ASIX

SIGMA & OMEGA

SIGMAP01 - Reading STF File

Application Note

Address:	ASIX s.r.o. Staropramenna 4 150 00 Prague Czech Republic
E-Mail:	sales@asix.net (sales inquiries, ordering) support@asix.net (technical support)
WWW:	www.asix.net
Tel.:	+420-257 312 378
Fax:	+420-257 329 116

1. File composition

The file format is closely linked with each logic analyzer, *SIGMA* or *OMEGA*. File used by *SIGMA* is ***SIGMA Test File*** and file used by *OMEGA* is ***OMEGA Test File***. Both files are using same three letter extension, **.stf**.

2. SIGMA Test File

The file is divided into three parts, each separated by *NULL* character (last byte of first and second part is *0x00*). First and second parts are 8-bit text only, encoding does not matter, third part contains binary data, so it can contain *ASCII* control characters including *NULL*.

2.1 First part: magic

This part is fixed length, containing 16 bytes, 15 characters and *NULL*. It contains string „`Sigma Test File`“ (the white spaces are 0x20) terminated by *NULL* character.
If further (incompatible) file format version would be needed to introduce, this magic will be changed to indicate such incompatibility.

2.2 Second part: settings

This part is variable length and contains only text characters. It is terminated by *NULL* character. Each line is separated by *CR-LF* characters (bytes *0x0D 0x0A*).

The line is in format `Something=Value`. *Something* is identifier and can contain characters `A-Z`, `a-z`, numbers `0-9`, dots „`.`“ and underscore „`_`“, but it cannot start with dot „`.`“. *Value* is value of that identifier. It can represent string, integer number, … *Value* can be any combination of characters containing semicolons, equal signs, ... except quotes, which are not allowed. Special formatting inside *Value* may apply.

The *Values* to be written into the file are withdrawen from all components in the system which are registered to be so, containing plugins. This means that number and types of *Values* stored in the file depends on intalled plugins in the system. Unknown *Values* should be quietly ignored by the reader.

Next table summaries fundamental *Values* which are found in every test file.

Value	Meaning	Notes	Example
`DateTime`	Creation time of file, number of seconds since 1.1.1970		`DateTime=1165156868`
`TestFirstTS`	First valid *TS* in test. Never smaller than 1.	For meaning of *TS*, see third part of file. All *TS* values can range to more than 2^{37}.	`TestFirstTS=8018015`
`TestLengthTS`	Last valid *TS* in test. Length of the test in *TS* is *LestLengthTS*-*TestFirstTS*+1.		`TestLengthTS=14740456`
`TestTriggerTS`	*TS* where is trigger. If zero, trigger did not occur.		`TestTriggerTS=8926421`
`TestCLKTime`	Sample rate in *PU*. If indeterministic, value of `15016` is here.	15015 *PU*=1 ns	`TestCLKTime=300300`
`Sigma.ClockSource`	Several options separated by semicolons		`Fall=0;Rise=0;`...
`ClockScheme`	0 for 50 MHz and less 1 for 100 MHz 2 for 200 MHz 3 for Asynchronous mode 4 for Synchronous mode		`ClockScheme=0`
`Period`	Used only when `ClockScheme=0` Clock period=`Period`*20 ns	Maximum 256	`Period=1`
`Pin`	Used in Asynchronous and Synchronous mode Input for clock source. `Pin`=0 is Input1.	In synchronous mode, only Pin=0 and 8 are supported by *SIGMA*.	`Pin=0`
`Fall`	Used in Asynchronous and Synchronous mode `Fall`=1 clocks from falling edge.	In synchnous mode, `Fall`=1 and `Rise`=1 cannot be set simultaneously.	`Fall=0`
`Rise`	Used in Asynchronous and Synchronous mode		`Rise=0`

	`Rise`=1 clocks from rising edge.		
`Sigma.SigmaInputs`	Input names separated by semicolons	Unallowed characters are escaped using % sign. See note below table.	`SCLK;MISO;MOSI;;;;;...`
`Sigma.Trigger`	Several options separated by semicolons	Triggering options are documented in separate file, *SIGMAP04 – Trigger Options*.	
`Traces.Traces`	Each trace separated by semicolon		`Trace1;Trace2;...`
	One trace Each subopntion separated by colon		`Caption=SCLK:Radix=16:Digits=2:...`
`Caption`	Caption visible in GUI		`Caption=SCLK`
`Radix`	Number format when configured as `Bus`. `Separator` contains number of grouped digits. (typically 2 for `Radix`=16, 3 for `Radix`=10 and 4 for `Radix`=2)		`Radix=16`
`Digits`			`Digits=2`
`Separator`			`Separator=0`
`Type`	*Trace* type `Input` consists of one *Input*. *Trace* type `Bus` consists of many *Inputs*. *Trace* type `Plugin` is trace imported from plugin. Older versions of *SIGMA* software used also types `Analog` and `Digital`. These types are identical to `Input` type.		`Type=Input`
`Input0`	*InputIDs* which generates *Trace*. Bus traces are generated by more than one *Input*, other types are generated by just one *Input*.	For meaning of *InputIDs*, see notes below.	`Input0=0`
`Input1` ...			`Input1=...`

`DataClass`	Data storage method for OMEGA Test File.
	`TOmegaChainChunkedData` is the legacy format (see 3.4.2).
	`TOmegaStreamedData` is streamable format (see 3.4.1).
			`DataClass=`
			`TOmegaStreamedData`

TS stands for *TimeStamp*. *TimeStamp* increments by one each time *SIGMA* can produce one sample, which is always 16 bits.

PU stands for *PicoUnit*. One ns is 15015 *PicoUnits*.

Test length in seconds is *TestCLKTime**(*TestLengthTS*-*TestFirstTS*+1).

In synchronous mode, *TestCLKTime* is not known, value of *TestCLKTime* is 15016, test length in seconds is unknown and formula above has no meaning.

In asynchronous mode, *TestCLKTime* is valid, equal to 300300 (*SIGMA* is sampling at 50 MHz).

In 100 MHz and 200 MHz sampling mode *TS* increments every 20 ns, *TestCLKTime* is 300300. *SIGMA* makes „one sample“ in 20 ns by combining 4 or 8 inputs into 16 bits.

Maximum clock rate is *TestCLKTime* = 76876800 *PU*, maximum *TS* is ~2^{37}, so maximum *TestCLKTime***TS* is about $1.1*10^{19}$, ~$2^{63.2}$. Limit 2^{63} *PU* is more than 170 hours.

Where applicable, unallowed characters are escaped by % sign followed by two hexadecimal digits.

InputID is identifier of *Input*, either of *SIGMA* or plugin. Plugin Traces have separate *ID* space, Plugin Input 0x10000 and Plugin Trace 0x10000 may not be the same.

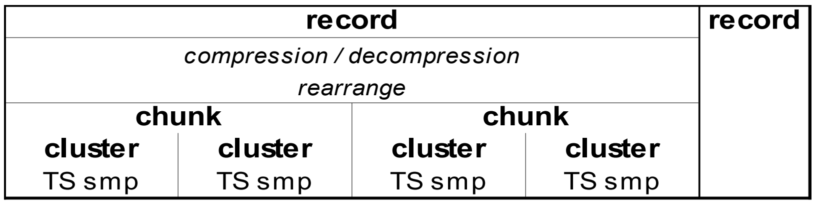
InputID	**Meaning**
0x0000 - 0x00FF	*SIGMA* analyzer Inputs
0xFFF8	*SIGMA* analyzer's virtual rising edge sampling clock
0xFFF9	*SIGMA* analyzer's virtual falling edge sampling clock
0x00010000 - 0x7FFFFFFF	Plugin Input. Upper word is identifier of plugin (hash generated from plugin's file name), lower word is plugin's own identifier.

2.3 Third part: binary data

Binary data are organized in series of *records*. Each *record* holds several *chunks*. One *chunk* holds 64 *clusters*. One *cluster* contains info on one *TimeStamp* and seven consecutive *samples*. To improve compression techniques used in the file, *TimeStamps* and *samples* from all *clusters* in one *chunk* are rearranged, so *TimeStamps* are consecutive and than *samples* are consecutive.

Multibyte integer values are stored in **little endian** format.

In *SIGMA* hardware (but not in STF file!) *cluster* size is 16 bytes and *chunk* size is 1024 bytes.



2.3.1 Record structure

Meaning	Offset [bytes]	Length [bytes]	Notes
payload length	0	4	little endian; in bytes; max 1M
CRC32 of payload	4	4	little endian, same alg. as ethernet
data payload	8	payload length	compressed LZO1X
next record	8+payload length		

Last *record* of file is indicated by *payload length* = `0xFFFFFFFF` and *CRC32* = `0x00000000` (correct checksum for zero length data). Every *STF* file ends with bytes `0xFF 0xFF 0xFF 0xFF 0x00 0x00 0x00 0x00`. This does not mean that this sequence cannot be found in the file itself.

For purpose of error checking, *payload length* must not be greater than 1 Mbyte (1048576). *SIGMA* software itself do not create *record* larger than 128 kbyte.

Structure names used in file are in next table:

	Structure name	**Size after decompression**	**Notes**
largest	*record data payload*	*num_chunks*·1440	*num_chunks* x *chunk* = *record*
↓	*chunk*	1440	64x *cluster* = *chunk*
↓	*cluster*	22	1x *TimeStamp* & 1x *samples* = *cluster*
↓	*samples*	14	7x *sample* = *samples*
smallest	*TimeStamp*	8	
	sample	2	

Data payload is compressed using *miniLZO* library, algorithm *LZO1X*. The library can be downloaded at address http://www.oberhumer.com/opensource/lzo/. Data payload of one *record* is compressed/decompressed at once.

When decompressed, one cluster occupies 1440 bytes. Than, `number_of_chunks_in_record` = `record_length` / 1440.

2.3.2 Record structure data payload

Decompressed data payload holds rearranged *chunk* and *cluster* data as shown in next table, `number_of_chunks_in_record` as *n*.

Meaning	Offset [bytes]	Length [bytes]	Notes
Chunk #1 info	0	32	*chunk infos* total size = *chunk_count*·32
...		...	
Chunk #n info	$(n\text{-}1)\cdot 32$	32	
Chunk #1 *Cluster* #1 *TimeStamp*	$n\cdot 32$	8	*chunk.cluster TimeStamps* total size = *chunk_count*·64·8
...		...	
Chunk #1 *Cluster* #64 *TimeStamp*	$n\cdot 32 + 63\cdot 8$	8	
Chunk #2 *Cluster* #1 *TimeStamp*	$n\cdot 32 + 1\cdot 64\cdot 8$	8	
...		...	
Chunk #n *Cluster* #64 *TimeStamp*	$n\cdot 32 + (n\text{-}1)\cdot 64\cdot 8 + 63\cdot 8$	8	
Chunk #1 *Cluster* #1 *Samples*	$n\cdot 32 + n\cdot 64\cdot 8$	14	*chunk.cluster Samples* total size = *chunk_count*·64·14
...			
Chunk #n *Cluster* #64 *Samples*	$n\cdot 32 + n\cdot 64\cdot 8 + (n\text{-}1)\cdot 64\cdot 14 + 63\cdot 14$	14	
structure length	$n\cdot 1440$		

2.3.3 Chunk info structure

Chunk info holds several useful information about *clusters* inside.

Meaning	Offset [bytes]	Length [bytes]	Notes
Min	0	0	minimum vector of all *samples* in *chunk*
Max	2	2	maximum vector of all *samples* in *chunk*
Chunk ID	4	4	reserved, can be zero, used only in communication with *SIGMA* hardware
FirstTS	8	8	*TimeStamp* of first *cluster* in *chunk*
LastTS	16	8	*TimeStamp* of last *cluster* in *chunk*
ChunkLength	24	8	*next_chunk_FirstTS – current_chunk_FirstTS*

2.3.4 Cluster structure

Cluster is *TimeStamp*, a 64 bit integer value of *TimeStamp* of first sample and 7 *samples*, each 16 bits. One *cluster* is 22 bytes long.

Each sample is 16 bits. When sampling 8 inputs at 100 MHz, one sample is produced per 20 ns by sampling 8 inputs two times. When sampling 4 inputs at 200 MHz, one sample is produces per 20 ns by sampling 4 inputs four times.
TimeStamp increases by one per one 16 bit sample. Thus, when sampling at 100 MHz and 200 MHz, TimeStamp increases by one per 20 ns.

2.3.5 Sample encoding

For 16 (or 15) inputs modes, every pin is mapped to one bit of the sample (for 15 inputs with one pin used as clock for synchronous clock mode, the corresponding bit is undefined).

For the 4 inputs / 200 MHz mode and for the 8 inputs / 100 MHz mode are pins mapped into 16 bit sample according to table.

Sample bit number	**8 inputs / 100 MHz mode**	**4 inputs / 200 MHz mode**
0	Input 1 in time +0 ns	Input 1 in time +0 ns
1	Input 1 in time +10 ns	Input 1 in time +5 ns
2	Input 2 in time +0 ns	Input 1 in time +10 ns
3	Input 2 in time +10 ns	Input 1 in time +15 ns
4	Input 3 in time +0 ns	Input 2 in time +0 ns
5	Input 3 in time +10 ns	Input 2 in time +5 ns
6	Input 4 in time +0 ns	Input 2 in time +10 ns
7	Input 4 in time +10 ns	Input 2 in time +15 ns
8	Input 5 in time +0 ns	Input 3 in time +0 ns
9	Input 5 in time +10 ns	Input 3 in time +5 ns
10	Input 6 in time +0 ns	Input 3 in time +10 ns
11	Input 6 in time +10 ns	Input 3 in time +15 ns
12	Input 7 in time +0 ns	Input 4 in time +0 ns
13	Input 7 in time +10 ns	Input 4 in time +5 ns
14	Input 8 in time +0 ns	Input 4 in time +10 ns
15	Input 8 in time +10 ns	Input 4 in time +15 ns

3. OMEGA Test File

Because of several reasons including:

- Allow users more simple method to read test file
- Computers are now much faster to allow better comression than *LZO*
- OMEGA stores compared to SIGMA huge amount of data, better, yet still fast, compression was required; therefore instead of *LZO* algorithm, *DEFLATE* algorithm is used.

3.1 File format

OMEGA Test File can be read just like it was normal .zip file. Just rename it to .zip extension (or pass it directly without renaming) and decompress it by some unzip utility. A .zip file can have arbitrary number of bytes on its beginning and up to 64kB bytes on its end. This feature is used by SIGMA/OMEGA software to detect the file as OMEGA Test File.

3.2 Bytes before the first zip file header

There are exactly 16 bytes of data before first „PK“ zip header. Very similar to SIGMA Test File, these bytes are zero terminated ASCII „`Omega Test File`“.

3.3 Bytes after the zip file

There are exactly 48 bytes after the end of the zip file. First 32 bytes are fingerprint used for licensing purposes. Last 16 bytes are again zero terminated ASCII „`OMEGA Test File`“.

3.4 Files inside the zip file

There are several files inside the zip file. Test should always include at least one `Settings` file (without any extension). Other (data) files depends on actual file format and measurement method.

3.4.1 Streamable data file format

Streamable data file format is used for *Real-Time* mode, but in the future it can be used for saving acquired data from other modes.

3.4.1.1 Settings file (Optional)

The file `Settings` contains settings in text form in exactly the same format as *SIGMA Test File*. The file is not terminated by null character, but simply with newline (not required during file loading). If the file is missing in the archive, it is treated like it would be empty[1].

3.4.1.2 Omega.Data file (Required)

File Omega.Data contains all acquired data in sequence. It can be generated and treated like a stream, even during continuous *Real-Time* acquisition.

The file is sequence of samples, each 6 bytes long.

Length (bytes)	Offset (bytes)	Format	Meaning	
2	0	uint16	Time from previous sample, in timestamp (=10 ns) units.	
4	2	uint32	Lower 16 bits	First sample, at time +0 ns.
			Upper 16 bits	Second sample, at time +5 ns.

All values are stored as little endian. Time from previous sample of the first sample in the stream should be ignored. Except for the first sample, it should not be zero. For consecutive samples, it should be one.

3.4.1.3 Omega.Triggers (Optional)

File contains list of int64 (little endian) values of positions of all triggers in the acquisition. If the file is zero length or is missing, there is no trigger in the acquisition.

3.4.1.4 Omega.Overflows (Optional)

File contains list of records of two int64 (little endian) values. Each record contains first and last timestamp of regions where was overflow of FIFO memory of the logic analyzer during acquisition.

If the file is missing or is zero length, there is no overflow event in the acquisition.

3.4.2 Legacy file format

Legacy file format is used for all modes except *Real-Time* mode, but this may change in the future. There are three files for each *OMEGA* in daisy chain. If only one *OMEGA* was used to acquire the test, only one set of files is included. The files are named `Omega`*n*`.Data`, `Omega`*n*`.Index` and `Omega`*n*`.Offsets`, numbered from zero.

3.4.2.1 Settings file

The file `Settings` contains settings in text form in exactly the same format as *SIGMA Test File*. The file is not terminated by null character, but simply with newline (not required during file loading).

1 The software will issue a warning.

3.4.2.2 .Index file

The `.Index` file contains index of chunks. Each chunk represents one row in *OMEGA* memory ruding test. Maximum number of chunks is 2^{15}. The `.Index` file contains five 32 bit integers per chunk, therefore its byte size must be always divisible by 20. The integers are stored in little endian order.

First integer	Lower 16 bits contains „min“ vector		
	Upper 16 bits contains „max“ vector		
Second integer	Chunk length in timestamps		Difference between timestamps of first sample in this chunk and the next chunk
Third integer	Lower 32 bits	Timestamp of first sample in this chunk	
Fourth integer	Upper 32 bits		
Fifth integer	Data length (number of nodes)		

Sum of all data length in all chunks must match length of `.Data` and `.Offsets` files (in 32 bit integers).

Each node contains one timestamp and two 16bit or four 8bit samples depending on measurement mode (200 MHz / 400 MHz).

3.4.2.3 .Data and .Offsets files

Each file contains one 32 bit integer per node for all chunks. Nodes of all chunks are stored in a row.

One node contains 32 bits, that is two or four samples (depending on 8 or 16 inputs). Offsets contains difference from previous node. First value in `.Offsets` file of each chunk does not contain difference to anywhere (timestamp of this node is exactly the timestamp of first node stored in `.Index` file) and must be zero.

The `.Data` contains sample value in this node. Lower bits contains previous samples, higher bits contains consecutive samples.

3.5 Requirements for OMEGA Test File

Logic analyzer software will read any zip file with any file extension (if you will force it to do so). The detection is based on presence of „`Settings`“ file (case insenzitive). Strings „`OMEGA Test File`“ before and after the zip file structure are not required, but without them software will show a warning message.

4. Document revision history

Version	When	What
1.00	23.7.2008	First official release
2.00	13.10.2011	Added a new format for OMEGA
2.01	15.12.2012	Typo corrections
2.02	19.3.2013	Corrections in paragraph 2.3.3
2.03	29.3.2013	Clarification of SIGMA sample encoding
2.04	15.12.2016	Add a new streamable format for OMEGA