Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе № 3**

**Дисциплина**: Автоматизация проектирования дискретных устройств

Выполнил студент гр. 3530901/70203 И.Д. Иванов

(подпись)

Преподаватель А.A. Антонов (подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург

2020

Оглавление

[1. Цель работы 3](#_Toc36993661)

[2. Структура и алгоритм работы проекта 3](#_Toc36993662)

[3. Ход работы 4](#_Toc36993663)

[3.1. Создание аппаратной части проекта 4](#_Toc36993664)

[3.2. Интеграция аппаратной части проекта 10](#_Toc36993665)

[3.3. Создание программной части проекта 11](#_Toc36993666)

[3.4. Анализ и оптимизация размера исполняемого кода программы 13](#_Toc36993667)

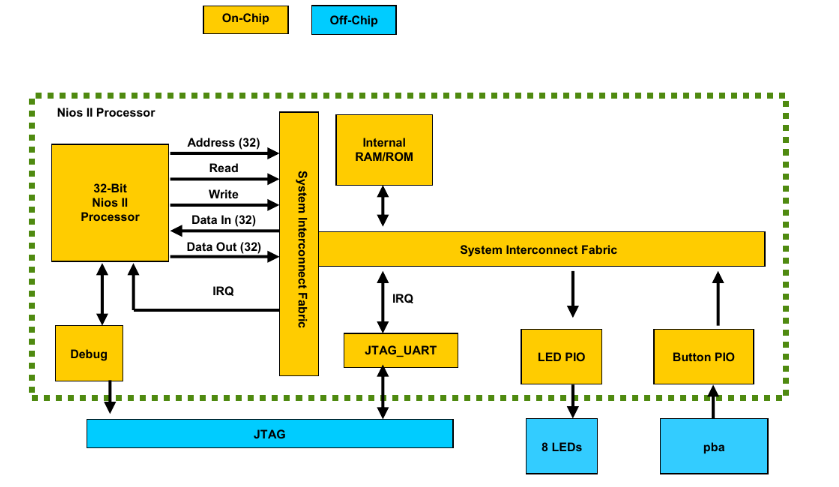
[4. Выводы 17](#_Toc36993668)

# Цель работы

Расширить знакомство с возможностями по реализации проектов на базе процессора NIOSII.

# Структура и алгоритм работы проекта

Структура проекта:



Алгоритм работы проекта:

Под управлением процессора NIOSII обеспечивается:

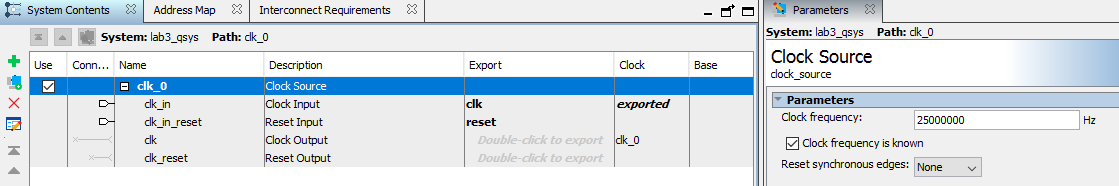
* Опрос состояния кнопок pbb
* Формирование на консоли сообщений о нажатой кнопке
* При каждом нажатии кнопки pbb происходит изменение номера включенного светодиода от led1 к led8 на одну позицию (с циклическим переходом от led8 к led1).

# Ход работы

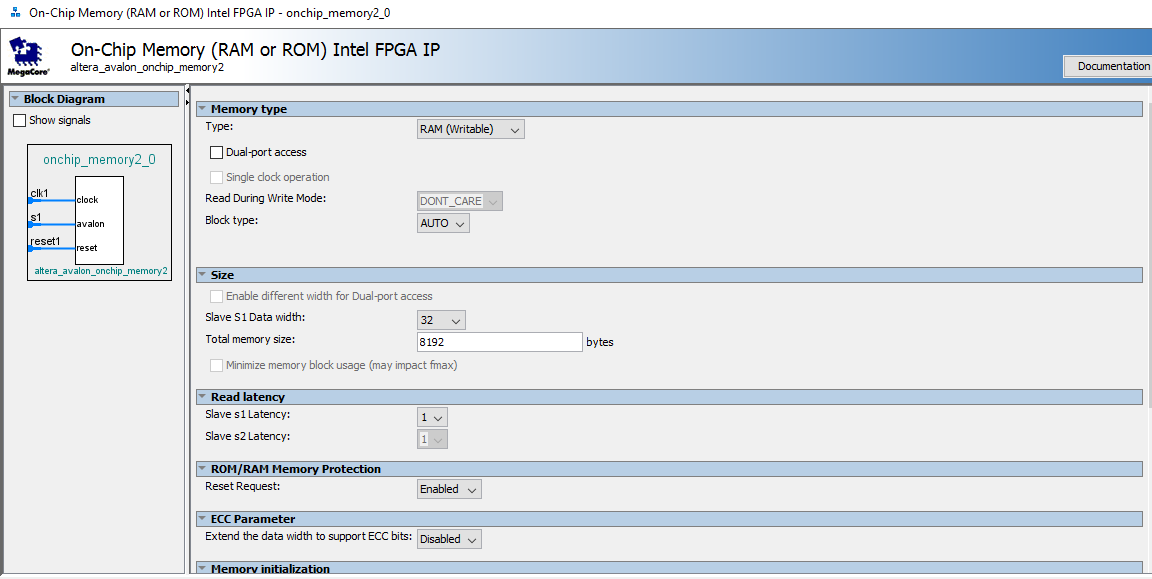
В пакете QuarusII был создан новый проект с необходимыми настройками.

# 3.1. Создание аппаратной части проекта

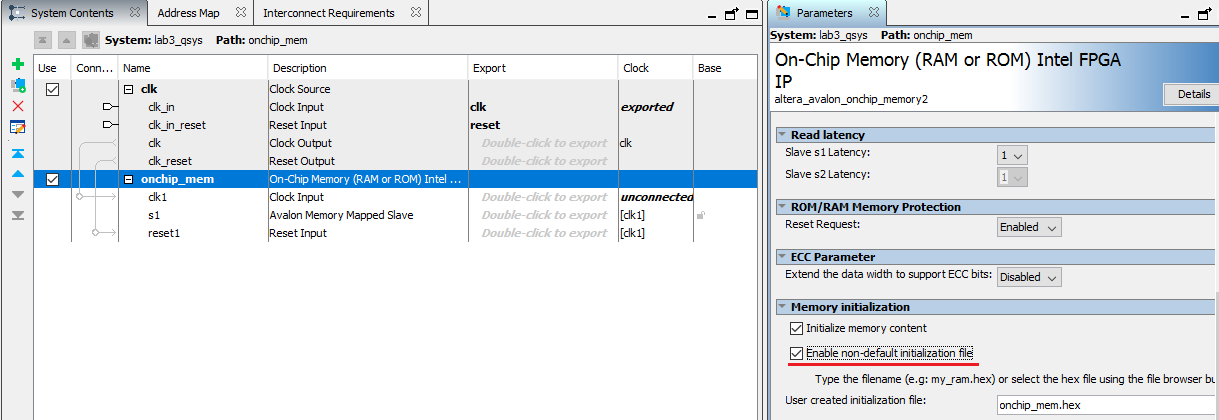
Создадим новый Qsys файл и зададим компоненту source clock, добавленному по умолчанию, частоту 25 МГц:



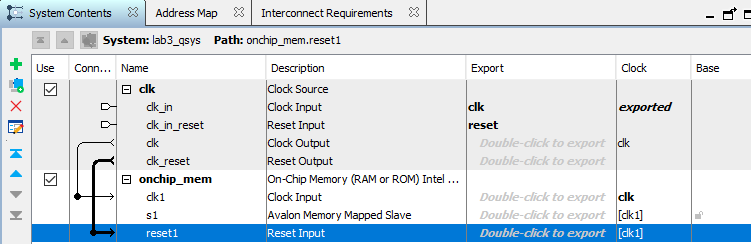
Создадим память для команд и данных процессора с помощью On-Chip Memory:



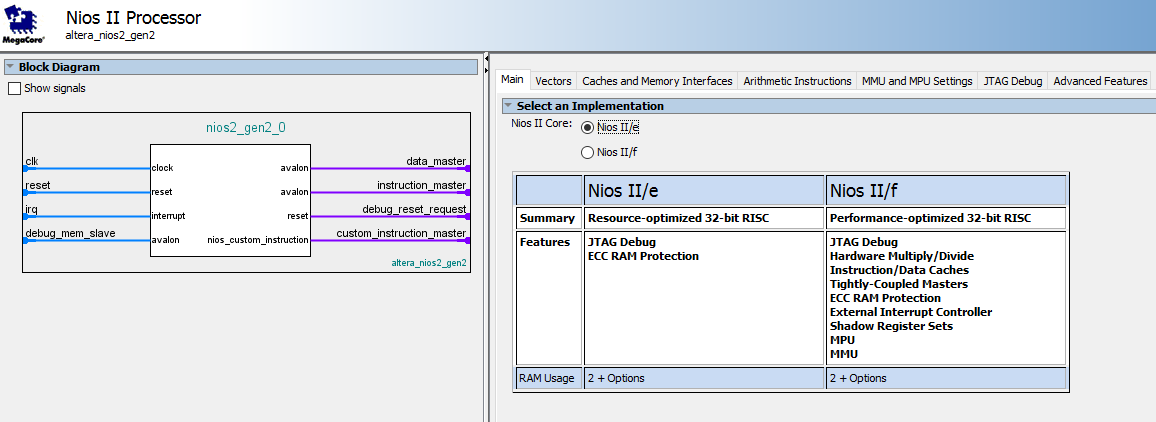
У созданного компонента установим опцию Enable non-default initialization file:



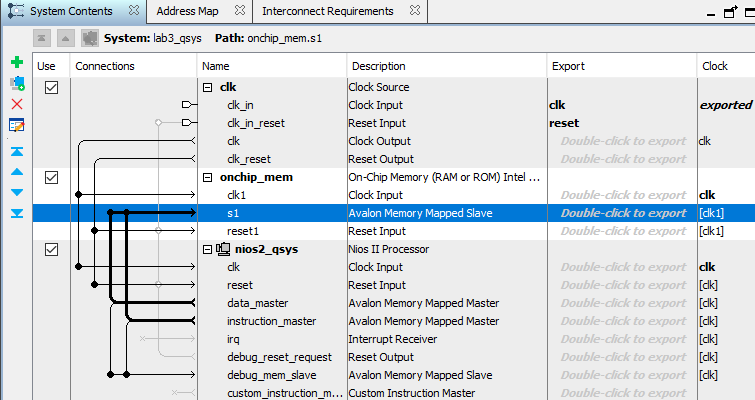
Соединим выход clk компонента clk с входом clk1 компонента onchip\_mem, а выход clk\_reset компонента clk с входом reset1 компонента onchip\_mem:



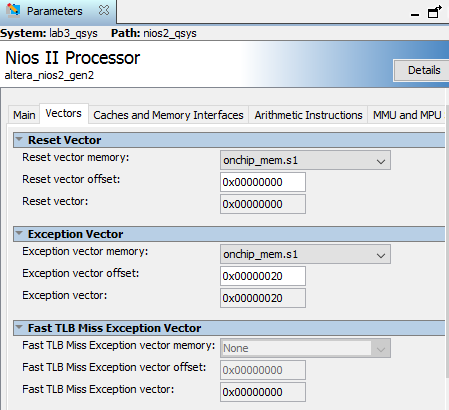
Сконфигурируем и подключим к системе ядро процессора NIOSII:



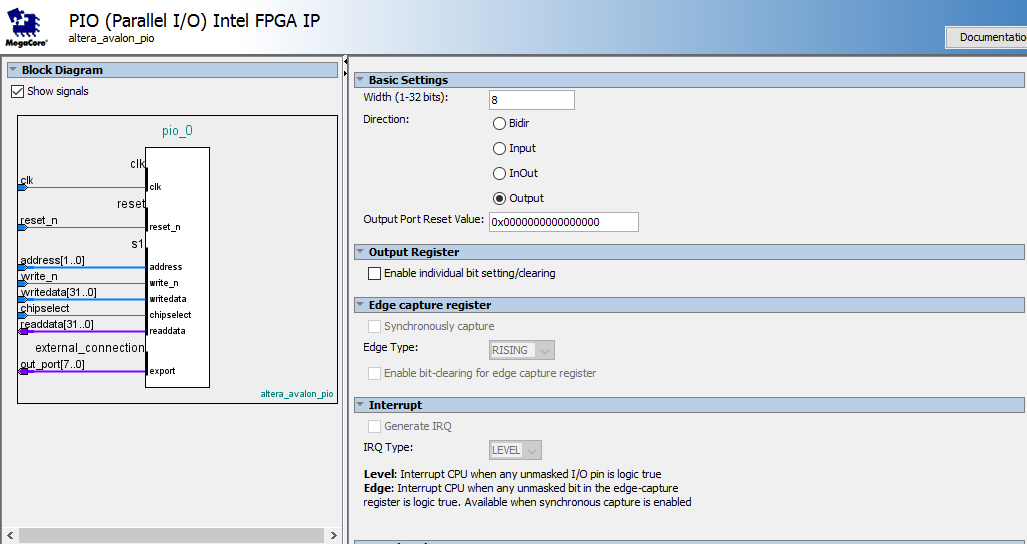
Соединим вход clk компонента nios2\_qsys с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента nios2\_qsys. Кроме того, cоединим вход s1 компонента onchip\_mem с выходами data\_master и instruction\_master компонента nios2\_qsys:



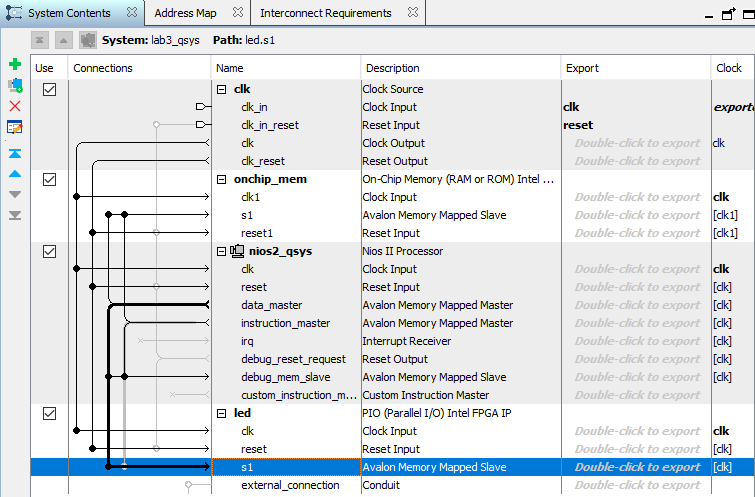
Укажем память для reset вектора и exception вектора:



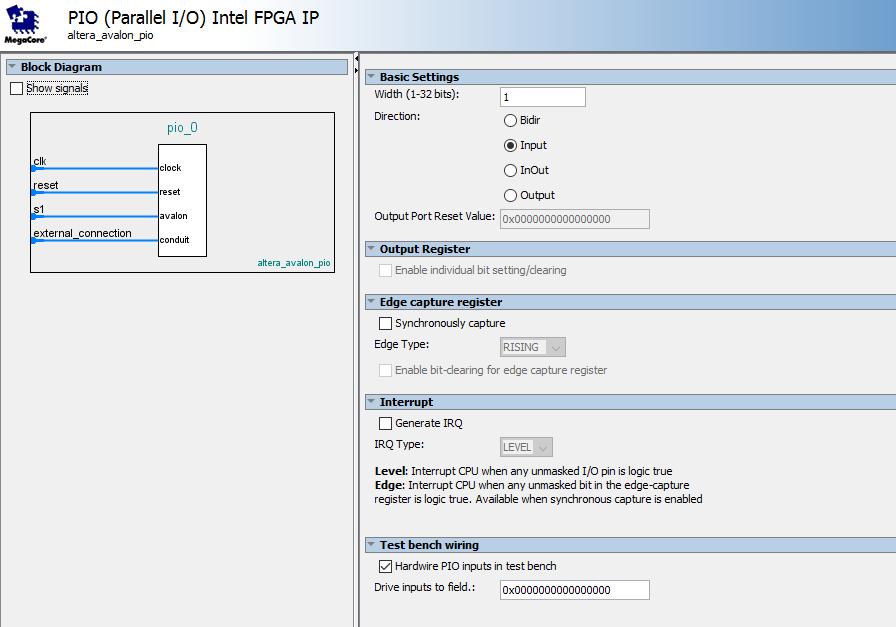
Сконфигурируем и подключим к системе модуль PIO (led):



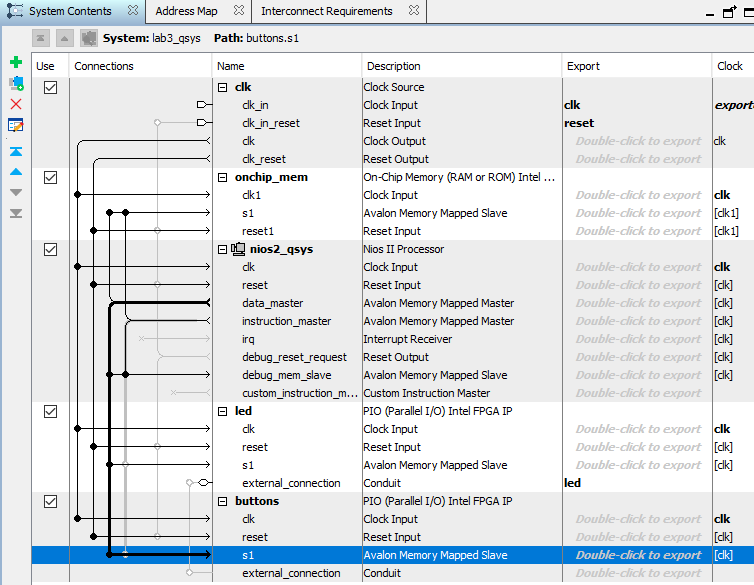
Соединим вход clk компонента led с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента led. Кроме того, соединим вход s1 компонента led с выходом data\_master компонента nios2\_qsys:



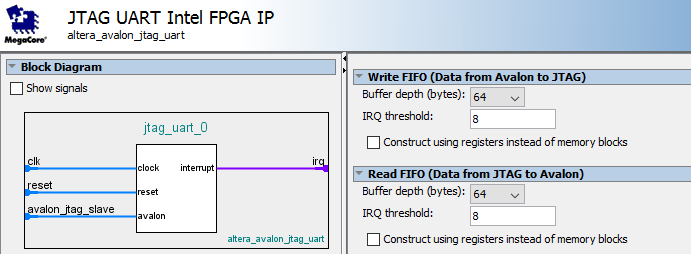
Сконфигурируем и подключим к системе ещё один модуль PIO (buttons):



Соединим вход clk компонента buttons с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента buttons. Кроме того, соединим вход s1 компонента buttons с выходом data\_master компонента nios2\_qsys:

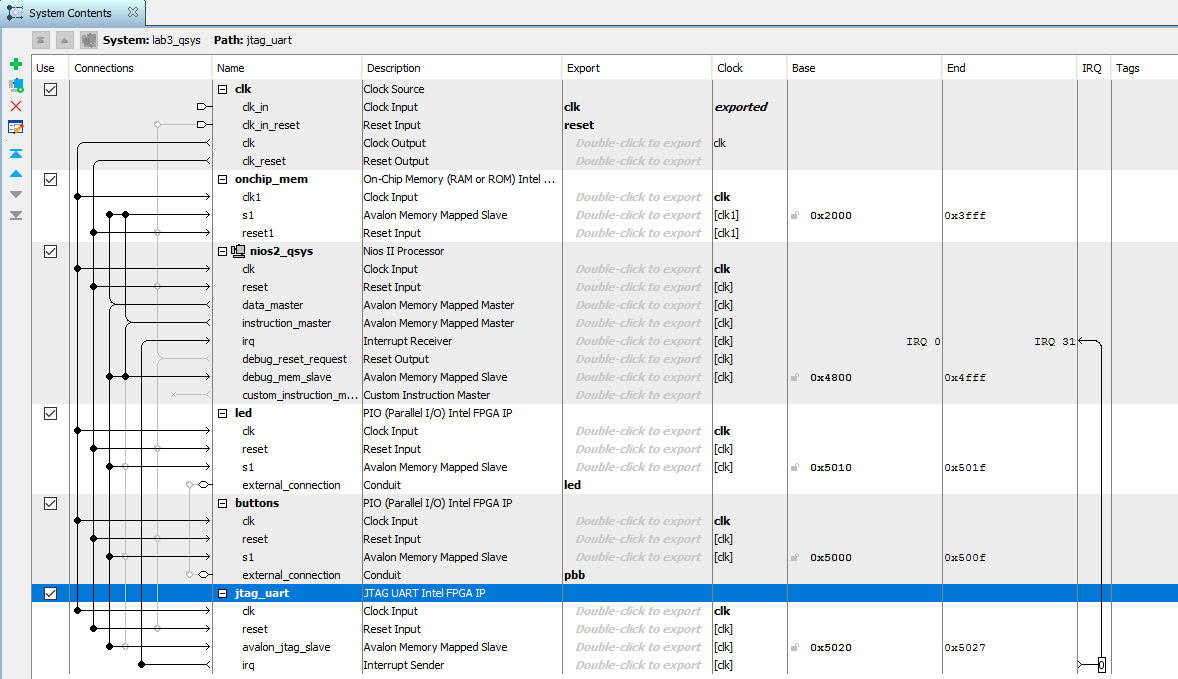


Сконфигурируем и подключим к системе модуль JTAG UART:

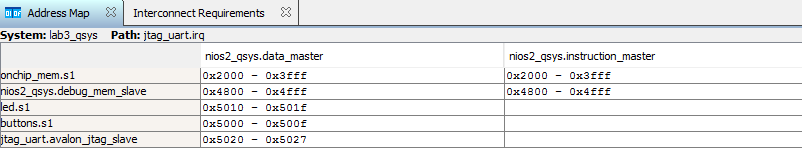


Соединим вход clk компонента jtag\_uart с выходом clk компонента clk, а выход clk\_reset компонента clk с входом reset компонента jtag\_uart. Кроме того, соединим вход avalon\_jtag\_slave компонента jtag\_uart с выходом data\_master компонента nios2\_qsys и выход прерывания (колонка IRQ) компонента jtag\_uart с входом прерывания компонента nios2\_qsys.

Внешний вид созданной системы, закладка System Contents:

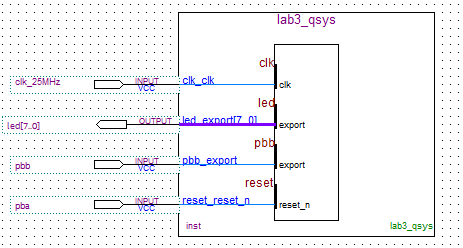


Внешний вид созданной системы, закладка Address Map:

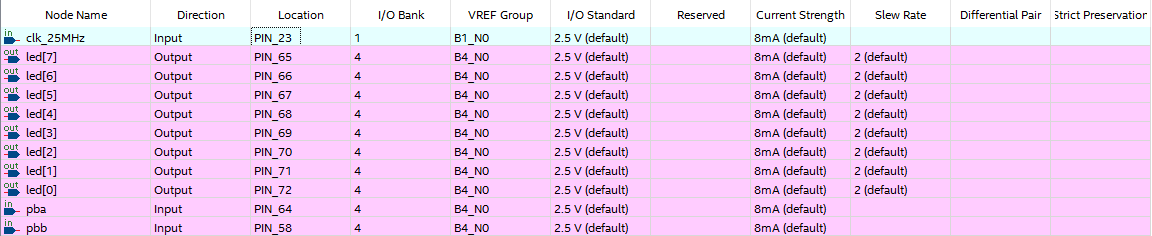


# 3.2. Интеграция аппаратной части проекта

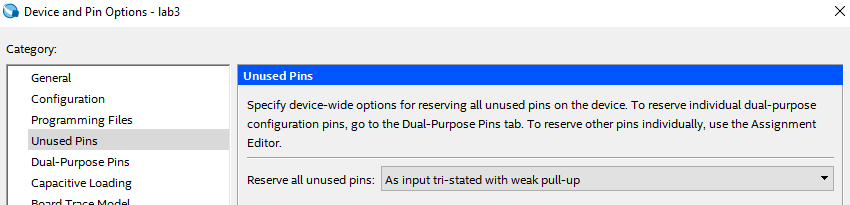
Созданная в графическом редакторе схема проекта:



Назначение выводов проекта:



Назначение опции проекта:

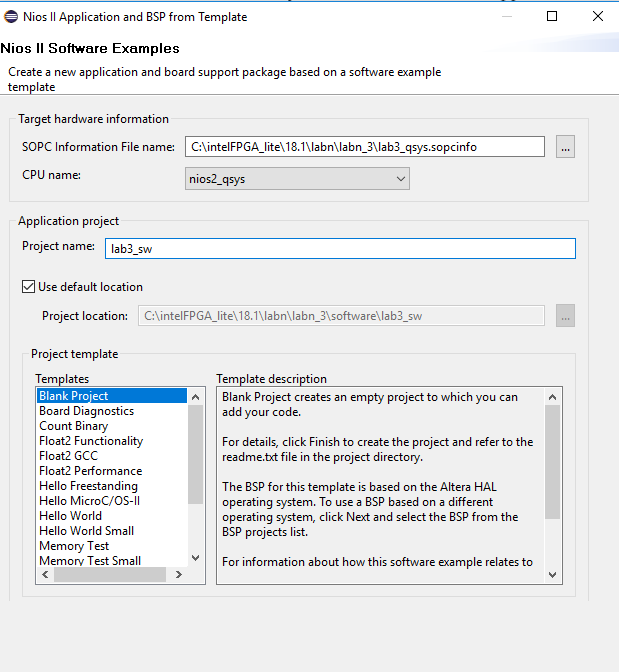


Была проведена полная компиляция проекта.

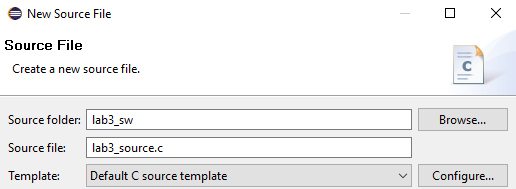
# 3.3. Создание программной части проекта

С помощью среды разработки Eclipse (с Nios II Software) создадим и настроим программную часть проекта.

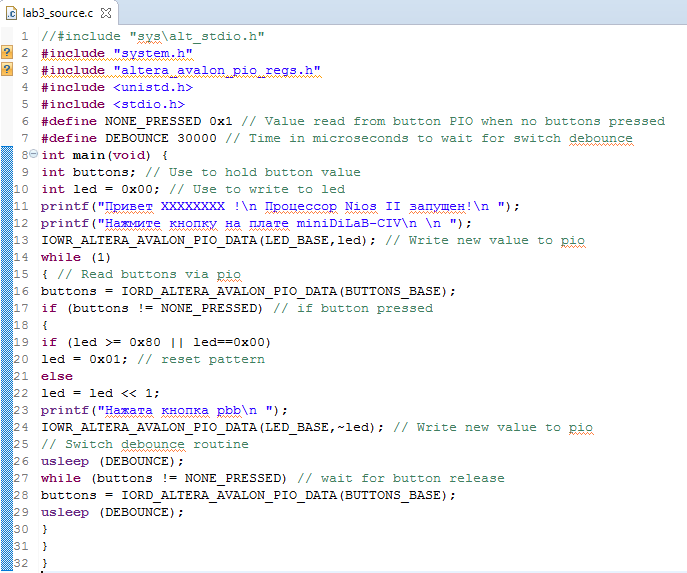
Для начала создадим сам проект:



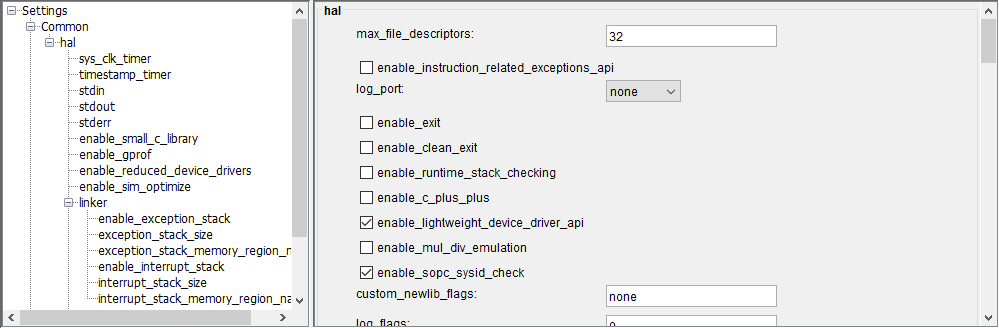
Создадим Source File:



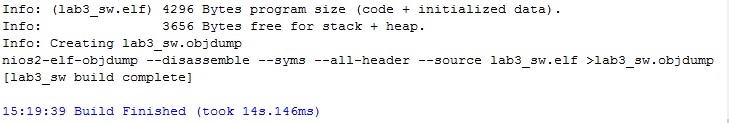
Текст программы на языке Си:



Настройки в BSP Editor:

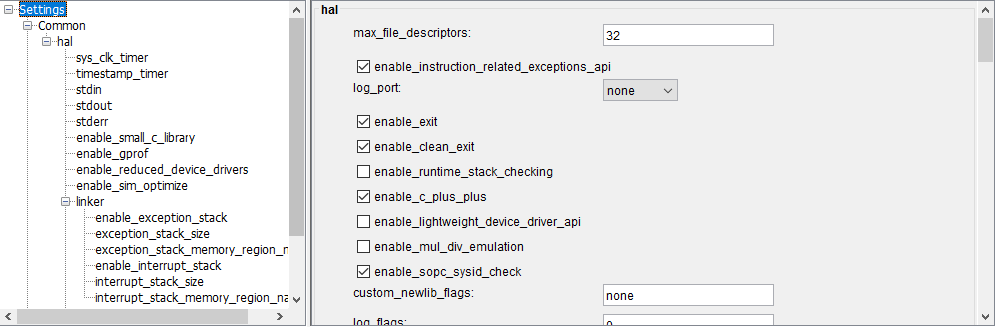


Проект был успешно собран:

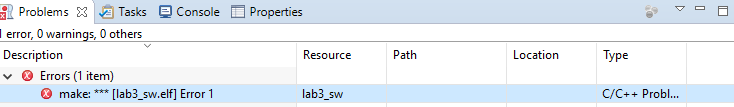


# 3.4. Анализ и оптимизация размера исполняемого кода программы

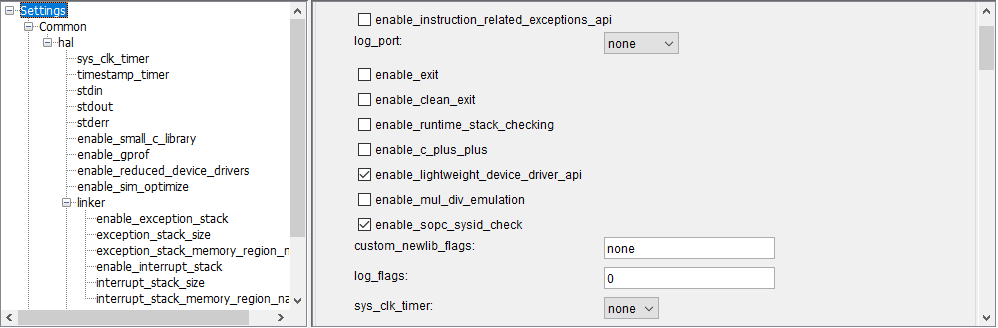
Для проекта lab3\_sw\_bsp установим опции BSP по умолчанию:



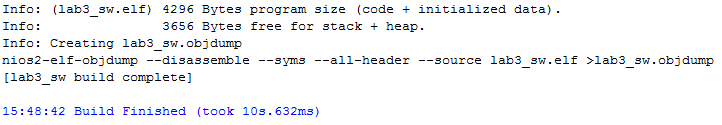
В результате компиляции получаем ошибку, так как для исходного кода программы не хватает памяти:



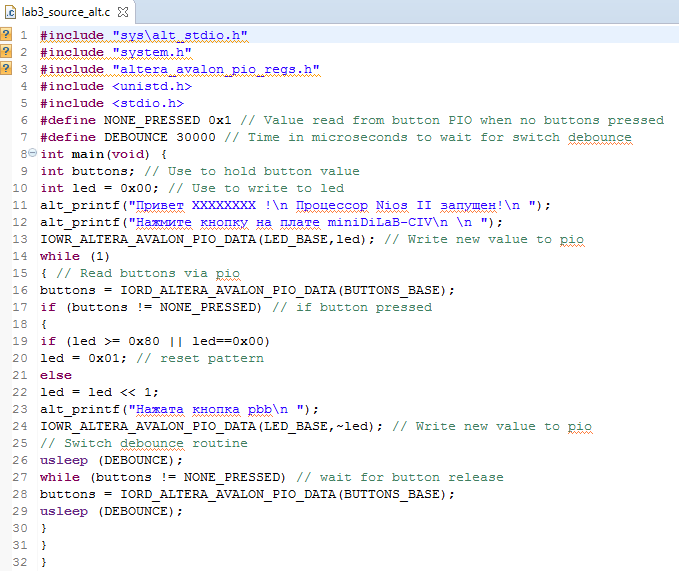
Вернём настройки BSP, предназначенные для уменьшения размера кода:



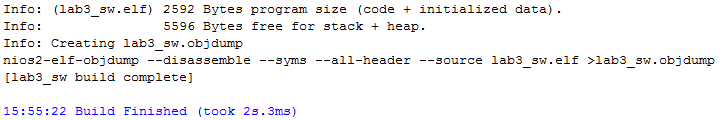
Результаты компиляции показывают, что исходный код программы и инициализационные данные поместились в 4296 байт. При этом под стек и данные осталось 3656 байт:



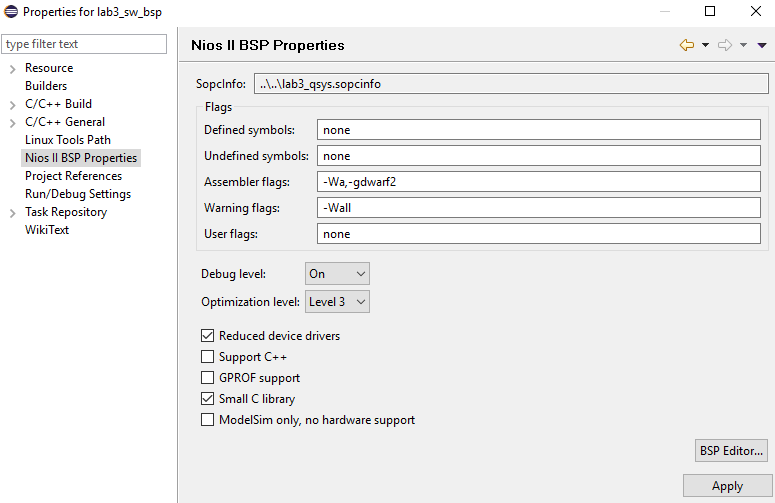
В исходном файле lab3\_source.c подключим alt\_stdio.h и заменим все операторы printf() на alt\_printf():

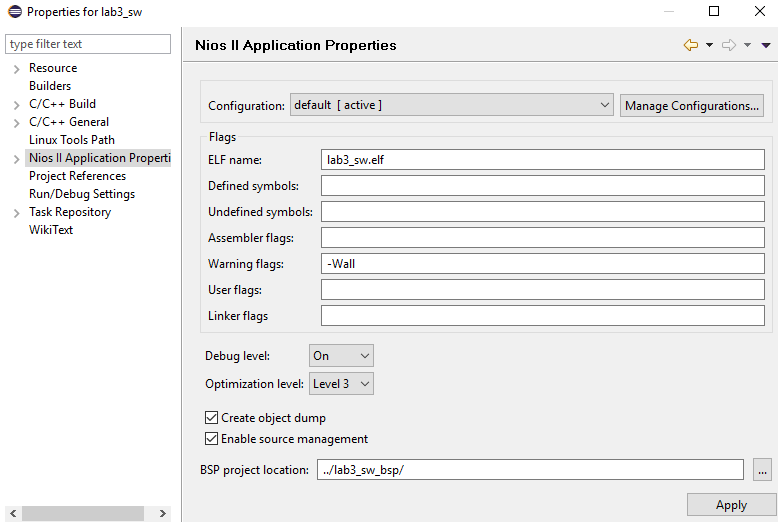


Результаты компиляции показывают, что требование к объему памяти для исходного кода программ и инициализационных данных сократилось до 2592 байт. При этом под стек и данные осталось 5596 байт:

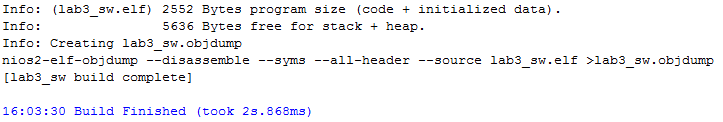


Для lab3\_sw\_bsp и lab3\_sw установим Optimization Level равным Level 3:

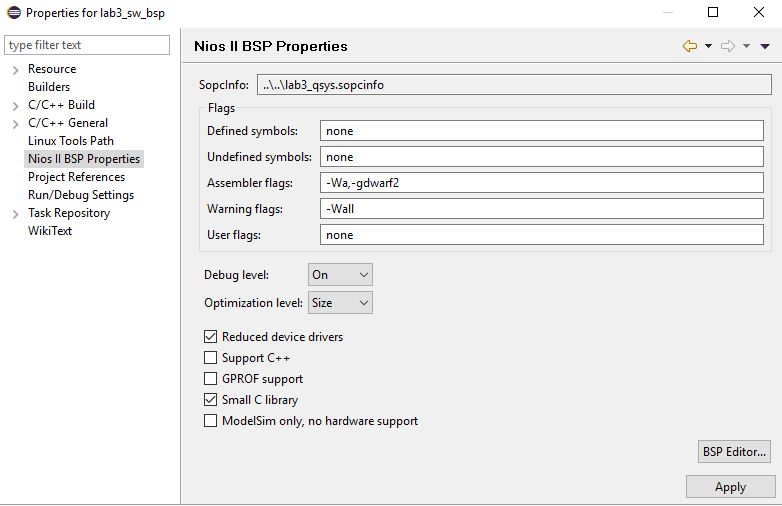


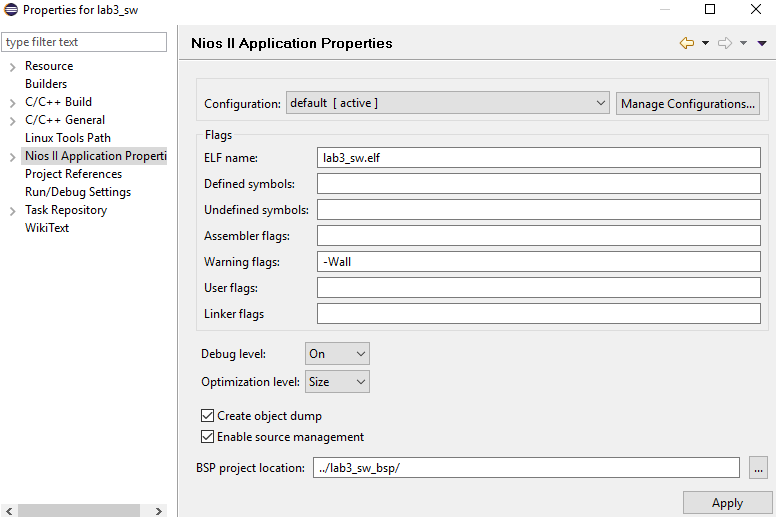


Результаты компиляции показывают, что требование к объему памяти для исходного кода программ и инициализационных данных сократилось до 2552 байт. При этом под стек и данные осталось 5636 байт:

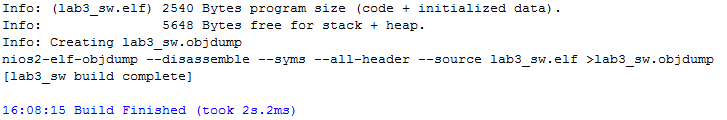


Для lab3\_sw\_bsp и lab3\_sw установим Optimization Level равным Size:





Результаты компиляции показывают, что требование к объему памяти для исходного кода программ и инициализационных данных сократилось до 2540 байт. При этом под стек и данные осталось 5648 байт:



# Выводы

В ходе выполнения данной лабораторной работы было расширено знакомство с возможностями по реализации проектов на базе процессора NIOSII.