DBACKUP DOCUMENTATION

REVISION 2.6

Table of Content

Purpose

This document describes the usage of the program ***dmfmbkup***. This is the program for backing up disks using the **Discovery Cartridge**.

Disk Backup, Legality and Ethics

Happy Computers cannot offer a legal opinion. Please consult your own legal counsel for accurate information. We have heard that in the United States, there is a "**fair use**" provision of the copyright laws. It may be considered fair use to make a backup of a copyrighted computer program you have purchased, and use the backup in place of the original. We have heard that it may be illegal to make copies and distribute them in any manner.

Software authors work very hard to produce the marvellous programs that make your computer so much fun and so useful. Generally speaking, they are not doing this for the fun of it. They do it to make money. They have to eat, make rent or mortgage payments, and purchase cars, VCRS, and clothes (no order of importance is implied). They need to be compensated for their work, like any other person. Copy protection is placed on disks as an attempt to ensure that authors receive fair compensation for their work. Our disk backup system was created to allow users to make backups. It is not our intention to deprive authors of income. With a powerful backup system such as ours, the copy protection on the disk will at least serve as a reminder that authors should be compensated.

There really is a need for backups. Disks can and do fail, for a whole variety of reasons. Almost anyone who has used computers and floppy disks for some period of time can attest to this. The safest thing to do is make backups, and put the original away for safe keeping. It's best if you also make a backup to stash away just like the original. Check the laws of your own locality before copying any computer program.

The ***dmfmbkup*** program, in conjunction with the **Discovery Cartridge**, creates a disk backup system more useful and powerful than most other backup systems available for almost any personal computer system. This system can directly copy disks using copy protection that certainly defies all "**Software Only**" copy programs. It also copies disks that most other hardware products can't touch, even those on other computers.

The program can also be used to copy disks that are not copy protected. When writing disks that are not copy protected, the **HART** chip has a speed advantage over the disk controller chip in your computer. The **HART** chip can format and write in one revolution of the disk. The disk controller in your computer needs 1 disk revolution to format, and another disk revolution to write the data.

When writing to the destination disk, this program verifies all of the sector data written. There is no option to continue if a verify error occurs.

The program has a feature for efficiently making many copies of the same disk, providing you are using 2 drives, and your computer's memory is big enough to read the entire program in at one time. This feature is called "Alternate Destination Drives". It is described in the section below on Backup Control File Contents.

The program has a feature called "**Backup Control**". Instead of selecting parameters that control the backup process from menus, the program allows the user to select a **backup control** (BC) file. The backup control file "tells" the program how to read the source disk. Some backup control files are provided by **Happy Computers**. The users may also write their own backup control files. These backup control files do NOT bear any resemblance to the PDB files of the 8 bit HAPPY. All programs copied by the **Discovery Cartridge** do not require a **Discovery Cartridge** to run the copy.

Many copy protected disks can be copied without the need to specify a BC file at all. Other copy protected disks may require the use of a particular BC file to generate a working backup. Some backup control files are tailored to providing fast copying. Other BC files can used to copy a whole variety of copy protected disks that otherwise won't copy.

The ***dmfmbkup*** program is written to provide good copying speed while being able to copy many disks without specifying any backup control. We could have written the program to be more automatic, eliminating the need for backup control files in more cases. If this were the case, the program would be much slower in operation, almost intolerable. The use of BC files is a trade off between copying speed, and the complexity of the backup process.

Copying Disks - Step By Step Procedure

1) Execute the ***dmfmbkup***. TOS program.
2) Select the **options** menu.
3) Select the source and destination drives.
4) Select the copy mode, the simplest case is **floppy To floppy**.
5) Select a **Backup Control file**, the simplest case is "(none)".
6) Exit from the **options** menu.
7) Begin the copying process.
8) Insert the source and destination disks as instructed by the most recent message displayed at the bottom of the screen.
9) During the copying process, you may abort the copying operation by holding down the **Alternate** key. You may have to hold down the **Alternate** key for several seconds before the program stops.
10) The program will report any error conditions. If an error occurs, it is likely that the disk has not copied correctly.

Option Details

The first menu allows the user to begin copying, or select the options menu, or exit the program. The options menu allows selection of **copying mode** and **backup control files**, discussed below. The options menu also allows the user to **select the source and destination** drives. Up to 4 different source and destination drives are possible if the DRIVE 3/4 option is installed in your **Discovery Cartridge**. See the **dddrive.doc** file for further information. If the conditions for accessing a 3rd and 4th drive are not met, then only drive A and B can be selected as the source and destination drives.

Copying Modes

Floppy to Floppy Mode

The copy mode selection lets you select copying from **floppy to floppy**. In this case, the disk read from the drive specified as the source drive will be copied to the disk in the drive specified as the destination drive. There is no image file generated or utilized.

Floppy to File Mode

Another mode selection is to copy from **floppy to file**. In this case, the image of the disk read from the drive specified as the source drive is copied into a file. The file written is named "**mfmbtemp**". The file is written to the same drive that the ***dmfmbkup*** program was loaded from. In the current version, there is no way to instruct ***dmfmbkup*** to use a different drive or file name for this purpose. No destination disk is created. There must be sufficient space for the file on the disk where the file is written. It is not possible to tell in advance exactly how much space is needed. Usually, over 400K will be needed for the image of an 80 track single sided disk. It is possible that up to 9 megabytes can be required for some disks. In most cases, the image file for an 80 track single sided disk will not exceed the 720K disk space available for file storage on one double sided floppy. If the file **mfmbtemp** already exists, it is erased and replaced with the new image file. The disk used for the file must be formatted in advance. You must not insert or remove the disk that contains the file while the drive select light is on.

There are many different reasons for copying to an image file. The first would be to allow the program to be transmitted by modem. The ***dmfmbkup*** program makes no attempt to compact the size of the file it writes. We suggest that you "**arc the file**" to compact it before transmission. **Happy Computers** does not currently supply instructions on what this means or how to accomplish this. Please note that it may be illegal to transmit a copy protected computer program by modem. The compacted file may allow several file images to be placed on one floppy. It would also reduce the space required to store the image of the file on your hard drive.

Another reason to copy to an image file would be to allow editing of the contents. Image files contain a definition of the disk format along with the data. Currently, **Happy Computers** does not supply an editor. We do supply the information which documents the contents of the image file, see below.

Another reason to copy to an image file is to save time. The process of reading and analyzing a copy protected disk takes longer than the time it takes to read back in an image file you have already created. If you were going to make copies of a program at different times, you may want to save the program as an image file rather than as an executable backup. Each time you wanted to make an executable copy, you could do the job faster by reading from the image file rather than reading the original disk.

File to Floppy Mode

Although this program can create a file which represents the image of the copy protected disk, the copied program cannot be executed while it is in the image file form. An executable backup copy can be created from the image file by using the **file to floppy** copying mode. In this mode, the image file named "**mfmbtemp**" is read from the drive that the ***dmfmbkup*** program was loaded from. This file specifies the format and data of each track that will be written on the destination disk, which is inserted in the drive specified as the destination drive. With the current version of ***dmfmbkup*** it is not possible to specify a different name for this file, or a different drive for reading this file from.

The image file may have been created by the **floppy to file** mode of this program. It may have been edited or completely created by the user. It may have been received over a modem, and "**extracted from an arc file**". In any case, the data in this file must conform to the format which is described below. If there is some problem in the format, an "**error in write code processing**" can occur. Unmodified files that are produced by the floppy to file mode will not cause this error.

While in the **file to floppy** mode, the backup control file specified has no control over the format and data written to the destination disk. Complete control over the format and data written is determined by the contents of the image file. For this reason, in the current software version, the **alternate destination drives** mode, which is selected by the backup control file, is not available with the **file to floppy** mode.

The backup control file that was used when the file was created, if any, does not have to be selected to recreate a backup from a file. The file itself contains all of the information needed to reconstruct the backup.

Selecting Backup Control Files

The file "**dbkupcf.s**" is one large file which contains all of the backup control files (BC) that can be accessed from the ***dmfmbkup*** program. We suggest that you print out this whole file, and read through the English language descriptions. One BC file may include a whole list of program names that do not appear in the title of the file. If you find a particular file that backs up a particular program that is not listed, make a note of this for your own future convenience.

Software companies may change their disk protection method from time to time. The disk that **Happy Computers** has examined may not have the same copy protection as the disk you have, even though the title and even the version number may be the same. For this reason, a program may not copy with the backup control file even though the instructions say to use it.

Error Messages

Some error messages may just need a common sense solution. Unusual error messages may imply that something is wrong with your computer, disk drive, destination disk, or **Discovery Cartridge**. Error messages may also imply a bug in the program. Before contacting **Happy Computers** for help in resolving errors, you must have ready the specific set of circumstances that produced the error. Computer type, other resident programs, disk drive type, diskette manufacturer, serial number of **Discovery Cartridge**, name of program being copied, statistics concerning how often the error occurs, are all necessary before we can begin to resolve the problem. It really does help to try a different set of hardware before placing fault on any one particular item. You may just be using the product incorrectly, please read our documentation first! Getting write / verify errors will usually mean the destination disk or drive has some defect. Also, check to make sure that the cables are all plugged in correctly before calling us!

Problems Copying a Program

Please see the section in the file DISCOVER. DOC titled *What To Do If You Can't Back It Up* for help in this area. Does this before you call our office!

Technical Information - Backup Philosophy

Happy Computers Inc. has the philosophy that a backup program should always do the most to ensure the integrity of the resultant copy. The copy produced should operate just like the original. Our software does not remove the protection, or modify the program being copied in any way. Checksums are usually written for each segment of data on floppy disks. **Happy Computers** believes that the backup program must do the up most to check that the copy produced is correct, with correct checksums. For these reasons, **Happy Computers** believes that dumb "bit" copiers should be avoided. Our ***dmfmbkup*** program follows in the steps its successful predecessor on the 8 bit Atari, the **Happy Backup Program**. The ***dmfmbkup*** program does not just blindly copy bits or flux spacings. The ***dmfmbkup*** completely analyzes the disk being copied track by track, to create a complete description of each track. The ***dmfmbkup*** program uses this decoded description to recreate each track on the destination disk.

Copying Other Formats

Analyzing the data read really does make a difference. Other hardware copying devices we have seen that do not completely analyze the data are so erratic, you wind up never knowing if the copy you made will work. Our system produces consistent results. This also implies that software must be written to handle each format properly. In its present form, this program can copy **Single Density, High Density, Macintosh Format, And Amiga Format**. However, the program is not currently designed to analyze the data read for these formats. There would be a risk that copies produced may not consistently be correct. As time goes on, **Happy Computers**' software library will expand to properly analyze and copy more disk formats. The **Discovery Cartridge** hardware will never be the limiting factor. Cast your vote for a new feature by writing your request in a letter mailed in. Phone calls are not considered votes, just passing desires.

Technical Information - Backup Control File Contents

The following is a description of the contents of the file "**dbkupcf.s**" which contains all of the backup control files that the user can select while operating the ***dmfmbkup*** program. **Happy Computers** provides this information for several reasons. First, the user can examine the BC files provided by **Happy Computers** as examples of how these are written and integrated together. Second, the user may be able to utilize this information to generate their own BC files. Aside from the information presented here, **Happy Computers** cannot provide additional help regarding the content of BC files.

The **dbkupcf.s** file can be printed to the printer, and edited with an ASCII text file editor, which **Happy Computers** does not supply. Use the editor you are comfortable with. There are no special control characters, other than standard ASCII characters. We suggest that you do not modify the BC files provided by **Happy Computers**. We suggest that you copy any section you wish to modify to the end of the file, and make your modifications there. This will effectively add new selections. The selection numbers that we have provided will not be modified.

Each BC file set within **dbkupcf.s** starts with **a title line**. All of the characters through the first CR or LF will be displayed as the title for the user to make a selection from. The selection number is computed by the ***dmfmbkup*** according to the position within the file. The selection numbers are not contained in the **dbkupcf.s** file. For example, if you add one new BC file set at the start of the **dbkupcf.s** file, all of the previous selections will have their selection number increased by one. For compatibility with low resolution, **Happy Computers** has limited all text in our file to 40 characters per line. The title line of the currently selected BC file or "(none)" is displayed by the ***dmfmbkup*** options menu.

Following the title line is an optional **information display area**. This information is displayed by the **"I"** command from the option selection menu of the ***dmfmbkup*** program, and is also displayed just after a particular BC file is selected. All of the text starting with the title line through, but not including the first line that has a single exclamation mark as the first character on the line, is considered the information display area. Characters within the information display area do not control the read process.

The first line after the line which had the exclamation mark (!) begins the area where the backup control commands are located. You must use the exact upper case character shown.

Alternate Destination Drives Mode

This mode is selected by having the **letter A** as the first character on the first line following the **exclamation mark (!) terminator**. This mode can be selected for any BC file by adding the line with the letter A, provided that the other conditions for using it are met. With the ***dmfmbkup*** program in this mode, once the source disk is read in, the source and destination drives alternate as destination drives. This allows the user very efficient utilization, to write many disks at one sitting. The system writes to the disk in the other drive, they alternate as soon as the disk just written is ejected, with no keyboard entry required. Be sure that the disk in the other drive is ready to be written before ejecting the disk that just completed. The source disk is not read in again until the user selects to terminate this mode.

After writing to a disk is completed, the system sounds one bell, and flashes the select light continuously on the drive that just completed writing. The user ejects the disk from the drive with the flashing light, puts that disk in the "done" box, and inserts another disk. In the meantime, just as soon as the current disk has ejected, the system begins writing on the other drive. The message on the screen also announces that the writing has completed correctly.

If a write / verify error occurs, the system reports this on the screen, and besides flashing the drive select light on the drive with the bad disk, the system also keeps ringing a bell tone through the monitor speaker. This audio indication should be sufficient to cause the operator to place the disk just ejected into the "**bad disks**" box. The operator should only need to observe the disk drive select lights, and pay attention to the bell sound, to copy disks in this mode.

Conditions required to use this mode are as follows:

1) the 'A' must be the first control line in the backup control set
2) the source and destination must be different drives
3) the whole source disk must fit and read into memory at one time

WARNING: There are some defective drives that do not always recognize the change in the **write protect** status that this mode uses to determine when the destination disk is ejected. If you have a drive that sometimes does not allow the system to recognize the disk eject, usually the eject will be recognized when the disk is re-inserted and re-ejected. However, it is strongly recommended that this drive not be used with GEM DOS files, since GEM can really screw up the contents on the disk if it doesn't recognize that the disk has been changed.

Side Selection

Unless otherwise specified, all reading commands are directed to the first side of the disk. Most copy protected disks do not utilize the back side of the disk, for compatibility with single sided drives. To direct the program to read from the second side, the **letter S** is placed as the first character on one line by itself. After the occurrence of this line with the letter S, all backup control command lines that follow, will read from the disk's second side. There is no way to switch back to the first side. With no backup control specified, only the first side of the disk is read.

Comment

A line that begins with an asterisk **"*"** is considered to be a comment line. The text on that line will be ignored by the backup control system.

Track By Track Control

If a line begins with a number **"tt"**, which is allowed to be between 0 and 99, then this is considered to be a track number, which will be followed by a colon ":", and further backup control specifications. The "tt" numbers do not have to be in ascending order, but this may be more logical. The order in which reading is specified is the order in which the data will be written to the destination disk. With no backup control specified, the ***dmfmbkup*** program will read and analyze tracks 0 through 79.

tt : R xx

The tt is the track number, the **": R"** is the read mode specifier. This is for fast reading of unprotected tracks. The xx value is the quantity of sectors per track, values from 8 to 11 are allowed. The sectors should be normal MFM data with 512 bytes per sector. These tracks are efficiently read by the disk controller in the ST, without the need of any special twister format.

tt : D

The tt is the track number, the **": D"** is the read mode specifier. This specifies that the current track is read and analyzed by the **HART** chip. This is used to read tracks when you are not sure if they are protected or not. This type of reading takes about 5 times as long as the "R" type reading. This is the same type of reading that would be done for each track in the case where no backup control had been specified. However, this can be used to read from tracks higher than track 79, and it can be used to read from the second side.

tt : U t bbb

The tt is the track number, the **": U"** is the read mode specifier. This specifies that the program should do HART chip reading and analysis for the current track, and also look for data which may be "**under the index hole**". The **"t"** field is the type of reading. Currently only type t = 1 is supported. For t = 1, the program will attempt to synchronize by finding the logical end of the last sector on the track whose address/data field falls under the index hole. The end of writing is the end of the data field of this sector, plus **bbb** bytes. This permits copying some extra bytes that may follow the sector, if needed. With type t = 1, it is not mandatory to find data under the index. Therefore, this type of copying can be blindly specified for any track, but it will increase the time required for reading the disk by about 25%. When used blindly, set bbb = 0.

tt : W t gg bbbb sss

The tt is the track number, the **": W"** is the read mode specifier. This specifies that a portion of the disk will be read directly as flux transition spacings, and stored as direct flux spacing control bytes for the $000A writing command. There is no analysis, decode, checksums, or filtering performed which corresponds to any particular type of disk recording method. A large quantity of storage space is used by the resultant data. It is possible that if too much of a track is read using this method, and the data rate is faster than 500K bits per second, that the data will not fit in the buffers, and the program will not function. This will not usually happen with 3. 5" MFM double density. Even high density disks will not usually cause this.

One purpose for using this feature is to read and write sectors that have "**wandering weak bits**". These are *non repeatable CRC error sectors* that have some bytes that change, and some bytes that don't change that are mixed together. It is a special disk protection method that is only handled through the "W" reading and $000A writing.

Some copy protected tracks may *stretch the bits out*, causing certain timing values. This feature could be used to copy those tracks.

Another use would be to copy tracks that are not FM or double density MFM, such as MACINTOSH or high density.

This method should be avoided for reading normal sectors, as the loss of filtering, checksums and write pre-compensation makes the resultant sector data written less reliable.

The **"t"** specifies the type of synchronization used to switch to this method of reading. The **"gg"** specifies the quantity of good CRC sectors that are expected on this track without retrying. Since you must specify the expected quantity of good sectors expected, and since this quantity must match the actual quantity found on the source disk, the **"W"** type reading cannot be blindly applied unless **"gg"** is zero, and you read the whole track as flux spacings. The **"bbbb"** specifies the equivalent quantity of MFM bytes that are to be read. If the total flux spacing of all transitions read is added up and divided by 32E-6 seconds, this will approximately equal **"bbbb"**. This is not the quantity of bytes that will result in the buffer. The maximum value allowed is "bbbb"=6450, corresponding to reading from a drive that is turning about 3% slow. If the end of track is encountered before the desired byte count is reached, there is no error. This allows using a "bbbb" = 6450 to read in the whole track. The **"sss"** value must be 0 < **sss** < 2048.

For **t = 0**, the flux spacing reading method begins at the start of track, immediately upon the occurrence of the index hole from the drive. In this case, since no MFM bytes need to be read for synchronization, almost any type of disk data can be read. Please note that the practical limit for this program would be to read flux spacings that are at least 2µsec apart. This would include high density MFM data. In any case, the "bbbb" byte count is based on a 32µsec byte time, so specifying 6250 bytes, would actually read the equivalent of 12,500 bytes from a high density 500K bit / sec disk. The **"sss"** value is not utilized by type **t = 0**, but still must be in the correct range shown above.

With **tt : W 0 0 6450 1**, the whole track can be blindly read in as flux transitions. This could experimentally be used to copy certain tracks that couldn't copy in any other way, or could be used to copy **Macintosh** or **High Density** tracks that were aligned to the index hole.

For **t = 1**, the program switches to the flux spacing reading method, immediately following the **"sss"** occurrences of 3 of the special $A1 missing clock sync marks in a row. More than one of this type of "W" field may be on one line. In this case the **"gg"** quantity for both must be the same, and the **"sss"** for each is relative to the previous. The **"sss"** for the first **"W"** on a line is the quantity of mark triplets from the start of the track determined by the index hole.

R : tt

This is the repeat through track number command. It causes the previous command line to be repeated through track "tt". This substantially condenses the number of lines required.

End of a Particular Backup File Specification

Following all of the commands which control the reading process, a line which begins with the **right parenthesis ")"** terminates the current backup control file. The next line in the file **dbkupcf.s** must either be the title line for the next BC file, or the **end marker** below.

End of BC File Marker

Following the line which has the ")" which terminates the last BC file specification within the **dbkupcf.s** file, the very last line which indicates the end of the **dbkupcf.s** file is one that has the **right bracket "]"** as the first character on the line. Any lines after this in the file **dbkupcf.s** will not be accessed by the program.

Technical Information - Contents of the Mfmbtemp File

This section describes the contents of the file **mfmbtemp** that is produced by the **floppy to file** mode, and is read as input for the **file to floppy** mode.

The data in the file is binary, not ASCII. It cannot be edited by an ASCII text editor. **Happy Computers** does not currently supply an editor for this purpose.

The data in the file consists of blocks of data; each block begins on an even byte boundary. Each block begins with a code from $0000 to $000F. There should be an even quantity of bytes in the file. **Happy Computers** cannot currently provide further information on this data, beyond what is presented here.

$0008 s0tt nnnn nnnn

Each track's specification begins with this code. The quantity of bytes in this track's definition is **nnnn nnnn**. If **nnnn nnnn** was added to the location of the $0008 code word, this sum would point to the location of another $0008 code word for the next track, or the sum would point to the location of a $0007 code. The **s0tt** word specifies the side and track to be written. The side is in bit 15. If bit 15 of the **s0tt** word is zero, the first side of the disk is written. If bit 15 of the S0TT word is one, the second side will be written. The TT value is the track to be written

$0007

This $0007 marks the end of the specification for the destination disk.

$0009 SSGG 00gg

This $0009 code specifies that the track consists of normal MFM address and data fields. The following 512 * SS bytes are for a track with SS normal sectors. The gap between the data field and the next address field is GG bytes. This is the gap that follows the CRC1 CRC2 $FF after the data field, and precedes the 12 $00 before the address field sync. This code $0009 is the definition for an entire track; it must follow a $0008 code. The gg gap is the gap between the CRC1 CRC2 bytes of the address field and before the 12 $00 bytes before the data field sync marks.

Recommended values: (generated by the tt:R xx backup control command). If the destination drive RPM exceeds that shown, the track can't write.

SECTOR PER TRACK	GG gap	gg gap	bytes/track	MAX RPM	% over
8 (min)	176	22	6008	312. 1	4%
9	92	22	6003	312. 3	4%
10	38	22	6130	305. 8	2%
11 (max)	1	8	6182	303. 3	1%

NOTE: The 11 sector track can only be written with the HART chip, individual sectors can not later be re-written, since they will collide over the next sector's address field. These are "**read only**" tracks.

$000E aabb ccdd eeff nnnn

Must follow the $0008 code. This will be the code generated for tracks read by the HART chip. Only the byte needs to be accurate for track writing. The other values are statistics. STATISTICS (note: a minus byte value is an overflow value):

- **aa** is the qty of address fields found with good CRC.
- **bb** is the qty of address fields found with bad CRC.
- **cc** is the qty of data fields found with good CRC.
- **dd** is the qty of data fields found with bad CRC.
- **ee** is the qty of $A1 sync marks found that were < 3 in a row.
- **ff** is the qty of at least 3 $A1 sync marks in a row that did not precede an address or data mark.
- **nnnn** is the qty of MFM bytes plus byte spaces found during read decoding.

At 300 RPM and 4µsec per byte, this is usually 6250. This is not the quantity of bytes in the buffer; it tends to indicate the bit width and RPM of the drive/system that wrote this track. If the user modifies the length of data on the track, the **nnnn** value should be updated to represent the actual bytes contained, as the program will use this to match the writing length to the destination drive speed.

**** NOTE: The remainder of codes shown below will follow the $000E code.

$0000 nnnn . . . (nnnn bytes of DD data)

Following the header, there are **nnnn** bytes of normal double density data. **nnnn** may be odd, but the next block must start at an even address. These bytes do not have missing clocks. If FM single density data is read, it may appear as code $0000 blocks.

$0001 nnnn . . . (nnnn bytes of ccdd clock data)

Following the header, there are **nnnn**/2 pairs of bytes which are **cc** clock and **dd** data. These bytes may have a missing clock pattern according to the coding rules of MFM data. The last **ccdd** word in the block collected may not have a missing clock.

$0002 sssn ccdd

There is a short byte with only **n** bits with **cc** clock and **dd** data. The **cc dd** bytes will each be right justified with 8-**n** bits of leading zeroes in each byte. If the least significant bit of the **dd** byte is set, then the last transition of this short byte was at the data point in a bit cell. This 0002 code is generated to complete a short byte when there is => 2 bit times (about 9 µsec) between flux transitions.

The **sss** 12 bit portion will be 100x xxxx xxxx if there is some flux spacing value between transitions x xxxx xxxx, where this 9 bit value is just like a $0003 code had followed the $0002 code. The **sss** 12 bit portion will be 0xxx xxxx xxxx if there is no transition for at least xxx xxxx xxxx count, just like having a code $0004 following the $0002, with a limit of $7FF for the code $0004. The code $0002 is generated when there are some normal MFM bit spacings followed by excessive flux spacing.

$0003 nnnn

There is an interval of **nnnn** * 62.5E-9 ns between flux transitions. This is generated by reading analysis when a flux spacing exceeds that which is usually allowed for MFM, = 2 bits times, or > 8µsec. The maximum value generated by the reading analysis will be $13E, or a count of 318, which is 19.875 µsec. For user generated code, the $0003 will accept **nnnn** values between 50 ($32) and 390 ($186), or 3.125 µsec to 24.375µsec. Use of the $000A will provide greater efficiency and flexibility for user generated flux spacing values. A $0004 code will be generated by the reading analysis when the spacing exceeds $13E. A post reading analysis process could compress groups of $0003 and $0004 type codes into a smaller number of bytes in the $000A code, but this is not done by this program.

$0004 nnnn

There is an interval of **nnnn** *62.5E-9 ns with no flux transition. **nnnn** will be greater than $FF and no more than $FFFF (1 word). Reading analysis will generate a value between $100 and $FFFF. During writing, values of $64 to $FFFF will be accepted. It may be more efficient and accurate to use the "W" type reading, and $000A type writing for non MFM code spacings.

$0005 nnnn

This marks the special $A1 MFM sync byte with a missing clock bit as expected. **nnnn** is the quantity of bits of the previous byte that were really part of the sync mark (0-7).

$0006

This will only precede a $0005 code. It indicates that the sync mark needs to be advanced in writing by 1/2 bit cell time. It is caused by having the sync mark detected in the clock pattern rather than the data pattern.

$000A nnnn nnnn wwww wwww . . . (nnnn nnnn data bytes of xx)

Switch to direct flux spacing control mode. The **wwww wwww** field has the total HART chip count spanned by the flux spacings generated, which is useful during read analysis to determine this section's contribution to the track length, and is unused during writing. **nnnn nnnn** is the quantity of bytes in this block = block length.

The **xx** bytes specify the spacing between flux transitions. Each count is 62.5 nanoseconds. The nominal 3.5" DD bit width is 64 counts = 4µsec. Flux spacings normally generated from MFM data block codes 0000, 0001 and 0002 include filtering and write pre-compensation, and adjustment to match original the track length. The flux spacings written by the $000A are taken verbatim, with none of these adjustments. Valid counts, when a single byte specifies a flux spacing, are 25 to 255, corresponding to 1.5625µsec to 15.9375µsec spacing.

A $00 byte specifies that the 2 bytes following the $00 are (hi lo) a count in which there is no flux transition. The "W" reading can generate values from $00 $00E8 to $00 $FFFF. The $000A writing will accept values from $00 $0064 to $00 $FFFF.

A $01 byte specifies that the 2 bytes following the $01 are (hi lo) a count at the end of which there is a flux transition. The "W" reading can generate values from $01 $0100 to $01 $013E. The $000A writing will accept values from $01 $0032 to $01 $0186. It is wasteful to use the $01 three byte sequence for counts < $100, since a single byte could do this.

Byte values $02 to $18 (2 to 24) are not allowed, unless they are in the hi lo pair following a $00 or $01.

$000B xxxx

This code is utilized during read analysis. It is not utilized during the writing process. It follows the occurrence of 3 type $0005 sync mark code blocks. The **xxxx** will be 0000 if the field following is neither an address nor a data field. The **xxxx** field is $FFFF if the field following is a data field with a bad CRC. The **xxxx** field is $7FFF if the field following is an address field with a bad CRC.

$000C nnnn

During reading analysis, some $000B code blocks are replaced with $000C code blocks. It specifies that the first **nnnn** bytes of the following $0000 code block were checked and did have a valid CRC, which is contained in the buffer at the usual location of CRC bytes. The length includes the data or address mark byte and the CRC bytes, so a 512 byte data sector needs a length of 512 + 1 + 2 = 515. An address field needs 7.

If the $0000 data block following the $000C is modified, the $000C should be replaced with a $000D to allow the program to recompute the correct CRC, or a verify error will occur. In the case of data fields, the read analysis will place a $000C over the $000B, only if a verify read back with the FDC will succeed.

The requirements include the following:

1) valid data field mark & CRC computed based on length in address field
2) data field follows an address field with valid CRC
3) data mark follows an address field's CRC within 43 bytes

$000D nnnn

This code is not generated by the reading process. It can be inserted by the user to cause the program to generate valid CRC's when the user has created or edited the data. The length includes the data/address mark byte and the CRC bytes, so a 512 byte data sector needs a length of 512 + 1 + 2 = 515. An address field needs 7. A $0000 code block of sufficient length must follow this block. The first byte of the $0000 block following, must be either an address mark $FE, or a data mark $F8 thru $FB. An address field will specify a track and sector number that may be verified. If the data field has a $000D code preceding it, a verify/read back will be done, so the address and data field must form a proper pair, or a verify error will occur.

$000F wwww

This code instructs the writing procedure to delay the start of writing by **wwww** bytes after the index hole. This delays the start of writing, allowing the data stream written to continue under the index hole. The **wwww** qty does not enter into the track length matching calculation scheme. The $000F code block must be the last block code for the track, preceding $0008 or $0007. **wwww** is 2047 maximum. A byte is considered to be 32µsec here, or 1/6250 of a revolution.

OPTIONAL DISPLAY?

There are some optional display modes built into the ***dmfmbkup*** program. Currently, the information displayed is only for use by **Happy Computers**. Future versions may document the information displayed, and how to control the display process.

WRITING SPEED IMPROVEMENT

The actual speed improvement obtained using the HART chip to format, write, and verify is detailed below. WITHOUT HART CHIP, 4 revolutions required

1) 1 revolution = move head to next track, plus head settle delay
2) 1 revolution = write the address fields of the track (format)
3) 1 revolution = write the data fields of the track
4) 1 revolution = verify the data written

With the HART chip, this is reduced to 3 revolutions, since the operation of writing the address and data fields is done in one revolution. A "software only" based program cannot be as fast unless it does not properly verify.