Guardant Code API

While developing the loadable code there is a high probability that you may face the necessity to address the dongle resources located in **EEPROM** area (protected items, algorithms) or the timer. Therefore, a special Guardant Code API was developed (See Guardant API help system, **GrdAPI.chm** file). The library of this API contains most functions of Guardant API adapted for the use from within the loadable code.

The main issue while working with **Guardant Code API** lies in the fact that the handle of protected container inside the loadable code loses its sense, since this code, first, has access only to one dongle and, secondly, there can be no situation of competitive access to dongle resources from various streams of one application and from various applications.

Also, the HANDLE type parameter is transferred to the functions of internal API of the loadable code. This is done in order to maintain consistency and ensure the convenience of debugging the loadabable code.

Guardant Code API supports the main Guardant API functions, related to storing data and working with the algorithms.

Besides, there is a capability of calling encryption algorithms from the API of loadable code not using the descriptors but directly, similar to the way software-based algorithms are called in Guardant API. For this a special reserved algorithm name is specified to the item containing the descriptor instead of a numerical name.

If Guardant API functions are present in the loadable code (for instance, an algorithm which was previously protected by Guardant dongles is transferred into a dongle), then for most of these functions there is an analog in Guardant Code API and porting will be confined to the change of prefix from GrdXXX to GcaXXX or GccaXXX.

Function	Description
GcaCrash()	
GcaExit()	Exiting the loadable code
GcaLedOn()	Turning LED on
GcaLedOff()	Turning LED off
GcaRead()	Read EEPROM data, GrdRead() analog
GcaWrite()	Write EEPROM data, GrdWrite() analog
GcaPI_Read()	Read protected item data, GrdPI_Read() analog
GcaPI_Update()	Read protected item data, GrdPI_Update() analog
GcaPI_GetTimeLimit()	Receiving the remaining algorithm operating time
GcaPI_GetCounter()	Get algorithm counter value, GrdPI_GetCounter() analog
GcaGetTime()	Get dongle’s timer state, GrdGetTime() analog. For Guardant Code Time only
GcaGetRTCQuality()	Real time clock testing
GcaGetLastError()	Receiving the last error code
GccaCryptEx()	Data encryption, GrdCryptEx() analog
GccaSign()	Generate the digital signature, GrdSign() analog
GccaVerifySign()	Verify the digital signature, GrdVerifySign() analog
GccaGenerateKeyPair()	Creating key pairs
GccaHash()	Calculate hash, GrdHash() analog
GccaGetRandom()	Random number generation
GcaSetTimeout	Setting the maximum allowed loadable code operating time
GcaCodeGetInfo	Request information from the loadable code descriptor
GcaCodeRun	Real time clock (RTC) testing

Important information

Since GSII64 algorithm and its derivatives (HASH64, RAND64, etc.) are not implemented in Guardant Code, you will probably have to rework the existing protection scheme for the use of AES128 algorithms for encryption and SHA256 for hashing. All other capabilities of the previous dongle models are present in Guardant Code.