Revision 1.02

Crypto iButton™
Firmware Reference Manual

September 2, 1997

Introduction

The Crypto iButton is a single-chip, physically secure coprocessor with integrated 1024-bit arithmetic accelerator and continuously running true time clock in a self-contained stainless steel package. In contrast to other products the Crypto iButton requires just a single data line plus ground reference for communication and power supply. Its true time clock and the internal NVSRAM are powered by an internal lithium cell.

The built-in firmware of the Crypto iButton is easy to use for a great variety of high security applications. The non-volatile memory together with the well designed firmware functions make the Crypto iButton very cost effective since several independent applications may share the same physical device. Each service provider reserves its own private memory section (Transaction Group) inside the device without the risk of overwriting other service provider's data.

Privacy is established by using PINs (Personal Identification Numbers). If desired, the device can be made inaccessible to others by setting the common PIN or be locked completely. Locking, however, does not even allow the service provider to make any more changes to the device's original configuration.

The Crypto iButton is set up by the service provider for an application by creating a transaction group that contains all data objects required to perform the handling and processing of data. This group may be locked and protected by a PIN to prevent unauthorized access. After this preparation phase the Crypto iButton is used by loading new data into input objects, invoking a script (an object stored in the transaction group containing instructions) and, after the computation is done, reading the result from output objects.

Firmware Inside The Crypto iButton

The Crypto iButton contains 32K Bytes of pre-programmed ROM containing the device's firmware. This firmware is developed and maintained solely by Dallas Semiconductor, not by the user of the Crypto iButton or service provider. The major portion of this manual is dedicated to explaining this firmware. Dealing with the firmware makes application development for the Crypto iButton more efficient and faster than writing assembly language code for a microcontroller.

The firmware of the Crypto iButton consists of four layers

a) elementary communication and power management
b) command interpreter to execute single commands
c) script interpreter to apply a series of operations and functions to data stored in the device

d) library of functions accessible to the script interpreter

The functions of layer a) are invisible to the user. What they accomplish and how they work is described in detail in**Appendix D**, Device Communications. The firmware functions that realize an operating system to execute commands sent by the bus master (layer b) are explained in the section **API Specification** (Application Program Interface). Except for the API functions that logically singulate and address a specific Crypto iButton and provide error code information to the application software, each of the functions has a direct firmware equivalent to be used if the application platform is not supported by an API. The section **Script Language** defines the elements and syntax of the script language and discusses examples that represent a variety of typical Crypto iButton applications.

Development Support

In a typical application the Crypto iButton is temporarily connected to a DS1410E adapter that interfaces it to the parallel port (LPT) of a computer. Application software running on the computer calls API functions that, in turn, call operating system functions of the Crypto iButton's firmware and invoke scripts that the service provider has implemented when preparing the Crypto iButton for the application. They also manage the power supply to the Crypto iButton. This API is currently available from Dallas Semiconductor for IBM-compatible computers running under WINDOWS 3.1x , WINDOWS 95 and WINDOWS NT. APIs for other computer types and operating systems are in preparation.

Scripts are very compact sets of instructions to be applied to data already transferred to the Crypto iButton. To simplify script development and testing, Dallas Semiconductor has developed a text based script compiler that is available for several different computer types. Which computers and operating systems are currently supported and how this compiler is invoked is explained in **Appendix C**, Script Compiler.

Software Development and Usage Model

The Crypto iButton's API is provided as Dynamic Link Library (DLL). This allows the service provider to develop application software using any high level language that is supported by a compiler that creates Windows or Windows 95 compatible code. For currently unsupported target machines the software development is more complex since one has to deal directly with the firmware functions that realize the operating system of the Crypto iButton.

After the Crypto iButton's functionality (usage model) to be implemented in the application program and the application program itself are defined, the software development goes through three phases, the preparation phase, setup phase and debug phase.

In the **preparation phase**, the software developer

- defines all data and script objects needed to perform the data processing inside the Crypto iButton
- writes and compiles the script(s) using the script compiler
- writes a setup program that allows calling functions of the Crypto iButton's operating system
- writes a test version of the application program that writes objects of the transaction group, invokes script(s), reads objects, displays data and allows interaction for debugging purposes.

In the **setup phase**, the software developer uses the setup program to

- create a transaction group for this usage model in the Crypto iButton
- set a PIN for the transaction group (recommended)
- create all data and script objects needed to perform the data processing
- set attributes of these objects

In the **debug phase**, the software developer

- uses the test version of the application program to debug both the script(s) and the functions calling on the Crypto iButton's operating system

To modify scripts or objects inside the transaction group, one uses the setup program.

After the scripts are debugged one locks the transaction group and the first device is ready for use. More devices can now be set up automatically by re-creating the same transaction group and its objects and writing the same data into the objects. All of this assumes no key generation.

Now the application program can be optimized and debugged. The use of the Crypto iButton typically consists of the same sequence of calls, which first write data to input objects of the transaction group, invoke script(s) and then read the output objects to obtain the results.

API Specification

This section of the Firmware Reference Manual describes all Application Program Interface (API) and Firmware Function Commands in a standardized way. The API is highly pointer-oriented whereas the firmware function call basically exchanges bytes with the Crypto iButton. The information to be provided or received is essentially the same.

The Firmware Function Commands are relevant if there is no API for the desired platform available. Otherwise the API should be preferred since it frees the developer from the burden of having to write software for communicating with the Crypto iButton on a hardware level.

When communicating directly with the Crypto iButton on a hardware level, the information listed in the section **Transmit** has to be written to the Intermediate Product Register (IPR), the information listed under **Receive** is to be read from the IPR. In either case the information in the IPR is accompanied by an 8-byte block header containing transfer management data. This block header is generated and written to the I/O buffer by the bus master when data is *transmitted* to the Crypto iButton. When *receiving* the result of the execution of a firmware command, the Crypto iButton generates the header and makes it available to the bus master through the I/O buffer so that the data in the IPR can be read correctly and error-checked by using the block header information.

Details on how the block header is generated and other relevant information on communicating directly with the firmware are found in Appendix D, Device Communications. For timing specifications of the electrical communications protocol and hardware command codes to access the registers and to run the microcontroller inside the device please refer to the DS1954 Crypto iButton Data Sheet.

Calling Conventions

The Crypto iButton API uses the same calling conventions as the WIN32 API functions.

FindCiBs

The FindCiBs function searches all of the peripheral ports with 1-wire bus drivers for Crypto iButtons.

API Call & Return

```
LPBYTE DLLEXPORT FindCiBs(
        LPWORD lpCiBNum             // Pointer to number of CiBs found
);
```

If the function succeeds, the return value is a pointer to the top of the buffer containing the ROM IDs of all of the Crypto iButtons found during the search. If the function fails for any reason, the return value is a NULL pointer.

FIRMWARE Call & Return

This function is realized by the **hardware** of the Crypto iButton.

Parameters And Description

Name	Description
lpCiBNum (output)	pointer to a word that contains the number of Crypto iButtons found during the search

Firmware Equivalent

Name	Length
(n/a)	(This function has no firmware equivalent)

Error Codes

Name	API	Firmware	Explanation
ERR_NO_CIBS_FOUND	F000H	(n/a)	No Crypto iButtons were found during the previous search.
ERR_ADAPTER_NOT_FOUND	F300H	(n/a)	No 1-wire adapter could be found on system.

Remarks

The buffer containing the ROM IDs is simply a contiguous list. The **GetCiBError** function may be used to retrieve error information.

SelectCiB

SelectCiB is called to specify which Crypto iButton will be addressed for following communications.

API Call & Return

```
    BOOL DLLEXPORT SelectCiB(
        LPBYTE lpRomID              // Pointer to ROM data
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE.

FIRMWARE Call & Return

This function is realized by the **hardware** of the Crypto iButton.

Parameters And Description

Name	Description
lpRomID (input)	pointer to the ROM data of a Crypto iButton.

Firmware Equivalent

Name	Length
(n/a)	(This function has no firmware equivalent)

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_CIB_ROM	F100H	(n/a)	The specified ROM was not found in the previous search.

Remarks

All other API functions use the ROM data set by SelectCiB when accessing the 1-wire bus. Therefore, SelectCiB must be called before any of the functions that communicate with the Crypto iButton firmware. If the specified ROM data was found during the last search (see **FindCiBs**), SelectCiB will return TRUE. Otherwise SelectCiB will return FALSE.

SetCommonPIN

The SetCommonPIN function changes the common PIN (personal identification number).

API Call & Return

```
BOOL DLLEXPORT SetCommonPIN(
        LPPIN lpCommonPIN,          // Pointer to current common PIN structure
        LPPIN lpNewPIN,             // Pointer to new common PIN structure
        BYTE OptionByte             // Common PIN option byte
        LPRETPACKET lpRP            // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 01H, old PIN, new PIN, PIN option byte

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description

lpCommonPIN (input) pointer to a structure that contains the current common PIN, that is used to access system level commands (such as the master erase command). The PIN supplied must match the actual common PIN exactly for SetCommonPIN to succeed

lpNewPIN (input) pointer to a structure that contains the PIN that will replace the old common PIN.

OptionByte (input) 1 byte, see below

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
old PIN	0 to 8 bytes
new PIN	1 to 8 bytes
PIN option byte	1 byte, see table below

Option Byte

Name	**Value**	**Explanation**
PIN_TO_ERASE	00000001b	The common PIN is required to execute the master erase command.
PIN_TO_CREATE	00000010b	The common PIN is required to create a transaction group.

The PIN option byte may be the bitwise-or of any of the above values.

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_COMMON_PIN	0081H	81H	The common PIN match failed.
ERR_BAD_PIN_LENGTH	0083H	83H	The supplied PIN was longer than 8 bytes.

ERR_CIB_NOT_FOUND F200H (n/a) The selected Crypto iButton can no longer be found.

Remarks

Both, the common and group PINs are up to 8 bytes in length and are purely binary values. Initially, the Crypto iButton has a PIN (Personal Identification Number) of 0 (Null) and an option byte of 0. Once a PIN has been established, it can only be changed by providing the old PIN or by a Master Erase. However, if the PIN_TO_ERASE bit is set in the option byte, the PIN can **only** be changed through the set common PIN command. If no PIN has been set the length byte in the **PIN** structure must be set to 0.

Changing and not publishing the common PIN will prevent other service providers from executing the following commands:

SetCommonPIN	always
LockCiB	always
DisableKeySetGeneration	always
CreateTransactionGroup	only if the PIN_TO_CREATE bit is set
MasterErase	only if the PIN_TO_ERASE bit is set

Therefore, when setting the common PIN it is highly recommended to set the PIN_TO_ERASE bit to 1 and leave the PIN_TO_CREATE bit at 0. This allows the creation of additional transaction groups but prevents accidental erasure of the Crypto iButton and further changes of the common PIN.

MasterErase

The MasterErase function deletes all of the transaction groups.

API Call & Return

```
BOOL DLLEXPORT MasterErase(
        LPPIN lpCommonPIN,          // Pointer to common PIN
        LPRETPACKET lpRP            // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code, call the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 02H, Common PIN

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description

lpCommonPIN (input) pointer to a structure that contains the current common PIN, that is used to access system level commands.

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Common PIN	1 to 8 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_COMMON_PIN	0081H	81H	The common PIN match failed.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the LSB (least significant bit) of the **PIN option byte** is clear (i.e. PIN not required for Master Erase) then a 0 is transmitted for the **Common PIN** value. In general this text will always assume a PIN is required. If no PIN has been established, a 0 should be transmitted as the PIN. This is true for the common PIN and group PINS (see below). If the PIN was correct the firmware deletes all groups (see below) and all objects within the groups. The common PIN and common PIN option byte are both reset to zero.

See also the remarks at SetCommonPIN.

CreateTransactionGroup

The CreateTransactionGroup function allows the service provider to create a new transaction group within the CryptoiButton provided it has not already been locked.

API Call & Return

```
BOOL DLLEXPORT CreateTransactionGroup(
        LPPIN lpCommonPIN,            // Pointer to common PIN structure
        LPNAME lpGroupName,           // Pointer to new group name structure
        LPPIN lpGroupPIN              // Pointer to PIN for new group
        BYTE GroupAttr                // Group attribute byte
        LPBYTE lpGroupID              // Pointer to group ID byte
        LPRETPACKET lpRP              // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 03H, Common PIN, Group name, Group PIN, Group Attribute byte

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 1 if successful, 0 otherwise
Output Data = Group ID if successful, 0 otherwise

Parameters And Description

Name Description

lpCommonPIN (input) pointer to a structure that contains the current common PIN.
lpGroupName (input) pointer to a structure that contains the initial name for the transaction group to be created. The name must be less than or equal to 16 bytes in length.
lpGroupPIN (input) pointer to a structure that contains the initial PIN for the transaction group to be created. The PIN must be less than or equal to 8 bytes in length.
GroupAttr (input) initial Group Attribute byte, reserved, should be set to 0.
lpGroupID (output) pointer to a byte that contains the firmware assigned ID for the newly created group
lpRP (output) pointer to a structure which receives the return packet from the CryptoiButton.

Firmware Equivalent

Name	Length
Common PIN	1 to 8 bytes
Group name	1 to 16 bytes
Group PIN	1 to 8 bytes
Group Attribute byte	1 byte

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_COMMON_PIN	0081H	81H	The common PIN match failed.
ERR_BAD_PIN_LENGTH	0083H	83H	The supplied PIN was longer than 8 bytes.
ERR_BAD_NAME_LENGTH	0085H	85H	The supplied group name was more than 16 bytes long.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough memory to create a new transaction group.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_OPEN_GROUP	0096H	96H	There is an unlocked transaction group in the Crypto iButton.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

All transaction groups must be locked before a new group can be created. There must also be at least 512 bytes of RAM available in the Crypto iButton to create a new transaction group, even if the new group will occupy less than 512 bytes. A transaction group can be created without knowing the common PIN if the PIN_TO_CREATE bit of the Option Byte is 0. See SetCommonPIN for details.

SetGroupPIN

The SetGroupPIN function changes the PIN of a specific transaction group.

API Call & Return

```
    BOOL DLLEXPORT SetGroupPIN(
        BYTE GroupID                // Desired transaction group's ID
        LPPIN lpGroupPIN,           // Pointer to current group PIN structure
        LPPIN lpNewPIN,             // Pointer to new group PIN structure
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 04H, Group ID, old GPIN, new GPIN

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the current PIN for the transaction group specified by GroupID.
lpNewPIN (input) pointer to a structure that contains the PIN that will replace the old group PIN.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
old GPIN	0 to 8 bytes
new GPIN	1 to 8 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_PIN_LENGTH	0083H	83H	The new PIN length was greater than 8 bytes.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

Both, the common and group PINs are up to 8 bytes in length and are purely binary values. If no PIN has been set, the length byte in the **PIN** structure must be set to 0. The Group PIN only restricts access to objects within the group specified by the group ID transmitted.

CreateCiBObject

The CreateCiBObject function creates new objects within an open transaction group.

API Call & Return

```
    BOOL DLLEXPORT CreateCiBObject(
        BYTE GroupID                  // ID of open transaction group
        LPPIN lpGroupPIN              // Pointer to group PIN
        LPCIBOBJ lpNewObject          // Pointer to object data structure
        LPBYTE lpObjectID             // Pointer to newly created object ID
        LPRETPACKET lpRP              // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 05H, Group ID, Group PIN, Object type, Object attributes, Object data

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 1 if successful, 0 otherwise
Output Data = object ID if successful, 0 otherwise

Parameters And Description

Name Description
GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.

lpNewObject (input) pointer to a structure containing the type, attributes and data of the object to be created. Refer to **CIBOBJ** in Appendix B for the structure definition. Valid object types and attributes are listed on the next page.

lpObjectID (output) pointer to a byte that contains the firmware assigned ID for the newly created object

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes
Object type	1 byte
Object attributes	1 byte
Object data	1 to 128 bytes

Object Type

Name	**Value**	**Explanation**
OUTPUT_OBJ	00H	
WORKING_REG_OBJ	01H	
ROM_DATA_OBJ	02H	
RANDOM_FILL_OBJ	03H	
RSA_MODULUS_OBJ	20H	RSA modulus
RSA_EXPONENT_OBJ	21H	RSA exponent
MONEY_REGISTER_OBJ	22H	Money register
COUNTER_OBJ	23H	Transaction counter
SCRIPT_OBJ	24H	Transaction script
CLOCK_OFFSET_OBJ	25H	Clock offset
SALT_OBJ	26H	Random SALT
CONFIG_DATA_OBJ	27H	Configuration object
INPUT_OBJ	28H	Input data object
DESTRUCTOR_OBJ	29H	Destructor

Object Attributes

Name	**Value**	**Explanation**
LOCKED_OBJ	00000001b	The object is read-only.
PRIVATE_OBJ	00000010b	The object is only accessible by transaction scripts.
DESTRUCTIBLE_OBJ	00000100b	The object will become inaccessible to transaction scripts once a destructor object becomes active.
CIB_CREATED_OBJ	10000000b	The object was created by a Crypto iButton.

The object attribute byte may be the bitwise-or of any of the above values.

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough memory to create a new transaction group.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The group specified by GroupID has been locked.
ERR_BAD_OBJECT_TYPE	008AH	8AH	The object type specified either does not exist, or may not be created.
ERR_BAD_SIZE	008CH	8CH	The length of the object data is not valid.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.

ERR_CIB_NOT_FOUND F200H (n/a) The selected Crypto iButton can no longer be found.

Remarks

Once a transaction group has been locked, object creation within that group is impossible. If the CreateCiBObject command is successful the Crypto iButton firmware returns the Object's ID within the group specified by the Group ID. If the PIN supplied by the host was incorrect or the group has been locked by the Lock Group command (described below) the Crypto iButton returns an error code. An object creation will also fail if the object is invalid for any reason. For example if the object being created is an RSA modulus (object type 20H) and it is greater than 1024 bits in length. Objects may also be locked, privatized and made destructible after creation by using the SetCiBObjectAttr command described below. The CIB_CREATED_OBJ bit may only be set by the firmware during the execution of one of the key set generation commands described below.

There is no command to change the size of an object once it is created. Therefore, to change the size of an object, one has to delete the transaction group the object belongs to and then newly create the transaction group with all of its objects. If the objects are created exactly in the same sequence as they were before, they will keep their object IDs and there will be no need to re-compile the scripts.

SetCiBObjectAttr

The SetCiBObjectAttr function allows the service provider to lock, privatize or make destructible a specific object. Locking an object makes it read-only. Privatizing an object makes it accessible only to transaction scripts. Making an object destructible limits the length of time that a specific object is accessible to a transaction script.

API Call & Return

```
BOOL DLLEXPORT SetCiBObjectAttr(
        BYTE GroupID                    // ID of open transaction group
        LPPIN lpGroupPIN                // Pointer to group PIN
        BYTE ObjectID                   // ID of object to lock
        BYTE Attr                       // Attributes to be set
        LPRETPACKET lpRP                // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit data

Transmit	06H, Group ID, Group PIN, Object ID	(Lock Object)
	07H, Group ID, Group PIN, Object ID	(Privatize Object)
	08H, Group ID, Group PIN, Object ID	(Make Object Destructible)

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.

ObjectID (input) 1 byte value that uniquely identifies the object within the transaction group specified by GroupID.
Attr (input) 1 byte value that specifies the new attributes for the object specified by Object ID. For valid attributes see next page.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
Object ID	1 byte

Object Attributes

Name	Value	Explanation
LOCKED_OBJ	00000001b	The object is read-only.
PRIVATE_OBJ	00000010b	The object is only accessible by transaction scripts.
DESTRUCTIBLE_OBJ	00000100b	The object will become inaccessible to transaction scripts once a destructor object becomes active.

The object attribute byte may be the bitwise-or of any of the above values.

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The group specified by GroupID has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_OBJECT_ID	008EH	8EH	The specified object does not exist.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the Group ID, Group PIN and Object ID are valid, the appropriate object attribute will be set. **Setting any object attribute bit is an irreversible operation.**

LockCiB

The LockCiB function automatically locks an open transaction group if one exists and disables group creation capability.

API Call & Return

```
BOOL DLLEXPORT LockCiB(
        BYTE GroupID                    // ID of open transaction group
        LPPIN lpCommonPIN               // Pointer to common PIN
        LPRETPACKET lpRP                // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 09H, Group ID, Common PIN

Receive CSB = 0 if successful, appropriate error code otherwise

Output length = 0
Output Data = 0

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpCommonPIN (input)	pointer to a structure that contains the common PIN for the Crypto iButton.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte, contents is 00H
Common PIN	1 to 8 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_COMMON_PIN	0081H	81H	The common PIN match failed.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the host supplied Common PIN is correct and the Crypto iButton has not previously been locked, the command will succeed. When the Crypto iButton is locked it will neither accept any new groups or objects nor allow transaction groups to be deleted. This implies that all groups are automatically locked.

See also the remarks at SetCommonPIN.

LockGroup

The LockGroup function locks a transaction group. Once a transaction group has been locked, no more objects can be created within that group.

API Call & Return

```
    BOOL DLLEXPORT LockGroup(
        BYTE GroupID                  // ID of open transaction group
        LPPIN lpGroupPIN              // Pointer to group PIN
        LPRETPACKET lpRP              // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0AH, Group ID, Group PIN

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
lpRP (output) pointer to a structure that receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The group specified by GroupID has already been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group PIN provided is correct, the Crypto iButton firmware will not allow further object creation within the specified group. Locked groups may be deleted if the Crypto iButton has not been locked. Since groups are completely self-contained entities they **may** be deleted by executing the Delete Group command (described below).

InvokeScript

The InvokeScript function executes a transaction script within a specific group in the Crypto iButton.

API Call & Return

```
    BOOL DLLEXPORT InvokeScript(
        BYTE GroupID              // ID of transaction group
        LPPIN lpGroupPIN          // Pointer to group PIN
        BYTE ObjectID             // ID of script object
        WORD RunMS                // Number of milliseconds to allow the script
                                  // to complete
        LPRETPACKET lpRP          // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0BH, Group ID, Group PIN, Object ID

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 1 if successful, 0 otherwise
Output Data = estimated completion time

Parameters And Description

Name Description
GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ObjectID (input) 1 byte value that uniquely identifies the object within the transaction group specified by GroupID. ObjectID must be a handle to a script object.
RunMS (input) 16-bit value that specifies the length of time (in milliseconds) required for the script to complete.
lpRP (output) pointer to a structure that receives the return packet from the CryptoiButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes
Object ID	1 byte

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_OBJECT_ID	008EH	8EH	The specified object does not exist.
ERR_NOT_SCRIPT_ID	0095H	95H	The specified object was not a transaction script.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

The invoke script command may take several seconds to complete. It blocks communication to any 1-wire device on the 1-wire bus. If an error code was returned in the CSB, the time estimate will be 0.

ReadCiBObject

The ReadCiBObject function reads an object's attributes, type, length, and data.

API Call & Return

```
BOOL DLLEXPORT ReadCiBObject(
        BYTE GroupID                    // ID of transaction group
        LPPIN lpGroupPIN                // Pointer to group PIN
        BYTE ObjectID                   // ID of object to read
        LPCIBOBJ lpObject               // Pointer to object data structure
        LPRETPACKET lpRP                // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0CH, Group ID, Group PIN, Object ID

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = object length if successful, 0 otherwise
Output Data = object data if successful, 0 otherwise

Parameters And Description

Name **Description**

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ObjectID (input) 1 byte value that uniquely identifies the object within the transaction group specified by GroupID.
lpObject (output) pointer to the object structure that will receive the object's data.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes
Object ID	1 byte

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_OBJECT_ID	008EH	8EH	The specified object did not exist within the group.
ERR_OBJECT_PRIVATE	0091H	91H	The object is private and may not be read.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

Only open or locked objects may be read. If the Group ID, Group PIN and Object ID were correct, the Crypto iButton checks the attribute byte of the specified object. If the object has not been privatized, the Crypto iButton will transmit the object data.

WriteCiBObject

The WriteCiBObject function writes new data into an open object.

API Call & Return

```
BOOL DLLEXPORT WriteCiBObject(
        BYTE GroupID                  // ID of transaction group
        LPPIN lpGroupPIN              // Pointer to group PIN
        BYTE ObjectID                 // ID of object to write
        LPCIBOBJ lpObject             // Pointer to object data structure
        LPRETPACKET lpRP              // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0DH, Group ID, Group PIN, Object ID, Object Size, Object Data

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.

ObjectID (input) 1 byte value that uniquely identifies the object within the transaction group specified by GroupID.

lpObject (input) pointer to the object structure that contains the data to write to the object.

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID,	1 byte
Group PIN	1 to 8 bytes
Object ID	1 byte
Object Size	1 byte
Object Data	1 to 128 bytes

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_SIZE	008CH	8CH	The object data length specified was illegal.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_OBJECT_ID	008EH	8EH	The specified object did not exist within the group.
ERR_OBJECT_LOCKED	0090H	90H	The object is locked and is read-only.
ERR_OBJECT_PRIVATE	0091H	91H	The object is private and is read-only.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

Only open objects may be written. If the Group ID, Group PIN and Object ID are correct, the Crypto iButton checks the attribute byte of the specified object. If the object has not been locked or privatized, the Crypto iButton will clear the objects previous size and data and replace it with the new object data. **Note that the object type and attribute byte are not affected**.

ReadGroupName

The ReadGroupName function reads a transaction group's name by specifying it's ID.

API Call & Return

```
    BOOL DLLEXPORT ReadGroupName(
        BYTE GroupID                // ID of open transaction group
        LPNAME lpGroupName          // Pointer to transaction group name
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0EH, Group ID

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = length of group name, 0 otherwise

Output Data = group name, 0 otherwise

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupName (output) pointer to a buffer that contains the name of the transaction group specified by GroupID. Refer to **RETPACKET** in Appendix B for the structure definition to obtain the length of the group name. A group name may be up to 16 bytes long.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

All byte values are legal in a group name. Transaction group IDs are numbered sequentially starting from 1. Using the ReadGroupName function one can determine the transaction group of interest without first knowing the group ID.

DeleteGroup

The DeleteGroup function deletes a locked transaction group.

API Call & Return

```
    BOOL DLLEXPORT DeleteGroup(
        BYTE GroupID                // ID of open transaction group
        LPPIN lpGroupPIN            // Pointer to group PIN
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 0FH, Group ID, Group PIN

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group PIN and group ID are correct the Crypto iButton will delete the specified group. Deleting a group causes the automatic destruction of all objects within the group. **If the Crypto iButton has been locked the Delete Group command will fail.**

If the Crypto iButton has been locked, the MasterErase function must be called to remove the group. Note however, that a successful call to the MasterErase function deletes all of the transaction groups within the Crypto iButton.

GetGroupID

If one knows the name of the transaction group of interest, the GetGroupID function allows to retrieve the group's ID.

API Call & Return

```
    BOOL DLLEXPORT GetGroupID(
        BYTE GroupID                // ID of open transaction group
        LPNAME lpGroupName          // Pointer to group name structure
        LPBYTE lpGroupID            // Pointer to group ID byte
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 10H, Group name

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 1 if successful, 0 otherwise
Output Data = Group ID if successful, 0 otherwise

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupName (input)	pointer to a structure containing the name of the desired transaction group.

lpGroupID (output) pointer to a byte that contains the group ID that belongs to the name pointed to by lpGroupName.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group name	1 to 16 bytes

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_NAME_LENGTH	0085H	85H	The name length specified was greater than 16 bytes.
ERR_GROUP_NOT_FOUND	0098H	98H	A matching group name was not found.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

This function provides a quick method for determining if the desired transaction group exists within a Crypto iButton. No PIN is required.

GetCiBConfiguration

The GetCiBConfiguration function is called to retrieve important Crypto iButton configuration information

API Call & Return

```
    BOOL DLLEXPORT GetCiBConfiguration(
        LPCIBINFO lpConfig          // Pointer to configuration data
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 11H

Receive CSB = 0
Output length = 2
Output Data = Crypto iButton configuration structure

Parameters And Description

Name Description

lpConfig (output) pointer to a structure that contains the Crypto iButton's configuration information. Refer to **CIBINFO** in Appendix B for the structure definition.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
(n/a)	(the function call requires no parameters)

Configuration Structure

Name	Sequence	Explanation
GroupNum	byte 1	number of transaction groups currently within the Crypto iButton.
CiBFlags	byte 2	Flag byte (see below)

Flag Byte

Name	Value	Explanation
CIB_LOCKED	00000001b	The Crypto iButton has been locked.
PIN_TO_CREATE	00000010b	The Crypto iButton requires the common PIN to allow transaction group creation.

The flag byte is the bitwise-or of any of the above values

Error Codes

Name	API	Firmware	Explanation
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

This function provides a quick method for determining the number of transaction groups within the Crypto iButton.

ReadRealTimeClock

The ReadRealTimeClock function reads the contents of the Real Time Clock in the Crypto iButton.

API Call & Return

```
BOOL DLLEXPORT ReadRealTimeClock(
        LPDWORD lpRTCSeconds      // 4 most significant bytes of the RTC
        LPRETPACKET lpRP          // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 15H

Receive CSB = 0
Output length = 4
Output Data = 4 most significant bytes of the RTC

Parameters And Description

Name	Description
lpRTCSeconds (output)	pointer to a 4 byte unsigned number that receives the 4 most significant bytes of the RTC.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
(n/a)	(the function call requires no parameters)

Error Codes

Name	API	Firmware	Explanation

ERR_CIB_NOT_FOUND F200H (n/a) The selected Crypto iButton can no longer be found.

Remarks

This command is normally used by a service provider to compute a clock offset during transaction group creation. The value returned is the total number of seconds that have elapsed since the battery was attached at the factory. Only the 4 most significant bytes of the RTC are read by this command. The sub-second bytes are not returned. The value is not adjusted with a clock offset.

ReadTrueTimeClock

The ReadTrueTimeClock function reads the value of the Real Time Clock added to a clock offset (specified by ObjectID).

API Call & Return

```
    BOOL DLLEXPORT ReadTrueTimeClock(
        BYTE GroupID                // ID of transaction group
        LPPIN lpGroupPIN            // Pointer to group PIN
        BYTE ObjectID               // ID of clock offset object
        LPDWORD lpSeconds           // RTC bytes + offset
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 16H, Group ID, Group PIN, ID of offset object

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 4 if successful, 0 otherwise
Output Data = Real time clock + clock offset ID

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.

ObjectID (input) 1 byte value that uniquely identifies the object within the transaction group specified by GroupID. ObjectID must be a handle to a clock offset object.

lpSeconds (output) pointer to a 4 byte unsigned number that receives the 4 most significant bytes of the RTC added to the 4 bytes of the clock offset. The addition is performed modulo 2^{32}.

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
ID of offset object	1 byte

Error Codes

Name API Firmware Explanation

ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_OBJECT_ID	008EH	8EH	The specified object does not exist.
ERR_BAD_OBJECT_TYPE	008AH	8AH	The specified Object ID is not a clock offset.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

This command succeeds if the group ID and group PIN are valid, and the object ID is the ID of a clock offset. The clock offset object's value is computed (by the service provider) as the difference between the 4 most significant byte of the RTC, and some meaningful time (such as the number of seconds since 12:00 AM January 1, 1970). The Crypto iButton adds the clock offset to the current value of the 4 most significant bytes of the RTC and returns that value in the output data field.

CheckGroupCRC

The CheckGroupCRC function verifies the integrity of a transaction group.

API Call & Return

```
    BOOL DLLEXPORT CheckGroupCRC(
        BYTE GroupID                // ID of transaction group
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1DH, Group ID

Receive CSB = 0 if CRC was good, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name	**Description**
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_BAD_GROUP_CRC	0097H	97H	The saved group CRC did not match the CRC just computed by firmware.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

The Crypto iButton firmware maintains a CRC16 value for each transaction group. The integrity of each group may be checked at any time.

ReadRandomBytes

The ReadRandomBytes function gives convenient access to a source of high quality random numbers.

API Call & Return

```
    BOOL DLLEXPORT ReadRandomBytes(
        BYTE nBytes                 // Desired number of random bytes
        LPBYTE lpRandomBuff         // Pointer to buffer for random bytes
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 17H, Length (L)

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = L if successful, 0 otherwise
Output Data = L bytes of random data if successful

Parameters And Description

Name	Description
nBytes (input)	number of random bytes requested
lpRandomBuff (output)	pointer to the buffer that will receive the random bytes from the Crypto iButton.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Length (L)	1 byte unsigned binary number in the range of 1 to 128

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_SIZE	008CH	8CH	The number of bytes requested was too large.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

ReadRandomBytes can return as many as 128 bytes of random data. This command provides a good source of cryptograhpically useful random numbers.

ReadFirmwareVersionID

The ReadFirmwareVersionID function returns the firmware version ID string.

API Call & Return

```
    BOOL DLLEXPORT ReadFirmwareVersionID(
        LPNAME lpFirmwareID         // Pointer to firmware ID string
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 18H

Receive CSB = 0
Output length = Length of firmware version ID string
Output Data = Firmware version ID string

Parameters And Description

Name Description
lpFirmwareID (output) pointer to a structure that receives the firmware version ID string.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name Length
(n/a) (the function call requires no parameters)

Error Codes

Name	API	Firmware	Explanation
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If a good communication link exists between the host and the Crypto iButton, this function should never fail. This command returns the firmware version ID as a Pascal type string (length + data).

ReadFreeRAM

The ReadFreeRAM function returns the amount of RAM still available in the Crypto iButton for transaction groups.

API Call & Return

```
    BOOL DLLEXPORT ReadFreeRAM(
        LPWORD lpFreeRam            // Pointer to free RAM word
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 19H

Receive CSB = 0
Output length = 2
Output Data = 2 byte value containing the amount of free RAM

Parameters And Description

Name Description
lpFreeRAM (output) pointer to an unsigned short integer that will receive the number of free bytes of RAM.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
(n/a)	(the function call requires no parameters)

Error Codes

Name	**API**	**Firmware**	**Explanation**
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the Crypto iButton is locked this function will return 0 bytes free.

ChangeGroupName

The ChangeGroupName function changes the name of the transaction group (or the name of the Crypto iButton) provided one knows the group PIN.

API Call & Return

```
    BOOL DLLEXPORT ChangeGroupName(
        BYTE GroupID                // ID of transaction group
        LPPIN lpGroupPIN            // Pointer to group PIN
        LPNAME lpGroupName          // Pointer to new group name
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1AH, Group ID, Group PIN, New Group name

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name Description
GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
lpGroupName (input) pointer a structure that contains the new name for the transaction group.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes
New Group name	1 to 16 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_BAD_NAME_LENGTH	0085H	85H	The length of the new name was greater than 16 bytes.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton and the PIN supplied is correct, the transaction group name is replaced by the new group name supplied by the host. To change the name of the Crypto iButton, set GroupID to 0 and set lpGroupPIN to the common PIN. This will replace the Crypto iButton's name by the new name supplied by the host.

DisableKeySetGeneration

The DisableKeySetGeneration function is used to free RAM normally reserved for generating RSA key sets.

API Call & Return

```
    BOOL DLLEXPORT DisableKeySetGeneration(
        LPPIN lpCommonPIN           // Pointer to the common PIN
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1BH, Group ID, Common PIN

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 0
Output Data = 0

Parameters And Description

Name	Description
lpCommonPIN (input)	pointer to a structure that contains the Crypto iButton's common PIN.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte, value = 0
Common PIN	1 to 8 bytes

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_COMMON_PIN	0081H	81H	The common PIN match failed.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_NO_KEY_GENERATION	0099H	99H	Key set generation has already been disabled.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

This command enables the service provider to free memory normally required by key set generation commands for use by transaction groups. Disabling key set generation is an irreversible operation. If the common PIN transmitted by the host is valid further RSA key set generation will be impossible. Note that locking the CryptoiButton automatically disables key set generation.

See also the remarks at SetCommonPIN.

GenerateRSAKeySet

The GenerateRSAKeySet function instructs the CryptoiButton to generate a new RSA key set on behalf of a specific transaction group.

API Call & Return

```
        BOOL DLLEXPORT GenerateRSAKeySet(
                BYTE GroupID                    // ID of transaction group
                LPPIN lpGroupPIN                // Pointer to group PIN
                BYTE ModulusSize                // Number of bytes in modulus
                LPBYTE lpModulusID              // Pointer to modulus ID
                LPBYTE lpPublicExpID            // Pointer to public exponent ID
                LPBYTE lpPrivateExpID           // Pointer to private exponent ID
                LPRETPACKET lpRP                // Pointer to return packet
        );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1CH, Group ID, Group PIN, Modulus size in bytes

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 3 if successful, 0 otherwise
Output Data = Modulus ID, public exponent ID, private exponent ID

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the CryptoiButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ModulusSize (input) number of bytes in the modulus to be generated
lpModulusID (output) pointer to a byte that contains the object ID assigned to the newly created modulus
lpPublicExpID (output) pointer to a byte that contains the object ID assigned to the newly created public exponent.
lpPrivateExpID (output) pointer to a byte that contains the object ID assigned to the newly created private exponent.
lpRP (output) pointer to a structure which receives the return packet from the CryptoiButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
Modulus size in bytes	1 byte unsigned binary number in the range of 4 to 128

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough free RAM to store all of the new objects.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The specified transaction group has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_NO_KEY_GENERATION	0099H	99H	Key generation has been disabled.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton , the PIN supplied is correct and key generation capability is enabled, the firmware will generate an entire RSA key set. The modulus and one of the exponents will immediately be locked by the firmware. The other exponent will be privatized. If successful this command will return the object ID's of the modulus, public exponent and private exponent respectively. All objects created by Crypto iButton key generation commands have the CIB_CREATE bit set in the attribute byte to make them distinguishable from objects created by the service provider.

All of the key set generation commands that create a modulus object immediately destroy the prime factors P and Q used to generate the modulus N (where N = P * Q). However Φ(N) = (P - 1) * (Q - 1) is saved until the transaction group is locked. This gives the service provider the ability to generate additional RSA exponent pairs using the same modulus. **Even though the Crypto iButton remembers Φ for each modulus created on behalf of an open group, Φ may never be read.**

GenerateRSAModAndExp

The GenerateRSAModAndExp gives the service provider the ability to specify his own public exponent and have the Crypto iButton generate a modulus and private exponent.

API Call & Return

```
    BOOL DLLEXPORT GenerateRSAModAndExp(
        BYTE GroupID                  // ID of transaction group
        LPPIN lpGroupPIN              // Pointer to group PIN
        BYTE ModulusSize              // Number of bytes in modulus
        BYTE ExponentID               // ID of public exponent
        LPBYTE lpModulusID            // Pointer to modulus ID
        LPBYTE lpPrivateExpID         // Pointer to private exponent ID
        LPRETPACKET lpRP              // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1FH Group ID, Group PIN, Modulus size in bytes, Exponent ID

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 2 if successful, 0 otherwise

Output Data = Modulus ID, Private Exponent ID

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input)	pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ModulusSize (input)	number of bytes in the modulus to be generated
ExponentID (input)	1 byte value that uniquely identifies an RSA public exponent created by the service provider
lpModulusID (output)	pointer to a byte that contains the object ID assigned to the newly created modulus
lpPrivateExpID (output)	pointer to a byte that contains the object ID assigned to the newly created private exponent.
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
Modulus size in bytes	1 byte unsigned binary number in the range of 4 to 128
Exponent ID	1 byte

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough free RAM to store all of the new objects.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The specified transaction group has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_NO_KEY_GENERATION	0099H	99H	Key generation has been disabled.
ERR_BAD_MODULUS_ID	009AH	9AH	The specified modulus does not exist.
ERR_BAD_EXPONENT_ID	009BH	9BH	The specified exponent does not exist.
ERR_NOT_CIB_CREATED	009CH	9CH	The modulus specified was not created by a Crypto iButton.
ERR_EXP_NOT_REL_PRIME	009DH	9DH	The specified public exponent was not relatively prime to the Φ of the modulus generated by the Crypto iButton.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton , the PIN supplied is correct and key generation capability is enabled, the firmware will generate a new RSA modulus N and a new exponent D such that $E * D \text{ Mod } \Phi(N) = 1$. E is the RSA exponent whose ID was passed in the transmit data packet and $\Phi(N) = \Phi(P * Q) = (P - 1) * (Q - 1)$. The modulus object N will be locked and the exponent D will be privatized by the firmware. This allows the service provider to choose a public exponent E without ever knowing the private exponent D. The prime factors P and Q used to generate the modulus N are destroyed and Φ is saved until the transaction group is locked

The firmware first generates the modulus N ($N = P * Q$). It then calculates $\Phi(N) = (P - 1) * (Q - 1)$. If the public exponent is not relatively prime to $\Phi(N)$, the firmware destroys P, Q, N and Φ. This causes the command interpreter to return the error code ERR_EXP_NOT_REL_PRIME. However, the command may be retried since a new $\Phi(N)$ will be generated.

GenerateRSAKeySetNP

The GenerateRSAKeySetNP function instructs the Crypto iButton to generate a new RSA key set on behalf of a specific transaction group. Unlike the GenerateRSAKeySet command, this command does not privatize one of the exponents automatically. Once the key set components have been read one of the exponents must be privatized before using the transaction group.

API Call & Return

```
    BOOL DLLEXPORT GenerateRSAKeySetNP(
        BYTE GroupID                // ID of transaction group
        LPPIN lpGroupPIN            // Pointer to group PIN
        BYTE ModulusSize            // Number of bytes in modulus
        LPBYTE lpModulusID          // Pointer to modulus ID
        LPBYTE lpExp1               // Pointer to 1st exponent ID
        LPBYTE lpExp2               // Pointer to 2nd exponent ID
        LPRETPACKET lpRP            // Pointer to return packet
    );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1CH, Group ID, Group PIN, Modulus size in bytes

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 3 if successful, 0 otherwise
Output Data = Modulus ID, public exponent ID, private exponent ID

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ModulusSize (input) number of bytes in the modulus to be generated
lpModulusID (output) pointer to a byte that contains the object ID assigned to the newly created modulus
lpExp1 (output) pointer to a byte that contains the object ID assigned to the newly created exponent.
lpExp2 (output) pointer to a byte that contains the object ID assigned to the newly created exponent.
lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
Modulus size in bytes	1 byte unsigned binary number in the range of 4 to 128

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough free RAM to store all of the new objects.

ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The specified transaction group has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_NO_KEY_GENERATION	0099H	99H	Key generation has been disabled.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton , the PIN supplied is correct and key generation capability is enabled, the firmware will generate an entire RSA key set. The modulus and both of the exponents will immediately be locked by the firmware. NO exponents will be privatized. If successful this command will return the object ID's of the modulus, and both exponents. None of these objects will have the CIB_CREATE bit set in the attribute byte.

GeneratePrime

The GeneratePrime function instructs the Crypto iButton to generate a prime number from 1 to 128 bytes in length.

API Call & Return

```
        BOOL DLLEXPORT GeneratePrime(
                BYTE GroupID                // ID of transaction group
                LPPIN lpGroupPIN            // Pointer to group PIN
                BYTE PrimeSize              // Number of bytes in prime number
                LPBIGNUM lpPrime            // Pointer to prime data
                LPRETPACKET lpRP            // Pointer to return packet
        );
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1DH, Group ID, Group PIN, Modulus size in bytes

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = Length of prime number in bytes
Output Data = Prime number data LSB first

Parameters And Description

Name Description

GroupID (input) 1 byte value that uniquely identifies the transaction group within the Crypto iButton.

lpGroupPIN (input) pointer to a structure that contains the PIN for the transaction group specified by GroupID.

PrimeSize (input) number of bytes in the prime to be generated

lpPrime (output) pointer to a structure which receives the prime number

lpRP (output) pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	**Length**
Group ID	1 byte
Group PIN	1 to 8 bytes
Prime size in bytes	1 byte unsigned binary number in the range of 1 to 128

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough free RAM to store all of the new objects.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The specified transaction group has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_NO_KEY_GENERATION	0099H	99H	Key generation has been disabled.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton , the PIN supplied is correct and key generation capability is enabled, the firmware will generate the prime number.

GenerateRandomExponent

The GenerateRandomExponent function instructs the Crypto iButton to generate an automatically privatized random exponent.

API Call & Return

```
BOOL DLLEXPORT GenerateRandomExponent(
        BYTE GroupID                // ID of transaction group
        LPPIN lpGroupPIN            // Pointer to group PIN
        BYTE ExpSize                // Number of bytes in exponent
        LPBYTE lpExpID              // Pointer to object ID of exponent
        LPRETPACKET lpRP            // Pointer to return packet
);
```

If the function succeeds, the return value is TRUE. If the function fails for any reason, the return value is FALSE. To retrieve the error code use the **GetCiBError** function.

FIRMWARE Call & Return

Transmit 1EH, Group ID, Group PIN, Exponent size in bytes

Receive CSB = 0 if successful, appropriate error code otherwise
Output length = 1 if successful, 0 otherwise
Output Data = Object ID of newly created exponent object

Parameters And Description

Name	Description
GroupID (input)	1 byte value that uniquely identifies the transaction group within the Crypto iButton.
lpGroupPIN (input)	pointer to a structure that contains the PIN for the transaction group specified by GroupID.
ExpSize (input)	number of bytes in the exponent to be generated
lpExpID (output)	pointer to a byte which receives the exponent object ID
lpRP (output)	pointer to a structure which receives the return packet from the Crypto iButton.

Firmware Equivalent

Name	Length
Group ID	1 byte
Group PIN	1 to 8 bytes
Exponent size in bytes	1 byte unsigned binary number in the range of 1 to 128

Error Codes

Name	API	Firmware	Explanation
ERR_BAD_GROUP_PIN	0082H	82H	The group PIN match failed.
ERR_INSUFFICIENT_RAM	0086H	86H	There was not enough free RAM to store all of the new objects.
ERR_CIB_LOCKED	0087H	87H	The Crypto iButton has been locked.
ERR_GROUP_LOCKED	0089H	89H	The specified transaction group has been locked.
ERR_BAD_GROUP_ID	008DH	8DH	The specified transaction group does not exist.
ERR_NO_KEY_GENERATION	0099H	99H	Key generation has been disabled.
ERR_CIB_NOT_FOUND	F200H	(n/a)	The selected Crypto iButton can no longer be found.

Remarks

If the group ID specified exists in the Crypto iButton , the PIN supplied is correct and key generation capability is enabled, the firmware will randomly generate a new private exponent.

GetCiBError

The GetCiBError function returns the last error that occurred while communicating with the Crypto iButton.

API Call & Return

WORD DLLEXPORT GetCiBError(VOID);

This function never fails.

FIRMWARE Call & Return

This is an API function only. The firmware returns error codes in the Command Status Byte (CSB).

Parameters And Description

Name	Description
(n/a)	(this function requires no parameters)

Firmware Equivalent

Name	Length
(n/a)	(This function has no firmware equivalent)

Error Codes

Name	API	Firmware	Explanation
(n/a)	(n/a)	(n/a)	(This function always returns valid data.)

Remarks

The low byte of the return value is used for command interpreter and script interpreter errors. The high byte is used for low level communication errors and data formatting errors. A listing of possible error codes is provided in Appendix A.

Script Language

The firmware functions described in the previous section of this manual provide the handles to creating objects, setting attributes and PINs and many other essential operations. The most important of these firmware functions is the one that activates the script interpreter, the highest layer of the Crypto iButton's firmware.

As a computer makes use of registers, data memory, I/O channels, peripherals and program memory, the script interpreter does the same with the objects of a transaction group. Currently, there are 14 different object types, each for a specific purpose (see CreateCiBObject description). The object that equivalents the program memory of a common computer is called script. Such scripts store very compact program code that is step by step interpreted and executed by the script interpreter whenever the InvokeScript command is called.

The simple script language supported by the Crypto iButton script interpreter is described in detail in the document entitled Cryptographic iButton Script Language.

Appendix A: Error Code Definitions

Error Name	Error Code	Error Source	Description
ERR_BAD_COMMON_PIN	81H	Command Interpreter	This error code will be returned when a command requires a common PIN and the PIN supplied does not match the Crypto iButton's common PIN. Initially the common PIN is set to 0.
ERR_BAD_GROUP_PIN	82H	Command Interpreter	Transaction groups may have their own PIN. If this PIN has been set (by a set group PIN command) it must be supplied to access any of the objects within the group. If the Group PIN supplied does not match the actual group PIN, the Crypto iButton will return this error code.
ERR_BAD_PIN_LENGTH	83H	Command Interpreter	There are 2 commands that can change PIN values. The set group PIN and the set common PIN commands. Both of these require the new PIN as well as the old PIN. This error code will be returned if the old PIN supplied was correct, but the new PIN was greater than 8 characters in length.
ERR_BAD_NAME_LENGTH	85H	Command Interpreter	A transaction group name may not exceed 16 characters in length. If the name supplied is longer than 16 characters, this error code is returned.
ERR_INSUFFICIENT_RAM	86H	Command Interpreter	The create transaction group and create object commands return this error code when there is not enough heap available in the Crypto iButton.
ERR_CIB_LOCKED	87H	Command Interpreter	When the Crypto iButton has been locked, no groups or objects can be created or destroyed. Any attempts to create or delete objects will generate this error code.
ERR_CIB_NOT_LOCKED	88H	Command Interpreter	If the Crypto iButton has not been locked.
ERR_GROUP_LOCKED	89H	Command Interpreter	Once a transaction group has been locked object creation within that group is not possible. Also the objects' attributes and types are frozen. Any attempt to create objects or modify their attribute or type bytes will generate this error code.
ERR_BAD_OBJECT_TYPE	8AH	Command Interpreter	When the host sends a create object command to the Crypto iButton, one of the parameters it supplies is an object type (see command section). If the object type is not recognized by the firmware it will return this error code.
ERR_BAD_OBJECT_ATTR	8BH	Command Interpreter	When the host sends a create object command to the Crypto iButton, one of the parameters it supplies is an object attribute byte (see command section). If the object attribute byte is not recognized by the firmware this error code will be returned.
ERR_BAD_SIZE	8CH	Command Interpreter	This error code is normally generated when creating or writing an object. It will only occur when the object data supplied by the host has an invalid length.
ERR_BAD_GROUP_ID	8DH	Command Interpreter	All commands that operate at the transaction group level require the group ID to be supplied in the command packet. If the group ID specified does not exist in the Crypto iButton it will generate this error code.
ERR_BAD_OBJECT_ID	8EH	Command Interpreter	All commands that operate at the object level require the object ID to be supplied in the command packet. If the object ID specified does not exist within the specific transaction group (also specified in the command packet) the Crypto iButton will generate this error code.
ERR_OBJECT_LOCKED	90H	Command Interpreter	Locked objects are read only. If a write object command is attempted and it specifies the object ID of a locked object the Crypto iButton will return this error code.

ERR_OBJECT_PRIVATE 91H Command Interpreter Private objects are not directly readable and may not be modified by the write object command. If a read object command or a write object command is attempted, and it specifies the object ID of a private object, the CryptoiButton will return this error code.

ERR_MAX_GROUPS 92H Command Interpreter Only 32 (= MAX_GROUPS) transaction groups may be created. If a service provider attempts to create more transaction groups than MAX_GROUPS, the firmware will return this error code.

ERR_MAX_OBJECTS 93H Command Interpreter Each transaction group may have as many as 127 (= MAX_OBJECTS) objects. Any attempt by a service provider to create more will result in this error code being returned.

ERR_NOT_SCRIPT_ID 94H Command Interpreter If the object ID passed to the script interpreter for the invoke script command is not the ID of a script object, this error code will be returned.

ERR_OPEN_GROUP 95H Command Interpreter If a service provider attempts to create a new transaction group while an existing group is unlocked, the command interpreter will return this error code.

ERR_BAD_GROUP_CRC 96H Command Interpreter This error code is only returned by the check group crc command if the crc check fails.

ERR_BAD_PACKET_LEN 97H Command Interpreter If the ReadRandomBytes command is executed and requests more than 128 bytes this error code is returned.

ERR_GROUP_NOT_FOUND 98H Command Interpreter This error code is generated by the get group id command if the name supplied does not match the name of any of the transaction groups in the CryptoiButton.

ERR_NO_KEY_GENERATION 99H Command Interpreter If any of the key set generation commands are called after the CryptoiButton has been locked or the disable key set generation command has been called, the command interpreter will return this error code.

ERR_BAD_MODULUS_SIZE 9AH Command Interpreter The generate RSA key set command requires a requested modulus size. If the modulus size specified is illegal, the command interpreter will return this error code.

ERR_KEY_GEN_DISABLED 9BH Command Interpreter This error code is returned when a key set generation command is executed after key set generation has been disabled.

ERR_NO_CIBS_FOUND F000H Access System DLL This error occurs when the **FindCiBs** function is unable to find any CryptoiButtons during its search.

ERR_BAD_CIB_ROM F100H Access System DLL This error occurs when the ROM data specified in a call to **SelectCiB** was not found in the last search performed by **FindCiBs**.

ERR_CIB_NOT_FOUND F200H Access System DLL The currently selected CryptoiButton can no longer be found.

ERR_ADAPTER_NOT_FOUND F300H Access System DLL During the last search by FindCiBs, no 1-wire adapters were found.

Appendix B: Defines And Structures

DEFINES

```
#define MAX_PIN_LEN        8   // Maximum PIN length
#define MAX_NAME_LEN       16  // Maximum group name length
#define MAX_PACKET_LEN     128     // Maximum data packet length
#define MAX_OBJ_LEN        128     // Maximum length of object data
```

STRUCTURES

1) RETPACKET

The RETPACKET structure defines the information returned by the Crypto iButton's command interpreter.

```
typedef struct _RETPACKET
{
	BYTE CSB;
	BYTE GroupID;
	BYTE DataLen;
	BYTE CmdData[MAX_PACKET_LEN];
}
RETPACKET, *PRETPACKET, NEAR *NPRETPACKET, FAR *LPRETPACKET;
```

Members And Description
Name Description
CSB CSB (command status byte) is set to 0 upon successful completion of any command. If a command fails CSB is set to the appropriate error code (see appendix A).
GroupID The group ID for which the command was executed
DataLen DataLen specifies the number of bytes returned in the CmdData array.
CmdData CmdData is an array of bytes that contains all of the data returned by the command interpreter. All of the API functions return this same data in a command specific structure.

2) PIN

PIN defines the structure of the Crypto iButton's common and group PINS.

```
typedef struct _PIN
{
	BYTE Len;
	BYTE PINData[MAX_PIN_LEN];
}
PIN, *PPIN, NEAR *NPPIN, FAR *LPPIN;
```

Members And Description
Name Description
Len Len specifies the PIN length in bytes.
PINData PINData is an array of bytes that specifies a group or common PIN.

3) NAME

NAME defines the structure of transaction group names.

```
typedef struct _NAME
{
	BYTE Len;
	BYTE NameData[MAX_NAME_LEN];
}
NAME, *PNAME, NEAR *NPNAME, FAR *LPNAME;
```

Members And Description
Name Description
Len Len specifies the length of a group name in bytes.
NameData NameData is an array of bytes that specifies a transaction group name

4) CIBOBJ

CIBOBJ defines the generic structure of any Crypto iButton object.

```
typedef struct _CIBOBJ
{
        BYTE Attr;
        BYTE Type;
        BYTE Len;
        BYTE ObjData[MAX_OBJ_LEN]
}
CIBOBJ, *PCIBOBJ, NEAR *NPCIBOBJ, FAR *LPCIBOBJ;
```

Members And Description

Name Description

Attr Attr specifies the attributes of an object. For details on the attributes, please refer to **CreateCiBObject** in the main section of this document.

Type Type is the object type specification byte. For details on types, please refer to **CreateCiBObject** in the main section of this document.

Len Len specifies the length of the object data in bytes.

ObjData ObjData is an array of bytes that contain the actual object data.

5) CIBINFO

CIBINFO defines the structure of the data returned by a call to the **GetCiBConfiguration** command.

```
typedef struct _CIBINFO
{
        BYTE GroupNum;
        BYTE CiBFlags;
}
CIBINFO, *PCIBINFO, NEAR *NPCIBINFO, FAR *LPCIBINFO;
```

Members And Description

Name Description

GroupNum GroupNum specifies the number of transaction groups currently within the Crypto iButton.

CiBFlags CiBFlags is a flag byte. For details on flags, please refer to **GetCiBConfiguration** in the main section of this document.

6) BIGNUM

BIGNUM defines the structure of the data returned by a call to the **GeneratePrime** command.

```
typedef struct _BIGNUM
{
        BYTE Len;
        BYTE NumArr[MAX_PACKET_LEN];
}
BIGNUM, *PBIGNUM, NEAR *NPBIGNUM, FAR *LPBIGNUM;
```

Members And Description

Name	Description
Len	Length of large integer in bytes
NumArr	Byte array of binary representation of large integer

Appendix C: Device Communications

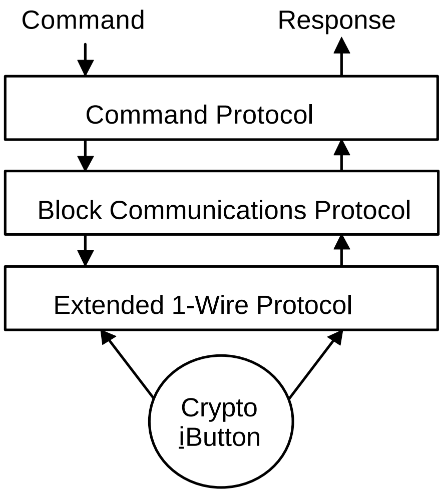
Introduction

The Crypto iButton operates in an environment where communication and power supply share the same conducting path, and where the available amount of power is limited. To make operation under these conditions possible, the device separates communication from execution, performing each at different times. Every firmware function starts with the bus master (host) communicating with the I/O buffer and Intermediate Product Register (IPR) to set up an operation, then issuing a RUN command and then providing power on the line for some fixed amount of time while the command is carried out.

Some commands may be processed quickly while others may take several seconds to complete. An internal timer controlled by the OWUS register causes an alarm so that the device may terminate work-in-progress and be prepared for the loss of power as the bus master comes back on line to check status. The bus master and the device must agree on the run time period prior to beginning an execution cycle. If the bus master removes power and attempts to communicate while the microcontroller in running, a power failure will occur and the work in progress will be interrupted.

The communication protocol of the Crypto iButton provides several types of commands and signaling for managing this interaction. These include commands to read the status of the Crypto iButton and to send status information back to the device, and also two different RUN command, one called **Start Program**, the other called **Continue Program**.

When the bus master wishes to execute a firmware function, it must adhere to appropriate protocols at various levels, as shown below.



The **Command Protocol** defines the type of operation to be performed by the device and with this the result of the operation. What information is to be transmitted on byte level and the expected format of the result are explained for each operation in the section **API Specification** under the headline **FIRMWARE Call & Return**.

The **Block Communications Protocol** delivers the Command Messages or Responses reliably and handles fragmentation of the message or response when necessary. This protocol includes the interaction with the device that is necessary to execute the operation to completion. The Block Communications Protocol *logically* writes to and reads from the I/O buffer and Intermediate Product Register (IPR). The I/O buffer is used to receive/transmit the header information that applies to and safeguards the command and result data that is exchanged through the IPR. Details on this header are discussed later in **this section**.

The **Extended 1-Wire Protocol** is the standard Dallas 1-Wire Multidrop Serial Communications Protocol with extensions to support the power transfer. This protocol directly interacts with the hardware of the Crypto iButton. It synchronizes bus master and Crypto iButton on the Crypto iButton's hardware command level and *physically* communicates with the I/O buffer and IPR on bit and byte level. This protocol is described in the DS1954 **Crypto iButton Data Sheet**.

Execution Of A Firmware Function Command

General Firmware Function Command Flow Chart Figure 1

Execute a Firmware Function Command (simplified)
Access device (Reset/Presence Sequence, Match ROM)
Write data block header to I/O Buffer (Write I/O Buffer)
Access device
Write data block data to IPR (Write IPR)
Access device
Write run time value to OWUS (Write Status)
Access device
Run Micro (Start Program)
Power on
Wait for as long as the run time value specifies
Power off
Access device
Get device status (Read Status)
Command completed ? (OWMS bit 5 = 0) no yes
Access device
Write new run time value to OWUS (Write Status)
Access device
Run Micro (Continue Program)
Continue this loop until command is completed
Access device
Read data block header from I/O Buffer (Read I/O Buffer)
Access device
Read data block data from IPR (Read IPR)
1-Wire Reset (Reset/Presence Sequence)

Simplifications:

- All input data required to execute the firmware function command fits into one data block. For multiple input data blocks see Figure 2.
- All output data generated by the firmware function command fits into one data block. For multiple output data blocks see Figure 3.
- The device is assumed to be ready to receive a new firmware function command. To verify the device status and complete an interrupted command see Figure 4.
- Data written to the device is not read back for verification. For verification see note following Figure 4.
- No error handling is done. Error codes, their occurance, meaning and corrective actions are discussed later in this appendix.

In case N data blocks have to be transmitted rather than 1 the first four statements of Figure 1 are replaced by the flow chart in Figure 2.

Flow Chart For Multiple Data Blocks To Be Transmitted Figure 2

For data blocks 1 to N-1

Access device (Reset/Presence Sequence, Match ROM)

Write data block header to I/O Buffer (Write I/O Buffer)

Access device

Write data block data to IPR (Write IPR)

Access device

Write minimum run time value to OWUS (Write Status)

Access device

Run Micro (Start Program)

Power on

Wait for as long as the run time value specifies

Power off

Access device

Write **last** data segm. header to I/O Buffer (Write I/O Buffer)

Access device

Write **last** data block data to IPR (Write IPR)

(continued as shown in Figure 1)

In case the output data generated by a firmware function command extends over several data blocks the end section of Figure 1 is replaced by the flow chart in Figure 3.

Flow Chart For Multiple Data Blocks To Be Received Figure 3

(from Figure 1)

Continue this loop until command is completed

Access device (Reset/Presence Sequence, Match ROM)

Read data block header from I/O Buffer (Read I/O Buffer)

Access device

Read data block data from IPR (Read IPR)

Last block ? (MS Bit of Block Nr. = 1) no yes

Access device

Write minimum run time value to OWUS (Write Status)

Access device

Run Micro (Continue Program)

Power on

Wait for as long as the run time value specifies

Power off

Continue this loop until all blocks are received

1-Wire Reset (Reset/Presence Sequence)

In any case it is recommended to verify that the Crypto iButton is ready to receive a new command before one tries to execute another firmware function. The flow chart in Figure 4 shows the necessary steps.

After this check any previously interrupted command will definitely be completed and one can continue with the flow chart of Figure 1. Any output data that could have been generated by the interrupted command will be discarded automatically by the firmware in order to maintain privacy.

Check If The Device Is Ready For A New Command Figure 4

Check for readiness / complete interrupted command

Access device (Reset/Presence Sequence, Match ROM)

```
Get device status (Read Status)
 yes      Accellerator running  ? (CPST≠ 0)          no
          Cmd. compl. ? (OWMS Bit 5 = 0)  no         yes
          Access device
          Write run time value to OWUS (Write Status)
Access device
Run Micro (Continue Program)
Power on
Wait for as long as the run time value specifies *
Power off
Continue this loop until command is completed
```

* If the arithmetic accellarator is running the run time value cannot be specified. Therefore the waiting time has to be 3812.5 ms to be on the safe side.

Data Verification
Verification of data written to the I/O buffer is solely based on the CRC16 that the Crypto iButton responds with after as many bytes as indicated by the length byte have been transmitted. Data written to the IPR may be read back for verification. However, it requires less time and program code and it is safe to rely on the CRC16 that the Crypto iButton responds with after the data has been written to the IPR. More details on reading and writing the I/O buffer and the IPR are found in the Crypto iButton data sheet.

OWUS Run Time Specification
If power is available, the microcontroller inside the Crypto iButton will run as long as the code written to the OWUS register specifies. Only the lower 4 bits of the OWUS content are relevant. The formula is: Run_Time = number * 250 ms + 62.5 ms. For the majority of firmware functions the minimum value of 62.5 ms is by far enough. For the number-crunching functions such as generation of key sets or modulus and exponent and de- or encryption the run time can be a few seconds or longer. Even with the maximum run time value of 3812.5 ms several cycles may be necessary.

OWMS Error Codes

The portion of the firmware that takes care of the correct data transfer to and from the Crypto iButton has its own set of error codes. These codes are available to the bus master through the lower 6 bits of the OWMS register that is read with the Read Status command. They must not be confused with the error codes that are generated by the command interpreter or script interpreter (see Appendix A). Those error codes are read by the bus master from the IPR as the result of the execution of a firmware function command.

0 CE_Reset System is reset
Occurrence: if the Crypto iButton is ready for a new command
Corrective action: none

1 CE_MsgInComp Message incomplete
Occurrence: if one or more blocks of a multi-block command have been transmitted
Corrective action: send the remaining blocks

2 CE_BadSeq Blocks missing or out-of-sequence
Occurrence: if blocks of a multi-block command are not transmitted in their natural sequence or a block is skipped
Corrective action: reset Crypto iButton (Reset Micro command) and repeat sending the firmware command and its data

3 CE_BufOverrun Message Buffer overrun occurred

Occurrence: if a multi-block command exceeds the size of the internal command buffer; currently the buffer size is 256 bytes.
Corrective action: modify the command and its parameters to fit into 256 bytes

4 CE_BadCKSum Running checksum failure
Occurrence: if the checksum in the header of a block does not match the checksum calculated by the CryptoiButton
Corrective action: re-transmit header and data of the block and start the micro again

5 CE_HdrSize Bad header length found in I/O buffer
Occurrence: if the block header is not 8 bytes long
Corrective action: re-transmit the header and start the micro again

6 CE_DataSize Bad data length found in IPR
Occurrence: if the number of bytes written to the IPR differs from the block length value in the block header
Corrective action: re-transmit header and data of the block and start the micro again

7 CE_BadCRC Bad CRC check between header & data block
Occurrence: if the CRC in the header of a block does not match the CRC calculated by the Crypto iButton
Corrective action: re-transmit header and data of the block and start the micro again

9 CE_FFONotEmpty Master failed to read I/O buffer completely
Occurrence: if the bus master has not read all bytes of the I/O buffer
Corrective action: read status to get the number of unread bytes and read the I/O buffer again for the remaining bytes

10 CE_Standby No more data, standing by
Occurrence: if a firmware command is completed and the micro is run again (continue program command)
Corrective action: none

11 CE_ResponseRdy Response message to host has been loaded
Occurrence: if the CryptoiButton has the first block of a multi-block response message ready in the I/O buffer and IPR for the bus master to read
Corrective action: read I/O buffer to get the length of the data block and then read the data from the IPR

12 CE_RespIncomp Response message incomplete
Occurrence: if the CryptoiButton has another block of a multi-block response message ready in the I/O buffer and IPR for the bus master to read
Corrective action: read I/O buffer to get the length of the data block and then read the data from the IPR

13 CE_NoHeader No header found after Start Program command
Occurrence: if the CryptoiButton is run (Start Program command) and the bus master has not written a data block header to the I/O buffer
Corrective action: re-transmit header and data of the block and start the micro again

29 CE_FirstBirth Device is in first-birthday initialization
Occurrence: if a master erase command has been sent that has not yet been completed
Corrective action: give power for 4 seconds to complete the command

32 to 63 CE_CIInComp Command interpreter incomplete status

Occurrence: if a firmware function command is not yet completed; the lower the number, the closer the command is to completion

Corrective action: write a new run time value to OWUS and run the micro (Continue Program command)

Message Fragmentation and Block Formatting

When a message longer than 128 bytes or a smaller user-defined size is exchanged between bus master and Crypto iButton it is necessary to fragment the message into blocks. To be able to re-assemble the message error free either inside the CryptoiButton or the bus master each block is accompanied by a control header. A header is always eight bytes in length. The size of the data block may vary from 1 to 128 bytes.

The 8-byte header is formed as follows:

Byte Number	**Description**
1	Block Number
2	Block Length
3	Remaining Length, Low byte
4	Remaining Length, High byte
5	Block CRC-16, Low byte
6	Block CRC-16, High byte
7	Check sum, Low byte
8	Check sum, High byte

Definitions

Block Number Counting starts with 0 and increments by 1 with every subsequent block. For the last block the most significant bit of the block number is set to 1. This convention allows detecting blocks that are out of sequence. The maximum number of blocks that can be sent in a single message to 128.

Block Length The number of bytes to be exchanged through the IP Register. Valid numbers are 1 to 128 decimal or 1 to 80 hex. A zero value is not allowed.

Remaining Length This 16-bit value represents the number of message bytes that have *not yet been transmitted successfully*, *including* those in this block. In the first block, this value wil be the length of the entire message. In the last block, this value will equal the block length byte.

Block CRC-16 This 16-bit value is the non-inverted CRC-16 check of the length of the block and the message data in the block. The CRC generator starts with all zeros at the beginning of each block. See the example on the subsequent pages for details.

Check sum This 16-bit value represents the running modulo-65536 sum of all the data and header bytes that have been sent since the start of the message up to but not including these check sum bytes themselves. The check sum accumulator starts will all zeros at the first block. Then every byte starting with the block number, block length, remaining length followed by the data bytes and the CRC16 of the block are added. This results is then transmitted as check sum for the first block. The starting value of the check sum accumulator for the next block is obtained by adding the low byte and high byte of the transmitted check sum to the transmitted check sum. See the example on the subsequent pages for details.

Block Fragmentation Example

The following pages show the calculation of the headers of a 12-byte message that is to be transmitted in three blocks might appear as follows. The data content of this example message is a 01,02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C (all hexadecimal). The block length is chosen to be 4 bytes for the first block, 3 for the second and 5 for the last block.

Header Calculation For The First Block

Description	**Input Value**	**CRC16 Calculation**	**Check Sum Calculation**	**Resulting Header**
starting condition		0000	0000	
block number	**00**	not counted	0000	**00**
block length	**04**	C301	0004	**04**
remaining length low	**0C**	not counted	0010	**0C**
remaining length high	**00**	not counted	0010	**00**
data byte #1	01	00C3	0011	
data byte #2	02	90C1	0013	
data byte #3	03	9111	0016	**50**
data byte #4	04	CF50	001A	
CRC low	50		006A	
CRC high	CF		0139	**CF**
check sum low	39		0172	**39**
check sum high	01		0173	**01**

Header Calculation For The Second Block

Description	**Input Value**	**CRC16 Calculation**	**Check Sum Calculation**	**Resulting Header**
starting condition		0000	0173	
block number	**01**	not counted	0174	**01**
block length	**03**	0140	0177	**03**
remaining length low	**08**	not counted	017F	**08**
remaining length high	**00**	not counted	017F	**00**
data byte #1	05	F3C0	0184	
data byte #2	06	5273	018A	**52**
data byte #3	07	2752	0191	
CRC Low	52		01E3	
CRC High	27		020A	**27**
check sum low	0A		0214	**0A**
check sum high	02		0216	**02**

Header Calculation For The Last (Third) Block

Description	**Input Value**	**CRC16 Calculation**	**Check Sum Calculation**	**Resulting Header**
starting condition		0000	0216	
block number	**82**	not counted	0298	**82**
block length	**05**	03C0	029D	**05**
remaining length low	**05**	not counted	02A2	**05**
remaining length high	**00**	not counted	02A2	**00**
data byte #1	08	9602	02AA	
data byte #2	09	C7D7	02B3	
data byte #3	0A	5907	02BD	

data byte #4	0B	0559	02C8	**C5**
data byte #5	0C	3FC5	02D4	
CRC Low	C5		0399	
CRC High	3F		03D8	**3F**
check sum low	D8		04B0	**D8**
check sum high	03		04B3	**03**

A message of 12 bytes is normally not split into three blocks and it is also not common to change the size of each block. The example above was chosen to explain the most general case only. One could have transmitted the same message in a single piece. In that case the header would have been 80, 0C, 0C, 00, 47, 9A, C7, 01, all values in hexadecimal.

Due to the hardware design of the IPR, the maximum size of a data block is 128 bytes. This is sufficient for almost all commands. However, depending on the electrical contact between Crypto iButton and bus master, a smaller block size may be more efficient if the contact is intermittent. It takes less time to resent a few bytes rather than 128 bytes if only a single byte is corrupted. Regardless what block size is cosen, all but the last block have the same size.

The manual calculation of the check sum and the CRC16 is very time consuming and error prone. To simplify the debugging of a application-specific program that communicates with the Crypto iButton on the hardware level, a simple program has been developed that generates the header information based on the specified block size and hexadecimal input data. This program is listed on the following pages. Copies can be requested via EMAIL to AutoID.Support@dalsemi.com.

Header Calculation Program

```
'
' Com Layer Message Former -
'
Start:
    CLS
    LOCATE 24, 1
    INPUT "Maximum Segment Length (default=128): "; MaxBlock%
    IF MaxBlock% = 0 THEN MaxBlock% = 128
ReDo:
    PRINT "Input a string of two-character hex values separated by spaces (Q to quit):"
    INPUT a$
    IF UCASE$(a$) = "Q" THEN END
    a$ = LTRIM$(RTRIM$(a$)) + " "
    msg$ = ""
    FOR n% = 1 TO LEN(a$) STEP 3
        IF MID$(a$, n% + 2, 1) <> " " THEN
            BEEP
            PRINT "Bad hex string format"
            GOTO ReDo
        END IF
        msg$ = msg$ + CHR$(VAL("&H" + MID$(a$, n%, 2)))
    NEXT n%
    PRINT : PRINT

    BlockNumber% = 0
    Remain% = LEN(msg$)
    cksum& = 0

    DO WHILE Remain% > 0
        PRINT
        PRINT "Block Number ="; BlockNumber%; TAB(30); "Bytes remaining:"; Remain%
        PRINT
```

```
' Compute the length of the message segment to send -
IF Remain% <= MaxBlock% THEN
    SegLen% = Remain%              ' Use length of remaining msg
ELSE
    SegLen% = MaxBlock%            ' Use maximum length
END IF

' Extract the desired segment from the message -
segment$ = MID$(msg$, (BlockNumber% * MaxBlock%) + 1, SegLen%)

' Build the header -
IF SegLen% = Remain% THEN
    x% = BlockNumber% OR &H80
ELSE
    x% = BlockNumber%
END IF
cksum& = 65535 AND (cksum& + x%)
PRINT " Header byte 1"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

' Add segment length and length remaining to header -
x% = SegLen%
Header$ = Header$ + CHR$(x%)
cksum& = 65535 AND (cksum& + x%)
PRINT " Header byte 2"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

x% = Remain% AND 255
Header$ = Header$ + CHR$(x%)
cksum& = 65535 AND (cksum& + x%)
PRINT " Header byte 3"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

x% = Remain% \ 256
Header$ = Header$ + CHR$(x%)
cksum& = 65535 AND (cksum& + x%)
PRINT " Header byte 4"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)
PRINT

' Compute the crc16 and checksum of the message segment -
crc& = 0
x% = SegLen%
GOSUB DoCRC16
PRINT "Segment Length (hex) = "; HEX$(SegLen%); TAB(40); "crc = "; HEX$(crc&)
PRINT
FOR ln% = 1 TO SegLen%
    byt% = ASC(MID$(segment$, ln%, 1))
    cksum& = 65535 AND (cksum& + byt%)
    x% = byt%
    GOSUB DoCRC16
    char% = byt% AND 127
    IF char% < 32 THEN char% = 32
    PRINT "Segment byte"; ln%; " (hex)"; TAB(25); HEX$(byt%); TAB(30); CHR$(char%); TAB(40);
"crc = "; HEX$(crc&); TAB(55); "cksum = "; HEX$(cksum&)
NEXT ln%
PRINT

x% = crc& AND 255
Header$ = Header$ + CHR$(x%)
cksum& = 65535 AND (cksum& + x%)
PRINT " Header byte 5"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

x% = crc& \ 256
Header$ = Header$ + CHR$(x%)
```

```
        cksum& = 65535 AND (cksum& + x%)
        PRINT " Header byte 6"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

        z& = cksum&

        x% = z& AND 255
        Header$ = Header$ + CHR$(x%)
        cksum& = 65535 AND (cksum& + x%)
        PRINT " Header byte 7"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)

        x% = z& \ 256
        Header$ = Header$ + CHR$(x%)
        cksum& = 65535 AND (cksum& + x%)
        PRINT " Header byte 8"; " (hex)"; TAB(25); HEX$(x%); TAB(55); "cksum = "; HEX$(cksum&)
        BlockNumber% = BlockNumber% + 1
        Remain% = Remain% - SegLen%
              PRINT
        INPUT "Hit ENTER key..."; z$
        PRINT : PRINT

    LOOP
    GOTO Start
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' The value in x% is the input byte value, crc& is the running result -
' The CRC-16 polynomial is 0xA001 (1001 0000 0000 0001)

DoCRC16:

    ' Repeat the iteration once for each of the eight bits -
    FOR n% = 0 TO 7

        ' Compute the XOR sum of the LS bit of x% with the lsb of crc% -
        bit% = (crc& XOR x%) AND 1
             ' Rotate crc& right one position (zero into ms bit) -
        crc& = crc& \ 2
        ' If the xor of the ls bits was a '1', apply the polynomial -
        IF bit% = 1 THEN crc& = (65535 AND (crc& XOR &HA001))
        ' Rotate the input byte to get the next bit into LS position -
        x% = x% \ 2

    NEXT n%
    RETURN
```

Glossary

API

The Crypto iButton Application Programming Interface (API) is a document containing the function prototypes of every high-level function available to the user's program and a description of the actions, formal parameters, and return values of the functions. The API is presented in the Crypto iButton Firmware Reference Manual The PC implementation of the Crypto iButton API is provided in the dynamic link library CiBAPI.dll. The UNIX implementation is provided in the archive file cibapi.a

Blue Dot™ Receptor

A mating connector for a Crypto iButton. Pressing the Blue Dot with the Crypto iButton snaps it into position. The Blue Dot Receptor is connected to a PC through a DS1410E parallel port adapter. It may also be connected to a UNIX machine (or other computer having an RS232C serial port) with a DS9097U serial port adapter.

Clock Offset

This is one of the basic data types supported by the Crypto iButton. To obtain the actual date and time (Real Time) as a long integer number of seconds since a predefined "zero" date, the value of this offset is added to the value provided by the True Time Clock. Since every Transaction Group can define a different Clock Offset object, the definition of the starting point for time measurement is under the control of the Service Provider who programs the Transaction Group. The commonly used UNIX standard reference date for time measurement is 00:00:00 a.m. on January 1, 1970.

Communication Layer

This is an intermediate layer of software that manages the complex task of providing error-free communication between the host and the Crypto iButton. The Win32 implementation of the Communication Layer is contained in the dynamic link library CiBComm.dll.

Configuration Data

This is one of the basic data types supported by the Crypto iButton. It is an unstructured data type which can accept any type of data. Each Transaction Group may contain one or more Configuration Data objects. The configuration data may be any kind of information required by the application. Examples include text to identify the version number and other relevant information about the application, certificates signed by a Certifying Authority binding a public key with a particular end user, and even as intermediate storage for complex calculations performed by Transaction Scripts.

Destructor

This is one of the basic data types supported by the Crypto iButton. A destructor is a data object that can be added to any Transaction Group. It contains a value of the True Time Clock which acts as an expiration time for any destructible Transaction Script or other data object. When the value of the True Time Clock is greater than or equal to the value in the destructor, the destructible Transaction Scripts and destructible data objects no longer function and return an error message. A destructor can be used to selectively eliminate certain scripts or data objects on a particular date in the future. A destructor has no effect on the operation of the group until it expires.

Documentation

Technical Documentation for the Crypto iButton is available on the internet through the URL http://www.ibutton.com/crypto in the section titled "Additional Information". The primary technical reference document to be found there is the Crypto iButton Firmware Reference Manual. This reference manual outlines the design and development process for Crypto iButton applications and provides the complete API specification for the Crypto iButton's embedded firmware. Other significant documents include Cryptographic iButton Script Language, which serves as a guide to creation of the scripts to support specialized applications of the Crypto iButton, and Guide to the Open Standard Feature Set of the Crypto iButton, which describes the cryptographic features expressed in the Dallas Primary Group.

E-Mail Demonstration

Dallas Semiconductor provides an encrypted e-mail demonstration website which uses the Primary Group stored in the Crypto iButton to encrypt and decrypt messages sent from one registered user to another. The present URL for this site is http://crypto.ibutton.com/email . A full-featured encrypted e-mail service is under development by another company.

Exponent

An integer number used in RSA encryption. There are two exponents, one used for encryption and the other for decryption. The decryption exponent is a large number, whereas the encryption exponent may be large or small. The Crypto iButton allows a 1024 bit exponent. This is one of the basic data types supported by the Crypto iButton.

Input Data

This is one of the basic data types supported by the Crypto iButton. Data can be transmitted to a Crypto iButton from the host and placed in an Input Data object, to be acted on by a Transaction Script. The Transaction Script typically performs an operation involving the input data and one or more stored data values and places the result in one or more Output Data objects.

Modulus

A large integer number used in RSA encryption. The modulus and the public exponent constitute the public key, and the modulus and private exponent constitute the private key. The Crypto iButton allows 1024 bit moduli. This is one of the basic data types supported by the Crypto iButton.

Money Register

This is one of the basic data types supported by the Crypto iButton. It is an unsigned integer with 1 to 128 bytes of precision, as determined by the Service Provider who programs the Transaction Group. The values in Money Registers can be manipulated by the arithmetic operators contained in the Transaction Scripts.

Output Data

This is one of the basic data types supported by the Crypto iButton. Output data resulting from the execution of a Transaction Script is placed in one or more Output Data objects, where it can be read by the host.

Primary Group

A Transaction Group that Dallas Semiconductor programs into every Crypto iButton to support basic cryptographic services. The services provided by this group are described in the document entitled, Guide to the Open Standard Feature Set of the Crypto iButton. This group is stored by the name "Dallas Primary" in the Crypto iButton. It supports RSA encryption/decryption and digital signature generation. (There is an exportable version which supports only digital signature.) The encryption and decryption capability can be used with the Dallas Semiconductor Crypto iButton E-Mail Demonstration website to send and receive RSA encrypted e-mail to other registered users.

Random Salt

This is one of the basic data types supported by the Crypto iButton. It is a random number which can be used as a challenge to authenticate another Crypto iButton. The Crypto iButton can remember the previously issued Random Salt so that it can confirm the validity of the response. This is useful when passing monetary value securely from one Crypto iButton to another.

Real Time Clock

This is the time calculated by adding the value of the Clock Offset in a Transaction Group to the value provided by the True Time Clock. The Real Time Clock provides a measure of the number of seconds since a predefined zero reference date. The Clock Offset is used by the Service Provider to set the Real Time Clock to the correct date and time.

Script Compiler

A program which takes a formal description of a Transaction Group written in Crypto iButton Script Language and produces a bytecode representation of the scripts which can be interpreted and executed by the Script Interpreter. The Script Compiler is named SCompile.exe. It takes an input Script Language file named <script> and a symbol declaration file named <script>.sym and produces a bytecode file named <script>.out. Sample contents of these files are presented in the documents entitled Cryptographic iButton Script Language and Guide to the Open Standard Feature Set of the Crypto iButton.

Script Interpreter

One of the major components of the Crypto iButton's operating system. The operating system firmware of the Crypto iButton contains an input-output subsystem to communicate with the outside world, a command interpreter to intercept and execute commands from the outside world, a memory-management subsystem, and a Script Interpreter that interprets the stored Transaction Scripts.

Script Language

The source code for specifying the contents of a Transaction Group and writing Transaction Scripts. The document entitled Cryptographic iButton Script Language available as a link from http://www.ibutton.com/crypto describes the attributes of the Script Language and provides example code.

Service Provider

Crypto iButtons are designed to be issued to a Service Provider. The Service Provider designs an application for the Crypto iButton, writes application-level software for the host, and programs a Transaction Group in the Crypto iButton to provide the security services required by the application. The Service Provider then co-issues the Crypto iButtons to his own customers. The Service Provider may be able to utilize the features provided in the Primary Group for some services (such as secure web e-mail) without having to program a proprietary Transaction Group.

Software

Software has been written for Windows 95, Windows NT, and UNIX to implement the Crypto iButton API specification as defined in the Crypto iButton Firmware Reference Manual For Windows 95 and Windows NT, this API is made available through the dynamic link library CiBAPI.dll which provides all of the high-level function calls defined in the specification. This library communicates with the Crypto iButton through the Communication Layer provided in CiBComm.dll. The Communication Layer uses error detecting and correcting methods to transport data without errors between the Crypto iButton and the PC. When used with the Blue Dot receptor attached to the DS1410E parallel port adapter, the Communication Layer must call on a device driver to transfer bits, bytes, and reset signals to and from the Crypto iButton. The Windows 95 device driver is VSauthD.vxd, and the Windows NT device driver is DS1410D.sys. For UNIX, the API is made available through cibapi.a, which contains both the API implementation and the Communication Layer.

Transaction Counter

This is one of the basic data types supported by the Crypto iButton. Its purposes are to maintain a record of the number of transactions performed and to identify transactions by a unique set of sequential numbers. When a Transaction Counter appears on the right-hand side of an assignment statement in a script, the Script Interpreter will automatically increment the Transaction Counter.

Transaction Group

A collection of constants, variables, and procedures in a Crypto iButton that are designed to accomplish particular tasks or provide a particular set of services. Each Transaction Group stored in the Crypto iButton is independent of every other group. Memory in a Transaction Group may be allocated for the following types of data:

Clock Offset
Configuration Data
Destructor
Exponent
Input Data
Modulus
Money Register
Output Data
Random Salt
Transaction Counter

In addition to these data types, there are also executable procedures called Transaction Scripts which can perform mathematical and cryptographic operations among the data types listed above.

Transaction Script

This is one of the basic data types supported by the Crypto iButton. A Transaction Script is a bytecode or p-code procedure which acts on the contents of one or more data objects in the transaction group and produces one or more results which are stored in data objects of the group. Transaction Scripts are obtained by writing a set of procedures in the Script Language of the Crypto iButton and compiling them with the Script Compiler to produce the Transaction Scripts that are stored in the Crypto iButton. (In Object-Oriented Programming terminology, a Transaction Group may be thought of as an object that is an instance of the class defined by the Script Language, and the Transaction Scripts are methods which act on the instance variables of the class.)

True Time Clock

This is a continuously running clock implemented in hardware in the Crypto iButton, controlled by a quartz crystal, and continuously powered by the built-in lithium energy source. This clock starts when the Crypto iButton is assembled and runs continuously. Attempts to alter its timekeeping generate a tamper response that leaves evidence of the abuse.