БАЛТИЙСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ "ВОЕНМЕХ"

им. Д.Ф. УСТИНОВА

Отчет

О прохождении преддипломной практики

«Программная документация»

САНКТ-ПЕТЕРБУРГ, 2009 г

**Содержание**

Цель

Техническое задание

Общие сведения о предприятии

Общие сведения о разработанной программе

Алгоритм программы

Режимы работы программы

Результаты тестирования

Входные параметры

Заключение

Приложение 1 – Исходные тексты программы

# Цель

Целью прохождения производственной практики является: подготовка к написанию дипломной работы, сбор материалов к данной работе и получение практических навыков.

Практические навыки играют определяющую роль в профессиональной деятельности любого специалиста. Чем больший опыт накоплен человеком по практическому использованию своих теоретических знаний, тем эффективнее работа такого сотрудника.

Подготовка к написанию дипломной работы предусматривает изучение темы будущей работы, знакомство со всеми ее тонкостями и нюансами. Необходимо составить наиболее полное представление о предмете работы и хорошо ориентироваться в данном вопросе.

На этом этапе также ставится задача ознакомиться со спецификой предприятия, составить базу будущей работы, состоящую из собственных наблюдений, материалов и информации, используемых в работе организации.

# Техническое задание

Требуется разработать программу контроля версий компонент изделий и подготовить к выпуску программную документацию.

Программная документация должна быть выполнена в соответствии с ГОСТ 19.105-78 (ЕСПД. Общие требования к программным документам).

Комплект программного обеспечения должен содержать описание программы, текст программы и исполняемый модуль.

Программа предназначена для автоматизированной проверки версий всех компонентов изделия (модуля или прибора). Программа должна обеспечивать контроль версии, как отдельных модулей, так и приборов в различной комплектации.

Программа должна запускаться с помощью программы MonU «Монитор для сдачи работ» 643.0691.00249-01. Вывод информации должен осуществляться в окно монитора по каналу межпроцессного обмена. Программа взаимодействует с прибором через интерфейс RS-232. Программа может содержать часть, исполняемую на приборе, а может взаимодействовать со "Служебной программой для модуля М207" 643.0691.00255-01. Эталонные значения версий компонентов изделий должны храниться в текстовом файле в виде, удобном для чтения и редактирования в текстовом редакторе.

Общий алгоритм работы программы:

1. Программа запускается с помощью скрипта программы MonU. Изделие, версии компонентов которого требуется проверить, определяется параметрами командной строки.

2. Программа получает доступ к COM порту (или к иному интерфейсу связи с изделием).

3. Подключается к MonU используя канал межпроцессного обмена.

4. Открывает файл, содержащий эталонные версии компонентов изделий.

5. Читает версии из изделия и сравнивает с эталонными. В случае различия, версии выводятся с помощью программы MonU. В случае, если версии всех компонентов изделия совпали, программа выводит "Контроль версий произведён. Результат: НОРМА". В случае, если имели место расхождения: "Контроль версий произведён. Результат: НЕНОРМА".

6. Программа должна возвращать код результата. 0 – НОРМА, -1 – НЕНОРМА.

Входные параметры, пример:

ver\_verify.exe M207

ver\_verify.exe ПОТОК-3VSB X7

ver\_verify.exe БЦВМб134

ver\_verify.exe БС12

# 

# Общие сведения о предприятии

Акционерное общество «ГРАНИТ-ВТ» было образовано в 1992 году на базе коллектива сотрудников научно-исследовательского отдела ЦНИИ «ГРАНИТ».

Основным видом деятельности предприятия были и остаются разработка, производство и поддержка эксплуатации вычислительных и периферийных электронных модулей и систем на их основе. Основной сферой применения разрабатываемой аппаратуры являются системы, эксплуатируемые в тяжелых условиях, таких как широкий диапазон рабочих температур, повышенная влажность окружающей среды, прочие неблагоприятные климатические факторы, а также повышенные уровни механических воздействий.

Удачным примером работы компании может служить разработка в 1993-94гг. при содействии сотрудников российского представительства компании Intel® вычислительного модуля КРЕДО-486, используемого в настоящий момент в ряде железнодорожных и авиационных систем.

Отдельным этапом деятельности стала организация в 1994 г. совместно с АО «Гамма» (ныне ООО «Гамма Плюс») и при содействии представительства Intel® регионального центра поддержки разработок на основе элементов программируемой логики. В рамках этой деятельности более 40 предприятий (в основном – северо-западного региона России) были снабжены начальной версией САПР для разработки схем на основе программируемой логики, комплектом переводной литературы, возможностью программирования базовых PLD-схем, техническими консультациями. Со многими участниками работы регионального центра ЗАО «ГРАНИТ-ВТ» поддерживает технические контакты и коммерческую кооперацию и по сей день.

В активе предприятия:

- разработка, производство и поддержка в эксплуатации ряда совместимых комплектов электронных модулей на основе магистралей ISA, VME и CompactPCI;

- разработка локомотивной информационно-управляющей системы повышения безопасности движения для модернизации локомотивов Октябрьской железной дороги;

- участие в разработке комплексной бесплатформенной навигационной системы для модернизации авионики самолетов гражданской авиации;

- разработка поездной информационно-управляющей системы с функцией обеспечения автоведения для высокоскоростной магистрали Москва-Санкт-Петербург.

ЗАО «ГРАНИТ-ВТ» предлагает потребителю ряд вычислительных модулей общего назначения, модулей расширения памяти, интерфейсных и системозависимых модулей. Особенно рекомендуются модули расширения разработки ФГУП ЦНИИ «Гранит» и источники вторичного питания производства ООО «Авионика-Вист».

Помимо представленной серийно выпускаемой продукции предприятие радо предложить услуги в области проектирования и производства как электронных систем управления любой степени сложности, так и любых их составных частей – приборов, модулей, критичного программного обеспечения, контрольно-проверочной аппаратуры.

Можно заказать разработку специализированных модулей «под ключ» или любую составную часть такой разработки:

- разработку схемотехнических решений;

- проектирование логических функций в заданном базисе (программируемые логические микросхемы);

- разработку топологии печатных плат любой степени сложности с организацией при необходимости их производства;

- автоматический и ручной монтаж и ремонт электронных модулей практически без ограничений по типу корпусов;

- нанесение полипараксилиленового влагозащитного покрытия на изделия заказчика;

- разработку тестового и функционального программного обеспечения;

- разработку специализированного стендового оборудования.

За долгие годы разработки специальной управляющей техники коллектив ЗАО «ГРАНИТ-ВТ» приобрел глубокий опыт создания вычислительных модулей, приборов и систем, предназначенных для управления подвижными объектами в реальном времени в тяжелых эксплуатационных условиях (климатические и механические воздействия). Основная предлагаемая ЗАО «ГРАНИТ-ВТ» продукция, в первую очередь, флагманская линия вычислительных и периферийных модулей, предназначена именно для этого сегмента рынка. Сфера применения диктует основные технические принципы, которых, на наш взгляд, следует придерживаться при разработке высоконадежной аппаратуры для ответственных применений и которые являются основой технической политики ЗАО «ГРАНИТ-ВТ»:

- ориентация на промышленные стандарты в части общей вычислительной архитектуры, внутренних и внешних интерфейсов, конструкторских решений, общего программного обеспечения, стендового оборудования, технологических процессов;

- открытость технических решений, что подразумевает возможность, при необходимости, предоставления потребителю всей технической информации (схемотехническая и конструкторская документация, исходные тексты встроенного, инструментального и тестового программного обеспечения собственной разработки). Открытость документации позволяет пользователю провести сертификацию конечной продукции;

- использование элементной базы, удовлетворяющей условиям эксплуатации конечной аппаратуры. Как правило, это компоненты в индустриальном исполнении, в расширенном температурном исполнении, в отдельных обоснованных случаях – компоненты, выполненные по военным нормам;

- долгосрочная поддержка воспроизводства (возможность выпуска идентичной аппаратуры в течении длительного срока эксплуатации систем). Основывается на преимущественном использовании компонентов, обладающих долгосрочной поддержкой производителя (long-life support) или являющихся де-факто промышленным стандартом и выпускаемых несколькими производителями;

- минимизация энергопотребления за счет использования преимущественно микромощных компонентов, постепенного перехода на пониженные уровни напряжения электропитания, внутримодульного формирования нестандартных номиналов напряжений электропитания.

- обеспечение гарантированного теплосъема путем использования теплопроводящих прокладок, сплошных утолщенных металлизированных слоев в многослойных печатных платах, кондуктивного отвода тепла на несущие конструкции через металлические теплостоки и клиновые замки;

- использование влагозащитного покрытия на основе полипараксилилена, обеспечивающего отсутствие “воздушных подушек” под SMD-компонентами и, как следствие, отсутствие открытого конденсата на поверхности печатной платы и компонентах при термоциклировании;

- использование технических решений, гарантирующих требуемую устойчивость к механическим воздействиям, – использование преимущественно легких компонентов в корпусах для поверхностного монтажа, закрепление компонентов при необходимости, отсутствие механически нестойких элементов (панелей под микросхемы, соединителей-джамперов, кнопок, механических контакторов и пр.);

- использование ремонтопригодных технологий, обеспечение долгосрочного гарантийного и послегарантийного обслуживания;

- дифференцированный подход к встроенному системному и функциональному программному обеспечению: возможность разработки специализированных программных комплексов для ответственных критичных по времени применений в управляющих системах реального времени, обеспечивающих рестартуемость (восстановление прерванного хода вычислительного процесса) вычислительного процесса при кратковременных отказах электропитания и функциональных сбоях с сохранением интегральных величин; возможность разработки минимального программного обрамления для обеспечения программирования функциональных задач на языке высокого уровня; совместимость с универсальными ОС реального времени типа VxWorks™, QNX™, ОС2000 и др., подтвержденная опытными работами.

# 

# Общие сведения о разработанной программе

Наименование программы − “Программа проверки версий изделий”.

Обозначение исполняемого файла программы – Version-verifier.exe.

Программа написана на языке C++ – CBuilder версии 10.0 фирмы Borland Software Corporation для ОС Windows.

Программа производит проверку версий отдельного модуля или версий модулей входящих в состав прибора по усмотрению пользователя.

Программа располагается и исполняется на стендовой ЭВМ.

База эталонов версий создается и хранится в файлах “device\_list.xml” и “module\_list.xml”.

Вызов программы осуществляется запуском исполняемого модуля программы средствами операционной системы, либо из командной строки с параметрами командной строки.

У программы есть один параметр командной строки – наименование изделия.

Пример:

C:\Version-verifier.exe BS12

Или

C:\Version-verifier.exe M207

Через параметр командной строки в программу вводится наименование изделия, подлежащего проверке. Так же наименование изделия можно передать через скрипт “Сервисной программы монитор для сдачи работ” 643.0691.00249-01. Подробнее о скриптах “Сервисной программы монитор для сдачи работ” можно прочитать в руководстве программиста 643.0691.00249-01 33 01.

Информацию о модулях программа получает из XML-файлов “device\_list.xml” и “module\_list.xml”.

Поправки в работу программы можно внести с помощью файла “corrections.txt”.

Программа может быть запущена через “Сервисную программу монитор для сдачи работ” 643.0691.00249-01. В этом случае результаты работы программы будут выводиться в окне “Сервисной программы монитор для сдачи работ”. О запуске приложений через “Сервисную программу монитор для сдачи работ” можно прочитать в руководстве программиста 643.0691.00249-01 33 01.

# 

# Алгоритм программы

Параметром командной строки в программу передается наименование изделия, которое необходимо проверить.

Производится поиск наименования изделий в файле “device\_list.xml”. Сначала производится поиск среди тегов “device” (список приборов), если изделие не обнаружено, то поиск продолжается в теге “default” среди тегов “module”(список модулей).

Если изделие найдено, то программа получает тип изделия (прибор или модуль). Если изделие не найдено, то выводится сообщение об ошибке.

Если с входными параметрами было получено наименование модуля, то программа считывает из файла “device\_list.xml” базовый адрес модуля по умолчанию.

Затем из файла “module\_list.xml” считываются адреса регистров версий, значения версий и разрядность версий соответствующего модуля.

Затем программа считывает фактические значения версий модуля и сравнивает их со значениями, считанными из файла “module\_list.xml”. При совпадении этих значений программа выдает норму, при несовпадении – ненорму.

Если с входными параметрами было получено наименование прибора, то программа считывает из файла “device\_list.xml” базовые адреса всех модулей, входящих в состав прибора.

Затем из файла “module\_list.xml” считываются адреса регистров версий, значения версий и разрядность версий для каждого модуля, входящего в состав прибора.

Затем программа считывает фактические значения версий модулей, входящих в состав прибора, и сравнивает их со значениями, считанными из файла “module\_list.xml”. При совпадении этих значений для каждого модуля программа выдает норму, при несовпадении хотя бы для одного модуля – ненорму.

# Режимы работы программы

Если программа была запущена “Сервисной программой монитор для сдачи работ” 643.0691.00249-01, то она будет свернута, и вся информация будет выводиться в окне “Сервисной программы монитор для сдачи работ”. После выполнения проверки, программа будет завершена самостоятельно.

Если программа была запущена из командной строки независимо от “Сервисной программы монитор для сдачи работ”, то вся информация будет выводиться в текстовом поле окна программы. После выполнения проверки, программа останется открытой.

# Результаты тестирования

При проверке версий прибора фиксируется, результат проверки для каждого модуля, входящего в состав прибора. Если проверка всех модулей завершилась с результатом НОРМА, общий результат – НОРМА. Если проверка хотя бы одного модуля завершилась с результатом НЕНОРМА, то общий результат - НЕНОРМА.

При проверке версий отдельного модуля, результат НОРМА - если проверка завершилась успешно; и НЕНОРМА – если проверка завершилась неудачно.

**Входные параметры**

### Структура XML-файла “device\_list.xml”

Файл “device\_list.xml” состоит из тегов “device” и тега “default”.

Тег “device” имеет атрибут “name”, в котором содержится название прибора, соответствующего этому тегу. Внутри тегов “device” содержатся теги “module”, которые имеют атрибут “baseaddress”, содержащий базовый адрес модуля в приборе. Внутри тега “module” указывается название модуля.

Тег “default” содержит в себе теги “module”. Тег “module” имеет один атрибут “baseaddress”, в котором содержится базовый адрес модуля по умолчанию. Внутри тега “module” указывается название модуля.

### Структура XML-файла “module\_list.xml”

Файл “module\_list.xml” состоит из тегов “module”.

Тег “module” содержит в себе тег “name” и теги “version”. Тег “name” содержит в себе название модуля. Тег “version” имеет один атрибут “description”, в котором содержится информация о версии, описываемой соответствующим тегом “version”. Тег “version” содержит в себе теги “address” (адрес регистра версии модуля), “value” (значение версии модуля) и “type” (разрядность версии модуля).

При выпуске новой версии модуля, тег “name” в соответствующем теге “module” дополняется датой выпуска следующей версии. Затем создается новый тег “module” с наименованием модуля и обновленными значениями версий в теге “version”.

### Структура TXT-файла “corrections.txt”

Если есть необходимость проверить прибор с более ранними версиями каких-либо модулей, входящих в его состав, составляется файл поправок “corrections.txt”. Структура файла “corrections.txt” по п. 0.

Если в папке с программой содержится файл “corrections.txt”, то программа производит обработку этого файла.

В процессе обработки заполняется массив поправок, содержащих два наименования модулей: с поздней версией и с более ранней версией.

Файл “corrections.txt” предназначен для внесения поправок в работу программы.

Файл состоит из записей вида:

<name>=<new\_name>;

Здесь “name” – наименование модуля, который необходимо проверить с особыми значениями версий; “new\_name” – наименование модуля в файле “module\_list.txt” с особыми значениями версий.

# Заключение

В ходе преддипломной практики (в период с 15 декабря 2008 по 15 февраля 2009 г.) мною были выполнены поставленные предварительно задачи:

- получены практические навыки работы в коллективе инженеров,

- разработана программа проверки версий компонент изделий,

- подготовлена к выпуску программная документация.

# Приложение 1 – Исходные тексты программы

//--------------------------------------------------------------

#ifndef Unit2H

#define Unit2H

//--------------------------------------------------------------#include <Classes.hpp>

//--------------------------------------------------------------class Executing : public TThread

{

private:

protected:

void \_\_fastcall Execute();

public:

bool end;

\_\_fastcall Executing(bool CreateSuspended);

};

//--------------------------------------------------------------#endif

//--------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include <stdio.h>

#include "ProgExchCl\_b.h"

#include "Udevices.h"

#define SecondLevelNodeAccess Form1->XMLDocument1->DocumentElement->ChildNodes

#define ThirdLevelNodeAccess(a) Form1->XMLDocument1->DocumentElement->ChildNodes->GetNode(a)->ChildNodes

#define FourthLevelNodeAccess(a, b) Form1->XMLDocument1->DocumentElement->ChildNodes->GetNode(a)->ChildNodes->GetNode(b)->ChildNodes

char received\_data[255];

PModule Module; // массив модулей

PCorrection Correction; // массив исправлений

HANDLE m\_Disp = NULL; // для монитора

int wrtd, // для функции mWrite

cor\_count; // количество исправлений

TCom\* ComPort; // порт

//--------------------------------------------------------------

void \_\_fastcall msg(char \*str)//сообщение для монитора

{

if(!mWrite(m\_Disp, (AnsiString(str)+AnsiString("\n\r")).c\_str(), strlen(str)+2, &wrtd));

}

void \_\_fastcall logmsg(AnsiString mes)//сообщение в поле мемо

{

Form1->LogText->Lines->Add(mes);

}

//--------------------------------------------------------------char \_\_fastcall DeviceOrModule(AnsiString Parameter, int\* item\_index) // определить что было передано с начальными параметрами: прибор или модуль и выдать порядковый номер нужного тега.

{

int i, all\_nodes;

// ищем прибор

all\_nodes = SecondLevelNodeAccess->GetCount(); // количество тегов второго уровня

for (i = 0; i < all\_nodes; i++) {

if(AnsiString(SecondLevelNodeAccess->GetNode(i)->Attributes[WideString("name")] ) == Parameter){

\*item\_index = i;

return 1;

}

}

// если прибор не найден, ищем модуль в теге default

all\_nodes = SecondLevelNodeAccess->FindNode("default")->ChildNodes->GetCount(); // количество тегов третьего уровня в теге default

for (i = 0; i < all\_nodes; i++) {

if(AnsiString(SecondLevelNodeAccess->FindNode("default")->ChildNodes->GetNode(i)->Text) == Parameter){

\*item\_index = i;

return 2;

}

}

return 0;

}

//--------------------------------------------------------------

void \_\_fastcall GetModuleList(AnsiString Parameter, PModule\* ModuleList, int item\_index, int\* modules\_counter) // получение списка модулей входящих в состав прибора

{

int i, j, k, p, all\_modules, version\_count;

AnsiString mes;

\*modules\_counter = ThirdLevelNodeAccess(item\_index)->GetCount(); // количество модулей в приборе

\*ModuleList = new TModule[\*modules\_counter]; // выделяем память под массив модулей

for (i = 0; i < \*modules\_counter; i++) { // заполняем поля name и baseaddress

if(cor\_count)

{

for(j = 0; j < cor\_count; j++) {

if (AnsiString(ThirdLevelNodeAccess(item\_index)->GetNode(i)->GetNodeValue()) == Correction[j].name) {

(\*ModuleList)[i].name = Correction[j].new\_name;

break;

}

else (\*ModuleList)[i].name = AnsiString(ThirdLevelNodeAccess(item\_index)->GetNode(i)->GetNodeValue());

}

}

else (\*ModuleList)[i].name = AnsiString(ThirdLevelNodeAccess(item\_index)->GetNode(i)->GetNodeValue());

(\*ModuleList)[i].baseaddress = StrToInt(ThirdLevelNodeAccess(item\_index)->GetNode(i)->Attributes[WideString("baseaddress")]);

}

// LOG->Add(AnsiString());

// активизация dll со списком модулей

Form1->XMLDocument1->FileName = "module\_list.xml";

Form1->XMLDocument1->Active = true;

all\_modules = SecondLevelNodeAccess->GetCount(); // сколько всего модулей в базе

for (i = 0; i < \*modules\_counter; i++) {

for (j = 0; j <= all\_modules; j++) { // заполняем поля version

if (j >= all\_modules) { // не нашли модуль (ошибка в файле с исправлениями)

mes = AnsiString("Не найден модуль ") + (\*ModuleList)[i].name;

MessageBox(NULL, mes.c\_str() , "Ошибка", MB\_ICONERROR | MB\_OK );

break;

}

if ((\*ModuleList)[i].name == AnsiString(ThirdLevelNodeAccess(j)->GetNode(0)->GetNodeValue())) {

(\*ModuleList)[i].version\_count = ThirdLevelNodeAccess(j)->GetCount() - 1;

for (k = 0; k < (\*ModuleList)[i].version\_count; k++) {

(\*ModuleList)[i].version[k] = new TVersion;

(\*ModuleList)[i].version[k]->address = StrToInt(FourthLevelNodeAccess(j, k+1)->GetNode(0)->NodeValue);

(\*ModuleList)[i].version[k]->value = StrToInt(FourthLevelNodeAccess(j, k+1)->GetNode(1)->NodeValue);

(\*ModuleList)[i].version[k]->type = StrToInt(FourthLevelNodeAccess(j, k+1)->GetNode(2)->NodeValue);

(\*ModuleList)[i].version[k]->description = AnsiString(ThirdLevelNodeAccess(j)->GetNode(k+1)->Attributes[WideString("description")]);

}

break;

}

}

// LOG->Add(ModuleList[i].name + AnsiString(" ") + ModuleList[i].baseaddress);

}

}

//--------------------------------------------------------------

void \_\_fastcall GetModule(AnsiString Parameter, PModule\* ModuleList, int item\_index) // получаем информацию для одного модуля

{

int i, j, all\_modules;

\*ModuleList = new (TModule);

(\*ModuleList)[0].name = AnsiString(Form1->XMLDocument1->DocumentElement->ChildNodes->FindNode("default")->ChildNodes->GetNode(item\_index)->GetNodeValue());

(\*ModuleList)[0].baseaddress = StrToInt(Form1->XMLDocument1->DocumentElement->ChildNodes->FindNode("default")->ChildNodes->GetNode(item\_index)->Attributes[WideString("baseaddress")]);

Form1->XMLDocument1->FileName = "module\_list.xml";

Form1->XMLDocument1->Active = true;

all\_modules = SecondLevelNodeAccess->GetCount();

for (i = 0; i <= all\_modules; i++) {

if (i >= all\_modules) {

MessageBox(NULL, "Модуль не был найден. Версия недоступна", "Error", MB\_ICONERROR | MB\_OK );

break;

}

if ((\*ModuleList)[0].name == AnsiString(ThirdLevelNodeAccess(i)->GetNode(0)->GetNodeValue())) {

(\*ModuleList)[0].version\_count = ThirdLevelNodeAccess(i)->GetCount() - 1;

for (j = 0; j < (\*ModuleList)[0].version\_count; j++) {

(\*ModuleList)[0].version[j] = new TVersion;

(\*ModuleList)[0].version[j]->address = StrToInt(FourthLevelNodeAccess(i, j+1)->GetNode(0)->NodeValue);

(\*ModuleList)[0].version[j]->value = StrToInt(FourthLevelNodeAccess(i, j+1)->GetNode(1)->NodeValue);

(\*ModuleList)[0].version[j]->type = StrToInt(FourthLevelNodeAccess(i, j+1)->GetNode(2)->NodeValue);

(\*ModuleList)[0].version[j]->description = AnsiString(ThirdLevelNodeAccess(i)->GetNode(j+1)->Attributes[WideString("description")]);

}

break;

}

}

/\* LOG->Add((\*ModuleList)[0].name + AnsiString(" ") + (\*ModuleList)[0].baseaddress);

for (i = 0; i < (\*ModuleList)[0].version\_count; i++) {

LOG->Add((\*ModuleList)[0].version[i]->address);

LOG->Add((\*ModuleList)[0].version[i]->value);

LOG->Add((\*ModuleList)[0].version[i]->type);

} \*/

}

//--------------------------------------------------------------

unsigned int \_fastcall get\_version(unsigned int BaseAddress, unsigned int type) // считываем версию с модуля

{

unsigned short i, j;

unsigned short SHsum; // контрольная сумма

unsigned char\* pTarget = new unsigned char[10];

unsigned char\* pReceived;

unsigned short MessageSize = 6;

unsigned int version;

ComPort = new TCom("COM2", 115200);

i = ComPort->link\_on();

if(i) {

MessageBox(NULL, AnsiString(i).c\_str(), "Ошибка инициализации порта", MB\_OK);

return 0;

}

pTarget[0]= 0x31;

pTarget[1]=(unsigned char)(MessageSize); //младший байт

pTarget[2]=(unsigned char)(MessageSize>>8); //старший байт

pTarget[3] = 0x04;

pTarget[4] = (unsigned char) type;

\*(int\*)(pTarget + 0x05) = BaseAddress;

version = 0;

SHsum=0x0;

for ( i=3; i<=8; i++ ){SHsum+=pTarget[i];} //подсчёт КС

// MessageBox(NULL, AnsiString(SHsum).c\_str(), "Контрольная сумма", MB\_OK);

pTarget[9]=(unsigned char)(SHsum);//младший байт

pTarget[10]=(unsigned char)(SHsum>>8);//старший байт

ComPort->write(pTarget, 3+MessageSize+2);

delete [] pTarget;

Sleep(100);

i = ComPort->read();

if(i>0)

{

// MessageBox(NULL, AnsiString(i).c\_str(), "Принято байт", MB\_OK);

pReceived = ComPort->get\_receiver\_ptr(); //считываем полученное значение

if ( pReceived[0] != 0x55 || pReceived[1] != 0xAA) {

MessageBox(NULL, pReceived, "Ошибка", MB\_OK);

return 0;

}

// наложение масок

switch (type) {

case 1:

version = (unsigned int) (\*( char\* )(pReceived + 5))&0x000000FF;

break;

case 2:

version = (unsigned int) (\*( short\* )(pReceived + 5))&0x0000FFFF;

break;

case 4:

version = (unsigned int) \*( int\* )(pReceived + 5);

break;

default:

version = (unsigned int) (\*( short\* )(pReceived + 5))&0x0000FFFF;

break;

}

}

ComPort->link\_off();

return version;

}

\_\_fastcall Executing::Executing(bool CreateSuspended)// конструктор

: TThread(CreateSuspended)

{

end = false; // метка окончания = ложно

}

//--------------------------------------------------------------

int \_\_fastcall get\_correction(PCorrection\* CorrectionList) // обработка файла исправлений

{

int i = 0;

int j = 0;

int n = 0;

int correction\_counter = 0; // количество исправлений

FILE\* CorrectFilePtr; // файл с исправлениями

char tempchar = ' ';

char\* tempFile; //массив под файл с сиправлениями

char control\_buff[20]; // массив знаков препинания

if ((CorrectFilePtr = fopen("corrections.txt","r")) == NULL) {

return 0;

}

// узнаем размер файла и выделяем временный массив под файл

fseek(CorrectFilePtr,0L,SEEK\_END);

n = ftell(CorrectFilePtr);

fseek(CorrectFilePtr,0L,SEEK\_SET);

tempFile = new char[n];

//исправляем возможные ошибки пунктуации

while(tempchar == '\n' || tempchar == ' '){

tempchar = fgetc(CorrectFilePtr);

}

tempFile[0] = tempchar;

i = 1;

while (1){

tempchar = fgetc(CorrectFilePtr);

if(tempchar == '\n'){

continue;

}

if(tempchar == ' '){

continue;

}

if(tempchar == '='){

control\_buff[j] = tempchar;

j++;

}

if(tempchar == ';'){

control\_buff[j] = tempchar;

correction\_counter++;

j++;

}

if(tempchar == EOF){

tempFile[i] = tempchar;

control\_buff[j] = tempchar;

break;

}

tempFile[i] = tempchar;

i++;

}

//правильно ли расставлены знаки припенания

for (i = 0; i <= j; i++){

if(control\_buff[i] == '='){

if(control\_buff[i+1] != ';'){

logmsg("В файле корректировки обнаружена ошибка. Корректировки учтены не будут33");

return 0;

}

}

if(control\_buff[i] == ';'){

if(!(control\_buff[i+1] == '=' || control\_buff[i+1] == EOF)){

logmsg("В файле корректировки обнаружена ошибка. Корректировки учтены не будут331");

return 0;

}

}

}

// MessageBox(NULL, tempFile,"Пpочитано",MB\_OK);

\*CorrectionList = new TCorrection[correction\_counter];

//заполянем массив исправлений

tempchar = 0;

i = 0;

n = 0;

while (tempchar != EOF) {

while(tempFile[i] != '=') {

(\*CorrectionList)[n].name += tempFile[i++];

}

// (\*CorrectionList)[n].name[j] = '\0';

i++;

while(tempFile[i] != ';') {

(\*CorrectionList)[n].new\_name += tempFile[i++];

}

// (\*CorrectionList)[n].new\_name[j] = '\0';

n++;

tempchar = tempFile[++i];

}

/\* for (j = 0; j < correction\_counter; j++) {

MessageBox(NULL, (\*CorrectionList)[j].name,"",MB\_OK);

MessageBox(NULL, (\*CorrectionList)[j].new\_name,"",MB\_OK);

} \*/

delete [] tempFile;

fclose(CorrectFilePtr);

return correction\_counter;

}

//--------------------------------------------------------------void \_\_fastcall Executing::Execute()

{

CoInitialize(NULL);

char item\_type;

DWORD res;

char received\_data[255];

unsigned int received\_version = 0;

AnsiString mes;

int i, j, index, modules\_counter, all\_modules, p, param\_num;

int error = 0;

if(!FileExists("device\_list.xml")){

MessageBox(NULL, "Файл device\_list.xml не был найден ", "Ошибка", MB\_ICONERROR | MB\_OK );

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

if(!FileExists("module\_list.xml")){

MessageBox(NULL, "Файл module\_list.xml не был найден ", "Ошибка", MB\_ICONERROR | MB\_OK );

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

// обработка входных параметров

if (ParamCount() == 0) {

MessageBox(NULL, "Не было передано ни одного параметра!", "Ошибка", MB\_ICONERROR | MB\_OK );

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

// получить управление над функциями библиотеки ProgExchCl.dll

res = ProgExchCl\_LoadLibrary();

if(res != 0x0)

{

sprintf(received\_data, "Ошибка %d при обращении к библиотеке \"ProgExchCl.dll\".", res);

MessageBox(NULL, received\_data, "Ошибка", MB\_OK);

}

//открытие канала выдачи данных на экран

sprintf(received\_data, "PipeDisplay%d", GetCurrentProcessId());

res = mOpen(received\_data, &m\_Disp);

if(res == 0x0)

{

// перевести вывод в режим отладочной (0x01 0x02) или отчетной (0x01 0x01) информации

received\_data[0] = 0x01; received\_data[1] = 0x01;

res = mWrite(m\_Disp, received\_data, 2, &wrtd);

if(res != 0x0)

{

sprintf(received\_data, "Ошибка %d при переводе вывода в режим отладочной информации", res);

MessageBox(NULL, received\_data, "Ошибка", MB\_OK);

logmsg(received\_data);

}

Form1->WindowState = wsMinimized;

}

else {

sprintf(received\_data, "Ошибка %d при открытии канала выдачи данных на экран", res);

logmsg(AnsiString(received\_data));

logmsg(AnsiString("Режим автономной работы"));

Form1->WindowState = wsNormal;

}

cor\_count = get\_correction(&Correction);

// открытие XML-файла

Form1->XMLDocument1->FileName = "device\_list.xml";

Form1->XMLDocument1->Active = true;

item\_type = DeviceOrModule(ParamStr(1), &index);

// получение списка модулей входящих в состав прибора

if (item\_type == 0){

MessageBox(NULL, "Прибор или модуль не был найден", "Ошибка", MB\_ICONERROR | MB\_OK );

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

if (item\_type == 1) {

GetModuleList(ParamStr(1), &Module, index, &modules\_counter);

if(index == -1) {

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

mes = AnsiString("Проверка версии прибора: ") + ParamStr(1);

}

if (item\_type == 2) {

GetModule(ParamStr(1), &Module, index);

if(index == -1) {

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

return;

}

mes = AnsiString("Проверка версии модуля: ") + ParamStr(1);

modules\_counter = 1;

}

msg(" ");

msg(" ");

msg(mes.c\_str());

logmsg(mes.c\_str());

// перевести вывод в режим отладочной (0x01 0x02) или отчетной (0x01 0x01) информации

received\_data[0] = 0x01; received\_data[1] = 0x02;

res = mWrite(m\_Disp, received\_data, 2, &wrtd);

if(res != 0x0)

{

// sprintf(received\_data, "Ошибка %d при переводе вывода в режим отладочной информации", res);

// MessageBox(NULL, received\_data, "Ошибка", MB\_OK);

}

// вывод результатов тестирования по каждому модулю

for (i = 0; i < modules\_counter; i++) {

// название модуля

msg(" ");

msg(Module[i].name.c\_str());

logmsg(" ");

logmsg(Module[i].name.c\_str());

for (j = 0; j < Module[i].version\_count ; j++) {

received\_version = get\_version( (unsigned int) (Module[i].baseaddress + Module[i].version[j]->address), Module[i].version[j]->type);

// если версии совпадают, то НОРМА, если не совпадают, то НЕНОРМА

if(received\_version == Module[i].version[j]->value){

mes =AnsiString(" ") + Module[i].version[j]->description + AnsiString(" 0x") + IntToHex((int) Module[i].version[j]->value, Module[i].version[j]->type) + AnsiString(" :HOPMA ");

msg(mes.c\_str());

logmsg(mes.c\_str());

}

else {

mes =AnsiString(" ") + Module[i].version[j]->description + AnsiString(" 0x") + IntToHex((int) received\_version, Module[i].version[j]->type) + AnsiString(" :HEHOPMA ") + AnsiString("(должно быть 0x") + IntToHex((int) Module[i].version[j]->value, Module[i].version[j]->type) + AnsiString(")");

msg(mes.c\_str());

logmsg(mes.c\_str());

error+=1;

}

}

}

// перевести вывод в режим отладочной (0x01 0x02) или отчетной (0x01 0x01) информации

received\_data[0] = 0x01; received\_data[1] = 0x01;

res = mWrite(m\_Disp, received\_data, 2, &wrtd);

if(res != 0x0)

{

// sprintf(received\_data, "Ошибка %d при переводе вывода в режим отчетной информации", res);

// MessageBox(NULL, received\_data, "Ошибка", MB\_OK);

}

// общий результат тестирования

if(error) {

msg(" ");

msg("Результат: НЕНОРМА");

logmsg(" ");

logmsg("Результат: НЕНОРМА");

}

else {

msg(" ");

msg("Результат: НОРМА");

logmsg(" ");

logmsg("Результат: НОРМА");

}

// освобождаем динамически выделенную память

if (item\_type == 1) {

delete [] Module;

}

else delete Module;

delete [] Correction;

if(mClose != NULL)

{

// отключить каналы обмена

mClose(m\_Disp);

}

ProgExchCl\_FreeLibrary();

CoUninitialize();

PostMessage(Form1->Handle, WM\_CLOSE, 0,0);

}

//--------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//--------------------------------------------------------------#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <msxmldom.hpp>

#include <XMLDoc.hpp>

#include <xmldom.hpp>

#include <XMLIntf.hpp>

//--------------------------------------------------------------

typedef struct \_SVersion{

unsigned int address; //адрес регистра версии

unsigned int value; //считанное значение версии

unsigned int type; //разрядность версии

AnsiString description;//описание версии

} TVersion,\* PVersion;

typedef struct \_SModule{

AnsiString name; //имя модуля

unsigned int baseaddress; //базовый адрес модуля

int version\_count;//количество версий у модуля

PVersion version[10]; //массив версий модуля

} TModule,\* PModule;

typedef struct \_SCorrection{

AnsiString name;

AnsiString new\_name;

} TCorrection,\* PCorrection;

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TXMLDocument \*XMLDocument1;

TMemo \*LogText;

void \_\_fastcall FormActivate(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//--------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//--------------------------------------------------------------

#endif

//--------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include <stdio.h>

//--------------------------------------------------------------#pragma package(smart\_init)

#pragma resource "\*.dfm"

Executing \*Thread;

TForm1 \*Form1;

tagMSG lpMsg;

//--------------------------------------------------------------\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::FormActivate(TObject \*Sender)

{

Thread = new Executing(FALSE); // запускаем поток

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction &Action)

{

Thread->WaitFor();

delete (Thread);

}

//--------------------------------------------------------------

//---------------------------------------------------------------------------

#ifndef UdevicesH

#define UdevicesH

#include <classes.hpp>

//#include "U\_MK\_pipe.h"

#include <Mask.hpp>

//--------------------------------------------------------------// Абстрактный класс, реализующий базовые функции любого устройства

class TDevFather: public TObject

{

\_\_published:

private: // User declarations

protected:

bool active;

char receiver[0x1000];//априорно выделено 4кб буфера, приготовлены для отправки классу монитору

int receiver\_size;//размер реально принятых байт в буфере

public:

virtual \_\_fastcall TDevFather();

virtual \_\_fastcall ~TDevFather();

virtual int \_\_fastcall link\_on();

virtual int \_\_fastcall link\_off();

//функции для работы с мостом в целях ввода/вывода

//чтение из буфера устройтва

virtual int \_\_fastcall scan();//Возвращает признак состояния устройства

virtual int \_\_fastcall read();//Если положительное число, то это число пинятых байт, если отрицательное, то это код ошибки

virtual char\* \_\_fastcall get\_receiver\_ptr();//

//отправить данные

virtual int \_\_fastcall write(char\* chs, int size);//передача символов/байтов на устройство

//запись в буфер

//virtual int \_\_fastcall add\_to\_sender(char\* chs, int size);//передача символов/байтов в передающий буфер

};

//--------------------------------------------------------------

class TCom: public TDevFather

{

\_\_published:

private: // User declarations

protected:

AnsiString comname;

int baudrate;

HANDLE com\_hndl;

DCB st\_dcb;

public:

virtual \_\_fastcall TCom(char\* comname, int baudrate);

virtual \_\_fastcall ~TCom();

int \_\_fastcall link\_on();

int \_\_fastcall link\_off();

//функции для работы с мостом в целях ввода/вывода

//чтение из буфера устройтва

int \_\_fastcall scan();//Возвращает признак состояния устройства

int \_\_fastcall read();//Если положительное число, то это число пинятых байт, если отрицательное, то это код ошибки

//virtual char\* \_\_fastcall get\_receiver\_ptr();// метод наследуется без изменения

//отправить данные

virtual int \_\_fastcall write(char\* chs, int size);//передача символов/байтов на устройство

//запись в буфер

//virtual int \_\_fastcall add\_to\_sender(char\* chs, int size);//передача символов/байтов в передающий буфер

void \_\_fastcall set\_comname(AnsiString comname);

void \_\_fastcall set\_baudrate(int baudrate);

AnsiString \_\_fastcall get\_comname();

int \_\_fastcall get\_baudrate();

};

//--------------------------------------------------------------#endif

//--------------------------------------------------------------

#pragma hdrstop

#include <Forms.hpp>

#include <Mask.hpp>

#include "Udevices.h"

//--------------------------------------------------------------\_\_fastcall TDevFather::TDevFather(){}

//--------------------------------------------------------------\_\_fastcall TDevFather::~TDevFather(){}

//--------------------------------------------------------------int \_\_fastcall TDevFather::link\_on()

{ return 0; }

//--------------------------------------------------------------int \_\_fastcall TDevFather::link\_off()

{ return 0; }

//-------------------------------------------------------------- //функции для работы с мостом

int \_\_fastcall TDevFather::scan()//Возвращает признак состояния устройства

{

//

return receiver\_size;

}

//---------------------------------------------------------------------------

int \_\_fastcall TDevFather::read()//Если положительное число, то это число принятых байт, если отрицательное, то это код ошибки

{

//

return receiver\_size;

}

//---------------------------------------------------------------------------

char\* \_\_fastcall TDevFather::get\_receiver\_ptr()//

{

return receiver;

}

//---------------------------------------------------------------------------

int \_\_fastcall TDevFather::write(char\* chs, int size)

{

return size;

}

//---------------TCom-------------------------------------------\_\_fastcall TCom::TCom(char\* comname, int baudrate)

{

//

TCom::comname=comname; //="COM1";

TCom::baudrate=baudrate; //=115200;

active=false;

}

//--------------------------------------------------------------\_\_fastcall TCom::~TCom()

{

//

if(active)

{

link\_off();

}

}

//--------------------------------------------------------------int \_\_fastcall TCom::link\_on()

{

//

if (!active)

{

com\_hndl=CreateFile(comname.c\_str(), GENERIC\_READ | GENERIC\_WRITE , 0, NULL, OPEN\_EXISTING, 0, 0);

if (com\_hndl == INVALID\_HANDLE\_VALUE)

{

//MessageBoxA(0,"Invalid handle value","Simple\_COM", MB\_OK);

active=false;

return -1;

}

//настройка параметров соединения

st\_dcb.DCBlength= sizeof(DCB);

st\_dcb.Parity=NOPARITY; //NOPARITY, ODDPARITY, MARKPARITY , EVENPARITY

st\_dcb.StopBits=ONESTOPBIT; //ONESTOPBIT , ONE5STOPBITS , TWOSTOPBITS

st\_dcb.ByteSize=8; //4-8

st\_dcb.BaudRate=baudrate; //CBR\_115200

SetCommState(com\_hndl, &st\_dcb);

if ( com\_hndl == INVALID\_HANDLE\_VALUE)

{

//MessageBoxA(0,"Invalid handle value","Simple\_COM", MB\_OK);

active=false;

return -1;

}

SetupComm(com\_hndl, 40000, 40000);//размеры банков

if (com\_hndl == INVALID\_HANDLE\_VALUE)

{

//MessageBoxA(0,"Invalid handle value","Simple\_COM", MB\_OK);

active=false;

return -1;

}

SetCommMask(com\_hndl,EV\_RXCHAR);

if (com\_hndl == INVALID\_HANDLE\_VALUE)

{

//MessageBoxA(0,"Invalid handle value","Simple\_COM", MB\_OK);

active=false;

return -1;

}

active=true;

};

if (!active)

{

return -1;

}else{

return 0;

}

}

//--------------------------------------------------------------int \_\_fastcall TCom::link\_off()

{

//

if (active)

{

active=false;

CloseHandle(com\_hndl);

return 0;

}

else

{

active=false;

return -1;

}

}

//--------------------------------------------------------------int \_\_fastcall TCom::scan()//Возвращает признак состояния устройства

{

COMSTAT st;

unsigned long Err;

if (active)

{

if ( ClearCommError(com\_hndl, &Err, &st) )

{

return st.cbInQue;

}

else

{

//MessageBoxA(0,"Com scan error","Simple\_COM", MB\_OK);

return -1;

}

}

else

{

return 0;

}

}

//--------------------------------------------------------------int \_\_fastcall TCom::read()//Если положительное число, то это число пинятых байт, если отрицательное, то это код ошибки

{

COMSTAT st;

unsigned long Err;

unsigned long readed\_bytes;

//

readed\_bytes=0;

if (active)

{

if ( ClearCommError(com\_hndl, &Err, &st) )

{

if ( (Err & CE\_BREAK)>0 )

{

//MessageBoxA(0,"Com read error : CE\_BREAK","Simple\_COM", MB\_OK);

return -2;

}

if ( 0x1000>st.cbInQue )

{

if (! ReadFile(com\_hndl, receiver, st.cbInQue, &readed\_bytes, NULL) )

{

//MessageBoxA(0,"Com read error : ReadFile 1","Simple\_COM", MB\_OK);

return -3;

}

}

else

{

if (! ReadFile(com\_hndl, receiver, 0x1000, &readed\_bytes, NULL) )

{

//MessageBoxA(0,"Com read error : ReadFile 2","Simple\_COM", MB\_OK);

return -4;

}

}

}

else

{

//MessageBoxA(0,"0шибка сканирования устройства COM","Simple\_COM", MB\_OK);

}

}

return (int)readed\_bytes;

}

//--------------------------------------------------------------int \_\_fastcall TCom::write(char\* chs, int size)

{

unsigned long sended;

//

sended=0;

if (! WriteFile(com\_hndl, chs, (unsigned long)size, &sended, NULL) )

{//ошибка

//MessageBoxA(0, "Ошибка передачи данных", "Simple\_COM", MB\_OK);

return -1;

}

return (int)sended;

}

//--------------------------------------------------------------

void \_\_fastcall TCom::set\_comname(AnsiString comname){}

void \_\_fastcall TCom::set\_baudrate(int baudrate){}

AnsiString \_\_fastcall TCom::get\_comname()

{//

return comname;

}

//--------------------------------------------------------------int \_\_fastcall TCom::get\_baudrate()

{//

return baudrate;

}