



HDD REPAIR TOOL (HRT)

Общая документация по комплексу

Даже идеальное качество молотка не гарантирует, что он не будет гнуть гвозди и даже того, что Вы не попадёте им себе по пальцу. Всё зависит от качества гвоздей и от Вашего мастерства

Введение

Все накопители чем-то похожи друг на друга. По этой причине, разделы документации разных частей комплекса HRT заимствуют те или иные части друг у друга, ведь суть – одна и та же. Однако, в кое-каких деталях, накопители имеют и отличия. Авторы постарались указать на них в тексте.

Отсюда следует вывод, что если Вы видите главу (а иногда – и подраздел), знакомый Вам по документации на другие части комплекса, то не пропускайте её, как уже знакомую. Не исключено, что в ней есть некоторые отличия от аналогов, помещённых в других книгах, прилагающихся к комплексу.

Предостережения

а) Программа рассчитана на подготовленного пользователя. Она позволяет изменять различные системные параметры накопителей. Авторы не несут НИКАКОЙ ответственности за последствия Ваших действий. Никакой ущерб, связанный с применением программы, не будет возмещён.

б) Программа не накладывает никаких ограничений на Ваши действия. В противном случае, есть опасность, что программ запретит Вам применить какую-либо, придуманную лично Вами новую методику, о которой авторы и не подозревали. Разумеется, если бы было ограничение, то программа могла заблокировать Ваши действия.

Однако, с другой стороны, некоторые ошибочные действия могут привести к фатальным последствиям (вплоть до порчи ремонтируемого накопителя). Авторы не ставили перед собой задачи использования программы неподготовленным пользователем. Наоборот, авторы постарались дать максимальную свободу тем, кто старается «ухватиться за соломинку» при ремонте накопителей.

Таким образом, программа НЕ ЗАПРЕЩАЕТ задавать любые, даже самые экзотические параметры накопителей. Ваша задача – внимательно проанализировать возможные последствия от предполагаемых действий.

в) Программа постоянно развивается и может содержать как пункты меню, так и элементы диалогов, не описанные в данном руководстве.

Главное окно программы

После запуска, программа выводит на экран панель управления, представленную на Рис. 1.

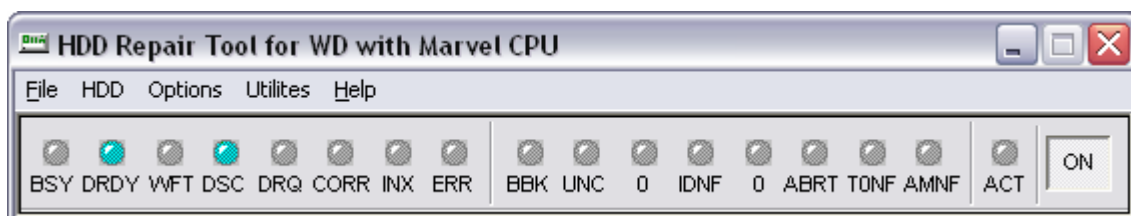


Рис. 1

В нижней части окна расположено две «панели светодиодов». Левая панель отображает содержимое регистра состояния накопителя, правая – состояние регистра ошибок. Под каждым «светодиодом», подписано его мнемоническое обозначение. По показаниям «светодиодов» можно легко ориентироваться о том, в каком состоянии находится накопитель.

Расшифровка обозначений регистра состояния, представлена в Табл. 1

Табл. 1

Обозначение	Расшифровка	Перевод на русский язык
BSY	Busy	Накопитель занят
DRDY	Drive Ready	Накопитель готов к выполнению команды
WFT	Write Fault	Неисправность в накопителе или попытка записи с некорректными параметрами
DSC	Drive Seek Complete	Головки завершили операцию поиска
DRQ	Data Request	Запрос на передачу данных в/из буфера
CORR	Corrected Data	Была ошибка, скорректированная внутренней логикой накопителя
INX	Index	Индексное отверстие проходит под датчиком
ERR	Error	Произошла некорректируемая ошибка

Расшифровка обозначений регистра ошибок, приведена в Табл. 2:

Табл. 2

Обозначение	Расшифровка	Перевод на русский язык
BBK	Bad Mark Block	Блок помечен, как дефектный
UNC	Uncorrected Data	Ошибка ECC в поле данных.
IDNF	ID Not Found	Требуемый цилиндр, сектор, головка не могут быть обнаружены или ошибка ECC в поле заголовка сектора
ABRT	Aborted	Команда отвергнута. Возникает при неверных параметрах команды или неисправностях накопителя
T0NF	Track 0 Not Found	Не найдена нулевая дорожка
AMNF	Data Adress Mark Not Found	Не найден адресный маркер данных сектора, при том, что идентификатор сектора – найден

В главном меню есть подменю «HDD». Также это подменю можно вызвать, если нажать на пустом окне правую кнопку «мыши». Данное подменю содержит список моделей накопителей, поддерживаемых программой.

Одновременно может быть выбрано несколько накопителей. Разумеется, для каждого из них необходимо выбрать свой порт (как это сделать – см. ниже). При этом для каждого накопителя будет открыто своё окно, в котором будет отображаться процесс работы с ним. При работе, программа расщепляется на несколько потоков, поэтому позволяет вести ремонт сразу нескольких накопителей, то есть, пока у одного производится тестирование поверхности, можно производить перекоммутацию другого накопителя и т.п.

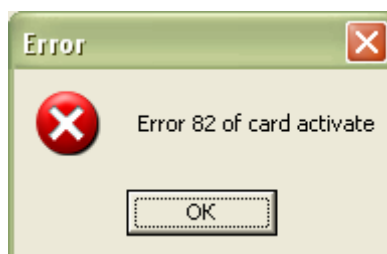
Однако при этом встаёт проблема – состояние регистров какого накопителя отображать на «светодиодах». В данной программе принято следующее решение: на «светодиодах» отображается содержимое регистров того накопителя, соответствующее которому окно было активизировано последним. Таким образом, переключаясь между окнами, Вы можете следить за состоянием интересующего Вас накопителя.

С другой стороны, Вы можете открыть несколько окон накопителя, настроив их на один и тот же порт. В этом случае, в одном окне Вы можете, скажем, открыть диалог для работы с Flash-памятью, в другом – для работы с модулями служебной области и т.п.

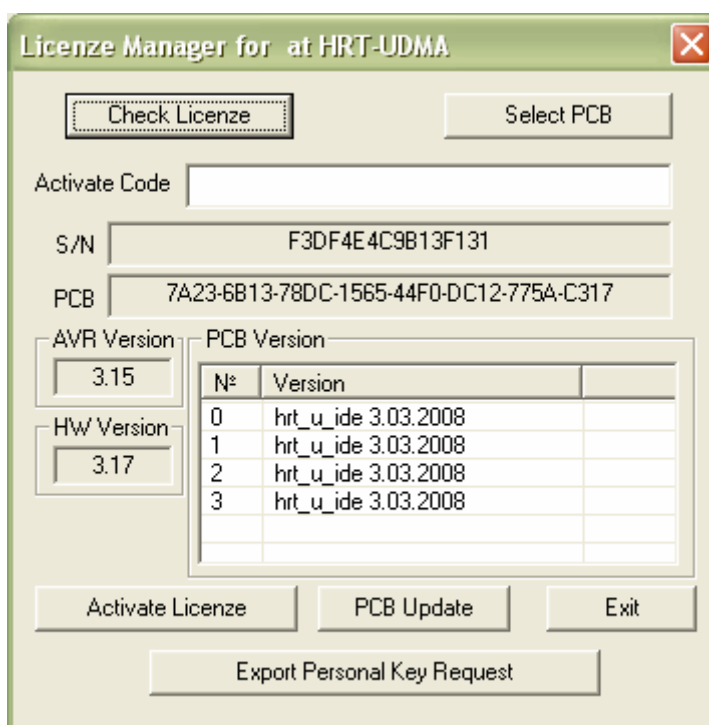
Маленькая хитрость: Если «щёлкнуть» «Мышью» по светодиоду АСТ, в накопитель уйдёт сигнал сброса.

Лицензирование

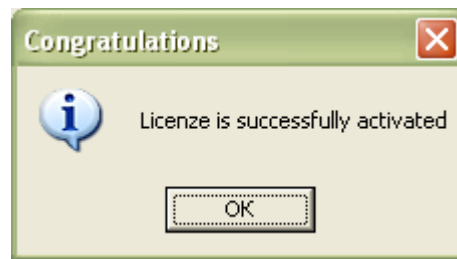
Если ранее лицензия для утилиты не активировалась, то при первом запуске программа выдаст сообщение об ошибке:



после нажатия клавиши ОК, появится диалог активации лицензий:



Для получения кода активации необходимо скопировать поля **S/N**, **PCB**, и вместе с номером утилиты, который был вложен в коробку с CD диском, прислать запрос на активацию на адрес электронной почты: support@bvg-group.ru. На данный запрос Вам будет выслан код активации, который нужно будет ввести в поле **Activate Code** и нажать клавишу **Activate License**. Если код активации был введен правильно, утилита выдаст сообщение:



Достаточно всего однократной активации, после чего карта, на которой была активирована лицензия, всегда будет работать с активированной утилитой. Проверить, активирована ли лицензия для данной утилиты или нет можно при помощи клавиши **Check License**, если лицензия уже активирована, то будет выдано сообщение:



Если с обновлением появилась новая версия схемы карты, то Вы можете обновить старую схему при помощи клавиши **PCB Update**. После нажатия этой клавиши, с помощью появившегося файл селектора, нужно выбрать файл с новой схемой.

Типы трансляции

Для того чтобы обратиться к блоку данных, необходимо знать его положение на диске. Иными словами, надо знать его координаты. На самых первых накопителях, координаты были простыми – цилиндр, головка и сектор. Цилиндр (C)– это номер дорожки, над которой парит блок головок. Головка (H) – это номер поверхности в цилиндре, к которой идёт обращение и, наконец, сектор (S) – это номер сектора на дорожке/поверхности, с которым будет вестись работа.

В те далёкие времена, всё было просто. Диски были довольно маленькими, малой была и плотность записи, что позволяло держать дорожки единой длины по всей поверхности. Правда, даже тогда приходилось включать предкомпенсацию записи (настройка которой перекочевала даже в некоторые современные BIOS).

Шли годы. Плотность дисков стала расти, а разрядная сетка под поля цилиндра, головки и сектора, оставалась неизменной. Уж непонятно, кто всё это придумал, но номер сектора, например, может быть равен от 1 до 64 и никак иначе. Поэтому у способа трансляции CHS наметились большие проблемы.

С другой стороны, контроллеры жёстких дисков стали более интеллектуальными. Если раньше они представляли собой низкоуровневый интерфейс между центральным процессором и диском, то со временем контроллер переехал из слота ЭВМ на сам диск и обрёл значительный уровень интеллекта.

Из этих двух вещей (наличие проблемы числа секторов на дорожку – Sector Per Track или просто SPT и повышение интеллекта контроллера) родился новый тип трансляции – логические CHS. Пользователь не знает о физической структуре диска. Диск наружу отдаёт логические размеры, которые прекрасно влезают в разрядную сетку, выделенную для полей C, H и S, а все координаты, приходящие от центрального процессора, просто преобразует к своему виду, который более удобен для физической реализации накопителя.

Например, процессор думает, что у диска 1000 цилиндров, 16 головок, каждая из которых содержит по 63 сектора (получаем 1008000 адресуемых секторов), а реально у диска всего 4 головки, по 100 секторов на каждой и, соответственно, 2520 цилиндров. Объём диска при этом получается тот же, но все конфликты разрядных сеток улажены.

Способ трансляции логических CHS решил массу других проблем. Во-первых, на всех накопителях имеются дефектные блоки. Если координаты всё равно преобразуются от логических к физическим, то нет нужды обращаться к сбойным секторам. Достаточно пометить сбойные сектора в специальных таблицах (дефект-листах) и, если преобразование логических координат в физические указывает на такой сбойный сектор, то подменить его номер другим сектором, не содержащим сбоя (способы подмены определяются конкретной моделью накопителя).

Кроме того, у современных накопителей число секторов на физическую дорожку непостоянно. Если бы она была постоянна, то чем ближе к центру, тем выше был бы физическая плотность записи. Это потребовало бы создания сложной электроники, работающей в широком частотном диапазоне. Разработчики пошли по другому пути – разбили диск на зоны. Чем ближе зона к центру диска, тем меньше у неё секторов на дорожку. Таким образом, получается примерно одинаковая линейная плотность записи по всей поверхности, что упрощает работу электроники.

Логические же размеры дорожки всегда одинаковы. Накопитель просто производит преобразование координат по внутренней таблице зон и ни одна программа, выполняющаяся на центральном процессоре ЭВМ, не обязана знать о внутреннем устройстве диска (таблицы дефектов, таблицы зонного распределения и т.п.).

Подводя итог всем этим рассуждениям, можно сказать, что существует две трансляции – логическая и физическая. В логической трансляции накопитель представляет собой устройство с равномерным распределением координат (логические CHS или LBA), в физической – устройство с весьма сложной системой таблиц и взаимосвязей.

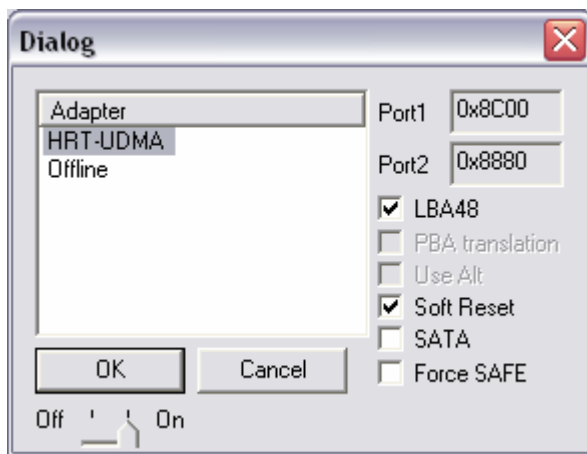
Конечный пользователь работает в логической трансляции, поэтому выходной контроль накопителя необходимо осуществлять именно в ней. Однако при ремонте, знание логических координат дефектов нам ничего не скажет. Лучше рассматривать все дефекты в физической трансляции, так как именно в ней сразу будут видны неисправные цилиндры, неисправные головки и радиальные царапины (ни того, ни другого, ни третьего в логической трансляции не увидит, так как там проходит масса преобразований, скрывающих физическую картину диска)

Настройка порта

Если Вы выберете накопитель, как показано ниже



на экран будет выдан диалог настройки порта:

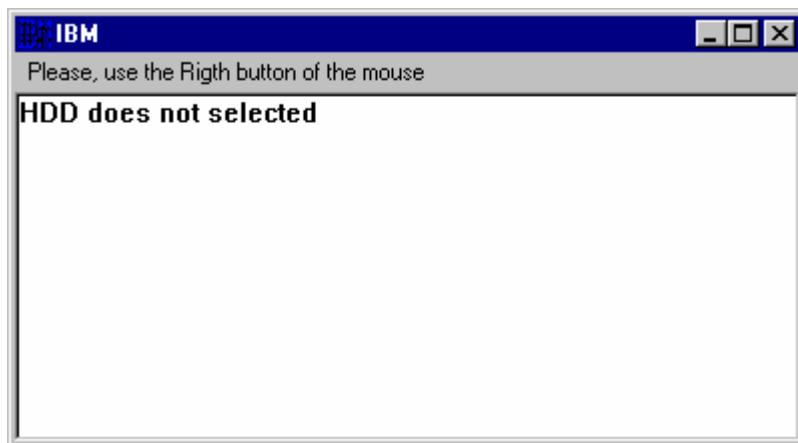


Переключатель On/Off позволяет Вам оперативно выключать и включать питание накопителя.

Список с конфигурациями позволяет выбрать активный адаптер.

Но не всегда это бывает нужно. Накопитель может находиться в таком режиме, когда нельзя прерывать его нормальную работу. Типичный пример – Вы запустили offline-тесты SMART и нечаянно закрыли главное окно программы. Самое лучшее решение – открыть новое окно, зайти в диалог ATA-команд и проследить за поведением накопителя через режим **Monitor Mode** окна ручной подачи ATA-команд. Но встанет проблема – при открытии окна, на накопитель уйдёт сигнал сброса, тест прервётся, и наблюдение за состоянием его регистров уже не будет иметь никакого смысла.

Как раз для этого, и введён режим **Offline**. Если Вы выберете его – накопитель не будет проинициализирован. Главное окно при этом будет иметь вид:



а показания светодиодов могут быть самыми разными. Сразу же, в главном меню снова выбирайте пункт *Setup Port* и в нём устанавливайте нужный порт. После этого, показания светодиодов будут отражать реальное состояние системы.

Корректная работа других пунктов в режиме **Offline** не гарантируется, но и не заблокирована.

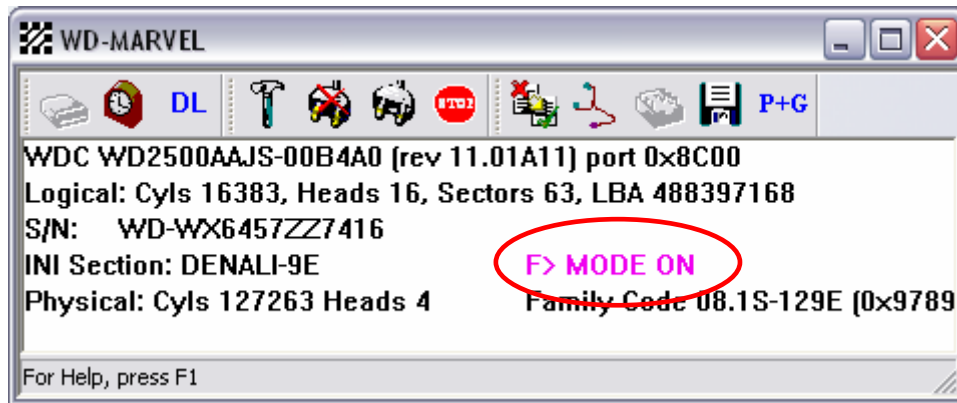
Флажок **LBA48** оставлен для накопителей, не понимающих команды LBA48 (ёмкость которых менее 128 гигабайт), которые из-за проблем в служебной области сообщают, что они понимают такие команды. В этом случае, необходимо снять флажок вручную, сообщив утилите, что она не может даже пытаться подавать «новые» команды.

Флажок **Use Alt** нужен только для накопителей **Seagate**. Дело в том, что эти накопитель поддерживают не только АТА команды, но и команды, передаваемые через СОМ-порт (альтернативная подача команд). Если у Вас не подключён СОМ-адаптер, а программа пытается пользоваться СОМ-портом, система «зависнет». И в этом случае, флажок **Use ALT** следует снять. В остальных случаях, оставляйте его значение «по умолчанию».

Флажок **Soft Reset** определяет посылку сигнала сброса в накопитель при запуске утилиты. Самые первые SATA накопители при этом «зависали» и флажок приходилось снимать. В остальных случаях (IDE накопители и современные SATA накопители) этот флажок снимать не требуется.

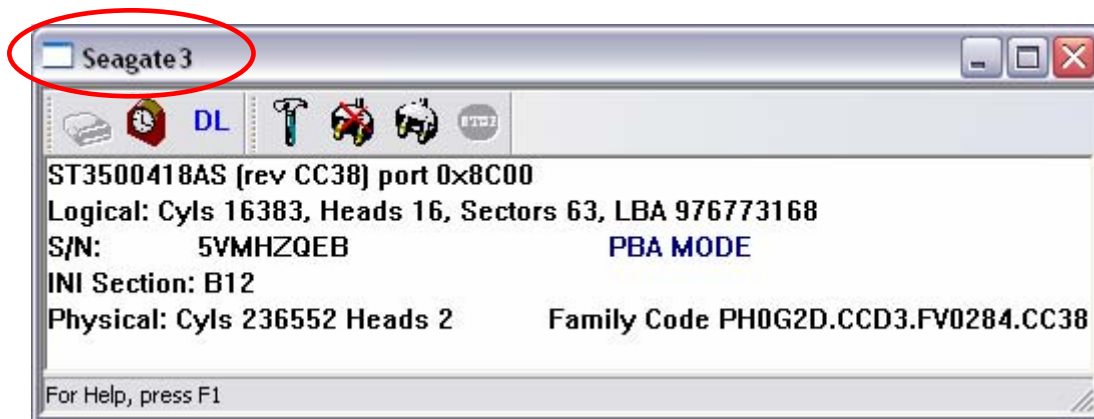
Флажок SATA также был необходим для самых первых SATA накопителей. Если они «зависали» даже со снятым флажком **Soft Reset**, надо было выставить этот флажок. К счастью, эта проблема была быстро устранена производителями накопителей.

Флажок **Force SAFE** необходим, если Вы подключаетесь к накопителю, находящемуся в **SAFE** режиме. В некоторых случаях (накопители Seagate и Western Digital с интерфейсом IDE), утилита может сама определить, что накопитель находится в **SAFE Mode**. Однако, для накопителей Western Digital с интерфейсом SATA это сделать невозможно. А в режиме **Safe** отличаются команды, которые следует подавать в накопитель. Поэтому в этом случае устанавливайте данный флажок. Когда накопитель находится в режиме **Safe**, в окне накопителя появляется соответствующее предупреждение (Вы можете изменить его положение и текст через правку файла `hdd.ini`):



Настройка COM порта

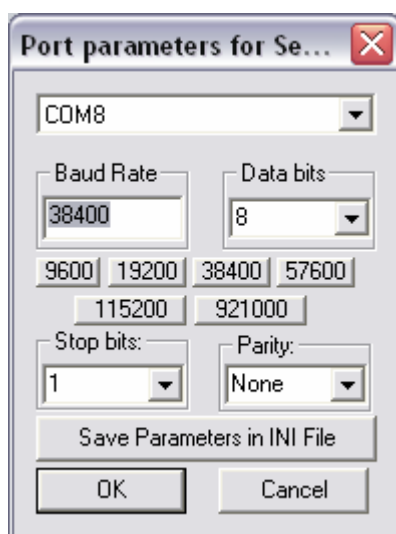
Все сведения о настройках COM порта сохраняются в INI файле накопителя. Имя INI файла легко узнать по главному окну накопителя. Например, если окно такое:



То INI файл называется Seagate3.ini. В целом, настройки хранятся в секции [COM PORT] (пример приведён ниже), но реально Вам изменять их в INI файле не требуется. Для этого есть специальный диалог, выдаваемый при запуске утилиты.

```
[COM PORT]
Port number=3
Baud rate=38400
Data bits=2
Stop bits=0
Parity=0
```

Рассмотрим этот диалог:



В верхнем выпадающем списке показаны все допустимые COM порты на данном компьютере. В поле Baud Rate Вы можете вписать скорость работы COM порта. Для облегчения этой работы, типичные скорости вынесены на кнопки. Нажимая их, Вы автоматически измените содержимое данного поля. Поле Data Bits содержит число

битов данных (обычно 8 бит). Поле Stop Bits задаёт число стоповых битов, поле Parity определяет тип контроля чётности.

Выбрав необходимые параметры, нажмите ОК, чтобы начать работу. Эти параметры будут утеряны при закрытии окна накопителя. Если же Вам необходимо запомнить настройки на будущее, нажмите Save Parameters in INI File, а уже затем ОК.

Маленькая хитрость: Если Вам надоело при каждом запуске утилиты видеть параметры настройки COM порта и нажимать ОК, Вы можете вручную подправить INI файл. Впишите в секцию [COM PORT] строку:

Setup At Start=0

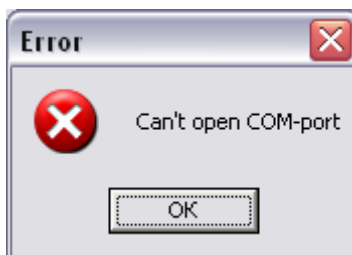
Ещё хитрость: Если Вы хотите сохранять протокол данных, проходящих через COM порт, впишите в секцию [COM PORT] строку

Log File Name=<имя файла лога>

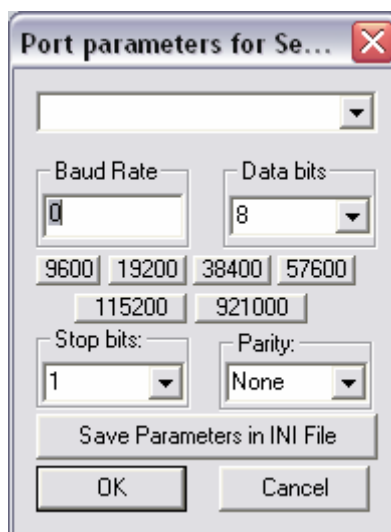
Например

Log File Name=

Полезный совет: Если номер COM порта изменился, или COM порт занят другой программой, при запуске утилиты будет выдано сообщение:

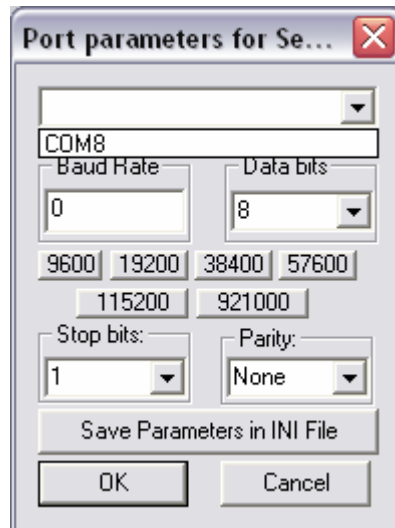


После чего появится диалог настройки параметров COM порта (даже если он был отключён через INI файл)



Если у Вас нет адаптера, либо нет желания освободить порт другой программой, просто нажмите Cancel и будьте готовы, что программа станет работать без использования COM порта (это критично для накопителей Seagate Barracuda 2-10).

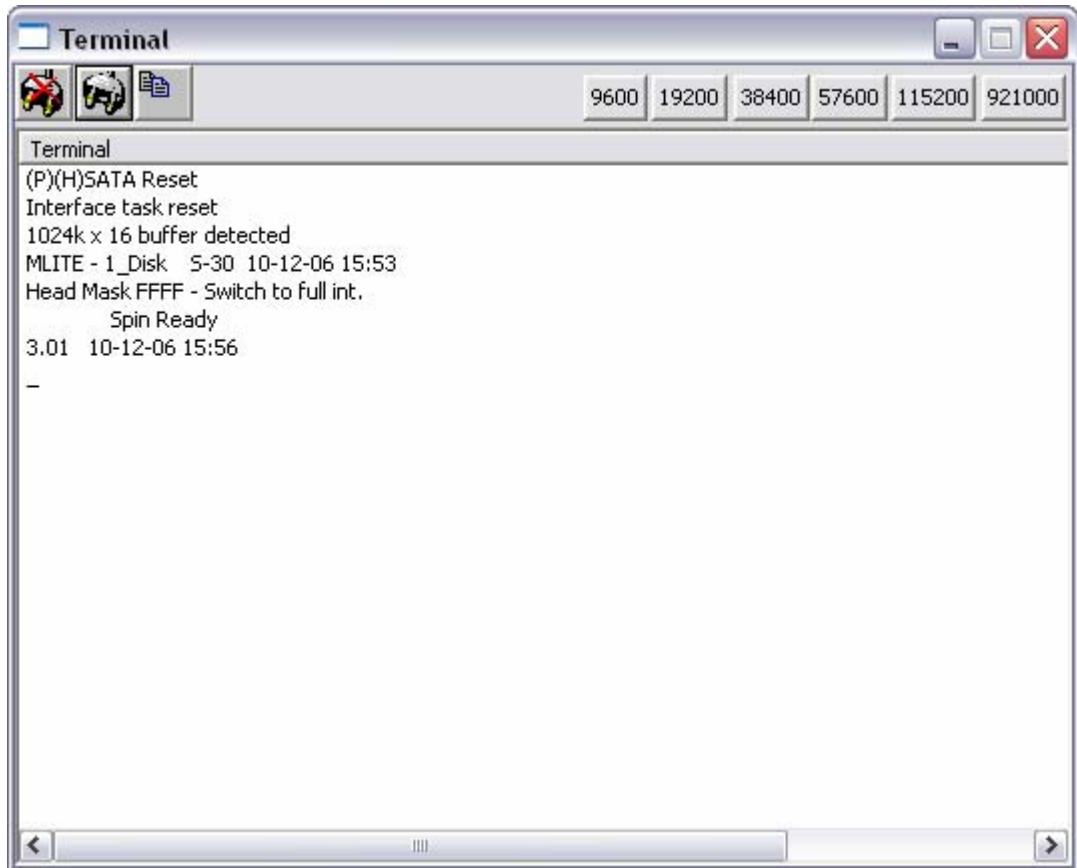
Если же необходимо просто изменить номер порта, то выберите его из списка



Далее, задайте скорость работы (обычно нажав ту или иную кнопку), при необходимости – сохраните параметры, после чего нажмите ОК.

Терминал

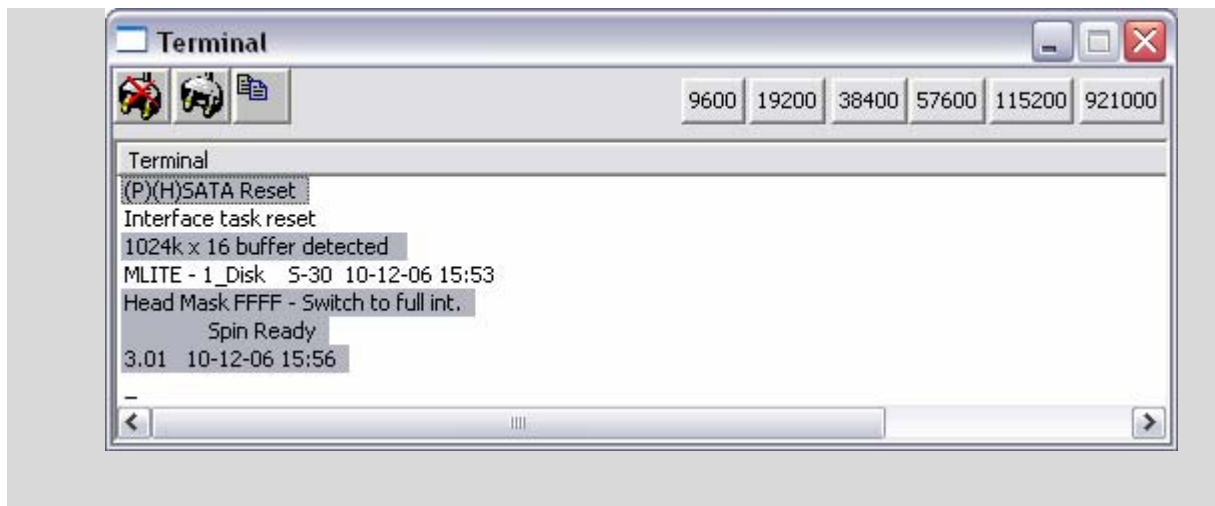
Окно терминала имеет следующий вид:




Терминал построен по строчному принципу. Поэтому если Вам необходимо взять в буфер обмена блок, то



- 1) Выделите первую его строку
- 2) Нажмите клавишу Shift
- 3) Выделите последнюю строку

Маленькая хитрость: Можно выделять не только блоки, но и фрагменты. Строка, выделенная при нажатой клавише Ctrl, не снимет выделение с других строк



Когда блок выделен, нажмите кнопку  в панели инструментов. Выбранный текст окажется в буфере обмена и будет доступен для вставки в сторонние редакторы. Приведённое выше выделение даст следующий текст (обратите внимание, что не выделенные строки отсутствуют):

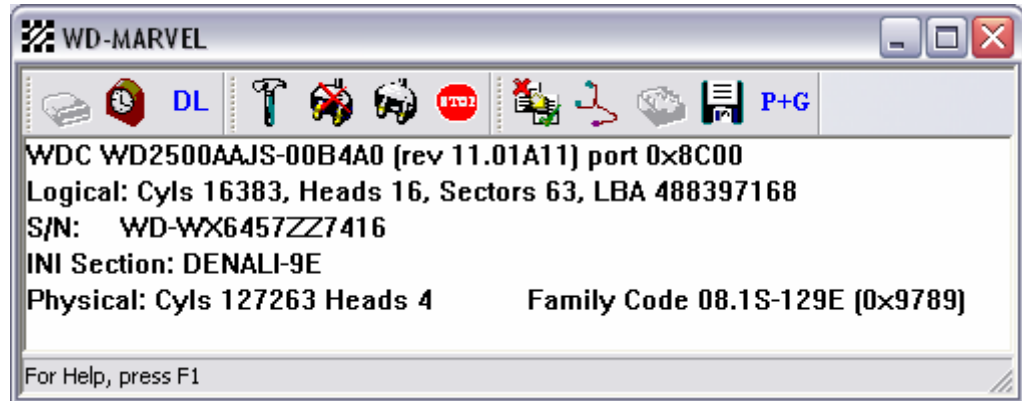
```
(P)(H)SATA Reset
1024k x 16 buffer detected
Head Mask FFFF - Switch to full int.
    Spin Ready
3.01 10-12-06 15:56
```

Кнопки  и  позволяют выключать и включать питание накопителя, не прибегая к помощи основного окна программы. Это бывает полезно для ручного перевода накопителей Barracuda 7-10 в режим F>.

Кнопки с числами позволяют быстро переключить скорость работы COM порта. Имейте в виду, что речь идёт именно о скорости работы COM порта. Скорость порта накопителя при этом не изменяется!

Окно накопителя

Окно накопителя имеет вид, представленный ниже:



Вы можете изменить интерфейс (тексты, их положение и цвета) через настройки в файле HDD.INI, однако, в стандартной поставке, всё настроено именно на такое отображение.

Первая строка содержит имя и номер версии микрокода накопителя, а также базовый порт, через который программа общается с накопителем. Если Вы ремонтируете одновременно несколько накопителей, то по адресу порта Вы сможете ориентироваться в окнах.

Вторая строка содержит логические параметры накопителя.

В третьей строке расположен серийный номер накопителя, а также индикаторы текущего режима работы.

Четвёртая строка отображает имя секции инициализационного файла, с которым работает программа. Секция выбирается на основе кода семейства накопителя.

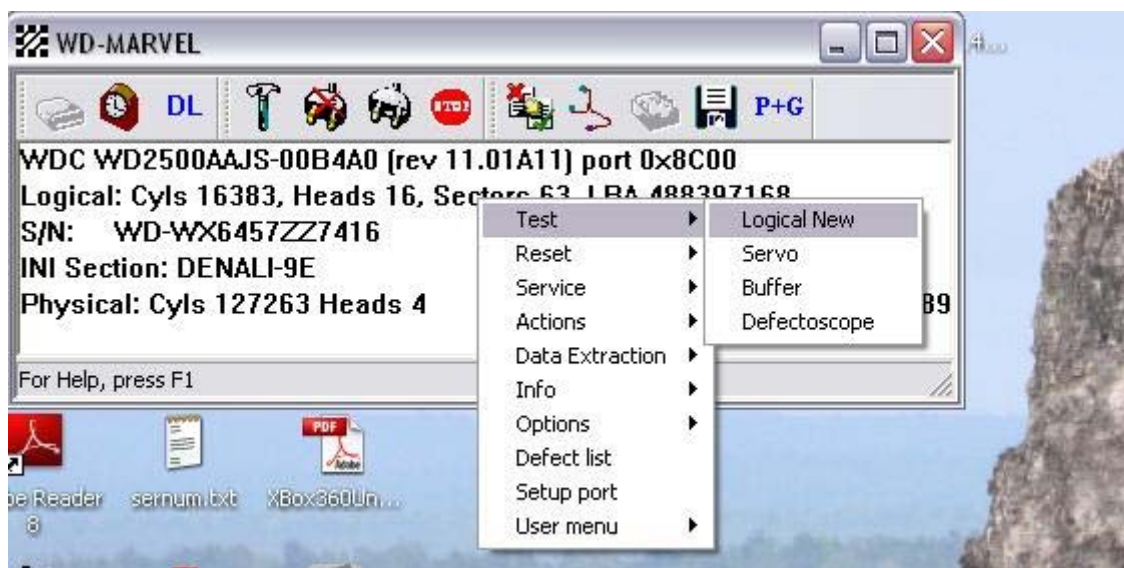
Пятая строка содержит физические параметры накопителя – число физических цилиндров и физических головок. Параметр **Family Code** для каждого накопителя строится по характерным только для него правилам. Подробности о формировании Family Code Вы можете посмотреть в документации на конкретную утилиту.

При нажатии правой кнопки «Мыши», программа выдаёт на экран главное меню, иерархия пунктов которого приведена в документации на конкретную утилиту. Ниже будут рассмотрены пункты, которые встречаются в большинстве утилит (но не обязательно в каждой из них).

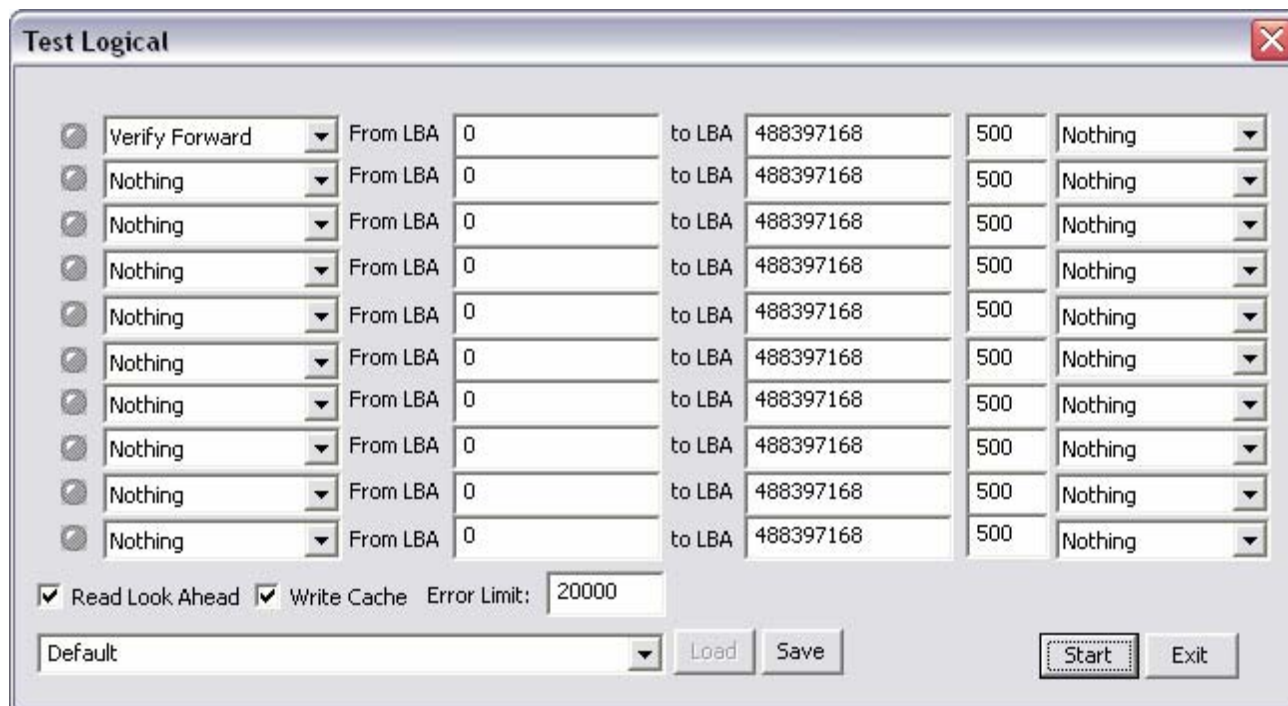
Тестирование накопителя

В прошлых версиях документации традиционно описывалось тестирование в физической и логической трансляции. Однако, в настоящий момент, можно сказать, что тестирование в физической трансляции утратило всяческий смысл. Первоначально оно было полезно, так как некоторые накопители «зависали» на определённых дефектах в логической трансляции. Однако, те времена канули в лету. Достаточно логической трансляции, а если есть желание посмотреть физическое расположение найденных дефектов – можно оттранслировать логические координаты в физические.

Поэтому из утилиты было изъято тестирование по физике и оставлено только тестирование по логике. Оно вызывается пунктом меню Test->Logical New



Окно настройки теста имеет следующий вид:



Каждая строка определяет шаг тестирования. Для каждого шага тестирования следует задать:

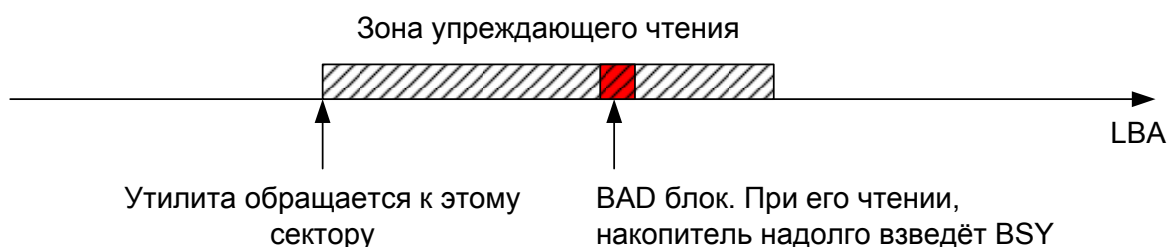
- 1) Тип теста (выбирается из выпадающего списка)

Тип теста	Описание
Nothing	Пропустить шаг
Write Forward	Производить запись, двигая головку от младших LBA к старшим. При этом в тело каждого сектора записывается его LBA для того, чтобы затем можно было проверить транслятор.
Write Reverse	Производить запись, двигая головку от старших LBA к младшим. При этом в тело каждого сектора записывается его LBA для того, чтобы затем можно было проверить транслятор.
Verify Forward	Верификация поверхности без передачи данных от накопителя к утилите. Работает на самой высокой из возможных скоростей, так как передача данных занимает максимум времени. Является основным типом тестирования.
Verify Reverse	Верификация поверхности с ведением головок от конца к началу.
Read Forward	Тест чтения. То же, что верификация, но ещё передаются данные. Никаких преимуществ перед верификацией не имеет.
Read Reverse	Тест чтения от старших LBA к младшим
Read And Check	Тест чтения с проверкой содержимого. Если содержимое не соответствует тому, что было записано при тесте записи, будет выдано сообщение об ошибке транслятора. Крайне полезный тест после каких-либо операций по переконфигурации накопителя, так как после них часто сбивается транслятор.
Seek And Read	Позиционирование с последующим чтением. Данный тест даёт повышенную нагрузку на позиционер, так как ради каждого сектора головки перемещаются на $\frac{1}{2}$ диска. Идеален для целей проверки устойчивости накопителя к нагреванию, а также, на уход параметров позиционера.

- 2) Начальный LBA
- 3) Конечный LBA
- 4) Время таймаута в миллисекундах
- 5) Метод скрытия дефектов по окончании теста

Метод	Описание
Nothing	Ничего не делать, оставить список дефектов на скрытие следующими шагами
Clear	Очистить список найденных дефектов
Recovery	Скрыть, как логические дефекты
Log Phys	Сконвертировать дефекты в физические, затем скрыть их в P-LIST

Флажок **Read Look Ahead** требует у утилиты включать упреждающее чтение. В результате, накопитель проводит тестирование намного быстрее, но с другой стороны, ошибки с таймаутами могут быть выявлены не на том секторе, на котором они реально встретились. Ситуация проиллюстрирована ниже.

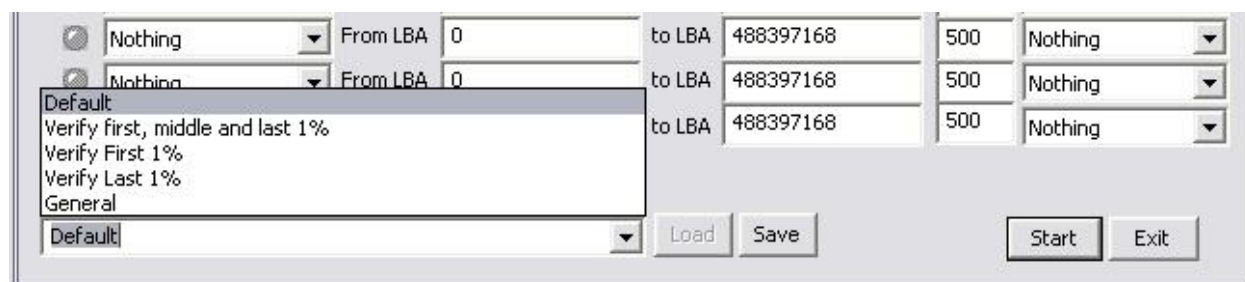


Как видно, BSY будет взведён надолго, так как накопитель встретил BAD блок. Однако, утилита обращается к сектору с меньшим LBA, поэтому таймаут будет зафиксирован именно для него. Если упреждающее чтение отключено, такая ситуация не возникнет.

Флажок **Write Cache** включает кэширование записи, в результате чего тест записи будет выполняться быстрее.

Поле **Err Limit** задаёт число ошибок, при достижении которого тест будет прерван досрочно, так как такой накопитель тестировать бесполезно.

Внизу окна располагается группа для выбора конфигураций тестирования. По умолчанию доступны следующие конфигурации:



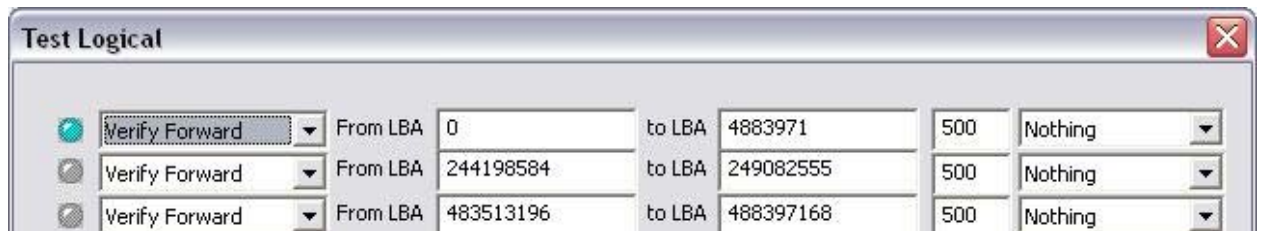
Необходимо выбрать одну из конфигураций и нажать Load.

Новые конфигурации можно задавать следующим образом:

- 1) Настроить все шаги
- 2) Ввести имя конфигурации
- 3) Нажать кнопку Save

Все конфигурации хранятся в файле **Test Groups.ini**. При желании, можно редактировать этот файл напрямую.

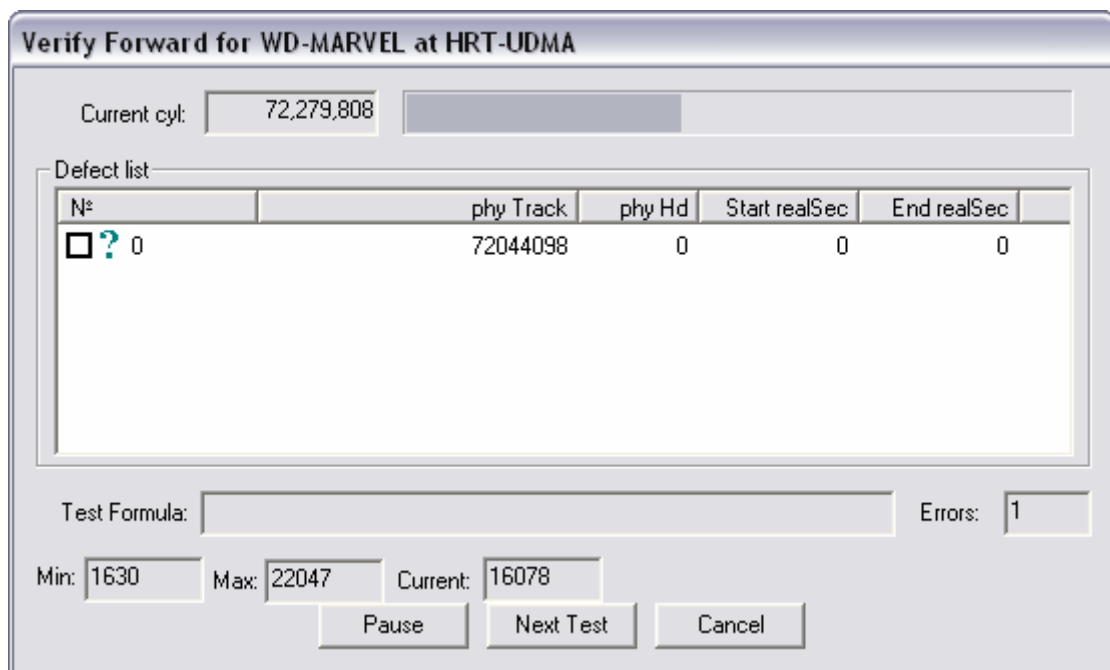
По мере прохождения шагов теста, около каждого шага будет загораться светодиод. Голубой – идёт тест



Серый – тест успешно завершён

Красный – тест выявил ошибки

Кроме того, во время тестирования появляется дополнительное окно статуса процесса.



Заголовок окна определяет текущий тест. Поле **Current Cyl** задаёт номер цилиндра, который тестируется в данный момент. Справа от этого поля, расположен индикатор, показывающий отношение выполненного теста к тому, что ещё предстоит выполнить. В середине показан дефект-лист с обнаруженными дефектами.

Символ около дефекта показывает его состояние:

? - дефект обнаружен. Попыток восстановления пока не производилось.

V - Дефект помещён в таблицу (либо скрыт иным методом)

X - попытка помещения дефекта в таблицу (либо скрытия иным методом) завершена неудачно.

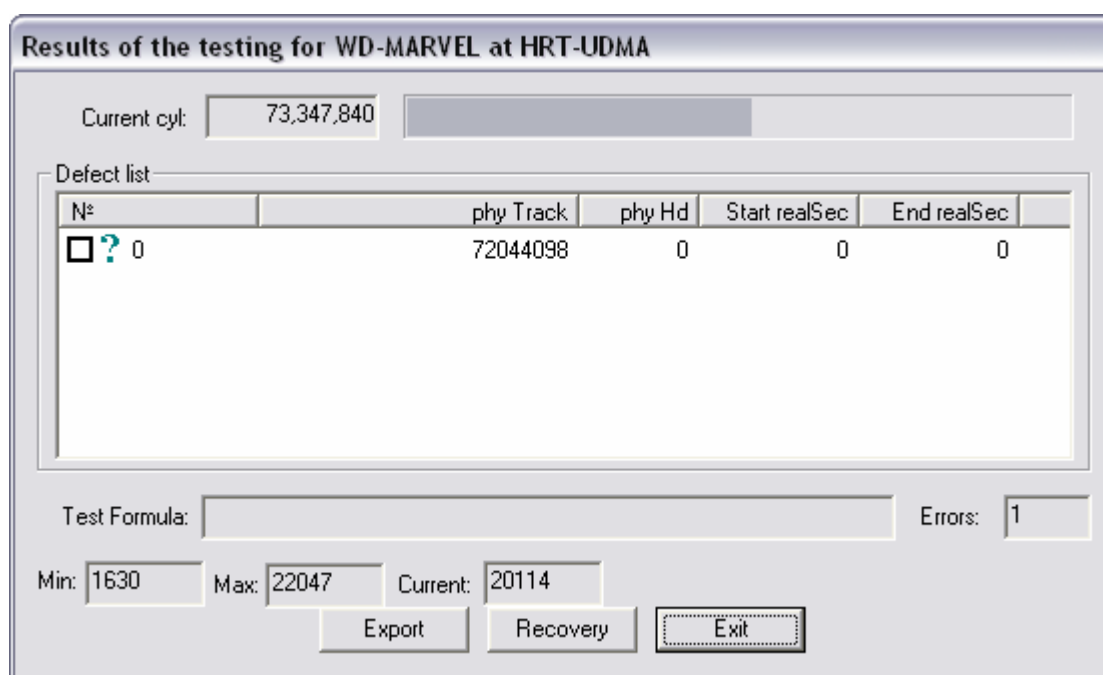
Поле **Errors count** показывает число дефектов, обнаруженных при данном проходе теста.

Кнопка **Next Test** позволяет прервать текущий тест и перейти к следующему. Прерывание теста произойдёт не моментально, а по окончании тестирования текущего сектора. Если накопитель надолго ушёл в состояние BSY, то это может продлиться несколько секунд – до истечения критического времени, заданного при настройке теста.

Кнопка **Cancel** прервёт весь процесс тестирования.

Кнопка **Pause** позволяет приостановить тестирование накопителя. В это время, в другом окне, настроенном на тот же адаптер, можно выполнить те или иные действия (посмотреть атрибуты SMART, проверить не затёрлась ли служебная область и т.п.).

По окончании теста, окно примет вид:



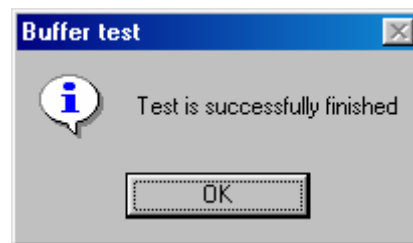
В данном случае, тест был прерван на LBA=73347840, и не производилось попыток скрытия дефектов (так как при настройке был выбран метод скрытия Nothing). Если нажать на кнопку **Exit**, то произойдёт выход из диалога тестирования.

Однако, Вы можете прямо сейчас скрыть обнаруженные дефекты. Для этого нажмите кнопку **Recovery**. Если же нажать кнопку **Export**, то перечень обнаруженных дефектов будет сохранён в файл. Этот файл в дальнейшем можно будет использовать в других программах (как ремонтных, так и аналитических), а также его можно будет импортировать в диалог расширенной работы с дефект-листом, который описан ниже.

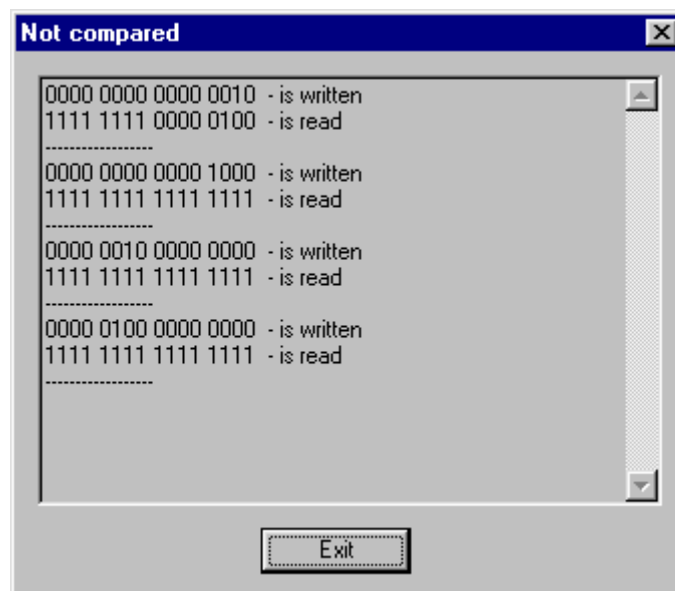
Тест буфера

Тест буфера получил своё название от команд, которыми он пользуется (Write Buffer и Read Buffer). Но на самом деле, проверить буферное ОЗУ может только сам накопитель, так как у каждого семейства организация памяти своя. Зато данный тест прекрасно проверяет качество пайки интерфейсных линий платы накопителя, а также исправность внутренней шины, по которой данные из интерфейса попадают в ОЗУ.

Для запуска теста буфера, необходимо выбрать пункт меню Test->Buffer. Если тест не обнаружил ошибок, то будет выдано сообщение



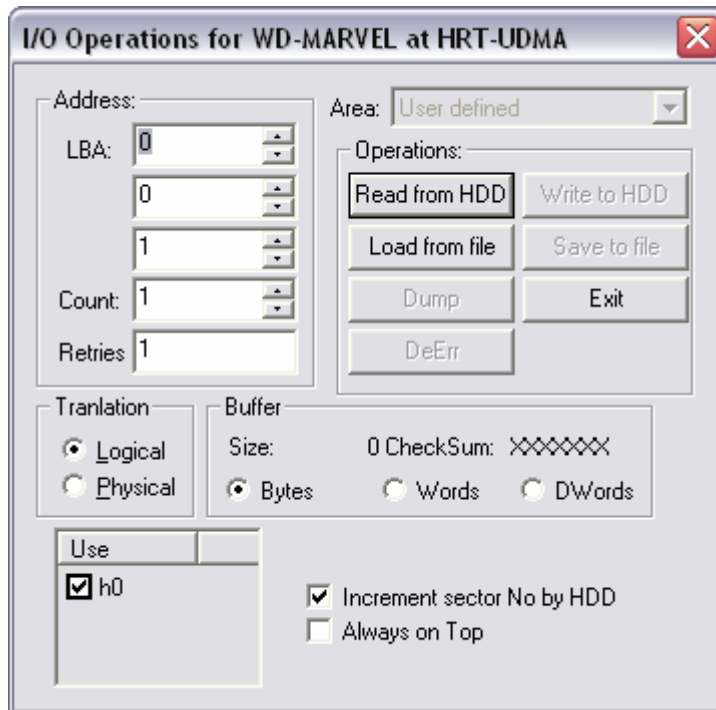
В противном случае, будет выдан перечень, состоящий из кодов, посланных в накопитель и полученных обратно.



Операции ввода-вывода

Чтение и запись по логическим координатам

Для выполнения операций ввода-вывода, необходимо выбрать пункт *Service->I/O Operations*. При этом в режиме LBA на экран будет выдан следующий диалог:



Поле **LBA** содержит номер начального блока, который будет использоваться при операции.

Два следующих поля при работе в логической трансляции не используются.

Поле **Count** содержит число секторов, подлежащих передаче.

Все поля позволяют вводить как десятичные, так и шестнадцатеричные значения. Перед шестнадцатеричными значениями должен идти префикс «0x». Так, например, значение 16 равно значению 0x10 (**однако, запись 0X10 будет считаться ошибочной!!!**).

Группа элементов **Buffer** отображает состояние буфера. Если в него считаны данные, то будут отображаться длина массива и его контрольная сумма, рассчитанная одним из трёх способов: сложением байтов (если выбрана радио кнопка **Bytes**), сложением слов (если выбрана радио кнопка **Words**) или сложением двойных слов (радио кнопка **Dwords**).

Флажок **Increment Sector Number By HDD** определяет тип вычисления координат очередного сектора. Если он установлен, то параметр Count просто

передаётся в накопитель и именно накопитель определит число считываемых секторов. **Разумеется, если Count будет больше, чем 256 (именно этим числом секторов ограничен объём пересылки за один проход), то накопитель отработает запрос некорректно.**

Однако, если флажок **Increment Sector Number By HDD** сброшен, программа будет запрашивать сектора у накопителя по одному, автоматически увеличивая координаты. Этот механизм работает НАМНОГО медленней, но зато с его помощью можно считывать массивы большего объёма.

Радио кнопки **Logical** и **Physical** определяют тип работы (по логическим или по физическим координатам). Работа по физическим координатам будет описана ниже.

Флажок **Always On Top** позволяет разместить диалог поверх других окон. Это иногда бывает полезно, если Вы производите какие-либо параллельные действия с этим же накопителем в другом окне. При этом дочерние окна другого потока не будут закрывать этот диалог.

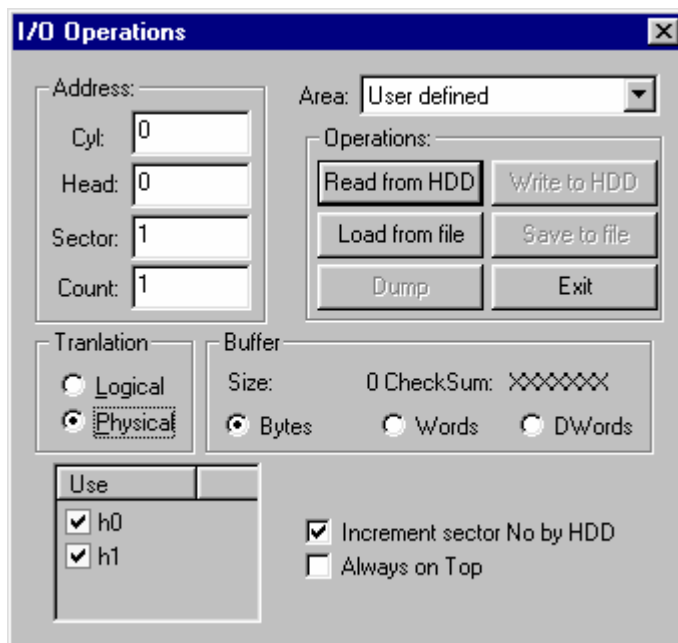
Флажок **Maximize Dump Window** используется для указания режима работы окна дампа. Если флажок сброшен, то размер окна дампа будет такой же, как и у окна I/O Operations. В целом, данный флажок введён скорее для совместимости с предыдущими версиями программы, в которых окно дампа было нельзя максимизировать, и предназначен для тех, кто привык к данному режиму. Для новых пользователей можно рекомендовать не сбрасывать данный флажок.

Кнопки **Read From HDD** и **Write To HDD**, соответственно, считывают и записывают данные с/на накопитель.

Кнопка **Load From File**, считывает данные из файла.

Кнопка **Save To File** записывает данные в файл.

Кнопка **Dump** позволяет просмотреть и отредактировать данные, расположенные в буфере.



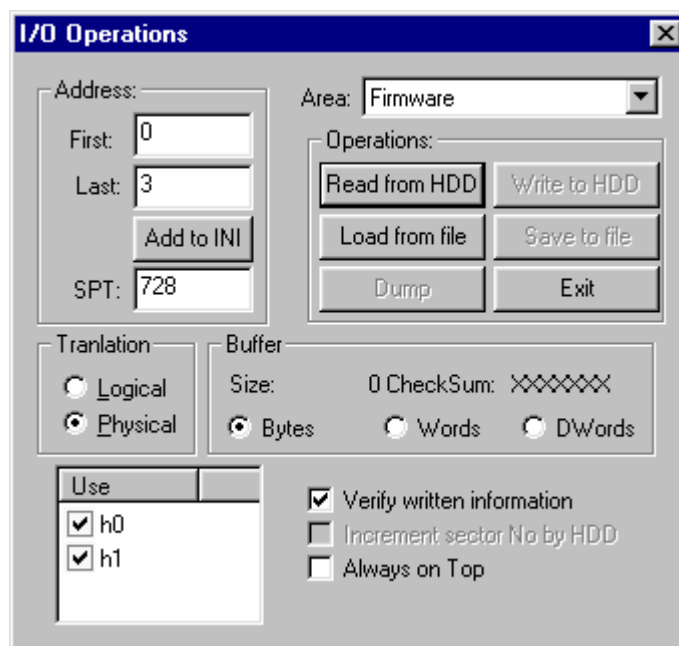
Кнопка **Exit** закрывает диалог I/O Operations. Данные из буфера при этом будут утеряны.

Чтение и запись по физическим координатам

Для перехода к работе по физическим координатам, необходимо выбрать радио кнопку **Physical**. Накопитель при этом перейдёт в режим физических CHS. Окно примет вид:

Как видно из рисунка, теперь в качестве координат можно задавать номер цилиндра, головки и начального сектора. Назначение всех элементов управления, аналогично работе в логической трансляции.

Также при работе в физической трансляции, открылось поле **Area**. по умолчанию, там выбран режим User Defined. Однако, для накопителей Samsung, можно выбрать и значение Firmware. При этом окно примет вид:



Поле First задаёт номер начального цилиндра служебной области.

Поле Last задаёт номер конечного цилиндра служебной области.

Поле SPT задаёт число секторов на дорожку, которое следует считывать.

Кнопка Add To INI позволяет добавить параметры, введённые вручную, в файл Samsung.ini. После этого, параметры для семейства будут подставляться автоматически.

Таблица Use позволяет исключать из работы в данном сеансе те или иные головки.

Флажок Cerify written information включает режим контрольного считывания записанной информации.

Кнопки Read, Write, Load и Save действуют аналогично прочим режимам.

Следует отметить, что в данном режиме, на нулевую и первую головки записывается одна и та же информация. Соответственно, считывается – тоже. То есть, считается, что первая головка является точной копией второй головки. Это позволяет вдвое уменьшить размер файла ресурсов. Однако, для большинства современных накопителей это не всегда верно, поэтому авторы настоятельно рекомендуют работать со служебной областью ТОЛЬКО через диалог Full Firmware. Данная же устаревшая функциональность не заблокирована только на тот случай, если Вы придумаете ей какое-либо применение.

Работа с памятью

Общие сведения

Работа с памятью может быть полезна для исследовательских целей. Условно, память накопителей IBM можно разбить на три типа – оперативная память процессора, буферная память кэша диска и энергонезависимая память. Накопители WD и Fujitsu поддерживают только оперативную и буферную память. У накопителей SAMSUNG – вообще оперативная память разбита на память программ (недоступную по интерфейсу ATA) и память данных. Физически, оперативная и буферная память расположены в одном и том же адресном пространстве, но разработчики накопителей предпочли вынести команды работы с кэш-памятью в отдельный класс. Это очень удобно, так как если Вы хотите узнать, что расположено в кэш-памяти диска, Вам не надо знать особенностей архитектуры конкретной модели, а просто вызвать соответствующую команду и получить доступ к нужной области.

С другой стороны, эти же данные можно узнать, обратившись к оперативной памяти, задав адрес начала кэш-буфера.

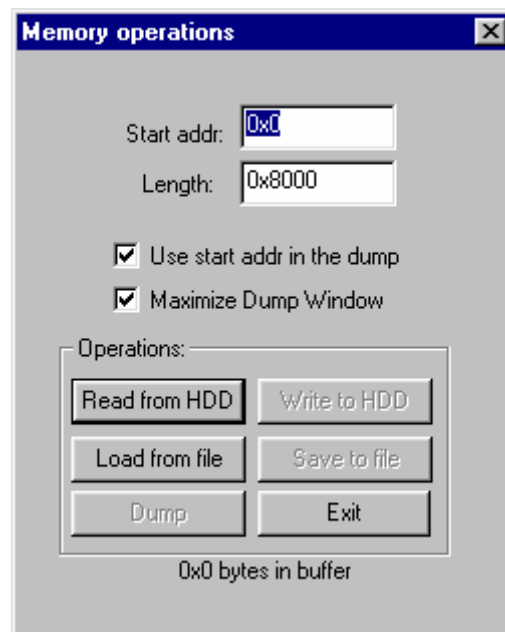
Энергонезависимая же память, хранит в себе базовые настройки накопителя, которые должны быть доступны сразу после включения питания, но которые не стоит зашивать в ПЗУ, так как они могут варьироваться в зависимости от конкретной электроники и механики.

Работа с ОЗУ/ПЗУ

Для работы с ОЗУ/ПЗУ, выберите пункт меню

Service->Memory->RAM/ROM Operations

При этом на экран будет выдан следующий диалог:



Поля **Start Addr** и **Length** задают начальный адрес рабочей области и её длину.

Флажок **Use start addr in the dump** определяет способ адресации в окне дампа. Если он установлен, то адресация начнётся с того адреса, который указан в строке **Start Addr** диалога. Если же флажок сброшен – адресация начнётся с нулевого адреса (иногда полезен один вариант, иногда – другой).

Флажок **Maximize Dump Window**, как и в диалоге *I/O Operations*, введён для совместимости. Кто привык к тому, что дамп не распаивается на весь экран, могут его сбрасывать. В большинстве случаев, это не нужно.

Кнопка **Read From HDD** считывает память с накопителя во внутренний буфер, а кнопка **Write To HDD** – наоборот записывает данные из внутреннего буфера в память. Разумеется, если буфер пуст, то кнопка заблокирована, ведь передавать в накопитель нечего.

Следует особо отметить тот факт, что бездумная запись в память может иметь весьма плачевные последствия. Например, накопитель может остановить двигатель, не запарковав головки, и они прилипнут к поверхности. Если Вы точно не уверены, что на ту область, куда Вы собираетесь писать, не спроецированы порты, то лучше не пишите туда. Авторы не несут никакой ответственности за последствия подобных экспериментов! Главное правило – если пишете в память, то делайте это со знанием дела!!!

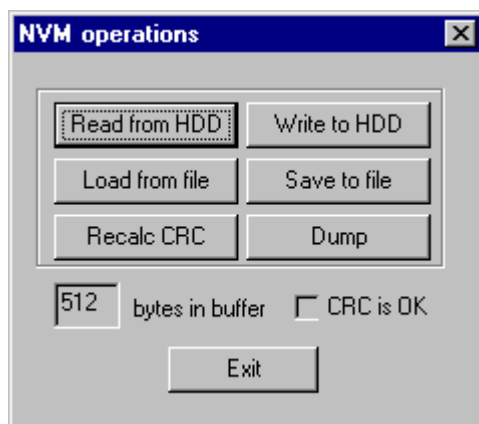
Кнопка **Load From File** считывает содержимое файла во внутренний буфер, а кнопка **Save To File** – записывает содержимое внутреннего буфера в файл. Таким образом, Вы можете считать какой-либо участок памяти в буфер и записать его в файл для исследования при помощи внешних программ (дизассемблера, различных аналитических утилит). А можете – считать данные из файла и переслать их в память накопителя.

Кнопка **Dump** открывает окно дампа, в котором Вы можете просматривать и редактировать данные, находящиеся во внутреннем буфере.

Работа с энергонезависимой памятью

Для работы с энергонезависимой памятью, необходимо выбрать пункт меню **Service->Memory->NVM Operations**.

При этом на экран будет выдан следующий диалог:



Кнопка **Read From HDD** считывает данные из энергонезависимой памяти (Non Voltage Memory – NVM) во внутренний буфер. Кнопка **Write To HDD** – записывает содержимое внутреннего буфера в энергонезависимую память.

Кнопка **Save To File**, записывает содержимое внутреннего буфера в файл. При этом Вам предлагается на выбор два расширения имени – NVM или BIN. Формат сохраняемого файла от расширения не зависит, просто расширение «NVM» подчёркивает, что содержимое файла является образом NVM накопителя, а расширение «BIN» подчёркивает, что файл имеет двоичное содержимое. Вы можете использовать любое из них, однако, рекомендуемое расширение, всё-таки, NVM.

Кнопка **Load From File** считывает файл во внутренний буфер.

Кнопка **Dump** открывает окно дампа, в котором Вы можете просмотреть и отредактировать данные, находящиеся во внутреннем буфере.

Кнопка **Recalc CRC** позволяет пересчитать контрольную сумму буфера данных, так как накопитель проверяет её при старте. Индикатор же **CRC is OK** отображает текущее состояние дел (если он установлен, значит контрольная сумма блока в порядке).

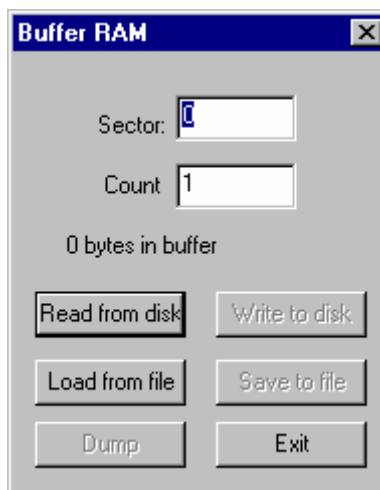
Работа с буферной памятью накопителя

Работа с буферной памятью накопителя, является чисто вспомогательной. Точно не известно, зачем она может пригодиться. Возможно, её можно использовать для выполнения операций по восстановлению данных с разрушенных секторов. Также она может оказаться полезной при исследовании работы кэширования диска.

Для работы с буферной памятью накопителя, необходимо выбрать пункт меню

Service->Memory->Buffer RAM Operations

При этом на экран будет выдан следующий диалог:



Поле **Sector** задаёт начальный сектор буферной памяти для выполнения операции

Поле **Count** задаёт число секторов, подлежащих передаче. При его задании, учитывайте реальный объём кэш-памяти накопителя. Также учитывайте, что не вся память отведена под буфер.

Кнопки **Read From Disk** и **Write To Disk**, производят чтение буферной памяти с диска и запись буферной памяти на диск соответственно.

Кнопки **Load From File** и **Save To File** считывают и записывают данные в файл.

Кнопка **Dump** позволяет просмотреть и отредактировать содержимое внутреннего буфера программы

Кнопка **Exit** закрывает диалоговое окно.

Ручная подача команд

Программа позволяет подавать любые команды накопителю вручную. Если какой-то механизм в программе не реализован, Вы всегда можете выполнить сколь угодно сложную последовательность действий, пользуясь только диалогом ручной работы. Правда, на практике, этот диалог обычно используется для исследования действий тех или иных команд.

HandWork for WD-MARVEL at HRT-UDMA

Data	----		Go!	Load Regs	<input type="checkbox"/> Read Data Reg
Error	00,1e	04,04	Go -> Buf	Go <- Buf	<input checked="" type="checkbox"/> Ignore when BSY
Count	00,00	0C,42	Get buf	Send buf	<input type="checkbox"/> Swap on reading
Sector	00,00	00,00	Load from file	Save to file	<input type="checkbox"/> Always on Top
Cyl Low	00,00	44,44	Dump	Select Port	<input type="checkbox"/> Monitor mode
Cyl High	00,00	57,57	Make Buf		<input type="checkbox"/> Loop mode
Head	e0	E0,E0			<input checked="" type="checkbox"/> Auto Save State
Com	9a	51,51			<input type="checkbox"/> Make Log
Alt. state	0	51			
Dev. addr	00	0	Timeout	20000 ms	0 bytes in the buffer

Quick commands:

Add Go Extract Exit

Helper: read with retry

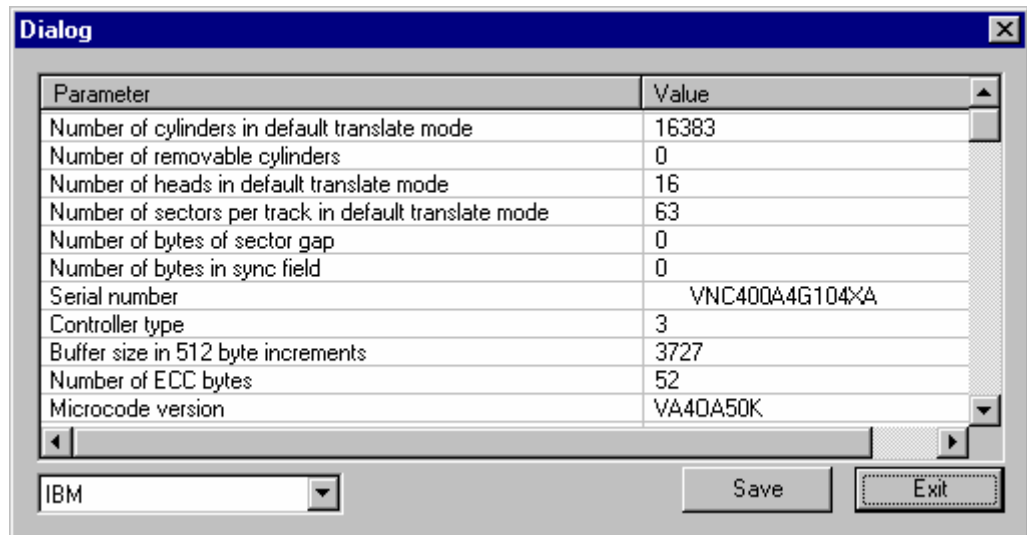
Get from ini Add to ini

Подробнее диалог подачи команд описан в документации на HRT DRE.

Информация о накопителе

Для получения полной информации о накопителе, необходимо выбрать пункт меню Info->Drive Info.

При этом на экран будет выдан следующий диалог:



Информация о накопителе, берётся из его паспорта (отдаваемого в ответ на команду ECh).

В списке отображаются имена параметров и их значения.

Выпадающий список выбора, определяет способ декодирования паспорта (некоторые поля у различных накопителей имеют разное название).

Кнопка SAVE сохраняет содержимое списка в текстовом файле. Это может быть удобно при исследовании некоторых недокументированных команд, влияющих на паспорт (сохраняем два паспорта в текстовых файлах, а затем, средствами ОС сравниваем их и находим различия в текстовом виде).

Кнопка Exit закрывает диалоговое окно.

Правила декодирования паспорта задаются в инициализационном файле info.ini. Таким образом, по мере выхода новых стандартов (либо по мере изучения тех или иных полей), Вы можете сами создавать секции ini-файла. Имя секции «по-умолчанию», совпадает с текстом заголовка окна накопителя (для накопителей Fujitsu всегда берётся секция [Fujitsu]).

Возможны следующие форматы декодирования:

Десятичное число.

NAME=offset DEC shift mask

offset – смещение в паспорте (в словах)

DEC – сигнатура декодирования в виде десятичного числа

shift – на сколько надо сдвинуть число, чтобы необходимые биты попали на то место, где они должны отображаться

mask – какую маску необходимо наложить, чтобы удалить лишние (незначащие для данного параметра) биты.

Пример. Отобразить число логических цилиндров. Смещение слова = 1. Параметр занимает всё слово, то есть, ни сдвигать, ни накладывать маску нам не нужно. Получаем следующую строку:

Number of cylinders in default translate mode=1 DEC 0 0xffff

Усложним пример. Отобразить активный режим Single Word DMA. Он располагается в старшем байте слова по смещению 62. Таким образом, надо сначала сдвинуть слово на 8 бит, а затем – наложить маску 0xff, чтобы оставить только младший байт. Получаем следующую строку:

Single word DMA transfer mode active=62 DEC 8 0xff

32-разрядное значение.

NAME=offset LONG

offset – смещение в паспорте (в словах)

LONG – сигнатура 32-разрядного числа

Пример. Отобразить объём накопителя в LBA. Смещение в паспорте равно 60. Получаем следующую строку:

Total number of User Addressable Sectors (LBA Mode only)=60 LONG

Строка

NAME=offset **STRING length**

offset – смещение в паспорте (в словах)

STRING – сигнатура строки

length – Длина строки (в символах)

Пример. Отобразить имя модели. Смещение в паспорте равно 27, длина – 38 символов. Получаем строку

Model number=27 STRING 38

Битовое поле (означающее «да» или «нет»)

NAME=offset **BOOL bit_no**

offset – смещение в паспорте (в словах)

BOOL – сигнатура битового поля

bit_no – номер анализируемого бита.

Пример. Поддержка или неподдержка упреждающего чтения, задаётся битом 6 слова 82. Получаем строку:

LOOK AHEAD supported=82 BOOL 6

Выбор

NAME=offset SWITCH bit_no value(0/1) decode

offset – смещение в паспорте (в словах)

SWITCH – Сигнатура поля выбора

bit_no – Номер анализируемого бита

value(0/1) – значение бита, при котором будет использована данная строка

decode – текстовая расшифровка значения.

Пример: Поле Standby Timer может принимать два значения - values as specified in ATA standard are supported и values are vendor specific. Это зависит от назначения бита 13 в слове 49. Получаем две строки:

Standby timer=49 SWITCH 13 1 values as specified in ATA standard are supported

Standby timer=49 SWITCH 13 0 values are vendor specific

В зависимости от того, какое значение отдаст накопитель, будет использована либо первая, либо вторая строка.

Равенство

NAME=offset EQUAL mask value

offset – смещение в паспорте (в словах)

EQUAL – Сигнатура равенства

mask – маска, которую надо наложить на слово перед сравнением

value – константа, с которой будет вестись сравнение

В отличие от выбора, здесь сравнение может вестись не с отдельными битами, а с целыми битовыми полями. Строка будет отображена только в том случае, если битовое поле равно заданному значению.

SMART

Общие сведения

Современные накопители содержат в себе систему самодиагностики, позволяющую пользователю дать оценку текущему состоянию накопителю. Некоторые даже пытаются по показаниям SMART предсказывать время выхода накопителя из строя и очень расстраиваются, что это произойдёт через каких-то 10 лет (забывая, что накопители в наше время морально устаревают гораздо быстрее).

В данном документе, мы не будем останавливаться на пользовательском понимании системы SMART, так как оно хорошо описано в большом количестве литературы, а рассмотрим его внутреннюю структуру.

Физически, сведения об атрибуте в рассматриваемом накопителе, хранятся в трёх ячейках.

Ячейка 1 - Число ошибок, накопленное при позапрошлом измерении

Ячейка 2 - Число ошибок, накопленное при прошлом измерении

Ячейка 3 - Копилка для текущего измерения

Всё очень просто. Если возникла ошибка, значение ячейки 3 увеличивается на единицу. Когда будет считано примерно 1220000 секторов, произойдёт сдвиг ячеек. Данные из ячейки 2 попадут в ячейку 1, данные из ячейки 3 попадут в ячейку 2, а ячейка 3 очистится (копилка ошибок обнулится).

Если же будет дана команда "считать данные атрибутов", то накопитель просуммирует значения ячеек 1 и 2 (так сказать, усреднит показания по времени) и пронормирует результат так, чтобы он уложился в диапазон от 1 до 100, а также 1 была минимумом, а 100 - максимумом.

Таким образом, можно сделать два вывода:

1) Искусственная правка атрибутов в служебной области, конечно, поможет обмануть пользователя, но вскоре правда всплывёт наружу, так как по прочтении 1220000 секторов, атрибут 01 сам обновится

2) Если мы ампутировали головку, которая "просадила" атрибут, то для восстановления атрибута надо считать 2 раза по 1220000 секторов. Почему 2 раза? Просто после первой порции плохое число перескочит из ячейки 2 в ячейку 1, а лишь после второй порции, оно будет забыто навсегда.

И напоследок, рассмотрим вариант, что мы, не дочитав немного до требуемого количества секторов, выключили питание накопителя. Что же произойдёт со счётчиком? Неужели он обнулится и всё придётся делать заново? Нет. Он просто будет считан из служебной области. Поэтому, если он не был заблаговременно сохранён, то считается то, было сохранено в последний раз. А как же сделать, чтобы записались текущие ячейки и счётчик? Очень просто. Они либо сохраняются автоматически, если вызвать команду SMART "Autosave Attributes", либо они будут сохранены при считывании текущего значения атрибутов. Таким

образом, регулярно проверяя атрибуты, Вы гарантированно сохраните "натикавшее" значение счётчика на диске.

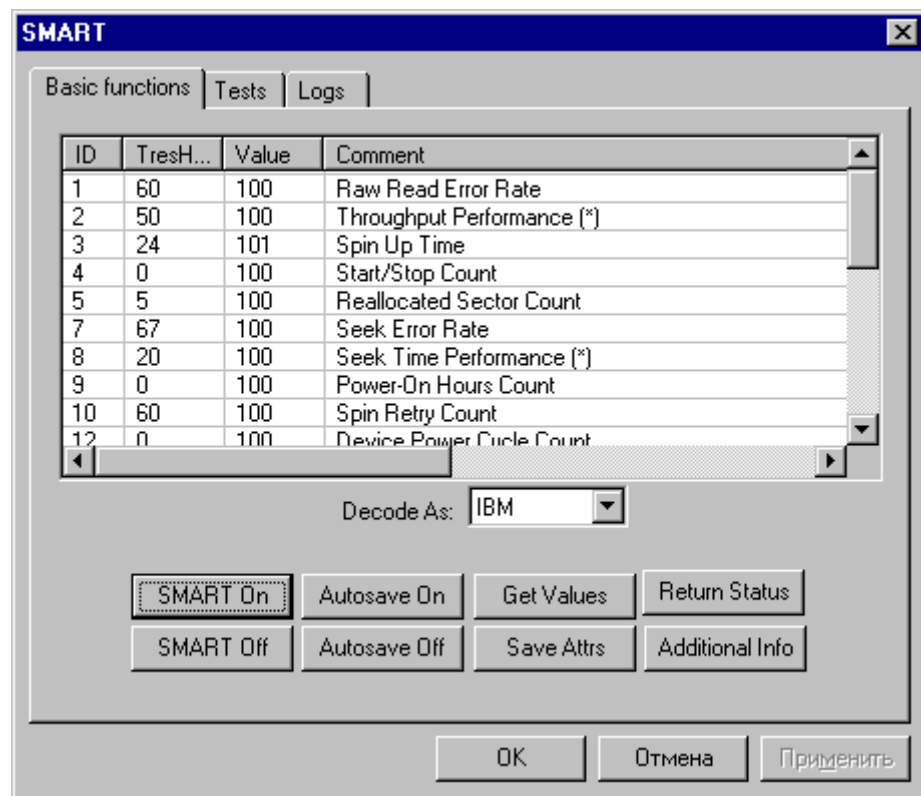
Большинство прочих атрибутов вычисляются примерно так же. У них тоже есть счётчики и три ячейки. Только условия увеличения счётчиков и значения, до которых они доходят для сдвига ячеек, у каждого атрибута свои.

Но есть и исключения. Например, атрибут *Reallocated Sectors Count* вычисляется на основе записи, кодирующей число дефектов в *G-LIST*.

Существуют также атрибуты, которые обновляются только после выполнения внутренних тестов SMART. Как их выполнить, будет описано ниже.

Работа с атрибутами

Чтобы просмотреть атрибуты SMART, а также выполнить иные действия, связанные с данной системой, необходимо выбрать пункт главного меню *Info->SMART*. При этом на экран будет выдан следующий диалог:



Если система SMART выключена, то будет выдано сообщение об ошибке и список в диалоге не будет заполнен. Попробуйте нажать последовательно кнопки **SMART On** и **Get Values**. Скорее всего, после этих действий, список окажется заполненным. Если же нет – значит у накопителя что-то не в порядке с системой SMART и его надо предварительно восстановить более сложными средствами (например, перезаписать модуль SMART, после чего выключить и включить питание).

Список выбора **Decode As** определяет способ декодирования имён атрибутов для тех случаев, когда атрибут характерен для тех или иных накопителей. Это связано с тем, что часть атрибутов определена в стандарте ATA, а часть – определяется производителями так, как им захочется.

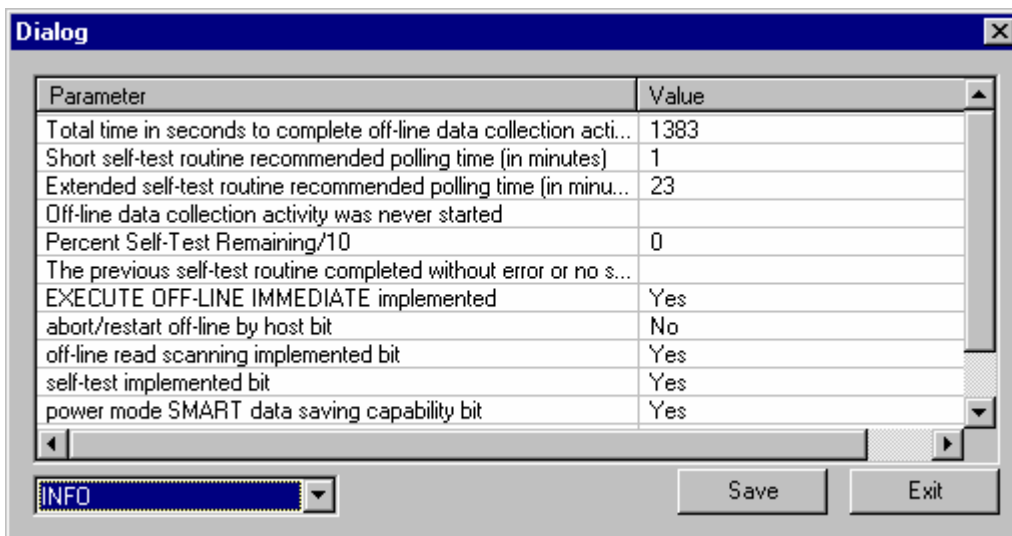
Правила декодирования атрибутов задаются в файле **SMART.INI**. Для каждого атрибута сначала производится попытка найти расшифровку в специализированной секции (в данном случае – **Fujitsu**), затем, при неуспехе – в секции **GENERAL**. Если же и там нет расшифровки, то поле **Comment** остаётся пустым. Декодирование через INI-файл, позволяет Вам гибко настраивать работу программы. Если Вы выяснили назначение тех или иных атрибутов (экспериментально, либо из документации), нет необходимости переделывать EXE-файл, достаточно подправить файл **SMART.INI**.

Описание атрибутов, можно узнать из руководства по эксплуатации накопителя, выложенного на сайте фирмы-производителя.

Кнопки **SMART On**, **SMART Off**, **Autosave On**, **AutoSave Off**, **Save Attrs** и **Return Status** посылают в накопитель соответствующие команды, согласно стандарту ATA.

Кнопка **Get Values** считывает с накопителя текущие значения атрибутов и заносит их в список.

Кнопка **Additional Info** берёт из накопителя дополнительную информацию SMART. При этом на экран выдаётся диалог следующего вида:

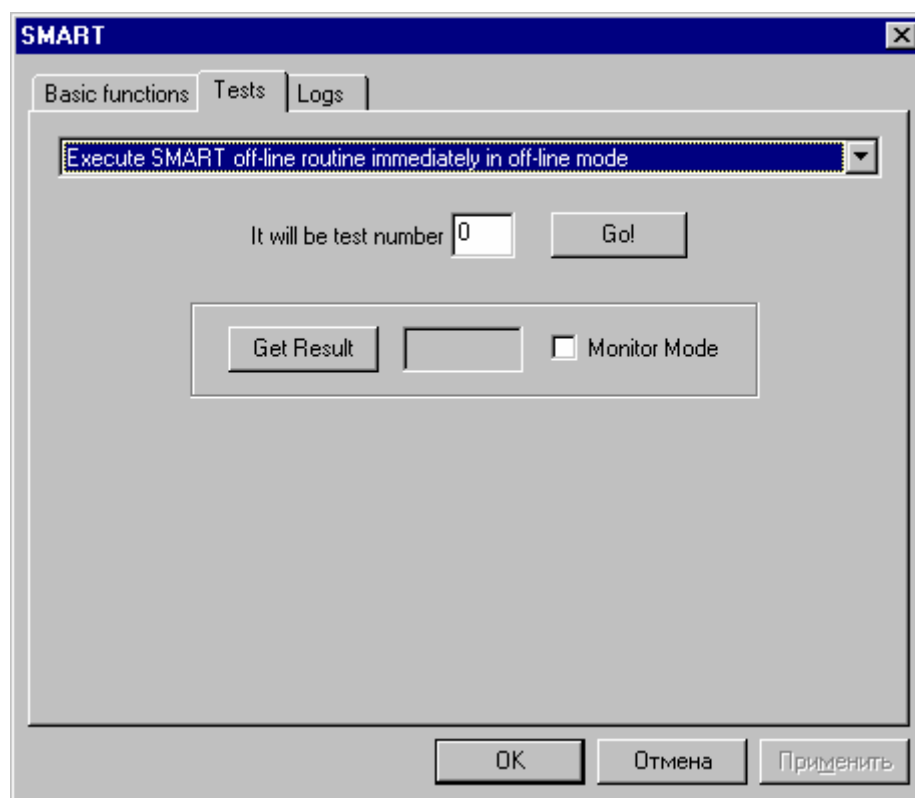


Дополнительная информация берётся с помощью стандартной ATA-команды. Правила декодирования дополнительной информации задаются секцией [INFO] файла SMART.INI. Формулы декодирования полностью аналогичны формулам декодирования информации о накопителе.

Тесты самодиагностики SMART

Система SMART, кроме того, что сообщает пользователю о состоянии накопителя, может также проводить те или иные тесты, собирающие сведения о

состоянии. Для вызова соответствующих функций, предназначена вкладка **Tests**. Её внешний вид представлен на рисунке ниже.



Список возможных тестов заполняется на основе секции **[OFFLINE]** файла **SMART.INI**. По умолчанию, туда введены значения, приведённые в фирменной документации Fujitsu. По мере совершенствования накопителей и появления новых тестов, Вы можете вводить туда новые значения.

Если выбрать тот или иной тест, его код появится в поле **It will be number...** В принципе, номера тестов можно вводить и непосредственно в данное поле, без выбора в списке.

При нажатии кнопки **Go**, программа запустит на выполнение тест с выбранным номером. Если тест выполняется долго, то программа выдаст соответствующее предупреждение и предложит Вам подождать окончания выполнения самостоятельно. Это будет видно по снятию сигнала BSY на панели светодиодов.

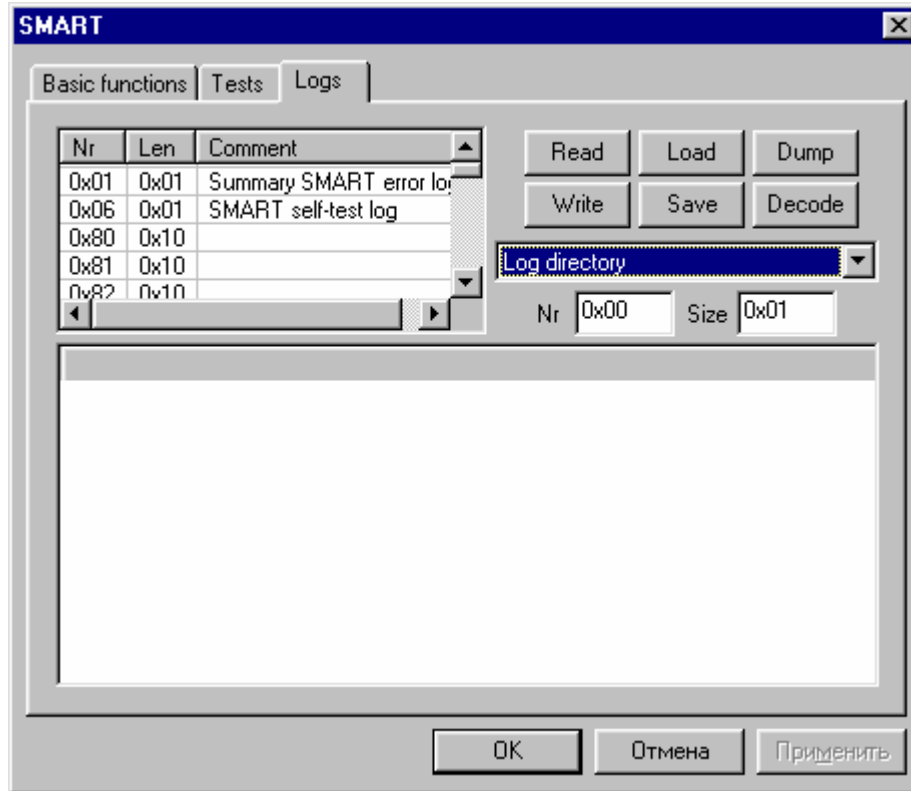
Некоторые тесты выполняются в «прозрачном» режиме, только во время простоев (когда к накопителю нет обращений). В этом случае, индикатор BSY не загорается, а о прохождении теста можно судить по данным в регистре цилиндра. Чтобы считать показания этого регистра, нажмите кнопку **Get Result**. Чтобы запустить циклическое чтение регистра, установите флажок **Monitor Mode**. Правила декодирования результата зависят от теста и описаны в документации на накопитель.

Просмотр логов SMART

Система SMART позволяет просмотреть логи ошибок, возникавших в процессе работы накопителя. К ним относятся ошибочные команды, «посмертные» дампы памяти и многое другое.

Так как, по мнению авторов, практического смысла в ремонте накопителей, логи не дают, их поддержка в программе минимальна и введена скорее для опытов, так как может быть, они пригодятся для каких-либо целей.

Внешний вид вкладки Logs представлен ниже.



Левый верхний список содержит каталог возможных логов. При выборе того или иного элемента списка, номер и длина выбранного лога попадут в поля **Nr** и **Size** соответственно. Текстовое описание логов берётся из секции [LOGS] файла SMART.INI.

Для накопителей, которые не поддерживают каталог логов, Вы можете выбирать логи, описанные в стандарте в выпадающем списке. Их перечень берётся из секции [LOGS] файла SMART.INI.

Кроме того, Вы можете заполнять эти поля самостоятельно.

Кнопка **Read** пошлёт на накопитель запрос на считывание лога (правда, не факт, что накопитель поддерживает логи данного типа).

Кнопка **Save** сохранит полученный из накопителя массив в файле.

Кнопка **Dump** позволит просмотреть дамп с буфером, возвращённым накопителем.

Прочие кнопки пока бездействуют. Если Вы хотите автоматизировать процесс декодирования лога, то можете сохранить буфер в файл и воспользоваться внешними программами.

Работа со служебной областью

Общие сведения

Для различных накопителей, работа со служебной областью осуществляется различными способами. Описание сути процессов Вы найдёте в документации на конкретный накопитель. В данном документе описаны основные диалоги, используемые для работы со служебной областью различных накопителей.

Служебная область накопителя содержит основную часть информации, необходимой для нормальной работы. Там содержатся резидентный модуль микропрограммы, оверлейные модули, служебные таблицы, протокол самотестирования, код самотестирования и множество других параметров. Без многих из них накопитель не может работать, но некоторые же модули являются чисто информационными. Так, например, протокол тестирования накопителя на заводе хоть и может пригодиться для восстановления разрушенного транслятора, но в повседневной работе он не нужен.

Большинство жизненно важных модулей накопителя, вынесено в специализированную карту. Это означает, что если Вы захотите узнать положение того или иного модуля, Вам достаточно справиться об этих данных в карте. Однако следует учитывать, что у накопителя IBM карта является чисто информационной. Если её изменить, накопитель всё равно будет искать модули по тем же координатам, что и ранее, так как эти координаты «защиты» в ПЗУ, а накопитель WD – наоборот, послушно сдвинет свои модули на новое место.

Кроме того, следует помнить, что не все модули помещены в карту. Так, например, в карту накопителя IBM не входит модуль, содержащий серийный номер накопителя. То есть, если сохранить в файле все модули, входящие в карту, то после восстановления, накопитель всё равно может не начать работать. Из этого факта, вытекает понятие полной и сокращённой служебной области.

Сокращённой служебной областью, мы будем называть те её модули, которые включены в карту. Часто их редактирование позволяет изменять структуру накопителя и производить ремонт. Но запись сокращённой служебной области от заведомо исправного накопителя не гарантирует того, что ремонтируемый накопитель начнёт работать в обычном режиме.

Полной служебной областью, мы будем называть посекторный образ диска в области, где расположена служебная область. Мы можем не знать назначения некоторых (и даже многих) секторов из этой области, но если они есть у накопителя, то зачем-то они туда были записаны, а производителю – виднее.

Однако, даже полная служебная область не является панацеей. В модулях служебной области содержатся таблицы, настраивающие программу под данный конкретный тип механики. Они учитывают множество факторов, включая такие экзотические, как упругость гибкого шлейфа БМГ и сопротивление магниторезистивного элемента. Поэтому бездумная запись таких таблиц от расшатанного гермоблока, может вызвать проблемы при работе накопителя, параметры гермоблока которого ещё не так расшатаны.

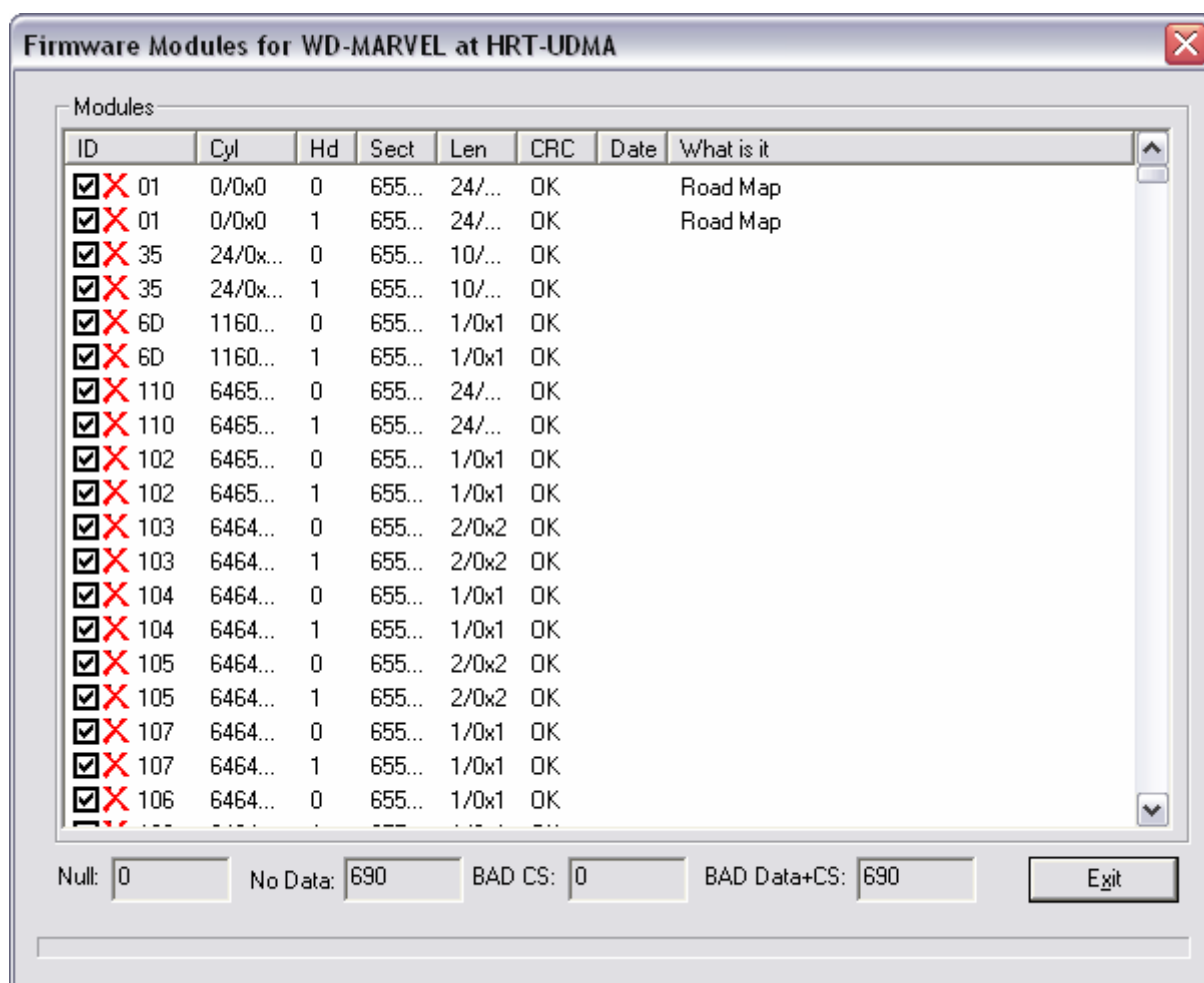
Какой может быть сделан вывод из всего вышеизложенного? Если у накопителя цела хотя бы часть модулей, включённых в карту, необходимо постараться сохранить их в файл. Когда Вы запишите служебную область от другого накопителя – обязательно восстановите те крупницы, которые удалось спасти от «родной» механики. Это может оказаться очень полезным.

Работа с сокращённой служебной областью (WD, IBM)

Для работы с сокращённой служебной областью, необходимо выбрать пункт меню

Service->Special Area->Structure

При этом на экран будет выдан следующий диалог:



Столбец **ID** содержит имена модулей служебной области. Флажок около него означает, что модуль будет участвовать в операциях. Иногда полезно исключить модуль из операций, сняв флажок при помощи «Мыши», либо при помощи меню Select.

Столбец **Cyl** содержит РВА модуля в десятичном и шестнадцатеричном виде.

Столбец **Head** содержит номер головки, на которой расположен модуль

Столбец **Sect** для накопителей IBM не имеет смысла

Столбец **Len** содержит длину модуля в секторах

Столбец CRC для накопителей IBM не имеет смысла, а для накопителей WD он отображает правильность контрольной суммы модуля. Если там написано BAD, значит модуль повреждён.

Столбец Date для накопителей WD отображает дату создания модуля.

Модуль What s it содержит расшифровку назначения модуля. Он заполняется на основании секции [MODULES] инициализационного файла накопителя (IBM.INI, либо WD.INI). Вы можете самостоятельно давать расшифровки для модулей, назначение которых стало Вам известно.

Галочка около имени модуля означает, что он будет участвовать в групповых операциях. По умолчанию, все галочки установлены. Иногда бывает полезно проводить операции только с отдельными модулями (по определённой головке, цилиндру, по назначению модуля и т.п). В этом случае, Вы можете включать и исключать модули из операции по своему усмотрению.

Красные кресты около имён модулей, означают, что они не загружены в память. Чтобы выполнить любые действия, наведите курсор «Мыши» на список и нажмите правую кнопку. При этом на экране появится контекстное меню

Чтобы считать с диска все модули, выберите пункт **I/O ->Read All Blocks**. Если Вас интересует конкретный модуль – выделите его и выберите пункт **I/O-> Read From HDD**.

Бывают ситуации, когда у накопителя явно неисправна одна головка. В этом случае, можно установить курсор на любой модуль, расположенный на живой головке, после чего выбрать пункт меню **I/O->Read All Blocks from current head**. *Операция оставлена для совместимости со старыми версиями, в современной версии полным аналогом будет последовательность **Select->Unselect All, Select->Current head, I/O->Read All Blocks**.*

Чтобы отредактировать модуль, выберите пункт **Dump**. Отредактированный модуль можно записать в одном экземпляре с помощью кнопки **I/O ->Write To Hdd**. Если же Вы хотите, чтобы записались все копии данного модуля – выберите пункт **I/O -> Write All Copies**. Если Вы пока не хотите записывать отредактированный модуль на диск, но хотите, чтобы изменения были произведены во всех копиях, выберите пункт **Block->Duplicate**. Этот же пункт может пригодиться для размножения успешно считанного модуля на несчитавшиеся места.

Маленькая хитрость. Двойной щелчок по строке с незагруженным модулем, произведёт попытку чтения указанного модуля, двойной же щелчок по строке с загруженным модулем, откроет дамп модуля.

Сохранить выделенный модуль в файле можно при помощи пункта меню **File Ops -> Save Module**. Сохранение произойдёт в двоичном виде. Такой модуль можно редактировать с помощью внешнего редактора, либо использовать для других целей. Чтобы загрузить модуль в выделенную позицию, необходимо выбрать пункт меню **File Ops->Load Module**.

Если Вы хотите сохранить все модули в виде одного файла, выберите пункт **File ops->Save Archive**.

Загрузить архив можно при помощи пункта меню **File Ops->Load Archive**. Это одна из немногих групповых операций, при которых галочки около модулей

игнорируются, ведь у архива имеется своя карта, которая вполне может не совпадать с той, которая в данный момент отображается на экране. Для загрузки архива с учётом галочек имеется пункт меню **File Ops->Load to current map**. При этом карта из файла будет проигнорирована, загрузятся только тела модулей. Разумеется, в таком случае галочки можно и проанализировать.

Пункт меню **File Ops->Repair From BLA** аналогичен пункту **File Ops->Load to current map**, но загружены будут только те модули, которые в настоящий момент отсутствуют в памяти. Это удобно для восстановления разрушенной служебной области. Как уже неоднократно отмечалось, необходимо стремиться сохранить КАК МОЖНО БОЛЬШЕ родных модулей. Но если какие-то модули разрушены, уже ничего не поделаешь. В таком случае, считываем модули с дисков, а затем – выполняем **Repair from BLA**, чтобы подгрузить ТОЛЬКО потерянные модули от другого накопителя.

Чтобы записать все модули, находящиеся в буфере (помеченные зелёными галочками) на диск, выберите пункт меню **I/O->Write All Blocks**. Если в памяти находится несколько копий модуля с одним и тем же именем, то в файл попадёт первая из них. Дубли в файл **НЕ ПОПАДУТ**.

Обратите внимание на разницу между пунктами **I/O->Write All Copies** (записать все копии выделенного модуля) и **I/O->Write All Blocks** (записать все блоки, какие только есть).

Пункт **Block->Recalc CRC** активизирует пересчёт контрольной суммы (для накопителей IBM он не имеет смысла, так как их модули не содержат контрольной суммы, что порождает просто огромное количество проблем). Это может потребоваться, например, если Вы подправили тело модуля в дампе.

Пункт **Block->Delete Content** удалит тело выделенного модуля.

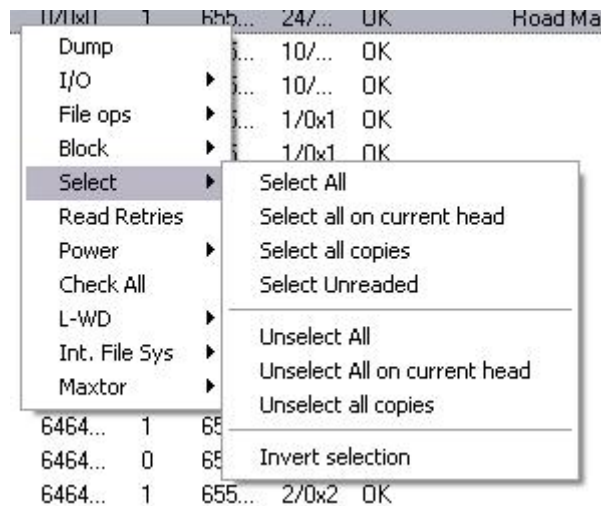
Пункт **Block->Decode** вызовет внешний модуль декодирования тела модуля. Вы можете самостоятельно написать декодировщик в виде внешнего файла DLL, который будет детально расшифровывать тело модуля. Это может быть расшифровка каких-либо структур, либо иных сущностей.

Группа **Select** позволяет устанавливать или снимать галочки с модулей.

Пункт **Check All** позволяет более наглядно проверить тела модулей на правильность контрольной суммы. Текст “OK” или “BAD” слабо бросается в глаза, а после выбора пункта Check All, модули с неверной контрольной суммой будут выделены красным значком.

Щелчок по заголовку любого столбца, отсортирует записи в порядке возрастания значений.

Группа меню Select позволяет устанавливать и снимать флажки с записей по тем или иным признакам:

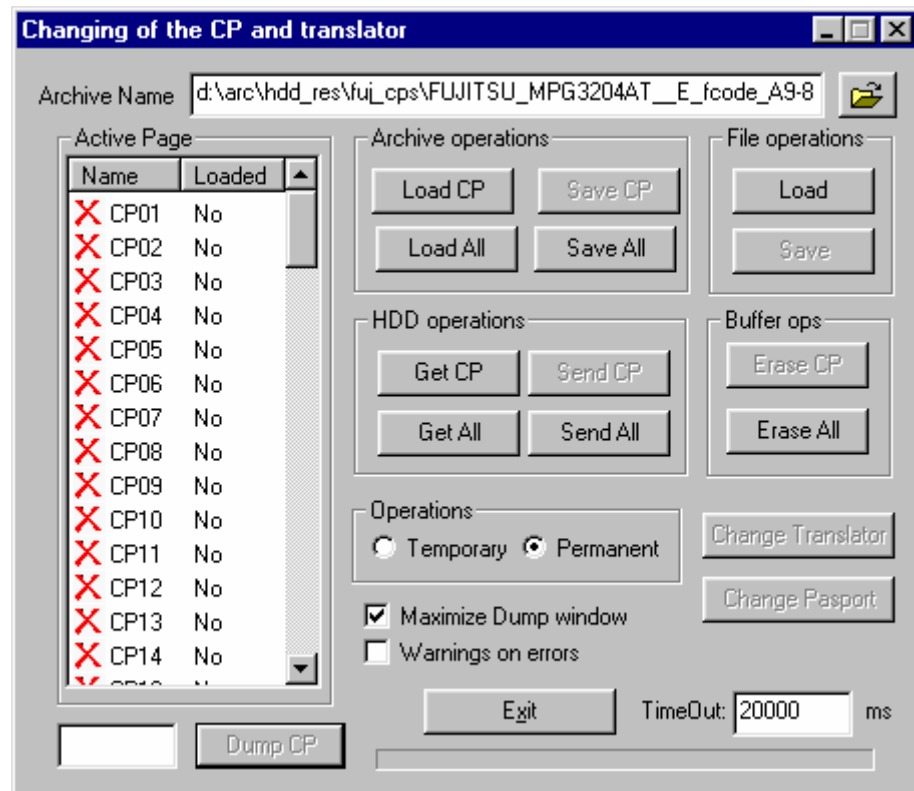


Данный диалог и формат файлов VLA являются мощным инструментом для ремонта и ручной переконфигурации накопителей.

Работа с конфигурационными страницами (Fujitsu, WD)

Программа HRT для накопителей Fujitsu, работает через штатные команды накопителя, дающие доступ к модулям служебной области. По сложившейся традиции (традиция сложилась ещё до введения в HRT накопителей Fujitsu), эти модули называются конфигурационными страницами (CP). Дело в том, что данные модули очень похожи на конфигурационные страницы накопителей Quantum, поэтому при вводе поддержки Fujitsu, идеология была просто распространена на них.

Для доступа к модулям служебной области, выберите пункт меню Service->CP Operations. при этом, программа откроет следующий диалог:




Слева расположен список модулей. Обратите внимание, что не все модули представлены в списке. Это связано с тем, что некоторые модули не используются в тех или иных подсемействах, а при обращении к некоторым из них - накопитель «зависает». Перечень модулей, попадающих в список, определяется файлом FUJITSU.INI, описанным ниже. В целом, правило простое. Если какой-то модуль просто не читается на всех накопителях – ничего страшного, его можно и не исключать из списка, а вот модули, приводящие к «зависаниям» исключать просто необходимо.

Модули загружаются в память программы. Около идентификаторов модулей, которые не загружены, стоит красный крест. Около идентификаторов загруженных модулей стоит зелёная галочка.

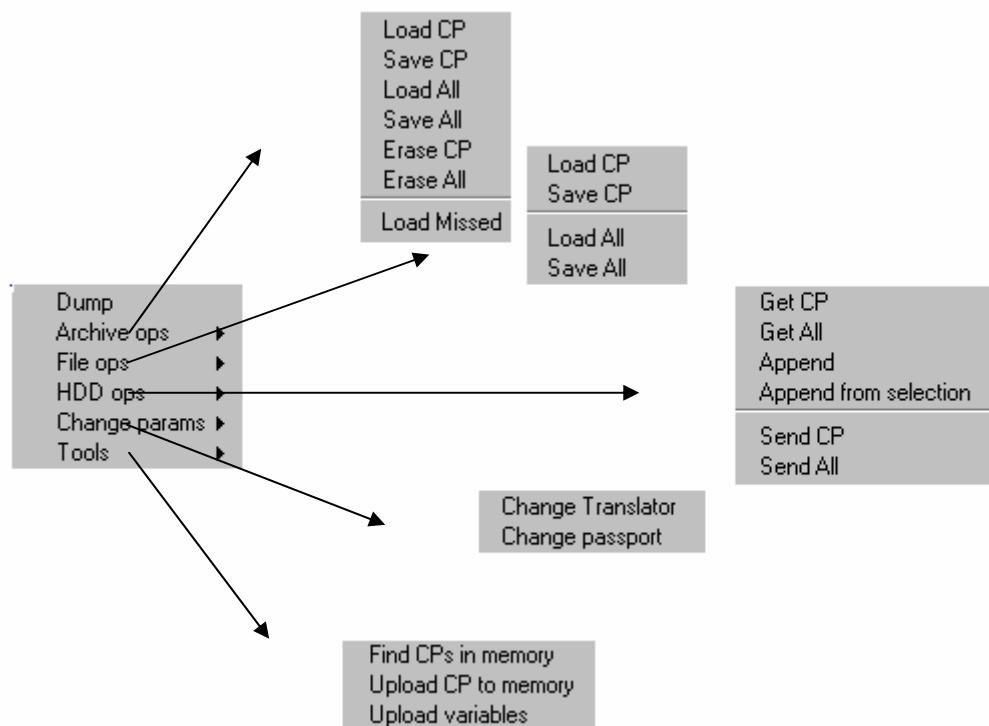
Группа радио кнопок **Operations** определяет тип работы с модулями. Если выбрана кнопка **Permanent**, работа будет вестись непосредственно с модулями на диске, если **Temporary** – с их образами в памяти. Режим **Temporary** нужен для первичной загрузки непроинициализированного накопителя, ведь если он не проинициализирован, то и на диски записать ничего не сможет. Если же залить модули в память, то диск станет более «понятливым» и, может быть, даже позволит сразу после этого записать данные и в режиме **Permanent**. Аналогично, режим **Temporary** позволяет иногда вычитать из памяти те модули, которые недоступны в режиме **Permanent** (приводят в этом режиме к долговому «хрюканью головками»).

Флажок **Maximize Dump Window** введён исключительно для совместимости со старыми версиями программы, которые не умели открывать окно дампа в режиме «во весь экран». В настоящий момент, авторы не видят причин, по которым этот флажок стоило бы снимать, кроме привычки.

Флажок **Warnings on errors** необходим для выдачи предупреждений при операциях групповой записи модулей в накопитель и будет описан ниже.

Сверху расположена строка редактирования имени архива, с которым будет вестись работа. Вы можете самостоятельно отредактировать имя в этой строке, либо выбрать его, нажав кнопку  справа от строки. Путь и имя файла по умолчанию, задаются в файле HDD.INI (описан ниже).

По правой кнопке «мыши», Вы можете вызвать контекстное меню. Наиболее часто используемые пункты контекстного меню продублированы и в виде кнопок диалога, но часть функций может быть вызвана ТОЛЬКО через контекстное меню. В нашем описании, будем базироваться именно на пунктах меню, указывая отдельно имя кнопки, дублирующей пункт. Структура меню следующая:



Пункт **Dump** (кнопка **Dump CP**) позволяет просмотреть и отредактировать шестнадцатеричный дамп выбранного модуля. Если дважды щёлкнуть «мышью» по загруженному модулю в списке, программа также выдаст его дамп.

Пункт **Decode** вызывает внешний модуль декодирования тела конфигурационной страницы. Он позволяет произвести детальный разбор тела. Вы можете создать набор своих разборщиков в виде файлов DLL (описаны ниже).

Группа **Archive Ops** (как в меню, так и в диалоге) объединяет в себе элементы для работы с архивом модулей. Архив – это файл, содержащий в себе все модули, загруженные в данный момент в память. Расширение файла архива модулей – CPS.

Archive Ops->Load CP – загрузить выделенный модуль из архива в память. Если в текущем файле архива выбранный модуль отсутствует, загрузки не произойдёт.

Archive Ops->Save CP – в данный момент, функция не реализована, так как она выбивается из концепции файлов CPS. Пункт меню введён на случай расширения концепции.

Archive Ops->Load All – считать все модули, имеющиеся в архиве, в память. Следует отметить, что если какой-либо модуль отсутствует в списке, но присутствует в архиве, он всё равно будет считан в память. Для получения доступа к нему в любой из операций, достаточно вписать номер вручную в поле ввода под списком модулей.

Archive Ops->Save All – сохранить все модули, имеющиеся в данный момент в памяти (считанные с накопителя или из различных типов файлов) в текущий архив. Старый архив – теряется. Так, например, пусть у Вас был архив ARCHIVE.CPS, содержащий в себе модули CP01, CP02 и CP04. Вы считали с накопителя модули CP03, CP04 и CP05, после чего выбрали пункт **Archive Ops->Save All**, а в качестве имени файла у Вас выбрано всё то же ARCHIVE.CPS. В результате, старый файл ARCHIVE.CPS будет стёрт, а на диске появится новый файл ARCHIVE.CPS, содержащий модули CP03, CP04 и CP05. Модули CP01 и CP02 пропадут.

Если Вы хотите не потерять содержимое старого архива, Вам надо позаботиться о том, чтобы соединить его с тем, что находится в памяти в данный момент. Это можно сделать, например, с помощью пункта **Archive Ops->Load Missed**. При этом из архива будут загружены те модули, которых в данный момент нет в памяти.

Рассмотрим всё тот же пример. В памяти у нас имеются CP03, CP04 и CP05. Имя архива выбрано ARCHIVE.CPS. Выбираем пункт **Archive Ops->Load Missed**. Так как CP01 и CP02 отсутствуют в памяти, они будут считаны из файла. А CP04 в памяти уже есть, поэтому из файла он считан не будет, а в памяти останется то, что было занесено в неё ранее. Таким образом, теперь выбираем пункт меню **Archive Ops->Save All** и получаем архив с модулями CP01, CP02, CP03, CP04 (не тем, который был в архиве, а тем, который был в памяти) и CP05.

Однако, истинное назначение пункта **Archive Ops->Load Missed** несколько иное, чем собирание коллекций архивов. Дело в том, что у накопителей Fujitsu в силу конструктивных и программных недостатков, часто портится содержимое тех или иных модулей. Можно, конечно, взять сохранённый архив от другого накопителя и залить все его модули «не глядя», но это делать **КРАЙНЕ НЕ РЕКОМЕНДУЕТСЯ!!!** Проблема в том, что среди модулей есть такие, которые содержат калибровочную информацию, совмещающую ДАННУЮ плату электроники с ДАННОЙ механикой (так называемые, адаптивные параметры). Эти параметры подобраны на заводе-изготовителе именно для данного накопителя. Залив вместо них параметры от другого накопителя Вы получите в лучшем случае, менее надёжный накопитель, а в худшем – совершенно неработоспособную вещь.

Поэтому золотое правило ремонтника звучит так: **КАК МОЖНО МЕНЬШЕ ЗАПИСЫВАТЬ В НАКОПИТЕЛЬ ЧУЖИХ МОДУЛЕЙ СЛУЖЕБНОЙ ОБЛАСТИ, ТОЛЬКО ТО, ЧТО ВСЁ РАВНО УЖЕ ПОТЕРЯНО.**

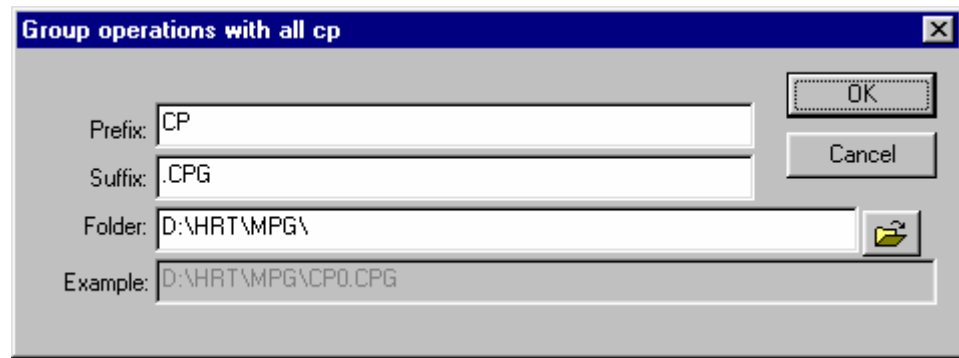
Для реализации данного правила, и введён пункт **Archive Ops->Load Missed**. Сначала считайте все модули из накопителя, а затем – выберите данный пункт. Программа загрузит из архива только те модули, которые не были считаны с дисков накопителя, то есть, постарается свести потери от записи чужих модулей к минимуму.

Пункт **Archive Ops->Erase CP** (продублирован кнопкой диалога **Buffer->Erase CP**) удаляет выделенный модуль из памяти. Это может пригодиться для той же операции **Archive Ops->Load Missed**, чтобы программа взяла модуль именно из памяти, либо для каких-то других действий.

Пункт **Archive Ops->Erase All** (продублирован кнопкой диалога **Buffer->Erase All**) удаляет все модули из памяти программы.


Группа пунктов меню **File Ops** (часть пунктов которой продублирована в группе диалога **File Operations**) работает с модулями, как с одиночными файлами. Пункт **File Ops->Save CP** запишет выделенный модуль в файл, а пункт **File Ops->Load CP** считает файл и поместит его в тот модуль, который в настоящий момент выделен (напоминаем, что если модуль не присутствует в списке, Вы всё равно можете оперировать с ним, вводя его номер вручную).

Два других пункта данной группы меню используются обычно для исследовательских, а не для ремонтных целей, поэтому кнопками диалога не продублированы. Пункт **File Ops->Save All** позволяет сохранить все модули, имеющиеся в памяти в виде отдельных файлов. Если выбрать данный пункт меню, будет выдан следующий диалог:



Prefix – префикс имени файла, формируемого при данной операции

Suffix – окончание имени файла, формируемого при данной операции

folder – Подкаталог, куда будут помещены все файлы. Вы можете отредактировать эту строку, как вручную, так и через диалог выбора подкаталога, нажав кнопку .

Example – пример автоматически сформированного имени файла

После нажатия на кнопку ОК, программа автоматически запишет все модули в виде отдельных файлов, имена которых будут начинаться с префикса, в середине содержать номер модуля, а оканчиваться на заданное окончание. Полученные файлы, Вы можете использовать в исследовательских или иных целях.

Пункт **File Ops->Load All** выполняет обратное действие – загружает одиночные файлы с модулями в память. Вам также будет предложено выбрать префикс, окончание и подкаталог. Все файлы в выбранном подкаталоге, удовлетворяющие указанным Вами условиям, будут загружены в память.

Группа **Hdd Ops** (продублированная группой HDD Operations на диалоге) предназначена для обмена модулями с ремонтируемым накопителем. В зависимости от положения радио кнопок Temporary и Permanent, работа будет вестись с памятью накопителя и с модулями непосредственно на дисках соответственно.

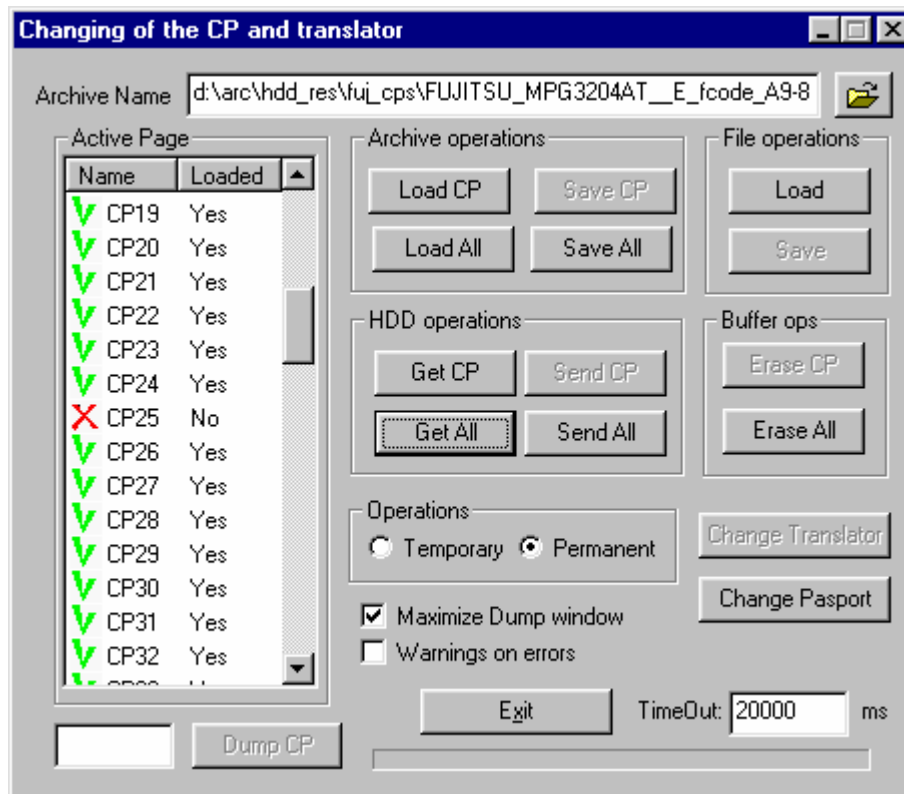
Пункт **Hdd Ops->Get CP** (продублирован диалоговой кнопкой **Get CP**) считывает указанный модуль с накопителя в память. Кроме того, этот же пункт будет вызван, если дважды щёлкнуть «мышью» по модулю в списке, около которого стоит красный крест (модуль ещё не загружен). Один и тот же модуль может считаться по-разному, в зависимости от режима чтения, ведь если модуль не читается с дисков, накопитель вполне может заполнить его значениями по умолчанию. Поэтому в режиме Temp считается содержимое, не считанное с дисков, а построенное программным путём внутренним процессором накопителя.

Пункт **Hdd Ops->Send CP** (продублирован диалоговой кнопкой **Send CP**) – наоборот запишет модуль в накопитель. В режиме Temporary модуль попадёт только в память и просуществует, в общем случае, до выключения питания, а в режиме Permanent, модуль попадёт как в память, так и на диск. Но некоторые модули накопитель анализирует только при старте микропрограммы, поэтому, чтобы накопитель учёл изменения, необходимо выключить и включить его питание.

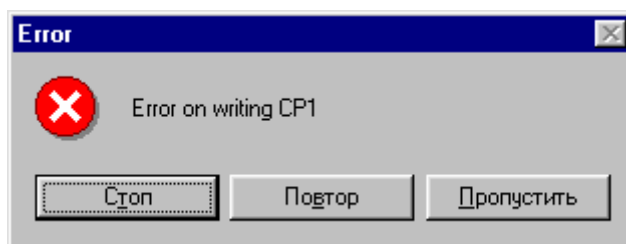
Для накопителей WD конфигурационные страницы имеют иной смысл, чем для остальных накопителей. Физически, они могут располагаться даже в ПЗУ, поэтому

операции записи CP для накопителей WD могут либо не работать вообще, либо работать только в режиме Temp.

Пункт **Hdd Ops->Get All** (продублирован диалоговой кнопкой **Get All**) считывает из накопителя все модули, присутствующие в списке. Если какой-то модуль не считался, то никаких диагностических сообщений не выдаётся, однако, в списке около несчитанного модуля останется красный крест. Например, на приведённом ниже рисунке, считались все модули, кроме CP25 (к слову сказать, он и не должен считываться, но так как обращение к нему не приводит к зависанию накопителя, авторы не стали исключать его из списка). При желании, Вы можете самостоятельно подправить существующий список или даже создать ряд альтернативных списков, учитывающих особенности разных прошивок (путём правки файла Fujitsu.ini).



Пункт **Hdd Ops->Write All** (продублирован кнопкой **Write All**) – наоборот запишет все модули на диск. Однако здесь возникает одна неприятная проблема. Если после чтения, сразу видно, что некоторые модули не считались (в списке присутствуют красные кресты), то при записи единственный способ сообщить об ошибке – выдать предупреждение. Однако, при сильно неисправном накопителе, таких предупреждений может быть множество и они не всегда желательны. Поэтому Вы можете настраивать поведение программы при помощи флажка **Warnings on errors**. Если этот флажок сброшен, то при ошибках записи, программа просто проигнорирует их. Если же он установлен, то при любой ошибке, Вы получите сообщение (тексты кнопок зависят от языковых настроек Windows):



Кнопка **Стоп (Abort)** прервёт операцию записи модулей.

Кнопка **Повтор (Retry)** повторит попытку записи модуля, при которой возникла ошибка

Кнопка **Пропустить (Skip)** пропустит проблемный модуль и перейдёт к записи следующего.

Совет: Если модули не пишутся в режиме **Permanent**, попробуйте сначала записать их в режиме **Temporary**. Возможно, для первичных операций ремонта, этого окажется достаточно. А затем – они зальются и в режиме **Permanent**.

Остальные пункты группы **Hdd Ops** предназначены для особо капризных накопителей, поэтому в виде диалоговых кнопок не продублированы. Существуют накопители, у которых модули читаются нестабильно. Один раз из трёх, из четырёх и т.п. при этом, каждая попытка считывает свой набор модулей. То нет CP02, то CP02 есть, но нет CP08 и т.д. Разумеется, постоянные **Hdd Ops->Get All** ситуацию не спасут, всё время что-то, да не считывается. Именно для этого случая и предназначен пункт меню **Hdd Ops->Append**. При его выборе, будут произведены попытки только тех модулей, которые в настоящий момент отсутствуют в памяти.

Если же какой-то модуль не только не читается, а ещё и приводит к зависанию накопителя, Вы можете выбрать любой модуль под ним и выбрать пункт **Hdd Ops->Append from selection**. При этом будут считаны те модули, которые расположены в списке ниже выделенного, и при этом не загружены в память.

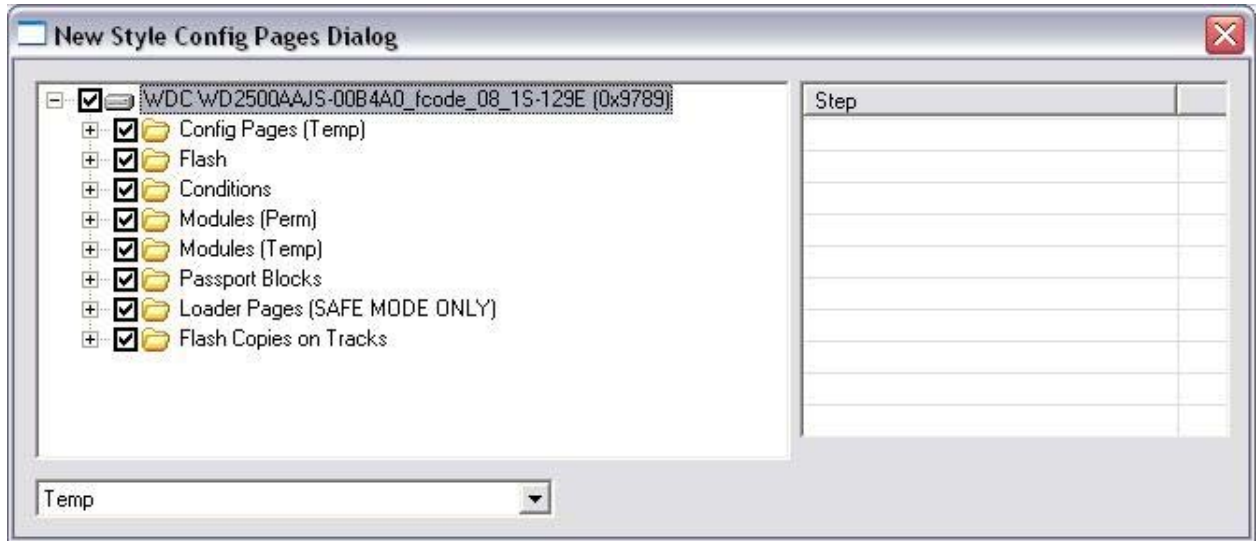
Группы меню **Change Params** и **Tools**, а также диалоговые кнопки **Change Translator** и **Change Passport** для накопителей Fujitsu не имеют смысла.

Новый внешний вид работы с конфигурационными страницами

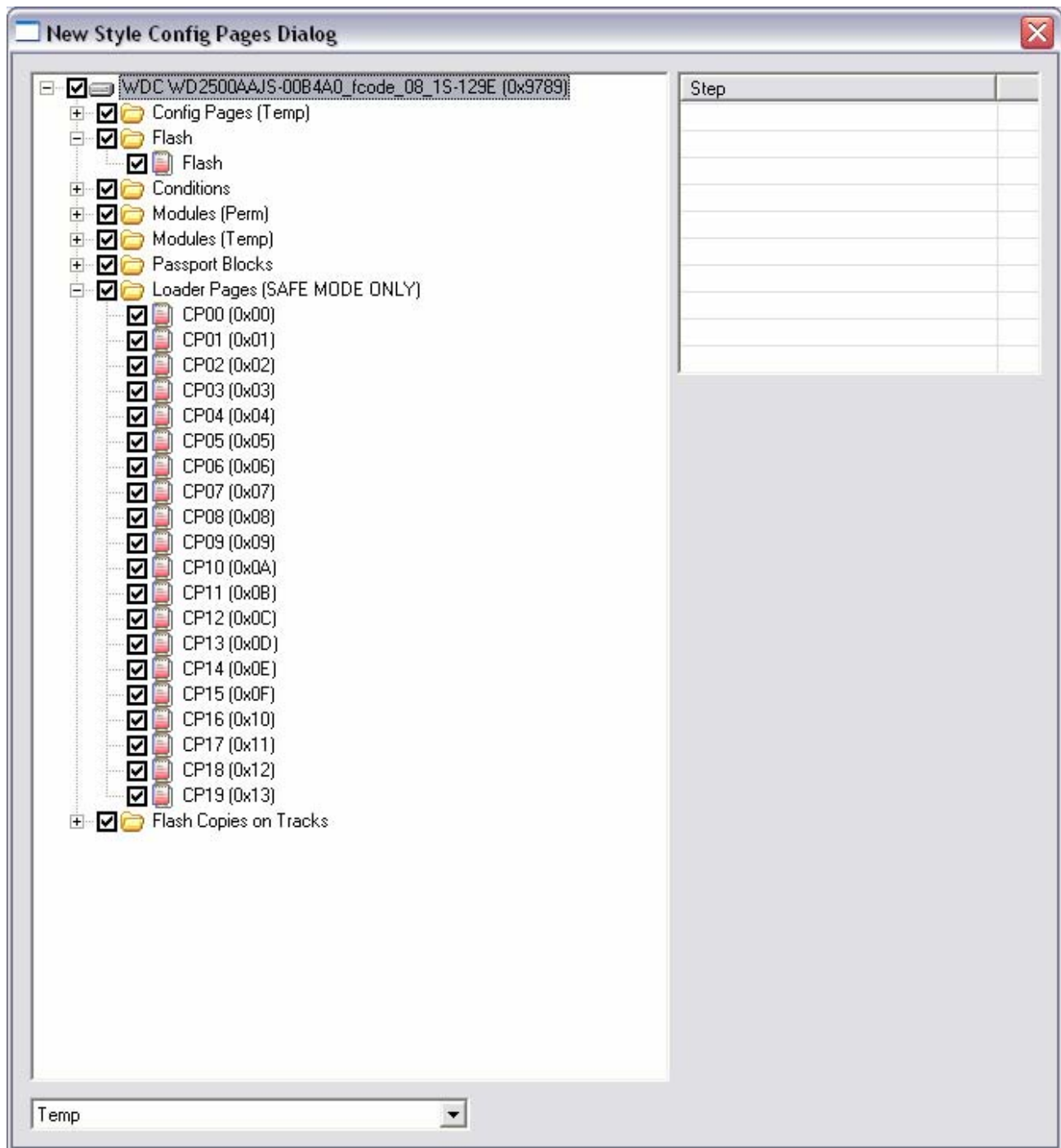
Со временем, у накопителей стало появляться всё больше и больше видов конфигурационных страниц. Если исходно конфигурационные страницы с флажком Temp были подмножеством конфигурационных страниц с флажком Perm (просто те, что загружены в память), то впоследствии у накопителей Fujitsu страницы Temp стали содержать и то, чего невозможно найти в Perm. С другой стороны, у накопителей Western Digital появилось много интересных вещей, оформленных как конфигурационные страницы (то есть, те, которые можно читать по ID). Даже модули служебной области стало возможным читать не только командами чтения с поверхностей, но и через ID.

Возникла проблема: Как сохранять все эти страницы? В виде многих файлов формата CPS? Но пользователю ничего не стоит запутаться в них. Поэтому было

принято решение создать новый формат файла, назвав его СРТ, где хранятся все виды конфигурационных страниц. Для работы с этим файлом, было разработано окно, где страницы представлены в древовидной форме.



Каждая ветвь дерева соответствует своему типу конфигурационных страниц. Если выбрать ветвь, то в ней отобразятся допустимые типы конфигурационных страниц:

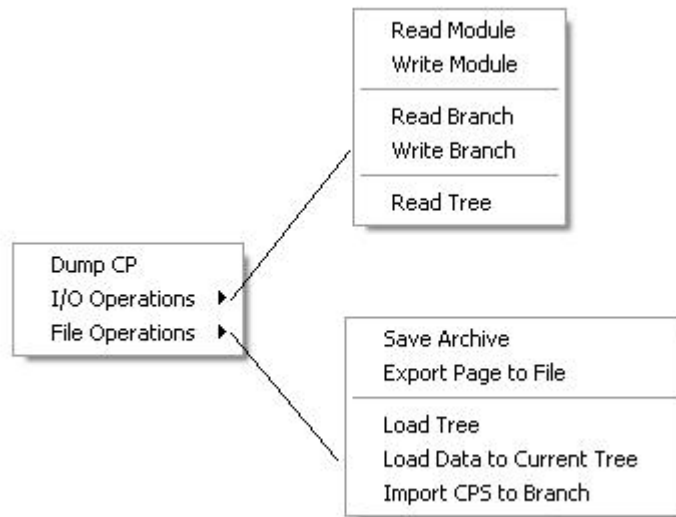


Как видно из рисунка, в конфигурационные страницы попало даже то, что к ним не имеет прямого отношения. Например, содержимое флэш-памяти. Это сделано умышленно, чтобы все важные сущности сохранились в одном и том же файле.

Список справа предназначен для сложных операций и в настоящее время функционирует только при «разлочке» накопителей Maxtor, поэтому выходит за рамки данного описания.

Аналогично, выпадающий список внизу.

Рассмотрим контекстное меню данного окна.

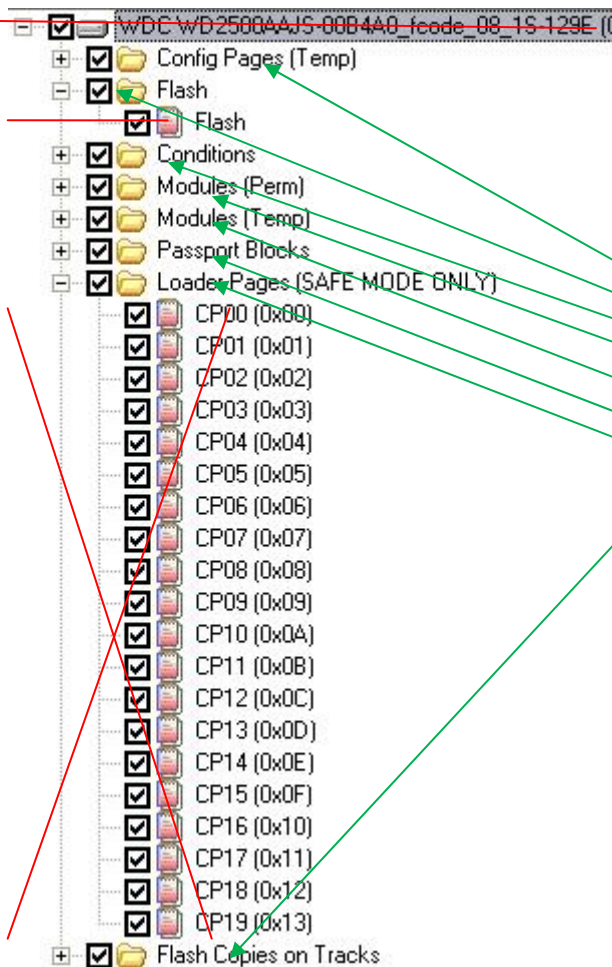


Dump CP – показать дамп страницы (с возможностью её редактирования)

I/O Operations->Read Module – Считать страницу с диска

I/O Operations->Write Module – Записать содержимое страницы в на диск

I/O Operations->Read Branch – для вызова данного пункта меню, следует выбрать корень одной из ветвей дерева



**Допустимые
для
выделения
элементы**

**Красным
зачёркнуты
недопустимые**

Данная ветвь будет считана с накопителя (все элементы, около которых установлены флажки)

I/O Operations->Write Branch – Аналогично, записать содержимое всех отмеченных страниц ветви на диск

I/O Operations->Read Tree – Считать всё дерево

File Operations->Save Archive – Сохранить всё дерево в файл с расширением CRT

File Operations->Export Page to File – Сохранить содержимое выбранной страницы в файл.

File Operations->Load Tree – Считать файл *.crt в память. Текущее дерево будет перестроено в соответствии с деревом, сохранённым в файле

File Operations->Load Data To Current Tree – Считать только данные. Структура дерева не будет изменена.

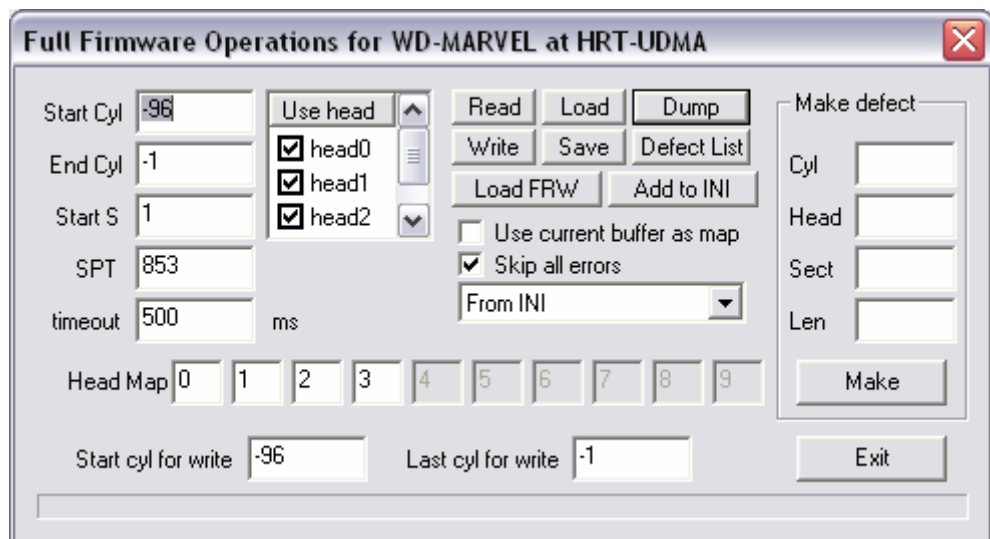
File Operations->Import CPS to branch – Импортировать файл конфигурационных страниц в старом формате на выбранную ветвь дерева (ведь старый формат не мог содержать более одной ветви)

Маленькая хитрость: Двойной щелчок по несчитанной странице вызовет её считывание с диска, а по считанной – откроет её дамп.

Данный диалог является чрезвычайно важным для накопителей Western Digital, так как позволяет загружать в них лоадеры.

Диалог Full Firmware (WD, Samsung)

Для работы с полной служебной областью, выберите пункт меню Service->Full Firmware. при этом, на экране появится следующий диалог:



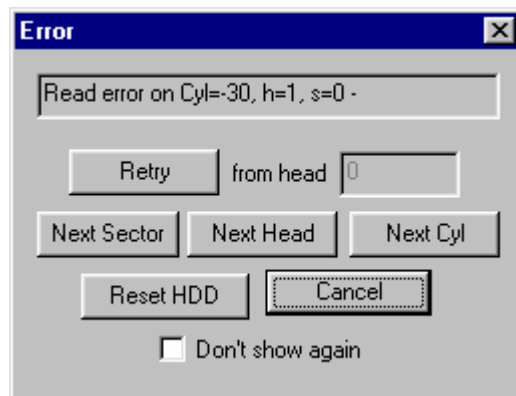
Поля **Start Cyl** и **End Cyl** задают начальный и конечный цилиндры служебной области. Поле **SPT** задаёт число секторов на дорожку (Sector Per Track) для служебной области. При необходимости, Вы можете скорректировать эти значения, но программа подставляет именно те, которые сообщил ей сам накопитель.

Поле **Start S** задаёт номер начального сектора на дорожке. В общем случае, для накопителей WD и Samsung он равен единице. Вы можете изменять его, чтобы получать более компактные файлы ресурсов, если по какой-то причине Вам не нужно сохранять начало дорожки.

Поле **timeout** задаёт время таймаута при чтении. Иногда, накопитель при считывании дефектных секторов, надолго «задумывается». При этом необоснованно тратится время. Задав не очень большое значение таймаута, Вы сократите время «раздумий» накопителя. Разумеется, при этом плохо читаемые сектора будут приняты за ошибки, поэтому не стоит делать его слишком малым.

Список **Use Head** позволяет указать, какие головки будут участвовать в операции. Если Вы по тем или иным причинам не хотите использовать головку, снимите флажок около её имени.

Для считывания служебной информации с накопителя, нажмите кнопку **Read**. При этом индикатор прогресса начнёт нарастать. Если будет обнаружена ошибка чтения, программа выдаст запрос:



В запросе указаны координаты того сектора, на котором произошла ошибка. Вы можете произвести одно из следующих действий:

Повторить попытку чтения. Есть ненулевая вероятность, что при повторе, сектор всё-таки считывается (особенно если значение таймаута велико). Для этого необходимо нажать кнопку **Retry**. Поле **From Head** для данного режима не используется, поэтому оно заблокировано.

Иногда помогает сбросить накопитель и повторить попытку. Для сброса накопителя, нажмите кнопку **Reset HDD**, а после неё – **Retry** (или иную)

Если же повторное чтение не даёт результата, можно пропустить сбойный сектор при помощи кнопки **Next Sector**. Если же Вы видите, что все сектора подряд на дорожке сбойные, можно предположить, что дорожка отформатирована не до конца. В этом случае, Вы можете нажать кнопку **Next Head**. Программа бросит считывание по данной головке и перейдёт к чтению со следующей. Если же видно, что до конца цилиндра ничего осмысленного не будет, то можно нажать кнопку **Next Cyl**. Кнопка **Cancel** же совсем прервёт считывание данных с накопителя.

Также Вы можете немного автоматизировать процесс пропуска. Для этого установите флажок **Don't show again** и нажмите на одну из кнопок **Retry**, **Next Sector**,

Next Head, Next Cylinder. Программа запомнит Ваш ответ и будет имитировать его автоматическую подачу после каждой ошибки. Разумеется, идеальным будет подать ответ **Next Sector**, но чтение при этом будет идти довольно долго. Рекомендуется оставлять чтение служебной области в данном режиме на ночь. Подав **Next Head**, Вы значительно ускорите процесс, но у Вас есть все шансы пропустить важные сектора из-за того, что какой-то один сектор не считался, а это значит – будут пропущены все сектора на головке, расположенные за ним. В целом, рекомендуем Вам работать так, как подсказывают Ваши опыт и интуиция.

Чтобы не мучаться с плохими секторами, установите флажок **Skip All Errors**. В этом случае, диалог сообщения об ошибке просто не будет выдаваться на экран.

Флажок **Use current buffer as map** позволяет сократить время чтения служебной области в десятки, а то и сотни раз. Если Вы заранее знаете что какая-то область у служебной области нечитаемая, Вы можете загрузить либо служебную область с аналогичного накопителя (неважно, что у него другая версия, сейчас важна сама сигнатура «BAD!BAD!BAD!...»), содержащаяся в теле файла), либо сформируйте такой файл при помощи собственной программы. Теперь взведите флажок **Use current buffer as map**. При чтении, все сектора, заполненные сигнатурой «BAD!BAD!BAD!...» будут пропущены, а считаются только те, где такой сигнатуры нет. Так как дефектные сектора читаются дольше всего, время считывания служебной области возрастет многократно.

Вы можете также самостоятельно сформировать записи о «дефектных» цилиндрах. Например, известно, что у большинства накопителей Samsung не отформатирован четвёртый цилиндр (это легко проверяется через диалог I/O Operations). Если у Вас уже есть служебная область, которой Вы можете воспользоваться, как картой – хорошо. А если нет? Неужели придётся ждать, пока накопитель «проползёт» через эту область или высчитывать положение секторов, принадлежащих к четвёртому цилиндру, вручную? Нет, достаточно воспользоваться группой элементов **Make Defect**.

Обратите особое внимание, что группой элементов Make Defect можно пользоваться тогда и только тогда, когда Вы уже правильно заполнили поля, определяющие положение и размер служебной области, ведь положение цилиндров напрямую зависит от значений в этих полях.

Итак. Вводим в группу **Make Defect** значения **cyl=4, head=0, s=1, len=768** (ведь нам надо пометить дорожку 4, головку 0, дорожка начинается с первого сектора и в нашем случае, имеет размер 768 секторов), после чего нажимаем кнопку **Make**. Дефектный участок сформирован. Аналогичным образом, Вы можете метить любые участки, не обязательно совпадающие по положению с дорожкой.

Кнопка **Save** сохранит считанный буфер в файл, а кнопка **Load** – считает данные из файла в буфер. При этом параметры (начальный и конечный цилиндры, SPT и число головок) также сохраняются в файле и при считывании, будут установлены именно они, а не то, что отдал накопитель (считается, что ресурс считан с заведомо исправного накопителя, а вот запись может идти и на частично неисправный, так что больше стоит верить тому, что было на исправном).

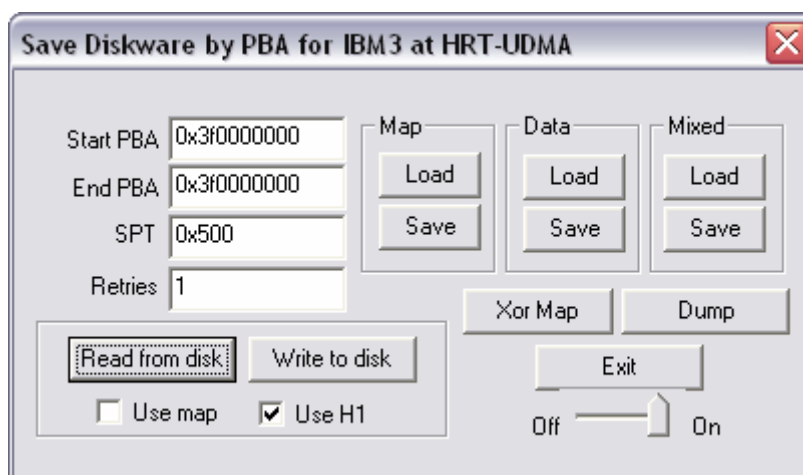
Записать данные на накопитель можно при помощи кнопки **Write**. При этом те сектора, которые не были считаны, записаны тоже не будут, чтобы симитировать формат оригинала.

Кнопка **Dump** позволяет просмотреть дампы буфера. Первый сектор в буфере – служебный. Он содержит заголовок и параметры служебной информации. Данные располагаются, начиная со смещения 0x200.

Кнопка **Add To Ini** добавляет сведения о служебной области в ini файл накопителя, и именно они в дальнейшем будут автоматически подставляться для данного семейства.

Полная служебная область в режиме PBA (IBM)

Для считывания полной служебной области необходимо выбрать пункт меню *Service->R/W Diskware*. На экран будет выдано следующее диалоговое окно:



Параметры «по умолчанию» могут меняться в зависимости от типа накопителя. В данном случае, они соответствуют накопителю AVER.

Параметр **Start PBA** задаёт PBA начала дорожки, с которой будет вестись считывание

Параметр **End PBA** задаёт PBA начальной дорожки, до которой будет вестись считывание.

Параметр **SPT** определяет число секторов, подлежащих считыванию с одной дорожки.

Параметр **Retries** определяет число попыток при ошибках считывания. В принципе, чем больше попыток, тем выше вероятность, что удастся считать сбойный сектор. Но с другой стороны, таких секторов может оказаться два-три, а вот неотформатированных секторов будет гораздо больше. И на каждый из них будут делаться дополнительные попытки чтения. Так для накопителя DTLA с SPT=0x100 и Retries=2, считывание служебной области шло 28 часов. Так что, решайте сами. Можно, конечно, поставить считываться служебную область на выходные, тогда число попыток можно сделать равным трём и даже четырём. В остальных случаях, оптимальным вариантом будет одна попытка.

Кнопка **Read From Disk** начинает процесс считывания данных с диска. При этом на экране появляется индикатор, отражающий процент выполненной части.

Сохранять и загружать данные можно в трёх форматах – непосредственно полная служебная область (расширение файла **IBW**), карта служебной области (расширение файла **IBM**) и смешанные данные - карта и служебка (расширение файла **IBX**). Соответственно, эти группы называются **Data**, **Map** и **Mixed**. В любой группе, кнопка **Save** запишет данные в файл, а кнопка **Load** – загрузит их.

Может возникнуть вопрос: Зачем в программе предусмотрено сразу три типа файлов? Дело в том, что в современных накопителях (например, AVER) нельзя бездумно записывать всю служебную область. Часть секторов необходимо оставить неотформатированными. Для того чтобы это осуществить, необходимо воспользоваться картой форматирования служебной области. Если выбран флажок **Use Map**, то запись будет произведена только в те сектора, которые отмечены в карте, как отформатированные.

С другой стороны, использование карты позволит Вам сэкономить массу времени при чтении служебной области, ведь у сходных моделей и карты похожи. Поэтому достаточно загрузить карту из какого-либо подходящего ресурса, а затем – нажать кнопку **Read From Disk**. Попытки чтения будут вестись только с тех секторов, которые отмечены в карте, как отформатированные.

Отсюда видно, что отдельный файл карты вполне может пригодиться. Например, у Вас есть накопитель. Вы хотите считать с него полную служебную область, но не хотите тратить время на попытки вычитывания неотформатированных секторов. Можно запросить карту у других пользователей программы. В принципе, карта есть и в файлах IBX, но зачем гнать по электронной почте мегабайты, если реальный размер карты – килобайты (которые, к тому же, хорошо архивируются)? Отсюда и рождается необходимость файлов с картами.

Также можно предложить массу других вариантов, когда из отдельно хранящихся карт и служебок, можно делать различные комбинации. Именно поэтому, в программе введены как смешанные файлы (сохранил и забыл), так и отдельные (для тех, кто любит комбинировать).

Кнопка **Dump** позволяет просмотреть и отредактировать содержимое буфера.

Кнопка **Exit** закрывает диалоговое окно.

В заключение, хочется привести несколько типовых алгоритмов работы с данным окном:

Вариант 1: Новый накопитель, аналогов в Вашей коллекции ресурсов нет. *Снять флажок Use Map*, нажать **Read From Disk**, по окончании – нажать **Save** в группе **Mixed**. Смысл данной операции состоит в том, что Вы не пользуетесь существующей картой (её ещё нет), а строите её заново.

Вариант 2: Новый накопитель. Аналоги уже есть в Вашей коллекции (файл *.IBX). Нажать **Load** в группе **Mixed**. Поставить флажок **Use Map**. Нажать **Read From Disk**. Нажать **Save** в группе **Mixed**. Смысл данной операции состоит в том, что Вы загрузили ресурс с картой, затем – считали данные с диска, пользуясь уже имеющейся картой (99% вероятности, что у одинаковых накопителей с разным числом головок карта сходная, за исключением одноголовых моделей). Затем – Вы произвели чтение данных с накопителя. Попыток чтения секторов, помеченных в карте, как отсутствующие, производиться не будет, что резко ускорит процесс чтения.

Вариант 3. Кто-то попросил у Вас карту поверхности имеющегося у Вас ресурса (чтобы не гонять весь многомегабайтный ресурс через сеть). Нажать **Load** в группе **Mixed**, Нажать **Save** в группе **Map**. Полученный файл с расширением IBM очень хорошо запакуется архиватором.

Вариант 4. Вам прислали карту поверхности, чтобы Вы смогли считать ресурс гораздо быстрее за счёт пропуска секторов, которые уже отмечены, как неотформатированные. Нажать кнопку **Load** в группе **Map**. Загрузить файл. Поставить флажок **Use Map**. Нажать **Read From Disk**. По окончании чтения, нажать кнопку **Save** в группе **Mixed**.

Вариант 5. Вы хотите провести исследование (а возможно – правку при помощи внешней программы) ресурса. Нажать кнопку **Load** группы **Mixed**. Нажать кнопку **Save** группы **Data**. полученный файл с расширением IBW представляет собой точную копию области данных накопителя, безо всяких служебных полей.

Вариант 6. Вы хотите записать результаты редактирования информации, полученной в предыдущем пункте, обратно на накопитель. Нажать кнопку **Load** группы **Mixed**. Загрузить «основу». Нажать кнопку **Load** группы **Data**. Загрузить изменённые данные, установить флажок **Use Map**. Нажать **Write To Disk**.

Возможна масса других вариантов. Авторы предоставляют Вам попробовать их самостоятельно.

Файл PATH.INI

По умолчанию, INI файлы берутся из того же каталога, где расположен сам EXE файл. Однако, при работе по сети, бывает удобно пользоваться одними и теми же файлами Fujitsu.INI и HDD.INI. В противном случае, частенько бывает путаница – на одной машине Вы внесли сведения о новом накопителе, а на другой – нет.

Чтобы воспользоваться этим механизмом, в подкаталоге с EXE-файлами, необходимо разместить файл PATH.INI, который будет указывать на подкаталог, откуда брать все прочие инициализационные файлы.

Если такого файла нет, инициализационные файлы будут браться из того же подкаталога, что и исполняемые файлы.

Пример файла path.ini:

```
[PATH]
ini files=c:\HRT\INI\
```

Настройка внешнего вида окна накопителя

Пользователь может настраивать внешний вид окна накопителя под свои привычки. Для этого следует произвести правку секции [WINDOW] файла hdd.ini.

Секция [WINDOW] определяет параметры окна накопителя.

Ключ **CX** задаёт размер окна по горизонтали

Ключ **CY** задаёт размер окна по вертикали

Ключ **textN** задаёт отображение того или иного текста в окне накопителя. Вы можете ввести до ста текстов, которые программа будет отображать в главном окне, дав ключам имена от text0 до text99. Первый же разрыв в нумерации, говорит программе, что необходимо прекратить отображение. Так, если у Вас есть строки с ключами text0, text1, text2, text4 и text5, то будут отображены только строки text0, text1 и text2. Отсутствие строки с ключом text3 прервёт отображение.

Формат ключа следующий:

textN=0xCOLOR,X,Y,TYPE"VALUE

0xCOLOR Цвет надписи в формате BGR (именно так хранятся цвета в WINDOWS)

0x00000000 – чёрный

0x00000080 – красный 50% яркости

0x000000FF – ярко-красный

0x00008000 – зелёный, 50% яркости

0x0000FF00 – ярко-зелёный

0x00800000 – синий, 50% яркости

0x00FF0000 – ярко-синий

0x00808080 – серый

0x00FFFFFF – белый

и т.п.

X координата X надписи (0 – левая граница окна)

Y Координата Y надписи (0 – верхняя строка)

TYPE Может принимать 2 значения

b – BOOL, логическое значение. При наличии условия – отображается, при его отсутствии – не отображается

s – STRING – строка символов

“ Кавычка, отделяющая строку

VALUE Для текстов типа BOOL – условие и отображаемая строка

Для текстов типа STRING – просто отображаемая строка.

Тексты типа BOOL могут иметь вид:

textN=0xCOLOR,X,Y,b" %LBA_MODE%некий текст – отображать некий текст, если в программе выбран режим LBA

textN=0xCOLOR,X,Y,b" %PBA_MODE%некий текст – отображать некий текст, если в программе выбран режим PBA

textN=0xCOLOR,X,Y,b" %WAITING%некий текст – отображать некий текст, если идёт длительная операция (быстрая очистка диска, тестирование и т.п.). В этом режиме, меню в окне накопителя, заблокировано и не может быть вызвано.

textN=0xCOLOR,X,Y,b" %LOCKED% некий текст – отображать некий текст, если при определении накопителя, выяснилось, что у него установлен пароль.

Примеры:

```
text0=0x00008000,20,2,b"%LBA_MODE%LBA MODE
text1=0x00800000,28,2,b"%PBA_MODE%PBA MODE
text2=0x000000f0,0,5,b"%WAITING%Накопитель немного занят
text3=0x000000f0,0,5,b"%LOCKED%Кто-то поставил пароль!!!
```

Тексты типа String имеют произвольную форму. В любом месте такого текста, может быть вставлен один из следующих макросов:

%NAME%	Полное название накопителя
%PORT%	Базовый порт адаптера, с которым идёт работа
%LC%	Число логических цилиндров
%LH%	Число логических головок
%LS%	Число логических секторов
%PC%	Число физических цилиндров
%PH%	Число физических головок
	Число физических секторов отобразить невозможно из-за зонной структуры накопителя
%MAX_LBA%	Размер накопителя в LBA
%MAX_LBA_HEX%	Размер накопителя в LBA, отображённый в 16-ричной системе счисления
%FCODE%	Код семейства накопителя
%INI_SECTION%	Имя секции файла HDD.INI, с которой работает в настоящий момент утилита
%SN%	Серийный номер накопителя

Примеры использования:

```
text0=0x00000000,0,0,s"%NAME% (версия %REV%) сидит на порту %PORT%
text1=0x00000000,0,1,s"По логике: %LC% цилиндров, %LH% головок, %LS% секторов
text2=0x00000000,0,2,s"Заводской номер: %SN%
```

ТЮНИНГ ОКНА СВЕТОДИОДОВ

Секция [LEDS WINDOW] файлы HDD.INI определяет положение и размер окна со светодиодами

Ключ **CX** задаёт размер окна по горизонтали

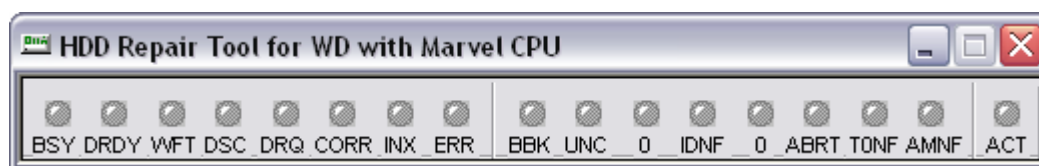
Ключ **CY** задаёт размер окна по вертикали

Ключ **Upper Correct** необходим для учёта ширины панели задач. Он задаёт подъём панели светодиодов относительно нижней границы рабочего стола. Если задать его равным нулю, то светодиоды будут располагаться в самом низу, закрывая собой панель задач WINDOWS.

Ключ **Menu** позволяет скрывать меню на панели светодиодов, что в свою очередь, позволяет уменьшить размер самой панели (при помощи ключа CY). Значение *MENU=0* скрывает меню, *MENU=1* – отображает меню. Следует отметить, что для работы с лицензиями, меню необходимо отобразить.

Ключ **skin** позволяет задавать формат окна светодиодов.

Skin=0 задаёт всем известный вид:



Skin=1 позволяет переключить вид на вертикальный



Другие полезные ключи файла HDD.INI

Секция [POWER MANAGEMENT] определяет режим управления питанием. При работе со специализированным PCI-адаптером она играет второстепенную роль. Она была важнее в те времена, когда каждый делал управление питанием, как мог. В настоящий момент, смысл имеют два ключа:

Ключ **Enabled** определяет текущий режим управления (включён или выключен). Следует отметить, что при работе с дефект-листами, значение этого ключа не учитывается, и управление питанием происходит всегда.

Ключ **Switch off on Select IDE port** определяет логику работы с питанием при появлении диалога настройки порта. Если данный ключ имеет значение 1, то при появлении диалога настройки порта, питание накопителя отключается. Этот ключ можно отнести скорее к анахронизмам, и он оставлен для совместимости с тех времён, когда на панели настройки порта ещё не было удобного переключателя питания.

Секции [IBM], [FUJITSU], [WD] и [Samsung] определяет имена подкаталогов, в которых хранятся ресурсы для соответствующих накопителей.

Ключ **Firmware** задаёт подкаталог «по-умолчанию» для файлов FRW.

Ключ **Full Firmware** задаёт подкаталог «по-умолчанию» для файлов FFW и IBW.

Ключ **Config Pages** задаёт подкаталог «по-умолчанию» для файлов CPS.

Ключ **ROM** задаёт подкаталог «по-умолчанию» для файлов с образами ПЗУ.

Ключ **Resource name format** задаёт формат имени для ресурса. Формат может быть произвольным набором символов, в которые могут быть включены следующие специальные макроподстановки:

%l – полное имя накопителя. Например, для накопителя FUJITSU MPG3204AT E вместо такого макроса подставится **FUJITSU MPG3204AT E**

%s – Короткое имя. Для современных накопителей Fujitsu не имеет смысла, а в целом – подставит имя от конца и до самого правого пробела, поэтому для уже упоминавшегося накопителя FUJITSU MPG3204AT E будет подставлено имя **E**.

%r – Версия микрокода

%f – Код семейства

%n – Заводской номер накопителя. Может быть полезен в случае гарантийного возврата (если Вы будете записывать все служебные области проходящих через Ваши руки накопителей).

%i – Имя секции файла FUJITSU.INI, с которой в настоящий момент идёт работа

Пример формата имени: *MY_RESOURCE_FOR%l_REV_%r*

Или просто %s_FCODE_%f

Для более детального разбиения на подкаталоги - %i\%s_FCODE%f

Секция [GENERAL]

Ключ	Умолчание	Назначение
Show DL on start dialog	1	Если вписать 0, то при старте диалога дефект-листа, он будет чистым. Чтение можно будет инициировать только через меню.
Get DL on type changed	1	Чрезвычайно полезный пункт. Если вписать 0, то при смене типа дефект листа в диалоге, он не перечитывается. А значит, его можно будет сохранить в лист другого вида.
LBA Divider	. (точка)	Разделитель групп знаков при отображении LBA. Если после знака равенства ничего нет, то группы не разделяются.
СХЕМА	1	Для РЮ-карты – ключ совместимости схемы. Значение берётся у разработчика схем. Если не совпадёт с фактическим – схема не загрузится.
Rev	2	
PCB Addr	0x200	Для РЮ карты – номер загружаемой схемы (0, 1, 2, 3 – это адреса 0x000, 0x200, 0x400, 0x600)
Verbal LBA at write test	0	Что писать в сектора при тесте записи. Если 0, то увеличивающуюся 64-битную константу, если 1 – слова «LBA такой-то»
32 Bit Transfer	0	Для РЮ карт. 1 – включает режим 32 битной передачи данных по умолчанию (затем можно настроить в диалоге выбора IDE порта)

Секция [LOCALIZE]

Содержит перевод **некоторых** окон. В настоящий момент переведено не всё. Если функция будет востребована – можно будет добавить. Формат

Фраза, как она на экране=Фраза, как _НАДО_ отображать (например, на русском)

Общие ключи файлов «Заголовок окна накопителя.ini»

Секция [GENERAL]

Ключ	Умолч	Назначение
Auto Save CP	0	Если 1, то при закрытии диалога CP Operations, файл CPS автоматически сохранится. С одной стороны, удобно – ресурсы сами копятяся, с другой – коварно, если была считана всего одна страничка (чисто чтобы глянуть её), то в файл попадёт только она. И вполне может затереть существующий полноценный файл. Поэтому, каждый решает пользоваться этим ключом или нет, в силу специфики СВОЕЙ работы.
CP MAX	100	Максимальное число CP для данного накопителя, подлежащее отображению через старые механизмы, а также в накопителях, где это число невозможно вычислить автоматически.
INI BLA	Нет	Имя файла, содержащего список дополнительных модулей для файла BLA

Именные секции накопителя

Ключ	Умолч	Назначение
CP Decode	ЗаголовокОкна _cp.ini	Имя файла, содержащего декодировку имён модулей для диалога CP Operations (new) данного конкретного накопителя. Позволяет разбить файлы с декодировкой, если они получаются слишком громоздкими...
CP eout	20000	Таймаут в миллисекундах при чтении CP. Это значение будет отображено в диалоге. Пользователь сможет его скорректировать
CP List	Нет	Список CP. Можно использовать запятые и тире (для задания диапазона)
Decode DLL	default_dcd.dll	Имя DLL для интеллектуального декодирования содержимого страниц данного семейства
decode param	0	Параметр, передаваемый в DLL. Удобно, если одна библиотека обслуживает 2-3 похожих семейства, но всё-таки надо передать в неё что-то, уточняющее сведения о семейства. Вот это поле и

		будет передано.
Имя Накопителя	Имя накопитеоля	Позволяет упростить имя накопителя при сохранении ресурса в файл. Особенно актуально у накопителей MAXTOR, где имена довольно длинные
Firmware	нет	Положение системной области в формате: Нач_цил кон_цил SPT (разделено пробелами). В современных накопителях, где можно взять положение из самого накопителя, по возможности, доверяется автоматике. При неуспехе – всё равно идёт обращение к этой формуле.
Fill BLA Heads	0	Используется при формировании псевдо карты модулей на основе ИНИ файла
System Heads	0	Число системных головок. Если физических головок меньше – системных будет столько, сколько физических. Если физических головок больше – будет взято значение данного ключа

Секция [CP Operations]

Ключ	Умолч	Назначение
Not Only Temp and Perm	0	Если 1, то взять дополнительные типы CP из ИНИ файла
Любое иное	Нет	Ключ будет внесён в список типов CP, а значение – это аргумент программы, задающий идентификатор этого типа

Секция [CP LIST] (устар)

Содержит списки CP

Формат: Family Code=список

Используется для уточнения списка, если конкретная модель в семействе имеет список, отличный от описанного внутри именной секции

Пример из файла Fujitsu.ini:

```
[CP LIST]
;MPF AH
00-5C14=1-24,32-34,39-41,45,48-50,112,128-159
01-5C14=1-24,32-34,39-41,45,48-50,112,128-159
```

Секция [CP Comments]

Формат ключа:

CPxx=расшифровка назначения

Пример из файла wd-bb.ini:

```
[CP Comments]  
CP00=Zone Table
```

Секция [LISTS]

Содержит перечень доступных дефект-листов. Заполняется авторами программы. Пользователь может (при желании) отредактировать имена листов или удалить записи, но не может добавить новых. Вернее, добавить может, но программа их поддерживать не станет.

Пример из файла MAXTOR.INI:

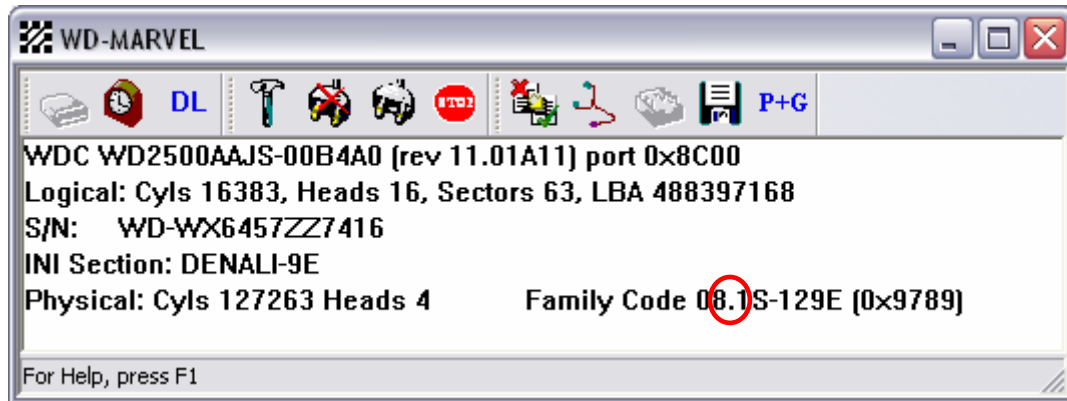
```
[LISTS]  
default=1  
100=Firmware  
05=Reassign
```

Ключи файла WD-MARVEL.INI

Секция MODELS

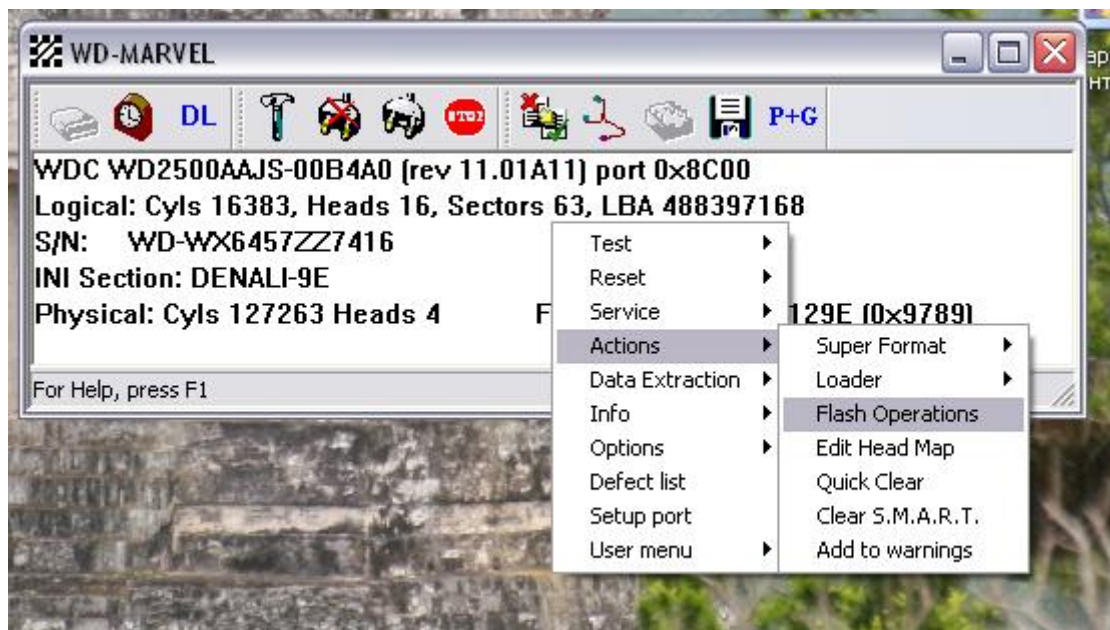
AGE=Section Name

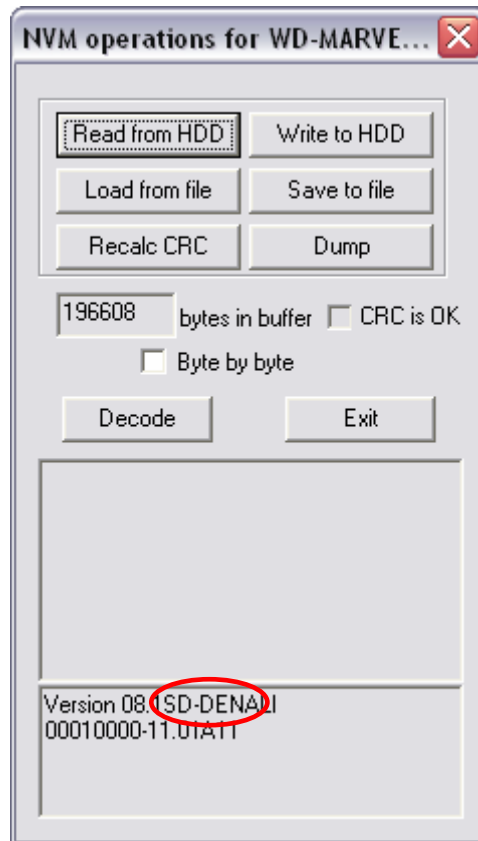
Поле AGE берётся из Family Code. Например, для



Поле AGE=9E.

Имя семейства берётся из декодированного имени ПЗУ:





Значит, семейство называется DENALI.

Получаем строку:

```
[MODELS]
...
9E=DENALI-9E
```

И последующую именованную секцию

[DENALI-9E]

Ключи именных секций

Ключ	У молч	Назначение
SA Mode	U BA1	Способ доступа к служебной области. Выбирается методом перебора, пока не начнут читаться модули служебной области диалога BLA. Для современных накопителей всегда равен UBA2. Для промежуточных равен UBA1. Существуют модели, для некоторых представителей которых надо писать UBA1, для некоторых – UBA2. Закономерность не найдена. Для самых старых следует вписывать CHS.
Zone formula	н ет	Ссылка на формулу декодирования зонного распределения

CP List	0- 254	Список CP по умолчанию.
firmware		Формула чтения полной служебной области. Если не задана, берётся вся зона 0

Пример именной секции:

```
[DENALI-9E]
SA Mode=UBA2
Zone formula=default
CP List=100-255
```

Ключи файла Barracuda.ini

Секция [GENERAL]

Ключ	У МОЛЧ	Назначение
Special Seek Format	s %x,%x	Команда для позиционирования в служебку, если нет иных указаний в именной секции
User Seek Format	s %x,%x	Команда для позиционирования в область данных пользователя, если нет иных указаний в именной секции
Physical Write Format	w ,%x,%x	
Physical Read Format	r, %x,%x	

Секция [MODELS]

Содержит ключи двух видов.

Вид 1: ATA_REVISION=семейство

Вид 2: Строка_идентификации=семейство

Строка идентификации – это строка, возвращаемая накопителем в ответ на команду ^L.

Именные секции накопителя

Ключ	Умолч	Назначение
Write Delay	1000	Задержка для остановки двигателя при записи по физике, начиная с накопителей 7200.8. Подбирается опытным путём. Может достигать даже 10000 или 20000. Для 2.5" моделей не нужна.

		Цель задержки – дождаться, когда накопитель остановит шпиндель, так что величина напрямую зависит от массы пакета дисков.
FMode Read Buffer	0x60	В режиме F> нет команды, которая сообщила бы, с какого сектора идёт буфер чтения. Поэтому его приходится подбирать экспериментально и вписывать в этот ключ.
Write Buffer Signature	WR:	
QR	0	Работает только на Barracuda 7. Если поставить равным 1, то чтение по физике будет частично вестись через IDE порт, что значительно быстрее. К сожалению, это чисто экспериментальный ключ, так как ни до, ни после, он не работает
Special Seek Format	Одноимённый ключ из [GENERAL]	Суть данных ключей проста. Сначала строка для всего семейства BARRACUDA была одна, поэтому хватало секции [GENERAL] (отличавшейся от таковой для накопителей Uxx). Затем пошли отличия. Поэтому теперь строка, характерная для большинства накопителей, вписана в секцию [GENERAL], а если какие-то семейства отличаются от умолчания, для них строки задаются в именных секциях
User Seek Format	Одноимённый ключ из [GENERAL]	
Physica l Write Format	Одноимённый ключ из [GENERAL]	
Physica l Read Format	Одноимённый ключ из [GENERAL]	
New Write Cmd	0	0 до Barracuda 7 включительно. Для остальных вписать 1.
Auto Track Jump Step	1	
Read Flash By Bytes	0x200	Чтение флэш-памяти накопителей Seagate Barracuda производится блоком, это намного быстрее, чем по байту. Однако начало памяти блоком считать невозможно. Этот ключ задаёт размер участка, который следует считывать по байту. На современных накопителях рекомендуется считывать по байту всю флэш-память, иначе возможны проблемы, гораздо большие, чем потеря нескольких минут.
Flash Buffer	0x400	Номер буфера, через который следует считывать флэш-память. Уникально для каждого накопителя. Если не удаётся подобрать – можно считать память по байту, там ничего подбирать не

		требуется, только ждать...
Flash Write Buffer	0x400	Буфер для записи флэш-памяти
Support ATA Terminal	0	Для накопителей Barracuda2-Barracuda4 можно вставить 1, тогда программа даст возможность имитировать COM порт через IDE интерфейс. Программа работает быстрее, но не устойчиво.
System SPT	Берётся из накопителя	Позволяет перекрыть SPT системной области вручную
Firmware	0 0 0	Начальный и конечный цилиндры служебной области, а также её SPT

Ключи файла IBM.INI

Секция [MODELS]

Код_семейства=Семейство

Анализируется столько символов кода семейства, сколько указано в ключе

Именные секции

Ключ	У молч	Назначение
NVM LENGTH	1 00	Длина NVM в байтах (16-ричное число без префикса 0x!!!)
Zone formula	H ет	Имя формулы зонного распределения. Кроме того, возможны следующие предопределённые значения: MULTI
USAG	0 x0ffe00 0	PBA модуля USAG (карты модулей)
Base ABA	0 3FE000 0	Базовый PBA служебной области (16-ричное значение!)
Head Shift	8	Смещение номера головки в PBA служебной области
DiskWare SPT	0 x100	
Head Count offset	0	Положительное значение – смещение в модуле ZONE Отрицательное значение – смещение в NVM
Super G- LIST	0	1 – брать G-LIST Super командой, иначе – декодировать модуль (устаревший вариант)
Get Defect List	S UPER	SUPER – брать P-LIST Super командой DPTA – декодировать модуль «как у DPTA»
FW_DUPLI CATE	H ет	Правила вычисления координат «третьей копии» модуля служебной области
Zero Track Number	H ет	Используется для работы с трековым дефект-листом
SRVM Track Offset	H ет	Используется для работы с трековым дефект-листом
Max Track Defect	H ет	Используется для работы с трековым дефект-листом
PSHT Offset	H ет	Используется при занесении дефектов в P-LIST
Quick Clear Type	2	1 – используется команда SECURITY_ERASE 2 – Используется внутренняя команда IBM
Clear TL	0 x00,0x0	Формула для очистки P-LIST (модуля PSHT)

	0	
wedges	7 0	Число сервометок на дорожку. Используется при детальном анализе сервометок (при неверном значении график не будет точно соответствовать биению диска, хотя амплитуда всё равно будет верна)

Ключи файла IBM2.INI

Секция [MODELS]

Код_семейства=Семейство

Анализируется столько символов кода семейства, сколько указано в ключе

Именные секции

Ключ	Умол ч	Назначение
NVM LENGTH	100	Длина NVM в байтах (16-ричное число!!!)
Zone formula	Нет	Имя формулы зонного распределения. Кроме того, возможны следующие предопределённые значения: MULTI
USAG	0x7FE 20000	PBA модуля USAG (карты модулей)
Base ABA	7FE00 000	Базовый PBA служебной области (16-ричное значение!)
Head Shift	12	Смещение номера головки в PBA служебной области
DiskWare SPT	0x100	
Head Count offset	0	Положительное значение – смещение в модуле ZONE Отрицательное значение – смещение в NVM
Super G- LIST	0	1 – брать G-LIST Super командой, иначе – декодировать модуль (устаревший вариант)
Get Defect List	SUPE R	SUPER – брать P-LIST Super командой DPTA – декодировать модуль «как у DPTA»
FW_DUPLI CATE	Нет	Правила вычисления координат «третьей копии» модуля служебной области
Zero Track Number	Нет	Используется для работы с трековым дефект-листом
SRVM Track Offset	Нет	Используется для работы с трековым дефект-листом
Max Track Defect	Нет	Используется для работы с трековым дефект-листом
PSHT Offset	Нет	Используется при занесении дефектов в P-LIST
Quick Clear Type	2	1 – используется команда SECURITY_ERASE 2 – Используется внутренняя команда IBM
Clear TL	0x00,0 x00	Формула для очистки P-LIST (модуля PSHT)
wedges	70	Число сервометок на дорожку.

		Используется при детальном анализе сервометок (при неверном значении график не будет точно соответствовать биению диска, хотя амплитуда всё равно будет верна)
--	--	--

Ключи файла TOSHIBA.ini**Секция [MODELS]**

Версия микропрограммы=Семейство

Секция [CP LIST]

Ключ	У молч	Назначение
Код семейства	Н ет	Если в пределах одной именной секции имеется несколько версий микропрограммы с разным списком страниц, то их перекрывают в этой секции.

Именные секции

Ключ	У молч	Назначение
Read Mem Formula	4 5 3 6	
CP LIST	Н ет	Если для текущей версии микропрограммы не найдено строки в секции [CP LIST], то список CP берётся из этого ключа (так как именная секция может обслуживать несколько версий микропрограммы)
CPxx	Н ет	Расшифровка назначения CP для отображения
CP Size	0 1	

Ключи файла Maxtor.INI

Именные секции

Ключ	У молч	Назначение
UNLOCK Version	0	0 – «Старые» 1 – GRIZZLY
BLA copies list	0 ,1	
BLA Copies Offset	0 x700	
Forced LBA48	0	Для некоторых накопителей DMP9 необходимо взвести в 1, иначе не работает чтение по физике
POKER	1	Для накопителей с процессором DSP установить в 0
Super Convert	1	
min sys heads	2	Если число физических головок меньше этого значения, число системных головок ВСЁ РАВНО будет присвоено этому значению
P-list formula	H ет	
G-List formula	H ет	
Alt Offset	0 xb1	
Zone formula	H ет	
G_LIST SRC	0 xB1	
ID93 Formula	H ет	

Секция [UNLOCK]

Сигнатура__заголовок_модуля=На_что_заменить|Флаги

Флаги – метод пересчёта KC

Пример:

```
[UNLOCK]
ONP_ILTS=TAP_LD1 |1
ONG_ILTS=TAP_LO0 |0
ONU_ILTS=_UILTS00|0
_UILTS10=_UILTS00|0
OND_CM'S=MDSC 1 |0
TAP_LD1=TAP_LD1 |1
TAP_LO0=TAP_LO0 |0
_UILTS00=_UILTS00|0
```

MDSC 1=MDSC 1 |0

Ключи файла FUJITSU.INI

Секция [MODELS]

Код семейства=Семейство

В коде семейства рассматривается столько символов, сколько вписано в ИНИ файл. Таким образом, удаётся избежать занудного дублирования строк, у которых начало одинаково, а отличается только конец. Однако есть риск, что разные семейства со сходным началом дадут неверную идентификацию. В этом случае, примеряется метод, аналогичный следующему:

```
[MODELS]
...
911=Family_MPG_AT_E
91=Family_MPA
...
0322=Family_MPC_AH
0344=Family_MPD_AT
03=Family_MPE_AT
```

То есть, более детальные строки помещаются выше. При несовпадении, они будут пропущены, а затем – проанализирована более общая строка.

Именные секции

Ключ	У молч	Назначение
Format	0 x66	Команда внутреннего форматирования
Send CP	0 x64	Команда записи CP
ROM Addr	0	Адрес проекции ПЗУ
Old Mode	N o	No – Yes -
HeadM ap	H et	Формат модуля с картой головок. HH – число каналов, hh – число головок. Возможные значения: map hh HH hh HH map HH hh map
Max head	8	Макс. Число головок для данного семейства. Важный параметр при разборе дефект-листа
Max track defect	0 xffff	Число записей под трековые дефекты. Важный параметр при записи дефект-листа
Zone formula	H et	Имя формулы для декодирования зонного распределения. Имеются также предопределённые значения: 16bit

		32bit
Adapti ves	:\	С Путь, где хранятся ресурсы для подбора адаптивов

Расширение функциональности

Одним из достоинств комплекса HRT является возможность расширения его функциональности без внесения изменений в код. Разумеется, сложную функциональность таким образом встроить сложно – ведь нужна будет мощная база для ветвления, обработки ошибок, модификации массивов и прочее, прочее, прочее, а это проще реализовать при написании комплекса. Однако, зачастую пользователю нужны какие-то дополнительные функции, которых, как назло, нет под рукой. Можно вбить соответствующие коды в АТА-диалог, но сначала диалог надо открыть, затем – найти нужную строку в окне помощи... А если команда требует прокачки каких-то данных, то ещё надо найти и загрузить необходимый двоичный файл. В общем, оно, конечно, удобно, но не всегда.

Поэтому в комплекс HRT был введён дополнительный механизм расширения функциональности – внедрение пунктов в меню.

Для всех накопителей, Вы можете править файл `usermenu.ini`, для накопителей Samsung, необходимо править файл `Samsung.ini`. Отличие состоит именно в том, что пункт из `usermenu.ini` будут отображаться во всех утилитах HRT, а пункты из `Samsung.ini` – только в утилитах для накопителей Samsung.

В файле `samsung.ini` (или `usermenu.ini`) можно ввести секцию

[Additional menu]

Строки в этой секции будут определять дополнительные пункты меню. Имя ключа попадёт в качестве пункта меню, а значение определяет секцию, обрабатывающую данный ключ. Например, если мы хотим ввести в команду, очищающую Master Boot Record накопителя, данная секция примет вид:

```
[Additional menu]
Clear MBR=Clear Master Boot Record
```

В примере, мы умышленно ввели разные обозначения для имени пункта меню и имени обрабатывающей секции, чтобы не вносить путаницу. В жизни, это правило не обязательно, Вы можете вводить единые обозначения. Всего Вы можете добавить до ста новых пунктов меню.

Для обработки нашей функции, необходимо создать секцию

[Clear Master Boot Record]

В данной секции, возможны следующие ключи:

Main Menu Item – номер группы главного меню, в которое мы хотим внедрить наш пункт. Возможны следующие значения:

№	Группа
0	Test
1	Reset
2	Service
3	Actions
4	Info
-1 (именно минус один)	Внизу в меню появится группа User Menu, данный пункт добавится в неё.

Значение по умолчанию, равно двум.

Previous Command – Используется для функций, которые выполняются путём подачи ряда команд. Причина, по которой цепочка подаётся не с начала, а с конца – проста. Очень часто, команды требуют перед собой команду перехода в технологический режим (SuperOn). И для десяти цепочек придётся десять раз описывать команду SuperOn, что приведёт к необоснованному росту ini-файла. При текущей организации же, мы можем просто указать для всех команд, что ПЕРЕД ними должна быть подана команда SuperOn, а значит – описать её всего один раз. Длина цепочки – не ограничена (вернее, она ограничена самой ОС WINDOWS, которая некорректно работает с ini-файлами, длиной, свыше 64К). Если команда не требует перед своим выполнением какой-либо иной команды, то данный ключ включать в секцию не следует.

taskfile – последовательность регистров 0x171-0x176, которые следует послать в накопитель. Формат задан жёстко, каждый байт должен быть обрaмлён префиксом «0x» и все байты должны быть разделены запятыми.

Need Data – определяет необходимость прокачки данных после выполнения операции. Если значение этого ключа задать равным единице, программа будет прокачивать буфер с данными, если же нулю – прокачки не произойдёт.

Save or Load – имеет смысл только при Need Data=1. Определяет направление прокачки данных ПО ОТНОШЕНИЮ К ФАЙЛУ с данными. Если задать значение ключа «S» - данные будут приняты из накопителя и сохранены в файл. Если же «L» - данные будут считаны из файла и прокачаны в накопитель.

Кроме того, Вы можете запустить внешний редактор для коррекции полученных данных. Для этого ключ Save Or Load должен принять значение «E». В качестве имени файла, необходимо указать имя EXE-файла. Утилита будет ждать, пока вызванная программа не завершит свою работу, то есть Вы можете организовать режим работы «Считали, сохранив результаты в файл, запустили внешний редактор, по окончании редактирования – считали результаты из файла и переслали обратно в накопитель».

Ask for file name – если Вы должны прокачать файл с жёстко заданным именем, задайте значение этого ключа, равным 0. Если же задать значение, равным единице, система выдаст диалог запроса файла с данными, подлежащими прокачке, либо запуску.

File Name Имя файла с данными, подлежащего прокачке, либо имя исполняемого файла. Формат следующий:

маска_имени|ключ_подкаталога

Маска имени задаётся по тем же правилам, что и строка **resource file name**, описанная ранее. Она также может содержать всевозможные символы формата (%l, %s, %g и т.д.).

Ключ подкаталога также относится к описанной ранее секции, задающей подкаталоги. Вы можете ввести в эту секцию любой ключ, после чего использовать его, как базовый. Например, Вы ввели в файл hdd.ini следующую секцию:

```
[Universal]
Resource name format=%s_fcode_%f
Firmware=d:\arc\hdd_res\UNI_frw\bla\
Full Firmware=d:\arc\hdd_res\UNI_frw\ibx\
NVM=d:\arc\hdd_res\UNI_frw\
Failopomojka=c:\my_files\4hrt\
```

Пусть дополнительное меню ссылается на секцию [my_doing], а там есть строка

File Name=saved_for_%s.bin|failopomojka

При этом Вы выбрали накопитель, как Universal и в данный момент к комплексу подключён накопитель Quantum Fireball ST 2.1

В этом случае, имя файла, с которым будет работать программа, получится следующее:

c:\my_files\4hrt\saved_for_Quantum_Fireball_ST_2_1.bin

Используя собственные ключи подкаталога и маски для задания имени, Вы можете гибко настраивать работу дополнительных меню под свои нужды.

Вернёмся теперь к нашему примеру. Итак, мы делаем функцию очистки MBR. Сформируем файл, длиной 512 байт и состоящий из одних нулей (при помощи любых подручных средств) и положим его в какой-либо подкаталог. Например, в c:\hrt\additional_menu_info\ и назовём CLEAR_MBR.BIN.

В файле **hdd.ini** добавим в секцию [universal] ссылку на этот подкаталог, например,

menu info= c:\hrt\additional_menu_info\

В файле **universal.ini** у нас уже создана секция

[Clear Master Boot Record]

Начинаем её заполнять.

```
; Внедряем в меню Actions
Main Menu Item=3
; COUNT=1, LBA=0, команда = 0x31
```

```
taskfile=0x00,0x01,0x00,0x00,0x00,0xE0,0x31
; После подачи требуется прокачка данных
Need Data=1
; Данные читаются из файла
Save or Load=L
; Имя файла задано жёстко, поэтому запрашивать не надо
Ask for file name=0
; Для запроса - маска по умолчанию, для жёстких файлов - имя файла.
File Name= CLEAR_MBR.BIN|menu info
```

Аналогично, Вы можете создавать любые другие пункты меню. Для длинных цепочек, после любой ошибки накопителя, выполнение будет прервано на той команде, которая вызвала эту ошибку.

Разработка модулей декодирования

Как уже отмечалось в описании диалогов для работы с сокращённой служебной областью, Вы можете декодировать модули служебной области и конфигурационные страницы, в соответствии со своим усмотрением. Для этого, необходимо разработать свой файл DLL, который будет производить декодирование. Язык программирования, на котором разрабатывается файл, не имеет значения.

Чтобы DLL мог декодировать модули служебной информации, он должен содержать функцию:

```
__declspec(dllexport) int DecodeModule(BYTE* pData, int size, DWORD param, char* result)
```

pData – указатель на тело модуля, подлежащего декодированию

size – размер модуля в байтах

param – произвольный параметр, позволяющий определить семейство (берётся из ключа **Decode Param** именной секции ini-файла накопителя)

result – указатель на буфер, в который функция должна поместить текст расшифровки тела модуля. Текст должен оканчиваться нулевым байтом (ASCIIZ-строка).

Для декодирования тел конфигурационных страниц, DLL должен содержать функцию:

```
__declspec(dllexport) int DecodeCP(int nCP, BYTE* pData, int size, DWORD param, char* result)
```

nCP – номер конфигурационной страницы

pData – указатель на тело конфигурационной страницы, подлежащей декодированию

size – размер данных в байтах

param – произвольный параметр, позволяющий определить семейство (берётся из ключа **Decode Param** именной секции ini-файла накопителя)

result – указатель на буфер, в который функция должна поместить текст расшифровки тела конфигурационной страницы. Текст должен оканчиваться нулевым байтом (ASCIIZ-строка).

Обе функции в настоящий момент должны возвращать значение 0.

В качестве примера, к комплексу прилагается демонстрационный проект, в котором производится минимальная расшифровка модулей и конфигурационных страниц накопителя WD (так как только он содержит и то и другое одновременно). Вы можете во-первых, расширить функциональность этого проекта, а во-вторых, по аналогии разработать DLL для других накопителей.

Оглавление

ВВЕДЕНИЕ	2
ПРЕДОСТЕРЕЖЕНИЯ	2
ГЛАВНОЕ ОКНО ПРОГРАММЫ	3
ЛИЦЕНЗИРОВАНИЕ	5
ТИПЫ ТРАНСЛЯЦИИ.....	7
НАСТРОЙКА ПОРТА	9
НАСТРОЙКА СОМ ПОРТА.....	12
ТЕРМИНАЛ.....	15
ОКНО НАКОПИТЕЛЯ.....	17
ТЕСТИРОВАНИЕ НАКОПИТЕЛЯ	18
ТЕСТ БУФЕРА.....	23
ОПЕРАЦИИ ВВОДА-ВЫВОДА.....	24
Чтение и запись по логическим координатам	24
Чтение и запись по физическим координатам	26
РАБОТА С ПАМЯТЬЮ.....	28
Общие сведения	28
Работа с ОЗУ/ПЗУ	28
Работа с энергонезависимой памятью	30
Работа с буферной памятью накопителя	30
РУЧНАЯ ПОДАЧА КОМАНД.....	31
ИНФОРМАЦИЯ О НАКОПИТЕЛЕ	33
SMART	36
Общие сведения	36
Работа с атрибутами	37
Тесты самодиагностики SMART	38
Просмотр логов SMART	39
РАБОТА СО СЛУЖЕБНОЙ ОБЛАСТЬЮ	41
Общие сведения	41
Работа с сокращённой служебной областью (WD, IBM).....	42
Работа с конфигурационными страницами (Fujitsu, WD).....	45
Новый внешний вид работы с конфигурационными страницами	51

Диалог Full Firmware (WD, Samsung)	55
ПОЛНАЯ СЛУЖЕБНАЯ ОБЛАСТЬ В РЕЖИМЕ РВА (IBM).....	59
Файл PATH.INI	61
НАСТРОЙКА ВНЕШНЕГО ВИДА ОКНА НАКОПИТЕЛЯ	62
ТЮНИНГ ОКНА СВЕТОДИОДОВ.....	64
ДРУГИЕ ПОЛЕЗНЫЕ КЛЮЧИ ФАЙЛА HDD.INI.....	65
Секция [GENERAL].....	66
Секция [LOCALIZE].....	66
ОБЩИЕ КЛЮЧИ ФАЙЛОВ «ЗАГОЛОВОК ОКНА НАКОПИТЕЛЯ.INI».	67
Секция [GENERAL].....	67
Именные секции накопителя	67
Секция [CP Operations].....	68
Секция [CP LIST] (устар).....	68
Секция [CP Comments].....	69
Секция [LISTS].....	69
КЛЮЧИ ФАЙЛА WD-MARVEL.INI.....	70
Секция MODELS.....	70
Ключи именных секций	71
КЛЮЧИ ФАЙЛА BARRACUDA.INI.....	72
Секция [GENERAL].....	72
Секция [MODELS]	72
Именные секции накопителя	72
КЛЮЧИ ФАЙЛА IBM.INI.....	75
Секция [MODELS]	75
Именные секции.....	75
КЛЮЧИ ФАЙЛА IBM2.INI.....	77
Секция [MODELS]	77
Именные секции.....	77
КЛЮЧИ ФАЙЛА TOSHIBA.INI.....	79
Секция [MODELS]	79
Секция [CP LIST]	79
Именные секции.....	79

КЛЮЧИ ФАЙЛА MAHTOR.INI.....	80
Именные секции.....	80
Секция [UNLOCK].....	80
КЛЮЧИ ФАЙЛА FUJITSU.INI	82
Секция [MODELS].....	82
Именные секции.....	82
РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ.....	84
РАЗРАБОТКА МОДУЛЕЙ ДЕКОДИРОВАНИЯ.....	88
ОГЛАВЛЕНИЕ	89

