RFID UHF SDK

TABLE OF CONTENTS

Introduction to Reader

The UHF Handheld reader features outstanding 3-D reading capability for scanning SpaceCode RFID-tagged items quickly,

Handheld reader gives you flexibility to move reader across your inventory and fast reading of items.

Ideal for accelerating processing and reading of items as they are passed between processes and individuals.

Installation

This devices have two type of connectivity options

1. Bluetooth
    a. Open Bluetooth device menu
    b. Search for new Device
    c. Find RFID READER
    d. Key Is 1234

From both above methods Bluetooth is more convenient to users and Used by most of Clients,

Because it gives wireless flexibility to move reader around

Operating instructions

- This device works on UHF frequency which not harmful to human.
- Device operates on Battery which last around 3 to 4 hours continuous use.
- Device has inbuilt buzzer, to indicate reading
- Device has two physical buttons
    - Trigger
    - Power button
- UHF tags require a litter free space around in order to increase reading accuracy
- Items in metal box will not read by reader as reader cannot penetrate through metal barrier
- Maintain 3 feet perimeter around device so that it won't pick up any unwanted tag.

Programming guide with .net C# SDK

List of Functions

This functions commands devices for functionalities like Scanning and Lighting

```
public static bool connectReader()
public static void startScan(bool asciiResult)
public static void stopScan()
public static void findTag(string tag)
public static bool encodeTag(string newUID)
public static void readSingleTag()
public static bool SetPower(int val)
public static int GetPower()
```

Delegates

Device will reply back via static delegates which you have to assign functions address

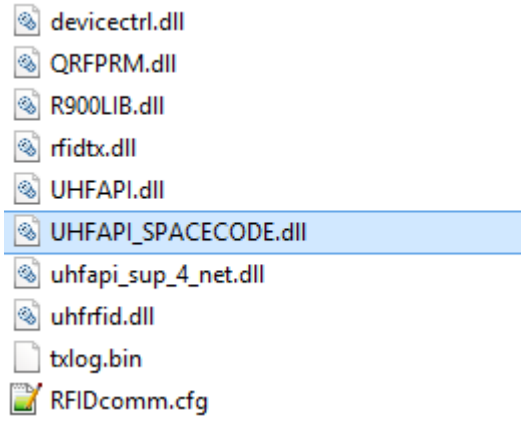
```
public delegate void scanCompleteDelegate(ArrayList str);
public delegate void tagsDelegate(string str, string r);
public delegate void deviceStatusDelegate(string str);
public delegate void errorDelegate(string str)
```

Public Status Variables

```
ArrayList  Tags_id
Bool  IsFindTag
bool  IsInScan
```

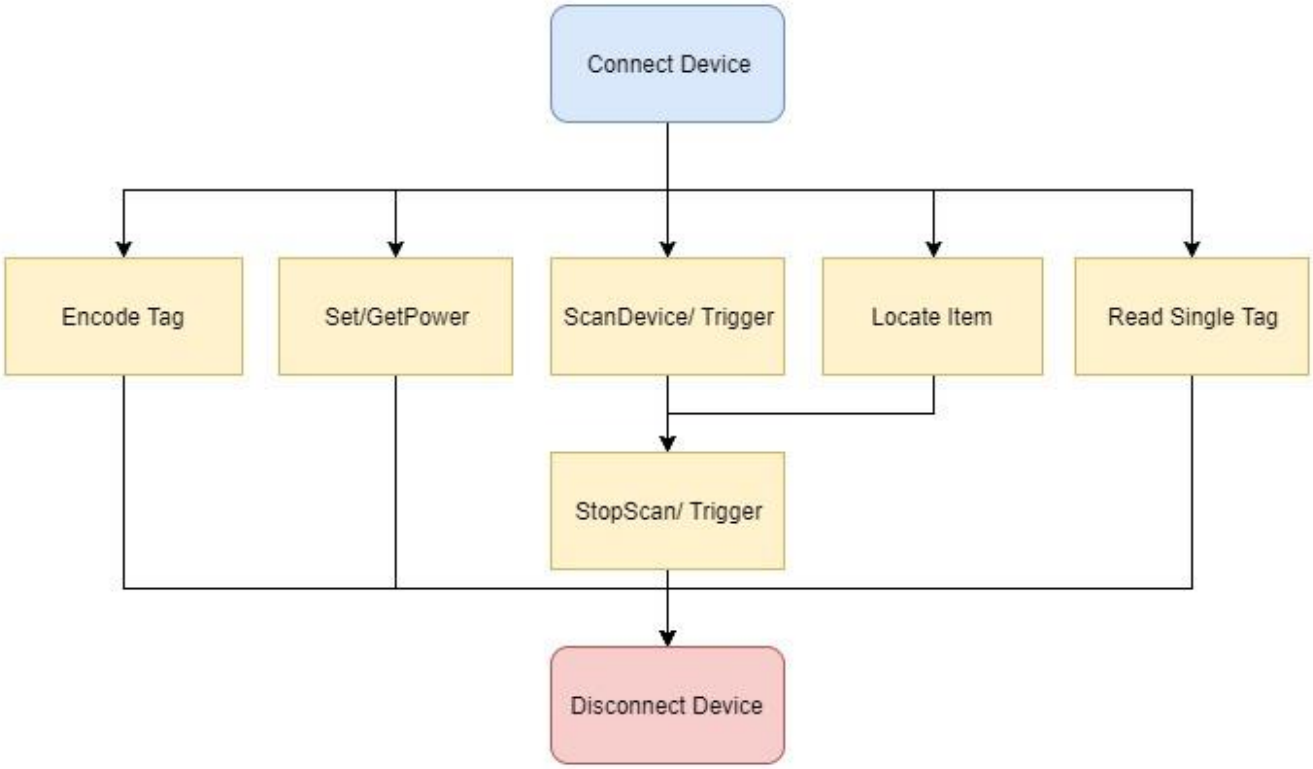
DLL and references

You need to add reference of following dll into your .net project. And rest of Dlls need to present in exe directory



SPC_RFID is the static class which includes all functions and delegate listed above.

Flowchart of function calls



Description of functions

Connect Device

Returns --> true if connected, false if failed

Signature --> public bool connectReader ()

Scan RFID device

Parameter name = " asciiResult " --> tag id in ASCII or Hex

Signature --> public void startScan(bool asciiResult)

Scan can be started by two methods, using function call and using RFID trigger button.

Pass true in parameter as result will in human readable form

Stop RFID Scan

Signature --> public void StopScan()

Scan can also stop by trigger button, if scan is running

Locate Item

Parameter name = "tag" --> tag id to locate

ignature --> public void findTag(string tag)

Reader can locate item by selecting particular tag to be found, it will give result in tagsDelegate with numerical values ranging 0 to 70 (70 means near, 0 means far)

This function is really useful to search item location.

Encode tag

Parameter name = "tag" --> tag id to locate

Signature --> public void encodeTag (string tag)

This method is used to encode tag with string value, we can encode string upto length 12

Read Single tag

Signature --> public string readSingleTag()

This method is useful in case you want to read only one tag near to reader

Result will returned by method call

Reader Range

Returns --> true if power set, false if failed

Parameter name = " val " --> power value between 0 to 31

Signature --> public bool SetPower(int val)

To set reader reading range we have to set reader power,

Power value range is 0 to 31 (0 min, 31 max)

To get Current power setting in reader we have to use following function

Returns --> power value from reader setting, from power Range

Signature --> public int GetPower()

Reader Duty

Returns --> true if duty set, false if failed

Parameter name = " val " --> duty value between 5 to 95

Signature --> public bool SetDuty(int val)

To set reader reading duty we have to set read speed,

Duty value range is 5 to 95 (5 min, 95 max)

To get Current duty setting in reader we have to use following function

Returns --> power value from reader setting, from power Range

Signature --> public int GetDuty()

SendStringToPrinter

Returns --> true if print succefull

Parameter name = "printer name " --> printer name from device and printers

Parameter name = "prn " --> string of lables in PRN format

Signature --> public bool SendStringToPrinter(string printer, string prn)

We need to generate PRN to print and encode labels in Zebra RFID printer

Returns --> Boolean values

Description of delegates

- Delegates are defined in static UHF_RFID class
- Device will respond back using this delegates
- You have to assign local function address to this static delegates before calling any function
- See example below

```
SPC_RFID _SPC_RFID = null;
  public Form1()
  {
      _SPC_RFID = SPC_RFID.getInstance();

      InitializeComponent();
  }

  private void button1_Click(object sender, EventArgs e)
  {
      _SPC_RFID.tagsDelegate= tagAdd;
      _SPC_RFID.deviceStatusDelegate = devStAdd;
      _SPC_RFID.errorDelegate = errorAdd;
      _SPC_RFID.scanCompleteDelegate = scanComplted;

      _SPC_RFID.connectReader(ReaderMake.Dtr);

  }
  public void tagAdd(string a, string r)
  {
      this.Invoke((MethodInvoker)delegate
      {
          a = (!string.IsNullOrWhiteSpace(txtTagsList.Text)) ? "\r\n" + a :
a;
          txtTagsList.AppendText(a);
          lblCount.Text = "Count : " + _SPC_RFID.Tags_id.Count;
          txtTagsList.ScrollToCaret();

      });
  }


  public void scanComplted(ArrayList tag)
  {
      this.Invoke((MethodInvoker)delegate
      {
          lblmsg.Text = "ScanCompleted:"+ tag.Count+" :tags ";

      });
```

```
}


public void errorAdd(string a)
{
    this.Invoke((MethodInvoker)delegate
    {
        //lblError.Text = a;
    });

}

public void devStAdd(string a)
{
    this.Invoke((MethodInvoker)delegate
    {
        lblConnection.Text = a;
    });
}


private void btnScan_Click(object sender, EventArgs e)
{
    if(btnScan.Text.Equals("Start Scan"))
    {
        trackBarPower.Value = 31;

        _SPC_RFID.AllowDuplicateTagScan = false;
        _SPC_RFID.SetPower(trackBarPower.Value);
        _SPC_RFID.SetDuty(trackBarDuty.Value);
        _SPC_RFID.startScan(true);
        btnScan.Text = "Stop Scan";
    }
    else
    {
        _SPC_RFID.stopScan();
        btnScan.Text = "Start Scan";
    }

}
```

You have to use Invoke function to change any form attributes as this functions are called by RFID device thread to avoid cross thread error in windows from.

Description of delegates

- **tagsDelegate**
  - this delegate will called when each tag is scanned by reader
  - if we have 100 tagged items, so when we scan those items, this will fire 100 times
  - you can use it for showing live count of scan
  - you will get all tags, and distance indicator values in other parameter

- **scanCompleteDelegate**
  - this delegate will be called once scan is completed,
  - Once we call ScanStop() function or press trigger on reader, this delegate will fired

- **deviceStatusDelegate**
  - this delegate will indicate device connection and disconnection
- **errorDelegate**
  - this delegate will be called if there is any error in Scan or device