

Programming sirit 610 notes

Written By:

Mustafa S. Al-Ghafli

Revision

0.1v (12/29/2012)

Contents

Introduction	3
Sirit Interfaces	3
Sirit NameSpaces	3
Command Category	4
Command Responses.....	5
Sirit Program	5
Lan interface	14
Useful commands (read tag, change tag id, ...)	16
TCP client programming to Sirit	16
To-Do.....	24

Introduction

This is a getting started with Sirit 610 programming and interface. This document assume the following:

- programming knowledge
- TCP client programming knowledge (sending, receiving and connecting to socket (IP, PORT))
- software installed (Sirit program , putty , jdk)

Sirit Interfaces

Sirit interfaces are means to connect to the reader to send commands and receive data from the reader

Sirti 610 reader has 3 interfaces:

- Serial interface
- USB interface
- LAN interface

This document will explain how to interface with the reader using LAN interface.

Sirit NameSpaces

NameSpaces are top level category provided by the reader. It is a nice and neat way to organize commands. Sirit has the following nameSpaces:

- **Reader** – Generic reader level functions
- **Setup** – Reader setup
- **Info** – Reader information
- Version – Hardware and software versions
- Com – Communication level features
- **Tag** – Tag control features
- DIO – Digital input and output features
- **Antennas** – Antenna configuration
- Modem – Low-level modem control
- User – User defined variables
- Errors – Errors returned by reader
- Events – Events returned by reader
- Diag – Diagnostic features
- Enum – Lists (arrays) of values for specific commands

highlighted above the important and common NameSpaces. This means if you would like to send commands to the reader regarding **tag** operations you will use the **Tag** nameSpace Example of this:

tag.read_id()

and the reader will response with the tag id (more on that later)

another example if you would like to know wither or not the reader is functional you would send this command:

reader.is_alive()

and the reader will response with ok or error message.

Command Category

Sirit uses three types of Commands.

Get

A Get action retrieves the value of a specified reader configuration variable. The response is either **OK** or an error message. The syntax of a Get command is as follows:

variable_name

In the following example, the **com.serial.baudrate** variable is retrieved:

```
--> com.serial.baudrate
```

```
<-- ok 57600
```

please note the following notations:

```
--> means command sent to the reader
```

```
<-- means the reader response
```

Set

A Set action sets the value of the specified reader configuration variable. The syntax of a Set command is as follows:

variable_name = value

In the following example, the **com.serial.baudrate** variable is set to 9600:

```
--> com.serial.baudrate = 9600
```

```
<-- ok
```

Execute

An Exec action executes a reader function. The syntax of a Exec command is as follows:

```
function_name(name1=value1, name2=value2)
```

In the following example, the tag.write command writes a kill code of 0x12345678 to a tag with an EPC of 0x30112233445566778899aabb:

```
-->tag.write (tag_id = 0x30112233445566778899aabb,  
Kill_pwd = 0x12345678)  
<-- ok
```

Command Responses

As shown in the previous section, the reader supports three types of commands: Get, Set and Exec. The response to all commands is either **OK** or an error message.

Get Command Response

For a Get command the successful response is **ok** followed by the value of the variable.

Set Command Response

For a set command the successful response is simply **ok**.

Exec Command Response

An Exec action executes a reader function. For an Exec command, the successful response is **ok** followed by data returned by the function. The data returned is different for each function.

Error Responses

In the event a command fails, an error is returned. The errors returned by the reader are contained in the error namespace. All possible errors in the error name space are shown in Sirit manual.

Sirit Program

This section will walk you through the sirit program. you will learn how to use the Sirit program to do the following:

obtain the IP address of the reader

obtaining the ip address of the reader is important if you want to interface the reader using the lan interface. The following steps will show you how to obtain the reader ip address using Sirit program.



Readers Discovered on the Network										
Mac Address	IP Address	Host Name	Serial Number	Version	Method (IPv4/IPv6)	Location	Zone	Subnet	Gateway	Model
00:23:68:C7:00:84	172.16.0.56 fe80::223:68ff:fe7d:84/64	N610C70084.pc.ccae.kdupm.edu.sa	96FC4404038C78DC	1.0.17999	dhcp/radv_only	unknown	unknown	255.255.0.0	172.16.0.253 none	610

Setup a profile

setting up a profile is important because you specify the setup information like number of antennas and how many tags the reader will at max read at once. The following steps will show you how to do that.
(Side note: the admin password by default is **readeradmin**)

Discovery

Refresh

Options

Reader Toolbox

Setup

Network Setting

Test

Configure

Diagnose

Advanced Tools

Lane Mapper

WTC



Welcome to the INfinity 610 Setup Wizard

This Wizard will guide you through the initial setup of the INfinity 610 reader.

To continue, click Next>.

< Back

Next >


Cancel

Help

Infinity 610: Reader Setup Wizard (172.16.0.56)

Protocol Selection

Enable reader protocols.



Select the protocols to enable.

☒ ISO 18000-6C (ISOC) - EPC1 Gen2

Next generation of UHF RFID Tags which are standardized by EPCGlobal and ISO.

☒ ISO 18000-6B (ISOB)

Standardized ISO tags which are used in Europe.

☐ EASAlarm

Next generation of UHF Gen2 tags based on NXP silicon which provide custom features for Electronic Alarm Surveillance.

< Back

Next >

Cancel

Help

Antenna Selection

Select your antenna configuration.



Please select the antenna(s) to enable

☒ 1

☐ 2

☐ 3


☐ 4


< Back

Next >

Cancel

Help

INfinity 610: Reader Setup Wizard (172.16.0.56) 

Tag Volume
Select the tag volume. 

Estimate the number of tags presented to the reader at any one time.

☐ A Single Tag

☐ Very Low (2-8)

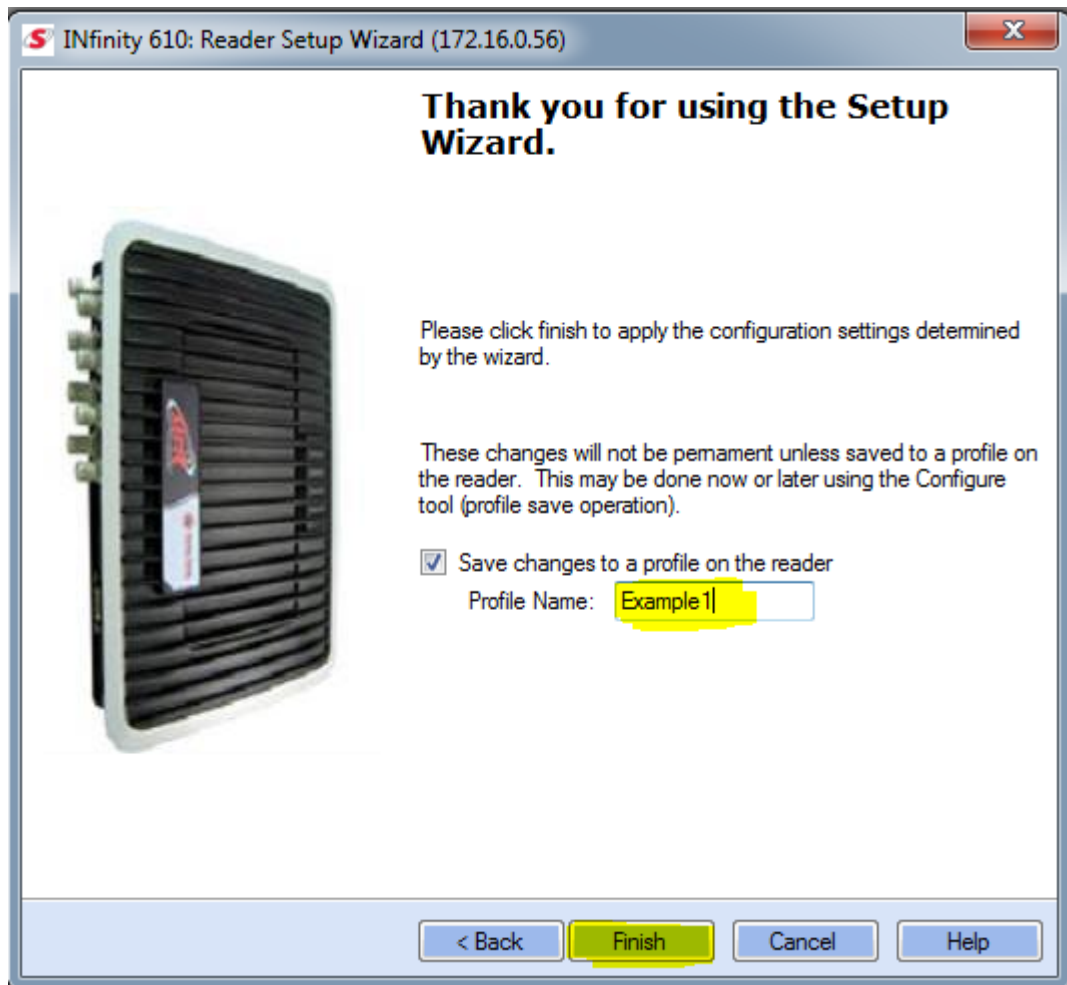
☐ Low (9-64)

☒ Medium (65-256)

☐ Large (257-512)

☐ High (513-1024)

☐ Very High (1025+)



read tags and change tags id

This section will show you how to read the tags and test the reader, also it will show you how to change the tag id this is important because all the tag have the same tag id by default. remember to put one tag at a time and change its tag id. it's also useful to write on the tag itself using a pen the tag id.

Discovery

Refresh

Options

Reader Toolbox

Setup

Network Setting

Test

Configure

Diagnose

Reader Test Tool (RTT) - 172.16.0.56

File Edit Reader Operating Mode Protocols Antennas

Region: etsi SubRegion: en302208_dense Reader Status: OK

General Page Tag Performance Tag Management Macros Event Handling Antenna Settings

Command: reader.is_alive() Send Retain Command

```
->reader.is_alive()
<-ok
```

MAC Address: 00:23:68:C7:0D:84 Infinity 610 Firmware: 1.0.17999 Operating Mode: Standby Login: admin

Reader Test Tool (RTT) - 172.16.0.56

FileEditReaderOperating ModeProtocolsAntennas

Region: etsiSubRegion: en302208_denseReader Status: OK

General PageTag PerformanceTag ManagementMacrosEvent HandlingAntenna Settings

Performance Information

Total Unique Tags: 2Tag Read Count: 16Cumulative Rate: 0Current Rate: 0

Tag Database Query

Query Interval (ms): 1000Total Query Time (ms): 0

StartGet Once☐ Purge every query☐ Beep On

Tag Acquisition Analysis

Min: 8

Avg: 8

Max: 8

Scan Operation

Scan Time (ms): 100

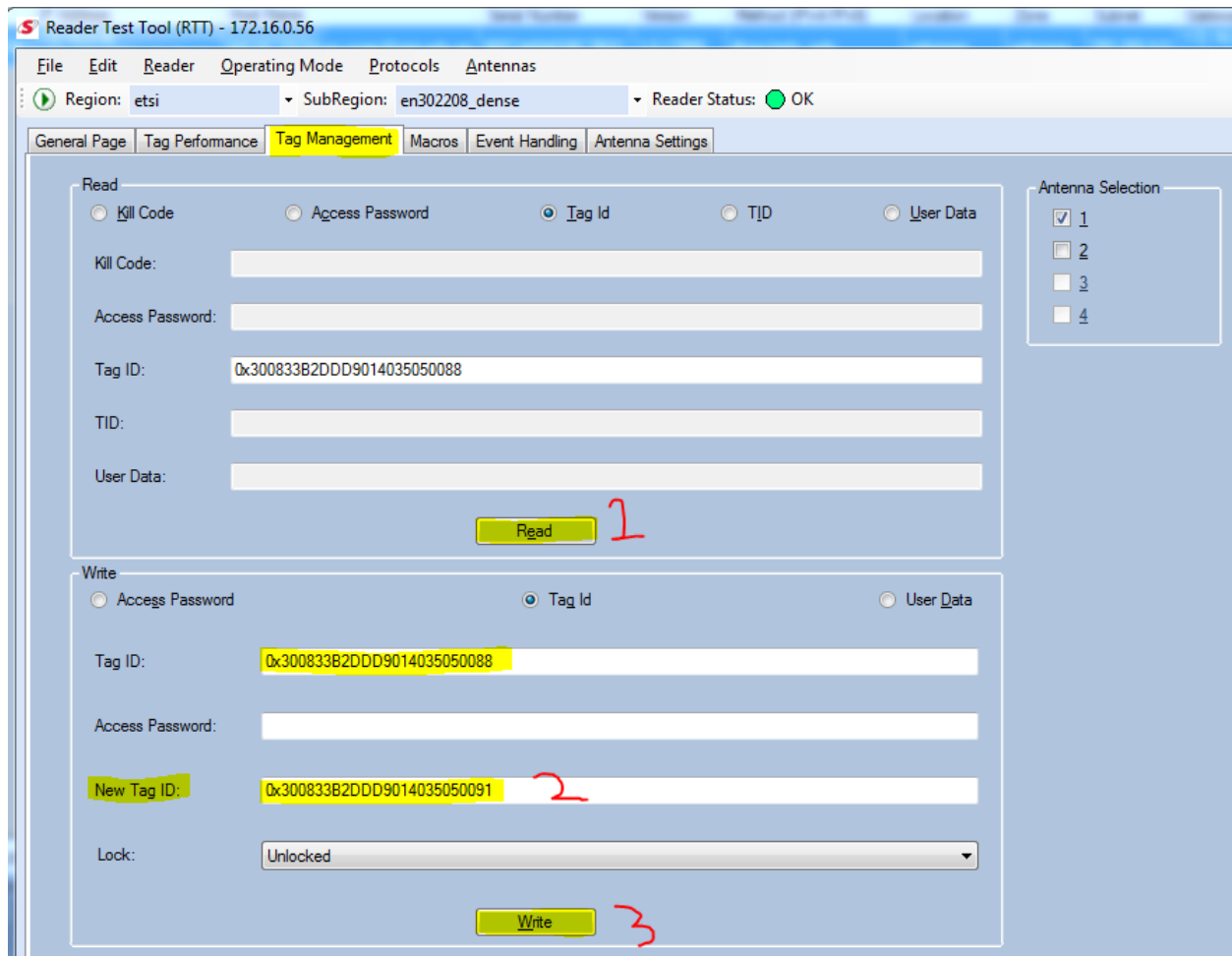
Scan Tags

☐ EPC Decode

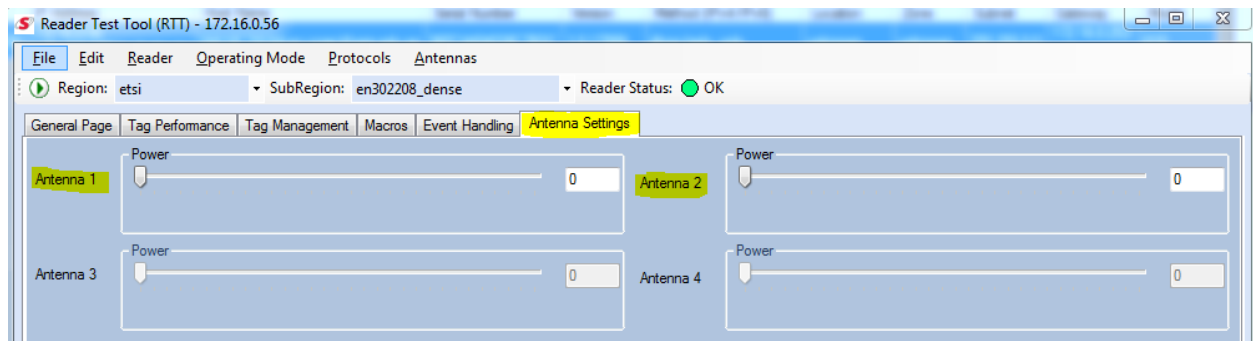
Tag ID	Type	Total	Rate	Antenna	Epc
0x300833B2DD9014035050088	ISOC	8	0		
0x300833B2DD90140350500000	ISOC	8	0		

☐ Tag Filtering

MAC Address: 00:23:68:C7:0D:84Infinity 610Firmware: 1.0.17999Operating Mode: StandbyLogin: admin



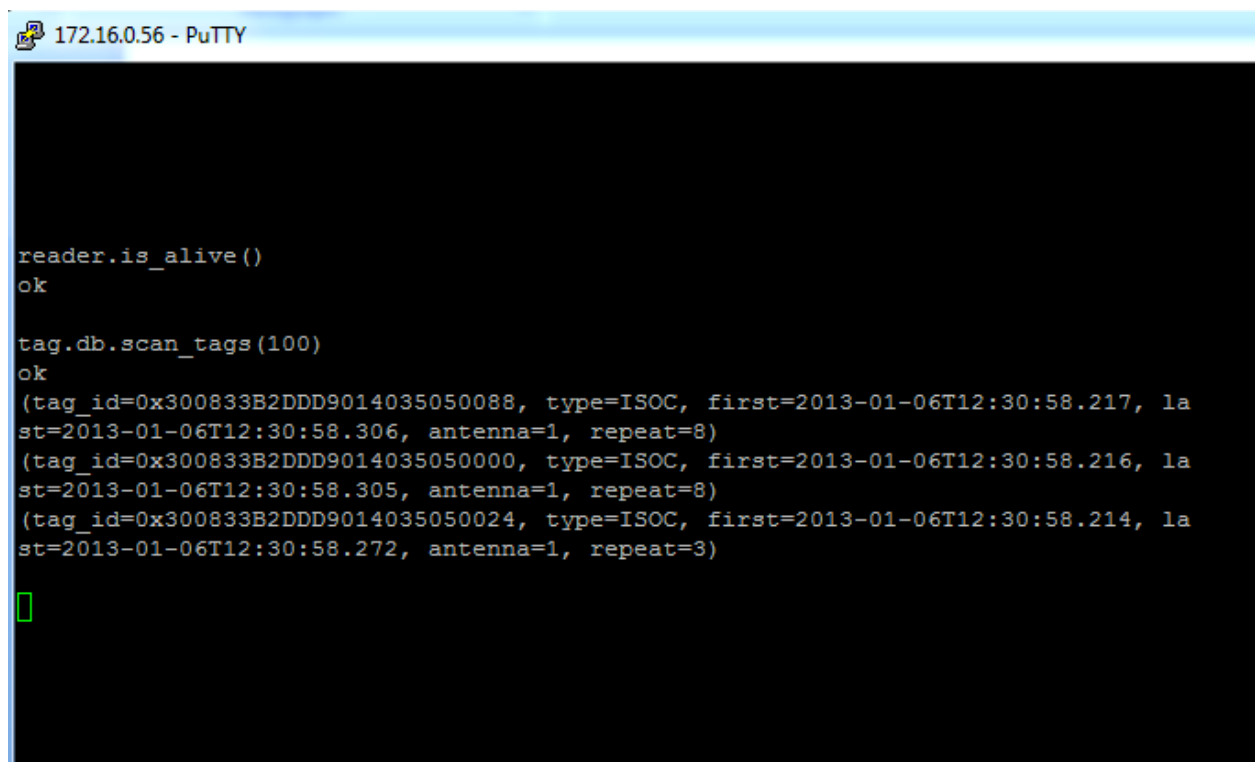
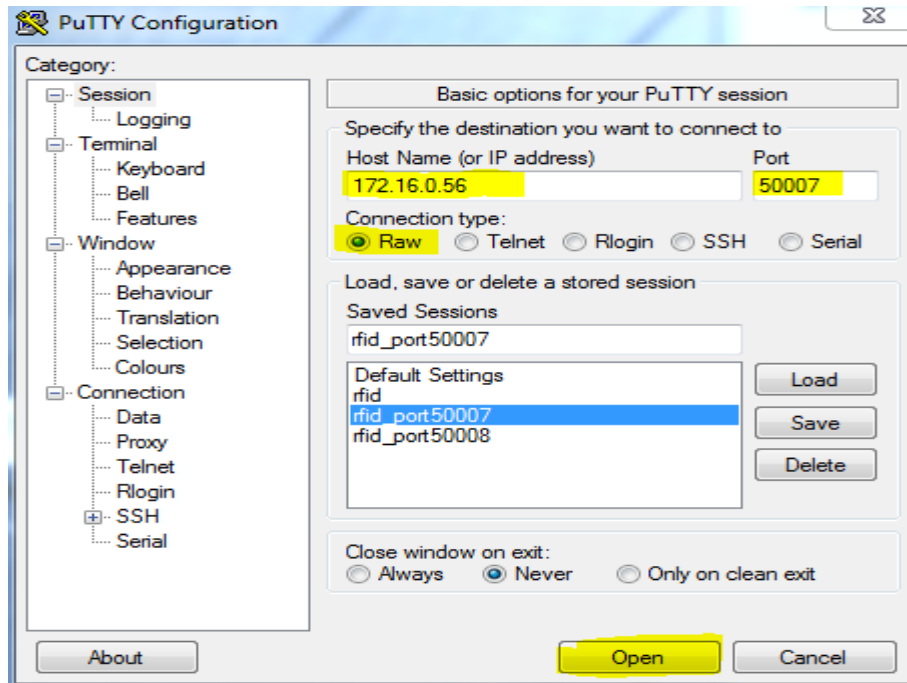
change the power gain of the antennas



Lan interface

the reader has a command port (50007). using the obtained ip address you can establish socket connection with the reader provided that you are in the same network with the reader. Keep in mind that the reader will assign itself an IP address this IP address may exist in the same network and will get a lot of error because the packets you sent will be dropped because of duplicate ips on the same network. to solve this problem it is advices that you use a modem connecting the pc and the reader

only. This issue will most likely to happen because the CCSE network is a big network and the reader will sometime assign itself an existed ip. Using putty to connect using to the reader:



as you can see the reader is listening to port 50007 the command port. Once you connect to it using its ip and the command port (50007) you can send (get, set , execute) commands and you can see the

response. Using this principle you can write a program using a language that support TCP socket connection ip. Once you establish the connection use the api provided by the language to send string stream and read from the read buffer.

Useful commands (read tag, change tag id, ...)

- `reader.is_alive()`
- `reader.check_status()`
- `tag.db.scan_tags(1000)`
- `antennas.1.advanced.gain`
- `antennas.1.conducted_power`
- `tag.read_id()`
- `tag.write_id(new_tag_id = 0x306800095EFDDF80002)`

TCP client programming to Sirit

The following code uses java to establish socket connection to the reader. Also it uses `PrintWriter` to write to sirit reader and `BufferedReader` to read the responses from sirit reader. Also the code uses two Timers (threads). one will fire frequently to send scan command to the reader. The other one will fire to check the `BufferedReader` to get the response from the reader. The reader code was design for a proof of concept. By all means the following code is not fully optimized nor clean. The program will simply scan every 100ms~300ms the area. if a tag is detected it will toggle its states to either (taken or Not taken) and then it will wait for 15 seconds and it will resume the polling process. this is done to avoid the situation where the tag will be read more than one time. so the program will keep toggling the id status.

```
import java.net.*;

import java.io.*;

import javax.swing.*;

import java.awt.event.*;
import java.awt.print.Book;
import java.awt.*;

public class Gui extends JFrame implements ActionListener{

    private JButton connect;
```



```
private JTextField ip;

private JLabel status;

private JTextArea console;

private JTable inventory;

private JButton send;

private JTextField cmd;

// this is used to establish a socket with the reader

private Socket commandSocket;

// These are used to read and write to the socket

private PrintWriter cmdOut;

private BufferedReader cmdIn;

// This will check the BufferedReader and read it and display the result in the screen

private Timer updater;

// This will keep polling the reader and will send it scan command

private Timer poller;

private static String[] ids = {

    "0x300833B2DDD9014035050088","0x300833B2DDD9014035050099","0x300833B2DDD9014035050077",
    "0x300833B2DDD9014035050055",

    "0x300833B2DDD9014035050033","0x300833B2DDD9014035050084","0x300833B2DDD9014035050044",
    "0x300833B2DDD9014035050022",

    "0x300833B2DDD9014035050024","0x300833B2DDD9014035050011","0x300833B2DDD9014035050034",
    "0x300833B2DDD9014035050066"

};

private static String[][] tableData= new String[12][2];

public Gui(){
```

```

        super("RFID Project");

        setSize(600,400);

        //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        addWindowListener(new java.awt.event.WindowAdapter() {

public void windowClosing(java.awt.event.WindowEvent e) {

        if(commandSocket != null){

                try{

                        commandSocket.close();

                }catch(Exception ex){

                        System.out.println("erro"+ex);

                }

        }

        System.exit(0);

    }

});

        setLayout(new BorderLayout());

        // create components

        makeGui();

        updater = new Timer(100,this);

        updater.setActionCommand("updater");

        poller = new Timer(300, this);

        poller.setActionCommand("poller");

    }

```

```
private void makeGui(){

    // create components

    connect = new JButton("connect");

    ip = new JTextField("172.16.0.185");

    status = new JLabel("Not connected");

    send = new JButton("send");

    cmd = new JTextField();

    String[] headerStrings = {"id", "status"};

    fillTableArray();

    inventory = new JTable(tableData, headerStrings);

    JScrollPane sInventory = new JScrollPane(inventory);

    JPanel northPanel = new JPanel(new BorderLayout());

    JPanel centerPanel = new JPanel(new BorderLayout());

    JPanel innerPanel = new JPanel(new BorderLayout());

    JPanel disPanel = new JPanel(new GridLayout(1, 2));

    console = new JTextArea();

    JScrollPane sConsole = new JScrollPane
(console, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);

    // setup parameters

    connect.addActionListener(this);

    connect.setActionCommand("connect");

    send.addActionListener(this);

    send.setActionCommand("send");

    ip.addActionListener(this);
```

```

        ip.setActionCommand("connect");

        cmd.addActionListener(this);

        cmd.setActionCommand("send");

        // add them to the window

        northPanel.add(new JLabel(" IP address: "),BorderLayout.WEST);

        northPanel.add(ip,BorderLayout.CENTER);

        northPanel.add(connect,BorderLayout.EAST);


        innerPanel.add(new JLabel(" command: "),BorderLayout.WEST);

        innerPanel.add(cmd,BorderLayout.CENTER);

        innerPanel.add(send,BorderLayout.EAST);

        disPanel.add(sConsole);

        disPanel.add(sInventory);

        centerPanel.add(disPanel,BorderLayout.CENTER);

        centerPanel.add(innerPanel,BorderLayout.NORTH);


        add(centerPanel,BorderLayout.CENTER);

        add(northPanel,BorderLayout.NORTH);

        add(status,BorderLayout.SOUTH);

    }


    public void actionPerformed(ActionEvent evt){

        if(evt.getActionCommand().equals("connect")){

            status.setText("connecting...");

```

```

        try{

            commandSocket = new Socket(ip.getText(),50007);

            //eventSocket = new Socket(ip.getText(),50008);

            cmdOut = new PrintWriter(commandSocket.getOutputStream(),true);

            cmdIn = new BufferedReader(new
InputStreamReader(commandSocket.getInputStream()));

            status.setText("connected");

            if(updater != null){

                updater.start();

            }

            if(poller != null){

                poller.start();

            }


        }catch(Exception err){

            status.setText("failed "+err);

        }

    }else if(evt.getActionCommand().equals("send")){

        if(commandSocket!= null && commandSocket.isConnected()){

            status.setText("sending...");

            cmdOut.println(cmd.getText());

            status.setText("done sending");

        }

    }

    }else if(evt.getActionCommand().equals("updater")){

        String result = "";

```

```

        try{

            while(cmdIn.ready())

                result+=cmdIn.readLine()+"\n";

        }catch(Exception e){

            console.append("error "+e+"\n");

            updater.stop();

        }

        console.append(result);

        // find the start index of =0x

        // todo do better than this

        int si = result.indexOf("="0x");

        String id="";

        if(si >-1){

            id = result.substring(si+1,result.indexOf(",", si));

            System.out.println(id);

            changeStatus(id);

            if(poller != null){

                poller.stop();

                poller.setInitialDelay(15000); // wait time

                poller.start();

            }

        }

    }else if(evt.getActionCommand().equals("poller")){

        if(commandSocket!= null && commandSocket.isConnected()){

            status.setText("polling...");

```

```
        cmdOut.println("tag.db.scan_tags(100)");

        status.setText("done polling..");

    }

}

}

private void fillTableArray(){

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

        tableData[i][0] = ids[i];

    }

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

        tableData[i][1] = "Not Taken";

    }

}

private void changeStatus(String id){

    int index=0;

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

        if(id.equals(ids[i])){

            index = i;

            break;

        }

    }

    String newStatus = (((String)inventory.getModel().getValueAt(index, 1)).equals("Not
Taken")) ? "Taken":"Not Taken";

    inventory.getModel().setValueAt(newStatus, index, 1);

}
```

```

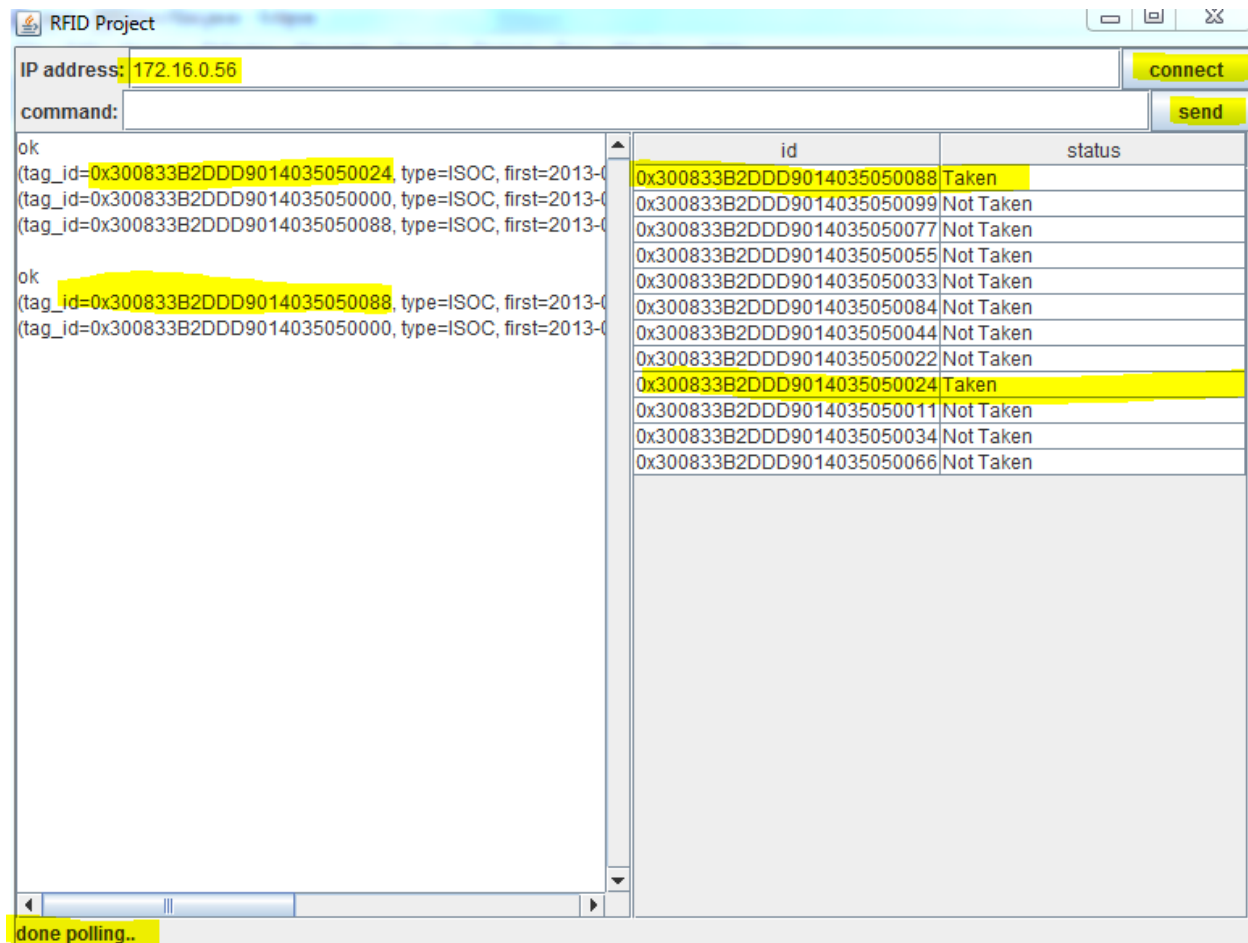
public static void main(String args[]){

    new Gui().setVisible(true);

}

}

```



To-Do

hopefully by now you are familiar with how to interface and program sirit. The following list of thing that need to be done.

- Try to connect with sirit using Serial port (you can use putty to connect to serial port)
- do test and measurements (range, # of tag that can be read at once)
- observe the effects when you change the (# of antennas , power gain , antenna orientation)

- if a duplicate ip was assigned to the reader interface the reader using serial command and search the manual on how to change the ip address of the reader (side note: to know wither or not the reader has a duplicate ip. disconnect the reader from the network and ping the ip address if that ip address is reachable that means the reader has a duplicate ip)
- improve the code to read more than one tag id (search the result for all tags and push all the tags in arraylist and then iterate through the arraylist and invoke change status
- Document any progress you make
- Read Sirit manuals for more information and details