APPLICATION NOTE

Smart Card Reader Application with TDA8029 Mask 04

AN01024

Philips Semiconductors

PHILIPS

Abstract

This application note describes the software implemented in TDA8029 mask 04 (TDA8029HL/C104) in order to handle a communication between a system controller and a smart card.

This mask04 can support all the asynchronous smart cards using either T=0 or T=1 protocol and some synchronous smart cards (S=9, S=10 and I2C).

The control of the TDA8029 by the host controller can be done either with a RS232 serial interface or with an I2C-bus.

© Philips Electronics N.V. 2001
All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent - or other industrial or intellectual property rights.

APPLICATION NOTE

Smart Card Reader Application with TDA8029 Mask 04

AN01024

Author(s):
Thierry LEJOSNE
Systems & Applications
Business Unit Identification – Business Line RIC
Caen - France

Keywords
TDA8029
Smart card interface
ISO 7816-3&4
E.M.V. 3.1.1

Number of pages : 56

Date : 2001/07/17

CONTENTS

1 INTRODUCTION

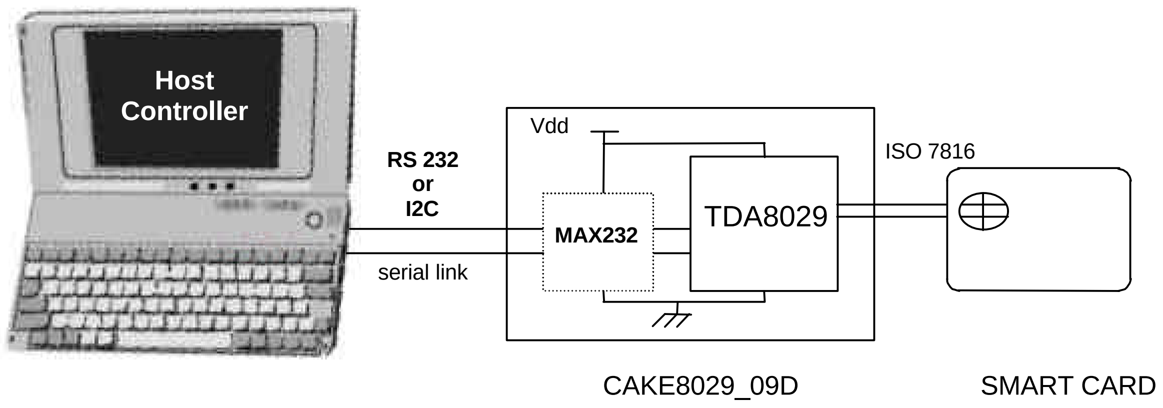
TDA8029 is a smart card coupler providing all the analogue electrical interface signals to the smart card. This coupler is able to manage asynchronous cards due to its specific ISO7816 UART and to its embedded 80C51 microcontroller core ; it can also manage synchronous cards such as I2C cards or prepaid telephone cards.

The software embedded in this device is able to support any ISO7816 asynchronous smart card (T=0 or T=1 protocol) and some synchronous cards (I2C, S9 and S10).
It completely handles the communication layer between the card and the host system.

A specific protocol called "ALPAR" has been defined on the serial interface between TDA8029 and the host system ; it uses the APDUs frame types to convey the asynchronous card commands and specific frames for the synchronous cards. A dedicated command has been added to carry TPDUs frames for T=1 protocol only.

A board has been built in order to demonstrate a communication between a smart card and a host system. When the host is a PC communicating with the TDA8029 using a RS232 link, a software called SCRTester can be used (see detailed description in Annex I, page 50).

The following diagram illustrates this application.



2 HARDWARE AND POWER MANAGEMENT

2.1 Hardware

The board CAKE 8029_09D (see Annex II, page 52) has been made to demonstrate the features of the TDA8029 with Mask 04 software.
This board is supplied under +2.7V to +6V.

In case of RSR232 host interface configuration, it may be connected to a PC by means of the serial port.
Depending on the PC which is used, the communication between the PC and the board can be fixed at different baud rates (from 4800 to 115200 baud).
The default baud rate is 38400 and it can be changed by a special command in SCRTester (set_serial_baud_rate, page 31).

In case of I2C-bus host interface configuration, it should be connected to the I2C-bus master using *SDA* and *SCL* lines and possibly *WakeUpSlave* and *SlaveI2CMute* lines.

2.2 Host Controller Interface

The TDA8029 with Mask04 software can be interfaced to the host controller by using either a RS232 serial link or an I2C-bus.

The choice of the interface is done according to the state of **P17** (i.e. pin 1 of the TDA8029) at the powering on or at the reset of the TDA8029.

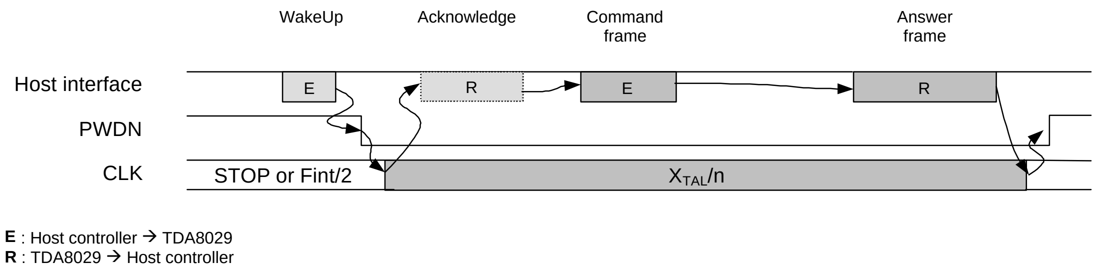
Interface	P17 (#1)	CAKE8029_09D
RS232 (see §5)	VDD or not connected	ST4 open / ST3 soldered or open
I2C (see §6)	GND	ST4 soldered / ST3 open

2.3 Power management : Energy Saving Mode

In order to benefit from the low power features of the TDA8029, the Mask 04 software implements a special management of the TDA8029 called Energy Saving Mode (ESM).

In this mode, outside an exchange of commands between the host and the TDA8029, the card clock is either switched off (level high or low) or set to Fint/2 depending on the clock stop mode described in its ATR and the microcontroller is set in power down mode. This mechanism allows a lower average current consumption for the TDA8029.

A general description of ESM mechanism can be summarised on the following figure :



In this figure, the "host interface" symbolises the link between the host controller and the TDA8029 : it may be either a RS232 link or an I2C-bus.

When the host wants to send a command frame to the TDA8029, it first wakes it up. The waking up process is different according to the type of the interface used. It will be detailed hereafter (§5.2 and §6.3).

As soon as it is able to receive and treat a command frame, the TDA8029 indicates it to the host controller using an acknowledge process. When using an I2C-bus interface, this stage may be possibly skipped (as the I2C-bus mechanism already implements an acknowledge notion).

Then, the host controller can send the complete command frame to be treated by the TDA8029.

When the TDA8029 has finished the execution of the command, it sends back the corresponding answer to the host.

When the exchange is completed, the TDA8029 goes back in power down mode after having switched the clock of the card to the lower current consumption available configuration. If the card does not support the clock stop mode or does not specify it in its ATR, the clock will be set to Fint/2. Note that a command (set_ESM_properties, page 32) allows to force a clock stop mode even if the card does not specify it. Stop the clock rather than leave it at Fint/2 provides a significant power consumption saving.

To activate this mode **P26** (i.e. pin 25 of the TDA8029) has to be connected to V_{DD} or to be left open at the powering on of the board (or at the reset), whereas connecting **P26** (pin 25) to ground will force the TDA8029 to never enter into this energy saving mode.

ESM	P26 (#25)	CAKE8029_09D
ON	VDD or not connected	ST6 open / ST5 soldered or open
OFF	GND	ST6 soldered / ST5 open

3 SOFTWARE ASPECT

This mask 04 has been developed in order to be used either in ISO7816-3 or E.M.V.3.1.1 environment.
It is compliant with erratum #3 TBD/EXE/T01 2.04, November 29th 2000.

Some specific error messages are dedicated to the EMV environment (ATR parameters not allowed).

4 PROTOCOL "ALPAR"

The communication between the host controller and the TDA8029 obeys to a protocol named ALPAR.
This protocol encapsulates the useful data of a message in an invariant frame structure and defines a dialog structure of messages exchanges.

Frame structure :

Data is exchanged between the host controller and TDA8029 in blocks, each made up of binary characters on one byte :

4 header characters
0 to 506 data characters (C-APDU or R-APDU)
1 LRC character

4 bytes	0 to 506 bytes	1 byte
Header	C-APDU or R-APDU	LRC
	Information field	

The 4 header bytes includes:

1st byte	2nd byte	3rd byte	4th byte
A 1 1 0 0 0 0 0			
↓	Data length to transmit excluding header and LRC		Command byte

A=0 ➔ Acknowledge of the frame (1st byte = 60)
A=1 ➔ Nack of the frame (message with a status error, 1st byte = E0)

LRC byte:

The LRC (Longitudinal Redundancy Check) byte is such that the exclusive-oring of all bytes including LRC is null.

4.1 General dialog structure

The host controller is the master for the transmission ; each command from the master is followed by an answer from TDA8029 including the same command byte as the input command.

However, in some cases (card insertion or extraction, a time out detection on Rx line or an automatic emergency deactivation of the card) the TDA8029 is able to initiate an exchange.

When configured in I2C-bus interface, as the TDA8029 is a pure slave, it implements a special mechanism to warn the host for these outgoing commands.

4.1.1 Successful command

System to TDA8029

60	XX XX	YY	nnnnnnnnnnnnnnnnnnn	ZZ
ACK	length	code	Data (C-APDU)	LRC

TDA8029 to System

60	UU UU	YY	mmmmmmmmmmmmmmm	TT
ACK	length	code	Data (R-APDU)	LRC

The same command byte YY is returned in the answer from TDA8029.

4.1.2 Unsuccessful command

System to TDA8029

60	XX XX	YY	nnnnnnnnnnnnnnnnnnn	ZZ
ACK	length	code	Data (C-APDU)	LRC

TDA8029 to System

E0	UU UU	YY	SS	TT
NACK	length	code	status	LRC

In that case, the status contains the error code information (see **error list**, page 27).

4.1.3 Answer with an acknowledge (power_off, idle_mode, power_down_mode)

System to TDA8029 (example : power_off)

60	00 00	4D	2D
ACK	Length	code	LRC

TDA8029 to System

60	00 00	4D	2D
ACK	Length	code	LRC

In the case where the answer is an acknowledge of the command, the TDA8029 sends back a frame with the same content of the command.

5 RS232 INTERFACE

5.1 General description

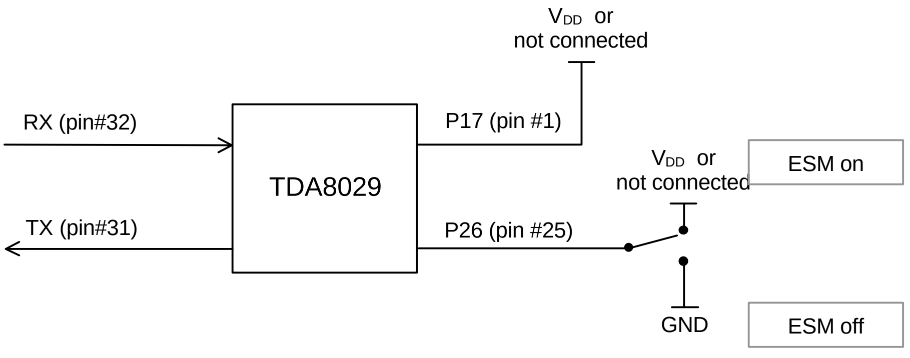
The serial interface between the TDA8029 and the host controller is a full duplex interface using the two lines RX and TX.
RX (pin 32) is used to receive data from the host controller, TX (pin 31) is used to send data to the host controller.
No flow control or supplementary line is used (no hand check).

The serial data format used is :

1 start bit
8 data bits
1 stop bit, no parity

The default baud rate is 38400 baud, but it can be changed (from 4800 to 115200 baud) by a host command set_serial_baud_rate (p 31).

In case of RS232 interface mode configuration, the TDA8029 has to be connected as follows :



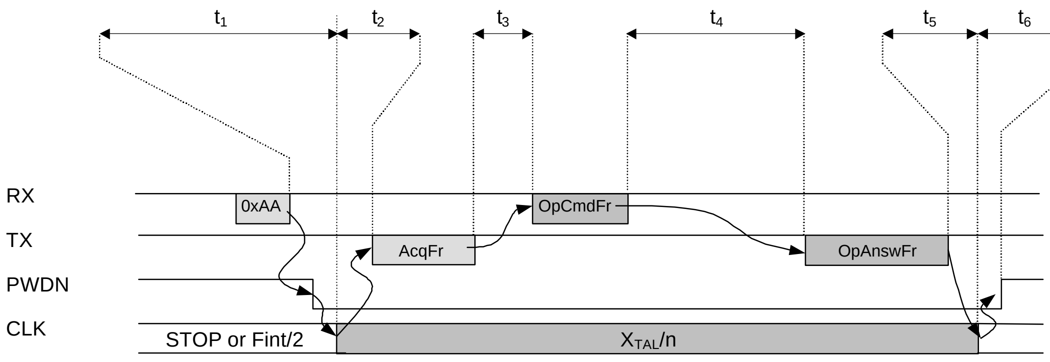
RS232 Interface configuration

5.2 Dialog structure in case of Energy Saving Mode activated

As it is explained in §2.3, the Mask 04 implements a special management of the TDA8029 for energy savings purpose. This Energy Saving Mode is activated when **P26** (i.e. pin 25) is tied to V_{DD} of left open at reset of the TDA8029.

Due to this Energy Saving Mode implementation, the serial interface is adapted as follows.

5.2.1 Communication initiated by the system controller



Communication initiated by the system controller.

To initiate a normal exchange with the TDA8029, the host has to first send a specific frame composed of only one character (0xAA) to wake up the TDA8029. When the TDA8029 is completely waked up, it sends an acknowledged frame back to the host.

System to TDA8029 (Specific waking up frame)

AA

TDA8029 to System (Acknowledgement frame)

60	00 00	BB	DB
ACK	length	code	LRC

After this synchronisation stage, the operational communication can start as described in paragraph §4.1.

System to TDA8029

60	XX XX	YY	nnnnnnnnnnnnnnnnnnnn	ZZ
ACK	length	code	Data (C-APDU)	LRC

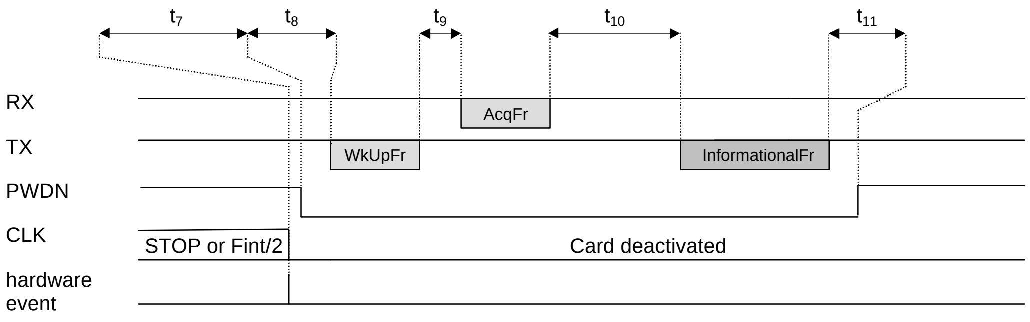
TDA8029 to System

60 or E0	XX XX	YY	nnnnnnnnnnnnnnnnnnnn	ZZ
ACK or NACK	length	code	Status or Data (R-APDU)	LRC

After having sent its complete frame, the TDA8029 returns in power down.

5.2.2 Communication initiated by the TDA8029

Any card event (extraction or insertion, overcurrent on VCC or RST, overheating) will wake up the TDA8029.
In that case, this is the TDA8029 which initiates the communication.



Communication initiated by the TDA8029.
(in this example, card deactivated due to a hardware event)

It first sends the following specific frame to warn the host that it wants to send data. This frame is the same than the one used to acknowledge a received waking-up character (§5.2.1) except that the length in that case is equal to one (as it is equal to zero in case of a normal acknowledge frame).

TDA8029 to System

60	00 01	BB	01	DB
ACK	length	code	data	LRC

After that, the TDA8029 waits until the host replicates this frame.

System to TDA8029

60	00 01	BB	01	DB
ACK	length	code	data	LRC

Then, the TDA8029 sends the informational frame and returns in power down mode.

TDA8029 to System

60 or E0	XX XX	YY	nnnnnnn	ZZ
ACK or NACK	length	code	Data	LRC

For instance, a card insertion will generate the following informational frame :

TDA8029 to System

60	00 01	A0	01	C0
ACK	Length	code	data	LRC

whereas in the case of a card extraction , the frame will be :

TDA8029 to System

60	00 01	A0	00	C1
ACK	Length	code	data	LRC

5.2.3 Abnormal communication process

Three different cases can be met :

- time out detected on Rx line (more than 10 ms between the leading edge of two characters inside the command frame sent by the host controller). As soon as the time out is detected, the TDA8029 sends back an error frame :

System to TDA8029	0xAA	
TDA8029 to system	0x60 0x00 0x00 0xBB 0xDB	
System to TDA8029	ACK Length Code (Parameters) LRC	(*erroneous frame*)
TDA8029 to system	NACK 0x00 0x01 Code 0xFF LRC	

  Then the TDA8029 goes back to power down mode.

- card extraction detected during a card IO card session : if the TDA8029 detects a card extraction as it is processing an APDU with the card, it returns two consecutive messages back to the host controller

System to TDA8029	0xAA	
TDA8029 to system	0x60 0x00 0x00 0xBB 0xDB	
System to TDA8029	ACK Length 0x00 Parameters LRC	(*C-APDU*)
TDA8029 to system	NACK 0x00 0x01 0x00 0xC0 LRC	(*card absent*)
TDA8029 to system	ACK 0x00 0x01 0xA0 0x00 LRC	(*card extraction*)

  Then the TDA8029 goes back to power down mode.

- unexpected reception detected during a communication process ; the TDA has not finished to process a received command frame (it has not sent completely its answer frame while the host controller sends a new command frame) : in that case, the TDA8029 sends the correct answer to the first received command and then sends the error frame informing the host controller that it has lost at least a command frame.

System to TDA8029	0xAA
TDA8029 to system	0x60 0x00 0x00 0xBB 0xDB
System to TDA8029	ACK Length Code1 (Parameters) LRC
System to TDA8029	ACK Length Code2 (Parameters) LRC
TDA8029 to system	ACK or NACK Length Code1 (Parameters) LRC
TDA8029 to system	NACK 0x00 0x01 Code2 0xF1 LRC

  Then the TDA8029 goes back to power down mode.

5.2.4 Timing considerations

The following timings are referenced in the figures page 14 and 15.

- t_1 : Waking up and clock switching time (typically 550 µs).
- t_2 : TDA8029 reaction time (typically 33 µs).
- t_3 : Host-dependant reaction time. (no limit).
- t_4 : Process time (depends on the type of the command frame).
- t_5 : Clock switching time (typically 80 µs).
- t_6 : Power Down setting time (typically 0.3 ms).
- t_7 : Waking up and clock switching time (typically 550 µs).
- t_8 : TDA8029 reaction time (typically 630 µs).
- t_9 : Host-dependant reaction time. (no limit).
- t_{10}: TDA8029 reaction time (typically 630 µs).
- t_{11}: Power Down setting time (typically 0.3 ms).

6 I2C-BUS INTERFACE

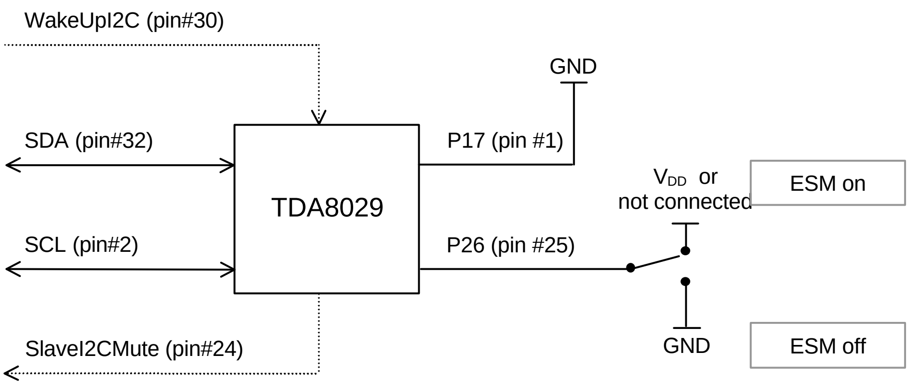
6.1 General description

As specified in the I2C-bus specification, only two lines may be used to manage the serial link between the TDA8029 and the system controller :

- a serial data line (SDA), has to be connected to RX (pin 32 of the TDA8029)
- and a serial clock line (SCL), has to be connected to P16 (pin 2 of the TDA8029).

In addition to I2C specification, two other lines can be used to manage Energy Saving Mode mechanism :

- WakeUpSlave, line used to wake up the TDA8029 before sending an I2C frame to it has to be connected to INT1 (pin 30 of the TDA8029)
- SlaveI2CMute, line used by the TDA8029 to indicate to the host controller either that it is ready to receive a command frame or to send the corresponding answer, or to signal an hardware event has to be connected to P27 (pin 24 of the TDA8029).



I2C-bus Interface configuration

In fact, ESM may be used even with a pure 2 lines I2C-bus.
In that case, the pin 30 of the TDA8029 has to be connected together with SDA line.

I2C Mode	ESM	RX (#32)	P26 (#26)	INT1 (#30)	P27 (#24)
2 lines	**OFF**	SDA	GND	nu	nu
2 lines	**ON**	SDA	VDD	SDA	nu
4 lines	**ON**	SDA	VDD	WakeUpSlave	SlaveI2CMute

As the system controller is the I2C-bus master, it will initiate all the exchanges. Each command from the master is followed by an answer from TDA8029.

Normally, as it is a I2C-bus slave, the TDA8029 can not warn the host controller by means of the bus when an hardware event happens (abnormal deactivation of the card, movement detection, …). The line SlaveI2CMute can be used for that.
When such an event occurs, the TDA8029 falls down the SlaveI2CMute line so that the host controller can known that something has happened on the reader side.

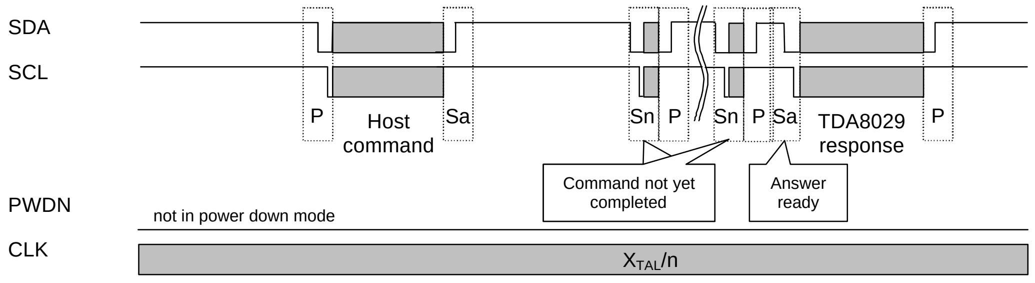
The host controller can send a GetReaderStatus (see page 31) command frame to receive details on the current state of the TDA8029.
This particularity may be deactivated by means of the SetESMProperties command (see page 32). In that case, the TDA8029 does not inform the host controller when a specific event happens on the card. Of course, the TDA8029 takes in charge the security of the card and automatically deactivates it if needed.

6.2 Energy Saving Mode deactivated

When the TDA8029 is not configured in Energy Saving Mode, it is able to accept a command sent by the host controller as soon as it has finished to handle the previous one (no waking up delay).

On the other hand, after having received a complete command frame from the host, the TDA8029 will need a variable delay time to achieve the related task before to send back an answer to the host.
This time depends on the kind of the command, the kind of the card (baudrate, CWT, BWT, …), the length of the card exchange, and so on…
During this execution time, the TDA8029 will be mute, i.-e. it will not acknowledge any incoming messages from the host controller.

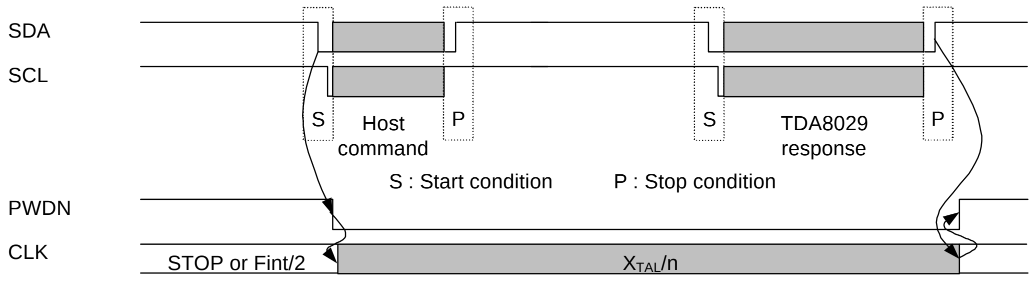
Once the command message is processed, the TDA8029 will be available to give its answer when the host controller will address it.



Sn : Start condition not acknowledged
Sa : Start condition acknowledged by the TDA8029
P : Stop condition

6.3 Energy Saving Mode activated

When used in Energy Saving Mode, the goal is that the TDA8029 stays in power down mode outside an I2C-bus exchange with it.

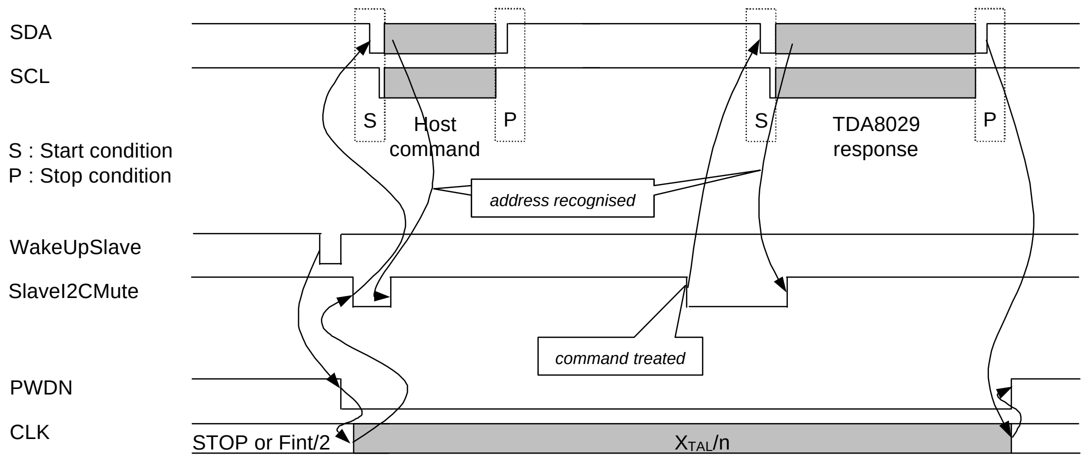


6.3.1 4 lines I2C-bus

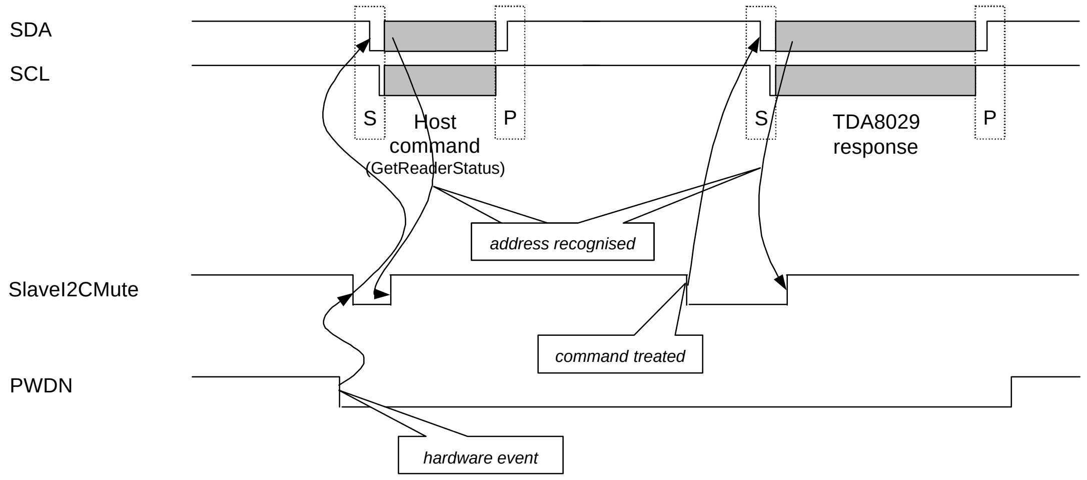
If the host controller sends directly an I2C frame to the TDA8029 as it was asleep, it will not be able to acknowledge its address (due to waking-up delay).

Consequently, the lines WakeUpSlave and SlaveI2CMute are used to manage the complete exchange :

1 - Before sending a frame to the TDA8029, the host controller wakes it up with a negative pulse on WakeUpSlave line (minimum duration of 0.8µs)
2 - As soon as it is completely waked up, the SlaveI2CMute line falls down,
3 - The host controller can now send the I2C write command frame. The SlaveI2CMute line goes up after the TDA8029 has recognised its I2C address (0x50)
4 - Once the command frame is received, the TDA8029 processes it. When it is ready to give corresponding results to the host controller, the SlaveI2CMute line falls down again
5 - The host controller can now send I2C read command frame. The SlaveI2CMute line goes up after the TDA8029 has recognised its I2C address (0x51)
6 - Once the answer has been totally read by the host controller, the TDA8029 returns in power down mode until the next host controller exchange.



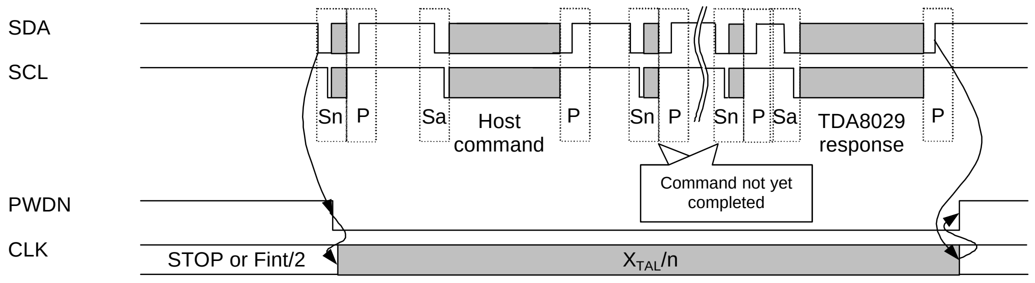
Moreover, in that configuration, if the TDA8029 has to warn the host controller to inform it about an hardware event, it can do it by falling down the SlaveI2CMute line outside a normal exchange. Then, the host sends a command with GetReaderStatus (see page 31) opcode to get detailed information from the TDA8029. In that case, the host controller should not use the WakeUpSlave line as the TDA8029 is already waked up.



Remark : The main advantage of this mode (4 lines I2C-bus) is that the current consumption of the TDA8029 is completely optimised.
When the I2C-bus master addresses to other slave than the TDA8029, the latter is not waked up (this is not SDA or SCL line which wakes it up but the dedicated WakeUpSlave line).

6.3.2 2 lines I2C-bus

As described in the table in §6.1, the Energy Saving Mode can be used even with a host controller using a pure I2C-bus interface, without the two additional lines *WakeUpSlave* and *SlaveI2CMute*.
In that case, the SDA line has to be connected on pin 30 of the TDA8029 in addition to pin 32.
Thus, when the TDA8029 is asleep, every frame on the I2C-bus wakes it up ; even if the frame is not addressed to the TDA8029. That is the main drawback (according to the current consumption) of this configuration.
As soon as it recognises its address on the I2C-bus, the TDA8029 acknowledges it and then the normal exchange can go.
The embedded microcontroller needs a delay time before to come back completely operational when waked up by a I2C-bus frame. During this period, the I2C command will not be acked by the TDA8029 and the host has to try again until its command will be correctly acked.



Sn : Start condition not acknowledged
Sa : Start condition acknowledged by the TDA8029
P : Stop condition

6.4 Data link layer

I2C-bus slave :
Slave address : 0x50

Frequency :
Maximum SCL frequency : # 60 kHz

Clock synchronising :
As an I2C-bus slave, the TDA8029 can slow down the bus clock by extending each clock low period. The speed of any I2C-bus master is thereby adapted to the internal operating rate of the TDA8029.
This synchronising mechanism is also called clock stretching.

6.5 I2C transactions

The I2c transactions use the protocol 'ALPAR' described previously.

6.5.1 I2C write command message

The host system sends the following I2C message structure to the TDA8029 :

System Controller **TDA8029**

1. $\xrightarrow{\text{WRITE SLAVE ADDRESS STAGE}}$

   'I2C START CONDITION', 50h

2. $\xrightarrow{\text{PROTOCOL 'ALPAR' STAGE}}$

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

3. $\xrightarrow{\text{STOP STAGE}}$

   'I2C STOP CONDITION'

6.5.2 I2C read command message, normal answer

In this case, the I2C message structure is:

System Controller **TDA8029**

1. $\xrightarrow{\text{READ SLAVE ADDRESS STAGE}}$

   'I2C START CONDITION', 51h

2. $\xleftarrow{\text{PROTOCOL 'ALPAR' STAGE}}$

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

3. STOP STAGE →

‘I2C STOP CONDITION’

The pattern 60h indicates a normal answer from the TDA8029.

6.5.3 I2C read message, error answer.

The I2C message structure is shown below:

System Controller **TDA8029**

1. READ SLAVE ADDRESS STAGE →

‘I2C START CONDITION’, 51h

2. ← PROTOCOL ‘ALPAR’ STAGE

E0h, 00h, 01h, Opcode, Error Code, LRC

3. STOP STAGE →

The pattern **E0h** indicates an error answer from the TDA8029.

6.5.4 Examples of I2C transactions

This table below shows some I2C transactions between the host system and the TDA8029. The data sent by the TDA8029 are in **bold** :

Opcode	**I2C write message**	**I2C read messages**
Check card presence	S, 50, **60, 00, 00, 09, LRC**, P	If card present: S, 51, **60, 00, 01, 09, 01**, **LRC**, P If card absent: S, 51, **60, 00, 01, 09, 00, LRC**, P
Mask Number	S, 50, **60, 00, 00, 0A, LRC**, P	S, 51, **60, 00, 0E, 0A, data, LRC**, P
Power up card 3v, EMV	S, 50, **60, 00, 01, 6D, 01, LRC**, P	If card powered up successfully: S, 51, **60, 00, 0F, 6D, ATR, LRC,** P If card absent, error message: S, 51, **E0, 00, 01, 6D, C0**, **LRC,** P
Power up card 5v, ISO	S, 50, **60, 00, 01, 6E, 00, LRC**, P	If card powered up successfully: S, 51, **60, 00, 0D, 6E, ATR, LRC**, P If card absent, error message: S, 51, **E0, 00, 01, 6E, C0, LRC**, P
Power off card	S, 50, **60, 00, 00, 4D, LRC**, P	S, 50, **60, 00, 00, 4D, LRC**, P

Note: In the I2C messages given above:
- ‘S’ stands for I2C start condition
- ‘P’ means I2C stop condition.

7 COMMAND BYTES

The following command bytes are available (listed in numerical order) :

Command	***Code***	***Answer from reader***	***(page)***
card_command (APDU)	00_H	Card response (APDU) or error message	*(36)*
process_T=1_command	01_H	T=1 frame or error message	*(36)*
write_I2C	02_H	Acknowledge or error message	*(45)*
read_S9	03_H	Data read from the card or error message	*(41)*
read_S9_protection	04_H	Data read from the card or error message	*(41)*
write_S9_protected	05_H	Acknowledge or error message	*(41)*
write_S9_unprotected	06_H	Acknowledge or error message	*(42)*
verify_pin_S9	07_H	Acknowledge or error message	*(42)*
compare_S9	08_H	Acknowledge or error message	*(42)*
check_pres_card	09_H	Indication of the card presence	*(31)*
send_num_mask	$0A_H$	1 parameter giving the mask number	*(31)*
set_card_baud_rate	$0B_H$	Acknowledge	*(38)*
IFS_request	$0C_H$	Acknowledge or error message	*(37)*
set_serial_baud_rate	$0D_H$	Acknowledge or error message	*(31)*
negotiate (PTS)	10_H	Acknowledge or error message	*(37)*
set_clock_card	11_H	Acknowledge or error message	*(38)*
read_I2C	12_H	Data read from the card or error message	*(44)*
read_2C_extended	13_H	Data read from the card or error message	*(44)*
read_current_I2C	23_H	Data read from the card or error message	*(44)*
power_off	$4D_H$	Acknowledge	*(36)*
power_up_iso	69_H	ATR from the card or error message	*(36)*
power_up_S9	$6B_H$	ATR from the card or error message	*(41)*
power_up_I2C	$6C_H$	Acknowledge or error message	*(44)*
power_up_3V	$6D_H$	ATR from the card or error message	*(35)*
power_up_5V	$6E_H$	ATR from the card or error message	*(35)*
idle_mode (clock stop low)	A2H	Acknowledge	*(33)*
power_down_mode	A3H	Acknowledge	*(34)*
idle_mode (clock stop high)	A4H	Acknowledge	*(33)*
set_NAD	$A5_H$	Acknowledge or error message	*(39)*
get_card_param	$A6_H$	Fi, Di, CLK, T of the card in use or error message	*(40)*

get_reader_status	AA_H	Information about the current state of the reader	*(31)*
host_ready	BB_H	Informational message	*(15)*
set_ESM_properties	BC_H	Acknowledge	*(32)*
power_up_S10	$C1_H$	ATR from the card or error message	*(43)*
process_S10	$C2_H$	Data read from the card or error message (read operation) Acknowledge or error message (write operation)	*(43)*
read_IO	CE_H	Value on the IO pins	*(46)*
set_IO	CF_H	Acknowledge	*(46)*

Outgoing commands (only) :

	Code	*Parameter*
Card_take_off	$A0_H$	00_H
Card_insertion	$A0_H$	01_H

These commands are sent as soon as a card is inserted or extracted without any command coming from the system. These commands use the same operating code but the extra parameter gives the additional information.

These outgoing commands are sent only when the host is waiting for a reply or is in stand by; when the card is extracted whereas the host is sending a frame to TDA8029, the card_take_off message will be sent from TDA8029 only when it has received the complete frame coming from the host controller. This system prevents any conflict on the serial line.

	Code	*Parameter*
Card deactivated	XX_H	$A1_H$
	The card is deactivated due to a hardware problem (short on Vcc, overcurrent)	
Time out	XX_H	FF_H
	Time out problem on (TDA8029) Rx line This command is used in order to warn the host controller that the last communication has broken down (time out problem) so that the Rx line of TDA8029 does not remain blocked. The time out condition is a silence greater than 10 ms in the host command frame.	
Frame lost	XX_H	$F1_H$
	An unexpected host controller command frame has been received by the TDA8029 while it was busy to process a previous command frame.	

In these three commands, the code value is the previous code value used during a normal exchange.

8 ERROR LIST

The error list gives the status code identification and a brief signification of the status error code.

8.1 Exhaustive list of possible error code

Status code	*Meaning*
08_H	Length of the data buffer too short
$0A_H$	3 consecutive errors from the card in T=1 protocol
20_H	Wrong APDU
21_H	Too short APDU
22_H	Card mute now (during T=1 exchange)
24_H	Bad NAD
25_H	Bad LRC
26_H	Resynchronized
27_H	Chain aborted
28_H	Bad PCB
29_H	Overflow from card
30_H	Non negotiable mode (TA2 present)
31_H	Protocol is neither T=0 nor T=1 (negotiate command)
32_H	T=1 is not accepted (negotiate command)
33_H	PPS answer is different from PPS request
34_H	Error on PCK (negotiate command)
35_H	Bad parameter in command
38_H	TB3 absent
39_H	PPS not accepted (no answer from card)
$3B_H$	Early answer of the card during the activation
55_H	Unknown command
80_H	Card mute (after power on)
81_H	Time out (waiting time exceeded)
83_H	4 parity errors in reception
84_H	4 parity errors in transmission
86_H	Bad FiDi
88_H	ATR duration greater than 19200 etus (E.M.V.)
89_H	CWI not supported (E.M.V.)
$8A_H$	BWI not supported (E.M.V.)
$8B_H$	WI (Work waiting time) not supported (E.M.V.)
$8C_H$	TC3 not accepted (E.M.V.)
$8D_H$	Parity error during ATR
90_H	3 consecutive parity errors in T=1 protocol
91_H	SW1 different from 6X or 9X
92_H	Specific mode byte TA2 with b5 byte=1
93_H	TB1 absent during a cold reset (E.M.V.)
94_H	TB1different from 00 during a cold reset (E.M.V.)
95_H	IFSC<10H or IFSC=FFH
96_H	Wrong TDi
97_H	TB2 is present in the ATR (E.M.V.)

98_H	TC1 is not compatible with CWT
$9B_H$	Not T=1 card
$A0_H$	Procedure byte error
$A1_H$	Card deactivated due to a hardware problem
$B0_H$	Writing attempt in a protected byte (S9 cards)
$B1_H$	Pin Code error (S9 cards)
$B2_H$	Writing error (S9 cards)
$B3_H$	Too much data requested in a reading operation (S9 cards)
$B4_H$	Error counter protected (S9 cards)
$B5_H$	Writing attempt without Pin Code verification (S9 cards)
$B6_H$	Protected bit already set (S9 cards)
$B7_H$	Verify Pin Code error (S9 cards)
$C0_H$	Card absent
$C1_H$	I/O line locked while the TDA8029 attempts to access to an I2C or S10 card
$C3_H$	Checksum error
$C4_H$	TS is neither 3B nor 3F
$C6_H$	ATR not supported
$C7_H$	VPP is not supported
CC_H	No acknowledge from the I2C synchronous card
CD_H	Generic error during an exchange with an I2C synchronous card
$E1_H$	Card clock frequency not accepted (after a set_clock_card command)
$E2_H$	UART overflow
$E3_H$	Supply voltage drop-off
$E4_H$	Temperature alarm
$E9_H$	Framing error
$F0_H$	Serial LRC error
$F1_H$	At least one command frame has been lost
FF_H	Serial time out

8.2 Error code for each command

COMMAND		POSSIBLE RETURNED ERROR CODE
Power UP 3V, 5V		31h, 35h, 38h, 3Bh, 80h, 85h, 86h, 88h, 89h, 8Ah, 8Bh, 8Ch, 8Dh, 92h, 93h, 94h, 95h, 96h, 97h, 98h, C0h, C3h, C4h, C6h, C7h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
Power up in ISO mode		31h, 35h, 3Bh, 80h, 96h, C0h, C3h, C4h, C6h, C7h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
Card Command	T=0	08h, 20h, 21h, A1h, 81h 83h, 84h, 91h, A0h, C0h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
	T=1	08h, 22h, 24h, 25h, 26h, 27h, 28h, 29h, A1h, 83h, 90h, C0h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
Negotiate		30h, 31h, 33h, 34h, 35h, 39h, A1h, C0h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
Set Clock Card		C0h, E1h, F0h, F1h, FFh
Set card baud rate		86h, C0h, F0h, F1h, FFh
Set NAD		24h, F0h, F1h, FFh
Get card parameters		A1h, C0h, F0h, F1h, FFh
IFS request		0Ah, A1h, 9Bh, C0h, E2h, E3h, E4h, E9h, F0h, F1h, FFh
Send mask number		F0h, F1h, FFh
Check presence card		F0h, F1h, FFh
Set serial baud rate		55h, F0h, F1h, FFh
Power off		F0h, F1h, FFh
Set ESM properties		F0h, F1h, FFh
Idle mode clock stop low and high		55h, F0h, F1h, FFh
Get reader status		F0h, F1h, FFh
Power down mode		55h, F0h, F1h, FFh
Read IO and Set IO		F0h, F1h, FFh
Power up I2C		C0h, F0h, F1h, FFh
Read I2C		C0h, C1h, CCh, CDh, F0h, F1h, FFh
Read I2C extended		C0h, C1h, CCh, CDh, F0h, F1h, FFh
Read current I2C		C0h, C1h, CCh, CDh, F0h, F1h, FFh
Write I2C		55h, C0h, C1h, CCh, CDh, F0h, F1h, FFh
Power up S9		C0h, F0h, F1h, FFh
Read S9		B3h, C0h, F0h, F1h, FFh

COMMAND	POSSIBLE RETURNED ERROR CODE
Read S9 protection	B3h, C0h, F0h, F1h, FFh
Write S9 unprotected	55h, B0h, B2h, B5h, C0h, F0h, F1h, FFh
Write S9 protected	55h, B0h, B2h, B5h, C0h, F0h, F1h, FFh
Verify PIN S9	55h, B1h, B4h, B7h, C0h, F0h, F1h, FFh
Compare S9	55h, B6h, B7h, C0h, F0h, F1h, FFh
Power up S10	C0h, F0h, F1h, FFh
Process command S10	C0h, C1h, F0h, F1h, FFh

9 COMMANDS DESCRIPTION

9.1 General commands

9.1.1 send_num_mask

This command is used to identify the software version which is masked in TDA8029 ROM.

For example the current software will be coded as : “04 Release 1.8” (14 ASCII characters)
System to TDA8029 : 60 00 00 0A 6A
TDA8029 to System : 60 00 0E 0A 30 33 20 52 65 6C 65 61 73 65 20 31 2E 38 0E

9.1.2 check_card_presence

This command is used to check the presence of a card.

System to TDA8029 : 60 00 00 09 69
TDA8029 to System : 60 00 01 09 PRES LRC

Where PRES indicates the presence of a card (00 if there is no card, 01 if a card is present).

9.1.3 get_reader_status

This command is used to check the status of the reader.

System to TDA8029 : 60 00 00 AA CA
TDA8029 to System : 60 00 01 AA STATUS LRC

Where the latched state of the TDA8029 is given in STATUS byte.

nu	nu	nu	nu	SUPL	PROTL	PTL	PRES

PRES	card presence (0 : card absent, 1 : card present)
PTL	overheating detection
PROTL	default detected on card reader (protection on VCC or RST)
SUPL	supervisor activation

The byte STATUS is cleared (except PRES bit) after having launch this command.

9.1.4 set_serial_baud_rate

This command is used for changing the baud rate onto the serial link between the host and the interface card. The default value is set to 38400 baud.

A parameter has to be transmitted in order to choose the baud rate :

```
System to TDA8029 :   60   00 01   0D   PAR   LRC
TDA8029 to System :   60   00 00   0D   6D
```

Baud rate (Baud)	**Parameter**
4800	00
9600	01
19200	02
38400	03
57600	04
76800	05
115200	06

After a baud rate change, the new value takes place for the next command sent by the host.

Note : This command can be used only when the TDA8029 is configured in RS232 interface mode. If launched when configured in I2C-bus interface mode, an *UNKNOWN_COMMAND* error will be returned by the TDA8029.

9.1.5 set_ESM_properties

This command is used to fix the behaviour of the TDA8029 if the Energy Saving Mode is activated.
By default, the clock stop information contained in the ATR of the activated card is used during the session with this card to set the clock when the TDA8029 enters in power down mode. Thus, if the card does not explicitly indicate that it supports clock stop mode (High or Low), the card clock will be set to Fint/2.
To save even more energy during these periods, the clock mode can be forced using this command.

```
System to TDA8029 :   60   00 02   BC   STOP STATE EVENT   LRC
TDA8029 to System :   60   00 00   BC   DC
```

Where	STOP	indicates the clock stop request **00** to set the clock according to card indications *(default behaviour)* **01** to force the clock stopping
	STATE	is the clock stop level if requested (when STOP = 01) **00** to stop clock LOW **01** to stop clock HIGH
	EVENT	is used to force the TDA8029 to not warn the host controller with a outgoing command (card movement, hardware problem, …). If configured in RS232 interface mode, the TDA8029 will not send a frame to the host controller. If configured in I2C-bus interface mode, the TDA8029 will not falls down the SalveI2CMute line. **00** : the TDA8029 warns the host controller *(default behaviour)* **01** : the TDA8029 does not warn the host controller

Note 1 : This command can be used only when the Energy Saving Mode is activated. If launched when the Energy Saving Mode is not activated, an *UNKNOWN_COMMAND* error will be returned by the TDA8029.

Note 2 : Once this command has been launched to the TDA8029, all the further activations of cards will follow the behaviour defined within this command. One has to use this command again to change the behaviour, e.g. to come back to a clock at Fint/2. Furthermore, even if this command is used to force a clock mode, when a card with defined clock stop conditions is encountered, the clock stop mode indicated in the card's ATR will be used.

9.1.6 time_out

This command is sent from TDA8029 to the host controller if, during a transmission from the host controller to TDA8029, the time interval between 2 characters exceeds 10ms. This timing is calculated between each character of a frame, starts after the first character, and is disabled after the last character of the frame. This feature has been implemented in order to avoid any blocking of the transmission line between the host controller and TDA8029.

```
TDA8029 to System :   E0   00 01   6F   FF   71
```

Note : This command may be used by the TDA8029 only when it is configured in RS232 interface mode.

9.1.7 idle_mode (clock stop low)

This command is used to set the controller in idle mode. The card, if activated, has its clock (CLK) set to low level but is still active.
Any command from the host on the serial line will wake up the device.

```
System to TDA8029 :   60   00 00   A2   C2
TDA8029 to System :   60   00 00   A2   C2
```

Note : This command can be used only when the Energy Saving Mode is not activated. If launched when the Energy Saving Mode is activated, an *UNKNOWN_COMMAND* error will be returned by the TDA8029.

9.1.8 idle_mode (clock stop high)

This command is used to set the controller in idle mode. The card, if activated, has its clock (CLK) set to high level but is still active.
Any command from the host on the serial line will wake up the device.

```
System to TDA8029 :   60   00 00   A4   C4
TDA8029 to System :   60   00 00   A4   C4
```

Note : This command can be used only when the Energy Saving Mode is not activated. If launched when the Energy Saving Mode is activated, an *UNKNOWN_COMMAND* error will be returned by the TDA8029.

9.1.9 power_down_mode

This command is used to set the controller in power down mode; if the card is active, it is then deactivated. Exiting this mode is possible with a hardware reset of TDA8029 or an external interruption (INT0, INT1 or Rx).

```
System to TDA8029 :   60   00 00   A3   C3
TDA8029 to System :   60   00 00   A3   C3
```

Note : This command can be used only when the Energy Saving Mode is not activated. If launched when the Energy Saving Mode is activated, an *UNKNOWN_COMMAND* error will be returned by the TDA8029.

9.2 Asynchronous card related commands

9.2.1 power_up commands

There are three different power up commands (3V, 5V or ISO). Two of them (power_up_3V and power_up_5V) have to be followed by a parameter :

- 00_H indicates that all the parameters of the ATR of the card compliant with ISO7816-3 will be taken into account.
- 01_H indicates that only the ATR of cards whose parameters are inside the E.M.V.3.1.1 specification scope will be taken into account; cards having an ATR which does not comply with EMV3.1.1 requirements will be rejected.

9.2.1.1 power_up_3V

This command allows to activate the card at a VCC of 3V. All the signals going to the card will be referenced to this VCC=3V.
An activation sequence is processed following the ISO7816-3 normalisation (VCC is rising, I/O is enabled, CLK is started, and RST is processed). If the card answers to this command, the answer will content all the ATR parameters; these parameters are memorised in TDA8029 and will be taken into account during the whole card session (till the card is deactivated or till a warm reset is processed). The structure of the answer is the following:

System to TDA8029

60	00 01	6E	01	0F
ACK	length	code	EMV	LRC

TDA8029 to System

60	XX XX	6E	nnnnnnnnnnnnnnnnnnn	ZZ
ACK	length	code	ATR parameters	LRC

If the card is in specific mode, TDA8029 will process the next command directly using the new interface parameters of this specific mode. If the card proposes a different Fi/Di in the ATR than the default value (Fi/Di=372), it is up to the application to make a PPS command by using the negotiate command. If the card proposes 2 different protocols in its ATR, it is up to the application to make a PPS command by using the negotiate command.
If the card does not answer to the reset, a status giving an error code is returned to the application.

In the case of EMV compliant power up, if the card is using T=1 protocol, just after having received the ATR, TDA8029 sends an IFSD request to the card indicating that the reader can manage a data buffer of 254 bytes (FEH).

The power_up_3V command can be used to generate a warm reset if the card is already activated.

9.2.1.2 power_up_5V

This command allows to activate the card at a VCC of 5V. Every signal going to the card will be referenced to this VCC=5V.
See power_up_3V for the other characteristics.

9.2.1.3 power_up_iso

This command does not need any argument. The principle consists to activate the card as described in ISO 7816-3 :

- attempt to activate the card at a VCC of 3V, if the cards answers correctly and if it indicates in its ATR that it is a class A or a class AB card (TAi with T=15), then the command is finished and the ATR is returned to the host,
- if in the previous stage, the card did not answer correctly or did not specify in its ATR that it was a class A or a class AB card, a new activation of the card is launched at 5V. If the card does not answer to the reset, a status giving an error code is returned to the application, otherwise the answer contains all the parameters of the card.

See power_up_3V for the other characteristics (when parameter of the command is ISO, not EMV).

9.2.2 power_off

This command is used to deactivate the card whatever it has been activated for 3V or 5V operation. A deactivation sequence is processed following the ISO 7816-3 normalization in about 100µs.

```
System to TDA8029 :   60   00 00   4D   2D
TDA8029 to System :   60   00 00   4D   2D
```

9.2.3 card_command (APDU)

This command is used to transmit card commands under APDU format from system to TDA8029 whatever T=0 or T=1 protocol are used. Short or extended commands (see limitations in §8.1) can be used.
An answer to such a command is also made in APDU format from TDA8029 to the system.

Example :
```
System to TDA8029 :   60   00 07   00   00 A4 00 00 02 4F 00   8E
TDA8029 to System :   60   00 02   00   90 00   F2
```

9.2.4 process_T=1_command

This command may be used if the application layer provides the complete T=1 frame including prologue, information and epilogue fields. If it is not the case, the above `card_command` opcode shall be used.
This command is used from the application layer in order to send a complete T=1 frame to the card. This command includes the specific framing used in T=1 protocol (Prologue Field, Information Field, Epilogue Field) and will be sent transparently to the card. The answer from the card will be sent as a complete T=1 frame to the application layer. The internal timing of a block (Character Waiting Time) will be handled by TDA8029. The block Waiting Time will also be controlled by TDA8029. In case of Waiting Time Extension request (WTX) from the card, it will be taken into account by the TDA8029.

System to TDA8029	60	XX XX	01	NAD PCB LEN A_1 A_2 …… A_N EDC	LRC
TDA8029 to System	60	00 06	01	NAD PCB LEN SW1 SW2 EDC	LRC

Where	A1 A2…..An	is information field sent to the card
	XX XX	is the length of the frame from NAD to EDC

In case of chaining :

System to TDA8029	60	00 XX	01	NAD 20 LEN A_1 A_2 …… A_N EDC	LRC
TDA8029 to System	60	00 04	01	NAD 90 00 EDC	LRC
System to TDA8029	60	00 YY	01	NAD 40 LEN A_{N+1} A_{N+2} … … A_Z EDC	LRC
TDA8029 to System	60	00 ZZ	01	NAD PCB LEN D_1 D_2 … … D_N EDC	LRC

9.2.5 negotiate

This command is used to make a PPS (Protocol and Parameter Selection) to the card, if in its ATR the card proposes a different Fi/Di or 2 different protocols. By using this command a PPS will be made to the card with the Fi or Di and protocol type entered as a parameter (PP). It is up to the host to make the correct Fi/Di submission to the card.

Example :

System to TDA8029 :	60	00 02	10	PP FD	LRC
TDA8029 to System :	60	00 00	10	70	

Where FD is the ratio Fi/Di given by TA1 parameter of the ATR and PP is the protocol to be used.

If the command is acknowledged, any subsequent exchanges between the card and TDA8029 will be made by using the new parameters.

9.2.6 IFSD request

This command is used to send a S(IFS request) block to the card indicating the maximum length of information field of blocks which can be received by the interface device in T=1 protocol. The initial size following the answer to reset is 32 bytes and this size shall be used throughout the rest of the card session or until a new value is negotiated by the terminal by sending a S(IFS request) block to the card.
In EMV mode, the IFSD size is automatically negotiated to 254 just after the ATR has been received.

System to TDA8029 :	60	00 01	0C	PAR	LRC
TDA8029 to System :	60	00 00	0C	6C	

Where PAR is the IFSD size.

9.2.7 set_clock_card

This command is used for changing the card clock frequency. The default value is set to FXTAL/4 which is 3.68625 MHz.
A parameter has to be transmitted in order to choose the card clock frequency:

System to TDA8029 : 60 00 01 11 PAR LRC

Frequency	Parameter
Fxtal =14.745MHz	00
Fxtal/2=7.37MHz	02
Fxtal/4=3.68MHz	04
Fxtal/8=1.84MHz	06

After a card clock frequency change, all the waiting times are internally set to the new value.

Before applying the requested clock, the compatibility of the frequency with the current Fi used by the card is checked as described in ISO7816-3. For example, if the card has answered in its ATR a Fi parameter of 372 or 558 (fmax $\leq$ 6MHz), a change of the card clock frequency to Fxtal (14.745MHz) or Fxtal/2 (7.37MHz) will not be processed and an error status will be sent to the application.

9.2.8 card_take_off and card_insertion

These two commands are sent directly to the system processor as soon as a card extraction or insertion has occurred.

TDA8029 to System :	60 00 01 A0 00 C1	for a card extraction
	60 00 01 A0 01 C0	for a card insertion.

9.2.9 set_card_baud_rate

This command is used mainly for cards which are not fully ISO 7816-3 compliant with specific and negotiable modes. As a matter of fact some cards are in specific mode but they do not give TA2 parameter in their answer to reset. So the UART has to be set to the right baud rate by means of this specific command which programs the baud rate. For non ISO baud rates there is a possibility to increase the capability of the reader by setting the bit CKU which divides by 2 the number of clock cycles of the etu and thus doubles the baud rate of the ISO UART.

Example :
System to TDA8029 : 60 00 02 0B XX CKU LRC
TDA8029 to System : 60 00 00 0B LRC

Where XX is the value of FiDi
if CKU=0, the baud rate is defined by FiDi
if CKU=1, the baud rate is 2 * the baud rate is defined by FiDi

For an etu of 372 clock cycles : XX=FiDi=0x11
prescaler = 31, divider = 12 ➜ 31 * 12 = 372, CKU=0.

Note : With the mask04 firmware, the following baud rates are supported :

TA1	0x01	0x02	0x03	0x04	0x08
CLK/ETU	372	186	93	46.5	31

TA1	0x11	0x12	0x13	0x14	0x18
CLK/ETU	372	186	93	46.5	31

TA1	0x21	0x22	0x23	0x28
CLK/ETU	558	279	139.5	46.5

TA1	0x31	0x32	0x33	0x34	0x35	0x38
CLK/ETU	744	372	186	93	46.5	62

TA1	0x41	0x42	0x43	0x44	0x48
CLK/ETU	1116	558	279	139.5	93

TA1	0x51	0x52	0x53	0x54	0x55	0x56	0x58
CLK/ETU	1488	744	372	186	93	46.5	124

TA1	0x61	0x62	0x63	0x64	0x68	0x69
CLK/ETU	1860	930	465	232.5	155	93

TA1	0x91	0x92	0x93	0x94	0x95	0x96
CLK/ETU	512	256	128	64	32	16

TA1	0xA1	0xA2	0xA3	0xA4	0xA5	0xA6	0xA8
CLK/ETU	768	384	192	96	48	24	64

TA1	0xB1	0xB2	0xB3	0xB4	0xB5	0xB6
CLK/ETU	1024	512	256	128	64	32

TA1	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC8
CLK/ETU	1536	768	384	192	96	48	128

TA1	0xD1	0xD2	0xD3	0xD4	0xD5	0xD6
CLK/ETU	2048	1024	512	256	128	64

Note : As the baud rates in dark boxes are using CKU bit, they are not reachable when CLK = Xtal.

9.2.10 set_NAD

This command is used from the application layer in order to specify a SAD (source address) and a DAD (destination address) for a logical connection using T=1 protocol as defined in ISO7816-3. The default value is 00 and will be kept until the send NAD command has been notified to the TDA8029. Any NAD submission where SAD and DAD are identical (except 00) will be rejected. If bits b4 or b8 of the NAD required are set to 1 (VPP programming) the NAD will be rejected.
The NAD shall be initialised before any information exchange with the card using T=1 protocol, otherwise and error message will be generated.

System to TDA8029 : 60 00 01 A5 NAD LRC
TDA8029 to System : 60 00 00 A5 LRC

Where NAD is the new value of NAD immediately taken into account.

9.2.11 get_card_param

This command is used from the application level in order to get the Fi and Di parameters of the card in use, the current card clock frequency, and the protocol in use.
FiDi parameter will be given on one byte (FiDi), the card clock frequency on one byte (CC), and the protocol on one byte (TT).
FiDi will give the value of the current Fi Di (Example 11H for Fi=372 and Di=1)
CC will take value 01 H for Fxtal, 02H for FXtal/2, 04H for FXtal/4 and 08H for FXtal/8.
TT will take value 00H for protocol T=0 and value 01H for protocol T=1.

If there is no card in use, an error message will be generated.

```
System to TDA8029 :  60   00 00   A6   C6
TDA8029 to System :  60   00 03   A6   FiDi CC TT   LRC
```

Where	FIDI	gives the current FIDI coded as in TA1 parameter,
	CC	gives the value of the card clock frequency as coded in CCR register of TDA8029,
	TT	gives the protocol used by the card (00 for protocol T=0, 01 for protocol T=1).

9.3 Synchronous card related commands

9.3.1 Synchronous card S=9

9.3.1.1 power_up_S9

The card is powered under 5V and answers 4 bytes as Answer To Reset.

System to TDA8029 : 60 00 00 6B OB
TDA8029 to System : 60 00 04 6B XX_1 XX_2 XX_3 XX_4 LRC

Where XX_1 XX_2 XX_3 XX_4 are the data sent by the card in its ATR.

The card is then ready to operate.

9.3.1.2 read_S9

This command allows to read bytes of 8 bits in the card from the specified address.

System to TDA8029 : 60 00 04 03 AD_H AD_L NB_H NB_L LRC
TDA8029 to System : 60 NB_H NB_L 03 D_1 D_2 D_3…D_n LRC

Where	AD_H AD_L	indicates the address where to read (coded on 2 bytes)
	NB_H NB_L	is the number of bytes to read (coded on 2 bytes)
	D_1 D_2 D_3…D_n	are the NB_H NB_L data read

9.3.1.3 read_S9_protection

This command allows to read bytes of 8 bits + the protect bit as the 9th bit in the card from the specified address.

System to TDA8029 : 60 00 04 04 AD_H AD_L NB_H NB_L LRC
TDA8029 to System : 60 (NB_H NB_L)*2 04 D_1 0/1 D_2 0/1 D_3 0/1…D_n 0/1 LRC

Where	AD_H AD_L	indicates the address where to read (coded on 2 bytes)
	NB_H NB_L	is the number of bytes to read (coded on 2 bytes)
	D_1 0/1 D_2 0/1 D_3 0/1 …D_n 0/1	are the NB_H NB_L data read

The process is the same as for the command read_8bit_S9 except that the value of the protect bit is added in the answer.
Each byte read is followed by one byte that informs if the byte is protected or not (0x00 : protected, 0x01 not protected).

9.3.1.4 write_S9_protected

This command allows to write bytes with protected bit as 9th bit from the specified address.

System to TDA8029 : 60 NB_H NB_L 05 AD_H AD_L D1 D2 D3 Bn LRC
TDA8029 to System : 60 00 00 05 LRC

Where	AD_H AD_L	indicates the address where to write (coded on 2 bytes)
	$(NB_H NB_L)$-2	is the number of bytes to write (coded on 2 bytes)
	D1 D2 D3…Dn	are the data to write in the card

9.3.1.5 write_S9_unprotected

This command allows to write bytes without protection from the specified address.

System to TDA8029 : 60 NB_H NB_L 06 AD_H AD_L D1 D2 D3 Dn LRC
TDA8029 to System : 60 00 00 06 LRC

Where	AD_H AD_L	indicates the address where to write (coded on 2 bytes)
	$(NB_H NB_L)$-2	is the number of bytes to write (coded on 2 bytes)
	D1 D2 D3…Dn	are the data to write in the card

9.3.1.6 verify_pin_code

System to TDA8029 : 60 00 03 07 XX PIN1 PIN2 LRC
TDA8029 to System : 60 00 00 07 LRC

Where	XX	in the bit mask for error counter
	PIN1	is the first PIN CODE
	PIN2	is the second PIN CODE

9.3.1.7 compare

System to TDA8029 : 60 00 03 08 AD_H AD_L XX LRC
TDA8029 to System : 60 00 00 08 LRC

Where	AD_H AD_L	indicates the address of byte to compare
	XX	is the byte to compare

9.3.2 Card S=10

9.3.2.1 power_up_S10

This command powers up the S10 card ; 4 bytes of Answer To Reset from the card are expected.

System to TDA8029 : 60 00 00 C1 A1
TDA8029 to System : 60 00 04 C1 xx_1 xx_2 xx_3 xx_4 LRC

Where xx_1 xx_2 xx_3 xx_4 are the data sent by the card in its ATR.

The card is then ready to operate.

9.3.2.2 process_S10

This command allows either to read or to write bytes from or into an S10 card from the specified address.

<u>In case of a read command</u> :
System to TDA8029 : 60 00 03 C2 CB AD NB LRC
TDA8029 to System : 60 NB_H NB_L 12 D1 D2 D3…Dn LRC

Where	CB	is the control byte
	AD	is the address byte
	NB	is the number of bytes to read
	D1 D2 D3…Dn	are the NB data read

<u>In case of a write command</u> :
System to TDA8029 : 60 ML_H ML_L C2 CB AD D1 D2 D3…Dn LRC
TDA8029 to System : 60 00 00 C2 A2

Where	ML_H ML_L	is the total message length
	CB	is the control byte
	AD	is the address byte
	D1 D2 D3…Dn	are the data to write in the card

9.3.3 I2C cards

9.3.3.1 power_up_I2C

This command powers up the I2C card ; no data are expected from the card.

System to TDA8029 : 60 00 00 6C OC
TDA8029 to System : 60 00 00 6C OC

The card is then ready to operate.

9.3.3.2 read_I2C

This command allows to read bytes from the specified address in a standard I2C card.

System to TDA8029 : 60 00 05 12 I2CAd AD_H AD_L NB_H NB_L LRC
TDA8029 to System : 60 NB_H NB_L 12 xx xx xx xx xx xx xx xx xx LRC

Where	I2CAd	is the physical I2C address of the embedded component
	AD_H AD_L	indicates the address where to read (coded on 2 bytes)
	NB_H NB_L	is the number of bytes to read (coded on 2 bytes)
	xx xx xx	are the NB_H NB_L data read

9.3.3.3 read_I2C_extended

This command allows to read bytes from the specified address in an extended I2C card.

System to TDA8029 : 60 00 05 13 I2CAd AD_H AD_L NB_H NB_L LRC
TDA8029 to System : 60 NB_H NB_L 13 xx xx xx xx xx xx xx xx xx LRC

Where	I2CAd	is the physical I2C address of the embedded component
	AD_H AD_L	indicates the address where to read (coded on 2 bytes)
	NB_H NB_L	is the number of bytes to read (coded on 2 bytes)
	xx xx xx	are the NB_H NB_L data read

9.3.3.4 read_current_I2C

This command allows to read bytes from the current address in a standard I2C card.

System to TDA8029 : 60 00 03 23 I2CAd NB_H NB_L LRC
TDA8029 to System : 60 NB_H NB_L 23 D1 D2 D3…Dn LRC

Where	I2CAd	is the physical I2C address of the embedded component
	NB_H NB_L	is the number of bytes to read (coded on 2 bytes)
	D1 D2 D3…Dn	are the NB_H NB_L data read

9.3.3.5 write_I2C

This command allows to write bytes from the specified address in an I2C card that is not using the extended mode. This function manages itself the MSB in the physical I2C address.

System to TDA8029 : 60 NB_H NB_L 02 I2CAd AD_H AD_L D1 D2 D3 … Dn LRC
TDA8029 to System : 60 00 00 02 LRC

Where	I2CAd	is the physical I2C address of the embedded component
	AD_H AD_L	indicates the address where to write (coded on 2 bytes)
	(NB_H NB_L)-3	is the number of bytes to write (coded on 2 bytes)
	D1 D2 D3…Dn	are the data to write in the card

Remark : This function does not manage the change of segment in the EEPROM. The maximum length of the data stream that can be programmed in one step depends of the embedded component.

9.4 General purpose IO commands

9.4.1 read_IO

This command is used to read the current state of the four general purpose IO of the TDA8029.

```
System to TDA8029 :   60   00 00   CE   AE
TDA8029 to System :   60   00 01   CE VAL   LRC
```

VAL is coded as follows :

0	0	0	0	P2.7	P2.6	P1.7	P1.6

0 if IO=GND, 1 if IO=V_{DD}

9.4.2 set_IO

This command is used to set one of the general purpose IO of the TDA8029 to a specified logic level.

```
System to TDA8029 :   60   00 02   CF   IO VAL   LRC
TDA8029 to System :   60   00 00   CF   AF
```

Where :

- IO is coded as follows,

IO	Port
0x01	P1.6
0x02	P1.7
0x03	P2.6
0x04	P2.7

- VAL is the value to apply to the IO.

VAL	level to apply
0x00	GND
0x01	V_{DD}

10 INFORMATION FIELD FOR ASYNCHRONOUS CARDS

The data buffer has a size of 512 bytes whose 6 bytes located at the end of the buffer are used by the internal library; so the data buffer has a real size of 506 bytes.
The information field that can include up to 506 bytes is composed of APDUs (Application Protocol Data Unit) according to the ISO7816-4 normalization definition.
Different examples are given according to Annex A of the EMV'96 in T = 0.

TAL (System)		**TTL (TDA8029)**

Case 1 command

$\{\underbrace{60, 00, 04, 00}_{\text{4 header bytes}}, CLA, INS, P1, P2, LRC\}$ ⇒

⇐ {60, 00, 02, 00, 90, 00, LRC}

Case 2 command

{60, 00, 05, 00, CLA, INS, P1, P2, 00, LRC} ⇒

⇐ {60, Licc+2, 00, [Data (Licc)], 90, 00, LRC}

Case 3 command

{60, Lc+5, 00, CLA, INS, P1, P2, Lc, [data Lc], LRC} ⇒

⇐ {60, 00, 02, 00, 90, 00, LRC}

Case 4 command

{60, Lc+5+1, 00, CLA, INS, P1, P2, Lc, [data Lc], 00, LRC} ⇒

⇐ {60, Licc+2, 00, [data Licc], 90, 00, LRC}

Case 2 command using the 61 and 6C procedure byte

Le = Licc or Le ≥ Licc

{60, 00, 05, 00, CLA, INS,P1, P2, 00, LRC} ⇒

⇐ {60, D1+D2+Dn+2, 00, [data D1+D2+Dn], 90, 00, LRC}

10.1 Extended cases

In T=0 protocol, the extended cases for APDUs are not supported on this mask.

In T=1 protocol, the use of the extended cases for APDUs is transparent from the host point of view as explained below as the TPDUs are identical to the APDUs.

Case 2 extended example :
APDU : CLA INS P1 P2 00 B2 B3 where B2 B3 is the length coded on 2 bytes (from 1 to 65535). With this mask 04 release, B2 B3 shall never exceed 498 bytes.

System to TDA8029 :	60 00 07 00 CLA INS P1 P2 00 B2 B3 LRC
TDA8029 to card :	NAD PCB 07 CLA INS P1 P2 00 B2 B3 EDC
Card to TDA8029 :	NAD PCB LEN1 D1 D2 …. Di EDC Where LEN1 is related to the negotiated data buffer size.
TDA8029 to card :	Rblock for acknowledge.
Card to TDA8029 :	NAD PCB LEN2 Di+1 D1+2 …. Dn SW1 SW2 EDC Where n = B2 B3

For this example it is supposed that only one chaining step is necessary.

TDA8029 to System :	60 B4 B5 00 D1 D2 …….. Dn SW1 SW2 LRC (B4 B5 =n+2)

References: ISO 7816-4 §5.3 and Annex B.

11 APPLICATION RECOMMENDATIONS FOR LOW TEMPERATURE OPERATION

For low temperature operation (-20°C, -30°C), the following procedure at supply voltage powering on is recommended :

1. VDD and DCIN are rising,
2. Wait 30ms (or less till CDEL is loaded) and set RESET pin high for about 20µs,
3. Wait another 30ms,
4. TDA8029 is ready to operate.

This operating mode can of course be used on the whole temperature range.

12 CONCLUSION

The following features give the general characteristics of the mask 04 :

- 3V and 5V cards supported
- E.M.V.3.1.1 validated but possibility to switch to full ISO 7816-3
- Optional Energy Saving Mode (hardware selection with **P26**, i.e. pin 25 of the TDA8029)
- Data buffer up to 506 bytes
- Asynchronous protocols (T=0 and T=1) supported
- A few synchronous cards (I2C, S9, S10) supported
- Two host interface modes for control and communication : RS232 (with variable baud rates from 4800 to 115200 baud) or IC2-bus (hardware selection with **P17**, i.e. pin 1)
- Automatic hardware protections in the event of card take off, supply voltage drop short circuit or overheating
- All ISO7816-3 baud rates supported on the I/O line
- Possible selection of card clock frequencies
- Communication with the host made at the APDU level (asynchronous cards) or also possible at TPDU level for protocol T=1
- Single +2.7V to +6.0V supply voltage
- Configuration summary :

Host Interface	**P17 (pin #1)**	**CAKE8029_09D**
RS232 (see §5)	VDD or not connected	ST4 open / ST3 soldered or open
I2C (see §6)	GND	ST4 soldered / ST3 open

ESM	**P26 (pin #25)**	**CAKE8029_09D**
ON	VDD or not connected	ST6 open / ST5 soldered or open
OFF	GND	ST6 soldered / ST5 open

13 ANNEX I : SCRTester

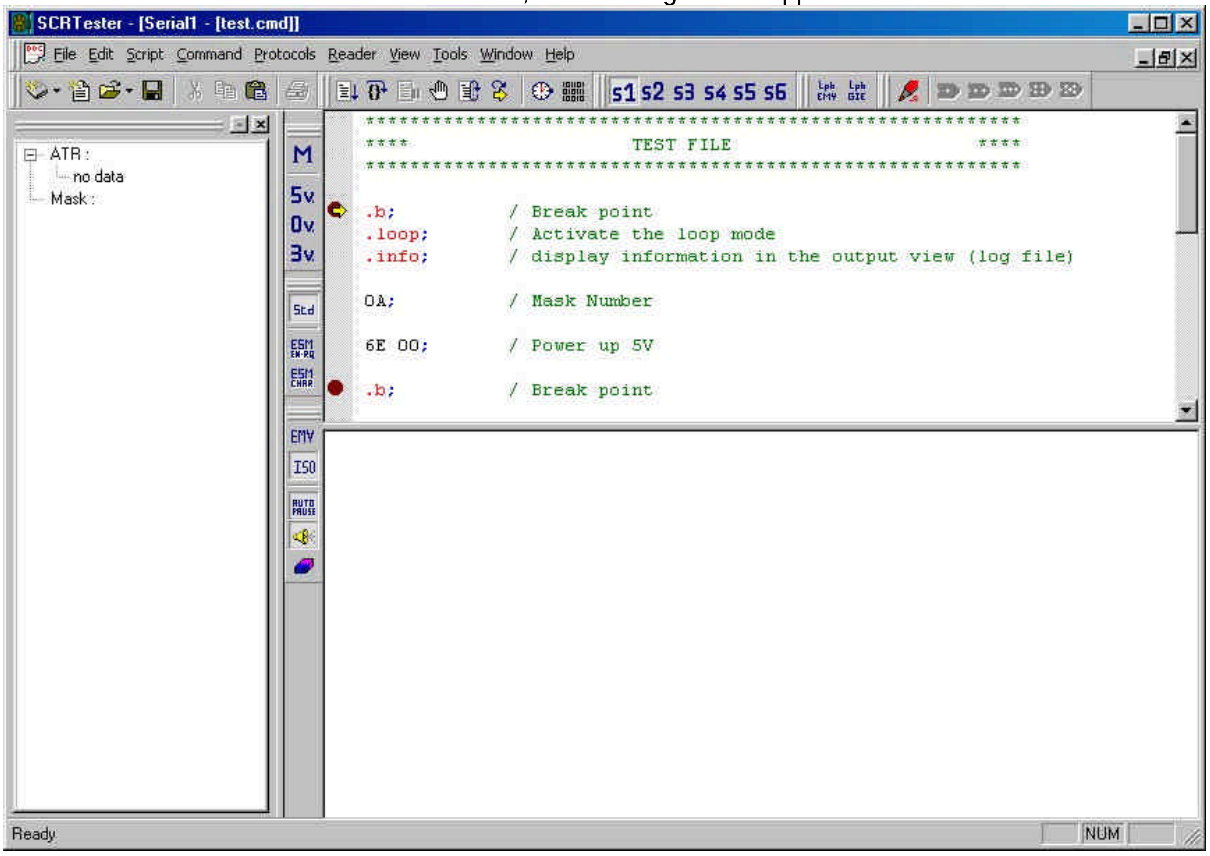
SCRTester is a PC software allowing to communicate with a Philips smart card reader (CAKE8029_09D for instance) through an RS232 serial link.
SCRTester can be used when the TDA8029 mask04 is configured to be interfaced with a host controller by using a RS232 serial link (not I2C-bus).

13.1 Installation

SCRTester is supplied in two floppy disks. Run the *setup.exe* file located on the floppy #1 to install SCRTester on your computer, and then follow the given indications.
Once installed, SCRTester is available in **c:\Philips Semiconductors** directory (*SCRTester.exe*).

13.2 Run SCRTester

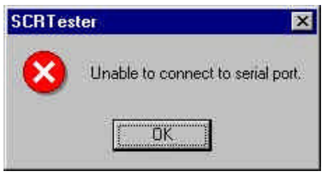
When SCRTester is launched for the first time, the following screen appears :


SCRTester - [Serial1 - [test.cmd]]
File Edit Script Command Protocols Reader View Tools Window Help
S1 S2 S3 S4 S5 S6
ATR :
no data
Mask :
M
5v
0v
3v
Std
ESM EN-PQ
ESM CHAR
EMV
ISO
AUTO PAUSE
Ready
NUM

```
*************************************************************
****                      TEST FILE                      ****
*************************************************************

.b;           / Break point
.loop;        / Activate the loop mode
.info;        / display information in the output view (log file)

0A;           / Mask Number

6E 00;        / Power up 5V

.b;           / Break point
```

SCRTester includes a complete help file that can be launched by pressing **F1** function key or by selecting the **Help→Contents** menu item.

SCRTester tries to establish a serial connection with the reader on COM1 port. If this port is not available, the following warning message appears and then the user has to manually configure the port used by the reader by using the Reader menu item (select the correct port **COMx** and after that use the **Connect** command).


SCRTester

Unable to connect to serial port.

13.2.1 The top right window

The top right window contains the command script file, which can be directly modified by the user.
The commands have to be written following the correct format defined in this application note. By default, SCRTester is configured in **Command** mode (**Script** menu item), i.-e. ALPAR header frame (except Command byte) and LRC character are not needed.
For example, to send a *send_num_mask* command to the reader (p. 31), one has to write

```
0A;
```

in the script window. SCRTester will automatically send the complete frame to the reader :

```
60 00 00 0A 6A
```

13.2.2 The bottom right window

The bottom right window contains all the commands sent to the reader (in red colour) and the received answers (in blue colour).

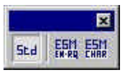
13.2.3 The left window : card parameters

In case of *send_num_mask* or *power_up* commands, the left window is refreshed with received information from the reader :

- the current mask number string is displayed,
- the complete ATR is decomposed into individual fields.

13.2.4 Energy Saving Mode (ESM) Configuration

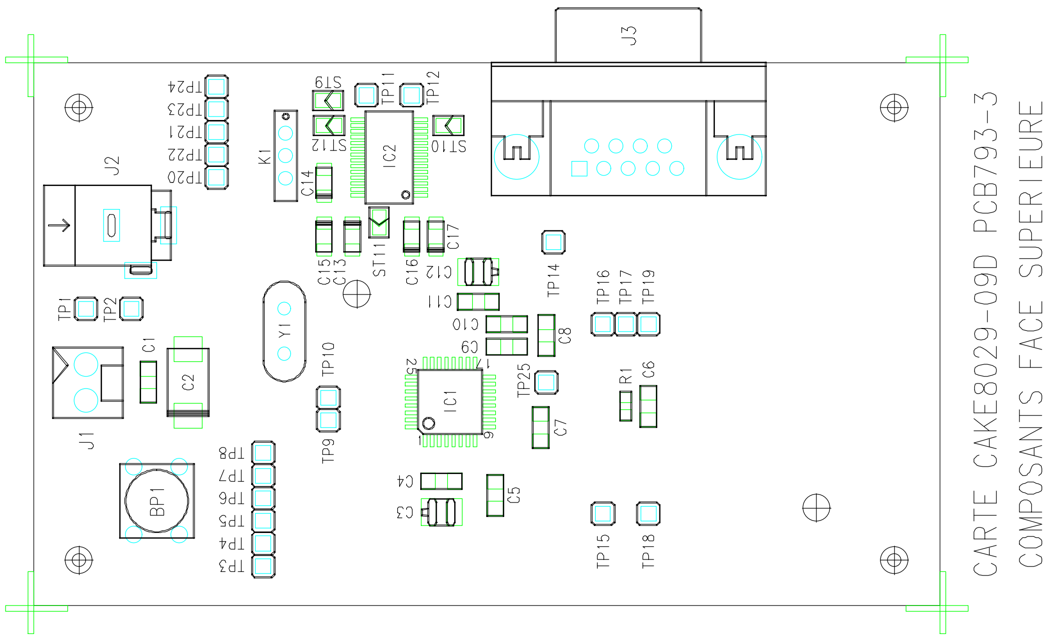
SCRTester has to be properly configured according to the ESM configuration of the connected TDA8029 reader.
In SCRTester, the Energy Saving Mode configuration is done by using the **Protocol→Mode** menu item or the following **Mode toolbar** palette.

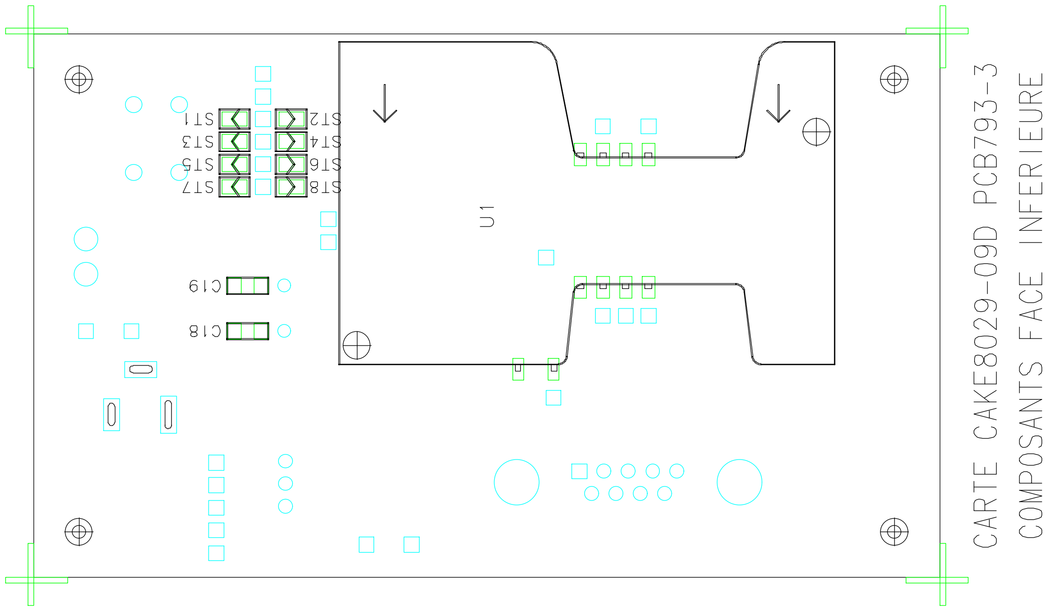


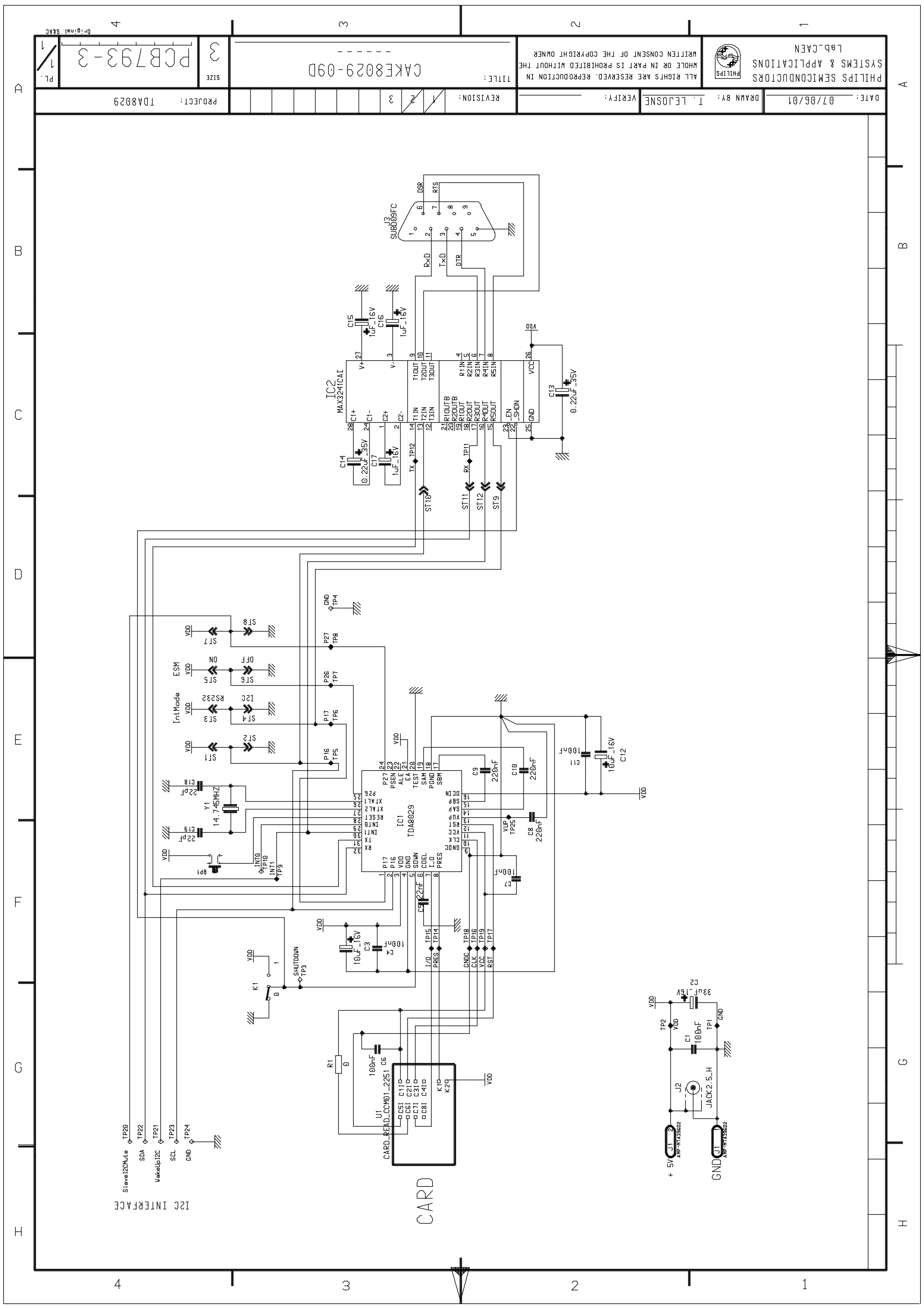
When the TDA8029 mask04 is configured to work in non Energy Saving Mode, choose the **Std** mode, as **ESM char** mode has to be used when TDA8029 Mask04 is configured in ESM ON.

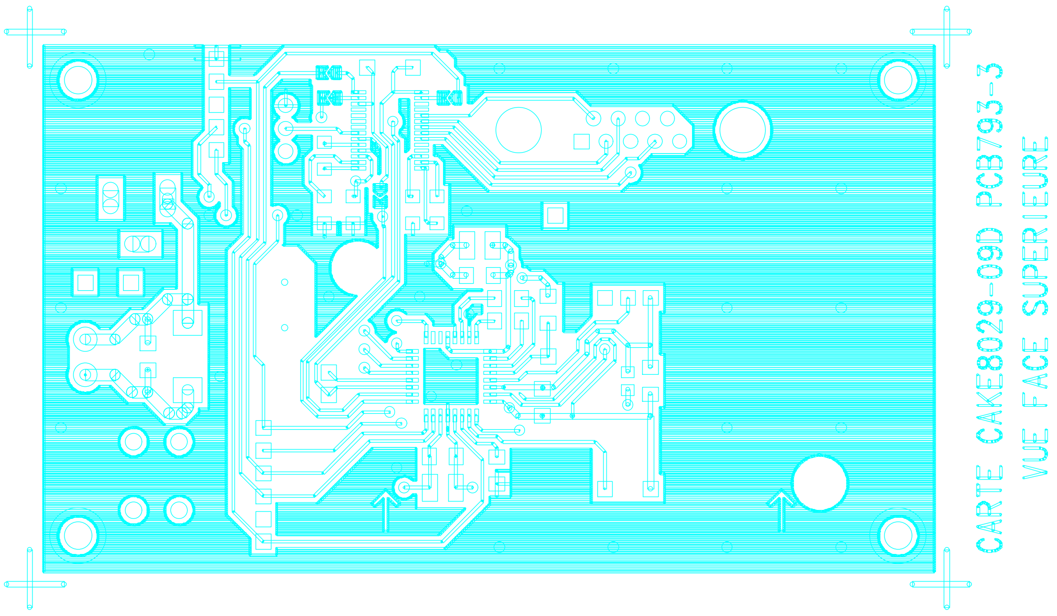
SCRTester completely handles the protocols extra frame associated to the Energy Saving Mode, as defined in §5.2., p. 13.

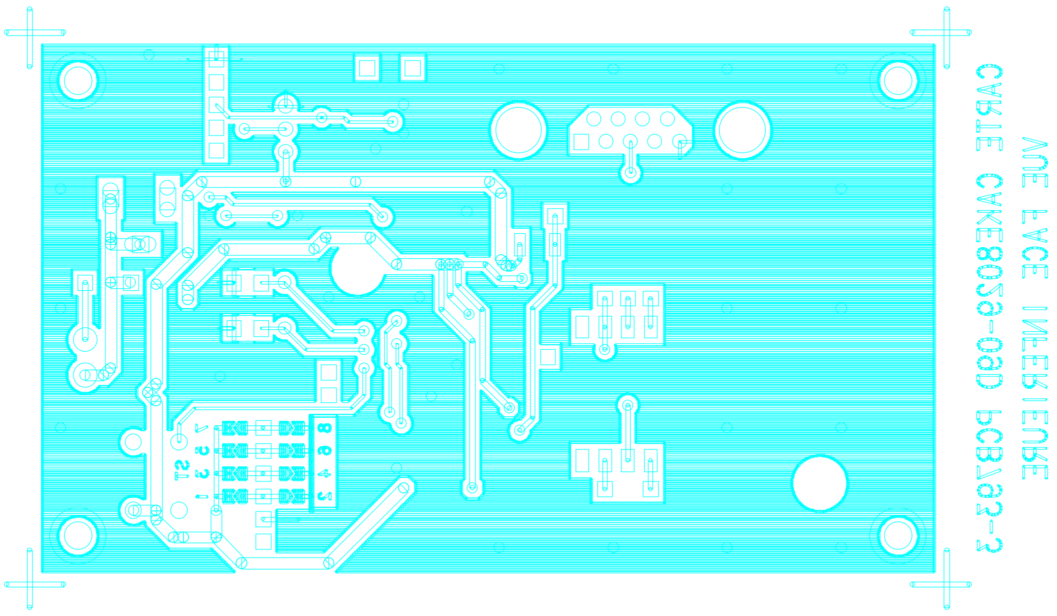
14 ANNEX II : HARDWARE INFORMATION






CAKE8029-09D
PCB7793-3
TDA8029
PHILIPS SEMICONDUCTORS
SYSTEMS & APPLICATIONS
Lab_CAEN
07/06/01
T. LEJOSNE
IC1
TDA8029
IC2
MAX3241CAI
U1
CARD
I2C INTERFACE


CARTE CAKE8029-09D PCB793-3
VUE FACE SUPERIEURE


CARTE CAKE8029-09D PCB793-3
VUE FACE INFERIEURE

CARTE	**PCB793**
IND.	**3**
ETUDE	**CAKE8029-09D**
PROJET	**TDA8029**
DATE	**08 Juin 2001**

REFERENCE	GEOMETRY	VALUE	SPECIFICATION
---------	--------	-----	-------------
BP1	microcosmos	MICROCOSMOS	poussoir2
C1	c1206	100nF	capa
C2	c293d_d	33uF_16V	capa_pol
C3	c595d_b	10uF_16V	capa_pol
C4	c1206	100nF	capa
C5	c1206	22nF	capa
C6	c1206	100nF	capa
C7	c1206	100nF	capa
C8	c1206	220nF	capa
C9	c1206	220nF	capa
C10	c1206	220nF	capa
C11	c1206	100nF	capa
C12	c595d_b	10uF_16V	capa_pol
C13	c293d_a	0.22uF_35V	capa_pol
C14	c293d_a	0.22uF_35V	capa_pol
C15	c293d_a	1uF_16V	capa_pol
C16	c293d_a	1uF_16V	capa_pol
C17	c293d_a	1uF_16V	capa_pol
C18	c1206	22pF	capa
C19	c1206	22pF	capa
IC1	sot358_1	TDA8029	tda8029
IC2	sot341_1	MAX3241CAI	max3241
J1	mta396d2	AMP:MTA396D2	edge
J2	jack2.5_h	JACK2.5_H	cinch_h
J3	subd_09fc	SUBD09FC	subd_09
K1	int_1k2	1K2	int_1k2
R1	r0805	0	res
ST1	chevron	CHEVRON	chevron
ST2	chevron	CHEVRON	chevron
ST3	chevron	CHEVRON	chevron
ST4	chevron	CHEVRON	chevron
ST5	chevron	CHEVRON	chevron
ST6	chevron	CHEVRON	chevron
ST7	chevron	CHEVRON	chevron
ST8	chevron	CHEVRON	chevron
ST9	chevron	CHEVRON	chevron
ST10	chevron	CHEVRON	chevron
ST11	chevron	CHEVRON	chevron
ST12	chevron	CHEVRON	chevron
TP1	tp0.9	TEST_Bar1	test
TP2	tp0.9	TEST_Bar1	test
TP3	tp0.9	TEST_Bar1	test
TP4	tp0.9	TEST_Bar1	test
TP5	tp0.9	TEST_Bar1	test
TP6	tp0.9	TEST_Bar1	test
TP7	tp0.9	TEST_Bar1	test
TP8	tp0.9	TEST_Bar1	test
TP9	tp0.9	TEST_Bar1	test
TP10	tp0.9	TEST_Bar1	test
TP11	tp0.9	TEST_Bar1	test
TP12	tp0.9	TEST_Bar1	test
TP14	tp0.9	TEST_Bar1	test
TP15	tp0.9	TEST_Bar1	test
TP16	tp0.9	TEST_Bar1	test
TP17	tp0.9	TEST_Bar1	test

TP18	tp0.9	TEST_Bar1	test
TP19	tp0.9	TEST_Bar1	test
TP20	tp0.9	TEST_Bar1	test
TP21	tp0.9	TEST_Bar1	test
TP22	tp0.9	TEST_Bar1	test
TP23	tp0.9	TEST_Bar1	test
TP24	tp0.9	TEST_Bar1	test
TP25	tp0.9	TEST_Bar1	test
U1	card_read_ccm01_2251	CARD_READ_CCM01_2251	card_read_ccm01_2251
Y1	hc49s	14.745MHZ	quartz