AMALTHEA

(ITEA 2 – 09013)

Model Based Open Source Development Environment for Automotive Multi-Core Systems

Specification of the Hardware Trace Format

Version: 1.0

Contact	Name	Partner
Technical contact	Lukas Mewes lukas.mewes@bhtc.com	Behr-Hella Thermocontrol GmbH
Work package contact	Erik Kamsties erik.kamsties@fh-dortmund.de	Fachhochschule Dortmund
Project contact	Harald Mackamul harald.mackamul@de.bosch.com	Robert Bosch GmbH

ITEA 2 INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

[1] This specification was created in the context of the AMALTHEA ITEA2 project (ID 01IS11020E) and is released on the Eclipse Auto IWG.

History

Rev.	Content	Resp. Partner	Description	Date
1.0	Released Version	BHTC (Lukas Mewes)	Released on Eclipse Auto IWG	2014-04-25
0.9	Draft Version	BHTC (Lukas Mewes)	Initial specification	2014-02-15

Contents

1. Introduction

The HTF (Hardware Trace Format) is a result of the ITEA2 AMALTHEA[1] project. It is a compact trace format for the tracing of events on embedded systems, including multicore systems. Its binary representation shortens the execution time of the tracing instructions and improves the storage effeciency of the tracing data on the target system, compared to other trace formats. On the host system, the target traces can be saved in an ASCII file **.htf* without the need for any extra conversion. Still, it is compatible to other trace formats (e.g. BTF[2], OT1[3]), so a conversion into these formats is possible.

Structure of a HTF-File
HTF files are divided into three sections. The first part is the header where meta data can be found. Exemplary parameters are the creation date, a description and the target system name. In the second section, reference tables are defined. These tables represent the association between the recorded events and the AMALTHEA software components. Every table entry consists of a specific hex value and a textual description. Subsequently, reference tables allow the interpretation of the binary trace data. The last section of the HTF file contains the recorded trace datasets. Each HTF trace dataset consists of a timestamp, an entity, and an event. When tracing single core systems, this part has one data section, whereas the trace of multicore systems can produce additional data sections.

[1]http://www.amalthea-project.org
[2]http://wiki.eclipse.org/Auto_IWG#Publications
[3]https://gliwa.com/ot1

2. Header

The header provides the global information of a tracing session. Each entry is headed by a hash symbol for a simplified parsing of the file. The following list contains all defined parameters of the header, each with an explanation and a brief illustrative example.

2.1. Format

The first parameter is the *Format*, which has to have the value "HTF" in a valid HTF file.

Example: `#Format HTF`

2.2. Version

The parameter *Version* shows the version number of the used HTF specification.

Example: `#Version 1.0`

2.3. URL

The parameter *URL* references a website where the HTF specification document can be found.

Example: `#URL http://wiki.eclipse.org/Auto_IWG#Publications`

2.4. Project

The *Project* parameter provides information about the project context of the tracing session. For instance, the project, the respective department, or the institution can be mentioned.

Example: `#Project ITEA2 AMALTHEA`

2.5. TargetSystem

Information about the target system, on which the tracing was done, is saved in the parameter *TargetSystem*.

Example: `#TargetSystem Freescale MPC5668G evaluation board`

2.6. Description

In the parameter *Description* a short textual description of the software can be given. Conceivably, the application name with some special features of the program can be put in.

Example: `#Description Test-Application with 7 tasks and 2 interrupts`

2.7. NumberOfCores

The *NumberOfCores* parameter saves how many of cores of the target system are traced.

Example: `#NumberOfCores 2`

2.8. CreationDate

The parameter *CreationDate* refers to the point in time when the tracing session was started. The value contains the date as well as the time and requires the following format:
Year-Month-Day Hour:Minute:Second (yyyy-mm-dd hh:mm:ss)

Example: `#CreationDate 2014-04-04 13:15:25`

2.9. TimeScale

The *TimeScale* parameter specifies the unit of the timestamps in the recorded trace. Applicable time units in an HTF are:

- picosecond (ps)
- nanosecond (ns)
- microsecond (us)
- millisecond (ms)
- second (s)

Example: `#TimeScale ns`

2.10. TimeScaleNumerator and TimeScaleDenominator

Usually, the generated timestamps of an embedded system are not scaled in the base unit of time (ps, ns, us, ms, s). The *TimeScaleNumerator* and *TimeScaleDenominator* parameters allow to convert the timestamps into the above-mentioned time units. Via the *TimeScaleNumerator* a multiplier is defined, while the *TimeScaleDenominator* provides a divisor. This way, the generated timestamps can be converted into the specified base unit. The timestamps are multiplied with the *TimeScaleNumerator* and then divided by the *TimeScaleDenominator*. The result is a modified timestamp with the required resolution.

Example:

```
#TimeScaleNumerator 4
#TimeScaleDenominator 1
```

2.11. TimestampLength

The parameter *TimestampLength* defines the byte size of the timestamps. The value represents the number of used bytes. The following table provides the maximum values for systems with 8-bit, 16-bit, 32-bit, and 64-bit architectures.

Number of bytes	Maximum in hexadecimal	Maximum in decimal
1	FF	256
2	FF FF	65.536
4	FF FF FF FF	4.294.967.296
8	FF FF FF FF FF FF FF FF	18.446.744.073.709.600.000

Table 2.1.: HTF maximum values of timestamps

Example: `#TimestampLength 4`

2.12. EntityLength

The parameter *EntityLength* shows the number of bytes which an entity ID consists of.

Example: `#EntityLength 2`

2.13. EventLength

The parameter *EventLength* shows the number of bytes which an event ID consists of.

Example: `#EventLength 1`

3. Reference tables

In the reference tables a hexadecimal value gets linked with a string. Hereby, the recorded trace can be transformed into a legible format. Again, the hash sign is used as a starting symbol, while a blank character separates the hexadecimal values from the referenced string.

3.1. TypeTable

System events can be generated or invoked by diverse entities. For example, operating system behaviors, the events of tasks, and interrupt service routines can be recorded. The *TypeTable* assigns IDs to the entity types. An exemplary *TypeTable* is shown in the extract of an HTF file below.

```
#TypeTable
#-00 Task
#-01 ISR
#-02 Runnable
#-03 CodeBlock
#-04 Signal
#-05 Semaphore
```

3.2. EventTables

In the following reference tables the events are assigned to the respective entity types.

```
#TaskEventTable
#-00 activate
#-01 start
#-02 resume
#-03 preempt
#-04 terminate
#-05 wait
#-06 release
#-07 poll
#-08 run_polling
#-09 park
#-0A poll_parking
#-0B release_parking
```

```
#ISREventTable
#-00 start
#-01 resume
#-02 preempt
#-03 terminate
```

```
#RunnableEventTable
#-00 start
#-01 suspend
#-02 resume
#-03 terminate
```

```
#CodeBlockEventTable
#-00 start
#-01 stop
```

```
#SignalEventTable
#-00 read
#-01 write
```

```
#SemaphoreEventTable
#-00 lock
#-01 unlock
```

3.3. EntityTable and EntityTypeTable

In the *EntityTable* each software entity gets an ID, while in the *EntityTypeTable* a type is assigned to each entity. The following extract of an HTF file shows an example for both reference tables.

```
#EntityTable                      #EntityTypeTable
#-0000 TaskZ6One                  #-0000 00
#-0001 TaskZ6Two                  #-0001 00
#-0080 TaskZ0One                  #-0080 00
#-0081 TaskZ0Two                  #-0081 00
#-0082 TaskZ0Three                #-0082 00
#-0083 TaskZ0Four                 #-0083 00
#-00EE Z6_interrupt               #-00EE 01
#-00FF Z0_interrupt               #-00FF 01
```

4. TraceData

In the last section of the HTF file, the binary tracing data are filed. The beginning of this section is marked with the keyword *TraceData*. After that keyword, the hash symbol is only used to mark the beginning of a tracing session for each core. Every line contains one dataset and an optional comment. Each dataset is a hexadecimal number. This number comprises three columns which contain the information concerning the timestamp, the entity ID, and the event ID. The actual width of each column is defined by the values of *TimestampLength*, *EntityLength*, and *EventLength*. Every comment starts with a "//". A example trace of a dual-core system is shown below.

```
#TraceData     //| 4 Byte Timestamp  | 2 Byte Entity ID | 1 Byte Event ID |
#-01           // Core 1
00000000000000 //Timestamp: 00000000  Entity: 0000->T1  Event: 00->activate
00000064000001 //Timestamp: 00000064  Entity: 0000->T1  Event: 01->start
00002710000100 //Timestamp: 00002710  Entity: 0001->T2  Event: 00->activate
00002774000003 //Timestamp: 00002774  Entity: 0000->T1  Event: 03->preempt
00002780000101 //Timestamp: 00002780  Entity: 0001->T2  Event: 01->start
0000417E000104 //Timestamp: 0000417E  Entity: 0001->T2  Event: 04->terminate
000041E2000002 //Timestamp: 000041E2  Entity: 0000->T1  Event: 02->resume
00004EE7000004 //Timestamp: 00004EE7  Entity: 0000->T1  Event: 04->terminate

#-02           //Core 2
00000000000200 //Timestamp: 00000000  Entity: 0002->T3  Event: 00->activate
00000055000202 //Timestamp: 00000055  Entity: 0002->T3  Event: 02->start
00450145000205 //Timestamp: 00450145  Entity: 0002->T3  Event: 05->terminate
```

A. Appendix

A.1. Example HTF-File

```
#Format HFT
#Version 1.0
#URL http://wiki.eclipse.org/Auto_IWG#Publications
#Project ITEA2 AMALTHEA
#TargetSystem Freescale MPC5668G
#Description HVAC Demonstrator
#NumberOfCores 2
#CreationDate 2014-03-25 10:21:33
#Timescale ns
#TimeScaleNumerator 10
#TimeScaleDenominator 1
#TimeStampLength 4
#EntityLength 2
#EventLength 1

#TypeTable
#-00 Task
#-01 ISR
#-02 Runnable
#-03 CodeBlock
#-04 Signal
#-05 Semaphore

#TaskEventTable
#-00 activate
#-01 start
#-02 resume
#-03 preempt
#-04 terminate
#-05 wait
#-06 release
#-07 poll
#-08 run_polling
#-09 park
#-0A poll_parking
```

```
#-0B release_parking

#ISREventTable
#-00 start
#-01 resume
#-02 preempt
#-03 terminate

#RunnableEventTable
#-00 start
#-01 suspend
#-02 resume
#-03 terminate

#CodeBlockEventTable
#-00 start
#-01 stop

#SignalEventTable
#-00 read
#-01 write

#SemaphoreEventTable
#-00 lock
#-01 unlock

#EntityTable
#-0001 TRACEID_TASK_CP0
#-0002 TRACEID_TASK_CP1
#-0003 TRACEID_TASK_PP0
#-0004 TRACEID_TASK_PP1
#-0005 TRACEID_TASK_PP2
#-0010 TRACEID_Z6_20MS_ISR
#-0011 TRACEID_Z0_20MS_ISR
#-0012 TRACEID_Z0_100MS_ISR
#-00F0 TRACEID_hmi_sendToUI
#-00F1 TRACEID_hmi_receiveFromUI
#-00F2 TRACEID_coordinator_runCycle
#-00F3 TRACEID_drvTempAdapter_runCycle
#-00F4 TRACEID_drvTempFlapCtrl_runCycle
#-00F5 TRACEID_passTempAdapter_runCycle
#-00F6 TRACEID_passTempFlapCtrl_runCycle
#-00F7 TRACEID_blowerCtrl_runCycle
#-00F8 TRACEID_hvacBlower_setPower
```

```
#-00F9 TRACEID_hvacItos_readTemp
#-00FA TRACEID_hvacFlaps_setFlaps

#EntityTypeTable
#-0001 00
#-0002 00
#-0003 00
#-0004 00
#-0005 00
#-0010 01
#-0011 01
#-0012 01
#-00F0 02
#-00F1 02
#-00F2 02
#-00F3 02
#-00F4 02
#-00F5 02
#-00F6 02
#-00F7 02
#-00F8 02
#-00F9 02
#-00FA 02

#Tracedata
#-00
001E701E001000
001E7192000100
001E7304001003
001E7460000101
001E75DE00F100
001F6A8700F103
001F6C1300F200
001F7B3700F203
001F7CC300FA00
001FA23800FA03
001FA3C4000104
003CF4A0001000
003CF614000100
003CF786001003
003CF8E2000101
003CFA6000F100
003D37F500F103
003D396E00F200
```

```
003D470400F203
003D488100FA00

#-01
001E72B4001100
001E77C8000300
001E7CD4001103
001E8202000301
001E866600F300
001E96F400F303
001E9B5800F500
001EAB1800F503
001EAFA200F000
001EC3C600F003
001EC822000304
003CF736001100
003CFC4A000300
003D0156001103
003D0696000301
003D0AFA00F300
003D1AC400F303
003D1F2A00F500
003D2E4400F503
003D32D200F000
```