

**ОТЧЕТ**

**по производственной практике ПП.01.01**

**по профессиональному модулю ПМ.01**

**«Разработка программных модулей программного обеспечения для компьютерных систем»**

Специальность 09.02.03 «Программирование в компьютерных

системах»

Выполнил студент(ка) 3 курса группы П2-17

Андрейко Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)

Принял преподаватель

Родичкин П. Ф.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка)

Королёв 2020

Введение

[1.Характеристика объекта практики 4](#_Toc42259130)

[1.1. Структура организации, характеристики основных видов деятельности 4](#_Toc42259131)

[1.2. Описание структуры управления, где проходит практика 4](#_Toc42259132)

[Основные функции управления: 5](#_Toc42259133)

[1.3. Должностные обязанности техника-программиста, инженера-программиста 5](#_Toc42259134)

[1.4. Документооборот организации (структурного подразделения) 6](#_Toc42259135)

[1.5. Состав технических средств обработки данных 7](#_Toc42259136)

[1.6. Состав пакетов прикладных программ (ППП) функционального и процедурно-ориентированного типов, виды автоматизированных систем обработки информации и управления 7](#_Toc42259137)

[1.7. Характеристика обобщенных технологических процессов сбора, обработки и передачи информации 8](#_Toc42259138)

[1.8. Краткая характеристика применяемых методов проектирования подсистем и отдельных задач 8](#_Toc42259139)

[Информационные технологий предприятия 8](#_Toc42259140)

[Программное обеспечение предприятия 8](#_Toc42259141)

[2. Разработка отдельных программных модулей и их листинги 8](#_Toc42259142)

[2.1 Разработка технического задания 8](#_Toc42259143)

[2.2 Разработка 9](#_Toc42259144)

[ЗАКЛЮЧЕНИЕ 10](#_Toc42259145)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 11](#_Toc42259146)

[Приложение 1 12](#_Toc42259147)

[Листинг 1. Пишем заголовочный файл RTSCM.h 12](#_Toc42259148)

[Листинг 2. Пишем файл RTSCM.cpp для системы телеметрических измерений РТСЦМ 13](#_Toc42259149)

[Листинг 3. Пишем заголовочный файл Orbita.h 17](#_Toc42259150)

[Листинг 4. Пишем файл Orbita.cpp для обработки и декодирования данных передоваемых в рамках “Орбита” 20](#_Toc42259151)

[Листинг 5. Пишем заголовочный файл decCCSDS.h 34](#_Toc42259152)

[Листинг 6. Создаём файл decCCSDS.cpp для принятия и декодирования CCSDS-пакетов ТМИ. 35](#_Toc42259153)

[Приложение 2 40](#_Toc42259154)

**Введение**

На 3 курсе обучения в ККМТ, студентом группы П2-17 Андрейко Борисом была пройдена производственная практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» в Научно-производственное предприятии «МЕРА». Студент получил задание реализовать обработку данных, передаваемых по каналам связи (радиоканалам) в форматах "Орбита".

# 1.Характеристика объекта практики

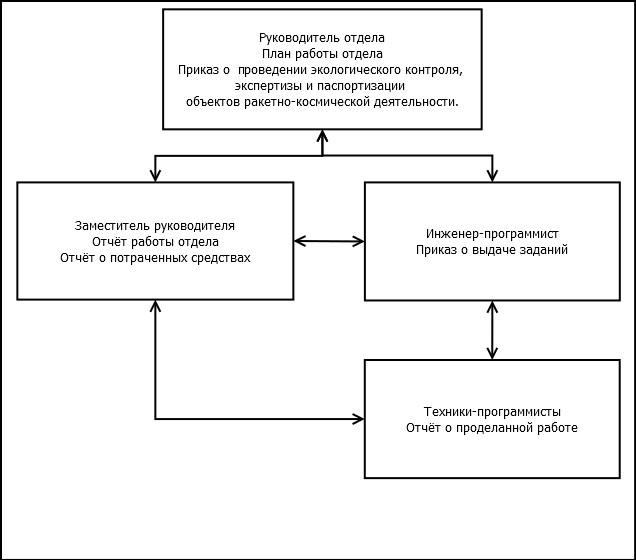
## 1.1. Структура организации, характеристики основных видов деятельности

Организационная структура организации представлена на Рис. 6.1 в Приложении 2.

## 1.2. Описание структуры управления, где проходит практика

Во главе стоит руководитель управления, в его подчинении находятся 3 подразделения:

1. Отдела мониторинга, лицензирования и аккредитации.
2. Центр диагностики качества образования и содействия трудоустройству.



**Рис. 1.2.2.** Информационная структура отдела

## Основные функции управления:

* контроль и управление системой менеджмента качества на предприятии;

## 1.3. Должностные обязанности техника-программиста, инженера-программиста

Ведущий программист:

* На основе анализа математических моделей и алгоритмов решения

экономических и других задач разрабатывает программы, обеспечивающие возможность выполнения алгоритма и, соответственно, поставленной задачи средствами вычислительной техники, проводит их тестирование и отладку.

* Разрабатывает технологию решения задачи по всем этапам обработки

информации.

* Осуществляет выбор языка программирования для описания алгоритмов и структур данных.
* Определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля.
* Осуществляет контроль за проведением профилактических работ, устранением неисправностей, возникающих в процессе эксплуатации средств вычислительной техники.
* Определяет объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их

функциональному назначению.

* Осуществляет запуск отлаженных программ и ввод исходных данных,

определяемых условиями поставленных задач.

* Проводит корректировку разработанной программы на основе анализа

выходных данных.

* Разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию.
* Определяет возможности использования готовых программных продуктов.
* Осуществляет сопровождение внедренных программ и программных

средств.

* Разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации.
* Выполняет работу по унификации и типизации вычислительных процессов.
* Принимает участие в создании каталогов и картотек стандартных программ, в разработке форм документов, подлежащих машинной обработке, в проектировании программ, позволяющих расширить область применения вычислительной техники.

## 1.4. Документооборот организации (структурного подразделения)

В комплект документов входят следующие виды документов:

* + Положение о подразделениях;
  + Должностные инструкции;
  + Рабочие инструкции;
  + Технологические паспорта;
  + Данные о качестве;
  + Государственные стандарты;
  + Отраслевые стандарты;
  + Стандарты предприятия;
  + документы по планированию деятельности отдела;
  + перечень видов записей и данных по качеству отдела и, собственно, записи и данные;
  + результаты проверок состояния помещений и соответствующего оборудования (энергоснабжение, вентиляция и т.д., при необходимости);
  + перечень оборудования;
  + сведения о ремонтах оборудования, его проверках;
  + наличие других локальных актов, определяющих деятельность отдела.

## 1.5. Состав технических средств обработки данных

|  |
| --- |
| Мышь Logitech Wireless Mouse M170, USB, беспроводная, черный |
| Ноутбук Lenovo IdeaPad S340-15 AMD Ryzen 5 3500U |
| Операционная система MICROSOFT Windows 10 Professional OEM |

## 1.6. Состав пакетов прикладных программ (ППП) функционального и процедурно-ориентированного типов, виды автоматизированных систем обработки информации и управления

* Visual Studio Code
* C++

## 1.7. Характеристика обобщенных технологических процессов сбора, обработки и передачи информации

* Порядок сбора данных:
* Анализ данных;
* Отслеживание изменений данных
* Изменение и отправка изменённых данных

## 1.8. Краткая характеристика применяемых методов проектирования подсистем и отдельных задач

## Информационные технологий предприятия

* GitHub

### Программное обеспечение предприятия

* Windows 10 Professional OEM
* Visual Studio Code

# Разработка отдельных программных модулей и их листинги

## 2.1 Разработка технического задания

* Данный проект создан для декоммутации телеметрической информации.

1. Назначение разработки:
   1. Функциональное назначение программы:

Функциональным назначением программы является обработка информации, связанная с поступающими данными.

* 1. Эксплуатационное назначение программы:

Программа может эксплуатироваться в любых условиях. Не требует специального оборудования и дополнительных установок. Пользователями программы должны иметь познания в телеметрии, либо же быть доверенными лицами компаний.

1. Требования к программному изделию:

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* функция чтения конфигурации из ini-файла
* функция формирования информации об орбите на основе данных телеметрии
* функция формирования видео файлов в формате MPEG и последовательности JPEG файлов на основе данных от БСВК
  1. Организация входных и выходных данных

Входные данные программы представлены в виде поток байт в формате CCSRS.

* 1. Требования к составу и параметрам технических средств
     + Радиопередатчик
     + Аппаратные средства, достаточные для приёма и обработки потока информации объёмом в 125000 байт/с

1. Требования к программным средствам, используемым программой:

* Драйверы для получения информации от радиоприёмников

## 2.2 Разработка

Программный код приложения приведен в Приложении 1.

# ЗАКЛЮЧЕНИЕ

Перед прохождением производственной практики в Научно Производственном предприятии “МЕРА” мной были поставлены следующие основные цели:

* приобрести опыт работы по специальности;
* закрепить теоретические знания, полученные во время учебы;
* выполнение требований и действий, предусмотренных программой производственной практики и заданием руководителя;
* закрепить навыки в разработке проектной и технической документации;
* закрепить навыки отладки и тестирования программных модулей.

По окончании практики я достиг своих целей. Для этого я использовал учебную литературу, интернет – источники, документацию средств разработки.

Во время прохождения практики я приобрел опыт работы по специальности. Также был закреплен навык разработки проектной и технической документации и навык отладки и тестирования программных модулей.

По окончании практики был составлен отчет.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. <https://code-live.ru/tag/cpp-manual/>
2. <https://habr.com/ru/>

# Приложение 1

## Листинг 1. Пишем заголовочный файл RTSCM.h

#ifndef DECRTSCM\_H

#define DECRTSCM\_H

#include "Decoder.h"

class CComMMP;

class CComBMP;

///Декодер РТСЦМ

class CDecoderRtscm : public CDecoder

{

public:

CDecoderRtscm(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db);

~CDecoderRtscm();

virtual void init();

virtual void reset();

virtual void resetCounters();

virtual void loadComFromIni(CIniFile\*, CIniFile\*, QStringList&, QStringList&);

protected:

///состояние синхронизации

int m\_stkadr;

int m\_ssk;

int m\_kss;

int m\_allWords;

unsigned m\_cntSWErrors;

CComMMP\* m\_mmp;

//CComBMP\* m\_bmp;

CCommutator\* m\_mas[11];

virtual bool loadSpecialProps(CIniFileSection\*){return true;};

virtual bool saveSpecialProps(CIniFileSection\*){return true;};

virtual void scanTMI();

bool isMarker(unsigned short);

void createErrContainers();

void createExtraContainers();

virtual void processKadr();

void calcExtraPars();

unsigned calcCRCMMP();

};

#endif

## Листинг 2. Пишем файл RTSCM.cpp для системы телеметрических измерений РТСЦМ

#include "RTSCM.h"

#include "ComRTSCM.h"

#include "Parameter.h"

#include "Ini/FastIni.h"

unsigned short crc16( unsigned \*pcBlock, unsigned len )

{

    unsigned short crc = 0xFFFF;

    unsigned char i;

    while( len-- ) {

        crc ^= \*pcBlock++ << 8;

        for( i = 0; i < 8; i++ )

            crc = crc & 0x8000 ? ( crc << 1 ) ^ 0x1021 : crc << 1;

    }

    unsigned char\* bptr = (unsigned char\*)&crc;

    return (bptr[0] << 8) | bptr[1];

}

CDecoderRtscm::CDecoderRtscm(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db)

    : CDecoder(a\_base, a\_stream, \_db, PORT\_RTSCM, 640)

{

    reset();

    m\_mmp = dynamic\_cast<CComMMP\*>(createChildCommutator("ММП",COMTYPE\_RTSCM\_MMP));

    if (m\_mmp){

        m\_mas[0] = m\_mmp->createChildCommutator("МАС1",COMTYPE\_RTSCM\_MAS);

        m\_mas[1] = m\_mmp->createChildCommutator("БАСУ1",COMTYPE\_RTSCM\_MAS);

        m\_mas[2] = m\_mmp->createChildCommutator("МАС3",COMTYPE\_RTSCM\_MAS);

        m\_mas[3] = m\_mmp->createChildCommutator("МАС4",COMTYPE\_RTSCM\_MAS);

        m\_mas[4] = m\_mmp->createChildCommutator("МАС5",COMTYPE\_RTSCM\_MAS);

        m\_mas[5] = m\_mmp->createChildCommutator("БАСУ2",COMTYPE\_RTSCM\_MAS);

        m\_mas[6] = m\_mmp->createChildCommutator("МАС71",COMTYPE\_RTSCM\_MAS);//ps1

        m\_mas[7] = m\_mmp->createChildCommutator("МАС72",COMTYPE\_RTSCM\_MAS);//ps1

        m\_mas[8] = m\_mmp->createChildCommutator("МАС81",COMTYPE\_RTSCM\_MAS);

        m\_mas[9] = m\_mmp->createChildCommutator("МАС82",COMTYPE\_RTSCM\_MAS);

        m\_mas[10] = m\_mmp->createChildCommutator("МАС9",COMTYPE\_RTSCM\_MAS);//ps2

    }

    createChildCommutator("БМП",COMTYPE\_RTSCM\_BMP);

}

CDecoderRtscm::~CDecoderRtscm()

{

};

void CDecoderRtscm::reset()

{

    m\_kss = 0;

    m\_ssk = 0;

    m\_stkadr = 0;//синхронизация еще не поймана

    m\_allWords = 0;

    m\_cntSWErrors = 0;

}

void CDecoderRtscm::resetCounters()

{

    m\_mtx.lock();

    m\_cntSWErrors = 0;

    m\_mtx.unlock();

}

void CDecoderRtscm::init()

{

    loadBase();

    createServContainers();

    reset();

    createErrContainers();

    createExtraContainers();

}

void CDecoderRtscm::loadComFromIni(CIniFile\* a\_iniCom, CIniFile\* a\_iniPar,

    QStringList& allComSections, QStringList& allParSections)

{

    if (m\_vComs.size()==2){//т.е. они должны быть всегда

        for (int i=0;i<11;++i){

            m\_mas[i]->loadComFromIni(a\_iniCom, a\_iniPar,    allComSections, allParSections);

        }

        m\_vComs[1]->loadComFromIni(a\_iniCom, a\_iniPar,  allComSections, allParSections);//БМП

    }

}

bool CDecoderRtscm::isMarker(unsigned short w)

{

    switch (m\_stkadr)

    {

        case 0:

            if ( (w & 0xff)==0x1A )

                m\_stkadr = 1;

            break;

        case 1:

            if ( (w & 0xff)==0xCF )

                m\_stkadr = 2;

            else

                m\_stkadr = 0;

            break;

        case 2:

            if ( (w & 0xff)==0xFC )

                m\_stkadr = 3;

            else

                m\_stkadr = 0;

            break;

        case 3:

            if ( (w & 0xff)==0x1D ) {

                m\_stkadr = 0;

                if (m\_curLevel!=0)

                    m\_cntSWErrors++;

                m\_curLevel = 0;

                return true;

            }

            else

                m\_stkadr = 0;

            break;

        default:

            m\_stkadr=0;

            break;

    }

    return false;

}

void CDecoderRtscm::scanTMI()

{

    for(unsigned short i = 0; i < m\_sizebuf; ++i) {

        calcRawData(m\_localBuffer[i]);

        if(extractServInf(m\_localBuffer[i]))//ВИшка дальше в обработку не идет

            continue;

        //начальную службу не смотрим

        if (m\_stationTime.isCorrect())

            pushLevel((unsigned)(m\_localBuffer[i]&0xFF), m\_stationTime.getPseudoTime());

        bool wasMarker = isMarker(m\_localBuffer[i]);

        if((m\_curLevel==0)&&(wasMarker==false)) {

            m\_cntSWErrors++;

        }

        m\_allWords++;

        //пока просто каждые 2000 слов проверяем четность,

        //желательно это делать раз в секунду(независимо от информативности)

        if (m\_allWords>20000){

            unsigned ttime = m\_stationTime.getPseudoTime();

            CDataPoint tpu(ttime, m\_cntSWErrors);

            m\_errPars[0]->pushPoint(tpu); //всего сбоев МФр

            m\_allWords = 0;

        }

    }//for i

}

void    CDecoderRtscm::createErrContainers()

{

    m\_mtx.lock();

    CParameter\* pPar = new CParameter(this,"сбоев CC", m\_stream);

    pPar->setLimitsMaxMin(0, 10);

    m\_errPars.push\_back(pPar);

    m\_mtx.unlock();

}

void    CDecoderRtscm::createExtraContainers()

{

    m\_mtx.lock();

    CParameter\* pPar = new CParameter(this,"Синхрослово", m\_stream);

    pPar->setLimitsMaxMin(0x1ACFFC1D, 0x1ACFFC1D);

    m\_extraPars.push\_back(pPar);

    pPar = new CParameter(this,"КСС", m\_stream);

    pPar->setLimitsMaxMin(0,255);

    m\_extraPars.push\_back(pPar);

    pPar = new CParameter(this,"ССK", m\_stream);

    pPar->setLimitsMaxMin(0,127);

    m\_extraPars.push\_back(pPar);

    pPar = new CParameter(this,"CRC ММП", m\_stream);

    pPar->setLimitsMaxMin(1,1);

    m\_extraPars.push\_back(pPar);

    pPar = new CParameter(this,"Время", m\_stream);

    pPar->setLimitsMaxMin(0, 65535);//хотя мксимум 64 минуты

    pPar->setBitCount(16);

    m\_extraPars.push\_back(pPar);

    m\_mtx.unlock();

}

void CDecoderRtscm::calcExtraPars()

{   //Синхрослово

    CDataPoint tp( m\_stationTime.getPseudoTime(),

        ((m\_levels[636]&0xFF)<<24) + ((m\_levels[637]&0xFF)<<16)

        + ((m\_levels[638]&0xFF)<<8) + (m\_levels[639]&0xFF) );

    m\_extraPars[0]->pushPoint(tp);

    //КСС

    tp.vUns = m\_kss = m\_levels[0]&0xFF;

    m\_extraPars[1]->pushPoint(tp);

    //ССK

    tp.vUns = m\_ssk = m\_levels[1]&0xFF;

    m\_extraPars[2]->pushPoint(tp);

    tp.vUns = calcCRCMMP();

    m\_extraPars[3]->pushPoint(tp);

    //Время

    tp.vUns = (m\_levels[1]&0x3F) + (m\_levels[2]&0x3F)\*60 ;

    m\_extraPars[4]->pushPoint(tp);

}

unsigned    CDecoderRtscm::calcCRCMMP()

{

    unsigned short tCrc = crc16(&m\_levels[4], 256);

    unsigned short curCrc = m\_levels[260]+(m\_levels[261]<<8);

    if (tCrc==curCrc)

        return 1;

    return 0;

}

void CDecoderRtscm::processKadr()

{

    calcExtraPars();

    if (m\_mmp) {

        m\_mmp->pushData(&m\_levels[4], m\_kss, m\_ssk, m\_curPseudoTime);

    }

}

## Листинг 3. Пишем заголовочный файл Orbita.h

#ifndef DECORBITA\_H

#define DECORBITA\_H

#include "Decoder.h"

class QVideoEncoder;

///Декодер Орбита

class CDecoderOrbita : public CDecoder

{

public:

CDecoderOrbita(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& a\_dataBuf, int a\_portType = PORT\_ORBITA, int a\_okLen=16384);

~CDecoderOrbita();

virtual void init();

virtual void reset();

virtual void loadComFromIni(CIniFile\*, CIniFile\*, QStringList&, QStringList&);

int lengthFrame() const { return m\_lengthFrame;};

void setLengthFrame(int);

void pushLevel(unsigned, unsigned);

virtual void resetCounters();

protected:

/// наличие синхронизации по маркеру группы

bool m\_isMGr;

/// номер фразы(64), но не совсем, на самом деле их 128, но маркер только в 1 3 5...

int m\_NPhrase;

/// номер группы

int m\_NMGroup;

/// текущая позиция в кадре

int cntbuf;

/// наличие синхронизации по маркеру фразы

bool m\_isMOK;

/// длина кадра, включая служебную часть информативность = lf\*8192 потока

int m\_lengthFrame;

/// m\_lengthframe/2 чтоб не считать постоянно

int m\_lengthFrame2;

/// m\_lengthframe\*512 чтоб не считать постоянно

int m\_lengthFrame512;

/// Сбои по маркеру фразы

int m\_iBadMPhr;

/// Сбои по маркеру группы

int m\_iBadMGroup;

int m\_idatacnt;

/// текущее значение слова СЛЖ

unsigned char m\_meanSI;

/// массив слов СЛЖ в группе

unsigned char m\_SI[16];

/// номер цикла

int m\_NMCycle;

/// позиция маркера во фразе

int m\_MarkerPos;

///будем мерять период следования маркера групп

int groupPeriod;

///счетчик всех сбоев по МФр

unsigned m\_cntMPhErrors;

virtual bool loadSpecialProps(CIniFileSection\*);

virtual bool saveSpecialProps(CIniFileSection\*);

virtual void scanTMI();

bool isMarker(unsigned short);

virtual void calculateData();

virtual void processKadr();

void createErrContainers();

void createExtraContainers();

void calcBrtTime();

unsigned m\_last32bits;

unsigned m\_levelsLocal[32];

};

///Декодер Орбита16

class CDecoderOrbita16 : public CDecoderOrbita

{

public:

CDecoderOrbita16(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db);

~CDecoderOrbita16(){};

virtual void loadComFromIni(CIniFile\*, CIniFile\*, QStringList&, QStringList&);

protected:

virtual void calculateData();

virtual bool saveSpecialProps(CIniFileSection\*);

};

///Декодер БСВК

class CDecoderOrbita16BSVK : public CDecoderOrbita

{

Q\_OBJECT

public:

CDecoderOrbita16BSVK(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db);

~CDecoderOrbita16BSVK(){};

virtual void reset();

void reset2();

virtual void getData(int a\_camNum, QPixmap& a\_pic, unsigned& a\_time, int& a\_cnt, int& a\_sn );

///Генерация файлов mpeg

virtual bool getMpeg(const QString& datStr,const QString& mpegPath, QProgressBar \* pb, bool& stopped, bool autoLevels=false,int br=0,int cont=0);

///Генерация Jpeg последовательностей

virtual bool getJpegSeq(const QString& datStr,const QString& jpegPath, QProgressBar \* pb, bool& stopped, bool autoLevels=false,int br=0,int cont=0);

protected:

virtual void calculateData();

virtual bool saveSpecialProps(CIniFileSection\*){return true;};

private:

unsigned char m\_bsvkBuff[46];

unsigned char m\_HRKheaderFFD8[14];

int m\_lastCnt;

unsigned char m\_epilogue[16];

unsigned char m\_jpegBody[50000];

int m\_jpegSizeCnt;

bool m\_mpegMode;

QString m\_mpegPaths[4];

QVideoEncoder\* m\_encoder[4];

bool m\_jpegSeqMode;

QString m\_jpegSeqPaths[4];

bool m\_autoLevels;

int m\_brightness;

int m\_contrast;

QPixmap m\_tempPixmap;

/// 0-ждем header HRKontur, 1-начинается jpeg

int m\_state;

QPixmap m\_camImg[4];

unsigned m\_camTime[4];

int m\_camFrameCnt[4];

int m\_camSN[4];

};

#endif

## Листинг 4. Пишем файл Orbita.cpp для обработки и декодирования данных передоваемых в рамках “Орбита”

python manage.py startapp polls#include <QImageReader>

#include <QFileInfo>

#include <QDir>

#include <QApplication>

#include <QProgressBar>

#include <QImageWriter>

#include "Orbita.h"

#include "Parameter.h"

#include "Ini/FastIni.h"

#include "Service/Exif.h"

#include "ffmpeg/QTFFmpegWrapper/QVideoEncoder.h"

void processImage(const QPixmap& tPixmap, bool autoLevels,int tBright,int tCont, QPixmap& outPixmap);

CDecoderOrbita::CDecoderOrbita(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db, int a\_portType, int a\_okLen)

: CDecoder(a\_base, a\_stream, \_db, a\_portType, a\_okLen)

{

m\_bitCnt = 12;

m\_last32bits = 0;

}

CDecoderOrbita::~CDecoderOrbita()

{

}

void CDecoderOrbita::init()

{

if(!loadBase()) {//тогда умолчания

setLengthFrame(16);

}

//Заполним служебные контейнеры

createServContainers();

createErrContainers();

createExtraContainers();//доп служ параметры связанные с конкретной бртс

reset();

}

void CDecoderOrbita::loadComFromIni(CIniFile\* a\_iniCom, CIniFile\* a\_iniPar,

QStringList& allComSections, QStringList& allParSections)

{

CCommutator::loadComFromIni(a\_iniCom, a\_iniPar, allComSections, allParSections);

linkClbChannels(); //подлинковать калибровки которые на чужих коммутаторах

}

void CDecoderOrbita::reset()

{

m\_isMGr = false;

m\_NPhrase = 0; // номер фразы(64), но не совсем, на самом деле их 128, но маркер только в 1 3 5...

m\_NMGroup = 0; // номер группы

cntbuf = 0; // текущая позиция в кадре

m\_isMOK = false; // наличие синхронизации по маркеру фразы

m\_meanSI = 0; // текущее значение слова СЛЖ

memset(m\_SI,0,16); // массив слов СЛЖ в группе

m\_NMCycle = 0; // номер цикла

groupPeriod = 0; // будем мерять период следования маркера групп

m\_iBadMPhr = 0;

m\_iBadMGroup = 0;

m\_idatacnt = 0;

m\_last32bits = 0;

m\_cntMPhErrors = 0;

}

void CDecoderOrbita::resetCounters()

{

m\_mtx.lock();

m\_cntMPhErrors = 0;

m\_mtx.unlock();

}

void CDecoderOrbita::setLengthFrame(int lf)

{

if (m\_port!=PORT\_ORBITA)

return;

m\_lengthFrame = lf;

m\_lengthFrame2 = m\_lengthFrame/2; //чтобы в scandata не считать

m\_lengthFrame512 = m\_lengthFrame\*512; //чтобы в scandata не считать

m\_MarkerPos = 0;

//переразмерить буфер если надо

setCntChannels(2048 \*m\_lengthFrame);//один цикл

m\_dTime = 2500./m\_kadrLen; //m\_kadrLen=2048 \*m\_lengthframe

}

bool CDecoderOrbita::loadSpecialProps(CIniFileSection\* a\_commonSect)

{

if ((m\_port==PORT\_ORBITA16)||(m\_port==PORT\_BSVK)) {

m\_lengthFrame=32;

m\_lengthFrame2=16;

m\_lengthFrame512 = 16384;

m\_MarkerPos = 31;

}

else {

setLengthFrame(a\_commonSect->getInt("frameLength"));

}

return true;

}

bool CDecoderOrbita::saveSpecialProps(CIniFileSection\* a\_commonSect)

{

if (!a\_commonSect)

return false;

a\_commonSect->setValue("frameLength",QString::number(m\_lengthFrame));

return true;

}

void CDecoderOrbita::calcBrtTime()

{

if (m\_SI[1]==1) {

unsigned brtTime = ((m\_SI[2]>>4)&0xF)\*1000 + (m\_SI[2]&0xF)\*100 +

((m\_SI[3]>>4)&0xF)\*10 + (m\_SI[3]&0xF);

CDataPoint tp(m\_stationTime.getPseudoTime(), brtTime);

m\_extraPars[0]->pushPoint(tp);

double timeMPK = ( ((((unsigned short)m\_SI[4] )&0xFE)<<6)+((m\_SI[5]&0xFE)>>1) )/16384.;

CDataPoint tpMPK(m\_stationTime.getPseudoTime(), timeMPK);

m\_extraPars[1]->pushPoint(tpMPK);

}

}

void CDecoderOrbita::scanTMI()

{

for(unsigned short i = 0; i < m\_sizebuf; ++i) {

calcRawData(m\_localBuffer[i]);

if (extractServInf(m\_localBuffer[i])) //ВИшка дальше в обработку не идет(маска 0xB000)

continue;

m\_last32bits = ( (m\_last32bits<<1) | ((m\_localBuffer[i] >> 11)&1) );

if (cntbuf == m\_lengthFrame2) { //m\_lengthframe2 = m\_lengthframe/2 начало четной фразы, вычисляем Служебную из 12-го бита

m\_meanSI = m\_meanSI << 1;

m\_meanSI = m\_meanSI | (m\_localBuffer[i] >> 11);

if (((m\_NPhrase + 1) & 7) == 0)

m\_SI[(m\_NPhrase + 1) >> 3] = m\_meanSI;

}

if (cntbuf == m\_lengthFrame) { // Обрабатываем конец кадра

cntbuf = 0;

calculateData();

m\_NPhrase = ((m\_NPhrase + 1) & 0x3F); //%64

if(!m\_NPhrase)

m\_NMGroup = ((m\_NMGroup + 1) & 0x1F); //%32

switch(m\_meanSI){

case 0x72:

if (m\_isMOK) {

if (((m\_NPhrase == 0) && m\_isMGr) || (!m\_isMGr)) {

m\_isMGr = true;

m\_NPhrase = 0;

m\_meanSI = 0;

groupPeriod = 0;

calcBrtTime();

if (m\_SI[1] == 1) //маркер кадра (раз в секунду)

m\_NMCycle = 0;

}

}

if (m\_NPhrase) {

if (groupPeriod > (m\_lengthFrame<<6))//\*64 , также пропускаем возможные лишние маркеры внутри группы

m\_isMGr = false;

}

break;

case 0x8d: // маркер цикла

if(m\_NPhrase == 0) {

m\_NMGroup = 0;

m\_meanSI = 0;

groupPeriod = 0;

m\_NMCycle = ((m\_NMCycle+1)&3) ;//%4

setStartKadr();

}

break;

}

if ((m\_NPhrase==0)&&(!m\_isMGr)){

++m\_iBadMGroup;

}

}//(cntbuf == m\_lengthframe)

if ((cntbuf == m\_MarkerPos) || (!m\_isMOK)) { // если синхронизация потеряна, ищем очередной маркер

if (!isMarker(m\_localBuffer[i])) {

if (cntbuf == m\_MarkerPos) {

++m\_iBadMPhr;

++m\_cntMPhErrors;

if (!m\_NPhrase)

++m\_iBadMGroup;

}

}

}

m\_levelsLocal[cntbuf] = m\_localBuffer[i] & 0xFFF;

cntbuf++;

groupPeriod++;

m\_idatacnt++; //для подсчета статистики по сбоям

if (m\_idatacnt > m\_lengthFrame512) {

unsigned ttime = m\_stationTime.getPseudoTime();

double d = m\_iBadMPhr \* 100.0\*m\_lengthFrame / m\_idatacnt;

if (d>100)

d = 100;

CDataPoint tp(ttime, d);

m\_errPars[0]->pushPoint(tp); //сбои МФр

d = m\_iBadMGroup \* 100.0 \* m\_lengthFrame \* 32 / m\_idatacnt;

if (d > 100)

d = 100;

tp.v = d;

m\_errPars[1]->pushPoint(tp); //сбои МГр

CDataPoint tpu(ttime, m\_cntMPhErrors);

m\_errPars[2]->pushPoint(tpu); //всего сбоев МФр

m\_iBadMPhr = 0;

m\_iBadMGroup = 0;

m\_idatacnt = 0;

}

}//for i

}

bool CDecoderOrbita::isMarker(unsigned short uiData)

{

m\_isMOK = false;

if (m\_lengthFrame==32){//на m16 не смотрим на аппаратный маркер

if ((m\_last32bits&0xAAAA2AAA)==0x2A8020A2) {

cntbuf = m\_MarkerPos;

m\_isMOK = true;

}

}

else {

if (uiData & 0x4000) {

cntbuf = m\_MarkerPos;

m\_isMOK = true;

}

}

return m\_isMOK;

}

void CDecoderOrbita::createErrContainers()

{

m\_mtx.lock();

CParameter\* pPar = new CParameter(this,"% сбоев МФр", m\_stream);

pPar->setLimitsMaxMin(-1.,0.1);

pPar->setViewMinMax(-1, 101);

m\_errPars.push\_back(pPar);

pPar = new CParameter(this,"% сбоев МГр", m\_stream);

pPar->setLimitsMaxMin(-1.,0.1);

pPar->setViewMinMax(-1, 101);

m\_errPars.push\_back(pPar);

pPar = new CParameter(this,"сбоев МФр", m\_stream);

pPar->setLimitsMaxMin(0,10);

m\_errPars.push\_back(pPar);

m\_mtx.unlock();

}

void CDecoderOrbita::createExtraContainers()

{

m\_mtx.lock();

CParameter\* pPar = new CParameter(this,"Бортовое время", m\_stream);

pPar->setLimitsMaxMin(0,65565);

m\_extraPars.push\_back(pPar);

pPar = new CParameter(this,"МПК", m\_stream);

pPar->setLimitsMaxMin(-1.,65565.); //чтоб не краснело

m\_extraPars.push\_back(pPar);

m\_mtx.unlock();

}

void CDecoderOrbita::calculateData()

{

if (!m\_stationTime.isCorrect())

return;

unsigned ttime = m\_stationTime.getPseudoTime();

for (int i = 0; i < m\_lengthFrame; ++i)

pushLevel(m\_levelsLocal[i],ttime);

}

void CDecoderOrbita::pushLevel(unsigned a\_level, unsigned ps\_time)

{ //уже и так в мьютексе

m\_curPseudoTime = ps\_time;

if (m\_levels)

m\_levels[m\_curLevel] = a\_level;

m\_curLevel++;

//обработку цикла разделим на 8 периодов, а то 4 раза в секунду - уж очень редкое обновление

if ( (m\_curLevel & ((m\_kadrLen>>3)-1) )== 0 ) {//один из 8-ми периодов

if (m\_levels)

processKadr();

if (m\_curLevel>=m\_kadrLen)

m\_curLevel=0;

}

}

void CDecoderOrbita::processKadr()//1 цикл/8

{

if (m\_curPseudoTime<2500/8)

return;

int startAddr = m\_curLevel - (m\_kadrLen>>3);

int endAddr = m\_curLevel-1;

QVector<CCommutator\*>::iterator i = m\_vComs.begin();

while(i!=m\_vComs.end()){

if (\*i){

const int\* tUsedCycles=0;

const QVector<int>& tAddresses = (\*i)->getOrbInternalAddresses(tUsedCycles);

QVector<int>::const\_iterator j = tAddresses.constBegin();

while (j!=tAddresses.constEnd()){

if (((\*j)>=startAddr)&&((\*j)<=endAddr))

(\*i)->pushLevel(m\_levels[\*j], m\_curPseudoTime + ((\*j)- startAddr)\*m\_dTime -2500/8.);//-2500

if ((\*j)>endAddr)

break;

++j;

}

}

++i;

}

QVector<CParameter\*>::iterator k = m\_vPars.begin();

CDataPoint tp(m\_curPseudoTime, 0.0);

while(k!=m\_vPars.end()) {

if (\*k){

const int\* tUsedCycles = 0; // make sense for 1 2 Hz pars

const QVector<int>& tAddresses = (\*k)->getOrbInternalAddresses(tUsedCycles);

if (tUsedCycles==0)

break; //ситуация невероятная, но на всякий случай

QVector<int>::const\_iterator j = tAddresses.constBegin();

while (j!=tAddresses.constEnd()){

if ( ((\*j)>=startAddr) && ((\*j)<=endAddr) && (tUsedCycles[m\_NMCycle])) {

tp.t = m\_curPseudoTime/\* + ((\*j)- startAddr)\*m\_dTime - 2500/8.\*/;//-2500//вообще конечно и тут нужно корректировать время

tp.v = m\_levels[\*j];

(\*k)->pushPoint(tp);

}

if ((\*j)>endAddr)

break;

++j;

}

}

++k;

}

}

//--------------------------------------------------------------------------------------------------

CDecoderOrbita16::CDecoderOrbita16(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db)

: CDecoderOrbita(a\_base, a\_stream, \_db,PORT\_ORBITA16, 32)

{

createChildCommutator("П1",COMTYPE\_POTOKM16,(void\*)&m\_NMCycle);

createChildCommutator("П2",COMTYPE\_POTOKM16,(void\*)&m\_NMCycle);

m\_lengthFrame=32;

m\_lengthFrame2=16;

m\_lengthFrame512 = 16384;

m\_MarkerPos = 31;

}

void CDecoderOrbita16::loadComFromIni(CIniFile\* a\_iniCom, CIniFile\* a\_iniPar,

QStringList& allComSections, QStringList& allParSections)

{

if (m\_vComs.size()==2){//т.е. они должны быть всегда

m\_vComs[0]->loadComFromIni(a\_iniCom, a\_iniPar, allComSections, allParSections);

m\_vComs[1]->loadComFromIni(a\_iniCom, a\_iniPar, allComSections, allParSections);

}

}

void CDecoderOrbita16::calculateData()

{ //разделение П1 П2

if (!m\_stationTime.isCorrect())

return;

unsigned ttime = m\_stationTime.getPseudoTime();

for (int i = 0; i < m\_lengthFrame; ++i)

m\_vComs[i&1]->pushLevel(m\_levelsLocal[i], ttime);

}

bool CDecoderOrbita16::saveSpecialProps(CIniFileSection\*)

{

return true;

}

//--------------------------------------------------------------------------------------------------

CDecoderOrbita16BSVK::CDecoderOrbita16BSVK(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db)

: CDecoderOrbita(a\_base, a\_stream, \_db,PORT\_BSVK, 32),

m\_jpegSeqMode(false),

m\_mpegMode(false)

{

m\_encoder[0]=m\_encoder[1]=m\_encoder[2]=m\_encoder[3]=0;

m\_lengthFrame=32;

m\_lengthFrame2=16;

m\_lengthFrame512 = 16384;

m\_MarkerPos = 31;

reset2();

}

bool CDecoderOrbita16BSVK::getMpeg(const QString& datStr,const QString& mpegPath, QProgressBar\* pb, bool& stopped, bool autoLevels,int br,int cont)

{

if (!QFile::exists(datStr)) {//path

return false;

}

int width=640;

int height=480;

int bitrate=5000000;

int gop = 2; //gop: maximal interval in frames between keyframes

int fps = 10;

QByteArray srcBA = datStr.toLocal8Bit();

FILE\* fs = fopen(srcBA.data(), "rb");

if (!fs) {

return false;

}

QFileInfo tFI(mpegPath);

for (int i = 0; i < 4; ++i){

m\_mpegPaths[i] = QDir::toNativeSeparators(tFI.absoluteDir().absolutePath() +"\\"+ tFI.baseName() + QString("\_%1.avi").arg(i));

m\_encoder[i] = new QVideoEncoder();

m\_encoder[i]->createFile(m\_mpegPaths[i],width,height,bitrate,gop,fps); // Fixed frame rate

//else

// encoder.createFile(filename,width,height,bitrate\*1000/fps,gop,1000); // For variable frame rates: set the time base to e.g. 1ms (1000fps),

// and correct the bitrate according to the expected average frame rate (fps)

}

QFileInfo inputFI(datStr);

quint64 tSz = inputFI.size();

quint64 tCurPos = 0;

m\_mpegMode = true;

m\_autoLevels=autoLevels;

m\_brightness = br;

m\_contrast = cont;

fseek(fs,512,SEEK\_SET);

tCurPos = 512;

unsigned short buf[8192];

quint64 tCnt = 0;

while ((!stopped)) {

int wr = fread(buf, 2, 8192, fs);

if (wr < 1)

break;

m\_dataBuff.push(buf, wr);

newData();//processing and write pics to mp4

tCurPos+=(2\*wr);

if ( pb && (tCnt&0x7F)==0 ) {

pb->setValue(((tCurPos+1)\*100)/tSz);

QApplication::processEvents();

}

++tCnt;

}

fclose(fs);

m\_mpegMode = false;

for (int i = 0; i < 4; ++i) {

m\_encoder[i]->close();

delete m\_encoder[i];

m\_encoder[i] = 0;

}

if (pb) {

pb->setValue(100);

QApplication::processEvents();

}

return true;

}

bool CDecoderOrbita16BSVK::getJpegSeq(const QString& datStr,const QString& jpegPath, QProgressBar\* pb, bool& stopped, bool autoLevels,int br,int cont)

{

if (!QFile::exists(datStr)) {

return false;

}

QFileInfo tFI(jpegPath);

QDir tDir;

if (!QFile::exists(jpegPath)) {

//если отсутствует каталог, то все равно его создаем

tDir.mkpath(tFI.absoluteFilePath());

}

m\_jpegSeqMode = true;

m\_autoLevels=autoLevels;

m\_brightness = br;

m\_contrast = cont;

m\_jpegSeqPaths[0] = QDir::toNativeSeparators(tFI.absoluteFilePath()+"\\0");

tDir.mkpath(m\_jpegSeqPaths[0]);

m\_jpegSeqPaths[1] = QDir::toNativeSeparators(tFI.absoluteFilePath()+"\\1");

tDir.mkpath(m\_jpegSeqPaths[1]);

m\_jpegSeqPaths[2] = QDir::toNativeSeparators(tFI.absoluteFilePath()+"\\2");

tDir.mkpath(m\_jpegSeqPaths[2]);

m\_jpegSeqPaths[3] = QDir::toNativeSeparators(tFI.absoluteFilePath()+"\\3");

tDir.mkpath(m\_jpegSeqPaths[3]);

QFileInfo inputFI(datStr);

quint64 tSz = inputFI.size();

quint64 tCurPos = 0;

QByteArray srcBA = datStr.toLocal8Bit();

FILE\* fs = fopen(srcBA.data(), "rb");

if (!fs) {

m\_jpegSeqMode = false;

return false;

}

fseek(fs,512,SEEK\_SET);

tCurPos = 512;

unsigned short buf[8192];

quint64 tCnt=0;

while ((!stopped)){

int wr = fread(buf,2,8192,fs);

if (wr<1)

break;

m\_dataBuff.push(buf, wr);

newData();//processing and write pics

tCurPos+=(2\*wr);

if ( pb && (tCnt&0x7F)==0 )

{

pb->setValue(((tCurPos+1)\*100)/tSz);

QApplication::processEvents();

}

++tCnt;

}

fclose(fs);

m\_jpegSeqMode = false;

if (pb) {

pb->setValue(100);

QApplication::processEvents();

}

return true;

}

//int \_GCnt = 0;

void CDecoderOrbita16BSVK::calculateData()

{

int nFirstWInRS = ((m\_NMGroup&3)\*2944) + m\_NPhrase\*46 ;

if (!m\_stationTime.isCorrect())

return;

unsigned ttime = m\_stationTime.getRealTime();

m\_bsvkBuff[0] = ((m\_levelsLocal[0]>>3)&0xFF) ;

m\_bsvkBuff[1] = ((m\_levelsLocal[0]&0x7)<<5) + ((m\_levelsLocal[1]>>7)&0x1f) ;

m\_bsvkBuff[2] = ((m\_levelsLocal[1]&0x7f)<<1) + ((m\_levelsLocal[2]>>10)&1) ;

m\_bsvkBuff[3] = ((m\_levelsLocal[2]>>2)&0xFF) ;

m\_bsvkBuff[4] = ((m\_levelsLocal[2]&0x3)<<6) + ((m\_levelsLocal[3]>>6)&0x3F) ;

m\_bsvkBuff[5] = ((m\_levelsLocal[3]&0x3f)<<2) + (( m\_levelsLocal[4]>>9)&0x3) ;

m\_bsvkBuff[6] = ((m\_levelsLocal[4]>>1)&0xFF) ;

m\_bsvkBuff[7] = ((m\_levelsLocal[4]&1)<<7) + ((m\_levelsLocal[5]>>5)&0x7F) ;

m\_bsvkBuff[8] = ((m\_levelsLocal[5]&0x1F)<<3) + ((m\_levelsLocal[6]>>8)&0x7) ;

m\_bsvkBuff[9] = (m\_levelsLocal[6]&0xFF);

m\_bsvkBuff[10] = ((m\_levelsLocal[7]>>4)&0xFF);

m\_bsvkBuff[11] = ((m\_levelsLocal[7]&0xF)<<4) + ((m\_levelsLocal[8]>>7)&0xF);

m\_bsvkBuff[12] = ((m\_levelsLocal[8]&0x7F)<<1) + ((m\_levelsLocal[9]>>11)&1);

m\_bsvkBuff[13] = ((m\_levelsLocal[9]>>3)&0xFF);

m\_bsvkBuff[14] = ((m\_levelsLocal[9]&0x7)<<5) + ((m\_levelsLocal[10]>>6)&0x1F);

m\_bsvkBuff[15] = ((m\_levelsLocal[10]&0x3f)<<2) + ((m\_levelsLocal[11]>>10)&0x3);

m\_bsvkBuff[16] = ((m\_levelsLocal[11]>>2)&0xFF);

m\_bsvkBuff[17] = ((m\_levelsLocal[11]&3)<<6) + ((m\_levelsLocal[12]>>5)&0x3F);

m\_bsvkBuff[18] = ((m\_levelsLocal[12]&0x1F)<<3) + ((m\_levelsLocal[13]>>9)&7);

m\_bsvkBuff[19] = ((m\_levelsLocal[13]>>1)&0xFF);

m\_bsvkBuff[20] = ((m\_levelsLocal[13]&1)<<7) + ((m\_levelsLocal[14]>>4)&0x7F);

m\_bsvkBuff[21] = ((m\_levelsLocal[14]&0xF)<<4) + ((m\_levelsLocal[15]>>8)&0xF);

m\_bsvkBuff[22] = (m\_levelsLocal[15]&0xFF);

m\_bsvkBuff[23] = ((m\_levelsLocal[16]>>3)&0xFF) ;

m\_bsvkBuff[24] = ((m\_levelsLocal[16]&0x7)<<5) + ((m\_levelsLocal[17]>>7)&0x1f) ;

m\_bsvkBuff[25] = ((m\_levelsLocal[17]&0x7f)<<1) + ((m\_levelsLocal[18]>>10)&1) ;

m\_bsvkBuff[26] = ((m\_levelsLocal[18]>>2)&0xFF) ;

m\_bsvkBuff[27] = ((m\_levelsLocal[18]&0x3)<<6) + ((m\_levelsLocal[19]>>6)&0x3F) ;

m\_bsvkBuff[28] = ((m\_levelsLocal[19]&0x3f)<<2) + (( m\_levelsLocal[20]>>9)&0x3) ;

m\_bsvkBuff[29] = ((m\_levelsLocal[20]>>1)&0xFF) ;

m\_bsvkBuff[30] = ((m\_levelsLocal[20]&1)<<7) + ((m\_levelsLocal[21]>>5)&0x7F) ;

m\_bsvkBuff[31] = ((m\_levelsLocal[21]&0x1F)<<3) + ((m\_levelsLocal[22]>>8)&0x7) ;

m\_bsvkBuff[32] = (m\_levelsLocal[22]&0xFF);

m\_bsvkBuff[33] = ((m\_levelsLocal[23]>>4)&0xFF);

m\_bsvkBuff[34] = ((m\_levelsLocal[23]&0xF)<<4) + ((m\_levelsLocal[24]>>7)&0xF);

m\_bsvkBuff[35] = ((m\_levelsLocal[24]&0x7F)<<1) + ((m\_levelsLocal[25]>>11)&1);

m\_bsvkBuff[36] = ((m\_levelsLocal[25]>>3)&0xFF);

m\_bsvkBuff[37] = ((m\_levelsLocal[25]&0x7)<<5) + ((m\_levelsLocal[26]>>6)&0x1F);

m\_bsvkBuff[38] = ((m\_levelsLocal[26]&0x3f)<<2) + ((m\_levelsLocal[27]>>10)&0x3);

m\_bsvkBuff[39] = ((m\_levelsLocal[27]>>2)&0xFF);

m\_bsvkBuff[40] = ((m\_levelsLocal[27]&3)<<6) + ((m\_levelsLocal[28]>>5)&0x3F);

m\_bsvkBuff[41] = ((m\_levelsLocal[28]&0x1F)<<3) + ((m\_levelsLocal[29]>>9)&7);

m\_bsvkBuff[42] = ((m\_levelsLocal[29]>>1)&0xFF);

m\_bsvkBuff[43] = ((m\_levelsLocal[29]&1)<<7) + ((m\_levelsLocal[30]>>4)&0x7F);

m\_bsvkBuff[44] = ((m\_levelsLocal[30]&0xF)<<4) + ((m\_levelsLocal[31]>>8)&0xF);

m\_bsvkBuff[45] = (m\_levelsLocal[31]&0xFF);

//получили 46 б видео

for(int i = 0; i < 46; ++i){

nFirstWInRS = ((m\_NMGroup&3)\*2944) + m\_NPhrase\*46 + i;

if (nFirstWInRS>11519)

{

continue;

}

switch (m\_state){

case 0://ждем HRKontur

//m\_HRKheader

memmove(m\_HRKheaderFFD8, &m\_HRKheaderFFD8[1],13);

m\_HRKheaderFFD8[13]=m\_bsvkBuff[i];

if ((m\_HRKheaderFFD8[0]==0x48)&&(m\_HRKheaderFFD8[1]==0x52)&&(m\_HRKheaderFFD8[2]==0x4B)&&(m\_HRKheaderFFD8[3]==0x6F)

&&(m\_HRKheaderFFD8[4]==0x6E)&&(m\_HRKheaderFFD8[5]==0x74)&&(m\_HRKheaderFFD8[6]==0x75)&&(m\_HRKheaderFFD8[7]==0x72)

&&(m\_HRKheaderFFD8[12]==0xFF)&&(m\_HRKheaderFFD8[13]==0xD8)){//заголовок есть

m\_lastCnt = m\_HRKheaderFFD8[11];

m\_state = 1;

m\_jpegBody[0]=0xFF;

m\_jpegBody[1]=0xD8;

m\_jpegSizeCnt = 2;

}

break;

case 1://накапливаем jpeg и ждем FFD9, но не более 50000 байт

m\_jpegBody[m\_jpegSizeCnt++]=m\_bsvkBuff[i];

memmove(m\_epilogue, &m\_epilogue[1],15);

m\_epilogue[15]=m\_bsvkBuff[i];

if ((m\_epilogue[0]==0x4B)&&(m\_epilogue[1]==0x6F)&&(m\_epilogue[2]==0x6E)&&(m\_epilogue[3]==0x74)

&&(m\_epilogue[4]==0x75)&&(m\_epilogue[5]==0x72)&&(m\_epilogue[6]==0x46)&&(m\_epilogue[7]==0x52)){//эпилог есть

int tCnt = m\_epilogue[11];

if (tCnt==m\_lastCnt){//счетчик корректный

//проверим маркер конца и длину

unsigned tLen = m\_epilogue[14]\*256 + m\_epilogue[15];

if ((tLen>10)&&(m\_jpegBody[tLen-2]==0xFF)&&(m\_jpegBody[tLen-1]==0xD9)) {//jpeg корректный

m\_state = 0;

Cexif exif;

int tCamNum = exif.getUserComment1(m\_jpegBody, tLen);

int tCamNum2 = (tCamNum & 0x3);

m\_camImg[tCamNum2].loadFromData(m\_jpegBody,tLen);

m\_camTime[tCamNum2] = ttime;

m\_camFrameCnt[tCamNum2] = tCnt;

m\_camSN[tCamNum2] = tCamNum;

if (m\_jpegSeqMode) { //write image to file

QString newPath = QDir::toNativeSeparators(m\_jpegSeqPaths[tCamNum2] + "//" + QString("%1s\_%2.jpg").arg(ttime/10000, 6, 10, QLatin1Char('0')).arg(tCnt));

QByteArray dstBA = newPath.toLocal8Bit();

FILE\* f1 = fopen(dstBA.data(), "wb");

if ((m\_autoLevels == false)&&(m\_brightness==0)&&(m\_contrast==0)) {

fwrite(m\_jpegBody,tLen,1,f1);

}

else {

processImage(m\_camImg[tCamNum2],m\_autoLevels,m\_brightness,m\_contrast,m\_tempPixmap);

QImageWriter tWriter(newPath,"JPG");

tWriter.setQuality(90);

tWriter.write(m\_tempPixmap.toImage());

}

fclose(f1);

}

else if(m\_mpegMode) //generate mpeg file

{

if (m\_encoder[tCamNum2]) {

if ((m\_autoLevels == false)&&(m\_brightness==0)&&(m\_contrast==0)) {

m\_encoder[tCamNum2]->encodeImage(m\_camImg[tCamNum2].toImage());

}

else {

processImage(m\_camImg[tCamNum2],m\_autoLevels,m\_brightness,m\_contrast,m\_tempPixmap);

m\_encoder[tCamNum2]->encodeImage(m\_tempPixmap.toImage());

}

}

}

else {

switch(tCamNum2) {

case 0:

emit newVideoData1();

break;

case 1:

emit newVideoData2();

break;

case 2:

emit newVideoData3();

break;

case 3:

emit newVideoData4();

break;

}

}

}

}

}

if (m\_jpegSizeCnt>50000){//превышен лимит, переходим в ожидание

m\_jpegSizeCnt=0;

m\_state = 0;

}

break;

default:

break;

}

}

}

void CDecoderOrbita16BSVK::reset()

{

CDecoderOrbita::reset();

reset2();

}

void CDecoderOrbita16BSVK::reset2()

{

memset(m\_bsvkBuff,0,46);

memset(m\_HRKheaderFFD8,0,14);

m\_state=0;

m\_lastCnt = -1;

memset(m\_jpegBody, 0, 50000);

m\_jpegSizeCnt = 0;

memset(m\_epilogue, 0, 16);

m\_autoLevels = false;

m\_brightness = 0;

m\_contrast = 0;

}

void CDecoderOrbita16BSVK::getData(int a\_camNum, QPixmap& a\_pic, unsigned& a\_time, int& a\_cnt, int& a\_sn)

{

m\_mtx.lock();

a\_pic = m\_camImg[a\_camNum];

a\_time = m\_camTime[a\_camNum];

a\_cnt = m\_camFrameCnt[a\_camNum];

a\_sn = m\_camSN[a\_camNum];

m\_mtx.unlock();

}

## Листинг 5. Пишем заголовочный файл decCCSDS.h

#ifndef DECCCSDS\_H

#define DECCCSDS\_H

#include "Decoder.h"

///Декодер CCSDS

class CDecCCSDS : public CDecoder

{

public:

CDecCCSDS(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db);

~CDecCCSDS();

virtual void init();

virtual void reset();

virtual void resetCounters();

virtual void pushLevel(unsigned a\_level, unsigned ps\_time);

protected:

///состояние синхронизации

//int m\_stkadr;

///длина инф части кадра

int m\_realLength;

///источник в заголовке CCSDS

int m\_nSource;

int m\_cnt;

bool m\_validHeader;

bool m\_validCRC;

unsigned m\_cntErrors;

unsigned m\_crcErrorsCnt;

unsigned m\_packetsCnt;

int m\_curHL;//0 1, текущее полуслово

virtual void scanTMI();

bool isMarker(unsigned short);

void createErrContainers();

void createExtraContainers();

virtual bool loadSpecialProps(CIniFileSection\*){return true;};

virtual bool saveSpecialProps(CIniFileSection\*){return true;};

void calcExtraPars();

private:

unsigned m\_markWord;

int m\_nWrdFromLastMarker;//считаем слова от последнего аппаратного маркера

};

#endif

## Листинг 6. Создаём файл decCCSDS.cpp для принятия и декодирования CCSDS-пакетов ТМИ.

#include "DecCCSDS.h"

#include "Parameter.h"

#include "Ini/FastIni.h"

const int G\_MaxLenCCSDS=1600;

CDecCCSDS::CDecCCSDS(const QString& a\_base, const CStream\* a\_stream, CDataBuffer& \_db)

: CDecoder(a\_base, a\_stream, \_db, PORT\_CCSDS, G\_MaxLenCCSDS)

{

reset();

}

CDecCCSDS::~CDecCCSDS()

{

};

void CDecCCSDS::reset()

{

m\_realLength = G\_MaxLenCCSDS-112;

m\_nSource = 0;

m\_cnt = 0;

m\_validHeader = false;

m\_validCRC = false;

m\_crcErrorsCnt = 0;

m\_packetsCnt = 0;

m\_cntErrors = 0;

m\_markWord = 0;

m\_nWrdFromLastMarker = -1;

m\_curHL = 0;

}

void CDecCCSDS::resetCounters()

{

m\_crcErrorsCnt = 0;

m\_packetsCnt = 0;

m\_cntErrors = 0;

}

void CDecCCSDS::init()

{ // загрузим базу

if(!loadBase()) { //тогда умолчания

}

//Заполним служебные контейнеры

createServContainers();

setCntChannels(G\_MaxLenCCSDS);

createErrContainers();

createExtraContainers();

}

bool CDecCCSDS::isMarker(unsigned short w)

{

//reimpl

m\_markWord = (m\_markWord<<4)+(w&0xf) ;

if (((m\_markWord&0xFFFFFF)==0xFAF320)&&

((m\_nWrdFromLastMarker==1)||(m\_nWrdFromLastMarker==2)))

{

m\_levels[0]=0xFA;

m\_levels[1]=0xF3;

m\_levels[2]=0x20;

m\_curLevel = 2;

m\_curHL = 0;

return true;

}

return false;

}

void CDecCCSDS::pushLevel(unsigned a\_level, unsigned ps\_time)

{

//newData в декодере уже закрыла мьютекс

m\_curPseudoTime = ps\_time;

if (!m\_levels)

return;

//reimpl

if (m\_curHL==0) {

m\_levels[m\_curLevel] = (m\_levels[m\_curLevel]&0xF0) + (a\_level&0xF);

m\_curHL = ((m\_curHL+1)&1);

++m\_curLevel;

}

else {

m\_levels[m\_curLevel] = (m\_levels[m\_curLevel]&0xF) + ((a\_level&0xF)<<4);

m\_curHL = ((m\_curHL+1)&1);

return;

}

if (m\_curLevel>=m\_kadrLen) {

/\* unsigned tNull = 0;

fwrite(&tNull,4,1,m\_log);

fwrite(&tNull,4,1,m\_log);

fwrite(&tNull,4,1,m\_log);

fwrite(&tNull,4,1,m\_log);

for (int i=0;i<m\_kadrLen;++i) {

fwrite(&m\_levels[i],2,1, m\_log);

}\*/

m\_curLevel=0;

processKadr();

}

}

void CDecCCSDS::scanTMI()

{

for(unsigned short i = 0; i < m\_sizebuf; ++i) {

calcRawData(m\_localBuffer[i]);

if(extractServInf(m\_localBuffer[i]))//ВИшка дальше в обработку не идет

continue;

if (m\_localBuffer[i]&0x4000) {

m\_nWrdFromLastMarker = 0;

}

else if (m\_nWrdFromLastMarker!=-1){

++m\_nWrdFromLastMarker;

}

for (int j = 8; j > -1; j-=4)

{

isMarker((m\_localBuffer[i]>>j)&0xF);

if (m\_stationTime.isCorrect())

pushLevel((unsigned)((m\_localBuffer[i]>>j)&0xF), m\_stationTime.getPseudoTime());

if ((m\_curHL==0)&&(m\_curLevel==8)) { // попробуем обработать заголовок

if ((m\_levels[0]==0xFA)&&(m\_levels[1]==0xF3)&&(m\_levels[2]==0x20)) {//корректное синхрослово

m\_validHeader = true;

m\_realLength = m\_levels[6] + ((m\_levels[5]&7)<<8);

if (m\_realLength > G\_MaxLenCCSDS-112) {

m\_realLength = G\_MaxLenCCSDS-112;

m\_validHeader = false;

}

m\_nSource = ((m\_levels[5]>>4)&7);

}

}

if ((m\_curHL==0)&&(m\_curLevel == 12 + m\_realLength)) {//возможно собрался полный кадр

m\_validCRC = false;

if (m\_validHeader){//проверим CRC

++m\_packetsCnt;//считаем общее кол-во пакетов с корректным crc

unsigned calcCrc=0;

for (int i = 3; i < 8 + m\_realLength; ++i)

calcCrc += m\_levels[i];

unsigned calcCrc2 =

m\_levels[8 + m\_realLength] + (m\_levels[9 + m\_realLength]<<8) +

(m\_levels[10 + m\_realLength]<<16)+(m\_levels[11 + m\_realLength]<<24);

m\_validCRC = (calcCrc2==calcCrc);

}

if (!m\_validCRC)

++m\_crcErrorsCnt;

else{

for(int c = m\_curLevel; c < G\_MaxLenCCSDS; ++c) {

m\_levels[c] = 0;

}

}

CDataPoint tp(m\_stationTime.getPseudoTime(), m\_crcErrorsCnt);

m\_errPars[0]->pushPoint(tp); //счетчик ошибок crc

if (m\_packetsCnt) {

double tPrcCntErrors = ((m\_crcErrorsCnt\*100.)/m\_packetsCnt);

if (tPrcCntErrors>100.)

tPrcCntErrors = 100.;

CDataPoint tp2(m\_stationTime.getPseudoTime(), tPrcCntErrors);

m\_errPars[1]->pushPoint(tp2); //счетчик % ошибок crc

}

int tCnt = m\_levels[7];

if (((m\_cnt+1)&0xFF)!=tCnt)

++m\_cntErrors;

m\_cnt = tCnt;

tp.vUns = m\_cntErrors;

m\_errPars[2]->pushPoint(tp); //ошибок счетчика

calcExtraPars();

}

}

}//for i

}

void CDecCCSDS::calcExtraPars()

{ //Счетчик

unsigned tPT = m\_stationTime.getPseudoTime();

CDataPoint tp(tPT , m\_levels[7] );

m\_extraPars[1]->pushPoint(tp);

//Счетчик всего пакетов

tp.vUns = m\_packetsCnt;

m\_extraPars[2]->pushPoint(tp);

//Синхрослово

tp.vUns = (m\_levels[2]&0xFF) + ((m\_levels[1]&0xFF)<<8) + ((m\_levels[0]&0xFF)<<16);

m\_extraPars[0]->pushPoint(tp);

}

void CDecCCSDS::createErrContainers()

{

m\_mtx.lock();

CParameter\* pPar = new CParameter(this,"сбоев CRC", m\_stream);

pPar->setLimitsMaxMin(0,10);

m\_errPars.push\_back(pPar);

pPar = new CParameter(this,"% сбоев CRC", m\_stream);

pPar->setLimitsMaxMin(-1.,1.);

m\_errPars.push\_back(pPar);

pPar = new CParameter(this,"сбоев счетчика", m\_stream);//[2]

pPar->setLimitsMaxMin(0,10);

m\_errPars.push\_back(pPar);

m\_mtx.unlock();

}

void CDecCCSDS::createExtraContainers()

{

m\_mtx.lock();

CParameter\* pPar = new CParameter(this,"Синхрослово", m\_stream);

pPar->setLimitsMaxMin(0xFAF320, 0xFAF320);

pPar->setVisibleBase(VB\_HEX);

m\_extraPars.push\_back(pPar);

pPar = new CParameter(this,"Счетчик", m\_stream);

pPar->setLimitsMaxMin(0, 255);

m\_extraPars.push\_back(pPar);

pPar = new CParameter(this,"Всего пакетов", m\_stream);

pPar->setLimitsMaxMin(0, 0x7FFFFFFF);

m\_extraPars.push\_back(pPar);

m\_mtx.unlock();

}

## Приложение 2

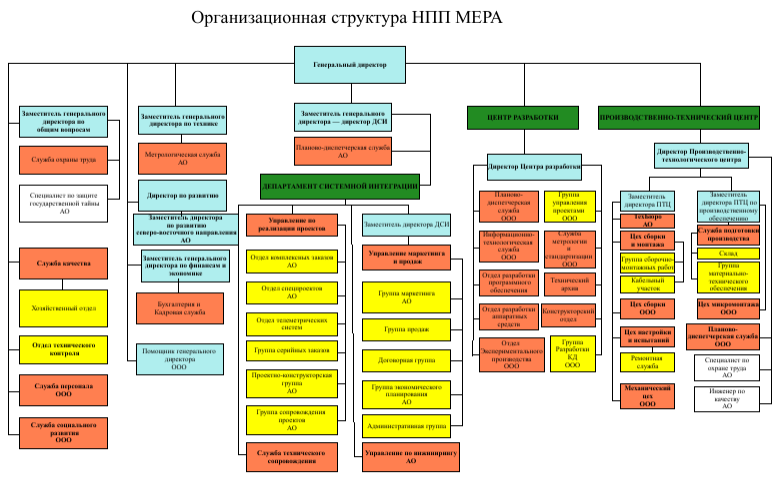


Рис. 6.1. Организационная структура НПП МЕРА.