

**课 程 实 验 报 告**

**课程名称： 射频识别技术原理及应用**

**专业班级： 物联网1501**

**学 号： U201514868**

**姓 名： 徐臻林**

**指导教师： 甘早斌**

**报告日期： 2018.6.22**

**计算机科学与技术学院**

目录

[第一次实验内容:低频读写器实验 3](#_Toc517836437)

[一、实验目的 3](#_Toc517836438)

[二、实验内容 3](#_Toc517836439)

[三、实验要求 3](#_Toc517836440)

[四、实验内容及结果 4](#_Toc517836441)

[五、实验体会与总结 5](#_Toc517836442)

[六、开发实例源码 5](#_Toc517836443)

[第二次实验内容：高频读写器实验(ISO14443A) 13](#_Toc517836444)

[一、实验目的 13](#_Toc517836445)

[二、实验内容 13](#_Toc517836446)

[三、实验要求 13](#_Toc517836447)

[四、实验内容及结果 14](#_Toc517836448)

[五、 实验体会与总结 16](#_Toc517836449)

[第三次实验内容：高频读写器实验(ISO15693) 17](#_Toc517836450)

[一、实验目的 17](#_Toc517836451)

[二、实验内容 17](#_Toc517836452)

[三、实验要求 17](#_Toc517836453)

[四、实验内容及结果 17](#_Toc517836454)

[五、实验体会与总结 24](#_Toc517836455)

[第四次实验内容：超高频读写器实验 25](#_Toc517836456)

[一、实验目的 25](#_Toc517836457)

[二、实验内容 25](#_Toc517836458)

[三、实验要求 25](#_Toc517836459)

[四、实验内容及结果 26](#_Toc517836460)

[五、实验体会与总结 32](#_Toc517836461)

[第五次实验内容：RFID综合应用实验 (ISO15693) 33](#_Toc517836462)

[一、实验目的 33](#_Toc517836463)

[二、实验内容 33](#_Toc517836464)

[三、实验要求 33](#_Toc517836465)

[四、实验内容及结果 34](#_Toc517836466)

[五、实验体会与总结 50](#_Toc517836467)

# 第一次实验内容:低频读写器实验

## 一、实验目的

1. 通过本次实验了解博创科技 RFID 读写器的结构组成，掌握试验箱的安装和 连接方法，学会使用该试验箱进行相关的实验。
2. 了解低频读写器的基本原理，学会如何使用低频读写器。学习和掌握在低频
3. 读写器的编程操作，对标签进行读操作，了解低频读写器的工作机理，并完成一

个示例程序。

## 二、实验内容

1、完成低频读写器的标签读取试验；

2、熟悉低频 LF-125K 开发实例，完成低频读写器编程实验，熟悉和了解低 频读写器 API 函数；分析应用 demo 存在的问题，进而完善应用 demo 的功能。 完善的功能包括：记录保存进出的历史记录。

## 三、实验要求

1、学会通过试验箱对低频标签的读取，通过反复循环读取十张低频电子标签，记录在读取过程中遇到的问题或发现的现象，并分析遇到的这些问题或现象 的原因；

2、掌握低频读写器 API 函数的调用方法，编写程序打开串口，建立连接， 读取标签信息；

3、将标签移进移出低频模块天线，能够记录标签读取的次数；

## 四、实验内容及结果

完成低频读写器的标签读取试验

依次将10张标签卡放在低频125KHz模块上进行读取。

熟悉低频 LF-125K 开发实例，完成低频读写器编程实验，熟悉和了解低频读写器 API 函数；分析应用 demo 存在的问题，进而完善应用 demo 的功能。 完善的功能包括：记录保存进出的历史记录。

在QT环境下运行实验平台所提供的代码，发现这个demo可以完成一些基本的操作，但是存在着不足：只能够记录一次操作的情况，以后的卡片刷新后会抹掉前面的记录。将demo的代码更改如下：

void MainWindow::readData()

{

if(serialPort->*bytesAvailable*() < 5)

return;

QByteArray data = serialPort->readAll();

if(m125dll->LF125K\_FrameAnalysis((uint8 \*)(data.data())) == 0)

{

QString tagId = CharStringtoHexString(tr(" "),data.data(),data.length());

QString time = CurrentDateTime();

int index = model->findRecord(tagId);

if(index >= 0 )

{

QString text = model->record(index).value(2).toString();

if(text == tr("进"))

model->addRecord(tagId,time,tr("出"));//修改updateRecord函数为addRecord函数

else

model->addRecord(tagId,time,tr("进"));//修改updateRecord函数为addRecord函数

}

else {

model->addRecord(tagId,time,tr("进"));

}

}

}

更改后发现刷卡的每一条状态都进行了记录，但是重复刷新一张卡后发现从第二次开始以后的状态都保持为“出”。将demo的代码更改如下：

/\*\*

\* @brief RecordTableModel::findRecord

\* @param tagId 卡号

\* @return 如果成功返回该卡所在的行,否则返回-1

\* 根据卡号查找记录

\*/

int RecordTableModel::findRecord(const QString &tagId)

{

int flag=-1;

for(int row=0; row<*rowCount*(); row++){

if(*data*(*index*(row, 0)).toString() == tagId)

flag=row;

}

return flag;

}

在代码中加入一个flag变量，保证每次返回的变量都是相关行的最新最新状态。

## 五、实验体会与总结

本次实验第一次接触到了射频识别技术的实验平台，将平时在课堂上学习到的理论知识运用到了实际的操作中。起初在老师和助教的引导下，了解了博创科技 RFID 读写器的结构组成，掌握试验箱的安装和连接方法，并且学会使用该试验箱进行了后续的相关实验。在此基础上，了解到了低频读写器的基本原理，学会如何使用低频读写器。学习和掌握在QT的编译环境下低频读写器的编程操作，对标签进行读操作，了解低频读写器的工作机理，并完成了相关的示例程序。 在进行第一个实验内容的实际操作中，发现了读写器读卡操作并不够稳定并且对卡与读写器之间距离的限制也很大，有的卡本身也存在问题无法正常读取；第二个实验内容在看懂实验平台所给的实验代码和实验平台的操作原理后，更改两处的代码后不难实现。

## 六、开发实例源码

**1.mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "recordtablemodel.h"

#include <QMessageBox>

#include <QDebug>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*作者: jianghj@up-tech.com

\*日期: 2016-09-30

\*描述: 125K演示程序主要代码,此处模拟的人员通道,进出需要刷卡,

\* 125K在实际应用中主要也是这个功能,比如小区的门禁卡.

\* 注意:人为主动刷卡,2.4G是被动刷卡

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->fillPortsParameters(ui->baudRateBox);//波特率填充

this->serialPort = new QSerialPort(this);

db = new Database(this);

model = new RecordTableModel(this);

ui->tableView->*setModel*(model);

ui->tableView->resizeColumnsToContents();

ui->tableView->horizontalHeader()->setStretchLastSection(true);

intValidator = new QIntValidator(0, 4000000,this);

ui->btn\_connect->setEnabled(true);

ui->btn\_refresh->setEnabled(true);

ui->btn\_disconnect->setEnabled(false);

this->on\_btn\_refresh\_clicked();

m125dll = new M125Dll();

//关联相关槽函数

connect(ui->baudRateBox, SIGNAL(currentIndexChanged(int)),this, SLOT(checkCustomBaudRatePolicy(int)));

connect(serialPort, SIGNAL(error(QSerialPort::SerialPortError)), this, SLOT(handleError(QSerialPort::SerialPortError)));

connect(serialPort, SIGNAL(readyRead()), this, SLOT(readData()));

}

MainWindow::~*MainWindow*()

{

model->submitAll();

delete model;

delete db;

delete m125dll;

delete intValidator;

delete serialPort;

delete ui;

}

//Baudrate parameter init

void MainWindow::fillPortsParameters(QComboBox \*box)

{

box->clear();

box->addItem(QStringLiteral("9600"), QSerialPort::Baud9600);

box->addItem(QStringLiteral("19200"), QSerialPort::Baud19200);

box->addItem(QStringLiteral("38400"), QSerialPort::Baud38400);

box->addItem(QStringLiteral("57600"), QSerialPort::Baud57600);

box->addItem(QStringLiteral("115200"), QSerialPort::Baud115200);

box->addItem(tr("Custom"));

}

/\*\*

\* @brief MainWindow::on\_btn\_connect\_clicked

\* 连接串口

\*/

void MainWindow::on\_btn\_connect\_clicked()

{

QString name = ui->serialNameBox->currentText();

QString baud = ui->baudRateBox->currentText().trimmed();

if(baud.isEmpty())

{

QMessageBox::critical(this, tr("Error"), "波特率输入错误！");

return ;

}

serialPort->setPortName(name);

serialPort->setBaudRate(baud.toInt(),QSerialPort::AllDirections);

if (serialPort->*open*(QIODevice::ReadWrite)) {

ui->btn\_connect->setEnabled(false);

ui->btn\_disconnect->setEnabled(true);

ui->btn\_refresh->setEnabled(false);

} else {

ui->btn\_connect->setEnabled(true);

ui->btn\_refresh->setEnabled(true);

ui->btn\_disconnect->setEnabled(false);

QMessageBox::warning(this,tr("提示"),tr("初始化%1失败！请检查串口是否已经被占用？").arg(name),QMessageBox::Yes);

}

}

/\*\*

\* @brief MainWindow::on\_btn\_disconnect\_clicked

\* 断开连接

\*/

void MainWindow::on\_btn\_disconnect\_clicked()

{

if(!serialPort->isOpen())

return ;

serialPort->*close*();

ui->btn\_connect->setEnabled(true);

ui->btn\_refresh->setEnabled(true);

ui->btn\_disconnect->setEnabled(false);

}

/\*\*

\* @brief MainWindow::on\_btn\_refresh\_clicked

\* 刷新按钮点击事件

\*/

void MainWindow::on\_btn\_refresh\_clicked()

{

QStringList list = getSerialName();

ui->serialNameBox->clear();

ui->serialNameBox->addItems(list);

}

/\*\*

\* @brief MainWindow::checkCustomBaudRatePolicy

\* @param idx combox被选中的索引值

\* 设置自定义波特率

\*/

void MainWindow::checkCustomBaudRatePolicy(int idx)

{

QComboBox \*box = dynamic\_cast<QComboBox\*>(QObject::sender());

bool isCustomBaudRate = !box->itemData(idx).isValid();

box->setEditable(isCustomBaudRate);

if (isCustomBaudRate) {

box->clearEditText();

box->setValidator(intValidator);

}

}

/\*\*

\* @brief MainWindow::readData

\* 读取串口数据

\*/

void MainWindow::readData()

{

if(serialPort->*bytesAvailable*() < 5)

return;

QByteArray data = serialPort->readAll();

if(m125dll->LF125K\_FrameAnalysis((uint8 \*)(data.data())) == 0)

{

QString tagId = CharStringtoHexString(tr(" "),data.data(),data.length());

QString time = CurrentDateTime();

int index = model->findRecord(tagId);

if(index >= 0 )

{

QString text = model->record(index).value(2).toString();

if(text == tr("进"))

model->addRecord(tagId,time,tr("出"));

else

model->addRecord(tagId,time,tr("进")); //将updateRecord改为addRecord，以显示出所有的记录

}

else {

model->addRecord(tagId,time,tr("进"));

}

}

}

/\*\*

\* @brief MainWindow::handleError

\* @param error SerialPortError枚举类,详细请看SerialPortError的定义

\* 处理错误信息

\*/

void MainWindow::handleError(QSerialPort::SerialPortError error)

{

if (error == QSerialPort::ResourceError) {

QMessageBox::critical(this, tr("Critical Error"), serialPort->errorString());

this->on\_btn\_disconnect\_clicked();

}

}

**2.recordtablemodel.cpp**

#include "recordtablemodel.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*作者: jianghj@up-tech.com

\*日期: 2016-09-20

\*描述: 记录表的model

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

RecordTableModel::RecordTableModel(QObject \*parent) : QSqlTableModel(parent)

{

tableName = "person\_Records";

header<<QObject::trUtf8("卡号")<< QObject::trUtf8("时间")<< QObject::trUtf8("状态");

if(!this->tableExist(tableName))

this->createTable();

this->*setTable*(tableName);

this->*select*();

}

/\*\*

\* @brief DBManager::tableExist

\* @param tableName 表名

\* @return 如果存在返回true，否则返回false

\* 用于判断表是否存在

\*/

bool RecordTableModel::tableExist(const QString &tableName)

{

int count = 0 ;

QString sqlText = QObject::tr("select count(\*) from sqlite\_master where type='table' and name='%1'").arg(tableName);

QSqlQuery query;

query.exec(sqlText);

if(query.next())

count = query.value(0).toInt();

if(count > 0)

return true;

else

return false;

}

/\*\*

\* @brief RecordTableModel::createTable

\* @param tableName 数据块表名称

\* 用于创建消费记录表

\*/

void RecordTableModel::createTable()

{

QSqlQuery query;

QString str;

str = tr("create table ") + tableName + tr(" ( ");

str += header.at(0) + tr(" varchar not null, ");

str += header.at(1) + tr(" varchar, ");

str += header.at(2) + tr(" varchar) ");

qDebug()<<"Sql: " << str.toUtf8().data();

bool ret = query.exec(str);

if(ret == true){

qDebug()<<tableName<<QObject::tr(" table create success");

}

else{

qDebug()<<tableName<<QObject::tr(" table create failed");

}

}

/\*\*

\* @brief RecordTableModel::findRecord

\* @param tagId 卡号

\* @return 如果成功返回该卡所在的行,否则返回-1

\* 根据卡号查找记录

\*/

int RecordTableModel::findRecord(const QString &tagId)

{

int flag=0;

for(int row=0; row<*rowCount*(); row++){

if(*data*(*index*(row, 0)).toString() == tagId)

flag = row;

}

return flag; //

}

/\*\*

\* @brief RecordTableModel::deleteByTagId

\* @param tagId 卡号

\* @return 如果成功返回true，否则false

\* 根据卡号删除记录

\*/

bool RecordTableModel::deleteByTagId(const QString &tagId)

{

for(int row=0; row<*rowCount*(); row++){

if(*data*(*index*(row, 0)).toString() == tagId)

removeRow(row);

}

return submitAll();

}

/\*\*

\* @brief RecordTableModel::updateRecord

\* @param row 待更新记录所在行

\* @param tagId 卡号

\* @param time 当前的时间

\* @param status 当前的状态

\* 更新记录

\*/

void RecordTableModel::updateRecord(int row,QString &tagId, QString &time, QString status)

{

QSqlRecord record;//也可以直接使用sql语句进行,但是sql语句更容易写错

record.append(QSqlField(header.at(0), QVariant::String));

record.append(QSqlField(header.at(1), QVariant::String));

record.append(QSqlField(header.at(2), QVariant::String));

record.setValue(0, QVariant(tagId));

*setData*(*index*(row,0),tagId);

*setData*(*index*(row,1),time);

*setData*(*index*(row,2),status);

this->insertRecord(row, record);

}

/\*\*

\* @brief RecordTableModel::addRecord

\* @param tagId 卡号

\* @param time 消费时间

\* @param addr 消费地址

\* @param readerId 读卡器ID

\* @param value 消费金额

\* @param remark 备注

\* @return 记录条数

\* 添加记录

\*/

int RecordTableModel::addRecord(QString &tagId, QString &time, QString status)

{

QSqlRecord record;//也可以直接使用sql语句进行,但是sql语句更容易写错

record.append(QSqlField(header.at(0), QVariant::String));

record.append(QSqlField(header.at(1), QVariant::String));

record.append(QSqlField(header.at(2), QVariant::String));

record.setValue(0, QVariant(tagId));

record.setValue(1, QVariant(time));

record.setValue(2, QVariant(status));

insertRecord(-1, record);

return *rowCount*();

}

# 第二次实验内容：高频读写器实验(ISO14443A)

## 一、实验目的

通过本次实验了解高频读写器的基本原理，学会如何使用高频读写器，掌握系统命令参数的意义和设置方式。

进一步加深对 S50 卡的存储结构和 ISO14443A 协议的理解。通过读写器试验箱，掌握对 S50 各个扇区数据的读写方法，并熟悉高频读写器 API 函数。

## 二、实验内容

1、完成 ISO14443A 协议下标签密钥修改实验；

2、完成 ISO14443A 协议下存取控制位修改实验 ；

3、完成 ISO14443A 协议下标签数据读写实验 ；

4、完成 ISO14443A 协议下标签加值减值实验；

5、熟悉和了解高频HF1356M 14443A开发实例，掌握高频读写器API函数， 并通过编程实验对 S50 卡的卡号读写功能。

## 三、实验要求

1、学会通过试验箱对 ISO14443A 协议下标签密钥进行修改；

2、学会通过试验箱对 ISO14443A 协议下标签存取控制位进行修改，并能够 通过存取控制位控制相应扇区的数据读写；

3、学会通过试验箱对 ISO14443A 协议下标签进行加值减值操作；

4、掌握高频读写器API函数的调用方法，并能够通过编程实现对ISO14443A 协议下标签数据的读写控制。

## 四、实验内容及结果

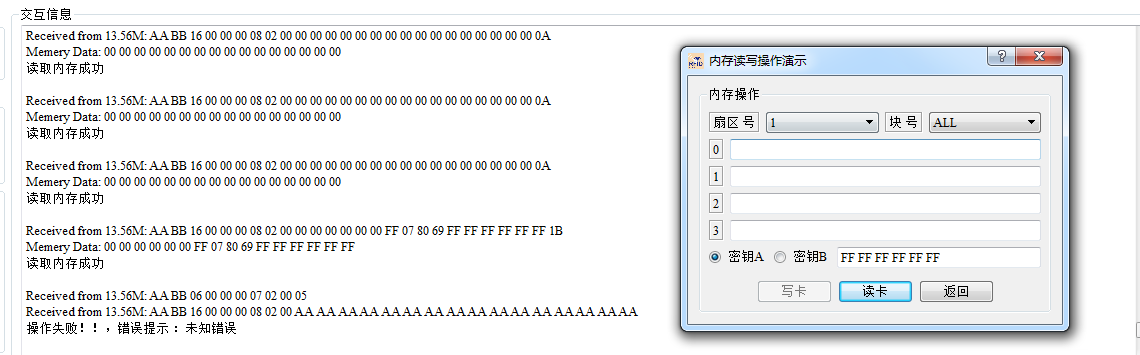
1.基本实验操作的实现

对高频读写器ISO14443A协议进行验证，首先正确连接线路板串口，打开RFID综合实训系统，切换到ISO13.56M模块，连接串口。可见交互信息中显示串口打开成功，并且通讯记录成功记录。



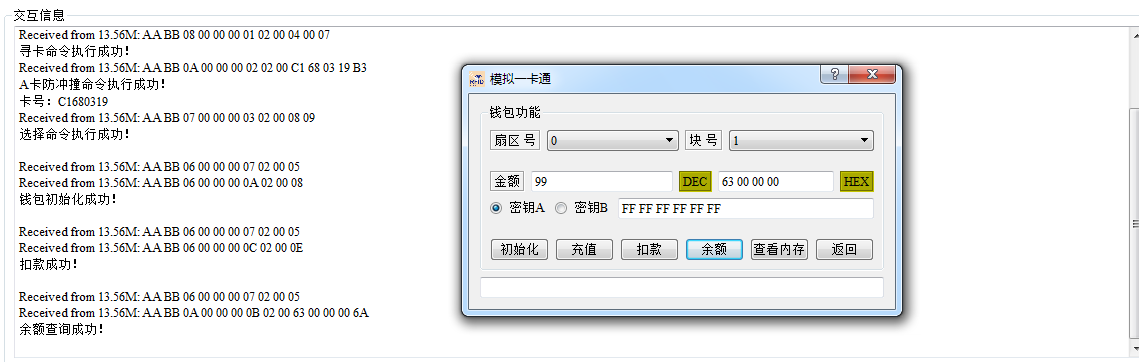
2.内存读写及修改实验

选中对应的卡号，点击内存读写按钮，就可以调出内存读写操作窗口，对块号3进行操作，输入密钥A（默认为FF FF FF FF FF FF）即可进行读写。



1. 一卡通的模拟实验

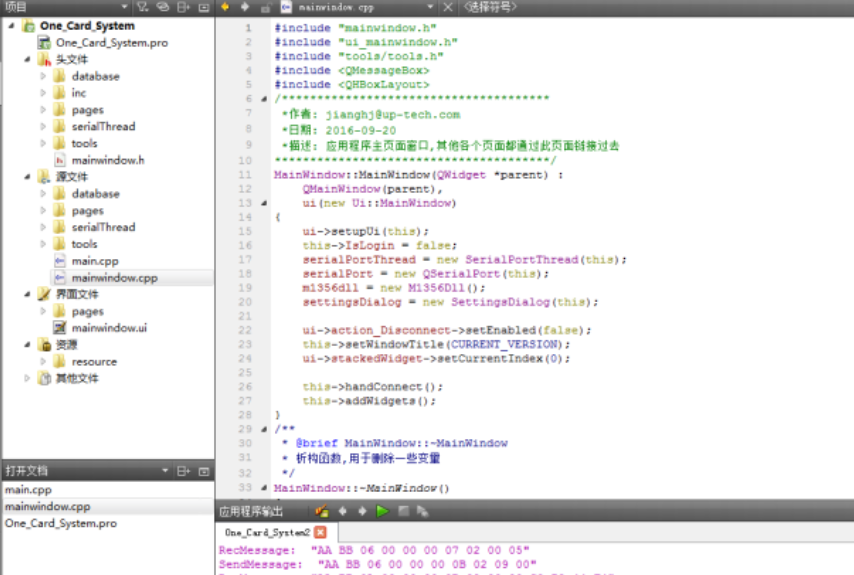
成功进行钱包的初始化，并且顺利完成扣款以及余额的查询操作



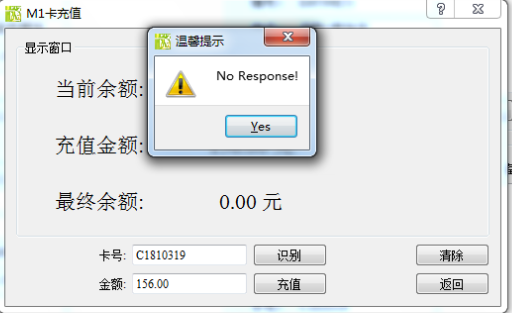
1. 在QT环境下模拟一卡通的消费、充值等操作

在模拟操作的过程中出现了一下情况，卡中的余额存在上限，若超出阈值会出现友情提醒，程序异常结束





若消费金额超出卡内余额，操作失败并弹出No Response！



## 五、 实验体会与总结

在完成第一个实验的基础上第二次实验上手起来还是比较快，在实验前的课堂上我们在理论知识方面上了解了Mifare1射频卡的存储结构、控制块的结构、数据块的存取控制以及控制块的存取控制。所以在基本的实验操作上进行还是比较顺利的，在操作过程也发现了一些临界状态，收获颇丰。

# 第三次实验内容：高频读写器实验(ISO15693)

## 一、实验目的

通过本次实验了解高频读写器的基本原理，学会如何使用高频读写器，掌握 系统命令参数的意义和设置方式。 进一步加深对ISO15693协议下标签的存储结构以及ISO15693协议的理解。 通过读写器试验箱，掌握对 ISO15693 协议下标签读写操作，并熟悉高频读写器 API 函数。

## 二、实验内容

1、完成 ISO15693 协议下标签的数据读写实验 ；

2、完成 ISO15693 协议下标签 AFI 的读写和块锁定实验 ；

3、完成 ISO15693 协议下标签 DSFID 的读写和块安全位的读取实验；

4、熟悉和了解高频 HF1356M 15693 开发实例，掌握高频读写器 API 函数， 并通过编程实现 ISO 15693 协议下标签的读写功能。

## 三、实验要求

1、学会通过试验箱对 ISO15693 协议下标签数据的读写；

2、理解和掌握应用族标识符（AFI）和数据存储格式标识符（DSFID）基本 概念和含义；

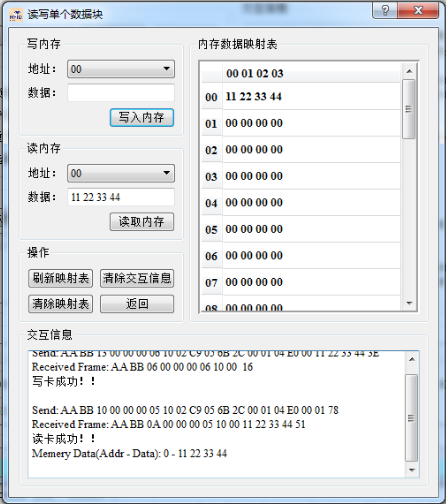
3、掌握高频读写器 API 函数的调用方法，并能够通过编程实现对 ISO15693 协议下标签数据的读写控制。

## 四、实验内容及结果

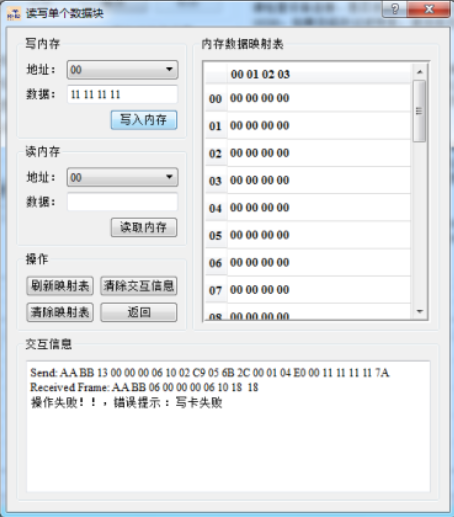
1.寻卡：寻卡命令为寻找在读写器感应范围内的15693标签，寻找到标签并读取标签ID然后通过串口发送给请求方。



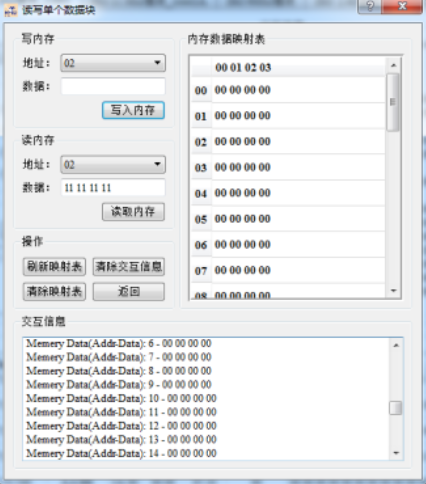
2.读写单个数据块：15693协议下支持读取标签的单个数据块。读取数据块是标签的存储器，因此首先也要寻卡之后进行操作，否则不知道要读取的是哪张卡。写单个块是用来写一个指定的标签的一个指定块。



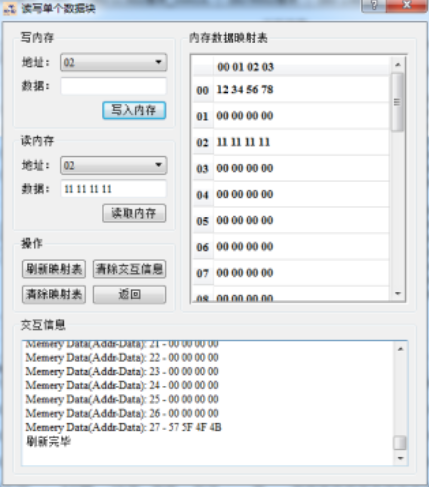




写卡操作提示失败，该卡的00地址在后面的锁定操作中被锁定



点击写入命令后自动清空输入框，此时点击内存按钮进行查看写入是否成功

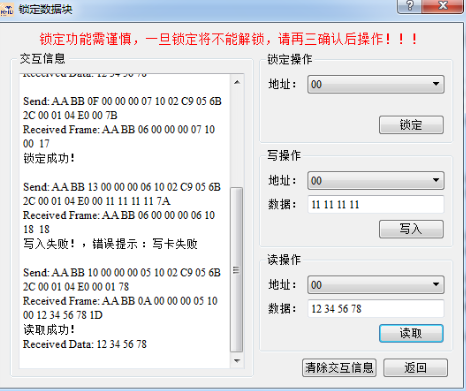


写入成功后会在操作信息窗口提示，同时也会将写入命令在此显示，同样会在原界面上的操作记录上添加Send行和Received行，各个字段和前面的意义意义。当写入了新的数据，再读取的时候会将写入的信息读取出来并显示的数据和内存映射表上。

3.读取多个数据块：下图所示为读取00到03结束的地址的数据，同时将发送信息和响应提示信息显示在对话框的交互信息窗口，在交互信息窗口也有详细显示各个地址对应的数据



4.锁定数据块：锁定了数据块之后该数据块不能被解锁，锁定之后该数据块不能再次写入，因此注意在实验中少锁几块



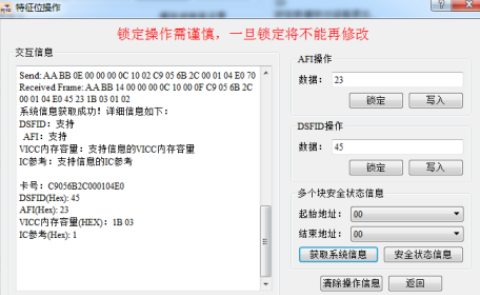
以00地址为例，首先写入再读取

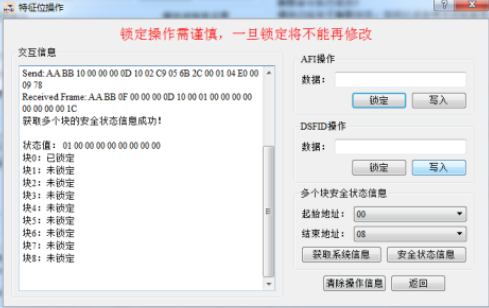


然后锁定数据块，再次写入就会报错



5.获取系统信息：获取系统信息主要是获取与标签相关的信息，请求命令字为Ox100C。请求与写DFSID一致。



1. 获取多个块安全状态：点击安全状态信息按钮可以获取多个块的安全信息状态信息，在按钮的上方选择地址区
2. 

7.波特率设置：波特率设置为 设置13.56MHz模块的串口波特率，14443和15693两种协议下设置的波特率均有效



1. 读书馆管理系统demo



模拟借书操作显示借书成功剩余本数少了一本



读书管理中剩余本数回到35本，表明还书成功

## 五、实验体会与总结

通过本次实验了解了高频读写器的基本原理，学会了如何使用高频读写器，掌握了系统命令参数的意义和设置方式。进一步加深了对ISO15693协议下标签的存储结构以及ISO15693协议的理解。 通过读写器试验箱，掌握了对 ISO15693 协议下标签读写操作，并熟悉了高频读写器 API 函数。

# 第四次实验内容：超高频读写器实验

## 一、实验目的

通过本次实验了解超高频读写器的基本原理，学会如何使用超高频读写器，掌握超高频读写器和标签参数的含义和设置方法。

进一步加深对 Gen2 协议下标签的存储结构以及 Gen2 协议的理解。通过读写器试验箱，掌握对 Gen2 协议下标签读写操作，并熟悉超高频读写器 API函数的调用。

## 二、实验内容

1、超高频读写器的基本认知；

2、完成 Gen2 协议下标签读写实验；

3、完成 Gen2 协议下标签锁实验；

4、完成 Gen2 协议下标签 TID 区分析实验；

5、完成 Gen2 协议下标签密钥修改实验；

6、熟悉和了解超高频 UHF-900M 开发实例，掌握超高频读写器 API函数，并通过编程实现 Gen2 协议下标签的读写功能。

## 三、实验要求

1、学会通过试验箱对 Gen2 协议下标签数据的读写；

2、理解和掌握 Gen2 协议下标签存储器结构的特点及含义；

3、掌握超高频读写器 API函数的调用方法，并能够通过编程实现对 Gen2 协议下标签数据的读写控制。

## 四、实验内容及结果

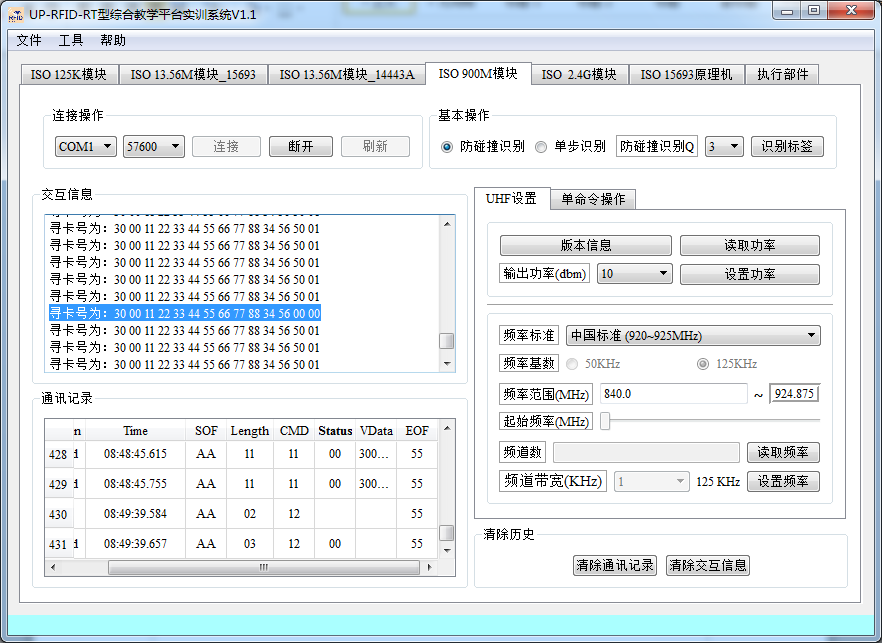
1.单步识别:连接串口后默认选择单步识别，此时点击识别标签可得到如图结果：



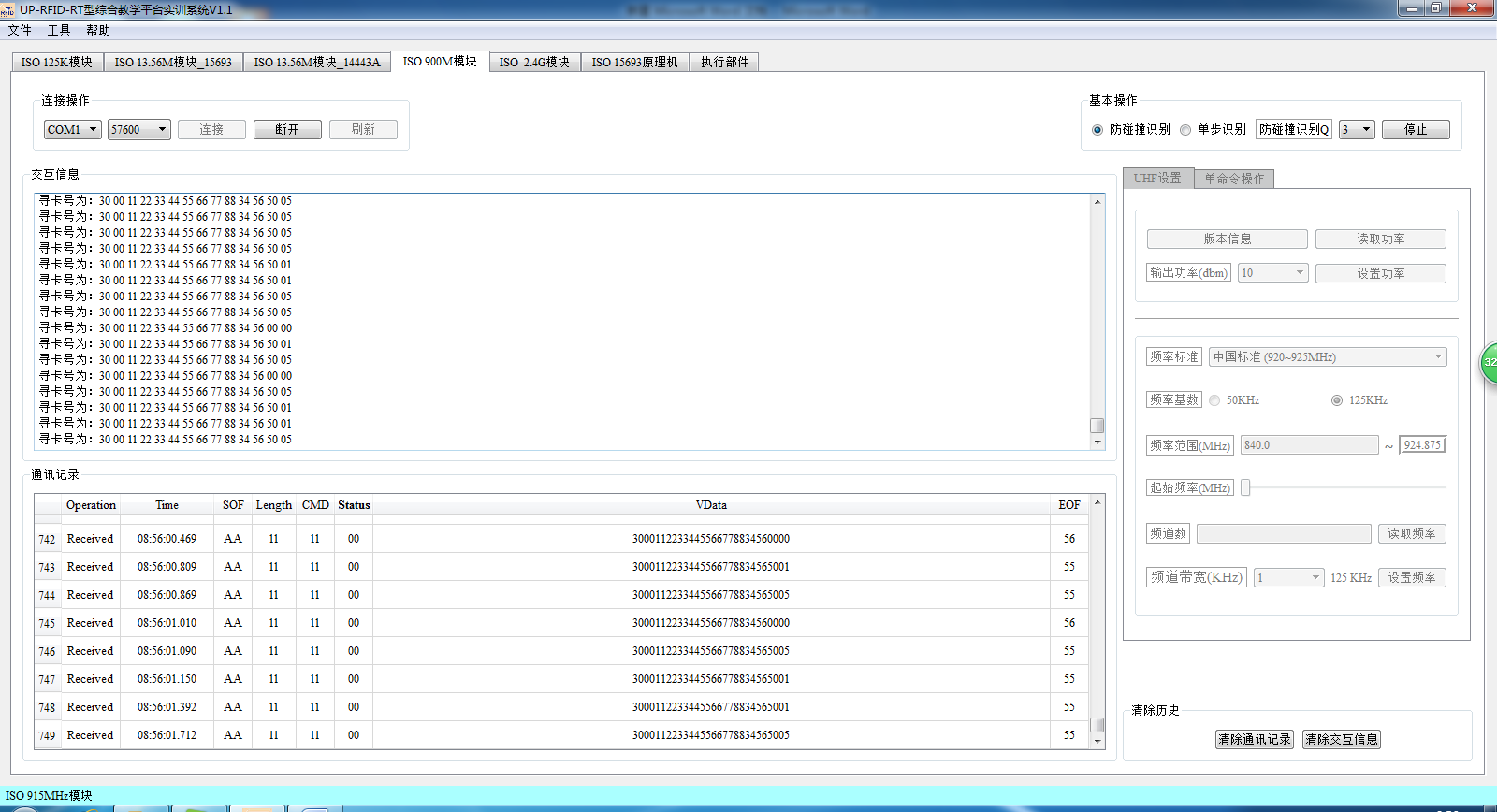
如图所示为寻卡成功的状态，在交互信息窗口显示当前的卡号、操作提示以及串口接收的数据信息，在通讯记录中记录了收发的命令，并按照900MHz模式的串口信息协议进行展示



2.防碰撞识别：防碰撞识别与单步识别类似，防碰撞识别时读卡器内部一直识别，这个和13.56MHz里的自动识别有本质的识别，该功能室读卡器本身的功能。

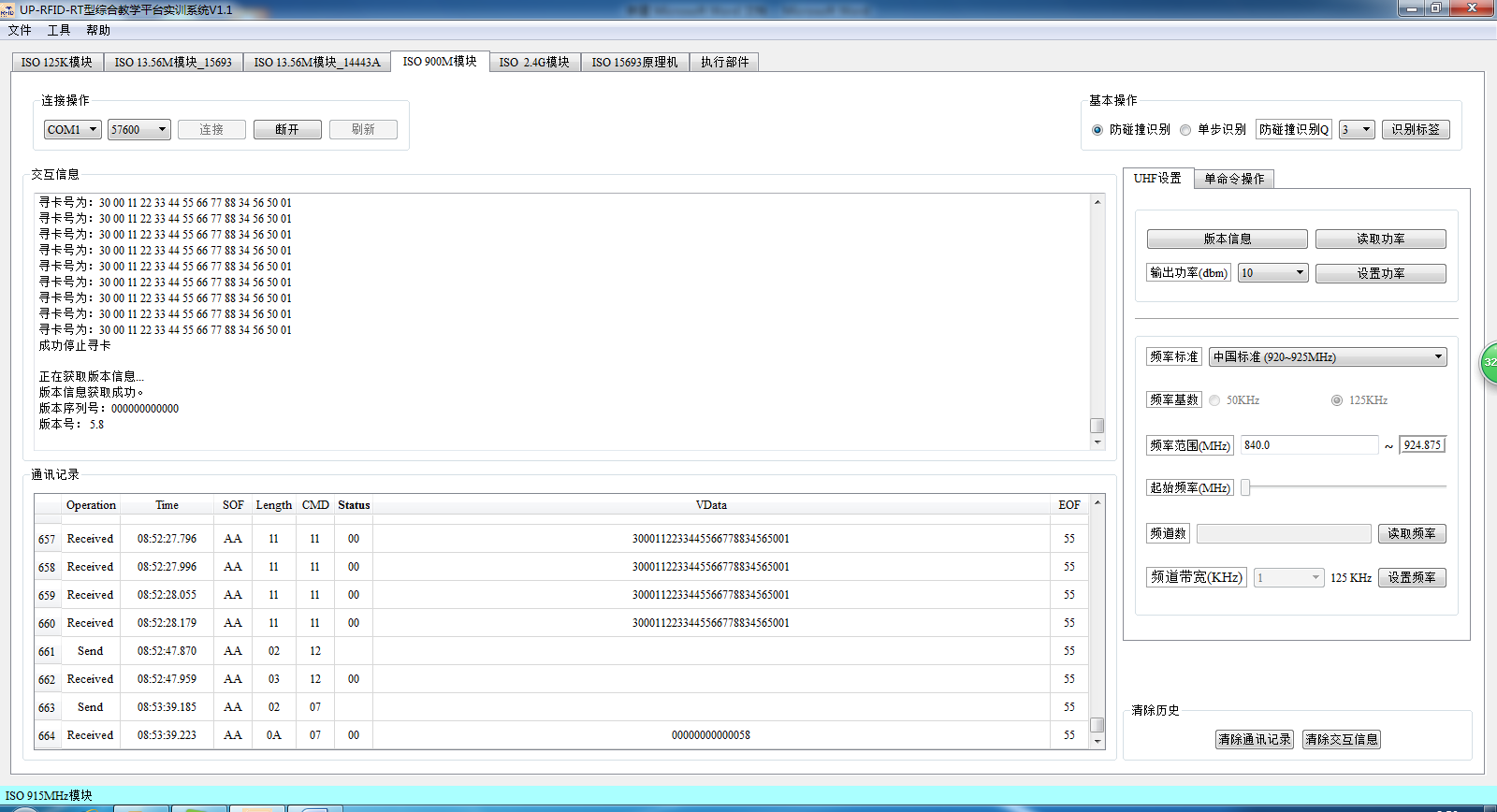


选择防碰撞识别后点击识别标签按钮的UI，由于我的900M标签的卡号相同无法辨析是否读取到了两张卡，在寻卡过程中偶尔会出现30 00 11 22 33 44 55 66 77 88 34 56 00 00的卡号，不太清楚具体原因

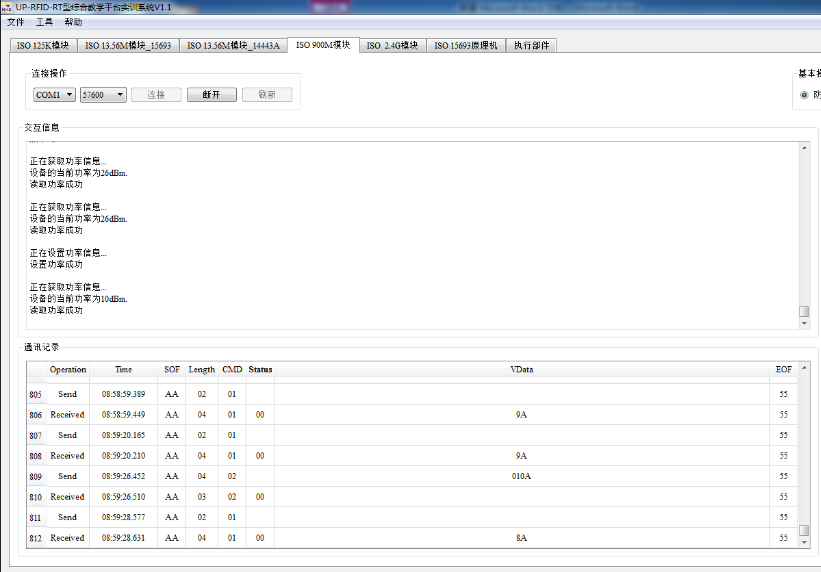


换来一张同学的标签（与自己的标签卡号不同），这时可以看到有两个不同的卡号，不过每张卡识别到的次数未必相等，主要看位置、受干扰程度了。此时，在寻卡过程中偶尔会出现30 00 11 22 33 44 55 66 77 88 34 56 00 00的卡号

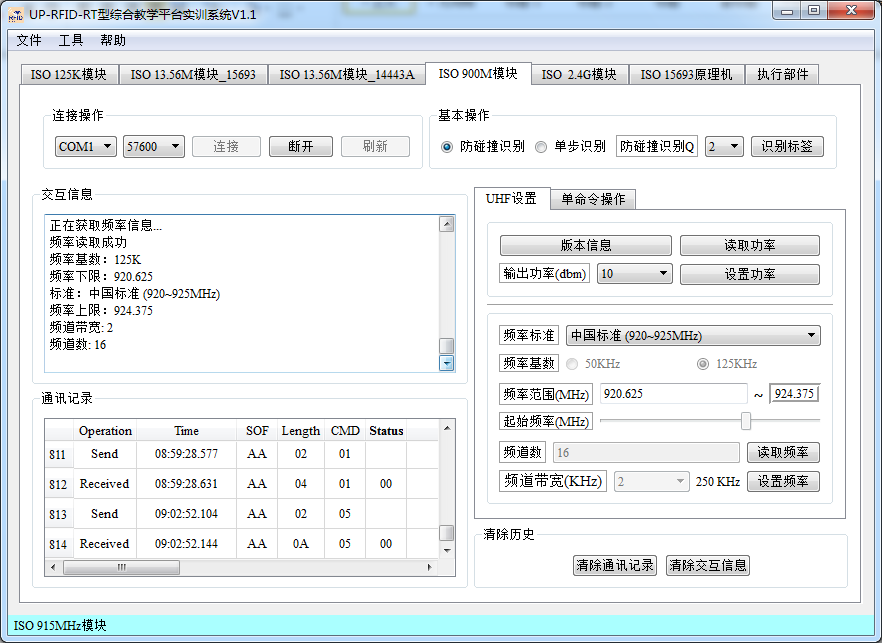
3.获取版本信息：当成功连接串口后点击版本信息按钮即可，可以获取到版本号和序列号

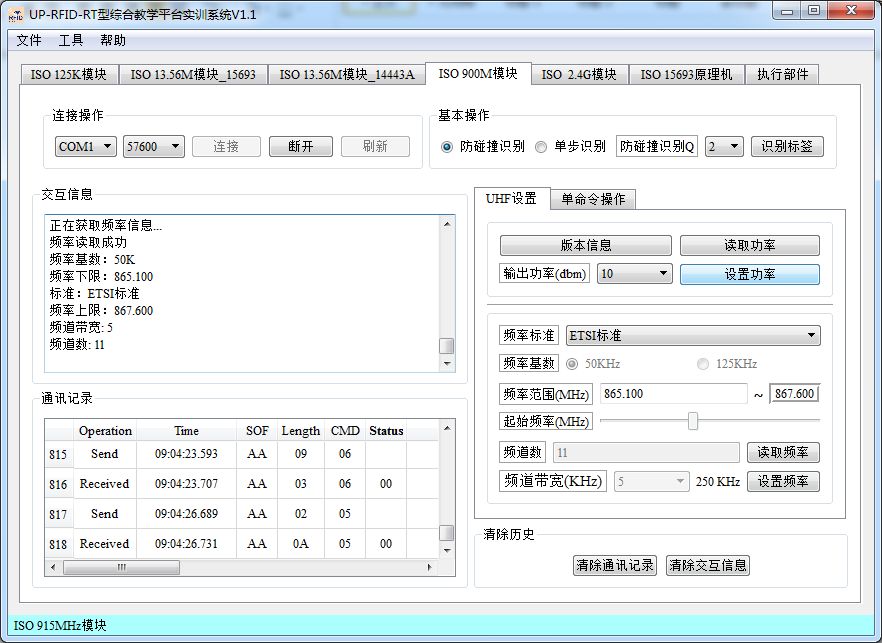


4.读取并设置功率：正常来说在使用的时候需要设置模块的功率，如果没有设置使用默认的功率



5.读取并设置频率：点击读取频率按钮，发送命令为AA 02 05 55，响应命令中VDATA部分包含了频率信息





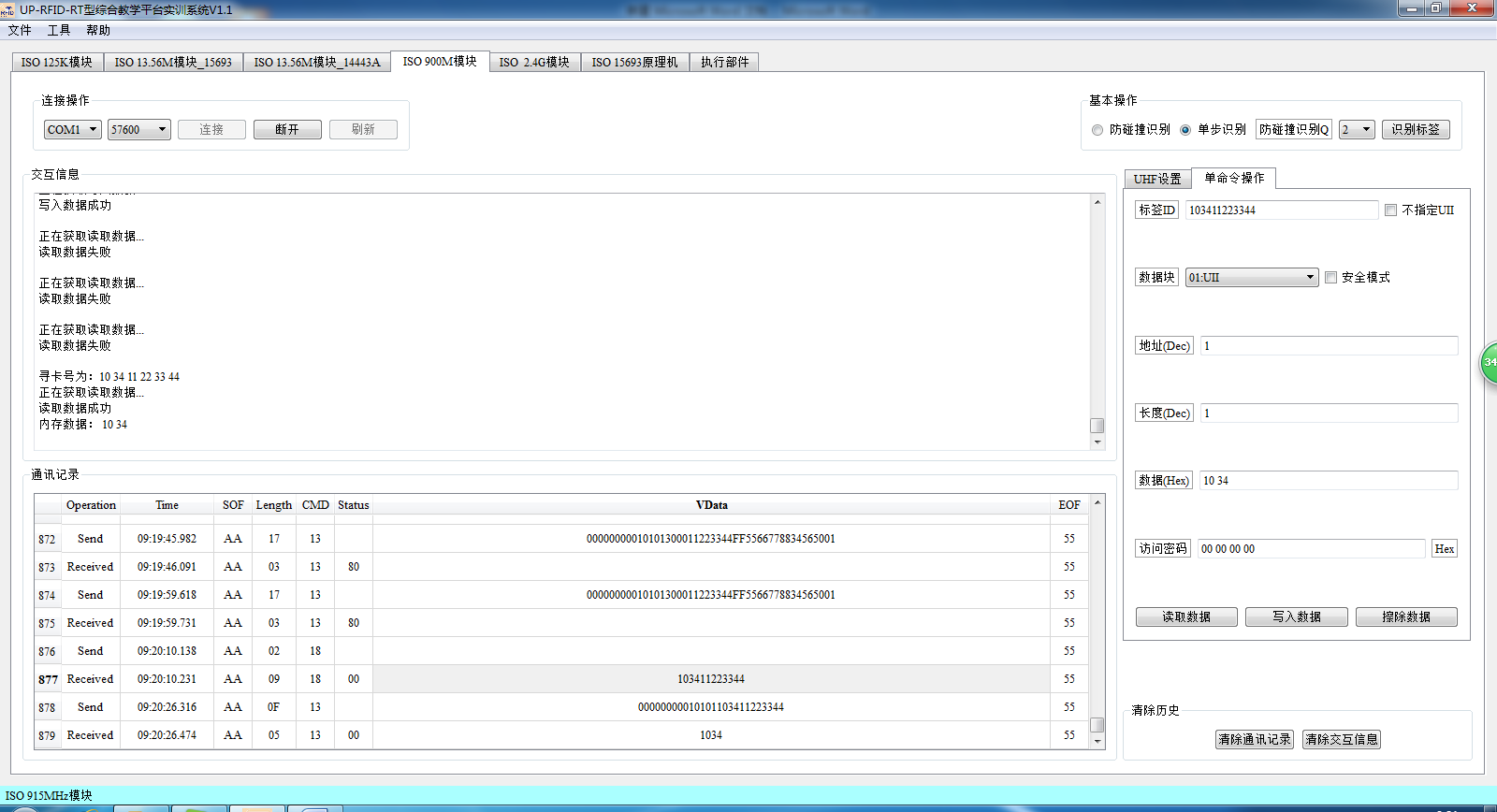
6.读取数据：读取数据分两种情况：指定UII和不指定UII。指定UII是必须指定一个UII才能读取数据，如果有多张卡则读取指定卡号的数据，一般只有一张卡的时候使用不指定UII进行读取





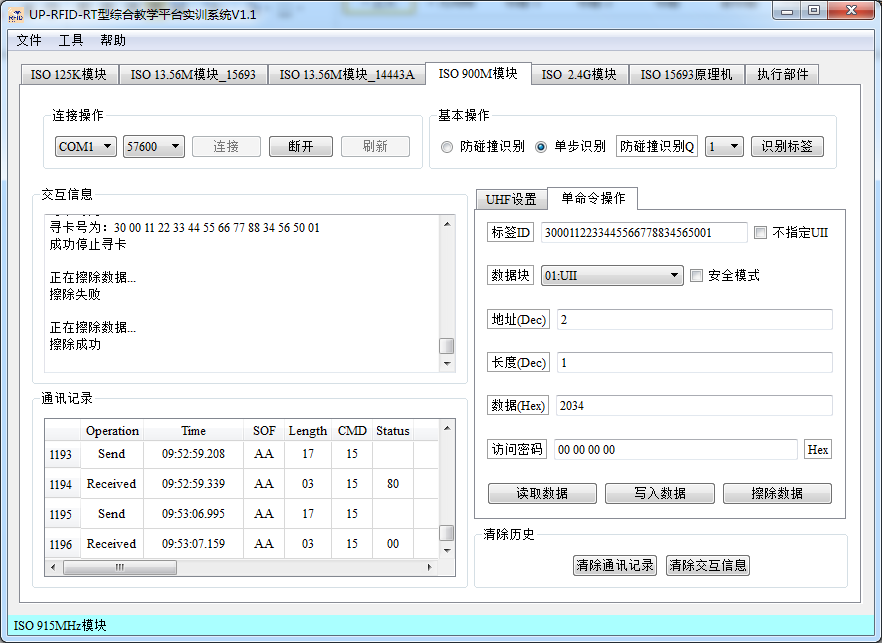
请求帧和响应帧在通讯录窗口中均详细显示，其中前四个字节为存取密码，后六个字节为卡号

7.写入数据：写入数据和读取数据类似，但是长度只能是1，也就是只能单个字进行写入。如果需要写入多个字是可以逐个写入



和读取一样，前面四个字节为存取密码，接着的分别是数据块、地址、长度，接下来的2个字节为写入的实际内容，后面的字节为写入卡的卡号。写入的时候需要注意，当修改UII数据块地址1处的数据，会引起卡号的变更，UII的第一个字为协议控制字

8.擦除数据：擦除数据和写入数据类似，目前也只支持擦除一个字，擦除之后该地址处所有数据均为0，也就是擦0的方式擦除



## 五、实验体会与总结

通过本次实验了解了超高频读写器的基本原理，学会了如何使用超高频读写器，掌握了超高频读写器和标签参数的含义和设置方法。进一步加深对 Gen2 协议下标签的存储结构以及 Gen2 协议的理解。通过读写器试验箱，掌握了对 Gen2 协议下标签读写操作，并熟悉了超高频读写器 API 函数的调用。

# 第五次实验内容：RFID综合应用实验 (ISO15693)

## 一、实验目的

本次实验属于综合性应用实验，要求能够灵活应用 RFID 技术原理，解 决实际生活中遇到的应用问题，培养分析问题、解决问题的能力以及综合知 识的应用能力。 由于 RFID 技术应用范围非常广泛，本次实验限定应用 13.56M 读写器、基 于 ISO14443A 协议的电子标签、基于 ISO15693 协议电子标签开发两套综合应用 系统。

## 二、实验内容

采用北京博创 RFID 实验箱模拟图书管理系统的读卡设备、支持 ISO15693 协议的 S50 卡（5 张）模拟图书，一张卡作为用户身份的唯一识别卡，其他四张 卡与唯一的一本图书关联。

用户首次申请领用该卡（称用户卡）时，保存个人手机号，并对卡进行初始 化。将卡与个人手机关联、姓名关联起来（采用实名制，便于挂失）。 另外四张卡初始化与四本图书关联，标识图书的唯一性。将最近的 5 次借阅 信息同时存储在卡内和数据库中。

一旦用户卡丢失，马上通知管理员挂失，系统自动将该卡设置为未激活状态， 锁定消费。当然，每次消费时，一定要判断该用户卡是否处于激活状态。

假设不同的图书可供借阅的时间长短不一样，用户利用该用户卡借阅不同图 书，在卡内记录最近五条借还明细，借还明细同时写入系统数据库表中。系统能 够规则，提示图书偿还时间。

假定读写器设备与上位机始终保持联系，上位机与数据库服务器始终保持联

系。

## 三、实验要求

系统功能

(1) 用户发卡管理；

(2) 图书与电子标签关联管理；

(3) 用户毕业时的销卡管理，清除卡内借/还数据以及个人手机号；

(4) 用户借或还图书时，在卡内和数据库中同时保存借阅记录信息，假定记 录信息不超过 5 条；

(5) 图书借/还明细查询。显示借/还明细时，必须同时显示用户手机号、电子 标签 ID 号；

系统表结构

(1) 用户基本信息表（卡号、姓名、手机号、…、是否激活）；

(2) 用户借/还信息表（卡号、姓名、电子标签 Id、借阅时间、还书时间、…）；

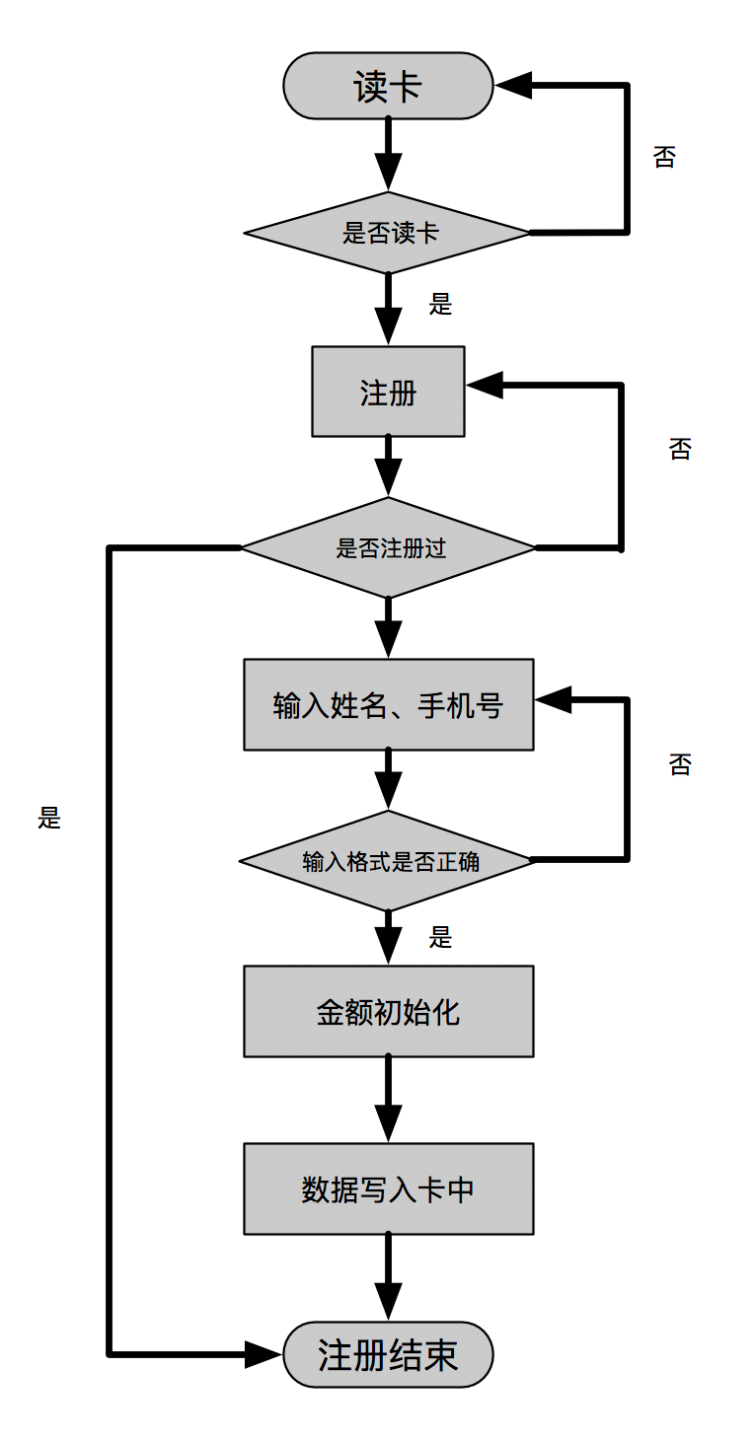
(3) 图书基本信息表（商品编号、电子标签 Id、图书名称、作者、出版社、 出版时间…）；

开发工具：采用C++编程语言，Mysql 作为数据库管理平台

## 四、实验内容及结果

1、实现的第一个功能：用户的发卡管理：

先构思流程图，如下：

C:\Users\asus\AppData\Roaming\Tencent\Users\377817149\QQ\WinTemp\RichOle\RBEKT_QNI3DB~8A$5@@9BC8.png



首先，连上设备以后，我们进入到用户管理界面。

通过以下代码读取RFID卡的卡号：

bool UHF\_Thread::ReadCardID()

{

uint16 cmd = RC632\_CMD\_ISO15693\_INVENTORY16;

char \*data = (char \*)Dll->RC632\_SendCmdReq(cmd,NULL,0);//获取读卡指令

if(serialport->write(data+2,data[0]))//data,len

{

return true;

}

else

{

return false;

}

}

在实验过程中我发现，15693卡的坏卡率十分的高，刚开始着手这个实验的时候，以为是函数问题，五张卡完全都读不了。但是后来检查了代码以后觉得可能还是设备问题，再借了几张，果然如此。



读取卡号后，添加姓名电话性别来进行用户卡的注册。

这里我写了写卡的函数，可以把姓名电话都写入卡中，完成卡的初始化，以下是代码：

bool UHF\_Thread::WriteCard(QString UID, int ADDR, uint8 \*DATA){

uint16 cmd = RC632\_CMD\_ISO15693\_WRITE\_SM;

uint8 Flag,Uid[8],Addr;

Flag=02;

uint8 UID\_len = tool->StringToHex(UID,Uid);

Addr = ADDR;

uint8 vdata[14];

uint8 len=0;

vdata[len]=Flag;

len++;

for(uint8 i=0;i<UID\_len;i++,len++){

vdata[len]=Uid[i];

}

vdata[len]=Addr;

len++;

for(uint8 i=0;i<4;i++,len++){

vdata[len]=DATA[i];

}

qDebug()<<"vdata is";

for(int i=0;i<14;i++){

qDebug("%x",vdata[i]);

}

char \*senddata = (char \*)Dll->RC632\_SendCmdReq(cmd,vdata,14);

qDebug()<<"senddata is";

for(int i=0;i<senddata[0];i++){

qDebug("%x",senddata[i]);

}

if(serialport->write(senddata+2,senddata[0])>0)//data,len

{

qDebug("WRITE SUCCESS!");

}

else

{

qDebug("WRITE FAILED!");

}

return true;

}

代码通过串口读取了文本框中输入的信息，并且放到了卡的0号寄存器当中。

但是由于姓名是汉字，我们必须先把姓名转化成十六进制的数字才能实现，于是在TOOLS.CPP文件中我加入了转换函数：

QString Tools::ChineseToHex(QString Str){

QTextCodec \*codec = QTextCodec::codecForName("utf8");

QString strout;

QString unidata = codec->toUnicode(Str.toUtf8().data());

for (int i=0; i<unidata.length(); ++i)

{

ushort num = unidata[i].unicode();

if (num < 255)

strout += "00";

strout += QString::number(num,16);

}

qDebug()<<"unidata="<<unidata;

qDebug()<<"strout="<<strout;

return strout;

}

这里要注意到，我们需要通过“添加”这个按钮来作为触发，来实现读取，所以我们需要在USERMANAGE.CPP中添加触发的函数：

void UserManage::add\_user()

{

QString LabelName[] = {"卡号：", "姓名：", "性别：", "年龄："};

for(int i = 0; i < Edit\_Count\_USER; i++)

{

if(Edit[i]->text().isEmpty())

{

QMessageBox::warning(NULL, "warning", LabelName[i]+"不能为空！", QMessageBox::Yes, QMessageBox::Yes);

return;

}

}

if (sql->SelectBooks(Edit[ID\_User]->text()).next())

{

QMessageBox::warning(NULL, "warning", "卡号已经注册为书籍！", QMessageBox::Yes, QMessageBox::Yes);

return;

}

int ret = sql->InsertUser(Edit[ID\_User]->text(),Edit[Name\_User]->text(),Edit[Gender\_User]->text(),Edit[Age\_User]->text().toInt());

if(!ret)

{

QMessageBox::warning(NULL, "warning", "添加失败，编号已存在！", QMessageBox::Yes, QMessageBox::Yes);

return;

}

QMessageBox::warning(NULL, "warning", "添加成功！", QMessageBox::Yes, QMessageBox::Yes);

//这是添加的写卡函数，通过这个和按钮相连。

EmitWriteUsrInfo(Edit[ID\_User]->text(),Edit[Name\_User]->text(),Edit[Gender\_User]->text(),Edit[Age\_User]->text(),0);

ClearEdit();

ShowTable(sql->SelectUser());

}

然后我们便可以通过DEMO来查看卡内的情况





可以发现，写卡成功。第一个功能完成。

2、实现图书与卡的关联

QString LabelName[] = {"卡号：", "书名：", "作者：", "出版社：", "总数（本）", "可借时长（天）"};

for(int i = 0; i < Edit\_Count\_BOOKS; i++)

{

if(i==Residue\_Books) continue;//剩余数量可以不填

if(Edit[i]->text().isEmpty())

{

QMessageBox::warning(NULL, "warning", LabelName[i]+"不能为空！", QMessageBox::Yes, QMessageBox::Yes);

return;

}

}

依葫芦画瓢，按照其初始的写法，我加入了一行可借时长，这个修改比较简单，需要修改一些函数和常量，如Edit\_Count\_BOOKS，还需要在后面的表格的代码段加入可借时长这一参数。

同样是通过READ\_CARD函数，我们实现卡和书的绑定：

bool UHF\_Thread::ReadCardID()

{

uint16 cmd = RC632\_CMD\_ISO15693\_INVENTORY16;

char \*data = (char \*)Dll->RC632\_SendCmdReq(cmd,NULL,0);//获取读卡指令

if(serialport->write(data+2,data[0]))//data,len

{

return true;

}

else

{

return false;

}

}



就这样完成了书和卡的绑定，并增加了一个参数，可借时间。

1. 用户毕业时的销卡管理



我将销卡管理和删除管理合二为一，再删除的同时完成卡的注销工作。

首先我们需要修改

QString LabelName[] = {"卡号：", "姓名：", "性别：", "年龄："}; //标签文本

QString ButtonName[] = {"添加", "注销", "修改", "搜索","挂失"}; //按钮文本

//布局

QVBoxLayout \*MainLayout = new QVBoxLayout();//主布局

QHBoxLayout \*ButtonLayout = new QHBoxLayout();//按钮布局

QHBoxLayout \*EditLayout = new QHBoxLayout();//文本框布局

QHBoxLayout \*TableLayout = new QHBoxLayout();//表格布局

QGroupBox \*UserTable = new QGroupBox();//用户表格组合框

QGroupBox \*UserInfo = new QGroupBox();//用户信息组合框

sql = new Sqlite();//数据库操作相关的对象

uhf = new UHF\_Thread();

tool = new Tools();

//初始化文本框和标签

for(int i = 0; i < Edit\_Count\_USER; i++)

{

Edit[i] = new QLineEdit();

Label[i] = new QLabel(LabelName[i]);

EditLayout->addWidget(Label[i]);//标签添加到布局中

EditLayout->addWidget(Edit[i]);//文本框添加到布局中

}

这样才能得到上图所示的界面。

与写卡功能类似，注销功能即把卡内的信息全部清零，以上一张用户卡为例



单击卡后选择注销，我利用了写卡的函数做修改，使其回到原始状态：

bool UHF\_Thread::ClearCard(QString UID, int ADDR, uint8 \*DATA){

uint16 cmd = RC632\_CMD\_ISO15693\_WRITE\_SM;

uint8 Flag,Uid[8],Addr;

Flag=02;

uint8 UID\_len = tool->StringToHex(UID,Uid);

Addr = ADDR;

uint8 vdata[14];

uint8 len=0;

vdata[len]=Flag;

len++;

for(uint8 i=0;i<UID\_len;i++,len++){

vdata[len]=Uid[i];

}

vdata[len]=Addr;

len++;

for(uint8 i=0;i<4;i++,len++){

vdata[len]=DATA[i];

}

qDebug()<<"vdata is";

for(int i=0;i<14;i++){

qDebug("%x",vdata[i]);

}

char \*senddata = (char \*)Dll->RC632\_SendCmdReq(cmd,vdata,14);

qDebug()<<"senddata is";

for(int i=0;i<senddata[0];i++){

qDebug("%x",senddata[i]);

}

if(serialport->write(senddata+2,senddata[0])>0)//data,len

{

qDebug("WRITE SUCCESS!");

}

else

{

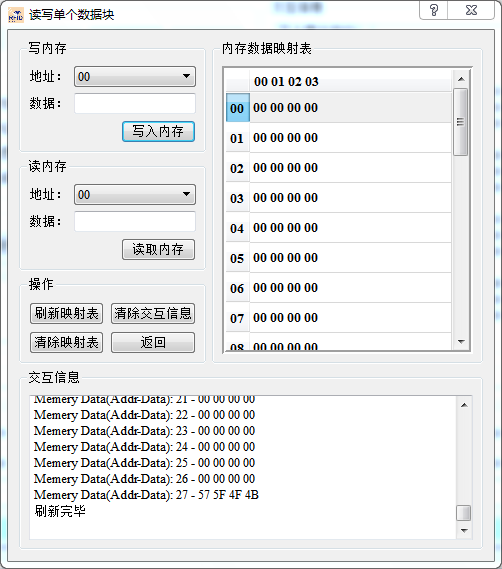
qDebug("WRITE FAILED!");

}

return true;

}

在使用了函数后，我们通过DEMO来查看数据：



全部清零，又回到了初始化的状态，功能3完成。

1. 图书借还查询

首先我们通过以下函数来实现，当我们借书时候，如果先录入书的时候我们需要点击书，在录入卡，就能完成借书的操作。

如果我们先录入卡的话，直接录入书便能完成借书的操作。

void Borrow\_Return::SetInfo(QString cardID)

{

//将用户信息显示到文本框中

QSqlQuery query = sql->SelectUser(cardID);

if(query.next())//如果是用户

{

for(int i=0; i < Edit\_Count\_BORROW\_RETURN; i++)

{

Edit\_User[i]->setText(query.value(i).toString());

}

Borrowable=1;

//将书信息显示到表格中

ShowTable(sql->SelectBooksOfBorrow(cardID));//显示表格内容

return;

}

query = sql->SelectBooks(cardID);

if(query.next())//如果是书

{

if(Edit\_User[CardId\_User\_Borrow]->text().isEmpty())

{

return;

}

if(Borrow->isChecked())

{

if(Borrowable==0){

QMessageBox::warning(NULL,"warning","请先归还过期书籍！",QMessageBox::Yes,QMessageBox::Yes);

return;

}

if(sql->SelectRecord(Edit\_User[CardId\_User\_Borrow]->text(), query.value(0).toString()).next())

{

return;

}

if(query.value(5).toInt() <= 0)

{

return;

}

if(sql->InsertRecord(Edit\_User[CardId\_User\_Borrow]->text(), query.value(0).toString()))//将用户ID和书籍编号添加到数据表中

{

//书籍的剩余数量-1

sql->UpdataBooks(query.value(0).toString(),query.value(1).toString(),query.value(2).toString(),query.value(3).toString(),query.value(4).toInt(),query.value(5).toInt()-1);

}

}

else

{

if(!sql->SelectRecord(Edit\_User[CardId\_User\_Borrow]->text(), query.value(0).toString()).next())

{

return;

}

if(sql->DeleteRecord(Edit\_User[CardId\_User\_Borrow]->text(), query.value(0).toString()))//将用户ID和书籍编号添加到数据表中

{

//书籍的剩余数量+1

sql->UpdataBooks(query.value(0).toString(),query.value(1).toString(),query.value(2).toString(),query.value(3).toString(),query.value(4).toInt(),query.value(5).toInt()+1);

}

}

ShowTable(sql->SelectBooksOfBorrow(Edit\_User[0]->text()));//显示表格内容

}

}

在这里，需要借的时候需要先判断是否有超时书籍，如果有需要先还以后再借。



借书成功后可以看到书籍的剩余数量变为了0，同时显示了借书时间等信息。此时查看借书列表：



当我们进行还书操作时，我们需要通过主界面点击还书按钮，再次测试时查看图书管理系统的借书列表会发现已经清空。第四个要求完成。

## 五、实验体会与总结

在这次实验中我充分了解到了RFID的应用的方式，同时对RFID也产生了一些兴趣。

实验课上与老师交谈了很久，深切的感受到了，想要把一件事情做好就需要先发自内心的觉得这件事有意思，有兴趣，这样才会继续深入思考，然后得到最后的答案。

这一次实验我分到的是15693的卡，15693与14443的不同之处就在于，同样是QT编写的，14443就有图形化的编写界面可以来完成，而15693直接是由代码构成了整个的窗口框架。但是也是由于是代码来写的框架，我才会耐着性子一点点看，否则可能便不会看懂整个代码的结构和编写的技巧。

在完成整个RFID课设的过程中遇到了很多问题，在初始阶段一个人埋头看，但是由于15693的卡的损坏率很高，一度以为是代码的问题，看了挺久的代码，感觉也没什么问题，才转身想到试试其他的卡，发现果然是卡的原因，虽然初期浪费了很多的时间，但是对于整个代码的理解其实还是有帮助。

在实验中期的时候，我发现很多出错的地方，比如只能改define定义的值才不会产生BUG等很多奇怪的事情。所以我在修改界面的时候小心翼翼，做一步调试一步，才算是一点点调试成功，很多问题试了几个地方成功的，我其实还有一些地方说不出个所以然。

这次实验总的来说应该是大体上85%的功能都实现了，在很紧张的期末阶段能完成这样，我觉得也还算是比较满意。

非常感谢老师在实验课上的指导，不仅仅实在学术方面，更是对于平常的生活中的一些感悟和点播，让我觉得有种茅塞顿开，拨云见日之感。