DALLAS SEMICONDUCTOR

Application Note 114
TMEX Extended File Structure
Revision 3.10

I. TMEX EXTENDED FILE STRUCTURE
The Extended File Structure is the foundation of the Presentation Layer of iButton–TMEX. It provides a directory structure for iButton™ data, allowing named files to be randomly accessed as they are on a diskette. The "Book of DS19xx iButton Standards" provides a basic specification of the Extended File Structure. This document extends the basic definition to include bitmap files for large capacity iButtons, multiple sub–directories, extended file attributes, passwords, date–stamps, owner identification, and other useful constructs.

The definitions and rules of the Extended File Structure are sufficient to store multiple files in nested directories using device capacities up to 64K bytes. These devices may be organized as 4...256 pages of 32...256 bytes. The rules given in this document are scaleable to all combinations of memory organization within this range. (Note that in all discussions of iButton data structure, the first available page of memory is called page 0.)

A. Data Organization
The data organization of the Extended File Structure is very similar to that of a floppy disk. A sector of a floppy roughly corresponds to a page of an iButton. The directory tells which files are stored, where the data is located in the device, and how many pages it occupies. In this way information can be randomly accessed for quick response.

The organization of data within a page of a file or directory is shown below. (The numbers shown assume a 32 byte page length, but the same structure applies for devices with page lengths up to 256 bytes.)

DATA PACKET Figure 1

length binary 1...29 1 byte	data ASCII or binary 0 to 28 bytes	cont.–pointer binary 1 byte	$\overline{\text{CRC16}}$ binary 2 bytes	not used 28 to 0

Each page of a file or directory begins with a length byte, contains a continuation pointer, and ends with an inverted CRC16 check. The continuation pointer is the page address where the file or directory is continued. A continuation pointer value of 0 marks the last page. The length byte indicates how many valid bytes a page contains, not counting the length byte itself or the CRC. The CRC calculation, however, also includes the length byte. The CRC accumulator is initialized by setting it equal to the iButton page number. Every byte of a page is transmitted to or from an iButton least significant bit first. The length byte is the first to be transmitted. Of the two CRC bytes, the least significant will be sent first.

Each memory iButton must be formatted before it can be used with the extended file structure. During the process of formatting, the root directory file is created. The root directory always begins in the first page of the iButton (page 0). **The organization of data within the first page of the root directory** is shown below:

ROOT DIRECTORY START Figure 2

length binary 8...29 1 byte	control data 7 bytes	file entries ASCII & binary 0 to 21 bytes	cont.–pointer binary 1 byte	$\overline{\text{CRC16}}$ binary 2 bytes	not used 21 to 0

directory mark "AA" 1 byte	re–served "00" 1 byte	bitmap control xyyyyyzp 1 byte	bitmap of used pages or address of bitmap file 4 byte binary number LS–byte/ / /MS–byte

Instead of data, the directory contains management information and file entries. The control field of seven bytes has the same length as a file entry. The bitmap of used pages serves TMEX as a fast lookup table to find free pages. In this bitmap, used pages are marked with a 1, empty pages with a 0. The least significant bit corresponds to page 0. This local bitmap is only used for non–EPROM devices with less then 32 pages of data. All other devices have remote bitmap files. In large NV–RAM iButtons the bitmap file page number refers to normal data space. In EPROM devices the page number refers to status memory. For EPROM devices the unprogrammed state is 1 and the programmed state is 0. Due to this constraint the bits in the EPROM bitmaps are inverted. An empty page has a 1 and an occupied page has a 0.

The most significant bit, “x”, of the bitmap control byte specifies whether the bitmap is stored immediately in the first directory packet or in a separate file. If this bit is a 1, the 4 byte bitmap immediately follows the bitmap control byte. If this bit is a 0, the two bytes following the bitmap control byte are zero. The “y” bits in the bitmap control byte specify directory attributes. These bits are from most significant to least significant. These attributes are not yet implemented in TMEX 3.10.

read–only:	The directory that contains this attribute and all files and sub–directories can be read but not deleted or modified. If it is set on the root level then the entire device is read only.
archive:	The directory that contains this attribute has had some or all of its files modified since the last backup.
system:	The directory that contains this attribute is designated to have system files.
encrypt:	The directory that contains this attribute has all of its files encrypted.

The “z” bit in the bitmap control byte is not used. The least significant bit “p” is an in–progress bit. This bit can be set when doing non–interruptible operations like file optimization. Note that TMEX does not set this attribute because none of its operations are non–interruptible. The bit is cleared after the operations are complete.

The next two bytes contain the starting page address and the number of pages required by the bitmap file. Note that the bitmap or the nameless bitmap file is created during the process of formatting. Continuation pages of the directory don’t need a control field. This space is available to store another file entry.

File entries consist of the 4–byte file name, one–byte file extension, the starting page address where the file begins, and the number of pages the file occupies. This structure is shown below:

FILE ENTRY Figure 3

file name	extension	start page	# pages
ASCII, blank filled	binary	binary	binary
left justified	aeeeeeee		
4 bytes	1 byte	1 byte	1 byte

File names must consist of ASCII characters only, as with DOS. However, if the first byte of a file entry has a value greater than 127, then it represents the first byte of an extended directory entry. The extended directory entry is a 7–byte or multiple 7–byte data packet that applies to the next directory entry. Each 7–byte data packet must have a first byte greater than 127. The additional bytes in an extended directory entry may be used to specify additional file attributes, passwords, file ownership, date/time stamps, and other special–purpose information regarding the file. The extended directory entry is not supported in TMEX 3.10.

The 7 least significant “e” bits of the extension represent the extension number of the file entry. Extensions 0–99 decimal are reserved for normal file entries, 100 for an AddFile that resides on an EPROM iButton device and 101 for a Monetary File that is only created on a DS1962, DS1963, DS2422 and DS2423. The data page of the Monetary File can only be located on a page that has a corresponding counter. If a page with a counter is not available on the device then the Monetary File will not be created. Extension 127 decimal designates a sub–directory. Extensions 102–126 are reserved for future specialty file types. The most significant bit of the file extension “a” is an attribute flag for the entry. If it is set and the entry is a normal file with extensions 0–99 then the file is read–only. If it is set and the entry is a sub–directory with extension 127 then the sub–directory is hidden. File entries denoting sub–directories contain the real “start page” of the sub–directory file. The “# pages” specifier for a sub–directory entry does not contain a valid number. It always contains 0 and need not be updated when the length of a sub–directory file changes. A sub–directory file contains a back reference to the next higher directory file. If there is no higher sub–directory file the entry will be “ROOT”. The start page of ROOT is always 0. The first packet of a sub–directory file has the following structure:

SUB–DIRECTORY START Figure 4

length binary 8...29 1 byte	control data 7 bytes	file entries ASCII & binary 0 to 21 bytes	cont.– pointer binary 1 byte	$\overline{\text{CRC16}}$ binary 2 bytes	not used 21 to 0

directory mark "AA" 1 byte	re– served "00" 1 byte	reference to higher directory level ASCII 4 bytes	start page higher dir. binary 1 byte

Again, the numbers shown above presume a 32–byte page length, but the basic structure applies for greater page lengths. A continuation packet of a root directory or a sub–directory file follows the definition of a data packet. It has the following structure:

(SUB–)DIRECTORY CONTINUATION Figrue 5

length binary 1...29 1 byte	data ASCII or binary (0...4) 0 to 28 bytes	cont.– pointer binary 1 byte	$\overline{\text{CRC16}}$ binary 2 bytes	not used 28 to 0

B. Features

The iButton Extended File Structure is carefully designed to provide high speed and the best performance in a Touch environment. Every memory page can be read, CRC–checked or written without the need to access other pages. If a file is modified, only the affected pages need to be rewritten. This provides a significant speed advantage. Pages of a file need not be contiguous. Files can be extended by redefining continuation pointers. Files can be grouped into nested sub–directories. Attributes defined for a directory apply for all files within it. The iButton file structure also accommodates future types of iButtons with up to 256 pages with a maximum of 256 bytes per page.

C. Properties

The notes below summarize many of the significant properties of the Extended File Structure as specified above. These notes may be helpful in understanding the implications of the Extended File Structure with regard to data storage and management.

- **The page length is derived from the family code.** iButton devices currently available have a 32 byte page length, but the definition of the Extended File Structure also supports greater page lengths.
- **Each page of data in the Extended File Structure is a Universal Data Packet (UDP), starting at a page boundary.** The Universal Data Packet of the Extended File Structure is somewhat limited compared with the general Universal Data Packet. This Universal Data Packet may be shorter but never longer than one page.
- **Each Universal Data Packet starts with a length byte, contains a continuation pointer and ends with a CRC16 check.** The pointer indicates the number of the page where a file is continued. A continuation pointer "0" marks the last packet of a file.
- The **length byte** indicates the number of bytes between length byte and CRC. The **CRC calculation** includes all of the preceding data, including the length byte. The CRC accumulator is initialized by setting it equal to the iButton page number.
- **Each Universal Data Packet may be read and CRC–checked independently,** without reference to data from other packets.
- **All types of files consist of one or several Universal Data Packets.** The Universal Data Packets contain either application data (data packet) or directory information (directory packet).
- **Every byte** of a data packet is transmitted least significant bit first. The length byte is the first to be transmitted.
- The maximum number of application data bytes within a data packet is (page length – 4).
- **There is exactly one main directory file in a memory iButton.** This file is called the root directory and always starts at page 0.
- The maximum number of entries in the first packet of a (sub)directory file is (page length–11)/7. The maximum number of entries in continuation packets is (page length–4)/7.

- **The bitmap serves TMEX as a fast lookup table to find free pages.** The data of the bitmap is treated as a 4–byte binary number. The least significant bit corresponds to page 0. A used page is marked with a “1”, an empty page with a “0”. The least significant byte of the bitmap is stored at the lower address and is transmitted first. The bitmap file must be used with iButtons of capacity greater than 32 pages. It may be used with smaller devices depending on formatting.

- **Reserved extensions** are: 100 for an AddFile, 101 for a Monetary File, 102 – 126 (to be assigned), and 127 is for sub–directories; the extensions 0 – 99 are available for general purpose application files.

- **There may be empty or partially filled data packets.** An empty packet consists of the length byte, pointer and CRC16 bytes and acts as pointer to the next data packet of the same file or (sub–) directory. A partly filled data packet contains less than 29 bytes of data. The unused space must follow the CRC. There is no restriction on the use of empty and partly filled data packets.

II. UNIVERSAL DATA PACKET (UDP)

The Universal Data Packet (UDP) is a structure used to store data on iButtons. It contains one to two length bytes, data and two inverted CRC16 bytes. The structure is:

UNIVERSAL DATA PACKET Figure 6

length binary 0...508 1 or 2 bytes	data ASCII or binary 0 to 508 bytes	$\overline{\text{CRC16}}$ binary (LO/HI) 2 bytes

The UDPs always start on page boundaries but can end anywhere. The length is the number of data bytes not including the length byte(s) and the CRC16 bytes. There is one length byte if the number of data bytes is less then 255. If there are 255 or more data bytes then the first length byte is 255 and the next length byte is 0 to 253. The first and second length bytes added together provide the number of data bytes. The CRC16 is first initialized to the starting page number. This provides a check to verify the page that was intended is being read. The CRC16 is then calculated using the length and data bytes. The CRC16 is then inverted and stored low byte first followed by the high byte. A detailed description of the CRC16 can be found in Application Note 27.

The Extended File Structure implemented in TMEX uses a subset of the Universal Data Packet. It limits the length so that the whole structure will fit in one page. For the current devices with 32 bytes per page the length is limited to 29 data bytes. The last data byte is used as a continuation pointer leaving 28 true data bytes.

The Default Data Structure (DDS) is a previous standard that specifies only one Universal Data Packet starting at page zero. This standard does not provide a directory or file type operations but it has speed advantages for simple applications.