APPLICATION NOTE

TDA8030 Mask 01
SINGLE USB SMART CARD READER

AN/01013

Philips
Semiconductors

PHILIPS

ABSTRACT

This document describes the software specifications that have been developed for the single USB smart card reader CAKE 8030A. This reader uses the device TDA8030 which integrates in a single chip a smart card interface and an USB interface.

The software, embedded in the TDA8030, is able to handle the communication between a host system and an asynchronous smart cards according to T=0 or T=1 protocols. The communication and the controls of the reader is done via an USB cable.

APPLICATION NOTE

SINGLE USB SMART CARD READER USING TDA8030 INTERFACE

Author(s):
Jean-Luc LUONG
System & Applications
Business Unit Identification – Business Line RIC Caen
France

Keywords
TDA8030
ISO7816-3, EMV3.1.1
PC/SC standards
USB
Single smart card reader

Date: 26 April 2001

CONTENTS

I. INTRODUCTION

I.1. Purpose

This document describes the software specifications of the single USB smart card reader CAKE 8030. This reader uses the capabilities of the USB smart card interface TDA8030.

I.2. Conventions

Unless otherwise specified, the standard bit pattern is used for all the databytes (the msb bit of a byte is at the leftmost position):

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

A decimal number is followed by a ‘d’ character.
A hexadecimal number is followed by a ‘h’ character.
A binary number is followed by a ‘b’ character.
XX stands for ‘do not care’.

I.3. Abbreviations

API	Application Programming Interface.
APDU	Application Protocol Data Unit, command message used to perform I/O operations with smart cards.
ATR	Answer To Reset.
IFD	InterFace Device, stands for the smart card interface.
IFSD	Information Field Size for the interface device, a value indicating the maximum length of blocks which can be received by the interface device in T=1 protocol. The initial value is 32.
HS	Host System.
OPCODE	OPerating CODE, command defined for this application.
PC/SC	Personal Computer Smart Card, smart card specifications for PC.
SCID	Smart Card Interface Devices, device class specification for USB smart card IFD.
S-block	Supervisory block, a type of block used to exchange control information between an interface device and the card in T=1 protocol.
SC(R)	Smart Card (Reader).
VSR	Vendor Specific Request, a non-standard USB command message used to perform I/O operations with an USB device.

I.4. Referenced documents

Document Title	Author	Reference Number
Universal Serial Bus Specification	USB work group	Revision 1.1
Data sheet of TDA8030	Philips Semiconductors	Revision of 1998 Aug. 18
USB Device Class Specification for Chip/Smart Card Interface Devices	USB Smart Card Interface Devices work group	Revision 1.0

II. FEATURES OF THE USB READER

- **Smart Card Support**
  - All asynchronous T=0, T=1 smart cards in accordance with ISO 7816-3 and PC/SC.
  - 3V and 5V smart cards.
  - Smart card data exchange from 9,600 bauds to 115,000 bauds.
  - Short circuit protection, over-current limitation, protects smart card and reader.
- **Communications**
  - Connector type B.
  - USB Specification 1.1 compliant.
  - Full speed USB (12 Mhz).
  - Card data communication via bulk in/out endpoints #1.
  - Card insertion/removal messages via interrupt endpoint #2.
  - Maximum packet size of bulk endpoint of 32 bytes.
  - Proprietary 'ALPAR' or Universal SCWG protocols.
- **Power**
  - Bus-powered.
  - Remote wakeup

III. REFERENCE HARDWARE

The schematic given in the next page shows the masked TDA8030 with the hereby described software in a typical application.

III.1. Power supply

The demoboard CAKE 8030 is configured as a bus-powered USB device. It is supplied as soon as the USB cable is connected to the host PC. The maximum power consumption is 100 mA@5V.

III.2. Interface with PC

The demoboard CAKE 8030 is provided with an USB connector type B. Use an USB cable which has a plug type A on one end and plug type B into the other end to connect the demoboard to the host PC.

III.3. Key components of the demoboard

The demoboard uses only one key component:

- SC Interface Device TDA8030 - OTP version - which has on-board 256 bytes RAM, 512 bytes auxiliary RAM and 16 kbytes of ROM.

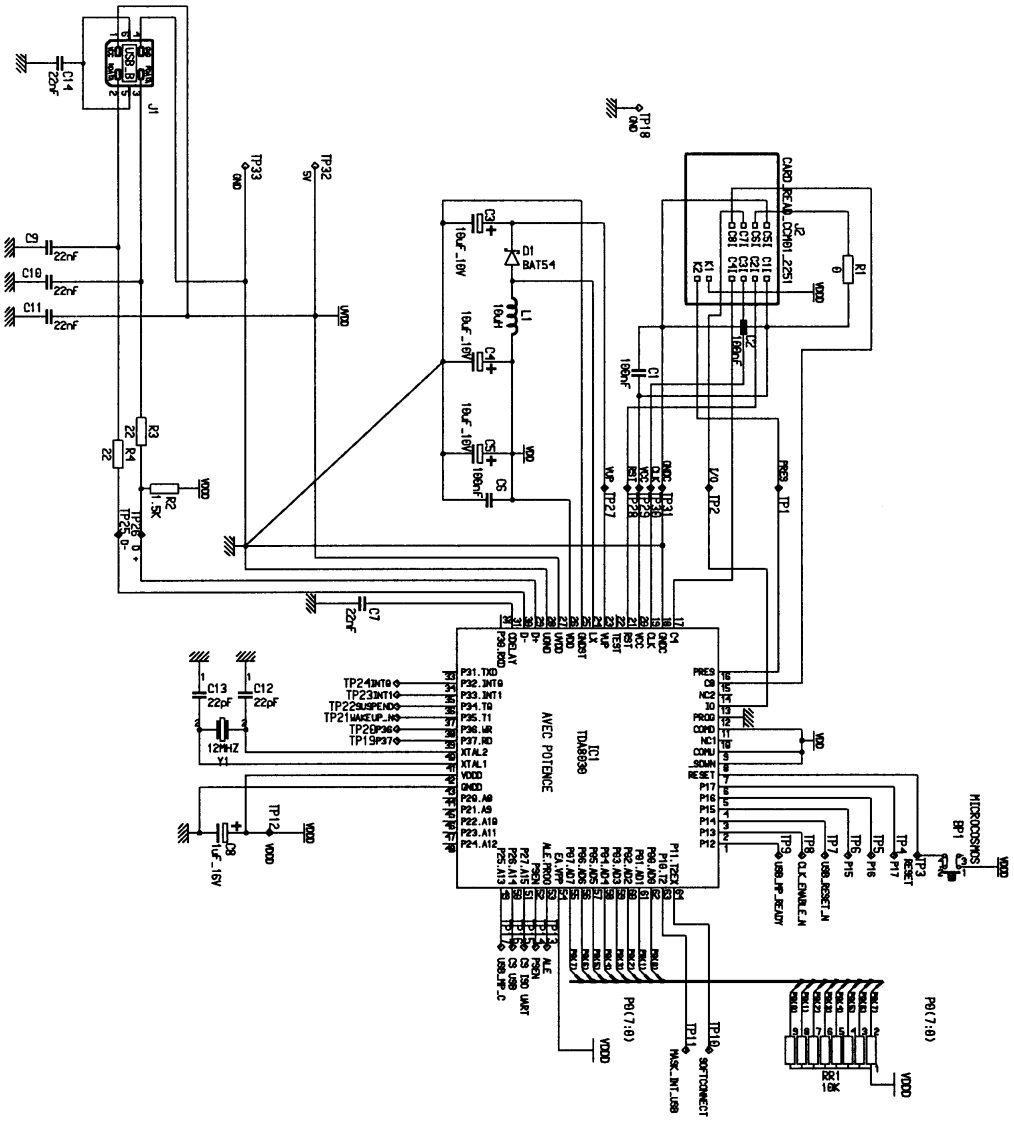


Figure 1: Smart card reader CAKE 8030-A using TDA8030 interface device

IV. USB INTERFACE

IV.1. USB description of smart card reader

An USB modelling of smart card reader is proposed below:

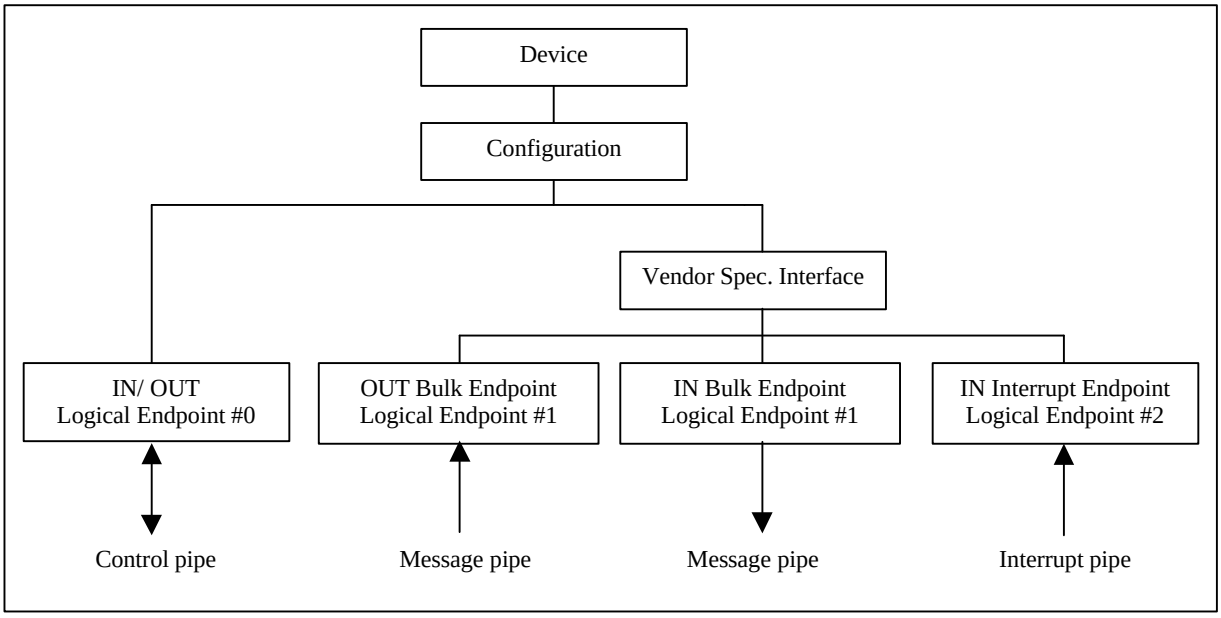


Figure 2: USB description of smart card reader

The smart card reader uses 3 endpoints which are part of the Vendor Specific Interface and the mandatory default endpoint #0:

- **Logical endpoint 0, control in/out:**
  - ♦ It is needed for initialising and configuring the logical device once the device is attached and powered.
  - ♦ It provides access to the device's information and allows generic USB status and control access.
  - ♦ Supports control transfers.

- **Logical endpoint 1, bulk out:**
  - ♦ This endpoint performs transfers to supply data to the SC reader.

- **Logical endpoint 1, bulk in:**
  - ♦ This endpoint performs transfers to retrieve data from to the SC reader.

- **Logical endpoint 2, interrupt in:**
  - ♦ It is configured as an IN endpoint, able to handle interrupt transfers in which the messages of card insertion, card extraction are notified.

IV.2. USB descriptors

USB descriptors report the attributes of an USB device. They are data structure with a fixed format defined in the document Universal Serial Bus Specification.
The descriptors of the USB reader CAKE 8030-A are listed below:

IV.2.1. Device descriptor

Offset	Field	Size	Value	Description
0	Blength	1	12h	18 bytes of descriptor length
1	BdescriptorType	1	01h	**Device descriptor**
2	BcdUSB	1	1001h	USB Spec. 1.1
4	BdeviceClass	1	00h	--
5	BdeviceSubClass	1	00h	--
6	BdeviceProtocol	1	00h	--
7	BmaxPacketSize0	1	08h	**Max. packet size of 8 bytes**
8	IdVendor	2	7104h	**PHILIPS**
10	IdProduct	2	0108h	**Product SC Reader**
12	BcdDevice	1	0001h	Device release number 1.0
13	Imanufacturer	1	00h	--
15	Iproduct	1	00h	--
16	IserialNumber	1	00h	--
17	BnumConfigurations	1	01h	**1 configuration**

IV.2.2. Configuration descriptor

Offset	Field	Size	Value	Description
0	Blength	1	09h	9 bytes of descriptor length
1	BdescriptorType	1	02h	Configuration Descriptor
2	WtotalLength	2	2700h	39 bytes of total length (9 + 9 + 7 + 7 + 7)
4	BnumInterfaces	1	01h	**1 interface**
5	BconfigurationValue	1	00h	--
6	Iconfiguration	1	00h	--
7	BmAttributes	1	A0h	**Bus powered** **Remote wakeup**
8	MaxPower	1	32h	**100 mA of max. power**.

IV.2.3. Interface descriptor

Offset	Field	Size	Value	Description
0	Blength	1	09h	9 bytes of descriptor length
1	BdescriptorType	1	04h	**Interface descriptor**
2	BinterfaceNumber	1	00h	--
3	BalternateSetting	1	00h	--
4	BnumEndpoints	1	03h	**3 endpoints**
5	BinterfaceClass	1	FFh	**Vendor Specific**
6	BinterfaceSubClass	1	FFh	**Vendor Specific**
7	BinterfaceProtocol	1	FFh	**Vendor Specific**
8	Iinterface	1	00h	--

IV.2.4. Endpoint 1 descriptor

Offset	Field	Size	Value	Description
0	Blength	1	07h	7 bytes of descriptor length
1	BdescriptorType	1	05h	Endpoint descriptor
2	BendpointAddress	1	81h	**Physical Ept #1, type IN**
3	BmAttributes	1	02h	**BULK endpoint**
4	WmaxPacketSize	2	2000h	**32 bytes of max. packet size**
6	Binterval	1	0FFh	255 ms

IV.2.5. Endpoint 2 descriptor

Offset	Field	Size	Value	Description
0	Blength	1	07h	7 bytes of descriptor length
1	BdescriptorType	1	05h	Endpoint descriptor
2	BendpointAddress	1	01h	**Physical Ept #1, type OUT**
3	BmAttributes	1	02h	**BULK endpoint**
4	WmaxPacketSize	2	2000h	**32 bytes of max. packet size**
6	BInterval	1	0FFh	255 ms

IV.2.6. Endpoint 3 descriptor

Offset	Field	Size	Value	Description
0	BLength	1	07h	7 bytes of descriptor length
1	BDescriptorType	1	05h	Endpoint descriptor
2	BendpointAddress	1	82h	**Physical Ept #2, type IN**
3	BmAttributes	1	03h	**INTERRUPT endpoint**
4	WmaxPacketSize	2	0800h	**08 bytes of max. packet size**
6	Binterval	1	0FFh	255 ms

V. PROTOCOL 'ALPAR'

The communication between the host system and the SC reader obeys to a proprietary protocol named 'ALPAR'. This protocol encapsulates the useful data of a message in an invariant frame structure and defines a dialog structure of message exchanges.

V.1. Frame structure

Every message has the following frame structure:

- 1 PATTERN byte
- 2 LENGTH bytes of the information field only
- 1 OPCODE (command byte)
- Information field

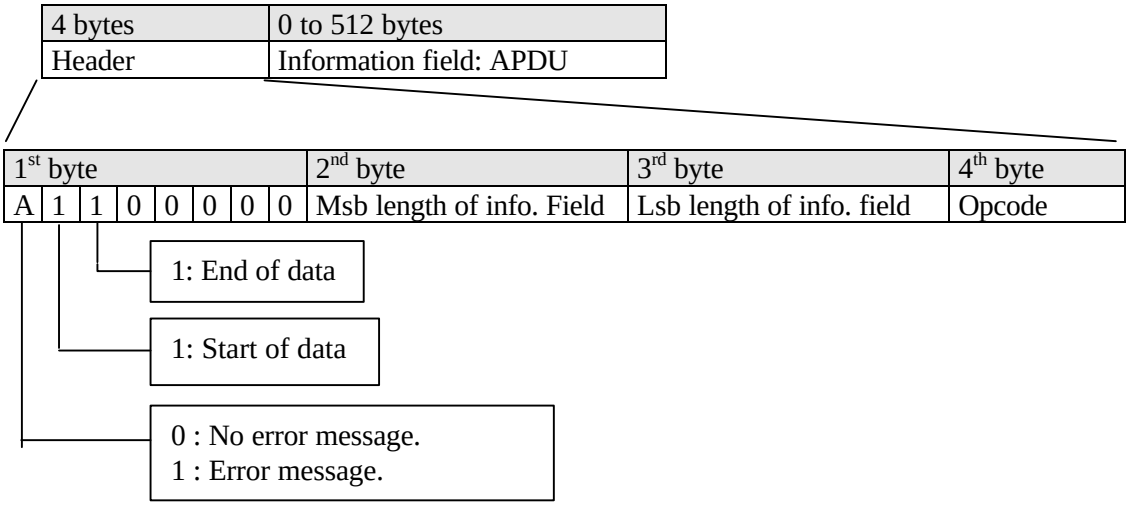


Figure 2: Frame structure of a message

So, according to the protocol rules stated above, two types of pattern can be found:

- **NORMAL PATTERN of 60h which is the first byte of every command or no error answer message.**
- **ERROR PATTERN of E0h which indicates an error answer message.**

Note : It's important to note that the protocol 'ALPAR' is not applicable for the card move messages.

V.2. Dialog structure

Most commands are initiated by the host system which is the master for the transmission. Each request message sent by the host system is followed by an answer message from the SC reader. **The opcodes of both command and its corresponding answer are the same**.

The paragraphs below show some typical dialogs.

V.2.1. Successful dialog

Host system to SC reader:	60h xx xx OPCODE1	D1 D2 D3… Dxx xx
SC reader to host system:	60h yy yy OPCODE1	d1 d2 d3 … dyy yy

V.2.2. Unsuccessful dialog

Host system to SC reader:	60h xx xx OPCODE2	D1 D2 D3… Dxx xx
SC reader to host system:	E0h 00 01 OPCODE2	ERROR CODE

V.2.3. Acknowledge

Host system to SC reader::	60h 00 02 OPCODE3, D1, D2
SC reader to host system:	60h 00 02 OPCODE3, D1, D2

In this case, **the SC reader returns a frame with the same content of the command**. This answer is also called as a **'mirror answer'**.

V.3. Usb command and response

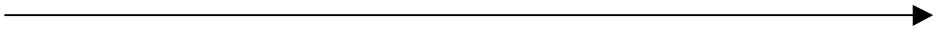
The commands are sent on the BULK-OUT endpoint #1. The response is sent on the BULK-IN endpoint #1. A new command will not be sent until the response to the current command is received. So, an exchange between the host and the usb reader is made of 2 transactions:

- bulk out transactions,
- bulk in transactions.

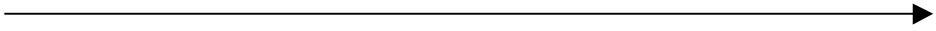
- **Bulk out transactions**

Host SC Reader

1. Bulk Out EndPoint #1



60h, Msb of BulkOutLength, Lsb of BulkOutLength, Opcode, Data[1], …, Data[60]

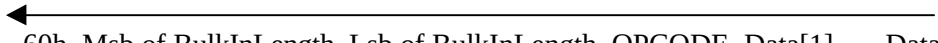


Data[N], Data[N+1],…, Data[BulkOutLength]

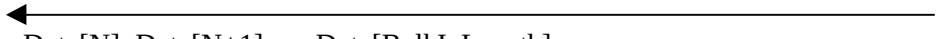
- **Bulk in transactions**

Host SC Reader

2. Bulk In Endpoint #1



60h, Msb of BulkInLength, Lsb of BulkInLength, OPCODE, Data[1], ..., Data[60]



Data[N], Data[N+1],…, Data[BulkInLength]

Packet #	Sync	OUT	ADDR	ENDP	CRC5	Idle
2386	00000001	0x87	0x03	0x2	0x1E	6

Packet #	Sync	DATA1	DATA	CRC16	Idle
2387	00000001	0xD2	00 A4 00 00 02 4F 00	0x16D4	5

Packet #	Sync	ACK
2388	00000001	0x4B

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
3648	00000001	0x96	0x03	0x1	0x07	8

Packet #	Sync	DATA1	DATA	CRC16	Idle
3649	_0000001	0xD2	60 00 02 00 90 00	0xD93F	10

Packet #	Sync	ACK
3650	00000001	0x4B

Figure 4: Example of command/response pair

Moreover, the host PC uses interrupt transactions to poll in a regular basis (255ms) the INTERRUPT IN endpoint #1 of the SC reader to get the card move status.

- **Interrupt in transaction – Card 1 inserted.**

Host SC Reader

3. Interrupt In Endpoint #2

00h, 00h, 00h, 11h

- **Interrupt in transaction – Card 1 extracted**

Host SC Reader

3. Interrupt In Endpoint #2

00h, 00h, 00h, 10h

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
22	__000001	0x96	0x02	0x1	0x18	7

Packet #	Sync	DATA1	DATA	CRC16	Idle
23	00000001	0xD2	00 00 00 10	0x7FE8	10

Packet #	Sync	ACK
24	00000001	0x4B

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
495	00000001	0x96	0x02	0x1	0x18	7

Packet #	Sync	DATA0	DATA	CRC16	Idle
496	00000001	0xC3	00 00 00 11	0xFCEB	10

Packet #	Sync	ACK
497	00000001	0x4B

Figure 5: Example of card 1 extraction/insertion messages

VI. COMMANDS DESCRIPTION

VI.1. List of Commands

Two groups of commands are defined for this application:

- General commands

These commands are used to manage the SC reader. The following table shows the corresponding instructions:

GENERAL COMMANDS	VALUE	INPUT PARAMETERS	RESPONSE FROM READER
Check card presence	09h	--	1 parameter: ❑ 01h: Card present ❑ Or 00h: Card absent
Mask number	0Ah	--	A string of 15 ASCII characters For instance '**01 Release 1.0'**

- Card related commands

These commands are used to perform the complete dialog with the card connected to the TDA8008 interface device. The following instructions are defined:

GENERAL COMMANDS	VALUE	INPUT PARAMETERS	RESPONSE FROM READER
Power up card 3V	6Dh	Power up parameter: ❑ 00h: ISO ❑ 01h: EMV	❑ ATR + 2 status bytes ❑ Or error message
Power up card 5V	6Eh	Same as above.	❑ Same as above
Power off card	4Dh	--	❑ Acknowledge ❑ Or error message
Card command	00h	N bytes: CLA...P3 + P3 data	❑ Card response ❑ Or error message
Negotiate	10h	2 parameters: ❑ Protocol ❑ FD value	❑ Acknowledge ❑ Or error message
Set card clock	11h	Card clock parameter	❑ Same as above
Set divider	0Bh	Divider parameter	❑ Same as above
IFSD request	0Ch	IFSD parameter	❑ Same as above

VI.2. Card move messages

When a card is inserted (resp. extracted), the SC reader sends a card inserted (resp. card extracted) messages to the host.

MESSAGES	HEX STRINGS
Card 1 extracted	00 00 00 11h
Card 1 inserted	00 00 00 10h

VI.3. General commands

VI.3.1. Check card presence

The command is used to check the card status of the current active slot. A parameter – 01h for card present, 00h for card absent - is returned.

VI.3.2. Mask number

This command is used to identify the software embedded in the TDA8008. A string of 15 ASCII characters for instance '**01 Release 1.0**' is sent back.

VI.4. Card related commands

VI.4.1. Power up card 3V

This command allows to activate a card with a 3V Vcc voltage. All the signals supplied to it are referenced to this voltage. The activation sequence is the same as the one of power up card 5V.

VI.4.2. Power up card 5V

This command is used by the host system to activate the card and to initiate the communication with the card.

First, an activation sequence conform to ISO7816-3 normalisation is performed (if the input parameter is 00h). If the contacts already were active, only the RST pin is switched (warm reset).

The ATR is expected from the card according to the ISO standard activation sequence. The parameters of ATR are decoded and analysed.

If the work waiting time is elapsed and no ATR is received, the card is considered as "mute" and deactivated.

The activation is always initiated at a card clock frequency of 3 MHz (the card clock frequency is get by dividing by 4 the 12 MHz crystal frequency of the TDA8008) for both cards, with no guard time and with the default work waiting time of 10.

If the card has answered to the reset sequence, the parameters TS, T0, TA1, TB1, TC1, TD1, TA2, TB2, TC2, TD2, TA3, TB3 and TC3 are evaluated by the software.

The timing parameters i.e. extra guard time, work waiting time (applicable for T = 0 protocol), character waiting time and block waiting time (applicable for T = 1) are automatically extracted from the ATR, and memorised. They will be used for all further communications with the card.

At the end of a successful (resp. unsuccessful) activation, the SC reader sends back the ATR and the status words (resp. an error message).

VI.4.3. Power off card

This command deactivates the card according to ISO7816-3 standards whatever it has been activated for 3V or 5V operation.
This function is automatically initiated either:

- the events like card take-off, supply voltage drop, short-circuit or overheating occur. For the three last events, it is done by the hardware of the TDA8008.
- the card is considered as absent or mute during a card activation.

An acknowledge is returned to the host system

VI.4.4. Card commands

This command is used to transfer C-APDU and R-APDU to/from the asynchronous card.
The format of a C-APDU is the following:

CLA	INS	P1	P2	P3	data
…					

The number of data is given by the parameter P3. ***A maximum number of 255 data can be sent to the card.*** The card answers two parameters SW1 and SW2.
For a length (P3) of 0, no data is transferred to the card. In this case, the card may or not send an acknowledge byte before the status words. ***The maximum value of P3 is 255.***
This command is also used to **get data from the card**. In read operations, ***if P3 is equal to 0, a maximum of 256 data can be requested from the card.***
The card answers either with the procedure byte, P3 data plus and the status words SW1 and SW2 or only with SW1, SW2 if there was an error in the command. The procedure byte is analysed but never transferred to the host system.
After a successful (resp. unsuccessful) operation, the SC reader sends back the R-APDU (resp. an error message).

VI.4.5. Negotiate

If the card which is in negotiable mode (TA2 is not present in the ATR) is able to handle different protocols (T=0 and T=1) and supports to work at a higher baud rate on I/O line (Fi/Di values are different from default value 372), a PPS request can be initiated just after an ATR to negotiate:

- a protocol,
- the ratio Fi/Di.

The response of the SC is evaluated. If the negotiation is accepted, an acknowledge is sent to the host and the new parameters will be used in all subsequent exchanges. Otherwise, an error is reported.

VI.4.6. Set card clock

This command is used to select a new card clock frequency. The chosen configurations are given to the SC reader through the clock status parameter, coded as follows:

Clock status parameter in hex.	Clock frequency
2h	Fxtal/2 (6 Mhz)
4h	Fxtal/4 (3 Mhz)
6h	Fxtal/8 (1,5 Mhz)

At the beginning of a card session, the clock status parameter is initialised at 2 (Fclk = Fxtal/4 -3 MHz- during the ATR phase) and remains unchanged until a next “SET CARD CLOCK” command is used to change the clock frequency of the card.
After a successful (resp. unsuccessful) operation, an acknowledge (resp. error message) is returned.

VI.4.7. IFSD request

This command is used to send a S-block to the card indicating the maximum length of information field which can be received by the interface device in T=1 protocol.
After a successful (resp. unsuccessful) operation, an acknowledge (resp. error message) is returned.

VII. ERROR LIST

The table below enumerates the error codes used in this application and their meaning:

Error code	Meaning of error code
08h	Buffer length too short.
0Ah	3 errors in T=1.
20h	Wrong APDU.
21h	Too short APDU.
22h	Card mute during T=1 exchange.
24h	Bad NAD in a prologue field in T= 1
25h	Bad LRC in T = 1
26h	Card resynchronised in T = 1
27h	Chain aborted.
28h	Bad PCB in a prologue field in T = 1
29h	Card data overflow.
2Ah	NAD not initialised
30h	Non negotiable mode
31h	Protocol different of T= 0 and T=1.
32h	Negotiation of protocol T=1 failed.
33h	PPS answer different of PPS request.
34h	Error on PCK.
35h	Parameter error
36h	Buffer overflow.
37h	BGT not reached.
38h	TB3 not present.
39h	PPS not accepted.
40h	Card deactivated
55h	Unknown command
80h	Card mute after power on.
81h	Time out.
82h	3 parity errors in TS.
83h	3 parity errors in reception
84h	3 parity errors in transmission.
85h	ATR too long.
86h	Bad FiDi
87h	Card clock not accepted.
88h	ATR duration longer than 19200 ETUs
89h	CWI not supported.
8Ah	BWI not supported.
8Bh	WI not supported.
8Ch	TC3 not supported.
8Dh	Parity errors during ATR.
8Eh	Error on SW1

90h	3 parity errors in T=1
91h	SW1 different of 6X and 9X
92h	Specific mode TA2 (b5=1)
93h	TB1 absent during cold reset
94h	TB1 different from 0 during cold reset
95h	IFSC less than 10h or IFSC equal to FFh
96h	Wrong Tdi
97h	TB2 present in ATR
98h	TC1 not compatible with CWT
99h	IFSD not accepted
A0h	Procedure byte error
B0h	Protected byte.
B1h	Pin error.
B2h	Write error.
B3h	Data length too long.
B4h	Error counter too long.
B5h	Pin code not verified.
B6h	Protected bit already set.
B7h	Verify pin error.
C0h	Card absent.
C1h	Io line locked
C2h	Protocol different T=0 and T=1.
C3h	Checksum error.
C4h	TS neither 3B or 3F.
C5h	Bad FiDi.
C6h	ATR not supported.
C7h	VPP not supported.
C8h	VCC error.
C9h	Card removed.
Cah	Class error without deactivation.
CBh	Class error with deactivation.
D0h	Bad secret code.
D1h	Card locked.
D2h	Write error.
D3h	Max erase time reached.

VIII. ANNEX 1: USB TRANSACTIONS

Three types of transactions are used by the host to communicate with the endpoints of the SC reader:

- Control transaction.
- Interrupt transaction.
- Bulk transaction.

VIII.1.Control transaction

Control transaction uses a three phase transaction consisting of token, data and handshake packets. Firstly a SETUP token is issued to transmit information to the control endpoint of the SC reader. The data stage (when present) consists of one or more IN or OUT tokens which carry the amount of data.

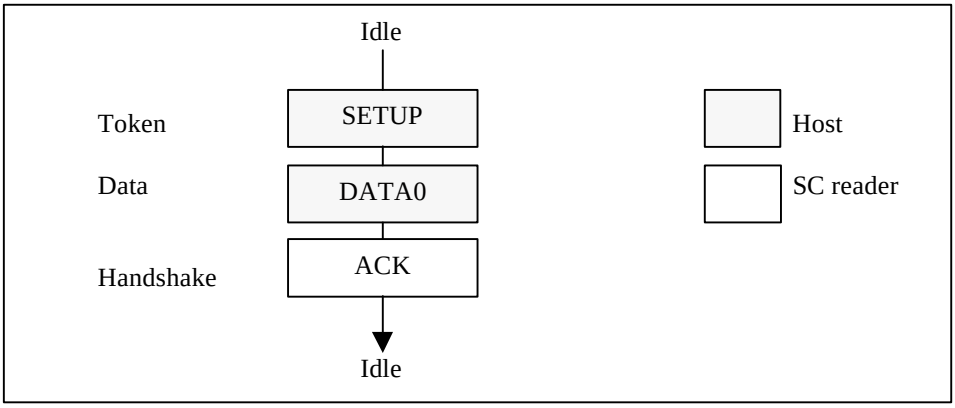


Figure 6: Control transaction format

Packet #	Sync	SETUP	ADDR	ENDP	CRC5	Idle
21	00000001	0xB4	0x03	0x0	0x0A	6

Packet #	Sync	DATA0	DATA	CRC16	Idle
22	00000001	0xC3	41 00 00 00 00 00 04 00	0x1E13	5

Packet #	Sync	ACK
23	00000001	0x4B

Packet #	Sync	OUT	ADDR	ENDP	CRC5	Idle
1246	00000001	0x87	0x03	0x0	0x0A	6

Packet #	Sync	DATA1	DATA	CRC16	Idle
1247	00000001	0xD2	60 00 07 00	0xC7D7	5

Packet #	Sync	ACK
1248	00000001	0x4B

Figure 7: An example of control transaction

The control transactions are used to carry the header part of the messages exchanged between the host PC and the SC reader. They are followed by bulk transactions which carry the data part of messages.

VIII.2.Interrupt transaction

Interrupt transaction consists solely of IN token. After receiving an IN token, the endpoint responds by returning eiher a DATA packet or a NAK or STALL handshake.

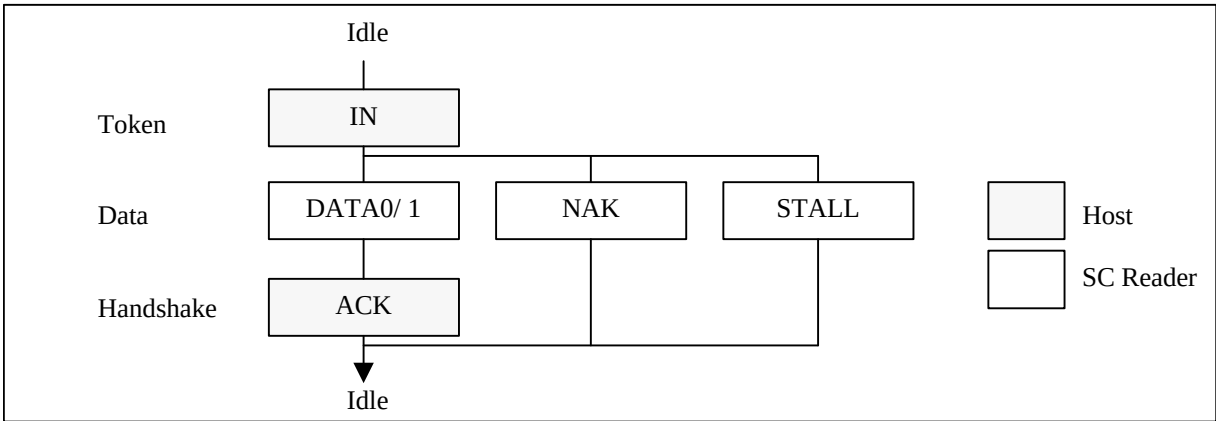


Figure 8: Interrupt transaction format

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
22	__000001	0x96	0x03	0x3	0x13	6

Packet #	Sync	DATA0	DATA	CRC16	Idle
23	00000001	0xC3	60 00 01 A0 00	0x4EEC	10

Packet #	Sync	ACK
24	00000001	0x4B

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
345	00000001	0x96	0x03	0x3	0x13	6

Packet #	Sync	DATA1	DATA	CRC16	Idle
346	00000001	0xD2	60 00 01 A0 00	0x4EEC	10

Packet #	Sync	ACK
347	00000001	0x4B

Figure 9: An example of interrupt transactions

The interrupt transactions are used by the host PC every 255 ms to get from the SC reader the card presence status.

VIII.3.Bulk transaction

Bulk transaction uses a three phase transaction consisting of token, data and handshake packets.
When the host wishes to receive bulk data, it issues an IN token. The endpoint responds by returning eiher a DATA packet or a NAK or STALL handshake.
When the host wishes to transmit bulk data, it first issues an OUT token followed by a DATA packet.

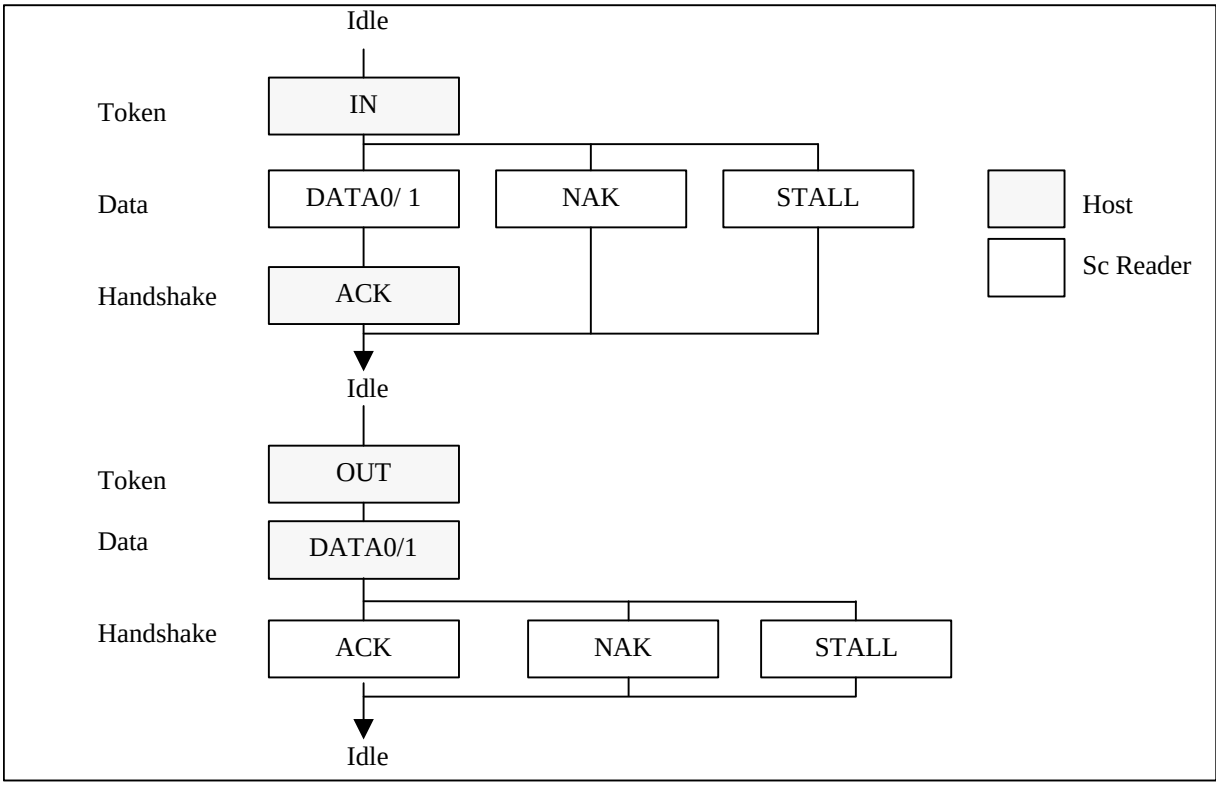


Figure 10: Bulk transaction format

Packet #	Sync	OUT	ADDR	ENDP	CRC5	Idle
2386	00000001	0x87	0x03	0x2	0x1E	6

Packet #	Sync	DATA1	DATA	CRC16	Idle
2387	00000001	0xD2	00 A4 00 00 02 4F 00	0x16D4	5

Packet #	Sync	ACK
2388	00000001	0x4B

Packet #	Sync	IN	ADDR	ENDP	CRC5	Idle
3648	00000001	0x96	0x03	0x1	0x07	8

Packet #	Sync	DATA1	DATA	CRC16	Idle
3649	_0000001	0xD2	60 00 02 00 90 00	0xD93F	10

Packet #	Sync	ACK
3650	00000001	0x4B

Figure 11: Examples of bulk out and bulk in transactions

The bulk transactions are used to carry the data part of the messages exchanged between the host PC and the SC reader. They are preceded by control transactions which carry the header part of messages.

IX. ANNEX I : USB INTERFACE DRIVER

The USB interface driver (USBID) plays the role of host system for the demoboard CAKE 808-A It uses the layered architecture shown below :

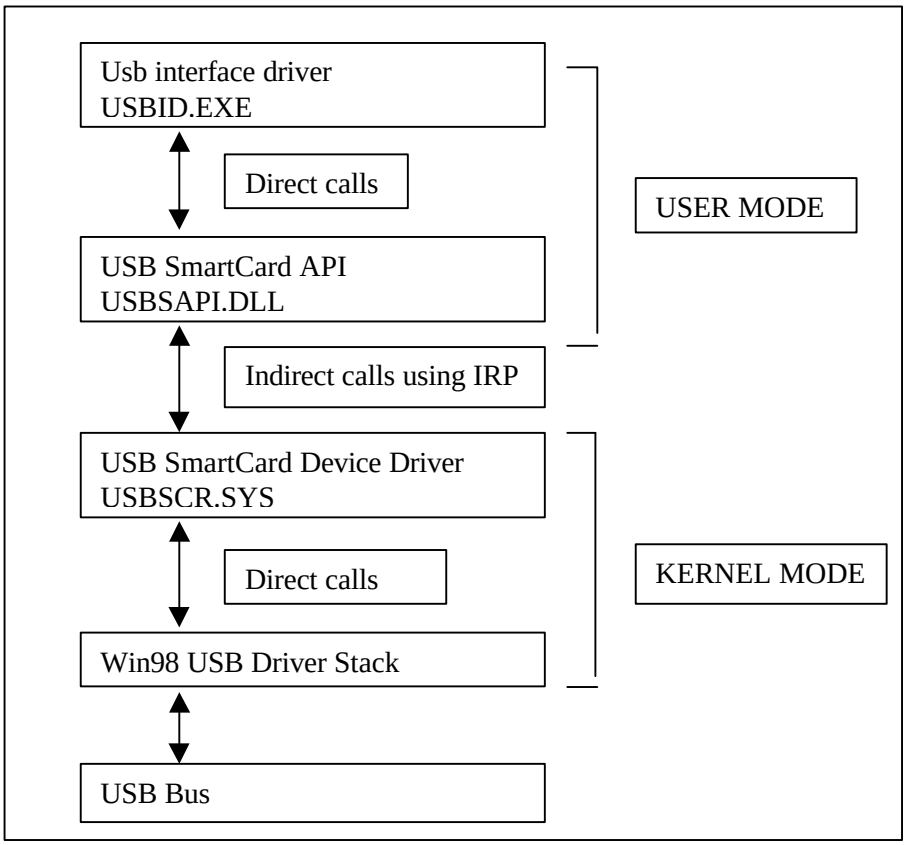


Figure 12: Layered architecture of USB interface driver.

The USB interface driver calls directly the SC API functions of the file USBDAPI.DLL.
The API functions performs indirect calls using IRP (I/O Request Packets) to I/O functions of the device driver USBSCR.SYS.
This module uses the capabilities of the Win98 USB Driver Stack to achieve I/O operations in the USB bus.

X. ANNEX II: PROTOCOL ‘SCID’

In addition to the proprietary protocol ‘ALPAR’, an another protocol called ‘SCID’ can be used for this application. This protocol issued by a international workgroup which includes most of the actors on the smart card market, defines Device Class Specifications for USB smart card interface devices.
As the protocol ‘SCID’ is published in the USB SCID device class specification document, it is not detailed in this report.
Some SCID messages, defined in the document are not supported in this application. The table below shows the support status of SCID messages.

Message Name	Value	Options	Supportable
PC_to_RDR_CardPowerOn	62h	Automatic 5V 3V 1.8V	No Yes No Yes
PC_to_RDR_CardPowerOff	63h	--	Yes
PC_to_RDR_GetSlotStatus	65h	--	Yes
PC_to_RDR_XfrBlock	6Fh	--	Yes
PC_to_RDR_GetParameters	6Ch	--	No
PC_to_RDR_ResetParameters	6Dh	--	No
PC_to_RDR_SetParameters	61h	--	Yes
PC_to_RDR_Escape	6Bh	Mask number (0Ah) Ifs request (0Ch) Negociate (10h) Select card (6Ah) Check reader status (AAh)	Yes Yes Yes Yes Yes
PC_to_RDR_IccClock	6Eh	--	Yes
PC_to_RDR_T0APDU	6Ah	--	No
PC_to_RDR_Secure	70h	--	No
PC_to_RDR_Mechanical	71h	--	No
RDR_to_PC_NotifySlotChange	50h	--	Yes
RDR_to_PC_HardwareError	52h	--	Yes

XI. ANNEX III: USB SMART CARD API

XI.1. FUNCTIONS

XI.1.1. SCOpen

HANDLE SCOpen();

Opens the SCR device.

Parameters

Return Value
Returns the handle for opened SCR device if successful. Otherwise returns STATUS_INVALID_HANDLE.

Remarks

XI.1.2. SCClose

BOOL SCClose (HANDLE hDevice **);**

Closes the opened SCR device.

Parameters
hDevice Handle of the device to close.

Return Value
Returns TRUE if successful. Otherwise returns FALSE.

Remarks

XI.1.3. SCReset

VOID SCReset (HANDLE hDevice **);**

Resets an opened SCR device.

Parameters
hDevice Handle of the device.

Return Value

Remarks
This function resets the USB host controller driver.

XI.1.4. SCRead

UINT SCRead (HANDLE hDevice, **LPVOID** lpBuffer, **UINT** nLength **);**

Reads requested bytes of data from an opened SCR device.

Parameters

hDevice	Handle of the device.
lpBuffer	Address of the buffer that receives data.
nLength	Number of bytes to read.

Return Value

Returns the number of the bytes read. If the function fails the return value is 0.

XI.1.5. SCWrite

UINT SCWrite (HANDLE hDevice, **LPCVOID** lpData, **UINT** nLength **);**

Sends requested amount of data to SCR device.

Parameters

hDevice	Handle of the device.
lpData	Address of the data to send.
nLength	Number of bytes to send.

Return Value

Returns the number of the bytes written. If the function fails the return value is 0.

XI.1.6. SCCommand

BOOL SCCommand (HANDLE hDevice, **LPCVOID** lpCmd, **UINT** nLength **);**

Sends the requested command to the SCR device.

Parameters

hDevice	Handle of the device.
lpCmd	Pointer to command to send.
nLength	Length of the command.

Return Value

Returns TRUE if operation is successful. Otherwise returns FALSE.

XI.1.7. SCNotifyWindow

HWNDSCNotifyWindow (HWND hWnd **);**

Sets the window to receive notification messages from SC API.

Parameters
hWnd Handle of the window to receive notification messages.

Return Value
Returns the handle of the previous window that receives notification messages. Otherwise returns NULL.

Remarks
This function sets the window that receives the SC API notification messages. When a notification event occurs API sends **WM_SC_NOTIFY** message to the associated window. For more information about WM_SC_NOTIFY message see WM_SC_NOTIFY section in SC API documentation. Use this function to receive notification events unless you are not programming in Windows GUI. Otherwise you can use **SCWaitEvent.**

XI.1.8. SCWaitEvent

UINT SCWaitEvent (DWORD dwTimeOut **);**

Waits for a notification event from SC API in the specified time duration.

Parameters
dwTimeOut Timeout value in miliseconds for this request.

Return Value
Returns code of the notification event or ERR_SC_TIMEOUT if there is no notification events received in specified duration.

Remarks
Function returns the code of the notification event if there is one fired in specified duration. Otherwise it will return ERR_SC_TIMEOUT. For notification event codes, see *Notification Events* section.

This function blocks the calling thread until an event fired or timeout value exceeded which means you cannot receive any Windows message.

If you are programming with Windows GUI use **SCNotifyWindow** to receive notification events. This will enable the calling thread to continue its user interaction and message processing.

XI.1.9. SCIsDeviceActive

BOOL SCIsDeviceActive ();
Checks SCR device is active or not.

Parameters

Return Value
Returns TRUE if SCR device is active. Otherwise returns FALSE.

Remarks
This function tries to find out an active SC device in the system. If it finds one, returns TRUE.

XI.2. MESSAGES

XI.2.1. WM_SC_NOTIFY

Sent by SC API to the default notification window set by **SCNotifyWindow** when a notification event occurs.

WM_SCNOTIFY
uEventCode = (UINT) lParam

Parameters

wParam	Value of the wParam contains 16 bit event code. See *Remarks*
lParam	Value of the lParam contains 32 bit event code. See *Notification Events*

Return Value
An application processing this message should return zero.

Remarks
Value of the lParam contains the 32 bit event code and data sent from USB SCR hardware. In previous version of the API, event code can contain only 16 bits of data. Since the new version it supports 32 bit event codes it is no longer suitable to use wParam parameter of this message. But wParam still contains first 16 bit of the data for backward compatibility.

XI.2.2. WM_SC_DEVICE_READY

Sent by SC API to the default notification window set by **SCNotifyWindow** when device driver is ready for communicating with USB SCR hardware. It is sent when USB SCR device plug is inserted and the device is enumerated by Windows correctly.

WM_SC_DEVICE_READY

Parameters

wParam Not used.
lParam Not used.

Return Value

An application processing this message should return zero.

XI.2.3. WM_SC_DEVICE_REMOVED

Sent by SC API to the default notification window set by **SCNotifyWindow** when device driver is no longer available for communicating with USB SCR hardware. It is sent when USB SCR device plug is removed and before device driver is unloaded.

WM_SC_DEVICE_REMOVED

Parameters

wParam Not used.
lParam Not used.

Return Value

An application processing this message should return zero.

XI.3. NOTIFICATION EVENTS

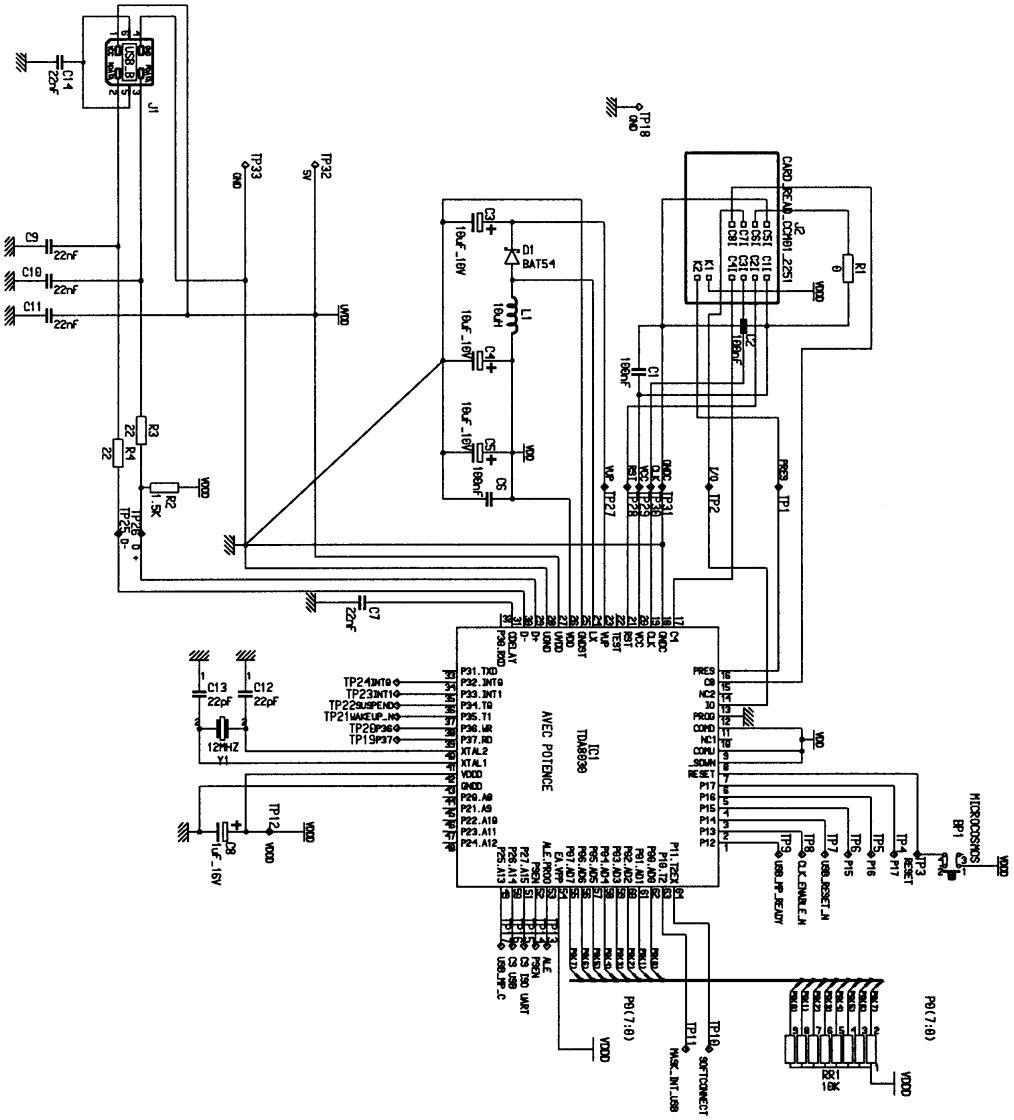
XI.3.1. EV_SC_CARD_INSERTED

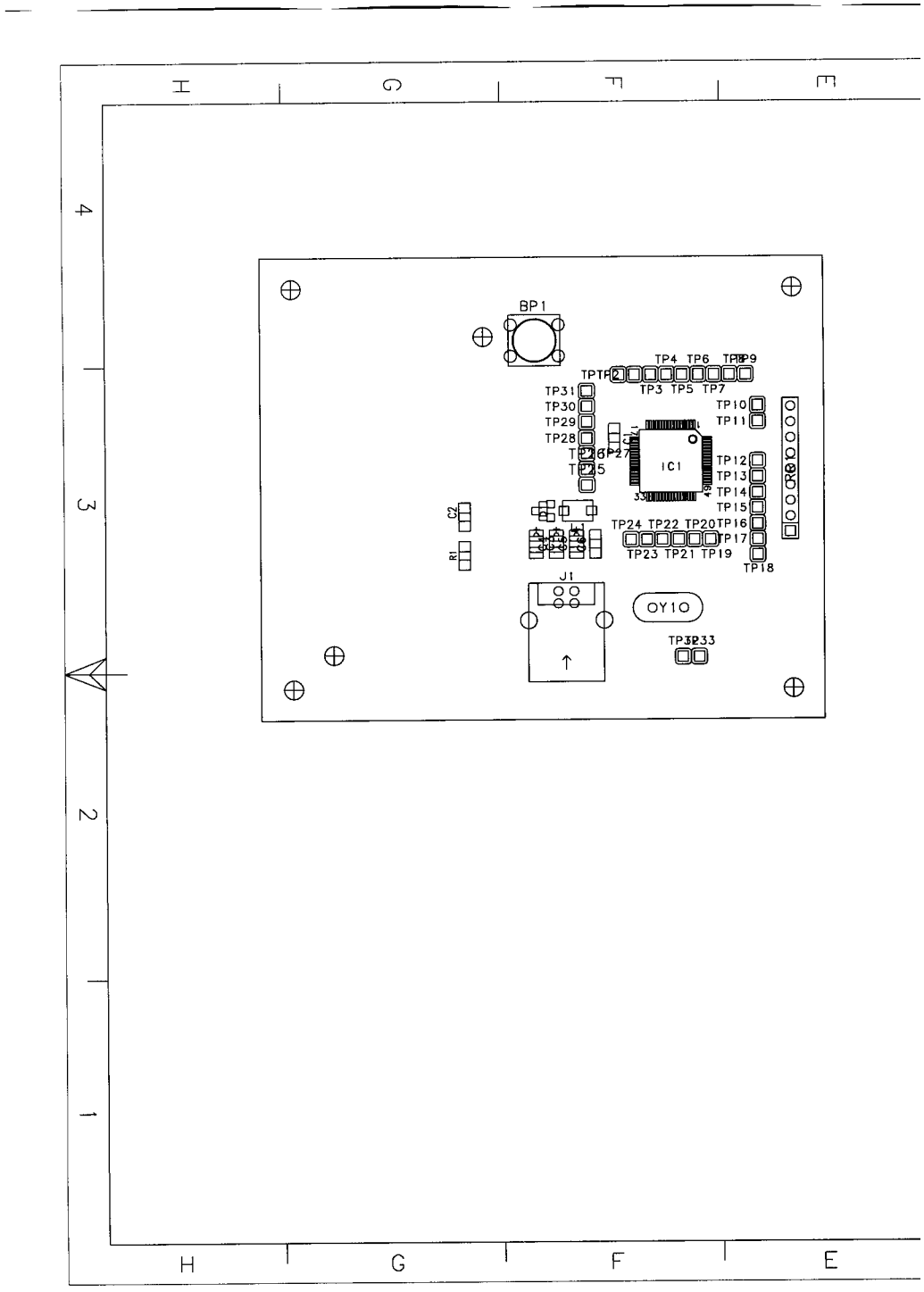
Notifies that a card inserted into the device.

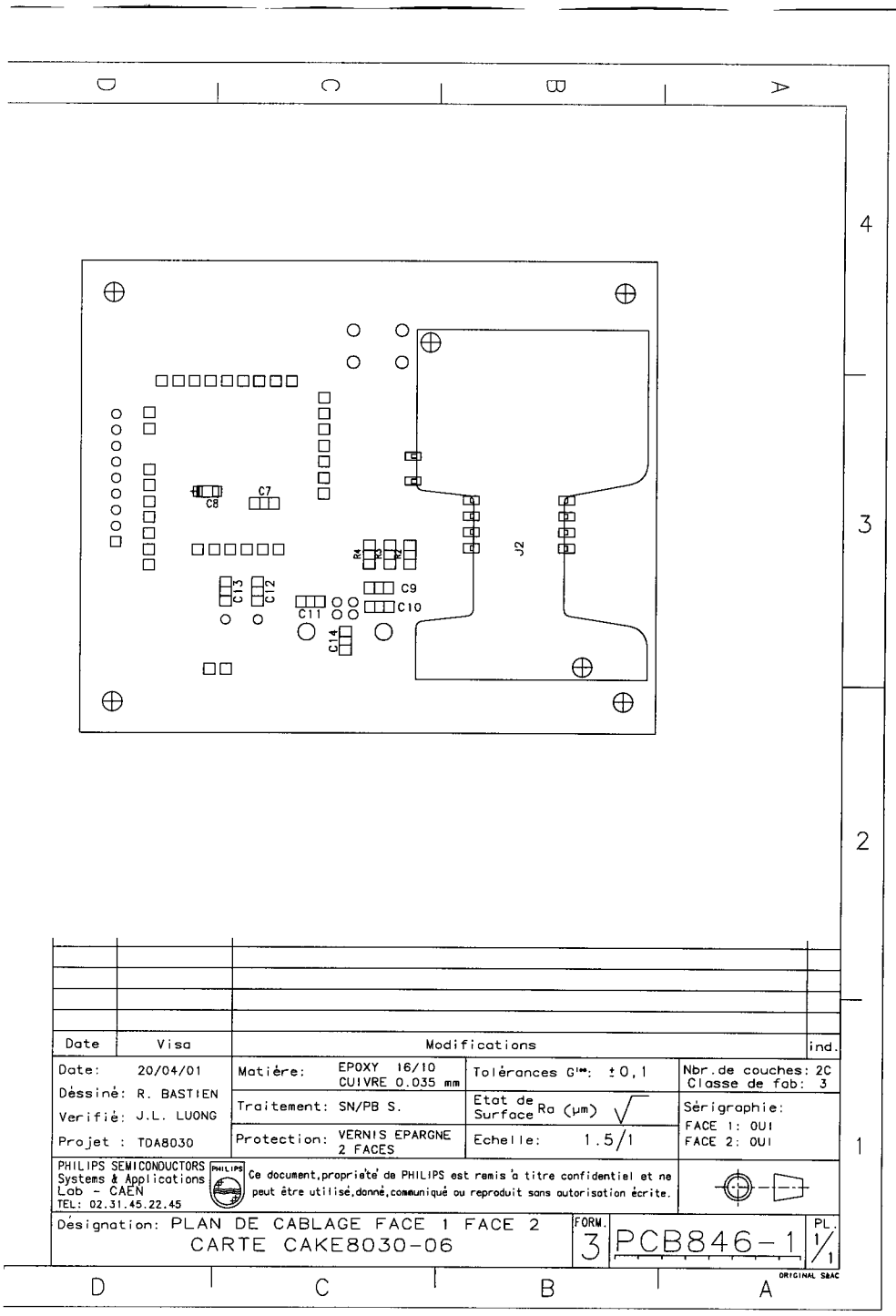
XI.3.2. EV_SC_DEVICE_REMOVED

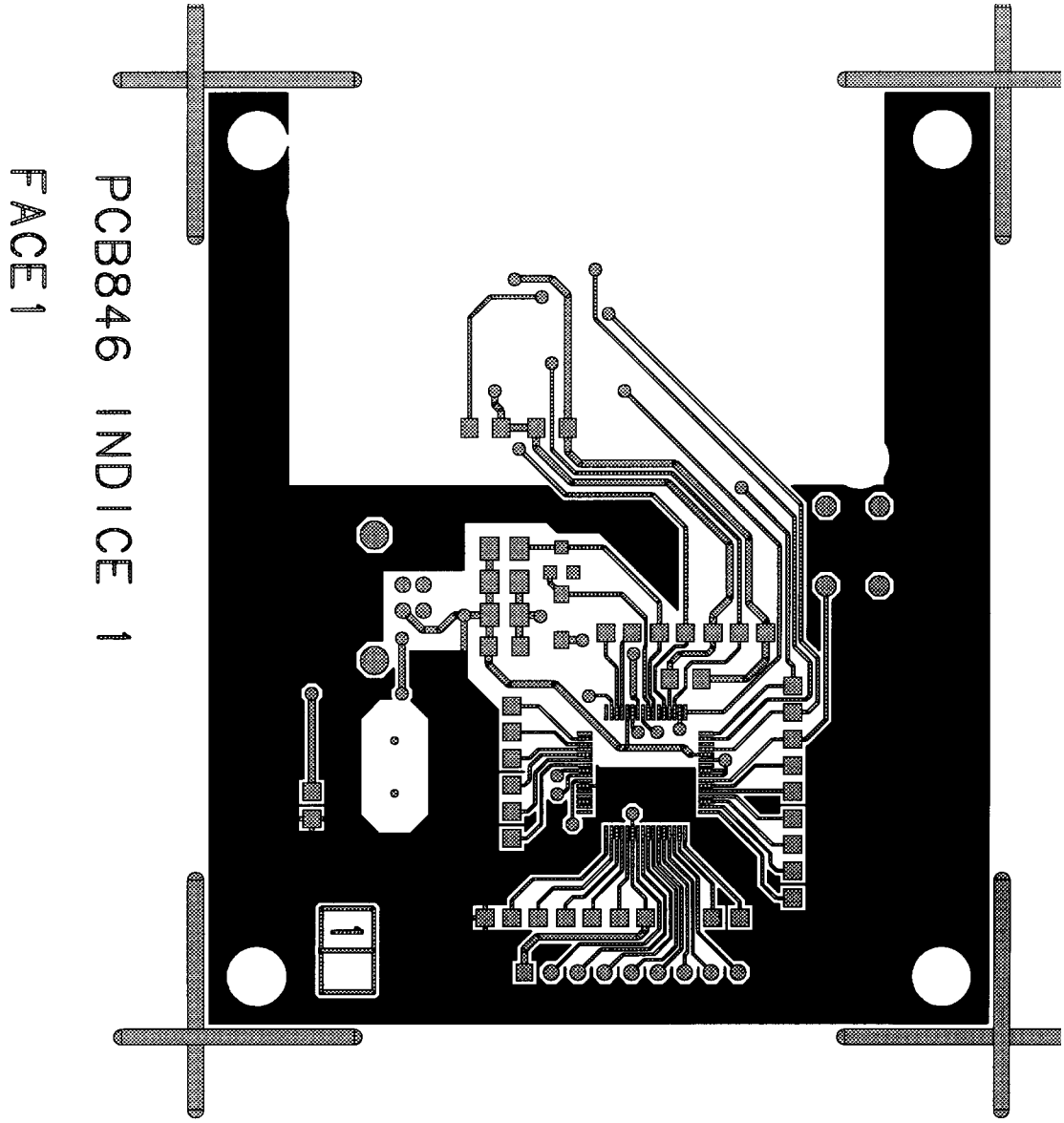
Notifies that a card removed from the device.

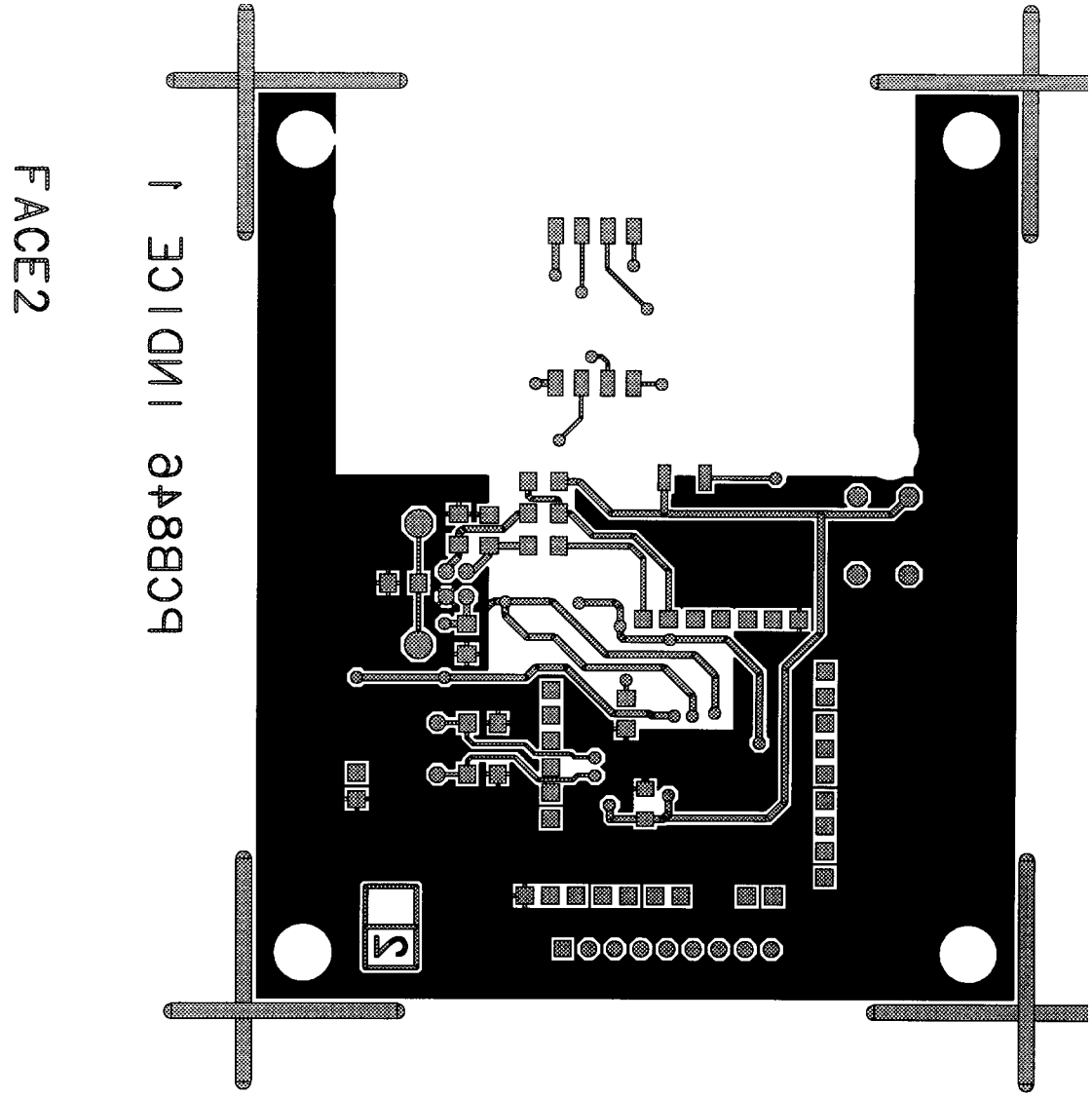
XII. ANNEX IV: DEMOBOARD




H
G
F
E
4
3
2
1
BP1
TP4 TP6
TP3 TP5 TP7
TP10
TP11
TP31
TP30
TP29
TP28
IC1
TP12
TP13
TP14
TP15
TP24 TP22 TP20
TP16
TP17
TP23 TP21 TP19
TP18
J1
OY10
TP32
TP33


D
C
B
A
4
3
2
1
C8
C7
C13
C12
C11
C14
C9
C10
R4
R3
R2
J2
Date
Visa
Modifications
ind.
Date: 20/04/01
Dessiné: R. BASTIEN
Verifié: J.L. LUONG
Projet : TDA8030
Matière: EPOXY 16/10 CUIVRE 0.035 mm
Traitement: SN/PB S.
Protection: VERNIS EPARGNE 2 FACES
Tolérances G'': ±0,1
Etat de Surface Ra (µm)
Echelle: 1.5/1
Nbr.de couches: 2C
Classe de fab: 3
Sérigraphie:
FACE 1: OUI
FACE 2: OUI
PHILIPS SEMICONDUCTORS
Systems & Applications
Lab - CAEN
TEL: 02.31.45.22.45
Ce document,propriété de PHILIPS est remis à titre confidentiel et ne peut être utilisé,donné,communiqué ou reproduit sans autorisation écrite.
Désignation: PLAN DE CABLAGE FACE 1 FACE 2
CARTE CAKE8030-06
FORM.
3
PCB846-1
PL.
1/1
ORIGINAL SBAC


PCB846 INDICE 1
FACE1


FACE2
PCB84e INDICE 1

LABORATOIRES D APPLICATIONS PHILIPS
2 Rue de la Girafe
BP 5120
14079 CAEN CEDEX 5
R.BASTIEN TEL: 02.31.45.31.17
FAX: 02.31.45.30.70

CARTE : PCB846 IND : 1 ETUDE: CAKE8030_06

PROJET : CAKE8030 LE : 20 Avril 2001

FAIT PAR : bastien

LISTE DES COMPOSANTS FACE 1 FACE 2

REFERENCE	GEOMETRY	DESCRIPTION
BP1	microcosmos	poussoir2, MICROCOSMOS
C1	c1206	capa, 100nF
C2	c1206	capa, 100nF
C3	c595d_a	capa_pol, 10uF_10V
C4	c595d_a	capa_pol, 10uF_10V
C5	c595d_a	capa_pol, 10uF_10V
C6	c1206	capa, 100nF
C7	c1206	capa, 22nF
C8	c293d_a	capa_pol, 1uF_16V
C9	c1206	capa, 22nF
C10	c1206	capa, 22nF
C11	c1206	capa, 22nF
C12	c1206	capa, 22pF
C13	c1206	capa, 22pF
C14	c1206	capa, 22nF
D1	sot23	schottky, BAT54
IC1	sot314_2	tda8030, TDA8030
J1	usb_b	usb_b, USB_B
J2	card_read_ccm01_2251	card_read_ccm01_2251
L1	self3613	self, 10uH
R1	r1206	res, 0
R2	r1206	res, 1.5K
R3	r1206	res, 22
R4	r1206	res, 22
RR1	sip9	rres1c8r, 10K
TP1	tp0.9	test, TEST_Bar1
TP2	tp0.9	test, TEST_Bar1
TP3	tp0.9	test, TEST_Bar1
TP4	tp0.9	test, TEST_Bar1
TP5	tp0.9	test, TEST_Bar1
TP6	tp0.9	test, TEST_Bar1
TP7	tp0.9	test, TEST_Bar1
TP8	tp0.9	test, TEST_Bar1
TP9	tp0.9	test, TEST_Bar1
TP10	tp0.9	test, TEST_Bar1
TP11	tp0.9	test, TEST_Bar1
TP12	tp0.9	test, TEST_Bar1
TP13	tp0.9	test, TEST_Bar1
TP14	tp0.9	test, TEST_Bar1
TP15	tp0.9	test, TEST_Bar1
TP16	tp0.9	test, TEST_Bar1

TP17	tp0.9	test, TEST_Bar1
TP18	tp0.9	test, TEST_Bar1
TP19	tp0.9	test, TEST_Bar1
TP20	tp0.9	test, TEST_Bar1
TP21	tp0.9	test, TEST_Bar1
TP22	tp0.9	test, TEST_Bar1
TP23	tp0.9	test, TEST_Bar1
TP24	tp0.9	test, TEST_Bar1
TP25	tp0.9	test, TEST_Bar1
TP26	tp0.9	test, TEST_Bar1
TP27	tp0.9	test, TEST_Bar1
TP28	tp0.9	test, TEST_Bar1
TP29	tp0.9	test, TEST_Bar1
TP30	tp0.9	test, TEST_Bar1
TP31	tp0.9	test, TEST_Bar1
TP32	tp0.9	test, TEST_Bar1
TP33	tp0.9	test, TEST_Bar1
Y1	hc49uv	quartz, 12MHZ