

J4210U API Documentation

Table of Contents

Auto Detection Tags.....	2
Demo.....	2
DLL Functions.....	2
unsigned char AvailablePorts(char ports[256][8]);.....	2
unsigned char OpenPort(unsigned char* port);.....	3
void ClosePort();.....	3
unsigned char LoadSettings(unsigned char* readerinfo).....	3
unsigned char SaveSettings(unsigned char* readerinfo).....	4
int Inventory(unsigned char filter).....	4
unsigned char GetResult(unsigned char *scanresult, int index).....	4
unsigned char GetTID(unsigned char* epc, unsigned char epclen, unsigned char* tid, unsigned char* tidlen).....	5
unsigned char SetPassword(unsigned char* epc, unsigned char epclen, unsigned char* pwd, unsigned char pwrlen).....	5
unsigned char SetKillPassword(unsigned char* epc, unsigned char epclen, unsigned char* kpwd, unsigned char kpwrklen).....	5
void LastError(char* error).....	5
unsigned char Auth(unsigned char* pwd, unsigned char pwrlen).....	6
unsigned char WriteMemWord(unsigned char* epc, unsigned char epclen, unsigned char* data, unsigned char windex).....	6
unsigned char ReadMemWord(unsigned char* epc, unsigned char epclen, unsigned char* data, unsigned char windex).....	6
unsigned char SetFilter(int maskAdrByte, int maskLenInByte, unsigned char* maskDataByte).....	6
int TagType().....	7
unsigned char TagName(char* name).....	7
unsigned char WriteEpcWord(unsigned char* epc, unsigned char epclen, unsigned char* epcword, unsigned char windex).....	7
unsigned char TagExists(unsigned char* epc, unsigned char epclen).....	8
Jence Uhf App.....	8
How to download and use the Application.....	8
Optimal Setup.....	15
Troubleshooting.....	17
1. Could not connect to PC.....	17
2. Could not detect type of card.....	17
3. Could not perform inventory.....	17
4. Not all tags are detected in the first scan.....	17
Version History.....	17
Version 1.3.....	17
Version 1.2.....	17
Version 1.1.....	17
Version 1.0.....	17

There is JAVA and C API. JAVA project contains example code to read card.

Go to **demo** directory, you will find J4210U.dll. This is the main DLL for accessing the reader. There are other DLL, which will also be needed. This is a 64-bit DLL. 32-bit is no longer supported.

This DLL can also be loaded in C# and Python using standard procedure.

Auto Detection Tags

HIGGS_3, HIGGS_4, HIGGS_EC, HIGGS_9, MONZA_4I, MONZA_4E, MONZA_4D, MONZA_4QT, MONZA_R6, UCODE_DNA, UCODE_7, EM4423. Auto detection enables internal knowledge of the card, therefore, programming a card become easier.



Demo

There is a demo folder which contains a card dump program. The program is written in Java and the source code is provided with detailed comment. The demo will run on Windows PC. In order to run on Linux and Mac OS X, open the program with Eclipse on respective platform and run from Eclipse.

The demo program is written using SWT API. This GUI API is fast and have a native look and feel. GUI written in SWT runs on multiple platform. Only the swt.jar file needs to be replaced with the platform specific version.

DLL Functions

Following DLL functions are available.

unsigned char AvailablePorts(char ports[256][8]);

Auto detects available serial ports and copies them into the two dimensional array.

Parameters:

ports: com port name array. Must be of size 256 x 8 or a single dimension array of 2048.

Example: ports[0] = "COM4", ports[1] = "COM7", etc

Returns: 1 on success.

unsigned char OpenPort(unsigned char* port);

Opens a COM port.

Parameters:

port: com port name. Example "COM4", etc

Returns: 1 on success.

void ClosePort();

Closes COM port that was opened. Otherwise does nothing.

unsigned char LoadSettings(unsigned char* readerinfo)

Loads settings into the array. The array size should be sufficient enough to hold the data. The data structure to hold the data is as follows:

```
struct ReaderInfo {
    // the following constants are reader constants
    // they cannot be changed.
    int Serial = 0;
    char VersionInfo[2] = {0,0};
    unsigned char Antenna = 0;
    unsigned char ComAdr;
    unsigned char ReaderType;

    // the following variables are reader parameters
    // which can be changed. Call SetSettings function
    // to change the settings.
    unsigned char Protocol;
    unsigned char Band;
    unsigned char Power;
    unsigned char ScanTime;
    unsigned char BeepOn;

    // unused
    unsigned char Reserved1;
    unsigned char Reserved2;

    // cannot be changed
    int MaxFreq = 0;
    int MinFreq = 0;
    int BaudRate = 0;
};
```

Parameters:

readerinfo: pointer to ReaderInfo structure.

Returns: 1 on success.

unsigned char SaveSettings(unsigned char* readerinfo)

Saves reader info. The ReaderInfo struct may be modified and passed to this function. Only the Protocol, Band, Power, ScanTime, BeepOn can change.

Parameters:

readerinfo: pointer to ReaderInfo structure.

Returns: 1 on success.

int Inventory(unsigned char filter)

Performs inventory with or without filter. The filter parameters may be set by calling Filter function prior to calling this function. If the return value is non-zero (positive), then the inventory items could be retried by using zero based index by calling GetResult function.

Parameters:

filter: 1 to perform filtering, otherwise pass 0.

Returns: Number of unique tags found. Returns 0, if no tags found.

unsigned char GetResult(unsigned char *scanresult, int index)

Gets the inventory item at index (zero based). If not found, returns 0. If found stores the scan data into scanresult array. ScanResult data structure is shown below.

```
struct ScanResult {  
    unsigned char ant; // 1 byte  
    char RSSI; // 1 byte  
    unsigned char count; // 1 byte  
    unsigned char epclen;  
    unsigned char epc[12]; // 12 byte or 62 by7e  
};
```

Parameters:

scanresult: pointer to the ScanResult structure.

index: index of the inventory item.

Returns: 1 on success.

unsigned char GetTID(unsigned char* epc, unsigned char epclen, unsigned char* tid, unsigned char* tidlen)

Gets TID for the tag with given EPC.

Parameters:

epc: EPC code of the tag for which TID is requested.

epclen: length of EPC.

tid: an array to hold the TID. Provide an array of sufficient size.

tidlen: actual number of bytes copied into the tid array.

Returns: 1 if successful.

unsigned char SetPassword(unsigned char* epc, unsigned char epclen, unsigned char* pwd, unsigned char pwrlen)

Sets password for the card with the given EPC.

Parameters:

epc: EPC code of the tag for which TID is requested.

epclen: length of EPC.

pwd: an array holding the password.

pwrlen: length of the password array. The length must be 4 bytes at least.

Returns: 1, if successful.

unsigned char SetKillPassword(unsigned char* epc, unsigned char epclen, unsigned char* kpwd, unsigned char kpwrklen)

Sets kill password for the card with the given EPC.

Parameters:

epc: EPC code of the tag for which TID is requested.

epclen: length of EPC.

kpwd: an array holding the password.

kpwrlen: length of the password array. The length must be 4 bytes at least.

Returns: 1, if successful.

void LastError(char* error)

Stores the last error.

Parameters:

error: an array to hold the error. Pass an array of sufficient length. 256 byte array recommended. The error is a C type string, terminated by a NULL character.

Returns: none.

unsigned char Auth(unsigned char* pwd, unsigned char pwrlen)

Sets the tag password to be used for all successive inventory and read/write operations.

Parameters:

pwd: an array containing password.

pwrlen: length of the password array. Minimum is 4 bytes (32 bits).

Returns: 1 if command was successful. If no card found, will return 0.

unsigned char WriteMemWord(unsigned char* epc, unsigned char epclen, unsigned char* data, unsigned char windex)

Writes the 2-byte word in the data to the Tag's User memory at word index windex.

Parameters:

epc: EPC of the tag.

epclen: number of bytes in EPC.

data: 2 byte array with data, MSB byte first followed by LSB byte.

windex: word index. Tags are indexed by word, 2 two bytes are written simultaneously.

Returns: 1, if successful.

unsigned char ReadMemWord(unsigned char* epc, unsigned char epclen, unsigned char* data, unsigned char windex)

Reads 2-byte word into data from Tag's User memory at word index windex.

Parameters:

epc: EPC of the tag.

epclen: number of bytes in EPC.

data: 2 byte array with data to be saved, MSB byte first followed by LSB byte.

windex: word index. Tags are indexed by word, 2 two bytes are written simultaneously.

Returns: 1, if successful.

unsigned char SetFilter(int maskAdrByte, int maskLenInByte, unsigned char* maskDataByte)

Sets Tag's EPC filtering to be used during inventory. Example: If there are tags with the following EPC:

1. ABCD1234FEDC5679
2. ABCD5679FEDC1234

3. FEDC1234ABCD5679

Then `SetFilter(0, 2, [0xAB, 0xCD])` will return tags #1 and #2 upon inventory. `SetFilter(2, 2, [0x12, 0x34])` will return tag #1 and #2. `SetFilter(6, 2, [0x56, 0x78])`. And if only a specific tag is required, the `SetFilter(0, 8, [0xAB, 0xCD, 0x56, 0x79, 0xFE, 0xDC, 0x12, 0x34])` will return only #2.

This operation is very useful to search inventory of products with know EPC prefix or suffix.

Parameters:

maskAdrByte: Number of bytes from the beginning of EPC code.

maskLenInByte: Number of bytes in the mask byte.

maskDataByte: mask bytes of length provided in second parameter.

Returns: 1, if successful.

int TagType()

Returns the Tag type, usually the chip used. The number returned is an index to an array of known tag chips. This operation must be called after calling `getTID()`. Because, TID contains the tag's manufacturer information for most chips.

Parameters: none.

Returns: a non zero index. If zero is returned, then tag chip could not be determined.

unsigned char TagName(char* name)

Stores the tag's name identified by index returned by `TagType()`.

Parameters:

name: an array to store the name. The array should at least be 64 byte. The array is C type NULL terminated string.

Returns: 1, if successful.

unsigned char WriteEpcWord(unsigned char* epc, unsigned char epclen, unsigned char* epcword, unsigned char windex)

Writes the 2-byte EPC word at word index `windex`. This function may be used if only a word is required to be changed in EPC.

Parameters:

epc: EPC of the tag.

epclen: number of bytes in EPC.

data: 2 byte array with EPC data, MSB byte first followed by LSB byte.

windex: word index. Tags are indexed by word, 2 two bytes are written simultaneously.

Returns: 1, if successful.

unsigned char TagExists(unsigned char* epc, unsigned char epclen)

Checks if the tag with the given EPC is found in the field.

Parameters:

epc: EPC of the tag.

epclen: number of bytes in EPC.

Returns: 1, if tag found.

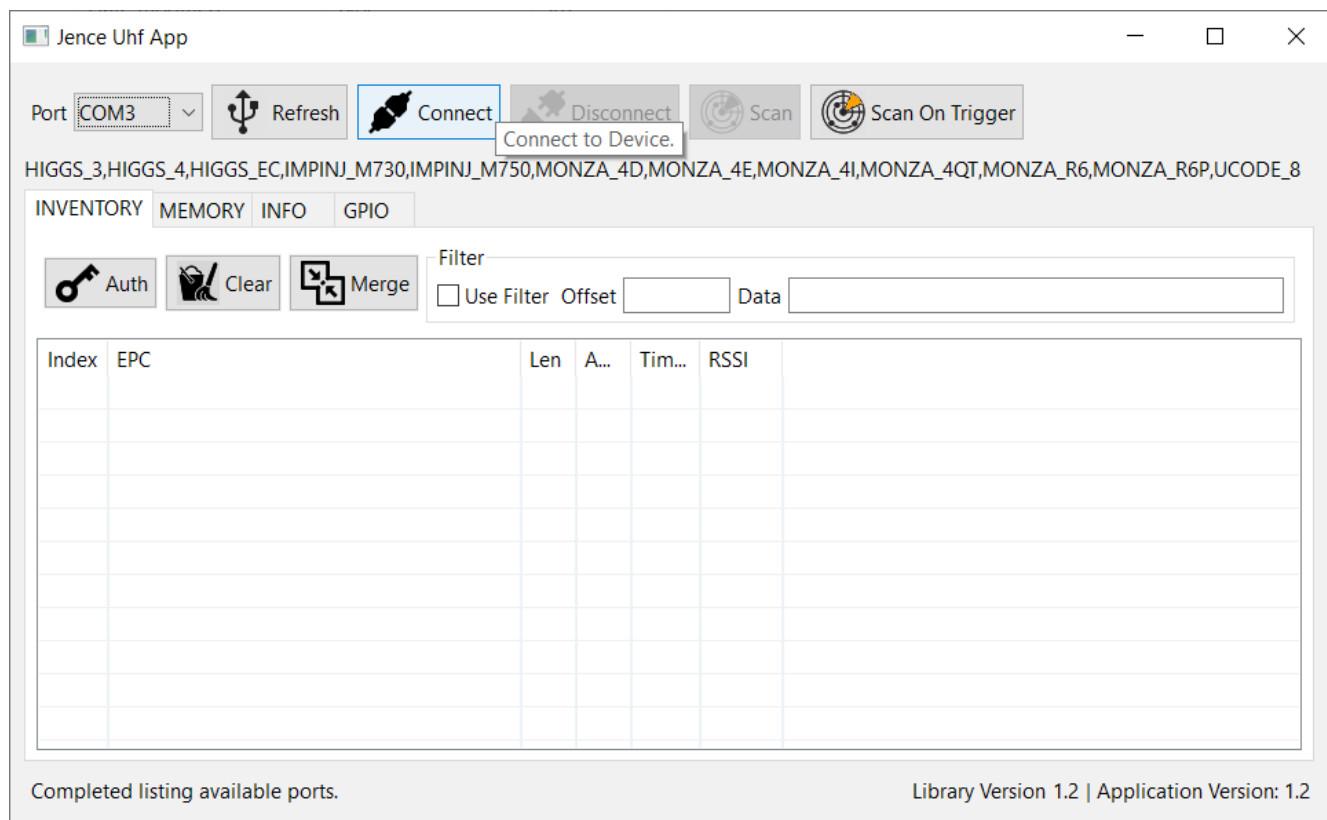
Jence Uhf App

A Java application called UhfApp is provided with full source code. This application uses SWT (Standard Widget Toolkit) library for all its GUI frontend. SWT uses underlying OS'es native API to display the widgets and therefore the widgets get a native look. In addition, due to native calls, all hardware accelerations are handled by the OS itself. The application can be easily modified to run on Swing, but this is unnecessary. SWT library is a single jar file which is available for all popular OSes (this includes Windows, Mac OSX, Linux, Raspbian, and more).

How to download and use the Application

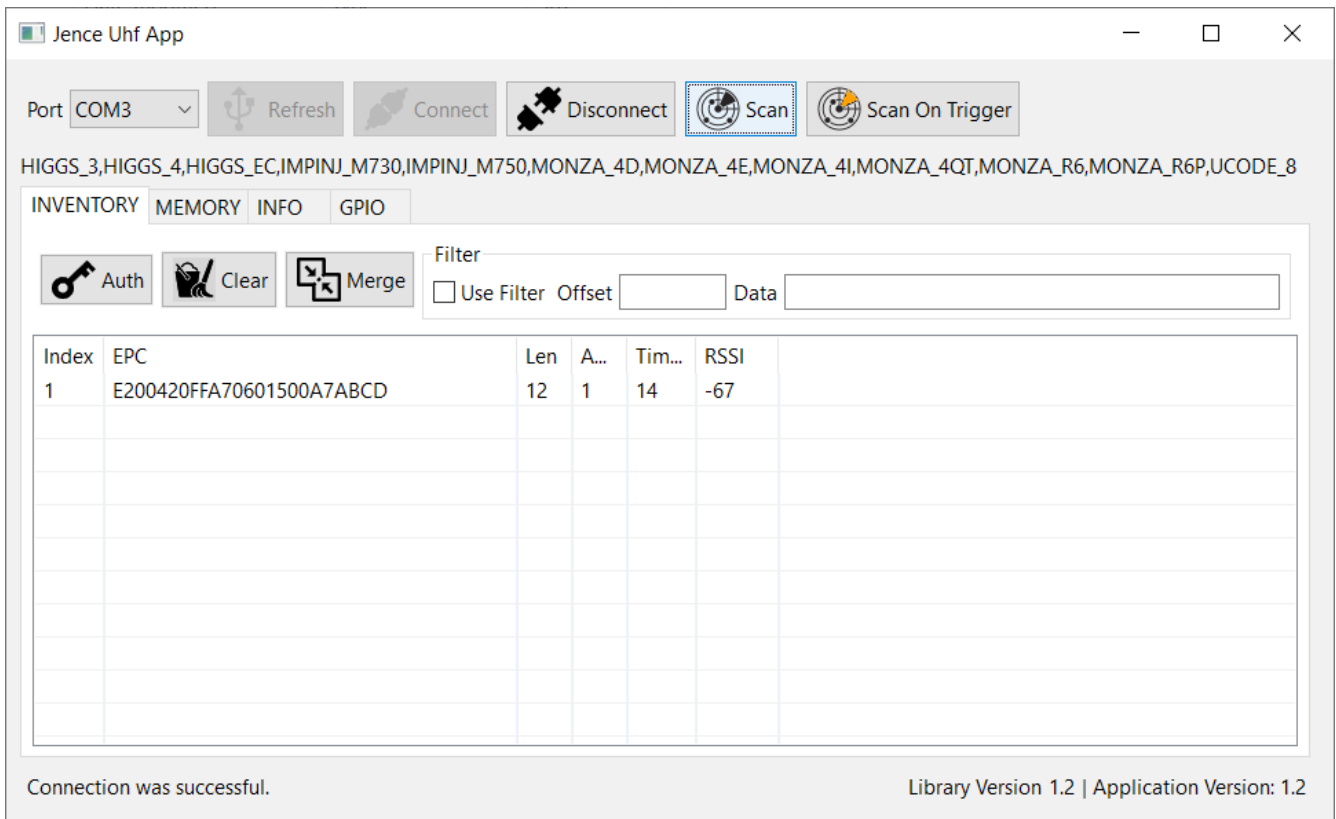
This software application tool associates with our two products UHF Desktop Reader/Writer hardware 4210U and UHF Handheld reader 4211U. At first, you have to download the application file from our official website: <http://jence.com/web/>. In the search option of the website, type UHF reader and you will see both of our products in the suggestion. Click on any of this to go to the product page. In the product page, after expanding the 'show more' in the bottom, you will find the link for downloading the SDK. Click on this link to download the zip file of the application.

After the download, please unzip the folder. Then go to the demo folder. Inside the demo folder, there is an application file j4210u.exe. If you run this application file, two command prompt window will appear. Now connect the UHF hardware with the computer through USB cable and find the com port inside the device manager. Inside the jence UHF App window at first click on the refresh button, then the com port will appear in the port option, select the com port and finally click on the connect button to connect the software tool with the hardware.

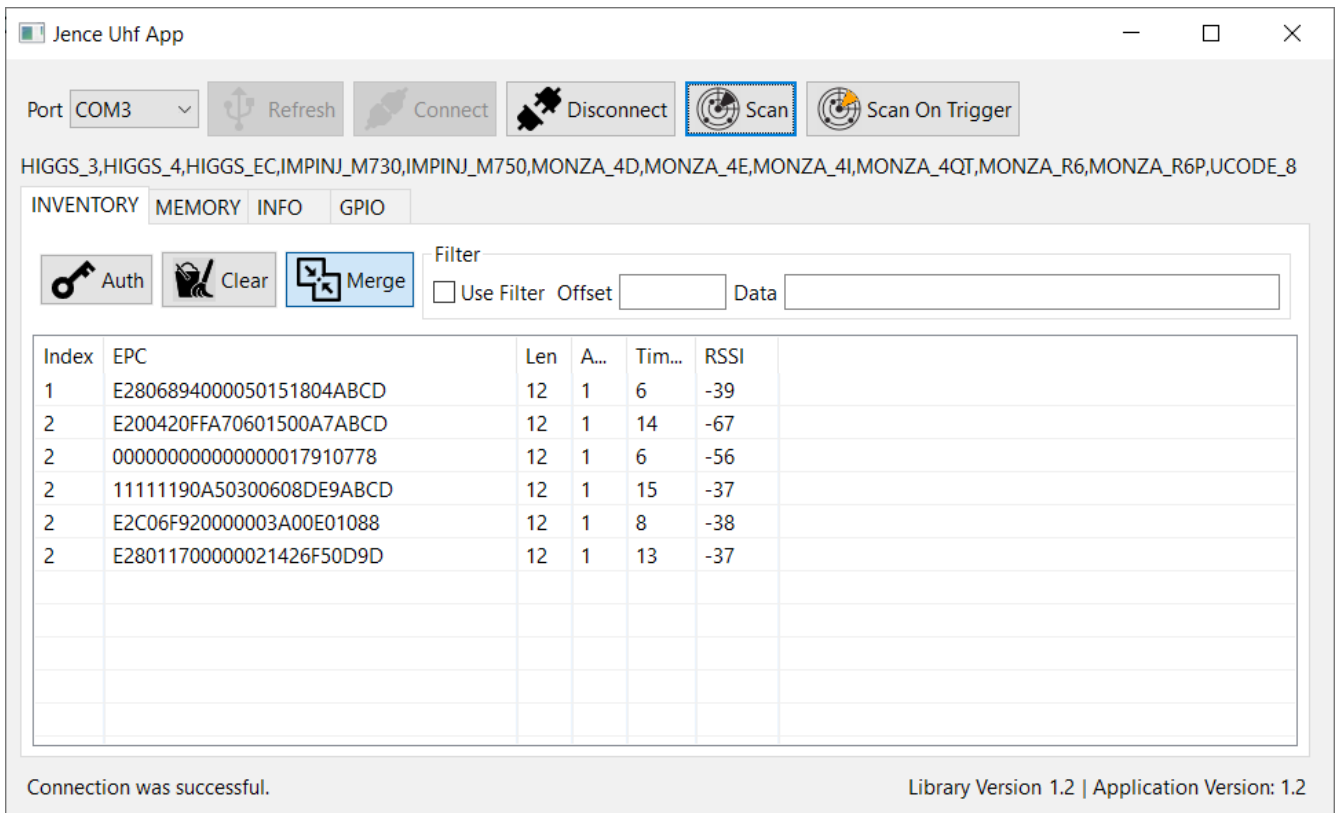


This UHF software tool can directly detect Ucode-8, Monza 4Q1, Monza R6P, Monza 4D, HIGGS 3, IMPINJ M730 and Monza R6 UHF RFID CHIPS. Apart from these chips, it can read other unknown chips as well. This software tool can detect several tags or cards at a time. If you use UHF Desktop hardware, in that case place UHF cards or tags on the hardware device and click on the Scan button inside the software. The software tool will scan the cards. If you use UHF Handheld reader, click on the Scan on Trigger button inside the software. Now if you press the button of the UHF Handheld reader, it will scan any UHF tags within the range of the reader.

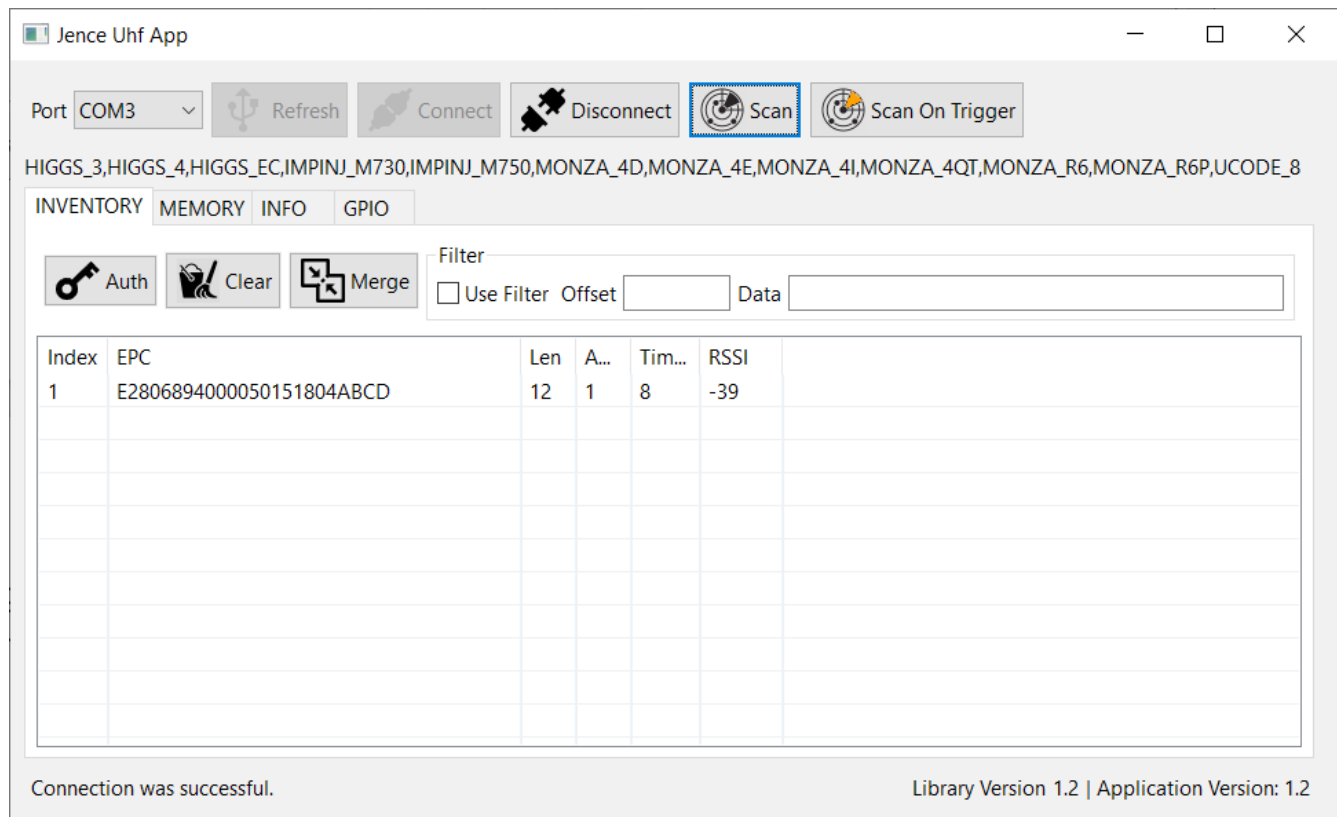
Inventory Tab: When the software application scans cards or tags, we will see the EPC (Electronic Product Code) of those cards inside the inventory tab.



In this tab, there is merge option. If we click on this button, it will show the previous card information along with the new card information while scanning the new card.



In non-merge mode (when we unselect the merge option), each time we scan new card, we can only see that cards' information. Information of the previous cards are removed from the software during each new scan.



There is a filter option inside the inventory tab. If we want to search tag with similar value, such as similar suffix or prefix, we can use this option. At first, we have to give the offset value. We should consider even offset. To elaborate: If we want to search tags with ABCD suffix in it, let's assume these are the EPC no of two tags with ABCD in it.

E280	6894	0000	5015	1804	ABCD	– EPC No						
0	1	2	3	4	5	6	7	8	9	10	11	– offset value

E200	420F	FA70	6015	00A7	ABCD	– EPC NO						
0	1	2	3	4	5	6	7	8	9	10	11	– offset value

Here, 0,1,2,3,4,5,6,7,8,9,10,11 are offset values. Offset value must be 2 byte which means 4 hex values. As we should always consider only even offset and as ABCD value is in 10 11 offset where 10 is the even offset, we will write 10 in offset option and we will write ABCD in data option. We must check Use Filter option. Now if we search with UHF Handheld reader,

Jence Uhf App

Port: COM3 [Refresh] [Connect] [Disconnect] [Scan] [Scan On Trigger]

HIGGS_3,HIGGS_4,HIGGS_EC,IMPINJ_M730,IMPINJ_M750,MONZA_4D,MONZA_4E,MONZA_4I,MONZA_4QT,MONZA_R6,MONZA_R6P,UCODE_8

INVENTORY MEMORY INFO GPIO

[Auth] [Clear] [Merge] Filter: ☒ Use Filter Offset: 10 Data: ABCD

Index	EPC	Len	A...	Tim...	RSSI
1	E200420FFA70601500A7ABCD	12	1	3	-62
2	E2806894000050151804ABCD	12	1	12	-53
3	11111190A50300608DE9ABCD	12	1	4	-63
4	E2801170000002150E64ABCD	12	1	20	-44

Filter is used during scan. Library Version 1.2 | Application Version: 1.2

we will find tags with this ABCD suffix. It is very useful to find tags with similar prefix, suffix or values in large inventories.

Memory tab: Now if we double click on the EPC no while card is within the range of UHF hardware and go to the memory tab, we will see the inside content of the card, such as chip type, total memory, PWD size, EPC size, TID size and user size. The numbers marked with yellow colour are in bit format and others are in byte format.

Jence Uhf App

Port: COM3 [Refresh] [Connect] [Disconnect] [Scan] [Scan On Trigger]

HIGGS_3,HIGGS_4,HIGGS_EC,IMPINJ_M730,IMPINJ_M750,MONZA_4D,MONZA_4E,MONZA_4I,MONZA_4QT,MONZA_R6,MONZA_R6P,UCODE_8

INVENTORY MEMORY INFO GPIO

Operations: [Refresh] [Write] [Clean] [Auth] [Exists]

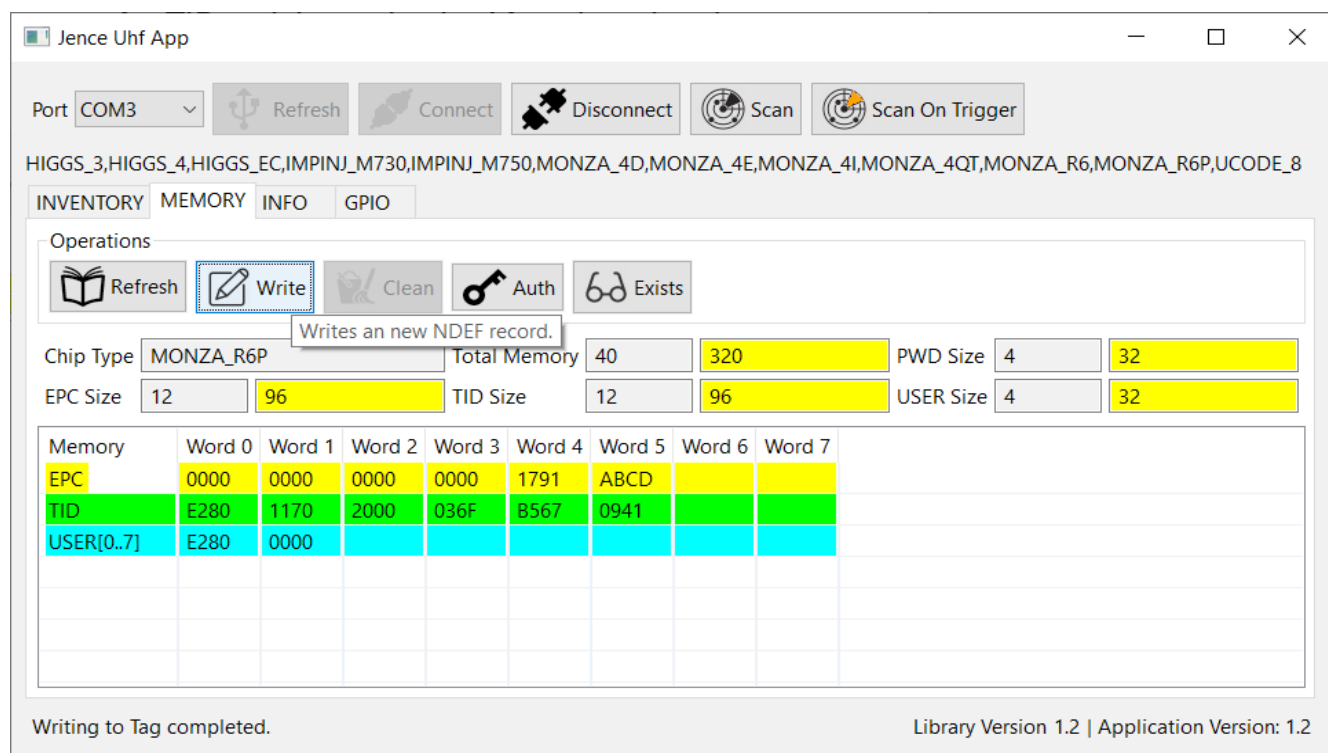
Chip Type: MONZA_R6P Total Memory: 40 320 PWD Size: 4 32

EPC Size: 12 96 TID Size: 12 96 Total Memory in Bits

Memory	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
EPC	0000	0000	0000	0000	1791	0778		
TID	E280	1170	2000	036F	B567	0941		
USER[0..7]	E280	0000						

Memory Load completed. Library Version 1.2 | Application Version: 1.2

We can alter the EPC and user data here except for TID as it is read only. After changing the data, we have to click on Write button to save the data.



Port: COM3

HIGGS_3,HIGGS_4,HIGGS_EC,IMPINJ_M730,IMPINJ_M750,MONZA_4D,MONZA_4E,MONZA_4I,MONZA_4QT,MONZA_R6,MONZA_R6P,UCODE_8

INVENTORY MEMORY INFO GPIO

Operations: Refresh, **Write** (Writes an new NDEF record.), Clean, Auth, Exists

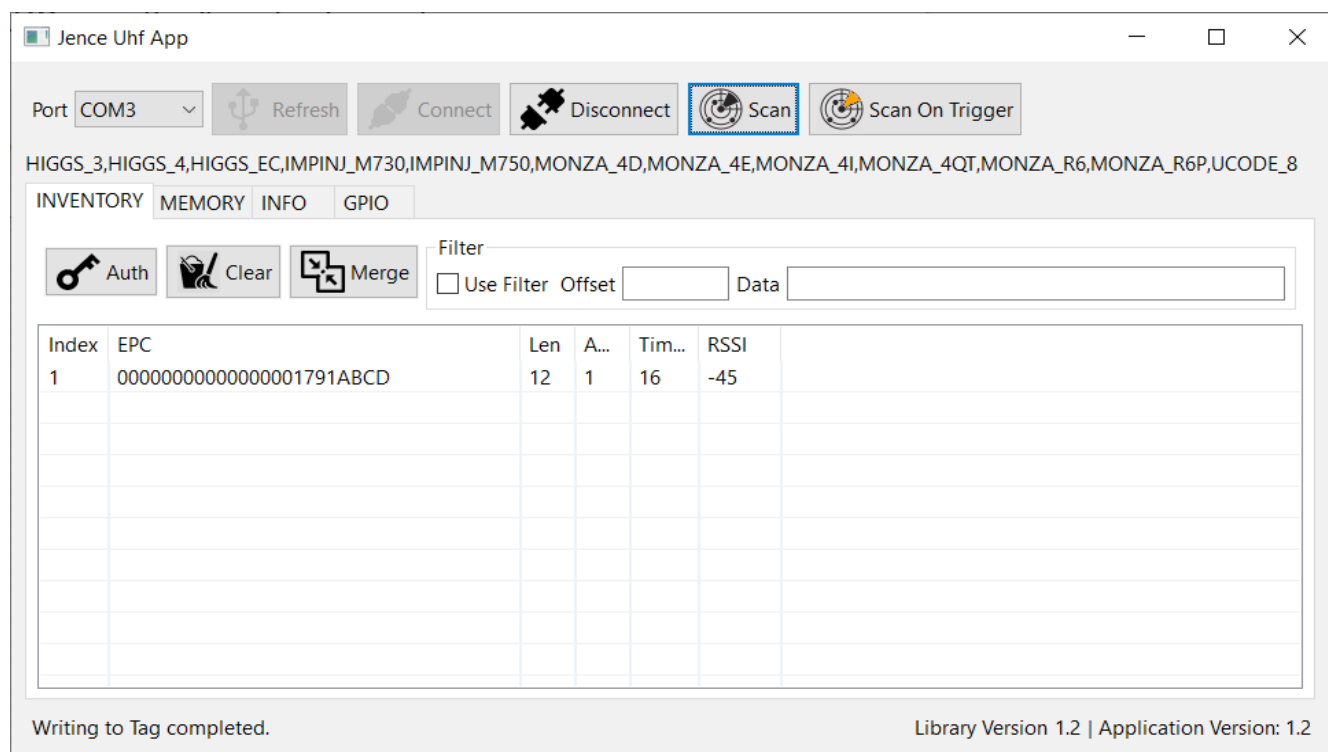
Chip Type: MONZA_R6P Total Memory: 40 PWD Size: 4

EPC Size: 12 TID Size: 12 USER Size: 4

Memory	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
EPC	0000	0000	0000	0000	1791	ABCD		
TID	E280	1170	2000	036F	B567	0941		
USER[0..7]	E280	0000						

Writing to Tag completed. Library Version 1.2 | Application Version: 1.2

Now if we scan the card, we will see the EPC no with altered value in it.



Port: COM3

HIGGS_3,HIGGS_4,HIGGS_EC,IMPINJ_M730,IMPINJ_M750,MONZA_4D,MONZA_4E,MONZA_4I,MONZA_4QT,MONZA_R6,MONZA_R6P,UCODE_8

INVENTORY MEMORY INFO GPIO

Auth, Clear, Merge

Filter: ☐ Use Filter Offset: Data:

Index	EPC	Len	A...	Tim...	RSSI
1	00000000000000001791ABCD	12	1	16	-45

Writing to Tag completed. Library Version 1.2 | Application Version: 1.2

In this way, we can give similar suffix or prefix or middle value to several tags to search them promptly. There is an Exists button inside the memory tab.

Jence Uhf App

Port: COM3 | Refresh | Connect | Disconnect | Scan | Scan On Trigger

HIGGS_3,HIGGS_4,HIGGS_EC,IMPINJ_M730,IMPINJ_M750,MONZA_4D,MONZA_4E,MONZA_4I,MONZA_4QT,MONZA_R6,MONZA_R6P,UCODE_8

INVENTORY | **MEMORY** | INFO | GPIO

Operations: Refresh | Write | Clean | Auth | **Exists**

Chip Type: MONZA_R6P | Total Memory: 40 | 320 | PWD Size: 4 | 32

EPC Size: 12 | 96 | TID Size: 12 | 96 | USER Size: 4 | 32

Memory	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
EPC	0000	0000	0000	0000	1791	ABCD		
TID	E280	1170	2000	036F	B567	0941		
USER[0..7]	E280	0000						

Writing to Tag completed. | Library Version 1.2 | Application Version: 1.2

If tag is in the inventory and within the range of UHF Handheld reader and if we click on this button, it will show the pop up message “Tag found in the inventory”.

Tag FOUND in the inventory

OK

If tag is not in the inventory or not within the range of UHF reader, it will show the message “Tag not found near the reader”.

Tag NOT FOUND near the reader.

OK

Info tab: In this info tab, we will get the information of the UHF reader, such as its serial no, max frequency, version, min frequency, baud rate, scan time etc. There is Beep on option inside inside the tab. If we uncheck this option during scan, reader will not produce any sound and if we check it during scan, reader will produce beep sound. During check and uncheck, we must press on Write button to save the setting.

The screenshot shows the 'Jence Uhf App' window. At the top, there's a port selection dropdown set to 'COM3' and several action buttons: 'Refresh', 'Connect', 'Disconnect', 'Scan', and 'Scan On Trigger'. Below these is a list of detected tags: HIGGS_3, HIGGS_4, HIGGS_EC, IMPINJ_M730, IMPINJ_M750, MONZA_4D, MONZA_4E, MONZA_4I, MONZA_4QT, MONZA_R6, MONZA_R6P, and UCODE_8. A tabbed interface shows 'INVENTORY' as the active tab, with other tabs being 'MEMORY', 'INFO', and 'GPIO'. The main area contains two columns of settings:

Serial No	276828705	Version	3.30
Max Freq	927250	Min Freq	902750
Antenna	1	ComAdr	0
Reader Type	16	Protocol	108
Band	USA	Baud Rate	57600
Power (dB)	25	Scan Time (ms)	300

Below the settings, there is a 'Beep ON' checkbox which is checked. At the bottom left is a 'Write' button with a pencil icon. The status bar at the bottom indicates 'Settings loaded.' on the left and 'Library Version 1.2 | Application Version: 1.2' on the right.

Optimal Setup

The reason why some tags are read and others not is due to the RSSI (tag sensitivity). If you put tags one over the other, the antenna of the bottom tag obstructs the antenna of the top tags. Therefore, to always read well, spread the tags so the antennas do not block. In addition, the environmental issue plays a role due to the fact that the tags do not go through the same reflection path than the previous scan because of collision avoidance protocol. Typically, RFID reader software is designed to read continuously to read all surrounding tags. Therefore, you can press the scan button continuously or modify the java code so the scan button scans more than one time. You may also increase the Scan Time to allow more time to read tags.

This is a standard behavior of any UHF RFID reader in which the first iteration may not find all surrounding tags, but in the second and successive iteration it will find the tags. This is the reason why there is Merge button in the demo software. This keeps record of the previously detected tag.

COLLISION

UHF RFID have this behavior due to Collision in which after a Collision the tag stops transmitting for a short moment. This is not a problem with the device, instead this is how the UHF RFID protocol work. The rule is, if the tag could not be found in the first iteration, it will be found in successive iterations. Eventually all surrounding tags would be found.

In addition, the tag itself could be the cause of the problem. In your case, it does not seem that the tag is an issue because it is found eventually.

Use the Merge button. Right now the Scan button only scan once. But this is a demo software, so we didn't use multiple scans with a single button press. In a professionally coded software,

scans should be done multiple times. You can modify the software to scan several time with single press or just use the Merge option.

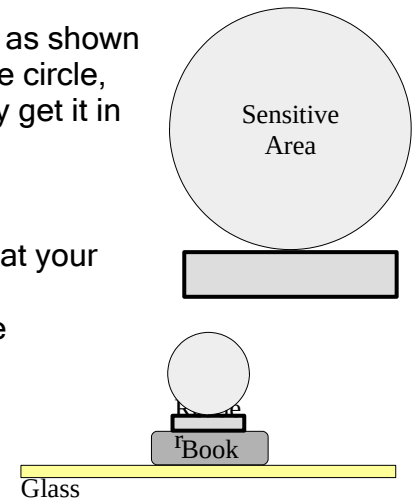
With that said, we will add option in our future demo that will have the option to set number of scans per Scan button press. Our next software release will have this feature.

ANTENNA LOBE

With the reader on the table, draw a 1 meter diameter circle reader as shown below. This is the sensitive area of the reader. If the tag is within the circle, most likely the reader will catch it in the first scan. Otherwise, it may get it in successive scan or not at all if it is way out of the cone.

LONG RANGE

If you need long range, use tags with bigger antenna. Make sure that your desk is not made of Glass. Glass is a high dielectric material which significantly affects the reader's impedance matching as well as the tag's impedance. Having Glass in close proximity will reduce the read range. If you have Glass desk, use a 1" thick book to place your reader and tags. This is shown below.



Troubleshooting

1. Could not connect to PC.

A. Make sure that the COM port number (for Windows) or TTY (for Linux and Mac OSX) appear when the device is connected. If the serial port is not recognized, then uninstall and then reinstall the driver. In some cases, you may need to reboot the PC after driver installation.

2. Could not detect type of card.

A. If card type is not detected, then the card may be password protected. In this case, the user may programmatically set the card type.

3. Could not perform inventory.

A. Possibly there were no tag on the device. If there are tags, make sure that they are UHF RFID tags. If still could not detect tag, then move the tags to the center of the reader then try again.

4. Not all tags are detected in the first scan.

A. Read through Optimal Setup for detailed discussion on this topic.

Version History

Version 1.5

MAC OSX GUI modification and .app creation.

Version 1.4

MAC OSX support implemented completely.

Version 1.3

GetSettings function bug fix. Initial coding of Mac OSX driver.

Version 1.2

Thingmagic Nano integration.

Version 1.1

Auto detection support added to MONZA_4QT, UCODE_8, MONZA_R6, MONZA_R6P, MONZA_4I, IMPINJ_M730, IMPINJ_M750.

Version 1.0

Initial version. Auto detection support added to HIGGS_3, HIGGS_4, HIGGS_EC, MONZA_4D, MONZA_4E, ICODE_7, ICODE_DNA, EM4423.

For questions, contact Jence.

JENCE

<http://www.jence.com>

Email: jence@jence.com