WiMi-net 无线自组网管理平台

Win32　SDK 说明书

（高级版）

微网高通（北京)无线技术有限公司

目　录

1. 多基站并发：初始化

1.1 指定文件路径：公共绝对地址

函数名	**char WiMinet_SetTask_WorkPath** （unsigned char iShell, char * pTxPath）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	char * pTxPath	可寻址的文件路径，比如“C:\Windows\System32”,最大长度 1023 字节
返回值	0X00=操作失败 0X01=操作成功	

1.2 设置回调函数：详细任务参数

函数名	**void WiMinet_SetTask_CallBack** （unsigned char iShell, WiMinet_TxFile_Notice pNotice）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄，0XFF 代表所有的连接句柄，0X01~0XFE 为单个连接句柄
参数二	WiMinet_TxFile_Notice pNotice	回调函数的入口，函数的类型为： void Sample_CallBack （unsigned char iShell， unsigned char iStatus） 其中参数定义如下： iShell 为连接的句柄号码 iStatus：通知的状态消息名称 0X80：基站发送唤醒报文，唤醒目标节点 0XA0：文件已经发送完毕 0XC0：已经完成了基站文件发送通知的接收 0XE0：已经完成了节点上报通知的接受 0XFF：整个文件发送任务已经结束，可以接收新的发送任务
返回值	无	

1.3 设置唤醒基站：现场部署配置

函数名	**char WiMinet_SetServer_Wakeup** （unsigned char iShell, unsigned char iSector, unsigned char iTIndex）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	unsigned char iSector	基站的区域编号：不同区域的基站意味着物理空间上不重叠，多个基站可以同步并行发送 有效范围：0X00~0XFE，代表此基站参与电波唤醒 无效数值：0XFF，代表此基站不参与电波唤醒 默认数值：0XFF 数值大小：没有意义，仅代表区域的编号
参数三	unsigned char iTIndex	基站的区域等级：同一个区域内部的基站等级 有效范围：0X00~0XFE，代表此基站参与电波唤醒 无效数值：0XFF，代表此基站不参与电波唤醒 默认数值：0XFF 大小意义：0X00 是最高等级，第一轮发射，1-2-3-4 一次递减
返回值	0X00=操作失败； 0X01=操作成功	

1.4 设置基站并发：最大并发数量

函数名	**char WiMinet_SetTask_Paralell** （unsigned char iSize）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iSize	同时进入并发模式的基站的最多数量， 注意： (1) 实际并发的数量不会超过基站的数量 (2) 基站由于基站安装位置的关系，实际并发数可能少设定数值 (3) 设置数值 0 或 1 则禁止并发，每次使用一个基站发送数据
返回值	设定的并发数值	

1.5 清除广播地址：清除广播队列

函数名	**void WiMinet_Broadcast_Delete** （unsigned char * pTable, unsigned short iSize）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char * pTable	需要清除的节点地址列表，Microsoft Windows 平台的字节顺序，低字节在前，高字节在后 注意： (1) pTable=0X00，iSize=0XFFFF 清除所有的节点 (2) 其余情况，清除列表中指定的节点
参数二	unsigned short iSize	字节数量，是实际节点数量的 2 倍
返回值	无	

1.6 增加广播地址：添加广播节点

函数名	**void WiMinet_Broadcast_Insert** （unsigned char * pTable, unsigned short iSize）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char * pTable	需要添加的节点地址列表，Microsoft Windows 平台的字节顺序，低字节在前，高字节在后 注意： （1）pTable=0X00，iSize=0XFFFF 添加所有的节点 （2）其余情况，添加列表中指定的节点
参数二	unsigned short iSize	字节数量，是实际节点数量的 2 倍
返回值	无	

2. 多基站并发：文件发送

2.1 检查任务队列：是否可写任务

函数名	**char WiMinet_GetTask_Writable** （unsigned char iShell, unsigned char * pSize）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数一	unsigned char * pSize	当前连接的最大可写入任务深度
返回值	当前连接可以写入的任务个数	

2.2 发送图片文件：三种文件格式

函数名	**char WiMinet_BMPTask_SendFile** （unsigned char iShell, unsigned short iObject, char * pTxFile） **char WiMinet_EPDTask_SendFile** （unsigned char iShell, unsigned short iObject, char * pTxFile） **char WiMinet_ZipTask_SendFile** （unsigned char iShell, unsigned short iObject, char * pTxFile）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	物理接口：1~0XFE 意义：通过 OpenWiMinetShell 返回的接口句柄 虚拟接口：0X00 意义：代表最近通过 OpenWiMinetShell 打开的接口句柄 虚拟接口：0XFF 意义：代表由系统自动选择最佳信号强度的物理接口
参数二	unsigned short iObject	目标节点的 16 位网络地址，如果是虚拟地址 0XFFFF 需要先设置广播地址列表
参数三	char * pTxFile	本次磁盘文件的名称，最大长度是 251 字符；可以是绝对路径，比如“c:\test.bin”；也可以是相对路径，比如“test.bin”；如果没有磁盘符

		号，则被判定为相对路径，会在将公共磁盘路径和相对路径组合成绝对文件路径 **该函数有三个版本，分别对应不同的文件类型：** **(1) WiMinet_BMPTask_SendFile 原始的 BMP 文件** **(2) WiMinet_EPDTask_SendFile 文件是 BMP 文件被转换成 EPD 格式，尾缀 bwrg** **(3) WiMinet_ZipTask_SendFile 文件是被压缩过的 BMP 或者 EPD 文件**
返回值	0X00=操作失败； 0X01=操作成功	
备注	该函数以异步方式返回，内部有队列可以记录用户提交的任务，最大任务数量=32	

2.3 保存图片文件：三种文件格式

函数名	**char WiMinet_BMPTask_SaveFile** （unsigned char iShell， unsigned short iObject， char * pTxFile， char index） **char WiMinet_EPDTask_SaveFile** （unsigned char iShell， unsigned short iObject， char * pTxFile， char index） **char WiMinet_ZipTask_SaveFile** （unsigned char iShell， unsigned short iObject， char * pTxFile， char index）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	**说明**
参数一	unsigned char iShell	物理接口：1~0XFE 意义：通过 OpenWiMinetShell 返回的接口句柄 虚拟接口：0X00 意义：代表最近通过 OpenWiMinetShell 打开的接口句柄 虚拟接口：0XFF 意义：代表由系统自动选择最佳信号强度的物理接口
参数二	unsigned short iObject	目标节点的 16 位网络地址，如果是虚拟地址 0XFFFF 需要先设置广播地址列表
参数三	char * pTxFile	本次磁盘文件的名称，最大长度是 251 字符；可以是绝对路径，比如“c:\test.bin”；也可以是相对路径，比如“test.bin”；如果没有磁盘符号，则被判定为相对路径，会在将公共磁盘路径和相对路径组合成绝对文件路径 **该函数有三个版本，分别对应不同的文件类型：**

		(4) WiMinet_BMPTask_SendFile 原始的 BMP 文件 (5) WiMinet_EPDTask_SendFile 文件是 BMP 文件被转换成 EPD 格式，尾缀 bwrg (6) WiMinet_ZipTask_SendFile 文件是被压缩过的 BMP 或者 EPD 文件
参数四	char index	需要保存的文件编号，默认最大 4 个文件，编号范围是 0~3
返回值	0X00=操作失败； 0X01=操作成功	
备注	该函数以异步方式返回，内部有队列可以记录用户提交的任务，最大任务数量=32	

2.4 查看图片文件：指定文件编号

函数名	char WiMinet_CmdTask_ViewFile (unsigned char iShell, unsigned short iObject, char index)	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	物理接口：1~0XFE 意义：通过 OpenWiMinetShell 返回的接口句柄 虚拟接口：0X00 意义：代表最近通过 OpenWiMinetShell 打开的接口句柄 虚拟接口：0XFF 意义：代表由系统自动选择最佳信号强度的物理接口
参数二	unsigned short iObject	目标节点的 16 位网络地址，如果是虚拟地址 0XFFFF 需要先设置广播地址列表
参数三	char index	通过函数 WiMinet_BMPTask_SaveFile ， WiMinet_EPDTask_SaveFile ， WiMinet_BMPTask_SaveFile 保存的文件的编号
返回值	0X00=操作失败； 0X01=操作成功	
备注	该函数以异步方式返回，内部有队列可以记录用户提交的任务，最大任务数量=32	

2.5 更新设备固件：两种文件格式

函数名	**char WiMinet_RawTask_Firmware** （unsigned char iShell, unsigned short iObject, char * pTxFile） **char WiMinet_ZipTask_Firmware** （unsigned char iShell, unsigned short iObject, char * pTxFile）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	物理接口：1~0XFE 意义：通过 OpenWiMinetShell 返回的接口句柄 虚拟接口：0X00 意义：代表最近通过 OpenWiMinetShell 打开的接口句柄 虚拟接口：0XFF 意义：代表由系统自动选择最佳信号强度的物理接口
参数二	unsigned short iObject	目标节点的 16 位网络地址，如果是虚拟地址 0XFFFF 需要先设置广播地址列表
参数三	char * pTxFile	本次磁盘文件的名称，最大长度是 251 字符；可以是绝对路径，比如“c:\test.bin”；也可以是相对路径，比如“test.bin”；如果没有磁盘符号，则被判定为相对路径，会在将公共磁盘路径和相对路径组合成绝对文件路径 **该函数有两个版本，分别对应不同的文件类型：** **(1) WiMinet_RawTask_Firmware 原始的 bin 格式固件文件** **(2) WiMinet_ZipTask_Firmware 文件是被压缩过的 bin 格式固件文件**
返回值	0X00=操作失败 ； 0X01=操作成功	
备注	该函数以异步方式返回，内部有队列可以记录用户提交的任务，最大任务数量=32	

3. 多基站并发：状态查询

查询任务状态：简要任务状态

函数名	char WiMinet_GetTask_Complete (unsigned char iShell, unsigned char * pStatus, unsigned long * pSerial)	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	unsigned char * pStatus	文件发送任务的状态信息，如果该指针填写 0X00，则代表不需要读取，各比特位定义如下： BIT0：电波唤醒通知报文的应答状态，0 为正常，1 为异常 BIT1：文件发送通知报文的应答状态，0 为正常，1 为异常 BIT2：文件发送成功与否的查询结果，0 为正常，1 为异常 BIT3：远程节点上报数值的应答状态，0 为正常，1 为异常
参数三	unsigned long * pSerial	文件发送业务的流水序号，默认数值为 0X00，发送第一个文件的业务序号是 0X01，此后依次递增；如果该指针填写 0X00，则代表不需要读取
返回值	0X00：文件发送没有结束，需要继续查询等待 0X01：文件发送已经结束，可以读取发送的状态信息，或者发送下一个文件	

4. 多基站并发：任务报告

4.1 查询发送状态：文件发完与否

函数名	**char WiMinet_GetStatus_TxFile** (unsigned char iShell, WiMinet_IOT_Record * pTxFile)	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	WiMinet_IOT_Record * pTxFile	文件发送状态的信息，包含起始时间以及详细状态参数
返回值	0X00：状态信息是无效的； 0X01：状态信息是有效的	

4.2 查询节点上报：是否收到报告

函数名	**char WiMinet_GetReport_Client** (unsigned char iShell, WiMinet_IOT_Record * pReport)	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	WiMinet_IOT_Record * pReport	节点上报状态的信息，包含起始时间以及详细状态参数
返回值	0X00：状态信息是无效的； 0X01：状态信息是有效的	

4.3 查询基站通知：文件发送结果

函数名	**char WiMinet_GetNotice_AirTCP** （unsigned char iShell, WiMinet_IOT_Record * pAirTCP）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	WiMinet_IOT_Record * pAirTCP	基站 TCP 文件发送状态，包含起始时间以及详细状态参数
返回值	0X00：状态信息是无效的； 0X01：状态信息是有效的	

4.4 查询基站通知：唤醒目标节点

函数名	**char WiMinet_GetNotice_Wakeup** （unsigned char iShell, WiMinet_IOT_Record * pWakeup）	
头文件	API-WiMinet.h	
静态库	WiMinet.lib	
动态库	WiMinet.dll	
	形式	说明
参数一	unsigned char iShell	通过 OpenWiMinetShell 返回的接口句柄
参数二	WiMinet_IOT_Record * pWakeup	基站电波唤醒发送状态，包含起始时间以及详细状态参数
返回值	0X00：状态信息是无效的； 0X01：状态信息是有效的	

5. 多基站并发：测试例程

5.1 测试例程：多基站并发

```
// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
#include <conio.h>

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
#include <stdio.h>

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
#include "API-WiMinet.h"

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
#define MAX_SHELL_SIZE                                          0X02

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned long dwStatic_Serial[MAX_SHELL_SIZE] = { 0X00, 0X00 };

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char  iStatic_XShell[MAX_SHELL_SIZE] = { 0X00, 0X00 };

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned short pStatic_Object[MAX_SHELL_SIZE] = { 0XC3DF, 0X8E45 };

/*
// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
char * pOutputFileName[] = {
   "G:\\BMP\\1.bwrg.mLZO",
   "G:\\BMP\\2.bwrg.mLZO",
   "G:\\BMP\\3.bwrg.mLZO",
   "G:\\BMP\\4.bwrg.mLZO",
   "G:\\BMP\\5.bwrg.mLZO",
   "G:\\BMP\\6.bwrg.mLZO",
   "G:\\BMP\\7.bwrg.mLZO",
   "G:\\BMP\\8.bwrg.mLZO",
   "G:\\BMP\\9.bwrg.mLZO",
   "G:\\BMP\\10.bwrg.mLZO",
   "G:\\BMP\\11.bwrg.mLZO",
   "G:\\BMP\\12.bwrg.mLZO",
```

```
    "G:\\BMP\\13.bwrg.mLZO",
    "G:\\BMP\\14.bwrg.mLZO",
    "G:\\BMP\\16.bwrg.mLZO",
    "G:\\BMP\\19.bwrg.mLZO" };
*/

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
char * pOutputFileName[] = {
    "1.bwrg.mLZO",
    "2.bwrg.mLZO",
    "3.bwrg.mLZO",
    "4.bwrg.mLZO",
    "5.bwrg.mLZO",
    "6.bwrg.mLZO",
    "7.bwrg.mLZO",
    "8.bwrg.mLZO",
    "9.bwrg.mLZO",
    "10.bwrg.mLZO",
    "11.bwrg.mLZO",
    "12.bwrg.mLZO",
    "13.bwrg.mLZO",
    "14.bwrg.mLZO",
    "16.bwrg.mLZO",
    "19.bwrg.mLZO" };


// ************************************************************************
// Design Notes:
// ------------------------------------------------------------------------
unsigned long Translate_Output_File_Number( unsigned long dwChar )
{
    unsigned long dwSize;

    // Select the file ID
    if ( ( dwChar >= '0' ) && ( dwChar <= '9' ) )
    {
        // Skip off the base value
        dwChar -= '0';
    }
    else if ( ( dwChar >= 'a' ) && ( dwChar <= 'f' ) )
    {
        // Skip off the base value
        dwChar -= 'a';

        // Add the basic value
        dwChar += 0X0A;
    }
    else if ( ( dwChar >= 'A' ) && ( dwChar <= 'F' ) )
    {
        // Skip off the base value
        dwChar -= 'A';

        // Add the basic value
        dwChar += 0X0A;
    }
    else
```

```
    {
        dwChar = 0XFFFFFFFF;
    }

    // The max file counter
    dwSize = sizeof( pOutputFileName ) / sizeof( pOutputFileName[0X00] );

    // Validate the input character
    if ( dwChar >= dwSize )
    {
        dwChar = 0XFFFFFFFF;
    }

    // The new output file number
    return dwChar;
}

// ***************************************************************************
// Design Notes:
// ---------------------------------------------------------------------------
char * Translate_Output_File_Name( unsigned long dwIndex )
{
    return pOutputFileName[dwIndex];
}

// ***************************************************************************
// Design Notes:
// ---------------------------------------------------------------------------
WORD NEW_Console_Color( WORD dwColor )
{
    HANDLE hConsole;
    CONSOLE_SCREEN_BUFFER_INFO info;

    // Get the console handle
    hConsole = GetStdHandle( STD_OUTPUT_HANDLE );

    // Get the current console information
    GetConsoleScreenBufferInfo( hConsole, &info );

    // Intensity the color
    dwColor |= ( FOREGROUND_INTENSITY | BACKGROUND_INTENSITY );

    // Active items
    SetConsoleTextAttribute( hConsole, dwColor );

    // The default console color
    return info.wAttributes;
}

// ***************************************************************************
// Design Notes:
// ---------------------------------------------------------------------------
void END_Console_Color( void )
{
    HANDLE hConsole;
    WORD dwColor;

    // Get the console handle
```

```
    hConsole = GetStdHandle( STD_OUTPUT_HANDLE );

    // The white color
    dwColor = ( FOREGROUND_BLUE | FOREGROUND_GREEN | FOREGROUND_RED );

    // Active items
    SetConsoleTextAttribute( hConsole, dwColor );
}

// ************************************************************************
// Design Notes:
// ------------------------------------------------------------------------
void UserInputKey_Handler( unsigned long dwChar )
{
    char * pFile;
    unsigned char index;
    unsigned long dwFile;

    // Translate the input character
    dwFile = Translate_Output_File_Number( dwChar );

    // Validate the file index
    if ( dwFile == 0XFFFFFFFFUL )
    {
        printf( "Invalid File Number!\r\n" );
        return;
    }

    // Translate the file names
    pFile = Translate_Output_File_Name( dwFile );

    // Mode-1: X2.Server + X2.Client
    for ( index = 0X00; index < MAX_SHELL_SIZE; index++ )
    {
        WiMinet_ZipTask_SendFile( iStatic_XShell[index], pStatic_Object[index], pFile );
    }

    // Mode-2: X1.Server = X2.Client
    //WiMinet_ZipTask_SendFile( iStatic_XShell[0X01], pStatic_Object[0X00], pFile );
    //WiMinet_ZipTask_SendFile( iStatic_XShell[0X01], pStatic_Object[0X01], pFile );
}

// ************************************************************************
// Design Notes:
// ------------------------------------------------------------------------
void PrintTaskInfomation_Time( WiMinet_IOT_Time64 * pTime )
{
    FILETIME nFileTime;
    SYSTEMTIME nTimerA;
    SYSTEMTIME nTimerB;
    unsigned char iStatus;

    // Check if time out
    iStatus = ( pTime->m_iStatus & WIMINET_X64TIMER_STOP );

    // Validate the timer status
    if ( !iStatus )
    {
```

```
      printf( "   Warning: Event Time Out! The Information Is Invalid!\r\n" );
      return;
   }

   // Convert to local file time
   FileTimeToLocalFileTime( ( FILETIME * )&pTime->m_qwTimeA, &nFileTime );

   // Convert file time to system time
   FileTimeToSystemTime( &nFileTime, &nTimerA );


   // Convert to local file time
   FileTimeToLocalFileTime( ( FILETIME * )&pTime->m_qwTimeB, &nFileTime );

   // Convert file time to system time
   FileTimeToSystemTime( &nFileTime, &nTimerB );

   // The date time header
   printf( "   SN=%lu,Active=0X%02X,Time=", pTime->m_dwX32SN, pTime->m_iStatus );

   // The open timer
   printf( "%04lu-%02lu-%02lu %02lu-%02lu-%02lu.%03lu",
      nTimerA.wYear,
      nTimerA.wMonth,
      nTimerA.wDay,
      nTimerA.wHour,
      nTimerA.wMinute,
      nTimerA.wSecond,
      nTimerA.wMilliseconds );

   // The date time header
   printf( " -> " );

   // The open timer
   printf( "%04lu-%02lu-%02lu %02lu-%02lu-%02lu.%03lu",
      nTimerB.wYear,
      nTimerB.wMonth,
      nTimerB.wDay,
      nTimerB.wHour,
      nTimerB.wMinute,
      nTimerB.wSecond,
      nTimerB.wMilliseconds );

   // The offset timer
   printf( ",Offset=%lu(ms)\r\n", pTime->m_dwTimeX );
}

// ****************************************************************************
// Design Notes:
// ----------------------------------------------------------------------------
void PrintTaskInfomation_Node( WiMinet_IOT_Packet * pNode )
{
   // The node information
   printf( "   <1> Node=0X%04X\r\n", pNode->m_iX16NET );

   // The packet attribute
   printf( "   <2> Attr=0X%02X\r\n", pNode->m_iAttrib );
```

```
    // The packet size
    printf( "    <3> Size=%lu Bytes\r\n", pNode->m_dwCount );
}

// ************************************************************************
// Design Notes:
// ------------------------------------------------------------------------
void Sub_Sample_CallBack_Notice_Wakeup( unsigned char iShell )
{
    WiMinet_IOT_Record nWakeup;
    WiMinet_TxReport * pTxReport;

    // The event information
    printf( "Notice.Wakup\r\n" );

    // Get the event contents
    WiMinet_GetNotice_Wakeup( iShell, &nWakeup );

    // The task time information
    PrintTaskInfomation_Time( &nWakeup.m_nIOTime );

    // The task node information
    PrintTaskInfomation_Node( &nWakeup.m_nPacket );

    // Check the code value
    if ( nWakeup.m_nPacket.m_iAttrib != WIMINET_EVENT_WOR_COMMUTE )
    {
        printf( "    Invalid Event Code\r\n" );
        return;
    }

    // The report status
    printf( "    [4] Events=WOR Commute END\r\n" );

    // The TxReport status
    pTxReport = ( WiMinet_TxReport * )nWakeup.m_nPacket.m_pBuffer;

    // The TaskID number
    printf( "    [5] TaskID=%u\r\n", pTxReport->m_iTaskID );

    // The Tx data size
    printf( "    [6] TxSize=%lu Bytes\r\n", pTxReport->m_dwCount );
}

// ************************************************************************
// Design Notes:
// ------------------------------------------------------------------------
void Sub_Sample_CallBack_Status_TxFile( unsigned char iShell )
{
    unsigned long dwTimerX;
    WiMinet_IOT_Record nTxFile;

    // The event information
    printf( "Status.TxFile\r\n" );

    // Get the event contents
    WiMinet_GetStatus_TxFile( iShell, &nTxFile );
```

```
// The task time information
PrintTaskInfomation_Time( &nTxFile.m_nIOTime );

// The task node information
PrintTaskInfomation_Node( &nTxFile.m_nPacket );

// Get the task timer
memcpy( &dwTimerX, nTxFile.m_nPacket.m_pBuffer, sizeof( dwTimerX ) );

// The task completed status
switch ( nTxFile.m_nPacket.m_iAttrib )
{
case TXD_TASK_STATUS_END_SUCCESS:
   {
      // The task status
      printf(
         "   [4] +%lu Tx Completed Successfully,Time=%lu(ms)\r\n",
         nTxFile.m_nIOTime.m_dwX32SN,
         dwTimerX );
   }
   break;

case TXD_TASK_STATUS_END_FAILURE:
   {
      // NEW the color
      NEW_Console_Color( FOREGROUND_GREEN | BACKGROUND_BLUE );

      // The task status
      printf(
         "   [4] +%lu Tx Completed With Error,Time=%lu(ms)\r\n",
         nTxFile.m_nIOTime.m_dwX32SN,
         dwTimerX );

      // END the color
      END_Console_Color();
   }
   break;

case TXD_TASK_STATUS_END_NOCRC32:
   {
      // NEW the color
      NEW_Console_Color( FOREGROUND_GREEN | BACKGROUND_BLUE );

      // The task status
      printf(
         "   [4] +%lu Tx Completed Without CRC32,Time=%lu(ms)\r\n",
         nTxFile.m_nIOTime.m_dwX32SN,
         dwTimerX );

      // END the color
      END_Console_Color();
   }
   break;

case TXD_TASK_STATUS_ERR_CONNECT:
   {
      // NEW the color
      NEW_Console_Color( FOREGROUND_GREEN | BACKGROUND_BLUE );
```

```
            // The task status
            printf(
               "   [4] +%lu Tx Device Connection Error,Time=%lu(ms)\r\n",
               nTxFile.m_nIOTime.m_dwX32SN,
               dwTimerX );

            // END the color
            END_Console_Color();
         }
         break;

      case TXD_TASK_STATUS_ERR_ABANDON:
         {
            // NEW the color
            NEW_Console_Color( FOREGROUND_GREEN | BACKGROUND_BLUE );

            // The task status
            printf(
               "   [4] +%lu Tx Device Transmit Give Up,Time=%lu(ms)\r\n",
               nTxFile.m_nIOTime.m_dwX32SN,
               dwTimerX );

            // END the color
            END_Console_Color();
         }
         break;

      default:
         {
            // NEW the color
            NEW_Console_Color( FOREGROUND_GREEN | BACKGROUND_BLUE );

            // The task status
            printf(
               "   [4] +%lu Tx Completed With Unknown Error,Time=%lu(ms)\r\n",
               nTxFile.m_nIOTime.m_dwX32SN,
               dwTimerX );

            // END the color
            END_Console_Color();
         }
         break;
      }
}

// ****************************************************************************
// Design Notes:
// ----------------------------------------------------------------------------
void Sub_Sample_CallBack_Notice_AirTCP( unsigned char iShell )
{
   unsigned char iTaskID;
   WiMinet_IOT_Record nAirTCP;
   WiMinet_TxReport * pTxReport;

   // The event information
   printf( "Notice.AirTCP\r\n" );
```

```
// Get the event contents
WiMinet_GetNotice_AirTCP( iShell, &nAirTCP );

// The task time information
PrintTaskInfomation_Time( &nAirTCP.m_nIOTime );

// The task node information
PrintTaskInfomation_Node( &nAirTCP.m_nPacket );

// The event of the object
switch ( nAirTCP.m_nPacket.m_iAttrib )
{
case WIMINET_EVENT_COMMUTE_END:
   {
      // The TxReport status
      pTxReport = ( WiMinet_TxReport * )nAirTCP.m_nPacket.m_pBuffer;

      // The report status
      printf( "   [4] Events=TCP Commute End\r\n" );

      // The TaskID number
      printf( "   [5] TaskID=%u\r\n", pTxReport->m_iTaskID );

      // The Tx data size
      printf( "   [6] RxSize=%u Bytes\r\n", pTxReport->m_iNotUse );

      // The Tx data size
      printf( "   [7] TxSize=%lu Bytes\r\n", pTxReport->m_dwCount );

      // The report status
      printf( "   [8] Errors=0X%02X", pTxReport->m_iQError );

      // The detailed error code: No Header
      if ( pTxReport->m_iQError & WIMINET_IO_REPORT_NO_HEADER )
      {
         printf( "[No Header]" );
      }

      // The detailed error code: Invalid Size
      if ( pTxReport->m_iQError & WIMINET_IO_REPORT_ER_AMOUNT )
      {
         printf( "[Invalid Size]" );
      }

      // The detailed error code: Invalid CRC32
      if ( pTxReport->m_iQError & WIMINET_IO_REPORT_ER_CRCODE )
      {
         printf( "[Invalid CRC32]" );
      }

      // The end of this line
      printf( "\r\n" );
   }
   break;

case WIMINET_EVENT_COMMUTE_ERR:
   {
      // The report status
```

```
            printf( "   [4] Events=TCP Commute Error\r\n" );

            // Get the taskid
            iTaskID = nAirTCP.m_nPacket.m_pBuffer[0X00];

            // The TaskID number
            printf( "   [5] TaskID=%u\r\n", iTaskID );
        }
        break;

    case WIMINET_EVENT_CONNECT_ERR:
        {
            // The report status
            printf( "   [4] Events=TCP Connect Error\r\n" );

            // Get the taskid
            iTaskID = nAirTCP.m_nPacket.m_pBuffer[0X00];

            // The TaskID number
            printf( "   [5] TaskID=%u\r\n", iTaskID );
        }
        break;

    default:
        {
            // The report status
            printf( "   [4] Events=Unknown Events\r\n" );
        }
        break;
    }
}

// ****************************************************************************
// Design Notes:
// ----------------------------------------------------------------------------
void Sub_Sample_CallBack_Report_Client( unsigned char iShell )
{
    unsigned long dwSize;
    WiMinet_IOT_Record nReport;
    WiMinet_RxReport * pRxReport;

    // The event information
    printf( "Report.Client\r\n" );

    // Get the event contents
    WiMinet_GetReport_Client( iShell, &nReport );

    // The task time information
    PrintTaskInfomation_Time( &nReport.m_nIOTime );

    // The task node information
    PrintTaskInfomation_Node( &nReport.m_nPacket );

    // The TxReport status
    pRxReport = ( WiMinet_RxReport * )nReport.m_nPacket.m_pBuffer;

    // Get the device information
    printf(
```

```
    "   [4] Device=0X%02X(%d)\r\n",
    pRxReport->m_iDevice,
    pRxReport->m_iDevice );

// Get the firmware version
printf( "   [5] Version=0X%08lX", pRxReport->m_dwBuild );

// The build version for firmware update
printf(
    " --> [20%u.%u.%u+R%u]\r\n",
    ( unsigned char )( pRxReport->m_dwBuild >> 0X18 ),
    ( unsigned char )( pRxReport->m_dwBuild >> 0X10 ),
    ( unsigned char )( pRxReport->m_dwBuild >> 0X08 ),
    ( unsigned char )( pRxReport->m_dwBuild >> 0X00 ) );

// The received byte size by the remote endpoint
dwSize = ( ( unsigned long )pRxReport->m_iMBSize << 0X10 );

// The LSB value
dwSize += pRxReport->m_iLBSize;

// The client received byte size
printf( "   [6] Counter=%lu Bytes\r\n", dwSize );

// The battery voltage
printf( "   [7] Battery=%.3f(V)\r\n", pRxReport->m_iBATVol * 0.001f );

// The RxRSSI voltage
printf( "   [8] RxRSSI=%d(dBm)\r\n", pRxReport->m_iRxRSSI );

// The RxRSSI voltage
printf( "   [9] TxRSSI=%d(dBm)\r\n", pRxReport->m_iTxRSSI );

// The client receive error code
printf( "   [A] Errors=0X%02X", pRxReport->m_iQError );

// The detailed error code: Rx Time Out
if ( pRxReport->m_iQError & WIMIET_IO_REPORT_RX_TIMEOUT )
{
    printf( " [Rx Time Out]" );
}

// The detailed error code: No Header
if ( pRxReport->m_iQError & WIMINET_IO_REPORT_NO_HEADER )
{
    printf( "[No Header]" );
}

// The detailed error code: Invalid Size
if ( pRxReport->m_iQError & WIMINET_IO_REPORT_ER_AMOUNT )
{
    printf( "[Invalid Size]" );
}

// The detailed error code: Invalid CRC32
if ( pRxReport->m_iQError & WIMINET_IO_REPORT_ER_CRCODE )
{
    printf( "[Invalid CRC32]" );
```

```
    }

    // The end of this line
    printf( "\r\n" );
}

// ********************************************************************************
// Design Notes:
// --------------------------------------------------------------------------------
void Sub_Sample_CallBack_Task_Complete( unsigned char iShell )
{
    unsigned char iSize;
    unsigned char iStatus;
    unsigned char iRetVal;
    unsigned long dwSerial;

    // The event information
    printf( "Task.Complete\r\n" );

    // Get the task complete status
    iRetVal = WiMinet_GetTask_Complete( iShell, &iStatus, &dwSerial );

    // The task serial number
    printf( "   [1] Serial=%lu\r\n", dwSerial );

    // The task over status
    printf( "   [2] TxOver=%d\r\n", iRetVal );

    // The error bit mask
    printf( "   [3] ERMask=0X%02X", iStatus );

    // Check the error status:Wakeup
    if ( iStatus & WIMINET_MASK_NOTICE_WAKEUP )
    {
        printf( " + Wakeup" );
    }

    // Check the error status:AirTCP
    if ( iStatus & WIMINET_MASK_NOTICE_AIRTCP )
    {
        printf( " + AirTCP" );
    }

    // Check the error status:TxFile
    if ( iStatus & WIMINET_MASK_STATUS_TXFILE )
    {
        printf( " + TxFile" );
    }

    // Check the error status:Report
    if ( iStatus & WIMINET_MASK_REPORT_CLIENT )
    {
        printf( " + Report" );
    }

    // Check the error status:Submit
    if ( iStatus & WIMINET_MASK_SUBMIT_PACKET )
    {
```

```
      printf( " + Submit" );
   }

   // Get the writable status
   iStatus = WiMinet_GetTask_Writable( iShell, &iSize );

   // The new writable status
   printf( "\r\n   [4] Submit=%d/%d\r\n", iStatus, iSize );

   // The end of line status
   printf( "\r\n\r\n" );
   printf( "========================================" );
   printf( "========================================" );
   printf( "\r\n" );
}

// ****************************************************************************
// Design Notes:
// ----------------------------------------------------------------------------
void Sample_CallBack( unsigned char iShell, unsigned char iStatus )
{
   printf( "\r\nShell[%d]:Status=0X%02X,Event=", iShell, iStatus );

   // Process all the events
   switch ( iStatus )
   {
   case WIMINET_NOTICE_WAKEUP:
      {
         Sub_Sample_CallBack_Notice_Wakeup( iShell );
      }
      break;

   case WIMINET_STATUS_TXFILE:
      {
         Sub_Sample_CallBack_Status_TxFile( iShell );
      }
      break;

   case WIMINET_NOTICE_AIRTCP:
      {
         Sub_Sample_CallBack_Notice_AirTCP( iShell );
      }
      break;

   case WIMINET_REPORT_CLIENT:
      {
         Sub_Sample_CallBack_Report_Client( iShell );
      }
      break;

   case WIMINET_TASK_COMPLETE:
      {
         Sub_Sample_CallBack_Task_Complete( iShell );
      }
      break;

   default:
      {
```

```
      }
      break;
   }
}

// ********************************************************************************
// Design Notes:
// --------------------------------------------------------------------------------
void Active_Poll_Task_Status_X1( unsigned char iShell )
{
   unsigned char index;
   unsigned char iStatus;
   unsigned char iTxOver;
   unsigned long dwSerial;

   // Get the complete status
   iTxOver = WiMinet_GetTask_Complete( iShell, &iStatus, &dwSerial );

   // Check if task completed
   if ( !iTxOver )
   {
      return;
   }

   // Get the shell index
   index = ( iShell - 0X01 );

   // Compare the task serial
   if ( dwSerial == dwStatic_Serial[index] )
   {
      return;
   }

   // Update current task serial
   dwStatic_Serial[index] = dwSerial;

   // Print the notice.wakeup
   Sample_CallBack( iShell, WIMINET_NOTICE_WAKEUP );

   // Print the status.txfile
   Sample_CallBack( iShell, WIMINET_STATUS_TXFILE );

   // Print the notice.airtcp
   Sample_CallBack( iShell, WIMINET_NOTICE_AIRTCP );

   // Print the report.client
   Sample_CallBack( iShell, WIMINET_REPORT_CLIENT );

   // Print the task.complete
   Sample_CallBack( iShell, WIMINET_TASK_COMPLETE );
}

// ********************************************************************************
// Design Notes:
// --------------------------------------------------------------------------------
void Active_Poll_Task_Status( void )
{
   unsigned char index;
```

```
    unsigned char iShell;

    // Poll every task status
    for ( index = 0X00; index < MAX_SHELL_SIZE; index++ )
    {
        // Get the shell number
        iShell = iStatic_XShell[index];

        // Get the shell status
        Active_Poll_Task_Status_X1( iShell );
    }
}

// ****************************************************************************
// Design Notes:
// ----------------------------------------------------------------------------
int main( int argc, char* argv[] )
{
    unsigned long dwChar;

    // Open the shell
    iStatic_XShell[0X00] = OpenWiMinetShell( "192.168.0.248",12580, 0X01 );

    // Open the shell
    iStatic_XShell[0X01] = OpenWiMinetShell( "192.168.0.108",12580, 0X01 );

    // Validate the shell open interface
    if ( !iStatic_XShell[0X00] || !iStatic_XShell[0X01] )
    {
        printf( "Open shell failed!\r\n" );
        return 0X00;
    }

    // Set the task callback function for
    //WiMinet_SetTask_CallBack( 0XFF, Sample_CallBack );

    // Set the common file path
    WiMinet_SetTask_WorkPath( 0XFF, "G:\\BMP" );

    // The notice for user input
    printf( "Please select '0'-'9' for file, 'q' or 'Q' to exit!\r\n\r\n" );

    // The main thread service
    while( 0X01 )
    {
        // Release the processor control
        Sleep( 0X01 );

        // Poll the task status
        Active_Poll_Task_Status();

        // Check the keyboard input
        if ( _kbhit() )
        {
            // Get the input character
            dwChar = _getche();

            // Check if time to exit this process
```

```
            if ( ( dwChar == 'q' ) || ( dwChar == 'Q' ) )
            {
                break;
            }

            // The user input key handler
            UserInputKey_Handler( dwChar );
        }
    }

    // Close all the shell
    StopWiMinetShell( 0XFF );

    // The function exit code
    return 0X00;
}
```

5.2 测试程序说明

该例子程序首先打开两个基站的连接：

(1) 192.168.0.248

(2) 192.168.0.108

每一个基站对应一个目标节点，分别是 0XC3DF 和 0X8E45

然后等待用户的按键输入，输入的范围是

(1) 0~9：文件编号 0X00~0X09，文件列表见程序开头部分

(2) A~F：文件编号 0X0A~0X0F

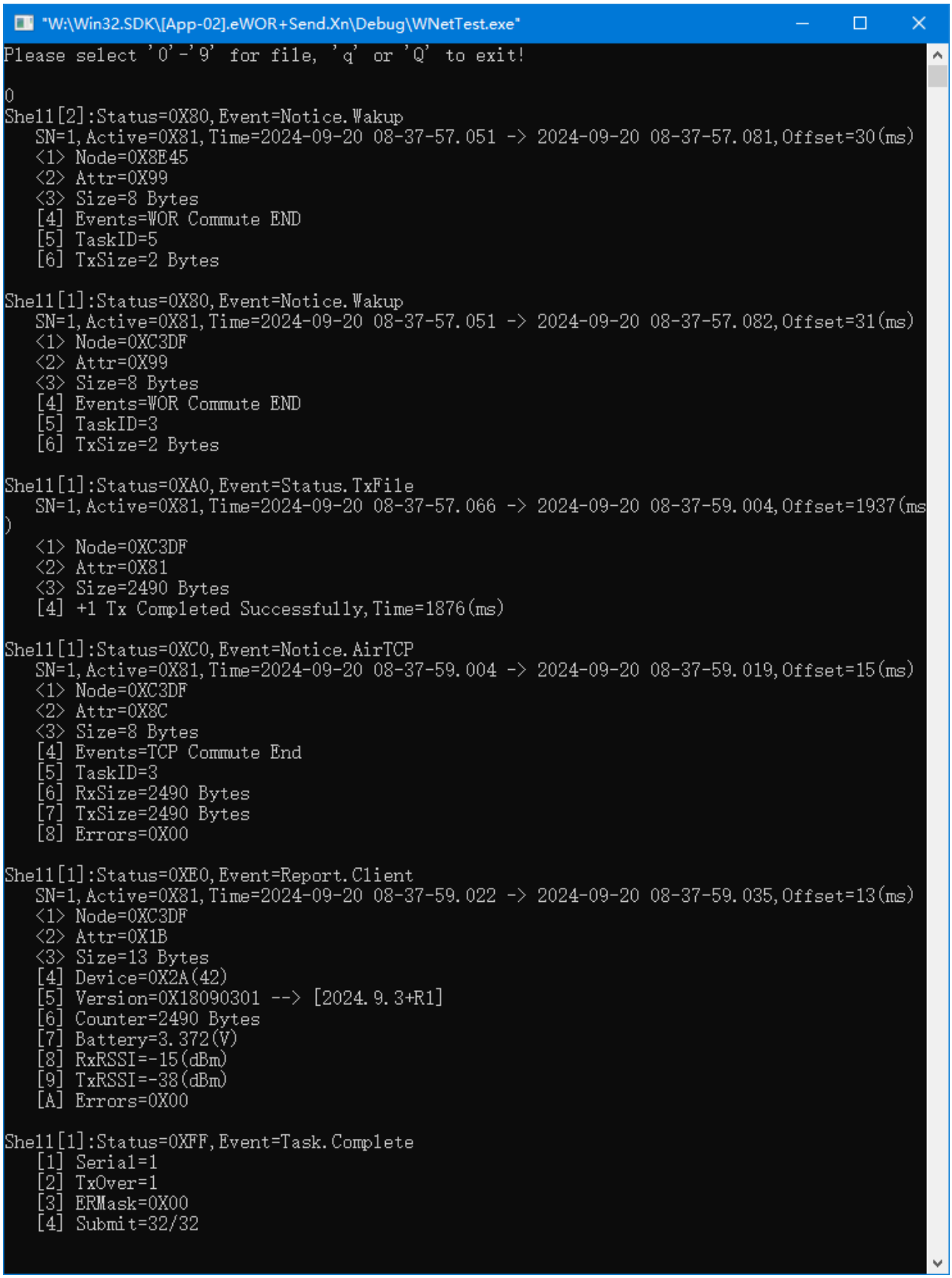
(3) a~f：同 A~F

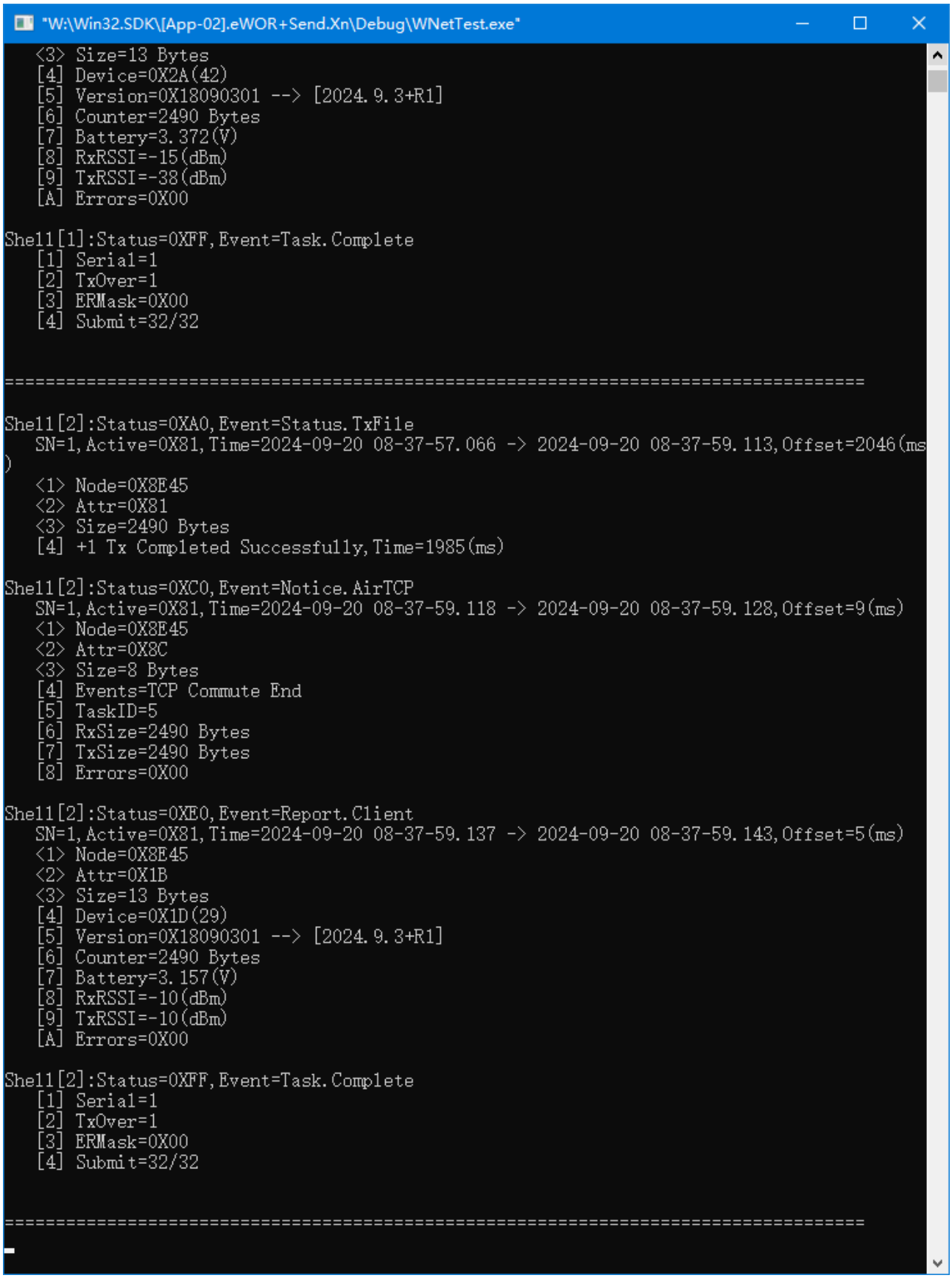
(4) q 或者 Q：退出本程序

当测试者输入有效的文件编号，就将该文件并发给对应的远程节点。

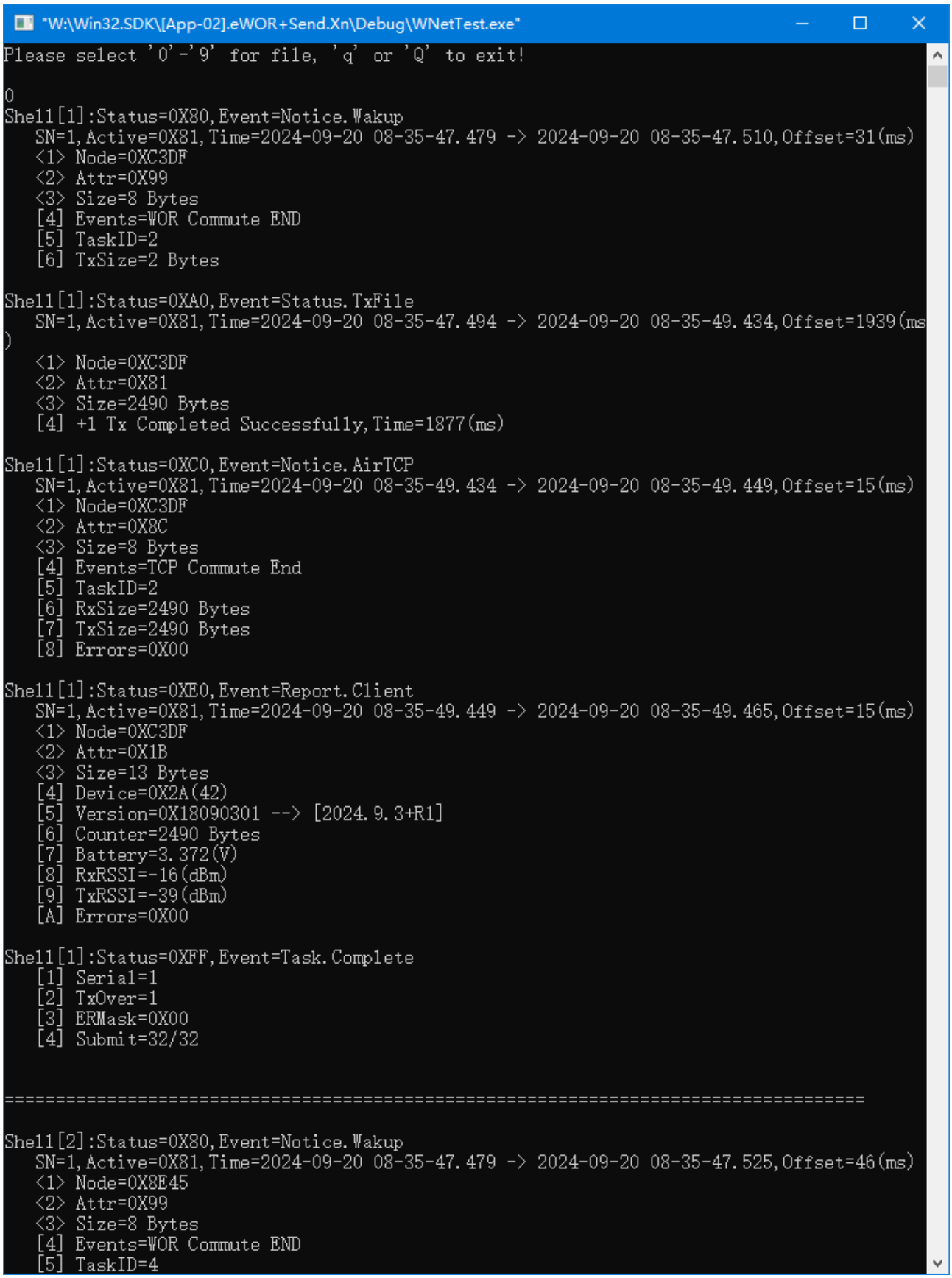
程序支持两种结果返回方式：

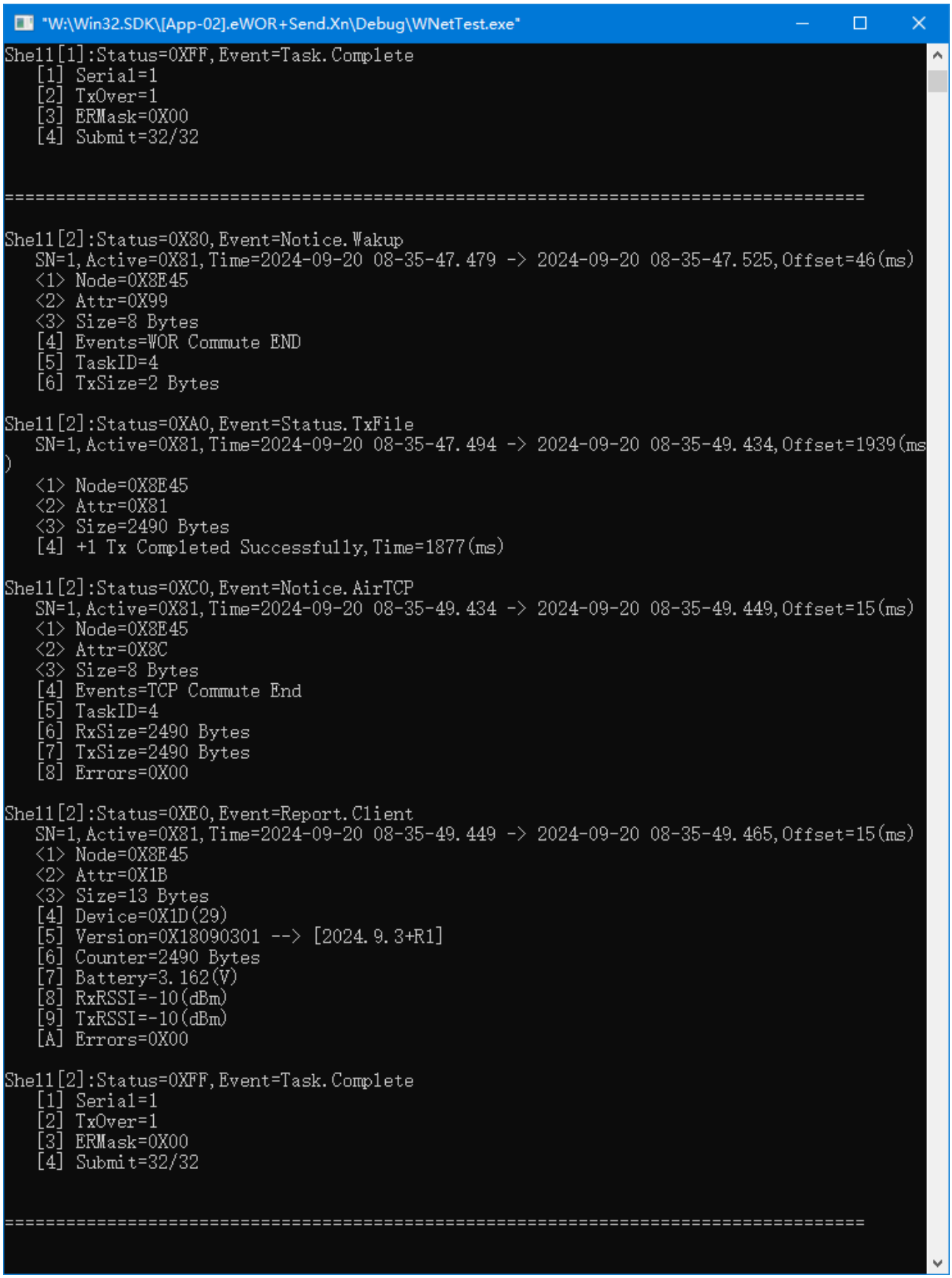
第一种是通过回调函数输出结果，此时需要打开函数 WiMinet_SetTask_CallBack，下面是运行的结果


"W:\Win32.SDK\[App-02].eWOR+Send.Xn\Debug\WNetTest.exe"
Please select '0'-'9' for file, 'q' or 'Q' to exit!
0
Shell[2]:Status=0X80,Event=Notice.Wakup
SN=1,Active=0X81,Time=2024-09-20 08-37-57.051 -> 2024-09-20 08-37-57.081,Offset=30(ms)
<1> Node=0X8E45
<2> Attr=0X99
<3> Size=8 Bytes
[4] Events=WOR Commute END
[5] TaskID=5
[6] TxSize=2 Bytes
Shell[1]:Status=0X80,Event=Notice.Wakup
SN=1,Active=0X81,Time=2024-09-20 08-37-57.051 -> 2024-09-20 08-37-57.082,Offset=31(ms)
<1> Node=0XC3DF
<2> Attr=0X99
<3> Size=8 Bytes
[4] Events=WOR Commute END
[5] TaskID=3
[6] TxSize=2 Bytes
Shell[1]:Status=0XA0,Event=Status.TxFile
SN=1,Active=0X81,Time=2024-09-20 08-37-57.066 -> 2024-09-20 08-37-59.004,Offset=1937(ms)
<1> Node=0XC3DF
<2> Attr=0X81
<3> Size=2490 Bytes
[4] +1 Tx Completed Successfully,Time=1876(ms)
Shell[1]:Status=0XC0,Event=Notice.AirTCP
SN=1,Active=0X81,Time=2024-09-20 08-37-59.004 -> 2024-09-20 08-37-59.019,Offset=15(ms)
<1> Node=0XC3DF
<2> Attr=0X8C
<3> Size=8 Bytes
[4] Events=TCP Commute End
[5] TaskID=3
[6] RxSize=2490 Bytes
[7] TxSize=2490 Bytes
[8] Errors=0X00
Shell[1]:Status=0XE0,Event=Report.Client
SN=1,Active=0X81,Time=2024-09-20 08-37-59.022 -> 2024-09-20 08-37-59.035,Offset=13(ms)
<1> Node=0XC3DF
<2> Attr=0X1B
<3> Size=13 Bytes
[4] Device=0X2A(42)
[5] Version=0X18090301 --> [2024.9.3+R1]
[6] Counter=2490 Bytes
[7] Battery=3.372(V)
[8] RxRSSI=-15(dBm)
[9] TxRSSI=-38(dBm)
[A] Errors=0X00
Shell[1]:Status=0XFF,Event=Task.Complete
[1] Serial=1
[2] TxOver=1
[3] ERMask=0X00
[4] Submit=32/32



第二种是主动查询的方式，此时需要函数 Active_Poll_Task_Status；此时运行结果如下：


"W:\Win32.SDK\[App-02].eWOR+Send.Xn\Debug\WNetTest.exe"
Please select '0'-'9' for file, 'q' or 'Q' to exit!
0
Shell[1]:Status=0X80,Event=Notice.Wakup
SN=1,Active=0X81,Time=2024-09-20 08-35-47.479 -> 2024-09-20 08-35-47.510,Offset=31(ms)
<1> Node=0XC3DF
<2> Attr=0X99
<3> Size=8 Bytes
[4] Events=WOR Commute END
[5] TaskID=2
[6] TxSize=2 Bytes
Shell[1]:Status=0XA0,Event=Status.TxFile
SN=1,Active=0X81,Time=2024-09-20 08-35-47.494 -> 2024-09-20 08-35-49.434,Offset=1939(ms
)
<1> Node=0XC3DF
<2> Attr=0X81
<3> Size=2490 Bytes
[4] +1 Tx Completed Successfully,Time=1877(ms)
Shell[1]:Status=0XC0,Event=Notice.AirTCP
SN=1,Active=0X81,Time=2024-09-20 08-35-49.434 -> 2024-09-20 08-35-49.449,Offset=15(ms)
<1> Node=0XC3DF
<2> Attr=0X8C
<3> Size=8 Bytes
[4] Events=TCP Commute End
[5] TaskID=2
[6] RxSize=2490 Bytes
[7] TxSize=2490 Bytes
[8] Errors=0X00
Shell[1]:Status=0XE0,Event=Report.Client
SN=1,Active=0X81,Time=2024-09-20 08-35-49.449 -> 2024-09-20 08-35-49.465,Offset=15(ms)
<1> Node=0XC3DF
<2> Attr=0X1B
<3> Size=13 Bytes
[4] Device=0X2A(42)
[5] Version=0X18090301 --> [2024.9.3+R1]
[6] Counter=2490 Bytes
[7] Battery=3.372(V)
[8] RxRSSI=-16(dBm)
[9] TxRSSI=-39(dBm)
[A] Errors=0X00
Shell[1]:Status=0XFF,Event=Task.Complete
[1] Serial=1
[2] TxOver=1
[3] ERMask=0X00
[4] Submit=32/32
==========
Shell[2]:Status=0X80,Event=Notice.Wakup
SN=1,Active=0X81,Time=2024-09-20 08-35-47.479 -> 2024-09-20 08-35-47.525,Offset=46(ms)
<1> Node=0X8E45
<2> Attr=0X99
<3> Size=8 Bytes
[4] Events=WOR Commute END
[5] TaskID=4


"W:\Win32.SDK\[App-02].eWOR+Send.Xn\Debug\WNetTest.exe"
Shell[1]:Status=0XFF,Event=Task.Complete
[1] Serial=1
[2] TxOver=1
[3] ERMask=0X00
[4] Submit=32/32
==========
Shell[2]:Status=0X80,Event=Notice.Wakup
SN=1,Active=0X81,Time=2024-09-20 08-35-47.479 -> 2024-09-20 08-35-47.525,Offset=46(ms)
<1> Node=0X8E45
<2> Attr=0X99
<3> Size=8 Bytes
[4] Events=WOR Commute END
[5] TaskID=4
[6] TxSize=2 Bytes
Shell[2]:Status=0XA0,Event=Status.TxFile
SN=1,Active=0X81,Time=2024-09-20 08-35-47.494 -> 2024-09-20 08-35-49.434,Offset=1939(ms
)
<1> Node=0X8E45
<2> Attr=0X81
<3> Size=2490 Bytes
[4] +1 Tx Completed Successfully,Time=1877(ms)
Shell[2]:Status=0XC0,Event=Notice.AirTCP
SN=1,Active=0X81,Time=2024-09-20 08-35-49.434 -> 2024-09-20 08-35-49.449,Offset=15(ms)
<1> Node=0X8E45
<2> Attr=0X8C
<3> Size=8 Bytes
[4] Events=TCP Commute End
[5] TaskID=4
[6] RxSize=2490 Bytes
[7] TxSize=2490 Bytes
[8] Errors=0X00
Shell[2]:Status=0XE0,Event=Report.Client
SN=1,Active=0X81,Time=2024-09-20 08-35-49.449 -> 2024-09-20 08-35-49.465,Offset=15(ms)
<1> Node=0X8E45
<2> Attr=0X1B
<3> Size=13 Bytes
[4] Device=0X1D(29)
[5] Version=0X18090301 --> [2024.9.3+R1]
[6] Counter=2490 Bytes
[7] Battery=3.162(V)
[8] RxRSSI=-10(dBm)
[9] TxRSSI=-10(dBm)
[A] Errors=0X00
Shell[2]:Status=0XFF,Event=Task.Complete
[1] Serial=1
[2] TxOver=1
[3] ERMask=0X00
[4] Submit=32/32
==========

6. 技术支持

邮箱：dingyg99@126.com
电话：13911821802 （微信同号）
座机：010-57222007

企业微信公众号