Delcom Products Inc. **Application Note 205**

Title:
Serial I2C EEPROM read/write example.

Document identifier:
Application Note 205

Location:
http://www.delcom-products.com/downloads/appnote205.pdf

Author:
Douglas Lovett

Editors:

Contributors:

Abstract:
This document illustrates how to read and/or write a Serial I2C EEPROM with the Delcom USB Chip Set.

Effects:
Supports DelcomDLL version 0.6 and greater.

Status:

Table of Contents

1 Introduction

This application note illustrates how to read and/or write a serial I2C eeprom. Here we show two methods the first uses the DelcomDLL and second method illustrates what the DelcomDLL code looks like. This code can be convert to communicate directly to the Delcom USB chip set driver.

2 Solutions

2.1 DelcomDLL Solution

```
// This example writes 2048 bytes to a I2C eeprom device at the
// starting address of 0x0. The write delay is 10ms.
// EEPROM Part: PIC 24LC16B, CtrlCode Nibble = 0xA
#include "DelcomDLL.h"
#define MemSize 2048

   char DeviceName[1024];
   char Data[MemSize];
   int Found, i;
   HANDLE hUsb = 0;              // global device handle to the the USB device


   Found = DelcomGetNthDevice(USBIODS, 0, DeviceName);
   hUsb = DelcomOpenDevice(DeviceName,0);

   for(i=0; i<MemSize; i++) Data[i] = (BYTE)i;  // pre-fill buffer with 0-255
   // Write 2048 byte to EEPROM at address of 0x0, CrtlCode=0xA, WrDelay=10ms
   DelcomWriteI2CEEPROM(hUsb, 0x0,  MemSize, 0xA, 10, Data);

   // Read all 2048 byte from the EEPROM, at start address 0x0
   DelcomReadI2CEEPROM(hUsb, 0, MemSize, 0xA, Data);

   DelcomCloseDevice(hUsb);                     // Always close the device
```

2.2 Direct Solution

```
//-------------------------------------------------------//
DWORD CUsbCode::WriteI2C(HANDLE hUsb, BYTE CmdAdd, BYTE Length, LPSTR DataExt)
{
          Packet.Recipient = 8;
          Packet.DeviceModel = 18;
          Packet.Length = Length;
          Packet.MajorCmd = 10;
          Packet.MinorCmd = 60;
          Packet.DataLSB = CmdAdd;
          Packet.DataMSB = Length;
          memcpy(Packet.ExtData, DataExt, Length);
          return( SendPacket(hUsb,&Packet,NULL) );
}

//-------------------------------------------------------//
DWORD CUsbCode::ReadI2C(HANDLE hUsb, BYTE CmdAdd, BYTE Length, LPSTR DataExt)
{
          Packet.Recipient = 8;
          Packet.DeviceModel = 18;
          Packet.Length = Length;
          Packet.MajorCmd = 11;
          Packet.MinorCmd = 60;
```

```
          Packet.DataLSB = CmdAdd;
          Packet.DataMSB = Length;
          return(SendPacket(hUsb,&Packet,(pPacketStruct)DataExt) );
}

//-------------------------------------------------------//
DWORD CUsbCode::SelReadI2C(HANDLE hUsb, BYTE SetAddCmd, BYTE Address, BYTE
ReadCmd, BYTE Length, LPSTR DataExt)
{

          Packet.Recipient = 8;
          Packet.DeviceModel = 18;
          Packet.Length = 0;
          Packet.MajorCmd = 10;
          Packet.MinorCmd = 61;
          Packet.DataLSB = SetAddCmd;
          Packet.DataMSB = Address;
          if(SendPacket(hUsb,&Packet,NULL))
                 return(1);
          Packet.Recipient = 8;
          Packet.DeviceModel = 18;
          Packet.Length = Length;
          Packet.MajorCmd = 11;
          Packet.MinorCmd = 61;
          Packet.DataLSB = ReadCmd;
          Packet.DataMSB = Length;
          return(SendPacket(hUsb,&Packet,(pPacketStruct)DataExt) );

}


//-------------------------------------------------------//
// ReadI2CEEPROM() - Reads data from a EEPROM serial rom   v0.6
// Address - the start address
// Length - the length of the data, must be in multiplyes of 8. 8,16,24,....
// CtrlCode the top nibble of the command byte
// ptr - pointer to data DECLARED by user, must be at least the size ofr
length...
// return - zero on success
DWORD CUsbCode::ReadI2CEEPROM(HANDLE hUsb, DWORD Address, DWORD Length, BYTE
CtrlCode,  LPSTR ptr)
{
   DWORD Result, i;
   CtrlCode = CtrlCode << 4;     // shift up the control code to the top nibble

   // read the first 8 bytes with
   Result = SelReadI2C( hUsb,    CtrlCode|((BYTE)((0x07 &(Address>>8))<<1)),
          // Write command with bank address set
                                                      BYTE(Address&0xFF),
                                          // LSB of the address
                                                      0x01
|(CtrlCode|(BYTE)((0x07&(Address>>8))<<1)),     // Read command with bank
address set
                                                      8, ptr );
                                                // Eight bytes data size and
data ptr
   if( Result ) return(Result);
   if( Length <= 8 ) return(0);

   for(i=8; i<Length; i+=8){                    // sequential reads
          Address += 8;
          Result = ReadI2C( hUsb, 0x01 | (CtrlCode | (BYTE) ((0x07 &
(Address>>8))<<1)), 8, &ptr[i] );
          if( Result ) return(Result);
          }

   return(0);

}
```

```
//-------------------------------------------------------//
// WriteI2CEEPROM() - Write data to an EEPROM serial rom
// DELCOM DLL version - v0.6
// Writes a maximum of 4 bytes at a time. Limit due to 8 byte data packet size
// Program sizes other than four did not same to work on the EEPROM device
// Address - the start address of the eeprom
// Length - length of the data
// CtrlCode the top nibble of the command byte
// ptr - pointer to data DECLARED by user, must be at least the size of
length...
// WriteDelay - time to wait after the write cycle, units ms, typically 5 or 10
ms
// return - zero on success
DWORD CUsbCode::WriteI2CEEPROM(HANDLE hUsb, DWORD Address, DWORD Length, BYTE
CtrlCode,  BYTE WriteDelay, LPSTR ptr)
{
   #define  MAX_WR_BYTE_SIZE 4

   DWORD Time2Wait, Result, i, WrSize;
   char DataExt[8];
   CtrlCode = CtrlCode << 4;     // shift up the control code to the top nibble

   for(i=0; i<Length; ){  // writes four bytes of data at a time
           DataExt[0] = (BYTE)(0xFF&Address);                          // LSB
address to start writing
           DataExt[1] = ptr[i+0];                                      // Data
           DataExt[2] = ptr[i+1];
           DataExt[3] = ptr[i+2];
           DataExt[4] = ptr[i+3];

           WrSize = Length - i;
           if( WrSize > MAX_WR_BYTE_SIZE) WrSize = MAX_WR_BYTE_SIZE;
           Result = WriteI2C( hUsb, CtrlCode |  ( ((BYTE)(0x07 &
(Address>>8))<<1)), (BYTE)WrSize +1, DataExt );
           if( Result ) return(Result);
           Time2Wait = GetTickCount() + WriteDelay;     // must wait for write
time before sending next write
           while( GetTickCount() <= Time2Wait );        // 5ms for 24LC16B, some
eeproms are 10ms
           Address += WrSize;
           i += WrSize;
           }

   return(0);
}



// Send the USBIODS packet to the USB Device & returns data if available.
// Return value
int CUsbCode::SendPacket( HANDLE hUsb, pPacketStruct pTx, pPacketStruct pRx )
{
   unsigned long nBytes,RxLen;
   BOOLEAN success;

   if(!hUsb)      {
           if(Verbose)
                   MessageBox(NULL,VerboseHeader, "SendPacket Failed: Device not
open!", MB_ICONHAND);
           return(0);
           }

   pTx->Recipient = 8;           // Always 8 for the USBIODS device
   pTx->DeviceModel= 18;         // Always 18 for the USBIODS device
   if(pRx==NULL)  RxLen = 0;
   else                   RxLen = 8;

   success = DeviceIoControl(hUsb,                     // call the send packet
function
              IOCTL_USBIO_SEND_PACKET,
```

```
            pTx, 8+pTx->Length, pRx, RxLen, &nBytes, NULL );


   if( !success )
          {
          if(Verbose)    {
                 LPVOID lpMsgBuf;
                 FormatMessage(
                        FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,
                        NULL, GetLastError(),
                        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default
language
                        (LPTSTR) &lpMsgBuf, 0, NULL );
                 Close(hUsb);          // Close before displaying message
otherwise the timer will try another command
                                                        // Readded this command
in v0.6, before it was commented out
                 // Display the string.
                 char msg[512];
                 msg[0]=NULL;
                 strncpy(msg,VerboseHeader,128);
                 strncat(msg,"Failed to send packet.",512);
                 MessageBox(NULL,(LPTSTR)lpMsgBuf,msg,MB_ICONHAND);
                 LocalFree( lpMsgBuf );  // Free the buffer.
                 }
          return(1);
          }

   return(0);
}
```

3 References

Delcom USB IO Chips - http://www.delcom-eng.com/products_USBIO.asp

Delcom Distribution Disk - http://www.delcom-eng.com/delcomcd/Start.htm

Delcom DLL - http://www.delcom-eng.com/productdetails.asp?PartNumber=890510

Appendix A. Revision History

Rev	Date	Author	Description
0	02/12/2007	Doug Lovett	Initial Release

Appendix B. Notices

DELCOM PRODUCTS INC. takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on DELCOM PRODUCTS INC procedures with respect to rights in DELCOM PRODUCTS INC. specifications can be found at the DELCOM PRODUCTS INC. website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers' or users of this specification, can be obtained from the DELCOM PRODUCTS INC. Executive Director.

DELCOM PRODUCTS INC. invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the DELCOM PRODUCTS INC. Executive Director.