

You may distribute this document in its entirety and/or use this information for non-commercial or education use without the author's permission. This document and the implementation of this specification may not be used commercially unless you obtain prior written consent from the author.

This is a small attempt at a small thing I call the "Enhanced Master Boot Record". It is an attempt at something I thought of a while back and have started to implement to see if it will work to my specifications and desires. Your comments are welcome at [fys \[pigtail\] fysnet \[dot\] net](mailto:fys@pigtail.fysnet.net).

A Brief Summary

Since the traditional style of a partition entry in the Master Boot Record (MBR) allowed for only eight bits of identification, this soon became unusable due to many different File Systems and uses. When multi-boot programs started to appear, there was little or no way of displaying what kind of file system and Operating System was on that partition.

The MBR also has a limit of 2^{32} sectors for a partition size and offset from the start of the drive. With drives becoming much larger than this limit, it is time for a different type of volume partitioning scheme.

This eMBR overcomes the single byte identification limit by allowing up to 64 characters to identify the partition. This partitioning scheme allows for a partition to start as far as 2^{64} sectors from the start of the media and 2^{64} sectors in size. To put this into a more understandable number, if the sectors are 512 bytes each, this will allow for a disk size of $2^1 * 2^{64} * 2^9$ bytes. This is a total of 2^{74} bytes. This is 16-Zetabytes of storage. See Table 1 in the Definitions section below for more information on Zetabytes.



Please note to access 2^{65} sectors, you have to have a processor and a drive that can handle more than 64-bits.

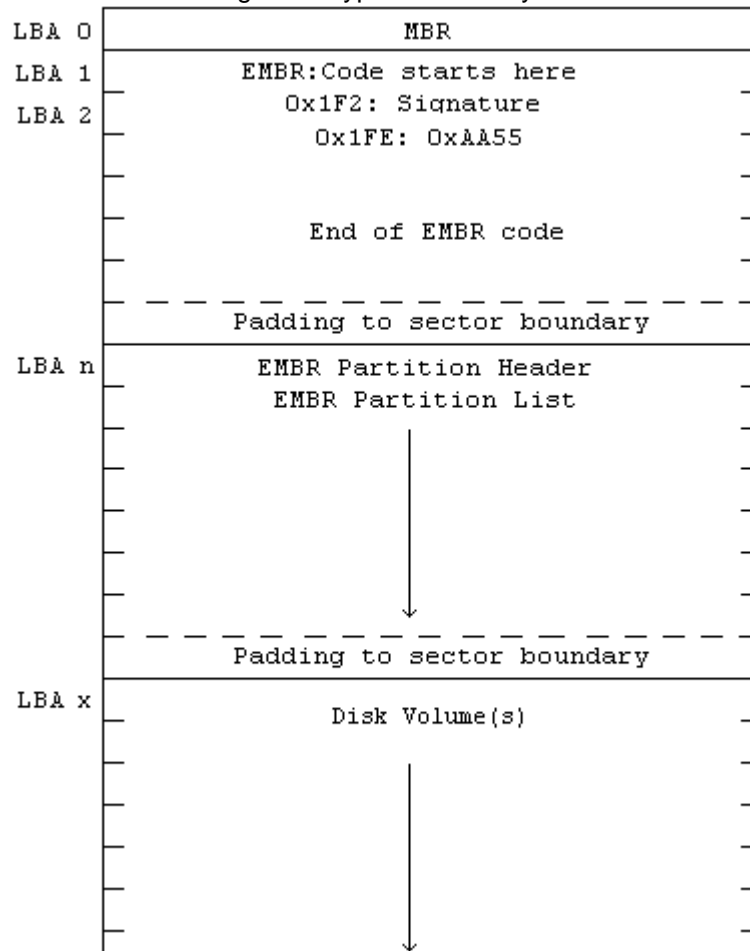
An Overlook of a Typical eMBR Partitioned Hard Drive

A typical eMBR partitioned media device would contain two or more partitions, each just after the other, all listed in the eMBR headers. At boot time, the eMBR application would load the specified amount of sectors into memory to be able to enumerate the partitions, display the partitions in a form that the user could choose which partition to boot. The eMBR app would also watch for a key press from the user. If a key press was not found within the specified amount of time, the eMBR app would boot the partition that was most recently booted.

This specification will show the required format and steps to create and boot the eMBR partition scheme and show the format of the partition entries.

Figure 1 on the next page shows a typical layout for an eMBR partitioned disk.

Figure 1: Typical eMBR layout



Definitions

The following words might be used within this specification.

The words *should*, *shall*, *will*, or *must* are used to indicate mandatory requirements. This means that it is mandatory for the eMBR app to follow.

The words *may*, *might*, or *can* are used to indicate that it is recommended or is a possibility for the eMBR app to follow. The word *may* is usually referred to as; *is permitted to*. The word *can* is usually referred to; *is capable of*.

Little-endian is used to indicate that a value larger than a byte is stored lowest byte first, then the next lowest byte, and so on. This is considered the Intel format. All values within this specification are written as and must be read as Little-endian format.

Any string literal enclosed in double quotes is written as string and when enclosed in single quotes, is written as a little-endian value. E.g: "EMBR" has the 'E' written first, then the 'M', 'B', and the 'R' last. 'eMBR' is written just the opposite.

Disk addressing is in the form of Linear Base Addressing, or LBA, and is zero based from the start of the media. Cylinder/Head/Sector (CHS) addressing is only used in the MBR at LBA 0.

All memory marked reserved must be preserved. This means that any value read that is reserved and preserved must be written back as the same value.

Table 1 on the next page shows the Binary Multiples and their names used within this specification.

Table 1: Binary Multiples

Name	Symbol	Value
Kilo	k/K	$2^{10} = 1,024$
Mega	M	$2^{20} = 1,048,576$
Giga	G	$2^{30} = 1,073,741,824$
Tera	T	$2^{40} = 1,099,511,627,776$
Peta	P	$2^{50} = 1,125,899,906,842,624$
Exa	E	$2^{60} = 1,152,921,504,606,846,976$
Zetta	Z	$2^{70} = 1,180,591,620,717,411,303,424$
Yotta	Y	$2^{80} = 1,208,925,819,614,629,174,706,176$

Media Format

The format of the first few sectors of the media, further known as drive, must follow the outline within this specification. Once the eMBR format is complete, the format of the remaining sectors of the drive is outside this specification.

Sector Size

This specification does not limit the sector size to 512 bytes per sector. However, please note that with the MBR explained below and the offset of the Signature Block within the first sector of the eMBR partition, the sector size is assumed 512. Therefore, if the sector size is more than 512, these offsets remain as they are indicated within the text below.

Master Boot Record

The first sector of the drive must contain a standard Master Boot Record. The format of this MBR must follow the format designed for PC DOS 2.0 in March of 1983. This MBR must contain the four partition entries, with the first one starting at offset 0x01BE. This MBR must contain the boot signature at offset 0x01FE, a byte of 0x55 at offset 0x01FE followed by a byte of 0xAA.

The first partition entry must point to the second sector of the drive, LBA 1. The partition table entry must have the values listed in Table 2 below.

Table 2: 1st Partition Entry Format

Partition Entry Format			
Offset	Size	Value	Description
0x00	1	0x80	Boot Indicator
0x01	3	0 2 0	Starting CHS Value
0x04	1	0xE0	System Type
0x05	3	FF FF FE	Ending CHS Value
0x08	4	0x00000001	Starting Sector
0x0C	4	0xFFFFFFFF	Partition Size in Sectors

The values in the other three partition entries are not defined within this specification. The system may use the other three entries to boot other partitions. This specification does not require that the eMBR be the only bootable partition. However, see the notes on “other partitions entries” below.

The Boot Indicator must be 0x80 to indicate that we want to load and boot the eMBR. However, if the system is set to boot one of the other three entries, the Boot Indicator must be 0x00.

The System Type must have a value of 0xE0.

The Starting and Ending CHS values must be the first Cylinder, first Head, and second Sector for the Starting and may be values of 0xFF, 0xFF, and 0xFE for the Ending.

The Starting Sector value must point to the second sector, which is LBA 1. The Sector Count value may be 0xFFFFFFFF.

Please note that it is not a requirement for the ending CHS and Sector Count fields to be the values defined above. If the drive is not larger than the amount specified, you must use the actual size values to denote the size of the drive. However, if the drive has less than 2^{32} sectors, there would be little reason to use this specification.

Other Partition Entries

This specification does not require the other three partition entries in the MBR at LBA 0 to be unused. These entries may point to partitions outside of the containment of the eMBR. These entries must not point to partitions within the containment of the eMBR.

eMBR Partition

The eMBR partition must start at LBA 1. Within these first few sectors is the code to load the remaining sectors that contain the eMBR partition entries, along with the code to display and allow the user to select which partition to boot. This specification does not limit the amount of sectors reserved for this code other than the size that will fit in a 16-bit word, 65,535. It is recommended that the remaining sector count is not to exceed 61 sectors. See below why this value should be much less than 65,535.

The first sector of the eMBR Partition must contain the following information at offset 0x01F2, shown in Table 3 below.

Table 3: eMBR Signature Block

Offset	Size	Value	Description
0x01F2	8 bytes	'EmbrrbmE'	EMBR Signature
0x01FA	2 bytes	varies	Sector offset to header block
0x01FC	2 bytes	varies	Count of remaining sectors
0x01FE	2 bytes	0xAA55	Standard boot signature

This Signature Block is used to indicate whether there is a valid eMBR code block at this location, the size of this code block, and the pointer to the eMBR Partition Entry Header Block.


The first 64-bit value must be 'EmbrrbmE' which is the value of 0x456D627272626D45. The eMBR app must verify this value and the value at offset 0x01FE to be correct before proceeding. It is outside of this specification for the process to take if these values are not correct.

It is recommended, but not required, that the MBR at LBA 0 also check these values before passing control to this code block.

The value at offset 0x01FA is a 16-bit word value indicating the sector offset, relative to LBA 0, of the eMBR Partition Entry Header Block defined later in this specification. This value points to the LBA that contains the Header Block that defines the Partition Entries.

The value at offset 0x01FC is a 16-bit word value indicating the remaining sectors that must be loaded to include all of the code (not counting this first sector), the eMBR Partition Header, and all of the eMBR Partition Entries that are within the containment of this eMBR partition. The eMBR app must load these sectors into memory, following the current sector, to be able to read and parse the available entries to display and boot.

The remaining area from LBA 1 to the LBA specified at offset 0x01FA, not counting the Signature Block specified above, is for the use of the eMBR app. This area must contain all of the code and data needed to parse, display, and boot the desired partition.

 It is recommended that you included a few unused sectors at the end of this code block for future improvements. It is also recommended that you included a few unused sectors at the end of the partition entry list for future partition entries. i.e.: The value at offset 0x01FA be a

few more sectors than is needed for the code, and the value at offset 0x01FC be a few more sectors than is needed to hold the current count of partition entries.

eMBR Partition Header

The eMBR Partition Header will be at the LBA specified in the 16-bit value at offset 0x01FA in the Signature Block detailed in the previous section. This block and all remaining partition entries should have been read into memory with the code specified in the previous section. The size of this block containing this eMBR Partition Header and all Partition Entries will be the value in offset 0x01FC, in the Signature Block above, plus 2 for the MBR and sector already read, minus the value in offset 0x01FA, times the size of a sector.

The eMBR Partition Header, detailed below, contains the information needed to know how many partition entries are included, the boot delay time to wait for user input, and a 32-bit CRC to verify validity of the entries. This header is shown in Table 4 below.

Table 4: eMBR Partition Header Block

Offset	Size	Value	Description
0x00	4 bytes	"EMBR"	First EMBR Header Signature
0x04	4 bytes	Varies	32-bit CRC of Header and Entries
0x08	2 bytes	Varies	Total Entries that follow
0x0A	1 byte	Varies	Boot Delay in Seconds
0x0B	1 byte	1.05	Version of eMBR (binary)
0x0C	8 bytes	Varies	n = Total Sectors (0 -> n)
0x14	8 bytes	Zeros	Reserved and preserved
0x1C	4 bytes	"RBME"	Last EMBR Header Signature

The eMBR app should verify the two 32-bit signature values at offset 0x00 and 0x1C along with verifying the 32-bit CRC of this header and the following partition entries. The CRC check follows that of the official CRC-32 standard described at the following URL under the CRC-32 name using the 0x04C11DB7 Polynomial.

http://en.wikipedia.org/wiki/Cyclic_redundancy_check

The CRC check is done consecutively on this 32-byte block and the amount of memory following sized by the value at offset 0x08 times the size of a partition entry defined below. If a partition entry is marked invalid, the CRC check is still done on that entry. For example, if the total count of partition entries is five (5), the CRC check would be done on the memory starting at this Partition Header Block and for the bytes that follow with the length calculated as:

$$(\text{size of this header} + (5 * \text{entry size})) = \text{total size to check}$$

The size will always be a multiple of 32-bits.

The value at offset 0x08 indicates the amount of partition entries that follow this header. The entries are consecutive, one after the other. There are no gaps between entries.

The value at offset 0x0A is the amount of seconds the app should wait before it boots an entry if no user input is found.

The value at offset 0x0B is the version of this document the volume supports. The high 3 bits are the major version, with the lower 5 bits as the minor version. If this document is version 1.05, it would be stored as a byte as 00100101b in binary, 0x25 in hex, or 37 in decimal. The first version of this specification had this byte as zero. Therefore, if this value is zero, a version less than 1.05 must be assumed.

The value at offset 0x0C is the total amount of sectors that this eMBR encompasses. In other words, this will be the count of sectors from LBA 0 to the last used sector this eMBR describes. This can also be used as the LBA of the first sector after all the sectors this eMBR encompasses.

eMBR Partition Entry

All eMBR Partition Entries follow the Partition Header described in the previous section. Each partition entry is 128 bytes in length and contains the format shown in Table 5 below.

Table 5: eMBR Partition Format

Offset	Size	Value	Description
0x00	4 bytes	Varies	Flags
0x04	4 bytes	'eMBR'	Signature
0x08	8 bytes	Varies	Base LBA
0x10	8 bytes	Varies	Sectors
0x18	64 bytes	Varies	8-bit UTF description
0x58	8 bytes	Varies	Created: Secs since 01Jan1980
0x60	8 bytes	Varies	Last Boot: Secs since 01Jan1980
0x68	8 bytes	Varies	OS Signature
0x70	16 bytes	Zeros	Reserved and preserved

A single partition entry is used to gather information about a partition. The value at offset 0x00 uses bit 0 and bit 1. The other bits of this field must be preserved. A value of 1 in bit 0 indicates that this partition entry is valid and should be enumerated and listed in the list of partitions to boot. A value of 1 in bit 1 indicates a must to hide this partition from the list of partitions to boot. Bit 1 overrides bit 0. However, if bit 0 is set, this entry is still valid even though it may be hidden. You should also check the Signature value to be sure this is a valid partition entry.

The 64-bit value at offset 0x08 is the Base LBA of this partition relative to LBA 0, the start of the drive. The 64-bit value at offset 0x10 is the size in sectors of this partition.

The 64-byte field at offset 0x18 is an 8-bit UTF description string used for displaying within the list of partitions to boot. Even though this is UTF-8 formatted, it must be zero terminated.

The 64-bit values at offset 0x58 and 0x60 are the elapsed seconds since 12:00:00am January 1st, 1980 when the partition was created and the last time this partition was booted, respectively. The Created value must not be modified any other time than time of creation. Creation means the time the partition was formatted to a specific file system. The Last Boot value must be updated just before this partition is booted.

The 8-byte field at offset 0x68 is the OS Signature block. The format of this block is not specified in this specification. However, it is recommended that the format of this field be used as shown in Table 6 below.

Table 6: OS Signature

Bits	Description
63:48	OS Signature
47:32	FS Signature
31:16	Partition usage. OS/FS specific/defined.
15:00	OS specified: partition order/cleanup

The last field in the Partition Entry must be preserved when written to.

Requirements

It is not required that all partition entries be valid. However, any valid entry must be at an offset of 128 bytes from the last entry with the first entry following directly after the Partition Header.

A partition must not overlap another partition. Any sector contained within one partition must not be contained within another partition.

The size of a partition does not have to occupy any sectors after that partition and before the next partition. Any sector count between partitions must be considered lost and unusable. This space must not be written to or if read from, not considered to be valid data.

The first partition, the partition with the lowest starting LBA, is not required to start directly after the Partition Entry List.

The booted entry must have its elapsed time updated to the current elapsed seconds since the Epoch date specified within this specification just before the eMBR app transfers control to the loaded partition. Other than this item, it is outside this specification on whether the remaining values within the partition entry and Partition Header are modified. However, please note that when you modify an entry, you must update the CRC-32 in the Partition Header.

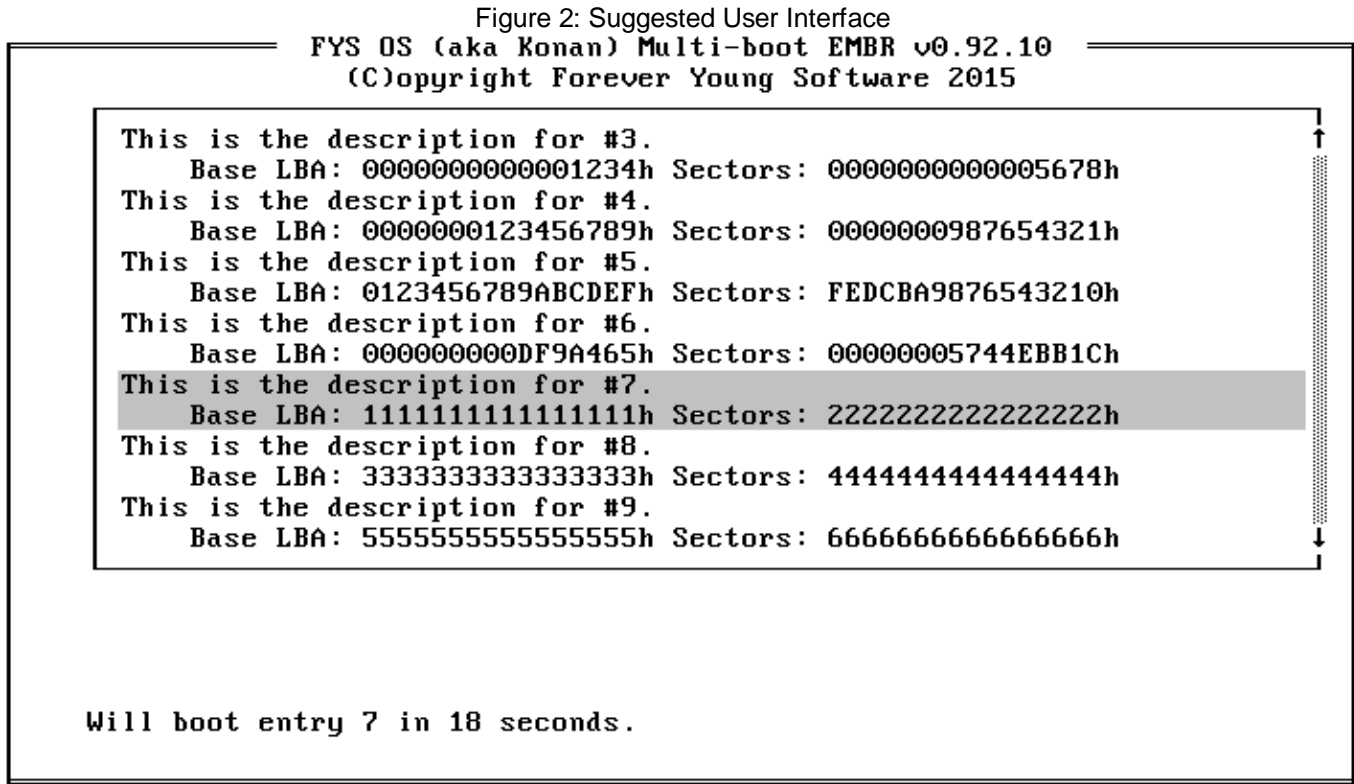
It is outside this specification on how a user adds, deletes, or modifies a Partition Entry, as long as the resulting modification follows the format listed within this specification.

All LBA's used within this specification, unless otherwise noted, are relative to the start of the drive, LBA 0.

When a value is listed within single quotes as in 'eMBR', this is written to the drive in little-endian format. The last character is written first, then the next to last, and so on, with the first character in the string written last. For example, the 'eMBR' will be written as a 32-bit value as 0x52424D65. If there are only two characters in the string, it is a 16-bit value. Eight characters in the string is a 64-bit value.

The User Interface

It is outside of this specification on how the user interfaces with the eMBR app or how the eMBR app displays the partitions to boot. It is suggested that a list be shown similar to Figure 2 below.



It is suggested that at the first point of user intervention, a key press, that the count down to boot be stopped and the interface wait for the user to continue. It is also suggested that the eMBR app allow the user to adjust the count down time for future boots.

It is suggested that the user be able to indicate to the eMBR app to display the OS Signature Block, Table 6, within the Partition Entry.

Things to Consider

Since this technique of media partitioning allows for a much larger media sizes than what was previously available, you will need to take the following notes into consideration.

FAT file systems:

Since the FAT file system contains a “hidden sectors” field in the BIOS Parameter Block, indicating how far away from the start of the disk this file system resides, take in to account that this BPB field is only 32-bits in size. Therefore, you may not be able to boot from this partition, and depending on how your file system driver is configured, you may or may not be able to read from this partition, if this partition is located beyond this 32-bit LBA size.

All systems:

Depending on your platform, you will need to have 64-bit unsigned integer storage to use this technique of media partitioning if you have more than 0xFFFFFFFF (4,294,967,295d) sectors.

Release Dates

01 Nov 2018

- Added the Size field in the Header. Doesn't break or effect anything other than an implementation that supports this new version, can grab this value to see how much of the drive this eMBR encompasses.
- Corrected the in-error value of offset 0x2C in this header. Should have been 0x1C.

28 Sept 2015

- Updated the source code, fixing a few errors, and adding functionality. No Specification changes were made.

03 Jan 2015

- Cosmetic changes to this document. No Specification changes were made.

06 Oct 2013

- Minor clarifications. No functionality was changed.

30 June 2013

- Official release as a specification document.

14 June 2011

- Initial release.