МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Ю.В. Китаев

ЛАБОРАТОРНАЯ УСТАНОВКА НА ОСНОВЕ ВИРТУАЛЬНЫХ ПРИБОРОВ И USB ИНТЕРФЕЙСА

Учебное пособие

Санкт-Петербург

2012

Китаев Ю.В. “Лабораторная установка на основе виртуальных приборов и USB интерфейса ”. Учебное пособие: СПб: СПб НИУ ИТМО, 2012. ___с.
Приведены лабораторные работы по проектированию и программированию автоматизированной системы управления и сбора данных на основ PIC контроллеров с применением виртуальных приборов и USB интерфейса.

Для студентов, обучающихся по направлениям “Приборостроение”, “Телекоммуникации” и “Оптотехника”: 210401 Физика и технология элементов систем оптической связи, 200600.62 Фотоника и оптоинформатика, 20020104 Лазерная технология

Рекомендовано к печати Советом ИФФ от 05 октября 2011г., протокол №2.

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики» на 2009–2018 годы.

О Г Л А В Л Е Н И Е

ЛАБОРАТОРНАЯ РАБОТА №17
“СИСТЕМА УПРАВЛЕНИЯ, СБОРА И МОНИТОРИНГА ДАННЫХ НА ОСНОВЕ МК PIC18F4550 СО ВСТРОЕННЫМ USB ИНТЕРФЕЙСОМ”

ЦЕЛЬ РАБОТЫ

1. Изучить совместную работу компьютера и пользовательского периферийного устройства (рабочей платы) с USB интерфейсом,
2. Научиться разрабатывать измерительно-управляющие системы с использованием виртуальных приборов.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. Разработать виртуальный прибор для управления сбором и отображением данных с датчиков температуры и напряжения,
2. Виртуальный прибор должен иметь цифровой и аналоговый измерители, ленточный самописец, светодиоды, а также органы управления,
3. Написать программу для виртуального прибора,
4. Написать программу для микроконтроллера PIC18F4550 с аппаратной поддержкой USB протокола (опционально).

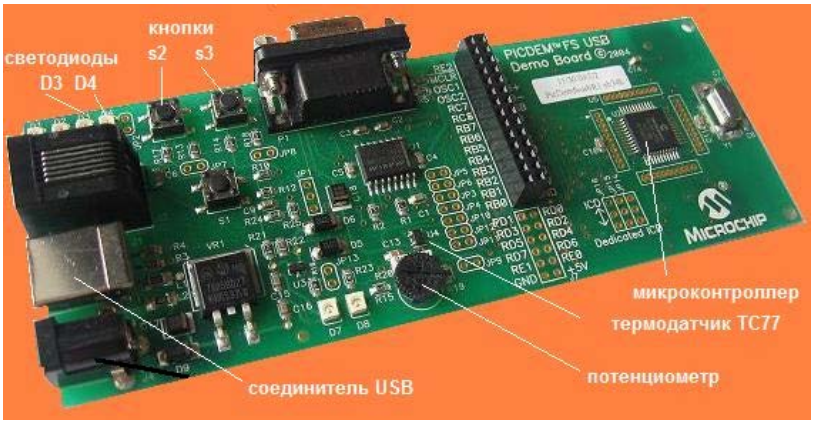
Разработка USB устройств с “нуля”, намного сложнее, чем устройств с интерфейсом RS-232. Для облегчения перехода с интерфейса RS-232 на USB, в Windows имеется возможность использования интерфейса USB в качестве виртуального коммуникационного COM-порта. В этом случае разработчику не нужно модифицировать свое программное обеспечение (ПО) для компьютера, а также писать драйверы, т.к. используются стандартные драйверы Windows. Такой метод позволяет использовать в ПО имеющиеся наработки, при этом команды для COM порта будут транслированы драйвером для выполнения с помощью USB интерфейса. Скорость обмена данными при этом методе доходит до 640Кб/сек, что значительно больше скорости при использовании RS-232.

Другим положительным моментом является то, что разработчику ПО не обязательно знать все тонкости USB протокола.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

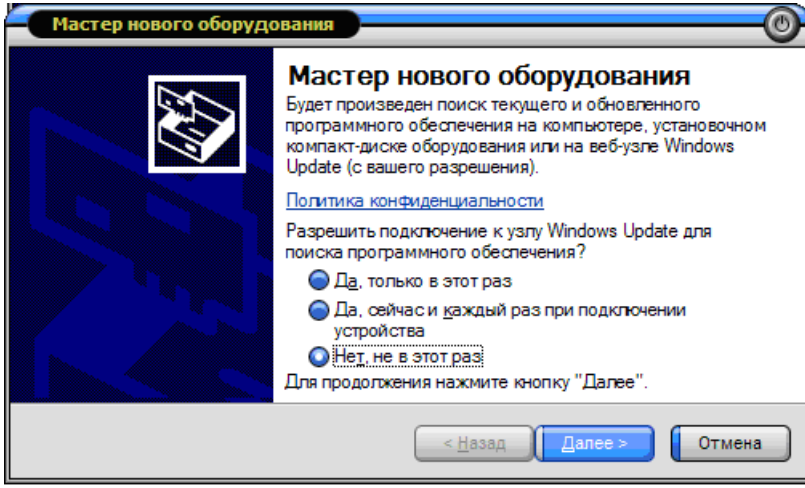
Первое подключение рабочей (целевой) платы "PICDEM FS USB" к USB порту компьютера.

На рабочей плате установлены: микроконтроллер PIC18F4550 с аппаратной поддержкой USB интерфейса, датчик температуры TC77, потенциометр (он же датчик напряжения или угла поворота), два рабочих светодиода и две кнопки.

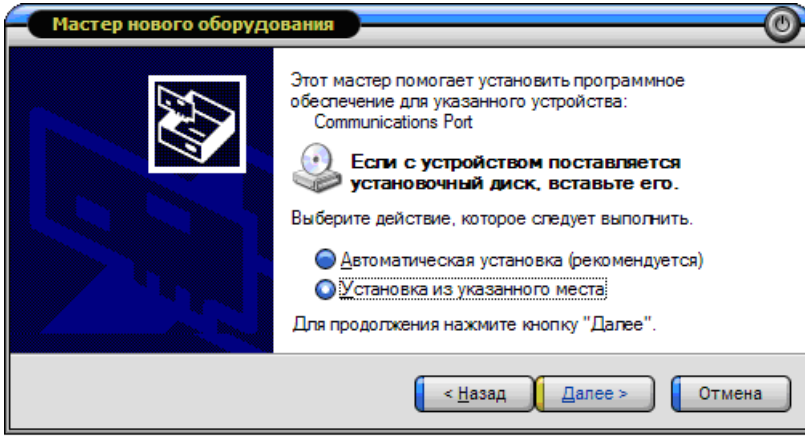


Сначала, в соответствии со спецификацией USB, необходимо указать класс нашего устройства (целевой платы) - Communication Device Class – CDC, который включает стандартную эмуляцию последовательного COM порта на основе USB интерфейса. Если этот этап был проделан ранее, то необходимость в нем отпадает.

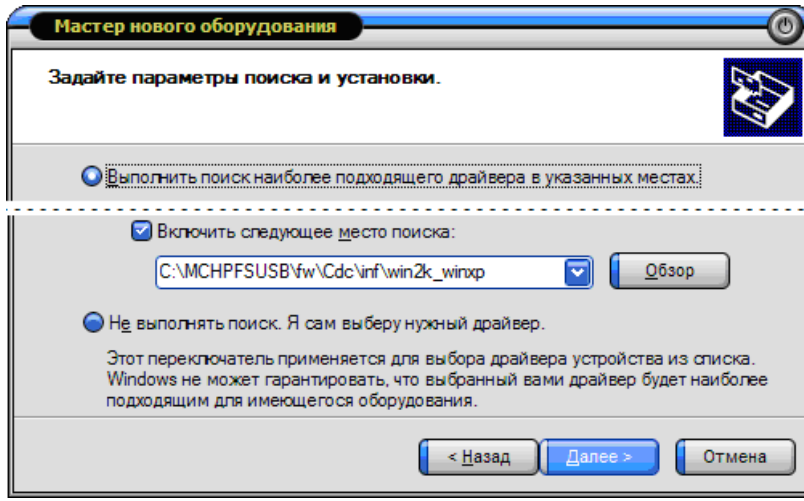
При первом подключении платы, Windows с помощью технологии Plug'n'Play находит новое устройство и определяет его класс – CDC RS232 Emulation (Communications Port). На панели задач появляются соответствующие сообщения (balloons или "пузыри") и стартует "Мастер установки программного обеспечения нового оборудования".



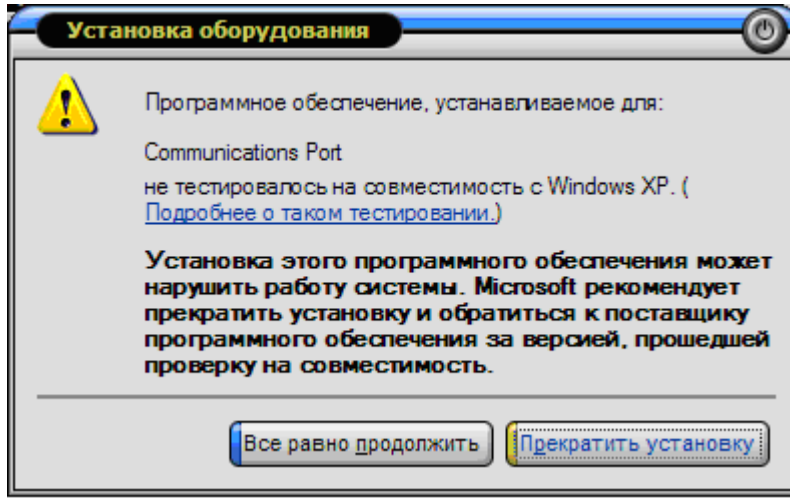
Далее в следующем диалоговом окне отметьте “Установка из указанного места”



Затем отметьте “Включить следующее место поиска”.

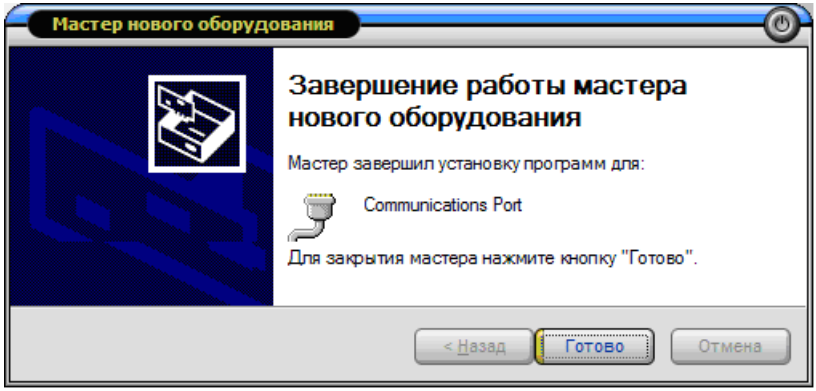


Нажмите кнопку “Все равно продолжить”.

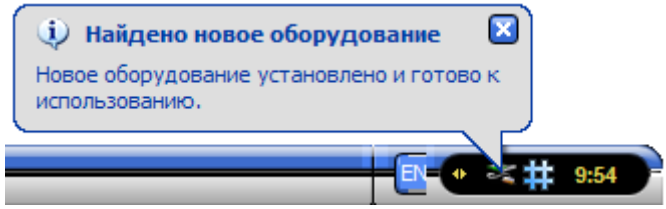


Начнется обновление базы данных установленных устройств в соответствии с INF файлом “C:\MCHPFSUSB\fw\Cdc\inf\win2k_winxp\mchpcdc.inf”, входящим в фирменное программное сопровождение платы “PICDEM FS USB”. И что важно, драйвер usbser.sys, на который ссылается файл mchpcdc.inf входит в состав

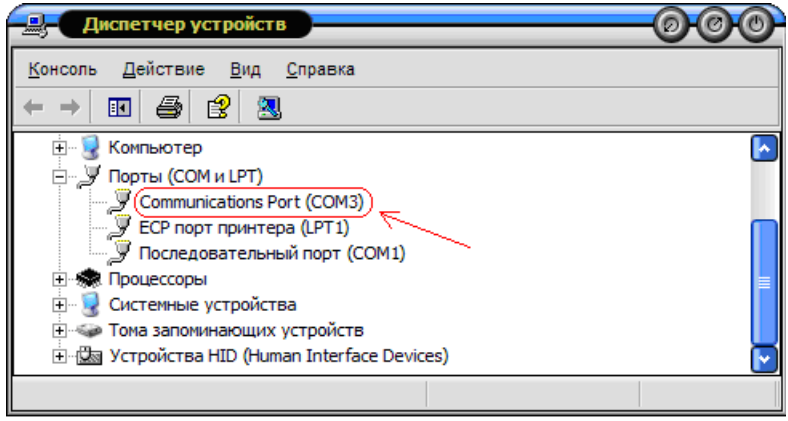
Windows. Поэтому разработчику ПО, то есть вам, не нужно писать собственный драйвер.
Если установка прошла успешно, появится завершающее диалоговое окно, а на панели задач



появится сообщение о готовности устройства.

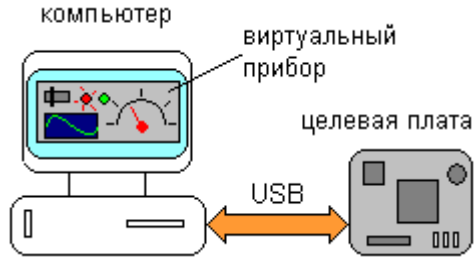


Теперь, если открыть окно диспетчера устройств панели управления, можно увидеть дополнительный виртуальный COM порт (в вашем случае номер порта может быть другим).



Обратите внимание: системные светодиоды D1, D2 на рабочей плате мигают ПООЧЕРЕДНО, что свидетельствует о готовности порта USB к работе.

РАЗРАБОТКА ВИРТУАЛЬНОГО ПРИБОРА (ВП) ДЛЯ ОТОБРАЖЕНИЯ ДАННЫХ И УПРАВЛЕНИЯ ЦЕЛЕВОЙ ПЛАТОЙ С ПОМОЩЬЮ USB ИНТЕРФЕЙСА



Разработку программы виртуального прибора можно производить с использованием любой подходящей среды программирования, например Microsoft Visual Studio, Delphi, Java Builder и т.п. Остановимся на Microsoft Visual Basic'е (в дальнейшем VB), входящим в состав Microsoft Visual Studio. VB позволяет быстро написать требуемую программу и легко импортировать сторонние библиотеки с набором нужных компонентов.

Создание формы графического интерфейса виртуального прибора

1). Запускаем программу “Microsoft Visual Basic 6.0” – VB6.exe

2). В появившемся диалоговом окне кликаем левой кнопкой по “Standart EXE”.

Если диалоговое окно не появилось, то из основного меню “File | New Project” также выбираем “Standart EXE”. На экране дисплея появится интегрированная среда разработки (IDE) с пустой формой, которую вы должны наполнить функциональным содержанием.

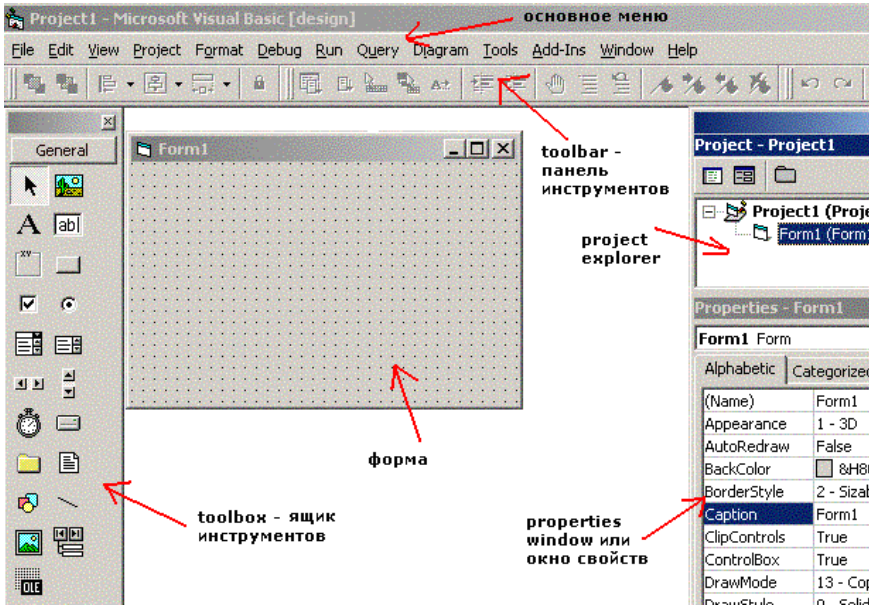


Рис.1

Сохраним файлы нашего проекта, который можно назвать, например “VirtPribor”, для чего выберем п. меню “File | Save Project” и в появившемся диалоговом (рис.2) окне перейдем в папку “C:\EMUL\Work\”, где правым кликом вызовем контекстное меню и создадим новую папку с номером учебной группы, например – gr4444. По-прежнему, не выходя из диалогового окна входим в эту папку и создаем новую с названием проекта, например VirtPribor.

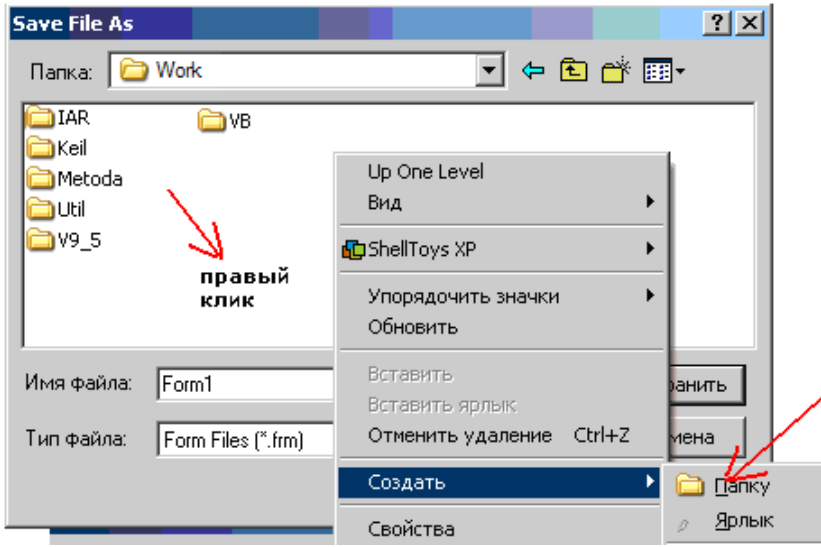
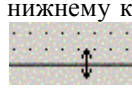
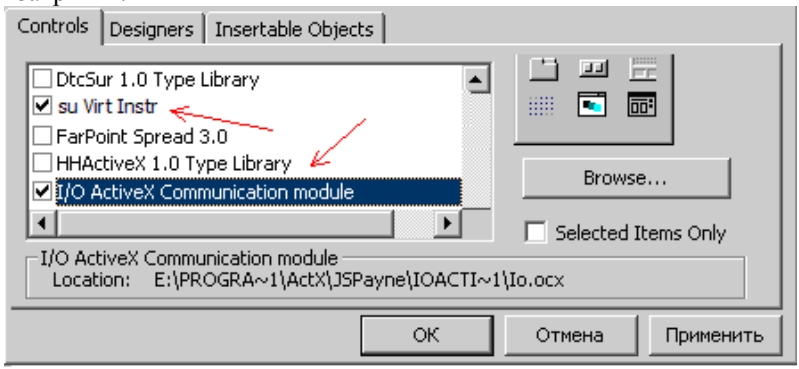


Рис.2

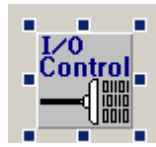
После чего переименуем название формы “Form1”, например на “ VirtPribor ” и нажмем кнопку “Сохранить” или “Save”. Сразу же появится второе диалоговое окно с просьбой сохранить проект. Не мудрствуя, вместо обезличенного “Project1” также подставим “ VirtPribor” и сохраним проект. На этом этапе в папке “C:\EMUL\Work\gr4444\VirtPribor\” должны храниться два файла: VirtPribor.vbp и VirtPribor.frm. Заодно (это можно и не делать) в окне “Project Explorer” (см. Рис.1) правым кликом по имени “Project1” перейдем к окну “Properties Window” (также см. Рис.1) и поменяем название “Project1” на какое-нибудь свое, например, опять “VirtPribor”.

3) Приступим к проектированию внешнего вида формы, т.е. лицевой панели виртуального прибора. Переход от окна “Code” т.е. текста программы к окну “Object” в нашем случае к форме производится с помощью двух кнопок  , расположенных сверху окна “Project Explorer” (см. Рис.1) или с помощью тех же кнопок из п. основного меню “View | Code” или “View | Object”. Сначала растянем форму до размера x=8865 и y=6375 условных единиц. Для этого подведем курсор к нижнему краю формы. Курсор приобретет форму двойной стрелки  . Зажав правую кнопку изменим вертикальный размер формы до нужного значения 6375. Аналогично изменим горизонтальный размер формы. В принципе значения ширина и высота формы могут быть и другие.

4) Теперь нам понадобится объект для работы с последовательным COM портом, который не входит в стандартный состав “ToolBox'a”. Библиотеки с такими объектами могут, либо входить в состав Windows, либо импортироваться от сторонних разработчиков. Воспользуемся одной из популярных библиотек для работы с COM, LPT и TCP – портами. Обычно для этого такие библиотеки необходимо стандартным образом инсталлировать, а затем импортировать их в проект. В нашем случае библиотека уже инсталлирована. Для импорта компонентов этой библиотеки кликнем ПРАВОЙ кнопкой по “ToolBox’у” и в появившемся окошке контекстного меню кликнем по п. “Components…”. Сразу же появится диалоговое окно. Отметим “птичкой” I/O ActiveX Communicatoin module и нажмем на кнопки “Применить” и “Закрыть”.

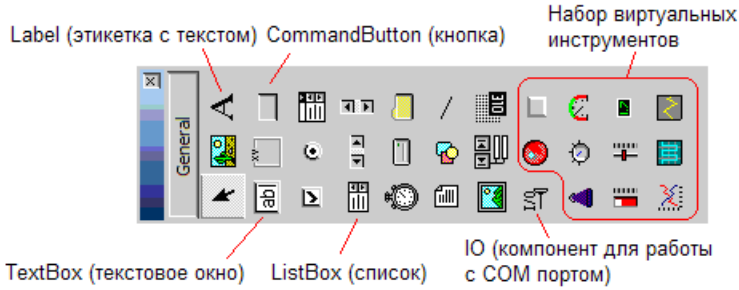


В этот момент на “ToolBox’е” (но пока не в нашем проекте) появится новый компонент  . Поместим этот компонент на форму, для чего кликнем по его пиктограмме и переведем курсор на форму (отметим, что в этот момент курсор приобрел форму креста). Затем нужно прижать левую кнопку, сделать легкое движение мышью и отпустить кнопку. В этот момент пиктограмма компонента для работы с портами ввода/вывода появится на форме.



Так как его иконка во время выполнения программы не видна, можно поместить ее в любом месте формы, например, внизу.

Аналогично установим набор компонентов с виртуальными инструментами, для чего отметим библиотеку “su Virt Instr”. В этот момент панель “ToolBox” будет иметь следующий состав:

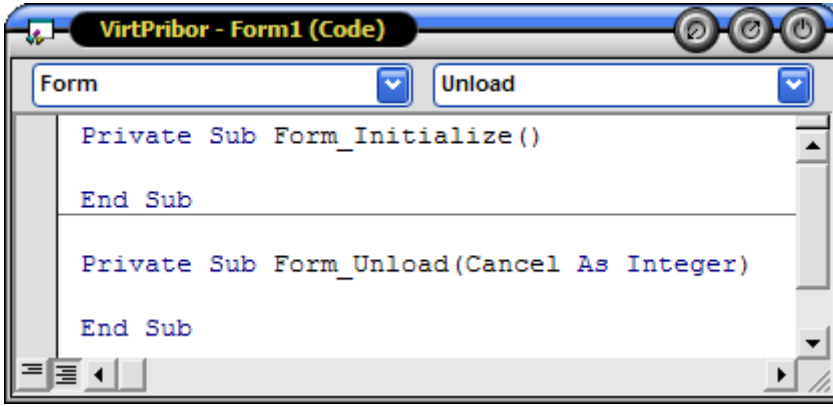


Для экономии места панель развернута горизонтально.

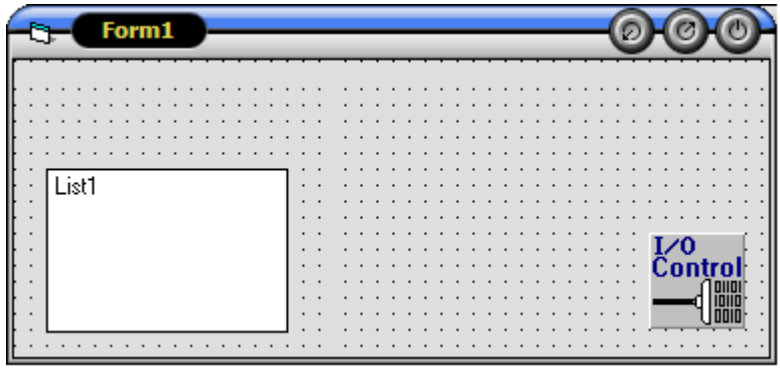
Разработка графического интерфейса виртуального прибора и разработка ПО для взаимодействия с рабочей платой.

5-1) Сначала в программе необходимо открыть канал связи с рабочей платой. Эти действия обычно производятся при запуске программы, во время ее инициализации в памяти компьютера. Создадим в программе обработчик этого события (Initialize), для чего перейдем к окну “Code”, кликнув по иконке (Project

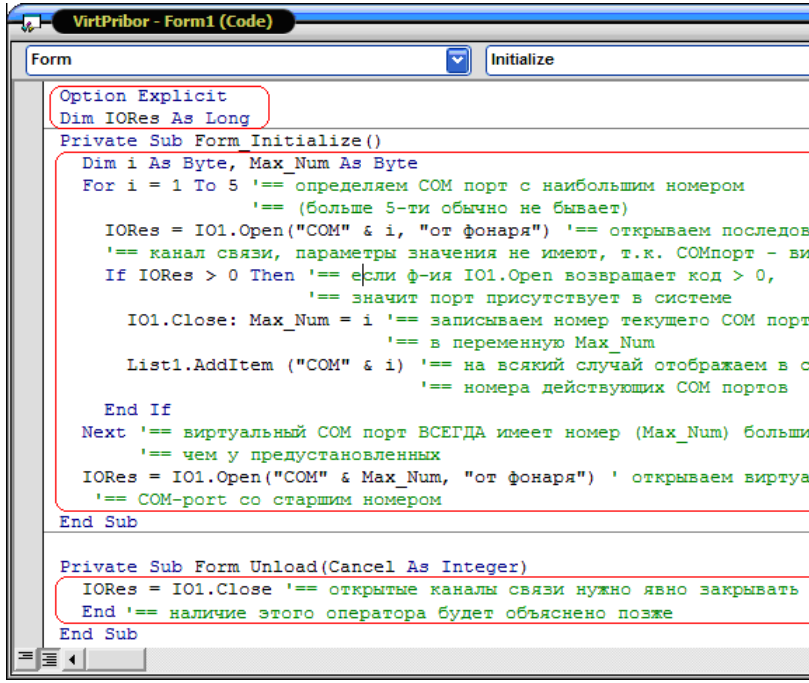
Explorer'a) или через п. меню View | Code. В верхней части этого окна из двух списков выберем Form и Initialize. Visual Basic автоматически создаст каркас процедуры соответствующего обработчика. Иногда VB создает по умолчанию обработчик событии Form_Load, но так как он в программе не нужен удалим его (можно этого и не делать так как VB удаляет пустые процедуры при первой же компиляции). Аналогичным образом выбираем из правого списка обработчик момента завершения программы (выгрузки формы из памяти – Unload). VB также автоматически создаст каркас процедуры Form_Unload().



Перед записью кода в обработчики событий, поместим на форму вспомогательный элемент – список (List), для чего сначала перейдем к окну с нашей формой, кликнув по значку Project Explorer'a или через п. меню View | Object и затем, кликнув по иконке "ListBox" на панели "ToolBox", перенесем список на форму. В этот момент форма будет иметь следующий вид (масштаб не соблюден):

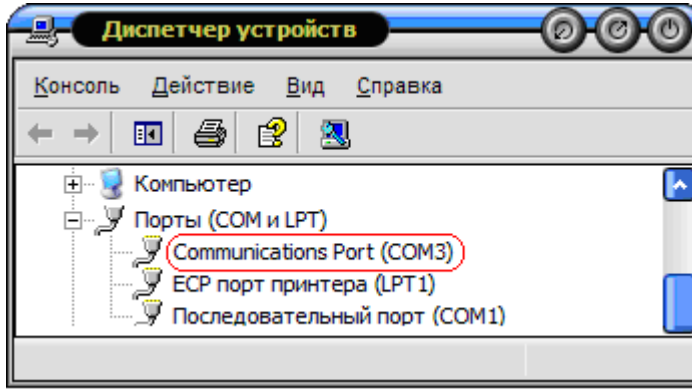


Теперь добавим в обработчики событий следующий код (на рисунке выделен рамкой) и заодно объявим вспомогательную переменную IORes (результат ввода/вывода библиотечных функций модуля IO):



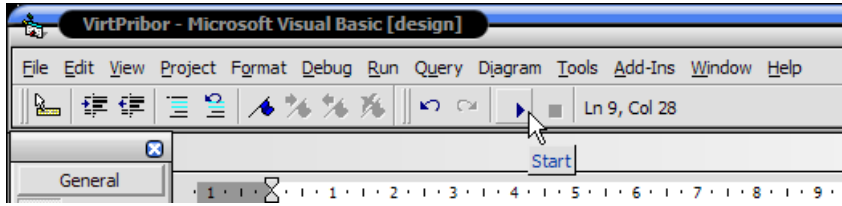
Оператор Option Explicit предотвращает появление в программе необъявленных переменных, и возникновение трудноуловимых

ошибок. Вспомогательная переменная IORes служит для анализа результатов операций с COM портом. Поиск номера виртуального порта можно и не производить, достаточно заглянуть в диспетчер устройств панели управления.

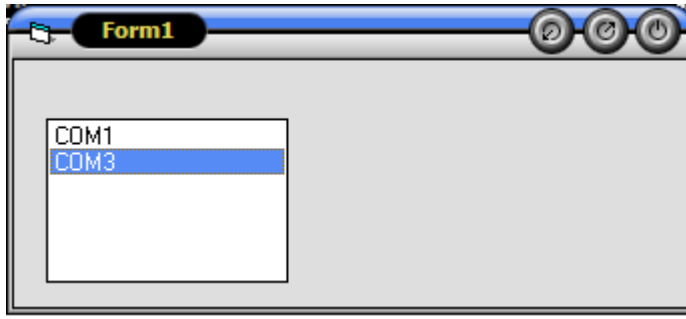


Однако, лучшим решением является автоматизация этого процесса, например с помощью циклического перебора установленных в системе портов: For i=1 To 10 ……. Next. Если функция IO1.Open() возвращает код больший “1”, то порт присутствует в системе. Функция IO1.Close закрывает временно открытый порт. Остается только зафиксировать максимальный номер присутствующего в системе порта – Max_Num, соответствующий виртуальному COM порту и снова открыть его, уже для работы (вспоминаем, что на самом деле будет открыт физический порт USB). Так как COM порт – виртуальный, то действительные его параметры не имеют значения и в строку настройки записываем, что угодно или оставляем строку пустой “ ”. В процедуре Form_Unload(), которая вызывается при выходе из программы по “Alt+F4” или по нажатию на кнопку  в правом верхнем углу окна программы, - закрываем открытый канал связи IO1.Close.

Произведите первый пробный запуск программы, для чего кликните по кнопке “Start”  на панели инструментов или нажмите “горячую клавишу” – “F5”.

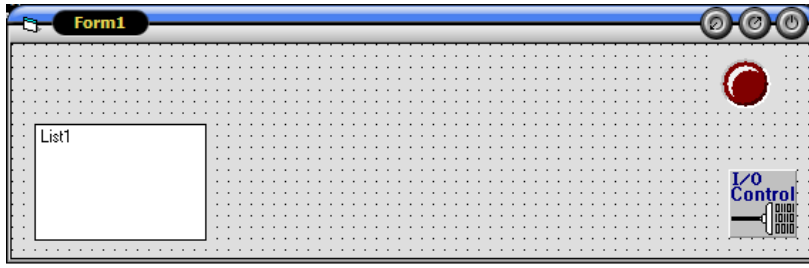


Если при записи кода в программу вы не “наделали” синтаксических и грамматических ошибок, то экране дисплея можно будет увидеть результат:

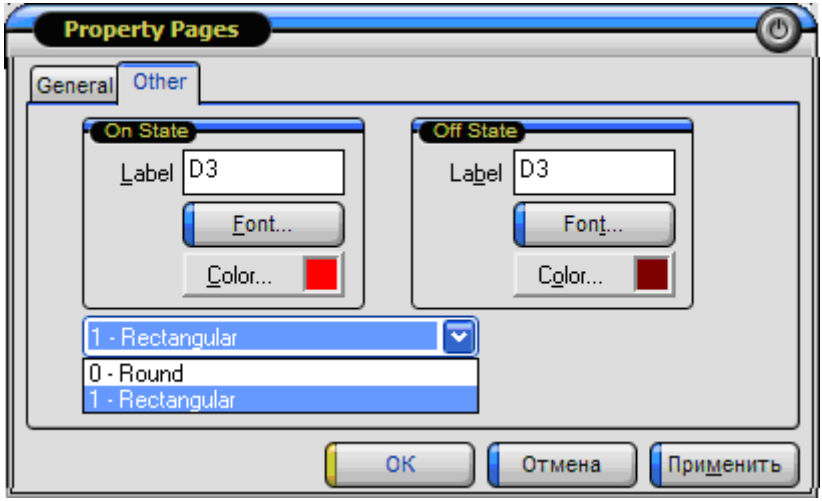


Как уже говорилось, число COM портов на разных компьютерах может быть иным. Остановим программу кнопкой  на панели инструментов Visual Basic’а или кликнув по  в верхнем правом углу окна.

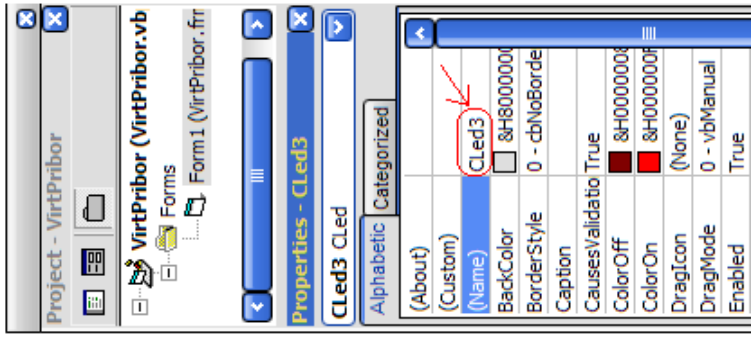
5-2) Снимем с панели ToolBox’а и поместим на форму первый регистрирующий (и одновременно управляющий) элемент – виртуальный светодиод  .



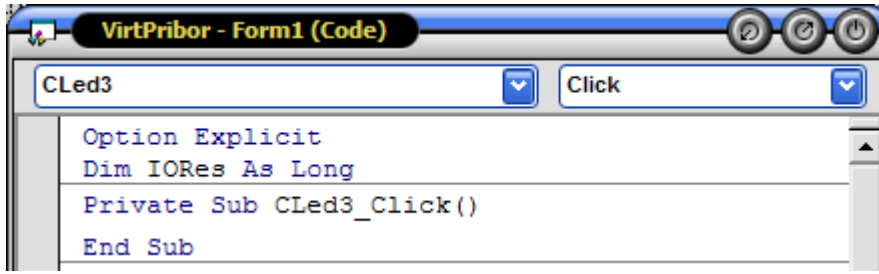
Придадим виртуальному светодиоду прямоугольную форму, свойственную командной кнопке, для чего кликнем по нему правой кнопкой и в появившемся контекстном меню выберем п. “Свойства…”. В открывшемся диалоговом окне отметим “Rectangular”, впишем метку “D3” и завершим операцию кнопкой “OK”.



Наш виртуальный светодиод предназначен для управления и отображения состояния реального светодиода №3, расположенного на рабочей плате, поэтому целесообразно свойство “Name” сменить с CLed1 на CLed3. Замену проще произвести в окне “Properties Window”, как показано на рисунке.



Теперь двойным кликом по светодиоду создадим обработчик события CLed3_Click().



Его положение в программе может отличаться от положения, приведенного на рисунке. Поместим в обработчик события дополнительный код, как показано на следующем фрагменте.

```
Option Explicit
Dim IORes As Long

Private Sub CLed3_Click() '== Обработчик события - "клик по вирт. свет
  If CLed3.Value = False Then '== если виртуальный светодиод D3 "не го
    CLed3.Value = True '== "зажигаем" виртуальный светодиод3 и
    IO1.WriteString ("1") '== посылаем на рабочую плату команду - 'заж
  Else '== иначе (если виртуальный светодиод D3 "горит") ....
    CLed3.Value = False '== "гасим" виртуальный светодиод3 и
    IO1.WriteString ("2") '== посылаем на рабочую плату команду - 'пог
  End If
End Sub
```

Возникает вопрос, откуда появились “волшебные” цифры “1” и “2”, которые включают и выключают светодиод? Проанализируем фрагмент программы, записанной во флэш-память микроконтроллера на рабочей плате: функция getsUSBUSART(input_buffer,1) принимает переданный символ IO1.WriteString(…), и если он равен ‘1’ – светодиод №3 на плате включается – mLED_3_on(), при двойке – выключается (mLED_3_off()).

```
void User_Process(void){
    if(getsUSBUSART(input_buffer,1)){
      switch (input_buffer[0]){
          case '1' : mLED_3_On(); break;
          case '2' : mLED_3_Off(); break;
          ..........................................................
```

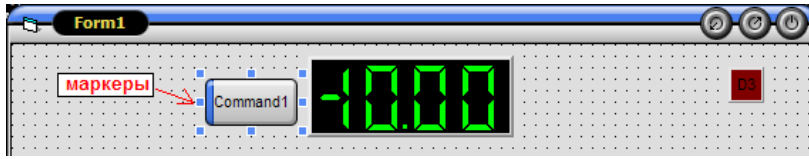
Снова запустим программу, кнопкой “Start”  на панели инструментов или “горячей клавишей” – “F5”. Кликая левой кнопкой убедитесь, что виртуальный и реальный светодиоды №3 включаются и выключаются синхронно (позовите преподавателя и покажите результат).

Результатом этого этапа является свидетельство работоспособности взаимодействия программы и рабочей платы с помощью интерфейса USB. В этот момент наша программа должна выглядеть следующим образом:

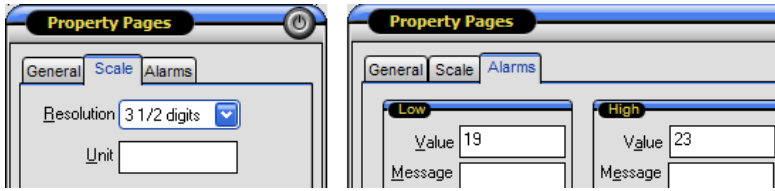
```
Option Explicit
Dim IORes As Long

Private Sub CLed3_Click() '== Обработчик события - "клик
по вирт. светодиоду3"
 If CLed3.Value = False Then '== если виртуальный
светодиод D3 "не горит", то
  CLed3.Value = True '== включаем виртуальный
светодиод3 и
  IO1.WriteString ("1") '== посылаем на рабочую плату
команду - включить светодиод D3
 Else '== иначе (если виртуальный светодиод D3 "горит") ....
  CLed3.Value = False '== выключаем виртуальный
светодиод3 и
  IO1.WriteString ("2") '== посылаем на рабочую плату
команду - 'погасить' светодиод D3
  IO1.WriteByte (Asc("2")) '== можно и так
 End If
End Sub

Private Sub Form_Initialize()
 Dim i As Byte, Max_Num As Byte
 For i = 1 To 5 '== определяем COM порт с наибольшим
номером
          '== (больше 5-ти обычно не бывает)
  IORes = IO1.Open("COM" & i, "от фонаря") '== открываем
последовательный
  '== канал связи, параметры значения не имеют, т.к.
COMпорт - виртуальный
  If IORes > 0 Then '== если ф-ия IO1.Open возвращает код
> 0,          '== значит порт присутствует в системе
```

```
            IO1.Close: Max_Num = i '== записываем номер текущего
COM порта
                                '== в переменную Max_Num
            List1.AddItem ("COM" & i) '== на всякий случай
отображаем в списке
                                '== номера действующих COM портов
          End If
        Next '== виртуальный COM порт ВСЕГДА имеет номер
(Max_Num) больший
          '== чем у предустановленных
        IORes = IO1.Open("COM" & Max_Num, "от фонаря") '
открываем виртуальный
          '== COM-port со старшим номером
      End Sub

      Private Sub Form_Unload(Cancel As Integer) ‘== завершение
программы
        IORes = IO1.Close '== открытые каналы связи нужно явно
закрывать
        End '== наличие этого оператора будет объяснено позже
      End Sub
```

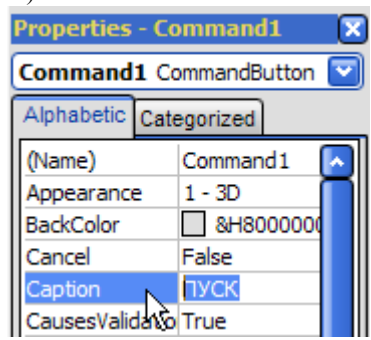
5-3) Добавим в наш виртуальный прибор – цифровой мультиметр (пиктограмма  на панели ToolBox) для отображения показаний датчика температуры (TC77 Microchip), расположенного на рабочей плате. Дополнительно поместим на лицевую панель командную кнопку (CommandButton в ToolBox’е), с помощью которой будем запускать процесс измерения и передачи результатов из рабочей платы в компьютер.



Правой кнопкой мыши вызовите контекстное меню цифрового мультиметра и выберите п. “Свойства…”. На закладке “Scale” очистите окно “Unit”, а на закладке “Alarms” выберите допустимый (или комфортный) на ваш взгляд диапазон температур. В дальнейшем, при выходе температуры из указанного диапазона можно будет предусмотреть какие-нибудь экстренные действия.

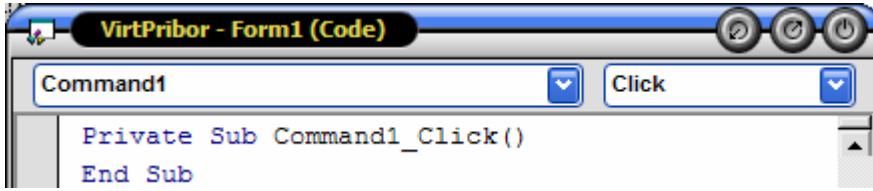


Свойство “Caption” кнопки “Command1” (т.е. текст, который будет на ней отображаться) измените на “ПУСК”.



ВНИМАНИЕ: 1) Свойство “Name” оставляем без изменения, т.е. “Command1”. 2) Здесь и далее, если вы меняете свойство элемента формы (в нашем случае командной кнопки “Command1”) он должен находиться в фокусе и по его периметру должны располагаться 8 квадратов – маркеров (см. рисунок). В противном случае вы рискуете сменить такое же свойство, НО У ДРУГОГО ЭЛЕМЕНТА.

5-3-1) Создадим обработчик события при нажатии на кнопку “ПУСК”, для чего дважды кликнем по ней.



Добавим в программу, выделенные на рисунке фрагменты кода.

```
Private Sub Form_Initialize()
 ..........................................................
 IORes = IO1.Open("COM" & Max_Num, "от фонаря") ' открываем виртуальный
  '== COM-port со старшим номером (на самом деле открываем USB порт)
 IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП" прекращения
                          '== считывания данных на целевой плате
 CDVM1.Value = 0 '== начальное значение цифрового мультиметра
End Sub

Private Sub Command1_Click()
  Dim InBuf(4) As Byte, temperature As Single, IOStatus As Long
  Dim Tempr_H As Long, Tempr_L As Long '== старший и младший байты темпер
  Const SERIAL_RXEMPTY = &H40 '== маска для бита "входной буфер COM порта
  IO1.WriteByte (Asc("P")) '== посылаем команду "ПУСК" - считывания
                           '== данных на целевой плате
  Do '== начало "бесконечного цикла" ожидания данных от рабочей платы
   DoEvents '== необходимо для обработки других событий
   IOStatus = IO1.SerialStatus '== проверка статуса вирт. COM порта
   If ((IOStatus And SERIAL_RXEMPTY) > 0) Then '== если рабочая плата не
    GoTo break '== данные (буфер пуст - EMPTY), то выйти на следующий цик
   Else '== иначе, если во входном буфере есть принятые данные
    DoEvents
    IORes = IO1.ReadBytes(InBuf, 4) '== то записать эти данные в массив I
    Tempr_H = InBuf(2): Tempr_L = InBuf(3) '== старший и младший байты t-
    temperature = ((Tempr_H * 256 + Tempr_L) And &HFFF8)/128'== поправочный
    CDVM1.Value = temperature '== отображаем t-ру на цифровом мультиметре
   End If
break:
  Loop '== повтор "бесконечного цикла"
End Sub
```

В обработчике события “инициализация формы” предусмотрим остановку считывания данных на целевой плате, для чего в нее необходимо переслать команду “СТОП” (например символ “S”): IO1.WriteByte(Asc(“S”)). Здесь же присваиваем нулевое начальное значение цифровому мультиметру.

В момент нажатия на кнопку “ПУСК” (Command1) стартует обработчик события Command1_Click(). Маска для бита “входной буфер для COM порта – пуст” равный 0x40 взят из описания библиотечного файла IO.OCX. Посылаем на рабочую плату команду “ПУСК” для запуска измерений температуры: IO1.WriteByte(Asc(“P”)). В соответствии с нашими командами “СТОП” и “ПУСК” необходимо изменить код пользовательской программы USER.C, записанной во флэш-память микроконтроллера на рабочей плате (на самом деле ИЗМЕНЕНИЯ УЖЕ СДЕЛАНЫ). Модификация производится в двух подпрограммах: ProcessIO() и UserProcess(). Как видно из приведенного фрагмента, 4-ре байта данных (два байта для температуры и два байта для напряжения) считываются и передаются в компьютер только по команде “ПУСК” (строка mUSBUSARTTxRam((byte*)output_buffer,4) подпрограммы

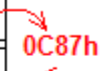
UserProcess()). Считывание кода напряжения будет произведено далее.

```
void main(void){
     InitializeSystem();
     while(1){
          USBTasks();
          ProcessIO();
     }//end while
}//end main
void ProcessIO(void){
   BlinkUSBStatus();
   counter++;
   if (counter==20000){
        counter=0;
     if(pusk == 'P'){//== Pusk
        AcquireTemperature();
        output_buffer[3] = LSB(temperature);
        output_buffer[2] = MSB(temperature);
.....................................................................................
        User_Process();
      }
 }//end ProcessIO

void User Process(void){
  if(getsUSBUSART(input_buffer,1)){
    switch (input_buffer[0]){
         case '1' : mLED_3_On(); break;
         case '2' : mLED_3_Off(); break;
         case 'P' : pusk='P'; break;//== ПУСК
         case 'S' : pusk='S'; break;//== СТОП
  }      default   : break; }
  if(pusk == 'P'){//==  ПУСК
     if(mUSBUSARTIsTxTrfReady()) {
       mUSBUSARTTxRam((byte*)output_buffer,4);}
  }//== end if pusk
}//end
```

Вернемся к обработчику события Command1_Click(). Оператор DoEvents в “бесконечных циклах” нужен для реагирования на внешние события Events, в противном случае может понадобиться “Ctrl+Alt+Del”. Остальные строчки достаточно прокомментированы. Поправочный делитель 128 в строке программы нетрудно рассчитать, используя справочные данные для термодатчика TC77:

```
temperature = ((Tempr_H * 256 + Tempr_L) And &HFFF8)/128
```

Temperature	Binary MSB / LSB	Hex
+125°C	0011 1110 1000 0111	3E 87h
+25°C	0000 1100 1000 0111	0B 87h
+0.0625°C	0000 0000 0000 1111	00 0Fh
0°C	0000 0000 0000 0111	00 07h
-0.0625°C	1111 1111 1111 1111	FF FFh
-25°C	1111 0011 1000 0111	F3 87h
-55°C	1110 0100 1000 0111	E4 87h



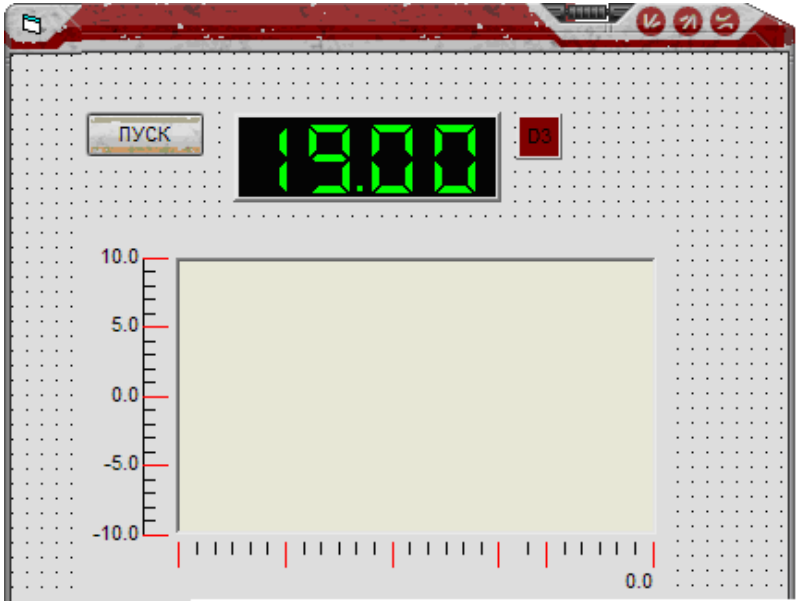
Для начала отметим, что вместо ошибочного значения 0B87h должно быть 0C87h. Из справочных данных TC77 следует, что для кодирования температуры (“дополнительный код”) используются только старшие 13 бит, а три младших бита – служебные (которые перед преобразованием кода в температуру необходимо отбросить или обнулить с помощью маски FFF8). В соответствии с этим составим табличку, из которой видно, что для получения ПОЛОЖИТЕЛЬНОЙ температуры в градусах Цельсия необходимо разделить код с обнуленными тремя младшими битами на 128.

+125	3E87	3E80 (16000)	16000/128=125
+25	0C87	C80(3200)	3200/128=25
+0.0625	000F	8	8/128=0.0625
0	0007	0	

Снова запустите программу, кнопкой “Start” ▶ на панели инструментов или “горячей клавишей” – “F5”. Убедитесь, что

цифровой прибор показывает температуру на поверхности рабочей платы. Результат покажите преподавателю.

5-4) Добавим к нашему виртуальному прибору ленточный самописец (CStripChart), на котором будем регистрировать температуру и напряжение в динамике. Для этого на панели ToolBox кликнем по пиктограмме  и, прижав левую кнопку на свободном поле нашей формы, растянем прямоугольник с экраном самописца до нужных размеров.



Займемся настройкой внешнего вида самописца. По техническому заданию необходимо регистрировать два процесса – изменение температуры и напряжения (угла поворота). Поэтому необходимо добавить еще одну шкалу, например, справа от экрана.

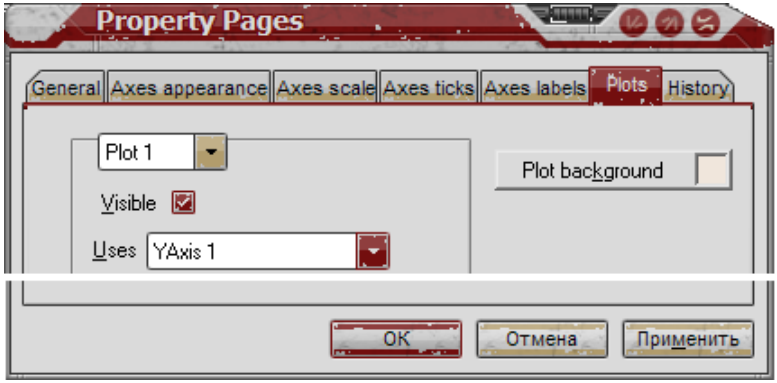
Эту шкалу будем использовать для отсчета температуры в диапазоне 15..29 °C. Левая шкала показывает значение напряжения в диапазоне -1..6 вольт.

Кликните правой кнопкой по полю экрана и в появившемся контекстном меню перейдите к п. “Свойства/Properties”. В открывшемся окне на странице “Plots - Графики” установите свойство “Visible” графика “Plot 0” в состояние “True”, т.е. сделайте этот график видимым. Затем, установите в выпадающем списке следующий график “Plot 1” и также сделайте его видимым. ОБЯЗАТЕЛЬНО привяжите график1 к оси Y1

, иначе график1 будет не виден. Отметьте, что по умолчанию цвет графика0 – красный, а графика1 – синий. Зафиксируйте изменения, нажав на кнопку “Применить”. Этого же эффекта можно достигнуть, записав в подпрограмму “обработчик события” Form_initialize() следующий код:

```
CStripChart1.Plot(0).Visible = True
CStripChart1.Plot(1).Visible = True
```

Перейдите к закладке “Axes appearance” и также отметьте “птичкой” свойства“Visible” для трех осей: X, Y0 и Y1. Для оси Y1 в свойстве “Position” отметьте правое расположение.

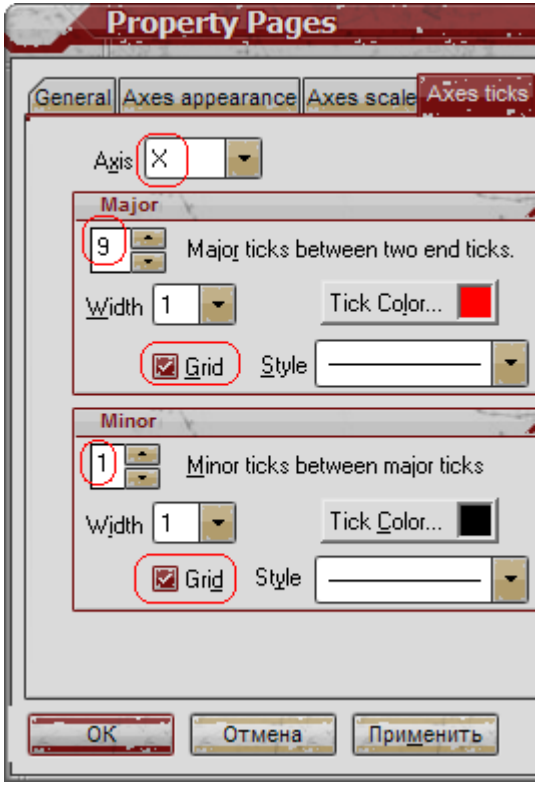


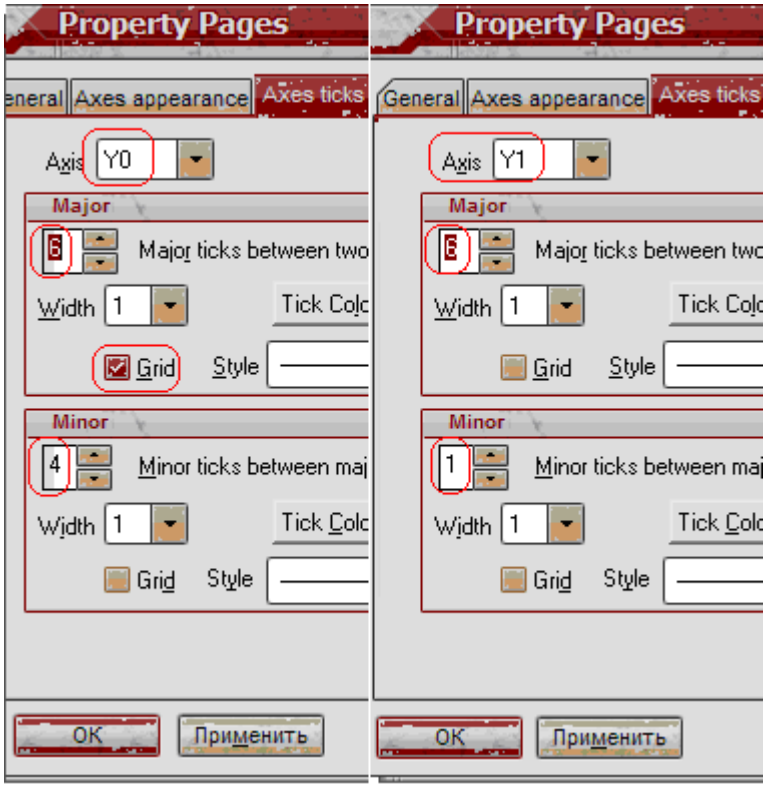
Затем перейдите к закладке “Axes scale” и установите для осей следующие свойства:

1) “Axis X” – максим. значение “Max 10”
2) “Axis Y0” – значения “Max 6” и
   “Min -1”, а также свойства “Low Alarm 0” и “High Alarm 4”

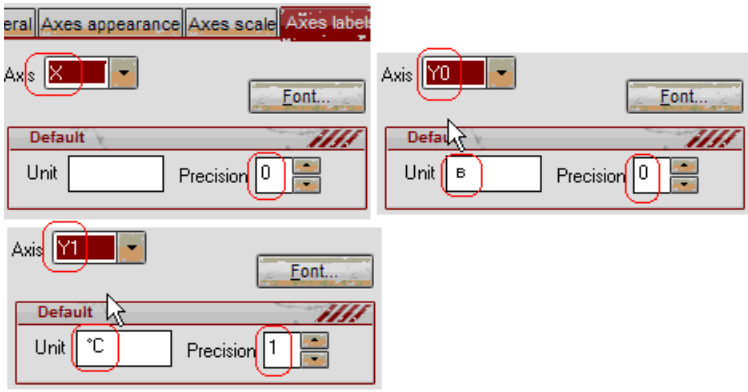
3) “Axis Y1” – значения “Max 29” и “Min 15”, а также свойства “Low Alarm 19” и “High Alarm 23”. Заодно подберите цвета “Alarm Colors”. 4) Не забудьте зафиксировать изменения кнопкой “Применить”.

Теперь перейдите к закладке “Axes ticks” и установите следующие значения:

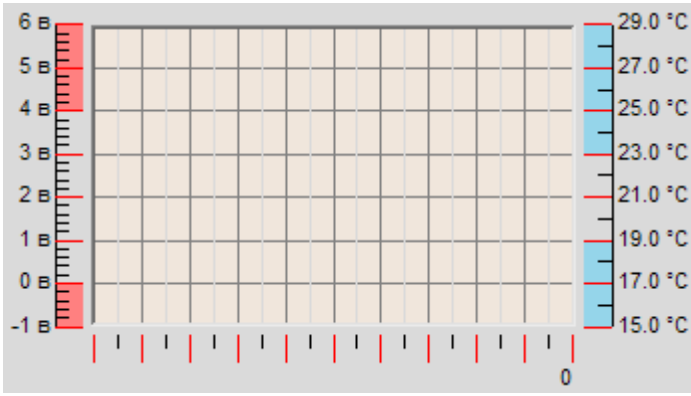




Последним штрихом, на закладке “Axes labels” установите следующие значения:



Не забудьте установить “Font” кириллическим. ASCII код “градуса” – B0(HEX) = 176(DEC) можно ввести из таблицы символов Windows. Окончательно, рабочее окно самописца будет выглядеть примерно следующим образом.



5-5) Поместим на форму вторую командную кнопку “СТОП”, которая будет посылать микроконтроллеру команду прекращения измерений и передачи данных в компьютер.

5-6) Теперь добавим в программу дополнительный код для отображения двух графиков – температуры и напряжения. Новые фрагменты обведены рамками.

```
Option Explicit
Dim IORes As Long
Dim Pusk As Boolean, FirstPusk As Boolean
Dim NumSamples As Long, IOStatus As Long, LowAlarm As Single
.....
Private Sub Form_Initialize()
 Dim i As Byte, Max_Num As Byte
 Pusk = False
 For i = 1 To 5 '== определяем COM порт с наибольшим номером
 .....
```

```
Private Sub Command2_Click() '== Обработчик события -
  IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП"
  Pusk = False '==  данных на целевой плате и передачи
End Sub              их в компьютер
  .....
Private Sub Command1_Click()
 Dim InBuf(4) As Byte, temperature As Single, IOStatus As Lor
 Dim Tempr_H As Long, Tempr_L As Long '== старший и младший б
 Const SERIAL_RXEMPTY = &H40 '== маска для бита "входной буфе
 Dim volt As Single, tt As Integer
 Dim ADC_H As Long, ADC_L As Long, tempr As Long
 Pusk = True
 IO1.WriteByte (Asc("P")) '== посылаем команду "ПУСК" - счить
  .....
  Tempr_H = InBuf(2): Tempr_L = InBuf(3) '== старший и младш
  ADC_H = InBuf(0): ADC_L = InBuf(1) '== старший и младший б
  If Pusk = False Then Exit Sub
  temperature = ((Tempr_H * 256 + Tempr_L) And &HFFF8) / 128
  CDVM1.Value = temperature '== отображаем t-ру на цифровом
  volt = (ADC_H / 256) * 5 '== 8-ми битн. АЦП (с выравн. влево
  If FirstPusk = True Then '== первое измерение отбросим
    tt = CStripChart1.UpdatePlot(0, 1, volt)
    tt = CStripChart1.UpdatePlot(1, 1, temperature)
    CStripChart1.DrawPlots
  End If
  FirstPusk = True
 End If
break:
```

Комментарии достаточны для пояснения программного кода. Добавление новых данных в буферную память графиков производится с помощью вызова функций CStripChart1.UpdatePlot(), параметрами которых являются номера графиков (0 или 1), количество добавляемых точек (одна) и имя отображаемой переменной (volt или temperature). Обновление графиков на экране производится процедурой CStripChart1.DrawPlots.

В этот момент программа должна выглядеть следующим образом:

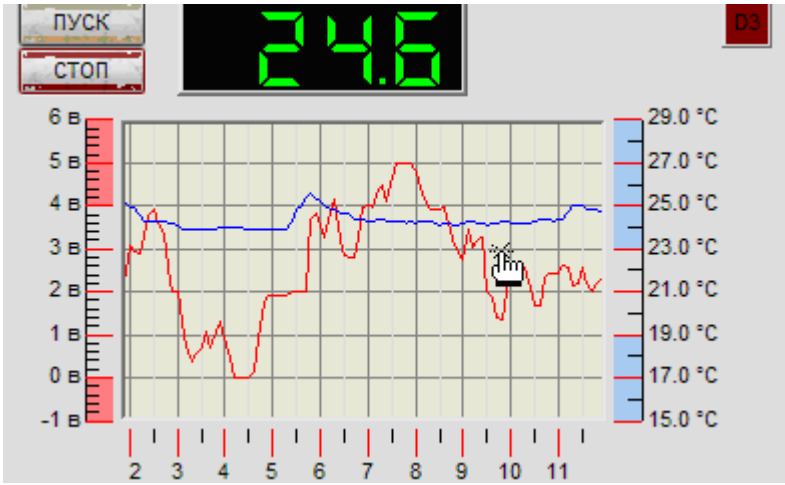
```
Option Explicit
Dim IORes As Long
Dim Pusk As Boolean, FirstPusk As Boolean
Dim NumSamples As Long, IOStatus As Long, LowAlarm As Single
```

```
Private Sub CLed3_Click() '== Обработчик события - "клик по вирт.
светодиоду3"
 If CLed3.Value = False Then '== если вирт-ый светодиод D3
выключен, то
  CLed3.Value = True '== включаем виртуальный светодиод3 и
  IO1.WriteString ("1") '== посылаем на рабочую плату команду -
'зажечь' св-диод D3
 Else '== иначе (если виртуальный светодиод D3 включен) ....
  CLed3.Value = False '== выключаем виртуальный светодиод3 и
  IO1.WriteString ("2") '== посылаем на рабочую плату команду -
'погасить' св-диод D3
 End If
End Sub

Private Sub Command2_Click() '== Обработчик события - нажата
кнопка "СТОП"
  IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП"
прекращения считывания
  Pusk = False '==  данных на целевой плате и передачи их в
компьютер
End Sub

Private Sub Form_Initialize()
 Dim i As Byte, Max_Num As Byte
 Pusk = False
 For i = 1 To 5 '== определяем COM порт с наибольшим номером
              '== (больше 5-ти обычно не бывает)
  IORes = IO1.Open("COM" & i, "от фонаря") '== открываем
последовательный
  '== канал связи, параметры значения не имеют, т.к. COMпорт -
виртуальный
  If IORes > 0 Then '== если ф-ия IO1.Open возвращает код > 0,
                '== значит порт присутствует в системе
   IO1.Close: Max_Num = i '== записываем номер текущего COM
порта
                      '== в переменную Max_Num
   List1.AddItem ("COM" & i) '== на всякий случай отображаем в
списке
                        '== номера действующих COM портов
  End If
 Next '== виртуальный COM порт ВСЕГДА имеет номер (Max_Num)
больший
```

```
    '== чем у предустановленных
 IORes = IO1.Open("COM" & Max_Num, "от фонаря") ' открываем
виртуальный
  '== COM-port со старшим номером (на самом деле открываем USB
порт)
 IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП"
прекращения
                    '== считывания данных на целевой плате
 CDVM1.Value = 0 '== начальное значение цифрового мультиметра
End Sub

Private Sub Command1_Click()
 Dim InBuf(4) As Byte, temperature As Single, IOStatus As Long
 Dim Tempr_H As Long, Tempr_L As Long '== старший и младший
байты температуры
 Const SERIAL_RXEMPTY = &H40 '== маска для бита "входной
буфер COM порта - пуст"
 Dim volt As Single, tt As Integer
 Dim ADC_H As Long, ADC_L As Long, tempr As Long
 Pusk = True
 IO1.WriteByte (Asc("P")) '== посылаем команду "ПУСК" –
считывания данных на целевой плате
 Do '== начало "бесконечного цикла" ожидания данных от рабочей
платы
  DoEvents '== необходимо для обработки других событий
  IOStatus = IO1.SerialStatus '== проверка статуса вирт. COM порта
  If ((IOStatus And SERIAL_RXEMPTY) > 0) Then '== если рабочая
плата не передает
   GoTo break '== данные (буфер пуст - EMPTY), то выйти на
следующий цикл ожидания
  Else '== иначе, если во входном буфере есть принятые данные
   DoEvents
   IORes = IO1.ReadBytes(InBuf, 4) '== то записать эти данные в
массив InBuf
   Tempr_H = InBuf(2): Tempr_L = InBuf(3) '== старший и младший
байты t-ры
   ADC_H = InBuf(0): ADC_L = InBuf(1) '== старший и младший
байты напряжения
   If Pusk = False Then Exit Sub ‘== если нажата кнопка “СТОП”, то
новые данные не показывать
   temperature = ((Tempr_H * 256 + Tempr_L) And &HFFF8) / 128 '==
поправочный делитель
```

```
   CDVM1.Value = temperature '== отображаем t-ру на цифровом
мультиметре
   volt = (ADC_H / 256) * 5 '== 8-ми битный АЦП (с выравниванием
влево и  отбрасыванием мл. байта)
   If FirstPusk = True Then '== первое измерение отбросим
     tt = CStripChart1.UpdatePlot(0, 1, volt) '== добавим новое значение
напряжения
     tt = CStripChart1.UpdatePlot(1, 1, temperature) '== добавим новое
значение напряжения
     CStripChart1.DrawPlots '== обновление графиков на экране
виртуального самописца
   End If
   FirstPusk = True
  End If
break:
 Loop '== повтор "бесконечного цикла"
End Sub
Private Sub Form_Unload(Cancel As Integer)
  IORes = IO1.Close '== открытые каналы связи нужно закрывать
  End '== выход в Windows
End Sub
```

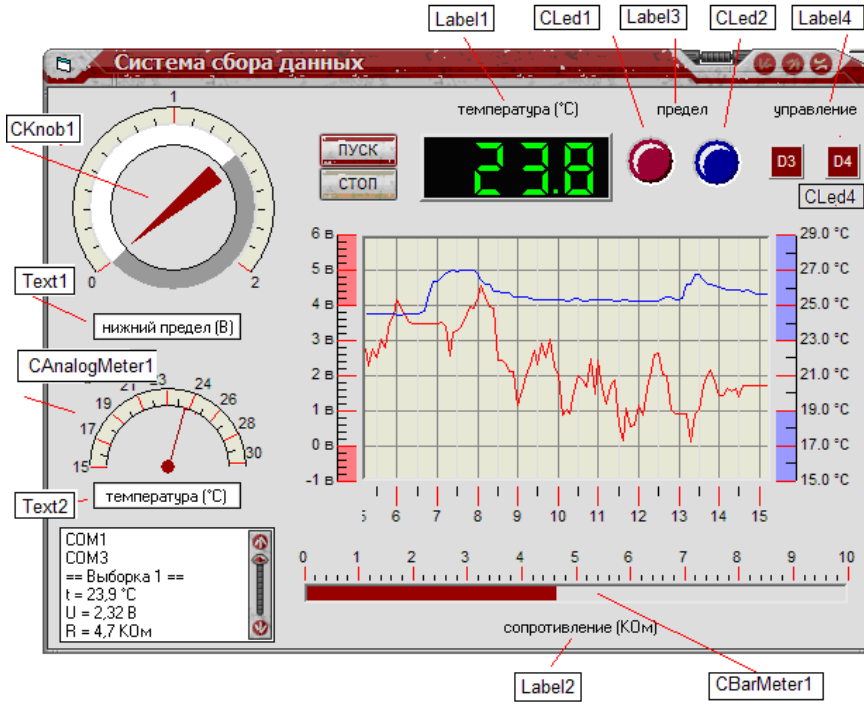
Снова запустимте программу, кнопкой “Start” на панели инструментов или “горячей клавишей” – “F5”. Убедитесь, что виртуальный ленточный самописец отображает графики температуры и напряжения. Результат покажите преподавателю.



5-7) Придадим форме окончательный вид, для чего разместим на ней дополнительные элементы регистрации и управления:

- CKnob1 – задает нижний допустимый предел датчика напряжения (о выходе за этот предел должен сигнализировать светодиод CLed1),
- CLed1 – см. выше,
- Text1, Text2 – поясняющие надписи на передней панели виртуального прибора,
- CAnalogMeter1 – аналоговый стрелочный измерительный прибор (температура),
- CBarMeter1 – линейный аналоговый индикатор (сопротивление),
- CLed2 – светодиод индикации выхода температуры за допустимый предел,
- CLed4 – светодиод, показывающий прохождение некоторой команды на рабочую плату,
- Label1..Label4 - поясняющие надписи на передней панели виртуального прибора.

Расположите указанные элементы и подписи к ним в соответствии с рисунком, а также добавьте в программу отмеченные дополнения.



Добавьте в программу отмеченные дополнения.

```
Private Sub Command1_Click()
 Dim InBuf(4) As Byte, temperature As Single, IOStatus As Long
   ..........................................................................
    volt = (ADC_H / 256) * 5 '== 8-ми битный АЦП (с выравниванием влево и
                             '== отбрасыванием мл. байта)
  If (temperature > 23#) Or (temperature < 19#) Then
    CLed2.Value = True '== если выход за предел - включить светодиод
    Else: CLed2.Value = False '== иначе погасить
  End If
  If (volt > 4#) Or (volt < LowAlarm) Then
    CLed1.Value = True '== если выход за предел - включить светодиод
    Else: CLed1.Value = False '== иначе погасить
  End If
  CAnalogMeter1.Pointer(0).Value = temperature '== вывести t-ру на аналог
  CBarMeter1.Pointer(0).Value = (ADC_H / 255) * 10 '//=== сопротивление
  If FirstPusk = True Then '== первое измерение отбросим
    tt = CStripChart1.UpdatePlot(0, 1, volt) '== обновить данные
```

Запустите программу и результат покажите преподавателю.

Окончательный текст программы

```
Option Explicit
Dim IORes As Long
Dim Pusk As Boolean, FirstPusk As Boolean
Dim NumSamples As Long, IOStatus As Long, LowAlarm As Single

Private Sub CLed3_Click() '== Обработчик события - "клик по вирт.
светодиоду3"
 If CLed3.Value = False Then '== если вирт-ый светодиод D3
выключен, то
  CLed3.Value = True '== включаем виртуальный светодиод3 и
  IO1.WriteString ("1") '== посылаем на рабочую плату команду -
'зажечь' св-диод D3
 Else '== иначе (если виртуальный светодиод D3 включен) ....
  CLed3.Value = False '== выключаем виртуальный светодиод3 и
  IO1.WriteString ("2") '== посылаем на рабочую плату команду -
'погасить' св-диод D3
 End If
End Sub

Private Sub CLed4_Click() '== Обработчик события - "клик по вирт.
светодиоду4"
  If CLed4.Value = False Then
    CLed4.Value = True
    IO1.WriteString ("3")
  Else
    CLed4.Value = False
    IO1.WriteString ("4")
  End If
End Sub

Private Sub CKnob1_KnobRotate(fValue As Single) '== Обработчик
события -
                                    '== "Ручка - Knob" повернута
  LowAlarm = fValue '== устанавливаем новое значение нижнего
порогового значения
  CStripChart1.LowAlarm = fValue '== только для одной оси -(Y0)
End Sub
```

```
Private Sub Command2_Click() '== Обработчик события - нажата
кнопка "СТОП"
  IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП"
прекращения считывания
  Pusk = False '==  данных на целевой плате и передачи их в
компьютер
End Sub

Private Sub Form_Initialize()
 Dim i As Byte, Max_Num As Byte
 Pusk = False: FirstPusk = False
 For i = 1 To 5 '== определяем COM порт с наибольшим номером
              '== (больше 5-ти обычно не бывает)
  IORes = IO1.Open("COM" & i, "от фонаря") '== открываем
последовательный
  '== канал связи, параметры значения не имеют, т.к. COMпорт -
виртуальный
  If IORes > 0 Then '== если ф-ия IO1.Open возвращает код > 0,
                '== значит порт присутствует в системе
   IO1.Close: Max_Num = i '== записываем номер текущего COM
порта
                       '== в переменную Max_Num
   List1.AddItem ("COM" & i) '== на всякий случай отображаем в
списке
                         '== номера действующих COM портов
  End If
 Next '== виртуальный COM порт ВСЕГДА имеет номер (Max_Num)
больший
      '== чем у предустановленных
 IORes = IO1.Open("COM" & Max_Num, "от фонаря") ' открываем
виртуальный
  '== COM-port со старшим номером (на самом деле открываем USB
порт)
 IO1.WriteByte (Asc("S")) '== посылаем команду "СТОП"
прекращения
                     '== считывания данных на целевой плате
 CDVM1.Value = 0 '== начальное значение цифрового мультиметра
End Sub

Private Sub Command1_Click()
 Dim InBuf(4) As Byte, temperature As Single, IOStatus As Long
```

```
 Dim Tempr_H As Long, Tempr_L As Long '== старший и младший
байты температуры
 Const SERIAL_RXEMPTY = &H40 '== маска для бита "входной
буфер COM порта - пуст"
 Dim volt As Single, tt As Integer
 Dim ADC_H As Long, ADC_L As Long, tempr As Long
 Pusk = True
 IO1.WriteByte (Asc("P")) '== посылаем команду "ПУСК" -
считывания
                    '== данных на целевой плате
 Do '== начало "бесконечного цикла" ожидания данных от рабочей
платы
  DoEvents '== необходимо для обработки других событий
  IOStatus = IO1.SerialStatus '== проверка статуса вирт. COM порта
  If ((IOStatus And SERIAL_RXEMPTY) > 0) Then '== если рабочая
плата не передает
   GoTo break '== данные (буфер пуст - EMPTY), то выйти на
следующий цикл ожидания
  Else '== иначе, если во входном буфере есть принятые данные
   DoEvents
   IORes = IO1.ReadBytes(InBuf, 4) '== то записать эти данные в
массив InBuf
   Tempr_H = InBuf(2): Tempr_L = InBuf(3) '== старший и младший
байты t-ры
   ADC_H = InBuf(0): ADC_L = InBuf(1) '== старший и младший
байты напряжения
   If Pusk = False Then Exit Sub
   temperature = (Tempr_H * 256 + (Tempr_L And &HF8)) / 128 '==
поправочный множитель
   CDVM1.Value = temperature '== отображаем t-ру на цифровом
мультиметре
   volt = (ADC_H / 256) * 5 '== 8-ми битный АЦП (с выравниванием
влево и
                    '== отбрасыванием мл. байта)
   If (temperature > 23#) Or (temperature < 19#) Then
    CLed2.Value = True '== если выход за предел - включить
светодиод
    Else: CLed2.Value = False '== иначе погасить
   End If
   If (volt > 4#) Or (volt < LowAlarm) Then
    CLed1.Value = True '== если выход за предел - включить
светодиод
```

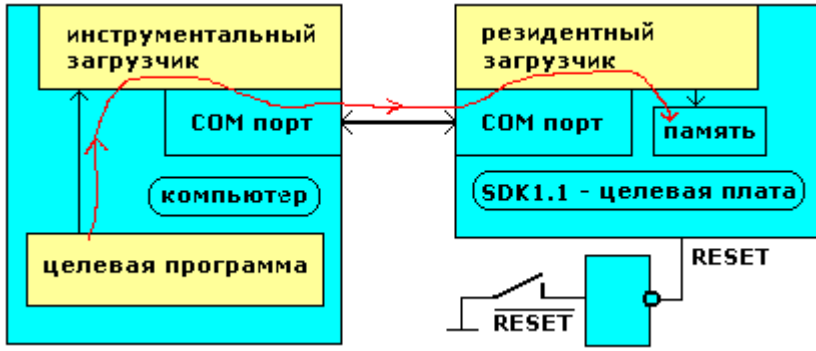
```
    Else: CLed1.Value = False '== иначе погасить
   End If
   CAnalogMeter1.Pointer(0).Value = temperature '== вывести t-ру на
аналог. прибор
   CBarMeter1.Pointer(0).Value = (ADC_H / 255) * 10 '//===
сопротивление (0..10 КОм)
   If FirstPusk = True Then '== первое измерение отбросим
    tt = CStripChart1.UpdatePlot(0, 1, volt) '== обновить данные
    tt = CStripChart1.UpdatePlot(1, 1, temperature) '== обновить данные
    CStripChart1.DrawPlots '== отобразить обновленные графики
    NumSamples = NumSamples + 1
    List1.AddItem ("== Выборка " & NumSamples & " ==")
    List1.AddItem ("t = " & Format(temperature, "##.0") & " °C")
    List1.AddItem ("U = " & Format(volt, "0.00") & " В")
    List1.AddItem ("R = " & Format((ADC_H / 255) * 10, "0.0") & "
КОм")
   End If
   FirstPusk = True
  End If
break:
 Loop '== повтор "бесконечного цикла"
End Sub

Private Sub Form_Unload(Cancel As Integer)
 IORes = IO1.Close '== открытые каналы связи нужно явно закрывать
 End '== (выход в Windows) наличие этого оператора будет
объяснено позже
End Sub
```

ВОПРОСЫ ДЛЯ ЗАЩИТЫ И ЭКЗАМЕНА

1. Сравнение COM и USB интерфейсов.
2. Работа внешнего устройства (целевой платы) в классе Communication Device Class – CDC.
3. Какие драйверы windows нужны для класса устройств - CDC?
4. Основные функции для работы с COM и USB портом.
5. Уметь комментировать каждую строчку программы.

ЛАБОРАТОРНАЯ РАБОТА №10
РАЗРАБОТКА ИНСТРУМЕНТАЛЬНОГО ЗАГРУЗЧИКА HEX ФАЙЛОВ В ФОРМАТЕ INTEL

Инструментальный загрузчик – это программа, выполняющаяся на компьютере. Ее назначение - загрузка на целевую (рабочую) плату (в нашем случае в учебный лабораторный комплекс SDK1.1) целевой (пользовательской) программы для микроконтроллера в специальном Intel'овском HEX формате. Целевую программу пишет разработчик, для каждого конкретного применения системы на основе того или иного микропроцессора или микроконтроллера (например ADuC812). Загрузка целевой программы в память SDK1.1 и ее выполнение позволяет оценить, как ее работоспособность, так и работоспособность рабочей платы. Во время трансляции исходного текста целевой программы генерируется файл с машинными кодами в специальном Intel'овском HEX формате



ФОРМАТ INTEL'ОВСКОГО HEX ФАЙЛА **(показан фрагмент):**

Файл состоит из однотипных строк

……
:0AC59B00FF7AC6792C12C51B80FE42
:04C666008F828E83AE
……
:00000001FF

:02C50006000033

Каждый байт машинного кода записывается двумя 16-ными цифрами (HEX код). Каждая строчка начинается с двоеточия “:”. Рассмотрим первую строчку. Следующие после двоеточия 2 символа, например 0A обозначают, количество информационных байтов в строке .Следующие 4 символа, C59B являются двухбайтовым адресом, с которого будут располагаться информационные байты текущей строки в памяти целевой платы. После адреса следуют два служебных символа – 00(строка данных) или 01(заверш. строка). Последние 2 символа в строке 42 – байт дополнения контрольной суммы всех предыдущих байтов строки до 256-ти. Оставшиеся байты – информационные. В первой строке примера их 10(0A): FF 7A C6 79 2C 12 C5 1B 80 FE.

Специальная строка :00000001FF завершает информационные строки. Последнюю нестандартную строку вида 02XXXX060000SS со стартовым адресом XXXX программист дописывает самостоятельно, т.к. она не генерируется транслятором. В этой строке 02 код команды перехода LJMP XXXX. XXXX – 16-ный адрес, с которого будет стартовать целевая программа. 060000 – служебные символы. SS - дополнение контрольной суммы всех предыдущих байтов строки до 256-ти.

ПРОТОКОЛ РАБОТЫ РЕЗИДЕНТНОГО ЗАГРУЗЧИКА УЧЕБНОГО ЛАБОРАТОРНОГО КОМПЛЕКСА SDK1.1.

При инициализации системы, во время включения источника питания или при нажатии на кнопку ‘RESET’ резидентный загрузчик с определенным интервалом ~200мсек посылает в компьютер символ “.” – приглашение к передаче строки HEX файла целевой программы. В ответ инструментальный загрузчик передает одну строку HEX файла, в том числе два завершающих символа любой текстовой строки – это символ “возврат каретки (CR)– код 0D” и “перевод строки (LF) – код 0A”. После приема каждой HEX-строки резидентный загрузчик вычисляет контрольную сумму и если она совпадает с принятой посылает инструментальному загрузчику символ “+”, если не совпадает “-“, что указывает на ошибку при передаче HEX-строки. Далее, при безошибочном приеме строки резидентный загрузчик снова посылает уведомление “.” и все повторяется снова. После приема HEX-строки со стартовым адресом 02XXXX060000SS резидентный загрузчик прекращает отправку “.” и передает

управление по адресу XXXX, т.е. запускает целевую программу на выполнение.

РАЗРАБОТКА ИНСТРУМЕНТАЛЬНОГО ЗАГРУЗЧИКА ДЛЯ SDK1.1

В комплекте поставки SDK1.1 имеется инструментальный загрузчик, к недостаткам которого можно отнести отсутствие графического пользовательского интерфейса, а также ряд мелких неудобств. Разработку программы инструментального загрузчика можно производить с использованием любой подходящей среды программирования, например Microsoft Visual Studio, Delphi, Java Builder и т.п. Остановимся на Microsoft Visual Basic'е (в дальнейшем VB), входящим в состав Microsoft Visual Studio и являющимся любимым детищем Б.Г. VB позволяет быстро написать требуемую программу и легко импортировать сторонние библиотеки с набором нужных компонентов.

Разработка формы графического пользовательского интерфейса инструментального загрузчика

1). Запускаем программу "Microsoft Visual Basic 6.0" – VB6.exe

2). В появившемся диалоговом окне кликаем левой кнопкой по "Standart EXE".

Если диалоговое окно не появилось, то из основного меню "File | New Project" также выбираем "Standart EXE". На эране дисплея появится интегрированная среда разработки (IDE) с пустой формой, которую вы должны наполнить функциональным содержанием.

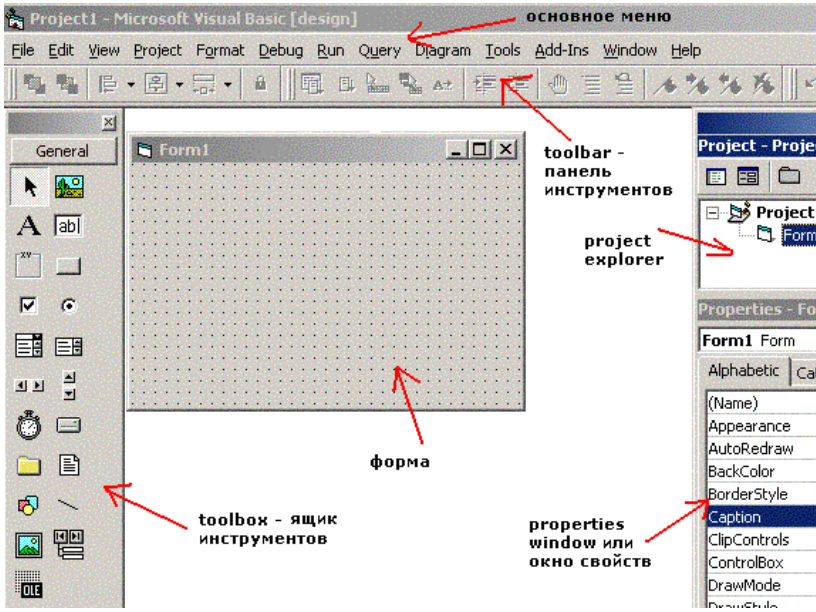


Рис.1

Сохраним файлы нашего проекта, который можно назвать, например “VB_Instr_Ldr”, для чего выберем п. меню “File | Save Project” и в появившемся диалоговом (рис.2) окне перейдем в папку “C:\EMUL\Work\SDK_11\”, где правым кликом вызовем контекстное меню и создадим новую папку с номером учебной группы, например – gr4455. По прежнему, не выходя из диалогового окна входим в эту папку и создаем новую с названием проекта, например My_vbLdr.

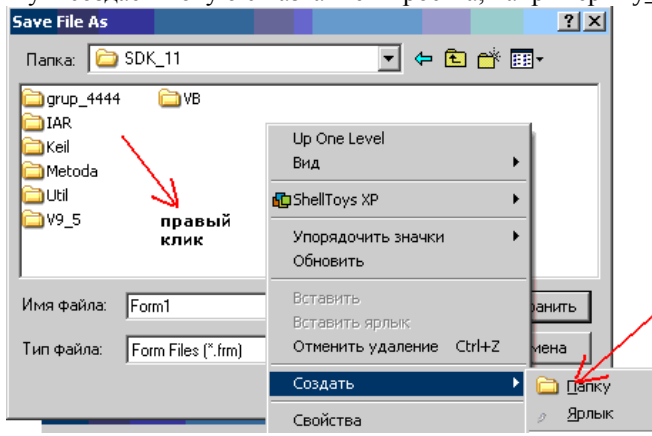
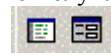
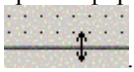


Рис.2

После чего переименуем название формы “Form1”, например на “VB_Instr_Ldr” и нажмем кнопку “Сохранить” или “Save”. Сразу же появится второе диалоговое окно с просьбой сохранить проект. Не мудрствуя, вместо обезличенного “Project1” также подставим “VB_Instr_Ldr” и сохраним проект. На этом этапе в папке “C:\EMUL\Work\SDK_11\gr4455\ My_vbLdr \” должны храниться два файла: VB_Instr_Ldr.vbp и VB_Instr_Ldr.frm. Заодно (это можно и не делать) в окне “Project Explorer” (см. Рис.1) кликнем по имени “Project1” и перейдя к окну “Properties Window” (также см. Рис.1) поменяем название “Project1” на какое-нибудь свое, например “VB_I_L”

3) Приступим к проектированию внешнего вида формы. Переход от окна “Code” т.е. текста программы к окну “Object” в нашем случае к форме производится с помощью двух кнопок  , расположенных сверху окна “Project Explorer” (см. Рис.1) или с помощью тех же кнопок из п. основного меню “View | Code” или “View | Object”. Сначала растянем форму до размера x=9765 и y=7800 условных единиц. Для этого подведем курсор к нижнему краю формы. Курсор приобретет форму двойной стрелки  . Зажав правую кнопку изменим вертикальный размер формы до нужного значения 7800. Аналогично изменим горизонтальный размер формы.

3-1) С помощью декоративных элементов называемых рамками – “Frame” разделим поле формы на три различных функциональных области. Для этого сделаем 3 двойных клика по пиктограмме элемента Frame, находящейся слева на ToolBox’е.

Рис. 2-2

Расположим и растянем, появившиеся три рамки следующим образом (см. рис 3). Далее поместим в рамки необходимые в дальнейшем элементы управления и отображения.

Поместить нужный элемент на форму можно и другим способом, однократно кликнув по нему на ToolBox'е, а затем переместив курсор в нужную точку формы и прижав левую кнопку мыши движением по диагонали придать элементу нужные размеры.

3-2) Для начала кликнем на ToolBox'е по элементу Label  . Отбуксирум появившуюся этикетку "Label1" в рамку "Frame3" и растянем приблизительно до размеров этой рамки. Теперь запишем информирующий текст в "Label1". Сделать это можно двумя способами: 1) на этапе проектирования в свойстве "Caption" для "Label1" в окне "Properties Window" (см. Рис.1), при этом нужно предварительно кликнуть по полю "Label1". Если текст длинный, это не совсем удобно, 2) во время выполнения программы при начальной инициализации (например во время загрузки формы). Воспользуемся вторым способом.

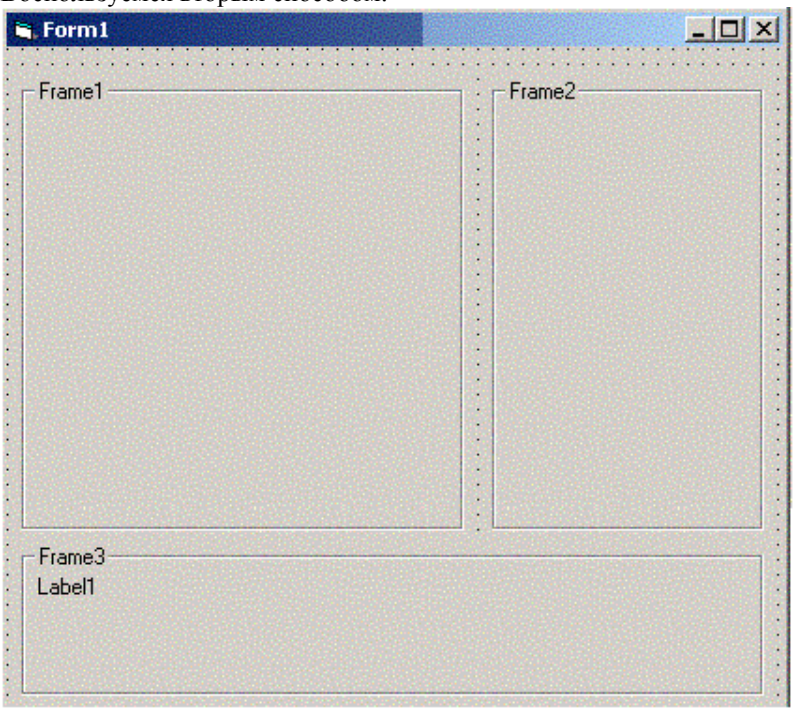
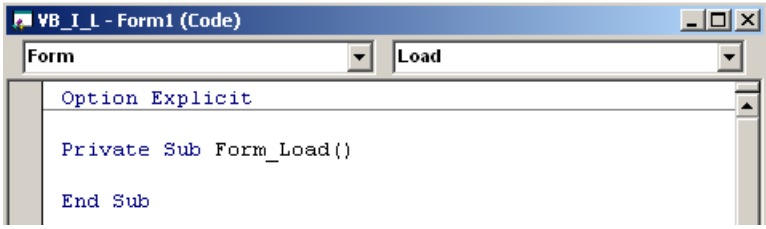
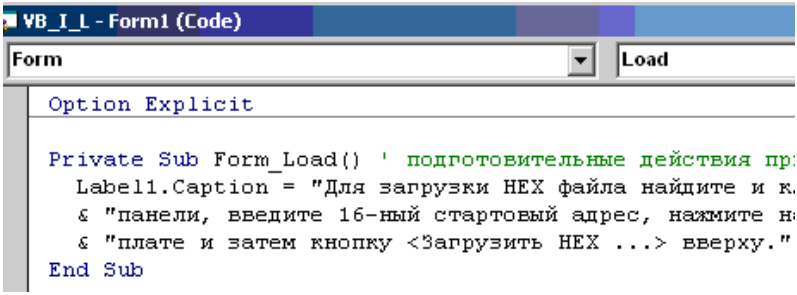


Рис.3

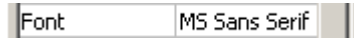
Для этого перейдем к окну “View | Code” и в двух верхних подокнах выберем объект “Form” и событие “Load”. Оболочка VB автоматически создаст каркас для обработчика этого события, куда сейчас мы и запишем свой код.



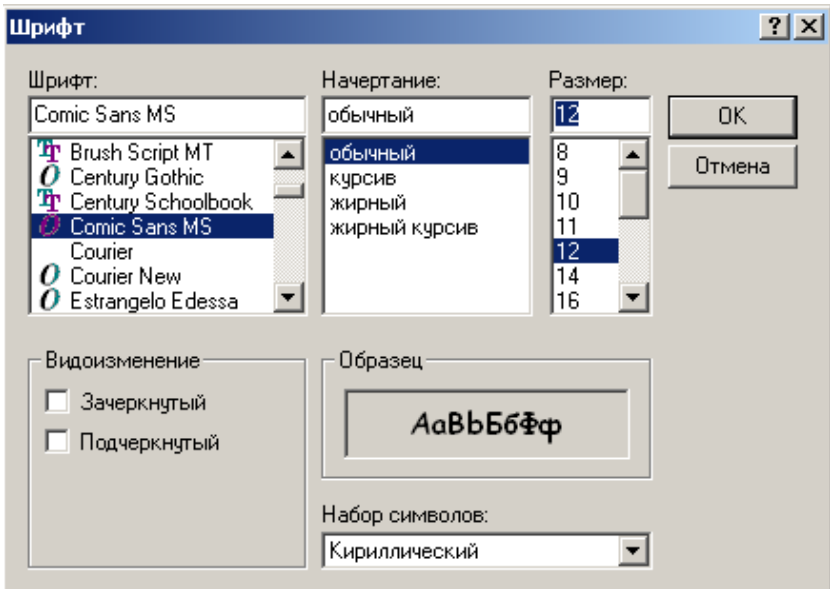


Приведенный на рисунке код вы должны записать в программу самостоятельно. Свойству “Caption” этикетки Label1 присваивается указанный текст. Перенос длинных строк производится с помощью символа “ _”, а конкатенация или слияние с помощью символа “&” или “+”. Оператор “Option Explicit” предотвращает от появления в программе необъявленных переменных, что иначе могло бы затруднить поиск возможных ошибок. Комментарии отделяются символом “ ’ “.

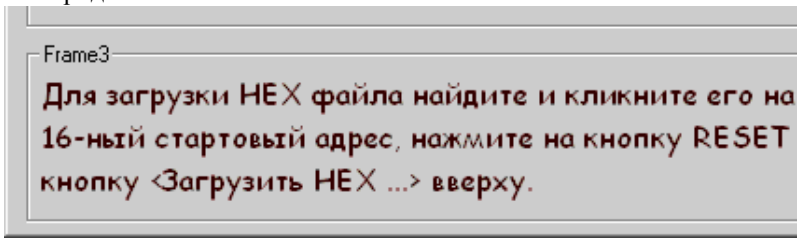
3-3) Проверим первый раз работоспособность программы, для чего кликнем по кнопке  на верхней панели инструментов или нажмем на <F5>. Вас наверно не устроит, что важная информация записана очень маленьким шрифтом. Поэтому остановим выполнение нашей программы кнопкой  и отредактируем свойства шрифта для объекта Label1. С помощью кнопки  поместим форму на передний план, затем кликнем по Label1. На Label1 появятся 8 прямоугольников – маркеров, а слева в окне Properties активизируется элемент Label1. Найдем строчку со

свойством “Font” 

и двойным кликом откроем диалоговое окно.

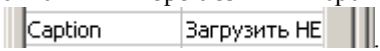


В этом окне произведем указанные установки и нажмем на “OK”. Далее можно поменять цвет надписи с помощью свойства “ForeColor” и т.д. Снова запустим программу  , и увидим что-то в этом роде.



СОВЕТ: Для выявления ошибок после добавления кода в программу делайте пробный запуск  .

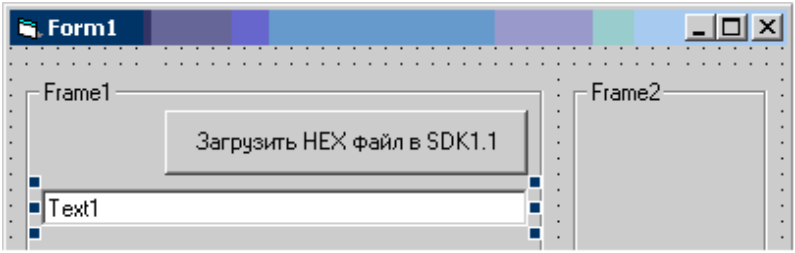
3-4) Прекратим выполнение программы кнопкой  и добавим на форму элемент “Command Button”. Для этого кликнем по

пиктограмме “кнопка” (Рис.2-2) и поместим ее на форму, затем в окне “Properties” переименуем свойство “Caption” 

. Запишем справа от “Caption” – “Загрузить HEX файл в SDK1.1”.

ВАЖНОЕ ПРИМЕЧАНИЕ: При изменении значений свойств того или иного элемента в окне “Properties” убедитесь, что этот элемент находится в фокусе, то есть вокруг него расположены 8 небольших прямоугольников - маркеров. Иначе вы можете случайно изменить такое же свойство, но совсем другого элемента.
ДРУГОЕ ПРИМЕЧАНИЕ: Свойства “Name” элементов здесь и далее оставляем без изменения.

Теперь поместим на форму текстовое окно “TextBox” с именем “Text1” (Рис.2-2), в котором будем отображать имя и путь загружаемого в SDK1.1 HEX-файла целевой программы. На этом этапе верхняя часть формы будет иметь следующий вид.



Далее разместим на форме остальные текстовые окна с “Text2” по “Text5” и еще две этикетки Label2 и Label3, как показано на рисунке (Рис. 4).

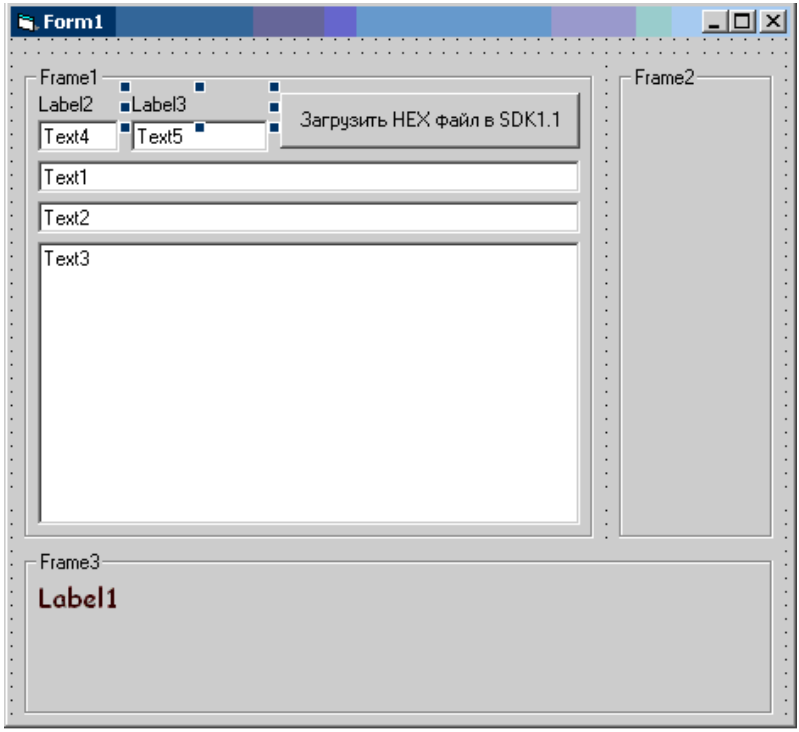
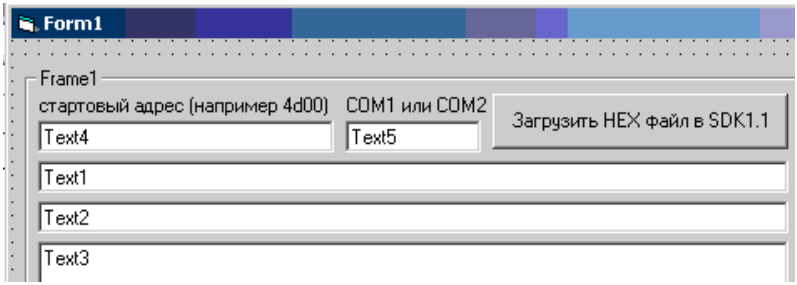
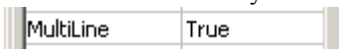


Рис.4

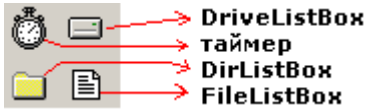
Окно со свойством имя Name=“Text2” используем для отображения текущей строки HEX-файла, передаваемой в SDK1.1. В окне “Text3” будут отображаться служебные символы, передаваемые от SDK1.1. В окно “Text4” пользователь будет записывать стартовый 16-ный адрес целевой программы, которая передается побайтно в SDK1.1. В пятое окно “Text5” пользователь будет записывать имя последовательного порта “COM1” или “COM2”. И, наконец “Label2” и “Label3” будут напоминать нам, что нужно записывать в окна “Text4” и “Text5”. Для этого кликнем по Label2 (появятся маркеры) и перейдем к окну свойств “Properties” (см. Рис.1). В свойство “Caption” элемента Label2 запишем строку - “стартовый адрес (например 4d00)”. Затем кликнем по Label3 и запишем в “Caption” строку “COM1 или COM2”. Теперь верхняя часть формы будет выглядеть таким образом.



Затем, по вашему усмотрению, можно запретить пользователю вводить случайно или намеренно информацию в текстовые окна “Text1”, “Text2” и “Text3”, так как они служат не для ввода, а для отображения. Для этого, в свойствах “Locked” этих окон нужно задать значение “True”. Для окна “Text3” в свойстве “MultiLine” установим значение “True”:



3-5) В правой панели “Frame2” расположим стандартные элементы для работы с файловой системой, которые вытащим из ящика инструментов “ToolBox” (рис. 1)



Заодно прихватим из ящика и таймер, который понадобится в дальнейшем для других целей. Так как таймер не является визуальным элементом формы и его иконка во время выполнения программы не видна, то его можно “бросить” на форму в любом месте. В этот момент форма приобретет почти окончательный вид.

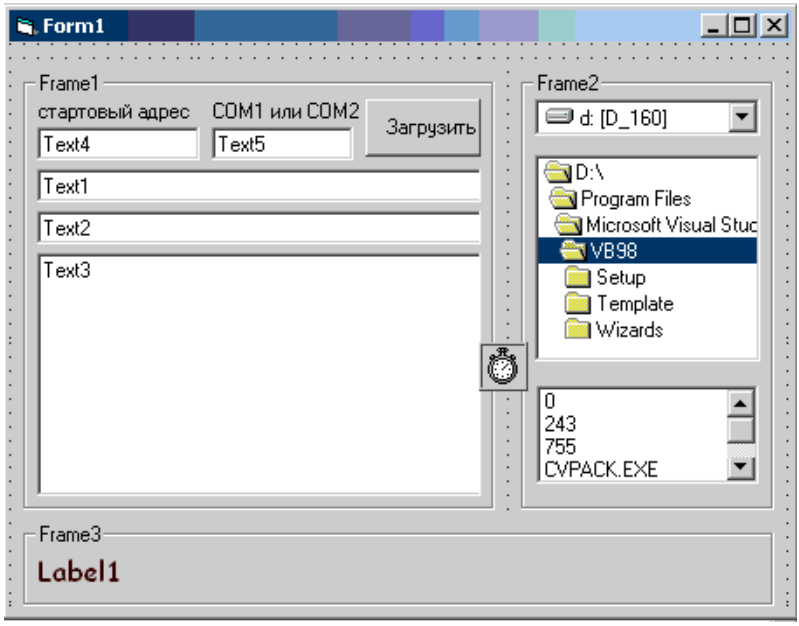
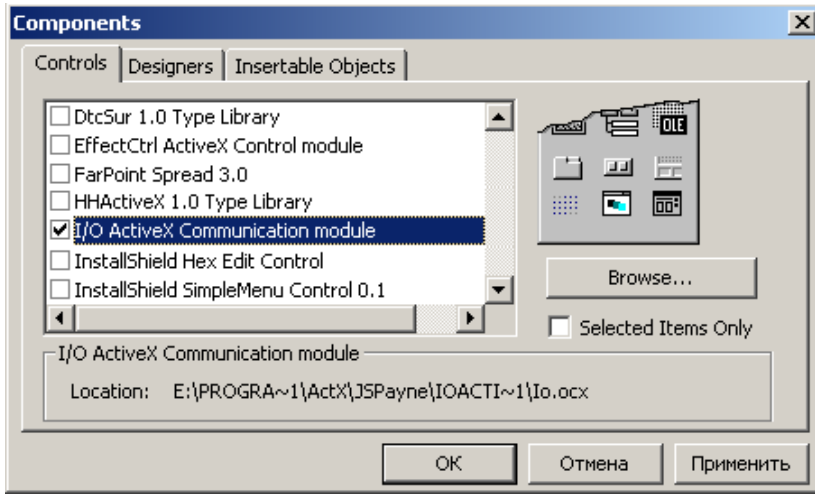


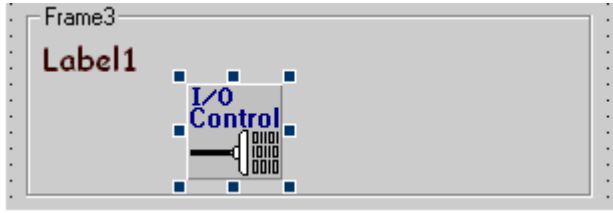
Рис. 4-1

3-6) Теперь нам понадобится объект для работы с последовательным портом, который не входит в стандартный состав “ToolBox'a”. Библиотеки с такими объектами могут, либо входить в состав Windows, либо импортироваться от сторонних разработчиков. Воспользуемся одной из популярных библиотек для работы с COM, LPT и TCP – портами. Обычно для этого такие библиотеки необходимо стандартным образом инсталлировать, а затем импортировать их в проект. В нашем случае библиотека уже инсталлирована. Для импорта компонентов этой библиотеки кликнем ПРАВОЙ кнопкой по “ToolBox’у” и в появившемся окошке контекстного меню кликнем по п. “Components…”. Сразу же появится диалоговое окно. Отметим “птичкой” I/O ActiveX Communicatoin module и нажмем на кнопки “Применить” и “Закрыть”.



В этот момент на “ToolBox’е” появится новый компонент 

. Этот элемент также является невидимым в процессе выполнения программы, поэтому “кинем” его в любом месте формы, например, внизу.



На этом процесс разработки формы закончен. Свойства некоторых компонентов можно подшлифовать, но этим, если останется время займемся позднее. А сейчас перейдем к следующему этапу.

4) ***Разработка программы инструментального загрузчика***

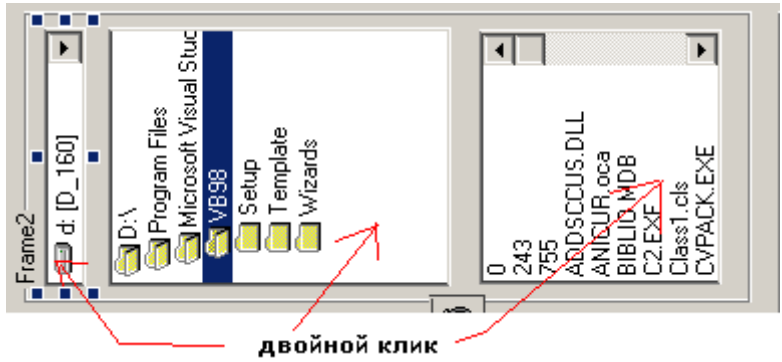
4-1)Так как основной целью инстр. загрузчика является доставка, разработанной вами целевой программы в формате Intel HEX в SDK1.1., то сначала недурно бы найти этот файл. Воспользуемся любым тестовым эквивалентом такого файла, который вы в следующих работах будете писать самостоятельно.

Для экономии времени используем уже готовый файл тестовой целевой программы “Test_Zagr.hex”, который располагается в папке “C:\EMUL\Work\SDK_11\”. Кстати, только для справки, исходный текст этой тестовой целевой программы, разработанной в среде программирования Keil uVision имеет вид:

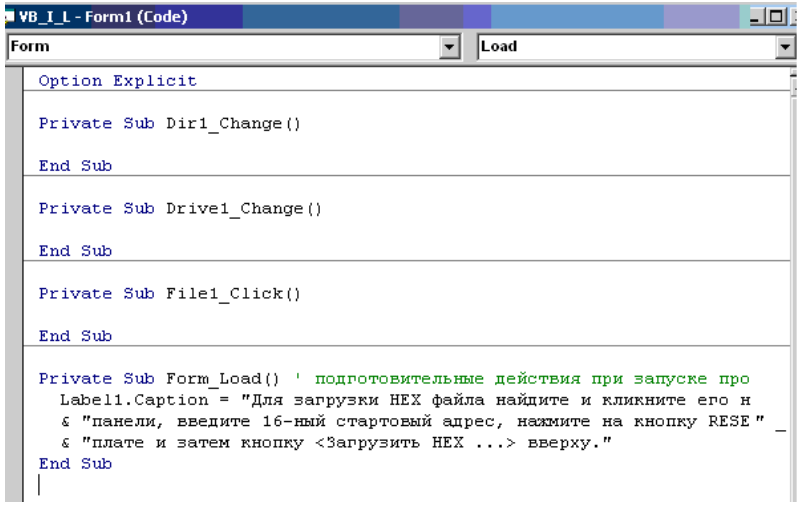
```
# include <ADUC812.H>
# include <stdio.h>
# include <LCD.H>
# include <MAX.H>
main(){
  WriteMax(LEDLINE,0xAA);
   InitLCD(); LCD_Clear();
  LCD_Type("Test Zagruzchika");
  LCD_GotoXY(0,1);LCD_Type("LITMO - 2006");
  while(2017);
}
```

Если загрузка и запуск пройдут успешно, на ЖК – индикторе появятся две строки: “Test Zagruzchika” и ‘LITMO - 2006", а на линейке светодиодов включенные светодиоды будут чередоваться (0xAA = 10101010).

Сделайте пробный запуск, разрабатываемого вами инструментальгого загрузчика, кликнув по кнопке  на панели инструментов VB. Теперь попробуйте “покликать” мышью по компонентам “DriveListBox”, ”DirListBox” и ”FileListBox”, которые находятся на правой панели “Frame2”. Вы увидите, что в среднем окне для работы с папками - ”DirListBox” никак не отображается смена текущего устройства – тома, производимого вами в окне “DriveListBox”. А в нижнем окне ”FileListBox”никак не отображается смена папок, производимая вами в окне “DirListBox”. Для этого прекратим выполнение программы, кликнув по кнопке  на панели инструментов VB. При этом мы вернемся в режим проектирования. Сделаем три двойных клика по указанным компонентам. Для экономии места панель “Frame2” на рисунке положена на бок.



Интегрированная среда разработки Visual Basic'a автоматически создаст три шаблона для трех обработчиков событий (по умолчанию это будут Dir1_Change, Drive1_change и File1_Click). Конечно, тип событий можно изменить, но обычно по умолчанию генерируется как раз то, что нужно).



Осталось добавить свой код в эти обработчики, как показано внизу.

```
VB_I_L - Form1 (Code)
Drive1                                                    Change
Option Explicit
Private Sub Dir1_Change() 'клик по папке в окне DirListBox
    ChDir Dir1.Path ' меняем текущую директорию на папку по кото
    File1.Path = Dir1.Path ' отображаем содержимое текущей папк
End Sub

Private Sub Drive1_Change() ' клик по выбранному устройству(том
                            ' DriveListBox
    ChDrive Drive1.Drive ' меняем текущее устройство
    Dir1.Path = Drive1.Drive ' отображаем папки текущего устрой
End Sub

Private Sub File1_Click() 'клик по выбранному файлу в окне File
    Text1 = File1.Path & "\" & File1.FileName ' отображаем выбр
                                              ' в текстовом окн
End Sub

Private Sub Form_Load() ' подготовительные действия при запуске
  Label1.Caption = "Для загрузки HEX файла найдите и кликните   " _
  & "панели, введите 16-ный стартовый адрес, нажмите на кнопку     " _
  & "плате и затем кнопку <Загрузить HEX ...> вверху."
End Sub
```

Снова запустите программу с помощью <F5> или  и покликав мышью перейдите в папку “C:\EMUL\Work\SDK_11\”. Вы увидите что-то вроде:

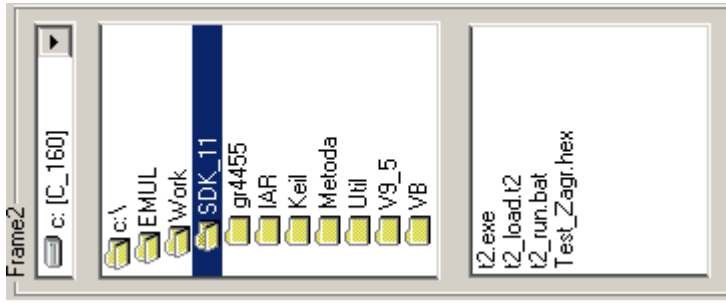
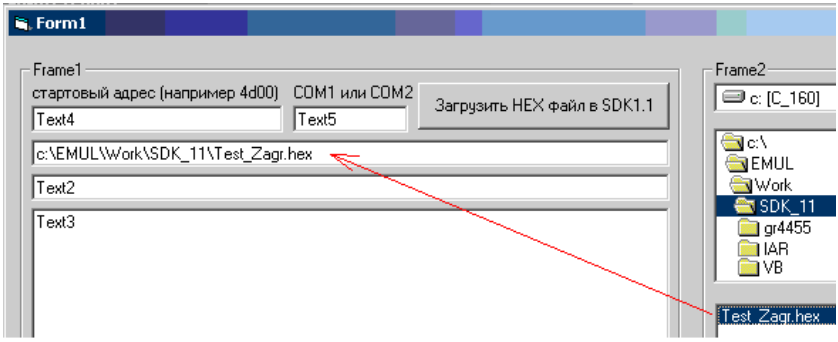


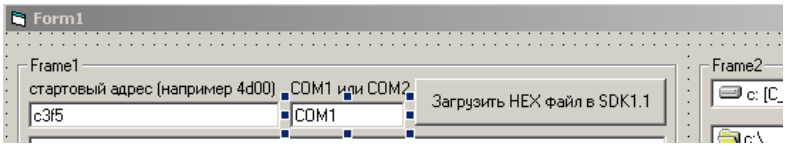
Рисунок для экономии места опять положен на бок. Так как файл “Test_Zagr.hex” может “потеряться” в других файлах с другими расширениями, полезно ввести в программу маску, которая будет отображать только файлы с расширениями “hex”. Для этого завершим выполнение программы  и добавим в обработчик “Form_Load” строку: File1.Pattern = "*.hex" (в красной рамке).

```
Private Sub Form_Load() ' подготовительные действия при
  File1.Pattern = "*.hex" ' отображаем в FileListBox фа
  Label1.Caption = "Для загрузки HEX файла найдите и кл
  & "панели, введите 16-ный стартовый адрес, нажмите на
  & "плате и затем кнопку <Загрузить HEX ...> вверху."
End Sub
```

Если снова запустить программу и кликнуть по “Test_Zagr.hex”, который теперь в одиночестве, то полный путь до файла автоматически появится в текстовом окне “Text1” (что и требуется).

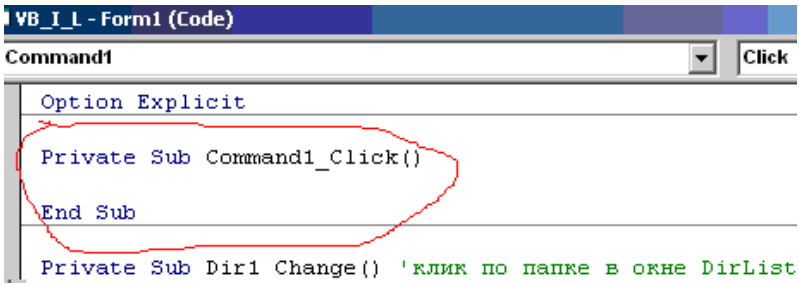


4-1) Перейдем к программированию основного цикла пересылки HEX-файла в SDK1.1. Сначала заменим с помощью “Properties Window” (рис. 1) обезличенные свойства “Text” в текстовых окнах “Text4” и “Text5” на принятые по умолчанию для нашего конкретного примера: Text4.Text → c3f5 и Text4.Text → COM1. Теперь форма будет иметь следующий вид:



Вся основная работа по пересылке HEX-файла будет производиться при клике по кнопке “Загрузить HEX файл в SDK1.1”, поэтому создадим обработчик события при нажатии на эту кнопку, двойным кликом по ней (естественно в режиме проектирования).

Сразу же в окне кода программы появится шаблон этого обработчика по умолчанию.



Для начала добавим вывод диагностических сообщений в окно “Text3”, если по какой-то причине в текстовых окнах “Text1”, “Text4” и “Text5” не окажется нужных записей.

Если в этих окнах окажутся корректные записи, можно открывать файл с именем в текстовом окне “Text1” (см. Рис. 4-1). По этой причине свойству Text текстового окна Text1 присвоим пустую строку стерев, имеющуюся там запись “Text1”: 

, иначе при запуске наша инструментальная программа будет искать файл “Text1” вместо “Test_Zagr.hex”.

Заодно в начале текста программы добавим объявления переменных, которые в дальнейшем нам понадобятся (в красной рамке).

Итак, в обработчик нажатия кнопки “Command1_Click()” вы должны добавить следующий код (тоже в красной рамке).

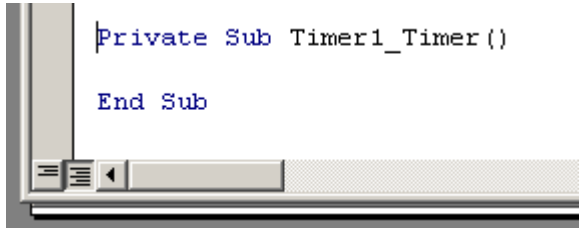
Теперь программа при запуске обязательно нам напомнит, если мы забудем заполнить какое-нибудь текстовое поле. Функция Trim() убирает пробелы и табуляции, которые вы могли записать ненароком в эти окна.

Обратите внимание, также на то, что текст записывается в свойства “Text” текстовых окон, например “Text1.Text”. Однако в Visual Basic’е 6.0 (но не в 7.0 или Visual Basic NET) можно не записывать свойства, которые подразумеваются по умолчанию. Например, можно записать более коротко <If Trim(Text4) = "" Then> вместо <If Trim(Text4.Text) = "" Then>, т.к. для текстовых окон свойством по умолчанию является свойство “Text” и его можно “опустить”.

```
Option Explicit
Dim fN As Integer, t As Byte, cntrSum As Long, tt1 As Integer
Dim s1 As String, s2 As String, s3 As String, st4 As String,

Private Sub Command1_Click()
    If Trim(Text4.Text) = "" Then
      Text3 = "ВВЕДИТЕ СТАРТОВЫЙ АДРЕС В 16-НОМ КОДЕ И ПОВТОРI
      Exit Sub ' больше нечего делать
    End If
    If Trim(Text5.Text) = "" Then
     Text3 = "ЗАПИШИТЕ В ОКНО 'COM1 или COM2' НАЗВАНИЕ ПОРТА J
             & "ЛИБО 'COM2' БЕЗ КАВЫЧЕК И ПОВТОРИТЕ ЗАПУСК"
      Exit Sub ' больше нечего делать
    End If
    If Trim(Text1.Text) = "" Then
      Text3 = "ВЫБЕРИТЕ HEX-ФАЙЛ И ПОВТОРИТЕ ЗАПУСК"
      Exit Sub ' больше нечего делать
    Else ' теперь все в порядке
      fN = FreeFile ' назначаем файловый номер
      Open Text1 For Input As fN ' открываем файл с именем в
    End If
    Command1.Enabled = False ' деактивируем кнопку запуска, д
    '.........               ' повторных нажатий
    '.........
    Close fN
End Sub
```

Поэтому, например вместо Text3.Text = "ВВЕДИТЕ СТАРТОВЫЙ ...” в программе использована короткая запись Text3 = "ВВЕДИТЕ СТАРТОВЫЙ ...”. В строке “Open Text1 For Input As fN” также использована короткая запись. В дальнейшем будет использоваться именно короткая запись.

4-2) Добавим в программу код, для инициализации последовательного канала и проверки, есть ли связь с SDK1.1. Для обнаружения, есть ли связь используем компонент “Timer1”, который мы предусмотрительно поместили на форму. Как было сказано в разделе “Протокол работы резидентного загрузчика..” резидентный загрузчик SDK1.1 посылает в последовательный канал символ “.” – приглашение к загрузке. Наша задача определить передается точка или нет. При положительном ответе – наладить передачу HEX-файла, в противном случае вывести диагностическое сообщение. Сначала добавим обработчик переполнения таймера, двойным кликом по иконке  , расположенной на форме (но не на ToolBox’е). В тексте программы появится шаблон.



Теперь добавим код в обработчик Timer1_Timer() и в обработчик Command1_Click(). На рисунке внизу новый код, который вам нужно записать в программу обведен красными рамками.

```
Private Sub Command1_Click()
   ..........
    Open Text1 For Input As fN ' открываем файл с именем в окне Text1 для чтения
  End If
  Command1.Enabled = False ' деактивируем кнопку запуска, для исключения
                           ' повторных нажатий
  IORes = IO1.Open(Trim(Text5) & ":","baud=9600 parity=N data=8 stop=1")
                                    ' открываем последовательный канал связи
  IORes = IO1.SetTimeOut(100) ' 100 мсек, SetTimeOut(0)-бескон.(м.б. зависание)
  Timer1.Interval = 500 ' таймер будет срабатывать через 0.5 секунды
  Timer1.Enabled = True ' запустить таймер
  Do
    t = IO1.ReadByte() And &HFF ' определяем идет ли передача со стороны SDK1.1
    If t = Asc(".") Then Timer1.Enabled = False: Exit Do ' если да то
                        ' остановить таймер и закончить проверку связи
    DoEvents ' необходимо для аварийного выхода по <ALT+F4> или кликом
  Loop       ' по кнопке окна 'X'
  '..........
  '..........
  IO1.Close ' закрываем последовательный канал связи
  Close fN ' Закрываем файл с номером fN (могло быть и больше откр-ых ф-лов)
End Sub
Private Sub Timer1_Timer() 'обработчик переполнения таймера
  If t <> Asc(".") Then ' прошло полсекунды, а связи с SDK1.1 нет
    Text3 = "======== НЕТ СВЯЗИ С SDK1.1 ========" & vbCrLf & vbCrLf _
    & "1. ПРОВЕРЬТЕ ПОДКЛЮЧЕНИЕ SDK1.1 К КОМПЬЮТЕРУ" & vbCrLf & vbCrLf _
    & "2. ПРОВЕРЬТЕ ПОДКЛЮЧЕНО ЛИ ПИТАНИЕ К SDK1.1" & vbCrLf & vbCrLf _
    & "3. ЕСЛИ SDK1.1 НЕ НАХОДИТСЯ В РЕЖИМЕ ОЖИДАНИЯ - НАЖМИТЕ КНОПКУ 'RESET'" _
    & "НА ЦЕЛЕВОЙ ПЛАТЕ" & vbCrLf & vbCrLf _
    & "4. В КРАЙНЕМ СЛУЧАЕ ПРОВЕРЬТЕ ПОДКЛЮЧЕН ПОРТ COM1 ИЛИ COM2"
    Timer1.Enabled = False ' выключить таймер, т.е. сработает он только один раз
  End If
End Sub
```

Первый оператор “IORes = IO1.Open(Text5 & ":", "baud=9600 parity=N data=8 stop=1")” открывает последовательный канал связи компьютера со скоростью передачи/приема 9600

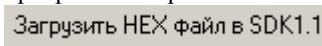
бит/сек, без контроля четности с одним стоп-битом (эти установки должны точно соответствовать установкам COM-порта в SDK1.1. В свойстве Text окна Text5 уже должно находиться наименование порта “COM1” или “COM2”. В переменную IORes записывается результат: удалось открыть COM-порт или нет, но мы эту проверку не производим. В следующей строке задается интервал TimeOut в мсек (если TimeOut =0, то функция IO1.ReadByte() будет ждать появление символа из последовательного канала бесконечно, и программа может зависнуть). Назначение следующих двух строк понятно из комментариев. В бесконечном цикле Do …. Loop периодически пытаемся прочитать символ из COM-порта с помощью IO1.ReadByte() и, если удалось прочитать символ приглашения “.”, то останавливаем таймер и выходим из цикла, если нет, то повторяем попытку чтения.

Для аварийного выхода из циклов, которые при определенных условиях могут стать бесконечными и безнадежно подвесить программу предусмотрен оператор DoEvents, который позволяет отследить внешние события, например нажатия на клавиши, в том числе <ALT+F4> или клики по кнопке  . Далее, по истечении 500 мсек, заданных в строке Timer1.Interval = 500 (если не будет принят символ “.”), Timer1 переполнится и возникнет событие Timer1_Timer(). Обработчик этого события выведет в текстовое окно Text3 диагностическое сообщение и выключит таймер (предопределенная константа vbCrLf обозначает управляющие символы возврата каретки и перевода строки – коды 13 и 10). Если символ “.” будет обнаружен, то таймер будет выключен досрочно, программа “двинется” дальше и начнется передача HEX-файла (этот блок в текст еще не включен). Далее, т.к. функция IO1.ReadByte() возвращает 32-битное значение, то нужно выделить младший байт с ASCII кодом символа “.” с помощью 8-ми битной маски “FF” а для того, чтобы не заморачиваться и не искать в справочнике ASCII код точки поручим это встроенной функции VB – Asc(“.”). И последнее примечание: в одной строчке кода можно размещать несколько операторов VB, но разделяются они не точкой с запятой, а двоеточием, как в строке “ If t = Asc(".") Then Timer1.Enabled = False**:** Exit Do”.

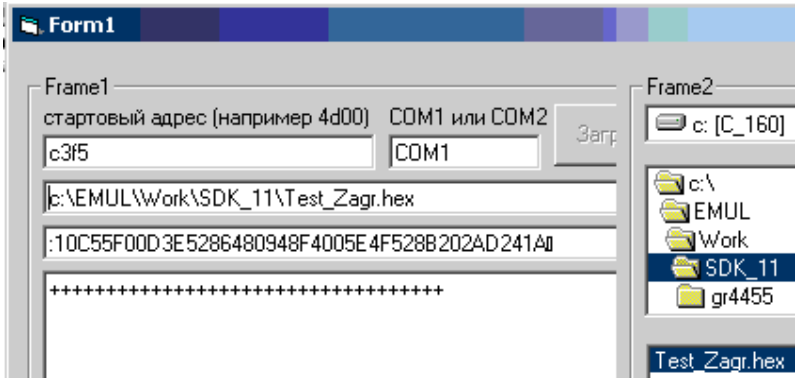
4-3) Перейдем к разработке основного цикла передачи HEX-файла в SDK. Тело этого цикла расположено между операторами: Do While Not EOF(fN) и Loop ' конец цикла While Not EOF(fN).

```
                              ' остановить таймер и закончить проверку связи
    DoEvents ' необходимо для аварийного выхода по <ALT+F4> или кликом
  Loop       ' по кнопке окна 'X'
  s3 = ""
  Do While Not EOF(fN) ' выполнять цикл пока не наступит конец файла - EOF
    Do 'ждем "." - приглашение к передаче строки Intel'овского HEX файла
      t = IO1.ReadByte() And &HFF ' читаем байт из COM-порта
      If (t = Asc("-")) Or (t = Asc("+")) Then ''+' - строка принята в SDK
        s3 = s3 + Chr(t): Text3 = s3 ' выводим строку прогресса передачи
        If t = Asc("-") Then  ' '-' - строка принята в SDK1.1 с ошибкой
          Text3 = s3 + " ПОВТОРИТЕ ЗАГРУЗКУ."
          GoTo err_exit ' целевая плата и компьютер не поняли друг друга
        End If
      ElseIf t = Asc(".") Then Exit Do 'теперь можно передавать текущую стр
      End If
      DoEvents 'для аварийного выхода по <ALT+F4> или кликом по кнопке окна
  Loop
  s2 = ""
  Do ' переправить в SDK1.1 очередную строку Intel'овского HEX файла
    s1 = Input(1, fN) ' прочитать из HEX-строки очередной байт
    IO1.WriteByte (Asc(s1)) ' и отправить его через COM порт в целевую пла
    If s1 = vbLf Then Exit Do 'закончить передачу строки, если достигнут
    s2 = s2 + s1 ' дописываем текущий символ s1 HEX строки
    DoEvents
  Loop
  Text2 = s2 'вывести текущую HEX строку в текстовое окно
  DoEvents
 Loop ' конец цикла While Not EOF(fN)
  '..........
  '..........
err_exit:
  Command1.Enabled = True 'активируем кнопку загрузки для следующего сеанса
  IO1.Close ' закрываем последовательный канал связи
  Close fN ' Закрываем файл с номером fN (могло быть и больше откр-ых ф-лов)
End Sub
```

Рис.5

Добавьте в свою программу два указанных на рисунке фрагмента. Выполните пробный запуск  для выявления возможных ошибок при копировании фрагментов. Если запуск программы прошел безошибочно, то при нажатии на кнопку 

начнется процесс передачи HEX-файла в SDK и вид формы будет примерно таким, как на рисунке.



В окне Text2 отображается текущая строка, а общее количество строк успешно переданных к этому моменту в SDK соответствует числу “+”-ов в окне Text3. Кнопка Загрузить HEX файл в SDK1.1 во время передачи деактивирована, чтобы избежать повторного запуска, если не закончился текущий. В случае успешной передачи в окне Text2 появится запись “:00000001FF”, а кнопка “Загрузить...” снова активизируется.

Теперь прокомментируем добавленный в программу код. Основной цикл “Do While Not EOF(fN)... Loop” выполняется пока (While) не будет обнаружен признак конца файла – “EOF (End Of File)”. Основной цикл содержит два других цикла: Do…Loop.

В первом определяем, какой ответ послал резидентный загрузчик SDK нашему инструментальному загрузчику: “+”, “-“ или “.” Во втором цикле Do…Loop пересылаем побайтно текущую строку HEX-файла. Вспомогательные текстовые переменные s2 и s3 служат буфером при отображении информации в текстовых окнах Text2 и Text3. В первом цикле Do…Loop для формирования диагностической строки используется встроенная в VB функция Chr(код символа), которая возвращает принятый символ с соответствующим кодом и добавляет его к текстовой переменной s3. Если раскодирование HEX-строки в SDK прошло без ошибки, то резидентный загрузчик посылает нам через COM-порт “+“, в противном случае “-“. При положительном исходе посылает в последовательный канал “.”, т.е. делает приглашение инструментальному загрузчику переслать очередную HEX-строку. При отрицательном исходе производится аварийный выход в конец обработчика Command1_Click() на метку “err_exit:”,при этом закрываются последовательный канал и открытый файл.

Во втором цикле Do…Loop читаем из текущей HEX-строки во вспомогательную переменную s1 очередной байт и отправляем его в последовательный канал. Очевидным концом строки любого текстового файла является в большинстве случаев последовательность двух управляющих ASCII символов (коды 13 и 10). Нам в цикле достаточно отследить появление только одного: с кодом 10, которому в VB соответствует предопределенная константа vbLF. После того как будет прочитан и передан последний байт, происходит выход из цикла: “Loop ' конец цикла While Not EOF(fN)”.

4-4) Теперь вместо многоточия запишем последний фрагмент программы, в котором сформируем и передадим резидентному загрузчику строку со стартовым адресом (см. Раздел ‘Формат Intel’овского HEX файла’).

```
    Loop ' конец цикла While Not EOF(fN)
    '..........
    '..........
err_exit:
    Command1.Enabled = True 'активируем кнопку загрузки для следующего сеанса
```

Ниже, на рисунке отмечен код последнего фрагмента, который перенесите в программу:

```
 Loop ' конец цикла While Not EOF(fN)
 '//== сформировать и передать строку со стартовым адресом 02XXXX060000SS<cr>
 Do '//== снова ждем приглашение - символ "."
   t = IO1.ReadByte() And &HFF ' читаем байт "." из целевой платы
   If t = Asc(".") Then Exit Do ' теперь переходим к формированию и передаче
   DoEvents                     ' строки со стартовым адресом
 Loop
 st4 = Trim(Text4) ' записываем адрес во вспом. переменную st4
 tt1 = Val("&H" + Left(st4, 2)) ' вычисляем ст. байт адреса
 tt2 = Val("&H" + Right(st4, 2)) ' вычисляем мл. байт адреса
 cntrSum = 256 - ((2 + 6 + tt1 + tt2) Mod &H100) ' находим дополн. контрол.
 s2 = ":02" & st4 & "060000" + Hex(cntrSum) + vbCr 'или vbLf
 IO1.WriteString (s2): Text2 = s2
 Text3 = s3 + " загрузка прошла успешно, контрольная сумма старт. строки = "_
 & Hex(cntrSum)
err_exit:
 Command1.Enabled = True 'активируем кнопку загрузки для следующего сеанса
 IO1.Close ' закрываем последовательный канал связи
 Close fN ' Закрываем файл с номером fN (могло быть и больше откр-ых ф-лов)
End Sub
```

Введите указанный фрагмент в программу и сверьтесь с полным окончательным текстом программы инструментального

загрузчика, который приведен ниже. Снова запустите программу  , если при трансляции возникли ошибки остановите программу  и исправьте ошибки. Если запуск прошел успешно и целевая программа была загружена в целевую плату, то на экране ЖК – дисплея SDK высветятся две строки: “Test Zagruzchika” и ‘LITMO - 2006", а на линейке светодиодов “зажженные” светодиоды будут чередоваться (0xAA = 10101010).

Несколько пояснений к последнему фрагменту программы: Этот фрагмент предназначен для формирования строки типа 02XXXX060000SS<cr> со стартовым адресом (см. раздел “Формат Intel’овского HEX файла”).В строке tt1 = Val("&H" + Left(st4, 2)) из текстовой строки st4= “c3f5” функцией Left будут вырезаны 2 символа слева, т.е. “c3”. Затем строка “&H” сольется со строкой “c3” и полученная строка “&Hc3” с 16-ной записью числа c3 будет преобразована в числовое значение c3(HEX) = 195(DEC). Это число будет помещено во вспомогательную переменную tt1. В следующей строке tt2 = Val("&H" + Right(st4, 2)) будут вырезаны два правых символа, т.е. “f5”, а затем преобразованы в числовое значение f5(HEX) = 245(DEC). Далее находим сумму всех байтов строки 02 + tt1 + tt2 + 06 + 00 + 00. Естественно, нули складывать не обязательно. Затем находим остаток этой суммы по модулю 256(&H100) – это и есть контрольная сумма. Вычтя из 256 этот остаток получим дополнительный до 256 или просто дополнительный код (не совсем корректно обозначенный cntrSum). Этот код затем преобразуется в 16-ное строковое представление функцией Hex(cntrSum). В следующей строке программы компонуем требуемую Intel’овскую строку s2 = ":02" & st4 & "060000" + Hex(cntrSum) + vbCr, которую завершает признак конца строки - предопределенный символ vbCr (можно и vbLf – резидентный загрузчик принимает и то и другое). Скомпонованную строку пересылаем одним оператором IO1.WriteString (s2) и отображаем в текстовом окне Text2 = s2.

Теперь целевая программа загружена в SDK1.1 и вы можете наблюдать результаты ее работы.

Окончательный текст инструментального загрузчика

```
Option Explicit
Dim fN As Integer, t As Byte, cntrSum As Long, tt1 As Integer, tt2 As
Integer
```

```
Dim s1 As String, s2 As String, s3 As String, st4 As String, IORes As
Long

Private Sub Command1_Click()
   If Trim(Text4.Text) = "" Then
     Text3 = "ВВЕДИТЕ СТАРТОВЫЙ АДРЕС В 16-НОМ КОДЕ И
ПОВТОРИТЕ ЗАПУСК"
     Exit Sub ' больше нечего делать
   End If
   If Trim(Text5.Text) = "" Then
     Text3 = "ЗАПИШИТЕ В ОКНО 'COM1 или COM2' НАЗВАНИЕ
ПОРТА ЛИБО 'COM1'" _
           & "ЛИБО 'COM2' БЕЗ КАВЫЧЕК И ПОВТОРИТЕ ЗАПУСК
"
     Exit Sub ' больше нечего делать
   End If
   If  Trim(Text1.Text) = "" Then
     Text3 = "ВЫБЕРИТЕ HEX-ФАЙЛ И ПОВТОРИТЕ ЗАПУСК"
     Exit Sub
   Else ' теперь все в порядке
     fN = FreeFile ' назначаем файловый номер
     Open Text1 For Input As fN ' открываем файл с именем в окне
Text1 для чтения
   End If
   Command1.Enabled = False ' деактивируем кнопку запуска, для
исключения
                      ' повторных нажатий
   IORes = IO1.Open(Trim(Text5) & ":", "baud=9600 parity=N data=8
stop=1") ' открываем
                                    ' последовательный канал связи
   IORes = IO1.SetTimeOut(100) ' 100 мсек, SetTimeOut(0)-
бескон.(м.б. зависание)
   Timer1.Interval = 500 ' таймер будет срабатывать через 0.5 секунды
   Timer1.Enabled = True ' запустить таймер
   Do
     t = IO1.ReadByte() And &HFF ' определяем идет ли передача со
стороны SDK1.1
     If t = Asc(".") Then Timer1.Enabled = False: Exit Do ' если да то
                      ' остановить таймер и закончить проверку связи
     DoEvents ' необходимо для аварийного выхода по <ALT+F4> или
кликом
   Loop       ' по кнопке окна 'X'
```

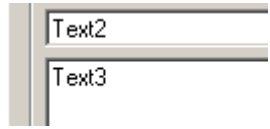
```
   s3 = ""
   Do While Not EOF(fN) ' выполнять цикл пока не наступит конец
файла - EOF
     Do 'ждем "." - приглашение к передаче строки Intel'овского HEX
файла
       t = IO1.ReadByte() And &HFF ' читаем байт из COM-порта
       If (t = Asc("-")) Or (t = Asc("+")) Then "+' - строка принята в SDK
успешно
         s3 = s3 + Chr(t): Text3 = s3 ' выводим строку прогресса
передачи
         If t = Asc("-") Then  ' '-' - строка принята в SDK1.1 с ошибкой
           Text3 = s3 + " ПОВТОРИТЕ ЗАГРУЗКУ."
           GoTo err_exit ' целевая плата и компъютер не поняли друг
друга
         End If
       ElseIf t = Asc(".") Then Exit Do 'теперь можно передавать
текущую строку
       End If
       DoEvents 'для аварийного выхода по <ALT+F4> или кликом по
кнопке окна 'X'
     Loop
     s2 = ""
     Do ' переправить в SDK1.1 очередную строку Intel'овского HEX
файла
       s1 = Input(1, fN) ' прочитать из HEX-строки очередной байт
       IO1.WriteByte (Asc(s1)) ' и отправить его через COM порт в
целевую плату
       If s1 = vbLf Then Exit Do 'закончить передачу строки, если
достигнут ее конец
       s2 = s2 + s1 ' дописываем текущий символ s1 HEX строки
       DoEvents
     Loop
     Text2 = s2 'вывести текущую HEX строку в текстовое окно
     DoEvents
   Loop ' конец цикла While Not EOF(fN)
   '//== сформировать и передать строку со стартовым адресом
02XXXX060000SS<cr>
   Do '//== снова ждем приглашение - символ "."
     t = IO1.ReadByte() And &HFF ' читаем байт "." из целевой платы
     If  t = Asc(".") Then Exit Do ' теперь переходим к формированию и
передаче
     DoEvents                    ' строки со стартовым адресом
```

```
    Loop
    st4 = Trim(Text4) ' записываем адрес во вспом. переменную st4
    tt1 = Val("&H" + Left(st4, 2)) ' вычисляем ст. байт адреса
    tt2 = Val("&H" + Right(st4, 2)) ' вычисляем мл. байт адреса
    cntrSum = 256 - ((2 + 6 + tt1 + tt2) Mod &H100) ' находим дополн.
контрол. суммы
    s2 = ":02" & st4 & "060000" + Hex(cntrSum) + vbCr 'или vbLf
    IO1.WriteString (s2): Text2 = s2
    Text3 = s3 + " загрузка прошла успешно, контрольная сумма старт.
строки = " _
    & Hex(cntrSum)
err_exit:
    Command1.Enabled = True 'активируем кнопку загрузки для
следующего сеанса
    IO1.Close ' закрываем последовательный канал связи
    Close fN ' Закрываем файл с номером fN (могло быть и больше
откр-ых ф-лов)
End Sub

Private Sub Dir1_Change() 'клик по папке в окне DirListBox
    ChDir Dir1.Path ' меняем текущую директорию на папку по
которой кликнули
    File1.Path = Dir1.Path ' отображаем содержимое текущей папки
End Sub

Private Sub Drive1_Change() ' клик по выбранному устройству(тому)
в окне  DriveListBox
    ChDrive Drive1.Drive ' меняем текущее устройство
    Dir1.Path = Drive1.Drive ' отображаем папки текущего устройства
End Sub

Private Sub File1_Click() 'клик по выбранному файлу в окне
FileListBox
    Text1 = File1.Path & "\" & File1.FileName ' отображаем выбранный
файл  в текстовом окне Text1
End Sub

Private Sub Form_Load() ' подготовительные действия при запуске
программы
  File1.Pattern = "*.hex" ' отображаем в FileListBox файлы только с
раширением HEX
```

```
  Label1.Caption = "Для загрузки HEX файла найдите и кликните его
на правой " _
  & "панели, введите 16-ный стартовый адрес, нажмите на кнопку
RESET на целевой " _
  & "плате и затем кнопку <Загрузить HEX ...> вверху."
End Sub

Private Sub Timer1_Timer() 'обработчик переполнения таймера
  If t <> Asc(".") Then ' прошло полсекунды, а связи с SDK1.1 нет
    Text3 = "======== НЕТ СВЯЗИ С SDK1.1 ========" & vbCrLf
& vbCrLf _
    & "1. ПРОВЕРЬТЕ ПОДКЛЮЧЕНИЕ SDK1.1 К КОМПЬЮТЕРУ"
& vbCrLf & vbCrLf _
    & "2. ПРОВЕРЬТЕ ПОДКЛЮЧЕНО ЛИ ПИТАНИЕ К SDK1.1" &
vbCrLf & vbCrLf _
    & "3. ЕСЛИ SDK1.1 НЕ НАХОДИТСЯ В РЕЖИМЕ ОЖИДАНИЯ
- НАЖМИТЕ КНОПКУ 'RESET'" _
    & "НА ЦЕЛЕВОЙ ПЛАТЕ" & vbCrLf & vbCrLf _
    & "4. В КРАЙНЕМ СЛУЧАЕ ПРОВЕРЬТЕ ПОДКЛЮЧЕН ПОРТ
COM1 ИЛИ COM2"
    Timer1.Enabled = False ' выключить таймер, т.е. сработает он
только один раз
  End If
End Sub
```

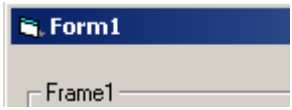
Задания для усовершенствования программы (если осталось время)**:**

1). Подсчитать и вывести в текстовое окно Text3 число переданных строк HEX-файла.
2). Уберите надписи Frame1, Frame2, Frame3 или запишите туда, что-нибудь свое.
3). При старте программы до нажатия на кнопку “Загрузить HEX...” в окнах Text2 и Text3 присутствуют одноименные строки не несущие никакой информации.

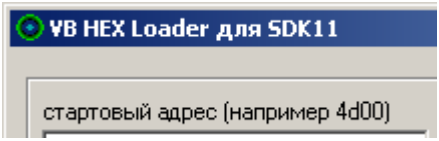


Подправьте программу, чтобы при запуске эти окна были пустыми.

4). Смените обезличенную иконку и название “Form1”



на следующие



Файл новой иконки находится в папке “C:\EMUL\Work\SDK_11\”. Для решения воспользуйтесь свойствами формы “Icon” и “Caption”.
5). Поместите на кнопку “Command1” рисунок, с помощью свойств “Style” и “Picture”.

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ И ЭКЗАМЕНА

1. Целевая программа. Назначение инструментального и резидентного загрузчиков.
2. Пояснить формат интеловского HEX – файла.
3. Дать комментарии ко всем строчкам инструментального загрузчика.

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики» на 2009–2018 годы.

КАФЕДРА ЭЛЕКТРОНИКИ

Заведующий кафедрой: д.т.н., проф. Г.Н. Лукьянов.

Кафедра Электроники (первоначальное название “Радиотехники”) была основана в 1945 году. Первым руководителем кафедры был С.И. Зилитинкевич известный в стране и за рубежом ученый в области физической электроники и радиотехники, активный работник высшей школы, заслуженный деятель науки и техники РСФСР, доктор технических наук, профессор ЛИТМО с 1938 г., инициатор создания в ЛИТМО инженерно-физического и радиотехнического факультетов (1946г.). С.И. Зилитинкевич заведовал кафедрой с 1945 до 1978 года. Под его научным руководством аспирантами и соискателями выполнено более 50 кандидатских диссертаций, многие его ученики стали докторами наук.

В дальнейшем, с 1978 г. по 1985 г. кафедру возглавил к.т.н., доцент Е.К. Алахов, один из учеников С.И. Зилитинкевича.

С 1985 г. по 2006 г. руководителем кафедры стал д.т.н., профессор В.В. Тогатов, известный специалист в области силовой

электроники и приборов для измерения параметров полупроводниковых структур.

Начиная с 2006 г. кафедрой заведует д.т.н., профессор Г.Н. Лукьянов, под руководством и при участии которого кардинально обновилось лабораторное оборудование в рамках инновационной программы развития.

Основные направления кафедры связаны с разработкой приборов для лазерной и медицинской техники, приборов для измерения параметров полупроводниковых структур, а также встраиваемых цифровых и микропроцессорных устройств.

Под руководством В.В. Тогатова было разработано и изготовлено большое число приборов различного назначения:

- Измеритель параметров ультрабыстрых диодов;
- Универсальное устройство для исследования переходных процессов в силовых полупроводниковых структурах;
- Измеритель времени жизни заряда в слаболегированных областях диодных, тиристорных и транзисторных структур;
- Универсальный разрядный модуль для накачки твердотельных лазеров;
- и ряд других.

На кафедре написаны и размещены на сайте ЦДО следующие материалы для дистанционного обучения (автор Ю.В. Китаев):

- Конспект лекций по дисциплине “Электроника и микропроцессорная техника”;
- свыше 600 вопросов к обучающим и аттестующим тестам;
- 18 дистанционных лабораторных и практических работ

На кафедре имеются следующие компьютеризированные учебные лаборатории:

- АРМС – полупроводниковые приборы;
- Устройства на полупроводниковых приборах;
- Цифровая техника;
- Микропроцессорная техника
- Моделирование электронных устройств.

Юрий Васильевич Китаев

ЛАБОРАТОРНАЯ УСТАНОВКА НА ОСНОВЕ ВИРТУАЛЬНЫХ ПРИБОРОВ И USB ИНТЕРФЕЙСА

Учебное пособие

В авторской редакции
Дизайн Китаев Ю.В.
Верстка Китаев Ю.В.
Редакционно-издательский отдел Санкт-Петербургского государственного университета информационных технологий, механики и оптики
Зав. РИО Н.Ф. Гусарова
Лицензия ИД 3 00408 от 05.11.99
Подписано к печати _____
Заказ № _____
Тираж 100 экз.
Отпечатано на ризографе

Редакционно-издательский отдел
Санкт-Петербургского национального
исследовательского университета
информационных технологий,
механики и оптики
197101, Санкт-Петербург,
Кронверкский пр., 49