



advanced card systems ltd



ADT60

Application Programming Interface



BioSIMKey

Fingerprint Scanner & Plug-in Smart Card Reader

Advanced Card Systems Ltd.

Room 302, 3/F., Shun Fat Industrial building,
17 Wang Hoi Road, Kowloon Bay, Kowloon, Hong Kong.

Tel: 852-2796 7873 Fax: 852-2796 1286

Website: www.acs.com.hk E-mail : info@acs.com.hk

Table of Contents

1. Scope	3
2. Abbreviations	3
3. Product Information	4
3.1 Product Description	4
3.2 Fingerprint Scanner Device	4
3.2.1 Biometric Process	4
3.2.2 GUICallback	5
3.3 Smartcard Reader Device	6
4. Biometrics Application Programmer Interface	7
4.1 Data Structure Definition	7
4.1.1 Data Types	7
4.1.2 Data Constants	7
4.1.2.1 Finger Position Status	8
4.1.2.2 Information Messages	8
4.1.2.3 Biometric Device Status	9
4.1.2.4 VerifyMatch Result	9
4.1.2.5 Biometric Error Codes	9
4.1.3 GUI Callback Structure11	11
4.1.3.1 _AC_ADT_GUI_CALLBACK	11
4.1.3.2 PAC_ADT_SET_FINGER_STATUS	12
4.1.3.3 PAC_ADT_SET_MESSAGE	13
4.1.3.4 PAC_ADT_SET_ADT_STATUS	14
4.2 Service Provider Interface	15
4.2.1 AC_ADT_Init	15
4.2.2 AC_ADT_Terminate	16
4.2.3 AC_ADT_Capture	17
4.2.4 AC_ADT_Process	19
4.2.5 AC_ADT_VerifyMatch	21
4.2.6 AC_ADT_FreeTemplate	23
4.2.7 AC_ADT_CompressTemplate	24
4.2.8 AC_ADT_SetGUICallbacks	25
4.2.9 AC_ADT_GetNVMSize	26
4.2.10 AC_ADT_ReadNVM	27
4.2.11 AC_ADT_WriteNVM	28

1. SCOPE

This document provides the user with the relevant information on the product and its usage. The product information section describes the device, its hardware components, and the underlying processes involved in its operations. The Application Programming Interface (API) defines the programming structures and functions needed by the user to be able to develop specific applications for the use of the device.

2. ABBREVIATIONS

Abbreviation	Explanation
API	Application Program Interface. A language and message format used by an application program to communicate with the operating system. APIs are implemented by writing function calls in the program, which provide the linkage to the required subroutine for execution.
FAR	False Acceptance Rate. The probability that a wrong sample is accepted as identical in characteristics to the reference data template.
FRR	False Rejection Rate. The probability that a right sample is found not identical in characteristics to the reference data template.
PC/SC	Personal Computer Smart Card. An implementation framework that integrate smart card applications into computer systems.
USB	Universal Serial Bus. A hardware interface for low-speed peripherals such as the keyboard, mouse, joystick, scanner, printer and telephony devices. USB has a maximum bandwidth of 12 Mbits/sec (equivalent to 1.5 Mbytes/sec).

3. Product Information

3.1 Product Description

The ADT60 is a USB security device that performs as a combination fingerprint scanner and smartcard reader.

By design, the device combines the inherent qualities and advantages attainable in the current states of biometrics and smart card technologies. The dual functionality of the device is properly integrated such that interfaces and functions are kept distinct and streamlined for ease of implementation. The device uses proprietary APIs for the fingerprint scanner component and the PC/SC framework for the smart card reader component. The ADT60 device is also designed with USB connectivity for portability and easy hardware integration.

By function, the device significantly increases the security benefits that can be derived by the combined use of the biometrics and smart card technologies as compared to singular devices using either technology. By adding the smart card security layer over the biometrics security layer, this singular device can effectively multiply the security level in an application by a factor of the added security layer.

The multi-functionality and portability of ADT60 make the device an ideal solution for most types of security applications.

3.2 Fingerprint Scanner Device

The fingerprint scanner component in ADT60 is a complete biometrics device that can read and process fingerprint data.

3.2.1 Biometric Process

The biometric process is a complete cycle of capturing, storing, reading and interpretation of a measurable physiological or behavioral characteristic of an individual. The ADT60 device implements biometrics using the fingerprint of the user as the primary data. The basic functions of the device are capture, process, and match. Refer to Fig.1.

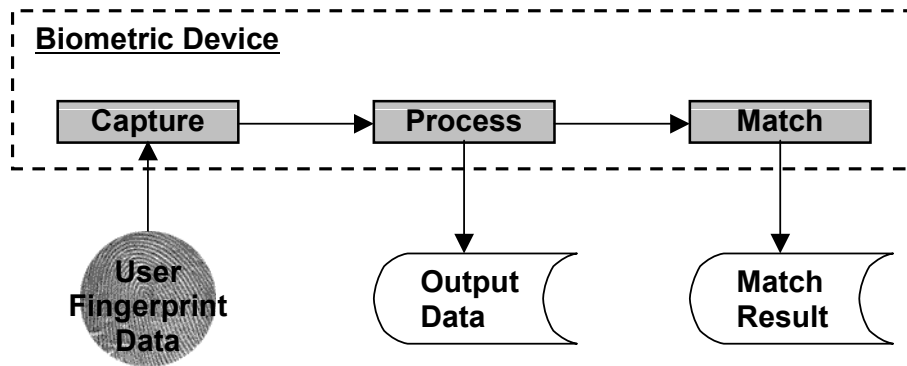


Fig.1. The basic biometric process

The capture stage detects the finger in the sensor area of the device, transforms the data into a usable form and provides the template into which the data can be stored. The capture process can be invoked for either enroll or verification purpose. For enroll purpose, at least three data samples must be captured for a successful image capture. For verification purpose, a single fingerprint image is needed. The device support library implements a `GUICallback` function that automates the execution of the image capture based on user interaction. (Please refer to the succeeding section.)

The process stage transforms the intermediate data template output from the capture stage into a usable form. The process output data may be stored or used directly for the matching process.

The match stage executes the proprietary algorithm to compare two data templates in order to determine whether both templates represent the same fingerprint image. The matching process can be controlled by the FRR and FAR values. Preference for each constraint is dependent on the type of application. For ADT60, the user can only specify the desired FAR constraint.

3.2.2 `GUICallback`

Considering the infinite ways a user may place his fingerprint on the sensor area, the biometric device library spares the user from the repetitious routine of successfully capturing the required fingerprint images. For this purpose, it does provide the mechanism within which the user can monitor the status and possible errors during the process execution through the `GUICallback` functions. (Refer to Fig. 2.)

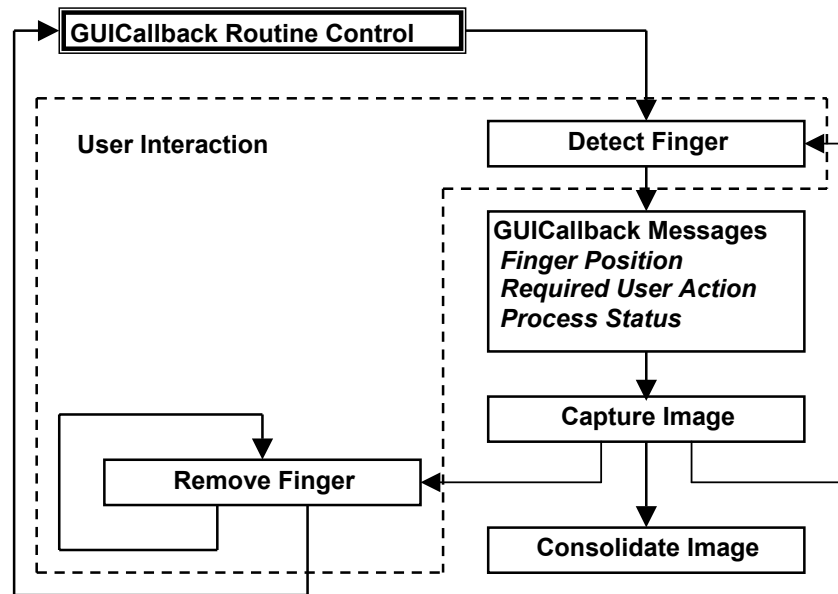


Fig. 2. Capturing the fingerprint image through the GUI callback routine

The GUICallback functions must be registered after the initialization of the fingerprint scanner device to enable the user to access the messages returned by the functions. The execution of the GUICallback routine is controlled by the module which in turns takes in the user interaction as input. For example, the programmer does not need to specify in his code when the program execution jumps to, say, first or succeeding captures of the image. The program automatically steps into successive stages, say, from “Detect Finger” to “Remove Finger” when the “Capture Image” step returns the required conditions. To guide the user on the proper interaction with the device, the GUICallback messages should be properly monitored and displayed. After completing the required number of successful image captures, the module executes “Consolidate Image” and returns the program execution to the control of the programmer.

3.3 Smart card Reader Device

The smart card reader component in ADT60 is a PC/SC-compliant device. For more information regarding PC/SC implementation, please refer to MSDN, or visit <http://msdn.microsoft.com/> or <http://www.pcscworkgroup.com/>.

4. BIOMETRICS APPLICATION PROGRAMMER INTERFACE

4.1 Data Structure Definition

This section defines the data types and structures used in the different modules for the biometrics device component of ADT60.

4.1.1 Data Types

Definition	Description
AC__VOID	Identifies a void.
AC__CHAR	Identifies a char (1 byte).
AC__UCHAR	Identifies an unsigned char (1 byte).
AC__SHORT	Identifies a short integer (2 bytes).
AC__USHORT	Identifies an unsigned short integer (2 bytes).
AC__INT	Identifies an integer (4 bytes).
AC__UINT	Identifies an unsigned integer (4 bytes).
AC__LONG	Identifies a long integer (4 bytes).
AC__ULONG	Identifies an unsigned long integer (4 bytes).
AC__BYTE	Identifies a byte (1 byte).
AC__BOOL	Identifies a Boolean: TRUE (1) or FALSE (0).
AC__STATUS	Identifies a result of a function or variable that defines a status (2 bytes).
AC__FLOAT	Identifies a float (4 bytes).
AC__BITFIELDS	Identifies a 32-bit integer (4 bytes).

4.1.2 Data Constants

This section defines the data constants used in the different modules for the biometrics device component of ADT60.

4.1.2.1 Finger Position Status

The finger position status is of AC_STATUS type.

Definition	Value	Description
ADT_FINGER_OK	0	The finger is correctly placed on the sensor.
ADT_MOVE_UP	-1	The finger is on the bottom part of the sensor area.
ADT_MOVE_DOWN	-2	The finger is on the top part of the sensor area.
ADT_MOVE_LEFT	-3	The finger is on the right part of the sensor area.
ADT_MOVE_RIGHT	-4	The finger is on the left part of the sensor area.
ADT_MOVE_PRES_HARDER	-5	The finger does not cover a sufficient area on the sensor.
ADT_BAD_QUALITY	-6	Something is detected on the sensor but it does not have the required quality for processing.
ADT_NO_FINGER	-7	There is no finger detected on the sensor area.

4.1.2.2 Information Messages

The information message value is of AC_INT type.

Definition	Value
VERIFY_FINGER	0x00000001
ENROL_FINGER0_FIRST_CAPTURE	0x00000002
ENROL_FINGER0_SECOND_CAPTURE	0x00000003
ENROL_FINGER0_THIRD_CAPTURE	0x00000004
CAPTURE_FINGER	0x00000005
ENROL_FINGER	0x00000006
REMOVE_FINGER	0x00000007
CONSOLIDATE_FINGER	0x00000008

4.1.2.3 Biometric Device Status

The biometric device status is of AC_UCHAR type.

Definition	Value
PROCESS_FINGER_DETECT	0x01
GRAB_IMAGE	0x02
PROCESS_ENROL	0x03
PROCESS_CONSOLIDATE	0x04
PROCESS_VERIFY	0x05

4.1.2.4 VerifyMatch Result

The VerifyMatch is of AC_STATUS type.

Definition	Value
ADT_MATCH	0
ADT_REJECT	-1

4.1.2.5 Biometric Error Codes

The biometric error code is of AC_STATUS type.

Definition	Value	Description
ADT_RET_OK	0	Successful.
ADT_RET_ERR	-1	Undefined error.
ADT_RET_BAD_PARAMETERS	-2	At least, one parameter is not correct.
ADT_RET_SENSOR_IS_DIRTY	-3	The sensor might need to be cleaned.
ADT_RET_NO_INIT	-4	The AC_ADT_Init function was not called.

ADT_RET_NO_CALLBACK	-5	The AC_ADT_SetGUICallbacks function was not called.
ADT_RET_CANCEL	-6	One of the callback functions returned FALSE in the parameter ContinueFlag.
ADT_RET_BAD_CONSOLIDATE	-7	Consolidation failed because of poor quality or non-matching images.
ADT_RET_DETECTED_TRACE	-8	It is possible that somebody tries to use a trace on the sensor to fool the system.
ADT_RET_NO_DEVICE	-9	The device is not found.
ADT_RET_BASE_FULL	-10	Database management error.
ADT_RET_DEVICE_IN_USE	-11	The device is already in use.
ADT_RET_ERR_SENSOR	-12	Error from the sensor.
ADT_RET_ERR_BASE_MGR	-13	Database management error.
ADT_RET_INVALID_TEMPLATE	-14	At least one template is not correct.
ADT_RET_NO_SC_READER	-15	The system cannot detect the smartcard reader.
ADT_RET_DRIVER_NO_INIT	-16	The driver has not been properly loaded.
ADT_RET_CORRUPTED_VARIABLE	-17	Data in non-volatile memory is corrupted.
ADT_RET_FLASH_NOT_SUPPORTED	-21	No data is read from the non-volatile memory.

4.1.3 GUI Callback Structure

4.1.3.1 _AC_ADT_GUI_CALLBACK

This callback function is used to transfer an information message from the device to the application

```
typedef struct _AC_ADT_GUI_CALLBACK {  
    PAC_ADT_SET_FINGER_STATUS      AC_ADT_SetFingerStatus;  
    PAC_ADT_SET_MESSAGE           AC_ADT_SetMessages;  
    PAC_ADT_SET_ADT_STATUS        AC_ADT_SetADTStatus;  
} AC_ADT_GUI_CALLBACK, *PAC_ADT_GUI_CALLBACK;
```

Parameters

AC_ADT_SetFingerStatus

The function that returns the fingerprint status information with respect to its position in the sensor, and returns the image buffer as an option.

AC_ADT_SetMessages

The function that returns information associated to an action that must be performed by the user.

AC_ADT_SetADTStatus

The function that returns the name of the process currently executed by the device.

Remarks

None

4.1.3.2 PAC_ADT_SET_FINGER_STATUS

This callback function provides the application with information about the positioning of the finger on the device. If requested, it also transmits an image of the sample. The image size is 90 rows and 64 columns.

```
typedef AC__STATUS
(WINAPI* PAC_ADT_SET_FINGER_STATUS) (
    AC__STATUS          FingerStatus,
    AC__UCHAR*         Image,
    AC__ULONG          RowNumber,
    AC__ULONG          ColumnNumber,
    AC__BOOL*         ContinueFlag
);
```

Parameters

FingerStatus

The finger position status.

Image

A valid image buffer that contains the pixels of the captured image. This value may be NULL if no image grabbing is specified or if the FingerStatus value is ADT_BAD_QUALITY or ADT_NO_FINGER.

RowNumber

The number of rows of the captured image.

ColumnNumber

The number of columns of the captured image.

ContinueFlag

The value returned by the application to indicate the module whether to continue processing the current command.

Return Values

Return Value	Meaning
0	Success

4.1.3.3 PAC_ADT_SET_MESSAGE

This callback function transmits an information message from the device to the application.

```
typedef AC__STATUS  
(WINAPI * PAC_ADT_SET_MESSAGE) (  
    AC__INT          MessageID,  
    AC__BOOL*       ContinueFlag  
);
```

Parameters

MessageID

The message identifier.

ContinueFlag

The value returned by the application to indicate the module whether to continue processing the current command.

Return Values

Return Value	Meaning
0	Success

4.1.3.4 PAC_ADT_SET_ADT_STATUS

This callback function transmits the current status of the module to the application.

```
typedef AC__STATUS  
(WINAPI * PAC_ADT_SET_ADT_STATUS) (  
    AC__UCHAR          StatusID,  
    AC__BOOL*         ContinueFlag  
);
```

Parameters

StatusID

The status identifier.

ContinueFlag

The value returned by the application to indicate the module whether to continue processing the current command.

Return Values

Return Value	Meaning
0	Success

4.2 Service Provider Interface

This section describes the different service provider interfaces that can be used to operate the ADT60.

4.2.1 AC_ADT_Init

This function initializes the module. This function should be called at least once by the application.

```
AC__STATUS AC_ADT_Init()
```

Parameters

None

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_DEVICE_IN_USE	Device Error
ADT_RET_NO_DEVICE	Device Error
ADT_RET_ERR_SENSOR	Device Error
ADT_RET_SENSOR_IS_DIRTY	Device Error

Remarks

None

See Also

AC_ADT_Terminate

4.2.2 AC_ADT_Terminate

This function terminates the caller's use of the current module and cleans up all internal states associated with the calling application. This function must be called once for each successful call to **AC_ADT_Init**.

```
AC__STATUS AC_ADT_Terminate()
```

Parameters

None

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error

Remarks

None

See Also

AC_ADT_Init

4.2.3 AC_ADT_Capture

This function captures fingerprint data samples from the device for enrollment or verification, and returns a raw data for further processing by the module.

```

AC__STATUS AC_ADT_Capture (
    AC__UCHAR      CaptureID,
    AC__ULONG      CapturePurpose,
    AC__BOOL       GrabImage,
    AC__UCHAR**    CaptureOutputTemplate,
    AC__ULONG*     CaptureOutputTemplateSize
)

```

Parameters

CaptureID

Input data that controls the capture process. The valid value can either be

- 0** enrollment with one finger,
first capture in enrollment with two fingers, or
verification, or
- 1** second capture process in enrollment with two fingers.

CapturePurpose

Input data that defines the purpose of the image capture. The valid value can either be

- PURPOSE_ENROL** image capture for enrollment, or
- PURPOSE_VERIFY** image capture for verification.

GrabImage

Input data that directs the function to grab the image or not. The valid value can either be

- TRUE**, or
- FALSE**.

CaptureOutputTemplate

The pointer to the template that contains the fingerprint data returned after a successful image capture. This value can be **NULL** in the case of the first capture of the enrollment with two fingers.

CaptureOutputTemplateSize

The size in bytes of the template that will contain the fingerprint data. This value can be **NULL** when the CaptureOutputTemplate is set to **NULL**.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_NO_CALLBACK	Process Error
ADT_RET_BAD_PARAMETERS	Process Error
ADT_RET_CANCEL	Process Error
ADT_RET_BAD_CONSOLIDATE	Process Error
ADT_RET_DETECTED_TRACE	Process Error
ADT_RET_ERR_SENSOR	Device error

Remarks

The module allocates the memory for the CaptureOutputTemplate. When a successful image capture is made, the memory must be de-allocated by a call to **AC_ADT_FreeTemplate**.

See Also

AC_ADT_FreeTemplate, AC_ADT_Process

4.2.4 AC_ADT_Process

This function converts the raw fingerprint data returned by **AC_ADT_Capture** into a structure usable with verification or identification functions.

```
AC__STATUS AC_ADT_Process (
    AC__UCHAR          ProcessID,
    AC__UCHAR*        ProcessInputTemplate,
    AC__ULONG         ProcessInputTemplateSize,
    AC__UCHAR**       ProcessOutputTemplate,
    AC__ULONG*        ProcessOutputTemplateSize
)
```

Parameters

ProcessID

Input data that indicates the type of template to be processed. The valid value can either be

- 1 enrollment with one finger, or
- 2 enrollment with two fingers.

ProcessInputTemplate

The pointer to the input template that contains the fingerprint data to be processed.

ProcessInputTemplateSize

The size in bytes of the input template.

ProcessOutputTemplate

The pointer to the output template that contains the processed data returned by the function.

ProcessOutputTemplateSize

The size in bytes of the output template.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_BAD_PARAMETERS	Process Error

Remarks

None

See Also

AC_ADT_Capture

4.2.5 AC_ADT_VerifyMatch

This function performs a 1-to-1 verification match between two templates.

```
AC__STATUS AC_ADT_VerifyMatch(
    AC__UCHAR    VerifyMatchID,
    AC__INT      FARRequest,
    AC__UCHAR*   VerifyMatchInputTemplate,
    AC__UCHAR*   VerifyMatchRefTemplate,
    AC__STATUS*  VerifyMatchResult
)
```

Parameters

VerifyMatchID

Input data that indicates the template types being matched. The valid value is

- 1** one-finger input template, one-finger reference template,
- 2** one-finger input template, two-finger reference template, or
- 4** two-finger input template, two-finger reference template.

FARRequest

Input data that indicates the requested FAR criterion for a successful verification.

Valid value ranges from **21475** to **214748364**.

VerifyMatchInputTemplate

The pointer to the input template that contains the fingerprint data to be verified.

VerifyMatchRefTemplate

The pointer to the reference template containing the fingerprint data, usually retrieved from storage, against which the user input shall be validated.

VerifyMatchResult

The result of the verification between two templates returned by the function. The value can either be

- ADT_MATCH** successful verification, or
- ADT_REJECT** failed verification.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_BAD_PARAMETERS	Process Error
ADT_RET_INVALID_TEMPLATE	Process Error

Remarks

The module can control the level of security required for the application through the FARRequest parameter. The False Acceptance Rate (FAR) is the probability that a false sample may be verified as identical to the reference data. The FAR value is determined by the formula:

$$\text{FAR} = \text{FARRequest} / (2^{31} - 1).$$

For instance, a FARRequest value of 2147836 yields a FAR value close to 10^3 .

4.2.6 AC_ADT_FreeTemplate

This function frees up the resources previously allocated by a successful call to a fingerprint image capture.

```
AC__STATUS AC_ADT_FreeTemplate(  
    AC__UCHAR** ResourceTemplate  
)
```

Parameters

ResourceTemplate

The pointer to the template resources that was previously allocated. The value may be **NULL**.

Return Values

Return Value	Meaning
ADT_RET_OK	Success

Remarks

None

See Also

AC_ADT_Process

4.2.7 AC_ADT_CompressTemplate

This function compresses 512-byte minutiae template obtained from other SAGEM products, such as MorphoKit and MorphoTouch, in order for said template to be made compatible with this module.

```
AC__STATUS AC_ADT_CompressTemplate (
    AC__UCHAR*      ExternalTemplate1,
    AC__UCHAR*      ExternalTemplate2,
    AC__UCHAR**     CompressOutputTemplate,
    AC__ULONG*      CompressOutputTemplateSize
)
```

Parameters

ExternalTemplate1

The pointer to 512-byte template of the first finger.

ExternalTemplate2

The pointer to 512-byte template of the second finger. This value can be **NULL**.

CompressOutputTemplate

The pointer to output template data.

CompressOutputTemplateSize

The size in bytes of the output template. This value can be **NULL**.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_BAD_PARAMETERS	Process Error

Remarks

None

4.2.8 AC_ADT_SetGUICallbacks

This function registers the callback routines so that the application may control the "look and feel" of the module.

```
AC__STATUS AC_ADT_SetGUICallbacks (  
    AC_ADT_GUI_CALLBACK    GUICallbackFunction  
)
```

Parameters

GUICallbackFunction

The structure that contains the address of the callback procedure. The address can be **NULL** if the user does not need to use the callback procedure.

Return Values

Return Value	Meaning
ADT_RET_OK	Success

Remarks

None

4.2.9 AC_ADT_GetNVMSize

This function returns the number of bytes available in the non-volatile memory of the device.

```
C__STATUS AC_ADT_GetNVMSize(  
    AC__INT* NVMSize  
)
```

Parameters

NVMSize

The size in bytes of the non-volatile memory.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_FLASH_NOT_SUPPORTEDED	Device Error

Remarks

The ADT60 device has an embedded 1024-byte non-volatile memory that can be accessed by specific functions in the module.

See Also

AC_ADT_ReadNVM, AC_ADT_WriteNVM

4.2.10 AC_ADT_ReadNVM

This function reads data from the non-volatile memory of the device.

```

AC__STATUS AC_ADT_ReadNVM(
    AC__UCHAR*      OutputNVMBuffer,
    AC__INT         NVMStartByte,
    AC__INT         NVMReadSize
)

```

Parameters

OutputNVMBuffer

The pointer to the buffer that will receive the data read from the non-volatile memory.

NVMStartByte

The address in the non-volatile memory of the first byte of the data to be read.

NVMReadSize

The size in bytes of the data to be read from the non-volatile memory.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_BAD_PARAMETERS	Process Error
ADT_RET_FLASH_NOT_SUPPORTED	Device error
ADT_RET_CORRUPTED_VARIABLE	Process Error

Remarks

None

See Also

AC_ADT_GetNVMSize, AC_ADT_WriteNVM

4.2.11 AC_ADT_WriteNVM

This function writes data into the non-volatile memory of the device.

```

AC__STATUS AC_ADT_WriteNVM(
    AC__UCHAR*      InputNVMBuffer,
    AC__INT         NVMStartByte,
    AC__INT         NVMWriteSize
)

```

Parameters

InputNVMBuffer

The pointer to the buffer that will receive the data read from the non-volatile memory.

NVMStartByte

The address in the non-volatile memory of the first byte of the data to be read.

NVMWriteSize

The size in bytes of the data to be read from the non-volatile memory.

Return Values

Return Value	Meaning
ADT_RET_OK	Success
ADT_RET_ERR	Undefined error
ADT_RET_NO_INIT	Process Error
ADT_RET_BAD_PARAMETERS	Process Error
ADT_RET_FLASH_NOT_SUPPORTED	Device error
ADT_RET_CORRUPTED_VARIABLE	Process Error

Remarks

None

See Also

AC_ADT_GetNVMSize, AC_ADT_ReadNVM

Copyright

Copyright by Advanced Card Systems Ltd. (ACS) No part of this reference manual may be reproduced or transmitted in any form without the expressed, written permission of ACS.

Notice

Due to rapid change in technology, some of specifications mentioned in this publication are subject to change without prior notice. Information furnished is believed to be accurate and reliable. ACS assumes no responsibility for any errors or omissions, which may appear in this document.