APPLICATION NOTE

Atmel AT02410: Smartcard Stack on SAM3S-EK

32-bit Atmel Microcontroller

Features

- Atmel® AT91SAM3S8C Microcontroller
  - Two ISO7816 interfaces
- ISO7816 Compatile Smart Card Stack Implementation
  - Answer-to-Reset
  - Information exchange
  - T=0 or T=1 protocols data exchange
- SIM Card Demo
  - Smart Card Stack user guide
  - Extension usage of Smart Card Stack

Introduction

This application note helps users to get familiar with the Smart Card hardware interfaces and software stack on the Atmel SAM3S. By running the demo application, users could quickly understand the usage of the Smart Card Stack. The SIM card which is compatible with GSM11.11 is used in the demo applications and it is a good starting point to be extended to customer's own applications.

The Smart Card Stack can also be easily ported to other Atmel SAM3/SAM4 MCUs.

Table of Contents

1. Smart Card Interface Description

1.1 Introduction

The Univeral Synchronous Receiver Transmitter (USART) on the Atmel SAM3S features an ISO7816-compatible operation mode. This mode permits interfacing with Smart Card and Security Access Module (SAM) communicating through an ISO7816 link. Both T=0 and T=1 protocols defined by the ISO7816 specification can be supported.

When operating in ISO7816 mode, either in T=0 or T=1 mode, the character format is fixed:

- 8 data bits
- 1 or 2 stop bits
- Even parity

For more details about USART in ISO7816 mode, please refer to the Atmel SAM3S datasheet.

1.2 I/O lines

Table 1-1 defines pins used for ISO7816 interface on SAM3S.

Table 1-1. Smart card pins.

SAM3S pin	Smart card pin	Function
USART_CLK	CLK	Smart card clock
USART_TXD	I/O	I/O data
GPIO	RST	Card reset
VDD/GPIO	Vcc	Power supply
GPIO	Vpp	Programming voltage

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory.

Both USART0 and USART1 can be configured as ISO7816 mode.

1.3 Clock generator

The USART is not continuously clocked. The programmer must first enable the USART clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

The ISO7816 specification defines the bit rate with the following formula:

$$B = (Di/Fi) \times f$$

Where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency [Hz]

The possible values for Di, Fi and Fi/Di radio can be found in the Atmel SAM3S datasheet.

Note: The clock selected in the Mode Register (US_MR) is first divided by the value programmed in the field Clock Divider (CD) in the Baud Rate Generator Register (US_BRGR). The resulting clock provides the smart card clock inputs.

2. ISO7816 Protocol Overview

ISO/IEC 7816 is an international standard related to electronic identification cards with contacts, especially Smart Cards, managed jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

The high level information of some of the ISO7816 standards include following:

- 7816-1: Physical characteristics
- 7816-2: Cards with contacts — Dimensions and location of the contacts
- 7816-3: Cards with contacts — Electrical interface and transmission protocols
- 7816-4: Organization, security and commands for interchange

ISO/IEC 7816-1 defines the physical characteristics of the Smart Card.

ISO/IEC7816-2 defines eight contacts for the Smart Card which six of them are usually used for communication.

We will give some background of ISO/IEC7816-3 and ISO/IEC7816-4 which are related to Smart Card Stack in the following sections. For detail of ISO/IEC7816-3 and ISO/IEC7816-4 standards, please refer to ISO7816 reference documents [3] and [4] in Chapter 5.

2.1 Smart card operations principles

The Smart Card operations begin from the mechanical insertion of the Smart Card. The interactions between the interface device and the Smart Card shall be conducted through the following operations:

- Smart Card device configuration
  - Detection of the Smart Card insertion to the interface device (optional)
  - Warm reset of the smart card by the Interface device
  - Answer-to-Reset (ATR) response by the Smart Card to the interface device
  - Protocol and Parameter Selection (PPS) exchange
- Information exchange
  - Execution of the commands between the Smart Card and the Interface device

2.2 Smart card device configuration

The ISO/IEC 7816-3 specifies electrical interface and transmission protocols for asynchronous cards, which includes the ATR mechanism. ATR establishes a basic communication channel between the Smart Card and the reader. The Protocol and PPS is used for setting the parameters responded by ATR.

2.2.1 Answer-to-Reset (ATR)

ATR is a series of characters responded to by the card after the successful reset operation. ATR characters determine the initial communication parameters, bit timing, and the data transfer details between the card and the interfacing device.

The ATR message, which less than 33 characters, consists of these fields:

- Initial byte (TS) (mandatory)
- Format byte (T0) (mandatory)
- Interface bytes (TAi,TBi,TCi and TDi) (optional)
- Historical bytes (T1,T2,...TK) (optional)
- Check bytes (TCK) (conditional)

The TS is used to establish the bit-signaling and bit-ordering conventions between the interface device and the card. TS have two possible patterns which should both be supported by the interface device.

The T0 byte is used to signal the presence or absence of subsequent interface bytes or Historical bytes. Interface bytes are used to tailor the characteristics of the I/O channel, including the specific protocol used by the card and interface device during subsequent exchange of commands and responses.

The Historical bytes, if present, are used to convey manufacturer information. Actually, there is no established standard for the information presented in the Historical bytes.

2.2.2 PPS request and response

The PPS request and response consists of an initial byte, PPSS, followed by a format byte PPS0, three optional parameter bytes, PPS1, PPS2, PPS3, and a check byte PCK as the last byte.

- PPSS identifies the PPS request or response and is set to 0xFF
- In PPS0 byte, bit 8 is reserved for future use and set to 0. Each bit 5, bit 6, or bit 7 set to '1' indicate the presence of an optional byte, PPS1, PPS2, PPS3, respectively. Bit 4 to bit 1 encode the protocol type T to propose a transmission protocol. (i.e., T = 0 or T = 1 or T = 2, etc.)
- PPS1 is encoded in the same way as in TA1 (Interface byte)
- PPS2 is encoded in the same way as the first TB byte for T = 15
- PPS3 is reserved for future use
- Exclusive-ORing all the bytes PPSS to PCK inclusive shall result as 0x00, any other value is invalid

In the most common cases, the PPS response is identical to the PPS request.

For more information on PPS exchange, refer to ISO7816 reference documents [2], [3], and [4] in Chapter 5.

2.3 Smart card data exchange

After the ATR and PPS exchange between the card and the interfacing device, the next step is to execute the command(s) between the card and the interfacing device.

Currently, there are two protocols, which are widely used for Smart Card communication:

- T = 0 (asynchronous half-duplex character transmission protocol)
- T = 1 (asynchronous half-duplex block transmission protocol)

2.3.1 Application protocol data unit

The Smart Card performs the requested operations and communicates the result as a response from the Smart Card. This command response message pair is known as an Application Protocol Data Unit (APDU).

Table 2-1. Command APDU structure.

Common header				Common body		
CLA	INS	P1	P2	[Lc Field]	[Data Field]	[Le Field]

Table 2-1 defines the structure of command APDU. APDU consists of a command header and a command-body. The command header includes CLA, INS, P1 and P2 fields. As in the T=0 protocol, CLA indicates the class of the command. INS indicates the command to process. P1 and P2 are the parameter bytes and indicate controls and options for processing the command. The body of the APDU can vary in size and is used to transmit data to the card's APDU processor as part of a command or to convey a response from the card to the interface device. The Lc field specifies the number of bytes to be transmitted to the card as part of the instruction, i.e., the length of the data field. The data field contains information that must be sent to the card to allow its APDU processor to execute the command specified in the APDU. The Le field specifies the number of bytes that will be returned to the reader in the response APDU.

The body of the APDU can take four different forms:

- Case 1: No data is transferred to or from the card, so the APDU only contains the header
- Case 2: No data is transferred to the card, but data is returned from the card. The body of the APDU only contains a non-null Le field
- Case 3: Data is transferred to the card, but none is returned from it. The body of the APDU includes the Lc and data fields
- Case 4: Data is transferred to the card and is also returned from the card as a result of the command. The body of the APDU includes the Lc, data and Le fields

Table 2-2. Response APDU structure.

Body	Trailer	
Data field	SW1	SW2

Table 2-2 defines the response APDU structure. It consists of a body and a trailer. The body is either null or it includes a data field – depending on the specific command. The length of the data field is determined by the Le field in the corresponding command APDU. The trailer consists of up to two fields of status bytes called SW1 and SW2. These fields return a status code in which one byte is used to specify an error category and the other is used to specify a command-specific status or error indication.

2.3.2 T=0 protocol

The T=0 protocol is a byte-oriented protocol where a character is transmitted across the channel between the interface device and the card.

Error handling is performed on each byte by looking at the parity bit. If the actual parity bit does not correspond to the parity of the transmitted data, an error must have occurred. In the T=0 protocol, the receiving side signals require the byte to be retransmitted in case of detecting a parity error. This is done by holding the I/O line low (normally the I/O line is set high before the transfer of a byte). When the transmitting side detects the I/O line low, it resends the byte that was not received by the receiving side correctly.

The APDU for T=0 protocol consists of two distinct structures:

- A command sent from the interface device to the card
- A response sent from the card to the interface device

The command header includes the following five fields; the length of each field is one byte:

- CLA: class designation of the command set to establish a collection of instructions
- INS: specifies a specific instruction from within the set of instructions
- P1: used to specify the addressing used by the [CLA, INS] instruction
- P2: also used to specify the addressing used by the [CLA, INS] instruction
- P3: specifies the number of data bytes transferred to or from the card as part of the [CLA, INS] instruction execution

Each value of CLA defines an application-specific set of instructions.

2.3.3 T=1 protocol

The T=1 protocol is a block-oriented protocol where a block is transmitted across the channel between the interface device and the card. A block is a byte string conveyed in asynchronous characters.

The T=1 transmission protocol defines three types of blocks.

- An information block (I-block) is used to convey information for use by the application layer. In addition, it conveys a positive or negative acknowledgment

- A receive ready block (R-block) is used to convey a positive or negative acknowledgment. Its information field shall be absent
- A supervisory block (S-block) is used to exchange control information between the interface device and the card. Its information field may be present depending on its controlling function

Table 2-3. Block frame structure.

Prologue field			Information field	Epilogue filed
Node address	Protocol control byte	Data length	Optional (INF)	Error detection (EDC)
1 byte	1 byte	1 byte	0 ~ 254 Bytes	1 or 2 Bytes

Table 2-3 defines the block frame structure. The three fields involved in the block frames are:

- Prologue field
- Information field
- Epilogue field

More information about the block frame structure can be referred to ISO7816 reference documents [2], [3], and [4] in Chapter 5.

3. Smart Card Stack Overview

The user could access a Smart Card by directly using the application layer. It allows initializing the Smart Card interface and doing the data exchanges between the card and the Atmel SAM3S device.

Figure 3-1. Smart card stack on SAM3S.

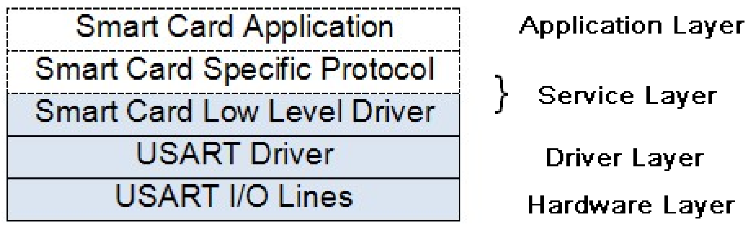


Figure 3-1 presents the Smart Card Stack layers on SAM3S. The Smart Card Stack implemented includes Smart Card low level driver and the USART driver. The SIM card demo described in the next section will show a sample Smart Card application with the specific GSM11.11 protocol.

3.1 File organization

Table 3-1 presents the modules of Smart Card Stack:

Table 3-1. Smart card stack.

Stack layer	File	Description
Smart card low level layer	sam\services\smart_card\Iso7816.c	Smart Card interfaces
	sam\services\smart_card\Iso7816.h	Smart Card definitions, function prototypes
USART driver	sam\drivers\usart\usart.c	USART driver implementation
	sam\drivers\usart\usart.h	USART function prototypes

3.2 Fuinction interfaces

Table 3-2 lists the major functions of the Smart Card Stack.

Table 3-2. Smart card stack application interfaces.

Function name	Description
iso7816_init	Smart Card interfaces initialize
iso7816_cold_reset	Smart Card interfaces cold reset
iso7816_warm_reset	Smart Card interfaces warm reset
iso7816_data_block_atr	Answer to reset handler
iso7816_decode_atr	Decode ATR
iso7816_xfr_block_apdu_t0	Transfer Block APDU with protocol T = 0
iso7816_xfr_block_apdu_t1	Transfer Block APDU with protocol T = 1

The sam_iso7816_opt_t structure is defined in the usart.h file:

```
typedef struct {
uint32_t iso7816_hz;
uint32_t fidi_ratio;
```

uint32_t parity_type;

uint32_t inhibit_nack;

uint32_t dis_suc_nack;

uint32_t max_iterations;

uint32_t bit_order;

uint32_t protocol_type;

} usart_iso7816_opt_t;

Sturcture items:

- Iso7816_hz: Set the frequency of the ISO7816 clock
- Fidi_radio: The number of ISO7816 clock ticks in every bit period (1 to 2047, 0 = disable clock). Baudrate rate = iso7816_hz / fidi_ratio
- parity_type: How to calculate the parity bit: US_MR_PAR_EVEN for normal mode or US_MR_PAR_ODD for inverse mode
- inhibit_nack: Inhibit Non Acknowledge, This bit will be used only in ISO7816 mode, protocol T = 0

  0: the NACK is generated

  1: the NACK is not generated
- dis_suc_nack: Disable successive NACKs. 0: NACK is sent on the ISO line as soon as a parity error occurs in the received character. Successive parity errors are counted up to the value in the max_iterations field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The ITERATION flag is asserted
- max_iterations: Bit order in transmitted characters:

  0: LSB first

  1: MSB first
- protocol_type: Which protocol is used:

  0: T = 0

  1: T = 1

Function prototype:

void iso7816_init(Usart p_usart, const usart_iso7816_opt_t *p_usart_opt,*

uint32_t ul_mck, uint32_t ul_rst_pin_idx)

Function description:

- Initializes an ISO7816 interface device

Parameters:

• p_usart	Pointer to the USART device
• p_usart_opt	Pointer to an ISO7816 instance
• ul_mck	USART module input clock frequency
• ul_rst_pin_idx	Control Smart Card RST pin index

Function prototype:

void iso7816_cold_reset(Usart p_usart)*

Function description:

- This function performs a cold reset to the ISO7816 interface device

Parameters:

p_usart	Pointer to the USART device

Function prototype:

void iso7816_warm_reset(Usart p_usart)*

Function description:

- This function performs a warm reset to the ISO7816 interface device

Parameters:

p_usart	Pointer to the USART device

Function prototype:

void iso7816_data_block_atr(Usart p_usart, uint8_t *p_atr, uint8_t *p_length)*

Function description:

- This function gets the channel information from Answer-to-Reset (ATR)

Parameters:

p_usart	Pointer to the USART device
p_atr	Pointer to ATR buffer
p_length	Pointer for store the ATR length

Function prototype:

uint8_t iso7816_decode_atr(Usart p_usart, uint8_t *p_atr)*

Function description:

- This function decodes the ATR and set the parameters gotten from ATR

Parameters:

p_usart	Pointer to the USART device
p_atr	Pointer to ATR buffer

Function prototype:

uint16_t iso7816_xfr_block_apdu_t0(Usart p_usart, const uint8_t *p_apdu,*
*uint8_t *p_message, uint16_t us_length)*

Function description:

- Transfer Block APDU with protocol T = 0

Parameters:

p_usart	Pointer to the USART device
p_apdu	Pointer to APDU buffer
p_message	Pointer to message buffer
us_length	Block length

Function prototype:

uint16_t iso7816_xfr_block_apdu_t1(Usart p_usart, const uint8_t *p_apdu,*

*uint8_t *p_message, uint16_t us_length)*

Function description:

- Transfer Block APDU with protocol T = 1

Parameters:

p_usart	Pointer to the USART device
p_apdu	Pointer to APDU buffer
p_message	Pointer to message buffer
us_length	Block length

3.3 Smart card stack extension

Based on the Smart Card Stack on the Atmel SAM3S, users can easily extend to specific Smart Card data access protocols.

Take the protocol of Smart Card Operating System for Transport Applications (SCOSTA) for example; the only stack consideration for the developer is encapsulating the Smart Card specific transmission commands. Supposing the Smart Card is using T=0 protocol, the smartcard_read () and smartcard_write () functions can be simpily implemented by using the function of iso7816_xfr_block_apdu_t0 () with the parameter fields of [CLA, INS] in Table 2-1 on page 6 of p_apdu filled with the read/write commands defined in the individual card specification according to SCOSTA. Then the user can use the smartcard_read () / smartcard_write () functions with specific access address and data by filling the [P1, P2] and command body (see Table 2-1 on page 6) to develop their own Smart Card applications which support SCOSTA.

4. Usage of Smart Card Stack

The SIM card demo is based on the Smart Card stack, it is provided in order to help the user to get familiar with the Smart Card Stack and to use it to develop your own Smart Card application with specific protocols.

The demo provides operations with an ISO 7816-3/4-compatible GSM11.11 Smart Card. Several APDU commands will be used to show how to interact with the connected SIM card. The International Mobile Subscriber Identification Number (IMSI) information will be finally read from the Smart Card.

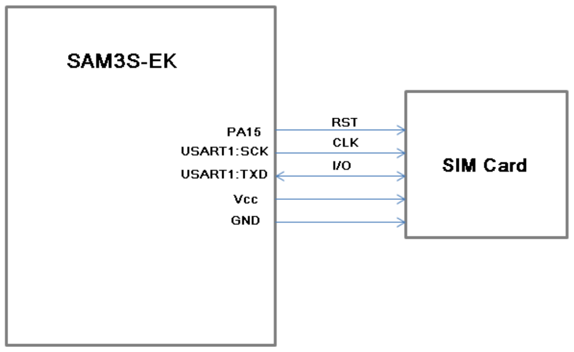
The demo should be used on Atmel Studio 6.0 and IAR Embedded Workbench® 6.40.

4.1 Hardware requirement

This demo runs on the Atmel SAM3S-EK and an SIM card compatible with GSM11.11 is required. The SIM card can be connected to the USART1 of SAM3S-EK either with the direct jumpped wire or through a Smart Card slot.

The SIM card is supposed using T=0 protocol in the demo.

Figure 4-1. SIM demo block diagram.



4.2 SIM card demo overview

The demo will initialize the Smart Card interface on SAM3S-EK. ATR and PPS operations will be performed consequencely. Then the master root and GSM directory will be opened. At the end of the demo, the IMSI file under GSM directory will be read and displayed in the debug console.

Two directories have been defined for the GSM Smart Card directory tree:

MasterRoot[3]={0x3F, 0x00};

GSMDir[3]={0x7F, 0x20};

4.2.1 Smart card initialization

The USART pins need to be configured to ISO7816 mode, the PMC for the USART needs to be enabled. The default baudrate is configured as 9600bps and T=0 protocol is used for transmission.

```
#define ISO7816_BAUDRATE     9600
#define ISO7816_FI_DI        372
conf_iso7816_t. iso7816_hz = ISO7816_BAUDRATE *ISO7816_FI_DI;
conf_iso7816_t. fidi_ratio = ISO7816_FI_DI;
conf_iso7816_t.parity_type= US_MR_PAR_EVEN;
conf_iso7816_t. inhibit_nack = 0x00;
conf_iso7816_t. dis_suc_nack = 0x00;
```

conf_iso7816_t. max_iterations = 0x03;

conf_iso7816_t. bit_order = 0x00;

conf_iso7816_t. protocol_type = 0x00;

iso7816_init(ISO7816_USART1, &conf_iso7816_t, sysclk_get_cpu_hz(), PIN_ISO7816_RST_IDX1);

When a card is detected, the ATR sequence is generated, received and decoded.

iso7816_warm_reset(ISO7816_USART1);

iso7816_data_block_atr(ISO7816_USART1, p_atr, &uc_size);

poc_type = iso7816_decode_atr(ISO7816_USART1, p_atr);

For a typical GSM card, the PPS procedure is as follows:

PTSS = 0xFF

PTS0 = 0x10

PTS1 = 0x94

PCK = 0x7B

Where PTS1 = 0x94 means F = 9 and D = 4, Fi = 512, Di = 8.

If the GSM card is using T=1 protocol, the USART interface device will be reconfigured to T=1.

conf_iso7816_t.protocol_type = 1;

conf_iso7816_t.max_iterations = 0;

usart_init_iso7816(ISO7816_USART1, &conf_iso7816_t,
sysclk_get_cpu_hz());

4.2.2 Get IMSI number

The specified reading path is assumed to be: MasterRoot/GSMDir/IMSI.

To reach it, the following actions are performed:

- Select master file
- Get response of select master file
- Select GSM directory
- Select IMSI
- Get response of select IMSI
- Read binary of IMSI

The APDU commands are pre-defined with data arrays:

- uint8_t sim_select_mf[] = { 0xa0, 0xa4, 0x0, 0x0, 0x2, 0x3f, 0x0};
- uint8_t sim_get_resp_mf[] = { 0xa0, 0xc0, 0x0, 0x0, 0x0 };
- uint8_t sim_select_gsm_dir[] = { 0xa0, 0xa4, 0x0, 0x0, 0x2, 0x7f, 0x20};
- uint8_t sim_select_imsi[] = { 0xa0, 0xa4, 0x0, 0x0, 0x2, 0x6f, 0x7};
- uint8_t sim_get_resp_imsi[] = {0xa0, 0xc0, 0x0, 0x0, 0x0};
- uint8_t sim_read_binary_imsi[] = {0xa0, 0xb0, 0x0, 0x0, 0x9};

Take select MF for example, the data in the sim_select_mf[] corresponds

- CLA = 0xa0 // class of GSM11.11
- INS = 0xa4 // Select file defined by GSM11.11
- P1 = 0 // P1 set to 0
- P2 = 0 //P2 set 0
- Lc = 2 // Data field length, as MF number is 0x3f 0x00
- Data Field = 0x3f 0x00 //MF number

If succeeded executed, a typical APDU respond would be

- 0x9F // SW1, succeeded responds
- 0xXX //SW2, MF information

The other APDU command execution flows are similar to select MF.

Finally the IMSI number will be displayed to the debug console:

- -I- Get IMSI Answer: 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0xXX 0x90 0x00

0x90 0x00 indicates a success execution of the read binary command.

5. References

[1]. Atmel SAM3S datasheet, doc6500

[2]. ISO Organization, ISO/IEC 7816-2:2007 Identification cards — Integrated circuit cards — Part 2: Cards with contacts — Dimensions and location of the contacts

[3]. ISO Organization, ISO/IEC 7816-3:2006 Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols

[4]. ISO Organization, ISO/IEC 7816-4:2005 Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange

6. Revision History

Doc. rev.	Date	Comments
32203B	05/2013	Part of the title is changed from “AT2410” to “AT02410”
32203A	02/2013	Initial document release

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370