Импорт замеров с виброприборов компании “Вибро-Центр”

v1.00 20.04.2023

AndreySchekalev@vibrocenter.ru

Схема уровней для связи с прибором

3. Разбор замера как структуры конкретного прибора. Преобразование замера в универсальный формат замера	ViAna-2: DoImportViana2() в файле ImportViana2.pas Диана-2М: ImportKorsarM() в файле ImportKorsar.pas ViAna-1: DoImportViana1() в файле ImportViana1.pas ViAna-4: DoImportViana4() в файле ImportViana4.pas Vibro Vision-2: DoImportVV2() в файле ImportVV2.pas	
2. Передача бинарных данных через последовательность структур TLinkList	Обмен низкоуровневыми фреймами TLinkList TestImportForm.pas	
1. Объект «Универсальный порт связи»	TUSBLinkPort в файле LinkPortClass.pas	TEthernetLinkPort в файле LinkPortClass.pas

0. USB или Ethernet server	USB Vendor Device:
	VC_USB_VID = $0441; // vendor ID
	VC_USB_PID = $51C9; // Product ID
	VCUSBHdr.pas через библиотеку lib_usb	

0. USB драйвер

В примере VCUSBHdr.pas обмен идёт через библиотеку LibUsb – A cross-platform user library to access USB devices.
Сайт: https://libusb.info/

Для Win32 можно использовать нашу прослойку VCUSBDLL.dll через файл VCUSBDllHdr.pas

LibUsb в Windows

Для установки драйвера и нашего файла vcusb.inf используйте:
USB/InstallDriver.exe

В каталог программы положить файл libusb-1.0.dll

LibUsb в Linux

```
sudo  apt-get install libusb-1.0
```

или

```
su  -c  "apt-get install libusb-1.0"
```

Чтобы разрешить работу с нашим прибором по USB нужно сделать файл ~/vc.rules:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="0441", ATTRS{idProduct}=="51c9", MODE="0666"
SUBSYSTEM=="usb_device", ATTRS{idVendor}=="0441", ATTRS{idProduct}=="51c9", MODE="0666"
```

и скопировать его:

```
sudo cp  ~/vc.rules   /etc/udev/rules.d/
sudo udevadm control --reload-rules
```

Пример файла лежит в
Linux/vc.rules

Используется USB Vendor Device:

```
#define  USB_DEVICE_VENDOR_ID            0x0441
#define  USB_DEVICE_PRODUCT_ID           0x51C9
#define  USB_DEVICE_MANUFACTURE_NAME     "Vibro-Center/Dimrus"
#define  USB_DEVICE_PRODUCT_NAME         "Vibro-Center/Dimrus Device"
```

Два Bulk Device endpoints:

```
#define UDI_VENDOR_EP_BULK_IN            0x81     // Передача с прибора на компьютер
#define UDI_VENDOR_EP_BULK_OUT           0x02     // Передача с компьютера на прибор
```

Для USB 1.1 максимальный размер пакета = 64 байта

Для USB 2.0 и выше максимальный размер пакета = 512 байт

Функции VCUSBHdr.pas или VCUSBDllHdr.pas:

Возвращают S_OK =0; или ошибки, если !=0

Инициализация – всегда вызывать один раз перед началом работы с USB. Проверяет, что драйвер LibUsb установлен.

```
function InitUsb(): HRESULT;
```

Освободить библиотеку – вызывать в конце программы

```
procedure DoneUsb();
```

Вызывать перед обменом с прибором.
Ищет наш USB device, создаёт объекты для обмена.

```
function VCUSBInit(): HRESULT;
```

Вызывать после окончания обмена – закрывает USB device, уничтожает объекты.
Можно вызывать неоднократно.

```
function VCUSBDone(): HRESULT;
```

Передать массив pBuffer размером cbBuffer байт из компьютера в прибор.
В случае успеха возвращеает cbWritten – сколько байт записано.
uTimeout – таймаут на ожидание в милисекундах. Не должен быть ==0. Например, 100

```
function VCUSBWrite(pBuffer: pointer; cbBuffer: DWORD; var cbWritten: DWORD; uTimeout: DWORD): HRESULT;
```

Принять cbBuffer байт в массив pBuffer размером из прибора в компьютер.
В случае успеха возвращает cbRead – сколько байт записано.
uTimeout – таймаут на ожидание в милисекундах. Не должен быть ==0. Например, 100

```
function VCUSBRead(pBuffer: pointer; cbBuffer: DWORD; var cbRead: DWORD; uTimeout: DWORD): HRESULT;
```

Типичный сценарий обмена:

```
InitUsb(); // В начале программы

…

If (VCUSBInit() = S_OK) then begin // Начало обмена
        while (data_count > 0) do begin
                if VCUSBWrite() = S_OK then // записать команду
                        VCUSBRead(); // принять ответ
        end;

        VCUSBDone(); // Конец обмена

End;

…

DoneUsb(); // В конце программы
```

1. Объект «Универсальный порт связи»

Позволяет использовать одинаковые методы для связи по USB, Ethernet или из файла (а раньше использовались ещё и COM и LPT).

Обычно использует метод:

```
function TLinkPort.SendCommandGetBuffer(var aCommand: TCommand;
        aGetBuf: pointer; aGetBufLen: LongWord): integer;
```

Метод посылает низкоуровневую команду типа TCommand и получает ответ в массив aGetBuf длиной до aGetBufLen байт. Проверяет CRC пришедших данных.

Возвращает LinkResultOk = 1 или код ошибки (в файле LinkTypes.pas)

Порядок байт в полях длинее 1 байта – Low8..High8 (little-endian, интеловский).
Упакованы на 1 байт. Без выравнивания!
#pragma pack(1)

Структура TCommand

Упакована на 1 байт. 14 байт длиной
Файл LinkTypes.pas или Link.h

```
Type PCommand =^TCommand;
 TCommand = packed record
   Sign     : TSign;  // 3 байта
   Command   : byte;  // 1 байт
   Param1    : word;  // 2 байта
   Param1dop : word;  // 2 байта
   Param2    : word;  // 2 байта
   Param2dop : word;  // 2 байта
   CRC       : word;  // 2 байта
 end;
Assert(sizeof(TCommand) = 14);
```

```
typedef __packed struct {
  TSign Sign;  // 3 байта
  u8   Command;   // 1 байт
  u16  Param1;  // 2 байта
  u16  Param1dop;  // 2 байта
  u16  Param2;    // 2 байта
  u16  Param2dop;  // 2 байта
  TCRC CRC;       // 2 байта
} TCommand;
const size_t szTCommand = sizeof(TCommand);
static_assert(szTCommand == 14, "");
```

Незаполненные поля забиваются 0.

```
Type  TSign     = array [1..3] of AnsiChar; //сигнатура
const Sign: TSign       = ('V','C','#');  //сигнатура команды
```

TCRC = word или uint16_t = 2 байта длиной беззнаковое
Считается по такому странному алгоритму (LinkUtil.pas):

```
function CalcCRC(var Buf;Len:longint):word;
var
  i:longint;
  CRC:word;
  Dop:byte;
begin
```

```
TCRC CalcCRC(void* Buf, u32 Len)
{
        u32 i;
        TCRC CRC, c, Dop;
        CRC = 0xAAAA;
        for (i = 0; i < Len; i++)
```

```
  CRC:=$AAAA;
  for i:=1 to Len do begin
    Dop:=0;
    if (CRC and $8000)<>0 then Dop:=1;
    CRC:=((CRC and $7FFF) shl 1)+Dop;
    CRC:=CRC xor TArrByte(Buf)[i];
  end;
  Result:=CRC;
end;
```

```
        {
                Dop = ((CRC & 0x8000) != 0)?1:0;
                CRC = ((CRC & 0x7FFF) << 1) | Dop;
                c = (TCRC)((char*)Buf)[i];
                CRC ^= c;
        }
        return CRC;
}
```

Считается по длине-2 байта.
В последних 2 байтах лежит CRC, пришедшее с прибора. С ним и сравниваем.

```
Command.CRC := CalcCRC(Command, szTCommand - 2);
```

```
TCRC crc := CalcCRC(Buf^, Len - 2);
```

Дополнительные байты для USB 1.1

Если прибор использует USB 1.1 (TInfoPribor.Protokol < 200), то все массивы при передаче и приёме дополняются (выравниваются) до длины кратной 64 байта (это размер Bulk пакета для протокола USB 1.1).

Это тяжёлое наследие Legacy кода, которое приходится поддерживать…

Например, при передаче структуры TCommand длиной 14 байт, после неё приписывается 50 байт нулей и передаётся блок 64 байта.

При приёме структуры TinfoPribor (для команды cmdTestPribor обязательно использует режим USB 1.1 для совместимости всех приборов) приходит 64 байта, но только 50 из них полезных. CRC проверяется по первым 48 байтам.

Все пришедшие массивы данных также выровнены до кратных 64 байта.

Это реализовано в файле LinkPortClass.pas, функции:

```
function TUsbLinkPort.UsbRead(buf: pointer; Len: integer): integer;
```

```
function TUsbLinkPort.UsbWrite(buf: pointer; Len: integer): integer;
```

Для USB 2.0 (TInfoPribor.Protokol >= 201) выравнивания нет. Какой размер передаём в функцию, столько и уходит по USB.

2. Передача бинарных данных через последовательность структур TLinkList

Обеспечивает формирование команды в структуру TCommand, передачу команды, приём ответа, повторы в случае проблем со связью.

Далее описан Список команд (поле TCommand.Command)

Тест прибора, получить информацию о приборе

```
const cmdTestPribor      = 1;
```

В ответе получает структуру TInfoPribor размером 50 байт. Пример:

```
Result := (SendCommandRepeat(cmdTestPribor, 0, 0, 0, 0, @Info, szTInfoPribor) = LinkResultOk);
```

Файл LinkTypes.pas или Link.h

```
 //структура информации о приборе - всего 50 байт
Type PInfoPribor =^TInfoPribor;
 TInfoPribor = packed record
   Sign       : TSign; //сигнатура
   Pribor     : TWord4; //тип прибора
   Numer      : TWord4; //
   Version    : TWord4; //версия програмного обеспечения прибора
   Protokol   : TWord4; //версия протокола обмена
   Flash      : TWord4; //размер flash
   EEPROM     : TWord4; //размер EEPROM
   CountClaster: TWord4; //число секторов fat
   ByteClaster : TWord2;  //размер сектора fat
   HideClaster : TWord2;
   FreeClaster : TWord2;
   AllClaster  : TWord2;
   Data       : array [1..9] of byte;
   CRC        : TCRC;
   Align64    : array[1..14]of byte; // Добивка до 64 байт, но полезного только 50 байт
 end;
const
   szTInfoPribor = 50;
   szTInfoPriborFull = SizeOf(TInfoPribor);
Assert(sizeof(TInfoPribor) = 50);
Assert(szTInfoPriborFull = 64);
```

```
//описание структуры информации о приборе
typedef __packed struct {
 TSign Sign;      //сигнатура
 u32 Pribor;      //тип прибора
 u32 Numer;       //номер прибора
 u32 Version;     //версия програмного обеспечения прибора
 u32 Protokol;    //версия протокола обмена
 u32 Flash;       //размер flash
 u32 EEPROM;      //размер EEPROM
 u32 CountClaster;//число данных секторов fat
 u16  ByteClaster; //размер сектора fat
 u16  HideClaster; //число скрытых секторов fat
 u16  FreeClaster; //число свободных кластеров FAT
 u16  AllClaster;  //всего секторов fat
 char Data[9];
 TCRC CRC;
 char Align64[14]; // Добивка до 64 байт для совместимости с USB 1.1 и USB 2.0
} TInfoPribor;
const size_t szTInfoPribor = sizeof(TInfoPribor);
static_assert(szTInfoPribor == 64, "");
```

Эта команда выполняется в режиме USB 1.1:

Структура TCommand дополняется до 64 байт.

Приходит 64 байта структуры TInfoPribor (для команды cmdTestPribor обязательно использует режим USB 1.1 и выравнивание блока до 64 байт для совместимости всех приборов) приходит 64 байта, но только 50 из них полезных. CRC проверяется по первым 48 байтам.

Специальный код отработки этой команды есть в файле LinkPortClass.pas:

```
function TUsbLinkPort.SendCommandGetBuffer(var aCommand: TCommand; aGetBuf: pointer; aGetBufLen: LongWord): integer;
```

Получить список каталогов и замеров

```
const cmdReadListZamer    = 9;
```

param2dop = 0 – возвращает число замеров

param2dop = от 1 до числа замеров – возращает инфу по замеру

Сперва запрос количества каталогов и замеров (поле TLinkFrame.Count):

```
result := SendCommandRepeat(cmdReadListZamer, 0, 0, 0, 0, @Frame, SizeOf(TLinkFrame));
```

Принимает структуру TLinkFrame

```
//нулевой фрейм
Type  PLinkFrame =^TLinkFrame;
 TLinkFrame = packed record
   Sign       : TSign; //сигнатура
   Numer      : word;  //номер
   Types      : byte;  //тип
   Count      : TWord2; //число фреймов
   Length     : TWord2; //длина одного фрейма
   CRC        : TCRC;  //CRC
 end;
Assert(sizeof(TLinkFrame) = 12);
```

```
typedef __packed struct {
  TSign       Sign;
  u16         Numer;
  u8          Types;
  u16         Count;
  u16         Length;
  TCRC        CRC;
} TLinkFrame;
const size_t szTLinkFrame = sizeof(TLinkFrame);
static_assert(szTLinkFrame == 12, "");
```

Затем цикл по всем фреймам с прибора:

```
for frame := 1 to Frame.Count do begin
            result := SendCommandRepeat(cmdReadListZamer, 0, 0, 0, frame, @FrameList, SizeOf(TFrameList));
End;
```

Каждый раз приходит структура TFrameList размером 71 байт.

```
Type PFrameList =^TFrameList;
 TFrameList = packed record
   Sign     : TSign;
   Numer    : word;
   List     : TLinkList;
   CRC      : word;
 end;
Assert(sizeof(TFrameList) = 71);
```

```
typedef __packed struct {
  TSign           Sign;
  u16             Numer;
  TLinkList       List;
  TCRC            CRC;
} TFrameList;
const size_t szTFrameList = sizeof(TFrameList);
static_assert(szTFrameList == 71, "");
```

Полезное в ней – поле List типа TLinkList:

Type PLinkList = ^TLinkList;
TLinkList = packed record
ID_Low : word; // ссылка на данные - младшее слово, если ID>64k
Numer : word; // порядковый номер
Types : word; // Тип данных
DateTime : TDateTimeMega; // Дата и время
UpLevel : TInt4; //Ссылка на верхний уровень, если есть; Для старого протокола (старая Диана-2М) ограничено 32к
Note : array[1..30] of AnsiChar;
ID_High : word; // ссылка на данные - старшее слово, если ID>64k
Reserv : array[1..14] of byte;
end;
Assert(sizeof(TLinkList) = 64);
	typedef __packed struct //всего 64 байта
	{
	u16 ID_Low; // ссылка на данные - младшее слово, если ID>64k
	u16 Numer; //порядковый номер
	u16 Type; //Тип данных
	struct {
	char DSec,Sec,Min,Hour;
	u16 Year;
	char Month,Day;
	} DateTime;
	//s16 UpLevel; //Ссылка на верхний уровень, если есть
	//u16 Size;
	s32 UpLevel; //Ссылка на верхний уровень, если есть
	char Note[30];
	u16 ID_High; // ссылка на данные - старшее слово, если ID>64k
	char reserv[14];
	} TLinkList;
	const size_t szTLinkList = sizeof(TLinkList);
	static_assert(szTLinkList == 64, "");

uint32_t ID = номер сектора во флешке прибора. Должна быть > 0. Вычисляется так:

```
// ID_High==0xFF или 0x00 - не используем
if (pntr^.ID_High<>$FF) then Result:=(TID(pntr^.ID_High) SHL 16) OR TID(pntr^.ID_Low)
else Result:=TID(pntr^.ID_Low);
```

TLinkList.Type = equFatTypeDir = 9 – каталог

Тогда в поле TLinkList.Note содержится строка однобайтных символов в кодировке CP1251 (null-terminated, как в языке C).

В поле TLinkList.UpLevel = ID предка в дереве или 0 для верхнего уровня (нет предка).

TLinkList.Type = equFatTypeData = 0 – замер

TLinkList.Numer = номер замера в каталоге-родителе.

TLinkList.DateTime = дата и время создания.

В поле TLinkList.UpLevel = ID предка в дереве или 0 для верхнего уровня (нет предка).

остальные типы пока не описал.

Считать замер с прибора

```
const cmdReadData    = 2;
```

param1   -   тип = equFatTypeData

param2   -   ID_Low

param1dop -   ID_High

param2dop -   0 - инфо по блокам, 1..N - номер блока

Получить количество блоков:

```
result := SendCommandRepeat(cmdReadData, equFatTypeData, ID_Low, ID_High, 0, @Frame,
SizeOf(TLinkFrame));
```

Затем цикл по всем блокам с прибора:

```
for block := 1 to Frame.Count do begin
        result := SendCommandRepeat(equFatTypeData, ID_Low, ID_High, block, @Buf, BufLen);
        if result = LinkResultOk then
            FileWrite(TmpFileHandle, PArrByte(Buf)^[SizeOf(TSign) + sizeof(word) + 1],
                    BufLen - SizeOf(TSign) - sizeof(word) - sizeof(TCRC));
End;
```

Каждый раз приходит буфер данных Buf размером Frame.Length байт.
Первые 5 байт – заголовок:
  Sign   : TSign;
  Numer   : word;
Далее Frame.Length-7 байт – полезные данные.
CRC   : word; – Последние 2 байта

Полезные данные можно сохранить в файл для дальнейшей разборки структуры замера или в поток (TMemoryStream, TFileStream).

3. Разбор замера как структуры конкретного прибора

Тут разбираем бинарный файл, сохранённый в предыдущем разделе, и преобразовываем в замер прибора. Для каждого прибора структура замера своя.

Порядок байт в полях длинее 1 байта – Low8..High8 (little-endian, интеловский).

В примерах замеры транслируются в нашу стандартную структуру TBufOneRec (ImportDefs.pas).
Каждый канал транслируется в одну запись TBufOneRec (сигнал или спектр).

ViAna-2

Описание структур в файле LinkTypesViana2.pas или cpp/ViAna2/Measurement_Viana2.h

Функция преобразования файла: DoImportViana2() в файле ImportViana2.pas

Все структуры замера выровнены на **4 байта**:
#pragma pack(4)

Замер ViAna-2 содержит:

```
Param: T_VV2_Parameters; // заголовок (=100 байт)
if Param.DataLen[0]>0 then begin
        MeasHdr: _Viana2_TMeas; // Параметры замера и каналов длиной sz_Viana2_TMeas (=1920 байт)
        if Param.DataLen[1]>0 then
                MeasData:  MeasData; // Массив отсчётов с данными длиной Param.DataLen[1] байт
end;
```

CRC считается тем-же алгоритмом, что и в п.1, но сначала поле CRC обнуляется, а затем считается по полной длине структуры, включая и обнулённое поле CRC.

```
function CheckCRC(var Buf; Len : TInt2; var aCRC : TCRC): boolean;
var CRC : TCRC;
begin
CRC:=aCRC;
aCRC:=0;
Result:= CRC=CalcCRC(Buf,Len);
aCRC:=CRC;
end;
```

```
TCRC CheckCRC16(void* Buf, u32 Len, TCRC* aCRC)
{
        register TCRC CRC, fl;
        CRC = *aCRC;
        *aCRC = 0;
        fl = (CRC == crc16((u8*)Buf, Len));
        *aCRC = CRC;
        return fl;
}
void SetCRC16(void* Buf, u32 Len, TCRC* aCRC)
{
        *aCRC = 0;
        *aCRC = crc16((u8*)Buf, Len);
}
```

MeasHdr.CH[] – 3 канала (CH_1, CH_2, CH_TACH)

MeasHdr.CH[ch].Table[_Viana2_ztWaveform] – Сигнал канала ch

MeasHdr.CH[ch].Table[_Viana2_ztSpectrum] – Спектр канала ch

MeasHdr.CH[ch].Table[_Viana2_ztXXX].OffT – смещение (в байтах) массива отсчётов в массиве MeasData. Тип одного отсчёта = float.

В зависимости от маски DataFormat спектр может быть:

Амплитудным: A[0], A[1], ... A[AllX-1]

Комплексным: Re[0], Im[0], Re[1], Im[1], ... Re[AllX-1], Im[AllX-1]

MeasHdr.CH[ch].ValueAvg[XXX] – различные посчитанные значения _Viana2_svXXX (СКЗ, Пик, gE). В поле ValueAvgCalculated – битовые флажки, какие из значений MeasHdr.CH[ch].ValueAvg[XXX] были посчитаны.

Диана-2М

Описание структур в файле LinkTypes.pas или cpp/Diana2M/ZAMER.H

Функция преобразования файла: ImportKorsarM() в файле ImportKorsar.pas

Все структуры замера выровнены на **4 байта**:
#pragma pack(4)

Hdr : THdr;

по смещению Hdr.Next лежит Hdr.Table штук Table : THdrTable;

ViAna-1

Описание структур в файле LinkTypesViana.pas

Функция преобразования файла: DoImportViana1() в файле ImportViana1.pas

ViAna-4

Описание структур в файле LinkTypesViana4.pas

Функция преобразования файла: DoImportViana4() в файле ImportViana4.pas

Vibro Vision-2

Описание структур в файле LinkTypesVV2.pas или cpp/VV2/HandleMeasurement.h

Функция преобразования файла: DoImportVV2() в файле ImportVV2.pas