EHANCEMENTS / BUG FIXES

Version 1.03 of MF

public release: December 15, 1993

KNOWN BUG FIXES:

- Fixed: mfReadList - continuation's did not work.
If you thought you could call readlist for a continued list of records -- it wouldn't have worked... It does now. A 'c' sample is included that demos the readlist and would show the bug that was in the readlist function.

- Fixed: NULL index records not deleting from indexes.
If you added a 'string' to an index and it was ALL NULLS it would not delete itself when you called mfDelete. Since a fully-nulled string was used to specify an 'empty' index field, the system did not handle it properly. On an mfSeek, you would still find the deleted record.

This may pose a problem for those of you with legacy data. The only option to fix it is to reindex everything.

CHANGES:
MF header -
support added to the MF header for C++ compilers.

MF.LIB -
Due to a fact (unknown to me till recently), if you don't explicitly export the functions in your DEF file, then the 'ordinal' values for the functions can 'change' every time you recompile. This means that programs compiled for older versions of MF.dll may not work for newer versions. I have fixed the .DEF to export a constant ordinal value for every function in the MF.dll. Hopefully, this will prevent problems down the road. It may also explain (to some of you) why the new versions of the DLL always required you to recompile your code to test them out. Sorry about the confusion.

ENHANCEMENTS:

- A really lousy C examples is now included.
In testing some SP functions, I threw together this little test application. May help document the SP functions -- or maybe not. But, at least it shows read/write/readlist.

I tried diligently to get a C example in to this release, but it never made it to completion...

Acknowledgements:

It only seems fitting that I say thanks to a few people. MF has been wildly unsuccesful, but a few people have helped find more bugs than I care to admit have been released:
Kenneth A. Argo

Who developed a C++ class wrapper for the MF db and patiently explained to me what bugs he had found.

Jerry Zemo
For being an interesting guy to type to. He never found a bug -- but, always had something interesting to say.

PsL
For publishing MF in its disk catalog. If they hadn't decided to do it, I don't think I would have continued doing this.

All the registered users (w/ and w/o distribution rights...)
For making this whole thing worth while. Thanks!

Version 1.02 of MF

KNOWN BUG FIXES:

- Fixed a potential GPF on an index change. Would probably NOT GPF, but the math was screwed up and it could (theoretically) GPF. (OK, OK, it DID GPF -sometimes- so, we finally tracked it down...)

ENHANCEMENTS:

New functions or enhanced functions:

mfIsDeleted - Checks if a record has been deleted

mfReIndex - Rebuild the indexes from data

mfLock
mfUnLock - Record locking/unlocking (alpha!)

mfInfoDB - Enhanced to have 'real' record count and high-water mark...
Requires one new parameter...
- Now supports NULL as a parameter if you don't care for the return value of a specific item
e.g.
If you don't need to know the recsize, you don't have to declare a dummy variable to recieve its value. Just pass a NULL for that parameter.

CHANGES:

- All error codes changed to mfERR_cccccccc. If this requires you to change your code, we're sorry. However, a global search and replace for ERR_ with mfERR_ should solve most problems. It seemed appropriate to be able to distinguish the 'errr' code by product, so we stuck the mf in front of the error codes.

NOTES:

- DB format lost compatibility with old format.
To facilitate the ReIndex and the IsDeleted functions, the database structure had to be changed. A conversion exe, dll and source code (in the CSAMPLES\CONVERT) have been provided to ease the transition. You can use the method most appropriate for your application.

- DB record now 8 bytes longer than before. (i.e. 8 more bytes of overhead per DB record...)

- This is a short-term interim release. Since we wanted to get the GPF fix out -- we put this out early.

Fixes in version 1.01 of MF

KNOWN BUG FIXES:
PROBLEM: Invalid data being loaded when more than 20 files are open.
SOLUTION: Fixed.

PROBLEM: mfReadList behaved unpredictably if you passed a fuzzy length greater than -1(e.g. 0) and specified a RECORD to start at -- but didn't specify a string. (The DOCS said to do it that way for RECORD sequential lists...)
SOLUTION: Now, if you pass a NULL as the fuzzy string, but specify a match length of ZERO (as the DOCS say), it will work properly.

Fixes in version 1.00 of MF

CHANGES:
- mfReadList now available. 1st of a number of severe-performance functions
- Modified BCARDS demo to show mfReadList speed enhancements
- Fixed garbage characters at front of database.
- mfInfoIndex now available
- (more) documentation, nothing urgent if your familiar with MF.
- minor mods to the C Header file (mf.h)

KNOWN BUG FIXES:
PROBLEM: The C header file specified some long and int pointers but didn't specify them as FAR pointers. This caused GPF's on anything compiled in anything but the LARGE memory model. (C only...). (Thanks to: Dan Schless)

SOLUTION: We put the FAR declarations into the header file...

PROBLEM: *** mfDelete and mfSkip ***
A word of caution about these two (when used in a 'loop'). Lets say you do an mfDelete(aRecordNum,...) and then try to do an mfSkip(aRecordNum,1,....). Since all the 'key' information about the previously deleted record has been removed, this WONT work! While this isn't a bug, it is kind of a nuisance if you aren't aware of it.

SOLUTION:
The general problem sneaks up like this:

```
' This loop should delete everything in the database that
' does NOT contain "Something Specal..." in aField...
lcurRec = mfTop(...)
do while lcurRec > 0
      mfRead(...)
      if bufferRead.aField <> "Something Special..." then
            mfDelete(lcurRec,...)
      endif
      lCurRec = mfSkip(lCurRec,1,...)
loop
```

In order to correct this problem, you could rewrite the loop like this:

```
lcurRec = mfTop(...)
do while lcurRec > 0
      mfRead(...)
      lOldRec = lCurRec
      lCurRec = mfSkip(lCurRec,1,...)
      ' Delete comes AFTER knowing what our next record will be...
      if bufferRead.aField <> "Something Special..." then
            mfDelete(lOldRec,...)
      endif
loop
```

NOTE: This did lead to discovering a 'bug' in the mfSkip code. Again, it is only 'sort' of a bug -- We did not validate that the record you are passing us was 'valid'. Being the gullible people we are, we thought you'd always pass a valid record. Anycase, the the Skip function would return a random 'record' when you called this function with a deleted value. Now, it will return MF_EOF.

Fixes in Version 1.00.01 BETA of MF.

CHANGES:
- OLD databases are not 100% compatible with the new database format. They may appear to work, but they will, more than likely, die a horrible death. If it's worth keeping the old data, you'll need to make a little conversion utility to suck the data in. Very sorry.

- The FIRST record in a database is now record 1.

- Improved 'bad-parameter' error checking - previously, a couple of functions where susceptible to bad task/handle/index parameters. They now kick back an error on a bad one. (As opposed to locking or GPFing)

- mfInit parameters modified: Now takes a structure pointing to an extension DLL. (or, list of extension DLL's). See mf.wri (mfInit) for documentation on this feature.

- User-defined keys (UDK) now supported. CSAMPLE\UDK contains a sample DLL for creating a UDK.

- Minor enhancment (speed wise) for Deletes and appends...

- Minor slow-down on repeated KEYS in anticipation of 'one->many' auto-indexing (e.g. previously, if you passed a index-key and it was the same as it was before, the system would not bother to check the index for the key. Now, it checks it even if the key matches...)

- - - Other Stuff - - -
- Severe-Performance functions pushed to release 1.1. Sorry.
- Better documentation? (I don't think it could have gotten any worse...)
- (Slightly) improved VB Sample. Demonstrates UDK's and multi-file support

KNOWN BUG FIXES:

PROBLEM: GPF on Record size > 8K?
SOLUTION: Yep. A sneaky bug got in and we weren't allocating enough storage for records larger than 8k. Sorry...

PROBLEM: Disappearing records / Reappearing records / Reused records on appends
SOLUTION: mfDelete caused alot of havoc. Inadvertently, we switched the record 'delete' ptr to a new value. When 'Record 0' was deleted, it hosed the delete tables. (Also, we killed the 'Record 0' being a record. Records now start at 1).

PROBLEM: Databases that contain only an INDEX get corrupted. (e.g. You defined a database with a data size of 0 bytes)
SOLUTON: Fixed. (See also WORKAROUNDS -- a Database MUST have at least 4 bytes total (inclusive of the index's))

WORKAROUNDS:

PROBLEM: Every database I have seems to work except a small linked list I have, is there some problem with small database? (Everything works fine until I delete something...)

SOLUTION: The MINIMUM size for a record is 4 BYTES. The size of your index(s) applies towards this minimum. If you are only storing 1,2 or 3 bytes in a DB, you will get an error when you start to DELETE records. mfCreateDB has been fixed to generate an ERROR if you try to create a database that is too small.

e.g.:
Where recSize refers to the 'size' of the 'DATA' portion of the record.
Where indSize() refers to the size of the 'index' for the index #.

Valid:	recSize = 0	4 bytes total
	indSize(0) = 2	
	indSize(1) = 2	
Invalid:	recSize = 0	3 bytes total
	indSize(0) = 2	
	indSize(1) = 1	
Valid:	recSize = 3	4 bytes total

indSize(0) = 1