Содержание

Лабораторная работа №1. Создание проекта в программе Proteus. Сборка электрической схемы для моделирования.

Тема: Изучение базовых принципов создания проекта в программе Proteus.
Цель: Овладеть навыками работы с программой Proteus.
Описание: В данной лабораторной работе студент знакомится с элементами интерфейса программs Proteus. Также будут приобретены практические навыки работы с программой Proteus.

Выполнение работы.

1. Создание проекта в среде Proteus (исп. версия программы 8.4 SP0 – 8.9 SP0).

Запустить программу Proteus 8 Professional. Будет открыто окно пуска программы (рис 1.1).

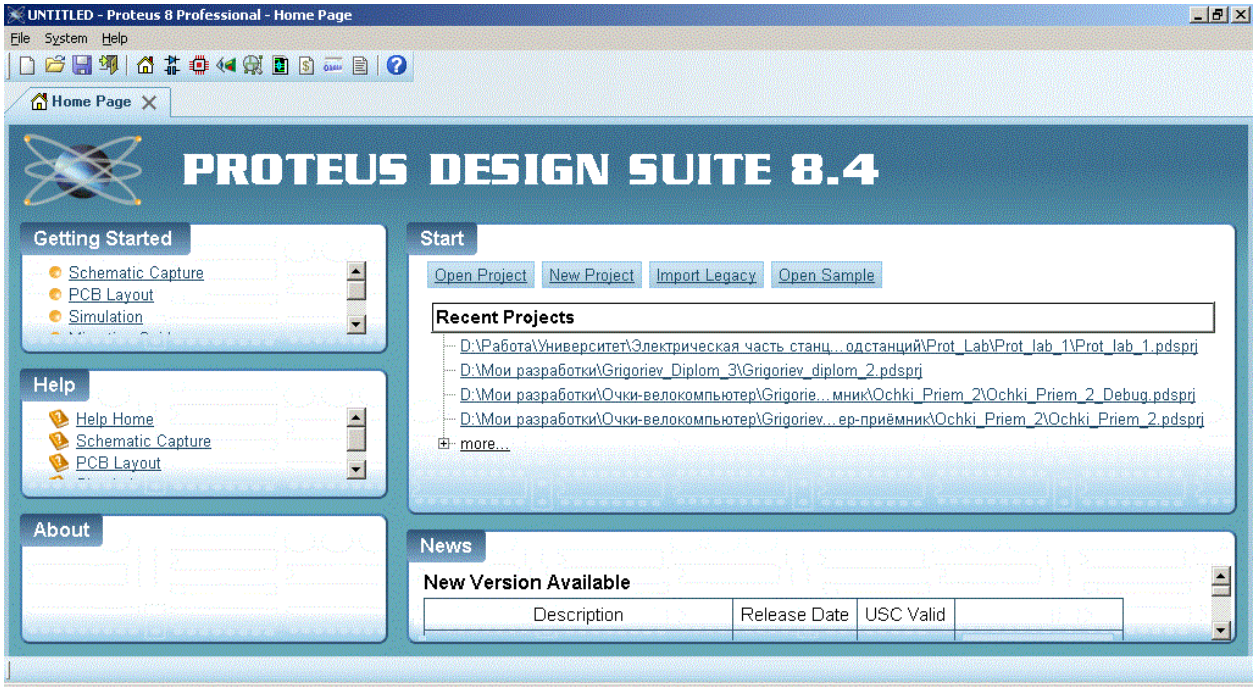

Рис. 1.1 – Окно запуска программы.

Далее во вкладке «File» выбрать пункт «New Project». Запустится окно «New Project Wizard: Start» (рис 1.2).

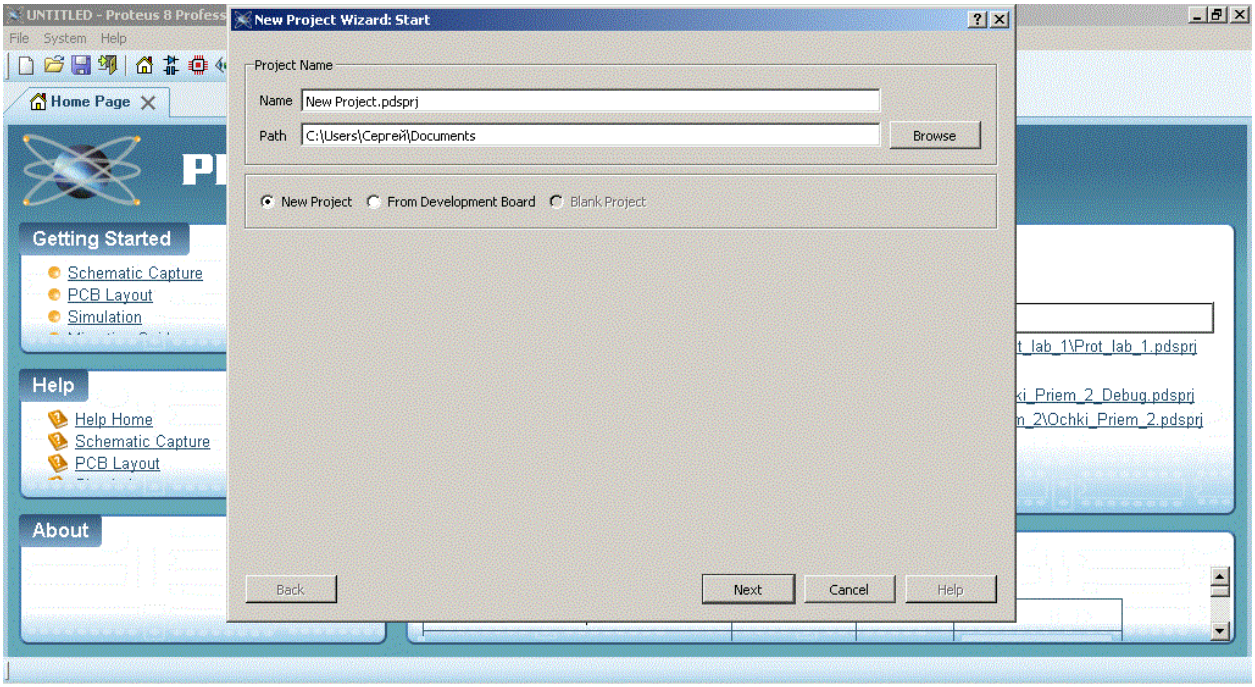

Рис 1.2 – Окно выбора названия нового проекта и пути к нему.

В открытом окне, в строке «Name», ввести имя своего проекта, например, «Ivanov_ESIS-16.pdsprj». Ниже, в строке «Path» необходимо указать путь к своему проекту. Для этого нажать справа, в этой же строке, кнопку «Browse» и выбрать папку на компьютере для сохранения там своего нового проекта (рис.1.3).

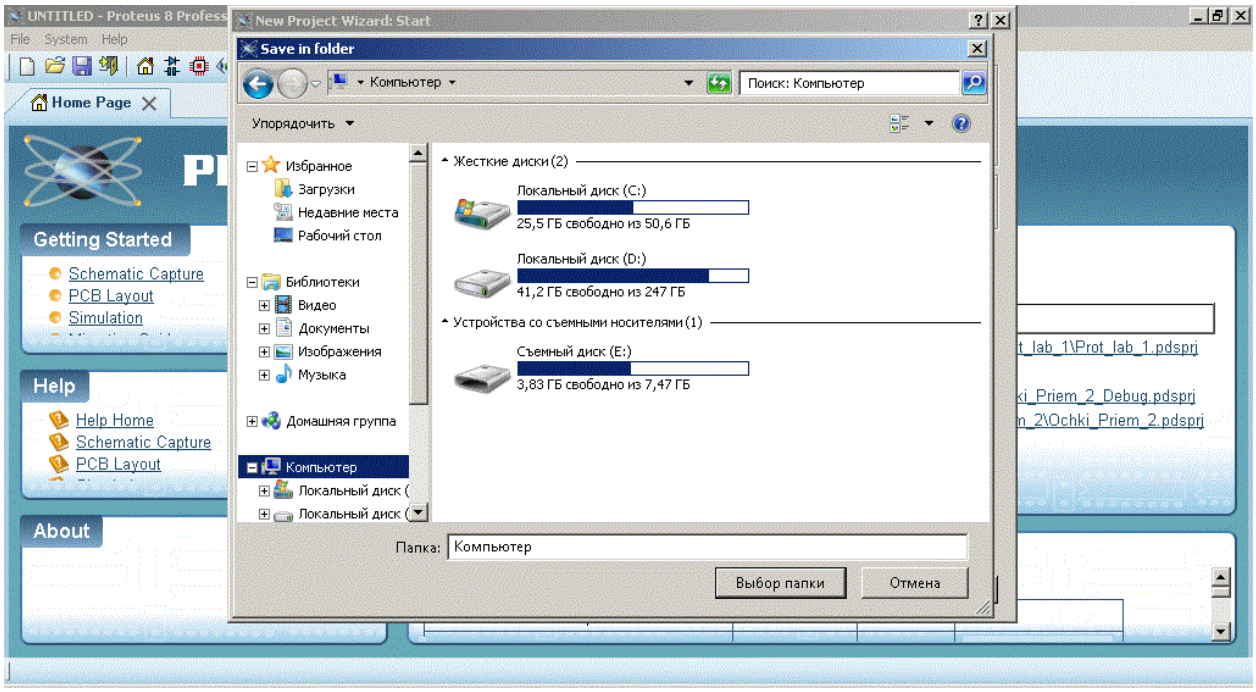

Рис. 1.3 – Окно выбора папки сохранения нового проекта.

После выбора необходимой папки, нажать кнопку «Выбор папки», после чего идет возврат к окну создания нового проекта (рис. 1.2). В нем, необходимо нажать кнопку «Next». Будет совершен переход в новое окно «New Project Wizard: Schematic Design» (рис 1.4).

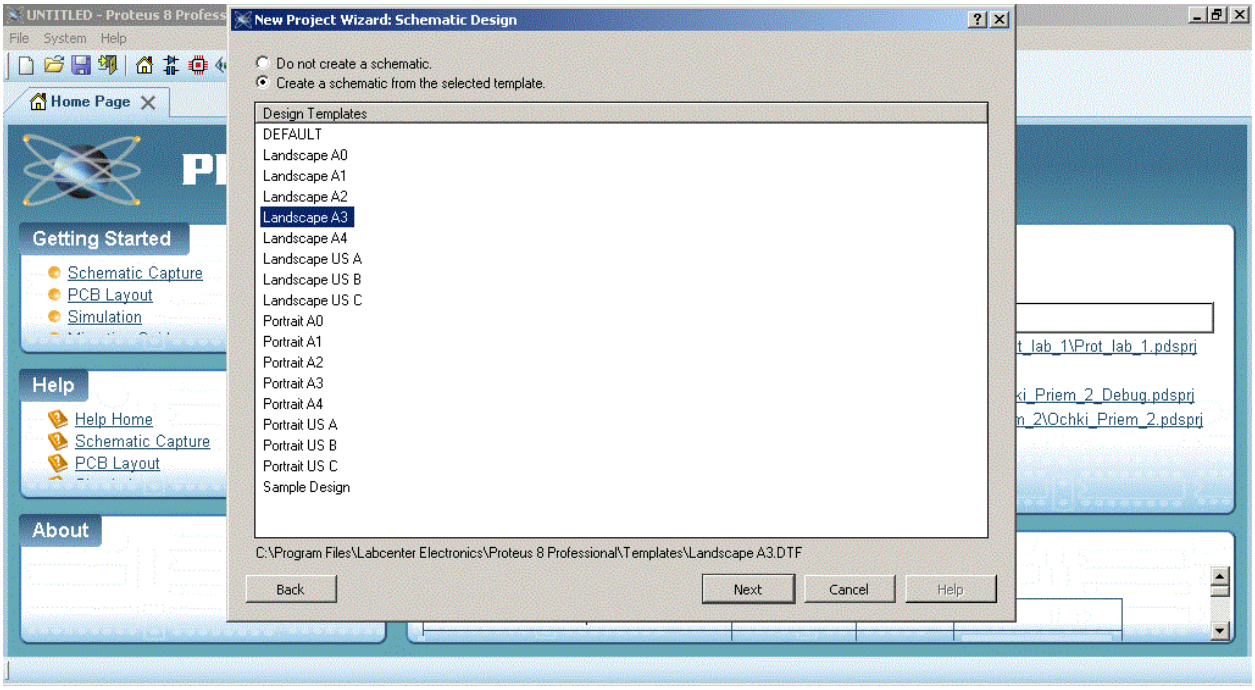

Рис. 1.4 – Окно выбора размера рабочего поля.

В нем, следует выбрать размер рабочего поля нового проекта, например, «Landscape A3», что соответствует размеру листа формата А3. Выбрав данный (или меньше) размер поля, необходимо нажать кнопку «Next». Будет совершен переход к окну «New Project Wizard: PCB layout» (рис. 1.5).

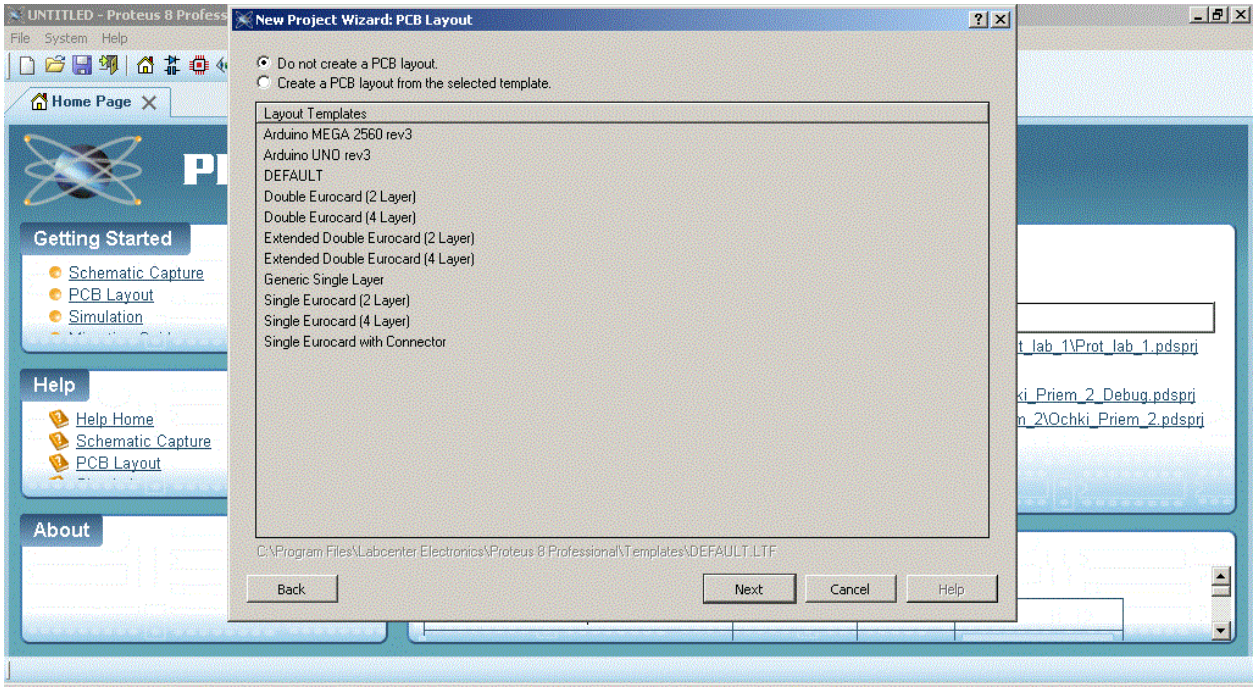

Рис. 1.5 – Окно выбора макета печатной платы.

В данном окне, ставим галочку напротив строки «Do not create a PCB layout» (не создавать макет печатной платы) и нажимаем кнопку «Next». Переходим к окну «New Project Wizard: Firmware» (рис 1.6).

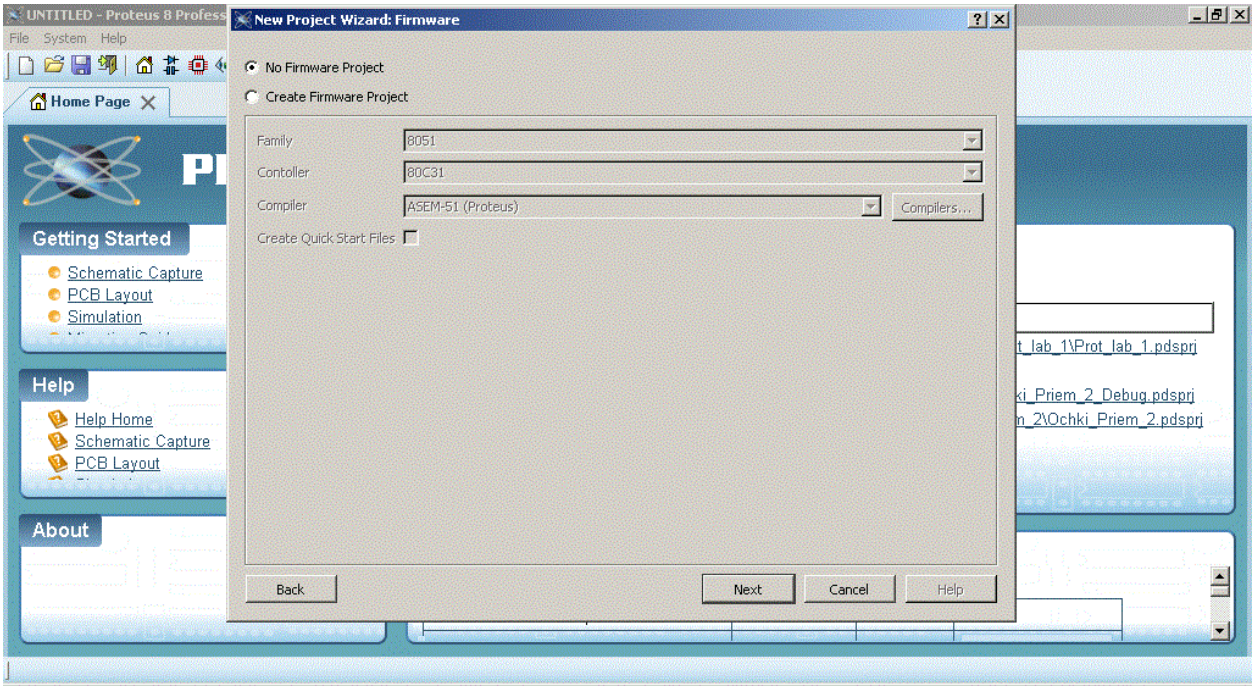

Рис. 1.6 – Окно выбора прошивки.

В данном окне, ставим галочку напротив строки «No Firmware Project» (не создавать проект прошивки) и нажимаем кнопку «Next». Переходим к окну «New Project Wizard: Summary» (рис. 1.7).

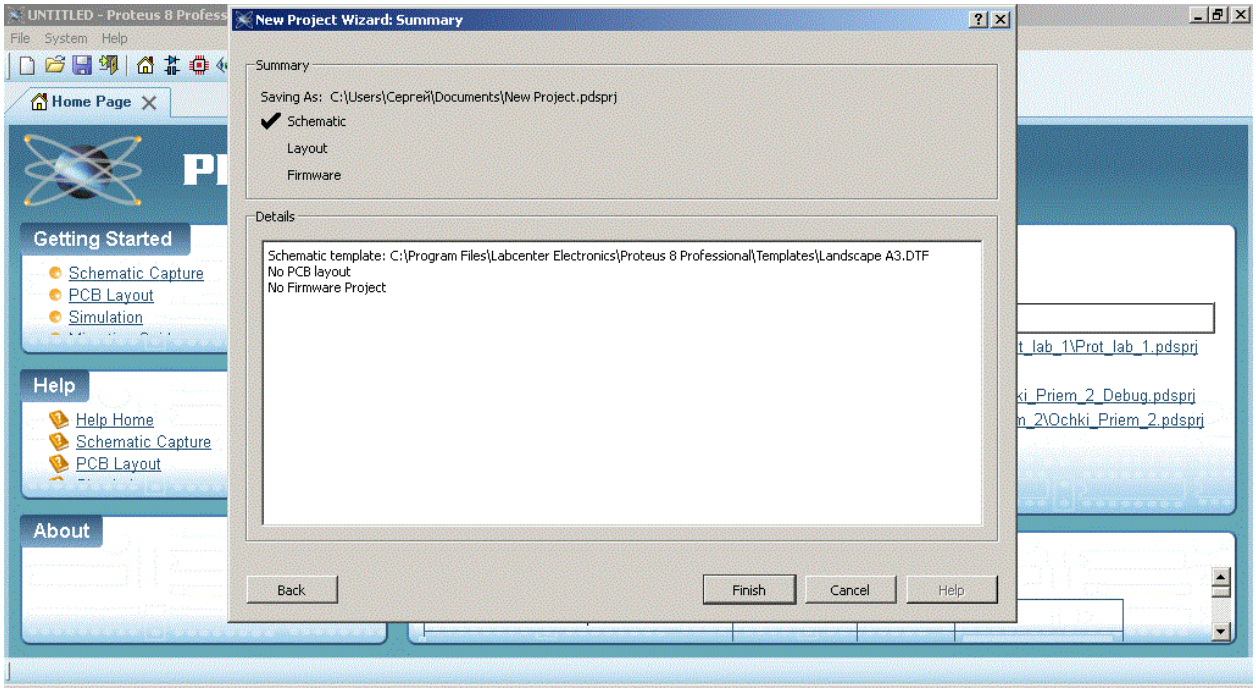

Рис. 1.7 – Окно завершения создания проекта с выводом всех выбранных параметров нового проекта.

Далее, нажимаем кнопку «Finish». Создаётся новый проект. Программа переходит в окно рабочего поля (рис. 1.8).

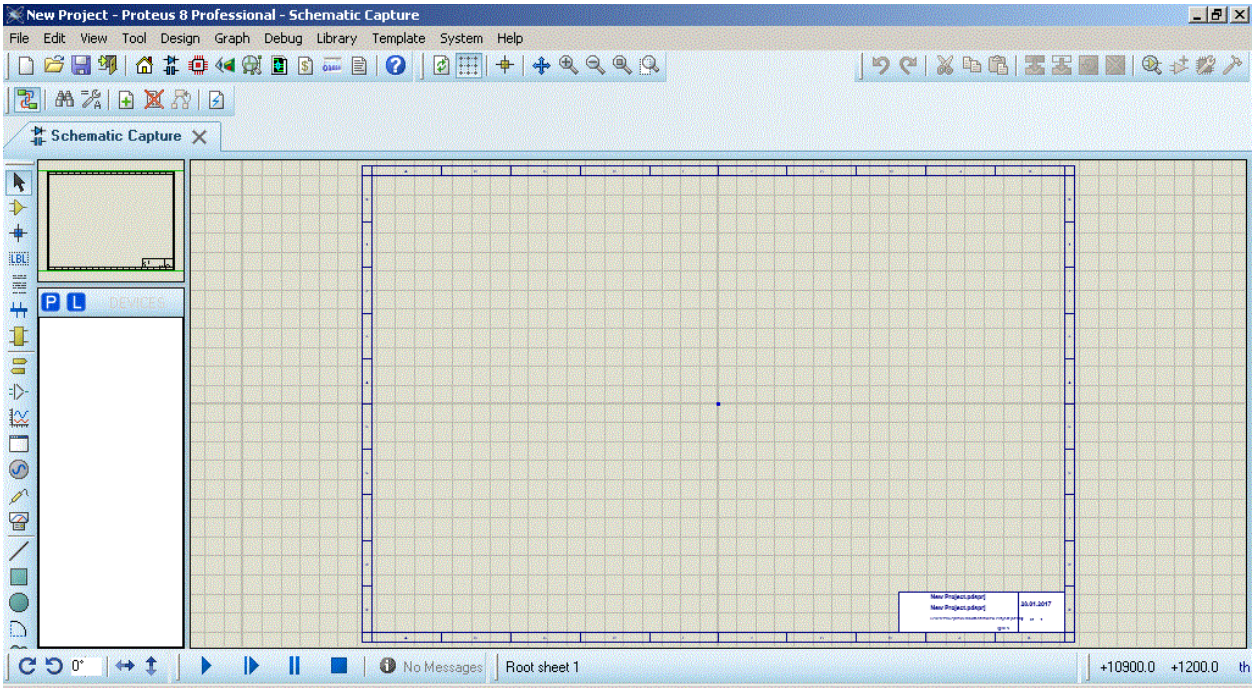

Рис. 1.8 – Окно рабочего поля программы.

2. Порядок работы с проектом в среде Proteus.

Для того, чтобы добавить необходимые элементы для симуляции схемы, необходимо знать, где они находятся в данной программе. Слева, чуть выше центра, под миниатюрой рабочего поля проекта, расположена кнопка «P». После ее нажатия, будет произведен переход в окно «Pick Devices» (рис. 1.9).

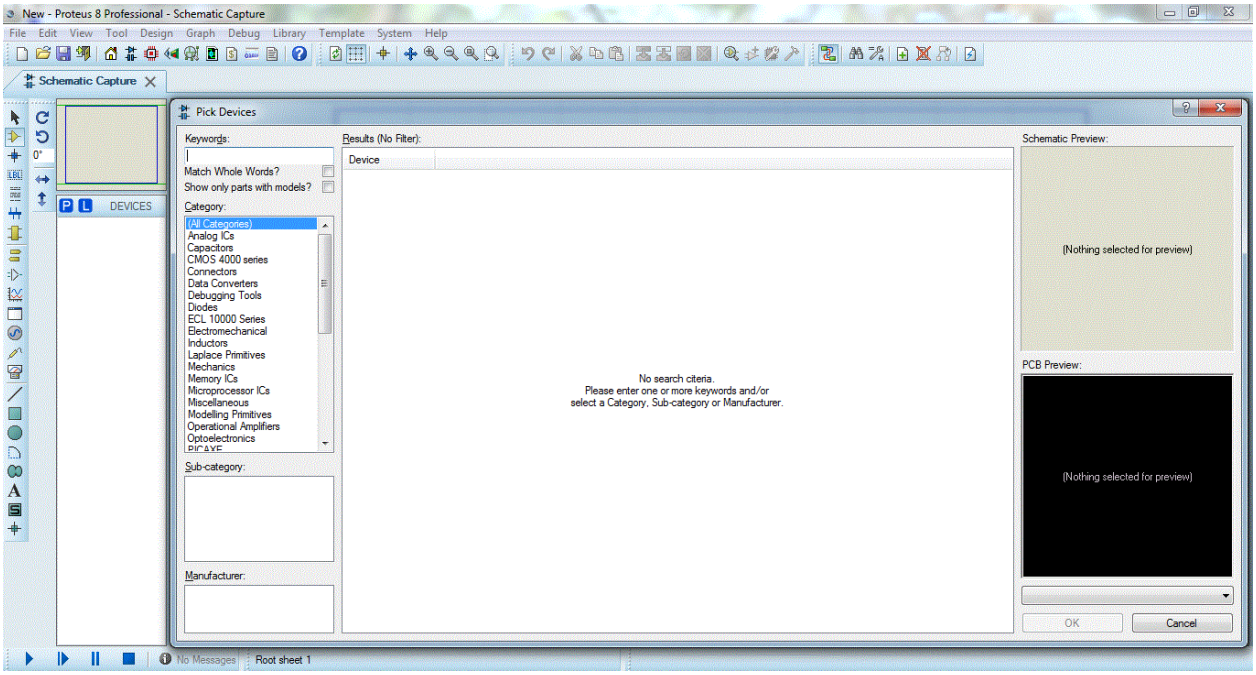

Рис. 1.9 – Окно добавления устройств для симуляции в рабочем поле.

В данном окне, в строке «Keywords», необходимо ввести интересующие параметры элемента симуляции, например, микроконтроллер «Atmega32». В поле «Result (No Filter)», будет предложен список наиболее подходящих, под данное описание, устройств (рис. 1.10).

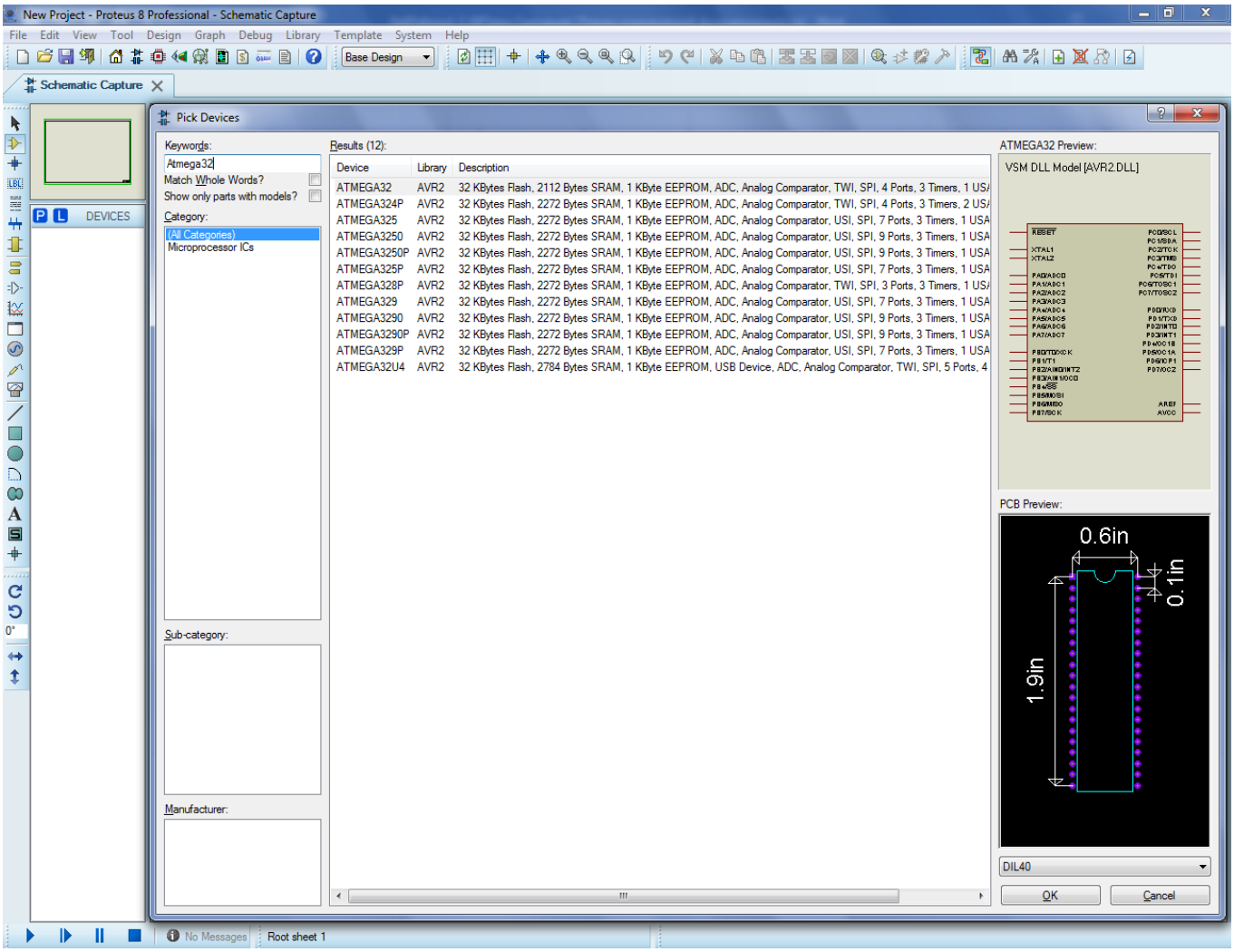

Рис. 1.10 – Окно выбора устройств симуляции. Микроконтроллер.

Для наших целей, вполне достаточно микроконтроллера Atmega32. Для выбора данного микроконтроллера, необходимо навести на него курсор в поле «Result (No Filter)» и сделать двойной щелчок левой кнопкой мыши (ЛКМ). Он появится в списке устройств, слева, в главном окне программы, в поле «DEVICES».

Также, необходимо добавить такие элементы как, «строчный символьный дисплей» и «кнопка». Для этого в поле «Keywords» необходимо ввести соответствующие запросы и добавить их, поочередно, в список устройств «DEVICES»: «Button» («Active», «SPST Push Button») и «LCD 16x2» («LM016L», «16x2 Alphanumeric LCD»). После чего, нажать кнопку «ОК», и программа перейдет обратно к главному окну.

Для добавления элемента на рабочее поле программы, необходимо выбрать желаемый элемент в поле «Devices», нажать ЛКМ в произвольном месте рабочего поля. Сразу после этого за курсором потянется фиолетовая фигура данного элемента, для выбора её места добавления на рабочем поле (рис. 1.11).

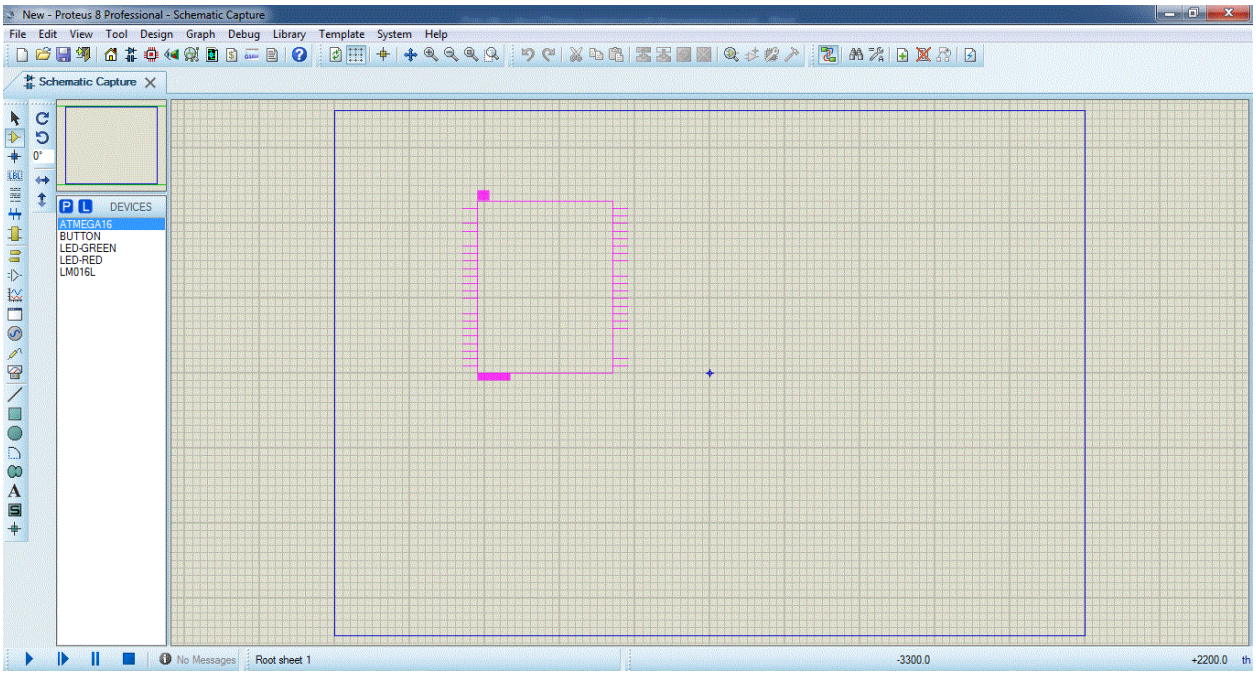

Рис. 1.11 – Выбор места установки элемента. Микроконтроллер.

После выбора нужного места, чтобы оставить там элемент, необходимо нажать ЛКМ. Элемент добавится на рабочее поле программы (рис. 1.12).

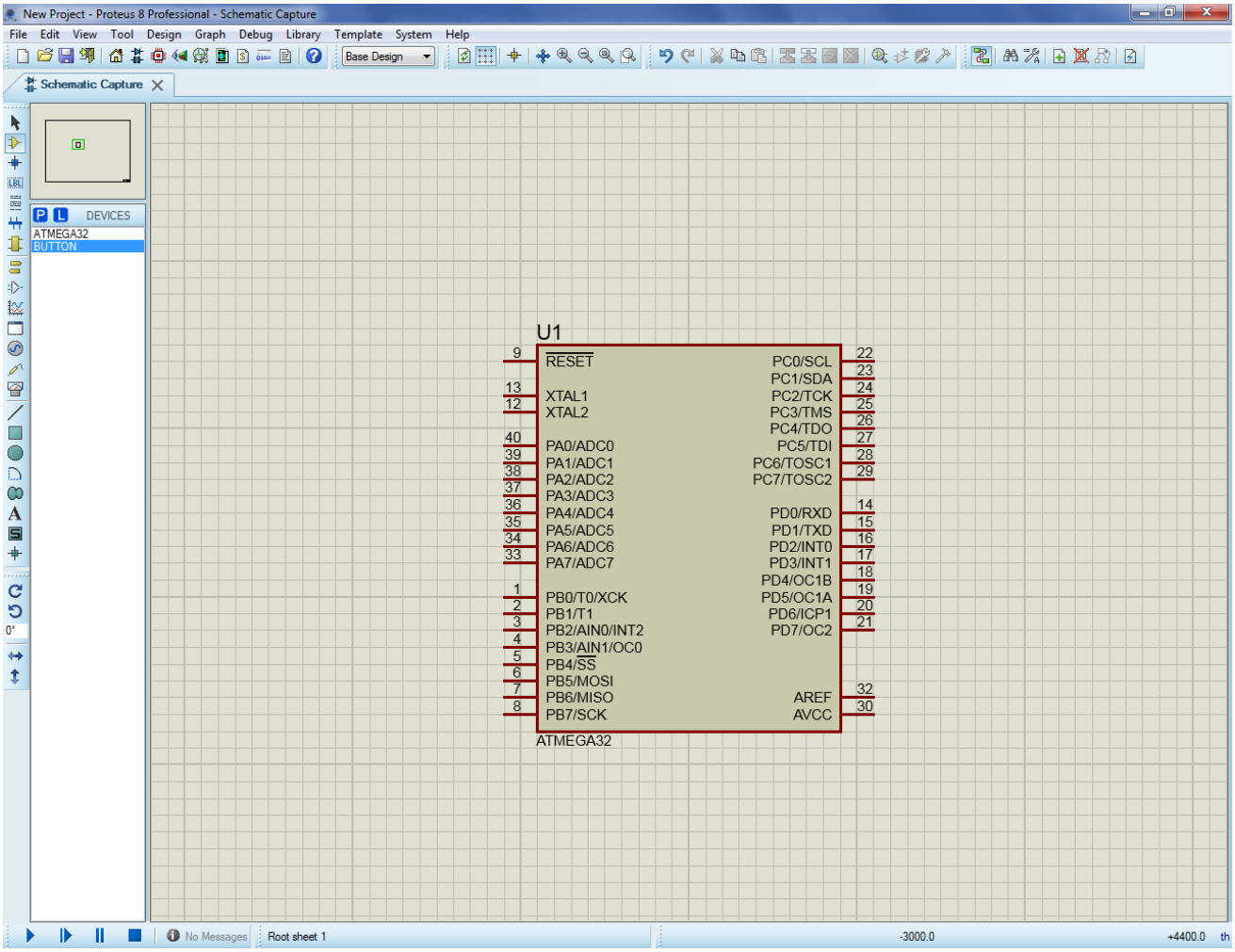

Рис. 1.12 – Установленный элемент на рабочем поле. Микроконтроллер.

После установки микроконтроллера, необходимо его настроить. Для этого необходимо навести на него курсор и два раза кликнуть по нему ЛКМ. Высветится окно свойств данного элемента (рис. 1.13).

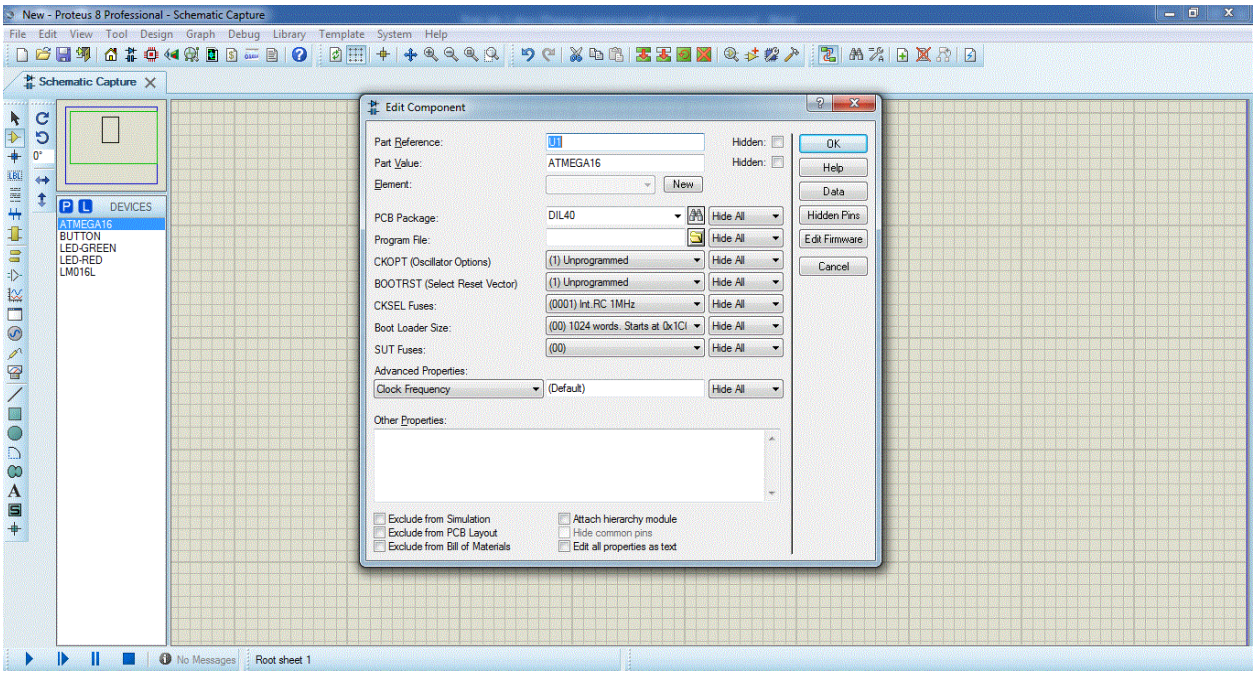

Рис. 1.13 – Окно свойств элемента. Микроконтроллер.

Далее, в поле «Program File» выбираем файл прошивки для нашего микроконтроллера в соответствующей папке. Для этого, необходимо нажать кнопку с иконкой раскрытой папки, после чего программа откроет проводник Windows, для выбора необходимого файла (рис. 1.14).

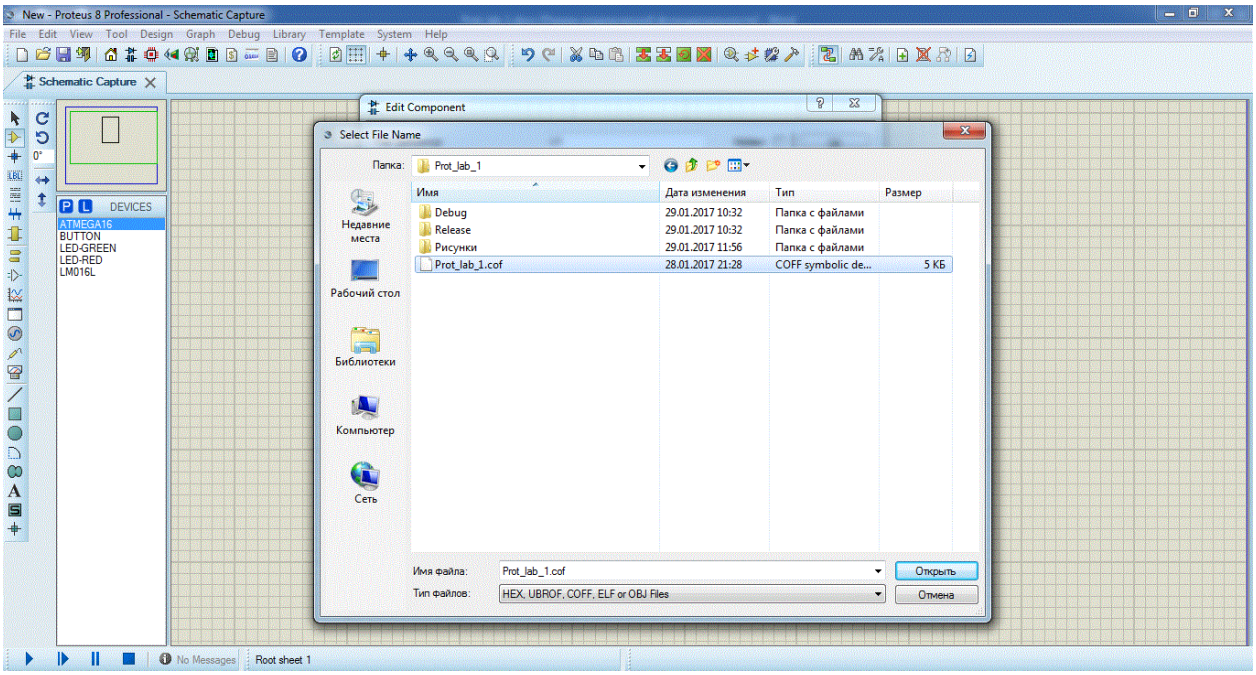

Рис. 1.14 – Выбор файла прошивки для микроконтроллера при помощи стандартного проводника Windows.

После выбора необходимого файла, нажать кнопку «Открыть». Программа вернется в окно свойств элемента (микроконтроллера) (рис. 2.6).

В поле «CKSEL Fuses» выбрать «(0100) Int. RC 8MHz» (частота работы внутреннего кварца микроконтроллера 8 МГц). Больше, для наших целей, ничего настраивать не нужно, нажимаем кнопку «ОК». Программа вернется к главному окну и рабочему полю.

Приблизительно справа снизу от микроконтроллера необходимо, аналогично установке микроконтроллера, установить, поочередно, 6 элементов «Button» («Кнопка»), как показано на рис. 1.15.

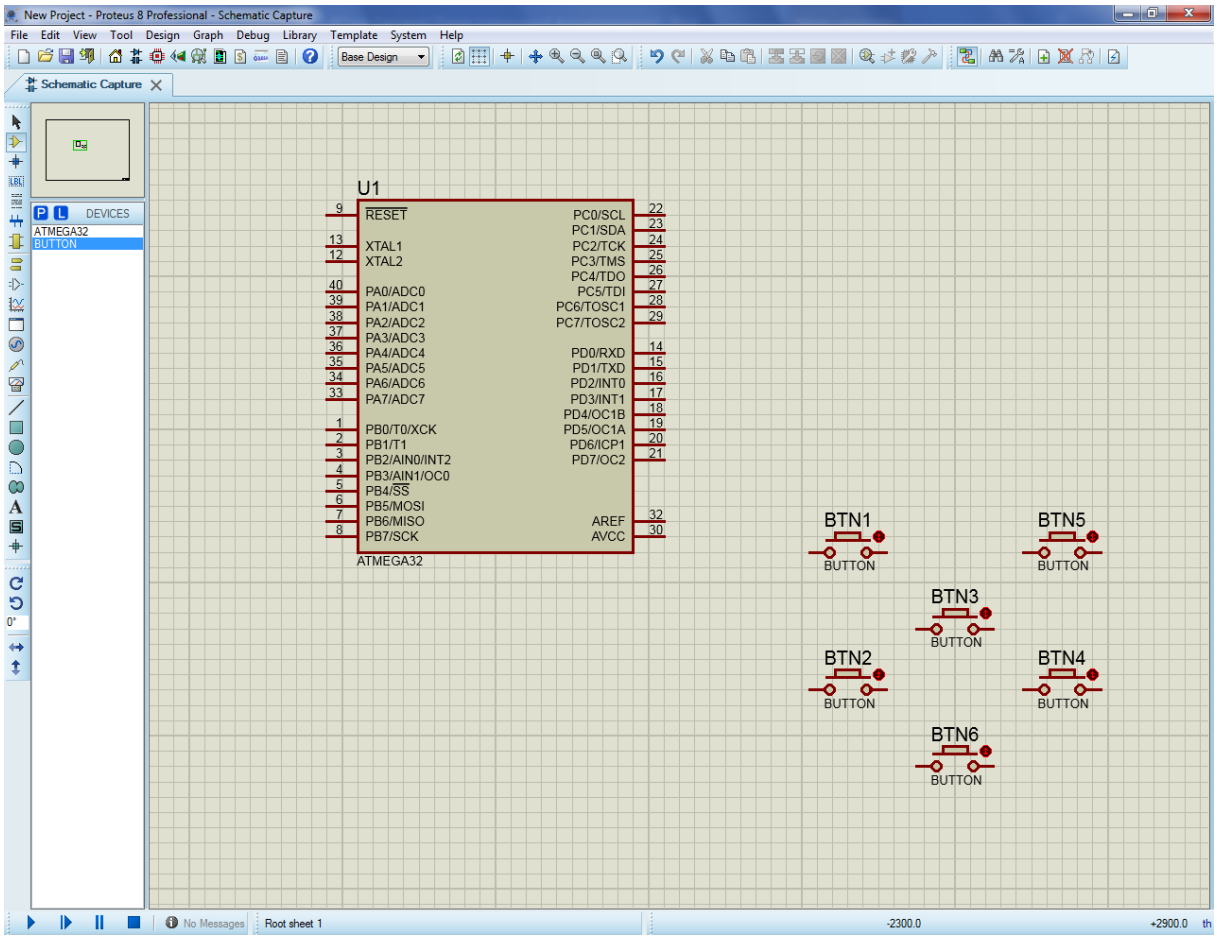

Рис. 1.15 – Установленные элементы «Button», без соединения.

Для соединения любого элемента на рабочем поле, необходимо курсор подвести к ножке или пину элемента, на ее (его) конце должна засветиться красный квадрат. Далее, необходимо нажать, там, ЛКМ, и, за курсором, потянется зеленая линия («электрическое» соединение элементов). Ее необходимо довести до необходимого пина микроконтроллера, на его конце должен засветиться красный квадрат, и, после этого, нажать ЛКМ. Кнопка соединена с выводом микроконтроллера (рис. 1.16).

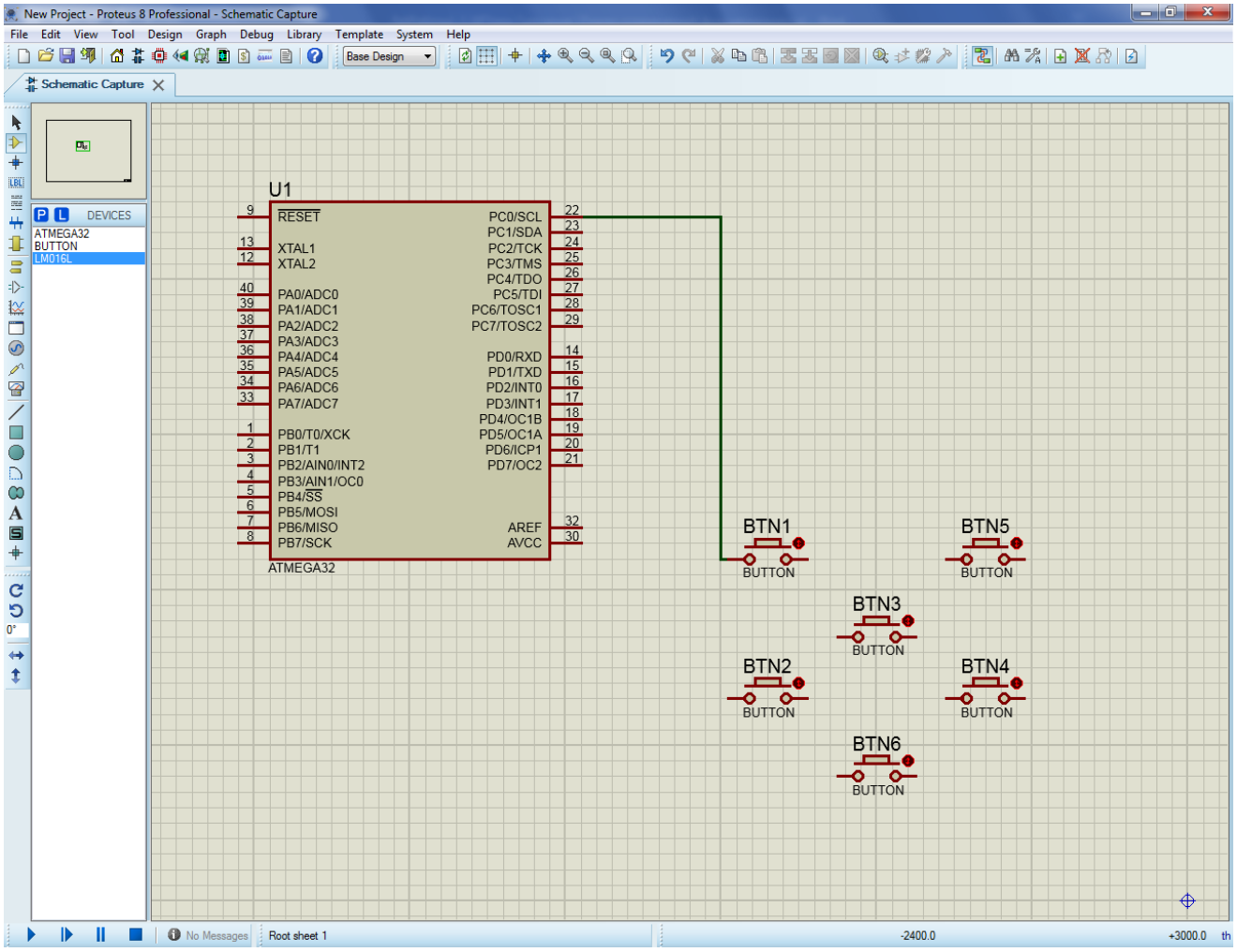

Рис. 1.16 – Соединенная кнопка с микроконтроллером.

Однако, соединять кнопки таким образом с микроконтроллером нерационально, ввиду того, что линии могут пересекаться и может возникнуть, в итоге, путаница.

Рекомендуемый метод соединения с помощью инструмента «BUS». Для того, чтобы им воспользоваться, необходимо на панели инструментов слева найти иконку с 2 разнонаправленными жёлтыми стрелками. И выбрать в появившемся меню «BUS» (рис. 1.17).

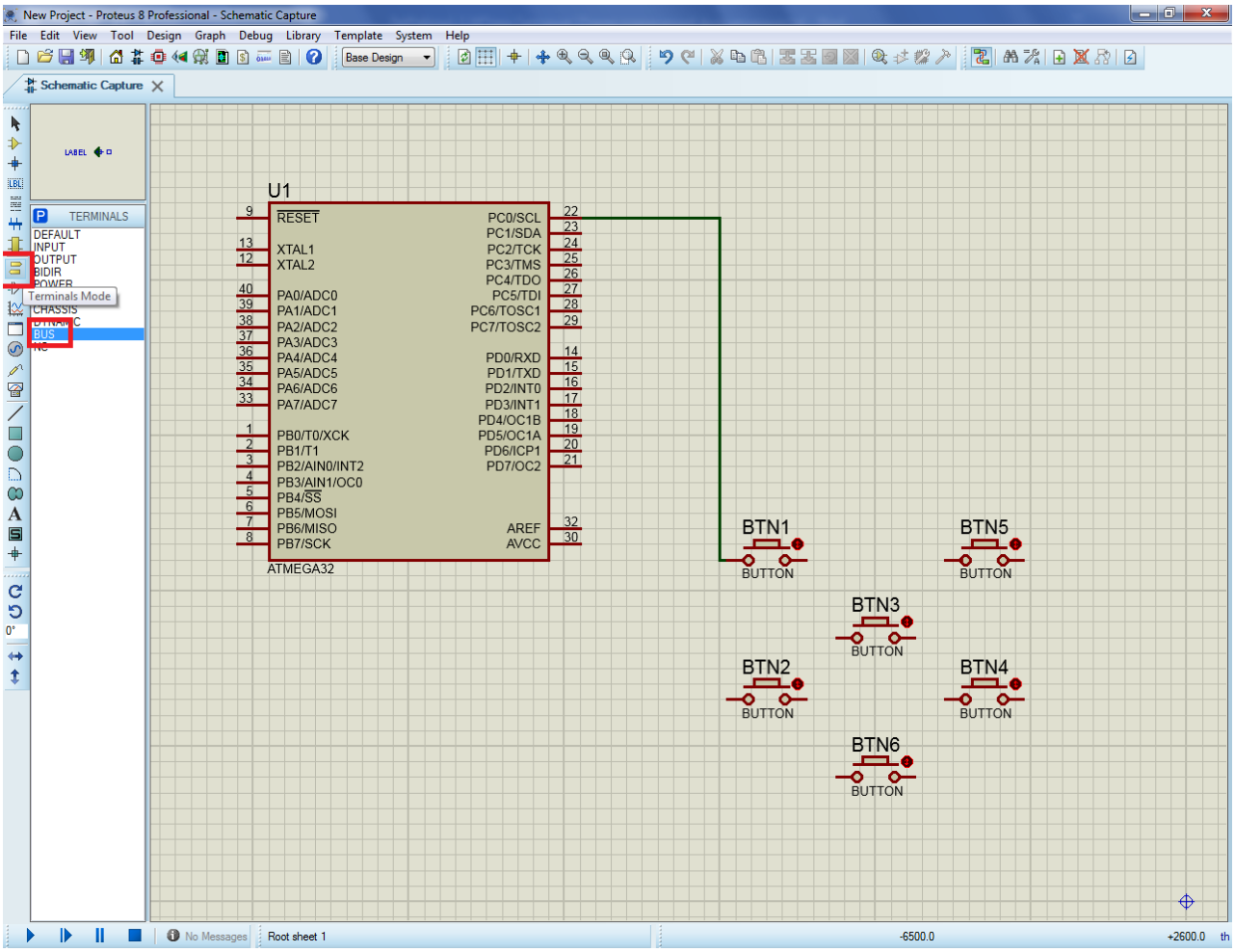

Рис. 1.17 – Выбор инструмента «BUS».

Далее, необходимо, поместить данный инструмент на рабочее поле. Делается это следующим образом:

- кликнуть ЛКМ в произвольном месте рабочего поля;
- переместить курсор в необходимое место рабочего поля (в нашем случае, это чуть выше нулевого пина порта С, «PC0»);
- нажать ЛКМ;
- инструмент предложит графически протянуть его на необходимое расстояние, что мы и сделаем напротив всего порта С, после чего дважды нажать ЛКМ (рис. 1.18);
- соединяем первые 6 пинов микроконтроллера с линией BUS (рис. 1.19);
- слева на панели инструментов выбираем иконку «LBL» (рис. 1.20);
- переводим курсор на нужную линию и нажимаем ЛКМ, после чего появится окно, в котором необходимо написать оригинальное имя для данной линии связи, в нашем случае, каждую линию будем называть соответственно названий кнопок, которые эти линии будут подключать (рис. 1.21);
- производим аналогичные действия уже для кнопок (рис. 1.22).

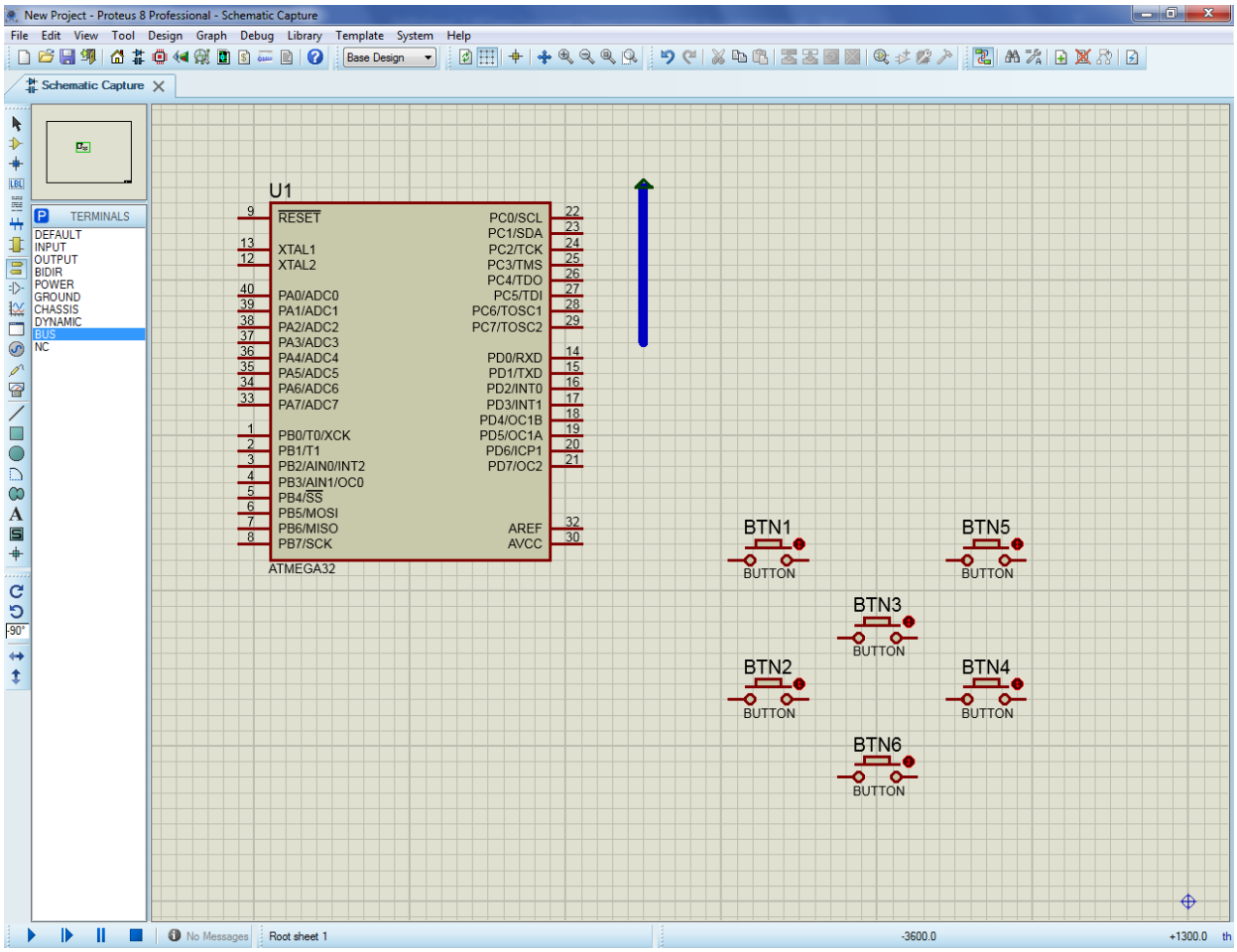

Рис. 1.18 – Линия BUS напротив порта C.

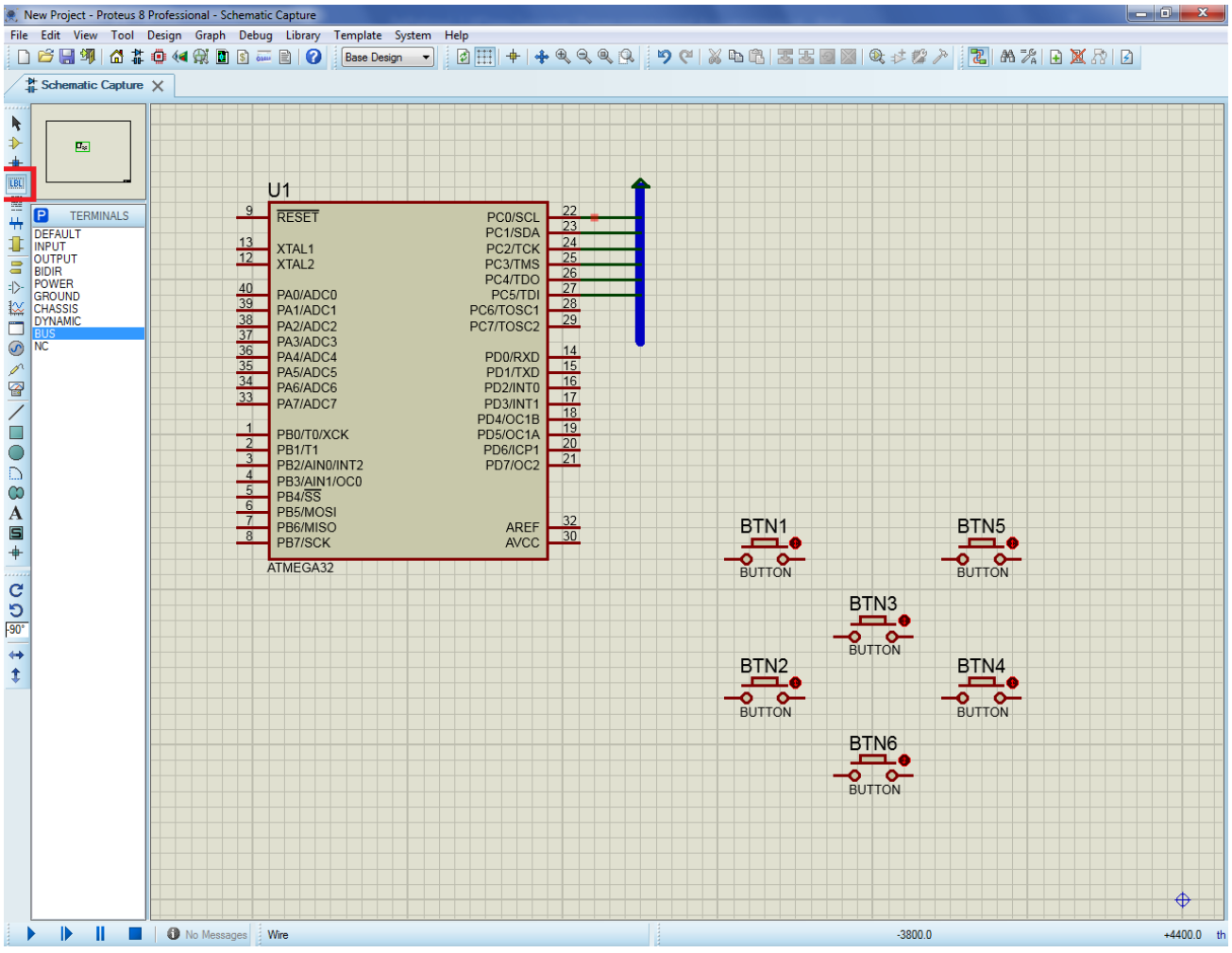

Рис. 1.19 – Проложенные линии связи к линии BUS.

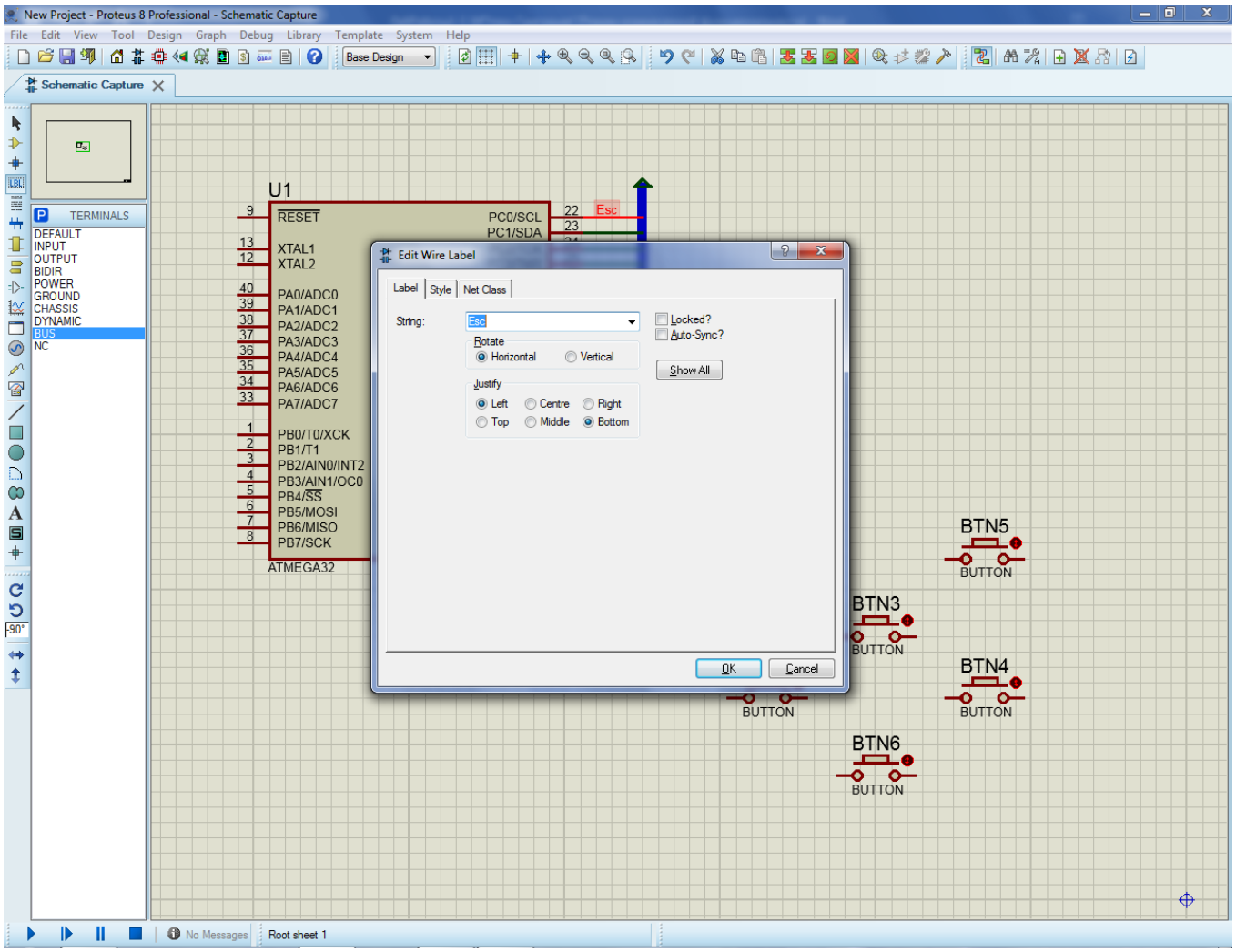

Рис. 1.20 – Окно ввода названия линии.

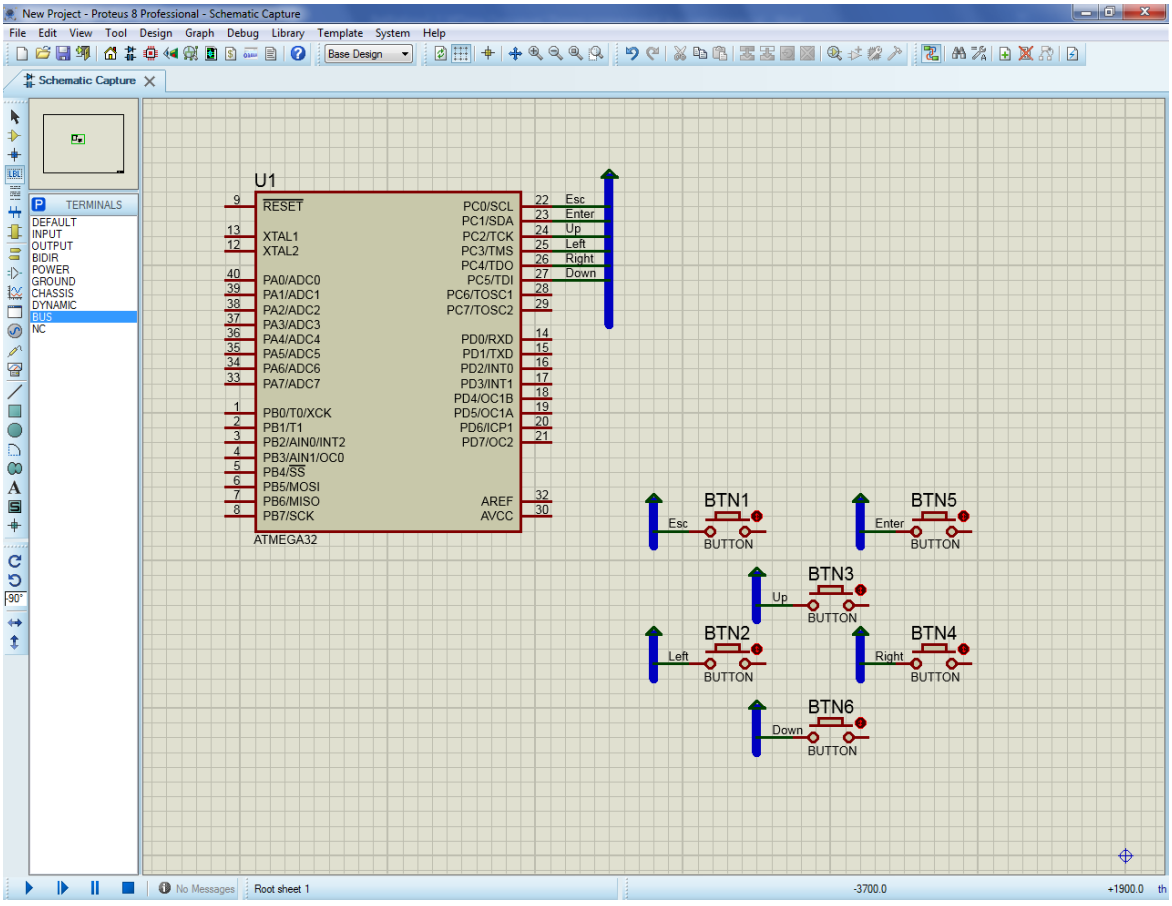

Рис. 1.21 – Готовое подключение кнопок к микроконтроллеру.

Противоположные стороны кнопок необходимо соединить с инструментом «GROUND» (земля), он находится там же, где и инструмент «BUS» (рис. 1.22).

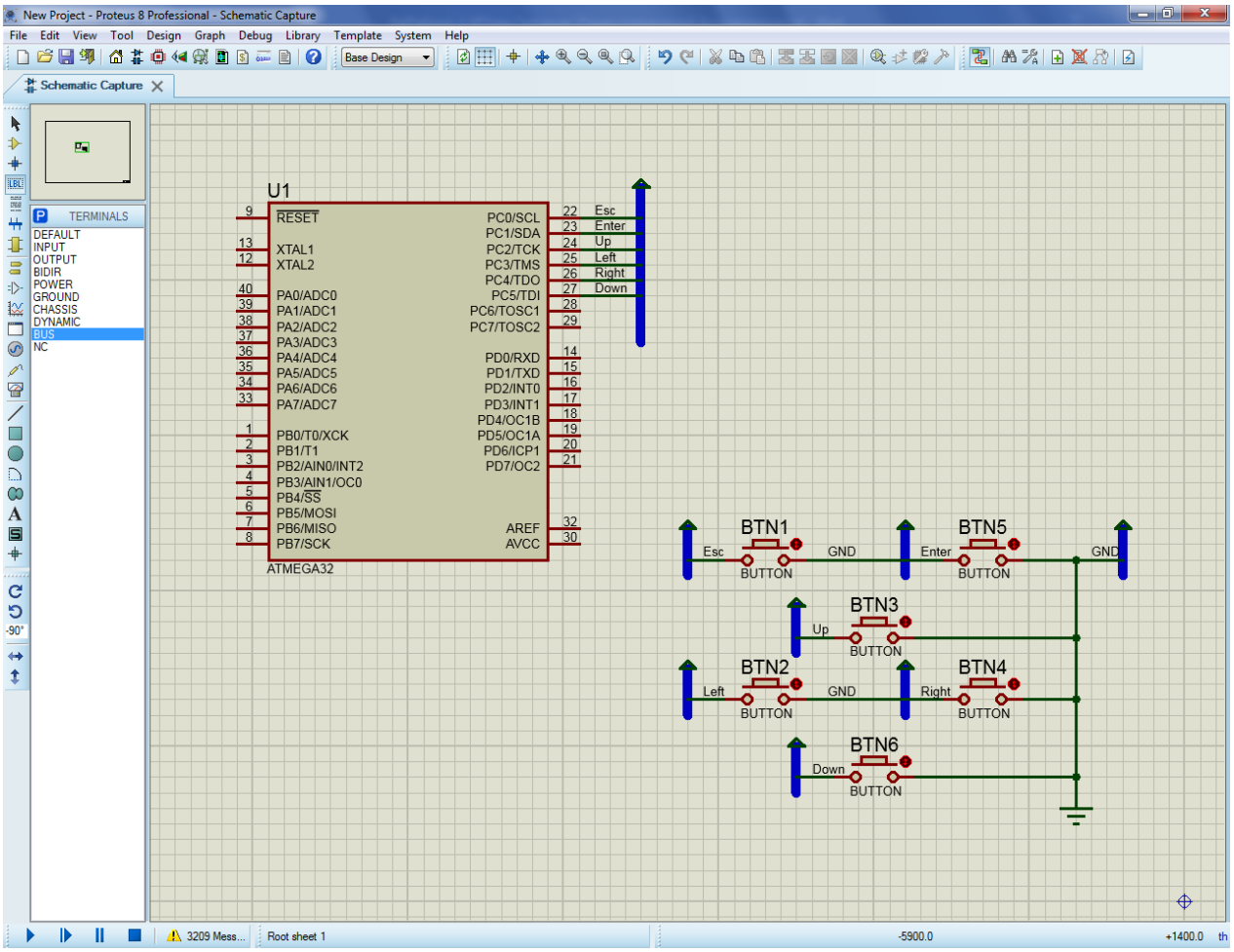

Рис. 1.22 – Кнопки подключены к инструменту GROUND.

Для соединения данных кнопок с инструментом GROUND, не обязательно соединять их непосредственно с ним. Достаточно присоединить их к линии связи одной подключенной кнопки. Для этого необходимо проделать аналогичные действия по рисованию линии связи от каждой из кнопок, только, довести ее, необходимо, до линии связи кнопки с инструментом GROUND, должен на линии загореться красный квадрат, и нажать ЛКМ (рис. 1.22).

Теперь мы можем подключить дисплей. Будем использовать символьный дисплей разрешением 16 строковых символов на 2 строки. Для этого в библиотеке элементов в поле «keywords» необходимо ввести «LM016L» (рис. 1.23).

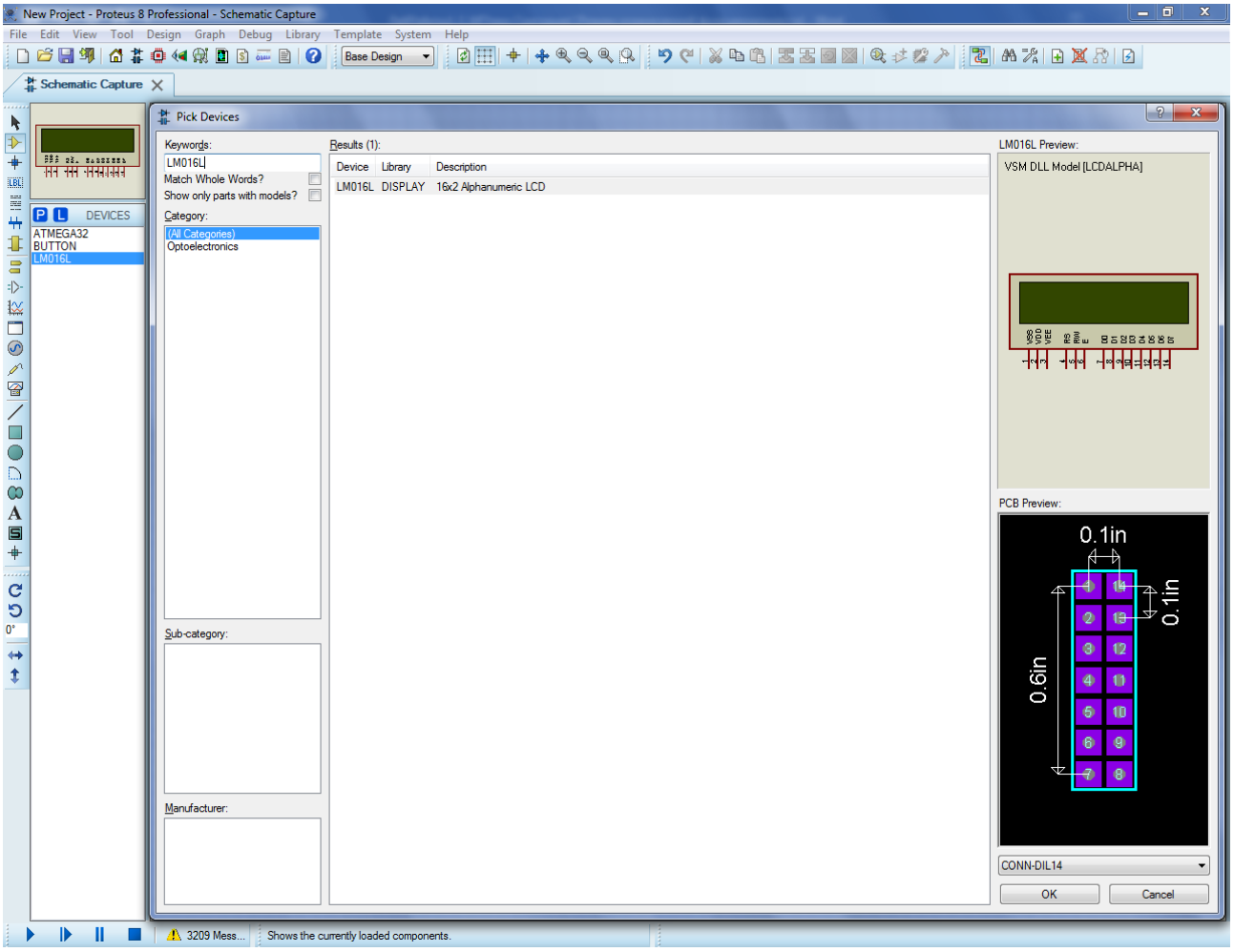

Рис. 1.23 – Поиск выбор дисплея в библиотеке элементов.

Помещаем дисплей на рабочее поле над клавиатурой, оставив место для создания линий связи при помощи инструмента BUS и подключаем аналогично кнопкам (рис. 1.24). Дисплей будем подключать по 4х битной шине данный. Для этого необходимо соединить соответствующие пины микроконтроллера (в нашем случае, это PD0-PD2, PD4-PD7) со следующими пинами дисплея (RS, RW, E, D4-D7) (рис. 1.24).

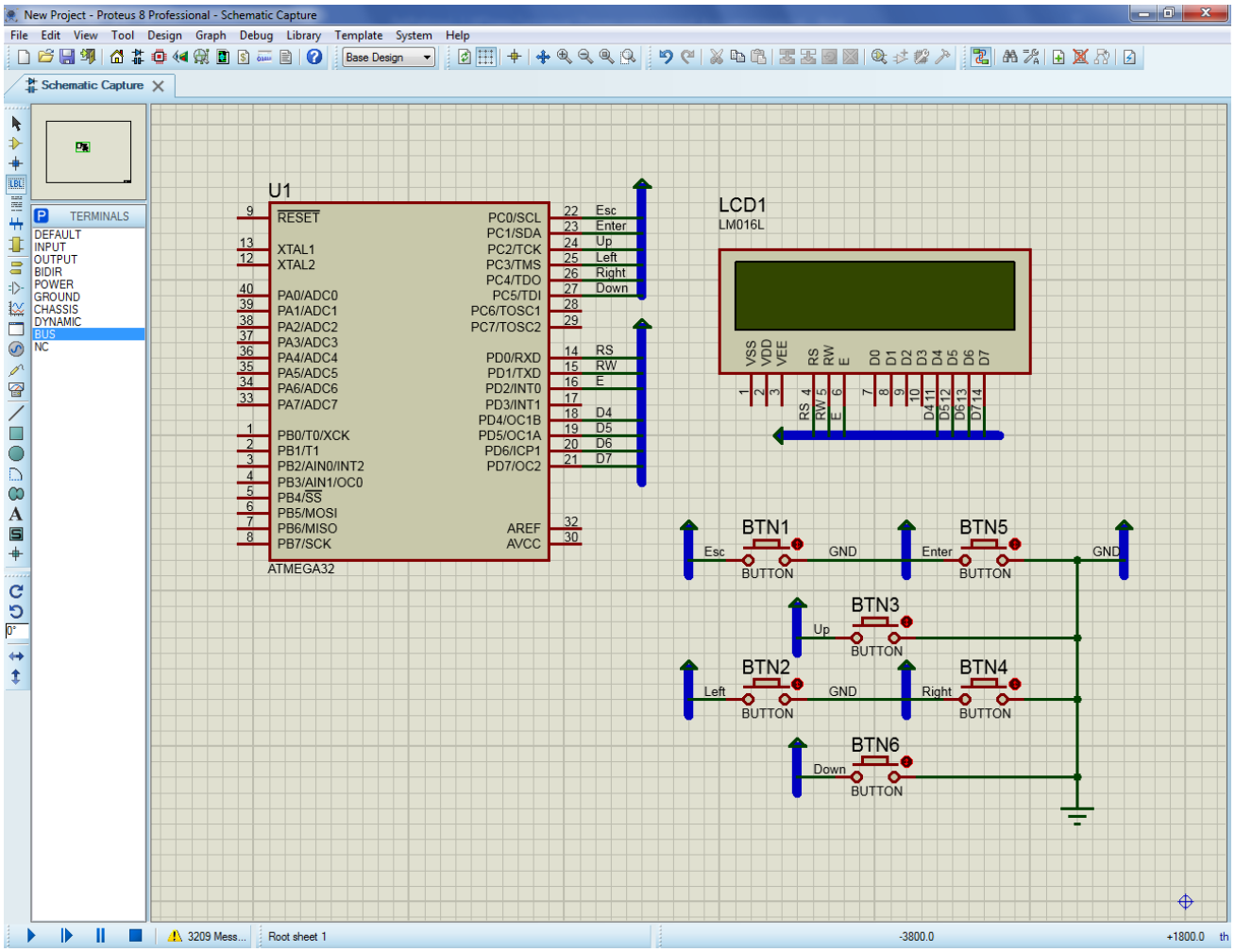

Рис. 1.24 – Подключенный дисплей к микроконтроллеру.

Т.к. в дальнейшем мы будем использовать аналогово-цифровой преобразователь АЦП, необходимо к пинам опорного напряжения микроконтроллера подключить источник питания «+5В». Его можно найти слева на панели инструментов. Нам нужна иконка изображающая генератор на электрической схеме. Далее, в поле «GENERATORS», необходимо выбрать элемент «DC» (источник постоянного напряжения). И поместить этот элемент справа сверху от необходимых пинов микроконтроллера (AREF, AVCC) (рис. 1.25).

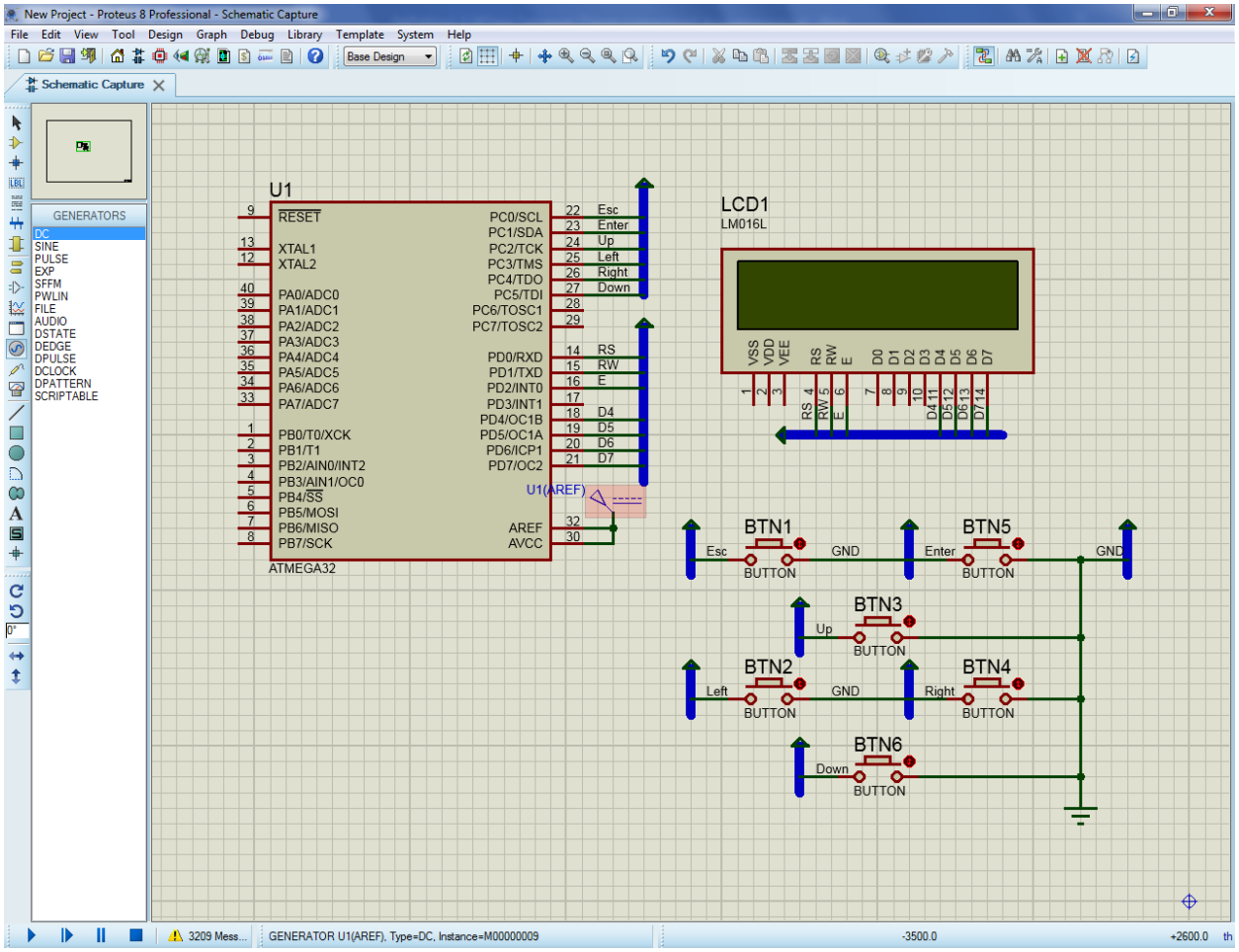

Рис. 1.25 – Источник постоянного напряжения соединен с микроконтроллером.

Далее, нам необходимо выбрать величину подаваемого источником напряжения. Для этого, необходимо зайти в свойства источника напряжения, аналогично, как заходили в свойства микроконтроллера, и выставить необходимый уровень напряжения в поле «Voltage (Volts)» (рис. 1.26). Необходимый уровень напряжения источника – 5 вольт.

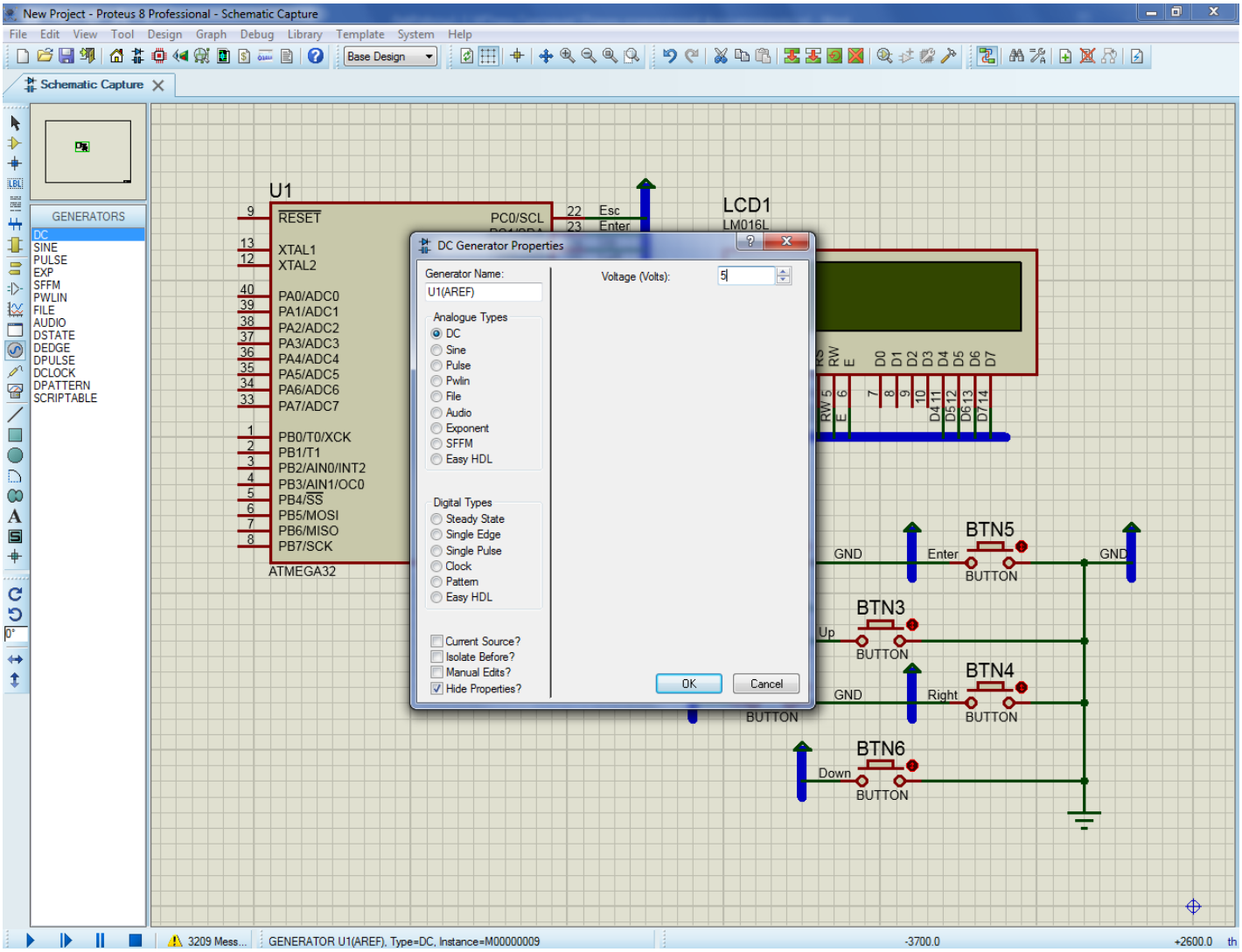

Рис. 1.26 – Окно свойств источника постоянного напряжения.

Далее, дадим соответствующее название каждой из кнопок, для этого на конкретной кнопке необходимо нажать ПКМ и выбрать пункт меню «Edit Properties» (рис. 1.27).

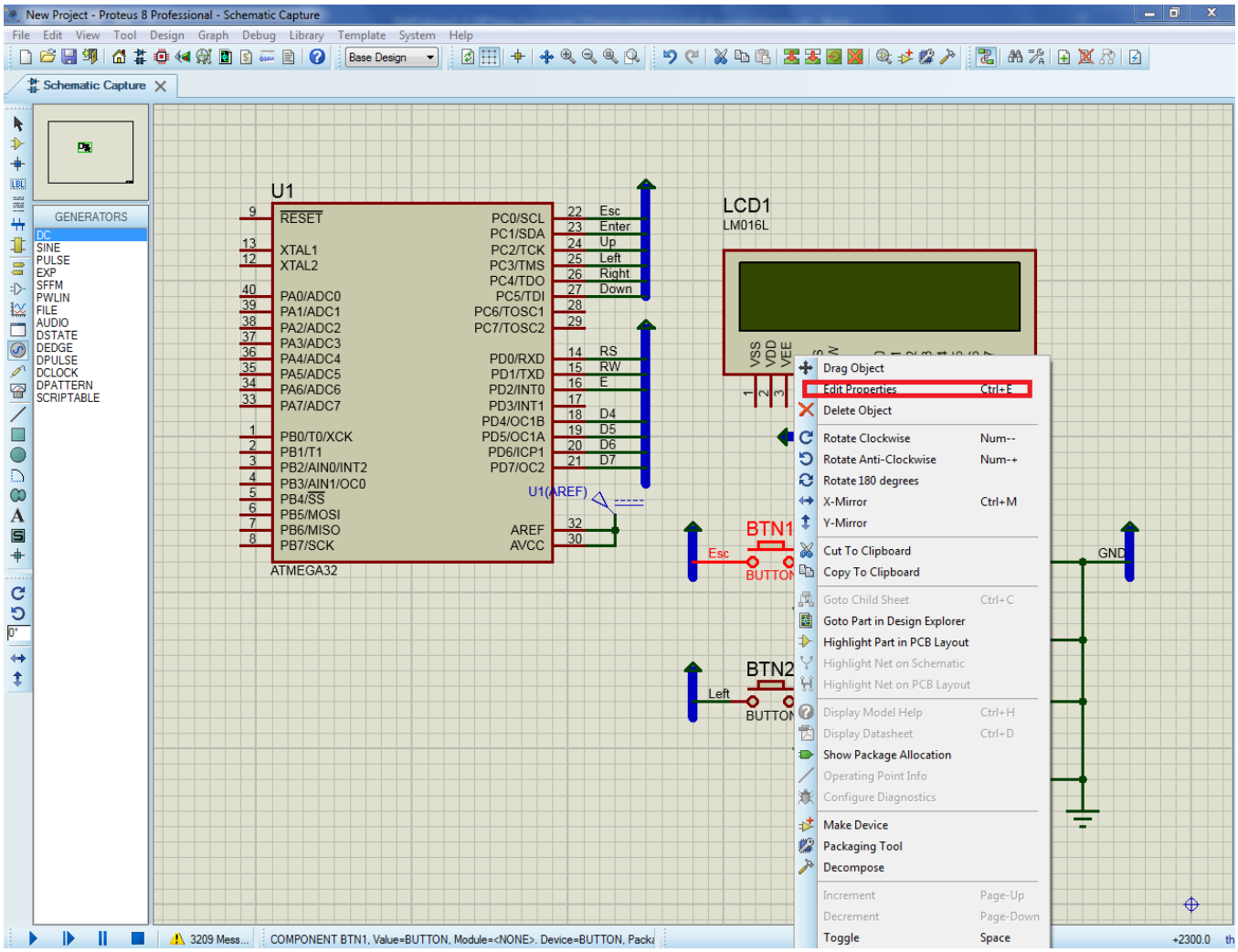

Рис. 1.27 – Меню редактирования элемента button.

В появившемся окне, напротив поля «Part Reference» ставим галочку «Hidden», в поле «Part Value» пишем необходимое название кнопки (рис. 1.28). Производим аналогичные действия для всех остальных кнопок (рис. 1.29).

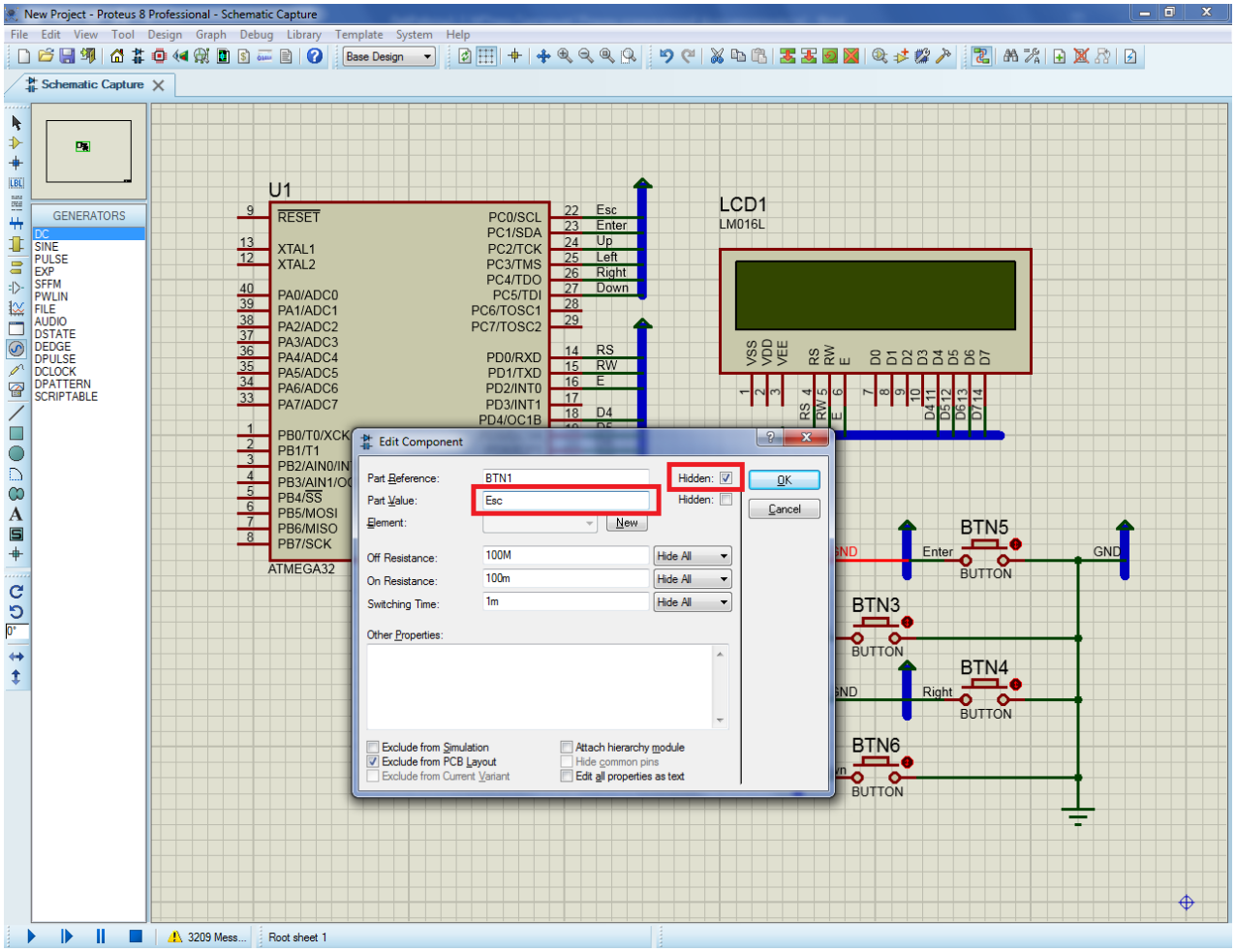

Рис. 1.28 – Свойства элемента button.

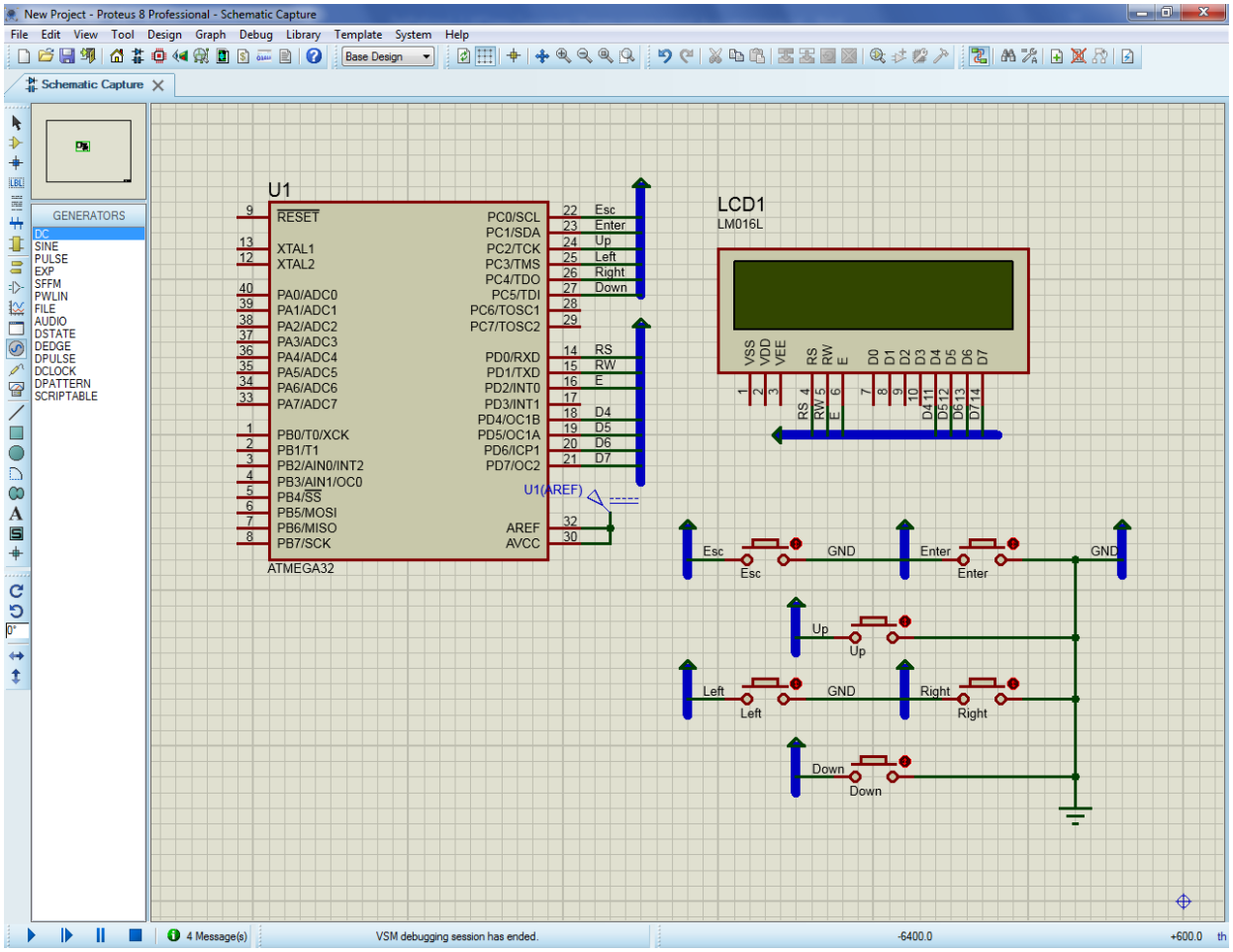

Рис. 1.29 – Кнопки со своими названиями.

Будем подключать не по одному светодиоду, а сразу несколько. Для этой цели идеально подходит элемент, который в поле «keywords» библиотеки элементов будем искать по ключевому слову «bargraph» (рис. 1.30).

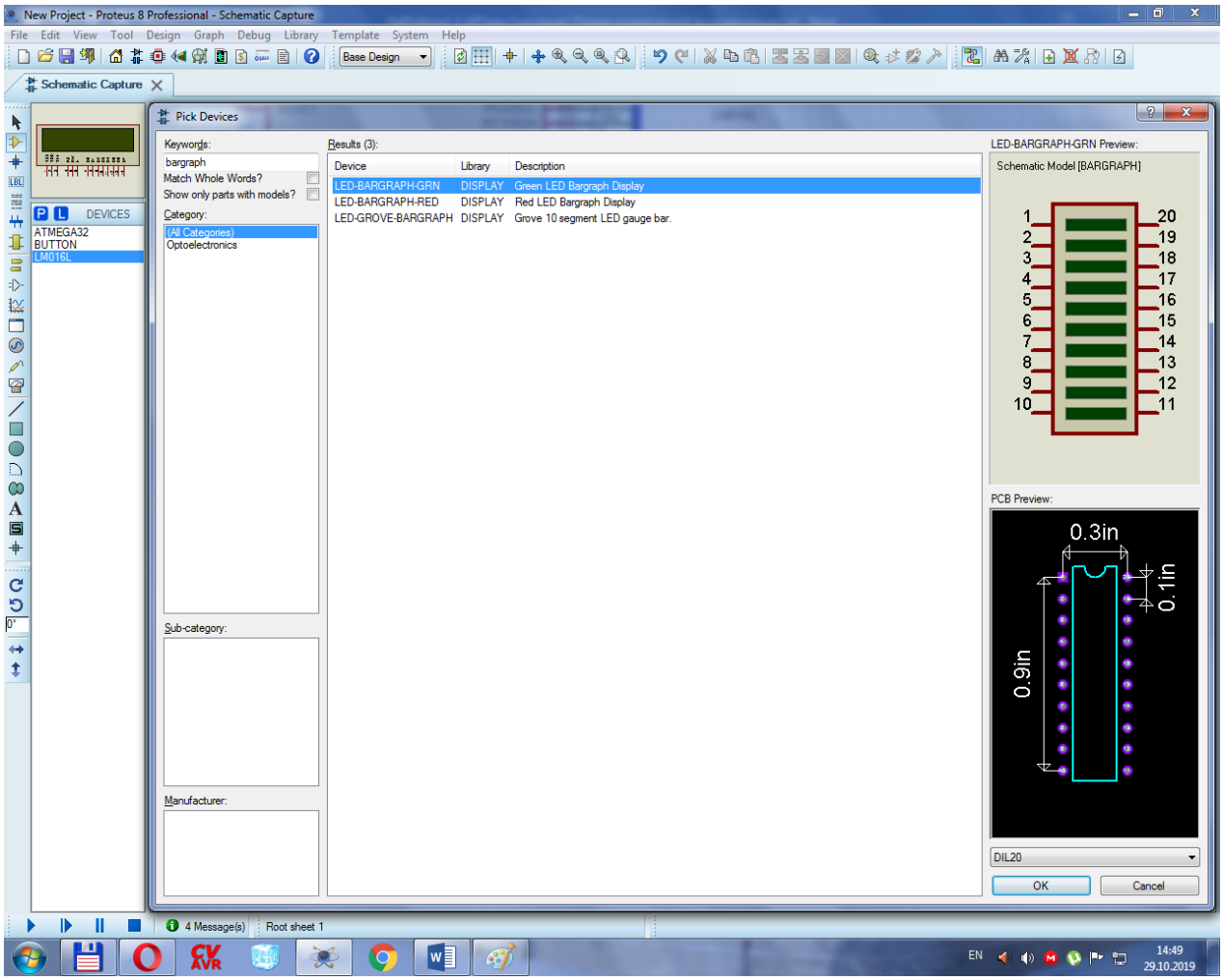

Рис. 1.30 – Поиск элемента bargraph.

Разместим данный элемент ниже порта B. И подключим его к порту B через инструмент BUS. Обратные выводы элемента bargraph соединим с инструментом GROUND (рис. 1.31).

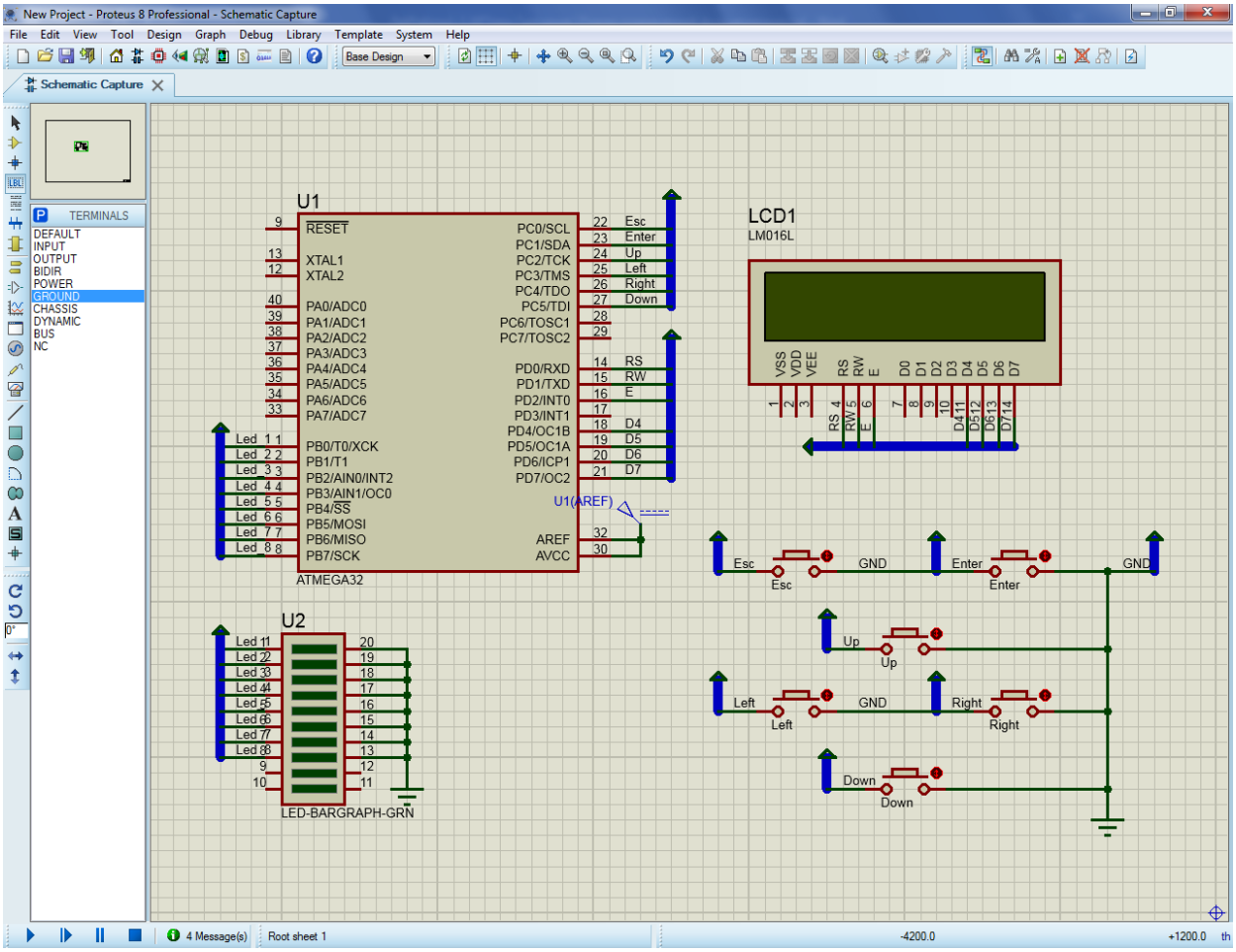

Рис. 1.31 – Подключенный к микроконтроллеру элемент bargraph.

Контрольные вопросы

1. Где найти необходимые для моделирования элементы?
2. Как разместить элементы на рабочем поле?
3. Как соединить элементы между собой?
4. Как работает инструмент BUS?
5. Как изменять свойства элементов?
6. Для чего нужен инструмент GROUND?
7. Как подключить дисплей к микроконтроллеру? Какая шина данных при этом используется?
8. Чем заменены одиночные светодиоды?

Лабораторная работа №2. Создание проетка в среде CodeVisionAVR (исп. версия программы 3.14).

Тема: Изучение базовых принципов создания проекта в программе CodeVisionAVR.

Цель: Овладеть навыками работы с программой CodeVisionAVR.

Описание: В данной лабораторной работе студент знакомится с элементами интерфейса программы CodeVisionAVR. Также будут приобретены практические навыки работы с программой CodeVisionAVR, а также будут рассмотрены вопросы структуры программного кода, созданного генератором кода CodeWizardAVR.

Выполнение работы

1. Создание нового проекта в программе CodeVisionAVR.

При помощи соответствующего ярлыка на компьютере, запустить программу CodeVisionAVR. После запуска будет отображено главное окно программы (рис. 2.1).

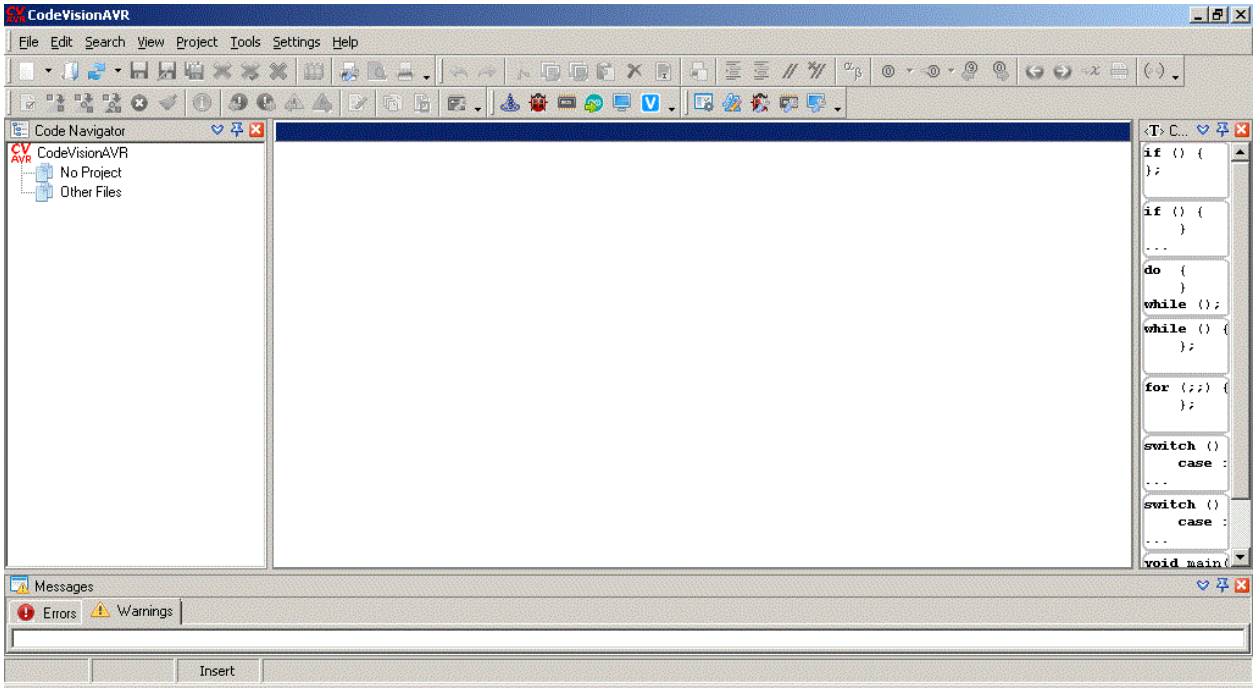

Рис. 2.1 – Главное окно программы CodeVisionAVR.

В левой части окна, можно увидеть поле «Code Navigator», используемое для навигации между файлами проекта. В центре, самое обширное по площади, располагается поле для ввода, отображения и редактирования программного кода и документов проекта. Справа, расположено поле «<T>», с шаблонами программного кода основных операторов языка Си. Внизу, расположена панель для вывода сообщений и ошибок при компиляции проекта. Вверху, стандартная, для большинства программ, иконочная панель основных функций программы.

Для того чтобы создать новый проект, необходимо нажать на вкладку «File» и выбрать пункт «New» -> «Project». Программа предложит запустить генератор кода для написания стандартных комманд и настроек микроконтроллера. Необходимо выбрать «Yes» (рис. 2.2).

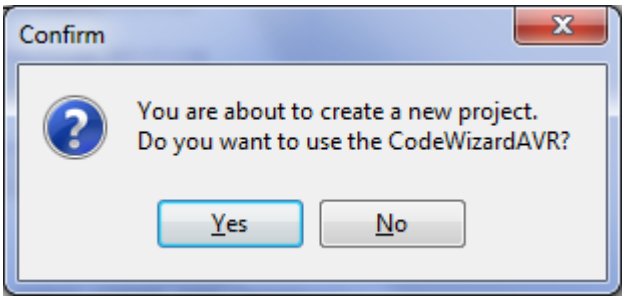

Рис. 2.2 – Диалоговое окно для разрешения на запуск генератора кода.

Откроется начальное окно генератора кода, где будет предложено выбрать тип микроконтроллера. Выбираем «AT90, ATtiny, ATmega» и нажимаем «OK» (рис. 2.3).

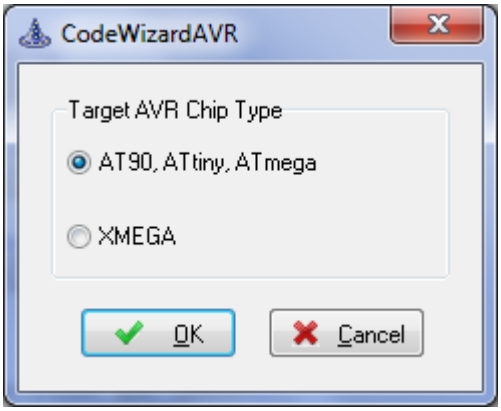

Рис. 2.3 – Выбор типа микроконтроллера в CodeWizardAVR.

Далее будет запущено окно выбора настроек проекта. Оно будет изначально открыто на вкладке «Chip». Здесь нужно выбрать микроконтроллер в поле «Chip», в нашем случае, это ATmega32A. В поле «Clock» ввести частоту «8.0000000» MHz (рис. 2.4).

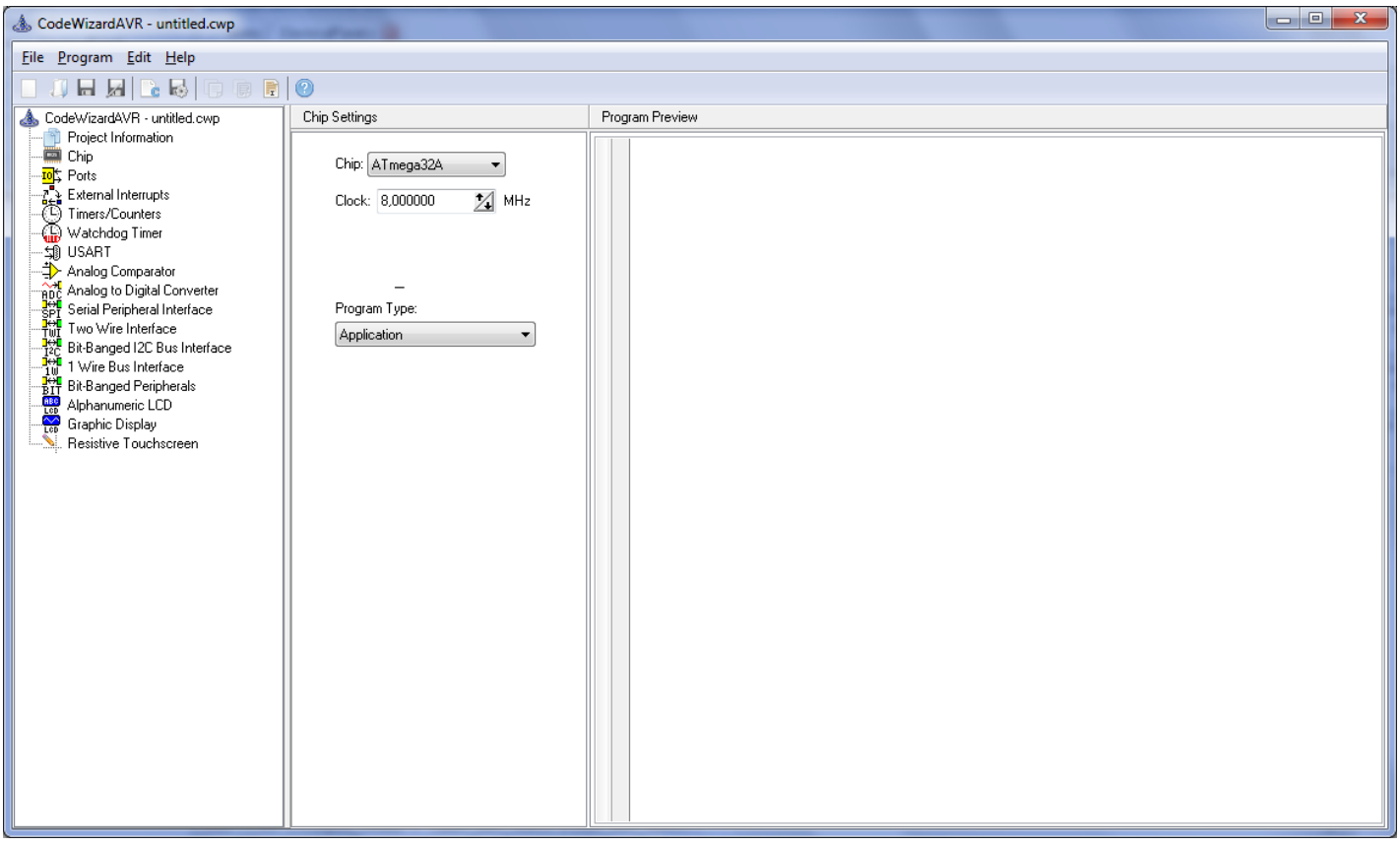

Рис. 2.4 – Вкладка Chip CodeWizardAVR.

Во вкладке «Project Information» вводим данные относительно исполнителя работы. В поле «Project Name:» вводим название проекта, чтобы совпадало (для удобства) с названием проекта созданного в программе Proteus «Ivanov_ESIS-16». В поле «Version:» указываем «1». В поле «Author:» указываем свое имя и фамилию, например «Иван Иванов». В поле «Company:» пишем «ДонНТУ» (рис. 2.5).

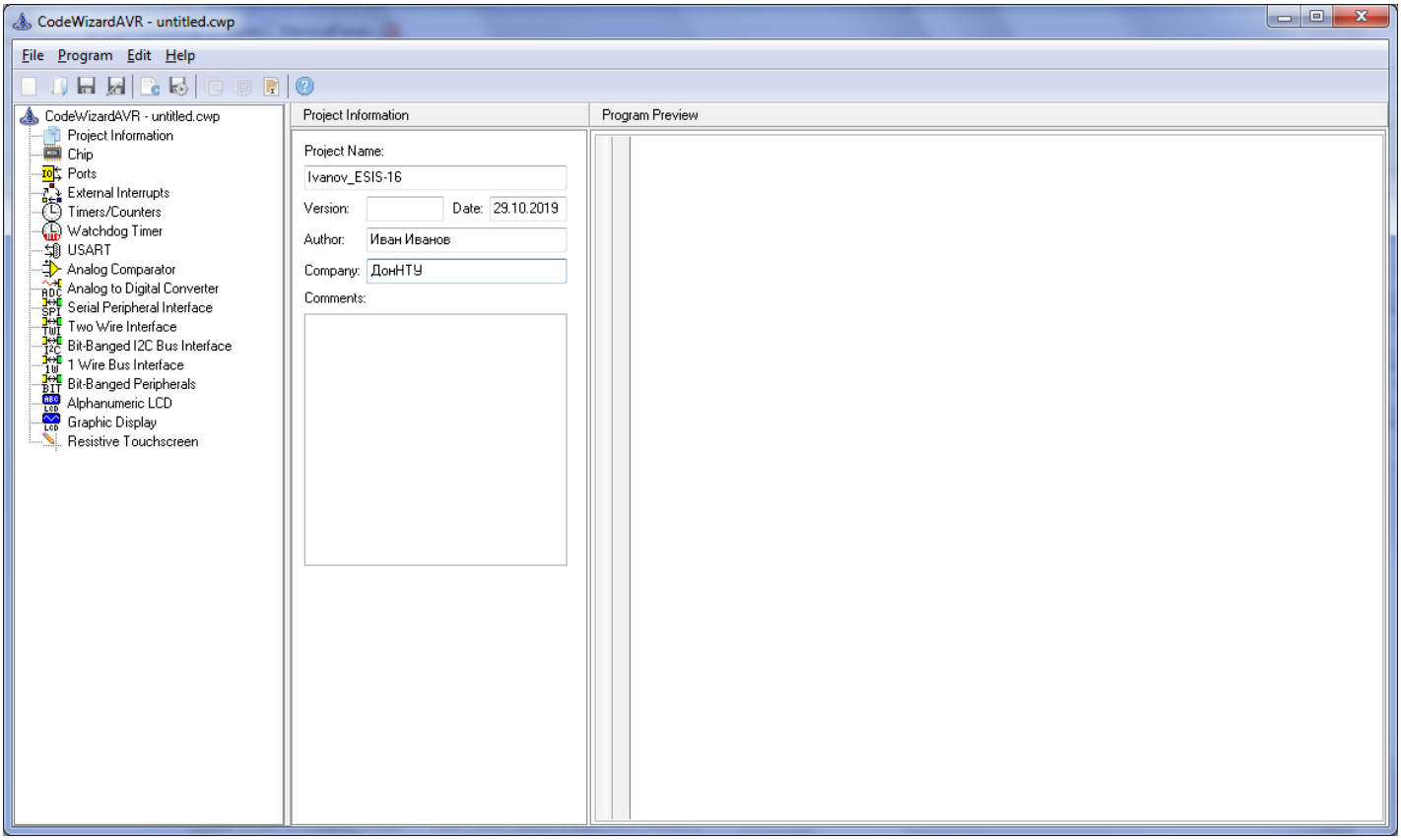

Рис. 2.5 – Вкладка Project Information CodeWizardAVR.

Во вкладке «Ports» имеется возможность изначальной настройки пинов микроконтроллера, отвечающих за ввод и вывод информации, объединенных в порты. Каждый такой пин может находится в 2 основных состояниях: «Вход» и «Выход» (в случае с CodeWizardAVR это «In» и «Out». Помимо этих основных состояний имеется возможность настройки дополнительных параметро для каждого из основных состояний. Для состояния «Вход» это режимы «P» (подключен к питанию через высокоомный резистор – на пине логическая «1») и «T» (пин переведён в режим «Hi-Z» – неподключен ни к минусу питания, ни к плюсу питания, находится в пассивном режиме) (рис. 2.6). Для состояния «Выход» это режимы «0» (подтянуто к минусу питания – на пине логический «0») и «1» (подтянуто к плюсу питания – на пине логическая «1», есть возможность подключения нагрузки до 20 мА).

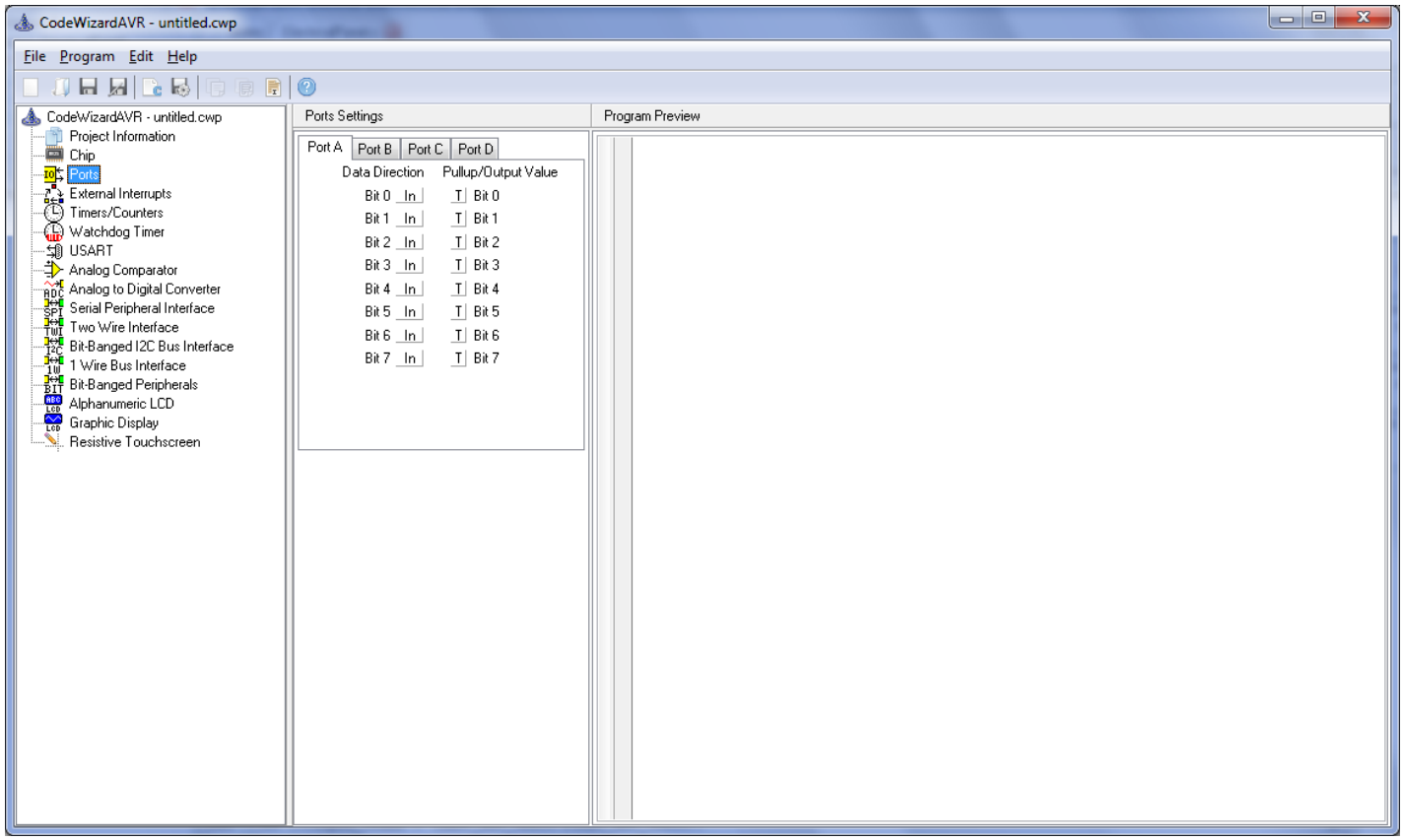

Рис. 2.6 – Вкладка Ports CodeWizardAVR.

Настроим пины порта B на «Выход» с «0», пины порта C с 0-го по 5-ый на «Вход» с «P».

Переходим ко вкладке «Timers/Couners». В ней имеется возможность настройки 3 таймеров микроконтроллера. 0-ой и 2-ой таймеры 8-ми битные, т.е. максимальное значение переменной счетчика таймера будет 255 (применяются для отсчетов коротких временных интервалов). 1-ый таймер 16-ти битный, имеет в своём арсенале старший и младший байты одной переменной (применяется для отсчетов больших временных интервалов). Для начала, настроим таймер 0. Создатели программы упростили нам задачу по выбору периода срабатывания таймера. В поле «Period:» можно выбрать значение времени в милисекундах с которым будет срабатывать таймер, например «1.00000000». После этого необходимо нажать на кнопку «Apply». CodeWizardAVR сам подберет необходимые настройки для таймера, покажет погрешность срабатывания по времени, если невозможно подобрать точные параметры, и после этого таймер 0 настроен (рис. 2.7).

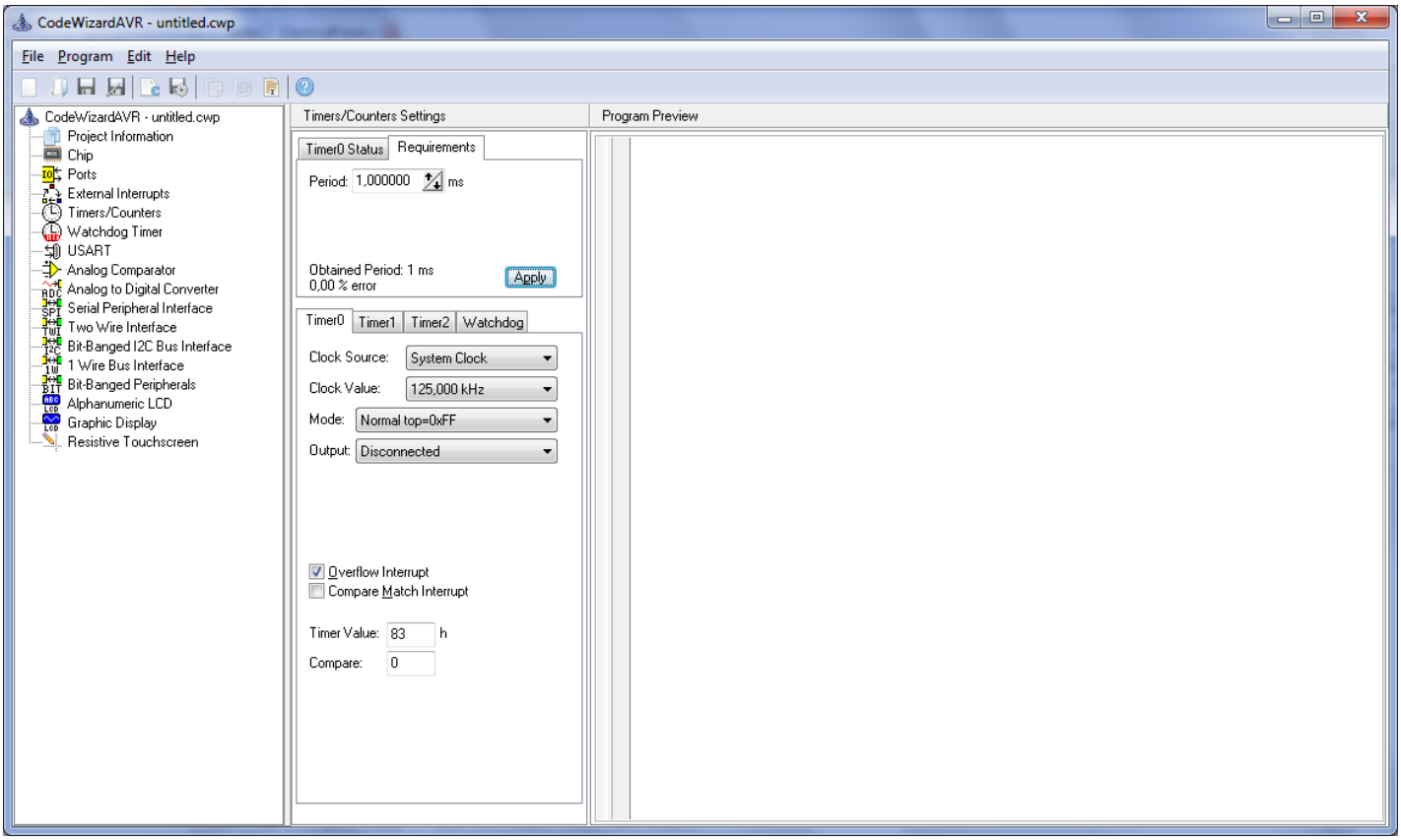

Рис. 2.7 – Вкладка Timers/Counters CodeWizardAVR.

Во вкладке «Analog to Digital Converter» можно настроить аналогово-цифровой преобразователь АЦП. Для этого (рис. 2.8):

- после перехода ко вкладке поставить галочку «ADC Enable», после чего откроются параметры АЦП;
- ставим галочку «Interrupt», чтобы получить доступ к преобразованиям АЦП напрямую;
- выбираем в выпадающем меню «Volt.Ref.:» пин, к которому подключено опорное напряжение, в нашем случае это либо «AREF pin», либо «AVCC pin»;
- в выпадающем меню «Clock:» выбираем частоту работы преобразований АЦП, лучше всего выбирать среднюю частоту между максимальной и минимальной, т.к. на границах частоты работы у данного АЦП наблюдаются сильные погрешности преобразования;
- в выпадающем меню «Auto Trigger Source», т.к. мы не используем наш проект для создания портативных устройств с низким энергопотреблением, выбираем «Free Running»;
- в поле «Automatically Scan Inputs» ставим галочку «Enabled», после чего появится возможность выбора входов для автоматического сканирования, выбираем «First» – «0», «Last» – «7».

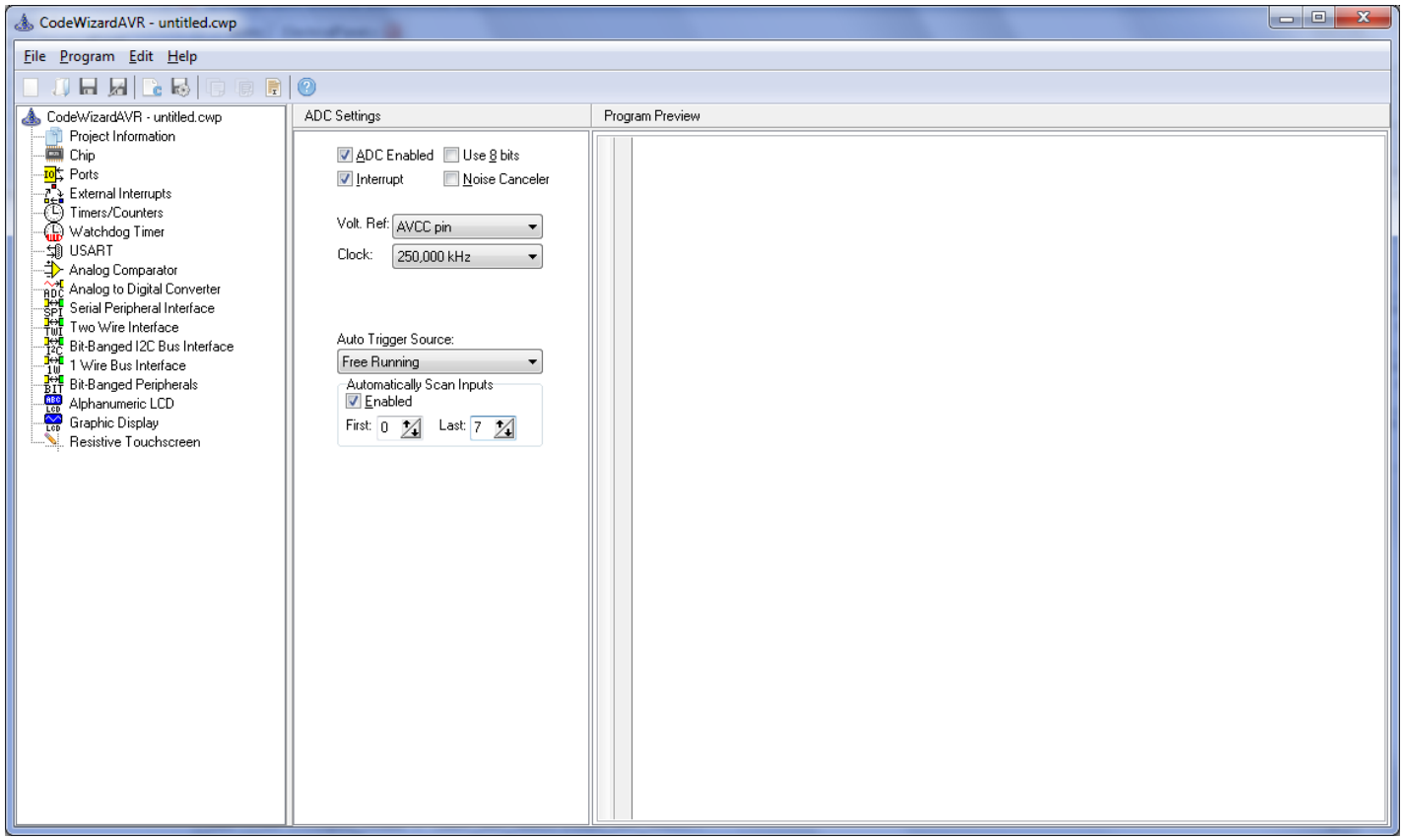

Рис. 2.8 – Вкладка Analog to Digital Converter CodeWizardAVR.

Встроенный АЦП имеет 8 входов от порта А. Но сам АЦП только один. Т.е. для преобразования со всех входов ему нужно преобразовать напряжение с первого входа, переключиться на второй и так далее до последнего, после чего произойдет переключение на первый. Т.е. скорость преобразования относительно изначально выбранной сократится обратно пропорционально количеству задействованных входов.

Следующей настраиваемой вкладкой будет «Alphanumeric LCD». После выбора вкладки ставим галочку «Enable Alphanumeric LCD Support», после чего появятся дополнительные параметры настройки, как во вкладке «Analog to Digital Converter». В поле «Characters/Line:» ставим количество задействованных символов в строке (в нашем случае, «16»). Далее, в поле «Connections» выбираем пины, к которым подключен наш дисплей в проекте Proteus-a. Для этого, во всех пинах просто меняем «PORTA» на «PORTD», сами пины портов уже подключены именно по такому порядку (рис. 2.9).

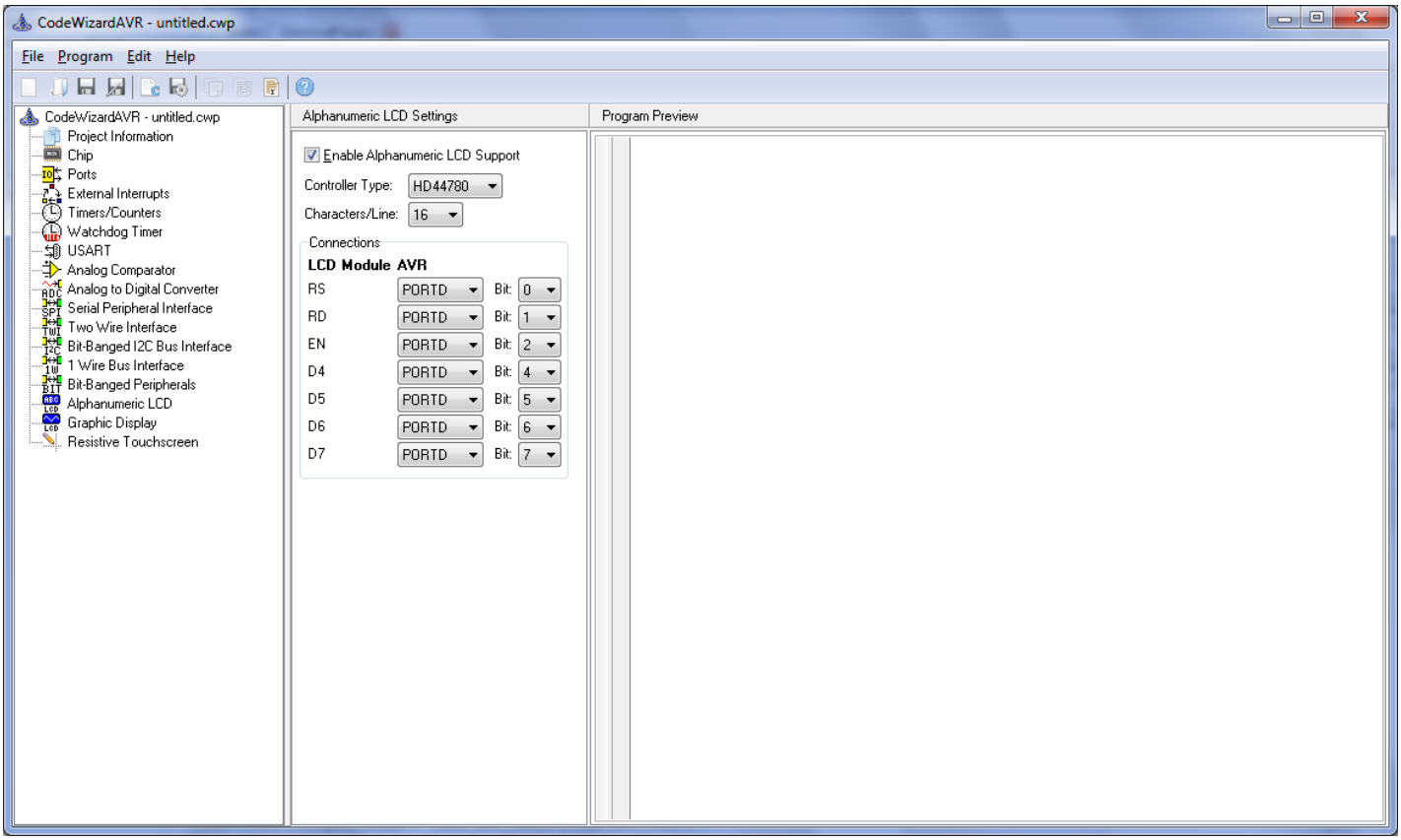

Рис. 2.9 – Вкладка Alphanumeric LCD CodeWizard AVR.

После всех проведенных манипуляций можем смело в верхнем меню нажимать «Program» - «Generate, Save and Exit». После чего необходимо ввести трижды название проекта для сохраняемых файлов с расширениями «*.c», «*.prj», «*.cwp». После чего откроется окно с готовым к работе проектом.

2. Изучение структуры «*.c» файла проекта в программе CodeVisionAVR.

После создания проекта, программа автоматически откроет «*.c» файл (рис.2.10).

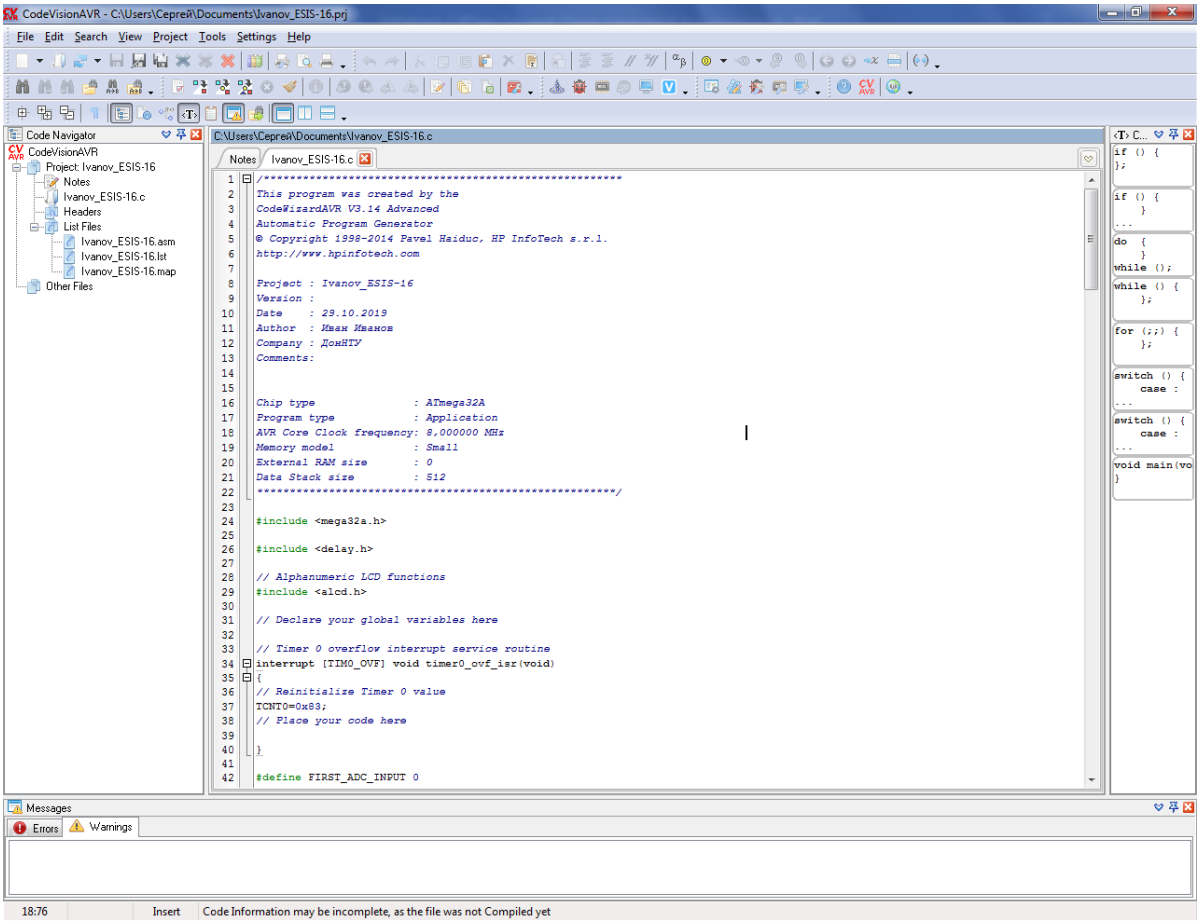

Рис. 2.10 – Открытый проект лабораторной работы №1.

В начале проекта, закомментированные строки (при помощи символов «/*» – начало комментирования, и «*/» – конец комментирования), означающие, в какой программе был создан проект, его название, кем и где создан и для какого кристалла (микроконтроллера) (рис. 2.11).

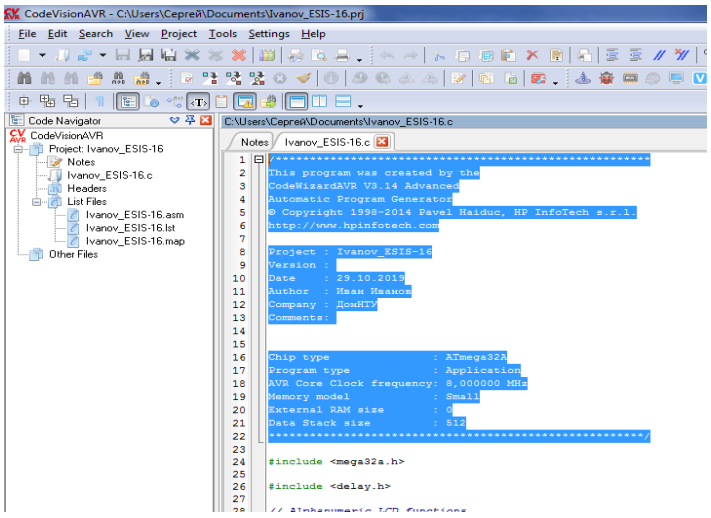

Рис. 2.11 – Информационная часть файла.

Далее, идет список подключаемых библиотек, при помощи оператора «#include», с названием библиотеки и расширением в кавычках (<имя файла библиотеки.h>) (рис. 2.12).

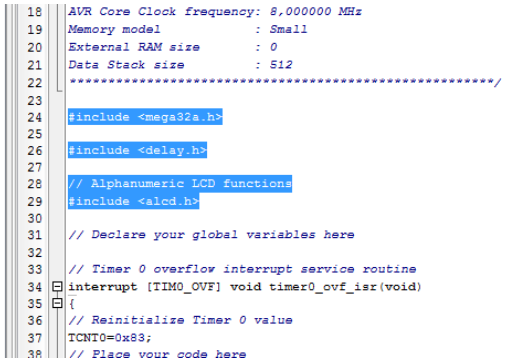

Рис. 2.12 – Список подключаемых дополнительных файлов.

Следом, идет список объявляемых глобальных переменных (после комментария «//Declare your global variables here») (Рис. 2.13).

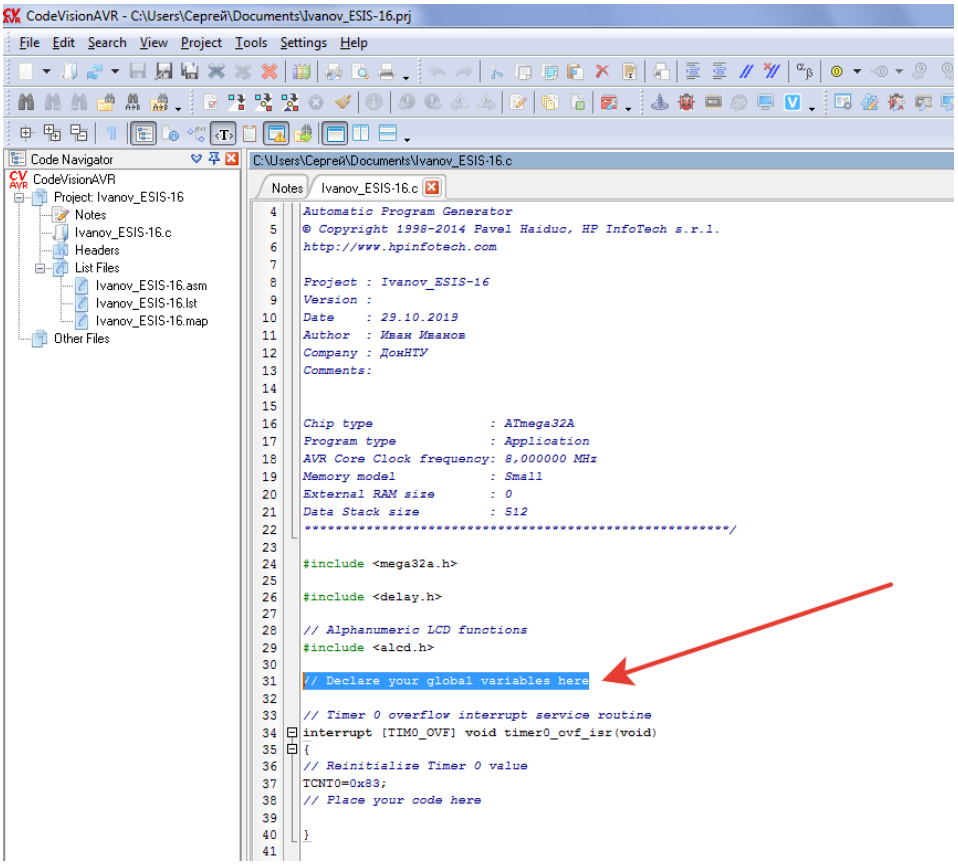

Рис. 2.13 – Место для объявления глобальных переменных.

Ниже списка глобальных переменных, в нашем файле, идут глобальные замены, обозначенные оператором «#define». После данного оператора указывается текст, который необходимо будет заменить, в тексте данного файла, на тот текст, который указан, через пробел, следом за этим текстом (например, «#define» (оператор «заменить на») – «Что найти в тексте файла» – «На что заменить в момент компиляции проекта») (*Изначально, они не присутствуют!*).

После этого, идет функция прерывания восьмибитного таймера T0 (т.к. мы его активировали при создании проекта). В ней, описано, что должен делать микроконтроллер в каждые строго установленные интервалы времени, например, как в нашем случае, каждую 1 мс. В нём может быть задан счетчик времени, с помощью которого осуществляется мигание светодиодами. Полсекунды, светодиод горит, ещё половину – погашен (Рис. 2.14).

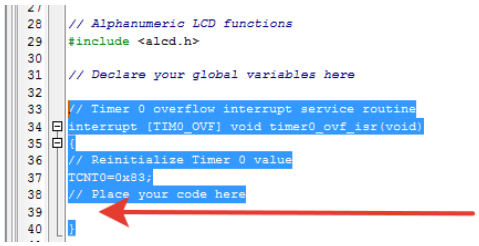

Рис. 2.14 – Функция прерывания таймера T0.

Ниже, в нашем проекте, находится функция прерывания, по завершению преобразования АЦП. В ней указан номер входа (канала), с которого будет браться напряжение для сравнения с опорным. После данных действий идет задержка 10 мкс для стабилизации напряжения на входе, и запуск нового измерения (Рис. 2.15).

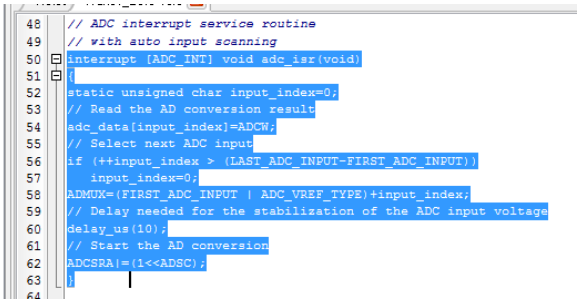

Рис. 2.15 – Функция вызова прерывания после окончания преобразования АЦП.

Далее, находится тело главной функции программы «void main (void) {}». В ней находятся все настройки нашего микроконтроллера:

1. Настройки портов («// Input/Output Ports initialization»):
   Структура «DDRx» отвечает за режим работы конкретных пинов порта:
   - «0» – «вход»;
   - «1» – «выход».

   Структура «PORTx» отвечает за логическую часть этих пинов:
   - «0» в режиме «вход» – режим «Hi-Z» – пин находится в абстрактном состоянии, нет ни логического «0», ни логической «1» – режим оожидания;
   - «1» в режиме «вход» – на пине присутствует логическая «1», пин подтянут к питанию внутри микроконтроллера через высокоомный резистор;
   - «0» в режиме «выход» – пин подтянут к минусу питания через токоограничивающий резистор;
   - «1» в режиме «выход» – пин подтянут к плюсу питания через токоограничивающий резистор.
2. Настройки таймера T0 («// Timer/Counter 0 initialization»).
3. Настройки таймера T1 («// Timer/Counter 1 initialization»).
4. Настройки таймера T2 («// Timer/Counter 2 initialization»).
5. Общие настройки запуска тактирования таймеров («// Timer(s)/Counter(s) Interrupt(s) initialization»).
6. Настройки внешних прерываний («// External Interrupt(s) initialization»).
7. Настройки интерфейса передачи данных USART («// USART initialization»).
8. Настройки аналогового компаратора («// Analog Comparator initialization»).
9. Настройки аналогово-цифрового преобразователя АЦП («// ADC initialization»).
10. Настройки интерфейса передачи данных SPI («// SPI initialization»).
11. Настройки интерфейса передачи данных TWI, тот же I2C («// TWI initialization»).

12. Настройки, в нашем случае, подключенного дисплея («// Alphanumeric LCD initialization»).
13. Разрешение глобальных прерываний («// Global enable interrupts»).

Далее, находится, необходимый нам, бесконечный цикл «while (1)». Его мы используем для основного кода, который будем использовать в дальнейшем. Цикл «while (1)» является бесконечным с той единственной целью, что микроконтроллер устроен так, что должен быть все время в работе, иначе он перестанет реагировать на запросы и тогда его можно будет только «сбросить» подав логический «0» на пин «Reset», либо отключить от питания и включить заново.

3. Задание.

- Открыть проект в программе Proteus из лабораторной работы №1.
- Собрать исходную схему и настроить.
- Создать проект в программе CodeVisionAVR. Изучить программный код исполняемого файла проекта.
- Скомпилировать проект при помощи сочетания горячих клавиш «Ctrl+F9».
- Подключить «*.hex» файл прошивки к модели микроконтроллера в проекте программы Proteus из лабораторной работы №1. Запустить симуляцию. Убедиться в работоспособности проекта.

В отчете предоставить:

- скриншот собранной схемы;
- программный код (листинг программы).

4. Контрольные вопросы:

1) Какая программа были рассмотрены в данной лабораторной работе?
2) Назовите основную структуру программного кода данного проекта.
3) Почему цикл «while (1) {}» называется «бесконечным циклом»?

Лабораторная работа №3 – Работа в CodeVision AVR и Proteus. Разработка и выполнение простейшей программы.

Тема: Изучение базовых принципов создания простейшей программы в программе CodeVisionAVR и запуска симуляции в программе Proteus.

Цель: Овладеть навыками работы с программным кодом на простейшем уровне.

Описание: Приобретение практических навыков программирования в программе CodeVisionAVR.

Выполнение работы

1. Написание простейшей программы.

Запустите программы **CodeVisionAVR** и **Proteus.** Откройте созданные ранее проекты и перейдите в проект в программе CodeVisionAVR.

Для начала, создадим глобальную переменную для счетчика (в предназначенном для этого месте программного кода). Пусть это будет переменная «time». Присвоим ей начальное значение «0». Т.к. эта переменная будет являться счетчиком, дадим ей тип «long». Например «long time=0;»

Перейдем в функцию прерывания таймера T0. В предназначенном месте для программного кода пользователя запустим наш счетчик на увеличение значения. Далее, создадим условие окончания счета при помощи условного оператора «if» и остановим счет на значении «1 секунда», и обнулим значение счетчика. После этого, создадим ещё 2 условных оператора для значений счетчика «до 0,5 секунды» и «после 0,5 секунды». В них мы присвоим поочередное изменение состояния пина «PB0». Пример кода:

```
time++;
if(time<500){PORTB.0=0;}
if(time>=500){PORTB.0=1;}
if(time==1000){time=0;}
```

Запускаем компиляцию проекта и переходим в проект в программе Proteus. Запускаем там симуляцию проекта. Если все сделано правильно. Первый светодиод на элементе bargraph будет мигать с частотой 1 Гц (Рис. 3.1).

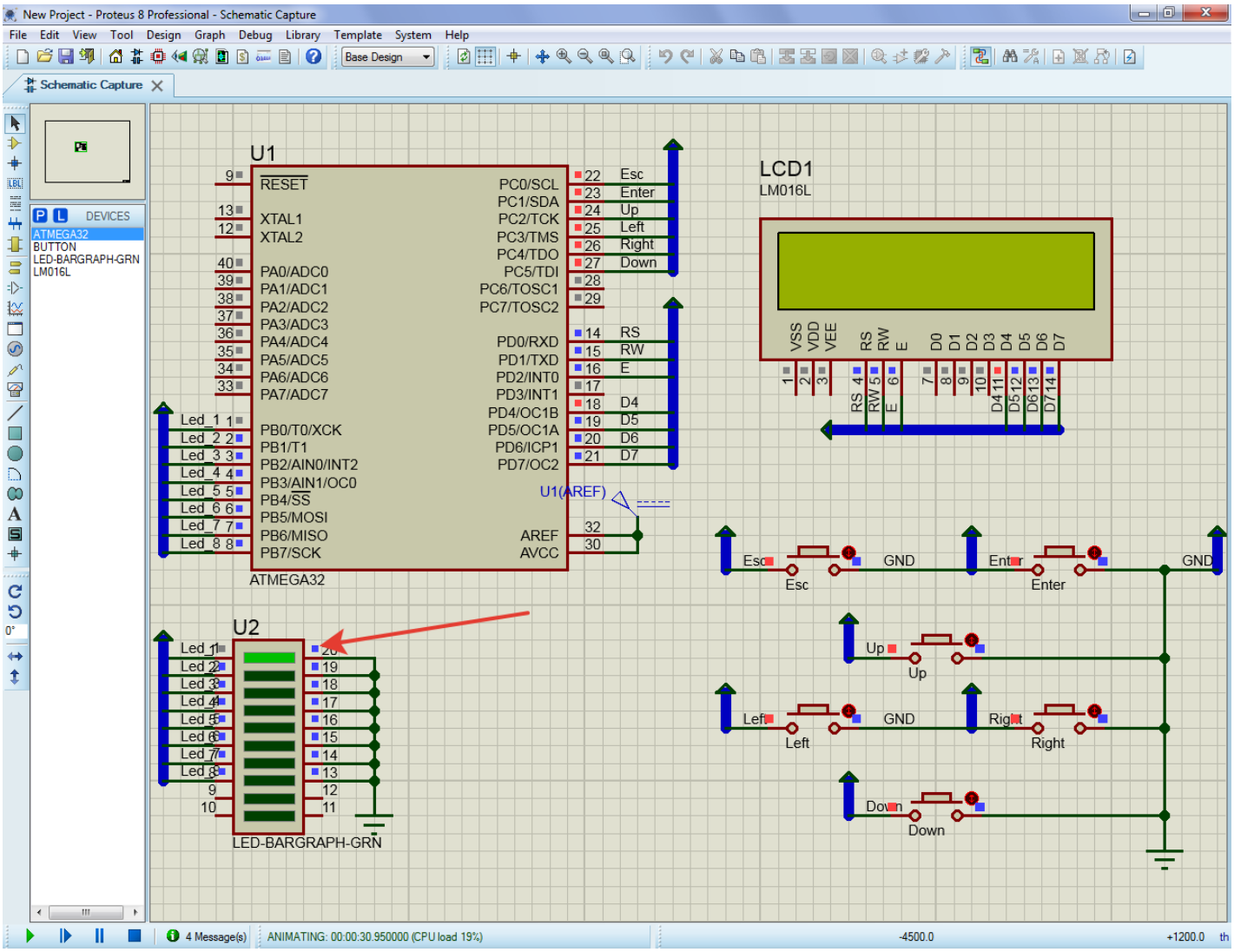

Рис. 3.1 – Мигающий светодиод элемента bargraph.

2. Обработка нажатия клавиш.

Для того, чтобы определить была ли нажата какая-либо клавиша, необходимо понимание процесса данного действия. Сам по себе микроконтроллер не обладает интеллектом и не обладает зрением и т.п. Поэтому он самостоятельно не способен распознавать такое понятие как «клавиша» и уж тем более «нажатие клавиши». Мы будем применять алгоритм нажатия клавиши, основанный на фильтрации информационного шума с использованием таймера микроконтроллера. Последовательность действий такова:

- пин порта, к которому подключена кнопка, находится в состоянии высокоомного входа с внутренней подтяжкой к напряжению питания;
- при нажатии клавиши, которая своей обратной частью подключена к минусу питания, пин микроконтроллера опускается до минуса питания, который соответствует логическому нулю;
- во время механического соприкосновения контактов реальной кнопки неизбежно происходит их дребезг, что и образовывает информационный шум;
- зная, что при замыкании контактов клавиши образовывается шум, его возможно отфильтровать при помощи таймера микроконтроллера, сделав задержку срабатывания по времени;
- при отпускании клавиши, пин возвращает себе высокий логический уровень, соответственно и логическую единицу.

Зная все эти особенности, возможно довольно несложно выполнить обработку нажатия клавиши:

1. Объявить переменные нажатой клавиши и промежуточной переменной нажатой клавиши, а также переменные счётчики для отстройки по дребезгу контактов:

```
int key=0, key0=0;
long nEsc=0,nEnter=0,nUp=0,nLeft=0,nRight=0,nDown=0;
```

2. Далее сразу присвоим названия клавиш значениям переменных key и key0 отличные от их значений по умолчанию:

```
#define Esc 1
#define Enter 2
#define Up 3
#define Left 4
#define Right 5
#define Down 6
```

3. Также рекомендуется сделать присваивания пинов микроконтроллера, к которым подключены соответствующие клавиши:

```
#define pEsc PINC.0
#define pEnter PINC.1
#define pUp PINC.2
#define pLeft PINC.3
#define pRight PINC.4
#define pDown PINC.5
```

4. Само по себе название пина является логическим значением и по умолчания равно логической единице. Из этого следует, что при нажатой клавише на пине будет обратное значение. Помимо этого, необходимо сделать отстройку по времени от дре-

безга контактов кнопки в функции прерывания таймера. Будем считать клавишу нажатой лишь после того, как она будет отпущена:

```
if(!pEsc){nEsc++;}
if(!pEnter){nEnter ++;}
if(!pUp){nUp ++;}
if(!pLeft){nLeft ++;}
if(!pRight){nRight ++;}
if(!pDown){nDown ++;}
if(pEsc&&nEsc>10){key0=Esc;nEsc=0;}
if(pEnter&&nEnter >10){key0=Enter;nEnter =0;}
if(pUp&&nUp>10){key0=Up;nUp=0;}
if(pLeft&&nLeft>10){key0=Left;nLeft=0;}
if(pRight&&nRight>10){key0=Right;nRight=0;}
if(pDown&&nDown>10){key0=Down;nDown=0;}
```

5. В бесконечном цикле while(1) необходимо добавить перенос значения нажатой клавиши из промежуточной переменной key0 в окончательную переменную нажатой клавиши key:

```
if(key0){key=key0;key0=0;}
```

6. Теперь для того, чтобы совершать какие-либо действия с клавишами необходимо на каждую из клавиш запрограммировать определённое действие программы и не забыть после выполнения этих действий освободить переменную нажатой клавиши, присвоив ей значение по умолчанию, во избежание повторного программного нажатия клавиши по ходу программы, например:

```
if(key==Enter){PORTB.1=1;key=0;}
```

3. Вывод на дисплей информации.

Для работы с LCD-дисплеем в программе CodeVisionAVR присутствует библиотека, которая содержит полезные функции для работы с дисплеем alcd.h. Легче всего её подключить на этапе создания проекта, указав, что будет подключен дисплей с указанием пинов микроконтроллера, к которым будет подключен дисплей.

Для вывода простой текстовой информации на дисплей, можно воспользоваться функцией lcd_putsf(«Текст») из состава библиотеки, где в кавычках необходимо указать текст, который необходимо вывести, например:

```
lcd_putsf(«Hello, world!»);
```

Для вывода заранее неизвестного текста, например, значение переменной-счётчика, используется несколько иной подход:

1. Подключение библиотеки stdio.h.

2. Объявление переменной массива для хранения текстовой строки, например:

```
char lcd_buffer[33];
```

3. Создание строки при помощи функции sprintf:

```
    sprintf(lcd_buffer, «Текст или флаг возможного значения перемен-
ной», переменные по очереди расстановки флагов);
```

4. Необходимо указать в какое именно место дисплея необходимо вывести данный текст, при помощи функции lcd_gotoxy(координата по X, координата по Y).

5. Выводим строку используя функцию lcd_puts(Переменная массива строки).

4. Задание.

- Открыть проект в программе Proteus из лабораторной работы №1.
- Собрать исходную схему и настроить.
- Создать проект в программе CodeVisionAVR. Изучить программный код исполняемого файла проекта.
- Написать программу, соответствующую индивидуальному заданию (Приложение А).
- Скомпилировать проект при помощи сочетания горячих клавиш «Ctrl+F9».
- Подключить «*.hex» файл прошивки к модели микроконтроллера в проекте программы Proteus из лабораторной работы №1. Запустить симуляцию. Убедиться в работоспособности проекта.

В отчете предоставить:

- скриншот собранной схемы;
- программный код (листинг программы).

ПРИЛОЖЕНИЕ А

Задание	Примечание
Составить меню из X пунктов.	X – соответствует последней цифре зачётной книжки.
Составить меню в виде списка с возможностью перехода в подменю на каждом пункте меню при помощи клавиш «влево» и «вправо».	Для нечётной последней цифры зачётной книжки (но не менее 3х).
Составить меню в виде 2 колонок с возможностью переключения между колонками и по колонкам вертикально.	Для чётной последней цифры зачётной книжки (но не менее 4х).
В первом пункте меню реализовать включение 3 светодиодов по очереди, в зависимости от величины напряжения полученного с АЦП.	Для нечётной последней цифры зачётной книжки.
В первом пункте меню реализовать включение по очереди, в зависимости от величины напряжения полученного с АЦП, соответствующей надписи об уровне напряжения.	Для чётной последней цифры зачётной книжки.
Во втором пункте меню реализовать бегущее переключение, по очереди («бегущий огонь»), X светодиодов.	X - для нечётной предпоследней цифры зачётной книжки (но не менее 3х).
Во втором пункте меню реализовать бегущее переключение, по очереди («бегущий огонь»), светодиодов.	Для чётной последней цифры зачётной книжки (но не менее 4х).
В третьем пункте меню реализовать ручное переключение светодиодов при помощи нажатия клавиш.	Для чётной предпоследней цифры зачётной книжки клавишами «вверх» и «вниз».
В третьем пункте меню реализовать ручное переключение светодиодов при помощи нажатия клавиш.	Для нечётной предпоследней цифры зачётной книжки клавишами «влево» и «вправо».
Реализовать в третьем пункте меню возможность цикличного переключения светодиодов.	Для чётной 3-ей от конца цифре зачётной книжки.
Реализовать в третьем пункте меню возможность не цикличного переключения светодиодов.	Для нечётной 3-ей от конца цифре зачётной книжки.
При запуске схемы выдавать на дисплее приветственную надпись.	«Фамилия, название группы» - для нечётной последней цифры зачётной книжки.
При запуске схемы выдавать на дисплее приветственную надпись.	«Фамилия, имя» - для чётной последней цифры зачётной книжки.
Реализовать возможность цикличного переключения пунктов меню.	Для чётной предпоследней цифры зачётной книжки.
Реализовать возможность не цикличного переключения пунктов меню.	Для нечётной предпоследней цифры зачётной книжки.

В контрольной работе предоставить:

1. Титульную страницу (Приложение Б).
2. Программный код из программы CodeVisionAVR с комментариями (листинг программы).
3. Скриншоты схемы из программы Proteus с демонстрацией работоспособности.

При защите контрольной работы знать:

1. Структуру программного кода из программы CodeVisionAVR.
2. Уметь составлять схемы в программе Proteus.
3. Умение переделывать программный код в программе CodeVisionAVR под соответствующее задание преподавателя.

Примечание: работа считается **успешно выполненной** при проверке преподавателем работоспособности программного кода на **реальном** оборудовании.