DAILY RFID DL990 LF RFID Reader

User’s Manual

DL990 LF RFID Reader use analogue serial Connection,COM analogue is in the toppest of the protocol groove, the dummy com is provided by RFCOMM .
Once the com is established, then the operation is the same as a serial method , it can be regarded as an established COM.

For the development of RFID , the following steps shall be taken

1. matching PPC with RFID reader (using bluetooth manager), matching code is 1234.
2. IF the matching is successful, PPC shall have an dummy COM connecting RFID reader (usually com6 or com8, but this may differ according to different types of machines)
3. now we can open testing program , which is actually a reading COM program. Click ‘reset’, set COM and its rate (9600).
4. click ‘open com’, RFID reader will be turned blue if connection is good, then click ‘receive’, data of hexadecimal system will be shown, meaning ‘ receiving state’.
5. Now RFID reader reads electrical tag, the number will be shown in the receiving area.

When writing programs, pay attention to the following questions:
The received RFID data will be a serial number of hexadecimal system. After this data is received,it need be changed. We do it in this way（C++）：.

```
  int length=(int)lParam; // received data length
  char *data=(char*)wParam; // data
   CString m_strDataReceived // after changed
   CString THex(_T(""));
int i;
if(length!=0)
{
    if(HexDisplay)
    {
        for(i=0;i<length;i++)
        {
            if(data[i]<10)
```

```
                {
                    THex.Format(_T(" %2x"),data[i]);
                }
                else
                {
                    THex.Format(_T("%x"),data[i]);
                }
                m_strDataReceived+=THex.Right(3);
            }
        }
```

Appendix: intrduction of some of serial com visiting functions（CE API）:

l Open serial com

```
hSerial = CreateFile( _T("COM1:"),
                       GENERIC_READ | ENERIC_WRITE,
                       0,
                       NULL,
                       OPEN_EXISTING,
                       0,
                       NULL);

if (hSerial == NULL )      //failed opening of serial com.
{
      return;
}
```

- Ø Creat File is used to open Serise Com equipment of all the fluid equipment drive programs.
- Ø Add number behind COM, then add a colon. Colon (:) is necessary in WINDOWS CE , thus to differ from desk machine naming code.
- Ø 0 must be translated to dwShareMode and hTemplateFile. CE does not support the overlapping IO of the eqipement, so FILE_FLAG_OVERLAPPED tag can not be translated in dwFlagsAndAttributes.
- Ø notice: the returned sentence handles are the opened sentence handles or INVALID HANDLE_VALUE. when they are opened unsuccessfully,

createfile will not return to O.this is different from most of window's functions.

- close serial

```
if ( hSerial != NULL )
{
     CloseHandle ( hSerial );
     hSerial = NULL;
}
```

use CloseHandle can close a serial com， the parameter is the com handle which will be opened by CreateFile .

- **For the normal use of coms, they must be set up well. A simple method is to use the two functions of GetCommState and SetCommState .DCB shall be initialized before the two functions being used .**

```
DCB PortDCB;
PortDCB.DCBlength = sizeof(DCB);
```

most of the defunct setting up of coms need not be revised. Commonly the rate, checking up of parity ect.

```
GetCommState ( hSerial, &PortDCB);
PortDCB.BaudRate = 115200;       // baud
PortDCB.ByteSize = 8;            // Number of bits/byte, 4-8
PortDCB.Parity = NOPARITY;       // parity
PortDCB.StopBits = ONESTOPBIT;
if ( ! SetCommState ( hSerial, &PortDCB ) )      // failed in setting up coms
{
     return;
}
```

Transmitting data to com.

-

```
WriteFile ( hSerial,      // Sentence handle
            &Byte,     // Data buffering address
               nByte,        // Size of data
               &dwNumBytes,       // return to the sent-off bytes
               NULL        // not support overlapping
            );
```

Ø Data will be transmitted successfully generally.

- But when the terminal eqipement needs some processing time of reacting , the time between the two writing operaton time must not be too short
- The exact reacting time is determined by the reacting time of terminal equipment and is related to the size of the buffering zone.
- CE does not support overlapping IO. It may cause the thread clogged due to waiting for the relative slow serial writing and reading, so it can not deal with other information in other windows.
- It is better to use solo thread to write or read serial COM.Read data of serial coms.

read serial com data

- Just as said in help file of Window Mobile . Serial com momunication is one of the simplest communications .but it has its trouble , besides thinking about receiving data in time, we also have to think about processing the received data.if you arrange the processing work in reading the thread,the data may be lost (terminal equipment has sent the data but it is not reveived). On the other hand, the data may not be lost (the time and length of data sent by terminal equipment is regular.). if we can ensure the the would-be received data can only be sent by terminal equipment after the processing is finished, then we can arrange the processing in reading thread.

```
//   if the received data are all bytes
UINT ReadThread ( LPVOID pParam )      // receive serial data thread
{
     HANDLE hPort = *(HANDLE*)pParam;

     BYTE Byte;

     int iCounter = 0;

     DWORD dwBytes;

     char ReceiveBuf[1000];          // size of buffering zone

SetCommMask ( hPort, EV_RXCHAR );    //only receive bytes

     while ( hPort != INVALID_HANDLE_VALUE )
     {
          DWORD dwCommStatus;

          WaitCommEvent(hPort, &dwCommStatus, 0);
```

```
            SetCommMask (hPort, EV_RXCHAR);
            // reset the waiting signal
//    receive data
        do
        {
            ReadFile(hPort, &Byte, 1, &dwBytes, 0);
            if(dwBytes == 1)
            {
                ReceiveBuf[iCounter++] = Byte;
                if ( iCounter == 1000)    // buffering zone is full
                {
                    return -1;
                }
            }
        } while (dwBytes == 1);

if ( iCounter == 0)      // data not received.
        {
            continue;
        }
        l  //save data
           char* pTmp = new char[iCounter + 1];

           if(pTmp == NULL)      //memory is not big enough，the programm of
           receiving serial Com is closed.
           {
            return -1;
           }

           memcpy(pTmp, ReceiveBuf, iCounter);
           pTmp[iCounter] = NULL;  // end of the bytes

           /******   set up the new thread processing data here*****/
//after the data processed ，don’t forget to delete[] pTmp;

        AfxBeginThread ( ProcessData, pTmp );

        iCounter = 0;      //// clear counter

} ///end while

return 0;
}
```