J4210U API Documentation

Revision Date: 2023-07-19

Table of Contents

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.

USB Serial Device Driver

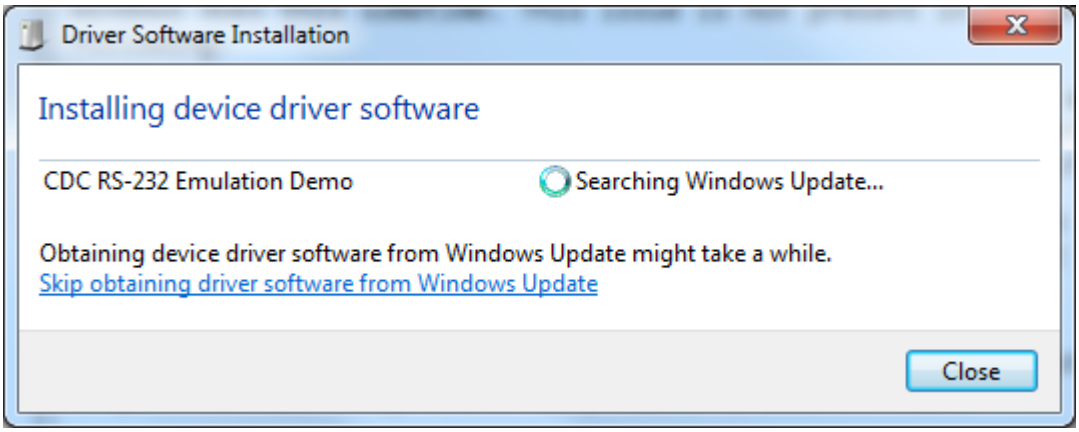
Depending on the version of operating system, USB Serial Device Driver may or may not be required. For Windows, the latest OS already come with most popular drivers. But if not fund can be downloaded from this page https://www.microchip.com/en-us/product/MCP2221. For Mac OS X (Aarch64), no driver is required. For Mac OS X Intel, driver may or may not be required. For most Linux, no driver is required. If the device could not be recognized by the OS, then driver can be installed from the above link. Currently, only 64-bit operating systems are supported as the demand for 32-bit OSes are shrinking.

Windows Driver Installation if not Automatically Detected

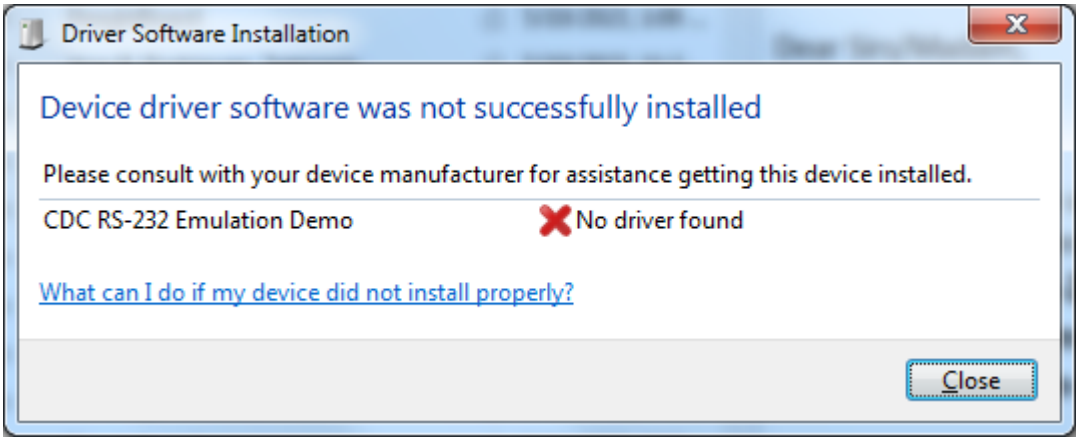
If, after connecting, J4210U to your PC's USB port, Windows could not determine the Serial Device, then do as follows:

1. You need to uninstall the driver. Go to Device Manager, and go to the list of COM ports. Choose the port that you have assigned for the J4210U and right click on it. Check the delete the driver checkbox also. Uninstall the driver. Remove the J4210U if it is connected to your PC.

2. Now connect the J4210U to the USB port again. Windows will try to reinstall the driver automatically. This will take a while. You will see an icon at the bottom right rolling.

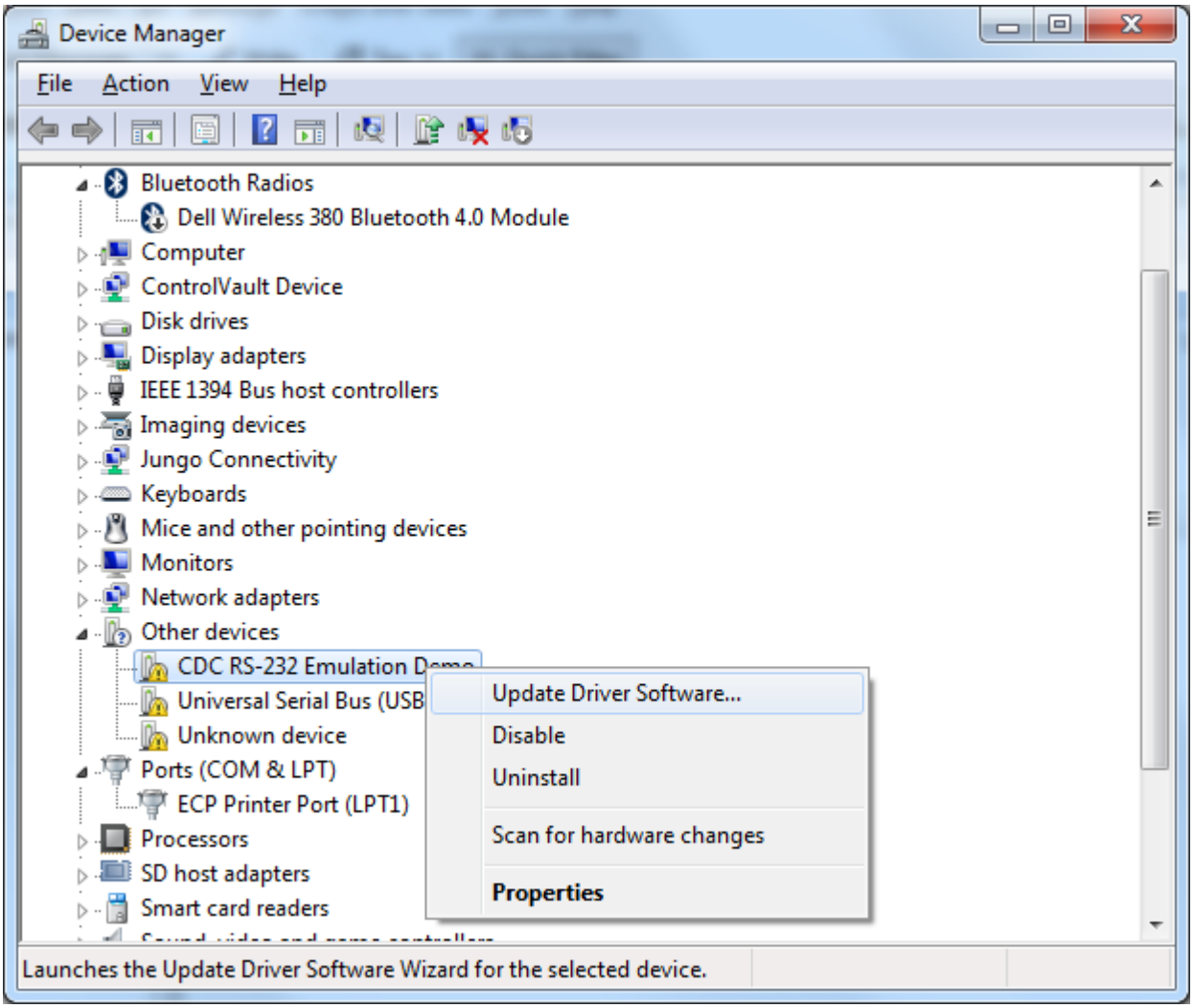


3. If this step finds the driver then try the app again. If the App works, then you are done and skip the rest of the steps.

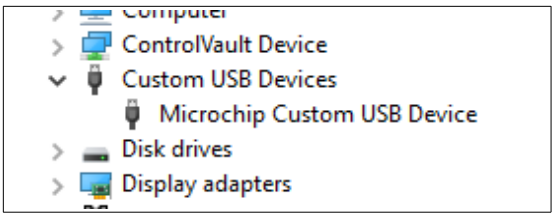


4. If this step fails then follow the steps below.

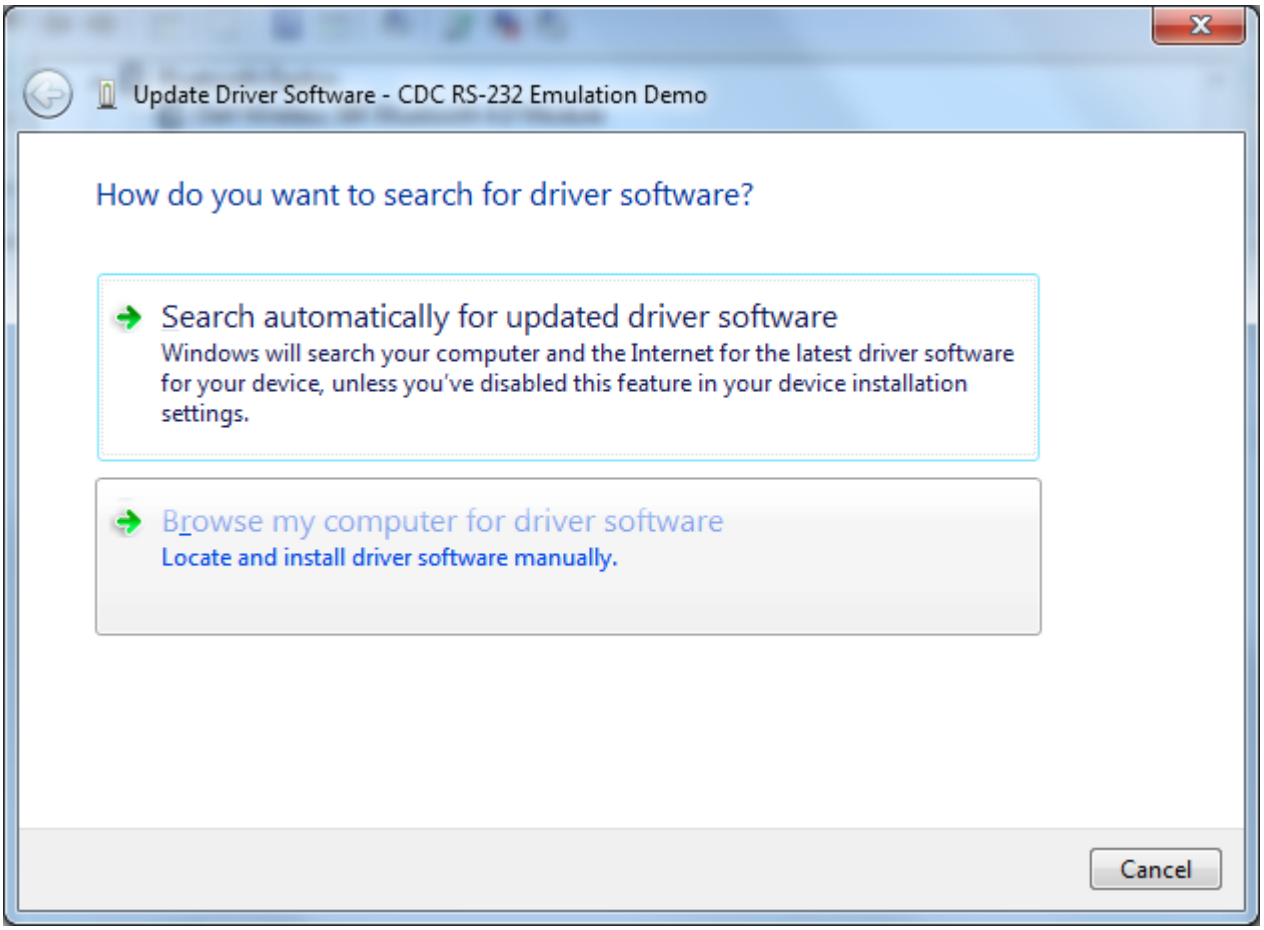
5. Right click on the CDC RS-232 Emulation Demo as shown and click on Update driver.



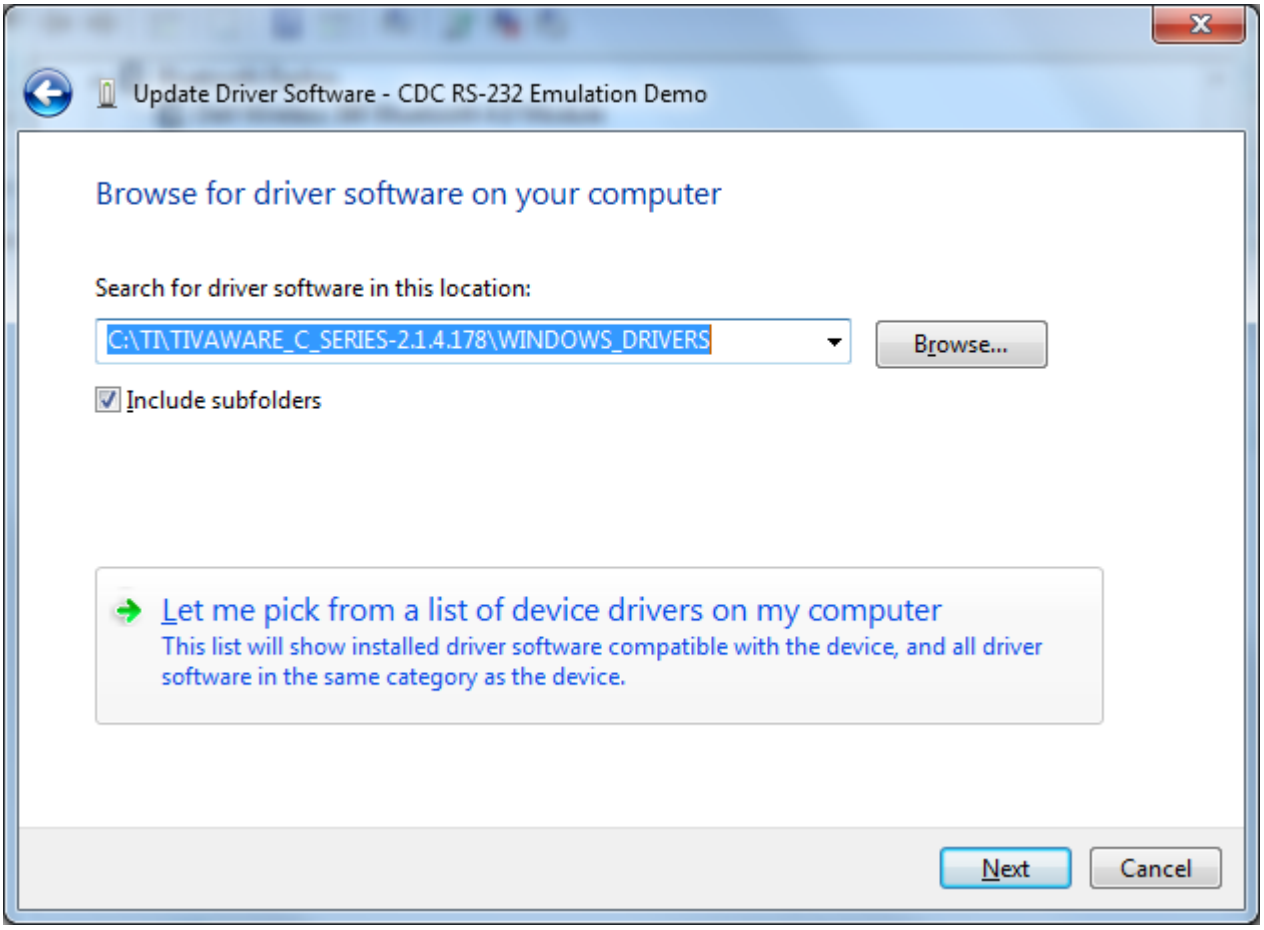
Or, if you get the following Device Information, where upon connecting the reader, a new Microchip USB Device is shown, then right click on the item and click on the Update Driver Software.

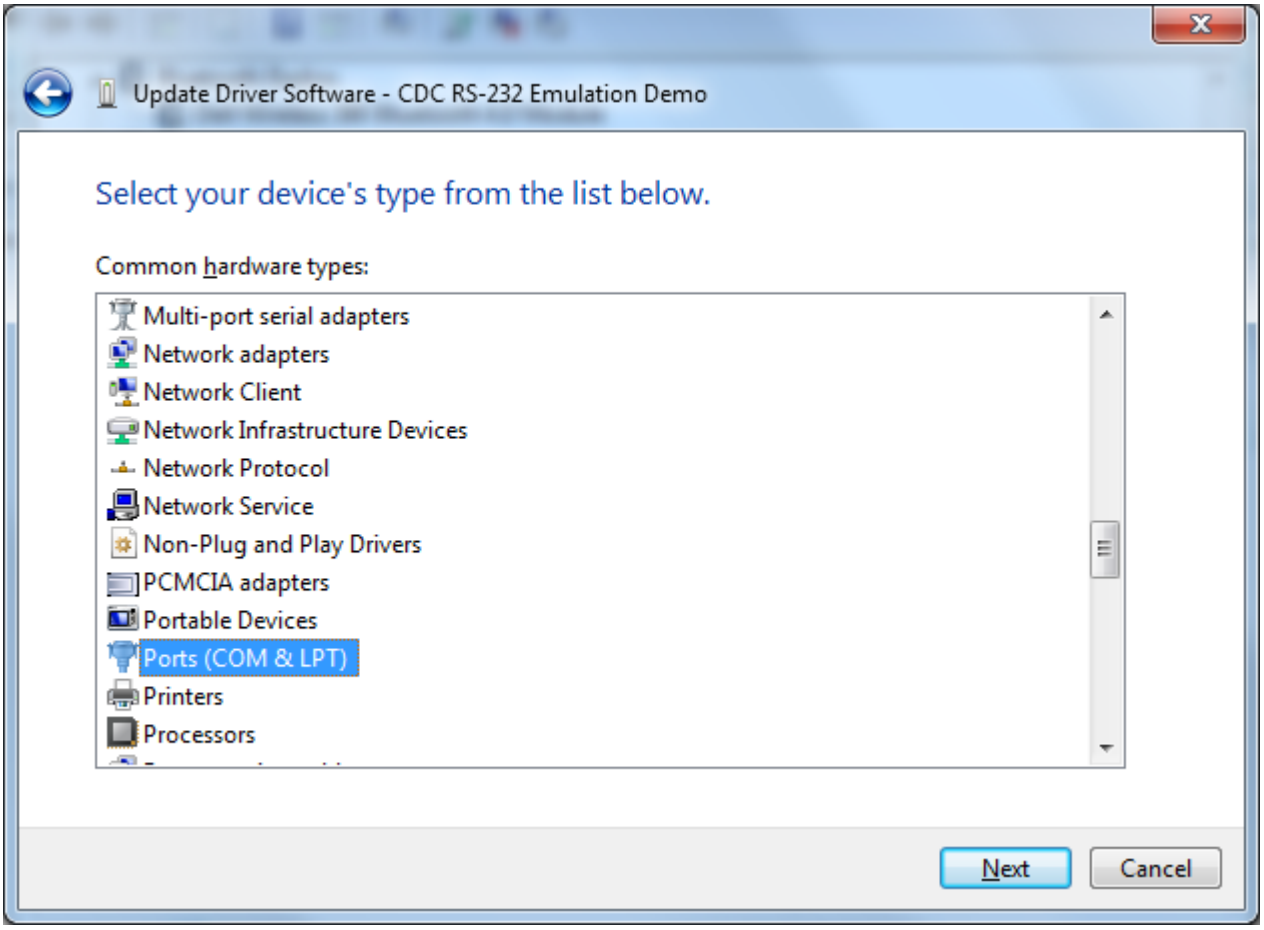


6. Choose Browse my computer for driver software.

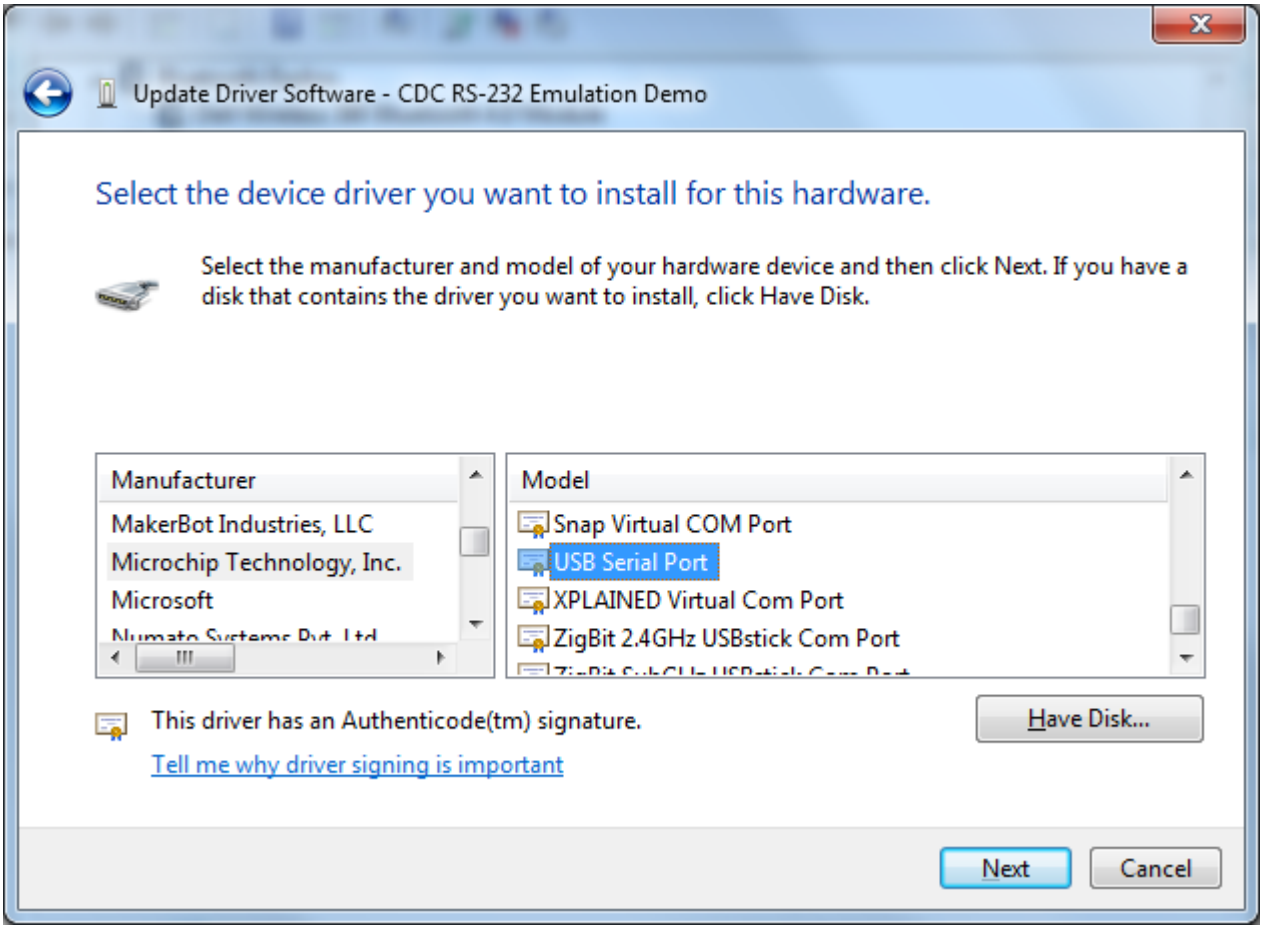


7. Choose let me pick from a list of device drivers on my computer. Choose Ports (COM & LPT). Click Next.

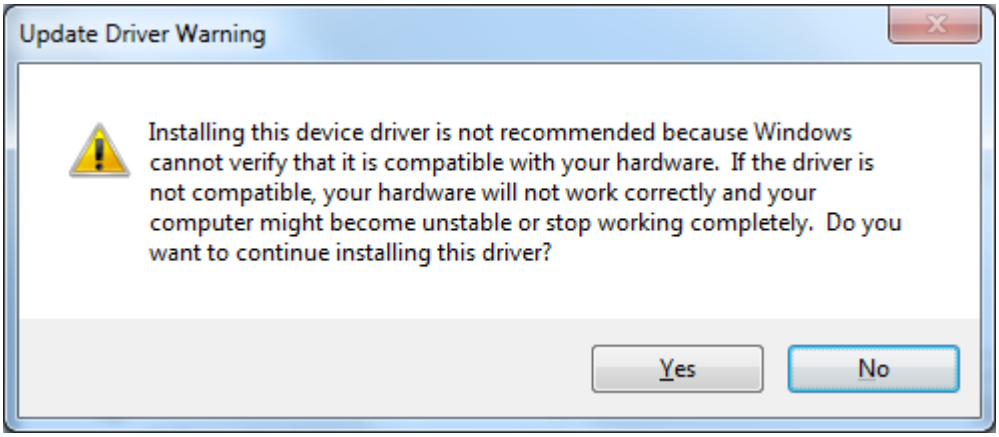

Update Driver Software - CDC RS-232 Emulation Demo
Browse for driver software on your computer
Search for driver software in this location:
C:\TI\TIVAWARE_C_SERIES-2.1.4.178\WINDOWS_DRIVERS
Browse...
Include subfolders
Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.
Next
Cancel



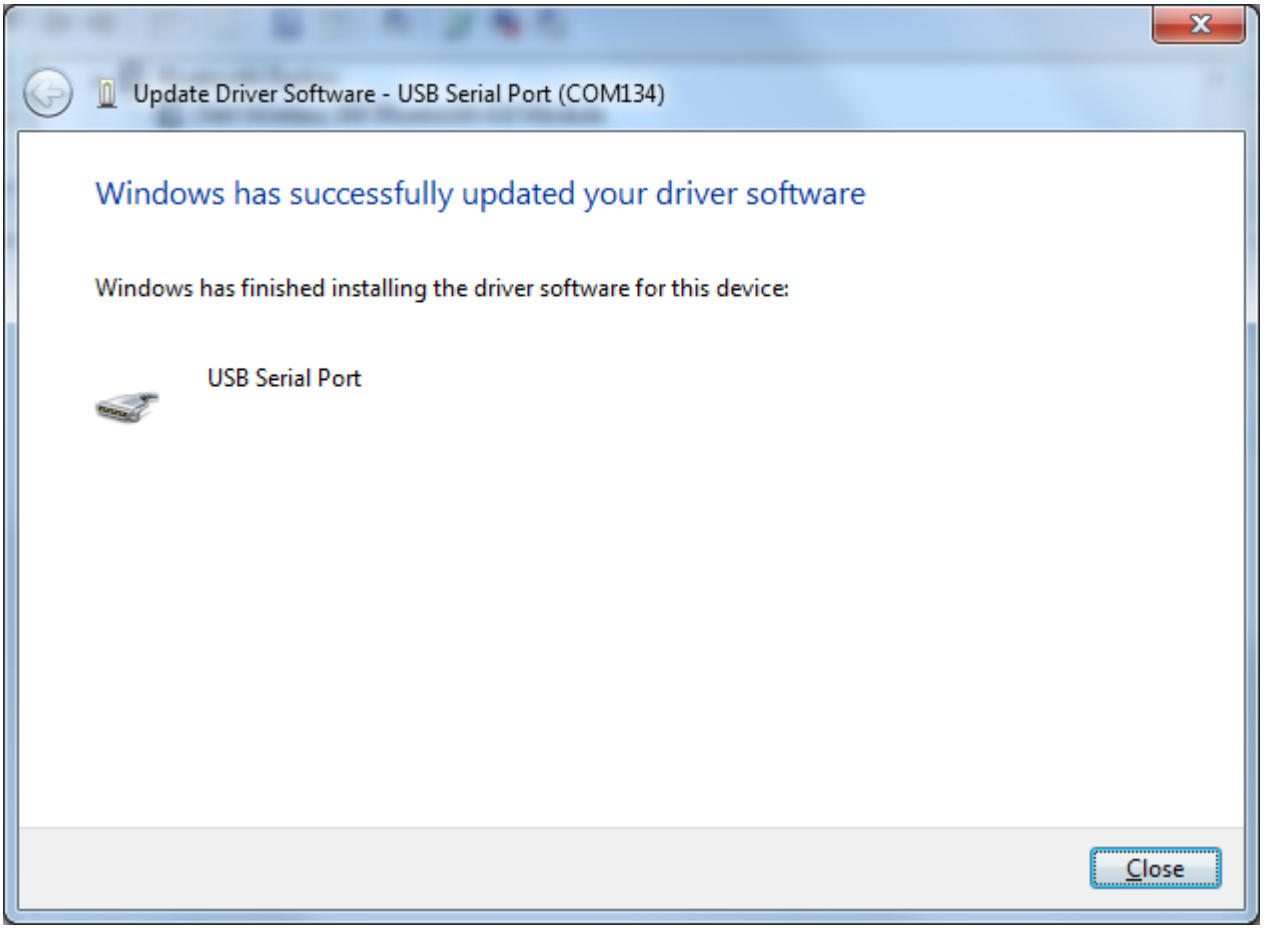
8. From the list choose Microchip Technologies Inc and Select USB Serial Port.



9. Install the driver anyways by click on Yes.



10. Click Close



You should have your COM port working again. Try your app again and see this fixes the problem.

Linux Driver Installation if not Automatically Detected

Typically, upon connecting the UHF Reader to the USB port, a tty port will appear which looks like ttyACM* in the /dev folder. Connect your UHF Reader to your Linux PC's USB port and do the following:

ls -alF /dev/ttyACM*

this should show you all the USB Serial Ports available. If none is found, you can force the cdc_acm to fire as follows:

```
sudo rmmod cdc_acm
sudo modprobe cdc_acm
```

```
sudo lsmod | grep cdc
sudo dmesg | grep ttyACM
ls /dev/ttyACM*
```

This step should fix the ACM issue and you should see a new tty port attached to the Linux PC.

Setup

Here is detailed description on how to setup other Operating Systems. In the following description we have shown how to install other software, like mysql database server, git-cola, tomcat, phpmyadmin, codeblocks, etc. These are not strictly necessary. But, if you are planning to build java based Servlet which could access database, you need both tomcat and mysql. If can skip installation of these software for now but install them anytime in the future.

Windows

Simply run the j4210u.exe. You must have Java 1.8 or higher installed in your PC. If you do not have Java, you may get it for free from https://www.java.com/en/download/.

Mac OS X

There is an App named UhfApp in the macosx platform folder. You may run the application. But you must have Java 1.8 or higher installed in you Mac. To get Java, visit https://www.oracle.com/java/technologies/downloads/ and follow instructions. Unless you are a developer, it is suggested that you install DMG instead of compressed archive. If you have Intel Mac OS X, use the app in the macosx[intel] folder.
If you get the following dialog box:

Go to Privacy & Security option from the System Settings and a click on Open Anyway.

Linux

First test if the current baudrate is 57600. Type the following:

```
stty -F /dev/ttyACM0
```

prefix with sudo, if the command could not be executed by the user. If the baudrate is not 57600, then type:

```
stty -F /dev/ttyACM0 57600
```

Just run the j4210u.sh file in a terminal. If it does not run, make it an executable by typeing the following and then press enter:

```
chmod 744 j4210u.sh
```

You can add the above stty command in the j4210u.sh so you do not need to type the command every time you use your PC. You must have java installed. To install java, follow the same steps described in Raspberry PI section below.

Raspberry PI

Raspberry PI Rasbian image 64-bit may be downloaded from the following link
https://www.raspberrypi.com/software/operating-systems/
Choose a 64-bit image.

Once the image is downloaded, burn the image in a micro SD card, then boot. For details on how to burn an image into SD card visit elsewhere. Boot the Banana PI from the image. If the image successfully boots, you will see the Raspbian desktop with background image and the top line containing Raspbian icon and following icons: browser, folder, terminal. Click on the Terminal icon to open a terminal window. NOTE: The default username is **pi** and password is **raspberry**

Installation

We will install the following for software development. First, you need to install Gparted to resize the partition to atleast 8GB.

```
sudo apt-get update
sudo apt-get install gparted
```

Run Gparted and resize the partition to 8GB or higher. Then execute all the following commands.

Download default Java Development Kit (JDK). There should be java already installed. Check if java is installed by typing:

```
java -version
```

This should show you the version of that is installed, otherwise and error message will be displayed if no java found.

```
sudo apt install default-jdk
```

Install Eclipse
```
sudo apt-get install eclipse
```

If the above command fails, install Eclipse manually as follows:

```
wget https://www.eclipse.org/downloads/download.php?file=/technology/epp/
downloads/release/2020-12/R/eclipse-java-2020-12-R-linux-gtk-aarch64.tar.gz
```

Upzip the .tar.gz file and the eclipse directory will contain the eclipse executable. Just double click the file and unzip the file in default directory.

Install Mysql Server. Maria DB is open source version of Mysql.
```
sudo apt-get install mariadb-server
```

Install CodeBlocks (For C/C++ development only)
```
sudo apt-get install codeblocks
```

Install Tomcat Server (For Java server development only)
```
sudo apt-get install tomcat9
```

Install PhpMyAdmin (For database development only)
```
sudo apt-get install phpmyadmin
```

Change Mysql root password from blank to something else
```
sudo mysql -u root -p
```
Password is blank, simply press enter.
In mysql command prompt type the following:
```
use mysql;
create user 'soacom'@'localhost' identified by 'soacom';
grant all on *.* to soacom@localhost;
```
{The following line with change the root password. Default password is blank}
```
alter user ‘root’@’localhost’ identified by ‘soacom’;
```
Note that we created a user **soacom** and set the password to **soacom**.

Open browser, type : `http://localhost/phpmyadmin`
User: `soacom`
Password: `soacom`
{The default phpmyadmin installation does not allow root access.}

If you use Git as source control, install Git Cola, a graphical Git utility.
```
sudo apt-get install git-cola
```

Now verify Processor Architecture:
```
getconf LONG_BIT
```

Get OS Info:
```
cat /etc/os-release
```

Get System Info:
```
cat /proc/cpuinfo
uname -m
```

Banana PI

Banana PI image may be downloaded from the following link
https://wiki.banana-pi.org/Banana_Pi_BPI-M4
This is a Raspbian image.

Other Banana PI images can be found from here:
https://download.banana-pi.dev/d/ca025d76afd448aabc63/?p=%2FImages%2FBPI-M4%2Flinux&mode=list
Any image can be used and the setup procedure is the same.

Once the image is downloaded, burn the image in a micro SD card, then boot. For details on how to burn an image into SD card visit elsewhere. Boot the Banana PI from the image. If the image successfully boots, you will see the Raspbian desktop with background image and the top line containing Raspbian icon and following icons: browser, folder, terminal. Click on the Terminal icon to open a terminal window. NOTE: The default username is **pi** and password is **raspberry**

Installation

We will install the following for software development. First, you need to install Gparted to resize the partition to atleast 8GB.

```
sudo apt-get update
sudo apt-get install gparted
```

Run Gparted and resize the partition to 8GB or higher. Then execute all the following commands.

Download default Java Development Kit (JDK)
```
sudo apt install default-jdk
```

Install Eclipse
```
sudo apt-get install eclipse
```

Install Mysql Server
```
sudo apt-get install mysql-server
```

Install CodeBlocks (For C/C++ development only)
```
sudo apt-get install codeblocks
```

Install Tomcat Server (For Java server development only)
```
sudo apt-get install tomcat8
```

Install PhpMyAdmin (For database development only)
```
sudo apt-get install phpmyadmin
```

Change Mysql root password from blank to something else
```
sudo mysql -u root -p
```

Password is blank, simply press enter.
In mysql command prompt type the following:

```
use mysql;
create user 'soacom'@'localhost' identified by 'soacom';
grant all on *.* to soacom@localhost;
```

{The following line with change the root password. Default password is blank}

```
alter user ‘root’@’localhost’ identified by ‘soacom’;
```

Note that we created a user **soacom** and set the password to **soacom**.

Open browser, type : `http://localhost/phpmyadmin`
User: `soacom`
Password: `soacom`
{The default phpmyadmin installation does not allow root access.}

If you use Git as source control, install Git Cola, a graphical Git utility.

```
sudo apt-get install git-cola
```

Now verify Processor Architecture:

```
getconf LONG_BIT
```

Get OS Info:

```
cat /etc/os-release
```

Get System Info:

```
cat /proc/cpuinfo
uname -m
```

Orange PI

Orange PI Ubuntu image may be downloaded from the following link
https://drive.google.com/drive/folders/1KzyzyByev-fpZat7yvgYz1omOqFFqt1k

For other images, visit http://www.orangepi.org/html/serviceAndSupport/index.html then choose the hardware. The OS download options would be in found in the page. Once the image is downloaded, burn the image in a micro SD card, then boot. For details on how to burn an image into SD card visit elsewhere. Boot the Orange PI from the image. If the image successfully boots, you will see the Ubuntu desktop with Orange Pi logo and the top line containing Ubuntu Application menu. NOTE: The default username is **orangepi** and password is **orangepi**

Installation

We will install the following for software development. First, you need to install Gparted to resize the partition to atleast 8GB.

```
sudo apt-get update
sudo apt-get install gparted
```

Run Gparted and resize the partition to 8GB or higher. Then execute all the following commands.

Download default Java Development Kit (JDK)
```
sudo apt install default-jdk
```

Install Eclipse
```
wget https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2020-12/R/eclipse-java-2020-12-R-linux-gtk-aarch64.tar.gz
```
Upzip the .tar.gz file and the eclipse directory will contain the eclipse executable.

Install Mysql Server
```
sudo apt-get install mysql-server
```

Install CodeBlocks (For C/C++ development only)
```
sudo apt-get install codeblocks
```

Install Tomcat Server (For Java server development only)
```
sudo apt-get install tomcat9
```

Tomcat is typically installed in directory `/usr/share/tomcat9`. The war or ear file location is `/var/lib/tomcat9`

Install PhpMyAdmin (For database development only)
```
sudo apt-get install phpmyadmin
```

Change Mysql root password from blank to something else
```
sudo mysql -u root -p
```
Password is blank, simply press enter.
In mysql command prompt type the following:
```
use mysql;
create user 'soacom'@'localhost' identified by 'soacom';
grant all on *.* to soacom@localhost;
```
{The following line with change the root password. Default password is blank}
```
alter user ‘root’@’localhost’ identified by ‘soacom’;
```
Note that we created a user **soacom** and set the password to **soacom**.

Open browser, type : `http://localhost/phpmyadmin`
User: `soacom`
Password: `soacom`
{The default phpmyadmin installation does not allow root access.}

If you use Git as source control, install Git Cola, a graphical Git utility.
```
sudo apt-get install git-cola
```

Now verify Processor Architecture:
```
getconf LONG_BIT
```

Get OS Info:

```
cat /etc/os-release
```

Get System Info:
```
cat /proc/cpuinfo
uname -m
```

BeagleBone

BeagleBone Debian image may be downloaded from the following link
https://beagleboard.org/getting-started

Install the image with desktop so you can use Eclipse and the browser in beaglebone. Once the image is downloaded, burn the image in a micro SD card, then boot. For details on how to burn an image into SD card visit elsewhere. Boot the BeagleBone from the image. If the image successfully boots, you will see the Ubuntu desktop with BeagleBone logo and the top line containing Ubuntu Application menu. NOTE: The default username is **debian** and password is **temppwd**
The **root** user have password **root**. In some distribution the password is blank (no password).

NOTE: Beaglebone Green does not have HDMI port. To display you need to purchase the SeedStudio BeagleBone HDMI Cape and the firmware to use it, which is available from here: https://debian.beagleboard.org/images/bone-debian-8.7-lxqt-4gb-armhf-2017-03-19-4gb.img.xz
We recommend that you use BeagleBone Black, which has built-in hdmi.

PocketBeagle

PocketBeagle does not have a dedicated Ethernet port, so you need to connect to the Internet through USB networking along with Internet Connection Sharing. The detailed setup guide is described by the following blogs:
Windows Host:
http://ofitselfso.com/BeagleNotes/HowToConnectPocketBeagleToTheInternetViaUSB.php
Mac Host: https://beagleboard.org/p/hologram/sharing-internet-with-the-pocketbeagle-on-osx-cd62b2
Linux Host: https://www.elementzonline.com/blog/Sharing-Internet-using-Network-Over-USB-in-PocketBeagle
After connection sharing try using

`ssh debian@192.168.7.2` (for Windows)
`ssh debian@192.168.6.2` (for Mac)
`ssh debian@beaglebone.local` (for any)
sudo
One of them should work.

Installation

We will install the following for software development. First, you need to install Gparted to resize the partition to atleast 8GB.

```
sudo apt-get update
```

```
sudo apt-get install gparted
```

Run Gparted and resize the partition to 8GB or higher. Then execute all the following commands.

Download default Java Development Kit (JDK)
```
sudo apt install default-jdk
```

Install Eclipse
```
wget https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2020-12/R/eclipse-java-2020-12-R-linux-gtk-aarch64.tar.gz
```
Upzip the .tar.gz file and the eclipse directory will contain the eclipse executable.

Install Mysql Server
```
sudo apt-get install mysql-server

if mysql server is not found, then type:

sudo apt-get install mariadb-server
```

Install CodeBlocks (For C/C++ development only)
```
sudo apt-get install codeblocks
```

Install Tomcat Server (For Java server development only)
```
sudo apt-get install tomcat8
```

Install PhpMyAdmin (For database development only)
```
sudo apt-get install phpmyadmin
```

Change Mysql root password from blank to something else
```
sudo mysql -u root -p
```
Password is blank, simply press enter.
In mysql command prompt type the following:
```
use mysql;
create user 'soacom'@'localhost' identified by 'soacom';
grant all on *.* to soacom@localhost;
```
{The following line with change the root password. Default password is blank}
```
alter user ‘root’@’localhost’ identified by ‘soacom’;
```
Note that we created a user **soacom** and set the password to **soacom**.

Open browser, type : `http://localhost/phpmyadmin`
User: `soacom`
Password: `soacom`
{The default phpmyadmin installation does not allow root access.}

If you use Git as source control, install Git Cola, a graphical Git utility.
```
sudo apt-get install git-cola
```

Now verify Processor Architecture:

```
getconf LONG_BIT
```

Get OS Info:
```
cat /etc/os-release
```

Get System Info:
```
cat /proc/cpuinfo
uname -m
```

Displaying Graphics on PC

You can display the X Windows graphics on your PC. Install Cygwin from https://www.cygwin.com and make sure to choose to install X11 and its components. Finish the installation and then go to your Windows Search menu and type “xwin”. Click on the XWin Server icon. You will see XDG Menu icon. Click on it and go to System Tools → Xterm. Inside Xterm window type:

ssh -XY debian@192.168.7.2

Default password is **temppwd**. Now all your beaglebone graphical display will appear on your PC.

BeagleBone AI-64

The process to setup the AI-64 is similar to beagle bone. AI-64 can be connected to a monitor and every thing can be done on the monitor. Follow the same steps like Beagle Bone. You do not need to run Gparted to extend the partition as BB AI-64 uses the entire 16GB emmc for linux.

Mac OS X

Mac OS X on the Intel platform is replaced by M2 processor. Apple will gradually remove their support on Intel Macs. There is a version of the demo that should run on Intel Mac, but it will no longer be supported in future. Only Mac (Aarch64) macs will be supported. Later generation Mac does not need drivers for USB serial devices.

Python

Python code is developed and tested on version 3.7. Python code is available from python3 directory. To run the test utility, use:

```
python j42xxtest.py
```

or

```
python3 j42xxtest.py
```

making sure that the platform dependent driver is in the same directory. For Windows, this file(s) is(are) located in platform/win64 directory. You must copy all the DLL files to the python3 directory. For, Mac OS X, this file is `libj4210u.dylib`. For all other OSes, this file is

`libj4210u.so`. Before running the test utility you must connect the device into the serial port and make sure that the right serial port is used to connect using the python code.

If your Linux OS does not have Python, then it can be installed by:

`sudo apt-get install python3`

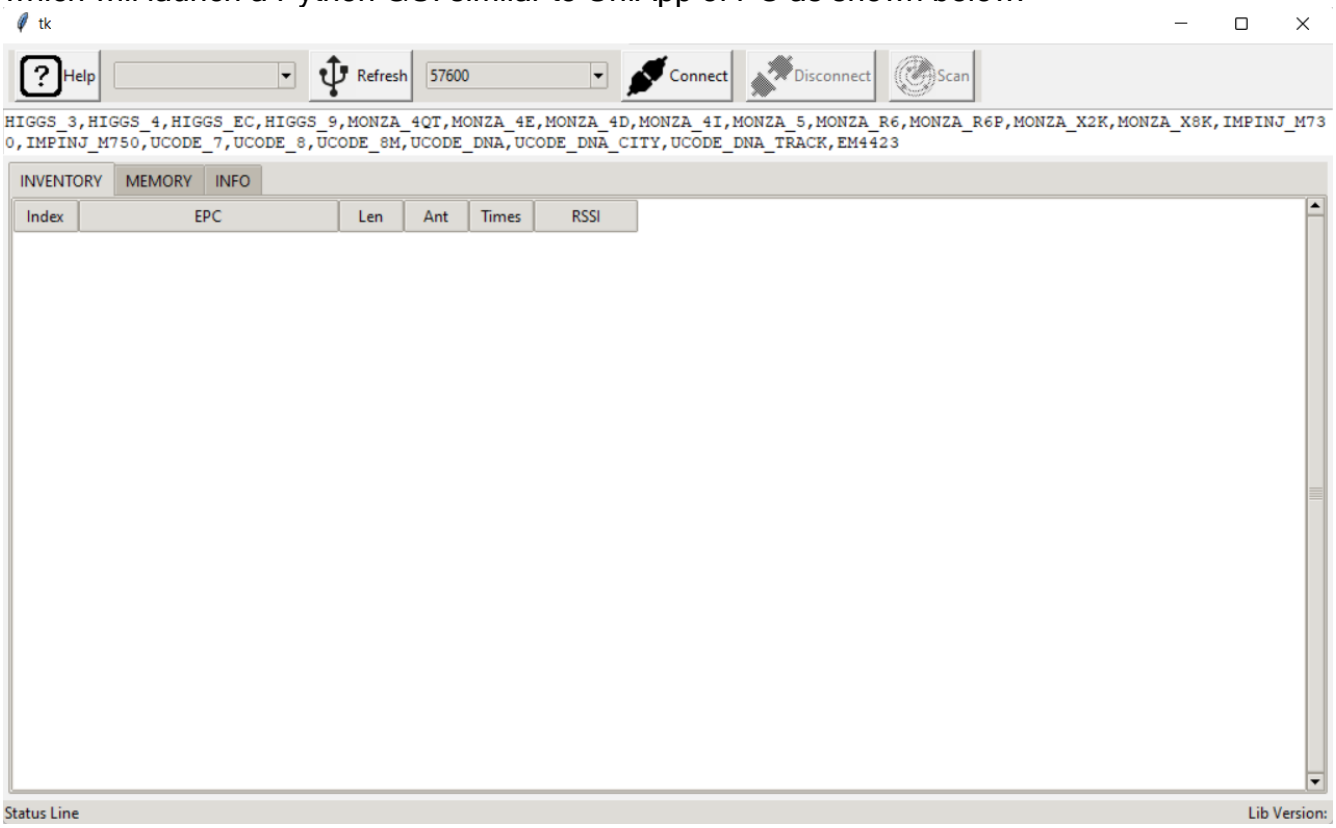
For GUI, you also need to install the Tkinter package for Python:

`sudo apt-get install python3-tk`

Alternatively, the `UhfApp.py` could be run in the following way:

`python UhfApp.py`

which will launch a Python GUI similar to UhfApp of PC as shown below.



Please click on Refresh button and select the comport of UHF device (here, it is COM14). To connect the device please clean on Connect button. If you place a UHF tag on the device and press scan button, the device will scan the tag and the unique EPC no of the tag will appear.

If you fail to connect to a Serial Port, most likely your user does not have access to the port. In that case, exit the App and the the following:

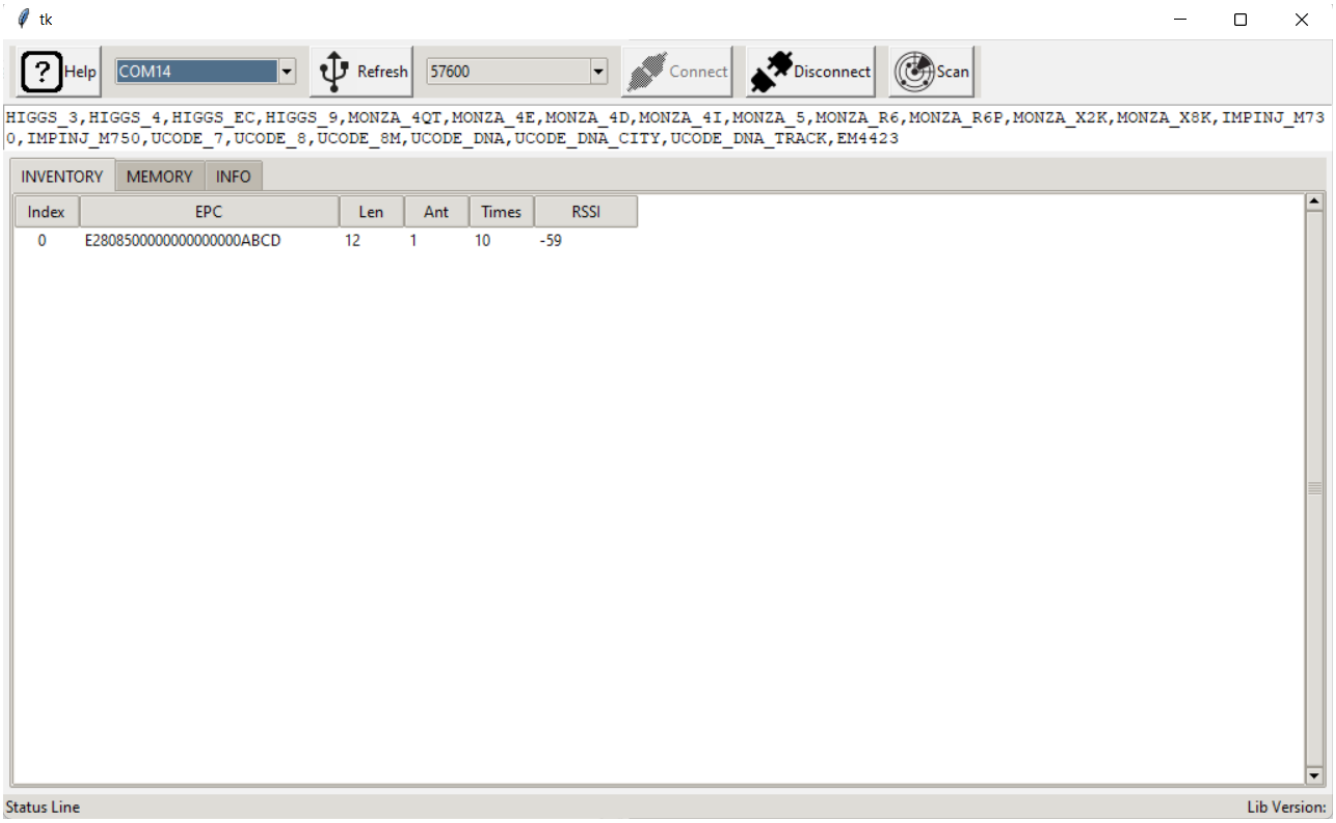
`sudo python3 UhfApp.py`

To avoid typing sudo every time, add the current user into the serial port owner group as follows:

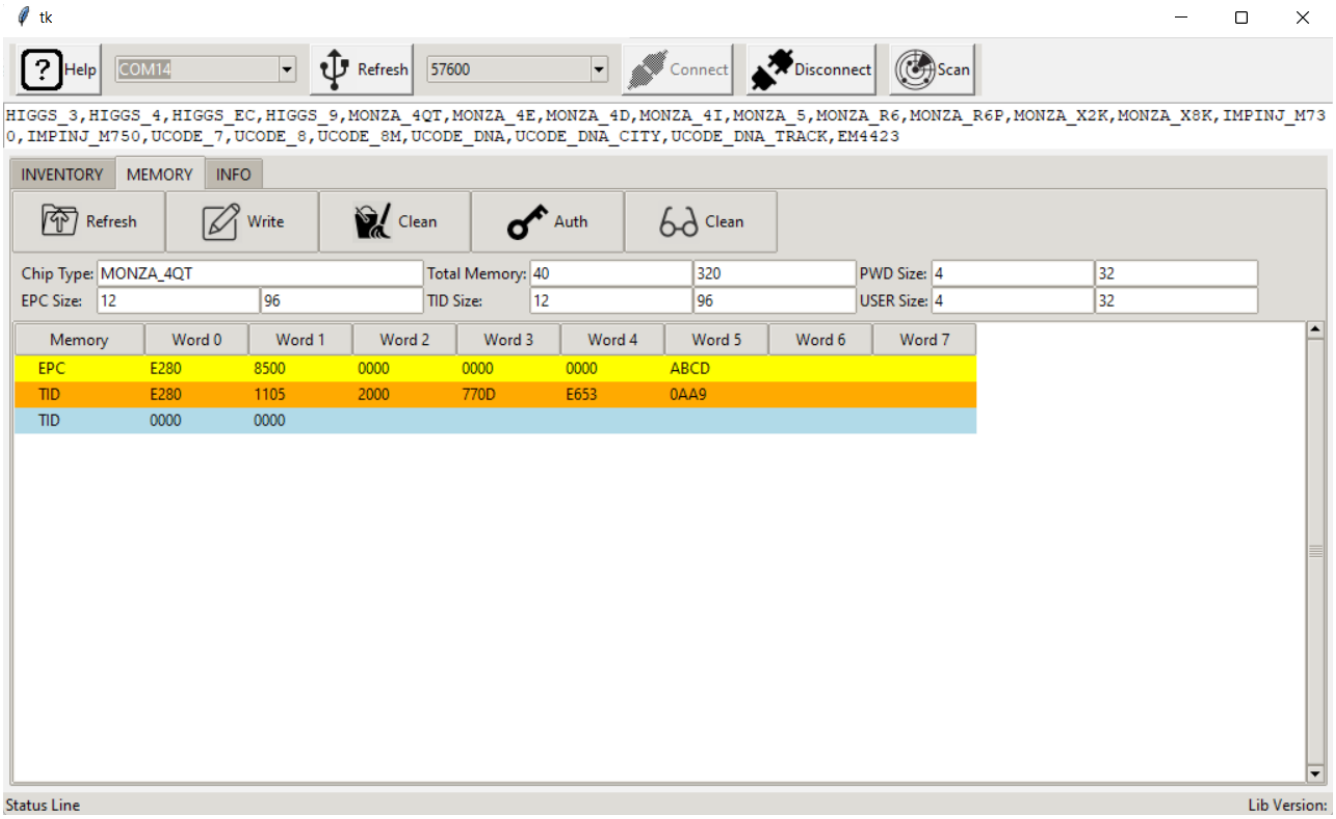
```
sudo usermod -a -G dialout $USER
```

where dialout is the user group of root (in Ubuntu).

Now you can run UhfApp.py without sudo.



One can see the details of the tag such as Chip Type, Total Memory, PWD Size, EPC Size, TID Size as well as USER size by double clicking on the EPC number.



Jence Uhf App

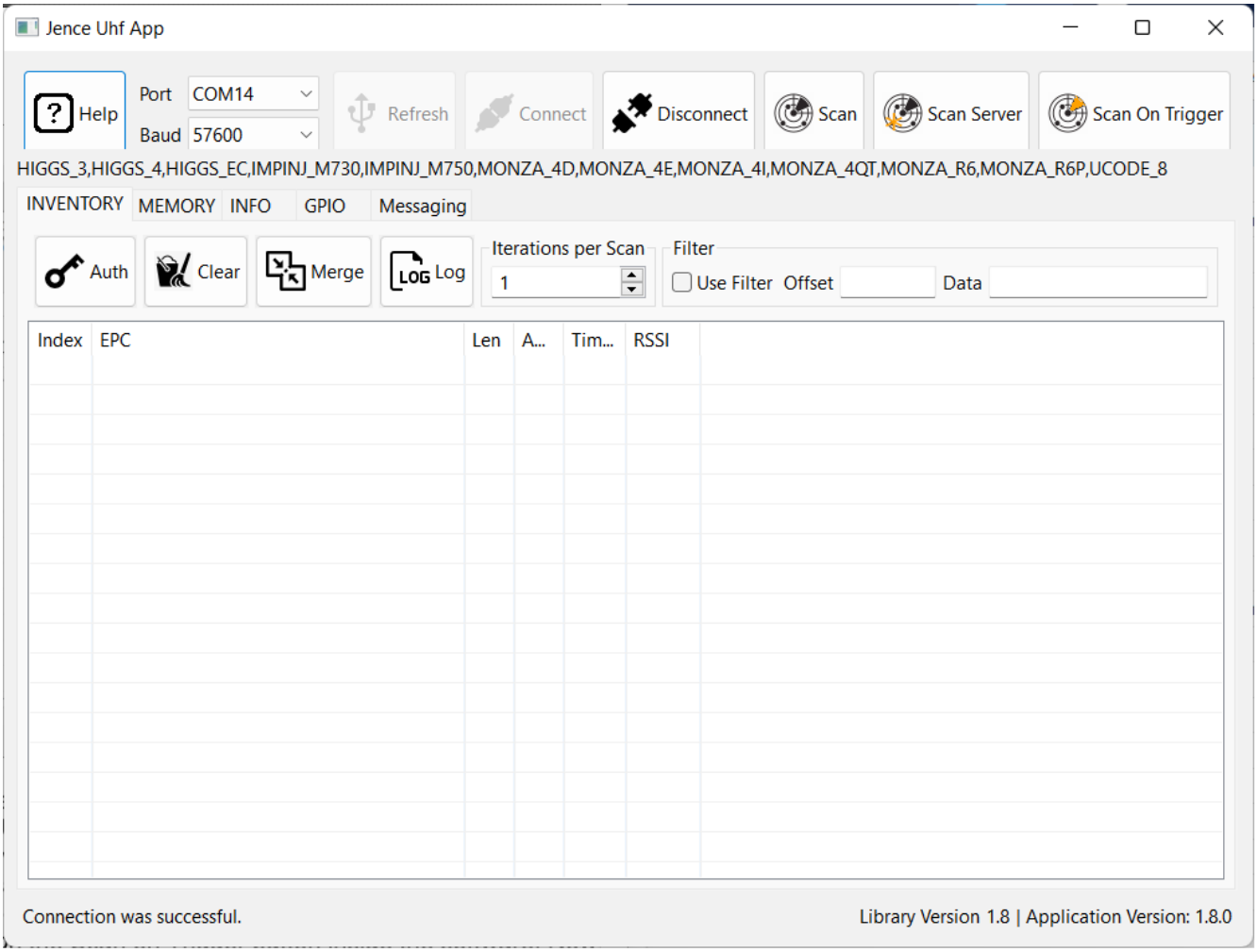
A Java application called UhfApp is provided with full source code. This application uses SWT (Standard Widget Toolkit) library for all its GUI frontend. SWT uses underlying OS'es native API to display the widgets and therefore the widgets get a native look. In addition, due to native calls, all hardware 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.

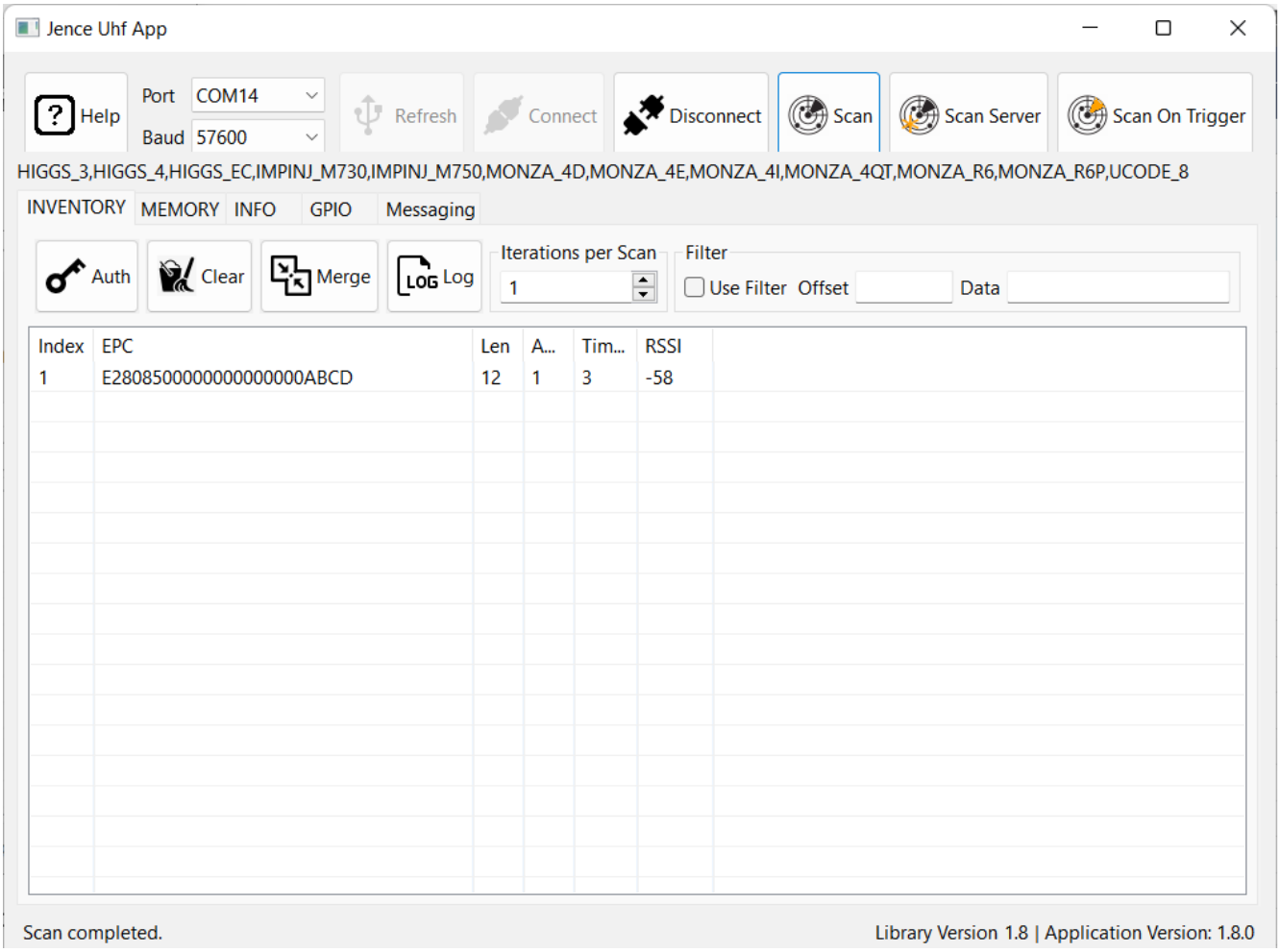
After the download, please unzip the folder. Then go to the demo folder. Inside the demo folder, there is an application file j4210u.exe. If you run this application file, two command prompt window will appear. Now connect the UHF hardware with the computer through USB cable and find the com port inside 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.

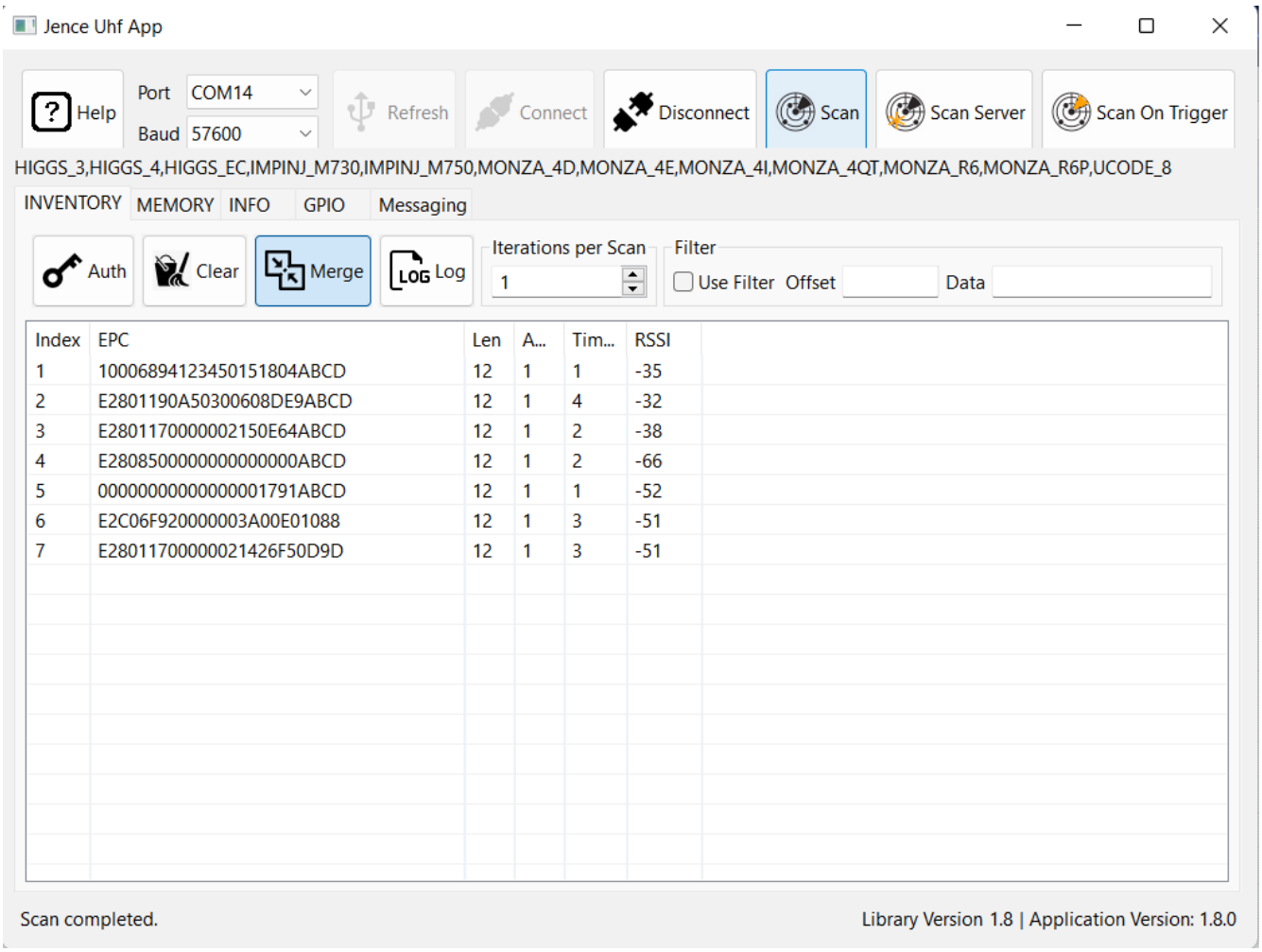


This UHF software tool can directly detect Ucode-8, Monza 4Q1, Monza R6P, Monza 4D, HIGGS 3, IMPNJ M730 and Monza R6 UHF RFID CHIPS. Apart from these chips, it can read other unknown chips as well. This software tool can detect several tags or cards at a time. If you use UHF Desktop hardware, in that 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.

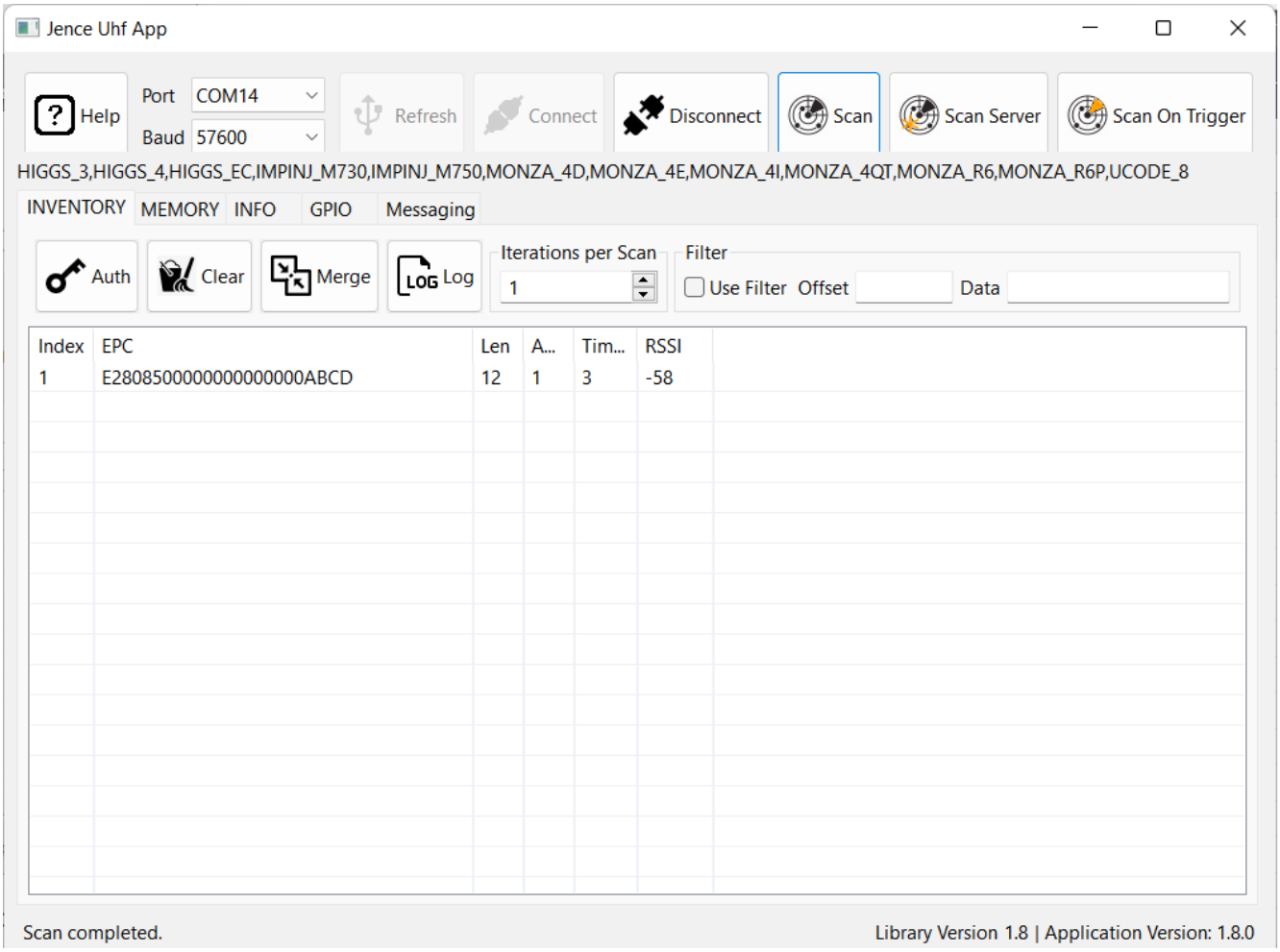
Inventory Tab: When the software application scans cards or tags, we will see the EPC (Electronic Product Code) of those cards inside the inventory tab.



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.

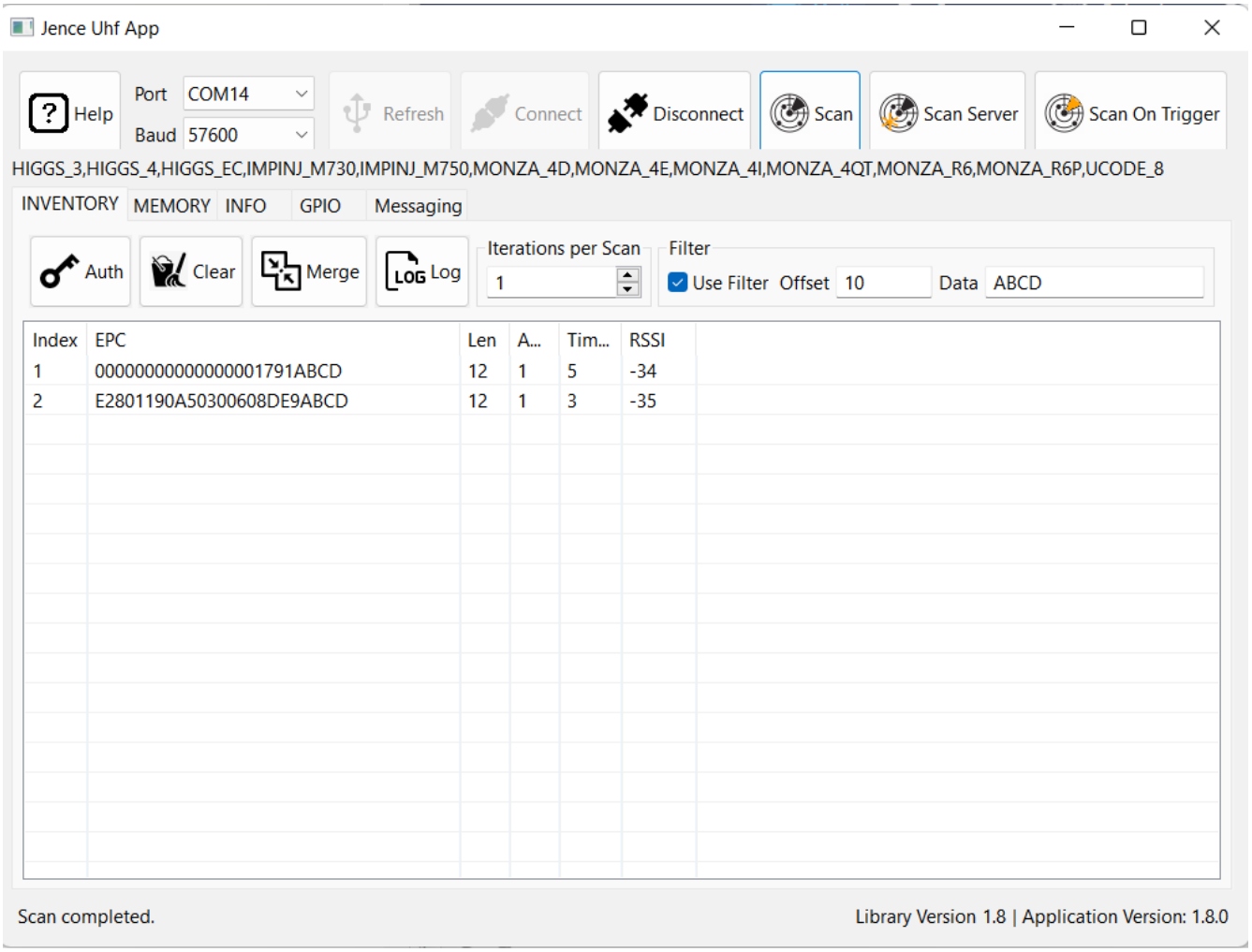


There is a filter option inside the inventory tab. If we want to search tag with similar value, such as similar suffix or prefix, we can use this option. At first, we have to give the offset value. We should consider even offset. To elaborate: If we want to search tags with ABCD suffix in it, let's assume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

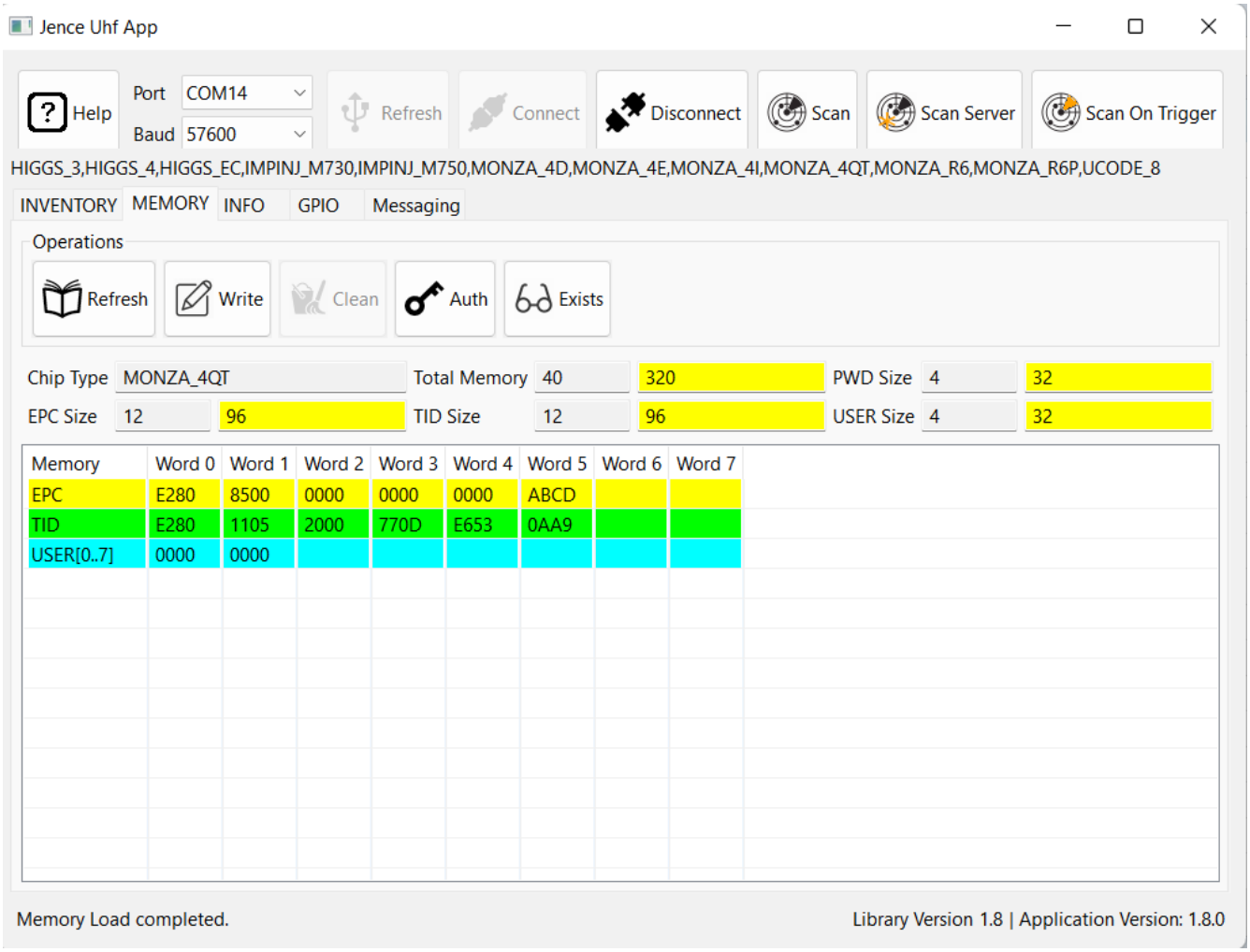
E280 6894 0000 5015 1804 ABCD – EPC No
0 1 2 3 4 5 6 7 8 9 10 11 – offset value

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

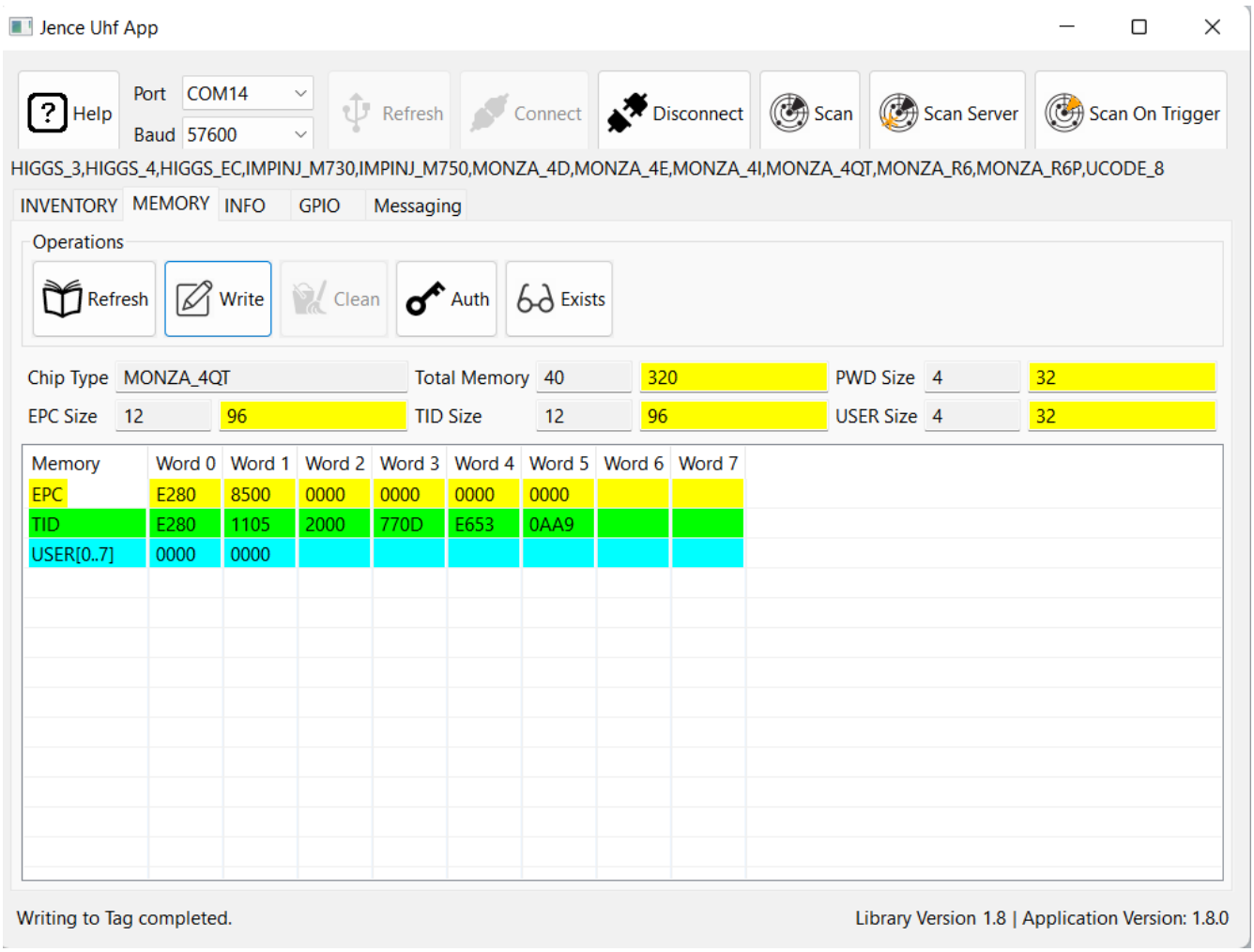
Here, 0,1,2,3,4,5,6,7,8,9,10,11 are offset values. Offset value must be 2 byte (4 hex values). Only even offset should be used. ABCD value is in 10 11 offset where 10 is the even offset, we will write 10 in offset option and we will write ABCD in data option. We must check Use Filter option. If we search with UHF Handheld reader, we will find tags with this ABCD suffix. It is very useful to find tags with similar prefix, suffix or values in large inventories.



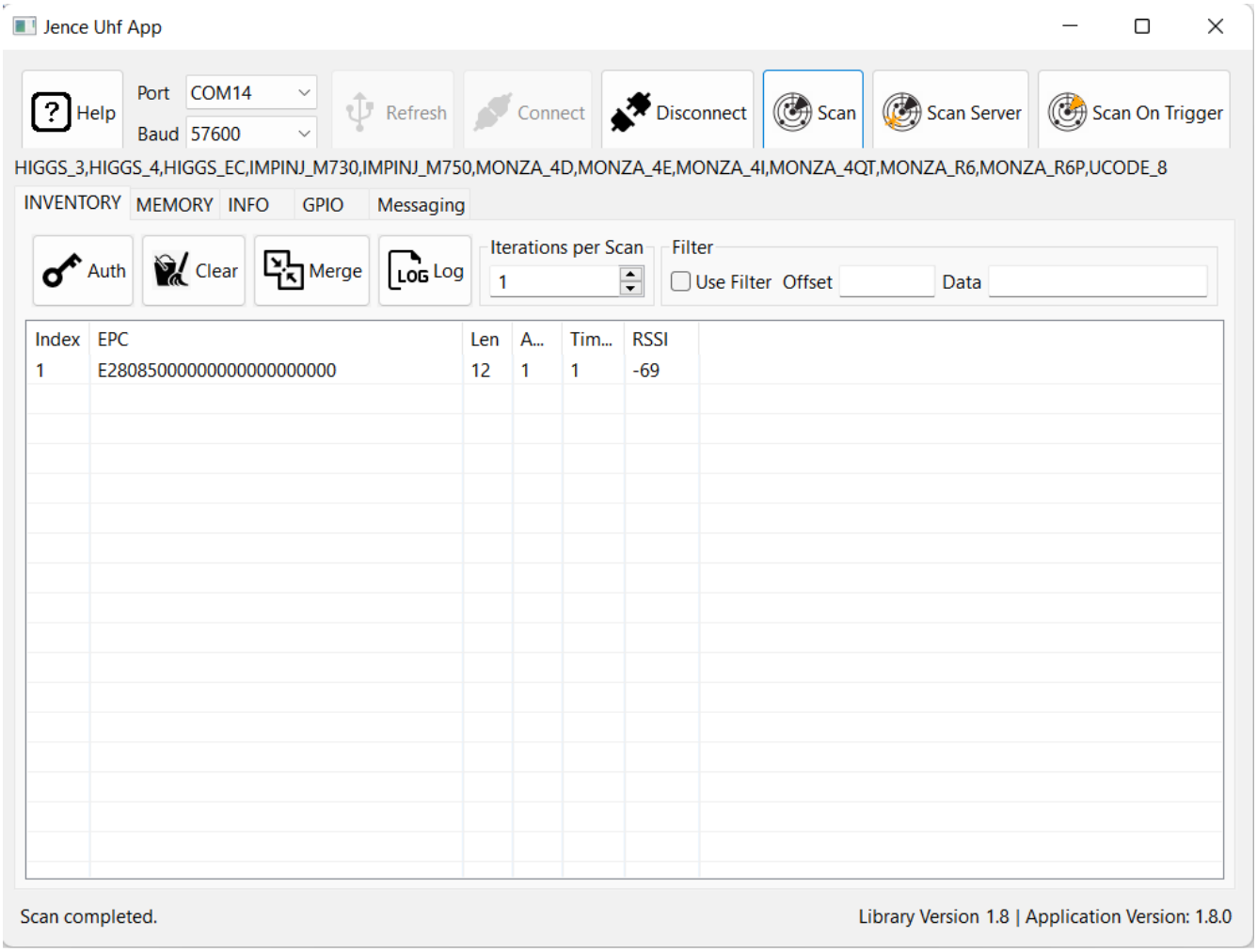
Memory tab: Now if we double click on the EPC no while card is within the range of UHF hardware and go to the memory tab, we will see the inside content of the card, such as chip type, total memory, PWD size, EPC size, TID size and user size. The numbers marked with yellow colour are in bit format and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



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.

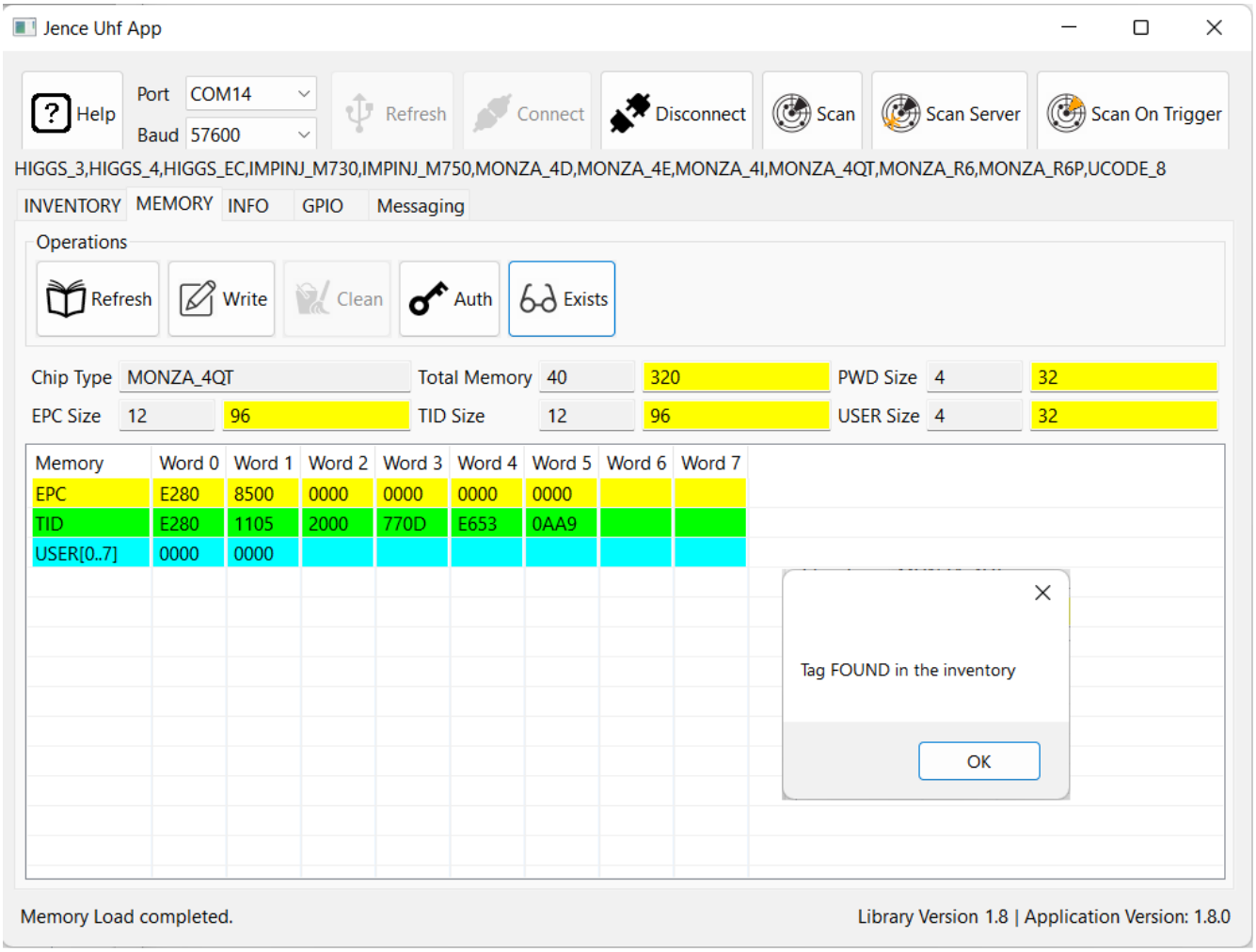


Now if we scan the card, we will see the EPC no with altered value in it.

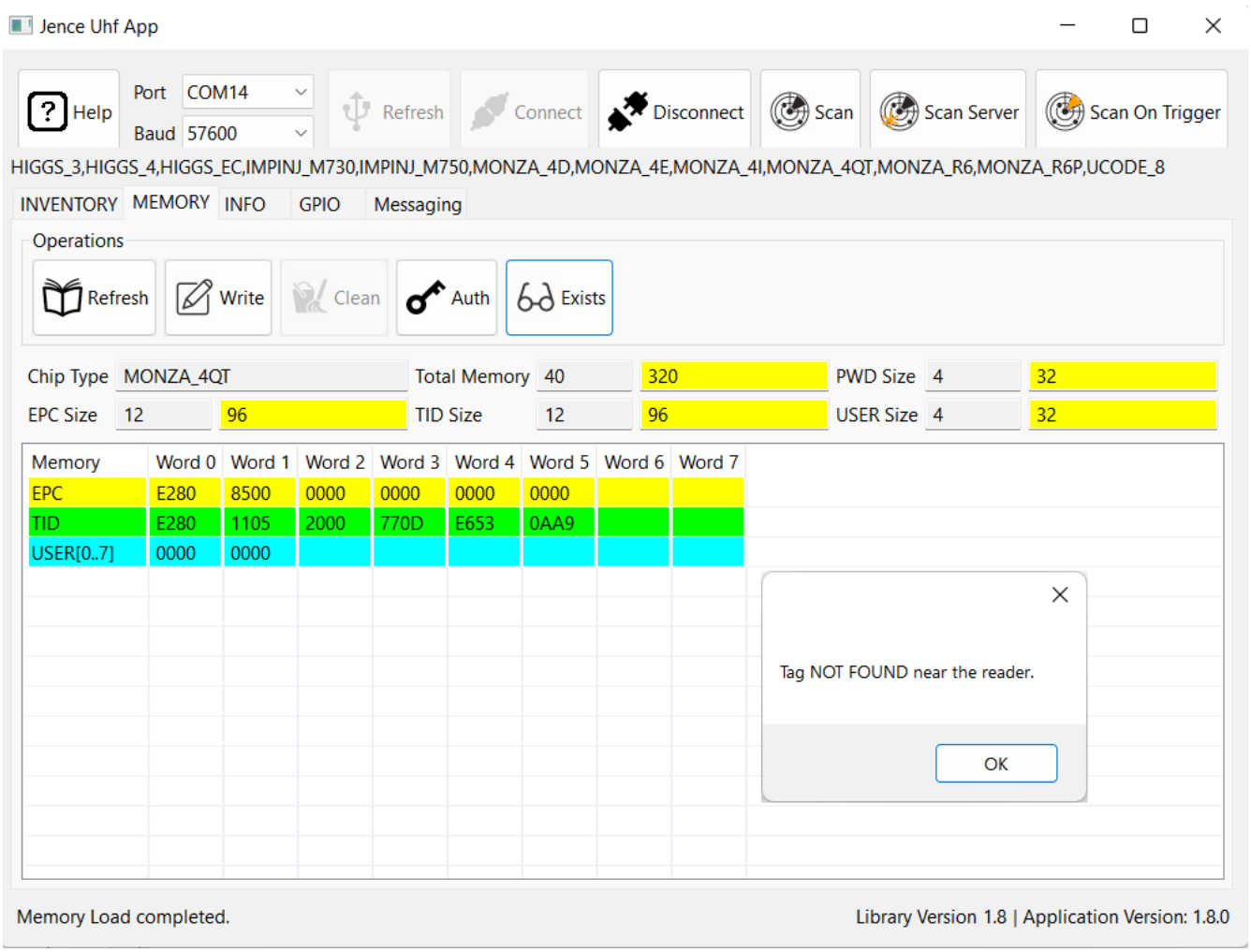


In this way, we can give similar suffix or prefix or middle value to several tags to search them promptly. There is an Exists button inside the memory tab.

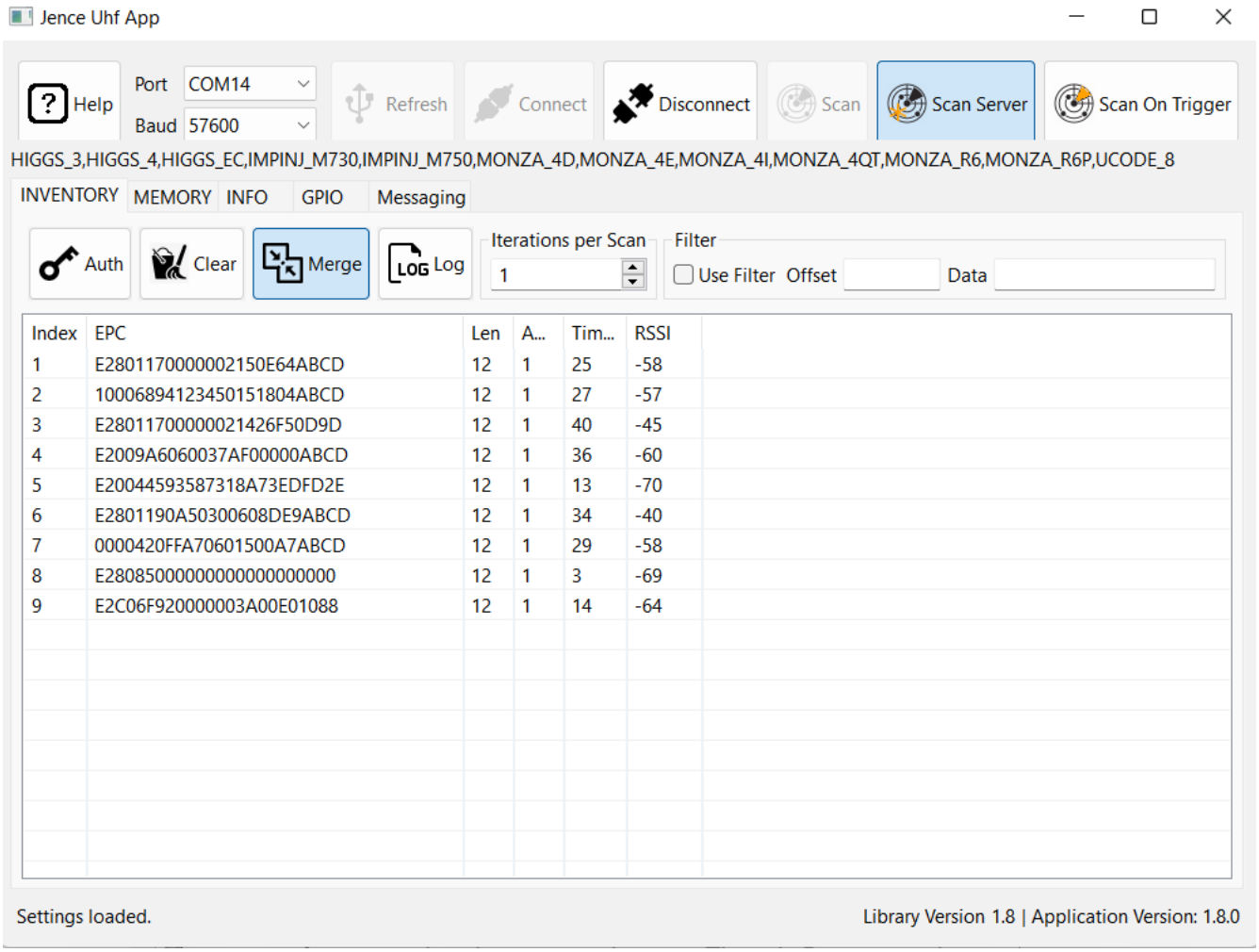
If tag is in the inventory and within the range of UHF Handheld reader and if we click on this button, it will show the pop up message “Tag found in the inventory”.



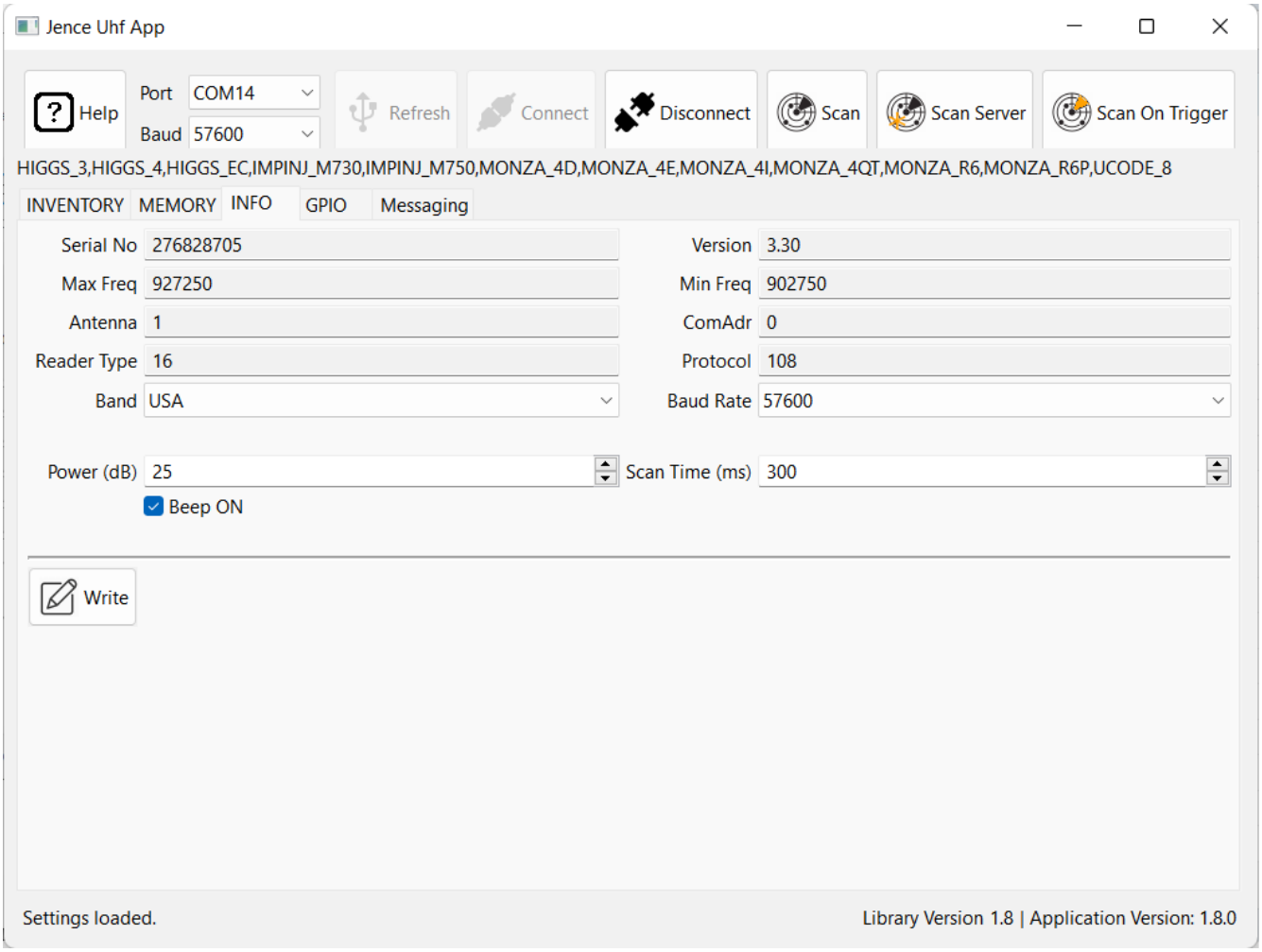
If tag is not in the inventory or not within the range of UHF reader, it will show the message“ Tag not found near the reader”.



Scan Server: This button causes a server to continuously scan and display the result on the table. This option could be used in combination with the Merge button. Scan Server button is a toggle. So, to stop the scan server, simply press the button if the button appear pressed.



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.

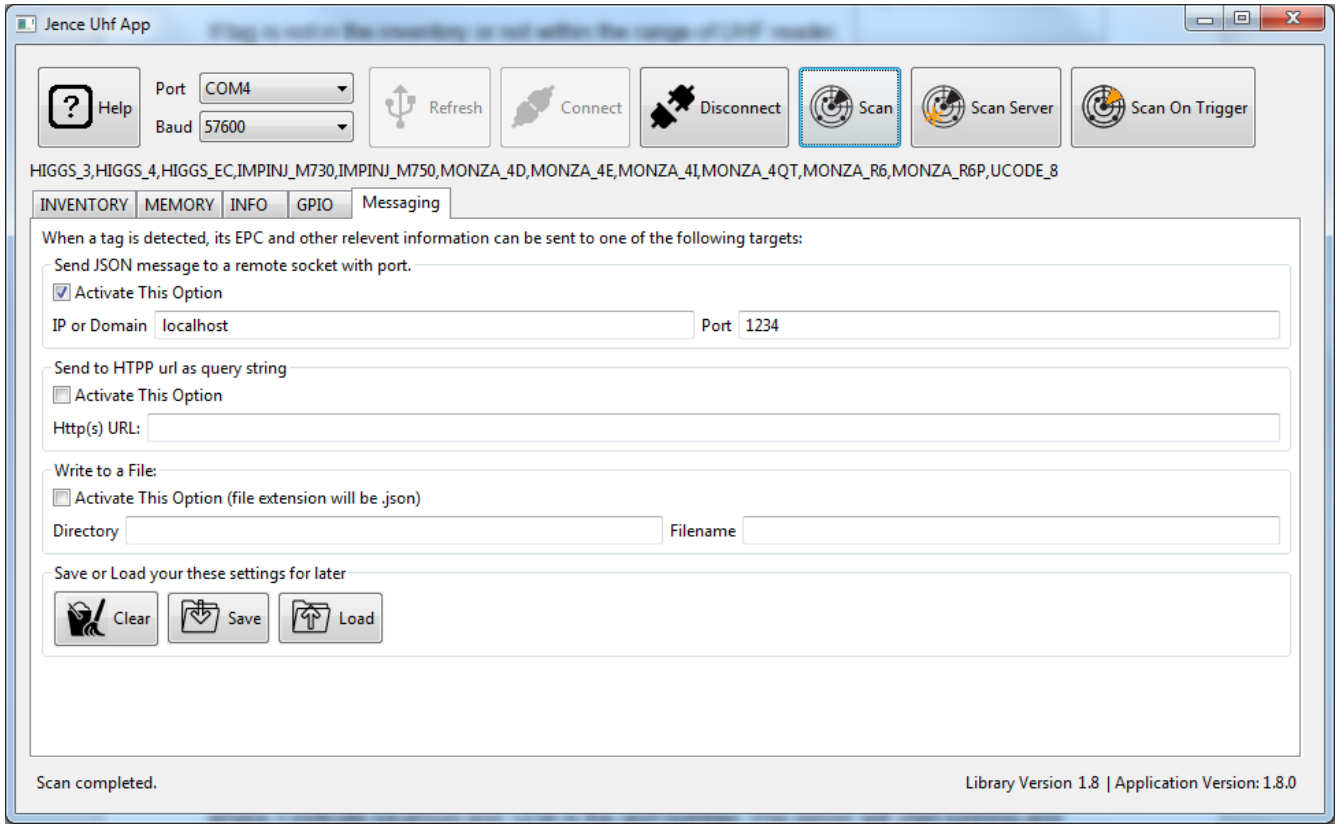


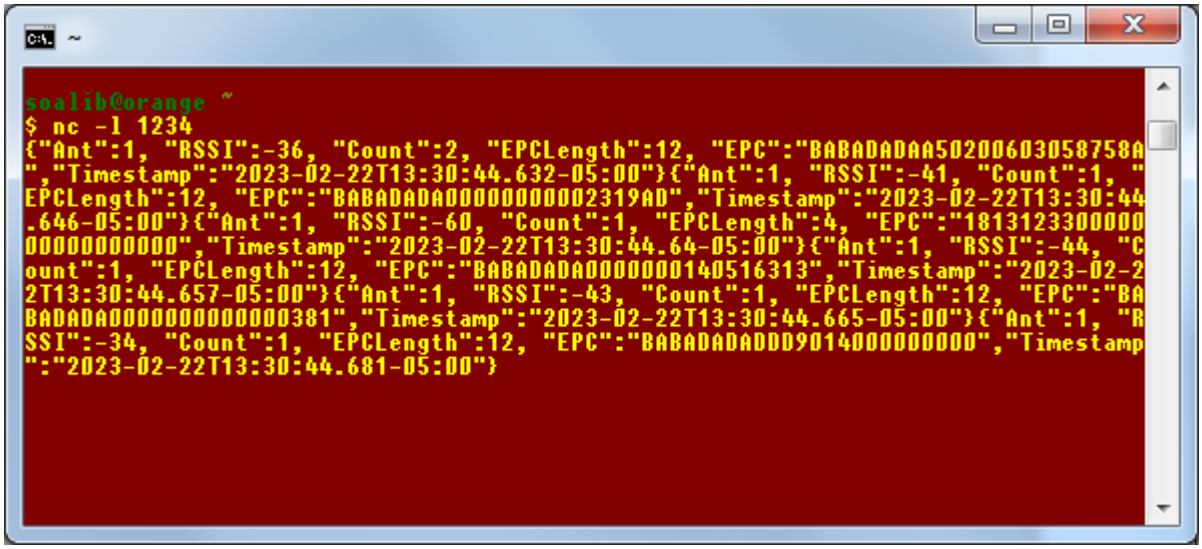
Messaging tab: This tab is useful for developers when they want the scanned data in JSON format. There are three ways to get JSON formatted message. The first option is send the JSON message to a Server Socket. In Windows, Cygwin and Linux or Mac OS X, **nc** command can be used to create a server with a single line.

Open Cygwin terminal or a terminal in Linux or Mac OSX and type as follows:

```
nc -l 1234
```

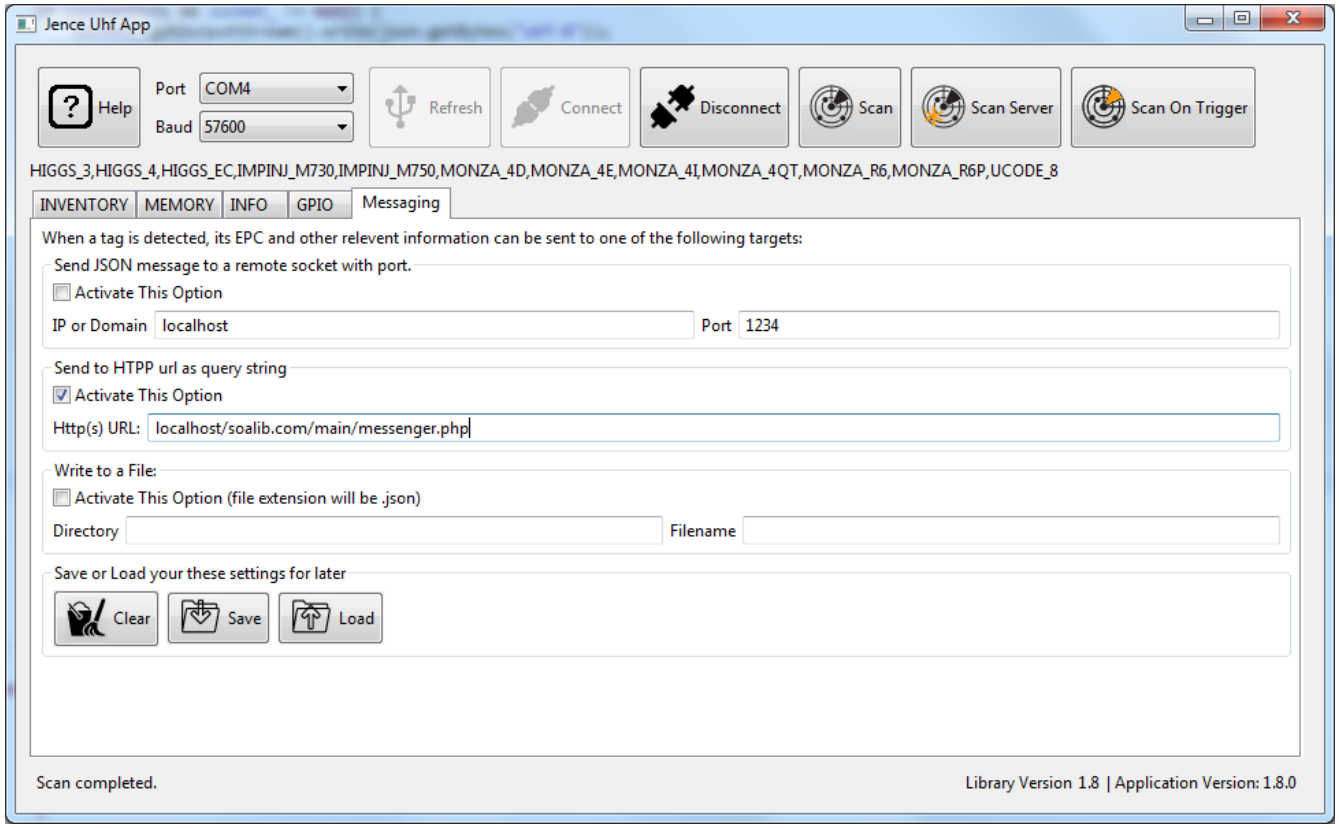
where -l indicate localhost and 1234 is the port number. The server will start running and listening to the port. In the Messaging tab, select the first option “Send JSON messaeg to remote socket with port” and enter localhost as IP and 1234 as port. If you have a server running at different IP and/or port, you may enter it here. Press the “Scan” button. The terminal window will display the individual card information in JSON format. Each tag information constitute individual JSON message.



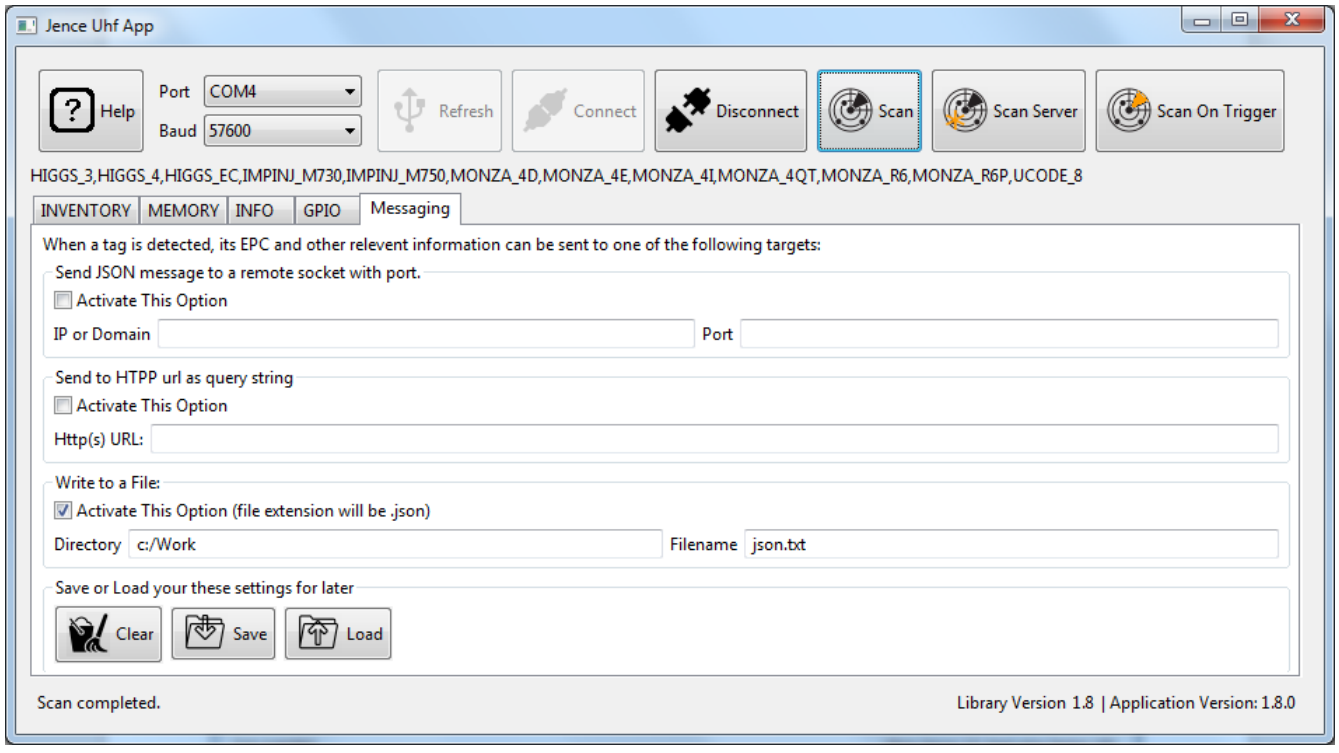


The settings may be saved by hitting the Save button, which may later be loaded using the Load button.

The second option sends the JSON message to a HTTP(S) as GET message. In the Http(s) URL, enter the URL string that will receive the JSON message. Pressing the Scan button will shoot cause one or more calls to the URL with JSON string for each card.



Third option logs JSON strings when the Scan button is pressed. The strings are logged into the directory and file specified in this option.



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 additional, the environmental issue plays a role due to the fact that the tags does 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 surroinging 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.

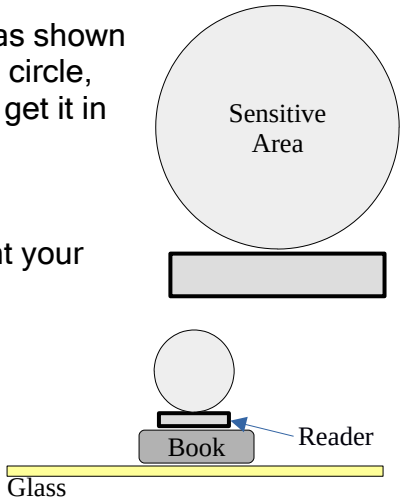
In addition, the tag itself could be the cause of the problem. In your case, it does not seem that the tag is an issue because it is found eventually.

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.

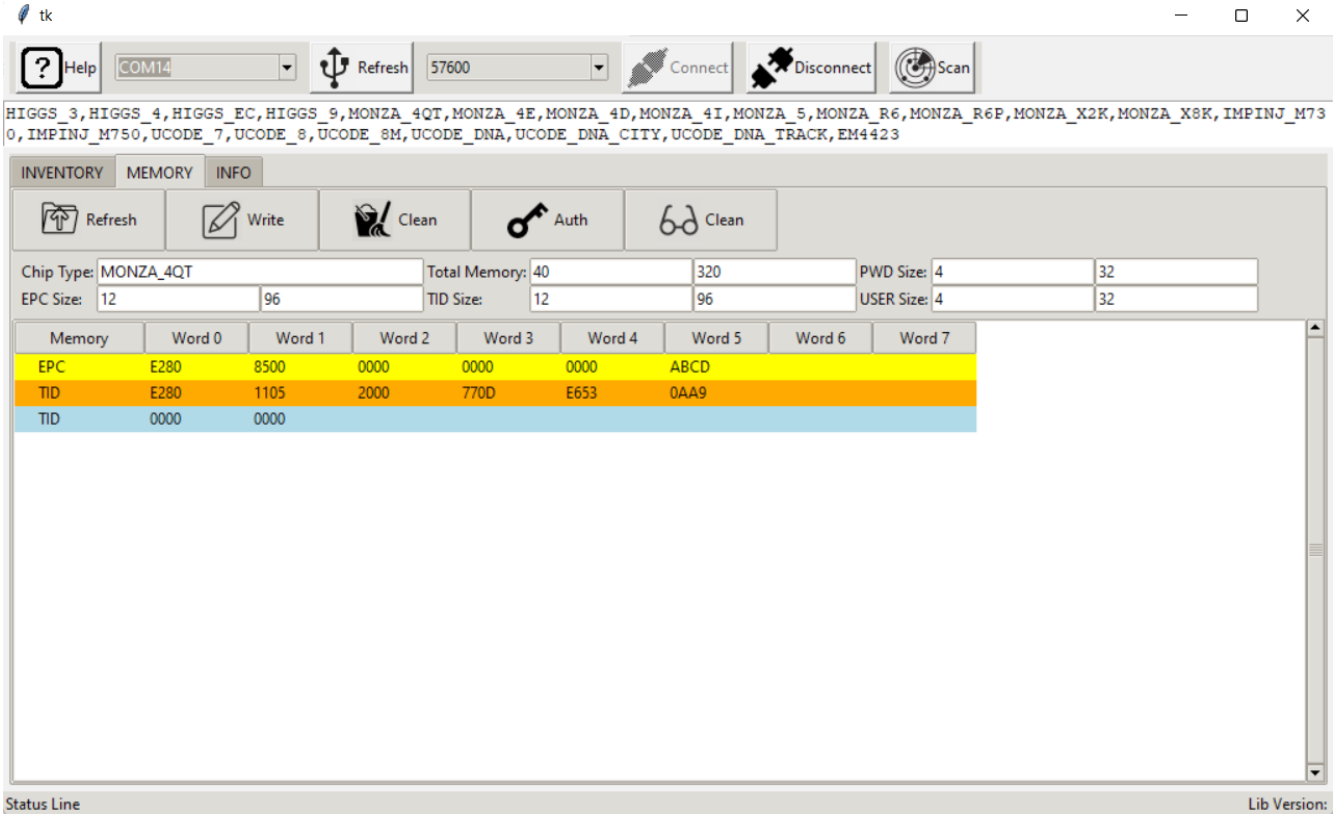
ANTENNA LOBE

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 device.

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. The j4210 driver is not in path.

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 pro 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.

5. Attaching device does not cause any serial port to appear.

A. This happens in older OSes. New OSes have automatic detection of USB to Serial generic driver and have no problem in communicating with the device. If your OS does not show a serial port when device is connected to USB port, first install a driver from here: https://www.microchip.com/en-us/product/MCP2221. If the driver does not resolve the problem, remove the device and go to /dev folder (for linux) and count the number of devices there. Then attach the device and count the number of devices in the /dev folder again. If there is a change, then find the name of the newly added device and send it to us. We will add the new device into our driver within a short time and release the new driver.

DLL Functions

Following DLL functions are available.

unsigned char AvailablePorts(char ports[2048]);

Auto detects avialable 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, int baud);

Opens a COM port.

Parameters:
port: com port name. Example “COM4”, etch
baud: baudrate, should be one of 57600, 115200.

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 GetTagInfo(unsigned char* tid, unsigned char* info)

Gets tag information for the tag ID. The info received has the following structure:

```
struct TagInfo {
   int type;
   int tidlen;
   unsigned char tid[64];
   char chip[16];
```

```
    int epclen;
    int userlen;
    int pwdlen;
};
```

Parameters:
tid: TID code of the tag for which TID is requested.
info: buffer to get the result returned.

Returns: 1, if successful.

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

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.
pwdlen: 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 kpwdlen))

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.
kpwdlen: 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 pwdlen)

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

Parameters:
pwd: an array containing password.
pwdlen: 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 #3. 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.

unsigned char SetGPO(unsigned char gpono)

Sets output of the GPIO to the value assigned. There are two GPIO outputs. Bit zero of gpono send output to GPO-0 and bit 1 sends to GPO-1. So, SetGPO(2) means, GPO-0 = 0 and GPO-1 = 1.

Parameters:
gpono: bit wise assignment of state per GPO output port.

Returns: 1, if successful.

unsigned char GetGPI(unsigned char gpino)

gets input from GPIO. There are two GPIO, GPI-0 and GPI-1. To get input from GPI-0, send gpino = 1, to get input from GPI-2, send gpino = 2. All other values are invalid.

Parameters:
gpino: GPI input filtering bit.

Returns: GPI input from the given port.

unsigned char SetQ(unsigned char Q)

Sets the Q parameter. For a small number of tags, set this to 5. The best value should be found by trial and error.

Parameters:
Q: Optimal Q value.
Returns: 1 on failure.

unsigned char SetSession(unsigned char sess)

Sets inventory session. Sessions range from 0 to 3. If there are other RFID reader around, each reader should use a different session so the inventory scan will not affect the other readers.

Parameters:

sess: session number between 0 and 3.
Returns: 1 on failure

Driver Version History

Version 1.8

Added support of Raspberry PI, Orange PI, Banana PI, BeagleBone, Mac OSX Intel. Added support for new hardware revision.

Version 1.7

Added functions SetQ and SetSession.

Version 1.6

Linux shared library and GUI modification and .app creation.

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