Astroma Ltd.

LOCK SDK v1.1

Система защиты программного продукта

Руководство разработчика

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Сегодня разработчикам программного обеспечения предлагаются десятки, а может быть и сотни средств и способов защитить свои авторские права. Однако, как показывает практика, абсолютной защиты до сих пор не существует, дешевые нелегальные копии программного продукта любого уровня сложности и востребованности продолжают появляться на пиратских рынках, а зачастую даже раньше выхода в тираж лицензионных копий. Система «LOCK» на сегодняшний день является самым надежным и прогрессивным способом защиты программного продукта от несанкционированного использования и распространения. Помимо этого, «LOCK» предоставляет практически неограниченную гибкость в построении и конфигурировании сложных систем защиты программного продукта. Несмотря на то, что система «LOCK» в качестве основной задачи предполагает защиту именно функциональных модулей программного обеспечения, она так же способна решать все задачи по защите информации, которые предлагаются другими производителями аппаратных средств защиты, при этом имея превосходство в производительности и универсальности.

2. АППАРАТНАЯ РЕАЛИЗАЦИЯ

USB-ключи «LOCK» представляют собой интеллектуальные модули, предназначенные для решения широкого круга задач. На сегодняшний день линейка USB-ключей «LOCK» включает в себя три модели устройств, отличающихся друг от друга объемом оперативной и FLASH-памяти, что определяется типом установленного внутри ключа микроконтроллера. Внешний вид устройств представлен на ***рис. 1***. Функциональных и внешних отличий между моделями USB-ключей не существует.

рис. 1. USB-ключи «LOCK»

2.1. Технические характеристики USB-ключей

Параметры	USB-ARM-64K	USB-ARM-128K	USB-ARM-256K
Процессорное ядро	ARM7TDMI	ARM7TDMI	ARM7TDMI
Разрядность ядра	32	32	32
Производительность (MIPS)	48 (до 55)	48 (до 55)	48 (до 55)
Объем FLASH-памяти (KB)	64	128	256
Объем оперативной памяти (KB)	16	32	64
Интерфейс подключения	USB 2.0	USB 2.0	USB 2.0
Режим интерфейса	FS Bulk	FS Bulk	FS Bulk
Скорость обмена данными (Mb/s)	12	12	12
Напряжение питания (от USB) (V)	5	5	5
Потребляемый ток (mA), не более	50	50	50
Габаритные размеры (mm)	60×18×8	60×18×8	60×18×8

3. ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ «LOCK»

Как уже говорилось выше, основной задачей системы защиты программного обеспечения “LOCK” является предотвращение создания нелегальных копий программного продукта с целью защиты авторских прав разработчиков. Очевидно, что программный продукт может быть скопирован или использован только при наличии полной его копии. В противном случае он будет частично или полностью неработоспособным, либо в нем будет отсутствовать часть функций.

Данная система защиты основана на использовании специального устройства (далее ключа), в память которого разработчик программного обеспечения самостоятельно помещает часть своей программы. Содержимое памяти ключа не может быть прочитано, что означает невозможность создания нелегальной копии устройства. Если ключ отсутствует, программное обеспечение не может функционировать, так как нарушена его целостность. Это равносильно тому, что Вы будете пытаться работать с программой, в середине которой вырезано несколько десятков или сотен килобайтов машинного кода.

3.1. Защита нескольких приложений одним ключом

Очень часто возникает необходимость защитить одним ключом несколько программ. Так, например, разрабатываемое программное обеспечение может состоять из нескольких отдельных модулей, которые способны функционировать независимо друг от друга. Для обеспечения их защиты целесообразно в каждом модуле выбрать по крайней мере один функциональный фрагмент, который в дальнейшем будет помещен в память USB-ключа «LOCK». Далее все выбранные фрагменты следует объединить в один общий модуль, который в дальнейшем будет “скрытым” в памяти ключа. При организации запросов к ключу в этом случае целесообразно указывать, какая именно функция вызывается, то есть передавать в потоке входных данных код функции, которую требуется выполнить. В свою очередь ключ, принимая данные должен уметь распознавать, к какой именно функции, хранящейся в его памяти, они относятся. Пример вызова различных функций приведен в листинге 1.

Листинг 1.

```
//-------------------------------------------------------------
void func0() {...}
void func1() {...}
void func2() {...}
void func3() {...}

void user_function()
{

unsigned int prog_index;

...

buffer=(unsigned char*)*pInAddrPtr;
prog_index=((unsigned int*)buffer)[0];

switch (prog_index)
  {
  case 0:
```

```
    func0();
    break;
  case 1:
    func1();
    break;
  case 2:
    func2();
    break;
  case 3:
    func3();
    break;

  default:
    *pOutBytes=0;
    break;
  }
}
//---------------------------------------------------------------
```

Если количество выполняемых функций достаточно велико (например, больше четырех), мы рекомендуем использовать вместо оператора «switch», приведенного в примере, таблицы адресов переходов для обеспечения более оперативного выбора. Вариант реализации такого решения показан ниже, в листинге 2.

Листинг 2.

```
//---------------------------------------------------------------
void func0() {...}

void func1() {...}

void func2() {...}

void func3() {...}

typedef void (*Fmas)(void);
Fmas fp[4]={func0,func1,func2,func3};
void user_function()
{
unsigned int prog_index;
…
prog_index=((unsigned int*)buffer)[0];

if (prog_index<7) fp[prog_index]();
else  *pOutBytes=0;
}
//---------------------------------------------------------------
```

3.2. Защита многопользовательского программного обеспечения

Существенная часть производимого в мире коммерческого программного продукта рассчитана на корпоративное использование, или говоря другими словами, подразумевает существование сетевых версий, где появляется понятие рабочих мест, или ограничение количества пользователей. Вариантов построения таких защит предлагается довольно много. Вы можете реализовать на базе ключей «LOCK» любой из вариантов, например, использовать один ключ для организации нескольких рабочих мест. Такой подход является наиболее дешевым, но при этом легко уязвимым, поскольку такие защиты довольно легко обходятся. Самым прочным вариантом построения защиты является использование для каждого рабочего места отдельного локального ключа, и для всей системы в целом так называемого master-ключа. Каждый локальный ключ должен содержать в памяти лишь часть «скрытых» алгоритмов, в то время как master-ключ может содержать как исключительно оставшуюся часть алгоритмов, так и их полный набор. В последнем случае master-ключ дополнительно играет роль локального ключа. Понятно, что при такой структуре защиты количество рабочих мест определяется количеством локальных ключей. Со всех рабочих мест производятся запросы через сеть к master-ключу для выполнения функций, отсутствующих в локальных ключах. Запросы к функциям локальных ключей не являются сетевыми, так как на каждом рабочем месте имеется свой локальный ключ.

3.3. Защита информации

Несмотря на то, что основной акцент системы «LOCK» направлен именно на защиту программного обеспечения, а точнее его функциональных модулей, данная система дает все и даже более широкие возможности по защите информации в целом. Для простоты объяснения принципа защиты здесь следует попытаться логически разделить понятия программного обеспечения и информации в общем. Как известно, любые данные являются информацией, в том числе и код программы. Однако, с точки зрения защиты информации, такое единство является неприемлемым. Дело в том, что между кодом программы и данными есть одна очень существенная разница. Данные всегда, или, по крайней мере, в какой-то момент времени должны быть доступны для восприятия, что является далеко не обязательным, а зачастую даже не желательным для кода программы, который для конечного пользователя, по сути, не несет никакой полезной информации.

На сегодняшний день единственным способом защиты данных является их кодирование, или как сейчас модно говорить – шифрование. Существует множество готовых решений, которые в той или иной степени решают эту задачу. На выбор предлагается огромный ассортимент криптографических алгоритмов с различными длинами ключей (от 64 до 256 бит), но, как показывает многолетняя практика, эффективность такой защиты стремится к нулю быстрее, чем успевают появляться новые разработки. USB-ключи серии «LOCK» не навязывают готовых решений, которые, как правило, имеют довольно жесткие рамки. Напротив, дается возможность разработчикам программного обеспечения реализовать свои собственные уникальные алгоритмы для защиты информации. При этом не ставится практически никаких ограничений, в том числе на длину ключа, которая может составлять даже тысячи бит.

4. ИСПОЛЬЗОВАНИЕ «IAR Embedded Workbench»

Идеология защиты, построенной с использованием USB-ключей серии «LOCK» предполагает, что часть кода программного обеспечения, которая не будет попадать в

память компьютера, должна быть представлена в машинном коде, понятном внутреннему процессору ключа. Для создания загружаемого в ключ файла, содержащего этот код, мы рекомендуем использовать среду разработки «IAR Embedded Workbench» для процессорного ядра ARM. Данная среда разработки поддерживает как ассемблер для ARM, так и язык C/C++. Использование C/C++ является предпочтительным по сравнению с ассемблером, так как защищаемое программное обеспечение может быть отлажено полностью на компьютере, а затем некоторые его функции могут быть перенесены в «IAR Embedded Workbench» для компиляции в код для ключа «LOCK». Кроме того, C/C++ намного удобнее и нагляднее в использовании по сравнению с ассемблером.

4.1. Настройка конфигурации

Конфигурация проекта задается в меню Projects->Edit Configuration. На ***рис. 2*** показано окно конфигурации проекта. Опция «RAM_Debug» используется при работе с отладочными комплектами и проектами, поддерживающими работу со специализированными отладочными интерфейсами. USB-ключи серии «LOCK» не имеют никаких отладочных интерфейсов, поэтому при выборе конфигурации проекта необходимо указать опцию «FLASH».

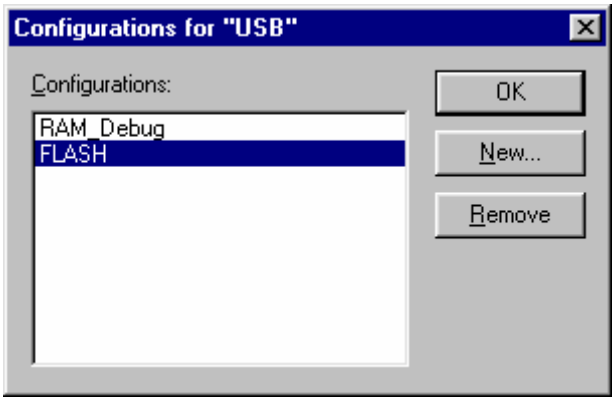

рис. 2. Окно конфигурации проекта

Выбор данной опции подразумевает, что выходной файл при компиляции проекта будет содержать машинный код, предназначенный для FLASH-памяти.

4.2. Настройки компилятора C/C++

Настройки компилятора задаются в меню **Projects → Options**. По умолчанию шаблон имеет предустановленные настройки. На ***рис. 3*** изображено окно настроек шаблона, где выбрана категория «General Options». Здесь необходимо выбрать опцию «Device», после чего указать тип используемого устройства:

AT91SAM7S64 для USB-ARM-64K,
AT91SAM7S128 для USB-ARM-128K,
AT91SAM7S256 для USB-ARM-256K;

в зависимости от того, какую модель ключа Вы используете. Также предустановлен режим Arm, как более быстрый. Однако, если размер Вашего кода велик и не помещается в доступной памяти ключа, Вы можете изменить режим процессора на Thumb, что может уменьшить размер откомпилированного кода примерно в полтора раза. При этом незначительно может уменьшиться скорость выполнения программы ключом.

ВАЖНО!!! Опцию «Generate interwork code» рекомендуется включить для обеспечения лучшей совместимости загружаемого в ключ программного обеспечения и ядра ключа. Установки опций «Endian mode» (little) и «Stack align» (4 bytes) изменять нельзя!

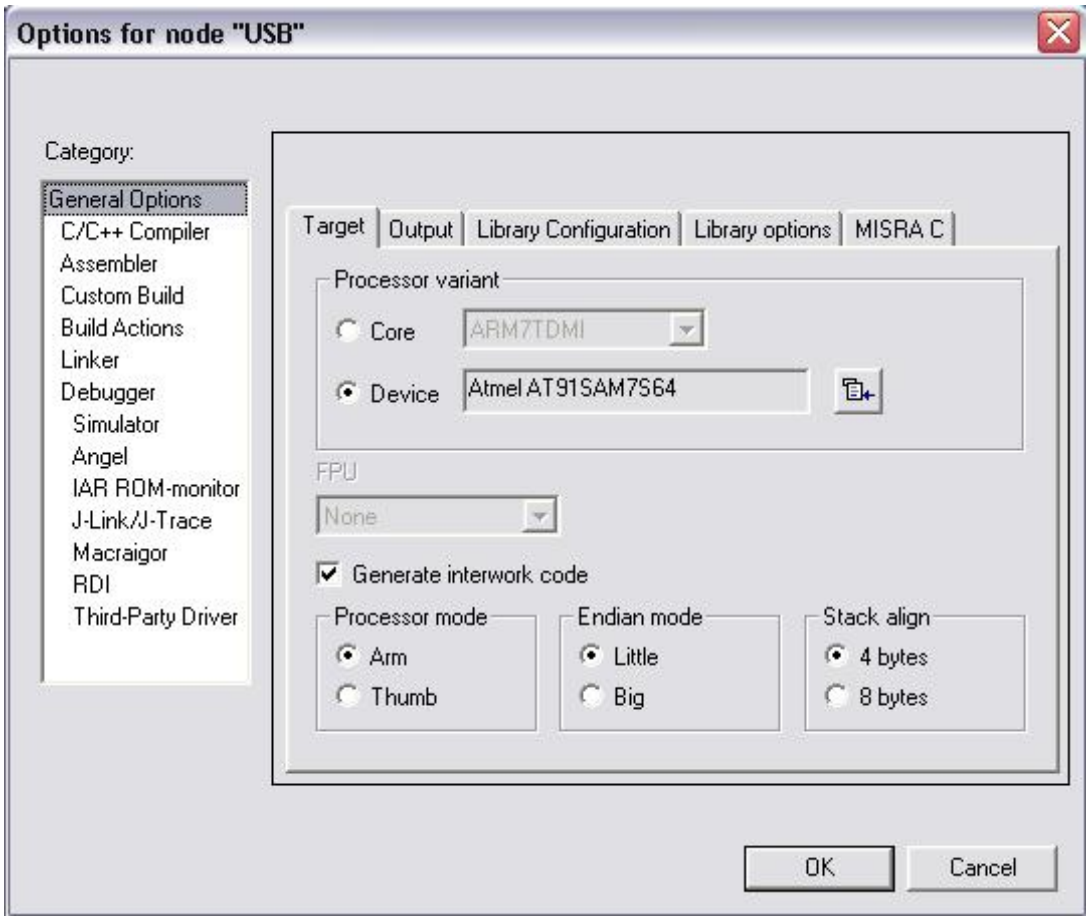

рис. 3. Окно настройки шаблона

После установки указанных опций необходимо выбрать категорию «C/C++ Compiler», а затем перейти на вкладку «Code», где выбрать желаемые опции настройки оптимизации кода. По умолчанию шаблон настроен на оптимизацию скорости работы (Speed). А также включены все оптимизации. Рекомендуем сохранять эти настройки. Однако, если размер Вашего кода велик, Вы можете варьировать их для достижения лучшего результата. Вид окна выбора опций оптимизации кода представлен на ***рис. 4***.

Далее необходимо произвести настройки линковщика.

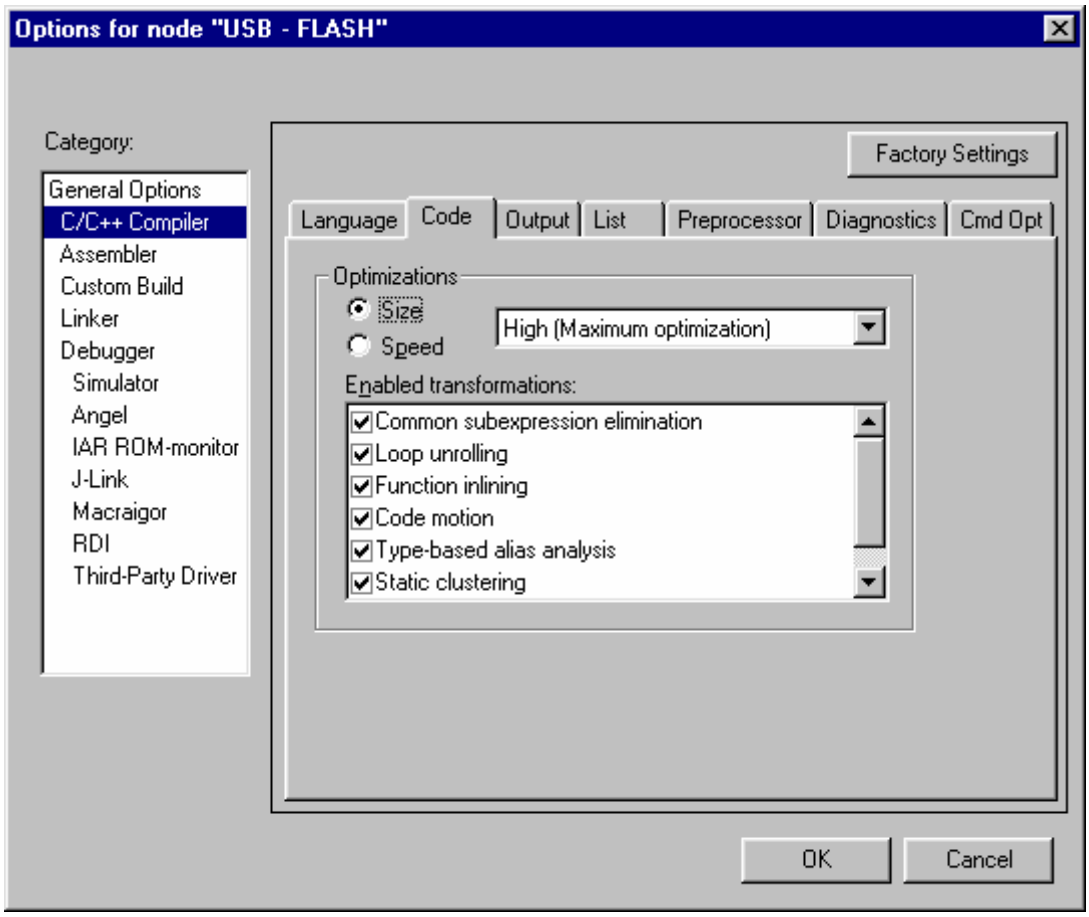

рис. 4. Окно настройки компилятора

4.3. Настройки линковщика

Настройки линковщика задаются так же в меню Projects → Options. Для доступа к настройкам опций линковщика необходимо выбрать категорию «Linker». При этом вид окна настроек примет вид, изображенный на ***рис. 5***. Здесь необходимо указать формат выходного файла, который впоследствии будет загружен в энергонезависимую память USB-ключа (FLASH-память). Для этого в области окна «Format» нужно выбрать опцию «Other», после чего станут активными поля «Output format», «Format variant» и «Module-local symbols». В поле «Output format» из предлагаемого списка необходимо выбрать тип формата «raw-binary».

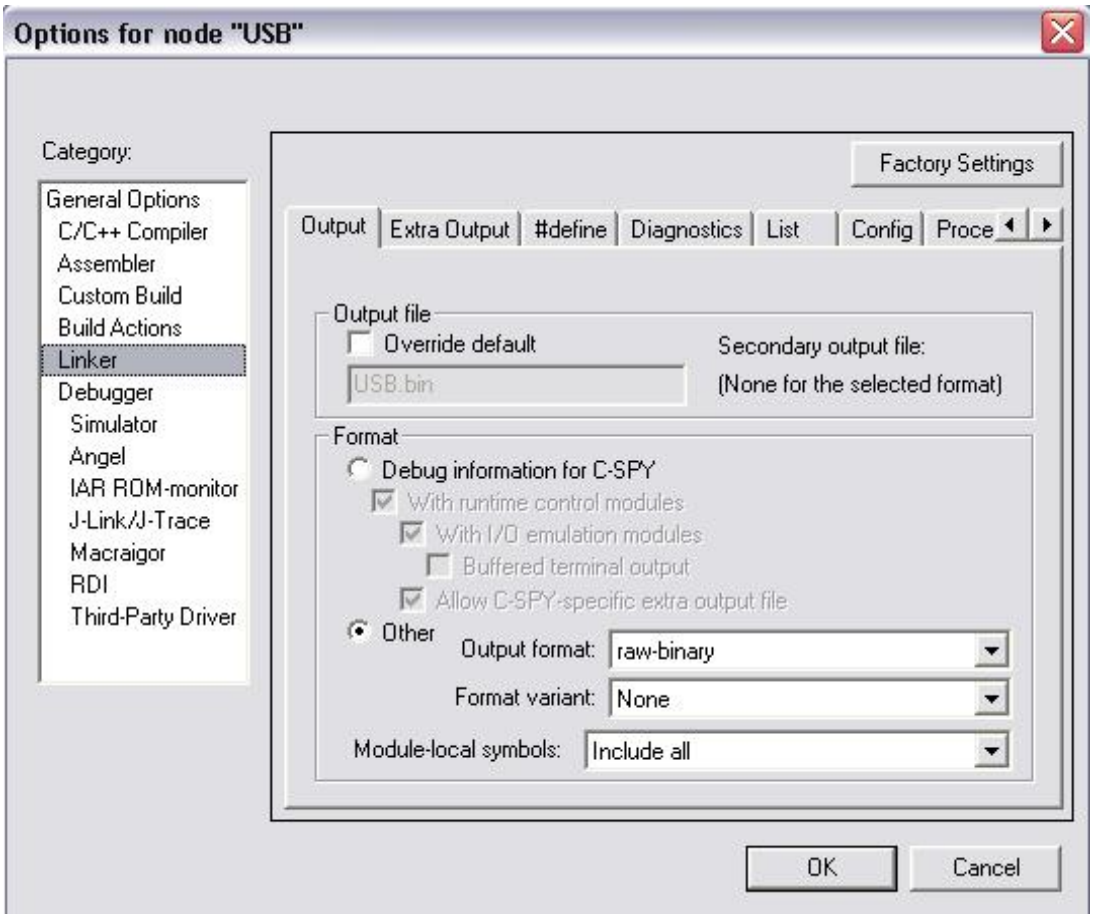

рис. 5. Окно настройки линковщика

<u>ВАЖНО!!!</u> Приложение «Secure Control» работает только с бинарными файлами. Если при выборе формата выходного файла будет указан какой-либо формат отличный от «raw-binary», приложение «Secure Control» будет работать с ним не корректно, что приведет к неработоспособности Вашей программы внутри ключа!

Созданный после компиляции и линковки исполняемый файл будет располагаться в папке **Compil\Flash\Exe** проекта и иметь расширение **.bin** (по умолчанию называется USB.bin).

4.4. Правила внедрения алгоритмов в память ключа

В состав комплекта разработчика «LOCK» входят заготовки проектов для каждого типа USB-ключей, то есть готовые шаблоны, которые мы рекомендуем использовать при внедрении Ваших алгоритмов в ключ. Они располагаются в папке

x:\Program Files\Astroma Ltd\LOCK SDK x.x\Templates

Для каждой модели ключей Вы здесь найдете свою папку:

\64_KeyUserEmptyTemplate для модели ключа USB-ARM-64K,
\128_KeyUserEmptyTemplate для модели ключа USB-ARM-128K,
\256_KeyUserEmptyTemplate для модели ключа USB-ARM-256K.

Код Вашего алгоритма необходимо внедрять в файл исходного текста **«UserFunction.c»** шаблона. В любой другой части шаблона внедрение алгоритма пользователя не рекомендуется. Ниже на листинге 3 приведено содержимое файла-шаблона **«UserFunction.c»** для ключа USB-ARM-64K.

Листинг 3.

```
//-------------------------------------------------------------
#include "UserFunction.h"

//обязательные переменные  и константы,
//их менять не следует

const unsigned int HeapSize=0x2000; //зависит от модели ключа
const unsigned int BuffSize=0x1000;
unsigned int *pInAddrPtr,*pInBytes,*pOutAddrPtr,*pOutBytes;
#define AT91C_ISRAM ((char *) 0x00200000)

void user_function()
{
//ввод-вывод
pInAddrPtr =(unsigned int*)(AT91C_ISRAM+HeapSize+BuffSize+0x08);
pInBytes =(unsigned int*)(AT91C_ISRAM+HeapSize+BuffSize+0x0C);
pOutAddrPtr=(unsigned int*)(AT91C_ISRAM+HeapSize+BuffSize+0x10);
pOutBytes=(unsigned int*)(AT91C_ISRAM+HeapSize+BuffSize+0x14);


//Здесь размещается реализация Ваших алгоритмов


//Ниже производится ОБЯЗАТЕЛЬНОЕ возвращение значений
//через указатели pOutAddrPtr и pOutBytes.
//Если UserFunction ничего не возвращает в результате своего
//выполнения, в pOutBytes ОБЯЗАТЕЛЬНО нужно прописать 0 !!!
//        *pOutBytes=0;

*pOutAddrPtr=*pInAddrPtr;
*pOutBytes=*pInBytes;
}
//-------------------------------------------------------------
```

Значения объявляемых переменных и констант менять не следует, так как это может привести к некорректной работе USB-ключа с Вашим программным обеспечением. Далее идет установка параметров ввода-вывода. Для ясности следует отметить, что все USB-ключи серии «LOCK» имеют область памяти размером 4096 байтов, которая отведена под буфер обмена для интерфейса USB. Когда ключ принимает от компьютера данные, они всегда оказываются в этой области. Первые 8 байтов содержат служебную системную информацию, поэтому реальный объем данных, которые можно единовременно послать в ключ, составляет 4096 - 8 = 4088 байтов. Адрес начала принятых данных в памяти ключа содержится в переменной, на которую указывает **pInAddrPtr**. Количество принятых байтов можно получить, используя указатель **pInBytes**. По завершении обработки принятых данных (после выполнения всех требуемых алгоритмов) необходимо задать параметры для отправки результатов в системных переменных, которые доступны посредством указателей pOutAddrPtr (начало отправляемых данных) и pOutBytes (количество отправляемых данных в байтах). Для отправки результатов не обязательно использовать ту же область памяти, куда принимаются исходные данные.

Отправка может производиться из любой области памяти. Разумеется, все данные предварительно должны быть подготовлены к отправке.

ВАЖНО!!! Для обеспечения корректной работы USB-ключей функция UserFunction должна в любом случае возвращать значение в *pOutBytes. Если в результате выполнения функции никаких выходных данных (результатов) нет, перед выходом из функции следует это указать следующим образом:

***pOutBytes = 0.**

В этом случае указывать начальный адрес отправляемых данных (через указатель pOutAddr) не требуется.

4.5. Организация в программе запросов USB-ключа

Листинг 4 иллюстрирует работу защищенного программного обеспечения с ключами серии «LOCK». Пример снабжен достаточным количеством комментариев, поэтому дополнительно рассматривать его не будем.

Листинг4.

```
//-------------------------------------------------------------
...
#define BUF_SIZE 4088

CMemFile mf;
int iCurrDevice,iError;
unsigned int ulRecieveSize,ulFileSize;
byte pInputDataBuff[BUF_SIZE];
byte pOutputDataBuff[BUF_SIZE];
...
void CmfcsampleDlg::OnBnClickedButton2()
{
...
//подготовка исходых данных для отправки в ключ
//выполняется при помощи CmemFile, так как данный класс имеет
// удобные функции для работы с бинарными данными.
int p;
byte oc;
mf.SeekToBegin();
mf.Flush();
p=100;
mf.Write(&p,4);
oc=100;
mf.Write(&oc,1);
//поиск нужного ключа
iCurrDevice=GetKey("BALLS");
//выполнение программы в найденном ключе
p=ExecKey();
if(!p || !pOutputDataBuff[0]) return;
...
}
//пример функции поиска нужного ключа
```

```
int CmfcsampleDlg::GetKey(char * UserString)
{
int j,i;
KEY_CAPABILITES kc;
//получение количества устройств
j=GetDeviceList();

for (i=0;i<j;i++)
        {
       //программный «захват» ключа
        iError=SoftLockKey(100);
       //если не удалось - возврат с некорректным номером ключа
        if(iError!=STATUS_OK) return -1;
       //иначе опросить ключ для получения информации о нем
        GetKeyCapabilites(i,&kc);
       //сброс программного «захвата» ключа
        SoftUnLockKey();
       //ес ли нужный ключ найден - возращается его номер.
       //возможен поиск по любым полям структуры KEY_CAPABILITES
        if (!strcmp((char*) kc.cUserString,UserString))
                return i;
        }
//если нужный ключ найти не удалось-возврат с кодом «-1»
return -1;
}
//пример функции выполнения кода в ключе
int CmfcsampleDlg::ExecKey()
{
//определение, сколько байт будет отправлено
ulFileSize=(unsigned int)mf.GetLength();
//полученных байт по умолчанию - 0
ulRecieveSize=0;
//обнуление буффера приема
ZeroMemory(pOutputDataBuff,BUF_SIZE);
//копирование подготовленных данных в массив для отправки в ключ
mf.SeekToBegin();
mf.Read(pInputDataBuff,ulFileSize);
//если нужный ключ был найден
if (iCurrDevice > -1)
        {
       //программный «захват» ключа
        iError=SoftLockKey(100);
       //если не удалось - возврат с ошибкой
        if(iError!=STATUS_OK) return 0;
        //выполнение функции в ключе
        iError = SendData (iCurrDevice, (unsigned
char*)pInputDataBuff,
                        ulFileSize, (unsigned
                char*)pOutputDataBuff,
                        &ulRecieveSize);
       //сброс программного «захвата» ключа
        SoftUnLockKey();
```

```
        //если выполнение функции в ключе удалось
         if(iError == STATUS_OK)
                 return 1;
         }
//если нужный ключ не был найден, или выполнение функции
// в ключе не удалось - возврат с ошибкой
return 0;
}
//-------------------------------------------------------------
```

Дополнительно в составе комплекта разработчика «LOCK SDK» имеются несколько полностью рабочих примеров защищенных программ с исходными текстами для сред разработок: Borland C++, Visual C++ и C#. Тексты программ, предназначенные для компьютера можно найти в каталоге **\LOCK SDK x.x\Samples**. Исходные тексты программного обеспечения, загружаемого в память ключа, находятся в каталоге **\LOCK SDK x.x\Templates**. Здесь находятся три примера для трех моделей ключей:

\64_KeyUserTemplate для модели ключа USB-ARM-64K,
\128_KeyUserTemplate для модели ключа USB-ARM-128K,
\256_KeyUserTemplate для модели ключа USB-ARM-256K.

Для того чтобы воспользоваться примерами, следует учитывать, что это такие же защищенные ключами «LOCK» программы (пусть примитивные, и их уровень защиты оставляет желать лучшего, так как специально приводится для демонстрации), как и распространяемые сегодня на рынке программного обеспечения, а система защиты сама по себе не имеет демонстрационных версий. Поэтому для корректной работы примеров необходимо соблюдать все правила конфигурирования ключей, как для реальных защищаемых программ.

ВАЖНО!!! Для работы с примерами при конфигурировании ключа в разделе «Key Capabilities» следует заполнить поле «User String» значением «BALLS» (см. раздел 6.4.2), в противном случае пример функционировать не будет.

5. БИБЛИОТЕКА «UserLibrary.dll»

Для удобства организации работы с ключами «LOCK» существует ряд готовых функций, которые входят в состав библиотеки «UserLibrary.dll». Они решают задачи обмена данными между компьютером и USB-ключами, тем самым, освобождая разработчика программного обеспечения от проблемы реализации протокола обмена. Если программное обеспечение использует аппаратную защиту «LOCK», то библиотека «UserLibrary.dll» становится неотъемлемой частью проекта и должна поставляться конечному пользователю в составе защищенного программного обеспечения, собственно, так же как и драйвер USB-ключей. Далее описаны функции, входящие в состав библиотеки.

5.1. Функция GetDeviceList

Начало работы любого приложения, защищенного ключами «LOCK» должно начинаться с выполнения именно этой функции, которая позволяет определить количество локально подключенных к компьютеру USB-ключей. Если не подключено ни одного ключа, функция вернет значение 0. В противном случае возвращаемое значение будет указывать количество доступных устройств. Функция не имеет входных параметров.

Формат функции:
INT GetDeviceList(VOID)

5.2. Функция GetKeyCapabilities

Данная функция предназначена для чтения параметров указанного устройства. В первую очередь возникает необходимость ее использования для идентификации USB-ключей серии «LOCK», так как их количество, подключаемое к компьютеру, может достигать 127, но каждое приложение, использующее данную защиту, должно найти именно свой ключ.

Формат функции:
INT GetKeyCapabilities(INT DevNumber, PKEY_CAPABILITIES pCapabilities)

Входные параметры:

DevNumber – номер устройства из подключенных. Здесь следует указывать значение в диапазоне от 0 до полученного с помощью функции GetDeviceList минус 1;

pCapabilities – указатель на структуру типа ***KEY_CAPABILITES***, описание которой приведено ниже.

Листинг5.

```
//-------------------------------------------------------------
typedef struct _KEY_CAPABILITIES{
 UINT     iType;              //идентификатор модели устройства
 UINT     iSerialNumber;      //серийный номер производителя
 UINT     iUserDefined;       //поле разработчика
 UCHAR    cUserString[28];    //текстовое поле разработчика
 UINT     iUserSerial;        //серийный номер разработчика
 UINT     iMaxCodeSize;       //объем памяти программ
 UINT     iMaxDataSize;       //объем памяти данных
 UINT     iStatus;            //статус устройства
 UINT     iCoreVersion;       //версия ядра
 UINT     iLicNumber;         //количество временных лицензий
 UINT     iLicPeriod;         //длительность интервалов лицензий
 UINT     iLicCounter;        //остаток временных лицензий
 } KEY_CAPABILITES,*PKEY_CAPABILITES;
//-------------------------------------------------------------
```

При удачном завершении функция возвращает значение **STATUS_OK**. Это означает, что устройство найдено, и структура ***KEY_CAPABILITIES*** заполнена соответствующими данными выбранного устройства. Иначе возвращаются коды **STATUS_ERROR** (заданы неверные параметры) или **STATUS_DEVICE_NOT_RESPOND** (устройство не подключено или не отвечает).

5.3. Функция SendData

По сути, данная функция является основной в системе защиты «LOCK». Именно благодаря ей, осуществляется обмен данными между компьютером и USB-ключом, а так же исполнение программного кода, хранящегося (скрытого) в USB-ключе.

Формат функции:
INT SendData(INT DevNumber, UCHAR* pSendBufferPtr, UINT SendBufferLength, UCHAR* pRecieveBufferPtr, UINT* pRecieveBytes)

Входные параметры:

DevNumber – номер устройства из подключенных. Диапазон от «0» до значения, возвращаемого функцией GetDeviceList минус 1;

pSendBufferPtr – указатель на буфер с данными, подготовленными для отправки в ключ;

SendBufferLength – количество отправляемых в ключ байтов.

pRecieveReturnPtr – указатель на буфер, куда должны быть помещены возвращаемые данные;

pRecieveBytes – возвращенное ключом количество байтов.

При удачном завершении функция возвращает значение **STATUS_OK**. Это означает, что выполнение функции завершено успешно и выходные параметры (результаты) переданы в компьютер. Иначе возвращаются коды завершения **STATUS_ERROR** (заданы неверные параметры) или **STATUS_DEVICE_NOT_RESPOND** (устройство не подключено или не отвечает).

ВАЖНО!!! При подготовке данных для обмена между компьютером и ключом следует учитывать, что двухбайтовые параметры должны быть выровнены в буферах по границе кратной двум, а четырехбайтовые параметры – по границе кратной четырем соответственно.

5.4. Функция SoftLockKey

Если один ключ используется одновременно несколькими программами, для исключения конфликтных ситуаций следует производить так называемый «захват» ключа. Собственно для этих целей и предназначена функция SoftLockKey. Программное приложение при помощи данной функции производит «захват» ключа с указанием времени ожидания захвата. Пока ключ захвачен, остальные приложения не будут иметь к нему доступа. Поэтому по окончании сеанса работы с ключом приложение должно «освободить» ключ. Это позволяет сделать функция SoftUnLockKey , описанная далее.

Формат функции:

```
INT     SoftLockKey(DWORD dwTimeOut)
```

Входные параметры:

dwTimeOut – значение времени ожидания «захвата» (в миллисекундах). Если здесь задать значение 0, то функция однократно произведет попытку «захвата» без какого-либо ожидания, и незамедлительно произойдет возврат. При задании значения INFINITE (значение эквивалентное -1) возврат из функции будет произведен только после выполнения «захвата». Время ожидания в этом случае будет не ограничено.

Функция возвращает значение **STATUS_OK**, если «захват» произведен удачно. В противном случае будет возвращаться значение **STATUS_TIMEOUT_EXPIRED** (время ожидания истекло).

5.5. Функция SoftUnLockKey

Данная функция используется для отмены «захвата» ключа, который производится с помощью функции SoftLockKey, если ключ одновременно используется несколькими приложениями. Выполнение этой функции обязательно после завершения сеанса работы с

ключом, чтобы предоставлялся доступ к ключу другим приложениям, которые так же должны использовать механизм захвата ключа с целью предотвращения возникновения конфликтных ситуаций.

Формат функции:

INT SoftUnLockKey()

Входные параметры:

отсутствуют.

После выполнения функция всегда возвращает значение **STATUS_OK**.

6. КОМПЛЕКТ РАЗРАБОТЧИКА «LOCK SDK»

В комплект разработчика входит все необходимое программное обеспечение для разработки и реализации защиты с использованием USB-ключей «LOCK»:

- драйвер USB-ключей «LOCK»,
- библиотека API-функций UserLibrary.dll,
- программное приложение «Secure Control»,
- заготовки проектов для всех типов ключей «LOCK»,
- примеры реализации защиты,
- руководство разработчика.

В данном разделе речь пойдет об использовании программного приложения «Secure Control», которое входит в состав комплекта разработчика, для программирования USB-ключей серии «LOCK». Данный программный продукт дает возможность быстро и удобно осуществлять загрузку программного обеспечения в USB-ключи, устанавливать при необходимости защиту требуемого уровня надежности от стирания или дальнейшей перезаписи содержимого памяти ключей, а также организовывать работу с временными лицензиями, позволяющими вводить ограничения на время использования защищенного программного продукта.

ВАЖНО!!! Программное приложение «Secure Control» предназначено только для использования разработчиком защищаемого программного обеспечения и не должно входить в состав поставляемого защищенного программного продукта!

6.1. Установка пакета «LOCK SDK»

6.2. Установка драйвера USB-ключей «LOCK»

Перед началом установки драйвера подключите ключ к свободному порту USB компьютера, после чего операционная система автоматически запустит мастер установки нового оборудования. На экране появится окно, изображенное на ***рис. 6***, где будет запрошено разрешение на подключение через Интернет к узлу Windows Update для поиска подходящего драйвера.

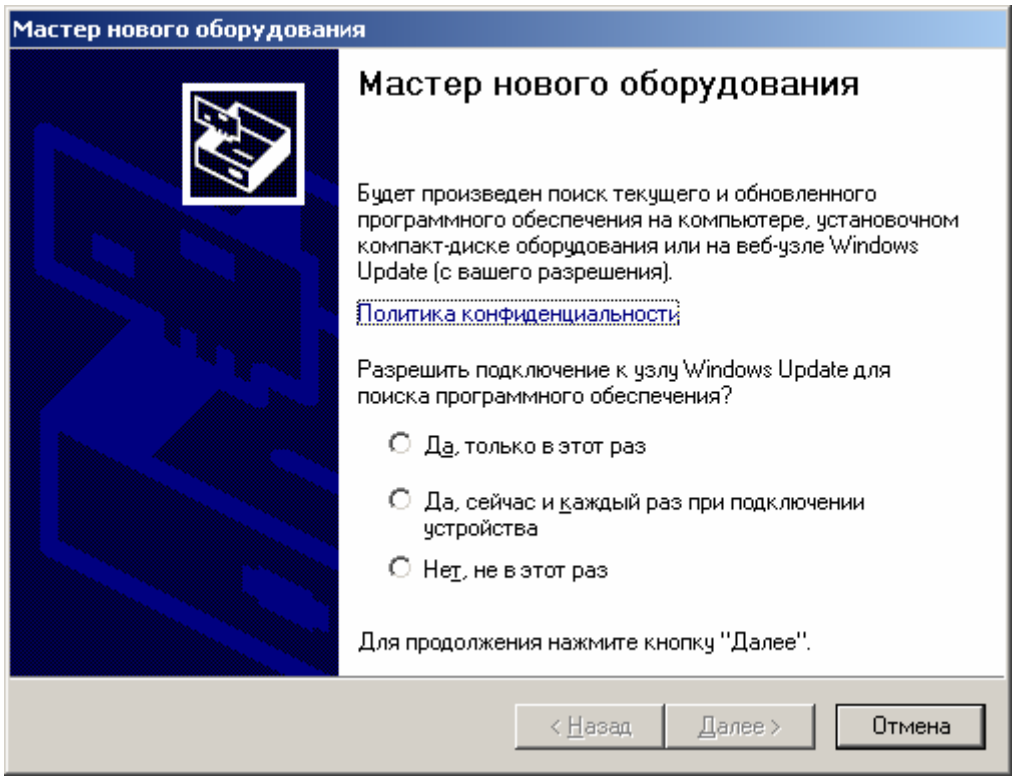

рис. 6. Опция подключения к узлу Windows Update

В данном случае должна быть выбрана опция «Нет, не в этот раз», после чего нажимается кнопка «Далее».

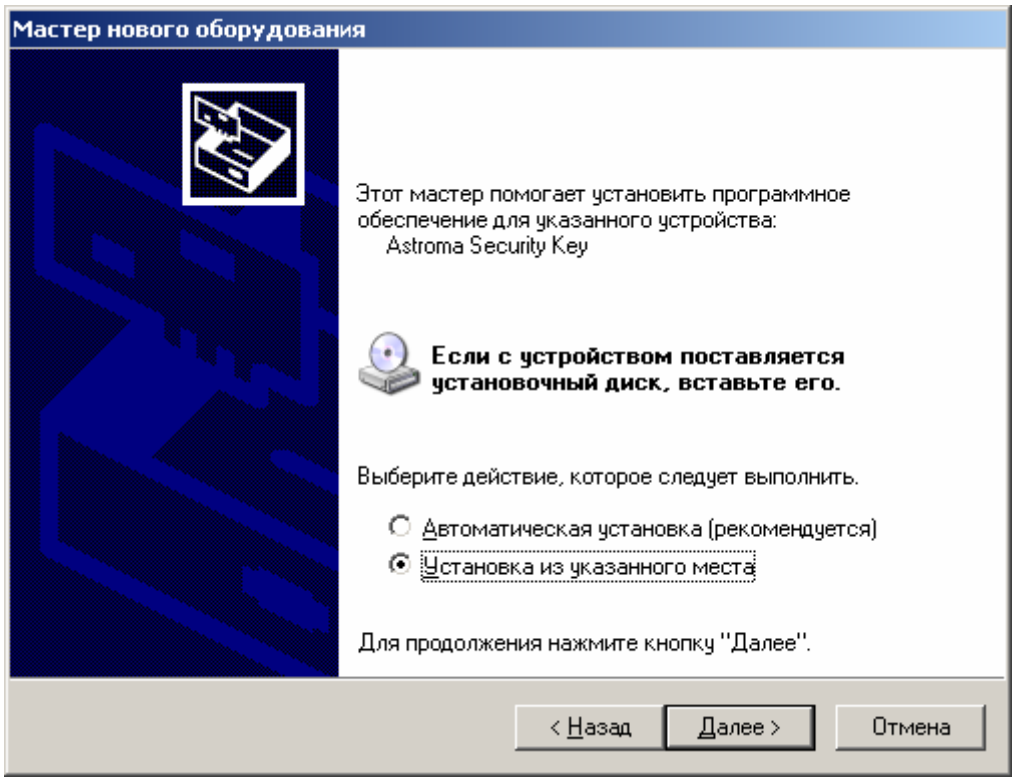

рис. 7. Выбор способа установки драйвера

Следующим шагом требуется указать способ установки драйвера. Для этого необходимо в окне, изображенном на ***рис. 7***, выбрать опцию «Установка из указанного места» и нажать кнопку «Далее».

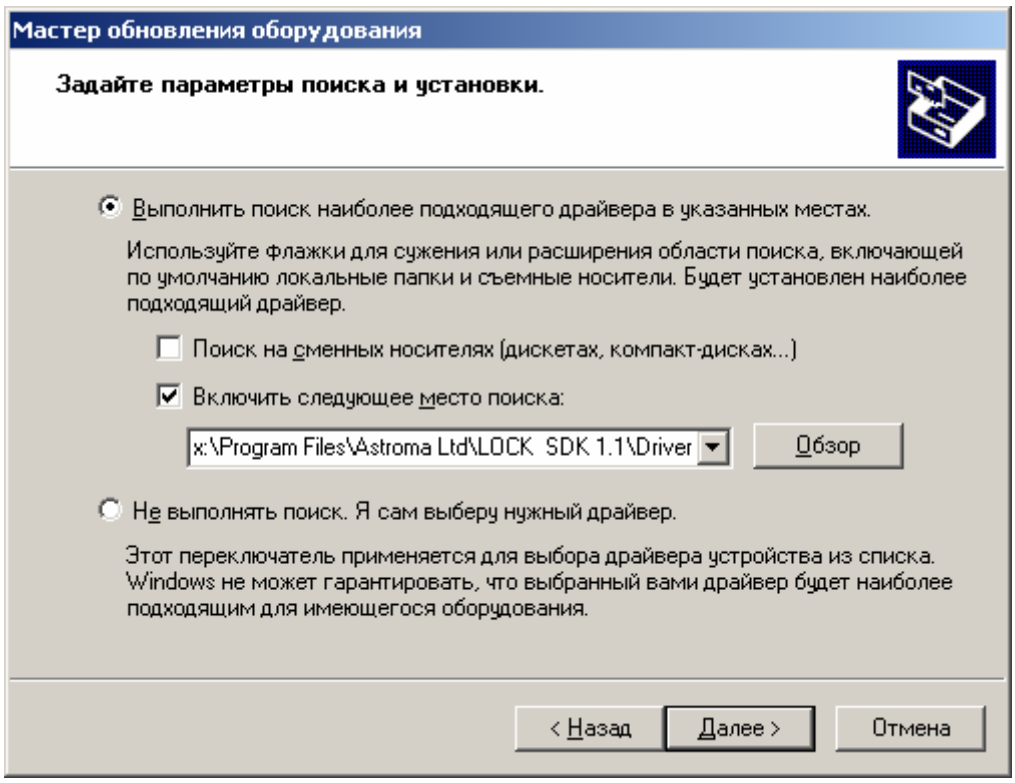

рис 8. Указание местонахождения драйвера

В появившемся при этом окне (см. ***рис 8***) следует выбрать опции «Выполнить поиск наиболее подходящего драйвера в указанных местах» и «Включить следующее место поиска», после чего в строке указать местоположение папки с драйвером:

x:\Program Files\Astroma Ltd\LOCK SDK 1.1\Driver ,

где x: - буквенное обозначение носителя информации, на который был установлен пакет «Secure Control». После этого нажмите кнопку «Далее».

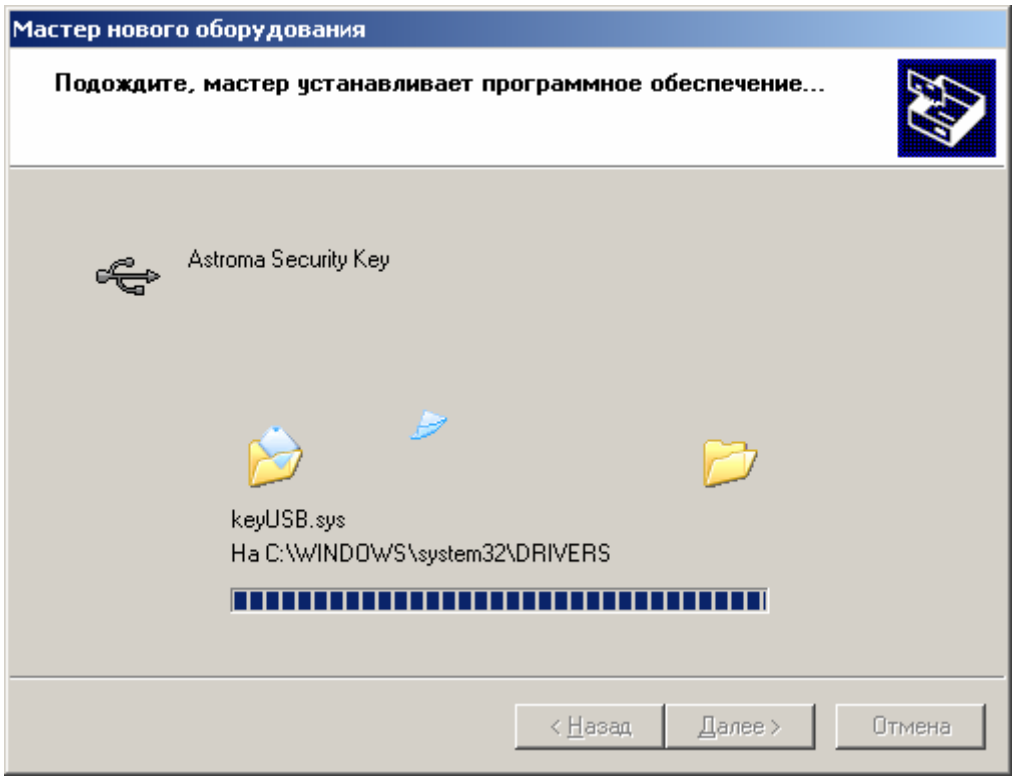

рис. 9. Копирование файлов в системные папки

Мастер нового оборудования произведет копирование всех необходимых файлов в системные папки.

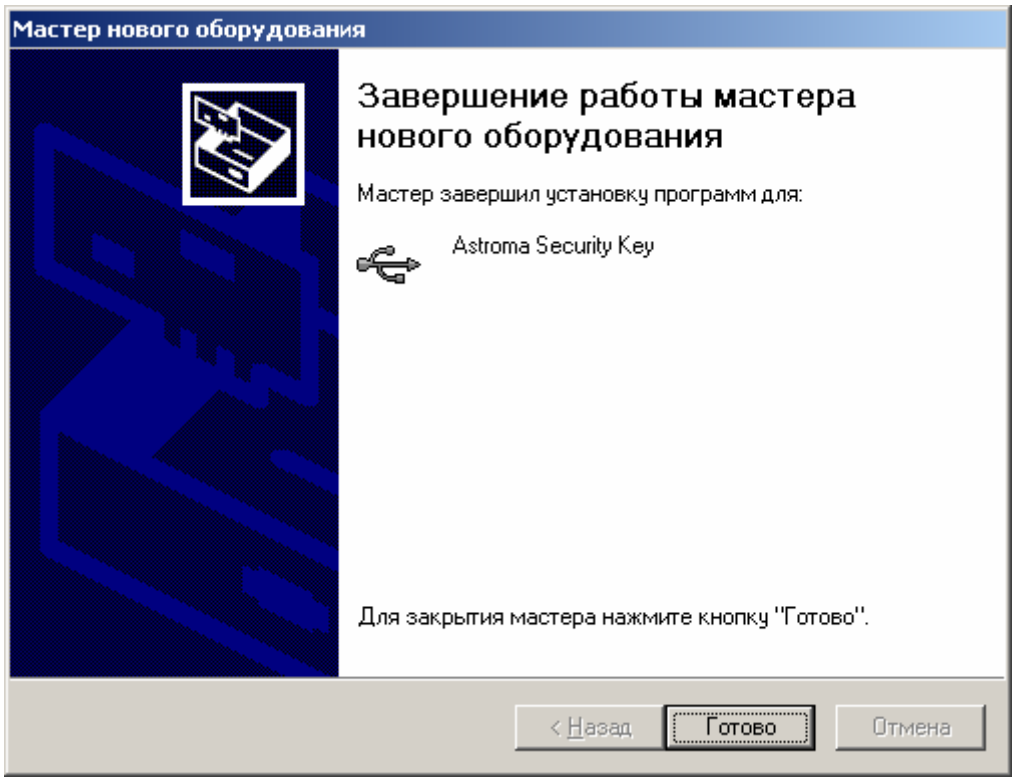

рис. 10. Окончание процедуры установки драйвера

По окончании процедуры копирования файлов нажмите кнопку «Готово» (см. ***рис. 10***). Драйвер ключа аппаратной защиты установлен. С этого момента ключи «LOCK» должны корректно распознаваться системой и нормально функционировать.

6.3. Запуск пакета «Secure Control»

Если на Вашем компьютере еще не установлено программное обеспечение «Secure Control», произведите его установку обычным образом, после чего запустите. Для работы с данной программой необходимо подключить к компьютеру, по крайней мере, один USB-ключ серии «LOCK». На открывшейся панели управления (см. ***рис. 11***) поля 7, 8 и 9 будут пустыми, а поле 6 будет содержать надпись «Select device». При этом будут активны две кнопки: «1 – выбор устройства» и «5 – о программе». Нажатие кнопки «о программе» вызывает панель с информацией о данном программном продукте.

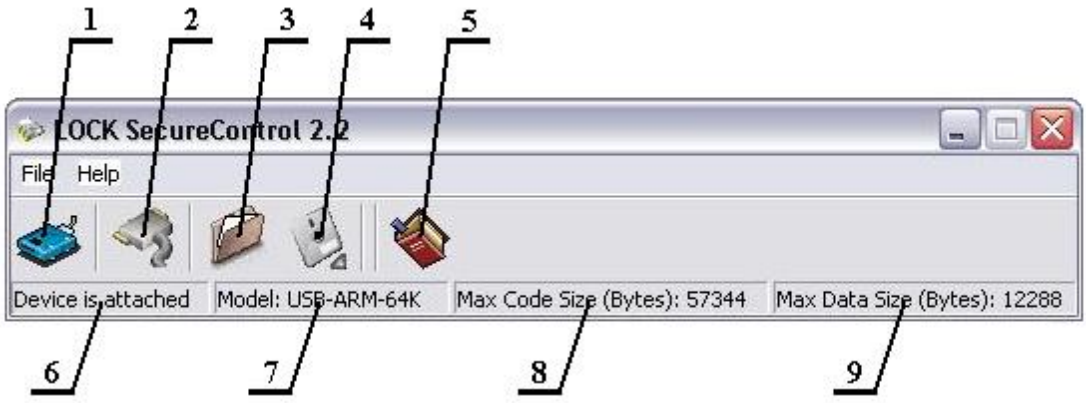

рис. 11. Панель управления «Secure Control»

1 – выбор устройства и просмотр текущих характеристик; 2 – задание новых характеристик выбранному устройству; 3 – выбор файла для загрузки в память ключа; 4 – загрузка выбранного файла в память ключа; 5 – о программе; 6 – индикация связи с устройством; 7 – модель выбранного устройства; 8 – доступный объем энергонезависимой памяти; 9 – доступный объем оперативной памяти.

Работу по программированию USB-ключей всегда следует начинать с выбора устройства, так как их может быть подключено одновременно несколько. Выбор ключа производится в окне выбора устройства «Key Devices», которое открывается путем нажатия кнопки 1 панели управления - «выбор устройства» (см. ***рис. 11***). При этом на экране появится окно, изображенное на ***рис. 12***, в котором и следует произвести выбор USB-ключа для последующей работы с ним. Выбор устройства производится из списка «Available Devices», в котором перечислены все доступные USB-ключи «LOCK», подключенные к компьютеру.

ВАЖНО!!! Приложение «Secure Control» работает только с ключами, локально установленными на компьютере. Сетевые ключи так же должны быть подключены локально.

При указании в списке USB-ключа информация о нем будет отображаться в полях окна выбора устройства, которые описаны ниже. Некоторые из полей относятся к разделу памяти ключа «Key Capabilities», который будет подробно рассмотрен в разделе 6.4 данного руководства.

- **«Key Current Status»** показывает код текущего состояния USB-ключа. Если в памяти ключа присутствует программа, код состояния должен иметь значение 6. В противном случае, если программа не загружена, либо стерта, код состояния должен принимать значение 4. Может возникнуть ситуация, когда код состояния будет иметь значение 0, но после установки параметров «Key Capabilities», либо загрузки в USB-ключ программы значение кода состояния должно принять значение 4, или 6. При состоянии USB-ключа отличном от 4 или 6 поле «Key Current Status» будет выделяться красным цветом.

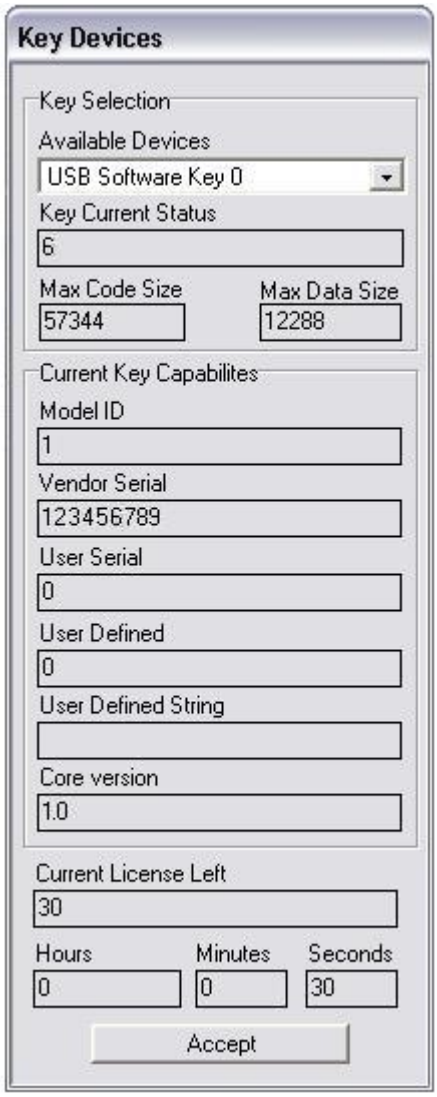

рис. 12. Окно выбора ключа

ВНИМАНИЕ!!! **Если код состояния USB-ключа принимает какие-либо другие значения, это говорит о том, что ключ находится в неисправном состоянии, и его использование недопустимо! В этом случае рекомендуем Вам обратиться за помощью к разработчику.**

- **«Max Code Size»** отображает информацию о доступном объеме энергонезависимой памяти в байтах для хранения программного кода.
- **«Max Data Size»** показывает доступный объем пространства оперативной памяти. Значение также отображается в байтах.
- **«Model ID»** - код модели USB-ключа.
- **«Vendor Serial»** - четырехбайтовый серийный номер устройства, задаваемый производителем USB-ключей.
- **«User Serial»** - четырехбайтовый серийный номер устройства, устанавливаемый разработчиком программного продукта при установке защиты «LOCK». Данное поле может содержать числовое значение от 0 до 4294967295.
- **«User Defined»** - четырехбайтовое числовое поле, которое может использоваться для идентификации USB-ключа. Данное поле заполняется разработчиком программного продукта при конфигурировании ключа и может содержать числовое значение в диапазоне от 0 до 4294967295.
- **«User Defined String»** - текстовое поле, которое так же может использоваться для идентификации USB-ключа, заполняется разработчиком программного обеспечения. Максимальная длина строки, которая прописывается в данное поле, не должна превышать 27 символов.
- **«Core Version»** отображает номер версии ядра USB-ключа.
- **«Current License Left»** - информационное поле, которое отображает текущее значение счетчика временных лицензий, если они в данном ключе активированы. Отображается оставшееся количество тиков счетчика до окончания лицензии.

• **«Hours», «Minutes», «Seconds»** - поля, отображающие время до окончания действия временной лицензии, если она активирована.

После указания в списке «Available Devices» необходимого USB-ключа для подтверждения следует нажать кнопку «Accept», расположенную внизу окна «Key Devices». Это будет означать, что USB-ключ выбран, и дальнейшая работа будет производиться именно с ним. После выбора устройства в поле 6 панели управления «Secure Control» (см. ***рис. 11***) отобразится надпись «Device is attached», а поля 7, 8 и 9 будут отображать соответствующие характеристики выбранного устройства. Помимо этого станут активными кнопки «2» и «3».

6.4. Установка/изменение параметров «Key Capabilities»

Каждый USB-ключ серии «LOCK» содержит в себе раздел «Key Capabilities», благодаря чему является уникальным и может быть точно идентифицирован. Это является необходимым, так как пользователь может использовать несколько различных пакетов программного обеспечения одного или разных производителей, использующих для защиты своих авторских прав USB-ключи серии «LOCK», при этом каждый пакет программного обеспечения должен установить связь с ключом, относящимся именно к нему. Помимо этого раздел «Key Capabilities» содержит информацию об используемых временных лицензиях, если это необходимо, а также об уровне защиты USB-ключа от стирания или перепрограммирования. Далее более подробно будут описаны возможности различных настроек USB-ключей посредством раздела «Key Capabilities».

6.4.1. **Параметры, устанавливаемые производителем**

Раздел «Key Capabilities» USB-ключей «LOCK» содержит несколько полей, которые устанавливаются производителем на стадии начального программирования. Эти поля не могут быть модифицированы разработчиком программного обеспечения при программировании ключа, однако их содержимое может быть использовано для детальной (индивидуальной) идентификации устройства. К таким полям относятся:

• **«Model ID»** - код модели USB-ключа: 1 – USB-ARM-64K, 2 – USB-ARM-128K, 3 – USB-ARM-256K;

• **«Vendor Serial»** - серийный номер устройства, задаваемый производителем (возможно появление одинаковых серийных номеров, но только для разных моделей USB-ключей);

• **«Core Version»** - версия ядра USB-ключа (не зависит от модели, модификации вносятся во все устройства единовременно, если это позволяют их технические возможности).

6.4.2. **Параметры, задаваемые разработчиком программного обеспечения**

Теперь рассмотрим поля раздела «Key Capabilities», которые заполняются разработчиком программного обеспечения при конфигурировании USB-ключей. Несколько полей не оказывают на работу ключа никакого влияния и предназначены исключительно для облегчения процесса идентификации устройства, то есть информация, записываемая в эти поля, может быть в любой момент прочитана защищаемым программным обеспечением. К таким полям относятся:

- «User Defined»,
- «User Serial»,
- «User String».

Перечисленные поля были достаточно подробно описаны выше в разделе 6.3 при рассмотрении окна выбора устройства «Key Devices». Другие же поля раздела «Key Capabilities» служат для управления работой USB-ключа. Такими полями являются:

- «Password» - поле длинной 8 байтов для задания пароля на стирание/перезапись содержимого памяти USB-ключа (см. раздел 6.6);
- «Lic intervals» - количество интервалов счетчика временных лицензий (если используются временные лицензии).

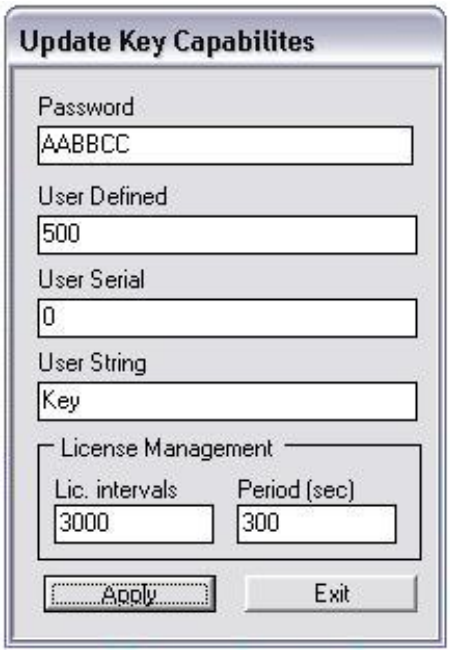

рис. 13. Окно задания параметров USB-ключа

- «Period (sec)» - длительность одного интервала счетчика временных лицензий (в секундах). Работа с временными лицензиями описана далее в разделе 6.4.3.

<u>ВНИМАНИЕ!!!</u> Если в ключе не активирована защита от перезаписи, любая попытка обновления параметров раздела «Key Capabilities» ключа неизбежно приведет к стиранию области Flash-памяти, предназначенной для хранения программного обеспечения.

6.4.3. **Использование временных лицензий**

Все модели USB-ключей серии «LOCK» позволяют организовывать управление временными лицензиями. То есть ключ может быть запрограммирован на ограничение времени работы защищаемого программного обеспечения. Привязка к реальному времени (системному таймеру) в ключах серии «LOCK» не производится, поэтому попытка перевода системных часов компьютера не может заставить ключ продолжать работать после окончания действия временных лицензий. То есть в ключе задается фактическое время работы защищаемого программного обеспечения, подобно тому, как предоставляется время доступа по карточкам в Интернет, или работают тарифные планы операторов сотовой связи. Такой путь реализации выбран исключительно с целью обеспечения максимальной надежности защиты, так как другие варианты обладают гораздо более низким уровнем стойкости. Для реализации этих возможностей в разделе «Key Capabilities» USB-ключа присутствуют два поля, посредством которых происходит задание режима работы временных лицензий. К этим полям относятся «Lic. intervals» и «Period (sec)» (см. ***рис. 13***). В поле «Lic. intervals» задается количество временных интервалов для списывания лицензий, длительность временного интервала в секундах указывается в поле «Period (sec)».

ВАЖНО!!! В случае использования временных лицензий доступная для хранения программ память USB-ключей несколько уменьшается: для ключа USB-ARM-64K - на 512 байтов, для ключей USB-ARM-128K и USB-ARM-256K – на 1 KB.

Если при задании/изменении параметров «Key Capabilities» (см. выше) Вы не указали, что время функционирования защищаемого программного продукта должно быть ограничено, это будет означать, что ключ предоставит пользователю временную лицензию «Unlimited». Время работы задается в окне установки «Key Capabilities» вместе с другими параметрами ключа. По истечении указанного времени работы ключ будет блокировать выполнение загруженного в его память пользовательского программного обеспечения. Механизм управления временными лицензиями не привязывается к реальному времени, а отслеживает фактическое время работы, что не позволяет обойти заданные временные рамки путем перевода системных часов. Учет времени ведется с момента начала работы защищенного программного обеспечения до выключения компьютера или отключения ключа.

ВНИМАНИЕ!!! Если вы используете временные лицензии, старайтесь задавать в поле длительности интервалов («Period (sec)») по возможности большие значения (например, 300 – длительность интервала, равная 5 минутам). Это связано с ограниченностью ресурса FLASH-памяти ключа, так как каждое изменение значения счетчика интервалов (начальное значение задается в поле «Lic. intervals») сопровождается процедурой перезаписи страниц FLASH-памяти, отвечающих за временные лицензии. Минимальное количество циклов перезаписи, гарантированное производителем микросхемы, составляет 320 000, типичное значение – 3 200 000.

6.5. Загрузка программного обеспечения в память ключа

Перед тем, как Вы приступите к процедуре загрузки в память ключа программного кода, убедитесь, что все поля раздела «Key Capabilities» правильно заполнены (см. ***рис. 13***), в противном случае процедуру загрузки в ключ программы придется повторить, так как при любом изменении содержимого раздела «Key Capabilities» происходит предварительное стирание содержимого памяти ключа. Далее нажмите кнопку 3 – «выбор файла для загрузки в память ключа» (см. ***рис. 11***) панели управления «Secure Control». На экране появится окно, изображенное на ***рис. 14***, в котором стандартным образом указывается файл для загрузки в ключ. После выбора файла на панели управления «Secure Control» станет активной кнопка 4 – «загрузка выбранного файла в память ключа». После ее нажатия на экране появится окно, изображенное на ***рис. 16***, где требуется ввести параметры защиты от перезаписи памяти, которые будут запрошены в следующий раз при попытке перепрограммирования ключа. Работа с этим окном подробно рассмотрена в разделе 6.6. Следует только уточнить, что если ключ уже программировался, и при этом использовалась парольная защита от перезаписи, в поле «Current password» следует ввести установленный ранее пароль.

ВНИМАНИЕ!!! Если при последнем перепрограммировании ключа была активирована опция «Full Lock», дальнейшее перепрограммирование ключа будет невозможным!

После выбора необходимых опций следует нажать «OK» для начала процедуры загрузки, либо «Cancel» для отмены программирования. Процедура загрузки начинается со

стирания памяти ключа, после чего происходит загрузка программы и установка защиты от перезаписи (если Вы ее активируете) для следующей процедуры перепрограммирования. По завершении загрузки в память ключа программного кода на экране появится одно из сообщений изображенных на ***рис. 15***. Если программирование прошло успешно, в сообщении отобразится информация о времени прошивки FLASH-памяти. При программировании нескольких ключей одинаковых ключей процедуру выбора файла для загрузки повторять не требуется.

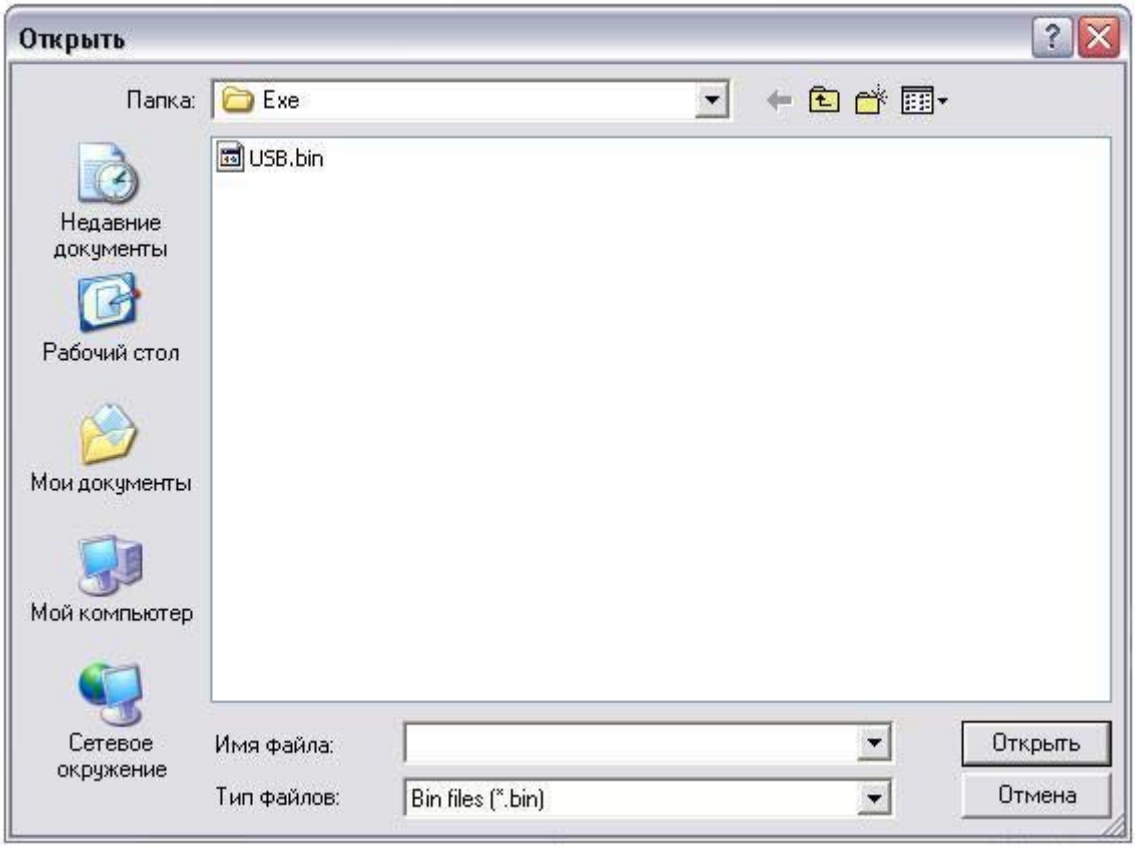

рис. 14. Окно выбора загружаемого в память ключа файла

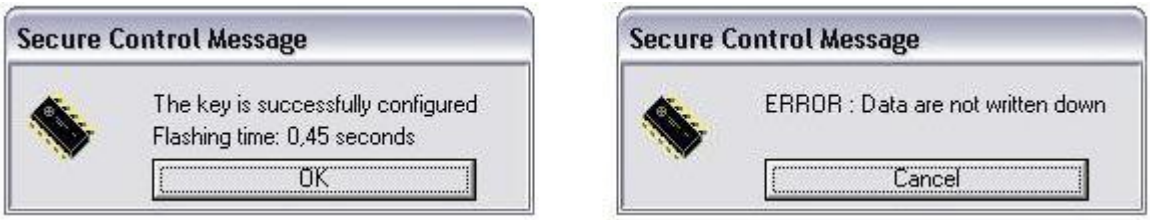

рис. 15. Основные сообщения о результате программирования ключа
Удачное завершение (слева), неудачная попытка программирования ключа (справа)

6.6. Использование защиты от перезаписи

Сразу следует уточнить, что защита от перезаписи содержимого памяти USB-ключей серии «LOCK» не имеет никакого отношения к надежности защиты программного обеспечения, а точнее говоря, устойчивости ко взлому. Функции защиты от перезаписи введены лишь для того, чтобы предотвратить возможность случайного стирания или перезаписи памяти ключей при некорректном обращении с ними. Как уже неоднократно отмечалось, чтение скрытых в ключе фрагментов защищаемого программного обеспечения невозможно ни при каких обстоятельствах, однако ключи могут быть многократно стерты и перепрограммированы. Это означает, что нерадивый пользователь, а возможно вирус, написанный им, может оказывать воздействие на USB-ключи, и как следствие произвести стирание содержимого памяти, после чего защищаемый программный продукт окажется неработоспособным. Защита от перезаписи содержимого памяти USB-ключей активируется непосредственно перед загрузкой в память ключа кода защищаемого программного обеспечения. Если вы указываете опцию «Full Lock» (см. ***рис. 16***), это будет означать, что по завершении загрузки в память ключа кода программы будет активирована полная защита от перезаписи, которая больше не позволит ничего поменять во FLASH-памяти ключа. Поэтому с этой опцией следует проявлять осторожность и использовать ее только в том случае, когда Вы полностью уверены, что программное обеспечение не содержит ошибок, и в дальнейшем не будут вноситься никакие изменения. Также есть возможность использовать парольную защиту от перезаписи. Длина пароля составляет 8 байтов. Пароль вводится в виде шестнадцатеричного значения с подтверждением в поля «New Password» и «Confirm». Если до этого в ключе уже использовался пароль, его следует ввести в поле «Current password», в противном случае изменение содержимого FLASH-памяти ключа будет невозможным. Следует отметить, что ключи имеют защиту от подбора пароля, поэтому заниматься этим бессмысленно. Если ключ получит неверный пароль, он заблокирует прием пароля на 2 секунды (даже верного пароля, отвечая, что пароль неверный). Теперь можно посчитать, какое время потребуется на подбор пароля.

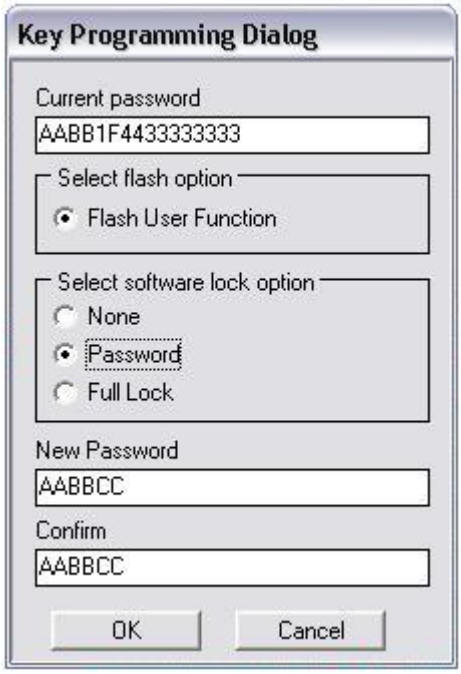

рис. 16. Диалог программирования ключа

Astroma Ltd.

http://www.astroma-spb.ru

Адрес: **Россия 192007, г. Санкт-Петербург, ул. Курская, д. 21 лит. "Б"**
Тел./Факс: **+7(812)320-21-90**
E-mail: **info@astroma-spb.ru** (общие вопросы)
support@astroma-spb.ru (техническая поддержка)