Примеры использования загружаемого кода

Каждый пример находится в директории установки комплекта разработчика Guardant (по умолчанию **"Program Files\Guardant\SDK <версия>\Samples\Loadable Code API"**), в отдельной папке вида **XX - SampleName**, и состоит из двух частей:

- *Проект Visual Studio 2005 .NET* в папке **Win32**, демонстрирующий использование Guardant API (т. е. необходимую инициализацию и запуск загруженного кода)
- *Пример загружаемого кода* в папке **Loadable Code**, содержащий проект GCC-ARM в виде исходного кода на C и **makefile**. При сборке примера получаются ***.bin** и ***.bmap** файлы, которые необходимо указывать **GrdUtil** для генерации **GCEXE**-файла соответствующего алгоритма.

> **Важно!**
> Для корректной работы примеров **Guardant Code** необходимо записать **Mask.nsd** в ключ при помощи **GrdUtil.**

Структура используемого файла маски

Файл маски (**Mask.nsd**), используемый в примерах для ключей **Guardant Code / Code Time**, отличается от файла маски **GrdUtil** по умолчанию (**default.nsd**) и имеет следующую структуру, одинаковую для всех примеров:

1. Алгоритм #0 AES128
2. Алгоритм #01 SHA256
3. Алгоритм #02 AES128, зависит от ID ключа
4. Защищенная ячейка #03 – только для чтения
5. Защищенная ячейка #04 – для чтения и записи
6. Алгоритм #05 AES128 Demo
7. Алгоритм #06 SHA256 Demo
8. Защищенная ячейка #07 – Таблица лицензий
9. Алгоритм #08 – ECC160
10. Алгоритм #09 Загружаемый код №– Занимает 4 сектора. В него по очереди загружаются все примеры

По умолчанию файл маски **Mask.nsd** находится в директории **"Program Files\Guardant\SDK <версия>\DEMONVK\Samples\ARM"**.

Краткая характеристика примеров

1. Шаблон проекта загружаемого кода. На его основе можно разрабатывать собственные модули.
2. Функция main(), возвращающая определенный код (демонстрация передачи параметров dwRet и dwP1).
3. Демонстрация использования буферов ввода-вывода.
4. Демонстрация работы необнуляемого буфера (внутренняя память). В одном вызове можно записать данные, а в следующем – считать обратно.
5. Разделение загружаемого приложения на несколько логических частей (обработка команд).
6. Управление индикатором. Демонстрация работы функций GcaLedOn()/GcaLedOff().
7. Создание интерфейса к EEPROM. Обеспечивает чтение/запись участка в 512 байт EEPROM блоками по 16 байт. Блоки шифруются на AES-128 в режиме ECB, на ключе, хранящемся внутри загружаемого приложения.
8. Демонстрация работы с защищенными ячейками. Проход по аппаратным алгоритмам и получение информации о них (сервисы времени, чтения, модификации, счетчик запусков).
9. Модификация содержимого заданной защищенной ячейки.
10. Демонстрация функций работы с таймером (только для Guardant Code Time). Получение текущего времени и валидности RTC. Функция установки времени отсутствует. Единственный способ установить время в электронном ключе – при помощи GrdUtil.
11. Получение времени жизни защищенной ячейки.
12. Функции криптографии. Демонстрация генерации ключевой пары, подписи и проверки этой подписи по ECC160 с возвратом всех буферов в PC.
13. Использование аппаратного алгоритма для подписи.
14. Подсчет хэша.
15. Возврат блока случайных данных.
16. Отличие форматов хранения чисел с плавающей точкой в GCC ARM и PC.
17. Демонстрация использования встроенных математических функций.
18. Пример реализации вычислительной задачи (вычисление заданной точки на кривой Безье порядка N, параметры кривой хранятся внутри загружаемой микропрограммы).
19. Реализация целочисленной задачи. Шифрование на алгоритме Blowfish для блока данных на заданном ключе.
20. Демонстрация использования отладочной библиотеки.Для запуска примера необходимо запустить проект **Demo.sln**

Пример демонстрирует, как можно использовать отладочную DLL в проектах. В подкаталоге **LoadableCode** содержится проект загружаемого модуля, который собирается в виде самостоятельного модуля, так и в составе проекта **Demo.sln**.

По умолчанию в проекте задано использование отладочной библиотеки, при этом код из файла **lc.c** собирается в составе VS-проекта. Если убрать опцию **DEBUG_DLL** и прошить в ключ маску из подпапки **LoadableCode**, код из файла **lc.c** будет выполняться уже внутри ключа. Как видно, в **lc.c** допустимо использование вызовов Guardant Code API.

Однако в случае исполнения в составе VS-проекта налагается ограничение на функцию GcaExit(), и она не вызовет выход из кода в **lc.c**.