GemCore V1.10-Based Reader

Reference Manual

Version 1.1

October 1999

SPECIFIC WARNING NOTICE

All information herein is either public information or is the property of and owned solely by GEMPLUS who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of GEMPLUS's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:
- **the copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.**
- **this document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.**

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided « AS IS » without any warranty of any kind. Unless otherwise expressly agreed in writing, GEMPLUS makes no warranty as to the value or accuracy of information contained herein. The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, GEMPLUS reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

GEMPLUS HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE INFORMATION CONTAINED HEREIN, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL GEMPLUS BE LIABLE, WHETHER IN CONTRACT, TORT OR OTHERWISE, FOR ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER INCLUDING BUT NOT LIMITED TO DAMAGES RESULTING FROM LOSS OF USE, DATA, PROFITS, REVENUES, OR CUSTOMERS, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION CONTAINED IN THIS DOCUMENT.

Smart Cards and Smart Card Readers are patent protected by Innovatron and Bull CP8 and are produced by GEMPLUS under license.

MS-DOS® and Windows® are registered trademarks of Microsoft Corporation.

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: DPD14449A00

CONTENTS

List of Figures

PREFACE

This document provides information about the GemCore V1.1-Based Reader software. A detailed description of the GemCore V1.1-Based Reader hardware is provided in the GemCore Preliminary Technical Data sheet.

Audience

This document is intended for anyone wishing to develop electronic systems using a smart card interface.

Notation

By default, a numeric is expressed in decimal.

A hexadecimal number is followed by the h character. For example, the decimal value 13 expressed in hexadecimal becomes **0Dh**.

A byte B consists of eight bits $b_7b_6b_5b_4b_3b_2b_1b_0$: b_7 is the most significant (the highest) bit and b_0 the least significant (the lowest) bit:

One byte	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

A string of bytes consists of n concatenated bytes $B_nB_{n-1}….B_1B_0$: B_n is the most significant (the highest) byte and B_0 the least significant (the lowest) byte:

A string of n bytes	B_n	B_{n-1}	…	B_4	B_3	B_2	B_1	B_0

S stands for “status” in command results.

OVERVIEW

This document describes the interface between the user's application and the GemCore V1.1-Based Reader.

The GemCore V1.1-Based Reader, which consists of one programmed controller and up to nine Gemplus IC100 smart card interface chips, is designed to simplify the integration of smart card interfaces in electronic devices and manages communication with ISO 7816 1-2-3-4 compatible smart cards.

The software inside the reader is compatible with the Gemplus Reader Operating System (OROS). It implements communication protocols for the host system (GBP or TLP protocol) as well as protocols for synchronous and asynchronous smart cards.

Depending on the reader, the software may also manage hardware interfaces with, for instance, a display, a keypad or an external memory. The connection with the host system takes place via a serial asynchronous port at the TTL level.

GEMCORE PRINCIPLES

GemCore applications are developed in modules, which may also be referred to as *device handlers* or *application tasks*. Each module is identified by a number.

There are two types of device handlers in GemCore applications: those provided by GemCore, called *system device handlers* and the others, called *application device handlers*. The system device handlers are described in the *"GemCore V1.1-Based Reader Commands"* section: each command set, such as the configuration or card interface command sets, corresponds to a system device handler and each individual command corresponds to an elementary operation.

Modules

All modules have a common interface called the operation list. Each module can handle up to eight *operations* (0 to 7), and each operation performs a specific function such as reading data from a card or displaying data on an LCD. These operations make up the interface between the modules.

The modules exchange information using a request/response mechanism.

Requests contain a module number, an operation number, and generally also contain a list of one or more parameters. Requests use the following format:

<MODULE NUMBER + OPERATION NUMBER> [PARAM1] [PARAM2]…

Responses return a status code and the result(s) of the operation as specified in the request. Responses use the following format:

<STATUS CODE> [RESULT1] [RESULT2]…

The module sending the request must wait for the response.

The first two operations run by each module (operations 0 and 1) are conventional and must be as described below.

Operation 0

Operation 0 is called RUN. GemCore searches for this command at every loop. This operation does not use any parameters.

Operation 1

Operation 1 is called CTRL. It controls the behavior of the module. This operation uses at least one parameter. The default values are shown below.

PARAM1	**Meaning**
0	‘init’ - initialize device handler
1	‘end’ - end device handler execution
X	optional specific operation

***Note**: When CTRL is sent with the ‘init’ or ‘end’ parameters, no result is returned.*

Module and Operation Identifier Numbers

Module numbers are coded on one byte. The three least significant bits are always set to 0. The module number is thus always a multiple of eight.

Bit	7	6	5	4	3	2	1	0
	Module #					0	0	0

This coding enables the user to define up to 32 different modules, but it is possible to define modules with the same number. This may be useful to override or overload certain operations.

The operation number is coded in another byte. The three least significant bits are used to store the operation number; the others are always set to 0.

Bit	7	6	5	4	3	2	1	0
	0	0	0	0	0	Operation #		

An EXCLUSIVE OR (XOR) of these two bytes defines a command code.

Example : The command code 4Bh specifies operation 3 of the module 48h:
Module # 48 : 0 1 0 0 1 0 0 0
Operation #3 : 0 0 0 0 0 0 1 1
XOR : 0 1 0 0 1 0 1 1 gives command code 4Bh.

The Module List

GemCore uses a module list to find the modules making up an application. The module list is also used for the following purposes:

- To execute CTRL operations for all modules when the system is started.
- To execute the RUN operations for all modules at every loop.
- To exchange messages between modules.

GemCore scans the modules in the order specified in the list.

The module list can contain several modules with the same number. In this case, the GemCore kernel sends the command to the first module with the specified number in the list. If the module does not contain the command, it returns the status code 01 (unknown command number) to GemCore, which then sends it to the next module in the list with the same number. This feature can be used to override operations (for example, to interface with different smart cards using the same command or to modify the parameters sent to a device handler).

Each line of the module list is made up of three bytes:

BYTE0, BYTE1, BYTE2

These bytes contain the following information:

BYTE0

Bit	7	6	5	4	3	2	1	0
	Module number					Flags		

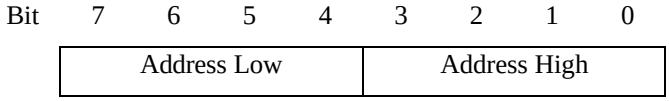
In addition to the module number, bits 7 to 3 can also hold the following values:

- 00h, indicating that the device handler is deactivated.
- FFh, indicating a link to the next part of the module list.

Bits 2 to 0 are flags. When they are set to 1, these bits indicate the following:

Bit 2 (drv_task_bit)	The module is an application task with its own context (stack).
Bit 1 (drv_filter_bit)	The module is a filter. This means that any command will be sent to operation 2 of this module.
Bit 0 (drv_run_bit)	The RUN operation is defined. If bit 2 is set to 0, the task uses the kernel context.

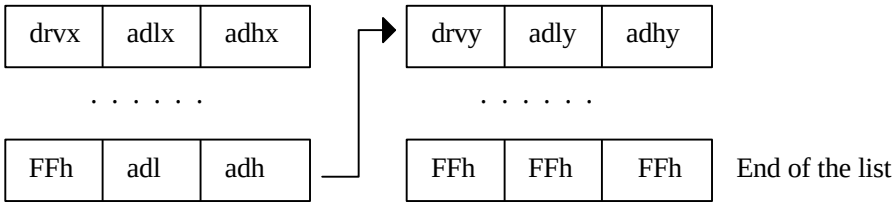
BYTE1, BYTE2



The addresses indicate either module operation table locations or the location of the next part of the module list.

An address holding the value FFFFh indicates the end of the module list only if byte 0 is set to FFh.

Example :



The GemCore Kernel

The main function of the GemCore kernel is to manage the modules. The GemCore kernel is an endless loop. When the system starts up, GemCore initializes the CTRL operation for each module. After initializing the CTRL operations, GemCore activates the RUN operation of each application task as shown in the following diagram.

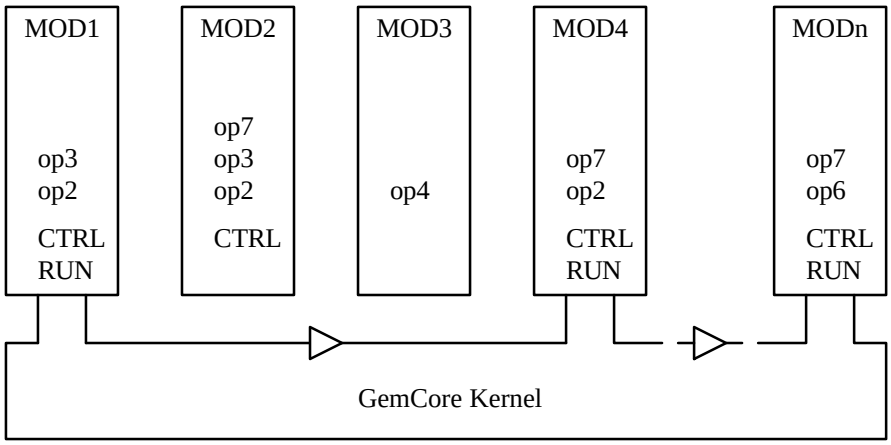


Figure 1 The GemCore Kernel

The GemCore kernel provides the system entry points. The main entry points can:

- Run an operation.
- Suspend the current operation until the next scan.

Real-Time Emulation

At each loop, GemCore pauses for a period defined in the SYS_TIC data byte. This period is set in 10-ms steps. The default value is 20 ms.

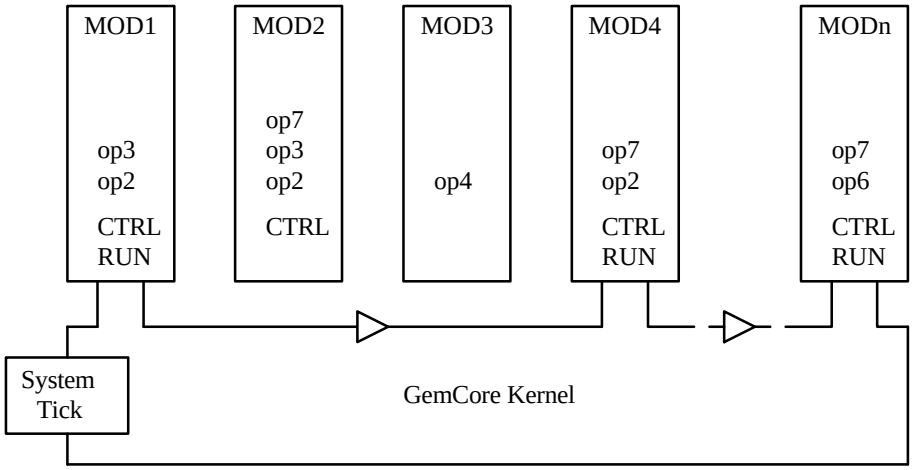


Figure 2 Real-Time Emulation

Interface Areas

GemCore uses three Interface Areas to store the modules' parameters (list, interruption vectors, etc.). The System Interface Area (SIA) is reserved for the kernel and the system device handlers. The first Application Interface Area (AIA1) is reserved for Gemplus extensions. The second Application Interface Area (AIA2) contains the modules' parameters. They are 64 bytes long (40h). The system scans the application interface areas in the following order:

1. AIA2

2. AIA1

3. SIA

AIA1 starts at address 4000h (SYS_AIA1).

AIA2 starts at location FFC0h (SYS_AIA2).

This scanning order allows the user to develop modules with the same number as a system device handler in order to override certain operations.

GEMCORE MEMORY ORGANIZATION

GemCore manages different types of memory. These memory areas are either used by the operating system or dedicated to customer applications.

The User DATA #0, User DATA #1, User DATA #2, User Application AIA2 memory areas are reserved for customer development when available on the GemCore V1.1-Based Reader.

A memory page mechanism allows the use of up to 512 Kbytes for the User DATA #0 area and 512 Kbytes for the User Application AIA2 area.

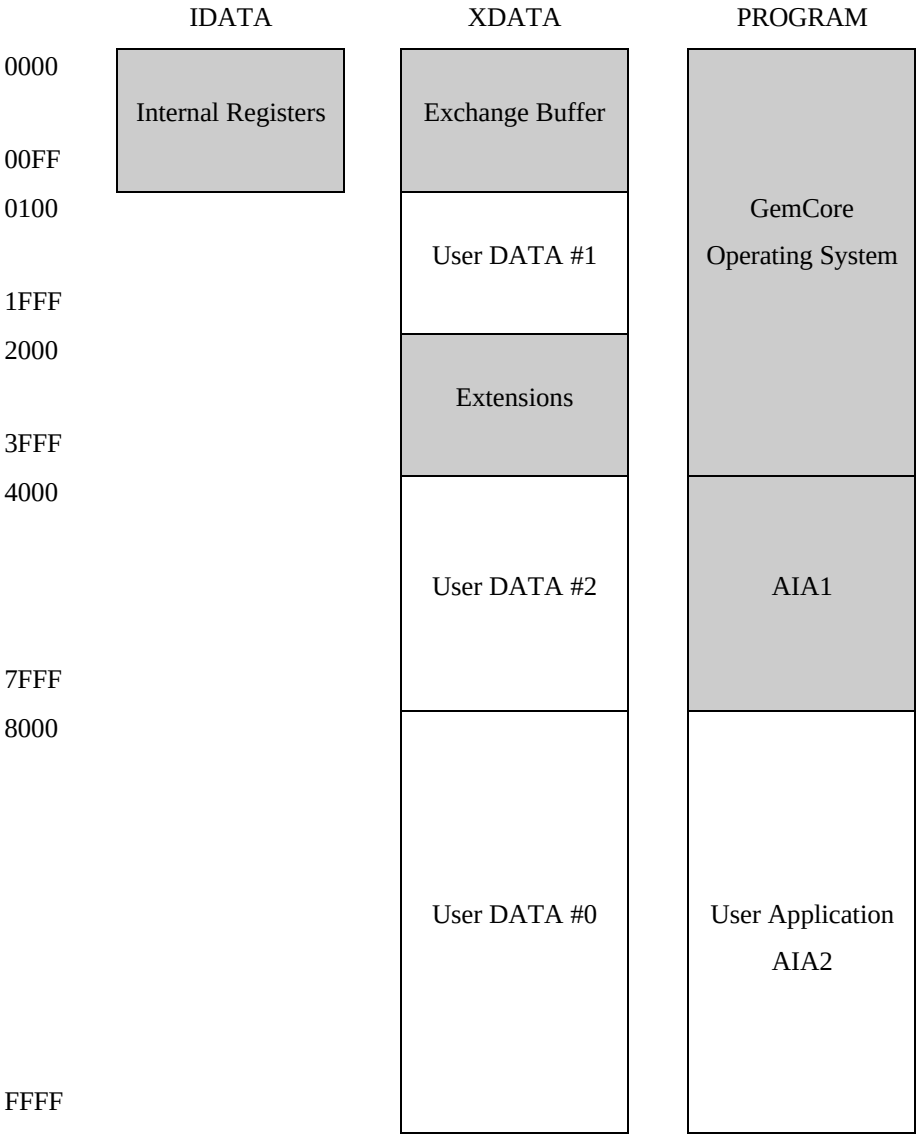


Figure 3 GemCore V1.1-Based Reader Memory Mapping.

GEMCORE APPLICATION MANAGEMENT

GemCore can manage up to four customer applications. These applications are stored in User Application Area AIA2 and can use external memory located in the User DATA #0, User DATA #1 and the User DATA #2 areas.

As a general rule, User DATA #1 and User DATA #2 areas are used as working RAM or as storage space for intermediate data.

***Note**: Depending on the reader used, the User DATA #1 and User DATA #2 areas may not be available.*

512 Kbytes of memory are available for the User Application AIA2 area and 512 Kbytes are available for area User DATA #0.

The size of the applications is predefined:

Application 1: Program 64 Kbytes, DATA 64 Kbytes
Application 2: Program 64 Kbytes, DATA 64 Kbytes
Application 3: Program 128 Kbytes, DATA 128 Kbytes
Application 4: Program 256 Kbytes, DATA 256 Kbytes

When the system is powered up, GemCore looks for the defined applications and automatically runs the first application it finds.

GEMCORE V1.1-BASED READER PROTOCOLS

All transmissions with the GemCore V1.1-Based Reader are handled by three protocol layers:

- The command layer
- The transport layer
- The physical layer

The command layer handles and interprets the GemCore V1.1-Based Reader commands. Commands are made up of a command code, data, and parameters.

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

The physical layer handles the data transmission itself.

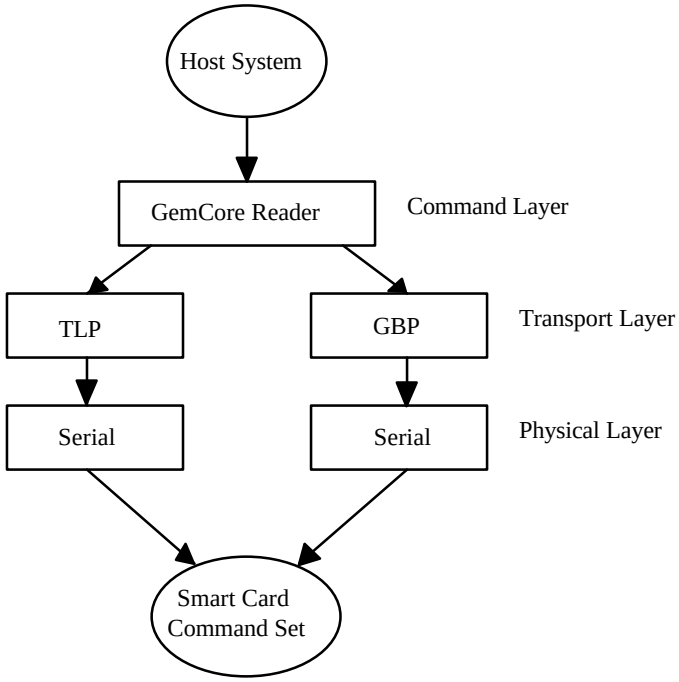


Figure 4 Three-Layer GemCore-Based Reader Protocol.

The following paragraphs describe the protocol layers in more detail.

The Command Layer

The command layer handles and interprets the GemCore V1.1-Based Reader commands. Commands are made up of a command code, data, and parameters.

Commands are sent 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 V1.1-Based Reader Interface Commands"* section describes the `CommCode`, `Parameters`, and `Data` field values for each command.

The GemCore V1.1-Based Reader anwers every command it receives with a status code which is formatted as follows:

```
|S|Data|
```

Where:

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

The Transport Layer

The transport layer handles message addressing, specifies the transmission type, and validates each transmission. The GBR (GemCore-Based Reader) transport layer can use one of two protocols: the TLP224 protocol or the Gemplus Block Protocol. The following paragraphs describe these protocols.

TLP224

TLP protocol processing consists of two steps.

Step 1

The first step is to construct the message to be transmitted. Under the TLP224 protocol, the messages have the following format:

For messages transmitted without errors:

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

Where:

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

For messages transmitted with errors:

`<NACK><LN><LRC>`

Where:

`NACK`	E0h, indicating an error occurred in the message transmission.
`LN`	00
`LRC`	E0

Step 2

During the second step, the source performs the following processes:

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

For example, to transmit the **Power Down** command which uses the command code 4Dh and no parameter under the TLP224 protocol, the following sequence would be sent:

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

The timeout between each character is 100 ms.

The Gemplus Block Protocol (GBP)

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:

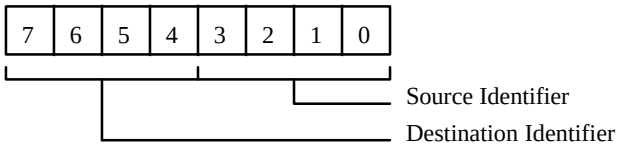
- I-Blocks (Information Blocks). I-Blocks hold the data to be exchanged between the source and the destination.
- R-Blocks (Receive Ready Block). R-Blocks hold positive or negative acknowledgments to transmissions.
- S-Blocks (Supervisory Block). S-Blocks synchronize transmissions between the source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

The data is exchanged in the following format :

NAD	PCB	LEN	DAT	EDC

Where:

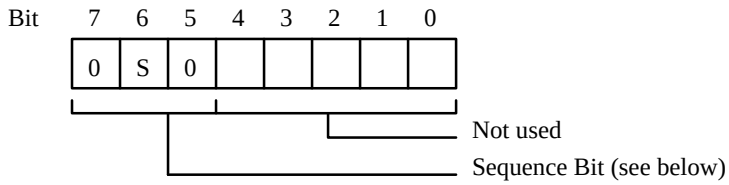
- NAD is the source and the destination identifier formatted on one byte as follows:



The GemCore V1.1-Based Reader identifier is 4 and the host system identifier is 2.

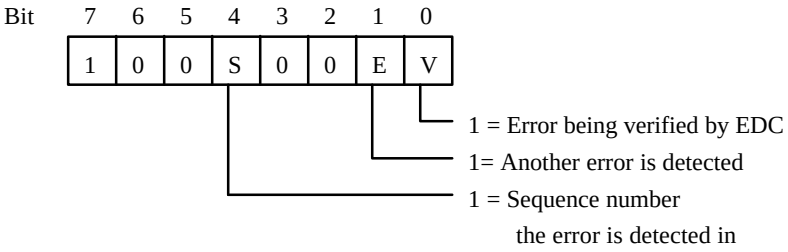
- PCB indicates the block type. Its format depends on the block type, as described below:

I-Block PCBs use the following format:



The sequence bit is set to 0 at 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 one every time it sends an information block. The GemCore V1.1-Based Reader and the host system generate sequence bit values independently.

R-Block PCBs use the following format:



S-Blocks request that the destination set the sequence bits to 0 and return a response to the source indicating that the transmission is completed.

S-Block PCBs use the following format:

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

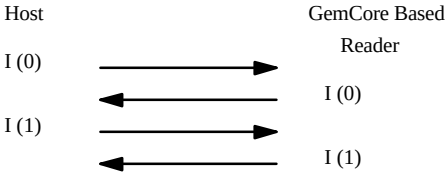
Bit	7	6	5	4	3	2	1	0	
	1	1	1	0	0	0	0	0	Resynch response

- LEN specifies on one byte the number of bytes in the INF field.
- DAT holds the data being transmitted.
- EDC is the result of an exclusive OR performed on the NAD, PCB, LEN, and DAT bytes.

Examples

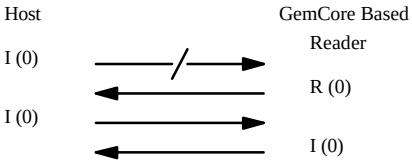
The following examples illustrate different types of transmissions under the GBP protocol.

Transmission without errors:

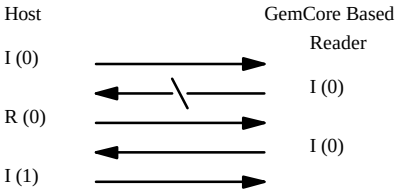


Transmission with errors:

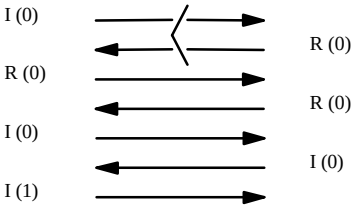
Case 1.



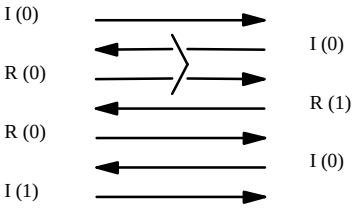
Case 2.



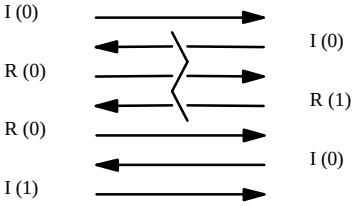
Case 3.



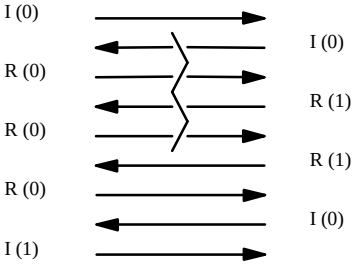
Case 4.



Case 5.



Case 6.



The 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 with transmission characteristics (such as speed and parity) which are determined by the GemCore-Based Reader configuration.

The default configuration is 9,600 Baud, eight bits, no parity and one stop bit.

GEMCORE V1.1-BASED READER COMMANDS

This section describes the GemCore V1.1-Based Reader commands. For each command it indicates:

- The function it performs,
- The syntax,
- The data it returns.

Command Format

Commands are sent to the reader 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 reader responds to every command it receives with a response formatted as follows:

```
|S|Data|
```

Where:

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

"Appendix A" lists the status codes and their meanings.

GemCore V1.1-Based Reader Configuration Commands

The GemCore V1.1-Based Reader configuration commands are used to define reader settings.

The GemCore V1.1-Based Reader configuration commands are:

- **Configure SIO Line**
- **Set Mode**
- **Set Delay**
- **Read Firmware Version**
- **Restart**
- **Restart And Run Specified Application**
- **Deselect Application Procedure**

Each command is described in the following pages.

Configure SIO line

This command sets the SIO line parity, Baud, and number of bits per character. After a power up, the line defaults to no parity, eight bits per character, 9,600 Baud and 1 stop bit.

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

Format

0Ah CB

Where:

CB is the configuration byte. Configuration flag settings are defined in the following table:

Bit	Value	Option Selected
7 to 5		Not used
4	0	No parity
	1	Even parity
3	0	Eight bits per character
	1	Seven bits per character
2 to 0	xxx	Sets the Baud with the following values: 000 RFU 001 76,800 010 38,400 011 19,200 100 9,600 101 4,800 110 2,400 111 1,200

Result

S

Set Mode

This command enables the user to disable ROS command compatibility and to define the reader operating mode (TLP or Normal). The reader defaults to ROS command compatibility enabled and TLP mode.

Notes: *Disabling ROS command compatibility disables this command. ROS command compatibility can only be enabled once again by performing a hardware reset on the reader so that the default configuration is restored.*

Disabling ROS command compatibility also disables TLP mode, regardless of the value of bit 3 (see below).

Format

`01h 00h` [OB]

Where:

[OB] Is the option selection byte.

Flag settings are described in the following table:

	Native	ROS	TLP
xxxx1xx1	*Ú*	*Ú*	*Ú*
xxxx0xx1	*Ú*	*Ú*	
xxxx0xx0 or xxxx1xx0	*Ú*		

Note: *If this byte is not sent, the reader operation mode stays unchanged, but the result is still returned.*

Result

S <mode>

Where:

[mode] Is the mode the reader is operating in. The mode is returned on one byte that indicates the operating mode as shown the following table:

	Native	ROS	TLP
00001001	*Ú*	*Ú*	*Ú*
00000001	*Ú*	*Ú*	
00000000	*Ú*		

Note: *In TLP mode, the GemCore V1.1-Based Reader adds the TA1, TB1, TC1, TD1 bytes if they are not present in the asynchronous card Answer To Reset.*

Set Delay

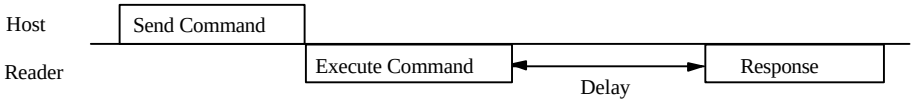
If a slow host computer is used with the GemCore V1.1-Based Reader, this command can be used to delay responses.

Format

`23h 01h 00h 67h 01h` `Delay`

Where:

`Delay` Is the response delay in ms. Enter a value between 0 and 255. When the system is powered up, the delay time defaults to 0.



Read Firmware Version

Returns the version of the firmware installed in the reader.

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

Result **S** Version

Where:

Version (GemCore-1.10) is the installed software version in ASCII (where y can be a space or an M).

OROS-compatible command:

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

Result **S** OROS-R2.99-R1.10

Restart

This command is used to reset the GemCore operating system.

All the parameters are initialized with the default values.

No result is returned.

Format `0Ch 00h 00h 00h`

Restart And Run Specified Application

This command is used to run the specified application.

If the application does not exist, an error code is returned.

Format

`0Ch 00h 00h APP`

Where:

APP	Is the number of the application to be run (1, 2, 3, or 4).

Result

S	Status

03h if the specified application does not exist.

***Note**: If APP is set to 00h, this restarts the reader.*

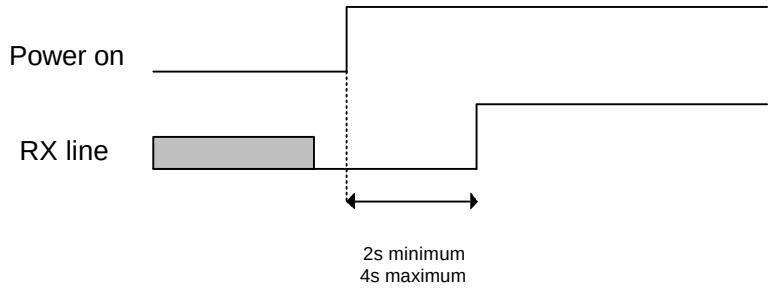
Deselect Application Procedure

This procedure instructs GemCore to run with no active application. It can be used before downloading a new customer application for a stand-alone device.

This procedure is used when disabling applications using the GemCore Tool Kit Loader.

The GemCore Reader's RX line must be set to 0 during a minimum of two seconds and a maximum of four seconds, while GemCore is powered up.

The RX line should be set to 0 before the reader is powered on.



Applications are disabled until a **Restart** command is performed or until the next power-up sequence takes place.

This procedure does not affect external memories such as the customer’s application memories.

Card Interface Commands

The card interface commands manage the communication with smart cards.

The GemCore operating system can simultaneously manage up to nine smart card interfaces. In order to limit the command set, ~~the~~ smart card interfaces are organized in one main smart card interface and ~~8~~eight auxiliary smart card interfaces.

The auxiliary smart card interface required should be selected before use.

The card interface commands are:

- **Power ~~d~~Down**
- **Power U~~u~~p**
- **ISO O~~o~~utput**
- **ISO I~~i~~nput**
- **Exchange APDU**
- **Define M~~m~~ain C~~c~~ard T~~t~~ype**
- **Define T~~t~~ype A~~a~~nd S~~s~~elect A~~a~~uxiliary C~~c~~ard**
- **Card S~~s~~tatus**
- **Directory**

Each command is described in the following pa~~ragraph~~ges.

See "*Appendix A*" for a description of status codes.

Power down

Use this command to power down the card. The GemCore V1.1-Based Reader is powered down automatically when the card is removed.

GBR Format

```
11h Main Card
19h Selected Auxiliary Card
```

ROS Format

```
4Dh 00h 00h 00h Main Card only
```

Result

```
S                Status
```

The **Power down** command always ends normally if a card is present in the reader.

If no card is inserted, the command returns the FBh "card missing" error.

Power Uup

This command powers up and resets a card.

GBR Format

12h [CFG][PTS0,PTS1,PTS2,PTS3,PCK] Main Card

1Ah [CFG][PTS0,PTS1,PTS2,PTS3,PCK] Selected Auxiliary Card

ROS Format

6Eh 00h 00h 00h Main Card only

If the CFG parameter is not specified, the card is powered with 5V, there is no PTS management and the operating mode is compatible with OROS2.2X

If the CFG parameter is specified:

XXXXXX01	Class A: Vcc for Card is 5V
XXXXXX10	Class B: Vcc for Card is 3V
XXXXXX11	Class AB: Vcc for Card is 5V or 3V
0000XXXX	Operation is compatible with OROS2.2X
0001XXXX	Reset and no PTS management. The reader stays at 9,600 baud if the card is in negotiable mode.
0010XXXX	Reset and automatic PTS management. The reader uses the highest speed proposed by the card. It switchesChange to T=1 protocol if there is a choice between T=0 and T=1.
1111XXXX	Manual PTS management. This command does not reset the card. It must be preceded by a command with the PTS parameter set to 0001XXXX. The parameters from PTS0 to PCK are sent to the card at 9,600 baud. If the card replies with PTS RESPONSE, the reader is configured using the parameters returned.

Result

S <card response>

wWhere:

<card response> iIs the card Answer To Reset (ATR).

***Note**: For cards which do not return an ATR, a default ATR is returned* (3Bh 00h 00h 00h 00h 00h).

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

R1: compatibility mode: 28h for TLP and 01h for ROS

R2: current card type

R3: ATR length

***Note:** When TLP compatibility is enabled (see "**Set Mmode**" command) the TA1, TB1, TC1 and TD1 bytes missing in the ATR are returned with their default values:*

	TA1	TB1	TC1	TD1
Asynchronous Card	11h	25h	00h	00h
Synchronous Card	00h	00h	00h	00h

Note: *When TLP compatibility is enabled, the missing bytes are returned but* ***T0 is not modified****. The syntax of the ATR is therefore not valid.*

Example:

```
3B  A0  00  81  71  27  42  00  35
```

becomes:

```
3B  A0  11  00  00  81  71  27  42  00  35
```

TCK and T0 are not valid.

ISO Output

This command sends ISO OUT commands, that is, commands which retrieve data from the card. For memory cards, specific commands formatted in the same way as ISO commands are accepted. See the chapter *"Using the GemCore V1.1-Based Reader With Memory Cards"* for a list of the respective memory card commands.

For microprocessor cards, this command can return up to 252 data bytes in one operation. Two operations are required in order to obtain 256 data bytes.

For memory cards, the length of data retrieved from the card in one application must not exceed 249 bytes.

GBR Format

```
13h CLA INS A1 A2 LN Main Card
1Bh CLA INS A1 A2 LN Selected Auxiliary Card
```

ROS Format

```
DBh CLA INS A1 A2 LN Main Card 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 concerning the card being used. The ISO header is transmitted directly to microprocessor cards (asynchronous cards) and is interpreted by the GemCore V1.1-Based Reader for Gemplus memory cards.

Result

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

Where:

`<data>` Is the data returned by the card.

If a smart card error or GemCore V1.1-Based Reader error is detected (S<>0 and S<>E7h), the GBR does not return any data.
The card may return any number of bytes up to `LN`.

If the number of data bytes to be returned is greater than 252, the first 252 bytes are contained in the <`data`> field without the SW1 SW2 bytes.

In order to obtain the rest of the response with the SW1 SW2 status bytes, the following command must be sent:

GBR Format

```
13h FFh FFh FFh FFh FFh Main card
1Bh FFh FFh FFh FFh FFh Selected auxiliary card
```

Result

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

Where:

`<data>` Is the rest of the response (`LN`-252 bytes).

***Note**: The GBR returns the error code 1Bh if a card interface command other than the above is sent.*

ISO Input

This command sends ISO IN commands, that is, commands which send data to a card. For memory cards, the GemCore V1.1-Based Reader accepts specific commands that are formatted in the same way as ISO commands. See *"Using the GemCore V1.1-Based Reader With Memory Cards"* for a list of the respective memory card commands.

For microprocessor cards, this command can send up to 248 data bytes in one operation. Two operations are required to send 255 data bytes.

For memory cards, the length of data sent to the card in one operation must not exceed 248 bytes.

GBR Format

```
14h CLA INS A1 A2 LN <data> Main Card
1Ch CLA INS A1 A2 LN <data> Selected Auxiliary Card
```

ROS Format

```
DAh CLA INS A1 A2 LN <data> Main Card 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 concerning the card being used. The ISO header is transmitted directly to microprocessor cards (asynchronous cards) and is interpreted by the GemCore V1.1-Based Reader 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 `SW1` and `SW2` bytes hold the standard status codes returned by the card. Their respective values are 90h and 00h if the operation is successful.

***Note**: SW1 and SW2 are returned with Gemplus memory cards.*

If the number of data bytes to be transmitted is greater than 248, the command below containing the last data bytes must be sent before the 'normal' ISO INPUT command containing the first 248 data bytes.

GBR Format

```
14h FFh FFh FFh FFh (LN-248) <data248.dataLN>Main Card
1Ch FFh FFh FFh FFh (LN-248) <data248.dataLN> Selected
Auxiliary Card
```

Result

```
S SW1 SW2
```

Exchange APDU

Sends a command Application Data Protocol Unit (APDU) to a card, and retrieves the response APDU. This command can only be executed on T=1 protocol cards or memory cards.

For memory cards, the GemCore-Based Reader accepts specific commands that are formatted in the same way as APDU commands. See "*Using the GemCore-Based Reader with Memory Cards*" for a list of the respective memory card commands.

GBR Format

```
15h APDU Main Card
1Dh APDU Selected Auxiliary Card
```

Where:

`APDU` is 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. Please refer to the documentation concerning the card currently used for APDU command details.

For memory cards, APDU command length cannot exceed 252 bytes.

For microprocessor cards, up to three operations are required to perform an ISO short APDU exchange of maximum length (261 bytes for APDU commands and 258 bytes for APDU responses).

Result

```
S Response APDU
```

Where:

`Response APDU` is the response APDU to the command. If the card replies in chained blocks, they are concatenated.

For memory cards, the response APDU must not exceed 251 bytes.

If the APDU command length (LC) exceeds 254 bytes, the command below containing the last part of the APDU command must be sent before the "normal" APDU exchange command containing the first 254 bytes.

GBR Format

```
15h FFh FFh FFh FFh (LC-254) <apdu255.apduLC> Main Card
1Dh FFh FFh FFh FFh (LC-254) <apdu255.apduLC> Selected
Auxiliary Card
```

If the length of the APDU response (LR) exceeds 254 bytes, the first 254 bytes of the response are returned with the status code 1Bh indicating that the command below must be sent to retrieve the last bytes of the response:

```
15h FFh FFh FFh FFh (LR-254) Main Card
1Dh FFh FFh FFh FFh (LR-254) Selected Auxiliary Card
```

APDU Format

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

APDUs can belong to one of several types, depending on the length and contents of the APDU. The GemCore V1.1-Based Reader handles the following cases:

Case 1 No command or response data.
Case 2 Short format: command data between 1 and 252 bytes and no response data.
Case 3 Short format: no command data, response data between 1 and 253 bytes.
Case 4 Short format: command data between 1 and 252 bytes, response data between 1 and 253 bytes.

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

Command Format

Commands are sent in the following format:

Header	Body		
CLA INS P1 P2	Lc	Parameters/data	Le

The fields are described below:

Header Fields

Header fields are mandatory, and are as follows:

Field Name	Length	Description
CLA	1	Instruction class.
INS	1	Instruction code. This is given in 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	Length of the data field.
Data	Lc	Command parameters or data.
Le	1	Expected length of the data to be returned.

For complete details about the header and about body field contents, refer to the documentation concerning the card currently being used.

Response Format

Responses to commands are received in the following format.

Body	Trailer
Data	SW1, SW2

The body is optional and holds any data returned by the card.

The trailer includes the following two mandatory bytes:

SW1: Status byte 1 which returns the command processing status
SW2: Status byte 2 which returns the command processing qualification

For full details about the response field contents refer to the documentation concerning the card currently used.

IFSC/IFSD

When block chaining is used, the buffer length is determined by the IFSC and IFSD parameters. The default value is 32 bytes for IFSD and IFSC (data buffer length).

If the smart card indicates an IFSC value in the ATR, the reader uses this value to send data to the card.

Define Main Card Type And Card Presence Detection

The GemCore V1.1-Based Reader does not have a smart card recognition algorithm. It is therefore necessary to define the type of card used. This command sets the card type.

***Notes**: ROS and GBR versions of this command are different. The two formats are described below.*

When the GemCore V1.1-Based Reader is reset or powered up, the card type defaults to standard microprocessor card mode (type 2).

GBR Format **17h** T [00h [P]]

ROS Format **02h** T P

Where:

T is the card type selection byte. The card type codes are as follows:

Enter this code	To use this card
01h	Other synchronous smart cards; interpreted driver.
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 (read only).
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).
0Dh	GPM276/GAM275.
0Eh	GPM271/GAM273.
0Fh	GAM226.

If the command is sent with a family number that does not match the current main card, the current main card is powered down.

P is the card presence byte. This optional parameter is used to modify the card presence indication options. When this parameter is not specified, card presence is not indicated.

Enter this code	To indicate card presence:
00h	On P1.6 (SCL line), card present = 1
01h	On P1.6 (SCL line), card present = 0
02h	On P3.1 (TxD line), card present = 1
03h	On P3.1 (TxD line), card present = 0

Result **S**

Define Type And Select Auxiliary Card

The GemCore V1.1-Based Reader does not have a smart card recognition algorithm. It is therefore necessary to define the type of card currently being used. This command sets the auxiliary card type and selects the auxiliary card number.

***Note**: When the GemCore V1.1-Based Reader is reset or powered up, the auxiliary card type defaults to standard microprocessor card mode (type 2) and auxiliary card number 1 is selected.*

Commands sent to the auxiliary card are executed by the auxiliary card currently selected.

Switching from one auxiliary card to another does not affect the status of unselected auxiliary cards.

GBR Format **1Fh** `T N`

Where:

`T` is the card type selection byte. The card type codes are as follows:

Enter This Code	To Use This Card
01h	Other synchronous smart cards; interpreted driver.
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 (read only).
04h	GPM416/GPM896 in Standard Mode.
06h	GFM2K/GFM4K.
07h	GPM103.
08h	GPM8K(SLE4418/4428).
09h	GPM2K(SLE4432/4442 or PCB2032/2042).
0Dh	GPM276/GAM275.
0Eh	GPM271/GAM273.
0Fh	GAM226.

If the command is sent with a family number that does not match the current auxiliary card, the current auxiliary card is powered down.

`N` is the auxiliary Card Number (1 to 8).

Result **S**

Card Status

This command is used to obtain the status of the main card interface or that of the auxiliary card. It returns information indicating:

- The type of card currently being used,
- Card presence,
- The power supply value,
- The card power status,
- The communication protocol (T=0 or T=1),
- The speed parameters between the card and the reader.

GBR Format

17h Main Card
1Fh Selected Auxiliary Card

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	T=0 protocol
	XXXX1XXX	T=1 protocol
TYPE	Activated Card type	
CNF1 CNF2 CNF3 CNF4	CNF1=TA1 (FI/DI) CNF2=TC1 (EGT) CNF3=WI CNF4=00	T=0 Card as per ISO 7816/3
CNF1 CNF2 CNF3 CNF4	CNF1=TA1 (FI/DI) CNF2=TC1 (EGT) CNF3=IFSC CNF4=TB3 (BWI/CWI)	T=1 Card as per ISO 7816/3
CNF1 CNF2 CNF3 CNF4	CNF1=00 CNF2=00 CNF3=00 CNF4=00	Synchronous smart cards

Directory

This command is used to obtain the types of cards handled as well as the release number and the characteristics for each card driver.

GBR Format **17h 00h**

Result **S [TYPE, CMD, REV] ... [TYPE, CMD, REV]**

Where:

Type	Card type (e.g.: 02h asynchronous card)
CMD:	00: ISO IN/OUT
	01: APDU
	02: ISO IN/OUT and APDU
REV:	Card driver release (2 bytes)

Reader Memory Management Commands

Reader memory management commands are:

- **Read Memory**
- **Write Memory**
- **Erase Flash Memory**
- **Select External Memory Page**
- **Read CPU Port**
- **Write CPU Port**

Each command is described in the following pages.

See "*Appendix A*" for a description of status codes.

Read Memory

Reads the contents of all the memory areas which can be addressed by the reader. This command is only operative provided that the memory under consideration is not read-protected.

Format

22h `Type [Page] ADH ADL LN`

Where :

`Type` is the type of memory to be read, mapped as follows:

b7	b6	b5	b4	b3	b2	b1	b0
0	P	0	0	T	T	T	T

P is the page parameter flag. If set, this bit specifies that the optional `Page` parameter is present.

TTTT is the type of memory read.

Value	Memory Type
0001	IDATA (Internal CPU data memory)
0010	XDATA (External data memory)
0101	Code memory
0110	XDATA (External data memory, FLASH Atmel)

`Page` is the optional byte indicating the XDATA and CODE page to be selected before reading can occur. If this parameter is not present, the page currently selected is read. See *"Select External Memory Page"* for further details.

Note: *The current page is not modified.*

`ADH,ADL` is the 16-bit address of the first byte to be read. ADH is the most significant byte and ADL is the least significant byte.

`LN` is the length of data to be read in bytes.

Result

S <data bytes>

Write Memory

Writes in all memory areas which can be addressed by the reader. This command is only operative provided that the memory under consideration is not write-protected.

Format

23h `Type [Page] ADH ADL LN <data>`

Where :

`Type` is the type of memory to be written, mapped as follows:

b7	b6	b5	b4	b3	b2	b1	b0
E	P	0	0	T	T	T	T

E is the FLASH/EEPROM memory type flag (only for XDATA or CODE memory types). If set, this bit indicates a FLASH-type memory.

P is the page parameter flag. If set, this bit specifies that the optional `Page` parameter is present.

TTTT is the type of memory to be read.

Value	Memory Type
0001	IDATA (Internal CPU data memory)
0010	XDATA (external data memory)
0101	CODE memory
0110	Xdata (External data memory, Flash Atmel)

Examples :

1) 23h 02h 80h 00h 01h <Data>: Writes one byte in the RAM memory, in the XDATA area, at location 8000h.
2) 23h 85h 80h 00h 01h <Data>: Writes one byte in the FLASH memory, in the CODE area, at location 8000h.
3) 23h 86h 80h 00h 01h <Data>: Writes one byte in the FLASH Atmel memory, in the XDATA area, at location 8000h.

`Page` is the optional byte indicating the XDATA and CODE page to be selected before the **Write** command can be performed. If this parameter is absent, the currently-selected page is written. See *"Select External Memory Page"* for details.

Notes: *The current page is not modified.*
Writing to the FLASH Atmel memory can only take place on readers with 256 bytes of RAM at XDATA address 100h.

Blocks to be written to the FLASH Atmel must not exceed 256 bytes in length. Moreover, the block to be written must not overlap two consecutive sectors of the FLASH.

`ADH,ADL` define the 16-bit address of the first byte of memory to be written. ADH is the most significant byte and ADL is the least significant byte.

`LN` is the length of data to be written in bytes.
`<data>` is the data to be written.

Result

S

Memory Read And Write Protection

Both the program memory and the data memory can be protected against read or write access. Two 8-byte codes are used for this purpose: the first code protects the program memory, and the second protects the data memory.

When the program memory is protected:

- The 22 01 ADH ADL LNG command returns error code 1F.
- The 22 X5 ADH ADL LNG command returns error code 1F.
- The 23 01 ADH ADL LNG <DATA> command returns error code 1F.
- The 23 X5 ADH ADL LNG <DATA> command returns error code 1F.
- The 26 85 ADH ADL DATA command returns error code 1F.

When the data memory is protected:

- The 22 X2 ADH ADL LNG command returns error code 1F.
- The 23 X2 ADH ADL LNG <DATA> command returns error code 1F.
- The 26 82 ADH ADL DATA command returns error code 1F.

In order to be efficient, the data memory protection must be used along with a protected program memory.

Memory protection codes are located in the application program memory area and must be downloaded with the application software.

The program memory protection code is located between address FFB0 and address FFB7.

The data memory protection code is located between address FFA0 and address FFA7.

In order to be validated, the eight bytes of the protection code must be followed by eight bytes representing the complementary code.

Example :
FFA0: 11 22 33 44 55 66 77 88 FF FF FF FF FF FF FF FF
FFB0: 01 02 03 04 05 06 07 08 FE FD FC FB FA F9 F8 F7

The access to the data memory is free (that is, its code is not validated). The program memory is protected.

In order to enable read or write access to a protected area, the following write command must be used:

Code memory: 23 X5 FF B0 08 < 8 byte code >
Data memory: 23 X2 FF A0 08 < 8 byte code >

In all cases, the reader response is the status code **1F**.

If the code presented is correct, the next read or write command will be executed.

Erase Flash Memory

Erases all or part of the contents of the Flash memory. This command is only operative provided that the memory is not considered write-protected.

***Note**: Execution of this command can last up to one minute.*

Format

26h Type [Page] ADH ADL <CODE>

Where :

Type is the type of memory to be written, mapped as follows:

b7	b6	b5	b4	b3	b2	b1	b0
1	P	0	0	T	T	T	T

P is the page parameter flag. If set, this bit specifies that the optional Page parameter is present.

TTTT is the type of memory to erase.

Value	Memory type
0010	XDATA memory
0101	CODE memory
0110	XDATA memory (Flash Atmel)

Page is an optional byte indicating the XDATA and CODE page to be selected before writing can take place. If this parameter is absent, the page currently selected is erased. See the *"Select External Memory Page"* command for details.

***Notes**: The current page is not modified.*

For Flash Atmel, it is not necessary to erase the memory before writing. If memory erasing is requested, the entire memory must be erased.

ADH, ADL define the 16-bit start address for the erase command. ADH is the most significant byte and ADL is the least significant byte.

<CODE> is the erase command code.

It is 10h if the whole memory is to be erased (the address should then be D555h), or 30h if one sector only is to be erased (the address should then be the sector address).

Result

S

Example 1:

The following commands erase the data stored in the AMD 29F010 Flash memory used for program storage, starting from address 8000h. This memory is organized into eight sectors of 16 Kbytes each.

Memory configuration:

	Page#0	Page#1	Page#2	Page#3
8000h BFFFh	Sector 1	Sector1	Sector 1	Sector 1
C000h FFFFh	Sector 2	Sector 2	Sector 2	Sector 2

Erase chip command: 26h 85h D5h 55h 10h
Erase first sector command: 26h 85h 80h 00h 30h
Erase second sector command: 26h 85h C0h 00h 30h
Erase first sector in code page 2 command: 26h C5h 20h 80h 00h 30h

Example 2:

The following commands erase the data stored in the AMD 29F040 Flash memory used for program storage starting from address 8000H. This memory is organized into eight sectors of 64 Kbytes each.

Erasing one sector erases two pages of 32 Kbytes each.

Memory configuration:

	Page#0	Page#1		Page#6	Page#7
8000h FFFFh	Sector 0		...	Sector 3	

Erase chip command: 26h 85h D5h 55h 10h
Erase sector 0 command: 26h 85h 80h 00h 30h

Example 3:

The following command erases the entire Flash Atmel memory:

26h 86h D5h 55h 10h

Select External Memory Page

GemCore can manage up to sixteen 32-Kbyte pages of CODE memory, and sixteen 32-Kbyte pages of XDATA memory. This command selects the active page.

When 512 Kbytes of memory are used, the physical memory is split into two blocks of eight pages each.

- Application #1 is mapped on pages 0 and 1 of the first block.
- Application #2 is mapped on pages 2 and 3 of the first block.
- Application #3 is mapped on pages 4, 5, 6 and 7 of the first block.
- Application #4 is mapped on pages 0 to 7 of the second block.

By default, Application #1, CODE Page 0 and XDATA Page 0 are selected after the GBR has been powered up.

Format

27h `Page [App]`

Where :

`Page` is the byte indicating the XDATA and CODE page to select in the following format:

b7	b6	b5	b4	b3	b2	b1	b0
0	C	C	C	0	D	D	D

Bits 2 to 0: indicate the XDATA page to select.
Bit 3: not used.
Bits 6 to 4: indicate the CODE page to select.
Bit 7: not used.

`App` is an optional byte indicating the application number.

Memory organization:

8000h FFFFh	Page#0	Page#1	Page#2	Page#3	...	Page#7
XDATA	xxxxx000	xxxxx001	xxxxx010	xxxxx011	...	xxxxx111
CODE	x000xxxx	x001xxxx	x010xxxx	x011xxxx	...	x111xxxx

Result

S

Read CPU Port

Reads the state of a CPU port.

Format

24h PORT

Where:

PORT is the number of the port to be read as defined in the following table:

Value	Port
00	Port0- P0
01	Port1-P1
02	Port2-P2
03	Port3-P3

Result

S Value

Where:

Value is the value read from the specified CPU port.

Write CPU Port

Writes to a CPU port.

Format **25h** PORT VALUE

Where:

PORT is the number of the port to be written to, as defined in the following table:

Value	Port
00	Port0-P0
01	Port1-P1
02	Port2-P2
03	Port3-P3

VALUE is the value to be written to the CPU port.

Result **S**

LCD Commands

The LCD commands are used to control the LCD. They must be used with LCD modules that are compatible with the HITACHI HD 44780 LCD Controller.

LCD commands are:

- **Init The LCD**
- **Display Character String**
- **Display Character**
- **Send LCD Command**

Each command is described in the following pages.

See "*Appendix A*" for a description of status codes.

Init The LCD

Initializes the LCD.

Format **2Ah**

Result **S**

Display Character String

Displays a string of characters on the LCD.

Format

2Bh [POS] CHARS

Where:

[POS] is the beginning of the character string. This parameter starts at 80h for character 1 line 1, 81h for character 2 line 1, C0h for character 1 line 2 and so on. If this byte is omitted, the character string is displayed at the current cursor position. Bit 7 of this byte must always be set to 1.

CHARS is the character string to be displayed in ASCII.

Result

S

Display Character

Displays a character on the LCD at the current cursor position.

Format **2Ch** CHAR

Where:

CHAR is the character to be displayed in ASCII.

Result **S**

Send LCD Command

Sends an LCD control command.

Format

2Dh COMCODE

Where:

COMCODE is one of the command codes listed below:

Command code	Action
01h	Clears the LCD.
02h	Cursor home.
04h	Moves the cursor to the left after a **Display Character** command.
05h	Moves the text to the right after a **Display Character** command.
06h	Moves the cursor to the right after a **Display Character** command.
07h	Moves the text to the left after a **Display Character** command.
08h	LCD off.
0Ch	LCD on and no cursor.
0Dh	LCD on and blink character at cursor position.
0Eh	LCD on and display fixed cursor.
0Fh	LCD on and display blinking cursor.
10h	Moves the cursor to the left.
14h	Moves the cursor to the right.
18h	Moves the text to the left.
1Ch	Moves the text to the right.

Result

S

Keypad and Buzzer Commands

GemCore can control a 4x4 keypad and a buzzer with the following commands:

- **Set Key Press Timeout**
- **Sound Buzzer**

Each command is described in the following pages.

See *"Appendix A"* for a description of status codes.

Set Key Press Timeout

Sets the number of seconds the reader waits for a key to be pressed and switches the 25 ms key tone on and off.

Format

32h TIME BEEP

Where:

TIME is the number of seconds the reader waits for a key to be pressed, in units of 100 ms. For example, 07h specifies 700 ms.

BEEP switches the key tone on/off. 00h switches it off and 01h switches it on.

Result

S KEY

Where:

KEY is the code of the key pressed (before the time-out). The following table lists the key codes:

Key	Code	Key	Code	Key	Code	Key	Code
1	11h	2	21h	3	31h	F1	41h
4	12h	5	22h	6	32h	F2	42h
7	13h	8	23h	9	33h	F3	43h
<	14h	0	24h	>	34h	F4	44h

Sound Buzzer

Sounds the buzzer and specifies its frequency and duration.

Format

33h `TIME [FREQ]`

Where:

`TIME` is the duration of the buzzer. The units of time are a function of the frequency.

`[FREQ]` is the frequency of the buzzer (between 1,183 Hz and 68,267 Hz). This is an optional parameter. If it is omitted, the sound frequency defaults to 3,600 Hz.

The following formulas can be used to determine approximate values for these parameters:

`TIME` = T(ms) * N(Hz) / 36000
`[FREQ]` = 307200 / N(Hz) – 4

Result

S

Real Time Clock Commands

GemCore can read and update the date and time stored in the reader's clock with the following commands:

- **Read Date and Time**
- **Update Date And Time**

Each command is described in the following pages.

These commands must be used with Real-Time Clock chips which are compatible with the OKI MSM6242 chip.

Read Date And Time

Reads the real-time clock date and time.

Format **3Ah**

Result **S** YEAR MONTH DAY HOUR MINUTE SECOND

Where:

YEAR	Is the year value, in BCD.
MONTH	Is the month value, in BCD.
DAY	Is the day value, in BCD.
HOUR	Is the hour value, in BCD.
MINUTE	Is the minute value, in BCD.
SECOND	Is the second value, in BCD.

For example, November 25, 1999, 17:15:00 is coded 99 11 25 17 15 00.

Update Date and Time

Updates the real-time clock date and time.

Format

3Bh YEAR MONTH DAY HOUR MINUTE SECOND

Where:

YEAR	Is the new year value, in BCD.
MONTH	Is the new month value, in BCD.
DAY	Is the new day value, in BCD.
HOUR	Is the new hour value, in BCD.
MINUTE	Is the new minute value, in BCD.
SECOND	Is the new second value, in BCD.

Note: *A value must be entered for all the above fields.*

GCR410 Control Commands

The following commands are only available on GCR410 readers. Some of the commands have no effect but are required in order to ensure compatibility with GCR400 products.

The GCR410 reader simulates a GCR400 with an external power supply.

GCR410 commands are:

- **GCR410 Set Timeout**
- **GCR410 Refresh**
- **GCR410 Power Down**
- **GCR410 LED Management**
- **GCR410 Status**

Each command is described in the following pages.

GCR410 Set Timeout

This command is only available on GCR410 readers.

This command has no effect; it is only used for GCR400 compatibility.

Format `52h T`

Result `S = B0h`

GCR410 Refresh

This command is only available on GCR410 readers.

It has no effect; it is only used for GCR400 compatibility.

Format **53h**

Result **S**

GCR410 Power Down

This command is only available on GCR410 readers.

It has no effect; it is only used for GCR400 compatibility.

Format **54h**

Result **S = B0h**

GCR410 LED Management

This command is only available on GCR410 readers.

It controls the LED.

Format

55h LED

Where:

LED = 00h : LED Off

LED = 01h : LED On

LED = 02h : Default value (the LED blinks when the smart card is powered down and comes on when the smart card is powered up).

Result

S

GCR410 Status

This command is only available on GCR410 readers.

It returns the GCR410 status.

Format **56h**

Result **S Status 00h**

Where:

`Status` is the current reader status

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	1	0	LED1	LED0

Bits 1 and 0: indicate the LED status
00 : LED Off
01 : LED On
10 : Default value (the LED blinks when the smart card is powered down and comes on when the smart card is powered up).

USING THE GEMCORE V1.1-BASED READER WITH MICROPROCESSOR CARDS

The GemCore V1.1-Based Reader handles ISO 7816-3 T=0 and T=1 protocol microprocessor cards. The following section describes the implementation of these standards.

Clock Signal

The GemCore V1.1-Based Reader 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 (that is above ISO specifications, for cards which can operate at this frequency).

The operating mode is specified while selecting the card type with the **Define card type** command. Card type 02h should be selected for standard mode and card type 12h for double-speed mode.

Global Interface Parameters

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

TA1

The GemCore V1.1-Based Reader interprets this parameter to match its communication rate with that of the card, according to 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 ATR is 9909.68 baud in the standard mode and 19819.35 baud in double-speed mode.

After receiving the ATR, the GemCore V1.1-Based Reader selects the communication rate according to TA1. Tables **Error! Reference source not found.**1 and 2 show the clock rate conversion factors, the bit rate conversion factors, and the selected baud according to TA1 values for both the standard mode and the double-speed mode.

***Note**: The TA1 values handled by the GemCore V1.1-Based Reader are shaded in Tables 1 and 2.*

TB1 and TB2

The Vpp option is not available on the GemCore V1.1-Based Reader. 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.

D=	1		2		4		8		12		16		20		32	
F=	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)
372	**01**	9 909.68	**02**	19 819.35	**03**	39 638.71	**04**	79 277.42	**08**	-	**15**	158 554.84	**09**	-	**06**	-
372	**11**	9 909.68	**12**	19 819.35	**13**	39 638.71	**14**	79 277.42	**18**	118 916,13	**15**	158 554.84	**19**	-	**16**	-
558	21	-	**22**	13 212.90	**23**	26 425.81	**24**	52 851.61	**28**	79 277,42	**25**	105 703.23	**29**	-	**26**	-
744	31	-	**32**	9 909.68	**33**	19 819.35	**34**	39 638.71	**38**	-	**35**	79 277.42	**39**	-	**36**	-
1116	41	-	42	-	**43**	13 212.90	**44**	26 425.81	**48**	39 638,71	**45**	52 851.61	**49**	-	**46**	-
1488	51	-	52	-	**53**	9 909.68	**54**	19 819.35	**58**	-	**55**	39 638.71	**59**	-	**56**	-
1860	61	-	62	-	63	-	**64**	15 855.48	**68**	-	**65**	31 710.97	**69**	39 638,71	**66**	-
512	91	-	**92**	14 400.00	**93**	28 800.00	**94**	57 600.00	**98**	86 400,00	**95**	115 200.00	**99**	-	**96**	-
768	A1	-	A2	-	**A3**	19 200.00	**A4**	38 400.00	**A8**	57 600,00	**A5**	76 800.00	**A9**	-	**A6**	-
1024	B1	-	B2	-	**B3**	14 400.00	**B4**	28 800.00	**B8**	-	**B5**	57 600.00	**B9**	-	**B6**	115 200,00
1536	C1	-	C2	-	C3	-	**C4**	19 200.00	**C8**	28 800,00	**C5**	38 400.00	**C9**	-	**C6**	76 800,00
2048	D1	-	D2	-	D3	-	**D4**	14 400.00	**D8**	-	**D5**	28 800.00	**D9**	36 000,00	**D6**	57 600,00

Table 1. TA1 Values Handled in Standard Mode (Frequency: 3.6864 MHz)

D=	1		2		4		8		12		16		20		32	
F=	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)	TA1	Rate (bd)
372	**01**	19819.35	**02**	39 638.71	**03**	79 277.42	**04**	158 554.84	**08**	-	**05**	-	**09**	-	**06**	-
372	**11**	19819.35	**12**	39 638.71	**13**	79 277.42	**14**	158 554.84	**08**	-	**15**	-	**19**	-	**16**	-
558	**21**	13 212.90	**22**	26 425.81	**23**	52 851.61	**24**	105 703.23	**18**	-	**25**	-	**29**	-	**26**	-
744	**31**	9 909.68	**32**	19 819.35	**33**	39 638.71	**34**	79 277.42	**28**	-	**35**	-	**39**	-	**36**	-
1116	41	-	**42**	13 212.90	**43**	26 425.81	**44**	52 851.61	**38**	-	**45**	-	**49**	-	**46**	-
1488	51	-	**52**	9 909.68	**53**	19 819.35	**54**	39 638.71	**48**	-	**55**	-	**59**	-	**56**	-
1860	61	-	62	-	**63**	15 855.48	**64**	31 710.97	**58**	-	**65**	-	**69**	-	**66**	-
512	**91**	14 400.00	**92**	28 800.00	**93**	57 600.00	**94**	115 200.00	**68**	-	**95**	-	**99**	-	**96**	-
768	A1	-	**A2**	19 200.00	**A3**	38 400.00	**A4**	76 800.00	**98**	-	**A5**	-	**A9**	-	**A6**	-
1024	B1	-	**B2**	14 400.00	**B3**	28 800.00	**B4**	57 600.00	**A8**	-	**B5**	-	**B9**	-	**B6**	-
1536	C1	-	C2	-	**C3**	19 200.00	**C4**	38 400.00		-	**C5**	-	**C9**	-	**C6**	-
2048	D1	-	D2	-	**D3**	14 400.00	**D4**	28 800.00		-	**D5**	-	**D9**	-	**D6**	-

Table 2. TA1 Values Handled in Double-Speed Mode (Frequency: 7.3728 Mhz)

Communication Protocols

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

Value	Protocol
0	T=0
1	T=1

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

T=0 Protocol

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

To send instructions to a T=0 microprocessor card, the ISO INPUT and ISO OUTPUT commands are used.

T=1 Protocol

To send instructions to a T=1 microprocessor card, the **Exchange APDU** command is used. The T=1 specific interface bytes are interpreted as per clause 9 of the ISO 7816-3 standard. These bytes are TA3, TB3, TC3.

TA3 codes the Information Field Size of the card (IFSC). The default value is 32 bytes.

TB3 codes the BWI (Block Writing Time Integer) and the CWI (Character Waiting Time Integer).

TC3 defines the Error Detection Code (EDC) type.

USING THE GEMCORE-BASED READER WITH MEMORY CARDS

Memory cards cannot interpret smart card instructions in the same way as ISO 7816-3 microprocessor cards can.

T=0 formatted instructions are therefore interpreted and converted into the appropriate timing sequences required to control the memory cards listed in the tables below. For further details, refer to the relevant card documentation.

These instructions are send to the reader using the **ISO Input**, **ISO Output**, or **APDU Exchange** commands.

GPM256	**Card Type = 03h**
Read Byte:	(ISO OUT) 00h B0h 00h (Start Address) (Read Length)

GPM416	**Card Type = 04h**
Read Byte:	(ISO OUT) 00h B0h 00h (Start Address) (Read Length)
Write Byte:	(ISO IN) 00h D0h 00h (Start Address) (Write Length) (Data, .., Data)
Erase Word	(ISO IN) 00h DEh (Number of Word) (Start Address) 00h
Present Card Secret Code:	(ISO IN) 00h 20h 04h 08h 02h (Code2, Code1)
Present Erase Secret Code:	(ISO IN) 00h 20h 40h 28h 04h (Code4, .., Code1)
Change Fuse State:	(ISO IN) 00h D4h 00h 00h 00h

GPM896	**Card Type = 04h**
Read Byte:	(ISO OUT) 00h B0h 00h (Start Address) (Read Length)
Write Byte:	(ISO IN) 00h D0h 00h (Start Address) (Write Length) (Data, .., Data)
Erase Word	(ISO IN) 00h DEh (Number of Word) (Start Address) 00h
Present Card Secret Code:	(ISO IN) 00h 20h 04h 0Ah 02h (Code2, Code1)
Present Erase Secret Code #1:	(ISO IN) 00h 20h 00h 36h 06h (Code6, .., Code1)
Present Erase Secret Code #2:	(ISO IN) 00h 20h 80h 5Ch 04h (Code4, .., Code1)
Change Fuse State:	(ISO IN) 00h D4h 00h 00h 00h

GPM103	**Card Type = 07h**
Read Byte:	(ISO OUT) 00h B0h 00h (Start Address) (Read Length)
Write Byte:	(ISO IN) 00h D0h 00h (Start Address) (Write Length) (Data, .., Data)
Read Counter Value:	(ISO OUT) 00h B2h 05h 08h 02h
Write New Counter Value:	(ISO IN) 00h D2h 05h 08h 02h (Value MSB, Value LSB)
Erase and Write Carry:	(ISO IN) 00h E0h 01h (Counter Address) 00h

GAM226	**Card Type = 0Fh**
Read Byte:	(APDU) 00h B0h 00h (Address) (Read Length)
Write Byte:	(APDU) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Erase and Write Carry:	(APDU) 00h E0h 01h (Address)
Present Card Secret Code:	(APDU) 00h 20h 00h 00h 03h (Code3, Code2, Code1)
Authenticate:	(APDU) 00h 88h 01h A0h 06h (Alea6, .. , Alea1) 02h
Restore:	(APDU) 00h D4h 00h 00h

GPM271	**Card Type = 0Eh**
Read Byte:	(APDU) 00h B0h 00h (Address) (Read Length)
Write Byte:	(APDU) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Erase and Write Carry:	(APDU) 00h E0h 01h (Address)
Present Card Secret Code:	(APDU) 00h 20h 00h 00h 03h (Code3, Code2, Code1)
Restore:	(APDU) 00h D4h 00h 00h
Blow Fuse:	(APDU) 00h DAh 00h 00h

GAM273	**Card Type = 0Eh**
Read Byte:	(APDU) 00h B0h 00h (Address) (Read Length)
Write Byte:	(APDU) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Erase and Write Carry:	(APDU) 00h E0h 01h (Address)
Present Card Secret Code:	(APDU) 00h 20h 00h 00h 03h (Code3, Code2, Code1)
Authenticate:	(APDU) 00h 88h 00h 00h 04h (Alea4, Alea3, Alea2, Alea1) 01h
Restore:	(APDU) 00h D4h 00h 00h
Blow Fuse:	(APDU) 00h DAh 00h 00h

GPM276	**Card Type = 0Dh**
Read Byte:	(APDU) 00h B0h 00h (Address) (Read Length)
Write Byte:	(APDU) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Erase and Write Carry:	(APDU) 00h E0h 01h (Address)
Present Card Secret Code:	(APDU) 00h 20h 00h 00h 03h (Code3, Code2, Code1)
Restore:	(APDU) 00h D4h 00h 00h
Blow Fuse:	(APDU) 00h DAh 00h 00h

GAM275	**Card Type = 0Dh**
Read Byte:	(APDU) 00h B0h 00h (Address) (Read Length)
Write Byte:	(APDU) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Erase and Write Carry:	(APDU) 00h E0h 01h (Address)
Present Card Secret Code:	(APDU) 00h 20h 00h 00h 03h (Code3, Code2, Code1)
Authenticate:	(APDU) 00h 88h 00h 00h 04h (Alea4, Alea3, Alea2, Alea1) 01h
Restore:	(APDU) 00h D4h 00h 00h
Blow Fuse:	(APDU) 00h DAh 00h 00h

GFM2K/4K	**Card Type = 06h**
Read Byte Area	(ISO OUT) 00h B0h (AddressH) (AddressL) (Read Length)
Write Byte Area	(ISO IN) 00h D0h (AddressH) (AddressL) (Write Length) (Data, .., Data)

GPM2K	Card Type = 09h
Read Data Area	(ISO OUT) 00h B0h 00h (Address) (Read Length)
Write Data Area	(ISO IN) 00h D0h 00h (Address) (Write Length) (Data, .., Data)
Read Protection Area	(ISO OUT) 00h B0h 80h 00h 04h
Write Protection Area	(ISO IN) 00h D0h 80h (Address) (Write Length) (Data, .., Data)
Read Security Area	(ISO OUT) 00h B0h C0h 00h 04h
Write Security Area	(ISO IN) 00h D0h C0h (Address) (Write Length) (Data, .., Data)
Present Card Secret Code:	(ISO IN) 00h 20h 00h 00h 03h (Code3, Code2, Code1)

GPM8K	Card Type = 08h
Read Data Area	(ISO OUT) 00h B0h (AddressH) (AddressL) (Read Length)
Write Data Area	(ISO IN) 00h D0h 00h (AddressH) (AddressL) (Write Length) (Data,..,Data)
Present Card Secret Code:	(ISO IN) 00h 20h 00h 00h 02h (Code2, Code1)
Read Protection Area	(ISO OUT) 00h B0h (80h + AddressH) 00h 20h
Write Protection Area	(ISO IN) 00h D0h (80h + AddressH) (AddressL) 01h (Data)
Read Security Area	(ISO OUT) 00h B0h C0h 00h 03h
Write Security Area	(ISO IN) 00h D0h C0h (Address) (Write Length) (Data, .., Data)

Table 3. Summary of the Memory Card Commands

APPENDIX A - STATUS CODES

The status codes returned the cards are listed in the table below.

Code	Meaning
01h	Unknown driver or command.
02h	Operation impossible with this driver.
03h	Incorrect number of arguments.
04h	Reader command unknown. The first byte of the command is an invalid command code.
05h	Response too long for the buffer.
10h	Response error at the card reset. The first byte of the response (TS) is not valid.
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 powered down. A power up command must be sent to the card before any other operation.
1Bh	A command has been sent with an incorrect number of parameters.
1Ch	Overlap on writing to the Flash memory.
1Dh	The TCK check byte is incorrect in a microprocessor card. Answer To Reset.
1Eh	An attempt has been made to write to write-protected external memory.
1Fh	Incorrect data has been sent to the external memory. This error is returned after a write check during a downloading operation. Can occur if the memory is protected.
A0h	Error in the card reset response, such as unknown exchange protocol, or TA1 byte not recognized. The card is not supported. The card Answer To Reset 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 timing-out.
A3h	Parity error during a microprocessor exchange. This error only occurs after several unsuccessful attempts to resend.
A4h	Card has aborted chaining (T=1).
A5h	Reader has aborted chaining (T=1).
A6h	RESYNCH successfully performed by GemCore.
A7h	Protocol Type Selection (PTS) error.
B0h	GCR410 command not supported.
CFh	Other key 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 remains to be sent or received).
E7h	Error returned by the card. The SW1 and SW2 bytes returned by the card are different than 90h 00.
F7h	Card removed. The card has been withdrawn during the execution 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 missing. 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> Main Card

1Eh CLA INS A1 A2 Lin <DATA IN> Lout Lcode <CODE> Selected Auxiliary Card

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 1Eh 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

```
1h 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 is 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 (for example, INC A = 04h).

***Note**: Instructions in italics are macro-commands. See the "Macro-Commands" section in "Appendix B" 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 exists in 8051 Assembler code 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 exists in 8051 Assembler code 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.

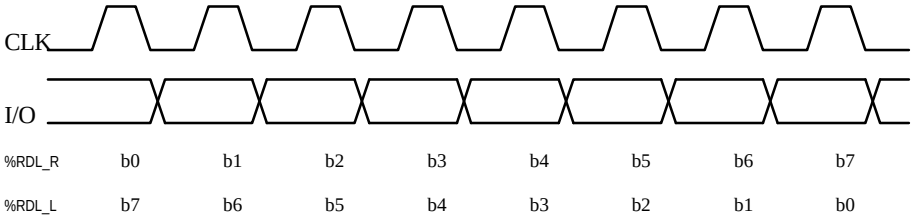
%SEND_D When the 3.68 MHz asynchronous clock is activated on CLK, this command writes the contents of A on the I/O in asynchronous mode using the direct convention.
The configuration is 9,600 baud, 8 bits, even parity, 1 stop bit, 1s time-out.

%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:



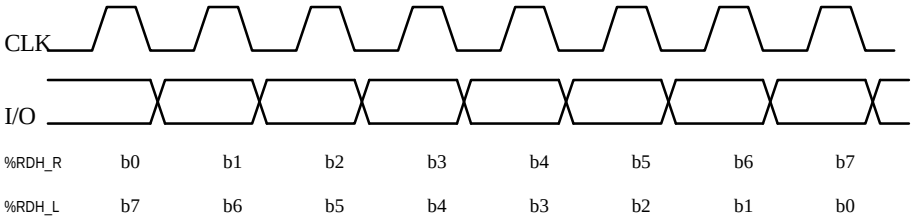
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs.

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:



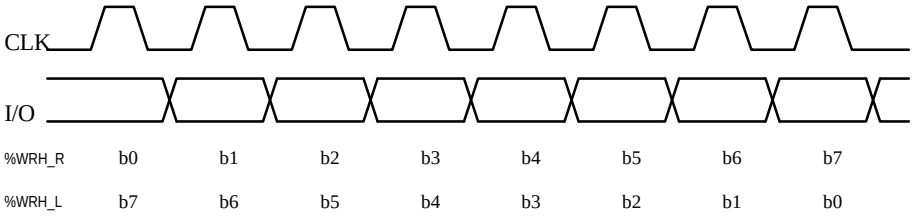
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs.

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:



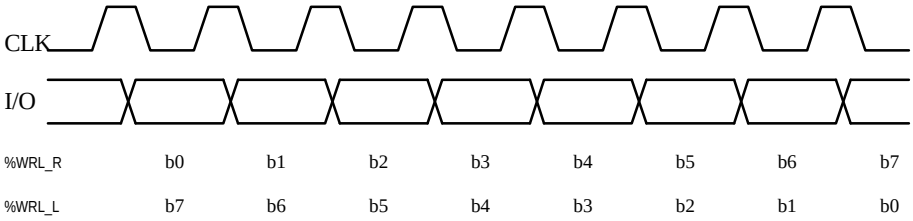
- CLK contact set to 0 for 10 µs.
- CLK contact set to 1 for 10 µs

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:



- 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.

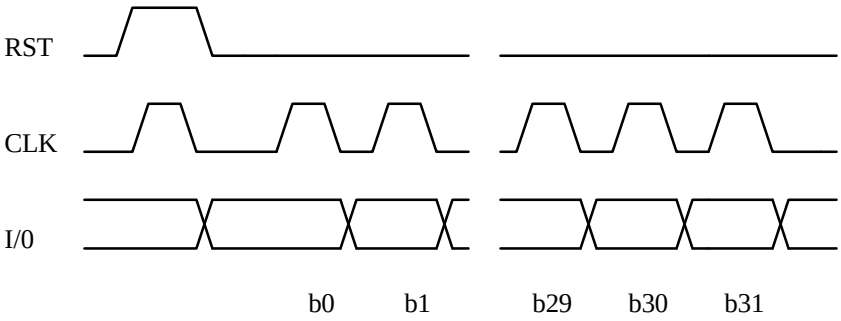
%CLK_PUL This command generates a logical pulse 1 for 10 µs on the CLK line and then resets the line level to 0.

%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`

INDEX

TERMINOLOGY

Abbreviations

ACK	Acknowledgement byte
ADH	Used in the **Read Memory** and **Write** Memory commands, ADH is the most significant byte of the 16-bit address of the first byte to be read or written.
ADL	Used in the **Read Memory** and **Write** Memory commands, ADL is the least significant byte of the 16-bit address of the first byte to be read or written.
APDU	Application Protocol Data Unit
BWI	Block Waiting time Integer
CRC	Cyclic Redundancy Check
CWI	Character Waiting time Integer
DAT	DATa (being transmitted)
EDC	Error Detection Code
EOT	End Of Transmission
etu	elementary time unit
GBP	Gemplus Block Protocol
GBR	GemCore-Based Reader
I-Block	Information Block
IFSC	Information Field Size of the Card
IFSD	
ISO	International Standards Organization
LCD	Liquid Crystal Display
LEN	
NAD	
OROS	
PCB	
PTS	Protocol Type Selection
R-Block	Receive Ready Block
ROS	Reader Operating System
S-Block	Supervisory Block
TLP	
TTL	
UART	

Glossary

Application Device Handler	
APDU	Application Protocol Data Unit; data exchange protocol between a card and a reader. The APDU can be changed to ensure that it meets reader requirements for the user's site. For example, in the GCR400 card reader, the APDUMAXIN is 248 and the APDUMAXEXP is 251.
Baud	Rate of signals per second transmitted over a communication channel.
Block	Logically contiguous data memory that is allocated when requested for data field
Command Layer	The command layer handles and interprets the GemCore V1.1-Based Reader commands.
GemCore	
OROS	
Physical Layer	The physical layer handles the data transmission.
ROS	
System Device Handler	
T = 0 Protocol	Character-oriented asynchronous half duplex transmission protocol
T = 1 Protocol	Block-oriented asynchronous half duplex transmission protocol
Transport Layer	The transport layer handles message addressing, specifies the transmission type, and validates each transmission. The transport layer can use one of two protocols: the TLP224 protocol or the Gemplus Block Protocol.