Interoperability Specification for ICCs and Personal Computer Systems

Part 4. IFD Design Considerations and Reference Design Information

Bull CP8, a Bull Company

Gemplus SA

Hewlett-Packard Company

IBM Corporation

Microsoft Corporation

Schlumberger SA

Siemens Nixdorf Informationssysteme AG

Sun MicroSystems Inc.

Toshiba Corporation

VeriFone Inc.

Version 1.0-
December 1997

Contents

1. SCOPE **1**

1.1 General Purpose of this Document **1**

1.2 Scope of Information Found in this Document **2**

1. **2**

2. PS/2 KEYBOARD–INTEGRATED IFD **3**

2.1 Rationale **3**

2.2 Design Assumptions **3**

2.3 PS/2 Keyboard Communication Interface **4**

2.4 Keyboard Electronics **5**

2.5 **5**

Incremental IFD Requirements **6**
2.5.1 Card Socket 6
2.5.2 ICC Interface 6
2.5.3 6
Power 7

2.6 Keyboard/IFD-to-PC Interface **7**
2.6.1 IFD Command Set Summary 7
2.6.2 PC to Keyboard Extended Commands 7
2.6.3 10
Keyboard-to-PC Extended Commands 11
2.6.4 IFD Status Register 12
2.6.5 IFD Error Register 13
2.6.6 Security Assurance Feature Register 13
2.6.7 Requirements for Multiplexed Operation 14

2.7 **14**

Security Assurance Features **15**

3. **15**

USB ICC IFDS **16**

3.1 Rationale **16**

3.2 Device Configuration **16**

3.3 **17**

USB IFD Functionality **18**

1. Scope

1.1 General Purpose of this Document

This document discusses design information for IFD devices compliant with the *Interoperability Specification for ICCs and Personal Computer Systems*.

An ICC Reader Device (IFD), as depicted in Figure 1.1, is a hardware device that :

- Provides an interface with the ICC, as described in Part 2 of this specification.
- Communicates with the PC-Hosted IFD Handler software (described in Part 3 of this specification), through a given I/O channel.

The purpose of this document is to present design information on several types of IFDs, and especially design information related to the I/O channel used by those IFDs to communicate with the PC.

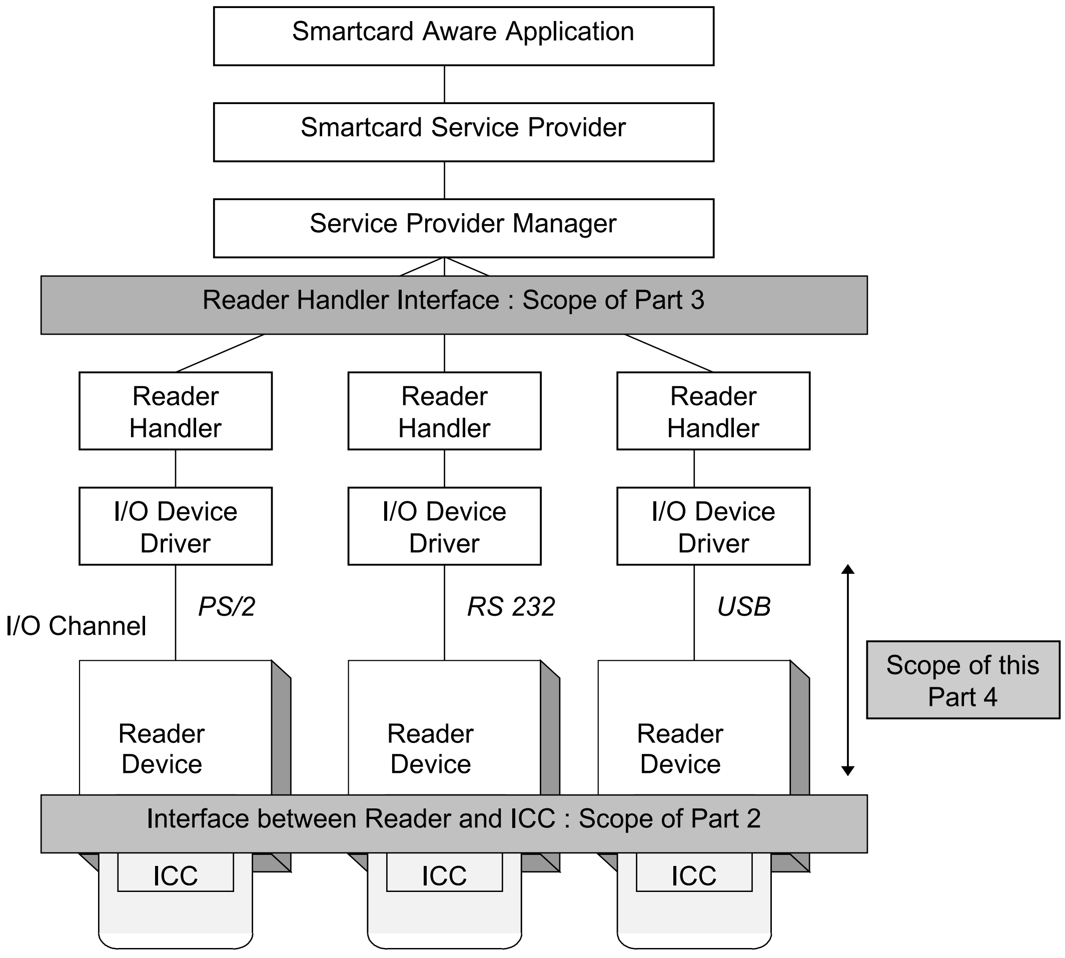

Figure 1-1. IFD Subsytem components and interfaces with other components

Four types of IFDs will be mentionned in this document :

- Integrated PS/2 keyboard IFDs, which the authors of these specifications strongly believe will be widely adopted as PC-connected IFDs
- USB IFDs
- RS-232-connected IFDs, which are today the most common type of IFDs connected to desktop PC's
- PC Card–based IFDs, targeted to the laptop and notebooks environment

1.2 Scope of Information Found in this Document

As depicted in Figure 1.2, a device (and its counterpart on the PC side) can be layered into three levels.

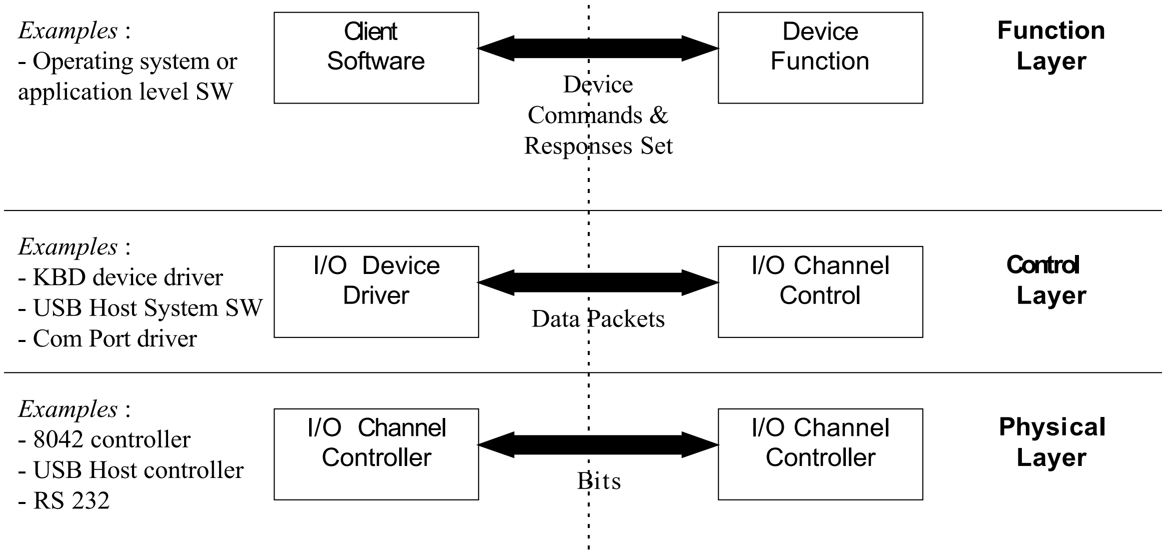

Figure 1-2. Device layers

Among the different IFDs available on the market (that are either serial or PC Card IFD's) as of today, there does not exist a standard set of commands for IFDs, that is, there is no standardization at the Function layer. No reference design will be presented neither for serial nor PC Card readers; vendors providing those types of readers should provide interoperability at the IFD handler level (this topic is discussed in Part 3 of these specifications).

The situation is different for the PS/2 keyboard: information related to the three levels is presented, especially a complete command set that can provide a standard at the Function layer, for PS/2 keyboard based IFDs.

Finally, for USB IFDs, the rationale for building USB IFDs and the basic design assumptions are exposed. All requirements at the Control and Physical layers can be found in the USB specifications.

1.

2. PS/2 Keyboard–Integrated IFD

This section provides design information for vendors wishing to build a compliant IFD integrated with an industry-standard PS/2 PC keyboard. It is believed this design will become the preferred approach to deploying IFDs for desktop PCs. The PS/2-compatible keyboard is the most widely deployed keyboard on PC/workstation-class machines and is supported by multiple Operating Systems. In addition, a PS/2-based design limits the amount of desktop space required to deploy such IFDs and places them in a position that is readily visible and accessible for the user. This information is included herein to encourage multiple vendors to produce compliant products that will work with common device drivers.

The design information provided addresses only the components and I/O interfaces required to build a PS/2 keyboard–integrated IFD. That is, it covers design from the keyboard connector on a PC to the ICC socket on the keyboard, inclusive.

2.1 Rationale

An obvious place to put an IFD is in the PC keyboard. It is a convenient location, being situated for easy access by the PC user, and acting as a primary interface to the PC. In particular, it tends to be directly in front of the user, making ICC insertion/removal simple and keeping the ICC in view of the user. In addition, it is a large enough device to allow incorporation of an IFD without major redesign. Furthermore, by using the keyboard microprocessor to control the IFD, the incremental cost can be very low.

Three different keyboard interface specifications exist for PCs compatible with IBM hardware specifications. The older PC/XT keyboard is obsolete, unidirectional, and will not be discussed here. The PS/2 keyboard is the current standard, and is bidirectional. It is likely however that Universal Serial Bus (USB)–based keyboards will be dominant, in the future.

2.2 Design Assumptions

The design presented herein is appropriate for IFDs that take maximum advantage of existing PS/2 keyboard components to minimize device cost. In particular, it is assumed that the design will take advantage of the existing keyboard microprocessor, clock circuits, power, and so on. As such, it is appropriate to limit the sophistication and flexibility of the IFD. The following table summarizes expected functionality and how this is implemented between the IFD and IFD Handler. The Keyboard device driver is assumed to act as a part of the transmission path, but to provide no intelligence related to the IFD.

Table 2-1. Functionality Assignment for PS/2-Integrated IFD

Function	IFD	IFD Handler
IFD activation	Upon initialization request from IFD Handler.	Must request device activation/determine device presence, following power-on sequence.
ICC insertion detection	Detect ICC insertion/removal and generate an event.	Field event and pass to active Service Provider, if any.
ICC contact management	Automatically activate ICC contacts on card insertion; de-activate upon removal. May optionally support ability to deactivate ICC contacts under PC control.	No direct requirements, unless IFD implements deactivation of ICC contacts while card inserted.
ICC initialization	Must perform cold reset following ICC contact activation, read ATR sequence if present, and send ATR to the PC. If ATR response time is exceeded, shall mark card type as unknown.	May request warm reset of ICC. Should process ATR, validate and determine whether T=0 or T=1 protocol is to be used.
Protocol support	Implements physical layer, including T=0 mandatory handling of bit-parity error processing. IFD is expected to implement only default CLK frequency and timing parameters.	Implements T=0 and T=1 data link layer processing and associated error handling.
I/O line management	Following reset and ATR sequence, assumes transmission control is determined by the IFD Handler. Will transmit to the ICC when a block of data is received from the IFD Handler. It then returns bytes from the ICC to the Handler as received until the next Handler data block is received.	Must send ICC data to the device only when it has the right to transmit. At other times, it should wait for ICC response.
Security Assurance and Authentication Devices	Implementation optional.	Implementation optional.

2.3 PS/2 Keyboard Communication Interface

The PS/2 keyboard communicates with a PC over a 5 wire I/O port. The interface supports:

- Clock
- Data
- Spare
- Gnd
- +5V

The channel is bidirectional, half duplex. Arbitration is handled by holding the clock and data lines in various states. There are 11 clocks per “character”: start bit (low), eight data bits, parity bit, stop bit (high) with a nominal data rate of 10,000 cps.

The following commands are presently defined for PC communication with the PS/2 keyboard.

Table 2-2. PC-to-Keyboard Command Set

Command name	Command code	Description
RESET	FF	Keyboard reboots itself.
RESEND	FE	Keyboard resends last byte.
SCAN SET 3	F7-FD	Reserved for SCAN SET 3.
SET DEFAULT	F6	Keyboard resets to power-on state.
DEFAULT DISABLE	F5	Keyboard resets to power-on state, no key scanning.
ENABLE	F4	Keyboard resumes key scanning.
SET TYPEMATIC RATE	F3	Keyboard sends ACK ACK, and sets rate.
ID BYTE REQUEST	F2	Keyboard sends ACK, plus ID (*xx*,*yy*)
INVALID	F1	Keyboard responds with RESEND (fe).
SET SCAN SET	F0	Keyboard responds with ACK.
INVALID	EF	Keyboard responds with RESEND (fe).
ECHO	EE	Keyboard sends ECHO back (ee).
INDICATOR CONTROL	ED	Keyboard sends ACK ACK, sets LEDs.

The following commands may be sent by a PS/2 keyboard to the PC.

Table 2-3. Keyboard-to-PC Command Set

Command name	Command code	Description
RESEND	FE	PC resends last byte.
ACK	FA	Acks all except ECHO and RESEND.
OVERRUN	FF for ScanSet 1 00 for ScanSet 2 and 3	Marks fifo as full.
DIAGNOSTIC FAILURE	FC	Error during bat, or other failure.
BREAK CODE PREFIX	F0	Prefix for key release code.
DIAG. COMPLETION	AA	Diags completed successfully.
ECHO RESPONSE	EE	Sent in response to ECHO command.

Note that the PS/2 data exchange protocol is an asymmetrical one. It requires that every byte sent by the PC to the keyboard be acknowledged by the keyboard, by sending an ACK, whereas data sent from the keyboard to the PC is not explicitly acknowledged.

2.4 Keyboard Electronics

A typical keyboard has very simple electronics, a single voltage power line, buffers for the signal and clock lines to the PC, drivers for the LED's, and a microprocessor. The microprocessor handles the communication protocol, scans the key switches, and controls the LEDs.

2.5

Incremental IFD Requirements

The addition of an ICC IFD compliant with this specification will require the following components for a minimal implementation:

- ICC socket
- Interface lines between the ICC socket and the keyboard microprocessor/supporting circuitry:
  - RST
  - CLK
  - I/O
  - Card detection
- ICC power
  - VCC
  - GND
- Optional I/O line(s) for ICC status LED (s), alphanumeric displays, and so on
- Buffers for signals
- ROM code to implement the required IFD logic

2.5.1 Card Socket

ICC sockets are available in a wide range of form factors from simple manual insertion devices to sophisticated motor driven devices (socket pulls it in, software control ejects it). Given that compliant ICC IFDs are likely to be end-user devices operated in an office environment, it is believed that the simplicity and low cost associated with a manual socket will be preferred. This is, however, a vendor decision. In any event, it is strongly recommended that the vendor implement a Landing Card or Landing Contact socket, as opposed to a wiping contact socket, for long life and to minimize ICC damage. A microswitch is recommended over a blade switch for card present detection.

2.5.2 ICC Interface

The keyboard microprocessor will require the following I/O lines, in addition to the ones already required for normal keyboard operations (Host I/O, key scanning, LED's, and others):

- ICC Reset (RST).
- ICC Clock (CLK).
- ICC I/O.
- ICC insertion switch.
- Card Status LEDs (optional). These would logically be used to indicate status such as proper card insertion; contact activation status, and requests for user authentication input (if such devices are supported).

PS/2 keyboard IFDs are expected to use the existing keyboard controller clock to generate the CLK signal to the ICC. As such, it is expected they will support only a default CLK frequency (in the range 1–5 MHz). It is also expected they will support only the default clock conversion factor (F) of 372 and default bit rate conversion factor (D) of 1, though a vendor may support alternate protocol parameters. Note that if an IFD does implement protocol and/or parameter selection, it must also implement the PTS protocol.

2.5.3

Power

Most existing ICCs are +5.0-volt devices that draw less than 50 mA current (10 mA typical). This is well within the source capabilities of the +5V keyboard power supply provided by existing PCs. An external power supply should not be required.

It is expected that lower-power ICC devices ($\leq$ 3.3 V at < 10 mA) will be common. At present, there is no standard for determining the proper voltage for operation of ISO/IEC 7816 ICCs. ISO/IEC 7816-3 is currently being revised to address this issue. It has been proposed that two classes of cards be defined: (5 V - Class A and 3 V - Class B) IFDs will be designated as Class A, Class B, or Class AB. When the standard for voltage determination is established, support for < 5V ICC operation will be a required addition to this specification.

2.6 Keyboard/IFD-to-PC Interface

When a PS/2 Keyboard with integrated IFD first powers up, it should appear to the PC as a normal PS/2 Keyboard. It is the responsibility of PC device drivers to initialize the IFD section of the device. Until this initialization occurs, insertion of an ICC into the IFD will not result in any signals/data being sent to the PC. The requirement on the IFD for managing the interface to the ICC in this situation. is that the vendor must leave the ICC contacts in the inactive state. Once the device driver has initialized the IFD, it must follow the activation sequence of the ICC in accordance with the operational requirements defined in this specification.

2.6.1 IFD Command Set Summary

To support communication between the PC and the IFD, an extended command set is defined. These commands are exchanged between the PC and PS/2 keyboard controller using the existing command interface. The commands, as defined below, have some built-in redundancy to provide interface flexibility.

2.6.2 PC to Keyboard Extended Commands

Table 2-4. Extended (IFD) PC to Keyboard Command Set

Command name	Command code	Description
PC_to_RDR_GetRdrType	0x60	Gets IFD type.
PC_to_RDR_SetMode	0x61	Sets mode.
PC_to_RDR_CardPowerOn	0x62	Turns on power to the card socket.
PC_to_RDR_CardPowerOff	0x63	Turns off power to the card socket.
PC_to_RDR_Reset	0x64	Sends reset pulse to the smart card.
PC_to_RDR_GetRdrStatus	0x65	Gets IFD status.
PC_to_RDR_SendRdrByte	0x66	Sends one byte to IFD.
PC_to_RDR_SendRdrBlock	0x67	Sends data block to IFD.
PC_to_RDR_ResendBlock	0x68	Resends last data block.
PC_to_RDR_GetRdrCaps	0x69	Gets IFD capabilities.
PC_to_RDR_DeActivateRdr	0x6A	Deactivates IFD.
PC_to_RDR_Escape	0x6B	Activates IFD-dependent feature(s).
PC_to_RDR_RFU	0x6C - 0x6F	Reserved for future use.

Table 2-5. Implementation Notes for Extended PC-to-Keyboard Commands

Command name	**Byte sequence**	**Expected response** (* = see Note 2.6.1)	**Notes**
PC_to_RDR_GetRdrType	0x60	RDR_to_PC_Type	The keyboard will activate the IFD following power-up only upon receipt of this command. This should be the first IFD command sent to the keyboard.
PC_to_RDR_SetMode	0x B61, *rr*, *ss*, *tt*, *uu, vv* (cf. Table 2-6 and 2-7)7)	*	Followed by five bytes *rr*, *ss*, *tt*, *uu, vv*. Before this command is sent to the IFD, any command can be sent to the IFD except: • PC_to_RDR_SendRdrByte • PC_to_RDR_SendRdrBlock,
• PC_to_RDR_CardPowerOn	0x62	* followed by a RDR_to_PC_ DataBlock containing the ATR	Power saving mode, and others.
PC_to_RDR_CardPowerOff	0x63	*	Power saving mode, and others.
PC_to_RDR_Reset	0x64	* followed by a RDR_to_PC_ DataBlock containing the ATR	Forces an immediate "warm reset" of the ICC if inserted. If no ICC is present, this is a no-op.
PC_to_RDR_GetRdrStatus	0x65	RDR_to_PC_Status	IFD will send back 4-byte status.
PC_to_RDR_SendRdrByte (See Note 2.6.2)	0x66, *rr*	*	*rr* = data byte. The IFD shall immediately transmit the data byte to the ICC, if present.
PC_to_RDR_SendRdrBlock	0x B67, *rr*, *ss*, DataBlock	* followed by a RDR_to_PC_ DataBlock containing the answer from the ICC	*rr*, *ss* = 16-bit block size. The block size may be optimized by the IFD vendor to minimize potential for blocking for keyboard input, but should never exceed the maximum T=1 block size of 260 bytes. This is followed immediately by the specified number of data bytes, which should be immediately transmitted to the ICC, if present.
PC_to_RDR_ResendBlock	0x68	RDR_to_PC_ DataBlock	IFD will resend last data block.
PC_to_RDR_GetRdrCaps	0x B69	RDR_to_PC_Caps	Get IFD Capabilities as a TLV.
PC_to_RDR_DeActivateRdr	0x6A,	*	Resets the IFD section of a keyboard back to its power-up state.

PC_to_RDR_Escape	0x B6B, *rr*, *ss*, DataBlock (vendor-defined parameters in DataBlock)	* followed by a RDR_to_PC_Escape	*rr*, *ss* = 16-bit block size. The escape function allows the IFD manufacturer to define and access extended features. Information sent via this command is processed by the IFD control logic. This may be used, for example, to request that biometric input data be returned from IFDs so equipped.

Table 2-6. PC_to_RDR_SetMode Parameters

Parameter	ATR parameter(s) encoded	Possible values	Description
rr	T, C, CKS	00, 02, 10, 11, 12, 13 80	Protocol Type (b4=0 for T=0, b4=1 for T=1) Convention used (b1=0 for direct, b1=1 for inverse) Checksum type (b0=0 for LRC, b0=1 for CRC) Special case (See note 2.6.3) Waiting Time extension request (the value of the requested extension is specified by *vv*)
rr	T, C, CKS	00, 02, 10, 11, 12, 13	Protocol Type (b4=0 for T=0, b4=1 for T=1) Convention used (b1=0 for direct, b1=1 for inverse) Checksum type (b0=0 for LRC, b0=1 for CRC)
ss	N	00 to FE	Extra Guardtime (0 to 254 etu between two characters)
tt	WI	00 to FF	Work Waiting Time (character time-out for T=0)
uu	BWI, CWI	00 to FF	Block and Char. Waiting Time (block and character time-out for T=1)
vv	WTX	00 to FF	Block Waiting Time Extension. 00 = no WTX is requested by the ICC. vv is the multiplier of the BWT value

In the event all the parameters are not transmitted in the ATR, the IFD Handler software should use the following default values.

Table 2-7. Default values for the PC_to_RDR_SetMode Parameters

Mode	*rr*	*ss*	*tt*	*uu*	*vv*
T=0	00	00	0A	00	00
T=1	10	00	00	4D	00

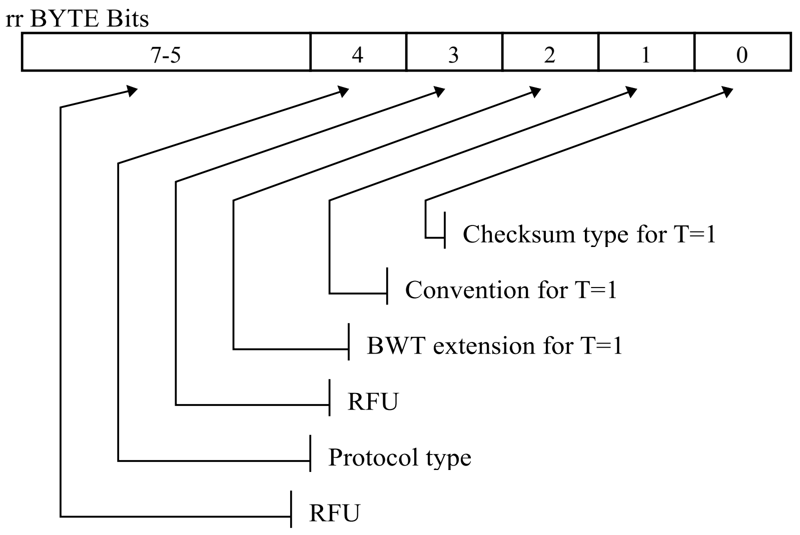

Figure 2-1. rr Byte encoding

Table 2-8. rr Byte encoding

Protocol Type T	0 if T=0 1 if T=1
Block Working Time Extension WTX	1 if WTX is requested
Convention C	0 for direct convention 1 for inverse convention
Checksum CK	0 for LRC 1 for CRC

Note 2.6.1: The PS/2 protocol and, more precisely, its implementation by the various micro-controllers found on "IBM PC compatibles" types of PC's, require each byte sent by the PC be acknowledged by the keyboard. This holds true also for data sent to the ICC. In other words, every single byte of a given command must be acknowledged by the keyboard with a RDR_to_PC_Ack. This is clearly a trade-off that favors broad compatibility with the installed base versus data throughput. However, this should still be sufficient to accommodate the needs of today's smart cards.

Note 2.6.2: A compliant IFD may not implement the PC_to_RDR_SendRdrByte command. If the keyboard consistently Nacks this command, the IFD Handler software layer should use the PC_to_Rdr_SendBlock commands instead.

Note 2.6.3 : The Waiting Time Extension, when requested, applies only to the outstanding transaction. It is automatically reset to 0 once the ICC has sent its response.

2.6.3

Keyboard-to-PC Extended Commands

Table 2-9 Extended (IFD) Keyboard-to-PC Command Set

Command name	Command code	Description
RDR_to_PC_Type	0x60	Sends type info to PC.
RDR_to_PC_Status	0x61	Sends status to PC.
RDR_to_PC_Ack	0x62	ACKS last IFD command from PC.
RDR_to_PC_DataByte	0x63	One byte from IFD to PC.
RDR_to_PC_DataBlock	0x64	Sends a Data Block to PC.
RDR_to_PC_CardIn	0x65	Card inserted.
RDR_to_PC_CardOut	0x66	Card removed.
RDR_to_PC_Nack	0x67	Nacks last command (see Note 2.6.4).
RDR_to_PC_Caps	0x68	Sends TLV structure of capabilities.
RDR_to_PC_Escape	0x69	IFD-dependent feature.
RDR_to_PC_RFU	0x6A - 0x6F	Reserved for future use.

Table 2-10 Implementation Notes for Extended Keyboard–to-PC Commands

Command name	Byte sequence	Notes
RDR_to_PC_Type	0x60, Type Sequence	This returns a 16-byte vendor-defined IFD ID. It is recommended this be generated in a manner that ensures statistical uniqueness. It may be of the type “PNPxxxx”, as used by plug and play keyboards, or may be a vendor-specific ID.
RDR_to_PC_Status	0x61,*s1*, *s2*, *s3*, *s4*	*s1* = IFD status register as defined below. *s2* = Error. *s3* = Security Assurance feature register as defined below. *s4* = RFU.
RDR_to_PC_Ack	0x62	n/a
RDR_to_PC_DataByte	0x63, *rr*	One byte of data from the ICC.
RDR_to_PC_DataBlock	0x64, *rr*, *ss*, DataBlock	Block of data. *Rr*, *ss* = 16-bit block size. The block size may be optimized by the IFD vendor to minimize potential for blocking for keyboard input, but should never exceed the maximum T=0 command size of 260 bytes (5 bytes command; 255 bytes data). This is followed immediately by the specified number of data bytes that should be immediately transmitted to the PC.
RDR_to_PC_CardIn	0x65	Indicates an ICC insertion event has occurred.
RDR_to_PC_CardOut	0x66	Indicates an ICC removal event has occurred.
RDR_to_PC_Nack	0x67	Nacks last command (see Note 2.6.4).
RDR_to_PC_Caps	0x68, rr, ss, DataBlock	rrss = 16 bit block size. The DataBlock is a TLV structure that returns part of the information described in section 3.1.2 of Part 3 of this specification.
RDR_to_PC_Escape	0x69, *rr*, *ss*, DataBlock (vendor-defined parameters	*rr*, *ss* = 16-bit block size. The escape function allows the IFD manufacturer to

	in DataBlock)	define and access extended features. Information sent via this command is processed by the PC. This may be used for any feature that a vendor wishes to define.

Note 2.6.4 : After a command has been Nacked, the driver should query the Error status register and/or the IFD Status Register to find out the exact nature of the error (Card deactivated, Card Out, Wrong Command, Wrong Parameter Length, and so on).

2.6.4 IFD Status Register

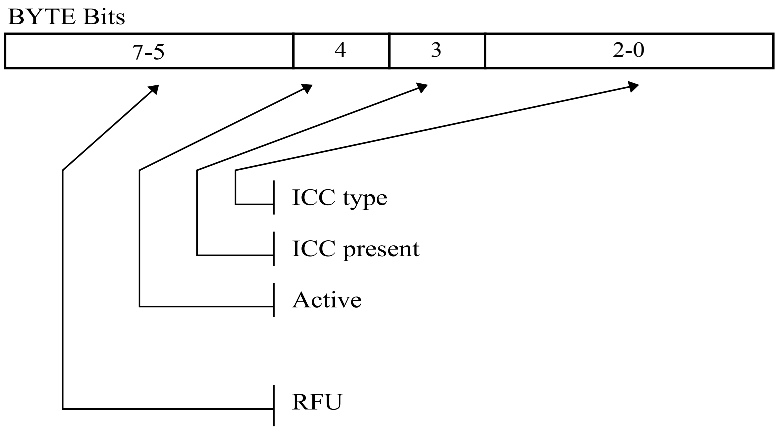

Figure 2-2. IFD Status Register

In the status register, the status bits are encoded as follows.

Table 2-11. IFD Status Register Encoding

• ICC type	Three bits encoded as a value in the range 0–7 • 0 = unknown • 1 = 7816 Asynchronous • 2 = 7816 Synchronous • 3–7 are RFU The ICC type should be determined based on the presence or absence of an ATR sequence.
• ICC present	1 if an ICC is inserted in the IFD socket
• Active	1 if the ICC contacts are active

2.6.5 IFD Error Register

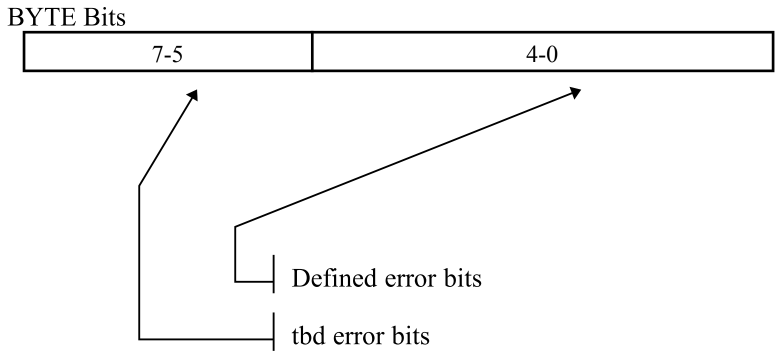

Figure 2-3. IFD Error Register

In the error register, the error bits are encoded as follows.

Table 2-12. IFD Error Register Encoding

• Bit 0	Set to 0: Communication parameters not set
• Bit 1	Set to 1: IFD turned OFF
• Bit 2	Set to 1: Invalid command received
• Bit 3	Set to 1: Length of block parameter > MAX
• Bit 4	Set to 1: Character parity error

The value indicating no error for this register will be 0x01 (b0 = 1 indicating that the communication parameters have been set, and all other bits being set to 0).

2.6.6 Security Assurance Feature Register

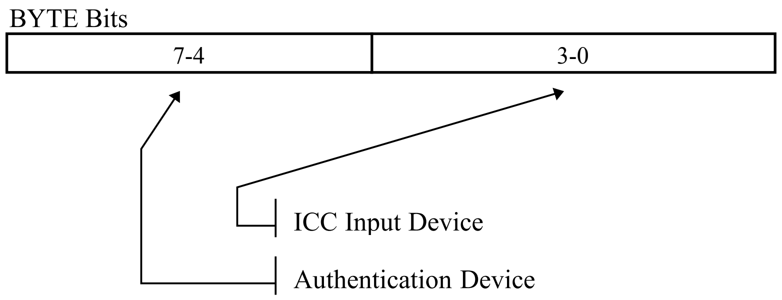

Figure 2-4. Security Assurance Feature Register

Table 2-13. Security Assurance Feature Register Encoding

ICC Input Device	Four bits encoded as a value in the range 0–15 indicating which device, if any, is currently active and exchanging data with the ICC: 0 = no device active 1 = RFU 2 = PIN (numeric) keypad 3 = Keyboard 4 = Fingerprint scanner 5 = Retinal scanner 6 = Image scanner 7 = Voice print scanner 8 = Display device 9–11 are RFU 12–15 may be used for vendor defined devices
Authentication Device	Four bits encoded as a value in the range 0–15 indicating which device, if any, is currently active collecting user authentication data for transmission to the PC: 0 = No device active 1 = RFU 2 = Encrypting PIN (numeric) keypad 3 = Encrypting keyboard 4 = Fingerprint scanner 5 = Retinal scanner 6 = Image scanner 7 = Voice scanner 8 = Display device 9–11 are RFU 12–15 may be used for vendor defined devices

2.6.7 Requirements for Multiplexed Operation

Using the command protocols defined in the preceding subsections, both the PC drivers and keyboard controller can maintain logical separation between keyboard I/O and ICC I/O operations. Hence it is a fairly simple matter to support multiplexing of ICC and command data over the keyboard I/O channel.

A more important issue is the potential impact of IFD operation on the processing of keyboard events. The keyboard is a primary input device for the system, hence operation of the IFD should not interfere with user input for extended periods of time. To insure logical separation of the event streams, it is necessary to treat IFD commands and standard keyboard commands as atomic transfers. This introduces the potential for blocking normal keyboard operation for lengthy periods of time. Based on ISO/IEC 7816 protocols, the maximum data size of a single data block will not exceed 260 bytes (T=0, 5 command bytes followed by 255 data bytes). Hence, given standard keyboard I/O data rates, the maximum latency induced could be as high as 600 ms (time required for the PC to send a 260-byte block to the keyboard). Designers should consider this factor to insure that they have adequate keystroke buffering to avoid loss of user input during ICC operation.

2.7

Security Assurance Features

A vendor may choose to implement security assurance features, such as PIN entry or biometric capabilities. Sections 2.6.3 and 2.6.6 define the associated status registers and the use of the "escape" commands to access these functions. In the opinion of the authors, these features are deemed to have significant value to end users, in terms of protecting their interests, which will generally exceed their implementation cost given the existing keyboard functionality. It should be noted that although this document has place holders for using these capabilities, much work remains to be done before these capabilities can be implemented in anything other than a vendor specific way.

3.

USB ICC IFDs

This section provides design information for vendors wishing to build a compliant USB ICC IFD. Both dependent and independent IFDs will be discussed. An independent IFD is a stand-alone device. A dependent IFD shares a USB interface processor with another device, such as a keyboard.

3.1 Rationale

Today, the addition of external peripherals is constrained by port availability. The Universal Serial Bus is the answer to connectivity for PC's. It is a bidirectional, isochronous, and low-cost bus that enables hot plugging and supports data rates up to 12 Mbps. This is the reason why USB peripherals should become common on PCs. Because PC hardware platforms are starting to support USB devices, a USB IFD could be the low-cost solution for future PC's.

3.2 Device Configuration

As described in figure 3.1, a USB device is made of three layers: the USB Interface layer, the USB Logical Device layer and the Function layer. The Interface layer provides the physical link to the USB and communicated with the USB Host controller. The Logical Device layer communicates with the PC driver. The Function layer provides a specific application like keyboard, IFD, phone, speaker, mouse, and so on.

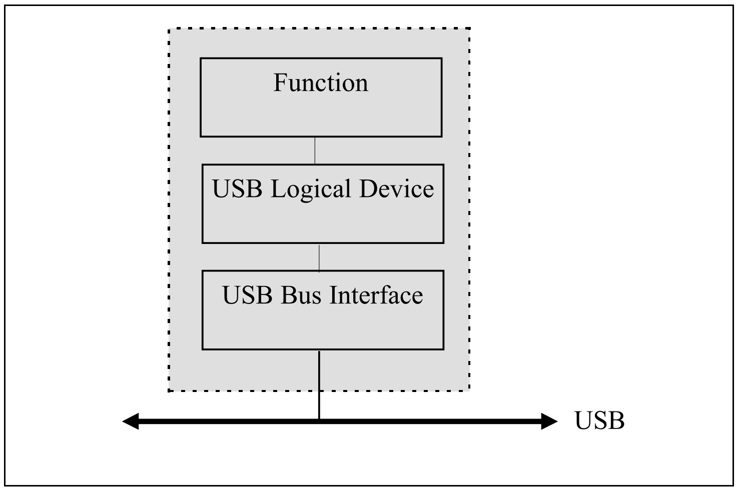

Figure 3-1. USB Device Architecture

Such an architecture enables several configurations for an IFD (see figure 3-2):

- The IFD can be a separate peripheral device with a cable that plugs into a port on a hub or directly on the Host PC. Such an IFD has its own USB interface processor and is called an independent IFD (device 1 on Figure 3-2)
- An independent IFD can implement an embedded HUB with a single USB cable. Such a device, known as a compound device, allows connections of other peripherals like phone, speakers, mouse or microphone. See device 2 on Figure 3-2.
- The IFD can also be part of a physical package that implements multiple functions with a single USB cable. Such an IFD is a called a dependent IFD. The different functions would share one USB interface processor. These functions, being keyboard and IFD (or phone and IFD, etc.), are permanently attached to an internal HUB connected to the USB. In USB terms, it is a compound device. See device 3 in Figure 3-2.

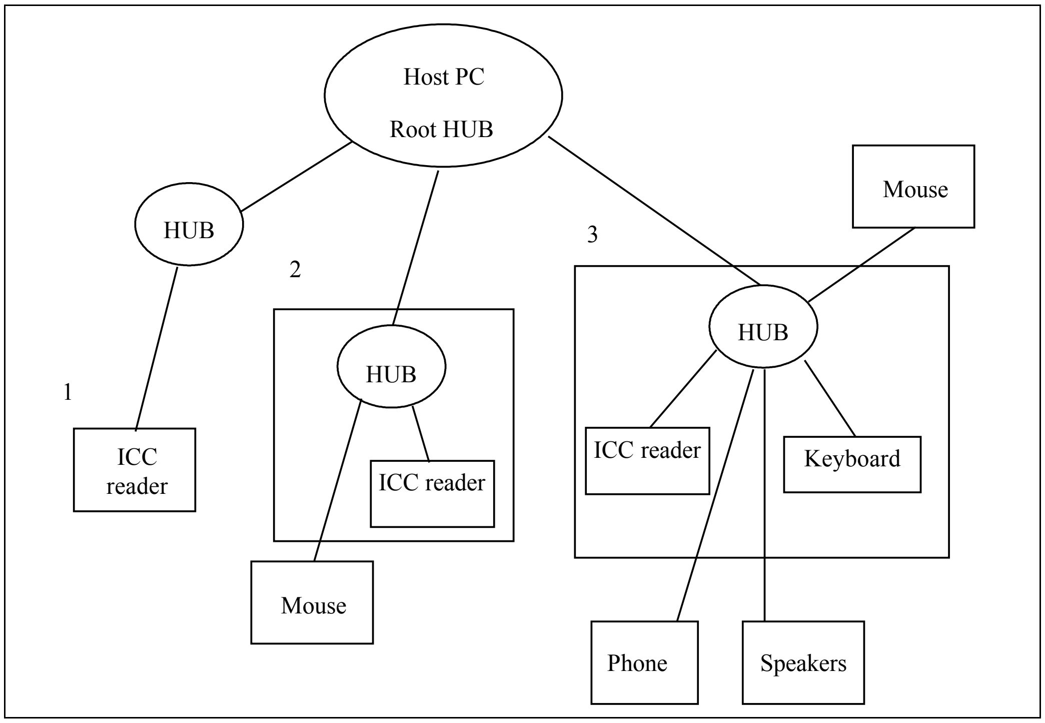

Figure 3-2. Different USB IFD configurations

The authors of this specification believe the last configuration will be the most appropriate one. Indeed, it presents most of the advantages related to the PS/2 keyboard configuration (ergonomics, cost) and, given the higher speed of USB, it should also allow concurrent transactions from the PC to the keyboard and the IFD. Finally, such a device can also be used as a HUB to easily connect other peripherals.

3.3

USB IFD Functionality

The following table summarizes expected functionality for a USB IFD, which has low cost as a primary goal.

Table 3-1. Functionality Assignment for USB IFDs

Function	Description
IFD activation	Automatic activation while IFD is attached (hot-plugging USB functionality).
ICC insertion detection	Detection of ICC insertion/removal and generation of an event to PC.
ICC contact management	Automatic activation of ICC contacts on card insertion; IFD deactivates upon removal. May optionally support ability to deactivate ICC contacts under PC control.
ICC initialization	IFD must perform cold reset following ICC contact activation, read ATR sequence if present, and send ATR to the PC. If ATR response time is exceeded, shall mark card type as unknown.
Protocol support	IFD implements physical layer, including T=0 mandatory handling of bit-parity error processing. It is expected that USB IFDs could handle the whole T=0 and T=1 data link layer, taking full advantage of the USB bandwidth available.
I/O line management	Following reset and ATR sequence, IFD assumes transmission control is determined by PC. Will transmit to the ICC when a block of data is received by the PC. It then returns bytes from the ICC to the PC as received until the PC data block is received.