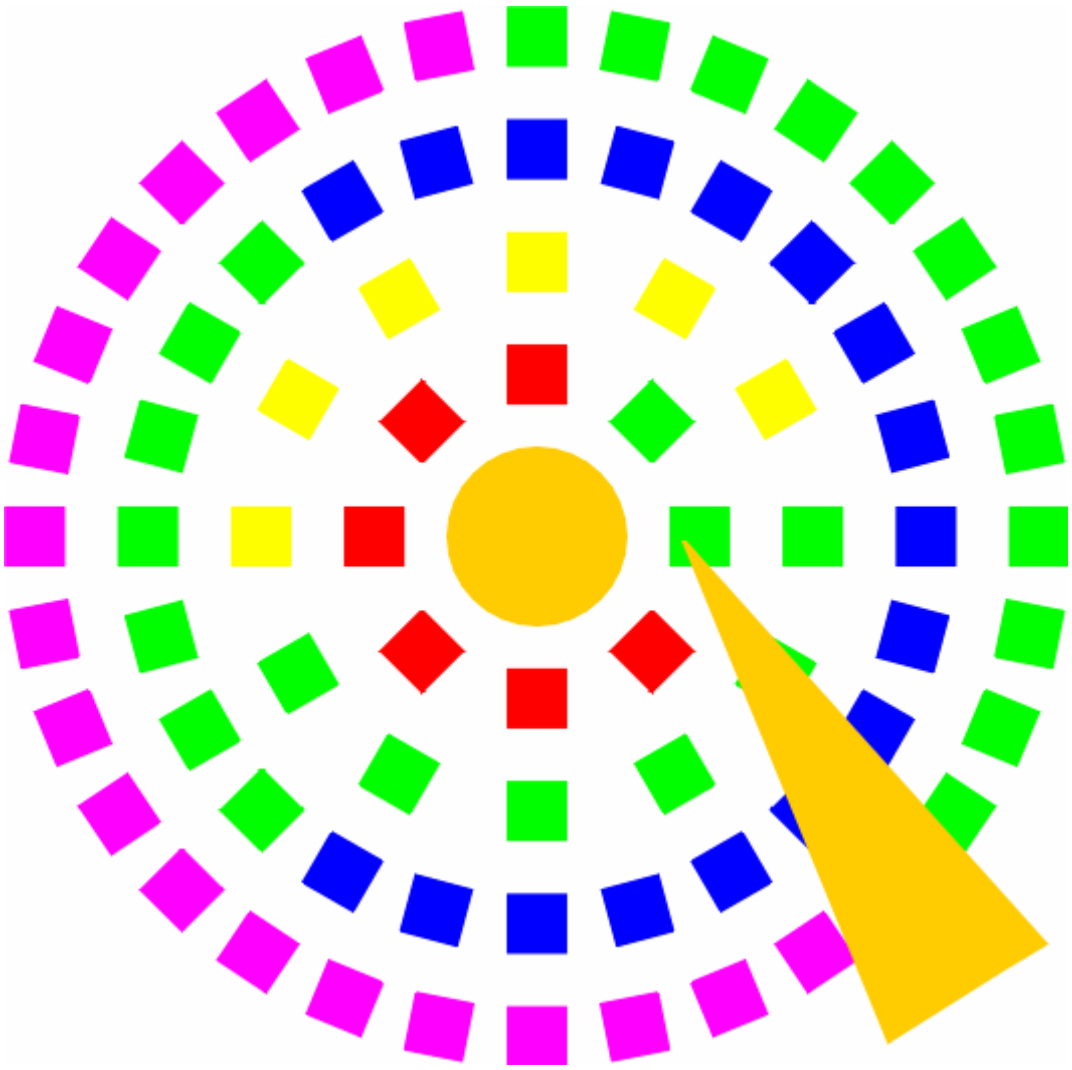


HDD REPAIR TOOL
DATA RECOVERY
EDITION
(HRT DRE)

КОМПЛЕКС HRT
УТИЛИТА ВОССТАНОВЛЕНИЯ ДАННЫХ

http://www.bvg-group.ru

Введение

Утилита HRT DRE предназначена для восстановления данных с неисправных накопителей. Все свои возможности она развивает в паре со специализированными утилитами комплекса HRT, но возможна работа с ней и без внешних утилит, просто часть возможностей при этом не будет реализована.

ОСНОВНОЕ ПРАВИЛО

Первое правило при восстановлении информации с неисправного накопителя: НАКОПИТЕЛЬ В ЛЮБОЙ МОМЕНТ МОЖЕТ ОКОНЧАТЕЛЬНО ВЫЙТИ ИЗ СТРОЯ, ПОЭТОМУ ЧИСЛО ОБРАЩЕНИЙ К НЕИСПРАВНОМУ НАКОПИТЕЛЮ ДОЛЖНО БЫТЬ МИНИМАЛЬНЫМ. Именно из этого правила выводится большинство концепций, заложенных в утилиту.

Основные принципы восстановления информации

Данные следует восстанавливать уже с копии

Раз число обращений к неисправному накопителю должно быть минимальным, нельзя производить глобальный анализ файловой структуры на нём самом. Необходимо скопировать данные на исправный накопитель. В качестве приёмника информации может выступать как физический накопитель равной или большей ёмкости, так и файл-образ. Следует, однако, помнить, что файловая система FAT не позволяет полноценно работать с файлами более 2Г (а более 4Г – не только полноценно, но и неполноценно), поэтому файлы-образы лучше хранить на разделе с файловой системой NTFS.

Файл-образ в дальнейшем может быть скопирован на другой накопитель, либо проанализирован внешними программами, рассчитанными на работу с файлами-образами. Кроме того, файл-образ можно подключить в систему, как виртуальный диск и различные внешне утилиты восстановления данных (например, R-Studio, да и проводник ОС Windows) не заметят, что физически работают с файлом. Они будут уверены, что работа ведётся с физическим накопителем.

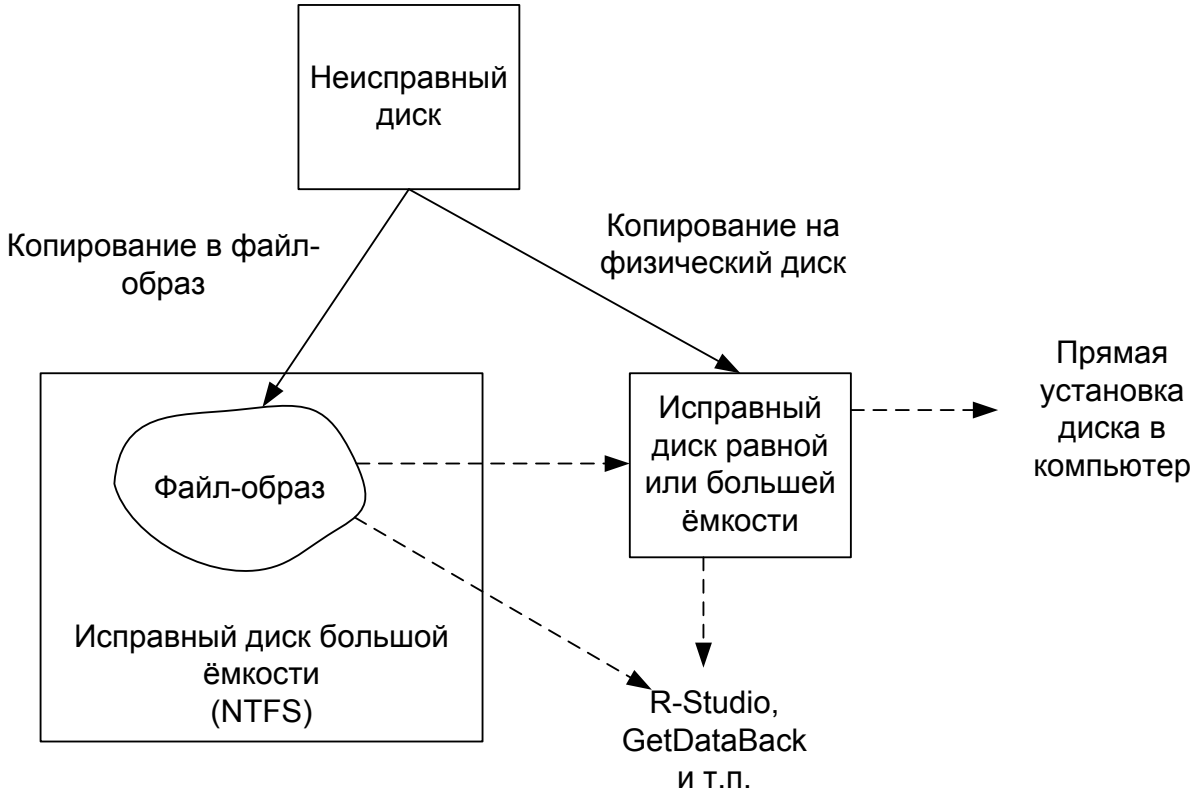


Рис. 1. Варианты копирования данных и их дальнейшего использования

Пояснение методики пропуска блоков

Посекторное копирование некоторых накопителей может вестись достаточно долго (сутками и даже неделями). Чтобы ускорить процесс, необходимо применять различные ухищрения. Самое простое из них – если встретилось некоторое небольшое количество BAD-секторов подряд, предположить, что идёт пачка BAD-блоков и сделать пропуск большого количества секторов. При этом процесс ускоряется очень и очень значительно, а потери обычно – минимальны. Этот механизм иллюстрируют Рис. 2 и Рис. 3.

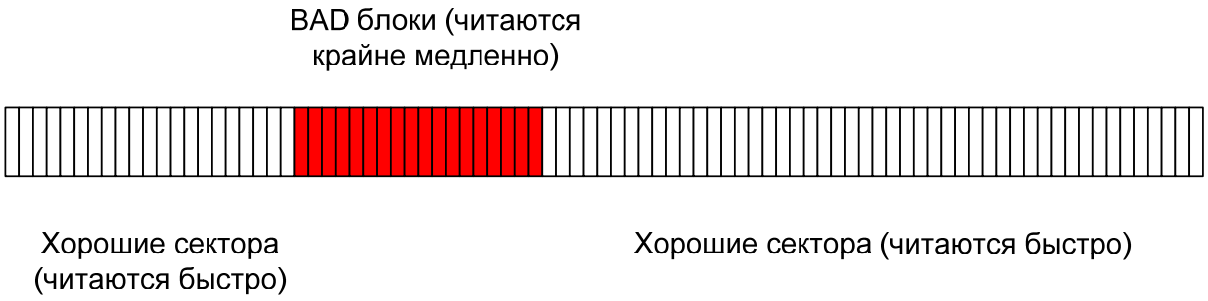


Рис. 2. Исходное состояние диска

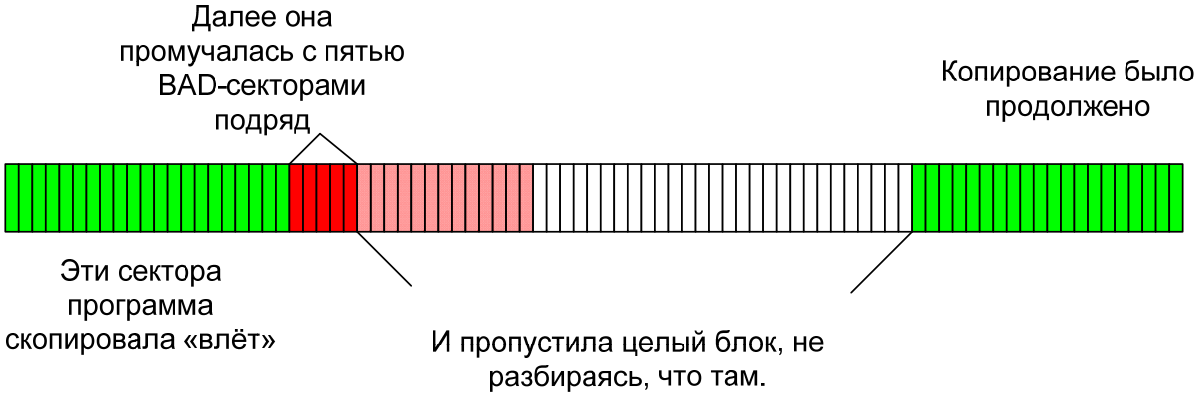


Рис. 3. Принцип пропуска блока

Как видно, в пропущенный блок попали и хорошие сектора. Однако мы помним, что главная задача – скопировать диск как можно быстрее. Пока головки будут пытаться вычитать BAD-область, диск может выйти из строя (в том числе, и потому, что головки «скребут» по царапине). А вычитывание пропущенных участков можно будет произвести уже при втором проходе, когда максимум данных уже спасён и риска потерять всё, мучаясь с малым участком уже нет (так как всё находится в копии).

Восстановление данных при помощи файлового разбора

А что делать, если простое копирование идёт крайне медленно (диск сильно разрушен)? Оно может длиться неделями или даже месяцами. Если необходимо восстановить только небольшое количество заранее известных файлов, можно воспользоваться файловым разбором. Если пути к файлам не утеряны, есть шанс или восстановить файлы напрямую, или отметить пространство, занятое ими к копированию, а всё остальное – не копировать (то есть, не тратить на него время). Однако ещё раз повторим, что файловый разбор – это именно средство для того, чтобы отфильтровать ненужное для копирования. Если копирование и так идёт хорошо, лучше к файловому разбору не прибегать. Хотя бы потому, что на неисправном накопителе файловая система тоже могла быть разрушена.

Копирование по головкам

Любой современный накопитель работает в трансляции LBA. То есть, для обращения к сектору, необходимо задать его порядковый номер. Тем не менее, внутри накопителя по-прежнему используются номера цилиндров, головок и секторов (PCHS). И об этой физической структуре следует помнить.

Простейший случай, когда эта информация важна – «осыпающийся» накопитель. Допустим, у него плохо читается одна поверхность, а остальные – хорошо. И следует сначала скопировать все хорошие поверхности, пока пыль от «умирающей» не загрязнила их. Например, у диска, приведённого на Рис. 4, следует сначала скопировать сектора, расположенные на поверхностях 0, 1, 3-5, так как они считаются быстро, а уже затем пытаться вычитать то, что читается с поверхности 1.

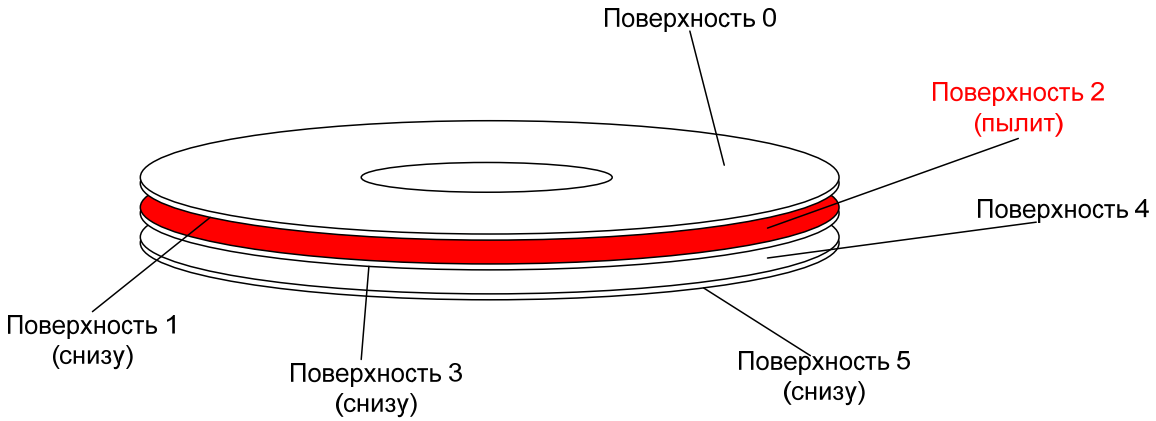


Рис. 4. Пример диска с "пылящей" поверхностью

Однако, как это сделать? Очень просто. Опять же, необходимо провести два прохода. Для первого прохода отметить к исполнению все сектора, относящиеся к хорошим головкам (благо накопители позволяют преобразовывать координаты LBA в PCHS, а утилиты HRT, пользуясь командами преобразования, могут построить карту), а для второго – к плохим.

Второй случай копирования по головкам более сложен. Дело в том, что сервометки записываются на накопитель «родным» блоком головок. Если он вышел из строя и был заменён, добиться качества его позиционирования, какое было у «родного» практически невозможно. Экспериментально выяснено, что если головка «зацепилась» за сервометки, то дальше она будет худо-бедно читать данные. А процесс переключения головок приведёт к лихорадочным поискам сервометок на другой поверхности. Чтобы сократить время копирования, необходимо и достаточно сначала считать все данные с головки 0, затем – все данные с головки 1 и т.п.

Здесь же следует остановиться ещё на одном случае вычитывания по головкам. Допустим, у накопителя 6 головок (от 0 до 5). И по какой-то причине у него вышли из строя головки 2 и 4. В наличии имеются совместимые блоки головок с живой головкой 2, но мёртвой 4 и с живой головкой 4, но мёртвой 2. В этом случае, на первом проходе копируем головки 0, 1, 3, 5 «родным» блоком головок, затем ставим блок головок с живой головкой 2 и копируем её, затем – снова меняем блок головок и вычитываем по головке 4.

Суть карты копирования

Из всего вышесказанного следует, что во время копирования могут остаться какие-либо блоки, которые было бы хорошо обработать в будущем (так как они были пропущены и не ясно, относятся они к BAD блокам или нет), а также такие блоки, которые мы бы хотели и не хотели обработать (например, при файловом разборе или при копировании по головкам). Для того чтобы учитывать всю эту массу информации, в программе применена карта, пример которой приведён на Рис. 5.

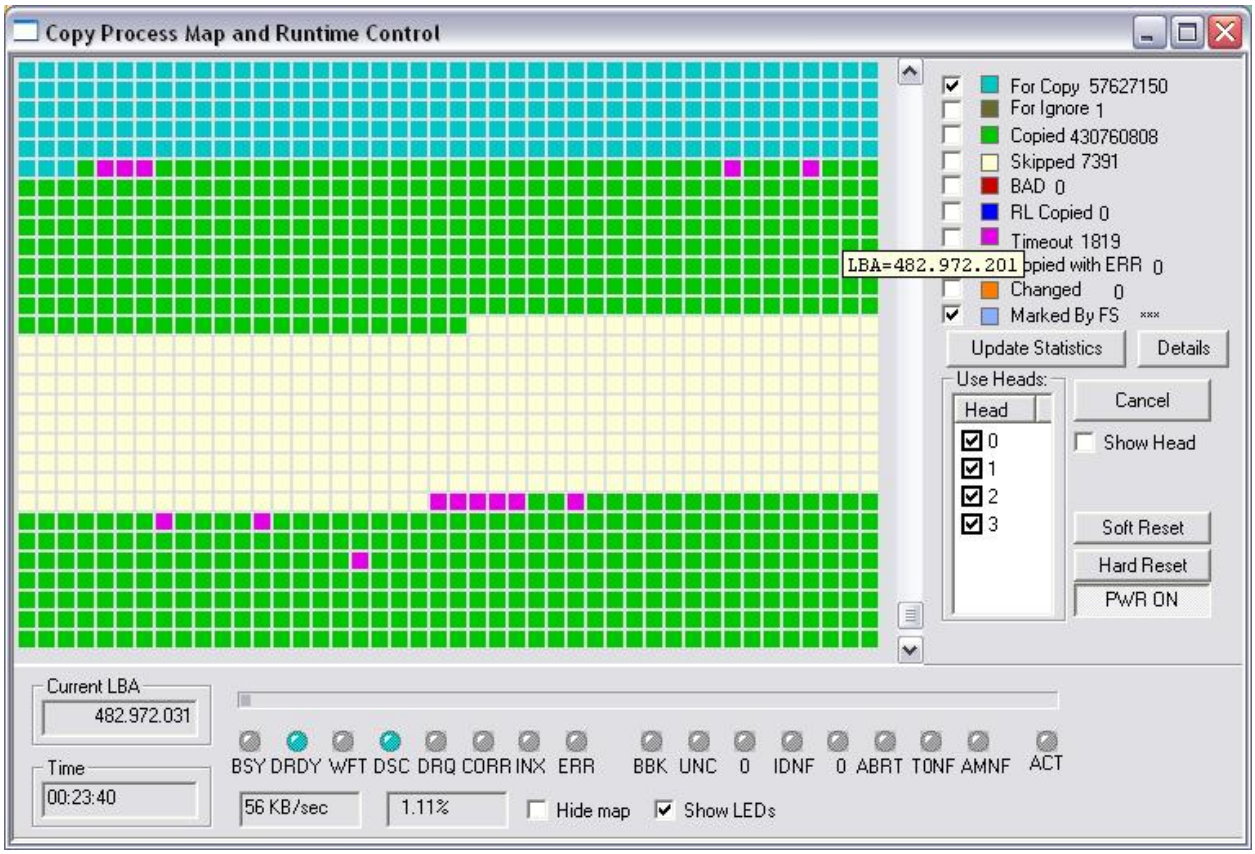


Рис. 5. Пример карты

Каждому сектору неисправного диска соответствует один элемент карты, который отмечает состояние сектора. Изначально, все сектора имеют метку «С данным сектором не следует выполнять никаких операций». Поэтому, если просто выбрать накопитель и начать процесс его копирования, то вскоре Вы получите радостное сообщение, что процесс завершён, но результатов не будет никаких. И это понятно. Чтобы минимизировать число обращений к источнику, карта пуста. Вы должны самостоятельно внести в неё пометки, которые заставят утилиту копировать те или иные сектора. В простейшем случае – пометьте всю карту и процесс пойдёт. Более сложные методы пометки карты будут описаны ниже.

По мере прохождения процесса, сектора из состояния «Должен быть скопирован» будут переходить либо в состояние «Скопирован», либо в то или иное проблемное состояние. Через равные промежутки времени, карта будет автоматически сохраняться. То есть, если система «зависнет», либо отключат электроэнергию или случится иная непредвиденная ситуация, пропадут только изменения состояния, произошедшие с

момента предыдущего автосохранения. Все остальные сведения будут сохранены, и процесс начнётся не с самого начала.

В целом, процесс можно прерывать и начинать многократно, ведь он базируется на карте, а карта – отражает текущее состояние. Надо прервать копирование для срочного ремонта – прекрасно, прервали, затем – загрузили и начали процесс с точки останова. Выделили пофайлово недостаточный участок и необходимо докопировать другие – никаких проблем, отмечайте их и копируйте. Скопировали всё, но есть подозрения, что некоторые пропущенные (после множественных BAD-блоков) участки содержат полезную информацию – тоже всё просто, перемаркировали «Пропущенное» в «Следует скопировать» и провели процесс. Всё, что уже скопировано – не будет затронуто. То есть, не стоит бояться, что какой-либо дополнительный процесс займёт очень много времени за счёт повторных обращений. Нет, никаких повторов не будет. Время будет затрачено только для новой операции.

При отметке файлов, карта также будет самостоятельно «зеленеть». Это связано с тем, что для файлового разбора необходимо считывать ту или иную служебную информацию. А раз она всё равно считывается, то производится её автоматическая запись на приёмник. Если в дальнейшем она понадобится (при втором проходе, при иных обстоятельствах), все «зелёные» сектора будут считаны не с источника, а с приёмника.

Информация из BAD-блоков

А как быть с BAD-блоками? Что делать с их содержимым? Можно заполнять той или иной сигнатурой (типичные варианты – бесконечная строка BAD!BAD!BAD!, удобная для анализа или нейтральная константа 00000000, прелесть которой заключается в том, что если какое-то ПО берёт из такого сектора указатель, оно не уйдёт в неведомые дали). Это хорошо для тех накопителей, которые вместо ошибочных секторов возвращают нечто странное. С другой стороны, некоторые накопители (например, IBM) возвращают вполне осмысленные данные. Для таких накопителей лучше игнорировать флаг ошибки и записать данные из буфера на приёмник. Вполне возможно, что впоследствии из этих данных удастся сложить пусть повреждённый, но приемлемый файл. И, наконец, можно попытаться вычитать данные статистическим методом, то есть, считать их несколько раз и положить на каждое место то значение, которое лежало на этом месте наиболее частое количество раз. Этот способ хорош, если накопитель отдаёт данные неустойчиво

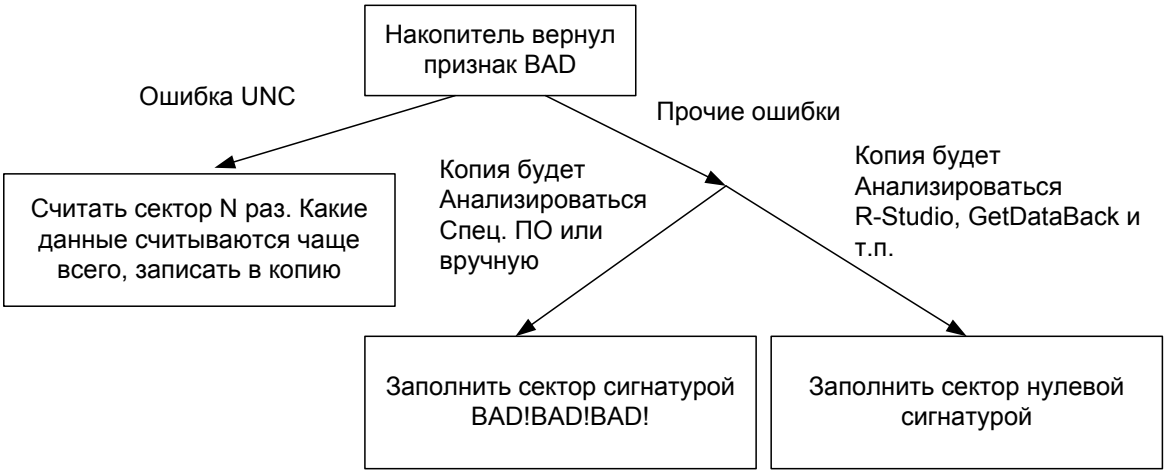


Рис. 6. Рекомендуемое заполнение секторов копии, соответствующих BAD-блокам

Выводы

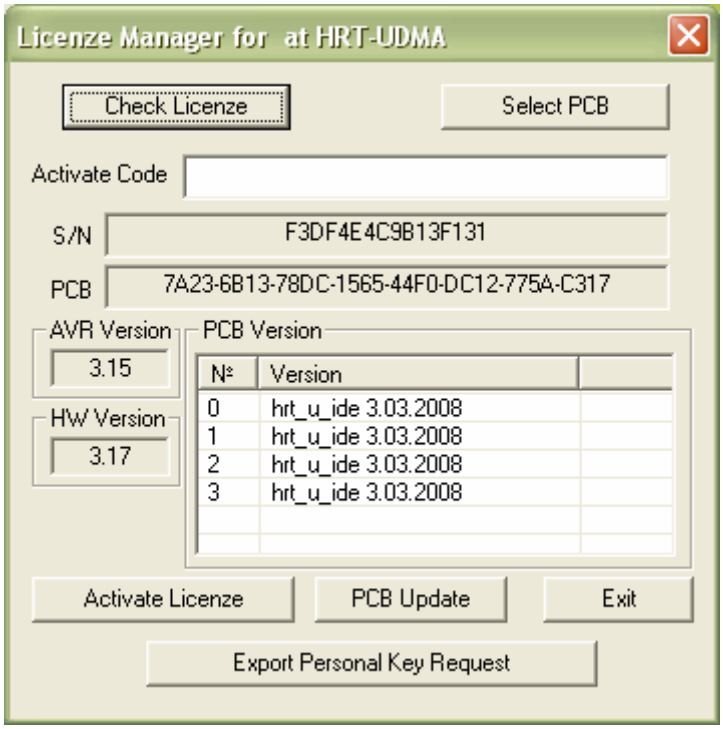
.
Итак. Из всего вышесказанного, можно сделать выводы:

1) Данные следует копировать на исправный носитель и дальше работать уже с образом данных
2) Образ может располагаться как на физическом накопителе, так и в файле (но файл должен быть расположен на NTFS-томе)
3) Если встретилось много BAD-блоков подряд, лучше пропустить большой участок и попытать счастья за ним
4) Копировать лучше только заведомо нужные участки
5) Даже из BAD-блоков можно попытаться вычитать данные

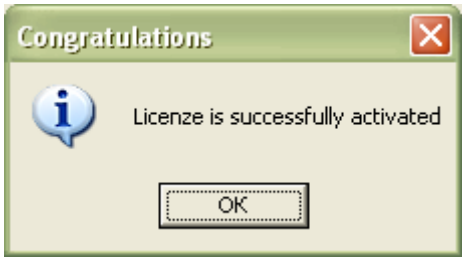
Лицензирование

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

после нажатия клавиши OK, 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 Licenze**. 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 Licenze,** 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.

Понятие проекта

Для работы утилиты, используется большое количество разнообразных настроек. При выходе из утилиты, настройки сохраняются в INI файлах. При повторном запуске – они загружаются вновь. Однако, это чревато. Допустим, Вы копировали данные для одного Заказчика, но тут пришёл другой со срочным заказом. Вы тут же прекратили копирование прошлого заказа и начали работу с новым. А там накопитель с другой неисправностью, поэтому пришлось изменить настройки. И что? Неужели при возврате к прежнему заказу придётся снова всё перенастраивать? Нет! Дело в том, что утилита для каждого заказа организует свой проект.

Проект хранится в отдельном каталоге, и в него копируются все INI-файлы. При открытии проекта, будут загружены не глобальные, а локальные INI Файлы. Поэтому настройки будут восстановлены именно те, которые соответствовали проекту. В том же каталоге хранится карта копирования. А если копирование ведётся в образ, то в том же каталоге хранится файл 0.BIN – образ диска.

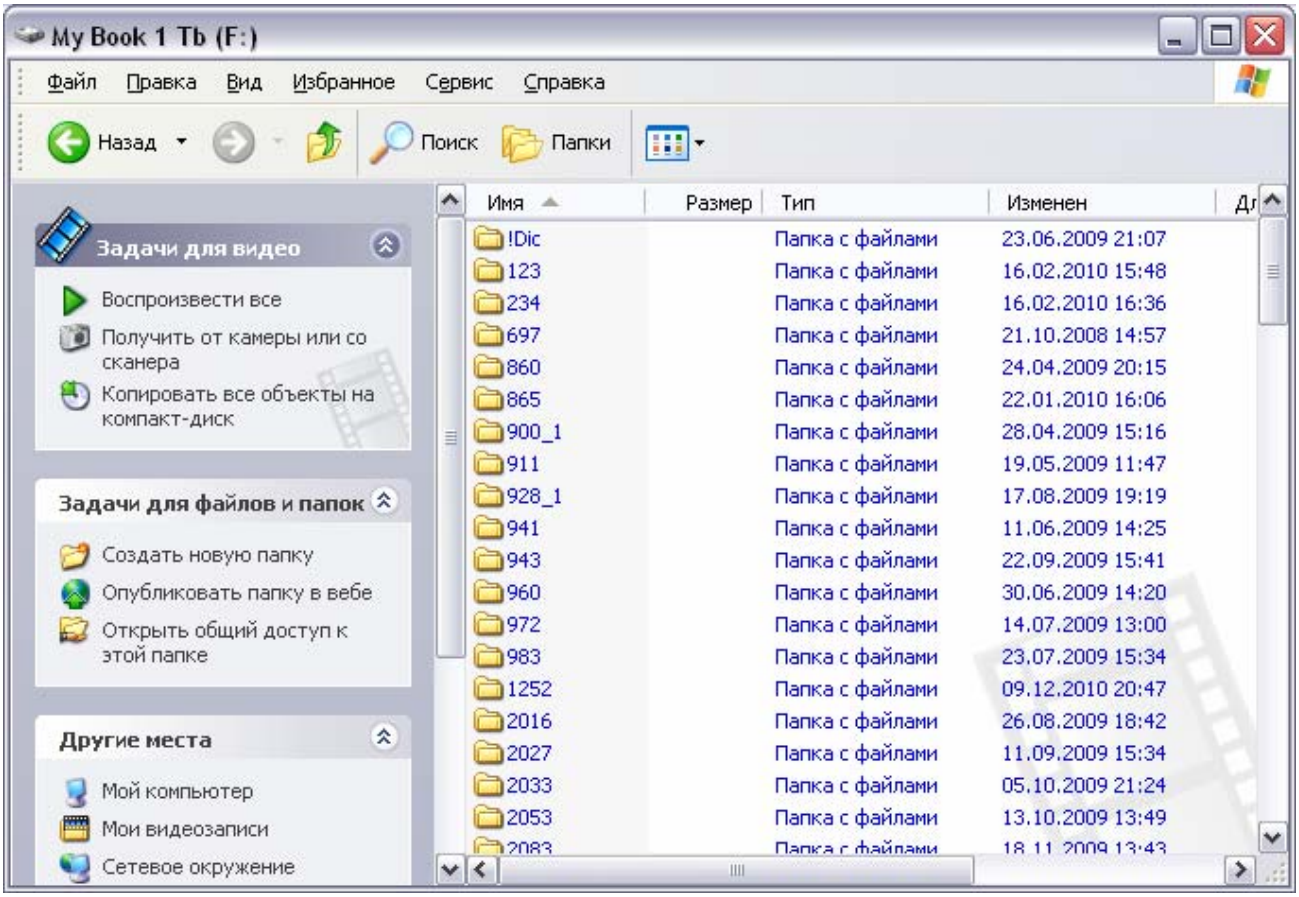


Рис. 7. Пример диска, на котором хранятся проекты

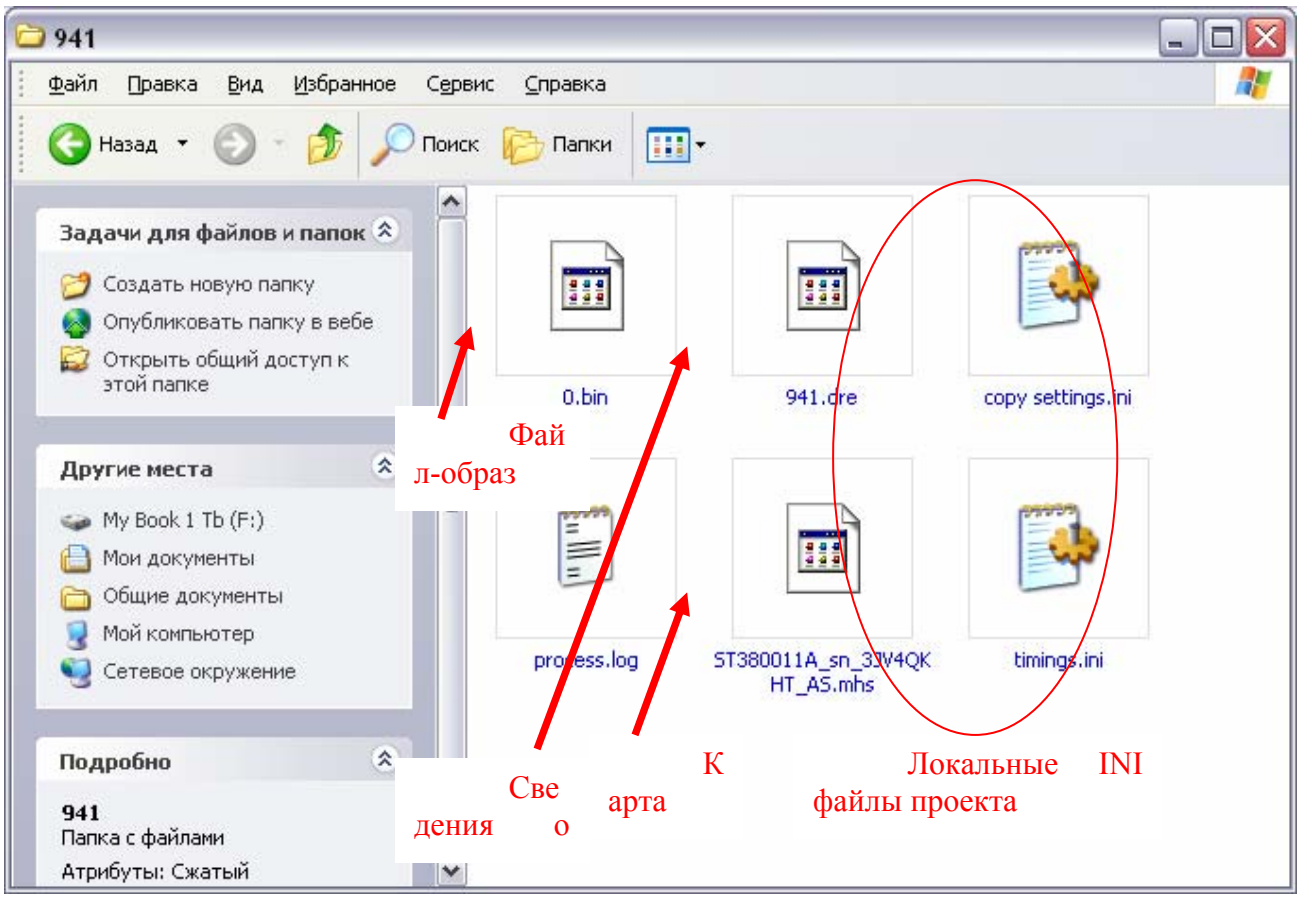
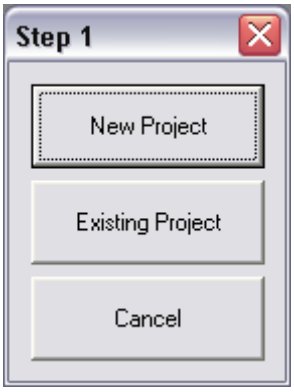


Рис. 8. Пример содержимого каталога проекта

Начало работы с утилитой

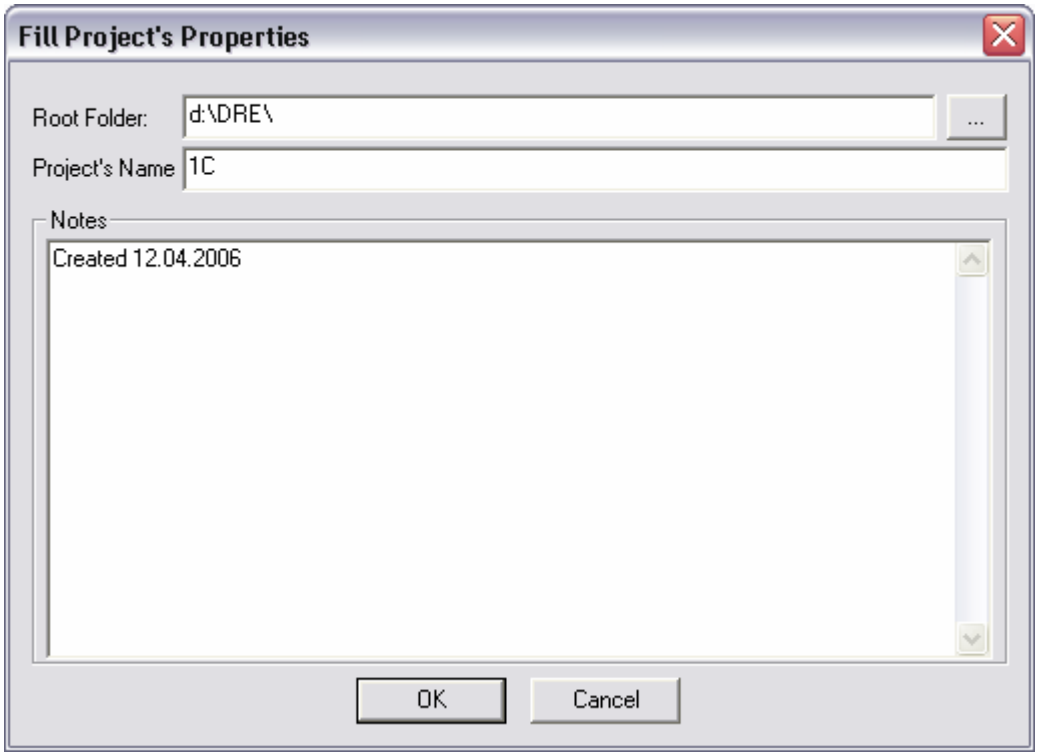
При работе с утилитой, используется понятие «проект». Это каталог, в котором сохраняются все необходимые файлы, а также – служебные пометки, которые могут добавляться в процессе работы.

При запуске утилиты, Вам будет предложено создать новый проект или выбрать ранее созданный:



Создание нового проекта

Если выбрать создание нового проекта, то Вам будет предложено заполнить его свойства:

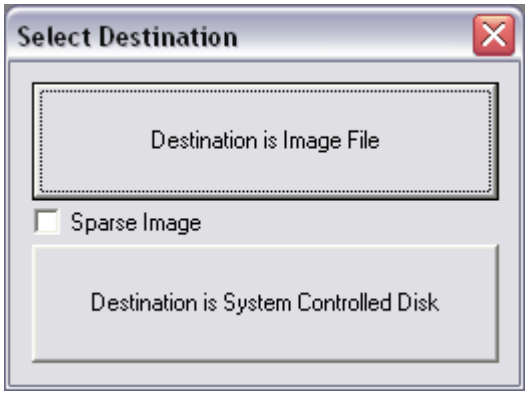


Строка Root Folder определяет каталог, в который будут помещаться все проекты. По умолчанию, она соответствует каталогу, в котором расположен исполняемый файл. Однако, при необходимости, каталог по умолчанию может быть изменён путём правки строки **Root Folder** секции **[DRE]** файла HDD.INI.

Строка Project's Name определяет уникальное имя проекта. На её основе будет сформировано имя каталога, в который будет помещаться вся информация, относящаяся к проекту.

В строке Notes Вы можете делать любые заметки, касающиеся ремонта. Кто принёс, что необходимо восстановить, какие проблемы были в процессе работы и т.п. Заметки могут быть изменены при каждом последующем открытии проекта. По умолчанию, туда подставляется дата создания проекта, так как она необходима в большинстве случаев.

После нажатия на кнопку OK, Вам будет предложено выбрать тип приёмника.



Приёмник типа файл будет располагаться в каталоге проекта, и иметь имя 0.BIN.

Однако здесь следует остановиться вот на каком нюансе. Во времена, когда накопители имели объём менее 250 гигабайт, всё было хорошо. Файлы-образы были основным инструментом восстановления данных. Однако когда неисправный накопитель имеет ёмкость в районе 500-750 гигабайт (или более), возникает ошибка, предположительно связанная с файловой системой Windows. Система выдаёт сообщение «Ошибка отложенной записи», после чего ранее сделанный образ (а он мог делаться неделю и более) становится недоступен. Тщательная проверка программы показала, что скорее всего, дело именно в Windows. Поэтому если Вы собираетесь копировать большие накопители в файлы-образы, Вы делаете это на свой страх и риск.

Само собой разумеется, не стоит размещать проекты на томах, размеченных в файловой системе FAT, так как максимальный размер файла там 2-4 гигабайта. И если образ имеет больший размер, во время копирования возникнет ошибка.

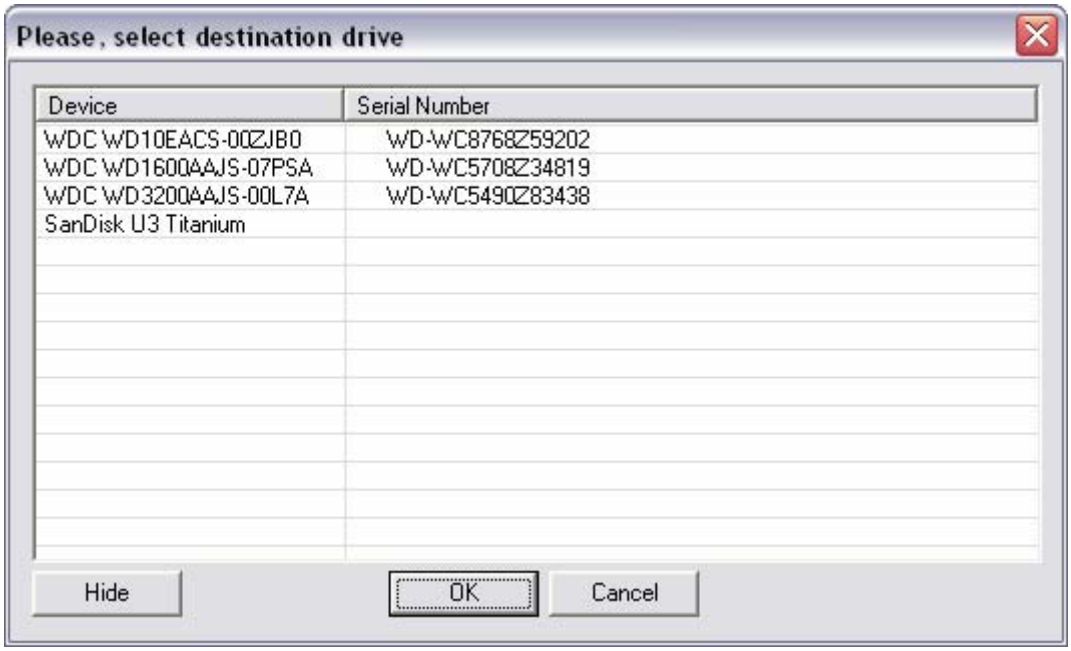
Флажок Sparse File необходим для создания «разряжённого» файла. В противном случае, при начале работы с образом (а именно – при файловом разборе или при копировании в обратную сторону) система может «повиснуть». При этом будет не ясно, что же произошло. Программа «висит» и не может быть даже «снята» через менеджер задач. А дело в том, что если в файле произошёл скачок, Windows заполнит пространство от места последней записи до точки «скачка» нулями. Заполнение идёт со скоростью 1-2 гигабайта в минуту. Сколько понадобится, чтобы занулить 200 гигабайт – не сложно рассчитать. И всё это время программа не будет подавать признаков жизни.

Для решения данных проблем, применяется механизм «разрежённых» файлов файловой системы NTFS. Некоторое время, программа HRT DRE открывала файл только в разрежённом виде. Однако ошибка отложенной записи на таких файлах возникает намного чаще, чем на обычных. Поэтому пришлось вынести эту функцию во флажок.

В целом, рекомендации могут быть следующие:

1) Для накопителей до 100 гигабайт, допускается установка флажка
2) Для накопителей от 100 до 250 гигабайт рекомендуется не устанавливать флажок, но и не пугаться, когда при файловом разборе или копировании в обратную сторону программа надолго «зависнет»
3) Для накопителей более 250 гигабайт, стараться не пользоваться файлом-образом (однако, на свой страх и риск – можно)

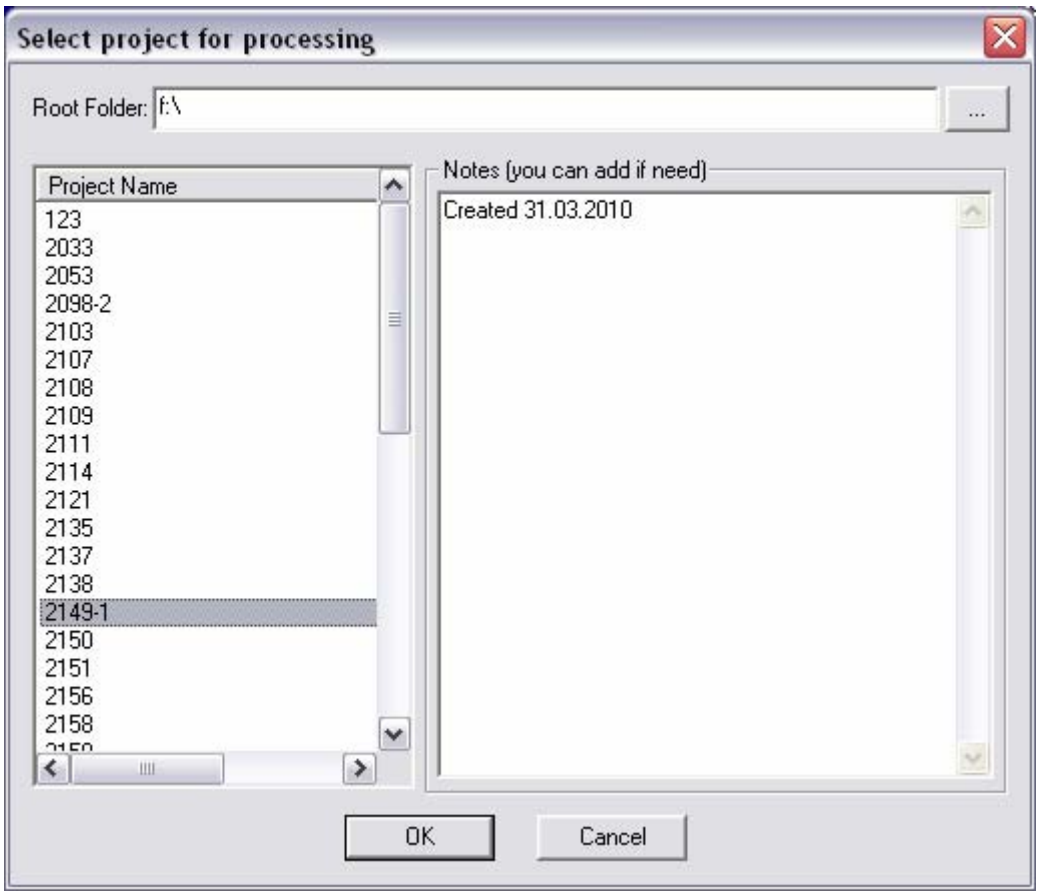
В случае с приёмником-накопителем, Вам будет дополнительно предложено выбрать его.



Будьте внимательны и не выбирайте системные диски. В противном случае, рано или поздно, система перестанет работать. Чтобы скрыть системный диск, необходимо выбрать его и нажать кнопку Hide. После этого, системный диск перестанет появляться в списке.

Открытие существующего проекта

При открытии существующего проекта, Вам будет выведен список всех проектов, расположенных в текущем корневом каталоге



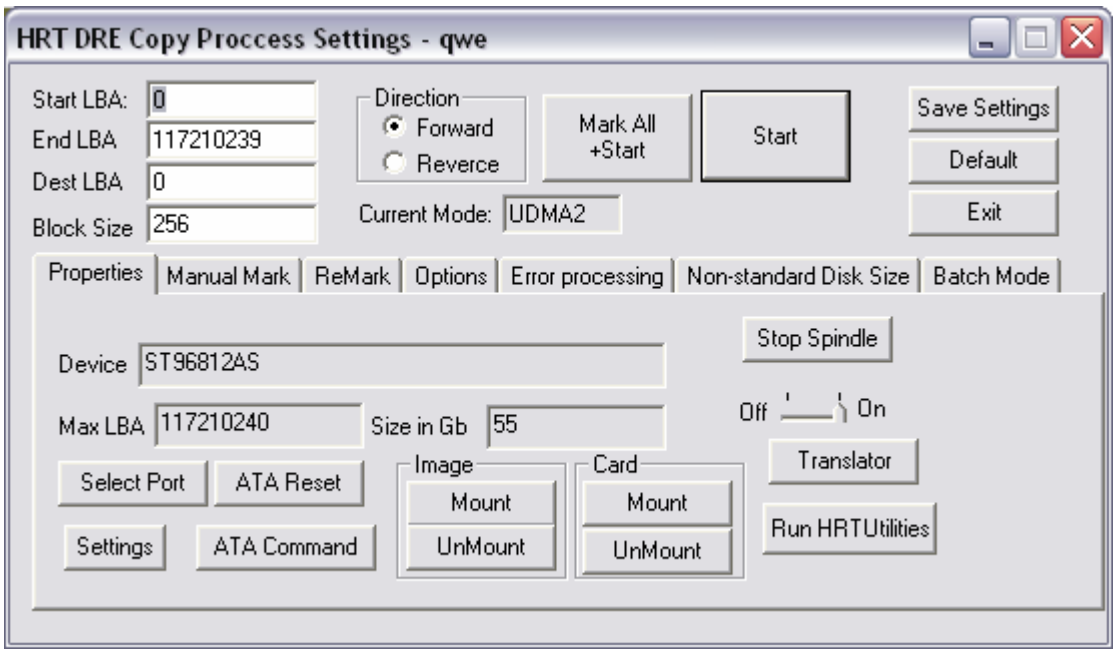
Сменить корневой каталог можно, нажав кнопку …

При выборе тех или иных записей в списке проекта, также будет отображаться комментарий, связанный с данным проектом. Вы можете не только просмотреть комментарий, но и отредактировать его. При нажатии на кнопку ОК, изменения будут сохранены, а проект – открыт.

С каждым проектом неразрывно связана карта. При открытии проекта, она будет загружена автоматически.

Главное окно программы

Главное окно программы содержит общую часть, расположенную сверху, а также переключаемые вкладки, расположенные ниже. В общей части сосредоточены параметры копирования.



Начальный и конечный **LBA** в комментариях не нуждаются. Они задают координаты начала и конца выполнения операции. Параметр **Dest LBA** обычно равен начальному LBA. Его назначение следующее: Допустим, Вы работаете с накопителем большой ёмкости. С него следует восстановить достаточно небольшой раздел. У Вас в наличии имеется только накопитель-приёмник малой ёмкости, на который указанный раздел поместится, а весь накопитель - нет. Как известно, ОС Windows прекрасно справится с накопителем, у которого нет таблицы разделов, а с LBA0 расположен сам раздел (режим «большой дискеты»). Параметр Dest LBA позволяет создать такую «большую дискету». Достаточно выбрать параметр Start LBA, равный началу копируемого раздела, а Dest LBA равным нулю. И данные будут смещены в начало накопителя-приёмника (или файла-образа). Как всегда, напомним, что при использовании сжатых каталогов NTFS данная проблема не стоит. Она стоит при копировании на накопитель-приёмнк или в обычный каталог NTFS. Узнать параметры разделов можно через механизм **Mark By FS**, описанный ниже.

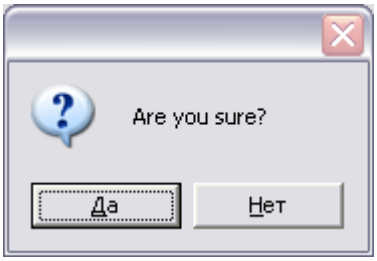
Группа кнопок **Direction** задаёт направление копирования. Оно может вестись или вперёд или назад. Обычно накопители копируются вперёд. Если же в начале имеется много секторов, замедляющих скорость копирования, то можно запустить копирование назад.

Как ни странно, иногда при копировании назад, удаётся вычитать даже те сектора, которые выдавали ошибку при прямом копировании. То есть, смена направления может быть произведена, если у Вас возникли затруднения при копировании вперёд. Причём достаточно прервать копирование, сменить направление и снова запустить процесс. Так как копирование базируется на карте, к накопителю-источнику не будет произведено ни одного лишнего обращения. Как только процесс дойдёт до той точки, где прервалось копирование в прямом направлении, он завершится.

В поле **Current Mode** выводится текущий режим карты HRT, т.е. в каком режиме в данный момент считываются данные с источника (PIO, UDMA).

Кнопка **Mark All+Start** запускает самый простой процесс копирования – без предварительной подготовки. Она просто отмечает ВСЕ сектора накопителя-источника, как подлежащие копированию и начинает процесс. **В простейшем случае, из всех кнопок нужна только эта. А все прочие настройки можно оставить по умолчанию.**

Внимание! Эта кнопка хороша при первом запуске. Однако если Вы случайно нажали на неё уже после того, как в карте появились какие-либо отметки, она «сметёт» их, не глядя (кроме отметки «Успешно скопировано»). Иногда это нежелательно. Поэтому при нажатии на кнопку выдаётся дополнительный запрос:



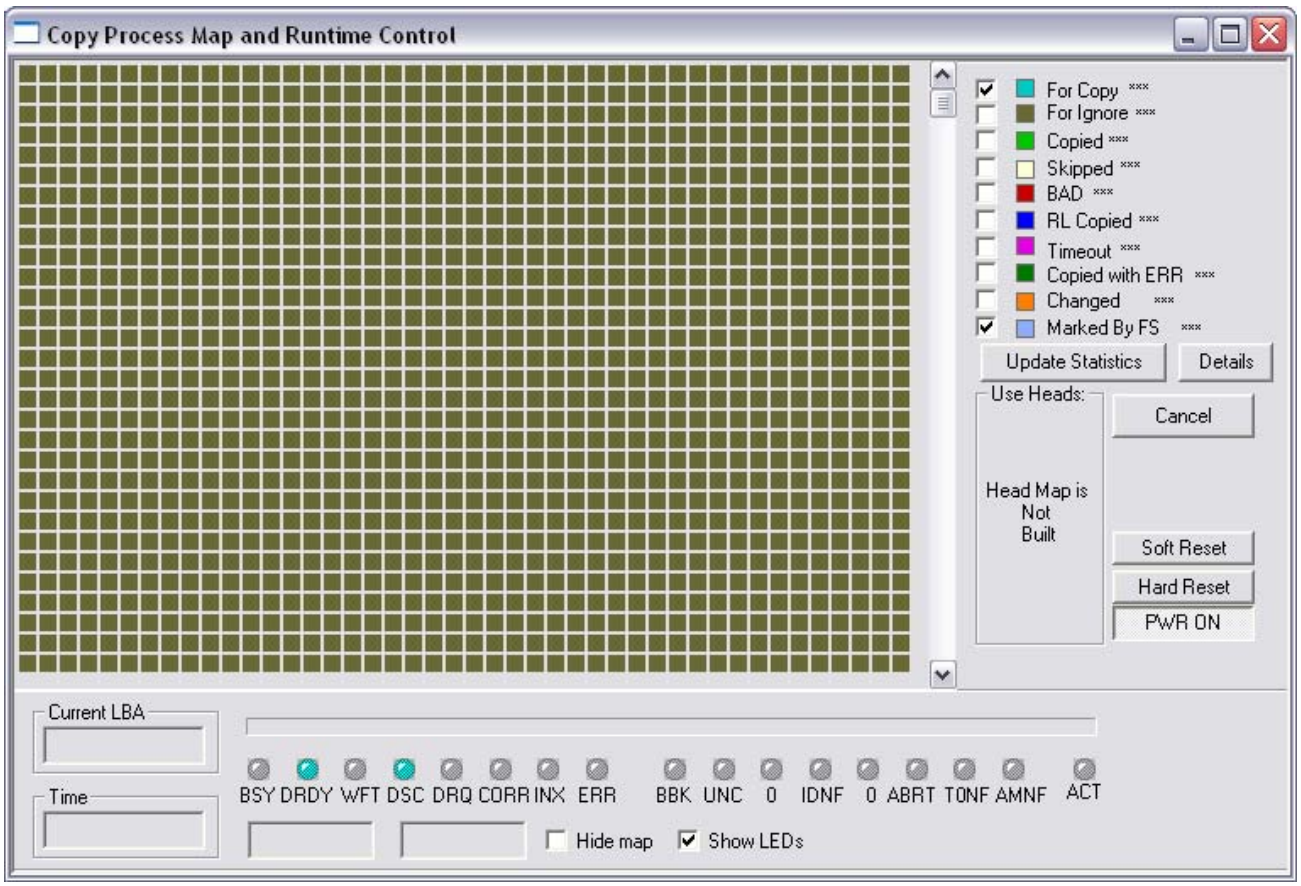
Кнопка же **Start** начнёт процесс для заранее подготовленной карты. Если карта не подготовлена (проект только что создан), процесс завершится довольно быстро, к источнику не будет произведено ни одного обращения, а на приёмник не будет записано ни одного байта данных. Карту следует подготовить одним из следующих методов (каждый из которых будет рассмотрен ниже):

- Прямым редактированием
- Редактированием на основе Файловой Системы
- Поголовочным построением при помощи специализированной утилиты комплекса HRT
- Редактированием карты по результатам предыдущего прохода копирования

Назначение вкладок, расположенных в нижней части окна, мы рассмотрим позже. Пока же перейдём к рассмотрению окна карты.

Окно карты

Текущее состояние карты всегда можно посмотреть в параллельном окне, внешний вид которого представлен ниже:



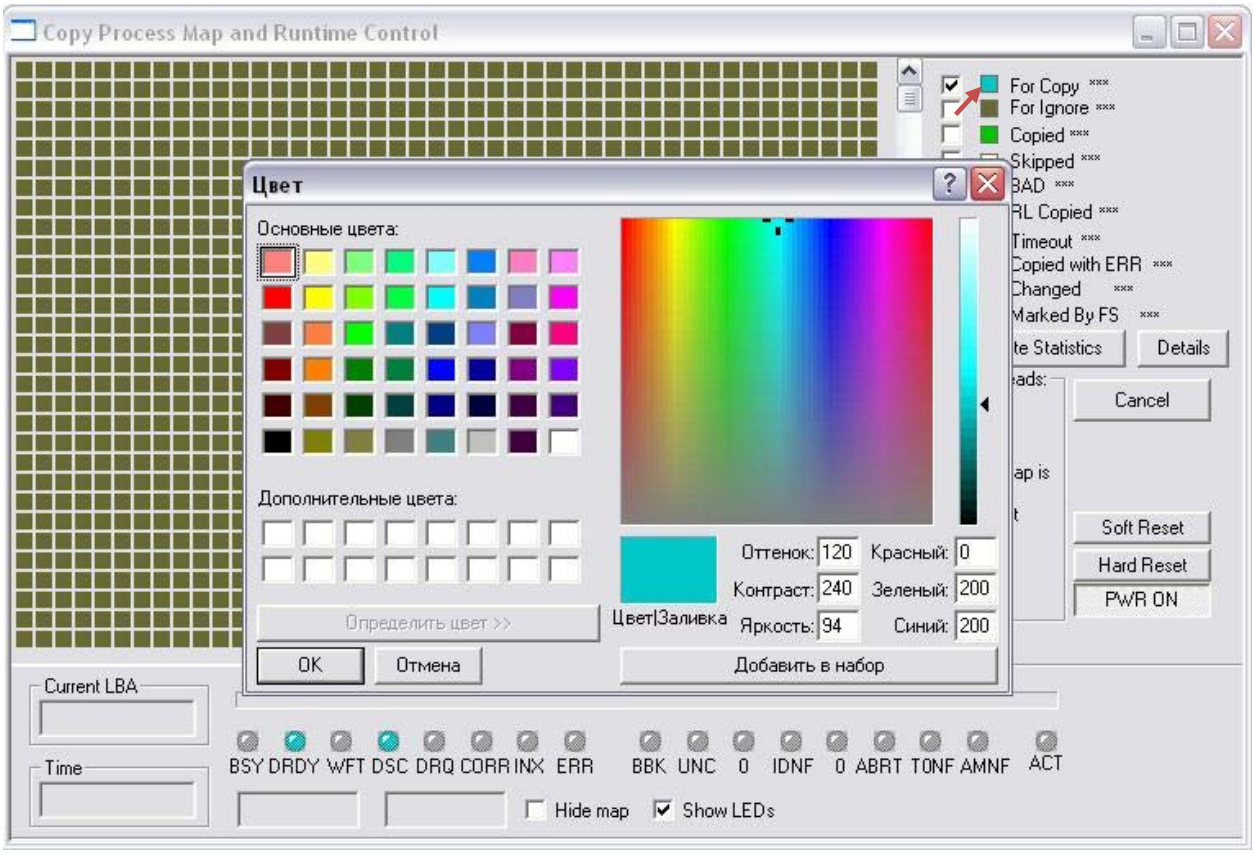
Один квадратик в карте соответствует одному сектору копируемого накопителя. Цвет квадратика определяет состояние сектора. В правом верхнем углу расположен список всех возможных состояний сектора:

Напротив каждой записи есть флажок, если он установлен, то сектора с выбранным состоянием участвуют в копировании. **Данные флажки можно изменять и в процессе копирования.**

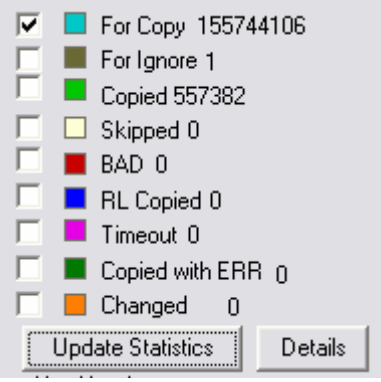
Тип	Пояснение
For Copy	Основная пометка для копирования. Ставится, например, при нажатии Mark All + Start. По умолчанию, копируются только сектора с такой пометкой.
For Ignore	Заполнение карты по умолчанию. Исходно считается, что никакие сектора не подлежат копированию. Однако, в дальнейшем, Вы можете изменить пометку секторов с этого типа на Marked for Copy, либо Marked By FS (во время файлового разбора). Разумеется, изменять можно на любые типы, но данные два изменения – наиболее типичны.

Copied	Все успешно скопированные сектора отмечаются данным типом. Многие обработчики не меняют данный тип ни на какой другой, так как если данные успешно скопированы, то чего ещё можно желать? Таким образом, это состояние – одно из самых устойчивых.
Skipped	К сектору не было попыток обращения. Просто на некотором расстоянии до него была обнаружена группа BAD секторов, и программа осуществила пропуск прыжком, отметив все пропущенные сектора данным образом.
BAD	При чтении данного сектора, накопитель выставил бит ERR в регистре статуса, а код ошибки был не UNC (либо в настройках программы стояло, что ошибки UNC не следует обрабатывать), поэтому программа записала в соответствующий сектор копии сигнатуру BAD.
RL Copied	Сектор скопирован при помощи операции Read Long. Несмотря на красивое описание, данная операция крайне ненадёжна. Попробуйте считать хорошие сектора при помощи этой команды, получите массу неверных битов. Однако традиционно все копировщики умеют использовать данную команду. То есть, данный тип копирования оставлен в угоду традициям и не рекомендуется для применения.
Timeout	Во время чтения возник таймаут. Вероятность того, что сектор был сбойным – 50 на 50. Или сбойный или ещё вычитается. Однако для экономии времени разбирательство было оставлено на следующий проход копирования.
Copied with ERR	Данный сектор читается с ошибкой. Однако произведена попытка восстановления данных с максимальной правдоподобностью (алгоритм описан ниже).
Changed	Сектор изменён в дампе. Суть изменений будет описана ниже
Marked By FS	Сектор помечен к копированию в файловом разборщике.

Щелчок по цветному квадратику рядом с флажком позволит изменить цвет квадратика.



Кнопка **Update Statistics** позволяет посмотреть текущую статистику отметки секторов в карте. При нажатии напротив каждого типа сектора выводится статистика, сколько секторов такого типа присутствует на приемнике:



Для более детальной статистики введена кнопка **Details**, при ее нажатии появляется диалог полной статистики, в которой уже присутствует статистика по каждой физической головке в отдельности:

Detail Statistics By Heads

State	Global	h0	h1	Extra
For Copy	155744106	77902516	77841590	0
For Ignore	67408	0	0	67408
Copied	557382	278717	278665	0
Skipped	0	0	0	0
BAD	0	0	0	0
RL Copied	0	0	0	0
Timeout	0	0	0	0
Copy W/O CRC	0	0	0	0
Changed	0	0	0	0

Update Exit

Клавиши **Soft Reset** и **Hard Reset** служат для подачи накопителю программного или аппаратного сброса. В некоторых случаях позволяет вывести накопитель из зависания. Эти кнопки допускается нажимать даже во время копирования.

Кнопка **Cancel** работает, когда идёт процесс копирования, прерывая его. Это необходимо для временной приостановки процесса, например, если необходимо подправить параметры, ответственные за обработку ошибок, построить карту по головкам, изменить направление копирования или, в конце концов, использовать адаптер HRT для более срочного заказа. Впоследствии процесс будет продолжен согласно карте, то есть, с точки, где было произведено его прерывание.

Теперь рассмотрим нижнее поле, расположенное под картой

Светодиоды аналогичны таковым для утилит HRT. Однако есть одна тонкость, связанная с особенностями микросхем ITE и Marvel. Если копировать накопитель с интерфейсом IDE, никаких проблем не возникает. Однако такие накопители уходят в прошлое. Им на смену приходят накопители с интерфейсом SATA. И если копировать их, то при возникновении ошибки иногда микросхема ITE завешивает весь компьютер. Чтобы этого не происходило, необходимо отключать отображение светодиодов. Для этого следует снять флажок Show LEDs.

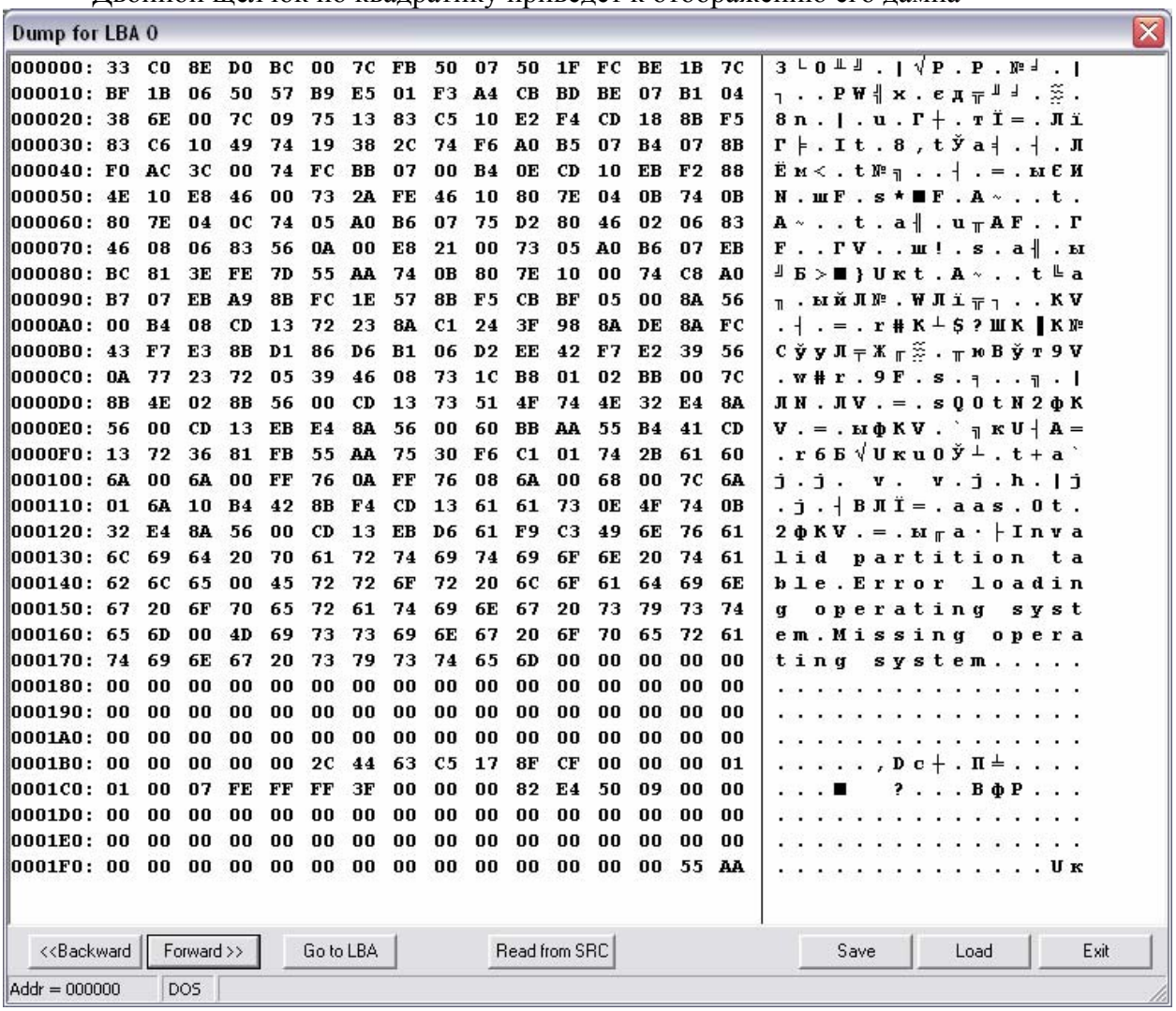
Поле **Current LBA** показывает номер текущего обрабатываемого сектора. Во время копирования это текущий копируемый сектор. Если же щёлкнуть по любому из квадратиков «мышью», то в поле появится значение LBA, соответствующего квадратику.

Флажок **Hide Map** позволяет ускорить процесс копирования примерно на 10-30% за счёт того, что система не будет тратить время на перерисовку карты. Рекомендуется снимать его только в случаях, когда процесс копирования подтормаживает, чтобы рассмотреть характер проблемы, а в остальных случаях – устанавливать.

Поле **Time** автоматически обновляется в процессе копирования, отображая время, затраченное на сам процесс, причем время считается от момента очередного запуска процесса копирования.

Дамп сектора

Двойной щелчок по квадратику приведёт к отображению его дампа



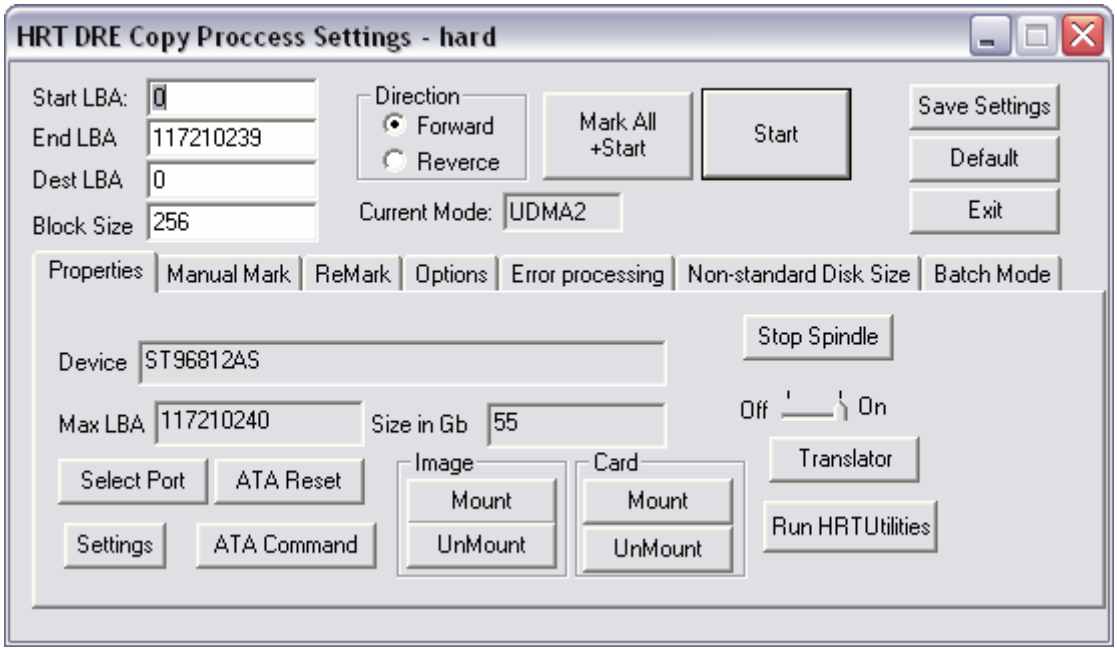
Клавиши **Forward** и **Backward** выводят дамп последующего и предыдущего сектора соответственно. С помощью клавиши **Go to LBA** можно перейти к любому LBA блоку. Как уже говорилось ранее, обращение к источнику происходит всего один раз, и все последующие чтения производятся с приемника. Если же все-таки необходимо повторно прочитать сектор с источника, то это можно сделать, нажав клавишу **Read from SRC.** Также, можно сохранить считанный сектор в файл и восстановить его обратно из файла, причем запишется сектор на диск приемник или в образ, в который сохраняются данные пользователя.

Основная цель данного дампа следующая: Допустим, сектор считался с ошибкой. Однако из опыта работы Вы знаете, что там содержатся важные сведения для файловой системы. Тогда Вы заходите в дамп, правите содержимое (да хотя бы зануляете, чтобы не было ложных указателей) и сохраняете результат. Чтобы подчеркнуть тот факт, что сектор был изменён в дампе, ему будет присвоен статус Changed.

Теперь, когда мы познакомились с самыми важными блоками (верхняя часть основного окна и окно карты), можно приступить к рассмотрению вкладок, обеспечивающих точные настройки копирования.

Вкладка Properties

На данной вкладке сгруппированы элементы для просмотра свойств накопителя, а также посылки управляющих воздействий на него.



В поле **Device** отображается имя накопителя, а в полях **Max LBA** и **Size in GB** – объём накопителя в секторах и в «честных» гигабайтах (1 гигабайт равен 1024 мегабайта).

Кнопка **Select Port** позволяет изменить адаптер, к которому подключён накопитель (если у Вас установлено несколько адаптеров HRT).

Кнопка **ATA Reset** позволяет послать в накопитель программный сброс.

Кнопка Settings позволяет изменить настройки работы программы. В том числе, она позволяет переключаться между режимами PIO и UDMA, а также задать тип UDMA.

Важная информация: Экспериментально выяснено, что самым надёжным режимом является UDMA2. Повышение режима с некоторыми кабелями приводит к появлению битовых ошибок. Поэтому крайне рекомендуется работать в режиме UDMA не выше второго.

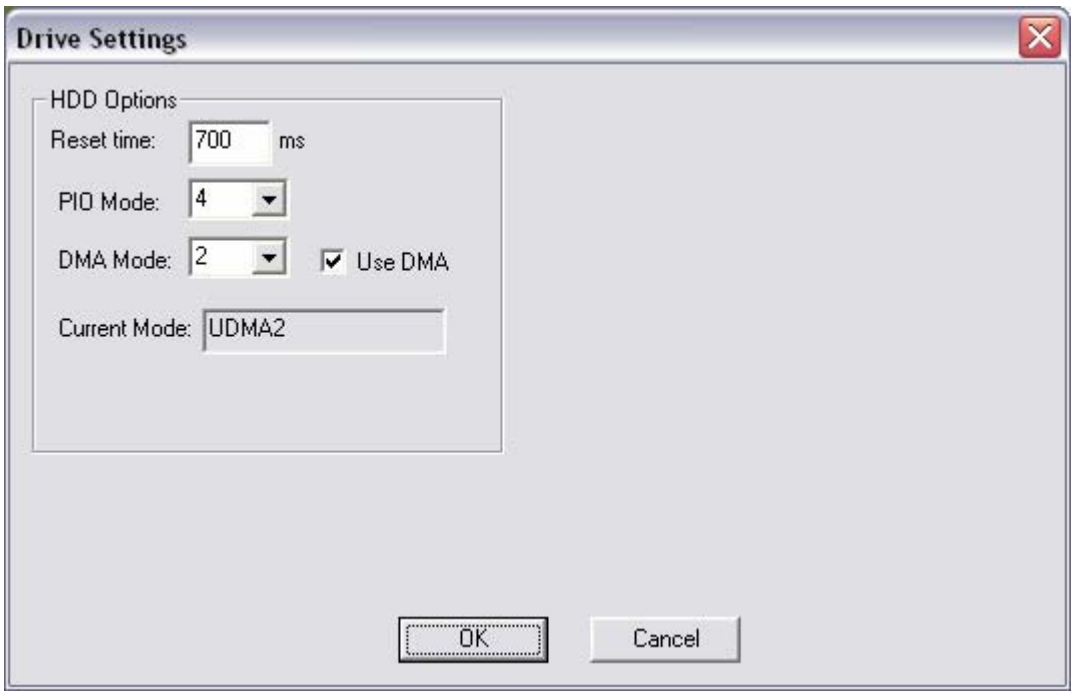


Рис. 9. Диалог, появляющийся при нажатии кнопки Settings

Кнопка **ATA Command** вызывает стандартный диалог HRT, позволяющий подавать ATA команды. Этот механизм настолько важен, что его рассмотрение мы вынесем в отдельный раздел ниже.

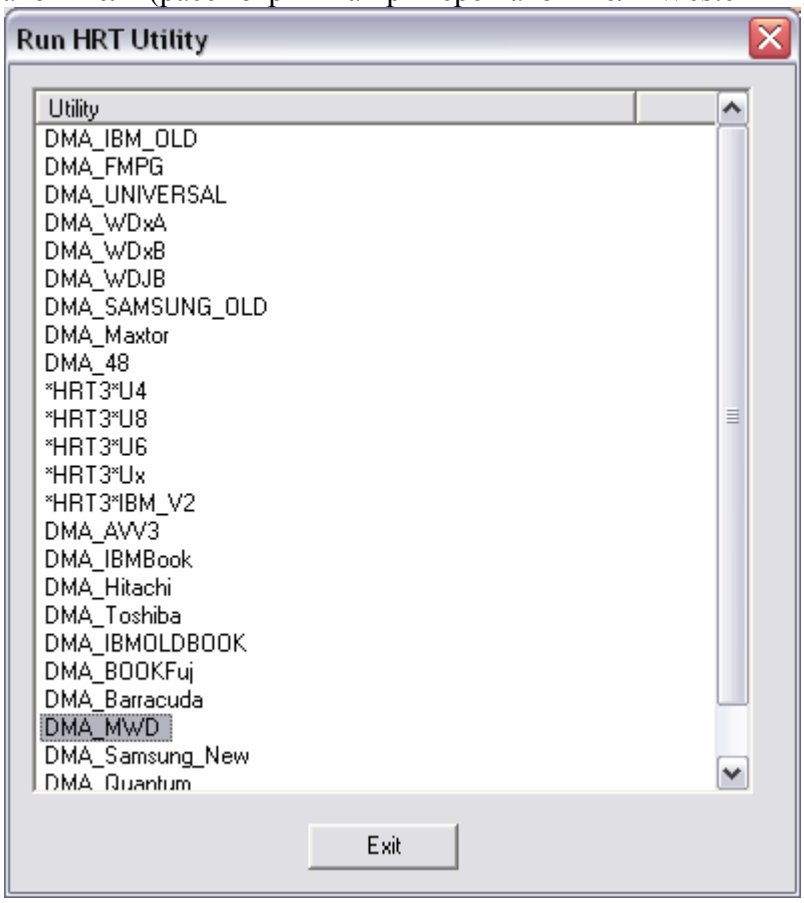
Переключатель **Off/On** позволяет управлять питанием накопителя

Кнопка **Stop Spindle** посылает накопителю команду остановки шпиндельного двигателя. Эта методика традиционно применяется для выполнения операции Hot Swap.

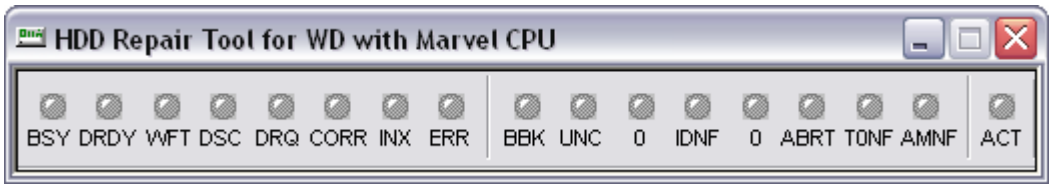
Историческое отступление. Метод Hot Swap начал применяться в те далёкие времена, когда почти никто из ремонтников не слышал о таком понятии, как «лоадер». В некоторых случаях, он применяется и в наше время. Суть метода проста: Пусть имеется накопитель, служебная область которого настолько разрушена, что он не может нормально стартовать. И пусть отсутствует штатная методика оживления такого накопителя. В этом случае, берётся накопитель, который называется «донор». При необходимости (и наличии возможности), его служебная область приводится в соответствие со служебной зоной неисправного накопителя (например, копируется дефект-лист). Далее донор включается и выходит в готовность. Теперь у него останавливается шпиндельный двигатель, плата отвинчивается (без отключения питания) и переставляется на неисправный гермоблок. Получаем накопитель с загруженной микропрограммой. Поэтому во многих случаях, с этого накопителя можно считывать данные. Детали в каждом случае свои, но в общем и целом – Hot Swap выполняется именно так.

Кнопка Run HRT Utility выполняет чрезвычайно важную функцию: вызывает специализированную утилиту HRT для построения карты по головкам. Дело в том, что для ATA накопителей, в отличие от SCSI, не существует универсальной команды преобразования логического сектора к физическим координатам PCHS. Каждый производитель делает это по-своему. Поэтому невозможно построить карту по головкам, пользуясь только стандартной утилитой. Необходимо для каждого типа использовать специализированную утилиту. В целом, алгоритм преобразования выглядит следующим образом:

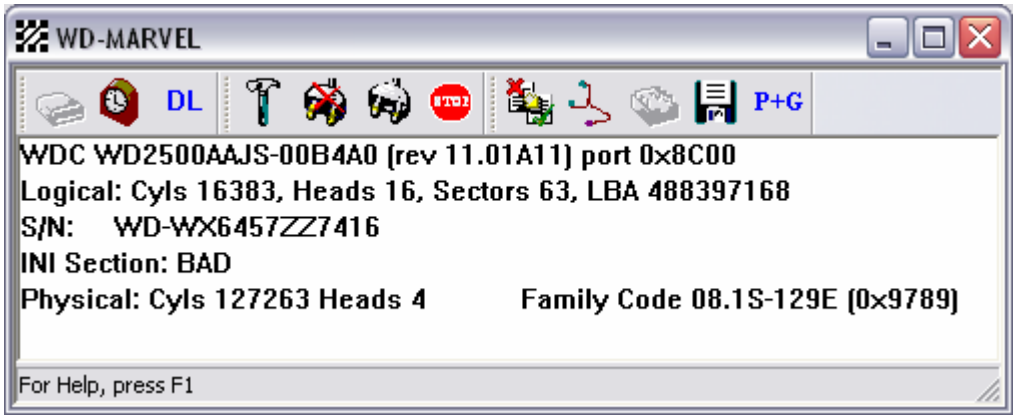
Нажать кнопку Run HRT Utilities. В появившемся окне найти в списке соответствующий накопитель (рассмотрим на примере накопителя Western Digital)



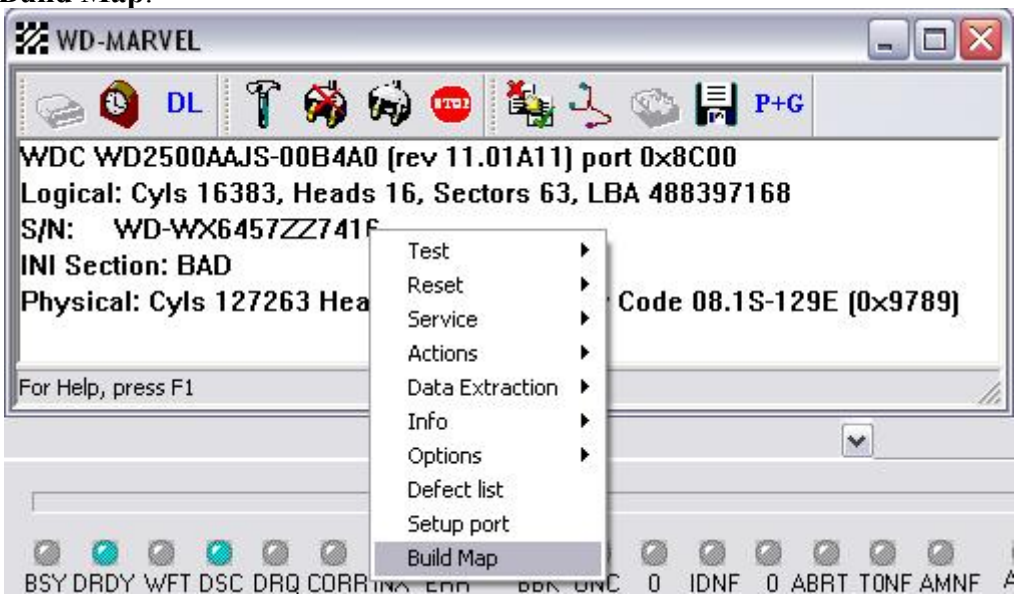
Двойной щелчок по элементу списка откроет утилиту HRT:



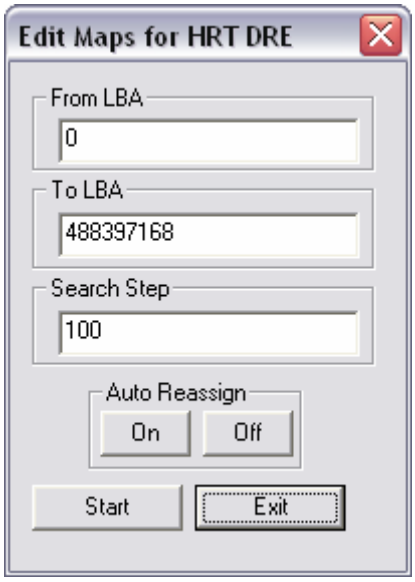
В ней стандартным образом выбираем накопитель. Получаем стандартное окно накопителя.



Однако, это окно отличается от классического варианта. В конце меню появляется пункт **Build Map**:

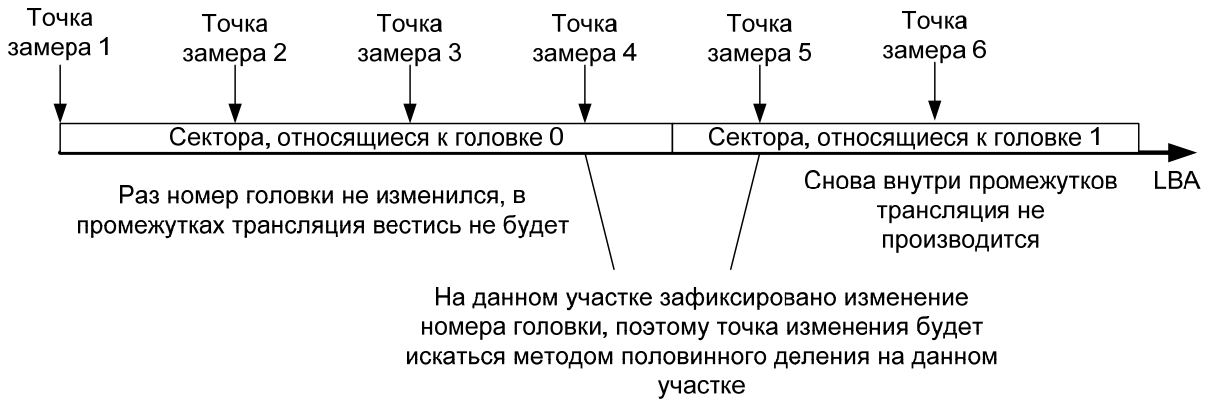


При выборе данного пункта, открывается окно построения карты

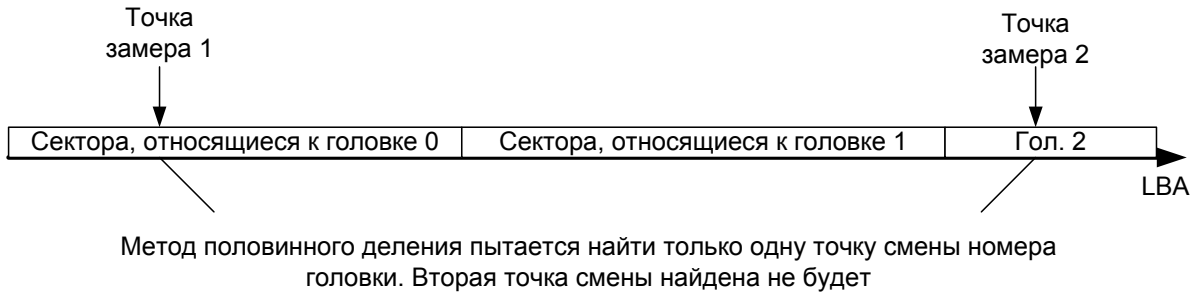


Поля From LBA и To LBA не вызывают никаких вопросов. Поле Search Step требует некоторых комментариев. Дело в том, что команда преобразования логического сектора в физический требует слишком много времени. Если производить преобразование для каждого сектора, процесс будет длиться несколько дней, что в условиях умирающего накопителя (да и нетерпеливого клиента) совершенно неприемлемо. Поэтому программа пользуется тем фактом, что некоторое количество секторов принадлежит одной головке.

Поэтому программа транслирует сектор LBA и LBA + Step. Если и там и там номер головки совпадает, начинается анализ следующего промежутка, размером Step. Если же номер головки изменился, то на рассматриваемом диапазоне имеется точка перехода между головками. Эта точка ищется методом половинного деления.



Таким образом, чем больше параметра Search Step, тем быстрее завершится процесс трансляции. Но с другой стороны, если параметр задан слишком большой, то на рассматриваемом диапазоне может произойти не одна, а две (или даже более) смены головок, а утилита сочтёт это за одну смену и отработает некорректно.



Существуют накопители, переключающие головки по окончании каждого цилиндра (обычно это старые накопители – Quantum, Fujitsu и т.п.). Для них данный параметр должен быть не более половины минимального SPT (именно из этих соображений выбрано значение по умолчанию 100). Большинство же современных накопителей осуществляют трансляцию «кольцами». При этом по мере роста LBA переключается номер цилиндра, а номер головки – остаётся прежним. И лишь по достижении определённого значения (например, на каждом 128-м цилиндре) происходит смена головки. Для таких накопителей можно задавать значения в районе 1000 и даже более (вплоть до 50000). Характер трансляции можно выявить, пользуясь диалогом Issue Translator, имеющимся в большинстве утилит комплекса HRT. На Рис. 10 приведён пример настройки с шагом 10000, выставленный для накопителя Western Digital.

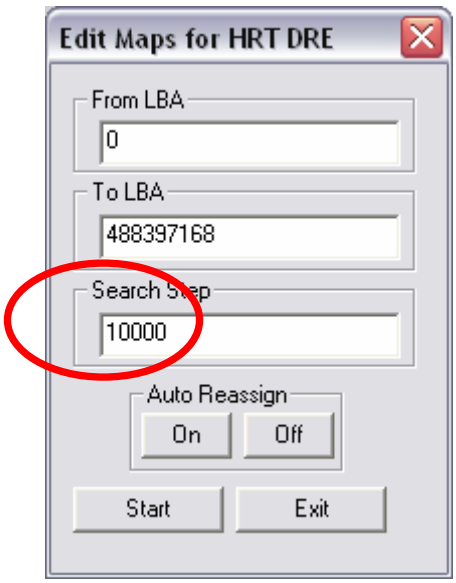


Рис. 10. Пример изменённого шага поиска для накопителя Western Digital

Когда трансляция завершена, выходим из специализированной утилиты HRT. В окне карты появляется дополнительный список головок (см. Рис. 11). Снимая галочки, можно исключать головки из процесса копирования. Снимать и устанавливать галочки можно даже не останавливая процесса.

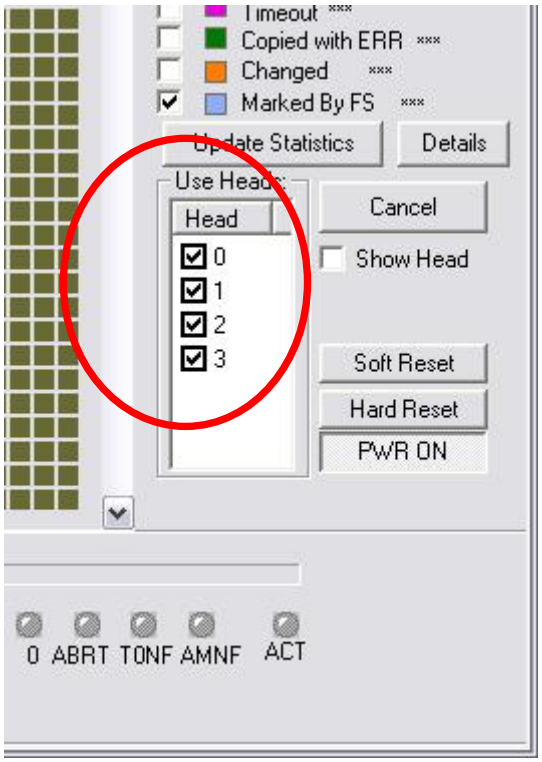


Рис. 11. Появилась возможность исключать головки из процесса копирования

Кроме того, если установить флажок Show Heads, каждый квадратик на карте начнёт отображать номер головки, которой соответствует текущий LBA (см.). Если какой-то сектор не был оттранслирован (например, процесс расчёта головок был прерван пользователем), номер головки будет F. **Стоит отметить, что вывод номеров головок замедляет процесс работы на 5-10%. Поэтому включать отображение стоит только по необходимости.**

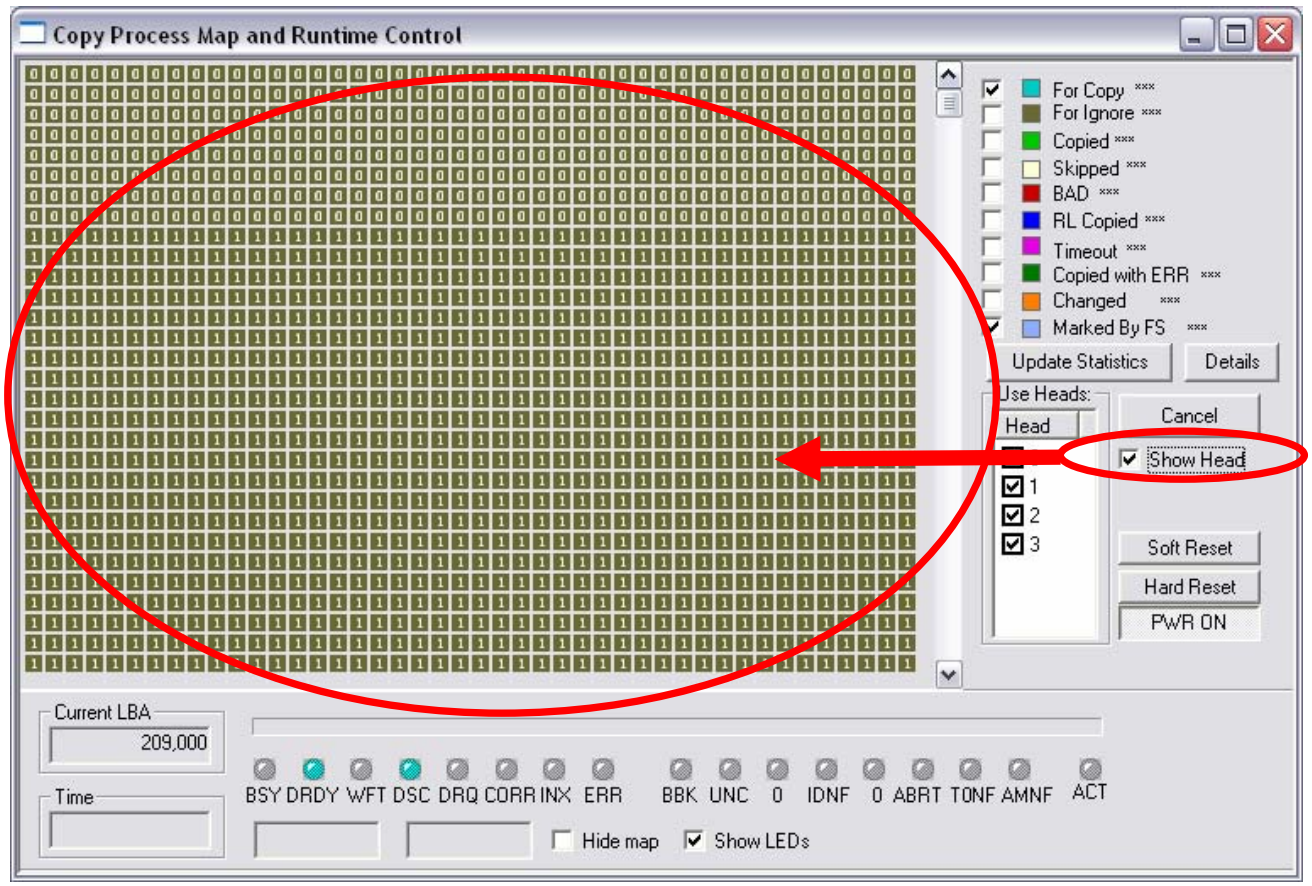


Рис. 12. Карта с отображением номеров головок на каждом из квадратиков

Пока писался этот раздел документации, на соседней машине копировался накопитель Western Digital, покрывшийся множественными BAD блоками. Причём характер вычитывания был такой: Сначала большой кусок читается быстро, затем – начинаются тормоза. Через некоторое время – снова быстро. А ведь накопитель WD имеет стеклянные пластины, поэтому если он посыпался – читать надо как можно быстрее, иначе вскоре процесс разрушения станет просто ужасным. Поэтому процесс копирования был прерван, произведено построение карты по головкам. Далее, включено отображение номеров головок на карте и процесс запущен вновь. Сразу стало ясно, что «пылит» головка 1. Отображение головок было выключено (для ускорения работы), а в списке головок – снята галочка около головки 1. В результате, все прочие головки были скопированы с огромной скоростью. Дальше, головка 1 была возвращена и процесс был запущен вновь. Разумеется, головка 1 читалась всю ночь, но прочим поверхностям летящая с неё пыль уже не угрожала. Они уже были считаны.

Отдельно следует упомянуть о том, что если у накопителя не пуст G-LIST, даже построенная карта по головкам не защитит на 100% от обращения к неисправной головке. Дело в том, что резервные сектора не обязательно будут размещаться на той же головке, что и основные. Например, резервные сектора накопителей IBM (и последовавших за ними HITACHI) всегда располагались на головке 0. И дальше ситуация получается точно такая же, как и со слишком большим шагом поиска. На Рис. 13 показан случай, когда замещающий сектор вообще не будет замечен. А на Рис. 14 показана ситуация, где замещающий сектор может даже вызвать ошибку в алгоритме поиска точки изменения номера головки. Лучшее решение в такой ситуации – задавать размер шага равным единице, но тогда время построения карты будет неприемлемо велико. Второе решение – при возможности очищать G-LIST. Но в целом, практика показала, что проблема обычно не мешает работе, просто про неё следует помнить.

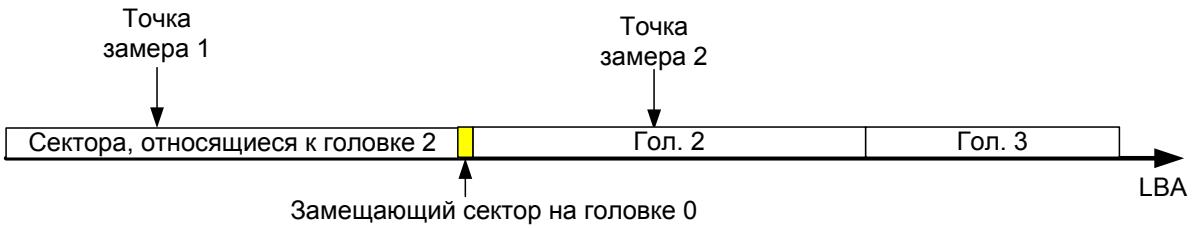


Рис. 13. Ситуация, в которой замещающий сектор вообще не будет обнаружен

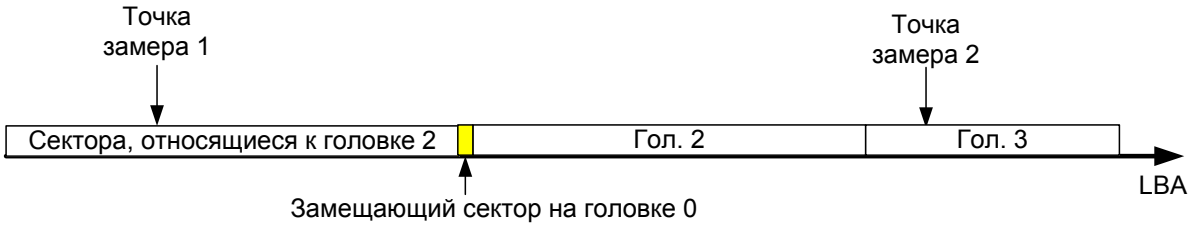


Рис. 14. Ситуация, когда замещающий сектор может создать ошибочную ситуацию

Диалог подачи ATA команд

Диалог подачи ATA команд позволяет подавать произвольные команды в накопитель. Этот механизм предназначен в первую очередь для исследовательских целей (например, для подготовки скриптов, описанных ниже), но может пригодиться и для подготовки накопителя к копированию. Внешний вид окна подачи ATA команд приведён на Рис. 15.

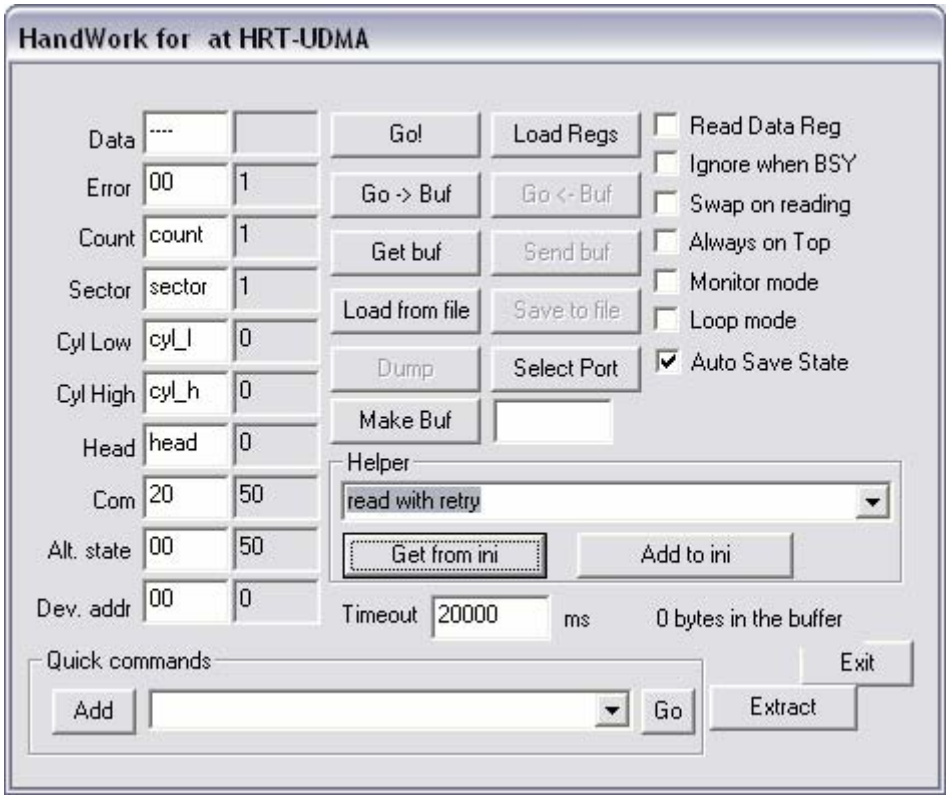


Рис. 15. Внешний вид окна подачи команд

Слева расположен столбец параметров, отправляемых в порты. При этом первый порт – это порт данных, он 16-разрядный, далее идут 7 основных и 2 альтернативных порта. В классическом случае, они 8-разрядные. Под классическим, мы подразумеваем, стандарт ATA в том виде, в каком он появился изначально, ещё до ввода команд с LBA48.

Бывают случаи (особенно при подаче Super-команд), когда запись в какой-либо порт приведёт к неработоспособности команды (в обычных ATA командах такой проблемы нет). На этот случай, имеется специальный заполнитель – (два минуса). Для регистра данных, соответственно, четыре минуса. Рассмотрим пример подачи команды DEVICE IDENTIFY (ECh) с заполнением только нужных портов.

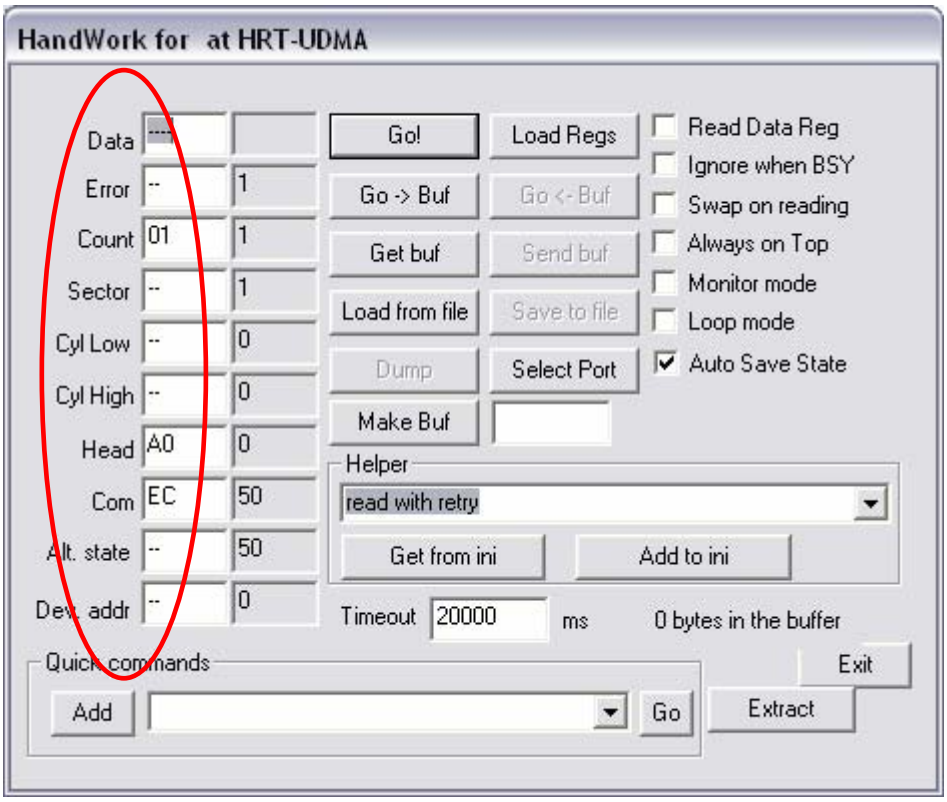


Рис. 16. Заполнение "--" говорит о том, что данный регистр не следует записывать

Когда все регистры заполнены, их можно передать в порты. Для этого следует нажать кнопку Go!. Чтобы считать отклик накопителя, следует нажать Load Regs. Содержимое же регистра статуса можно наблюдать на светодиодах, расположенных на карте. Если взведён бит ERR, то также будет отображаться и содержимое регистра ошибок.

Если команда подразумевает передачу или приём данных, накопитель взведёт флаг DRQ (к недостаткам стандарта ATA можно отнести тот факт, что невозможно выяснить, чего хочет накопитель, он всегда взведёт DRQ, будь то желание передать или принять данные по протоколу PIO или UDMA).

Рис. 17. Пример взведённого флага DRQ

Чтобы принять данные, следует нажать кнопку Get Buf. Чтобы передать – Send Buf. Однако, возможна такая ситуация, когда накопитель собирается отдавать данные, а Вы нажали Get Buf. А в ответ на кнопку Get Buf, утилита будет принимать данные, пока горит DRQ. А накопитель никогда не погасит его, ведь он ждёт, что данные будут не принимать от него, а наоборот, отдавать ему. И так будет, пока не переполнится память. Чтобы

разорвать этот порочный круг, отключите питание накопителя. Сигнал DRQ при этом будет снят, и процесс остановится.

Если Вы заранее знаете, что команда передаёт данные в накопитель, то вместо последовательного нажатия на Go! и Send Buf, нажмите Go->Buf. И наоборот, если Вы знаете, что команда передаёт данные из накопителя, вместо Go! и Get Buf нажмите Go<-Buf.

Кнопка Save to File позволяет сохранить содержимое буфера в файл, а кнопка Load From File – наоборот, загрузить содержимое файла в буфер.

Кнопка Dump открывает окно дампа (описано ниже). В дампе Вы можете отредактировать содержимое буфера.

Кнопка Select Port позволит выбрать другой адаптер HRT, если у Вас их несколько.

Теперь поговорим об использовании буфера. Чтобы что-то передать в накопитель, надо это что-то где-то взять. Можно считать из накопителя через подачу команды, возвращающей данные. Можно – считать из файла. Но в некоторых случаях неудобно ни то ни другое. Типичный пример – работа с парольной защитой. Если надо быстро установить пароль, или снять его, пользуясь ATA командами, проще всего воспользоваться исходно чистым буфером, в который быстро вписать средствами дампа текст пароля. Поэтому в утилите имеется возможность создать пустой буфер. Впишите в поле справа от кнопки Make Buf размер в байтах (напоминаем, что простое число будет считаться десятичным, а число с префиксом «0x» - шестнадцатеричным, так что записи 512 и 0x200 идентичны) и нажмите кнопку Make Buf. Теперь Вы можете редактировать содержимое буфера в дампе.

Рассмотрим теперь доступные флажки.

Флажок **Read Data Reg** обычно сброшен. Если он установлен, то при нажатии кнопки **Load Regs**, среди прочих регистров, считывается и регистр данных. Это полезно, если Super-команда возвращает в нём какие-то полезные сведения. Также это удобно, чтобы определить, зачем взвёлся **DRQ**. Если из регистра читаются разные сведения – значит накопитель точно хочет нам что-то передать. Если не читаются – ещё ничего не значит. Возможно, накопитель ждёт от нас данные, но может так оказаться, что он просто шлёт одинаковые значения, а мы воспринимаем их за отсутствие чтения. В остальных случаях, этот значок вреден. Ведь если взведён **DRQ**, то каждое чтение из регистра данных приводит к тому, что очередное слово навсегда уходит из буфера накопителя. И если мы считали содержимое портов, чтобы посмотреть регистр статуса или любой другой регистр, а затем нажали **Get Buf**, то накопитель отдаст не 512, а 510 байт. Куда делись два байта? Они ушли при чтении регистров, так как флажок **Read Data Reg** был установлен.

Флажок **Monitor Mode** хоть и расположен значительно ниже, стоит того, чтобы описать его в этом месте, иначе без него будет сложно пояснить назначение других флажков. Светодиоды позволяют просматривать содержимое регистра статуса. В крайнем случае – статуса и ошибки. А иногда, особенно при исследовании различных режимов самодиагностики накопителей, полезно смотреть содержимое всех регистров. Нет ничего проще. Устанавливаем флажок Monitor Mode и утилита будет имитировать нажатие кнопки **Load Regs** 10 раз в секунду. Таким образом, можно будет постоянно смотреть содержимое всех регистров.

Флажок **Ignore When BSY** полезен при установленном флажке Monitor Mode. Допустим, накопитель большую часть времени находится в состоянии BSY. А когда взведён флажок BSY, накопитель не только не реагирует на внешние воздействия, он ещё и не отдаёт никакие другие регистры. Поэтому все сведения, которые появлялись между состояниями BSY, будут утеряны. Чтобы этого не происходило, установите флажок **Ignore When BSY**. В этом случае, порты будут отображаться только при сброшенном BSY. И, если накопитель «ушёл в себя», Вы сможете контролировать последнее осознанное содержимое портов.

Флажок **Swap On Reading** позволяет прочесть имя накопителя и прочие сведения, возвращаемые по команде ECh, ведь они возвращаются с переставленными байтами. Установите флажок, подайте команду ECh, примите буфер, откройте дамп. Все строки будут читаемые. Другие задачи для данного флажка пока не придуманы.

Флажок **Always On Top** закрепляет диалог поверх других окон. В целом, окно является масштабируемым, а основные кнопки задублированы в контекстном меню. Поэтому можно установить данный флажок и привести окно примерно к такому виду:

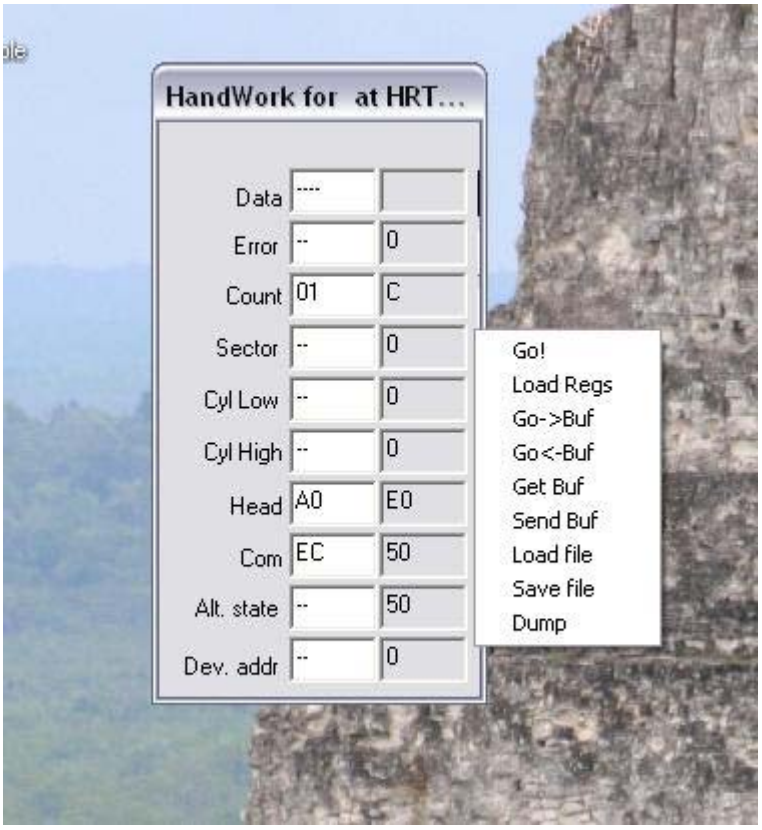


Рис. 18. Уменьшенное окно подачи команд с раскрытым меню, замещающим собой кнопки, ушедшие из области видимости

Тогда его можно удобно разместить, чтобы оно всё время было видно, но не мешало другим приложениям. Правда, с приходом мониторов с HD-разрешением, а также многомониторных систем, эта функциональность утратила свою актуальность. Для тех, кто до сих пор работает с разрешением 1024x768 (она была разработана в те времена), она может оказаться полезной.

Флажок **Loop Mode** – антипод флажка **Monitor Mode**. Если он установлен, программа 10 раз в секунду имитирует нажатие кнопки **Go!**. В первую очередь, это

полезно для тех накопителей, которым надо постоянно слать какую-либо константу при включении питания. Но в целом – при работе с Super командами, эта функциональность часто оказывается востребована.

Если установлен флажок **Auto Save State**, при закрытии окна содержимое всех полей будет запомнено, а при последующем открытии – восстановлено.

Теперь рассмотрим механизм добавления команд в справочник. Справочник хранится в каталоге ATA_COMM.INI. Как и все прочие INI-файлы, этот файл копируется в каталог проекта. Поэтому если Вы правили справочник, результаты правки необходимо затем перенести в глобальный INI файл.

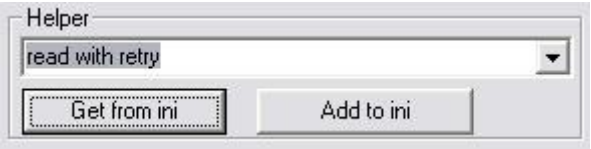


Рис. 19. Область справочника

Для того чтобы взять команду из справочника, необходимо раскрыть список, найти в нём требуемую строку (поиск по первым символам обеспечивается ОС Windows), после чего нажать кнопку Get from ini.

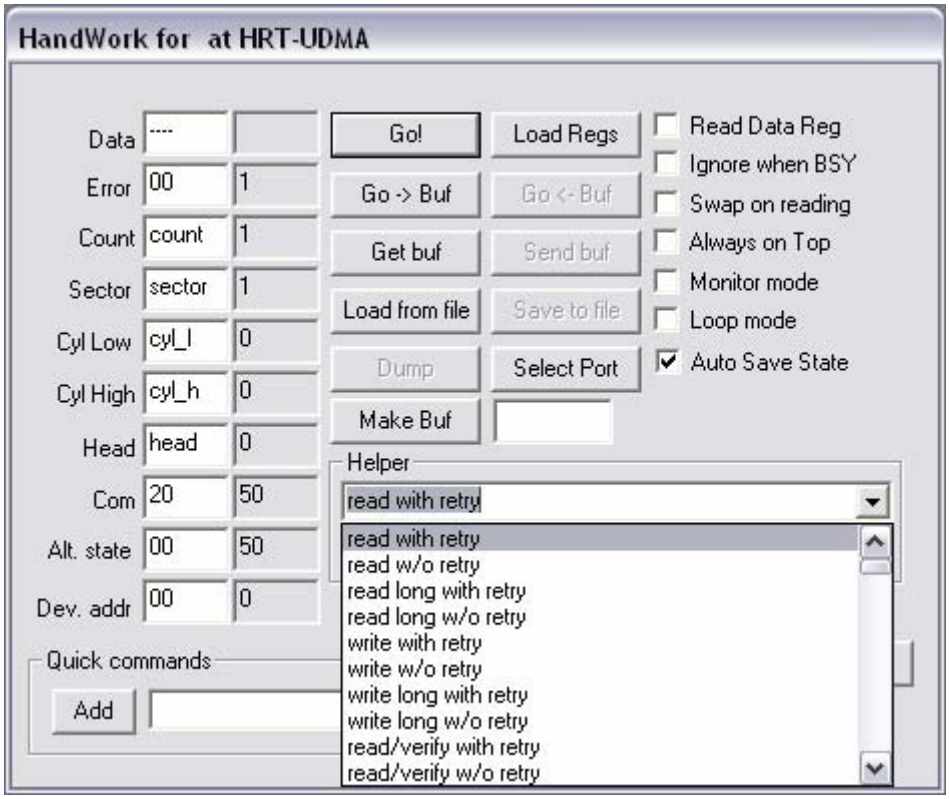


Рис. 20. Пример раскрытого списка

Обычно в справочнике часть полей имеет символическое обозначение. Это подсказки о заполнении полей. Передавать в накопитель в таком виде их нельзя, накопитель понимает только числа. Поэтому все подсказки необходимо заменить числовыми значениями, только после этого нажимать кнопки Go!, Go->Buf или Go<-Buf.

Чтобы добавить запись в справочник, впишите её имя и нажмите Add to ini.

Теперь, когда мы рассмотрели основы работы с диалогом ручной подачи команд, перейдём к более сложным случаям. Первый из них – подача команды в формате LBA48. В целом, при этом в каждый порт (кроме некоторых) необходимо записать не одно, а два значения. Эти значения просто следует записать через запятую. Например, для чтения сектора с LBA=0x12345678, следует заполнить поля следующим образом:

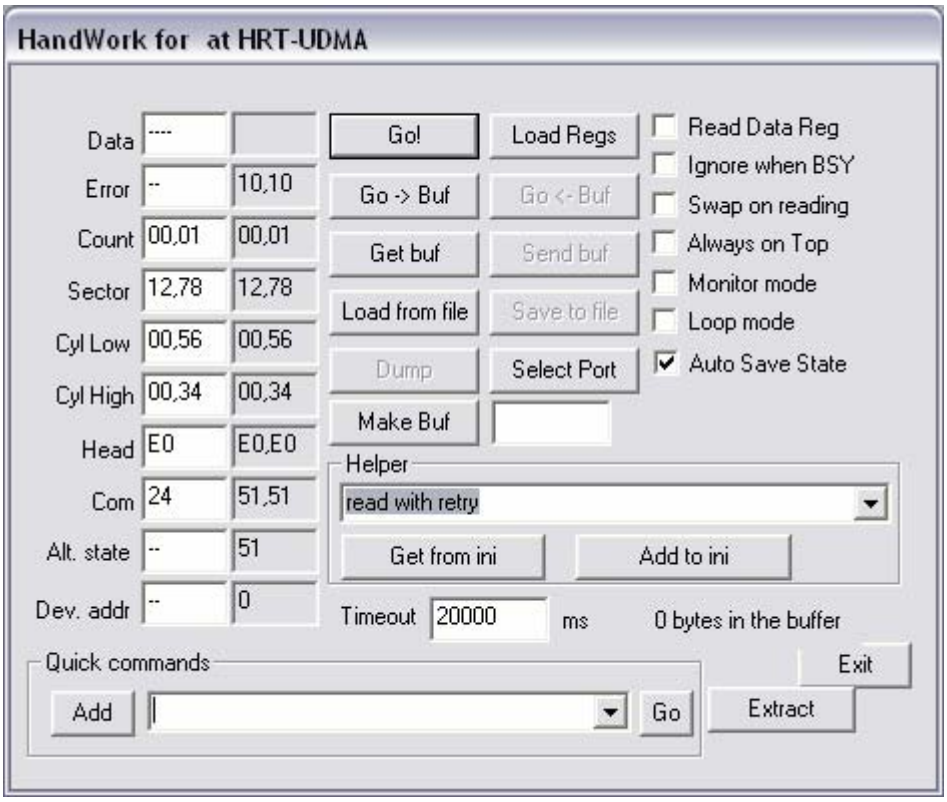


Рис. 21. Заполнение полей для чтения сектора с LBA=0x12345678

Если хоть в одном из полей есть запятая, кнопка Load Regs также будет считывать ответы в формате LBA48.

И, наконец, рассмотрим случай, когда необходимо подавать несколько команд. Заполнение полей – задача не из лёгких. Если надо оперативно переправить все поля – времени уйдёт много. А если через 10 секунд потребуется вернуть всё в исходное положение? В принципе, никто не мешает открыть несколько диалогов подачи команд, так как утилита позволяет сделать это, так как кнопка ATA Command при открытии окна ATA команд, не блокируется (см. Рис. 22).

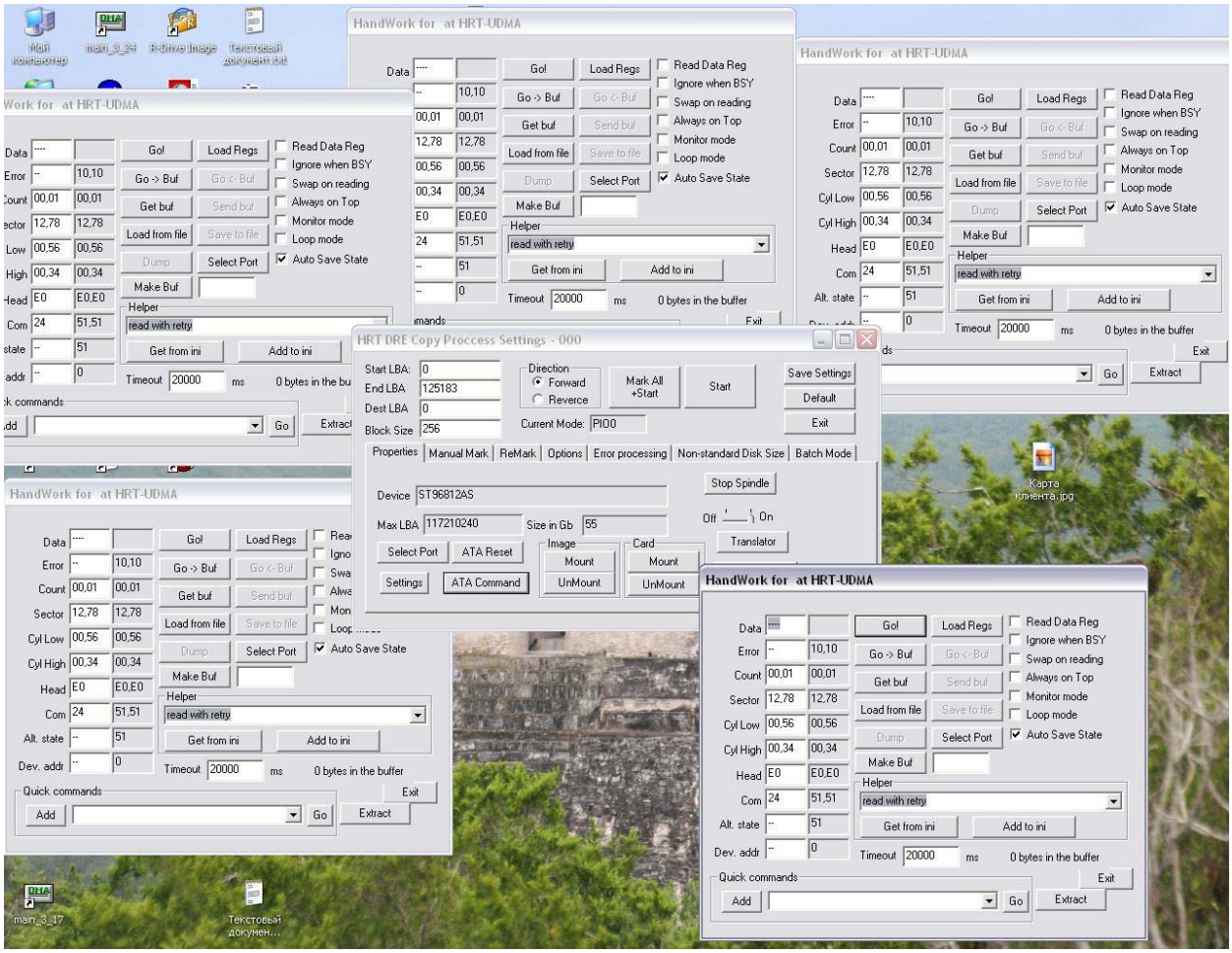


Рис. 22. Весь рабочий стол покрыт диалогами подачи АТА-команд

Но и в одиночном окне подачи АТА команд имеется механизм подачи множественных команд, расположенный в группе Quick commands.

1) Введите команду в основных полях
2) Нажмите Add в группе Quick Commands. В результате, команда будет добавлена в список.
3) Аналогично добавьте в список другие необходимые команды

Теперь возможны два сценария. Выберите команду в списке и либо нажмите Go в группе Quick Commands (в этом случае, команда уйдёт немедленно), либо нажмите Extract, и команда будет извлечена в основные поля. При этом, выбранную команду из списка, можно отредактировать прямо в той же строке, где она отображается.

Окно дампа

Теперь рассмотрим окно дампа, используемое во всех утилитах HRT. Оно позволяет просматривать и редактировать данные, находящиеся в буфере. Внешний вид окна приведён на Рис. 23.

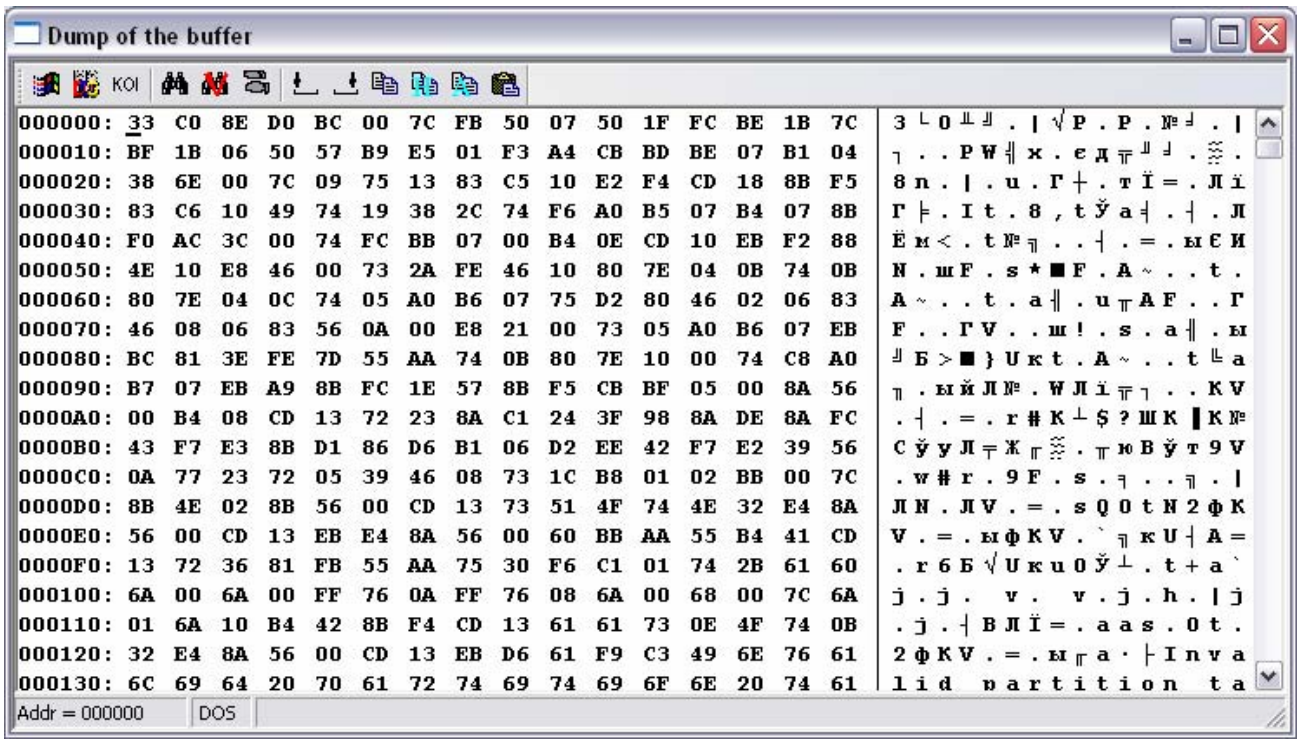


Рис. 23. Пример окна дампа

Поля дампа стандартные. Слева располагается адрес, далее – шестнадцатеричное содержимое буфера, справа – ASCII расшифровка. Если дамп имеет размер более чем 1 сектор, на границе секторов (0x200 байт) цвет шрифта будет изменяться для подчёркивания перехода через границу:

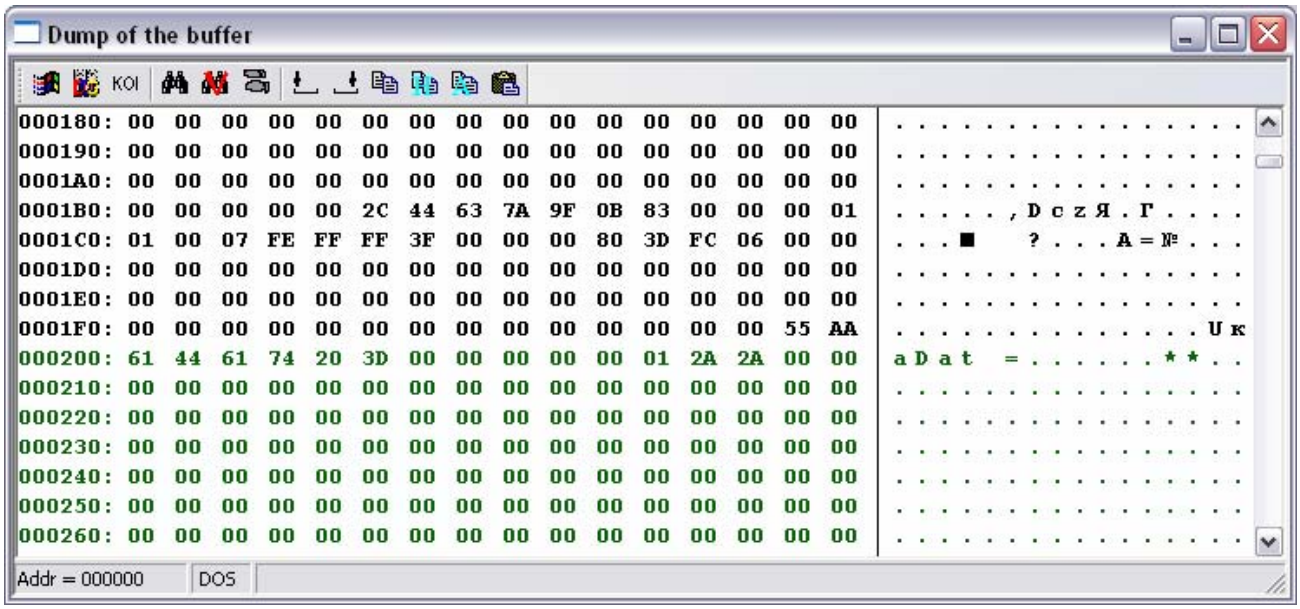


Рис. 24. Пример смены цвета шрифта при переходе через границу сектора

Установив курсор на шестнадцатеричное или текстовое поле, Вы можете редактировать их. Установку курсора можно выполнять при помощи «Мыши», а быстро переключаться между полями – при помощи клавиши Tab. Адрес, соответствующий положению курсора, отображается в левом нижнем углу окна. Там же отображается текущая кодировка русских символов, используемая при отображении текстового поля.

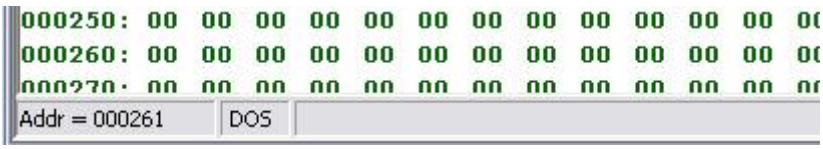
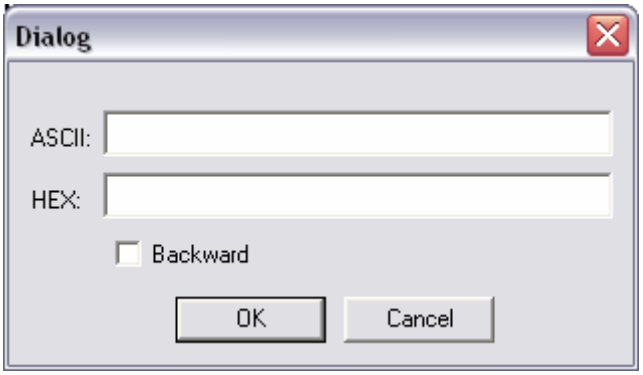


Рис. 25. Левый нижний угол окна

Рассмотрим теперь панель инструментов окна дампа.

- группа выбора типа кодировки (WINDOWS 1251, DOS 866 или KOI-8)


- поиск в дампе. При этом будет выдан следующий запрос модели поиска:



Вы можете ввести модель либо в ASCII, либо в HEX строке (по мере ввода в одной строке, другая будет отображать альтернативное представление введённых данных и Вы можете при помощи клавиш Tab и Shift+Tab перейти к его редактированию), после чего нажать кнопку OK. Если флажок Backward снят, поиск будет вестись вниз, начиная с позиции, справа от курсора. Если флажок установлен, поиск будет вестись вверх, с позиции слева от курсора.


- повторный поиск по той же модели


- переход по заданному адресу


- отметить позицию курсора, как начало блока


- отметить позицию курсора, как конец блока

Пример выделенного блока показан на Рис. 26.

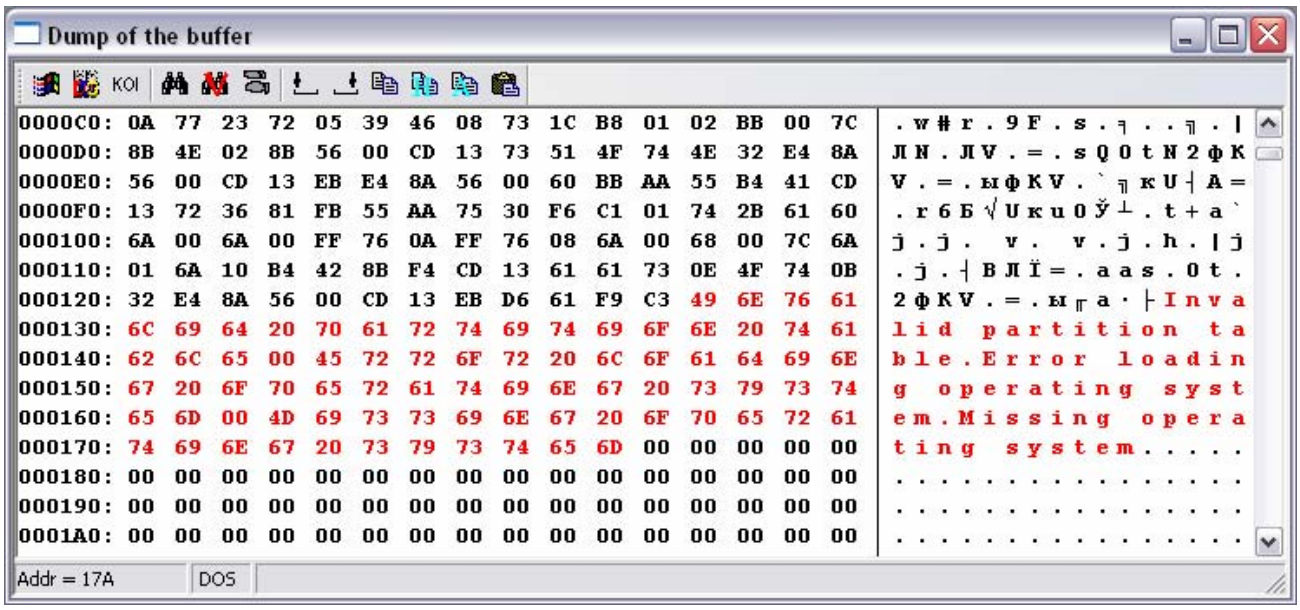


Рис. 26. Пример выделенного блока дампа

- Взять выделенный фрагмент дампа в буфер обмена для последующего копирования в текстовый редактор, либо в другой участок дампа. Возьмём ранее выделенный участок в буфер обмена и вставим в данный документ:

```
00000120:                                49 6E 76 61          Inva
00000130: 6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 lid partition ta
00000140: 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E ble.Error loadin
00000150: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating syst
00000160: 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 em.Missing opera
00000170: 74 69 6E 67 20 73 79 73 74 65 6D               ting system
```

Заменим шрифт на пропорциональный:

```
00000120:                                        49 6E 76 61          Inva
00000130: 6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 lid partition
ta
00000140: 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E ble.Error
loadin
00000150: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating
syst
00000160: 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 em.Missing
opera
00000170: 74 69 6E 67 20 73 79 73 74 65 6D                ting system
```

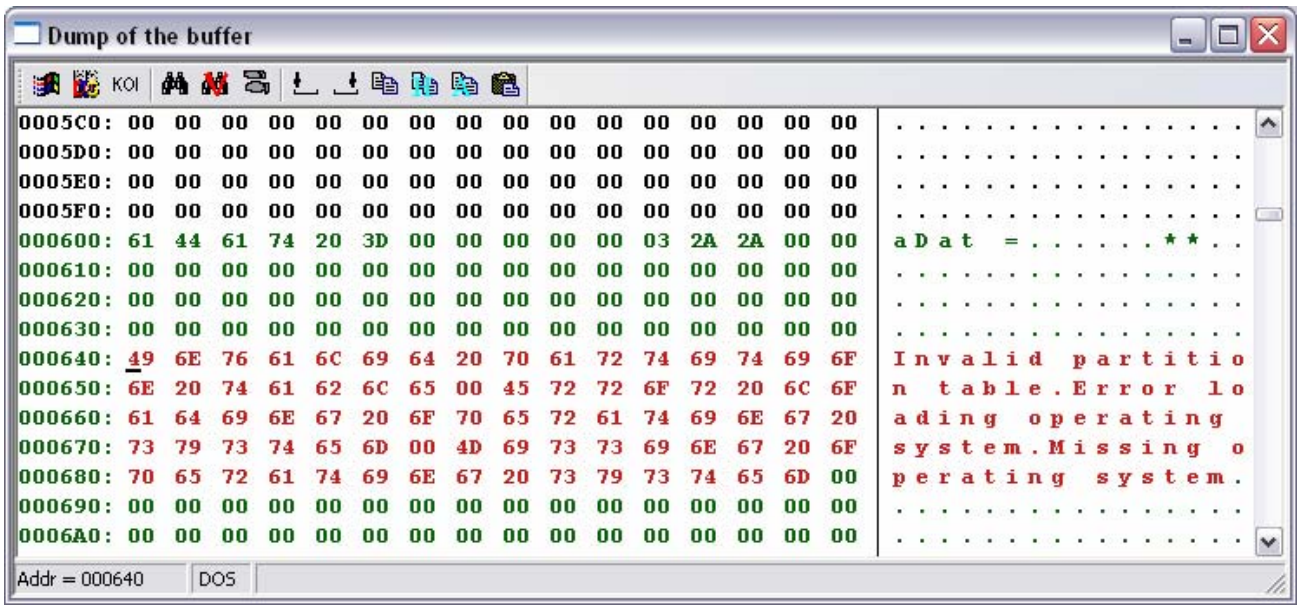
- Взять шестнадцатеричный дамп выделенного фрагмента в буфер обмена. Выполним эту операцию для того же фрагмента дампа:

```
49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 62 6C 65 00
45 72 72 6F 72 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79
73 74 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73
74 65 6D
```

- Взять текстовую расшифровку выделенного фрагмента в буфер обмена. Для уже известного фрагмента получим:

```
Invalid partition table.Error loading operating system.Missing operating system
```

- Вставить дамп из буфера обмена в буфер, начиная с позиции курсора. Дамп должен быть взят первым способом ( ). Ниже приведён пример вставки уже хорошо знакомого фрагмента на адрес 0x640.



Теперь рассмотрим контекстное меню окна дампа.

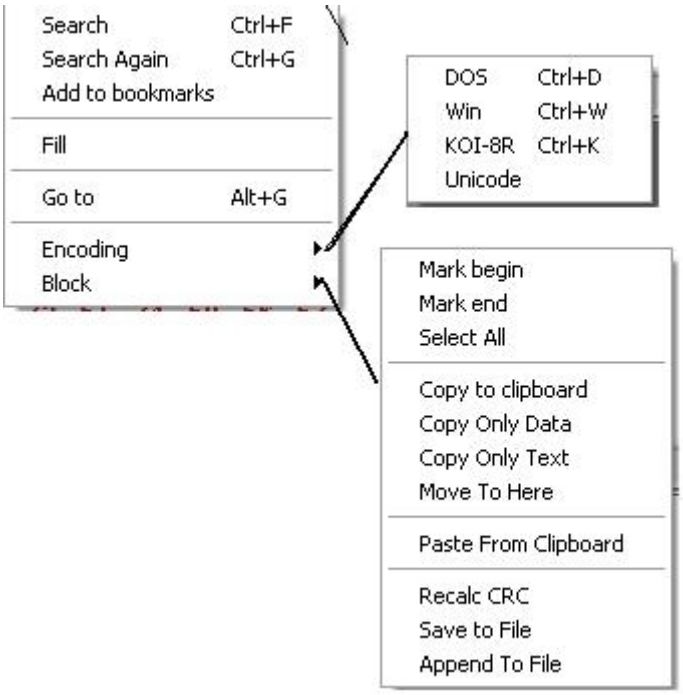
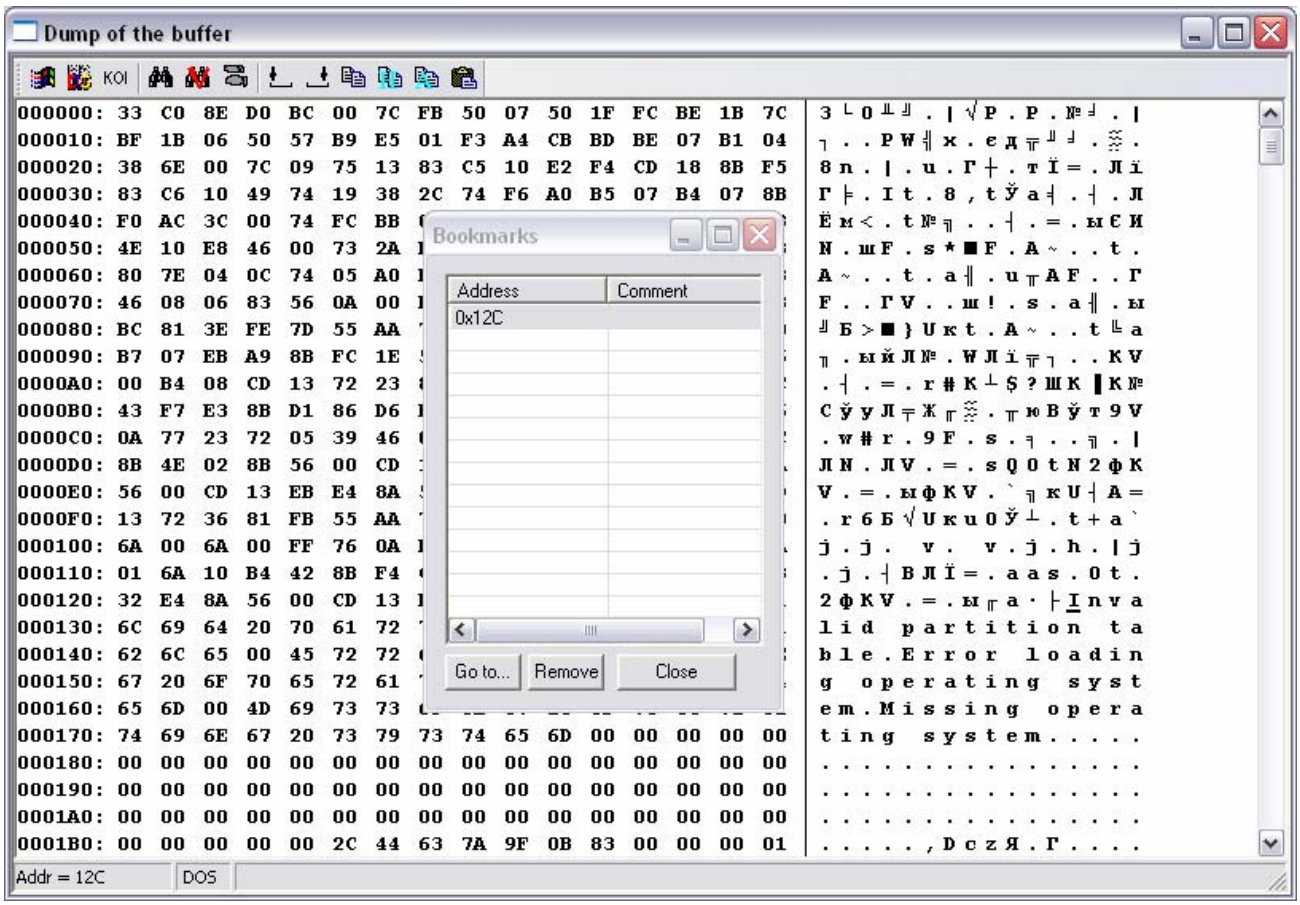


Рис. 27. Меню окна дампа

Пункты **Search** и **Search Again** рассмотрены выше при рассмотрении панели инструментов.

Пункт **Add to bookmarks** добавляет текущий адрес к закладкам. При её первом нажатии, открывается дополнительное окно:



Это окно будет находиться поверх других окон. Все закладки будут утрачены при закрытии окна закладок, поэтому оставляйте его до тех пор, пока закладки Вам нужны. Для любой закладки можно вставить комментарий. Для этого «щелкните» «мышью» по соответствующему полю Comments и введите текст комментария:

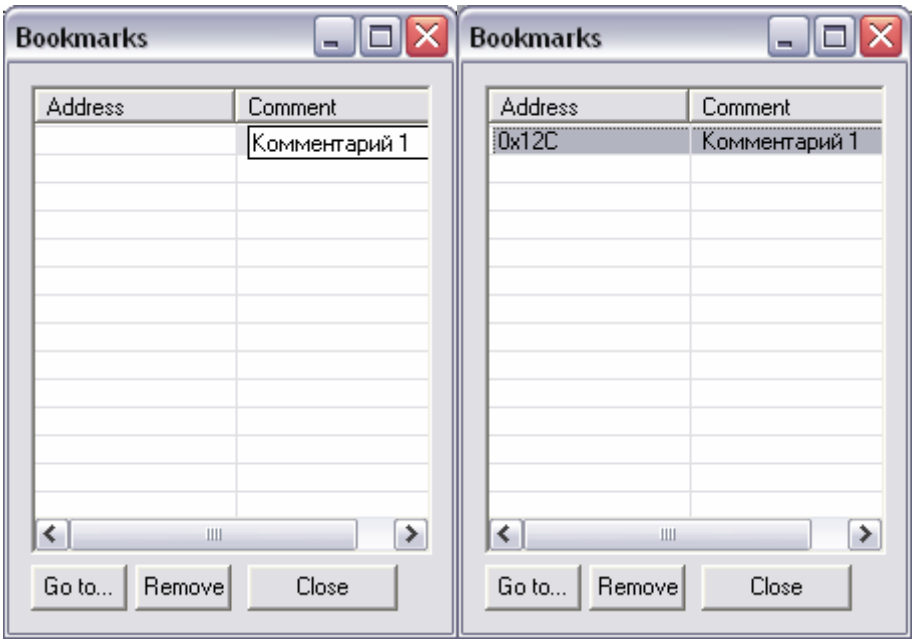
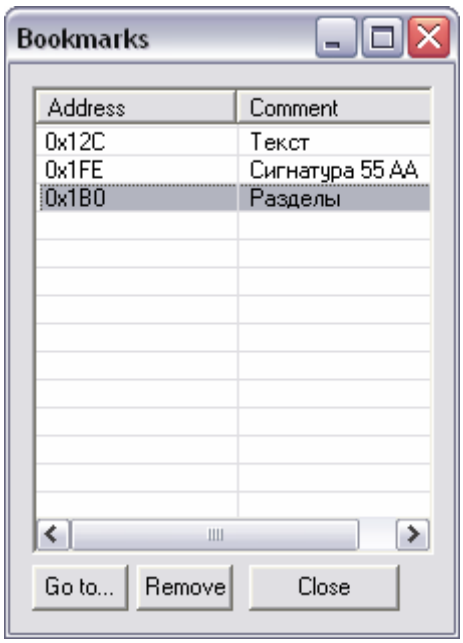


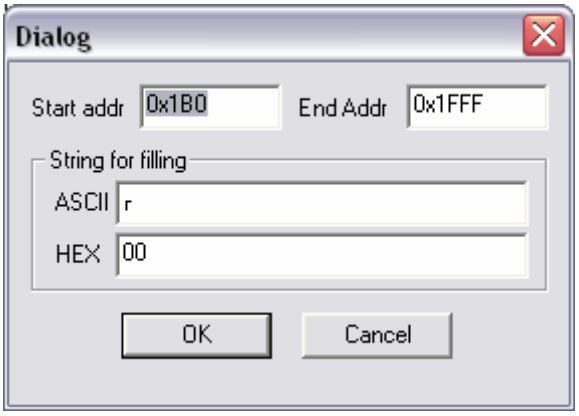
Рис. 28. Комментарий во время ввода (слева) и после ввода

По мере работы с дампом, Вы можете выбирать пункт add to bookmarks вновь и вновь, новые закладки будут добавляться в список.

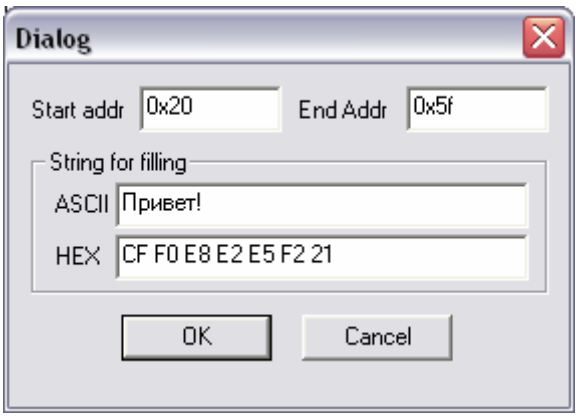


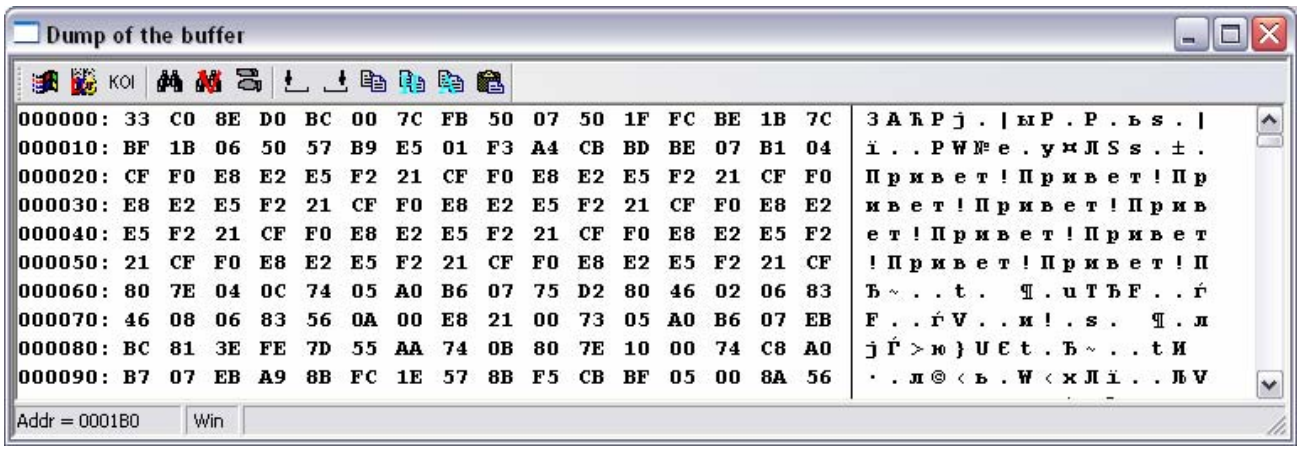
Выбрав строку списка и нажав кнопку Go to… Вы спозиционируете дамп на адрес, соответствующий закладке.

Продолжим рассмотрение пунктов меню окна дампа. Пункт **Fill** открывает окно заполнения дампа. По умолчанию, поля заполнены так, чтобы занулить участок от курсора до конца дампа.



Строку-заполнитель, как и строку поиска, можно заполнять как в ASCII, так и в HEX формате. Рассмотрим пример заполнения дампа с адреса 0x20 до адреса 0x5f строкой «Привет!»:





Пункт **Go To** позволяет перейти по заданному адресу в буфере.

Пункты группы Encoding уже рассмотрены при описании строки инструментов. Единственное дополнение – возможность переключить отображение ASCII поля на формат **UNICODE**.

Пункты **Block->Mark Begin** и **Block->Mark End** соответствуют выделителям на панели инструментов, а пункт Block->Mark All позволяет выделить весь буфер.

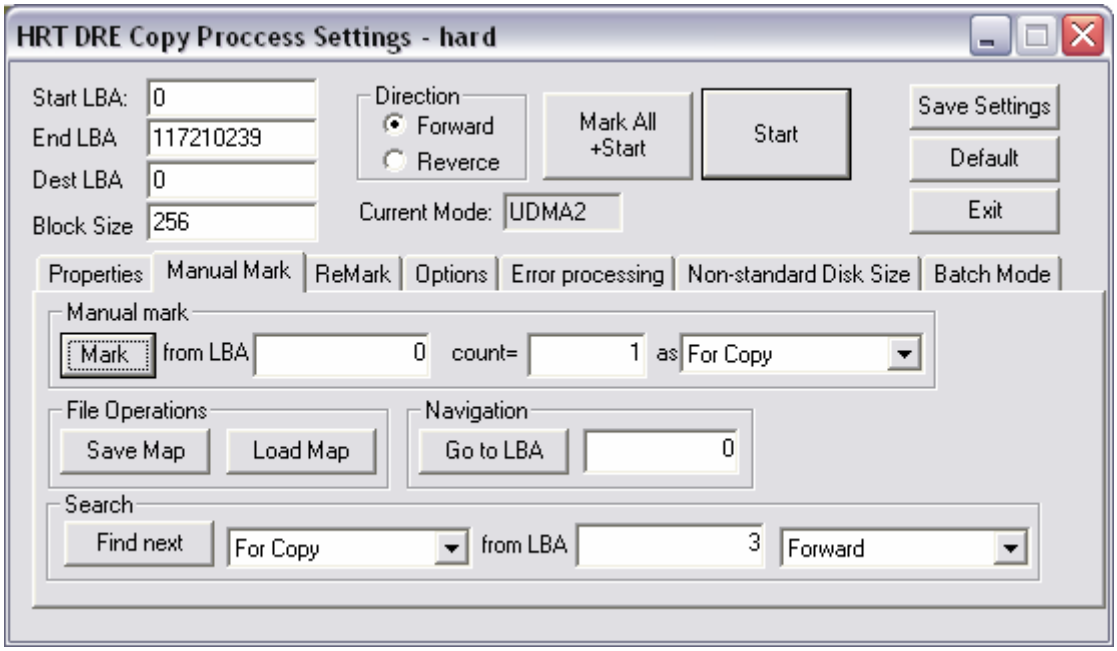
Пункты **Block->Copy to clipboard**, **Block->Copy only data**, **Block->Copy Only Text** и **Block->Paste From Clipboard** имеют аналоги в панели инструментов.

Пункт **Block->Move To Here** копирует выделенный фрагмент буфера в область, начиная с позиции курсора.

Пункт **Block->Recalc CRC** имеет смысл в специализированных утилитах. При его выборе, производится расчёт контрольной суммы выбранного участка по правилам, характерным для текущего накопителя. В универсальной утилите (к каким относится DRE) это невозможно.

Пункт **Block->Save To File** сохраняет выбранный участок дампа в файл, а **Block->Append to File** дописывает выбранный участок буфера в конец файла.

Вкладка Manual Mark



Данная вкладка позволяет перемаркировывать карту, а также – сохранять её и загружать другие карты. В целом, перемаркировка карты относится к тем временам, когда у неё не было флажков, а копировались только сектора типа Marked to Copy. Поэтому перед каждым проходом приходилось перемаркировывать те или иные результаты в Marked to Copy и только потом запускать процесс. В те времена перемаркировка карты была жизненно важной функцией. После изменения идеологии, эта операция стала сугубо вспомогательной, и авторы даже подумывали о её изъятии из программы. Однако, при первой же попытке, бета тестеры просто взвыли. Оказалось, что они уже привыкли к тому, что можно что-то перемаркировать. Так что как впомогательная операция, перемаркировка всё же оказалась полезной. Так что рассмотрим, какие возможности перемаркировки предоставляет утилита.

В первую очередь, это марикрование цепочки секторов в определённый тип. Необходимо выставить начальный LBA и количество в соответствующие поля группы Manual Mark, выставить желаемый тип, после чего нажать Mark. Тип For Copy выбран по умолчанию не случайно. До появления кнопки Mark All + Start карта метилась этим путём.

Группа File Operations позволяет сохранить карту под именем, отличным от стандартного, либо загрузить нестандартную карту. В целом, это было задумано для переноса карт с машины на машину, но после появления понятия «проект», стало проще копировать каталог проекта.

Группа Navigation позволяет перейти к выбранному LBA в карте. Важно помнить, что этот LBA будет спозиционирован не в левый верхний угол карты, а в некую окрестность, строго приходящуюся на отображаемый участок. Попробуем перейти на LBA=123456. Результат представлен на Рис. 29. Как видно, квадратик находится в верхней строке, но не строго в её начале.

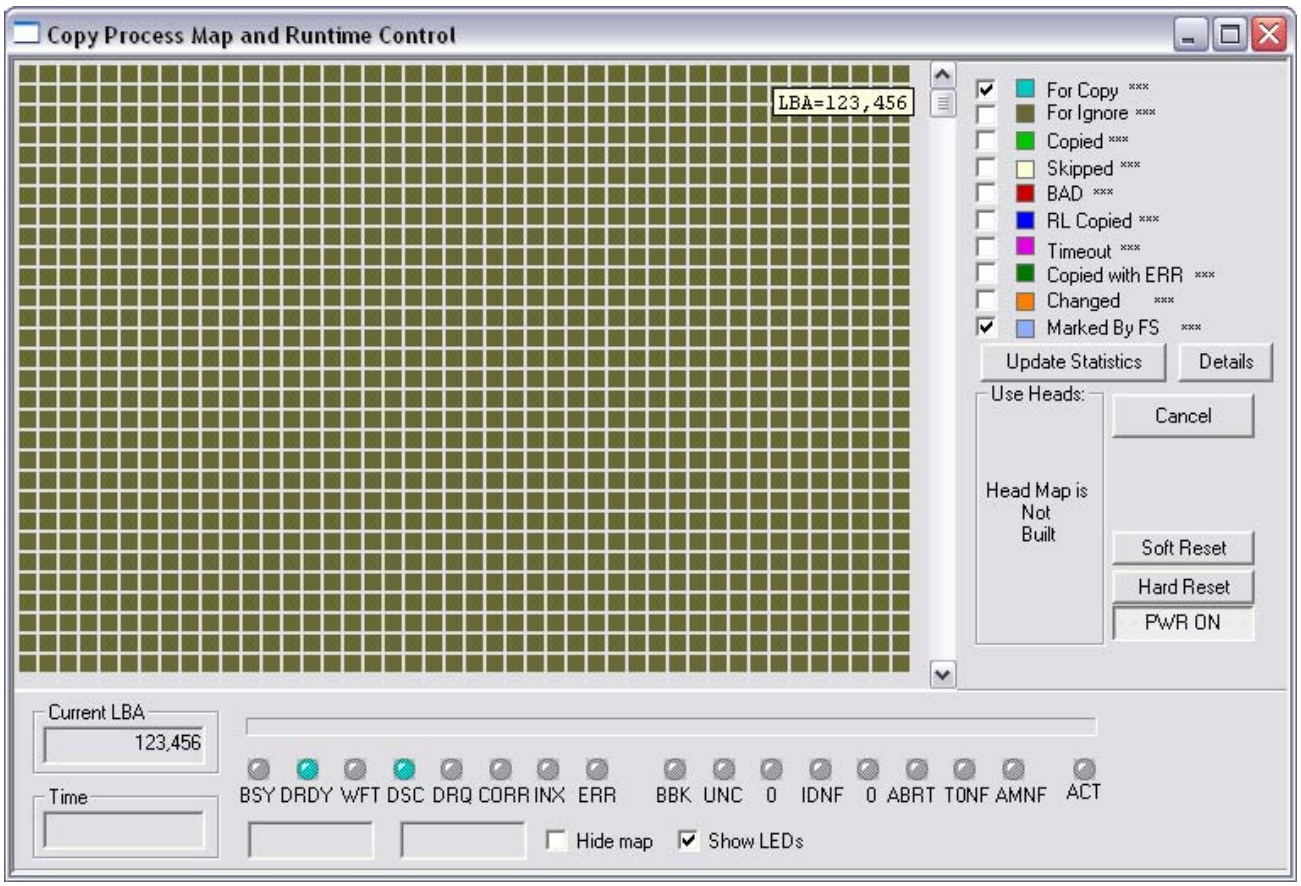


Рис. 29. Произведён переход на LBA123456. Соответствующий квадратик в верхней строке, но не в её начале

Группа Search необходима для поиска того или иного типа секторов в карте. Искать можно вверх или вниз.

Маленькая хитрость: Вообще-то экспресс-поиск есть и в окне карты. Как уже отмечалось, если в правой её части щёлкнуть на цветной квадратик, откроется окно редактирования его цвета. Если же щёлкнуть по поясняющему тексту (например, по слову BAD), то будет произведён поиск ближайшего квадратика этого типа, размещённого ниже области, отображаемой в окне. Это удобно для поиска ближайшей BAD-области, ближайшего пропуска и т.п.

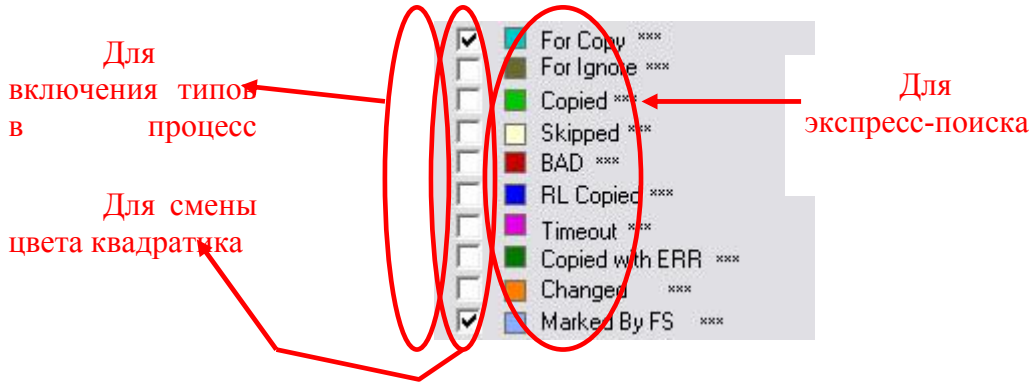
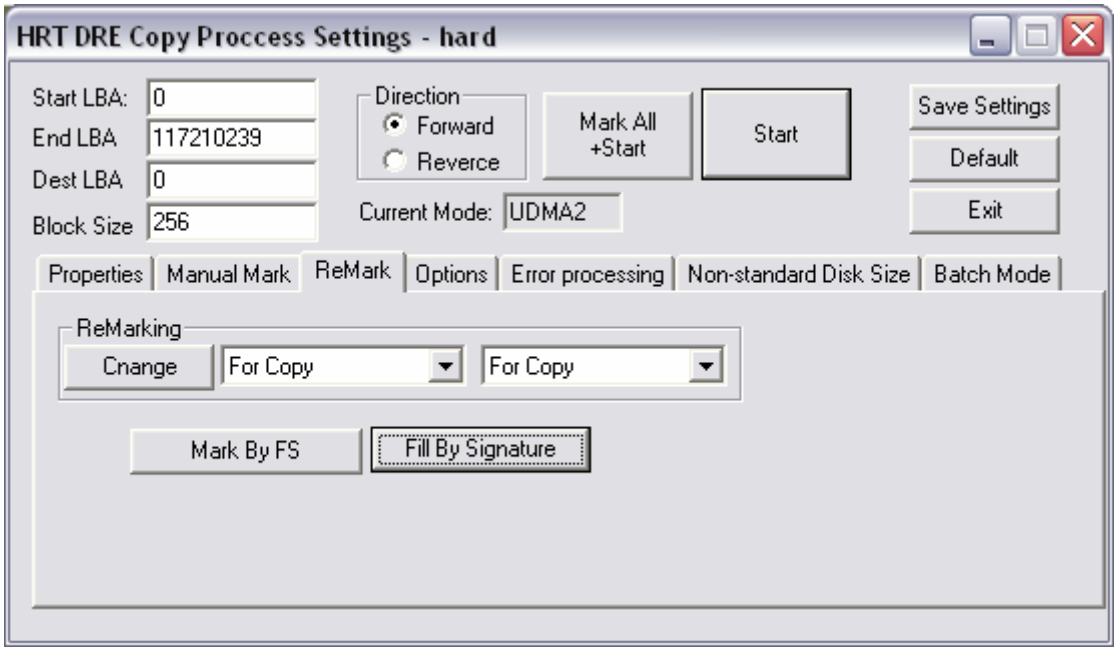


Рис. 30. Щёлкните по элементу области для...

Вкладка Manual ReMark



Ранее данная вкладка была единой со вкладкой Manual Mark, но в далоьнейшем, отпочковалась от неё, чтобы подчеркнуть, что Mark – метит, а ReMark – изменяет пометки. Поэтому группа ReMarking имеет чисто историческую ценность.

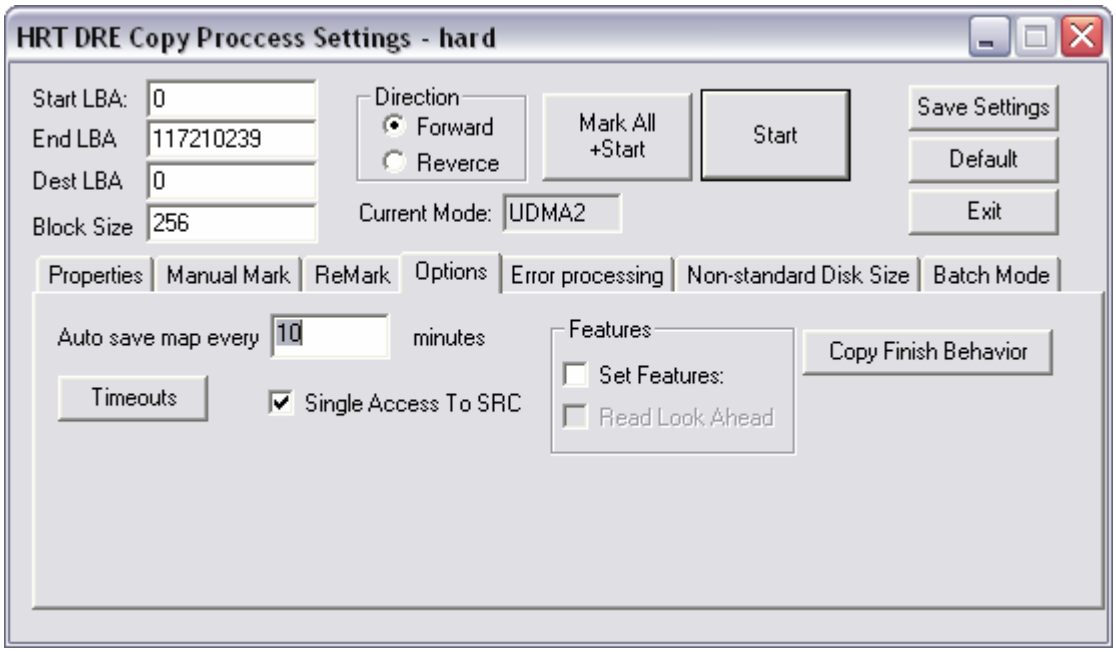
Что же касается кнопок Mark By FS и Fill By Signature, то они вполне действующие. Кнопку Mark By FS рассмотрим в отдельном разделе, так как она заслуживает отдельной главы.

Кнопка **Fill By Signature** введена по просьбе представителей фирм, имеющих традиции, уходящие корнями глубоко в прошлое. Эти фирмы пользуются программами, разработанными задолго до HRT DRE, поэтому ориентирующимися не на карту, а на какие-то специальные метки, хотя на самом деле, карта – это всё. Это и задание на работу, это и протокол работы.

Этим фирмам необходимо метить определённые сектора некоей сложной сигнатурой. В частности, BAD-блоки они метят более сложной меткой, чем простой «BAD!BAD!BAD!...». Чтобы решить их проблемы, была добавлена функциональность заполнения сигнатурой любой сложности секторов, имеющих ту или иную отметку в карте. При нажатии на кноку Fill By Signature, откроется следующее окно:

Галочки задают тип секторов, которые следует запонить сигнатурой. Саму сигнатуру можно либо вбить вручную (кнопка Dump), либо загрузить из файла (кнопка Load). После чего нажать кнопку Go. Утилита запишет во все сектора выбранного типа заданную сигнатуру.

Вкладка Options



На данной вкладке сосредоточены элементы управления, задающие режимы работы утилиты.

Поле Auto Save map every XXX minutes задаёт период автоматического сохранения карты. В принципе, в нынешних версиях карта реализована средствами проецирования файла в память, поэтому процесс сохранения её происходит чрезвычайно быстро. Так что даже сохраняя её каждую минуту, Вы не сильно замедлите процесс копирования. Однажды в одной версии программы была допущена ошибка, и карта сохранялась непрерывно. Только сохранилась – сразу начинает сохраняться вновь. Пользователи многоядерных машин не заметили подвоха. Ошибку нашли пользователи одноядерных процессоров. Там скорость копирования падала существенно, но тоже не до нуля.

Кнопку Timeouts мы рассмотрим ниже.

Флажок **Single Access to SRC** снимать не рекомендуется. Пока он стоит, утилита производит не более одного успешного обращения к каждому сектору источника. Если сектор считался успешно (будь то для просмотра дампа, для файлового разбора или ещё для какой цели), он тут же попадёт на приёмник. И при следующих обращениях к этому сектору, данные уже будут браться из копии. Неудобство возникает в одном случае: Допустим, Вы копируете данные в файл-образ без флажка **Sparse File**. И захотели посмотреть содержимое дампа последнего сектора диска. И всё. Система, как уже было описано выше, «зависнет», так как ОС Windows начнёт заполнять файл-образ нулями от начала до конца (по 2-4 гигабайта в минуту). Но если копирование ведётся на диск, либо в файл с атрибутом sparse, такой проблемы не возникнет. Но иногда из-за этой проблемы

приходится временно снимать флажок **Single Access to SRC**. Но делать это следует исключительно в крайнем случае.

Группа Features позволяет управлять режимом работы накопителя. Если флажок Set Features снят, управление кэшированием накопителя не ведётся. Если же он установлен, можно или включить или выключить упреждающее чтение (установив или сняв флажок Read Look Ahead). Главное назначение этого флажка в следующем: Если из работы исключена неисправная головка, но включено опережающее кэширование, накопитель всё равно будет на неё «забегать», кэшируя данные. Разумеется, при этом он начнёт «тормозить». Так что если Вы копируете с исключением головок – отключайте упреждающее кэширование.

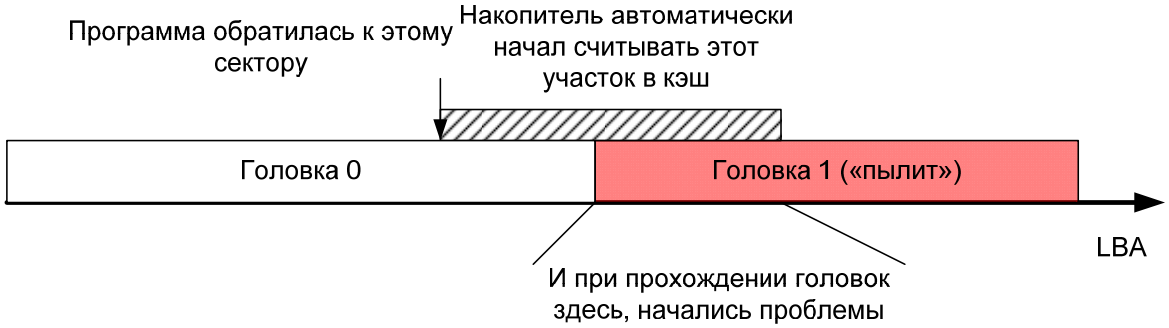
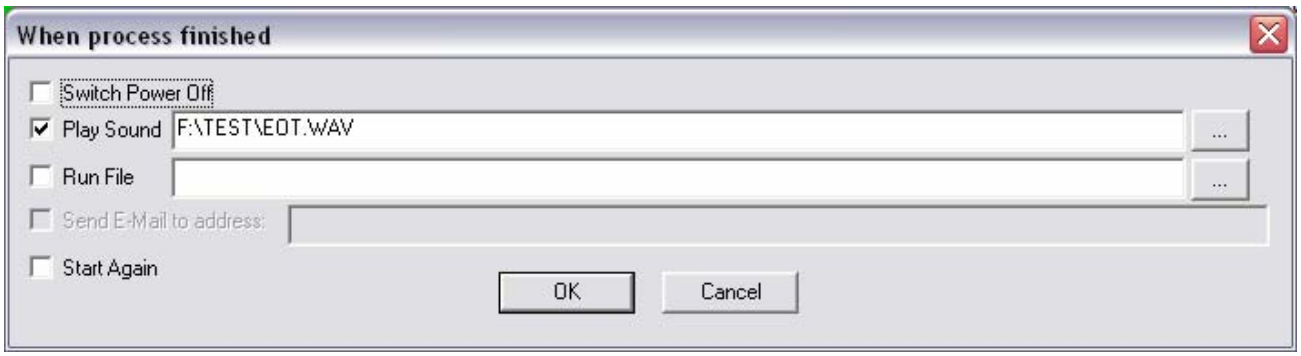


Рис. 31. Пояснение проблем, возникающих при упреждающем кэшировании с неисправной головкой

Флажок Set Features	Флажок Read Look Ahead	Режим
Снят	Снят	Не изменяется. Берётся значение накопителя по умолчанию
Снят	Установлен	
Установлен	Снят	Отключён
Установлен	Установлен	Включён

Кнопка Copy Finished Behavior определяет поведение утилиты при завершении копирования. В первую очередь, это важно для процесса копирования, оставленного на ночь. Представьте себе, накопитель «пылит», а процесс копирования завершился в час ночи. И он будет «пылить» с часу ночи до девяти утра, когда Вы придетё на работу. Поэтому в ранних версиях утилиты по окончании копирования автоматически выключалось питание накопителя. Но в этом случае, возникает новая проблема. Допустим, Вы рядом. И копируете накопитель, запущенный при помощи процесса Hot Swap. Кончился проход тестирования. Пора запустить новый. Но вот незадача – утилита взяла и вероломно отключила питание накопителя. Придётся снова выполнять Hot Swap. Налицо противоречие.

Чтобы разрешить его, был добавлен пункт для настройки поведения утилиты по завершении. А раз уж добавлена возможность щелчка питанием, было решено увеличить перечень возможностей.



Флажок Switch Power Off настраивает утилиту на режим выключения питания по окончании копирования.

Если установлен флажок Play Sound, то будет проигрываться выбранный WAV файл. По умолчанию используется мелодия, написанная для комплекса HRT (EOT.WAV).

Для более сложных сценариев, Вы можете разработать свою программу (она может посылать E-Mail, посылать SMS, выключать питание компьютера, включать приготовление кофе и т.п.). Установив флажок Run File и прописав имя файла, Вы настроите утилиту на запуск этой программы по окончании копирования.

Установив флажок Start Again, Вы сообщите утилите, что необходимо запустить следующую итерацию при тех же настройках копирования. Это может пригодиться, например, для вычитывания пропущенных прыжком секторов, ведь обращения к ним ещё не было. В остальных случаях, лучше пользоваться более интеллектуальным методом Batch Mode, описанным ниже.

Маленькая хитрость. Если Вы хотите изменить настройку этого окна по умолчанию, Вам необходимо отредактировать секцию [Copy Finish] файла hdd.ini, расположенного в том же каталоге, где и исполняемый файл. Именно этот файл копируется в каталоги вновь создаваемых проектов, а данная секция используется, как основа для настроек «по умолчанию» для завершения процесса.

Настройка таймаутов

Разговор о настройке таймаутов настолько важен, что он вынесен в отдельный раздел. Для вызова настроек таймаутов, необходимо нажать кнопку Timeouts вкладки Options. В результате, утилита выдаст окно, показанное на Рис. 32.

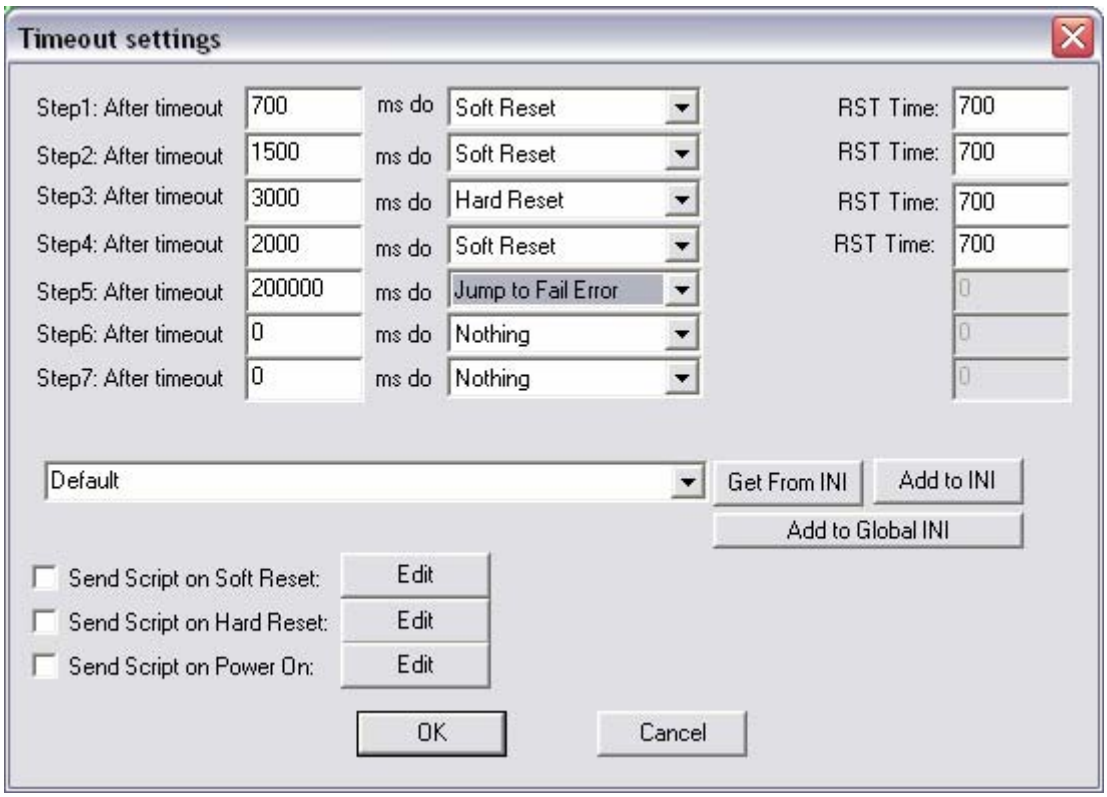


Рис. 32. Диалог настройки таймаутов

Чтобы пояснить, что такое таймаут, следует рассказать, как выполняются команды накопителем.

У накопителя имеется регистры, в которые вписываются параметры (например, LBA), а затем – подаётся команда (например, команда чтения). При этом накопитель взводит состояние «Занято» (бит BSY среди светодиодов). Пока накопитель занят, он не готов воспринимать ни новые параметры, ни новые команды. Если запрошенные сектора считались быстро, накопитель быстро снимет бит BSY и будет готов к следующей команде. Если же встретился BAD Сектор, накопитель будет по каким-то внутренним алгоритмам пытаться вычитать его. При этом будет всё время взведён бит BSY и накопитель не будет воспринимать новых команд. Никак.

Можно дождаться, когда накопитель снимет BSY. Но это иногда может занимать десятки секунд или даже минуты. А можно – попытаться сбросить накопитель. Для этого имеются разные средства:

Средство	Пояснение
Программный сброс	В регистр альтернативного статуса посылается константа 4, что говорит накопителю, что надо сбросить процессор. Длительность установленного бита иногда также играет роль. Обычно достаточно держать его установленным в течение 700 мс, но некоторые пользователи с пеной у рта доказывают, что существуют другие «волшебные» значения. Авторы не верят в такие значения, но не хотят связывать пользователя, поэтому дают возможность производить настройки. После сброса, накопитель может выйти в готовность, а может – и не выйти (если по какой-то причине процессор не видит соответствующий бит порта). Самый «мягкий» тип сброса.
Аппаратный	В шине IDE имеется специальная линия, которая вызывает

сброс	сброс процессора. В отличие от бита в порту, она работает более жёстко. Хотя, если накопитель потерял сервометки, то даже такой тип сброса может не помочь. Это более жёсткий сброс, при нём могут быть утеряны те или иные настройки. Поэтому он рекомендуется только если не помог программный сброс
Щелчок питания	Самое действенное средство сброса «стучащего» накопителя. Однако, оно совершенно неприемлемо, если накопитель был запущен через операцию Hot Swap. Кроме того, сброс выполняется от долей секунды до двух секунд, а щелчок питанием – десятки секунд. Поэтому рекомендуется только как самое «бронебойное» средство разрешения таймаута на накопителе, который умеет выходить в готовность без Hot Swapa.

Итак. Мы выяснили, что если операция выполняется слишком долго, её завершение можно попытаться форсировать. Но из приведённых выше описаний видно, что попытка форсирования выхода из состояния BSY может завершиться неудачно. Да, мы увидели длительный BSY в ответ на команду чтения, мы подали программный сброс, но BSY как был, так и остался. А если он остался, то всё равно накопитель не готов принимать от нас новые аргументы и команды. Надо пробовать применять более жёсткий метод форсирования снятия BSY. Поэтому операция разрешения таймаута должна быть многоходовой. Если помог первый шаг – замечательно, продолжаем копирование. Не помогло – идём на второй шаг. Затем на третий, на четвёртый и так далее. В первых версиях программы использовалось пять шагов и в большинстве случаев, этого хватало. Для любителей особо сложных сценариев разрешения таймаутов, число шагов было увеличено до семи.

Механизм разрешения работает следующим образом:

ЕСЛИ при чтении сектора сигнал BSY держится дольше, чем указано в параметре Step1: After Timeout, ТО выполнить действие Step 1: do

ЕСЛИ после завершения этого действия сигнал BSY пропал менее чем за период, указанный в поле Step 2: After Timeout, то считаем ситуацию разрешённой. ИНАЧЕ выполнить действие Step2: do.

И так далее. Каждая последующая строка задаёт длительность таймаута на выполнение операции из предыдущей строки. Если BSY пропал раньше, мы успешно выходим. Если не пропал – пытаемся разрешить, как указано в текущей строке и переходим к следующей.

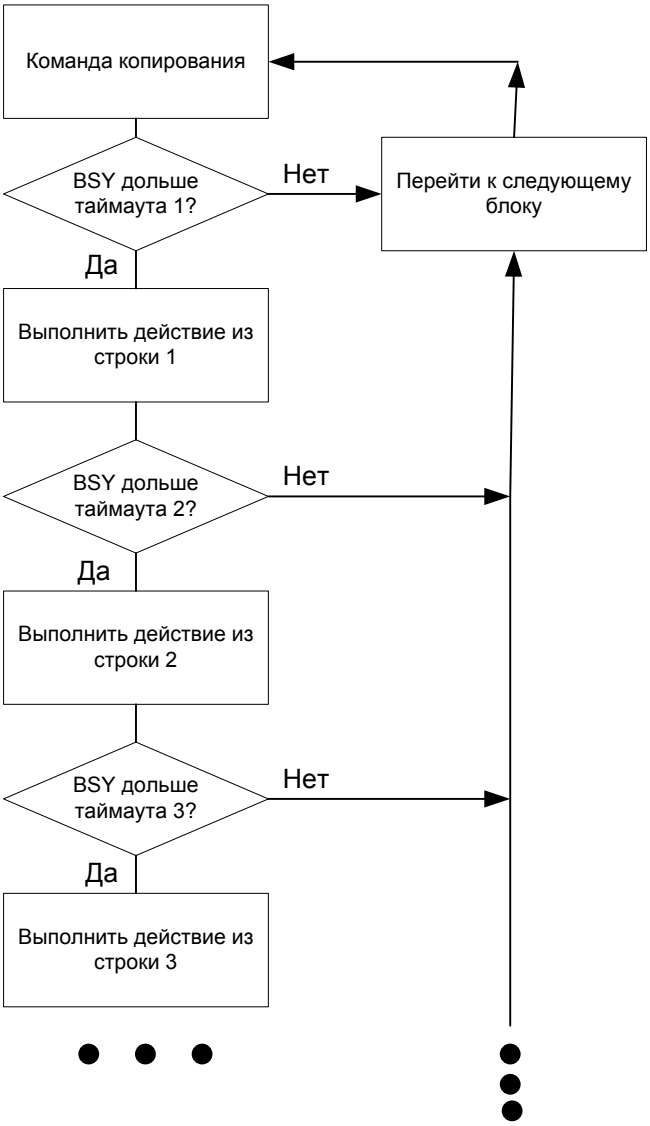


Рис. 33. Алгоритм разрешения таймаутов

Мы так тщательно рассказываем про таймауты, потому что многие многоопытные ремонтники после кратких пояснений задавали вопрос: «А почему у меня вылетает процесс копирования? Я же сказал ему подать Reset и затем – перейти к копированию следующего сектора!». Задаёшь ему вопрос: «А как же BSY?». И ответ просто поражает: «Какой ещё BSY? Пусть будет BSY, переходи к следующему блоку и всё тут». И приходится повторять, как заклинанье: **Пока стоит BSY, накопитель не воспринимает никаких команд. НИ-КА-КИХ!!!** Помогает на месяц. Затем, при следующей встрече, вопрос возникает вновь.

Так что выписываем в рамочку:

Разрешение таймаутов необходимо для того, чтобы снять сигнал BSY. Пока сигнал BSY стоит – накопитель не может воспринять ничего. Совсем ничего. Продолжение копирования НЕВОЗМОЖНО. Если сигнал BSY снять не удалось – копирование можно прерывать. Успешным разрешением таймаута считается такая ситуация, когда сигнал BSY снят.

Рассмотрим возможные варианты действий:

Действие	Пояснение	Аргумент
Soft Reset	Программный сброс (при помощи записи в порт)	RST Time – время активного сигнала сброса
Hard Reset	Аппаратный сброс (при помощи линии IDE)	RST Time – время активного сигнала сброса
Power off/on	Кратковременное выключение питания	Off Time – время, на которое следует отключить питание
Nothing	Ничего не делать. Обычно используется для временного исключения какого-либо действия из цепочки. Однако, после этого ничегонеделанья, будет произведено ожидание готовности с таймаутом из следующей строки. Так что по сути своей, таймауты текущей и следующей строки просуммируются, а при их достижении вызовется действие из следующей строки.	
Wait	Простое ожидание, во время которого не производится проверки порта BSY. Ряд ремонтников утверждает, что часто накопитель, который «долбают» на предмет чтения BSY, не выходит в готовность, а который на некоторое время оставили в покое – выходит самостоятельно	Sleep Time – время, в течение которого к накопителю не производится никаких обращений
Jump to fail error	Прекращение попыток разрешения таймаута и прерывание процесса копирования	

Для каждой ситуации подходят свои таймауты. Со временем, Вы накопите наборы таймаутов, подходящих для разных производителей и разных степеней «убитости». Для того чтобы быстро сохранять и загружать таймауты, в утилите реализован удобный механизм описанный ниже.

Рис. 34. Фрагмент окна Timeout Settings, отвечающий за сохранение и загрузку настроек таймаутов

Настройки таймаутов хранятся в файле tinings.ini. Каждая секция этого файла хранит свой набор настроек. Вы можете в любой момент сохранить или обновить набор, воспользовавшись кнопкой Add to INI (вписав или выбрав из списка имя набора).

Как уже говорилось, бывают глобальные INI файлы, а бывают локальные копии в каталоге проекта. При этом все настройки в рамках проекта сохраняются в локальный INI файл. И в новые проекты они не попадут. Однако если Вы считаете, что подобранные значения чрезвычайно важны, Вы можете воспользоваться кнопкой Add to Global INI. В этом случае, набор будет сохранён в INI файле, на основе которого делаются локальные

копии. И во всех новых проектах, этот набор будет доступен. Именно новых. Чтобы стал доступен и в старых – скопируйте глобальный файл timings.ini в соответствующий проект вручную.

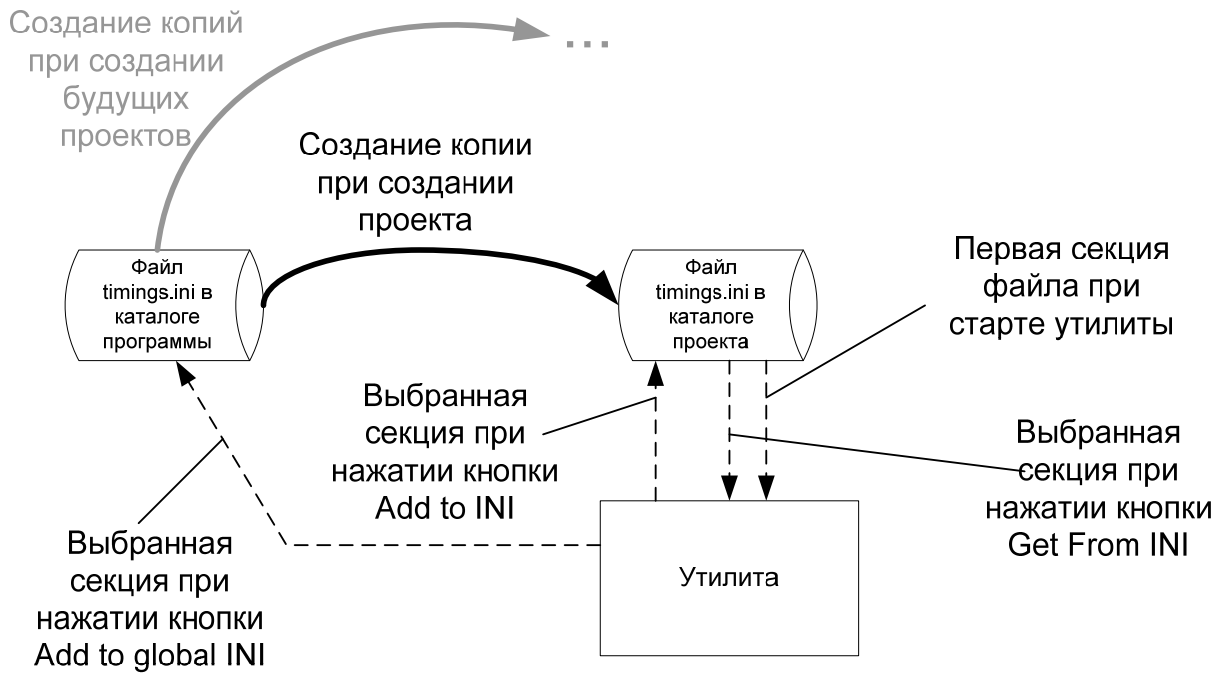


Рис. 35. Пояснение по использованию таймаутов

Скриптовая группа не обладает всей мощью, какая присутствует в специализированных утилитах HRT, но всё равно может принести ощутимую пользу.

Рис. 36. Скриптовая группа настроек таймаутов

Все ремонтники знают некие «волшебные» последовательности, после которых копирование убитых накопителей начинается намного эффективнее. Причём если посадить рядом двух ремонтников, они будут спорить до хрипоты, доказывая, что «Твоя последовательность ничего не даст, а вот моя – это сила». Но при сбросе накопителя, половина результатов этих последовательностей сбросится. А при щелчке питания – сбросятся все. Было бы хорошо их восстановить. Учитывая, что таких последовательностей существуют если не сотни, то хотя бы десятки, авторы просто добавили скриптовую поддержку в утилиту. При рассмотрении поддержки возникнет вопрос: «А зачем в скрипте щелчки питания при разруливании ситуации после щелчка питания?». Просто механизм подачи скриптов универсальный. В данном конкретном случае, интерес представляет подача команд. Всё остальное мы рассмотрим сейчас, а когда будет надо – просто дадим ссылку на данный раздел.

Скрипт редактируется в редакторе, внешний вид которого показан на Рис. 37.

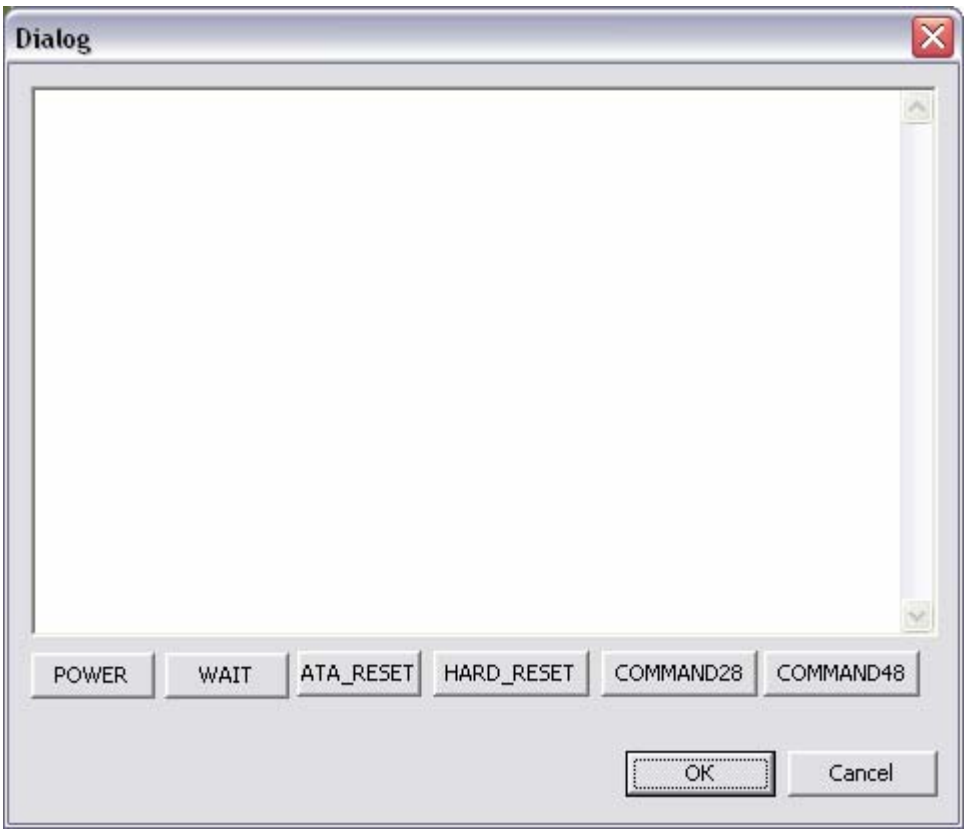


Рис. 37. Редактор скрипта

Текст скрипта пишется в поле редактирования. Его можно брать в буфер обмена вставлять из буфера обмена (сохраняя в каком-либо документе, либо получив от других ремонтников). Все возможные скриптовые команды приведены в кнопках под текстовым полем. При нажатии на кнопку, в текст вставляется команда и подсказка об аргументах. Например, нажатие на кнопку POWER приведёт к появлению надписи:

POWER <0/1> [<TIMEOUT>]

Это означает, что команда POWER имеет один обязательный и один необязательный параметр. Первый параметр может принимать значение 0 (выключить питание) и 1 (включить питание). Как известно, сразу после включения питания, накопитель взводит флаг BSY. И вот время таймаута при ожидании снятия этого сигнала, можно задавать во втором аргументе. Таймаут задаётся в миллисекундах. Значение по умолчанию 20000 (20 секунд).

Примеры подачи команды:

POWER 0
POWER 1
POWER 1 10000
POWER 1 60000
POWER 1 600000

Рассмотрим остальные команды

WAIT <TIMEOUT>

Пауза в выполнении скрипта. Обязательный аргумент задаёт время паузы в миллисекундах. Добавлена в скрипт для организации задержки между выключением и включением питания.

Пример:

POWER 0
WAIT 2000
POWER 1

ATA_RESET

Подать программный сброс в накопитель. Аргументы отсутствуют

HARD_RESET

Подать аппаратный сброс в накопитель. Аргументы отсутствуют.

COMMAND28 XX XX XX XX XX XX XX TIMEOUT

Подать ATA команду в формате LBA28. Это может быть нестандартная команда Set Features, SuperOn (если Вы знаете соответствующий таскфайл) и т.п.

Аргументы – классические и соответствуют таковым в диалоге ATA команд (порты Features, Sector Count, Sector, Cyl_Lo, Cyl_Hi, Dev/Head, Command). Последний аргумент – таймаут в миллисекундах.

Пример подачи команды Set Features с аргументом Disable Write Cache:

COMMAND28 82 00 00 00 00 A0 EF 1000

COMMAND48 FFSS FFSS FFSS FFSS FFSS FFSS FFSS TIMEOUT

То же, но в формате LBA48. FF – первый байт в порт, SS – второй байт в порт.

Та же команда будет выглядеть так:

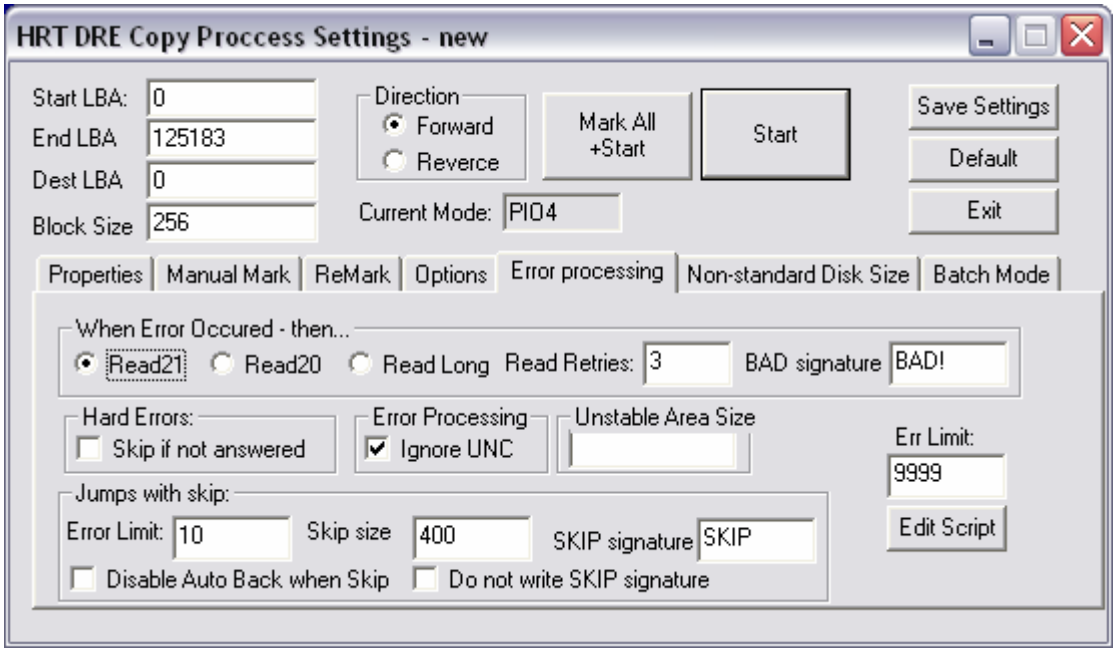
COMMAND48 0082 0000 0000 0000 0000 00A0 00EF 1000

Команда считывания сектора с LBA 0x12345678 будет выглядеть так:

COMMAND48 0000 0001 1278 0056 0034 00E0 0024 1000

Отредактировав скрипт, устанавливаем флажок в настройках таймаутов, после чего после выполнения действия, если оно успешно завершено, этот скрипт будет исполнен. Разумеется, в рамках разрешения таймаута, ценность представляют команды COMMAND28 и COMMAND48. Но с их помощью можно производить тонкую настройку накопителя.

Вкладка Error Processing



Раз уж работа идёт с неисправным накопителем (исправный бы не принесли в ремонт), при его считывании будут возникать массовые или одиночные ошибки. И при разных степенях убитости накопителя, эти настройки следует подстраивать под конкретную ситуацию. Основные настройки сосредоточены на вкладке Error Processing.

Группа **When error Occurred – then…** содержит методику разрешения ошибок. Сбойный сектор будет считан Read Retries раз. Если при этом хоть раз не возникло ошибки, его содержимое будет сохранено в копии. Если же все попытки завершились с ошибкой, разбор будет идти следующим образом:

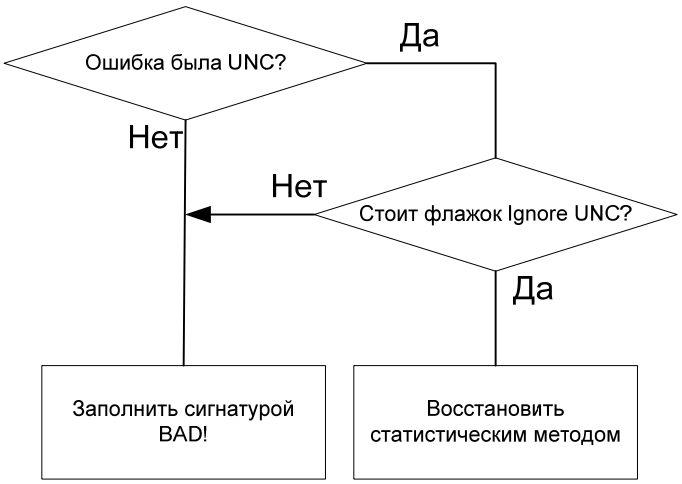


Рис. 38. Алгоритм заполнения сектора копии, соответствующего сбойному сектору источника

Здесь следует пояснить, что такое ошибка UNC. Если возникли ошибки AMNF, IDNF и прочие – это значит, что накопитель не нашёл сектор. А если сектор не найден, то где он возьмёт данные? Данных нет и быть не может. Но вот возникает ситуация: Сектор найден. И данные считаны. Но контрольная сумма не совпадает. Не беда! Есть корректирующий код ECC, который позволяет восстановить данные. Однако, разрушения столь велики, что и ECC не позволяет восстановить корректную информацию. Вот в этом случае, накопитель взводит бит UNC (Uncorrected). Тем не менее, он отдаёт данные. В

своё время, у накопителей IBM, при проблемах с транслятором, накопитель взводил UNC, но данные при этом были верны, просто записанный в сектор и ожидаемый LBA не совпадали, и накопитель считал, что что-то здесь невозможно скорректировать. Поэтому при возникновении ошибки UNC, можно попытаться спасти хотя бы часть данных из сектора. Для текстовых файлов это – спасение.

Статистическое восстановление производится следующим образом: При вычитывании сектора программа получила **Read Retries** вариантов данных. Далее программа бежит по каждому байту буфера и смотрит, какое значение было считано чаще всех. Именно оно попадёт в копию.

Вычитывание может производиться командой 20 (Read With Retried – выполняется дольше всех на сбойных секторах), 21 (Read Without retries) и 22 (Read Long – не рекомендуемая команда, хотя бы потому, что она не работает на накопителях более 128 гигабайт). Рекомендуется ставить радио кнопку в положение **Read 21**.

Теперь рассмотрим настройку пропуска методом прыжка

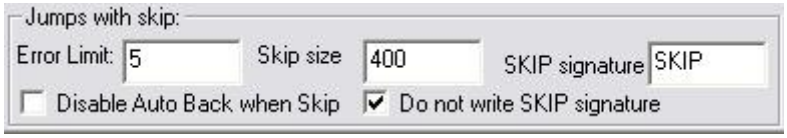
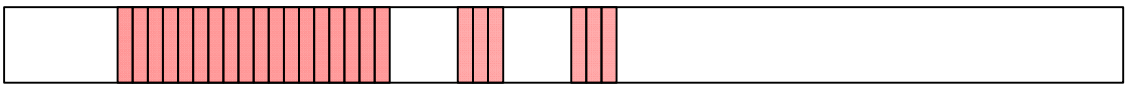
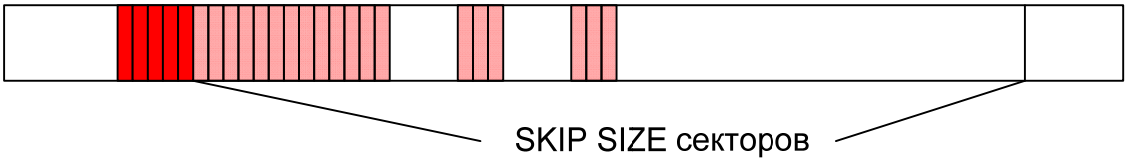


Рис. 39. Настройки пропуска методом прыжка

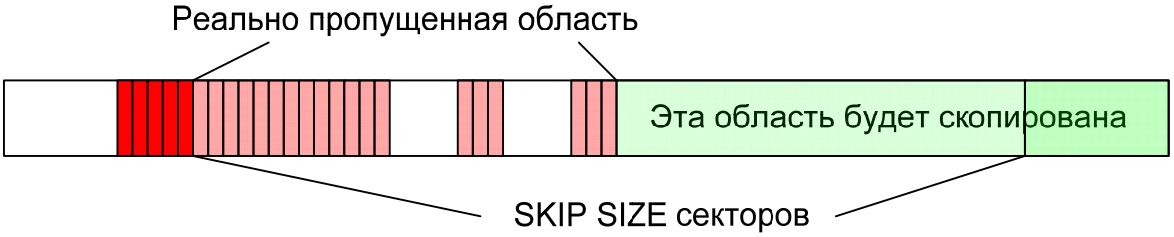
Как уже описывалось ранее, множественные дефекты лучше не вычитывать до конца, а пропустить их окрестность, оставив её на следующий проход. Если встретилось Err Limit ошибок подряд (не важно каких, главное, что ошибок), то будет пропущено Skip Size секторов. Однако алгоритм пропуска будет следующим: Пусть на диске имеются группы BAD блоков, отмеченные красным:



Утилита нашла 5 BAD блоков и сделала прыжок на SKIP SIZE секторов:



В реальной жизни, утилита начнёт идти от границы влево, пока не найдёт ближайший BAD блок. То есть, реально, будет пропущена следующая область:



Чтобы отключить механизм «отката от границы влево», следует установить флажок Disable Auto Back when skip. Он введён по настоятельным просьбам некоторых пользователей, хотя авторы не видят в нём особого смысла.

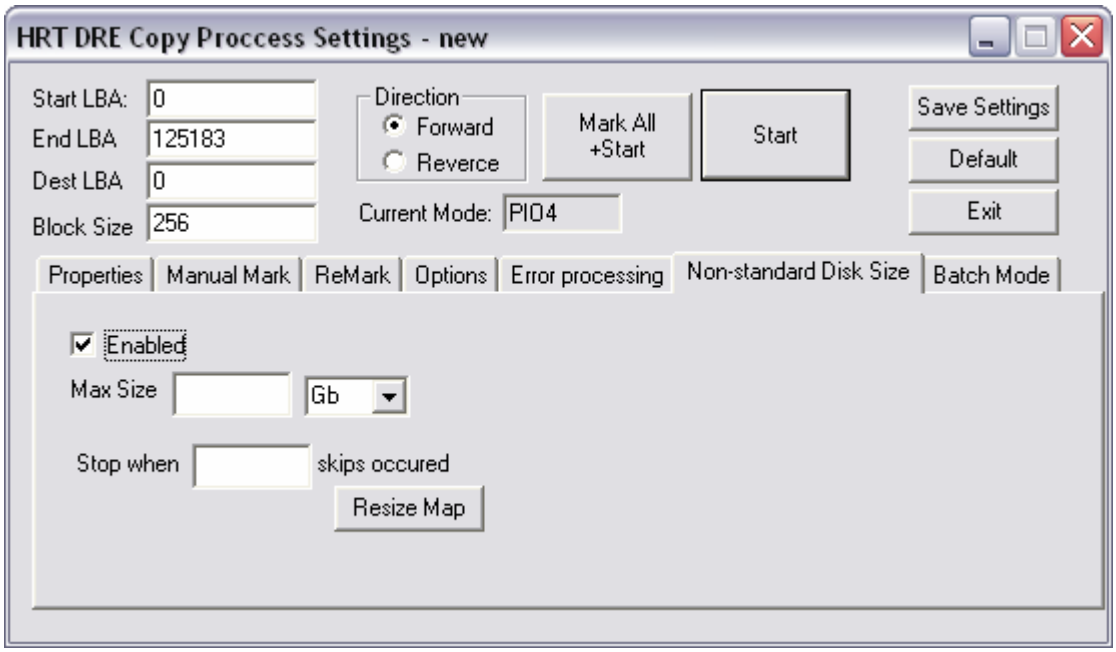
Если установлен флажок Do not write SKIP signature, в копии, соответствующей пропущенным секторам, будет прописана сигнатура SKIP (опять же, дань традициям некоторых фирм, ориентирующихся на содержимое). Если флажок установлен, пропущенная область не будет заполнена никакой сигнатурой.

Поле **Unstable Area Size** выполняет следующую функцию: некоторые ремонтники уверяют, что в сбойном месте лучше вычитывать данные посекторно (одна из причин этого состоит в том, что при посекторном вычитывании используется режим PIO, а накопители реагируют на ошибки более лояльно в режиме PIO). А ошибки обычно кучкуются. Поэтому, встретив ошибочный сектор, они предпочитают переводить утилиту в посекторный режим на случай, если скоро встретится ещё один сбойный сектор. И только прочитав некоторое количество гарантированно хороших секторов, согласны вернуться в блочный режим. В поле **Unstable Area Size** как раз задаётся размер области, которую следует копировать в посекторном режиме после ошибки. Если область не заполнена, она не используется и посекторный режим не включается.

Группа Err Limit позволяет отредактировать скрипт, выполняемый при возникновении большого количества ошибок. Например, замечено, что при копировании накопителей Seagate Barracuda 11, иногда начинают сыпать массовые ошибки. Если же перещёлкнуть питание, эти же сектора начинают читаться вполне нормально. Поэтому можно задать число ошибок подряд, после возникновения которого, следует выполнить скрипт. И именно здесь пригодятся функции POWER, и XXX RESET. По умолчанию задано нереально высокое число ошибок.

Маленькая хитрость. Если Вам не нравятся настройки по умолчанию, настройте параметры по своему усмотрению, нажмите кнопку Save Settings, а затем скопируйте файл copy settings.ini из каталога проекта в каталог с утилитой. Теперь эти настройки будут использоваться по умолчанию. Сохранятся даже отредактированные цвета квадратиков карты. А что не нравится – вычистите из этого файла, данные параметры будут браться из программы. В первую очередь, придётся вычистить имя накопителя, его размер, имя последней сохранённой карты и т.п., иначе они будут подставляться во все новые проекты.

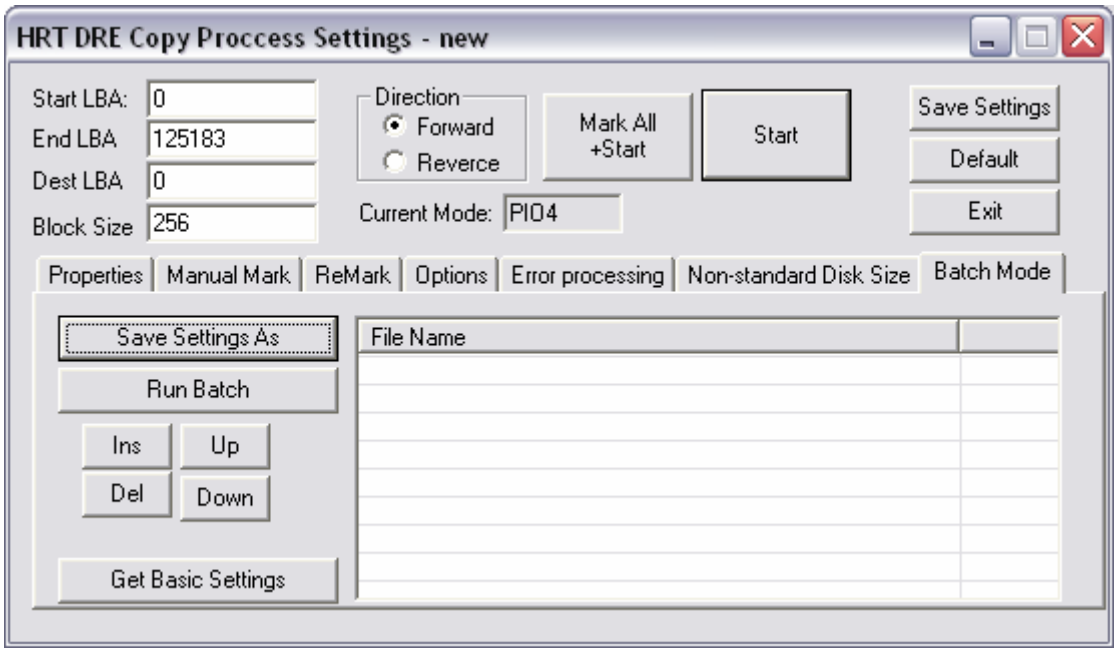
Вкладка Non Standard Disk Size



Данная вкладка введена для накопителей, у которых неверно определяется объём, но которые отдают данные, несмотря на это.

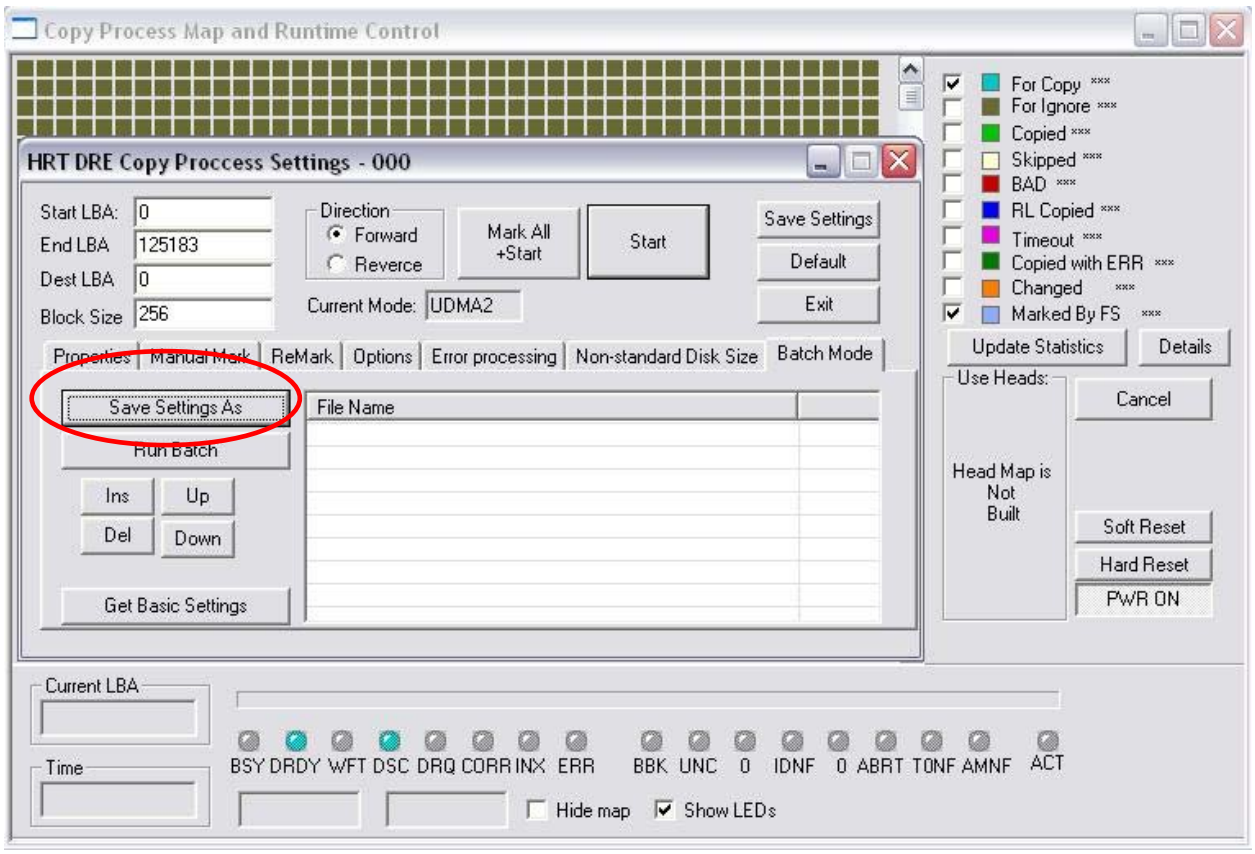
Установив флажок Enabled, Вы сможете изменить свойства утилиты для работы с такими накопителями. Установите размер накопителя в гигабайтах (вообще, можно и в мегабайтах, но это – анахронизм) и нажмите кнопку Resize Map. Карта будет увеличена под новый размер. Кроме того, можно задать порог количества пропусков подряд, после которого копирование будет остановлено. Много пропусков подряд с высокой степенью вероятности говорят о том, что достигнут физический предел накопителя, ведь размер установлен «на глазок».

Вкладка Batch Mode

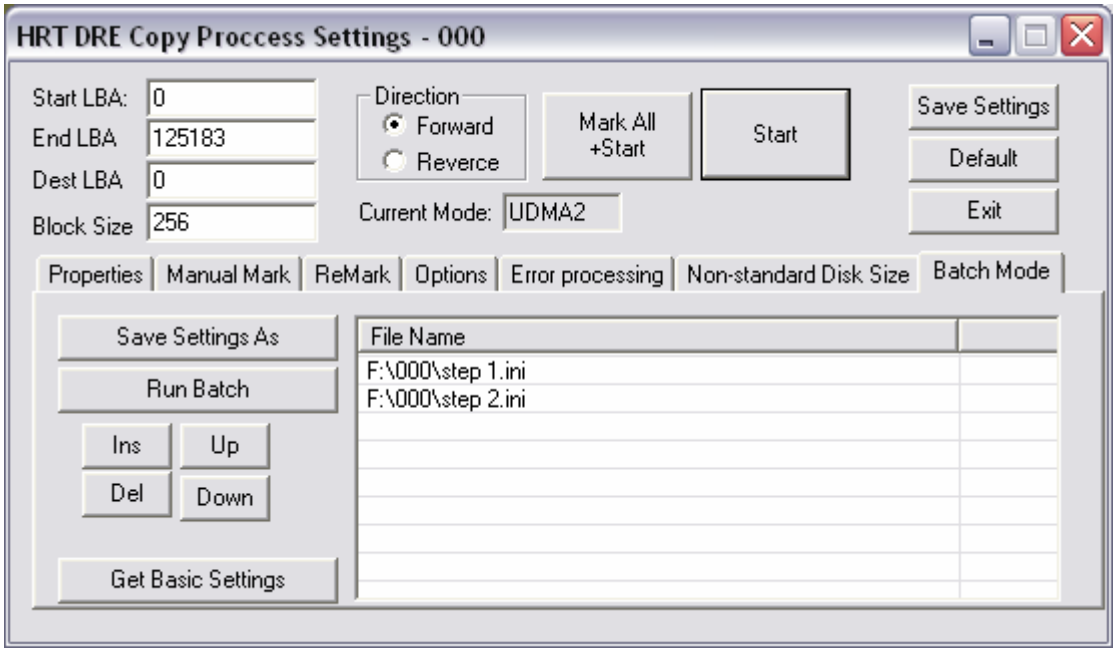


Данная вкладка позволяет запрограммировать пакетный режим для того, чтобы выполнить несколько проходов копирования. Например, первый проход с «зажатыми» таймаутами, второй проход – вычитывание SKIPов, третий проход – детальное вычитывание таймаутных секторов с большим таймаутом. Другой вариант – сначала копирование по головке 0, затем – по головке 1 и т.п. Именно для этого и сделан механизм пакетного режима.

Задайте параметры для очередного шага (все настройки на вкладках, таймауты, скрипты, флажки на карте и т.п.), после чего перейдите на вкладку Batch Mode и нажмите кнопку Save Settings As, после чего задайте имя файла с набором настроек.



Аналогично задайте настройки для прочих шагов. Имейте в виду, что настройки, сделанные в рамках одной работы, можно использовать вновь и вновь. Поэтому их можно сохранять не только в каталог проекта, но и в любой другой каталог. Когда сделаны все необходимые настройки, вставьте имена файлов в список, пользуясь кнопкой Ins.



Выбирая настройки в списке, Вы сразу же будете видеть их в основном окне и коне карты. Если необходимо вернуться к основным настройкам утилиты, нажмите Get Basic Settings.

Кнопки Up и Down перемещают записи в списке (пакетный файл будет исполняться сверху вниз), а кнопка Del удаляет файл настроек из списка.

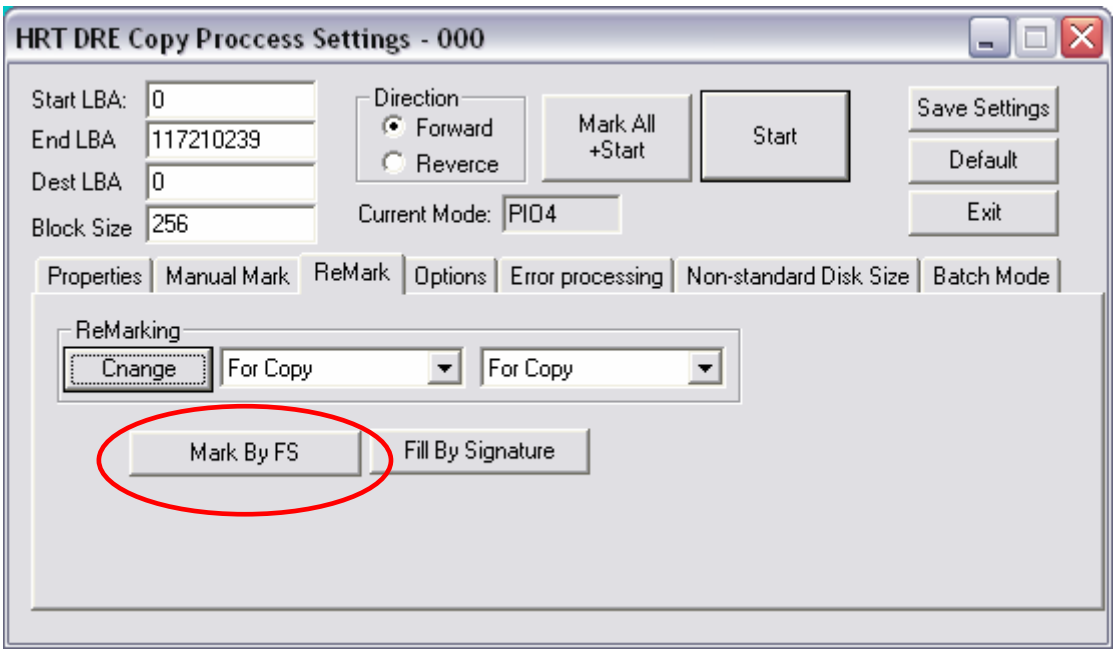
Когда всё готово, нажмите кнопку Run Batch, после чего начнётся процесс копирования согласно настроек.

Копирование пространства, занятого файлами

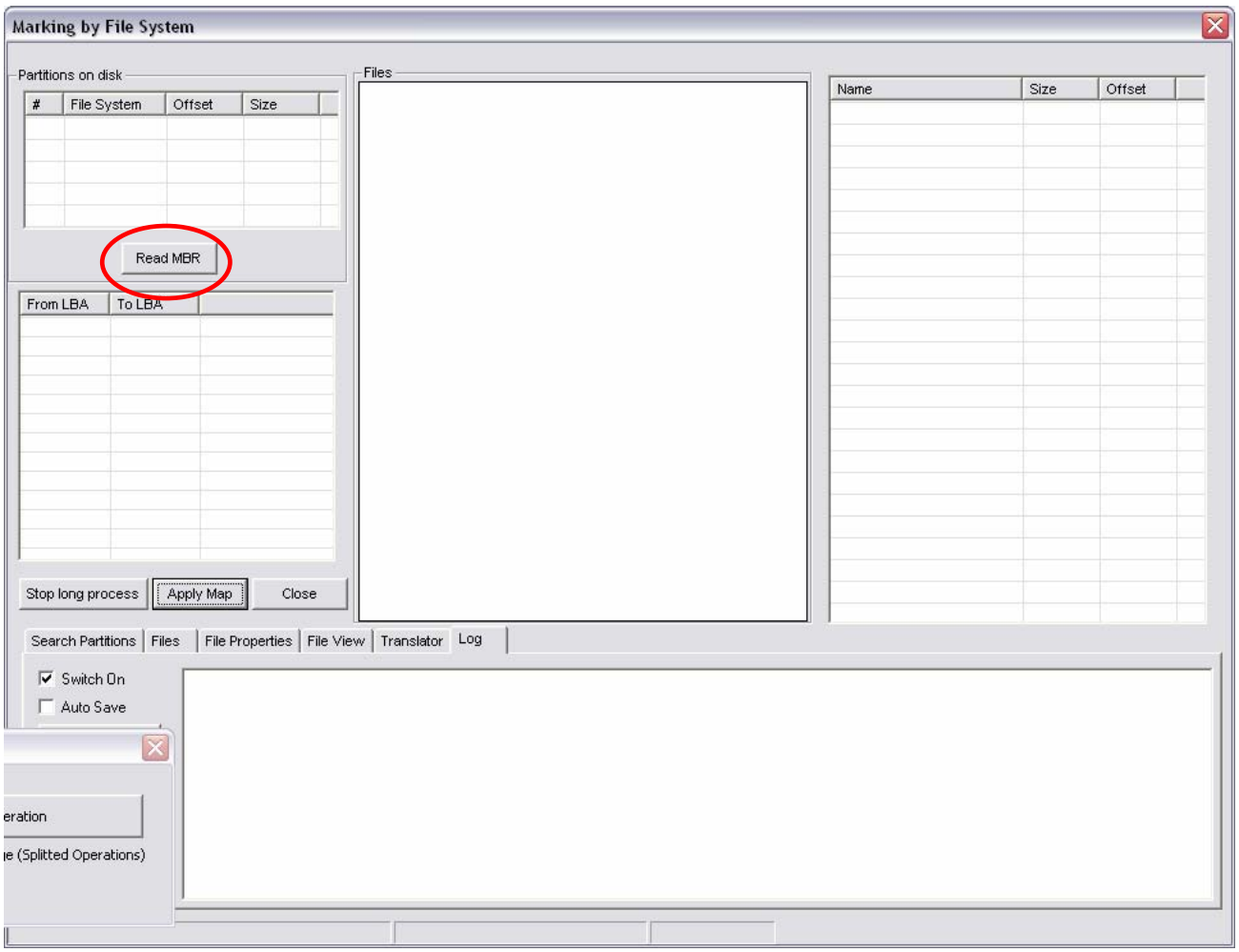
Первый способ сократить объём копируемой информации – это выделить только занятое файлами пространство. Файловые системы NTFS, Ext2 и т.п. содержат специальный файл BITMAP, в котором отмечены те сектора, которые заняты файловой системой. Файловая система FAT выделенного файла с битовой картой не имеет, но сама по себе таблица FAT – это ни что иное, как битовая карта. Поэтому выделение занятого пространства возможно и в ней. Рассмотрим методику, как это сделать. На данном этапе мы будем смотреть именно на сам процесс. Теорию отложим на следующий раздел.

Итак. Рассмотрим накопитель, на котором мы хотим выделить только занятое файловой системой. При этом, скорее всего, копирование уже началось, но оно шло так долго, что было решено прервать его и начать копировать только занятое. То есть, карта уже помечена, как Marked to Copy.

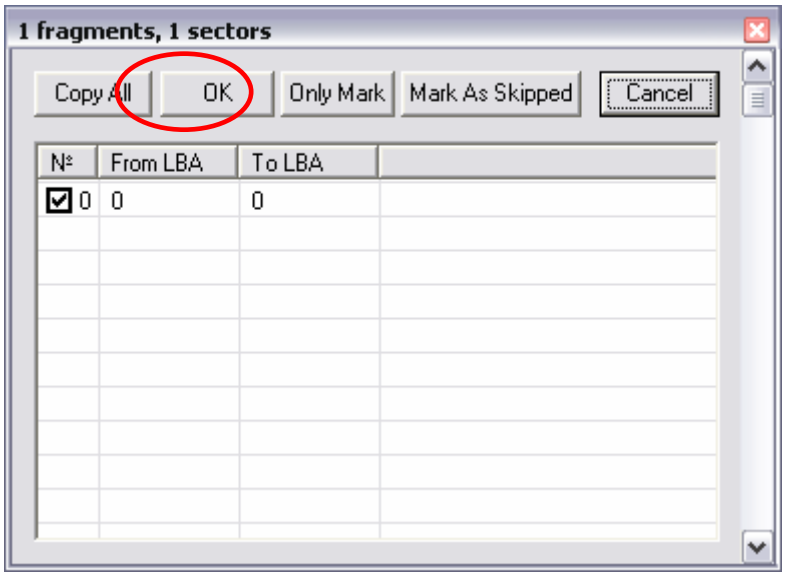
Переходим на вкладку ReMark и нажимаем Mark By FS



В появившемся окне нажимаем Read MBR

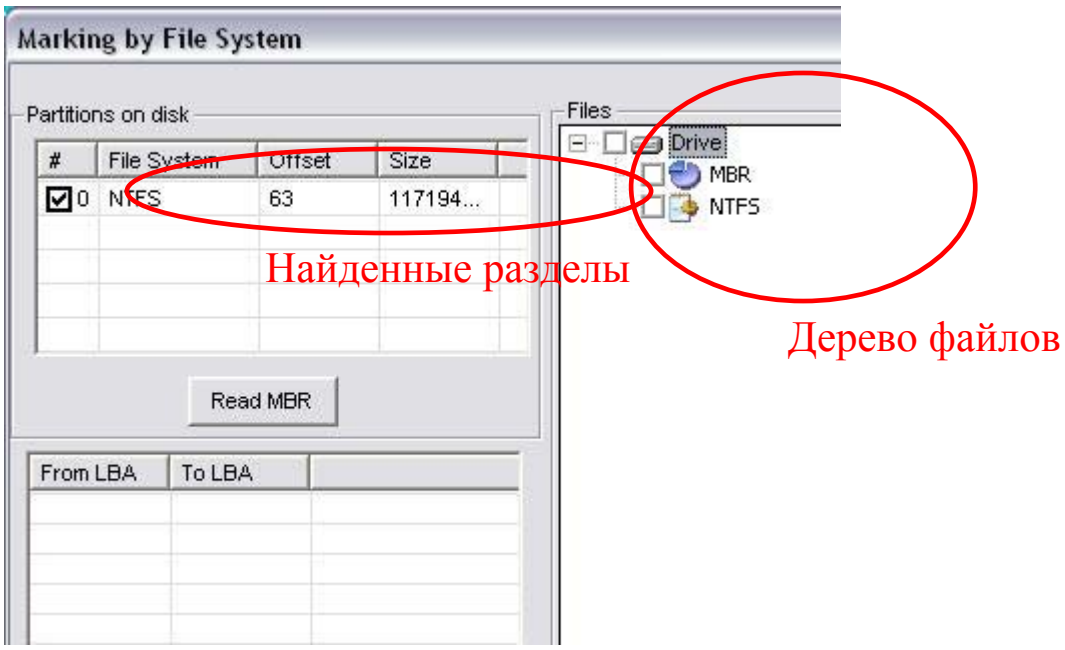


При всех чтениях будет появляться запрос вида:

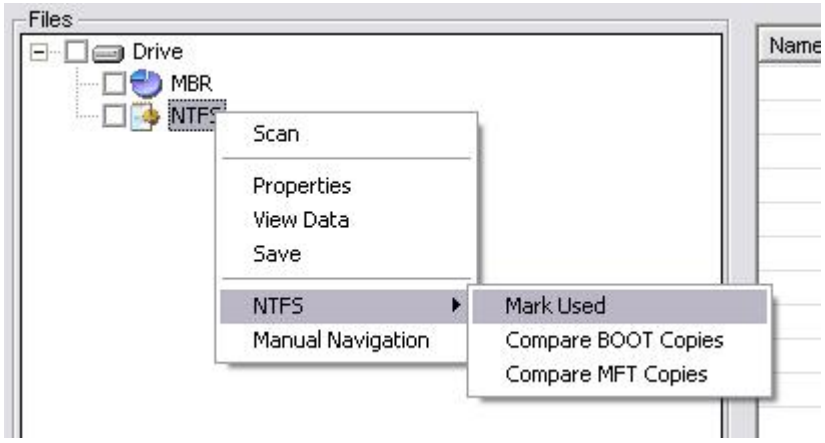


Пока примем за основу, что там надо нажимать Copy All. Потом рассмотрим, как это окно помогает разрешать проблемы.

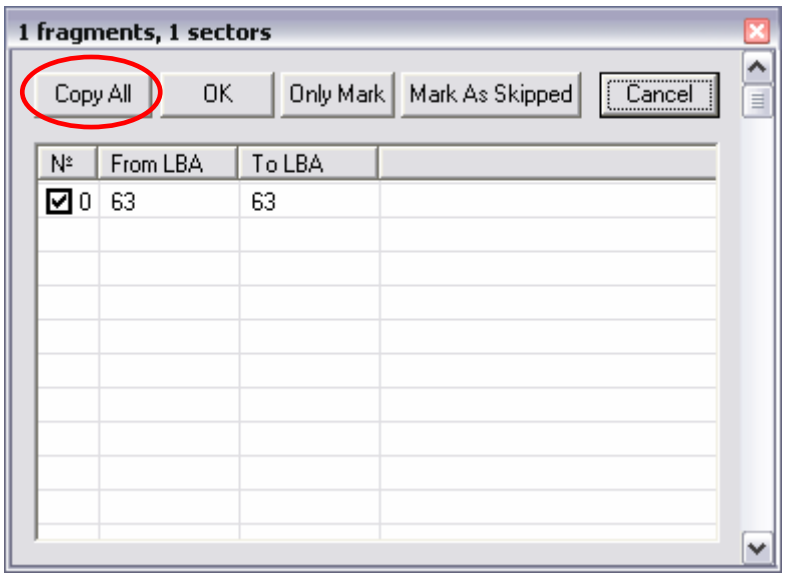
В главном окне появилось дерево файлов

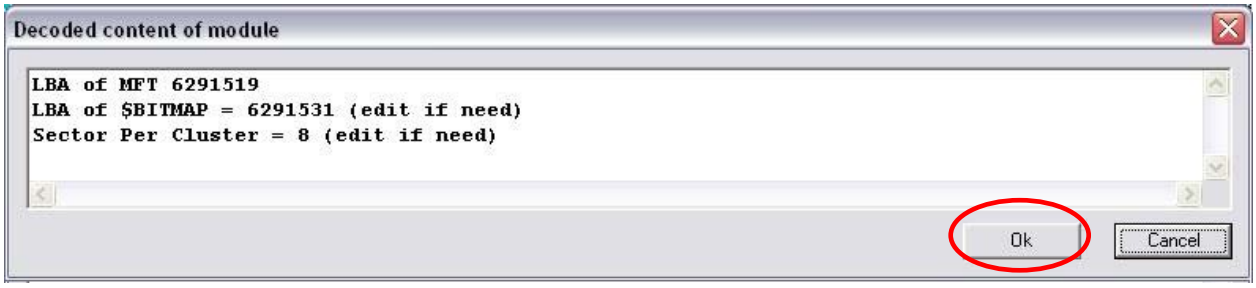


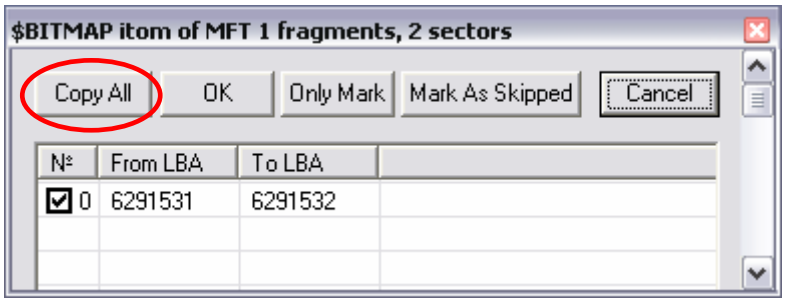
Наводим курсор на раздел в дереве, нажимаем правую кнопку «Мыши» и выбираем пункт меню NTFS->Mark Used (даже если это раздел FAT, мы же выяснили, что у FAT таблица аналогична битовой карте)

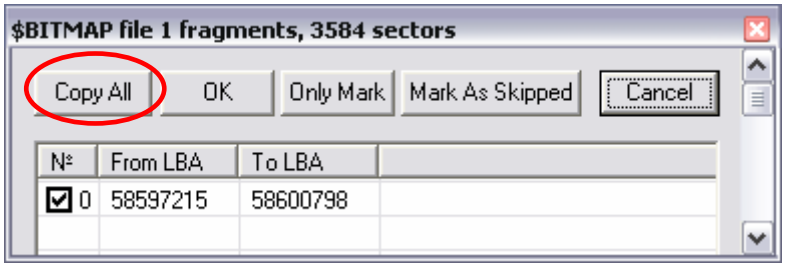


Дальше утилита начинает задавать вопросы. В целом, достаточно всегда отвечать на них OK. Смысл данных вопросов в том, что накопитель неисправный. Данные могли считаться с ошибкой. И утилита взяла за основу неверные значения. Если Вы в состоянии увидеть факт ошибки и знаете, как её исправить – отредактируйте значения и после этого нажмите OK.

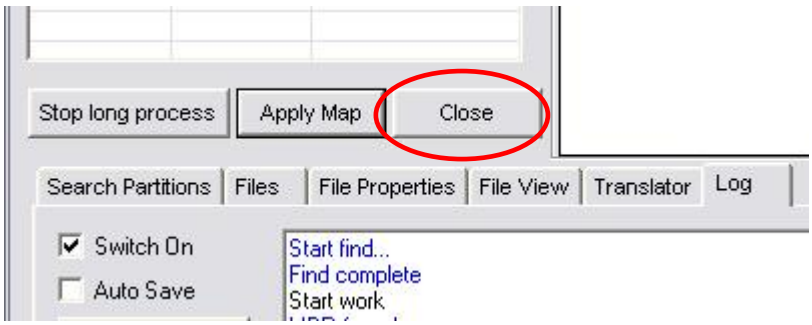




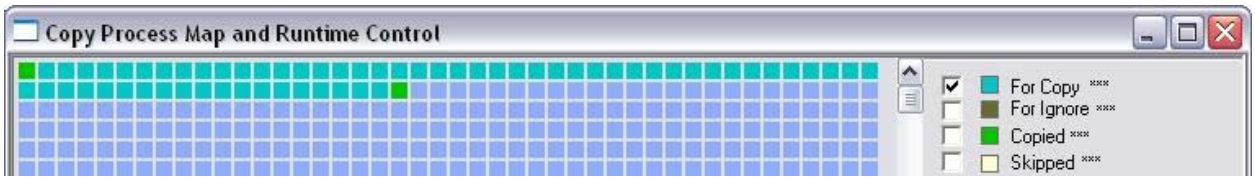




Собственно, всё, окно файлового разборщика можно закрывать.



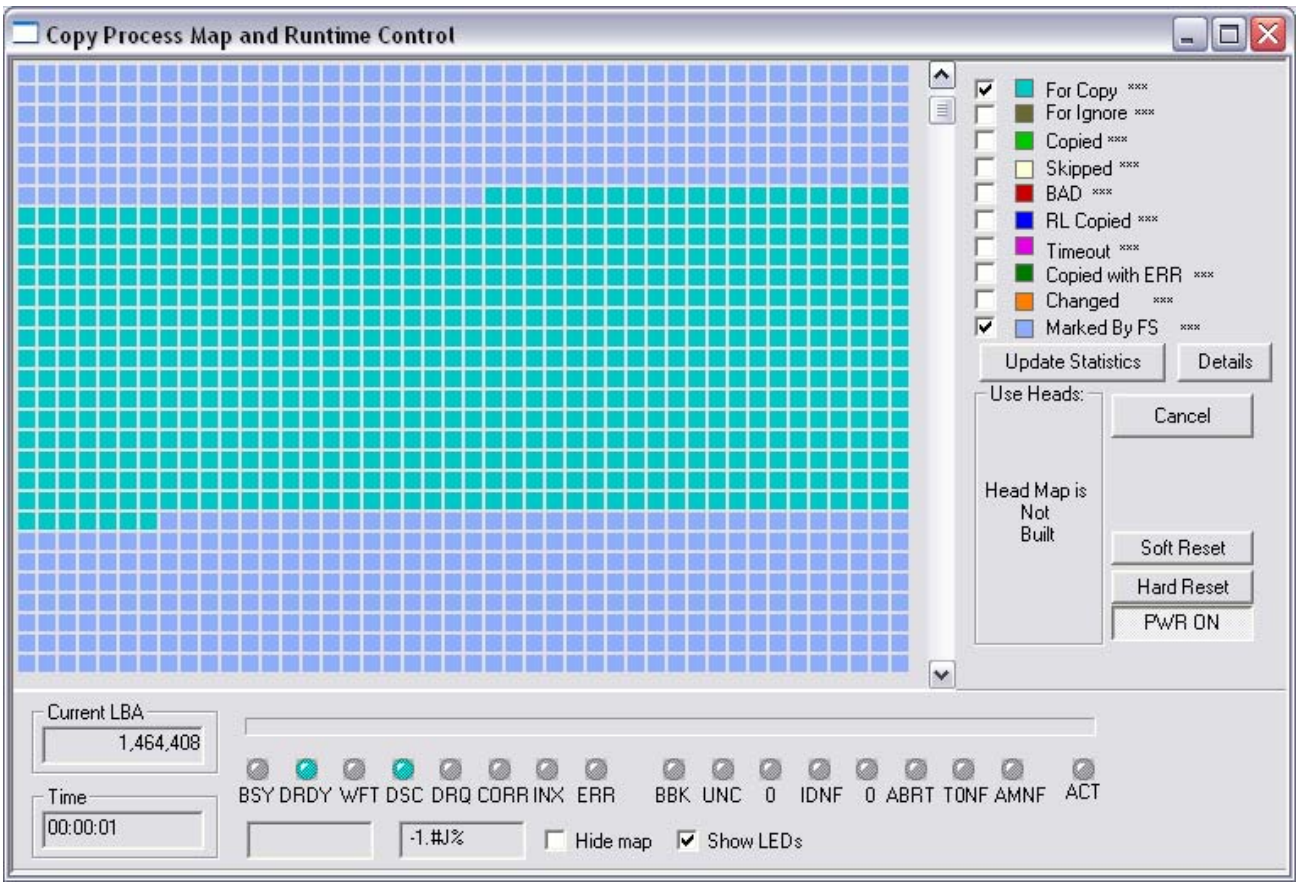
Посмотрим на карту. Для написания документации использовалась чистая карта, отмеченная значками Marked to copy. После описанных действий, её начало стало выглядеть так:



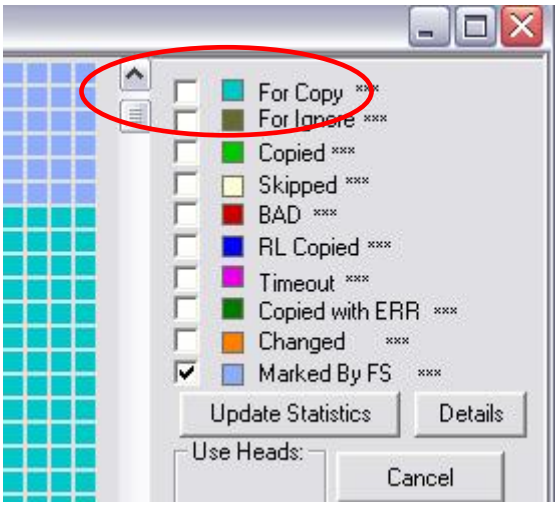
Зелёные сектора – это те, которые были считаны файловым разборщиком. Раз они всё равно считаны, утилита автоматически сохранила их в копию. При последующих обращениях к этим секторам, будет взято содержимое копии.

Голубые сектора – не тронуты файловым разборщиком

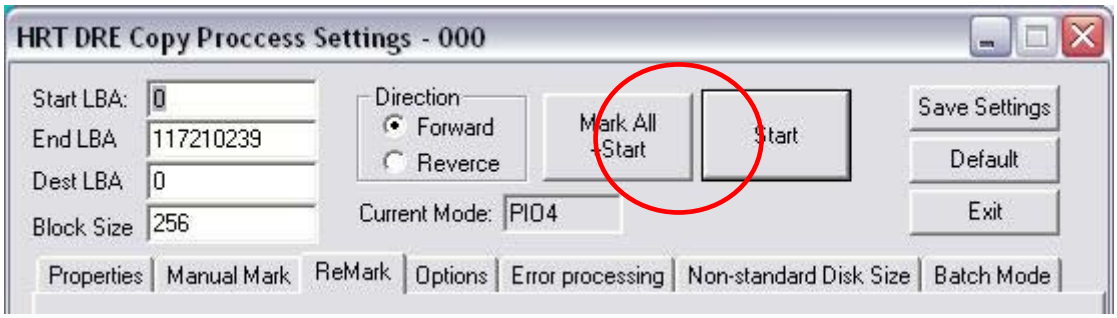
Сине-серые сектора – это те, которые заняты файловой системой. Файловый разборщик отметил их меткой Marked By FS. Полистаем карту. Вот виден фрагмент, не занятый файлами.



Теперь снимаем флажок For Copy и оставляем флажок Marked By FS:

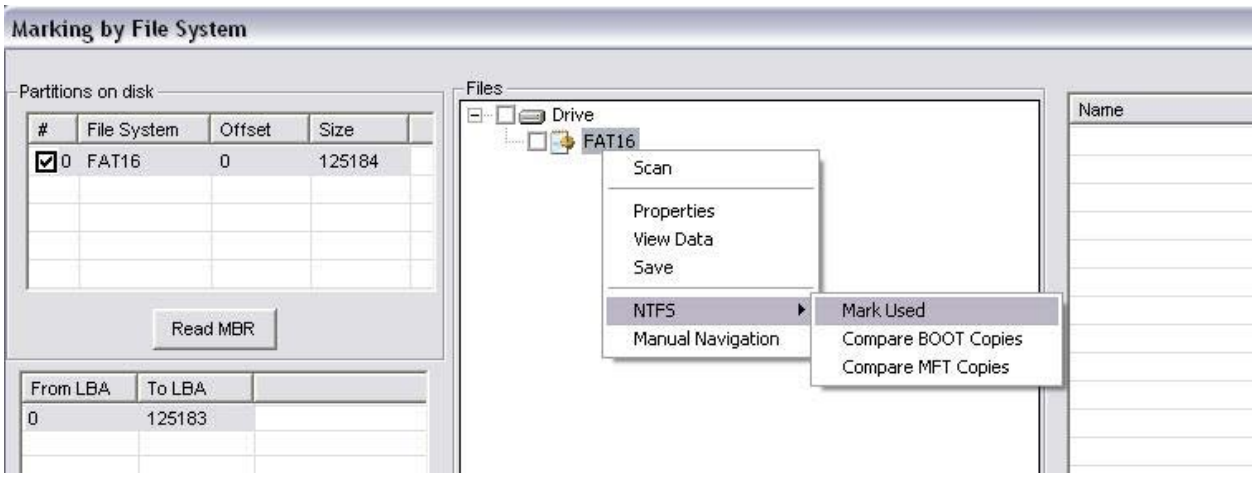


И начинаем процесс копирования

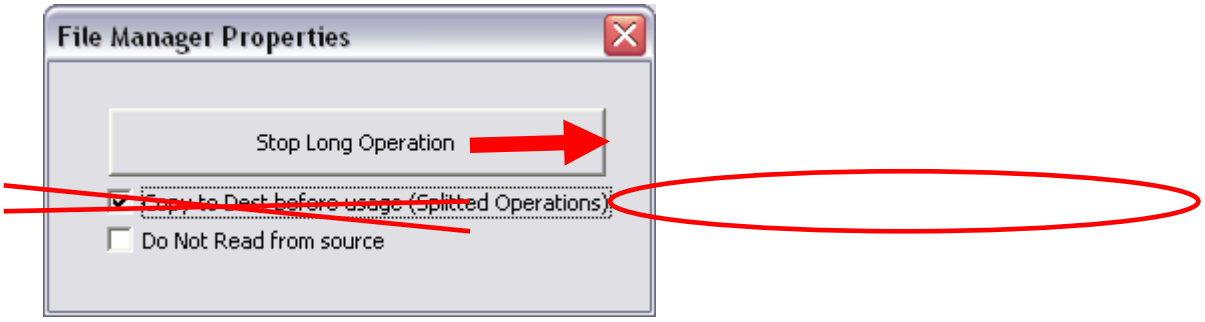


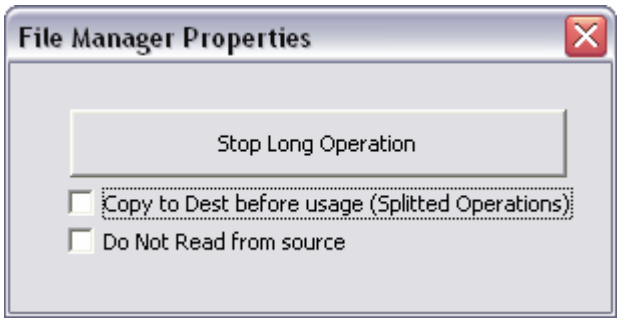
Утилита не будет пытаться считывать сектора, не относящиеся к файловой системе, чем значительно сократит время копирования сильно побитого диска.

Для FAT всё делается точно так же!



С одной оговоркой: там выдаётся слишком много запросов на копирование. Поэтому допускается просто снять флажок copy to dest before usage (Splitted Operations)





Оглавление