CY Status API (CyStat.dll, CyStat64.dll)

Command Reference

Ver. 1.31

CITIZEN SYSTEMS JAPAN CO.,LTD.

September 9, 2013

Revision History Chart		First Edition: May 20, 2011			Page 1/1	
Revised Item	Type of Revision	Revision No.	Document No.	Revision Date	Approved	Designer
P4 P5, P13, P23	･Addition of Windows 8 to supported OS. ･Clerical error correction of API function name (acquisition of counter value) ･The case of a 2inch cut is added to a count-up operation table. ･The 2inch cut mode is added to a cutter control command. (Change of only a document)	1.31	1.31	2013/09/09		

Table of Contents

Introduction

Application Scope

This document is a command reference manual for the CY Printer Status API.

Compatible Operating Systems and Operating Environment

This API runs on Windows XP, Windows Vista, Windows 7, and Windows 8.

API function

■Notes

- When using C language, include " CyStat.dll " for a 32 bit OS, and " CyStat64.dll " for a 64 bit OS.
- When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.
- Both CyStat.vb are defined such that they can be used and switched between 32⇔64 Bit OS environments. For the 32 bit OS, make the first line of the CyStat.vb " #Const x64=False ", and for the 64 bit OS make the first line "#Const x64=True".

	API function name		Point of notice
DLL Initialize	CvInitialize()	PortInitialize()	
Get Printer Version Information	CvGetVersion()	GetFirmwVersion()	
Get Printer Sensor Information	CvGetSensorInfo()	GetSensorInfo()	

Get Printer Resolution	CvGetResolutionH()
	CvGetResolutionV()
		GetResolutionH()
		GetResolutionV()	

Get Printer Media Code	CvGetMedia()	GetMedia()	
Get Printer Status	CvGetStatus()	GetStatus()	

Get Printer Counter Value	CvGetCounterL()
	CvGetCounterA()
	CvGetCounterB()
		GetCounterL()
		GetCounterA()
		GetCounterB()
		GetCounterP()
		GetCounterMatte()
		GetCounterM()	

Clear Printer Counter Value	CvSetClearCounterA()
	CvSetClearCounterB()
		SetClearCounterA()
		SetClearCounterB()
		SetClearCounterM()	

Set Printer Counter Value(P)		SetCounterP()	
Get the Number of Free Image Buffers	CvGetFreeBuffer()	GetFreeBuffer()	
Get Remaining Print Quantity	CvGetPQTY()	GetPQTY()	

Get the Media Counter of Remaining Sheets	CvGetMediaCounter()	GetMediaCounter()
		GetMediaCounterR()	

Get Media Color Offset Value of the Lot	CvGetMediaColorOffset()	GetMediaColorOffset()	
Get Media Lot Information	CvGetMediaLotNo()	GetMediaLotNo()	
Get Printer Serial Number	CvGetSerialNo()	GetSerialNo()	
Set Firmware Update Mode	CvSetFirmwUpdateMode()	SetFirmwUpdateMode()	
Write Firmware Data	CvSetFirmwDataWrite ()	SetFirmwDataWrite()	VB.Net Wrapper function
Clear Color Data	CvSetColorDataClear()	SetColorDataClear()	
Write Color Data	CvSetColorDataWrite()	SetColorDataWrite()	VB.Net Wrapper function
Set Color Data Version	CvSetColorDataVersion()	SetColorDataVersion()	VB.Net Wrapper function
Get Color Data Version	CvGetColorDataVersion()	GetColorDataVersion()	
Get Color Data Checksum	CvGetColorDataChecksum()	GetColorDataChecksum()	
Cutter Control Command		SetCutterMode()	
Get Media ID setting value		GetMediaIdSetInfo()	
Get media class		GetRfidMediaClass()	
Get RF-ID reserve data		GetRfidReserveData()	
Get initial media count		GetInitialMediaCount()	
Common Set Command	CvSetCommand()	SetCommand()	VB.Net Wrapper function
Common Get Command	CvGetCommandEX()	GetCommandEX()	VB.Net Wrapper function

DLL Initialize

[Format]	long PortInitialize(LPWSTR p);
	long CvInitialize(LPWSTR p);	

[Parameter]	p:	Pointer to the port name WSTR (2 byte Unicode)
[Return]	True:	Port number
	False:	-1

[Note] Initializes API and returns the port number. If more than one port are used, please get each port number by generating the command repeatedly.

[Sample Coding]

< Visual C >
```
long CY;
if(( CY = PortInitialize( L"USB001" )) < 0 ){
            // error
}
```

< VB.NET >
```
Dim CY As Integer
CY = PortInitialize("USB001");
If CY < 0 Then GoTo Error
```

Get Printer Version Information

[Format]	long GetFirmwVersion(long lPortNum, LPSTR p);
	long CvGetVersion(long lPortNum, LPSTR p);	

[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received in buffer p
	False:	-1

[Note] Receives the printer version information in the buffer.

[Sample Coding]

< Visual C >
```
char rbuf[256];
if( GetFirmwVersion( CY, (LPSTR)rbuf ) > 0 ){
            // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
l = GetFirmwVersion(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

[Version Information]

CY *.**	Firmware version information

Get Printer Sensor Information

[Format]

```
long GetSensorInfo( long lPortNum, LPSTR p );
long CvGetSensorInfo( long lPortNum, LPSTR p );
```

[Parameter]

lPortNum:	Port number
p:	Pointer to the receiving buffer

[Return]

True:	The number of characters received in buffer p
False:	-1

[Note] Receives value of each sensor (through AD converter) in the buffer.

[Sample Coding]

< Visual C >

```
char rbuf[256];
if( GetSensorInfo( CY, (LPSTR)rbuf ) > 0 ){
        // Next process
}
```

< VB.NET >

```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetSensorInfo(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

[Sensor Information for]

HDT-***;	Head temperature	
MDT-***;	Media temperature	
PMK-***;	Paper mark	
RML-***;	Ribbon mark left	(Unused,"000" always returns to value)
RMC-***;	Ribbon mark center	
RMR-***;	Ribbon mark right	
PSZ-***;	Paper size	(Unused,"000" always returns to value)
PNT-***;	Paper notch	(Unused,"000" always returns to value)
PJM-***;	Paper jam	(Unused,"000" always returns to value)
PED-***;	Paper end	
PET-***;	Paper empty	(Unused,"000" always returns to value)
HDV-***;	Head voltage	
HMD-***;	Humidity	

Get Printer Resolution

[Format]	long GetResolutionH(long lPortNum);
	long CvGetResolutionH(long lPortNum);
	long GetResolutionV(long lPortNum);
	long CvGetResolutionV(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True: False:	Horizontal, or Vertical resolution (dpi) -1
[Note]	GetResolutionH(),CvGetResolutionH() returns Horizontal resolution (dpi). GetResolutionV(),CvGetResolutionV() returns Vertical resolution (dpi).	

[Sample Coding]

< Visual C >

```
long rh;
if(( rh = GetResolutionH( CY )) >= 0 ){
        // Returns Horizontal resolution to rh
}
```

< VB.NET >

```
Dim i As Integer
i = GetResolutionH(CY)
If i >= 0 Then Text1.Text = Str(i) Else Text1.Text = "ERROR!"
```

Get Printer Media Code

[Format]	long GetMedia(long lPortNum, LPSTR p);
	long CvGetMedia(long lPortNum, LPSTR p);	

[Parameter]	lPortNum: p:	Port number Pointer to the receiving buffer
[Return]	True: False:	The number of characters received by buffer p -1
[Note]	Receives media code in the buffer.	

[Sample Coding]

< Visual C >

```
char rbuf[256];
if( GetMedia( CY, (LPSTR)rbuf ) > 0 ){
            // Next process
}
```

< VB.NET >

```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetMedia(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

[Media code]

Media code is defined by 5 decimal bytes. The return value for the Get Media Code command is referred to as the following 5 bytes (ASCII numeric).

4th byte		3rd, 2nd byte		1st byte	
(0n000)	Paper Type	(00nn0)	Paper Size	(0000n)	Positioning Mark
00000	Normal Paper	00200	5x3.5 (L)	00000	No Mark
01000	Sticker	00210	5x7 (2L)	00001	With Mark (Back Print)
		00300	6x4 (PC)		
		00310	6x8 (A5)		

Example

Paper Size	Paper Type	(Size: Width x Height)	Media Code
5x3.5 (L)	Normal paper	(127.0 x 89.0 mm)	00200
6x4 (PC)	Normal paper	(152.0 x 102.0 mm)	00301
5x7 (2L)	Normal paper	(127.0 x 178.0 mm)	00210
6x8 (A5)	Normal paper	(152.0 x 203.0 mm)	00310

Get Printer Status

[Format]	DWORD GetStatus(long lPortNum);	
	DWORD CvGetStatus(long lPortNum);	
[Parameter]	lPortNum:	Port number
[Return]	True:	Status
	False:	STATUS_ERROR

[Note] This returns the printer status.
Bit position of the status is defined in CyStat.h by macro.(CyStat.bas for Visual Basic)
Meanings of the symbols are as follows:

GROUP_USUALLY	[Usual Operation] Group identification bit
GROUP_SETTING	[Setting Error] Group identification bit
GROUP_HARDWARE	[Hardware Error] Group identification bit
GROUP_SYSTEM	[System Error] Group identification bit
GROUP_FLSHPROG	[Rewriting Mode] Group identification bit
STATUS_ERROR	Status Receiving Error
STATUS_USUALLY_IDLE	Idle
STATUS_USUALLY_PRINTING	Printing
STATUS_USUALLY_PAPER_END	Paper End
STATUS_USUALLY_RIBBON_END	Ribbon End
STATUS_USUALLY_COOLING	Head Cooling Down
STATUS_USUALLY_MOTCOOLING	Motor Cooling Down
STATUS_SETTING_COVER_OPEN	Cover Open
STATUS_SETTING_PAPER_JAM	Paper Jam
STATUS_SETTING_RIBBON_ERR	Ribbon Error (Detect Error, Ribbon break)
STATUS_SETTING_PAPER_ERR	Paper Definition Error
STATUS_SETTING_DATA_ERR	Data Error (Illegal command)
STATUS_HARDWARE_ERR01	Head Voltage Error
STATUS_HARDWARE_ERR02	Head Position Error
STATUS_HARDWARE_ERR03	Power Supply Fan Stopped
STATUS_HARDWARE_ERR04	Cutter Error (Cut jamming etc)
STATUS_HARDWARE_ERR05	Pinch Roller Position Error
STATUS_HARDWARE_ERR06	Abnormal Head Temperature
STATUS_HARDWARE_ERR07	Abnormal Media Temperature
STATUS_HARDWARE_ERR08	Ribbon Tension Error
STATUS_HARDWARE_ERR09	RFID Module Error
STATUS_HARDWARE_ERR10	Abnormal Motor Temperature
STATUS_SYSTEM_ERR01	System Error
STATUS_FLSHPROG_IDLE	Idling for receiving rewriting data
STATUS_FLSHPROG_DEVICE_ERR1	Device Error
STATUS_FLSHPROG_OTHERS_ERR1	Other Error

[Sample Coding]

< Visual C >

```
include "CyStat.h"
long stat;

stat = GetStatus( CY );
if( stat & GROUP_USUALLY ){ // Usual Operation Status Group
        switch( stat ){
                case STATUS_USUALLY_IDLE: ;
                case STATUS_USUALLY_PRINTING: ;
                case STATUS_USUALLY_PAPER_END: ;
                  :
        }
}
if( stat & GROUP_SETTING ){ // Setting Error Status Group
        switch( stat ){
                case STATUS_SETTING_COVER_OPEN: ;
                  :
                  :
        }
}
```

< VB.NET >

```
Dim stat as Integer

stat = GetStatus(CY)
If stat And GROUP_USUALLY Then       ' Usual Operation Status Group
        Select Case stat
                Case STATUS_USUALLY_IDLE: Text1.Text = "IDLE"
                Case STATUS_USUALLY_PRINTING: Text1.Text = "PRINTING"
                Case STATUS_USUALLY_PAPER_END: Text1.Text = "PAPER_END"
                Case STATUS_USUALLY_RIBBON_END: Text1.Text = "RIBBON_END"
                Case STATUS_USUALLY_COOLING: Text1.Text = "COOLING"
        End Select
ElseIf stat And GROUP_SETTING Then  ' Setting Error Status Group
        ' Operation Error ( )
ElseIf stat And GROUP_HARDWARE Then          ' Hardware Error Status Group
        ' Hardware Error ( )
ElseIf stat And GROUP_SYSTEM Then   ' System Error Status Group
        ' System Error ( )
End If
```

Get Printer Counter Value

[Format]

```
long GetCounterL( long lPortNum );
long CvGetCounterL( long lPortNum );

long GetCounterA( long lPortNum );
long CvGetCounterA( long lPortNum );

long GetCounterB( long lPortNum );
long CvGetCounterB( long lPortNum );

long GetCounterP( long lPortNum );

long GetCounterMatte( long lPortNum );
long GetCounterM( long lPortNum );
```

[Parameter]

lPortNum:	Port number

[Return]

True:	Counter Value
False:	-1

[Note]

GetCounterL(), CvGetCounterL()　Returns Life Counter Value.
GetCounterA(), CvGetCounterA()　Returns Counter A Value.
GetCounterB(), CvGetCounterB()　Returns Counter B Value.
GetCounterP(long lPortNum)　Returns Counter P Value
GetCounterMatte()　Returns Matte Counter Value.
GetCounterM()　Returns Counter M Value.

Each counter is counted up 1 for each photo printed. But when printing 6x8(A5) or 5x7(2L) size, it will be counted up as two.
When printing multi-cut image, it will be counted up 2 when the second image is printed.

Counter P is initialized when power on.　Random setting is OK with SetCounterP().
Counter P value is countered according to each discharge of image.

When overcoat finish is matte print, Matte Counter and Counter M will be counted up (Life Counter and Counter A/B are counted up also).

Counter M is clearable with SetClearCounterM() function.
Specification of count up is the same as that of the above-mentioned Life Counter and Counter A/B.

[Sample Coding]

< Visual C >

```
long counter;
if(( counter = GetCounterL( CY )) >= 0 ){
        // Returns Life Counter Value to counter
}
```

< VB.NET >

```
Dim c As Integer
c = GetCounterL(CY)
If c >= 0 Then Text1.Text = Str(c) Else Text1.Text = "ERROR!"
```

Operation of Counter L/A/B/P count up

Count timing of counter is after cutting a print picture correctly.
When an error occurs, a counter does not go up.

	Print Size		Counter L/A/B Matte/M	Counter P
Single-cut	5x3.5 (L)		+1	+1
	5x7 (2L)		+2	+1
	6x4 (PC)		+1	+1
	6x8 (A5)		+2	+1
Multi-cut	(6x4)x2	1st Image	---	+1
		2nd Image	+2	+1
2inch-cut(*1)	6x4 (PC)	1st sheet	---	+1
		2nd sheet	+1	+1
	6x8 (A5)	1st sheet	---	+1
		2nd sheet	---	+1
		3rd sheet	---	+1
		4th sheet	+2	+1

*1. Effective at firmware Ver.1.10 or later,   Refer to Cutter Control Command

Clear Printer Counter Value

[Format]	BOOL SetClearCounterA(long lPortNum);
	BOOL CvSetClearCounterA(long lPortNum);
	
	BOOL SetClearCounterB(long lPortNum);
	BOOL CvSetClearCounterB(long lPortNum);
	
	BOOL SetClearCounterM(long lPortNum);

[Parameter]

[Parameter]	PortNum:	Port number
[Return]	True:	TRUE
	False:	FALSE

[Note]

SetClearCounterA(), CvSetClearCounterA()	Clears Counter A.
SetClearCounterB(), CvSetClearCounterB()	Clears Counter B.
SetClearCounterM()	Clears Counter M.

[Sample Coding]

< Visual C >
```
if( SetClearCounterA( CY )){
            // Counter A was cleared
}
```

< VB.NET >
```
If SetClearCounterA(CY) <> 0 Then
            ' Counter A was cleared
EndIf
```

Set Printer Counter Value (P)

[Format] BOOL SetCounterP(long lPortNum、long lCounter);

[Parameter]	lPortNum:	Port number
	lCounter:	Set counter value
[Return]	True:	TRUE
	False:	FALSE

[Note] SetCounterP() Sets Counter value P.
Set random number to Counter P. When power off, it is initialized to 0.

[Sample Coding]

< Visual C >
```
if( SetCounterP( CY, 100)){
            // Set Counter P
}
```

< VB.NET >
```
If SetCounterP(CY, 100) <> 0 Then
            ' Set Counter P
EndIf
```

Get the Number of Free Image Buffers

[Format]	long GetFreeBuffer(long lPortNum);
	long CvGetFreeBuffer(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True: False:	The number of free print buffers -1
[Note]	Printer will return the number of free image buffers.	

[Sample Coding]

< Visual C >
```
long bn;
if(( bn = GetFreeBuffer( CY )) >= 0 ){
        // Returns the number of free buffers to bn
}
```

< VB.NET >
```
Dim bn As Integer
bn = GetFreeBuffer(CY)
```

Get Remaining Print Quantity

[Format]	long GetPQTY(long lPortNum);
	long CvGetPQTY(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True: False:	Remaining number of images to be printed -1
[Note]	Returns the number of images remaining to be printed.	

[Sample Coding]

< Visual C >
```
long number;
if(( number = GetPQTY( CY )) >= 0 ){
        // Returns the number of remaining prints to number
}
```

< VB.NET >
```
Dim number As Integer
number = GetPQTY(CY)
```

Get the Media Counter of Remaining Sheets

[Format]

```
long GetMediaCounter( long lPortNum );
long CvGetMediaCounter( long lPortNum );

long GetMediaCounter_R( long lPortNum );
```

[Parameter] lPortNum: Port number

[Return] True: The number of remaining sheets
False: -1

[Note] Printer will return the number of sheets remaining in the printer.
Media tag number of sheets has the value of +50 sheets rather than the number of sheets which can be printed as follows.

GetMediaCounter() and CvGetMediaCounter() function return the value of media tag number of sheets as it is. In case you use these functions, please use it in consideration of the number of sheets of +50.

A GetMediaCounter_R function returns the value subtracted 50 from the number of sheets of a media tag, as the naumber of the remainder (aim) which can be printed, and when media tag number of sheets becomes 50 or less, it returns 0.

With operating conditions of media, the actual number of sheets which can be printed, and media tag number of sheets may be different. Detection of a media end should use together with the ribbon or paper end status of GetStatus() function.

■Initial value of number of sheets for each media

Media size	Number of sheets which can be printed	Number of sheets of media tag GetMediaCounter(), CvGetMediaCounter()	Number of sheets which can be printed (aim) GetMediaCounter_R()
5x3.5 (L)	700	750	700
6x4 (PC)	700	750	700
	200	250	200
5x7 (2L)	400	450	400
6x8 (A5)	400	450	400

[Sample Coding]

< Visual C >

```
long number;
if(( number = GetMediaCounter( CY )) >= 0 ){
            // Returns the number of remaining sheets to number
}
```

< VB.NET >

```
Dim number As Integer
number = GetMediaCounter(CY)
```

Get Media Color Offset Value of the Lot

[Format]	long GetMediaColorOffset(long lPortNum);
	long CvGetMediaColorOffset(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True:	Media color offset value of the lot
	False:	-1

[Note] Printer will return the offset value.

[Sample Coding]

< Visual C >

```
long offset;
if(( offset = GetMediaColorOffset( CY )) >= 0 ){
            // Returns the value to offset
}
```

< VB.NET >

```
Dim offset As Integer
offset = GetMediaColorOffset(CY)
```

[Example] In the case where offset = 169082895 (decimal) --> 0x0A14000F (hex), the offset values for each color are defined as below.

Color	Offset value
Yellow	10 (0x0A)
Magenta	20 (0x14)
Cyan	0 (0x00)
Op	15 (0x0F)

Get Media Lot Information

[Format]	long GetMediaLotNo(long lPortNum, LPSTR p);
	long CvGetMediaLotNo(long lPortNum, LPSTR p);	

[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	Printer will return the information stored in RFID tag of the media.	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if( GetMediaLotNo( CY, (LPSTR)rbuf ) > 0 ){
            // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetMediaLotNo(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

[Media tag information]

ML****************	User-specific information <16Bytes>

Get Printer Serial Number

[Format]	long GetSerialNo(long lPortNum, LPSTR p);
	long CvGetSerialNo(long lPortNum, LPSTR p);	

[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	Printer will return the printer serial number.	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if( GetSerialNo( CY, (LPSTR)rbuf ) > 0 ){
            // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetSerialNo(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

Set Firmware Update Mode

[Format]	BOOL SetFirmwUpdateMode(long lPortNum);
	BOOL CvSetFirmwUpdateMode(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True:	TRUE
	False:	FALSE

[Note] Changes the printer mode to firmware update mode.

[Sample Coding]

< Visual C >
```
if( SetFirmwUpdateMode(CY) > 0 ){
        // Next process
}
```

< VB.NET >
```
if ( SetFirmwUpdateMode(CY) > 0 ) Then
        ' Next Process
End if
```

Write Firmware Data

[Format]
BOOL SetFirmwDataWrite(long lPortNum, LPSTR lpData, DWORD dwDataLen);
BOOL CvSetFirmwDataWrite(long lPortNum, LPSTR lpData, DWORD dwDataLen);

[Parameter]	lPortNum:	Port number
	lpData:	Pointer to the buffer where the data is to be rewritten
	dwCmdLen:	The number of characters of the data
[Return]	True:	TRUE
	False:	FALSE

[Note] Sends data to rewrite firmware to the printer. The data is supplied by Motorola S format file. When this command is issued, a buffer (approx. 5M byte) is necessary for reading the file and for storing the data temporarily.
After completion of data writes, the printer is rebooted automatically, and Update Mode is reset.

<About VB.Net wrapper function>
When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.

[Sample Coding]

< VB.NET >
```
Dim fd(5000000)
Dim c as Long, n As Long

c=0
Open fname For Binary Access Read As #1
FileLength = LOF(1)
For n = 0 To FileLength - 1
        Get #1, , fd(c)
        c = c + 1
Next n
Close #1

SetFirmwDataWrite( CY, fd, c )
```

Clear Color Data

[Format]	BOOL SetColorDataClear(long lPortNum);
	BOOL CvSetColorDataClear(long lPortNum);	

[Parameter]	lPortNum:	Port number
[Return]	True: False:	TRUE FALSE

[Note] Clear color control data stored in the printer.
When rewriting color control data, please first clear color control data with this command.

[Sample Coding]

< Visual C >
```
if(SetColorDataClear ( CY )){
        // Color control data successfully cleared
}
```

< VB.Net >
```
If SetColorDataClear(CY) <> 0 Then
        ' Color control data successfully cleared
EndIf
```

Write Color Data

[Format]	BOOL SetColorDataWrite(long lPortNum, LPSTR lpData, DWORD dwDataLen);
	BOOL CvSetColorDataWrite(long lPortNum, LPSTR lpData, DWORD dwDataLen);	

[Parameter]	lPortNum: lpData: dwCmdLen:	Port number Pointer to the buffer where the color data is to be rewritten The number of characters of the data
[Return]	True: False:	TRUE FALSE

[Note] Sends color data to rewrite.
Before sending the new color data with this command, please first clear existing color data.
Color data is provided in an original format binary file, and when this command is issued, a buffer (approx. 100k byte) is necessary for reading the file and for storing the data temporarily.

<About VB.Net wrapper function>
When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.

[Sample Coding]

< VB.Net >
```
Dim fd(100000)
Dim c as Long, n As Long

c=0
Open fname For Binary Access Read As #1
FileLength = LOF(1)
For n = 0 To FileLength - 1
        Get #1, , fd(c)
        c = c + 1
Next n
Close #1

SetColorDataWrite( CY, fd, c )
```

Set Color Data Version

[Format]	BOOL SetColorDataVersion(long lPortNum, LPSTR lpData, DWORD dwDataLen);
	BOOL CvSetColorDataVersion(long lPortNum, LPSTR lpData, DWORD dwDataLen);	

[Parameter]	lPortNum:	Port number
	lpData:	Pointer to the buffer where the color data version is stored
	dwCmdLen:	The number of characters of the data
[Return]	True:	TRUE
	False:	FALSE

[Note]

Sets the color control data version.
After rewriting color control data, please set the version name as the file name of the color control data supplied.

<About VB.Net wrapper function>
When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.

[Sample Coding]

< VB.Net >
```
Dim fname As String
' Store version name in fname
SetColorDataVersion( CY, fname, Len(fname) )
```

Get Color Data Version

[Format]	long GetColorDataVersion(long lPortNum, LPSTR p);
	long CvGetColorDataVersion(long lPortNum, LPSTR p);	

[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	Printer will return the version name in the buffer.	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if(GetColorDataVersion( CY, (LPSTR)rbuf ) > 0 ){
        // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetColorDataVersion(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

Get Color Data Checksum

[Format]	long GetColorDataChecksum(long lPortNum, LPSTR p);
	long CvGetColorDataChecksum(long lPortNum, LPSTR p);	

[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	Printer will return checksum of the color data in the buffer.	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if(GetColorDataChecksum( CY, (LPSTR)rbuf ) > 0 ){
        // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetColorDataChecksum(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

Cutter Control Commands

[Format] long SetCutterMode(long lPortNum, DWORD ctMode);

[Function]

lPortNum:	port number
ctMode:	cutter mode selection

[Return]

Successful:	True
Failure:	False

[Explanation] This is to designate the cutter operation.
The cutter operation designation has a macro definition of CyStat.h (CyStat.bas for VB).
The symbols have the following meanings:

CUTTER_MODE_STANDARD	Standard cutter operation
CUTTER_MODE_NONSCRAP	Non-scrap cutter operation
CUTTER_MODE_2INCHCUT	2inch cut operation (Effective at firmware Ver.1.10 or later)

Note) The cutter control command sets the operation before the image data is sent. The command is valid for 1 image. After performing the designated cut for the printed image, the printer will return to its standard cut operation.
2inch cut operation is effective only in paper size 6x4 or 6x8.
When 2 inch cut setting of a printer driver is "Enable", the setup of operation by this command is invalid.

[Example]

< Visual C >
```
SetCutterMode(CY, (DWORD)CUTTER_MODE_NONSCRAP);
```

< VB.Net >
```
SetCutterMode(CY, CUTTER_MODE_NONSCRAP)
```

Get Media ID Set Info

[Format] long GetMediaIdSetInfo(long lPortNum);

[Parameter] lPortNum: Port number

[Return]

True:	Media ID set info
False:	-1

Return Value	The contents of a Media ID set info
0	CY is Always 0.

[Note] Return of the Media ID set info.

[Sample Coding]

< Visual C >
```
ong idval;
if(( idval = GetMediaIdSetInfo( CY )) >= 0 ){
            // Returns the Media ID set info
}
```

< VB.NET >
```
Dim idval As Integer
idval = GetMediaIdSetInfo(CY)
```

Get Medi Class

[Format]	long GetRfidMediaClass(long lPortNum, LPSTR p);	
[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	The media class data recorded in the RF-ID tag is returned. (ASCII character of four digits)	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if(GetRfidMediaClass( CY, (LPSTR)rbuf ) > 0 ){
            // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetRfidMediaClass(CY, s)
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

Get RF-ID reserve data

[Format]	long GetRfidReserveData (long lPortNum, LPSTR p, DWORD dwPage);	
[Parameter]	lPortNum:	Port number
	p:	Pointer to the receiving buffer
	dwPage:	Page of reserve data
[Return]	True:	The number of characters received by buffer p
	False:	-1
[Note]	The reserve data recorded in the RF-ID tag is returned. (ASCII character of four digits)	

[Sample Coding]

< Visual C >
```
char rbuf[256];
if(GetRfidReserveData ( CY, (LPSTR)rbuf ,0x01 ) > 0 ){ // Page 1
            // Next process
}
```

< VB.NET >
```
Dim s As String = New String("", 255)
Dim i As Integer
i = GetRfidReserveData (CY, s, &h2 )   'Page 2
If i > 0 Then Text1.Text = VB.Left(s, i) Else Text1.Text = "ERROR!"
```

Get initial media count

[Format]	long GetInitialMediaCount(long lPortNum);	
[Parameter]	lPortNum:	Port number
[Return]	True:	Initial media count
	False:	-1

[Explanation] This returns the initial media count.
The returned for the initial media count is the actual number +50, so when using a function, be sure to factor in the +50 count.

[Sample Coding]

< Visual C >

```
long number;
if(( number = GetInitialMediaCount ( CY )) >= 0 ){
        // Returns the number of initial media count to number
}
```

< VB.NET >

```
Dim number As Integer
number = GetInitialMediaCount (CY)
```

Common Set Command

[Format]	BOOL SetCommand(long lPortNum, LPSTR lpCmd, DWORD dwCmdLen);
	BOOL CvSetCommand(long lPortNum, LPSTR lpCmd, DWORD dwCmdLen);	

[Parameter]	lPortNum:	Port number
	lpCmd:	Pointer to the buffer where the command is stored
	dwCmdLen:	The number of characters of the command
[Return]	True:	TRUE
	False:	FALSE

[Note] Sends the command to the printer

<About VB.Net wrapper function>
When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.

Common Get Command

[Format]

```
long GetCommandEX( long lPortNum,
                   LPSTR lpCmd,
                   DWORD dwCmdLen,
                   LPSTR lpRetBuff,
                   DWORD dwRetBuffSize
                   );
long CvGetCommandEX( long lPortNum,
                   LPSTR lpCmd,
                   DWORD dwCmdLen,
                   LPSTR lpRetBuff,
                   DWORD dwRetBuffSize
                   );
```

[Parameter]	lPortNum:	Port number
	lpCmd:	Pointer to the buffer where the command is stored
	dwCmdLen:	The number of characters of the command
	lpRetBuff:	Pointer to the buffer to store receipt data
	dwRetBuffSize:	Available size of receiving buffer
[Return]	True:	The number of bytes received (Receipt data by RetBuff)
	False:	-1

[Note] After the command is sent to the printer, receipt data is stored in the buffer.

<About VB.Net wrapper function>
When using VB.Net, since the pointer cannot be given directly from VB to the DLL function argument, a wrapper function for VB.Net with the same name as the DLL has been prepared. The definition is written in the CyStat.vb file, so add the file to the VB project to use it.

Note for Sample Program

The Status API sample program is a program made with VB.Net, which gets the printer information using the Printer Status API.　How to use it is explained below.

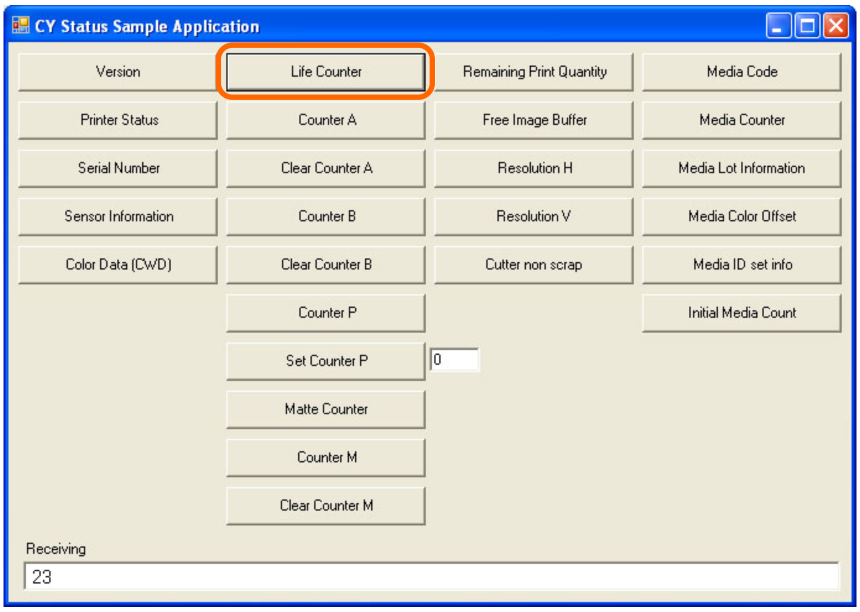
If you run the sample program, the screen will appear.　By clicking the buttons, you can get each type of information. An example of when you click the LifeCounter button is shown below.

When the LifeCounter button is clicked, the program below will be run, and with the status API function GetCounterL() the life counter value will be retrieved from the printer and displayed on the screen.

```
'********************* Get Life Counter
Private Sub Command4_Click()
Dim c As Long

    c = GetCounterL(CY)
    If c >= 0 Then Text2.Text = Str(c) Else Text2.Text = "ERROR!"

End Sub
```



Set Counter P: 0

Receiving: 23