

# **GBP Interface Library Kit**

---

## **Programmer's Guide**



**GEMPLUS**

All information herein is either public information or is the property of and owned solely by Gemplus S.A. 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' 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.**

**Gemplus does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemplus be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemplus products. Gemplus disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.**

© Copyright 2002 Gemplus S.A. All rights reserved. Gemplus, the Gemplus logo, and GemCore are trademarks and service marks of Gemplus S.A. and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners. Certain smart cards produced by Gemplus are covered by Bull CP8 Patents.

GEMPLUS, B.P. 100, 13881 GEMENOS CEDEX, FRANCE.

Tel: +33 (0)4.42.36.50.00 Fax: +33 (0)4.42.36.50.90

Printed in France.

Document Reference: DOC 107821A0

Document Version: 1.0

March 14, 2002

# Contents

---

<b>Introduction</b>		<b>vii</b>
Who Should Read this Book		vii
Reference Documents		viii
Contact our Hotline		viii
Corporate and EMEA		viii
Americas		viii
From our Web Site		viii
Conventions		ix
GemCore Interface Library License Agreement		x
<b>Chapter 1</b>	<b>Installation under the Linux Operating System</b>	<b>1</b>
Description		1
Example		2
tar -xvzf Gem_TL.tar.gz		2
Building the GBP Library		4
Building a Sample		4
<b>Chapter 2</b>	<b>Validation</b>	<b>5</b>
<b>Chapter 3</b>	<b>The Gemplus Block Protocol</b>	<b>7</b>
Example		7
<b>Chapter 4</b>	<b>The GBP Interface Library</b>	<b>9</b>
Library Architecture		9
Status Codes		11
GBP Library Management		12
Representation		12
Normal Execution		13
LRC or Parity Error Detected		13
Sequence Number Error Detected		15
GBP Automatic Resynchronization		16
Reader is Mute		17
Abnormal Behavior		17

<b>Chapter 5</b>	<b>The GBP Interface Library API</b>	<b>19</b>
	Transport Layer Commands . . . . .	19
	Get_TL_Version . . . . .	19
	TL_Open . . . . .	20
	TL_Send_Receive . . . . .	20
	TL_GetTimeOut . . . . .	21
	TL_SetTimeOut . . . . .	21
	TL_Close . . . . .	22
	TL_Reset . . . . .	22
	GBP Specific Commands . . . . .	23
	TL_GBP_GetNAD . . . . .	23
	TL_GBP_SetNAD . . . . .	23
	TL_GBP_GetSeqNumber . . . . .	23
	TL_GBP_SetSeqNumber . . . . .	24
	Use Case Sample . . . . .	25
<b>Chapter 6</b>	<b>Porting Instructions</b>	<b>27</b>
	Programming Rules . . . . .	27
	Variable Naming: The Hungarian Notation . . . . .	27
	Authorized Data Types . . . . .	29
	File Header . . . . .	29
	Function Header . . . . .	30
	Block Reception Algorithm . . . . .	31
	GBP Interface Library files Description . . . . .	32
	TL.h and TL.c . . . . .	32
	VP.h and VP.c . . . . .	32
	PP.h and PP.c . . . . .	32
	Error.h . . . . .	32
	GBP Interface Library Porting Instruction . . . . .	33
	Steps to Port the Library . . . . .	33
	Private Commands . . . . .	34
	VP_Open . . . . .	34
	VP_Close . . . . .	34
	VP_Send . . . . .	35
	VP_Receive . . . . .	35
	VP_SetTimeOut . . . . .	35
	VP_GetTimeOut . . . . .	36
	Example . . . . .	36
<b>Chapter 7</b>	<b>Application Samples</b>	<b>39</b>
	GemCore 1.10 . . . . .	40
	GPM2K Synchronous Card . . . . .	40
	T=0 Smart Card Management . . . . .	41
	T=1 Smart Card Management . . . . .	41
	GemCore Lite 1.21-GK . . . . .	42
	GPM2K Synchronous Card . . . . .	42
	T=0 Smart Card Management . . . . .	43
	T=1 Smart Card Management . . . . .	43
	EMV Smart Card Management in EMV Mode . . . . .	44
	Non-EMV Smart Card Management when EMV Mode Fails . . . . .	44

GemCore-1.21-GM .....	45
GPM2K Synchronous Card .....	45
T=0 Smart Card Management .....	46
T=1 Smart Card Management .....	46
GemCore 1.22 .....	47
GPM2K Synchronous Card .....	47
EMV Smart Card Management in EMV Mode .....	48
GemCore Lite 1.32-GK .....	49
GPM2K Synchronous Card .....	49
T=0 Smart Card Management .....	50
T=1 Smart Card Management .....	51
EMV Smart Card Management in EMV Mode .....	51
Non-EMV Smart Card Management when the EMV Mode Fails .....	52

**Terminology**

**53**

Abbreviations .....	53
Glossary .....	54



# Introduction

---

This guide describes how to use the Gemplus Block Protocol (GBP) Interface Library Kit Version 1.0 to develop an application driving a Gemplus device based on the GBP protocol (Reader, GemCore chipset, ...).

The GBP Interface Library Kit optimizes independence from the environment (OS and physical port). It can thus be ported easily to any operating environment.

A sample implementation on Linux is provided using the Red Hat 7.1 distribution and an Intel i686-based computer. This example illustrates communication with a physical RS-232 interface in a Linux test environment.

The library is compatible with the following GemCore versions:

- GemCore standard 1.10, 1.21-GM and 1.22
- GemCore Lite 1.21-GK and 1.32-GK

The C source code, the library and the samples are commented and comply with programming rules described in the chapter “*Porting Instructions*”.

## Who Should Read this Book

This document has been designed for developers who need to interface a GemCore device based on the Gemplus Block Protocol. No particular familiarity with the GBP is required.

This guide assumes that the user has programming experience in the C language. It also assumes familiarity with serial port programming.

## Reference Documents

- *Identification cards - Integrated Circuit(s) Cards with Contacts - Part 3: Electronic Signals and Transmission Protocols* ISO/IEC 7816-3 1989
- *Identification Cards - Integrated Circuit(s) Cards with Contacts - Part 4: Inter-Industry Commands for Interchange* ISO/IEC 7816-4 1995
- *EMV '96 Version 3.1.1 Specifications*
- *EMV 2000 Specifications*
- *The C Programming Language* ISO/IEC 9899 1999
- *GemCore Technical Specifications (V1.10/ V1.22 and V1.21-GM)*
- *GemCore Lite Technical Specifications (V1.21-GK)*
- *GemCore Lite V1.3x Technical Specifications (V1.32-GK)*
- *GemCore 1.10-Based Reader Reference Manual (V 1.10)*
- *GemCore 1.21-Based Reader Reference Manual (V1.21-GM)*
- *GemCore EMV-Based Reader Reference Manual (V1.22)*
- *GemCore Lite-Based Reader Reference Manual (V1.21-GK)*
- *GemCore Lite V1.3x-Based Reader Reference Manual (V 1.32-GK)*
- *GemCore Chipset Host Interface Programmer's Guide (all versions)*

## Contact our Hotline

If you do not find the information you need in this manual, or if you find errors, contact the Gemplus hotline by phone, fax, or e-mail. In your e-mail, please include the document reference number, your job function, and the name of your company. (You will find the document reference number at the bottom of the legal notice on the inside front cover).

### Corporate and EMEA

Hotline: +33 (0)4 42 36 50 50

Hot Fax: +33 (0)4 42 36 50 98

Email: [hotline@gemplus.com](mailto:hotline@gemplus.com)

### Americas

Hotline: 1 (877) 436-7233

Hot Fax: 1 (215) 390-1586

Email: [hotlineusa@gemplus.com](mailto:hotlineusa@gemplus.com)

### From our Web Site

<http://www.gemplus.com>

## Conventions

The following conventions are used in this document:

### Numeric values

By default, numeric values are expressed in decimal notation.

- Binary numbers are followed by the ‘b’ character. For example, the decimal value 13 is expressed in binary as **1101b**.
- Hexadecimal numbers are followed by the ‘h’ character. For example, the decimal value 13 is expressed in hexadecimal as **0Dh**.

### RFU values

The value 00h is assigned to each RFU (Reserved for Future Use) byte.

### Bit Numbering

A byte consists of 8 bits,  $b_8$  to  $b_1$ , where  $b_8$  is the most significant bit and  $b_1$  the least significant bit, as shown below:

One byte

$b_8$	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$
-------	-------	-------	-------	-------	-------	-------	-------

### Byte Numbering

A string of  $n$  bytes consists of  $n$  number of concatenated bytes:  $B_n B_{n-1} \dots B_2 B_1$ .  
 $B_n$  is the most significant byte and  $B_1$  is the least significant byte:

String on  $n$  bytes

$B_n$	$B_{n-1}$	...	$B_4$	$B_3$	$B_2$	$B_1$
-------	-----------	-----	-------	-------	-------	-------

## GemCore Interface Library License Agreement

Notice to user: THIS AGREEMENT STATES THE TERMS AND CONDITIONS UPON WHICH GEMPLUS OFFERS TO YOU THE GEMCORE INTERFACE LIBRARY (the "SOFTWARE") INCLUDED IN THIS KIT. BY OPENING THIS PACKAGE YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS AGREEMENT, PROMPTLY RETURN THE PACKAGE UNOPENED TO GEMPLUS, AVENUE DU PIC DE BERTAGNE, PARC D'ACTIVITÉ DE GÉMENOS, 13420 GEMENOS, FRANCE.

1. LICENSE GRANTED. GEMPLUS grants to you a royalty-free, non-exclusive and non-transferable right to use the SOFTWARE, including corrective patch if any, on a single central processing unit, for the sole purpose of providing an equipment integrating a card interface to end customers. The license herein granted conveys the right to modify the SOFTWARE in order to port the SOFTWARE on any type of central processing unit or to modify its functionality, and to make copies of the SOFTWARE into any machine-readable form.

The license herein granted conveys no right to grant sub-licenses, is not to be deemed transferable for any purpose and is indivisible and non-assignable. You will not lease or assign even free of charge in whole or in part the SOFTWARE.

The license to use, modify and copy the SOFTWARE is conditioned upon your compliance with the terms of this Agreement.

2. COPYRIGHT. The SOFTWARE is protected by copyright law and international treaty provisions. The SOFTWARE is licensed not sold to you. Title and full ownership rights to the SOFTWARE, its accompanying written materials, on-line and electronic documentation, and all copies of the SOFTWARE remain the exclusive property of GEMPLUS and/or its suppliers; the right to use the SOFTWARE is exclusive of any other right and particularly shall not imply any transfer of ownership. Any reproduction or copying of the accompanying written materials of the SOFTWARE shall be the exclusive property of GEMPLUS.

3. DISCLAIMER OF WARRANTY - GEMPLUS PROVIDES THE SOFTWARE « AS IS », FOR EXPERIMENTAL USE ONLY. GEMPLUS MAKES NO WARRANTIES THAT THE SOFTWARE WILL PERFORM WITHOUT ERRORS OR WILL BE FREE OF DEFECTS. GEMPLUS MAKES NO WARRANTIES AS TO THE QUALITY, SUITABILITY TO A SPECIAL PURPOSE, ACCURACY, AND THAT THE SOFTWARE WILL CAUSE NO DAMAGES ARISING OUT OF THE USE OF, PERFORMANCE, OR INABILITY TO USE THIS GEMPLUS PRODUCT, TO ANY HARDWARE, SOFTWARE, NETWORK, SERVER OR ANY OTHER MATERIAL OF THE CLIENT OR ANY THIRD PARTY, EVEN IF GEMPLUS HAD BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

In particular, as the software MAY be modified and SHALL BE validated by yourself, you shall bear FULL AND SOLE LIABILITY FOR ANY KIND OF LOSSES and damages resulting from the use of the software BY YOU OR ANY END CUSTOMER.

4. NO OTHER WARRANTIES - GEMPLUS disclaims all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability, QUALITY and fitness for a particular purpose AND NON INFRINGEMENT OF THIRD PARTY RIGHTS with respect to the SOFTWARE, ITS ACCOMPANYING HARDWARE AND WRITTEN MATERIALS, OR ANY APPLICATION SOFTWARE DEVELOPED BY YOU USING THE SOFTWARE.

5. NO LIABILITY FOR DAMAGES - In no event shall GEMPLUS OR ITS SUPPLIERS be liable for any damages whatsoever either DIRECT OR INDIRECT, including CONSEQUENTIAL, SPECIAL, INCIDENTAL OR indirect damages OF ANY KIND (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use OF, PERFORMANCE or inability to use THIS GEMPLUS PRODUCT, OR ANY APPLICATION YOU DEVELOPED THEREWITH even if GEMPLUS has been advised of the possibility of such damages. In any case, GEMPLUS' entire liability FOR ANY CLAIM, WHETHER IN CONTRACT, TORT OR ANY OTHER THEORY OF LIABILITY shall be limited to the amount actually paid by you to GEMPLUS for the SOFTWARE LICENSE.

6. SEVERABILITY - In the event of invalidity of any provisions of this license, such invalidity shall not affect the validity of the remaining portions of this license.

7. TERMINATION - This License is effective until terminated. You may terminate it any time by destroying the SOFTWARE together with all copies of the SOFTWARE. Also, GEMPLUS has the option to terminate immediately the present License if you fail to comply with any term or condition of this License. You agree upon such termination to destroy the SOFTWARE together with all copies of the SOFTWARE.

8. CONFIDENTIALITY - You undertake not to disclose or transfer to any third party in any way whatsoever in all or in part the documents data or information of whatever nature transmitted or made available to You by GEMPLUS, related but not limited to the SOFTWARE.

9. GOVERNING LAW - This License is governed by the laws of FRANCE to the exclusion of its conflict of laws rules and is subject to the exclusive jurisdiction of the Courts of Paris.



# Installation under the Linux Operating System

---

This chapter refers specifically to installation under the Linux operating system. For other operating systems, refer to the chapter “*Porting Instructions*”.

## Description

The product is provided in the “.gz” file format. The archive name is **Gem\_TL.tar.gz**.

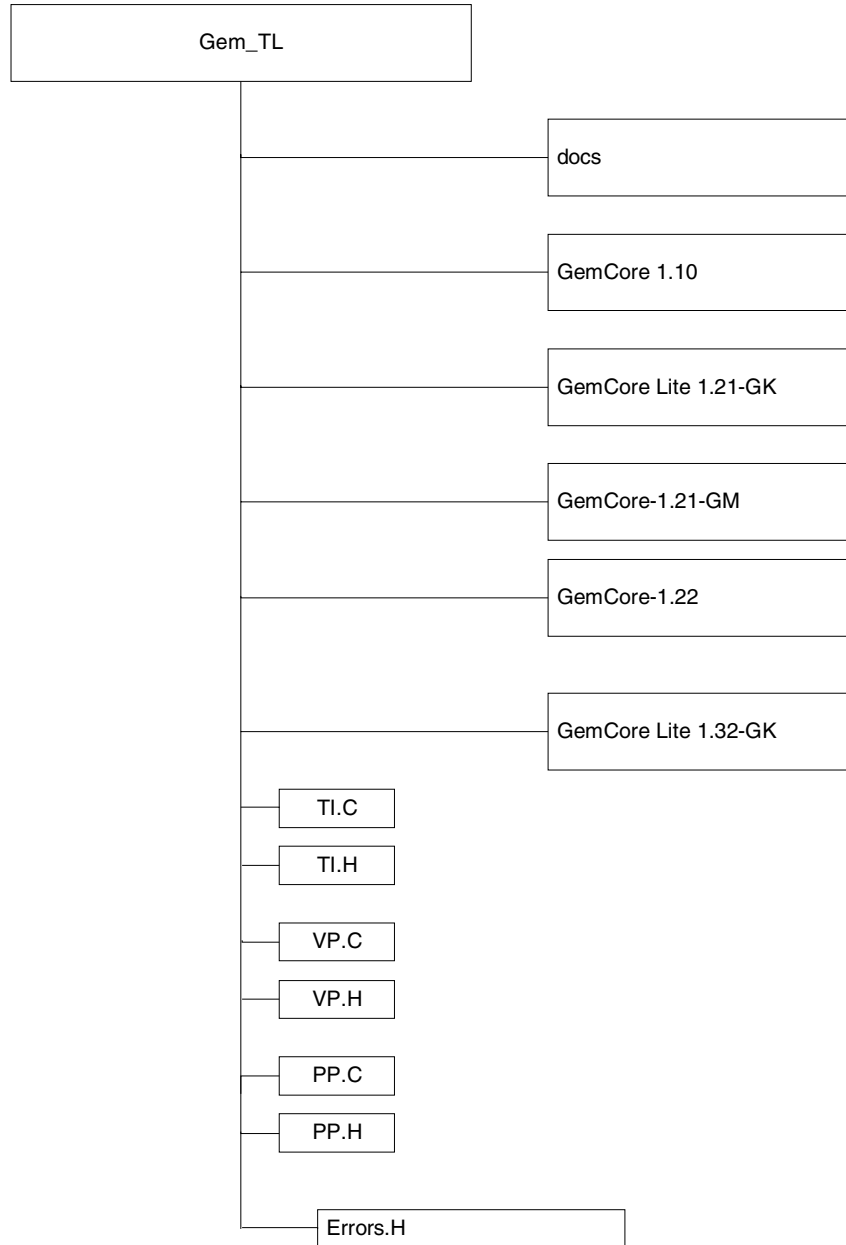
This file must be copied to a working directory. Once it has been copied, the archive can be extracted.

To extract files, use the “tar” program provided by Linux.

## Example

**tar -xvzf Gem\_TL.tar.gz**

Once extracted, the following directory tree is displayed:



**Figure 1 - Extracted File Directory**



## Building the GBP Library

In the directory the archive has been extracted to, run the make file.

---

	<b>Run</b>
To build the library	make gbp
To clean the library	make clean

---

**Table 2 - Building the GBP Library**

---

**Note:** For the sake of simplification, once the library has been built, it is copied to the following directory: /usr/lib.

---

## Building a Sample

Go to the directory the sample will be built in, for example, GemCore-1.21/SCardT0.

---

	<b>Run</b>
To build the sample	make

---

**Table 3 - Building a Sample**

Once the sample has been built, it can be launched using a terminal console by typing:

./Sample

## Validation

---

All files (GBP library, samples) have been validated with readers based on the following GemCore versions:

- GemCore-1.10
- GemCore-1.21-GM
- GemCore Lite 1.21-GK
- GemCore-1.22
- GemCore Lite 1.32-GK.

The tests were performed using a Linux operating system configured as follows:

- Red Hat Linux release 7.1 (Seawolf)
- Kernel 2.4. 2-2 on an i686
- Compiler gcc-2.96.

The following computers were used for the tests:

- Siemens Celsius 400
- Compaq Deskpro EN Series 6333/3.2 FRE
- Toshiba Tecra 8100.







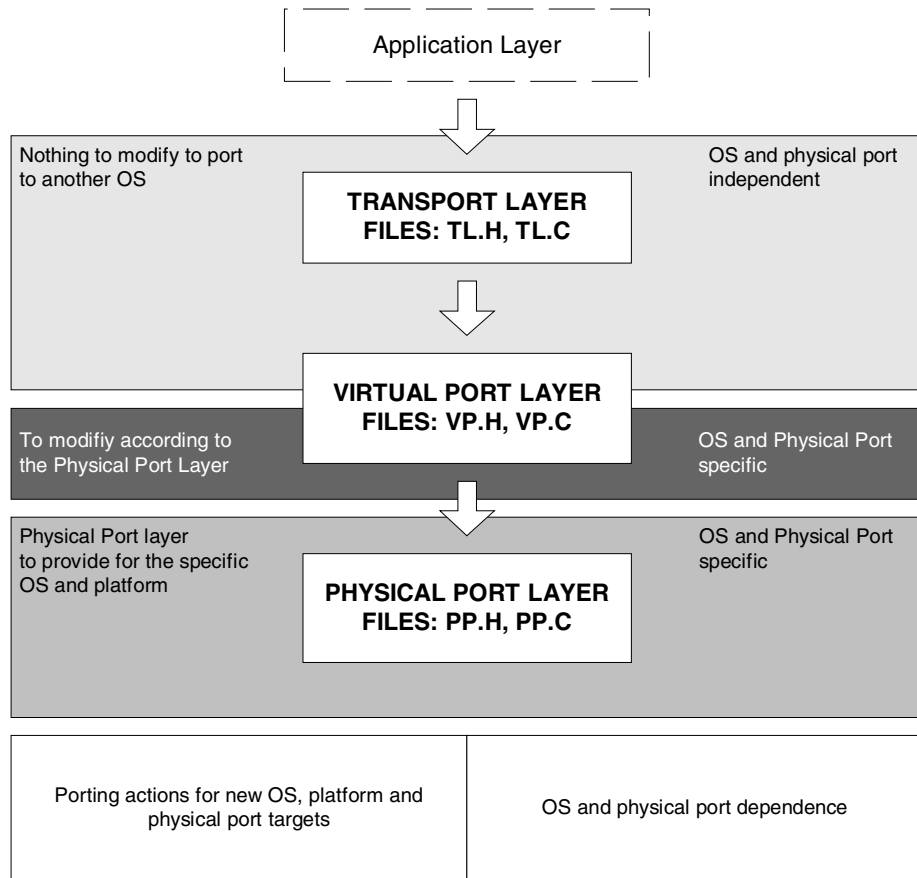
# The GBP Interface Library

---

This chapter describes the various layers of the GBP Interface Library and the corresponding source files.

## Library Architecture

Due to the GBP Interface Library structure, the application can use the same interface regardless of the physical layer. This structure also allows for the easy porting of the library onto another operating system and another physical port (for example, I2C). End developers' main task consists of interfacing the virtual port layer with their own specific physical port layer. The GBP-specific transport layer remains unchanged (and thus protected).



**Figure 1 - The GBP Interface Library**

**Note:** See the chapter “*Porting Instructions*” for additional information about API layers.

The GBP interface library is not designed for real-time operating systems and multi-reader management, but its design should help developers to add this feature easily.

## Status Codes

All error codes are described in the file **Errors.H**. Each error code corresponds to the base value "ERR\_BASE" + another value X. This value is set to 1000h and it can be changed to match other customer constraints.

These codes are:

NO_ERR	0		/*Everything is OK.*/
ERR_LRC		ERR_BASE+2	/*Wrong LRC.*/
ERR_READER_MUTE		ERR_BASE+3	/*Reader is mute.*/
ERR_SEQUENCE_NUMBER		ERR_BASE+4	/*Incorrect sequence number received.*/
ERR_RESYNCH		ERR_BASE+5	/*The API had to send a <b>Resynch</b> command to the reader.*/
ERR_SERIOUS_ERROR		ERR_BASE+8	/*A serious error occurred. <b>Resynch</b> command failed.*/
ERR_NACK		ERR_BASE+9	/*The reader has sent NACK value (only applicable with the TLP protocol).*/

The following codes are only used for physical port management:

ERR_COMNUM		ERR_BASE	/*Wrong Com Port number or the requested serial port does not exist.*/
ERR_INITPORT		ERR_BASE+1	/*One of the arguments (BaudRate, start bit,...) is wrong.*/
ERR_WRITEPORT		ERR_BASE+6	/*An error occurred during the TL_SendReceive command.*/
ERR_READPORT		ERR_BASE+7	/*An error occurred during the TL_SendReceive command.*/

# GBP Library Management

## Representation

### Application/Library communication

- Cmd, [l] ----->  
The application sends the **Buffer** command containing “l” data commands.
- <-----status, [l]

Where:

[l]= buffer holding “l” data which transport the command.

### Library / GBP device communication layer

Block format: [NAD][PCB][LEN][DATA][EDC]

For further information about the GBP Protocol, refer to the *GemCore Reference Manual*.

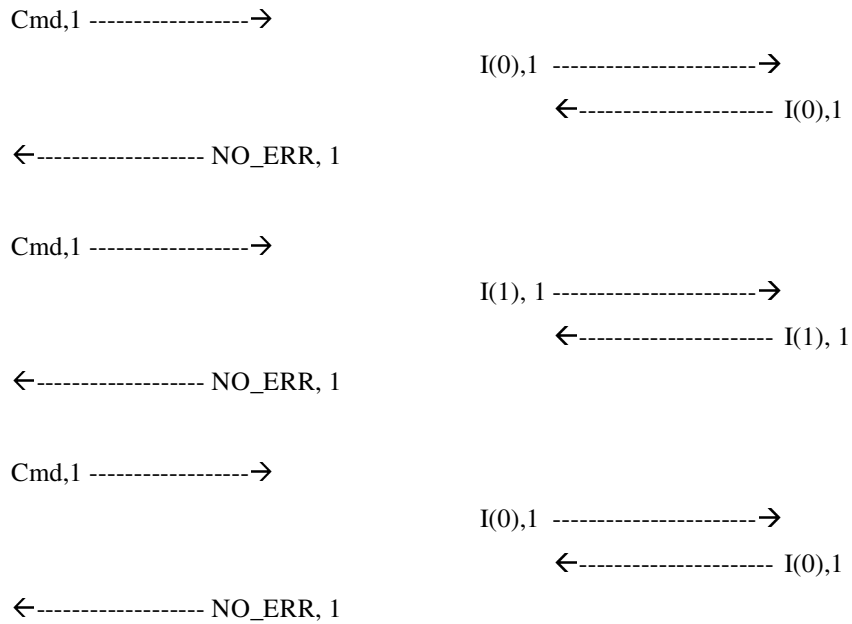
- I[(x)],[l] ----->  
Host sends I-Block with sequence number “x”, transporting a buffer of “l” DATA
- <----- I[(x)],[l]  
Device sends I-Block with sequence number “x”, transporting a buffer of “l” DATA
- <----- R(81)  
Device sends R-Block for EDC and/or parity error on received I(0) from host
- <----- R(91)  
Device sends R-Block for EDC and/or parity error on received I(1) from host
- <----- R(82)  
Device sends R-Block for other error on received I(0) from host
- <----- R(92)  
Device sends R-Block for other error on received I(1) from host
- <-----X-----  
Time-out upon reception.
- <-----E/P-----  
EDC or parity error in the response
- S(Rreq)----->  
Host sends an S-Block to request resynchronization with the GBP device
- <----- S(Rrep)  
Device sends a S-Block to reply to a resynchronization with the host
- <----- Not S(Rrep)  
Device sends an X-Block (which is not a response to the resynchronization with the host)

---

**Note:** Sequence number “x” is omitted when “x” value has no impact on the test behavior. It can hold any value in the range [0, 1].  
DATA length “l” is omitted when value “l” has no impact on the test behavior. It can hold any value in the range [1, FF].

---

## Normal Execution

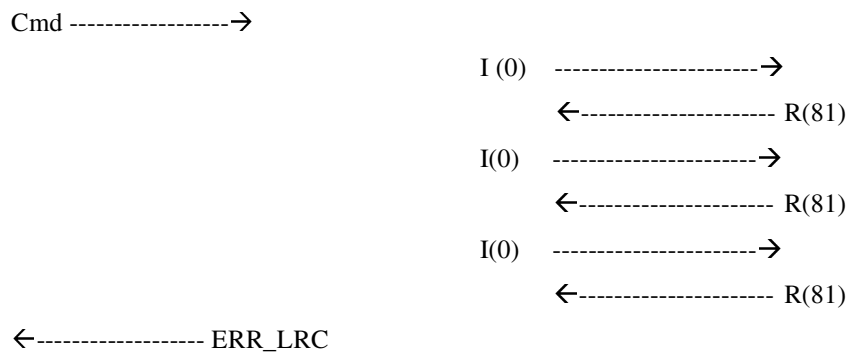


The above behavior is normal.

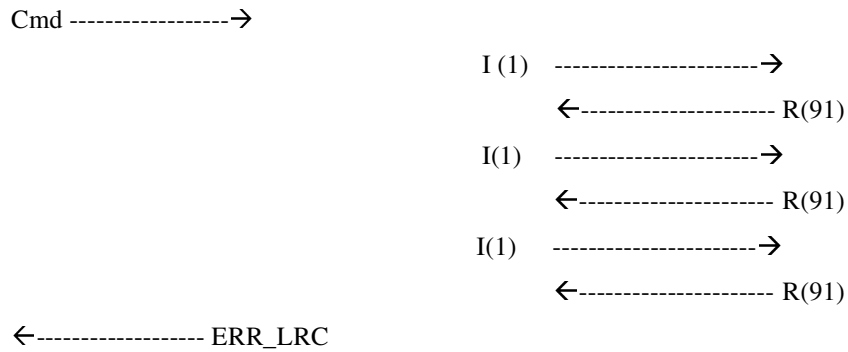
## LRC or Parity Error Detected

- LRC or Parity Error Detected by the Reader

### Case 1

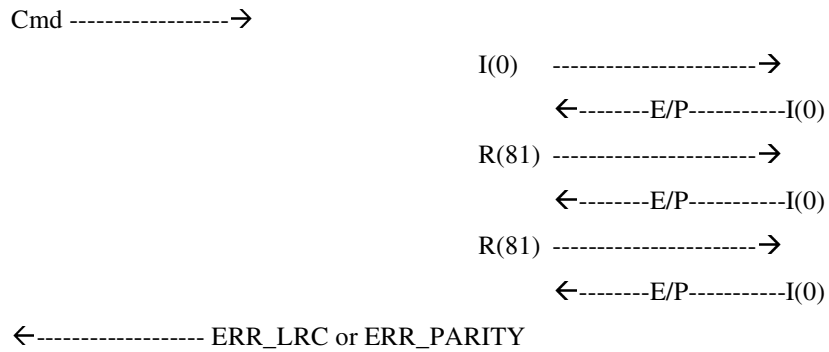


**Case 2**

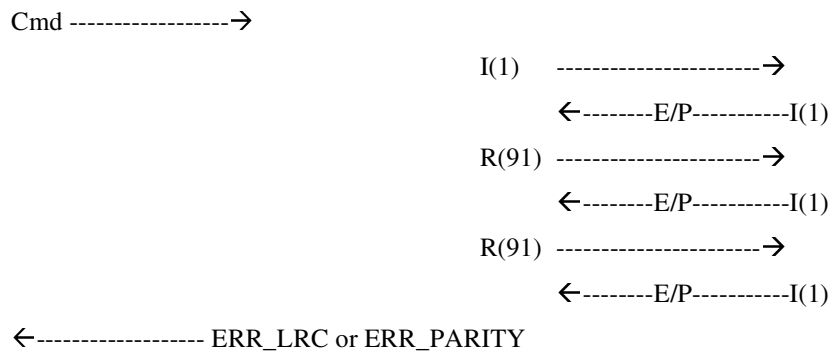


- **LRC or Parity Error Detected by the Host**

**Case 1**



**Case 2**




---

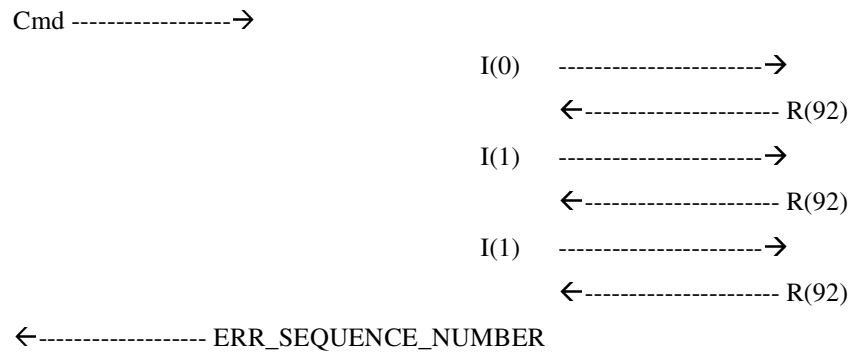
**Note:** In both cases, the same error code is returned to the application.

---

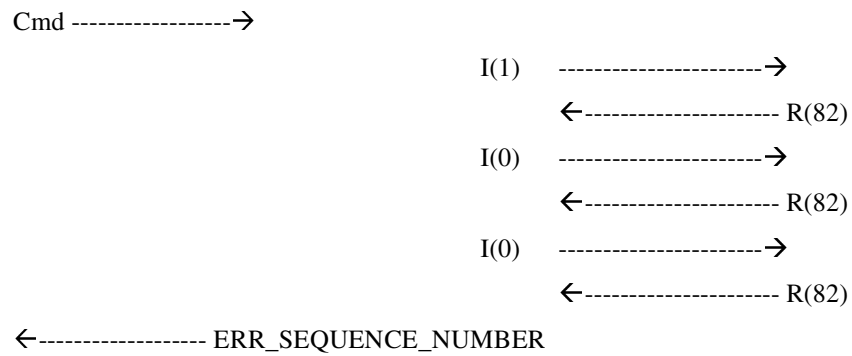
When this type of error occurs, the application may send the reader a resynchronization instruction using the **TL\_Reset** API command described below.

## Sequence Number Error Detected

### Case 1



### Case 2

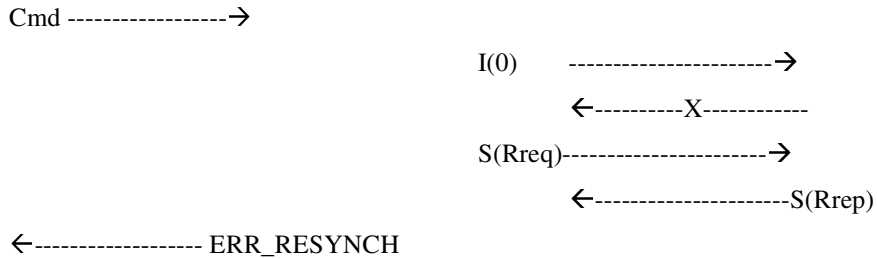


When this type of error occurs, the application may send the reader a resynchronization instruction using the **TL\_Reset** API command described below.

## GBP Automatic Resynchronization

Resynchronization may be due to the one of the following causes:

### Time-out

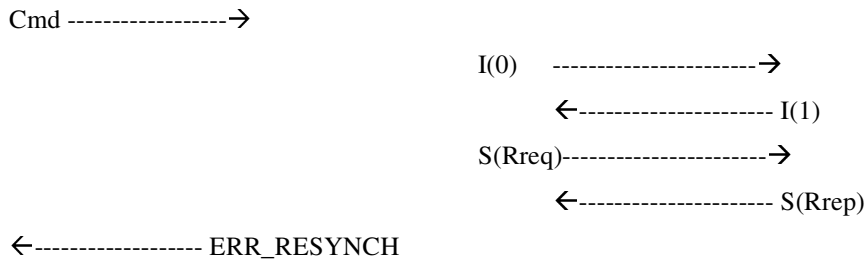


In this case, the time-out value may be too short. `TL_GetTimeOut()` API is used to get the current time-out value and to change it, if necessary.

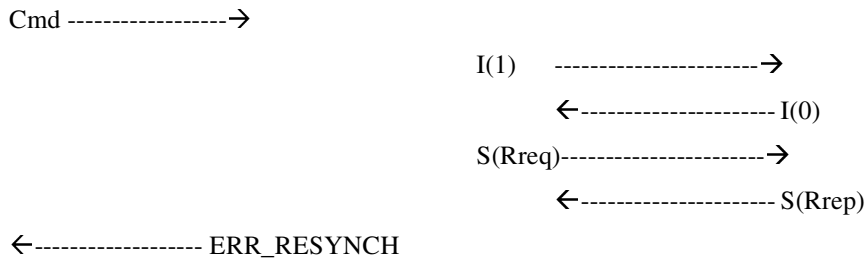
The application can continue to run. Transport layer parameters have been reset in order to proceed with the next exchange normally. The application must resend the same command to the reader.

### Error in Sequence Number (Detected by the Host)

#### Case 1



#### Case 2



In this case, the application can continue running. Transport layer parameters have been reset in order to proceed with the next exchange normally.

The error may occur because the host and the reader are not synchronized. The application must resend the same command to the reader to complete the exchange.

## Reader is Mute

Cmd ----->

I(0) ----->

←-----X-----

S(Rreq)----->

←-----X-----

←----- ERR\_READER\_MUTE

The library failed to receive the transmission from the reader. The application must handle this error.

## Abnormal Behavior

Cmd ----->

I(0) ----->

←-----X-----

S(Rreq)----->

←----- Not S(Rrep)

←----- ERR\_SERIOUS\_ERROR



# The GBP Interface Library API

---

All the following commands can be found in the **TL.C** and **TL.H** files.

## Transport Layer Commands

### Get\_TL\_Version

This command returns a string version with the following format "TL\_VERSION-Vx.y". The x and y values are changed in accordance with modifications to the library.

#### Format

```
char * Get_TL_Version();
```

#### Input

None.

#### Response

Version in a string buffer using the following format: "TL\_Version-Vx.y". Values for x and y are modified according to changes made to the library.

#### Example

```
Printf("The current version library is %s",Get_Tl_Version());
```

## TL\_Open

This command opens a GBP transport session. It opens the virtual port with the parameters of the incoming argument.

### Format

```
short int TL_Open(unsigned char ucComNumber,
                 unsigned long uliBaudRate);
```

### Input

ucComNumber	The application uses this parameter to specify which virtual port to open. Normal values range from 1 to 4 and are typically assigned to serial ports 1 to 4 for the physical serial ports.
UliBaudRate	Specifies the communication speed for the session. Typically, values for a physical communication port are 9600, 19200 or 38400 bps.

### Response

NO\_ERR  
 ERR\_COMNUM  
 ERR\_INITPORT

### Example

```
/*to open serial port 2 at a Baud rate of 9600 bps.*/
Err= TL_Open(2,9600);
```

---

**Note:** For all GemCore chipsets, the speed after power on is 9600 bps.

---

## TL\_Send\_Receive

This command sends an instruction to the device and returns the device response.

### Format

```
short int TL_SendReceive(unsigned char ucCommandLen,
                        unsigned char *prgucCommand,
                        unsigned char *pucResponseLen,
                        unsigned char *prgucResponse)
```

### Input

UCCommandLen	Length of the command to be sent to the device.
PrgUCCommand	Buffer command to be sent.

### Output

UCResponseLen	Length of the device response.
PrgUCResponse	Response buffer.

---

**Note:** The correct buffer size must be allocated. A buffer size of 256 bytes is recommended.

---

**Response**

NO\_ERR  
ERR\_LRC  
ERR\_SEQUENCE\_NUMBER  
ERR\_NACK  
ERR\_READER\_MUTE  
ERR\_RESYNCH  
ERR\_SERIOUS\_ERROR

**Example**

```
/*Send GemCore Power Down command to the GemCore device*/  
/*The GemCore device responds 00h*/  
Err= TL_SendReceive(1, "\x11", &ResLen, Resp);  
Where:  
x11 corresponds to the GemCore Power Down command.
```

## TL\_GetTimeOut

Returns the current time-out value. The default value is 2000 ms.

**Format**

```
unsigned long TL_GetTimeOut()
```

**Input**

None.

**Response**

Time-out value in ms.

**Example**

```
Printf("Current Time-out is %d", Get_TL_TimeOut());
```

## TL\_SetTimeOut

This command changes the current time-out value.

**Format**

```
void TL_SetTimeOut(unsigned long uliNewTimeOut)
```

**Input**

UliNewTimeOut          New time-out in ms.

**Response**

None.

**Example**

```
/*Initialize 5s time-out.*/  
TL_SetTimeOut(5000);
```

## TL\_Close

This command closes the serial port.

### Format

```
short int TL_Close()
```

### Input

None.

### Response

No\_Err

### Example

```
/*To close the serial port previously opened by a TL_Open command.*/  
Err= TL_Close();
```

## TL\_Reset

This command resets communication with the reader.

---

**Note:** This command resets the sequence number coded in the PCB byte to its default value (0).

---

### Format

```
short int TL_Reset()
```

### Input

None.

### Response

NO\_ERR

ERR\_LRC

ERR\_SEQUENCE\_NUMBER

ERR\_NACK

ERR\_READER\_MUTE

ERR\_SERIOUS\_ERROR

### Example

```
/*Reset Communication */  
Err= TL_Reset();
```

---

## GBP Specific Commands

### TL\_GBP\_GetNAD

This command returns the current NAD. The NAD value is used in the GBP protocol. The default value is 42h. See the chapter “*The Gemplus Block Protocol*” for details.

**Format**

```
unsigned char TL_GBP_GetNAD()
```

**Input**

None.

**Response**

NAD value

**Example**

```
/*To get the current NAD used with the GBP.*/  
printf("Current NAD Value is %X",TL_GBP_GetNAD());
```

### TL\_GBP\_SetNAD

This command changes the NAD currently in use.

**Format**

```
void TL_GBP_SetNAD(unsigned char ucNewNAD)
```

**Input**

UCNewNad                      New NAD value for the next exchange.

**Response**

None.

**Example**

```
/*To set a new NAD value.*/  
TL_GBP_SetNAD(0x43);
```

## TL\_GBP\_GetSeqNumber

This command returns the current sequence number (PCB). The PCB value is used with the GBP. See “*The Gemplus Block Protocol*” later in this document for additional details.

### Format

```
unsigned char TL_GBP_GetSeqNumber()
```

### Input

None

### Response

PCB value. This value is 40h or 00h.

### Example

```
/*Get PCB currently used with the GBP.*/
printf("Current Sequence Number Value is %X",TL_GBP_GetSeqNumber());
```

## TL\_GBP\_SetSeqNumber

This command changes the current sequence number (PCB).

### Format

```
void TL_GBP_SetSeqNumber(unsigned char ucNewPCB)
```

### INPUT

UCNewPCB                      Is the new PCB value for the next exchange. This value is 00h or 40h.

### RESPONSE

None.

### EXAMPLE

```
/*To set a new PCB value.*/
TL_GBP_SetNAD(0x00);
```

## Use Case Sample

The following C sample shows how to:

- Open a virtual port
- Get and modify the NAD value
- Get and modify the time-out value
- Exchange a command with a reader.

```

void main()
{
    unsigned char rgucResp[256],
    ucLgRep,
    ucLgCmd,
    rgucCmd[256],
    ucIndice=0;
    unsigned long uliMydwTimeOut;
    unsigned short int usiErr;

    /*Open Serial Port 1 at 9600 bds*/
    printf("Open Com at 9600 Bds\n");

    if(TL_Open(1,9600)!= No_Err)
    {
        printf("Fail on Open Com\n");
        exit(0);
    }

    /*Get NAD in use*/
    printf("Current NAD value: %X\n", TL_GBP_GetNAD());
    /*This value must be changed as follows*/
    TL_GBP_SetNAD(0X43);
    Printf("New NAD value: %X\n",TL_GBP_GetNAD());

    /*Set Time Out in ms*/
    uliMydwTimeOut= TL_GetTimeOut();
    printf("Default TimeOut is %d\n",uliMydwTimeOut);
    /*We set a new Time Out value in ms*/
    uliMydwTimeOut= 5000 ;/*5s*/
    TL_SetTimeOut (uliMydwTimeOut);

    /*===== Build First GemCore Command*/
    ucIndice= 0 ;
    rgucCmd[ucIndice++]= 0x12 ; /*Power Up*/
    rgucCmd[ucIndice++]= 0x13 ;/*Without PTS and with Class Selection*/
    ucLgCmd= ucIndice;

    usiErr= TL_SendReceive(ucLgCmd, rgucCmd, &ucLgRep, rgucRep);
    if(usiErr!= No_Err)
    {
        printf("Error %X occured",usiErr);
        exit(0);
    }
    /*Display Result*/

```

```

for(ucIndice=0 ; ucIndice < ucLgRep ; ucIndice++)
printf(" %X ",rgucRep[ucIndice]) ;
printf("\n") ;

/*===== Build Second GemCore Command*/
/*APDU Command. We select a DF File on a GemSafe Smart Card*/
USIERR= TL_SendReceive(11,"\x15\x00\xA4\x04\x00\x05\x47\x54\x4F\x4B\x31",&LgRep,Rep) ;
If(usiErr!= No_Err)
{
printf("Error %X occured",usiErr) ;
exit(0) ;
}
/*Display Result*/
for(ucIndice=0 ; ucIndice < ucLgRep ; ucIndice++)
printf(" %X ",rgucRep[ucIndice]) ;
printf("\n") ;

/*===== Build Third GemCore Command*/
ucIndice= 0 ;
rgucCmd[ucIndice++]= 0x11 ;/*Power Down*/
ucLgCmd= ucIndice ;

USIERR= TL_SendReceive(ucLgCmd,rgucCmd,&ucLgRep,rgucRep) ;
If(usiErr!= No_Err)
{
printf("Error %X occured",usiErr) ;
exit(0) ;
}
/*Display Result*/
for(ucIndice=0 ; ucIndice < ucLgRep ; ucIndice++)
printf(" %X ",rgucRep[ucIndice]) ;

printf("\n") ;

/*===== End*/
printf("Close Serial Port\n") ;
/*Close Serial Port*/
TL_CLOSE() ;
exit(0) ;
}

```

## Porting Instructions

---

The GBP Interface Library is designed to maximize independence from the environment (OS and physical port) and to be easily ported to any CPU environment.

The porting workload for a developer familiar with the C language and with both physical port management and CPU time management should be less than one week.

The GBP Interface Library implements the GBP on a virtual port, independently from the physical port communication layer (RS-232/422, I2C or equivalent).

Communication with a physical RS-232 interface in a Linux test environment is supplied as an example.

An implementation is provided as a sample on Linux using the Red Hat 7.1 distribution and on a Intel i686-based computer.

One purpose of the library consists of minimizing the required memory. Another objective consists of facilitating re-use by simplifying the update of the source code in order to port it to another OS and/or to a different hardware platform.

Whenever possible, the library uses only the necessary memory to fulfill those two requirements. It does not use coding shortcuts (which decrease memory leaking but make the source code source less clear).

All the C source code (library and samples) are commented and comply with the programming rules detailed below.

## Programming Rules

### Variable Naming: The Hungarian Notation

The Hungarian notation is a variable-naming convention that includes C++ information about the variable in its name (such as data type, whether it is a reference variable or a constant variable, etc).

There is no standard Hungarian notation and the following is just an overview of the rules applied in this product.

Prefix	Type	Example
b	boolean	bool bStillGoing;
c	character	char cLetterGrade;
str	C++ string	string strFirstName;
si	short integer	short siChairs;
i	integer	int iCars;
ii	long integer	long liStars;
f	floating point	float fPercent;
d	double-precision floating point	double dMiles;
ld	long double-precision floating point	long double ldLightYears;
sz	old-style null terminated string	char szName[NAME_LEN];
if	input file stream	ifstream ifNameFile;
is	input stream	void fct(istream &risIn);
of	output file stream	ofstream ofNameFile;
os	output stream	void fct(ostream &rosIn);
S	declaring a struct	struct SPoint {
C	declaring a class	class CPerson {
struct name or abbreviation	declaring an instance of a struct	SPoint pointLeft; SPoint ptLeft; // or abbreviation (be consistent)
class name or abbreviation	declaring an instance of a class	CPerson personFound; CPerson perFound; // or abbreviation (be consistent)

**Table 1 - Hungarian Notation Conventions**

The following table contains the letters that go before the above prefixes.

Pre-Prefix	Type	Example
u	unsigned	unsigned short usiStudents;
k	constant formal parameter	void fct(const long kliGalaxies)
R	reference formal parameter	void fct(long &rliGalaxies)
S	static	static char scChoice;
rg	array (stands for range)	float rgfTemp[MAX_TEMP];
m_	member variable of a struct or class	char m_cLetterGrade;
p	pointer to a single thing	char *pcGrade;
prg	dynamically allocated array	char *prgcGrades;

**Table 2 - Hungarian Notation Pre-Prefixes**

## Authorized Data Types

To prevent ambiguity regarding the size of the data between platform types (typically 16 bits and 32 bits), integers “signed int” and “unsigned int” are not used in the source code.

## File Header

All project files use the following header template:

```

/*****
                                TL.c - description
                                -----
begin                          : Wed May 30 2001
copyright                      : (C) 2001 by Gemplus
email                          :
*****/

```

## Function Header

All functions use the following header template :

```

/*****/
/* Description : Sends the GBP command and retrieves      */
/*              the reader response                       */
/*                                                      */
/* Input  :                                             */
/*   unsigned char ucLnCmd   -> Length of the command    */
/*   unsigned char *pszCmd   -> Buffer with the command   */
/*   unsigned char *pucLnResp -> Length of the response  */
/*   unsigned char *pszResp  -> Buffer with response     */
/*                                                      */
/* Output :                                             */
/* Returns :                                             */
/*   NO_ERR -> Everything is ok                          */
/*   ERR_LRC                               */
/*   ERR_SEQNUMBER                          */
/*   ERR_NACK                               */
/*   ERR_READERMUTE                        */
/*                                                      */
/* Global Variables :                                  */
/*                                                      */
/*                                                      */
/*****/

```

## Block Reception Algorithm

As shown in the following figure, only the transport layer knows which protocol is used. But to optimize the speed of the algorithm, the lower layer must know when to end character reception in accordance with the protocol (for example, byte 3 of the GBP block is the length of the block).

Otherwise, the only way to end block reception consists of waiting for a timeout on the character reception.

The GBP library uses a callback function between the transport layer and the lower layers. Upon reception of each character, the **TL\_Callback** command is called and the transport layer informs lower layers that block reception has been aborted.

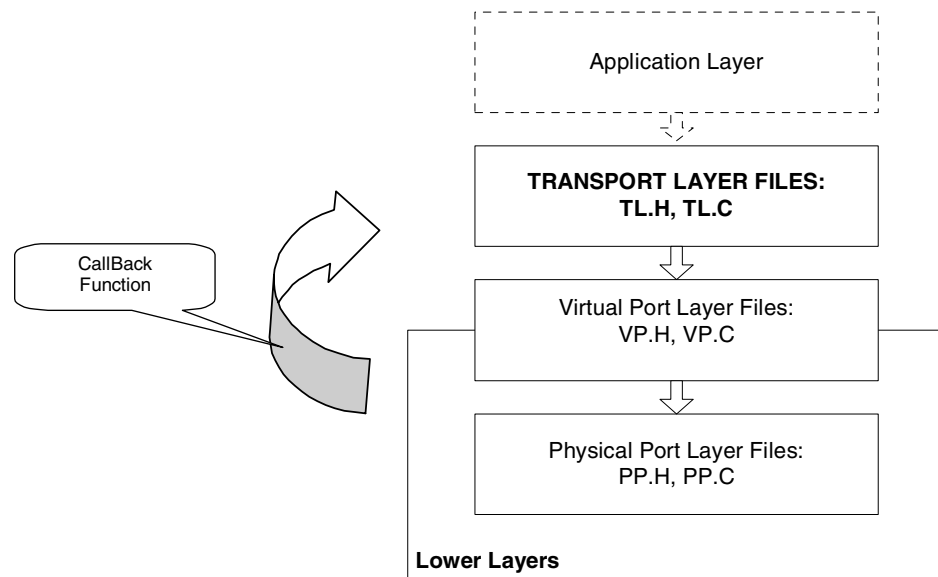


Figure 1 - The Callback Function

## GBP Interface Library files Description

The project contains the following files:

### TL.H and TL.C

These files include the API functions and should not be modified because they have been designed to be independent from the OS and from the physical port. In order to optimize the size of the library, the processing speed or multi reader management, the source code must be modified by the end developer.

### VP.H and VP.C

These files are used to virtualize physical port access, to simplify the porting process.

### PP.H and PP.C

These files are samples that illustrate physical serial port management under Linux. The user may use his own library and interface it with the virtual port layer.

### Error.H

**ERROR.H** contains the definition of all error constants.

All error codes are described in the **Errors.H** file. Each error code corresponds to the base value "ERR\_BASE" + another value X. This value is set to 1000h and it can be changed to match other customer constraints.

## GBP Interface Library Porting Instruction

**PP.C** is the only file that must be updated to handle a new physical layer. Nonetheless, **VP.C** can also be modified to handle an existing physical port management library, if necessary, without impacting either the **TL.C** module or the applications.

The GBP library provides a **PP.C** template to be completed by the developer.

### Steps to Port the Library

- Create a new directory for the project
- Copy the following files from the Gem\_TL\Linux Version\ directory to the project directory:
  - **TL.C**
  - **TL.H**
  - **VP.C**
  - **VP.H**
  - **PP.H**
  - **Errors.H**
- Copy the following file from the Gem\_TL\Template\ to the project directory:
  - **PP.C**
- Add one **Main.C** sample application according to your GemCore version
- Build all.

---

**Note:** For some compilers which use precompiled headers by default, this option must be deactivated in the settings; otherwise, compiling errors could occur.

---

- Modify the **PP.C** module to manage your own physical port layer.

## Private Commands

The following commands are used to differentiate protocol (GBP) and serial port management. They may be modified by end users to implement their own library.

The following commands must be modified in order to implement private commands:

### VP\_Open

This command calls the serial port library to configure communication parameters (Baud rate,...).

#### Format

```
short int VP_Open(char cComNumber,
                  char cBitsNumber,
                  char cParity,
                  unsigned long uliBaudRate)
```

#### Input

cComNumber	Is the number of the serial port number to be opened. Values can range from 1 to 4.
cBits	Number the number of bits used in serial communication. Values between 5 to 8.
cParity	Sets the parity used in serial communication. Values are 'O','E' or 'N'.
UliBaudRate	Sets the Baud rate. Values are 9600,19200,38400 bps.

#### Response

NO\_ERR  
 ERR\_COMNUM  
 ERR\_INITPORT

### VP\_Close

Calls the port close function provided by the physical port layer.

#### Format

```
Void VP_Close()
```

#### Input

None.

#### Response

None.

## VP\_Send

Calls the write function provided by the physical port layer.

### Format

```
short int VP_Send(unsigned short int usiLen,  
                 unsigned char *prgucBytes)
```

### Input

UsiLen                                    Is the length of the buffer to be sent.

PrgucBytes                              Is the buffer with the data to be sent.

### Response

NO\_ERR

ERR\_WRITEPORT

## VP\_Receive

Calls the **Read** command in the serial port library and uses a callback function. See the chapter “*Porting Instructions*” in this document for additional details.

### Format

```
short int VP_Receive(unsigned short int *usiResLen,  
                    unsigned char rgucResponse,  
                    short int (*pCallBack)(unsigned char*, unsigned short int))
```

### Input

pCallBack                                Is the callback function address.

### Output

usiResLen                                Contains the length of the response.

rgucResponse                             Contains the response.

### Response

NO\_ERR

ERR\_READPORT

## VP\_SetTimeOut

Sets the a new timeout value. It calls the **Timeout** function in the serial port library.

### Format

```
void VP_SetTimeOut(unsigned long uliTOut)
```

### Input

uliTOut                                  Is the new communication timeout in ms.

### Response

None.

## VP\_GetTimeOut

Gets the new timeout value. It calls the **Timeout** function in the serial port library.

### Format

```
unsigned long VP_GetTimeOut()
```

### Input

None.

### Response

Current timeout in ms.

## Example

The **VP\_Open** command provided in the sample works as follows:

```
short int VP_Open( char cCom_Nb,
                  char cBits,
                  char cParity,
                  unsigned long uliBaudRate )
{
    /*Check whether the port number is correct ( Port 1 to 4)*/
    if ( ( cCom_Nb < 1 ) || ( cCom_Nb > 4 ) )
        return ( ERR_COMNUM );

    /*Add your own function to initialize the serial port*/
    if ( PP_Open( uliBaudRate, cBits, cParity, cCom_Nb ) == FALSE )
        return ( ERR_INITPORT );

    return ( NO_ERR );
}
```

If the end developer already has a physical port library, the GBP Library **PP\_Open** command must be replaced by the developer's "open and initialize serial port command".

The code is as follows:

```
short int VP_Open( char cCom_Nb,
                  char cBits,
                  char cParity,
                  unsigned long uliBaudRate)
{
    /*Check whether the port number is correct ( port 1 to 4)*/
    if ( ( cCom_Nb < 1 ) || ( cCom_Nb > 4 ) )
        return ( ERR_COMNUM );
}
```

```
/*Add your own function to initialize the serial port*/  
if ( OpenAndInitializeSerialPort( uliBaudRate, cCom_Nb ) == FALSE )  
    return ( ERR_INITPORT ) ;  
  
    return ( NO_ERR ) ;  
}
```

All functions can be modified in the same manner, if necessary.



## Application Samples

---

Various software scenarios are proposed; they feature the use of different GemCore commands.

These scenarios are presented by GemCore version. They are written in C ANSI source code and follow the same programming rules as the library.

The purpose of these scenarios consists of:

- Rapidly evaluating GemCore
- Clarifying how to use specific GemCore features
- Using GemCore (with or without modifications) in its application software.

The scenarios do not provide a complete or partial algorithm for the requested items. Instead, the scenarios list a sequence of commands for each item showing “how to”.

The MMI for these scenarios is an output stream which uses the **Printf** command.

The developer can switch this output stream off/on with a make parameter :

- `make mmi`                   => mmi on
- `make nommi`               => mmi off

# GemCore 1.10

## GPM2K Synchronous Card

This sample shows how to use a GPM2K card with the GemCore-1.10 firmware.

The sample uses embedded GPM2K driver commands and the GemCore command interpreter to manage the card.

For further information about GemCore, see the *GemCore 1.10-Based Reader Reference Manual*.

For further information about the GPM2K synchronous card, refer to the following web site :

**<http://www.gemplus.fr/developers/products/gpm2k/index.htm>**

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets firmware version
- Disables TLP compatibility
- Sets GemCore to use the embedded GPM2K synchronous card driver
- Powers up the smart card
- Presents the secret code
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h
- Reads five bytes in the application area from address 20h
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h
- Reads five bytes in the application area from address 20h
- Powers down the smart card
- Sets GemCore to use the embedded command interpreter
- Powers up the smart card using the command interpreter
- Presents the secret code using the command interpreter
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h, using the command interpreter
- Reads five bytes in the application area from address 20h, using the command interpreter
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h, using the command interpreter
- Reads five bytes in the application area from address 20h, using the command interpreter
- Powers down the smart card
- Closes the serial port

## T=0 Smart Card Management

This sample shows how to exchange data between a T=0 smart card and the GemCore-1.10 firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Powers down the smart card
- Closes the serial port

---

**Note:** See ISO 7816-4 for additional information about the various cases for APDU commands.

---

## T=1 Smart Card Management

This sample shows how to exchange data with a T=1 smart card with the GemCore-1.10 firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

# GemCore Lite 1.21-GK

## GPM2K Synchronous Card

This sample briefly shows how to use a GPM2K card with the GemCore Lite 1.21-GK firmware.

The sample uses the GemCore command interpreter to manage the card.

For further information about this version of GemCore, see the *GemCore Lite-Based Reader Reference Manual*.

For further information about the GPM2K synchronous card, refer to the following web site:

**<http://www.gemplus.fr/developers/products/gpm2k/index.htm>**

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Sets GemCore to use the embedded command interpreter
- Powers up the smart card using the command interpreter
- Presents the secret code using the command interpreter
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h using command interpreter/
- Reads five bytes in the application area from address 20h using the command interpreter
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h using the command interpreter
- Reads five bytes in the application area from address 20h using the command interpreter
- Powers down the smart card
- Closes the serial port

## T=0 Smart Card Management

This sample shows how to exchange data under a T=0 protocol with the GemCore Lite 1.21-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Powers down the smart card
- Closes the serial port

## T=1 Smart Card Management

This sample shows how to exchange data with a T=1 smart card with the GemCore Lite 1.21-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

## EMV Smart Card Management in EMV Mode

This sample shows how to exchange data between an EMV smart card and the GemCore Lite 1.21-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

For further information about this version of GemCore, see the *GemCore EMV-Based Reference Manual*.

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Sets reader to EMV mode
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

## Non-EMV Smart Card Management when EMV Mode Fails

This sample shows how to switch between ISO and EMV modes with the GemCore Lite 1.21-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

The sample performs the following operations :

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Sets the EMV mode
- Powers up the smart card
- Sets the ISO mode
- Powers up the smart card
- Performs an **Exchange APDU** command Case 4
- Powers down the smart card
- Closes the serial port

---

# GemCore-1.21-GM

## GPM2K Synchronous Card

This sample briefly shows how to use a GPM2K card with the GemCore 1.21-GM firmware.

The sample uses the embedded GPM2K driver commands as well as the GemCore command interpreter to manage the card.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

For further information about this version of GemCore, see the *GemCore 1.21-Based Reader Reference Manual*.

For further information about the GPM2K synchronous card, refer to the following web site :

**<http://www.gemplus.fr/developers/products/gpm2k/index.htm>**

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Sets GemCore to use the embedded GPM2K synchronous card driver
- Powers up the smart card
- Presents the secret code
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h
- Reads five bytes in the application area from address 20h
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h
- Reads five bytes in the application area from address 20h
- Powers down the smart card
- Sets GemCore to use the embedded command interpreter
- Powers up the smart card using the command interpreter
- Presents the secret code using the command interpreter
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h using command interpreter
- Reads five bytes in the application area from address 20h, using the command interpreter
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h, using the command interpreter
- Reads five bytes in the application area from address 20h using the command interpreter
- Powers down the smart card
- Closes the serial port

## T=0 Smart Card Management

This sample shows how to exchange data with a T=0 smart card with the GemCore Lite 1.21-GM firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Powers down the smart card
- Closes the serial port

## T=1 Smart Card Management

This sample shows how to exchange data with a T=1 smart card with the GemCore Lite 1.21-GM firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

# GemCore 1.22

## GPM2K Synchronous Card

This sample briefly shows how to use a GPM2K card with the GemCore-1.22 firmware. It uses the GemCore command interpreter to manage the card.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

For further information about this version of GemCore, see the *GemCore Lite-Based Reader Reference Manual*.

For further information about the GPM2K synchronous card, refer to the following web site:

**<http://www.gemplus.fr/developers/products/gpm2k/index.htm>**

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Sets GemCore to use the embedded command interpreter
- Powers up the smart card using the command interpreter
- Presents the secret code using the command interpreter
- Writes five bytes 0, 0, 0, 0, 0 in the application area from address 20h, using the command interpreter
- Reads five bytes in the application area from address 20h, using the command interpreter
- Writes five bytes 1, 2, 3, 4, 5 in the application area from address 20h, using the command interpreter
- Reads five bytes in the application area from address 20h, using the command interpreter
- Powers down the smart card
- Closes the serial port

## EMV Smart Card Management in EMV Mode

This sample shows how to exchange data with an EMV smart card with the GemCore-1.22 firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Disables TLP compatibility
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

# GemCore Lite 1.32-GK

## GPM2K Synchronous Card

This sample shows how to switch between an EMV reader in ISO Mode with the firmware version GemCore Lite 1.32-GK.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

For further information about this version of GemCore, see the *GemCore Lite-Based Reader Reference Manual*.

For further information about the GPM2K synchronous card, refer to the following web site:

**<http://www.gemplus.fr/developers/products/gpm2k/index.htm>**

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Powers up the smart card
- Sets the ISO mode
- Powers up the smart card
- Performs an **Exchange APDU** command
- Powers down the smart card
- Closes the serial port

## T=0 Smart Card Management

This sample shows how to exchange data with a T=0 smart card with the GemCore Lite 1.32-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Powers up the non-EMV smart card to switch to ISO mode
- Switches to ISO mode
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 2
- Performs an **Exchange APDU** command Case 3
- Powers down the smart card
- Closes the serial port

## T=1 Smart Card Management

This sample shows how to exchange data with a T=1 smart card with the GemCore Lite 1.32-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Powers up the non-EMV smart card to switch to ISO mode
- Switches to ISO mode
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

## EMV Smart Card Management in EMV Mode

This sample shows how to exchange data with an EMV smart card with the GemCore Lite 1.32-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Powers up the smart card
- Performs an **Exchange APDU** command Case 3
- Performs an **Exchange APDU** command Case 2
- Powers down the smart card
- Closes the serial port

## Non-EMV Smart Card Management when the EMV Mode Fails

This sample shows how to switch between an EMV reader in ISO Mode with the GemCore Lite 1.32-GK firmware.

---

**Note:** To use this sample for your specific smart card, the sample must be updated to reflect your card specifications.

---

This sample performs the following operations:

- Opens the serial communication port at 9,600 bps
- Gets the firmware version
- Powers up the smart card
- Sets the ISO mode
- Powers up the smart card
- Performs an **Exchange APDU** command Case 4
- Powers down the smart card
- Closes the serial port

# Terminology

---

## Abbreviations

<b>APDU</b>	Application Protocol Data Unit
<b>API</b>	Application Programming Interface
<b>EDC</b>	Error Detection Code
<b>EMV</b>	Europay MasterCard Visa
<b>GBP</b>	Gemplus Block Protocol
<b>I-Block</b>	Information Block
<b>LRC</b>	Longitudinal Redundancy Check
<b>MMI</b>	Man Machine Interface
<b>NACK</b>	Negative Acknowledgement
<b>NAD</b>	Node Address
<b>OS</b>	Operating System
<b>PCB</b>	Protocol Control Block
<b>PP</b>	Physical Port
<b>R-Block</b>	Receive Ready Block
<b>S-Block</b>	Supervisory Block
<b>TL</b>	Transport Layer
<b>TLP</b>	Transport Layer Protocol
<b>VP</b>	Virtual Port

## Glossary

<b>Gemplus Block Protocol</b>	Simplified Gemplus-proprietary version of the T=1 card protocol. Under the GBP, data is transmitted in blocks between the source and the destination. These three types of blocks are I-blocks, R-blocks and S-blocks. For a GBP-based device, the data carried by the I-blocks are device commands (for example, GemCore commands).
<b>I-Block</b>	Information blocks hold the data to be exchanged between the source and the destination.
<b>Node Address</b>	The node address is used to identify the source and the intended destination of the block. Bits b0 to b2 are the source node address (SAD) and bits b4 to b6 are the destination node address (DAD). Bits b3 and b7 should be set to 0.
<b>R-Block</b>	Receive Ready blocks hold positive or negative acknowledgements about transmissions. A negative acknowledgement implies an error occurred in the incoming block (as indicated by the longitudinal redundancy check or the cyclic redundancy check).
<b>S-Block</b>	Supervisory blocks convey control information used to synchronize transmissions between the source and the destination.
<b>T=1 Protocol</b>	Block-oriented protocol whereby a collection of data, is transmitted as a single block between the reader and the card.
<b>Virtual Port Layer</b>	Abstraction whereby files in this layer virtualize the physical port access to simplify the porting process.