Архитектура и программирование Dendy.

Игровая приставка NES (Nintendo Entertainment System) - версия для Америки, известная также как Famicom (FAMily COMputer) - версия для Японии, в нашей стране, благодаря фирме Стиплер (Steepler), известна как Dendy. И не смотря на то, что в середине девяностых годов прошлого века Денди была действительно «народной» приставкой – со стремительным ростом технического прогресса она сейчас, как игровая система, отошла далеко-далеко на задний план (уступив место ПК и другим игровым консолям). Но именно в сегодняшние дни Денди более интересна не геймерам, а радиолюбителям – как простейшая микро-ЭВМ …

Вопросом программирования Денди заинтересовался и я.

В начале изучения и создания материала, который представлен вам ниже, а именно в 2003 году - разнообразной информацией Интернет не изобиловал (хотя кое-что и было) – вся документация англоязычная (хотя встречались и ее прямые переводы – безо всякого осмысления переводимого). Тогда же вышла даже книга и мультимедийный диск (разных издательств, но одинакового содержания – книга даже более полного) – в которых легко узнавались все теже англоязычные источники (а часть информации взята «с потолка» и абсолютно некорректна). Поэтому многое пришлось осмыслять на практике (весь текст исключительно мой, авторский).

Вторая итерация и второе существенное дополнение и переработка данного материала была в 2007 году … Всего и не вспомнить, некоторые несущественные дополнения и корректировки были и до и после. Также благодаря отзывам читателей были исправлены некоторые недочеты, однако не все изменения сразу публиковались на сайте.

И вот в 2016 году текст очередной, что называется «обновленной и дополненной», редакции перед вами …

Материал ориентирован на читателя, обладающего базовыми сведениями по организации ЭВМ.

Общие сведения об архитектуре Денди.

Игровая приставка Денди является микро-ЭВМ для домашнего (игрового) применения (далее речь пойдет именно о Денди – как наиболее распространенной вариации консоли NES, а вернее ее японской версии Famicom, в России), ориентированная на использование телевизора в качестве дисплея. Подключение к телевизору может осуществляться одним из двух способов – «по низкой частоте» (двумя кабелями – звук и композитный видео) или «по высокой частоте» (при помощи «антенного» кабеля). Китайские совместимые поделки, представленные в нашей стране в большом разнообразии – архитектурно совместимы с Денди, однако зачастую имеют ряд схемотехнических «упрощений», не позволяющих полностью реализовать функционал оригинальных консолей (и даже стиплеровской «Денди»).

Денди и прочие клоны консолей NES/Famicom (как в общем-то и их оригиналы) сконструированы на базе микропроцессора (**CPU**), совместимого с **MSC6502** (микросхемы: RP2A03, RP2A07, UM6527, HA6827 и пр.) и видеопроцессора (**PPU**) (микросхемы: RP2C02, RP2C07, UM6538, UM6528, HA6838 и пр.), которые работают в тесном взаимодействии (такая вот «двухпроцессорная» система). Пары CPU + PPU подбираются с учетом формата формируемого кадра – NTSC или PAL. У каждого процессора своё адресное пространство и своя оперативная память (не путать с адресным пространством). Адресные пространства CPU и PPU **не пересекаются**. Микросхема CPU, применяемая в Денди, имеет встроенный звуковой сопроцессор – pAPU (чего нет у базового процессора 6502), но при этом отсутствует блок

двоично-десятичной арифметики. Помимо CPU и PPU, совместно со статической оперативной памятью, работу приставки обеспечивают так же прочие «мелкие» компоненты (регистры, логические элементы), сопрягающие работу всей системы. Архитектура Денди предусматривает хранение (и исполнение) программ со сменных модулей – «картриджей». Об архитектуре картриджа см. ниже – здесь лишь скажем, что часть картриджа адресует CPU, а часть PPU. Изначально плата приставки Денди содержала несколько корпусов микросхем (CPU и память, PPU и память, регистры логические элементы и прочее …) – как и оригинальные консоли. Последние ревизии моделей Денди (1996 год) собраны на заказной микросхеме (**UM6561A**) – которая совмещает на одном кристалле все «микросхемы» приставки (SoC – System on Chip). В ревизиях моделей Денди более ранних версий можно встретить «гибридный вариант» - память (оперативную и видео) установленную в виде двух отдельных микросхем (UM6516, UM6116 – аналог КР537РУ10), видимо, в центральный «SoC» (xx1818) её нет.

Вне зависимости от элементной базы, структурная схема Денди имеет вид, показанный на рисунке 1.

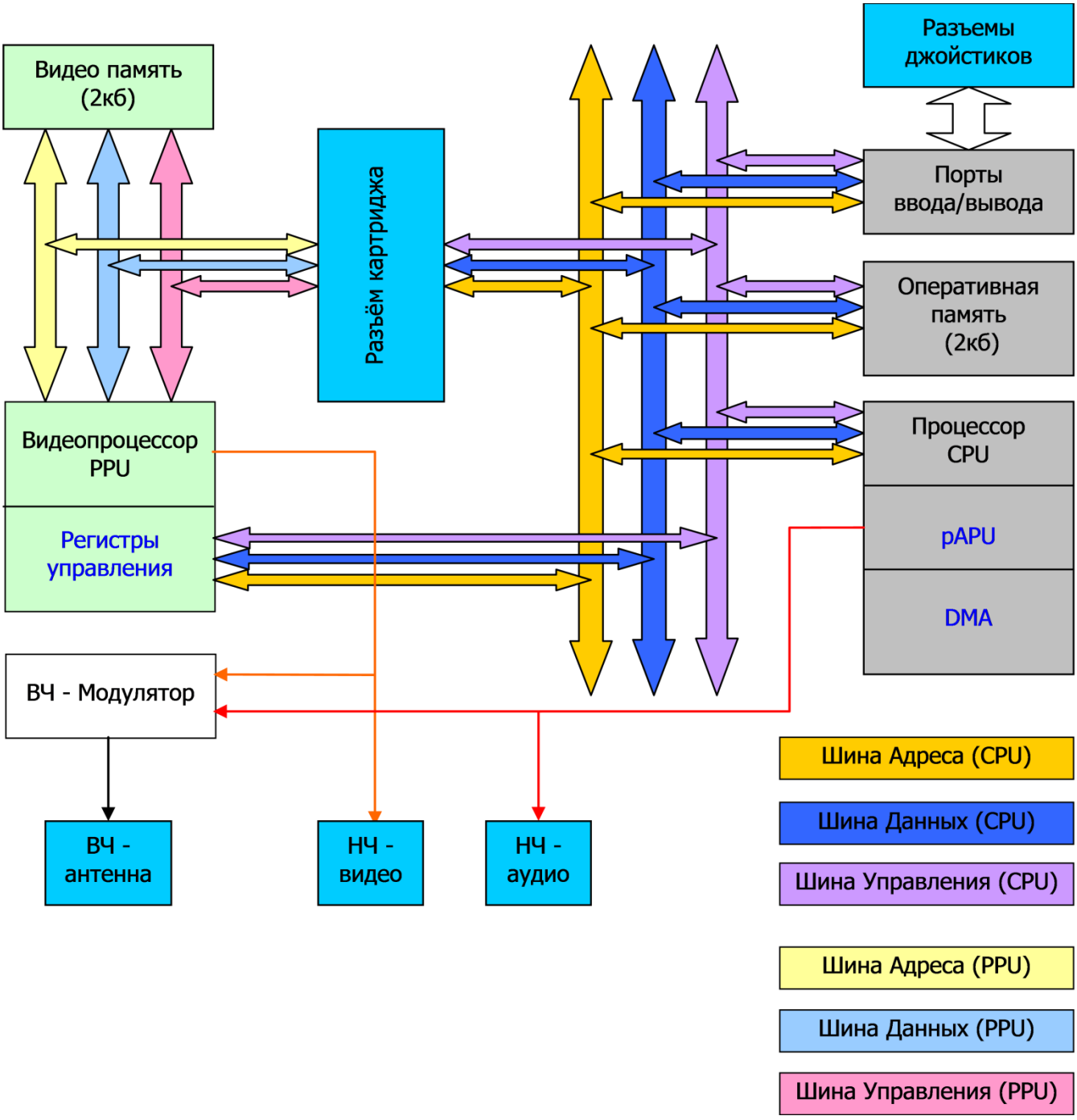


Рисунок 1.

Процессор Денди (CPU).

Как отмечалось ранее, первая (многокорпусная) модель NES/Famicom строилась на базе процессора Ricoh RP2A03 (архитектура 6502) или его аналога. Все последующие модели сверху вниз совместимы с ней. Не исключено, что SoC UM6561 более совершенен, имеет расширенную систему команд, а возможно и другие "фичи" (которые, надо понимать, в дань совместимости, как правило, не используются). Поэтому, говоря о процессоре Денди, – опираемся на базовую архитектуру 6502, реализованную в «классическом» RP2A03 (универсальном), или RP2A07 - только для варианта PAL.

Итак - CPU 6502 имеет:

- Шина Адреса – 16 бит (адресное пространство 64k);
- Шина Данных – 8 бит;
- Эффективная частота – 1.78 МГц;
- Встроенный звуковой сопроцессор «pAPU» (пятиканальный: 4 «аналоговых» + 1 «цифровой» DMC-канал);
- Контроллер DMA;

Подробнее о системе команд 6502 можно прочитать в приложении 1. Примеры принципиальных схем и их описания - в приложении 5.

Здесь же мы рассмотрим, как этот процессор используется именно в Денди.

Распределение адресного пространства.

Модель 6502 предусматривает единое адресное пространство кода и данных (принстонская архитектура). Также в процессоре отсутствуют инструкции обращения к портам ввода/вывода (типа in/out) – порты должны отражаться на адресное пространство памяти. Т.е. адресное пространство процессора делится между портами и памятью. В архитектуре Денди это происходит так (кстати, по-моему, не очень рационально):

Адрес	**Размер**	**Назначение**
$0000-$07FF	2k	**RAM**
$0800-$1FFF	6k	RAM Mirror (x3)
$2000-$2007	$8 = 8 байт	**Registers Video**
$2008-$3FFF	$1FF8 = 8184 байта	Registers Video Mirror (x1023)
$4000-$4017	$20 = 32 байта	**Registers Audio & DMA & I/O**
$4018-$4FFF	$0FE8 = 4072 байта	Not used
$5000-$5FFF	4k	**Expansion ROM\RAM** (etc. in MMC5)
$6000-$7FFF	8k	**SRAM (aka WRAM)** (etc. in MMC3)
$8000-$BFFF	16k	**PRG-ROM** (1)
$C000-$FFFF	16k	**PRG-ROM** (0)

Таблица 1.

RAM – Оперативная память (на плате приставки или внутри SoC) объемом 2k – её распределение следующее:

- $0000-$00FF – “Нулевая страница” (Zero Page) – используется для системных целей (для пользовательских программ предпочтительны адреса $10-$9F, прочие лучше не трогать). Процессор имеет отдельные инструкции, обеспечивающие быстрый доступ к нулевой странице. Таким образом, с учетом малого количества регистров в самом процессоре, нулевая страница может выполнять роль «регистрового банка».

- $0100-$01FF – Аппаратный стек от старших адресов (размер, понятно, 256 байт).
- $0200-$07FF – Пользовательская область оперативной памяти, «не много не мало» а всего 1.5k.

Начиная с адреса $0800 из-за неполной дешифрации адреса, идут три «зеркала» RAM. Т.е. обращение к ячейке $0000 эквивалентно обращению к $0800 или к $1000 или к $1800. (Интересно, что мешало разработчикам установить еще 6 килобайт памяти – лишнего наверняка бы не было).

Registers Video [8 шт.] – через эти регистры видеопроцессора (и только через эти регистры) осуществляется связь центрального процессора с видеопроцессором, а через него и с видеопамятью. Подробнее см. главу «Видеопроцессор Денди (PPU)». Далее с адреса $2008 идут «зеркала» регистров видеопроцессора.

Registers Audio & DMA & I/O [24 шт.] – регистры звукового сопроцессора, контроллера DMA и портов ввода/вывода. Подробнее см. соответствующие главы. Назначение остальных ($0FE8=4072 шт.) адресов не определено, но может использоваться внешними звуковыми процессорами, располагающимися на картридже.

Expansion ROM\RAM [4k] – расширение ПЗУ\ОЗУ (или любое другое, располагающееся на картридже) – используется, например, маппером MMC5 (см. главу «Картридж Денди»)

SRAM (она же **WRAM**) [8k] – статическое ОЗУ, которое также может находиться на картридже. Являясь полноценным ОЗУ для процессора, может использоваться для любых целей, в том числе и для хранения (исполнения) программы. Как правило, в этой области игры хранят данные состояния текущей игровой локации в процессе игры (маппер MMC3). При необходимости сохранения игровых ситуаций, эта микросхема питается от батарейки, расположенной на картридже (сохранение игры на время выключения питания, с возможностью последующего продолжения).

PRG-ROM [32k] – зачастую представлено 2 окнами по 16k (иногда 4 по 8k или другой вариант логического разбиения - в зависимости от маппера, для более гибкого варьирования страничной адресацией памяти картриджа, см. главу «Картридж Денди»). Чтобы подробно рассмотреть механизмы эффективного использования этого участка памяти – необходимо иметь представление об архитектуре конкретного картриджа (а вариантов тут много), речь о них пойдет позже. Самая простая архитектура картриджа маппера не содержит, в картридже есть лишь две ПЗУ размером 32k и 8k, подключаются они к шинам процессора (CPU) и видеопроцессора (PPU) соответственно (см. рис.1). Первая (в нашем примере 32k) микросхема ПЗУ функционально называется PRG-ROM, и содержит в себе программу и возможно данные. Первые 16k ПЗУ отображаются в окне 1 ($8000-$BFFF), последние 16k в окне 0 ($C000-$FFFF). Организовано это на схемотехническом уровне. В конце нулевого окна располагаются вектора прерываний процессора.

Система прерываний.

Архитектура 6502 предусматривает всего 3 прерывания. Рассмотрим подробнее случаи их возникновения, предварительно указав адреса в таблице векторов прерываний.

$FFFA – NMI (VBlink)

Немаскируемое прерывание. Поступает с видеопроцессора и сигнализирует о том, что PPU закончил прорисовку очередного кадра и «луч пошел обратно» (в начало экрана). Обработчик NMI должен производить в этот момент (ограниченное время) необходимые по программе обновления данных видеопамяти, т.к. во время прорисовки строк кадра видеопамять

недоступна (из нее осуществляет чтение PPU). Существует возможность программного отключения видеопроцессора – в этом режиме доступна запись в видеопамять (неограниченное время) прерывание NMI формироваться не будет, но также будет отсутствовать и картинка на экране (используется при начальной инициализации).

$FFFC – RESET

Происходит по факту включения питания или при нажатии на клавишу «Reset». Стоит помнить, что в последнем случае изменяется только программный счетчик, все регистры, а тем более память сохраняют свои значения.

$FFFE - IRQ/BRK

Единственное в системе «пользовательское» прерывание. Возникает по активному сигналу на соответствующем входе процессора, который выведен на разъём картриджа и порт расширений (или может вызываться программно – инструкция BRK). Таким образом вызывать обработчик данного прерывания может как периферия, расположенная на картридже (например, маппер), так и устройства, подключенные к порту расширений («разъем второго джойстика»).

Соответственно по указанным адресам (в «таблице векторов прерываний») должны находиться инструкции перехода к обработчикам соответствующих прерываний. Приоритет прерываний следующий (от более приоритетного к менее): **RESET, NMI, IRQ/BRK**.

Видеопроцессор Денди (PPU).

Видеопроцессор (PPU) ориентирован на генерацию картинки на выходе в одном из стандартов: для американской NES и японской Famicom – это NTSC; для российских Денди (и китайщины, продаваемой у нас) – это, как правило, PAL (в редких случаях SECAM – что не меняет формата размера картинки в сравнении с PAL). Далее рассматриваем именно вариант PAL.

PPU формирует «готовый» низкочастотный видеосигнал, который непосредственно (или, в зависимости от модели приставки, через каскад(ы) усилителя) подается на видеовыход Денди и на вход модулятора полного («антенного») сигнала.

Характеристики и возможности PPU:

- Шина Адреса – 14 бит (адресное пространство 16k) + 256 байт отдельно адресуемой памяти спрайтов;
- Шина Данных – 8 бит;
- Количество цветовых оттенков всего – 64;
- Одновременно на экране – до 25 (любые из 64), а именно: до 13-и для фона и до 12-и для спрайтов;
- Количество спрайтов на экране – 64 (не более 8-и на линии);
- Разрешение выводимой картинки (PAL/NTSC) – 256x240 / 256x224;
- Частота регенерации (PAL/NTSC) – 50Гц/60Гц;

Для программиста (и CPU) – видеопроцессор это 8 регистров, адресуемые CPU по адресам $2000-$2007 (в архитектуре Денди) – они обеспечивают возможность полного управления видеопроцессором. Помимо этого, запись в память спрайтов PPU может осуществлять так же и контроллер DMA (понятно, что без непосредственного участия процессора). Иных способов общения с PPU нет!

Адресное пространство PPU распределяется следующим образом:

Адрес	Размер	Назначение
$0000-$0FFF	4k	**CHR-ROM** Знакогенератор **0** (На картридже)
$1000-$1FFF	4k	**CHR-ROM** Знакогенератор **1** (На картридже)
$2000-$23BF	$3C0 = 960 байт	**VRAM** Экранная страница 1 – Символы (В приставке)
$23C0-$23FF	$40 = 64 байта	**VRAM** Экранная страница 1 – Атрибуты (В приставке)
$2400-$26BF	$3C0 = 960 байт	**VRAM** Экранная страница 2 – Символы (В приставке)
$27C0-$27FF	$40 = 64 байта	**VRAM** Экранная страница 2 – Атрибуты (В приставке)
$2800-$2BBF	$3C0 = 960 байт	**VRAM** Экранная страница 3 – Символы (На картридже)
$2BC0-$2BFF	$40 = 64 байта	**VRAM** Экранная страница 3 – Атрибуты (На картридже)
$2C00-$2FBF	$3C0 = 960 байт	**VRAM** Экранная страница 4 – Символы (На картридже)
$2FC0-$2FFF	$40 = 64 байта	**VRAM** Экранная страница 4 – Атрибуты (На картридже)
$3000-$3EFF	$F00 = 3840 байт	VRAM Mirror of $2000-2EFF
$3F00-$3F0F	$10 = 16 байт	**Палитра фона** (в регистрах PPU)
$3F10-$3F1F	$10 = 16 байт	**Палитра спрайтов** (в регистрах PPU)
$3F20-$3FFF	$E0 = 224 байт	Не используются

Регистры видеопроцессора выполняют следующие функции:

Регистр (адресация по CPU)	Биты	**Назначение**
$2000 w/o		**Управление видеопроцессором**
	7	Формирование запроса прерывания NMI при кадровом синхроимпульсе (0 - запрещено; 1 - разрешено)
	6	Не используется (должен быть 0)
	5	Размер спрайтов (0 - 8x8; 1 - 8x16)
	4	Выбор знакогенератора фона (0/1)
	3	Выбор знакогенератора спрайтов (0/1)
	2	Выбор режима инкремента адреса при обращении к видеопамяти (0 – увеличение на единицу «горизонтальная запись»; 1 - увеличение на 32 «вертикальная запись»)
	1,0	Адрес активной экранной страницы (00 – $2000; 01 – $2400; 10 – $2800; 11 - $2C00)
$2001 w/o		**Управление видеопроцессором**
	7-5	Яркость экрана/интенсивность цвета в RGB (в Денди не используется)
	4	0 – Спрайты не отображаются; 1 – Спрайты отображаются
	3	0 – Фон не отображается; 1 – Фон отображается
	2	0 – Спрайты невидны в крайнем левом столбце; 1- Все спрайты видны
	1	0 – Рисунок фона невиден в крайнем левом столбце; 1- Весь фон виден
	0	Тип дисплея: Color/Monochrome (в Денди не используется)
$2002 r/o		**Состояние видеопроцессора**. Чтение сбрасывает некоторые биты!
	7	1 – PPU генерирует обратный кадровый импульс; 0 – PPU рисует картинку на экране. Сбрасывается при чтении.
	6	Устанавливается в 1 после вывода спрайта с номером 0. Сбрасывается при чтении или при кадровом синхроимпульсе.
	5	1 – На линии больше 8-и спрайтов; 0 - меньше
	4	1 – Запись в видеопамять разрешена; 0 - запрещена
	3-0	Не используются
$2003 w/o, **$2004 r/w**		**Операции с памятью спрайтов**. В регистр $2003 записывается адрес в памяти спрайтов ($00-$FF). После чего с регистром $2004 производится операция чтения/записи. После каждой операции происходит автоинкремент адреса на единицу.
$2005 w/o(x2)		**Аппаратный скроллинг фоновой картинки**. В регистр последовательно записываются два байта. Первый – абсолютное значение вертикального скроллинга; второе – горизонтального (подробнее см. главу «Отражение экранных страниц».)
$2006		**Операции с видеопамятью**. Обеспечивают доступ к любой ячейке адресного пространства видеопроцессора. Регистр $2006 – адрес (2байта), сначала

w/o(x2),
 	
$2007 r/w
		записывается старший. Регистр $2007 – операционный буфер (чтение/запись). После каждой операции происходит автоинкремент адреса на 1 или на 32 (см. регистр $2000- бит 2).
		
		Всвязи с тем, что шины адреса и данных у PPU совмещены, в архитектуре Денди предусмотрен регистр для хранения текущего адреса. Чтение происходит из ячейки адресуемой этим регистром, а лишь затем он выдает вновь записанный адрес. Соответственно содержимое новой ячейки будет доступно при следующей операции чтения. Рассмотрим это на примере:
		
		Пусть в VRAM с адреса $2000 содержится: $AA $BB $CC $DD.
		
		При операциях с VRAM инкремент равен 1.
		
		Результаты исполнения кода приведены в комментариях.
		
		`LDA #$20`
		`STA $2006`
		`LDA #$00`
		`STA $2006 ; VRAM address now set at $2000`
		`LDA $2007 ; A=?? VRAM Buffer=$AA`
		`LDA $2007 ; A=$AA VRAM Buffer=$BB`
		`LDA $2007 ; A=$BB VRAM Buffer=$CC`
		`LDA #$20`
		`STA $2006`
		`LDA #$00`
		`STA $2006 ; VRAM address now set at $2000`
		`LDA $2007 ; A=$CC VRAM Buffer=$AA`
		`LDA $2007 ; A=$AA VRAM Buffer=$BB`
		
		При работе с палитрами буфер адреса не используется (т.к. они находятся специальных регистрах видеопроцессора). При записи в память не требуется лишнего цикла.

Таблицы 2 и 3.

Теперь подробнее:

CHR-ROM (Она же **VROM**) [8k] – Вспомним нашу модель картриджа (из предыдущей главы), в которой второе ПЗУ размером 8k подключалось к шинам PPU. Функционально оно называется CHR-ROM. В нем хранятся два знакогенератора (по 4k каждый). **Знакогенератор** содержит «иконки» (они же «тайлы», «спрайты», «значки»), размер каждой иконки 8x8 пикселей. На каждый пиксель выделяется 2 бита. В каждом знакогенераторе 256 таких иконок. Все что «умеет» видеопроцессор – это отображать содержимое знакогенераторов (и ничего более!)

VRAM [2k] **–** Ровно столько в Денди оперативной видеопамяти. Далее под словом «видеопамять» - понимаем именно VRAM (не путать с адресным пространством PPU, которое включает, в том числе, и VRAM). В видеопамяти хранятся **экранные страницы** или иными словами – фоновая картинка (точнее: информация о том из каких иконок она состоит – **область символов**; два старших бита цвета иконки – **область атрибутов**). Каждая экранная страница занимает ровно 1k видеопамяти (960 байт – информация о номерах иконок из знакогенератора + 64 байта – информация о цвете групп иконок). В видеопамяти приставки (2k) умещается только две экранных страницы – в то время как архитектурно (для PPU) предусмотрено четыре экранных страницы. Подробнее об экранных страницах см. ниже в подразделе «Отражение экранных страниц».

Палитры фона и **спрайтов** [16 + 16 байт] – PPU может отображать 64 цвета (каждому номеру строго определенный цвет - см. рисунок 2). Соответствие цветов формату RGB см. приложение 2.

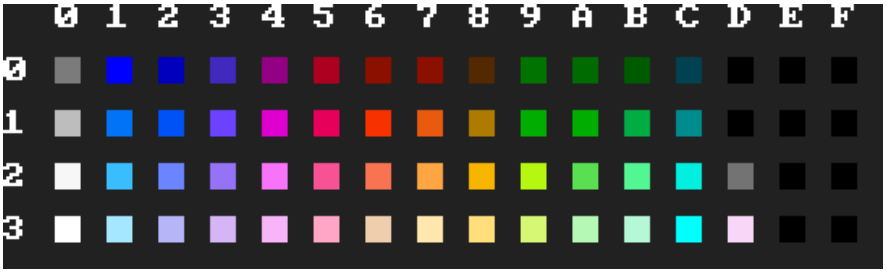


Рисунок 2.

PPU содержит в себе **регистры палитр** (адресуются через адресное пространство PPU). Предусмотрено две палитры – **палитра фона** (**$3F00-$3F0F**) и **палитра спрайтов** (**$3F10-$3F1F)**. При помощи палитры происходит выбор любых 16-ти цветов из 64-х возможных для текущего отображения на экране (в ячейку палитры заносится номер цвета (00h-3Fh) - см. рисунок 2).

Слои. Общий принцип формирования изображения.

PPU, работающий в системе PAL, формирует картинку с разрешением 256x240. PPU формирует изображение из четырех слоев путем их последовательного наложения (каждый следующий слой «заслоняет» предыдущий, если не является прозрачным в данной точке):

Задний план – холст, окрашенный цветом из палитры фона с индексом 0 (цвет любой - см. рисунок 2).

Слой спрайтов с битом приоритета = 0 (подробнее см. в разделе «Спрайты. Контроллер DMA.»)

Фоновый рисунок – картинка, составленная из иконок знакогенератора (как из мозаики 32x30). Фоновый рисунок хранится в экранной странице (VRAM). В области символов - 960 байт (32x30) должны быть записаны номера иконок знакогенератора фона (см. регистр $2000 - бит 4). На экране отображается содержимое активной экранной страницы (см. регистр $2000 - бит 1,0). Экранная страница содержит только информацию о фоновом рисунке. Знакогенератор содержит иконки с двумя младшими битами цвета для каждого пикселя иконки. Два старших бита цвета – общие для всей иконки, берутся из области атрибутов экранной страницы. Полученное четырехбитное число определяет номер цвета в палитре фона. Таким образом, каждая иконка (без смены палитры) может иметь до четырех вариантов раскраски. Т.е. при помощи двух бит из области атрибутов экранной страницы – палитра фактически разбивается на 4 части (по 4 цвета в каждой), номер части определяется битами области атрибутов, номер цвета (внутри части) – битами пикселя в знакогенераторе.

Слой спрайтов с битом приоритета = 1 (подробнее см. в разделе «Спрайты. Контроллер DMA.»)

Далее рассмотрим двоичные форматы перечисленных структур.

Знакогенератор.

В знакогенераторе отводится по 16 байт на каждую иконку (16x256=4096). Первые 8 байт определяют построчно младший бит цвета иконки. Следующие 8 бит – старший (см. рисунок).

Пиксель иконки, со значением обеих бит равными нулю – считается прозрачным (вне зависимости от старших бит из области атрибутов).

```
Символ в                      Результат
знакогенераторе               расшифровки
---------------               -----------
%00010000 = $10 --¬            ...1....  . - 00 (прозрачный)
%00000000 = $00   ¦            ..2.2...  1 - 01
%01000100 = $44   ¦            .3...3..  2 - 10
%00000000 = $00   +-- Бит 0    2.....2.  3 - 11
%11111110 = $FE   ¦            1111111.
%00000000 = $00   ¦            2.....2.
%10000010 = $82   ¦            3.....3.
%00000000 = $00 ---            ........
%00000000 = $00 --¬
%00101000 = $28   ¦
%01000100 = $44   ¦
%10000010 = $82   +-- Бит 1
%00000000 = $00   ¦
%10000010 = $82   ¦
%10000010 = $82   ¦
%00000000 = $00 ---
```

Область символов экранной страницы.

Последовательно хранятся номера иконок из знакогенератора фона (величина смещения иконки от начала знакогенератора деленная на 16). Экран заполняется иконками построчно (32x30=960 иконок на экране).

Область атрибутов экранной страницы.

Для каждой иконки в фоновой картинке, два старшие бита цвета, выбираемого из палитры фона, определяются соответствующими битами байта атрибута области атрибутов. Причем старшие биты цвета одинаковы для группы иконок.

$2000	**$2001**	**$2002**	**$2003**
$2020	**$2021**	**$2022**	**$2023**
$2040	**$2041**	**$2042**	**$2043**
$2060	**$2061**	**$2062**	**$2063**

Табличка выше иллюстрирует фрагмент VRAM, адреса из области символов Экранной страницы 1 («левый верхний угол экрана») – изображены ячейки с адресами начала первых четырех строк экрана. Цветом показаны группы иконок, имеющие одинаковый атрибут (два старших бита индекса цвета в палитре фона).

Байт области атрибутов (для указанного выше фрагмента: «левый верхний угол» - по адресу $23C0) содержит атрибуты четырех смежных групп:

Цветом указано соответствие битов байта атрибутов группам иконок экранной страницы.

Весь экран построчно разбивается на группы (8 строк и 8 столбцов) – последняя строка неполная. Соответственно за каждую группу отвечает один байт атрибутов.

Палитры.

На самом деле не из 16-ти цветов палитры происходит выбор цвета. Происходит это потому, что и здесь имеет место явление «отражения» (mirroring) – на этот раз применительно к цветам палитры. Происходит это так. Как отмечалось выше, пиксель иконки знакогенератора, оба бита которого нулевые, вне зависимости от атрибута, считается прозрачным. Из этого следует, что в палитре фона есть 3 ячейки, содержимое которых нас не интересует (цвет то и так прозрачный по условию). Это ячейки с адресами $3F04, $3F08 и $3F0C. В ячейке $3F00 (как упоминалось выше) хранится номер цвета холста, и именно он виден сквозь «прозрачные» пиксели. Так вот – архитектура PPU Денди предусматривает отражение «лишних» ячеек палитры ($3F04, $3F08 и $3F0C) в ячейку цвета холста ($3F00).

С палитрой спрайтов таже история – у спрайтов (подробнее о спрайтах в следующем разделе) нет «своего» холста, и поэтому ячейки палитры спрайтов, на которые приходятся прозрачные цвета ($3F10, $3F14, $3F18 и $3F1C), также отражаются на ячейку $3F00.

Спрайты. Контроллер DMA.

Со спрайтами все очень просто. В качестве спрайтов выступают все теже иконки знакогенератора (определенного, «как знакогенератор спрайтов» - см. регистр $2000 - бит 3). Внутри PPU есть отдельно адресуемая (от основного адресного пространства PPU – 16k) память спрайтов, размером 256 байт. Эта память хранит «записи» о 64 спрайтах, размер каждой записи 4 байта. Располагаются записи последовательно от младших адресов.

Формат записи следующий:

Байт записи	**Биты**	**Назначение**
0		**Абсолютная координата верхнего левого угла спрайта по вертикали**
1		**Номер иконки из знакогенератора**
2		**Атрибуты спрайта**
	7	Отражение спрайта относительно вертикальной оси. 0 – Обычный; 1 - Зеркальный
	6	Отражение спрайта относительно горизонтальной оси. 0 – Обычный; 1 - Зеркальный
	5	Приоритет спрайта. 1 – Спрайт перед фоном; 0 – Спрайт за фоном
	4-2	Не используются
	1,0	Два старших бита цвета (аналог атрибута цвета для фона)
3		**Координата верхнего левого угла спрайта по горизонтали**

Таблица 4.

Спрайты нумеруются начиная с нуля. После вывода нулевого спрайта (а он может быть в любом месте экрана) устанавливается бит 6 регистра $2002 PPU. Запись/чтение в/из память/и спрайтов производится при помощи регистров $2003 (адрес) и $2004 (данные). При каждой операции адрес автоинкрементируется на 1.

Существует более быстрый способ записи в память спрайтов – запись через контроллер DMA. Контроллер DMA занимает в адресном пространстве CPU один адрес - $4014. При записи

в этот порт числа $XX, контроллер DMA отправляет в память спрайтов содержимое 256-ти ячеек памяти, начиная с адреса = $XX x $100. (Например: при записи в регистр контроллера DMA $02 – в память спрайтов отправится содержимое ячеек $0200-$02FF). В программах рекомендуется использовать именно этот способ записи – как наиболее быстрый (около 100мкс.)

Отражение экранных страниц.

Как уже упоминалось выше – в Денди установлено 2k VRAM (сразу еще раз ругнемся на разработчиков – пожалели еще одну микросхему ОЗУ). Да именно пожалели, если посмотреть на карту адресного пространства PPU (см. выше) – то можно видеть, что реально видеопроцессор может работать с четырьмя страницами VRAM. Оно так и бывает, если на картридже установлены недостающие 2k VRAM – уймись программист или раскошелься пользователь, если разработчик решил сэкономить!

Но вернемся к архитектуре – какие возможности добавляют нам лишние 2k VRAM?

Рассмотрим все по порядку …

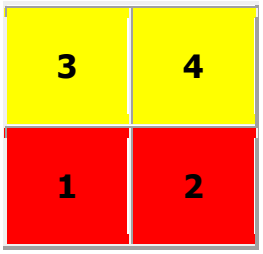
PPU Денди обеспечивает такой эффект, как «скроллинг» (прокрутка) фонового рисунка (независимо от спрайтов) – причем, абсолютно аппаратно. Скроллинг может быть как горизонтальным, так и вертикальным или и тем и тем сразу (по диагонали) – таким образом на в видимую область попадает содержимое нескольких экранных страниц. Экранные страницы имеют следующую пространственную модель (нумерация с привязкой к адресам в адресном пространстве PPU):

3 $2800	**4 $2C00**
1 $2000	**2 $2400**

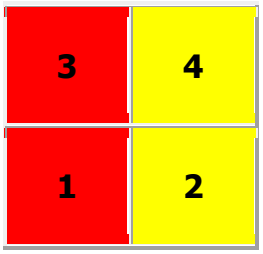
В регистр $2005 последовательно записываются два значения - абсолютное вертикальное и горизонтальное смещение (соответственно) относительно текущей активной страницы. Например, активная страница 1 ($2000) – горизонтальное смещение ноль, а вертикальное $E0 (240) – как результат, на экране видно содержимое страницы 3. Варьируя значение вертикального смещения, можно получить фон, состоящий из части страниц 1 и 3 (их стыковку). Если «закрутить» эту операцию в цикл – то можно получить эффект «скроллинга» фоновой картинки. Аналогично происходит горизонтальный скроллинг (прокрутка).

Программы, не использующие функцию скроллинга, должны записывать в регистр $2005 два нуля при каждой обработке прерывания VBlank, иначе изображение будет смещенным!

Но есть один нюанс – в Денди нет памяти подо все четыре страницы. Как и в прочих случаях, тут тоже имеет место «отражение» - и на этот раз экранных страниц (друг на друга). Вид отражения определяет архитектура используемого картриджа. Существует 2 вида отражения – горизонтальное (1=2 и 3=4) и вертикальное (1=3 и 2=4). То есть содержимое указанных страниц идентично (т.е. зеркалируется).

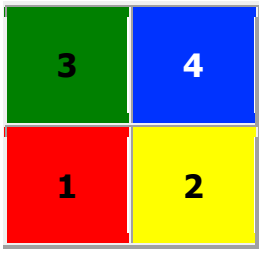


Горизонтальное отражение экранных страниц



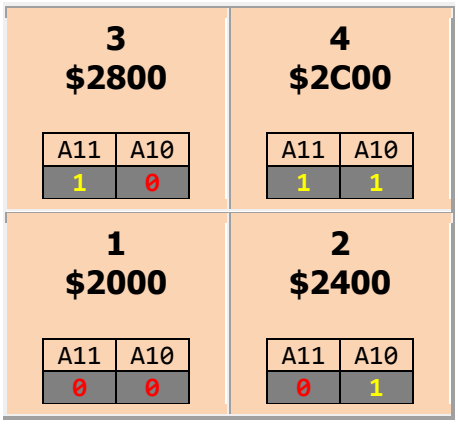
Вертикальное отражение экранных страниц

Если в картридже установлены недостающие 2k VRAM – то содержимое всех четырех страниц может быть различным.



Вне зависимости от способа отражения (или при наличии всех четырех страниц) – скроллинг возможен в любом направлении (в том числе и по диагонали).

Аппаратная реализация выбора способа отражения экранных страниц следующая. Старшая линия физической микросхемы VRAM в приставке (A10) выведена исключительно на разъем картриджа VRAM A10 (см. приложение 4 и Приложение 5), таким образом, управлять данной линией может либо маппер (например, MMC3) – либо данная линия жестко «закольцована» в картридже на одну из адресных линий PPU (A10 или A11), тем самым и определяя вид зеркалирования экранных страниц. Лучше понять суть описанного поможет следующая (уже знакомая) иллюстрация.



Горизонтальное отражение – источником сигнала VRAM A10 является линия A11 (состояние линии A10 не определяет адрес в микросхеме VRAM);

Вертикальное отражение - источником сигнала VRAM A10 является линия A10 (состояние линии A11 не определяет адрес в микросхеме VRAM);

(В раскраске бит соответствующих адресных линий видна аналогия с иллюстрациями видов отражений выше.)

Картридж Денди.

Денди спроектирована так, что вся программа и все данные хранятся исключительно на картридже. Сама приставка не содержит никакой постоянной памяти. Картридж устанавливается в разъём, на который выведены как шины CPU, так и PPU (см. приложение 4). До сих пор мы, для простоты и наглядности, ссылались на модель картриджа, содержащую 32k PRG-ROM и 8k CHR-ROM без наличия маппера. Такие картриджи подходят для хранения небольших игр - их память полностью адресуется CPU и PPU. Для большинства же игр такой объём явно недостаточен – поэтому большинство картриджей имеют значительно больший объём памяти (до 1024k PRG-ROM и 1024k CHR-ROM). Понятно, что непосредственно процессор не может адресовать такой объём памяти. В Денди применяется метод «страничной адресации» памяти картриджа. Картриджи большого объёма содержат, помимо микросхем памяти, еще и переключатель страниц (маппер). Маппер («mapper») – комбинационная схема (или контроллер), состояние которого однозначно определяет блоки ПЗУ картриджа, отображаемые в данный момент в адресное пространство процессора, или знакогенератор PPU. Размер переключаемых окон адресного пространства CPU зависит от типа маппера, а с некоторыми мапперами может даже варьироваться, типичные конфигурации - 2*16k, 2*8k+16k, 4*8k (возможны и иные конфигурации - причем, как правило, последнее окно неперключаемое).

Видов мапперов существует очень много (наиболее распространены около десяти) – каждый из них имеет свой функционал по управлению аппаратной частью картриджа: поддержка того или иного вида и объёма используемых в картридже микросхем памяти, возможность использования дополнительной памяти (ОЗУ), наличие других дополнительных функций (звукового процессора, таймеров и пр.). Мапперы имеют свои (отличные от других) «системы команды» управления их состоянием. Большим количеством модификаций «классических» мапперов мы обязаны «пиратам» - стремящимся либо с сэкономить на железе, любо запихнуть на один картридж много всего разного, а зачастую - и то и другое сразу ;-) Несмотря на все разнообразие (см. перечень мапперов), базовый функционал всех без исключения мапперов по большому счету идентичен – состояние регистров маппера определяет (управляет) старшими линиями адреса микросхем PRG-ROM и/или CHR-ROM (для которых не хватает адресных линий в разъеме картриджа).

И так, при наличии маппера, картридж Денди содержит:

PRG-ROM – микросхема ПЗУ, хранящая программу и данные (неотъемлемая часть любого картриджа). Подключается к шинам процессора. Старшие линии адреса микросхемы ПЗУ, а также управляющие входы подключаются к мапперу (если маппер предусматривает переключение страниц PRG-ROM).

CHR-ROM/CHR-RAM – микросхема памяти, подключаемая к шинам PPU - возможен один из двух вариантов:

- ***ПЗУ (CHR-ROM)*** – хранит заранее «прошитые» знакогенераторы PPU, которые могут переключаться, если это предусматривает маппер. Подключается к шинам PPU и мапперу (если маппер предусматривает переключение страниц знакогенератора).

- ***ОЗУ (CHR-RAM)*** – статическая память, размером 8k (два знакогенератора). В этом случае «иконки» знакогенераторов должны храниться в PRG-ROM (или генерироваться алгоритмически) и программно (через регистры управления PPU) загружаться в **CHR-RAM**. Преимуществом такой организации является то, что есть возможность изменения одной иконки знакогенератора (или даже ее части), при сохранении всех остальных иконок неизменными. *В предыдущем случае (CHR-ROM) имелась возможность только выбирать заранее созданные и прошитые в ПЗУ пресеты знакогенераторов (если их несколько) – зато путем переключения можно намного быстрее изменить группу иконок, чем поштучно их подгружать.*

Mapper – комбинационная схема (или контроллер), осуществляющие коммутацию блоков микросхем ПЗУ в адресные пространства CPU и/или PPU – т.е. фактически управляет старшими линиями адреса микросхем ПЗУ, для которых «не хватает» адресных линий в слоте картриджа и/или адресном пространстве процессора. Маппер находится на картридже и подключается к шинам CPU и/или PPU и соответствующим микросхемам памяти. Некоторые мапперы состоят из нескольких микросхем. Любая программа (игра) пишется под определенный тип маппера. *Т.е., говоря современным языком: драйвер маппера «вшивается» в код программы, более того – он зачастую «размазан» по всему коду … Возможность адаптировать (модифицировать) программу (дамп игры) под другой маппер – задача довольно рутинная – нужно найти и откорректировать все места, где происходит управление маппером.*

Также картридж (в редких случаях) может содержать:

SRAM (она же **WRAM) -** (Адресное пространство CPU **$6000-$7FFF) –** Статическое ОЗУ, может питаться от компактной батарейки, располагающейся на картридже. Предназначено для «сохранения» игр (если есть батарейка), или просто служит дополнительным ОЗУ (хорошее дополнение). Для него зарезервировано «окно» в адресном пространстве CPU размером 8k (в 4 раза больше встроенного в приставку ОЗУ, а скорость таже и прямая адресация) - существуют картриджи, несущие на борту до 32k ОЗУ (т.е. 4 страницы, переключаемые маппером в этом окне).

Expansion ROM\RAM - (Адресное пространство CPU **$5000-$5FFF) –** Дополнительные 4k памяти (для CPU). В целом аналогично SRAM (используется, как правило, с маппером MMC5).

VRAM - (Адресное пространство PPU **$2800-$2FFF) –** «Недостающие» 2k VRAM для двух экранных страниц PPU (3 и 4). Редкая фишка - используется очень немногими играми.

Звуковой процессор - предназначен для более качественного синтеза звука, нежели это реализуется встроенным pAPU (например, в мапперах VRC6, VRC7 и других).

Большинство аппаратных возможностей Денди могут быть использованы (дополнены), например, маппером MMC5 или MMC3 (большинство технологичных игр используют именно их). Разумеется, ничто не мешает разработать новый маппер самому (в архитектуре приставки ничего менять не придется! - всю логику управления маппером содержит программа на картридже), чем китайские разработчики («пираты») увлекаются больше других, а вместе с этим производят и совместимые аналоги «официальных» микросхем мапперов – наиболее популярные из которых следующие:

MMC1 – AX5904

MMC3 – AX5202

VRC4 – AX5208

Программирование мапперов.

В адресном пространстве CPU - адреса $8000-$FFFF выделены для отражения на них памяти картриджа - PRG-ROM. Логически может быть организовано несколько окон, управляемых маппером. Размер и количество окон определяется архитектурой маппера, некоторые типы предусматривают переменный размер переключаемого окна (например, MMC3). Наиболее распространенный вариант 2*16k, 2*8k+16k или 4*8k - в последнее окно ($xxxx-$FFFF) позиционируется или «хвост» ПЗУ, или его начало. Также, зачастую, последнее окно переключению не подлежит (в нем находятся вектора прерываний).

Каждый маппер имеет определенное количество регистров управления, доступных только для записи. Адресуются они через области, отведенные для PRG-ROM. То есть – чтение из памяти по адресам $8000-$FFFF приводит к чтению ПЗУ (PRG-ROM), а запись, по некоторым адресам из этого диапазона, приводит к записи управляющих слов в соответствующие регистры маппера.

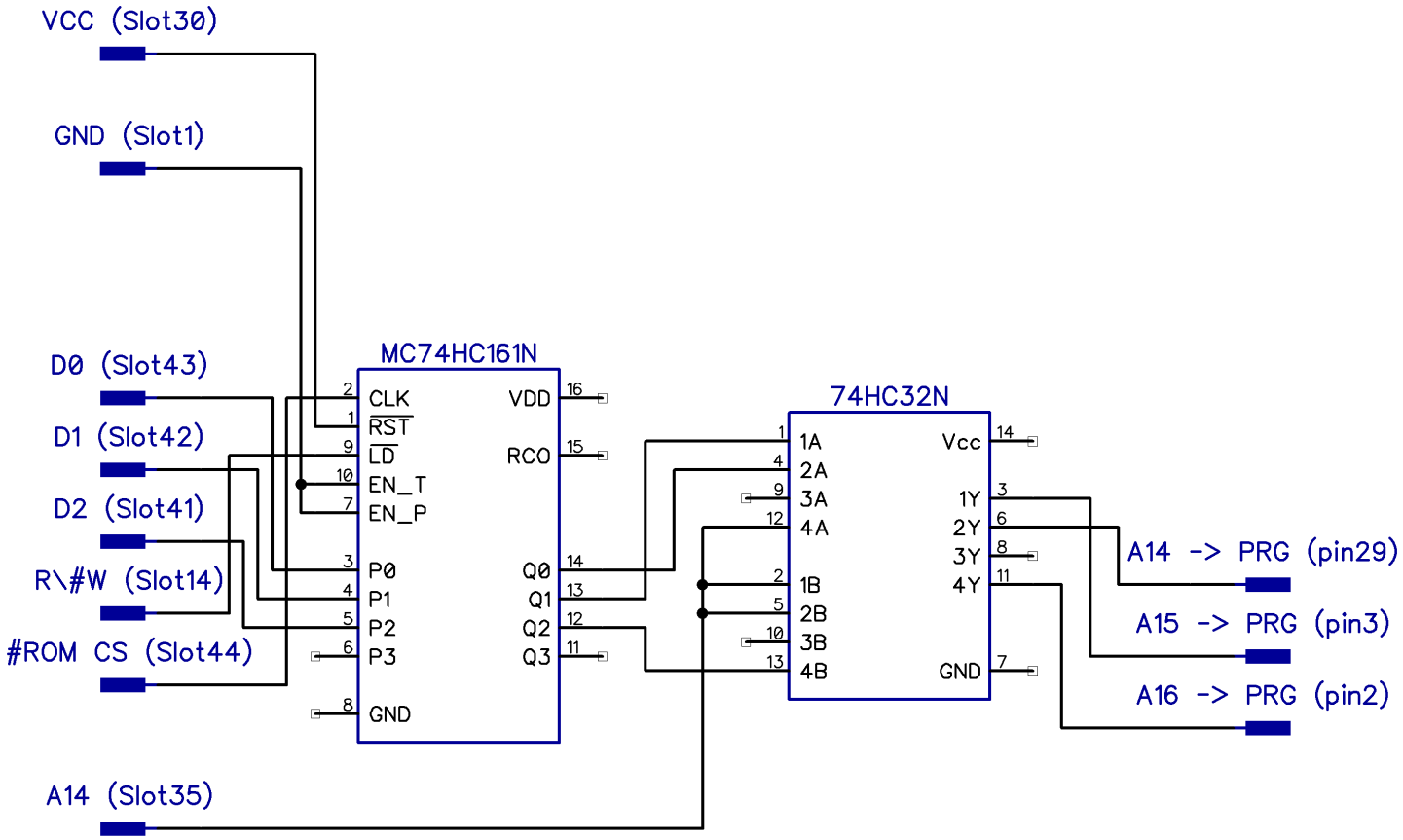
Подробное описание разных мапперов – не цель данного материала, однако некоторые из них рассмотрим подробнее.

Маппер UNROM.

Для примера рассмотрим маппер UNROM – один из самых простейших мапперов, но весьма удобный для знакомства с программированием Денди и написания несложных программ. Он предусматривает работу с PRG-ROM объёмом 128k (17 линий адреса, 3-мя старшими линиями управляет маппер), в качестве CHR используется непосредственно ОЗУ размером 8k (без участия маппера). Маппер строится на базе двух микросхем (74HC161 и 74HC32) и программно представлен как один регистр, который отражается на всё пространство PRG-ROM ($8000-$FFFF). То есть запись по любому адресу из этого диапазона приведет к записи в регистр маппера. Записываемый байт (а вернее 3 младших его бита) определяет блок микросхемы ПЗУ (16k), подключаемый в окно $8000-$BFFF (т.е. при A14=0) - блоки нумеруются с нуля и с начала адресации микросхемы ПЗУ. В окно $C000-$FFFF (т.е. при A14=1, вне зависимости от содержимого регистра маппера) всегда подключен последний блок (последние 16k) ПЗУ (A14=A15=A16=1 на адресных входах ПЗУ). Таким образом, возможен вариант (при «xxxxx111» в регистре маппера) когда последний блок ПЗУ будет последовательно продублирован в адресном пространстве.

Знакогенератор CHR-RAM - заполняется программно, через регистры PPU ($2006,$2007) – запись должна производиться в адресное пространство PPU, отведенное под знакогенератор ($0000-$1FFF). Вид отражения экранных страниц определяется распайкой (коммутацией) перемычки на плате картриджа.

Принципиальная схема маппера UNROM такова:



Разумеется, на схеме выбор (из двух) логических входов элементов 2-ИЛИ может быть произвольным, как и выбор используемых 3 из 4 таких элементов 2-ИЛИ из микросхемы 74HC32 (аналог «ЛЛ1») – на рисунке изображен один из множества возможных вариантов (точная копия коммутации на примере реализации в картридже на плате HVC-UNROM-03). Счетчик 74HC161 (аналог «ИЕ10») в данной схеме используется фактически как регистр (а не счетчик).

Super Hik.

Отдельно можно отметить такое древнее и весьма интересное инженерное решение китайских разработчиков как «Super Hik» - это картридж, содержащий в себе несколько (как правило, от 4 до 8) полноценных «больших» игр под маппер MMC3 (каждая)! Большинство игр MMC3 имеют размер 128k или 256k PRG-ROM и столько же CHR-ROM. Супер-хики имеют микросхемы памяти 512k или 1024k, в роли маппера MMC3 выступает совместимый (китайский) аналог AX5202 (в корпусе DIP-40 или бескорпусном варианте), также на нескольких логических микросхемах (триггерах, мультиплексорах, дешифраторах) сконструирован еще один маппер - «маппер верхнего (по отношению к MMC3) уровня». Получается так: в слоте картриджа присутствует 15 линий адреса (32k), штатный маппер MMC3 (в исполнении AX5202) берет на себя (в т.ч.) управление 16-ой и 17-ой линиями адреса памяти (ПЗУ 128k) и 18-ой (ПЗУ 256k) – что обеспечивает штатное функционирование любой из игр. Переключение же игр – управление линиями адреса 18-ой и 19-ой (ПЗУ 512k – 4 игры) и 20-ой (ПЗУ 1024k – 7 или 8 игр) осуществляется маппером верхнего уровня (сконструированного, например, на связке 74LS174 и 74LS157 для четырех игр), управляется он записью по адресу, не пересекающемуся с управлением MMC3, в неиспользуемое пространство памяти программ одной из игр добавляется код меню выбора игры (этакий «патч»), который и записывает требуемое значение в соответствии с выбранной игрой в регистр маппера верхнего уровня, для MMC3 это абсолютно прозрачно. В качестве примера работы маппера картриджа «Supper Hik» рассмотрим маппер под номером «49».

Маппер 49 (Super Hik 4-in-1).

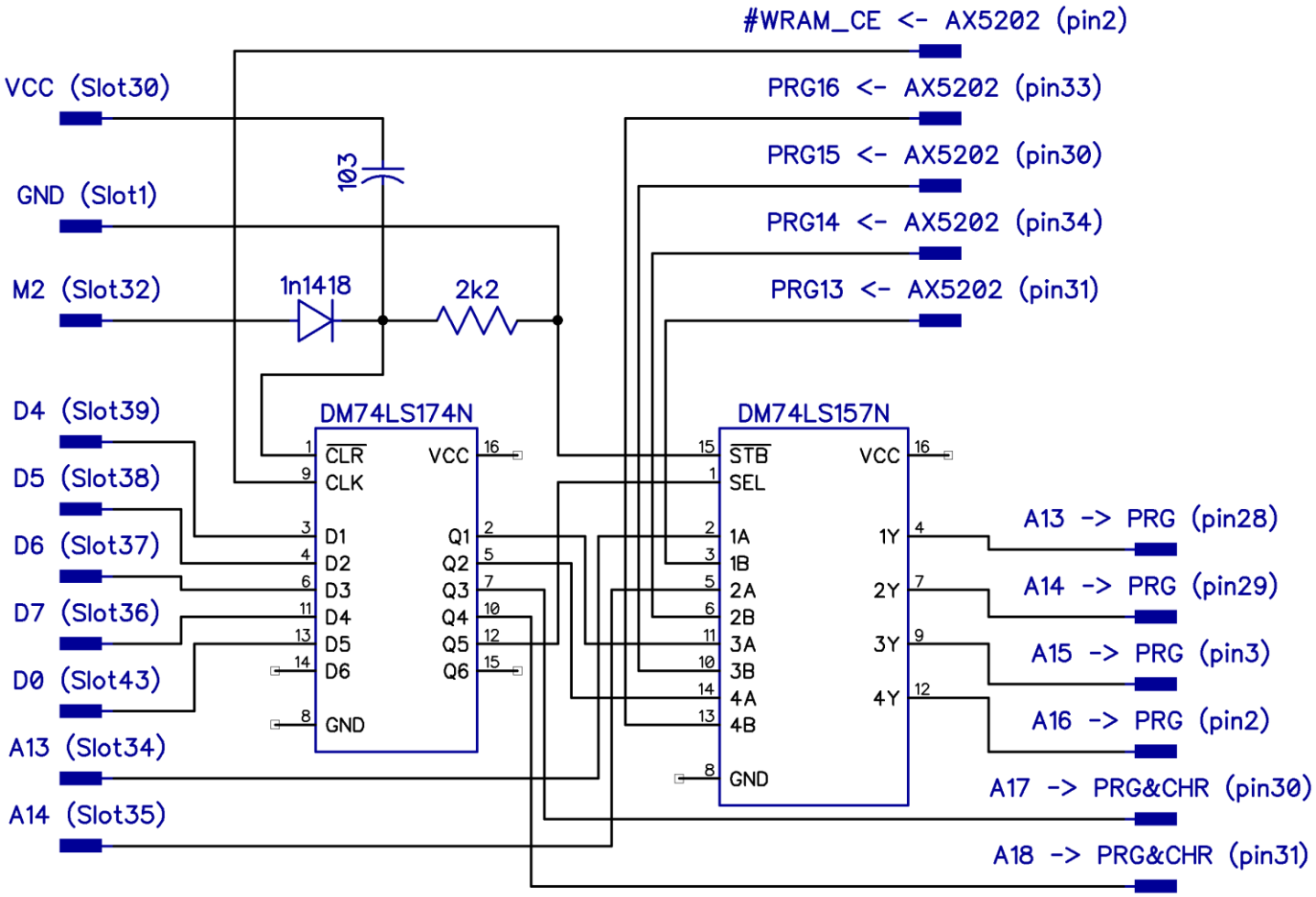
Многие игры используют далеко не весь функционал, предоставляемый маппером MMC3, и в первую очередь в части размера PRG и CHR. Таким образом, выглядит вполне логичным (с технической точки зрения) создание мульти-игрового картриджа без микросхемы ОЗУ (WRAM), где под каждую из игр зарезервирован слот размером 128k (и в PRG и в CHR) – т.е. применение микросхем емкостью 4 мегабита (19 адресных линий). Игр, которым такой объем ресурсов совместно с маппером MMC3 (его аналогом AX5202) достаточен - предостаточно!

Рассмотрим механизм переключения слотов с играми такого мульти-игрового картриджа с точки зрения схемотехники и управления ей. Фактически мульти-игровой картридж «Super Hik» содержит в себе два маппера! MMC3 (полноценный, аппаратный) – именно с ним работают игры (выбранная в данный момент игра), и «маппер верхнего уровня» (этакий «гипервизор») – который обеспечивает выбор слота с игрой, которая и будет запущена … А так как для каждой из игр маппер верхнего уровня прозрачен (что логично), рассмотрим именное его принцип работы.

Маппер верхнего уровня представляет собой регистр «только для записи», доступный в диапазоне $6000-$7FFF. Т.к. за доступ к данному диапазону адресов отвечает MMC3, то необходимо сначала разрешить такой доступ (управляющая команда к MMC3). Конечно жаль, что из-за такого способа обращения к регистру маппера теряется возможность установки на картридж WRAM и стало быть запуска игр, требующих дополнительное ОЗУ по этим адресам, но сейчас не об этом.

Общая цель маппера верхнего уровня проста – обеспечить выбор (адресацию) старших адресных линий микросхем ПЗУ (PRG и CHR) за пределами 128k, которыми управляет MMC3. Т.е. должна быть возможность управления линиями A17 и A18 у обоих микросхем ПЗУ, хотя и изменять состояние этих линий предстоит вобщем то не часто: при включении/сбросе приставе на этапе выбора игры (слота с игрой внутри микросхем памяти).

Принципиальная схема «маппера верхнего уровня» внутри «маппера 49» такова:



Таким образом, в регистре маппера присутствуют только 5 бит. С программной точки зрения возможны два режима работы маппера (выбираемые битом D0) в части переключения страниц PRG-ROM.

«Основной режим» (D0=1) – в этом режиме слот с игрой (один из четырех) выбирается битами D6 и D7 (выполняют роль сигналов адресных линий A17 и A18 физических ПЗУ как PRG-ROM, так и CHR-ROM); значение бит D4 и D5 в регистре - безразлично; адресными линиями A13-A16 PRG-ROM и A10-A16 CHR-ROM управляет маппер MMC3. Игра не выходит за пределы своего слота и управляет только маппером MMC3.

«Альтернативный режим» (D0=0) – именно этот режим активируется при загрузке (сбросе), т.к. аналоговая часть схемотехники маппера верхнего уровня (RC-цепочка) сбрасывает весь регистр маппера в ноль. В этом режиме MMC3 не управляет адресными линиями PRG-ROM – ими управляет исключительно маппер верхнего уровня. Биты D6 и D7, как и прежде, подключены к линиям A17 и A18 как и PRG-ROM, так и CHR-ROM (т.е. для CHR-ROM оба режима абсолютно одинаковы) – а вот биты регистра маппера D4 и D5 теперь задают значения адресных линий A15 и A16 PRG-ROM соответственно, а линии A13 и A14 берутся непосредственно со слота картриджа. В этом режиме получается программная модель переключения страниц PRG-ROM, в некоторой степени, аналогичная мапперу UNROM – в данном случае 4 старших бита регистра маппера выбирают одну из 16 страниц размером 32k микросхемы PRG-ROM, подключаемую в окно $8000-$FFFF (после сброса – самое начало ПЗУ). *Повторюсь: переключение страниц CHR-ROM в «альтернативном режиме» происходит аналогично «основному режиму» - линии A17 и A18 выбираются маппером верхнего уровня (биты D6 и D7), более младшими битами CHR-ROM всегда управляет маппер MMC3.*

В целом, для использования такого симбиоза мапперов в конечном решении, необходимо как минимум вносить изменение в дамп игры, помещаемый в первый слот. Т.е. искать игру у

которой, хотя бы, хвост первых 32k дапма свободен для размещения векторов прерываний, а также в этом блоке есть место для размещения их обработчиков (в лучшем случае - кода для реализации меню выбора игр, или накрайняк – для переключения банков «альтернативным способом» на более свободный, где уже и разместить код меню). Не забываем, что картинку меню придется складывать из иконок выбранного блока CHR-ROM (что возможно повлечет необходимость или выбора одной из игр с подходящими символами в одной из страниц, которую нужно будет выбрать двумя мапперами, или поиск игры с неиспользуемыми ячейками CHR-ROM, в которые можно разместить нужные символы).

Ввод/Вывод в Денди.

Архитектура Денди не изобилует большим (а может и достаточным) количеством портов ввода/вывода. Программно под порты внешней периферии выделены две ячейки из адресного пространства CPU (на аппаратном уровне архитектуры). Это регистры с адресами $4016 и $4017. Аппаратных же линий и того меньше – а именно: 3 линии (бита) на выход и 5 + 2 линии на вход. Опрос линий чисто программный.

Вывод – 3 бита **"OUT"** (непосредственно с процессора), адресуются путем записи значения в регистр $4016, младшие биты. Бит 0 используется в качестве стробирующего сигнала-защелки состояния джойстиков, обоих сразу (см. ниже).

Ввод – порт $4016 **"J1"** (чтение), аппаратно присутствуют только два младших бита. Бит D0 выведен на «Порт A» и используется для чтения последовательного кода состояния джойстика 1. Бит D1 выведен на «Порт B».

Ввод – порт $4017 **"J2"** (чтение), аппаратно присутствуют 5 младших бит (два старших бита используются pAPU – см. ниже).

Наиболее распространены (в России) модели приставок Денди с двумя «разъёмами для джойстиков» (так их называют в быту) по 15 контактов каждый. Встречаются варианты и с двумя 9-ти контактными разъемами, или шнуры джойстиков могут быть «вделаны» в приставку (как у оригинального Famicom) - в этих случаях есть еще один 15-ти штырьковый разъём («порт расширений»). Условимся называть разъём для подключения первого джойстика «Порт A», второго «Порт B» (каждый по 15 или 9 штырьков). Если оба порта имеют по 9 штырьков (или джойстики неразъемные), то есть третий (или первый) «Порт C» – 15-ти штырьковый, который дублирует «Порт B» и содержит дополнительные сигналы. Если, как в большинстве случаев для Денди, оба разъема имеют 15 контактов, то к «Порту A» подключено не более шести, а ко второму – все 15 (в том числе и сигналы с «Порта A», правда, на другие контакты), т.е. получается что разъем «второго» джойстика - «Порт B», какбы объединен с портом расширений - «Порт C».

В общем случае «Порт A» предназначен для подключения «первого» джойстика (аппаратно возможно и подключение светового пистолета - «Zapper»). К «Порту B & C» (порту расширений) подключаются все другие периферийные устройства («второй» джойстик, световой пистолет, разветвитель на 4 джойстика и многое другое), фактически на него выведены все линии портов ввода/вывода Денди (за исключением одного из «Порта A»). Теперь подробнее об аппаратной реализации и программной модели.

Распиновка портов (по разъему совместимых с DB-15) приведена на рисунке.

«Порт A» (15 контактов)

1	GND
2	SND IN

8	J1 D0
9	CLK1
12	OUT0 (STRB)
15	VCC

«Порт B & C» (15 контактов)

1	GND
2	SND IN
3	#IRQ
4	J2 D4
5	J2 D3
6	J2 D2
7	J2 D1
8	J2 D0
9	CLK2
10	OUT2
11	OUT1
12	OUT0 (STRB)
13	J1 D1
14	CLK1
15	VCC

Представленный «ассортимент» и распиновка портов типичны как для Dendy Junior, Dendy Classic.

При каждом чтении из $4016 или $4017 процессор выдает сигналы (#OE), разрешающие чтение из буферов портов (для каждого порта свой сигнал – процессором предусмотрено аппаратно, см. Приложение 5). Соответственно считываются значения входных линий. Кстати опять, ничто не мешало разработчикам NES (Денди) сделать на каждом порту по 8 линий на вход.

О других сигналах в портах ввода/вывода:

SND IN – аналоговый выход с pAPU (см. ниже) – например, в световой пистолет-пулемет («фирменный» от Стиплера) встроен динамик, из которого раздается звук стрельбы (этакий «фидбэк»), также на рукоятке данного пистолета-пулемета, помимо переключателя одиночной и автоматической стрельбы, была продублирована кнопка «B» (в некоторых играх, как то «Operation Wolf», таким пистолетом-пулеметом можно было не только постреливать вражье очередями, но и метать в них гранаты);

#IRQ – линия запроса пользовательского прерывания CPU (которая выведена и сюда и на разъем картриджа). Периферийное устройство может быть как источником прерывания, так и его обработчиком (для собственных нужд), если источником является, например, электроника на картридже (маппер);

CLK1/CLK2 – синхроимпульс опорной (тактовой) частоты процессора (1.78МГц для PAL) – аналогично сигналу #M2 в разъеме картриджа;

Некоторые (как правило, «не Денди») приставки имеют девяти контактные разъемы (совместимые с DB-9) для джойстиков, распиновка которых может быть такой:

«Порт A» (9 контактов)

2	J1 D0
3	OUT0 (STRB)

4	CLK1
6	VCC
8	GND

«Порт B» (9 контактов)

2	J2 D0
3	OUT0 (STRB)
4	CLK2
6	VCC
8	GND

А вместе с этим и 15-ти контактный порт расширений, для подключения другой периферии помимо джойстиков (или второго джойстика в 15-ти контактном «исполнении»):

«Порт C» (15 контактов)

1	GND
2	SND IN
3	#IRQ
4	J2 D4
5	J2 D3
6	J2 D2
7	J2 D1
8	J2 D0
9	CLK2
10	OUT2
11	OUT1
12	OUT0 (STRB)
13	J1 D1
14	CLK1
15	VCC

«Стандартные» джойстики.

Оригинальные джойстики NES (Famicom) имеют по 8 клавиш. Джойстик, предназначенный для использования в качестве «второго» в оригинальном варианте (Famicom) не имеет кнопок Select и Start, но зато имеет микрофон с регулятором громкости для него (усиленный сигнал с которого подается в приставку). Джойстики китайских клонов, а также российского клона консоли - Денди - имеют, как правило, кнопки Select и Start на обоих джойстиках, так как допускают их подключение к любому из портов (для одинокого геймера второй джойстик был запасным). Помимо этого мы привыкли и воспринимаем как должное наличие турбо-вариантов кнопок A и B. Вещь, несомненно, удобная в ряде игр – но рассматривать их стоит как «чит» - т.к. в оригинальных версиях консоли их нет. Разумеется, что турбо-кнопки не генерирует каких-либо состояний в системе отличных от тех, что определяет факт нажатия обычных кнопок A и B – просто используя турбо-кнопки нет необходимости «долбить» по клавишам. Да и не любой геймер сможет так быстро, равномерно и беспристрастно «долбить» как встроенный триггер ;-)

В примере наиболее распространенного варианта консоли: с двумя разъемами под джойстики по 15 ног в каждом (см. рисунок выше) - «стандартные» джойстики используют контакты: 1, 8, 9, 12 и 15.

Нажатие на клавиши джойстиков не вызывает в приставке никаких событий. Опрос джойстиков чисто программный, его частота должна выбираться программистом, исходя из необходимой точности и скорости реакции пользователя на события системы. Естественно, что

очень частый опрос может замедлить работу системы, да и не стоит забывать, что джойстик это механическое устройство, а пользователь не может работать со скоростью ЭВМ (пусть даже такой как NES).

И так – как было сказано выше, для опроса состояния джойстиков используются два порта из адресного пространства CPU - $4016 и $4017. Алгоритм следующий: сначала необходимо «зафиксировать» состояние кнопок в сдвиговом регистре джойстиков (обоих сразу, если оба подключены). Для этого формируется строб, «защелкивающий» этот регистр – т.е. по адресу $4016 младший бит сначала устанавливается в единицу, а затем в ноль (строб). Далее можно прочесть состояние кнопок джойстика на момент «защелкивания». Информация считывается в последовательном коде (из сдвигового регистра джойстика). Младший бит порта $4016 – первый джойстик (Порт A), порта $4017 – второй джойстик (Порт B). Информация считывается в следующей последовательности:

```
1 = A            9 = Ignored    17 = ID bite*
2 = B           10 = Ignored    18 = ID bite*
3 = SELECT      11 = Ignored    19 = ID bite*
4 = START       12 = Ignored    20 = ID bite*
5 = UP          13 = Ignored    21 = 0
6 = DOWN        14 = Ignored    22 = 0
7 = LEFT        15 = Ignored    23 = 0
8 = RIGHT       16 = Ignored    24 = 0
```

** ID bite - вероятно, в оригинальных NES-джойстиках биты 17-20 содержали дополнительную информацию, не связанную с нажатием клавиш. Во всех неоригинальных джойстиках, в том числе и «родных» Денди - аппаратно присутствуют только первые 8-бит, отвечающие непосредственно за нажатия кнопок. Дальнейшее чтение с такими джойстиками бессмысленно.*

Нажатой кнопке соответствует единица, не нажатой – ноль.

«Мультикнопочный» джойстик.

Есть для консоли Famicom такой интересный аксессуар как «мультикнопочный» (название произвольное, авторское) джойстик (HVC-051). Вернее он является дополнением для другой «штуковины» - модема! (HVC-050). Посчастливилось раздобыть в коллекцию экземпляр такого набора. Модем вставляется вместо картриджа, этаким «вторым этажом» - нет, не для онлайн игр нескольких пользователей (а для их скачивания со специализированных BBS-ок? - вроде так, но это было не в нашей стране, и очень давно). Еще в модем предусмотрена установка собственного картриджа «уникального» формата (отдаленно напоминающего PCMCIA-устройства конструктивно) … Не факт, что предусматривалось использование мульти-джойстика (в части дополнительных кнопок) отдельно от программного обеспечения используемого совместно с модемом … Однако, внутри пользовательских разработок, ничего не мешает опрашивать и обрабатывать на свое усмотрение все кнопки такого джойстика.

Подключается к приставке мультикнопочный джойстик весьма экзотично: его необходимо вставлять исключительно в порт расширений («Порт B & C» или «Порт C») – однако при этом приставка будет его видеть не как «второй», а как «третий» (внезапно?! – это по аналогии с адаптером на 4 джойстика и его аппаратно-программной моделью). Некоторые программы (игры) в прочем работают с ним как с «первым» … При этом, соответственно, «второй» джойстик штатным образом уже подключить не удастся (разъем занят) … Причина? – распиновка:

Штекер «мультикнопочного» джойстика

1	GND
12	OUT0 (STRB)
13	J1 D1
14	CLK1
15	VCC

Опрашивается такой джойстик, штатным для системы образом, но в нем присутствуют три последовательно включенных регистра. Назначение битов кнопок приведено ниже:

```
1 = A           9 = 0          17 = 8
2 = B          10 = 1          18 = 9
3 = SELECT     11 = 2          19 = *
4 = START      12 = 3          20 = #
5 = UP         13 = 4          21 = .
6 = DOWN       14 = 5          22 = C
7 = LEFT       15 = 6          23 = Ignored
8 = RIGHT      16 = 7          24 = Func
```

Первые 8-бит – стандартные для всех джойстиков, далее биты остальных кнопок. Биту 23 нет соответствия в виде кнопки – т.е. не используется (всегда нулевой). Кнопка, соответствующая биту 24 названа «Func» условно, т.к. англоязычная маркировка для данной кнопки отсутствует.

Джойстик Super-NES.

В качестве справочного материала, для сравнения, возможного использования в собственных проектах, а также для конструирования враппера и т.п. приведем таблицу «скан-кодов» джойстика приставки Super-NES (более известной у нас как Супер-Нинтендо) – т.к. фактически такой джойстик является вариацией «мультикнопочного» джойстика описанного

выше (с меньшим количеством кнопок, и двумя регистрами вместо трех), и является для Super-NES «стандартным» …

```
1 = B           9 = A           17 = Ignored
2 = Y          10 = X           18 = Ignored
3 = SELECT     11 = L           19 = Ignored
4 = START      12 = R           20 = Ignored
5 = UP         13 = Ignored     21 = Ignored
6 = DOWN       14 = Ignored     22 = Ignored
7 = LEFT       15 = Ignored     23 = Ignored
8 = RIGHT      16 = Ignored     24 = Ignored
```

Количество и назначение аппаратных линий, а также алгоритм опроса джойстика Супер-Нинтендо абсолютно идентичен таковому в Денди.

Формирование звуков в Денди.

Этому разделу, несомненно, можно посвятить отдельный труд. Ибо нестандартных решений в этом направлении достаточно много, хотя они и не сильно распространены. Возможности синтеза звуков средствами самой приставки относительно скудны, и пригодны, пожалуй, лишь для генерации спецэффектов, но не музыки (речь идет о pAPU сопроцессоре, «синтезаторе»).

Сэкономив (в очередной раз) на синтезаторе, разработчики NES всёже реализовали возможность расширения аппаратных средств генерации звука. Как и в предыдущих случаях, это расширение должно находиться на картридже (а где же еще). Т.е., как альтернативу встроенному pAPU, картридж может содержать свой звуковой синтезатор (содержится в мультисхемах вместе с мапперами: VRC6, VRC7, FDS, MMC5, Namco129, FME-07 …), вписывающийся своими регистрами управления в адресное пространство процессора приставки (аналогично мапперам), на выходе же имеющий аналоговый сигнал сформированного звука. На разъёме картриджа для этого предусмотрено два контакта (см. приложение 4) – один из них: аналоговый выход с pAPU (встроенного в процессор Денди); второй: разведен на звуковой выход с приставки. В случае если картридж не содержит своего звукового процессора («синтезатора»), то эти контакты просто замкнуты. В случае использования синтезатора на картридже pAPU может быть отключен от выхода с приставки, хотя возможно использование микшера для объединения сигнала pAPU и синтезатора на картридже, не исключен и вариант установки двух звуковых процессоров на картридже (гипотетически) … но это уже все зависит от разработчиков картриджа. Увы, но картридж содержит ограниченное число игр (а в случае с «лицензионной» продукцией - одну), а покупать для каждой игры одно и тоже оборудование (хотябы даже один и тот же синтезатор) удовольствие дорогое, и как следствие мало распространенное. Можно отметить, что картридж со всеми возможными расширениями (большое ПЗУ (PRG-ROM, CHR-ROM), дополнительное ОЗУ, видео ОЗУ (экранных страниц), ОЗУ для сохранения игр, маппер и звуковой процессор, т.п.) может получиться сложнее («интеллектуальнее») самой приставки – и соответственно дороже ее (покупать ради новой игры еще один такой картридж стали бы не многие). Поэтому большинство производителей игр не гнались за качеством (хотя аппаратные ограничения и предполагали повышенный уровень искусства программистов), а брали дешевизной железа, используя лишь необходимый минимум (память программ и знакогенераторов, ну и маппер, как вынужденную необходимость). Ну а для звука – что есть – встроенный pAPU, который мы и рассмотрим чуть подробнее ниже.

Сопроцессор pAPU.

Звуковой сопроцессор pAPU является именно той «изюминкой», наличие которой отличает базовый процессор архитектуры MSC6502 и CPU, примененный в NES/Famicom/Dendy. А так как

первым «официальным» процессором был (видимо) Ricoh RP2A03, то и звуковой сопроцессор pAPU получил уже как нарицательное имя **2A03** (хотя входит в состав и других совместимых процессоров и SoC).

И так, pAPU 2A03 представляет собой пятиканальный синтезатор. Причем первые два канала генерируют сигнал прямоугольной формы, третий – треугольной (пила), четвертый – «шум», пятый – «цифровой канал» (дельта-модуляция, DMC). В некоторых «китайских» вариациях Денди (на неизвестной бескорпусной SoC) цифровой канал мог отсутствовать.

Подробное рассмотрение всех особенностей программирования 2A03 достаточно объёмно, даже про нюансы программирования цифрового канала (DMC) писались отдельные статьи. Здесь же (ниже) дадим лишь краткие сведения о pAPU – т.к. в англоязычной документации информация исчерпывающая, а заниматься чисто ее переводом я не стал (да и не думаю, что сегодня кого-либо всерьез заинтересует программирование звука в Денди с нуля или портированием с других источников). По-любому, если я займусь этой темой глобально, эта глава, несомненно, будет дополнена. А пока для общей картины, помимо изложенного выше, приведем таблицу с описанием регистров pAPU (более подробно можно прочитать в англоязычной документации см. приложение 6).

Регистры pAPU выполняют следующие функции:

Регистр (адресация по CPU)	Операция	Назначение
$4000	**w/o**	**Регистр управления канала 1 (*Прямоугольник [Square]*).** `ddle nnnn` – duty, loop env/disable length, env disable, vol/env *(режим, флаг зацикливания, регулирование громкости/пакетный сигнал, значение громкости/размер пакета)*
$4001	**w/o**	**Регистр управления генератором канала 1.** `eppp nsss` - enable sweep, period, negative, shift
$4002	**w/o**	**Регистр управления частотой (периодом) канала 1.** `pppp pppp` - period low *(период сигнала – младшая часть)*
$4003	**w/o**	**Регистр управления частотой (периодом) канала 1.** `llll lppp` - length index, period high *(длительность, период сигнала – старшая часть)*
$4004	**w/o**	**Регистр управления канала 2 (*Прямоугольник [Square]*).** См. описание $4000
$4005	**w/o**	**Регистр управления генератором канала 2.** См. описание $4001
$4006	**w/o**	**Регистр управления частотой канала 2.** См. описание $4002
$4007	**w/o**	**Регистр управления частотой канала 2.** См. описание $4003

$4008	**w/o**	**Регистр управления канала 3 (*Пила [Triangle]*).** `clll llll` - control, linear counter load
$4009	**-**	**-**
$400A	**w/o**	**Регистр управления частотой (периодом) канала 3.** `pppp pppp` - period low *(период сигнала – младшая часть)*
$400B	**w/o**	**Регистр управления частотой (периодом) канала 3.** `llll lppp` - length index, period high *(длительность, период сигнала – старшая часть)*
$400C	**w/o**	**Регистр управления канала 4 (Шум *[Noise]*).** `--le nnnn` - loop env/disable length, env disable, vol/env period
$400D	**-**	**-**
$400E	**w/o**	**Регистр управления частотой (периодом) канала 4.** `s--- pppp` - short mode, period index
$400F	**w/o**	**Регистр управления частотой (периодом) канала 4.** `llll l---` - length index
$4010	**w/o**	**Регистр управления цифровым каналом**. `il-- ffff` - IRQ enable, loop, frequency index
$4011	**w/o**	**Регистр управления громкостью цифрового канала.** `-ddd dddd` - DAC
$4012	**w/o**	**Регистр адреса цифрового канала.** `aaaa aaaa` - sample address
$4013	**w/o**	**Регистр длины блока данных цифрового канала.** `llll llll` - sample length
$4015	**w**	**Регистр управления pAPU.** `---d nt21` **–** Enable: DMC, noise, triangle, pulse 2, 1
	r	**Регистр статуса pAPU.** `if-d nt21` - DMC IRQ, frame IRQ, length counter statuses
$4017 [7-6]	**w/o**	`fd-- ----` - 5-frame cycle, disable frame interrupt

Приложение №1. Система команд 6502.

Рассмотрим систему команд архитектуры 6502. В Денди используется именно эта модель, за исключением блока десятичной арифметики.

Процессор оперирует целыми восьмиразрядными числами. Содержит всего 6 программно-доступных регистров. Из них 5 – восьмиразрядных, и 1 – шестнадцатиразрядный программный счетчик.

A – регистр - аккумулятор. Как и большинство «простых» архитектур – 6502 является «аккумуляторной», то есть во всех, или почти во всех, операциях явно или неявно участвует аккумулятор.

PC – программный счетчик. Единственный 16-ти разрядный регистр – указатель выполняемой команды (классически).

S – регистр - указатель вершины стека (первой свободной ячейки). Стек находится в пространстве адресов 0100h – 01FFh и «растёт» от старшего к младшему.

P – регистр флагов.

P[0] - "C" – "Carry" – флаг переноса из старшего разряда (классически).

P[1] - "Z" – "Zero" – признак нулевого результата (классически).

P[2] - "I" – "Interrupt" – флаг маскирования аппаратного прерывания на линии IRQ – варьируется командами SEI/CLI.

P[3] - "D" – "Decimal" – флаг режима десятичной арифметики. Так как в Денди этот режим отсутствует, то этот флаг не используется процессором, может использоваться программистом. Варьируется командами SED/CLD.

P[4] - "B" – "Break" – флаг программного прерывания (команда BRK).

P[5] - **"1"**

P[6] - "V" – "" – флаг переноса в знаковый разряд (из 6-ого в 7-ой - классически).

P[7] - "N" – "" – флаг знака результата операции – дублирует седьмой разряд (классически).

X, Y – регистры индексной адресации (или просто "общего назначения").

В общем итоге получается регистров для "юзания" всего навсего 3 (A,X,Y) - да и то, многие команды жестко привязаны именно к одному из них, очень часто неудобно бывает, что рождает лишние мнемоники и обращения в память.

Напомним, что система команд процессора оперирует или с перечисленными выше регистрами, или с адресным пространством памяти (64кб) – все устройства, управляемые процессором, должны отображаться своими регистрами на память.

И так – о способах адресации памяти.

ABS - прямая, в команде указывается полный 16-разрядный адрес операнда:

ABS, X (ABX)- индексированная по X, указывается базовый 16-разрядный адрес, к которому прибавляется смещение из регистра X;

ABS, Y (ABY) - индексированная по Y, указывается базовый 16-разрядный адрес, к которому прибавляется смещение из регистра Y.

ACC - аккумуляторная, подразумевает наличие операнда в регистре **A** процессора (что требуется указать в мнемонике).

IMM - непосредственная, 8-разрядный операнд расположен сразу за кодом команды (в мнемонической записи перед непосредственным операндом ставится символ «#»).

IMPL - неявная (указывается лишь мнемоника), операнды не указываются – жесткая логика работы инструкции.

IND - косвенная, задается адрес ячейки памяти, в которой хранится адрес операнда - бывает следующих видов:

IND, X (NDX) - индексно-косвенная, указывается 8-разрядный адрес в нулевой странице (в квадратных скобках), к которому прибавляется содержимое регистра X, после чего из ячейки памяти с вычисленным адресом и следующей за ней извлекается полный 16-разрядный адрес операнда (в мнемонике после адреса через запятую ставится и "X"). Другими словами - в области нулевой страницы создается таблица адресов, а при помощи переменной в регистре "X" можно по ним "бегать";

IND, Y (NDY) - косвенно-индексная, указывается 8-разрядный адрес в нулевой странице (в квадратных скобках), после чего из заданной ячейки памяти и следующей за ней считывается 16-разрядный базовый адрес, к которому прибавляется содержимое регистра Y, и из ячейки с вычисленным адресом извлекается операнд (в мнемонике после адреса через запятую ставится и "Y"). В данном случае переменная в регистре "Y" играет роль смещения относительно адреса базы, хранящегося в нулевой странице;

REL - относительная, в команде указывается 8-разрядное смещение относительно содержимого счетчика команд PC;

ZP - адресация нулевой страницы, в команде задается 8-разрядный адрес, определяющий ячейку памяти нулевой страницы, где хранится операнд;

ZP, X (ZPX) - индексированная по X адресация нулевой страницы, указывается 8-разрядный базовый адрес в нулевой странице, к которому прибавляется содержимое регистра X, и из ячейки памяти с вычисленным адресом извлекается операнд;

ZP, Y (ZPY) - индексированная по Y адресация нулевой страницы, в нулевой странице указывается 8-разрядный базовый адрес, к которому прибавляется содержимое регистра Y, и из ячейки памяти с вычисленным адресом извлекается операнд.

Ниже приведем саму систему команд (Всего 56 мнемоник инструкций).

Мнемоника	Краткое описание	Методы адресации	Запись на языке ассемблера	Код команды	Число байтов	Изменяемые флаги
ADC	Сложение с учетом	IMM	ADC #d8	69 d8	2	V, N, Z, C

	флага переноса: A + d8 + C. Результат в аккумуляторе A и флаге переноса C	ZP	ADC 08	65 a8	2	
		ZP, X	ADC a8, X	75 a8	2	
		ABS	ADC a16	6D a16l a16h	3	
		ABS, X	ADC a16, X	7D a16l a16h	3	
		ABS, Y	ADC a16, Y	79 a16l a16h	3	
		IND, X	ADC (a8, X)	61 a8	2	
		IND, Y	ADC (a8), Y	71 a8	2	
AND	Поразрядное логическое И аккумулятора и операнда	IMM	AND #d8	29 d8	2	N, Z
		ZP	AND a8	25 a8	2	
		ZP, X	AND a8, X	35 a8	2	
		ABS	AND a16	2D a16l a16h	3	
		ABS, X	AND a16, X	3D a16l a16h	3	
		ABS, Y	AND a16, Y	39 a16l a16h	3	
		IND, X	AND (a8, X)	21 a8	2	
		IND, Y	AND (a8), Y	31 a8	2	
ASL	Арифметический сдвиг операнда влево, без начального учета флага C (умножение на 2)	ACC	ASL A	0A	1	N, Z, C
		ZP	ASL a8	06 a8	2	
		ZP, X	ASL a8, X	16 a8	2	
		ABS	ASL a16	0E a16l a16h	3	
		ABS, X	ASL a16, X	1E a16l a16h	3	
BCC	Переход, если флаг C = 0	REL	BCC i8	90 18	2	
BCS	Переход, если флаг C = 1	REL	BCS 18	B0 18	2	
BEQ	Переход, если флаг Z = 1	REL	BEQ 18	F018	2	
BIT	Установка флагов в соответствии с результатом выполнения поразрядного логического И над содержимым	ZP	BIT a8	24 a8	2	N, V, Z
		ABS	BIT a16	2C a16l a16h	3	

	аккумулятора и операнда. Бит 6 результата копируется во флаг V, а бит 7 - во флаг N					
BMI	Переход, если флаг N = 1	REL	BMI 18	3018	2	
BNE	Переход, если флаг Z = 0	REL	BNE 18	D0 18	2	
BPL	Переход, если флаг N = 0	REL	BPL 18	1018	2	
BRK	Программное прерывание	IMPL	BRK	00	1	I
BVC	Переход, если флаг V = 0	REL	BVC 18	5018	2	
BVS	Переход, если флаг V = 1	REL	BVS 18	7018	2	
CLC	Сброс флага C	IMPL	CLC	18	1	C
CLD	Сброс флага D	IMPL	CLD	D8	1	D
CLI	Сброс флага I (разрешение прерываний)	IMPL	CLI	58	1	I
CLV	Сброс флага V	IMPL	CLV	B8	1	V
CMP	Установка флагов в соответствии с результатом вычитания операнда из содержимого аккумулятора	IMM	CMP #d8	C9 d8	2	N, Z, C
		ZP	CMP a8	C5 a8	2	
		ZP, X	CMP a8, X	D5 a8	2	
		ABS	CMP a16	CD a16l a16h	3	
		ABS, X	CMP a16, X	DD a16l a16h	3	
		ABS, Y	CMP a16, Y	D9 a16l a16h	3	
		IND, X	CMP (a8, X)	C1 a8	2	
		IND, Y	CMP (a8), Y	D1 a8	2	
CPX	Установка флагов в соответствии с результатом вычитания операнда из содержимого регистра X	IMM	CPX #d8	E0 d8	2	N, Z, C
		ZP	CPX a8	E4d8	2	
		ABS	CPX a16	EC a16l a16h	3	
CPY	Установка флагов	IMM	CPY #d8	C0 d8	2	N, Z, C

	в соответствии с результатом вычитания операнда из содержимого аккумулятора	ZP	CPY a8	C4 a8	2	
		ABS	CPY a16	CC a16l a16h	3	
DEC	Уменьшение операнда на 1	ZP	DEC a8	C6 a8	2	N, Z
		ZP, X	DEC a8, X	D6 a8	2	
		ABS	DEC a16	CE a16l a16h	3	
		ABS, X	DEC a16, X	DE a16l a16h		
DEX	X = X-1	IMPL	DEX	CA	1	N, Z
DEY	Y = Y-1	IMPL	DEY	88	1	N, Z
EOR	Поразрядное Исключающее ИЛИ содержимого аккумулятора и операнда	IMM	EOR #d8	49 d8	2	N, Z
		ZP	EOR a8	45 a8	2	
		ZP, X	EOR a8, X	55 a8	2	
		ABS	EOR a16	4D a16l a16h	3	
		ABS, X	EOR a16, X	5D a16l a16h	3	
		ABS, Y	EOR a16, Y	59 a16l a16h	3	
		IND, X	EOR (a8, X)	41 a8	2	
		IND, Y	EOR (a8), Y	51 a8	2	
INC	Увеличение операнда на 1	ZP	INC a8	E6 a8	2	N, Z
		ZP, X	INC a8, X	F6 a8	2	
		ABS	INC a16	EE a16l a16h	3	
		ABS, X	INC a16, X	FE a16l a16h	3	
INX	X = X+1	IMPL	INX	E8	1	N, Z
INY	Y = Y+1	IMPL	INY	C8	1	N, Z
JMP	Переход по указанному адресу	ABS	JMP a16	4C a16l a16h	2	
		IND	JMP (a 16)	6C a16l a16h	2	
JSR	Вызов подпрограммы с указанным адресом. В стеке	ABS	JSR a16	20 a16l a16h	3	

	сохраняется только адрес возврата					
LDA	Загрузка операнда в аккумулятор	IMM	LDA #d8	A9d8	2	N, Z
		ZP	LDA a8	A5a8	2	
		ZP, X	LDA a8, X	B5a8	2	
		ABS	LDA a16	AD a16l a16h	3	
		ABS, X	LDA a16, X	BD a16l a16h	3	
		ABS, Y	LDA a16, Y	B9 a16l a16h	3	
		IND, X	LDA (a8, X)	A1 a8	2	
		IND, Y	LDA (a8), Y	B1 a8	2	
LDX	Загрузка операнда в регистр X	IMM	LDX #d8	A2 d8	2	H, Z
		ZP	LDX a8	A6a8	2	
		ZP, Y	LDX a8, Y	B6a8	2	
		ABS	LDX a16	AE a16l a16h	3	
		ABS, Y	LDX a16, Y	BE a16l a16h	3	
LDY	Загрузка операнда в регистр Y	IMM	LDY #d8	A0 d8	2	N, Z
		ZP	LDY a8	A4 a8	2	
		ZP, Y	LDY a8, Y	B4 a8	2	
		ABS	LDY a16	AC a16l a16h	3	
		ABS, Y	LDY a16, Y	BC a16l a16h	3	
LSR	Логический сдвиг операнда вправо (деление на 2)	ACC	LSR A	4A	1	N, Z, C
		ZP	LSR a8	46 a8	2	
		ZP, X	LSR a8, X	56 a8	2	
		ABS	LSR a16	4E a16l a16h	3	
		ABS, X	LSR a16, X	5E a16l a16h	3	
NOP	Нет операции	IMPL	NOP	EA	1	
ORA	Поразрядное логическое ИЛИ содержимого аккумулятора и операнда	IMM	ORA #d8	09 d8	2	N, Z
		ZP	ORA a8	05 a8	2	
		ZP, X	ORA a8, X	15 a8	2	
		ABS	ORA a16	0D a16l	3	

				a16h		
		ABS, X	ORA a16, X	1D a16l a16h	3	
		ABS, Y	ORA a16, Y	19 a16l a16h	3	
		IND, X	ORA (a8, X)	01 a8	2	
		IND, Y	ORA (a8), Y	11 a8	2	
PHA	Помещение содержимого аккумулятора в стек	IMPL	PHA	48	1	
PHP	Помещение регистра состояния в стек	IMPL	PHP	08	1	
PLA	Помещение байта с вершины стека в аккумулятор	IMPL	PLA	68	1	
PLP	Помещение байта с вершины стека в регистр состояния	IMPL	PLP	28	1	Все флаги
ROL	Циклический сдвиг операнда влево	ACC	ROL A	2A	1	N, Z, C
		ZP	ROL a8	26 a8	2	
		ZP, X	ROL a8, X	36 a8	2	
		ABS	ROL a16	2E a16l a16h	3	
		ABS, X	ROL a16, X	3E a16l a16h	3	
ROR	Циклический сдвиг операнда вправо	ACC	ROR A	6A	1	N, Z, C
		ZP	ROR a8	66 a8	2	
		ZP, X	ROR a8, X	76 a8	2	
		ABS	ROR a16	6E a16l a16h	3	
		ABS, X	ROR a16, X	7E a16l a16h	3	
RTI	Возврат из прерывания	IMPL	RTI	40	1	Все флаги
RTS	Возврат из подпрограммы	IMPL	RTS	60	1	
SBC	Вычитание операнда из содержимого аккумулятора с	IMM	SBC #d8	E9 d8	2	N, V, Z, C
		ZP	SBC a8	E5 a8	2	
		ZP, X	SBC a8, X	F5 a8	2	

	учетом флага переноса	ABS	SBC a16	ED a16l a16h	3	
		ABS, X	SBC a16, X	FD a16l a16h	3	
		ABS, Y	SBC a16, Y	F9 a16l a16h	3	
		IND, X	SBC (a8, X)	E1 a8	2	
		IND, Y	SBC (a8), Y	F1 a8	2	
SEC	Установка флага C	IMPL	SEC	38	1	C
SED	Установка флага D	IMPL	SED	F8	1	D
SEI	Установка флага I (запрещение прерываний)	IMPL	SEI	78	1	I
STA	Запись содержимого аккумулятора в память	ZP	STA a8	85 a8	2	
		ZP, X	STA a8, X	95 a8	2	
		ABS	STA a16	8D a16l a16h	3	
		ABS, X	STA a16, X	9D a16l a16h	3	
		ABS, Y	STA a16; Y	99 a16l a16h	3	
		IND, X	STA (a8, X)	81 a8	2	
		IND, Y	STA (a8)	91 a8	2	
STX	Запись содержимого регистра X в память	ZP	STX a8	86 a8	2	
		ZP, Y	STX a8, Y	96 a8	2	
		ABS	STX a16	8E a16l a16h	3	
STY	Запись содержимого регистра Y в память	ZP	STY a8	84 a8	2	
		ZP, X	STY a8, X	94 a8	2	
		ABS	STY a16	8C a16l a16h	3	
TAX	Пересылка содержимого аккумулятора в регистр X	IMPL	TAX	AA	1	N, Z
TAY	Пересылка содержимого аккумулятора в регистр Y	IMPL	TAY	A8	1	N, Z
TSX	Пересылка содержимого	IMPL	TSX	BA	1	N, Z

	указателя стека в регистр X					
TXA	Пересылка содержимого регистра X в аккумулятор	IMPL	TXA	8A	1	N, Z
TXS	Пересылка содержимого регистра X в указатель стека	IMPL	TXS	9A	1	
TYA	Пересылка содержимого регистра Y в аккумулятор	IMPL	TYA	98	1	N, Z

Условные обозначения:

a16 - 16-разрядный адрес

a16h - старший байт 16-разрядного адреса

a16l - младший байт 16-разрядного адреса

a8 - 8-разрядный адрес в нулевой странице

d8 - непосредственный 8-разрядный операнд

i8 - 8-разрядное смещение в диапазоне от -128 до 127

Также приведем матрицу системы команд (красным цветом отмечены недокументированные инструкции – в базовой архитектуре 6502 они отсутствуют, но контроллером 6561, вероятно, поддерживаются).

76543210	xxx000xx	xxx010xx	xxx100xx	xxx110xx	xxx001xx	xxx011xx	xxx101xx	xxx111xx
000xxx00	BRK IMM	PHP	BPL REL	CLC	DOP ZP	TOP ABS	DOP ZPX	TOP ABX
001xxx00	JSR ABS	PLP	BMI REL	SEC	BIT ZP	BIT ABS	DOP ZPX	TOP ABX
010xxx00	RTI	PHA	BVC REL	CLI	DOP ZP	JMP ABS	DOP ZPX	TOP ABX
011xxx00	RTS	PLA	BVS REL	SEI	DOP ZP	JMP IND	DOP ZPX	TOP ABX
100xxx00	DOP IMM	DEY	BCC REL	TYA	STY ZP	STY ABS	STY ZPX	SYA ABX
101xxx00	LDY IMM	TAY	BCS REL	CLV	LDY ZP	LDY ABS	LDY ZPX	LDY ABX
110xxx00	CPY IMM	INY	BNE REL	CLD	CPY ZP	CPY ABS	DOP ZPX	TOP ABX
111xxx00	CPX IMM	INX	BEQ REL	SED	CPX ZP	CPX ABS	DOP ZPX	TOP ABX
000xxx01	ORA NDX	ORA IMM	ORA NDY	ORA ABY	ORA ZP	ORA ABS	ORA ZPX	ORA ABX
001xxx01	AND NDX	AND IMM	AND NDY	AND ABY	AND ZP	AND ABS	AND ZPX	AND ABX
010xxx01	EOR NDX	EOR IMM	EOR NDY	EOR ABY	EOR ZP	EOR ABS	EOR ZPX	EOR ABX
011xxx01	ADC NDX	ADC IMM	ADC NDY	ADC ABY	ADC ZP	ADC ABS	ADC ZPX	ADC ABX
100xxx01	STA NDX	DOP IMM	STA NDY	STA ABY	STA ZP	STA ABS	STA ZPX	STA ABX

101xxx01	LDA NDX	LDA IMM	LDA NDY	LDA ABY	LDA ZP	LDA ABS	LDA ZPX	LDA ABX
110xxx01	CMP NDX	CMP IMM	CMP NDY	CMP ABY	CMP ZP	CMP ABS	CMP ZPX	CMP ABX
111xxx01	SBC NDX	SBC IMM	SBC NDY	SBC ABY	SBC ZP	SBC ABS	SBC ZPX	SBC ABX
000xxx10	KIL	ASL A	KIL	NOP	ASL ZP	ASL ABS	ASL ZPX	ASL ABX
001xxx10	KIL	ROL A	KIL	NOP	ROL ZP	ROL ABS	ROL ZPX	ROL ABX
010xxx10	KIL	LSR A	KIL	NOP	LSR ZP	LSR ABS	LSR ZPX	LSR ABX
011xxx10	KIL	ROR A	KIL	NOP	ROR ZP	ROR ABS	ROR ZPX	ROR ABX
100xxx10	DOP IMM	TXA	KIL	TXS	STX ZP	STX ABS	STX ZPY	SXA ABY
101xxx10	LDX IMM	TAX	KIL	TSX	LDX ZP	LDX ABS	LDX ZPY	LDX ABY
110xxx10	DOP IMM	DEX	KIL	NOP	DEC ZP	DEC ABS	DEC ZPX	DEC ABX
111xxx10	DOP IMM	NOP	KIL	NOP	INC ZP	INC ABS	INC ZPX	INC ABX
000xxx11	SLO NDX	AAC IMM	SLO NDY	SLO ABY	SLO ZP	SLO ABS	SLO ZPX	SLO ABX
001xxx11	RLA NDX	AAC IMM	RLA NDY	RLA ABY	RLA ZP	RLA ABS	RLA ZPX	RLA ABX
010xxx11	SRE NDX	ASR IMM	SRE NDY	SRE ABY	SRE ZP	SRE ABS	SRE ZPX	SRE ABX
011xxx11	RRA NDX	ARR IMM	RRA NDY	RRA ABY	RRA ZP	RRA ABS	RRA ZPX	RRA ABX
100xxx11	AAX NDX	XAA IMM	AXA NDY	XAS ABY	AAX ZP	AAX ABS	AAX ZPY	AXA ABY
101xxx11	LAX NDX	ATX IMM	LAX NDY	LAR ABY	LAX ZP	LAX ABS	LAX ZPY	LAX ABY
110xxx11	DCP NDX	AXS IMM	DCP NDY	DCP ABY	DCP ZP	DCP ABS	DCP ZPX	DCP ABX
111xxx11	ISC NDX	SBC IMM	ISC NDY	ISC ABY	ISC ZP	ISC ABS	ISC ZPX	ISC ABX

Приложение №2. Палитра в RGB.

Видеопроцессор, примененный в Денди, работает не в RGB режиме, поэтому палитра относительно составляющих по RGB может показаться весьма странной. В палитре содержится 64 ячейки (некоторые содержат одинаковые цвета). Соответствие цветов палитры формату RGB приведено ниже.

	R	G	B		R	G	B		R	G	B		R	G	B
00	117	117	117	10	188	188	188	20	255	255	255	30	255	255	255
01	39	27	143	11	0	115	239	21	63	191	255	31	171	231	255
02	0	0	171	12	35	59	239	22	95	151	255	32	199	215	255
03	71	0	159	13	131	0	243	23	167	139	253	33	215	203	255
04	143	0	119	14	191	0	191	24	247	123	255	34	255	199	255
05	171	0	19	15	231	0	91	25	255	119	183	35	255	199	219
06	167	0	0	16	219	43	0	26	255	119	99	36	255	191	179
07	127	11	0	17	203	79	15	27	255	155	59	37	255	219	171
08	67	47	0	18	139	115	0	28	243	191	63	38	255	231	163
09	0	71	0	19	0	151	0	29	131	211	19	39	227	255	163
0A	0	81	0	1A	0	171	0	2A	79	223	75	3A	171	243	191
0B	0	63	23	1B	0	147	59	2B	88	248	152	3B	179	255	207
0C	27	63	95	1C	0	131	139	2C	0	235	219	3C	159	255	243

0D	0	0	0	1D	0	0	0	2D	0	0	0	3D	0	0	0
0E	0	0	0	1E	0	0	0	2E	0	0	0	3E	0	0	0
0F	0	0	0	1F	0	0	0	2F	0	0	0	3F	0	0	0

Приложение №3. Перечень мапперов.

Перед написанием любой программы для Денди, программисту необходимо сделать предварительную оценку объёма программы и необходимости применения прочих аппаратно-программных средств, с тем, чтобы определиться под какой тип маппера писать программу (если он конечно нужен вообще).

Мапперов существует бесчисленное множество, наиболее часто используемые – около десяти. Каждый маппер имеет условное название. Для совместимости дампов и эмуляторов мапперы пронумерованы, а заголовок iNes стал стандартом де-факто. Приведем список типов мапперов (из документации на различные эмуляторы) с наиболее известными играми их использующими.

Номер по iNES	Название	Примеры игр
0	Without mapper	Battle City, Balloon Fight, Mario Bros, Sky Destroyer, Lode Runner
1	Nintendo **MMC1**	Snake Rattle & Roll, Chip & Dale, Darkwing Duck, RoboCop 2, RoboCop 3, Operation Wolf
2	Nintendo **UNROM** (PRG Switch)	Prince of Persia, Contra (U), Little Mermaid, Mega Man, Duck Tales, Duck Tales 2, Bomberman 2, Castelian
3	Nintendo **CNROM** (CHR Switch)	Castle Excellent, Tetris
4	Nintendo **MMC3**	Dracula, Jurassic Park, Mighty Final Fight, Contra Force, Captain America and The Avengers, Ninja Crusaders, Ninja Gaiden 2, Ninja Gaiden 3, Batman, Vice, Double Dragon 2, Double Dragon 3, Tiny Toon Adventures,Tiny Toon Adventures 2, Contra (J)
5	Nintendo **MMC5**	Castlevania 3, Romance of The Three Kingdoms II, Just Breed, Bandit Kings of Ancient China
6	FFE F4 Series	
7	Rare's **AOROM**	Battletoads, Battletoads & Double Dragon, Cabal, Captain Skyhawk
8	FFE F3 Series	
9	Nintendo MMC2	Punchout!
10	Nintendo MMC4	Fire Emblem, Fire Emblem Gaiden
11	Color Dreams	Crystal Mines, Bible Adventures
12	FFE F6 Series	Dragon Ball Z 5 ("bootleg" original)
13	CPROM	Videomation
15	Multi-cart(bootleg)	100-in-1: Contra Function 16
16	Bandai	Dragon Ball Z, SD Gundam Gaiden
17	FFE F8 Series	
18	Jaleco SS806	Pizza Pop, Plasma Ball
19	Namco 106/129/163	Splatter House, Mappy Kids
20	Famicom Disk System (FDS)	

21	Konami VRC4 2A	WaiWai World 2, Ganbare Goemon Gaiden 2
22	Konami VRC4 1B	Twinbee 3
23	Konami VRC4 2B	WaiWai World, Crisis Force
24	Konami **VRC6**	Akumajou Densetsu
25	Konami **VRC4**	Gradius 2, Bio Miracle
26	Konami VRC6 A0-A1 Swap	Esper Dream 2, Madara
32	IREM G-101	Image Fight 2, Perman
33	Taito TC0190/TC0350	Don Doko Don
34	NINA-001 and BNROM	Impossible Mission 2, Deadly Towers, Bug Honey
40	(bootleg)	Super Mario Bros. 2
41	Caltron 6-in-1	Caltron 6-in-1
42	(bootleg)	Mario Baby
44	Multi-cart(bootleg)	Super Hik 7 in 1 (MMC3)
45	Multi-cart(bootleg)	Super Hik X in 1 (MMC3)
46	Game Station	Rumble Station
47	Multi-cart(bootleg)	2 in 1 (MMC3)
48	Taito TC190V	Flintstones
49	Multi-cart(bootleg)	Super HiK 4 in 1 (MMC3)
50	(bootleg)	Super Mario Bros. 2
51	Multi-cart(bootleg)	11 in 1 Ball Games
52	Multi-cart(bootleg)	Super Hik 7 in 1 (MMC3)
57	Multi-cart(bootleg)	Game Star GK-54
58	Multi-cart(bootleg)	68-in-1 Game Star HKX5268
60	Multi-cart(bootleg)	4 in 1(Reset-selected)
61	Multi-cart(bootleg)	20 in 1
62	Multi-cart(bootleg)	Super 700 in 1
64	Tengen RAMBO 1	Klax, Rolling Thunder, Skull and Crossbones
65	IREM H-3001	Daiku no Gensan 2
66	GNROM	SMB/Duck Hunt
67	Sunsoft	Fantasy Zone 2
68	TENGEN	After Burner 2, Nantetta Baseball
69	Sunsoft FME-7	Batman: Return of the Joker, Hebereke
70	Bandai	Kamen Rider Club
71	Camerica	Fire Hawk, Linus Spacehead
72	Jaleco	Pinball Quest
73	Konami VRC3	Salamander
74	Taiwanese MMC3 CHR ROM w/ VRAM	Super Robot Wars 2
75	Jaleco SS8805/Konami VRC1	Tetsuwan Atom, King Kong 2
76	Namco 109	Megami Tensei
77	Irem VRAM (Four Screen)	Napoleon Senki

78	Irem 74HC161/32	Holy Diver
79	NINA-06/NINA-03	F15 City War, Krazy Kreatures, Tiles of Fate
80	Taito X-005	Minelvation Saga
82	Taito X1-17	Kyuukyoku Harikiri Stadium - Heisei Gannen Ban
83	Multi-cart(bootleg)	Dragon Ball Party
85	Konami **VRC7**	Lagrange Point
86	Jaleco JF-13	More Pro Baseball
87	Jaleco/Konami	Argus
88	Namco 118	Dragon Spirit
89	Sunsoft Early	Mito Koumon
90	Pirate MMC5-style	Aladdin, Super Mario World
91	HK-SF3	Mari Street Fighter 3 Turbo
92	Jaleco Early	MOERO Pro Soccer
93	UNROM-style	Fantasy Zone
94	UNROM-style	Senjou no Ookami
95	Namco MMC3-Style	Dragon Buster
96	Bandai	Oeka Kids
97	Irem – PRG HI	Kaiketsu Yanchamaru
99	VS System 8KB VROM Switch	VS SMB, VS Excite Bike
105	NES-EVENT	Nintendo World Championships
107	??	Magic Dragon
112	Asder	Sango Fighter, Hwang Di
113	MB-91	Deathbots
114	??	The Lion King
115	??	Yuu Yuu Hakusho Final
117	??	San Guo Zhi 4
118	MMC3-TLSROM/TKSROM Board	Ys 3, Goal! 2, NES Play Action Football
119	MMC3-TQROM Board	High Speed, Pin*Bot
140	Jaleco ??	Bio Senshi Dan
144	??	Death Race
151	Konami VS System Expansion	VS The Goonies, VS Gradius
152	??	Arkanoid 2, Saint Seiya Ougon Densetsu
153	Bandai ??	Famicom Jump 2
154	Namco ??	Devil Man
155	MMC1 w/o normal WRAM disable	The Money Game, Tatakae!! Rahmen Man
156	??	Buzz and Waldog
157	Bandai Datach ??	Datach DBZ, Datach SD Gundam Wars, **EEPROM NOT SUPPORTED
158	RAMBO 1 Derivative	Alien Syndrome
180	??	Crazy Climber

182	??	Super Donkey Kong
184	??	Wing of Madoola, The
189	??	Thunder Warrior, Street Fighter 2 (Yoko)
193	Mega Soft	Fighting Hero
200	Multi-cart(bootleg)	1200-in-1
201	Multi-cart(bootleg)	21-in-1
202	Multi-cart(bootleg)	150 in 1
203	Multi-cart(bootleg)	35 in 1
206	DEIROM	Karnov
207	Taito ??	Fudou Myouou Den
208	??	Street Fighter IV (by Gouder)
209	??	Mortal Kombat 3 - Special 56 Peoples
210	Namco ??	Famista '92, Famista '93, Wagyan Land 2
225	Multi-cart(bootleg)	58-in-1/110-in-1/52 Games
226	Multi-cart(bootleg)	76-in-1
227	Multi-cart(bootleg)	1200-in-1
228	Action 52	Action 52, Cheetahmen 2
229	Multi-cart(bootleg)	31-in-1
230	Multi-cart(bootleg)	22 Games
231	Multi-cart(bootleg)	20-in-1
232	BIC-48	Quattro Arcade, Quattro Sports
234	Multi-cart ??	Maxi-15
235	Multi-cart(bootleg)	Golden Game 150 in 1
240	??	Gen Ke Le Zhuan, Shen Huo Le Zhuan
242	??	Wai Xing Zhan Shi
244	??	Decathalon
246	??	Fong Shen Ban
248	??	Bao Qing Tian
249	Waixing ??	??
250	??	Time Diver Avenger
255	Multi-cart(bootleg)	115 in 1

Документация на мапперы от разработчиков игр недоступна (оно и понятно). Зато есть масса информации от разработчиков эмуляторов (но большинство в виде исходников - без каких либо комментариев). К счастью есть еще и «неофициальная» документация - ясно дело в лучшем случае на английском ... переводом заниматься пока желания нет, ссылки на оригиналы документов в последней главе.

Приложение №4. Разъём картриджа.

Фронтальная сторона разъёма (вид сверху) – справа. Тыльная – слева. Цвета на рисунке несут информацию о том, к каким шинам подключается та или иная линия разъема картриджа (см. легенду рис.1) Белым цветом обозначены линии земли (GND) и питания (VCC=5В), а также некоторые служебные сигналы, а именно (**направление сигналов** In/Out **разъема картриджа, по отношению схемотехнике картриджа**):

GND	1	31	VCC
A11	2	32	#M2
A10	3	33	A12
A9	4	34	A13
A8	5	35	A14
A7	6	36	D7
A6	7	37	D6
A5	8	38	D5
A4	9	39	D4
A3	10	40	D3
A2	11	41	D2
A1	12	42	D1
A0	13	43	D0
R/#W	14	44	#ROM CS
#IRQ	15	45	SND In
GND	16	46	SND Out
#VRAM OE	17	47	#VRAM WE
VRAM A10	18	48	#VRAM CS
PA6	19	49	#PA13
PA5	20	50	PA7
PA4	21	51	PA8
PA3	22	52	PA9
PA2	23	53	PA10
PA1	24	54	PA11
PA0	25	55	PA12
PD0	26	56	PA13
PD1	27	57	PD7
PD2	28	58	PD6
PD3	29	59	PD5
VCC	30	60	PD4

Линия **#M2** (In) – выход сигнала опорной (тактовой) частоты с процессора (1.78МГц – т.е. частота внешнего тактового генератора /12, скважность 62.5%), используемого схемотехникой всей приставки. В рамках картриджа используется мапперами, в том числе для того, чтобы определить момент нажатия клавиши «Reset» (для переключения на другую игру и/или инициализации состояния регистров маппера), а также для отсчета временных интервалов и пр.

Линии **SND In, SND Out** – аналоговый выход с pAPU (In) и вход (Out) на аудио выход (о как!) приставки. В 99 случаях из 100 эти контакты замыкаются перемычкой (на картридже). Тем самым, коммутируя звук, формируемый аудио сопроцессором, на выходной разъем приставки (короче слышим то, что играет pAPU). Альтернатива – картридж содержит свой звуковой процессор, подключаемый к выходу приставки (SND Out). pAPU в этом случае отключен – выход (SND In) «висит в воздухе» (или может стоять программно-управляемый коммутатор), хотя теоретически может использоваться и микшер. Некоторые «китайские» приставки «зарубают» подобную фичность на корню – перемычка стоит на плате приставки, или эти контакты просто не выведены на разъём картриджа (выход pAPU с процессора скоммутированы на выход приставки в обход разъема картриджа).

Далее опишем назначение сигналов (к которым, по-моему, требуются пояснения):

Линия **#VRAM WE** (In) используется лишь в случае использования CHR-RAM (ОЗУ) – в качестве памяти под знакогенераторы. Иначе картридж просто его не использует.

Линии **PA0-PA13** (In) подключены к шине адреса PPU (#PA13 – дополнительно через инвертор). Но линия **VRAM A10** (Out) !!! – подключена к адресной линии A10 микросхемы VRAM (микросхемы ОЗУ экранных страниц, т.е. видеопамяти) на приставке. Управляет ей или маппер, задавая тем самым способ отражения VRAM (горизонтальное/вертикальное), или режим отражения может быть строго определенным - в этом случае на картридже запаиваются перемычки (см. главу «Видеопроцессор Денди (PPU)»).

Линии **#PA13** (In) и **#VRAM CS** (Out) замыкаются (в 99 случаях из 100) перемычкой. При активном сигнале на линии #VRAM CS (0, т.е. #PA13=0 или PA13=1, а это адреса $2000-$3FFF адресного пространства PPU) возможна работа с VRAM(2k), установленной в приставке. На картриджах с дополнительными 2k VRAM перемычки нет. #PA13 картридж не использует, а линией #VRAM CS управляет контроллер страниц (маппер). При #VRAM CS = 1 – отключается VRAM приставки и маппер должен «подставлять» в адресное пространство VRAM, располагающуюся на картридже.

Приложение №5. Схемотехника.

Схемотехника Денди по большей степени классична для любой ЭВМ: содержит микросхему процессора, видеопроцессора, оперативной памяти и прочей сопрягающей "рассыпухи". В 80-х годах прошлого столетия консоли Famicom и их «клоны» изготавливались именно в «многокорпусном варианте». Один из типовых вариантов принципиальной схемы приведен ниже (при клике на картинку открывается полномасштабный вариант - чуть более 100кб.)

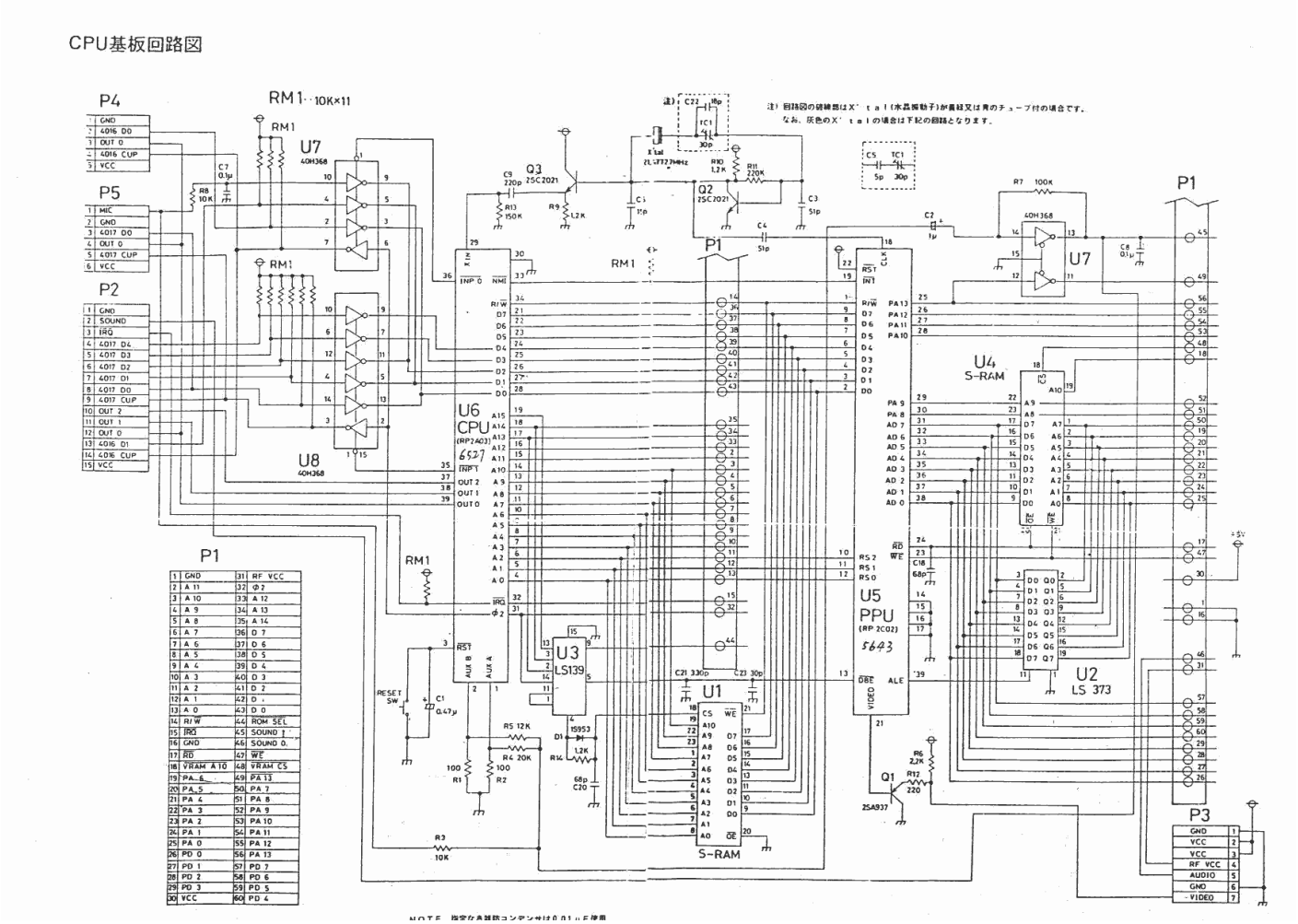


Схема 1.

Несмотря на то, что документ явно «китайского» происхождения, на нем «один-в-один» изображена схема японского Famicom редакции HVC-CPU-07 (проверено). По-сему схему можно использовать не только в целях ремонта и т.п. но и сконструировать по ней систему «с нуля».

Массовую популярность на российском рынке Денди приобрела в большей степени (чем китайцам) благодаря компании "Стиплер", - которая, по всей видимости, имела "пиратские" корни. Лично я очень сильно сомневаюсь в том, что Стиплер делала лицензионные отчисления в пользу Nintendo. Потому что как тогда, для примера, объяснить само возникновение брэнда "Денди"? - а имя оригинального разработчика, кстати, всеми усилиями скрывалось. Или, например то, что в один прекрасный момент эта фирма (казалось бы так раскрученная) внезапно исчезла? (Можете попробовать поискать на гугле ...) Ну да разговор не об том. Внезависимости, занималась ли реально Стиплер работой над схемотехнической архитектурой консолей, либо же просто заказывала OEM-партии клона Famicom под своим логотипом - продукция была действительно высокого качества, не в сравнение морю явных "китайских" подделок (скупаемых "там" за копейки и мешками завозимыми к нам). Наиболее известными

"ремэйками" консоли NES в исполнении Стиплер были модели "Dendy Junior" и "Dendy Classic" отличались они лишь дизайном (ну и некоторыми малосущественными нюансами), хотя "классик" стоила несколько дороже. Кстати именно дизайн Dendy Junior являлся точной копией японского Famicom, а дизайн Dendy Classic повторял дизайн «китайских» клонов (или наоборот?).

В месте с тем, за весь период своего производства "начинка" консолей (как стиплеровских так и «китайских») претерпела определенные изменения (не заметные для рядового пользователя, не отразившиеся на внешнем виде изделия и базовых функциях). Интеграция и миниатюризация в электронном мире идет огромными шагами. Первые редакции консоли были «многокорпусными» (см. схему выше) – это и все японские Famicom`ы, и первые Dendy от Стиплера, и даже китайщина концов 80-х начала 90-х годов прошлого века. Но уже к середине 90-х годов купить новую «многокорпусную» Денди было почти не реально. Первая Денди ("Junior") которую я увидел изнутри (не моя – отдали на ремонт) уже была собрана в соответствии с концепцией System-on-a-Chip (система на одном кристалле) - лишь микросхемы памяти были "внешними" - сейчас есть в моей коллекции подобная система, но в исполнении "Classic" (система PAL - на чипе 1818).

Пару слов (и картинок) о моей первой дендюшке ("Junior II"), той самой - купленной в 1995 году. Это последняя вариация консоли в исполнении "Junior" (внешний вид как у Famicom, см. картинку в уголке) - абсолютно все компоненты системы находятся в едином чипе (включая память). Фотографии обоих сторон платы приведены ниже (плата однослойная, с лицевой стороны есть лишь отдельные проволочные перемычки).

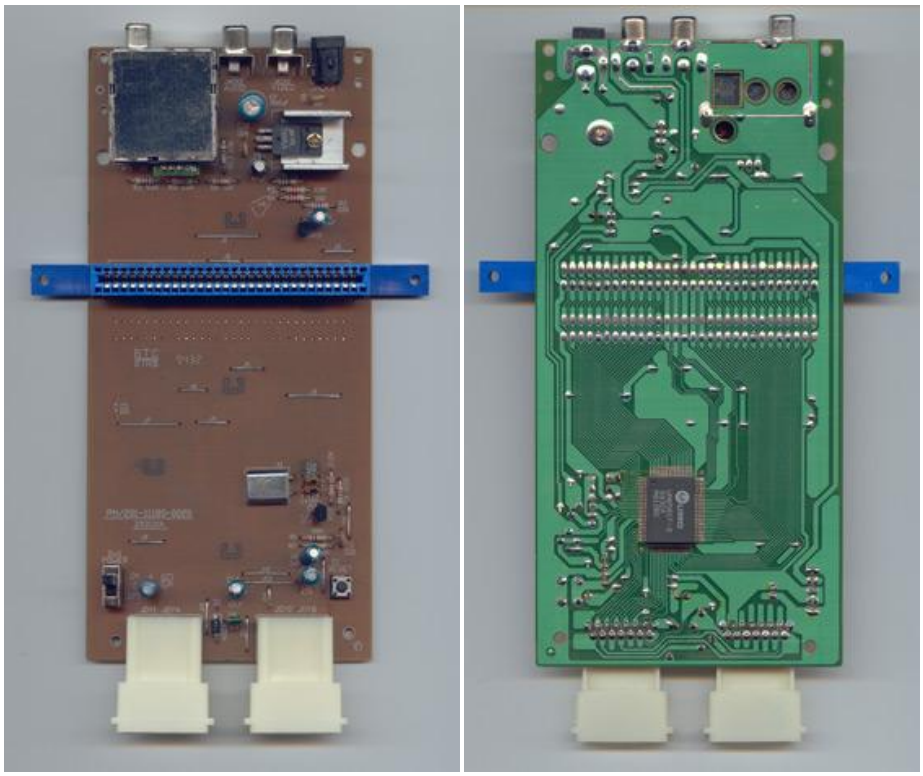


Рисунок 1.

Ну а далее ... конечно же, я срисовал схему устройства (что не сложно, но весьма кропотливо) - результат перед вами ниже (при клике на картинку открывается полномасштабный вариант - около 400кб.)

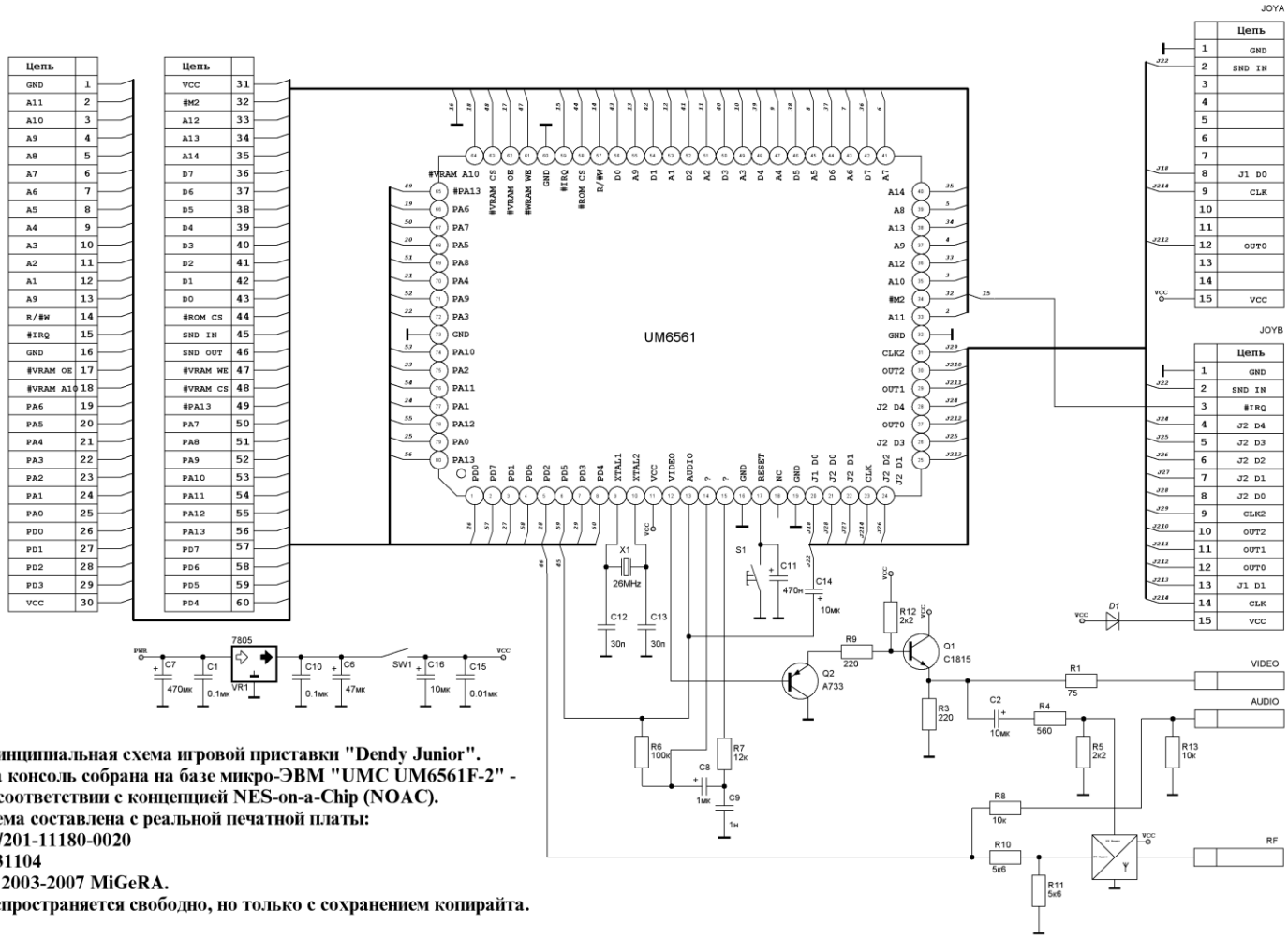
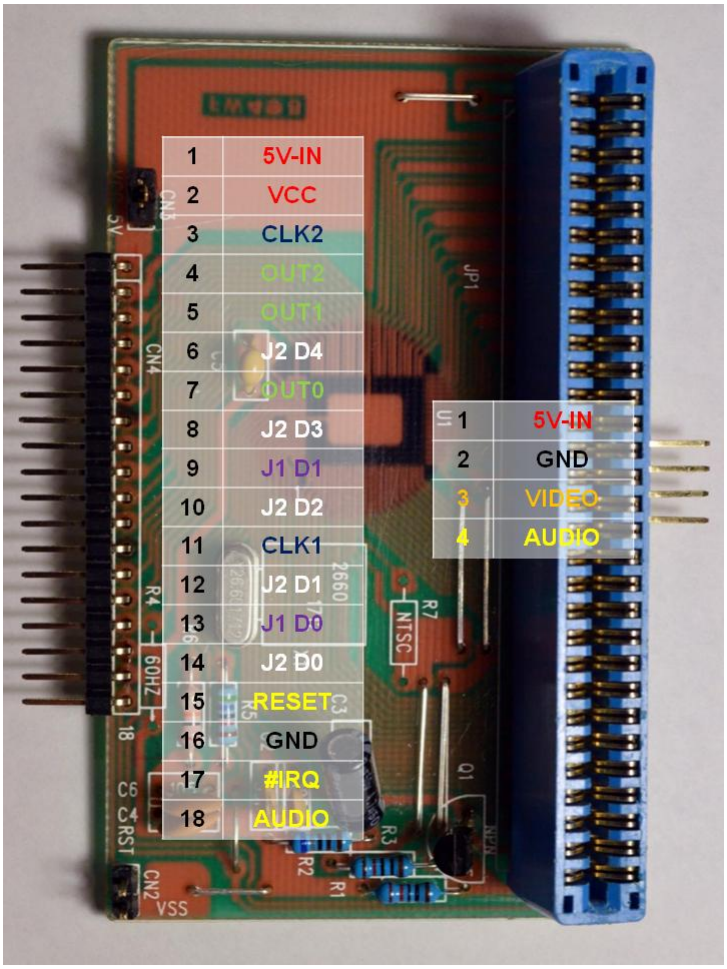


Схема 2.

Устройство предельно простО. Микро-ЭВМ с необходимой аналоговой обвязкой плюс разъемы - ничего лишнего. Когда зарисовывал схему, складывалось впечатление, что распиновка чипа специально создавалась под конкретную топологию печатной платы, с целью минимизации всех возможных переходов и соединений. Что лишний раз наталкивает на мысль - микросхема UM6561 - есть изделие заказное (а не универсальное, массового производства). В эту пользу говорит и факт отсутствия какой-либо технической документации на данную микросхему, как у самого разработчика - UMC (кстати, достаточно крупного), так и во всевозможных информационных базах. Интересен еще и другой вопрос - "под чей заказ" делалась эта Микро-ЭВМ. Уж не сами ли Стиплер ее заказывали? ;-) С другой стороны - маловероятным кажется и то, что такой крупный мировой производитель электронной базы как UMC приняли заказ на изготовление "чипа" - клона консоли NES (разумеется, при полном отсутствие у заказчика каких любо прав на эту консоль). Тогда под какое применение (официально) был этот чип заказан? - и сколько у него "недокументированных" возможностей? Ладно, оставим мистику в стороне ;-) Хотя о какой "документированности" можно говорить при полном отсутствие какой бы то ни было документации ... Все что можно однозначно сказать о Микро-ЭВМ UM6561 - так это то, что в приведенном на схеме включении она полностью (нареканий вроде пока нет) эмулирует работу игровой консоли NES.

Уже в те годы китайцы жмотились на текстолит (хотя и использовали более качественный, нежели в Денди), а также имели неуемную тягу к бескорпусным микросхемам («кляксам»). Почти любая китайская консоль заката эпохи Денди – вторая половина 90-х годов века двадцатого, была сконструирована в виде однокристаллки подобной приведенной выше Dendy Junior II, но в бескорпусном варианте.



На картинке выше фотка центральной платы одной из китайских консолей, внезапно приказавшей долго жить (в далеком 1995-ом) и после почти 20-летнего ожидания в закромах – восстановленной и в настоящий момент вполне себе работоспособной ;-) Трабл был в отказавшем кварце, под замену которого был специально приобретен десяток идентичных ;-) Для удобства тестирования впаяны штыревые гребенки, поменяны резисторы, установлен транзистор …

Расскажу еще о более интересном - о многокорпусных консолях. В 2014 году вновь вспомнилась уже порядком подзабытая тематика о Денди и были куплены у япошек несколько б/у Фамикомов (обзорные материалы о данных консолях размещены отдельно). Здесь же коснемся архитектурной части. Логично, что Фамиком – т.е. NES, рассчитанная на японского потребителя, формирует сигнал в формате NTSC, а радиочастотный модулятор настроен на тамошюю частотную сетку (90 или 96 MHz). Формат формируемого видео-сигнала определяется вариантом (экземпляром) видеопроцессора (PPU). Т.к. схемотехнически различные варианты микросхем PPU идентичны, есть возможность заменить PPU (в случае «многокорпусной» консоли) и тем самым изменить стандарт формируемого видео-сигнала. Так например можно заPALить Фамиком … Вместе с процессором нужно будет поменять и «кварц», а в ряде случаев и сам процессор (CPU). Варианты «наборов», позволяющих реализовать получение выходного сигнала того или иного формата представим в виде таблицы.

Формат сигнала	CPU	PPU	Кварц
NTSC	Ricoh RP2A03 (G,E)	Ricoh RP2C02 (G,E)	21.47727 MHz
PAL	Ricoh RP2A07	Ricoh RP2C07	26.601712 MHz
NTSC	UMC UA6527	UMC UA6528	21.47727 MHz
PAL	UMC UA6527 (P)	UMC UA6538	26.601712 MHz
SECAM	UMC UM6557	UMC UM6558 + UM6559	21.312516 MHz

Приложение №6. История, Ссылки, Прочее.

В ру.нете достаточно много сайтов посвященных эмуляции, но там инфы по-существу никакой (многие просто "передирают" общие фразы друг у друга), разве что эмуляторов и дампов игрушек накачать можно … Но с годами ситуация все же чуть-чуть но меняется в лучшую сторону ;-)

Ссылки.

Общие материалы.

- NesDev Wiki – Онлайн сборник материалов и документации по NES (сейчас там много всего разного, лет 10-15 назад, когда я писал этот труд - такого не было) [англ.];
- Коллекция документации – сборник документации по NES начала 2000-х (как правило txt-формат), многое из которой легло в основу данного труда NES [англ.];

Видеопроцессор Денди (PPU) - нюансы использования и программирования.

- Устройство спецэффектов для игр под NES – Статья о принципах использования возможностей PPU для практической реализации графических эффектов [рус.];
- Устройство спецэффектов для игр под NES (продолжение);
- Решение по замене PPU на FPGA ради получения сигнала в формате RGB – Заметка с форума [англ.];

Звуковой сопроцессор Денди (pAPU) - нюансы использования и программирования.

- Sound Hardware;

Маппер и прочие компоненты картриджа.

- Сборник спецификаций мапперов;

Эмуляция Денди - околосистемные вопросы.

- Mesen – без преувеличения, лучший на сегодняшний день (2019г.) эмулятор NES с потрясающе функциональным отладчиком и на удивление исчерпывающей (для бесплатного продукта) документацией!
- Аппаратная эмуляция компонентов системы - Исходники;
- NES реализация на FPGA – Обзорный материал, как факт, без исходников и т.п.;

История.

Как создавался этот материал (в обратном хронологическом порядке):

июль – август 2016 года – Компиляция различных накопившихся корректировок и дополнений, общестилистические правки по тексту (Главы 1, 3, 4);
ноябрь 2014 года – Корректировки в плане уточнения деталей по тексту, расширения Приложения 5;
март 2008 года – Корректировки, добавление ссылок;
октябрь 2007 года - Переоформление и корректировка графического материала, исправление мелких неточностей;
сентябрь 2007 года - Добавление схемы Dendy Junior, переработка Приложения 5;
июнь 2005 года - Добавление раздела о звуковом сопроцессоре pAPU - Глава 5, общая корректировка докумета;

2003 год - Практическое изучение консоли, начальное создание всей этой документации и т.п.

Удачи.

Продолжение и дополнения следуют …

Автор не несет ответственности за корректность информации, хотя прикладывал все усилия, чтобы последняя была максимально точной, корректной и доступной для понимания читателем.

Последняя редакция от 2019 года