oberon service
beyond printing

Oberon SSD/IO (Safety Secure Digital I/O)

API Developers Guide

Manual release	: 1.3D
Manual date	: October 2006

NOTE

- This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
  (1) this device may not cause harmful interference.
  (2) this device must accept any interference received, including interference that may cause undesired operation.
- Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.
- The information contained in this document is subject to change without prior notice
- This document contains proprietary information, which is protected by copyright. All rights reserved. This document cannot be photocopied, reproduced or translated without prior written approval by Oberon Service.
- All other Companies and/or product names are trademarks or registered trademarks of their respective company.
- Oberon Service 2006. All rights reserved.

Index

Addendum A

Addendum B

Oberon Service FIRMWARE LICENSE AGREEMENT

IMPORTANT: Please read this License carefully before using the Firmware contained in the Serial Secure Digital, hereinafter referred to as "SSD" and Serial Secure digital IO referred to as “SSDIO”. The right to use this Firmware is granted only if the Customer agrees with the terms of this License. If you do not agree with the terms of this License, you may return the unused Firmware product for a refund. HOWEVER, THE USE OF THIS FIRMWARE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS.

FIRMWARE LICENSE AND COPYRIGHT

This Firmware is copyrighted and all rights are reserved by Oberon Service srl., hereinafter referred to as "Oberon". In return for the payment of a one-time license fee, which was included in the purchase price of the SSD/SSDIO Firmware, Oberon grants you a non-exclusive right to use the Firmware, subject to the following terms and conditions. No title or ownership of the Firmware is conferred with the License.

1. The Firmware may not be duplicated or copied.

2. The Firmware may not be duplicated, transmitted, transcribed, disassembled, decompiled, decrypted or reverse engineered unless Oberon's prior written consent is either obtained or required by law. Upon request, the user will provide Oberon with reasonably detailed information regarding any disassembly, decompilation, decryption or reverse engineering.

3. Any third party supplier of the Firmware may protect its rights in the Firmware against violation of this License.

4. Oberon reserves the right to terminate this License upon breach. In the event of a termination, all copies of the Firmware shall be returned to Oberon or, with Oberon’s prior written consent, a certificate of destruction of all copies may be provided to Oberon. Any use of the Firmware in violation of the copyright laws constitutes termination of this agreement and prosecution will begin.

5. This License and the Firmware product may not be transferred to a third party.

6. **Itellectual property; limited license to users**

The Firmware product is protected by copyright, trademark, patent, and/or other intellectual property laws, and any unauthorized use of this license may violate such laws and the Terms of Use. Except as expressly provided herein, Oberon and its suppliers do not grant any express or implied rights to use this Firmware. You agree not to copy, republish, frame, download, transmit, modify, assign, distribute, license, sublicense, reverse engineer, or create derivative works based on this Firmware, except as expressly authorized herein.

7. If the Firmware is licensed for use by the U.S. Government, the user agrees that the Firmware has been developed entirely at private expense and is delivered as "Commercial Computer Software" (as defined in DFARS 252.227-7013) or as "Restricted Computer Software" (as defined in FAR 52.227-19).

LIMITED WARRANTY

To the original purchaser, Oberon warrants the SSD/SSDIO on which the Firmware is stored, to be free of defects in materials and faulty workmanship for a period of ninety (90) days from the date the Firmware is delivered. If during this period a defect in this SSD/SSDIO should occur, you may return the SSD/SSDIO with a copy of your receipt or other proof of payment to Oberon or to an authorized Oberon distributor, and Oberon will replace the SSD/SSDIO without charge. Your sole and exclusive remedy in the event of a defect is expressly limited to replacement of the SSD/SSDIO as provided above.

Oberon does not warrant that the functions contained in this Firmware will meet your requirements or that the Firmware operation will be uninterrupted or error free. Information contained in the user manual is subject to change without notice and does not represent a commitment on the part of Oberon.

IN NO EVENT WILL Oberon OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THIS FIRMWARE BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS PROGRAM, OR FOR ANY CLAIM BY ANY OTHER PARTY. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE PROGRAM AND MANUAL IS ASSUMED BY YOU.

This agreement shall be construed, interpreted and governed by the Italian laws. You agree that this is the complete and exclusive statement of this agreement, which supersedes any prior agreement or other communication between us on this subject.

GENERAL DESCRIPTION

The SSD (Safety Secure Digital) and SSDIO (Safety Secure Digital I/O) are storage media compliant with the SD standard Rev 1.01. The SD is a low-cost mass storage device implemented as a removable card, very small and easy to move from one device to another with high performance in terms of read and write capabilities. The SD are commonly used on audio and video consumer electronic devices due to the high speed transfer rate and at their compactness/thickness: 2.1 mm for the normal and 1.4 mm for the thin version.

On SSD and SSDIO Oberon implemented a serial interface in order to connect it to external devices, keeping the SD specifications; the products are designed around Hyperstone's S2-16X flash Memory Controller, while the serial communication is handled by a UART 8051 processor. The products comes with a library of API in order to write and integrate the custom application in a simple way.

❐ Standard features

- Voltage range: basic communication 2.0 - 3.6 Volt
  other commands and memory access 2.7 – 3.6 Volt

- Clock: 0-25 Mhz (40 Mhz using R-C oscillator)
- Data transfer rate to flash memory up to 40 MBytes/s
- NAND type Flash Memory
- Support 32, 64, 128, 256, 512 and 1,2 Gbit NAND flash
- Error Correcting Code capabilities with 6 bytes in a 512 byte sector
- Flash memory power down logic and flash memory write protect control
- Firmware storage in flash memory
- Firmware is loaded into internal memory by the boot ROM (8 Kbyte)
- On-chip ECC unit
- 2 Digital I/O pins
- Serial RS-232 I/O capabilities

❑ Safety features

The Oberon SD implements a security feature that can be useful to protect applications. It simply works as a Password/Answer concept.

❑ Serial and I/O Features

The Oberon SD has the capability to communicate with devices using RS232 or driving/reading sensors using 2 DIGITAL I/O pins.

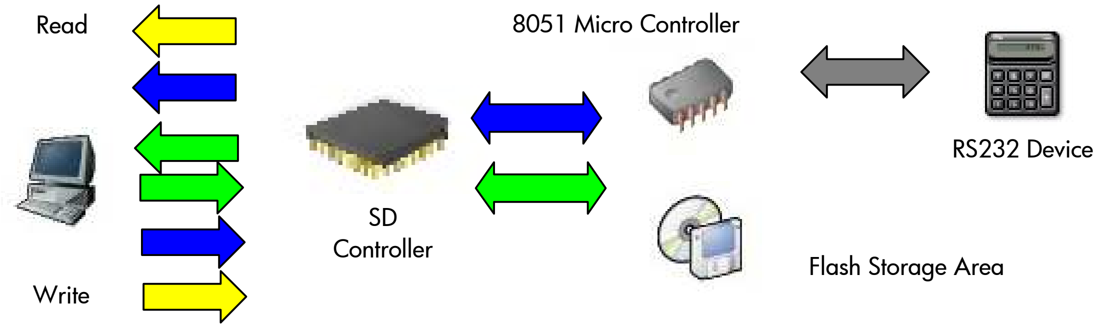
The RS232 communication is configurable from 2400bps up-to 57600bps, while the communications parameters are fixed to N81.

The 2 digital I/O pins are configurable as Input or Output and it is possible to drive them individually. The Safety feature can be implemented as well.

THEORY OF OPERATION

The flash storage sector 0 (first 512byte) is **virtually** used as R/W channel to drive the 8051 communication controller. The firmware running on the SD controller manages data received and transmitted during Read/Write commands issues by the host. A special filter is performing real write and read to storage device and virtual RS232 commands, to perform communications to/from serial devices.

Those commands, useful to communicate with serial devices, are simply performed reading and writing data on Sector 0 of the Oberon SD.



Legenda:

- □ Green : Real Read/Write on storage area
- □ Blue : Virtual Read/Write driving 8051 Micro Controller
- □ Yellow : Virtual Read/Write Security check

The following example shows how to perform those commands in Unix environment using C language:

```
/* Open the SD device */
int f = open("/dev/sd1", O_RDWR| O_DIRECT);

/* Positioning at sector  0  (not necessary at first opening)*/
lseek(f, 0, SEEK_SET);

/* Write to SSD device */
write(f, "Command or data to send to SD....", ...);

/* Repositioning at sector  0 */
lseek(f, 0, SEEK_SET);

/* Read data back from SD device*/
read(f, "data back from SD....", ...);

/* Closing device */
close(f);
```

Part numbers

The SSD and SSD-Io have the following P/N:

Safety Secure Digital 16 MB:	SSD90016
Safety Secure Digital 32 MB:	SSD90032
Safety Secure Digital 64 MB:	SSD90064
Safety Secure Digital I/O 16 MB:	SSD90116
Safety Secure Digital I/O 32MB:	SSD90132
Safety Secure Digital I/O 64 MB:	SSD90164

COMMANDS SUMMARY

Safety Commands

The SD controller recognizes the security commands by the 4 bytes (1111) four ones before the password sent from the host. In this case, when those bytes are received on sector 0, the controller will skip physical writing on the storage device. At the next reading of sector 0 it will answer with the answer key burned on its firmware, in case the password is correct. Otherwise the controller will return to the real flash sector 0 content.

RS232 and Digital I/O Commands

The SD controller recognizes RS232 and Digital I/O (IO0, IO1) commands by the 4 bytes (0000) four zeros in front of the real command. Driving the 8051 is always performed reading and writing the special commands on sector 0 of the storage device. Some of these commands are composed by strings and integer values and it is necessary to pay attention to descriptions.
The available commands are:

Single Command (Strings)	General Description
0000QQ	Query information about 8051 UART status
0000CC	Clear the 8051 UART buffer content
0000RR	Read the 8051 UART buffer content (data sent by the external rs232 device)
Composed Command (Strings + Hex values)	**General Description**
0000W%qS%s	Write data to 8051 UART buffer (data sent to the external rs232 device). %s is the data to send (string) while %q is the data length in hex.
0000W0x01B%b	Set RS232 speed as %b
0000W0x01C%i	Set Digital I/O direction as %i
0000W0x01O%s	Drive Digital I/O pins. %s is the status of pins to drive

RS232 COMMANDS SPECIFIC DESCRIPTION

Query 8051 UART - 0000QQ

This command allows to query information about 8051 UART buffer status and Digital I/O pins conditions. After the host writes this command, it has to read the sector 0 and the SD will answer with a sequence of 4 bytes repeated 64 times (default is 00-FF-02-0F).

The description of these values is:

- Byte 0
  Is the amount of buffer data available for reading. The returned value must be calculated as (Byte 0) + 1.Default at SD power-on is 0x00 (no data to read).

- Byte 1
  Is the amount of buffer data available for transmission. The returned value must be calculated as (Byte 1) + 1. Default at SD power-on is 0xFF (no data to transmit).

- Byte 2
  Is the control byte about the 8051 UART status. Default at SD power-on is 0x02

7	6	5	4	3	2	1	0
X	X	X	X	Tx data Overrun	Rx data Overrun	Tx data Empty	Rx data Avail

Bit 0	1 means new data are available to be read. 0 means no new data received by external device. Use the value of Byte 1 to calculate the quantity of data to read.
Bit 1	1 means the buffer is totally empty. 0 means that data are present on buffer waiting for transmission to the external device.
Bit 2	This bit is raised to 1 when new data are received by external device but the buffer if full.
Bit 3	This bit is raised to 1 when the buffer is full and new data are sent on it by the host.

❐ Byte 3

Is the control byte about the Digital I/O pins. Default at SD power-on is 0x0F

7	6	5	4	3	2	1	0
X	X	IO1 Direction	IO0 Direction	X	X	IO1 Status	IO0 Status

Bit 0	1 = UP (Default); 0 = DOWN
Bit 1	1 = UP (Default); 0 = DOWN
Bit 2	Always UP (1)
Bit 3	Always UP (1)
Bit 4	1 = OUTPUT; 0 = INPUT (Default)
Bit 5	1 = OUTPUT; 0 = INPUT (Default)

Clear 8051 UART - 0000CC

This command will clear the Receive/Transmit buffer. After issuing this command, the host has to perform a read call to check if the buffer was correctly cleared, expecting a buffer filled of character "C" repeated 256 times.

Read 8051 UART - 0000RR

This command reads the 8051's buffer content. After testing the status of the buffer using the query command (0000QQ) and check the number of data available (Byte 0), the host has to read the data from buffer using the read command (0000RR).

Write 8051 UART - 0000W%qS%s

This command writes the quantity %q of data %s to the 8051's buffer.
The write call must be performed in one time linking together command and data.

Example:

```
char *command = "0000W\10SABCDEFGHIK";
write(f, command, strlen(command));
```

where 0000WW is the write command, \10 is the hex value of data quantity and S is the command for 8051 to perform the send.

Configure 8051 UART Baud Rate - 0000W0x01B%b

This command will configure the baud rate %b of the 8051 UART.

Note: The value is kept until the SD is powered.

9600N81 is the factory default baud rate for UART at SD power on.

0000W0x01B0x01	2400N81
0000W0x01B0x02	4800N81
0000W0x01B0x03	9600N81 (default at power-on)
0000W0x01B0x04	19200N81
0000W0x01B0x05	38400N81
0000W0x01B0x06	57600N81

Example:

```
        char *command = “0000W\1B\x06”; /* Setting at 57600,n,8,1  */
        write(f, command, strlen(command));
```

DIGITAL I/O COMMANDS SPECIFIC DESCRIPTION

Configure Digital I/O directions - 0000W0x01C%i

This command is useful to set the direction of the Digital I/O pins named IO0, IO1.
The default configuration of these pins at power on is direction INPUT and status UP(1).

Note : When configuring the pins as OUTPUT, the status will automatically change to DOWN (0).

7	6	5	4	3	2	1	0
X	X	IO1 Direction	IO0 Direction	x	x	x	x

Bit 4	1 = OUTPUT; 0 = INPUT (Default)
Bit 5	1 = OUTPUT; 0 = INPUT (Default)

Example:
This command will set IO0 as INPUT and IO1 as OUTPUT

```
char *command = "0000W\1C\x20";
write(f, command, strlen(command));
```

Set Digital I/O - 0000W0x01O%s

After setting directions of Digital I/O pins with previous function (0000W0x01C%i), with this command it is possible to change the status (UP or DOWN) of pins configured as OUTPUT.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	IO1 Status	IO0 Status

Bit 0	0 = DOWN; 1 = UP
Bit 1	0 = DOWN; 1 = UP

Example:
This command will set IO1 pin as UP and IO0 pin as DOWN.

```
char *commandDrive = "0000W\1C\x02";
write(f, commandDrive, strlen(commandDrive));
```

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

Parameter	Symbol	Value	Unit
Supply voltage	Vcc	-02 to 3.6	V
Input Voltage ON pin	Vin	-0.5 to Vcc+0.5	V
Operating Temperature	Topr	-40 to +85	°C
Storage Temperature	Tstg	-40 to +125	°C

RECOMMENDED DC OPERATING CONDITIONS

Parameter	Symbol	Range	Unit
Supply Voltage	Vcc	2.7/3.6	V
Input High Voltage	Vih	Vcc + 0.3	V
Input Low Voltage	Vih	-0.3 to +0.8	V
Ambient operating temperature	Ta	-40 to 85	°C

DC PARAMETERS

Parameter	Symbol	Conditions	Range	Unit
Output High Voltage	Voh	Data at 1mA	2.4	V
Output Low Voltage	Vol	Data at 4mA	0.4	V
Input Current sleep mode	Icc0	Vcc 3.3V	0.3	mA
Input Current operating at 20 MHz	Icc1	Vcc 3.3V	30	mA
Input Current operating at 40 MHz	Icc2	Vcc 3.3V	50	mA
Input/output Capacitance	Ci/o	Vcc 3.3V	10	pF

AC PARAMETERS

Parameter	Symbol	Conditions	Range	Unit
Clock, Data transfer mode	Fpp	Cl ≤ 100pF	20	MHz
Clock Identification mode	Fod	Cl ≤ 250pF	400	kHz
Clock low time	Twl	Cl ≤ 100pF	10 - 50	ns
Clock high time	Twh	Cl ≤ 100pF	10 – 50	ns
Clock rise time	Ttlh	Cl ≤ 100pF	10 – 50	ns
Clock fall time	Twl	Cl ≤ 100pF	10 - 50	ns

AC PARAMETERS (continue)

Parameter	Symbol	Conditions	Range	Unit
CMD, DAT input setup time	Tisu	Cl ≤ 25pF	5	ns
CMD, DAT input hold time	Tih	Cl ≤ 25pF	5	ns
CMD, DAT output delay (data transfer)	Toldy	Cl ≤ 25pF	0 - 14	ns
CMD, DAT output delay (identification)	Toldy	Cl ≤ 25pF	0 – 50	ns

ELECTRICAL STATIC DISCHARGE (ESD) requirements

The ESD parameters are the following:

Contact Pads

+/- 4kV, Human body model according to the IEC61000-4-2 specifications

Non contact Pads area

+/- 8 kV (coupling plane discharge)

+/- 15 kV (air discharge)

DENSITY AND CMOS NAND E2PROM

The SSD90xx supports the following NAND flash devices TSOP 48 pins mounting.

128 MBIT

PRODUCER	P/N	Density	Organisation*	Vcc
TOSHIBA	TC58DVM72A1FT00	16M x 8 bits	528 x 32 x 1024	2.7V-3.6V
SAMSUNG	K9F2808U0C-PCB0	16M x 8 bits	528 x 32 x 1024	2.7V-3.6V
ST	NAND128W3A-2BN6	16M x 8 bits	528 x 32 x 1024	2.7V-3.6V
HYNIX	HY27UF08122M-TPC	16M x 8 bits	528 x 32 x 1024	2.7V-3.6V

256 MBIT

PRODUCER	P/N	Density	Organisation*	Vcc
TOSHIBA	TC58DVM82A1FT00	32M x 8 bits	528 x 32 x 2048	2.7V-3.6V
SAMSUNG	K9F5608U0C-PCB0	32M x 8 bits	528 x 32 x 2048	2.7V-3.6V
ST	NAND256W3A-2BN6	32M x 8 bits	528 x 32 x 2048	2.7V-3.6V
HYNIX	HY27UF08562M-TPC	32M x 8 bits	528 x 32 x 2048	2.7V-3.6V

512 MBIT

PRODUCER	P/N	Density	Organisation*	Vcc
TOSHIBA	TC58DVM92A1FT00	64M x 8 bits	528 x 32 x 4096	2.7V-3.6V
SAMSUNG	K9F1208U0A-PCB0	64M x 8 bits	528 x 32 x 4096	2.7V-3.6V
SAMSUNG	K9F1208U0M-PCB0	64M x 8 bits	528 x 32 x 4096	2.7V-3.6V
ST	NAND512W3A-2BN6	64M x 8 bits	528 x 32 x 4096	2.7V-3.6V
HYNIX	HY27UF08122M-TPC	64M x 8 bits	528 x 32 x 4096	2.7V-3.6V

1 GBIT

PRODUCER	P/N	Density	Organisation*	Vcc
SAMSUNG	K9F1G08U0A-PCB0	128M x 8 bits	2112 x 64 x 1024	2.7V-3.6V
HYNIX	HY27UF081G2M-TPC	128M x 8 bits	2112 x 64 x 1024	2.7V-3.6V

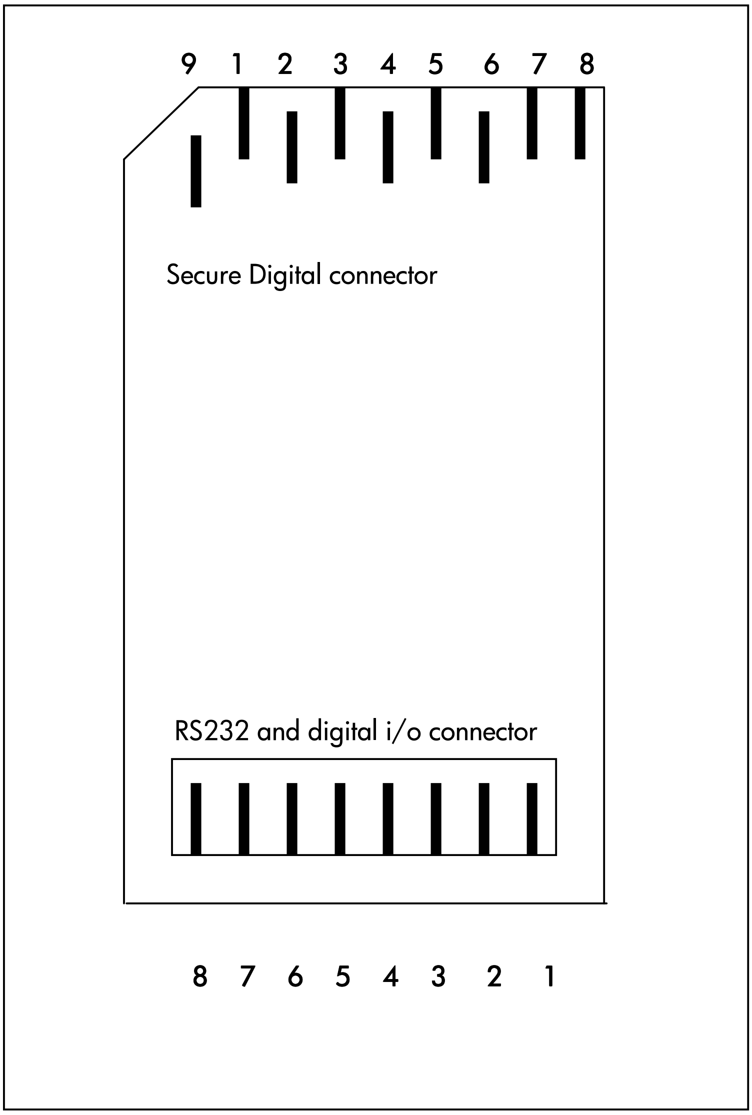
2 GBIT

PRODUCER	P/N	Density	Organisation*	Vcc
SAMSUNG	K9F2G08U0A-PCB0	256M x 8 bits	2112 x 64 x 2048	2.7V-3.6V
HYNIX	HY27UF081G2M-TPC	256M x 8 bits	2112 x 64 x 2048	2.7V-3.6V

- Bytes x pages x blocks

PIN SPECIFICATION

The SSD90xx SDS card has two connectors; the frontal connector is for SD slot insertion and the rear Molex connector is used to link external serial devices. The specifications are the following:



SSD90xx SDS pin specification

Secure Digital connector

Description	Pin #	Name	Type	Note
Card detect/Data Line Bit 3	1	CD/DAT 3	I/O/PP	I/O and Push Pull
Command/response	2	CMD	PP	Push Pull
Supply voltage ground	3	Vss1	S	Power Supply
Supply voltage	4	Vdd	S	Power Supply
Clock	5	CLK	I	Input
Supply voltage ground	6	Vss2	S	Power Supply
Data Line Bit 0	7	DAT 0	I/O/PP	I/O and Push Pull
Data Line Bit 1	8	DAT 1	I/O/PP	I/O and Push Pull
Data Line Bit 2	9	DAT 2	I/O/PP	I/O and Push Pull

Serial RS-232 and digital I/O connector

Description	Pin #	Name	Type	Note
Power supply	1	VCC	S	Power Supply
Digital I/O #1	2	DIO 1	PP	Push Pull
Digital I/O #2	3	DIO 2	PP	Push Pull
Not used	4	N/A	N/A	N/A
Not used	5	N/A	N/A	N/A
Transmit data	6	TXD 232	I/O	Input/Output
Receive data	7	RXD 232	I/O	Input/Output
Ground	8	GND	S	Power Supply

Serial RS-232 and digital I/O electrical specification

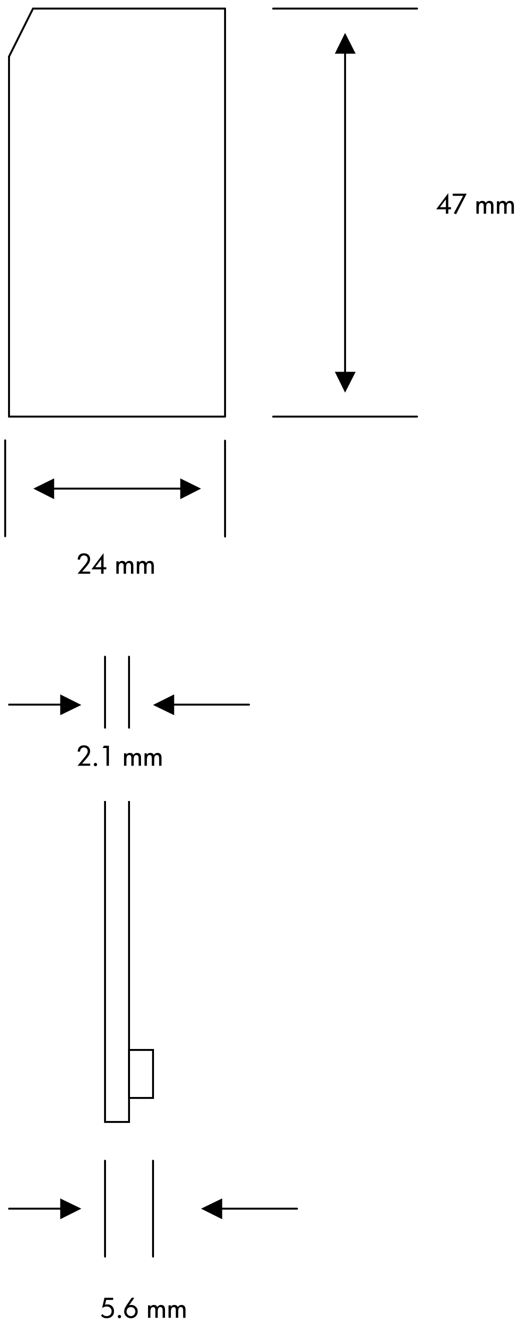
Description	Pin #	Range	Unit
Digital I/O #1	2	0 + 3.3	Volt
Digital I/O #2	3	0 + 3.3	Volt
Transmit data	6	-5 + 5	Volt
Receive data	7	-5 + 5	Volt

Serial RS-232 and digital I/O connector

The connector used is a 8 pin male connector, Molex 53398-0890 that allow the board to wire connection with female 8 pin Molex 51021-0800.

MECHANICAL SPECIFICATION

The SSD90xx follows SD mechanical specifications for the X-axis and for the thickness; the Y-axis is longer due to the RS-232 connector placed on the end-edge. The measurements are the following:



ADDENDUM A

Digital i/o commands with external Pull-up (Driving open drain)

Configure Digital I/O status- 0000W0x01C%i

This command is useful to set the output Digital I/O pins named IO0, IO1.
The default configuration of these pins at power on is status UP

7	6	5	4	3	2	1	0
X	X	IO1 status	IO0 status	X	X	X	X

Bit 4 1 = DOWN; 0 = UP (Default)
Bit 5 1 = DOWN; 0 = UP (Default)

Example:
This command will set IO1 pin as DOWN and IO0 pin as UP

7	6	5	4	3	2	1	0
X	X	IO1 status = 1	IO0 status = 0	X	X	X	X

```
char *command = “0000W\1C\x20”;
write(f, command, strlen(command));
```

ADDENDUM B

SSD and SSDIO card Identification command

The Developers should be careful in writing sector 0 of standard SD (not Oberon SSD and SSDIO) since writing this sector will turn the device unusable by the printers.
To avoid this mistake, an application should verify if SSDIO or SSD card from Oberon is inserted in the target slot.
The ATA controller recognizes the Identification commands by the 4 digits 1111 in front of the key ABCD and no real write will be performed on sector 0.
If the remaining 4 bytes (“ABCD”) match with the code burned on the controller, the read() call will return a 4 bytes (“OKOK”) answer, burned on the controller as well; otherwise the controller returns the first 4 bytes present on sector 0.

The sample code below shows how to restore the value of sector 0. By sending the sequence of char "1111ABCD" to the SD, the SSDIO or SSD must reply with the string "OKOK"; if it returns some other value, the following code restores the original value of sector 0, and the SD will work properly.

```
class RicohSD {

        private static final String password = "ABCD";
        private static final String device = "/dev/rsdc1c";

        public static void main(String[] args)  throws IOException{

                System.out.println("Hello SD Ricoh!");

                /* Check the original SSD  */
                SSDcheck _SSDcheck = new SSDcheck();
                //save the sector 0 of a SD for avoid the broker of not Ricoh SD
                _SSDcheck.saveSector();
                if(!_SSDcheck.checkOriginal(device,password)){
                        //restore the content of sector 0, the SD isn't a SDD Ricoh
                        _SSDcheck.restoreSector();
                }else{
                        // to do somethig, the SD is an original SSD Ricoh
                        ;
                }
        }
}
```

source 1

```
class SSDcheck{

        private final String answer = "OKOK";
        private String device = null;
        private byte[] saveBuffer; // buffer where to store the original 512 byte of the sector 0

        String strSecurity = null;
        boolean isSecurity = false;

        public SSDcheck(){
        }

        /* Check Security */
        public boolean checkOriginal(String device, String strPassword){
                boolean isSecurity = false;
                this.device = device;
                try{

                        strSecurity = sendPassword(strPassword);
                        if(strSecurity.equals(this.answer))
                                isSecurity = true;
                        else
                                isSecurity = false;
                }catch (Exception e){
                        isSecurity = false;
                }
                return isSecurity;
        }

        //send the password "ABCD" to the SD, if the SD is RICOH SD the method return the string "OKOK"
        public String sendPassword(String data) throws IOException {

                try{
                        String str = "";
                        byte[] dateString = data.getBytes();
                        FileOutputStream fos = new FileOutputStream (device);
                        byte[] date = new byte[512];
                        date[0] = '1';
                        date[1] = '1';
                        date[2] = '1';
                        date[3] = '1';
                        for (int i = 0; i < dateString.length ; i++ ){
                                date[i+4] = dateString[i];
                        }
                        fos.write(date);
                        fos.close();

                        FileInputStream fis = new FileInputStream (device);
                        byte[] readDate = new byte[512];
                        fis.read(readDate);
            for(int i= 0; i<4; i++){
                str = str + (char)readDate[i];
            }
            fis.close();

            return str;
                }catch (IOException e){
                        e.printStackTrace();
                        throw new IOException("SerialSSD :: checkSecurity - > "+e);
                }
        }
        public void saveSector() throws IOException {
                        FileInputStream fis = new FileInputStream (device);
                        saveBuffer = new byte[512];
                        fis.read(saveBuffer);
            fis.close();
        }
        public void restoreSector() throws IOException {
                FileOutputStream fos = new FileOutputStream (device);
                fos.write(saveBuffer);
                fos.close();
        }
}
```