Discover the Experience

ADTF DAT File Format 2.1.1

EB Assist ADTF

ADTF DAT File Format – EB Assist ADTF

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 DE-Erlangen, Germany
+49-9131-7701-0
+49-9131-7701-6333
info.automotive@elektrobit.com

Technical support

EB Assist ADTF Support

Phone: +49-9131-7701-7777

http://automotive.elektrobit.com/support

Table of Contents

1 Description

1.1. General Description

The ADTF DAT File Format (*DAT Format*) enables the synchronous recording of arbitrary many data streams in a file. The file size limit of the underlying file system (e.g. 4 GB on FAT32) does not hold as 64-bit data types are utilized for data management. The *DAT File Format* is optimized for high data rates at a minimal memory requirement and suits for live- as well as batch-processing. In addition to data-streaming the *DAT File Format* also enables efficient data retrieval in large data volumes. Supplementary meta-information can be filed as application specific data blocks in file-extension-packets.

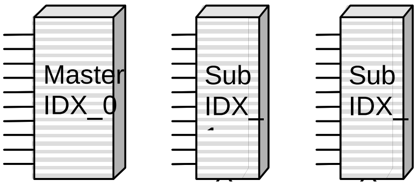
The *DAT File Format* borrows from the „Interchange File Format“ (IFFFormat/ Electronic Arts). Essential concept is the encapsulation of user data into standard container blocks („chunks“) to be able to save arbitrary content uniformly.

Additional information can be found in the *Streaming Library* with which *DAT Files* can be read or written (see StreamingLibrary.pdf and streaminglib.chm).

1.2. Elements

Header	The *DAT File* header contains general information on the saved data, e.g. number of data sets, size of user data, number of index tables.
Chunk	Container block which contains user data. Chunks contain among other things type information, size of the contained user data, stream mapping,
Extension	A file extension block can hold any application specific user data. Extensions are distinguished by their identifier and an optional type-ID.

1.3. Special Elements

	The index block consists of several index tables. The master index table (IDX_0) contains the indices of the user data of all streams. Generally an index entry exists in certain intervals but it can be generated as well with certain flags (e.g. key-frame). Besides the master index there are sub index tables (or stream index IDX_1 to IDX_n). For each stream in the file a sub index table is created. The sub index entries are a collection of references to the master index table which are only assigned to that stream. The index tables are realized as file extensions.

1.4. Context

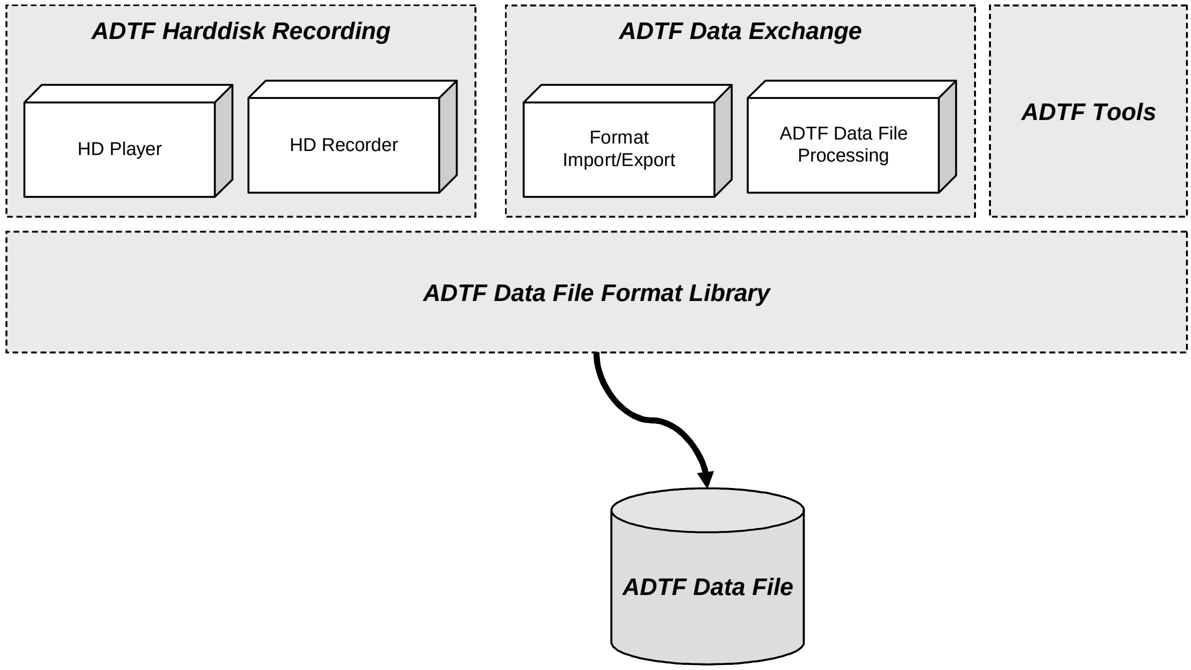


Figure 1.1 Context

1.5. Common Format Layout

The *DAT File Format* has the following schematic structure:

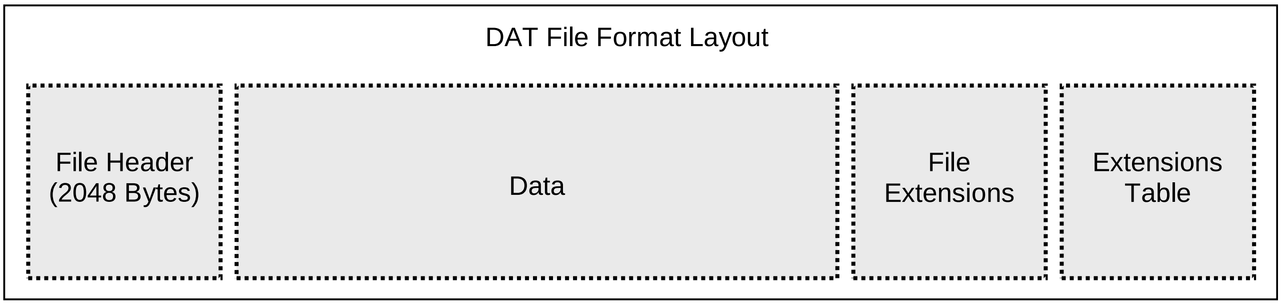


Figure 1.2 DAT File Format Layout

2 File Header

2.1. Structure

```
typedef struct tagFileHeader
{
    tUInt32                         ui32FileId;
    tUInt32                         ui32VersionId;
    tUInt32                         ui32Flags;
    tUInt32                         ui32ExtensionCount;
    tUInt64                         ui64ExtensionOffset;
    tUInt64                         ui64DataOffset;
    tUInt64                         ui64DataSize;
    tUInt64                         ui64ChunkCount;
    tUInt64                         ui64MaxChunkSize;
    tUInt64                         ui64Duration;
    tUInt64                         ui64FileTime;
    tUInt8                          ui8HeaderByteOrder;
    tUInt64                         ui64TimeOffset;
    tUInt8                          ui8PatchNumber;
    tInt8                           _reserved[54];
    tInt8                           strDescription[1912];
} tFileHeader _set_aligment_1_;   // size is 2048 Bytes
```

2.2. Elements

Name	**Description**
nFileId	Identifier for every *DAT File*. Intel-CPU implementations use *((tUInt32*) "IFHD", Motorola use *((tUInt32*) "DHFI“)
nVersionId	Format-version of *DAT Files*. Current version 0x0201.
ui32Flags	Flags
ui32ExtensionCount	Amount of extensions blocks.
ui64ExtensionOffset	File offset to the begin of extension table block (absolute).
ui64DataOffset	File offset to the begin of the data block (absolute).
ui64DataSize	Size of the data block (in byte)
ui64ChunkCount	Amount of chunks
ui64MaxChunkSize	Greatest user data size of chunk
ui64Duration	Time stamp of the last chunk
ui64FileTime	Creation time of file.
ui8HeaderByteOrder	Endianess of management structures (Little Endian oder Big Endian)
ui64TimeOffset	Time offset every time within the file is referred to („timestamp zero“)
ui8PatchNumber	Patch number
strDescription	file description: short- and long description is separated by „\n“

Table 2.1. Elements

3 File-Extensions

3.1. Structure (Header)

```
#define MAX_FILEEXTENSIONIDENTIFIER_LENGTH       384
typedef struct tagFileExtension
{
    tInt8     strIdentifier[MAX_FILEEXTENSIONIDENTIFIER_LENGTH];
    tUInt16    ui16StreamId;
    tUInt8    _reserved1[2];
    tUInt32   ui32UserId;
    tUInt32   ui32TypeId;
    tUInt32   ui32VersionId;
    tUInt64   ui64DataPos;
    tUInt64   ui64DataSize;
    tUInt8    _reserved[96];
} tFileExtension _set_aligment_1_; //512 Bytes
```

3.2. Elements (Header)

Name	Description
strIdentifier	Identifier.
ui32StreamId	Number of stream it belongs to (0 for every / 1> id >=Max Streams)
ui32UserId	Optional User identifier
ui32TypeId	Optional Type identifier
ui32VersionId	Optional Version identifier.
ui64DataPos	File-Offset of Extension data (absolute).
ui64DataSize	Size of Extension data in Bytes.

Table 3.1. DAT File Extension Elements

3.3. Layout

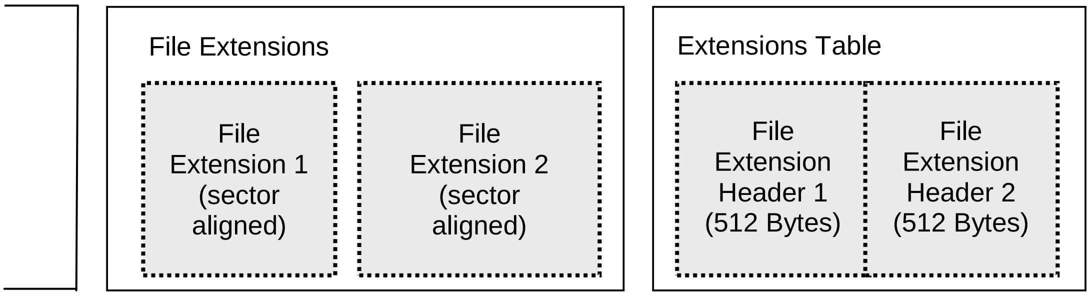

File Extensions
File Extension 1 (sector aligned)
File Extension 2 (sector aligned)
Extensions Table
File Extension Header 1 (512 Bytes)
File Extension Header 2 (512 Bytes)

Figure 3.1 DAT File Extension Layout

3.4. ADTF Clock Extension

3.4.1. Structure

```
typedef struct
{
    tTimeStamp nStreamTime;
    tTimeStamp nClockTime;
} tADTFClockRelation;

tyedef struct
{
    tChar strClockName[256];
} tADTFClockExtensionItem;
```

3.4.2. Elements

Name	Description
nStreamTime	A Timestamp of the active stream clock
nClockTime	The corresponding timestamp of the other clock.

Table 3.2: ADTF Clock Relation Header

Name	Description
strClockName	The Name of the Clock, it can be maximal 256 characters

Table 3.3: ADTF Clock Extension Item

3.4.3. Description

The ADTF Harddisk Recorder will track the time of all available clocks and store it in the following file extensions:

adtf_clock_ext:

This extension contains an array of all recorded clocks stored in **tADTFClockExtensionItem** elements. The first element specifies the ADTF Time Clock and the second element the StreamTime clock that have been used during recording. The remaining elements list all clocks. The amount of elements is defined by dividing the size of the extension by sizeof(**tADTFClockExtensionItem**).

adtf_clock_ext_<clock_name>:

For each of the available clocks there exists an extension that stores an array of **tADTFClockRelation** elements. Once again the amount of elements is defined by dividing the size of the extension by sizeof(**tADTFClockRelation**). Each element specifies a timestamp of StreamTime and a timestamp of the clock, both taken at the same instant. During recording the harddisk recorder will generate those clock relations roughly every 5 seconds.

4 Chunks (User Data)

4.1. Structure

```
typedef struct tagChunkHeader
{
    tUInt64            ui64TimeStamp;
    tUInt32            ui32RefMasterTableIndex;
    tUInt32            ui32OffsetToLast;
    tUInt32            ui32Size;
    tUInt16            ui16StreamId;
    tUInt16            ui16Flags;
    tUInt64            ui64StreamIndex;
} tChunkHeader _set_aligment_1_;  // size is 32 Bytes
```

4.2. Elements

Name	Description
ui64TimeStamp	Timestamp of chunk.
ui32RefMasterTableIndex	Referring to the master index table
ui32OffsetToLast	Relative byte offset to previous chunk (in bytes)
ui32Size	Size of chunk (in bytes)
ui16StreamId	Stream number the chunk belongs to
ui16Flags	KeyData / Flags.
ui64StreamIndex	Number of the Chunks within Stream it belongs to

Table 4.1. Chunk Elements

4.3. Layout

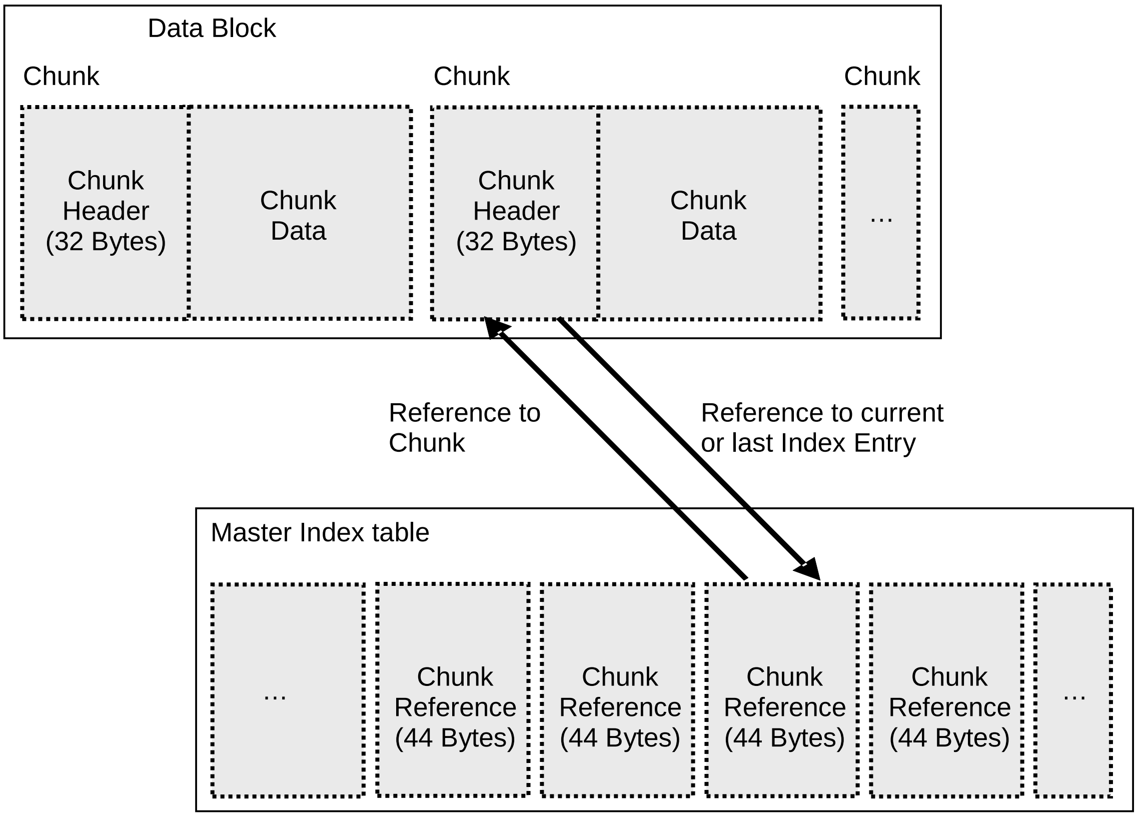

Data Block
Chunk
Chunk
Chunk
Chunk Header (32 Bytes)
Chunk Data
Chunk Header (32 Bytes)
Chunk Data
…
Reference to Chunk
Reference to current or last Index Entry
Master Index table
…
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
…

Figure 4.1 Data Block References

5 Index tables

5.1. Structure

```
Typedef struct tagChunkRef
{
    tUInt64                         ui64TimeStamp;
    tUInt32                         ui32Size;
    tUInt16                         ui16StreamId;
    tUInt16                         ui16Flags;
    tUInt64                         ui64ChunkOffset;
    tUInt64                         ui64ChunkIndex;
    tUInt64                         ui64StreamIndex;
    tUInt32                         ui32RefStreamTableIndex;
} tChunkRef _set_aligment_1_;    // size is 44 Bytes

typedef struct tagStreamRef
{
    tUInt32                         ui32RefMasterTableIndex;
} tStreamRef _set_aligment_1_; // size 4 Bytes

#define MAX_STREAMNAME_LENGTH       228
typedef struct tagStreamInfoHeader
{
    tUInt64                         ui64StreamIndexCount;
    tUInt64                         ui64StreamFirstTime;
    tUInt64                         ui64StreamLastTime;
    tUInt32                         ui32InfoDataSize;
    tInt8                           strStreamName[MAX_STREAMNAME_LENGTH];
} tStreamInfoHeader _set_aligment_1_; // size is 256 Byte
```

5.2. Elements

Name	Description
ui64TimeStamp	Timestamp of chunk it refers to (in microseconds)
ui32Size	Size of chunk it refers to
ui16StreamId	Stream Number of chunk it refers to
ui16Flags	KeyData / Flags of chunk it refers to.
ui64ChunkOffset	File offset position of chunk it refers to (in byte)
ui64ChunkIndex	Number of Chunk
ui64StreamIndex	Number of Chunk within the stream it belongs to
ui32RefStreamTableIndex	Number of stream index table entry this master index entry belongs to

Table 5.1. Chunk Elements

Name	Description
ui32RefMasterTableIndex	Number of master index entry it belongs to

Table 5.2. Stream Reference Elements

Name	Description
i64StreamIndexCount	Amount of chunk belonging to that stream
ui64StreamFirstTime	First time of stream
ui64StreamLastTime	Last Time of Stream
ui32InfoDataSize	Size of following additional information block
strStreamName	Name of the stream

Table 5.3. Stream Info Elements

5.3. Layout

The following figure shows the layout of the index table entries.

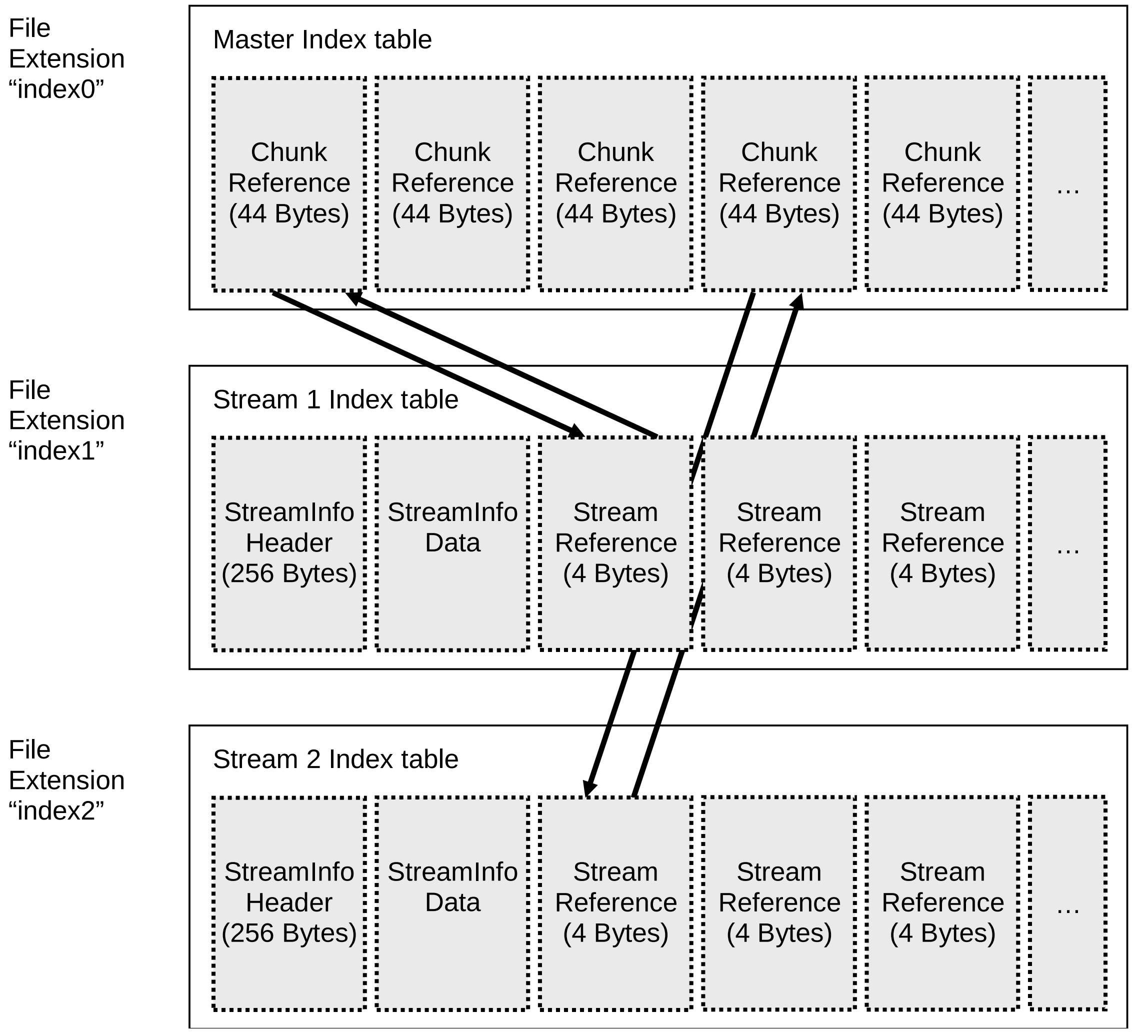

File
Extension
“index0”
Master Index table
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
Chunk Reference (44 Bytes)
…
File
Extension
“index1”
Stream 1 Index table
StreamInfo Header (256 Bytes)
StreamInfo Data
Stream Reference (4 Bytes)
Stream Reference (4 Bytes)
Stream Reference (4 Bytes)
…
File
Extension
“index2”
Stream 2 Index table
StreamInfo Header (256 Bytes)
StreamInfo Data
Stream Reference (4 Bytes)
Stream Reference (4 Bytes)
Stream Reference (4 Bytes)
…

Figure 5.1 Common index tables layout

Table of figures

Table of tables