Способ защиты программ в микроконтроллерах C8051FXXX фирмы Silicon Labs

Алексей Кузьминов (Москва)

В статье описывается простой способ защиты флэш-памяти микроконтроллеров от несанкционированного доступа, заключающийся в загрузке программы по адресу регистра блокировки. Программа состоит из одного-двух байт и может быть записана в микроконтроллер стандартным способом с помощью USB-DEBUG-адаптера. В качестве примера приводятся тексты и результаты работы двух программ, предназначенных для защиты памяти микроконтроллеров C8051F321 и C8051F067.

Введение

Для защиты флэш-памяти микроконтроллера от несанкционированного доступа требуется обнулить регистры блокировки чтения и записи/стирания. Стандартным способом такая запись данных может быть организована либо из программы пользователя (что достаточно сложно), либо с помощью интерфейсов JTAG или C2 (в зависимости от типа микроконтроллера), что ещё сложнее, т.к. для этого требуется манипуляция сигналами этих интерфейсов. Возможность защиты флэш-памяти, к сожалению, не предусмотрена в программном обеспечении (IDE) USB-DEBUG-адаптера, с помощью которого программируются микроконтроллеры.

Для защиты памяти программ от несанкционированного доступа в микроконтроллерах C8051FXXX фирмы Silicon Labs, совместимых с I8051, предусмотрены специальные регистры блокировки. В одних микроконтроллерах, как правило, простых и программируемых с помощью интерфейса C2, существует единственный регистр блокировки чтения и записи/стирания, который расположен в старших адресах памяти программ. Например, в микроконтроллере C8051F321 [1] с максимальным объёмом программной памяти 16 Кб (3fffh байт) такой регистр блокировки расположен по адресу 3dffh. В более сложных микроконтроллерах, программируемых с помощью интерфейса JTAG, предусмотрено два подобных регистра: один – для блокировки чтения, второй – для блокировки записи/стирания. Адреса этих регистров также расположены в старших адресах памяти программ и следуют один за другим. Например, в микроконтроллере C8051F067 [2] с максимальным объёмом памяти 32 Кб (7fffh байт) адрес регистра блокировки записи/стирания равен 7ffeh, а регистр блокировки чтения расположен по адресу 7fffh.

Если требуется защитить от несанкционированного доступа одни секторы памяти программ, оставив незащищёнными другие, то в биты, соответствующие защищаемым секторам, должны быть записаны нули (после полного стирания памяти программ во всех битах этих регистров записаны единицы). Если же требуется защитить всю память программ, то регистры защиты должны быть полностью обнулены. Такая защита памяти программ является оптимальной, и дальнейшее изложение посвящено именно ей.

Для обнуления регистров защиты существует два официальных способа, приведённых в [3] и в справочном листке на конкретный микроконтроллер. Первый способ – из программы пользователя, второй – с помощью либо интерфейса C2, либо JTAG [4], в зависимости от того, по какому интерфейсу программируется микроконтроллер.

Первый способ защиты, на взгляд автора, неприемлем, поскольку программа пользователя должна содержать только тот код, который требуется для выполнения её задачи, а дополнительная подпрограмма защиты не только отнимает ресурсы памяти основной программы, но и является потенциально опасным кодом. Кроме того, такая сложная подпрограмма защиты требует определённого времени на разработку, а решение о защите той или иной программы должен принимать сам пользователь. Поэтому способ защиты должен быть не связанным с пользовательской программой и применяться при необходимости. Второй способ защиты [3, 4], на взгляд автора, ещё сложнее.

В возможностях программного обеспечения (IDE) для USB-DEBUG-адаптера [5], применяемого, в частности, для программирования микроконтроллеров, защита памяти программ микроконтроллера, к сожалению, также не предусмотрена. В то же время защитить память программ от несанкционированного доступа иногда просто необходимо.

Таким образом, появилась задача обнулить регистры защиты, но не путём записи данных (в данном случае нулей). Как оказалось, сделать это довольно легко.

Основная идея заключается в следующем. Если адрес регистра защиты известен (например, у микроконтроллера C8051F321 это 3dffh), а байт данных равен нулю, то путём записи программы, состоящей из одного оператора NOP (код операции которого равен 00h), в программную память микроконтроллера с адреса 3dffh регистр защиты можно обнулить. Несмотря на то что программа состоит всего из одного (нулевого) байта, она может быть запрограммирована в микроконтроллер стандартным способом с помощью USB-DEBUG-адаптера. Но сможет ли такая программа защищать память микроконтроллера от несанкционированного доступа?

Программы защиты памяти микроконтроллеров C8051F321 и C8051F067

Наиболее подходящим языком программирования для написания такой программы, по мнению автора, является ассемблер. Далее приведены две программы, написанные на ассембле-

ре и предназначенные для защиты памяти микроконтроллеров C8051F321 и C8051F067.

Рассмотрим программу защиты памяти микроконтроллера C8051F321. Эта программа должна обнулить единственный регистр блокировки чтения и записи/стирания, расположенный по адресу 3dffh. Начальный адрес программы в памяти микроконтроллера определяется настройкой ассемблера. Например, если используется ассемблер 2500 A.D. Macro Assembler v.4.02a, то для того чтобы расположить программу, например, с адреса 3dffh, необходимо установить .org 3dffh. Если используется ассемблер Keil A51 Macro Assembler v.6.14, эта настройка имеет вид CSEG AT 3DFFH. Что касается кода операции nop, то он всегда равен 00h, т.к. зависит не от типа транслятора, а от самого ассемблера I8051-совместимых микроконтроллеров.

Таким образом, программа защиты памяти микроконтроллера C8051F321 на 2500 A.D. Macro Assembler состоит из одного оператора (nop) и трёх параметров и выглядит следующим образом:

```
.CODE
.org 3dffh
nop
.end
```

Назовём эту программу zas321.asm. Для трансляции этой программы и получения её в Intel-Hex-формате потребуется *.bat-файл следующего содержания:

```
x8051 zas321.asm -t -d
pause
link -c zas321.obj
pause
```

Назовём этот файл zas321.bat.

Результат трансляции zas321.asm с ассемблера показывает (см. рис. 1), что по адресу 3dffh памяти записан нулевой байт данных (строка 3). По сборке программы видно, что размер (Size) кода программы равен одному байту, программа расположена по адресу 3dffh, а файл был создан в hex-формате zas321.hex. Текст файла приведён ниже:

```
:013DFF0000C3
:00000001FF
```

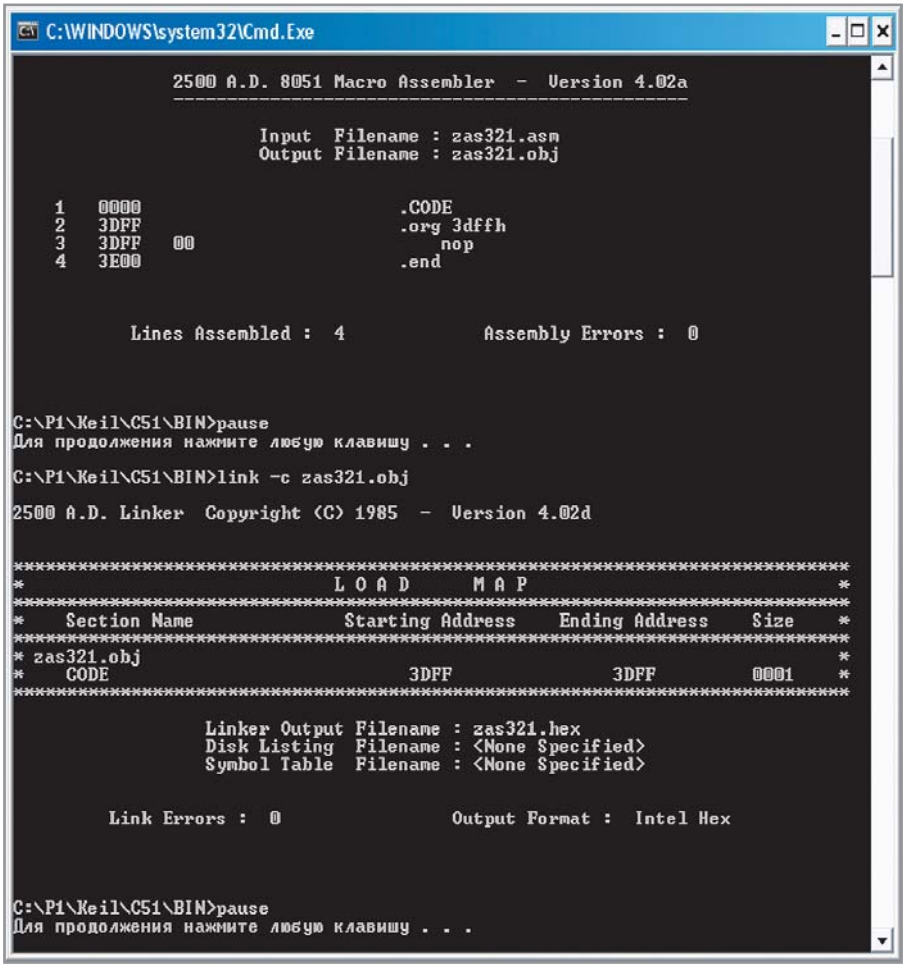


Рис. 1. Результат трансляции программы zas321.asm с помощью файла zas321.bat

Программа защиты памяти микроконтроллера C8051F321 на ассемблере Keil A51 Macro Assembler v.6.14 выглядит следующем образом:

```
CSEG AT 3DFFH
nop
END
```

Назовём эту программу zas321.A51. Для трансляции этой программы и получения её в Intel-Hex-формате необходим *.bat-файл следующего содержания:

```
a51.exe zas321.a51 print
object(zas321.obj)
pause
bl51.exe zas321.obj to zas321
ixref print(bl_zas321.lst)
oh51 zas321
pause
```

Назовём этот файл zas321_A51.bat. Фрагмент листинга результата ассемблирования программы zas321.A51 с помощью файла zas321_A51.bat приведён ниже:

```
LOC OBJ LINE SOURCE
---- 1 CSEG AT 3DFFH
3DFF 00 2 nop
 3 END
_A51 MACRO ASSEMBLER ZAS321
```

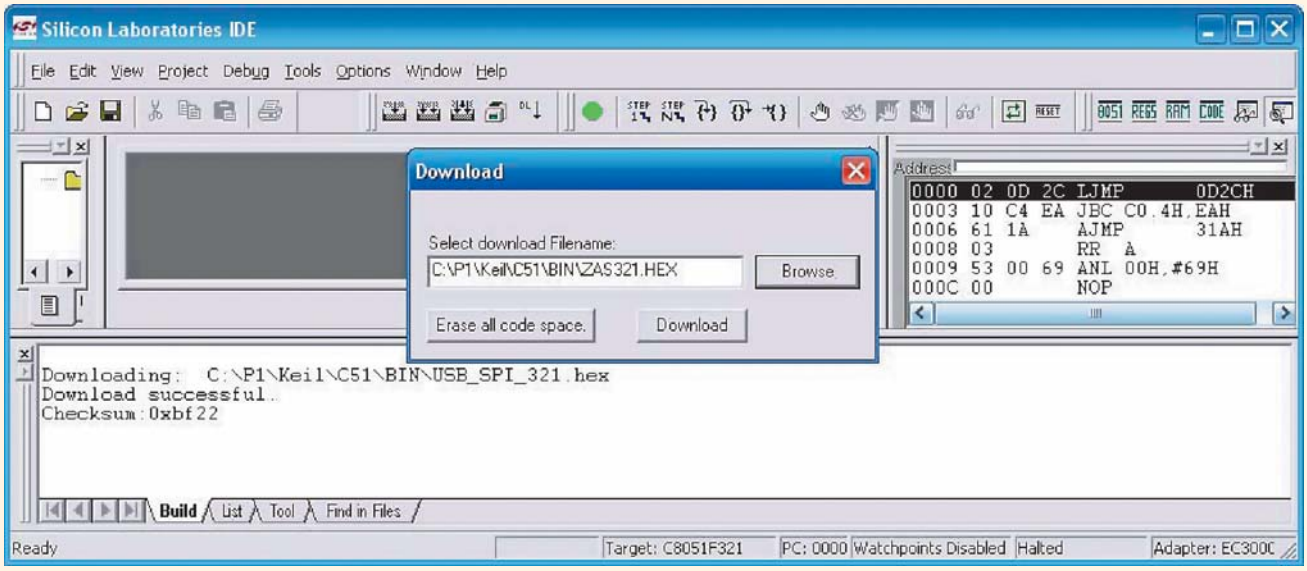
Видно, что по адресу памяти 3dffh записан нулевой байт данных.

Если перед запуском файла zas321_A51.bat удалить файл zas321.hex предыдущей трансляции, полученный с помощью файла zas321.bat, то после запуска файла zas321_A51.bat вновь сформируется файл zas321.hex, который будет точно таким же:

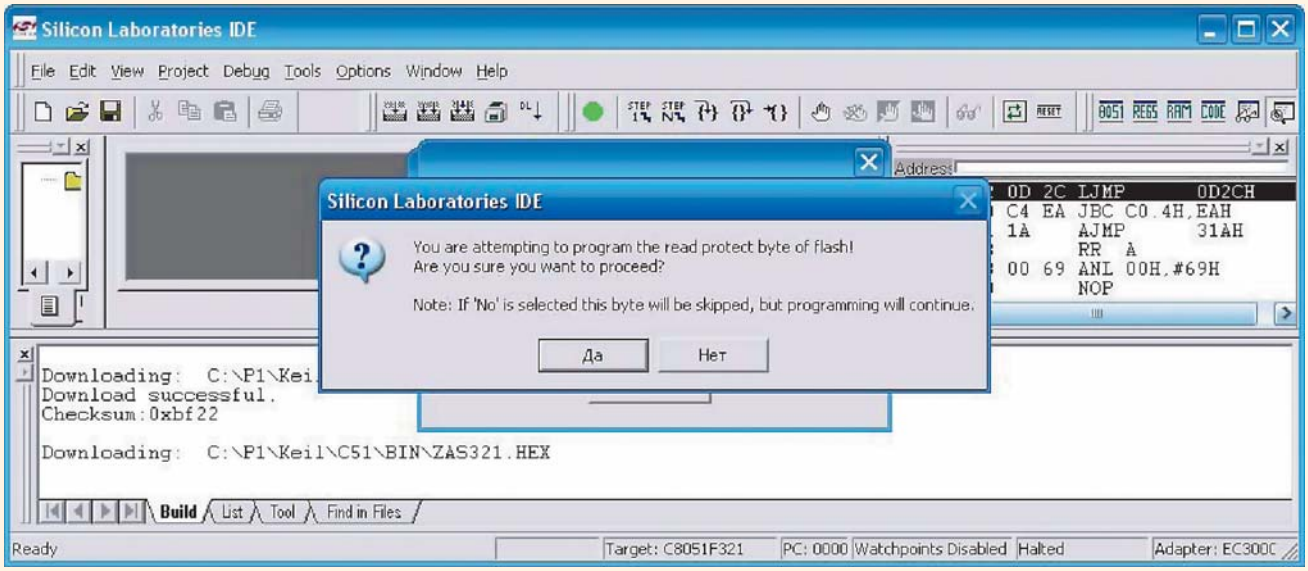
```
:013DFF0000C3
:00000001FF
```

Это означает, что, независимо от транслятора, результатом будет программа zas321.hex.

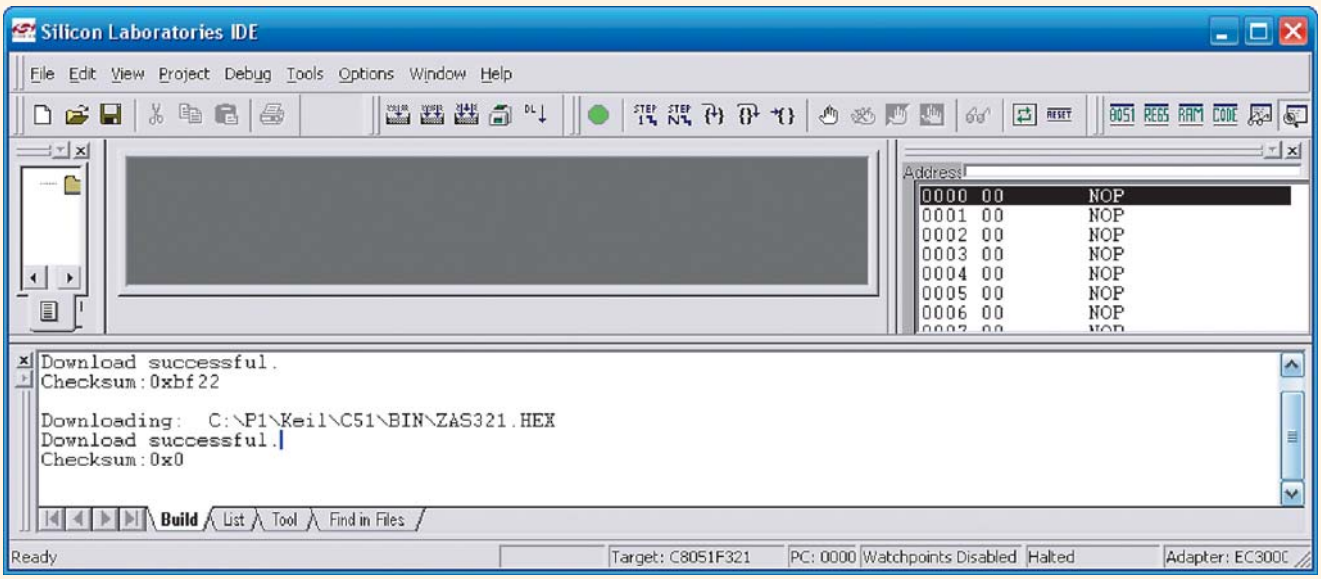
Теперь рассмотрим программу защиты памяти микроконтроллера C8051F067. Эта программа должна обнулить два регистра: регистр блокировки записи/стирания, расположенный по адресу 7ffeh, и регистр блокировки чтения, расположенный по адресу 7fffh. Программа на языке 2500 A.D. Macro Assembler выглядит следующим образом:



а)

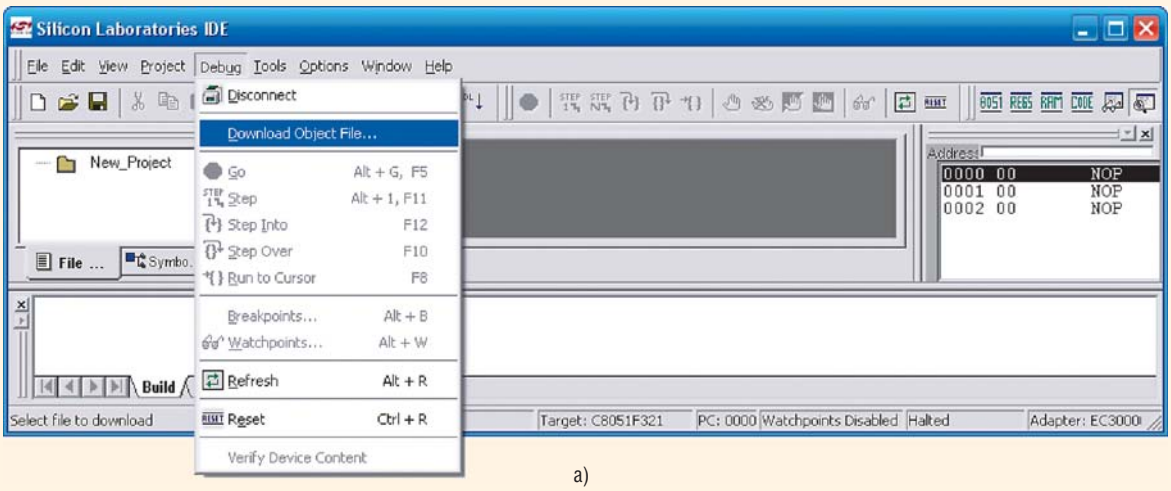


б)

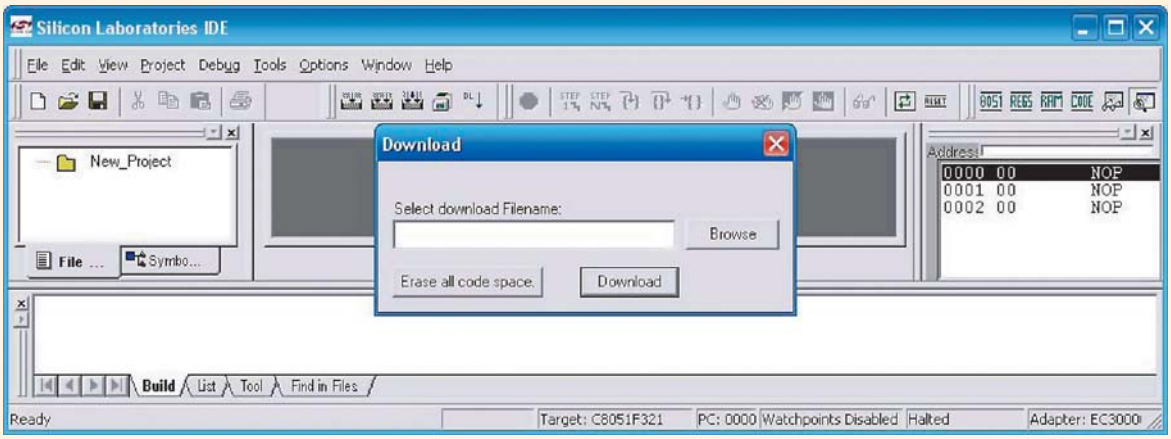


в)

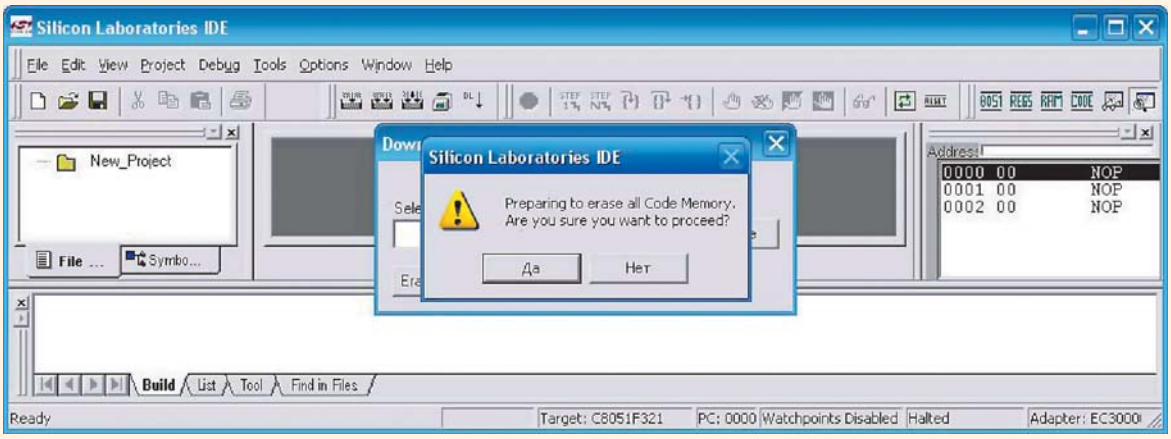
Рис. 2. Работа программы zas321.hex по защите памяти микроконтроллера C8051F321



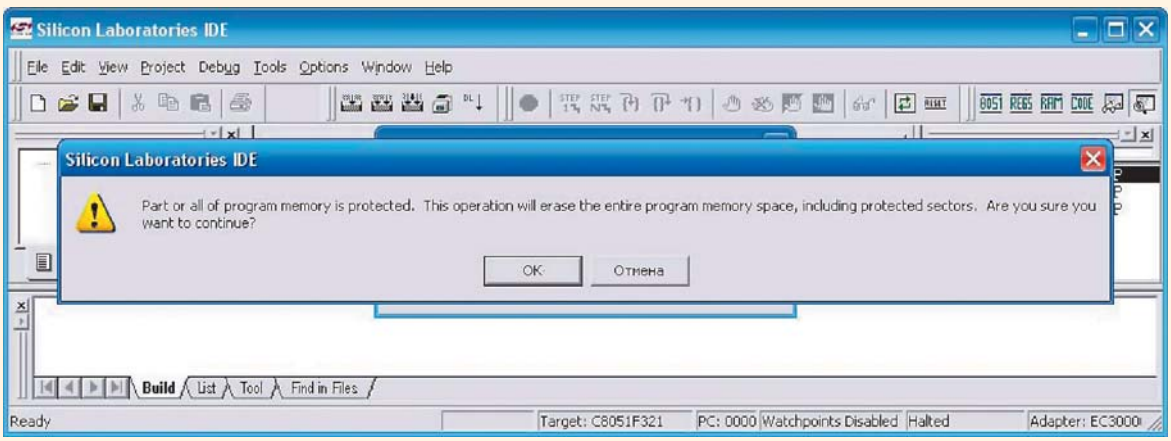
а)



б)

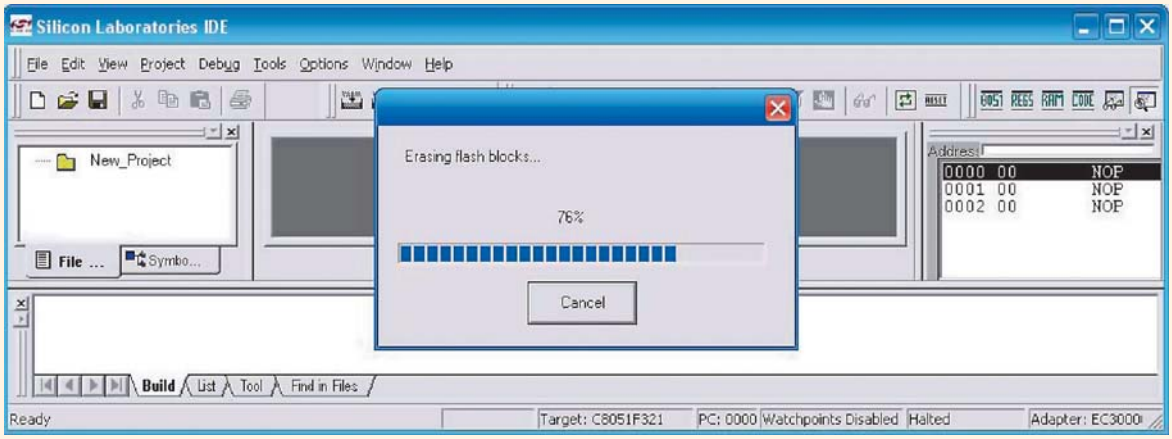


в)

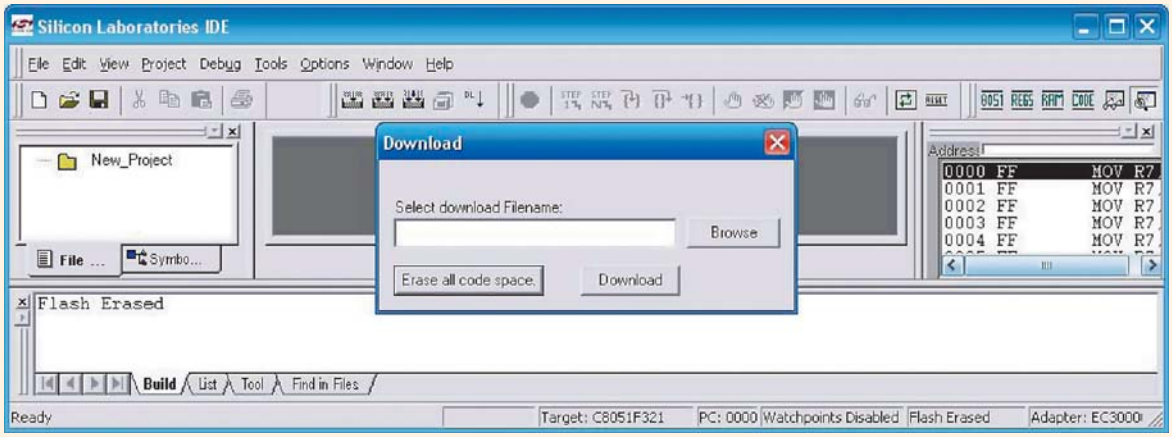


г)

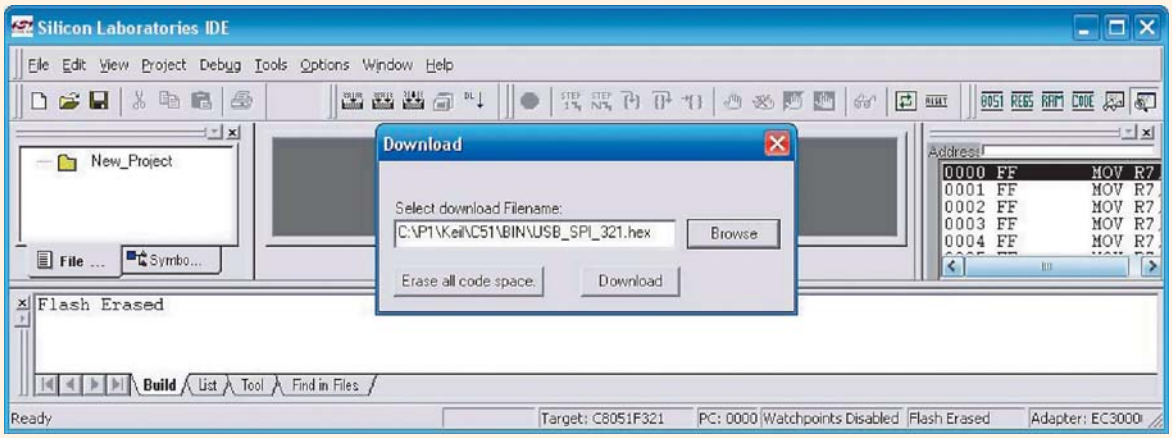
Рис. 3. Первый метод стирания заблокированной памяти программ микроконтроллера C8051F321 для её разблокировки



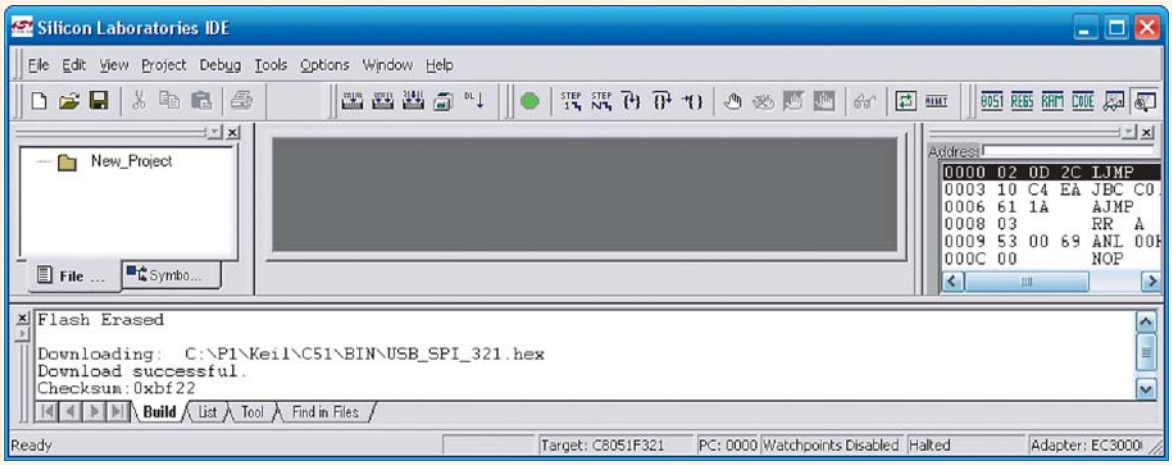
д)



е)

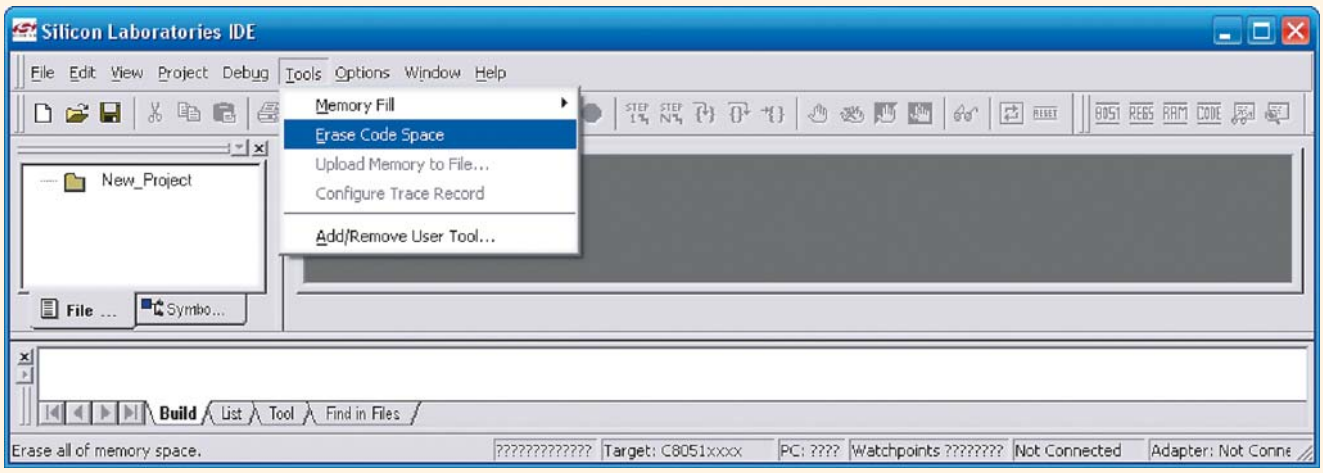


ж)

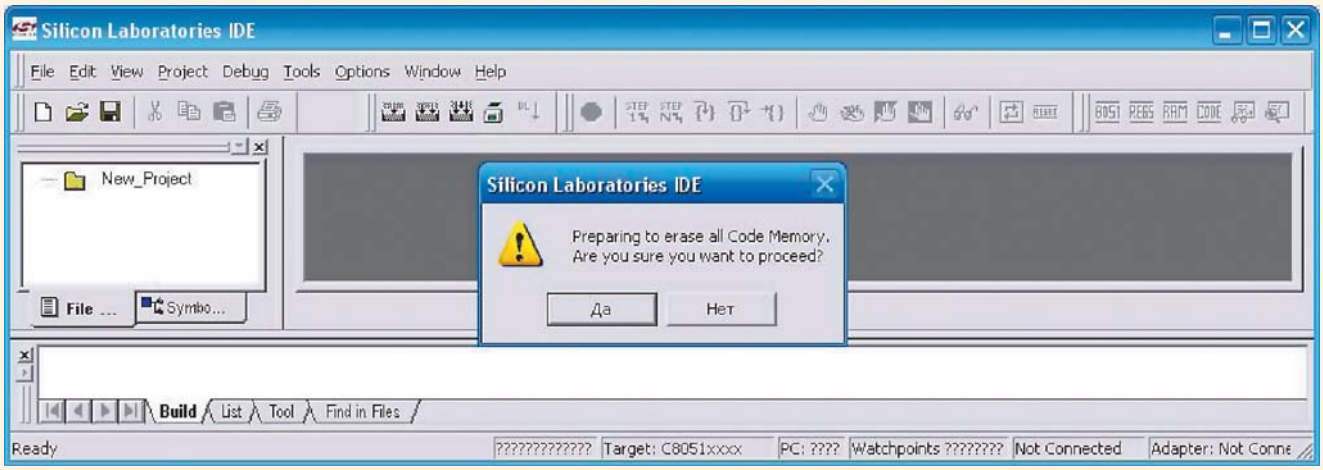


з)

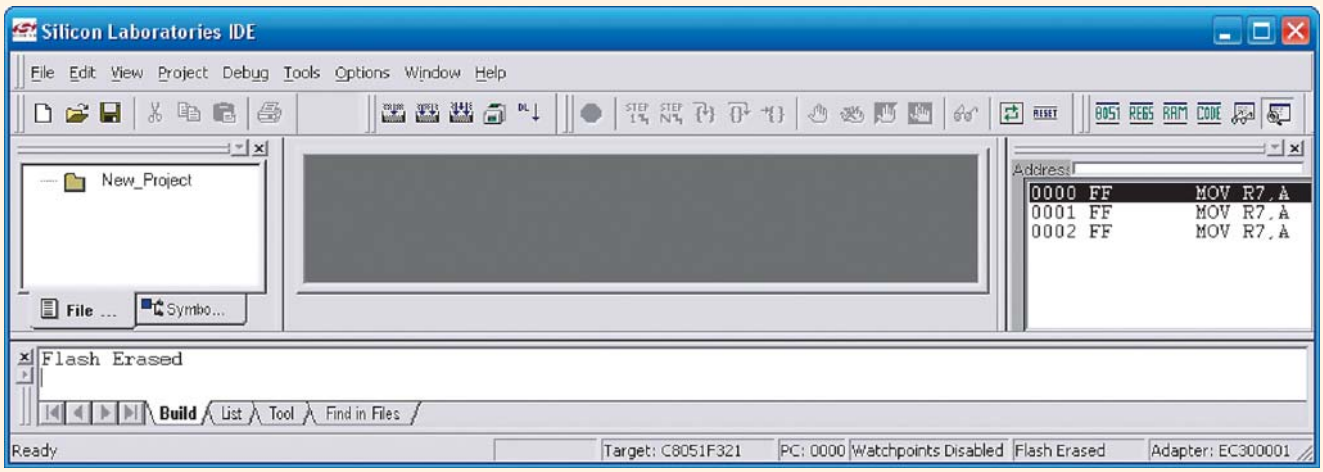
Рис. 3. Первый метод стирания заблокированной памяти программ микроконтроллера C8051F321 для её разблокировки (окончание)



а)



б)



в)

Рис. 4. Второй метод стирания заблокированной памяти программ микроконтроллера C8051F321 для её разблокировки

```
.CODE
.org 7ffeh
nop
nop
.end
```

Назовём эту программу zas067.asm. Для трансляции этой программы и получения её в Intel-Hex-формате необходим *.bat-файл следующего содержания:

```
x8051 zas067.asm -t -d
pause
link -c zas067.obj
pause
```

Назовём этот файл zas067.bat.

Результат трансляции программы zas067.asm с помощью этого *.bat-файла (экранная форма аналогична рисунку 1) даёт следующую информацию: по адресу памяти 7ffeh записан нулевой байт данных (строка 3), такой же нулевой байт данных записан по адресу 7fffh (строка 4), размер (Size) кода программы равен 2 бай-

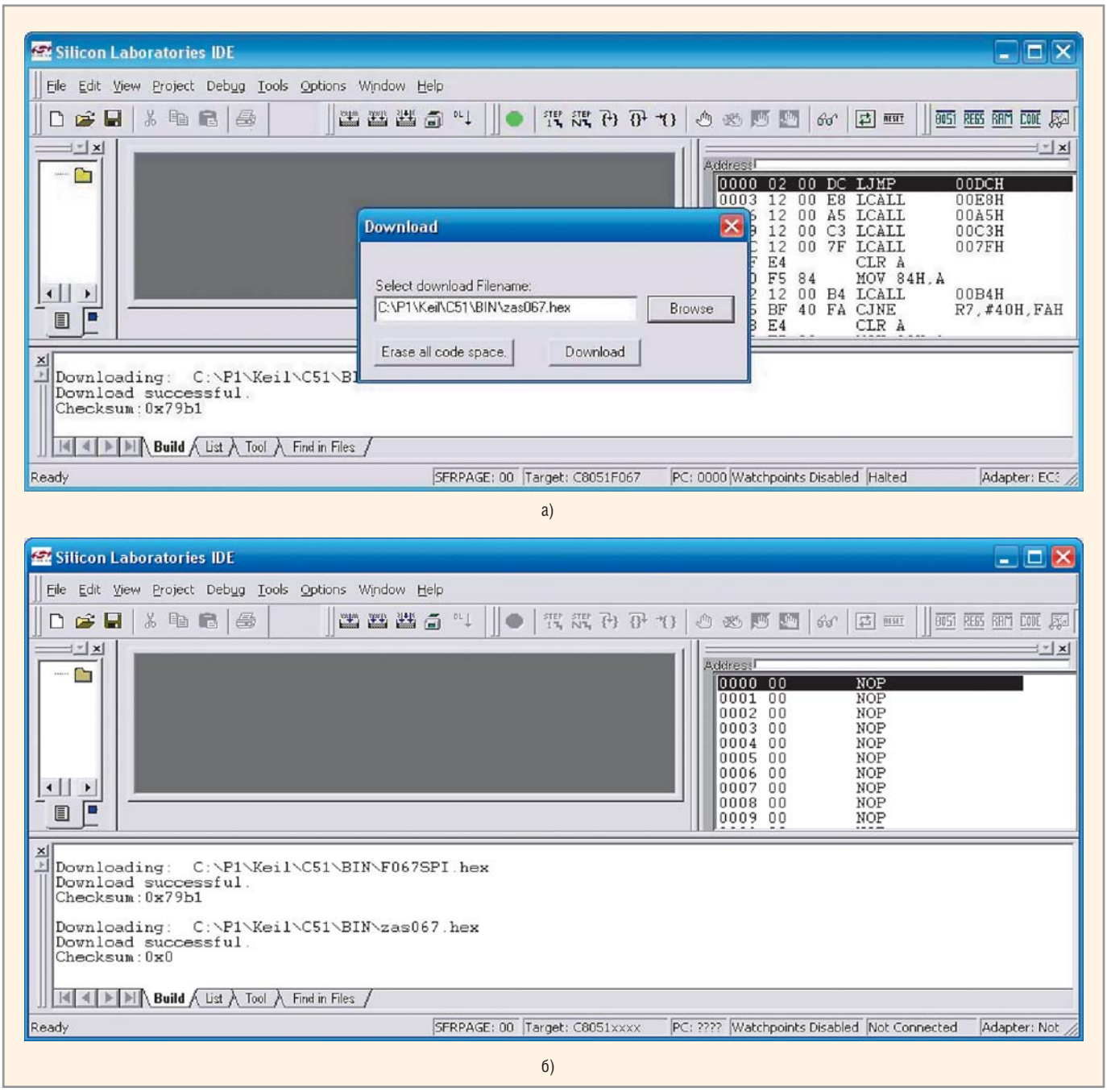


Рис. 5. Работа программы zas067.hex по защите памяти микроконтроллера C8051F067

а, б – защита памяти, в, г – стирание памяти для её разблокировки

там, файл был создан в HEX-формате zas067.hex. Текст файла приведён ниже:

```
:027FFE00000081
:00000001FF
```

Программа защиты памяти микроконтроллера C8051F067 на ассемблере Keil A51 Macro Assembler v.6.14 выглядит следующем образом:

```
CSEG AT 7FFEH
nop
nop
END
```

Назовём эту программу zas067.A51. Для трансляции этой программы и получения её в Intel-Hex-формате необходим *.bat-файл следующего содержания:

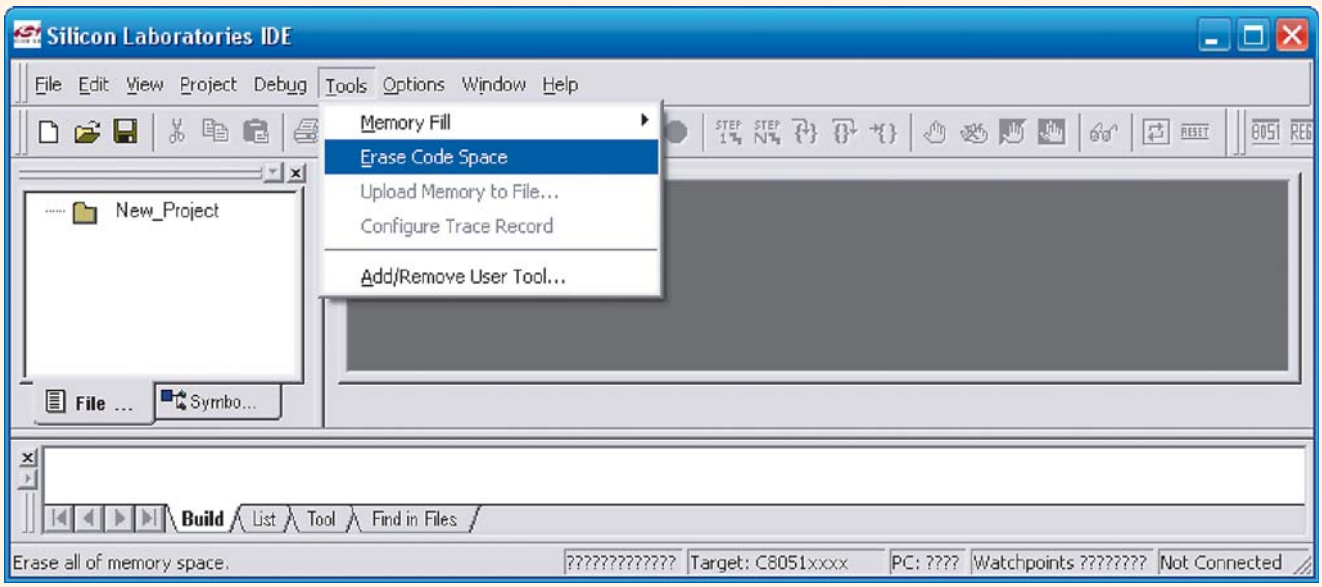
```
a51.exe zas067.a51 print
object(zas067.obj)
pause
bl51.exe zas067.obj to zas067
ixref print(bl_zas067.lst)
oh51 zas067
pause
```

Назовём этот файл zas067_A51.bat. Фрагмент листинга результата ассемблирования программы zas067.A51 с помощью файла zas067_A51.bat приведен ниже:

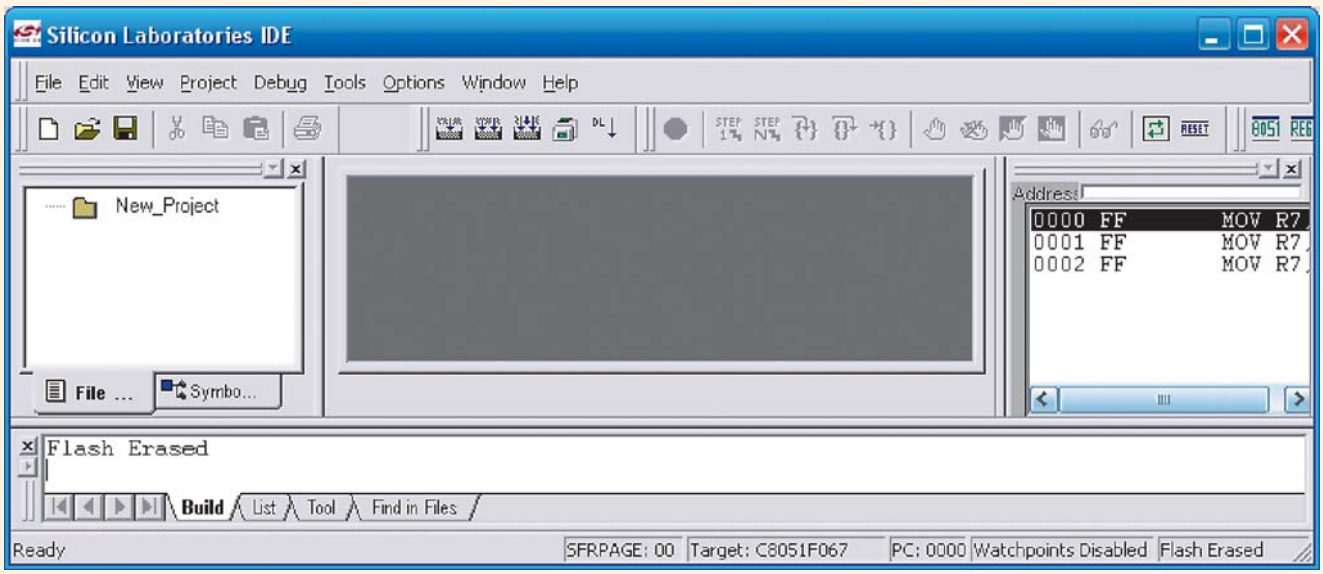
```
LOC OBJ LINE SOURCE

---- 1 CSEG AT 7FFEH
7FFE 00 2 nop
7FFF 00 3 nop
 4 END
_A51 MACRO ASSEMBLER ZAS067
```

Видно, что по адресам памяти 7ffeh и 7fffh записаны нулевые байты данных.

После запуска *.bat-файла zas067_A51.bat сгенерируется программа zas067.hex, имеющая необходимое содержание:



в)



г)

Рис. 5. Работа программы zas067.hex по защите памяти микроконтроллера C8051F067 (окончание)

а, б – защита памяти, в, г – стирание памяти для её разблокировки

```
:027FFE00000081
:00000001FF
```

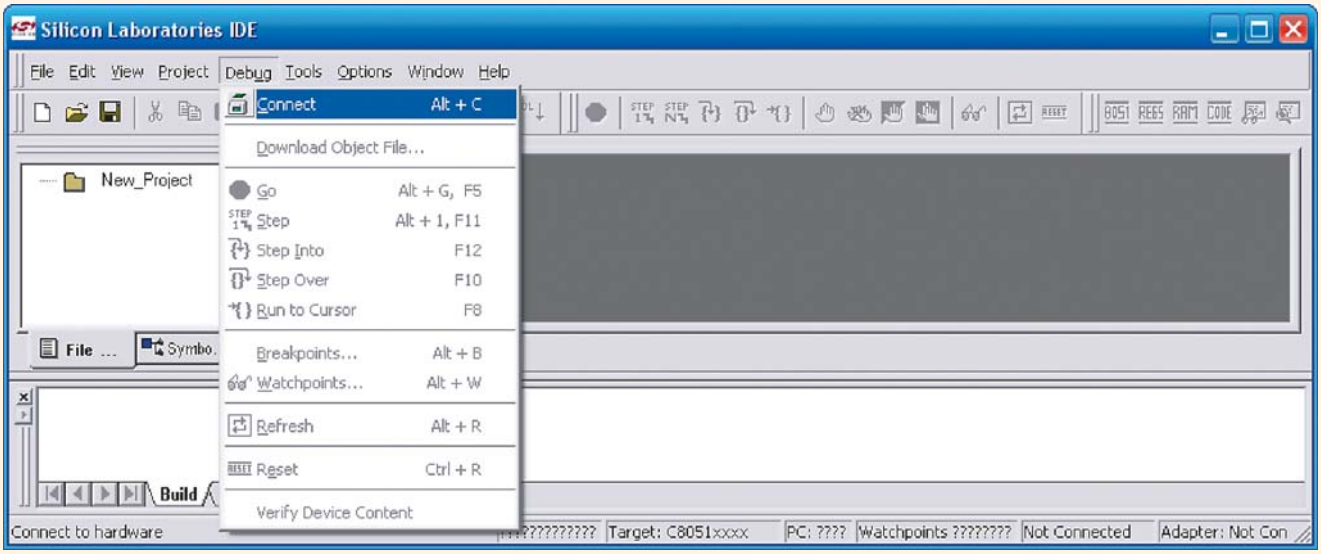
После получения программ в hex-форматах (zas321.hex и zas067.hex) их необходимо записать в соответствующие микроконтроллеры после основной (защищаемой) программы.

Результаты работы программ защиты памяти микроконтроллеров C8051F321 и C8051F067

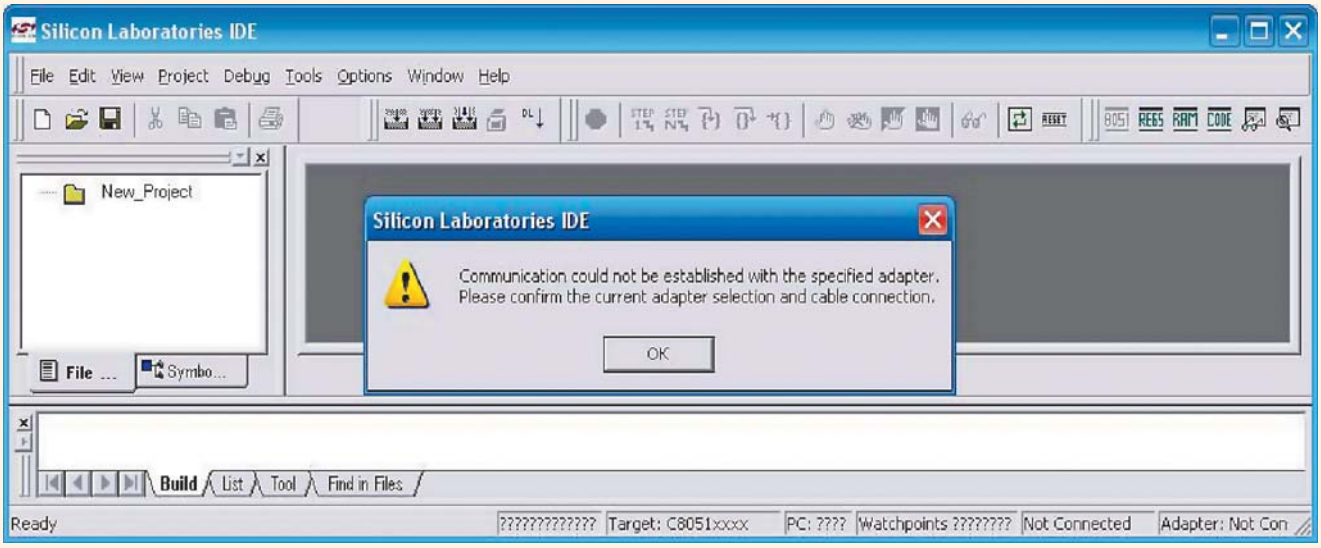
Методика проверки работоспособности программ защиты состояла в следующем. В микроконтроллер загружалась заведомо работоспособная программа, подлежащая защите. Затем загружалась программа защиты (zas321.hex или zas067.hex). После этого проверялась работоспособность защищённой программы. Далее производилась разблокировка памяти микроконтроллера (путём полного стирания всей его программной памяти) с целью восстановления возможности записи в память новой программы.

Вначале проверялась работоспособность программы zas321.hex. Для этого в микроконтроллер C8051F321 загружалась защищаемая программа USB_SPI_321.hex [5], затем – программа защиты zas321.hex. Из рисунков 2а, 2б видно, что перед загрузкой программы zas321.hex в памяти программ находятся коды программы USB_SPI_321.hex (в правой части экранных форм). После загрузки программы zas321.hex (см. рис. 2в) в памяти программ микроконтроллера видны только коды операций NOP, т.е. программа zas321.hex полностью защитила память программ микроконтроллера от несанкционированного доступа.

Для разблокировки памяти микроконтроллера C8051F321 необходимо стереть всю его программную па-



а)



б)

Рис. 6. Демонстрация неверной попытки разблокировки памяти микроконтроллера C8051F067

мять. Это можно сделать двумя способами.

Первый способ заключается в следующем. После подключения микроконтроллера к USB-DEBUG-адаптеру и включения питания микроконтроллера выбираем параметр Debug → Connect. После этого в нижней строке экранной формы определяется микроконтроллер C8051F321 (Target: C8051F321), а в правой части экранной формы отображается содержимое его памяти программ, которое в данном случае заблокировано и содержит только коды операций NOP. Выбираем параметр Download Object File (см. рис. 3а). В образовавшемся окне нажимаем кнопку с надписью Erase all code space (см. рис. 3б). Получаем два предупреждения (см. рис. 3в, 3г). При утвердительном ответе («Да» – рис. 3в и OK – рис. 3г) стирается память программ микроконтроллера (см. рис. 3д) и выводится окно с предупреждением, что память программ стёрта. При нажатии кнопки OK в этом окне память программ уже разблокирована (см. рис. 3е), поскольку в ней записаны одни единицы (коды ffh). Выбрав соответствующую программу и нажав кнопку Download см. рис. 3ж), в микроконтроллер можно загрузить новую программу, в данном случае, USB_SPI_321.hex (см. рис. 3з). В этой экранной форме справа показаны коды загруженной программы.

Второй способ стирания всей памяти программ для разблокировки микроконтроллера состоит в следующем. После подключения микроконтроллера к USB-DEBUG-адаптеру и включения питания микроконтроллера выбираем параметр Tools → Erase Code Space (см. рис. 4а). Получив предупреждение (см. рис. 4б) и утвердительно ответив на него, запускаем процесс стирания памяти, после чего в неё записываются все единицы (ffh) (см. правую часть рис. 4в). Кроме того, в нижней части этой экранной формы определяется подключенный микроконтроллер (Target: C8051F321). Теперь память микроконтроллера разблокирована, и он готов к загрузке новой программы.

Ниже приведены результаты проверки работоспособности программы zas067.hex, предназначенной для защиты памяти программ микрокон-

троллера C8051F067. Вначале в этот микроконтроллер была загружена основная (защищаемая) программа F067SPI.hex [6]. Затем для её защиты была загружена программа zas067.hex. Из рисунков 5а, 5б видно, что программа zas067.hex защищает память микроконтроллера C8051F067 аналогично программе zas321.hex.

Для разблокировки памяти микроконтроллера C8051F067 необходимо стереть всю его программную память. Сделать это можно *единственным* способом, выбрав параметр Tools → Erase Code Space (см. рис. 5в). По окончании стирания всей памяти микроконтроллера его память будет разблокирована (см. рис. 5г), т.е. в ней записаны одни единицы (ffh). С этого момента микроконтроллер готов к загрузке новой программы.

Если стирание памяти микроконтроллера C8051F067 осуществить способом Debug → Connect (см. рис. 6а), то на монитор будет выведено предупреждение о невозможности соединения с USB-DEBUG-адаптером (см. рис. 6б), загорятся оба светодиода, и всё «зависнет». Единственным способом выхода из этого состояния является следующий: выключить питание микроконтроллера, выйти из программы обслуживания (IDE) USB-DEBUG-адаптера, вынуть из него и вставить обратно кабель USB.

Результаты работы программ защиты памяти микроконтроллеров C8051F321 и C8051F067 позволяют сделать следующие выводы:

- описанные программы zas321.hex и zas067.hex надёжно защищают память микроконтроллеров C8051F321 и C8051F067 от несанкционированного доступа;
- отсутствие возможностей защиты памяти микроконтроллеров в программном обеспечении (IDE) USB-DEBUG-адаптера, на взгляд автора, является недоработкой фирмы Silicon Labs, хотя на защиту памяти программ микроконтроллеров IDE реагирует с помощью двойного предупреждения (см., например, рис. 3в, 3г). Это означает, что метод защиты памяти, приведённый автором в статье, известен разработчиками программного обеспечения USB-DEBUG-адаптера. Однако не ясно, почему он не описан в документации фирмы Silicon Labs.

Заключение

Описанный в статье способ загрузки регистров блокировки памяти программ с помощью программного кода намного проще их загрузки способом записи данных и позволяет надёжно защищать память микроконтроллеров C8051FXXX фирмы Silicon Labs от несанкционированного доступа.

Литература

1. www.silabs.com/../C8051F320/1.pdf.
2. www.silabs.com/../C8051F06X.pdf.
3. Flash security user guide. www.silabs.com/../AN120.pdf.
4. Programming flash through the JTAG interface. www.silabs.com/../AN105.pdf.
5. www.silabs.com/../USB Debug Adapter User's Guide.pdf.
6. *Кузьминов А.* Преобразователь интерфейсов USB–SPI с гальванической развязкой. Современная электроника. 2012. № 1, 2.