bool initDLL(void);
bool TPrintLibIO(char* cpstrPRNname, char* cpstrCMDtype, char* cpstrCommand, int* ipCommand, char* cpstrCom2Result, int* ipCom2Result);
bool exitDLL(void);

Command Description	Selected Printer (cpstrPRNname)	Command Type (cpstrCMDtype)	Command String (cpstrCommand)	Command Value (ipCommand)	Result String Maximum 1024 bytes	Result Value (ipCom2Result)	Comments:
Enumerate all installed printers	“”	PRN_READ	“”	0 = get the amount of all installed printers 1 = first installed printer 2 = second installed printer 3 = … and so on	“” Printer name Printer name …	n = amount of all installed printers 0 0 0	
	Printer name	GET_PRN_INFO	“”	0 = Get Device ID 1 = Original Driver Name 2 = is printer supported? 3 = is printer connected?	Device ID Info string Org. Driver Name “” “USB” if a USB printer	0 0 true = Yes, false = No true = Yes, false = No	
	“”	DLL_VERSION	“”	0	Version string	0	
	Printer name	FW_VERSION	“”	0	Version as String	Version as Integer	Do not use when PRN busy!
	Printer name	DRIVER_VERSION	“”	0	Version as String	0	
	Printer name	SERIAL_NUMBER	“”	0	Serial number as string	0	
	Printer name	READ_LEFT_POS	“”	0	Left Position as String	Left Position as integer	Do not use when PRN busy!
	Printer name	READ_TOP_POS	“”	0	Top Position as String	Top Position as integer	Do not use when PRN busy!
	Printer name	READ_INSERT_POS	“”	0	Insert Position as String	Insert Position as integer	Do not use when PRN busy!
	Printer name	SET_LEFT_TOP_POS (set both pos. together)	Left Position As String	0	Top Position as String (Used as Input to DLL)	0	Do not use when PRN busy!
	Printer name	SET_INSERT_POS	Insert Pos. as String	0	“”	0	Do not use when PRN busy!
	Printer name	CLEAN	“”	0 = light cleaning 1 = medium cleaning 2 = strong cleaning	“”	0	Do not use when PRN busy!
	Printer name	MOVE_OUT MOVE_IN	Timeout value as string in seconds: “8” = 8 seconds	0	“”	0 = DLL replies immediately 1 = DLL waits until the movement is finished	Do not use when PRN busy!
	Printer name	READ_PRN_COUNTER	“”	0	Print counter as string	Print counter as integer	Do not use when PRN busy!
	Printer name	MANUFACTURING_CODE	“”	0	Manufacturing code	0	Do not use when PRN busy!
	Printer name	PRINTER_STATUS	“”	0	“”	Details see appendix A	
	Printer name	CARTRIDGE_STATUS	“”	0	Details see appendix B	0	
	Printer name	POWER_ON	“”	0	“”	0	Printer power ON
	Printer name	POWER_OFF	“”	0	“”	0	Printer power OFF

Command Description	Selected Printer (cpstrPRNname)	Command Type (cpstrCMDtype)	Command String (cpstrCommand)	Command Value (ipCommand)	Result String Maximum 1024 bytes	Result Value (ipCom2Result)	Comments:
new in 2.2.4.43	Printer name	TRAY_STATUS	“”	0	“”	0 = depends on return value false = status unknown true = Printer BUSY 1 = Tray IN (empty) 2 = Tray OUT 3 = reserved 4 = Tray IN (media present)	Can be read also via the DeviceID now: Last value of substring S: is showing this status values as well.

Implementation: (example)

```
//definition in the header file of the external application:
#define STRINGLENGTH 1024
char* cpstrPRNname;
char* cpstrCMDtype;
char* cpstrCommand;
int* ipCommand;
char* cpstrCom2Result;
int* ipCom2Result;



// include functions:
#include "TPrintLIB.h"
#pragma comment(lib,"TPrintLIB.lib")


// initalise TPrintLIB class
CTPrintLIB *TPL = new CTPrintLIB;


//allocating memory during initialisation of the external application:
cpstrPRNname = (char*)malloc (STRINGLENGTH);
memset (cpstrPRNname, 0, STRINGLENGTH);
cpstrCMDtype = (char*)malloc (STRINGLENGTH);
memset (cpstrCMDtype, 0, STRINGLENGTH);
cpstrCommand = (char*)malloc (STRINGLENGTH);
memset (cpstrCommand, 0, STRINGLENGTH);
ipCommand = (int*)malloc (10);
cpstrCom2Result = (char*)malloc (STRINGLENGTH);
memset (cpstrCom2Result, 0, STRINGLENGTH);
ipCom2Result = (int*)malloc (10);


//load valid content to the variables
lstrcpy(cpstrPRNname,"");
lstrcpy(cpstrCMDtype,"DLL_VERSION");
lstrcpy(cpstrCommand,"");
*ipCommand = 0;
lstrcpy(cpstrCom2Result,"");
*ipCom2Result = 0;


// call the functions
if(TPL)
       bool bRet = TPL->TPrintLibIO(cpstrPRNname,cpstrCMDtype, cpstrCommand, ipCommand, cpstrCom2Result, ipCom2Result);
```

Examples:

```
Enumerate all installed printers: bool bRet = TprintLibIO(“”,”PRN_READ “,””,0,””,0); => ipCom2Result = 2 (2 installed printers found)

Get first installed printer:    bool bRet = TprintLibIO(“”,”PRN_READ “,””,1,””,0);  =>  cpstrCom2Result = “ODP 200 renamed”
Get 2nd installed printer:      bool bRet = TprintLibIO(“”,”PRN_READ “,””,2,””,0);  =>  cpstrCom2Result = “ODP 500 Series”

Read printer information        bool bRet = TprintLibIO(“ODP 200 renamed”,”GET_PRN_INFO”,””,1,””,0);=> cpstrCom2Result = Orig. Driver name
                                                                                                       = “ODP 200 series”
Read printer information        bool bRet = TprintLibIO(“ODP 500 Series”,”GET_PRN_INFO”,””,1,””,0); => cpstrCom2Result = Orig. Driver name
                                                                                                       = “ODP 500 Series”

Read printer information        bool bRet = TprintLibIO(“ODP 200 renamed”,”GET_PRN_INFO”,””,2,””,0); =>  ipCom2Result  = true/false
Read printer information        bool bRet = TprintLibIO(“ODP 500 Series”,”GET_PRN_INFO”,””,2,””,0);  =>  ipCom2Result  = true/false

Read printer information        bool bRet = TprintLibIO(“ODP 200 renamed”,”GET_PRN_INFO”,””,3,””,0); =>  ipCom2Result  = true/false
Read printer information        bool bRet = TprintLibIO(“ODP 500 Series “,”GET_PRN_INFO”,””,3,””,0); =>  ipCom2Result  = true/false

Read Device ID                  bool bRet = TprintLibIO(“ODP 200 renamed”,”GET_DEV_ID”,””,0,””,0);   =>  cpstrCom2Result = Device ID
Read Device ID                  bool bRet = TprintLibIO(“ODP 500 Series”,”GET_DEV_ID”,””,0,””,0);    =>  cpstrCom2Result = Device ID
```

Note:

The calling application has to allocate the memory for the parameter values, also the result values “cpstrCom2Result” and “ipCom2Result”.
The string length of the allocated strings has to be 1024 bytes long to make sure that all possible data can be transferred.

The Original Driver Name is the name of the installed driver.
This name is always the same e.g. “ODP 500 Series”), independent how the driver is named in the printer settings, e.g. “ODP 500 renamed driver name”.
Therefore the original driver name can be used to identify the printer, independent how the driver is renamed after installation!

The TPrintLIB.DLL is creating some LOG files in the hidden directory: “C:\Documents and Settings\All Users\Application Data\”
These files can be used to see if the DLL is working properly, or, if a problem occurs, to figure out the reason.

The DLL cannot access the printer when the printer is in error state, means when the “Resume” LED or “Ink” LED is blinking.
The reason for a blinking “Ink” LED is a problem with the cartridges.
The reason for a blinking “Resume” LED is, when the printer has received printing data, but no media is available to print on. (Out of Paper)

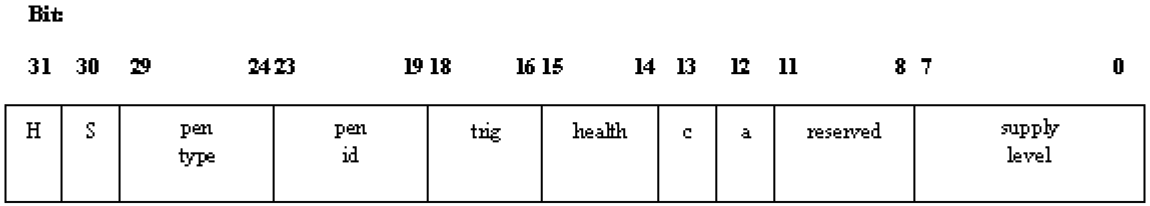
Commands which are marked in red should not be used while the printer is busy, because they can interrupt the current running command and the printer may hang!
Check the status first with the command “PRINTER_STATUS” and apply the red commands only when the result is 0 (zero) which means the printer is in idle state.

Appendix:

A: PRINTER_STATUS

Dec. Value	Hex. Value		Meaning
0	**0**	**Ready / Idle**	**waiting for work**
1	**1**	**Initializing / busy**	**printing or some other activity**
2	**2**	**io printing**	**Printing I/O received job**
3	**3**	**off**	**Turning off**
4	**4**	**reports printing**	**Printing internal print job**
5	**5**	**canceling**	**Cancelling a job**
6	**6**	**io stall**	**Form feed needed, incomplete job**
7	**7**	**dry time wait**	**Dry time wait**
8	**8**	**pen change**	**Cover open and cartridge change occurring**
9	**9**	**out of print media**	**No passport inserted, or not fully inserted (Out of paper)**
10	**0A**	**banner ejected needed**	**Banner eject needed**
11	**0B**	**banner mismatch**	**Banner mismatch**
12	**0C**	**photo mismatch**	Photo tray mismatch **(N/A)**
13	**0D**	**duplex mismatch**	**Duplex mismatch**
14	**0E**	**media jam**	**Paper jam**
15	**0F**	**carriage (printhaed) stall**	**Carriage stall**
16	**10**	**print media (cage) stall**	**Paper stall**
17	**11**	**pen / head failure**	**Pen failure**
18	**12**	**hard error**	**Severe hard error**
19	**13**	**powering down**	**Powering down**
20	**14**	**fp test**	**Manufacturing front panel test**
21	**15**	**hyde missing**	**Clean out tray is missing**
22	**16**	**---**	**Not used as valid value**
23	**17**	**media size mismatch**	**media size mismatch**

B: CARTRIDGE STATUS



Bit:

31	30	29 24	23 19	18 16	15 14	13	12	11 8	7 0
H	S	pen type	pen id	trig	health	c	a	reserved	supply level

The cartridge status contains the information of the two cartridge chutes, e.g. “C1840064C2500020”
This example shows the status of a Black cartridge in the left eight values, the right eight values are showing a Color pen.
Here is a sample value of a Photo cartridge: 3580062

- **Cartridge Status (8 nibbles per cartridge):** The next 8 nibbles comprise a field of 32 bits with various subfields. This 8-nibble/32-bit field describes a pen (for example, a print head), ink supply, or both. Additional pens or supplies require additional pen/supply information fields.

For the ODP 200, there are only two pen chutes, and the ink supply is built into the pens; therefore, there are two pen/supply info fields defined and both the print head and ink supply bits are set to 1. The first chute can potentially take two different types of cartridges: Black/K) or Photo/KCM.
The other chute can take CMY pens only.

Bit 31 (1 bit) 1 if these fields describe a print head, always 1

Bit 30 (1 bit) 1 if these fields describe an ink supply, always 1

Bits 29 .. 24 (6 bits) describes the pen/supply type:
0 = none
1 = Black cartridge (also known as K)
2 = Color cartridge CMY (C=cyan, M=Magenta, Y=Yellow)
3 = Photo cartridge KCM (K=black, C= Magenta, M= Magenta)

Bits 23..19 (5 bits) describes the pen id:

0 = none
9 = Black cartridge (pen/ink combo; k)
10 = Color cartridge (pen/ink combo; cmy)
11 = Photo cartridge (pen/ink combo; kcm)

Bits 18 ..16 (3 bits) ink level trigger:

0 = sufficient ink
5 = may be low on ink
6 = probably out of ink
7 = almost definitely out of ink

Bits 15 ..14 (2 bits) pen health:

0 = OK
1 = Misinstalled
2 = Pen incorrect
3 = Pen fail

Bit 13 (1 bit) pen change:

0 = always

Bit 12 (1 bit) alignment needed:

0 = always

Bit 11 (1 bit) printer is confused about this pen:

1 = confused
0 = NOT confused

Bit 10 .. 8 (3 bits) reserved for future use

Bits 7 .. 0 (8 bits) ink supply level - this low order byte is interpreted as a value in the range 0 to 100, to indicate the percentage of ink remaining.

For example, there are two separate pen/supply info fields. Status value for example: **C1840064 C2500020**

Black pen information: **C1840064** (1100 0001 0100 1000 0000 0000 0110 0100)

```
11        = is a print head and an ink supply
00 0001   = K black pen
0100 1    = Black
000       = ink level ok
00        = pen health ok
0         = pen not changed
0         = alignment is NOT needed
0         = printer NOT confused about pen
000       = reserved
0110 0100 = 64H = 100% ink left
```

Color pen information: **C2500020** (1100 0010 0101 0000 0000 0010 0000)

```
11        = is a print head and an ink supply
00 0010   = CMY color pen
0101 0    = Color
000       = ink level ok
00        = pen health ok
0         = pen not changed
0         = alignment is NOT needed
0         = printer NOT confused about pen
000       = reserved
0010 0000 = 20H = 32% ink left
```

Photo cartridge information is similar: **C3580062** (1100 0011 0101 1000 0000 0010 0000)

```
11        = is a print head and an ink supply
00 0011   = KCM photo pen
0101 1    = Photo
000       = ink level ok
00        = pen health ok
0         = pen not changed
0         = alignment is NOT needed
0         = printer NOT confused about pen
000       = reserved
0110 0010 = 62H = 98% ink left
```

Change log:

2.2.0.22:	Bug fixed in exported function “exitDLL()” Exported functions optimised for new D3000 with two motors.
2.2.0.23:	New function “PRINT_BLOCKING” integrated
2.2.0.24:	Adding support for printer C 1000
2.2.0.25:	Bug fix for usage under Windows 2000
2.2.0.26:	Adding support for printer C 4000.
2.2.0.27:	Adding support for reading the manufacture code for D 3000 New maximum value for SET_INSERT_POS and MOVE_RELATIVE: 17500 instead of 17250 for C 3000
2.2.0.28	Bug fix for printer C 1000
2.2.0.29	New functions included to support the new “Universal Printer Feeder”, UDF for the C 1000
2.2.0.30	Adding new functions “Keep Paper after print” for C3000 and D3000 Adding new functions “Use color cartridge only” for C3000 and adding support for reading the manufacture code for C3000 Command “CLOSE_AFTER_PRINT” replaced by the command “KEEP_AFTER_PRINT” for C3000 and D3000 (command is working now as “KEEP_AFTER_PRINT”) Command “Dry_TIME” modified. 15 steps of 1.0 seconds, instead of 7 steps of 1.5 seconds. Reading back setting modified. For C3000
2.2.0.31:	Adding support for printer ODP 200 and CDP 5000.
2.2.0.32:	Adding support for printer C 5000.
2.2.0.33:	C 5000: Support for FW 113 and bug fix.
2.2.0.34:	C 3000: Fix of bug in MOVE_REL command(s).
2.2.3.35:	TPrintLIB released now for Windows 7 32bit. D3000: Modified to read the Windows 7 driver version.
2.2.3.36:	All other printers: Modified to read the Windows 7 driver version.
2.2.3.37:	Added the support of the new calibration file which compensates the 1.65mm offset against the old XP driver.
2.2.3.38:	TPrintLIB released now for Windows 7 64bit.
2.2.4.39:	Command added to switch the camera LEDs inside the C3000 printer by software.
2.2.4.40:	Remove bug in function MOVE_OUT for D 3000 and ODP 500. Add Power ON/OFF function for ODP 200
2.2.4.41:	Bug removed in D 3000 Terminal which hangs after a while
2.2.4.42:	Adding "INSERT position" support for ODP200. Adding support for printer C 3100.
2.2.4.43:	Tray Status of the printer is now part of the DeviceID and can be read with the command “TRAY_STATUS”