Quality of HAL libraries

T.O.M.A.S Team





Goal of this part 3

Demonstrate basic information about code analysis techniques and its tools

Present ST practices used for HAL and LL libraries quality verification



Program analysis

Program analysis is the process of analyzing the behavior of software programs to ensure quality attributes such as:

- correctness,
- reliability
- robustness,
- safety,

Program analysis can be performed:

- during code review
- without executing the program (static program analysis),
- during runtime (dynamic program analysis),
- combination of all above.





Static program analysis

Static program analysis is the analysis of computer software that is performed without executing a program.

Two popular static program analysis methods:

- **Control-flow**: obtaining information about which functions can be called at various points during the execution of a program.
- Data-flow: gathering information about the values at each point of the program and how they change over time.







Dynamic program analysis

Dynamic program analysis is analysis performed on executing program.

Popular dynamic program analysis methods:

• **Testing**: executing the program with a given input and evaluating its behavior and the produced output.



- **Monitoring**: records and logs different kinds of information about the program such as resource usage, events and interaction
- **Program slicing**: reducing the program to the minimum form that still produces the selected behavior.





Program analysis tools

• Free software released under the terms of the GNU General Public License

• **Splint** - tool for statically checking C programs for security vulnerabilities and coding mistakes.

Splint's output:

life.auamented

Variable c used before definition Suspected infinite loop. No value used in loop test (c) is modified by test or loop body. Assignment of int to char: c = getchar()

- Gcov and Gprof source code coverage analysis and statement-by-statement profiling tools. Gcov provides the following information:
 - How often each line of code executes
 - What lines of code are actually executed •
 - How much computing time each section of code uses •

• LCOV - a more user-friendly graphical visualization f the Gcov output.



Generaled by: LCOV version 1.9

else 1 | Keturn(0)]

Program analysis tools

Commercial software

- **CodeSonar** static analysis tool from Grammatech, which identifies bugs that can result in system crashes, unexpected behavior, and security breaches.
 - Like a compiler, CodeSonar builds code using build environment, but instead of creating object code, CodeSonar creates an abstract model of entire program.
 - From the derived model, CodeSonar's symbolic execution engine explores program paths, reasoning about program variables and how they relate.



Tool used by ST to achieve high quality of Cube HAL







STM32Cube High Quality Process Maintenance Releases Policy 10

- The STM32Cube FW package is maintained regularly through:
 - Full release (marking x.y.0)
 - Patch release (marking x.y.z)
- The updater tool available with STM32CubeMX PC tool allows automatic notification and download of new STM32Cube release or patch





STM32Cube High Quality Process Static code analysis

No properties have been set.

warning details.

This code writes past the end of the buffer pointed to by strcpy:parameter 1

CODESONAR Search code in this analysis

Static Code analyze

life.auamentec

- Home > gnuchess-x86 > gnuchess-x86 analysis 1 > Warning 71.85 • HAL going through Code Analysis Tool: Buffer Overrun at gnuchess 1st:39958 CodeSonar Industry reference tool Jump to warning location identifying vulnerabilities at compilation show Events | Options return_append_str (gnuchess.lst) time
 - Show Events | Options 39950 text:08065900 loc 8065900 A 39951 text:08065900 mov dword [esp],edi A 39952 text:08065903 call thunk .strlen A 39953 text:08065908 mov dword [esp],eax A 39954 . text:0806590B call thunk .malloc A Event 15: malloc () returns the address of a new object. This points to the buffer that will be overrun later 🔺 💌 bido A 39955 text:08065910 dword [esp+4],edi mov A 39956 text:08065914 lea ebx. [eax+1] A 39957 text:08065917 dword [esp],ebx mov A 39958 text:0806591A call thunk .strcpy Buffer Overrun

for

edit properties

Report available on demand

Figure 4: Warnings by File



Example of the report

Search 2 Advanced Search

Text | XML | ReML | Visible Warnings: active



-

STM32Cube High Quality Process Dynamic code analysis and validation report

ST Dynamic validation test and Validation report on CUBE HAL

- Test are run by cube HAL i.e per sub family.
- One validation test and report per peripheral / function and package in Doxygen format
- Test are run (dynamic) in semi automatic way and use ST evaluation boards. |STM32L4xx_HAL_Validation_Rep

Modules

Main Page

Files

Data Structures

Directories



Hide Locate Back Forward Stop Refres	sh Ha	n D ome	 Font	Print	Deptions
Contents Index Search Favorites Fi	Firmware Reference				
C STM32F4xx HAL Drivers C STM32F4xx HAL Driverv1.0.0RC4 Tests C ADC HAL_Driverv1.0.0RC4 Tests C Related Pages C Modules C Modules C Do HAL ADC Extension Test Form	CorexM CMSIS drivers V3.20 SVN Revision 20 STM32F4xx CMSIS drivers V2.0.0RC6 SVN Revision 206 STM32F4xx HAL drivers V1.0.0RC4 SVN Revision 2431				
□ (□) FAL_ADC_Extrative_rest_rund Value □ (□) Functions □ HAL_ADC_E_test01 □ HAL_ADC_E_test02 □ HAL_ADC_E_test03 □ HAL_ADC_E_test03 □ HAL_ADC_E_test04 □ HAL_ADC_E_test06 □ HAL_ADC_E_test06 □ HAL_ADC_E_test07 □ HAL_ADC_E_test08 □ HAL_ADC_E_test08 □ HAL_ADC_E_test08 □ HAL_ADC_E_test10 □ HAL_ADC_E_test11 □ HAL_ADC_E_test11 □ HAL_ADC_E_test12	• st • st • st • st • st • st • st • st	tm32f4xx_hal_adc_valid.h tm32f4xx_hal_adc_valid.c tm32f4xx_it.n tm32f4xx_it.c tm32f4xx_hal_adc_valid_e.h tm32f4xx_hal_adc_valid_e.c tm32f4xx_hal_adc_valid_f.h tm32f4xx_hal_adc_valid_f.c ted			
HAL_ADC_E_Test 13	xhau	stive T	ests		
Interpret and the interpret in the interpret interp		TEST	NAME		DESCRIPTION
ort	Â	HAL_	ADC_E	_Test	11 assert_param Test: ADC Functions Run Time Check
		HAL_	ADC_E	_Test	13 HAL_ADC_Definit rest
		HAL	ADC_E	Test	04 HAL_ADC_Init Test
	E	HAL	ADC_E	_ _Test(J5 HAL_ADC_Start and HAL_ADC_Stop Test
		HAL_	ADC_E	_Test(HAL_ADC_Start_IT, HAL_ADC_Stop_IT, HAL_ADC_InjectedStart_IT and HAL_ADC_InjectedStop_IT Test
tm32l4xx-20_02_2015.txt		HAL_	ADC_E	_Test	17 HAL_ADC_Start_DMA and HAL_ADC_Stop_DMA Test
	_	HAL_	ADC_E	_Test	08 HAL_ADC_ConfigChannel Test
ential		HAL_	ADC_E	_Test(09 HAL_ADC_InjectedConfigChannel Test
		HAL_	ADC_E	_Test	0 HAL_ADC_InjectedConfigChannel Test
		HAL_	ADC_E	_Test	1 HAL_ADC_InjectedStart and HAL_ADC_InjectedStop Test
		HAL_	ADC_E	_lest	HAL_ADC_InjectedConfigChannel Test



STM32Cube High Quality Process MISRA C compliancy 1/2

STM32Cube HALs and ST Middleware C code compliant with MISRA-C

- · Compliancy with a few exceptions listed and explained
- Check for MISRA-C 2004 is made using the IAR MISRA-C Checker (w/compiler: IAR C/C++ Compiler for ARM).
- Rules excluded from MISRA C Check (all Drivers):

MISRA-C Required/A Summarv Root Cause 2004 dvisorv Compiler is configured to allow extensions - all code shall conform to ISO 9899 IAR compiler extensions are enabled. This was allowed to support new CMSIS 1.1 Required standard C, with no extensions permitted types. Identifiers (internal and external) shall not rely on significance of more than Some long parameters names are defined for code readability. Required 5.1 31 characters Noprototype seen - functions shall always have prototype declarations and This rule is violated as there is no functions prototypes for WFI and WFE Required 8.1 the prototype shall be visible at both the function definition macros in the CMSIS layer. The value of an expression of integer type shall not be implicitly converted to 10.1 Required Complexity a different underlying type. A 'U' suffix shall be applied to all constants of 'unsigned' type The "stdint.h" defined types are used to be CMSIS compliant. 10.6 Required Conversionsshall not be performed between a pointer to object and any type 11.2 Required Needed when addressing memory mapped registers otherthan an integral type, another pointer to object type or a pointer to void. 11.3 A cast should not be performed between a pointer type and an integral type. Needed when addressing memory mapped registers Advisory The right-hand operand of a logical or I I operator shall not This rule is broken in some drivers when conditional instruction "if" and a long 12.4 Required expression is used, the solution is to split this instruction in several and small contain side effects. In some case we need to exit from function w/ status of Timeout or error A function shall have a single point of exit at the end of the function. 14.7 Required

Example of the report

immediately and not waiting for the end of process function to update the

What have we learnt? 14

Demonstrate basic information about code analysis techniques and its tools

Present ST practices used for HAL and LL libraries quality verification



Further reading 15

More information can be found in the following documents (separately for each STM32 family):

• STM32xx HAL library validation report

available on demand -> please contact your ST technical contact person

STM32xx HAL library MISRA C compliancy report

available on demand -> please contact your ST technical contact person







www.st.com/mcu

