Using Command Line Mode - HelpID: 21460

Analysis from the command line utilises the console version of *LDRA Testbed* named *Contestbed*. In order to start *LDRA Testbed* from the command line use the **`start /wait`** command in conjunction with **`contestbed`** to invoke the tool. This can be undertaken from either the *LDRA* installation directory or from a path that includes the LDRA Installation directory.

```
start /wait contestbed
```

If you wish to take full advantage of Command Line Mode and begin processing automatically, the Windows command line syntax shown below should be followed

```
start /wait contestbed source_file_or_set /param2 /param3 /param4...
```

The first parameter, **`source_file_or_set`** on the command line must be the name (and optionally path) of the file or set to be to be analysed. The set or file must exist, see Creating Sets on the Command Line (Manual) on page 662 for details on how to Create Sets on the command line. Alternatively, see Creating Sets on the Command Line (Auto) on page 664 for creating Sets automatically from an IDE Project File. Any following options are introduced by a slash (**`/`**).

The second parameter is interpreted as a string of characters which relate to analysis routines required, e.g.:

```
start /wait contestbed triangle.c/cpp /112a34567q
```

Would mean:

`1`	Goto Static Analysis Phase
`1`	Perform Main Static Analysis
`2`	Perform Complexity Analysis
`a`	Analyse All procedures in source files
`3`	Perform Static Data Flow Analysis
`4`	Perform Cross Reference
`5`	Perform Information Flow Analysis
`6`	Perform Data Object Analysis
`7`	Perform MCDC Test Case Planner
`q`	Quit

This, then executes *LDRA Testbed* and performs full Static Analysis. (Not all options may be available on your configuration).

Command Line Mode can also perform both Instrumentation and Dynamic Analysis of the source file / Set. Continuing from the previous example:

```
start /wait contestbed triangle.c/cpp /112a3456702132yq
```

Would mean:

`1`	Goto Static Analysis Phase
`1`	Perform Main Static Analysis
`2`	Perform Complexity Analysis
`a`	Analyse All procedures in source files
`3`	Perform Static Data Flow Analysis
`4`	Perform Cross Reference
`5`	Perform Information Flow Analysis
`6`	Perform Data Object Analysis
`7`	Perform MCDC Test Case Planner
`0`	Return to Main Menu

2	Select Instrumentation Phase
1	Instrument Source File
3	Enable Exact Semantic Analysis
2	Compile Instrumented Source File
y	Use default Compilation Command
q	Quit

The above example will statically analyse the source file, instrument it, enable Exact Semantic Analysis and utilise the default compilation command set up for the File/Set to compile the instrumented program.1

The instrumented program can also be executed and perform Dynamic Coverage Analysis via Command Line Mode. The complication is that it is only sensible to execute the program where no user interaction is required during execution. If this is the case, then the user is limited to utilising Command Line Mode for performing Dynamic Coverage Analysis, Data Set Analysis and Profile Analysis after the instrumented program has been executed in its required environment.

For Dynamic Coverage Analysis only, the third parameter in the command line syntax is utilised:

```
start /wait contestbed source_file /param2 /param3 /param4...
```

It is interpreted as a data set name, e.g.

```
start /wait contestbed triangle.c/cpp /32pan34q /dataset=firstrun
```

Would mean:

3	Goto Dynamic Analysis Menu
2	Dynamic Coverage Analysis
p	Produce procedure by procedure analysis
a	Analyse All procedures in source file
n	Select No Trace
3	Perform Dynamic Data Set Analysis
4	Perform Profile Analysis
q	Quit

Relative Paths

Relative paths on the command line are relative to the workarea directory (`C:\LDRA_Workarea` or `~/ldra_workarea` by default).

If the `TESTBED.INI` flag `SWITCH_TO_WORKAREA=FALSE` is set this will cause relative paths to be relative to the directory the command is ran from.

Relative paths contained inside TCF files are relative to the location of the TCF file.

Examples of Command Line Mode

Perform Main Static Analysis, Complexity Analysis, Data Flow and Cross Reference:

```
start /wait contestbed SetName /112a34q
```

Perform Main Static Analysis, Complexity Analysis, Data Flow, Cross Reference, MCDC Test Case Planner:

```
start /wait contestbed SetName /112a347q
```

Perform all Static Analysis, Instrumentation and Build process:

```
start /wait contestbed SetName /112a345670212q
```

Same process as above, all Static Analysis, Instrumentation and Build process, but in two commands to separate Static and Instrumentation phases from each other:

```
start /wait contestbed SetName /112a34567q
start /wait contestbed SetName /212q
```

Perform Main Static Analysis, Instrumentation, Build, Execution and Dynamic Coverage Analysis:

```
start /wait contestbed SetName /112a3456702120312panq
```

Same process as above, all Static Analysis, Instrumentation and Build Execution and Dynamic Coverage Analysis, but in three commands to separate Static and Instrumentation phases from each other:

```
start /wait contestbed SetName /112a34567q
start /wait contestbed SetName /212q
start /wait contestbed SetName /312panq
```

Command Line Analysis Options

Analysis options are accessed via the Command Line by pre-typing menu selections. Below is a list of all menu options and their relevant selection letters/numbers:

Menu Structures for Command Line Mode

Please Note: That the menu structure does vary from platform to platform, the list below addresses this situation, therefore not all the options described below will be available in your implementation.

Main Options (top level menu)

- `0` Ignored
- `q` Quit
- `1` Select Static Phase Menu
- `2` Select the Instrumentation Phase Menu
- `3` Select the Dynamic Phase Menu

Static Options (Static Phase menu)

- `0` Return to the top level menu
- `q` Quit
- `1` Perform Main Static Analysis
- `2` Perform Complexity Analysis (this must be followed by an indication of whether full output is required for all or none of the procedures)
  - `a` full output for all procedures
  - `n` full output for no procedures
- `3` Perform Static Data Flow Analysis
- `4` Perform Cross-Reference
- `5` Perform Information Flow Analysis
- `6` Perform Data Object Analysis
- `7` Perform MCDC Test Case Planner

Please Note: Some analysis options cannot be performed until the successful completion of previous ones, i.e. Complexity Analysis requires Main Static Analysis to be performed first.

Instrumentation Options (Instrumentation Phase menu)

- `0` Return to the top level menu
- `q` Quit
- `1` Instrument the program
- `3` Enable Exact Semantic Analysis
- `2` Build the instrumented program

Please Note: Appropriate paths and environment variables must be set for compilation and linking.

Dynamic Options (Dynamic Phase Menu)

- `0` Return to the top level menu
- `q` Quit
- `1` Execute instrumented program
- `2` Perform Dynamic Coverage Analysis, followed by:

  - `p` Perform a procedure by procedure analysis
  - `f` Perform a full file analysis

  If Procedure by procedure was selected:
  - `a` Provide details for all procedures
  - `n` Provide details for no procedures

  If Procedure by procedure was selected:
  - `n` Do not provide a trace (mandatory for procedure based analysis)

  If full file was selected:
  - `n` Do not provide a trace
  - `y` Provide a trace

  If full file and trace was selected:
  - `a` Trace all lines (mandatory)

- `3` Perform Dynamic Data Set Analysis
- `4` Perform Profile Analysis
- `5` Perform Dynamic Data Flow

Deleting Workfiles via the Command Line

To delete workfiles from the command line use the parameter /94q:

```
start /wait contestbed <file_or_set> /94q
```

This will delete ALL workfiles, selected deletion can be acheived by using the commands:

/91q	Delete Workfiles
/92q	Delete Workfiles and Results
/93q	Delete Instrumented Source and Executable Program(s)
/94q	Delete All
/95q	Delete Dynamic Coverage Analysis Results

Creating Sets on the Command Line (Manual)

To analyse a set via command line mode, you will first need to create the set either from within the *LDRA* GUI or by calling *contestbed* from the command line. Once the set is created, the following steps will not have to be repeated.

Creating a Set on the Command Line

`/create_set` - Creates a Set.

`/add_set_file` - Add files to an existing Set.

`/reanalyse_changed_set` - used in conjunction with `/add_set_file` when adding a file to a System set, this will re-run the analysis for the set with the new file. Not required if analysis on the set has not been done prior to adding the file. If adding multiple files the command does not need to be repeated for each file.

`/remove_set_file` removes a file from an existing set.

`/delete_set` deletes a set.

1. Open a Command Prompt and change directory to the LDRA Installation directory.
2. To create a system set called "myset" run the following command:

   ```
   start /wait contestbed myset /create_set=system /1q
   ```

   Alternatively you can create a group set by running the following command:

   ```
   start /wait contestbed myset /create_set=group /1q
   ```

3. To add files to the set run the following command:

   ```
   start /wait contestbed myset /add_set_file="path_to_file\file.c" /1q
   ```

   e.g.

   ```
   start /wait contestbed myset
   /add_set_file="Examples\C_testbed_examples\
   Ggset\Ggset_main.c" /1q
   ```

Please Note: It is possible to combine the above commands from stages 2 and 3 to create the set and add files to it in one operation.

E.g.

```
start /wait contestbed tbsdemo /create_set=system /add_set_file="Exampl
es \C_testbed_examples\Tbsdemo\Tbsdem1.c" /add_set_file="Examples\
C_testbed_examples\Tbsdemo\Tbsdem2.c" /add_set_file="Examples\
C_testbed_examples\Tbsdemo\Tbsdem3.c" /1q
```

Removing Files from a Set via Command Line

Use the following command line qualifier to remove a specific file from the set:

```
/remove_set_file=
```

If using spaces in file names remember to use quotation marks around the location:

```
/remove_set_file="C:\Source Files\remove_me.c"
```

Worked examples:

```
start /wait contestbed "C:\Ggset\Ggset.vcproj" /1q /remove_set_file="C:\Ggset\GGset_how_many_favourite_fruit.c"

start /wait contestbed C:\Ggset\Ggset.vcproj /1q /remove_set_file=C:\Ggset\GGset_get_number_of_pence.c

start /wait contestbed C:\Ggset\Ggset.vcproj /1q /remove_set_file=GGset_fruit_cheaper_than.c

start /wait contestbed C:\Ggset\Ggset.vcproj /1q /remove_set_file=Ggset\GGset_buy_fruit.c
```

Deleting Sets via the Command Line

Use the following command line qualifier to delete a set:

```
/delete_set=set_name
```

Worked example:

```
start /wait contestbed /delete_set=tbsdemo
```

Creating Sets on the Command Line (Auto)

To create sets automatically from the command line, a Project file (of a supported format) must be provided which details the files that make up the set.

Set Creation Command File Commands

LDRA Testbed can create sets from the following file formats:

- *LDRA* **.tcf**
- Microsoft **.dsp**
- Microsoft **.dsw**
- Microsoft **.vcproj**
- Microsoft **.vcxproj**
- Microsoft **.sln**
- Green Hills **.bld**
- Green Hills **.gpj**
- Wind River **.wrproject**

Analysing sets from the command line is a similar process to analysing single files. The file holds the information on which files are included in the set and where to find them. An example of this process would be:

```
start /wait contestbed C:\myset\set.vcproj /create_set=system /112a34q
```

- The command `create_set` instructs contestbed to create the Set from the Project File as a System Set.

Note: When using Microsoft's **.dsw** and **.sln** (workspace and solution) files, *LDRA Testbed* creates a set for each project found in the file. If analysis flags are used, analysis is performed on the last set created.

Confirming Analysis Results via the Command Line

LDRA Testbed uses the following flag to test whether results for the selected phases exist. If results exist for a selected phase, it will not be run.

```
start /wait contestbed <path>\/Dispense.cpp /112q /batch_mode_verify_
results
```

Example Output:

```
-------------------------
Command Line Mode Started
-------------------------
Main Static Analysis results verified
Complexity Analysis results verified
```

Partial Instrumentation in Command Line Mode

For users wishing to only instrument certain files/procedures in their analysis scope, the following command line qualifiers can be used when analysing from the command line.

To force the Instrumenter to not instrument a file in a Set use the following qualifier:

```
/no_inst_file=<path>\<filename.ext>
```

E.g.

```
start /wait contestbed tbsdemo /1120212q /no_inst_file=C:\ldra_workar
ea\examples\tbsdem1.c
```

To force the Instrumenter to not instrument a specific procedure, use the following qualifier:

```
/no_inst_file=<path>\<filename.ext>:<procedure>
```

E.g.

```
start /wait contestbed tbsdemo /1120212q /no_inst_file=C:\ldra_workar
ea\examples\tbsdem1.c:equal_sides
```

TBpublish Command Line Mode

TBpublish directory can be on the command line with argument:

/publish_to_dir=<directory>

If `<directory>` does not exist, it will be created. If this command line argument is used in conjunction with batch mode analysis, results will be published to `<directory>` at the end of analysis.

For Batch Mode Analysis the following qualifier can be used in conjunction with a Testbed.ini flag:

/publish_to_default_dir

Publishes to the directory set by Testbed.ini, `PUBDIR=`

To specify an Index Template:

/html_index_template=<template>

Specify one of the available templates from: *MISRA, DYNAMIC, MANAGEMENT, AV_STANDARD* or *FULL*. This overrides any default set by Testbed.ini entry `HTML_INDEX_TEMPLATE`.

To set the Published report type:

/publish_rep_type=[ASCII,HTML]

This can also be specified via a Testbed.ini entry

PUBLISH_REP_TYPE_OVERRIDE=[ASCII,HTML]

HTML reports can be regenerated without links to the source code via ActiveX. Set Testbed.ini entry `PUBLISH_NO_ACTIVEX=TRUE` to enable as default.

To publish as a text file (.txt)

/publish_as_txt

/nopublish_as_txt

This can also be specified via a Testbed.ini entry

PUBLISH_AS_TXT=[TRUE,FALSE]

Using **/nopublish_as_txt** will override the INI setting.

General Command Line Qualifiers

Qualifier	Description
batch_mode_verify_results	When used with command line analysis, tests whether results for the selected phases exist. If results exist for a selected phase, it will not be run.
create_set=<group/system>	Enforces a particular Set type when loading a project file from the command line
DATASET=*string*	Specify name of data set used by Dynamic Coverage Analysis. See Dynamic Coverage Options Dialog - HelpID: 27366 on page 401.
<lang>_DIALECT=<dialect>	Change dialect for analysis. E.g. **CPP_DIALECT=RHAPDY**
EXHDIR	Specify the directory where *LDRA Testbed* will look for an execution history e.g. /exhdir=C:\my_exhdir See Execute the Instrumented Program on page 535.
export_tbed_tcf=<tcf name>	Generates a Testbed TCF at the specified location
IDIR=<DIRECTORY>	The *LDRA Testbed* will attempt to create instrumented files in the named directory. The default is the current working directory. Note that the user must have permission to write in the **/idir** directory. This option is not allowed with **/iprog** when using a full filename. Once used, this option persists for the file under analysis until the analysis results for this file are deleted.
c/cppINSTRFILE=<...>	Specify filename of (non-standard) instrumentation data file. The filename may include its path. Once used, this option persists for the file under analysis until the analysis results for this file are deleted.
IPROG=<FILENAME>	This option tells *LDRA Testbed* to write the instrumented source to the specifically named file. No **/idir** option is allowed in combination when using a full filename. This option is not usable with *Sets*. If **/iprog** is not used the instrumented file is placed, by default, in the **/idir** directory using a name derived from the source filename with the prefix "inszt_". It is the users responsibility to ensure that the /**iprog** file can be written. i.e. any named directories in the full file name must exist and the **/iprog** must be writeable.
INCLUDE	Expand include files where possible.
NOINCLUDE	Do not expand include files
no_inst_file=<path_to_file>	**See Partial Instrumentation in Command Line Mode on page 665.**
no_inst_proc=<path_to_file>:<procedure>	**See Partial Instrumentation in Command Line Mode on page 665.**
c/cppPENFILE=<...>	Specify filename of (non-standard) penalty file

Qualifier	Description
PERMDIR=<DIRECTORY>	The *LDRA Testbed* will use the named directory for its "permanent" files. These are the **staticid.dat** and the **LDRA.00** files. The **staticid.dat** file contains a database of all analysed source files and is used, in some *LDRA Testbeds*, to generate unique instrumentation identifiers. The **LDRA.00** files hold the analysis settings of the analysed files. Using **/permdir** enables a set of unique identifiers to be shared by source files from a number of different directories. The default for **/permdir** is **workdir**. Using this Command Line Qualifier modifies the permdir entry in Testbed.ini causing this setting to persist for all future analysis until it is modified again.
<lang>quality_model=<model>	Sets the <lang>STANDARDS_MODEL in INI File
quit	Equivalent to /1q
reanalyse_changed_set	Used with batch analysis of systems, forces interdependent results file deletion and reanalysis. For example, if a File is added/removed to a Set, the qualifier deletes interdependent results and performs the necessary analysis. In some cases, existing Results may still be upto date and not require a full workfile deletion and reanalysis.
reset_options	Resets all options to their defaults for the file/set named on the command line. Option changes are typically made through the User Interface or by importing settings from a TCF file
SYSCALLS_DATA_FILE=<...>	Specify syscalls data file. e.g. **SYSCALLS_DATA_FILE=msvc_syscalls.dat**
<Lang>SYSCALLSFILE=<...>	Specify Language specific syscalls data file. e.g. **CPPSYSCALLSFILE=msvc_syscalls.dat**
SYSEARCH=<...>	Specify (non-standard) filename and location of the file named **sysearch.dat** as default.
SYSPPVAR=<...>	Specify (non-standard) filename and location of the file named **sysppvar.dat** as default.
tbedtcf=<tcf options file>	Used to specify a TCF file containing Options, Sysearch Include File Entries or Sysppvar Preprocessor Macros sections. These will override existing settings for a file or set specified on the command line.
THISDIR	Puts instrumented program and execution history in current directory using *LDRA Testbed*, not the directory of the original source code. See also **IPROG**.
c/cppVALSFILE=<...>	Specify (non-standard) filename and location of the file named **c/cppvals.dat** as default

Qualifier	Description
getstaticid	Use with a source file or set command line argument /get-staticid=file. Writes the staticid number of the named source file to the standar error channel. Use /getsid_outfile=outfile to write the staticid number to a named file. The result is less than 1 if the staticid number cannot be found. This qualifier should be used in place of the utility TBgetstaticid.
review	Runs Static analysis phases, Equivalent to /1120345q
run_required_dynamic	Runs phases required for Dynamic Analysis: If no results exist is equivalent to the command: /112a302120312panq If results already exist for a phase will not repeat analysis for that phase.
run_required_dyndflow	Runs phases required for Dynamic Dataflow Coverage. Replaces the comand **/112a34021203125q** e.g. contestbed testrian.c /run_required_dyndflow
force_analysis	Overrides result checking and forces analysis to be re-ran even if results are present and up-to-date. Use: contestbed testrian.c /112a345q /force_analysis
no_coverage_inst	No Instrumentation for Coverage, no procedures will be instrumented. Testbed will use the original file(s)
generate_overview_rep	Use with a source file or set to create the Test Manager Report.
delete_single_file	Deletes a single file. e.g. /delete_single_file=C:\Project\Myfile.c This will delete all workfiles for this file including the GLH

Advanced Command Line Qualifiers

The following Command Line options may be used under the guidance of the LDRA Support Team. Further assistance with the usage of these qualifiers will be provided.

Qualifier	Description
SHORTEN	Create temporary version of source file (sourcefile-name.cod) which has has code length shortened, carriage returns added and white-space eliminated. Part of Main Static Analysis, performs some preprocessing and include file search. Generated .cod file retained with KEEP qualifier. This is useful when analysing preprocessed files.
NOSHORTEN	This Command Line Qualifier can be utilised if *LDRA Test-bed* defaults are altered.

.

Modifying Testbed.ini file from a Batch/Script File (TBini.exe)

The set-up of the Testbed.ini file, situated in the Windows directory of your machine or located via the TESTBED environment variable is instrumental to the behaviour of *LDRA Testbed* for any analysis. A change of some of the control flags within the file will cause the results of a *LDRA Testbed* analysis to differ.

LDRA has developed a program for automation called `TBini.exe`, that has the capability of changing the `Testbed.ini` file from the command prompt , or within a batch file. At the start of each analysis, `tbini.exe` can be invoked to change or add any flag that the user requires.

For example, if the user wishes the "`BUILD_OPTIONS_FILE=`" flag to use a different compiler data file, then

```
start /wait TBini BUILD_OPTIONS_FILE=g:\LDRA_Toolsuite\MSVC_TESTBED.d
at
```

should be used.

NOTE: The commands used in these examples show the commands ran from the LDRA Tool Suite installation directory, if ran from another directory full or relative paths to TBini will need to be included.

By default, all entries passed to `TBini.exe` go into the `[<lang> LDRA Testbed]` section of the `Testbed.ini` file. Users with multiple language LDRA installations or multiple compiler configurations may need to specify the section of the `Testbed.ini` file to write to.

For example, to define a full section name, use the *Section* argument:

```
/Section="C/C++ LDRA Testbed"
```

```
start /wait tbini /Section="C/C++ LDRA Testbed"
WORKDIR=g:\LDRA_Workarea\
```

Alternatively use the *Profile* argument to specify the section:

```
/Profile="C/C++"
```

```
start /wait TBini /Profile="C/C++" WORKDIR=g:\LDRA_Workarea\
```

To remove a flag using TBini, enter the name without an assignment, for example:

```
start /wait TBini /profile="C/C++" WORKDIR
```

This will revert to the default as defined in the *LDRA* executable.

Note that some flags are entered by default with the name and the "=" but without the assignee for example:

```
C_DIALECT=
```

In some cases there is a difference between the lack of a flag and one with a null assignment. One example would be:

```
EXHSTRIP_REPRINT_FORMAT
```

Check the appropriate documentation as to the valid assignments

Multiple Testing on the Same File

Within *LDRA Testbed*, multiple testing of a single source file would result in one execution history being produced for all of the executions of that file. When producing multiple execution histories from outside of the *LDRA Testbed* environment, the execution history file will only contain the last execution history. The batchfile will need to append new executable histories to an arbitrary file that is not overwritten when the source file is being tested repeatedly.

When a new execution history file is produced the contents should be written to a master exh file, named so that it can not conflict with the testing. Every time the instrumented executable is run and a new execution history file is produced, the text from the file should be appended to the master exh file.

So *LDRA Testbed* will know when a new execution history has finished and a new one will start, the termination character

```
-1
```

should be written between each execution history. Please note that if you are using any instrumentation strategy other that standard single point streamed then you need to replace the -1 with the appropriate splitter marker. See the Instrumentation section for further details.

The text in the execution histories will resemble a stream of numbers, -1 will need to be placed in the identical column as these numbers. There should be no termination string after the final execution history has been written to the master exh file. When all execution histories have been collected, the master exh file should be renamed to the original execution history name. Only then can Dynamic Coverage Analysis be performed.

Here is an example of the above methodology with triangle.c as the source:

```
inszt_triangle
```

The instrumented exe is executed

```
copy triangle.exh master_triangle.txt
```

The resulting `triangle.exh` file is copied to a master text file

```
echo      -1>> master_triangle.txt
```

The termination string -1 is piped to the master text file, exactly 6 characters long.

```
inszt_triangle
```

The instrumented exe is executed for the second time

```
type triangle.exh >> master_triangle.txt
```

The body of the new `triangle.exh` is appended to the master text file

```
echo      -1>> ch_triangle.exh
```

The termination string -1 is appended to the master text file

```
inszt_triangle
```

The instrumented exe is executed for the last time. No termination string is needed

```
type triangle.exh >> master_triangle.txt
```

The body of the new `triangle.exh` is appended to the master text file

```
del triangle.exh
```

The new `triangle.exh` file is deleted

```
copy master_triangle.txt triangle.exh
```

The master text file become the execution history file

```
del master_triangle.txt
```

The master text file is then deleted

```
start /wait %testbed_inst%\testbed triangle.c /32q
```

Dynamic Coverage Analysis is run with the complete execution history file of three test cases

Additional

LDRA Testbed TCF Entries

Storage of content and settings of a source file or set loaded into LDRA Testbed.

TCF information:

```
# Begin Testbed Set

SINGLE_FILE = TRUE
```

or

```
SET_TYPE = SYSTEM [or GROUP]
SET_NAME = tbsdem

GENERATED_BY = Testbed 9.0.0 [or other tool]
```

Source information:

```
# Begin Source Files

RelativeFile = .\Tbsdem1.c
File = F:\c_cpp_testbed\Examples\C_testbed_examples\Tbsdemo\Tbsdem1.c
RelativeFile = .\Tbsdem2.c
File = F:\c_cpp_testbed\Examples\C_testbed_examples\Tbsdemo\Tbsdem2.c
...

# End Source Files
```

When importing a TCF, the path used in the RelativeFile entry is relative to the current TCF file directory. If source locations cannot be determined when importing, LDRA Testbed will display a dialog to allow the user to locate source files.

When exporting, the path assigned to the RelativeFile entry is relative to the source root directory. Hence it is advisable to place a TCF file for a set in the source root directory.

Include file search information (specification of any entries found in the sysearch.dat file currently in use when exported):

```
# Begin Sysearch Include File Entries

SearchPath = F:\c_cpp_testbed\Examples\C_testbed_examples\Tbsdemo\
ExcludedFile_201                                                =
F:\c_cpp_testbed\Examples\C_testbed_examples\Tbsdemo\Tbsdem2.h
RelativeInclude = .\tbsdem3.h

# End Sysearch Include File Entries
```

The RelativeInclude entry is automatically generated by Testbed when exporting to a TCF, and is only available after Main Static analysis. It shows paths of include files that have been analysed, relative to source or search paths that were used. When importing a moved or copied TCF, it is used to help locate include files and directories.

When the TCF is imported these entries are converted back into a new sysearch.dat file. Relative paths in these entries have the same meaning as they do in a sysearch.dat file i.e relative to an analysed source file.

Macro definitions (as specified in the sysppvar.dat file currently in use when exported):

```
# Begin Sysppvar Preprocessor Macros

MacroEntry = DEBUG 1

# End Sysppvar Preprocessor Macros
```

When the TCF is imported these are converted back into a new sysppvar.dat file

Options section.

Configuration options settings for the current file or set. Any relative paths are relative to the LDRA Testbed start-in directory. Where appropriate, macros such as $(Userdir) are used when the TCF is exported.

<Default> indicates the installed default value.

```
# Begin Options

set_attributes = 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
nb_mechanism = build
nb_start_in_dir = $(SourceDir)
nb_build_command = cl "$$d$$f.$$e" -o "$$i$$x$$g"
nb_compile_command = cl /c "$$d$$f.$$e"
nb_link_command = link /out:"$$i$$x$$g" "$$d$$f.obj"
nb_ide_command = msdev "$$o.dsw"
nb_makefile_command = nmake /f "$$m"
nb_makefile_name = $$o.mak
nb_compile_each_file = T
nb_build_each_file = T
nb_makefile_in_start_in_dir = T
nb_proj_nmake_command =
nb_proj_nmake_file =
nb_proj_msdev_command =
nb_proj_msdev_file =
nb_projfile_in_start_in_dir = T
nb_proj_makefile_in_start_in_dir = T
nb_remove_pch = F
nb_substitute_source = F
sub_unused_files = F
nb_pipe = F
nb_force_console = 2
enable_pre_build = 0
enable_post_build = 0
make_greenhills_bld = 0
nb_harness = "$$i$$x$$g"
nb_executepath = $(Userdir)
nb_execute_each_file = F
exhdir = $(Userdir)
modinst_exh_stem = $(SourceStem)_$(Modnum)
using_exhstrip = F
using_validexh = T
single_set_exh = F
compress_exh = F
threaded_exh = F
memory_address_exh = F
```

```
compress_exh_method = bitmap
do178b_instr = T
tbcapture_instr = F
array_bounds_check = F
copy_ref_instr = 0
archive_exh = T
force_dynsplit = T
dyn_scan_option = 3
dyn_scan_custom_coverage = 5 60 100 100 60 100
dyn_scan_current_coverage = 5 60 100 100 60 100
dyn_scan_html = F
dyn_oview_sort = 0
dyn_oview_options_changed = 0
dyn_oview_file_by_file = 0
dyn_oview_files_table = 0
syntax_only = F
include = T
open_all_includes = F
last_complexity_choice = 0
keep = T
shorten = T
tbset_data_file = tbset.dat
workdir = tbwrkfls\
y2k_text_format = 0
y2k_column_width = 80
ddf_text_format = 0
ddf_column_width = 80
tbr_level = F:\Rel_TCF_test\TCFs\Grocers\Ggrocers.c 39
iprog = inszt_$$o
iext = $$t
idir = $(Userdir)
exe_name = $$f
exe_append = .exe
cinstr_data_file = <Default>
cppinstr_data_file = <Default>
cpenfile = ..\config\Standards\cpen.dat
cpppenfile = <Default>
cvalsfile = <Default>
cppvalsfile = <Default>
cglbfile = <Default>
cppglbfile = F:\C_CPP_Testbed\cpp\cppsysglobal.glb
cstandards_model = <No Model - all standards>
cexternal_standard = None
c_dialect = MSVC
cppstandards_model = <No Model - all standards>
cppexternal_standard = JSF++ AV
cpp_dialect = MSVC
sysearch = <Default search - Source dir(s) - Installdir>
sysppvar = <Default search - Source dir(s) - Installdir>
dynamic_run_number = 1
history = 10
coverage = 3
exhwidth = 0
logging = F
thisdir = T
dosnames = F
scanondemand = F
iprog_switch_os_format = F
```

```
iprog_flush_exh = F
preprocess_iprog = T
instr_template_io = F
current_compiler = Microsoft Visual C/C++ v6.0
baseline_datestamp = 0
baseline_datestring =
baseline_wf_datestamp = 0
baseline_wf_datestring =
baseline_num = 0
tb_settings_modified = 0
baseline_path = F:\C_CPP_Testbed\tbblfls\Ggrocers_183\

# End Options

# End Testbed Set
```