IFX STACK DEPTH ANALYSER
USER MANUAL

Doc ID: IFXSDA-UM-1.11
Release Date: 8th Dec 2006
Version 1.11

Edition 8th Dec 2006

Published by Infineon Technologies AG,
D-81726 Munich, Germany

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

1. Introduction

- ***Overview***

  This tool reports Stack Usage of a particular TriCore application from its linker output. The linker output (.elf) is generated using the TASKING EDE build process. This tool is a static tool and does not track run-time stack usage. It tracks the application user stack, interrupt stack and the number of context blocks used.

- ***Resource Requirements***

  The tool is released as pre compiled and build executable for Intel x86 machine with Windows 32-bit platform.

  Hardware
  The host machine will be an Intel PC.
  The target hardware is TC1766, TC1796.

  Software
  The host PC should run Windows2000/XP. Support for Linux based machines is also available but the binaries have to be rebuilt.

  The TASKING EDE or the TASKING Tool Chain v2.2r3 patch [P1] has to be used for generating the image file for the intended application on Windows.

- ***Assumptions and limitations***

  The tool has following assumptions and limitations:

  1. The input ELF file should be compatible with the Application Binary Interface (ABI).

  2. The input ELF file should have a valid symbol table where the information about each function in that application is present.

  3. This tool essentially scans for TASKING compiler generated assembly instruction combinations for stack size manipulation. It may not be able to interpret all innovative combinations of assembly instructions hand-coded by the end user for stack size manipulations.

  4. In TASKING Compiler, projects containing static functions should be built with custom project settings specified in Appendix C.

  5. IFXSDA requires a few minutes for computing the best and worst case of interrupt stack usage for files of the order of a few mega bytes in size.

6. IFXSDA analyses only the following TriCore instructions generated by TASKING for calculating the stack requirement of an application:
    - SUB.A A[10], const8
    - LEA A[a], A[b], off10
    - LEA A[a], A[b], off16
    - MOVH.A A[c], const16
    - ADD.A A[a], A[b]
    - CALL disp24
    - CALL disp8
    - CALLI A[a]
    - J disp24
    - J disp8

   Projects containing Assembly source code should adhere to the above specified instructions for stack manipulation and for function calls.

2. Installing IFXSDA

IFXSDA comes with an installer which installs the IFXSDA tool on the Windows system. The user has to invoke setup.exe to start the tool.
Follow the steps given bellow to install the Tool.

1. Welcome dialog: Click **Next** to continue.
2. Select the directory where tool need to be installed. Use **Browse** to select directory other than the default. Click **Next** to continue.
3. Select the **Start** menu folder where a shortcuts directory is created.
   Ready to install: Check the directory and start menu folder in the information box. Selection of the directories can be changed by using **Back**. Click **Install** to Install the software on the system.
4. To bypass reading the Release Notes uncheck the **View Released Notes.pdf** checkbox. Click **Finish** to finish the installation procedure.

   Short cut to the tool is provided for invoking the tool from **Start** menu as Start->Programs->IFXSDA-> IFX Stack Depth Analyzer

3. Using IFXSDA

IFX SDA can be invoked from command line as well as graphical user interface (GUI). In the command line mode, user input can be given either on command line itself or through the user written command file. In the GUI mode tool works interactively.

- ***Invoking IFXSDA in command line mode***

If the user has selected the default installation path for the tool then executable can be found as *Default Program files directory\IFXSDA\IFXSDA.exe*.
(Note: Default Program files directory is the location of the System program files. E.g. C:\Program Files)
The general method of invoking the IFXSDA tool in command line is as follows:
IFXSDA <Elf file_name> -nogui <command options>
Note: *Elf file name* should be specified in MS-DOS file name convention. Please refer **Appendix B** to know more about the MS-DOS file name convention.
The –nogui option is required to run the tool in the command line mode. If this option is not present on command line, the tool will ignore all other command line arguments and start working on GUI mode.
In the command line mode user can give input via following command options.

Command options
-h
Display a summery of command line options.
E.g. IFXSDA –h –nogui
Note that if –h option is encountered on command line the tool will ignore all other options and quit after displaying the help.

-f <entry point function>
Variable entry point function from which stack usage info is desired.

-i
This option instructs the tool that indirect recursion information is desired. By default tool assumes that the indirect recursion information is not desired.
Indirect recursion in a call graph means a call sequence of the kind N1callsN2, N2callsN3, N3callsN1 (which means call graph has a cycle in it.)
if this option is seen the output will give the possible indirect recursion paths and their stack usage as if the path is executed once.

-I <caller function name> <called function name>
This option is for specifying the name of the indirectly called function.
An indirect function call means a call to a function through a pointer. In that case the call is resolved at run-time depending on the content of the function pointer. By analyzing statically, the tool will be unable to guess which function is being called. So we need the user to mention which function he was intending to call through the pointer.

-r <name of the recursive function> < recursion limit>
This option is for specifying the recursion limit for a particular recursive function. A function which calls itself is a recursive function. How many times the recursion will occur is determined at runtime based on input parameters to the function. There is no way of predicting this statically. So the user needs to give an estimate of the number of times recursion will occur for this function.

-o <Output file name>
This option lets the user to direct all the tool messages to a file. With –o option all the tool messages are printed to Output file named as well as on the screen.

-v
In this mode it prints the product, version and information and the company information. Note that, this option can not be specified in command file as this option does not affect the functional behavior of the tool.
e.g. 1. IFXSDA.exe -v –nogui
2. IFXSDA.exe ex.elf –v nogui
In the first example the tool prints the required information for –v option and exits. In the second example the tool prints the required information for –v option and also show the results for ex.elf file.

-t
This option prints the timing information about tools current run. Note that, this option can not be specified in command file as this option does not affect the functional behavior of the tool.

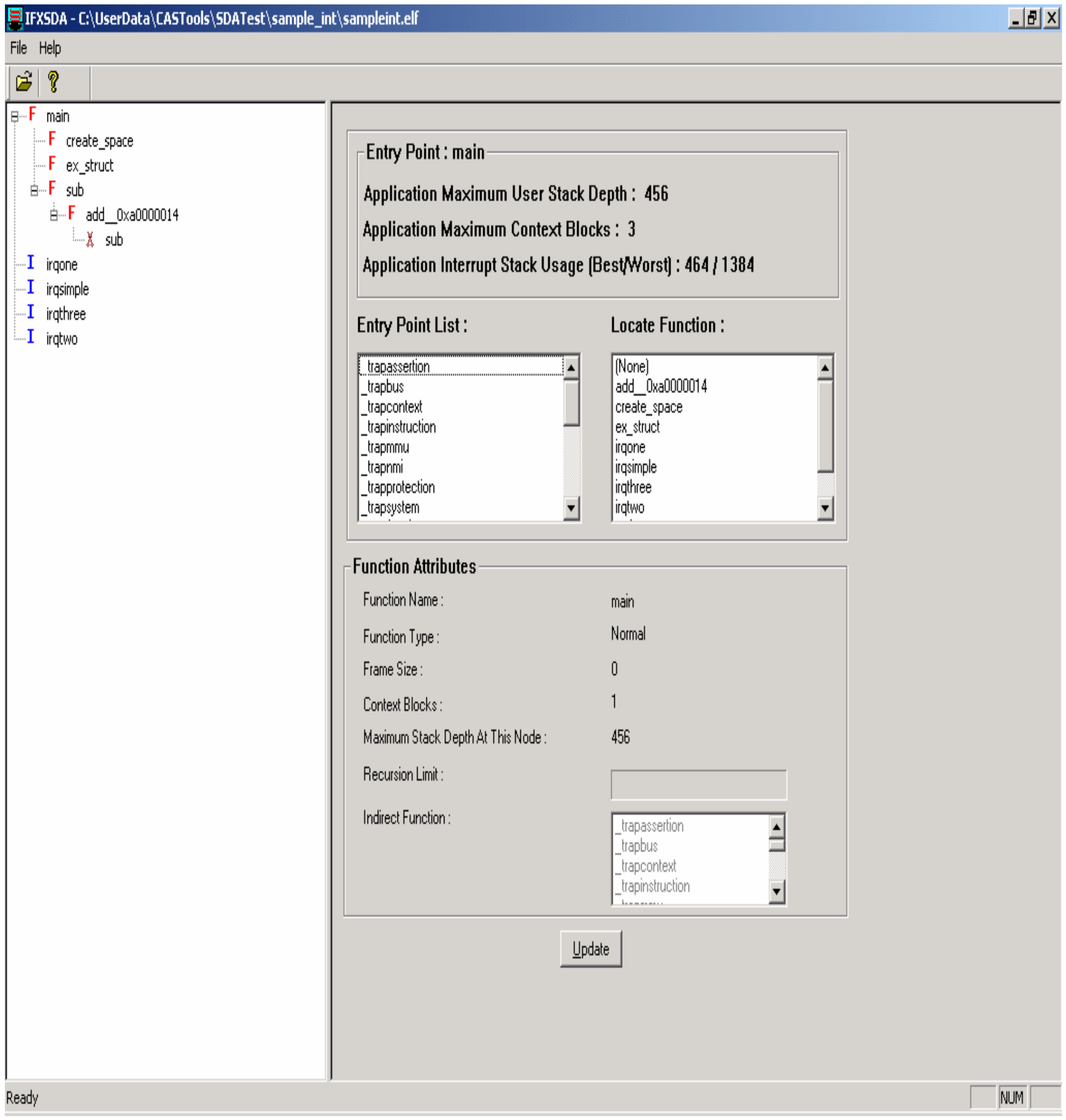
There are two ways to specify the command options to the tool. In both the cases the executable file name should be provided as the first argument to the tool except –h and –v option.

1. Specify all the command options directly on the command line.
   IFXSDA <ELF File> [Command option1] [Command option2] … [Command option n] <-nogui>
   -nogui option instructs the tool to run command line mode.
   E.g. IFXSDA sample.elf –I main fact –r fact 5 –i –o result.txt -nogui

2. Specify the command options from command file.
   IFXSDA <ELF File> -c <Command file name> <-nogui>
   If command file is provided on the command line then all other options specified on the command line are ignored. Tool starts reading the command file and get the command options from it. Each option is given on a new line in the command file.
   -nogui option instructs the tool to run command line mode.
   e.g. IFXSDA sample.elf –c sample.cmd -nogui
   Contents of sample.cmd are
   -I main fact
   -r fact 5
   –i

- ***Invoking IFXSDA in Graphical User Interface mode***

  If the IFXSDA is invoked without **–nogui** option, the tool opens a window to enable the user to interactively work with the tool. The GUI mode is more expressive and interactive than the command line mode. The user can open the tool in GUI mode by clicking the following link provided to the executable.
  Start->Programs->IFXSDA-> IFX Stack Depth Analyzer



IFXSDA - C:\UserData\CASTools\SDATest\sample_int\sampleint.elf

```
F main
    F create_space
    F ex_struct
    F sub
        F add__0xa0000014
            X sub
I irqone
I irqsimple
I irqthree
I irqtwo
```

Entry Point : main

Application Maximum User Stack Depth : 456

Application Maximum Context Blocks : 3

Application Interrupt Stack Usage (Best/Worst) : 464 / 1384

Entry Point List :	Locate Function :
_trapassertion	(None)
_trapbus	add__0xa0000014
_trapcontext	create_space
_trapinstruction	ex_struct
_trapmmu	irqone
_trapnmi	irqsimple
_trapprotection	irqthree
_trapsystem	irqtwo

Function Attributes

Function Name :	main
Function Type :	Normal
Frame Size :	0
Context Blocks :	1
Maximum Stack Depth At This Node :	456
Recursion Limit :	
Indirect Function :	_trapassertion _trapbus _trapcontext _trapinstruction

A User View of IFXSDA Tool.

Operating with the tool:

- **Invoking the help about the tool**
  This can be achieved by three ways.
  The quickest way to press function F1 key.
  Click '?' button on the toolbar.
  Click *Help->IFXSDA* Help button in the menu bar.

- **Opening a file :**
  This can be achieved by clicking Menu **File->Open** command or direct **Open** command on the Tool bar. Once this command is successfully executed, the call tree is displayed in the left pane of the window and the rest of the information is shown on the right pane of window. By default the call tree will start from "main" for applications whose entry point is "main". Otherwise the call tree is not displayed until the user chooses a valid entry point.

- **Changing Function attributes :**
  For every function in the tree there is an icon attached to it. This specifies one of the following qualifiers of the function.

  Usual functions types:

F	Normal node without any recursive or indirect call. The function attribute "Interrupt" will be set to "No" for nodes which are not interrupt nodes.
F_i	Indirect node. Specify the desired name for this node in the Indirect Function list box.
F^r	Recursive node. Specify the Recursion limit for this node in the Recursion limit edit box.
F^r_i	Recursive indirect node. User can do the above two operations for this node.
I	Interrupt node. The function attribute "Interrupt" will be set to "Yes" for this node.

  Clicking on a node in the call tree shows its corresponding attributes in "Function Attributes" panel.

- **Locating a function in the call tree :**
  This feature provides a way to the user to find out all the calls of a particular function in the call tree. The user has to select a function name from the Locate Function list and click **Update** to view all of the related function calls in the Call Tree. Each function node with the corresponding name is displayed with one of the following attributes:

Normal node without any recursive or indirect call.

Indirect node. Still user can specify the desired name for this node in the Indirect Function list box.

Recursive node. Specify the Recursion limit for this node in the Recursion limit edit box. For editing this value Set Enable Recursion combo box value to “Yes”.

Recursive indirect node. User can do the above two operations for this node.

To navigate forward-backward in the tree, click on the left pane of the main window where the call tree is situated. Use Function key ***F3***, ***Shift+F3*** to navigate forward, backward respectively in the tree. The focus will move forward on the next locate item in the tree with one key press of ***F3***. The focus will move backward on the previous locate item in the tree with one key press of **Shift+*F3***.

- **Changing the entry point :**
  The user can select the entry point of the application at any point of the time during which the application is loaded in the tool. By default the call tree will start from “main” for applications whose entry point is “main”. Otherwise the call tree is not displayed until the user chooses a valid entry point. Once the user changes the entry point the “**Application Maximum User/Interrupt Stack Depth**” value gets modified accordingly after clicking the **Update** button.

- **Exiting from the tool**
  To exit from the tool Press Ctrl+Q or menu button File->Exit

- **Using *Most recent file* list**
  The tool records 4 most recently used file names. This feature enables use to quickly open recently opened file without knowing its path. Whenever the tool is installed on the machine this list will be empty. With every file open this list gets updated with new file names.
  The tool records most recently file names in the *File* menu tab. The list will be placed in between menu buttons ‘*open’* and ‘exit’.

- **Assigning indirect call to a function**
  Click on the indirect function item (caller function) in the call tree pane. Then click a called function from the “*Indirect Function*” list. Click update button which will modify according to the action.

- **Modifying the recursion limit of a function**
  Click on the recursive function item in the call tree pane. Then enter the recursion limit for the function in “*Recursion Limit”* edit box. Click update button which will modify according to the action.

• Analyzing output of IFXSDA in Command line mode

In the command line mode the tool outputs all its tool messages to console. The tool –o command line option can be used to direct this output to a file.

Take an example of an ex1.elf which includes following calls
main calls F1
F1 calls F2
F2 calls F3 and F4
F3 calls F4 and F5
For ex1.elf file tool generates the call tree as below:

```
Main(16,60)
              -> F1(16,40) -> F2(8,24) -> F3(8,16) -> F4(8,8)
              -> F1(16,44) -> F2(8,28) -> F3(8,20) -> F5(12,12)
              -> F1(16,32) -> F2(8,16) -> F4(8,8)

Maximum user stack depth from the entry point "main": 60

Call sequence for maximum user stack depth:
main->F1->F2->F3->F5

Interrupt stack depth of the application (Best/Worst): 16/24

Maximum number of CSA blocks used from the entry point "main":
5

Call sequence for maximum number of CSA blocks used:
main->F1->F2->F3->F4
```

Note: *The call sequence for maximum user/interrupt stack depth and the call sequence for maximum number of CSA blocks used can be different for a given application.*

The tool has generated all possible sequences from the entry point main. Each called node has pair of attributes attached to it enclosed by '(', ')'. The first attribute represents the individual frame size of that function and second attribute represents maximum stack depth from that function call. Maximum stack depth for a function is sum of its own frame size and Maximum of stack depths of its called functions. Tool has shown the Maximum stack depth from the entry point (e.g.: 60 in the above case). Tool has shown the Maximum sequence from the entry point.

Indirect recursion Identification:

All the indirect recursion nodes in the call tree are indicated with (IR) attribute.

e.g.

```
main
          -> max(8,16) -> xfact(8,8) -> max(IR)
```

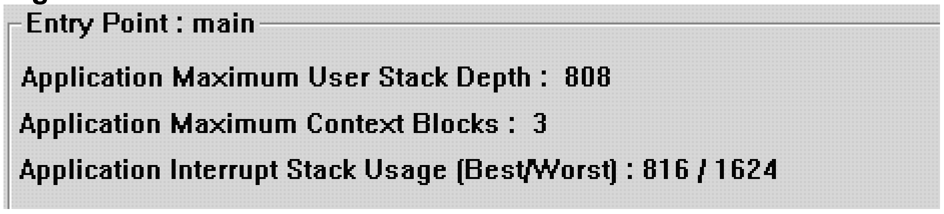
In the above case, the function “max” is calling “xfact” and “xfact” in-turn calls “max” again. This sequence is a candidate for indirect recursion.
If user has given “–i” option on the command line, the tool shows all indirect recursions found in the input file. Indirect node (attributed with IR) is not included in the stack depth calculation. In above case max (IR) node doest not take part in Stack depth calculation as this is a dummy node just to indicate user that there is an indirect recursion in the sequence.

Indirect Function:
All the Indirect functions are given the name “nI” (where n can be any number) and I to indicate that it is an indirect function.

- ***Analyzing the output of IFXSDA in Graphical user interface mode***

  In GUI mode the tool generates its output on the text boxes which are assigned for following individual attributes.

  E.g.:



Entry Point : main

Application Maximum User Stack Depth : 808

Application Maximum Context Blocks : 3

Application Interrupt Stack Usage (Best/Worst) : 816 / 1624

The user should analyze the above output as follows:

- “main” is the selected/default entry point of the application.
- “main” is a normal function (NOT an interrupt) and user stack will be used in this function. The maximum user stack depth of the application from “main” is 808. If the selected entry point is an interrupt then the output will be “Application Maximum Interrupt Stack Depth”.
- Maximum number of context blocks required for the application from the selected entry point (i.e., main) is 3.
- The best and worst case values of interrupt stack usage of the application are 816 and 1624 respectively.

There are few function related attributes. They are listed bellow.
Function Name:
This attribute indicates the name of currently selected function.

Interrupt:
This attribute indicates whether the function is an interrupt or not.

Frame Size:
This attribute indicates the frame size of currently selected function.

Context Blocks:
This attribute indicates the number of context blocks used by this function.

Maximum Stack Depth At This Node:
This attribute for a function indicates sum of its own frame size and maximum of stack depths of its called functions.

Indirect recursion Identification:
In the GUI mode indirect recursion identification is by default enabled. Tool shows the indirect recursion with the following types of icon. For the indirect recursive node the Frame size and max stack depth information is ignored.

Indicates an indirect recursive node.

Indicates an indirect recursive node which is also an indirect function.

Indicates a recursive node which is also a recursive function.

Indicates a recursive and indirect node which is also recursive and indirect function.

Using the 'Entry Point', 'Locate function' and 'Indirect Function' list boxes

- Jump to a specific function name in the list box:
    1. Set the focus on the enabled list box. Note that if the list box is disabled, the function can't be located. Setting of focus can be done using keyboard (using TAB key) or mouse.
    2. Go to the first function name entry in the list box.
    3. Type the function name to be searched until it sets a focus on the desired function. If after typing the complete name of the function, still the tool doesn't set focus on the desired function, it means that the function name is not present in the list. In the most of the cases the match is found after typing first few characters of the function.

- View the complete name of functions longer than the list box width:
    1. Select a function name to be viewed from the enabled list box.
    2. Now just move the mouse pointer over the list box. The complete name of the selected function will be seen in a tool tip box near the mouse pointer.

4. Appendix A

Error codes:

Return Value	Internal Error Code	Description
01	ERR_NO_ENTRY_POINT	User has not given the entry point function for –f.
02	ERR_NUM_VALUE_R	The second parameter given for –r option is not a valid numeric value.
03	ERR_NO_MAX_LIMIT_R	User has not specified the numeric Max recursion limit value for –r option.
04	ERR_NO_FUNC_NAME_R	User has not given a function name for –r option.
05	ERR_NO_CALLED_FUNC_I	Called function is not specified for –l option.
06	ERR_NO_CALLER_FUNC_I	Caller function is not specified for –l option.
07	ERR_NO_OUT_FILE_NAME_O	Out file is not given for –o option.
08	ERR_INVALID_OPTION	This error is occurred when an invalid option is encountered. There can be some typo mistake in the command file or command line.
09	ERR_UNEXPECTED_CMD_ARGUMENT	Possible cause of this error is the elf file name is specified at wrong position. e.g. IFXSDA –f max ex.elf . In this example ex.elf is wrongly placed.
10	ERR_NO_ELF_FILE_NAME	User has not specified the elf file. User has to pass the .elf file name as the first parameter to the tool.
11	ERR_OPEN_CMD_FILE	There is error opening the command file. Verify that the file is present at the specified path.
12	ERR_UNEXPECTED_TOKEN	An unexpected token has found in the command file.
13	ERR_ID_EXPECTED	Encountered a token which is not identifier where identifier is expected. E.g. –f 0main. In this case 0main is not a valid identifier.
14	ERR_RESERVED_1	Reserved error number for future.

15	ERR_CMD_FILE_NAME_EXPECTED	Command file is not specified for the –c option. e.g. –c –f fact. In this example –c is expecting the file name but has encountered –f option instead.
16	ERR_CANT_CREATE_OUTPUT_FILE	There is a error creating the output file required by the tool internally. Make sure that disk is not full.
17	ERR_OPTION_NOT_ELFFILE	Unexpected token is found in the command file. e.g following command found in the command file on a new line. Main –f max. Rule is each command start with '-' in the command file.
18	ERR_INTERNAL_ERROR	There is a internal error occurred in the tool.
19	ERROR_EXEC_CALLER_NOT_FOUND	A caller function not found in the ELF file. E.g for a command –r max fact if max is not present in the elf file as a caller then this error will be generated.
20	ERR_EXEC_NON_RECURISIVE	The function specified with –r option is not recursive. e.g. –r fact 4. If fact is not found the recursive function then this error will be generated.
21	ERROR_EXEC_CALEE_NOT_FOUND	A called function is not found.
22	ERR_EXEC_NON_INDIRECT_FUNC	The caller in –I command has no indirect function.
23	ERR_NOT_VALID_ELF	The file given to a tool is not a valid elf file.
24	ERR_NOT_TRICORE	The file has valid elf file format but is not a valid tricore elf file.
25	ERR_UNABLE_READER	Can't create ELF reader
26	ERR_RESERVED_2	Reserved error number for future.
27	ERR_UNABLE_SYMTAB	Unable to read the Symbol Table Properly
28	ERR_UNABLE_READ_FILE	Unable to read Data from File.
29	ERR_UNABLE_OPEN	Unable to open the elf file
30	ERR_MEM_ALLOC	Unable to allocate memory.
31	ERR_SET_CORRECT_ENTRYPOINT	User has to set the correct entry point. The user may have given a wrong entry point to the tool.
32	ERR_WRITING_OUT_FILE	There is error wring to output file specified by –o option. The tool catches the reason of problem from

		Operating system and displays on the screen.
33	ERR_NOT_VALID_RECURSION_RANGE	The recursion limit specified for a function is not in the valid range of 0 to 9999999.
34	ERR_UNABLE_SYM_ADDR_ELF	Unable to find a symbol in symbol table with address yyyyy in the file
35	WARN_MULTIPLE_C_OPTIONS	Multiple –c options are encountered on the command line. In the case tool takes the first option.
36	WARN_MULTIPLE_F_OPTIONS	Multiple –f options are encountered on the command line/command file. In the case tool takes the first option.
37	WARN_MULTIPLE_R_OPTIONS	Multiple –r options are encountered on the command line/command file. In the case tool takes the first option.
38	WARN_MULTIPLE_I_OPTIONS	Multiple –I options are encountered on the command line/command file. In the case tool takes the first option.
39	WARN_MULTIPLE_O_OPTIONS	Multiple –o options are encountered on the command line/command file. In the case tool takes the first option.
40	ERR_STK_FN	Internal system Error in module STK_FN
41	ERR_GET_CLIST	Internal system Error in module GET_CLIST
42	ERR_STK_INT	Internal system Error in module STK_INT
43	ERR_GET_INT	Internal system Error in module GET_INT
44	ERR_PRF_UNKN	Internal system Error in module PRF_UNKN
45	ERR_NO_SECTION_READER	Unable to Create Section.
46	ERR_UNABLE_LOAD_ELF	Unable to Load the Elf File.
99	ERR_FATAL	Fatal error has come with the tool. Please report this bug to IFXSDA team.

5. Appendix B

MS-DOS file name convention

1. *Long path name gets short form.*
e.g. *c:\MyDocuments\abc.txt* should be given as *c:\MyDocu~1\abc.txt*
The last two characters will be of given as ~1, ~2, ~3 etc depending on the file names. Long files names are the one with file name length more than 8 characters.
2. *Spaces are not allowed in between the MS-DOS file name.*
e.g. *c:\Imp\Imp File.txt* should be given as *c:\Progra~1\abc.txt*
3. If there are two files with first 8 characters files will be given different numbers.
e.g. *c:\Impdata1* and *c:\Impdata2* should be given as *c:\Impdat~1* and *c:\Impdat~2* respectively.
4. All the above rules apply for the file names as well as directory names.

6. Appendix C (TASKING Compiler Project Settings for Projects containing static functions)

The following project settings should be set in TASKING compiler for projects containing static functions to generate ELF file.

(1) Select 'project | project options' MENU
(2) Click 'linker' PAGE
(3) Click 'Map File' PAGE
(4) Tick 'module local options' OPTION
(5) Disable both compiler and assembler debug information in 'C Compiler-Debug Information' PAGE and 'Assembler-Debug Information' PAGE respectively.

7. Appendix D (Identification of static functions)

Following Appendix C, IFXSDA has implemented the following for correctly identifying static functions with same names across modules.

Consider the following example containing a static function with same name in both modules (source file).

e.g.:

```
sample1.c
~~~~~~~~~~

static int stat ()
{
 //some code
}

int main ()
{
  stat();
  return 0;
}
```

```
sample2.c
~~~~~~~~

static int stat ()
{
//some code
}
```

As function names may not be unique for static functions, IFXSDA displays static functions as the following:

StaticFunctionName__Address

Where, “StaticFunctionName” refers to the function name of the static function and “Address” refers to the unique hexadecimal address of the function.

NOTE: The address of a function is unique for all types of function.

In the above example, in sample1.c, if function *stat* () is situated at address 0xa00000b8, the tool displays the function as:

stat__0xa00000b8

All the occurrences of *stat* will be replaced by stat__0xa00000b8.

NOTE: The User has to provide static function name in “StaticFunctionName__Address” format while providing function names as input to the tool to indicate the correct static function. (For e.g. in command line, while specifying the indirect function’s name information).