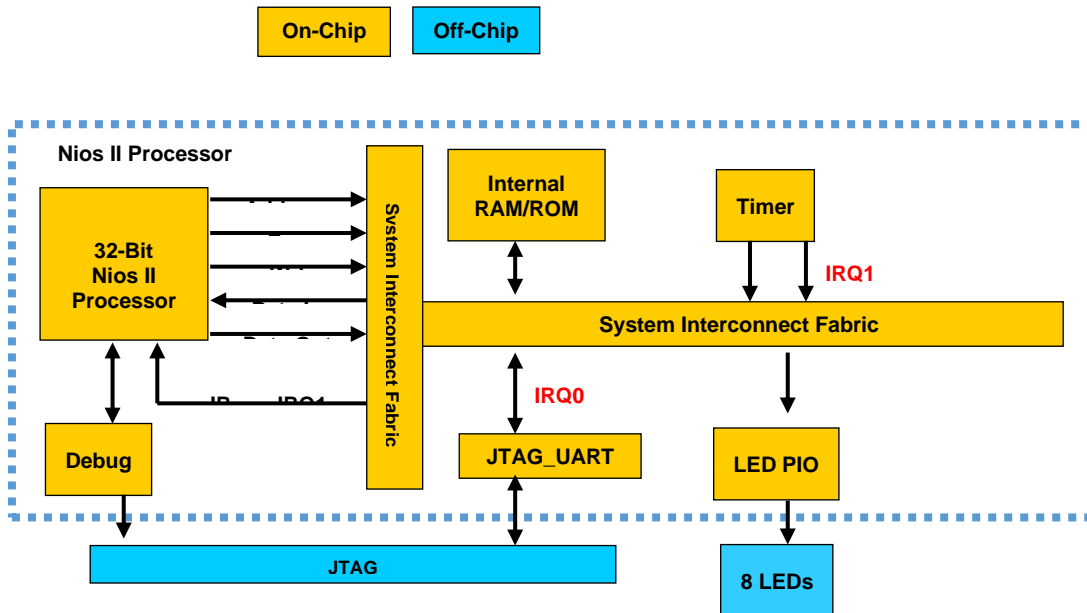


Задание labn_5

Введение:

Цель упражнения – познакомиться с возможностями процессора NIOSII при работе с таймером.

Структура проекта



Алгоритм работы:

Под управлением процессора NIOSII обеспечивается:

- Циклический вывод на светодиоды платы miniDiLab значений 0x03; 0x0c; 0x30; 0xc0 с формированием на консоли соответствующего сообщения.
- Измерение числа **ticks**, необходимых для выполнения 4 итераций циклического вывода значений на светодиоды.
- Отображение на консоли:
 1. числа **ticks** в секунду (частоты)
 2. числа **ticks**, требуемых на выполнение 4 итераций циклического вывода значений на светодиоды.

Часть 1 – Создание проекта

1. Запустите пакет QuartusII

2. Создайте проект Lab5:

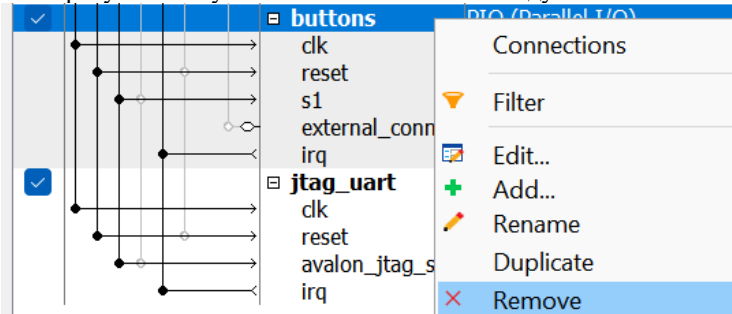
What is the working directory for this project? Рабочая папка (с помощью браузера найдите рабочую папку проекта)	C:\Intel_trn\Q_NIOS\Lab5\
What is the name of this project? Имя проекта	Lab5
What is the name of the top-level design entity for this project? Имя модуля верхнего уровня в иерархии проекта.	Lab5

- в разделе **Available devices** укажите СБИС **EP4CE6E22C8**.

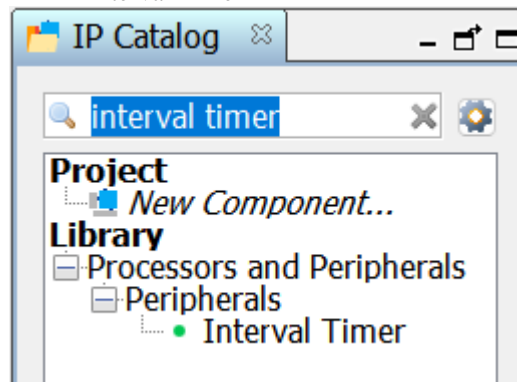
Часть 2 – Создание аппаратной части проекта

1. Скопируйте систему из проекта Lab4_nios.qsys в рабочую папку проекта.
2. Выполните команду **Tools => Qsys**. Будет запущен Qsys.
3. Откройте систему Lab4_nios.qsys и сохраните ее под именем Lab5_nios.qsys
4. Удалите из системы компонент buttons.

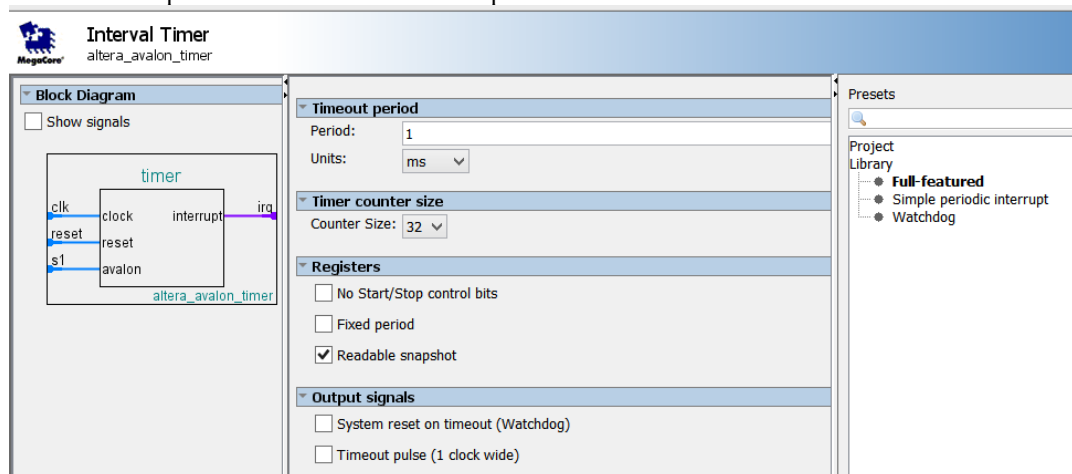
- ✓ Выберите компонент buttons
- ✓ Нажмите правую кнопку мыши и выполните команду Remove



5. Создание и настройка таймера
 - ✓ Выберите компонент Interval Timer



- ✓ Настройте его как показано на картинке:



6. Переименуйте компонент timer_0 в timer.
7. Вход clk компонента timer соедините с выходом clk компонента clk
8. Вход reset компонента timer соедините с выходом clk_reset компонента clk
9. Вход s1 компонента timer соедините с выходом data_master компонента nios2_qsys
10. Выход irq компонента timer соедините с входом irq компонента nios2_qsys
11. Выполните команду System => Assign Base Addresses
12. Назначьте вектора прерывания так, как показано на рисунке

System: Lab5_nios Path: timer_irq

Use	Connections	Name	Description	Export	Clock	Base	End	I...
<input checked="" type="checkbox"/>		clk	Clock Source	clk	<i>exported</i>			
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk				
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset				
<input checked="" type="checkbox"/>		clk	Clock Output	clk				
<input checked="" type="checkbox"/>		clk_reset	Reset Output	reset				
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM o...					
<input checked="" type="checkbox"/>		clk1	Clock Input	clk				
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped ...					
<input checked="" type="checkbox"/>		reset1	Reset Input					
<input checked="" type="checkbox"/>		nios2_qsys	Nios II Processor					
<input checked="" type="checkbox"/>		clk	Clock Input	clk				
<input checked="" type="checkbox"/>		reset	Reset Input	reset				
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped ...					
<input checked="" type="checkbox"/>		instruction_m...	Avalon Memory Mapped ...					
<input checked="" type="checkbox"/>		irq	Interrupt Receiver					
<input checked="" type="checkbox"/>		debug_reset_r...	Reset Output					
<input checked="" type="checkbox"/>		debug_mem_...	Avalon Memory Mapped ...					
<input checked="" type="checkbox"/>		custom_instru...	Custom Instruction Master					
<input checked="" type="checkbox"/>		led	PIC (Parallel I/O)					
<input checked="" type="checkbox"/>		clk	Clock Input	clk				
<input checked="" type="checkbox"/>	reset	Reset Input	reset					
<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped ...						
<input checked="" type="checkbox"/>	external_conn...	Conduit						
<input checked="" type="checkbox"/>	jtag_uart	JTAG UART						
<input checked="" type="checkbox"/>	clk	Clock Input	clk					
<input checked="" type="checkbox"/>	reset	Reset Input	reset					
<input checked="" type="checkbox"/>	avalon_jtag_sl...	Avalon Memory Mapped ...						
<input checked="" type="checkbox"/>	irq	Interrupt Sender						
<input checked="" type="checkbox"/>	timer	Interval Timer						
<input checked="" type="checkbox"/>	clk	Clock Input	clk					
<input checked="" type="checkbox"/>	reset	Reset Input	reset					
<input checked="" type="checkbox"/>	s1	Avalon Memory Mapped ...						
<input checked="" type="checkbox"/>	irq	Interrupt Sender						

13. Выполните команду File=>Save
14. Выполните команду Generate=>Generate HDL.
15. Задайте опции так, как показано на рисунке

Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis:

☐ Create timing and resource estimates for third-party EDA synthesis tools.

☐ Create block symbol file (.bsf)

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscrip* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model:

Output Directory

Path:

16. Нажмите кнопку Generate.
 17. Не должно появиться ошибок генерации. Предупреждение
- Warning: Lab5_nios.jtag_uart: JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board можно проигнорировать.

Часть 3 – Интеграция аппаратной части проекта

1. Создайте в текстовом редакторе файл (имя файла Lab5.sv) верхнего уровня в иерархии проекта (для этого целесообразно использовать файл Lab5_nios_inst.v из папки C:\Intel_trn\Q_NIOS\Lab5\Lab5_nios)
 - a. Пример файла приведен на рисунке

```

1  module Lab5 (
2      input bit clk,          // Clock
3      input bit pbb,          // System reset active low
4      output bit [7:0] led
5  );
6  Lab5_nios u0 (
7      .clk_clk      (clk), // clk.clk
8      .reset_reset_n (pbb), // reset.reset_n
9      .led_export    (led) // led.export
10 );
11 endmodule

```

2. Назначьте файл Lab5.sv файлом верхнего уровня в иерархии проекта.
3. Добавьте к проекту файл Lab5_nios.qip из папки C:\Intel_trn\Q_NIOS\Lab5\Lab5_nios\synthesis
4. Проверка синтаксиса проекта.
 - a. Выполните команду Processing=>Start=>Start Analysis and Elaboration
5. Назначение выводов проекта. Назначьте выводы так, как показано на рисунке ниже

Node Name	Direction	Location	I/O Bank	/REF Group	I/O Standard	Current Strength
in clk	Input	PIN_23	1	B1_N0	3.3-V LVTTL	8mA (default)
out led[7]	Output	PIN_65	4	B4_N0	2.5 V	8ma
out led[6]	Output	PIN_66	4	B4_N0	2.5 V	8ma
out led[5]	Output	PIN_67	4	B4_N0	2.5 V	8ma
out led[4]	Output	PIN_68	4	B4_N0	2.5 V	8ma
out led[3]	Output	PIN_69	4	B4_N0	2.5 V	8ma
out led[2]	Output	PIN_70	4	B4_N0	2.5 V	8ma
out led[1]	Output	PIN_71	4	B4_N0	2.5 V	8ma
out led[0]	Output	PIN_72	4	B4_N0	2.5 V	8ma
in pbb	Input	PIN_58	4	B4_N0	2.5 V	8mA (default)

18. Назначение опции проекта:

Assignment=>Device=>Device and Pin Options=>Unused pin=>As input tri-stated with weak pull-up resistor
19. С помощью Timing Analyzer или в текстовом редакторе создайте файл (**Lab5.sdc**) с требованиями к временным параметрам проекта. Пример файла приведен на рисунке

```

##
## DEVICE "EP4CE6E22C8"
##

#####
# Time Information
#####

set_time_format -unit ns -decimal_places 3

#####
# Create Clock
#####

create_clock -name {clock} -period 40.000 -waveform { 0.000 20.000 } [get_ports {clk}]

#####
# Set Clock Uncertainty
#####

derive_clock_uncertainty

#####
# Set Input Delay
#####

set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {pbb}]

set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {altera_reserved_tdi}]
set_input_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {altera_reserved_tms}]

#####
# Set Output Delay
#####

set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[0]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[1]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[2]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[3]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[4]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[5]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[6]}]
set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {led[7]}]

set_output_delay -add_delay -clock [get_clocks {clock}] 10.000 [get_ports {altera_reserved_tdo}]

```

20. Подключите файл Lab5.sdc к проекту.
21. Осуществите полную компиляцию проекта: команда **Processing => Start Compilation**.
22. Компиляция должна завершиться без ошибок. Все требования к временным параметрам должны быть выполнены.
23. Осуществите программирование СБИС ПЛ на плате: *на плате загорится зеленый светодиод*.

Часть 4 – Создание программной части проекта

1. Запустите оболочку для разработки/отладки программ - NIOSII SBT
✓ Рабочую папку задайте как: C:\Intel_trn\Q_NIOS\Lab5\workspace
2. Создайте проект lab5_sw (lab5_sw и lab5_sw_bsp)

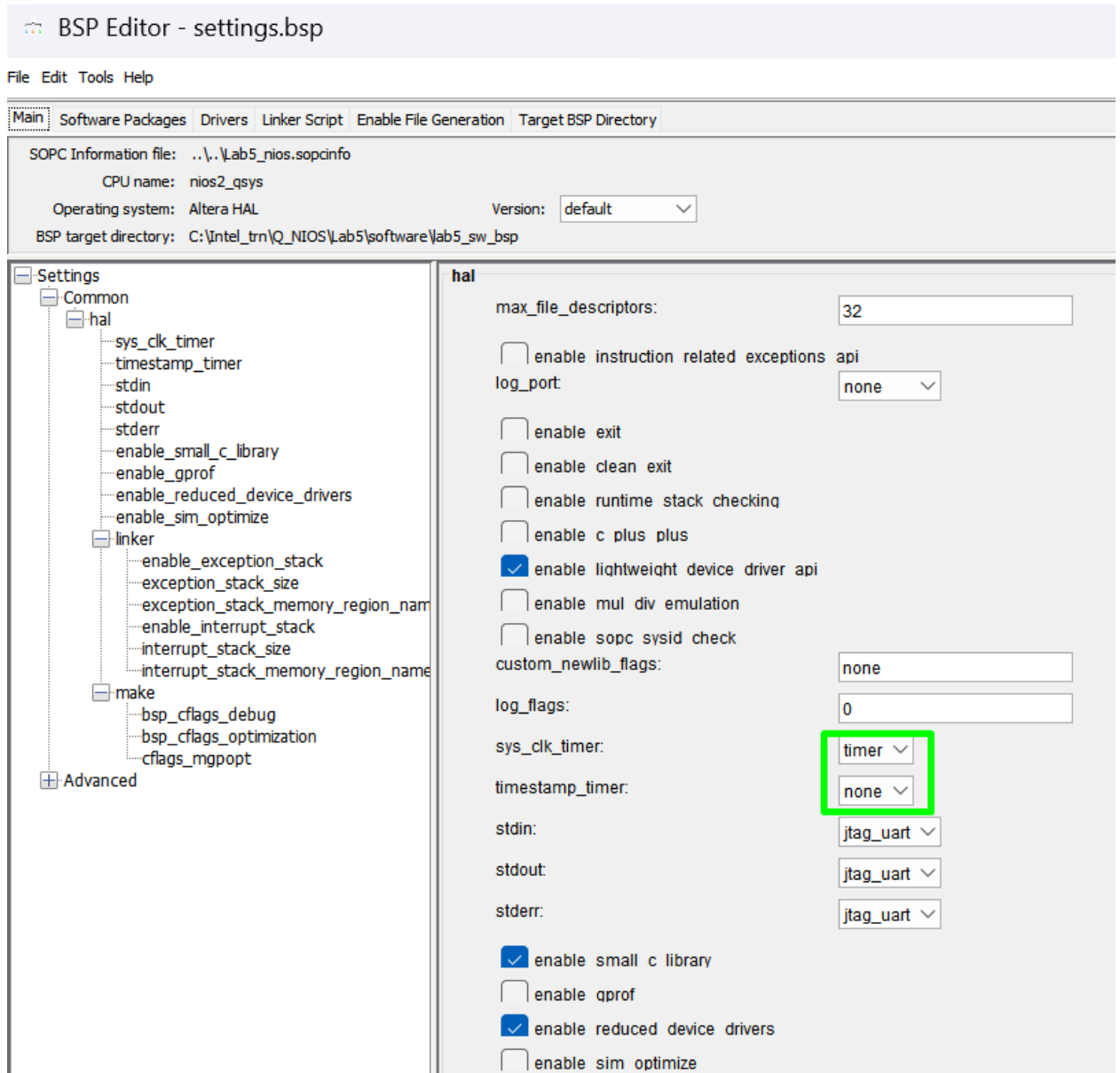
Этап 1.

1. В проекте lab5_sw создайте исходный файл lab5_source_a.c
2. Введите исходный код программы

```
lab5_source_a.c
1 #include "system.h"
2 #include "altera_avalon_pio_regs.h"
3 #include <time.h>
4 #include <unistd.h>
5 #include <sys\alt_alarm.h>
6 #include "stdio.h"
7 int main() {
8     int i;
9     alt_u32 num_ticks = 0;
10    alt_u32 time1, time2, timer_overhead;
11    printf("Процессор Nios II запущен!\n");
12    // Initialize the registers in the LED PIO peripheral:
13    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0x00);
14    while (1) {
15        // 1. Determine the timer overhead involved to record time stamp: ***
16        time1 = alt_nticks();
17        time2 = alt_nticks();
18        timer_overhead = time2 - time1;
19        // 2. Sample time1: ***
20        time1 = alt_nticks();

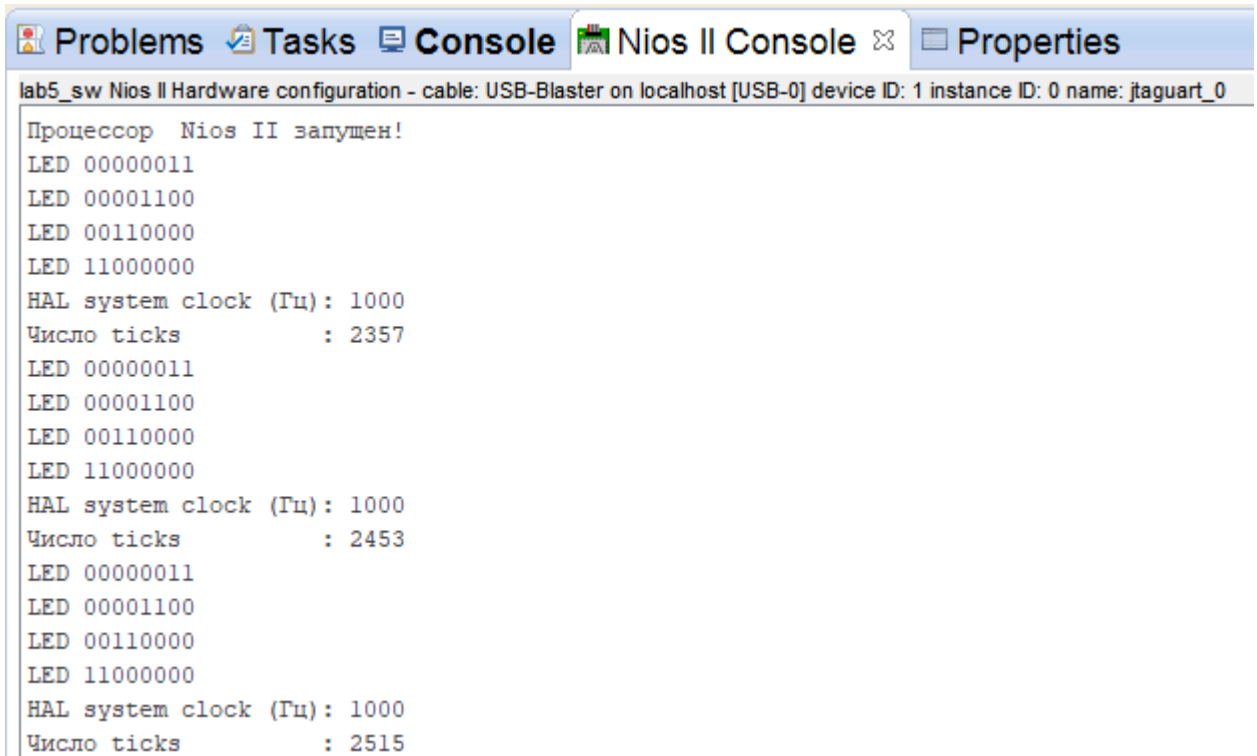
21    // Test Loop:
22    for (i=1; i<5; i++) {
23        switch (i) {
24            case 4:
25                IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0xC0);
26                printf("LED 11000000\n");
27                usleep(500000);
28                break;
29            case 3:
30                IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0x30);
31                printf("LED 00110000\n");
32                usleep(500000);
33                break;
34            case 2:
35                IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0x0C);
36                printf("LED 00001100\n");
37                usleep(500000);
38                break;
39            case 1:
40                IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0x03);
41                printf("LED 00000011\n");
42                usleep(500000);
43                break;
44        } // end switch
45    } // end for loop
46    // 3. Sample Time 2: ***
47    time2 = alt_nticks();
48    num_ticks = time2 - time1 - timer_overhead;
49    // Print out the time it took to perform this loop:
50    printf("HAL system clock (Tn): %u \n", (unsigned int)alt_ticks_per_second());
51    printf("Число ticks : %u \n", (unsigned int)num_ticks);
52 } // end while loop
53 return 0;
54 }
```


3. В lab5_sw_bsp установите опции, минимизирующие объем кода исполняемого файла и настройте таймеры как показано на рисунке, приведенном ниже.



4. Нажмите кнопку generate, затем exit
5. Запустите порождение исполняемого кода для проекта lab5_sw.
6. Проверьте результат компиляции:


```
Info: Linking lab5_sw.elf
nios2-elf-g++ -T'../lab5_sw_bsp/linker.x' -msys-crt0='../lab5_sw_bsp/obj/HAL/src/crt0.o' -ms
nios2-elf-insert lab5_sw.elf --thread_model hal --cpu_name nios2_qsys --qsys true --simulation_
Info: (lab5_sw.elf) 6744 Bytes program size (code + initialized data).
Info: 9388 Bytes free for stack + heap.
Info: Creating lab5_sw.objdump
nios2-elf-objdump --disassemble --syms --all-header --source lab5_sw.elf >lab5_sw.objdump
[lab5_sw build complete]
```
7. В NIOSII SBT выберите папку lab5_sw и запустите загрузку и выполнение программы на плате
8. Проверьте работу программы:
 - a. На консоли отображается:
 - i. Число передаваемое на светодиоды
 - ii. число ticks в секунду (частота)
 - iii. число ticks, требуемых на выполнение 4 итераций циклического вывода значений на светодиоды.
 - iv. Циклический вывод на светодиоды платы miniDiLab значений 0x03; 0x0c; 0x30; 0xc0.



```
lab5_sw Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
Процессор Nios II запущен!
LED 00000011
LED 00001100
LED 00110000
LED 11000000
HAL system clock (Гц): 1000
Число ticks      : 2357
LED 00000011
LED 00001100
LED 00110000
LED 11000000
HAL system clock (Гц): 1000
Число ticks      : 2453
LED 00000011
LED 00001100
LED 00110000
LED 11000000
HAL system clock (Гц): 1000
Число ticks      : 2515
```

9. Измените программу так, чтобы кроме числа ticks отображалось и процессорное время (в ms), потраченное на выполнение 4 итераций циклического вывода значений на светодиоды.
10. Запустите ее выполнение и зафиксируйте результаты.

Этап 2

1. В проекте lab5_sw создайте файл lab5_source_b.c :

```

lab5_source_b.c
1 #include "sys\alt_stdio.h"
2 #include "system.h"
3 #include "altera_avalon_pio_regs.h"
4 #include <time.h>
5 #include <unistd.h>
6 #include <sys/alt_alarm.h>
7 alt_u32 my_alarm_callback (void* context){
8     volatile int* leds_val_ptr = (volatile int *) context;
9     if ((*leds_val_ptr)==0x80) *leds_val_ptr=0x01;
10    else *leds_val_ptr = (*leds_val_ptr) <<1;
11    alt_printf ("подпрограмма активизирована\n");
12    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE,*leds_val_ptr);
13    return alt_ticks_per_second(); // введите код ( return numbers of tick before next alarm)
14 }
15 int main(){
16     volatile int leds=0x01; // Declare variable "button"
17     static alt_alarm alarm;
18     // Initialize the registers in the LED_PIO peripheral:
19     IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE,0x00);
20     if ( alt_alarm_start(&alarm, alt_ticks_per_second(),my_alarm_callback, (void*)&leds) < 0)
21         alt_printf ("No system clock available\n" );
22     else alt_printf ("Процессор Nios II запущен!\n");
23     while (1)
24     {
25     }
26     return 0;
27 }

```

2. Исключите файл lab5_source_a.c из компиляции.
3. Выберите папку lab5_sw и выполните компиляцию проекта. При возникновении ошибок исправьте их.
4. При успешном окончании компиляции запустите выполнение программы (при необходимости осуществите перепрограммирование СБИС ПЛ).
5. Проверьте правильность функционирования программы:
 - a. На консоли формируются сообщения:


```

Процессор Nios II запущен!
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована
подпрограмма активизирована

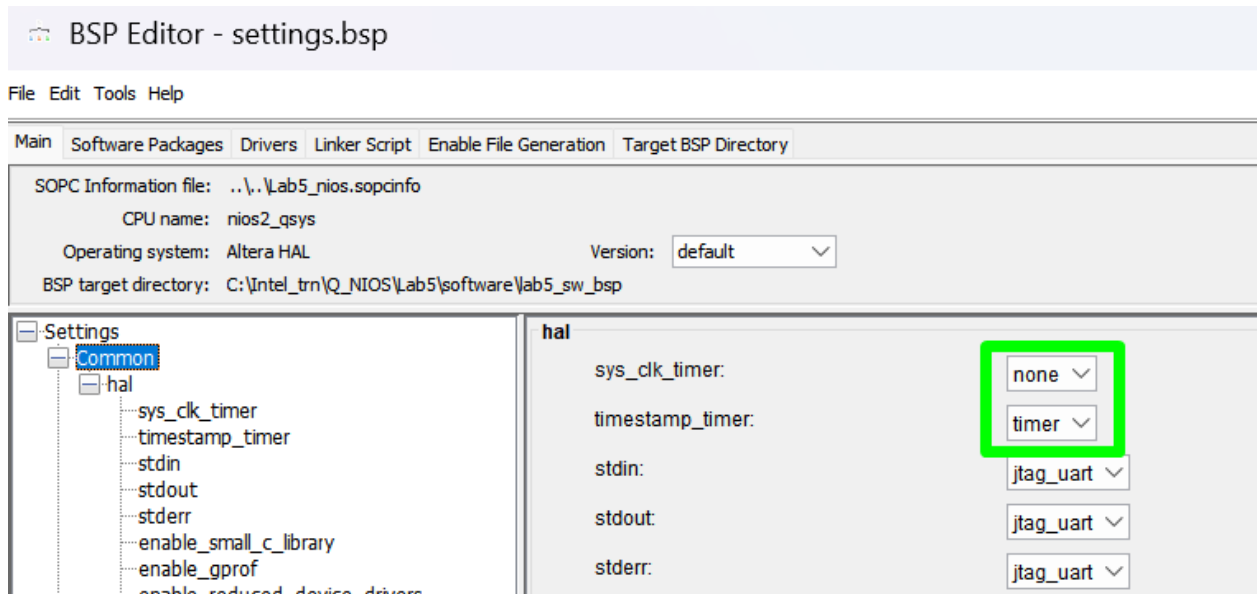
```
 - b. На светодиодах осуществляется циклический сдвиг **выключенного** светодиода от led1 к led8, с циклическим переходом к led1
6. **Измените программу так**, чтобы на светодиодах осуществлялся циклический сдвиг **включенного** светодиода от led1 к led8, с циклическим переходом к led1.

Этап 3

1. В проекте lab5_sw создайте файл lab5_source.c:

```
lab5_source.c
1 #include <stdio.h>
2 #include "system.h"
3 #include "altera_avalon_pio_regs.h"
4 #include <time.h>
5 #include <unistd.h>
6 #include <sys/alt_timestamp.h>
7 int main() {
8     int i;
9     alt_u32 num_ticks = 0;
10    alt_u32 time1, time2, timer_overhead;
11    // Initialize the registers in the LED_PIO peripheral:
12    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, 0x00);
13    // Check Timer:
14    if (alt_timestamp_start() < 0)
15        printf("Проблема инициализации таймера \n");
16    else
17        printf("Процессор Nios II запущен!\n");
18
19    while (1) {
20        // 1. Determine the timer overhead involved to record time stamp: ***
21        time1 = alt_timestamp();
22        time2 = alt_timestamp();
23        timer_overhead = time2 - time1;
24        // 2. Sample time1: ***
25        time1 = alt_timestamp();
26        // Test Loop:
27        for (i=1; i<5; i++) {
28            switch (i) {
29                case 4:
30                    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, ~0xC0);
31                    printf("LED 11000000\n");
32                    usleep(500000);
33                    break;
34                case 3:
35                    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, ~0x30);
36                    printf("LED 00110000\n");
37                    usleep(500000);
38                    break;
39                case 2:
40                    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, ~0x0C);
41                    printf("LED 00001100\n");
42                    usleep(500000);
43                    break;
44                case 1:
45                    IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, ~0x03);
46                    printf("LED 00000011\n");
47                    usleep(500000);
48                    break;
49            } // end switch
50        } // end for loop
51        // 3. Sample Time 2: ***
52        time2 = alt_timestamp();
53        num_ticks = time2 - time1 - timer_overhead;
54        if (alt_timestamp_start() < 0) //Reset Timer
55            printf("Проблема инициализации таймера \n");
56        // Print out the time it took to perform this loop:
57        printf("Частота работы процессора - CPU clock (Гц): %u \n", (unsigned int)alt_timestamp_freq());
58        printf("Число ticks : %u \n", (unsigned int)num_ticks);
59    } // end while loop
60    return 0;
}
```

2. Внесите изменения в настройки BSP.



3. Исключите файл lab5_source_b.c из компиляции.
4. Выберите папку lab5_sw и выполните компиляцию проекта. При возникновении ошибок исправьте их.
5. При успешном окончании компиляции запустите выполнение программы.
6. Проверьте правильность функционирования программы:
 - b. На консоли отображается:
 - i. Число, передаваемое на светодиоды
 - ii. число ticks в секунду
 - iii. число ticks, требуемых на выполнение 4 итераций циклического вывода значений на светодиоды.
 - iv. Циклический вывод на светодиоды платы miniDiLab значений 0x03; 0x0c; 0x30; 0xc0.



11. Измените программу так, чтобы кроме числа ticks отображалось и процессорное время (в ns), потраченное на выполнение 4 итераций циклического вывода значений на светодиоды. Сравните полученное значение со значением, полученным при выполнении этапа 1.

Упражнение 5 завершено.