

ALIEN TECHNOLOGY®

ALR-H450/H460 Developer Guide

June, 2019



ALR-H450
ALR-H460

Legal Notices

Copyright ©2019 Alien Technology, LLC. All rights reserved.

Alien Technology, LLC and/or its affiliated companies have intellectual property rights relating to technology embodied in the products described in this document, including without limitation certain patents or patent pending applications in the U.S. or other countries.

This document and the products to which it pertains are distributed under licenses restricting their use, copying, distribution and decompilation. No part of this product documentation may be reproduced in any form or by any means without the prior written consent of Alien Technology, LLC and its licensors, if any. Third party software is copyrighted and licensed from Licensors. Alien, Alien Technology, the Alien logo, Nanoblock, Fluidic Self Assembly, FSA, Gen2Ready, Squiggle, Nanoscanner and other graphics, logos, and service names used in this document are trademarks of Alien Technology, LLC and/or its affiliated companies in the U.S. and other countries. All other trademarks are the property of their respective owners. U.S. Government approval required when exporting the product described in this documentation.

Federal Acquisitions: Commercial Software -- Government Users Subject to Standard License Terms and Conditions. U.S. Government: If this Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE HEREBY DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



Alien Technology®

ALR-H450/H460 Developer Guide



Table of Contents

| | | |
|----------|---|-----------|
| 1 | INTRODUCTION..... | 1 |
| 1.1 | Audience..... | 1 |
| 1.2 | Type Conventions..... | 1 |
| 1.3 | Overview..... | 1 |
| 1.4 | Documentation and Sample Code..... | 1 |
| 1.5 | System Requirements..... | 1 |
| 2 | RFID READER..... | 2 |
| 2.1 | Introduction..... | 2 |
| 2.2 | Enumerations..... | 3 |
| 2.2.1 | Bank..... | 3 |
| 2.2.2 | Target..... | 3 |
| 2.2.3 | Session..... | 3 |
| 2.2.4 | LockType..... | 3 |
| 2.2.5 | RFIDInfo..... | 3 |
| 2.3 | Connecting to RFID Module..... | 4 |
| 2.4 | Tag Operations..... | 4 |
| 2.4.1 | Mask..... | 4 |
| 2.4.2 | Reading a single tag..... | 5 |
| 2.4.3 | Continuous Inventory..... | 6 |
| 2.4.4 | Reading and Writing tag's memory..... | 7 |
| 2.4.5 | Lock Fields..... | 9 |
| 2.4.6 | Locking and Unlocking Tag Memory..... | 9 |
| 2.4.7 | Kill tags..... | 10 |
| 2.4.8 | Tag class..... | 11 |
| 2.5 | Setting RFID Parameters..... | 12 |
| 2.5.1 | Transmit Power..... | 12 |
| 2.5.2 | Q Parameter..... | 12 |
| 2.5.3 | Session Parameter..... | 13 |
| 2.5.4 | Inventory Target..... | 13 |
| 2.5.5 | RFID Subsystem Information..... | 14 |
| 2.5.6 | Device Information..... | 14 |
| 3 | BARCODE READER..... | 15 |
| 3.1 | Introduction..... | 15 |
| 3.2 | Controlling Barcode Reader..... | 15 |
| 3.2.1 | Scan 1D and 2D barcode..... | 15 |
| 3.2.2 | Controlling Barcode Parameters..... | 16 |
| 3.3 | Symbology Parameters..... | 17 |
| 3.3.1 | List of Barcode Symbology Parameters..... | 17 |
| 3.3.2 | Barcode Symbology Parameter Values..... | 19 |
| 4 | KEY CODES..... | 27 |
| 4.1 | ALR-H450 (Android 4.4.2)..... | 27 |
| 4.2 | ALR-H460 (Android 6.0)..... | 27 |
| 5 | DEVELOPING APPLICATIONS WITH ANDROID STUDIO..... | 28 |
| 5.1 | Install Android Studio..... | 28 |
| 5.2 | Install Google USB Driver..... | 28 |
| 5.3 | Enable Developer Mode for your handheld..... | 29 |
| 5.4 | Developing your first RFID application..... | 30 |
| 5.4.1 | Create Android Project..... | 30 |
| 5.4.2 | Use RFID functionality in your project..... | 34 |

1 Introduction

The Alien ALR-H450/H460 Software Developer Kit (SDK) provides libraries and sample code for programmatically controlling Alien ALR-H450/H460 handheld readers, which are running on an Android operating system. Alien provides class libraries and sample applications to help you get up-and-running developing your custom applications to run directly on the device.

1.1 Audience

We assume that the readers of this guide:

- are proficient Android developers,
- have minimal previous knowledge of RFID, and other relevant technologies.

1.2 Type Conventions

- Regular text appears in a plain, sans-serif font.
- External files and documents appear in *italic text*.
- Class names appear in a fixed-width serif font.
- Things you type in, and sample code appear:
indented, in a fixed-width serif font.
- Longer blocks of sample code appear like below:

```
// Obtain RFIDReader instance
RFIDReader reader = RFID.open();
// Release RFIDReader instance
reader.close();
```

1.3 Overview

This document focuses on controlling the ALR-H450/H460 handheld readers using the Alien Android API. The Android Library provided in the SDK, **alienapi.aar**, supports controlling both the RFID Reader and 1D/2D Barcode Scanner.

| Library | Hardware Module | ALR-H450/H460 (Android) |
|--------------|-----------------------|-------------------------|
| alienapi.aar | RFID Reader | ✓ |
| | 1D/2D Barcode Scanner | ✓ 1D & 2D |

1.4 Documentation and Sample Code

This Guide provides complete documentation of all API elements. Additional usage tips can be gleaned by examining and modifying the source code samples provided by Alien. The sample applications demonstrate most of the major features of each hardware module.

1.5 System Requirements

You will need Android Studio 1.4 (or newer) to develop applications for the Alien handhelds.

2 RFID Reader

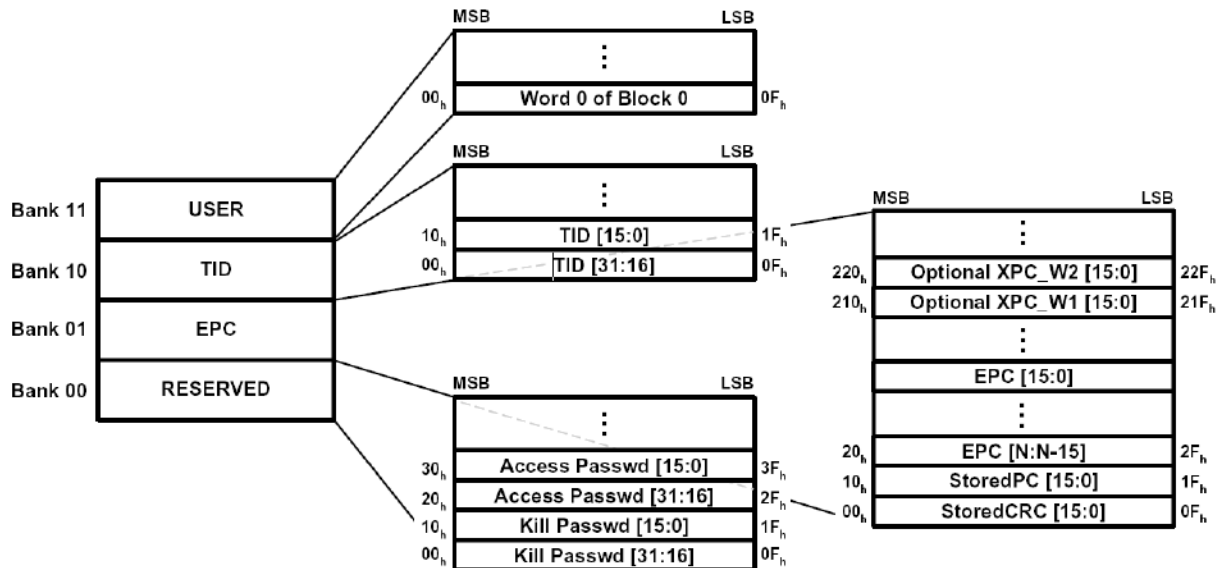
2.1 Introduction

The Alien `alienapi.aar` API library contains classes to provide interface with the RFID module in your Alien handheld reader. After initializing the `RFIDReader` class object, use its methods to communication with the RFID module.

You have control over most aspects of the Gen2 protocol, including access operations like write, lock, and kill, using tag masks, as well as reading tags' EPC IDs and Return Signal Strength Indication (RSSI).

To take full advantage of the RFID module and the Gen2 protocol, you are encouraged to read and understand the relevant portions of the EPCglobal Gen2 specification. One thing you must be aware of when accessing individual portions of memory in the tag is the layout of tag memory. As shown in the diagram below, all Gen 2 tags have four banks of memory (RESERVER, EPC, TID, USER) and each of those banks can be broken down into fields (Kill Password and Access Password within the RESERVED bank).

When reading and writing tag memory, all data operations are performed in one-word (2 bytes, 16 bits) increments, and only on word boundaries within a single bank. Some masking operations allow addressing memory down to the nibble (1/2 byte, 4 bits) or bit level.



2.2 Enumerations

2.2.1 Bank

The Bank enumeration is used to specify a particular memory bank, for locking, reading, and writing. See the Gen2 memory diagram at the start of this chapter for details of each bank.

```
public enum Bank{
    RESERVED,
    EPC,
    TID,
    USER
}
```

2.2.2 Target

Controls which Gen2 protocol A/B target is used when performing tag inventories.

```
public enum Target {
    A,
    B
}
```

2.2.3 Session

Controls which Gen2 protocol session is used when performing tag inventories.

```
public enum Session {
    S0,
    S1,
    S2,
    S3
}
```

2.2.4 LockType

Controls the lock type to use for lock/unlock operations.

```
public enum LockType {
    LOCK,
    UNLOCK,
    PERMALOCK,
    PERMAUNLOCK
}
```

2.2.5 RFIDInfo

Defines the available RFID subsystem information.

```
public enum RFIDInfo {
    FIRMWARE_VER,
    HARDWARE_VER,
    MODULE_ID,
    REGION,
}
```

2.3 Connecting to RFID Module

The Alien API's `RFIDReader` class allows controlling the RFID module. Using the `RFIDReader` instance, an application can adjust radio parameters, and execute tag operations (inventory, read, write, kill, lock) according to the ISO 18000 EPC Class 1 Generation 2 (Gen2) tag protocol.

Obtaining `RFIDReader` instance:

```
// Obtain RFIDReader instance
RFIDReader reader = RFID.open();
```

The `RFID.open()` method returns a **global** instance of `RFIDReader` which will be used by all `RFIDReader` methods to control the RFID module.

When finished using the `RFIDReader`, you should call the `close()` method to release the `RFIDReader` resources.

```
// Release RFIDReader instance and resources
reader.close();
```

2.4 Tag Operations

2.4.1 Mask

The Alien API provides the `Mask` class to give an application the ability to filter which tags participate in RFID operations, such as inventory, read, write, kill, and lock.

Constructor:

```
public Mask(Bank bank, int bitOffset, int bitLength, String data)
```

Parameters:

| | |
|------------------------|--------------------------------------|
| <code>bank</code> | which bank for masking |
| <code>bitOffset</code> | the bit offset where mask data start |
| <code>bitLength</code> | the bit length of the mask data |
| <code>pattern</code> | the data in hex string to filter |

Example:

To mask on a tag which has the EPC bank value of "11AA" (16 bits) at the bit offset of 32, set the mask as follows:

```
Mask mask = new Mask(Bank.EPC, 32, 16, "11AA");
```

No filtering:

The `Mask` class has the static `NONE` instance which disables masking for tag operations.

Example:

```
Mask mask = Mask.NONE;
```


Mask on the EPC starting with a pattern:

For more convenience, the `Mask` class has a static `maskEPC()` method to create masks that would match tags that have EPC start with a certain data string.

```
public static Mask maskEPC(String data)
```

Parameters:

`data` the data the tag's EPC starts with

Returns:

`Mask` object that contains mask information

Example:

To mask on tags that have an EPC starting with "11AA".

```
Mask mask = Mask.maskEPC("11AA");
```

2.4.2 Reading a single tag

The `RFIDReader` class has the `read` method to perform an inventory that would read a single tag.

```
public RFIDResult read(Mask mask [optional]) throws ReaderException
```

Parameters:

`mask` the mask to use when reading tags

Returns:

`RFIDResult` object that contains `Tag` object.
Calling `RFIDResult.getData()` will return a `Tag` object that contains EPC and RSSI information

Exceptions:

`ReaderException` is raised if the operation fails.

Example:

To read one tag with no filtering

```
RFIDReader reader = RFID.open();
RFIDResult result = reader.read();
if (result.isSuccess()) {
    Tag tag = (Tag)result.getData();
    String epc = tag.getEPC();
    double rssi = tag.getRSSI();
}
```

2.4.3 Continuous Inventory

The `RFIDReader` class provides the `inventory` method to initiate a continuous inventory and `stop` method to stop it.

The `RFIDCallback` **interface** is used to retrieve Tag information from the continuous inventory operation, via the `onTagRead` method callback. The caller needs to implement this method to receive tag information.

The `RFIDReader` class provides the `isRunning` method to check if continuous inventory is running.

```
public void inventory(RFIDCallback callback, Mask mask [optional])
    throws ReaderException
```

Parameters:

`callback` the callback to receive tag information.
`mask` the mask object for filtering

Returns:

`Tag` object that contains EPC and RSSI information. For convenience, the `Tag` object has its own methods to execute operations for that specific tag.

Exceptions:

`ReaderException` is raised if there is an error.

NOTE:

- Continuous inventories run **asynchronously**.
- The reader cannot respond to other commands during continuous inventories. It is necessary to stop the continuous inventory before executing other commands. An `RFIDBusyException` will be raised if you attempt to execute other commands while the continuous inventory is running.

```
public interface RFIDCallback {
    void onTagRead(Tag tag);
}
```

Parameters:

`Tag` - The `Tag` object that contains tag information.

NOTE: Only EPC and RSSI information can be obtained from the tag object while inside the callback or while continuous inventory is running. To perform other operations on the tag object you must stop the continuous inventory.

```
public void stop() throws ReaderException
```

Exceptions:

`ReaderException` is raised if there is an error.

```
public boolean isRunning()
```

Returns:

`boolean` true if a continuous inventory is running.

Example:

To perform a continuous inventory for tags which EPCs start with "1122"

```
RFIDReader reader = RFID.open();
reader.inventory(new RFIDCallback() {
    @Override
    public void onTagRead(Tag tag) {
        String epc = tag.getEPC();
        double rssi = tag.getRSSI();
    }
},
Mask.maskEPC("1122")
);
```

To stop continuous inventory:

```
reader.stop();
```

To check if continuous inventory is running:

```
boolean isRunning = reader.isRunning();
```

2.4.4 Reading and Writing tag's memory

The `RFIDReader` class provides `read` and `write` methods to read and write data from and to specific locations of tag's memory.

```
public RFIDResult read(Bank bank, int wordPointer, int wordCount,
    Mask mask [optional], String accessPassword [optional]) throws ReaderException
```

Parameters:

| | |
|-----------------------------|--|
| <code>bank</code> | the memory bank to read data from |
| <code>wordPointer</code> | the offset within the bank where to read the data from (in word units) |
| <code>wordCount</code> | the number of words to read |
| <code>mask</code> | the mask for filtering |
| <code>accessPassword</code> | the access password used when accessing protected memory. If not using password, pass this parameter as an empty string ("") |

Returns:

| | |
|-------------------------|---|
| <code>RFIDResult</code> | the results of the operation. Call <code>isSuccess()</code> on the <code>RFIDResult</code> object to check if the operation succeed. Call <code>getData()</code> on the <code>RFIDResult</code> object to get data read from the tag. |
|-------------------------|---|

Exceptions:

`ReaderException` is raised if there is an error.

```
public RFIDResult write(Bank bank, int wordOffset, String data,
    Mask mask [optional], String accessPassword [optional]) throws ReaderException
```

Parameters:

| | |
|----------------|---|
| bank | the memory bank to write data to |
| wordOffset | the offset within the bank where to write data to (in word units) |
| data | the data as a hex string to write |
| mask | the mask for filtering |
| accessPassword | the access password used when accessing protected memory. If not using password, passing this parameter as empty string (""). |

Returns:

| | |
|------------|---|
| RFIDResult | the results of the operation. Call <code>isSuccess()</code> on the <code>RFIDResult</code> object to check if the operation succeeded. |
|------------|---|

Exceptions:

`ReaderException` is raised if there is an error.

Example: To write and read a word from the USER bank

```
RFIDReader reader = RFID.open();
// Write "AABB" to the beginning of the USER memory bank of the tag which
// EPC starts with "1122"
RFIDResult writeResult = reader.write(Bank.USER,0,"AABB",Mask.maskEPC("1122"));
if (writeResult.isSuccess()) {
    // Write operation succeeded.
}

// Read 1 word from the beginning of the USER memory bank of the tag which
// EPC starts with "1122"
RFIDResult readResult = reader.read(Bank.USER,0,1,Mask.maskEPC("1122"));
if (readResult.isSuccess()) {
    // Read operation succeeded.
    String data = readResult.getData(); // data returned as a hex string
}
```

2.4.5 Lock Fields

The Alien API provides the `LockFields` class to be used as a parameter for tag lock/unlock operations. A `LockFields` object specifies which tag's memory fields will be affected when executing lock operations.

The `LockFields` class defines the following fields: `ACCESS_PWD`, `KILL_PWD`, `EPC` and `USER`

Descriptions:

| | |
|-------------------------|---|
| <code>ACCESS_PWD</code> | Access Password is used to protect tag memory from access. The Access Password is stored within the RESERVED memory bank. |
| <code>KILL_PWD</code> | Kill Password is used when kill a tag. The Kill Password is stored within RESERVED memory bank. |
| <code>EPC</code> | EPC field, same as EPC bank. |
| <code>USER</code> | USER field, same as USER bank |

Constructor:

```
public LockFields(int fields)
```

Parameters:

`fields` a bitmap of fields to be locked

Example:

Create a `LockFields` bitmap specifying EPC and USER lock fields:

```
LockFields fields = new LockFields(
    LockFields.EPC | LockFields.USER
);
```

2.4.6 Locking and Unlocking Tag Memory

The `RFIDReader` class provides `lock` method to lock, unlock, permlock or permaunlock tag memory fields.

```
public RFIDResult lock(LockFields fieldBitmap, LockType lockType,
    Mask mask [optional], String accessPassword [optional]) throws ReaderException
```

Parameters:

| | |
|-----------------------------|--|
| <code>fieldBitmap</code> | a bitmap of fields to be locked |
| <code>lockType</code> | lock type (lock, unlock, permalock, permaunlock) |
| <code>mask</code> | the mask object for filtering |
| <code>accessPassword</code> | the access password used when accessing protected memory |

Returns:

`RFIDResult` the results of the operation.
Call `isSuccess()` on the `RFIDResult` object to check if the operation succeeded.

Exceptions:

`ReaderException` is raised if there is an error.

NOTE:

- If a field is locked, a correct access password must be supplied in order to write to the memory
- If the EPC or USER fields are locked, they are **read-only** without knowing the access password. To write, you are required to pass the correct access password.
- If the ACCESS_PWD or KILL_PWD fields are locked, they are not accessible at all (**cannot read or write**) without knowing the access password.
- To unlock, it is necessary to know the correct access password.
- Use caution when using `lockType PERMALOCK` or `PERMAUNLOCK` – the results are **permanent!** If a field is permanently locked, it cannot be unlocked. If a field is permanently unlocked, it cannot be locked.

```

RFIDReader reader = RFID.open();
// Lock EPC and USER fields for the tags which EPC start with "1122"
LockFields fields = new LockFields(LockFields.EPC | LockFields.USER);
RFIDResult lockResult = reader.lock(fields, LockType.LOCK, Mask.maskEPC("1122"));
if (lockResult.isSuccess()) {
    // Lock operation succeed.
}

// Unlock EPC and USER fields for the tags which EPC start with "1122"
RFIDResult unlockResult = reader.lock(fields, LockType.UNLOCK,
    Mask.maskEPC("1122"), accessPassword);
if (unlockResult.isSuccess()) {
    // Unlock operation succeed.
}

```

2.4.7 Kill tags

The `RFIDReader` class provides the `kill` method to kill RFID tags. This prevents the tags from communicating further with an RFID reader. Once a tag is killed, that tag can't be recovered.

```

public RFIDResult kill(String killPassword, Mask mask [optional])
    throws ReaderException

```

Parameters:

`killPassword` non-zero kill password
`mask` mask object for filtering

Returns:

`RFIDResult` the result of the operation.
 Call `isSuccess()` on the `RFIDResult` object to check if the operation succeeded.

Exceptions:

`ReaderException` is raised if there is an error.

2.4.8 Tag class

The Alien API library provides the `Tag` class to provide more convenience for the developer to run operations for a specific tag.

A `Tag` object is returned by reading a single tag, or via a callback from a continuous inventory. The data returned inside the `RFIDResult` is a `Tag` object.

```
RFIDReader reader = RFID.open();
RFIDResult result = reader.read(Mask.NONE);
Tag tag = (Tag)result.getData();
String epc = tag.getEPC();
double rssi = tag.getRSSI();
```

The `Tag` class provides wrapper methods, below, which allow you to run all operations for the tag directly on the tag itself. You don't need to supply a mask for that tag as it's going to automatically mask on the tag's EPC when calling an `RFIDReader` method.

- **Obtaining Data From The Tag Object (without issuing RFID commands):**

```
public String getEPC(); // tag's EPC
public double getRSSI(); // tag's Returned Signal Strength
```

- **Executing Rfid Operations On The Tag Object:**

Write EPC:

```
public RFIDResult writeEPC(String epc) throws ReaderException;
```

Read/Write ACCESS PWD:

```
public RFIDResult readAccessPwd() throws ReaderException
public RFIDResult writeAccessPwd() throws ReaderException
```

Read/Write KILL PWD:

```
public RFIDResult readKillPwd() throws ReaderException
public RFIDResult writeKillPwd() throws ReaderException
```

Read TID memory bank:

```
public RFIDResult getTID() throws ReaderException
```

Read/Write USER memory bank:

```
public RFIDResult readUser() throws ReaderException
public RFIDResult writeUser() throws ReaderException
```

Read/Write tags's memory:

```
public RFIDResult read(Bank bank, int wordOffset, int wordCount,
    String accessPassword [optional])
public RFIDResult write(Bank bank, int wordOffset, String data,
    String accessPassword [optional]) throws ReaderException
```

Lock/Unlock tag's memory:

```
public RFIDResult lock(LockFields fields, LockType lockType,
    String accessPassword [optional]) throws ReaderException
```

Kill tag:

```
public RFIDResult kill(String killPassword) throws ReaderException
```

2.5 Setting RFID Parameters

The `RFIDReader` class includes methods to configure RFID related parameters, as well as to obtain information about the device.

2.5.1 Transmit Power

The `RFIDReader` class provides `getPower` and `setPower` methods to manage the RFID module's transmit power.

```
public int getPower() throws ReaderException
```

Parameters:

None

Returns:

`int` the RFID module's power attenuation in dBm

Exceptions:

`ReaderException` is raised if there is an error.

```
public void setPower(int power) throws ReaderException
```

Parameters:

`power` the RFID module's transmit power in dBm (1-30)

Exceptions:

`ReaderException` is raised if there is an error.

`InvalidParamException` is raised if the parameter value is not valid

2.5.2 Q Parameter

The `RFIDReader` class provides `getQ` and `setQ` methods to configure the starting Q parameter value that is used in the Gen2 protocol to read tags. It indicates approximately how many tags to expect (2^Q).

```
public int getQ() throws ReaderException
```

Parameters:

None

Returns:

`int` the starting Q parameter value

Exceptions:

`ReaderException` is raised if there is an error.

```
public void setQ(int QValue) throws ReaderException
```

Parameters:

`QValue` starting Q parameter value. Valid Q values are from 0 to 15.

Exceptions:

`ReaderException` is raised if there is an error.

`InvalidParamException` is raised if the parameter value is not valid

2.5.3 Session Parameter

The `RFIDReader` class provides `getSession` and `setSession` methods to configure the session parameter of the Gen2 protocol to keep track of inventoried tags. Please refer to the Gen2 protocol specification for information on the Session value.

```
public Session getSession() throws ReaderException
```

Parameters:

None

Returns:

`Session` the Gen2 protocol Session parameter

Exceptions:

`ReaderException` is raised if there is an error.

```
public void setSession(Session session) throws ReaderException
```

Parameters:

`Session` the Gen2 protocol Session parameter

Exceptions:

`ReaderException` is raised if there is an error.

2.5.4 Inventory Target

The `RFIDReader` class provides `getTarget` and `setTarget` methods to configure the Inventory target parameter used in the Gen2 protocol.

```
public Target getTarget() throws ReaderException
```

Parameters:

None

Returns:

`Target` the Gen2 protocol inventory target parameter

Exceptions:

`ReaderException` is raised if there is an error.

```
public void setTarget(Target target) throws ReaderException
```

Parameters:

`target` the Gen2 protocol inventory target parameter

Exceptions:

`ReaderException` is raised if there is an error.

2.5.5 RFID Subsystem Information

The `RFIDReader` class provides `getInfo` methods to obtain the device information, such as hardware version, firmware version, and model number.

```
public String getInfo(DeviceInfo type)
```

Parameters:

`type` the information type to obtain

Return:

`String` device information returned as a string, or null if the information is not available

```
public Map<DeviceInfo, String> getInfo()
```

Parameters:

None

Return:

`Map` a Map object containing all available device information

Example

```
RFIDReader reader = RFID.open();
String rfidHardwareVersion = reader.getInfo(DeviceInfo.RFID_HARDWARE_VER);
String rfidFirmwareVersion = reader.getInfo(DeviceInfo.RFID_FIRMWARE_VER);
String rfidModuleId = reader.getInfo(DeviceInfo.RFID_ID);
```

2.5.6 Device Information

The `DeviceInfo` class provides several methods to obtain the device information, such as the device model and the device ID.

```
public String getDeviceID()
```

Return:

`String` device ID returned as a string.

```
public String getModel()
```

Return:

`String` device model returned as a string.

Example

```
DeviceInfo dev = new DeviceInfo(context);
String id = dev.getDeviceID();
String model = dev.getModel();
```

3 Barcode Reader

3.1 Introduction

The Alien API library includes classes to control and communicate with the Barcode Scanner module. Use `BarcodeReader` instance's methods to communicate with the Barcode Scanner.

3.2 Controlling Barcode Reader

The `BarcodeReader` allows scanning 1D and 2D barcodes.

```
public BarcodeReader(Context context)
```

Parameters:

`context` the Android context where the `BarcodeReader` instance is used

Example:

```
// Create BarcodeReader instance
BarcodeReader barcodeReader = new BarcodeReader(androidContext);
```

3.2.1 Scan 1D and 2D barcode

The `start` and `stop` methods control the Barcode Reader module to start and stop barcode scanning.

NOTE: The scanning will automatically stop after a barcode has been successfully scanned.

The `BarcodeCallback` **interface** is used to retrieve barcode information from the scanning operation, via the `onBarcodeRead` method callback. The caller needs to implement this method in order to receive barcode information.

The `isRunning` method is used to determine if barcode scanning is currently in progress.

```
public void start(BarcodeCallback callback)
```

Parameters:

`callback` the callback to receive barcode information

Returns:

None

```
public void stop()
```

Parameters:

None

Returns:

None

```
public boolean isRunning()
```

Parameters:

None

Returns:

true if the barcode scanning is currently in progress

Example:

```
// Start barcode scan
barcodeReader.start(new BarcodeCallback() {
    @Override
    public void onBarcodeRead(String barcode) {
        String detectedBarcode = barcode;
        // Scan will automatically stop after successfully scanning a barcode
    }
});
```

Stop barcode scan:

```
// Stop barcode scan
barcodeReader.stop();
```

3.2.2 Controlling Barcode Parameters

The default Barcode Reader settings can be changed using the methods listed below.

NOTE: Refer to the next section for the list of Barcode Reader parameters and descriptions of their values.

```
public boolean setParameter(int param, int value)
```

Parameters:

param barcode parameter ID. Use -1 to reset all parameters to the default values
value barcode parameter value

Returns:

boolean true if success

```
public int getParameter(int param)
```

Parameters:

param barcode parameter ID

Returns:

int parameter value or -1 if failed

```
public void setAllSymbologies(boolean enable)
```

Parameters:

enable boolean true/false to enable/disable all supported symbologies
NOTE: enabling all symbologies initializes their parameters to default values.
Use `setParameter()` method to set a custom parameter value.

Returns:

None

3.3 Symbology Parameters

3.3.1 List of Barcode Symbology Parameters

| Parameter | ID | Default |
|--|----------|--------------------|
| UPC/EAN | | |
| UPC-A | 1 | Enable |
| UPC-E | 2 | Enable |
| UPC-E1 | 12 | Disable |
| EAN-8/JAN 8 | 4 | Enable |
| EAN-13/JAN 13 | 3 | Enable |
| Bookland EAN | 83 | Disable |
| Decode UPC/EAN/JAN Supplementals (2 and 5 digits) | 16 | Ignore |
| UPC/EAN/JAN Supplemental Redundancy | 80 | 10 |
| Decode UPC/EAN/JAN Supplemental AIM ID | 672 | Combined |
| Transmit UPC-A Check Digit | 40 | Enable |
| Transmit UPC-E Check Digit | 41 | Enable |
| Transmit UPC-E1 Check Digit | 42 | Enable |
| UPC-A Preamble | 34 | System Character |
| UPC-E Preamble | 35 | System Character |
| UPC-E1 Preamble | 36 | System Character |
| Convert UPC-E to A | 37 | Disable |
| Convert UPC-E1 to A | 38 | Disable |
| EAN-8/JAN-8 Extend | 39 | Disable |
| Bookland ISBN Format | 576 | ISBN-10 |
| UCC Coupon Extended Code | 85 | Disable |
| Coupon Report | 730 | New Coupon Symbols |
| ISSN EAN | 617 | Disable |
| Code 128 | | |
| Code 128 | 8 | Enable |
| Set Length(s) for Code 128 | 209, 210 | Any Length |
| GS1-128 (formerly UCC/EAN-128) | 14 | Enable |
| ISBT 128 | 84 | Enable |
| ISBT Concatenation | 577 | Disable |
| Check ISBT Table | 578 | Enable |
| ISBT Concatenation Redundancy | 223 | 10 |
| Code 39 | | |
| Code 39 | 0 | Enable |
| Trioptic Code 39 | 13 | Disable |
| Convert Code 39 to Code 32 (Italian Pharmacy Code) | 86 | Disable |
| Code 32 Prefix | 231 | Disable |
| Set Length(s) for Code 39 | 18 19 | 2 to 55 |
| Code 39 Check Digit Verification | 48 | Disable |
| Transmit Code 39 Check Digit | 43 | Disable |
| Code 39 Full ASCII Conversion | 17 | Disable |
| Code 93 | | |
| Code 93 | 9 | Disable |
| Set Length(s) for Code 93 | 26, 27 | 4 to 55 |
| Code 11 | | |
| Code 11 | 10 | Disable |
| Set Lengths for Code 11 | 28, 29 | 4 to 55 |
| Code 11 Check Digit Verification | 52 | Disable |
| Transmit Code 11 Check Digit(s) | 47 | Disable |
| Interleaved 2 of 5 (ITF) | | |
| Interleaved 2 of 5 (ITF) | 6 | Enable |
| Set Lengths for I 2 of 5 | 22, 23 | 14 |
| I 2 of 5 Check Digit Verification | 49 | Disable |
| Transmit I 2 of 5 Check Digit | 44 | Disable |
| Convert I 2 of 5 to EAN 13 | 82 | Disable |
| Discrete 2 of 5 (DTF) | | |
| Discrete 2 of 5 | 5 | Disable |
| Set Length(s) for D 2 of 5 | 20, 21 | 12 |
| Codabar (NW - 7) | | |
| Codabar | 7 | Disable |

| | | |
|--|---------|------------------|
| Set Lengths for Codabar | 24 25 | 5 to 55 |
| CLSI Editing | 54 | Disable |
| NOTIS Editing | 55 | Disable |
| MSI | | |
| MSI | 11 | Disable |
| Set Length(s) for MSI | 30, 31 | 4 to 55 |
| MSI Check Digits | 50 | One |
| Transmit MSI Check Digit | 46 | Disable |
| MSI Check Digit Algorithm | 51 | Mod 10/Mod 10 |
| Chinese 2 of 5 | 408 | Disable |
| Korean 3 of 5 | 581 | Disable |
| Matrix 2 of 5 | | |
| Matrix 2 of 5 | 618 | Disable |
| Matrix 2 of 5 Lengths | 619 620 | 14 |
| Matrix 2 of 5 Redundancy | 621 | Disable |
| Matrix 2 of 5 Check Digit | 622 | Disable |
| Transmit Matrix 2 of 5 Check Digit | 623 | Disable |
| Inverse 1D | 586 | Regular |
| Postal Codes | | |
| US Postnet | 89 | Enable |
| US Planet | 90 | Enable |
| Transmit US Postal Check Digit | 95 | Enable |
| UK Postal | 91 | Enable |
| Transmit UK Postal Check Digit | 96 | Enable |
| Japan Postal | 290 | Enable |
| Australia Post | 291 | Enable |
| Australia Post Format | 718 | Autodiscriminate |
| Netherlands KIX Code | 326 | Enable |
| USPS 4CB/One Code/Intelligent Mail | 592 | Disable |
| UPU FICS Postal | 611 | Disable |
| GS1 DataBar (formerly RSS, Reduced Space Symbology) | | |
| GS1 DataBar-14 | 338 | Enable |
| GS1 DataBar Limited | 339 | Disable |
| GS1 DataBar Limited Security Level | 728 | 3 |
| GS1 DataBar Expanded | 340 | Disable |
| Convert GS1 DataBar to UPC/EAN | 397 | Disable |
| Composite | | |
| Composite CC-C | 341 | Disable |
| Composite CC-A/B | 342 | Disable |
| Composite TLC-39 | 371 | Disable |
| UPC Composite Mode | 344 | Never Linked |
| GS1-128 Emulation Mode for UCC/EAN Composite Codes | 427 | Disable |
| 2D Symbologies | | |
| PDF417 | 15 | Enable |
| MicroPDF417 | 227 | Disable |
| Code 128 Emulation | 123 | Disable |
| Data Matrix | 292 | Enable |
| Data Matrix Inverse | 588 | Regular |
| Decode Mirror Images (Data Matrix Only) | 537 | Never |
| Maxicode | 294 | Enable |
| QR Code | 293 | Enable |
| QR Inverse | 587 | Regular |
| MicroQR | 573 | Enable |
| Aztec | 574 | Enable |
| Aztec Inverse | 589 | Regular |
| Han Xin | 1167 | Disable |
| Han Xin Inverse | 1168 | Regular |
| Symbology-Specific Security Levels | | |
| Redundancy Level | 78 | 1 |
| Security Level | 77 | 1 |
| Intercharacter Gap Size | 381 | Normal |

3.3.2 Barcode Symbology Parameter Values

NOTE: Asterisk (*) designates the default value

| | | |
|------------|--|---|
| 1 | UPC-A | • 1* - Enable • 0 - Disable |
| 2 | UPC-E | • 1* - Enable • 0 - Disable |
| 12 | UPC-E1 | • 0* - Disable • 1 - Enable NOTE: UPC-E1 is not UCC (Uniform Code Council) approved symbology |
| 4 | EAN-8/JAN-8 | • 1* - Enable • 0 - Disable |
| 3 | EAN-13/JAN-13 | • 1* - Enable • 0 - Disable |
| 83 | Bookland EAN | • 0* - Disable • 1 - Enable NOTE: If you enable Bookland EAN, select a Bookland ISBN Format. Also select either Decode UPC/EAN Supplementals, Autodiscriminate UPC/EAN Supplementals, or Enable 978/979 Supplemental Mode in Decode UPC/EAN/JAN Supplementals |
| 16 | Decode UPC/EAN/JAN Supplementals | Supplementals are barcodes appended according to specific format conventions (e.g., UPC A+2, UPC E+2, EAN 13+2). Select one of the following options: • 0* - Ignore UPC/EAN with Supplementals - if the decoder is presented with a UPC/EAN plus supplemental symbol, the decoder decodes UPC/EAN and ignores the supplemental characters. • 1 - Decode UPC/EAN with Supplementals - the decoder only decodes UPC/EAN symbols with supplemental characters, and ignores symbols without supplementals. • 2 - Autodiscriminate UPC/EAN Supplementals - decoder decodes UPC/EAN symbols with supplemental characters immediately. If the symbol does not have a supplemental, the decoder must decode the bar code the number of times set via UPC/EAN/JAN Supplemental Redundancy before transmitting its data to confirm that there is no supplemental. If you select one of the following Supplemental Mode options, the decoder immediately transmits EAN-13 bar codes starting with that prefix that have supplemental characters. If the symbol does not have a supplemental, the decoder must decode the bar code the number of times set via UPC/EAN/JAN Supplemental Redundancy before transmitting its data to confirm that there is no supplemental. The decoder transmits UPC/EAN bar codes that do not have that prefix immediately. • 4 - Enable 378/379 Supplemental Mode • 5 - Enable 978/979 Supplemental Mode NOTE: If you select 978 Supplemental Mode and are scanning Bookland EAN bar codes, see Enable/Disable Bookland EAN to enable Bookland EAN, and select a format using Bookland ISBN Format • 7 - Enable 977 Supplemental Mode • 6 - Enable 414/419/434/439 Supplemental Mode • 8 - Enable 491 Supplemental Mode • 3 - Enable Smart Supplemental Mode - applies to EAN-13 bar codes starting with any prefix listed previously. • 9 - Supplemental User-Programmable Type 1 - applies to EAN-13 barcodes starting with a 3-digit user-defined prefix. Set this 3-digit prefix using User-Programmable Supplementals. • 10 - Supplemental User-Programmable Type 1 and 2 - applies to EAN-13 bar codes starting with either of two 3-digit user-defined prefixes. Set the 3-digit prefixes using User-Programmable Supplementals • 11 - Smart Supplemental Plus User-Programmable 1 - applies to EAN-13 barcodes starting with any prefix listed previously or the user-defined prefix set using User-Programmable Supplementals • 12 - Smart Supplemental Plus User-Programmable 1 and 2 - applies to EAN-13 barcodes starting with any prefix listed previously or one of the two user-defined prefixes set using User-Programmable Supplementals NOTE: To minimize the risk of invalid data transmission, select either to decode or ignore supplemental characters |
| 579 580 | User-Programmable Supplemental 1 Supplemental 2 | If you selected a Supplemental User-Programmable option from Decode UPC/EAN/JAN Supplementals, select User-Programmable Supplemental 1 to set the 3-digit prefix select User-Programmable Supplemental 2 to set a second 3-digit prefix |
| 80 | UPC/EAN/JAN Supplemental Redundancy | With Autodiscriminate UPC/EAN/JAN Supplementals selected, this option adjusts the number of times a symbol without supplementals is decoded before transmission. The range is from two to 30 times. Five or above is recommended when decoding a mix of UPC/EAN/JAN symbols with and without supplementals, and the autodiscriminate option is selected. The default is set at 10 |

| | | |
|----------------|--|---|
| 672 | UPC/EAN/JAN Supplemental AIM ID Format | Select an output format when reporting UPC/EAN/JAN bar codes with Supplementals with Transmit Code ID Character set to AIM Code ID Character: <ul style="list-style-type: none"> • 0 - Separate - transmit UPC/EAN with supplementals with separate AIM IDs but one transmission, i.e.:]E<0 or 4><data>]E<1 or 2>[supplemental data] • 1* - Combined – transmit UPC/EAN with supplementals with one AIM ID and one transmission, i.e.:]E3<data+supplemental data> • 2 - Separate Transmissions - transmit UPC/EAN with supplementals with separate AIM IDs and separate transmissions, i.e.:]E<0 or 4><data>]E<1 or 2>[supplemental data] |
| 40 41 42 | Transmit Check Digit UPC-A UPC-E UPC-E1 | The check digit is the last character of the symbol used to verify the integrity of the data. Select whether to transmit the bar code data with or without the check digit. It is always verified to guarantee the integrity of the data. <ul style="list-style-type: none"> • 1* - Transmit Check Digit • 0 - Do Not Transmit Check Digit |
| 34 35 36 | Preamble UPC-A UPC-E UPC-E1 | Preamble characters are part of the UPC symbol, and include Country Code and System Character. There are three options for transmitting a preamble to the host device. Select the appropriate option to match the host system: <ul style="list-style-type: none"> • 1* - Transmit System Character Only (<SYSTEM CHARACTER> <DATA>) • 2 - Transmit System Character and Country Code ("0" for USA) (< COUNTRY CODE> <SYSTEM CHARACTER> <DATA>) • 0 - Transmit no preamble (<DATA>) |
| 37 38 | Convert UPC-E to UPC-A UPC-E1 to UPC-A | Enable this to convert UPC-E /E1 decoded data to UPC-A format before transmission. After conversion, the data follows UPC-A format and is affected by UPC-A programming selections (e.g., Preamble, Check Digit). When disabled, UPC-E/E1 decoded data is transmitted as UPC-E/E1 data, without conversion. <ul style="list-style-type: none"> • 0* - Do Not Convert to UPC-A (Disable) • 1 - Convert to UPC-A (Enable) |
| 39 | EAN-8/JAN-8 Extend | Enable this parameter to add five leading zeros to decoded EAN-8 symbols to make them compatible in format to EAN-13 symbols. Disable this to transmit EAN-8 symbols as is. <ul style="list-style-type: none"> • 0* - Disable EAN/JAN Zero Extend • 1 - Enable EAN/JAN Zero Extend |
| 576 | Bookland ISBN Format | If you enabled Bookland EAN using Enable/Disable Bookland EAN, select one of the following formats for Bookland data: <ul style="list-style-type: none"> • 0* - Bookland ISBN-10 - The decoder reports Bookland data starting with 978 in traditional 10-digit format with the special Bookland check digit for backward-compatibility. Data starting with 979 is not considered Bookland in this mode. • 1 - Bookland ISBN-13 - The decoder reports Bookland data (starting with either 978 or 979) as EAN-13 in 13-digit format to meet the 2007 ISBN-13 protocol. <p>NOTE: For Bookland EAN to function properly, first enable Bookland EAN using Enable/Disable Bookland EAN, then select either Decode UPC/EAN Supplementals, Autodiscriminate UPC/EAN Supplementals, or Enable 978/979 Supplemental Mode in Decode UPC/EAN/JAN Supplementals</p> |
| 85 | UCC Coupon Extended Code | Enable this parameter to decode UPC-A bar codes starting with digit '5', EAN-13 bar codes starting with digit '99', and UPC-A/EAN-128 Coupon Codes. UPCA, EAN-13, and EAN-128 must be enabled to scan all types of Coupon Codes. <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable <p>NOTE: Use the Decode UPC/EAN Supplemental Redundancy parameter to control auto discrimination of the EAN128 (right half) of a coupon code</p> |
| 730 | Coupon Report | Traditional coupon symbols (old coupon symbols) are composed of two bar codes: UPC/EAN and Code128. A new coupon symbol is composed of a single Databar Expanded bar code. The new coupon format offers more options for purchase values (up to \$999.99) and supports complex discount offers such as a second purchase requirement. <p>An interim coupon symbol also exists that contains both types of bar codes: UPC/EAN and Databar Expanded. This format accommodates both retailers that do not recognize or use the additional information included in the new coupon symbol, as well as those who can process new coupon symbols.</p> <ul style="list-style-type: none"> • 0 - Old Coupon Symbols - Scanning an old coupon symbol reports both UPC and Code 128, scanning an interim coupon symbol reports UPC, and scanning a new coupon symbol reports nothing (no decode). • 1* - New Coupon Symbols - Scanning an old coupon symbol reports either UPC or Code 128, and scanning an interim coupon symbol or a new coupon symbol reports Databar Expanded. • 2 - Both Coupon Formats - Scanning an old coupon symbol reports both UPC and Code 128, and scanning an interim coupon symbol or a new coupon symbol reports Databar Expanded. |
| 617 | ISSN EAN | • 0* - Disable • 1 - Enable |
| 8 | Code 128 | • 1* - Enable • 0 - Disable |

| | | |
|------------|---|--|
| 209 210 | Set Lengths for Code 128 Length1 Length2 | Length1 = 0..55, default=0 Length2 = 0..55, default=0 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Code 128 to decode either one or two discrete lengths, or a range of lengths. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Code 128 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0. • Two Discrete Lengths - To limit the decoding of Code 128 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode Code 128 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Code 128 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Code 128 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 14 | GS1-128 formerly UCC/ EAN-128 | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 84 | ISBT 128 | ISBT 128 is a variant of Code 128 used in the blood bank industry. If necessary, the host must perform concatenation of the ISBT data. <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 577 | ISBT Concatenation | <ul style="list-style-type: none"> • 0* - Disable ISBT Concatenation - The device does not concatenate pairs of ISBT codes it encounters. • 1 - Enable ISBT Concatenation - There must be two ISBT codes in order for the device to decode and perform concatenation. The device does not decode single ISBT symbols. • 2 - Auto discriminate ISBT Concatenation - The device decodes and concatenates pairs of ISBT codes immediately. If only a single ISBT symbol is present, the device must decode the symbol the number of times set via ISBT Concatenation Redundancy before transmitting its data to confirm that there is no additional ISBT symbol. |
| 578 | Check ISBT Table | The ISBT specification includes a table that lists several types of ISBT bar codes that are commonly used in pairs. If you enable ISBT Concatenation, enable Check ISBT Table to concatenate only those pairs found in this table. Other types of ISBT codes are not concatenated. <ul style="list-style-type: none"> • 1* - Enable Check ISBT Table • 0 - Disable Check ISBT Table |
| 223 | ISBT Concatenation Redundancy | With ISBT Concatenation set to Autodiscriminate, this option sets the number of times the device must decode an ISBT symbol before determining that there is no additional symbol. The range is from two to 20 times. The default is 10 |
| 0 | Code 39 | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 13 | Trioptic Code 39 | Trioptic Code 39 is a variant of Code 39 used in the marking of computer tape cartridges. Trioptic Code 39 symbols always contain six characters. <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable NOTE: Trioptic Code39 and Code39 Full ASCII cannot be enabled simultaneously |
| 86 | Convert Code 39 to Code 32 | Code 32 is a variant of Code 39 used by the Italian pharmaceutical industry. <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable NOTE: Code 39 must be enabled for this parameter to function |
| 231 | Code 32 Prefix | Enable or disable adding the prefix character "A" to all Code 32 bar codes. <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable NOTE: Convert Code 39 to Code 32 must be enabled to use this parameter |
| 18 19 | Set Lengths for Code 39 Length1 Length2 | Length1 = 0..55, default=2 Length2 = 0..55, default=55 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Code 39 to decode either one or two discrete lengths, or a range of lengths. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Code 39 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14 , Length2 = 0 . • Two Discrete Lengths - To limit the decoding of Code 39 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2 . For example, to decode Code 39 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Code 39 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Code 39 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 48 | Code 39 Check Digit Verification | Enable this to check the integrity of all Code 39 symbols to verify that the data complies with specified check digit algorithm. Only Code 39 symbols which include a modulo 43 check digit are decoded. Enable this feature if the Code 39 symbols contain a Modulo 43 check digit. <ul style="list-style-type: none"> • 0* - Disable Code 39 Check Digit Verification • 1 - Enable Code 39 Check Digit Verification |
| 43 | Transmit Code 39 Check Digit | <ul style="list-style-type: none"> • 0* - Do Not Transmit Code 39 Check Digit (Disable) • 1 - Transmit Code 39 Check Digit (Enable) NOTE: Code 39 Check Digit Verification must be enabled |

| | | |
|----------|--|--|
| 17 | Code 39 Full ASCII Conversion | Code 39 Full ASCII is a variant of Code 39 which pairs characters to encode the full ASCII character set. <ul style="list-style-type: none"> • 0* - Disable Code 39 Full ASCII • 1 - Enable Code 39 Full ASCII NOTE1: Trioptic Code 39 and Code 39 Full ASCII cannot be enabled simultaneously. NOTE2: Code 39 Full ASCII to Full ASCII Correlation is host-dependent. |
| 9 | Code 93 | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable |
| 26 27 | Set Lengths for Code 93 Length1 Length2 | Length1 = 0..55, default=4 Length2 = 0..55, default=55 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Code 93 to decode either one or two discrete lengths, or a range of lengths. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Code 93 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0 . • Two Discrete Lengths - To limit the decoding of Code 93 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode Code 93 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Code 93 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Code 93 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 10 | Code 11 | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable |
| 28 29 | Set Lengths for Code 11 Length1 Length2 | Length1 = 0..55, default=4 Length2 = 0..55, default=55 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Code 11 to decode either one or two discrete lengths, or a range of lengths. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Code 11 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0. • Two Discrete Lengths - To limit the decoding of Code 11 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode Code 11 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Code 11 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Code 11 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 52 | Code 11 Check Digit Verification | This feature allows the decoder to check the integrity of all Code 11 symbols to verify that the data complies with the specified check digit algorithm. This selects the check digit mechanism for the decoded Code 11 bar code. To enable this feature, set the number of check digits encoded in the Code 11 symbols: <ul style="list-style-type: none"> • 0* - Disable Code 11 Check Digit Verification • 1 - 1 Check Digit • 2 - 2 Check Digits |
| 47 | Transmit Code 11 Check Digits | <ul style="list-style-type: none"> • 0* - Do Not Transmit Code 11 Check Digit(s) (Disable) • 1 - Transmit Code 11 Check Digit(s) (Enable) NOTE: Code 11 Check Digit Verification must be enabled |
| 6 | Interleaved 2 of 5 | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 22 23 | Set Lengths for Interleaved 2 of 5 Length1 Length2 | Length1 = 0..55, default=14 Length2 = 0..55, default=0 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for I 2 of 5 to decode either one or two discrete lengths, or a length within a specific range. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of I 2 of 5 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0 . • Two Discrete Lengths - To limit the decoding of I 2 of 5 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode I 2 of 5 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only I 2 of 5 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode I 2 of 5 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 NOTE: Due to the construction of the I 2 of 5 symbology, it is possible for a scan line covering only a portion of the code to be interpreted as a complete scan, yielding less data than is encoded in the bar code. To prevent this, select specific lengths (one or two discrete lengths) for I 2 of 5 applications |
| 49 | Interleaved 2 of 5 Check Digit Verification | Enable this feature to check the integrity of all I 2 of 5 symbols to verify the data complies with either the specified Uniform Symbology Specification (USS), or the Optical Product Code Council (OPCC) check digit algorithm. <ul style="list-style-type: none"> • 0* - Disable • 1 - USS Check Digit • 2 - OPCC Check Digits |

| | | |
|----------|---|--|
| 44 | Transmit Interleaved 2 of 5 Check Digit | <ul style="list-style-type: none"> • 0*- Do Not Transmit I 2 of 5 Check Digit (Disable) • 1 - Transmit I 2 of 5 Check Digit (Enable) |
| 82 | Convert Interleaved 2 of 5 to EAN-13 | <p>Enable this parameter to convert 14-character Interleaved 2 of 5 codes to EAN-13, and transmit to the host as EAN-13. To accomplish this, the Interleaved 2 of 5 code must be enabled, and the code must have a leading zero and a valid EAN-13 check digit.</p> <ul style="list-style-type: none"> • 0*- Do Not Convert I 2 of 5 to EAN-13 (Disable) • 1 - Convert I 2 of 5 to EAN-13 (Enable) |
| 5 | Discrete 2 of 5 | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 20 21 | Set Lengths for Discrete 2 of 5 Length1 Length2 | <p>Length1 = 0..55, default=12 Length2 = 0..55, default=0</p> <p>The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for D 2 of 5 to decode either one or two discrete lengths, or a range of lengths.</p> <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of D 2 of 5 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0. • Two Discrete Lengths - To limit the decoding of D 2 of 5 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode D 2 of 5 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2 . • Length Within Range - To decode only D 2 of 5 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode D 2 of 5 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 <p>NOTE: Due to the construction of the D 2 of 5 symbology, it is possible for a scan line covering only a portion of the code to be interpreted as a complete scan, yielding less data than is encoded in the bar code. To prevent this, select specific lengths (one or two discrete lengths) for D 2 of 5 applications.</p> |
| 7 | Codabar | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 24 25 | Set Lengths for Codabar Length1 Length2 | <p>Length1 = 0..55, default=5 Length2 = 0..55, default=55</p> <p>The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Codabar to decode either one or two discrete lengths, or a range of lengths.</p> <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Codabar to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14 , Length2 = 0 . • Two Discrete Lengths - To limit the decoding of Codabar to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2 . For example, to decode Codabar codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Codabar codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Codabar codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 54 | CLSI Editing | <p>Enable this parameter to strip the start and stop characters and insert a space after the first, fifth, and tenth characters of a 14-character Codabar symbol.</p> <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable <p>NOTE: Symbol length does not include start and stop characters</p> |
| 55 | NOTIS Editing | <p>Enable this parameter to strip the start and stop characters from a decoded Codabar symbol.</p> <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 11 | MSI | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 30 31 | Set Lengths for MSI Length1 Length2 | <p>Length1 = 0..55, default=4 Length2 = 0..55, default=55</p> <p>The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for MSI to decode either one or two discrete lengths, or a length within a specific range.</p> <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of MSI to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0 . • Two Discrete Lengths - To limit the decoding of MSI to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode MSI codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only MSI codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode MSI codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 <p>NOTE: Due to the construction of the MSI symbology, it is possible for a scan line covering only a portion of the code to be interpreted as a complete scan, yielding less data than is encoded in the bar code. To prevent this, select specific lengths (one or two discrete lengths) for MSI applications.</p> |

| | | |
|------------|---|---|
| 50 | MSI Check Digits | With MSI symbols, one check digit is mandatory and always verified by the reader. The second check digit is optional. If the MSI codes include two check digits, select Two Check Digits to enable verification of the second check digit: <ul style="list-style-type: none"> • 0* - One MSI Check Digit • 1 - Two MSI Check Digits See MSI Check Digit Algorithm to select second digit algorithms |
| 46 | Transmit MSI Check Digit(s) | <ul style="list-style-type: none"> • 0* - Do Not Transmit MSI Check Digit(s) (Disable) • 1 - Transmit MSI Check Digit(s) (Enable) |
| 51 | MSI Check Digit Algorithm | Select one of two algorithms for the verification of the second MSI check digit: <ul style="list-style-type: none"> • 1* - MOD 10/MOD 10 • 0 - MOD 10/MOD 11 |
| 408 | Chinese 2 of 5 | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable |
| 581 | Korean 3 of 5 | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable NOTE: The length for Korean 3 of 5 is fixed at 6 |
| 618 | Matrix 2 of 5 | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable |
| 619 620 | Set Lengths for Matrix 2 of 5 Length1 Length2 | Length1 = 0..55, default=14 Length2 = 0..55, default=0 The length of a code refers to the number of characters (i.e., human readable characters), including check digit(s) the code contains. Assign lengths for Matrix 2 of 5 to decode either one or two discrete lengths, or a range of lengths. <ul style="list-style-type: none"> • One Discrete Length - To limit the decoding of Matrix 2 of 5 to one specific length, assign this length to the Length1 parameter and 0 to the Length2 parameter. For example, for fixed length 14, set Length1 = 14, Length2 = 0. • Two Discrete Lengths - To limit the decoding of Matrix 2 of 5 to either of two specific lengths, assign the greater length to the Length1 parameter and the lesser to Length2. For example, to decode Matrix 2 of 5 codes of either 2 or 14 characters only, set Length1 = 14, Length2 = 2. • Length Within Range - To decode only Matrix 2 of 5 codes that fall within a specific length range, assign the lesser length to the Length1 parameter and the greater to the Length2 parameter. For example, to decode Matrix 2 of 5 codes of length 4 through 12 characters, set Length1 = 4, Length2 = 12 |
| 621 | Matrix 2 of 5 Redundancy | <ul style="list-style-type: none"> • 0* - Disable Matrix 2 of 5 Redundancy • 1 - Enable Matrix 2 of 5 Redundancy |
| 622 | Matrix 2 of 5 Check Digit | The check digit is the last character of the symbol used to verify the integrity of the data. <ul style="list-style-type: none"> • 0* - Disable Matrix 2 of 5 Check Digit • 1 - Enable Matrix 2 of 5 Check Digit |
| 623 | Transmit Matrix 2 of 5 Check Digit | <ul style="list-style-type: none"> • 0* - Do Not Transmit Matrix 2 of 5 Check Digit • 1 - Transmit Matrix 2 of 5 Check Digit |
| 586 | Inverse 1D | Set the 1D inverse decoder setting: <ul style="list-style-type: none"> • 0* - Regular Only - the decoder decodes regular 1D bar codes only. • 1 - Inverse Only - the decoder decodes inverse 1D bar codes only. • 2 - Inverse Autodetect - decode both regular and inverse 1D bar codes. |
| 89 | US Postnet | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 90 | US Planet | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 95 | Transmit US Postal Check Digit | Select whether to transmit US Postal data, which includes both US Postnet and US Planet, with or without the check digit: <ul style="list-style-type: none"> • 1* - Transmit US Postal Check Digit • 0 - Do Not Transmit US Postal Check Digit |
| 91 | UK Postal | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 96 | Transmit UK Postal Check Digit | <ul style="list-style-type: none"> • 1* - Transmit UK Postal Check Digit • 0 - Do Not Transmit UK Postal Check Digit |
| 290 | Japan Postal | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 291 | Australia Post | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 718 | Australia Post Format | <ul style="list-style-type: none"> • 0* - Autodiscriminate (or Smart mode) - Attempt to decode the Customer Information Field using the N and C Encoding Tables • 1 - Raw Format - Output raw bar patterns as a series of numbers 0 through 3. • 2 - Alphanumeric Encoding - Decode the Customer Information Field using the C Encoding Table • 3 - Numeric Encoding - Decode the Customer Information Field using the N Encoding Table |
| 326 | Netherlands KIX Code | <ul style="list-style-type: none"> • 1* - Enable Netherlands KIX Code • 0 - Disable Netherlands KIX Code |
| 592 | USPS 4CB/One Code/Intelligent Mail | <ul style="list-style-type: none"> • 0* - Disable USPS 4CB/One Code/Intelligent Mail • 1 - Enable USPS 4CB/One Code/Intelligent Mail |
| 611 | UPU FICS Postal | <ul style="list-style-type: none"> • 0* - Disable UPU FICS Postal • 1 - Enable UPU FICS Postal |
| 338 | GS1 DataBar-14 | <ul style="list-style-type: none"> • 1* - Enable GS1 DataBar-14 • 0 - Disable GS1 DataBar-14 |
| 339 | GS1 DataBar Limited | <ul style="list-style-type: none"> • 0* - Disable GS1 DataBar Limited • 1 - Enable GS1 DataBar Limited |

| | | |
|-----|--|--|
| 728 | GS1 DataBar Limited Security Level | <p>There are four levels of decode security for GS1 DataBar Limited bar codes. There is an inverse relationship between security and scanner aggressiveness. Increasing the level of security may result in reduced aggressiveness in scanning, so only choose the level of security necessary.</p> <ul style="list-style-type: none"> • 1 - Level 1 – No clear margin required. This complies with the original GS1 standard, yet might result in erroneous decoding of the DataBar Limited bar codes when scanning some UPC symbols that start with the digits “9” and “7”. • 2 - Level 2 – Automatic risk detection. This level of security may result in erroneous decoding of DataBar Limited bar codes when scanning some UPC symbols. If a misdecode is detected, the scanner operates in Level 3 or Level 1. • 3*- Level 3 – Security level reflects newly proposed GS1 standard that requires a 5X trailing clear margin. • 4 - Level 4 – Security level extends beyond the standard required by GS1. This level of security requires a 5X leading and trailing clear margin. |
| 340 | GS1 DataBar Expanded | <ul style="list-style-type: none"> • 0*- Disable GS1 DataBar Expanded • 1 - Enable GS1 DataBar Expanded |
| 397 | Convert GS1 DataBar to UPC/EAN | <p>This parameter only applies to GS1 DataBar-14 and GS1 DataBar Limited symbols not decoded as part of a Composite symbol. Enable this to strip the leading 010 from GS1 DataBar-14 and GS1 DataBar Limited symbols encoding a single zero as the first digit, and report the bar code as EAN-13. For bar codes beginning with two or more zeros but not six zeros, this parameter strips the leading 0100 and reports the bar code as UPC-A. The UPC-A Preamble parameter that transmits the system character and country code applies to converted bar codes. Note that neither the system character nor the check digit can be stripped.</p> <ul style="list-style-type: none"> • 0*- Disable Convert GS1 DataBar to UPC/EAN • 1 - Enable Convert GS1 DataBar to UPC/EAN |
| 341 | Composite CC-C | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable <p>NOTE: Before enabling a composite code, first enable Multi Decode Mode</p> |
| 342 | Composite CC-A/B | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 371 | Composite TLC-39 | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 344 | UPC Composite Mode | <p>Select an option for linking UPC symbols with a 2D symbol during transmission as if they were one symbol:</p> <ul style="list-style-type: none"> • 0*- UPC Never Linked - transmit UPC bar codes regardless of whether a 2D symbol is detected. • 1 - UPC Always Linked - transmit UPC bar codes and the 2D portion. <p>If 2D is not present, the UPC bar code does not transmit.</p> <ul style="list-style-type: none"> • 2 - Autodiscriminate UPC Composites - the imager determines if there is a 2D portion, then transmits the UPC, as well as the 2D portion if present. |
| 427 | GS1-128 Emulation Mode for UCC/EAN Composite Codes | <ul style="list-style-type: none"> • 0*- Disable GS1-128 Emulation Mode for UCC/EAN Composite Codes • 1 - Enable GS1-128 Emulation Mode for UCC/EAN Composite Codes |
| 15 | PDF417 | <ul style="list-style-type: none"> • 1*- Enable • 0 - Disable |
| 227 | MicroPDF417 | <ul style="list-style-type: none"> • 0*- Disable • 1 - Enable |
| 123 | Code 128 Emulation | <p>Enable this parameter to transmit data from certain MicroPDF417 symbols as if it was encoded in Code 128 symbols. Transmit AIM Symbology Identifiers must be enabled for this parameter to work.</p> <ul style="list-style-type: none"> • *0 - Disable Code 128 Emulation - transmits these MicroPDF417 symbols with one of the following prefixes:]L3 if the first codeword is 903-905]L4 if the first codeword is 908 or 909]L5 if the first codeword is 910 or 911 • 1 - Enable Code 128 Emulation - transmits these MicroPDF417 symbols with one of the following prefixes:]C1 if the first codeword is 903-905]C2 if the first codeword is 908 or 909]C0 if the first codeword is 910 or 911 <p>NOTE: Linked MicroPDF codewords 906, 907, 912, 914, and 915 are not supported. Use GS1 Composites instead.</p> |
| 292 | Data Matrix | <ul style="list-style-type: none"> • 1*- Enable • 0 - Disable |
| 588 | Data Matrix Inverse | <ul style="list-style-type: none"> • 0*- Regular Only - the decoder decodes regular Data Matrix bar codes only. • 1 - Inverse Only - the decoder decodes inverse Data Matrix bar codes only. • 2 - Inverse Autodetect - decode both regular and inverse Data Matrix codes |
| 537 | Decode Mirror Images (Data Matrix Only) | <ul style="list-style-type: none"> • 0*- Never - do not decode Data Matrix bar codes that are mirror images • 1 - Always - decode only Data Matrix bar codes that are mirror images • 2 - Auto - decode both mirrored and unmirrored Data Matrix bar codes |
| 294 | Maxicode | <ul style="list-style-type: none"> • 1*- Enable • 0 - Disable |
| 293 | QR Code | <ul style="list-style-type: none"> • 1*- Enable • 0 - Disable |
| 587 | QR Inverse | <ul style="list-style-type: none"> • 0*- Regular Only - the decoder decodes regular QR bar codes only. |

| | | |
|------|-------------------------|---|
| | | <ul style="list-style-type: none"> • 1 - Inverse Only - the decoder decodes inverse QR bar codes only. • 2 - Inverse Autodetect - decode both regular and inverse QR bar codes. |
| 573 | MicroQR | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 574 | Aztec | <ul style="list-style-type: none"> • 1* - Enable • 0 - Disable |
| 589 | Aztec Inverse | <ul style="list-style-type: none"> • 0* - Regular Only - the decoder decodes regular Aztec bar codes only. • 1 - Inverse Only - the decoder decodes inverse Aztec bar codes only. • 2 - Inverse Autodetect - the decoder decodes both regular and inverse Aztec bar codes. |
| 1167 | Han Xin | <ul style="list-style-type: none"> • 0* - Disable • 1 - Enable |
| 1168 | Han Xin Inverse | <ul style="list-style-type: none"> • *0 - Regular Only - decode Han Xin bar codes with normal reflectance only • 1 - Inverse Only - decode Han Xin bar codes with inverse reflectance only • 2 - Inverse Autodetect - decode both regular and inverse Han Xin bar codes |
| 78 | Redundancy Level | <p>The decoder offers four levels of decode redundancy. Select higher redundancy levels for decreasing levels of bar code quality. As redundancy levels increase, the decoder's aggressiveness decreases</p> <ul style="list-style-type: none"> • 1* - Redundancy Level 1 <p>The following code types must be successfully read TWICE before being decoded (Code Type / Code Length):</p> <ul style="list-style-type: none"> ○ Codabar 8 characters or less ○ MSI 4 characters or less ○ D 2 of 5 8 characters or less ○ I 2 of 5 8 characters or less <p>2 - Redundancy Level 2</p> <p>The following code types must be successfully read TWICE before being decoded (Code Type / Code Length):</p> <ul style="list-style-type: none"> ○ All All <ul style="list-style-type: none"> • 3 - Redundancy Level 3 <p>Code types OTHER THAN the following must be successfully read TWICE before being decoded. The following codes must be read THREE times:</p> <ul style="list-style-type: none"> ○ MSI Plessey 4 characters or less ○ D 2 of 5 8 characters or less ○ I 2 of 5 8 characters or less ○ Codabar 8 characters or less <ul style="list-style-type: none"> • 4 - Redundancy Level 4 <p>The following code types must be successfully read THREE times before being decoded (Code Type / Code Length):</p> <ul style="list-style-type: none"> ○ All All |
| 77 | Security Level | <p>The decoder offers four levels of decode security for delta bar codes, which include the Code 128 family, UPC/EAN, and Code 93. Select increasing levels of security for decreasing levels of bar code quality. There is an inverse relationship between security and decoder aggressiveness, so choose only that level of security necessary for any given application.</p> <ul style="list-style-type: none"> • 0 - Security Level 0: This setting allows the decoder to operate in its most aggressive state, while providing sufficient security in decoding most "in-spec" bar codes. • 1* - Security Level 1: Select this option if misdecodes occur. This default setting eliminates most misdecodes. • 2 - Security Level 2: Select if Security level 1 fails to eliminate misdecodes. • 3 - Security Level 3: If misdecodes still occur with Security Level 2, select this security level. Be advised, selecting this option is an extreme measure against mis-decoding severely out of spec bar codes. Selecting this level of security significantly impairs the decoding ability of the decoder. If this level of security is necessary, try to improve the quality of the bar codes. |
| 381 | Intercharacter Gap Size | <p>The Code 39 and Codabar symbologies have an intercharacter gap that is typically quite small. Due to various bar code-printing technologies, this gap can grow larger than the maximum size allowed, preventing the decoder from decoding the symbol. If this problem occurs, select Large Intercharacter Gaps to tolerate these out-of-specification bar codes.</p> <ul style="list-style-type: none"> • 6* - Normal Intercharacter Gaps • A - Large Intercharacter Gaps |

4 Key Codes

The physical keys on the handheld each generate a unique Key Code, and when you handle key events in your application you look at the reported key code to determine which button was pressed.

The Alien API library provides the `KeyCode` class that defines codes for special physical keys on the handheld.

4.1 ALR-H450 (Android 4.4.2)

The `KeyCode.ALH450` class defines the following key codes for the ALR-H450 handheld:
`SCAN`, `SIDE_LEFT`, `SIDE_RIGHT`.

Descriptions:

| | |
|-------------------------|---|
| <code>SCAN</code> | Key code of the Scan button. |
| <code>SIDE_LEFT</code> | Key code of the button on the Left Side of the unit. |
| <code>SIDE_RIGHT</code> | Key code of the button on the Right Side of the unit, below the Power button. |

4.2 ALR-H460 (Android 6.0)

The `KeyCode.ALH460` class defines the following key codes for the ALR-H460 handheld:
`SCAN`, `MENU`, `ENTER`, `BACK`, `FUNC`, `HANDLE_TRIGGER`.

Descriptions:

| | |
|-----------------------------|---|
| <code>SCAN</code> | Key code of the SCAN button (both Left and Right sides) |
| <code>MENU</code> | Key code of the MENU button (1st button on the Front side) |
| <code>ENTER</code> | Key code of the ENTER button (3rd button on the Front side) |
| <code>BACK</code> | Key code of the BACK button (4th button on the Front side) |
| <code>FUNC</code> | Key code of the FUNC button (gray button on the Left side) |
| <code>HANDLE_TRIGGER</code> | Key code of the HANDLE TRIGGER |

5 Developing applications with Android Studio

5.1 Install Android Studio

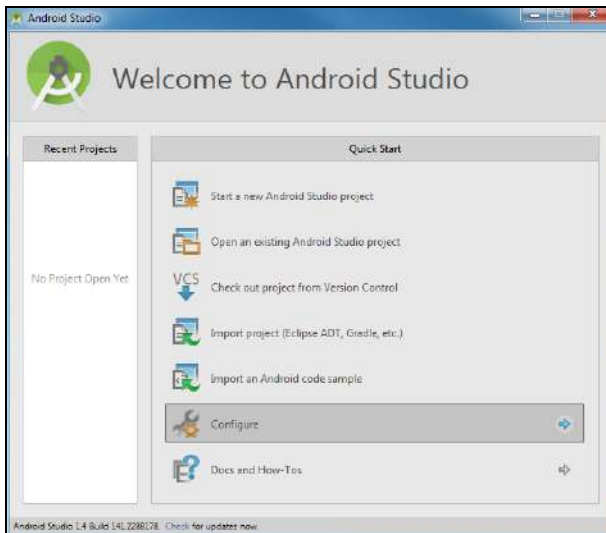
In order to develop RFID applications running on the Alien ALR-H450/H460 handheld, you need to install Android Studio 1.4 (or newer) on your computer.

Download Android Studio from <http://developer.android.com/sdk/index.html>

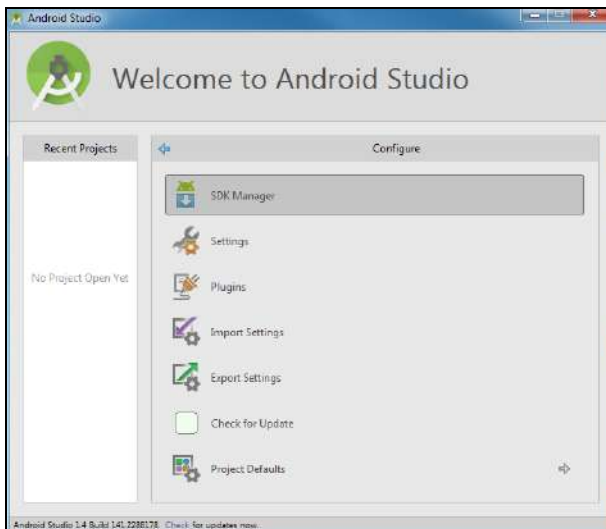
Run the installer and use all the default settings. The installer will automatically download and install the required components, including Android Support Repository and Android SDK Tools.

5.2 Install Google USB Driver

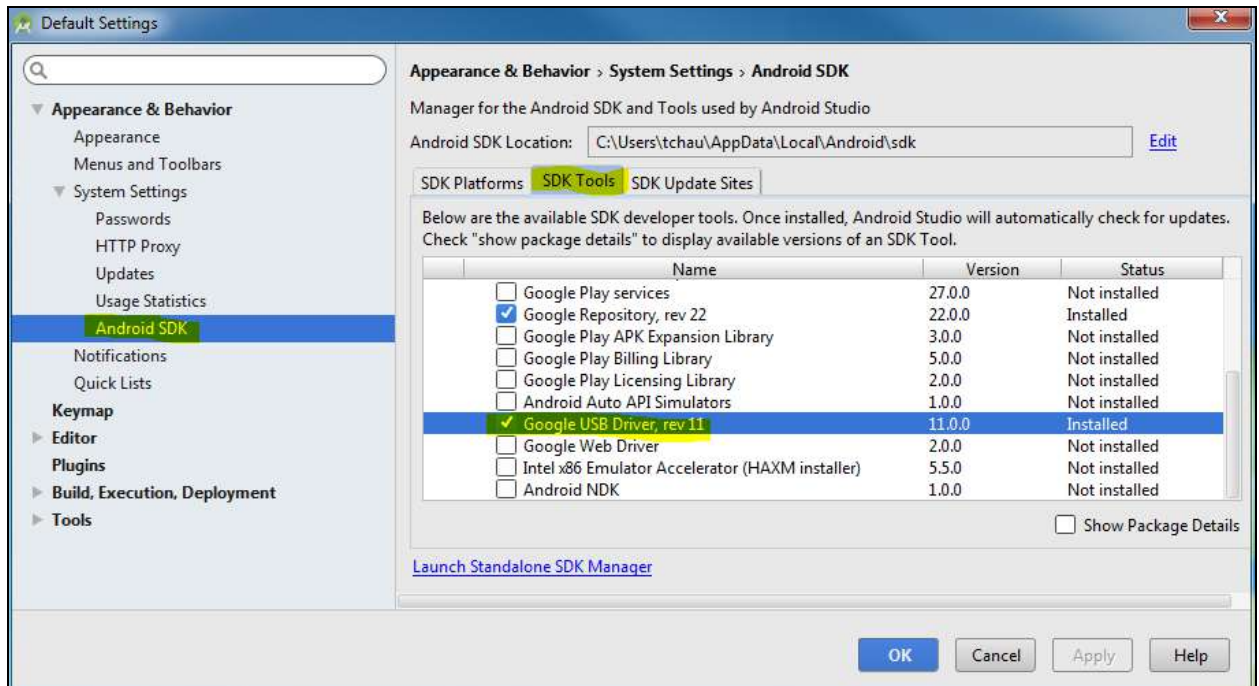
Open Android Studio. On the “Welcome to Android Studio” screen select Configure:



Select “SDK Manager”:



Select “SDK Manager”, then “Android SDK”, then “SDK Tools” tab, and check the “Google USB Driver”. Click OK:



Make sure your PC is connected to the Internet.

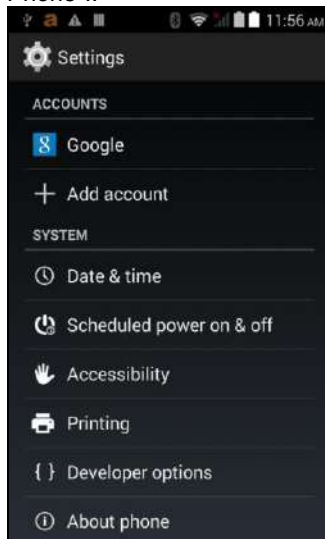
Configure Windows to automatically download device drivers:

- Go to Control Panel > Devices and Printers
- Right click on the Computer icon and select “Device installation Settings”
- Select “Yes” to automatically download device drivers

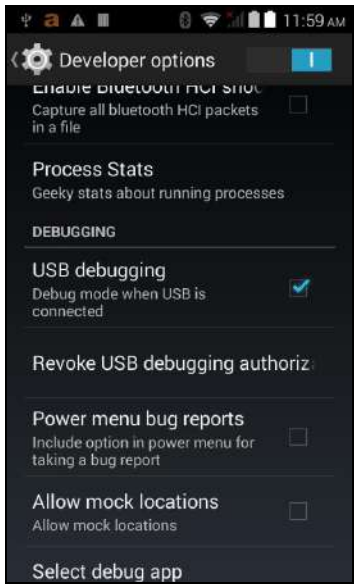
5.3 Enable Developer Mode for your handheld

Go to the “Settings” app on the handheld, tap on “About Phone”. Then tap the “Build Number” 7 times. This will enable Developer Mode for your handheld.

Go back to the “Settings” and you will see a new “Developer options” menu item right above the “About Phone”..



Tap on “Developer options”, check “USB Debugging” and click OK to enable USB debugging:



Verify your device is properly detected by your PC:

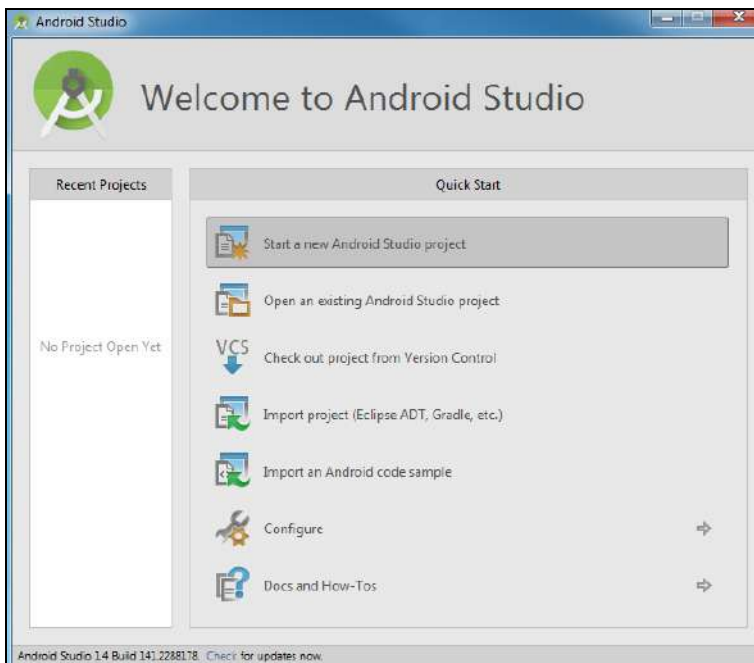
- Connect the cradle to the PC with a USB cable and insert the handheld into the cradle.
- Open the Device Manager to verify that the device has been identified under “Android USB Devices”

Now you are ready to develop applications for the handheld.

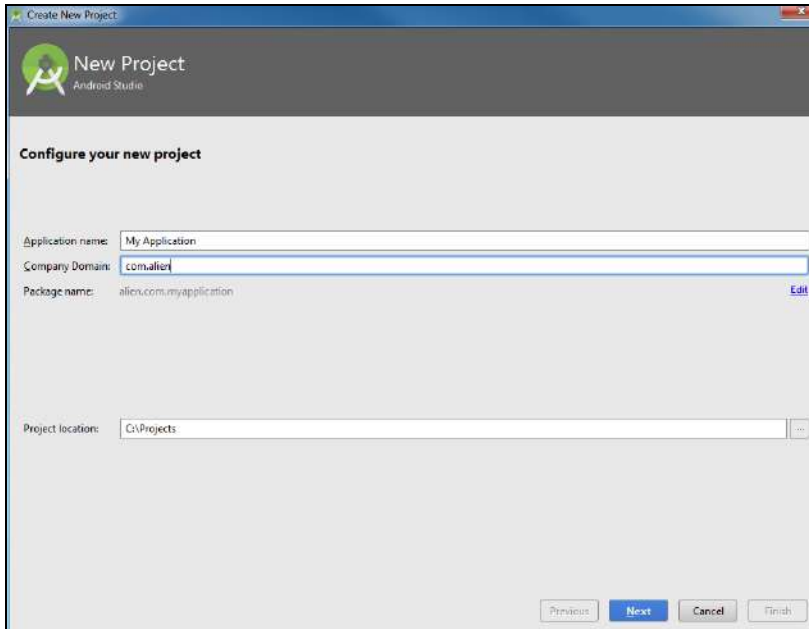
5.4 Developing your first RFID application

5.4.1 Create Android Project

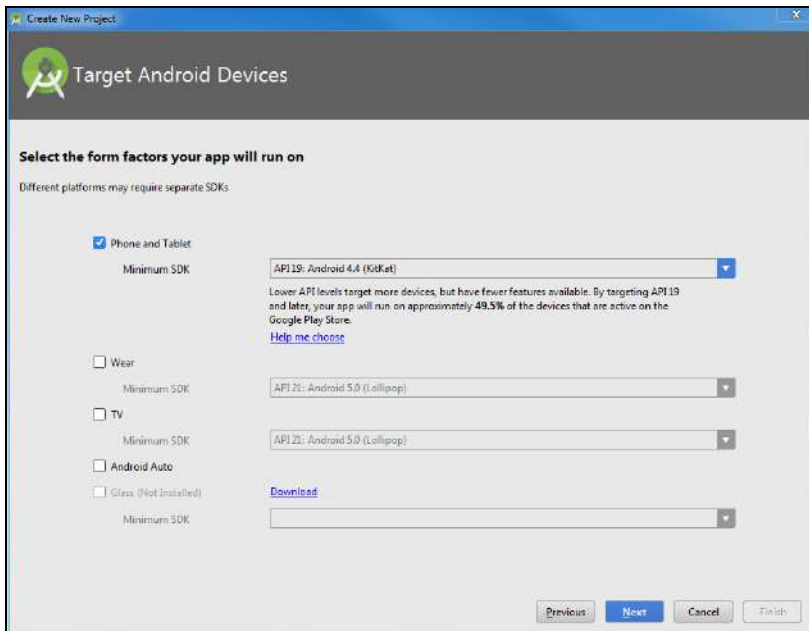
To create a new project, open Android Studio and click “Start new Android Studio project”:



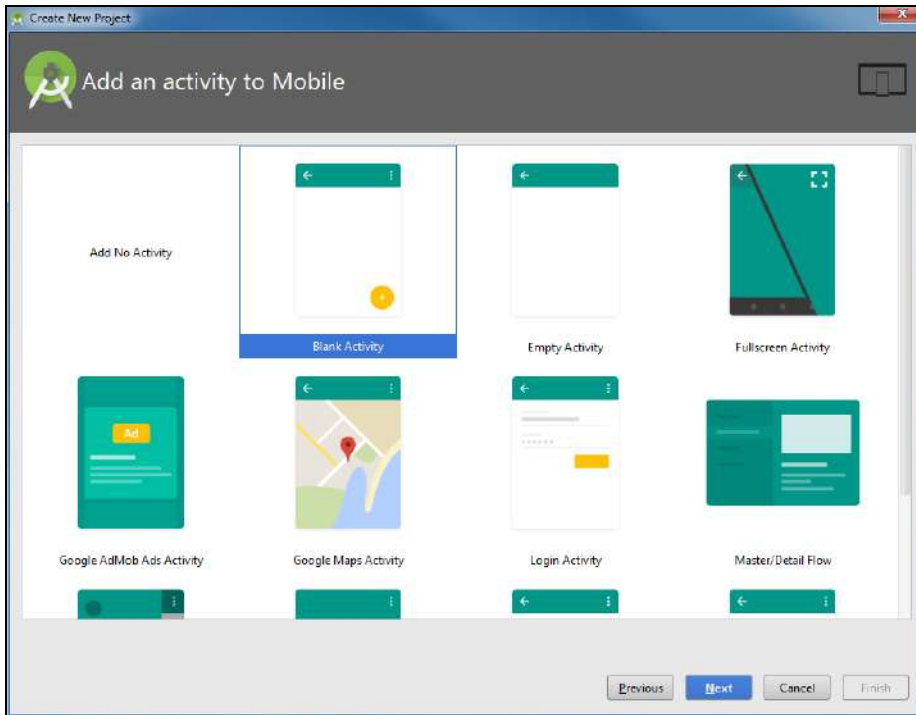
In the “New Project” dialog, use the default settings or enter new names for Application Name, Domain and Project location for your project. Click “Next”:



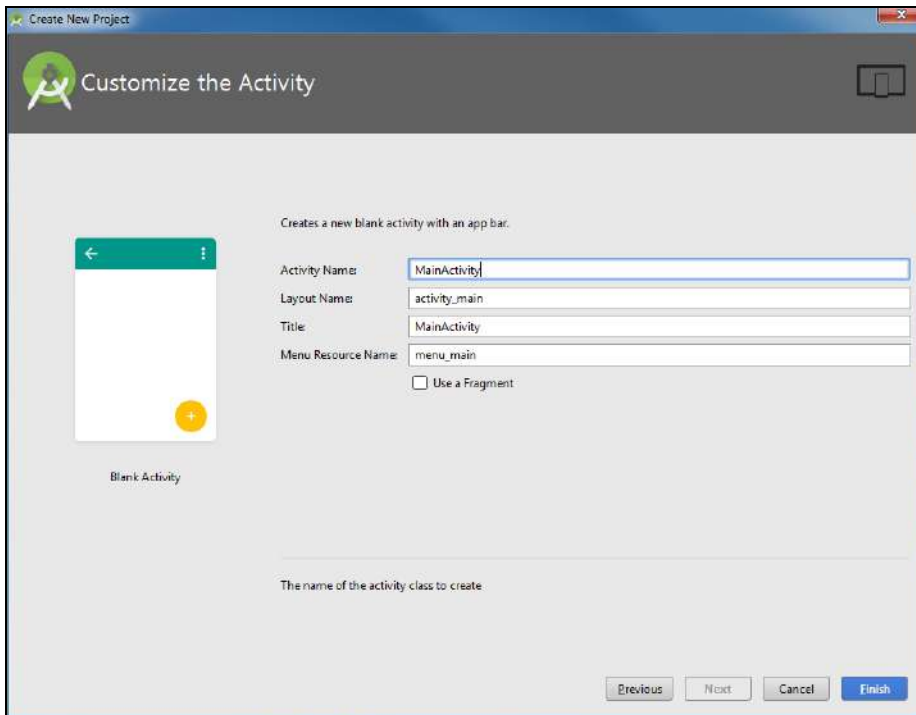
In the “Target Android Devices” dialog, check the “Phone and Tablet” checkbox and select “API19: Android 4.4 (KitKat)” as the Minimum SDK. Click “Next”:



In the “Add an activity to Mobile” dialog, select “Blank Activity” and click “Next”:

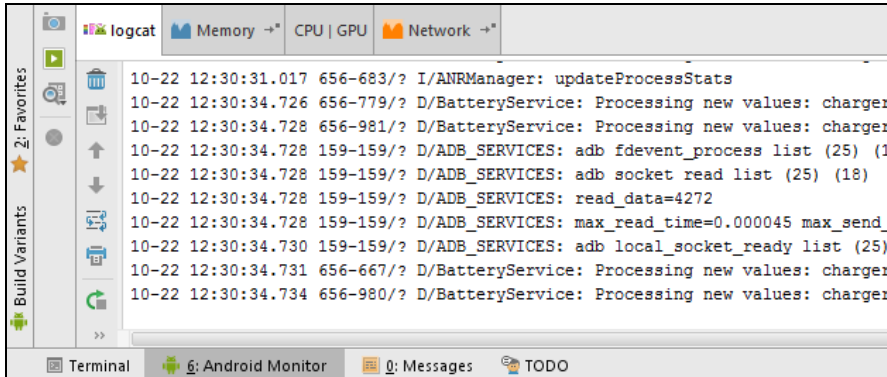


In the “Customize the Activity” dialog, use the default names or enter new names for Activity Name, Layout Name, Title and Menu Resource Name. Click “Finish”:



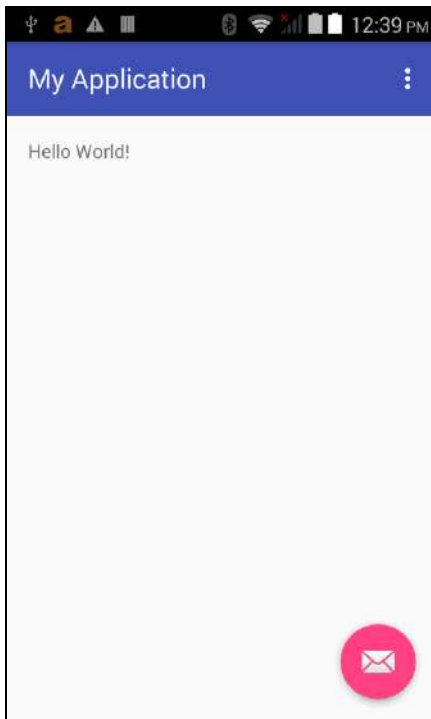
Connect the cradle to your PC, and insert the handheld into the cradle. You should see “Connected as USB storage” and “USB Debugging Enabled” notifications on the handheld screen.

In Android Studio, click “Android Monitor” at the bottom left. Then the “Allow USB debugging” will show up on your handheld’s screen. Check “Always allow from this computer” and tap OK. This will allow Android Studio to deploy the application into your handheld as well as to get log info from the handheld. Logs from the handheld will display in the “logcat” tab:



```
logcat
Memory →* CPU | GPU Network →*
10-22 12:30:31.017 656-683/? I/ANRManager: updateProcessStats
10-22 12:30:34.726 656-779/? D/BatteryService: Processing new values: chargerP
10-22 12:30:34.728 656-981/? D/BatteryService: Processing new values: chargerP
10-22 12:30:34.728 159-159/? D/ADB_SERVICES: adb fdevent_process list (25) (12
10-22 12:30:34.728 159-159/? D/ADB_SERVICES: adb socket read list (25) (18)
10-22 12:30:34.728 159-159/? D/ADB_SERVICES: read_data=4272
10-22 12:30:34.728 159-159/? D/ADB_SERVICES: max_read_time=0.000045 max_send_t
10-22 12:30:34.730 159-159/? D/ADB_SERVICES: adb local_socket_ready list (25)
10-22 12:30:34.731 656-667/? D/BatteryService: Processing new values: chargerP
10-22 12:30:34.734 656-980/? D/BatteryService: Processing new values: chargerP
```

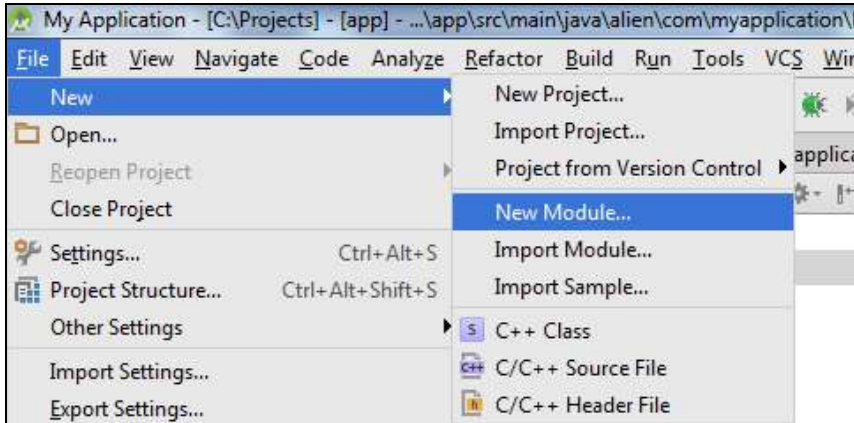
In Android Studio, click “Run app” toolbar button to run your application. The “Device Chooser” dialog will pop up. Select your device and click OK to run your “Hello World” application on the handheld:



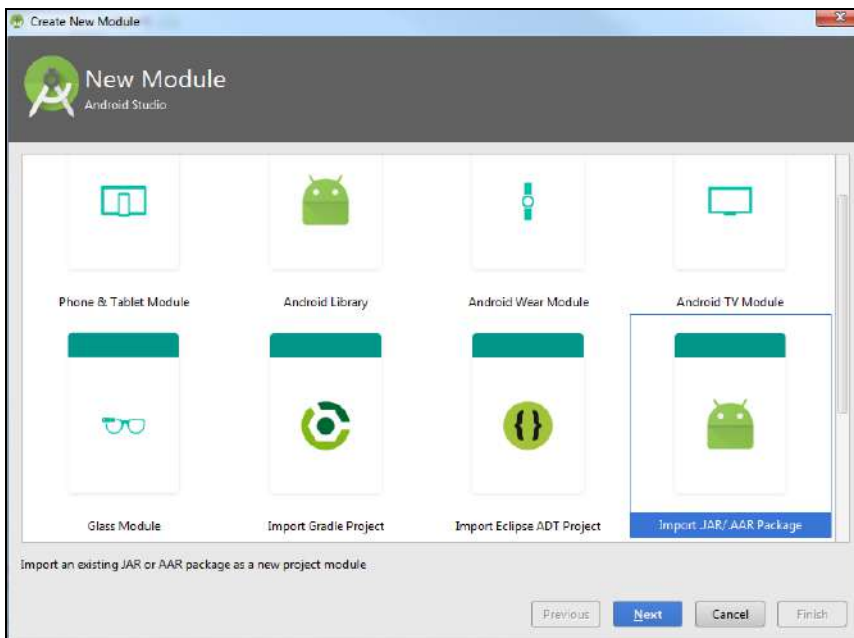
5.4.2 Use RFID functionality in your project

In order to use RFID functionality, you have to add the Alien `alienapi.aar` library to your project.

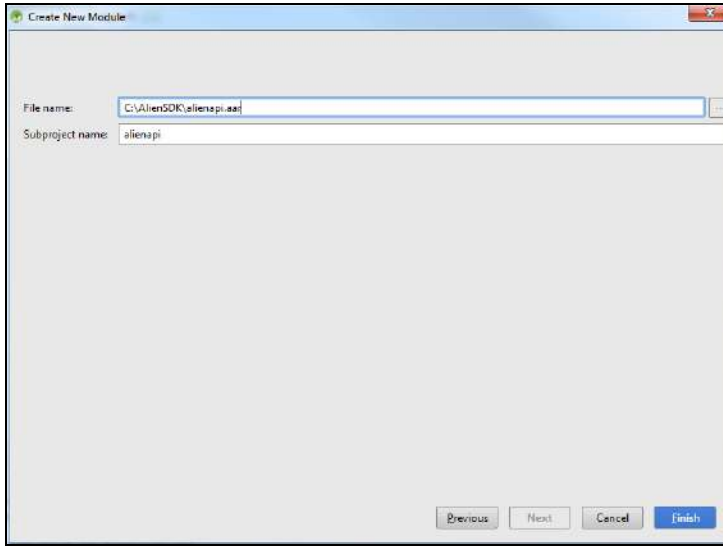
Select menu: File > New > New Module:



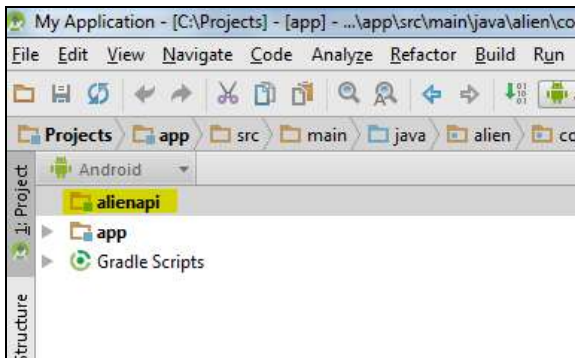
In the “New Module” dialog, select “Import .JAR/.AAR Package” and click “Next”:



Browse to select the `alienapi.aar` library file and click “Finish”:

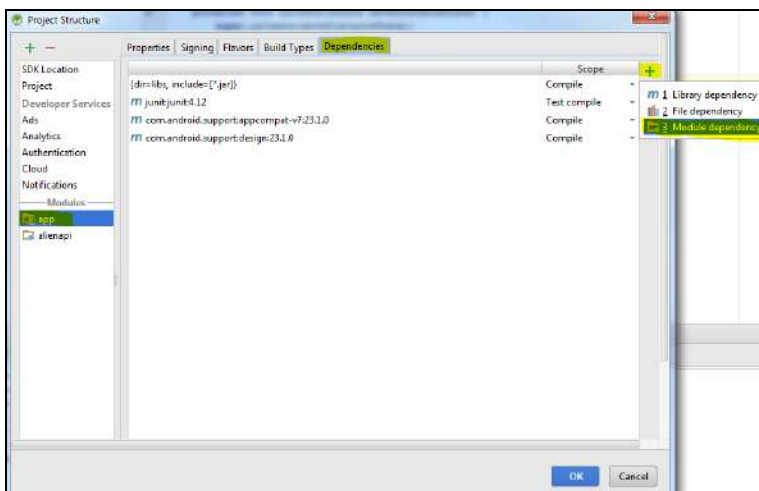


Now, you should see the “**alienapi**” module added to your project:

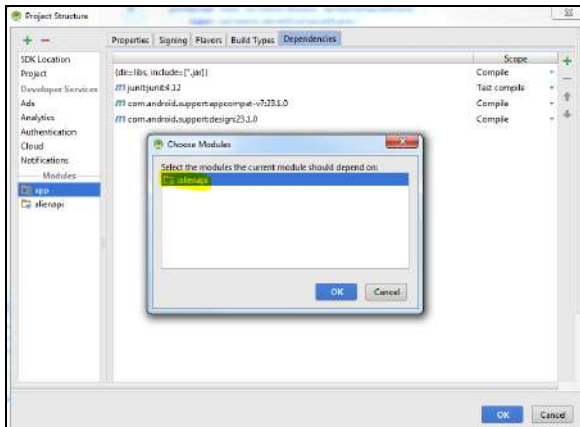


Next, you need to configure the “app” module to use “alienapi” module. To do this:

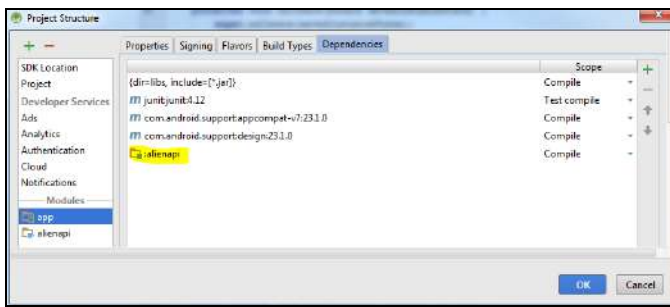
- Right click on the “app” module, and select “Open Module Settings”.
- Select the “app” module, select the “Dependencies” tab, click “+” button and select “Module dependencies”.



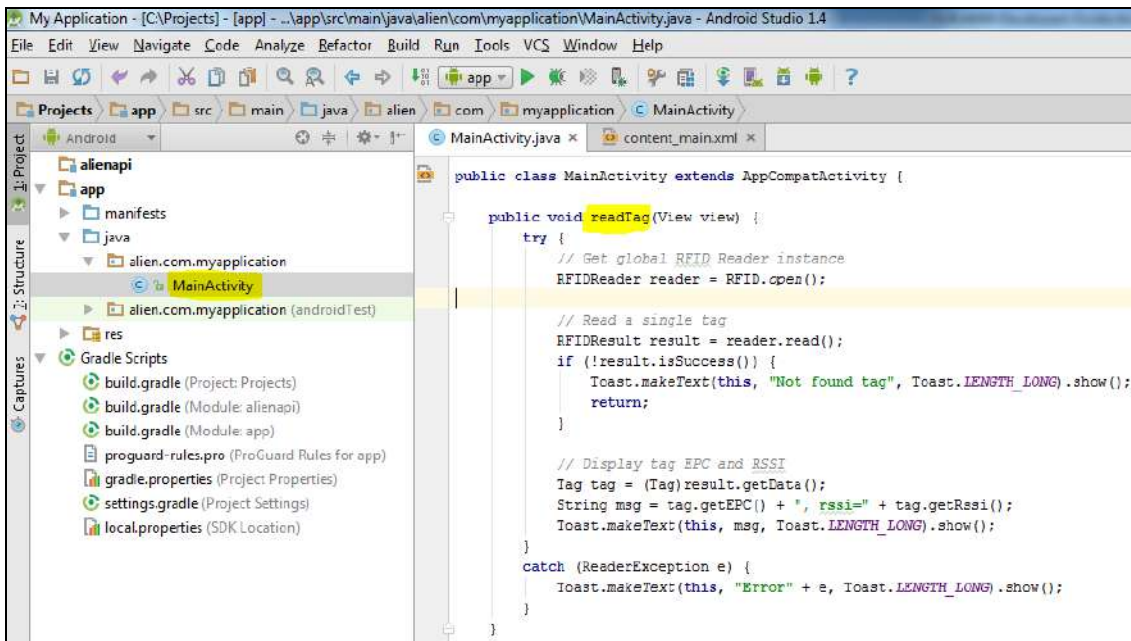
Select “:alienapi” and click OK:



Now you should see the “:alienapi” in the Dependencies list. Click “OK”:



Add the readTag () method to your MainActivity class as below:



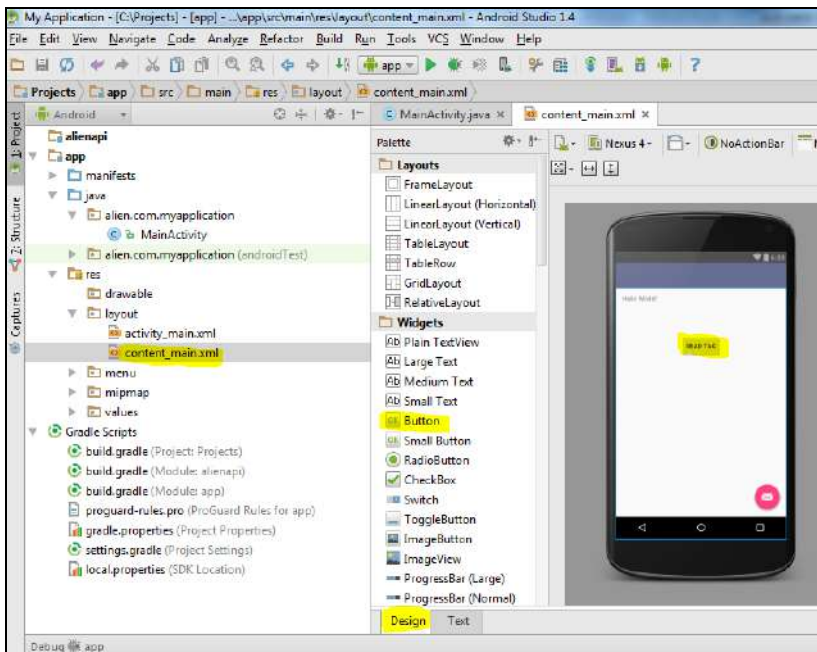

```

public void readTag(View view) {
    try {
        // Get global RFID Reader instance
        RFIDReader reader = RFID.open();
        // Read a single tag
        RFIDResult result = reader.read();
        if (!result.isSuccess()) {
            Toast.makeText(this, "No tags found ", Toast.LENGTH_LONG).show();
            return;
        }
        // Display tag EPC and RSSI
        Tag tag = (Tag)result.getData();
        String msg = tag.getEPC() + ", rssi=" + tag.getRSSI();
        Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
    }
    catch (ReaderException e) {
        Toast.makeText(this, "Error: " + e, Toast.LENGTH_LONG).show();
    }
}
}

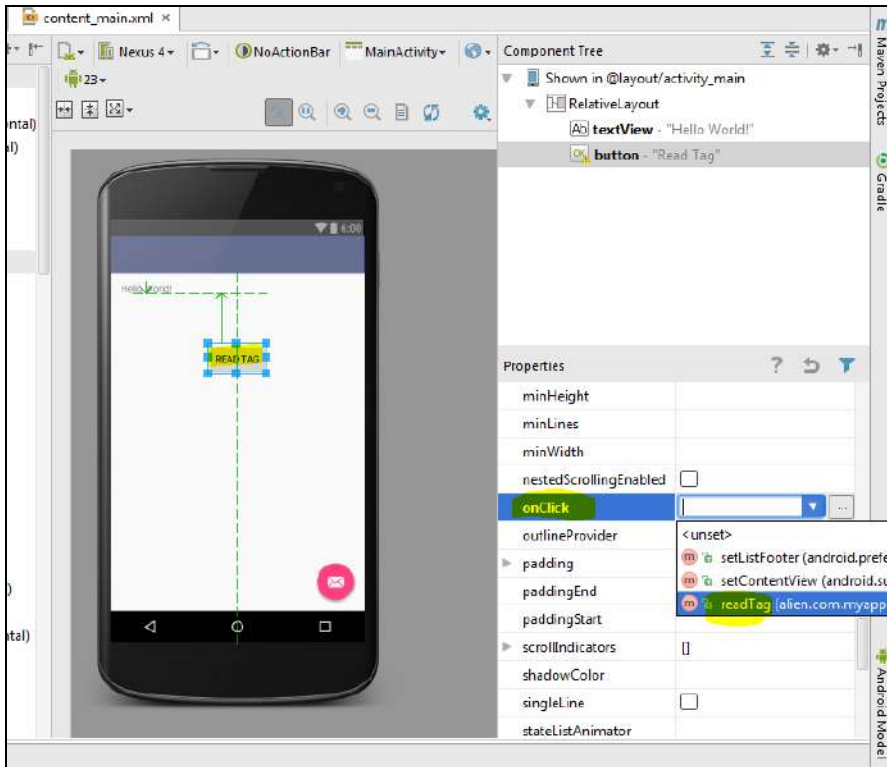
```

Next add a Button to call the `readTag` method that you just created:

- Double click on “content_main.xml”.
- Select “Design” tab.
- Drag and Drop a “Button” widget into the view.
- Double click the button and change the name to “READ TAG”.



- Select the button, then select “onClick” in the “Properties” panel, and choose “readTag”:



Now test the application:

- Click “Run App” toolbar button to start the application on your handheld.
- Place a RFID tag in front of your handheld.
- Tap the “READ TAG” button in the application window to read and display a tag as shown below

