Implementation Guide

Contents

[Document History 2](#_Toc127055778)

[Purpose 2](#_Toc127055779)

[Overview 2](#_Toc127055780)

[Hardware 2](#_Toc127055781)

[Power Methods 3](#_Toc127055782)

[Powering via Battery (Default) 3](#_Toc127055783)

[Powering via Micro USB Cable 3](#_Toc127055784)

[Device Functions 4](#_Toc127055785)

[Software 4](#_Toc127055786)

[nColorConvert 5](#_Toc127055787)

[GalleryPalette2BIN 6](#_Toc127055788)

[Mobile Application Configuration 6](#_Toc127055789)

[IOS 7](#_Toc127055790)

[Android 7](#_Toc127055791)

[Mobile Application 8](#_Toc127055792)

[Connect to Device 9](#_Toc127055793)

[Write Card 10](#_Toc127055794)

[Display Card 11](#_Toc127055795)

[Delete Card 11](#_Toc127055796)

[Delete All Cards 11](#_Toc127055797)

[Disconnect 11](#_Toc127055798)

# Document History

|  |  |  |
| --- | --- | --- |
| Date (Rev): | By: | Description: |
| 2/1/23 (1) | LW | First issue |

# Purpose

This document describes the process for converting and transferring images to the E-ink screen, using a reference mobile application and development board.

# Overview

Upon unboxing, the e-ink screen must be attached to the development board, and power must be applied using one of the supported methods.

Before images can be transferred to the screen, they must undergo a transformation. First, their color spaces are transformed using the nColorConvert utility. Later, the resulting images are processed using the GalleryPallette2Bin utility, to prepare them for the digital transfer process using the mobile application software.

After conversion, the images are loaded at runtime by the mobile application, allowing for them to be transferred to the device when the appropriate commands are selected by the user.

# Hardware

The packaging includes one development board, along with a 4-inch E-ink screen. Upon unboxing, the screen must be attached to the development board as depicted in Figure 1 below. After the screen is attached, power can be applied to test a proper connection.

Text

Description automatically generated with medium confidence

Figure Dev Kit Contents

## Power Methods

The development board supports two power methods, with battery power being the default. Only one power mode is supported at a time, so please make the best decision for the chosen application.

### Powering via Battery (Default)

The default method to power the development board is to place two 2032 Coin Cell batteries into the battery holders. Once in place, the device will immediately power on and display a test pattern upon first use.

### Powering via Micro USB Cable

To use a Micro USB cable with the development board, remove the batteries and short the highlighted region using a jumper. If there is never a desire to use battery power, a conductive epoxy can be applied to bridge this connection permanently.

A screenshot of a computer

Description automatically generated with medium confidence

Figure USB Shorting Diagram

## Device Functions

The development board implements a state machine with the following states and rules:

* Once power is applied, device boots, shows first card if loaded, otherwise a test pattern, then proceeds to advertise for 60 seconds.
* If the device was asleep, a button press awakes the device and it advertises for 60 second, then sleeps.
* A long button press (2 seconds) wakes the device, advertises for 60 seconds, then goes to sleep.
* A short press paginates sequentially to display the next card slot. Each press resets the awake timer to 60 seconds.
* Button presses are ignored when paired with a mobile app and the awake timer is disabled.
* When paired with mobile app, the device can receive Bluetooth commands as per the application interface document.

# Software

This guide makes use of nColorConvert and GalleryPalette2BIN executables, which were provided by E INK Holdings, Inc. There is also a reference Xamarin Forms application that fully implements all supported functions, available as an IOS and Android mobile app. The process follows a flow described in Figure 3.

Figure Application Flow

## nColorConvert

nColorConvert is leveraged to convert source images into a valid color space.

1. Source images should be 640 X 400 (or as close to possible for best results).
2. Copy desired images to the docs\nColorConvert\input folder
3. Click on the nColorConvert.bat file
4. This will copy an image\_proof (image preview) and image\_fp (source file) for each image to the output folder as illustrated in Figure 4.
5. If you would like to process an individual file, Run the following command from the root directory, nColorConvert.exe --res 640x400 --image image\_name

Graphical user interface, text, application

Description automatically generated

Figure NColorConvert Execution

Notes:

1. you can review the GalleryPaletteConvert User Guide v1.2.1 User Guide for more specialized instruction.
2. The source images included in the sample were provided by E-Ink holdings.

## GalleryPalette2BIN

GalleryPalette2Bin converts the resulting color space files into a binary format that is interpretable by the E-ink screen.

1. Copy all source files to the docs\nColorConvert\input folder
2. Click on the GalleryPalette2BIN.bat file
3. This will create image\_fb.bin files for each source file in the output folder as illustrated in Figure 5.
4. In order to process an individual file, Run the following command from the root directory, GalleryPalette2BIN.exe --epd AC040TC1 --image image\_name\_fb

Note: The resulting bin files are ready for transfer to the mobile application.

Graphical user interface, application

Description automatically generated

Figure GalleryPaletteConvert Execution

## Mobile Application Configuration

The reference IOS/Android mobile application is built using Xamarin forms and Visual Studio. Free versions of this software can be downloaded using the links below.

Visual Studio: https://visualstudio.microsoft.com/downloads/

Xamarin (PC) https://visualstudio.microsoft.com/xamarin/

Xamarin (MAC) <https://visualstudio.microsoft.com/vs/mac/xamarin/>

When installing Xamarin, ensure that both IOS and Android tools are installed if there is a plan to develop for both platforms. Please note that this program cannot run on a emulated device as Bluetooth is needed in order to transfer data to the Development Board.

### IOS

If IOS is needed, an active Apple Developer Account is required, along with the ability to set up a provisioning profile which is outside the scope of this documentation. A compatible version of Xcode must also be installed on the MAC machine. Once everything is configured, plug in the IOS device, and execute the program using Visual Studio for MAC. In Figure 6, Iphone(2) is a physical device that is ready to run the application once the play button is pressed.

Graphical user interface, text, application, email

Description automatically generated

Figure Deploying from Visual Studio for MAC

### Android

Android deployment is a bit simpler. After opening the project, simply make sure your phone is in development mode and USB debugging is enabled. Once the phone is plugged into the computer, pressing the green play button as per the screenshot below will deploy the application to the connected device. In Figure 7, there is a Google Pixel device ready to run the application .A screenshot of a computer

Description automatically generated with medium confidence

Figure Program Execution for Android Device

# Mobile Application

The mobile application implements the specification as outlined in the Application Interface Documentation. The interface offers options to connect to the device, select the device slot of interest, and offers a menu of associated management actions for cards.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure Mobile Application Interface

## Connect to Device

This method Initiates Bluetooth Scanning, and pairs with the development board. Click this button once the device is powered and awaiting connection. Upon a successful connection, the connect button is disabled, the remaining buttons are activated, and the scanning process stops.   
Graphical user interface, text, application, chat or text message

Description automatically generated

Figure Button State after Connecting

## Write Card

This method loads a source file into memory and transfers it to the selected card slot. Cards are pulled from the embedded resources folder inside of the EinkStarter project. Copy the bin files that resulted from executing GalleryPallette2Bin into the DigmeStarter\EmbeddedResources .

A screenshot of a computer

Description automatically generated with medium confidence

Figure Embedded Resources Folder

After copying files to this folder, right click on each file and set its type to embedded resource and copy always.

A screenshot of a computer

Description automatically generated with medium confidence

Figure Embedded Resources Setting

Images are loaded at runtime and are available in the 2nd dropdown. Select desired device slot, along with the image name. Click the write card button to initiate the transfer process. The card is divided into chunks based on the connection NTU. Each chunk is set over one by one until they have been sent as per the application interface specification. In the example in Figure 12, ad1\_fb\_packed.bin has been written to Slot 1 on the device.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure Writing Card Sequence

## Display Card

This method changes the display to show the card in the selected slot (if present). The slot can be changed by selecting a different value in the 1st dropdown.

## Delete Card

This method deletes the card in the selected slot (if present). The slot can be changed by selecting a different value in the 1st dropdown.

## Delete All Cards

This method deletes all cards on the device.

## Disconnect

This method disconnects and unpairs from the device.