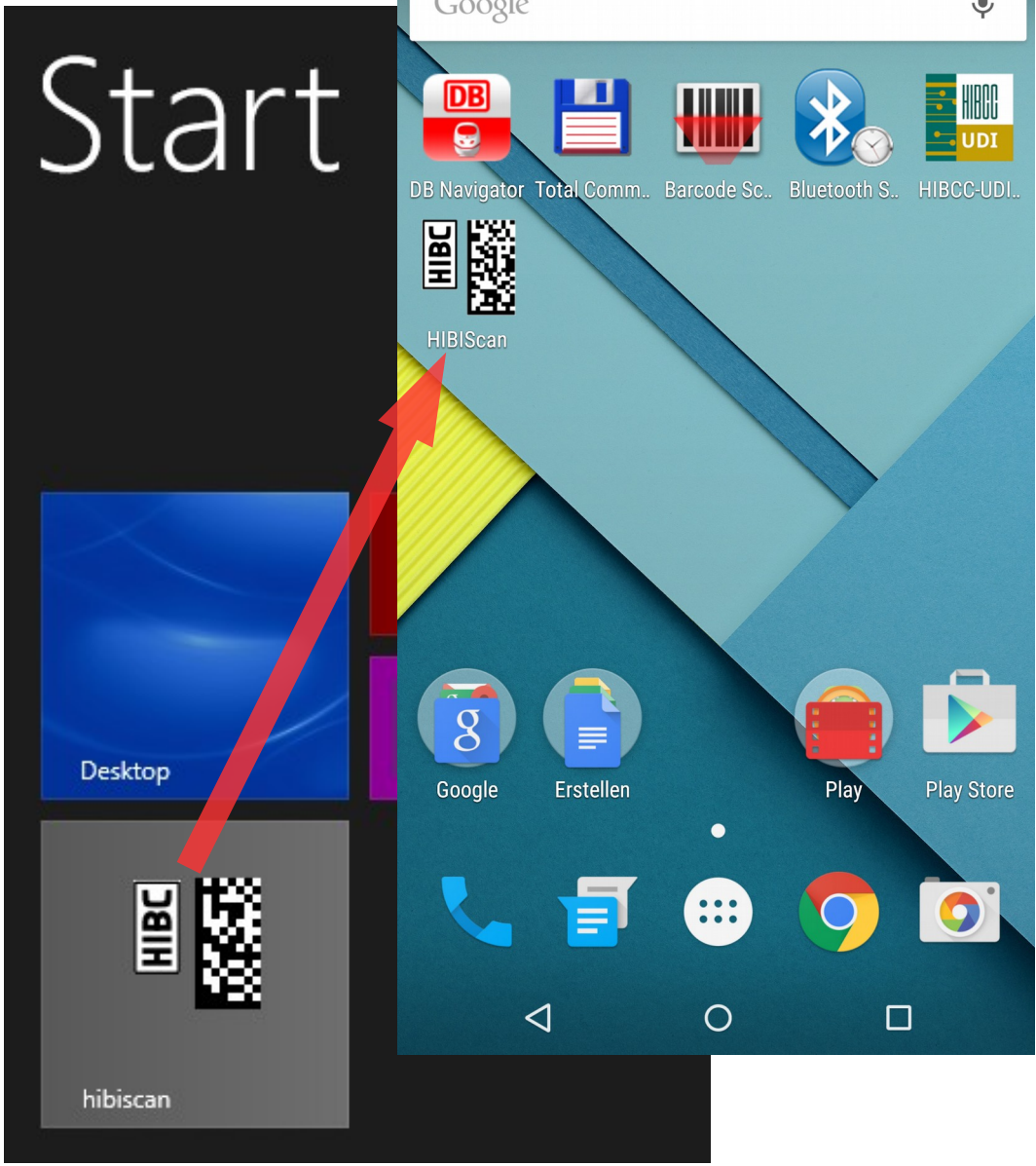
Porting a simple TCL Application to Android

- Elmicron
- HIBIScan
- AndroWish start
- User Interface
- APK
- TCL Activities
- Wish List



Harald.Oehlmann@elmicron.de
Naumburg/Germany
Chat: oehhar, Wiki: hao

Company Elmicron

Auto ID
→ Barcode and RFID
→ IoT Standards

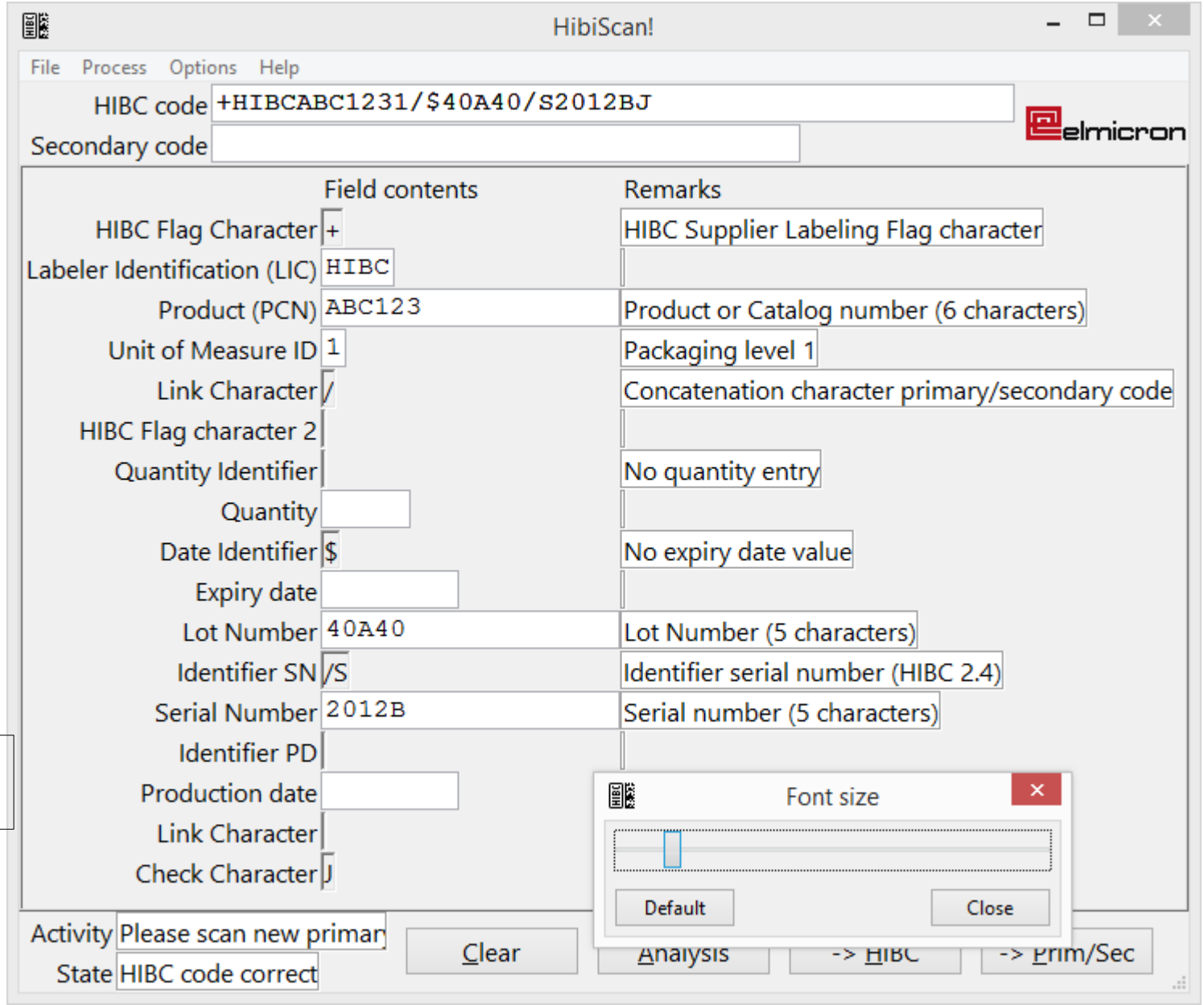
8 Employees

TclDevKit Licence

HIBIScan

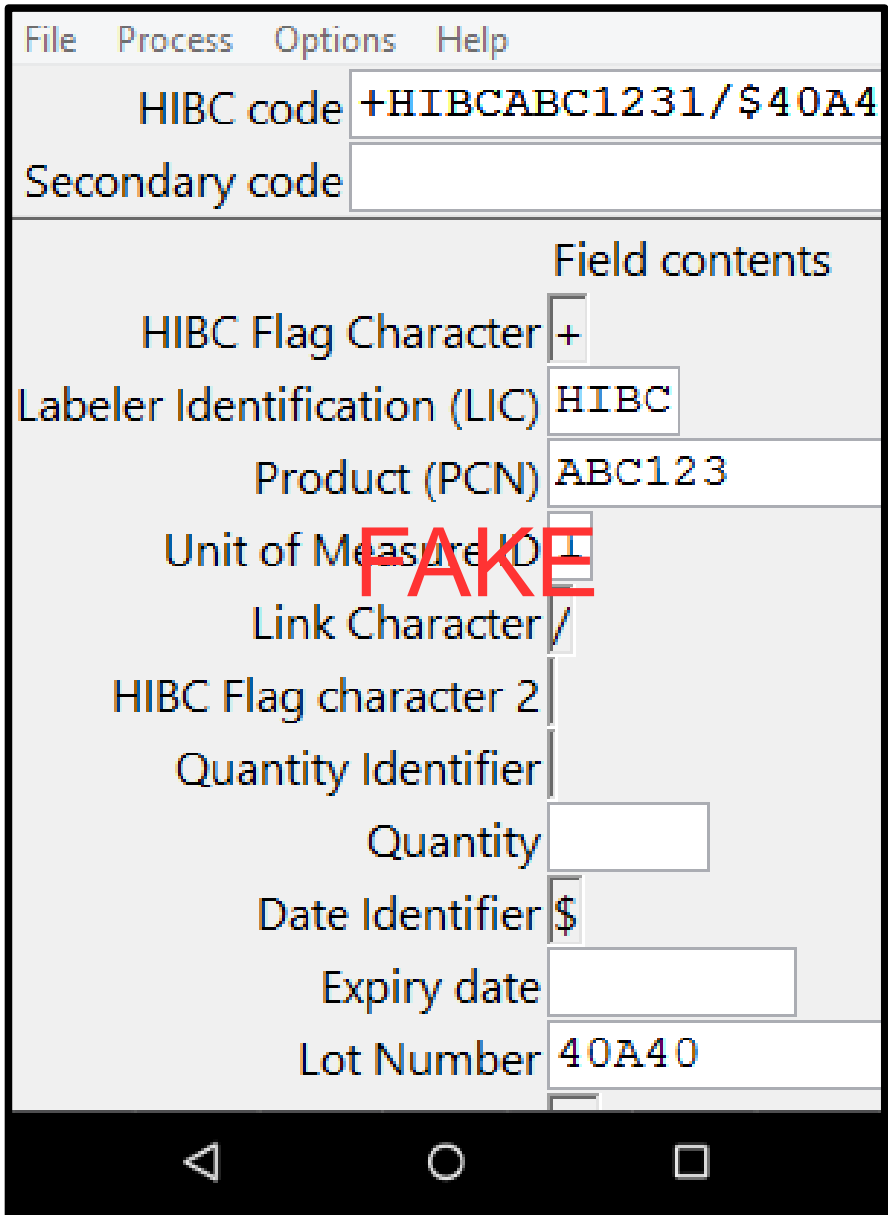
- TCL/Tk8.6.2, Ttk
- Win CE
- Fix window with no scrollbars. It clips if window to small
- Adjustable Font Size with slider control

```
font configure\
   MyFont -size 32
```



Start with AndroWish: just works!

- Google Nexus 5: Android 5, 1440x2560 Pixel
- Click on tcl-file starts program (Total Commander)
- Big virtual screen
- Move with one finger
- Virtual Zoom with two fingers pinching (not font size change)
- Keyboard shows up when entry has focus
- Horizontal to vertical change when device is turned



Console ?

- In Android script, activate tkcon client

```
package require tkconclient
tkconclient::start 12345
```

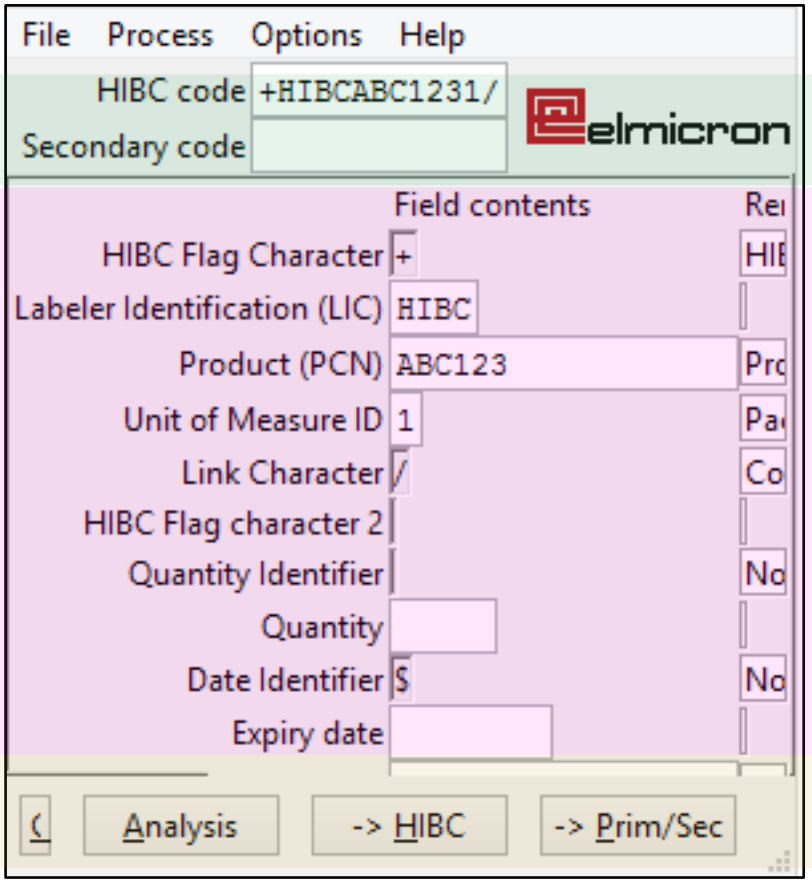
- On PC bridge Android tcp/ip port to Windows tcp/ip port

```
adb forward tcp:12345 tcp:12345
```

- On PC start TkCon and attach to socket "localhost 12345"

UI Design

- No viewport functionality: manage on my own
- No menu
- Always visible Buttons below. Height always two Text lines. Smaller width -> no text
- Two entry lines always visible
- Center area with scrollbar and moving with one finger
- Font size resize with two fingers pinch (also resizes buttons)
- Full screen info page



Finger scrolling

- Switch viewport off: sdltk touchtranslate 3
- "sframe.tcl" by Paul Walton from TCL wiki (has mousewheel scrolling)
- Extended by finger scroll
- X/Y units are screen width/height divided by 10000 -> Should exactly scroll with the finger
- Small scrollbars good for orientation

```
bind $t <<FingerDown>>  [list\
    +motion start $p %W %x %y %s]
bind $t <<FingerMotion>>   [list\
    +motion motion $p %W %x %y %s]

proc motion {mode p W X Y Finger} {
  if {$mode eq "motion"} {
    $p.canvas xview scroll [expr {
      ($mx - $X) *
      [winfo screenwidth .]/10000}]
      units
    $p.canvas yview scroll ...
  }
  set mx $X
  set my $Y
}
```

Pinch to Change Font Size

- Bind function
- State values:
  0:Motion, 1:Start, 2:End 1st Finger, 3:End Both Fingers
- Unit of motion is ???
- Font size values are small (3) even for huge screen resolution
- Change unit which felt good for font size (points) was 10/screen width

```
bind . <<PinchToZoom>> {+PiZo %x %s}

proc PiZo {X State} {
  switch -exact -- $State {
    1 { # Start
      set Value $X
      set FontSize $CurrSize
    }
    0 - 2 { # Motion, End
      if {$Value > 0} {
        NewFontSize [expr {
          $FontSize
          + ($X - $Value) * 10
          / [winfo screenwidth .] }]
      }
    }
  }
}
```

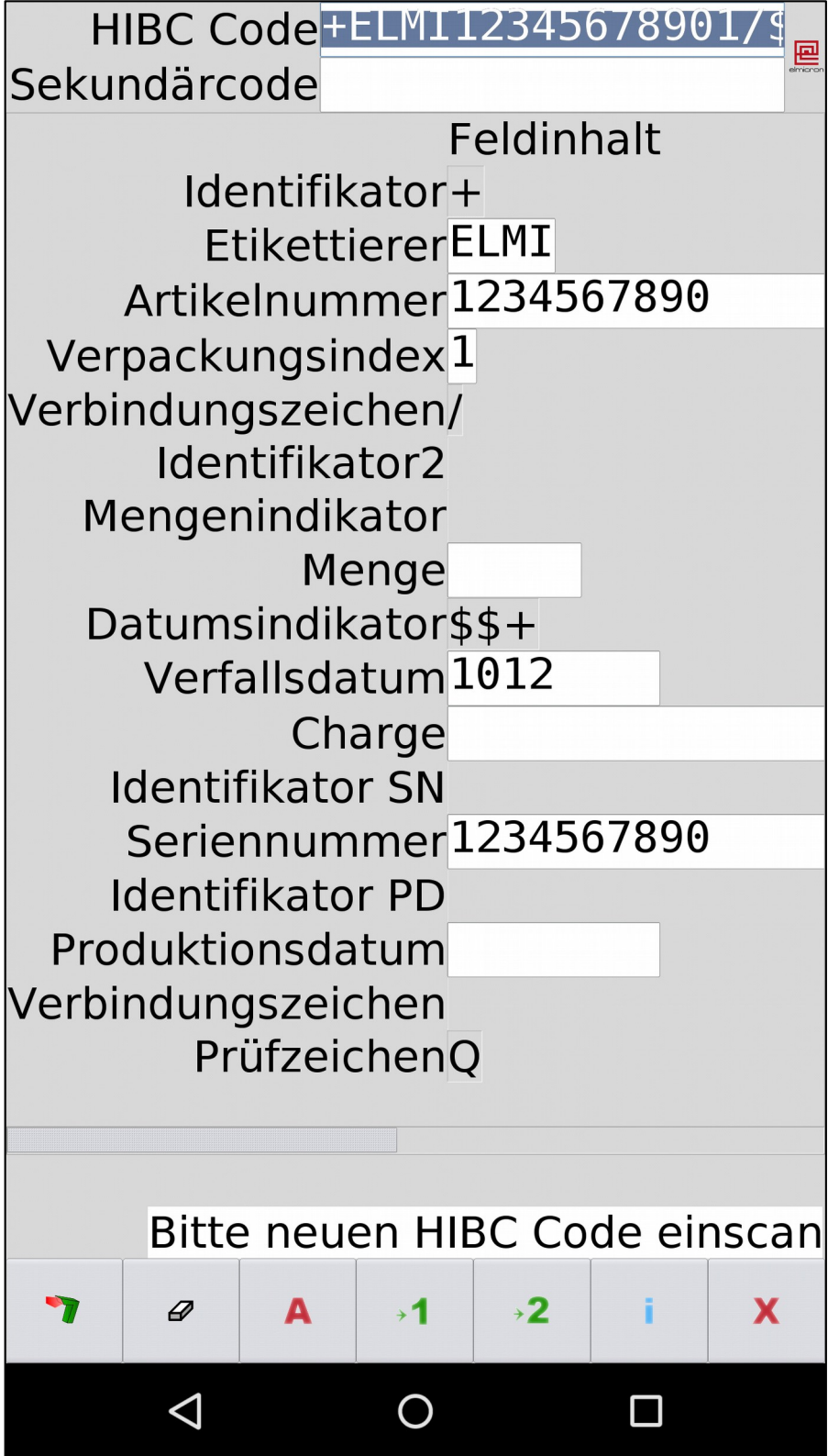
Button line with 2 lines hight

- Frame with:
  - A frame to set height
  - Packed Buttons with -fill both -expand true
- Buttons contain one of two images, 32x32 and 64x64
- Frame height and image choice changes with "pinch to zoom"
- Call "sframe resize"

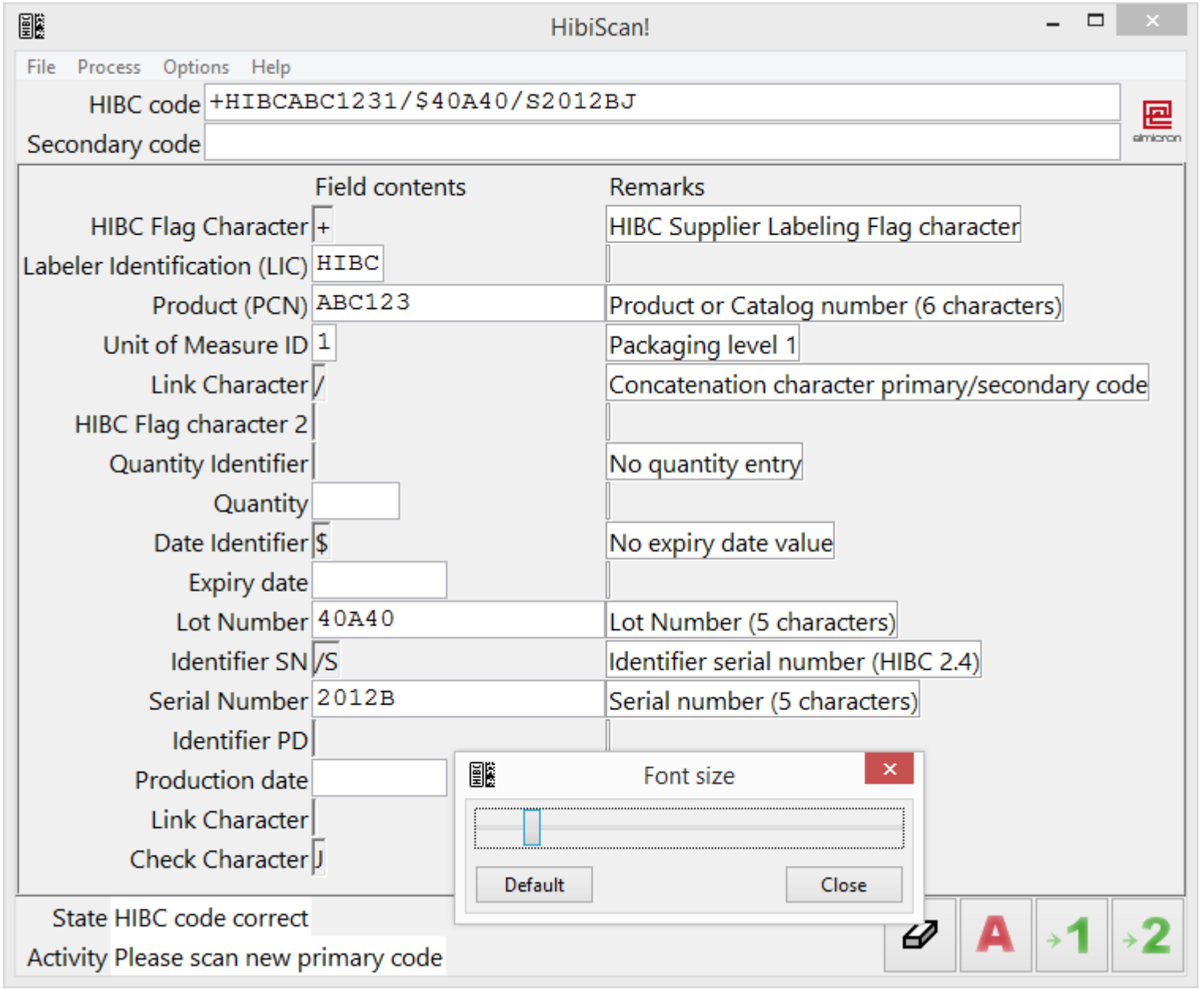
```
proc NewFontSize {Size} {
  # Set all font sizes
  font configure Font -size $Size
  sframe resize $P

  # Measure widget with two text
  # lines
  $LabelWidget configure -font Font
  set Height [winfo reqheight\
      $LabelWidget]

  #Change images to $Height<64?32:64
  # Set button height frame widget
  set BuHeight [expr {
      entier($Height/2)}]
  if {$BuHeight<32}{set BuHeight 32}
  $HeightFrame configure\
      -height $BuHeight

}
```

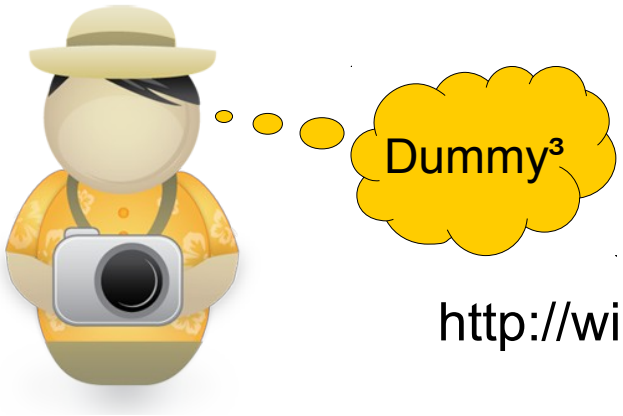
Final program Android

- Looks ok on Nexus with resolution of 2400 pixel
- Looks also ok on HTC with resolution 800x320 pixel



Final Program: Desktop





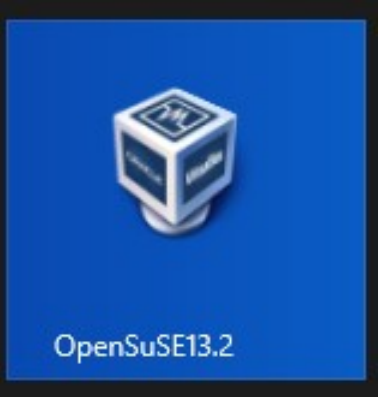
The APK story

http://wiki.tcl.tk/AndroWish tells you the way (CW):

Please fetch the sources (the big .tar.bz2), unpack it, have Android SDK and NDK installed, don't use Eclipse, adapt local.properties to where you've installed Android SDK, have your PATH properly set so that ndk-build can do its job, then invoke "ant debug", be patient, and you'll finally will have bin/AndroWish-debug.apk ready to be installed onto your device. I have never verified the build process in combination with Eclipse. Once upon a time, I did my very first steps using the tips from the SDL documentation regarding Android.
When you want to wrap your own app written as Tcl code, you should add it below assets/app, have the launching script as main.tcl, fiddle the toplevel AndroidManifest.xml to have your app/class name in, remove that AndroWishScript/Launcher stuff from the manifest (since not needed for a standalone app), derive your app main class (yes, some Java required) from src/tk/tcl/wish/AndroWish.java, e.g.

```
import tk.tcl.wish.AndroWish;
public class TclTkRules extends AndroWish {}
```

fiddle the res directory with a new really kooool icon and title for your app.



Steps to success

- Build AndroWish: SuSELinux, Java7, SDK, NDK9, locale.properties, ant debug
- Customizing AndroWish:
  - Delete unneeded stuff: packages, x86, fonts
  - Add own scripts (in assets/main.tcl)
  - Change package name (including java class)
  - Only required permissions (no phone, bluetooth)
  - Start with intend action
  - Icon
- Prepare signing -> ant release -> 7.8MB apk


Thanks to Christian Werner!

TCL/Tk activities

- BWidget: Koen Dankart, Kevin Waltzer
- msgcat (TIP399/412) Jos Descoster
- Core Windows socket driver (TIP427/428): Reinhard Max
- Zint Bar code generator TCL interface: Robin
- Rivet (Massimo Manghi):
  tcl survives fork (Alexandre Ferrieux) ,
  form package (Lehnbauer), rpm's (Reinhard)
- Img patch (write resolution field of bmp): Jan Nijtmans
- Core Windows make file: Donal(d)
- TCL Web Service (Gerald Leister)

TCL Wishlist

- Help in the TIP process
- TIP for OO-msgcat
- TCL WinCE (ETCL 8.5.7) (Bar Code Terminals)
- Themed tabbed widget with scrolling of tabs
- List equal ($L1 eql $L2), also in switch
- Dict equal -noorder/-order
- Windows 8 "App" Theme
- Android Theme
- Transparency in Tk