

SF2-Junior-KIT

Руководство быстрого старта

Введение

Цель данного руководства – знакомство с циклом разработки проектов для системы-на-кристалле (СнК) Microsemi SmartFusion2 с использованием среды разработки проектов Microsemi Libero SoC.

Данное руководство знакомит со следующими этапами проектирования и программными компонентами среды разработки Libero SoC:

- Libero SoC
 - Маршрут проектирования
 - Smart Design
 - Catalog
- Synopsys Synplify Pro ME
- Microsemi Designer
 - Compile
 - Place and Route
 - I/O Attribute Editor
 - Pin Editor
- FlashPro
- SoftConsole

Необходимое программное обеспечение

Для изучения материала, изложенного в данном руководстве необходимо следующее программное обеспечение:

- среда разработки [Microsemi Libero SoC v11.8](#) с пакетом обновления [SPI](#);
- утилита для программирования микросхем СнК и ПЛИС FlashPro v11.8 или более поздняя версия, которая может быть установлена как часть пакета программ Microsemi Libero SoC и может быть запущена внутри Libero SoC или отдельно;
- среда разработки встраиваемого программного обеспечения SoftConsole v3.4 или более поздняя, которая может быть установлена как часть пакета программ Microsemi Libero SoC или отдельно;
- программа-оболочка HyperTerminal или аналогичное программное обеспечение (PuTTY или TeraTerm);
- драйверы шины USB, которые можно установить, пройдя по ссылке:
http://www.microsemi.com/document-portal/doc_download/131593-usb-uart-driver-files

Необходимое аппаратное обеспечение

Вам понадобится отладочный набор [SF2-Junior-KIT](#), который включает следующие компоненты:

- 1) Модуль SF2-Junior-KIT;
- 2) Жидкокристаллический дисплей 320x240 с интерфейсом SPI и сенсорной панелью (touchscreen);
- 3) Программатор FlashPro4;
- 4) USB – Bluetooth донгл;
- 5) Модуль приемопередатчика Bluetooth – UART;
- 6) Преобразователь напряжения AC-DC 9В 1А;
- 7) Кабель USB 2.0 A-male to mini-B.

Краткий обзор архитектуры СнК SmartFusion2

Структурная схема СнК SmartFusion2 представлена на рис. 1. Основными структурными элементами СнК SmartFusion2 являются микроконтроллерная подсистема (MSS) с большим набором аппаратно реализованных блоков интерфейсов, матрица ПЛИС и системный контроллер с его подсистемой безопасности.

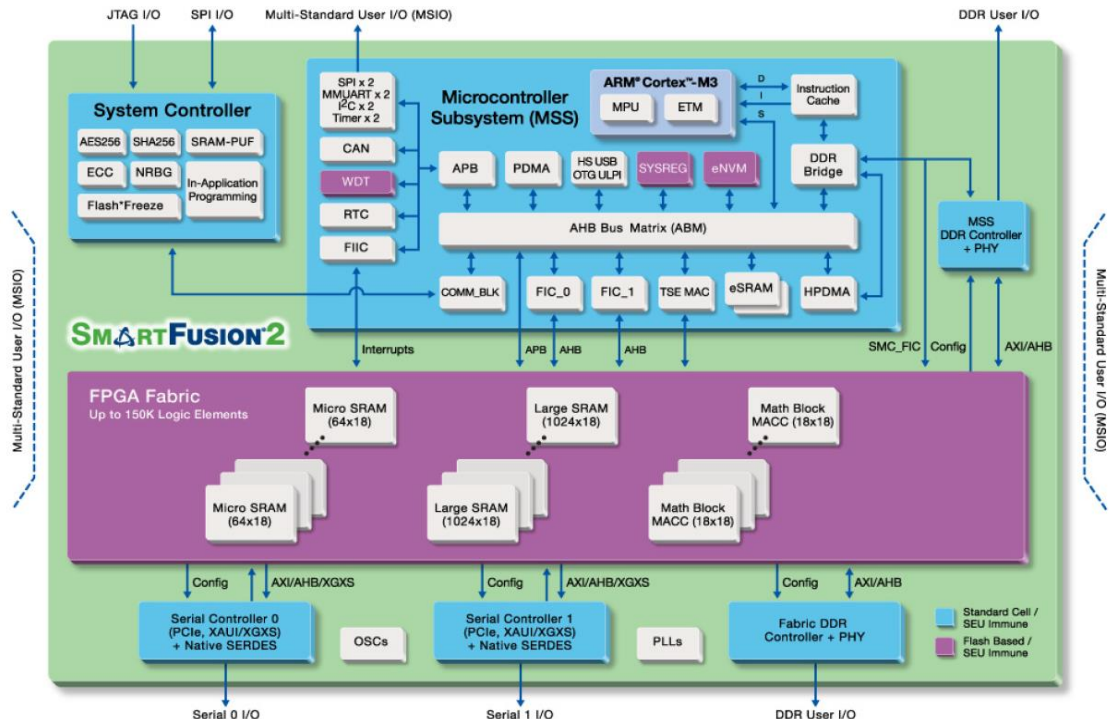


Рис. 1. Структура СнК SmartFusion2

Система на кристалле SmartFusion2 включает в себя процессор 166 MHz ARM® Cortex™-M3, Ethernet MAC, контроллер USB, по два контроллера интерфейсов SPI, I2C, UART, контроллер DMA, часы реального времени, встроенную энергонезависимую память eNVM, программируемую логическую матрицу FPGA Fabric, выполненную по Flash технологии, статическую память SRAM, схему синтеза сигналов тактирования (CCC) и связанный с ней блок фазовой автоподстройки частоты (PLL), встроенные RC-генераторы на 50 МГц и 1 МГц, а также математические блоки (умножитель плюс сумматор 18-разрядных операндов).

Процессор ARM Cortex-M3 имеет контроллер прерываний NVIC. SmartFusion2 имеет 16 выделенных линий прерываний в направлении ПЛИС – MSS MSS_INT_F2M[15:0]. Кроме того, в микроконтроллерной подсистеме имеются 32 ввода-вывода общего назначения, которые могут быть сконфигурированы как входные входы, выходы или как двунаправленные контакты. Будучи сконфигурированными как источники входного сигнала они могут быть использованы как дополнительные источники прерываний.

Обзор примера

В рассматриваемом примере используется аппаратная реализация интерфейса УАПП (универсальный асинхронный приемопередатчик) MSS_UART_0 для вывода тестового сообщения в программе терминале на персональном компьютере.

После выполнения необходимы подключений, подачи питания на отладочный набор, загрузки конфигурационной последовательности FPGA в матрицу и кода встроенного программного обеспечения процессора в память SRAM процессора, в окне программы-терминала, запущенной на персональном компьютере появляется сообщение “Hello World!”, в окне терминала отображаются символы нажатых на клавиатуре ПК клавиш. Одновременно на печатной плате отладочного набора мигают два светодиода, причем управление свечением первого светодиода реализовано в коде встраиваемого программного обеспечения процессора Cortex-M3, а мигание вторым осуществим с помощью матрицы ПЛИС.

Для реализации описанных функций используются следующие ресурсы СнК SmartFusion2:

- Микроконтроллерная подсистема MSS с процессором ARM Cortex-M3;
- Аппаратно реализованный блок интерфейса УАПП MMUART_0 микроконтроллерной подсистемы;
- Один контакт GPIO микроконтроллерной подсистемы;
- 50 МГц RC-генератор;
- Генератор сигнала сброса SYSRESET;
- Ресурсы матрицы FPGA Fabric.

Создание проекта системы на кристалле

Запустите приложение Libero SoC 11.8. дважды кликнув на ярлычок на рабочем столе или на аналогичный в меню Пуск > Все программы > Microsemi > Libero SoC v11.8.

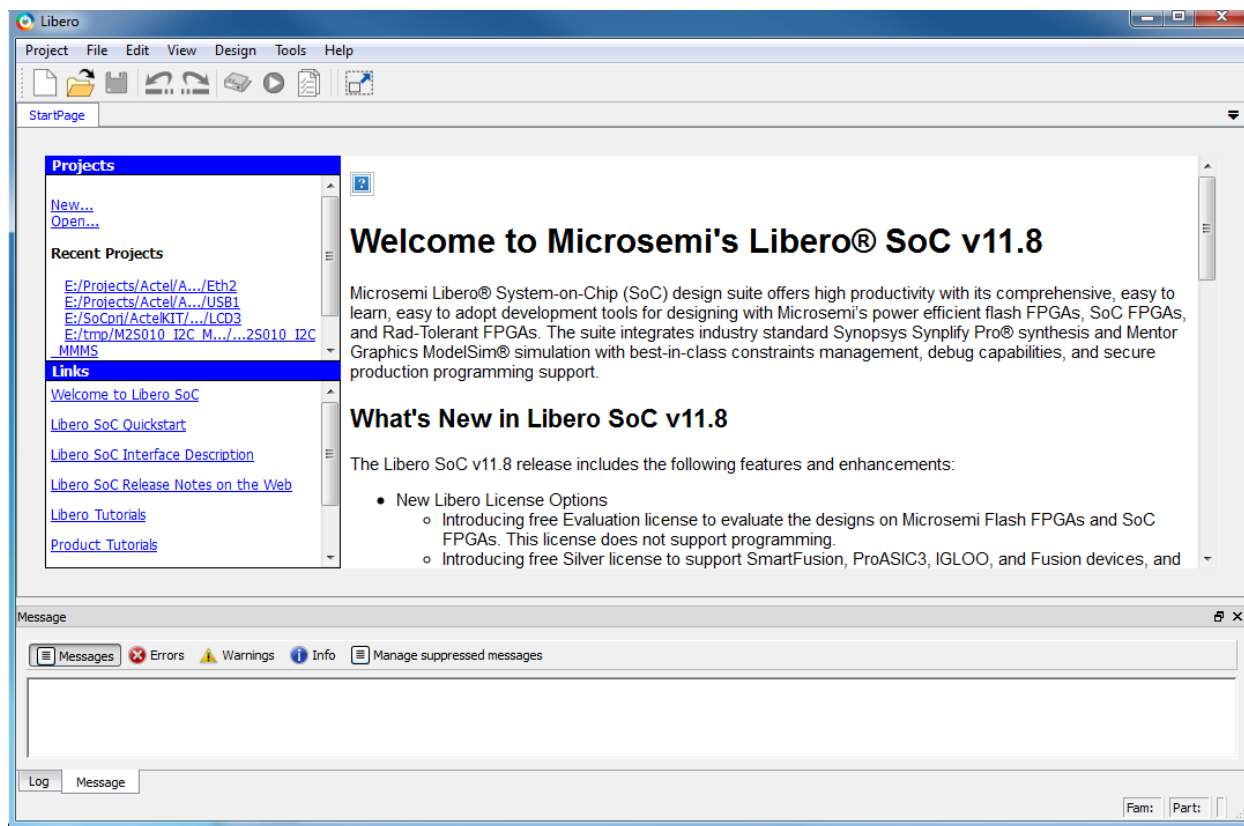


Рис. 2.

В главном меню LiberoSoC выполните команду **Project/New Project**, запустится мастер создания нового проекта. В появившемся окне укажите название проекта, например **MyFirstProject**, место расположения нашего проекта на диске и предпочитаемый язык проектирования - VHDL. Опцию Enable block creation в рамках данного примера устанавливать не нужно. Нажмите кнопку «Next».

В появившемся окне «Device selection» выбором желаемых параметров в выпадающих списках укажите PartNumber микросхемы, с которой будем работать. При работе с отладочным комплектом SF2-Junior-KIT необходимо выбрать **M2S010-TQ144**, после чего нажать «Next».

В следующем окне выберите настройки стандарта ввода-вывода по умолчанию LVCMOS 2.5V, напряжение питания PLL 2.5 V и величину задержки старта микросхемы после сигнала Reset 100 ms. Нажмите кнопку «Next».

В следующем появившемся окне предлагается выбрать мастер, который будет использоваться для настройки микроконтроллерной подсистемы. Необходимо выбрать пункт «Create a microcontroller (MSS) based design», после чего нажать «Next». При выборе способа установки проектных ограничений нажмите кнопку «Use Enhanced Constrain Flow». В следующих появляющихся окнах ничего не меняем, просто нажимаем «Next» до появления окна изображенного на рис. 6.

Созданный мастером компонент **MyFirstProject_MSS_0** отражает состояние настроек микроконтроллерной подсистемы «по умолчанию». Нам необходимо изменить эти настройки в соответствии с задачами, решаемыми нашим приложением.

Наше приложение будет отправлять сообщение «Hello, world!», используя аппаратный интерфейс UART микроконтроллерной подсистемы, и мигать светодиодами, причем включение/выключение первого светодиода будет выполнять процессор Cortex-M3 в соответствии с заложенной программой, а мигание вторым светодиодом осуществляет счетчик тактовых импульсов, реализованный в матрице FPGA Fabric.

Настроим соответствующие блоки архитектуры MSS, для этого дважды щелкнем на компоненте MyFirstProject_MSS_0. Откроется окно MyFirstProject_MSS_0 настроек микроконтроллерной подсистемы. В нем мы видим все доступные компоненты MSS.

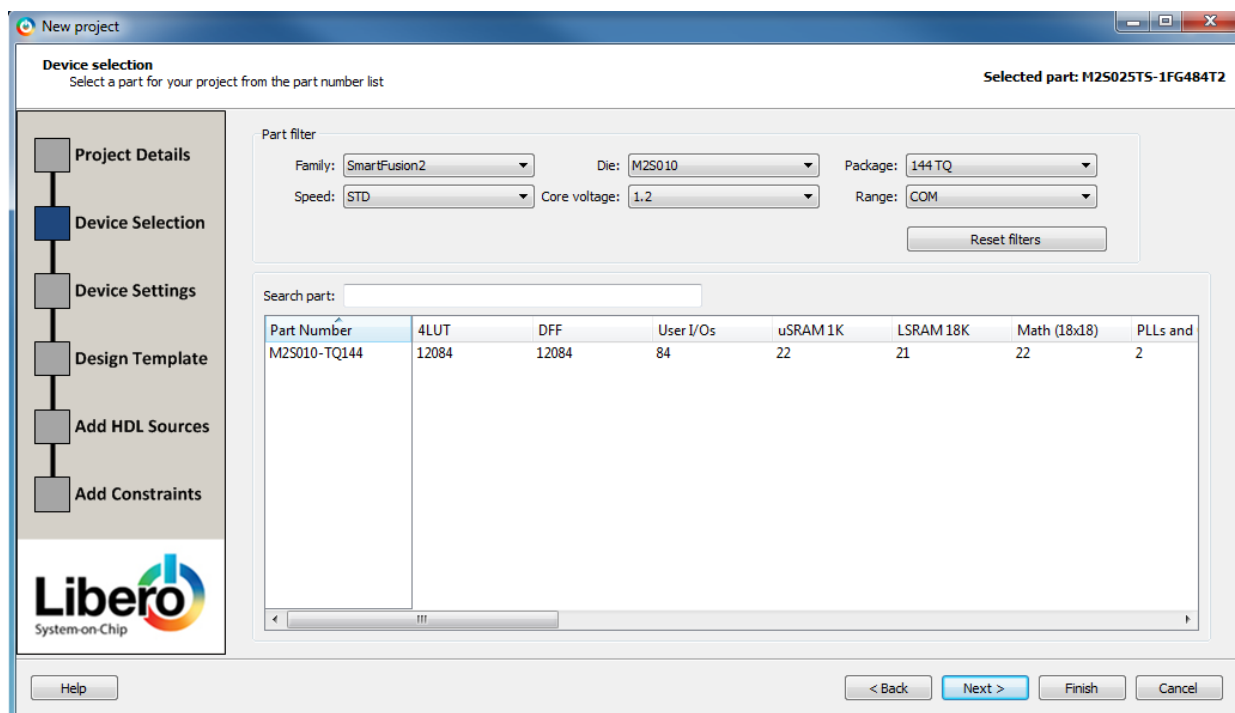


Рис. 3.

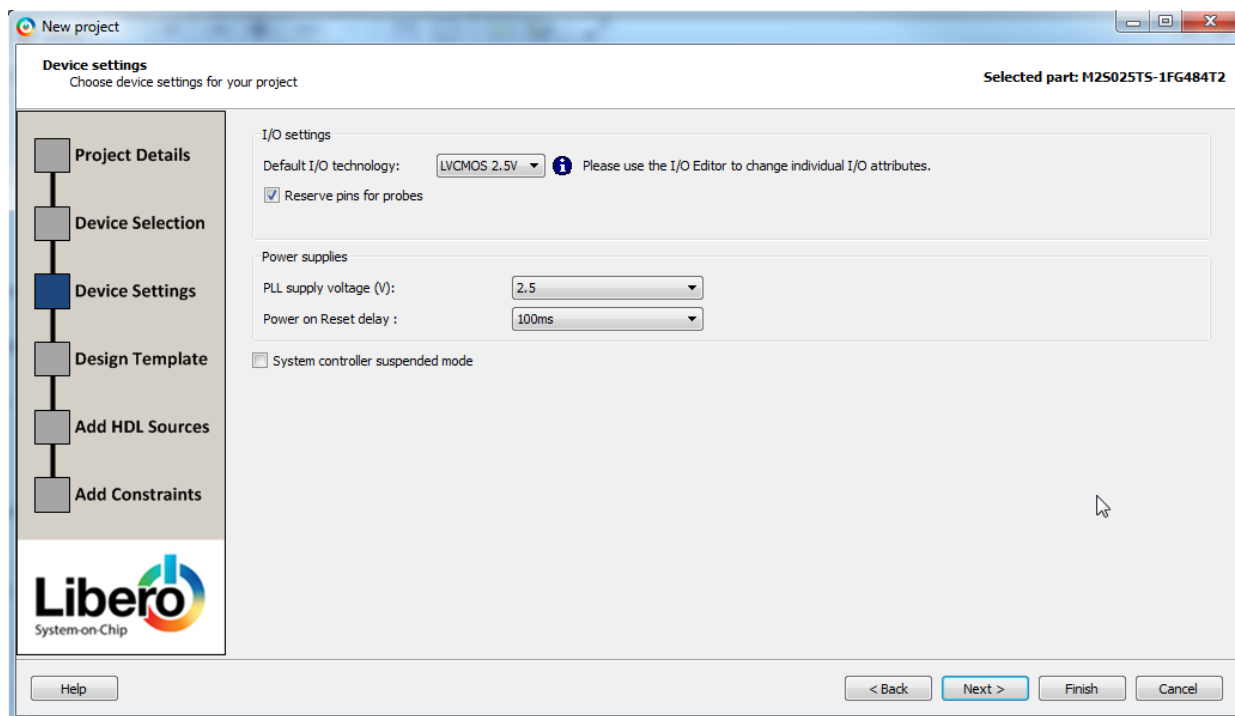


Рис. 4.

В рамках создаваемого проекта необходим один контроллер интерфейса UART, например MMUART_0, и один контакт общего назначения GPIO, подключенный к микроконтроллерной подсистеме. Все остальные компоненты MSS, а именно: USB, Ethernet, MMUART_1, SPI_0, SPI_1,

I2C_0, I2C_1, PDMA, CAN, WatchDog, FIC_0, FIC_1 в рамках нашего первого проекта не понадобятся, их необходимо отключить, т. е. снять галочку в правом нижнем углу перечисленных компонентов (рис. 7).

Теперь настроим используемые в проекте компоненты. Для этого дважды щелкнем на блоке MMUART_0. В появившемся окне выберем следующие опции (рис. 8):

Duplex Mode: Full Duplex.

Async/Sync Mode: Asynchronous.

Use Modem Interface: снять галочку

RHD: Fabric

TXD: Fabric

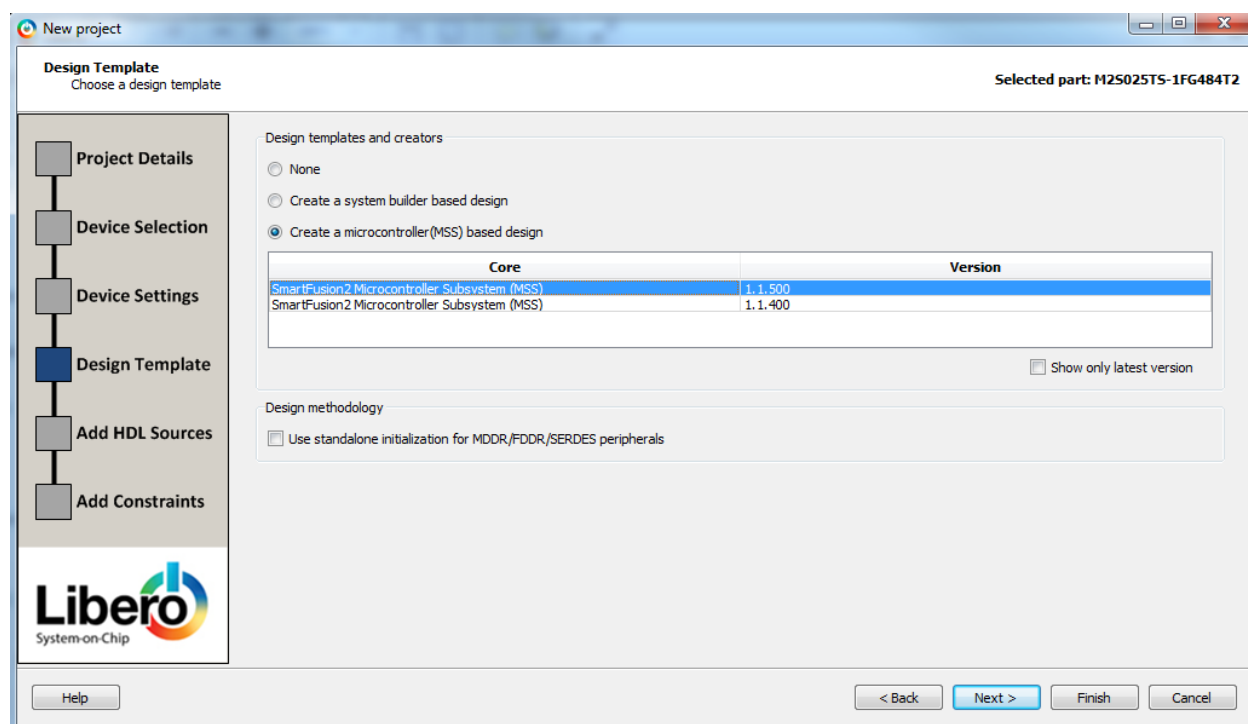


Рис. 5.

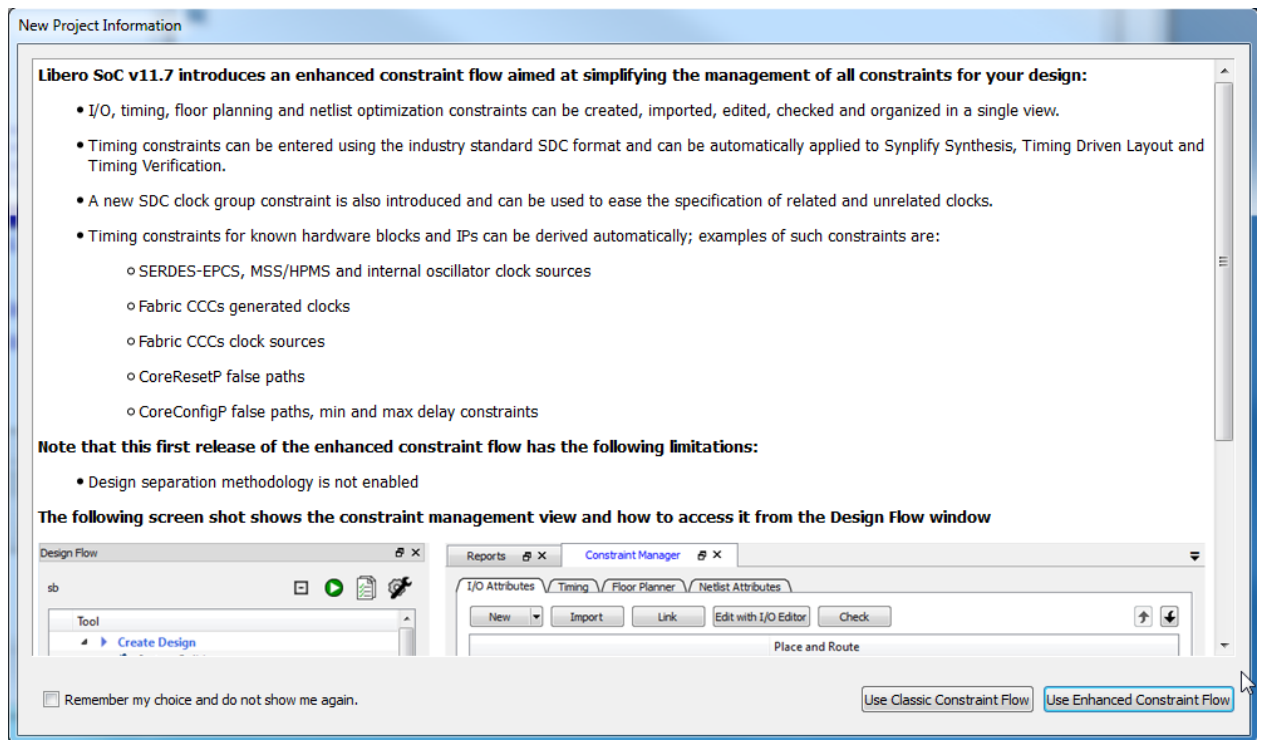


Рис. 6.

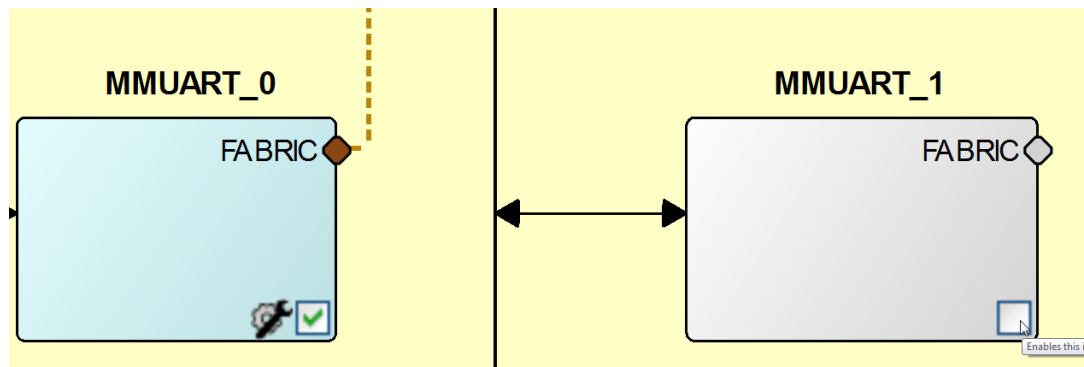


Рис. 7.

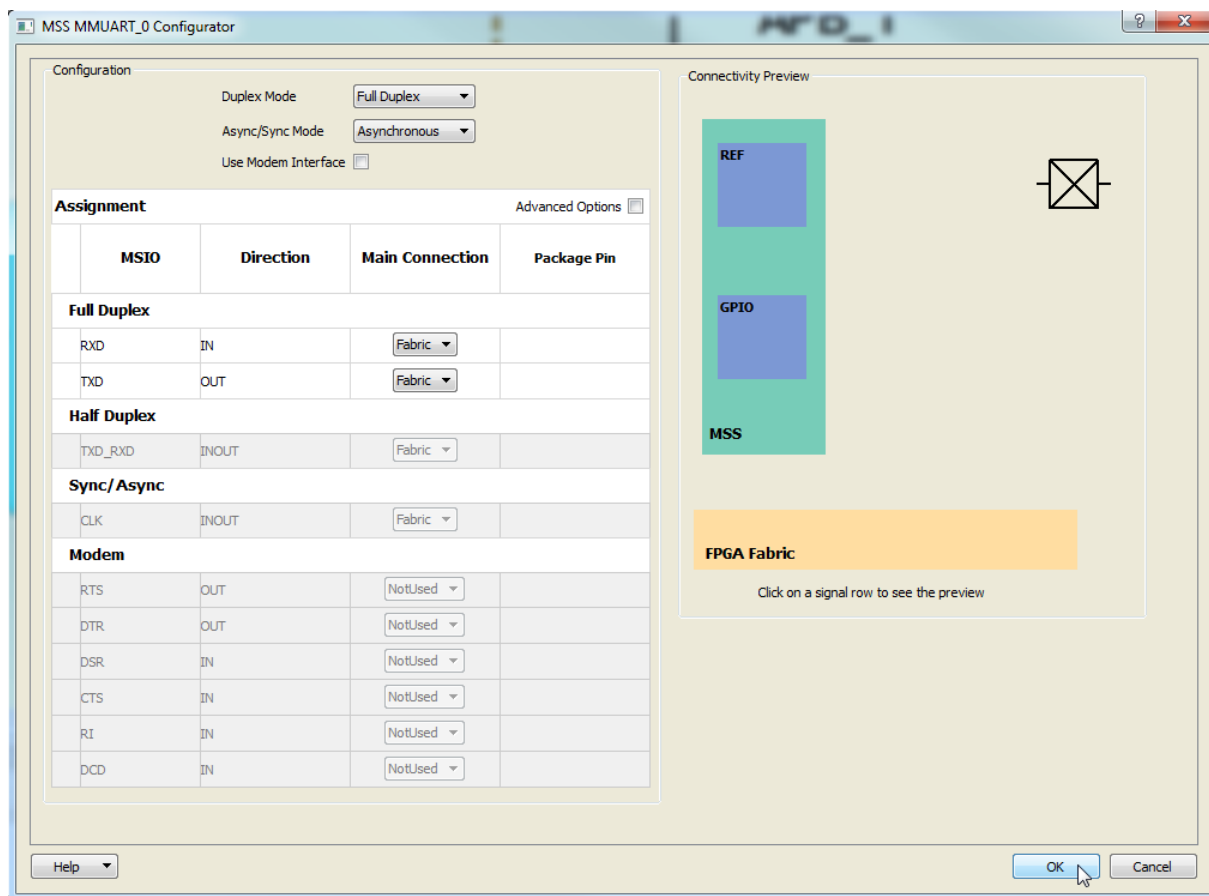


Рис. 8.

Для подключения к MSS контакта ввода-вывода микросхемы, включите компонент GPIO установкой галочки в нижнем левом углу компонента, и дважды щелкните на компоненте левой кнопкой мыши для запуска мастера редактирования его свойств. В появившемся окне MSS GPIO Configurator измените свойство Direction контакта GPIO_0 с «Not Used» на «Output» при этом окно настроек компонента GPIO примет вид показанный на рис. 9.

Настройки подсистемы тактирования микроконтроллерной подсистемы MSS_CCC оставьте без изменения.

Для обеспечения работы нашего проекта остается настроить подсистему сброса (MSS_Reset) микроконтроллерной подсистемы. Указанная подсистема позволяет реализовать различные способы прохождения сигналов сброса внутри системы. В рамках нашего проекта установите настройки, показанные на рис. 10.

После выполнения описанных действий окно настроек микроконтроллерной подсистемы должно выглядеть, как показано на рис. 11.

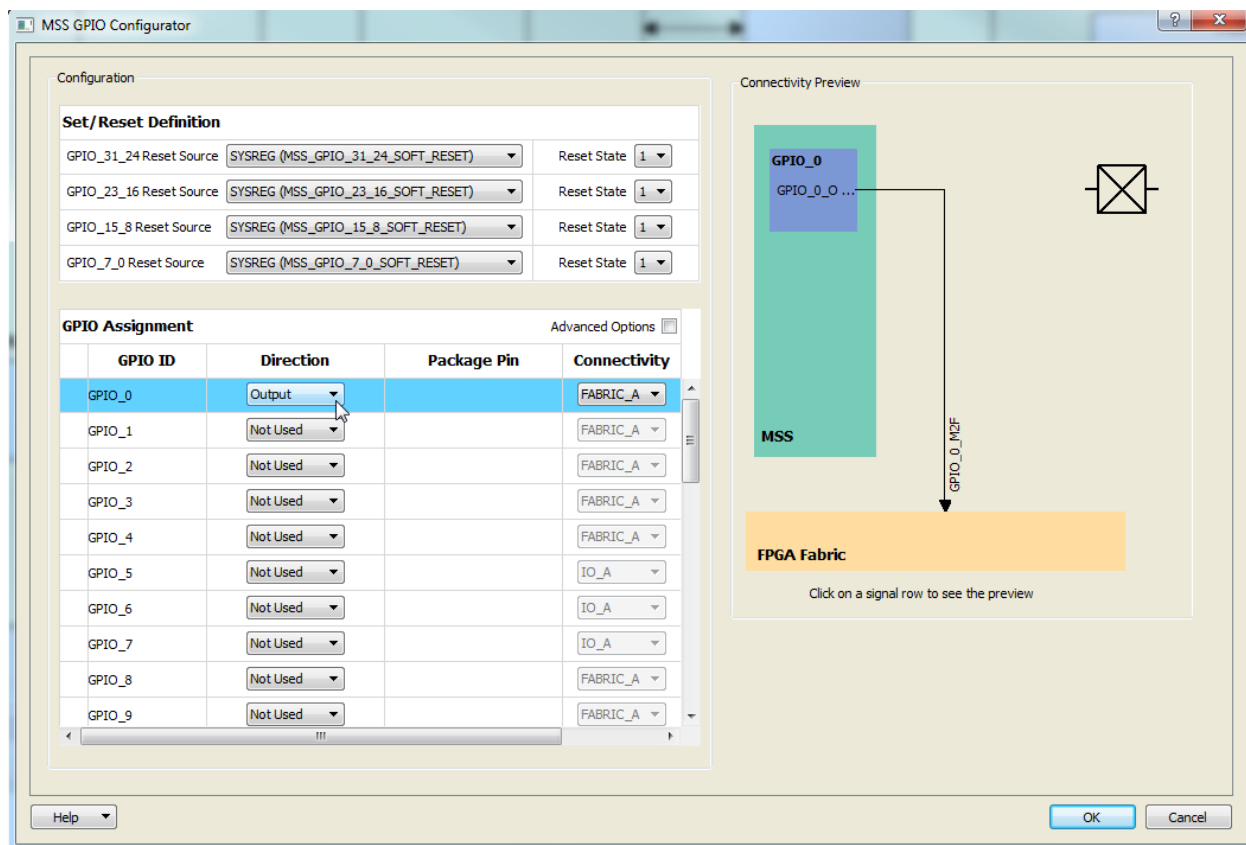


Рис. 9.

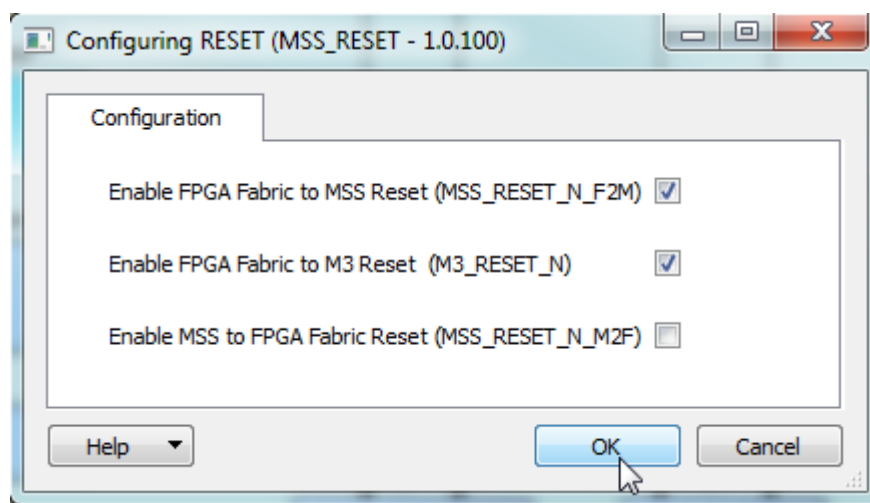



Рис. 10.

Выполните генерацию компонента Микроконтроллерной подсистемы с нашими настройками, выполнив команду SmartDesign/Generate Component основного меню, или нажав горячую клавишу быстрого меню , после чего сохраните текущие настройки, выполнив команду Project/Save.

Переходим в окно проектирования верхнего уровня нашего проекта MyFirstProject. После настройки микроконтроллерной подсистемы внешний вид компонента MyFirstProjec_MSS_0 изменился – появился восклицательный знак на желтом фоне в правом верхнем углу компонента (рис. 12). Это означает, что описание верхнего уровня компонента MyFirstProjec_MSS_0 нужно обновить. Для этого нажмите правую кнопку мыши на компоненте и в появившемся меню выберите команду Update Instance with Latest Component.

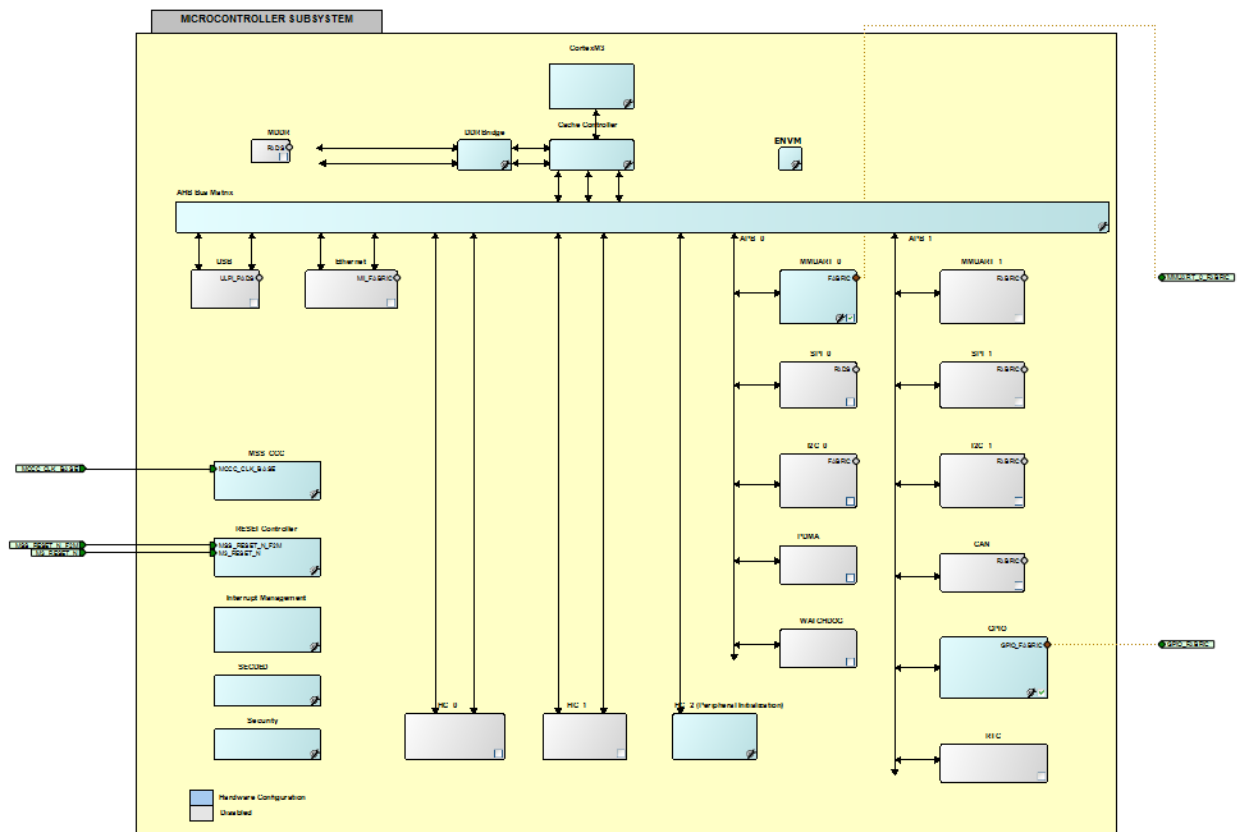


Рис. 11.

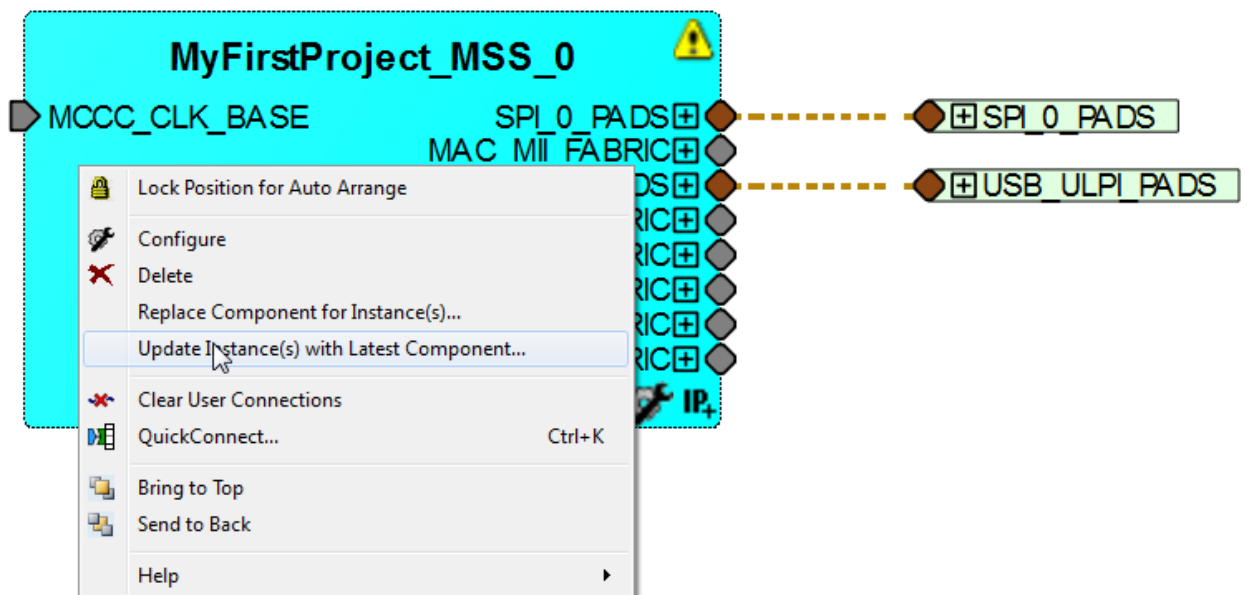


Рис. 12.

После обновления компонента его внешний вид приходит в соответствие введенными нами настройками, то есть отображаются только включенные нами блоки и сигналы (UART, GPIO, Reset, Clock), все отключенные нами блоки не видны (рис. 13).

Теперь из стандартного каталога Libero SoC (рис. 14) нужно добавить IP-ядра и компоненты, отвечающие за тактирование, системный сброс и мигание светодиодом. Для генерации сигнала тактирования будем использовать компонент **Chip Oscillators** палитры **Clock**

& **Management** из каталога. Окно настройки компонента вызывается, двойным щелчком левой кнопки мыши на компоненте.

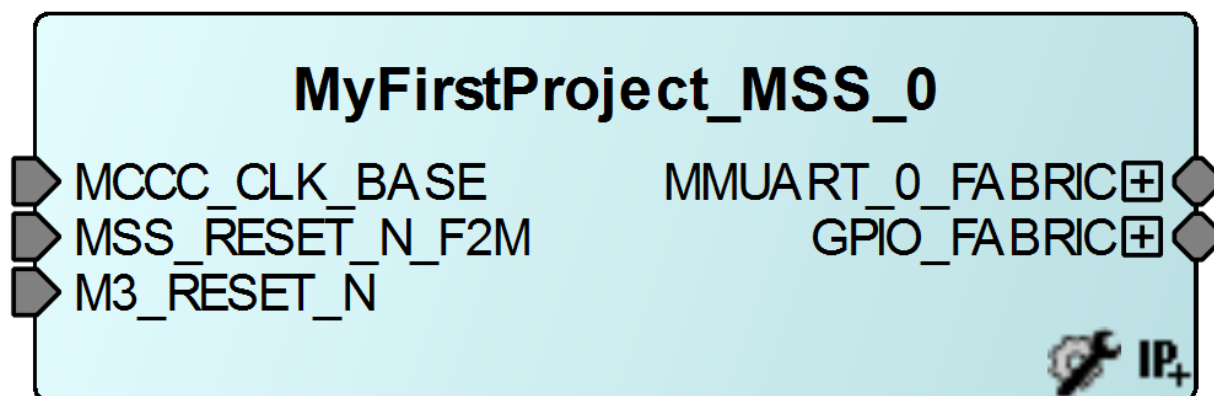


Рис. 13.

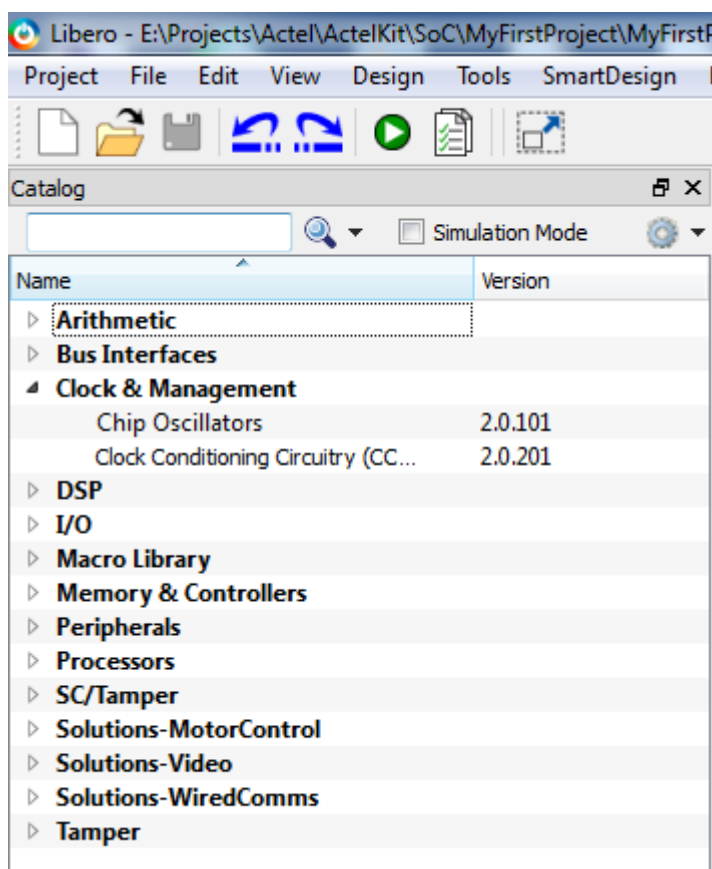


Рис. 14.

Для тактирования работы СнК SmartFusion2 можно использовать как внешние кварцевые генераторы, подключаемые к выделенным или обычным контактам микросхемы, так и внутренние RC-генераторы имеющие частоты 1 и 50 МГц. Для нашего первого проекта будем использовать внутренний генератор на 50 МГц. Тактирующий сигнал нужно подать на пользовательскую логику как показано на рис. 15.

Компонент **SYSRESET**, отвечающий за системный сброс, находится в палитре компонентов **Macro Library** стандартного каталога, и в настройке не нуждается.

Теперь создадим компонент ПЛИС реализующий мигание вторым светодиодом. Для выбранной нами частоты внутреннего генератора 50 МГц для обеспечения заметного

человеческому глазу мигания светодиода можно использовать счетчик на 23 разряда. Создадим файл HDL-описания счетчика, выполнив команду **File>New>HDL**. В появившемся окне выберем язык HDL-описания (VHDL или Verilog) и введем имя создаваемого компонента. По нажатию кнопки «OK» кнопки среда создаст текстовый файл, в котором нужно ввести HDL-описание нашего счетчика на выбранном нами языке описания. Текст HDL-описания счетчика на языке VHDL представлен на рис. 16 и в файле MyCounter.vhd в папке /Source архива файлов проекта.

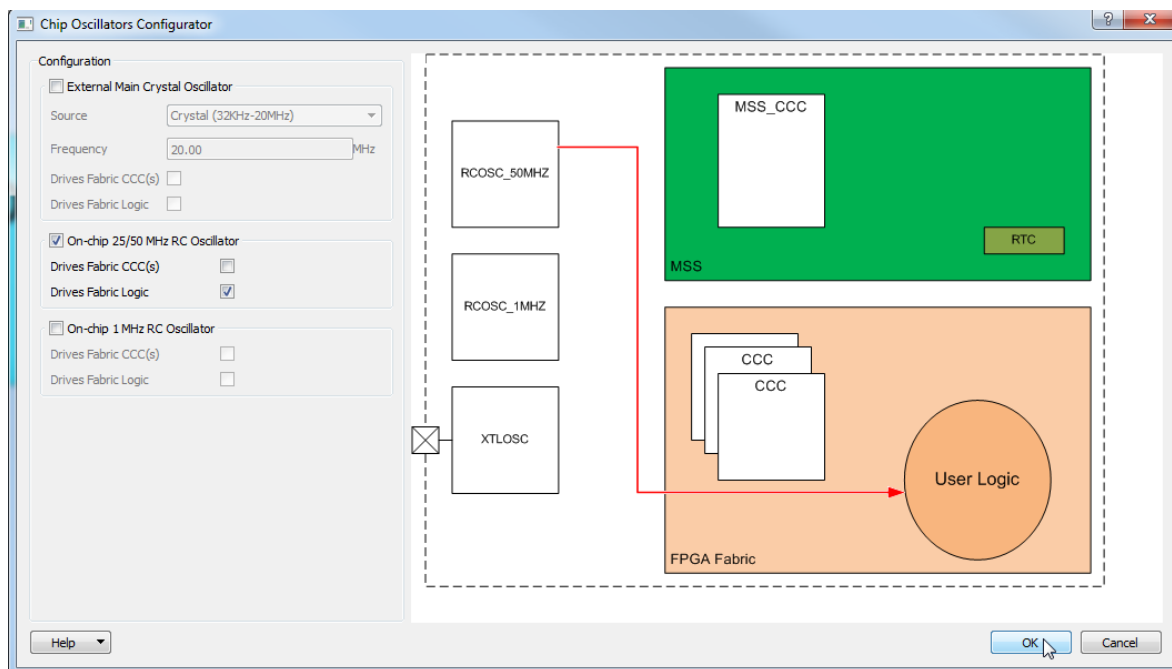


Рис. 15.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_arith.all;
4  use IEEE.std_logic_unsigned.all;
5
6  entity MyCounter is
7  port(Clock : in std_logic;
8       Q : out std_logic_vector(22 downto 0);
9       Aclr : in std_logic;
10      Tcnt : out std_logic);
11 end MyCounter;
12
13 architecture behavioral of MyCounter is
14     signal Qaux : UNSIGNED(22 downto 0);
15 begin
16     process(Clock, Aclr)
17     begin
18         if (Aclr = '0') then
19             Qaux <= (others => '0');
20         elsif (Clock'event and Clock = '1') then
21             Qaux <= Qaux + 1;
22         end if;
23     end process;
24     Q <= std_logic_vector(Qaux);
25     process(Qaux)
26     variable aux : std_logic;
27     begin
28         aux := '0';
29         if Qaux = 8 then aux := '1';
30         end if;
31         Tcnt <= aux;
32     end process;
33 end behavioral;
34

```

Рис. 16.

Одновременно с сохранением файла HDL описания во вкладке Design Hierarchy появилось название созданного нами компонента, таким образом, мы получили возможность использовать созданный компонент в нашем проекте. Перенесем мышкой название компонента на рабочее поле проекта, аналогично тому, как мы это делали с компонентами из стандартного каталога Libero SoC.

Для реализации в проекте возможности рестарта встроенного программного обеспечения процессора по нажатию кнопки, расположенной на плате комплекта, добавим в проект элемент **2AND** из раздела **Macro Library** стандартного каталога.

Чтобы выполнить соединения контактов используемых компонент необходимо при нажатой кнопке клавиатуры «Ctrl» выделить мышкой контакты, которые собираемся соединить, затем на одном из выбранных контактов нажать правую кнопку мыши, в появившемся меню выбрать пункт «Connect». Ниже приведен список соединений проекта в формате IP-ядро.Контакт – IP-ядро.Контакт:

1. OSC_0.RCOSC_25_50MHZ_O2F – MyFirstProject_MSS_0. MCCC_CLK_BASE.
2. OSC_0.RCOSC_25_50MHZ_O2F – MyCounter_0.Clock.
3. SYSRESET_0.POWER_ON_RESET_N – AND2_0.B
4. AND2_0.Y – MyFirstProject_MSS_0.MSS_RESET_N_F2M
5. AND2_0.Y – MyFirstProject_MSS_0. M3_RESET_N
6. AND2_0.Y – MyCounter.AcIr.

Пункты, имеющие общий контакт образуют единую цепь прохождения сигнала.

Выход счетчика в нашем проекте имеет разрядность [22:0] , для зажигания светодиода мы будем использовать только самый старший разряд. Следовательно, этот разряд нужно вычлнить из группы. Для этого нажимаем на контакте Q[22:0] правой кнопкой мыши и выбираем пункт **Edit Slice**. В появившемся окне нажимаем кнопку **Add Slice Entry** (зеленый знак «плюс») и заполняем таблицу по образцу, представленному на рис. 17.

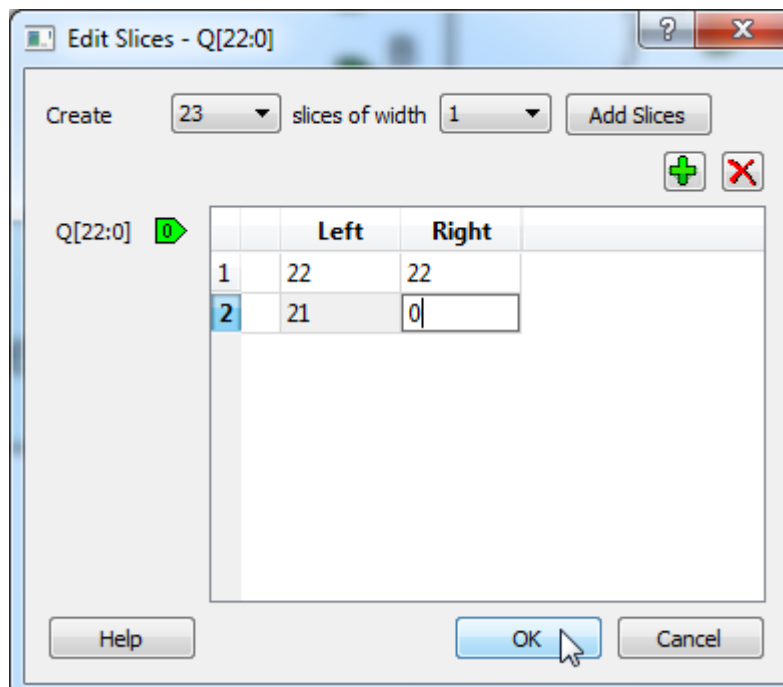


Рис. 17.

Выводим нужные сигналы на верхний уровень для дальнейшего подключения к контактам СнК. Для этого последовательно нажимаем на контакты Q[22] компонента MyCounter_0, MMUART_0_FABRIC и GPIO_FABRIC компонента MyFirstProject_MSS_0, контакт A компонента AND2_0, в выпадающем меню выбираем пункт **Promote to Top Level**.

Неиспользуемые контакты компонентов MyCounter.Tcnt и MyCounter.Q[21:0] следует пометить Mark Unused выбором соответствующего пункта в меню, выпадающем при нажатии правой кнопки мыши на контакте.

Переименуем контакты верхнего уровня для лучшей читабельности схемы: контакт «A» в «BtnReset», контакт «Q» в «FPGA_LED», контакт «GPIO_0_M2F» в «M3_LED».

В результате описанных действий верхний уровень проекта приобретает вид, показанный на рис. 18.

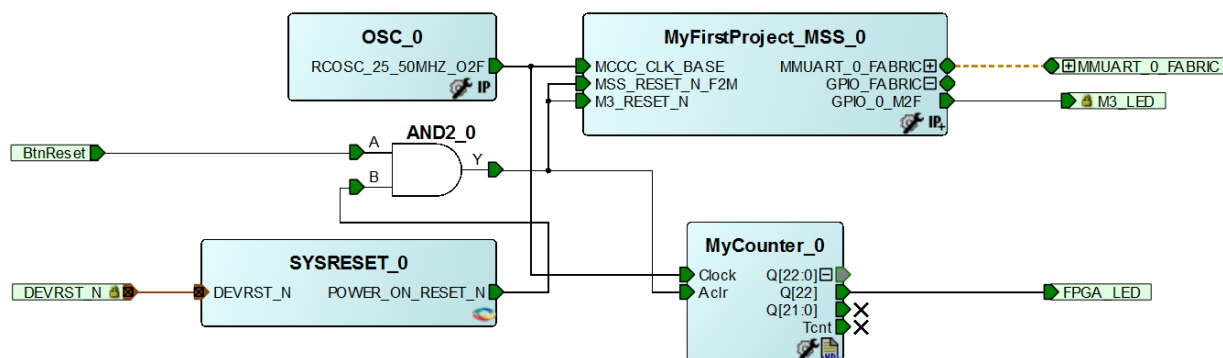


Рис. 18.

Выполним команды меню «Save» и «Generate Component», затем Synthesize > Clean and Run All.

Переходим во вкладку DesignFlow и выполняем команду Constraints > Manage Constraints > Edit with I/O Editor. Назначаем номера контактов внешним соединениям проекта (рис. 19).

I/O Editor - MyFirstProject							
File Edit View Tools Help							
Ports Package Pins Package Viewer							
	Port Name	Direction	I/O Standard	Pin Num	Locked	Macro Cell	Bank Name
1	BtnReset	Input	LVC MOS25	13	<input checked="" type="checkbox"/>	ADLIB:INBUF	Bank7
2	DEVRST_N	Input	--	72	<input checked="" type="checkbox"/>	ADLIB:SYSRESET	--
3	FPGA_LED	Output	LVC MOS25	67	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank4
4	M3_LED	Output	LVC MOS25	63	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank4
5	MMUART_0_RXD_F2M	Input	LVC MOS25	9	<input checked="" type="checkbox"/>	ADLIB:INBUF	Bank7
6	MMUART_0_TXD_M2F	Output	LVC MOS25	10	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank7

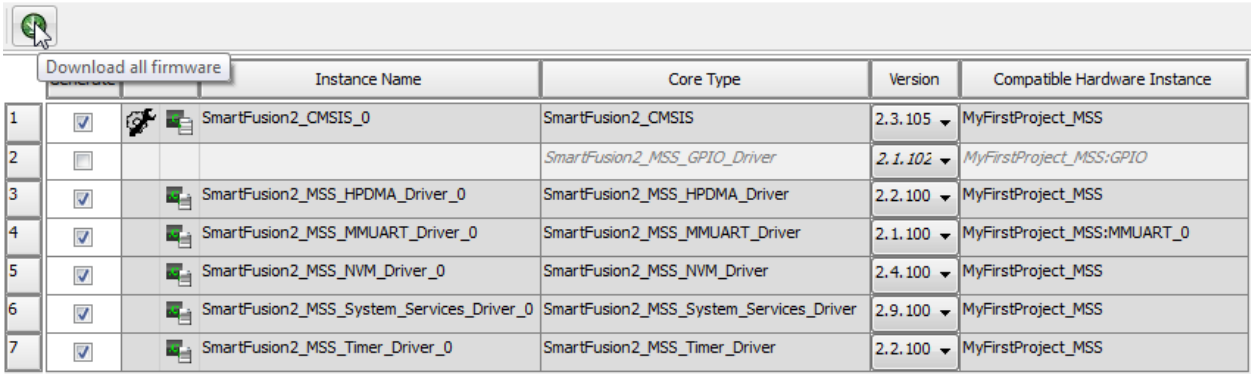
Рис. 19.

В рамках данного примера ввиду простоты проекта мы опускаем моделирование дизайна в симуляторе Modelsim, входящим в пакет Libero SoC. Разумеется, при разработке более сложных проектов пропускать исследования поведенческой модели не следует.

Подключаем программатор к плате отладочного комплекта, подаем на плату питание, запускаем команду Program Design>Run PROGRAM Action. В результате выполнения данной команды будет создан файл конфигурационной последовательности, который будет загружен в кристалл. Об окончании процесса можно судить по зажиганию зеленого светодиода на программаторе и миганию красного светодиода на плате отладочного набора подключенного к контакту FPGA_LED.

Разработка встроенного программного обеспечения процессора Cortex-M3.

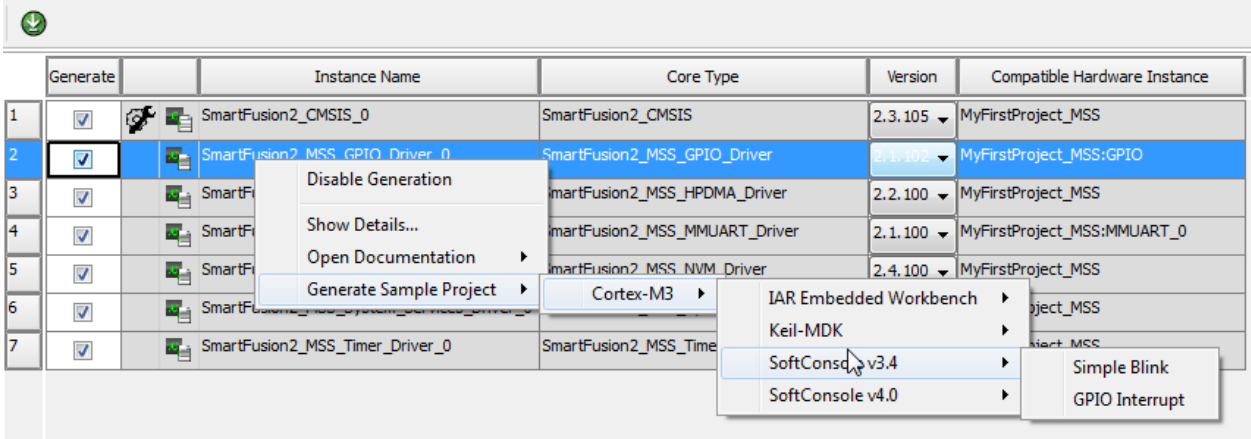
Выполняем конфигурирование драйверов блоков архитектуры. Для этого в окне Design Flow выполняем команду **Configure Firmware cores**, в появившемся окне выполняем команду **Download all firmware** (рис. 20).



		Instance Name	Core Type	Version	Compatible Hardware Instance
1	<input checked="" type="checkbox"/>	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.3.105	MyFirstProject_MSS
2	<input checked="" type="checkbox"/>	SmartFusion2_MSS_GPIO_Driver_0	SmartFusion2_MSS_GPIO_Driver	2.1.102	MyFirstProject_MSS:GPIO
3	<input checked="" type="checkbox"/>	SmartFusion2_MSS_HPDMADriver_0	SmartFusion2_MSS_HPDMADriver	2.2.100	MyFirstProject_MSS
4	<input checked="" type="checkbox"/>	SmartFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.1.100	MyFirstProject_MSS:MMUART_0
5	<input checked="" type="checkbox"/>	SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.4.100	MyFirstProject_MSS
6	<input checked="" type="checkbox"/>	SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.9.100	MyFirstProject_MSS
7	<input checked="" type="checkbox"/>	SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.2.100	MyFirstProject_MSS

Рис. 20.

В составе среды разработки Libero SoC поставляются примеры проектов встраиваемого программного обеспечения (ВПО), демонстрирующие работу основных компонент микроконтроллерной подсистемы. Посмотрите, какие демонстрационные проекты прилагаются каждому блоку архитектуры микроконтроллерной подсистемы, эти примеры могут оказаться полезны при разработке собственных проектов. Для генерации файлов проекта примера необходимо нажать правой кнопкой мыши на драйвере соответствующего блока и выбрать интересующий пример (рис. 21).



	Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
1	<input checked="" type="checkbox"/>	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.3.105	MyFirstProject_MSS
2	<input checked="" type="checkbox"/>	SmartFusion2_MSS_GPIO_Driver_0	SmartFusion2_MSS_GPIO_Driver	2.1.102	MyFirstProject_MSS:GPIO
3	<input checked="" type="checkbox"/>	SmartFusion2_MSS_HPDMADriver_0	SmartFusion2_MSS_HPDMADriver	2.2.100	MyFirstProject_MSS
4	<input checked="" type="checkbox"/>	SmartFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.1.100	MyFirstProject_MSS:MMUART_0
5	<input checked="" type="checkbox"/>	SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.4.100	MyFirstProject_MSS
6	<input checked="" type="checkbox"/>	SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.9.100	MyFirstProject_MSS
7	<input checked="" type="checkbox"/>	SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.2.100	MyFirstProject_MSS

Рис. 21.

Теперь создадим наш проект встраиваемого программного обеспечения (ВПО). Для этого выполните команду **Export Firmware** во вкладке Design Flow. В появившемся окне (рис. 21) можно выбрать среду разработки ВПО, в которой в дальнейшем будем работать. Microsemi предлагает четыре варианта среды для разработки ВПО: SoftConsole 3.4, SoftConsole 4.0, IAR EWARM и Keil-MDK. Мы в рамках данного примера выбираем **SoftConsole 3.4**, ставим галочку **Create software including hardware configuration and firmware drivers** и нажимаем OK (рис. 22).

Во вкладке «Files» проекта контролируем появление каталога SoftConsole с каталогами и файлами созданного нами проекта.

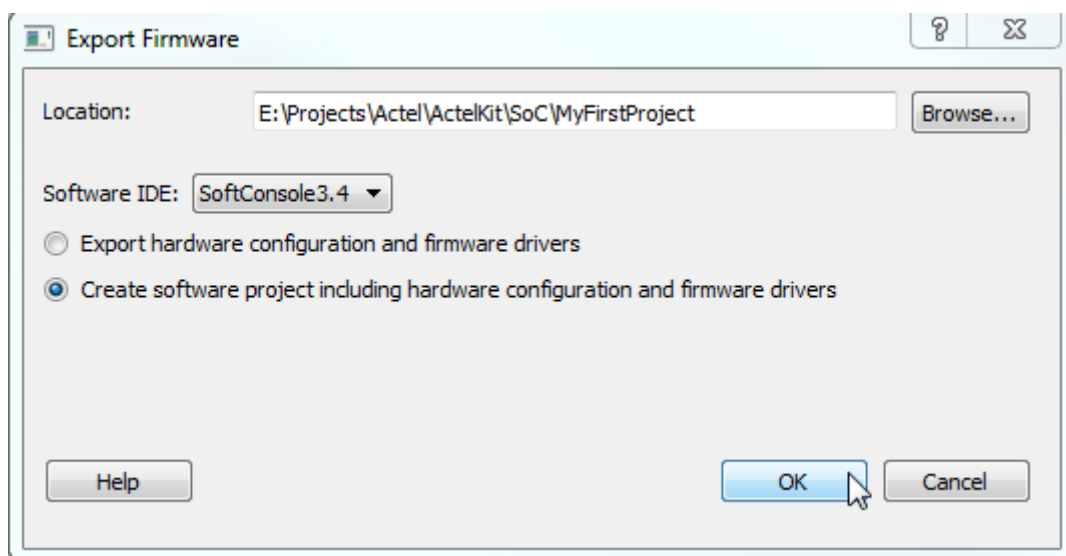


Рис. 22.

Файл с исходным кодом встроенного программного обеспечения доступен в папке Source прилагаемого к данному описанию [архива](#). Копируем его содержимое в файл **main.c** созданного проекта ВПО.

Теперь необходимо отладить наше приложение, получить файл и исполняемым кодом и скомпоновать файлы конфигурационной последовательности ПЛИС с исполняемым файлом встраиваемого приложения в единый файл прошивки. Для нас понадобится среда разработки и отладки встраиваемого ПО SoftConsole v3.4.

При старте SoftConsole просит указать проект, с которым мы собираемся работать. Указываем расположение папки LedUART_MSS_CM3 как показано на рис. 23.

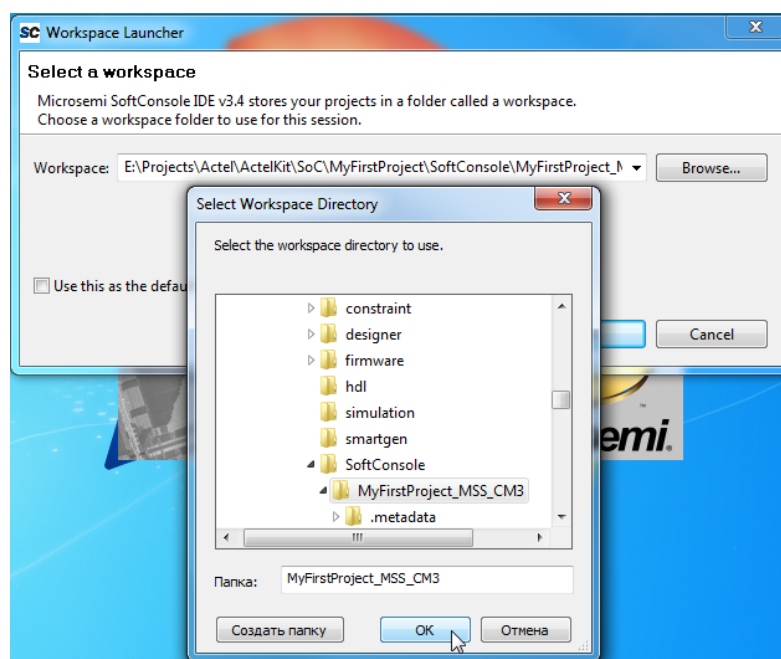


Рис. 23.

После этого откроется основное окно среды разработки SoftConsole, которое должно выглядеть так, как показано на рис. 24. Если во вкладке Project Explorer отсутствует проект его нужно загрузить «вручную», выполнив команды File>Import, в открывшемся окне во вкладке General выбрать пункт Existing Project into Workspace и в следующем появившемся окне указать расположение проекта Firmware. После чего окно примет вид, показанный на рис. 24.

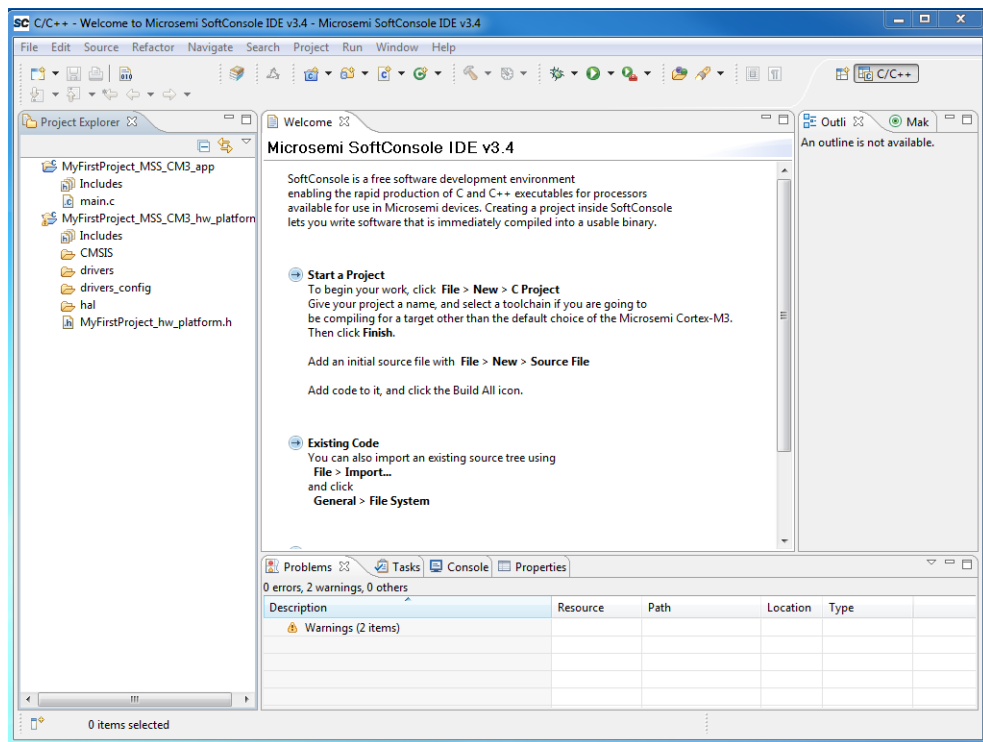


Рис. 24.

Откроем файл **main.c** и проверим его содержимое. Теперь создадим исполняемый файл, выполнив команду **Project>Clean** основного меню.

Порт UART модуля HC-06 может быть настроен на различные скорости передачи данных в диапазоне от 110 до 115200 бит/с. Убедитесь, что блок интерфейса UART микроконтроллерной подсистемы настроен на ту же скорость, что и UART модуля HC-06 (рис. 25).

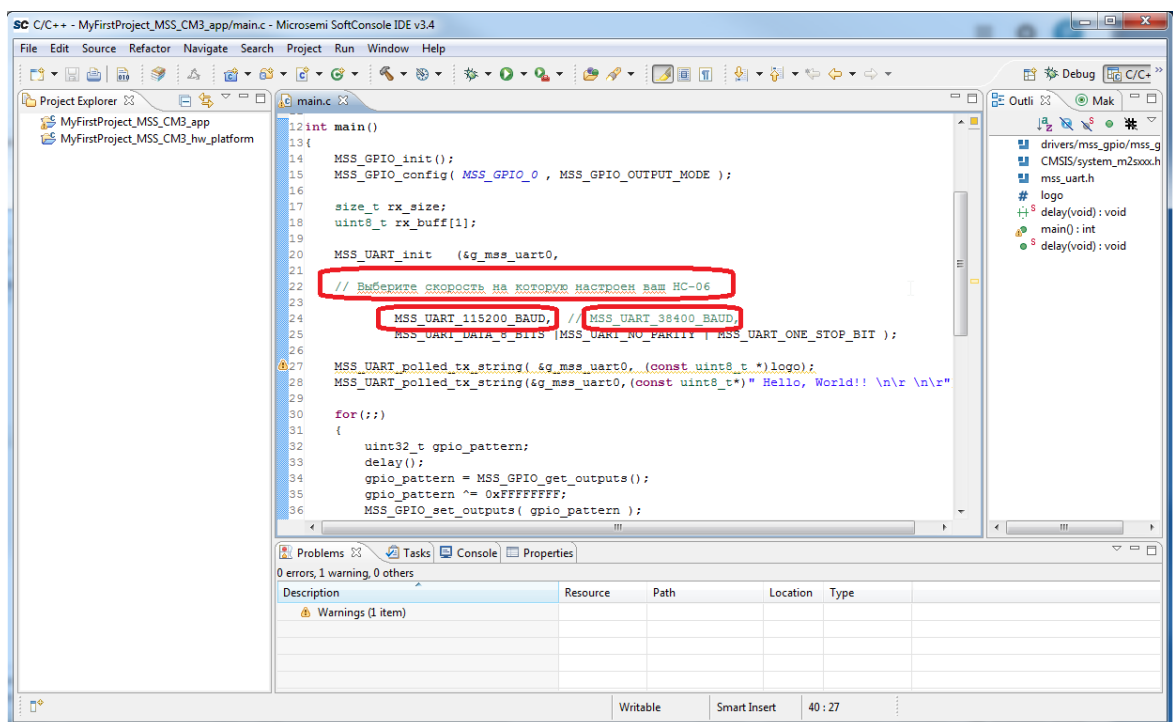


Рис. 25.

Пошаговая отладка кода встраиваемого программного обеспечения

Для выполнения отладки необходимо подключить программатор FlashPro4 к отладочной плате и персональному компьютеру, выполнить необходимые соединения для подключения SmartFusion2 с персональным компьютером по интерфейсу MMUART_0.

В состав отладочного набора входит модуль приемопередатчика Bluetooth – UART **HC-06** и USB Bluetooth донгл.

Подключаем HC-06 к отладочному набору как показано на рис. 26. USB-донгл включаем в USB разъем персонального компьютера.

После этого необходимо запустить программу терминал для отображения на экране ПК посылаемых SmartFusion2 сообщений.

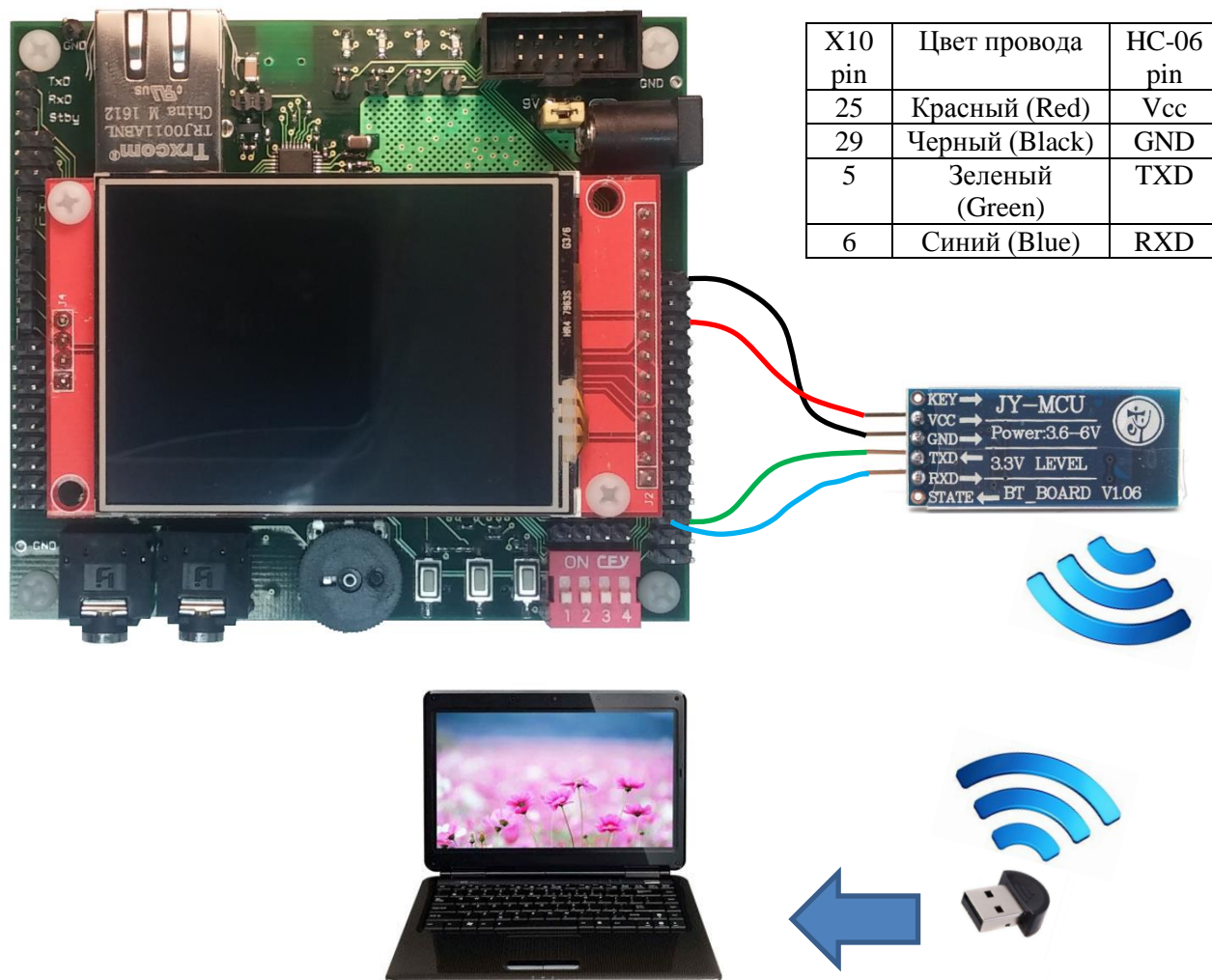




Рис. 26.

Приступим к пошаговой отладке программы, для этого выполним команду основного меню **Softconsole Run>Debug Configurations**. В появившемся окне дважды щелкаем левой кнопкой мыши на строке «**Microsemi Cortex-M3 Target**», появляется пункт «**LedUART_MSS_CM3_app**». Щелкаем на нем, и нажимаем кнопку **Debug** (рис. 26). В результате описанных действий запускается пошаговая отладка разработанного нами приложения. При нажатии кнопки Step Over  или клавиши F6 клавиатуры выполняется очередная строка кода программы (рис. 27). Для выполнения кода в непрерывном режиме нажмем кнопку Resume  (клавиша F8 клавиатуры) и пронаблюдаем отправку в окно терминала символов вводимых нами с клавиатуры (рис. 28).

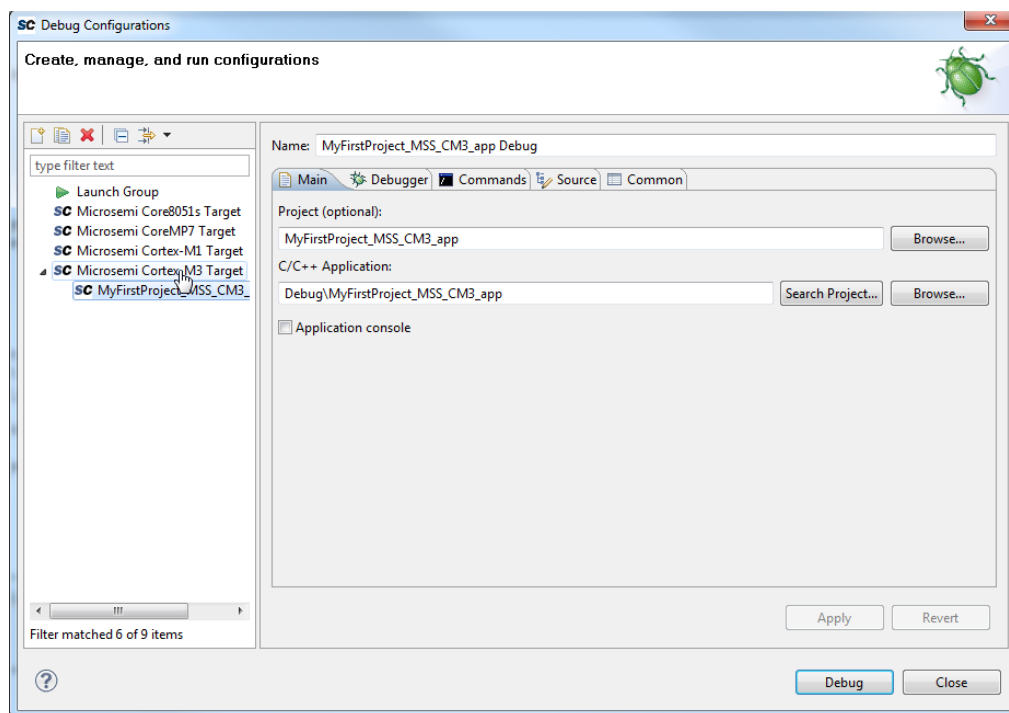


Рис. 27.

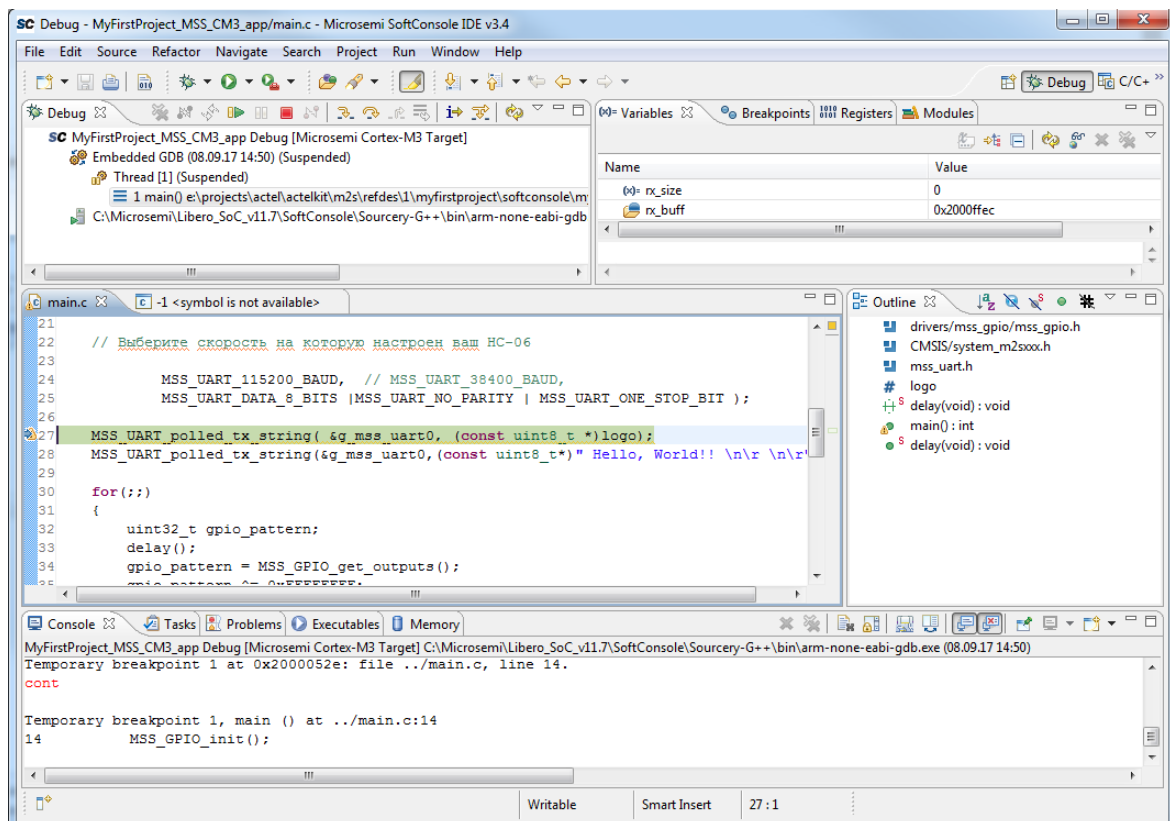


Рис. 28.

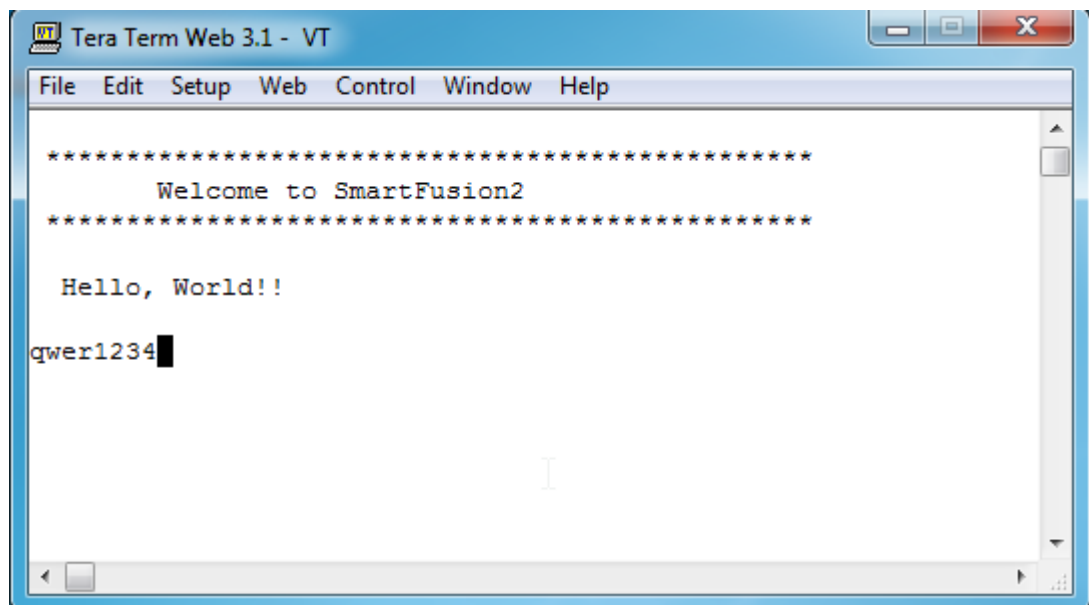


Рис. 29.

В ходе описанных выше действий код встраиваемого приложения процессора ARM Cortex-M3 был записан в память eSRAM процессора, следовательно, при выключении питания отладочного набора будет утрачен. Для того чтобы встроенное программное обеспечение выполнялось при следующем включении питания его необходимо загрузить в ПЗУ, в качестве которого в Microsemi SmartFusion2 выступает память eNVM. Процесс создания исполняемого файла ВПО и загрузки его в eNVM [рассмотрен](#) в следующем примере [проекта](#), доступном на сайте ООО «ПСР Актел».

Вопросы по материалу, изложенному в данном руководстве, можно задать сотрудникам службы технической поддержки компании ООО «ПСР Актел» по телефону **+7 (812) 740-60-09**, или по электронной почте **support@actel.ru**.