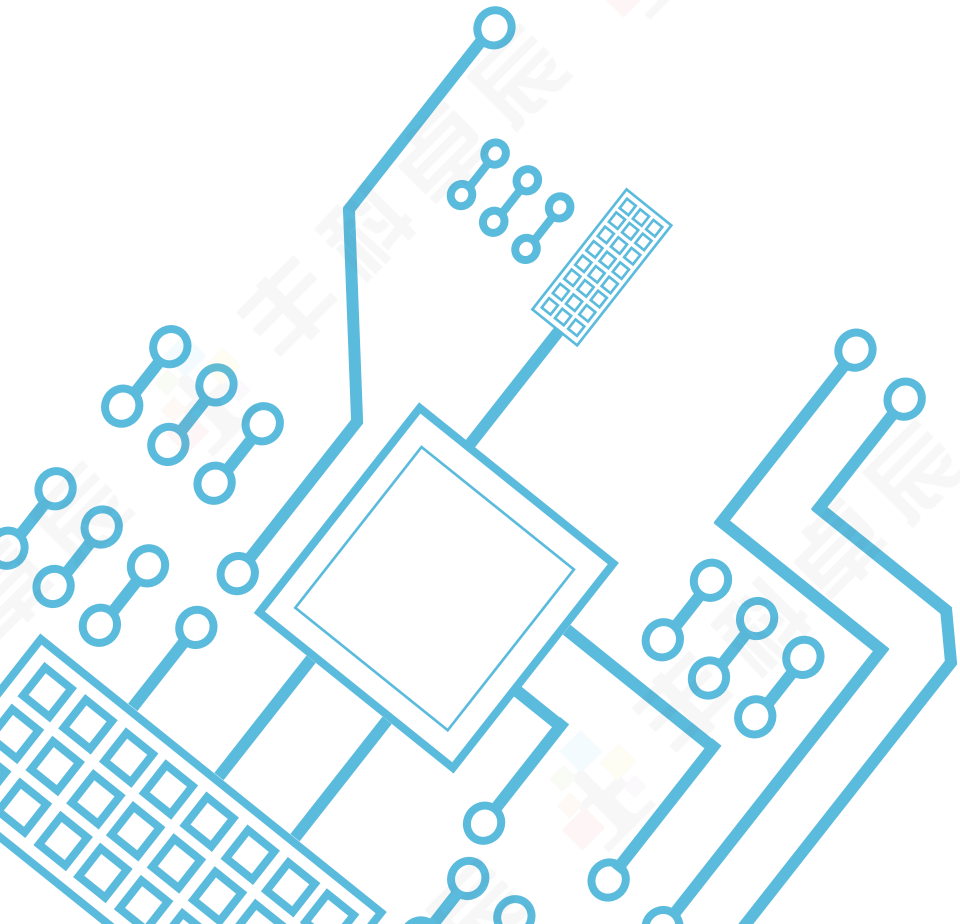
丰科卓辰

使用说明书

丰科卓辰-存储板卡

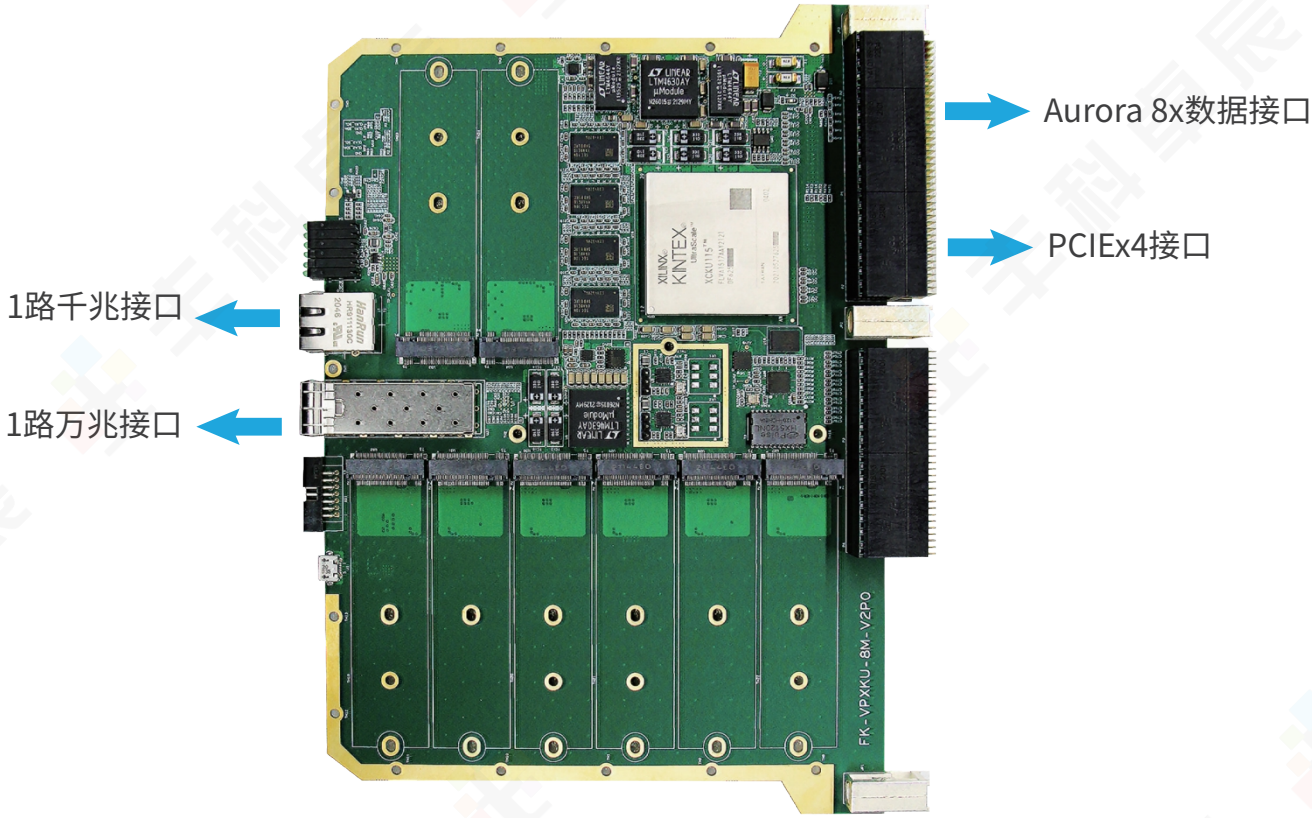


目 录

目 录

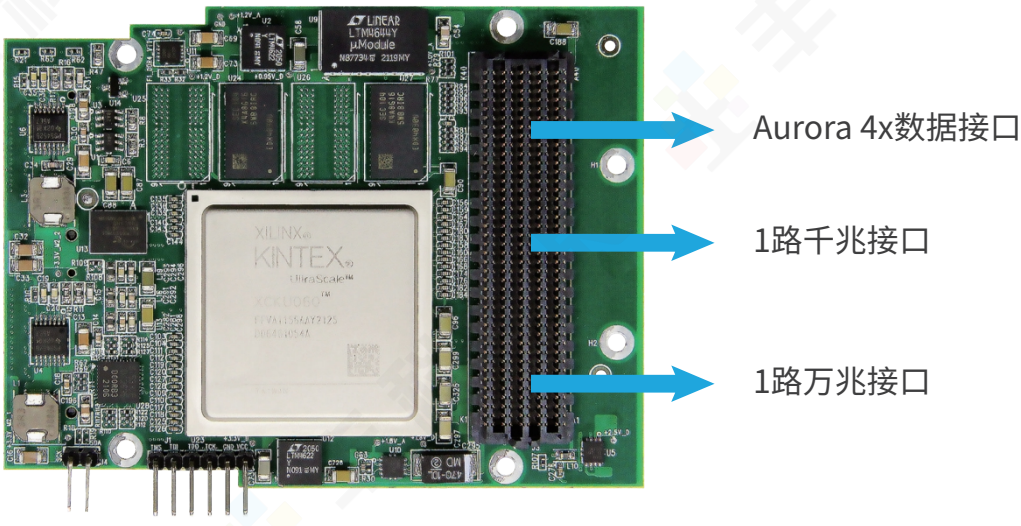
1、卡接口说明

1.1、6U VPX板卡接口

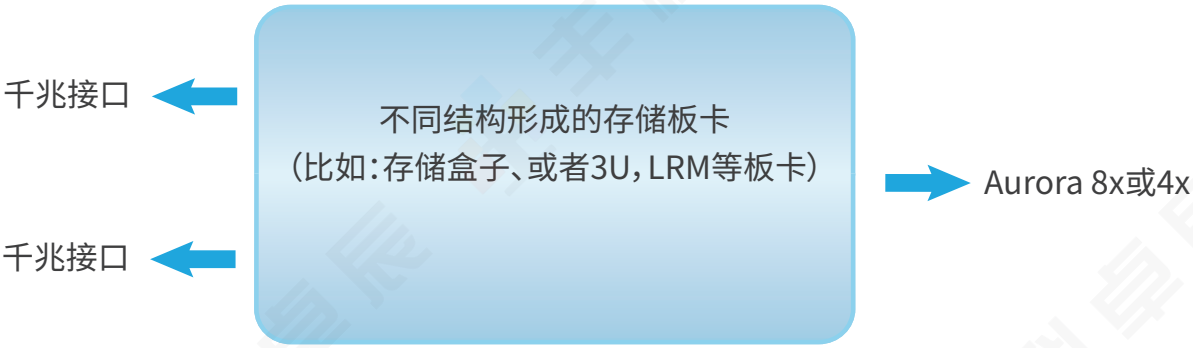


备注：1路万兆接口可选择从P2接插件提供，1路千兆接口可选择从P4接插件提供；

1.2、FMC存储子卡接口



1.3、其他结构形式的板卡接口



1.4接口说明

(一)控制及数据下载接口：

◇ 6U VPX板卡：1路千兆，1路万兆，1路PCIE可选择；如果只使用网络，则千兆，万兆可同时存在，且功能一样，具备控制及数据下载功能。如果使用了PCIE，则千兆接口网口保留，万兆 网口不提供功能。千兆网口与PCIE接口功能相同。

◇ FMC存储子卡：1路千兆，1路万兆，千兆和万兆可同时存在，且功能一样，具备控制及数据下载功能。

注意：千兆和万兆同时存在，但是同一时刻只有一个连接存在，即谁最后连接则谁控制。

(二)数据接口

6U VPX板卡：Aurora 8x接口，线速率为10Gbps;

FMC存储子卡：Aurora 4x接口，线速率为10Gbps；

2、控制流程概述

2.1、数据记录流程

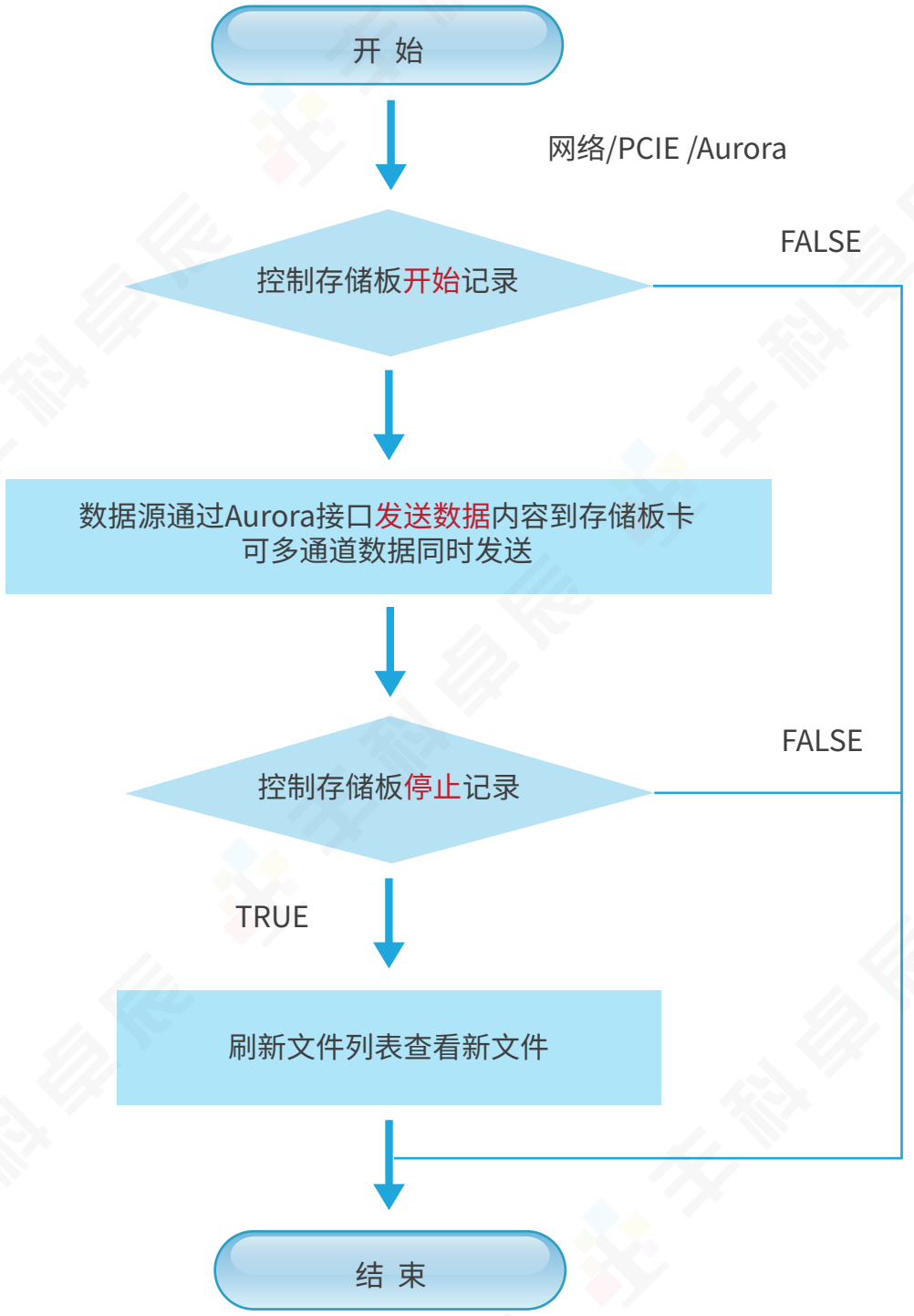
存储板卡的基本使用流程：

1）首先控制存储板卡创建文件，启动记录；可以启动多路数据文件；

2）通过Aurora接口向存储板卡发送数据；根据启动的几路数据文件记录，发送几路数据内容，每个数据文件通道形成一个文件；

3）控制存储板卡停止记录，更新文件。

流程示意如下：



2.1.1、启动存储记录

在开始工作时，首先要启动存储板卡的记录，正常可通过网络（千兆或万兆都可以）、PCIE接口、Aurora接口发送启动指令。

◇ 千兆或万兆接口发送自定义的指令（包含长度，文件名称）启动记录，启动的时候会通知启动哪个通道。

◇ PCIE接口发送启动记录指令时，是调用提供的驱动文件，指令内容与网络启动的指令内容格式一样。

◇ Aurora接口具备发送启动记录的功能，文件名称由存储板卡自动生成。提供给客户的Aurora接口文件中，会有该接口，一般应用于用户FPGA来控制开始/停止的场景。

2.1.2、发送数据内容

在启动了记录后，通过Aurora接口发送数据内容到存储即可。提供的Aurora接口功能中，可支持8个通道的数据文件同时发送。每个通道是独立的开始/停止，每个通道的带宽可不同。

备注：8个通道是逻辑层面的，都是使用Aurora 8x或4x接口。即使1个通道也可以跑接口的全带宽。

2.1.3、停止记录

不管数据源是否停止（正常情况下都是先停止数据源发送数据），都可以控制停止记录。停止记录后，在存储板卡上形成的文件才是最终状态的数据文件（数据内容及大小）。

可单独控制每个通道的记录停止。

2.2、数据下载流程

2.2.1、网络下载

支持FTP协议的数据下载，千兆或万兆都可以，但是同时只能使用一种，哪个最后连接使用哪个。

即可通过我的电脑中输入“fpt://192.168.1.5”登录到存储板卡上，然后可看到数据文件，可选择文件进行数据拷贝，拷贝到本地电脑磁盘中。

可通过FTP的工具（比如FileZilla）登录板卡，然后选择数据文件进行数据下载操作。

控制软件，可通过提供的控制软件刷新文件列表，然后下载数据文件；（原理就是自己写的FPT网络通信）。

2.2.2、PCIE下载

如果使用PCIE下载，则需客户开发下载软件，厂家提供PCIE接口的驱动文件及指令格式，及简单的PICe接口使用demo。

软件需要做的基本功能包括：检测PCIE连接是否正常，获取文件列表信息，发送下载指令，接收数据。

2.3、文件管理流程

文件管理操作基本包括：获取文件列表，删除文件，重命名文件，格式化磁盘，新建文件夹。

根据提供的控制接口（网络或PCIE）发送相关的指令，然后接收指令反馈或列表反馈即可。

3、控制协议及指令格式

3.1、网络接口

存储板卡通过前面板千兆万兆或背板千兆万兆网络实现FTP通信，并且对存储板卡的控制及数据访问。

存储板卡：FTP服务器；

外部计算机：FTP客户端。

存储板卡千兆网口默认IP一般为：192.168.1.5（可能会改变，以最后IP为准，该IP支持重新配置），万兆网IP一般为：192.168.1.6。将外部计算机设置在同一网段即可。

FTP访问的默认端口为21。

FTP访问的用户名与密码无，不填写即可。

3.2、PCIE接口

通过提供的PCIE接口驱动，发送相关控制指令（标准FTP指令及自定义指令）完成对存储板卡的控制。

3.3、标准FTP指令格式

分类	操作	指令	反馈码	反馈内容
标准FTP指令	用户名	USER	331	331 Anonymous login okay
	密码	PASS	230	230 OK. Current directory is /
	被动模式	PASV	227	227 Entering Passive Mode (192,168,1,6,128,0).
	改变目录	CWD	250	250 Changed to /
	显示目录	PWD	257	257 \"/\"
	获取列表	LIST	150	150 Accepted data connection from 192.168.1.190:19271
	列表发送完成		226	226-Options: -l\r\n226-25 matches total\r\n226 Total 33344520 KB (99 % free)
	重命名	RNFR	350	350 Rename 'xxxxx' ...
	重命名为	RNTO	250	250 Rename successful to 'xxxxx'
	删除文件	DELE	250	250 File \"xxxxx\" removed
	创建目录	MKD	257	257 \"%xxxxx\" directory xxxxx
	删除目录	RMD	257	257 \"xxxxx\" directory xxxxx
	回根目录	CDUP	250	250 Changed to xxxxx
	下载	RETR	150	150 Accepted data connection from 192.168.1.190:30487
	下载完毕		226	226 Closing connection 4295570224 bytes transmitted

3.4、自定义FTP指令格式

对于文件的操作访问、下载，均采用标准FTP协议即可。可参考第三方的FTP工具或其他说明文档。

对存储板卡的记录操作（开始、停止）采用自定义的指令格式。

该指令格式根据使用情况通过网口或PCIE接口发送与接收反馈。

自定义格式如下所示：

分类	操作	指令	反馈码	反馈内容
自定义指令	获取版本信息	VERS	900	900 version is 20211008
	获取容量信息	DF-H	888	888 Total 14904GB,Remain 14864GB, current store 0MB
	获取速率	SPD	883	883 Speed is: 310 MB/S
	格式化	FMT	880	880 Board format okay
	开始单次记录	DAQS	881	881 Board Start collecting 格式：DAQS 通道号 长度 目录+文件名（如果有目录，要确保已创建过） 例如：DAQS 0 4294967296 Data_CH0.dat
	开始持续记录	DADC	881	881 Board Start collecting 格式：DAQC 通道号 目录 例如：DAQS 0 Data_CH0
	停止记录	DAQP	882	882 Board Stop collecting 格式：DAQP 通道号 例如：DAQS 0
	回放	RDSG	898	898 localrdfile Action start 格式：RDSG 通道号 偏移量 回放大小 文件名称 例如：RDSG 1 0 1073741824 ABC.dat
	更改IP地址	IPCFG	889	889 ipconfig okay 格式：IPCFG 网口序号 IP地址 例如：DAQS 0 192.168.1.10
	读寄存器	RREG	8A0	8A0 Addr:0x00000001, Value:0x00000001 格式：RREG 地址 例如：RREG 1
	写寄存器	WREG	8A1	8A1 write ok 格式：WREG 地址 数值 例如：WREG 1 2

3.5、异常指令反馈码

分类	异常说明	反馈码	反馈内容
异常情况反馈码	指令没有正确执行	553	553 Requested action not taken.
	通道繁忙，记录或回放时使用	899	899 Channel Busy
	目录创建失败	895	895 Forder Create Failed
	文件创建失败	896	896 File Create Failed
	格式化失败	897	897 Command format error
	相应指令没有具体功能	500	500 Syntax error, command unrecognized

4、控制软件编程说明

4.1、FTP网络编程说明

该说明均为通用技术说明，所有资料可从网上查询到，而且目前具有很多的封装好的FTP的接口可使用。

文件传输协议（FTP）作为网络共享文件的传输协议，在网络应用软件中具有广泛的应用。FTP的目标是提高文件的共享性和可靠高效地传送数据。

在传输文件时，FTP 客户端程序先与服务器建立连接，然后向服务器发送命令。服务器收到命令后给予响应，并执行命令。

FTP 协议与操作系统无关，任何操作系统上的程序只要符合 FTP 协议，就可以相互传输数据。

FTP 第三方客户端如 FlashFXP，File Zilla 等被广泛应用，原理上都是用底层的 Socket 来实现。

4.1.1、FTP协议

FTP 客户端与服务器端进行数据交换必须建立两个套接字，一个作为命令通道，一个作为数据通道。前者用于客户端向服务器发送命令，如登录，删除某个文件，后者用于接收数据，例如下载或上传文件等。

FTP 使用 2 个端口，一个数据端口和一个命令端口（也叫做控制端口）。这两个端口一般是21 （命令端口）和 20 （数据端口）。控制 Socket 用来传送命令，数据 Socket 是用于传送数据。每一个 FTP 命令发送之后，FTP 服务器都会返回一个字符串，其中包括一个响应代码和一些说明信息。其中的返回码主要是用于判断命令是否被成功执行了。

4.1.2、被动模式

FTP数据传输分为主动模式（PORT）与被动模式（PASV）。如果使用被动模式，通常服务器端会返回一个端口号。客户端需要用另开一个 Socket 来连接这个端口，然后我们可根据操作来发送命令，数据会通过新开的一个端口传输。如果使用主动模式，通常客户端会发送一个端口号给服务器端，并在这个端口监听。服务器需要连接到客户端开启的这个数据端口，并进行数据的传输；当前存储板卡使用的是被动模式。

被动方式：命令连接和数据连接都由客户端发起，这样就解决了从服务器到客户端的数据端口的连接被防火墙过滤的问题。

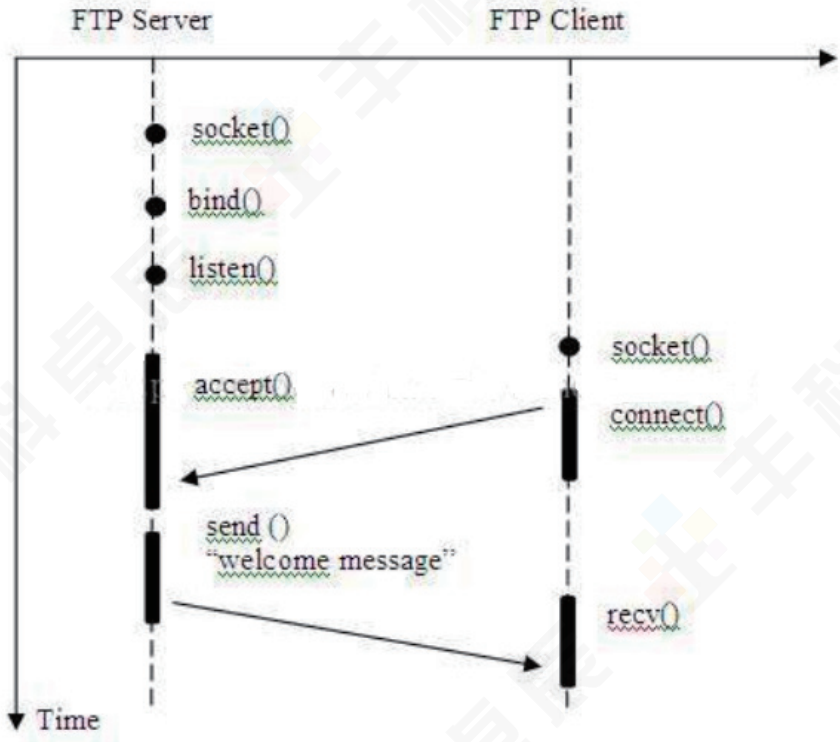
被动模式下，当开启一个 FTP 连接时，客户端打开两个任意的本地端口 (N > 1024 和 N+1) 。

第一个端口连接服务器的 21 端口，提交 PASV 命令。然后，服务器会开启一个任意的端口 (P > 1024)，返回如“227 entering passive mode (127,0,0,1,4,18)”。它返回了 227 开头的信息，在括号中有以逗号隔开的六个数字，前四个指服务器的地址，最后两个，将倒数第二个乘 256 再加上最后一个数字，这就是 FTP 服务器开放的用来进行数据传输的端口。如得到 227 entering passive mode (h1,h2,h3,h4,p1,p2)，那么端口号是 p1*256+p2，ip 地址为 h1.h2.h3.h4。这意味着在服务器上有一个端口被开放。客户端收到命令取得端口号之后，会通过 N+1 号端口连接服务器的端口 P，然后在两个端口之间进行数据传输。

4.1.3、网络编程流程

4.1.3.1、客户端和 FTP 服务器建立 Socket 连接

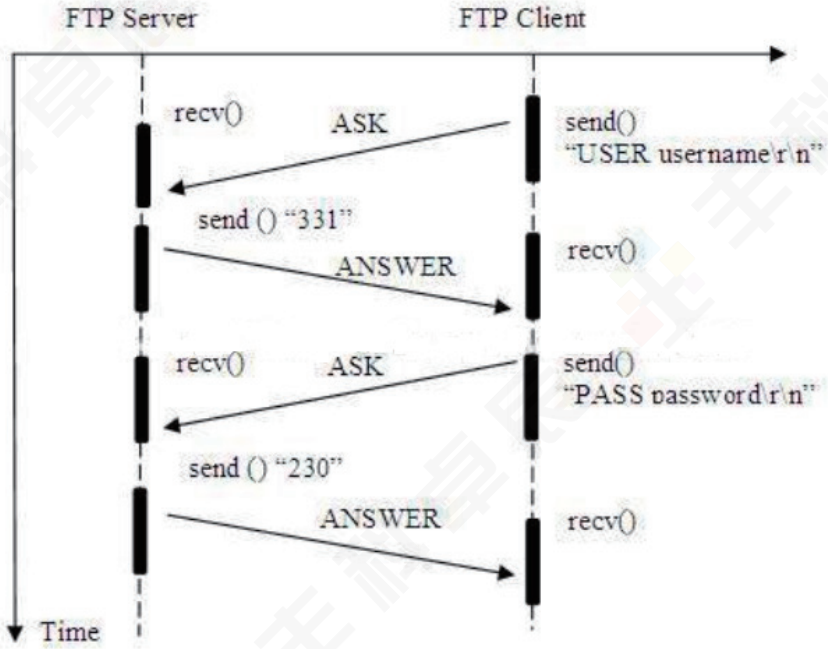
图 1. 客户端连接到服务器端



4.1.3.2、客户端登录 FTP 服务器

当客户端发送用户名和密码，服务器验证通过后，会返回 相应的响应码。然后客户端就可以向服务器端发送命令了。

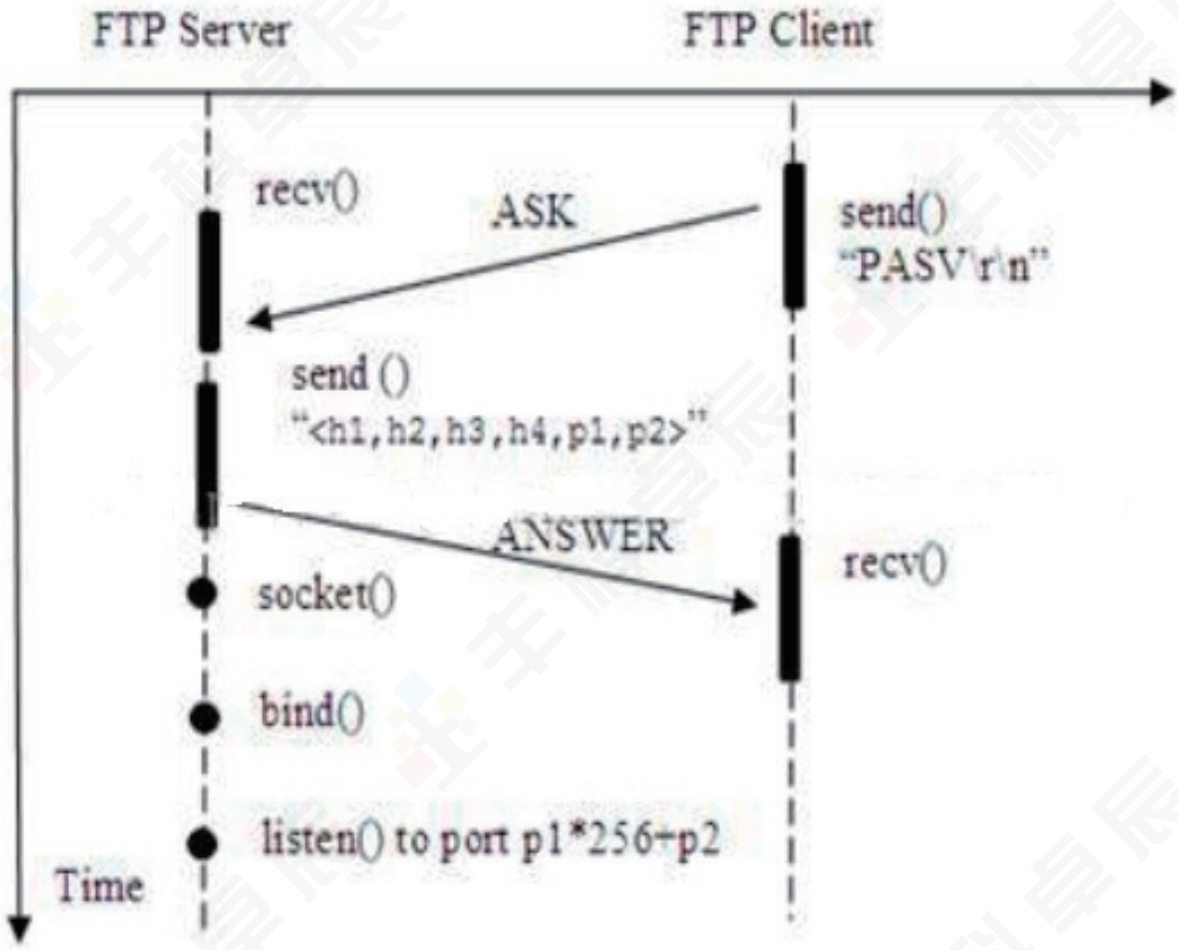
图 2. 客户端登录 FTP 服务器



4.1.3.3、客户端让 FTP 服务器进入被动模式

当客户端在下载/上传文件前，要先发送命令让服务器进入被动模式。服务器会打开数据端口并监听。并返回响应码和数据连接的端口号。

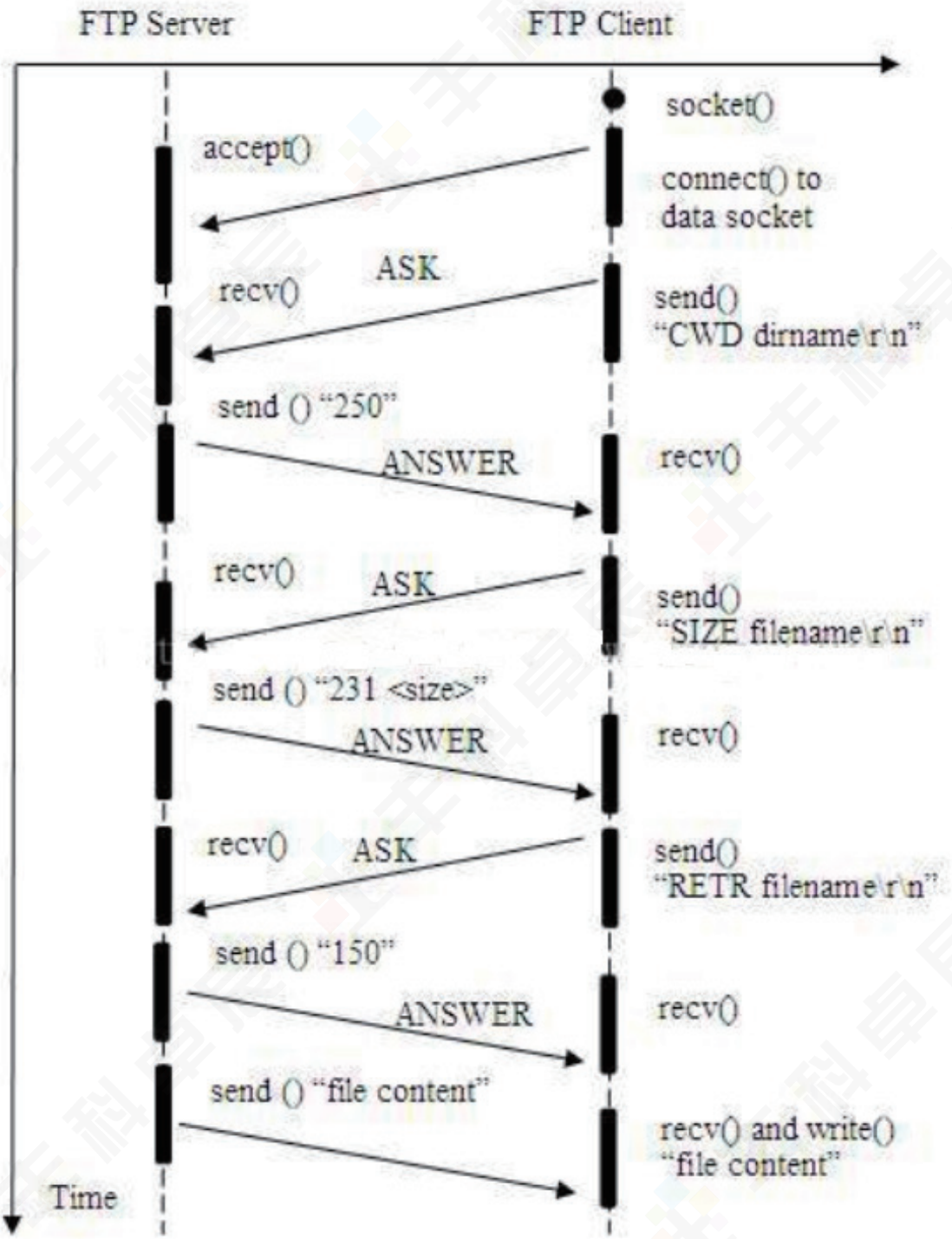
图 3. 客户端让服务器进入被动模式



4.1.3.4、客户端通过被动模式下载文件

当客户端发送命令下载文件。服务器会返回响应码，并向数据连接发送文件内容。

图 4. 客户端从FTP服务器端下载文件


FTP Server
FTP Client
socket()
accept()
connect() to
data socket
ASK
recv()
send()
"CWD dirname\r\n"
send () "250"
ANSWER
recv()
recv()
ASK
send()
"SIZE filename\r\n"
send () "231 <size>"
ANSWER
recv()
recv()
ASK
send()
"RETR filename\r\n"
send () "150"
ANSWER
recv()
send () "file content"
recv() and write()
"file content"
Time

4.1.4、参考C++程序编程

4.1.4.1、控制建立连接

```
WSADATA WSAData;
if (WSAStartup(MAKEWORD(2, 2), &WSAData) != 0)
    return false;

//创建控制连接Socket
g_SocketControl = socket(AF_INET, SOCK_STREAM, 0);
if (g_SocketControl == INVALID_SOCKET)
    return false;

memcpy( m_ServerIP, ServerIP,strlen(ServerIP) );
memcpy( m_ServerIP+strlen(ServerIP), _T("\r\n"), 2);

//定义Socket地址和端口
sockaddr_in serveraddr;
memset(&serveraddr, 0, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_port = htons(iPort);//端口
serveraddr.sin_addr.S_un.S_addr = inet_addr(ServerIP);//地址
//向FTP服务器发送Connect请求
int nConnect = connect(g_SocketControl, (sockaddr*)&serveraddr, sizeof(serveraddr));
if (nConnect == SOCKET_ERROR)
{
    return false;
}

//获得Connect应答信息
if (RecvReply())
{
    if (nReplycode != 220)  // 220代表服务就绪
    {
        closesocket(g_SocketControl);
        return false;
    }
}
```

4.1.4.3、数据连接

```
//向FTP服务器发送PASV命令
memset(Command, 0, MAX_SIZE);
memcpy(Command, "PASV", strlen("PASV"));
memcpy(Command + strlen("PASV"), "\r\n", 2);
if (!SendCommand())
    return false;
//获取PASV命令的应答信息
if (RecvReply())
{
    if (nReplycode != 227)
    {
        closesocket(g_SocketControl);
        return false;
    }
}
//分离PASV命令应答信息
char* part[6];
if (strtok(ReplyMsg, "("))
{
    for (int i = 0; i < 5; i++)
    {
        part[i] = strtok(NULL, ",");
        if (!part[i])
            return false;
    }
    part[5] = strtok(NULL, ")");
    if (!part[5])
        return false;
}
else
{
    return false;
}
//获取FTP服务器数据端口、
unsigned short ServerPort;
ServerPort = unsigned short((atoi(part[4]) << 8) + atoi(part[5]));
//创建数据连接Socket
SocketData = socket(AF_INET, SOCK_STREAM, 0);
if (SocketData == INVALID_SOCKET)
{
    return false;
}
//定义Socket地址和端口
sockaddr_in serveraddr2;
memset(&serveraddr2, 0, sizeof(serveraddr2));
serveraddr2.sin_family = AF_INET;
serveraddr2.sin_port = htons(ServerPort);
serveraddr2.sin_addr.S_un.S_addr = inet_addr(ServerAddr);
//向FTP服务器发送Connect请求
int nConnect;
nConnect = connect(SocketData, (sockaddr*)&serveraddr2, sizeof(serveraddr2));
if (nConnect == SOCKET_ERROR)
{
    return false;
}
```

4.1.4.4、获取文件信息LIST指令

```
//FTP服务器发送LIST命令
if (strlen(dir) != 0)// 不是根目录
{
    memset(Command, 0, MAX_SIZE);
    memcpy(Command, "CWD ", strlen("CWD "));
    memcpy(Command + strlen("CWD "), dir, strlen(dir));
    memcpy(Command + strlen("CWD ") + strlen(dir), "\r\n", 2);
    if (!SendCommand())
    {
        return false;
    }
    memset(Command, 0, MAX_SIZE);
    memcpy(Command, "PWD", strlen("PWD"));
    memcpy(Command + strlen("PWD"), "\r\n", 2);
    if (!SendCommand())
    {
        return false;
    }
    if (RecvReply())
    {
    }
}
memset(Command, 0, MAX_SIZE);
memcpy(Command, "LIST", strlen("LIST"));
memcpy(Command+strlen("LIST"), "\r\n", 2);
if (!SendCommand())
{
    return false;
}
//--------------获得LIST命令的应答信息-------------------------
if (RecvReply())
{
}

//获得LIST命令的目录信息
int nRecv;
char ListBuf[MAX_SIZE+2];

CString strList = _T("");
g_iFileCnt = 0;
memset( g_sFileBuf,0,MAX_FILE_BUF_SIZE );
g_pFileInfo = (sFileInfo*)g_sFileBuf;

while (true)
{
    memset(ListBuf, 0, MAX_SIZE);
    nRecv = recv(SocketData, ListBuf, MAX_SIZE, 0);
    if (nRecv == SOCKET_ERROR)
    {
        closesocket(SocketData);
        return false;
    }
    if (nRecv <= 0)
        break;

    ListBuf[nRecv]='\0';
    CString strTemp( ListBuf );
    CString strNowList = strList + strTemp;

    PraseFileList_UNIX(strNowList);
    strList = strNowList;
}
closesocket(SocketData);
```

4.1.4.5、文件格式

```
//-rwxrw-r--    1 user     group         3014 Nov 12 14:57 cwinvnc337.ESn
//以上的格式分解为：第一段为 - rwxrw - r--，
//第二段为1，
//第三段为user，
//第四段为group，
//第五段为文件长度，
//第六段为月，
//第七段为日，
//第八段为时间，
//第九段为文件名称。
//需要注意的是：如果格式串的第一个字符为d，表示为一个目录信息，比如drwxrw-r--
//另外，第八段有可能不是时间，而是年份，比如2005。
```

4.1.4.6、解析文件列表

```
/*************************************************
Function:  PraseFileList_UNIX
Description: 解析文件列表
Table Accessed:
Table Updated:
Parameter: 以回车换行分隔
Return:   无返回值
Others:
*************************************************/
void PraseFileList_UNIX(CString& sFileList)
{
    CString sLen = _T("");
    CString sFile;
    CString sOneFile;
    int nIdx = 0;
    CString sDate;
    CString stime;
    while (1)
    {
        nIdx = sFileList.Find(_T("\r\n"));
        if (nIdx == -1)
            break;
        sOneFile = sFileList.Left(nIdx);
        sFileList = sFileList.Mid(nIdx + 2);

        if (sOneFile.GetAt(0) == 'd')//第一个字母是d，表示是目录，忽略
        {
            sLen = _T("<DIR>");
        }
        // 获取权限
        CString strTemp = GetSegmentInfo(sOneFile, 0);
        // 获取组
        strTemp = GetSegmentInfo(sOneFile, 0);
        // 获取user
        strTemp = GetSegmentInfo(sOneFile, 0);
        // 获取group
        strTemp = GetSegmentInfo(sOneFile, 0);
        // 获取长度
        if( sLen.GetLength() == 0 )
        {
            sLen = GetSegmentInfo(sOneFile, 0);
        }
        else
        {
            strTemp = GetSegmentInfo(sOneFile, 0);
        }
        // 获取月
        strTemp = GetSegmentInfo(sOneFile, 0);
        sDate = strTemp;
        // 获取日
        strTemp = GetSegmentInfo(sOneFile, 0);
        sDate = sDate + _T(" ");
        sDate = sDate + strTemp;
        // 获取年
        strTemp = GetSegmentInfo(sOneFile, 0);
        sDate = _T(" ") + sDate;
        sDate = strTemp + sDate;

        // 获取文件名
        sFile = GetSegmentInfo(sOneFile, 0);
    }
}
```

4.2、PCIE接口驱动说明

PCIE接口驱动为Windows x64的编译环境，支持VS或QT的X64编译器。

4.2.1、驱动函数

```
// FPGA information
struct fpga_info_list
{
	int num_fpgas;
	int id[MAX_NUM_FPGAS];
	int num_chnls[MAX_NUM_FPGAS];
	char name[MAX_NUM_FPGAS][16];
	int vendor_id[MAX_NUM_FPGAS];
	int device_id[MAX_NUM_FPGAS];
};
typedef struct fpga_info_list fpga_info_list;

// Represents the FPGA device
struct fpga_t;
typedef struct fpga_t fpga_t;

/**
 * 功能：扫描系统内设备
 * 输入：设备信息数组指针，最大支持5个设备；
 * 返回：0代表扫描成功，非0代表扫描失败
 */
CTRLAPI int CTRLCALL fpga_list(fpga_info_list * list);

/**
 * 功能：打开某一个设备
 * 输入：设备ID，该ID号在扫描时获取到；
 * 返回：成功返回控制句柄，失败返回NULL；
 * 说明：每个FPGA设备在使用前，必须要打开，一旦打开，多个线程均可使用同一个控制句柄。
 */
CTRLAPI fpga_t * CTRLCALL fpga_open(int id);

/**
 * 功能：关闭某一个设备
 * 输入：设备控制句柄
 */
CTRLAPI void CTRLCALL fpga_close(fpga_t * fpga);
```

```
/**
 * 功能：发送数据到FPGA，长度单位为4字节，；
 * 输入：控制句柄
 *      通道号
 *      数据指针
 *      数据长度，
 *      数据偏移
 *      last：If last is 1, the channel should interpret the end of this send as the
end of a transaction
 *            If last is 0, the channel should wait for additional sends before the
end of the transaction
 *      超时时间：不为0，则等待ms时间后，FPGA返回，如果为0，则会阻塞，如果阻塞，
多线程操作同一通道会破坏数据。
 * 返回：发送的长度，单位4字节
 */
CTRLAPI int CTRLCALL fpga_send(fpga_t * fpga, int chnl, void * data, int len,
        int destoff, int last, long long timeout);

/**
 * 功能：从FPGA接收数据；
 * 输入：控制句柄
 *      通道号
 *      数据指针
 *      数据长度，
 *      超时时间：不为0，则等待ms时间后，FPGA返回，如果为0，则会阻塞，如果阻塞，
多线程操作同一通道会破坏数据。
 * 返回：接收的长度，单位4字节
 */
CTRLAPI int CTRLCALL fpga_recv(fpga_t * fpga, int chnl, void * data, int len,
        long long timeout);

/**
 * 功能：复位FPGA
 * 输入：控制句柄
 */
CTRLAPI void CTRLCALL fpga_reset(fpga_t * fpga);
```

4.2.2、使用说明

1)首先调用枚举查询系统内的PCIE设备；

2)获取想要操作的设备信息；

3)对设备进行读写操作；

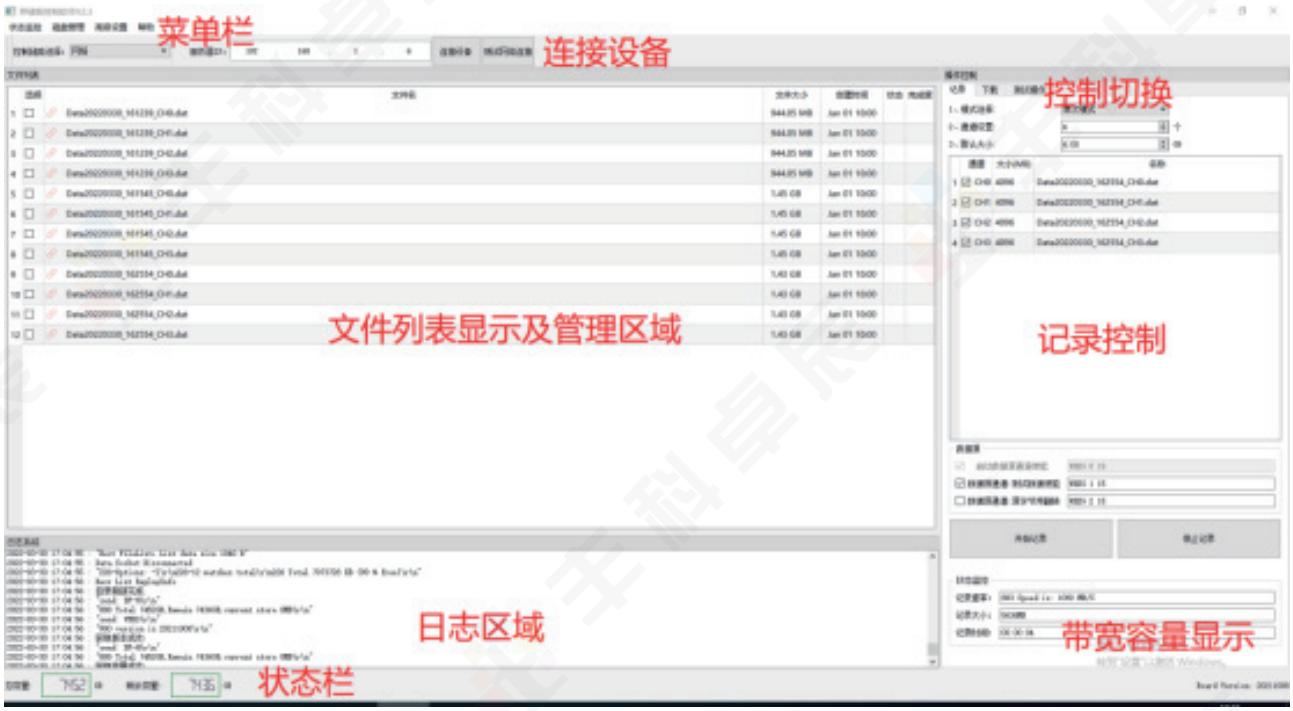
4)从设备进行数据的DMA读写数据。

5)所有的指令通过“通道0”进行发送与接收反馈;在下载或上传时的数据,均使用“通道1”进行传输;

6)每条指令都对应一条指令反馈

5、丰科控制软件使用说明

5.1、主界面



5.2、连接板卡

可选择网络或PCIE进行板卡的连接。如果是网络则需要输入IP地址,内部默认端口号为21。

5.3、记录控制

记录功能分两种,一种是单次记录,一种是连续记录。记录可启动多个通道。

◇ 单次记录:指记录到设置大小后不再记录。

指令中会携带一个数据大小设置,记录数据容量到达这个大小后,则不再将收到的数据写盘,最后停止下来后,则该文件为设置大小。如果数据量还未达到设置的大小,停止下来后,则为实际大小。

◇ 单次记录：指记录到设置大小后不再记录。

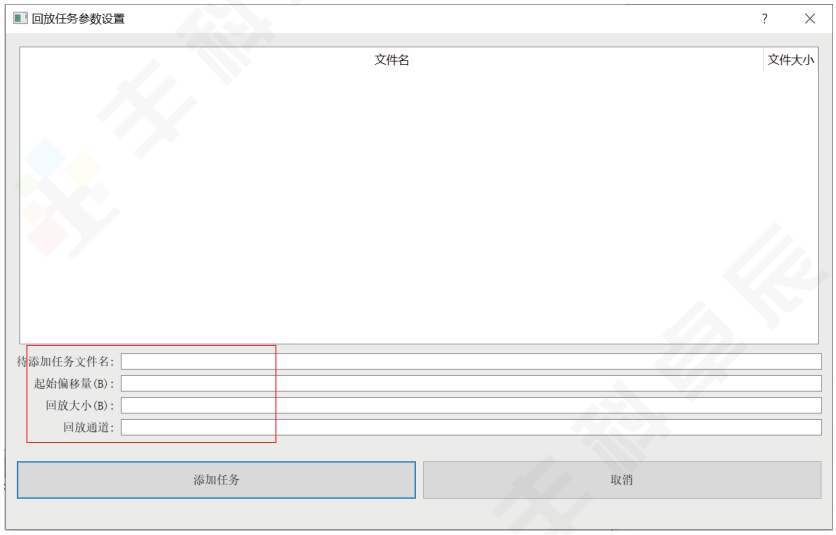
指令中会携带一个数据大小设置，记录数据容量到达这个大小后，则不再将收到的数据写盘，最后停止下来后，则该文件为设置大小。如果数据量还未达到设置的大小，停止下来后，则为实际大小。

5.4、下载控制

在文件列表区域右键可选择文件进行下载任务添加，支持添加多个任务。然后切换到下载窗口，进行开始下载。

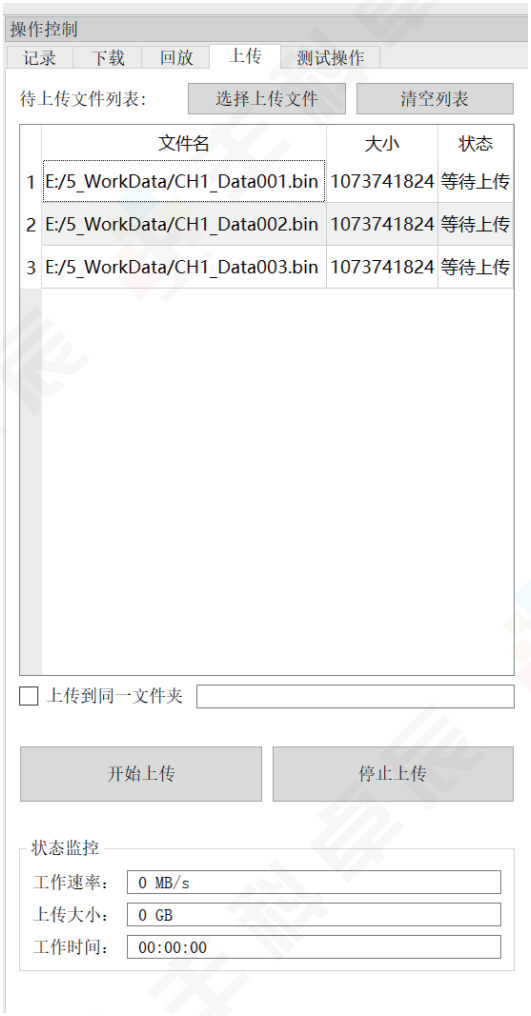
5.5、回放控制

切换到回放功能操作界面，可添加回放任务，添加完回放任务后，点击开始回放，可进行数据的回放控制。



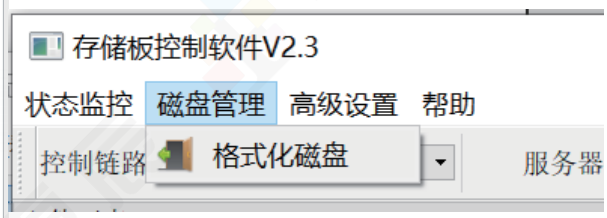
5.6、数据上传

数据上传需要数据文件是64KB的整数倍。通过“选择上传文件”，然后添加上传文件，然后点击开始上传，进行上传流程的操作。



5.7、文件操作

在文件列表显示区域，右键可弹出操作菜单。进行相关的文件操作。或在菜单中的“磁盘管理”上，可选择“格式化磁盘”操作。



北京丰科卓辰电子技术有限公司

v1.0 2022.5

联系热线 010-57325880
联系邮箱 guangrui.liu@fkzctech.com
首页网站 www.fkzctech.com
公司地址 北京市昌平区科技园区超前路甲1号6号楼308室