O P E R A T I N G M A N U A L

XTC/3

Communications Library

IPN 074-454-P1B

INFICON

O P E R A T I N G M A N U A L

XTC/3

Communications Library

IPN 074-454-P1B

INFICON Instruments for Intelligent Control™

GLOBAL HEADQUARTERS:

Two Technology Place, East Syracuse, NY 13057 USA
Tel: +1.315.434.1100 Fax: +1.315.437.3803 E-mail: reachus@inficon.com

Visit our website for contact information and sales offices worldwide. **www.inficon.com**

Trademarks

The trademarks of the products mentioned in this manual are held by the companies that produce them.

INFICON® is a trademark of INFICON Inc.

All other brand and product names are trademarks or registered trademarks of their respective companies.

The information contained in this manual is believed to be accurate and reliable. However, INFICON assumes no responsibility for its use and shall not be liable for any special, incidental, or consequential damages related to the use of this product.

Registration Card

Thank you for selecting INFICON® instrumentation.
Please fill out and return this postage paid card as soon as possible.

Model _____ Serial # _____

Name _____

Title _____

Company _____ Bldg./MS _____

Address _____ Phone # _____

City _____ State _____ Zip _____

Country _____ Fax# _____ Email _____

Your help is very important in our continuing efforts to improve our manuals.
Using the table below, please circle the appropriate rank for each aspect.
In the Importance column, please indicate the importance of each aspect.

Manual Title _____

Part # (see Title Page) 074-_____

Aspect	Very Dissatisfied	Dissatisfied	No Opinion	Satisfied	Very Satisfied	Importance (ranked from 1 to 5, where 1 is low and 5 is high)
Found everything I needed	VD	D	NO	S	VS	
Easy to read	VD	D	NO	S	VS	
Easy to use	VD	D	NO	S	VS	
Relevant to my work	VD	D	NO	S	VS	
Accurate information	VD	D	NO	S	VS	
Well-written	VD	D	NO	S	VS	
Well-organized	VD	D	NO	S	VS	
Technical Enough	VD	D	NO	S	VS	
Helped me solve problems	VD	D	NO	S	VS	

If you have additional comments, please contact INFICON®.

INFICON

EXECUTIVE OFFICES:

Two Technology Place, East Syracuse, NY 13057 USA
Tel: +1.315.434.1100 Fax: +1.315.437.3803 E-mail: reachus@inficon.com

Visit us on the web at: www.inficon.com

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 49 EAST SYRACUSE, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

INFICON INC.
Two Technology Place
East Syracuse, New York 13057-9714

Warranty

WARRANTY AND LIABILITY - LIMITATION: Seller warrants the products manufactured by it, or by an affiliated company and sold by it, and described on the reverse hereof, to be, for the period of warranty coverage specified below, free from defects of materials or workmanship under normal proper use and service. The period of warranty coverage is specified for the respective products in the respective Seller instruction manuals for those products but shall not be less than one (1) year from the date of shipment thereof by Seller. Seller's liability under this warranty is limited to such of the above products or parts thereof as are returned, transportation prepaid, to Seller's plant, not later than thirty (30) days after the expiration of the period of warranty coverage in respect thereof and are found by Seller's examination to have failed to function properly because of defective workmanship or materials and not because of improper installation or misuse and is limited to, at Seller's election, either (a) repairing and returning the product or part thereof, or (b) furnishing a replacement product or part thereof, transportation prepaid by Seller in either case. In the event Buyer discovers or learns that a product does not conform to warranty, Buyer shall immediately notify Seller in writing of such non-conformity, specifying in reasonable detail the nature of such non-conformity. If Seller is not provided with such written notification, Seller shall not be liable for any further damages which could have been avoided if Seller had been provided with immediate written notification.

THIS WARRANTY IS MADE AND ACCEPTED IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, WHETHER OF MERCHANTABILITY OR OF FITNESS FOR A PARTICULAR PURPOSE OR OTHERWISE, AS BUYER'S EXCLUSIVE REMEDY FOR ANY DEFECTS IN THE PRODUCTS TO BE SOLD HEREUNDER. All other obligations and liabilities of Seller, whether in contract or tort (including negligence) or otherwise, are expressly EXCLUDED. In no event shall Seller be liable for any costs, expenses or damages, whether direct or indirect, special, incidental, consequential, or other, on any claim of any defective product, in excess of the price paid by Buyer for the product plus return transportation charges prepaid.

No warranty is made by Seller of any Seller product which has been installed, used or operated contrary to Seller's written instruction manual or which has been subjected to misuse, negligence or accident or has been repaired or altered by anyone other than Seller or which has been used in a manner or for a purpose for which the Seller product was not designed nor against any defects due to plans or instructions supplied to Seller by or for Buyer.

NOTE: These instructions do not provide for every contingency that may arise in connection with the installation, operation or maintenance of this equipment. Should you require further assistance, please contact INFICON.

INFICON

EXECUTIVE OFFICES:

Two Technology Place, East Syracuse, NY 13057 USA
Tel: +1.315.434.1100 Fax: +1.315.437.3803 E-mail: reachus@inficon.com

Visit us on the web at: www.inficon.com

How To Contact Customer Support

Worldwide support information regarding:

- Technical Support, to contact an applications engineer with questions regarding INFICON products and applications, or
- Sales and Customer Service, to contact the INFICON Sales office nearest you, or
- Repair Service, to contact the INFICON Service Center nearest you,

is available at www.inficon.com.

If you are experiencing a problem with your instrument, please have the following information readily available:

- the serial number for your instrument,
- a description of your problem,
- an explanation of any corrective action that you may have already attempted,
- and the exact wording of any error messages that you may have received.

To contact Customer Support, see Support at www.inficon.com.

Table Of Contents

IPN 074-454-P1B

IPN 074-454-P1B

IPN 074-454-P1B

This page is intentionally blank.

IPN 074-454-P1B

Chapter 1
XTC/3 Communications Library (DLL)

1.1 Introduction

The XTC/3 Communications Library contains functions that allow creation of a program, for a remote PC, to control an XTC/3M or an XTC/3S instrument via an RS232 or TCP/IP connection.

The library contains two types of functions - **Communications** and **Command**. This document briefly describes each function, providing a prototype, arguments and structure definitions. An example is provided for each function, with one example shown where many functions have common prototypes.

The library consists of four files:

- **XTC3LibConn.h**, which contains the declaration of the TCP/IP and RS232 communication functions.
- **XTC3Lib.h**, which contains the declarations of the Data Acquisition functions (e.g. data send and query), Status functions and Remote functions.

  NOTE: These two **.h** files must be included in the user's project.
- **XTCComLib.dll** and **XTC3ComLib.lib** must be copied into the directory containing the .exe file. The user's project must be configured to link to **XTC3ComLib.lib** .

This manual contains many variables (e.g. **NUM_OF_LAYERS**) that are defined in the files listed above. Always refer to the files for the current definition of each variable.

IPN 074-454-P1B

1.2 Communications Functions

There are two groups of Communications Functions - Serial and Ethernet - and each group must be used to successfully establish the respective connection.

1.2.1 Setting the Serial Port

The following functions must be called sequentially — in the order of **SetXTC3Port**, **SetBaud**, **SetTimeOut** and **XTC3Open** — to open a serial connection with an XTC/3 instrument. The port must be closed prior to exiting the application by using the **XTC3Close** function.

1.2.1.1 SetXTC3Port

```
void SetXTC3Port(short Port);
```

Description

This function sets the port to be used for serial communications.

Arguments

short Port:

The port number on the PC to be selected (e.g. 1, if the PC only has one serial port).

1.2.1.2 SetBaud

```
void SetBaud(int Baud);
```

Description

This function sets the baud rate for the serial port selected.

Arguments

int Baud:

The Baud argument can be one of the following for the XTC/3:

- 9600
- 19200
- 38400
- 57600
- 115200

The recommended baud rate is **115200**.

1.2.1.3 SetTimeOut

```
void SetTimeOut(long TimeOut);
```

Description

This function sets the timeout for the serial port selected.

Arguments

long TimeOut:

A TimeOut is simply the length of time in milliseconds that a communication remains open before closing due to inactivity or improper communications.

IPN 074-454-P1B

1.2.1.4 XTC3Open

```
TC_ERROR XTC3Open();
```

Description

This function opens the serial port set previously.

Arguments

None

Returns

A TC_ERROR is returned if the port in question can not be opened.

If the port is successfully opened, a TC_E_SUCCESS value is returned.

TC_E_PORT_UNAVAILABLE is returned if the port is being used by another application, such as HyperTerminal, for example.

1.2.1.5 XTC3Close

```
TC_ERROR XTC3Close();
```

Description

This function closes the serial port previously opened.

Arguments

None

Returns

A TC_ERROR is returned if the port in question can not be closed.

If the port is successfully closed, a TC_E_SUCCESS value is returned.

1.2.1.6 Example

```
SetXTC3Port(1);
SetBaud(115200);
SetTimeOut(3000);
If(XTC3Open()==TC_ERROR)
   printf("Cannot open serial port");
//.........application code
XTC3Close();
```

IPN 074-454-P1B

1.2.2 Setting the Ethernet Port

The following functions must be called sequentially - in the order of **StartSocket**, **ConnectSocket** - to open an Ethernet port. The port must be closed prior to exiting the application by using the **CloseSocket** function.

1.2.2.1 StartSocket

```
int StartSocket();
```

Description

This function, which opens a socket on the Ethernet port, needs to be called only once (at the beginning of the application, for example).

Arguments

None

Returns

If the port is successfully opened, a TC_E_SUCCESS value is returned.

A TC_ERROR is returned if the port in question can not be opened.

TC_E_PORT_UNAVAILABLE is returned if the port is being used by another application, such as HyperTerminal, for example.

1.2.2.2 ConnectSocket

```
BOOL ConnectSocket(char* host);
```

Description

This function connects the socket to the host. After determining an IP address for the XTC/3, in the form of XXX.XXX.XXX.XXX, this function is called using that address as the host. This function also creates the socket if it does not exist when called.

Arguments

char host:*

The IP Address in the form of XXX.XXX.XXX.XXX

Returns

If the socket is successfully connected then a TRUE is returned.

If the socket is not connected then a FALSE is returned.

IPN 074-454-P1B

1.2.2.3 CloseSocket

```
void CloseSocket();
```

Description

This function closes the socket previously opened.

Arguments

None

1.2.2.4 Example

```
if(StartSocket()==0)
   printf("Socket could not be initialized.");
if(!ConnectSocket("10.211.70.209"))
   printf("Socket could not be connected!");
//.........application code
CloseSocket();
```

IPN 074-454-P1B

1.3 XTC/3 Command Functions

These functions acquire data from, and send data to, the instrument via either communications connection. The data is either packed in regular types such as *int*, *float*, and *char* or marshaled in structures for larger quantity of bytes (> 4 bytes).

1.3.1 Data Structures

There are many different data structures used by the command functions to pass data. This section defines each data structure.

1.3.1.1 XTC3AllData

```
struct XTC3AllData {
   unsigned short m_Length;
   unsigned char m_Data[XTC3_ALL_SIZE];
};
```

Description

This structure is used for downloading or uploading the instrument parameters, which can include all or part of the following:

- **General** = XTC3_MAX_GEN_PARAM x 4 bytes
- **Film** = NUM_OF_FILMS x MAX_FILM_PARAM x 4 bytes
- **IOMap** = (NUM_OF_RELAYS + NUM_OF_TTLs + XTC3_NUM_OF_INPUTS) bytes
- All **Film** Names (XTC/3M only): NUM_OF_FILMS x (up to XTC3_TITLE_LENGTH, null terminator included)
- Up to PART_NUM_OF_PROCESSES **Processes** filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x PART_NUM_OF_PROCESSES)

NOTE: See command UB1 and QB1 in the User's Guide document.

This structure is also used to hold the response returned by the XTC/3 instrument after a macro command is received. The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Variable m_Length is a 2 byte short that represents the length of data being marshaled in array m_Data.

IPN 074-454-P1B

1.3.1.2 XTC3String

```
struct XTC3String {
   unsigned char m_Data[XTC3_TITLE_LENGTH];
};
```

This structure is used mostly for strings such as names, version etc.. XTC/3 names each have a maximum length of 16 characters, which is the value of the constant XTC3_TITLE_LENGTH.

1.3.1.3 OutAllData:

```
struct OutAllData {
   unsigned char m_Data[XTC3_NUM_OF_OUTPUTS];
};
```

There are NUM_OF_RELAYS **Relays** and NUM_OF_TTLs **TTLs** in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This structure is used to hold a maximum of 20 (the value of constant XTC3_NUM_OF_OUTPUTS) events, with each event represented by a byte.

1.3.1.4 InAllData

```
struct InAllData{
   unsigned char m_Data[XTC3_NUM_OF_INPUTS];
};
```

There are XTC3_NUM_OF_INPUTS **Inputs** in the XTC/3 instrument. Each input can be programmed to react to a certain event. This structure is used to hold a maximum of XTC3_NUM_OF_INPUTS events, each event represented by a byte.

1.3.1.5 XTC3GenAllData

```
struct XTC3GenAllData {
   int m_Data[XTC3_MAX_GEN_PARAM];
};
```

There are XTC3_MAX_GEN_PARAM General Parameters each represented by 4 bytes.

1.3.1.6 LayerData

```
struct LayerData {
   unsigned char m_Data[NUM_OF_LAYERS + 2];
};
```

This structure can hold a maximum of NUM_OF_LAYERS **Layers**. Each layer is represented by a byte, which is basically the associated film number. Because most of the time not all NUM_OF_LAYERS **Layers** are programmed, this structure may be only partially filled and the first 2 bytes of m_Data represent that quantity.

IPN 074-454-P1B

1.3.1.7 SpecLayerData

```
struct SpecLayerData{
   unsigned char m_Data[3];
};
```

This is a 3 byte structure that contains the layer number (1 to NUM_OF_LAYERS) in the first 2 bytes, and the associated film number (1 to NUM_OF_FILMS) in the last byte.

1.3.1.8 FilmAllData

```
struct FilmAllData {
   unsigned char m_Data[MAX_FILM_PARAM*4];
};
```

This structure holds all the film parameters, namely MAX_FILM_PARAM x 4 bytes.

1.3.1.9 ProcInfoData

```
struct ProcInfoData {
   unsigned char m_Data[PROC_INFO_SIZE];
};
```

This structure can contain up to PROC_INFO_SIZE bytes worth of information (e.g. Rate, Rate Deviation, Thickness, Power, Frequency). The Data Logger primarily uses this data in the XTC/3 software application.

1.3.1.10 DataLogStruct

```
struct DataLogStructtag {
   unsigned char m_Data[DATA_LOG_SIZE];
};
```

This structure can contain up to DATA_LOG_SIZE bytes of status information (e.g. Layer#, Film#).

1.3.1.11 XTC3HelloData

```
struct XTC3HelloData {
   unsigned long m_Data[XTC3_HELLO_LENGTH];
};
```

This structure will hold XTC3_HELLO_LENGTH integers, namely:

- Structure number
- Compatibility number
- Range number
- Unit Type (0x4d = XTC/3M, 0x53 = XTC/3S)

IPN 074-454-P1B

1.3.1.12 XTC3HelloVersion

```
struct XTC3HelloVersion {
   unsigned long m_Data[XTC3_VERSION_SIZE];
};
```

This structure will hold XTC3_VERSION_SIZE bytes for the name and version string (e.g. "XTC/3M Version 1.00").

1.3.2 Command Functions

All the functions described in this section contain the following common arguments, which are described here instead of within each function.

1.3.2.1 Common Arguments

char ErrMsg*

ErrMsg is a string that will contain the error message. If the transmission has been successful then this string will be empty.

*unsigned short *len*

This is the length in bytes of the useful data in ***val***. The user can, for example, use *len* to extract the **val* data.

BOOL TCP

This variable must be initialized to TRUE for TCP/IP communications or FALSE for RS232 communications.

IPN 074-454-P1B

1.3.2.2 getHello

*int **getHello** (XTC3HelloVersion *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function gets the name and version of the XTC/3 instrument.

Arguments

*XTC3HelloVersion *val*

This structure is filled with the response, specifically the name and version of the XTC/3 instrument.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   unsigned short length = 0;
   memset(ErrStr,0,256);
   XTC3HelloVersion str;
   BOOL TCP = TRUE; // TCP/IP comm
   getHello (&str, ErrStr, &length, TCP);
   if(*ErrStr == 0)
      printf("Data Transfer successful/n");
      printf("Version and Name: %s/n", str.m_Data);
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.3 getHelloTwo

int **getHelloTwo** *(XTC3HelloData *val, char* ErrMsg, unsigned short *len, BOOL TCP)*;

Description:

This function gets the Structure Number, Compatiblity Number, Range Number, and Unit Type (XTC/3M vs XTC/3S).

Arguments:

*XTC3HelloData *val*

This 16 byte structure will hold the 4 integers described above, in that order, sent by the XTC/3 instrument.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example:

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   unsigned short length = 0;
   memset(ErrStr,0,256);
   XTC3HelloData val;
   BOOL TCP = TRUE; // TCP/IP comm.
   getHelloTwo (&val,  ErrMsg, length, TCP);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("Structure Number: %d\n"
         "Compatibility Number: %d\n"
         " Range Number: %d\n"
         " Unit Type: %d\n",
         val.m_Data[0], val.m_Data[1], val.m_Data[2],
         val.m_Data[3]);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.4 getXTC3Version

int **getXTC3Version** *(float *val, char* ErrMsg, unsigned short *len, BOOL TCP)*;

Description:

This function gets the firmware version number in float.

Arguments:

*float *val*

This is the pointer to a float number that represents the firmware version, sent by the XTC/3 instrument.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example:

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   unsigned short length = 0;
   memset(ErrStr,0,256);
   XTC3HelloData val;
   BOOL TCP = TRUE; // TCP/IP comm.
   getHelloTwo (&val,  ErrMsg, length, TCP);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("Structure Number: %d\n"
         "Compatibility Number: %d\n"
         " Range Number: %d\n"
         " Unit Type: %d\n",
         val.m_Data[0], val.m_Data[1], val.m_Data[2],
         val.m_Data[3]);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.5 get Echo

*int **getEcho** (XTC3String *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function is used to test the quality of communication by sending a string of at most 16 characters and receiving back the same string.

Arguments

*XTC3String *val:*

Holds the echo string

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3String str;
   strcpy((char*)str.m_Data,"Are you there?");
   unsigned short length = strlen((const char*)(str.m_Data))+1;
   BOOL TCP = TRUE; // TCP/IP comm
   getEcho(&str, ErrStr, &length, TCP);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("The returned string is %s\n",str.m_Data);
   }
   else
      printf(ErrStr);
   CloseSocket();

}
```

1.3.2.6 setGenParameter

*int **setGenParameter** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*

Description

SetGenParameter sets the value of a General Parameter in the instrument.

Arguments

*unsigned long *val:*

a 4 byte value of a General Parameter.

int index:

The index of the General Parameter as described in the User's Guide document.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
// This example will change the "Audio Feedback" to Yes.
// The index of this parameter is 12.
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned long val = 1; // 1 for YES and 0 for NO
   BOOL TCP = TRUE; // TCP/IP comm
   setGenParameter(&val, 12, ErrStr, TCP);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.7 getGenParameter

*int **getGenParameter** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

GetGenParameter gets the value of a General Parameter from the instrument.

Arguments:

*unsigned long *val:*

A 4 byte long that will contain the value of a General Parameter from the instrument.

int index:

The index of the General Parameter as described in the User's Guide document.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
// This example gets the "Recorder Mode".
// The index of this parameter is 9.
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");

   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned long val = 0;
            // 0 = Rate 1 = Thickness
            // 2 = Power 3 = Rate Deviation
   unsigned short len = 0;
   char* Response[] =
   {
      "Rate",
      "Thickness",
      "Power",
      "Rate Deviation"
   };

   BOOL TCP = TRUE; // TCP/IP comm
   getGenParameter(&val, 9, ErrStr, &len, TCP);
```

```
   ASSERT(val >= 0 && val < 4);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("The Current Recorder Mode is: %s \n",Response[val]);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.8 getGenAll

*int **getGenAll** (XTC3GenAllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

getGenAll gets all XTC3_MAX_GEN_PARAM General Parameters from the instrument.

Arguments

*XTC3GenAllData *val:*

An XTC3_MAX_GEN_PARAM x 4 bytes structure that will contain XTC3_MAX_GEN_PARAM General Parameters from the instrument.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
// This example gets all the General Parameters from the XTC/3
// instrument.
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3GenAllData val;
   unsigned short len = 0;

   BOOL TCP = TRUE; // TCP/IP comm
   getGenAll (&val, ErrStr, &len, TCP);
   if(*ErrStr == 0)
   {
      ASSERT(len >= 0 && len/4 <= XTC3_MAX_GEN_PARAM);
      printf("Data Transfer successful\n");
```

IPN 074-454-P1B

```
      for(int i = 0; i < len/4; i++)
         printf("%d\n",val.m_Data[i]);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.9 setGenAll

int ***setGenAll*** *(XTC3GenAllData *val, char* ErrMsg, BOOL TCP);*

Description

setGenAll sets all XTC3_MAX_GEN_PARAM General Parameters in the XTC/3 instrument.

Arguments:

*XTC3GenAllData *val:*

An XTC3_MAX_GEN_PARAM x 4 bytes structure that will contain XTC3_MAX_GEN_PARAM General Parameters sent to the instrument.

char ErrMsg*

BOOL TCP

Example

```
// This example sends all the General Parameters to the XTC/3
// instrument
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3GenAllData val;
   int i = 0;
   val.m_Data[i++] = 63; // This is an XTC/3M -> Process to run = 63
   val.m_Data[i++] = 0; // Do NOT Start Layer Without Backup Xtal
   val.m_Data[i++] = 1; // Stop on Alarms
   val.m_Data[i++] = 0; // Do NOT Stop On Max Power
   val.m_Data[i++] = 0; // Deposit Mode
   val.m_Data[i++] = 1; // Test ON
   val.m_Data[i++] = 0; // Sensor 1 Type: SINGLE
   val.m_Data[i++] = 0; // Sensor 2 Type: SINGLE
   val.m_Data[i++] = 1; // Source Control Voltage: 0 to -10
   val.m_Data[i++] = 0; // Recorder Mode : RATE
   val.m_Data[i++] = 1; // Recorder Range: 1000
```

IPN 074-454-P1B

```
   val.m_Data[i++] = 0; // Recorder Filter:Unfiltered
   val.m_Data[i++] = 0; // NO Beep
   val.m_Data[i++] = 0; // Do NOT Dim LCD
   val.m_Data[i++] = 4; // RS232 Baud Rate: 115200
   val.m_Data[i++] = 0; // RS232 Comm Protocol: Standard RS232
   val.m_Data[i++] = 1;
            // Auto Start Next Layer/Input Option: Standard
   unsigned short len = 0;
   BOOL TCP = TRUE; // TCP/IP comm
   setGenAll (&val, ErrStr,  TCP);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.10 getFilmAll

*int **getFilmAll** (FilmAllData *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

getFilmAll gets all MAX_FILM_PARAM Film Parameters from the instrument.

Arguments

*FilmAllData *val:*

A MAX_FILM_PARAM x 4 bytes structure that will contain MAX_FILM_PARAM Film Parameters from the instrument.

int index:

The film index (1 to MAX_FILM_PARAM for XTC/3M , 1 to NUM_OF_FILMS_FOR_SINGLE for XTC/3S)

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
// This example gets all the Film Parameters from the XTC/3
// instrument.
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
```

IPN 074-454-P1B

```
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   BOOL TCP = TRUE; // TCP/IP comm
   getFilmAll (&val, 1, ErrStr, &len, TCP);

   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("Film Parameters shown in Hex Format:\n");
      ASSERT(len  == MAX_FILM_PARAM*4);
      for(int i = 0; i < MAX_FILM_PARAM*4; i++)
      {
         if(!(i%4))
            printf("\n %d: ",i/4);
         if(val.m_Data[i] < 0x0f)
            printf("0%X ",val.m_Data[i]);
         else
            printf("%X ",val.m_Data[i]);
      }
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.11 setFilmAll

*int **setFilmAll** (FilmAllData *val, int index , char* ErrMsg, BOOL TCP);*

Description

setFilmAll sets all MAX_FILM_PARAM Film Parameters in the XTC/3 instrument.

Arguments

*FilmAllData *val:*

a MAX_FILM_PARAM x 4 bytes structure that will contain MAX_FILM_PARAM Film Parameters sent to the instrument.

int index:

The film index (1 to NUM_OF_FILMS for XTC/3M and 1to NUM_OF_FILMS_FOR_SINGLE for XTC/3S)

char ErrMsg*

BOOL TCP

Example

```
// This example sends all the Film Parameters to the XTC/3
// instrument. For simplicity, the example first uses
// getFilmAll (see above). Once the structure is filled,
```

```
// a couple of parameters are changed and sent through.

#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   BOOL TCP = TRUE; // TCP/IP comm
   getFilmAll (&val, 1, ErrStr, &len, TCP);

   if(*ErrStr == 0)
   {
      float finalThickness = 86.24F;
      memcpy((unsigned char*)val.m_Data + 10*4,&finalThickness,4);
      setFilmAll (&val, 1 , ErrStr,  TCP);
      if(*ErrStr == 0)
         printf("Data Transfer successful\n");
      else
         printf(ErrStr);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.12 Film Functions

All the Set and Get film functions described in Table 1-1 have the same arguments - except for the first argument - which can be an unsigned long or a float, as shown below:

Description

setFilmxxxx sets a Film Parameter in the XTC/3 instrument.

getFilmxxxx gets a Film Parameter from the XTC/3 instrument.

Arguments

*unsigned long *val or (float *val):*

a 4 byte value that represents the film parameter sent to or received from the instrument.

int index:

The film index (1 to NUM_OF_FILMS for XTC/3M and 1 to NUM_OF_FILMS_FOR_SINGLE for XTC/3S)

IPN 074-454-P1B

char ErrMsg*

BOOL TCP

An example is shown for the *getFilmSoakPowerOne* function and Soak Power 1 parameter.

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   float SoakPowerOne = 0.0F;
   getFilmSoakPowerOne (&SoakPowerOne, 1, ErrStr, &len, TRUE);

   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("%f\n",SoakPowerOne);
      SoakPowerOne = 88.23F;
      setFilmSoakPowerOne (&SoakPowerOne, 1 , ErrStr,  TRUE);
      if(*ErrStr == 0)
         printf(" SET Data Transfer successful\n");
      else
         printf(ErrStr);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

Table 1-1 Film Parameters accessible via Set and Get

Command ID	Description	Function
0	Rise Time 1	*int **setFilmRiseTimeOne** (unsigned long *val, int index,char* ErrMsg, BOOL TCP);*
		*int **getFilmRiseTimeOne** (unsigned long *val, int index, char* ErrMsg,unsigned short *len, BOOL TCP);*
1	Soak Power 1	*int **setFilmSoakPowerOne** (float *val, int index, char* ErrMsg, BOOL TCP);*
		*int **getFilmSoakPowerOne** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
2	Soak Time 1	*int **setFilmSoakTimeOne** (unsigned long *val, int index, char* ErrMsg, BOOL TCP);*
		*int **getFilmSoakTimeOne** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
3	Rise Time 2	*int **setFilmRiseTimeTwo** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmRiseTimeTwo** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
4	Soak Power 2	*int **setFilmSoakPowerTwo** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmSoakPowerTwo** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
5	Soak Time 2	*int **setFilmSoakTimeTwo** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmSoakTimeTwo** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
6	Idle Ramp	*int **setFilmIdleRamp** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmIdleRamp** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
7	Idle Power	*int **setFilmIdlePower** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmIdlePower** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
8	Deposition Rate	*int **setFilmDepRate** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmDepRate** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

IPN 074-454-P1B

Table 1-1 Film Parameters accessible via Set and Get (continued)

Command ID	Description	Function
9	Final Thickness	*int **setFilmFinThick** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmFinThick** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
10	Thickness Set Point	*int **setFilmThickSetOpt** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmThickSetOpt** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
11	New Rate	*int **setFilmNewRate** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmNewRate** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
12	Rate Ramp Time	*int **setFilmRateRampTime** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmRateRampTime** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
13	RateWatcher Accuracy	*int **setFilmRateWatchAcc** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmRateWatchAcc** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
14	RW Hold Time	*int **setFilmRWHoldTime** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmRWHoldTime** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
15	Sensor	*int **setFilmSensor** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmSensor** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
16	Tool Factor 1	*int **setFilmToolFctr1** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmToolFctr1** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
17	Tool Factor 2	*int **setFilmToolFctr2** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmToolFctr2** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

IPN 074-454-P1B

Table 1-1 Film Parameters accessible via Set and Get (continued)

Command ID	Description	Function
18	Xtal Stability Single	*int **setFilmXtalStbltySingle** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmXtalStbltySingle** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
19	Xtal Stability Total	*int **setFilmXtalStbltyTotal** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmXtalStbltyTotal** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
20	Xtal Quality Percent	*int **setFilmXtalQltyPcent** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmXtalQltyPcent** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
21	Xtal Quality Counts	*int **setFilmXtalQltyCnts** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmXtalQltyCnts** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
22	Source	*int **setFilmSource** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmSource** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
23	Crucible	*int **setFilmCrucible** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmCrucible** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
24	Control Gain	*int **setFilmControlGain** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmControlGain** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
25	Control Time Constant	*int **setFilmControlTimeCst** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmControlTimeCst** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
26	Control Dead Time	*int **setFilmControlDeadTime** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmControlDeadTime** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

IPN 074-454-P1B

Table 1-1 Film Parameters accessible via Set and Get (continued)

Command ID	Description	Function
27	Max Power	*int **setFilmMaxPower** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmMaxPower** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
28	Density	*int **setFilmDensity** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmDensity** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
29	Z-Ratio	*int **setFilmZRatio** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmZRatio** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
30	Time Power	*int **setFilmTimePower** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmTimePower** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
31	Delay option	*int **setFilmDelayOption** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmDelayOption** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
32	Transfer Sensor	*int **setFilmTrsferSensor** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmTrsferSensor** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
33	Transfer Tooling	*int **setFilmTrsferTooling** (float *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmTrsferTooling** (float *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
34	Control Delay Time	*int **setFilmCtrlDelayTime** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmCtrlDelayTime** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
35	Ion Assisted Deposit	*int **setFilmIonDep** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmIonDep** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Table 1-1 Film Parameters accessible via Set and Get (continued)

Command ID	Description	Function
36	Graph Label	*int **setFilmGraphLabel** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmGraphLabel** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*
37	Graph Scale	*int **setFilmGraphScale** (unsigned long *val, int index , char* ErrMsg, BOOL TCP);*
		*int **getFilmGraphScale** (unsigned long *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

NOTE: The values contained in the Command ID column above are only a reference to the raw commands described in the User's Guide. For example, *setFilmRiseTimeOne*, which has a Command ID of **0**, refers to UF**0** in the User's Guide.

IPN 074-454-P1B

1.3.2.13 setProcessLayerList

int ***setProcessLayerList*** *(LayerData *val, int index , char* ErrMsg, BOOL TCP);*

Description

setProcessLayerList fills a process with layers.

Arguments

*LayerData *val:*

A structure that contains the number of layers (first 2 bytes) followed with the layers (1 byte per, max of NUM_OF_LAYERS bytes) represented each with a film number (1 to NUM_OF_FILMS);

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   LayerData val;
   unsigned short NumOfLayers = 3;
   memcpy(val.m_Data,&NumOfLayers,2);
   val.m_Data[2] = 12; // add a Film12  layer to process 1
   val.m_Data[3] = 31; // add a Film31  layer to process 1
   val.m_Data[4] = 4; // add a Film4  layer to process 1
   setProcessLayerList (&val, 1 , ErrStr,  TRUE);

   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.14 getProcessLayerList

*int **getProcessLayerList** (LayerData *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

getProcessLayerList gets all the layers from a process at *index* from the XTC/3 instrument and puts them in a LayerData structure.

Arguments

*LayerData *val:*

A structure that contains the number of layers (first 2 bytes) followed with the layers (1 byte per, max of NUM_OF_LAYERS bytes) represented each with a film number (1 to NUM_OF_FILMS);

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   LayerData val;
   unsigned short NumOfLayers = 0;
   unsigned short len = 0;
   getProcessLayerList (&val, 1, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      memcpy(&NumOfLayers,val.m_Data,2);
      ASSERT(NumOfLayers <= NUM_OF_LAYERS);
      printf("The Layers of Process 1 are:\n");
      for(int i = 2; i < NumOfLayers + 2; i++)
      printf("%d\n",val.m_Data[i]);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.15 setProcessName

int ***setProcessName*** *(XTC3String *val, int index , char* ErrMsg, BOOL TCP);*

Description

setProcessName sets the process name (up to 15 characters + Null Terminator) in the XTC/3 instrument.

Arguments

*XTC3String *val:*

A structure that contains the name of the process (16 characters max)

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
//This example renames Process 1 to Process_ONE
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3String val;
   strcpy((char*)(val.m_Data),"Process_ONE");
   unsigned short len = 0;
   setProcessName (&val, 1, ErrStr,  TRUE);

   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();


}
```

1.3.2.16 getProcessName

*int **getProcessName** (XTC3String *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

getProcessName gets the process name (up to 15 characters + Null Terminator) from the XTC/3 instrument.

Arguments

*XTC3String *val:*

A structure that contains the name of the process (16 characters max)

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3String val;
   strcpy((char*)(val.m_Data),"Process_ONE");
   unsigned short len = 0;
   getProcessName (&val,1, ErrStr, &len, TRUE);

   if(*ErrStr == 0)
      printf("%s\n",(char*)(val.m_Data));
   else
      printf(ErrStr);
   CloseSocket();


}
```

IPN 074-454-P1B

1.3.2.17 setProcessSpecLayer

*int **setProcessSpecLayer** (SpecLayerData *val, int index , char* ErrMsg, BOOL TCP);*

Description

setProcessSpecLayer updates a Layer in a Process (e.g. changes its film number).

Arguments

SpecLayerData val:*

This is a 3 byte structure that contains the layer number (first 2 bytes, 1 to NUM_OF_LAYERS) and the film number with which the layer is to be updated (last byte).

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   SpecLayerData val;
   unsigned short layerNumber = 3;
   memcpy(val.m_Data, &layerNumber,2);
   val.m_Data[2] =13;
   unsigned short len = 0;
   setProcessSpecLayer (&val, 1, ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

1.3.2.18 getProcessSpecLayer

*int **getProcessSpecLayer** (SpecLayerData *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

getProcessSpecLayer gets the film number of a layer at a process.

Arguments

SpecLayerData val:*

This is a 3-byte structure, which is first filled with the requested layer number (first 2 bytes) before the function is called. The function response returns the film number in the first byte of the structure. The remaining two structure bytes are unused and should be ignored.

int index:

The process index (1 to NUM_OF_PROCESSES)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   SpecLayerData val;
   unsigned short layerNumber = 3;
   memcpy(val.m_Data, &layerNumber,2);
   unsigned short len = 0;
   getProcessSpecLayer (&val, 1, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("The film number at layer %d is”
      ”%d\n",layerNumber,val.m_Data[0]);
   }
   else
      printf(ErrStr);
   CloseSocket();


}
```

IPN 074-454-P1B

1.3.2.19 setOutAll

int ***setOutAll*** *(OutAllData *val, char* ErrMsg, BOOL TCP);*

Description

There are NUM_OF_RELAYS Relays and NUM_OF_TTLs TTLs in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This function will do that via *OutAllData.*

Arguments

*OutAllData * val:*

This structure should be filled with events for each relay and TTL (XTC3_NUM_OF_OUTPUTS bytes). See User's Guide document for event numbers and description.

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   OutAllData val;
   for(int i = 0; i < XTC3_NUM_OF_OUTPUTS; i++)
      val.m_Data[i] = i + 1;
   setOutAll (&val, ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

1.3.2.20 getOutAll

*int **getOutAll** (OutAllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

There are NUM_OF_RELAYS relays and NUM_OF_TTLs TTLs in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This function will fill the *OutAllData* structure with these events from the XTC/3 instrument. See User's Guide document for event numbers and description.

Arguments

*OutAllData * val:*

This structure will contain each relay or TTL event (XTC3_NUM_OF_OUTPUTS bytes). See User's Guide document for event numbers and description.

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   OutAllData val;
   unsigned short len = 0;
   getOutAll(&val,  ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("Relay Events are:\n");
      for(int i = 0; i < NUM_OF_RELAYS; i++)
         printf("%d\n",val.m_Data[i]);
      printf("TTL Events are:\n");
      for(; i < XTC3_NUM_OF_OUTPUTS ; i++)
         printf("%d\n",val.m_Data[i]);
   }
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.21 setOutParameter

int ***setOutParameter*** *(unsigned char *val, int index , char* ErrMsg, BOOL TCP);*

Description

There are NUM_OF_RELAYS relays and NUM_OF_TTLs TTLs in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This function will update a Relay or TTL to a certain event. See User's Guide document for event numbers and description.

Arguments

*unsigned char *val:*

The value of the event. See User's Guide document for event number.

int index:

The relay or TTL number (1 byte, 1 to XTC3_NUM_OF_OUTPUTS)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned char val = 31;
   setOutParameter (&val, 5, ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();


}
```

1.3.2.22 getOutParameter

*int **getOutParameter** (unsigned char *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

There are NUM_OF_RELAYS relays and NUM_OF_TTLs TTLs in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This function will get a Relay or TTL event. See User's Guide document for event numbers and description.

Arguments

*unsigned char *val:*

This byte will be filled with the value of the event at a relay or TTL from the XTC/3 instrument. See User's Guide document for event number.

int index:

The relay or TTL number (1 byte, 1 to XTC3_NUM_OF_OUTPUTS)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned char val = 0;
   unsigned short len = 0;
   getOutParameter (&val, 5, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("The value of Output %d is %d\n",5,val);
      printf("Data Transfer successful\n");
   }
   else
      printf(ErrStr);
   CloseSocket();


}
```

IPN 074-454-P1B

1.3.2.23 setInAll

int ***setInAll*** *(InAllData *val, char* ErrMsg, BOOL TCP);*

Description

There are XTC3_NUM_OF_INPUTS inputs in the XTC/3 instrument. Each input can be programmed to react to a certain event. This function programs the inputs via *InAllData.*

See the User's Guide document for event numbers and descriptions.

Arguments

*InAllData *val:*

Fill this structure with events for each Input (XTC3_NUM_OF_INPUTS bytes).

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   InAllData  val;
   for(int i = 0; i < XTC3_NUM_OF_INPUTS ; i++)
      val.m_Data[i] = i + 1;
   setInAll (&val, ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.24 getInAll

int ***getInAll*** *(InAllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

There are XTC3_NUM_OF_INPUTS Inputs in the XTC/3 instrument. Each Input can be programmed to react to a certain event. This function will fill the *InAllData* structure with this event from the XTC/3 instrument. See the User's Guide document for event numbers and descriptions.

Arguments

*InAllData * val:*

This structure will contain each Input event (XTC3_NUM_OF_INPUTS bytes).

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
   printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   InAllData val;
   unsigned short len = 0;
   getInAll(&val,  ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("Input Events are:\n");
      for(int i = 0; i < XTC3_NUM_OF_INPUTS ; i++)
         printf("%d\n",val.m_Data[i]);
   }
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.25 setInParameter

int ***setInParameter*** *(unsigned char *val, int index , char* ErrMsg, BOOL TCP);*

Description

There are XTC3_NUM_OF_INPUTS Inputs in the XTC/3 instrument. Each Input can be programmed to react to a certain event. This function will update an Input to a certain event. See the User's Guide document for event numbers and descriptions.

Arguments

*unsigned char *val:*

The value of the event.

int index:

The Input number (1 byte, 1 - XTC3_NUM_OF_INPUTS)

char* ErrMsg

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned char val = 11;
   setInParameter (&val, 5, ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

1.3.2.26 getInParameter

*int **getInParameter** (unsigned char *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

There are XTC3_NUM_OF_INPUTS Inputs in the XTC/3 instrument. Each relay and TTL can be programmed to react to a certain event. This function will get an Input event. See the User's Guide document for event numbers and descriptions.

Arguments

*unsigned char *val:*

This byte will be filled with the value of the event at an Input from the XTC/3 instrument.

int index:

The Input number (1 byte, 1 to XTC3_NUM_OF_INPUTS)

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned char val = 0;
   unsigned short len = 0;
   getInParameter (&val, 5, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("The value of Input %d is %d\n",5,val);
      printf("Data Transfer successful\n");
   }
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.27 getAll

*int **getAll** (XTC3AllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function is used for downloading all the instrument parameters, which are:

- **General** = XTC3_MAX_GEN_PARAM x 4 bytes
- **Film** = NUM_OF_FILMS x MAX_FILM_PARAM x 4 bytes
- **IOMap** = (NUM_OF_RELAYS + NUM_OF_TTLs + XTC3_NUM_OF_INPUTS) bytes
- All **Film** Names (XTC/3M only): NUM_OF_FILMS x (up to XTC3_TITLE_LENGTH, null terminator included)
- Up to PART_NUM_OF_PROCESSES **Processes** filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x PART_NUM_OF_PROCESSES)

NOTE: See command UB1 and QB1 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

XTC3AllData val:*

This structure is filled with all the instrument parameters (see description above) which amounts to a maximum of XTC3_ALL_SIZE bytes. The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in the array *val->m_Data*.

char ErrMsg*

*unsigned short *len*

BOOL TCP

IPN 074-454-P1B

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3AllData val;
   unsigned short len = 0;
   getAll (&val, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.28 setAll

int ***setAll*** *(XTC3AllData *val, char* ErrMsg, BOOL TCP);*

Description

This function is used for uploading all the instrument parameters, which are:

- **General** = XTC3_MAX_GEN_PARAM x 4 bytes
- **Film** = NUM_OF_FILMS x MAX_FILM_PARAM x 4 bytes
- **IOMap** = (NUM_OF_RELAYS + NUM_OF_TTLs + XTC3_NUM_OF_INPUTS) bytes
- All **Film** Names (XTC/3M only): NUM_OF_FILMS x (up to XTC3_TITLE_LENGTH, null terminator included)
- Up to PART_NUM_OF_PROCESSES **Processes** filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x PART_NUM_OF_PROCESSES)

NOTE: See command UB1 and QB1 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

*XTC3AllData *val:*

This structure is first filled with all the instrument parameters (see description above) which amounts to a maximum of XTC3_ALL_SIZE bytes.

The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in array *val->m_Data*.

char ErrMsg*

BOOL TCP

Example

```
// This example will send all Parameters to the XTC/3
// instrument. For simplicity, it will first do a getAll
// (see above) and, once the structure is filled, it will
// change a couple of parameters and send it through.
// Note that this example uses serial communications.

#include "XTC3Lib.h"
void main()
{
   SetXTC3Port(1);
   SetBaud(115200);
   SetTimeOut(5000);
   if(TC_E_SUCCESS != XTC3Open())
   {
   printf("Cannot open port\n");
```

```
return;
}
char ErrStr[256];
memset(ErrStr,0,256);
unsigned char val = 0;
XTC3AllData val;
unsigned short len = 0;
int temp = 1;
getAll (&val, ErrStr, &len, TRUE);
if(*ErrStr == 0)
{
   // change the 2nd and third General Parameters,
   // 'Start Without Backup' and 'Stop on Alarms',
   // to YES.
   memcpy(val.m_Data + 4, &temp, 4);
   memcpy(val.m_Data + 8, &temp, 4);
   setAll(&val, ErrStr, FALSE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
}
else
   printf(ErrStr);
XTC3Close();

}
```

IPN 074-454-P1B

1.3.2.29 getMinProc

*int **getMinProc** (XTC3AllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function is used for downloading all the instrument parameters without process layer lists and process names, which are:

- **General** = XTC3_MAX_GEN_PARAM x 4 bytes
- **Film** = NUM_OF_FILMS x MAX_FILM_PARAM x 4 bytes
- **IOMap** = (NUM_OF_RELAYS + NUM_OF_TTLs + XTC3_NUM_OF_INPUTS) bytes
- All **Film** Names (XTC/3M only): NUM_OF_FILMS x (up to XTC3_TITLE_LENGTH, null terminator included)

NOTE: See command UB2 and QB2 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

XTC3AllData val:*

This structure is filled with the instrument parameters as described above, which amounts to a maximum of XTC3_ALL_SIZE bytes. The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in the array *val->m_Data*.

char ErrMsg*

*unsigned short *len*

BOOL TCP

IPN 074-454-P1B

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3AllData val;
   unsigned short len = 0;
   getMinProc(&val, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.30 setMinProc

int ***setMinProc*** *(XTC3AllData *val, char* ErrMsg, BOOL TCP);*

Description

This function is used for uploading all the instrument parameters without process layer lists and process names, which are:

- **General** = XTC3_MAX_GEN_PARAM x 4 bytes
- **Film** = NUM_OF_FILMS x MAX_FILM_PARAM x 4 bytes
- **IOMap** = (NUM_OF_RELAYS + NUM_OF_TTLs + XTC3_NUM_OF_INPUTS) bytes
- All **Film** Names (XTC/3M only): NUM_OF_FILMS x (up to XTC3_TITLE_LENGTH, null terminator included)

NOTE: See command UB2 and QB2 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

*XTC3AllData*val:*

This structure is first filled with the instrument parameters as described above, which amounts to a maximum of XTC3_ALL_SIZE bytes.

The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in array *val->m_Data*.

char ErrMsg*

BOOL TCP

Example

```
// This example will send all Parameters except the process layer
// list to the XTC/3 instrument. For simplicity, it will first do a
// getMinProc (see above) and, once the structure is filled, it will
// change a couple of parameters and send it through.
// Note that this example uses serial communications.

#include "XTC3Lib.h"
void main()
{
   SetXTC3Port(1);
   SetBaud(115200);
   SetTimeOut(5000);
   if(TC_E_SUCCESS != XTC3Open())
   {
   printf("Cannot open port\n");
   return;
   }
   char ErrStr[256];
   memset(ErrStr,0,256);
```

```
unsigned char val = 0;
XTC3AllData val;
unsigned short len = 0;
int temp = 1;
getMinProc (&val, ErrStr, &len, TRUE);
if(*ErrStr == 0)
{
   // change the 2nd and third General Parameters,
   // 'Start Without Backup' and 'Stop on Alarms',
   // to YES.
   memcpy(val.m_Data + 4, &temp, 4);
   memcpy(val.m_Data + 8, &temp, 4);
   setMinProc(&val, ErrStr, FALSE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
}
else
   printf(ErrStr);
XTC3Close();
```

IPN 074-454-P1B

1.3.2.31 getLower50ProcInfo

*int **getLower50ProcInfo** (XTC3AllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function is used for downloading process information from the instrument (XTC/3M only) for process 1 to 50 only, which are:

- **Processes** 1 to 50 filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x 50, i.e., 2 byte length for the number of layers programmed, 1 byte for each programmed layer, and process names for **Processes** 1 to 50

NOTE: See command UB3 and QB3 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

XTC3AllData val:*

This structure is filled with the instrument parameters as described above, which amounts to a maximum of XTC3_ALL_SIZE bytes. The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in the array *val->m_Data*.

char ErrMsg*

*unsigned short *len*

BOOL TCP

IPN 074-454-P1B

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3AllData val;
   unsigned short len = 0;
   getLower50ProcInfo(&val, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.32 setLower50ProcInfo

int ***setLower50ProcInfo*** *(XTC3AllData *val, char* ErrMsg, BOOL TCP);*

Description

This function is used for uploading process information from the instrument (XTC/3M only) for process 1 to 50 only, which are:

- **Processes** 1 to 50 filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x 50, i.e., 2 byte length for the number of layers programmed, 1 Byte for each programmed layer, and process names for **Processes** 1 to 50

NOTE: See command UB3 and QB3 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

*XTC3AllData*val:*

This structure is first filled with all the instrument parameters (see description above) which amounts to a maximum of XTC3_ALL_SIZE bytes.

The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in array *val->m_Data*.

char ErrMsg*

BOOL TCP

Example

```
// This example will send process 1 to 50 parameters to the XTC/3
// instrument. For simplicity, it will first do a getLower50ProcInfo
// (see above)and, once the structure is filled, it will change a
// couple of parameters and send it through. You might get an error
// message if the data changed is not consistent with XTC/3 current
// process information.
// Note that this example uses serial communications.

#include "XTC3Lib.h"
void main()
{
   SetXTC3Port(1);
   SetBaud(115200);
   SetTimeOut(5000);
   if(TC_E_SUCCESS != XTC3Open())
   {
   printf("Cannot open port\n");
   return;
   }
   char ErrStr[256];
   memset(ErrStr,0,256);
```

```
unsigned char val = 0;
XTC3AllData val;
unsigned short len = 0;
getLower50ProcInfo (&val, ErrStr, &len, TRUE);
if(*ErrStr == 0)
{
   // change one of the programmed layer
   val.m_Data[3] = 5;
   setLower50ProcInfo(&val, ErrStr, FALSE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
}
else
   printf(ErrStr);
XTC3Close();
```

IPN 074-454-P1B

1.3.2.33 getHigher50ProcInfo

*int **getHigher50ProcInfo** (XTC3AllData *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function is used for downloading process information from the instrument (XTC/3M only) for process 51 to 99 only, which are:

- **Processes** 51 to 99 filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x 49, i.e., 2 byte length for the number of layers programmed, 1 byte for each programmed layer, and process names for **Processes** 51 to 99

NOTE: See command UB4 and QB4 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

XTC3AllData val:*

This structure is filled with the instrument parameters as described above, which amounts to a maximum of XTC3_ALL_SIZE bytes. The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in the array *val->m_Data*.

char ErrMsg*

*unsigned short *len*

BOOL TCP

IPN 074-454-P1B

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   XTC3AllData val;
   unsigned short len = 0;
   getHigher50ProcInfo(&val, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();

}
```

IPN 074-454-P1B

1.3.2.34 setHigher50ProcInfo

int ***setHigher50ProcInfo*** *(XTC3AllData *val, char* ErrMsg, BOOL TCP);*

Description

This function is used for uploading process information from the instrument (XTC/3M only) for process 51 to 99 only, which are:

- **Processes** 51 to 99 filled each with (NUM_OF_LAYERS + XTC3_TITLE_LENGTH + 2) x 49, i.e., 2 byte length for the number of layers programmed, 1 Byte for each programmed layer, and process names for **Processes** 51 to 99

NOTE: See command UB4 and QB4 in the User's Guide document.

The whole packet amounts to a maximum of XTC3_ALL_SIZE bytes.

Arguments

*XTC3AllData*val:*

This structure is first filled with the instrument parameters as described above, which amounts to a maximum of XTC3_ALL_SIZE bytes.

The variable *m_Length* is a 2 byte short that represents the length of data being marshaled in array *val->m_Data*.

char ErrMsg*

BOOL TCP

Example

```
// This example will send process 51 to 99 parameters to the XTC/3
// instrument. For simplicity, it will first do a
// getHigher50ProcInfo(see above)and, once the structure is filled,
// it will change a couple of parameters and send it through. You
// might get an error message if the data changed is not consistent
// with XTC/3 current process information.
// Note that this example uses serial communications.

#include "XTC3Lib.h"
void main()
{
   SetXTC3Port(1);
   SetBaud(115200);
   SetTimeOut(5000);
   if(TC_E_SUCCESS != XTC3Open())
   {
   printf("Cannot open port\n");
   return;
   }
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned char val = 0;
   XTC3AllData val;
```

```
   unsigned short len = 0;
   getHigher50ProcInfo (&val, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      // change one of the programmed layer
      val.m_Data[3] = 5;
      setHigher50ProcInfo(&val, ErrStr, FALSE);
      if(*ErrStr == 0)
         printf("Data Transfer successful\n");
      else
         printf(ErrStr);
   }
   else
      printf(ErrStr);
   XTC3Close();
}
```

IPN 074-454-P1B

1.3.2.35 STATUS Functions

All of the STATUS Functions have the same arguments - except for the first argument - which can be a float, unsigned long, unsigned short or a structure as described in Table 1-2. Three examples are shown after the table.

NOTE: The values contained in the Command ID column are only a reference to the raw commands described in the User's Guide. For example, *getProcInfo,* which has a Command ID of **0**, refers to S**0** in the User's Guide.

Table 1-2 Status Functions

Command ID	Description	Function
0	Process Information	int **getProcInfo** (ProcInfoData *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);
1	Current Rate	int **getCurrentRate** (float *val, char* ErrMsg, unsigned short *len, BOOL TCP);
2	Current Power	int **getCurrentPower** (float *val, char* ErrMsg, unsigned short *len, BOOL TCP);
3	Current Thickness	int **getCurrentThick** (float *val, char* ErrMsg, unsigned short *len, BOOL TCP);
4	Current State	int **getCurrentState** (long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
5	Current State Time	int **getCurrentStateTime** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
6	Active Layer	int **getActiveLayer** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
7	Active Film	int **getActiveFilm** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
8	Active Sensor	int **getActiveSens** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
9	Crystal Life	int **getXtalLife** (unsigned char *val, char* ErrMsg, unsigned short *len, BOOL TCP);
10	Power Source	int **getPwrSrce** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
11	Output Status Byte	int **getOutStatByte** (unsigned short *val, char* ErrMsg, unsigned short *len, BOOL TCP);
12	Input Status Byte	int **getInStatByte** (unsigned short *val, char* ErrMsg, unsigned short *len, BOOL TCP);
13	Raw Frequency	int **getRawFreq** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);

Table 1-2 (continued)Status Functions

Command ID	Description	Function
14	Xtal Fail	int **getXtalFail** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
15	Max Power	int **getMaxPwr** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
16	Crystal Switching	int **getXtalSwitching** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
17	End of process	int **getEndOfProc** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
18	Stop	int **getStopStatus** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
19	Data Log	int **getDataLog** (DataLogStruct *val, char* ErrMsg, unsigned short *len, BOOL TCP);
20	Active Process	int **getActiveProc** (unsigned char *val, char* ErrMsg, unsigned short *len, BOOL TCP);
21	Power up error flag	int **getPowerUpErr** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
30	Crystal Status	int **getXtalStat** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
31	Rolling Average	int **getRollingAvg** (float *val, char* ErrMsg, unsigned short *len, BOOL TCP);
32	Active Crystal	int **getActiveXtal** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
33	Status Messages	int **getStatusMessages** (unsigned long *val, char* ErrMsg, unsigned short *len, BOOL TCP);
34	Raw Rate	int **getRawRate** (float *val, char* ErrMsg, unsigned short *len, BOOL TCP);
35	Ethernet Parameters	int **getEthernetParam** (long64 *val, char* ErrMsg, unsigned short *len, BOOL TCP);

IPN 074-454-P1B

1.3.2.35.1 getProcInfo

*int **getProcInfo** (ProcInfoData *val, int index, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function gets the following status, sequentially, in the structure *ProcInfoData.* It is essentially used to monitor the crystal state.

Sensor status . . . . . 1 byte

Bit

0 . . . . . 0 = Good crystal, 1 = Failed crystal

1 . . . . . 0 = Not switching, 1 = Switching crystal

2 . . . . . 0 = Good reading, 1 = Invalid reading

State . . . . . 1 byte

Active Sensor . . . . . 1 byte

Active Crystal . . . . . 1 byte

Rate . . . . . Float (One second average)

Thickness . . . . . Float

Power . . . . . Float

Rate Deviation . . . . . Float

Frequency. . . . . int

Arguments

*ProcInfoData *val:*

This structure is first filled with all the data mentioned above in the order described.

int index:

Since this function will be used for monitoring data vs. time, this command (The S0 command, see the User's Guide document) needs to have a timer tick (1 byte, 0 to 255) tagged on to it. For example, tagging a 5 tells the XTC/3 system to send data acquired at timer tick 5.

char ErrMsg*

*unsigned short *len*

BOOL TCP

IPN 074-454-P1B

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   ProcInfoData  val;
   unsigned short len = 0;
   getProcInfo(&val, 1, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      int freq = 0;
      memcpy(&freq,val.m_Data + 5, 4);
      printf("The frequency at this time is: %2.6f\n",
      0.0034924596 * freq);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.35.2 getDataLog

*int **getDataLog** (DataLogStruct *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Description

This function provides the DataLog status as described below.

Arguments

*DataLogStruct *val:*

A 40 byte structure where the data will be marshaled in the following order:

int	Layer #
int	Film #
float	Rate (Å/Sec) [One second average]
float	Thickness (KÅ)
int	Deposit time in seconds
float	Average power %
int	S Value
byte	Q Value
float	Begin Frequency HZ
float	End Frequency HZ
byte	Crystal Life %
byte	Normal end (0), Time power end (1) or Stop (2)
byte	Cause of stop 0 = No stop 1 = Keyboard 2 = Time Power 3 = Crucible Error 4 = Shutter Delay Error 5 = Crystal Fail 6 = Max Power 7 = Hand Controller 8 = Communications 9 = Digital Input 10 = Power Loss 11 = Rate Dev Error 12 = Switcher fail

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   DataLogStruct  val;
   unsigned short len = 0;
   getDataLog (&val, ErrStr,&len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      float DT = 0;
      memcpy(&DT,val.m_Data + 16, 4);
      printf("The Deposit Time at this time is: %d\n", DT);
   }
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.35.3 getCurrentState

int ***getCurrentState*** *(long *val, char* ErrMsg, unsigned short *len, BOOL TCP);*

Arguments

*long *val:*

This 4 byte float will contain the Current State return by the XTC/3 instrument.

char ErrMsg*

*unsigned short *len*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   long currentState = 0;
   unsigned short len = 0;
   getCurrentState (&currentState, ErrStr, &len, TRUE);
   if(*ErrStr == 0)
   {
      printf("Data Transfer successful\n");
      printf("The Current State  is: %d\n", currentState);
   }
   else
      printf(ErrStr);
   CloseSocket();

}
```

1.3.2.36 REMOTE Functions

The Remote functions are noted in Table 1-3.

NOTE: The values contained in the Command ID column are only a reference to the raw commands described in the User's Guide. For example, *setRemLockOff*, which has a Command ID of **4,** refers to R**4** in the User's Guide.

Table 1-3 REMOTE Functions

Command ID	Description	Function
0	Start	int **setRemoteStart** (char* ErrMsg, BOOL TCP);
1	Stop	int **setRemoteStop** (char* ErrMsg, BOOL TCP);
2	Reset	int **setRemoteReset** (char* ErrMsg, BOOL TCP);
3	Remote lock on	int **setRemLockOn** (char* ErrMsg, BOOL TCP);
4	Remote lock off	int **setRemLockOff** (char* ErrMsg, BOOL TCP);
5	Crystal fail inhibit on	int **setRemXtalfailInhOn** (char* ErrMsg, BOOL TCP);
6	Crystal fail inhibit off	int **setRemXtalfailInhOff** (char* ErrMsg, BOOL TCP);
7	Soak hold 2 on	int **setRemSoakHold2ON** (char* ErrMsg, BOOL TCP);
8	Soak hold 2 off	int **setRemSoakHold2OFF** (char* ErrMsg, BOOL TCP);
9	Manual on	int **setRemManualON** (char* ErrMsg, BOOL TCP);
10	Manual off	int **setRemManualOFF** (char* ErrMsg, BOOL TCP);
11	Set power vv	int **setRemSetPwrVV** (float val, char* ErrMsg, BOOL TCP);
12	Zero thickness	int **setRemZeroThick** (char* ErrMsg, BOOL TCP);
13	Final thickness trigger	int **setRemFinThickTrig** (char* ErrMsg, BOOL TCP);
14	Crystal switch	int **setRemXtalSwitch** (char* ErrMsg, BOOL TCP);
15	Enter comm. I/O mode XTC/3S ONLY!	int **setRemEnterComm** (char* ErrMsg, BOOL TCP);
16	Exit comm. I/O mode XTC/3S ONLY!	int **setRemExitComm** (char* ErrMsg, BOOL TCP);
17	Set (close) digital output vv	int **setRemSetDigitOutVV** (unsigned char val, char* ErrMsg, BOOL TCP);
18	Clear (open) digital output vv	int **setRemClearDigitOutVV** (unsigned char val, char* ErrMsg, BOOL TCP);
19	Turn Backlight on	int **setRemTurnBacklightOn** (char* ErrMsg, BOOL TCP);
20	Turn Backlight off	int **setRemTurnBacklightOff** (char* ErrMsg, BOOL TCP);
21	Trigger beeper	int **setRemTriggerBeeper** (char* ErrMsg, BOOL TCP);

IPN 074-454-P1B

Table 1-3 REMOTE Functions

Command ID	Description	Function
22	Clear power up error flag	int **setRemClearPwrUpFlag** (char* ErrMsg, BOOL TCP);
26	Clear All Crystals	int **setRemClearAllXtals** (char* ErrMsg, BOOL TCP);
27	Rotate Head	int **setRemRotateHead** (char* ErrMsg, BOOL TCP);
28	Clear S and Q Counts	int **setRemClearSQCount** (char* ErrMsg, BOOL TCP);

All of the Functions listed in the table above have the same arguments, except for three functions, which have an additional unsigned char or float argument. Three examples for these functions are shown on the following pages.

1.3.2.36.1 setRemTurnBacklightOff

*int **setRemTurnBacklightOff** (char* ErrMsg, BOOL TCP);*

Arguments

char* ErrMsg

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   setRemTurnBacklightOff( ErrStr, TRUE);
   if(*ErrStr == 0)
   {
      Sleep(1000);
      setRemTurnBacklightOn( ErrStr, TRUE);
      if(*ErrStr == 0)
         printf("Data Transfer successful\n");
      else
         printf(ErrStr);
      }
      else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.36.2 setRemSetPwrVV

int ***setRemSetPwrVV*** *(float val, char* ErrMsg, BOOL TCP);*

Arguments

float val:

4 byte float that sets the % power for the active layer.

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   setRemSetPwrVV (.045F, ErrStr, TRUE); // setting the power for
   // the active layer to 0.045%
   setRemTurnBacklightOff( ErrStr, TRUE);
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();
}
```

1.3.2.36.3 setRemSetDigitOutVV

*int **setRemSetDigitOutVV** (unsigned char val, char* ErrMsg, BOOL TCP);*

Arguments

unsigned char *val*:

1 byte float that sets the relay#

char ErrMsg*

BOOL TCP

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   setRemSetDigitOutVV (1,  ErrStr, TRUE); // sets relay# to 1
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B

1.3.2.37 Macro Function

The Macro function is used to send a macro command string to the XTC/3 Controller and to get the response returned by the XTC/3 controller.

int ***getSendMacro****(XTC3AllData * returnVal, char * ErrMsg, char commandStr[], int commandStrlen, BOOL TCP);*

Arguments

*XTC3AllData*returnVal:*

This structure is filled with the response returned by the XTC/3 instrument (see description below), which amounts to a maximum of XTC3_ALL_SIZE bytes.

char commandStr[]:

The Macro command string, tagged in the format described below.

int commandStrLen:

The length of the Macro command string, limited by the maximum 60000 bytes.

char ErrMsg*

BOOL TCP

1.3.2.37.1 commandStr[ ]

The Macro function allows individual commands to be connected together (without spacing) in the ***commandStr[ ]*** parameter, using the following format:

ASCII Command <1 Byte Value>(2 Byte Value)[4 Byte Value]{String}

ASCII Command:

ASCII letter of commands such as H, E, QG, S.

<1 Byte Value>:

1 Byte Value, such as Command ID, Process Number and Film Number. A 1 Byte Value must be enclosed in the angle-type < > brackets if followed by another 1 Byte Value. The angle-type bracket is not required when using a single 1 Byte Value followed by an ASCII Command.

(2 Byte Value):

2 Byte values, such as Number of layers. 2 Byte values must be enclosed in parentheses ( ).

[4 Byte Value]:

4 Byte values, such as parameter values in integer or float format. 4 Byte values must be enclosed in the square-type [ ] brackets.

IPN 074-454-P1B

{String}:

String values such as the echo string, the film name and the process name. String values must be enclosed in the curly-type { } brackets.

The byte values, the strings and their sequences depend on the commands that are tagged together. The following is a list of examples:

Example 1: H<1>H<2>QG<0>, or H1H2QG0

Example 2: E{Hello}

Example 3: QP3<1>(1)S4

Note the second 1 Byte number has to be enclosed in < > bracket in order to be distinquished from the first unbracketed 1 Byte number.

Example 4: UF<0><1>[120]

Example

```
#include "XTC3Lib.h"
void main()
{
   if(StartSocket()==0)
      printf("Socket could not be initialized.");
   if(!ConnectSocket("10.211.70.209"))
      printf("Socket could not be connected!");
   char ErrStr[256];
   memset(ErrStr,0,256);
   unsigned short len = 0;
   char commandStr[] = "QG1"
   XTC3AllData val
   getSendMacro (&val,ErrStr,commandStr,strlen(commandStr),TRUE);
   // sets relay# to 1
   if(*ErrStr == 0)
      printf("Data Transfer successful\n");
   else
      printf(ErrStr);
   CloseSocket();
}
```

IPN 074-454-P1B