Algiz RT7 Developer Guide

Handheld Group

Revision 1.3

Table of Contents

List of Figures

Chapter 1. Introduction

The Algiz RT7 provides a standard, fully compliant Android implementation. 64 bit and 32 bit applications are fully supported.

Additionally, the Algiz RT7 includes a couple of features that are not available on standard Android devices:

- Hotkeys
- Barcode Scanner (optional accessory)

This guide shows how developers can use these features in their applications.

Chapter 2. Hotkeys

The Algiz RT7 has four user defined hotkeys. Using Settings, a user can map these keys to specific functions: home key, back key, start application, and scan barcode (if the unit is equipped with a barcode scanner). Each key is mapped independently. If any key is undefined, then it is passed to applications.

The four keys A, B, C, and D are mapped to standard Android keys F1, F2, F3 and F4, respectively. A user application can process these keys as shown in keytest.java.

keytest.java

```
public class TestActivity extends Activity {
    ...
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        switch(keyCode) {
            case KeyEvent.KEYCODE_F1:
            case KeyEvent.KEYCODE_F2:
            case KeyEvent.KEYCODE_F3:
            case KeyEvent.KEYCODE_F4:
                // take some action
                return true;
        }
        return false;
    }
    ...
};
```

Chapter 3. Barcode Scanner

The barcode scanner is an optional accessory. Based on the Motorola SE4710 engine, it provides robust support for a wide variety of 1D/2D barcodes, as well as postal codes. The complete specifications of the module are documented in the user guide.

If the device is equipped with a barcode scanner, then the user can map one of the four hotkeys to scan barcodes. Scanned barcodes are directly sent to the application. A generic set of configuration options is available from the Settings application.

Sometimes, this built in functionality is not enough to meet the application requirements. In some cases, you may want better control on the symbologies, and deeper customization. Or you may want to use the scanner in a "hands-free" mode. Or you may want to use the module as an imager. Elaborate control over the module can be achieved by developing custom applications.

The "barcode-sdk" directory has all the relevant files, in the following subdirectories:

- Applications : contains a sample application Section 3.2, “SDLGui application” that demonstrates the usage of the barcode SDK.
- Sources : contains com.zebra.adc.decoder.BarCodeReader class in source form. This needs to be included in custom applications.
- Documentation : This contains source documentation for BarcodeReader.java (see BarCodeReader_JavaDoc/index.html), and contains a PDF userguide (SWDecodeAndroid_IG.pdf) for the scanner API. Please ignore the following sections in the userguide - these do not apply to this device:
  - "Related Documents" and "Service Information" in "About this Guide"
  - "Software Installation" and "Setting up the Build Environment" in "Getting Started"
  - Chapter 3, "Software Decode Library API" can be completely ignored.
- Libraries : Custom applications need to include libraries in this folder. Both 32 bit and 64 bit libraries are provided.

The barcode module is exposed to Android a camera, with a cameraId of 3. Android will, however, report that the number of cameras available is 2; this prevents other applications from accidentally accessing the barcode as a front camera.

From a resource perspective, applications must treat the barcode scanner as a camera. Well behaved applications must carefully release the barcode object when the application stops using it, and as soon as the application is paused. If your application does not properly release the camera, then all subsequent attempts to access the camera, including those from your own application, will fail and may cause you or other applications to be shut down.

Developers can get started by looking at the PDF userguide and the sample applications. Android Studio or Eclipse may be used for application development.

3.1. Barcode SDK changes from version 1.2

A few things have changed in the SDK since the previous release. They are as given below:

1. Updated latest library's (both 32 and 64 bit) to avoid app loading issues.

Just import the source in Android studio and click on Run.

It will generate apk inside app/build/generated/outputs/apk.

3.2. SDLGui application

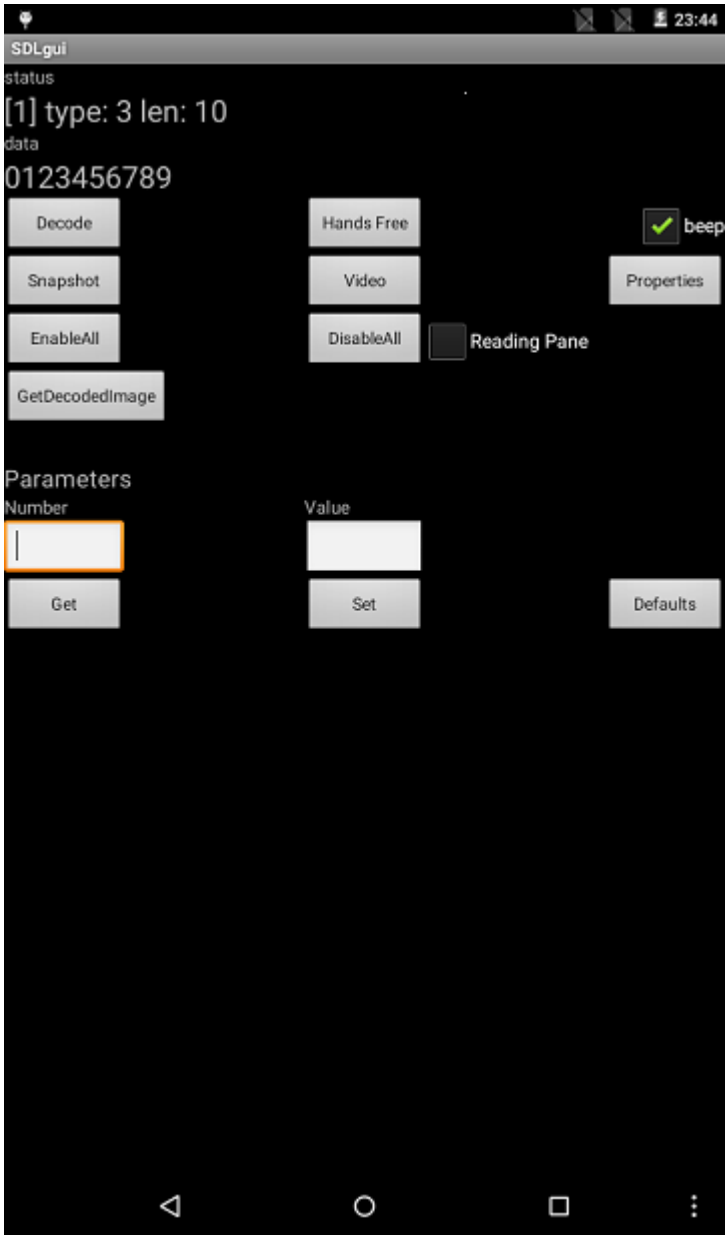

Figure 3.1. SDLGui app screenshot

SDLGui is an application used for barcode decoding. It has several buttons available whose functionality is explained below:

- Decode : Decodes one barcode at a time.
- Snapshot : Takes a photo from barcode camera and displays it. Touch the snapshot image to return to the main menu.
- EnableAll : Enables all the supported barcode types.

- GetDecodedImage : Displays the last image which was used for successful decode. Touch the decoded image to return to the main menu. For getting the decoded image, first you need to set the barcode parameter 905 (Refers to com.zebra.adc.decoder.BarCodeReader.ParamNum.RETRIEVE_LAST_DECODE in the file BarCodeReader.java) to 1. If this parameter is not set then blank screen will appear when you click on "GetDecodedImage" button. Refer "Parameters" button below for the details about setting a parameter.
- Handsfree : Continously decodes barcodes one by one. It can be disabled by clicking the button again.
- Video : Starts video in the application UI. It gives only the preview, no recording is done. Touch the video to return to the main menu.
- DisableAll : Disables all the supported barcodes.
- beep : It can be used to enable or disable beep sound. If it is enabled, a beep sound is heard after a successful decode.
- Parameters : This can be used to set and get the values of a barcode parameter. "Number" is the parameter number and "value" is the parameter value. For more information on parameters, please refer to the Chapter 4 of "SDK Software Decode" document, that is available in the location barcode-sdk/Documentation/SWDecodeAndroid_IG.pdf. This gives a list of all the user and imaging parameters and their functionality.
- Defaults : This is used to revert all the barcode parameters to their default values.

3.2.1. Decoding a barcode

Hold the barcode to be scanned in front of the barcode camera. Click the "Decode" button. In case of successful decode, "status" shows the barcode type and "data" shows the barcode value. In case of unsuccessful decode, "decode timed out" is displayed as the status.