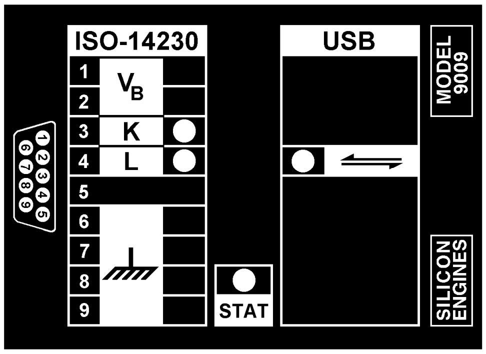
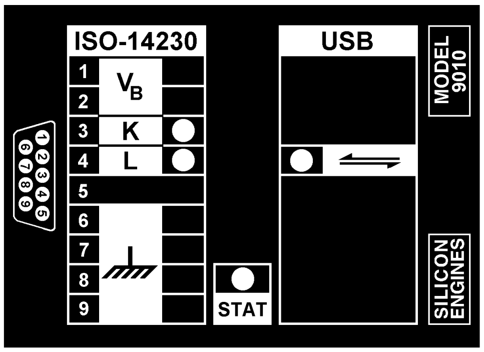
14230/USB CONVERTER

MODELS 9009 AND 9010

PROGRAMMER'S REFERENCE MANUAL





Copyright © 2005-2011
SILICON ENGINES, LTD.
All rights reserved. This document may not be copied, stored, or transmitted, by any means, without written permission.

COMMENTS

We would appreciate receiving corrections and suggestions regarding this document and the product it describes.
Please email to
14230@siliconengines.net

SILICON ENGINES

3550 W. Salt Creek Lane
Arlington Heights, IL 60005 USA
847-637-1180
FAX 847-637-1185
www.siliconengines.net

CONTENTS

1. OVERVIEW

1.1. INTRODUCTION

This document describes the 14230/USB DLL and the information necessary to write an application program that uses the Silicon Engines **14230/USB Converter**, a compact electronic device that allows a personal computer with a USB interface to connect to an automotive diagnostic data link compatible with ISO-14230.

This Programmer's Reference Manual applies both to the **Model 9009 14230/USB Converter**, as well as to the newer **Model 9010 14230/USB Converter**.

Note that the installation CD for this product includes full source code for the 14230 USB Message Center, which contains many usage examples and concrete code that can be re-used in a customer's own application program. The programmer who must write their own application for use with the 14230/USB converter is highly encouraged to start with the full source code provided on the installation CD as a starting point.

1.2. USER'S GUIDE

For general operating instructions, please refer to the User's Guide: GD9010 for the Model 9010, GD9009 for the Model 9009.

2. DESCRIPTION OF ROUTINES

2.1. ISO14230 USB EXTENDED OPEN

Function: Iso14230UsbExtendedOpen

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbOpen(
short *handle,
char SN[6],
unsigned long baudrate,
char errorstring[128]);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbExtendedOpen _
    Lib "14230Usb.dll" _
     (ByRef handle As Integer, _
     ByRef SerialNumber As Byte, _
     ByVal BaudRate As Long, _
     ByRef errorstring As Byte) _
  As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbExtendedOpen _
    Lib "14230USB.dll"  _
```

```
          (ByVal handle As System.IntPtr, _
           ByVal SerialNumber As System.IntPtr, _
           ByVal BaudRate As System.Int32, _
           ByVal errorstring As System.IntPtr) _
       As System.Int32 'returns 1 if ok
```

VB .NET 2005 64-Bit Calling Format:

```
     Public Declare Function Iso14230UsbExtendedOpen _
         Lib "14230USB64.dll"  _
         (ByVal handle As System.IntPtr, _
          ByVal SerialNumber As System.IntPtr, _
          ByVal BaudRate As System.Int32, _
          ByVal errorstring As System.IntPtr) _
      As System.Int32 'returns 1 if ok
```

Description: Use this function to (a) discover if any 14230/USB devices are connected to the computer and if so, obtain a handle to one of them, or (b) discover whether a 14230/USB device with a specified serial number is connected to the computer, and if so, obtain a handle to it, or (c) discover whether a 14230/USB device whose handle has already been obtained is still connected to the computer (although you can also use the function Iso14230UsbIsOpen which is less intrusive for that purpose).

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input/output] - This is the handle of the 14230/USB device being interrogated or set up. If the user wishes to direct the action of this routine to a specific 14230/USB device with a known handle, then that handle number should be put into this argument. If the user wishes to attach to the first available device then they should use a value of -1 for this parameter. On successful return, the variable pointed to by this argument will be set to the handle of the device that was connected to. All positive (zero or non-zero) integer values are legal values for the handle.

SerialNumber [input/output] – This is the serial number string of the 14230/USB device being interrogated or set up. If the user wishes to direct the action of this routine to a 14230/USB device with a specific serial number, then that serial number should be put into a standard C null-terminated ASCII string and pointed to by this argument. If the user wishes to attach to the first available device, then this argument should point to a byte buffer of at least 6 bytes long (5 byte long string with 1 byte null terminator). The serial number is at most 5 bytes long. On successful return, the string pointed to by this argument will contain the serial number of the device that was connected to.

BaudRate [input] - This specifies the baud rate of the ISO-14230 bus communications link (over the K-line) that this 14230/USB device will talk at. Valid baud rates are 1000 baud to 10417 baud for Model 9009 and 1000 baud to 115200 baud for Model 9010. Note that once this baud rate is set, it is remembered in non-volatile memory and used on the next power-up of the 14230/USB device.

errorstring [output] – The user should allocate a string of at least 128 single-byte characters and have a pointer to this string passed in this parameter. If this routine returns a code other than 1, then errorstring will contain a human-readable standard C null-terminated ASCII string containing a specific error message describing why the routine failed to execute the commanded operation.

2.2. ISO14230 USB OPEN

Function: Iso14230UsbOpen

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbOpen(
short *handle,
char SN[6],
short baudrate,
char errorstring[128]);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbOpen _
    Lib "14230Usb.dll" _
     (ByRef handle As Integer, _
     ByRef SerialNumber As Byte, _
     ByVal BaudRate As Integer, _
     ByRef errorstring As Byte) _
  As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbOpen _
    Lib "14230USB.dll"  _
    (ByVal handle As System.IntPtr, _
     ByVal SerialNumber As System.IntPtr, _
     ByVal BaudRate As System.Int16, _
     ByVal errorstring As System.IntPtr) _
 As System.Int32 'returns 1 if ok
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbOpen _
    Lib "14230USB64.dll"  _
    (ByVal handle As System.IntPtr, _
     ByVal SerialNumber As System.IntPtr, _
     ByVal BaudRate As System.Int16, _
     ByVal errorstring As System.IntPtr) _
```

```
        As System.Int32 'returns 1 if ok
```

Description: Use this function to (a) discover if any 14230/USB devices are connected to the computer and if so, obtain a handle to one of them, or (b) discover whether a 14230/USB device with a specified serial number is connected to the computer, and if so, obtain a handle to it, or (c) discover whether a 14230/USB device whose handle has already been obtained is still connected to the computer (although you can also use the function Iso14230UsbIsOpen which is less intrusive for that purpose). This routine is deprecated (provided for compatibility with existing applications): new users should use Iso14230UsbExtendedOpen. The drawback of using this routine is you will never be able to set a baud rate higher than 32767.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input/output] - This is the handle of the 14230/USB device being interrogated or set up. If the user wishes to direct the action of this routine to a specific 14230/USB device with a known handle, then that handle number should be put into this argument. If the user wishes to attach to the first available device then they should use a value of -1 for this parameter. On successful return, the variable pointed to by this argument will be set to the handle of the device that was connected to. All positive (zero or non-zero) integer values are legal values for the handle.

SerialNumber [input/output] – This is the serial number string of the 14230/USB device being interrogated or set up. If the user wishes to direct the action of this routine to a 14230/USB device with a specific serial number, then that serial number should be put into a standard C null-terminated ASCII string and pointed to by this argument. If the user wishes to attach to the first available device, then this argument should point to a byte buffer of at least 6 bytes long (5 byte long string with 1 byte null terminator). The serial number is at most 5 bytes long. On successful return, the string pointed to by this argument will contain the serial number of the device that was connected to.

BaudRate [input] - This specifies the baud rate of the Iso-14230 bus communications link (over the K-line) that this 14230/USB device will talk at. Valid baud rates are 1000 baud to 10417 baud for model 9009 and 1000 baud to 32767 baud for model 9010 (use Iso1423UsbExtendedOpen to get higher baud rates on the model 9010). Note that once this baud rate is set, it is remembered in non-volatile memory and used on the next power-up of the 14230/USB device.

errorstring [output] – The user should allocate a string of at least 128 single-byte characters and have a pointer to this string passed in this parameter. If this routine returns a code other than 1, then errorstring will contain a human-readable standard C null-terminated ASCII string containing a specific error message describing why the routine failed to execute the commanded operation.

2.3. ISO14230 USB IS OPEN

Function: Iso14230UsbIsOpen

Library: 14230USB.DLL

C/C++ Calling Format:

```
int _stdcall Iso14230UsbIsOpen(
short handle);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbIsOpen _
```

```
              Lib "14230Usb.dll" _
              (ByVal handle As Integer) _
             As Integer 'returns 1 if open
```

VB .NET 2005 32-Bit Calling Format:

```
     Public Declare Function Iso14230UsbIsOpen _
        Lib "14230USB.DLL" _
      (ByVal handle As System.Int16) _
      As System.Int32 'returns 1 if open
```

VB .NET 2005 64-Bit Calling Format:

```
     Public Declare Function Iso14230UsbIsOpen _
        Lib "14230USB64.DLL" _
      (ByVal handle As System.Int16) _
      As System.Int32 'returns 1 if open
```

Description: Use this routine to determine whether a given 14230/USB device handle still points to a valid, connected, working 14230/USB device.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates that the 14230/USB device specified by the handle input argument is connected and working properly and an open USB connection exists to this device. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.4. ISO14230 USB SET BUS CHARACTERISTICS

Function: Iso14230UsbSetBusCharacteristics

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetBusCharacteristics(
short handle,
unsigned char flags);
```

VB6 Calling Format:

```
Public Declare Function _
  Iso14230UsbSetBusCharacteristics _
  Lib "14230USB.DLL" _
 (ByVal handle As Integer, _
      ByVal flags As Byte) _
   As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
    Public Declare Function _
       Iso14230UsbSetBusCharacteristics _
       Lib "14230USB.DLL" _
       (ByVal handle As System.Int16, _
       ByVal flags As Byte) _
       As System.Int32 'returns 1 if ok
```

VB .NET 2005 64-Bit Calling Format:

```
    Public Declare Function _
       Iso14230UsbSetBusCharacteristics _
       Lib "14230USB64.DLL" _
       (ByVal handle As System.Int16, _
       ByVal flags As Byte) _
       As System.Int32 'returns 1 if ok
```

Description: Use this routine to set the Bus Load Resistor for a specified 14230/USB device. The bus load resistor is usually set based on whether this is a 12 volt K-line or 24-volt K-line or a 5-volt K-line (model 9010 only) or whether no pull-up resistor should be used at all (model 9010 only). The 14230/USB device can also be set to auto-detect the voltage on the K-line and to set the bus resistor based on the auto-detected voltage.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being modified. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

flags [input] - The byte value passed into this argument is interpreted on a bit-by-bit basis. Bits 7-4 are currently unused and should be set to 0. Bit 3 should be set if you want to specify a 5V bus. Bit 2 should be set if you want to specify that no bus load resistor should be used at all. Bit 1 indicates whether to force the resistance on the bus to what has been specified in bit 0. If bit 1 is 1, then the resistance on the bus is dictated by bit 0. If bit 1 is 0, then the resistance on the bus is dictated by whether the Iso14230Usb device senses that the battery voltage is at the 24 volt level or the 12 volt level (or as defined by bits 2 or 3). If bit 0 is active, then a 1 indicates use the 24 volts resistance and a 0 indicates use the 12 volts resistance.

2.5. ISO14230 USB SERIAL NUMBER

Function: Iso14230UsbSerialNumber

Library: 14230USB.DLL

C/C++ Calling Format:

```
int _stdcall Iso14230UsbSerialNumber(
     short handle,
     unsigned char SN[6]);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSerialNumber _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByRef SerialNumber As Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSerialNumber _
   Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
   ByVal SerialNumber As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSerialNumber _
   Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
   ByVal SerialNumber As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

Description: Use this routine to find out the serial number of a 14230/USB device that has already been opened by Iso14230UsbOpen or Iso14230UsbExtendedOpen.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being queried. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

SN [output] – This is a pointer to a buffer of at least 6 bytes to put the standard C null-terminated ASCII string containing the serial number of the 14230/USB device being queried. Upon successful completion of this routine, this field will be written by the routine with the serial number of the device.

2.6. ISO14230 USB OKAY TO TRANSMIT

Function: Iso14230UsbOkayToTransmit

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbOkayToTransmit(
short handle);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbOkayToTransmit _
    Lib "14230USB.DLL" _
    (ByVal handle As Integer) _
 As Integer 'returns 1 if ok to transmit
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbOkayToTransmit _
    Lib "14230USB.DLL" _
  (ByVal handle As System.Int16) _
As System.Int32 'returns 1 if ok to transmit
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbOkayToTransmit _
    Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16) _
As System.Int32 'returns 1 if ok to transmit
```

Description: This routine will return a code of 1 if the transmitter is idle and all messages commanded to transmit through the 14230/USB device and on to the K-line bus have been conveyed successfully to the 14230/USB device. It basically returns 1 if the transmit queue from the PC to the 14230/USB device of K-line bus messages that will go on the K-line bus is empty (all messages have been transmitted). This is a good indication that it is safe to send a new message on the K-line bus using Iso14230UsbWriteDataMsg.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates that the current transmit queue is empty and that it is okay to add another message to the 14230 transmit queue. Other values indicate specific errors (see Part 3) although a code of "2" could also mean simply that the transmit queue is not empty.

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.7. ISO14230 USB STATUS MSG WAITING

Function: Iso14230UsbStatusMsgWaiting

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbStatusMsgWaiting(
short handle);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbStatusMsgWaiting _
 Lib "14230USB.DLL" _
 (ByVal handle As Integer) _
 As Integer 'returns 1 if status message is waiting
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbStatusMsgWaiting _
 Lib "14230USB.DLL" _
 (ByVal handle As System.Int16) _
 As System.Int32 'returns 1 if status message is waiting
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbStatusMsgWaiting _
 Lib "14230USB64.DLL" _
 (ByVal handle As System.Int16) _
 As System.Int32 'returns 1 if status message is waiting
```

Description: This function should be used to determine if any status messages are waiting to be read by the PC from the specified 14230/USB device. Status messages are error conditions that can arise that are not directly caused by a PC command such as buffer overflows. If there are one or more status messages waiting to be read by the PC, then this routine will return a code of 1; otherwise it returns a code of 2. Read the status message(s) using the routine Iso14230UsbStatus.

Return Code: This function returns a 1 if there are one or more status messages waiting to be read; otherwise it returns 2.

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.8. ISO14230 USB DATA MSG WAITING

Function: Iso14230UsbDataMsgWaiting

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbDataMsgWaiting(
short handle);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbDataMsgWaiting _
```

```
        Lib "14230USB.DLL" _
       (ByVal handle As Integer) _
       As Integer 'returns 1 if message is waiting
```

VB .NET 2005 32-Bit Calling Format:

```
      Public Declare Function Iso14230UsbDataMsgWaiting _
        Lib "14230USB.DLL" _
       (ByVal handle As System.Int16) _
      As System.Int32 'returns 1 if message is waiting
```

VB .NET 2005 64-Bit Calling Format:

```
      Public Declare Function Iso14230UsbDataMsgWaiting _
        Lib "14230USB64.DLL" _
       (ByVal handle As System.Int16) _
      As System.Int32 'returns 1 if message is waiting
```

Description: This function should be used to determine if any data messages are waiting to be read by the PC from the specified 14230/USB device. Data messages are packages of 14230 data that have been seen on the K-line bus. If there are one or more data messages waiting to be read by the PC, then this routine will return a code of 1; otherwise it returns a code of 2. Read the data message(s) using successive calls to the routine Iso14230UsbReadDataMsg.

Return Code: This function returns a 1 if there are one or more data messages waiting to be read; otherwise it returns 2.

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.9. ISO14230 USB READ DATA MSG

Function: Iso14230UsbReadDataMsg

Library: 14230USB.DLL

C/C++ Calling Format:

```
      int __stdcall Iso14230UsbReadDataMsg(
      short handle,
      unsigned char *typemsg,
      unsigned short *count,
      unsigned char *buffer,
      unsigned char *error_code);
```

VB6 Calling Format:

```
      Public Declare Function Iso14230UsbReadDataMsg _
          Lib "14230USB.DLL" _
```

```
            (ByVal handle As Integer, _
            ByRef typemsg As Byte, _
            ByRef count As Integer, _
            ByRef buffer As Byte, _
            ByRef ErrorCode As Byte) _
          As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
     Public Declare Function Iso14230UsbReadDataMsg _
         Lib "14230USB.DLL" _
         (ByVal handle As System.Int16, _
         ByVal typemsg As System.IntPtr, _
         ByVal count As System.IntPtr, _
         ByVal buffer As System.IntPtr, _
         ByVal ErrorCode As System.IntPtr) _
     As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
     Public Declare Function Iso14230UsbReadDataMsg _
         Lib "14230USB64.DLL" _
         (ByVal handle As System.Int16, _
         ByVal typemsg As System.IntPtr, _
         ByVal count As System.IntPtr, _
         ByVal buffer As System.IntPtr, _
         ByVal ErrorCode As System.IntPtr) _
     As System.Int32 'returns 1 if successful
```

Description: Use this function to read messages that were received on the K-line bus. Normally a call to Iso14230UsbDataMsgWaiting is made first to see if this routine should or should not be called.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. A code of "2" can indicate that there are no received messages queued up currently. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

typemsg [output] – This argument points to a byte that is written if the routine returns successfully. The byte written will be either a "1" or a "0". It will be a "0" to indicate this is data that was received on the bus. It will be a "1" to indicate this data was actually transmitted by this 14230/USB device as part of its Tester Present message feature, which can be enabled to send out a message every 2 seconds.

count [input/output] – This argument points to an integer value that contains a count for the number of message bytes as well as the size of the buffer in the buffer argument. The value of this integer when this routine is called should be set to the maximum number of bytes that can be stored in the buffer in the buffer argument. The input value for count should be 31 or greater to hold the worst case minimum message size which is 31 for a Tester Present message. If the input value is less than the number of bytes that are pending, the bytes that are still pending will still be available in FIFO fashion on subsequent calls. When this routine returns, this integer value will contain the number of bytes in the K-line bus message that are now stored in the buffer pointed to by the buffer argument.

buffer [output] – This argument should contain a pointer to a buffer that will contain the full K-line bus message when this routine returns. The number of bytes allocated for this buffer should be put into the integer pointed to by the count argument. Note that every valid serial byte received on the K-line will be reported in the buffer. No attempt is made to accept valid message packets since the structure of message packets can vary and is specific to each application. Therefore all bytes are reported including checksums. On return from this routine, the buffer will contain the K-line bus message. A complete K-line bus message may consist of one or more buffers returned from this routine. The order of the bytes received will be preserved so you can receive the valid messages from the device under test by parsing the data one byte at a time.

ErrorCode [output] – This argument should point to a byte that can be written by the routine. The routine writes the final error code of the 14230/USB device to this byte. This error code is equal to 4F hex if there is no error. For a list of error codes, see Part 4. Often this error code reflects an error that occurred while receiving the current message, but it also can be a "leftover" error code from some error condition that occurred previously. To clear leftover error codes, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.10. ISO14230 USB ACK WAITING

Function: Iso14230UsbAckWaiting

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbAckWaiting(
short handle,
unsigned char *msg_acked,
     unsigned char *error_code);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbAckWaiting _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByRef msg_acked As Byte, _
  ByRef error_code As Byte) _
 As Integer 'returns 1 if ack received
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbAckWaiting _
  Lib "14230USB.DLL" _
 (ByVal handle As System.Int16, _
 ByVal msg_acked As System.IntPtr, _
 ByVal error_code As System.IntPtr) _
 As System.Int32 'returns 1 if ack received
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbAckWaiting _
  Lib "14230USB64.DLL" _
 (ByVal handle As System.Int16, _
 ByVal msg_acked As System.IntPtr, _
 ByVal error_code As System.IntPtr) _
 As System.Int32 'returns 1 if ack received
```

Description: This function should be used to determine if any command acknowledgements are waiting to be read by the PC from the specified 14230/USB device. Command acknowledgements are messages from the 14230/USB device that a specific command has just been executed. Normally this routine is not necessary since specific operations wait for the acknowledgement back before returning. This routine would only be used to determine if a "Lost Command" has finally executed. It can also be used to silently "kill" acknowledgement messages that get queued up that were orphaned from their original commands due to exceptional circumstances in order to keep the queue empty and free. If there are one or more acknowledgement messages waiting to be read by the PC, then this routine will return a code of 1; otherwise it returns a code of 2. If this routine returns 1 indicating an ACK is waiting, the ACK is automatically removed from the acknowledgement queue and the specific ACK message is stored in the msg_acked argument. Additionally, if this routine returns 1, then the current 14230/USB device error code is also loaded into the error_code argument.

Return Code: This function returns a 1 if there are one or more acknowledgement messages waiting to be read; otherwise it returns 2.

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being checked. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

msg_acked [output] – This argument should point to a byte. This byte is untouched unless the return code is 1. If the return code is 1, then this byte is filled with the FIFO acknowledgement queue first-in acknowledgement byte. The list of possible acknowledgement bytes is in Part 5.

error_code [output] – This argument should point to a byte that can be written by the routine. The routine writes the final error code of the 14230/USB device to this byte. This error code is equal to 4F hex if there is no error. For a list of error codes, see Part 4. Often this error code reflects an error that occurred while acting on the command/message acknowledged by the msg_acked argument's acknowledgement byte, but it also can be a "leftover" error code from some error condition that occurred previously. To clear leftover error codes, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.11. ISO14230 USB WRITE DATA MSG

Function: Iso14230UsbWriteDataMsg

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbWriteDataMsg(
short handle,
unsigned short count,
     unsigned char *buffer,
unsigned char *error_code,
short wait_for_ack);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbWriteDataMsg _
 Lib "14230USB.DLL" _
 (ByVal handle As Integer, _
 ByVal count As Integer, _
 ByRef buffer As Byte, _
 ByRef ErrorCode As Byte, _
 ByVal Wait_For_Ack As Boolean) _
As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbWriteDataMsg _
  Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
   ByVal count As System.Int16, _
   ByVal buffer As System.IntPtr, _
   ByVal ErrorCode As System.IntPtr, _
   ByVal wait_for_ack As System.Int16) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbWriteDataMsg _
  Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
   ByVal count As System.Int16, _
   ByVal buffer As System.IntPtr, _
```

```
    ByVal ErrorCode As System.IntPtr, _
    ByVal wait_for_ack As System.Int16) _
  As System.Int32 'returns 1 if successful
```

Description: This function is used to write a message to the K-line bus. Note that the entire message is transmitted first to the 14230/USB device and stored there before it is actually sent. This ensures that the entire message obeys the inter-byte spacing specified by the Iso14230UsbSetInterByteDelay command and there are no additional transmission latencies involved (other latencies are less than 1 msec). The maximum message length that can be sent is 9355 bytes.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

count [input] – This is the number of bytes of the K-line message to send including the K-line message header and checksum byte(s). The range for this value is 1-9355.

buffer [input] – This is the K-line message buffer for the message to send. This buffer should be allocated and filled with the number of bytes specified in the count argument. You will need to include all K-line message header data and K-line message checksum data as part of the data being sent from this buffer.

ErrorCode [output] – This argument should point to a byte that can be written by the routine. The routine writes the final error code of the 14230/USB device to this byte. This error code is equal to 4F hex if there is no error. For a list of error codes, see Part 4. Often this error code reflects an error that occurred while trying to send the specified message, but it also can be a "leftover" error code from some error condition that occurred previously. To clear leftover error codes, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

Wait_For_Ack [input] – If this is True (non-zero), then this routine will assure that the command to send the message is acknowledged successfully before returning a successful return code. You can also set this input parameter to False (zero) indicating that the command to send the message should be sent but that the routine should return immediately. This might be used for sending lots of fast messages on the K-line bus and then using the routine Iso14230UsbAckWaiting to confirm proper transmission of the messages afterwards.

2.12. ISO14230 USB READ STATUS

Function: Iso14230UsbReadStatus

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbReadStatus(
short handle,
unsigned char *code,
unsigned char *mode);
```

VB6 Calling Format:

```
     Public Declare Function Iso14230UsbReadStatus _
         Lib "14230USB.DLL" _
         (ByVal handle As Integer, _
         ByRef status As Byte, _
         ByRef mode As Byte) _
       As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
    Public Declare Function Iso14230UsbReadStatus _
         Lib "14230USB.DLL" _
        (ByVal handle As System.Int16, _
        ByVal status As System.IntPtr, _
        ByVal mode As System.IntPtr) _
    As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
    Public Declare Function Iso14230UsbReadStatus _
         Lib "14230USB64.DLL" _
        (ByVal handle As System.Int16, _
        ByVal status As System.IntPtr, _
        ByVal mode As System.IntPtr) _
    As System.Int32 'returns 1 if successful
```

Description: This function is used to get a status update on an 14230/USB device that is connected to the PC. It is a good command to send to get an overall picture of what state the 14230/USB device is in. It tells what current error code is present in the 14230/USB device and what mode the 14230/USB device is in (the mode feature is not used).

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

status [output] - This argument should point to a byte that can be written by the routine. The routine writes the current error code of the 14230/USB device to this byte. This error code is equal to 4F hex if there is no error. For a list of error codes, see Part 4. To clear this error code, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

mode [output] – This argument should point to a byte that can be written by the routine. This routine writes a 0 to this byte.

2.13. ISO14230 USB STATUS

Function: Iso14230UsbStatus

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbStatus(
short handle,
unsigned char *code);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbStatus _
    Lib "14230Usb.dll" _
    (ByVal handle As Integer, _
    ByRef status As Byte) _
  As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbStatus _
     Lib "14230USB.DLL" _
     (ByVal handle As System.Int16, _
     ByVal status As System.IntPtr) _
As System.Int32 'returns 1 if status msg waiting
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbStatus _
     Lib "14230USB64.DLL" _
     (ByVal handle As System.Int16, _
     ByVal status As System.IntPtr) _
As System.Int32 'returns 1 if status msg waiting
```

Description: This function is used to retrieve a status message from an 14230/USB device that is connected to the PC. The user should call the routine Iso14230UsbStatusMsgWaiting first to see if a status message is waiting to be read using this routine. This is used for status notifications from the 14230/USB device to the PC.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success indicating that the status argument has been filled with the previously queued status code. A return code of "2" indicates that there are no pending status messages in the status message queue.

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

status [output] - This argument should point to a byte that can be written by the routine. The routine writes the current error code of the 14230/USB device to this byte. This error code is equal to 4F hex if there is no error. For a list of error codes, see Part 4. To clear this error code, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.14. ISO14230 USB SET HIGH SPEED BAUD RATE

Function: Iso14230UsbSetHighSpeedBaudRate

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetHighSpeedBaudRate(
short handle,
unsigned long baudrate);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedBaudRate _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByVal BaudRate As Long) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedBaudRate _
  Lib "14230USB.DLL" _
 (ByVal handle As System.Int16, _
 ByVal BaudRate As System.Int32) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedBaudRate _
  Lib "14230USB64.DLL" _
 (ByVal handle As System.Int16, _
 ByVal BaudRate As System.Int32) _
As System.Int32 'returns 1 if successful
```

Description: This function is used to set the baud rate of the K-line bus communications link. Typical baud rates are 2400, 4800, 9600, and 10400. Valid values range anywhere from 1000 to 10417 (Model 9009) or 1000 to 115200 (Model 9010). Note that calls to Iso14230UsbOpen or Iso14230UsbExtendedOpen also set the baud rate of the device to a specified baud rate. Note that after this baud rate is set, it will be used every time the selected module is powered up at all times until this command or Iso14230UsbOpen or Iso14230UsbExtendedOpen with a different baud rate is executed.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

BaudRate [input] – This argument contains the new baud rate in units of bits per second. It can range from 1000 to 10417 (Model 9009) or 1000 to 115200 (Model 9010).

2.15. ISO14230 USB SET BAUD RATE

Function: Iso14230UsbSetBaudRate

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetBaudRate(
short handle,
unsigned short baudrate);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetBaudRate _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByVal BaudRate As Integer) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
     Public Declare Function Iso14230UsbSetBaudRate _
       Lib "14230USB.DLL" _
      (ByVal handle As System.Int16, _
      ByVal BaudRate As System.Int16) _
   As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
     Public Declare Function Iso14230UsbSetBaudRate _
       Lib "14230USB64.DLL" _
      (ByVal handle As System.Int16, _
      ByVal BaudRate As System.Int16) _
   As System.Int32 'returns 1 if successful
```

Description: This function is used to set the baud rate of the K-line bus communications link. This function is deprecated: new programs should use the function SetIso14230UsbHighSpeedBaudRate instead. Typical baud rates are 2400, 4800, 9600, and 10400. Valid values range anywhere from 1000 to 10417 (Model 9009) or 1000 to 32767 (Model 9010 – for higher baud rates, use Iso14230UsbSetHighSpeedBaudRate). Note that calls to Iso14230UsbOpen or Iso14230UsbExtendedOpen also set the baud rate of the device to a specified baud rate. Note that after this baud rate is set, it will be used every time the selected module is powered up at all times until this command or Iso14230UsbOpen or Iso14230UsbExtendedOpen with a different baud rate is executed.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

BaudRate [input] – This argument contains the new baud rate in units of bits per second. It can range from 1000 to 10417 (model 9009) or 1000 to 32767 (model 9010 – for higher baud rates, use Iso14230UsbSetHighSpeedBaudRate).

2.16. ISO14230 USB SET HIGH SPEED DUPLEX (MODEL 9010 ONLY)

Function: Iso14230UsbSetHighSpeedDuplex

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetHighSpeedDuplex(
short handle,
unsigned char duplexmode);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedDuplex _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByVal duplexval as Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedDuplex _
  Lib "14230USB.DLL" _
 (ByVal handle As System.Int16, _
 ByVal duplexval As System.Byte) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetHighSpeedDuplex _
  Lib "14230USB64.DLL" _
 (ByVal handle As System.Int16, _
 ByVal duplexval As System.Byte) _
As System.Int32 'returns 1 if successful
```

Description: This function is used to set the duplex mode for baud rates greater than 10417. In full duplex mode, transmitted bytes are echoed back as long as they were transmitted properly. This allows an application to guarantee delivery of its transmitted data by checking the echoed response. If the echoed response does not match, some sort of bus conflict must have occurred and the application can resend the message. This routine is not necessary for baud rates 10417 or lower because with those baud rates, bit assertions are automatically checked and an error code is returned automatically during the transmission message. This routine is only supported by the Model 9010.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

duplexval [input] – This argument contains the new duplex setting. A value of "0" indicates that data should NOT be echoed back when transmitted. The only guarantee that the data arrived is if the ECU responds in some way. If the value is "1" then it indicates data should be echoed back, if transmitted properly on the bus, for baud rates greater than 10417.

2.17. ISO14230 USB FAST INIT

Function: Iso14230UsbFastInit

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbFastInit(
short handle,
unsigned char functional,
unsigned char alternate,
unsigned char source_address,
unsigned char target_address,
unsigned char *error_code);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbFastInit _
    Lib "14230USB.DLL" _
    (ByVal handle As Integer, _
    ByVal functional As Byte, _
    ByVal alternate As Byte, _
    ByVal source_address As Byte, _
    ByVal target_address As Byte, _
    ByRef error_code As Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
     Public Declare Function Iso14230UsbFastInit _
         Lib "14230USB.DLL" _
         (ByVal handle As System.Int16, _
         ByVal functional As Byte, _
         ByVal alternate As Byte, _
         ByVal source_address As Byte, _
         ByVal target_address As Byte, _
         ByVal error_code As System.IntPtr) _
      As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
     Public Declare Function Iso14230UsbFastInit _
         Lib "14230USB64.DLL" _
         (ByVal handle As System.Int16, _
         ByVal functional As Byte, _
         ByVal alternate As Byte, _
         ByVal source_address As Byte, _
         ByVal target_address As Byte, _
         ByVal error_code As System.IntPtr) _
      As System.Int32 'returns 1 if successful
```

Description: This function attempts a standard ISO-14230 "fast initialization" sequence on the K-line and L-line bus with the specified parameters taken into consideration. This sequence consists of a wakeup pulse (25 milliseconds low followed by 25 milliseconds high) followed by a Start Communications message transaction that includes receiving the Keyword of the device under test. The keyword can then be obtained using the Iso14230UsbGetMostRecentKeyword command if this routine returns successful. If the routine returns successful, then it can be assumed that the device under test is "awake" and ready to receive and transmit messages on the K-line. Note that this "awake" status usually changes to "asleep" after 5 seconds if the device under test does not receive a Tester Present message before that 5 seconds is up. Therefore it is usually necessary to setup the Tester Present message before sending the Initialization command. You can setup the Tester Present message with the functions Iso14230UsbSetContinualWakeup and Iso14230UsbSetCustomTesterPresentMessage.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

functional [input] – This argument can be set to either "1" or "0". Set this argument to "1" if the device under test recognizes Functional Addressing commands. Set this argument to "0" if the device under test recognizes Physical Addressing commands. If you are unsure which to set it to, consider the device under test's address. Is this a functional address or a physical address? If still in doubt, try both values at least once to see which one works to initialize the device.

alternate [input] – This argument can be set to either "1" or "0". Set this argument to "1" to indicate an alternate method of initialization used by manufacturers such as Kia. Use an argument of "0" to indicate the standard method of initialization. If you are in doubt about which argument to try, try both values at least once to see which one works to initialize the device.

source_address [input] – This is the address of the Test Device (the 14230/USB device), as understood by the device under test. Specific devices under test will only respond to specific source addresses so it is important to set this to the correct value. A typical value for this is hex F1.

target_address [input] – This is the address of the device under test (DUT) or embedded control unit (ECU) that the 14230/USB device is connected on the K-line with.

error_code [output] – This is the code indicating whether initialization was successful or failed. This will be equal to 4F hex if the initialization succeeded. It will be equal to some other value if a specific error occurred during initialization. For a list of possible error codes, see Part 4. To clear this error code, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.18. ISO14230 USB EXTENDED SLOW INIT

Function: Iso14230UsbExtendedSlowInit

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbExtendedSlowInit(
short handle,
unsigned char functional,
unsigned char alternate,
unsigned char target_address,
unsigned char source_address,
     unsigned char *error_code);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSlowInit _
    Lib "14230USB.DLL" _
    (ByVal handle As Integer, _
    ByVal functional As Byte, _
    ByVal alternate As Byte, _
    ByVal target_address As Byte, _
    ByVal source_address As Byte, _
    ByRef error_code As Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSlowInit _
    Lib "14230USB.DLL" _
```

```
          (ByVal handle As System.Int16, _
          ByVal functional As Byte, _
          ByVal alternate As Byte, _
          ByVal target_address As Byte, _
          ByVal source_address As Byte, _
          ByVal error_code As System.IntPtr) _
      As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
      Public Declare Function Iso14230UsbSlowInit _
          Lib "14230USB64.DLL" _
          (ByVal handle As System.Int16, _
          ByVal functional As Byte, _
          ByVal alternate As Byte, _
          ByVal target_address As Byte, _
          ByVal source_address As Byte, _
          ByVal error_code As System.IntPtr) _
      As System.Int32 'returns 1 if successful
```

Description: This function attempts either a standard or alternate ISO-14230 “slow initialization” sequence on the K-line and L-line bus with the specified parameters taken into consideration. This initialization involves a 5 baud transaction as well as a faster speed transaction that includes receiving the Keyword of the device under test. The keyword can then be obtained using the Iso14230UsbGetMostRecentKeyword command if this routine returns successful. If alternate is specified as “1”, and the device is a model 9010, then the System ID can also be obtained by using the routine Iso14230UsbGetMostRecentSystemID. Note that the alternate argument is ignored by the Model 9009. If the routine returns successful, then it can be assumed that the device under test is “awake” and ready to receive and transmit messages on the K-line. Note that this “awake” status usually changes to “asleep” after 5 seconds if the device under test does not receive a Tester Present message before that 5 seconds is up. Therefore it is usually necessary to setup the Tester Present message before sending the Initialization command. You can setup the Tester Present message with the functions Iso14230UsbSetContinualWakeup and Iso14230UsbSetCustomTesterPresentMessage.

Return Code: This function returns a standard 14230/USB return code. A return code of “1” indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

functional [input] – This argument can be set to either “1” or “0”. Set this argument to “1” if the device under test recognizes Functional Addressing commands. Set this argument to “0” if the device under test recognizes Physical Addressing commands. If you are unsure which to set it to, consider the device under test’s address. Is this a functional address or a physical address? If still in doubt, try both values at least once to see which one works to initialize the device.

alternate [input] – This argument can be set to either "1" or "0". Set this argument to "1" if the device requires Alternate Slow Init. Such devices, such as some Fiat ECU's, require that every byte transmitted is echoed back by the receiver. They require custom application software for sending/receiving messages but such software can be written using the existing framework. Set this argument to "0" for normal slow initialization. If still in doubt, try both values at least once to see which one works to initialize the device. Note that this argument is ignored for the Model 9009. This argument only has effect in the Model 9010.

target_address [input] – This is the address of the device under test (DUT) or embedded control unit (ECU) that the 14230/USB device is connected on the K-line with.

source_address [input] – This is the address of the Test Device (the 14230/USB device), as understood by the device under test. Specific devices under test will only respond to specific source addresses so it is important to set this to the correct value. A typical value for this is hex F1.

error_code [output] – This is the code indicating whether initialization was successful or failed. This will be equal to 4F hex if the initialization succeeded. It will be equal to some other value if a specific error occurred during initialization. For a list of possible error codes, see Part 4. To clear this error code, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.19. ISO14230 USB SLOW INIT

Function: Iso14230UsbSlowInit

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSlowInit(
short handle,
unsigned char functional,
unsigned char target_address,
unsigned char source_address,
     unsigned char *error_code);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSlowInit _
    Lib "14230USB.DLL" _
    (ByVal handle As Integer, _
    ByVal functional As Byte, _
    ByVal target_address As Byte, _
    ByVal source_address As Byte, _
    ByRef error_code As Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSlowInit _
    Lib "14230USB.DLL" _
```

```
        (ByVal handle As System.Int16, _
        ByVal functional As Byte, _
        ByVal target_address As Byte, _
        ByVal source_address As Byte, _
        ByVal error_code As System.IntPtr) _
    As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
    Public Declare Function Iso14230UsbSlowInit _
        Lib "14230USB64.DLL" _
        (ByVal handle As System.Int16, _
        ByVal functional As Byte, _
        ByVal target_address As Byte, _
        ByVal source_address As Byte, _
        ByVal error_code As System.IntPtr) _
    As System.Int32 'returns 1 if successful
```

Description: This function attempts a standard ISO-14230 "slow initialization" sequence on the K-line and L-line bus with the specified parameters taken into consideration. This function is deprecated: new users should use the function Iso14230UsbExtendedSlowInit. This function is equivalent to Iso14230UsbExtendedSlowInit with the alternate argument equal to 0. This initialization involves a 5 baud transaction as well as a faster speed transaction that includes receiving the Keyword of the device under test. The keyword can then be obtained using the Iso14230UsbGetMostRecentKeyword command if this routine returns successful. If the routine returns successful, then it can be assumed that the device under test is "awake" and ready to receive and transmit messages on the K-line. Note that this "awake" status usually changes to "asleep" after 5 seconds if the device under test does not receive a Tester Present message before that 5 seconds is up. Therefore it is usually necessary to setup the Tester Present message before sending the Initialization command. You can setup the Tester Present message with the functions Iso14230UsbSetContinualWakeup and Iso14230UsbSetCustomTesterPresentMessage.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

functional [input] – This argument can be set to either "1" or "0". Set this argument to "1" if the device under test recognizes Functional Addressing commands. Set this argument to "0" if the device under test recognizes Physical Addressing commands. If you are unsure which to set it to, consider the device under test's address. Is this a functional address or a physical address? If still in doubt, try both values at least once to see which one works to initialize the device.

target_address [input] – This is the address of the device under test (DUT) or embedded control unit (ECU) that the 14230/USB device is connected on the K-line with.

source_address [input] – This is the address of the Test Device (the 14230/USB device), as understood by the device under test. Specific devices under test will only respond to specific source addresses so it is important to set this to the correct value. A typical value for this is hex F1.

error_code [output] – This is the code indicating whether initialization was successful or failed. This will be equal to 4F hex if the initialization succeeded. It will be equal to some other value if a specific error occurred during initialization. For a list of possible error codes, see Part 4. To clear this error code, you need to call Iso14230UsbClearBuffers. Error codes will persist until cleared or overridden by a different error code.

2.20. ISO14230 USB GET MOST RECENT KEYWORD

Function: Iso14230UsbGetMostRecentKeyword

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbGetMostRecentKeyword(
short handle,
int *keyword);
```

VB6 Calling Format:

```
 Public Declare Function Iso14230UsbGetMostRecentKeyword _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByRef keyword as Integer) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbGetMostRecentKeyword _
  Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
  ByVal keyword As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbGetMostRecentKeyword _
  Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
  ByVal keyword As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

Description: This function is used to retrieve the raw keyword learned during the Fast Init or Slow Init process. Note that there are some very peculiar rules about how the keyword is encoded and these rules vary from application to application so this function will report the raw 2 bytes that are sent directly from the device under test without trying to interpret them in any way (such as to remove parity bits). It is up to the user or the application to determine what the actual keyword is based on what is returned from this routine. Note that if initialization has not occurred, this routine will return a keyword of 0. Note also that the keyword returned in the keyword argument is not valid unless this routine returns successfully.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

keyword [output] – This argument will be filled with the most recent keyword from the most recent successful fast or slow initialization sequence. It will contain 0 if no successful fast or slow initialization sequence has occurred since the last power-up. The 2-byte value written to the keyword argument will be the raw 2 bytes received from the device under test and no attempt to remove parity bits or any other interpretative operations will be performed. That is up to the application to interpret this keyword.

2.21. ISO14230 USB GET MOST RECENT SYSTEM ID (MODEL 9010 ONLY)

Function: Iso14230UsbGetMostRecentSystemID

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbGetMostRecentSystemID(
short handle,
int *systemID);
```

VB6 Calling Format:

```
 Public Declare Function Iso14230UsbGetMostRecentSystemID _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByRef systemID as Integer) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbGetMostRecentSystemID _
  Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
  ByVal systemID As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbGetMostRecentSystemID _
```

```
  Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
  ByVal systemID As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

Description: This function is used to retrieve the system ID learned during the last alternate Slow Init process. Note that if alternate slow initialization has not occurred, this routine will return a system ID of 0. Note also that the system ID returned in the system ID argument is not valid unless this routine returns successfully. This function is currently only supported by the Model 9010.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

systemID [output] – This argument will be filled with the most recent system ID from the most recent successful alternate slow initialization sequence. It will contain 0 if no successful alternate slow initialization sequence has occurred since the last power-up.

2.22. ISO14230 USB SET INTER BYTE DELAY

Function: Iso14230UsbSetInterByteDelay

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetInterByteDelay(
short handle,
unsigned char interbytedelay);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetInterByteDelay _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer, _
  ByVal interbytedelay as Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetInterByteDelay _
  Lib "14230USB.DLL" _
 (ByVal handle As System.Int16, _
 ByVal interbytedelay As Byte) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetInterByteDelay _
```

```
           Lib "14230USB64.DLL" _
          (ByVal handle As System.Int16, _
          ByVal interbytedelay As Byte) _
        As System.Int32 'returns 1 if successful
```

Description: This function is used to set the inter-byte delay for all messages sent on the K-line by the 14230/USB device. This delay specifies the amount of time to leave the K-line bus idle before sending the next byte of data. This delay is specified in number of milliseconds and can range from 0 to 51.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

interbytedelay [input] – This argument contains the new inter-byte delay in number of milliseconds. It can range from 0 to 51.

2.23. ISO14230 USB SET CONTINUAL WAKEUP

Function: Iso14230UsbSetContinualWakeup

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetContinualWakeup(
short handle,
unsigned char cw);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetContinualWakeup _
     Lib "14230USB.DLL" _
     (ByVal handle As Integer, _
     ByVal cw as Byte) _
    As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetContinualWakeup _
   Lib "14230USB.DLL" _
   (ByVal handle As System.Int16, _
   ByVal cw As Byte) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetContinualWakeup _
   Lib "14230USB64.DLL" _
```

```
          (ByVal handle As System.Int16, _
          ByVal cw As Byte) _
       As System.Int32 'returns 1 if successful
```

Description: This function is used to set a flag inside the 14230/USB device that indicates whether or not to send a Tester Present message on the K-line every 2 seconds.

When enabled, a Tester Present message is sent every 2 seconds and the status of each transmission is sent to the PC when the USB cable is connected and the device has been opened using Iso14230UsbOpen or Iso14230UsbExtendedOpen. This Tester Present message is generated internally inside the 14230/USB device and does not require a PC to maintain. This allows for greater accuracy in timing and consistency for putting out this message. Note that in order for this message to be sent out, there must be no current error code and there must be no traffic on the bus for at least 100 milliseconds.

The nature of the Tester Present message is either standard or custom and is defined by the routine Iso14230UsbSetCustomTesterPresentMessage.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

cw [input] – This argument should be set to either "0" or "1". Set this argument to "0" to indicate that NO tester present messages should be sent. Set this argument to "1" to indicate that a tester present message should be sent every 2 seconds from the 14230/USB device.

2.24. ISO14230 USB SET CUSTOM TESTER PRESENT MESSAGE

Function: Iso14230UsbSetCustomTesterPresentMessage

Library: 14230USB.DLL

C/C++ Calling Format:

```
  int __stdcall Iso14230UsbSetCustomTesterPresentMessage(
      short handle,
      short length,
      unsigned char *message_buffer);
```

VB6 Calling Format:

```
      Public Declare Function _
         Iso14230UsbSetCustomTesterPresentMessage _
        Lib "14230USB.DLL" _
        (ByVal handle As Integer, _
         ByVal length as Integer, _
        ByRef message_buffer as Byte) _
       As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
 Public Declare Function Iso14230UsbSetCustomTesterPresentMessage _
       Lib "14230USB.DLL" _
       (ByVal handle As System.Int16, _
       ByVal length As System.Int16, _
       ByVal message_buffer As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
 Public Declare Function Iso14230UsbSetCustomTesterPresentMessage _
       Lib "14230USB64.DLL" _
       (ByVal handle As System.Int16, _
       ByVal length As System.Int16, _
       ByVal message_buffer As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

Description: This function is used to set the contents of the Tester Present message that is sent internally from the ISO/14230 USB device. If the length argument is 0, this indicates that a standard Tester Present message will be used. The standard Tester Present message borrows the same arguments/understanding from the Fast Init or Slow Init commands, such as the functional argument and the source and target addresses. In addition, these are stored in the EEPROM so that on a power cycle, the tester present message will remain consistent. If the length argument is non-zero, then a custom Tester Present message can be defined in the message_buffer argument. This means basically any message (31 bytes or less) can be set up to repeat every 2 seconds. To enable or disable the sending of the Tester Present message use the function Iso14230UsbSetContinualWakeup.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

length [input] – This argument contains the length of the custom tester present message, or it is equal to 0 to indicate a standard tester present message should be used. Valid values for this argument are 0-31. If this argument is 0, then the message_buffer argument can contain a NULL pointer. If this argument is non-zero, then the message_buffer must be allocated for at least the number of bytes in the length argument and contains the message to be used for continual sending.

message_buffer [input] –This argument contains the contents of the custom tester present message. It is allocated with size defined in the length argument. If the length argument is 0, then the message_buffer argument can be the NULL pointer. If the length argument is non-zero, then the message_buffer argument is a buffer of bytes that contain the message to be sent continually, normally to indicate that the tester is present.

2.25. ISO14230 USB SET MAX BUF SIZE

Function: Iso14230UsbSetMaxBufSize

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbSetMaxBufSize(
unsigned short xmitsize,
     unsigned short recvsize);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbSetMaxBufSize _
 Lib "14230USB.DLL" _
  (ByVal xmitsize As Integer, _
  ByVal recvsize As Integer) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetMaxBufSize _
 Lib "14230USB.DLL" _
 (ByVal xmitsize As System.Int16, _
 ByVal recvsize As System.Int16) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbSetMaxBufSize _
 Lib "14230USB64.DLL" _
 (ByVal xmitsize As System.Int16, _
 ByVal recvsize As System.Int16) _
As System.Int32 'returns 1 if successful
```

Description: This function sets the maximum number of data buffers for any subsequent devices that are opened after this command is finished. Devices that were open previously will use the old maximum number of data buffers settings. The maximum number of buffers defaults to 8192 for the transmit buffer queue (xmitsize) and 16384 for the receive buffer queue (recvsize).

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

xmitsize [input] – This parameter controls the number of transmit buffers that will be kept for each 14230/USB device. The transmit buffer holds messages that have not been sent yet and are waiting for the first available moment to be transmitted to the 14230/USB device. The default value is 4; it is assumed that for the most part messages are transmitted one at a time until the acknowledgement comes back so there is not usually a need for a large number of transmit buffers.

recvsize [input] – This parameter controls the number of receive buffers that will be kept for each 14230/USB device. The receive buffer holds messages that have not been processed yet by the PC. To prevent a receive buffer overflow, the program must constantly check to see if any messages (status, data, and ack's) are waiting and then to read them out so as to keep the receive queue clear. The number of receive messages that can be queued defaults to 16384; a large number that allows for the many time-consuming latencies of the Windows OS to make sure the messages are not lost before the Windows OS finally has time to process them.

2.26. ISO14230 USB CLEAR BUFFERS

Function: Iso14230UsbClearBuffers

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbClearBuffers(
short handle);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbClearBuffers _
  Lib "14230USB.DLL" _
  (ByVal handle As Integer) _
As integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbClearBuffers _
  Lib "14230USB.DLL" _
 (ByVal handle As System.Int16) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbClearBuffers _
  Lib "14230USB64.DLL" _
 (ByVal handle As System.Int16) _
As System.Int32 'returns 1 if successful
```

Description: This function has multiple purposes. It performs the following operations:

(1) It clears all receive and transmit buffers at all stages of the Windows PC DLL and 14230/USB device. This flushes out all pending requests and transmissions. (2) It clears the current device Error Code. It is reset to "No Error" which is a 4F error code. (This is the error code that can be read using Iso14230UsbReadStatus).

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.27. ISO14230 USB READ FIRMWARE REVISION

Function: Iso14230UsbReadFirmwareRevision

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbReadFirmwareRevision(
short handle,
unsigned char *rev_major,
unsigned char *rev_minor);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbReadFirmwareRevision _
Lib "14230USB.DLL" _
 (ByVal handle As Integer, _
 ByRef major_version As Byte, _
 ByRef minor_version As Byte) _
 As Integer 'returns 1 if successful
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbReadFirmwareRevision _
  Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
  ByVal major_version As System.IntPtr, _
  ByVal minor_version As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbReadFirmwareRevision _
  Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
  ByVal major_version As System.IntPtr, _
  ByVal minor_version As System.IntPtr) _
As System.Int32 'returns 1 if successful
```

Description: This function reads the major and minor firmware version of the 14230/USB device. The major version refers to the version of the USB microcontroller whereas the minor version refers to the version of the ISO-14230 microcontroller. Versions range from 1-255.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being used. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

major_version [output] – This argument should point to a byte that the routine will write to. The routine will write the current major version number to this byte which corresponds to the USB microcontroller firmware version.

minor_version [output] – This argument should point to a byte that the routine will write to. The routine will write the current minor version number to this byte which corresponds to the ISO-14230 microcontroller firmware version.

2.28. ISO14230 USB CLOSE

Function: Iso14230UsbClose

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbClose(
short handle,
unsigned char SN[6]);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbClose _
    Lib "14230USB.DLL" _
    (ByVal handle As Integer, _
     ByRef SerialNumber As Byte) _
  As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbClose _
  Lib "14230USB.DLL" _
  (ByVal handle As System.Int16, _
  ByVal SerialNumber As System.IntPtr) _
As System.Int32 'returns 1 if ok
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbClose _
  Lib "14230USB64.DLL" _
  (ByVal handle As System.Int16, _
  ByVal SerialNumber As System.IntPtr) _
As System.Int32 'returns 1 if ok
```

Description: This function closes a handle to a specified 14230/USB device and allows other programs to connect to the device. It should be called when the user is done working with a specific 14230/USB device but still needs to use the ISO14230USB DLL for other operations, possibly with other 14230/USB devices. Note that if a program is completely done with all 14230/USB device handling, the user should call Iso14230UsbUnload instead.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being closed. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

2.29. ISO14230 USB UNLOAD

Function: Iso14230UsbUnload

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbUnload(void);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbUnload _
    Lib "14230USB.DLL" _
    () _
  As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbUnload _
   Lib "14230USB.DLL" _
   () _
As System.Int32 'returns 1 if ok
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbUnload _
   Lib "14230USB64.DLL" _
   () _
As System.Int32 'returns 1 if ok
```

Description: This function MUST be called when the application is done using the ISO14230USB DLL. This terminates all open handles and all threads that are used to monitor those handles. It allows other applications in the future to use the 14230/USB devices that were used by the current application, which until now have locked out other applications from using those devices.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

None.

2.30. ISO14230 USB HIGH SPEED BAUD RATE

Function: Iso14230UsbHighSpeedBaudRate

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbHighSpeedBaudRate(
short handle,
unsigned long *baudrate);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbHighSpeedBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As Integer, _
   ByRef BaudRate As Long) _
   As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbHighSpeedBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As System.Int16, _
   ByVal BaudRate As System.IntPtr) _
As System.Int32 'returns 1 if ok and BaudRate filled in
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbHighSpeedBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As System.Int16, _
   ByVal BaudRate As System.IntPtr) _
As System.Int32 'returns 1 if ok and BaudRate filled in
```

Description: This function can be used at any time if the user wants to know what baud rate the 14230/USB device is currently operating at. Upon successful return, the BaudRate argument is filled in with the current baud rate of the K-line bus that the 14230/USB device is configured at.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being closed. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

BaudRate [output] – Upon successful completion, the integer pointed to by this argument will be filled in by the routine to contain the baud rate that the 14230/USB device is currently set to (in units of bits per second).

2.31. ISO14230 USB BAUD RATE

Function: Iso14230UsbBaudRate

Library: 14230USB.DLL

C/C++ Calling Format:

```
int __stdcall Iso14230UsbBaudRate(
short handle,
short *baudrate);
```

VB6 Calling Format:

```
Public Declare Function Iso14230UsbBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As Integer, _
   ByRef BaudRate As Integer) _
   As Integer 'returns 1 if ok
```

VB .NET 2005 32-Bit Calling Format:

```
Public Declare Function Iso14230UsbBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As System.Int16, _
   ByVal BaudRate As System.IntPtr) _
As System.Int32 'returns 1 if ok and BaudRate filled in
```

VB .NET 2005 64-Bit Calling Format:

```
Public Declare Function Iso14230UsbBaudRate _
   Lib "14230USB.DLL" _
   (ByVal handle As System.Int16, _
   ByVal BaudRate As System.IntPtr) _
As System.Int32 'returns 1 if ok and BaudRate filled in
```

Description: This function can be used at any time if the user wants to know what baud rate the 14230/USB device is currently operating at. This function is deprecated: new users should use the function Iso14230UsbHighSpeedBaudRate because this function is incapable of returning a baud rate higher than 32,767. Upon successful return, the BaudRate argument is filled in with the current baud rate of the K-line bus that the 14230/USB device is configured at.

Return Code: This function returns a standard 14230/USB return code. A return code of "1" indicates success. Other values indicate specific errors (see Part 3).

Parameter Description:

handle [input] - This is the handle of the 14230/USB device being closed. Only valid handles obtained using Iso14230UsbOpen or Iso14230UsbExtendedOpen are accepted by this routine.

BaudRate [output] – Upon successful completion, the integer pointed to by this argument will be filled in by the routine to contain the baud rate that the 14230/USB device is currently set to (in units of bits per second). Note: only the low 15 bits of the baud rate are guaranteed to be returned by this function: users should use the function Iso14230UsbHighSpeedBaudRate for new applications.

3. ISO14230 USB DLL RETURN CODES

3.1. DLL RETURN CODES

1	SUCCESS – OK
2	GENERAL ERROR – USUALLY COMMUNICATIONS ERROR
3	RWERR – DEVICE DRIVER/OS I/O ERROR
4	RANGE – ERROR IN ARGUMENT OUT OF RANGE IN ROUNTINE IN ISO14230USB DLL
5	MEMERR – OUT OF MEMORY
6	NODEV – NO 14230/USB DEVICE DETECTED
7	NONFCE – NO 14230/USB DEVICE INTERFACE DETECTED, IMPROPER INSTALLATION
8	NODETL – NO 14230/USB DEVICE DETAIL DETECTED, SYSTEM ERROR
9	NODETL2 – NO 14230/USB DEVICE DETAIL DETECTED, SYSTEM ERROR
10	NOHNDLS – NO MORE OS HANDLES ARE AVAILABLE TO USE
11	NORDHD – READ ERROR, UNUSED ERROR CODE
12	NOWRHD – WRITE ERROR, UNUSED ERROR CODE
13	NCTS – CLEAR TO SEND WAS DEACTIVE FOR A LONG PERIOD OF TIME INDICATING DEVICE IS TERRIBLY BUSY, STUCK, OR NON-FUNCTIONAL
14	NOORIG – FATAL ERROR – COULD NOT START 14230 READ THREAD
15	BOLA – THE PC RECEIVE BUFFER OVERFLOWED CONSTANTLY AND AS A RESULT THE ACKNOWLEDGEMENT TO THE COMMAND JUST SENT WAS LOST
16	LOCKERR – INTERNAL OS ERROR USING SYNCHRONOUS MUTEX'S
17	RESERVED – NOT USED
18	BAD HANDLE – AN IMPROPER HANDLE WAS PASSED TO ONE OF THE FUNCTIONS.

4. ISO-14230 USB DEVICE ERROR CODES

4.1. DEVICE ERROR CODES

00h Failed to bring both the K and L bus lines low. Both the K and L lines were held high by external hardware or perhaps by defective hardware or excessive capacitance on the K-line. This came about when the K and L lines were both asserted low during an initialization sequence that was commanded by the PC.

01h Failed to bring the K line low. The K line was held high by external hardware or perhaps by defective hardware or excessive capacitance on the K-line. This came about when the K line was asserted low during a message transmission or initialization sequence that was commanded by the PC.

02h Failed to bring the L line low. The L line was held high by external hardware or perhaps by defective hardware or excessive capacitance on the L-line. This came about when the L line was asserted low during an initialization sequence that was commanded by the PC.

03h Failed to bring both the K and L bus lines high. Both the K and L lines were held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on both the K and L lines. This came about when the K and L lines were both asserted high during an initialization sequence that was commanded by the PC.

04h Failed to bring the K bus line high. The K line was held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on the K line. This came about when the K line was asserted high during an initialization sequence or a message transmission that was commanded by the PC.

05h Failed to bring the L bus line high. The L line was held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on the K line. This came about when the L line was asserted high during an initialization sequence or message transmission that was commanded by the PC.

06h K and L bus lines are stuck low. Both the K and L lines were held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on both the K and L lines. This came about when the K and L lines were both asserted high during an initialization sequence that was commanded by the PC.

07h K bus line is stuck low. The K line was held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on the K line. This came about when the K line was asserted high during an initialization sequence or a message transmission that was commanded by the PC.

08h L bus line is stuck low. The L line was held low by external hardware, another device trying to transmit on the bus at the same time, or excessive capacitance on the K line. This came about when the L line was asserted high during an initialization sequence or message transmission that was commanded by the PC.

09h No response received to initialization. This indicates that there is no device under test connected to the K and/or L lines and/or the device under test does not recognize the particular form of initialization that has been tried and/or the wrong parameters have been supplied to the initialization function.

0Ah	An improper synchronization byte was received. This indicates the device-under-test did not recognize the initialization sequence and sent some strange data that is not in compliance with the standard initialization sequence. This could also indicate that the device under test is already awake and does not need to be initialized, if it sent some data that is normally sent in run mode, or that the device under test is improperly programmed or is set to a different baud rate.
0Bh	An improper inverse address byte was received. This indicates the device-under-test did not recognize the initialization sequence and sent some strange data that is not in compliance with the standard initialization sequence. This could also indicate that the device under test is already awake and does not need to be initialized, if it sent some data that is normally sent in run mode, or that the device under test is improperly programmed or is set to a different baud rate. It could also indicate the device under test is expecting a different address be specified in the Fast Init or Slow Init function arguments.
0Ch	An improper Start Communications Positive Response was received. This indicates the device-under-test did not recognize the initialization sequence and sent some strange data that is not in compliance with the standard initialization sequence. This could also indicate that the device under test is already awake and does not need to be initialized, if it sent some data that is normally sent in run mode, or that the device under test is improperly programmed or is set to a different baud rate.
0Dh	An improper Start Communications Positive Response checksum was received. This indicates the device-under-test did not recognize the initialization sequence and sent some strange data that is not in compliance with the standard initialization sequence or uses a different checksum method than what is specified in the ISO-14230 specification. This could also indicate that the device under test is already awake and does not need to be initialized, if it sent some data that is normally sent in run mode, or that the device under test is improperly programmed or is set to a different baud rate.
0Eh	The bus was busy. This indicates that a command to send a message or try an initialization sequence was tried but that the bus is so full of traffic (close to 100% bus utilization) that there was never an opportunity to try the command.
20h	INVALID CHECKSUM FROM PC. This indicates the PC has sent the wrong checksum on an internal command frame. This could indicate a faulty ISO14230USB DLL or an abrupt termination of an ISO14230USB DLL that was then restarted.
21h	INVALID COMMAND FROM PC. This indicates the PC has sent an invalid command on an internal command frame. This could indicate a faulty ISO14230USB DLL or an abrupt termination of an ISO14230USB DLL that was then restarted.
30h	RECEIVE BUFFER OVERFLOW. This indicates the 14230/USB device receive buffer (containing received message data, acknowledgements, and status updates) has overflowed. This can happen when a USB device is opened on a PC but then ignored such as when another USB device is selected, or when a USB device is opened on a PC and then made to listen during heavy, heavy bus traffic for a long period of time.
31h	TRANSMIT BUFFER OVERFLOW. This indicates the 14230/USB device transmit buffer (containing commands and requests) has overflowed. If this is occurring then the user should take care to wait for an Iso14230Usb… routine (any routine starting with "Iso14230Usb") to return before sending the next Iso14230Usb… command.
4Fh	NO ERROR. Everything is okay.
50h	TRANSMIT BUS OVERFLOW. This indicates that the data queued up to send on the 14230 bus has overflowed the 14230 bus transmit queue. If this has occurred, the user should take care to wait for an Iso14230Usb… routine to return before sending the next Iso14230Usb… command.

51h	RECEIVE BUS OVERFLOW. This indicates the receive queue for the 14230 bus has overflowed. This is an extremely rare occurrence. This would indicate that too much data has been received on the 14230 bus before the 14230/USB device has been able to process it, which has never been observed to happen under normal circumstances.
60h	EEPROM WRITE VERIFY ERROR. This indicates that after writing to Non-Volatile memory aboard the 14230/USB device, a subsequent check of the write failed to verify the data that was written indicating that the 14230/USB device should be replaced (or at least the small Non-Volatile memory chip connected to the MCU should be replaced). Contact Silicon Engines for assistance.
E0h	ROM CHECKSUM ERROR. This indicates the firmware was not properly installed on this device. The user should reinstall the firmware on the device.
E1h	EEPROM LOCKOUT ERROR. This indicates the firmware is busy reading/writing to the EEPROM and cannot fulfill the current request. If this error condition persists even after repeated Iso14230UsbClearBuffers calls then the module should be returned to Silicon Engines for analysis with explicit steps necessary to recreate the problem included.

5. ACKNOWLEDGEMENT COMMAND BYTES

5.1. ACKNOWLEDGEMENT COMMAND BYTES

These numbers might be seen during a call to Iso14230UsbAckWaiting. They indicate Acknowledgements from the 14230/USB device to commands that were sent long ago. Each Iso14230Usb… routine waits a certain amount of time for a command to complete. During this time, an acknowledgement is sent and recognized positively by the routine which then returns a successful return code. However there are some situations where the acknowledgement is not received during the time the Iso14230Usb… routine has allocated for waiting and the routine will return a FAILURE but the 14230/USB device still has the command queued and possibly sends the acknowledgement later on. In those cases, this chart will help a user to determine what command was just performed by the 14230/USB device in those cases where it was long stalled and not able to act on the current transmit queue until the moment the acknowledgement come back.

3Fh	Corresponds to Iso14230UsbReadStatus command
42h	Corresponds to Iso14230UsbSetBaudRate command
44h	Corresponds to Iso14230UsbSetHighSpeedDuplex command
49h	Corresponds to Iso14230UsbSetInterByteDelay command
58h	Corresponds to Iso14230UsbClearBuffers command
90h	Corresponds to message transmission command
91h	Corresponds to Iso14230UsbSetContinualWakeup command
92h	Corresponds to Isol14230UsbSetCustomTesterPresentMessage command
93h	Corresponds to Iso14230UsbSetBusCharacteristics command
99h	Corresponds to message transmission final launch command
A0h	Corresponds to a write to EEPROM command
A1h	Corresponds to a read EEPROM command

A2h	Corresponds to a read EEPROM single byte command
A6h	Corresponds to a program firmware operation, such as from the 14230 USB Message Center
ADh	Corresponds to a Model 9009 prepare program firmware operation, such as from the ISO14230USB Message Center
AEh	Corresponds to a program Model 9009 JB16 firmware operation such as from the ISO14230USB Message Center
AFh	Corresponds to a program Model 9009 GR16 firmware operation such as from the ISO14230USB Message Center
C6h	Corresponds to a program Model 9010 firmware operation such a from the ISO14230USB Message Center.
CDh	Corresponds to a Model 9010 prepare program firmware operation, such as from the ISO14230USB Message Center
CFh	Corresponds to a finish programming Model 9010 firmware operation such as from the ISO14230USB Message Center

6. REVISION HISTORY

6.1. REVISION D

1. Add Iso14230USBExtendedOpen.
2. Add Iso14230USBSetHighSpeedBaudRate.
3. Add Iso14230USBHighSpeedBaudRate.
4. Iso14230USBSetBusCharacteristics has extra bit definitions for model 9010.
5. Add Iso14230USBSetHighSpeedDuplex.
6. Add Iso14230USBExtendedSlowInit.
7. Add Iso14230USBGetMostRecentSystemID.
8. Iso14230UsbOpen description modified to describe limitation to this routine and that Iso14230UsbExtendedOpen should be used.
9. Iso14230UsbFastInit switch the order of arguments for source_address and target_address. This matches actual functionality in the firmware of the model 9009 and model 9010.
10. Every occurrence of Iso14230UsbOpen replaced with "Iso14230UsbOpen or Iso14230UsbExtendedOpen".
11. Iso14230UsbSetMaxBufSize routine description modified to include correct new default for number of transmit buffers that is used with the most recent drivers.
12. Updated section 3.1 to include new return code 18.
13. Updated section 5.1 to include new acknowledgement codes.

6.2. REVISION C

14. Rename document to clarify that this manual applies to both Model 9010 and Model 9009.
15. Add Model 9010 illustration to first page.

16. Change references to Model 9009 to Models 9009 and 9010 throughout.
17. Include proper declarations for VB .NET 2005 32-bit and 64-bit.
18. Change all typo references to ISO14230USB.DLL to 14230USB.DLL.

6.3. REVISION B

1. Sec. 2.8, Iso14230UsbReadDataMsg: Updated the description of Iso14230UsbReadDataMsg to reflect new functionality as of Rev. 1.0.2 of the PC-side software. This command now is more flexible and can return more bytes--it can return the actual total number of bytes that are in the receive queue.

6.4. REVISION A

Initial release.

☑