Отладка загружаемого кода

Разработка и первоначальная отладка загружаемого кода производится на компьютере. Для этого можно использовать любую IDE и отладчик языка C.

Основная проблема состоит в том, что отлаживать уже загруженный код затруднительно, поскольку нет возможности «залезть» отладчиком в контроллер ключа. Поэтому первоначально отлаживают сам алгоритм загружаемого кода.

Загруженный в ключ код имеет ограниченные возможности для трассировки. Например, нельзя вывести трассу на консоль или записать в файл. Однако некоторые средства все же есть. Для этой цели можно использовать **функции управления светодиодом**. Сигналы, подаваемые с его помощью, можно применять в качестве признаков прохождения тех или иных веток кода.

Как вариант, можно использовать принудительный возврат из загружаемого кода с соответствующим кодом возврата и передачей необходимых для отладки данных через буфер вывода.

Методы отладки кода такие же, как и при разработке на PC. Если в загружаемом коде используются вызовы Guardant Code API, то для отладки не нужно загружать этот код в ключ: можно использовать входящую в комплект разработчика отладочную библиотеку.

Описание отладочной библиотеки

Отладочная библиотека представлена двумя частями:

- Модуль для загрузки в электронный ключ,
- Динамическая библиотека, содержащая функции, прототипы которых аналогичны тем, что доступны для загружаемого кода внутри электронного ключа (GcaXXX и GccaXXX)

Порядок работы с отладочной библиотекой таков:

1. При помощи **GrdUtil.exe** создается файл маски, в котором один из алгоритмов представляет собой отладочный модуль загружаемого кода – **DebugModule.bin**. При этом нужно убедиться, что соответствующий модулю **bmap-**файл находится в той же директории. Можно взять готовый файл маски **DebugMask.nsd** из примера и изменить его для использования в собственном приложении. От файла маски зависит, какой номер будет у алгоритма, содержащего загружаемый код.
2. Полученный файл маски с отладочным модулем прошивается в электронный ключ при помощи **GrdUtil.**
3. К проекту загружаемого кода на PC подключается отладочная библиотека **gcaapidll.dll**. Для этого используется библиотека экспортов **gca apidll.lib**. Файл **GcaAPIdll.h** содержит описание прототипов функций **GcaXXX/GccaXXX.**
4. Перед вызовами функций **GcaXXX/GccaXXX** из **gcaapidll.dll** в исходном коде следует разместить вызов макроса **DEBUGDLL_ INIT (hHandle, dwAlgoNum)**, который настраивает библиотеку для работы с текущим контекстом Guardant API. Макрос осуществляет привязку библиотеки к используемому контексту Guardant API и открытому ключу, также ему передается номер аппаратного алгоритма, в который был загружен отладочный модуль.
5. Если код, предполагаемый для размещения в электронном ключе, использует вызовы внешнего Guardant API, то их требуется заменить на соответствующие вызовы Guardant Code API. В данном случае функции импортируются из отладочной библиотеки. Если же код содержит функции Guardant API, не имеющие прямых аналогов в Guardant Code API, требуется создать эквивалентные им конструкции из доступных функций.

Следует принимать во внимание, что ни сама отладочная библиотека, ни отладочный модуль не содержат логики работы функций. Они являются всего лишь своеобразным «туннелем», через который параметры вызова функций передаются в электронный ключ и возвращаются обратно.

Пример использования макроса DEBUGDLL_INIT:

```
main()
{
// Хэндл ключа, в котором запускается загруженный пользователем код:
HADNLE hGrd;

// Инициализация API и подключение к электронному ключу
...

#ifdef DEBUG
// Инициализация DLL
DEBUGDLL_INIT(hGrd, 1);
// hGrd – хендл открытого электронного ключа.
// 1 – номер аппаратного алгоритма, в котором находится
// загруженный отладочный модуль
#endif

// Передача параметра hGrd необязательна.
GcaGetRandom(0, &iodata[i]);

return 0;
}
```

Также стоит отметить, что функции **GcaExit()** и **GcaLedOn()/ GcaLedOff()** не могут работать в отладочном режиме. Первая – из-за того, что результат ее работы просто нельзя зафиксировать, а функции управления светодиодом – из-за того, что сразу после их вызова работа кода будет завершаться, при этом индикатор просто зажигается вновь.

Использование отладочной библиотеки демонстрируется в **примере №19** (см. Краткая характеристика примеров).