T1 Build Step Manual

Configuration of T1-TARGET-SW

User Guide – Version 1.16

www.gliwa.com

GLIWA GmbH embedded systems
Pollingerstr. 1
82362 Weilheim i.OB.
GERMANY

fon +49 - 881 - 13 85 22 - 0
fax +49 - 881 - 13 85 22 - 99
info@gliwa.com
www.gliwa.com

Document ID: T1target-30-20-20-10

This document explains the ***T1*** build step and gives a detailed description of the configuration of the T1-TARGET-SW and the corresponding configuration parameters. This is helpful during the process of ***T1*** integration and later on during ***T1*** usage. This release is up-to-date with respect to T1-TARGET-SW V3.3.0.1.

Release	*Date*	*Author*	*Comment*
1.0	2014-03-21	Peter Gliwa	First released version
1.1	2014-11-10	Toni Baier	Changed doc location and added correct DocID
1.2	2015-04-08	Andreas Knickenberg	Updated descriptions of -t1pSymbolIncludeFile, -targetExcludeFromTraceByName, -targetExcludeFromTraceById, -licInfo2, -initFeatureMask, and -cpuLoadCallback. Added -allocateCat1Isr.
1.3	2015-04-09	Andreas Knickenberg	Updated -t1pSymbolIncludeFile.
1.4	2015-05-06	Andreas Knickenberg	Renamed -allocateCat1IsrId to -allocateCat1IsrIdPrio and updated description.
1.5	2015-10-21	Andreas Knickenberg	Added remote core T1.cont support. Removed reference to data libs.
1.6	2015-11-23	Andreas Knickenberg	Added parameter -pSyncTimer.
1.7	2016-05-19	Andreas Knickenberg	Added external build ID. Minor update in the description of -nofFlexStopwatches. Extended description of -osBackgroundTaskName.
1.8	2016-09-06	Alexandre Baufumé, Nicholas Merriam, Andreas Knickenberg	Added Ethernet support. Added -configC, -maxPreemptionDepth, -staticRunnableID, -t1pAnnotationsFile, -canHwName and -endiannessAgnosticComSetup. Updated -usingMulticoreLibs. Sorted parameters alphabetically. Removed 'obsolete' marking of -bigEndian.

1.9	2016-11-08	Andreas Knickenberg	Updated -configC to -configGenC.
1.10	2017-05-29	Alexandre Baufumé, Andreas Knickenberg	Added -numberOfFocusMeasurements. Extended -symbolFile description. Added CAN FD support.
1.11	2017-06-09	Nicholas Merriam, Alexandre Baufumé	-pTimer becomes optional after T1_TraceEvent rework. Add Ms suffix to timeouts. Removed -canFDMaxTxBlocksize, added -canFDRxDataSize and -canFDTxDataSize.
1.12	2017-07-10	Nicholas Merriam, Andreas Knickenberg	Add information that sample point should be configured for CAN FD. Add -commsCoreOffset.
1.13	2018-03-09	Alexandre Baufumé, Nicholas Merriam, Andreas Knickenberg, Markus Kreutzer, Jialin Li	Add -targetType. Replace -32BitCpuWithSmallerTimer with -traceTimerBitLength. Update 'Introduction'. Update BID description with T1_configGen.c. Improve description of -flexAnalysisCapacity. Replace Tx/RxDataSize with MaxTx/RxDataSize. Add -diagOpenSessionType, -diagCustomSessionId and -diagTesterPresentPeriod. Update description of -traceTimerIsSyncTimer. Update description of -idHeaderInclude and -runnableHeaderInclude.
1.14	2018-05-14	Christian Herget, Nicholas Merriam, Andreas Knickenberg	Rename to -ethTxCycleMs and -ethTxCycleMs. Add -noRefT0p. Update -traceTimerDownCounting and add -syncTimerDownCounting.

1.15	2020-06-17	Alexander Stassis, Christian Herget, Nicholas Merriam, Andreas Knickenberg, Alexandre Baufumé	Created an Appendix for the T1-TARGET-SW revision history and moved existing documentation regarding earlier versions into it. Document relaxed requirements for sync timer. Moved -cpuLoadCallback to Appendix. Updated -analysisCapacity, -cpuLoadThreshold, -ethEcuIP, -ethEcuPort, -ethPcIP, -ethPcPort, -ethUseUdp, -numberOfConstraints, -numberOfDelays, -numberOfUserStpws, -pSyncTimer, -pTimer, -rawTickDurationNs, -rawTimerBitLength, -sid, -targetType, -traceTimeStampBitLength and -usingMulticoreLibs. Updated -t1pSymbolIncludeFile and -t1pAnnotationsFile. Added -externalTraceBuffer, -configBuffC, -allocateAF, -allocateUE, -keyValueMapUE, -allocateUserSW, -ethEcuDNSHostName and -ethUseIpv6.
1.16	2020-08-31	Nicholas Merriam, Christian Herget, Andreas Knickenberg	Updated -usingMultiCoreLibs and -traceTimerIsSyncTimer . Added -cortexMFPBversion, -allocateDL, -allocateUDE and -checkIntegration.

Table 1: Document History

Contents

1. Introduction **11**

2. Build ID Mechanism **12**
2.1. External Build ID 13

3. *T1* Perl script **14**
3.1. Parameter -Cfg 16
3.2. Parameter -I 16
3.3. Parameter -OsCfg 16
3.4. Parameter -OsPm 17
3.5. Parameter -UserCfg 17

4. *T1* User Configuration File **17**
4.1. Parameter -additionalCat1IsrIdOffset 18
4.2. Parameter -allocateAF 18
4.3. Parameter -allocateDL 19
4.4. Parameter -allocateCat1IsrIdPrio 21
4.5. Parameter -allocateUDE 22
4.6. Parameter -allocateUE 24
4.7. Parameter -allocateUserSW 25
4.8. Parameter -analysisCapacity 28
4.9. Parameter -bidHeader 28
4.10. Parameter -configBuffC 29
4.11. Parameter -configGenC 29
4.12. Parameter -configHeader 30
4.13. Parameter -configHeaderInclude 30
4.14. Parameter -copyIncludeHeader 31
4.15. Parameter -copySymbolFile 31
4.16. Parameter -cpuLoadThreshold 32
4.17. Parameter -enableDoxygenComments 33
4.18. Parameter -externalTraceBuffer 33
4.19. Parameter -featureMaskChanged 34
4.20. Parameter -flexAnalysisCapacity 34
4.21. Parameter -generateBuildId 35
4.22. Parameter -idHeader 35
4.23. Parameter -idHeaderInclude 36
4.24. Parameter -includeHeader 36
4.25. Parameter -initFeatureMask 37
4.26. Parameter -inlineHeader 37
4.27. Parameter -keyValueMapUE 38
4.28. Parameter -licInfo1 39
4.29. Parameter -licInfo2 39
4.30. Parameter -noRefT0p 40
4.31. Parameter -numberOfAdditionalCat1Isrs 40
4.32. Parameter -numberOfConstraints 41

4.33. Parameter -numberOfDelays 41
4.34. Parameter -numberOfUserStpws 42
4.35. Parameter -osBackgroundTaskId 42
4.36. Parameter -osBackgroundTaskName 43
4.37. Parameter -osBasicSchedFrameEventId 43
4.38. Parameter -osBasicSchedFrameId 44
4.39. Parameter -osBasicSchedFrameName 45
4.40. Parameter -projectFile 45
4.41. Parameter -projectName 46
4.42. Parameter -readPreviousT1p 46
4.43. Parameter -runnableHeader 47
4.44. Parameter -runnableHeaderInclude 47
4.45. Parameter -staticRunnableID 48
4.46. Parameter -storeTimingInformation 48
4.47. Parameter -symbolFile 49
4.48. Parameter -systemComment 49
4.49. Parameter -systemName 50
4.50. Parameter -t1pAnnotationsFile 50
4.51. Parameter -t1pSymbolIncludeFile 51
4.52. Parameter -targetExcludeFromTraceById 52
4.53. Parameter -targetExcludeFromTraceByName 53
4.54. Parameter -traceBufferEntries 53

5. *T1* Configuration File **54**
5.1. Parameter -bigEndian 54
5.2. Parameter -canBitrate 54
5.3. Parameter -canExtendedIds 55
5.4. Parameter -canFDBitrate 55
5.5. Parameter -canFDBitTiming 56
5.6. Parameter -canFDDataBitrate 57
5.7. Parameter -canFDDataBitTiming 57
5.8. Parameter -canFDDataSamplePoint 58
5.9. Parameter -canFDHardware 58
5.10. Parameter -canFDHwName 59
5.11. Parameter -canFDMustUseFixedBlockSize 59
5.12. Parameter -canFDOscillatorFrequencyMhz 60
5.13. Parameter -canFDMaxRxDataSize 60
5.14. Parameter -canFDSamplePoint 61
5.15. Parameter -canFDMaxTxDataSize 62
5.16. Parameter -canFDUsage 62
5.17. Parameter -canHardware 63
5.18. Parameter -canHwName 63
5.19. Parameter -canRxID 64
5.20. Parameter -canTxID 65
5.21. Parameter -canUsage 65
5.22. Parameter -checkIntegration 66
5.23. Parameter -commsCoreOffset 66

5.24. Parameter -connectionType 67
5.25. Parameter -contRunsOnCore 67
5.26. Parameter -cortex 68
5.27. Parameter -cortexMFPBversion 68
5.28. Parameter -cpuLoadAvgSamples 69
5.29. Parameter -cpuLoadTxPeriod 69
5.30. Parameter -diagAddressingMode 70
5.31. Parameter -diagCustomSessionId 71
5.32. Parameter -diagLocalIdentifier 71
5.33. Parameter -diagMinTxDataSize 72
5.34. Parameter -diagOpenSessionType 72
5.35. Parameter -diagMaxRxDataSize 73
5.36. Parameter -diagSourceAddr 74
5.37. Parameter -diagT1Identifier 74
5.38. Parameter -diagTargetAddr 75
5.39. Parameter -diagTesterPresentPeriod 75
5.40. Parameter -diagMaxTxDataSize 76
5.41. Parameter -diagUseServiceByLocalId 76
5.42. Parameter -dpcAlways 77
5.43. Parameter -dpcNever 77
5.44. Parameter -enableStreaming 78
5.45. Parameter -endiannessAgnosticComSetup 78
5.46. Parameter -ethEcuDNSHostName 79
5.47. Parameter -ethEcuIP 79
5.48. Parameter -ethEcuPort 80
5.49. Parameter -ethHardware 81
5.50. Parameter -ethHwName 81
5.51. Parameter -ethPcIP 82
5.52. Parameter -ethPcPort 83
5.53. Parameter -ethMaxRxDataSize 84
5.54. Parameter -ethTxCycleMs 84
5.55. Parameter -ethMaxTxDataSize 85
5.56. Parameter -ethUsage 85
5.57. Parameter -ethUseIpv6 86
5.58. Parameter -ethUseUdp 86
5.59. Parameter -maxPreemptionDepth 87
5.60. Parameter -mustUseFixedBlockSize 87
5.61. Parameter -numberOfFlexAddrs 88
5.62. Parameter -numberOfFlexStopwatches 88
5.63. Parameter -numberOfFocusMeasurements 89
5.64. Parameter -pSyncTimer 89
5.65. Parameter -pTimer 90
5.66. Parameter -rawTickDurationNs 91
5.67. Parameter -rawTimerBitLength 91
5.68. Parameter -retainIgnoreCase 92
5.69. Parameter -retainMax 93
5.70. Parameter -retainSymbol 93

5.71. Parameter -rxChannel 94
5.72. Parameter -sid 94
5.73. Parameter -symbolPrefix 95
5.74. Parameter -syncPeriodMs 95
5.75. Parameter -syncTimeBitLength 96
5.76. Parameter -syncTimerDownCounting 96
5.77. Parameter -syncTimerTickDurationNs 97
5.78. Parameter -systemType 98
5.79. Parameter -targetType 98
5.80. Parameter -t1FlexOverheadNs 99
5.81. Parameter -t1HandlerPeriodMs 99
5.82. Parameter -t1ScopeOverheadNs 100
5.83. Parameter -tickDurationNs 101
5.84. Parameter -timeoutResponseMs 101
5.85. Parameter -timeoutRxMs 102
5.86. Parameter -timeoutTxMs 102
5.87. Parameter -traceTimeStampBitLength 103
5.88. Parameter -traceTimerDownCounting 103
5.89. Parameter -traceTimerIsSyncTimer 104
5.90. Parameter -txChannel 105
5.91. Parameter -txCycleMs 105
5.92. Parameter -useSameConnectionForAllSystems 106
5.93. Parameter -useSameRxTxChannel 106
5.94. Parameter -usingMulticoreLibs 107

A. Revision History **108**
A.1. Migration from V2.6.y.z to V3.1.y.z 108
A.2. Migration from V2.5.y.z V2.6.y.z 110
A.3. Migration from V2.4.y.z V2.5.y.z 110
A.4. Migration from V2.3.y.z V2.4.y.z 110
A.5. Migration from V2.2.y.z V2.3.y.z 110
A.6. Migration from V2.1.y.z V2.2.y.z 110
A.7. Migration from V2.0.y.z V2.1.y.z 110

1. Introduction

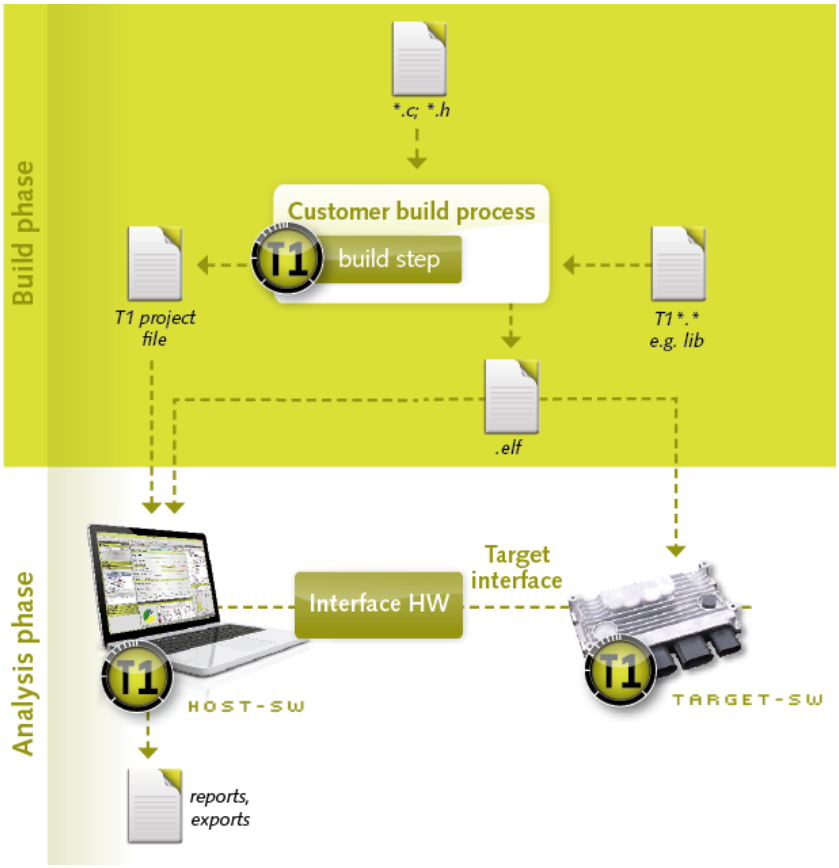

Figure 1: Overview : ***T1*** Build Phase and Analysis Phase

T1 is a measurement based timing analysis suite. It allows timing-debugging, timing-visualization, timing-measurements, timing-testing, timing-verification and timing-optimization. The product consists of the T1-HOST-SW running on a Windows PC and the T1-TARGET-SW. They communicate with each other via a target interface such as CAN or Ethernet, which is connected to the PC through the interface HW. The above figure shows this setup. At the target side no hardware changes are necessary. ***T1*** supports many processors, compilers and operating systems.

The process of generating an executable file from the C files, which consists of a compilation phase and a linking phase, is known as build process. The creation of a customized interface between the customer's application and the T1-TARGET-SW is done during the ***T1*** integration process.

There are a set of customer application specific functions which are adapted during the ***T1*** integration process. These functions are located in the following two files:

T1_AppInterface.c - Implements a set of functions to interface with the T1-TARGET-SW.

- Application wrappers for functions located in the T1-TARGET-SW libraries.

- Functions related to the communication interface for ***T1***.

T1_config.c - Implements a set of callback functions of the T1-TARGET-SW libraries.

- Configuration of the plugin table.
- Callback functions.

Additionally a central header file is provided which needs to be included wherever API functions of the T1-TARGET-SW are called:

T1_AppInterface.h - The main interface header which defines the following:

- Application features.
- Delays.
- Additional system elements such as user events, user stopwatches etc.

In the ***T1*** build step, a Perl script is used to generate the ***T1*** project file (*.t1p), *T1_config.h*, *T1_configGen.c* and further optional header files. During the execution of the Perl script the OS configuration is read as well as the ***T1*** configuration files, also called invocation files (*.inv). The following invocation files are used:

T1_UserCfg.inv - Here changes by the user can be performed.

T1_OsCfg.inv and T1_Cfg.inv - Typically no changes should be required in those files after the ***T1*** integration.

These invocation files are used to configure the T1-TARGET-SW for the system of the customer. The ***T1*** project file is generated automatically and matches the project specific settings in the T1-TARGET-SW.

To have a detailed overview about these configuration parameters, please refer to the Sections 4 and 5 which explain in detail all the parameters.

The T1-HOST-SW gets the information about the system elements (tasks, ISR's, application features, user events, ...) by reading the the ***T1*** project file (.t1p), which also includes the file *T1_AppInterface.h*. Additionally the symbol file is read by the T1-HOST-SW to get information about the functions and variables which is used by the T1.flex plugin.

2. Build ID Mechanism

During the build process a random number is generated for the current build. This number is taken as a reference or identifying number for a particular build known as build ID (BID).

The T1-HOST-SW checks that the BIDs match in ***T1*** project file, the loaded symbol file, and the executable running on the target. When there is any mismatch of the BIDs then T1-HOST-SW will display a warning message. An example is shown in Figure 2.

On the target side the BID is required by *T1_configGen.c*. The ***T1*** build step generates the BID by default into *T1_config.h*, which is included in *T1_configGen.c* via *T1_AppInterface.h*.

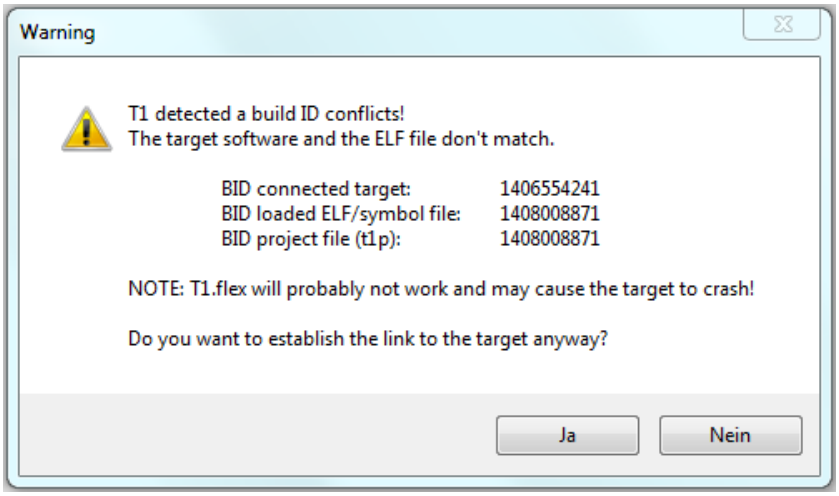

Figure 2: Warning by T1-HOST-SW for Build ID Mismatch

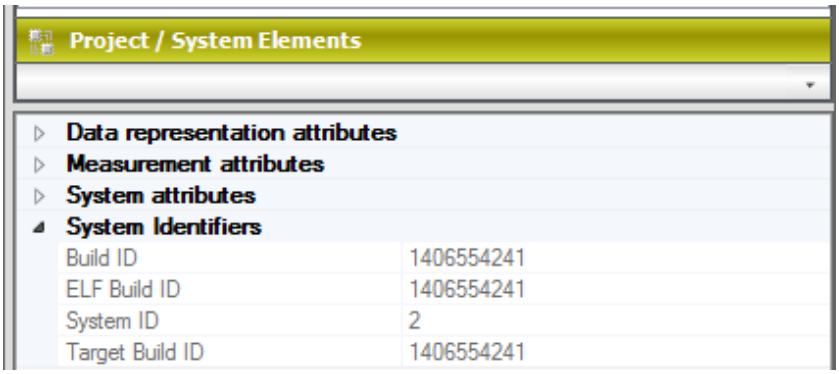

Figure 3: BID Overview in Project Explorer

The BID is re-generated by the ***T1*** build step during every execution of the Perl script. If the build duration is increased too much by a change in *T1_config.h*, the BID can be generated into its own header file. This can help to reduce the dependencies and hence limit the build duration increase. It can be enabled by specifying the parameter `-bidHeader` in *T1_UserCfg.inv*, see Section 4.9 for further details.

There is a possibility to disable the generation of the BID via the parameter `-generateBuildId=false`, see Section 4.21, although this is not recommended.

In the T1-HOST-SW the user can check the current status of the BIDs in the Project Explorer, as shown in Figure 3.

2.1. External Build ID

Ideally the BID should be generated at every run of the linker. However it is not sensible to change *T1_config.h*, or any header file, *after* compilation. If we change a header file at link time then we make a header file depend on the object files. But the object files already depend on the header files so this would create a circular dependency that will either completely break the build process or, at best, lead to wasteful recompila-

tion. An external (outside *T1_config.h* and therefore external to the ***T1*** configuration process) BID can be used, thereby avoiding this problem. See Section A.7.1 for information on earlier versions.

When `T1_EXTERNAL_BID` is defined (typically at the top of *T1_AppInterface.h*, before the includes), the symbols `T1_bid`, `T1_bid_xxxxx` and `T1_pBid` are not embedded in *T1_configGen.o* and must be provided externally. They should be generated immediately before linking, without changing *T1_config.h*.

If the external BID shall be used the following approach can be taken. GLIWA can provide on request an executable called *T1_bid.exe* which writes a random BID value to stdout and also replaces the BID in a given ***T1*** project file (.t1p) with the same value. This can be used in the build process as follows to define together with the template in Listing 1 the required variables. In this case the parameter `-bidHeader` (see Section 4.9) should be set to a dummy file name as the BID value generated by the Perl script is not used. Note that the build process must not treat *T1_bid.c* and *T1_bid.h* as normal C/H files, to avoid the circular dependency described above.

```
echo #define T1_BUILD_ID \> T1_bid.h
T1_bid.exe project.t1p >> T1_bid.h
```

Listing 1: Template T1_bid.c File

```
/*****************************************************************************************
* FILE: T1_bid.c
*
* DESCRIPTION: Compiled at link time to ensure a new T1 Build ID for each new
* link.
*
* $Author: andreaskni $
*
* $Revision: 26488 $
*
* Copyright: GLIWA GmbH embedded systems
* Weilheim i.OB.
* All rights reserved
*****************************************************************************************/

/*---------------------------------------------------------------------------------------*/
/*--- header includes -------------------------------------------------------------------*/
/*---------------------------------------------------------------------------------------*/

#include "T1_AppInterface.h"
#include "T1_bid.h"

/*---------------------------------------------------------------------------------------*/
/*--- global variables ------------------------------------------------------------------*/
/*---------------------------------------------------------------------------------------*/

const T1_uint8_t T1_CAT( T1_bid_, T1_BUILD_ID ) = 0u;
T1_uint8_t const * const T1_pBid = & T1_CAT( T1_bid_, T1_BUILD_ID );

const T1_uint32_t T1_bid = T1_CAT( T1_BUILD_ID , uL );
```

3. *T1* Perl script

The ***T1***Perl script is usually called with the following parameters:

```
perl.exe -I <Path to the T1 OS Perl module> ^
```

```
 -I <Path to T1_config.pm> ^
 <Path to>\T1_projGen.pl ^
-UserCfg=T1_UserCfg.inv ^
-Cfg=T1_Cfg.inv ^
-OsPm=<File name of the T1 OS Perl module> ^
-OsCfg=T1_OsCfg.inv
```

There are three different ***T1*** configuration files (*T1_UserCfg.inv*, *T1_Cfg.inv* and *T1_OsCfg.inv*) which are used as inputs to the T1-TARGET-SW build process, as described in Section 1. The two files *T1_UserCfg.inv* and *T1_Cfg.inv* are used as input for the ***T1*** Perl script (*T1_projGen.pl*) and *T1_OsCfg.inv* is used as input for the individual OS configuration parser Perl modules.

- The parameters that are should go into *T1_UserCfg.inv* are described in Section 4.
- The parameters that are should go into *T1_Cfg.inv* are described in Section 5.
- The parameters that are supported by *T1_OsCfg.inv* are specific to the individual OS configuration parser Perl modules and thus are not described in this document.

The two configuration files *T1_UserCfg.inv* and *T1_Cfg.inv* are in fact treated equally by the ***T1*** Perl module, but the parameters in this document are separated into parameters,

- that should only be set once during a ***T1*** integration, these usually go into *T1_Cfg.inv*,
- and parameters that may be changed later on by the user, these usually go into *T1_UserCfg.inv*.

Some parameters are global parameters that apply to all ***T1*** *systems* (cores), some are local parameters that only apply to a single ***T1*** system (core), and some can be applied globally or locally. The ***T1*** configuration files use the following syntax to tell the ***T1*** Perl script to which ***T1*** system the following parameters belong:

```
; Start of the configuration file
-paramA=1

#system 0:
-paramB=2
-paramC=3

#system 1:
-paramB=4
-paramC=5

#system all:
-paramD=6
```

`-paramA` and `-paramD` are both treated as global parameters, `-paramA`, because it is at the beginning of the file were `#system all:` is set implicitly and `-paramD`, because it is preceded by an explicit `#system all:`. `-paramB` and `-paramC` are all local parameters that are assigned to the ***T1*** *system 0* with the values 2 and 3 and to the ***T1*** *system 1* with the values 4 and 5. Parameters that appear on the command line are always global parameters.

It is possible to split the ***T1*** configuration files into several files. The files are then loaded by repeating the respective command line parameters `-UserCfg` (3.5) and `-Cfg` (3.1), this is not possible for the OS configuration file (`-OsCfg` (3.3)). The configuration files are processed in the order they appear on the command line. Therefore configuration files which are processed last can overwrite parameters that where set by previous configuration files.

Some parameters can also be set on the command line when calling the ***T1*** Perl script, two examples for this are the path to the application binary (ELF) file (`-symbolFile` (4.47)) and the path to and name of the ***T1*** project file (`-projectFile` (4.40)). The command line takes precedence if the same parameter is provided via an invocation file and on the command line.

3.1. Parameter -Cfg

Definition `-Cfg=`<*file name*>

Short description Specifies the path to and file name of the ***T1*** configuration file, see Section 5.

3.2. Parameter -I

Definition `-I` <*path*> or `-I`<*path*>

Short description User by Perl as a search path for additional search path for Perl modules.

See also `-OsPM` (3.4)

3.3. Parameter -OsCfg

Definition `-OsCfg=`<*file name*>

Short description Specifies the path to and file name of the ***T1*** OS configuration file.

3.4. Parameter -OsPm

Definition `-OsPm=<`*name*`>`

Short description Specifies the name of the ***T1*** OS configuration parser Perl module.

Detailed description Available OS configuration parser Perl modules are:

T1_FreeRTOS load *T1_FreeRTOS.pm*

T1_FreescaleOS load *T1_FreescaleOS.pm*

T1_Microsar load *T1_Microsar.pm*

T1_NonOs load *T1_NonOs.pm*

T1_Odin load *T1_Odin.pm*

T1_PXROSHR load *T1_PXROSHR.pm*

T1_ProOsek load *T1_ProOsek.pm*

T1_RtaOs load *T1_RtaOs.pm*

T1_SafeOs load *T1_SafeOs.pm*

T1_Tresos load *T1_Tresos.pm*

T1_VSTAR load *T1_VSTAR.pm*

T1_dummyOS load *T1_dummyOS.pm*

T1_gliwOS load *T1_gliwOS.pm*

The required Perl module is supplied by GLIWA during the ***T1*** integration.

See also `-I` (3.2)

3.5. Parameter -UserCfg

Definition `-UserCfg=<`*file name*`>`

Short description Specifies the path to and file name of the ***T1*** user configuration file, see Section 4.

4. *T1* User Configuration File

This section contains a complete list of all parameters which can be placed in the ***T1*** User Configuration Invocation File (*T1_UserCfg.inv*) including detailed information, examples, and cross-references.

4.1. Parameter -additionalCat1IsrIdOffset

Definition `-additionalCat1IsrIdOffset=<`*Offset of additional Cat1 ISR IDs*`>`

Short description Specifies the static offset of the external ISR IDs.

Detailed description This parameter is used if the external Cat1 ISRs are starting at a fixed offset. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-additionalCat1IsrIdOffset`.

```
-additionalCat1IsrIdOffset=40
; all external Cat1 ISR IDs start at
offset 40
```

As a result the following line is modified by setting `T1_EXT_ISR_ID_OFFSET_CORE0` in *T1_config.h* to the offset value.

```
#define T1_EXT_ISR_ID_OFFSET_CORE0 (40u)
```

See also `-allocateCat1IsrIdPrio` (4.4)
`-numberOfAdditionalCat1Isrs` (4.31)

4.2. Parameter -allocateAF

Definition `-allocateAF=<`*short name[, name[, init[, mask]]]*`>`

Short description Specifies that a ***T1*** "Application Feature" with a given name must be allocated.

Detailed description ***short name*** of the ***T1*** application feature, must only contain word characters (`[a-zA-Z0-9_]`). The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_AF_<`*short name*`>` This substitution will be skipped if the *short name* starts with `T1_`.

name (optional) is used for displaying the ***T1*** application feature in the T1-HOST-SW. The default is *short name*. The *name* can be put in quote marks in case it shall contain a comma character.

init (optional) specifies whether the ***T1*** application feature should be set during the initialization of the T1-TARGET-SW or not.

- `true`

- `false` (default)

mask (optional) can be used to assign a particular *bit mask* to the ***T1*** application feature. This parameter should not be used typically, as the ***T1*** Perl script will automatically assign *bit masks*. The *32-bit mask* shall be given as a hex value in the form `0x[0-9a-f]{1,8}`. **Please note**, that the position of this parameter might change in upcoming releases of the T1-TARGET-SW without further notice.

Please note, that the inline XML syntax (`/* @T1@ <AppFeature Name=...> */`) should not be mixed with the usage of this parameter to avoid the allocation of conflicting masks.

Example

The following example shows the usage of the parameter `-allocateAF`.

```
-allocateAF=T1_AF_TRACE_STARTUP, capture start-up of the ECU
-allocateAF=T1_AF_ALLOW_TRIGGER, , true
-allocateAF=T1_AF_TRACE_WHEELSPEED, , false
-allocateAF=T1_AF_TRACE_BM_RESULT
```

As a result the following entries are generated in *T1_config.h*,

```
#define T1_AF_TRACE_STARTUP (0x00000001uL)
#define T1_AF_ALLOW_TRIGGER (0x00000002uL)
#define T1_AF_TRACE_WHEELSPEED (0x00000004uL)
#define T1_AF_TRACE_BM_RESULT (0x00000008uL)

#define T1_INIT_FEATURE_MASK (T1_AF_ALLOW_TRIGGER)
```

and in the the ***T1*** project file

```
<AppFeature Name="capture start-up of the ECU">
  <Mask>1</Mask>
</AppFeature>
<AppFeature Name="T1_AF_ALLOW_TRIGGER">
  <Mask>2</Mask>
</AppFeature>
<AppFeature Name="T1_AF_TRACE_WHEELSPEED">
  <Mask>4</Mask>
</AppFeature>
<AppFeature Name="T1_AF_TRACE_BM_RESULT">
  <Mask>8</Mask>
</AppFeature>
```

See also

`-allocateUE` (4.6)
`-allocateUserSW` (4.7)
`-initFeatureMask` (4.25)

4.3. Parameter -allocateDL

Definition `-allocateDL=`*<short name[, name[, init value[, id]]]>*

Short description Specifies that a ***T1*** "Delay" with a given name must be allocated.

Detailed description The parameter `-numberOfDelays` will be automatically set to the number of allocated ***T1*** delays.

short name of the ***T1*** delay, must only contain word characters (`[a-zA-Z0-9_]`). The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_DL_<`*`short name`*`>_COREXX` where `COREXX` can be `CORE0` - `CORE15` or `CORE_ALL`, depending on the position in the invocation file:

- `#system all:` → `CORE_ALL`
- `#system 0:` → `CORE0`
- `#system 1:` → `CORE1`
- `#system 2:` → `CORE2`

This substitution will be skipped if the *short name* starts with `T1_`.

name (optional) is used for displaying the ***T1*** delay in the T1-HOST-SW. The default is *short name*. The *name* can be put in quote marks in case it shall contain a comma character.

init value (optional) can be used to define an initial value for the delay. The *init value* has to be provided as a time interval in the following format `[\d+[mu]s]`. The default value that is used is `0us`. Please note, that only whole numbers, with no decimal point, are allowed.

id (optional) can be used to assign a particular *ID* to the ***T1*** delay. This parameter should not be used typically, as the ***T1*** Perl script will automatically assign *ids*. The *id* shall be given as a decimal or hex value in the form `0x[0-9a-f]{2}`. The allowed range is 32 to 63 (0x20 to 0x3F). **Please note**, that the position of this parameter might change in upcoming releases of the T1-TARGET-SW without further notice.

Please note, that the inline XML syntax (`/* @T1@ <Delay Name="xyz"> */`) should not be mixed with the usage of this parameter to avoid the allocation of conflicting IDs.

Example The following example shows the usage of the parameter `-allocateDL`.

```
#system all:
-allocateDL=Task50ms_Runnable0, "Runnable0 in Task 50ms"
-allocateDL=Task50ms_Runnable1, "Runnable1 in Task 50ms", 100us
-allocateDL=Task50ms_Runnable2, "Runnable2 in Task 50ms", 200us
```

As a result the following entries are generated in *T1_config.h*,

```
/* Delays for all cores */
#define T1_DL_Task50ms_Runnable0_CORE_ALL (0u)
#define T1_DL_Task50ms_Runnable1_CORE_ALL (1u)
#define T1_DL_Task50ms_Runnable2_CORE_ALL (2u)

/* Delays for core 0 */
#define T1_DL_Task50ms_Runnable0_CORE0 \
   (T1_DL_Task50ms_Runnable0_CORE_ALL)
#define T1_DL_Task50ms_Runnable1_CORE0 \
   (T1_DL_Task50ms_Runnable1_CORE_ALL)
#define T1_DL_Task50ms_Runnable2_CORE0 \
   (T1_DL_Task50ms_Runnable2_CORE_ALL)
```

in *T1_configGen.c*,

```
static T1_delay_t const T1_SEC_CONST_8 T1_delayConfig0[3] =
{
  T1_US_TO_TICKS_CORE0( 0u ), /* Runnable0 in Task 50ms */
  T1_US_TO_TICKS_CORE0( 100u ), /* Runnable1 in Task 50ms */
  T1_US_TO_TICKS_CORE0( 200u ), /* Runnable2 in Task 50ms */
};
```

and in the the ***T1*** project file

```
<Delay Name="Runnable0 in Task 50ms" Id="0" />
<Delay Name="Runnable1 in Task 50ms" Id="1" />
<Delay Name="Runnable2 in Task 50ms" Id="2" />
```

See also `-numberOfDelays` (4.33)

4.4. Parameter -allocateCat1IsrIdPrio

Definition `-allocateCat1IsrIdPrio=`*<name of Cat1 ISR,Prio>*

Short description Specifies that a T1 "task" ID must be allocated for the given Category 1 ISR and defines its priority.

Detailed description This parameter is used in conjunction with manual instrumentation of Category 1 ISRs. The ISR is instrumented with T1_TraceStart(T1_CAT1_<name of Cat1 ISR>_ID) and T1_TraceStop(T1_CAT1_<name of Cat1 ISR>_ID)[1]. The appropriate macro is defined in T1_config.h and an interrupt element is inserted in the ***T1*** project file. The interrupt element is added to the system under which it is located in the invocation file. The priority given as parameter is incremented by a fixed offset of 2000.

Example The following example shows the usage of the parameter `-allocateCat1IsrIdPrio`.

[1] In practice, we usually know the interrupt state in a Category 1 ISR and would use the Fast or NoSusp variant of T1_TraceEvent.

```
-allocateCat1IsrIdPrio=isrDMA,100
; allocate Category 1 ISR ID
-allocateCat1IsrIdPrio=isrADC,101
; allocate Category 1 ISR ID
```

As a result the following entries are generated in *T1_config.h*.

```
#define T1_CAT1_isrDMA_ID (11u)
#define T1_CAT1_isrADC_ID (12u)
```

and in the the ***T1*** project file

```
<SystemElement Name="isrDMA" Priority="2100"
  ID="11" Type="Interrupt" />
<SystemElement Name="isrADC" Priority="2101"
  ID="12" Type="Interrupt" />
```

See also `-additionalCat1IsrIdOffset` (4.1)
`-numberOfAdditionalCat1Isrs` (4.31)

4.5. Parameter -allocateUDE

Definition `-allocateUDE=`<*short name[, name[, color[, id]]]*>

Short description Specifies that a ***T1*** "User Data Event" with a given name must be allocated.

Detailed description ***short name*** of the ***T1*** user data event, must only contain word characters (`[a-zA-Z0-9_]`). The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_UDE_<`*short name*`>_COREXX` where `COREXX` can be `CORE0` - `CORE15` or `CORE_ALL`, depending on the position in the invocation file:

- `#system all:` → `CORE_ALL`
- `#system 0:` → `CORE0`
- `#system 1:` → `CORE1`
- `#system 2:` → `CORE2`

This substitution will be skipped if the *short name* starts with `T1_`.

name (optional) is used for displaying the ***T1*** user data event in the T1-HOST-SW. The default is *short name*. The *name* can be put in quote marks in case it shall contain a comma character.

color (optional) specifies the color that shall be used to display the ***T1*** user data event in a downloaded T1.scope trace within the T1-HOST-SW. The color can be specified in HTML color format (e.g. `#008000` for the color green) or

as a predefined HTML color name (e.g. `Red` for the color red), a complete list of predefined colors can be found at the following web page: `https://www.w3schools.com/colors/colors_names.asp`

id (optional) can be used to assign a particular *ID* to the ***T1*** user data event. This parameter should not be used typically, as the ***T1*** Perl script will automatically assign *ids*. The *id* shall be given as a decimal or hex value in the form `0x[0-9a-f]{2}`. The allowed range is 0 to 31 (0x00 to 0x1F). **Please note**, that the position of this parameter might change in upcoming releases of the T1-TARGET-SW without further notice.

Please note, that the inline XML syntax (`/* @T1@ <SystemElement ... Type="UserEvent"> */`) should not be mixed with the usage of this parameter to avoid the allocation of conflicting IDs.

Example

The following example shows the usage of the parameter `-allocateUDE`.

```
#system all:
-allocateUDE=A_UDE, "A User Data Event"
#system 0:
-allocateUDE=ANOTHER_UDE, "Another User Data Event"
#system 1:
-allocateUDE=YET_ANOTHER_UDE, "Yet another User Data Event"
```

As a result the following entries are generated in *T1_config.h*,

```
/* User Events for all cores */
#define T1_UDE_A_UDE_CORE_ALL (0u)
/* User Events for core 0 */
#define T1_UDE_A_UDE_CORE0 (T1_UDE_A_UDE_CORE_ALL)
#define T1_UDE_ANOTHER_UDE_CORE0 (1u)
/* User Events for core 1 */
#define T1_UDE_A_UDE_CORE1 (T1_UDE_A_UDE_CORE_ALL)
#define T1_UDE_YET_ANOTHER_UDE_CORE1 (1u)
```

and in the the ***T1*** project file

```
<System Name="Core0">
  <SystemElement Name="A User Data Event"
                 ID="0" Type="UserEvent" />
  <SystemElement Name="Another User Data Event"
                 ID="1" Type="UserEvent" />
</System>
<System Name="Core1">
  <SystemElement Name="A User Data Event"
                 ID="0" Type="UserEvent" />
  <SystemElement Name="Yet another User Data Event"
                 ID="1" Type="UserEvent" />
</System>
```

See also

`-allocateAF` (4.2)
`-allocateUE` (4.6)
`-allocateUserSW` (4.7)

4.6. Parameter -allocateUE

Definition `-allocateUE=<`*short name[, name[, color[, id]]]*`>`

Short description Specifies that a ***T1*** "User Event" with a given name must be allocated.

Detailed description ***short name*** of the ***T1*** user event, must only contain word characters (`[a-zA-Z0-9_]`). The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_UE_<`*short name*`>_COREXX` where `COREXX` can be `CORE0` - `CORE15` or `CORE_ALL`, depending on the position in the invocation file:

- `#system all:` → `CORE_ALL`
- `#system 0:` → `CORE0`
- `#system 1:` → `CORE1`
- `#system 2:` → `CORE2`

This substitution will be skipped if the *short name* starts with `T1_`.

name (optional) is used for displaying the ***T1*** user event in the T1-HOST-SW. The default is *short name*. The *name* can be put in quote marks in case it shall contain a comma character.

color (optional) specifies the color that shall be used to display the ***T1*** user event in a downloaded T1.scope trace within the T1-HOST-SW. The color can be specified in HTML color format (e.g. `#008000` for the color green) or as a predefined HTML color name (e.g. `Red` for the color red), a complete list of predefined colors can be found at the following web page: `https://www.w3schools.com/colors/colors_names.asp`

id (optional) can be used to assign a particular *ID* to the ***T1*** user event. This parameter should not be used typically, as the ***T1*** Perl script will automatically assign *ids*. The *id* shall be given as a decimal or hex value in the form `0x[0-9a-f]{2}`. The allowed range is 32 to 63 (0x20 to 0x3F). **Please note**, that the position of this parameter might change in upcoming releases of the T1-TARGET-SW without further notice.

Please note, that the inline XML syntax (`/* @T1@ <SystemElement ... Type="UserEvent"> */`) should not be mixed with the usage of this parameter to avoid the allocation of conflicting IDs.

Example The following example shows the usage of the parameter `-allocateUE`.

```
#system all:
-allocateUE=T1_UE_T1_CONT_ERR_CORE0, Error callback c0, red
-allocateUE=T1_UE_T1_CONT_ERR_CORE1, Error callback c1, green
-allocateUE=T1_UE_T1_CONT_ERR_CORE2, Error callback c2, yellow
-allocateUE=T1_UE_T1_CONT_ERR_CORE3, Error callback c3, #F0F0F0
-allocateUE=T1_UE_T1_CONT_ERR_CORE4, Error callback c4
```

As a result the following entries are generated in *T1_config.h*,

```
/* User Events for all cores */
#define T1_UE_T1_CONT_ERR_CORE0 (32u) /* Error callback c0 */
#define T1_UE_T1_CONT_ERR_CORE1 (33u) /* Error callback c1 */
#define T1_UE_T1_CONT_ERR_CORE2 (34u) /* Error callback c2 */
#define T1_UE_T1_CONT_ERR_CORE3 (35u) /* Error callback c3 */
#define T1_UE_T1_CONT_ERR_CORE4 (36u) /* Error callback c4 */
```

and in the the ***T1*** project file

```
<SystemElement Name="Error callback c0" ID="32" Type="UserEvent">
  <Visualization Color="#FF0000" />
</SystemElement>
<SystemElement Name="Error callback c1" ID="33" Type="UserEvent">
  <Visualization Color="#008000" />
</SystemElement>
<SystemElement Name="Error callback c2" ID="34" Type="UserEvent">
  <Visualization Color="#FFFF00" />
</SystemElement>
<SystemElement Name="Error callback c3" ID="35" Type="UserEvent">
  <Visualization Color="#F0F0F0" />
</SystemElement>
<SystemElement Name="Error callback c4" ID="36" Type="UserEvent"/>
```

See also
`-allocateAF` (4.2)
`-allocateUDE` (4.5)
`-allocateUserSW` (4.7)
`-keyValueMapUE` (4.27)

4.7. Parameter -allocateUserSW

Definition `-allocateUserSW=<`*short name[, name[, type[, id]]]*`>`

Short description Specifies that a ***T1*** "user stopwatch" with a given name must be allocated.

Detailed description This parameter is used in conjunction with static ***T1*** user stopwatches instrumentation.

short name of the ***T1*** user stopwatch, must only contain word characters (`[a-zA-Z0-9_]`). The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_SW_<`*short name*`>_COREXX` where `COREXX` can be `CORE0` - `CORE15` or `CORE_ALL`, depending on the position in the invocation file:

- `#system all:` → `CORE_ALL`
- `#system 0:` → `CORE0`
- `#system 1:` → `CORE1`
- `#system 2:` → `CORE2`

This substitution will be skipped if the *short name* starts with `T1_`.

name (optional) is used for displaying the ***T1*** user stopwatch in the T1-HOST-SW. The default is *short name*. The *name* can be put in quote marks in case it shall contain a comma character.

type (optional) of the ***T1*** user stopwatch, this can be one of the following:

- `CET` → For a Core Execution Time stopwatch, this will use the macro `T1_CONT_CET_STOPWATCH( )`.
- `DA` → For a Data Age stopwatch, this will use the macro `T1_CONT_DATA_AGE_STOPWATCH( )`.
- `DEFAULT` (default) → This will use the macro `T1_CONT_DEFAULT_CONFIG_STOPWATCH( )` for initializing the stopwatch.
- `GET` → For a Gross Execution Time stopwatch, this will use the macro `T1_CONT_GET_STOPWATCH( )`.
- `SCOPE` → For a regular stopwatch that is not supervised by T1.cont, but visible in downloaded T1.scope traces.
- `SCOPE_DA` → For a Data Age stopwatch that is not supervised by T1.cont, but visible in downloaded T1.scope traces.

id (optional) can be used to assign a particular id to the ***T1*** user stopwatch. This parameter should not be used typically, as the ***T1*** Perl script will automatically assign *IDs* trying to optimize RAM utilization on the target device. **Please note**, that the position of this parameter might change in upcoming releases of the T1-TARGET-SW without further notice.

Please note, that the inline XML syntax (`/* @T1@ <SystemElement ... Type="Stopwatch"> */`) should not be mixed with the usage of this parameter to avoid the allocation of conflicting IDs.

Example The following example shows the usage of the parameter `-allocateUserSW`.

```
#system all:
-allocateUserSW=SYT_GET, T1_SW_SYT_GET_CORE_ALL, GET
#system 0:
-allocateUserSW=CHECK_INTEGRATION, "T1 Check Integration, Core 0"
#system 1:
-allocateUserSW=SYT
-allocateUserSW=T1SCOPE_ONLY, "T1.scope only stopwatch"', SCOPE
```

As a result the following entries are generated in *T1_config.h*,

```
/* User stopwatches for all cores */
/* The following user stopwatches are supervised by T1.cont */
#define T1_SW_SYT_GET_CORE_ALL (0u) /* T1_SW_SYT_GET_CORE_ALL */

/* User stopwatches for core 0 */
/* The following user stopwatches are supervised by T1.cont */
#define T1_SW_SYT_GET_CORE0 (T1_SW_SYT_GET_CORE_ALL) /* T1_SW_SYT_GET_CORE_ALL */
#define T1_SW_CHECK_INTEGRATION_CORE0 (1u) /* T1 Check Integration, Core 0 */

/* User stopwatches for core 1 */
/* The following user stopwatches are supervised by T1.cont */
#define T1_SW_SYT_GET_CORE1 (T1_SW_SYT_GET_CORE_ALL) /* T1_SW_SYT_GET_CORE_ALL */
#define T1_SW_SYT_CORE1 (1u) /* T1_SW_SYT_CORE1 */
/* The following user stopwatches are not supervised by T1.cont */
#define T1_SW_T1SCOPE_ONLY_CORE1 (2u) /* T1.scope only stopwatch */

/* Core-specific configuration for core 0 */
#define T1_NOF_USER_STPWS_CORE0 (2)

/* Core-specific configuration for core 1 */
#define T1_NOF_USER_STPWS_CORE1 (2)
```

in *T1_configGen.c*,

```
T1_stpwConfig_t T1_SEC_CONST_8 T1_stpwConfig0[2] =
{
  T1_CONT_GET_STOPWATCH( T1_SW_SYT_GET_CORE0 ),
  T1_CONT_DEFAULT_CONFIG_STOPWATCH( T1_SW_CHECK_INTEGRATION_CORE0 )
}

T1_stpwConfig_t T1_SEC_CONST_8 T1_stpwConfig1[2] =
{
  T1_CONT_GET_STOPWATCH( T1_SW_SYT_GET_CORE1 ),
  T1_CONT_DEFAULT_CONFIG_STOPWATCH( T1_SW_SYT_CORE1 )
}
```

and in the the ***T1*** project file

```
<System Name="Core0">
  <SystemElement Name="T1_SW_SYT_GET_CORE_ALL"
    ID="0" Type="Stopwatch" />
  <SystemElement Name="T1 Check Integration Core 0"
    ID="1" Type="Stopwatch" />
</System>
<System Name="Core1">
  <SystemElement Name="T1_SW_SYT_GET_CORE_ALL"
    ID="0" Type="Stopwatch" />
  <SystemElement Name="T1_SW_SYT_CORE1"
    ID="1" Type="Stopwatch" />
  <SystemElement Name="T1.scope only stopwatch"
    ID="2" Type="Stopwatch" />
</System>
```

See also

`-allocateAF` (4.2)
`-allocateUE` (4.6)

4.8. Parameter -analysisCapacity

Definition `-analysisCapacity=`<*T1.cont analysis capacity*>

Short description Limits the amount of processing which is done in each call to the T1.cont background handler.

Detailed description This parameter is used to define the maximum number of trace buffer entries which are analyzed during one call to `T1_ContBgHandler()`. If there is no reason to limit the amount of processing in the T1.cont background handler then this should be omitted as it defaults to the size of the trace buffer, ensuring that all traced events are processed by each call to `T1_ContBgHandler()`. This parameter applies to each system (core) individually.

Example The following example shows the usage of the parameter `-analysisCapacity`.

```
-analysisCapacity=1000
; maximum number of trace buffer entries
analyzed per T1.cont background handler call
(optional)
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CONT_ANALYSIS_CAPACITY_CORE0 (1000)
```

See also `-traceBufferEntries` (4.54)

4.9. Parameter -bidHeader

Definition `-bidHeader=`<*Name of generated **T1** build ID header file*>

Short description States name and path of the generated header file containing ***T1*** build ID.

Detailed description Defines name and path of the header file being generated containing ***T1*** build ID. This header file is automatically included into *T1_configGen.c*. If not specified, the ***T1*** build ID will be generated into the ***T1*** configuration header file (*T1_config.h*). Please see Chapter 2 for more details on the ***T1*** Build ID Mechanism.

Example The following example shows the usage of the parameter `-bidHeader`.

```
-bidHeader=..\..\T1_bidHeader.h
; Name and path of build ID header file
(optional).
```

4.10. Parameter -configBuffC

Definition `-configBuffC=`<*generated **T1** external trace buffer C file*>

Short description Name and path of the generated ***T1*** external trace buffer configuration C file.

Detailed description ***T1*** external trace buffer configuration and related data-structures are generated in this file when `-externalTraceBuffer` is set to true. If the customer has a T1-TARGET-SW license and requires a larger or smaller trace buffer, they can generate their own T1_configBuff.c and use this to replace exactly that part of the delivered software. The size of the trace buffer is set by default using the macro generated from `-traceBufferEntries`, it can be changed by defining globally or in T1_AppInterface.h the core specific macro T1_EXT_TRACEBUFFER_ENTRIES_COREx with x being the core ID.

Example The following example shows the usage of the parameter `-configBuffC`.

```
-configBuffC=..\..\T1_configBuff.c
; T1 external trace buffer file name and path.
```

See also `-externalTraceBuffer` (4.18)

4.11. Parameter -configGenC

Definition `-configGenC=`<***T1*** *generated configuration C file name*>

Short description Name and path of ***T1*** generated configuration C file

Detailed description ***T1*** data definitions required for the current configuration are generated. It is recommended to use the proposed filename *T1_configGen.c*.

Example The following example shows the usage of the parameter `-configGenC`.

```
-configGenC=..\..\T1_configGen.c
; T1 configuration file name and path.
```

See also `-configHeader` (4.12)

4.12. Parameter -configHeader

Definition `-configHeader=`<***T1*** *configuration header filename*>

Short description States name and path of ***T1*** configuration header file.

Detailed description Defines name and path of the header file being generated containing T1 configuration parameter macros (and task-/ISR-ID macros if `-idHeader` is not specified). The ***T1*** configuration header file contains all configuration macros that are customizable via the invocation files. It is recommended to use the proposed filename *T1_config.h*.

Example The following example shows the usage of the parameter `-configHeader`.

```
-configHeader=..\..\T1_config.h
; T1 configuration header file name and path.
```

See also `-idHeader` (4.22)
`-enableDoxygenComments` (4.17)

4.13. Parameter -configHeaderInclude

Definition `-configHeaderInclude=`<*including header file in T1_config.h*>

Short description Specifies name and path of an additional header file to be included in T1_config.h.

Detailed description This allows project specific header files to be included in the generated T1_config.h.

Example The following example shows the usage of the parameter `-configHeaderInclude`.

```
-configHeaderInclude="myFileToBeIncluded.h"
; Include header in T1_config.h.
```

As a result the following entry is generated in T1_config.h:

```
/*-----------------------------------------*/
/*--- header includes ---------------------*/
/*-----------------------------------------*/
#include "myFileToBeIncluded.h"
```

4.14. Parameter -copyIncludeHeader

Definition `-copyIncludeHeader=`<*true/false*>

Short description Enables copying of the header file which is referenced in the ***T1*** project file into its directory.

Detailed description If this parameter is set to `true`, then the header file which is included in the ***T1*** project file is copied into its directory. If it is set to `false` or not set at all, then a relative path is used for including the header file. In case the header file is only referenced via `-includeHeader` then it can optionally be copied into the path of the ***T1*** project file by using this parameter.

Example The following example shows the usage of the parameter `-copyIncludeHeader`.

```
-copyIncludeHeader=true
; copy included header file
[true/false(default)].
```

See also `-includeHeader` (4.24)

4.15. Parameter -copySymbolFile

Definition `-copySymbolFile=`<*true/false*>

Short description Enables copying of the symbol file which is referenced in the ***T1*** project file into its directory.

Detailed description If this parameter is set to `true`, then the symbol file which is included in the ***T1*** project file is copied into its directory. If it is set to `false` or not set at all, then a relative path is used for including the symbol file.

Example The following example shows the usage of the parameter `-copySymbolFile`.

```
-copySymbolFile=true
; [true/false(default)]
```

See also `-symbolFile` (4.47)

4.16. Parameter -cpuLoadThreshold

Definition `-cpuLoadThreshold=<`*CPU load threshold in percent*`>`

Short description This parameter defines the CPU load threshold for each system (core).

Detailed description T1.cont allows to have continuous supervision of the CPU load of each system and the parameter `-cpuLoadThreshold` allows the user to configure the CPU load threshold for each system.

T1.cont continuously calculates the CPU load and calls the CPU load callback function every time a BSF CPU load has been calculated, which then checks if there has been a CPU overload, determined by the parameter `-cpuLoadThreshold`. Therefore each system calls the same CPU load callback function, `T1_ContCPULoadCallbackPC()`. In turn, `T1_ContCPULoadCallbackPC()` calls `T1_ContCPULoadCallbackNoSuspPC()`. The default implementation of these functions halt tracing across all cores and log the event when there has been a CPU overload. The callback receives the CPU load threshold in *per 256th*, therefore the value provided to this parameter which is in *percent* will get converted into *per 256th*. Templates of these function can be found in *T1_config.c*

Prior to T1-TARGET-SW V3.1.0.0, each system called its own CPU load callback function. This was specified by a parameter and for more information regarding this, please see Section A.1.2.

Example The following example shows the usage of the parameter `-cpuLoadThreshold` for different cores.

```
#system 0:
-cpuLoadThreshold=0
; Threshold for CPU load callback [0..100 percent]
#system 1:
-cpuLoadThreshold=12.5%
#system 2:
-cpuLoadThreshold=95
#system 3:
-cpuLoadThreshold=100
#system 4:
-cpuLoadThreshold=100/3
#system 5:
-cpuLoadThreshold=208 / 256 * 100
```

As a result the following lines are generated in *T1_config.h*.

```
#define T1_CPULOAD_THRESHOLD_RAW_CORE0 ( 0u) /* 0.0% */
#define T1_CPULOAD_THRESHOLD_RAW_CORE1 ( 32u) /* 12.5% */
#define T1_CPULOAD_THRESHOLD_RAW_CORE2 (243u) /* 94.9% */
#define T1_CPULOAD_THRESHOLD_RAW_CORE3 (255u) /* 99.6% */
#define T1_CPULOAD_THRESHOLD_RAW_CORE4 ( 85u) /* 33.2% */
#define T1_CPULOAD_THRESHOLD_RAW_CORE5 (208u) /* 81.3% */
```

4.17. Parameter -enableDoxygenComments

Definition `-enableDoxygenComments=<`*false/after/before*`>`

Short description Enables the generation of *Doxygen* comments for certain macros generated in *T1_config.h* to further document those.

Detailed description The parameter can be set to one of the following values:

`false` (default) no additional comments

`after` place comments on the same line as the macro definition

`before` place comments in front of the macro definition

Example The following example shows the usage of the parameter `-enableDoxygenComments`.

```
-enableDoxygenComments=before ; Place Doxygen comments in front
; of the macro definition in T1_config.h
```

As a result comments like the following are generated in *T1_config.h*:

```
/*!
 * Specifies the maximum messages size sent by the T1-HOST-SW.
 */
#define T1_GCP_MAX_RX_FRAME_SIZE (24u)
```

See also `-configHeader` (4.12)

4.18. Parameter -externalTraceBuffer

Definition `-externalTraceBuffer=<`*true/false*`>`

Short description Enables the generation of the ***T1***trace buffers and related data-structures externally.

Detailed description This feature allows a supplier to deliver a configured T1-TARGET-SW to their customer with a separate module T1_configBuff.c for the trace buffer. The parameter can be set to one of the following values:

`true` create T1_configBuff.c with trace buffer and related data-structures.

`false` embed trace buffer and related data-structures in T1_configGen.c

See also `-configBuffC` (4.10)

4.19. Parameter -featureMaskChanged

Definition `-featureMaskChanged=`<*true/false*>

Short description Requests notification, on the relevant core, of any change to T1_featureMask by the T1-HOST-SW.

Detailed description When the T1-HOST-SW modifies T1_featureMask, `T1_Handler()` calls `T1_FeatureMaskChangedPC()` on each core for which `-featureMaskChanged=true`. This can be used to avoid wasteful polling of T1_featureMask.

Example The following example shows the usage of the parameter `-featureMaskChanged`.

```
#system 1:
-featureMaskChanged=true
; run T1_FeatureMaskChangedPC() on T1 logical core ID 1
; whenever the T1-HOST-SW updates T1_featureMask
```

As a result the following entry is generated in T1_configGen.c:

```
T1_featureMaskCallbackMask = 2u;
```

4.20. Parameter -flexAnalysisCapacity

Definition `-flexAnalysisCapacity=`<*T1.flex analysis capacity*>

Short description Limits the rate of trace events created by T1.flex measurements.

Detailed description This parameter is used to limit the rate of trace events created by T1.flex measurements in the trace buffer. This is done by specifying the maximum number of T1.flex exception handler calls within the ***T1*** handler period. When the limit is reached the T1.flex measurement is disabled until the next call to T1_Handler. This parameter needs to be applied for each system (core) individually. If this parameter is not set then the T1-HOST-SW uses $2^{32} - 1$ as limit.

Example The following example shows the usage of the parameter `-flexAnalysisCapacity`.

```
-flexAnalysisCapacity=2000;
; maximum allowed number of T1.flex exception
handler calls between two calls to T1_Handler.
```

As a result the following entry is generated in the ***T1*** project file:

```
<TargetAnalysis
 T1flexAnalysisCapacity="2000" />
```

4.21. Parameter -generateBuildId

Definition `-generateBuildId=`<*true/false*>

Short description Allows to switch off the build ID generation (see Section 2).

Detailed description By setting this parameter to `false` the build ID generation is disabled. This is achieved by using the constant build ID of `123456`. Disabling the build ID brings the danger that a wrong or outdated symbol file may be used without being noticed by the user unless an alternative mechanism generates a unique build ID. Leaving the build ID unchanged ensures that no new files are created when there is no change to the T1 configuration, which may be important in a 'make'-based build environment. To facilitate the checking for changes, a reference copy of the generated ***T1*** project file is stored with the file extension `.t0p` if the generation of the build ID is disabled and `-noRefT0p` is set to `false` (default).

Example The following example shows the usage of the parameter `-generateBuildId`.

```
-generateBuildId=false
; [true(default)/false]
```

See also `-noRefT0p` (4.30)

4.22. Parameter -idHeader

Definition `-idHeader=`<*Name of ID header file* >

Short description States name and path of the generated header file containing T1 task- and ISR-ID macros.

Detailed description Defines name and path of the header file being generated containing T1 task- and ISR-ID macros. If specified no ID macros are generated in the ***T1*** configuration header file.

Example The following example shows the usage of the parameter `-idHeader`.

```
-idHeader=..\..\T1_ids.h
; Name and path of ID header file (optional).
```

4.23. Parameter -idHeaderInclude

Definition `-idHeaderInclude=`<*header file to be included*>

Short description Specifies name and path of include files in the generated ID header file.

Detailed description This parameter allows to add additional include files in the generated header file, which is defined by the parameter `-idHeader` (4.22)

Example The following example shows the usage of the parameter `-idHeaderInclude`.

```
-idHeaderInclude=idHeader.h
; file to be included in the generated ID header file (optional).
```

4.24. Parameter -includeHeader

Definition `-includeHeader=`<*Filename*>

Short description Specifies name and path for referencing *T1_AppInterface.h* in ***T1*** project file.

Detailed description Defines the header file which is referenced in the ***T1*** project file. As an alternative use the parameter `-inlineHeader` (4.26)

Example The following example shows the usage of the parameter `-includeHeader`.

```
-includeHeader=.\src\T1_AppInterface.h
; The header file will only be referenced
in the \tone{} project file (optional).
```

As a result the following entry is generated in the T1 project file:

```
<HeaderInclude>
 .\src\T1_AppInterface.h
</HeaderInclude>
```

See also `-inlineHeader` (4.26)
`-copyIncludeHeader` (4.14)

4.25. Parameter -initFeatureMask

Definition `-initFeatureMask=`<*feature mask init. value*>

Short description Initial value of application feature mask.

Detailed description This parameter defines the initial value of the application feature mask. The application features are defined in `T1_AppInterface.h` and the application features can be activated at users choice at startup of the target with this parameter.

Example The following example shows the usage of the parameter `-initFeatureMask`.

```
-initFeatureMask=T1_AF_TRACE_STARTUP |
T1_AF_MEASURE_T1_HANDLER
; Application features selected
```

As a result the following entry is generated in `T1_config.h`:

```
#define T1_INIT_FEATURE_MASK
(T1_AF_TRACE_STARTUP |
T1_AF_MEASURE_T1_HANDLER)
```

See also `-allocateAF` (4.2)

4.26. Parameter -inlineHeader

Definition `-inlineHeader=`<*Filename*>

Short description Specifies name and path for in-lining T1_AppInterface.h into the ***T1*** project file.

Detailed description The contents of this header file is copied into the ***T1*** project file as external data. As an alternative use the parameter `-includeHeader` (4.24)

Example The following example shows the usage of the parameter `-inlineHeader`.

```
-inlineHeader=.\src\T1_AppInterface.h
; The header file will be copied to
the external data section of the
\tone{} project file (optional).
```

As a result the following entry is generated in the ***T1*** project file:

```
<ExternalData>
 <HeaderContent>
 ...
 </HeaderContent>
</ExternalData>
```

See also `-includeHeader` (4.24)

4.27. Parameter -keyValueMapUE

Definition `-keyValueMapUE=`<*short name, info, description*>

Short description Associate a text description with an info value for a given ***T1*** "User Event". This parameter should always be used in conjunction with `-allocateUE` (4.6).

Detailed description

short name of the ***T1*** user event, must only contain word characters (`[a-zA-Z0-9_]`). The short name shall be the same as for a previously allocated ***T1*** "User Event". The short name will be substituted to form a C macro name, this substitution follows the following rule: `T1_UE_<`*`short name`*`>_COREXX` where `COREXX` can be `CORE0` - `CORE15` or `CORE_ALL`, depending on the position in the invocation file:

- `#system all:` → `CORE_ALL`
- `#system 0:` → `CORE0`
- `#system 1:` → `CORE1`
- `#system 2:` → `CORE2`

This substitution will be skipped if the *short name* starts with `T1_`.

info is the specific value of the user event's info field.

description specifies the text to show when the T1-HOST-SW displays this event with this info field value.

Example The following example shows the usage of the parameter `-keyValueMapUE`.

```
#system all:
-allocateUE=T1_UE_T1_CONT_ERR_CORE0, Error callback c0, red
-keyValueMapUE=T1_UE_T1_CONT_ERR_CORE0, 0x01, "INF: Overheads calibrated"
```

As a result the following entries are generated in *T1_config.h*,

```
/* User Events for all cores */
#define T1_UE_T1_CONT_ERR_CORE0 (32u) /* Error callback c0 */
```

and in the the ***T1*** project file

```
<SystemElement Name="Error callback c0" ID="32" Type="UserEvent">
  <Info InfoStr="INF: Overheads calibrated" Index="0x01" />
</SystemElement>
```

See also `-allocateUE` (4.6)

4.28. Parameter -licInfo1

Definition `-licInfo1=`<*POTxxx*>

Short description Allows to define a project reference meaningful to GLIWA.

Detailed description This parameter is used to define a project reference which allows GLIWA employees to identify the corresponding customer project.

Example The following example shows the usage of the parameter `-licInfo1`.

```
-licInfo1=POT123
; Project reference meaningful to GLIWA
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ... LicInfo1="POT123" ... >
```

See also `-licInfo2` (4.29)

4.29. Parameter -licInfo2

Definition `-licInfo2=`<*LicIDyyyy*>

Short description Allows to define a license reference meaningful to GLIWA.

Detailed description This parameter is used to define a license reference which allows GLIWA employees to identify the corresponding customer license.

Example The following example shows the usage of the parameter `-licInfo2`.

```
-licInfo2=LicID0001
; License reference meaningful to GLIWA
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ... LicInfo2="LicID0001" ... >
```

See also `-licInfo1` (4.28)

4.30. Parameter -noRefT0p

Definition -noRefT0p=<*true/false*>

Short description Suppress generation of `.t0p` file, even if the build ID generation is disabled (`-generateBuildId=false`).

Detailed description By setting this parameter to `true` the generation of a reference ***T1*** project file (`.t0p`) is disabled, even if the generation of the build ID is disabled (`-generateBuildId=false`), see Section 4.21.

Example The following example shows the usage of the parameter `-noRefT0p`.

```
-noRefT0p=false ; [true/false(default)]
```

See also `-generateBuildId` (4.21)

4.31. Parameter -numberOfAdditionalCat1Isrs

Definition -numberOfAdditionalCat1Isrs=<*number of External Cat1 ISRs*>

Short description Specifies the number of additional ISRs that are not covered by OS-specific module.

Detailed description This parameter is used in order to specify the number of additional external ISRs. As a result of using this parameter the number of tasks for the selected core is increased accordingly. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-numberOfAdditionalCat1Isrs`.

```
-numberOfAdditionalCat1Isrs=3
; add three external cat1 ISR.
```

As a result the following line is modified by adding (3) to `T1_NOF_TASKS_CORE0` in *T1_config.h*.

```
#define T1_NOF_TASKS_CORE0 (13u)
```

See also `-allocateCat1IsrIdPrio` (4.4)
`-additionalCat1IsrIdOffset` (4.1)

4.32. Parameter -numberOfConstraints

Definition `-numberOfConstraints=`<*Number of constraints*>

Short description Specifies the number of constraints.

Detailed description This parameter specifies the maximum number of constraints. Up to 64 constraints are allowed per system (core). The default value is 1. This parameter needs to be applied for each system (core) individually. An alternative to using this parameter is to add the macro `T1_NOF_CSRNS_COREx` in *T1_AppInterface.h* (please refer to Section *Constraints* in *40_T1-TARGET-SW_API.chm*).

Example The following example shows the usage of the parameter `-numberOfConstraints`.

```
-numberOfConstraints=2 ; number of constraints [1..64] (optional)
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_NOF_CSRNS_CORE0 (2)
```

4.33. Parameter -numberOfDelays

Definition `-numberOfDelays=`<*Number of delays*>

Short description Defines the number of delays for T1.delay.

Detailed description This parameter is used to define the number of delay routines which can be introduced by T1.delay. Up to 255 delay routines are allowed per system (core). This parameter needs to be applied for each system (core) individually. An alternative to using this parameter is to add the enumerator `T1_NOF_DELAYS_COREx` in *T1_AppInterface.h* (please refer to Section *Delays* in *40_T1-TARGET-SW_API.chm*).

Example The following example shows the usage of the parameter `-numberOfDelays`.

```
-numberOfDelays=10 ; Number of delays [1..255] (optional)
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_NOF_DELAYS_CORE0 (10)
```

4.34. Parameter -numberOfUserStpws

Definition `-numberOfUserStpws=`<*Number of user stopwatches*>

Short description Specifies the number of user stopwatches.

Detailed description This parameter specifies number of user stopwatches analyzed by T1.cont. Please note, that the maximum for the total number of stopwatches is 255 and that stopwatches are shared between:

1. User stopwatches
2. Event chains (either 8 or 64, please refer to Section *Event chains* in *40_T1-TARGET-SW_API.chm*)
3. T1.flex stopwatches (see 5.62)

This parameter needs to be applied for each system (core) individually. An alternative to using this parameter is to add the macro `T1_NOF_USER_STPWS_COREx` in *T1_AppInterface.h* (see *40_T1-TARGET-SW_API.chm*).

Example The following example shows the usage of the parameter `-numberOfUserStpws`.

```
-numberOfUserStpws=3 ; Number of user stopwatches [0..255]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_NOF_USER_STPWS_CORE0 (3)
```

See also `-numberOfFlexStopwatches` (5.62)

4.35. Parameter -osBackgroundTaskId

Definition `-osBackgroundTaskId=`<*System Comment*>

Short description Specifies the ID of the background task, so that it can be excluded from the T1.cont and T1.scope CPU load calculation.

Detailed description This optional parameter is an alternative to `-osBackgroundTaskName`. The task with this ID is excluded from the T1.cont and T1.scope CPU load calculation.

Example The following example shows the usage of the parameter `-osBackgroundTaskId`.

```
-osBackgroundTaskId=10
; osBackground Task ID (optional)
```

As a result the following entries are generated in the ***T1*** project file

```
<SystemElement ... ID="10" ... ExcludeFromCpuLoad="true" .../>
```

and in the ***T1*** configuration header file

```
#define T1_BACKGROUND_TASK_ID_CORE0 (10)
```

See also `-osBackgroundTaskName` (4.36)

4.36. Parameter -osBackgroundTaskName

Definition `-osBackgroundTaskName=`<*OS background task name*>

Short description Specifies the name of the background task so that it can be excluded from the T1.cont and T1.scope CPU load calculation.

Detailed description This optional parameter specifies the OS-name of the background task. The task with this name is excluded from the T1.cont and T1.scope CPU load calculation. If neither this parameter nor `-osBackgroundTaskId` are used then `T1_INVALID_TASK_ID` is used and the background task is not excluded.

Example The following example shows the usage of the parameter `-osBackgroundTaskName`.

```
-osBackgroundTaskName=myBgTask;
; OS name of background task (optional).
```

As a result the following entries are generated in the ***T1*** project file

```
<SystemElement Name="myBgTask" ... ExcludeFromCpuLoad="true" .../>
```

and in the ***T1*** configuration header file

```
#define T1_BACKGROUND_TASK_ID_CORE0 (T1_myBgTask_ID)
```

See also `-osBackgroundTaskId` (4.35)

4.37. Parameter -osBasicSchedFrameEventId

Definition `-osBasicSchedFrameEventId=`<*T1_START/T1_ACTIVATION/T1_USEREVENT*>

Short description Specifies the event ID which starts the Basic Scheduling Frame.

Detailed description This parameter gives the event ID used for the Basic Scheduling Frame (BSF). It can be one of the following values: `T1_START/T1_ACTIVATION/T1_USEREVENT`. In case `T1_USEREVENT` is selected, use the parameter `-osBasicSchedFrameId` for specifying the ID of the user event. The default value is `T1_ACTIVATION`.

Example The following example shows the usage of the parameter `-osBasicSchedFrameEventId`.

```
-osBasicSchedFrameEventId=T1_ACTIVATION;
; BSF: Event ID [T1_ACTIVATION(d),T1_START]
(optional).
```

As a result the following entries are generated in the ***T1*** project file

```
<BasicSchedulingFrameEvent
 Type="T1_ACTIVATION" ... />
```

and in the ***T1*** configuration header file

```
#define T1_BSF_EVENT_ID_CORE0 (T1_ACTIVATION)
```

See also `-osBasicSchedFrameName` (4.39)
`-osBasicSchedFrameId` (4.38)

4.38. Parameter -osBasicSchedFrameId

Definition `-osBasicSchedFrameId=`*<Task or user event ID for Basic Scheduling Frame event>*

Short description ID of the task or user event for the Basic Scheduling Frame (BSF) event. The selected task or user event must be traced on this core.

Detailed description This parameter gives the ID of the task or user event used for the Basic Scheduling Frame (BSF) event. The BSF event determines the boundaries for CPU load calculation. Alternatively use `-osBasicSchedFrameName` if the OS task name is known. This value can be given as a hexadecimal number as well, e.g. 0x22. See Section A.1.1 for information on earlier versions.

Example The following example shows the usage of the parameter `-osBasicSchedFrameId`.

```
-osBasicSchedFrameId=2 ;
; Task ID for BSF
```

As a result the following entries are generated in the ***T1*** project file

```
<BasicSchedulingFrameEvent ...
 SystemElementID="2" ... />
```

and in the ***T1*** configuration header file

```
#define T1_BSF_EVENT_INFO_CORE0 (2)
```

See also `-osBasicSchedFrameName` (4.39)
`-osBasicSchedFrameEventId` (4.37)

4.39. Parameter -osBasicSchedFrameName

Definition `-osBasicSchedFrameName=`*<OS task name for Basic Scheduling Frame event>*

Short description The OS name of the task used for the Basic Scheduling Frame (BSF) event. This task must run on this core.

Detailed description This parameter gives the OS name of the task used for the Basic Scheduling Frame (BSF) event. The BSF event determines the boundaries for CPU load calculation. Alternatively use `-osBasicSchedFrameId` if only the task ID is known.

Example The following example shows the usage of the parameter `-osBasicSchedFrameName`.

```
-osBasicSchedFrameName=myTaskName
; Task name for BSF as provided by
OS-configuration
```

As a result the following entries are generated in the ***T1*** project file

```
<BasicSchedulingFrameEvent ...
 SystemElementID="2" ... />
```

and in the ***T1*** configuration header file

```
#define T1_BSF_EVENT_INFO_CORE0
 (T1_myTaskName_ID)
```

See also `-osBasicSchedFrameId` (4.38)
`-osBasicSchedFrameEventId` (4.37)

4.40. Parameter -projectFile

Definition `-projectFile=`*<**T1** project filename>*

Short description States the filename and optionally the path of the generated ***T1*** project file.

Detailed description Defines name and path of ***T1*** project file being generated. The path and filename must not contain any whitespace characters. A generic Perl script T1_projGen.pl and an OS-specific Perl-module allow to generate header files as well as the ***T1*** project

file in the specified path. A set of invocation files holds the project configuration and serves as input for the Perl script (T1_Cfg.inv, T1_UserCfg.inv, T1_OsCfg.inv).

Example The following example defines the name and path of a ***T1*** project file.

```
-projectFile=..\..\myApp.t1p
; Project filename.
```

4.41. Parameter -projectName

Definition `-projectName=`<*Name*>

Short description Defines the name of the T1 project.

Detailed description This parameter allows to define the project name which is shown in the 'Project Explorer' of the T1-HOST-SW.

Example The following example shows the usage of the parameter `-projectName`.

```
-projectName=My T1 Project
; Project name meaningful to customer
```

As a result the following entry is generated in the ***T1*** project file:

```
<Project Name="My T1 Project" ...
 ...
</Project>
```

4.42. Parameter -readPreviousT1p

Definition `-readPreviousT1p=`<*true/false*>

Short description Allows to read-in user-configured parameters from an already existing ***T1*** project file.

Detailed description If this parameter is set to `true` which is the default value, then the user-configured parameters of an already existing ***T1*** project file are read-in and retained. Setting it to `false` allows to disable this behavior.

Example The following example shows the usage of the parameter `-readPreviousT1p`.

```
-readPreviousT1p=false ; [true(default)/false]
; disable reading-in of user-configured
parameters from existing \tone{} project file
```

4.43. Parameter -runnableHeader

Definition `-runnableHeader=`*<Name of generated runnable header file>*

Short description States name and path of the generated runnable header file containing runnable IDs.

Detailed description Defines name and path of the header file being generated containing ***T1*** definitions for runnable start/stop macros (only if supported by OS-specific Perl-module).

Example The following example shows the usage of the parameter `-runnableHeader`.

```
-runnableHeader=..\..\T1_runnables.h
; Name and path of runnable header file
(optional).
```

4.44. Parameter -runnableHeaderInclude

Definition `-runnableHeaderInclude=`*<header file to be included>*

Short description Specifies name and path of include files in the generated runnables header file.

Detailed description This parameter allows to add additional include files in the generated runnable header file, which is defined by the parameter `-runnableHeader` (4.43).

Example The following example shows the usage of the parameter `-runnableHeaderInclude`.

```
-runnableHeaderInclude="runnablesHeader.h"
; file to be included in the generated runnable header file
(optional).
```

4.45. Parameter -staticRunnableID

Definition `-staticRunnableID=`<*Static runnable ID*>

Short description Defines the value of the static runnable ID.

Detailed description This parameter is used to define the value of the static runnable ID which can be used by static instrumentation to trace runnables.

Example The following example shows the usage of the parameter `-staticRunnableID`.

```
-staticRunnableID=980 ; [0..1023].
```

As a result the following entries are generated in the ***T1*** project file for each system

```
<System Name="CoreX" ... StaticRunnableID="980">
```

and in the ***T1*** configuration header file

```
#define T1_STATIC_RUNNABLE_ID (980u)
```

4.46. Parameter -storeTimingInformation

Definition `-storeTimingInformation=`<*true/false*>

Short description Allows to enable storing of timing information in the ***T1*** project file.

Detailed description If this parameter is set to `true` then the timing information is stored in the ***T1*** project file, which is for example used in the T1.diff feature. By default the timing information is not stored in the ***T1*** project file.

Example The following example shows the usage of the parameter `-storeTimingInformation`.

```
-storeTimingInformation=true ;
; store timing information in T1 project
file [true/false].
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ...
StoreTimingInformation="true"
...
</System>
```

4.47. Parameter -symbolFile

Definition `-symbolFile=`<*Name of symbol file*>

Short description Specifies name and path of the symbol file(s) in ELF format.

Detailed description Defines name and path of the symbol file(s) being referenced by the ***T1*** project file. The asterisk pattern character is allowed as wildcard. For most linkers the symbol filename extension is *.ELF. Other extensions are *.ELS (for diab-compilers), *.OUT (TI-Compiler) and *.AXF(GCC-Compiler). In case T1.flex is used it is mandatory to provide the symbol file.

Initially this was a single, global parameter only, so that one single symbol file could be added to each system (core). Starting with SVN revision 36357 of T1_projGen.pl this can be used multiple-times and as core-specific parameter as well. If this parameter is used prior to any system sections in the invocation file then it generates a global symbol file for all systems. However, if the parameter is located in a core-specific section then this symbol file is added only for the current system. The wildcard is only supported for global symbol files.

Example The following example shows the usage of the parameter `-symbolFile`.

```
-symbolFile=.\output\executable.elf
; Symbol file referenced in \tone{} project file.
```

As a result the following entry is generated in the ***T1*** project file:

```
<Symbols>
 <SymbolFilePath>
  .\output\executable.elf
 </SymbolFilePath>
</Symbols>
```

4.48. Parameter -systemComment

Definition `-systemComment=`<*System Comment*>

Short description Adds a comment to a system.

Detailed description This parameter is used to add information about the system as a comment.

Example The following example shows the usage of the parameter `-systemComment`.

```
-systemComment=any comment
; any comment related to the system.
```

As a result the following entry is generated in the ***T1*** project file:

```
<Comment>any comment</Comment>
```

4.49. Parameter -systemName

Definition `-systemName=`<*System Name*>

Short description Specifies the system name being displayed in in the T1-HOST-SW.

Detailed description This parameter is used to define a unique system name for each system in the project.

Example The following example shows the usage of the parameter `-systemName`.

```
-systemName=Core0 ; Any unique system name
[recommended: <8 characters].
```

As a result the following entry is generated in the ***T1*** project file:

```
<System Name="Core0" ... >
```

4.50. Parameter -t1pAnnotationsFile

Definition `-t1pAnnotationsFile=`<*Path and name of annotations files*>

Short description Specifies name and path of an annotations file to include.

Detailed description This parameter allows to include annotations from one file written or generated by customer-specific tools. This parameter can be repeated to include multiple files. If this parameter is used prior to any system sections in the invocation file then it generates a global include for all systems. However, if the parameter is located in a core-specific section then this include will be generated only for the current system. When multiple files are specified, they will be processed in order of filename.

Example The following example shows the usage of the parameter `-t1pAnnotationsFile`.

```
-t1pAnnotationsFile=..\..\T1_annotations.t1a
; Path and name of Annotation Include file
(optional).
```

As a result the following entry is generated in the ***T1*** project file:

```
<Symbols>
 ...
 <AnnotationInclude>..\..\T1_annotations.t1a</AnnotationInclude>
</Symbols>
```

The syntax of the files to be included in order to define a annotation is as follows:

```
<Annotations>
  <Indirect FromFunction="[Function A source of the call]"
    To="[Function A' target of the call]"
    AnnotationSourceType="Manual" Optional="true" />
  <Indirect FromFunction="[Function B source of the call]"
    To="[Function B' target of the call]"
    AnnotationSourceType="Manual" Optional="true" />
</Annotations>
```

4.51. Parameter -t1pSymbolIncludeFile

Definition `-t1pSymbolIncludeFile=`*<Path and name of Symbol Include file>*

Short description Specifies name and path of a symbol include file, containing for example symbol group information.

Detailed description This parameter allows to include symbol groups from one or more files generated by customer-specific tools, additionally to what can be provided by the OS-specific Perl-module. This can be for example a list of functions being used in a certain software component. The syntax of the files being included is similar to a ***T1*** project file. If this parameter is used prior to any system sections in the invocation file then it generates a global include for all systems. However, if the parameter is located in a core-specific section then this include will be generated only for the current system.

Example The following example shows the usage of the parameter `-t1pSymbolIncludeFile`.

```
-t1pSymbolIncludeFile=..\T1_symbols.t1p
; Path and name of Symbol Include file
(optional).
```

As a result the following entry is generated in the ***T1*** project file:

```
<Symbols>
 ...
 <Include>..\T1_symbols.t1p</Include>
</Symbols>
```

The syntax of the files to be included in order to define a symbol group is as follows:

```
<T1SymbolsPart>
  <Symbols>
    <SymbolGroup Name="MyGroup" IsCode="true"
     MeasureDuration="2000" FocusIntervalCount="1024">
       <Symbol>functionA</Symbol>
       <Symbol>functionB</Symbol>
    </SymbolGroup>
  </Symbols>
</T1SymbolsPart>
```

4.52. Parameter -targetExcludeFromTraceById

Definition `-targetExcludeFromTraceById=`*<task ID>*

Short description Allows to exclude a task from being traced.

Detailed description This parameter is used to exclude a task from being traced via its ID. This can be used for example to exclude a background task from the trace. This optional parameter can be used multiple times if more than one task needs to be excluded. It is only allowed to exclude the group of tasks at the lowest used priority level, as otherwise the CET and CPU load results will be incorrect! Also it needs to be noted that this parameter has an effect only if the OS instrumentation uses the macro `T1_TRACE_TASK`.

Example The following example shows the usage of the parameter `-targetExcludeFromTraceById`.

```
-targetExcludeFromTraceById=2
; ID of task being excluded
from tracing.
```

As a result the excluded task is added to the macro `T1_TRACE_TASK` as follows:

```
#define T1_TRACE_TASK( TASK_ID ) \
( 1 \
 && (TASK_ID != 2) \
)
```

See also `-targetExcludeFromTraceByName` (4.53)

4.53. Parameter -targetExcludeFromTraceByName

Definition `-targetExcludeFromTraceByName=<`*task name*`>`

Short description Allows to exclude a task from being traced.

Detailed description This parameter is used to exclude a task from being traced via its name. This can be used for example to exclude a background task from the trace. This optional parameter can be used multiple times if more than one task needs to be excluded. It is only allowed to exclude the group of tasks at the lowest used priority level, as otherwise the CET and CPU load results will be incorrect! Also it needs to be noted that this parameter has an effect only if the OS instrumentation uses the macro `T1_TRACE_TASK`.

Example The following example shows the usage of the parameter `-targetExcludeFromTraceByName`.

```
-targetExcludeFromTraceByName=osIdleTask ;
; OS-name of task being excluded from
tracing
```

As a result the excluded task is added to the macro `T1_TRACE_TASK` as follows:

```
#define T1_TRACE_TASK( TASK_ID ) \
( 1 \
 && (TASK_ID != T1_osIdleTask_ID) \
)
```

See also `-targetExcludeFromTraceById` (4.52)

4.54. Parameter -traceBufferEntries

Definition `-traceBufferEntries=<`*Number of trace buffer entries*`>`

Short description Defines the number of trace buffer entries.

Detailed description This parameter is used to define the size of the trace buffer(s) allocated on the target. The size of each trace buffer entry is four bytes. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-traceBufferEntries`.

```
-traceBufferEntries=1000
; number of trace buffer entries
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_TRACEBUFFER_ENTRIES_CORE0 (1000)
```

5. *T1* Configuration File

This section contains a complete list of all parameters which can be placed in the T1 Configuration Invocation File (*T1_Cfg.inv*) including detailed information, examples, and cross-references. This file contains the fundamental settings of a T1 integration which are not expected to be changed after an integration has taken place.

5.1. Parameter -bigEndian

Definition `-bigEndian=<`*true/false*`>`

Short description Defines the endianness of the target.

Detailed description If this parameter is set to `true` it indicates that a big endian target is used (e.g. MPC5xxx). By default a little endian target is assumed. There is an endianness-check at startup of the target link. However when opening saved traces with a newly generated T1 project file then this parameter is the only source of information about the endianness of the target. It must be correctly configured for the correct interpretation of traces with cross-core task activation tracing enabled. Also it is mandatory to ensure that the endianness is correctly configured when `-endiannessAgnosticComSetup` is used.

Example The following example shows the usage of the parameter `-bigEndian`.

```
-bigEndian=true
; big endian target
[true/false(default)].
```

As a result the following entry is generated in the ***T1*** project file:

```
<TargetSpecifics BigEndian="true" />
```

See also `-endiannessAgnosticComSetup` (5.45)

5.2. Parameter -canBitrate

Definition `-canBitrate=<`*CAN bitrate*`>`

Short description Specifies the CAN bitrate for communication between host and target.

Detailed description The supported bitrates are: 100000, 250000, and 500000 for U2C. For Vector 250000, 500000, 800000, and 1000000 are supported.

Example The following example shows the usage of the parameter `-canBitrate`.

```
-canBitrate=500000 ; CAN bitrate
```

As a result the following entry is generated in the ***T1*** project file:

```
<Bitrate>500000</Bitrate>
<BitTiming>
 <Brp>1</Brp>
 <Prop>1</Prop>
 <Phase1>6</Phase1>
 <Phase2>4</Phase2>
 <Sjw>1</Sjw>
 <Sam>3</Sam>
</BitTiming>
```

5.3. Parameter -canExtendedIds

Definition `-canExtendedIds=`<*true/false*>

Short description Specifies usage of extended CAN IDs.

Detailed description If this parameter is set to `true` then the extended frame format is used and 29-bit identifiers are supported.

Example The following example shows the usage of the parameter `-canExtendedIds`.

```
-canExtendedIds=true ; use extended CAN IDs
[true/false(default)]
```

As a result the following entry is generated in the ***T1*** project file:

```
<isExtended>false</isExtended>
```

5.4. Parameter -canFDBitrate

Definition `-canFDBitrate=`<*CAN FD bitrate*>

Short description Specifies the CAN FD bitrate for communication between host and target.

Detailed description This parameter specifies the bitrate of the CAN FD arbitration phase. If a CAN FD frame is used, this parameter is mandatory. If a higher bitrate for the data phase shall be used then this can be set with the optional parameter `-canFDDataBitrate`.

Example The following example shows the usage of the parameter `-canFDBitrate`.

```
-canFDBitrate=500000 ; CAN FD bitrate
```

As a result the following entry is generated in the ***T1*** project file:

```
<Bitrate>500000</Bitrate>
```

See also `-canFDDataBitrate` (5.6)

5.5. Parameter -canFDBitTiming

Definition `-canFDBitTiming=`*<[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]>*

Short description Specify the CAN bit timing values, separated by commas.

Detailed description This optional parameter allows to specify the specific bit timing values of the CAN bus, only when using CAN FD. If this value is given alongside the parameter `-canFDSamplePoint`, only the sample point is used. These values need to be given in the following order <[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]> separated by commas.

Example The following example shows the usage of the parameter `-canFDBitTiming`

```
-canFDBitTiming=4,0,13,6,5,3
; CAN FD bit timing values separated with "," in the specific
; order "[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]".
```

As a result the following entry is generated in the ***T1*** project file:

```
<BitTiming>
  <Brp>4</Brp>
  <Prop>0</Prop>
  <Phase1>13</Phase1>
  <Phase2>6</Phase2>
  <Sjw>5</Sjw>
  <Sam>3</Sam>
</BitTiming>
```

See also `-canFDSamplePoint` (5.14)

5.6. Parameter -canFDDataBitrate

Definition `-canFDDataBitrate=`<*CAN FD data bitrate*>

Short description Specifies the CAN FD data bitrate for communication between host and target.

Detailed description This optional parameter defines the data bitrate for CAN FD. If omitted, the bitrate given with `-canFDBitrate` is used also for the data phase.

Example The following example shows the usage of the parameter `-canFDDataBitrate`.

```
-canFDDataBitrate=2000000 ; CAN FD data bitrate
```

As a result the following entry is generated in the ***T1*** project file:

```
<DataBitrate>2000000</DataBitrate>
```

See also `-canFDBitrate` (5.4)

5.7. Parameter -canFDDataBitTiming

Definition `-canFDDataBitTiming=`<*[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]*>

Short description Specifies the CAN FD data bit timing values, separated by commas.

Detailed description This optional parameter allows to specify the specific data bit timing values of the CAN FD frame. If this value is given alongside the parameter `-canFDDataSamplePoint`, only the sample point is used. These values need to be given in the following order <[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]> separated by commas.

Example The following example shows the usage of the parameter `-canFDDataBitTiming`

```
-canFDDataBitTiming=4,0,13,6,5,3
; CAN FD DataBitTiming values separated with "," in the specific
; order "[Brp], [Prop], [Phase1], [Phase2], [Sjw], [Sam]".
```

As a result the following entry is generated in the ***T1*** project file:

```
<DataBitTiming>
  <Brp>4</Brp>
  <Prop>0</Prop>
  <Phase1>13</Phase1>
  <Phase2>6</Phase2>
  <Sjw>5</Sjw>
  <Sam>3</Sam>
</DataBitTiming>
```

See also `-canFDDataSamplePoint` (5.8)

5.8. Parameter -canFDDataSamplePoint

Definition `-canFDDataSamplePoint=`<*data sample point in percent*>

Short description Specifies the CAN FD data sample point in percent.

Detailed description This optional parameter allows to specify the specific data sample point for the CAN FD. The default value used by the T1-HOST-SW is 70%. If this value is given alongside the parameter `-canFDDataBitTiming`, only the sample point is used. When a sample point is specified but could not be exactly matched, the nearest available value is used. This value should be set as close as possible to the configuration on the target to avoid communication issues.

Example The following example shows the usage of the parameter `-canFDDataSamplePoint`

```
-canFDDataSamplePoint=80
; CAN FD data sample point value.
```

As a result the following entry is generated in the ***T1*** project file:

```
<DataSamplePoint>80</DataSamplePoint>
```

See also `-canFDDataBitTiming` (5.7)

5.9. Parameter -canFDHardware

Definition `-canFDHardware=`<*VECTOR*>

Short description Selects the type of CAN FD interface hardware on the host.

Detailed description This parameter does not select a specific item but rather chooses the Vector device. The Vector Virtual Channels are chosen last as their usage is mostly unintended. If only Ethernet communication is used, this parameter must be removed or empty.

Example The following example shows the usage of the parameter `-canFDHardware`.

```
-canFDHardware=VECTOR ; [VECTOR]
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 <Name>...</Name>
 <HwType>VECTOR</HwType>
</HwIdentifier>
```

See also `-canFDHwName` (5.10)
`-canFDUsage` (5.16)

5.10. Parameter -canFDHwName

Definition `-canFDHwName=`*<Name of CAN FD HW>*

Short description Allows to specify the name of the CAN FD hardware.

Detailed description This optional parameter allows to specify the name of the CAN FD hardware which shall be used by the T1-HOST-SW. This parameter can be used for VECTOR hardware.

Example The following example shows the usage of the parameter `-canFDHwName`.

```
-canFDHwName=VectorBox
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 <Name>VectorBox</Name>
 <HwType>VECTOR</HwType>
</HwIdentifier>
```

See also `-canFDHardware` (5.9)
`-canFDUsage` (5.16)

5.11. Parameter -canFDMustUseFixedBlockSize

Definition `-canFDMustUseFixedBlockSize=`*<true/false>*

Short description Allows to fix the data length code (DLC) of CAN FD messages sent by the T1-HOST-SW.

Detailed description If this optional parameter is set to `true` then CAN FD messages sent by the T1-HOST-SW have a fixed DLC. The default DLC is 64 bytes. This value is changed if `-canFDMaxTxDataSize` is less than 64 bytes at project or system level.

Example The following example shows the usage of the parameter `-canFDMustUseFixedBlockSize`

```
-canFDMustUseFixedBlockSize=true
; T1-HOST-SW sends only messages with fixed DLC
[true/false(default)]
```

As a result the following entry is generated in the ***T1*** project file:

```
<Buses FixedBlocksize="true">
```

See also `-canFDMaxTxDataSize` (5.15)

5.12. Parameter -canFDOscillatorFrequencyMhz

Definition `-canFDOscillatorFrequencyMhz=`*<Oscillator frequency in MHz>*

Short description Specifies the oscillator frequency of the CAN FD device.

Detailed description This optional parameter allows to specify the specific oscillator frequency of the CAN FD device used. The default value used by the T1-HOST-SW is 80 MHz.

Example The following example shows the usage of the parameter `-canFDOscillatorFrequencyMhz`

```
-canFDOscillatorFrequencyMhz=80
; Oscillator frequency of CAN FD device
```

As a result the following entry is generated in the ***T1*** project file:

```
<OscillatorFrequencyMhz>80</OscillatorFrequencyMhz>
```

5.13. Parameter -canFDMaxRxDataSize

Definition `-canFDMaxRxDataSize=`*<Max. CAN FD message size (target to T1-HOST-SW)>*

Short description Specifies the maximum size of CAN FD messages sent by the target to T1-HOST-SW.

Detailed description This parameter has to be set if CAN FD communication is used per system or project-wide respectively.

Example The following example shows the usage of the parameter `-canFDMaxRxDataSize`.

```
-canFDMaxRxDataSize=32 ;
; Maximum CAN FD message size (target to T1-HOST-SW)
```

As a result the following entry is generated in the ***T1*** configuration header file if the parameter is project-wide

```
#define T1_GCP_MAX_TX_FRAME_SIZE 32
```

See also `-canFDMaxTxDataSize` (5.15)

5.14. Parameter -canFDSamplePoint

Definition `-canFDSamplePoint=`<*sample point in percent*>

Short description Specifies the CAN FD sample point in percent.

Detailed description This optional parameter allows to specify the specific sample point for CAN FD. The default value used by the T1-HOST-SW is 70%. If this value is given alongside the parameter `-canFDBitTiming`, only the sample point is used. When a sample point is specified but could not be exactly matched, the nearest available value is used. This value should be set as close as possible to the configuration on the target to avoid communication issues.

Example The following example shows the usage of the parameter `-canFDSamplePoint`

```
-canFDSamplePoint=70
; CAN FD sample point value.
```

As a result the following entry is generated in the ***T1*** project file:

```
<SamplePoint>70</SamplePoint>
```

See also `-canFDBitTiming` (5.5)

5.15. Parameter -canFDMaxTxDataSize

Definition `-canFDMaxTxDataSize=`<*Max. CAN FD message size (T1-HOST-SW to target)*>

Short description Specifies the maximum size of CAN FD messages sent by the T1-HOST-SW to the target.

Detailed description This parameter has to be set if CAN FD communication is used per system or project-wide respectively. The project-wide value or the minimum of the system values defines the maximum DLC of CAN FD messages sent by the T1-HOST-SW. Possible values are [8, 12, 16, 20, 24, 32, 48, 64], if other values are specified, they are rounded down to the next possible value. If the parameter `-canFDMustUseFixedBlockSize` is set, this DLC will be used instead of the default 64 bytes.

Example The following example shows the usage of the parameter `-canFDMaxTxDataSize`.

```
-canFDMaxTxDataSize=24 ;
; Maximum CAN FD message size (T1-HOST-SW to target)
```

As a result the following entries are generated in the ***T1*** project file

```
<Buses MaxTxBlocksize="24">
```

and in the ***T1*** configuration header file if the parameter is project-wide

```
#define T1_GCP_MAX_RX_FRAME_SIZE 24
```

See also `-canFDMustUseFixedBlockSize` (5.11)
`-canFDMaxRxDataSize` (5.13)

5.16. Parameter -canFDUsage

Definition `-canFDUsage=`<*Preferred*>

Short description Indicates CAN FD as the preferred device for the T1-HOST-SW.

Detailed description If this parameter is set to `Preferred` then the T1-HOST-SW will try to use the CAN FD device in priority. Only one preferred device should be set.

Example The following example shows the usage of the parameter `-canFDUsage`.

```
-canFDUsage=Preferred
; [Preferred].
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier Usage="Preferred">
```

See also -canFDHardware (5.9)
-canFDHwName (5.10)

5.17. Parameter -canHardware

Definition -canHardware=<*U2C/VECTOR*>

Short description Selects the type of CAN interface hardware on the host (U2C or VECTOR).

Detailed description This parameter does not select a specific item but rather chooses the first device in the list being either a U2C or Vector device. The Vector Virtual Channels are chosen last as their usage is mostly unintended. If only Ethernet communication is used, this parameter must be removed or empty.

Example The following example shows the usage of the parameter -canHardware.

```
-canHardware=U2C ; [U2C(default),VECTOR]
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 <Name>U2C_</Name>
 <HwType>U2C</HwType>
</HwIdentifier>
```

See also -canHwName (5.18)
-canUsage (5.21)

5.18. Parameter -canHwName

Definition -canHwName=<*Name of CAN HW*>

Short description Allows to specify the name of the CAN hardware.

Detailed description This optional parameter allows to specify the name of the CAN hardware which shall be used by the T1-HOST-SW. This parameter can be used for VECTOR or U2C type of CAN hardware.

Example The following example shows the usage of the parameter `-canHwName`.

```
-canHwName=U2C_A123ABCD
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 <Name>U2C_A123ABCD</Name>
 <HwType>U2C</HwType>
</HwIdentifier>
```

See also `-canHardware` (5.17)
`-canUsage` (5.21)

5.19. Parameter -canRxID

Definition `-canRxID=<` *CAN ID target to host*>

Short description Defines the CAN ID for messages from the target to the host.

Detailed description This parameter is used to configure CAN IDs used per system or project-wide respectively.

Example The following example shows the usage of the parameter `-canRxID`.

```
-canRxID=0x7FA ; CAN ID target to host
[default: 0x7FA]
```

As a result the following entries are generated in the ***T1*** project file

```
<Id>2042</Id>
```

and in the ***T1*** configuration header file

```
#define T1_CAN_ECU_TO_PC_ID (0x7FA)
```

5.20. Parameter -canTxID

Definition `-canTxID=<` *CAN ID host to target*>

Short description Defines the CAN ID for messages from the host to the target.

Detailed description This parameter is used to configure CAN IDs used per system or project-wide respectively.

Example The following example shows the usage of the parameter `-canTxID`.

```
-canTxID=0x7CB ; CAN ID host to target
[default: 0x7CB]
```

As a result the following entries are generated in the ***T1*** project file

```
<Id>1995</Id>
```

and in the ***T1*** configuration header file

```
#define T1_CAN_PC_TO_ECU_ID (0x7CB)
```

5.21. Parameter -canUsage

Definition `-canUsage=<`*Preferred*>

Short description Indicates CAN as the preferred device for the T1-HOST-SW.

Detailed description If this parameter is set to `Preferred` then the T1-HOST-SW will try to use the CAN device in priority. Only one preferred device should be set.

Example The following example shows the usage of the parameter `-canUsage`.

```
-canUsage=Preferred
; [Preferred].
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier Usage="Preferred">
```

See also `-canHardware` (5.17)
`-canHwName` (5.18)

5.22. Parameter -checkIntegration

Definition `-checkIntegration=`<*true/false*>

Short description Used by GLIWA engineers to test the T1-TARGET-SW integration.

Detailed description If this parameter is set to `true` additional macros and data structures will be generated which can be used for testing the T1-TARGET-SW.

Example The following example shows the usage of the parameter `-checkIntegration`.

```
-checkIntegration=true ; [true/false(default)].
```

As a result the following macros are generated in the *T1_config.h*:

```
/*!
 * \brief Indicates that the integration checking mechanism is
 * enabled using the invocation file parameter
 * [-checkIntegration](\ref checkIntegration),
 * see T1_CheckIntegrationHandlerPC()
 */
#define T1_CHECK_INTEGRATION (1)
```

There is also additional code generated into *T1_configGen.c*, which is not shown here due to its large size.

5.23. Parameter -commsCoreOffset

Definition `-commsCoreOffset=`<*T1 logical core ID*>

Short description Selects the core running T1 communications services.

Detailed description This optional parameter specifies the core on which T1_ReceiveFrame is called and on which the T1-TARGET-SW must call T1_TransmitFrame. The default value is zero.

Example The following example shows the usage of the parameter `-commsCoreOffset`.

```
-commsCoreOffset=1
```

As a result the following entry is generated in T1_configGen.c:

```
T1_uint8_t const T1_SEC_CONST_8 T1_commsCoreOffset = 1u;
```

5.24. Parameter -connectionType

Definition `-connectionType=<`*pure T1/Diagnosis*`>`

Short description Selects the communication type between target and host.

Detailed description This parameter is used to specify whether T1 shall make use of the UDS diagnostic service (`Diagnosis`) via the ISO-TP transport protocol or directly use GCP over CAN or Ethernet (`pure T1`).

Example The following example shows the usage of the parameter `-connectionType`.

```
-connectionType=pure T1
; [pure T1,Diagnosis]
```

As a result the following entry is generated in the ***T1*** project file:

```
<LayerConfig>
 <Name>pure T1</Name>
 ...
</LayerConfig>
```

5.25. Parameter -contRunsOnCore

Definition `-contRunsOnCore=<`*Core ID offset*`>`

Short description Defines the core that actually runs T1.cont for this system

Detailed description Setting this enables "remote core" T1.cont, whereby the trace for one core can be analyzed by an instance of T1.cont running on a different core. Note that this increases the T1.scope overhead for *all* cores. Specifically, defines the core that calls T1_ContBgHandler for the current system

Example The following entry might be found in the invocation file.

```
#system 1:
-contRunsOnCore=0 ; Core 0 calls T1_ContBgHandlerPC( 1 )
```

As a result the required macros, including the following line, are generated in *T1_config.h*.

```
#define T1_CONT_REMOTE (1)
```

5.26. Parameter -cortex

Definition `-cortex=`*<true/false>*

Short description Defines if the target is an Arm Cortex-M or Cortex-R core.

Detailed description This parameter must be set to `true` if the target is an Arm Cortex-M or Cortex-R core (for example: NXP LPC-series).

Example The following example shows the usage of the parameter `-cortex`.

```
-cortex=true ; ARM Cortex-M/-R
[true/false(default)]
```

As a result the following entry is generated in the ***T1*** project file:

```
<TargetSpecifics
 NeedsUniqueSuccessorAddress="true" />
```

See also `-targetType` (5.79)

5.27. Parameter -cortexMFPBversion

Definition `-cortexMFPBversion=`*<1/2>*

Short description Only for ARM Cortex-M, specify the Flash Patch breakpoint version for T1.flex.

Detailed description The ARM v7-M architecture profile features two different versions of the Flash Patch breakpoint architecture. Version 2 allows code breakpoints at any address but version 1 supports breakpoints only in address range 0x0..0x1FFFFFFF. If you specify `-cortexMFPBversion=1` and T1.flex is required to set a code breakpoint outside this range, it is assumed that the code is in RAM and the breakpoint is implemented as a soft breakpoint, where the code is temporarily overwritten with a BKPT instruction. The default value is 2, if this parameter is not set explicitly. This value is checked against the FP_CTRL.REV value when ***T1*** is initialized and T1.flex is disabled in the event of a mismatch.

Example The following example shows the usage of the parameter `-cortexMFPBversion`.

```
-cortexMFPBversion=2
; Only for ARM Cortex-M, specify Flash Patch breakpoint version
(defaults to 2)
```

As a result the following line is generated in *T1_configGen.c*.

```
T1_uint8_t const T1_SEC_CONST_8 T1_fpCrtlRev = 2u;
```

5.28. Parameter -cpuLoadAvgSamples

Definition -cpuLoadAvgSamples=<*number of CPU load samples*>

Short description This parameter determines if the CPU load is calculated only for one single Basic Scheduling Frame or if it is calculated as an average over all values that were collected during the time frame defined by -cpuLoadTxPeriod.

Detailed description This parameter can be set to 1 or it must be set alternatively to the same value as -cpuLoadTxPeriod. This parameter needs to be applied for each system (core) individually.

Example 1 The following example shows the usage of the parameter -cpuLoadAvgSamples.

```
-cpuLoadAvgSamples=10
; Number of samples within CPU load TX-period
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CPULOAD_SAMPLES_CORE0 (10)
```

Example 2 The following example shows the usage of the parameter -cpuLoadAvgSamples with a value greater than 255. The (sentinel) value zero of T1_CPULOAD_SAMPLES_COREx denotes that the number of samples is equal to the TX-period and is used for values that cannot be represented using 8 bits.

```
-cpuLoadAvgSamples=1000
; Number of samples within CPU load TX-period
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CPULOAD_SAMPLES_CORE0 (0) /* denotes 1000 */
```

See also -cpuLoadTxPeriod (5.29)

5.29. Parameter -cpuLoadTxPeriod

Definition -cpuLoadTxPeriod=<*CPU load transmit period: 1...65535*>

Short description This parameter allows to configure how often the CPU load data is sent to the T1-HOST-SW.

Detailed description The CPU load is transmitted every x Basic Scheduling Frames. For example if the Basic Scheduling Frame has a length of 100ms and `-cpuLoadTxPeriod` is set to 10, then the CPU load is transmitted every second. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-cpuLoadTxPeriod`.

```
-cpuLoadTxPeriod=10
; The CPU load is transmitted
every 10th BSF
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CPULOAD_TX_PERIOD_CORE0 (10)
```

See also `-cpuLoadAvgSamples` (5.28)

5.30. Parameter -diagAddressingMode

Definition `-diagAddressingMode=`*<normal/extended>*

Short description Specifies the addressing mode for the diagnostic interface.

Detailed description Two addressing modes of ISO-TP are supported by ***T1***: Normal or extended whereby the extended addressing provides additional addressing capabilities by holding an additional addressing byte in the first payload byte. The extended mode is required if there is no unique CAN ID for diagnostic services for each target in the system.

Example The following example shows the usage of the parameter `-diagAddressingMode`.

```
-diagAddressingMode=extended
; addressing mode for diagnostic interface
[normal/extended]
```

As a result the following entry is generated in the ***T1*** project file:

```
<AddressingMode>extended</AddressingMode>
```

5.31. Parameter -diagCustomSessionId

Definition `-diagCustomSessionId=`<*Custom Session ID*>

Short description Specifies the concrete session type to be opened, if the parameter `-diagOpenSessionType` is set to 'Custom'.

Detailed description If the parameter `-diagOpenSessionType` is set to 'Custom', ***T1*** opens a diagnostic session using the provided ID value as session type. The value must be between 0x00 and 0x7F. Please consult the ISO document 14229 (keyword DiagnosticSessionControl) for detailed information about the meaning of each individual number. Please note, that ***T1*** will not check the validity of the chosen session type.

Example The following example shows the usage of the parameter `-diagCustomSessionId`.

```
-diagCustomSessionId=0x48
```

As a result the following entry is generated in the ***T1*** project file:

```
<CustomSessionId>72</CustomSessionId>
```

See also `-diagOpenSessionType` (5.34)

5.32. Parameter -diagLocalIdentifier

Definition `-diagLocalIdentifier=`<*Diagnosis: Local identifier*>

Short description Specifies local identifier for the diagnostic interface (deprecated: only supported by KWP2000).

Detailed description This parameter is similar to `-diagT1Identifier` but uses local ID configuration.

Example The following example shows the usage of the parameter `-diagLocalIdentifier`.

```
-diagLocalIdentifier=0xFA
; diagnosis: Local Identifier.
```

As a result the following entry is generated in the ***T1*** project file:

```
<T1LocalIdentifier>250</T1LocalIdentifier>
```

5.33. Parameter -diagMinTxDataSize

Definition `-diagMinTxDataSize=`<*Diagnosis: Minimum Transmit data size*>

Short description Specifies minimum TX data size of one frame for the diagnostic interface (using WriteByIdentifier service).

Detailed description Using this parameter - WriteDataByIdentifier Service. This service is used to send commands to the target. The minimum number of payload bytes (UDS-overhead already subtracted) being sent by the T1-HOST-SW are to be declared.

Example The following example shows the usage of the parameter `-diagMinTxDataSize`.

```
-diagMinTxDataSize=4
; diagnosis: min. number of bytes for
WriteByIdentifierservice (4..1023)
```

As a result the following entries are generated in the ***T1*** project file

```
<minTxDataSize>4</minTxDataSize>
```

and in the ***T1*** configuration header file

```
#define T1_GCP_MIN_RX_FRAME_SIZE (4)
```

5.34. Parameter -diagOpenSessionType

Definition `-diagOpenSessionType=`<*None/Default/Programming/Extended/Standby/Custom*>

Short description Specifies the session type to be opened when the target link is enabled.

Detailed description This parameter is used to automatically send a DiagnosticSessionControl request for the specified diagnostic session type, when the target link is enabled. The following diagnostic session types are supported:

- Default - requests a default diagnostic session. A default session does usually not need the TesterPresent service to keep the session alive.
- Programming - requests a programming diagnostic session. That usually enables all diagnostic services required for memory programming.
- Extended - requests an extended diagnostic session. That usually enables necessary diagnostic services for adjusting functions. It is also used to enable diagnostic services that are related but not tied to adjustment of functions.

- Standby - requests a stand by session. Usually used for 'slave' ECUs that are not to be manually controlled. ECUs operating on an activated stand by session shall use a constant power consumption.
- Custom - ***T1*** will try to open a diagnostic session using the value of the parameter `-diagCustomSessionId` as session type. In this case the parameter `-diagCustomSessionId` must be specified.

Example The following example shows the usage of the parameter `-diagOpenSessionType`.

```
-diagOpenSessionType=Default
```

As a result the following entry is generated in the ***T1*** project file:

```
<OpenSession>Default</OpenSession>
```

See also `-diagCustomSessionId` (5.31)

5.35. Parameter -diagMaxRxDataSize

Definition `-diagMaxRxDataSize=`<*Diagnosis: Receive data size*>

Short description Specifies maximum RX data size of one frame the diagnostic interface (using ReadByIdentifier service).

Detailed description Using this parameter - ReadDataByIdentifierService: This service is used to retrieve data from the target. Only the maximum number of payload bytes (UDS-overhead already subtracted) being sent by the target is to be declared to the T1-HOST-SW.

Example The following example shows the usage of the parameter `-diagMaxRxDataSize`.

```
-diagMaxRxDataSize=8
; diagnosis: max. number of bytes for
ReadByIdentifier service (4..1023).
```

As a result the following entries are generated in the ***T1*** project file

```
<RxDataSize>8</RxDataSize>
```

and in the ***T1*** configuration header file

```
#define T1_GCP_MAX_TX_FRAME_SIZE (8)
```

5.36. Parameter -diagSourceAddr

Definition `-diagSourceAddr=`<*Diagnosis source address*>

Short description Specifies source address for the diagnostic interface. It is typically the lowest byte of `-canRxID` e.g. 0xFA.

Detailed description This parameter must be used in extended addressing mode. It is used for communication from host to target.

Example The following example shows the usage of the parameter `-diagSourceAddr`.

```
-diagSourceAddr=0xFA
; source address for diagnostic interface
(host to target).
```

As a result the following entry is generated in the ***T1*** project file:

```
<SouceAddress>250</SouceAddress>
```

5.37. Parameter -diagT1Identifier

Definition `-diagT1Identifier=`<*Diagnosis: T1 identifier*>

Short description Specifies T1 identifier for Read and WriteByIdentifier services for the diagnostic interface.

Detailed description T1 makes use of the ReadByIdentifier (0x22) and WriteByIdentifier (0x2E). The T1-HOST-SW requires to have the same ID configured for both services. This ID is configured via this parameter.

Example The following example shows the usage of the parameter `-diagT1Identifier`.

```
-diagT1Identifier=0xFA
; diagnosis: T1 identifier for
Read and WriteByIdentifier services.
```

As a result the following entry is generated in the ***T1*** project file:

```
<T1Identifier>250</T1Identifier>
```

5.38. Parameter -diagTargetAddr

Definition `-diagTargetAddr=`<*Diagnosis target address*>

Short description Specifies target address for the diagnostic interface. It is typically the lowest byte of `-canTxID` e.g. 0xCB

Detailed description This parameter must be used in extended addressing mode. It is used for communication from target to host.

Example The following example shows the usage of the parameter `-diagTargetAddr`.

```
-diagTargetAddr=0xCB
; target address for diagnostic interface
(target to host).
```

As a result the following entry is generated in the ***T1*** project file:

```
<TargetAddress>203</TargetAddress>
```

5.39. Parameter -diagTesterPresentPeriod

Definition `-diagTesterPresentPeriod=`<*Period in ms*>

Short description Period with which the 'TesterPresent' message is sent.

Detailed description ***T1*** will periodically sent a 'TesterPresent' message while the target link is enabled. This is sometimes necessary to keep the diagnostic session alive. ***T1*** expects a positive response on the 'TesterPresent' message.

Example The following example shows the usage of the parameter `-diagTesterPresentPeriod`.

```
-diagTesterPresentPeriod=2000
```

As a result the following entry is generated in the ***T1*** project file:

```
<TesterPresentPeriod>2000</TesterPresentPeriod>
```

5.40. Parameter -diagMaxTxDataSize

Definition `-diagMaxTxDataSize=`<*Diagnosis: Transmit data size*>

Short description Specifies maximum TX data size of one frame the diagnostic interface (using WriteByIdentifier service).

Detailed description Using this parameter - WriteDataByIdentifier Service. This service is used to send commands to the target. The maximum number of payload bytes (UDS-overhead already subtracted) being sent by the T1-HOST-SW are to be declared.

Example The following example shows the usage of the parameter `-diagMaxTxDataSize`.

```
-diagMaxTxDataSize=32 ;
; diagnosis: max. number of bytes for
WriteByIdentifier service (4..1023).
```

As a result the following entries are generated in the ***T1*** project file

```
<TxDataSize>32</TxDataSize>
```

and in the ***T1*** configuration header file

```
#define T1_GCP_MAX_RX_FRAME_SIZE (32)
```

5.41. Parameter -diagUseServiceByLocalId

Definition `-diagUseServiceByLocalId=`<*true/false*>

Short description Specifies to use service by local ID for the diagnostic interface (deprecated: only supported by KWP2000).

Detailed description Using this parameter in extended mode the diagnosis usage service by local ID can be configured.

Example The following example shows the usage of the parameter `-diagUseServiceByLocalId`.

```
-diagUseServiceByLocalId=false
; Diagnosis service by local ID.
```

As a result the following entry is generated in the ***T1*** project file:

```
<UseServicesByLocId>false</UseServicesByLocId>
```

5.42. Parameter -dpcAlways

Definition `-dpcAlways=`<*symbolName*>

Short description Specifies that T1.flex shall Disable Pre-emption for CET (SWF) measurements of the named symbol.

Detailed description Very short functions can have their CET measured much more accurately if T1.flex disables pre-emption, from interrupts and other tasks, for the duration of the SWF measurement. For longer functions, care must be taken that the impact on the system scheduling is not too great. If <*symbolName*> does not already begin with <*prefix*>, <*prefix*> is prepended to the symbol name, see Section 5.73.

Example The following example shows the usage of the parameter `-dpcAlways`.

```
-dpcAlways=T1_ReceiveFrame
; Disable pre-emption during SWF on T1_ReceiveFrame
```

As a result the following entry is generated in the ***T1*** project file:

```
<SymbolExtension Name="T1_ReceiveFrame" DpcEligible="Preferred" />
```

See also `-dpcNever` (5.43)
`-symbolPrefix` (5.73)

5.43. Parameter -dpcNever

Definition `-dpcNever=`<*symbolName*>

Short description Specifies that T1.flex shall not Disable Pre-emption for CET (SWF) measurements of the named symbol.

Detailed description Very short functions can have their CET measured much more accurately if T1.flex disables pre-emption, from interrupts and other tasks, for the duration of the SWF measurement. For longer functions, care must be taken that the impact on the system scheduling is not too great. If <*symbolName*> does not already begin with <*prefix*>, <*prefix*> is prepended to the symbol name, see Section 5.73.

Example The following example shows the usage of the parameter `-dpcNever`.

```
-dpcNever=T1_HandlerPC
; Do not disable pre-emption during SWF on T1_HandlerPC
```

As a result the following entry is generated in the ***T1*** project file:

```
<SymbolExtension Name="T1_HandlerPC" DpcEligible="Never" />
```

See also `-dpcAlways` (5.42)
`-symbolPrefix` (5.73)

5.44. Parameter -enableStreaming

Definition `-enableStreaming=`<*true/false*>

Short description Enables the user interface features of the T1-HOST-SW that relate to streaming. This is typically set when the communication layer is known to support the bandwidth required for streaming, such as with Ethernet. Equally, it can be disabled when the bandwidth is insufficient and any streaming would be very interrupted with drop-outs. This parameter must be omitted if the generated T1 project file shall be used with T1-HOST-SW V2.

Example The following example shows the usage of the parameter `-enableStreaming`.

```
-enableStreaming=true
; Enable T1-HOST-SW streaming features
```

As a result the following entry is generated in the ***T1*** project file:

```
<Project ... EnableStreamingUI="true">
```

5.45. Parameter -endiannessAgnosticComSetup

Definition `-endiannessAgnosticComSetup=`<*true*>

Short description Enables compatibility of T1-HOST-SW with certain versions of the T1-TARGET-SW on big endian targets.

Note: This parameter is deprecated, please see Section A.6.1 for information on earlier versions.

5.46. Parameter -ethEcuDNSHostName

Definition `-ethEcuDNSHostName=`<*DNS Host Name*>

Short description DNS host name of the target.

Detailed description This optional parameter is used to configure the target DNS host name per system or project-wide respectively. Alternatively the IP address of the target can be set with `-ethEcuIP`.

Example The following example shows the usage of the parameter `-ethEcuDNSHostName`.

```
-ethEcuDNSHostName=target-host-name
; Ethernet target host name.
```

As a result the following entries are generated in the ***T1*** project file

```
<EcuDnsHostName>target-host-name</EcuDnsHostName>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_ECU_DNS_HOST_NAME target-host-name
```

See also `-ethEcuIP` (5.47)
`-ethHardware` (5.49)
`-ethUseIpv6` (5.57)

5.47. Parameter -ethEcuIP

Definition `-ethEcuIP=`<*IP Address*>

Short description IP address of the target.

Detailed description This optional parameter is used to configure the target IP address used per system or project-wide respectively. Alternatively the DNS host name can be set with `-ethEcuDNSHostName`.

Example (IPv4) The following example shows the usage of the parameter `-ethEcuIP` with IPv4.

```
-ethEcuIP=192.168.178.29
; [IPv4] Ethernet target IP address.
```

As a result the following entries are generated in the ***T1*** project file

```
<IpAddress>192.168.178.29</IpAddress>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_ECU_IP0 192
#define T1_ETH_ECU_IP1 168
#define T1_ETH_ECU_IP2 178
#define T1_ETH_ECU_IP3 29
```

Example (IPv6) The following example shows the usage of the parameter `-ethEcuIP` with IPv6.

```
-ethUseIpv6=true
-ethEcuIP=fd00::e989:3ef2:bd6d:b57d
```

As a result the following entries are generated in the ***T1*** project file

```
<IpAddress>fd00::e989:3ef2:bd6d:b57d</IpAddress>
...
<BusRef>
  <Name>Ethernet</Name>
  <BusType>ETHERNET_V6</BusType>
</BusRef>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_USE_IPV6 (1)
...
/* ECU IP address [fd00::e989:3ef2:bd6d:b57d]:40060 */
#define T1_ETH_ECU_IPV6 fd00::e989:3ef2:bd6d:b57d
#define T1_ETH_ECU_IPV6_HW0 fd00
#define T1_ETH_ECU_IPV6_HW1 0000
#define T1_ETH_ECU_IPV6_HW2 0000
#define T1_ETH_ECU_IPV6_HW3 0000
#define T1_ETH_ECU_IPV6_HW4 e989
#define T1_ETH_ECU_IPV6_HW5 3ef2
#define T1_ETH_ECU_IPV6_HW6 bd6d
#define T1_ETH_ECU_IPV6_HW7 b57d
```

See also `-ethEcuDNSHostName` (5.46)
`-ethHardware` (5.49)
`-ethUseIpv6` (5.57)

5.48. Parameter -ethEcuPort

Definition `-ethEcuPort=`<*Port*>

Short description Specifies the port on the target used for Ethernet communication.

Detailed description This parameter has to be set if Ethernet communication is used per system or project-wide respectively.

Example The following example shows the usage of the parameter `-ethEcuPort`.

```
-ethEcuPort=8100 ;
; Ethernet port of the target
```

As a result the following entries are generated in the ***T1*** project file

```
<EcuPort>8100</EcuPort>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_ECU_PORT 8100
```

See also `-ethHardware` (5.49)

5.49. Parameter -ethHardware

Definition `-ethHardware=`<*LAN_ADAPTER*>

Short description Selects the hardware on the host for Ethernet.

Detailed description If this parameter is set to `LAN_ADAPTER` then the Ethernet device will be enabled.

Example The following example shows the usage of the parameter `-ethHardware`.

```
-ethHardware=LAN_ADAPTER
; [LAN_ADAPTER]
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 ...
 <HwType>LAN_ADAPTER</HwType>
</HwIdentifier>
```

See also `-ethHwName` (5.50)
`-ethUsage` (5.56)

5.50. Parameter -ethHwName

Definition `-ethHwName=`<*Name of Ethernet HW*>

Short description Allows to specify the name of the Ethernet hardware.

Detailed description This optional parameter allows to specify the name of the Ethernet hardware which shall be used by the T1-HOST-SW. Alternatively the PC IP address can be set with `-ethPcIP`. If both are specified then the PC IP address is considered first by the T1-HOST-SW.

Example The following example shows the usage of the parameter `-ethHwName`.

```
-ethHwName=Ethernet
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier>
 <Name>Ethernet</Name>
 ...
</HwIdentifier>
```

See also `-ethPcIP` (5.51)
`-ethHardware` (5.49)
`-ethUsage` (5.56)

5.51. Parameter -ethPcIP

Definition `-ethPcIP=`<*IP Address*>

Short description IP address of the computer running the T1-HOST-SW.

Detailed description This optional parameter is used to configure PC IP address used for Ethernet communication. Alternatively the name of the Ethernet hardware can be set with `-ethHwName`. If both are specified then the PC IP address is considered first by the T1-HOST-SW.

Example (IPv4) The following example shows the usage of the parameter `-ethPcIP` with IPv4.

```
-ethPcIP=192.168.178.21 ; [IP] Ethernet PC IP address.
```

As a result the following entries are generated in the ***T1*** project file

```
<PcIpAddress>192.168.178.21</PcIpAddress>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_PC_IP0 192
#define T1_ETH_PC_IP1 168
#define T1_ETH_PC_IP2 178
#define T1_ETH_PC_IP3 21
```

Example (IPv6) The following example shows the usage of the parameter `-ethPcIP` with IPv6.

```
-ethUseIpv6=true ; Use IPv6
-ethPcIP=fe80::71dc:90:9d73:71ad ; [IP] Ethernet PC IP address.
```

As a result the following entries are generated in the ***T1*** project file

```
<Buses FixedBlocksize="false">
  <Name>Ethernet</Name>
  <Type>ETHERNET_V6</Type>
  <HwIdentifier Usage="Preferred">
    <Name>Ethernet</Name>
    <PcIpAddress>fe80::71dc:90:9d73:71ad</PcIpAddress>
    <HwType>LAN_ADAPTER</HwType>
  </HwIdentifier>
  <TxCycleMs>10</TxCycleMs>
</Buses>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_USE_IPV6 (1)
/* PC IP address [fe80::71dc:90:9d73:71ad]:50011 */
#define T1_ETH_PC_IPV6 fe80::71dc:90:9d73:71ad
#define T1_ETH_PC_IPV6_HW0 fe80
#define T1_ETH_PC_IPV6_HW1 0000
#define T1_ETH_PC_IPV6_HW2 0000
#define T1_ETH_PC_IPV6_HW3 0000
#define T1_ETH_PC_IPV6_HW4 71dc
#define T1_ETH_PC_IPV6_HW5 0090
#define T1_ETH_PC_IPV6_HW6 9d73
#define T1_ETH_PC_IPV6_HW7 71ad
```

See also `-ethHwName` (5.50)
`-ethHardware` (5.49)
`-ethUseIpv6` (5.57)

5.52. Parameter -ethPcPort

Definition `-ethPcPort=`<*Port*>

Short description Specifies the port on the PC used for Ethernet communication.

Detailed description This parameter has to be set if Ethernet communication is used.

Example The following example shows the usage of the parameter `-ethPcPort`.

```
-ethPcPort=8101 ;
; Ethernet port of the PC
```

As a result the following entries are generated in the ***T1*** project file

```
<PcPort>8101</PcPort>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_PC_PORT 8101
```

See also `-ethHardware` (5.49)

5.53. Parameter -ethMaxRxDataSize

Definition `-ethMaxRxDataSize=`*<Max. Ethernet messages size>*

Short description Specifies the maximum size of Ethernet messages sent by the target.

Detailed description This parameter has to be set if Ethernet communication is used

Example The following example shows the usage of the parameter `-ethMaxRxDataSize`.

```
-ethMaxRxDataSize=200;
```

As a result the following line is generated in the ***T1*** configuration header file.

```
#define T1_GCP_MAX_TX_FRAME_SIZE 200
```

See also `-ethHardware` (5.49)
`-ethMaxTxDataSize` (5.55)

5.54. Parameter -ethTxCycleMs

Definition `-ethTxCycleMs=`*<Ethernet TX cycle of T1-HOST-SW>*

Short description Specifies Ethernet TX cycle of T1-HOST-SW in milliseconds.

Detailed description For the correct choice of the T1-HOST-SW's TX-cycle, see 5.91. This parameter concerns Ethernet communication.

Example The following example shows the usage of the parameter `-ethTxCycleMs`.

```
-ethTxCycleMs=20
; Ethernet TX cycle of T1-HOST-SW in ms
```

As a result the following entry is generated in the ***T1*** project file:

```
<TxCycleMs>20</TxCycleMs>
```

See also `-ethHardware` (5.49)

5.55. Parameter -ethMaxTxDataSize

Definition `-ethMaxTxDataSize=`<*Max. Ethernet messages size*>

Short description Specifies the maximum size of Ethernet messages sent by the T1-HOST-SW.

Detailed description This parameter has to be set if Ethernet communication is used per system or project-wide respectively.

Example The following example shows the usage of the parameter `-ethMaxTxDataSize`.

```
-ethMaxTxDataSize=24 ;
; Maximum Ethernet message size (T1-HOST-SW to target)
```

As a result the following entries are generated in the ***T1*** project file

```
<TxDataSize>24</TxDataSize>
```

and in the ***T1*** configuration header file

```
#define T1_GCP_MAX_RX_FRAME_SIZE 24
```

See also `-ethHardware` (5.49)
`-ethMaxRxDataSize` (5.53)

5.56. Parameter -ethUsage

Definition `-ethUsage=`<*Preferred*>

Short description Indicate Ethernet as the preferred device for the T1-HOST-SW.

Detailed description If this parameter is set to `Preferred` then the T1-HOST-SW will try to use the Ethernet device in priority. Only one preferred device should be set.

Example The following example shows the usage of the parameter `-ethUsage`.

```
-ethUsage=Preferred
; [Preferred].
```

As a result the following entry is generated in the ***T1*** project file:

```
<HwIdentifier Usage="Preferred">
```

See also `-ethHardware` (5.49)
`-ethHwName` (5.50)

5.57. Parameter -ethUseIpv6

Definition `-ethUseIpv6=`<*true/false*>

Short description Select Internet Protocol (IP) version being used.

Detailed description The IP version is set to IPv4 by default (`-ethUseIpv6=false`). If this parameter is set to `true` then IPv6 is used. This parameter is only relevant for the T1-HOST-SW it is not used by the T1-TARGET-SW itself, but the configuration is reflected in the generated header file (T1_config.h) for use in the target's application code.

Example The following example shows the usage of the parameter `-ethUseIpv6`.

```
-ethUseIpv6=true
; [true, false] if true, IPv6 is used, if false, IPv4 is used.
```

As a result the following entries are generated in the ***T1*** project file

```
<BusType>ETHERNET_V6</BusType>
```

and

```
<Type>ETHERNET_V6</Type>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_USE_IPV6 (1)
```

See also `-ethHardware` (5.49)
`-ethEcuIP` (5.47)
`-ethPcIP` (5.51)

5.58. Parameter -ethUseUdp

Definition `-ethUseUdp=`<*true/false*>

Short description Select Ethernet protocol being used.

Detailed description The Ethernet protocol is set to UDP by default. If this parameter is set to `false` then the TCP protocol is used.

Example The following example shows the usage of the parameter `-ethUseUdp`.

```
-ethUseUdp=true ;
; [true, false] if false, TCP used.
```

As a result the following entries are generated in the ***T1*** project file

```
<UseUdp>true</UseUdp>
```

and in the ***T1*** configuration header file

```
#define T1_ETH_USE_UDP (1)
```

See also `-ethHardware` (5.49)

5.59. Parameter -maxPreemptionDepth

Definition `-maxPreemptionDepth=`<*pre-emption depth*>

Short description Overrides the default size of the T1.cont task stack

Detailed description Unless this is set, the default T1.cont task stack size is the larger of 20 or 4 + <number of tasks>/5.

Example The following entry might be found in the invocation file.

```
-maxPreemptionDepth=20 ; At most 20 levels of task and ISR
                       ; pre-emption
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_MAX_TASK_PREEMPTIONS_CORE0 (20)
```

5.60. Parameter -mustUseFixedBlockSize

Definition `-mustUseFixedBlockSize=`<*true/false*>

Short description Allows to fix the data length of CAN messages (DLC) sent by the T1-HOST-SW to 8 bytes.

Detailed description If this parameter is set to `true` then CAN messages sent by the T1-HOST-SW have a fixed data length code (DLC) of 8. While the pure T1 protocol and the ISO-TP protocols support to have a variable CAN data length it might be the case that some CAN configuration of the target only allow reception of messages with a fixed DLC of 8 bytes.

Example The following example shows the usage of the parameter `-mustUseFixedBlockSize`

```
-mustUseFixedBlockSize=true
; T1-HOST-SW sends only messages with DLC=8
[true/false(default)]
```

As a result the following entry is generated in the ***T1*** project file:

```
<Buses FixedBlocksize="true">
```

5.61. Parameter -numberOfFlexAddrs

Definition -numberOfFlexAddrs=<*Number of T1.flex addresses*>

Short description Defines size of T1.flex address buffer on the target.

Detailed description This parameter is used to specify number of T1.flex addresses which can be stored on the target. It must be at least equal to -numberOfFlexStopwatches. For SWC measurements two address entries on the target are required per configured measurement range. This address buffer also stores the address results of the NCA, PCA, and DCA measurements, therefore the maximum number of results is defined with this parameter. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter -numberOfFlexAddrs.

```
-numberOfFlexAddrs=16 ; [0..255]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_NOF_FLEX_ADDRS_CORE0 (16)
```

See also -numberOfFlexStopwatches (5.62)

5.62. Parameter -numberOfFlexStopwatches

Definition -numberOfFlexStopwatches=<*Number of T1.flex stopwatches*>

Short description This parameter is used to specify number of used T1.flex stopwatches.

Detailed description They are used for example by the feature 'linked SWF'. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter -numberOfFlexStopwatches.

```
-numberOfFlexStopwatches=16 ; [0..255]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_NOF_FLEX_STPWS_CORE0 (16)
```

See also `-numberOfFlexAddrs` (5.61)
`-numberOfUserStpws` (4.34)

5.63. Parameter -numberOfFocusMeasurements

Definition `-numberOfFocusMeasurements=`<*Number of focus measurements*>

Short description Defines the number of focus measurements configured in the T1-TARGET-SW.

Detailed description This optional parameter is used to specify the number of focus measurements available in parallel on each system. The focus measurement is used to retrieve additional timing information from T1.cont. 32 bytes of RAM are being used per focus measurement. It is possible to configure two focus measurements. See Section A.7.2 for information on earlier versions.

Example The following example shows the usage of the parameter `-numberOfFocusMeasurements`.

```
-numberOfFocusMeasurements=2 ; [1..2]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CONT_NOF_FOCUS_MEASUREMENTS (2u)
```

5.64. Parameter -pSyncTimer

Definition `-pSyncTimer=`<*address of synchronization timer*>

Short description Defines the address of the memory-mapped synchronization timer

Detailed description This parameter is required in multi-core projects if one or more systems have `-traceTimerIsSyncTimer=false`. This means a different timer is used as synchronization timer than the timers used as trace timers, for those cores. See Section A.1.5 for information on earlier versions.

Example The following example shows the usage of the parameter `-pSyncTimer`.

```
-pSyncTimer=0xFF811014
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_GET_SYNC_TIME( ) (*(T1_uint32_t volatile *)0xFF811014)
```

See also

-syncTimerTickDurationNs (5.77)
-syncTimeBitLength (5.75)
-syncTimerDownCounting (5.76)
-traceTimerIsSyncTimer (5.89)

5.65. Parameter -pTimer

Definition -pTimer=<*address of timer to be used*>

Short description Defines the address of the memory-mapped trace timer

Detailed description ***T1*** reads trace time-stamps from a memory-mapped timer, whose address is specified by -pTimer. For multi-core projects, this parameter has to specified for each core, if they will use the memory-mapped timer as the trace timer. This parameter relates to the core previously specified by -sid=<core ID>.

T1 also provides direct support for using a non-memory-mapped timer. If a non-memory mapped timer is used, -pTimer is only regarded if it is the same for every core. Otherwise, the integrator must define the macro T1_GET_RAW_TIME to return the trace time-stamp. See section A.1.6 for information on earlier versions.

Example The following entry might be found in an invocation file for the Perl script. Here, the processor is an Infineon AURIX.

```
-pTimer=0xF0000014 ; address of memory-mapped timer STM0_TMR1
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_PTIMER_CORE0 (0xF0000014)
```

See also

-traceTimerDownCounting (5.88)
-tickDurationNs (5.83)

5.66. Parameter -rawTickDurationNs

Definition `-rawTickDurationNs=`<*Raw (unscaled) Timer Register Tick Duration*>

Short description Defines the physical timer register tick duration in nanoseconds.

Detailed description This parameter is used to define the physical timer register tick duration in nanoseconds for each system. Using this parameter, the raw tick duration of the trace timer is provided to the T1-TARGET-SW. For this parameter also fractions are allowed as input values. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-rawTickDurationNs`.

```
-rawTickDurationNs=1000 / 64
; Tick duration for a 64 MHz timer
```

As a result the following entries are generated in the ***T1*** configuration header file, using Core 0 in the example

```
#define T1_RAW_TO_TRACE_TIME_CORE0( rawTime_ )
#define T1_RAW_TO_SYNC_TIME_CORE0( rawTime_ )
#define T1_GET_TRACE_TIME_PC( 0 ) T1_RAW_TO_TRACE_TIME_CORE0( T1_GET_RAW_TIME_CORE0( )[2] )
```

See also `-tickDurationNs` (5.83)
`-syncTimerTickDurationNs` (5.77)

5.67. Parameter -rawTimerBitLength

Definition `-rawTimerBitLength=`<*Number of counting bits in raw timer*>

Short description Defines the number of counting bits in the raw (unscaled) physical trace timer register.

Detailed description Defines the number of counting bits in the raw (unscaled) physical timer register, which will be used as the trace timer. This parameter needs to be applied for each system (core) individually. The Perl scripts use this parameter and the ratio of the tick durations in order to calculate the number of counting bits to be used in the trace timer. Note that if the synchronization timer will be used as the trace timer, then -rawTimerBitLength must be equal to -syncTimeBitLength. See Section A.1.3 for information on earlier versions.

[2] would be T1_GET_SYNC_TIME( )if the synchronization timer is used as the trace timer for that core.

Example The following example shows the usage of the parameter `-rawTimerBitLength`.

```
-rawTimerBitLength=31
; 32-bit CPU with a smaller timer
(example 31 bit) [16..32(default)]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_TRACE_TIMER_BIT_LENGTH_CORE0 (31)
```

See also `-rawTickDurationNs` (5.66)
`-tickDurationNs` (5.83)
`-syncTimeBitLength` (5.75)

5.68. Parameter -retainIgnoreCase

Definition `-retainIgnoreCase=`*true/false*

Short description If true, the "white list" created by `-retainSymbol` is case insensitive

Detailed description The T1-HOST-SW imports symbols from the supplied ELF file(s). Filters suppress the import of symbols that do not resemble the functions and variables that ***T1*** expects to measure. When a symbol must be imported in spite of filtering, use the `-retainSymbol` parameter, see Section 5.70.

Example The following entry might be found in an invocation file for the Perl script.

```
-retainIgnoreCase=false ; white list is case sensitive
```

As a result the following entry is generated in the ***T1*** project file:

```
<SpecialSymbolsInclusion ... RegExIgnoreCase="false">
```

See also `-retainMax` (5.69)
`-retainSymbol` (5.70)

5.69. Parameter -retainMax

Definition `-retainMax=`<*maximum number of symbols*>

Short description Defines maximum number of symbols to include via the "white list" created by `-retainSymbol`

Detailed description The T1-HOST-SW imports symbols from the supplied ELF file(s). Filters suppress the import of symbols that do not resemble the functions and variables that ***T1*** expects to measure. When a symbol must be imported in spite of filtering, use the `-retainSymbol` parameter, see Section 5.70.

Example The following entry might be found in an invocation file for the Perl script.

```
-retainMax=100 ; maximum number of symbols to pass filter
```

As a result the following entry is generated in the ***T1*** project file:

```
<SpecialSymbolsInclusion MaxInclusionCount="100" ...
```

See also `-retainSymbol` (5.70)
`-retainIgnoreCase` (5.68)

5.70. Parameter -retainSymbol

Definition `-retainSymbol=`<*symbolName*>

Short description Add <*symbolName*> to a "white list" of symbols to be imported regardless of symbol type.

Detailed description The T1-HOST-SW imports symbols from the supplied ELF file(s). Filters suppress the import of symbols that do not resemble the functions and variables that ***T1*** expects to measure. When a symbol must be imported in spite of filtering, use the `-retainSymbol` parameter.

Example The following entry might be found in an invocation file for the Perl script.

```
-retainSymbol=VECT_TAB_C0 ; import VECT_TAB_C0 regardless of filter
```

As a result the following entry is generated in the ***T1*** project file:

```
<RegExPattern>VECT_TAB_C0</RegExPattern>
```

See also `-retainIgnoreCase` (5.68)
`-retainMax` (5.69)

5.71. Parameter -rxChannel

Definition `-rxChannel=`<*CAN/CAN_FD/ETHERNET*>

Short description Selects the bus type of the RxChannel.

Detailed description This parameter specifies whether CAN, CAN FD or Ethernet is used as bus type for the RxChannel.

Example The following example shows the usage of the parameter `-rxChannel`.

```
-rxChannel=CAN
; [CAN/CAN_FD/ETHERNET]
```

As a result the following entry is generated in the ***T1*** project file:

```
<BusType>CAN</BusType>
```

5.72. Parameter -sid

Definition `-sid=`<*System ID*>

Short description Specifies the system ID for each core.

Detailed description The system ID (SID) is used to identify each core. The SID needs to be assigned linearly without gaps for each consecutive core. Typically the SID starts with the value of 2 for the first core. The *TargetId* is derived from the SID and cannot be zero ($TargetId = SID \bmod 32 \neq 0$).

Example The following example shows the usage of the parameter `-sid`.

```
-sid=2 ; System ID for current system typically [2..7]
; increase by one for each system
; -sid % 32 cannot be zero
; [1..255]
```

As a result the following entries are generated in the ***T1*** project file

```
<System ... SID="2" ... >
...
<TargetId>2</TargetId>
```

and in the ***T1*** configuration header file

```
#define T1_SID (2)
```

5.73. Parameter -symbolPrefix

Definition `-symbolPrefix=`<*prefix*>

Short description Specifies any prefix added by the compiler to C symbols

Detailed description Some compilers create the assembly level symbol by adding a prefix to the the C level symbol. For example, the Green Hills RH850 compiler prefixes symbols with an underscore. This option defines this prefix and is used in the generation of <SymbolRef> elements in the ***T1*** project file.

Example The following example shows the usage of the parameter `-symbolPrefix`.

```
-symbolPrefix=_ ; e.g. main becomes _main
```

5.74. Parameter -syncPeriodMs

Definition `-syncPeriodMs=`<*sync period*>

Short description Period in milliseconds of automatically generated sync events during streaming.

Detailed description This parameter defines the period of the sync events which are automatically generated within T1_Handler during streaming operation in multi-core projects. The default value is 100 milliseconds. The maximum value corresponds to 255 times the period of T1_Handler. Set this value to zero to indicate that the application logs sync events and that T1_Handler need not generate additional sync events.

Example The following example shows the usage of the parameter `-syncPeriodMs`.

```
-syncPeriodMs=0
; sync events are periodically logged by user code
```

As a result the following entry is generated in T1_configGen.c, in the event that two cores are configured:

```
T1_uint8_t const T1_SEC_CONST_8 T1_syncPeriod[2]
  = {T1_SYNC_PERIOD_CORE0, T1_SYNC_PERIOD_CORE1};
```

And in T1_config.h, we get

```
#define T1_SYNC_PERIOD_CORE0 (0u)
...
#define T1_SYNC_PERIOD_CORE1 (0u)
```

5.75. Parameter -syncTimeBitLength

Definition `-syncTimeBitLength=`<*sync time bit length*>

Short description Synchronization time bit length (up to 32 bits).

Detailed description Synchronization of multi-core traces requires a common timer, used by all systems. This parameter specifies the number of actively counting bits in the synchronization time-stamps, to allow valid comparisons. The synchronization timer must have sufficient counting bits so that the range of the timer spans the maximum interval between successive calls to T1_Handler. See Section A.1.7 for information on earlier versions.

Example The following example shows the usage of the parameter `-syncTimeBitLength`.

```
-syncTimeBitLength=32
; width in bits of the synchronization timer [1..32] (default: 32)
```

As a result the following entries are generated in the ***T1*** project file

```
<Project ... SyncTimeBitLength="32"
 ...
</Project>
```

and in the ***T1*** configuration header file

```
#define T1_SYNC_TIMER_WIDTH_BITS (32)
```

See also `-pSyncTimer` (5.64)
`-syncTimerTickDurationNs` (5.77)
`-traceTimerIsSyncTimer` (5.89)
`-rawTimerBitLength` (5.67)
`-t1HandlerPeriodMs` (5.81)

5.76. Parameter -syncTimerDownCounting

Definition `-syncTimerDownCounting=`<*true/false*>

Short description This parameter is used to specify that the synchronization timer is counting down.

Detailed description This parameter must be set to `true` if the used hardware timer of the synchronization timer counts down.

Example The following example shows the usage of the parameter `-syncTimerDownCounting`.

```
-syncTimerDownCounting=true
; synchronization timer is counting down
```

If the synchronization timer counts down, this results in the generated macro `T1_GET_SYNC_TIME()` negating the raw value read from the hardware timer register.

See also

`-pSyncTimer` (5.64)
`-syncTimeBitLength` (5.75)
`-traceTimerIsSyncTimer` (5.89)

5.77. Parameter -syncTimerTickDurationNs

Definition `-syncTimerTickDurationNs=`<*Tick duration of synchronization trace timer*>

Short description Duration of one tick of the synchronization timer in nanoseconds.

Detailed description This describes the speed of the synchronization timer to the T1-HOST-SW so that it can merge multi-core traces. Note that the synchronization timer does not have to be the trace timer of any one core but can be a completely independent timer. In older configurations, the name "mainTickDuration" was used.

Example The following example shows the usage of the parameter `-syncTimerTickDurationNs`.

```
-syncTimerTickDurationNs=1000/100
; synchronization timer is running at 100MHz
```

As a result the following entry is generated in the ***T1*** project file:

```
<Project ... MainTickDuration="10"
 ...
</Project>
```

See also

`-pSyncTimer` (5.64)
`-syncTimeBitLength` (5.75)
`-traceTimerIsSyncTimer` (5.89)

5.78. Parameter -systemType

Definition `-systemType=<`*System type*`>`

Short description Provides additional information about the system to the T1-HOST-SW.

Detailed description This parameter is used to provide additional information about the system by setting it to one of the following values: `OSEK/ErcosEK/gliwOS/PCSoftware/RTA OSEK/AUTOSAR OS/Unknown`. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-systemType`.

```
-systemType=AUTOSAR OS
; [OSEK/ErcosEK/gliwOS/PCSoftware/
RTA OSEK/AUTOSAR OS/Unknown]
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ... SystemType="AUTOSAR OS" ... >
```

5.79. Parameter -targetType

Definition `-targetType=<`*Target type*`>`

Short description Provides additional information about the target to the T1-HOST-SW.

Detailed description The parameter `-targetType`

- is required to specify an MPC project where the Power (non-VLE) instruction set has been used by setting it to: `PPC5xxx_nonVLE`.
- is required to indicate which type of Arm core is used by setting it to: `ARM7M` for Cortex-M cores or `ARM7R` Cortex-R cores.
- This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-targetType`.

```
-targetType=PPC5xxx_nonVLE
; [PPC5xxx_nonVLE, ARM7M, ARM7R]
```

As a result the following entry is generated in the ***T1*** project file:

```
<TargetSpecifics ... Target="PPC5xxx_nonVLE" ... />
```

See also `-cortex` (5.26)

5.80. Parameter -t1FlexOverheadNs

Definition `-t1FlexOverheadNs=`<*T1.flex overhead*>

Short description Specifies the additional overhead incurred at a T1.flex trace event in nanoseconds.

Detailed description T1.flex has many advantages but the time required to trace an event via T1.flex is significantly higher than when `T1_TraceEvent()` is directly called from the code. This value allows the T1-HOST-SW to deduct this additional overhead for each T1.flex event when calculating a CET value. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-t1FlexOverheadNs`.

```
-t1FlexOverheadNs=500 ; [ns]
```

As a result the following entries are generated in the ***T1*** project file

```
<MeasureOverheads Extended="true">
 ...
 <T1flexMeasureOverhead>
  <T1>500</T1>
 </T1flexMeasureOverhead>
</MeasureOverheads>
```

and in the ***T1*** configuration header file

```
#define T1_FLEX_OVERHEAD_CORE0_NS (500)
```

5.81. Parameter -t1HandlerPeriodMs

Definition `-t1HandlerPeriodMs=`<*Period of T1 Handler*>

Short description Specifies the period of calls to `T1_Handler()`.

Detailed description This parameter is the system-specific period of call to `T1_Handler()` in milliseconds. It is recommended to call it every 5 to 10ms. This parameter needs to be applied for each system (core) individually. If the synchronization timer has less than 32 bits, the upper bits are computed by T1_Handler.

Example The following example shows the usage of the parameter `-t1HandlerPeriodMs`.

```
-t1HandlerPeriodMs=10 ; [ms]
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ... T1HandlerPeriod="10" ... >
```

See also `-syncTimeBitLength` (5.75)

5.82. Parameter -t1ScopeOverheadNs

Definition `-t1ScopeOverheadNs=`*<T1.scope overhead>*

Short description Specifies the minimum duration of one call to `T1_TraceEvent()` in nanoseconds.

Detailed description Using the minimum duration of one call to `T1_TraceEvent()`, the T1-HOST-SW deducts this overhead time from CET values to report the CET that would have occurred without the T1-HOST-SW instrumentation. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-t1ScopeOverheadNs`.

```
-t1ScopeOverheadNs=178 ; [ns]
```

As a result the following entries are generated in the ***T1*** project file

```
<MeasureOverheads Extended="true">
 <MainMeasureOverhead>
  <T1>89</T1>
  <T2>89</T2>
 </MainMeasureOverhead>
        ...
</MeasureOverheads>
```

and in the ***T1*** configuration header file

```
#define T1_OVERHEAD_CORE0_NS (178)
```

5.83. Parameter -tickDurationNs

Definition `-tickDurationNs=`<*Trace Timer Tick Duration*>

Short description Defines the trace timer tick duration in nanoseconds.

Detailed description This parameter is used to define the trace timer tick duration in nanoseconds for each system. Using this parameter the tick duration is provided to the T1-HOST-SW and the T1-TARGET-SW. For this parameter also fractions are allowed as input values. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-tickDurationNs`.

```
-tickDurationNs=1000 / 64
; Tick duration for a 64 MHz timer
```

As a result the following entries are generated in the ***T1*** project file

```
<System ... TickDuration="15.625" ... >
```

and in the ***T1*** configuration header file

```
#define T1_TICK_DURATION_CORE0_NS (1000 / 64)
```

See also `-rawTickDurationNs` (5.66)
`-syncTimerTickDurationNs` (5.77)

5.84. Parameter -timeoutResponseMs

Definition `-timeoutResponseMs=`<*Target response timeout*>

Short description Timeout on the target side for receiving an acknowledgment of a message from the host.

Detailed description This is the maximum time permitted between transmitting the last frame of a message by the target that requires acknowledgment and receiving the last frame of that acknowledgment by the target from the host in milliseconds. It is typically much larger than the GCP timeout limits, since it needs to allow for the T1-HOST-SW to respond in addition to transmission and reception delays. It should be set to a multiple of the parameter `-t1HandlerPeriodMs` of the communication core.

Example The following example shows the usage of the parameter `-timeoutResponseMs`.

```
-timeoutResponseMs=10000
; Target response timeout in ms
```

As a result the following entry is generated in `T1_config.h`, assuming a 10ms `-t1HandlerPeriodMs`:

```
#define T1_TIMEOUT_RESPONSE (1000u)
```

5.85. Parameter -timeoutRxMs

Definition `-timeoutRxMs=`<*Target receive timeout*>

Short description Defines the target receive timeout.

Detailed description The target receive timeout is the maximum time allowed between reception of two frames of the same message by the target in milliseconds. This needs to be large enough to allow for network traffic and routing delays between the host and the target. It should be set to a multiple of the parameter `-t1HandlerPeriodMs` of the communication core.

Example The following example shows the usage of the parameter `-timeoutRxMs`.

```
-timeoutRxMs=1000
; Target receive timeout in ms.
```

As a result the following entry is generated in `T1_config.h`, assuming a 10ms `-t1HandlerPeriodMs`:

```
#define T1_TIMEOUT_RX (100u)
```

5.86. Parameter -timeoutTxMs

Definition `-timeoutTxMs=`<*Target transmit timeout*>

Short description Minimum time allowed for (retrying) the sending of a frame by the target.

Detailed description The target transmit timeout is the minimum time allowed for (retrying) the sending of a frame in milliseconds. This needs to be large enough to allow for queuing in the CAN layer(s). Typically a smaller or equal number than the target receive timeout is used. It should be set to a multiple of the parameter `-t1HandlerPeriodMs` of the communication core.

Example The following example shows the usage of the parameter `-timeoutTxMs`.

```
-timeoutTxMs=1000
; Target transmit timeout in ms.
```

As a result the following entry is generated in `T1_config.h`, assuming a 10ms `-t1HandlerPeriodMs`:

```
#define T1_TIMEOUT_TX (100u)
```

5.87. Parameter -traceTimeStampBitLength

Definition `-traceTimeStampBitLength=`<*16/32*>

Short description This parameter must be set to 32, only when using a T1-TARGET-SW variant that has 32 bit timestamps.

Detailed description This parameter can be omitted unless using a T1-TARGET-SW variant that has 32 bit timestamps. This is different from `-rawTimerBitLength` in that it specifies not the actual number of counting bits but rather the width of the timestamp bit field in the trace events logged by this particular T1-TARGET-SW variant. Note that it is possible for `-rawTimerBitLength` to be equal to, smaller or larger than `-traceTimeStampBitLength`.

Example The following example shows the usage of the parameter `-traceTimeStampBitLength`.

```
-traceTimeStampBitLength=32
; 32-bit timestamps
```

As a result the following attribute is generated in the ***T1*** project file, if rawTimerBitLength is also set to 32.

```
MaxAbsTime="4294967295"
```

See also `-rawTimerBitLength` (5.67)

5.88. Parameter -traceTimerDownCounting

Definition `-traceTimerDownCounting=`<*true/false*>

Short description This parameter is used to specify that the trace timer is counting down.

Detailed description This parameter must be set to `true` if the used hardware timer of this core counts down. This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-traceTimerDownCounting`.

```
-traceTimerDownCounting=true
; The trace timer is counting down
[true/false(default)]
```

If all cores use the same timer and that timer counts down, this results in the generated macro `T1_GET_TRACE_TIME()` negating the raw value read from the hardware timer register.

See also `-pTimer` (5.65)
`-traceTimerIsSyncTimer` (5.89)

5.89. Parameter -traceTimerIsSyncTimer

Definition `-traceTimerIsSyncTimer=`<*true/false*>

Short description Indicates whether the trace timer on this core is the same as the shared, synchronization timer.

Detailed description This parameter is used when different timers are used in multi-core systems. By indicating that the trace timer on this core is the same as the shared, synchronization timer, synchronization can be attempted by the T1-HOST-SW even when no explicit synchronization event (trigger) is present in the trace. This parameter needs to be applied for each system (core) individually. If the trace timer is obtained from the physical trace timer register with a shift operation and the synchronization timer is equal to the not shifted physical trace timer register then this parameter must be set to `false`.

Please note that for each system in which `-traceTimerIsSyncTimer=true`, the macro `T1_GET_RAW_TIME_COREx` is automatically defined by the Perl scripts and therefore the user should not define it. Otherwise, for that macro, there will be multiple definition compiler errors, for that macro. See Section A.7.3 for information on earlier versions.

Example The following shows the usage of the parameter `-traceTimerIsSyncTimer`, using core 0 as an example and with `-pSyncTimer=0xF0000010`

```
-traceTimerIsSyncTimer=true
; trace timer on this core is also the
synchronization timer [true(default)/false]
```

As a result the following entries are generated in the ***T1*** project file

```
<TraceMergeAttributes
 TraceTimerIsMainTimer="true" />
```

and in the ***T1*** configuration header file

```
#define T1_TRACE_TIMER_IS_SYNC_TIMER_CORE0 (1)
#define T1_PTIMER_CORE0 (0xF0000010uL)
#define T1_GET_RAW_TIME_CORE0( ) \
                (*(T1_tickUint_t volatile *)T1_PTIMER_CORE0)
#define T1_GET_SYNC_TIME( ) \
                (*(T1_uint32_t volatile *)0xF0000010uL)
```

5.90. Parameter -txChannel

Definition `-txChannel=`<*CAN/CAN_FD/ETHERNET*>

Short description Selects the bus type of the TxChannel.

Detailed description This parameter specifies whether CAN, CAN FD or Ethernet is used as bus type for the TxChannel.

Example The following example shows the usage of the parameter `-txChannel`.

```
-txChannel=ETHERNET
; [CAN/CAN_FD/ETHERNET]
```

As a result the following entry is generated in the ***T1*** project file:

```
<BusType>ETHERNET</BusType>
```

5.91. Parameter -txCycleMs

Definition `-txCycleMs=`<*TX cycle of T1-HOST-SW*>

Short description Specifies TX cycle of T1-HOST-SW in milliseconds.

Detailed description For the correct choice of the T1-HOST-SW's TX-cycle it is important to consider the period of T1_Handler handling the communication between the target and the T1-HOST-SW. The TX-cycle must be equal or greater than the T1_Handler period. This parameter is relevant for CAN and CAN FD communication.

Example The following example shows the usage of the parameter `-txCycleMs`.

```
-txCycleMs=10
; TX cycle of T1-HOST-SW in ms
```

As a result the following entry is generated in the ***T1*** project file:

```
<TxCycleMs>10</TxCycleMs>
```

5.92. Parameter -useSameConnectionForAllSystems

Definition `-useSameConnectionForAllSystems=`<*true/false*>

Short description If this parameter is set to `true` then the same connection settings are used for all systems within this project.

Detailed description The connection parameters are global parameters for all systems in a project if `-useSameConnectionForAllSystems` is set to `true`, which is the default value. If system specific connections settings are required then `-useSameConnectionForAllSystems` must be set to `false` and they need to be added to each system configuration.

Example The following example shows the usage of the parameter `-useSameConnectionForAllSystems`.

```
-useSameConnectionForAllSystems=true
; All systems use the same connection
[true(default),false]
```

5.93. Parameter -useSameRxTxChannel

Definition `-useSameRxTxChannel=`<*true/false*>

Short description Specifies whether or not the system uses the same channel for RX and TX.

Detailed description If this parameter is set to `false` then different communication channals are used for RX and TX direction.

Example The following example shows the usage of the parameter `-useSameRxTxChannel`.

```
-useSameRxTxChannel=true ;
; Same channel usage[true(default)/false].
```

As a result the following entry is generated in the ***T1*** project file:

```
<CommunicationConfig
 UseSameRxTxChannel="true">
 ...
</CommunicationConfig>
```

5.94. Parameter -usingMulticoreLibs

Definition `-usingMulticoreLibs=`<*true*>

Short description Defines if multi-core libraries are used.

Detailed description Only ***T1*** multi-core libraries are supported, even for targets with one core, therefore this parameter must always be set to `true`, or simply omitted, as it is optional and only supported for backward compatibility. See Section A.1.4 for information on earlier versions.

Example The following example shows the usage of the parameter `-usingMulticoreLibs`.

```
-usingMulticoreLibs=true
; using-multicore libs [true/false].
```

As a result the following line is generated in *T1_config.h* if only one core is used with ***T1***.

```
#define T1_NOF_CORES (1)
```

If `-usingMulticoreLibs` is set to `false`, the value will be ignored.

A. Revision History

A.1. Migration from V2.6.y.z to V3.1.y.z

A.1.1. Basic Scheduling Frame ID

Prior to T1-TARGET-SW V3.1.0.0, the value for `-osBasicSchedFrameId` cannot be given as a hexadecimal number.

A.1.2. Parameter -cpuLoadCallback

Definition `-cpuLoadCallback=`<*CPU load callback function*>

Short description Defines the name of the CPU load callback function.

Detailed description Prior to T1-TARGET-SW V3.1.0.0, this parameter defines the name of the CPU load callback function which is called when a violation of the CPU load threshold has occurred. The default name of this function is `T1_CPULoadCallbackCorex()` and templates of it are implemented in *T1_config.c*. This parameter needs to be applied for each system (core) individually if the function name should be changed from its default value.

Example The following example shows the usage of the parameter `-cpuLoadCallback`.

```
-cpuLoadCallback=My_T1_CPULoadCallbackCore0
; CPU load callback function name
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_CPULOAD_CALLBACK_CORE0
 (My_T1_CPULoadCallbackCore0)
```

A.1.3. Parameter -traceTimerBitLength

Starting with T1-TARGET-SW V3.1.0.0, the parameter `-rawTimerBitLength` has replaced the parameter `-traceTimerBitLength`.

Definition `-traceTimerBitLength=`<*16 .. 32*>

Short description This parameter is required for a 32-bit CPU when a trace timer with fewer than 32 counting bits is used.

Detailed description This parameter needs to be applied for each system (core) individually.

Example The following example shows the usage of the parameter `-traceTimerBitLength`.

```
-traceTimerBitLength=24
; 32-bit CPU with a smaller timer
(example 24 bit) [16..32(default)]
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_TRACE_TIMER_BIT_LENGTH_CORE0 (24)
```

A.1.4. Parameter -usingMulticoreLibs

Prior to T1-TARGET-SW V3.1.0.0, both single-core and multi-core ***T1*** libraries are supported. Therefore, if using a multi-core target, the parameter `-usingMulticoreLibs` must be set to true. If using a single-core target, it must be set to false.

A.1.5. Parameter -pSyncTimer

Definition `-pSyncTimer=`<*address of synchronization timer*>

Short description Defines the address of the memory-mapped synchronization timer

Detailed description Prior to T1-TARGET-SW V3.1.0.0, this parameter is required in multi-core projects if all systems have `-traceTimerIsSyncTimer=false`. This means a different timer is used as synchronization timer than the timers used as trace timers.

Example The following example shows the usage of the parameter `-pSyncTimer`.

```
-pSyncTimer=0xFF811014
```

As a result the following line is generated in *T1_config.h*.

```
#define T1_GET_SYNC_TIME( ) (*(T1_uint32_t volatile *)0xFF811014)
```

A.1.6. Parameter -pTimer

Starting from T1-TARGET-SW V2.5.0.0 ***T1*** also provides direct support for using a non-memory-mapped timer. If a non-memory mapped timer is used, `-pTimer` is only regarded if it is the same for every core. Otherwise, the integrator must define the macro T1_GET_TRACE_TIME to return the trace time-stamp.

Starting from T1-TARGET-SW V3.1.0.0, the integrator must define the macro T1_GET_TRACE_TIME to return the trace time-stamp.

A.1.7. Parameter -syncTimeBitLength

Prior to T1-TARGET-SW V3.1.4.0, the use of T1.flex SWD and/or cross-core task activation with a synchronization timer width less than 28 bits could lead to incorrect results when the result was larger than the number of counting bits.

Prior to T1-TARGET-SW V3.1.0.0, foreground T1.cont is not supported and so there is no requirement that 28 bits (or more) are required for the synchronization timer if at least one trace timer is different from the synchronization timer.

A.2. Migration from V2.5.y.z V2.6.y.z

A.3. Migration from V2.4.y.z V2.5.y.z

A.4. Migration from V2.3.y.z V2.4.y.z

A.5. Migration from V2.2.y.z V2.3.y.z

A.6. Migration from V2.1.y.z V2.2.y.z

A.6.1. Parameter -endiannessAgnosticComSetup

Definition `-endiannessAgnosticComSetup=<`*true*`>`

Short description Enables compatibility of T1-HOST-SW with certain versions of the T1-TARGET-SW on big endian targets.

Detailed description This parameter is used for compatibility of the communication protocol of the T1-HOST-SW with T1-TARGET-SW versions starting with V2.2.0.0 and prior to V2.2.4.0 on big endian targets (e.g. MPC5xxx). The resulting entry in the ***T1*** project file is supported starting with T1-HOST-SW V2.3.0. In case of a big endian target the endianness must be correly configured with `-bigEndian=true`.

Example The following example shows the usage of the parameter `-endiannessAgnosticComSetup`.

```
-endiannessAgnosticComSetup=true
```

As a result the following entry is generated in the ***T1*** project file:

```
<System ... EndiannessAgnosticComSetup="true" ... >
```

See also `-bigEndian` (5.1)

A.7. Migration from V2.0.y.z V2.1.y.z

A.7.1. External Build ID

The use of an external (outside *T1_config.h* and therefore external to the ***T1*** configuration process) BID was introduced starting with T1-TARGET-SW V2.1.17.1.

A.7.2. Number of Focus Measurements

Prior to T1-TARGET-SW V2.1.18.0, only one focus measurement is possible on each system.

A.7.3. Trace Timer is Synchronization Timer

Prior to T1-TARGET-SW V2.1.10.0, the name `-traceTimerIsMainTimer` was used.

GLIWA
embedded systems

www.gliwa.com

GLIWA GmbH embedded systems
Pollinger Str. 1
82362 Weilheim i.OB.
Germany

fon +49 - 881 - 13 85 22 - 0
fax +49 - 881 - 13 85 22 - 99
mail info@gliwa.com

Geschäftsführer (CEO) Peter Gliwa
Amtsgericht München | HRB 167925
USt-IdNr. DE814169157