GemCore ChipSet Controller Software

Version 2.0

GEMPLUS

February, 1998

ABOUT THIS DOCUMENT

This document provides information about the GemCore ChipSet Controller software. You will find a detailed description of the GemCore ChipSet hardware in the GemCore Technical Specifications.

Audience

This document is to be used by anyone wishing to build electronic system implementing the use of a Smart Card Interface.

How to Use This Guide

The following paragraphs tell you where to find information when you need it. It is important that you read this section in order to use this document to its full potential.

Overview

Read the Overview for an overall description of the GemCore ChipSet Controller.

The GCC Protocols

Read this section for a description of the existing protocols between the GemCore ChipSet Controller and the host system.

The GCC Interface Commands

Read the GCC Interface Commands section for a description of the GemCore ChipSet Controller commands, the functions they perform, their syntax and the format you send them in to the GemCore ChipSet.

Using the GCC with Microprocessor Cards

The GemCore ChipSet Controller supports ISO 7816-3 T=0 and the T=1 protocol microprocessor cards. Read this section for a description of these standards.

Using the GCC with Memory Cards

Read this section for a summary of a all the commands used with memory cards.

Status Codes

The status codes returned by the GemCore ChipSet Controller are listed in this section.

Interpreted Synchronous Smart Card Driver

This section describes the instructions and macro commands dedicated to synchronous cards handled by the GemCore interpreter

CONTENTS

OVERVIEW

The present document describes the interface between your application and the GemCore ChipSet Controller.

Composed of one programmed controller and one smart card interface chip **-the GEMPLUS IC100-** the GemCore ChipSet is designed to simplify the integration of Smart Card interface to electronic devices and manages the communication with ISO 7816 1/2/3/4 compatible smart card.

The software inside the controller is compatible with the GEMPLUS Readers Operating System (OROS). It includes the communication protocols with the host system (GBP or TLP protocol) as well as the protocols with synchronous and asynchronous smart cards.

The connection with the host system is done with a serial asynchronous port in TTL level.

GEMCORE CHIPSET PROTOCOLS

All transmissions with the GemCore ChipSet are handled by three protocol layers:

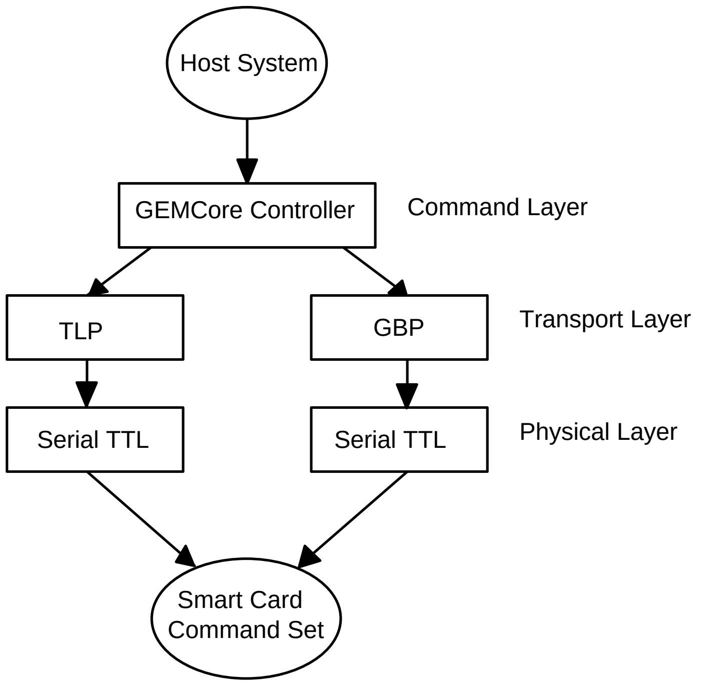
- the command layer
- the transport layer
- the physical layer

The command layer handles and interprets the GemCore ChipSet commands. It consists of the command code, data, and parameters.

The transport layer handles the message addressing, specifies the transmission type, and validates each transmission. The transport layer can use one of two protocols: the TLP224 protocol and the GEMPLUS Block Protocol.

The physical layer handles the data transmission itself. The physical layer uses the Serial TTL protocol.

The following diagram shows the three-layer protocol.



The following paragraphs describe the protocol layers in more details.

Command Layer

The command layer handles and interprets the GemCore ChipSet commands. It consists of the command code, data, and parameters.

You send commands in the following format:

```
|CommCode|Parameters|Data|
```

where:

`CommCode`	is the command code.
`Parameters`	are the parameters sent with the command.
`Data`	is the data accompanying the command, where appropriate.

The GemCore ChipSet Interface Commands section describes the CommCode, Parameters, and Data field values for each command.

The GCC replies to every command it receives with a status code formatted as follows:

```
|S|Data|
```

where:

`S`	Status code identifier.
`Data`	Data returned with the status code, where appropriate.

Transport Layer

The transport layer handles the message addressing, specifies the transmission type, and validates each transmission. The GCC transport layer can use one of two protocols: the TLP224 protocol and the GEMPLUS Block Protocol. The following paragraphs describe these.

TLP224

The TLP protocol processing consists of two steps. The first step is to construct the message to be transmitted. Under the TLP224 protocol, the exchange transmissions have the following format:

For messages transmitted without error:

`<ACK><LN><MESSAGE><LRC>`

where:

`ACK`	60h, indicating that the previous command or status code was transmitted without error.
`LN`	Length of the message (command or status code)
`MESSAGE`	Command or status code.
LRC	The result of an EXCLUSIVE OR (XOR) between the characters ACK, LN, and MESSAGE.

When an error is detected in the transmission:

`<NACK><LN><LRC>`

where:

`NACK`	E0h, indicating that there was an error in a message transmission.
`LN`	00
`LRC`	E0

During the second step the source performs the following processing:

- converts each byte to be transmitted into two ASCII characters. For example, to transmit the byte 3Ah, the source would transmit the values 33h and 41h. This prevents other equipment from interpreting the control characters.
- adds an End Of Transmission (EOT) byte to the end of the transmission. This has the value 03h.

For example, to transmit the power down command under the TLP224 protocol, which has the command code 4Dh and no parameter, the following sequence would be transmitted:

	ACK	LEN	Message	LRC	EOT
Command	60	01	4D	2C	
TLP Protocol Transmission	36 30	30 31	34 44	32 43	03

The time-out between each character is 100 ms.

GEMPLUS Block Protocol

The GEMPLUS Block Protocol (GBP) is a simplified version of the T=1 card protocol. Under the GBP, data is transmitted in blocks between the source and the destination. There are three types of blocks:

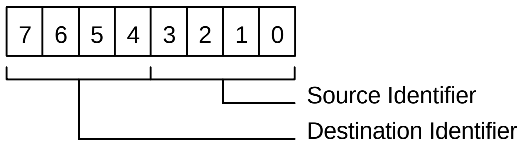
- I-Blocks. (Information Blocks). I-Blocks hold the data to be exchanged between the source and the destination.
- R-Blocks (Receive Ready Block). R-Blocks hold positive or negative acknowledgments to transmissions.
- S-Blocks (Supervisory Block). S-Blocks synchronize transmissions between the source and the destinations.

Data is exchanged in the following format:

NAD	PCB	LEN	DAT	EDC

where:

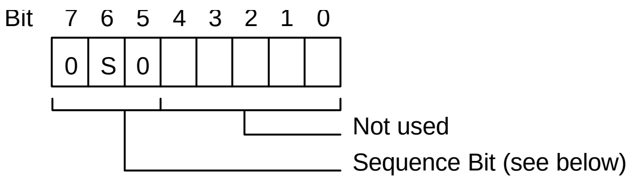
NAD is the source and the destination identifier formatted as follows on one byte:



The GemCore ChipSet identifier is 4 and the host system identifier is 2.

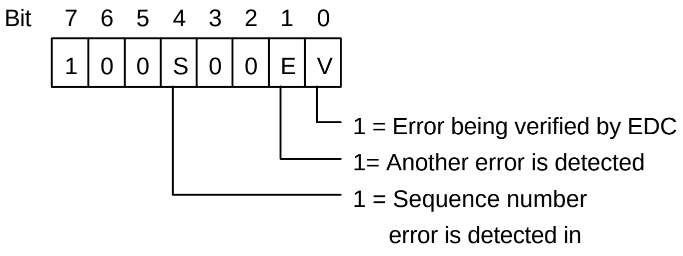
PCB identifies the block type. Its format depends on the block type, as described below:

I-Block PCBs have the following format:



The sequence bit is zeroized on power up. The source sends the first I-Block that it transmits with the sequence bit set to 0. It increments the sequence bit by 1 each time it sends an information block. The GemCore ChipSet Controller and the host system generate sequence bit values independently.

R-Block PCBs have the following format:


Bit 7 6 5 4 3 2 1 0
1 0 0 S 0 0 E V
1 = Error being verified by EDC
1= Another error is detected
1 = Sequence number error is detected in

S-Blocks request the destination to zeroize the sequence bits and return a response to the source; this response indicates that the response is fulfilled.

S-Block PCBs have the following format:

Bit	7	6	5	4	3	2	1	0	
	1	1	0	0	0	0	0	0	Resynch request

Bit	7	6	5	4	3	2	1	0	
	1	1	1	0	0	0	0	0	Resynch response

LEN specifies, on one byte, the number of bytes in the INF field (*see* below).

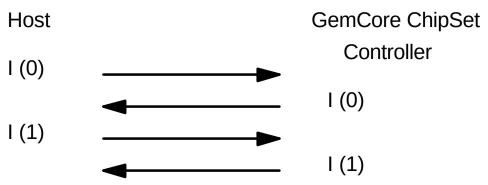
DAT holds the data being transmitted.

EDC is the result of an exclusive OR performed on the NAD, PCB, LEN, and DAT bytes.

Examples

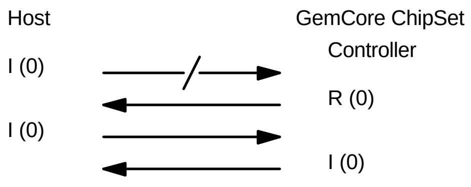
The following examples show some transmission types under the GBP protocol.

Transmission without error:

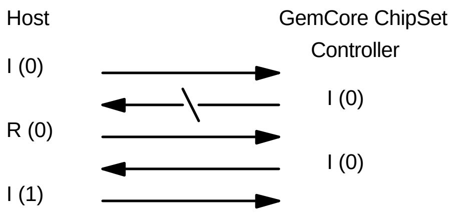


Transmission with error:

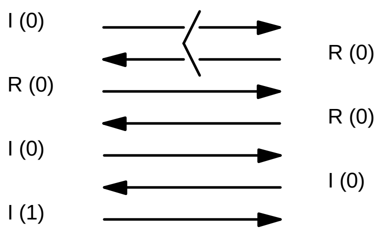
Case 1.



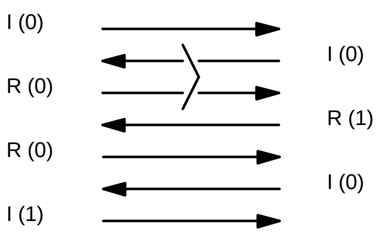
Case 2.



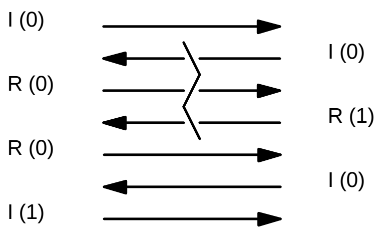
Case 3.



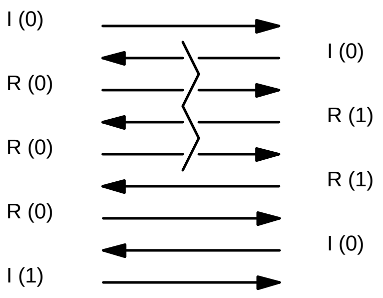
Case 4.



Case 5.



Case 6.



Physical Layer

The physical layer handles the data transmission itself. The physical layer uses the Serial protocol.

Serial Asynchronous Protocol

The Serial Asynchronous Protocol can be sent directly on the serial line.

The bytes are sent over the line by an UART whose transmission characteristics (such as speed and parity) are determined by the configuration of the GemCore ChipSet Controller.

The default configuration is 9600 baud, 8 bits, no parity and 1 stop bit.

GEMCORE CHIPSET CONTROLLER INTERFACE COMMANDS

This section describes the GemCore ChipSet Controller commands. For each command it describes:

- the functions it performs
- its syntax
- the data it returns

The commands are grouped into command sets. The GemCore ChipSet Controller command sets are:

- Configuration
- Card interface #0
- Card interface #1

The rest of this section describes the GemCore ChipSet Controller commands.

Command Format

You send commands to the GemCore ChipSet in the following format:

```
|CommCode|Parameters|Data|
```

where:

`CommCode`	is the command code.
`Parameters`	are the parameters sent with the command.
`Data`	is the data accompanying the command, where appropriate.

The GemCore ChipSet commands section describes the CommCode, Parameters, and Data field values for each command.

The GemCore ChipSet replies to every command it receives with a status code formatted as follows:

```
|S|Data|
```

where:

`S`	Status code identifier.
`Data`	Data returned with the status code, where appropriate.

Appendix A lists the status codes and their meanings.

Configuration GemCore ChipSet Controller Commands

The following pages describe the GemCore ChipSet Controller commands.

The GemCore ChipSet Controller configuration commands are:

- Configure SIO Line
- Set Mode
- Set Delay
- Read Firmware Version

Find in this section a description of these commands.

CONFIGURE SIO LINE

This command sets the SIO line parity, Baud rate, and number of bits per character. After a power up the line defaults to no parity, 8 bits per character and 9600 Baud.

***Note**: The line is reconfigured as soon as this command is executed. The response is returned with the newly specified parameters.*

Format `0Ah` *`CB`*

where:

`CB` = configuration byte. Flag the required configuration according to the following table:

Bit	Value	Option Selected
7 to 5		Not used
4	0	No parity
	1	Even parity
3	0	8 bits per character
	1	7 bits per character
2 to 0	xxx	Sets the baud rate according to the following table:

Value	Baud rate selected
000	RFU
001	76 800
010	38 400
011	19 200
100	9 600
101	4 800
110	2 400
111	1 200

SET MODE

This command enables you to disable ROS command compatibility and define the GemCore ChipSet operation mode (TLP or Normal). The GemCore ChipSet defaults to ROS command compatibility enabled and TLP mode.

Notes:
1. Disabling ROS command compatibility disables this command. You can only enable again ROS command compatibility by performing a hardware reset on the GemCore ChipSet so that the default configuration is reinstated.

2. Disabling ROS command compatibility also disables TLP mode, irrespective of the value of bit 4 (see below).

Format

`01h 00h` *[OB]*

where:

[OB] = option selection byte. Flag the required options according to the following table:

	Native	ROS	TLP
xxxx1xx1	Ú	Ú	Ú
xxxx1xx0	Ú	Ú	
xxxx0xx0	Ú		

Note: *If you do not send this byte, the GemCore ChipSet operation mode is not modified, however, the result is returned.*

Result

S <*mode*>

where:

[mode] = The mode the GemCore ChipSet is operating in. This is returned on one byte that flags the operation mode according to the following table:

	Native	ROS	TLP
00001001	Ú	Ú	Ú
00001000	Ú	Ú	
00000000	Ú		

Note: *In TLP mode, the GemCore ChipSet Controller adds the TA1, TB1, TC1, TD1 bytes if they are not present in an asynchronous card Answer to Reset.*

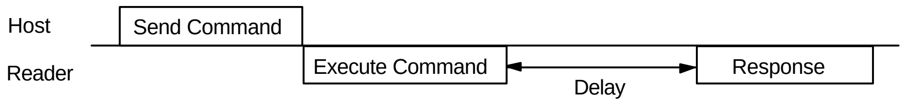
SET DELAY

If you are using a slow host computer with the GemCore ChipSet Controller, you can use this command to delay responses.

Format **`23h 01h 00h 4Ch 01h`** *`Delay`*

where:

`Delay` = response delay in ms. Enter a value between 0 and 255. On power up, the delay time defaults to 0.



READ FIRMWARE VERSION

Returns the version of the firmware installed in the GemCore ChipSet.

Format **`22h 05h 3Fh E0h 10h`**

Result **`S`** `Version`

where:

`Version` (GemCore-XXXX) is the installed software

version in ASCII.

OROS compatible command:

Format **`22h 05h 3Fh F0h 10h`**

Result **`S`** `OROS-R2.99-R1.00`

Card Interface Command Set

The card interface commands manage all communications with smart cards. The card interface commands are:

- Power Down
- Power Up
- ISO Output
- ISO Input
- Exchange APDU
- Define Card type
- Card Status

The following paragraphs describe these commands.

POWER DOWN

Use this command to power down the card. The GemCore ChipSet Controller powers down automatically when a card is removed.

GCC Format	`11h`	*Card #0*
	`19h`	*Card #1*
ROS Format	`4Dh 00h 00h 00h`	*Card #0 only*

Result **S**

The power down command always ends without error if a card is present in the GemCore ChipSet.

If no card is inserted, the command returns the Fbh error "card absent".

POWER UP

This command powers up and resets a card.

GCC Format

12h [CFG][PTS0,PTS1,PTS2,PTS3,PCK] *Card #0*

1Ah [CFG][PTS0,PTS1,PTS2,PTS3,PCK] *Card #1*

ROS Format

`6Eh 00h 00h 00h` *`Card #0 only`*

There is no CFG parameter: the card is powered by 5V, there is no PTS management and the operating mode is compatible with OROS2.2X

The parameter CFG is present:

XXXXXX01	Class A	Vcc for Card or Module is 5V
XXXXXX10	Class B	Vcc for Card or Module is 3V
XXXXXX11	Class AB	Vcc for Card or Module is 5V or 3V
0000XXXX	Operation is compatible with OROS2.2X	
0001XXXX	Reset and no PTS management. The GemCore ChipSet stays at 9600 baud if the card is in negotiable mode.	
0010XXXX	Reset and automatic PTS management. The GemCore ChipSet takes the highest speed proposed by the card. Change to T=1 protocol if there is a choice between T=0 and T=1.	
1111XXXX	Manual management of PTS. This command does not reset the card. It must be preceded by a command with the PTS parameter at 0001XXXX. The parameters from PTS0 to PCK are sent to the card at 9600 baud. If the card replies with PTS REQUEST, the GemCore ChipSet is configured using the sent parameters.	

Result

`S <card response>`

where:

`<card response> =` the card Answer to Reset.

***Note**: For cards that do not return an Answer to Reset, a default Answer to Reset is returned:* 3B 00 00 00 00 00

Using the ROS command, if TLP compatibility is enabled, the ATR is preceded by three bytes R1, R2, R3.

R1: compatibility mode 28h: TLP

01h: ROS

R2: current card type

R3: ATR length

Note: *When the TLP compatibility is enabled (see Set Mode command) the TA_1, TB_1, TC_1 and TD_1 bytes absent from the Answer to Reset are returned with their default value:*

	TA_1	TB_1	TC_1	TD_1
Asynchronous Card	11h	25h	00h	00h
Synchronous Card	00h	00h	00h	00h

ISO OUTPUT

This command sends ISO Out commands, that is, commands that retrieve data from a card. For memory cards, specific commands that are formatted in the same way as ISO commands are accepted. These commands are listed in the Using the GemCore ChipSet Controller with Memory Cards section.

GCC Format

```
13h CLA INS A1 A2 LN   Card #0
1Bh CLA INS A1 A2 LN   Card #1
```

ROS Format

```
DBh CLA INS A1 A2 LN   Card #0 only
```

where:

`CLA`, `INS`, `A1`, `A2`, and `LN` are the five ISO header bytes. For more details about the ISO header contents, refer to the documentation relevant to the card you are using. The ISO header is directly transmitted to microprocessor cards (asynchronous cards) and is interpreted by the GemCore ChipSet Controller for GEMPLUS memory cards.

Result

```
S
<data> SW1 SW2
```

where:

`<data>` = Up to 252 bytes of data returned by the card. If a smart card error or GCC error is detected (S<>0 and S<>E7h), the GCC does not return any data. The card may return any number of bytes up to LN.

ISO INPUT

This command sends ISO In commands, that is, commands that send data to a card. For memory cards, the GemCore ChipSet Controller accepts specific commands that are formatted in the same way as ISO commands. These commands are listed in the Using the GemCore ChipSet Controller with Memory Cards section.

GCC Format

```
14h CLA INS A1 A2 LN <data>   Card #0
1Ch CLA INS A1 A2 LN <data>   Card #1
```

ROS Format

```
DAh CLA INS A1 A2 LN <data>   Card #0 only
```

where:

`CLA`, `INS`, `A1`, `A2`, and `LN` are the five ISO header bytes. For more details about the ISO header contents, refer to the documentation relevant to the card you are using. The ISO header is directly transmitted to microprocessor cards (asynchronous cards) and is interpreted by the GCC for GEMPLUS memory cards.

`<data>` represents the `LN` data bytes transmitted to the card after the ISO header. The maximum length of the data is 248 bytes.

Result

```
S SW1 SW2
```

The bytes `SW1` and `SW2` hold the standard status codes returned by the card. Their respective values are 90h and 00h if the operation is successful.

Note*: For GEMPLUS memory cards, SW1 and SW2 are returned.*

EXCHANGE APDU

Sends a command Application Data Protocol Unit (APDU) to a card, and retrieves the response APDU. You can only execute this command on T=1 protocol cards.

GCC & ROS Format

15h *APDU* *Card #0*

1Dh *APDU* *Card #1*

where:

APDU = the command APDU. If the APDU command length is greater than the card information field size, it is truncated and sent to the card in several chained blocks. The command APDU must not exceed 248 bytes in length. See the documentation for the card in use for the APDU command details.

Result

S *Response APDU*

where:

Response APDU = the response APDU to the command. If the card replies in chained blocks, they are concatenated. The response APDU must not exceed 252 bytes in length. See the documentation for the card in use for the APDU response details.

APDU Format

The APDU format is defined by the ISO 7816-4 standard.

APDUs can belong to one of several cases, depending on the length and contents of the APDU. The GemCore ChipSet Controller supports the following cases

Case 1-no command or response data.

Case 2-Short format: command data between 1 and 255 bytes and no response data.

Case 3-Short format: no command data, between 1 and 256 bytes.

Case 4-Short format: command data between 1 and 255 bytes, response data between 1 and 256 bytes.

These cases are referred to as 1, 2S, 3S, and 4S respectively.

Command Format

Commands are accepted in the following format:

Header	Body		
CLA INS P1 P2	Lc	Parameters/data	Le

The fields are described below:

Header Fields

The Header fields are mandatory, and are as follows:

Field Name	Length	Description
CLA	1	Instruction class.
INS	1	Instruction code. This is given with the command descriptions.
P1	1	Parameter 1.
P2	1	Parameter 2.

Body Fields

The command body is optional. It includes the following fields:

Field Name	Length	Description
Lc	1	Data length
Data	Lc	Command parameters or data
Le	1	Expected length of data to be returned

For full details about the Header and Body field contents refer to the documentation for the card in use.

Response Format

Responses to commands are received in the following format.

Body	Trailer
Data	SW1, SW2

The Body is optional and holds the data returned by the card.

The Trailer includes the following two mandatory bytes:

SW1: Status byte 1 that returns the command processing status

SW2: Status byte 2 that returns the command processing qualification

For full details about the Response field contents refer to the documentation for the card in use.

IFSC/IFSD

In case of chaining, the buffer length is determined by IFSC and IFSD parameters. The default value is 32 bytes for IFSD (data buffer length).

If in the ATR the smart card indicates an IFSC (Card data buffer length) value, the GemCore ChipSet considers that value as the IFSD length and will use it for chained exchanges with the smart card.

DEFINE CARD TYPE

The GemCore ChipSet Controller does not have a smart card recognition algorithm. You must define the card type in use. This command sets the card type and programming voltage. Note that the ROS and GCC versions of this command are different. The two formats are described below.

***Note**: When the OROS based GemCore ChipSet is reset or powered up, the card type defaults to microprocessor card in standard mode (Type 2).*

GCC Format

```
17h T  Card #0
1Fh T  01h  Card #1
```

ROS Format

```
02h T   Card #0 only
```

where:

T = Card type selection byte. Enter the code for the card type that you are using on the four least significant bits (bits 3 to 0). The card type codes are as follows:

Enter this code:	To use this card:
01h	Other synchronous smart cards; interpreted driver. See *Appendix B.*
02h	Standard speed mode (clock frequency = 3.6864 MHz) ISO 7816-3 T=0 and T=1 microprocessor cards.
12h	Double speed mode (clock frequency = 7.3728 Mhz) ISO 7816-3 T=0 and T=1 microprocessor cards.
03h	GPM256
04h	GPM416/GPM896 in Standard Mode
14h	GPM416/GPM896 in Personalization Mode
06h	GFM2K/GFM4K
07h	GPM103
08h	GPM8K(SLE4418/4428)
09h	GPM2K(SLE4432/4442 or PCB2032/2042)
10h	GAM144

If the command is entered with a family number that is different from that of the current card, the current card is powered down. You can also use this command to modify the voltage without changing the card type in use by entering the same card code as that in use.

Result S

CARD STATUS

Sends the above command to know the smart card interface or the security module's status. It returns information relating to:

- card type in use
- card presence
- power supply value
- powered up card
- communication protocol (T=0, T=1)
- speed parameters between card and GemCore ChipSet

GCC Format

`17h` *`Card #0`*

`1Fh` *`Card #1`*

Result

`S   STAT   TYPE   CNF1   CNF2   CNF3   CNF4`

`where`

STAT:	NNNNXXXX	Card number
		0000XXXX=Card#0
		0001XXXX=Card#1
	XXXXXXX0	power supply = 5V
	XXXXXXX1	power supply = 3V
	XXXXXX0X	card not powered
	XXXXXX1X	card powered
	XXXXX0XX	card not inserted
	XXXXX1XX	card inserted
	XXXX0XXX	protocol T=0
	XXXX1XXX	protocol T=1

TYPE:	Activated Card type

CNF1	CNF1=TA1 (FI/DI)	T=0 Card according to ISO 7816/3
CNF2	CNF2=TC1 (EGT)	
CNF3	CNF3=WI	
CNF4	CNF4=00	

CNF1	CNF1=TA1 (FI/DI)	T=1 Card according to ISO 7816/3
CNF2	CNF2=TC1 (EGT)	
CNF3	CNF3=IFSC	
CNF4	CNF4=TB3 (BWI/CWI)	
CNF1	CNF1=00	Synchronous Smart Cards
CNF2	CNF2=00	
CNF3	CNF3=00	
CNF4	CNF4=00	

USING THE GEMCORE CHIPSET CONTROLLER WITH MICROPROCESSOR CARDS

The GemCore ChipSet Controller supports ISO 7816-3 T=0 and T=1 protocol microprocessor cards. The following section describes the implementation of these standards.

Clock Signal

The GemCore ChipSet Controller can transmit one of two clock frequency values to the card, depending on the previously selected operating mode:

- 3.6864 Mhz for the standard mode (ISO compliance)
- 7.3728 Mhz for the double speed mode (above the ISO range for cards that can operate at this frequency)

You specify the operating mode while selecting the card type using the DEFINE CARD TYPE command. Select card type 02h for the standard mode and card type 12h for the double speed mode.

Global Interface Parameters

These parameters are returned by the microprocessor card during the Answer to Reset. For more information on these parameters please refer to the ISO 7816-3 standard document.

TA1 Parameter

The GemCore ChipSet Controller interprets this parameter to match its communication rate with that of the card, based on the clock rate conversion factor F. F is coded on the most significant nibble and the bit rate adjustment factor D, is coded on the least significant nibble.

The initial communication rate used during the Answer to Reset is 9909.68 baud in the standard mode and 19819.35 baud in the double speed mode.

After it receives the Answer to Reset, the GemCore ChipSet Controller installs the communication rate depending on TA1. Table 1 and Table 2 below show the clock rate conversion factors, the bit rate conversion factors, and the baud rates installed in relation to the TA1 values for standard mode and double speed mode cards.

***Note**: The GemCore ChipSet Controller only supports the TA1 shaded values in Tables 1 and 2.*

TB1 and TB2

The Vpp option is not available on the GemCore ChipSet Controller. TB1 and TB2 parameters are ignored and the Vpp default value is set to 5V.

TC1

This parameter defines the extra guardtime N, required by the card. This parameter is processed when sending characters to the card, to ensure a delay of at least (12+N) etu between two characters.

Table 1. Supported TA1 values in standard mode (clock frequency = 3.6864 Mhz)

D=	1		2		4		8		16	
F=	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)
372	**11**	9 909.68	**12**	19 819.35	**13**	39 638.71	**14**	79 277.42	**15**	158 554.84
558	21	-	**22**	13 212.90	**23**	26 425.81	**24**	52 851.61	**25**	105 703.23
744	31	-	**32**	9 909.68	**33**	19 819.35	**34**	39 638.71	**35**	79 277.42
1116	41	-	42	-	**43**	13 212.90	**44**	26 425.81	**45**	52 851.61
1488	51	-	52	-	**53**	9 909.68	**54**	19 819.35	**55**	39 638.71
1860	61	-	62	-	63	-	**64**	15 855.48	**65**	31 710.97
512	91	-	**92**	14 400.00	**93**	28 800.00	**94**	57 600.00	**95**	115 200.00
768	A1	-	A2	-	**A3**	19 200.00	**A4**	38 400.00	**A5**	76 800.00
1024	B1	-	B2	-	**B3**	14 400.00	**B4**	28 800.00	**B5**	57 600.00
1536	C1	-	C2	-	C3	-	**C4**	19 200.00	**C5**	38 400.00
2048	D1	-	D2	-	D3	-	**D4**	14 400.00	**D5**	28 800.00

Table 2. Supported TA1 values for double speed mode (clock frequency = 7.3728 Mhz)

D=	1		2		4		8		16	
F=	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)
372	**11**	19819.35	**12**	39 638.71	**13**	79 277.42	**14**	158 554.84	15	-
558	**21**	13 212.90	**22**	26 425.81	**23**	52 851.61	**24**	105 703.23	25	-
744	**31**	9 909.68	**32**	19 819.35	**33**	39 638.71	**34**	79 277.42	35	-
1116	41	-	**42**	13 212.90	**43**	26 425.81	**44**	52 851.61	45	-
1488	51	-	**52**	9 909.68	**53**	19 819.35	**54**	39 638.71	55	-
1860	61	-	62	-	**63**	15 855.48	**64**	31 710.97	65	-
512	**91**	14 400.00	**92**	28 800.00	**93**	57 600.00	**94**	115 200.00	95	-
768	A1	-	**A2**	19 200.00	**A3**	38 400.00	**A4**	76 800.00	A5	-
1024	B1	-	**B2**	14 400.00	**B3**	28 800.00	**B4**	57 600.00	B5	-
1536	C1	-	C2	-	**C3**	19 200.00	**C4**	38 400.00	C5	-
2048	D1	-	D2	-	**D3**	14 400.00	**D4**	28 800.00	D5	-

Communication Protocols

The least significant nibble of the TD1 parameter in the Answer to Reset defines the protocol (T=0 or T=1) to be used by the GemCore ChipSet, according to the following table:

This value:	**Selects this protocol:**
0	T=0
1	T=1

If the GemCore ChipSet does not receive a TD1 value, it defaults to the T=0 protocol.

T=0 Protocol

The TC2 specific interface parameter is interpreted to set the value of the work waiting time, W. When this parameter is absent , a maximum of 960xD etu elapsed before timing-out on a character sent by the card takes place. Otherwise there is a maximum of 960xDxW before timing-out.

To send instructions to a T=0 microprocessor card, you use the ISO Input and ISO Output commands.

T=1 Protocol

To send instructions to a T=1 microprocessor card, you use the *Exchange APDU* command. The T=1 specific interface bytes are interpreted according to clause 9 of the ISO 7816-3 standard. These bytes are TA_3, TB_3, TC_3.

TA_3 codes the Information Field Size of the card (IFSC). The default value is 32 bytes.

TB_3 codes the BWI (Block Writing Time Integer) and CWI (Character Waiting Time Integer).

TC_3 defines the Error Detection Code (EDC) type.

USING THE GEMCORE CHIPSET CONTROLLER WITH MEMORY CARDS

Memory cards cannot interpret smart card instructions in the same way as ISO 7816-3 microprocessor cards can. Therefore T=0 are formatted instructions are interpreted and converted into the appropriate timing sequences required to control the memory cards listed in the table below.

For further details, refer to the relevant card documentation.

You send these instructions to the GemCore ChipSet, using the ISO Input and ISO Output commands

Memory Card Command Summary

		ISO Input/ISO Output Command Parameters (CLA = 00)			
Card Type	**Command Name**	**INS**	**A1**	**A2**	**Ln**
GPM256	Write Bytes	D0	00	Start Address	Write Length
	Read Bytes	B0	00	Start Address	Read Length
GPM103	Write Bytes	D0	00	Start Address	Write Length
	Erase and Write Carry	DE	01	Counter to erase	0
	Write New value to Counter	D2	05	08	02
	Read Bytes	B0	00	Start Address	Read Length
	Read Counter Value	B2	05	08	02

Continued on following page

Memory Card Command Summary (continued)

		ISO Input/ISO Output Command Parameters (CLA = 00)			
Card Type	**Command Name**	**INS**	**A1**	**A2**	**Ln**
GPM896	Write Bytes	D0	00	Start Address	Write Length
	Erase Word	DE	Number of words	Start Address	00
	Present Erase Code1	20	00	36	06
	Present Erase Code2	20	80	5C	04
	Present Card Secret Code	20	04	0A	02
	Present Secret Code	20	Number of bits in error counter	Start Address	Code Length
	Read	B0	00	Start Address	Read Length
GPM416	Write Bytes	D0	00	Start Address	Write Length
	Erase Word	DE	Number of words	Start Address	00
	Present Erase Code	20	40	28	04
	Present Card Secret Code	20	04	08	02
	Read	B0	00	Start Address	Read Length
GAM144	Write Bytes	D0	00	Start Address	Write Length
	Erase	0E	01	Start Address	00
	Write value	D2	05	08	02
	Restore	D4	No. bytes to restore	08	00
	Blow Fuse	DA	Fuse ID	1A	00
	Authenticate	88	12	19	00
	Read Bytes	B0	00	Start Address	Read Length
	Read Counter Value	B2	05	08	02
	Get Result	C0	00	00	01

Continued on following page

Memory Card Command Summary (continued)

		ISO Input/ISO Output Command Parameters (CLA = 00)			
Card Type	**Command Name**	**INS**	**A1**	**A2**	**Ln**
SLE4418/4428 GPM8K	Read Bytes	B0	00 = Data memory	Start Address Least Significant Nibble	Read Length
	Write Bytes	D0	00 = Data memory	Start Address Least Significant Nibble	Write Length
	Check Secret Code	20	00	00	02
SLE4432/4442 PCB2032/2042 GPM2K	Read Memory	B0	Memory Area: 00=Data Memory 80=Data Protection Area C0=Security Area	Read Start Address	Length of Data to Read
	Write Memory	D0	Memory Area: 00=Data Memory 80=Data Protection Area C0=Security Area	Write Start Address	Length of Write Data
	Check Secret Code	20	00	00	03

APPENDIX A. STATUS CODES

The returned status codes are listed in the table below.

Code	**Meaning**
01h	Unknown driver or command.
02h	Operation not possible with this driver.
03h	Incorrect number of arguments.
04h	GemCore ChipSet command unknown. The first byte of the command is not a valid command code.
05h	Response too long for the buffer.
09h	Communication protocol error. The header of a message is neither ACK or NACK (60h or E0h)
10h	Response error at the card reset. The first byte of the response (TS) is not valid
11h	ISO command header error. The byte INS in the ISO header is not valid (6x or 9x).
12h	Message too long. The buffer is limited to 254 bytes, of which 248 bytes are for the data exchanged with the card
13h	Byte reading error returned by an asynchronous card.
15h	Card turned off. A Power Up command must be applied to the card prior to any other operation.
16h	Programming voltage not available. The parameter V in the DEFINE CARD TYPE command is not valid.
17h, 18h	Communication protocol unknown or incorrectly initialized.
19h	Illegal access to external bus.
1Ah	Error in an ISO format card command. The parameter LN in the ISO header does not correspond to the actual length of the data.
1Bh	A command has been sent with an incorrect number of parameters.
1Dh	The check byte TCK of the response to reset of a microprocessor card is incorrect.
1Eh	An attempt has been made to write to external memory, which is write protected.
1Fh	Incorrect data has been sent to the external memory. This error is returned after a write check during a downloading operation.
A0h	Error in the card reset response, such as unknown exchange protocol, or byte TA1 not recognized. The card is not supported. The card reset response is nevertheless returned.
A1h	Card protocol error (T=0/T=1).
A2h	Card malfunction. The card does not respond to the reset or has interrupted an exchange (by time-out).
A3h	Parity error (in the course of an exchange microprocessor). The error only occurs after several unsuccessful attempts at re transmission.

A4h	Card has aborted chaining (T=1).
A5h	GemCore chipset has aborted chaining (T=1).
A6h	Protocol type Selection (PTS) error.
CFh	Overkey already pressed.
E4h	The card has just sent an invalid "Procedure Byte" (*see* ISO 7816-3).
E5h	The card has interrupted an exchange (the card sends an SW1 byte but more data has to be sent or received).
E7h	Error returned by the card. The bytes SW1 and SW2 returned by the card are different from 90h 00.
F7h	Card removed. The card has been withdrawn in the course of carrying out of a command. Check that the card instruction is not partially completed.
F8h	The card is consuming too much electricity or is short circuiting.
FBh	Card absent. There is no card in the smart card interface. The card may have been removed when it was powered up, but no command has been interrupted.

APPENDIX B. INTERPRETED SYNCHRONOUS SMART CARD DRIVER

Card Type 01h

This command is used to handle synchronous card protocols which are not supported by GemCore. The protocol to be used is defined by parameters specified in 8051 assembler code.

The 8051 assembler (INTEL ASM51) generates the commands to be executed and the GemCore software interprets the bytes as 8051 operation codes.

The GemCore interpreter can execute most 8051 instructions along with a few macro commands dedicated to synchronous cards.

Format

`16h` *`CLA INS A1 A2 Lin <DATA IN> Lout Lcode <CODE>`*

where:

`CLA, INS, A1, A2`	are the command parameters.
`Lin`	is the number of bytes present in the `DATA IN` field.
`DATA IN`	is the data to be sent to the card.
`Lout`	is the length of the expected response.
`Lcode`	is the number of bytes present in the `CODE` field.
`CODE`	is the 8051 executable code.

Result

`S <data byte>`

8051 Interpreter

The GemCore interpreter handles the following functions:

- An accumulator (A)
- Eight registers (R0 to R7)
- A carry (C)
- A program counter (PC)

All instructions concerning the IDATA or XDATA RAM memories, also have an incidence on the XDATA memory. The XDATA memory starts at address 0000h and ends at address 00FFh.

The instruction to be executed is registered in this memory area (command 16h).

Only relative jumps can be used.

Initialization

Upon reception of a 16h command, the interpreter registers are initialized as follows:

PC points to the first `<CODE>` byte.
C = 0
A = `CLA`
R0 and R4 point to the address following the last `<CODE>`byte.
R1 points to the address of the first `<DATA IN>` byte.
R2 = `Lin`
R3 = `Lout`
R5 = `INS`
R6 = `A1`
R7 = `A2`

```
16h CLA=A INS=R5 A1=R6 A2=R7 Lin=R2 <DATAIN> Lout=R3 Lcode <CODE>
                                    ↑ R1                R0/R4 ↑
```

Card Presence

Before executing a 16h command, the software checks that a card is actually present in the smart card connector.

If the card is missing the following error message will be returned: “CARD ABSENT” (S = FBh).

Card Withdrawal

As soon as the smart card is powered up, the GemCore card withdrawal interruption is activated.

If the card is withdrawn, the interpreted program is interrupted, all contacts with the smart card are deactivated and the following error message is returned : “CARD WITHDRAWN” (S = F7h).

Short Circuit

The card power up instructions check for short circuits between pins C1 (VCC) and C5 (GND).

If a short circuit is detected, the following error message is returned : “TOO MUCH CONSUMPTION” (S = F8h).

Instructions

The following table is used to obtain a hexadecimal instruction code. The line number defines the four most significant bits and the column number defines the least significant bits (e.g. INC A = 04h).

***Note**: The instructions in italics are macro-commands. See the "Macro-Commands" section for more details.*

	0	*1*	**2**	**3**	**4**	**5**	**6**	**7**
0	NOP 1/12	*VCC_OFF* 1/		RR A 1/16	INC A 1/18		INC @R0 1/22	INC @R1 1/22
1		*VCC_ON* 1/	*RESET* 1/	RRC A 1/19	DEC A 1/18		DEC @R0 1/22	DEC @R1 1/22
2		*CLR_RST* 1/13	RET (*) 1/	RL A 1/14	ADD A,#data 2/21		ADD A,@R0 1/26	ADD A,@R1 1/26
3		*SET_RST* 1/13	RETI (*) 1/	RLC A 1/21	ADDC A,#data 2/24		ADDC A,@RO 1/29	ADDC A,@R1 1/29
4	JC rel 2/15/19	*CLR_IO* 1/13	*RET_0K* 1/	*RDH_L* 1/	ORL A,#data 2/17		ORL A,@R0 1/22	ORL A,@R1 1/22
5	JNC rel 2/15/20	*SET_IO* 1/13	*RET_NOK* *2/*	*RDH_R* 1/	ANL A,#data 2/17		ANL A,@R0 1/22	ANL A,@R1 1/22
6	JZ rel 2/15/19	*CLR_CLK* 1/13	*RET_ERR* *3/*	*WRL_L* 1/	XRL A,#data 2/17		XRL A,@R0 1/22	XRL A,@R1 1/22
7	JNZ rel 2/17/20	*SET_CLK* 1/13	*CLK_INC* 1/14/XXX	*CLK_INC8* 1/14/XXX	MOV A,#data 2/19		MOV @R0, #data 1/27	MOV @R1, #data 1/27
8	SJMP rel 2/16	*CLR_C4* 1/13	*RDL_R* 1/	*RDL_L* 1/				
9		*SET_C4* 1/13	*WRH_L* 1/	*WRH_R* 1/	SUBB A,#data 2/		SUBB A,@R0 1/29	SUBB A,@R1 1/29
A		*CLR_C8* 1/13	*RST_PUL* 1/24	*WRL_R*				
B		*SET_C8* 1/13	*CLK_PUL* 1/24	CPL C 1/14	CJNE A,#data,rel 3/27/38		CJNE @R0,#data,rel 3/33	CJNE @R1,#data,rel 3/33
C		*SET_VPP* 2/229	*WAIT_US* 20/5100	CLR C 1/14	SWAP A 1/15		XCH A,@R0 1/27	XCH A,@R1 1/27
D			*WAIT_MS* 1ms/255ms	SETB C 1/14			XCHD A,@R0 1/25	XCHD A,@R1 1/25
E		*IO_TO_C* 1/16	*GET_D* 1/1100/1s	*GET_I* 1/1100/1s	CLR A 1/14		MOV A,@R0 1/25	MOV A,@R1 1/25
F		*C_TO_IO* 1/15	*SEND_D* 1/1100/1s	*SEND_I* 1/1100/1s	CPL A 1/14		MOV @R0,A 1/25	MOV @R1,A 1/25

Table 4. Hexadecimal instruction codes

1/12/23 means: instruction over one byte/12 µs min/23 µs max. (For the jump instructions, the time taken is maximum when the jump is executed).

(*) Instruction already existing in the 8051 but with a different function for the interpreter.

	8	9	A	B	C	D	E	F
0	INC R0 1 / 19	INC R1 1 / 19	INC R2 1 / 19	INC R3 1 / 19	INC R4 1 / 19	INC R5 1 / 19	INC R6 1 / 19	INC R7 1 / 19
1	DEC R0 1 / 19	DEC R1 1 / 19	DEC R2 1 / 19	DEC R3 1 / 19	DEC R4 1 / 19	DEC R5 1 / 19	DEC R6 1 / 19	DEC R7 1 / 19
2	ADD A,R0 1 / 24	ADD A,R1 1 / 24	ADD A,R2 1 / 24	ADD A,R3 1 / 24	ADD A,R4 1 / 24	ADD A,R5 1 / 24	ADD A,R6 1 / 24	ADD A,R7 1 / 24
3	ADDC A,R0 1 / 27	ADDC A,R1 1 / 27	ADDC A,R2 1 / 27	ADDC A,R3 1 / 27	ADDC A,R4 1 / 27	ADDC A,R5 1 / 27	ADDC A,R6 1 / 27	ADDC A,R7 1 / 27
4	ORL A,R0 1 / 20	ORL A,R1 1 / 20	ORL A,R2 1 / 20	ORL A,R3 1 / 20	ORL A,R4 1 / 20	ORL A,R5 1 / 20	ORL A,R6 1 / 20	ORL A,R7 1 / 20
5	ANL A,R0 1 / 20	ANL A,R1 1 / 20	ANL A,R2 1 / 20	ANL A,R3 1 / 20	ANL A,R4 1 / 20	ANL A,R5 1 / 20	ANL A,R6 1 / 20	ANL A,R7 1 / 20
6	XRL A,R0 1 / 20	XRL A,R1 1 / 20	XRL A,R2 1 / 20	XRL A,R3 1 / 20	XRL A,R4 1 / 20	XRL A,R5 1 / 20	XRL A,R6 1 / 20	XRL A,R7 1 / 20
7	MOV R0,#data 2 / 22	MOV R1,#data 2 / 22	MOV R2,#data 2 / 22	MOV R3,#data 2 / 22	MOV R4,#data 2 / 22	MOV R5,#data 2 / 22	MOV R6,#data 2 / 22	MOV R7,#data 2 / 22
8	-	-	-	-	-	-	-	-
9	SUBB A,R0 1 / 26	SUBB A,R1 1 / 26	SUBB A,R2 1 / 26	SUBB A,R3 1 / 26	SUBB A,R4 1 / 26	SUBB A,R5 1 / 26	SUBB A,R6 1 / 26	SUBB A,R7 1 / 26
A	-	-	-	-	-	-	-	-
B	CJNE R0, #data, rel 3 / 32 / 43	CJNE R1, #data, rel 3 / 32/ 43	CJNE R2, #data, rel 3 / 32 / 43	CJNE R3, #data, rel 3 / 32 / 43	CJNE R4, #data, rel 3 / 32 / 43	CJNE R5, #data, rel 3 / 32 / 43	CJNE R6, #data, rel 3 / 32 / 43	CJNE R7, #data, rel 3 / 32 / 43
C	XCH A,R0 1 / 21	XCH A,R1 1 / 21	XCH A,R2 1 / 21	XCH A,R3 1 / 21	XCH A,R4 1 / 21	XCH A,R5 1 / 21	XCH A,R6 1 / 21	XCH A,R7 1 / 21
D	DJNZ R0,rel 2 / 24 / 28	DJNZ R1,rel 2 / 24 / 28	DJNZ R2,rel 2 / 24 / 28	DJNZ R3,rel 2 / 24 / 28	DJNZ R4,rel 2 / 24 / 28	DJNZ R5,rel 2 / 24 / 28	DJNZ R6,rel 2 / 24 / 28	DJNZ R7,rel 2 / 24 / 28
E	MOV A,R0 1 / 20	MOV A,R1 1 / 20	MOV A,R2 1 / 20	MOV A,R3 1 / 20	MOV A,R4 1 / 20	MOV A,R5 1 / 20	MOV A,R6 1 / 20	MOV A,R7 1 / 20
F	MOV R0,A 1 / 19	MOV R1,A 1 / 19	MOV R2,A 1 / 19	MOV R3,A 1 / 19	MOV R4,A 1 / 19	MOV R5,A 1 / 19	MOV R6,A 1 / 19	MOV R7,A 1 / 19

Table 4. Hexadecimal instruction codes (continued)

1/12/23 means: instruction over one byte / 12 µs min / 23 µs max.

(*) Instruction already existing in the 8051 but with a different function for the interpreter.

Modified Instructions

RET

When the interpreter finds the RET code, the program is ended. GemCore returns the XDATA RAM memory data, R4 pointing to the first byte to be returned and R0 to the byte following the last response byte.

RETI

When the interpreter finds the RETI code, the program is ended. GemCore returns the contents of the registers in the following order:

```
PC A R0 R1 R2 R3 R4 R5 R6 R7 C
```

This instruction is used for software development.

Macro-Commands

%RET_OK

When the interpreter finds the RET_OK code, the program is ended. GemCore returns the last contents of the XDATA RAM memory, R4 pointing to the first byte to be returned and R0 to the byte following the last response byte.
S = 00h and the two status bytes SW1 = 90h and SW2 = 00H are added at the end of the message.

%RET_NOK (ERROR)

When the interpreter finds the RET_NOK instruction, the program is ended. GemCore returns the last contents of the XDATA RAM memory, R4 pointing to the first byte to be returned and R0 to the byte following the last response byte.
S = E7h, SW1 = 92h and SW2 returns an error code. These two bytes are added at the end of the message.

%RET_ERR (ERR1,ERR2)

Same as %RET_NOK but with SW1 = ERR1 and SW2 = ERR2.

%VCC_OFF

This command powers down all the smart card contacts as per ISO 7816-3 standard specifications.

%VCC_ON

This command initializes the smart card contacts.
If a card is present and is not short circuited, the following steps are carried out:

- VCC contact set at 5V.
- VPP contact set at 5V.
- RESET contact set to level 0.
- CLOCK contact set to level 0.
- I/O contact set to level 1 (high impedance).
- C4 contact set to level 0.
- C8 contact set to level 0.

%CLR_RST

This instruction sets the smart card's RESET contact to 0.

%SET_RST

This instruction sets the smart card's RESET contact to 1. It is only operative if the smart card is powered up.

%CLR_IO

This instruction sets the smart card's I/O contact to 0.

%SET_IO

This instruction sets the smart card's I/O contact to 1. It is only operative if the smart card is powered up.

%CLR_CLK

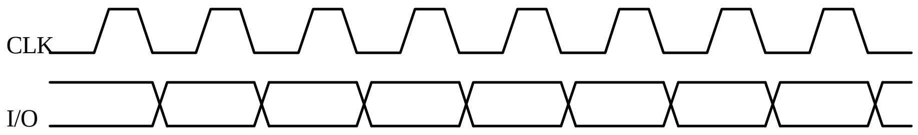
This instruction sets the smart card's CLOCK contact to 0.

%SET_CLK	This instruction sets the smart card's CLOCK contact to 1. It is only operative if the smart card is powered up.
%CLR_C4	This instruction sets the smart card's C4 contact to 0.
%SET_C4	This instruction sets the smart card's C4 contact to 1. It is only operative if the smart card is powered up.
%CLR_C8	This instruction sets the smart card's C8 contact to 0.
%SET_C8	This instruction sets the smart card's C8 contact to 1. It is only operative if the smart card is powered up.
%SET_VPP (VALUE)	This instruction sets the smart card's VPP contact to the voltage specified in the VALUE parameter and waits for 200 µs (VPP rise time). It is only operative if the smart card is powered up. ***Note***: *The VPP voltage value is coded in VALUE in 0.1V steps.*
%IO_TO_C	This instruction copies the state of the I/O contact into the C bit.
%C_TO_IO	This instruction copies the level held in C to the smart card's I/O contact. It is only operative if the smart card is powered up.
%CLK_INC	This instruction allows pulses to be generated on CLK. The total number of packets is indicated in A (0 to 255). CLK is set to 0 for 10 ms then to 1 for 10 ms. At the end of the sequence, CLK is set to 0.
%CLK_INC8	This instruction allows eight pulse packets to be generated on CLK. The total number of packets is indicated in A (0 to 255). CLK is set to 0 for 10 ms then to 1 for 10 ms. At the end of the sequence, CLK is set to 0.
%GET_D	When the 3.68 MHz asynchronous clock is activated on CLK, this command reads eight bits from the I/O in asynchronous mode and classes them in A using the direct convention. The configuration is 9,600 baud, 8 bits, even parity, 1 stop bit, 1s time-out.
%GET_I	Same as GET_D, but the eight bits read are classed in A using the inverse convention.
%SEND_D	When the 3.68 MHz asynchronous clock is activated on CLK, this command writes the contents of A on the I/O in asynchronous mode using the direct convention. The configuration is 9,600 baud, 8 bits, even parity, 1 stop bit, 1s time-out.
%SEND_I	Same as SEND_D, but the eight bits are written to the I/O using the inverse convention.

%RDL_R This command reads eight bits and classes them in A with a right rotation.

%RDL_L This command is the same as RDL_R but with a left rotation

The sequence for these two commands is as follows:



%RDL_R	b0	b1	b2	b3	b4	b5	b6	b7
%RDL_L	b7	b6	b5	b4	b3	b2	b1	b0

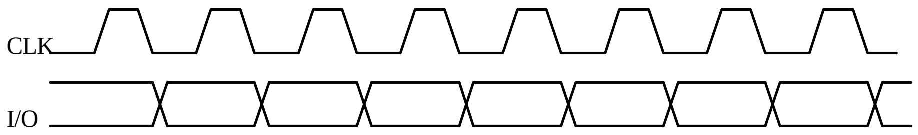
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs.

The I/O line is read 5µs **before** the CLK rising edge.

%RDH_R This command reads eight bits and classes them in A, with a right rotation.

%RDH_L This command is the same as RDH_R but with a left rotation

The sequence for these two commands is as follows:



%RDH_R	b0	b1	b2	b3	b4	b5	b6	b7
%RDH_L	b7	b6	b5	b4	b3	b2	b1	b0

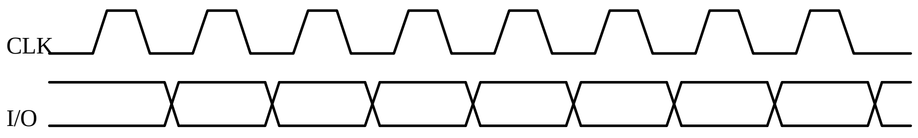
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs.

The I/O line is read 5µs **after** the rising edge of the clock.
The first bit to be read is b0 of A. The last bit to be read is b7 of A.
At the end of the command, CLK is set to level 0.

%WRH_R This command writes the contents of A on the I/O contact, with a right rotation.

%WRH_L This command is the same as WRH_R but with a left rotation (bit b7 of A is the first bit to be sent and bit b0 is the last).

The sequence for these two commands is as follows:



%WRH_R	b0	b1	b2	b3	b4	b5	b6	b7
%WRH_L	b7	b6	b5	b4	b3	b2	b1	b0

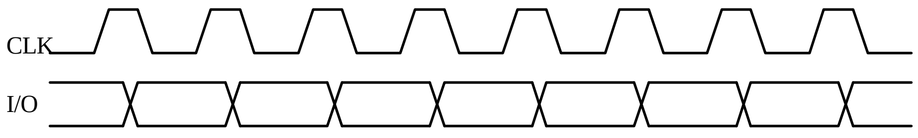
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs

The bit to be sent on I/O is set 5 µs **before** the rising edge of CLK.
Bit b0 of A is the first bit to be sent and bit b7 the last.
At the end of the command, CLK is set to level 0 and the I/O line is set to a high impedance level.

%WRL_R This command writes the contents of A on the I/O contact, with a right rotation.

%WRL_L Same as WRL_R but with a left rotation (b7 of A is the first bit to be sent and bit b0 is the last).

The sequence for these two commands is as follows:



%WRL_R	b0	b1	b2	b3	b4	b5	b6	b7
%WRL_L	b7	b6	b5	b4	b3	b2	b1	b0

- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs.

The bit to be sent on I/O is set 5 µs **before** the falling edge of CLK.
Bit b0 of A is the first bit to be sent and b7 the last.
At the end of the command, CLK is set to level 0 and the I/O line is set to a high impedance level.

%RST_PUL This command generates a logical pulse 1 for 10 µs on the RESET line and then resets the line to level 0.

%CLK_PUL This command generates a logical pulse 1 for 10 µs on the CLK line and then resets the line level to 0.

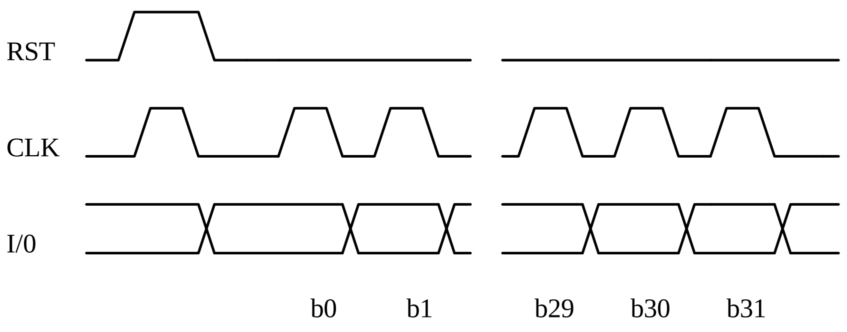
%WAIT_US (TIME) This command waits for the length of time specified in the TIME parameter.
The waiting time equals TIME * 10 µs.

%WAIT_MS (TIME)

This command waits for the length of time specified in the TIME parameter.
The waiting time equals TIME* 1ms.

%RESET

This command executes the RESET synchronous card sequence with the GPM2K/8K protocol. GemCore returns the 32 bit ATR.
Executing the command interrupts the current program.



The RST and CLK signals are forced to level 0 for 10µs.
The CLK signal rises 5µs after the RST rising edge and remains at 1 for 40µs.
The RST signal falls 5µs after CLK and remains at 0 until the end of the sequence.
The CLK high and low levels remain constant for 10µs while the ATR is read, and the data is read 5µs after the rising edge of the CLK.
b0 is the least significant bit of the first byte returned by GemCore, b7 being the most significant bit.
b8 is the least significant bit of the second byte returned by GemCore, b15 being the most significant bit.
b16 is the least significant bit of the third byte returned by GemCore, b23 being the most significant bit.
b24 is the least significant bit of the third byte returned by GemCore, b31 being the most significant bit.

Example

GPM256 Read Command

Interpreted GPM256 source code:

```
                                ;Initialization:
                                ;CLA, INS, A1: not used
                                ;A2 = R7: location of first byte to be read
                                ;Lout = R3: number of byte to read

81        %CLR_C4               ;
71        %SET_CLK              ;Clears the internal counter
61        %CLR_CLK              ;

91        %SET_C4               ;
EF        MOV A,R7              ;Selects the first byte to be read
73        %CLK_INC8             ;

82        READ:RDL_BYTE         ;Reads one byte

F6        MOV@R0, A             ;Puts the byte in the output buffer
08        INC R0                ;
DB FB     DJNZR3, READ          ;Reads the next byte

42        %RET_OK               ;Returns the result and adds 90h 00h when
                                ;all the bytes are read
```

Formatted GemCore Command

`16h CLA INS A1 A2 Lin <DATA IN> Lout Lcode <CODE>`

`CLA` = 00h	not used.
`INS` = B0h	not used. Only for card driver compatibility.
`A1` = 00h	not used.
`A2` = XXh	location of the first byte to be read.
`Lin` = 00h	no byte to be sent to the card.
`DATA IN`	not used, empty field.
`Lout` = YYh	number of bytes to be read.
`Lcode` = 0Ch	number of bytes in the code

`CODE` = 81h 71h 61h 91h EFh 73h 82h F6h 08h DBh FBh 42h

Command:

16h *00h B0h 00h XXh 00h YYh 0Ch 81h 71h 61h 91h EFh 73h 82h F6h 08h DBh FBh 42h*

Response:

S <YY bytes DATA READ> 90h 00h

Printed in France.

GEMPLUS, B.P. 100, 13881 GEMENOS CEDEX, FRANCE.
Tel: +33 (0)4.42.36.50.00 Fax: +33 (0)4.42.36.50.90

Document Reference : E5221031/DPD04611B00