# OROS
# Reference Manual

**Version 2.33**

GEMPLUS

March, 97

At press time, this guide is as thorough and correct as possible;
however, information herein contained may have been updated after this date.
GEMPLUS reserves the right to change the functions and specifications of its products at any time without prior notice.
This document was prepared by GEMPLUS for both its clients and for its own internal use. Information herein contained is the sole property of GEMPLUS and shall not under any circumstances be reproduced without prior consent of the company.

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: E5225050/DPD01582B33

# ABOUT THIS GUIDE

This guide provides reference information about the OROS (Open Reader Operating System).

## Audience

This guide assumes that you are familiar with smart card technology, as well as your PC hardware.

## How to Use This Guide

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

### Overview

Read the Overview for an overall description of the OROS operating system.

### OROS Principles

Read the OROS Principles section for a description of the principles behind the OROS.

### OROS Protocols

Read this section for a description of the existing protocols between OROS and the host system.

### OROS Interface Commands

Read the OROS Commands section for a description of the OROS commands, the functions they perform, their syntax, and the format that you send them in to the reader.

### Using OROS with Microprocessor Cards

OROS supports ISO 7816-3 T=0 and T=1 protocol microprocessor cards. Read the Using OROS with Microprocessor Cards section for a description of these standards.

### Using OROS with Memory Cards

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

# TABLE OF CONTENTS

# OVERVIEW

OROS is a 8051 CPU family smart card reader operating system. It manages CPU time between device handlers and application tasks. These device handlers are modules that control specific devices such as the keypad, and the LCD. Application tasks are programs that call device handler commands to know what key is pressed, to print a message on the LCD, etc.

A set of system device handlers is provided with OROS that can support a number of hardware devices. Some of these hardware devices are included with the OROS reader that you are using for your application. You can also design your own hardware extension set, using following the recommendations described in the GCI400 Development Board Reference Manual.

Find in the following section a description of the way OROS controls both the device handlers and the application tasks as well as how all the system device handler commands are included in the OROS Version 2.23.

# OROS PRINCIPLES

You develop OROS applications in modules, that are called *device handlers* or *application tasks* . Each module is identified by a number.

There are two types of device handler within OROS applications: those provided by OROS are called *system device handlers* and the others, *application device handlers* .
The first ones are described in the OROS Command section: each command set, such as configuration or card interface, is a system device handler and each command is an operation.

## Modules

All the modules have a common interface called the operation list. Each module can hold up to eight *operations* , -0 to 7-. Each operation performs a function, such as reading data from a card or displaying data on an LCD. These operations constitute the interface between the modules.

Modules exchange information between them using a request/response mechanism.

Requests contain the module number, the operation number, and a list of parameters. Requests have the following format:

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

Responses return a status code and the result of the previous request operation. Responses have the following format:

<STATUS CODE> [RESULT1] [RESULT2]…

The module that sent the request must wait for the response.

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

### Operation 0

Operation 0 is the RUN operation. OROS scans for this command at every loop. This operation doesn’t have any parameters.

### Operation 1

Operation 1 called CTRL controls the behavior of the module. It has at least one parameter. Default values are shown below.

| 1st param. value | Meaning |
|---|---|
| 0 | ‘init’ - initialize device handler |
| 1 | ‘end’ - end device handler execution |
| X | Optional specific operation |

***Note****: No result is returned when CTRL is sent with the parameter ‘init’ or ‘end’.*

## Module and Operation Identity Numbers

Module numbers are coded on one byte; the 3 less significant bits are always 0.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | Module No. | | | | | 0 | 0 | 0 |

So the module number is always a multiple of 8. This coding allows to define 32 distinct modules. But it is possible to define modules with the same number. This can be useful to override or overload some operations.

The 3 less significant bits are used by the operation numbers.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | Operation No. | | |

A logical OR of these two bytes constitutes a command code.

Example : the command code 4Bh indicates the operation 3 of the module 48h.

## The Module List

OROS uses a module list to find the modules that make up an application. OROS also uses the module list for the following functions:

- Executing the CTRL operations of all modules when the system starts
- Executing the RUN operations of all application tasks on every loop of OROS
- Exchanging messages between modules.

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

The module list can contain more than one module with the same number. In this case, OROS sends the command to the first module in the list. If OROS receives the status code 01 (unknown module) it sends the command to the next module in the list with the same number. You can use this feature, to override operations. For example, to interface with smart cards (where different cards have 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 No. | | | | | Flags | | |

In addition to the Module No., bits 7 to 3 can also hold the following values:

00h, meaning that the device handler is canceled

FFh, meaning link to the next part of the table

Bits 2 to 0 are flags. The value 1 in these bits indicates 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. When polled, the remaining command is sent to the operation 2 of this module.

Bit 0 (drv_run_bit) The RUN operation is defined. If bit 2 is off, the task uses the kernel context.

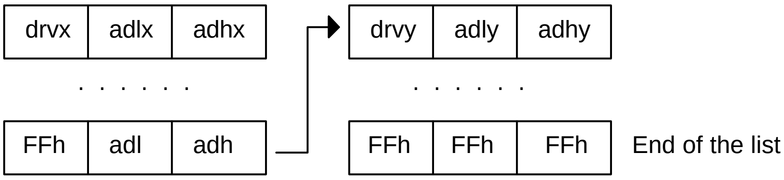
**BYTE1, BYTE2**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | Address Low | | | | Address High | | | |

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

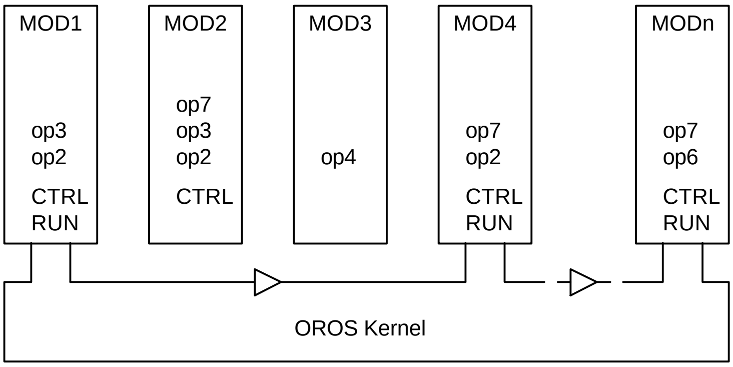
An address holding the value FFFFh indicates the end of the module list if byte 0 is set to FFh only.

**Example:**



## The OROS Kernel

The main function of the OROS kernel is the modules management. The OROS kernel is an endless loop. At startup, OROS initializes the CTRL operation of each modules. After initializing the CTRL functions, OROS activates the RUN operation of each application tasks as shown in the following diagram:

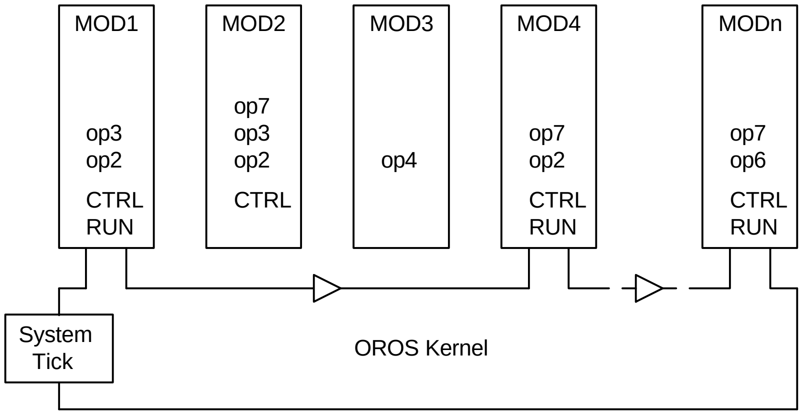


The OROS kernel provides the system entry points, the main entry points can:

- run an operation
- suspend the currently running operation until the next scan.

## Real-Time Emulation

On each loop, OROS pauses for a delay defined in the SYS_TIC data byte. The delay is in 10ms steps. The default value is 20ms.



## Interface Areas

OROS uses three Interface Areas to store the parameters of the modules (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 parameters of your modules. 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 you to develop modules with the same number as a system device handler to override some of its operations.

# OROS PROTOCOLS

All transmissions with OROS are handled by three protocol layers:

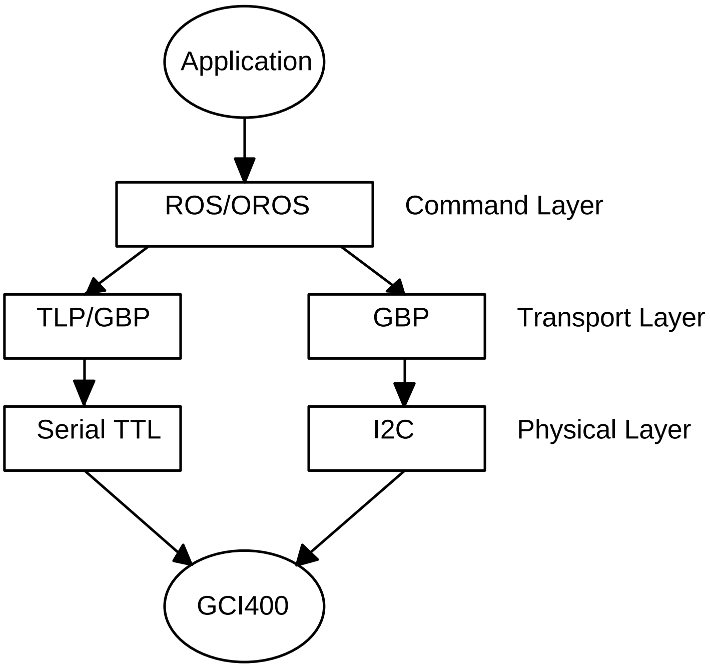
- the command layer
- the transport layer
- the physical layer

The command layer handles and interprets the OROS commands.  These commands can be either in ROS mode or OROS mode.  It consists of the command code, data, and parameters.  The ROS mode commands enable GCI/GCR200 applications to be.

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

The physical layer handles the data transmission itself. The OROS physical layer can use one of two protocols:  the I2C protocol and the Serial TTL protocol.

The following diagram shows the OROS three-layer protocol.



The following paragraphs describe the protocol layers in more details.

## Command Layer

The command layer handles and interprets the OROS commands. These commands can be either in ROS mode or OROS mode. It consists of the command code, data, and parameters.

You send commands to OROS 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 OROS Interface Commands section starting at page 14 describes the CommCode, Parameters, and Data field values for each command.

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

```
|S|Data|
```

*where:*

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

Appendix A lists the status codes and their meanings.

# Transport Layer

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

## TLP224

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

**For messages transmitted without error:**

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

*where:*

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

**When an error is detected in the transmission:**

`<NACK><LN><LRC>`

*where:*

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

During the second step the source performs the following processing:

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

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

| | ACK | LEN | Message | 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.

## GEMPLUS Block Protocol

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

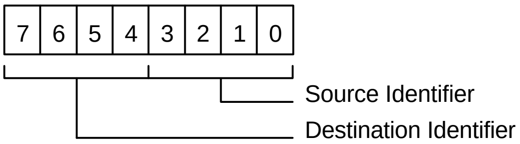
- I-Blocks. (Information Blocks). I-Blocks hold the data to be exchanged between the source and the destination.
- R-Blocks (Receive Ready Block). R-Blocks hold positive or negative acknowledgments to transmissions.
- S-Blocks (Supervisory Block). S-Blocks synchronize transmissions between the source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

The OROS and the host system exchange GBP blocks in the following format:

| NAD | PCB | LEN | DAT | EDC |
|---|---|---|---|---|

*where:*

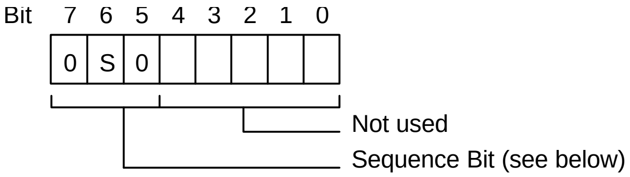
NAD is the source and the destination identifier formatted as follows on one byte:



The OROS identifier is 4 and the host system identifier is 2.

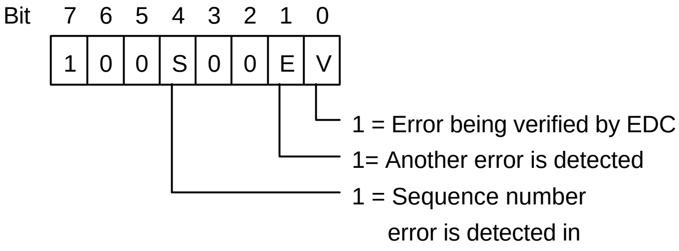
PCB identifies the block type. Its format depends on the block type, as described below:

**I-Block PCBs have the following format:**



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

**R-Block PCBs have the following format:**



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

**S-Block PCBs have the following format:**

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

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

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

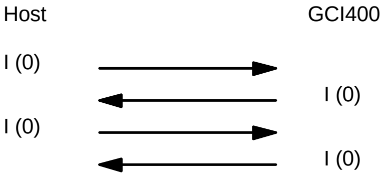
DAT holds the data being transmitted.

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

## Examples

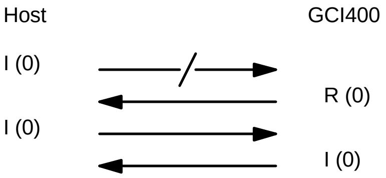
The following examples show some transmission types under the GBP protocol.

**Transmission without error:**

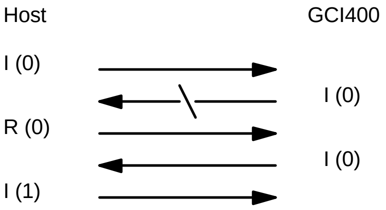


**Transmission with error:**

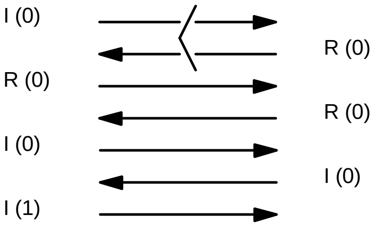
**Case 1.**



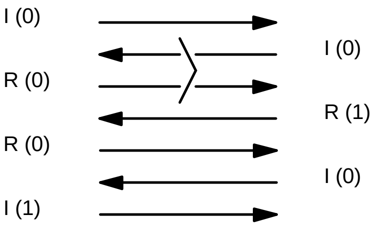
**Case 2.**



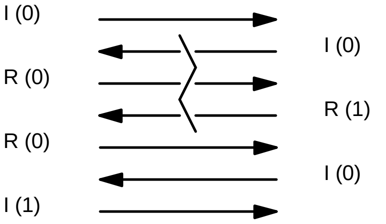
**Case 3.**



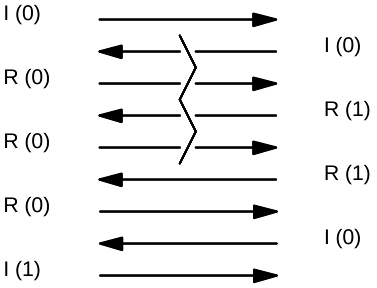
**Case 4.**



**Case 5.**



**Case 6.**



# Physical Layer

The physical layer handles the data transmission itself. The OROS physical layer can use one of two protocols: the I2C protocol and the Serial protocol.

## Serial Asynchronous Protocol

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

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

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

## I$^2$C Protocol

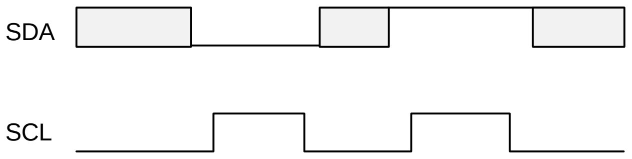
The I$^2$C bus is structured to enable all communications to be carried out using two wires, one for the data (SDA), and the other for the serial clock (SDL). Both SDA and SCL are bi-directional lines, connected to a positive supply voltage via a pull-up resistor. When the bus is free, both lines are HIGH.

The OROS is the slave during all transmissions with the host system. The host system generates the clock signal on the SCL line.

The I2C protocol enables several peripherals to be connected over the bus. Each peripheral is allocated a unique address.

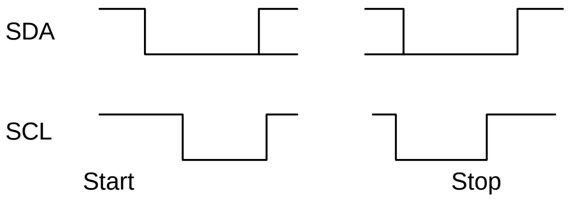
## Bit Transfer

The master generates one clock pulse for each data bit transferred. The data on the SDA line must be stable during the high period of the clock. The high or low state of the data line can only change when the clock signal on the SCL line is low. The following diagram illustrates the I2C bus bit transfer procedure.



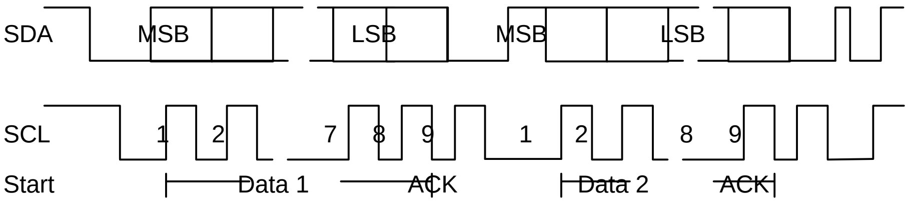
## Start and Stop Signals

The master sends the start signal by switching the SDA line from high to low while SCL is high. It sends the stop signal by switching the SDA line from low to high while SCL is high. The following diagram illustrates the I2C bus start and stop signals:



### Data Transfer

The I2C bus transfers 8-bit (1-byte) data elements. Each byte is followed by an Acknowledgment bit. Data is transferred with the most significant bit (MSB) first. If a receiver cannot receive another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL low to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases the clock line SCL. The following diagram illustrates the I2C bus data transfer procedure:



### Acknowledge

All 1-byte data transmissions must be followed by an Acknowledgment. During the Acknowledgment phase the sender positions the SDA line to high and the receiver positions SDA to low while SCL. The Acknowledgment related clock pulse is generated by the master.

Usually, a receiver which has been addressed is obliged to generate an Acknowledge after each byte has been received, except when the message starts with a CBUS address.

If the currently addressed OROS does not Acknowledge a transmission, the transmission is interrupted and the host system must send the Stop signal.

## Formats

The host sends the Start signal followed by the reader address on 7 bits, a data direction bit (read or write). The value 0 in the direction bit indicates the write direction (from the host system to the reader). The value 1 in the direction bit indicates the read direction (from the reader to the host system). The data transfer is always ended by the host system sending the Stop signal.

The host system sends commands to OROS in the following format:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Host** | [S]<ADDR>[W] | | <DATA1> | | <DATA2> | | <DATAn> | | [P] |
| **OROS** | | [A] | | [A] | | [A] | | [A] | |

The host system requests responses from OROS in the following format:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Host** | [S]<ADDR> [R] | | | [A] | | [A] | | [A] | [P] |
| **OROS** | | [A] | <DATA> | | <DATA2> | | <DATAn> | | |

The above transmission formats use the following abbreviations:
[A] = Acknowledgment
[P] = Stop signal
[R] = Direction bit set to 1. This indicates that the data is transferred from the reader to the host
[S] = Start signal
[W] = Direction bit set to 0. This indicates that the data is transferred from the host to the reader

When OROS has prepared the response, it acknowledges the request for response, then sends the response. If  OROS does not return the response immediately, the host can send again the request for response.

## Address

The host system codes the OROS address on the 7 MSBs of the first byte that it transmits. The LSB indicates the message direction. The host system codes the address as follows:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
|  | 1 | 1 | 0 | 0 | 1 | 1 | 1 | R/W |

Therefore the first byte holds the value CFh for Read and CEh for write.

# OROS INTERFACE COMMANDS

This section describes the OROS commands. For each command it describes:

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

Some OROS commands are also compatible with the Reader Operating System (ROS), so that GCR/GCI200 applications can be upgraded for use with the OROS based readers. The extension [R] to a command title indicates that command is ROS compatible.

The commands are grouped into command sets. The OROS command sets are:

- Configuration
- Card interface
- Security module interface
- Memory management commands
- Input/output commands
- LCD display commands
- Keyboard/buzzer commands
- Real Time Clock commands

The rest of this section describes the OROS commands. You can also refer to Appendix C for a summary of these commands.

## Command Format

You send commands 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 commands section describes the CommCode, Parameters, and Data field values for each command.

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

```
|S|Data|
```

*where:*

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

Appendix A lists the status codes and their meanings.

# OROS Commands

The following pages describe the OROS commands. For a summary of the OROS commands refer to Appendix B.

## Configuration

The OROS configuration commands are:

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

Find in this section a description of these commands.

## CONFIGURE SIO LINE

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

***Note**: The line is reconfigured as soon as this command is executed, therefore the fact that OROS is using the new configuration indicates that the command was successful.*

### Format

`0Ah` *`CB`*

*where:*

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

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

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

## SET MODE

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

***Notes:***

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

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

### Format

**01h 00h** [*OB*]

*where:*

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

| | Native | ROS | TLP |
|---|---|---|---|
| xxxx1xx1 | Ú | Ú | Ú |
| xxxx1xx0 | Ú | Ú | |
| xxxx0xx0 | Ú | | |

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

### Result

**S** *<mode>*

*where:*

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

| | Native | ROS | TLP |
|---|---|---|---|
| 00001001 | Ú | Ú | Ú |
| 00001000 | Ú | Ú | |
| 00000000 | Ú | | |

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

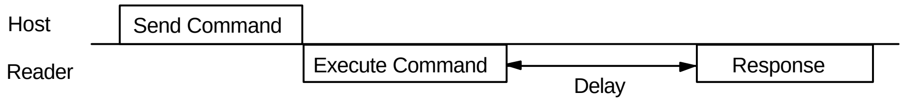
## SET DELAY

If you are using a slow host computer with the OROS reader, you can use this command to delay OROS responses.

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

*where:*

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



## READ FIRMWARE VERSION

Returns the version of the firmware installed in the reader.

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

**Result** **`S`** `Version`

*where:*

`Version` (OROS-R2.23) is the installed OROS version in ASCII.

## Card Interface Command Set

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

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

The following paragraphs describe these commands.

## POWER DOWN[R]

Use this command to power down the card. OROS powers down automatically when a card is removed.

**OROS Format** `11h`

**ROS Format** `4Dh`

**Result** `S`

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

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

## CARD PRESENCE STATUS

Indicates if a card is present in the reader.

**Format** `24h 03h`

**Result** `S` *<Status>*

*where:*

*<Status>* = is one byte specifying if a card is present in the reader. This has the following format:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | x | x | x | x | x | P | x | x |

P specifies if the card is present as follows:

| This value: | Means: |
|---|---|
| 0 | Card absent |
| 1 | Card present |

## POWER UP[R]

This command powers up and resets a card.

**OROS Format** **12h**

**ROS Format** **6Eh** `00h 00h 00h`

**Result** **S** *<card response>*

*where:*

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

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

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

R1: compatibility mode 28h: TLP

01h: ROS

R2: current card type

R3: ATR length

***Note:*** *When the TLP compatibility is enabled (* see *Set Mode command) the TA $_1$, TB$_1$, TC$_1$ and TD$_1$ bytes absent from the Answer to Reset are returned with their default value:*

| | TA$_1$ | TB$_1$ | TC$_1$ | TD$_1$ |
|---|---|---|---|---|
| Asynchronous Card | 11h | 25h | 00h | 00h |
| Synchronous Card | 00h | 00h | 00h | 00h |

## ISO OUTPUT[R]

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

**OROS Format** **13h** *CLA INS A1 A2 LN*

**ROS Format** **DBh** *CLA INS A1 A2 LN*

*where:*

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

**Result** **S**
*<data> SW1 SW2*

*where:*

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

## ISO INPUT[R]

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

**OROS Format** **14h** *CLA INS A1 A2 LN <data>*

**ROS Format** **DAh** *CLA INS A1 A2 LN <data>*

*where:*

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

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

**Result** **S** *SW1 SW2*

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

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

## EXCHANGE APDU[R]

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

### OROS & ROS Format

**15h** *APDU*

*where:*

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

### Result

**S** *Response APDU*

*where:*

*Response APDU* = the response APDU to the command. If the card replies in chained blocks OROS concatenates them. The response APDU must not exceed 252 bytes in length. See the documentation for the card in use for the APDU response details.

## APDU Format

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

APDUs can belong to one of several cases, depending on the length and contents of the APDU. OROS supports the following cases

**Case 1**-no command or response data.

**Case 2-S**hort format: command data between 1 and 255 bytes and no response data.

**Case 3-S**hort format: no command data, between 1 and 256 bytes.

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

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

## Command Format

OROS accepts commands in the following format:

| **Header** | **Body** | | |
|---|---|---|---|
| CLA INS P1 P2 | Lc | Parameters/data | Le |

The fields are described below:

### Header Fields

The Header fields are mandatory, and are as follows:

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

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

### Body Fields

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

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

### Response Format

OROS receives responses to commands in the following format.

| Body | Trailer |
|---|---|
| Data | SW1, SW2 |

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

The Trailer includes the following two mandatory bytes:

SW1: Status byte 1 that returns the command processing status

SW2: Status byte 2 that returns the command processing qualification

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

### IFSC/IFSD

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

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

## DEFINE CARD TYPE[R]

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

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

### OROS Format

**17h** *T [V [P]]*

*where:*

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

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

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

*[V]* = Programming voltage (Vpp) to be used to program the card (default = 0). This is an optional parameter that you can only use with readers with the Vpp Option. Enter a value from 50 to 250 representing the value of Vpp in 1/10 volts.

If V=0, OROS selects a default value according to card type for synchronous cards (21V for the GPM256 and 25 V for the GPM416). If a microprocessor card requires a programming voltage outside the OROS range, OROS selects 5 V.

*[P]* = Card presence byte. This is an optional parameter that you use to modify the card presence indicator options. These options are: the line over which card presence is indicated (default line is SCL) and the indicator logic selector (default is 1 = card present). If you do not use this parameter, the card presence signal is not activated.

To change either of these values, flag your option on bits 1 and 0 of this byte according to the following table:

**Values:**

| **Bit 1** | **Bit 0** | **Selects** |
|---|---|---|
| 0 | 0 | Card presence indicated on SCL line, card there = 1 |
| 0 | 1 | Card presence indicated on SCL line, card there = 0 |
| 1 | 0 | Card presence indicated on TXD line, card there = 1 |
| 1 | 1 | Card presence indicated on TXD line, card there = 0 |

***Notes**: 1. If you enter a value other than 0, this will only remain active until the next power down.*

*2. Only the programming voltage is controlled by the OROS.*

**ROS Format**

**`02h`** *T V*

*where:*

*T* = Card type selection byte. Use the same values as those indicated in the OROS Format description above.

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

*V* = Programming voltage (Vpp) to be used to program the card (default = 0). Enter a value from 50 to 250 representing the value of Vpp in 1/10 volts.

If V = 0, the ROS selects a default value according to card type for synchronous cards (21 V for the GPM256 and 25 V for the GPM416). If a microprocessor card requires a programming voltage outside the OROS range, OROS selects 5 V.

**Result**

**`S`**

## Security Module Interface Command Set

The security module interface commands manage all communications with a ISO7816-3 T=0 security module. The security module interface commands are:

- Deactivate Module
- Reset Module
- ISO Output
- ISO Input

The following paragraphs describe these commands.

## DISACTIVATE MODULE

Use this command to deactivate a module. Execute this command when OROS goes into standby mode. Execute the Reset Module command to reactivate the security module. Note that this command does not power down the security module, it stops the clock signal only.

**Format** `19h`

**Result** `S`

The deactivate command always terminates without error.

## RESET MODULE

This command resets a security module.

**Format** `1Ah`

**Result** **`S`** *`<module response>`*

*where:*

*`<module response>`* = Security module Answer to Reset.

## ISO OUTPUT

This command sends ISO Out commands, that is, commands that retrieve data from a module.

**Format**

**1Bh** *CL INS A1 A2 LN*

*where:*

*CLA*, *INS*, *A1*, *A2*, and *LN* are the five ISO header bytes. For details about the ISO header contents refer to the documentation for the card module you are using.

**Result**

**S** *<data> SW1 SW2*

*where:*

*<data>* = Up to 248 bytes of data returned by the card. If a module error or OROS error is detected (s<>0 and s<>E7h), OROS does not return any data.
The module can return any number of bytes up to LN.

## ISO INPUT

This command sends ISO In commands, that is, commands that send data to a module.

**Format**

**1Ch** *CLA INS A1 A2 LN <data>*

*where:*

*CLA*, *INS*, *A1*, *A2*, and *LN* are the five ISO header bytes. For details about the ISO header contents refer to the documentation for the module you are using.

*<data>* represents the*LN* data bytes transmitted to the card after the ISO header. The maximum length of the data is 252 bytes.

**Result**

**S** *SW1 SW2*

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

## Reader Memory Management Commands

The reader memory management commands manage the reader memory. The reader memory management commands are:

- Read Memory
- Write Memory
- Erase Memory
- Select Page
- Read CPU Port
- Write CPU Port

This section describes these commands.

## Read Memory

Reads the contents of all memory areas that can be addressed by the reader. This command is active, if the memory considered is not read protected.

**Format**

**22h** *Type [Page] ADH ADL LN*

*where :*

*Type* = the type of memory to read, mapped as follows:

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

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

TTTT is the type of memory to read

| **This value** | **Specifies this memory type** |
|---|---|
| 0001 | IDATA (Internal CPU data memory) |
| 0010 | XDATA (external data memory) |
| 0101 | CODE memory |

*Page* = one optional byte indicating the Xdata and Code page to select before a read.

If this parameter is not present, the current selected page is read.

*See* Select External Memory Page command for details.

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

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

*LN* = the length of data to read in bytes.

**Result**

**S** <data bytes>

## Write Memory

Writes to all memory areas that can be addressed by the reader. This command is active if the considered memory is not write protected.

**Format**

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

*where :*

*Type* = type of memory to be written to, mapped as follows:

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

E = FLASH /EEPROM memory type flag (only for XDATA or CODE memory types)

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

TTTT is the type of memory to read.

| **This value** | **Specifies this memory type** |
|---|---|
| 0001 | IDATA (Internal CPU data memory) |
| 0010 | XDATA (external data memory) |
| 0101 | CODE memory |

*Examples:*
1. Write one byte in RAM memory, in the XDATA area at the 8000h location:
23h 02h 80h 00h 01h <Data>

2. Write one byte in FLASH memory, in the CODE area at the 8000h location:
23h 85h 80h 00h 01h <Data>

*Page* = optional byte indicating the Xdata and Code page to select before the write command. If this parameter is not present, the current selected page is written.
See ***Select External MemoryPage*** command for details.
Note that the current page is not modified.

*ADH,ADL* = the 16-bits address of the first byte of memory to write to. *ADH* is the most significant byte and *ADL* is the least significant byte.

*LN* = the length of data to read in bytes
<data> = the data to write

**Result** **S**

## Memory read and write protection

Both program memory and data memory may be protected against read or write commands. Two codes of 8 bytes each can be used; the first code protects the program memory, and the second the data memory.

When the program memory is protected:

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

When the data memory is protected:

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

To be efficient, the data memory protection must be used with a protected program memory.

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

The program memory protection code is located from the address FFB0 to the address FFB7.

The data memory protection code is located from the address FFA0 to the address FFA7.

To be validated, the 8 bytes of the protection code must be followed by 8 bytes representing the complemented 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 (code not validated). The program memory is protected.

To enable the read or write access to one protected area, the following write command must be used:

code memory:    23 X5 FF B0 08 < 8 bytes code >
data memory:    23 X2 FF A0 08 < 8 bytes code >

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

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

## Erase Flash Memory

Erases part or all of the contents of the flash memory. Note that this command can take up to one minute to be executed. This command is enabled if the considered memory is not write protected.

**Format**

**26h** *Type [Page] ADH ADL <CODE>*

*where :*

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

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

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

TTTT is the type of memory to erase.

| **This value** | **Specifies this memory type** |
|---|---|
| 0010 | XDATA memory |
| 0101 | CODE memory |

*Page* = one optional byte indicating the Xdata and Code page to select before a write. If this parameter is not present, the current selected page is erased.

*See* Select External Memory Page command for details.

Note that the current page is not modified.

*ADH,ADL* = the 16-bits erase start address. *ADH* is the most significant byte and *ADL* is the least significant byte.

<CODE> = the erase command code.

It will be 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).

*Example*

The following commands erase data held in an AMD 29F010 FLASH memory starting from address 8000H and used for program storage.

Memory configuration:

| | Page#0 | Page#1 | Page#2 | Page#3 |
|---|---|---|---|---|
| 8000h<br>BFFFh | Sector 1 | Sector1 | Sector 1 | Sector 1 |
| C000h<br>FFFFh | Sector 2 | Sector 2 | Sector 2 | Sector 2 |

| | |
|---|---|
| Chip erase command: | 26h 85h D5h 55h 10h |
| First Sector erase command: | 26h 85h C0h 00h 30h |
| Second Sector erase command: | 26h 85h E0h 00h 30h |
| First sector in code page 2 erase command: | 26h C5h 20h E0h 00h 30h |

**Result** **S**

## Select External Memory Page

OROS can manage up to eight 32Kbyte page of CODE memory, and sixteen 32Kbyte of XDATA memory. This command enables you to select the active page.

CODE Page 0 and XDATA Page 0 are selected by default on power up.

**Format** **27h** *Page*

*where :*

*Page* = one 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 | D | D | D | D |

Bits 3 to 0: indicate the XDATA page to select.

Bits 6 to 4: indicate the CODE page to select.

Bit 7: not used.

XDATA organization:

| | | | | | | |
|---|---|---|---|---|---|---|
| 8000h<br>FFFFh | Page#0 | Page#1 | Page#2 | Page#3 | ... | Page#15 |
| Parameter | X0h | X1h | X2h | X3h | ... | XFh |

CODE organization:

| | | | | | | |
|---|---|---|---|---|---|---|
| 8000h<br>FFFFh | Page#0 | Page#1 | Page#2 | Page#3 | ... | Page#7 |
| Parameter | 0Xh | 1Xh | 2Xh | 3Xh | ... | 7Xh |

**Result** **S**

## Read CPU Port

Reads the state of a CPU port.

**Format**

**24h** *PORT*

*where:*

*PORT* = number of the port to read according to the following table:

| **This value:** | **Specifies this port:** |
|---|---|
| 00 | Port0- P0 |
| 01 | Port1-P1 |
| 02 | Port2-P2 |
| 03 | Port3-P3 |

**Result**

**S** *Value*

*where:*

*Value* = the value read from the specified CPU port.

## Write CPU Port

Writes to a CPU port.

**Format**

**25h** *PORT VALUE*

*where:*

*PORT* = number of the port to read according to the following table:

| **This value:** | **Specifies this port:** |
|---|---|
| 00 | Port0-P0 |
| 01 | Port1-P1 |
| 02 | Port2-P2 |
| 03 | Port3-P3 |

*VALUE* = the value to write to the CPU port.

**Result**

**S** *Value*

*where:*

*Value* = the value output to the specified CPU port.

## Input/Output Commands

The reader can control eight input lines and eight output lines. The input lines and output lines are independent from each other. The output lines are set to zero on power up. OROS has the following input/output commands:

- Read Input Lines
- Read Output Lines
- Write To Output Bit

Find in the present section, a description of these commands.

## Read Input Lines

Reads the values of the 8 input lines.

**Format** **42h**

**Result** **S** *IN*

*where:*

*IN* = the input line values.

## Read Output Lines

Reads the values of the 8 output lines.

**Format** **43h**

**Result** **S** *OUT*

*where:*

*OUT* = the output line values.

## Write To Output Bit

Writes the specified value to one of the eight output bits.

**Format** **44h** *BIT VALUE*

*where:*

*BIT* = the bit number to write to. Enter a value from 0 to 7 where 0 specifies bit 0, 1 specifies bit 1, and so on.

*VALUE* = the value to write to the specified bit. Enter 00h for 0 and 01h for 1.

**Result** **S**

## LCD Commands

The LCD commands control the LCD.

- Power Down LCD
- Power Up and Clear LCD
- Display Character String
- Display Character
- Send LCD Command

This section describes these commands.

## Power Down LCD

Powers down the LCD.

**Format** **29h**

**Result** **S**

## Power Up and Clear LCD

Powers up the LCD.

If the LCD is already powered, this command clears it.

**Format** **2Ah**

**Result** **S**

## Display Character String

Displays a string of characters on the LCD.

**Format** **2Bh** *[POS] CHARS*

*where:*

*[POS]* = the start position of the string of characters. This 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 in the current cursor position. Bit 7 of this byte must always be set to 1.

*CHARS* = the string of characters to display in ASCII.

**Result** **S**

## Display Character

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

**Format**

**`2Ch`** *`CHAR`*

*where:*

*`CHAR`* = the character to display in ASCII.

**Result**

**S**

## Send LCD Command

Sends a LCD control command.

**Format**

**`2Dh`** *`COMCODE`*

*where:*

*`COMCODE`* = one of the command codes listed in the following table:

| This command code: | Does this: |
|---|---|
| 01h | Clears the LCD |
| 02h | Positions the cursor in the first character position on the first line. |
| 04h | Moves the cursor one place to the right. |
| 05h | Moves the line one place to the right. |
| 06h | Moves the cursor one place to the left. |
| 07h | Moves the line one place to the left. |
| 0Ch | Hides the cursor. |
| 0Dh | Flashes the character in the cursor position. |
| 0Eh | Displays the cursor. |
| 0Fh | Displays a flashing cursor. |
| 10h | Moves the cursor one place to the left. |
| 14h | Moves the cursor one place to the right. |
| 18h | Moves the cursor and the text to the right of it one place to the left. |
| 1Ch | Moves the cursor and the text to the left of it one place to the right. |

**Result**

**S**

## Keyboard and Buzzer Commands

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

- Set Key Press Timeout
- Sound Buzzer

## Set Key Press Timeout

Sets the number of seconds the reader waits for a key to be pressed and turns a 25 ms buzzer for when the key is pressed on or off.

**Format** **32h** *TIME BEEP*

*where:*

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

*BEEP* = buzzer on/off. The value 00h turns it off and 01h turns it on.

**Result** **S** *KEY*

*where:*

KEY = the key code of the key that was pressed if it was pressed before timeout. 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** *DURATION [FREQ]*

*where:*

*DURATION*: the buzzer duration. The units of duration are a function of the frequency *[FREQ]* (see below).

*[FREQ]*: the sound frequency, in the range 1183Hz to 68 267Hz. This is an optional parameter. If you omit it, the sound frequency defaults to 6300Hz.

You can use the following formulas to get approximate values for these parameters:

*DURATION* = T(ms) * N(Hz) / 36000

*[FREQ]* = 307200 / N(Hz) – 4

**Result** **S**

## Real Time Clock Commands

The real time clock commands read and update the reader clock date and time:

- Read Date and Time
- Update Date and Time

## Read Date and Time

Reads the real time clock date and time.

**Format** **3Ah**

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

*where:*

| | |
|---|---|
| *YEAR* | = the new year value, in BCD. |
| *MONTH* | = the new month value, in BCD. |
| *DAY* | = the new day value, in BCD. |
| *HOUR* | = the new hour value, in BCD. |
| *MINUTE* | = the new minute value, in BCD. |
| *SECOND* | = the new 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* | = the new year value, in BCD. |
| *MONTH* | = the new month value, in BCD. |
| *DAY* | = the new day value, in BCD. |
| *HOUR* | = the new hour value, in BCD. |
| *MINUTE* | = the new minute value, in BCD. |
| *SECOND* | =the new second value, in BCD. |

**Note:** *You must enter a value for all the above fields.*

# USING OROS WITH MICROPROCESSOR CARDS

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

## Clock Signal

OROS can transmit one of two clock frequency values to the card, depending on the previously selected operating mode:

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

You specify the operating mode while selecting the card type using the DEFINE CARD TYPE command (*see* page 25). Select card type 02h for the standard mode and card type 12h for the double speed mode.

## Global Interface Parameters

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

### TA1 Parameter

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

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

After it receives the Answer to Reset, OROS installs the communication rate indicated by TA1. Table1 and Table 2 below show the clock rate conversion factors, the bit rate conversion factors, and the baud rates installed in relation to the TA1 values for standard mode and double speed mode cards.

***Note**: OROS only supports the TA1 shaded values in Tables 1 and 2.*

### TB1 and TB2

If the Vpp option is available on the reader, OROS interprets these two parameters to match its programming voltage to that of the card. As described in the ISO standard, the value can be adjusted from 5 to 25V with 1/10V steps.

If the Vpp option is not available on the reader, OROS ignores these parameters and sets the Vpp value to 5V.

## TC1

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

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

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

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

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

# Communication Protocols

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

| **This value:** | **Selects this 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

OROS interprets the TC2 specific interface parameter to set the value of the work waiting time W.  When this parameter is absent OROS waits for a maximum of 960xD ETU before timing-out on a character sent by the card. Otherwise OROS waits for a maximum of 960xDxW before timing-out.

To send instructions to a T=0 microprocessor card, you use the OROS ISO Input and ISO Output commands described on page 22.

## T=1 Protocol

To send instructions to a T=1 microprocessor card, you use the OROS *Exchange APDU* command described pages 23 and 24.

OROS interprets the T=1 specific interface bytes according to clause 9 of the ISO 7816-3 standard.  These bytes are $TA_3$, $TB_3$, $TC_3$.

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

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

$TC_3$ defines the Error Detection Code (EDC) type.

# USING OROS WITH MEMORY CARDS

Memory cards cannot interpret smart card instructions in the same way as ISO 7816-3 microprocessor cards can. OROS therefore interprets T=0 formatted instructions and converts them into the appropriate timing sequences required to control the memory cards listed in the table below.

As they are interpreted by OROS itself and not by the card, these instructions are summarized in the table below.  For further details, refer to the relevant card documentation.

You send these instructions to the reader, using the OROS ISO Input and ISO Output commands described page 22.

**Memory Card Command Summary**

| | | **ISO Input/ISO Output Command Parameters (CLA = 00)** | | | |
|---|---|---|---|---|---|
| **Card Type** | **Command Name** | **INS** | **A1** | **A2** | **Ln** |
| GPM256 | Write Bytes | D0 | 00 | Start Address | Write Length |
| | Read Bytes | B0 | 00 | Start Address | Read Length |
| GPM103 | Write Bytes | D0 | 00 | Start Address | Write Length |
| | Erase and Write Carry | E0 | 01 | Counter to erase | 0 |
| | Write New value to Counter | D2 | 05 | 08 | 02 |
| | Read Bytes | B0 | 00 | Start Address | Read Length |
| | Read Counter Value | B2 | 05 | 08 | 02 |

**Continued on following page**

**Memory Card Command Summary (continued)**

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

**Continued on following page**

**Memory Card Command Summary (continued)**

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

# APPENDIX A.  STATUS CODES

The status codes returned by OROS commands are listed in the table below.

| **Code** | **Meaning** |
|---|---|
| 01h | Unknown driver or command. |
| 02h | Operation not possible with this driver. |
| 03h | Incorrect number of arguments. |
| 04h | Reader command unknown. The first byte of the command is not a valid command code. |
| 05h | Response too long for the buffer. |
| 09h | Communication protocol error. The header of a message is neither ACK nor NACK (60h or E0h) |
| 10h | Response error at the card reset. The first byte of the response (TS) is not valid |
| 11h | ISO command header error. The byte INS in the ISO header is not valid (6x or 9x). |
| 12h | Message too long. The OROS buffer is limited to 254 bytes, of which 248 bytes are fc the data exchanged with the card |
| 13h | Byte reading error returned by an asynchronous card. |
| 15h | Card turned off. A Power Up command must be applied to the card prior to any other operation. |
| 16h | Programming voltage not available. The parameter V in the DEFINE CARD TYPE command is not valid. |
| 17h, 18h | Communication protocol unknown or incorrectly initialized. |
| 19h | Illegal access to external bus. |
| 1Ah | Error in an ISO format card command. The parameter LN in the ISO header does not correspond to the actual length of the data. |
| 1Bh | A command has been sent to OROS with an incorrect number of parameters. |
| 1Dh | The check byte TCK of the response to reset of a microprocessor card is incorrect. |
| 1Eh | An attempt has been made to write to external memory, which is write protected. |
| 1Fh | Incorrect data has been sent to the external memory. This error is returned after a wri check during a downloading operation. |
| A0h | Error in the card reset response, such as unknown exchange protocol, or byte TA1 nc recognized. The card is not supported by OROS. The card reset response is nevertheless returned. |
| A1h | Card protocol error (T=0/T=1). |
| A2h | Card malfunction. The card does not respond to the reset or has interrupted an exchange (by time-out). |
| A3h | Parity error (in the course of an exchange microprocessor). The error only occurs afte several unsuccessful attempts at re transmission. |

A4h Card has aborted chaining (T=1).

A5h Reader has aborted chaining (T=1).

A6h Protocol type Selection (PTS) error.

CFh Overkey already pressed.

E4h The card has just sent an invalid "Procedure Byte" to OROS (*See* ISO 7816-3).

E5h The card has interrupted an exchange with OROS (the card sends an SW1 byte but OROS has more data to send or receive).

E7h Error returned by the card. The bytes SW1 and SW2 returned by the card are differer from 90h 00.

F7h Card removed. The card has been withdrawn in the course of carrying out of a command. Check that the card instruction is not partially completed.

F8h The card is consuming too much electricity or is short circuiting (Icc > 100 mA).

FBh Card absent. There is no card in the smart card interface. The card may have been removed when it was powered up, but no command has been interrupted.

# APPENDIX B. OROS COMMAND SUMMARY

The following table summarizes the OROS commands.

| Command Set | Command Name | Command Code | Parameters | Action | Page |
|---|---|---|---|---|---|
| Configuration | Configure SIO Line | 0A | Parity, Bits per character, Baud rate | Modifies the SIO line configuration | 16 |
| | Set Mode | 01 | 00 + Mode to select (TLP, Normal, or ROS/TLP disabled). | Defines the reader operation mode | 17 |
| | Set Delay | 23 | 01h 00h 4Ch 01h | Delays GCR500 Responses | 18 |
| | Read Firmware Version | 22 | 05, 3F, F0, 10 | Returns the firmware version | 18 |
| Card Interface | Power Down | 11 | - | Powers down a card | 20 |
| | Power Up | 12 | - | Powers up a card | 21 |
| | ISO Output | 13 | ISO command header | Retrieves data from the smart card (ISO OUT) | 22 |
| | ISO Input | 14 | ISO command header | Sends data to the smart card (ISO IN) | 22 |
| | Exchange APDU | 15 | APDU command | Sends an APDU command | 23 |
| | Define Card Type | 17 | card type, programming voltage, card presence options | Sets card type and voltage and card presence options | 25 |
| Security Module Interface | Deactivate Module | 19 | - | Deactivates the security module | 28 |
| | Reset Module | 1A | - | Resets the security module | 28 |
| | ISO Output | 1B | ISO command header | Gets data from the module (ISO OUT) | 29 |
| | ISO Input | 1C | ISO command header | Sends data to the module (ISO IN) | 29 |

**Continued on the following page**

**OROS Command Summary (continued from the previous page)**

| Command Set | Command Name | Command Code | Parameters | Action | Page |
|---|---|---|---|---|---|
| Reader Memory Management | Read Memory | 22 | Memory type, Start address, Read length | Reads data from the card memory | 31 |
| | Write Memory | 23 | Memory type, start address, write data length | Writes to a card memory | 32 |
| | Erase Flash Memory | 26 | | | 34 |
| | Select External Memory Page | 27 | Page to select | Selects an active page | 36 |
| | Read CPU Port | 24 | Port number to read | Reads the state of a CPU port. | 37 |
| | Write CPU Port | 25 | Port number, value | Writes to a CPU port | 37 |
| | Read Input Lines | 42 | - | Reads the values of the 8 input lines. | 39 |
| | Read Output Lines | 43 | - | Reads the values of the 8 output lines. | 39 |
| | Write To Output Bit | 44 | Bit number | Writes an output bit. | 39 |
| LCD Commands | Power Down LCD | 29 | **-** | Powers down the LCD | 41 |
| | Power Up and Clear LCD | 2A | - | Clears the LCD | 41 |
| | Display Character String | 2B | Start Position, Characters | Displays a string | 41 |
| | Display Character | 2C | Character to display | Displays a character on the LCD in the current cursor position. | 42 |
| | Send LCD Command | 2D | Command code | Sends a LCD control command. | 42 |

**Continued on the following page**

**OROS Command Summary (continued from the previous page)**

| Command Set | Command Name | Command Code | Parameters | Action | Page |
|---|---|---|---|---|---|
| Keyboard and Buzzer Commands | Set Key Press Timeout | 32 | Timeout, Buzzer | Sets the number of seconds the reader waits for a key to be pressed and turns a warning on or off. | 44 |
| | Sound Buzzer | 33 | Duration, Frequency | Sounds the buzzer and specifies its frequency and duration. | 44 |
| Real Time Clock Commands | Read Date and Time | 3A | - | Reads the real time clock date and time | 46 |
| | Update Date and Time | 3B | New date and time | Updates the real time clock date and time. | 46 |

# APPENDIX C INTERPRETED SYNCHRONOUS SMART CARD DRIVER

## CARD TYPE 01h

This command deals with synchronous cards protocols not supported by OROS. The protocol to use is sent as a parameter in the command as the assembler code 8051.

The 8051 assembler type INTEL ASM51 generates the code to be executed; the OROS software interprets the bytes as opcodes 8051.

The OROS interpreter enables the execution of most of the 8051 instructions along with a few macro commands dedicated to synchronous cards.

### Format

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

*where:*

| | |
|---|---|
| *CLA, INS, A1, A2:* | Command parameters. |
| *Lin:* | Number of bytes presents in the DATA IN field. |
| *DATA IN:* | DATA to send to the card. |
| *Lout:* | Length of the expected response. |
| *Lcode:* | Number of bytes presents in the CODE field. |
| *CODE:* | 8051 executable code. |

### Result

**S <data byte>**

## INTERPRETER 8051

The OROS interpreter deals with the following functions:

- an accumulator A
- eight registers R0 to R7
- the carry C
- the program counter PC

All instructions operating on the RAM IDATA or XDATA memory, act on the XDATA memory. The XDATA memory starts at address 0000h and ends at address 00FFh.

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

Only relative jumps can be used.

## Initialization

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

PC points to the code first byte.

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

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

## Card Presence

Before any 16h command is executed, the software checks for the presence of a card in the smart card connector.

The absence of card will immediately return the error message “CARD ABSENT” (CRL = FBh).

## Card Withdrawal

As soon as the smart card is powered on, the OROS card withdrawal interruption is activated.

In the eventuality of card withdrawal, the interpreted program is interrupted, all contacts with the smart card are deactivated and the following error message “CARD WITHDRAWN” (CRL = F7h) is returned.

## Short Circuit

The card powered on instructions check the absence of short circuit in between pins C1 (VCC) and C5 (GND).

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

## INSTRUCTIONS

Carry out the following procedure to obtain an instruction hexadecimal code; the line number defines the four most significant bits and the column number defines the least significant bits (e.g. INC A = 04h).

***Note:*** *Use the instructions displayed in italic as macro-commands; read further for more details.*

| | 0 | *1* | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NOP<br>1 /  12 | *VCC_OFF*<br>1 / 13 | | RR A<br>1 / 16 | INC A<br>1 / 18 | | INC @R0<br> 1/ 22 | INC @R1<br>1 / 22 |
| 1 | | *VCC_ON*<br>1 / 13 | *RESET*<br>1 / | RRC A<br>1 / 19 | DEC A<br>1 / 18 | | DEC @R0<br>1 / 22 | DEC @R1<br>1 / 22 |
| 2 | | *CLR_RST*<br>1 / 13 | RET    (*)<br>1 / | RL A<br>1 / 14 | ADD A,#data<br>2 / 21 | | ADD A,@R0<br>1 / 26 | ADD A,@R1<br>1 / 26 |
| 3 | | *SET_RST*<br>1 / 13 | RETI    (*)<br>1 / | RLC A<br>1 / 21 | ADDC A,#data<br>2 / 24 | | ADDC A,@RO<br>1 / 29 | ADDC A,@R1<br>1 / 29 |
| 4 | JC  rel<br>2 / 15 / 19 | *CLR_IO*<br>1 / 13 | *RET_0K*<br>1 / | | ORL A,#data<br>2 / 17 | | ORL A,@R0<br>1 / 22 | ORL A,@R1<br>1 / 22 |
| 5 | JNC  rel<br>2 / 15 / 20 | *SET_IO*<br>1 / 13 | *RET_NOK*<br>*2 /* | | ANL A,#data<br>2 / 17 | | ANL A,@R0<br>1 / 22 | ANL A,@R1<br>1 / 22 |
| 6 | JZ rel<br>2 / 15 / 19 | *CLR_CLK*<br>1 / 13 | | | XRL A,#data<br>2 / | | XRL A,@R0<br>1 / | XRL A,@R1<br>1 / |
| 7 | JNZ  rel<br>2 / 17 / 20 | *SET_CLK*<br>1 / 13 | *CLK_INC*<br>1 / 14 / XXX | *CLK_INC8*<br>1 / 14 / XXX | MOV A,#data<br>2 / 19 | | MOV A,@R0<br>1 / 27 | MOV A,@R1<br>1 / 27 |
| 8 | SJMP  rel<br>2 / 16 | *CLR_C4*<br>1 / 13 | *RDL_BYTE*<br>1 / | *RDH_BYTE*<br>1 / | | | | |
| 9 | | *SET_C4*<br>1 / 13 | *WRL_BYTE*<br>1 / | *WRH_BYTE*<br>1 / | SUBB A,#data<br>2 / | | SUBB A,@R0<br>1 / 29 | SUBB A,@R1<br>1 / 29 |
| A | | *CLR_C8*<br>1 / 13 | *RES_PUL*<br>1 / 24 | | | | | |
| B | | *SET_C8*<br>1 / 13 | | CPL C<br>1 / 14 | CJNE A, #data, rel<br>3 / 27 / 38 | | CJNE @R0 , #data, rel<br>3 / 33 | CJNE @R1 ,#data, rel<br>3 / 33 |
| C | | *SET_VPP*<br>2 / | *WAIT_US*<br>2 / 10 / 2550 | CLR C<br>1 / 14 | SWAP A<br>1 / 15 | | XCH A,@R0<br>1 / 27 | XCH A,@R1<br>1 / 27 |
| D | | | *WAIT_MS*<br>2/ 1ms/ 255ms | SETB C<br>1 / 14 | | | XCHD A,@R0<br>1 / 25 | XCHD A,@R1<br>1 / 25 |
| E | | *IO_TO_C*<br>1 / 16 | | | CLR A<br>1 / 14 | | MOV A,@R0<br>1 / 25 | MOV A,@R1<br>1 / 25 |
| F | | *C_TO_IO*<br>1 / 15 | | | CPL A<br>1 / 14 | | MOV @R0,A<br>1 / 25 | MOV @R1,A<br>1 / 25 |

1 / 12 / 23 : Instruction over an byte / 12us min / 23us max

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

**INSTRUCTIONS**

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

1 / 12 / 23 : Instruction over an byte / 12us min / 23us max

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

# MODIFIED INSTRUCTIONS

### RET

When the interpreter encounters the RET code, it indicates the end of the program execution. OROS sends back the RAM XDATA data; R4 points to the first byte to send back and R0 points to the byte following the last byte in the answer.

### RETI

When the interpreter encounters the RETI code, it indicates the end of the code execution. OROS sends back the registers contents in the following order:

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

Use this instruction for software development.

# MACRO-COMMANDS

### %RET_OK

When the interpreter encounters the RET_OK code, the program execution ends. OROS sends back the last RAM XDATA contents; R4 points to the first byte to return and R0 points to the byte following the last byte of the answer.

CRL takes  the value 00h and two other bytes SW1 = 90h and SW2 = 00H are added to the message tail.

### %RET_NOK (ERROR)

When the interpreter encounters the RET_NOK instruction, the program execution ends. OROS sends back the last RAM XDATA contents; R4 points to the first byte to return and R0 points to the byte following the answer’s last byte.

CRL takes the value E7h,  SW1 = 92h and SW2 takes the ERROR value.  These two bytes are added to the message tail.

### %VCC_OFF

This command powers off all the smart card contacts according to the sequence specified in the ISO 7816-3 norm.

### %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.
- IO 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 RESET contact to 0.

### %SET_RST

This instruction sets the smart card RESET contact to 1.  It will only function if the smart card is powered on.

**%CLR_IO**

This instruction sets the smart card IO contact to 0.

**%SET_IO**

This instruction sets the smart card IO contact to 1. It will only function if the smart card is powered on.

**%CLR_CLK**

This instruction sets the smart card CLOCK contact to 0.

**%SET_CLK**

This instruction sets the smart card CLOCK contact to 1. It will only function if the smart card is powered on.

**%CLR_C4**

This instruction sets the smart card C4 contact to 0.

**%SET_C4**

This instruction sets the smart card C4 contact to 1. It will only function if the smart card is powered on.

**%CLR_C8**

This instruction sets the smart card C8 contact to 0.

**%SET_C8**

This instruction sets the smart card C8 contact to 1. It will only function if the smart card is powered on.

**%SET_VPP (VALUE )**

This instruction sets the smart card VPP contact to the voltage specified in the VALUE parameter and waits for 200us (VPP rise time). It will only function if the smart card is powered on.

***Note:*** *the VPP voltage value is coded in VALUE, steps 0.1V.*

**%IO_TO_C**

This instruction copies the IO contact state in the C bit.

**%C_TO_IO**

This instruction sets the level held in C on the smart card IO contact. It will only function if the smart card is powered on.

**%CLK_INC**

This instruction enables 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 us then to 1 for 10 us. At the end of the sequence, CLK is set to 0.

**%CLK_INC8**

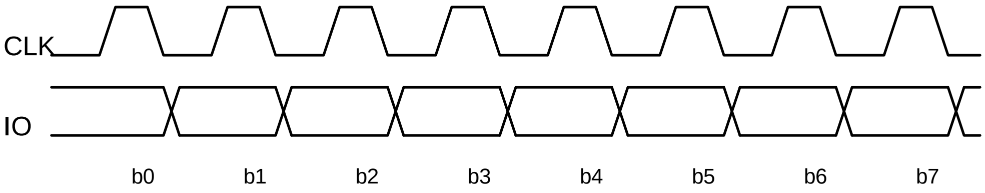
This instruction enables 8 pulses packets to be generated on CLK. The total number of packets is indicated in A (0 to 255).

CLK is set to 0 for 10 us then to 1 for 10 us. At the end of the sequence, CLK is set to 0.

## %RDL_BYTE

This command ******lits 8 bits and classes them in A.

The sequence is as follows:



- CLK contact set to 0 for 10us.

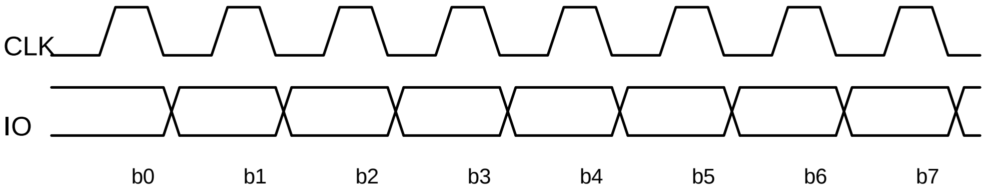
- CLK contact set to 1 for 10us.

The IO line is read 5us before the CLK rising edge.

## %RDH_BYTE

This command lits 8 bits and order them in A.

The sequence is as follows:



- CLK contact set to 0 for 10us.

- CLK contact set to 1 for 1us.

The IO line is read 5us after the rising edge of the clock.

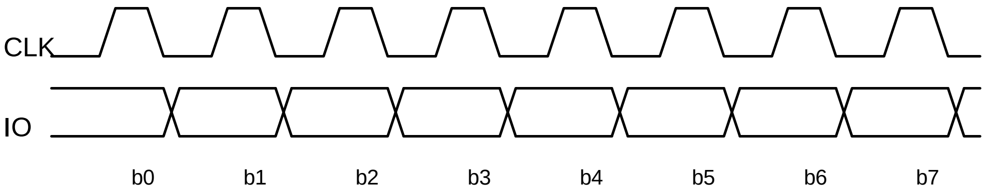
The first bit to be read is the bit b0 of A. The last bit to be read is the bit b7 of A.

At the end of the execution, CLK is set to level 0.

## %WRH_BYTE

This command gives on the IO contact, the A contents.

The sequence is as follows:



- CLK contact set to 0 for 10us.

- CLK contact set to 1 for 10us

The bit to exit on IO is set 5us **before** the rising edge of CLK.

Bit b0 of A is the first bit to exit. Bit b7 of A is the last bit to exit.

At the end of the execution, CLK is set to level 0 and the IO line is set to a high impedance level..

### %RST_PUL

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

### %WAIT_US (TIME)

This command waits for the length of time specified in the TIME parameter.

The waiting time equals TIME * 10us.

### %WAIT_MS (TIME)

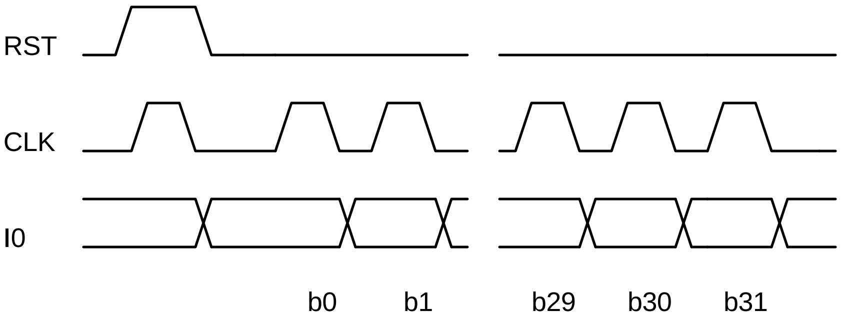
This command waits for the length of time specified in the TIME parameter.

The waiting time equals TIME* 1ms

### %RESET

This command execute the RESET synchronous cards sequence. OROS returns the 32 bits *Answer to Reset* .

Executing the command interrupts the current program.



The RST and CLK signals are forced to level 0 for 10us.

The CLK signal rises 5us after the RST rising edge and remains there for 40us.

The RST signal falls 5us after CLK and remains at 0 for what is left of the sequence.

During the reading of the*Answer to Reset* , the CLK high and low levels remain constant for 10us and the data is read 5us after the rising edge of the CLK.

b0 is the least significant bit of the first byte returned by OROS, b7 being the most significant bit.

b8 is the least significant bit of the second byte returned by OROS, b15 being the most significant bit.

b16 is the least significant bit of the third byte returned by OROS, b23 being the most significant bit.

b24 is the least significant bit of the third byte returned by OROS, 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                   ; Clear internal counter
61           %CLR_CLK                   ;

91           %SET_C4                    ;
EF           MOV A,R7                   ; Select 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               ; Go to read the next byte

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

### Formatted OROS 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 send to the card. |
| DATA IN | not used, empty field. |
| Lout = YYh | number of byte to be read. |
| Lcode = 0Ch | number of bytes of 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
```