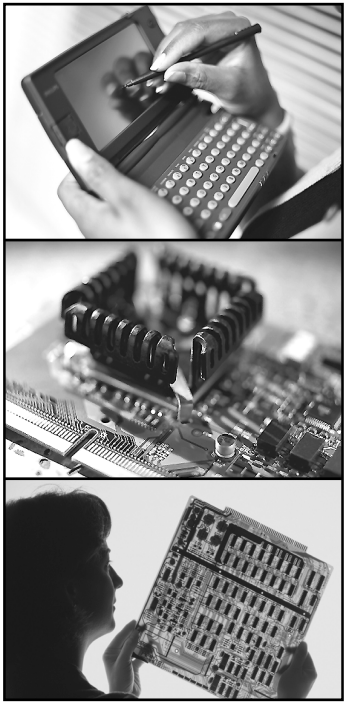
NedoPC is not PC

NedoPC

третий номер осень 2005

анонс

содержание



анонс следующего номера:

больше троичности.
компьюьеры прошлого.
программируем
в nedopc sdk.

Редакция журнала:

Главный Редактор Shaos

Оформление Olga

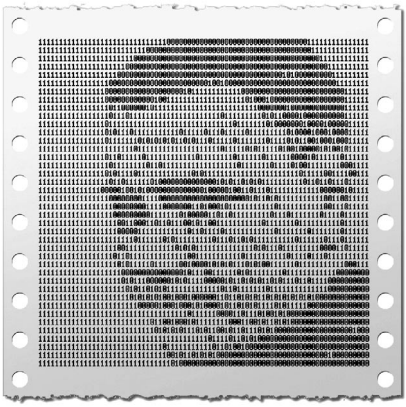
Тираж издания произвольный. Издание не подлежит регистрации, так как выпускается тиражом менее 1000 экземпляров.

E-mail: nedopc@mail.ru

http://www.nedopc.org

http://www.nedopc.com

По вопросам подписки обращаться по адресу: 109451, Москва, ул.Братиславская, д.13, корп.1, кв.228, Чунину Роману Валерьевичу



колонка редактора

Наконец-то вышел в свет долгожданный третий номер! Захватывающая фантастика о будущем компьютера Спринтер закончилась, и номер пришлось дополнять статьями технического характера. Если у тебя, дорогой читатель, есть способности фантаста, и ты любишь Спектрум или другой ретро-компьютер – напиши свою фантастическую историю и присылай. Если это действительно то, что нужно – напечатаем!

Темой номера на этот раз стало устройство, созданное нашими недописишными умельцами, которое реализует звуковой стандарт "Turbo Sound", широко известный в околоспектрумистских кругах, но реализует с помощью более простой схемы. Кроме описания девайса мы также расскажем о том, как его программировать.

Помимо темы спектрума в номере затронута историческая тема и тема использования программируемых логических матриц. Далее в номере можно найти две статьи на тему реализации в железе компьютера, построенного на принципах уравновешенного троичного исчисления. Завершает номер бессменная реклама компьютера "ATM Турбо-2+" и страничка объявлений.

Пользуясь случаем, перечислю сайты в интернете, где мы обитаем: www.nedopc.org, www.nedopc.com, atmturbo.nedopc.com и www.ternary.info. Приходи к нам – читай, пиши, вобщем принимай активное участие в жизни нашего сетевого сообщества нестандартных индивидуумов, распределенных по всей планете!

Почему-то никто не пользуется бланками бесплатных объявлений, а ведь это так просто – вырезал, заполнил и отправил по адресу, указанному на страничке слева. Кроме того, с нами можно связаться по электронной почте nedopc@mail.ru или пообщаться с нами на нашем форуме http://forum.nedopc.org (где мы обещали открыть для широкой общественности отдельный форум под названием "Журнал NedoPC", но так и не открыли, так как ждем тебя, дорогой читатель ;).

И напоследок небольшой Конкурс: На каком музыкальном инструменте играет девушка, изображенная на обложке этого номера журнала? Тот, кто раньше назовет марку и год производства этого чуда музыкальной техники – получит приз от команды NedoPC! А теперь – читать! :)

Главный редактор Shaos

turbo sound

автор CHRV

Здравствуйте, дорогие читатели. С помощью этой статьи я попытаюсь познакомить с нашей разработкой под названием Turbo Sound.

Что такое Turbo Sound? В далекие годы, когда Спектрум считался современным бытовым компьютером, у него была музыкальная подсистема, основанная на микросхеме AY-3-8910, AY-3-8912 или YM2149. Это микросхема позволяла генерировать программно различный звук, описываемый частотой, амплитудой и формой огибающей. Микросхема содержит три канала для генерации такого звука (подобные микросхемы использовались в игровых приставках, стационарных телефонах и даже музыкальных синтезаторах). Но, как обычно, трех каналов не хватало для самореализации, и встал вопрос о расширении возможностей.

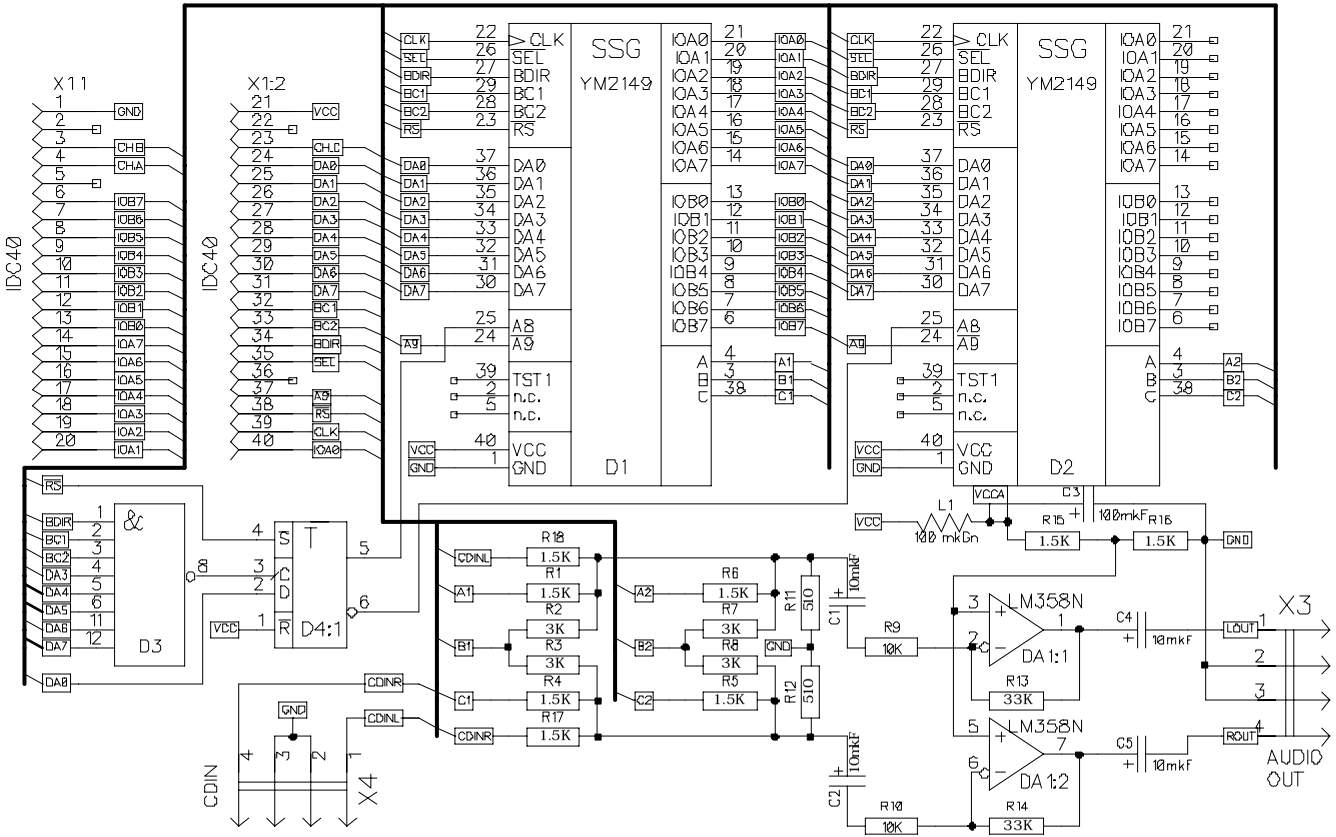
Западные производители сделали новый чип, который использовался в бытовом компьютер Sam Coupé , и содержал в себе возможность шестиканальной генерации звука по тем же принципам что у AY-3-8910 (к сожалению, программно не совместим с предшественником). А российские умельцы располагали только той элементной базой, которая доступна на рынке. Поэтому решили объединить два чипа в общую схему и с раздельным программным доступом к чипам. Так появилась первая схема TurboSound и музыкальный редактор для него. Автор схемы и названия – Илья Кудрявцев (Himik) музыкант, активный спектрумист из ижевской команды. К сожалению, эта схем была достаточно громоздка и тяжела для интеграции в компьютер, поэтому не получила большого распространения.

Нашей группе удалось сильно упростить устройство и главное сделать его легким в установке на компьютер (Turbo Sound устанавливается на посадочное место музыкального чипа через переходник). Потерялась программная совместимость со схемой Кудрявцева, но он согласился переделать свой редактор. Плата устройства обладает небольшими размерами и легко устанавливается в компьютер. Можно сказать, что только сейчас началось принятие TurboSounda спектрумистким сообществом. Поддержка реализована в эмуляторах (Unreal, EmuZWin). К сожалению, продемонстрировать Turbo Sound во всей красе не удалось на Chaos Construction 2005, но с гордостью замечу, что вся AY-музыка выводилась через наше устройство.

Разработчики новой схемы TurboSounda Чунин Роман и Виктор Рощупко. Свои вопросы по принципам работы можете спросить у нас на форуме. Вкратце принцип работы следующий: оба музыкальных чипа доступны по одним и тем же портам, как стандартный музыкальный чип Спектрума, но активным является только один музыкальный чип. Как известно, для работы с музыкальным чипом используются его внутренние адреса регистров от 00h до 0Fh, а переключение активного чипа мы спрятали в адреса FFh и FEh – т.е. последний бит адреса указывает, какой чип сейчас станет активным. Выходы с чипов смешиваются по стандарту ABC.

На данный момент наша группа подготовила в производство в соавторстве с Акимовым Вадимом устройство - аналог Turbo Sound, но дополнительно еще два канала с FM модуляцией звука (на базе музыкальных чипов YM2203). Следите за новостями на наших сайтах. ▲

##FF-set first AY D1,D2 - YM2149; D3-1533LA2; D4-1533TM2
##FE - set second AY D4-LM358



программирование
turbo sound

автор SHIRU OTAKU

ВЫБОР ЧИПА

Оба чипа управляются одними и теми же стандартными портами (#FFFD — выбор регистра AY, #BFFD — вывод значения в регистр AY). В один момент времени к этим портам подключается один из чипов, и все выводы в регистры AY направляются в него. При сбросе выбирается чип номер 0. Переключение между чипами производится путем выбора регистра AY с номером, находящимся вне диапазона существующих регистров. Выбор регистра #FF подключает нулевой чип, #FE — первый.

Условная процедура выбора нулевого чипа:

```
selChip0:
    LD BC,#FFFD
    LD A,#FF
    OUT (C),A
    RET
```

Аналогично происходит выбор второго чипа:

```
selChip1:
    LD BC,#FFFD
    LD A,#FE
    OUT (C),A
    RET
```

Можно сделать простую процедуру, выбирающую нужный чип по его номеру в аккумуляторе (0...1):

```
selChip:
    LD BC,#FFFD
    AND 1           ; во избежание ошибок при числе в аккумуляторе
                    не равном 0 или 1
    XOR B
    OUT (C),A
    RET
```

Полная совместимость TS со всем ранее написанным ПО обеспечивается при условии, что оно не пытается производить выбор несуществующих регистров AY (на текущий момент программ, не отвечающих этому условию, не замечено).

ИСПОЛЬЗОВАНИЕ TS

Так как второй чип не имеет собственных управляющих портов, обеспечивается возможность использовать любые, ранее написанные процедуры вывода звука без всяких доработок. Например, есть две процедуры: проигрывающая музыку и проигрывающая звуковые эффекты. На одном чипе для проигрывания звуковых эффектов параллельно с музыкой приходится временно выключать один или несколько каналов музыки, чтобы дать возможность проиграться звуковому эффекту. При наличии Turbo Sound можно запустить музыку на одном чипе, а эффекты на втором, используя те же процедуры. Нужно только вставить переключение чипов между ними.

Например, было:

```
    CALL musicPlay
    CALL soundPlay
```

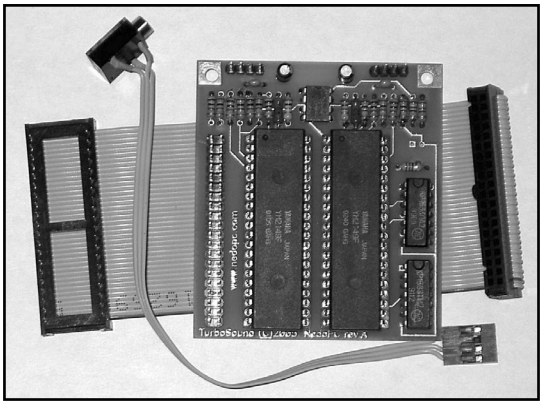
Стало:

```
    CALL selChip1
    CALL musicPlay
    CALL selChip0
    CALL soundPlay
```

При этом при отсутствии TS новый вариант будет работать точно также, как старый.

На текущий момент не существует специализированных плееров музыки под TS. Но в тестовых целях возможно осуществить такое проигрывание, создав два обычных трехканальных модуля (в каждом из них должно содержаться по три канала шестиканальной композиции), и откомпилировав их в разные адреса. Для проигрывания достаточно будет сделать следующее:

```
CALL selChip1
CALL player1
CALL selChip0
CALL player0
```



При этом каждому из плееров не нужно знать, на какой чип он выводит данные.

Важный момент: после завершения работы с TS нужно выбирать нулевой чип. Также это нужно делать перед выполнением программного сброса. В противном случае возможны проблемы с работой устройств, подключаемых к портам ввода-вывода AY, так как на разъем TS заводятся только линии портов ввода-вывода нулевого чипа. Примеры кода, приведенные выше, учитывают это требование.

ДРУГИЕ РЕАЛИЗАЦИИ TURBO SOUND

Все вышесказанное относилось к работе с TS версии NedoPC. Но существует также два старых варианта схемы Turbo Sound: QUADRA от Amazing Soft Making и вариант TS от Power of Sound.

Вариант QUADRA полностью несовместим с NedoPC TS — в нем предполагается наличие собственных управляющих портов для второго чипа (#EEFD — выбор регистра AY, #AFFD — вывод значения). Поддержка этого варианта схемы лишена смысла, ввиду ее непопулярности и полном отсутствии программного обеспечения.

Вариант TS от Power of Sound поддержан в единственном на данный момент редакторе шестиканальной музыки, Turbo Sound Editor. С точки зрения программирования он отличается только способом переключения чипов, что позволяет легко добавить его поддержку в создаваемом для NedoPC TS программном обеспечении. Для этого достаточно предоставить пользователю возможность выбора типа TS, и соответствующим образом доработать процедуры выбора чипа.

Переключение чипов в PoS TS происходит посредством вывода значения 0..1 в порт #1F (номер выбираемого чипа соответствует выводимому значению).

Ниже приводится вариант универсальной процедуры выбора чипа по его номеру в аккумуляторе. Предполагается, что тип TS сохранен по адресу chipType+1, нулевое значение соответствует варианту TS от NedoPC, ненулевое — варианту от PoS.

```
selChip:
    AND 1

chipType:
    LD B,0                  ; вместо нуля присутствует число,
                              соответствующее типу TS

    DEC B
    JR NC,chipPoS
    LD C,#FD
    XOR B
    OUT (C),A
    RET

chipPoS:
    OUT (#1F),A
    RET
```

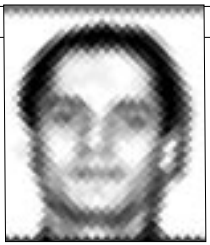
АВТООПРЕДЕЛЕНИЕ НАЛИЧИЯ И ТИПА TS

Ниже приводится текст процедуры автоматического определения наличия и типа TS (варианты NedoPC и PoS), использующейся в Turbo Sound Editor. Следует учесть, что в случае невозможности чтения значений регистров AY (встречается в редких случаях) автоматическое определение работать не будет. Поэтому желательно предусматривать возможность указания типа TS «вручную».

```
;Turbo-Sound checker by Himik's ZxZ/PoS-WT
;24.05.05 at work ;)
;Found:
;       No ay/ym chip on board
;       Single ay/ym chip on board
;       Turbo-Sound port by PoS & Bitwalker (port #1F for swith)
;       Turbo-Sound port by NedoPC (registors #FE-#FF selection)
                ORG     #61A8
C_1
                DI
                XOR     A
                LD      HL,#FE00
                LD      DE,#FFBF
                LD      BC,#FFFD
                OUT     (C),B ;SELECT TS AY0 CHRV
                OUT     (C),A ;SELECT REG 0
                LD      B,E
                OUT     (C),B ;WRITE #BF IN REG 0 AY0 CHRV
                INC     A
                OUT     (#1F),A ;SELECT TS AY1 POS
                OUT     (C),C ;WRITE #FD IN REG 0 AY1 POS
                LD      B,D
                OUT     (C),H ;SELECT TS AY1 CHRV
                OUT     (C),L ;SELECT REG 0
                LD      B,E
                OUT     (C),H ;WRITE #FE IN REG 0 AY1 CHRV
                LD      A,L
                OUT     (#1F),A ;SELECT TS AY0 POS
                OUT     (C),L ;WRITE #00 IN REG 0 AY0 POS
                INC     A
                OUT     (#1F),A ;SELECT TS AY1 POS
                LD      B,D
                OUT     (C),D ;SELECT TS AY0 CHRV
                OUT     (C),L ;SELECT REG 0
                IN      A,(C) ;READ BYTE FROM REG 0
                CP      C
;переходим, если найден Turbo Sound by NedoPC
                JR      Z,TS_ENABLE_CHRV
;переходим, если найден Turbo Sound by PoS
                CP      #FE
                JR      Z,TS_ENABLE_POS
;переходим, если ни одного чипа не найдено
                CP      #FF
                JR      Z,NO_AY
C_2
;найден только один чип
TE_DISABLE
                LD      A,1
                OUT     (#FE),A
                RET
NO_AY
                LD      A,2
                OUT     (#FE),A
                RET
TS_ENABLE_CHRV
                LD      A,4
                OUT     (#FE),A
                RET
TS_ENABLE_POS
                LD      A,6
                OUT     (#FE),A
                RET
DISPLAY /A, "Length: ",C_2-C_1
```

До самых первых

автор Mac Buster

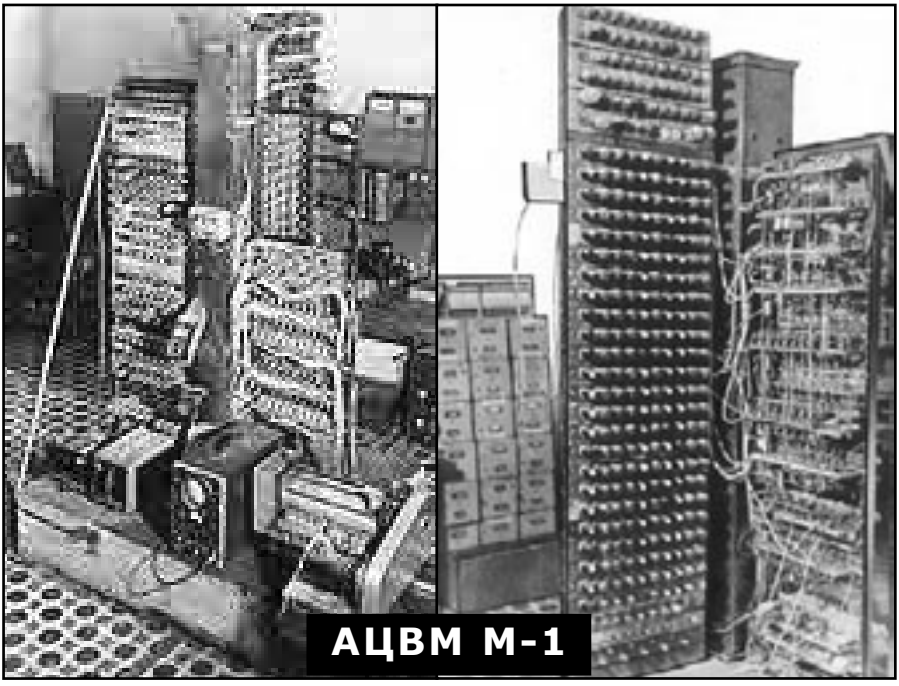


Мало кто знает, что созданию электронных вычислительных машин предшествовало создание своеобразных упрощенных макетов, предназначенных для оценки осуществимости, отработки технологий и определения наиболее подходящих компонентов. За рубежом одной из таких машин была британская Manchester SSEM (Small Scale Experimental Machine) или, как ее еще называли — Baby. День ее запуска — 21 июня 1948 года — считается официальным днем рождения электронной вычислительной техники и целой индустрии, которая сейчас называется IT.

Тем, кто интересуется историей вычислительной техники и, в частности, первыми электронными вычислительными машинами, будет очень интересно узнать о том, что существует эмулятор Baby (SSEM), на котором можно запустить ту самую — первую в мире — программу для электронной вычислительной машины с хранимой программой. Скачать этот эмулятор и инструкцию по программированию можно со следующего сайта:
http://www.davidsharp.com/baby/

Желаю вам успехов в освоении программирования этой замечательной машины! Вас ждет очень много интересного, в первую очередь – очень небольшой набор команд, который можно классифицировать как настоящий MISC – minimal instruction set computer. Освоив программирование вы сможете принять участие в конкурсе программистов Baby.

В Советском Союзе так же создавались машины-макеты. Если созданная в Феофании (пригород Киева) машина МЭСМ обычно представляется в качестве первой советской полноценной вычислительной машины (хотя это не совсем соответствует истине, на самом деле это был скорее самый настоящий макет ЭВМ) и достаточно



АЦВМ М-1

хорошо известна почти всем, кто учился в технических ВУЗах, то про М-1 знает очень небольшое количество человек. Надо сказать, что М-1 создавалась в тот же период времени что и МЭСМ, однако, считалась внутренним вузовским проектом и для нее не выделялись государственные средства и материалы, как для МЭСМ. Разработана она была группой молодых талантливых выпускников МЭИ под руководством Исаака Семеновича Брука. Не смотря на полулегальное положение, группе удалось самостоятельно разработать и построить машину не взирая на отсутствие знаний в новой (для всего мира) области, нехватку средств, необходимых элементов (следствие тяжелого положения в разрушенной войной стране), места (работы велись в Москве, в полуподвальном помещении площадью всего 15м^2, расположенном в правом крыле дома №18 по Ленинскому проспекту, в то время называвшемуся Большой Калужской улицей) и даже на прямое противодействие некоторых чиновников. Отчасти благодаря противодействию им удалось найти удачную замену дефицитным электронным лампам и в результате значительная часть машины была построена на полупроводниках, что позволило сократить размеры (всего до 4-х м2) и потребляемую мощность (до 8кВт).

ЭВМ М-1, в отличие от МЭСМ, была настоящей электронной вычислительной машиной с хранимой программой. Она имела двухадресную систему команд, оперативную память емкостью пятьсот двенадцать 25-разрядных слов. Причем, 256 слов хранилась в быстродействующем электростатическом запоминающем устройстве (электронно-лучевые трубки), а оставшиеся 256 слов на магнитном барабане. Числа представлялись с фиксированной точкой в двоичной системе счисления: 1 разряд отводился под знак, а остальные 24 разряда – под мантиссу числа. Производительность М-1 была невысока и составляла от 15 до 20 операций в секунду.

Первая выполненная М-1 программа.

Осенью 1951 года была завершена комплексная наладка. Одной из первых успешно выполненных на ЭВМ М-1 программ была несложная задача решения уравнения параболы y = x. Эта задача примечательна тем, что в процессе ее решения получались одинаковые значения y как для положительного так и для отрицательного значения x. Таким образом, сравнивая симметричные значения результатов можно было судить о правильности выполнения программы машиной. Это было удачной находкой, т.к. в то время еще не существовало даже понятия о специальных тестовых программах для проверки работоспособности ЭВМ. Сравнение параболы было первой тестовой программой для М-1.

Второй тестовой программой было решение уравнения y = 1/x. Успешным решением этих уравнений завершилась комплексная наладка машины. И вот, 15 декабря 1952 года, машина успешно прошла испытания и была официально введена в эксплуатацию. Одними из первых на машине решались задачи из области ядерных исследований. Выполнял их академик Сергей Львович Соболев, который в то время был заместителем директора по научной части в институте И.В.Курчатова. М-1 находилась в эксплуатации на протяжении трех лет, первые полгода из них она была единственной действующей ЭВМ на территории Российской Федерации!

Было бы очень интересно собрать всю доступную информацию об этой замечательной машине и написать ее эмулятор, однако, к сожалению, все мои попытки найти хотя бы частичное описание системы команд М-1 пока не привели к успеху. Если вы располагаете какими-нибудь материалами, пожалуйста, свяжитесь со мной, я буду крайне благодарен!

УЧИМСЯ РАБОТАТЬ С ПРОГРАММИРУЕМЫМИ ЛОГИЧЕСКИМИ МАТРИЦАМИ

автор CHRV

«ALTERA» — это камень, о который многие ушиблись мозгами (головой).
Open Letters. (C) Nemo

Современные устройства становятся все миниатюрнее и мощнее. Часы, мобильные телефоны, магнитолы могут все больше и больше. Это происходит в связи с укрупнением функций реализованных в микросхемах. Очень многие считают, что обычному радиолюбителю недоступна современная электронная база. Мы стараемся доказать, что это вовсе не так. С современными методами работы с микроконтроллерами мы уже знакомили в предыдущих номерах (и будем продолжать знакомить), а этой статьей я постараюсь раскрыть общие вопросы и прояснить азы работы с программируемыми логическими матрицами (далее ПЛМ).

Программируемые матрицы появились достаточно давно, изначально они представляли собой матрицу из булевых функций (обычно конъюнкций) с сеткой связей. Входы имели пережигаемые перемычки, которые при программировании пережигались, и можно было реализовывать достаточно сложные булевы функции. Такие матрицы имели небольшую емкость, но зато давали возможность по упрощению схемы там, где было ограничение по размеру. Эти программируемые матрицы называются «классическими», за рубежом известны серии PAL(однократно программируемые) и GAL(многократно программируемые), у нас производились серии 556рт и 1556рт (однократно программируемые). При подготовке программы для прошивки используются простенькие компиляторы, как правило, описывающие зависимость выхода, как булеву функцию входа. Некоторые модификации имеют регистры, несколько уровней обратных связей.

Затем появились более сложные ПЛМ, где вместо конъюнкции используется макроблок (LAB). Он имеет сложную схему, включающую в себя регистр, блок булевых функций, цепи переноса. Каждый пользовательский вывод микросхемы (pin) также связан с периферийным блоком, который позволяет использовать его в различных режимах. Также расположена сетка межблочных соединений, которые позволяют строить схемы с многоуровневыми обратными связями. Более подробно можно прочитать на сайтах производителей и в специальной литературе посвященной ПЛМ (смотрите список рекомендуемой литературы). В моей статье мы будем учиться работать именно с такими ПЛМ.

Наиболее известные производители:
Altera
Lattice
Atmel
Xilinx

В своей статье я буду рассматривать работу с ПЛМ серии EPM7xxx (MAX7000) фирмы Altera и серии ATF15xx фирмы ATMEL. ПЛМ выпускаются в различных корпусах: PLCC (рекомендую использовать для начинающих, так как для таких корпусов есть удобные панельки с шагом контактов 2.54 – т.е. их удобно использовать на макетных платах), QFP (для миниатюрных любительских устройств) и BGA (для промышленных устройств). ПЛМ оснащены многократно программируемым электрически стираемым ПЗУ (оно используется для загрузки микросхемы после включения питания).

Первый шаг с чего мы начнем, это освоение проектирования необходимой нам функции для реализации в ПЛМ. Для этого необходим специальный программный пакет (САПР). Как правило, фирмы продают САПР за большие деньги, но для начальных серий ПЛМ (а рассматриваемые серии начальные в своем классе) эти пакеты прилагаются бесплатно (т.е. ограничения на использование САПР при разработке функций для таких ПЛМ не работают). Для разработки функций используются следующие САПР:

Фирма	Наименование	Интернет адрес
Altera	MaxPlus BaseLine	http://www.altera.com
Altera	Quartrus	http://www.altera.com
Xilinx	ISE	http://www.xilinx.com

Я рекомендую использовать начинающим MaxPlus BaseLine (на данный момент доступна для скачивания версия 10.2). Сразу предупрежу, все подобные пакеты имеют только англоязычный интерфейс, поэтому минимальное умение понимать язык Вам потребуется. Обучать работе с пакетом я не буду, все это достаточно подробно написано в специальной литературе и в файле помощи по пакету. Замечу только, что САПР по своей идеологии не сильно отличается от программных оболочек для программирования. Описывать функцию можно различными способами:

Визуальный – в этом случае схему рисуем из базовых булевых библиотечных элементов и соединений;

Визуальный с использованием серии 74xx – этот способ удобен для быстрого переноса уже используемых схем для упаковки в ПЛМ. В этом случае, схема состоит из библиотеки элементов 74-серии (в отечественной маркировке 155 серия);

На языке описания схемы AHDL (Altera Hardware Language) – наиболее эффективный для построения схем для ПЛМ небольшой емкости. Как видно из названия языка он учитывает специфику архитектуры ПЛМ фирмы Altera;

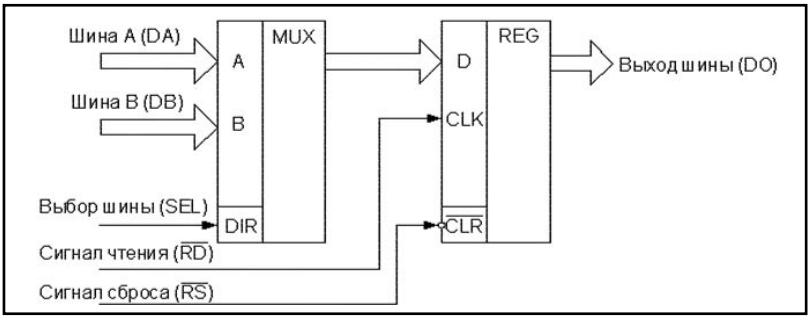
На языке VHDL или Verilog — эти языки стали стандартными и являются одинаковыми для любых САПР. Для начинающего язык достаточно сложный, имеет развитую библиотеку базовых функций, но зато разработки на этом языке позволяют достичь большей переносимости на другие типы матриц и даже для изготовителей микросхем.

В своих разработках я в основном использую AHDL, потому что мне он позволяет наиболее быстро разработать необходимую функцию и использовать полностью ресурсы ПЛМ. В общем случае описание функции (назовем ее программой) должна иметь следующую структуру:

```
TITLE "<наименование проекта>";
%секция описания включаемых файлов%
INCLUDE "<имя файла>";
...
%секция описания констант %
CONSTANT <имя константы>=<выражение>;
...
%секция описания макросов %
DEFINE <макрос>=<выражение макроса>;
...
%секция описания используемых выводов%
SUBDESIGN <имя>
(
    <имя вывода>:<назначение>[=<состояние>];
    ...
)
%секция описания переменных %
VARIABLE
    <имя переменной>:<тип>;
    ...
BEGIN
    % описание значений по умолчанию %
    DEFAULTS
            <переменная>=<значение по умолчанию>;
            ...
    END DEFAULTS;
    % описание функций %
    ...
END;
```

Как видно из структуры, проект напоминает некий язык программирования. По сути, это и есть так, только в нашем случае мы программируем функцию как зависимость выходов от состояния входов и внутренних переменных. Внутренними переменными могут являться регистры и защелки различного типа, а также просто промежуточные значения. Скобками комментария является символ – «%». Фунции могут просто быть арифметические или логические выражения, а также конструкции IF-ELSE и конструкции CASE и другие языковые конструкции. Моей целью не является обучение Вас программированию на языке AHDL, поэтому я не буду описывать полностью язык, а только рассмотрю некий пример.

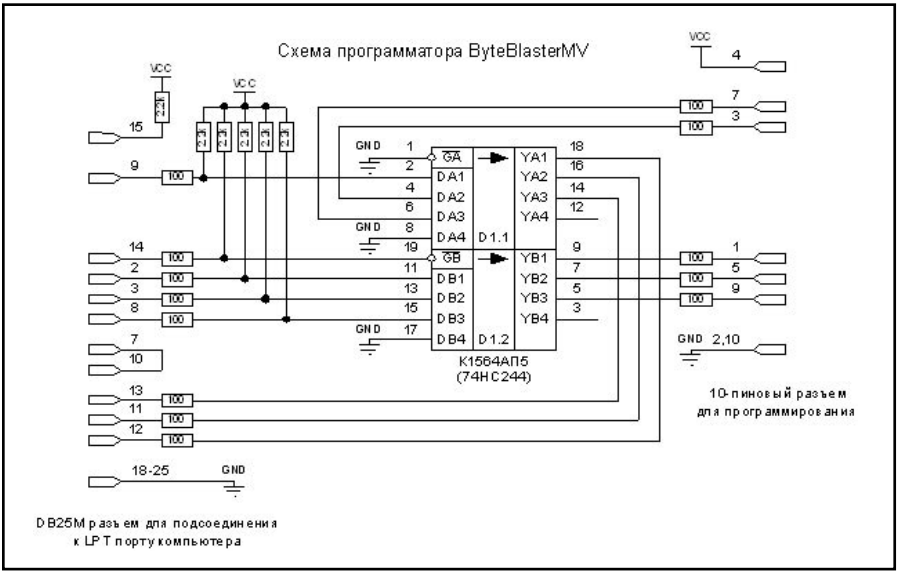
Задача: Сделать управляемый восьмибитный порт, который позволяет получать данные из одной шины или из другой в зависимости от управляющего сигнала, причем данные должны храниться до последующего чтения:



Опишем проект на языке AHDL:

```
TITLE "EXAMPLE";
SUBDESIGN Example
(
    DA[7..0]        :INPUT; % input bus A %
    DB[7..0]        :INPUT; % input bus B %

    SEL     :INPUT; % select bus %
    _RD     :INPUT; % read %
    _RS             :INPUT; % reset device %

    DO[7..0]        :OUTPUT; % output bus %
)
VARIABLE
    RG[7..0]        :DFF; % 8bit D-type register %
BEGIN
    DEFAULTS
            RG[] = 0;
    END DEFAULTS;

    % bus selecting %
    IF( SEL == GND ) THEN
            RG[].d = DA[];
    ELSE
            RG[].d = DB[];
    END IF;

    % register logic: latch data on front %
    RG[].clk = NOT(_RD);

    % reset logic %
    RG[].clrn = _RS;

    % output %
    DO[] = RG[].q;
END;
```

Как видите, проект оказался достаточно простым, при своей функциональной сложности. Вести проект можно с помощью встроенного редактора – пункт меню «MaxPlus/Text Editor». Сохраните проект в файл с расширением TDF. С помощью пункта меню «File/Project/Name» выберите сохраненный файл – теперь наш проект стал текущим. Перед компиляцией проекта выберем ПЛМ, в которую мы хотим упаковать нашу функцию – это возможно с помощью пункта меню «Assign/Device», в нашем случае выберем микросхему EPM7032SLC44. Теперь компилируем проект. После компиляции можно проанализировать временные характеристики «MaxPlus/Timing Analizer», симулировать работу схемы



для проверки правильности работы «MaxPlus/Simulator» (предварительно подготовив файл с входными данными с помощью «MaxPlus/WaveForm Editor»). Также можно посмотреть размещения функционала по макроблокам – пункт меню «MaxPlus/FloorPlan Editor».

По умолчанию при компиляции соответствие выводов микросхемы с сигналами нашей функции распределяются наиболее эффективно с точки зрения компилятора, но возможно (и в большинстве случаев используется) распределение выводов вручную. Это можно сделать с помощью пункта меню «Assign/Pin Location Chip». После распределения выводов микросхемы вручную обязательно перекомпилируйте проект.

В результате компиляции получается файл с расширением POF, предназначенный для программирования ПЛМ. Чтобы программировать ПЛМ нам понадобится программатор. Читая эти строки, Вы думаете, что это ужасно дорогостоящее устройство, которое сложно собрать самому. К счастью, это не так, и программатор при своей огромной функциональности (он еще к тому же поддерживает возможность программирования ПЛМ непосредственно на плате – режим in system programming), состоит всего из одной микросхемы и чуть больше десятка резисторов. Схема программатора приводится выше.

Питание зависит от типа ПЛМ и равняется напряжению питания микросхемы (данная схема программатора поддерживает напряжение питания 3.3 вольт и 5 вольт).

Таблица соответствия контактов разъема программирования выводам программирования ПЛМ:

Вывод	Режим PS		Режим JTAG	
	Имя сигнала	Описание	Имя сигнала	Описание
1	DCLK	тактирующая частота	TCK	тактирующая частота
2	GND	земля	GND	Земля
3	CONF_DONE	контроль конфигурирования	TDO	данные от устройства
4	VCC	Питание	VCC	Питание
5	nCONFIG	контроль конфигурирования	TMS	контроль статуса JTAG
6	-	не используется	-	не используется
7	nSTATUS	статус конфигурирования	-	не используется
8	-	не используется	-	не используется
9	DATA0	данные к устройству	TDI	данные к устройству
10	GND	земля	GND	земля

Для программирования микросхемы достаточно подсоединить ее выводы к соответствующим выводам разъема программирования (смотрите спецификацию микросхемы — datasheet, в которой описаны выводы ПЛМ). Для этого нужно предусмотреть ответную часть разъема программирования на плате вашего устройства или изготовить соответствующую насадку. Замечу, что у нас также можно приобрести программатор (с колодками для микросхем в корпусах PLCC84 и PLCC44). Программатор пригодится не только для программирования ПЛМ, но и для программирования микроконтроллеров и прочих микросхем, поддерживающих режим программирования JTAG.

После подсоединения микросхемы к программатору, запрограммируем наш проект. Это делается с помощью пункта меню «MaxPlus/Programmer». Появится диалог, в котором Вам нужно выбрать тип программатора, в нашем случае – «ByteBlaster(MV)». Если все сделано правильно, то микросхема успешно запрограммируется. На этом наш проект готов и Вы можете считать, что постигли самые азы.

Если Вы желаете использовать более дешевые ПЛМ фирмы Atmel серии ATF15xx, которые являются аналогами микросхем серии EPM7xxx фирмы Altera, то Вам понадобятся дополнительные утилиты с сайта фирмы Atmel (www.atmel.com):

Конвертер POF2JED – для конвертирования файла программирования из стандарта Altera в стандарт принятый фирмой Atmel;

Утилита программирования AtmelISP – для программирования ПЛМ.

Достаточно выбрать по таблице соответствующую микросхему фирмы Atmel. Затем с помощью конвертера POF2JED конвертировать Ваш проект в формат представления Atmel и с помощью утилиты AtmelISP запрограммировать ПЛМ. Таблицу соответствия ПЛМ можно также скачать с сайта фирмы Atmel.

Мы с Вами освоили ПЛМ начального уровня, которые наиболее уместны в разработке любительских устройств. Современные ПЛМ основаны как на статическом типе памяти, так и на ПЗУ и имеют очень большую емкость (т.е. большое количество макроблоков). Также они имеют различную структуру, существуют аналоговые версии ПЛМ, позволяющие собирать аналоговые схемы. На современном уровне ПЛМ применяются для подготовки производства и отладки микросхем высокой интеграции. Также многие ПЛМ включают в себя различные функциональные блоки (память, процессор, периферийные блоки …), которые позволяют создавать уникальные системы на чипе (SOS – system on chip).

Для современных тенденций разработки характерно уменьшение использования микросхем простейшей логики и реализация схем в ПЛМ. Почему это происходит, потому что с помощью ПЛМ достигаются следующие преимущества по сравнению с обычной логикой:

- Минимизация устройства;
- Изменение конфигурации или функциональности устройства без доработки (т.е. без применения пайки);
- Увеличение скорости работы устройств (самые медленные это межчиповые соединения);
- Увеличение надежности устройства (чем меньше паяных соединений, тем выше надежность устройства).

Поэтому современный разработчик обязан иметь в своем арсенале умение разрабатывать устройства на ПЛМ. Надеюсь, я Вам помог преодолеть Ваши заблуждения и начать полноценно осваивать технологию.

Рекомендуемая литература:

«Системы на микроконтроллерах и БИС программируемой логики» Бородин В.Б., Калинин А.В. ЭКОМ. Москва 2002.

«Плис фирмы Altera: элементная база, система проектирования и язык описания аппаратуры» В.Б.Стешенко. Додека. Москва 2002.

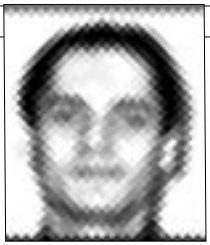
«Язык описания цифровых устройств AlteraHDL». А.П.Антонов. РадиоСофт. Москва 2002.

«Системы автоматизированного проектирования фирмы Altera MaxPlus II и Quartrus II». Д.А.Комолов, Р.А.Мяльк, А.А.Зобенко, А.С. Филиппов. РадиоСофт. Москва 2002.

«Языки VHDL и Verilog в проектировании цифровой аппаратуры». А.К.Поляков. Солон-пресс. Москва 2003.

РАЗВИТИЕ ТРОИЧНОЙ ТЕХНИКИ

автор Mac Buster



Для того, чтобы приблизить начало массового применения уравновешенного троичного кода и троичной логики в области вычислительной техники, я считаю необходимым сделать следующее:

1. Создание полноценных эмуляторов троичных машин «Сетунь» и «Сетунь-70»;
2. Создание симулятора троичной машины с архитектурой RISC;
3. Создание виртуальной троичной машины;
4. Разработка языков программирования для троичных машин;
5. Исследование эффективности троичных алгоритмов;
6. Разработка троичной аппаратуры;
7. Разработка многопользовательской многозадачной ОС для троичных машин.

Каждый этап работ имеет свою особую ценность. Так, создание полноценных эмуляторов ЭВМ «Сетунь» и «Сетунь-70» позволит всем заинтересованным в троичных машинах и просто любопытствующим поработать с полным функциональным аналогом этих уникальных машин. Сравнить их эффективность и судить о работе с троичным кодом на основании своего собственного опыта, а не на домыслах, выдумках и странных легендах, почерпнутых из Интернета и сомнительных печатных изданий, не стесняющихся публиковать измышления малосведущих в деле «специалистов». Кроме этого, возможно, удастся найти и сохранить программное обеспечение этих машин, что представляет уже историческую ценность. А так же можно проводить конкурсы по программированию, вроде тех, что проводятся англичанами с использованием эмулятора ЭВМ Manchester Baby.

Разработка и создание симулятора собственной троичной ЭВМ и последующее создание троичной виртуальной машины, позволит выработать разумный минимум команд для работы с данными представленными в уравновешенном троичном коде, разобраться с троичной логикой, попробовать применить существующие двоичные алгоритмы обработки информации, сравнить их эффективность с троичной реализацией, а так же выработать собственные троичные алгоритмы.

Разработка языков и средств программирования для троичных машин так же весьма интересна. В процессе работы потребуется осмыслить очень большое количество реализаций алгоритмического и функционального программирования и, возможно, выработать свой уникальный подход, который позволит программисту использовать троичное представление данных и троичную логику максимально эффективно.

Разработка троичной аппаратуры способна серьезно приблизить нас к тому моменту, когда все накопленные при использовании виртуальной машины, симулятора и эмуляторов навыки можно будет применить на практике.

Уже сейчас производятся попытки создать необходимый минимум троичных логических элементов следуя определенному стандарту. В последнее время произошел заметный сдвиг в исследовательской работе над процессорами с полиморфной архитектурой (PCA) и магниторезистивной оперативной памятью (MRAM), которые можно рассматривать в качестве основных кандидатов для аппаратной реализации троичных элементов [см. «В мире науки» №11 2005]. Каждая ячейка магниторезистивной памяти может пребывать в одном из четырех состояний, что прекрасно подходит для троичной техники. Можно попытаться реализовать троичную оперативную память, где каждая ячейка будет принимать одно из следующих состояний: +1, 0, -1 и «не задано». Последнее состояние позволит упростить отладку программ и немного снизить сложность аппаратуры. Предположительно, при старте машины вся память может быть заполнения 4-м значением и при попытке прочитать данные из этой области будет происходить аварийное прерывание. Другими словами - в следствие допущенной в вашей программе ошибки вам не удастся использовать неверное значение.

Кроме разработки аппаратуры в текущее время идет работа над симулятором троичной ЭВМ с архитектурой максимально приближенной к RISC, предполагающей использование разумного минимального набора команд. На

основе этого симулятора впоследствии будет создана виртуальная машина, которая позволит работать со всеми ресурсами двоичной хост-машины (накопители, средства ввода-вывода и т.п.) через промежуточный слой абстракции. Сейчас обсуждается набор и синтаксис команд. Предполагается писать программы на языке низкого уровня, по синтаксису приближенному к «Программированию в содержательных обозначениях» или к RTL (Register Transfer Language), что почти одно и тоже. Троичная машина будет иметь следующую архитектуру:

1. Все данные представлены в памяти в уравновешенном троичном коде;
2. Оперативная память делится на три независимые области:
   - команды;
   - данные;
   - стек.
3. Девять 18-разрядных равноправных регистров общего назначения;
4. Три регистра особого назначения (счетчик команд – PC, указатель стека – SP, регистр флагов-признаков – CC).

Примерная система команд и их синтаксис приведены ниже. Следует учесть, что все арифметические операции производятся с учетом знака числа и состояния флага переноса. Так же, в связи с RISC-архитектурой, запись и чтение данных может производится только двумя специальными командами, все остальные команды оперируют только с содержимым регистров. Rn обозначает любой регистр общего назначения.

запись слова: m[address] = Rn
чтение слова: Rn = m[address]
пересылка слова: Rn = Rn
пересылка одного произвольного разряда: Rn.n = Rn.n

сложение: Rn = Rn + Rn
вычитание (сравнение): Rn = Rn – Rn
умножение: Rn = Rn × Rn
деление: Rn = Rn : Rn (целочисленное)
деление: Rn = Rn ÷ Rn (остаток)

сдвиг на разряд: << и >> (через флаг переноса, по одному разряду)

логика: tlog

останов: stop
пропуск: none (для выравнивания команд и данных по началу слова)

условный переход: if (x) goto address (где x = eq, gt, lt)
возврат: ret

В наборе команд присутствует всего одна логическая операция tlog (от английского ternary или trinary logic). Однако в силу своей троичной природы, она покрывает как все команды двоичной логики (NOT, AND, OR и XOR), так и все унарные, бинарные и тринарные команды троичной логики (MAX, MID, MIN, MASK, SHIFT UP, SHIFT DOWN и т.д.), которые будут определяться через макросы в специальном стандартном, подключаемом при компиляции исходного кода, файле. Команда условного перехода может осуществлять переход сразу по трем разным адресам. Если все три адреса одинаковы, то переход называется безусловным.

Приведенный набор команд базируется на элементарных операциях, (которые выступают в роли своеобразного микрокода), производящих действия над внутренними регистрами процессора x, y и z. Потом результат выполнения операций копируется в 18-разрядные регистры общего назначения троичной ЭВМ и становятся доступными программисту.

1. Обнулить x (**Clear**);
2. Инвертировать x (**Negate**);
3. Копировать x в y (**Copy**);
4. Копировать x в y с инвертированием (**CopyN**);
5. Сдвинуть x вправо (разделить на три), предварительно скопировав младший разряд во флаг переноса c, затем поместить в старший разряд сохраненное значения флага переноса c (**ShiftR**);
6. Сдвинуть x влево (умножить на три), предварительно скопировав старший разряд во флаг переноса c и затем поместить в младший разряд сохраненное значение флага переноса c (**ShiftL**);
7. Сложить x и y, поместив сумму в z (**Add**);
8. Вычесть x из y, поместив разность в z (**Subtract**);
9. Сравнить x и y (**Compare**);
10. Умножить x на y, поместив произведение в z (**Multiply**);
11. Разделить x на y, поместив целую часть частного в z, а остаток в x (**Divide**).

В следующем выпуске я постараюсь подробно описать выполнение основных арифметических операций с числами, представленными в уравновешенном троичном коде.

Если у вас возникли вопросы или предложения по реализации всей программы работ или по какой-то ее части, вы можете высказать их в форуме NedoPC. Я буду очень рад ознакомиться с вашим мнением. ▲

Троичные элементы

автор Shaos

Если вы еще не представляете что такое троичное вычисление, то прежде чем двигаться дальше я бы посоветовал почитать статьи, на которые есть ссылки на нашем сайте http://www.ternary.info

Немного базовой информации – троичное вычисление, в отличие от двоичного имеет еще одно состояние -1 (в дополнение к 0 и 1). Для обозначения уровней сигналов мы примем распространненную систему, где буква N обозначает -1, буква O обозначает 0, а буква P обозначает +1.

Для некоторой унификации и стандартизации троичных элементов предлагается набор соглашений о входах и выходах троичных схем с 12 или 24 контактами и двуполярным питанием (-5В и +5В). Компоненты будем обозначать TRInn, где nn – две десятичные цифры, первая из которых будет обозначать категорию компонента:

TRI0n – программируемые элементы
TRI1n – унарные троичные функции
TRI2n – бинарные троичные функции
TRI3n – арифметические операции
TRI4n – селекторы и переключатели
TRI9n – элементы памяти

А вторая цифра будет считаться номером элемента внутри категории, причем если этот номер 0 (ноль), то будем считать что элемент имеет переключатели, которыми можно переключать его поведение вручную (в отличие от программируемых элементов, которые программируются дополнительными входными сигналами).

Пока мы не заостряем внимание на каких компонентах сделаны эти троичные элементы – на компараторах, электронных ключах, транзисторах или на оптоэлектронике – в любом случае требования по входам и выходам должны выполняться и элементы в общем случае должны быть совместимы.

ПРОГРАММИРУЕМЫЕ ТРОИЧНЫЕ ЭЛЕМЕНТЫ (TRI0N)

Программируемыми будем считать такие элементы, которые могут менять свое поведение в зависимости от внутреннего состояния, переключаемого частью входов элемента непосредственно в схеме.

TRI00 – универсальный 12-пиновый троичный элемент с 9 контактами, которые независимо друг от друга могут программироваться на вход или выход через внешний интерфейс к персональному компьютеру (или программатору):

1) IO1 – первый вход-выход
2) IO2 – второй вход-выход
3) IO3 – третий вход-выход
4) IO4 – четвертый вход-выход
5) IO5 – пятый вход-выход
6) IO6 – шестой вход-выход
7) IO7 – седьмой вход-выход
8) IO8 – восьмой вход-выход
9) IO9 – девятый вход-выход
10)-5В – отрицательное питание
11) 0В – земля
12)+5В – положительное питание

Такой элемент можно реализовать на каком-нибудь простом микроконтроллере и использовать для тестирования концепций или для замены запланированных, но еще неготовых троичных элементов.

TRI01 – строенный унарный универсальный элемент с программным управлением:

1) I1 – первый вход
2) O1 – первый выход
3) I2 – второй вход
4) O2 – второй выход
5) I3 – третий вход
6) O3 – третий выход
7) RN – результат если I=N
8) RO – результат если I=O
9) RP – результат если I=P
10)-5В – отрицательное питание
11) 0В – земля
12)+5В – положительное питание

Функция, выполняемая элементом, задается входами управления RN,RO,RP. Если на входе In появляется уровень N, то выход On принимает значение RN, если O, то RO, и если P, то на выход идет RP.

TRI02 – строенный бинарный универсальный элемент с программным управлением (имеет дополнительные 12 контактов):

1) A1 – первый аргумент первой функции
2) B1 – второй аргумент первой функции
3) C1 – результат первой функции
4) A2 – первый аргумент второй функции
5) B2 – второй аргумент второй функции
6) C2 – результат второй функции
7) A3 – первый аргумент третьей функции

8) B3 – второй аргумент третьей функции
9) C3 – результат третьей функции
10)-5B – отрицательное питание
11) 0B – земля
12)+5B – положительное питание
13) NN – результат если A=N и B=N
14) NO – результат если A=N и B=O
15) NP – результат если A=N и B=P
16) ON – результат если A=O и B=N
17) OO – результат если A=O и B=O
18) OP – результат если A=O и B=P
19) PN – результат если A=P и B=N
20) PO – результат если A=P и B=O
21) PP – результат если A=P и B=P
22)-5B – отрицательное питание
23) 0B – земля
24)+5B – положительное питание

У элемента 2 ряда 12-пиновых контактов. Выполняемая элементом функция задается входными сигналами от 13 до 21 (9 троичных сигналов управления дают 19683 варианта двоичных функций).

УНАРНЫЕ ТРОИЧНЫЕ ФУНКЦИИ (TRI1N)

TRI10 – счетверенный унарный универсальный элемент:

1) I1 – первый вход
2) O1 – первый выход
3) I2 – второй вход
4) O2 – второй выход
5) I3 – третий вход
6) O3 – третий выход
7) I4 – четвертый вход
8) O4 – четвертый выход
9) не используется
10)-5B – отрицательное питание
11) 0B – земля
12)+5B – положительное питание

Функция устанавливается тремя переключателями вручную.

TRI11 – счетверенный троичный буфер:

1) I1 – первый вход
2) O1 – первый выход
3) I2 – второй вход
4) O2 – второй выход
5) I3 – третий вход
6) O3 – третий выход
7) I4 – четвертый вход
8) O4 – четвертый выход
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI12 – счетверенный троичный инвертор:

1) I1 – первый вход
2) O1 – первый выход
3) I2 – второй вход
4) O2 – второй выход
5) I3 – третий вход
6) O3 – третий выход
7) I4 – четвертый вход
8) O4 – четвертый выход
9) не используется
10)-5B – отрицательное питание
11) 0B – земля
12)+5B – положительное питание

БИНАРНЫЕ ТРОИЧНЫЕ ФУНКЦИИ (TRI2N)

TRI20 – строенный бинарный универсальный элемент:

1) A1 – первый аргумент первой функции
2) B1 – второй аргумент первой функции
3) C1 – результат первой функции
4) A2 – первый аргумент второй функции
5) B2 – второй аргумент второй функции
6) C2 – результат второй функции
7) A3 – первый аргумент третьей функции
8) B3 – второй аргумент третьей функции
9) C3 – результат третьей функции
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

Функция устанавливается 9 переключателями вручную.

Стандартные компоненты, выполняющие фиксированные функции пока не унифицированы, но уже скоро ;)

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ (TRI3N)

TRI31 – два независимых полусумматора (A+B=S+C):

1) A1 – первое слагаемое первого полусумматора
2) B1 – второе слагаемое первого полусумматора
3) S1 – сумма от первого полусумматора
4) C1 – перенос (заем) от первого полусумматора
5) A2 – первое слагаемое второго полусумматора
6) B2 – второе слагаемое второго полусумматора
7) S2 – сумма от второго полусумматора
8) C2 – перенос (заем) от второго полусумматора
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI32 – один полный сумматор (C0+A+B=S+C1):

1) C0 – перенос (заем) от предыдущей цепи
2) A – первый аргумент
3) B – второй аргумент
4) S – сумма
5) C – перенос (заем)
6) не используется
7) не используется
8) не используется
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI33 – двухразрядный полный сумматор (C0+A1A2+B1B2=C1C2+C):

1) C0 – перенос (заем) от предыдущего разряда
2) A1 – первый аргумент в первом разряде
3) B1 – второй аргумент в первом разряде
4) S1 – сумма в первом разряде
5) A2 – первый аргумент во втором разряде
6) B2 – второй аргумент во втором разряде
7) S2 – сумма во втором разряде
8) C – перенос (заем) для следующего разряда
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI34 – трехразрядная схема инкремента-декремента:

1) C0 – перенос (заем) от предыдущего разряда
2) I1 – первый вход
3) O1 – первый выход
4) I2 – второй вход
5) O2 – второй выход
6) I3 – третий вход
7) O3 – третий выход
8) C – перенос для следующего разряда
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI35 – сравнение двух четырехразрядных троичных чисел:

1) A1 – первый разряд первого троичного числа
2) B1 – первый разряд второго троичного числа
3) A2 – второй разряд первого троичного числа
4) B2 – второй разряд второго троичного числа
5) A3 – третий разряд первого троичного числа
6) B3 – третий разряд второго троичного числа
7) A4 – четвертый разряд первого троичного числа
8) B4 – четвертый разряд второго троичного числа
9) M – результат сравнения (P если A>B, O если A=B, N если A<B)
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

СЕЛЕКТОРЫ И ПЕРЕКЛЮЧАТЕЛИ (TRI4N)

TRI41 – сдвоенный троичный селектор аналоговых сигналов:

1) S – троичный вход управления
2) N1 – подключаемый контакт первого селектора, если S=N
3) O1 – подключаемый контакт первого селектора, если S=O
4) P1 – подключаемый контакт первого селектора, если S=P
5) C1 – подсоединяется к N1,O1,P1 в зависимости от значения S
6) N2 – подключаемый контакт второго селектора, если S=N
7) O2 – подключаемый контакт второго селектора, если S=O
8) P2 – подключаемый контакт второго селектора, если S=P
9) C2 – подсоединяется к N2,O2,P2 в зависимости от значения S
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

TRI42 – троичный селектор аналоговых сигналов с 2 входами управления:

1) SA – первый троичный вход управления
2) SB – второй троичный вход управления
3) CC – подсоединяется к контактам 13...21 в зависимости от значения S1S2
4) не используется
5) не используется
6) не используется
7) не используется
8) не используется
9) не используется
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание
13) NN – подключаемый контакт селектора при SA=N и SB=N
14) NO – подключаемый контакт селектора при SA=N и SB=O
15) NP – подключаемый контакт селектора при SA=N и SB=P
16) ON – подключаемый контакт селектора при SA=O и SB=N
17) OO – подключаемый контакт селектора при SA=O и SB=O
18) OP – подключаемый контакт селектора при SA=O и SB=P
19) PN – подключаемый контакт селектора при SA=P и SB=N
20) PO – подключаемый контакт селектора при SA=P и SB=O
21) PP – подключаемый контакт селектора при SA=P и SB=P
10) -5B – отрицательное питание
11) 0B – земля
12) +5B – положительное питание

У элемента 2 ряда 12-пиновых контактов. Контакты от 4 до 9 не используются.

ЭЛЕМЕНТЫ ПАМЯТИ (TRI9N)

TRI91 – четырехтритный регистр:

1) I1 – первый вход
2) O1 – первый выход сохраненных данных
3) I2 – второй вход
4) O2 – второй выход сохраненных данных
5) I3 – третий вход
6) O3 – третий выход сохраненных данных
7) I4 – четвертый вход

8) O4 – четвертый выход сохраненных данных
9) W – вход управления (запись при W=P, иначе – сохранение)
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI92 – четырехтритный регистр с Z-состоянием:

1) I1 – первый вход
2) O1 – первый выход сохраненных данных
3) I2 – второй вход
4) O2 – второй выход сохраненных данных
5) I3 – третий вход
6) O3 – третий выход сохраненных данных
7) I4 – четвертый вход
8) O4 – четвертый выход сохраненных данных
9) C – вход управления (запись при C=P, хранение при C=O, вывод при C=N)
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI93 – две ячейки памяти с независимым управлением:

1) I1 – первый вход данных
2) C1 – первый вход управления (запись при C1=P, хранение при C1=O, вывод при C1=N)
3) O1 – первый выход сохраненных данных
4) I2 – второй вход данных
5) C2 – второй вход управления (запись при C2=P, хранение при C2=O, вывод при C2=N)
6) O2 – второй выход сохраненных данных
7) I3 – третий вход данных
8) C3 – третий вход управления (запись при C3=P, хранение при C3=O, вывод при C3=N)
9) O3 – третий выход сохраненных данных
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI94 – четырехтритный регистр со сбросом:

1) I1 – первый вход
2) O1 – первый выход сохраненных данных
3) I2 – второй вход
4) O2 – второй выход сохраненных данных
5) I3 – третий вход
6) O3 – третий выход сохраненных данных
7) I4 – четвертый вход
8) O4 – четвертый выход сохраненных данных
9) C – вход управления (запись при C=P, хранение и вывод при C=O, сброс при C=N)
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI95 – три ячейки памяти с независимым управлением и сбросом:

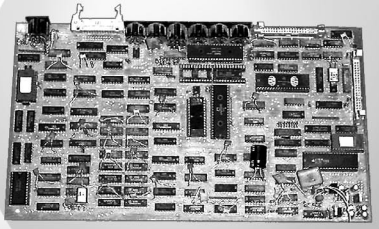
1) I1 – первый вход данных
2) C1 – первый вход управления (запись при C1=P, хранение и вывод при C1=O, сброс при C1=N)
3) O1 – первый выход сохраненных данных
4) I2 – второй вход данных
5) C2 – второй вход управления (запись при C2=P, хранение и вывод при C2=O, сброс при C2=N)
6) O2 – второй выход сохраненных данных
7) I3 – третий вход данных
8) C3 – третий вход управления (запись при C3=P, хранение и вывод при C3=O, сброс при C3=N)
9) O3 – третий выход сохраненных данных
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI96 – две ячейки с независимым управлением и двумя входами:

1) A1 – первый вход первой ячейки
2) B1 – второй вход первой ячейки
3) C1 – вход упр.1 (запись из A1 при C1=P, хранение и вывод при C1=O, запись из B1 при C1=N)
4) O1 – выход сохраненных данных первой ячейки
5) A2 – первый вход второй ячейки
6) B2 – второй вход второй ячейки
7) C2 – вход упр.2 (запись из A2 при C2=P, хранение и вывод при C2=O, запись из B2 при C2=N)
8) O2 – выход сохраненных данных второй ячейки
9) не используется
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание

TRI97 – трехтритный регистр с инкретементом/декрементом и сбросом:

1) I1 – вход первого разряда
2) O1 – выход первого разряда
3) I2 – вход второго разряда
4) O2 – выход второго разряда
5) I3 – вход третьего разряда
6) O3 – выход третьего разряда
7) C – выход переноса для следующего регистра
8) C1 – управление 1 (”P” – инкремент, “O” – хранение, “N” – декремент)
9) C2 – управление 2 (”P” – запись, “O” – хранение, “N” – сброс)
10) -5В – отрицательное питание
11) 0В – земля
12) +5В – положительное питание. ▲



АТМ Турбо от NedoPC

Позиция	Цена	Примечания
Плата голая Комплектность поставки: - плата; - ХЛ8 (прошитая); - описание (книжки); - софт (имиджи на сдром).	700	Платы при изготовлении электрически не проверяются
Плата собранная Комплектность поставки: - отлаженная плата с 1556ХЛ8, ПЗУ, Z80, 1818ВГ93, панелька под АУ; - описание (книжки); - софт (имиджи на сдром)	2950	Платы будут собраны и отлажены, т.е гарантировано рабочие. Музыкальный сопроцессор не поставляется
Конструктор для самостоятельной сборки: Комплектность поставки: - плата; - набор радиодеталей, необходимых для сборки; - необходимые ПЗУ (прошитые), ХЛ8; - описание (книжки); - софт (имиджи на сдром).	2300	Процессор, ОЗУ и ПЗУ проверяются перед отправкой. Поставщик не несет ответственности за сгоревшие в результате неправильной сборки или подключения детали
Комплект шлейфов: - шлейф FDD - шлейф IDE - шлейф COM - переходник AT-питание->5-DIN - шлейф для звука	200	Комплект облегчает установку в AT корпус
Музыкальный сопроцессор (YM2149)	160	
Прошивка ПЗУ основная (27С512 или аналог)	65	На текущий момент версия 1.07.13
Прошивка тест ОЗУ (27С512 или аналог)	65	На текущий момент версия 1.02
Прошивка знакогенератора (573РФ2 или аналог)	35	Символьная таблица знакогенератора
Прошивка контроллера клавиатуры (573РФ2 или аналог)	35	XT или AT клавиатура
Процессор КР1858ВМ3	40	Аналог Z80 на 6МГц
Процессор Z0840008	60	Z80 на 8МГц
Дополнительная, прошитая КР1556ХЛ8	40	Прошивка АТМ Турбо 2+

К общей стоимости заказа добавляется цена доставки (по России):

130 руб. для голой платы

200 руб. для платы в сборе

Порядок обработки заказов на платы:

1. Присылаем e-mail Чунину Роману Валерьевичу (chunin@mail.ru) или звоним по телефону +7(095)6547433, для выяснения есть ли свободные платы и согласования цены, комплектации;
2. Если не получен почтовый перевод в течении двух недель, то заявка снимается;
3. Дополнительно к стоимости добавляется стоимость услуг почты: 130 руб. для голой платы, 200 руб. для собранной;
4. Осуществляем почтовый перевод на адрес: 109451, Москва, ул.Братиславская, д.13, кор.1, кв.228, Чунину Роману Валерьевичу. Пожалуйста, указывайте обратный адрес (если через e-mail, то с указанием номера и даты почтового перевода);
5. После обработки и подготовки заказ отсылается по указанному вами адресу. Голые платы отсылаются в течении одной недели после оплаты, собранные в порядке очереди на сборку (на сборку одной платы уходит примерно неделя, собирать будут два человека параллельно);
6. Весь процесс оформления заказов будет отображаться на сайте http://nedopc.com/products.php

ОБЪЯВЛЕНИЯ

Куплю GS 512K за 1500-2000 руб. в Питере. Основное условие, чтобы рабочая была. Предложения на мэйл muchkin@mail.ru или на аську 8087845. Торг уместен.

Приму нерабочее и ненужное "железо". 109451, город Москва, ул.Братиславская, д.13, корп.1, кв.228, Чунину Роману Валерьевичу.

Просьба к читателям: сообщать об ошибках и опечатках, найденных на страницах издания по адресу nedopc@mail.ru

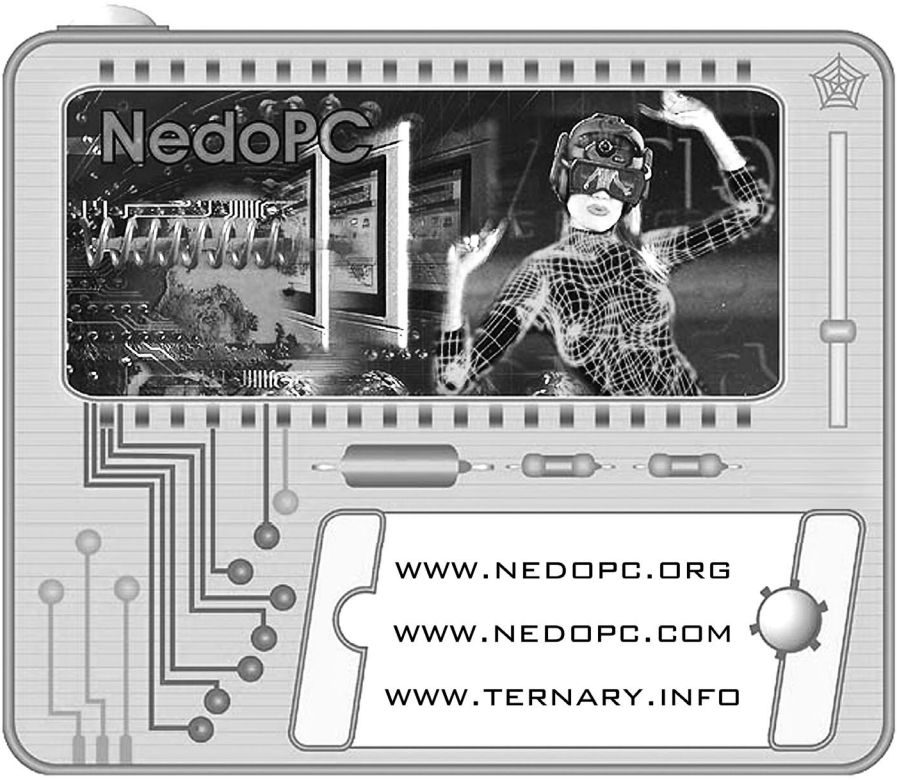
ЗДЕСЬ МОГЛА БЫ БЫТЬ ВАША РЕКЛАМА

NedoPC is not PC

NedoPC

Бланк Объявления
в журнал "NedoPC"

Бланк для размещения объявлений рассчитан на тех, кто предпочитает пользоваться обычной почтой.
Адрес: 109451, Москва, ул.Братиславская, д.13, кор.1, кв.228, Чунину Роману Валерьевичу
Для тех же, кто желает отправить текст объявления через e-mail, пишите на адрес nedopc@mail.ru
Текст объявления должен содержать не больше 256 символов



ВСЕГДА В ON-LINE!