SIGNATURE ANALYSIS GENERATOR (IFXSAG)
USER MANUAL

Doc ID: IFXSAG-UM-1.6
Version 1.6
Release Date: Dec 8th 2006

Edition 8th Dec 2006
Published by Infineon Technologies AG,
D-81726 Munich, Germany

LEGAL DISCLAIMER:

THE INFORMATION GIVEN IN THIS MANUAL IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS MANUAL MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS MANUAL.

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.

CONTENTS

1. Introduction

- ***Overview***

  The Signature Analysis Generator (SAG) Tool generates the CRC checksum for the data in a linked ELF format file before downloading. The tool can also write this checksum value in the Alternate Boot Mode (ABM) Headers for the TriCore series of Microcontrollers and append it to the same file.

- ***Resource Requirements***

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

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

  Software
  - The host PC should run Windows2000/XP.
  - The TASKING EDE or the TASKING Tool Chain for TriCore for generating the ELF file for the intended application.
  - PLS UDE for downloading the ELF file onto target's flash.

- ***Assumptions and Limitations***

  The tool has following assumptions and limitations:
  1. The input ELF file should be generated by TASKING Tool Chain for TriCore.
  2. There should be no gaps in sections in the data being read for generating checksum.
  3. The ABM header data is as per the processor Hardware Manual.
  4. **The addresses (in the ELF file) used as input parameters to the tool should have their section type as "PROGBITS".**

  Since the usage of this tool involves accessing/modifying ELF file addresses and symbols, it is assumed that the user has basic ELF format knowledge.

2. Using IFXSAG

IFXSAG is a command line utility for generation of CRC checksum for a range of data in an ELF file. It can also generate the ABM header and append it to the file for downloading the application to target.

- ***Invoking IFXSAG***

The tool assumes two input files: an ELF file and a command file. The format of invocation of IFXSAG is:
IFXSAG InputELFfilename Commandfilename <options>

Where,
The **InputELFfilename** is the executable file name for whose address ranges, the CRC is being calculated.

The **Commandfilename** is the name of the file containing the command file options and parameters to the tool.

For the ease of batch programming, the program returns different error codes, which are explained in Appendix B.

The return value '0' indicates no Errors.

- ***Command Line Options***

The command line options are NOT case sensitive. The various command line options to the tool are:

Options	Description
-c ZZZZ	Build a total checksum and write to address ZZZZ. i.e., a total checksum is built over all single checksums computed for the address ranges specified in the command file and write to address ZZZZ only. Checksums computed for address ranges specified in the command file will NOT be written in their respective destination addresses.
-cx ZZZZ	Build a checksum over all given address ranges and write to address ZZZZ only. There is no calculation of single checksums.
-k	Without this option, the options –c, -cx do not write to ELF file, but to IFXSAG.INF.
-p <target>	Specify the target. Targets supported are TC1766, TC1796. Tool uses the corresponding polynomial and CRC computation algorithm depending on the target selected. TC1766 is the default target.

Example: -p TC1796

-write_abm_header Inform the tool to write ABM headers in the ELF file. By default, these are written in the image file and configured to internal flash memory corresponding to the target. ABM headers contain only a single memory range. If multiple address ranges are specified in the command file, then the first memory range will be considered and the rest will be ignored. If this option is used, then the tool does not write CRC value at the address specified in command file.

-flash_internal | -flash_internal_non_cached
Configures ABM headers to non-cached internal flash memory of the target specified. Should be used along with "write_abm_header" option, otherwise, this will be ignored.

-flash_external | -flash_external_non_cached
Configures ABM headers to non-cached external flash memory of the target specified. Should be used along with "write_abm_header" option, otherwise, this will be ignored.

NOTE: TC1766 does not have external flash. Hence, this option is not applicable for TC1766.

-flash_internal_cached
Configures ABM headers to cached internal flash memory of the target specified. Should be used along with "write_abm_header" option, otherwise, this will be ignored.

-flash_external_cached
Configures ABM headers to cached external flash memory of the target specified. Should be used along with "write_abm_header" option, otherwise, this will be ignored.

NOTE: TC1766 does not have external flash. Hence, this option is not applicable for TC1766.

-o <abm_header_file_name>
Specify the output file name with ABM headers. Should be used along with -write_abm_header option, otherwise, this will be ignored.

-s No output to stdout. All tool messages are written to the file IFXSAG.INF in the current directory. If –s is not specified, output will be to the stdout.

-h Display the usage of the tool.

- ***Command File Options:***

A command file contains set of commands for the tool and on the basis of these commands the tool calculates the Checksum for a range of data. The command file options are NOT case sensitive. The format for the commands in the command file is:

<Options> write_address start_address – end_address [InitSeedValue]
Where,

write_address : The destination address within the ELF file, where the computed CRC has to be written.

start_address : The start address within the ELF file from where the data has to be considered for computing the CRC.

end-address : The end address (inclusive) within the ELF file till where the data has to be considered for computing the CRC.

InitSeedValue : **Optional** initial value that has to be assigned to Result Register of the processor selected while computing the CRC. **NOT** to be used with address ranges specified for computing CRC to be written in ABM headers.

Options:

-e|E	Exchange the data (32 bit) of a specified address (label or fixed address). This is a mandatory option.
-a|A	Address range is given directly in hexadecimal format.
-s|S	Address range is given in symbol format.
-v|V	This is an optional command for the –e|E option. It denotes that the destination address within the ELF file where the CRC-checksum will be written is given directly in hexadecimal format. Otherwise, the destination address is specified as a label. In general, the hexadecimal format of addresses must comply with the C standard.
-w|W	This option directs the tool to write first ABM header for the file for a particular address range. In case of multiple entries of commands with ‘w’ option, the first one is selected. In case, the ‘–write_abm_header’ option is not specified in the command line, the command with –w is ignored and no CRC is calculated for it.
-w2|W2	This option directs the tool to write second ABM header for the file for a particular address range. In case of multiple entries of commands with ‘w2’ option, the first one is selected.

In case, the '–write_abm_header' option is not specified in the command line, the command with –w2 is ignored and no CRC is calculated for it.

-i|I Initial seed value (to be specified in hexadecimal format) that has to be assigned to Result Register of the processor selected while computing the CRC. **NOT** to be used with '-w|W' option.

- ***Command File Options combinations:***

-ea <sym_vname> <addr_bgn> - <addr_end>
-eav <addr_vname> <addr_bgn> - <addr_end>
-es <sym_vname> <sym_addr_bgn> - <sym_addr_end>
-esv <addr_vname> <sym_addr_bgn> - <sym_addr_end>

-eai <sym_vname> <addr_bgn> - <addr_end> InitValue
-eavi <addr_vname> <addr_bgn> - <addr_end> InitValue
-esi <sym_vname> <sym_addr_bgn> - <sym_addr_end> InitValue
-esvi <addr_vname> <sym_addr_bgn> - <sym_addr_end> InitValue

Options valid only with –write_abm_header:

The following are the commands to specify the address range for the first ABM header.

-eaw <sym_vname> <addr_bgn> - <addr_end>
-eavw <addr_vname> <addr_bgn> - <addr_end>
-esw <sym_vname> <sym_addr_bgn> - <sym_addr_end>
-esvw <addr_vname> <sym_addr_bgn> - <sym_addr_end>

The following are the commands to specify the address range for the second ABM header.

-eaw2 <sym_vname> <addr_bgn> - <addr_end> <alt_start_address>
-eavw2 <addr_vname> <addr_bgn> - <addr_end> <alt_start_address>
-esw2 <sym_vname> <sym_addr_bgn> - <sym_addr_end>
<alt_start_address>
-esvw2 <addr_vname> <sym_addr_bgn> - <sym_addr_end>
<alt_start_address>

Where:

Addr_bgn	start address of Signature Analysis in hexadecimal format
Addr_end	end address of Signature Analysis in hexadecimal format
Sym_addr_bgn	symbol of start address of Signature Analysis
Sym_addr_end	symbol of end address of Signature Analysis
Sym_vname	variable name definition for storing the 32 bit CRC-Checksum
Addr_vname	hexadecimal address of reserved variable for storing the

	32 bit CRC-Checksum
InitValue	The initial value of Memory Checker Result Register
alt_start_address	The alternate start address of the application

NOTE: In case, the '–write_abm_header' option is specified in the command line and if address range is NOT specified in the command file for writing ABM header (i.e., no eaw, eaw2, eavw, eavw2, esw, esw2, esvw, esvw2) then the first address range will be considered for writing ABM headers.

If address range is specified only for second ABM header then the same data including the start address will be considered for the first ABM header and vice versa.

IFXSAG always reads ((end_address – start_address) + 4) bytes of data for each address range, starting from "start_address". IFXSAG always reads a minimum of one word (four bytes) of data. It is done in accordance to the ABM structure of TriCore. For more info on ABM structure, refer Appendix A.

E.g.: If the address range specified in the command file is "-eav 0xa0000014 0xa000000c - 0xa000000c", then four bytes of data will be read starting from the address "0xa000000c".

The checksum generated by IFXSAG will be one word (four bytes) in size.

Appendix A

Alternate Boot Mode (ABM)

For understanding the CRC procedure an idea of the ABM is necessary. The main feature of the Alternate Boot Mode is that it will only start program execution if valid code has been detected in Program Flash or in external memory (in case of external start). The validation of the code is done by a checksum calculated over one or two user-defined Flash memory ranges. If both checksums fail (program memory is disturbed or not loaded), a bootstrap loader is executed instead of program execution.

Basically the ABM consists of three steps:
Step 1: Access first ABM header and perform first memory check for valid code in the on-chip or external Flash memory and execute it if it is valid (normal program start at start address defined in first ABM header).

Step 2: If first check fails then access second ABM header and perform second memory check for valid code and execute it if it is true (program start at alternate start address in second memory part).

Step 3: If the second check also fails then execute the Bootstrap Loader for ABM.

The Alternate Boot Mode defines just a check algorithm and the location and structure of the two ABM headers. The user defines all parameters in the ABM headers, including – the program start address – the start address of the memory range which has to be checked with CRC checksum– the end address of the memory range which has to be checked with CRC checksum– the checksums for the checked memory range and for the ABM header. Just one true header check is sufficient to start user mode! The program start address in ABM mode always depends on the results of the memory check. It is the user's responsibility that reasonable code is located at the alternate start address. The ABM mode can also be combined with Flash read protection. As long as the ABM is selected it is performed with every reset.

5.1.1.1 Locations of ABM Header

The main (preferred) ABM header is located in first half of internal or external Flash with following addresses:

	Internal Flash:	**External Flash:**
– Base address:	8001 FFE0H	8100 FFE0H
– End address:	8001 FFFFH	8100 FFFFH

The second (alternative) ABM header is located in second part of internal or external Flash with following addresses:

– Base address:	8003 FFE0H	8108 FFE0H
– End address:	8003 FFFFH	8108 FFFFH

Note: Also segment address AH may be used for Flash or external memory.

Structure and Use of ABM Headers
The following table shows the structure of the ABM headers. All numbers represent hex data. The sizes are defined in bytes. .

TABLE A.1 Structure of ABM Headers

Offset	Size	Definition	Function	Example
00	4	addr(start of code)	Program Start Address	A000 0000
04	4	"DEADBEEF"$_H$	ID String; Header Confirmation	DEADBEEF
08	4	addr(checksum start)	Start address of memory range to be checked(32 bit aligned)	A000 0000
0C	4	addr(checksum end)	End address of memory range to be checked(last word addr)	A000 0400
10	4	$CRC32_{range}$	CRC result for checked range	3678 6677
14	4	$CRC32_{range}$	Inversed CRC Result	C987 9988
18	4	$CRC32_{head}$	CRC Result of Header(offs. 00-17)	1818 1818
1C	4	$CRC32_{head}$	Inversed CRC Result of Header	E7E7 E7E7

The CRC generation is hardware supported, using the memory checker registers in the Memory Checker Module of the DMA controller. Before a checksum is generated; the 32-bit checker result register is preloaded with the value FFFF FFFFH. After CRC generation, the CRC result must be identical to the result as defined in the ABM header. If both result comparisons are true, program execution will be started at address defined in first word of ABM header. If CRC32head or CRC32range do not match the predefined values, a new CRC checking sequence is started with the second ABM header.

Appendix B

Error Codes

Return Value	Error Code	Description
0	NO_ERROR	The tool returns with no errors.
1	ERR_NO_ADDR_CX	Address is not specified for -cx option.
2	ERR_NO_ADDR_C	Address is not specified for -c option.
3	ERR_ADDR_NUM_CX	Address should be in hexadecimal format for -cx option.
4	ERR_ADDR_NUM_C	Address should be in hexadecimal format for -c option.
5	ERR_UNRECOGNISED_TOKEN	Unrecognized token in the command file.
6	ERR_NO_TARGET_P	Target value not specified for -p option.
7	ERR_INVALID_OPTION	Invalid option specified on the command line.
8	ERR_NO_ELF_FILE	ELF file not specified on the command line
9	ERR_NO_CMD_FILE	Command file not specified on the command line
10	ERR_INVALID_TARGET_TYPE	Invalid value for the target type given by –p option
11	ERR_OPEN_CMD_FILE	Unable to open Command file for reading.
12	ERR_UNABLE_PERFORM _OPERATION	The tool is unable to perform the required operation.
13	ERR_INVALID_DATA	Invalid data
14	ERR_UNABLE_READER	Unable to open the reader
15	ERR_UNABLE_OPEN_ELF	Unable to open ELF file for Reading/Writing
16	ERR_INVALID_TRICORE	ELF file not a valid Tricore file
17	ERR_NOT_VALID_ELF	Not a valid ELF format file
18	ERR_GAP_SECTIONS	Gaps within Section
19	ERR_INVALID_ADDRESS	Address does not fall into any of the sections.
20	ERR_UNABLE_WRITE	Unable to write to the ELF file
21	ERR_INVALID_ADD_RANGE	Invalid address range specified for reading. Please check whether the start address is less than the end address in all address ranges specified.
22	ERR_MEMORY	Insufficient memory for running the application. Please check whether the start address is less than the end address in all address ranges specified.
23	ERR_PRCS_OPTS	Internal System Error
24	ERR_PRCS_CMDF	Internal System Error
25	ERR_PRCS_CMDN	Internal System Error
26	ERR_PRCS_UNKN	Internal System Error
27	ERR_1766_NO_EXTERNAL_FL ASH	External flash is not available on the target

		TC1766.
28	ERR_UNABLE_SYM_ELF	Unable to find symbol in ELF file
29	ERR_OPEN_ELF_FILE	Unable to open ELF file
30	ERR_ELF_READ_ONLY	ELF file is a read-only file
31	ERR_INTERNAL	Internal Error in processing ABM headers
32	ERR_INVALID_WRITE_ADDRESS	Specified address is an invalid write address
33	ERR_NO_SPACE_IN_SECTION	No enough space in the section to write checksum at destination address XXXX. Four bytes required.
34	WAR_IGNORE_O_OPTION	Warning: Ignoring –o option. This option is valid only when –write_abm_headers is specified
35	WAR_IGNORE_FLASH_MEM_OPTION	Warning: Ignoring –flash_internal or –flash_external option. This option is valid only when –write_abm_headers is specified
36	WARN_WRITE_ADDR	Warning: Write address falls within the read address range. This could possibly cause the hardware to generate different Checksum for the same range.
37	WARN_ADDR_32_BIT	Address is not 32-bit aligned.