

Bayer Contour Next USB Programming Guide

CIS-DDCUS-00036-Third-Party

Version 003

October 2, 2014

Author: Bayer

Bayer Responsibility: See EPN

!!! CAUTION !!!

This document provides specifications that **MUST** be followed for safety reasons. All provisions of this specification must be followed. It is the sole responsibility of the implementer to insure that its code is thoroughly tested, before using or publishing any product based upon this specification. This document is PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. The entire risk arising out of the use or performance of this Programming Guide remains with Licensee. In no event shall BAYER be liable for any damage whatsoever arising out of the use of or inability to use this Programming Guide, even if BAYER has been advised of the possibility of such damages.

Licensee agrees to waive all claims against BAYER and to defend and indemnify BAYER and hold BAYER harmless for all claims and damages arising from Licensee's access to or use of this Programming Guide.

Particular attention should be paid to **Section 5.1, "Mutual Exclusive Access"** to insure that only a single client application has access to the Bayer Contour Next USB at a time.

Confidentiality and Non Disclosure

This document contains confidential information of Bayer HealthCare LLC. The recipient of this document agrees that the information received from Bayer, this document and/or the information therein contained, in whole or in part herein, is the sole property of Bayer and includes valuable trade secrets of Bayer. Recipient agrees to treat material as confidential and will not: reproduce, copy, transmit, demonstrate, sell or market information; or publish or otherwise disclose information relating to performance or quality of this material to any third party; or modify, reuse, disassemble, decompile, reverse engineer or otherwise translate any portion of the material and/or the information therein contained, in whole or in part, or to suffer such action(s) by others, for any purpose, except with the written permission first obtained of Bayer HealthCare LLC and further agrees to surrender the same to Bayer HealthCare LLC upon demand.

Revision History

Revision	Description	Author	Date
000	Initial Draft	Bayer	Dec 01 2011
001	Changes from PR-DDCUS-00036-CIS-Third-Party	Bayer	Dec 15 2011
002	Append license agreement in last page	Bayer	Oct 11 2013
003	Update document footer per Legal	Bayer	Oct 2 2014

Table of Contents

1	SCOPE	5
2	LIST OF ACRONYMS	5
3	PHYSICAL AND VIRTUAL CONNECTIONS	6
3.1	CONTOUR NEXT USB HID	6
3.1.1	<i>Interface</i>	6
3.1.2	<i>Report Formats</i>	6
3.2	CLIENT (HOST) HID COMMUNICATIONS	7
3.2.1	<i>Tools</i>	7
3.2.2	<i>Setting Up</i>	7
3.2.3	<i>Opening a Connection</i>	10
3.2.4	<i>Sending Data</i>	11
3.2.5	<i>Receiving Data</i>	11
3.2.6	<i>Closing a Connection</i>	11
3.2.7	<i>Client Code Example: Identifying a Contour Next USB Meter</i>	12
4	COMMUNICATION PROTOCOL	16
4.1	OVERVIEW	16
4.2	HALF DUPLEX OPERATION	16
4.3	ASCII CONTROL CHARACTER NOTATION	16
4.4	ASTM STANDARD	17
4.4.1	<i>Frame Structure</i>	17
4.4.2	<i>ASTM E-1394 Records</i>	19
4.5	DATA TRANSFER AND REMOTE COMMAND MODES	27
4.5.1	<i>Data Transfer Mode</i>	27
4.5.2	<i>Remote Command Mode</i>	28
5	APPENDIX	32
5.1	MUTUAL EXCLUSIVE ACCESS	32
5.2	COMMUNICATION HINTS	32
5.2.1	<i>Determine Presence and Type of Meter</i>	32
5.2.2	<i>Configure Meter without Receiving Test Results</i>	32
5.3	EXAMPLES	33
5.3.1	<i>Header Record</i>	33
5.3.2	<i>Patient Record</i>	34
5.3.3	<i>Result Record</i>	35
5.3.4	<i>Message Terminator Record</i>	36
5.3.5	<i>N Command Sequence</i>	37
5.3.6	<i>Write Date and Time to Meter</i>	38

List of Tables

Table 1: ASCII Control Characters	16
Table 2: Header Record	20
Table 3: Patient Information Record	23
Table 4: Result Record	23
Table 5: Message Terminator Record	26
Table 6: Address Tokens for Remote Command Mode	29
Table 7: Example Header Record	33
Table 8: Example Patient Record	34
Table 9: Glucose Example Result Record	35
Table 10: Carbs Example Result Record	35
Table 11: Insulin Example Result Record	35
Table 12: Glucose "LO" Example Result Record	35
Table 13: Glucose No Markers Example Result Record	36
Table 14: Glucose Self Detected Control Example Result Record	36
Table 15: Glucose Plasma and mmol/L Example Result Record	36
Table 16: Glucose Absolute High Example Result Record	36
Table 17: Example Terminator Record	36

1 Scope

This document describes the computer interface of the Bayer Contour Next USB Blood Glucose meter. This document contains technical information that allows developers to write computer programs that communicate with the Contour Next USB meter. It describes the communication protocol and record formats for transmitting test results and for configuring features such as date & time on the meter, through its computer interface.

2 List of Acronyms

Acronym	Meaning
API	Application Programming Interface.
ASCII	American Standard Code for Information Interchange is a character encoding based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. Most modern character encodings — which support many more characters than did the original — have a historical basis in ASCII.
ASTM	ASTM International (ASTM) , originally known as the American Society for Testing and Materials, is an international standards organization that develops and publishes voluntary consensus technical standards for a wide range of materials, products, systems, and services.
HID	Human Device Interface. a type of computer device that interacts directly with, and most often takes input from, humans and may deliver output to humans. The term "HID" most commonly refers to the USB-HID specification.
PC	Personal Computer - a computer whose original sales price, size, and capabilities make it useful for individuals, and which is intended to be operated directly by an end user, with no intervening computer operator. A PC may be a desktop computer, a laptop computer or a tablet computer. The most common operating systems are Microsoft Windows, Mac OS X and Linux, while the most common microprocessors are x86 compatible CPUs. However, the term "PC" is often used only to refer to computers.
UOM	Unit of Measure. Refers to the method used to measure blood glucose concentration in the blood. This is either mg/dL (milligrams per deciliter) or mmol/L (millimoles per liter).
USB	Universal Serial Bus (USB) is a serial bus standard to interface devices. USB was designed to allow many peripherals to be connected using a single standardized interface socket and to improve the plug-and-play capabilities by allowing devices to be connected and disconnected without rebooting the computer (hot swapping).

3 Physical and Virtual Connections

The physical connection utilizes the USB HID protocol. This section provides detailed information on the HID interface in Contour Next USB.

3.1 Contour Next USB HID

3.1.1 Interface

The Contour Next USB Meter communicates with the host application using HID **Interrupt** transfers, using endpoint **0x01** for interrupt OUT and endpoint **0x81** for interrupt IN.¹ Endpoint 0x00 is a Control Pipe, and is always present in USB devices.

3.1.2 Report Formats

Report formats for HID Input Reports and HID Output Reports are 65 bytes in length. They are both identical, and follow the format illustrated below.

```
struct BAYER_CONTOUR_REPORT
{
    unsigned char reportID;    // HID report ID
    unsigned char checksum;    // checksum for hostID + deviceID + data (see footnote 2)
    unsigned char hostID;     // host ID assigned by communications manager
    unsigned char deviceID;    // device ID assigned by communications manager
    unsigned char length;     // length of data in buffer
    unsigned char data[60];    // data send with message
};
```

3.1.2.1 USB Frame Fields

Field	Description
reportID	This field is populated with the Contour Next USB Report ID during transmission. The Windows [™] HID drivers take care of initializing this field. It should be treated as read-only by the client.
checksum	This field contains a check sum value calculated on the contents of the hostID, deviceID and data fields.
hostID	This is a numeric value assigned to the message by communications management software in the host. The device should maintain this value in reply packets sent to the host.
deviceID	This is a numeric value assigned to the message by communications management software in the host. The device should maintain this value in reply packets sent to the host.
data	This field contains the data sent with the message. In this version of Contour Next USB, the data will always contain ASTM-based message text as described in this document under the description of the Communications Protocol.

¹ These endpoints are defined in the meter software.

² This field is currently not used in the Contour Next USB/HID implementation. Checksum calculation follows the same methodology as detailed in section 4.4.1.1 "Frame Checksum".

3.2 Client (Host) HID Communications

This section provides information on how a Windows [™] client communicates with the Contour Next USB meter.

NOTE: All code samples in this document are provided as guidelines only. It is expected that any user of this specification independently implements and thoroughly tests their interface and their own code. While the code samples given here are derived from working implementations, they should not be expected to be bug free or suitable for direct use in the reader's implementations.

3.2.1 Tools

The discussions and illustration in the following sections assume use of the following tools:

- Microsoft Windows DDK Version 3790.1830
- Microsoft Visual C++ Version 6.0

The compiler and linker tools should be setup to locate the necessary headers (.h files) and libraries from the DDK. At the time of this writing, these include the headers: setupapi.h, hidsdi.h and the libraries: winmm.lib hid.lib setupapi.lib.

3.2.2 Setting Up

Under Microsoft Windows [™] the following steps are necessary in order to locate and identify the USB-HID interface for the Contour Next USB Blood Glucose Meter:

- Get the GUID used to identify the HID device class in Windows.
- Retrieve a list of HID devices attached to the system.
- Enumerate the list of HID devices:
 - Get device interface details – this will return a device “path”.
 - Using the device path, get a handle to the device using your OS's “open” API or equivalent.
WARNING: The Contour Next USB should ALWAYS be opened in “exclusive” mode, to insure that only one application is accessing the Contour Next USB at a time. See Section 5.1, “Mutual Exclusive Access”.
 - Using the device handle, get device capabilities.
 - If the device belongs to the proper “Usage Page” (0x40 for Medical Instruments):³
 - Get device capabilities and attributes to retrieve USB Vendor ID and optionally, the Bayer USB Product ID.
 - Cache the device information for subsequent calls.
 - Send the necessary commands to retrieve the meter's serial number
 - Cache the identifying information (meter serial number, etc) for subsequent use

3.2.2.1 Get Windows HID-Class GUID

To retrieve the Windows HID Class GUID using C++, issue the following call:

```
GUID hidGuid;  
HidD_GetHidGuid( &hidGuid );
```

³ For example, under Microsoft Windows, the **HIDP_CAPS.UsagePage** will be filled in by the call to “HidP_GetCaps” API. See your own operating systems programming documentation, for more details.

The variable “hidGuid” will be used later to retrieve information related to this device class.

3.2.2.2 Retrieve List HID Devices

To get a handle representing a list of attached HID devices in C++, issue the following call:

```
HDEVINFO hDevList = SetupDiGetClassDevs(&HidGuid, NULL, NULL, DIGCF_DEVICEINTERFACE
| DIGCF_PRESENT);
if (hDevList == INVALID_HANDLE_VALUE) {
    // error...
}
```

The variable “hDevList” will be used later in device enumeration.

3.2.2.3 Enumerate List of Devices

Setup a block of memory to receive device interface information as illustrated below. The variable “DevData” will be used to retrieve information about HID device interfaces. Then, using a programming loop, where the loop **index** varies ranges from 0 to 126, perform the tasks outline in the following sections (there can be up to 127 USB devices attached to the system):

```
SP_DEVICE_INTERFACE_DATA DevData;
ZeroMemory(&DevData, sizeof(DevData));
DevData.cbSize = sizeof(SP_DEVICE_INTERFACE_DATA);

for ( int i = 0; i < 127; i++ )
{
    if ( SetupDiEnumDeviceInterfaces(hDevList, NULL, &HidGuid, index, &DevData) == 0 )
    {
        if ( GetLastError() == ERROR_NO_MORE_ITEMS )
            break;

        // See info in following sections...
    }
}
```


3.2.2.4 Get Device Interface Details

To get details about the interface, including the device path (to open the device), implement code similar to this:

```
SP_INTERFACE_DEVICE_DETAIL_DATA * pDevDetail = NULL;
DWORD                               ReqLen = 0;
BOOL                                rc = FALSE;
HANDLE                              hDevice = NULL;
//
// Call first with 0 "Required Length" to get the required length...
//
SetupDiGetDeviceInterfaceDetail(hDevList, pDevData, NULL, ReqLen, &ReqLen, NULL);
//
// allocate device details...
//
pDevDetail = (SP_INTERFACE_DEVICE_DETAIL_DATA*)malloc(ReqLen);
if (pDevDetail == NULL )
{
    // error...
}
//
// initialize device details structure...
//
ZeroMemory(pDevDetail, ReqLen);
pDevDetail->cbSize = sizeof(SP_INTERFACE_DEVICE_DETAIL_DATA);

//
// Get the details...
//
rc = SetupDiGetDeviceInterfaceDetail(hDevList, pDevData, pDevDetail, ReqLen, &ReqLen,
NULL);
if ( rc == FALSE)
{
    free(pDevDetail);
    // error...
}

// NOTE: Path to device for subsequent open call is in pDevDetails->DevicePath
```

3.2.2.5 Get Handle to the Device

Once you have the path to the HID device, use it to get a handle by making an operating system call to open the device, similar to this:

```
//
// Open a device handle to the HID device and retrieve HID information
//
HANDLE hDevice = CreateFile(pDevDetail->DevicePath, GENERIC_READ | GENERIC_WRITE,
0, NULL, OPEN_EXISTING, 0, NULL);
if (hDevice == INVALID_HANDLE_VALUE)
{
    // error...
}
```

3.2.2.6 Get Device Capabilities

Using the handle returned on the call to open the device, use code similar to the following to determine if the device is a Contour Next USB Meter:

```
//  
// Parse the information from HID device  
//  
HidD_GetPreparsedData(hDevice, &pPreparsedData);  
HidP_GetCaps(pPreparsedData, &Capabilities);  
//  
// Does the device match the Vendor specific page implementation?  
//  
if ( Capabilities.UsagePage == MEDICAL_DEVICE_PAGE )  
{  
    HIDD_ATTRIBUTES Attributes;  
    memset(&Attributes, 0, sizeof(Attributes));  
    Attributes.Size = sizeof(Attributes);  
  
    HidD_GetAttributes(hDevice, &Attributes);  
  
    if ( Attributes.VendorID == BAYER_VENDOR_ID )  
    {  
        //  
        // It's a Bayer USB device: Cache device information and  
        // proceed to query header to get serial number, etc..  
        //  
    }  
}
```

3.2.3 Opening a Connection

Once you have obtained the necessary device information, you communicate with the Contour Next USB meter using Windows File IO API's. To open a connection with the Contour Next USB meter, in preparation to send or receive data, use code similar to the following:

```
HANDLE hDevice = CreateFile( pDevicePath, GENERIC_READ | GENERIC_WRITE, 0, NULL,  
    OPEN_EXISTING, 0, NULL);  
  
if ( hDevice == INVALID_HANDLE_VALUE )  
{  
    // error...  
}
```

3.2.4 Sending Data

To send data to the Contour Next USB meter, on Windows, use the WriteFile API as shown below⁴. NOTE: The sender should reserve the 1st byte of the buffer. This is used by the Windows HID stack to store the report ID.

```
//
// allocate buffer size
//
U8 * pTmpBuf = (U8 *)calloc( OutputReportByteLength, sizeof(U8));
if (pTmpBuf == NULL)
{
    ;// memory allocation error...
}

//
// copy data to be sent, reserving the 1st byte for report ID
//
unsigned long Written = 0;
memcpy(pTmpBuf + 1, pDataToWrite, NumBytes);
WriteFile(hDevice, pTmpBuf, OutputReportByteLength, &Written, NULL) {
free(pTmpBuf);
```

3.2.5 Receiving Data

To receive data from the Contour Next USB meter, one can use the Windows ReadFile API as shown below.⁵ Note that the receive buffer length should allocate an extra byte at the front of the buffer, to account for the Report ID - the application should treat this byte as read only.

```
pTmpBuf = (U8 *)calloc( InputReportByteLength, sizeof(U8));
if (pTmpBuf == NULL)
{
    return error("memory allocation failed");
}
if (ReadFile( hDevice, pTmpBuf, InputReportByteLength, &nRead, NULL))
{
    memcpy(pAppBuffer, pTmpBuf + 1, nRead-1);
}
free(pTmpBuf);
```

3.2.6 Closing a Connection

To close a connection with the Contour Next USB meter, use the device handle returned on a previous CreateFile call, as shown below:

```
CloseHandle( hDevice );
hDevice = INVALID_HANDLE_VALUE;
```

⁴ Consult your operating systems documentation for equivalent API's.

⁵ ReadFile is a blocking call. Windows overlapped IO can also be used, to avoid blocking in the host application.

3.2.7 Client Code Example: Identifying a Contour Next USB Meter

The following code illustrates the previously defined steps for Contour Next USB HID client communications.

```
//
// ContourSample.cpp
//

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <setupapi.h>

extern "C" {
#include <hidsdi.h>
}

#define MAX_USB_DEVICE 127
#define BAYER_VENDOR_ID 0x1a79
#define BAYER_CONTOUR_PRODUCT_ID 0x???? // NOTE: USB Product ID at USB.ORG
#pragma pack( push, bayer_packet, 1 )

struct BAYER_CONTOUR_REPORT
{
    unsigned char    reportID;    // HID report ID (assigned by the host)
    unsigned char    checksum;    // checksum for hostID + deviceID + data
    unsigned char    hostID;      // host ID assigned by communications manager
    unsigned char    deviceID;    // device ID assigned by communications manager
    unsigned char    data[61];    // data send with message
};

#pragma pack( pop, bayer_packet )
```

```

//
// Sample application class to enumerate HID's find Contour Next USB
// meter, send and receive sample packets.
//
class SampleApp
{
public:

    SampleApp()
    {
        hDevice = INVALID_HANDLE_VALUE;
        memset(devicePath,0,sizeof(devicePath)/sizeof(char));
        outputReportLength = inputReportLength = 0;
    }

    void    run();

protected:
    void    progress( char* msg );
    void    identifyMeter();

private:
    GUID    hidGuid;
    HANDLE  hDevice;
    char    devicePath[512];
    int     outputReportLength;
    int     inputReportLength;
};

void SampleApp::run()
{
    progress( "STEP 1: getting HID Guid..." );
    HidD_GetHidGuid( & hidGuid );

    progress( "STEP 2: get HID device list..." );
    HDEVINFO hDeviceList = SetupDiGetClassDevs( &hidGuid, NULL, NULL, DIGCF_DEVICEINTERFACE |
DIGCF_PRESENT );
    if ( hDeviceList == INVALID_HANDLE_VALUE )
    {
        throw "Unable to get HID class device list";
    }

    progress( "STEP 3: enumerate HID devices..." );
    SP_DEVICE_INTERFACE_DATA DevData;
    ZeroMemory(&DevData,sizeof(DevData));
    DevData.cbSize = sizeof(SP_DEVICE_INTERFACE_DATA);

    BOOL foundContour = FALSE;

    for ( int index = 0; index < MAX_USB_DEVICE && foundContour == FALSE; index++ )
    {
        int result = SetupDiEnumDeviceInterfaces(hDeviceList, NULL, &hidGuid, index, &DevData);
        if ( result == 0 )
        {
            if ( GetLastError()== ERROR_NO_MORE_ITEMS)
                break;
            else
                SetupDiDestroyDeviceInfoList(hDeviceList);
            throw "no HID devices found";
        }
        else
        {
            SP_INTERFACE_DEVICE_DETAIL_DATA * pDevDetail = NULL;

```

```

DWORD ReqLen = 0;
BOOL rc = FALSE;

char msg[256];
sprintf( msg, "\tSTEP 4: get HID details for interface %d", index );
progress( msg );

//
// Call first with 0 "Required Length" to get the required length...
//
SetupDiGetDeviceInterfaceDetail(hDeviceList, &DevData, NULL, ReqLen, &ReqLen, NULL);

//
// Now allocate device details based on known length
//
pDevDetail = (SP_INTERFACE_DEVICE_DETAIL_DATA*)malloc(ReqLen);
if (pDevDetail == NULL )
{
    throw "allocation for device detail failed";
}

//
// initialize device details structure...
//
ZeroMemory(pDevDetail, ReqLen);
pDevDetail->cbSize = sizeof(SP_INTERFACE_DEVICE_DETAIL_DATA);

//
// Get the details...
//
if ( SetupDiGetDeviceInterfaceDetail(hDeviceList, &DevData, pDevDetail, ReqLen,
&ReqLen, NULL) == FALSE )
{
    free(pDevDetail);
    throw "could not get details for device interface";
}

sprintf( msg, "\tSTEP 5: get device handle for interface %d", index );
progress( msg );
//
// Open a device handle to the HID device and retrieve HID information
//
hDevice = CreateFile(pDevDetail->DevicePath, GENERIC_READ|GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
if (hDevice != INVALID_HANDLE_VALUE)
{
    sprintf( msg, "\tSTEP 6: get device capabilities & attributes for interface %d",
index );
    progress( msg );

    //
    // Parse the information from HID device
    //
    PHIDP_PREPARED_DATA pPreparedData;
    HIDP_CAPS Capabilities;
    HidD_GetPreparedData(hDevice, &pPreparedData);
    HidP_GetCaps(pPreparedData, &Capabilities);

    //
    // Does the device match the Vendor specific page implementation?
    //
    if ( Capabilities.UsagePage == MEDICAL_DEVICE_PAGE )
    {

```

```
HIDD_ATTRIBUTES Attributes;
memset(&Attributes, 0, sizeof(Attributes));
Attributes.Size = sizeof(Attributes);

HidD_GetAttributes(hDevice, &Attributes);

// NOTE: The USB connection can be made by matching the BAYER_VENDOR_ID
// and looking at the "Sender ID" field in the Header Record to identify
// a specific Bayer Product. Please refer to section 4.4.2.1 in this document for
// details about the header record and the Bayer "Product ID".

if ( Attributes VendorID == BAYER_VENDOR_ID &&
      Attributes.ProductID == BAYER_CONTOUR_PRODUCT_ID )
{
    //
    // It's a Contour Next USB: Cache device and exit loop
    //
    inputReportLength = Capabilities.InputReportByteLength;
    outputReportLength = Capabilities.OutputReportByteLength;
    strcpy( devicePath, pDevDetail->DevicePath );
    foundContour = TRUE;
}

CloseHandle( hDevice );
hDevice = INVALID_HANDLE_VALUE;
}

}
free( pDevDetail );
pDevDetail = 0;
}
}

if ( foundContour == TRUE )
{
    progress( "FOUND A CONTOUR METER..." );
    identifyMeter();
}

SetupDiDestroyDeviceInfoList(hDeviceList);
}

void SampleApp::progress( char* msg )
{
    fprintf( stdout, msg );
    fputc( '\n', stdout );
}
```

4 Communication Protocol

4.1 Overview

The communications between the Contour Next USB meter and data management software running on a PC, or other client software, is based on the ASTM Standard. The Physical communication between the meter and the client software is accomplished using USB HID protocol, as described in section 3, "Physical and Virtual Connections".

4.2 Half Duplex Operation

The mode of operation of the Contour Next USB meter is half-duplex. The Contour Next USB cannot transmit and receive characters at the same time.

4.3 ASCII Control Character Notation

The communication protocol includes non-printable ASCII (American Standard Code for Information Interchange) control characters. In this document, the convention for displaying control characters uses the notation <XYZ>. For example, <CR> stands for the single Carriage Return control character that is represented by decimal value 13 or hexadecimal value 0D.

Table 1: ASCII Control Characters

ASCII Abbreviation	Meaning	Hex Char Code	Decimal Char Code
<ACK>	<u>A</u> cknowledge	6	6
<CAN>	<u>C</u> ancel	18	24
<CR>	<u>C</u> arriage <u>r</u> eturn	0D	13
<ENQ>	<u>E</u> nquiry	5	5
<EOT>	<u>E</u> nd of transmission	4	4
<ETB>	<u>E</u> nd of transmission <u>b</u> lock	17	23
<ETX>	<u>E</u> nd of <u>t</u> ext	3	3
<LF>	<u>L</u> ine <u>f</u> eed	0A	10
<NAK>	<u>N</u> egative <u>a</u> cknowledge	15	21
<STX>	<u>S</u> tart of <u>t</u> ext	2	2

4.4 ASTM Standard

Communication between Contour Next USB meters and data management systems is carried out using the ASTM-E Standard Specification for Transferring Information between Clinical Instruments and Computer Systems.

4.4.1 Frame Structure

Frames are formatted per ASTM Standard E-1381.

The structure of an E-1381 frame is illustrated as follows:

```
<STX> FN text <ETB> C1 C2 <CR> <LF>
```

or

```
<STX> FN text <ETX> C1 C2 <CR> <LF>
```

where:

<STX>	=	Start of <u>T</u> ext transmission control character
FN	=	Single digit Frame Number ("0" to "7") The FN, <i>frame number</i> , is a single ASCII character in the range from "0" to "7". It permits the receiver to distinguish between new and retransmitted frames. The frame number is sent immediately following the <STX> character. The frame number begins with "1" with the first frame of the transfer phase and is incremented by one for every frame transmitted. After "7" the number rolls over to "0", and continues in this fashion.
text	=	Data content of the frame
<ETB>	=	End of <u>T</u> ransmission <u>B</u> lock transmission control character
<ETX>	=	End of <u>T</u> ext transmission control character
C1	=	Most significant character of encoded checksum ("0" to "9" and "A" to "F")
C2	=	Least significant character of encoded checksum ("0" to "9" and "A" to "F")
<CR>	=	<u>C</u> arriage <u>R</u> eturn ASCII control character
<LF>	=	<u>L</u> ine <u>F</u> eed ASCII control character

A frame is either an *intermediate frame*, or an *end frame*. In *intermediate frames* an <ETB> character follows the message text. In *end frames*, an <ETX> character follows message text. The <ETB> or <ETX> is followed by a two-character checksum, carriage return <CR>, and a line feed <LF>.

C1 and C2 represent the encoded checksum of the frame. The checksum permits the computer to detect defective frames. The checksum is encoded as two ASCII characters transmitted after the <ETB> or <ETX> character.

4.4.1.1 Frame Checksum

The checksum is *computed* by adding the binary values of the ASCII characters, but keeping only the least significant eight bits of the result. The checksum is initialized to 0 by the <STX> character. The first character included in the checksum computation is the *frame number*. Each character in the *text* is added to the checksum (modulo 256). The last character included in the checksum computation is the frame type character (<ETB> or <ETX>). The checksum computation does not include the <STX>, the checksum characters, or the trailing <CR> and <LF>.

The checksum is an 8-bit integer that can be considered as two groups of 4 bits. The checksum is *encoded* by converting the groups of 4 bits into the ASCII characters of hexadecimal representation ("0" to "9" or "A" to "F"). The two ASCII characters are transmitted as the encoded checksum with the most significant character first (C1).

Suppose the value of the checksum is *computed* to be 122. The decimal number 122 can be represented as 01111010 in binary or as 7A in hexadecimal. The *encoded* checksum is transmitted as the ASCII character "7" followed by the ASCII character "A".

Given the following record:

<STX>4R|1|^Glucose|0|mg/dL^B<CR><ETB>7B<CR><LF>

The following table illustrates explicitly how the check sum is calculated, using each character in the record, starting from the frame number, up to and including the final frame type character.

	Character	ASCII Value
Frame Number	4	52
Text 1	R	82
Text 2		124
Text 3	1	49
Text 4		124
Text 5	^	94
Text 6	^	94
Text 7	^	94
Text 8	G	71
Text 9	l	108
Text 10	u	117
Text 11	c	99
Text 12	o	111
Text 13	s	115
Text 14	e	101
Text 16		124
Text 17	0	48
Text 18		124
Text 19	m	109
Text 20	g	103
Text 21	/	47
Text 22	d	100
Text 23	L	76
Text 24	^	94
Text 25	B	66
Text 26	<CR>	13
Frame Type	<ETB>	23
	SUM	2362
	SUM mod 256	58
	HEX	3A

4.4.1.2 Defective Frames

The computer indicates it has received a defective frame by transmitting a <NAK> to the meter. Upon receiving the <NAK> the meter retransmits the last frame with the same frame number.

Any characters that appear before the <STX> or after the <ETB> or <ETX> are to be ignored by the computer during frame validation. The frame is rejected because the computer detects a character parity error, character framing error, etc. It is also rejected because the checksum computed on the received frame does not match the encoded checksum value received in the frame. Finally the frame is rejected if the *frame number* is not the same as the last accepted frame or one number higher.

Upon receiving a <NAK> or any character except an <ACK> or <EOT>, the meter increments a retransmit counter and retransmits the frame. If this counter shows a particular frame was sent and not received six times, the meter aborts the message by going to the termination phase (see ASTM Standard E-1381). An abort provides an escape from a condition where the transfer phase can not continue.

4.4.2 ASTM E-1394 Records

Each E-1394 record is sent as a separate E-1381 frame. This structure is a design decision, not a requirement of the ASTM standards (record boundaries do not have to be tied to frame boundaries). The E-1381 framing characters (i.e. <STX>, frame number, <ETB>, <ETX>, checksum characters, and trailing <CR><LF>) are not shown in the examples.

A message is transmitted as a number of variable length records. Each record contains multiple variable length fields. A message always starts with a Header Record. A Patient Record follows the Header Record. One or more Result Records will follow. The Result Record contains the necessary information to determine if the reading is a glucose or control result. The message always ends with a Message Terminator Record.

Note that the serial number is actually the model number followed by the serial number field. Also the model number can have a 5th character.

4.4.2.1 Header Record

Table 2: Header Record

Field #	Field Name	Contents
1	Record type	H
2	Delimiter definition	\^& = field delimiter \ = repeat delimiter ^ = component delimiter & = escape delimiter
4	INTERNAL USE	
5	Sender ID	p^v^s^k p = Meter product code (Bayer 7410). v = Digital-engine-software-version (xx.yy format)\ analog-engine-software-version (xx.yy format)\ AGP-software-version (xx.yy.zz format) Where: Digital-engine-software-version (xx.yy format) xx = Major revision yy = Minor revision Algorithm-version (xx.yy format) xx = Major revision yy = Minor revision AGP-software-version (xx.yy format). xx = Major Revision yy = Minor Revision s = Meter serial number (7410-SSSSSSS). The first 5 characters represent the meter model number (the dash “ – ” can be replaced with a capital letter for derived product codes). k = meter SKU identifier

6	Device Info	<p>A set of fields represented as key value pairs (key=value), and separated by carots ("^"). The "key" is the remote command token that would be used to read or write the item individually using a remote command, and the "value" is the value that would be read or written using a remote command. See Table 6: Address Tokens for Remote Command Mode for details.</p> <p>The values represented in this field are:</p> <table><tr><th>Token</th><th>Meaning</th></tr><tr><td>A</td><td>Result Marking Auto-Logging Enabled/Disabled</td></tr><tr><td>C</td><td>Configuration Bits</td></tr><tr><td>G</td><td>Meter Language</td></tr><tr><td>I</td><td>Test Reminder Interval</td></tr><tr><td>R</td><td>Reference Method</td></tr><tr><td>S</td><td>(Reserved/Internal)</td></tr><tr><td>U</td><td>Unit of Measure (for both Glucose and Insulin)</td></tr><tr><td>V</td><td>Glucose HI/LO Analysis Range</td></tr><tr><td>X</td><td>Personalized High/Low Glucose Value Range</td></tr><tr><td>Y</td><td>High/Low Value Range Limits</td></tr><tr><td>Z</td><td>Trends Mode Display</td></tr></table> <p>The format of the field is:</p> <p>A=val^C=val^G=val^I=val^R=val^S=val^U=val^V=val^X=val^Y=val^Z=val</p> <p>Where "val" is the value as returned by the respective command token on the left side of the equal sign.</p> <p>See the section titled "Commands" for details on the values and formats for each token.</p> <p>NOTE: The format illustrated here is an EXAMPLE used to illustrate the key-value pair pattern only. It is not to be used as the specification for individual values and value formatting. Formats and field lengths for each value are the same as that defined for "Remote Commands". Please refer to each corresponding remote command token (Table 6: Address Tokens for Remote Command Mode) for precise specifications.</p>	Token	Meaning	A	Result Marking Auto-Logging Enabled/Disabled	C	Configuration Bits	G	Meter Language	I	Test Reminder Interval	R	Reference Method	S	(Reserved/Internal)	U	Unit of Measure (for both Glucose and Insulin)	V	Glucose HI/LO Analysis Range	X	Personalized High/Low Glucose Value Range	Y	High/Low Value Range Limits	Z	Trends Mode Display
Token	Meaning																									
A	Result Marking Auto-Logging Enabled/Disabled																									
C	Configuration Bits																									
G	Meter Language																									
I	Test Reminder Interval																									
R	Reference Method																									
S	(Reserved/Internal)																									
U	Unit of Measure (for both Glucose and Insulin)																									
V	Glucose HI/LO Analysis Range																									
X	Personalized High/Low Glucose Value Range																									
Y	High/Low Value Range Limits																									
Z	Trends Mode Display																									
7	Result Count	Number of test results stored in the meter. 1 to 4 Characters.																								
8 – 11	Unused	Unused																								
12	Processing ID	P = Production																								
13	Specification Version	1																								
14	Date & Time of message	YYYYMMDDhhmmss format (year, month, day, hour, minute, seconds). Time of day is always in 24-hour clock format (hhmmss ranges from 000000 to 235959).																								

Notes:

- The type of meter can is generally determined by from the "meter product code".
- The "meter product code" is transmitted in the Header Record as a component of the Sender ID field.
- The Contour Next USB product code is Bayer7410.
- The meter model can be determined by the "meter serial number" which is transmitted in the Header Record as a component of the Sender ID field. The meter serial number may be used to associate a particular meter to a specific patient's test results.

For Contour Next USB, actual passwords are not calculated and sent in the header records. An empty field ("") is sent instead.

4.4.2.2 Patient Information Record

Table 3: Patient Information Record

Field #	Field Name	Contents
1	Record type	P
2	Sequence number	1
3	Practice assigned patient ID	Not used
4	Laboratory assigned patient ID	Not used

4.4.2.3 Test Order Record

Test Order Records are NOT used for Contour Next USB. Instead, the Result Record contains the necessary information to determine if the reading is a glucose or control result.

4.4.2.4 Result Record

Table 4: Result Record												
Field #	Field Name	Contents										
1	Record type	R										
2	Sequence number	1, 2, etc.										
3	Record ID	^^Glucose : record is a blood glucose reading, or control solution. ^^Carb : record is a carbohydrate reading ^^Insulin : record is an insulin reading Note: ^^ indicates that only part 4 of field 3 is used.										
4	Data/Masurement Value	Specifies the data value for the result / event. Glucose results within LO-HI range as defined by the “V” remote command.										
5	Units^Reference Method	Specifies measurement and reference details, based on the type of result record For Glucose: Units = “mg/dL” or “mmol/L”, depending upon on meter’s glucose Units of Measure setting Reference Method = “P” (Plasma) Thus, for a Glucose result, Units^Reference Method will be one of the following: mg/dL^P Plasma in milligrams per deciliter mmol/L^P Plasma in millimoles per liter For Carbs: Units = units used to capture carbs data as follows: <table><tr><th>Code</th><th>Definition</th></tr><tr><td>0</td><td>Carbs Units not set</td></tr><tr><td>1</td><td>Grams</td></tr><tr><td>2</td><td>Points</td></tr><tr><td>3</td><td>Choices</td></tr></table>	Code	Definition	0	Carbs Units not set	1	Grams	2	Points	3	Choices
Code	Definition											
0	Carbs Units not set											
1	Grams											
2	Points											
3	Choices											

Table 4: Result Record

Field #	Field Name	Contents										
		<p>Reference Method = 0 (empty field) , i.e., no reference method for carbohydrate readings</p> <p>For Insulin:</p> <p>Units = used to indicate the Insulin Type as follows:</p> <table><tr><th>Code</th><th>Definition</th></tr><tr><td>0</td><td>No reference method / not set</td></tr><tr><td>1</td><td>Fast-Acting</td></tr><tr><td>2</td><td>Long-Acting</td></tr><tr><td>3</td><td>Mixed</td></tr></table> <p>Reference Method = 0 (empty field) , i.e., no reference method for insulin readings</p>	Code	Definition	0	No reference method / not set	1	Fast-Acting	2	Long-Acting	3	Mixed
Code	Definition											
0	No reference method / not set											
1	Fast-Acting											
2	Long-Acting											
3	Mixed											
6	Unused	Unused										

Table 4: Result Record

Field #	Field Name	Contents																																																						
7	Markers	This field specifies combinations of markers, with repeat delimiters (" "), or empty.																																																						
		<table><tr><th>Marker</th><th>Description</th></tr><tr><td><</td><td>Result below meter scale (see "V" token).</td></tr><tr><td>></td><td>Result above meter scale (see "V" token).</td></tr><tr><td>B</td><td>Before Meal (i.e. "pre-meal")</td></tr><tr><td>A</td><td>After Meal (i.e., "post-meal")</td></tr><tr><td>D</td><td>Don't Feel Right</td></tr><tr><td>F</td><td>Fasting Flag</td></tr><tr><td>I</td><td>Sick (ill)</td></tr><tr><td>S</td><td>Stress</td></tr><tr><td>X</td><td>Activity (exercise)</td></tr><tr><td>C</td><td>Control result (auto-detected).</td></tr><tr><td>Mn</td><td>RESERVED</td></tr><tr><td>Tn</td><td>RESERVED</td></tr><tr><td>Zn</td><td>Time after meal</td></tr><tr><td></td><td>Hex digit indicates hours after meal:</td></tr><tr><td></td><td>1 = .25</td></tr><tr><td></td><td>2 = .50</td></tr><tr><td></td><td>3 = .75</td></tr><tr><td></td><td>4 = 1.00</td></tr><tr><td></td><td>5 = 1.25</td></tr><tr><td></td><td>6 = 1.50</td></tr><tr><td></td><td>7 = 1.75</td></tr><tr><td></td><td>8 = 2.00</td></tr><tr><td></td><td>9 = 2.25</td></tr><tr><td></td><td>A = 2.50</td></tr><tr><td></td><td>B = 2.75</td></tr><tr><td></td><td>C = 3.00</td></tr></table>	Marker	Description	<	Result below meter scale (see "V" token).	>	Result above meter scale (see "V" token).	B	Before Meal (i.e. "pre-meal")	A	After Meal (i.e., "post-meal")	D	Don't Feel Right	F	Fasting Flag	I	Sick (ill)	S	Stress	X	Activity (exercise)	C	Control result (auto-detected).	Mn	RESERVED	Tn	RESERVED	Zn	Time after meal		Hex digit indicates hours after meal:		1 = .25		2 = .50		3 = .75		4 = 1.00		5 = 1.25		6 = 1.50		7 = 1.75		8 = 2.00		9 = 2.25		A = 2.50		B = 2.75		C = 3.00
		Marker	Description																																																					
		<	Result below meter scale (see "V" token).																																																					
		>	Result above meter scale (see "V" token).																																																					
		B	Before Meal (i.e. "pre-meal")																																																					
		A	After Meal (i.e., "post-meal")																																																					
		D	Don't Feel Right																																																					
		F	Fasting Flag																																																					
		I	Sick (ill)																																																					
		S	Stress																																																					
		X	Activity (exercise)																																																					
		C	Control result (auto-detected).																																																					
		Mn	RESERVED																																																					
		Tn	RESERVED																																																					
		Zn	Time after meal																																																					
			Hex digit indicates hours after meal:																																																					
			1 = .25																																																					
			2 = .50																																																					
			3 = .75																																																					
			4 = 1.00																																																					
			5 = 1.25																																																					
			6 = 1.50																																																					
			7 = 1.75																																																					
			8 = 2.00																																																					
			9 = 2.25																																																					
			A = 2.50																																																					
			B = 2.75																																																					
			C = 3.00																																																					
		NOTES:																																																						
		<ul style="list-style-type: none">• "<" or ">" can appear with any other marker.• "M" is only present with GLU record• "B" and "A" are mutually exclusive.• "B", "A" and "F" are mutually exclusive.• "C" (control) can be used with "<" or ">".• ">" and "<" are mutually exclusive.• Markers other than "<" or ">" should be ignored if "C" marker is present.• Any combination of "<" or ">", "A" or "B", "D", "I", "S", "X", "M", "F", "T" is valid.• "Z" is only valid with "A"																																																						
		8	Unused	Unused																																																				
		9	Date & time of test	YYYYMMDDhhmm																																																				

4.4.2.5 Message Terminator Record

Table 5: Message Terminator Record

Fld #	Name	Contents	Length	Treated as
1	Record type	L	1	Char
2	Sequence number	1	1	Int
3	Read Key	nnnnnnnnnn When retrieving result readings, this field contains a key used to retrieve results on subsequent "N" commands (get new results).	11 or 0	Char (cookie/key)
4	Termination code	N (normal termination)	1	Char

Although other values for Termination Code are specified in the ASTM standard, the only Termination Code ever generated by the meter is N, Normal Termination. If the computer does not receive a Message Terminator Record containing the N Termination Code, then any preceding data should not be used.

Presence of the Message Terminator Record indicates validity of the preceding *Data Transfer Mode* test results message transmission from the meter. The absence of the Message Terminator Record should be interpreted as indicating that the preceding *Data Transfer Mode* test results message transmission from the meter is invalid, and the data should not be used. There is no other error reporting mechanism used to indicate validity of the test results data sent during *Data Transfer Mode*.

4.5 Data Transfer and Remote Command Modes

The meter's clinical results are sent to a data management system in *Data Transfer* mode. In *Remote Command* mode, the user can interrogate and change various settings on the Contour Next USB meter, such as the current date, time, etc.

The Contour Next USB meter configuration is changed by the user in *Setup Mode*. The meter's configuration can also be changed through its computer interface.

For meter-PC communications, ASTM Standard E 1394-91, "Standard Specification for Transferring Information between Clinical Meters and Computer Systems", defines the data format. The HID protocol implementation is discussed in section 3.1 "Contour Next USB HID".

4.5.1 Data Transfer Mode

In *Data Transfer* mode, the patient's clinical results, control readings, etc., are transferred to a computer through the interface. When the Contour Next USB meter enters *Communication Mode*, the meter attempts to detect if the interface is connected. If an interface is detected, the meter enters *Data Transfer* mode. As long as the meter remains in *Communication Mode*, *Data Transfer* mode can be re-entered even after the results are sent to the computer.

When the Contour Next USB meter and computer are connected, the meter determines if the interface is connected by initially sending an <ENQ> character. If data management software responds within 15 seconds by sending an <ACK> character, the meter proceeds with the data transfer. If the data management software responds within 15 seconds with a <NAK> character, the meter sends an <EOT> then attempts to enter data download mode by looking for an <ENQ> character from the data management software.

Any response within 15 seconds to the meter's <ENQ> other than <ACK> or <NAK> character causes the meter to send an <EOT>HEADER-RECORD<ENQ>.

When clinical results are sent in *Data Transfer* mode, all results in the meter's memory are sent, even those results that were previously transmitted. ⁶ If the computer system maintains a database of clinical results by patient, it must recognize that the meter may have previously sent some of the patient's clinical results.

Test results are sent in the order they were collected by the meter, regardless of the "time stamp" of the result. The most recently stored result is sent last (First-In-First-Out). **NOTE: Because the time stamp does not have a resolution lower than minutes, it IS theoretically possible to see two results with an identical time.**

Processing and proper handling of results with identical time stamps is the responsibility of the implementer.

Clinical results and results marked as Control may be mixed. The preceding order record indicates the specimen type. Information reported in *Data Transfer* mode is as follows:

- Version: meter software version
- Identification: meter model number and meter serial number. The meter serial number is embedded in the data stream.
- Configuration: reference method (plasma, whole blood or de-proteinized)
- Clinical results for blood samples: glucose value, glucose units of measure, date, time, result marker, result flags
- Control solution results for control samples: like clinical results, but marked as control solutions (identified by the preceding Order record)
- Checksum: the checksum value for the transmitted record

⁶ NOT to be confused with Data Transfer carried out in the scope of an "N" remote command. In this case, only new readings are transferred. Refer to Table 6: Address Tokens for Remote Command Mode.

4.5.2 Remote Command Mode

Remote commands are the way the computer configures and interrogates the Contour Next USB meter. Some configurable items are also changeable by the user, but not all (i.e.; reference method, model number, and serial number).

The ASTM instrument interface protocols are not used for the remote commands because the information the meter receives from the computer is not standardized and because the meter has a limited amount of memory to buffer received data. The protocol for receiving data requires the meter to store only a few bytes of data rather than an entire ASTM frame.

The computer indicates it has remote commands to send by transmitting an <ENQ> character, the meter responds with an <ACK> to indicate it is ready to receive. The computer may then send one or more remote commands to the meter. The computer must send an <EOT> to indicate it has completed sending remote commands. To establish communication, the computer must send <ENQ> within five seconds after the meter sends an <EOT> to complete the data download. A <CAN> character sent by the computer at any time causes the meter to exit the Data Download mode. Once communication is established, remote commands must be sent within 15 seconds of each other, otherwise the meter automatically exits the remote command mode.

4.5.2.1 Format for Remote Commands

The format for remote commands is:

action, delimiter, address token, delimiter, data, delimiter, terminator.

The computer must stop transmitting after each delimiter until the meter responds with an acknowledgment. In response to receiving the delimiter, the meter sends an <ACK> to indicate that the information received so far is acceptable and the computer may continue. After receiving the delimiter, the meter will respond within 500 milliseconds. The meter sends a <NAK> to indicate the information received is faulty in some way. After sending a <NAK>, the meter returns to the Establishment Phase.

The command delimiter is the '[' character. The character immediately preceding the first delimiter is interpreted as the command. The command terminator is the <CR> character. Characters received after the terminator and preceding the command are ignored by the meter (e.g. a <LF> character would be ignored).

For the remote commands that cause the meter to send data, after sending the second delimiter (']') the computer must not send any characters until the meter has finished sending all of the information, including the terminator and an <ACK>. No acknowledgment of the information sent by the meter is permitted, i.e. the meter continues to transmit after each delimiter until the terminator is sent, <ACK>s from the computer are not permitted. After receiving the command to send data, the meter will complete its response within 500 milliseconds.

There are two permissible remote command "action" values:

- W - for writing information to the meter
- R - for reading information from the meter

When using the W action commands, to confirm the changes in the meter settings are made as expected, "write" commands should be followed by the corresponding "read" commands and the data values compared.

Permissible addresses and data values for the R and W action commands are listed in Table 6: Address Tokens for Remote Command Mode, below.

Table 6: Address Tokens for Remote Command Mode

All data types are “Char” unless otherwise noted.

Token	Read/Write	Use	Len	Explanation																											
A	R	Result Marking/ Auto-logging	1	1 = Enabled (default), 0 = Disabled																											
C	R	Read Configuration Bits	2	<table border="1"><thead><tr><th>Bit</th><th>Function</th><th>Values</th></tr></thead><tbody><tr><td>1</td><td>Buzzer Enable</td><td>0 = disabled, 1 = enabled</td></tr><tr><td>2</td><td>Time Format</td><td>0 = 12 hr, 1 = 24 hr</td></tr><tr><td>4</td><td>RESERVED</td><td>RESERVED</td></tr><tr><td>8</td><td>Time Reminder</td><td>0 = disable, 1 = enable</td></tr><tr><td>16</td><td>RESERVED</td><td>RESERVED</td></tr><tr><td>32</td><td>Date Format</td><td>0 = m/d/y 1 = d.m.y</td></tr><tr><td>64</td><td></td><td></td></tr><tr><td>128</td><td></td><td></td></tr></tbody></table> <p>NOTE: Bits are specified in Hex format: i.e., 3B == 101011 buzzer enabled, 24 hour format, time reminder off, d.m.y date format.</p>	Bit	Function	Values	1	Buzzer Enable	0 = disabled, 1 = enabled	2	Time Format	0 = 12 hr, 1 = 24 hr	4	RESERVED	RESERVED	8	Time Reminder	0 = disable, 1 = enable	16	RESERVED	RESERVED	32	Date Format	0 = m/d/y 1 = d.m.y	64			128		
Bit	Function	Values																													
1	Buzzer Enable	0 = disabled, 1 = enabled																													
2	Time Format	0 = 12 hr, 1 = 24 hr																													
4	RESERVED	RESERVED																													
8	Time Reminder	0 = disable, 1 = enable																													
16	RESERVED	RESERVED																													
32	Date Format	0 = m/d/y 1 = d.m.y																													
64																															
128																															
D	RW	Date	6	YYMMDD (Year, Month, Day).																											
E	W	Turn off meter	0	Turn meter off and put into state safe for ejecting USB device.																											
I	R	Test Reminder Interval: hours and minutes until reminder issued.	4	Format: hhmm hh = hours mm = minutes																											

Token	Read/Write	Use	Len	Explanation
N	R	Get New Results	0-12	<p>R N KKKKKKKKKKKK</p> <p>This command allows all readings or only NEW readings to be retrieved from the meter. “NEW” readings are defined as those readings not yet retrieved by a specific client application, from a particular meter, using the N command.</p> <ul style="list-style-type: none"> KKK... = Key. The value and format of this key is under control of the meter software and is implementation specific. The client sets this value to either an empty string on first request, or to request all results. The meter returns a new key value in the terminator record of an N command, to be cached by the client and used on subsequent requests to get only “new” values since the previous read. The client should NOT attempt to construct this key by itself, unless requesting all records or on an initial read (empty value). The following is an example implementation, for purposes of illustration only: <ul style="list-style-type: none"> xxxxXXXXXXXX = Key, where: xxxx = “next index” as hexadecimal value indicating the next buffer index at which to start reading new results. The meter re-calculates this value with every N command and returns it in the KEY field of the terminator record of an N command. XXXXXXXX = “total_read” as hexadecimal value indicating the total tests read by the client via the N command, since the last time the meter was cleared (i.e., using the M command). The meter re-calculates this value with every N command and returns it in KEY field of the terminator record of an N command. <p>NOTE: The value passed should either be empty (“”) or the key value returned by the meter in the Terminator Record after retrieving results using the “N” Remote Command.</p> <p>The result format for returned data is identical to that documented for Data Transfer Mode, sent in a single stream, without the need for the client to acknowledge each record with an <ACK>.</p>
T	RW	Time of day	4	hhmm (Hour, Minute)
U	R	Read GLU Unit of Measure	3	<p>Reads Unit of Measure for Glucose and/or Carbs.</p> <p>Format = TGC</p> <p>T = Target Field Bit Mask that determines whether the Glucose value, Carbs value, or both should be evaluated when writing, as follows:</p> <p>1 = Glucose, 2 = Carbs, 3 = Glucose AND Carbs</p> <p>G = Glucose Unit of Measure, as follows:</p> <p>0 = mg/dL, 1 = mmol/L</p> <p>C = Carbs Unit of Measure as follows:</p> <p>0 = Not set, 1 = Grams, 2 = Points, 3 = Choices</p> <p>NOTE: When reading, any data provided is ignored. The meter always returns “3” for the target field and the values for both the GLU and Carbs units.</p>

Token	Read/Write	Use	Len	Explanation																																				
V	R	Read Glucose HI/LO	5	LLHHH (values in mg/dL) Gets the values for LO and HI cut off ranges for analysis, in mg/dL. LL = Low cut off HHH = High cut off																																				
X	R	High/Low Target Ranges	30	AAABBBCCDDDEEEFFFGGGHHHIIJJJ (Values in mg/dL) <table><tr><td></td><td colspan="2">Valid Range mg/dL</td></tr><tr><td></td><td>Lower Bound</td><td>Upper Bound</td></tr><tr><td>AAA Hypoglycemic Limit</td><td>Y token DDD</td><td>Y token CCC</td></tr><tr><td>BBB Overall Low</td><td>Y token FFF</td><td>Y token EEE</td></tr><tr><td>CCC Pre Food Low</td><td>Y token FFF</td><td>Y token EEE</td></tr><tr><td>DDD Post Food Low</td><td>Y token FFF</td><td>Y token EEE</td></tr><tr><td>EEE Overall High</td><td>Y token HHH</td><td>Y token GGG</td></tr><tr><td>FFF Pre Food High</td><td>Y token HHH</td><td>Y token GGG</td></tr><tr><td>GGG Post Food High</td><td>Y token HHH</td><td>Y token GGG</td></tr><tr><td>HHH Hyperglycemic Limit</td><td>Y token BBB</td><td>Y token AAA</td></tr><tr><td>III Fasting High</td><td>Y token FFF</td><td>Y token EEE</td></tr><tr><td>JJJ Fasting Low</td><td>Y token HHH</td><td>Y token GGG</td></tr></table>		Valid Range mg/dL			Lower Bound	Upper Bound	AAA Hypoglycemic Limit	Y token DDD	Y token CCC	BBB Overall Low	Y token FFF	Y token EEE	CCC Pre Food Low	Y token FFF	Y token EEE	DDD Post Food Low	Y token FFF	Y token EEE	EEE Overall High	Y token HHH	Y token GGG	FFF Pre Food High	Y token HHH	Y token GGG	GGG Post Food High	Y token HHH	Y token GGG	HHH Hyperglycemic Limit	Y token BBB	Y token AAA	III Fasting High	Y token FFF	Y token EEE	JJJ Fasting Low	Y token HHH	Y token GGG
	Valid Range mg/dL																																							
	Lower Bound	Upper Bound																																						
AAA Hypoglycemic Limit	Y token DDD	Y token CCC																																						
BBB Overall Low	Y token FFF	Y token EEE																																						
CCC Pre Food Low	Y token FFF	Y token EEE																																						
DDD Post Food Low	Y token FFF	Y token EEE																																						
EEE Overall High	Y token HHH	Y token GGG																																						
FFF Pre Food High	Y token HHH	Y token GGG																																						
GGG Post Food High	Y token HHH	Y token GGG																																						
HHH Hyperglycemic Limit	Y token BBB	Y token AAA																																						
III Fasting High	Y token FFF	Y token EEE																																						
JJJ Fasting Low	Y token HHH	Y token GGG																																						
Y	R	Valid Limits for X Token Ranges	12	AAABBBCCDDDEEEFFFGGGHHH <table><tr><th>Field</th><th>Meaning</th></tr><tr><td>AAA</td><td>Upper bound HyPERglycemic limit</td></tr><tr><td>BBB</td><td>Lower bound HyPERglycemic limit</td></tr><tr><td>CCC</td><td>Upper bound HyPOglycemic limit</td></tr><tr><td>DDD</td><td>Lower bound HyPOglycemic limit</td></tr><tr><td>EEE</td><td>Upper bound of LOW TARGET ranges</td></tr><tr><td>FFF</td><td>Lower bound of LOW TARGET ranges</td></tr><tr><td>GGG</td><td>Upper bound of HIGH TARGET ranges</td></tr><tr><td>HHH</td><td>Lower bound of HIGH TARGET ranges</td></tr></table>	Field	Meaning	AAA	Upper bound HyPERglycemic limit	BBB	Lower bound HyPERglycemic limit	CCC	Upper bound HyPOglycemic limit	DDD	Lower bound HyPOglycemic limit	EEE	Upper bound of LOW TARGET ranges	FFF	Lower bound of LOW TARGET ranges	GGG	Upper bound of HIGH TARGET ranges	HHH	Lower bound of HIGH TARGET ranges																		
Field	Meaning																																							
AAA	Upper bound HyPERglycemic limit																																							
BBB	Lower bound HyPERglycemic limit																																							
CCC	Upper bound HyPOglycemic limit																																							
DDD	Lower bound HyPOglycemic limit																																							
EEE	Upper bound of LOW TARGET ranges																																							
FFF	Lower bound of LOW TARGET ranges																																							
GGG	Upper bound of HIGH TARGET ranges																																							
HHH	Lower bound of HIGH TARGET ranges																																							
Z	RW	Trends Mode Display	1	0 = 7 Day Trends Mode 1 = 14 Day Trends Mode 2 = 30 Day Trends Mode 3 = 90 day Trends Mode																																				

5 Appendix

5.1 Mutual Exclusive Access

It is imperative to ensure that only one application has access to a Contour Next USB meter at a time. This is to avoid the potential hazard of confusing readings or command results between different applications. To ensure mutually exclusive device access, an application should open the device file handle by specifying flags for NO sharing, neither for read or for write access.

For example, to open a mutually exclusive connection with the Contour Next USB meter on the Windows operating system, use code similar to the following:

```
#define NO_SHARED_CONTOUR_USB_ACCESS 0

HANDLE hDevice = CreateFile( DevicePath, GENERIC_READ | GENERIC_WRITE,
NO_SHARED_CONTOUR_USB_ACCESS, NULL, OPEN_EXISTING, 0, NULL);

if ( hDevice == INVALID_HANDLE_VALUE)
{
    // error...
}
```

Code for different operating systems such as Mac OS X, Unix and Linux variants, etc., will be different from Windows, but modern operating systems typically provide device/file access API's that allow specification of sharing modes. Please refer to the programming documentation for your operating system to ensure that the Contour Next USB connection is opened for mutually exclusive access.

It is the responsibility of the application implementing the protocols described in this document to ensure mutually exclusive access to the Contour Next USB meter.

5.2 Communication Hints

This section contains hints/tips for communicating with the Contour Next USB meter. PLEASE NOTE: All of the following examples are intended to illustrate form and function only. Values are NOT to be taken literally or as a source of reference.

5.2.1 Determine Presence and Type of Meter

To determine if a meter is prepared to communicate with the computer, the program could just wait up to 16 seconds while listening for an <ENQ> character from the meter. To speed up this process, the PC can send any character except an <ACK> or <NAK> character to the meter, e.g. the character "X". In response, the meter will send <EOT> followed by a full HEADER record followed by and <ENQ>. The Header Record contains the meter product code, meter software version, meter serial number and configuration information.

Description	From Client	From Meter
Client Sends "X"	X	
Meter Responds		{Header Record}

5.2.2 Configure Meter without Receiving Test Results

Follow these steps to read or change date/time without first transferring results from the meter.

- 1) Send "X" to the meter.
- 2) Meter responds with <EOT> HEADER <ENQ> within 2 seconds.
- 3) Send <NAK> to reject data transfer from the meter.

- 4) Meter responds with <EOT> to indicate it is exiting the transfer phase.
- 5) Send <ENQ> to the meter to initiate sending remote commands.
- 6) Meter responds with <ACK> to indicate it is ready to receive remote commands.
- 7) Send remote commands.

Send <EOT> to the meter to indicate remote commands are complete.

5.3 Examples

5.3.1 Header Record

Table 7: Example Header Record

NOTE: Values are examples only. They are not intended as references for true defaults or real world values.

<pre><STX>1H \^& 27719 BayerXXXX^01.01\01.02\01.03^XXXX-0002193\YYYY A=1^C=9^G=en,ar\zh\hr\cs\da\nl\en\fi\fr\de\el\^I=hhmm^R=0^S=00^U=300^V=20600^ X=060080070085140130140160070120^Y=170130070050080060170130^Z=1 30 P 1 200806031310<CR><E TB>F3<CR><LF></pre>	
Record Number	1
Record type	H
Delimiter Definition	\^&
For Bayer Internal Use Only	nnnnn
Sender ID	BayerXXXX^01.01\01.02\01.03.00^XXXX-0002193\YYYY
Meter Product Code	BayerXXXX
Meter Software Version	01.01
Algorithm Version	01.02
AGP Version	01.03
Meter Serial Number	XXXX-0002193
Meter SKU Identifier	YYYYY
Device Info	A=1^C=9^G=en,ar\zh\hr\cs\da\nl\en\fi\fr\de\el\he^I=hhmm^R=0^S=00^U=100^V=20600^X=1108013090^Y=11025060110^Z=1
Autologging	A = 1 (Enabled)
Configuration Bits	C=9 (01001 = Buzzer on, 12 hour format, Time reminder on, m/d/y date format)
Language	G = en,ar\zh\hr\cs\da\nl\en\fi\fr\de\el\he
Test Reminder Interval	I = hhmm
Reference Method	R=0 (always Plasma)
Reserved	S=00
Unit of Measure	U=300=mg/dL (Specifying both GLU and Carb Units, GLU units = milligrams per deciliter, Carbs units = Grams)
GLU HI / LO Range	V=20600 (LO = 20, HI = 600)

Personalized High/Low Glucose Range	X=060080070085140130140160070120 Hypo Limit = 060 Overall Low = 080 Pre-low = 070 Post-Low = 085 Overall High = 140 Pre-High = 130 Post-High = 140 Hyper Limit = 160 Fasting Low = 070 Fasting High = 120
High Low Limits	Y=170130070050080060170130 Upper bound HyPERglycemic limit = 170 Lower bound HyPERglycemic limit = 130 Upper bound HyPOglycemic limit = 070 Lower bound HyPOglycemic limit = 050 Upper bound of LOW TARGET ranges = 080 Lower bound of LOW TARGET ranges = 060 Upper bound of HIGH TARGET ranges = 170 Lower bound of HIGH TARGET ranges = 130
Trends Mode Display	Z=1 (14 Day Trends Mode)
Readings Count	30
Processing ID	P
Specification Version	1
Dater & Time of Message	200806031310

5.3.2 Patient Record

Table 8: Example Patient Record

<STX>2P 1<CR><ETB>53<CR><LF>

5.3.3 Result Record

These examples illustrate variations of the Result Record. The examples do not include framing characters or checksum characters.

Table 9: Glucose Example Result Record

R|1|^^^Glucose|113|mg/dL^P||A/Z6||20061108101300<CR>

Universal Test ID	Glucose
Glucose Level	113
Fixed Units Label	mg/dL
Reference Method	Plasma
Result Marker	A: After meal (post-meal) Z6 : Time After Food = Code 6 = 1.5 hours
Result Status	None
Year, Month, Day	20061108
Hour, Minute	1013

Table 10: Carbs Example Result Record

R|1|^^^Carb|5|1^0|||20061108104500<CR>

Universal Test ID	Carbohydrate
Carbs Level	5
Units	1 (Grams)
Reference	0 (Empty/Unused)
Result Marker	None
Result Status	None
Year, Month, Day	20061108
Hour, Minute	1045

Table 11: Insulin Example Result Record

R|1|^^^Insulin|2|1^0|||20060808094800<CR>

Universal Test ID	Insulin
Insulin Level	2
Type	1 (Fast-Acting)
Reference	0 (Empty/Unused)
Result Marker	None
Result Status	None
Year, Month, Day	20060808
Hour, Minute	0948

Table 12: Glucose “LO” Example Result Record

R|6|^^^Glucose|19|mg/dL^P||<||20020831100700<CR>

Universal Test ID	Glucose
Glucose Level	19
Fixed Units Label	mg/dL
Reference Method	Plasma
Markers	< : LO (below configured meter scale).
Year, Month, Day	20020831
Hour, Minute	1007

Table 13: Glucose No Markers Example Result Record

R | 8 | ^^^Glucose | 37 | mg/dL^P | | | 20020831100900<CR>

Universal Test ID	Glucose
Glucose Level	37
Fixed Units Label	mg/dL
Reference Method	plasma
Markers	none
Year, Month, Day	20020831
Hour, Minute	1009

Table 14: Glucose Self Detected Control Example Result Record

R | 9 | ^^^Glucose | 47 | mg/dL^P | | C | 20020831101000<CR>

Universal Test ID	Glucose
Glucose Level	47
Fixed Units Label	mg/dL
Reference Method	plasma
Markers	C – self detected control
Year, Month, Day	20020831
Hour, Minute	1010

Table 15: Glucose Plasma and mmol/L Example Result Record

R | 10 | ^^^Glucose | 2.61 | mmol/L^P | | | 20020831101100<CR>

Universal Test ID	Glucose
Glucose Level	2.61
Fixed Units Label	mmol/L (47 mg/dL)
Reference Method	plasma
Markers	none
Year, Month, Day	20020831
Hour, Minute	1011

Table 16: Glucose Absolute High Example Result Record

R | 12 | ^^^Glucose | 601 | mg/dL^P | | > | 20020831101300<CR>

Universal Test ID	Glucose
Glucose Level	601 (> 600)
Fixed Units Label	mg/dL
Reference Method	plasma
Markers	> : above absolute high meter scale
Year, Month, Day	20020831
Hour, Minute	1013

5.3.4 Message Terminator Record

Table 17: Example Terminator Record

Without N command key	<STX>3L 1 N<CR><ETX>06<CR><LF>
With N Command key	<STX>3L 1 01700001AE N<CR><ETX>06<CR><LF>

5.3.5 N Command Sequence

In this example, the client sends an “N” remote command to the meter. The N command responds by sending a single stream of records, in the same sequence they would have been delivered in Data Transfer Mode, but without requiring the client to send acknowledgements between each record. The stream ends with a Terminator Record containing the “key” to be used with the next “N” command. The client then uses that key on a subsequent “N” command to retrieve only new results. **NOTE: The “From Meter” column below shows descriptions of what will come back from the meter as examples only. These should not be interpreted as “real world values”. For precise descriptions of header records, results records, etc., please refer to sections in this document that describe those objects in detail.**

Description	From Client	From Meter
On “day one” computer issues command to get all readings by sending the “N” command, with a NULL (empty) key.	R N <CR>	
Meter sends stream of records in the sequence as in Data Transfer Mode, but without waiting for the client to <ACK> each record.		{ Header Record }
Patient Record		{ Patient Record }
GLU Result		{ Result Record }
GLU Result		{ Result Record }
GLU Result		{ Result Record }
Terminator Record With “ KEY ” for use in next N command.		Terminator record, similar to the following, with “KEY” to be used with subsequent N Token command: <STX> 6L 1 0030000000003 N<CR><ETX>03 <CR><LF>
Meter send EOT and goes back into Remote Command Mode		<EOT>
...
The client saves the “key” and uses it on the subsequent “N” command	R N 0030000000003 <CR>	
Meter responds with Header record, etc.		{ Header Record }
Etc...	Etc...	Etc..
Terminator Record With “ KEY ” for use in next N command.		Terminator record, similar to the following, with “KEY” to be used with subsequent N Token command: <STX>0L 1 0050000000005 N<CR><ETX>03<CR><LF>
Meter send EOT and goes back into Remote Command Mode		<EOT>

5.3.6 Write Date and Time to Meter

In this example, the client application sets the date to December 31, 2010, and the time to 9:45PM. The computer then verifies the changed settings. The remote command traffic is shown below:

Description	From Client	Meter Response
Get meter's attention	X	<ENQ>
Switch to remote command mode	<NAK>	<EOT>
	<ENQ>	<ACK>
Write Date December 31, 2010	W	<ACK>
	D	<ACK>
	101231 <CR>	<ACK>
Read Date from meter	R	<ACK>
	D <CR>	D 101231 <CR>
(Client verifies read-back value)		
Write Time 9:45 PM	W	<ACK>
	T	<ACK>
	2145 <CR>	<ACK>
Read Time from meter	R	<ACK>
	T <CR>	D 2145 <CR>
(Client verifies read-back value)		



BAYER HEALTHCARE LLC – DIABETES CARE

COMMUNICATIONS PROTOCOL DEVELOPERS GUIDE (“CPDG”)

HIGHLIGHTS OF LICENSEE OBLIGATIONS UNDER CPDG LICENSE AGREEMENT

- Only download test data from Bayer HealthCare LLC Diabetes Care (“BDC”) Blood Glucose Meters.
- Do not alter blood glucose meter functions, and do not alter or erase data in any way, with the specific and limited exception that DM Software may offset the date and time settings.
- Do not indicate that BDC is a partner of Licensee.
- Do not indicate that BDC approves of, endorses, or regulates Licensee’s DM Software.
- Include a copyright legend (e.g., “Communication protocols © 1999-2013 Bayer. All rights reserved”) in the DM Software packaging and manual, indicating that BDC is the copyright owner of the meter communication protocols.
- Do not use any BDC logo or BDC trademark (other than specific references to BDC Blood Glucose Meters).
- Refer to BDC Blood Glucose Meters using the appropriate trademark registration designation, including: BREEZE®, BREEZE® 2, DIDGET®, CONTOUR® NEXT, CONTOUR® NEXT USB, CONTOUR® NEXT EZ, CONTOUR® NEXT LINK, CONTOUR® LINK, ELITE®, DEX®, CONFIRM®, COUTOUR®, CONTOUR® USB, CONTOUR® XT, and/or CONTOUR® TS.
- In any material using a BDC trademark, include an appropriately adapted statement that says:
“[Licensee’s product] includes proprietary communication protocols copyrighted by Bayer HealthCare LLC and licensed to [Licensee] to facilitate download of blood glucose meter data from Bayer devices. Bayer has not evaluated device compatibility, nor has Bayer approved or endorsed [Licensee’s Product].
Communication protocols © 1999-2013 Bayer. All rights reserved.”
- Characterize the DM Software as “intended for use with [Contour®, e.g.] blood glucose meters.”
- Before publicly using, distributing or selling the DM Software containing the BDC communication protocol, test and confirm that (1) the DM Software properly downloads BDC Blood Glucose Meter data and (2) does not interfere with the operation of BDC Blood Glucose Meters.
- Before publicly using, distributing or selling the DM Software, submit all necessary regulatory filings and apply for and receive all necessary approvals (e.g., FDA, CE, etc.).
- Maintain reasonable documentation necessary to establish such testing, confirmation, and regulatory filings.
- Maintain all Confidential Information, as defined in the License Agreement, in strict confidence and neither disclose such Confidential Information to any third party, nor allow any third party access to it or to use it for any purpose.
- Do not publish the CPDG or otherwise make it publicly available.
- Do not: (a) demonstrate, copy, sell or market the CPDG (or any portion thereof) to any third party; (b) publish or otherwise disclose information relating to performance or quality of the CPDG (or any portion thereof) to any third party; or (c) modify, reuse, disassemble, decompile, reverse engineer or otherwise translate the CPDG or any portion thereof.
- Maintain a valid email account and keep BDC apprised of the email address to which communication shall be sent.
- **Contact BDC to discuss a promotion agreement if Licensee desires to work with BDC to promote the interoperability of Licensee’s DM Software with BDC Blood Glucose Meters.**

These Highlights are summarized for Licensee’s convenience and do not include all terms and conditions in the License Agreement.
The terms and conditions of the License Agreement control and are not limited by this summary.