МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 4
З дисципліни: *"Програмування мікроконтролерів"*
На тему: *"* Статичний аналіз мікропрограмного коду з використанням стандарту MISRA-C *"*

Лектор:
доц. каф. ПЗ
Марусенкова Т.А.

Виконав:
ст. гр. ПЗ-42
Бурець В.В.

Прийняв:
доц. каф. ПЗ
Марусенкова Т.А.

« ____ » _____ 2022 р.

∑= ____ _____.

Львів-2022

Тема роботи: Статичний аналіз мікропрограмного коду з використанням стандарту MISRA-C.
Мета роботи: Засвоїти поняття формального інспектування та статичного аналізу коду як інструментів перевірки програмних кодів, що погано підлягають автоматизованому тестуванню, передусім — вбудованого програмного забезпечення, а також навчитися реалізовувати ці техніки на практиці вручну та за допомогою спеціальних програмних засобів.

ЗАВДАННЯ

1. Здійснити перевірку на відповідність довільним 10 вибраним правилам MISRA-C:2004 коду проекту, створеного у середовищі Keil uVision в рамках однієї з лабораторних робіт (на вибір студента) з дисципліни “Програмування мікроконтролерів”. Аналіз провести вручну. Скласти звіт про помилки у вигляді таблиці.
2. До знайдених власних помилок кодування, зумисне продумати та внести порушення тієї підвибірки правил MISRA-C: 2004, що задається номерами правил у індивідуальному завданні.
3. Для кожного згенерованого порушення правила перевірити, чи воно буде знайдене за допомогою статичного аналізатора коду.

Індивідуальне завдання: 1. Правила 15.1, 15.2, 15.3, 15.4 та 15.5.

ХІД РОБОТИ

Таблиця 1

Звіт про порушення правил MISRA-C: 2004, знайдені шляхом перегляду коду

Правило MISRA-C: 2004	Ім'я файлу, номер рядка	Зміст помилки	Коректний варіант
Група правил 14. Передача управління			

14.7 (R) У функції має бути лише один оператор return.	can.c. 2-20	Функція sendDataBlockToCan має більше ніж 1 оператор return	int sendDataToCan(const uint32_t id, uint8_t* data, int sizeInBytes){
			CanTxMsg message;
			
			message.DLC = 8;
			message.ExtId = id;
			message.IDE = CAN_Id_Extended;
			message.RTR = CAN_RTR_DATA;
			message.StdId = 0x321;
			
			int res = 0;
			
			if(sizeInBytes > 8 || sizeInBytes <0)
			res = -1;
			if(res == 0){
			for(int i=0; i< sizeInBytes; ++i){
			message.Data[i] = *data;
			}

			int status = CAN_Transmit(CAN1, &message);
			if(status == CAN_TxStatus_Failed){
			res = -1;
			}
			}
			else{
			}
			
			return res;
			}

14.7 (R) У функції має бути лише один оператор return.	Main.c: 60, 70, 78	У функції більше ніж один оператор return	int main(){
			SystemInit();
			SysTick_Config(SystemCoreClock/1000);
			CAN1_Config();
			
			ButtonInit();
			it_init();
			
			const int size= 17;
			uint32_t id = 0x10;
			uint8_t data[] = {0x56, 0x61, 0x6c, 0x65, 0x6e, 0x74, 0x79, 0x6e, 0x20, 0x42, 0x75, 0x72, 0x65, 0x74, 0x73};
			
			int res = 0;
			while(1){
			int status = 0;
			if(isSendDataToCan == 1){
			status = sendDataToCan(id, data, size);
			isSendDataToCan = 0;
			}
			else{
			}
			if(isGetDataFromCan == 1){
			status = receiveDataFromCan(id, data, size);
			isGetDataFromCan = 0;
			}
			else{
			}
			if(status !=0){
			res = -1;
			break;
			}
			else{
			}
			};
			return res;
			}

14.10 (R) Умовний оператор завжди повинен мати гілку else, навіть якщо вона пуста	Main.c: 56, 66	Умовний оператор if не має гілку else	while(1){
			int status = 0;
			if(isSendDataToCan == 1){
			status = sendDataToCan(id, data, size);
			isSendDataToCan = 0;
			}
			else{
			}
			if(isGetDataFromCan == 1){
			status = receiveDataFromCan(id, data, size);
			isGetDataFromCan = 0;
			}
			else{
			}
			if(status !=0){
			res = -1;
			break;
			}
			else{

			} };
Група правил 2.			

2.2 (R) Дозволені тільки коментарі /* */ (// використовувати не можна).	button_init.h:
	3;
	Button_init.c:
	17, 18, 20, 21, 23, 24, 25, 28, ,29, 30, 33
		Використано заборонений тип коментарів (// використовувати не можна).	#ifndef BUTTON_INIT_H
			#define BUTTON_INIT_H
			/* contain EXTI_InitTypeDef */
			#include <stm32f4xx_exti.h>
			void ButtonInit(void);
			void it_init(void);
			#endif
			
			void it_init() {
			
			/* set interrupt port port out port */
			/* associate PA0 with EXTI0 */
			SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA,EXTI_PinSource0);
			
			/* external interrupt */
			/* create struct for working with */
			EXTI_InitTypeDef exti;
			EXTI_IMR & EXTI_RTSR registers
			exti.EXTI_Line = EXTI_Line0;
			/* EXTI_IMR ? EXTI_EMR means event or interrapts */
			exti.EXTI_Mode = EXTI_Mode_Interrupt;
			
			/* EXTI_Trigger_Rising(Falling/Rizing_Falling) */
			exti.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
			
			/* set state */
			exti.EXTI_LineCmd = ENABLE;
			EXTI_Init(&exti);
			
			/* nested vector interrupt controller */
			/* set params for contoller vector interrupts */
			NVIC_InitTypeDef nvic;
			
			/* set Chanal IRG for activation/disactivation */
			nvic.NVIC_IRQChannel = EXTI0_IRQn;
			
			nvic.NVIC_IRQChannelPreemptionPriority = 13;
			nvic.NVIC_IRQChannelSubPriority = 13;
			
			/* set activity */
			nvic.NVIC_IRQChannelCmd = ENABLE;
			
			NVIC_Init(&nvic);
			}

Група правил 8. Оголошення та визначення			

8.7 (R) Якщо змінну можна зробити локальною, її слід зробити локальною (уникати глобальних змінних по можливості).	Main.c:
	9, 10, 12,
		Глобальні змінні	static __IO uint32_t isSendDataToCan = 0;
			static __IO uint32_t isGetDataFromCan = 0;
			
			static __IO uint32_t _tempTimingDelay;
			void Delay(__IO uint32_t nTime)
			{
			_tempTimingDelay = nTime;
			
			while(_tempTimingDelay != 0);
			}
			
			void TimingDelay_Decrement(void){
			if (_tempTimingDelay != 0x00){

			_tempTimingDelay--;
			}
			}
			void SysTick_Handler(void)
			{
			TimingDelay_Decrement();
			}
			
			void EXTI0_IRQHandler(void){
			Delay(20);
			if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0)){
			isSendDataToCan = 1;
			}
			EXTI_ClearITPendingBit(EXTI_Line0);
			}
			
			void CAN1_RX0_IRQHandler(void){
			isGetDataFromCan = 1;
			}

8.1 (R) Функція повинна мати прототип, причому видимий з точки визначення та виклику функції. Тоді компілятор зможе перевірити, чи виклик правильний (кількість та типи параметрів).	Main.c:
	13, 20,
		Функція Delay не має прототипу у хедері	Навність прототипу в main.h
			void Delay(__IO uint32_t nTime);
			void TimingDelay_Decrement(void);

Група правил 14. Передача управління			

14.1 (R) Не повинно бути недосяжних фрагментів коду.	Main.c:
	78
		Return є недосяжним. Через використання додаткових умов у вічному циклі, оператор return у кінці функції ніколи не буде виконаний.	int main(){
			SystemInit();
			SysTick_Config(SystemCoreClock/1000);
			CAN1_Config();
			
			ButtonInit();
			it_init();
			
			const int size= 17;
			uint32_t id = 0x10;
			uint8_t data[] = {0x56, 0x61, 0x6c, 0x65, 0x6e, 0x74, 0x79, 0x6e, 0x20, 0x42, 0x75, 0x72, 0x65, 0x74, 0x73};
			
			int res = 0;
			while(1){
			int status = 0;
			if(isSendDataToCan == 1){
			status = sendDataToCan(id, data, size);
			isSendDataToCan = 0;
			}

			else{
			}
			if(isGetDataFromCan == 1){
			status = receiveDataFromCan(id, data, size);
			isGetDataFromCan = 0;
			}
			else{
			}
			if(status !=0){
			res = -1;
			break;
			}
			else{
			}
			};
			return res;
			}

Група правил 16. Функції			

16.7 (A) Параметр-вказівник має бути const, якщо не передбачається зміна параметра в тілі функції .	main.c: 51;
	
	can.c: 3, 24
		Параметр не є const і не передбачається зміна параметра в тілі функції .	Зробити змінну константою;
			const uint32_t id = 0x10;
			
			int sendDataToCan(const uint32_t id, uint8_t* data, int sizeInBytes);
			int receiveDataFromCan(const uint32_t id, uint8_t* data, int sizeInBytes);

Таблиця 1

Звіт про знаходження фактів порушення правил MISRA-C: 2004 статичним аналізатором

Правило MISRA-C: 2004	Ім'я файлу, номер рядка	Чи був знайдений аналізатором (“+” або “—”)	Текст повідомлення статичного аналізатора
Група правил 14. Передача управління			
14.7 (R) У функції має бути лише один оператор return.	can.c. 2-20	+	*** LINT: can.c(20) note 904: return statement before end of function 'sendDataToCan' [MISRA 2004 Rule 14.7, required] *** LINT: can.c(13) note 904: return statement before end of function 'sendDataToCan' [MISRA 2004 Rule 14.7, required] *** LINT: can.c(20) note 904: return statement before end of function 'sendDataToCan' [MISRA 2004 Rule 14.7, required]
14.7 (R) У функції має бути лише один оператор return.	Main.c: 60, 70, 78	+	*** LINT: User\src\main.c(60) note 904: return statement before end of function 'main' [MISRA 2004 Rule 14.7, required] *** LINT: User\src\main.c(70) note 904: return statement before end of function 'main' [MISRA 2004 Rule 14.7, required] *** LINT: User\src\main.c(78) note 904: return statement before end of function 'main' [MISRA 2004 Rule 14.7, required]
14.10 (R) Умовний оператор завжди повинен мати гілку else, навіть якщо вона пуста	Main.c: 56, 66	-	
Група правил 2.			
2.2 (R) Дозволені тільки коментарі /* */ (// використовувати не можна).	button_init.h: 3; Button_init.c: 17, 18, 20, 21, 23, 24, 25, 28, ,29, 30, 33	-	
Група правил 8. Оголошення та визначення			
8.7 (R) Якщо змінну можна зробити локальною, її слід	Main.c: 9, 10, 12,	-	

зробити локальною (уникати глобальних змінних по можливості).			
8.1 (R) Функція повинна мати прототип, причому видимий з точки визначення та виклику функції. Тоді компілятор зможе перевірити, чи виклик правильний (кількість та типи параметрів).	Main.c: 14, 21, 26	+	*** LINT: User\src\main.c(21) note 957: function 'TimingDelay_Decrement' defined without a prototype in scope [MISRA 2004 Rule 8.1, required] *** LINT: User\src\main.c(14) note 957: function 'Delay' defined without a prototype in scope [MISRA 2004 Rule 8.1, required] *** LINT: User\src\main.c(21) note 957: function 'TimingDelay_Decrement' defined without a prototype in scope [MISRA 2004 Rule 8.1, required] *** LINT: User\src\main.c(26) note 957: function 'SysTick_Handler' defined without a prototype in scope [MISRA 2004 Rule 8.1, required]
Група правил 14. Передача управління			
14.1 (R) Не повинно бути недосяжних фрагментів коду.	Main.c: 81	+	*** LINT: User\src\main.c(81) warning 527: statement is unreachable due to unconditional transfer of control by 'return' statement
Група правил 15. Оператор switch			
15.2 (R) Кожен непустий case має закінчуватися break.	Main.c: 88	+	*** LINT: User\src\main.c(88) info 825: control flow falls through to next case without an intervening -fallthrough comment [MISRA 2004 Rule 15.2, required]
15.3 (R) switch має завершуватися гілкою default.	Main.c: 84	+	*** LINT: User\src\main.c(84) info 744: switch statement has no default [MISRA 2004 Rule 15.3, required]
15.4 (R) switch не застосовуємо з булівськими виразами.	Main.c: 113	+	*** LINT: User\src\main.c(113) warning 483: switching on a boolean value
15.5 (R) У кожному switch	Main.c: 113	+	*** LINT: User\src\main.c(113) info 764: switch with no cases [MISRA 2004 Rule 15.5, required]

має бути хоча б один case.			
Група правил 16. Функції			
16.7 (A) Параметр-вказівник має бути const, якщо не передбачається зміна параметра в тілі функції .	main.c: 51; can.c: 3, 24	-	

Виправлений код:

main.h

```
#ifndef MAIN_H
#define MAIN_H
		#include "button_init.h"
		#include "can_init.h"
		#include "can.h"

			void Delay(__IO uint32_t nTime);
			void TimingDelay_Decrement(void);

#endif
```

main.c

```
#include "stm32f4xx.h"
#include "main.h"

#include <stm32f4xx_rcc.h>
#include <stm32f4xx_gpio.h>
#include <misc.h>
#include <stm32f4xx_syscfg.h>

static __IO uint32_t isSendDataToCan = 0;
static __IO uint32_t isGetDataFromCan = 0;

static __IO uint32_t _tempTimingDelay;
void Delay(__IO uint32_t nTime)
{
  _tempTimingDelay = nTime;

  while(_tempTimingDelay != 0);
}

void TimingDelay_Decrement(void){
  if (_tempTimingDelay != 0x00){
    _tempTimingDelay--;
  }
}
void SysTick_Handler(void)
{
  TimingDelay_Decrement();
}

void EXTI0_IRQHandler(void){
		Delay(20);
		if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0)){
			isSendDataToCan = 1;
		}
```

```
	EXTI_ClearITPendingBit(EXTI_Line0);
}

void CAN1_RX0_IRQHandler(void){
	isGetDataFromCan = 1;
}

int main(){
	SystemInit();
	SysTick_Config(SystemCoreClock/1000);
	CAN1_Config();

	ButtonInit();
 it_init();

	const int size= 17;
	uint32_t id = 0x10;
	uint8_t data[] = {0x56, 0x61, 0x6c, 0x65, 0x6e, 0x74, 0x79, 0x6e, 0x20, 0x42, 0x75, 0x72, 0x65, 0x74, 0x73};

	int res = 0;
	while(1)
	{
		int status = 0;

		if(isSendDataToCan == 1){
			status = sendDataToCan(id, data, size);
			isSendDataToCan = 0;
		}
		else{
		}
		if(isGetDataFromCan == 1){
			status = receiveDataFromCan(id, data, size);

			isGetDataFromCan = 0;
		}
		else{
		}
		if(status !=0){
			res = -1;
			break;
		}
		else{
		}
	};

	return res;
}
```

can.c

```
int sendDataToCan(const uint32_t id, uint8_t* data, int sizeInBytes){
	CanTxMsg message;

	message.DLC = 8;
	message.ExtId = id;
	message.IDE = CAN_Id_Extended;
	message.RTR = CAN_RTR_DATA;
	message.StdId = 0x321;

	int res = 0;

	if(sizeInBytes > 8 || sizeInBytes <0){
		res = -1;
	}
	else{
```

```
                for(int i=0; i< sizeInBytes; ++i){
                        message.Data[i] = *data;
                }

                int status = CAN_Transmit(CAN1, &message);
                if(status == CAN_TxStatus_Failed){
                        res = -1;
                }
        }

        return res;
}
```

Код із знайденими та доданими згідно варіанту помилками:

main.c

```
1.  #include "stm32f4xx.h"
2.  #include "main.h"
3.  .
4.  #include <stm32f4xx_rcc.h>
5.  #include <stm32f4xx_gpio.h>
6.  #include <misc.h>
7.  #include <stm32f4xx_syscfg.h>
8.  #include <stdbool.h>
9.  .
10. static __IO uint32_t isSendDataToCan = 0;
11. static __IO uint32_t isGetDataFromCan = 0;
12. .
13. static __IO uint32_t _tempTimingDelay;
14. void Delay(__IO uint32_t nTime)
15. {
16. _tempTimingDelay = nTime;

17.     while(_tempTimingDelay != 0);
18. }
19. .
20. void TimingDelay_Decrement(void){
21.     if (_tempTimingDelay != 0x00){
22.       _tempTimingDelay--;
23.     }
24. }
25. void SysTick_Handler(void)
26. {
27.     TimingDelay_Decrement();
28. }
29. .
30. void EXTI0_IRQHandler(void){
31.     Delay(20);
32.     if(GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0)){
33.        isSendDataToCan = 1;
34.     }
35.     EXTI_ClearITPendingBit(EXTI_Line0);
36. }
```

```
.
void CAN1_RX0_IRQHandler(void){
    isGetDataFromCan = 1;
}
.
int main(){
SystemInit();
SysTick_Config(SystemCoreClock/1000);
CAN1_Config();
.
ButtonInit();
it_init();
.
const int size= 17;
uint32_t id = 0x10;
uint8_t data[] = {0x56, 0x61, 0x6c, 0x65, 0x6e, 0x74, 0x79, 0x6e, 0x20, 0x42, 0x75, 0x72, 0x65, 0x74, 0x73};
.
while(1)
{
    if(isSendDataToCan == 1){
    int status = sendDataToCan(id, data, size);

    if(status !=0){
        return -1;
    }
    else{
        isSendDataToCan = 0;
    }
}
if(isGetDataFromCan == 1){
    int status = receiveDataFromCan(id, data, size);
.
if(status !=0){
    return -1;
}
else{
    isGetDataFromCan = 0;
}
}
};

return 0;
char ch;
int val;

switch(ch) {
    case '1':
        val = 1;
        /*15.2 (R) Кожен непустий case має закінчуватися break.*/
    case '2':
        val = 2;
        break;
    case '3':
        val=3;
```

```
91.         break;
92.     /* 15.3 (R) switch має завершуватися гілкою default. */
93.     //      default  :
94.     //                  ;
95.  }
96.
97.  int x;
98.  bool foo;
99.  switch (x) {
100.    case 1: // Compliant
101.        if (foo) {
102.            case 2: // Noncompliant
103.            break;
104.        }
105.        break;
106.default: // Compliant
107.break;
108.}
109.
110./*15.4 (R) switch не застосовуємо з булівськими виразами.*/
111.bool flag = false;
112./* 15.5 (R) У кожному switch має бути хоча б один case. */
113.switch(flag){}
114.
115.}
```

can.c

```
1.   #include "can.h"
2.
3.   int sendDataToCan(uint32_t id, uint8_t* data, int sizeInBytes){
4.       CanTxMsg message;
5.
6.       message.DLC = 8;
7.       message.ExtId = id;
8.       message.IDE = CAN_Id_Extended;
9.       message.RTR = CAN_RTR_DATA;
10.      message.StdId = 0x321;
11.
12.      if(sizeInBytes > 8 || sizeInBytes <0)
13.          return -1;
14.      for(int i=0; i< sizeInBytes; ++i){
15.          message.Data[i] = *data;
16.      }
17.
18.      int status = CAN_Transmit(CAN1, &message);
19.      if(status == CAN_TxStatus_Failed)
20.          return -1;
21.      return 0;
22.  }
23.  .
24.  int receiveDataFromCan(uint32_t id, uint8_t* data, int sizeInBytes){
25.      int status = 0;
26.      CanRxMsg message;
```

```
27.     message.IDE = CAN_Id_Extended;
28.     message.DLC = sizeInBytes;
29.     message.ExtId = id;
30.     message.RTR= CAN_RTR_DATA;
31.     message.FMI = 0;
32.     CAN_Receive(CAN1,1, &message); //find fifo number
33.     data = message.Data;
34.     return status;
35.  }
```

ВИСНОВКИ

В ході виконання даної лабораторної роботи було вивчено основні правила з специфікації MISRA-C:2004. Також перевірено код лабораторної роботи №3 на наявність порушень правил MISRA-C. Використано статичний аналізатор а також перегляд коду і перевірено наявність помилок та те, чи всі знаходить аналізатор. У ході лабораторної роботи було встановлено що статичний аналізатор може не відображати всі порушення правил MISRA, але в перевагу він може надати інформацію про порушення, які були упущені при перегляді коду і це не потребує великих затрат часу.