (и какое dC) будет показано в окошке. Снова набираем команду "2", и набираем теперь число, равное $2 \cdot dF$, – чтобы получить $dC = 0$. Обманув таким образом Устройство (оно теперь запомнит значение $dC = 0$), возвращаемся на шаг 00196 и запускаем продолжение работы программки, создающей wav-файл.

В результате, создаётся файл с сигналом, аналогичным предыдущему – показанному на рис. 27, но с более выгодным отношением наклонов крутой и пологой частей пилы. Увеличенное изображение начала его информативной части, аналогичное рисунку 28, выглядит так:

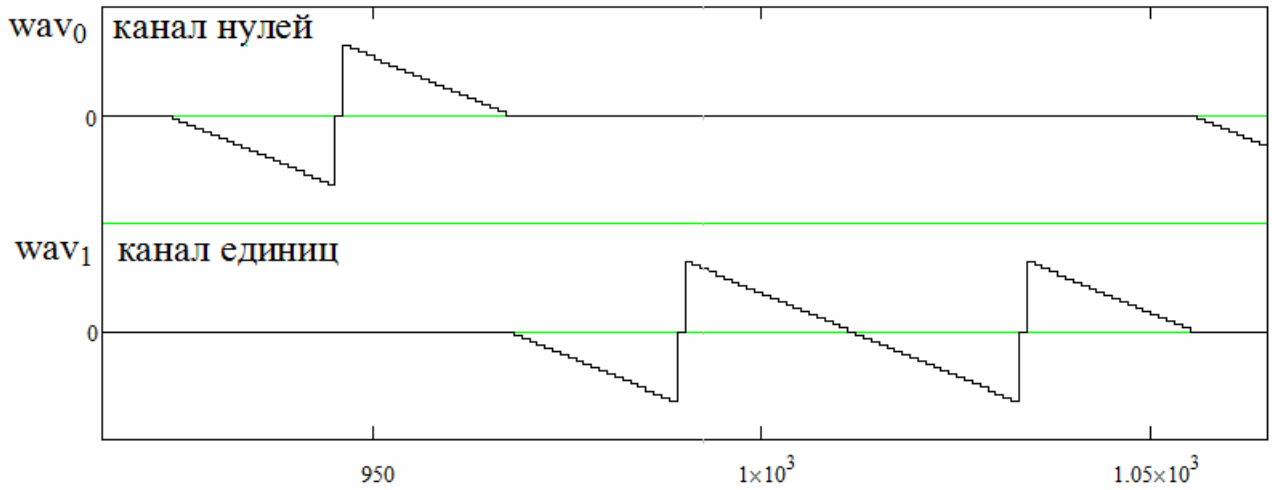


Рис. 29. Пилообразные импульсы с отношением наклонов ≈ 2·21 / 2 = 21.

На практике может оказаться, что длина пилы *dm*, равная 44 отсчётам при частоте дискретизации 44100 Гц, коротк*а*, т. е. машине требуется меньшая скорость загрузки с МЛ. Возможно, в зависимости от индивидуальных особенностей конкретного экземпляра Д3-28 и бытового м/ф, более подходящими для *dm* окажутся значения где-то в диапазоне 45 – 50. Может быть и так, что, с учётом, предположительно, вчетверо большей скорости протяжки ленты в НМЛ Д3-28, слишком короткими окажутся излишне крутые участки пилы. Если применяемый плеер способен проигрывать wav-файлы с произвольной частотой дискретизации *Fd*, то можно подбирать длительность периодов, выбирая для *Fd* нестандартные значения.

Рассмотренные выше примеры (в том числе с сигналами не в виде пилы) это именно лишь примеры; они показывают в принципе, как применять имеющиеся "u_программки". На деле, вероятно, придётся экспериментально подбирать значения параметров, чтобы добиться результата. Программки позволяют изменять значения параметров в широких пределах. Кроме того, пользователь при желании может изменять и сами программки – составлять свои собственные. Для такого рода попыток ниже приведены листинги основных программок.

(Напоминание о получении листингов. Листинг программок из txt-файлов можно получать встроенным в имитатор дизассемблером. Для этого не обязательно загружать интересующий txt-файл в "ОЗУ Д3-28", достаточно открыть его командой **File** > **Open txt-file as Tape**. Дизассемблироваться будут 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 набрать любое другое нужное значение. Конец текста дизассемблер определяет по местоположению команды 0512 (END); поэтому, 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. Запуск дизассемблера: **View** > **Display**, **F1**, **A** $_{\text{лат.}}$)

Пример 0: программка u_outo_kp-434_n-31.txt**, подающая команды в Устройство**:

Номер шага	Байт-код	Мнемоника команды	Пояснения действиям в команде
00000	0413 1000	CLR R00	очистка R_0: $R_0 \leftarrow (0000\ 0000)_{16}$,
00002	0413 0500	MOV R00,BD	очистка регистра BD: $R_0 \rightarrow$ BD
00004	1304 0715	MOV #0715,S04	$R_{10} \equiv S_4S_5 \leftarrow (0715\ 0000)_{16}$ = адрес старшего байта регистра R_0,
00006	1305 0000	MOV #0000,S05	
00008	0700	DIG 0	очистка X: $X \leftarrow 0$,
00009	0604	MOV X,Y	$X \rightarrow Y$ для индикации нулей на табло,
00010	0515	STOP	ожидание ввода в X номера макрокоманды,
00011	0604	MOV X,Y	$X \rightarrow Y$ для его индикации на Y-табло,
00012	0700	DIG 0	очистка X: $X \leftarrow 0$,
00013	0515	STOP	ожидание ввода в X параметра команды,
00014	0413 1200	MOVH X,R00	целая часть $(X)_{16} \rightarrow R_0$,
00016	1401 1010	OUTO 1010	вывод ст. байта R_0 в порт 0,
00018	1000 0110	ADD #01,R10	$R_{10} \leftarrow R_{10} + 1$ = адрес младшего байта регистра R_0,
00020	1401 1011	OUTO 1011	вывод мл. байта R_0 в порт 0,
00022	0605	MOV Y,X	$Y \rightarrow X$, это номер макрокоманды,
00023	0413 1201	MOVH X,R01	целая часть $(X)_{16} \rightarrow R_1$ (в младший байт),
00025	1000 0210	ADD #02,R10	$R_{10} \leftarrow R_{10} + 2$ = адрес младшего байта регистра R_1,
00027	1401 1014	OUTO 1014	вывод мл. байта R_1 в порт 4,
00029	1402 0114	BR 00000	переход в начало для ожидания нового ввода номера макрокоманды.
00031	0512	END	Здесь END это признак конца программки.

Эта программка предназначена для вспомогательных действий; главным образом, – для настройки тестирования получающихся 1-канальных wav-файлов. В ней не предусмотрена защита от некорректного ввода параметров. В частности, ввод значения параметра, превышающего 32767, приводит к зажиганию индикатора "Ошибка Программы" (ОП).

Пример 1: программка u_create_s-code_wav_kp-2226_n-182.txt**, предназначенная для создания wav-файла типа** "s-code из txt-программки" (как в *Примере 1*):

```
// Инициализация:
00000   0413 1000  CLR R00          очистка регистров R0, R1, R6 нулём,
00002   0413 1001  CLR R01
00004   0413 1006  CLR R06
00006   1304 0004  MOV #0004,S04    в регистр R10 ≡ S4S5 заносим число 1024(10),
00008   1305 0000  MOV #0000,S05
00010   0413 0510  MOV R10,BD       R10 ⟶ BD, стандартное BD для DC-ячеек,
00012   0413 1307  MOV BD,R07       сохраняем его в регистре R7,
00014   0702       DIG 2
00015   0705       DIG 5
00016   0706       DIG 6            константу 256 заносим в X
00017   0404 0001  MOV X,C1         и сохраняем её в DC-ячейке C1,

// начало первой из 3 команд для
// подготовки синтеза wav:
00019   0704       DIG 4            в X заносим номер команды 4,
00020   0514       GO               резервное место для 2-значного номера,
00021   0604       MOV X,Y          переписали X=4 в Y
00022   0700       DIG 0            и очистили X нулём;
00023   0515       STOP             стоп, это ожидание ввода в X значения
                                    параметра Fd, т.е. частоты дискретиз.

// пользователь что-то ввёл в X
// и нажал клавишу S
00024   0508       BLT Y,X 00027    при X > Y идём к шагу 00027,
00025   1403 0106  BR 00048         а иначе к шагу 00048 (будет старое Fd,
                                    это защита от ввода слишком малых чисел),
00027   0404 0002  MOV X,C2         сохранили X в C2,
00029   0604       MOV X,Y          и занесли его в Y, это новое Fd,

//выделяем из Fd ст. и мл. байты
00030   0405 0001  MOV C1,X         занесли в X константу 256 из C1,
00032   0603       DIV X,Y          Y ⟵ Y/X,
00033   0605       MOV Y,X          y ⟶ X,
00034   0413 1200  MOVH X,R00       целая часть (X)16 ⟶ R0 (в младший байт),
00036   0413 0400  MOVD R00,X       (R0)10 ⟶ X,
00038   0415 0001  MOV C1,Y         занесли в Y константу 256 из C1,
00040   0602       MUL X,Y          Y ⟵ Y·X,
00041   0405 0002  MOV C2,X         занесли в X значение Fd из C2,
00043   0606       SWA X,Y          обмен X ⟷ Y,
00044   0601       SUB X,Y          Y ⟵ Y-X,
00045   0606       SWA X,Y          обмен X ⟷ Y,
00046   0413 1201  MOVH X,R01       целая часть (X)16 ⟶ R1 (в младший байт),
00048   0413 0506  MOV R06,BD       R6=0 ⟶ BD для OUTO с адреса R10+BD,
00050   1304 0715  MOV #0715,S04    R10 ⟵ адрес младшего байта R0,
00052   1305 0001  MOV #0001,S05
00054   1401 1010  OUTO 1010        вывод мл. байта R0 в порт 0,
00056   1000 0210  ADD #02,R10       R10 ⟵ R10+2 = адрес младшего байта R1,
00058   1401 1011  OUTO 1011        вывод мл. байта R1 в порт 1,
00060   0704       DIG 4            в X заносим номер команды 4,
00061   0514       GO               резервное место для 2-значного номера,
00062   0413 1201  MOVH X,R01       целая часть (X)16 ⟶ R1 (в младший байт),
00064   1401 1014  OUTO 1014        вывод мл. байта R1 в порт 4,
```

```
// начало второй из 3 команд для
// подготовки синтеза wav:
00066   0705        DIG 5          в X заносим номер команды 5,
00067   0604        MOV X,Y        переписали X=5 в Y
00068   0700        DIG 0          и очистили X нулём;
00069   0515        STOP           стоп, ожидание ввода в X значения dF,

// пользователь что-то ввёл в X
// и нажал клавишу S
00070   0508        BLT Y,X 00073  при X > Y идём к шагу 00073,
00071   1402 0006   BR 00066       а иначе к 00066 для повторного ввода dF
                                   (защита от ввода слишком малых чисел),
00073   0604        MOV X,Y        переписали X в Y,
00074   0708        DIG 8
00075   0701        DIG 1
00076   0709        DIG 9
00077   0701        DIG 1          для защиты от ввода слишком больших
00078   0514        GO             чисел ввели X←8191 и затем
00079   0508        BLT Y,X 00082  при Y < X идём к шагу 00082,
00080   1402 0015   BR 00066       а иначе – к повторному вводу dF,
00082   0605        MOV Y,X        вернули Y = dF в регистр X,
00083   0413 1205   MOVH X,R05     целая часть (X)16 ⟶ R5,
00085   0413 0507   MOV R07,BD     R7=1024₁₀ ⟶ BD для десятичных вычислений
00087   0703        DIG 3
00088   0702        DIG 2          Задаём Ampl = 32000 в регистре X;
00089   0700        DIG 0          это значение здесь можно изменять
00090   0700        DIG 0          вручную,
00091   0700        DIG 0
00092   0514        GO             резервное место для 0711, т.е. NEG X,
00093   0404 0003   MOV X,C3        сохранили X=Ampl в DC-ячейке C3.

// Вычисление Ampl·sin(2·π·n / dF)  Будем вычислять значения перепада:
00095   0413 0405   MOVD R05,X     (R5)10 = dF ⟶ X, это число уже целое,
00097   0404 0004   MOV X,C4       сохранили его в десятичной ячейке C4,
00099   0609        PI             X ← PI = 3.14159265359,
00100   0604        MOV X,Y        переписали X в Y,
00101   0702        DIG 2          X ← 2,
00102   0602        MUL X,Y        Y ← Y·X = 2·PI,
00103   0405 0004   MOV C4,X       C4 = dF ⟶ X,
00105   0603        DIV X,Y        Y ← Y/X = 2·PI/dF,
00106   0414 0005   MOV Y,C5       Y сохранили в C5,
00108   0701        DIG 1
00109   0711        NEG X          в X занесли -1,
00110   0404 0006   MOV X,C6       занесли -1 в C6, C6 будет играть роль n

// начало цикла по n = 0, 1, …, dF–1
00112   0701        DIG 1          X ← 1,
00113   0400 0006   ADD X,C6       C6 ← C6 + X, это переменная n,
00115   0415 0006   MOV C6,Y       C6 = n ⟶ Y,
00117   0405 0005   MOV C5,X       C5 = 2·PI/dF ⟶ X,
00119   0602        MUL X,Y        Y ← Y·X = 2·PI·n /dF,
00120   0605        MOV Y,X        переписали Y в регистр X,
00121   0802        SIN            X ← sin(X),
00122   0604        MOV X,Y        переписали X в регистр Y,
00123   0405 0003   MOV C3,X       C3 = Ampl занесли в X,
00125   0602        MUL X,Y        Y ← Y·X = Ampl·sin(2·π·n / dF)
00126   0605        MOV Y,X        переписали Y в регистр X,
```

```
00127   1305 0000  MOV #0000,S05
00129   0413 1200  MOVH X,R00
00131   0413 0506  MOV R06,BD
00133   1401 1010  OUTO 1010
00135   1305 0001  MOV #0001,S05
00137   1401 1011  OUTO 1011
00139   0705       DIG 5
00140   0413 1200  MOVH X,R00
00142   1401 1014  OUTO 1014
00144   0413 0507  MOV R07,BD
// смотрим, не конец ли цикла:
00146   1408 0305  SOBZ R05 00150
00148   1402 0205  BR 00112

// начало 3-й из 3 команд для
// подготовки синтеза wav:
00150   0706       DIG 6
00151   0604       MOV X,Y
00152   0700       DIG 0
00153   0515       STOP

// пользователь что-то ввёл в X
// и нажал клавишу S
00154   0607       ABS X
00155   0604       MOV X,Y
00156   0708       DIG 8
00157   0701       DIG 1
00158   0709       DIG 9
00159   0701       DIG 1
00160   0514       GO
00161   0508       BLT Y,X 00164
00162   1402 0013  BR 00150
00164   0605       MOV Y,X
00165   0413 1200  MOVH X,R00
00167   1305 0000  MOV #0000,S05
00169   0413 0506  MOV R06,BD
00171   1401 1010  OUTO 1010
00173   1305 0001  MOV #0001,S05
00175   1401 1011  OUTO 1011
00177   0706       DIG 6
00178   0413 1200  MOVH X,R00
00180   1401 1014  OUTO 1014
00182   0512       END
```

```
S5←0, так что R10 = адрес ст. байта R0,
целая часть (X)16→R0,
R6=0→BD для OUTO с адреса R10+BD,
вывод ст. байта R0 в порт 0,
S5←1, так что R10 = адрес мл. байта R0,
вывод мл. байта R0 в порт 1,
в X занесли номер команды 5,
целая часть (X)16→R0 (в младший байт),
вывод номера команды = 5 в порт 4,
вернули в BD стандартное значение,

R5←R5-1, при R5←0 выходим из цикла,
а иначе идём к шагу 00112, к началу;



в X заносим номер команды 6,
переписали X=6 в Y
и очистили X нулём;
стоп, ожидание ввода в X значения dC,



X←|X|, на всякий случай убрали минус;
переписали X в Y,



для защиты от ввода слишком больших
чисел ввели X←8191 и затем
при Y < X идём к шагу 00164,
а иначе – к повторному вводу dC,
вернули введённое dC из Y в X,
целая часть (X)16→R0,
S5←0, так что R10 = адрес ст. байта R0,
R6=0→BD для OUTO с адреса R10+BD,
вывод ст. байта R0 в порт 0,
S5←1, так что R10 = адрес мл. байта R0,
вывод мл. байта R0 в порт 1,
в X занесли номер команды 6,
целая часть (X)16→R0 (в младший байт),
вывод номера команды = 6 в порт 4,
конец управления синтезом wav-файла
типа "s-code из txt-программки".
```

Пример 2: программка u_buff_to_s-code_wav_kp-2200_n-182.txt**, предназначенная для создания wav-файла типа** "s-code из чисел в Буфере" (как в *Примере 2*):

Эта программка почти такая же, как предыдущая; вот её отличающиеся шаги:

```
00019   0701       DIG 1          в X заносим номер команды 12 вместо 4,
00020   0702       DIG 2

00060   0701       DIG 1          в X заносим номер команды 12 вместо 4.
00061   0702       DIG 2
```

Пример 3: программка u_buff_to_m-code_wav_kp-2220_n-182.txt, **предназначенная для создания wav-файла типа** "m-code из чисел в Буфере" (как в *Примере 3*):

Отличия от программки Примера 1:

```
00019   0701       DIG 1          в X заносим номер команды 11 вместо 4,
00020   0701       DIG 1

00060   0701       DIG 1          в X заносим номер команды 11 вместо 4,
00061   0701       DIG 1

00101   0514       GO             для Ampl·cos( π·n / dF) не надо умножать
00102   0514       GO                                      2 на π.

00121   0803       COS            вычисляем cos(X) вместо sin(X).
```

Пример 4: программка u_create_m-code_wav_kp-2256_n-182.txt, **предназначенная для создания wav-файла типа** "m-code из txt-программки " (как в *Примере 4*):

Отличия от программки Примера 1:

```
00019   0708       DIG 8          в X заносим номер команды 8 вместо 4,

00060   0708       DIG 8          в X заносим номер команды 8 вместо 4,



00101   0514       GO             для Ampl·cos( π·n / dF) не надо умножать
00102   0514       GO                                    2 на π.

00121   0803       COS            вычисляем cos(X) вместо sin(X).
```

Пример 5: программка u_create_m-code-exp_wav_kp-2626_n-206.txt, **для создания wav-файла типа** "m-code из txt-программки" **с перепадами экспоненциальной формы**:

Отличия от программки Примера 1:

```
00019   0708       DIG 8          в X заносим номер команды 8 вместо 4,

00060   0708       DIG 8          в X заносим номер команды 8 вместо 4,

00117   0514       GO             теперь в цикле не вычисляем перепад явно,
00118   0514       GO             вычисления перенесены в подпрограмму
00119   0514       GO             с меткой "0315".
00120   0315       JSM 0315       Вызов подпрограммы "0315",
00121   0514       GO

00182   0515       STOP           конец управления синтезом wav-файла,
00183   0514       GO             просто резервное место,
00184   0514       GO             просто резервное место.
```

```
00185   0408 0315  MARK 0315       Метка "0315", начало подпрограммы
00187   0712       POINT           вычисления f(n) = 2·exp(–0.75·n) – 1,
00188   0707       DIG 7
00189   0705       DIG 5           ввели в X число 0.75 (оно влияет
                                   на крутизну экспоненциального перепада)
00190   0514       GO              резервное место для уточнения этого
00191   0514       GO              параметра вручную,
00192   0711       NEG X           изменение знака X, теперь X = -0.75,
00193   0415 0006  MOV C6,Y        занесли в Y переменную C6 = n = 0, 1, 2, …, dF–1,
00195   0602       MUL X,Y         Y ← Y·X = – 0.75·n ,
00196   0605       MOV Y,X         Y → X,
00197   0614       EXP             X ← exp(X),
00198   0604       MOV X,Y         X → Y,
00199   0702       DIG 2           занесли в X число 2,
00200   0602       MUL X,Y         Y ← Y·X
00201   0701       DIG 1           занесли в X число 1,
00202   0601       SUB X,Y         Y ← Y-X
00203   0514       GO
00204   0605       MOV Y,X         Y → X, это возвращаемое значение f(n) .
00205   0511       RTS             Возврат из подпрограммы, конец п/п.
00206   0512       END             Признак конца программки.
```

Пример 6: программка u_create_m-code-step_wav_kp-2594_n-206.txt, **для создания wav-файла типа** "m-code из txt-программки" **с перепадами ступенчатой формы**:

Отличия от программки Примера 5 (т. е. от предыдущего примера) есть только внутри подпрограммы вычисления перепада:

```
00187   0415 0004  MOV C4,Y        значение C4 = dF занесли в регистр Y,
00189   0702       DIG 2           число 2 занесли в регистр X,
00190   0603       DIV X,Y         Y ← Y/X,
00191   0605       MOV Y,X         Y → X,
00192   0608       INT             X ← целая часть от X = dF / 2,
00193   0404 0005  MOV X,C5        X → C5,
00195   0415 0006  MOV C6,Y        C6 = n → Y,
00197   0507       BGE Y,X 00200   при Y ≥ X переход к шагу 00200,
00198   0711       NEG X           а иначе изменяем знак X и заносим
00199   0604       MOV X,Y         X в регистр Y,
00200   0711       NEG X           изменяем знак X и заносим
00201   0604       MOV X,Y         X в регистр Y,
00202   0607       ABS X           X← |X|, убрали минус, если он был,
00203   0603       DIV X,Y         Y ← Y/X, это есть f(n) = ± 1.
```

Пример 7: программка u_create_s-code-step_wav_kp-2586_n-206.txt, **для создания wav-файла типа** "s-code из txt-программки" **с перепадами ступенчатой формы**:

Отличия от программки Примера 6 (т. е. от предыдущего примера):

```
00019   0704       DIG 4           в X заносим номер команды 4 вместо 8,

00060   0704       DIG 4           в X заносим номер команды 4 вместо 8.
```

Пример 8: программка u_create_2ch-d3-28_wav_kp-2258_n-182.txt,
для создания wav-файла типа "2-канальный обычный Д3-28" (как в *Примере 8*):

Отличия от программки Примера 4 (там был m-code, 1-канальный):

```
00019   0709        DIG 9           в X заносим номер команды 9 вместо 8,

00060   0709        DIG 9           в X заносим номер команды 9 вместо 8.
```

Эта же программка применялась в *Примере 9* и в *Примере 10*; там демонстрировалось только изменение знака начального уровня и перестановка каналов программкой подачи команд в Устройство (из Примера 0). При этом знак *Ampl* изменять нельзя, так как при $Ampl < 0$ образуются сигналы неправильной формы. Переходим к следующему примеру:

Пример 11: программка u_create_2ch-s-pulse_wav_kp-2196_n-182.txt,
для создания wav-файла типа "2-канальный с sin-импульсами" (как в *Примере 11*):

Отличия от программки Примера 1 (там был s-code, 1-канальный):

```
00019   0701        DIG 1           в X заносим номер команды 10 вместо 4,
00020   0700        DIG 0

00060   0701        DIG 1           в X заносим номер команды 10 вместо 4,
00061   0700        DIG 0
```

Пример 12: программка u_create_2ch-saw-pulse_wav_kp-2623_n-226.txt,
для создания wav-файла типа "2-канальный с пиловидными импульсами" (как в *Примере 12*):

Отличия от программки Примера 1 (там был s-code, 1-канальный):

```
00019   0701        DIG 1           в X заносим номер команды 10 вместо 4,
00020   0700        DIG 0

00060   0701        DIG 1           в X заносим номер команды 10 вместо 4.
00061   0700        DIG 0
                                    На шаге 00069 - ожидание ввода как бы dF,
                                    это будет длина крутого участка пилы,
00071   0705        DIG 5           если в X пользователь ввёл число < 5,
00072   0514        GO              то заменяем его на X = 5 (это защита
                                    от ввода слишком малых значений)
00087   0608        INT             X ← целая часть от X, это как бы dF,
00088   0404 0004   MOV X,C4        сохранили X в десятичной ячейке C4,
00090   0703        DIG 3
00091   0702        DIG 2           задаём Ampl = 32000 в регистре X;
00092   0700        DIG 0           это значение здесь можно изменять
00093   0700        DIG 0           вручную,
00094   0700        DIG 0
00095   0514        GO              резервное место для 0711, т.е. NEG X,
00096   0404 0003   MOV X,C3        сохранили X = Ampl в DC-ячейке C3.
00098   0604        MOV X,Y
00099   0702        DIG 2
00100   0602        MUL X,Y
00101   0405 0004   MOV C4,X
00103   0603        DIV X,Y         Y ← Y/X = 2·Ampl/dF,
00104   0414 0005   MOV Y,C5       сохранили Y = 2·Ampl/dF в ячейке C5,
```

```
00106   0706       DIG 6          в X заносим номер команды 6,
00107   0604       MOV X,Y        переписали X=6 в Y,
00108   0700       DIG 0          и очистили X нулём;
00109   0515       STOP           стоп, ожидание ввода в X как бы dC,
                                  это удвоенная длина пологого участка пилы,

00110   0607       ABS X
00111   0604       MOV X,Y
00112   0708       DIG 8
00113   0701       DIG 1
00114   0709       DIG 9
00115   0701       DIG 1          для защиты от ввода слишком больших чисел
00116   0514       GO             чисел ввели X = 8191 и затем
00117   0508       BLT Y,X 00120       при Y < X идём к шагу 00120,
00118   1402 0013  BR 00106            а иначе – к повторному вводу,
00120   0702       DIG 2
00121   0603       DIV X,Y
00122   0605       MOV Y,X
00123   0413 1204  MOVH X,R04     сохранили целую часть от dC/2 в R4
00125   0608       INT            (будет счётчиком для цикла),
00126   0404 0007  MOV X,C7
00128   0415 0003  MOV C3,Y
00130   0603       DIV X,Y
00131   0414 0008  MOV Y,C8       сохранили 2·Ampl/dC в ячейке C8,
00133   0700       DIG 0
00134   0404 0006  MOV X,C6       занесли 0 в C6, C6 играет роль n в формуле,
00136   1104 0403  MOV R04,R03    сохранили начальное значение из R4 в R3,

// начало первой части цикла "команды 5":
00138   0701       DIG 1          занесли 1 в X,
00139   0400 0006  ADD X,C6       занесли в C6 число C6+X, это текущее n,
00141   0415 0006  MOV C6,Y
00143   0405 0008  MOV C8,X
00145   0711       NEG X
00146   0602       MUL X,Y        в Y получили значение пилы -2·Ampl·n/dC,
00147   0705       DIG 5          в X занесли номер макрокоманды 5,
00148   0314       JSM 0314       вызов п/п для вывода в Устройство X и Y,

00149   1408 0304  SOBZ R04 00153    R4←R4-1, при R4←0 выходим из цикла,
00151   1402 0014  BR 00138          а иначе идём к шагу 00138, к началу;

00153   0700       DIG 0          готовимся ко второй части цикла,
00154   0404 0006  MOV X,C6

// начало второй части цикла "команды 5":
00156   0701       DIG 1
00157   0400 0006  ADD X,C6       заново считаем текущее n,
00159   0415 0006  MOV C6,Y
00161   0405 0005  MOV C5,X       X←C5 = 2·Ampl/dF,
00163   0602       MUL X,Y
00164   0405 0003  MOV C3,X       X ← Ampl
00166   0711       NEG X
00167   0600       ADD X,Y        в Y получили значение –Ampl + 2·Ampl·n/dF,
00168   0705       DIG 5          в X занесли номер макрокоманды 5,
00169   0314       JSM 0314       вызов п/п для вывода в Устройство X и Y,

00170   1408 0305  SOBZ R05 00174    R5←R5-1, при R5←0 выходим из цикла,
00172   1402 0101  BR 00156          а иначе идём к шагу 00156, к началу;
```

```
00174   0700       DIG 0          готовимся к третьей части цикла,
00175   0404 0006  MOV X,C6

// начало третьей части цикла "команды 5":
00177   0701       DIG 1
00178   0400 0006  ADD X,C6       заново считаем текущее n,
00180   0415 0006  MOV C6,Y
00182   0405 0008  MOV C8,X
00184   0711       NEG X
00185   0602       MUL X,Y
00186   0405 0003  MOV C3,X
00188   0600       ADD X,Y        в Y получили значение Ampl - 2·Ampl·n/dC,
00189   0705       DIG 5          в X занесли номер макрокоманды 5,
00190   0314       JSM 0314       вызов п/п для вывода в Устройство X и Y,

00191   1408 0303  SOBZ R03 00195    R3←R3-1, при R3←0 выходим из цикла,
00193   1402 0101  BR 00177           а иначе идём к шагу 00177, к началу;

00195   0701       DIG 1
00196   0604       MOV X,Y        в Y занесли значение 1 для dC в команде 6,
00197   0706       DIG 6          в X занесли номер макрокоманды 6,
00198   0314       JSM 0314       вызов п/п для вывода в Устройство X и Y,
00199   0515       STOP           останов после исполнения команды 6.
00200   1403 0109  BR 00226       Переход на END.

// Подпрограмма вывода в Устройство значений из X и Y:
00202   0408 0314  MARK 0314      метка начала подпрограммы "0314",
00204   0413 1202  MOVH X,R02
00206   0605       MOV Y,X
00207   0413 1200  MOVH X,R00
00209   0413 0506  MOV R06,BD
00211   1305 0000  MOV #0000,S05
00213   1401 1010  OUTO 1010
00215   1305 0001  MOV #0001,S05
00217   1401 1011  OUTO 1011
00219   1104 0200  MOV R02,R00
00221   1401 1014  OUTO 1014
00223   0413 0507  MOV R07,BD
00225   0511       RTS            возврат из подпрограммы,
00226   0512       END            признак конца программки.
```

Эта программка управляет созданием 2-канальной "пилы" типа показанной на рис. 28. Для получения более крутого участка пилы (как на рис. 29) надо вручную внести изменения (и затем действовать так, как указано в конце *Примера 12*):

```
00071   0514       GO             вместо 0705,

00195   0515       STOP           вместо 0701.
```

Детали создания wav-файлов "Устройством" можно посмотреть в исходнике имитатора (в текстовом файле imitator.cpp), найдя там поиском по файлу имена подпрограмм:

OurCreateWaveFile1 – для 1-канальных wav-файлов, OurCreateWaveFile2 – для 2-канальных wav-файлов,
OurWav2WritePeriodM – формирование периода для 2-канального "обычного Д3-28" wav-файла,
OurWav2WritePeriodS – формирование периода для 2-канального "с sin-импульсами" wav-файла,
OurWav2WriteByte – запись байта в 2-канальный файл, OurWav1WriteByte – в 1-канальный файл,
OurWav1WritePeriodMcode – формирование периода для 1-канального wav-файла типа m-code.
OurWav1WritePeriodScode – формирование периода для 1-канального wav-файла типа s-code.

4. Инструкция по изготовлению wav-файлов диалоговыми программками

В папке u_soft есть "дружественные к пользователю" программки – диалоговые: в них параметры создаваемого wav-файла можно задавать в форме числовых ответов на вопросы, появляющиеся на экране имитируемого фрязинского дисплея. Число 0 удобно вводить просто нажатием клавиши Enter.

Для создания wav-файлов различного типа (их примеры приведены в предыдущем разделе) я скомпилировал с помощью "Fortran-5M для Д3-28" семь рабочих программок (РП), предоставляющих в имитаторе диалоговый интерфейс:

1scw_v1_kp-70694_n-6368.txt – создаёт wav-файл типа "s-code из txt-программки". Этот тип wav считаю основным в данном проекте; он предназначен для загрузки ПО в Д3-28 через ИУ, обсуждавшееся в разделе 2, загрузчиком loader_m-code_addr-x_kp-1812_n-138.txt (раздел 6) 1scw означает 1-канальный s-code wav, v1 – версия 1.

1scw_v2_kp-71632_n-6467.txt – создаёт wav-файл типа "s-code из чисел в Буфере" без добавления "по умолчанию" начальных нулей, синхробайта и т. п. (всё нужное уже должно быть в Буфере). Предназначается для загрузки ПО через ИУ (преобразующее s-код в м-код), загрузчиком, который пользователь составил на свой лад. v2 – версия 2; указанным выше она и отличается от версии 1.

1mcw_v1_kp-70632_n-6366.txt – создаёт wav-файл типа "m-code из txt-программки". Для какого-то контроллера, способного напрямую передавать ТТЛ-сигнал с м-кодом в Д3-28, без преобразования из s-кода. Загрузчик: loader_m-code_addr-x_kp-1812_n-138.txt. 1mcw означает 1-канальный m-code wav.

1mcw_v2_kp-71674_n-6466.txt – создаёт wav-файл типа "m-code из чисел в Буфере", т. е. без добавок "по умолчанию", в остальном – аналогичный предыдущему. Построение данных в txt-файле для Буфера и загрузчик – пользовательские; это и означает метка версии 2.

2mcw_v1_kp-79117_n-7103.txt – создаёт wav-файл типа "2-канальный обычный Д3-28". Для какого-то контроллера, способного управлять токами в записывающей стерео-головке, включённой по схеме Д3-28 во внешнем ЛПМ (как это кратко обсуждалось в разделе 0). 2mcw означает 2-канальный меандрового вида код wav; соответствует примерам 8, 9, 10 в предыдущем разделе.

2scw_v1_kp-79280_n-7100.txt – создаёт wav-файл типа "2-канальный с sin-импульсами" вместо перепадов меандрового вида; см. пример 11 в пред. разд. Для управления токами в записывающей стерео-головке, включённой по схеме Д3-28 во внешнем ЛПМ, с 2-канальным ИУ, преобразующим D-триггерами sin-импульсы в сигнал меандрового вида; см. Приложение 2. Либо – для опытов с записью на МЛ "2-канальных sin-импульсов" бытовым м/ф. 2scw означает 2-канальный с sin-импульсным кодом wav.

2www_v1_kp-84758_n-7551.txt – создаёт wav-файл типа "2-канальный с saw-импульсами", т. е. с пилообразными импульсами; см. пример 12 в пред. разд. Для записи на МЛ бытовым м/ф. www – символизирует пилу.

Программки, создающие 2-канальные wav-файлы, позволяют настраивать "Устройство" – задавать знаки начальных уровней, длительность концевых пауз, переставлять каналы.

Кроме того, в папке u_soft есть бейсик-программка u_create_wav__bas.txt. Она создаёт wav-файлы под управлением Бейсика, причём она одна заменяет все перечисленные РП – позволяет в своём меню выбрать для создания любой из названных выше типов wav.

Создание wav-файлов в РП, скомпилированных в Fortran-5M

Рассмотрим сначала правила работы с перечисленными выше РП. Правила эти простые и примерно одинаковые для всех РП; подробно разберём по шагам один пример.

1) Первый шаг – определяемся с типом эксперимента по загрузке ПО в Д3-28 и соответственно выбираем тип создаваемых wav-файлов. Допустим, например, у нас есть интерфейсное устройство, подобное обсуждавшемуся в разделе 2 (т. е. ИУ, преобразующее s-код в m-код). Тогда желательно создавать wav-файлы типа "s-code из txt-программки".

2) Второй шаг – решаем, какую из программок хочется загрузить в Д3-28 (чтобы затем сохранить её на МЛ) на текущем этапе в первую очередь. Копируем txt-файл с выбранной программкой из папки txt из архива с имитатором (или из другого источника) в папку u_soft.

Например, если это ещё лишь первый эксперимент, то можно начать с короткой программки y-test__KP-88_N-8.txt, чтобы убедиться в перспективности дальнейших опытов. Затем можно будет взять более длинную программку statistika__KP-7074_N-806.txt; она входит в состав стандартного ПО для Д3-28 и хорошо документирована: существует брошюра с описанием работы в этой программе. Затем, в случае успеха, берём документированную в "Инструкции по эксплуатации" Д3-28 большую тест-программу 028-test_KP-73370_N-8921.txt. Всё это ПО работоспособно на "голой" Д3-28, без фрязинского дисплея. Если опыты удаются, и есть дисплейный терминал, то следующими кандидатами на загрузку будут большие программы с "языками высокого уровня" – Basic-157107, Forttran-5M, Выстра, ОС ВТ-МХТИ и др.

Если никакое ПО загрузить в исправную Д3-28 не удаётся, то одной из вероятных причин этого может оказаться неудачный выбор динамики загрузки. Тогда целесообразно составить свой собственный txt-файл, содержащий вместо программки, например, одни лишь нулевые байты в достаточном количестве, и пытаться загружать такого рода тест-сигнал, изменяя параметры wav-файлов и контролируя осциллографом динамику напряжений Ввв1, СИП и Вв на шине "Ввод-вывод" Д3-28. Возможно, в итоге таким путём удастся подобрать подходящую длительность периода в wav-файле и / или значение параметра задержки в загрузчике.

Далее для примера полагаю, что мы собрались загружать 028-test_KP-73370_N-8921.txt, и скопировали этот файл в папку u_soft. Создавать wav-файл решили программкой 1scw_v1_kp-70694_n-6368.txt.

3) Следует решить, задавать ли свои значения параметров или соглашаться на значения "по умолчанию" – частота дискретизации = 44100 сэмплов / сек, длина периода с одним битом из программки = 176 сэмплов; при этом время передачи одного бита равно 176 / 44100 ≈ 0.004 сек. Предполагаемые свои значения надо заранее обдумать, выполнить для них оценки.

Например, если загрузка с прежней скоростью уже удавалась, но теперь не устраивает большой размер получающихся wav-файлов, то можно попробовать уменьшить в одно и то же число раз длину периода и частоту дискретизации. Так, прежняя скорость передачи (1 бит за 0.004 сек) будет при частоте дискретизации = 10000 сэмплов / сек и длине периода = 40 сэмплов, а размер wav-файла при этом уменьшится примерно в 4 раза. Длина периода *dM* в сигнале типа s-code (и m-code) равна удвоенной сумме длины перепада и длины промежутка между ближайшими перепадами. Значит, в данном примере эта сумма = 20 сэмплов, и можно попробовать выбрать

для длин перепада *dF* и промежутка *dC*, например, значение 10 сэмплов. (От *dF* может зависеть надёжность работы преобразователя s-code в m-code в конкретной реализации ИУ, так что, не исключено, с этим параметром придётся поэкспериментировать). Аналогичным образом оценивается нужная длина периода при любом значении времени передачи одного бита: **количество сэмплов в периоде = (время передачи бита)·(частота дискретизации).**

Далее для примера полагаю, что принято решение воспользоваться значениями по умолчанию.

4) Запускаем на исполнение имитатор Д3-28, версия 1.2, – файл imitator.exe. Если он уже был запущен раньше, то "освежаем" его – делаем его ресет клавишей Escape.

5) Через меню имитатора командой **File** > **Open txt-file as Tape** открываем из папки u_soft файл 1scw_v1_kp-70694_n-6368.txt. Загружаем его в имитируемое ОЗУ Д3-28 на нулевой шаг и запускаем; всё как обычно – клавишами L, S. Загорается лампочка ожидания СИП.

6) В меню имитатора выбираем **View** > **Display**. Необязательное действие: сбрасываем в ноль параметры в служебной строке имитируемого экрана дисплея, т. е. нажимаем клавиши F8, F9, F8. Если этого не сделать, то между строками текста на экран будут выводиться лишние (не предусмотренные в программке) пустые строки; хорошо это или плохо – дело вкуса, на создание wav-файла это не влияет.

7) В этом же режиме (т. е. **View** > **Display**) нажимаем клавишу F1 (появляется изображение клавиатуры фрязинского терминала) и затем нажимаем клавишу U на своей клавиатуре ПК (появляется окошко с вопросом "Change ?"). Соглашаемся изменить статус "Устройства" с OFF на ON, т. е. включаем Устройство для создания wav-файлов.

8) В этом же режиме (**View** > **Display** > F1) нажимаем Т $_{лат}$ и открываем из папки u_soft txt-файл 028-test_KP-73370_N-8921.txt – это выбранная нами программка, которую собираемся преобразовать в wav. Если всё идёт нормально, то окошко сообщит, что Буфер с байт-кодами создан, количество данных в нём = 8922. Нажимаем там ОК.

9) Возвращаемся к экрану дисплея нажатием F1. Нажимаем F5 – это имитация включения режима связи ДУП-ЛИН дисплея с Д3-28. Программка 1scw_v1_kp-70694_n-6368.txt выведет на экран букву R – это признак готовности (Ready) к старту. Нажимаем S, она отобразится на экране как С в регистре РУС. Начинается диалог с вводом параметров.

Вариант: если до этого момента не были выполнены пункты 7 и 8 (например, мы забыли про них), то программка после старта ничего не выводит на экран, а ожидает СИП от Устройства. Пугаться этого не надо. Включаем Устройство, как учит пункт 7; программка предложит открыть в Устройстве txt-файл для преобразования в wav и попросит подтвердить выполнение. Выполняем пункт 8 (если надо), подтверждаем клавишей Enter, и тогда начинается диалог.

10) Диалог с вводом параметров:

```
КОЛ-ВО TXT-ДАННЫХ ДЛЯ КОНВЕРСИИ В S-CODE WAV:   8922
ЗАДАЕМ ПАРАМЕТРЫ КОНВЕРСИИ TXT-ФАЙЛА В S-CODE WAV

АМПЛИТУДА 32000 ? ДА - 0, НЕТ - НОВОЕ ЗНАЧЕНИЕ
```

Поскольку в данном примере мы решили соглашаться с параметрами по умолчанию, то просто нажимаем Enter (это то же самое, что вводить ноль, т. е. нажимать 0 и затем Enter). Так же, нажатием Enter, отвечаем и на остальные вопросы:

```
SAMPLE RATE 44100 ? ДА - 0, НЕТ - НОВОЕ ЗНАЧЕНИЕ
ОТСЧЕТОВ В S-ИМПУЛЬСЕ 23 ? ДА - 0, НЕТ – НОВОЕ

МЕЖДУ S-ИМПУЛЬСАМИ 65 ? ДА - 0, НЕТ – НОВОЕ
```

Затем программка для проверки выводит на экран принятые значения параметров и сообщает вычисленные ею ожидаемые размер и длительность воспроизведения wav-файла:

```
ИТАК, БУДЕТ СОЗДАН WAV-ФАЙЛ С ПАРАМЕТРАМИ:
ЧАСТОТА ДИСКРЕТИЗАЦИИ:  44100.
КАНАЛОВ: 1, БИТ В СЭМПЛЕ: 16, ТИП СИГНАЛА: S-CODE
dF:   23.    dC:   65.    dM:   176.    Ampl: 32000.
РАЗМЕР WAV-ФАЙЛА, БАЙТ:  25498926.
ПРОДОЛЖИТЕЛЬНОСТЬ, МИНУТ:   4.     СЕКУНД:  49.

СОЗДАТЬ ?  ДА - 0, НОВЫЙ ВВОД ПАРАМЕТРОВ - НЕ 0
```

Если что-то не устраивает, то можно ввести любое ненулевое число, и тогда диалог повторится. Кстати, чтобы остановить работу программы, можно в ответ на запросы параметров (кроме запроса амплитуды) вводить какое-нибудь отрицательное число, например, –1.

11) Если всё устраивает, то нажимаем Enter. Тогда, как и в *Примере* 1 раздела 3, появится окно со значениями параметров и вопросом: "начать конвертировать txt-данные в s-code-wav?" Если ответить "Нет", то имитируемая машина Д3-28 будет остановлена; тогда перед запуском той же или другой программки надо будет перейти в режим **View** > **Machine** и сделать сброс клавишей C.

Отвечаем "Да".

12) Появляется стандартное окно "Сохранить как". В нём wav-файлу можно дать желаемое имя. По умолчанию предлагается имя со значениями параметров *Fd*, *dM*, *dF*, с типом сигнала и номером N шага с кодом END: 001_Our_44100_s-code_dM-176_dF-23_N-8921.wav.

Нажимаем в этом окне кнопку "Сохранить". Устройство создаёт wav-файл, помещает его в папку u_soft и сообщает: Done, ok. Нажимаем OK. Программка пишет на экране:

```
S-CODE WAV-ФАЙЛ СОЗДАН
ПРОГРАММА ЗАВЕРШЕНА
```

В конце она выводит на экран букву R – признак готовности (Ready) к очередному старту.

Работа с остальными РП выглядит примерно так же.

В программки, создающие 2-канальные wav-файлы, включена возможность настраивать "Устройство" – каждая из таких программок начинается с предложения выбрать действие (здесь 2-ch wav означает 2-канальный wav-файл):

```
СОЗДАНИЕ 2-CH WAV - 0,  НАСТРОЙКА УСТРОЙСТВА - НЕ 0
```

После ввода какого-нибудь не равного нулю числа, например, 1, появятся два запроса:

```
ВВЕДИТЕ НОМЕР КОМАНДЫ ДЛЯ УСТРОЙСТВА
```

и

```
ВВЕДИТЕ ЗНАЧЕНИЕ ПАРАМЕТРА КОМАНДЫ
```

Допустимые номера команд здесь: 1, 2, 3 и 13, 14, 15, 16 (см. табл. 2 в разделе 3). Абсолютная величина значения параметра команды не должна превышать 32767. При создании двухканальных wav-файлов могут потребоваться только команды 15 и 16.

Значение параметра для команды 15 – десятичное число (по умолчанию равное нулю) в общем случае с тремя цифрами $A_3A_2A_1$. Оно управляет раскладкой каналов нулей и единиц с битами программки по двум каналам данных (wav_0 и wav_1) в wav-файле, а также задаёт знаки ненулевых начальных уровней в 2-канальном wav-файле "обычного для Д3-28" типа. Всё это иллюстрировалось в *Примерах 8, 9, 10* в предыдущем разделе; см. рисунки 22, 23, 24.

Команда 16 позволяет изменить длину концевых пауз *dP*, по умолчанию равную 20 длинам периода: $dP = 20 \cdot dm$. Нюанс: здесь значение нового коэффициента (вместо 20) надо вводить умноженным на 256. Например, чтобы получить новую длину паузы $dP = 10 \cdot dm$, надо в качестве значения параметра команды 16 ввести число $256 \cdot 10 = 2560$ (т. е. надо набрать 2560 на клавиатуре ПК и нажать Enter). При этом абсолютная величина вводимого числа не должна превышать 32767.

Пояснение: значение параметра программка записывает в 2-байтовую переменную A. 2-байтовая переменная в ЭВМ Д3-28 (и в Фортране для Д3-28) интерпретируется как двухбайтовое целое число со знаком. Номер команды записывается в младший байт двухбайтовой переменной B. Затем подпрограмма ABWY(K) в программке выводит A и B в однобайтовые порты "Устройства" следующим образом:

(порт 0)·256 + (порт 1) = A, если номер команды есть 1, 2 или 3,
(порт 2)·256 + (порт 3) = A, если номер команды есть 13, 14, 15 или 16,
(порт 4) = младший байт числа B.

Поэтому для команды 15, реагирующей только на содержимое порта 3, достаточно задавать значения A, не превышающие 255 (или даже 111, так как для этой команды важен лишь факт неравенства нулю десятичных цифр). A для команды 16, реагирующей только на содержимое порта 2, приходится задавать значения A, превосходящие 255, так как меньшие значения равноценны заданию нуля (нуль восстанавливает длину паузы "по умолчанию"). Чтобы получать (порт 2) > 127, следовало бы вводить отрицательные значения A, но вряд ли на практике потребуются такие длинные паузы.

Команды 1 – 3 могут применяться при тестировании в имитаторе загрузки 1-канального wav-файла; их удобнее подавать с помощью короткой перемещаемой программки в машинных кодах u_outo_kp-434_n-31.txt. Их применение вручную не актуально при диалоговом создании wav-файлов (хотя и можно пробовать их подавать с равным нулю значением параметра).

Команда 13 (параметр для неё не играет роли – можно просто нажимать Enter) записывает в папку u_soft текстовый файл с именем 001_Our_buffer.txt. В него выводится содержимое Буфера Устройства в виде столбца целых десятичных чисел.

Если в Буфер Устройства была открыта txt-программка (или bin-файл с hex-кодами), то выводятся в тетрадно-десятичной форме байт-коды данных в Буфере. Если в Буфер был открыт 1-канальный wav-файл, то выведутся десятичные значения сэмплов. Это может пригодиться, во-первых, для контроля, а во-вторых – для преобразования и редактирования данных bin ⇔ txt.

Например, некий найденный в интернете файл с расширением .rk можно переименовать в .bin, открыть в Буфер, распечатать в файл 001_Our_buffer.txt, отредактировать обычным текстовым редактором (добавить желаемым образом начальные нули, синхробайт и т. п. – всё в тетрадно-десятичной форме), открыть опять в Буфер, сохранить (если надо) командой 14 в файл 001_Our_buffer.bin, и создать wav-файл с м- кодом с желаемой длиной периода для соответствующей ЭВМ. Наши диалоговые программки ориентированы на ЭВМ Д3-28, они следят для защиты от ошибок при открытии txt- и bin-файлов за тем, чтобы размер данных в Буфере не превышал доступный объём ОЗУ Д3-28 – примерно 32 килобайта. Поэтому для создания таким путём wav-файлов из данных большего размера следует пользоваться программками "в машинных кодах", как пояснялось в предыдущем разделе.

Команда 14 (параметр для неё не играет роли, можно просто нажимать Enter) выводит в папку u_soft в файл с именем 001_Our_buffer.bin hex-коды содержащихся в Буфере данных; при этом данные в Буфере интерпретируются как тетрадно-десятичная запись байт-кодов.

Для написания собственных или редактирования имеющихся РП могут потребоваться исходные программы (ИП) на языке Fortran-5M. Они есть в файле ip_create_wav__frtrn5m.txt в папке u_soft. Сам Fortran-5M имеется в папке txt. О работе с ним, в том числе – как запускать Fortran-5M в имитаторе, загружать ИП, редактировать и компилировать ИП, превращая их в РП, – рассказано в "Руководстве к имитатору". Листинг ИП на экран дисплея выводится долго и быстро уходит из поля зрения, поэтому лучше выводить его в файл при включённом ТПУ; это делается фортранной командой S АЦПУ S Л с сопутствующими нажатиями клавиши Пробел.

Создание wav-файлов бейсик-программкой u_create_wav__bas.txt

Начальные шаги (1 – 4) – те же, что и выше; с тем лишь отличием, что вместо фортранной РП решаем применить программку u_create_wav__bas.txt под управлением Basic_D3-28_v3A__KP-157107__N-11343.txt, – этот файл с Бейсиком-157107 следует скопировать из папки txt в папку u_soft.

5) Командой **File** > **Open txt-file as Tape** открываем из папки u_soft файл с Бейсиком Basic_D3-28_v3A__KP-157107__N-11343.txt. Загружаем его в имитируемое ОЗУ Д3-28 на нулевой шаг и запускаем (как обычно – клавишами C, L, S в режиме **View** > **Machine**).

6) В режиме **View** > **Display** нажимаем F5 – это имитация включения ДУП-ЛИН для связи дисплея с Д3-28. Проводим начальный диалог с Бейсиком, отвечая на всё нажатием Enter. Через меню **File** закрываем файл с Бейсиком, открываем файл u_create_wav__bas.txt. Загружаем эту программку под управлением Бейсика командой LOAD, и запускаем командой RUN (о работе с Бейсиком в имитаторе рассказано в "Руководстве к имитатору"). На имитируемый экран дисплея программка выводит текст:

```
УПРАВЛЕНИЕ УСТРОЙСТВОМ СОЗДАНИЯ WAV-ФАЙЛОВ
(Устройство должно быть включенным)
```

Дальше ничего не происходит, если Устройство ещё не включено; надо включить Устройство и создать Буфер с данными согласно шагам 7 и 8 из предыдущего примера. То есть:

7) Нажимаем F1 (появляется изображение клавиатуры фрязинского терминала) и затем нажимаем U на своей клавиатуре ПК. В появившемся окошке изменяем состояние "Устройства" с OFF на ON, т. е. включаем Устройство для создания wav-файлов.

8) В этом же режиме (**View** > **Display** > F1) нажимаем Т $_{лат}$ и открываем из папки u_soft txt-файл 028-test_KP-73370_N-8921.txt – это выбранная нами программка, которую собираемся преобразовать в wav. Если всё идёт нормально, то окошко сообщит, что Буфер с txt-кодами создан, количество данных в нём = 8922. Нажимаем там ОК.

9) Возвращаемся к экрану дисплея нажатием F1. Программка для создания wav-файлов ещё не знает, что Буфер с данными уже есть, и поэтому пишет:

```
ОТКРОЙТЕ В "УСТРОЙСТВЕ" ФАЙЛ С БАЙТ-КОДАМИ ДЛЯ КОНВЕРСИИ В WAV
ГОТОВО?  ДА - 0, НЕТ - НЕ НОЛЬ
```

Нажимаем Enter (это то же самое, что ввод нуля).

10) Программка сообщает обнаруженное ею количество данных и начинает диалог о типе создаваемого wav-файла и о значениях параметров:

```
КОЛИЧЕСТВО БАЙТ-КОДОВ В БУФЕРЕ "УСТРОЙСТВА" =  8922
КАКОЙ WAV-ФАЙЛ БУДЕМ СОЗДАВАТЬ?  ВВОДИТЕ 0 ИЛИ НЕ НОЛЬ
1-КАНАЛЬНЫЙ - 0,  2-КАНАЛЬНЫЙ - НЕ НОЛЬ
```

В данном примере (как и в предыдущем) создаём файл типа "s-code из txt-программки" с параметрами "по умолчанию", поэтому на все предложения отвечаем нажатиями Enter:

```
S-CODE - 0,   M-CODE - НЕ НОЛЬ
С ДОБАВКАМИ ПО УМОЛЧАНИЮ - 0,   БЕЗ - НЕ НОЛЬ
```

```
ЗАДАЕМ ПАРАМЕТРЫ КОНВЕРСИИ БАЙТ-КОДОВ В WAV-ФАЙЛ

АМПЛ = 32000 ?  ДА - 0,  НЕТ - ВВЕДИТЕ ЖЕЛАЕМОЕ ЗНАЧЕНИЕ =

SAMPLE RATE Fd = 44100 ?  ДА - 0,  НЕТ - ЖЕЛАЕМОЕ ЗНАЧЕНИЕ =

ОТСЧЕТОВ В ПЕРЕПАДЕ dF =   23
ЕСЛИ ДА - ВВЕДИТЕ 0,  НЕТ - ЖЕЛАЕМОЕ ЗНАЧЕНИЕ =

ОТСЧЕТОВ МЕЖДУ ПЕРЕПАДАМИ dC =   65
ЕСЛИ ДА - ВВЕДИТЕ 0,  НЕТ - ЖЕЛАЕМОЕ ЗНАЧЕНИЕ =
```

Затем программка для проверки выводит на экран принятые значения параметров, вычисленные ею размер и длительность воспроизведения wav-файла, и ожидает согласия на продолжение:

```
ИТАК, БУДЕТ СОЗДАН WAV-ФАЙЛ С ПАРАМЕТРАМИ -

ЧАСТОТА ДИСКРЕТИЗАЦИИ Fd = 44100 Гц
КАНАЛОВ = 1,   РАЗРЯДНОСТЬ ОТСЧЕТА = 16 БИТ
ТИП СИГНАЛА =  S-CODE
dF =   23    dC =   65     dM =   176     Ampl = 32000
РАЗМЕР WAV-ФАЙЛА (БАЙТ, ПРИБЛИЗИТЕЛЬНО)   25498926
ДЛИТЕЛЬНОСТЬ,   МИНУТ =    4   СЕКУНД = 49

СОЗДАТЬ?  ДА - 0,  НОВЫЙ ВВОД ПАРАМЕТРОВ - НЕ НОЛЬ
```

11) Если всё устраивает, то нажимаем Enter. Тогда программка напишет:

```
ПАРАМЕТРЫ ПЕРЕДАЮТСЯ В "УСТРОЙСТВО"
ЖДЕМ ОТ НЕГО ОТВЕТ...
```

Затем, как и в *Примере* 1 раздела 3, появится окно с вопросом: "начать конвертировать txt-данные в s-code-wav?" Отвечаем "Да".

12) Откроется окно "Сохранить как"; там можно отредактировать предлагаемое по умолчанию имя wav-файла. Нажимаем в этом окне кнопку "Сохранить". Устройство создаёт wav-файл, помещает его в папку u_soft и сообщает о выполнении: Done, ok. Нажимаем ОК. На экране появляется завершающий текст:

```
WAV-ФАЙЛ СОЗДАН
ВЫХОДИМ ИЗ ПРОГРАММЫ.

ostanow w stroke7999
:
```

Программка u_create_wav__bas.txt закончила своё дело, но двоеточие в начале нижней строки означает, что Бейсик продолжает работать – он ожидает дальнейших команд и может выполнить ряд действий "в непосредственном режиме", без программки.

Одно из предусмотренных возможных действий – редактирование программки с целью создания сигналов, не предлагаемых в диалоге. В бейсик-программке наряду с теми же типами сигналов, которые создаются перечисленными выше фортранными РП (эти типы предлагаются в диалоге), есть подпрограммы для вычисления ступенчатых и экспоненциальных перепадов. Их можно задействовать, добавив специальную строку. А именно: для того чтобы перепады стали ступенчатыми (как в Примерах *6* и *7* из раздела 3, а также и в 2-канальных сигналах), надо в программку u_create_wav__bas.txt добавить строку (набираем её и нажимаем Enter):

```
4080 Z1=3
```

Для получения экспоненциальных перепадов (как в *Примере 5* из раздела 3, а также и в двухканальных сигналах) добавляем строку с тем же номером, но со значением Z1=4:

```
4080 Z1=4
```

При этом слова в диалоге не изменятся, так что, например, слова "sin-импульсы" надо будет понимать, соответственно, как "step-импульсы" или "exp-импульсы". Следует учитывать, что экспоненциальные перепады пригодны только в меандрообразных сигналах, т. е. в сигналах типа "m-code" и "2-канальный обычный Д3-28" (да и вообще: вряд ли exp-перепады будут чем-то полезнее обычных). Вот строки подпрограммы, вычисляющей экспоненциальный перепад:

```
550 REM П/П EXP-ПЕРЕПАД
555 LETU=-0.75*I
560 LETT=(2*EXP(U)-1)*D
565 RETURN
```

Здесь число **0.75**, влияющее на крутизну exp-перепада, можно изменять вручную, набирая заново строку 555.

До строки 999 (после неё начинается блок с перечислением переменных) имеется много свободного места. При желании пользователь может поместить в это место для экспериментов свою собственную подпрограмму вычисления перепада по формуле **T = D***f(**I**), где **D** = *Ampl*, **I** – переменная цикла, форма перепада задаётся пользовательской функцией f(**I**). Переход к своей подпрограмме следует поместить под строкой 550; например: **552 GOTO 600** где 600 – номер строки с началом подпрограммы (и тогда где-то до строки 999 необходимо разместить оператор **RETURN**); такая подпрограмма будет вызываться вместо подпрограммы exp-перепада. Если в формуле f(**I**) будут задействованы новые вспомогательные переменные, то их необходимо перечислить где-нибудь в строках 1000 – 1190, обязательно до переменной Z9. Переменная Z9 должна быть последней в блоке инициализации переменных.

Другое существенное действие, допустимое в непосредственном режиме, – управление Устройством с помощью команд 1, 2, 3 и 13, 14, 15, 16 (см. приведённые выше пояснения о работе с командами в фортрановских РП, а также описание команд в табл. 2 в разделе 3). При создании wav-файлов актуальны команды 15 и 16 – для перестановки каналов, изменения знаков начальных уровней ± *Ampl*, и для изменения длины концевых пауз в 2-канальных wav-файлах. В программке u_create_wav__bas.txt числовые значения выводятся в порты 0, 1, 2, 3, 4 Устройства из переменных с именами, соответственно, P0, P1, P2, P3 и P4 подпрограммами с начальными строками, соответственно, 100, 110, 120, 130 и 140.

Например, для вывода командой 15 в порт 3 значения *знаки* = 110 надо набрать и исполнить (т. е. завершить нажатием Enter) следующую ненумерованную строку

```
:P3=110:P4=15:GOSUB 130:GOSUB 140
```

Подобным же образом для установки концевых пауз с коэффициентом, например, 7 (т. е. $dP = 7 \cdot dm$) надо в порт 2 передать число 7 и подать Устройству команду 16:

```
:P2=7:P4=16:GOSUB 120:GOSUB 140
```

Важный нюанс: всё это сработает только в том случае, когда все переменные в программке u_create_wav__bas.txt инициализированы, т. е. когда программка хотя бы один раз запускалась оператором RUN. Для перехода к настройке Устройства до создания wav-файлов можно после запуска программки досрочно выйти из неё в непосредственный режим, нарочно вводя ошибочный символ, например, косую черту / , в ответ на первый же вопрос диалога.

В программке u_create_wav__bas.txt имеются также подпрограммы для ввода числовых значений параметров из портов 0 – 3 Устройства. Данные вводятся из портов 0, 1, 2, 3 Устройства в переменные с именами, соответственно, P0, P1, P2, P3 подпрограммами с начальными строками, соответственно, 200, 210, 220, 230. Это позволяет, в частности, командой 3 узнать текущие значения параметров dF и dC и затем командой 2 установить $dC = 0$. Такой приём используется для формирования "зуба пилы" с длиной, равной периоду

dm (в 2-канальном wav-файле с "saw-импульсами"), якобы как перепада, т. е. по алгоритму `T = D*`*f*`(I)`; см. строки 4500 – 4790 в листинге программы u_create_wav__bas.txt. Подобным же образом пользователь может запрограммировать вместо "зуба пилы" какую-нибудь свою произвольно сложную форму импульса, занимающую целиком период.

Листинг большой бейсик-программы удобно смотреть в текстовом файле, а не на имитируемом экране дисплея. Листинг выводится в текстовый файл в непосредственном режиме командой Бейсика-157107 LIST#1 при включённом ТПУ. Подробнее об этом рассказано в "Руководстве к имитатору".

Если отредактированную программку u_create_wav__bas.txt предполагается сохранить, то редактировать её (и затем сохранять бейсиковской командой SAVE в новый txt-файл, заранее открытый в меню **File**), следует под управлением Бейсика-132259, который загружается в имитатор из имеющегося в папке txt файла Basic_D3-28_v3A__KP-132259__N-9675.txt. В этом случае шрифт в диалоге не изменит своего нормального вида; такова особенность имитатора Однако, к сожалению, в Бейсике-132259 нет команды вывода листинга в текстовый файл.

5. Загрузка блочных программ на примере ОС ЛФТИ 32к и ВТ МХТИ 128С

Общий способ загрузки через разъём "Ввод-вывод" Д3-28, который был бы пригоден для всех блочных программ (записанных командой Д3-28 SAVER в виде 256-байтных блоков), ещё не придуман. Однако для каждого частного случая можно при желании и усердии подобрать индивидуальный алгоритм загрузки с последующим блочным копированием из ОЗУ на МЛ в НМЛ Д3-28. Поиск таких алгоритмов может быть основан на анализе работы имеющихся ("штатных") загрузчиков и копировщиков ("генераторов системы"), которыми были снабжены некоторые программы для Д3-28, записанные на МЛ в блочном формате.

Вот один из вариантов. По дизассемблированному коду штатного загрузчика вместе с дампом ОЗУ после загрузки ПО в имитаторе можно узнать адреса и способ размещения в ОЗУ данных из блоков этого ПО, расположение копировщика в ОЗУ, выяснить, где в ОЗУ остаётся свободное место (для нашего загрузчика м-кода), а также узнать, как инициализируются служебные регистры и запускается штатное копирование. Затем все эти данные мы оформляем в виде одного txt-файла (или двух, если надо), конвертируем в wav, загружаем через разъём "Ввод-вывод" в реальную Д3-28 и запускаем там копирование на реальную МЛ. В дальнейшем, после штатной загрузки через НМЛ с этой МЛ можно будет запускать данное ПО в реальной Д3-28 на исполнение, и, в том числе, – штатно запускать его копирование на новые МЛ. Ниже этот подход разбираю на двух примерах (проверенных в имитаторе).

ОС ВТ-МХТИ 32к, версия ЛФТИ

О штатной загрузке с МЛ и о работе с *ОС ВТ-МХТИ 32к в версии ЛФТИ* рассказано в разделе 5 "Руководства к имитатору".

Штатный загрузчик этой системы, состоящей из двух блочных частей, сам состоит из двух частей. 1-я часть загрузчика (файл os32k_zagr-1_kp-1168_n-79.txt в папке u_soft для наших целей) при штатной работе загружается на шаг 0, проверяет наличие включённого фрязинского дисплея, загружает на шаг 03280 вторую часть загрузчика, передаёт ей управление, и в конечном итоге 1-я часть загрузчика затирается 1-й частью системы, загружаемой на шаг 0. Эта первая часть загрузчика, в отличие от второй его части, не содержится в частях системы и, в результате, не может быть автоматически скопирована на новую МЛ.

Поэтому при штатном копировании надо 1-ю часть загрузчика прочитать с исходной МЛ и, не запуская, записать несколько раз подряд на новую МЛ вручную (нажатиями кнопки ЗЛ на пульте Д3-28). Только после этого следует запустить загрузку системы с исходной МЛ, а затем опять сменить МЛ и разрешить системе копироваться автоматически, когда на дисплее появится предложение задать количество дублей блоков:

```
КОПИЯ:ДУБЛЕЙ-
```

1-я часть системы, записанная в 13 блоках, занимает адреса 0 – 03535, вторая часть системы, записанная в 41 блоке, занимает адреса от 22104 (в машине с 32-килобайтным адресным пространством) и почти до конца рабочей зоны ОЗУ. Дизассемблирование загрузчика показало, что система перед загрузкой определяет объём ОЗУ и может настроиться на загрузку и работу также в 16-килобайтной машине; но в имитаторе нет имитации 16-килобайтной Д3-28, поэтому я не додумал до конца 16-килобайтный вариант и далее не пишу о нём.

Между 1-й и 2-й частями системы много свободного места, так что можно помещать наш загрузчик м-кода, например, на шаг 06000. Дампы ОЗУ с 1-й и 2-й частями системы, загруженной в имитаторе, я сделал программкой print_column__KP-782_N-57.txt, – она выводит в txt-файл столбиком байт-коды из участка ОЗУ, начало и конец которого задаются вводом десятичных адресов в X-регистр. В конец дампа 1-части добавил около 30 байт с командами такой же инициализации регистров, какая должна выполняться при штатной загрузке системы; начало этой инициализации – на шаге 03540, заканчивается она запуском копирования. Оба дампа находятся в папке u_soft, это файлы os32k_p1_3565.txt и os32k_p2_10336.txt. Последние числа в названиях этих файлов указывают количество содержащихся в них байт-кодов.

На шаге 03560 в os32k_p1_3565.txt командой 1301 0003 в регистр S1 заносится число 3 – это установленное по умолчанию число дублей при копировании блоков. Можно заменять его на 2 или 1 в разных попытках копирования. Чем оно меньше, тем меньше времени и места займёт запись на МЛ, но и тем менее надёжным будет считывания такой записи.

0) Для загрузки в Д3-28 через разъём "Ввод-вывод" методом м-кода (в форме "s-code" – этот вариант метода м-кода далее подразумеваю) надо преобразовать файлы os32k_p1_3565.txt и os32k_p2_10336.txt в wav-файлы типа s-code из txt-программки", как пояснялось в предыдущих разделах. Программку os32k_zagr-1_kp-1168_n-79.txt можно тоже конвертировать в wav, чтобы не набирать её вручную. Дальнейший план действий таков:

1) Подсоединяем ИУ к машине Д3-28 и включаем машину. Или "освежаем" её ресетом. На реальной Д3-28 ресет представляет собой выключение и затем включение блока питания машины выключателем, расположенным на её задней стенке. В зависимости от типа ИУ возможны нюансы насчёт линии СИП; см. в разделе 6 пояснения о подготовке аппаратуры.

2) С пульта Д3-28 в режиме ввода набираем кнопками прямого кодирования 139 байт нашего загрузчика м-кода loader_m-code_addr-x_kp-1812_n-138.txt на начальный шаг 06000.

3) Загружаем в Д3-28 методом м-кода на шаг 0 наш wav-файл с программкой-загрузчиком os32k_zagr-1_kp-1168_n-79. (Другой вариант – набрать её вручную с пульта машины; она небольшая, в ней всего 80 байт.)

4) В НМЛ Д3-28 на новую МЛ записываем командой ЗЛ несколько копий, например 3, этой программки-загрузчика. МЛ оставляем в машине – не перематываем её, не вынимаем.

5) Загружаем в Д3-28 методом м-кода на шаг 0 наш wav-файл с данными os32k_p1_3565.

6) Загружаем в Д3-28 методом м-кода на шаг 22104 наш wav-файл с данными os32k_p2_10336. (Адрес загрузки задаётся нашему loader_m-code_addr-x_kp-1812_n-138.txt в X-регистре перед его запуском; работа с ним подробнее пояснена в разделе 6.)

7) В режиме ввода на пульте машины устанавливаем шаг 03540 и запускаем на исполнение с этого адреса в режиме "Работа" кнопкой S имеющийся теперь в ОЗУ машины код. Должна начаться запись блоков системы на МЛ.

8) После того как все 54 блока будут записаны на МЛ, на пульте машины загорится светодиод "ожидание ответа ПУ" (СИП). При наличии подключенного дисплея это соответствовало бы выводу на экран предложения повторять копирование: **КОПИЯ:ДУБЛЕЙ-** и ожиданию ответа пользователя с клавиатуры дисплея; при ответе 0 можно было бы работать в уже загруженной системе. Но наша задача заключалась только в записи системы на МЛ, она выполнена, так что можно просто нажать С, чтобы остановить работу машины по программе.

Этот метод загрузки *ОС ВТ-МХТИ 32к в версии ЛФТИ* в Д3-28 и записи её на МЛ не требует наличия дисплейного терминала. Но для проверки читаемости получившейся записи дисплей будет нужен, так как штатный загрузчик этой системы ожидает отклик терминала на команды его активации и выводит на экран информацию о загрузке блоков, соответственно ожидая от терминала ответных синхроимпульсов (СИП).

ОС ВТ МХТИ 128С

Уважаемый Виталий Васильевич Колесник любезно поделился информацией из письма своего бывшего коллеги о штатной загрузке системы *ОС ВТ МХТИ-128С* с МЛ, включающей режим с копированием (с "генерацией системы") на МЛ:

. Чтобы загрузить и запустить на исполнение систему *ВТ МХТИ-128С* надо считать с МЛ (нажатием кнопки СЛ) загрузчик с контрольной суммой 5227 и нажать кнопку S. При правильном чтении очередного нужного блока загорается верхний красный светодиод, при чтении запасного (ненужного) блока – средний светодиод, при неправильном – нижний. После чтения лента перематывается назад.

. Чтобы загрузить систему в режиме с последующей генерацией, надо вместо S нажать **> >** (т. е. после чтения загрузчика с контрольной суммой 5227 два раза нажать кнопку "треугольник"). Когда машина остановится с индикацией цифры 1 на Y-табло, надо поставить в НМЛ чистую МЛ, ввести в X-регистр желаемое число дублей при копировании блоков, и нажать S. Выполнив запись, система снова так же останавливается, предлагая тем самым снова делать копирование. Разумно первую копию (МЛ для работы) сделать с 2-кратным дублированием, а вторую копию (резервную МЛ) – с 3-кратным. Для выхода из режима генерации в режим работы – нажать **>** и затем СК.

Оказывается эта система при копировании автоматически записывает 9 копий своего загрузчика (КП = 5227, номер последнего шага = 362), так что, в отличие от предыдущего примера, здесь не придётся записывать загрузчик на новую МЛ вручную.

Дизассемблирование загрузчика и дампов ОЗУ с загруженной системой в имитаторе показало, что эта система, как и предыдущая, состоит из двух частей. 1-я часть содержится на МЛ в 22 256-байтных блоках, 2-я часть – в 104 блоках. Загрузчик выполняет наряду с инициализацией регистров некоторые переключения физических сегментов ОЗУ, переписывает служебную зону в другой сегмент, так что загрузка системы и её работа осуществляется в 128-килобайтной машине в сегментах ОЗУ, часть из которых отсутствует в 32-килобайтной машине. Но этот факт не препятствует нашим целям – адресное пространство программы остаётся 32-килобайтным, поэтому для загрузки методом м-кода с целью копирования системы на МЛ той

же программой можно пользоваться исходными сегментами (в том числе можно пользоваться 32-килобайтными машинами).

При этом в доступных нам 32 килобайтах остаётся мало свободного места между 1-й и 2-й частями системы: свободны байты с адресами от 05913 до 06167. К счастью, этого достаточно для дополнения дампа 1-й части кодами инициализации регистров и для размещения нашего загрузчика м-кода loader_m-code_addr-x_kp-1812_n-138.txt – его удобно поместить, как и в предыдущем примере, на начальный адрес 06000.

0) Дампы обеих частей системы с необходимым дополнением находятся в папке u_soft, это файлы mhti128c_p1_5937.txt и mhti128c_p2_26210.txt. Последние числа в названиях этих файлов указывают количество содержащихся в них байт-кодов. Для загрузки в Д3-28 через разъём "Ввод-вывод" методом м-кода (в форме "s-code") надо преобразовать файлы mhti128c_p1_5937.txt и mhti128c_p2_26210.txt в wav-файлы типа s-code из txt-программки", как пояснялось в предыдущих разделах. Дальнейший план действий таков:

1) Подсоединяем ИУ к машине Д3-28 и включаем её (или "освежаем" её ресетом – включаем и выключаем блок питания выключателем на задней стенке машины). В зависимости от конкретной схемы ИУ возможны нюансы насчёт линии СИП; см. в разделе 6 пояснения о подготовке аппаратуры.

2) С пульта Д3-28 в режиме ввода набираем кнопками прямого кодирования 139 байт нашего загрузчика м-кода loader_m-code_addr-x_kp-1812_n-138.txt на начальный шаг 06000. Если есть его запись на МЛ, то, конечно, можно и с МЛ его загрузить: в режиме ввода после нажатия кнопки НШ набираем шаг 06000, затем в режиме "Работа" нажимаем кнопку СЛ.

3) Загружаем в Д3-28 методом м-кода на шаг 0 наш wav-файл с данными из mhti128c_p1_5937.

4) Загружаем в Д3-28 методом м-кода на шаг 06168 наш wav-файл с данными из mhti128c_p2_26210. Адрес загрузки задаётся нашему loader_m-code_addr-x_kp-1812_n-138.txt в X-регистре перед его запуском.

5) Ставим в НМЛ кассету с чистой МЛ.

6) В режиме ввода на пульте машины устанавливаем шаг 05917 и запускаем на исполнение с этого адреса в режиме "Работа" кнопкой S имеющийся теперь в ОЗУ машины код. На Y-табло появится цифра 1, и машина остановится – это приглашение ввести в X-регистр желаемое число дублей при копировании блоков. Вводим, например, 2 и нажимаем кнопку S. На МЛ должно автоматически записаться 9 копий загрузчика-копировщика (КП = 5227, N = 362), затем 22 блока 1-й части системы и 104 блока 2-й части.

7) После того как все 126 блоков будут записаны на МЛ, снова на Y-табло появится цифра 1, и машина остановится, приглашая тем самым ввести в X-регистр желаемое число дублей для нового копирования блоков. Можно поставить в НМЛ новую МЛ и повторить копирование, введя в регистр X число дублей, например, 3. Ну и можно продолжать плодить записи в том же духе, не забывая однако давать магнитофону в Д3-28 отдохнуть!

Запустить систему на исполнение таким методом не удастся – не были должным образом переключены сегменты ОЗУ и не подключался дисплейный терминал. Пробовать её запускать следует на 128-килобайтной Д3-28 с подключенным 15ИЭ-00-013 после штатной загрузки системы с МЛ. Проверить читаемость системы с МЛ можно и без дисплейного терминала, просто выполняя штатную загрузку, как рассказано в начале.

6. Загрузчик m-code. Тестирование wav-файлов с m-code и s-code в имитаторе

Подробное описание программки loader_m-code_addr-x_kp-1812_n-138.txt:

Номер шага	Байт-код команды	Мнемоника команды	Содержание команды	Время выполнения
00000	1104 1507	MOV R15, R7	Адрес из прогр. счётчика R15 = PC → в 2-байтовый регистр R7. В R15 старший бит = ПрП = 1 (это признак того, что машина работает по программе), поэтому число в R15 является отрицательным. Это число = с минусом адресу второго байта данной команды, потому что PC к моменту выполнения 2-байтовой команды содержит адрес её второго байта .	51 т (число тактов)
00002	0413 0807	ABS R7	| R7 | → R7 т.е. регистр R7 теперь содержит адрес второго байта программы в виде положительного числа.	28 т
00004	1001 0107	SUB #01, R7	R7 – 1 → R7 т.е. R7 теперь содержит адрес первого байта данной программы-загрузчика.	33 т
00006	0413 0607	MOV R7,BP	R7 → BP : число из R7 занесли в регистр «базового адреса программы». Это нужно потому, что в дальнейшем будет по метке вызываться подпрограмма, а машина ищет метку подпрограммы, начиная с адреса, указанного в BP. С нулевого адреса может начинаться загружаемый файл, и в нём может встретиться такая же метка; поэтому нужно, чтобы при работе загрузчика машина не искала бы подпрограмму загрузчика с нулевого адреса. Кроме того, на поиск метки подпрограммы уходит тем больше времени, чем больше байт машина просматривает перед меткой. Чтобы уменьшить это время, мы должны расположить метку подпрограммы как можно ближе к адресу в BP, указывающему начало нашего загрузчика.	37 т
00008	1403 0414	BR 00087	Безусловный переход на 4·16+14·1+1=79 шагов вперёд, (т. е. к шагу 00087 при отсчёте начала программы от нулевого шага). Это обход всего кода с подпрограммой.	20 т
00010	0408 0202	MARK 0202	Метка «0202» нашей подпрограммы ввода байта с линии Ввa1 (Ввa1 – это младший бит байтовой шины «Ввод»). Ниже идёт собственно байт-код подпрограммы.	23 т
00012	1309 0008	MOV #0008,S9	Число 8 → в 1-байтовый регистр S9 для счёта бит в байте, когда синхробайт уже не надо будет искать (см. ниже).	26 т
00014	1301 0000	MOV #0000,S1	Число 0 → в однобайтовый регистр S1: это очистка накопителя байта. В регистре S1 будет накапливаться загружаемый байт.	26 т
00016	1400 1014	INPO 1014	При УПР = 1014 ввод байта с шины «Ввод» в однобайтовую ячейку с адресом BD + число из регистра R10. У нас на стадии инициализации (см. ниже) установлено значение BD = 0, и число в регистре R10 = адресу регистра S3. Так что, здесь и ниже команда INPO принимает 1 байт с шины «Ввод» в регистр S3.	54+τ т (τ – время ответа (СИП))
00018	1108 0003	AND S0, S3	S0 AND S3 → S3: это очистка в ноль всех бит регистра S3, кроме младшего бита, в котором находится принятый с шины «Ввод» бит. Регистр S0 хранит константу 0001, именно для такой очистки ненужных бит в регистре S3.	37 т

00020	1112 0307	MOV S3, S7	S3 → S7: сохранили S3 в S7 для сравнений с ним принимаемого сигнала в ходе поиска перепада с битом информации.	37 т
00022 00024	1400 1014 1108 0003	INPO 1014 AND S0, S3	Ещё раз INPO: ввод байта с шины «Ввод» в регистр S3. S0 AND S3 → S3: это очистка в ноль ненужных бит в S3.	54+τ т 37 т
00026	1115 0307	BSA S3, S7	Если S3 = S7, то идём к .+4 (точка означает адрес первого байта данной команды, т. е. идём к шагу 00030). А иначе идём к следующей команде:	37 т
00028	1403 0003	BR 00032	Переход на 0·16+3·1+1 = 4 шага вперёд (к шагу 00032).	20 т
00030	1402 0009	BR 00022	Переход на 0·16+9·1–1 = 8 адресов назад, т. е. снова к INPO. Идея такая: в регистре S7 хранится результат ввода младшего бита «Ввод» в регистр S3 на стадии поиска перепада в принимаемом сигнале. Если перепад не обнаружен (т. е. если очередное принятое значение S3 совпадает с предыдущим, хранящимся в S7), то возвращаемся назад – снова принимать сигнал. А если перепад обнаружен, то принятый бит будем заносить в наш накопитель байта – в регистр S1:	22 т
00032	1302 0000	MOV #0000,S2	0 → S2: очистили старший байт 2-байтового регистра R9.	26 т
00034	1114 0103	SWA S1, S3	Сохранили S3 в S1, а то, что накопилось в S1, поместили в S3, т. е. в младший байт регистра R9.	39 т
00036	1102 0209	MUL R2, R9	R9·R2 → R9: умножение R9 на константу 2, хранящуюся в R2. Тем самым содержимое R9 сдвигается на один бит влево, а младший бит в R9 (и он же в S3) принимает значение 0.	420 т
00038	1114 0103	SWA S1, S3	Сдвинутые накопленные биты с нулевым младшим битом вернулись в S1, а из S1 вернулся в S3 «перепад». Принятый бит находится в младшем двоичном разряде регистра S3.	39 т
00040	1111 0301	OR S3, S1	S1 OR S3 → S1: принятый бит добавляется в младший разряд накопителя байта S1. <u>Следующие 4 команды образуют цикл «задержки на 0.75·T»</u>. Он нужен для того, чтобы игнорировать неинформативный стык соседних периодов; T – длительность периода сигнала с м-кодом.	38 т
00042 00044	1302 0000 1303 0001	MOV #0000,S2 MOV #0001,S3	0 → S2: очистили старший байт 2-байтового регистра R9. 1 → S3: занесли «константу чтения = 1» в младший байт регистра R9 и тем самым 1 → R9. Значение «константы чтения» 1 даст минимальную задержку. Предположительно, такой выбор подойдёт для «медленных» машин (у которых время машинного такта = 2 мкс), а для «быстрых» машин (у которых время такта = 1 мкс) надо, предположительно, вместо 1 задавать константу 23 (командой 1303 0107 на шаге 44). В общем, это то место загрузчика с которым, вероятно, придётся поэкспериментировать.	26 т 26 т
00046	1408 0309	SOBZ R9 00050	R9 – 1 → R9; если стало R9 = 0, то идём к .+4 то есть к шагу 00050, это будет <u>конец задержки</u>.	39 т
00048	1402 0003	BR 00046	А иначе переходим на 3 – 1 = 2 шага назад, т. е. продолжаем вычитать 1 из R9.	22 т

00050	1115 0800	BSA S8, S0	Если S8 = S0, то идём к .+4. Равенство S8 = S0 у нас служит признаком того, что синхробайт ещё не найден. А иначе:	37 т
00052	1403 0103	BR 00072	Переход к шагу 00072. Это обход поиска синхробайта.	20 т
00054	1008 1406	BNS #1406,S1	Если S1 ≠1406, то идём к .+4 проверять, не принят ли инвертированный синхробайт 1406, т. е. байт 0109.	28 т
00056	1403 0009	BR 00066	А иначе идём к шагу 00066 с принятым синхробайтом 1406.	20 т
00058	1008 0109	BNS #0109,S1	Если S1 ≠0109, то идём к к .+4 и затем к INPO в начале п/п, чтобы искать перепад и принимать следующий бит.	28 т
00060	1403 0003	BR 00064	А иначе идём к шагу 00064 с принятым синхробайтом 0109.	20 т
00062	1402 0215	BR 00016	Переход к команде INPO в начале п/п, чтобы искать перепад и принимать следующий бит.	22 т
00064	1306 1515	MOV #1515,S6	1515 → S6: занесли в S6 признак инверсии синхробайта. Он будет использоваться для инверсии принимаемого сигнала.	26 т
00066	1308 0000	MOV #0000,S8	0 → S8: убрали в S8 признак необходимости искать синхробайт. Одновременно это есть очистка старшего байта в двухбайтовом регистре R12. Этот регистр будет использоваться как счётчик битов в принимаемом «информационном» байте (т.е. уже не в синхробайте).	26 т
00068	1309 0009	MOV #0009,S9	Подготавливаем начальное значение счтчика битов R12: 9 → S9: заносим в младший байт регистра R12 число 9.	26 т
00070	1301 0000	MOV #0000,S1	0 → S1: это очистка накопителя байта от синхробайта перед приёмом информационного байта.	26 т
			Когда признак необходимости поиска синхробайта уже убран, то поиск синхробайта обходится, и управление от шага 00052 (т. е. после приёма бита) передаётся сюда:	
00072	1408 0312	SOBZ R12 76	R12 – 1 → R12; если R12 = 0, то идём к .+4, а иначе к .+2 Это счёт бит в принимаемом байте.	39 т
00074	1403 0004	BR 00079	Пока не будет принято 8 бит (т. е. пока остаётся R12 > 0), надо возвращаться к INPO в начало п/п, чтобы искать перепад и принимать следующий бит. Перед этим, однако, надо сделать дополнительную задержку, чтобы хоть немного выровнять паузу перед INPO при накоплении байта с паузой, возникающей при обработке накопленного байта.	22 т
00076	1109 0601	XOR S6, S1	Когда принят 8-й бит, то: S6 XOR S1 → S1: это инверсия принятого байта S1, если S6 = 1515 (вследствие инвертированности синхробайта). Либо здесь S1 не изменяется, если S6 = 0 (это значение S6 задаётся при инициализации и оно не изменяется, если был принят не инвертированный синхробайт).	37 т
00078	0511	RTS	Возврат из подпрограммы. Принятый информационный байт находится в регистре S1.	50 т
			Выравнивающая задержка:	
00079	1303 0002	MOV #0002,S2	2 → S3: стало R9 = 2.	26 т
00081	1408 0309	SOBZ R9 85	R9 – 1 → R9; при R9 = 0 переход к .+3+1.	39 т
00083	1402 0003	BR 00081	А иначе идём ещё раз к предыдущей команде.	22 т
00085	1402 0406	BR 00016	Задержка выполнена, возвращаемся к INPO.	22 т

			При запуске программы устанавливается BP, и затем приходим сюда, здесь – инициализация служебных ячеек памяти:	
00087	0413 1011	CLR R11	0 → R11: очистка двухбайтового регистра R11; он состоит из однобайтовых регистров S6 и S7.	33 т
00089	0413 0511	MOV R11,BD	R11 → BD: занесли значение 0 в «базовый регистр данных» BD.	37 т
00091	0413 1200	MOVH X, R0	Целая часть $(X)_{16}$ → R0: занесли значение X в регистр R0. Это у нас будет начальный адрес загрузки файла.	170 т
00093	1000 0211	ADD #02, R11	R11 + 2 → R11: получили в R11 число 2.	33 т
00095	1104 1102	MOV R11, R2	R11 → R2: занесли в регистр R2 константу 2.	51 т
00097	1304 0715	MOV #0715,S4	0715 → S4	26 т
00099	1305 0103	MOV #0103,S5	0103 → S5: тем самым в 2-байтовый регистр R10, состоящий из S4, S5, занесли адрес $(0715\ 0103)_{16} = 32531_{10}$ регистра S3.	26 т
00101	1310 0715	MOV #0715,S10	Вершину стека сдвинули в самый низ рабочей зоны ОЗУ.	26 т
00103	1300 0001	MOV #0001,S0	0001 → S0: занесли в регистр S0 константу 1.	26 т
00105	1308 0001	MOV #0001,S8	0001 → S8: занесли в S8 признак необходимости поиска синхробайта.	26 т
00107	0202	JSM 0202	Первый раз вызвали подпрограмму загрузки байта. Она обнаруживает синхробайт, сбрасывает в ноль признак необходимости поиска синхробайта, и загружает в S1 первый информационный байт из сигналов файла в линии «Ввод». Это должен быть старший байт двухбайтового числа (минус 1) байт, подлежащих загрузке из файла.	58+4·k = 98 т, k = 10 шагов до MARK
00108	1112 0102	MOV S1, S2	S1 → S2: занесли его в старший байт регистра R9, состоящего из однобайтовых регистров S2 и S3.	36 т
00110	1303 0001	MOV #0001,S3	0 → S3: занесли 0 в младший байт регистра R9.	26 т
00112	1104 0901	MOV R9, R1	R9 → R1: записали содержимое R9 в регистр R1, он будет у нас счётчиком принятых байт.	51 т
00114	0202	JSM 0202	Второй раз вызвали п/п загрузки байта. Теперь она уже не ищет синхробайт, а загружает в S1 из файла второй информационный байт – это должен быть младший байт «двухбайтового числа (минус 1) байт, подлежащих загрузке из файла».	58+4·k = 98 т
00115	1112 0103	MOV S1, S3	S1 → S3: занесли этот байт в мл. байт регистра R9.	36 т
00117	1302 0000	MOV #0000,S2	0 → S2: занесли 0 в старший байт регистра R9.	26 т
00119	1100 0901	ADD R9, R1	R9 + R1 → R1: получили в R1 «двухбайтовое число (минус 1) байт, подлежащих загрузке из файла».	62 т
			Начало цикла загрузки в ОЗУ программы из файла:	
00121	0202	JSM 0202	Вызов п/п загрузки байта из файла в S1.	58+4·k = 98 т
00122	0912 0100	MOV S1, @R0	Байт из S1 → в 1-байтовую ячейку ОЗУ с адресом в R0.	47 т
00124	1000 0100	ADD #01,R0	R0 + 1 → R0: нарастили в R0 адрес загрузки.	33 т
00126	1408 0301	SOBZ R1 00130	R1 – 1 → R1; при R1 = 0 переход к .+3+1 = 00130,	39 т
00128	1402 0008	BR 00121	а иначе – к началу цикла, принимать очередной байт.	22 т
00130	0413 1009	CLR R9	Конец цикла. Очистили R9: занесли в него нуль.	33 т
00132	0413 0609	MOV R9,BP	R9 → BP: восстановили нулевое значение в BP.	37 т

00134 00136 00138	0412 1210 1402 0003 0512	SMER BR 00134 END	Зажгли светодиод ОМ для индикации окончания загрузки и зацикливаемся (чтобы машина не остановилась). Признак конца (для записи и чтения загрузчика в НМЛ).	29 т 22 т

Пояснения о работе загрузчика м-кода

В общих чертах принцип работы такого загрузчика мы рассмотрели в разделе 1. Про сигнал, условно называемый здесь "м-кодом" (более точные термины – фазовая мнипуляция, двухфазное кодирование, манчестерский код), подробно рассказывалось в журнале «Радио» №9 1983 г. стр. 32 – 35 в статье о подключении кассетного магнитофона к ЭВМ "Микро-80" с названием "Модуль сопряжения"; авторы: Г. Зеленко, В. Панов, С. Попов. См. по ссылке (там же сканы остальных страниц статьи): http://archive.radio.ru/web/1983/09/035/

Динамику работы загрузчика поясняет рисунок-"осциллограмма" из указанной статьи. "Период Т" это интервал времени, который у нас в разделе 1 обозначался как *dM* / *Fd*. Каждый бит «0» представлен в сигнале перепадом уровня сигнала от высокого к низкому; каждый бит «1» представлен перепадом от низкого к высокому уровню. «Периодом» называем интервал времени, в середине которого присутствует перепад, представляющий «0» или «1». Перепады на стыках периодов нас не интересуют; такие перепады могут и отсутствовать:

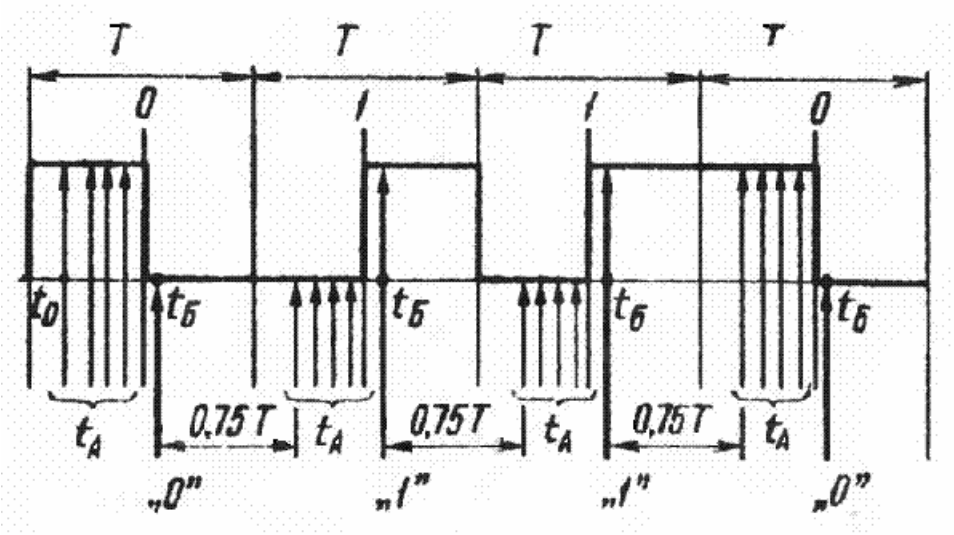


Рис. 30. Процесс чтения сигнала с м-кодом. Изображены биты 0, 1, 1. 0.

С какого-то момента времени t_0 программа-загрузчик начинает следить за уровнем сигнала, поступающего в ЭВМ из интерфейсного устройства, и сравнивает друг с другом принятые ею подряд значения уровня, чтобы обнаружить перепад. Моменты времени, в которых перепад ещё не обнаружился, обозначены на рис. 30 как t_A. Чтобы таких моментов до прихода перепада имелось не меньше двух, загрузчик должен успевать достаточно часто считывать значения уровня. После обнаружения перепада (на рис. 30 это моменты времени $t_Б$) очередной приём уровня сигнала задерживается на 0.75·Т, чтобы заведомо проигнорировать не содержащий информации о битах перепад на стыке соседних периодов, если он там есть.

Из приведённого выше кода загрузчика loader_m-code_addr-x_kp-1812_n-138.txt для ЭВМ Д3-28 можно видеть, что повторные попытки обнаружения перепада (см. шаги 16 – 30) следуют с интервалом порядка 150 тактов. Сколько это в долях секунды – зависит от Д3-28; эти машины бывают "быстрые" (1 такт ≈ 1 мкс) и "медленные" (1 такт ≈ 2 мкс). Вот информация о быстродействии разных исполнений Д3-28 из документа И5М3.857.100 ТО:

Исполнение	Сокращенное обозначение	Время выполнения микрокоманд, мкс	Объем ОЗУ, Кбайт
И5М3.857.100	15ВМ16–017	2	16
–01	15ВМ32–016	1	32
–02	15ВМ32–021	2	32
–03	15ВМ128–019	1	128
–04	15ВМ32–020	2	32
–05	15ВМ128–018	1	128

Делать паузу «0.75·T» с помощью вызова специальной подпрограммы задержки в нашем случае смысла нет, потому что в машине Д3-28 и без того между актами приёма бита будет возникать пауза из-за обработки принятого бита, и особенно – из-за обработки байта, накопленного из принятых бит.

Дело в том, что в Д3-28 нет команды сдвига бит в однобайтовой ячейке. Вместо «сдвига влево» приходится применять «умножение на 2», причём в Д3-28 оно реализуется как перемножение 2-байтовых регистров, и уже одна эта операция длится 420 тактов. В паузу входит также время возврата из подпрограммы приёма байта, время занесения байта на его место в ОЗУ, и время, занимаемое очередным вызовом п/п приёма байта. Длительность такого рода неизбежной паузы составляет порядка 1500 тактов – это время и служит оценкой для «0.75·T». Для «плавной настройки» загрузчика под скорость потока бит предусмотрены шаги 42 – 48 и 79 – 85, позволяющие увеличивать паузу (и частично выравнивать её длительность при накоплении байта и обработке байта) порциями в 61 такт.

В медленной машине 1500 тактов для задержки «0.75·T» длятся 1500·2 мкс = 3000 мкс, так что T = 4000 мкс = 0.004 сек. Для быстрой машины это время можно было бы уменьшить вдвое, но, возможно, будет удобнее на конкретной машине подстраивать программу-загрузчик (подбором «константы чтения» на шаге 00044) под готовый сигнал, чем изготавливать отдельные wav-файлы с разной скоростью передачи сигнала для разных машин. (Впрочем, наше "Устройство", синтезирующее wav-файлы, позволяет легко регулировать скорость передачи сигнала – выбором частоты дискретизации *Fd* и количества сэмплов *dM* в периоде.)

Ввод загрузчика в ОЗУ Д3-28:

До того, как загрузчик удастся сохранить на МЛ, его надо будет вводить в ОЗУ Д3-28 вручную – кнопками прямого кодирования на пульте машины. Для загрузки большой программки "одним куском" начальный адрес набора загрузчика следует выбирать ближе к концу рабочей зоны ОЗУ, так как в начало рабочей зоны загрузчик будет загружать программку. Например, можно выбрать 30000 в качестве начального шага загрузчика. Рабочей зоны ОЗУ перед ним хватит с запасом, поскольку размер имеющихся у нас программок, пригодных к загрузке "одним куском" (т. е. не блочных), не превышает 20 килобайт.

Для набора загрузчика надо перевести машину в режим ввода кнопкой «В», нажать «НШ», и цифровыми кнопками ввести 5-значный номер шага – в данном примере это 30000; он индицируется на X-табло машины, а правее него будет видно (в тетрадно-десятичной форме) содержимое байтовой ячейки ОЗУ с этим адресом. Затем кнопками прямого кодирования набираем в два приёма каждый байт загрузчика – сначала в коде 8-4-2-1 кнопками «80», «40», «20», «10» под светодиодами набираем старшую тетраду байта, после чего одной из кнопок «00», «01», … «15» вводим младшую тетраду байта. При этом номер шага автоматически увеличивается на единицу. Для просмотра ОЗУ байт за байтом и для установки адреса можно нажимать кнопки «Ш» (шаг вперёд) и «ШН» (шаг назад).

Для проверки контрольной суммы набранного загрузчика надо перевести машину в режим «Работа» кнопкой «Р», цифровыми кнопками ввести значение X, равное начальному адресу (в нашем примере 30000), и затем подать команду 1201 (т. е. сначала в коде 8-4-2-1 кнопками под светодиодами набрать тетраду 12, затем нажать кнопку «01»). Контрольная сумма изобразится на X-табло.

Правильно набранный загрузчик следует сохранить на МЛ. Для этого в режиме «В» задаём, как указывалось выше, начальный номер шага (в нашем примере 30000), переходим в режим «Р» и нажимаем кнопку ЗЛ. Есть и другой способ такой же записи: в режиме «Р» вводим в X начальный адрес (в нашем примере 30000) и кнопками прямого кодирования подаём команду 1203. Следует повторить сохранение несколько раз – чтобы записалось несколько копий.

Загрузка загрузчика с МЛ: в режиме «В» устанавливаем начальный адрес (в нашем примере 30000), и затем в режиме «Р» нажимаем кнопку «СЛ».

Для работы с блочными программами начальный адрес загрузчика выбирается в зависимости от количества и размеров подлежащих загрузке частей. Так, в двух примерах из предыдущего раздела начальным адресом загрузчика являлся шаг 06000.

Запуск загрузчика:

Перед запуском загрузчика важно не упустить из вида два момента – установить начальный адрес для загружаемой программки и привести в должное состояние аппаратуру.

Загрузчик loader_m-code_addr-x_kp-1812_n-138.txt загружает программку на начальный адрес, который перед запуском загрузчика установлен (в виде десятичного числа) в регистре X; об этом напоминает метка "addr-x" в названии загрузчика. Как правило, загружаемые "одним куском" программки вводятся на начальный адрес 0. Для установки 0 $\longrightarrow$ X достаточно нажать на пульте машины кнопку С. Если же нужен другой адрес загрузки (например, как при загрузке "вторых частей" блочных программ, о которых шла речь в предыдущем разделе), то просто надо его набрать в регистре X цифровыми кнопками пульта машины. Всё это – в режиме "Р".

Подготовка аппаратуры включает несколько аспектов:

Есть нюанс, обусловленный простотой формирования СИП в интерфейсном устройстве (ИУ) типа рассмотренного в разделе 2 и в Приложении 1. По-хорошему, периферийное устройство должно выдавать СИП в ответ на машинный синхросигнал Вв = 0 при определённом коде на шине УПР. Однако наше простенькое ИУ не анализирует шину УПР и формирует СИП прямо из сигнала Вв = 0. В то же время в документации говорится (если я правильно понял), что машина выдаёт Вв = 0 всякий раз, когда не работает по программе, а просто зажигает свои табло X, Y, и ожидает СИП от нажатий кнопок на пульте. Из своего давнишнего опыта уже не помню (и затрудняюсь выяснить по документации), могут ли пульт и внешнее периферийное устройство (ПУ) конфликтовать по линии СИП. Поэтому на всякий случай в кабеле, который должен соединять ИУ с разъёмом "Ввод-вывод" машины, я вставил выключатель в разрыв провода линии СИП; этот выключатель виден на схеме на рис. 34.

Таким образом, перед запуском загрузчика, и вообще, – перед включением ИУ и машины Д3-28 выключатель линии СИП должен быть разомкнут.

Аппаратура подготавливается по обычному принципу – "от периферии к центру" (выключать же всё можно будет в обратном порядке). Т. е. соединяем нужным образом нужными шнурами выключенные плеер, ИУ, их БП, подключаем кабель с выключенной линией СИП между ИУ и Д3-28. Включаем все устройства, Д3-28 в последнюю очередь. Загружаем, как указано выше, загрузчик м-кода.

В режиме "B" устанавливаем номер шага на начальный адрес загрузчика (для примера у нас упоминался шаг 30000, если речь не о загрузке блочных программ). Возвращаемся в режим "P", задаём начальный адрес загрузки программки (обычно это 0). В плеере находим начало файла с нужной программкой, либо какое-нибудь место вблизи конца предыдущего файла; проигрывание пока не включаем.

На пульте Д3-28 нажимаем кнопку S – это запуск загрузчика. С этого момента машина работает по программе, с погашенными табло. Должен зажечься индикатор ожидания СИП от ПУ. Включаем (замыкаем) наш выключатель линии СИП. Включаем в плеере проигрывание.

Ожидаемая длительность проигрывания заданного файла должна быть нам известна. По истечении этого ожидаемого интервала времени, если загрузка успешно осуществилась, должен зажечься индикатор ОМ – красный светодиод слева от Y-табло машины. Это будет означать, что программа загрузчика успешно подошла к концу и зациклилась; зациклилась, чтобы не останавливаться и тем самым избежать упоминавшегося возможного нюанса с СИП.

Если загрузка провалилась, то либо ничего не произойдёт даже по истечении большого интервала времени, либо загорится индикатор ОП ("ошибка программы" – такое может случиться, если из-за какого-то неконтролируемого заполнения ОЗУ окажется затёртой сама программа загрузчика).

Во всех случаях для окончания данного сеанса выключаем (размыкаем) наш выключатель линии СИП, и затем останавливаем машину – кнопкой Ш или ШН (тогда можно будет в режиме "B" посмотреть, на какой адрес указывает программный счётчик), или просто кнопкой сброса C.

Тестирование wav-файлов с m-code и s-code

Имитатор не воспроизводит динамику работы Д3-28 и поэтому в нём невозможно протестировать временной ход загрузки файлов. Но можно проверить хотя бы факт отсутствия логических ошибок в структуре создаваемых "Устройством" wav-файлов, а также – проверить отсутствие логических ошибок в программке-загрузчике.

Проверка заключается просто в имитации Устройством проигрывания wav-файла с м-кодом, когда в имитаторе в имитируемом ОЗУ Д3-28 запущена на исполнение txt-программка с загрузчиком м-кода (см. в разделе 3 в табл. 1 описание команды 1400 1014).

В качестве простейшего иллюстративного примера откроем в Буфер Устройства (командой **View** > **Display**, **F1**, **W**) упоминавшийся в разделе 3 в *Примере 4* созданный Устройством wav-файл 001_Our_44100_m-code_dM-176_dF-23_N-1.wav. Через меню **File** откроем и загрузим на шаг 30000 наш loader_m-code_addr-x_kp-1812_n-138.txt. Запустим его при X = 0. Согласимся, нажав "Да", с предложением Устройства загрузить м-код. Когда загорится индикатор ОМ, нажмём клавишу C. Клавишей V (или W) перейдём в режим ввода и посмотрим байты имитируемого ОЗУ на шагах 00000 и 00001 – там будут видны успешно загрузившиеся байт-коды иллюстративной 2-байтовой программки pi_kp-15_n-1.txt, т. е: 0609 0512.

Ещё пример: если аналогично загружать 001_Our_44100_m-code_dM-176_dF-23_buff-10.wav из Примера *3* в разделе 3, то увидим успешно загрузившиеся 5 байт. Тетрады последнего байта там, как и задумывалось, изображают контрольную сумму "15": 0609 0512 0000 0000 0105.

Для аналогичной проверки s-кода надо сначала преобразовать его в м-код командой 1 Устройства, см. табл. 2 в разделе 3. С этой целью можно загрузить в имитируемое ОЗУ Д3-28 на шаг, например, 25000 программку для подачи команд в Устройство u_outo_kp-434_n-31.txt, упоминавшуюся в разделе 3. Запускаем её клавишей S. Набираем в X номер команды 1, нажимаем S; ещё раз S, и соглашаемся преобразовать s-код в м-код. Затем, как и в предыдущих примерах, запустим с шага 30000 загрузчик м-кода, и он уже, как и раньше. сделает своё дело.

Приложение 1: принципиальная схема интерфейсного устройства

Структурная схема интерфейсного устройства (ИУ) была приведена на рис. 9. Ниже дано описание его "радиолюбительской" реализации с применением старинных деталей, имевшихся у меня под рукой. Разумеется, никого не призываю повторять именно такую конструкцию. Моей целью была проверка рассмотренных аппаратно-программных соображений о передаче ПО из txt-файлов через аудио-тракт плеера / компьютера; читатель, заинтересовавшийся такой темой, при желании разработает и создаст подходящее ИУ на современной элементной базе.

Вывод сигнала из плеера через разъём «Наушники»

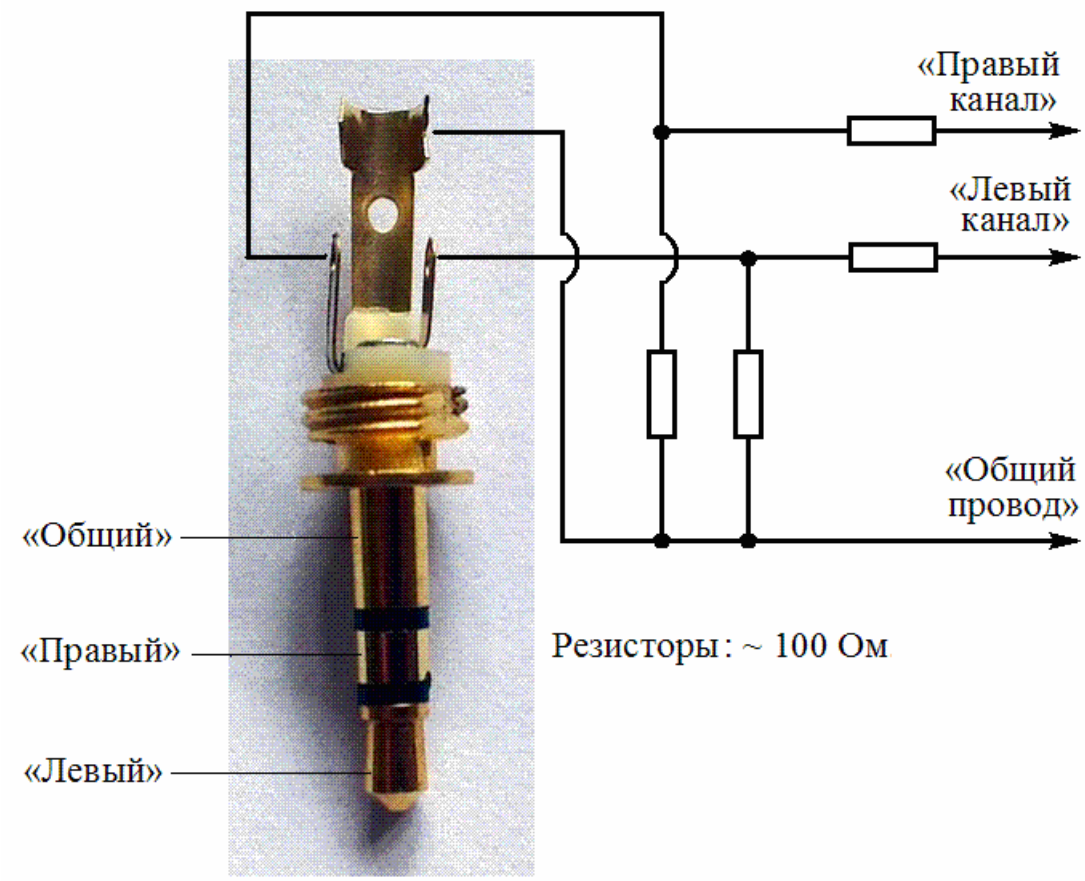


Рис. 31. Распайка выводов штекера, включаемого в разъём "Наушники" плеера.

Здесь можно применить резисторы с номиналом порядка 100 Ом, но это значение примерное; его, по-видимому, допустимо изменять в разы. Резисторы, изображённые вертикально, – нагрузочные, они заменяют собой сопротивления наушников. Резисторы, изображённые горизонтально – «защитные», они предохраняют выход плеера от замыкания сигнальных концов соединительного шнура друг на друга или на общий провод. У Синуса (т. е. у меня) номинал нагрузочных резисторов: 56 Ом, защитные резисторы: 620 Ом; не рассчитывал их, а просто шнур с такими резисторами уже был спаян раньше, для других целей.

Как показано ниже, сигнал одного из каналов плеера (например, "правого") подаётся на трансформатор во входной цепи ИУ; затем поступает на усилитель. Подразумевается, что оба канала плеера воспроизводят одну и ту же запись с одноканальным s-кодом.

Входная цепь ИУ, усилитель, формирователь синхроимпульсов, триггер

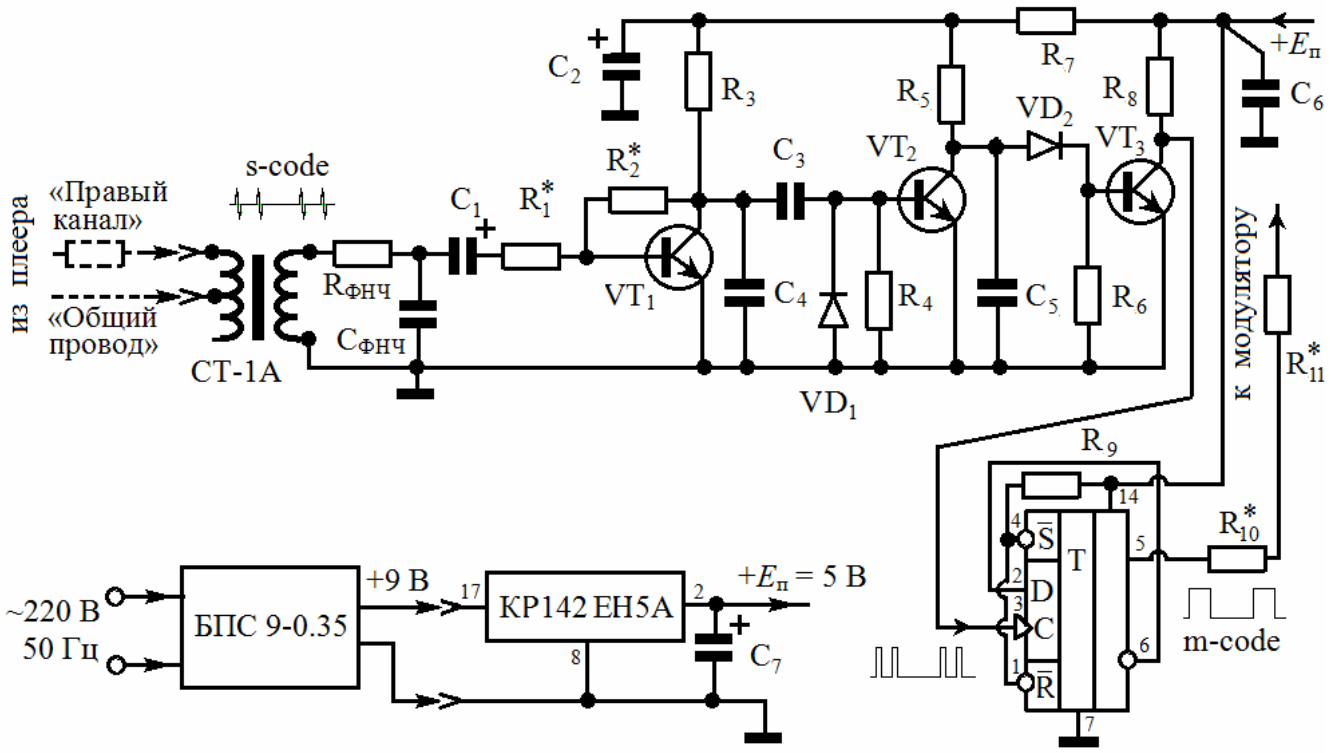


Рис. 32. Входной трансформатор, усилитель, формирователь импульсов для триггера, триггер и блок питания.

На входе ИУ – согласующий низкочастотный трансформатор СТ-1А от малогабаритного радиоприёмника; он обеспечивает гальваническую развязку между плеером и ИУ. Вторичная обмотка СТ-1А нагружена на RC-фильтр низких частот (ФНЧ), подавляющий ВЧ-наводки и помехи от блока питания (от зарядного устройства) плеера. Затем сигнал с "s-code" усиливается транзистором VT_1. Звёздочка * означает, что номинал данного элемента следует подбирать. Чем больше сопротивление R_1, тем меньше коэффициент усиления. R_2 подбирается так, чтобы напряжение между коллектором и эмиттером VT_1 равнялось примерно половине напряжения питания, т. е. $+E_п / 2 = 2.5$ В. Транзистор VT_2 в отсутствие сигнала закрыт (а VT_3 открыт), поскольку ток в базу VT_2 не подаётся. Положительной частью усиленного s-импульса VT_2 открывается, при этом VT_3 закрывается – так на коллекторе VT_3 формируется положительный синхроимпульс для триггера. Диод VD_1 устраняет отрицательную часть s-импульса. Диод VD_2 служит простейшей цепью "сдвига уровня" между коллектором VT_2 и базой VT_3.

Триггер ½ К555 ТМ2 преобразует последовательность синхроимпульсов в импульсы м-кода ТТЛ-формата. Эти импульсы управляют модулятором (показанным на следующем рисунке) – вместе с суммарным сопротивлением $R_{10} + R_{11}$ выход триггера служит источником тока базы транзистора в модуляторе. Подбором резисторов R_{11} и R_{10} работа модулятора "грубо" и "тонко" настраивается так, чтобы логической единице на выходе триггера соответствовал максимум напряжения между эмиттером $T_{дет\,1}$ и общим выводом (0В) в схеме, показанной на следующем рисунке.

Номиналы:

$R_{ФНЧ}$ = 1.5 кОм	$C_{ФНЧ}$ = 0.047мкФ	C_1 = 5.0 мкФ	C_2 = 100.0 мкФ	C_3 = 2.2 мкФ	C_4 = 2200 пФ
C_5 = 4700 пФ	C_6 = 0.1 мкФ	C_7 = 200 мкФ			
R_1 = 1 кОм	R_2 = 330 кОм	R_3 = 3.9 кОм	R_4 = 82 кОм	R_5 = 18 кОм	R_6 = 82 кОм
R_7 = 220 Ом	R_8 = 3.3 кОм	R_9 = 1.3 кОм	R_{10} = 1.3 кОм	R_{11} = 51 кОм	

Все транзисторы в моём макете ИУ – типа КТ315 (с разными буквами; уж какие оказались под рукой). Все диоды – кремниевые: Д223Б или Д220Б.

Упомянутый модулятор работает в паре с детекторным каскадом – на него сигнал с выхода модулятора подаётся через ВЧ-трансформатор. Такая пара служит аналогом оптронной гальванической развязки между триггером, формирующим м-код, и выходом ИУ, подключаемым к контактам разъёма ВВОД-ВЫВОД машины Д3-28:

Схема ВЧ-генератора и модулятора

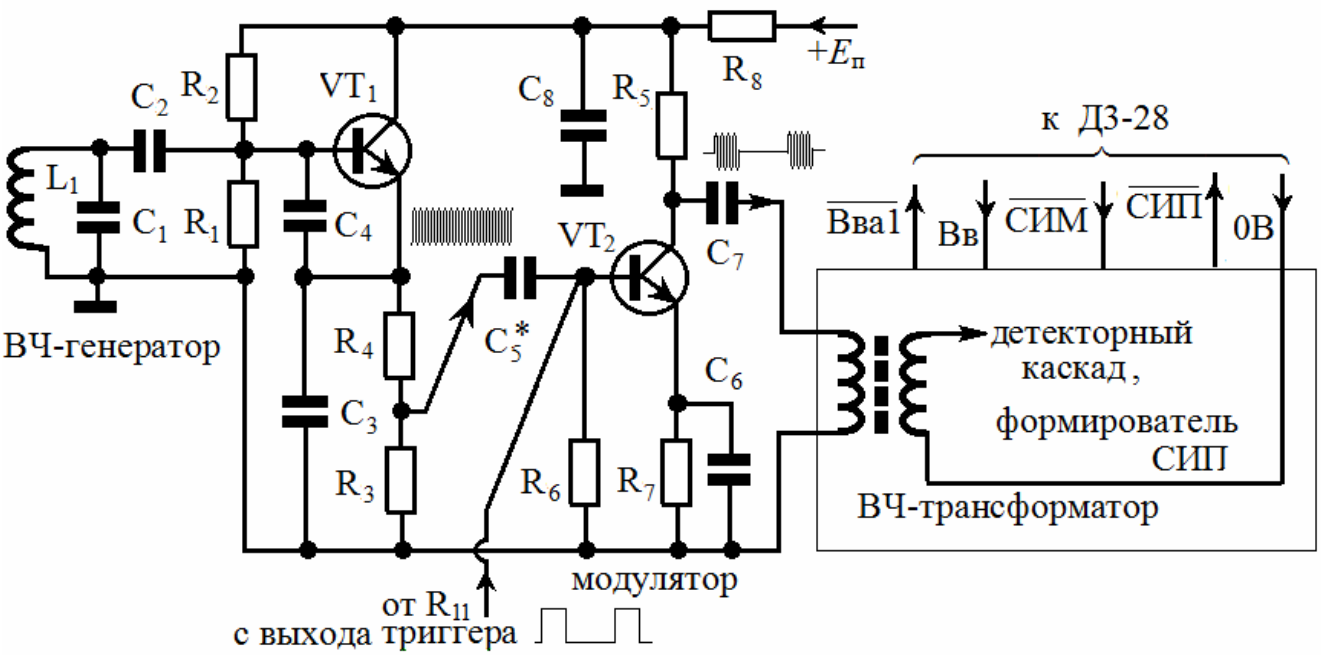


Рис. 33. Генератор ВЧ-напряжения и модулятор.

Генератор – "емкостная трёхточка" на транзисторе VT_1. Конкретное значение частоты и её стабильность здесь роли не играют. Колебательный контур L_1C_1 взят наугад: катушка L_1 – 50 витков на каркасе диаметром 8 мм, длина намотки 30 мм, сердечника нет; C_1 ≈ 120 пФ; при этом генерируемая частота равна примерно 6 МГц. Модулятор – ВЧ-усилитель на транзисторе VT_2. ВЧ-напряжение из генератора поступает на вход модулятора через C. При логическом нуле на выходе триггера ток, подаваемый в базу модулятора VT_2 через $R_{10} + R_{11}$ с выхода триггера (см. предыдущий рисунок), близок к нулю, так что модулятор VT_2 должен быть закрыт. Если при наладке модулятора окажется, что VT_2 открывается напряжением генератора, то следует уменьшить емкость C_5. Если же при логической единице на выходе триггера окажется, что транзистор модулятора VT_2 открывается слишком сильно (входит в насыщение), то следует увеличивать $R_{10} + R_{11}$, добиваясь максимума ВЧ-сигнала на выходе модулятора.

Номиналы:

C_1 = 120 пФ	C_2 = 68 пФ	C_3 = 180 пФ	C_4 = 150 пФ	C_5 = 680 пФ	C_6 = 4700 пФ
C_7 = 2200 пФ	C_8 = 10 нФ				
R_1 = 12 кОм	R_2 = 12 кОм	R_3 = 1.8 кОм	R_4 = 150 Ом	R_5 = 1.8 кОм	R_6 = 51 кОм
R_7 = 1 кОм	R_8 = 430 Ом				

Источник питания $+E_п$ = 5 В и "земля" здесь те же самые, что на предыдущей схеме, но не те же самые, что на следующем рисунке. Сигнал с выхода модулятора через конденсатор C_7 поступает на первичную обмотку ВЧ-трансформатора, обеспечивающего гальваническую развязку между ИУ и Д3-28; см. следующий рисунок:

Схема выходных цепей интерфейсного устройства

Варианты (А, Б, В, Г) подключения выхода ИУ к Д3-28 без гальванической связи с остальной частью ИУ уже были показаны на рис. 10 – 13 в разделе 2. На макете ИУ вариант выбирается перемычками, которые вставляются в гнёзда разъёма типа ГРПМ1; см. рис. 34 и 35:

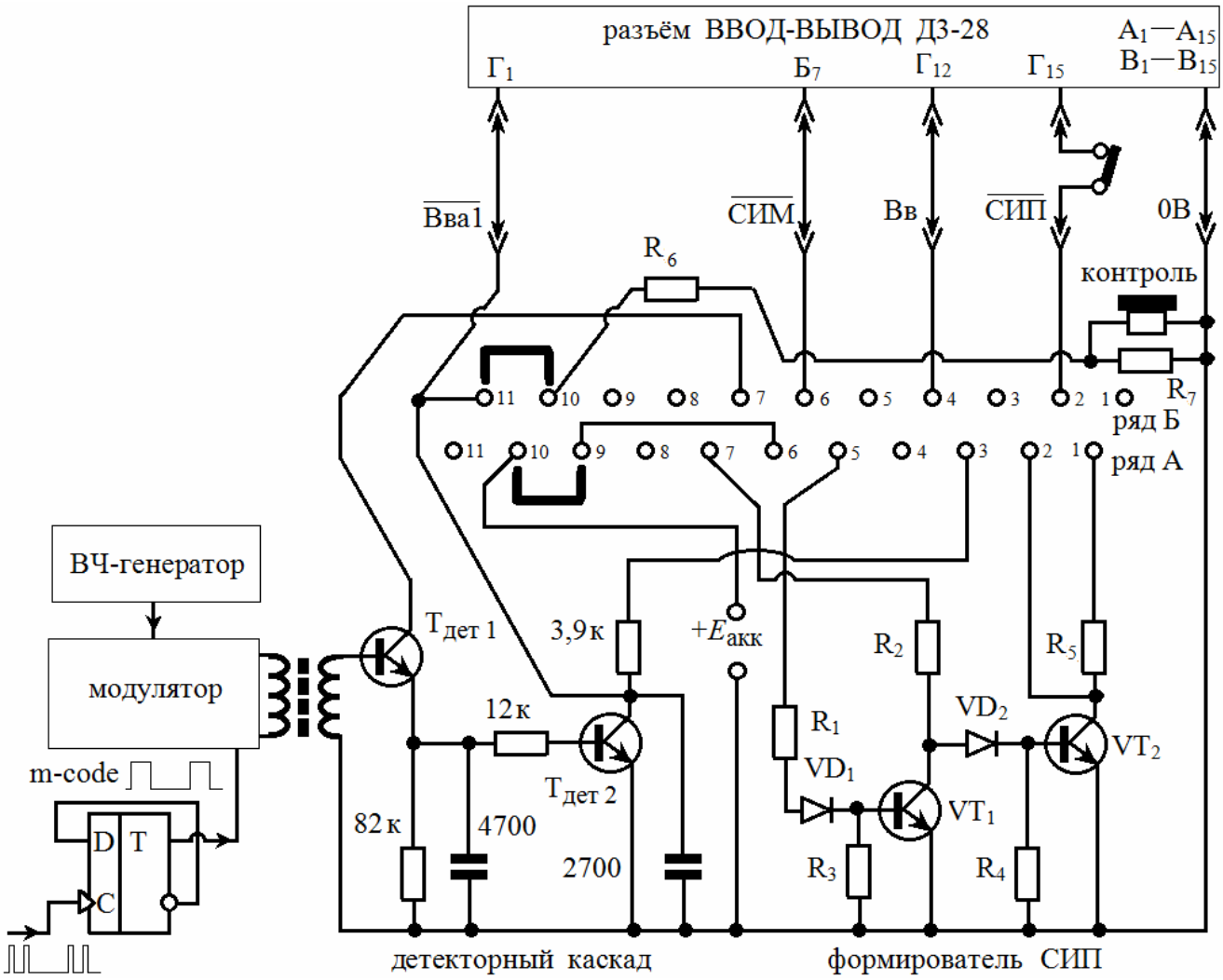


Рис. 34. Подключение выхода ИУ к Д3-28 без гальванической связи с остальной частью ИУ

Трансформатор – две обмотки по 40 витков на ферритовом кольце с внешним диаметром 10 мм. Номиналы резисторов – $R_1 = R_2 = 18$ кОм, $R_3 = R_4 = 82$ кОм, $R_5 = R_6 = 3.9$ кОм, $R_7 = 300$ Ом. Для контроля применён наушник ТОН-2 ($R_{\text{зв. катушки}} = 1.6$ кОм). Без перемычки 11–10 в ряду гнёзд Б контроль отключен и не нагружает выход ИУ Ввa1. Перемычка 10–9 в ряду А аналогично служит для вкл. / откл. батареи из четырёх аккумуляторов с суммарным напряжением $+E_{\text{акк}} = 4 \cdot 1.2 \approx 5$ В. Выбор вариантов А, Б, В, Г (они рассмотрены в разделе 2) осуществляется комплексами перемычек:

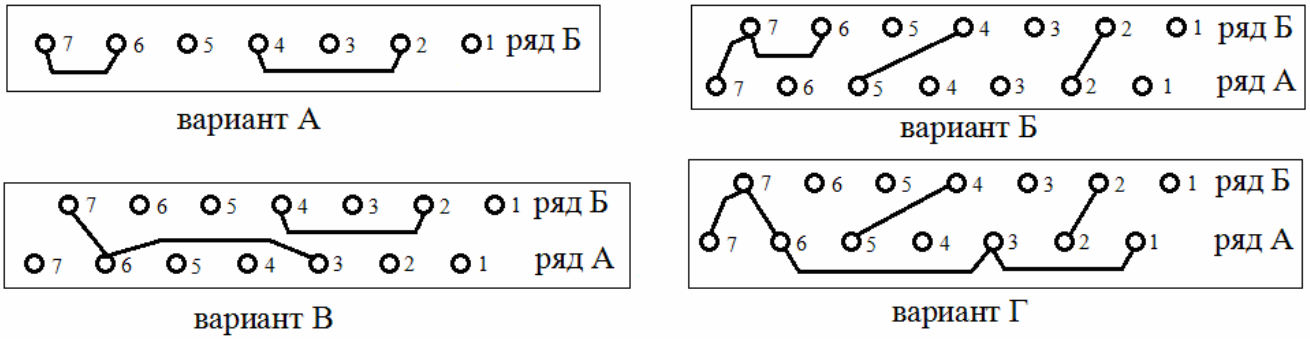


Рис. 35. Перемычки для схемы, показанной на рис. 34.

Приложение 2: схема двухканальной записи на МЛ

Источник токов записи (типа показанных на рис. 1 в разделе 0) для 2-канальной магнитной головки (МГ) изображён на рис. 36:

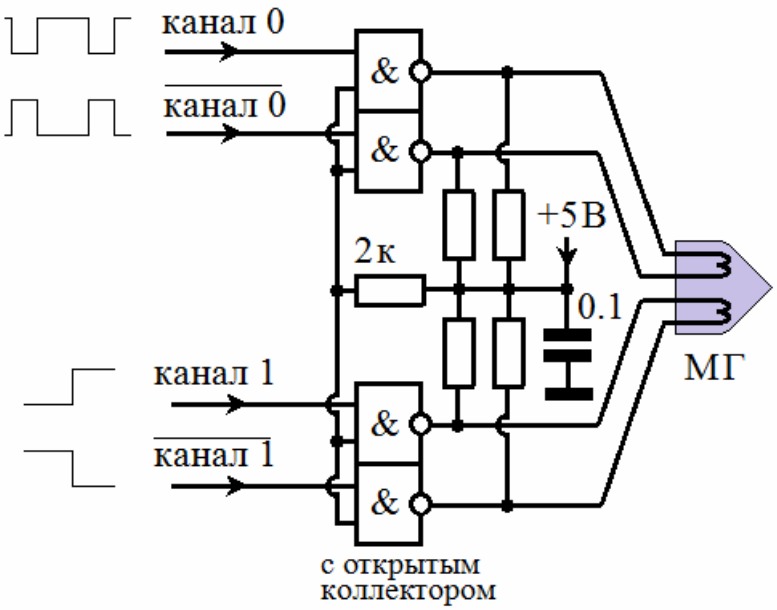


Рис. 36. Подключение 2-канальной записывающей МГ к источникам тока для формирования на МЛ записи типа "обычная 2-канальная для Д3-28".

В схеме Д3-28, отсканированной и любезно предоставленной уважаемым В. В. Колесником на его сайте d3-28.ru, в качестве элементов 2И-НЕ с открытом коллектором указана микросхема К155 ЛА8; номинал резисторов на выходах этих элементов: 2.7 кОм. Возможно, что при повторении подобной схемы в любительских условиях номинал таких резисторов, – они определяют величину тока записи, – придётся подбирать в зависимости от типа приобретённой МГ и типа МЛ. Думается, ток записи следует отрегулировать так, чтобы он был достаточным для намагничивания МЛ до насыщения.

В машине Д3-28 включение записи, т. е. подача на резисторы напряжения +5 В производится через транзисторный ключ, управляемый внутренними сигналами машины (при этом на соединённые входы элементов & с выхода триггера К155 ТМ2 подаётся напряжение логической единицы, но для простоты я изобразил эти входы подключенными к источнику +5 В через резистор, с навскидку выбранным номиналом 2кОм). В схеме на рис. 36 подразумевается, что режим записи включается вручную – подключением источника +5 В.

Изменения направления тока записи (типа показанных на рис. 1 в разделе 0) реализованы просто. В каждом канале в каждый момент времени на одном из двух выходов элементов & присутствует "напряжение логической 1", на другом – ноль; при этом ток через записывающую катушку течёт от выхода с "единицей" к выходу с нулём. После того, как на входы этих элементов & поступит очередной перепад, выходы меняются ролями, и направление тока в записывающей катушке соответственно изменяется на противоположное.

Схема на рис. 36 – не полная. Подразумевается, что прямой и инвертированный ТТЛ-сигналы, как для канала нулей, так и для канала единиц вырабатываются некоторым не изображённым на рис. 36 контроллером из 2-полярных данных в wav-файле типа "обычный 2-канальный Д3-28". Т. е. программа в контроллере берёт из wav-файла в каждом канале очередное число-сэмпл

и смотрит его знак: если число положительное, программа посылает в ТТЛ-порт на выходе контроллера логическую 1, а иначе – посылает 0; в ТТЛ-порт с инвертированным сигналом посылается всё наоборот: 0, если сэмпл положительный, 1 – в противном случае.

Наряду с тем, что такой контроллер должен уметь работать с какой-то памятью довольно большого объёма – ведь размер wav-файлов с ПО для Д3-28 может достигать нескольких десятков мегабайт, – он должен читать сэмплы из wav-файла со строго определённой скоростью, чтобы обеспечить заданную скорость записи бит на МЛ (соображения о скорости записи / чтения МЛ приводились в разделе 1 после рисунка 5). Т. е. фактически такой контроллер должен уметь "проигрывать" wav-файл из некоего устройства памяти, как плеер. Легко ли сконструировать такой аппарат, к сожалению, я не в курсе.

Привожу ещё одну структурную схему; вероятно, более простую с точки зрения старинного радиолюбительства. Она отличается от 1-канального ИУ типа рис. 9 только тем, что в ней должны быть задействованы оба стерео-канала плеера, два формирователя импульсов для двух D-триггеров, а вместо каскадов связи с Д3-28 в ней предполагается подсоединить источник токов записи с магнитной головкой, показанный на рис. 36:

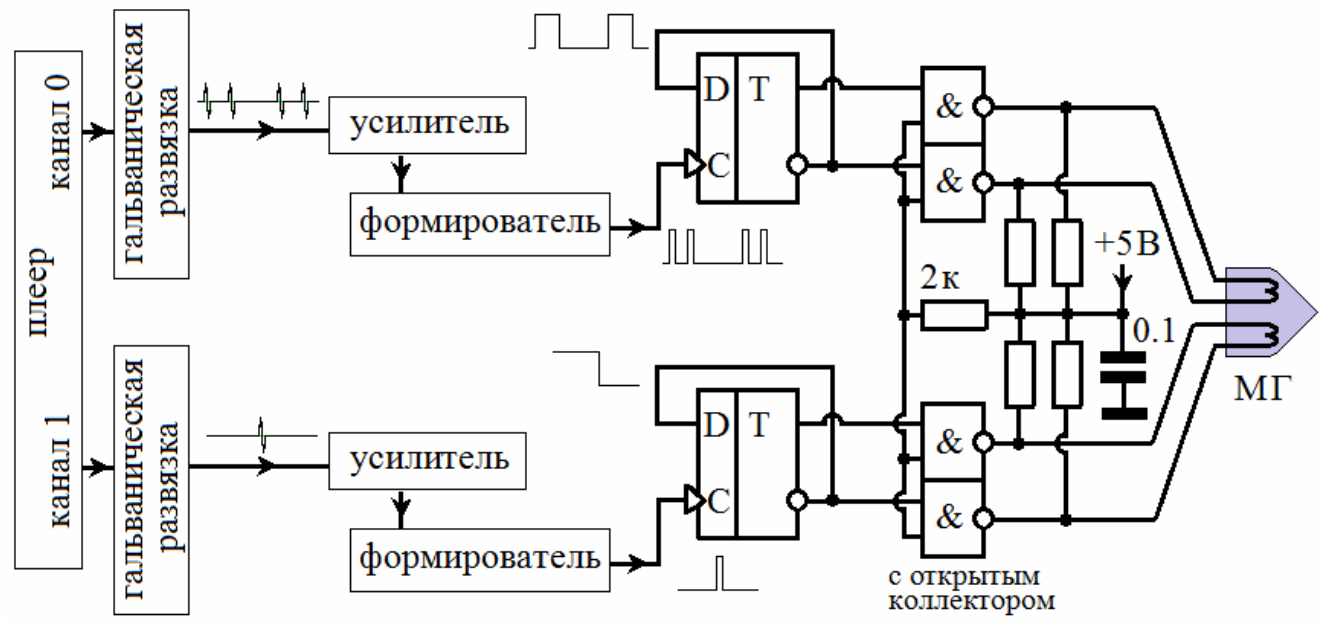


Рис. 37. Формирование записи типа "обычная 2-канальная для Д3-28" из wav-файла типа "2-канальный с sin-импульсами" с помощью внешнего ЛПМ.

Как и в схеме на рис. 36, номинал резисторов на выходах элементов & ориентировочно: 2.7 кОм. Для включения / выключения токов записи предполагается источник напряжения +5 В подключать / отключать вручную. Принципиальные схемы подачи сигналов из плеера и формирования синхроимпульсов триггеров здесь для каждого из двух каналов могут быть выбраны такие же, как для одного канала на рис. 32.

Август 2022. Sinus.