# Driving E Ink Segmented Displays

## Electronic Paper overview

E Ink segmented displays are direct drive displays consisting of E Ink Vizplex Imaging Film sandwiched between two electrode layers, the top plane (TP) and the backplane, and then encased in an environmental barrier solution to protect the film and the segment electrodes. To get the most from E Ink segmented Displays, you need to be aware of proper design and implementation. Your reward will be sharp, precise images, a pleasant transition from one image to the next, and a long battery life.

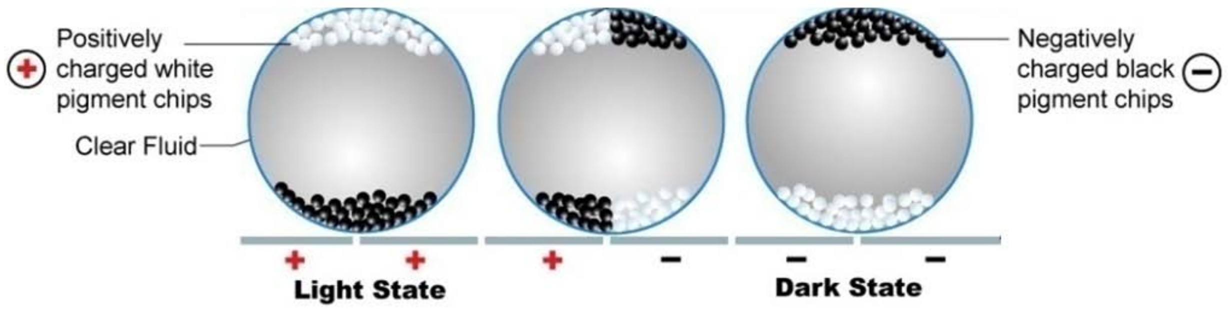

Positively charged white pigment chips
Clear Fluid
Negatively charged black pigment chips
+
+
+
−
−
−
Light State
Dark State

**Figure 1: Internal composition of an E Ink display**

E Ink segmented is an EPD (electrophoretic display) – a low power ultra thin display that uses electricity to move oppositely charged black and white particles, inside of tiny (sub 100 micron) microcapsules. The segmented display cell is divided into segments ("areas") that can be activated independently of each other. By applying a specific voltage to the display for a specific finite period of time will turn a white segment black or a black segment white. A segmented display is one where, for every segment, there is a corresponding electrode.

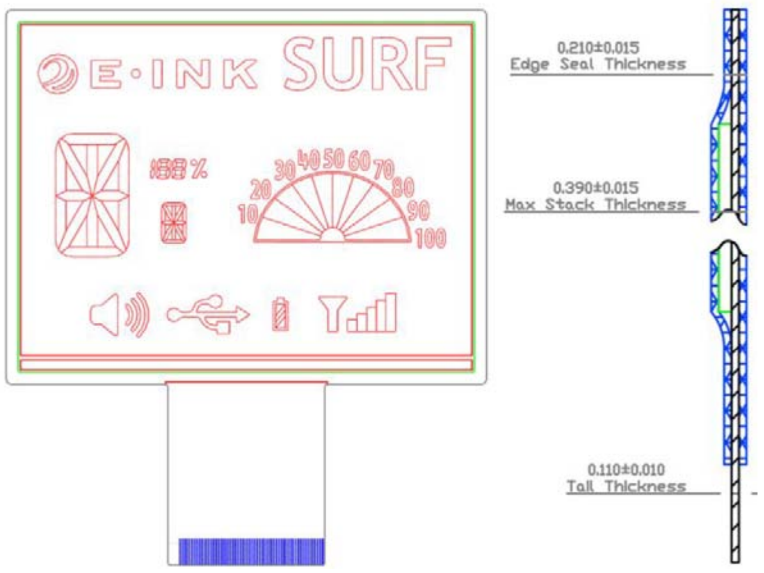


**Figure 2: Sample CAD drawing of an E Ink segmented display**

EPDs are addressed electrically only for fraction of a second (usually about 240 ms when driving the electrodes with 15V or 720ms or more when driving with 5V), consuming power only when the image changes making them fundamentally different from traditional displays. EPDs retain an image when disconnected from a driving source without consuming any additional energy (e.g. 0mW). Conversely traditional display technologies require a constant energy source to maintain an image. When driving an EPD, you need to know the current image as well as the next image to which the display will be driven. With these two pieces of information, you can create waveforms to drive your E Ink segmented e-Paper display properly.

## Waveforms

Waveforms work by defining the electrical impulses needed to transition from one desired optical state to the next. For segmented products, which are 1-bit devices, there are two optical states: black and white.

The EPD's optical state changes only when there is a difference in voltages between the backplane and

the top plane electrodes. To help make development easier, below are the waveforms shown for switching between the optical states when using a bilevel drive using 0 – 5V for 1 sec. A Bi-level drive display driving scheme is where the top plane and each segment can be driven with 0V or +V$_o$.

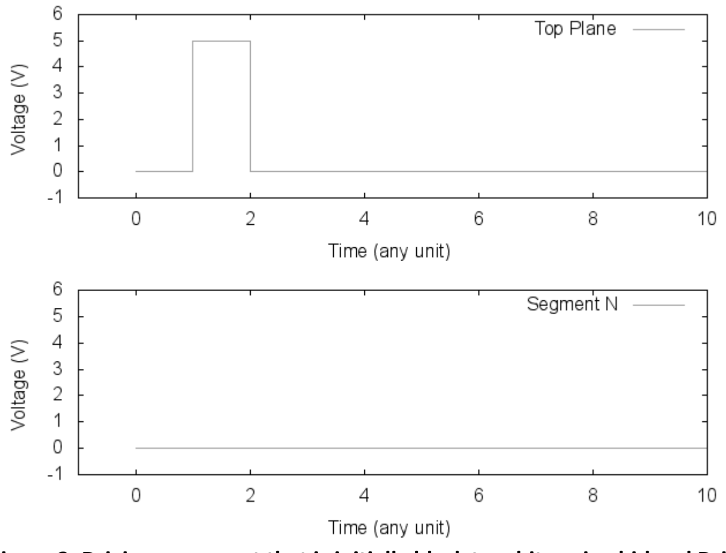


**Figure 3: Driving a segment that is initially black to white using bi-level Drive.**

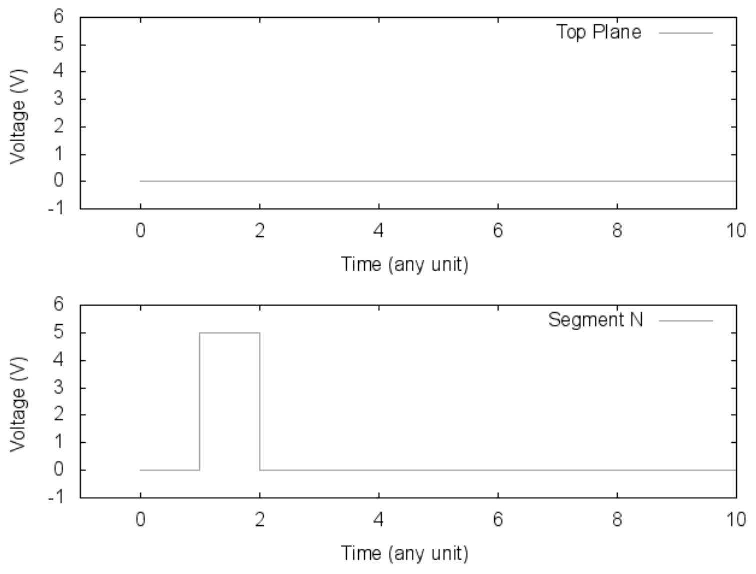


**Figure 4: Driving a segment that is initially white to black using bi-level Drive.**

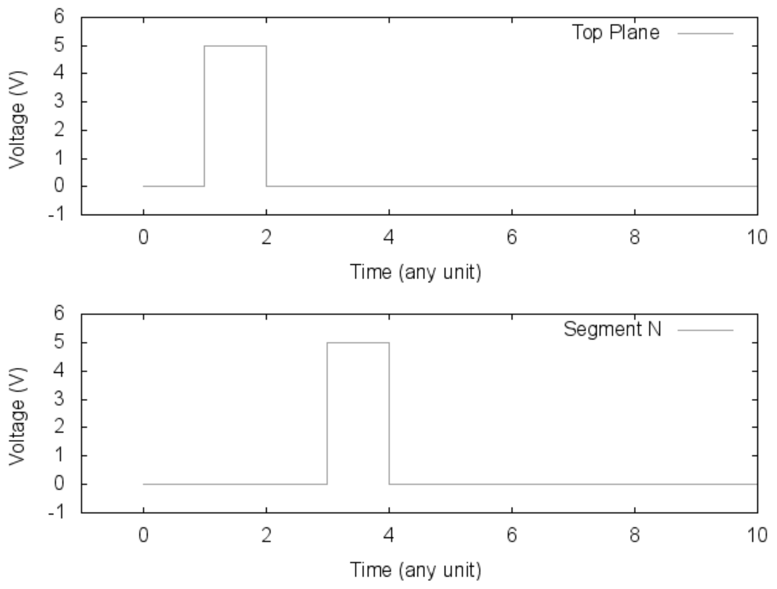


**Figure 5: Driving a segment that is initially black to black using bi-level Drive.**

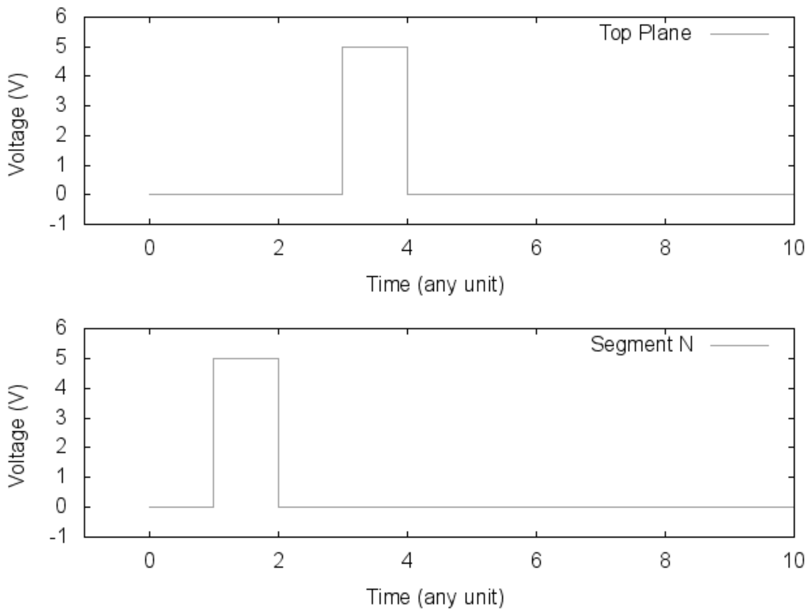


**Figure 6: Driving a segment that is initially white to white using bi-level Drive.**

Implementations of bilevel drive use less power and also cost less than other options. Although bilevel drive uses only two voltage levels, you achieve proper optical switching by electrically switching both the top plane and the individual segments. This means that, at any one time, segments on the display can see a voltage differential in only one direction. While this has some implications for transition appearance and restricts how the waveform can be constructed, it does allow for switching the display using the two voltage level solution, however, this tradeoff is offset by lower power consumption, a smaller footprint (board space), and improved battery life.

## Waveform Types:

The waveform also determines how the display looks as it switches. Once again, as long as you maintain DC balance, you can create your own waveform or modify an existing waveform to meet your needs. You can choose, for example, whether black segments switch first or white segments switch first, depending on whether your background is white or black.

While you will get the best static image quality by flashing and clearing the whole display, you may decide that does not fit with the usage model or it may take too long or it does not have the desired visual appearance. You might choose a local update waveform, as described in the later sections to get the visual effect you are seeking. Overall, some mix of waveforms below should fit your individual situational needs.

### Slide Show Waveform:

What distinguishes a slide show waveform is that it has both a positive and negative pulse for at least one transition. You can use slide show style waveforms to help improve the appearance of transitions within your display. Slide show waveforms provide a pleasant appearance when switching since there is no overlap between the current image and the next image. However, objects that do not change will also flash.

In a Slide Show White Waveform, every transition travels through white. This means that the B->B transition flashes white and back to black instead of holding in black, which gives a clean transition appearance when switching characters. This is excellent for use for images with white backgrounds since it erases old information completely before displaying the new, with no background flash. Slide show is especially effective for updating alphanumeric displays.

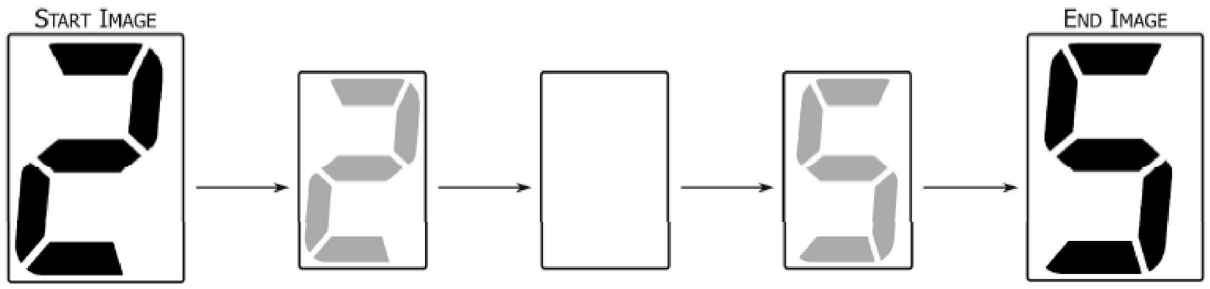


**Figure 7: Slide Show white waveform transitions**

In a Slide Show Black Waveform, every transition travels through black. In other words, the W->W transition flashes black instead of holding in white, which gives a clean transition appearance when switching characters. Excellent for use with black backgrounds, it is especially effective for updating alphanumeric displays, since it erases old information completely before displaying the new image, with no background flashing.

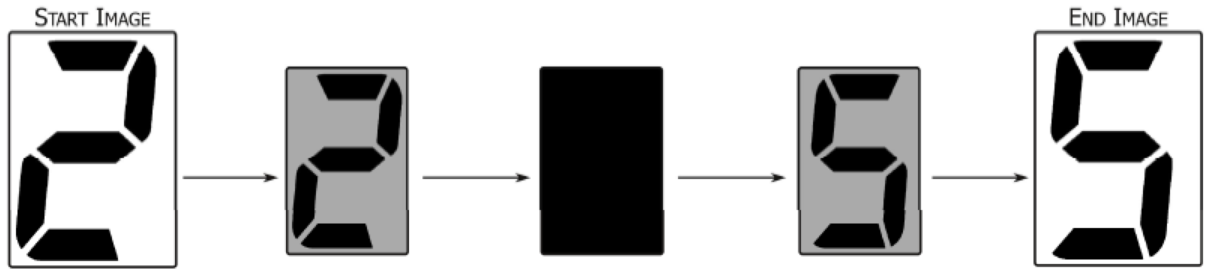


**Figure 8: Slide Show black waveform transitions**

### Global Update Waveform:

Global updates are subsets of slide show waveform where all the segments will switch – hence the term "global." Global waveforms give you the best possible performance and are the best in terms eliminating ghosting. For applications that require short update times or less flashy transitions, Global waveforms are recommended for device power on/off, as well as to periodically refresh the display during device operation to reduce ghosting.

In a Global Black White Waveform, all segments switch to black, then to white, before they display the final image. This waveform will clear any image artifacts, including ghosting, from the display, and, with optimal settings, provides the best image performance.

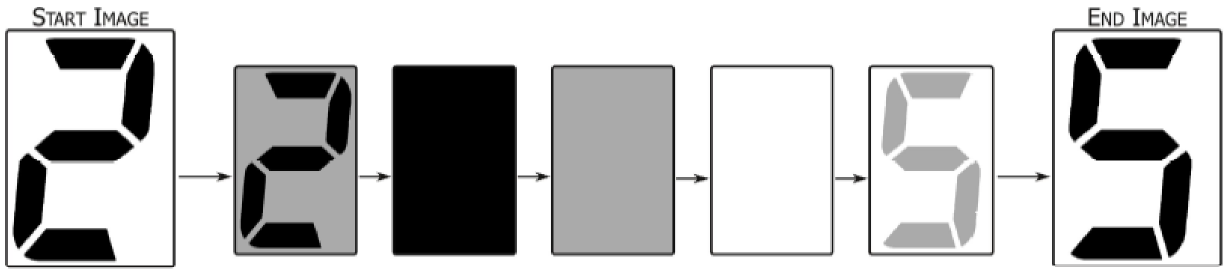


**Figure 9: Global Update White waveform transitions**

In a Global White Black Waveform, all segments switch to white, then to black, before they display the final image as shown in Figure below. This waveform clears any image artifacts, including ghosting, from the display, and, with optimal settings, provides the best image performance.

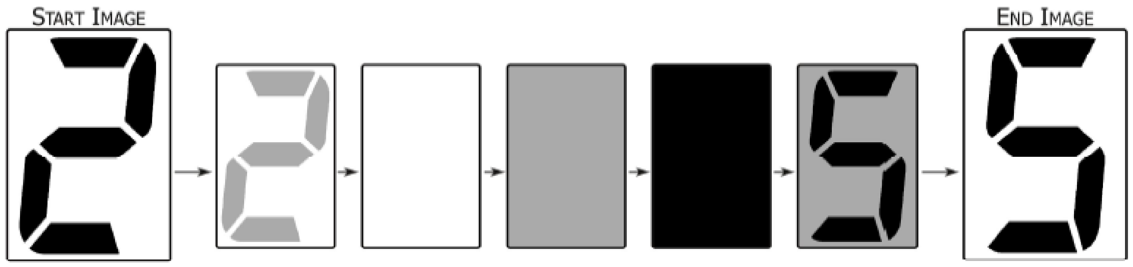


**Figure 10: Global Update Black waveform transitions**

### Picket Fence Waveforms

The Picket Fence (or "Picket Drive") Waveform is unique in that it alternates very rapidly between driving to black and driving to white. It switches the top plane between 0 to 5V (or 0 and 15V) for short periods of time so that to-white updates and to-black updates can be made incrementally. Picket Fence waveforms give smooth transitions between images and create the appearance that black and white segments are driven simultaneously, but the tradeoffs of the picket fence waveform is that it can increase ghosting and increase power consumption.

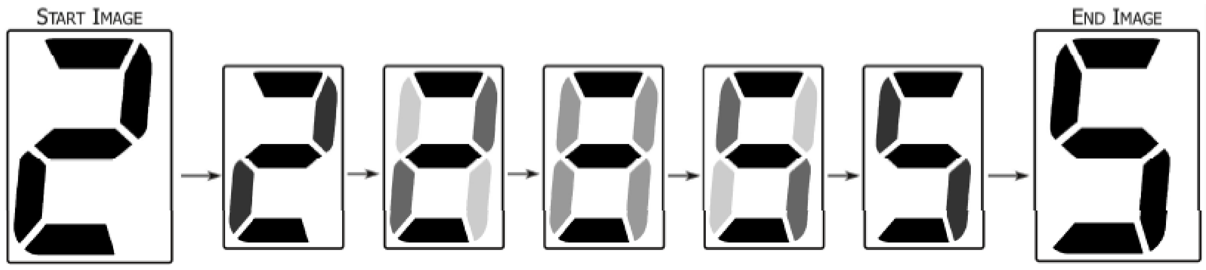


**Figure 11: Picket fence waveform transitions**

## Notes on driving the segments correctly:

## DC Balancing:

The most important element in waveform creation is to keep a systematic DC balance over time. DC balance is when the net voltage impulse, the sum of voltage over time ($\sum v \cdot t$), is near zero for all segments. DC imbalance can, depending on the amount and frequency of the imbalance, cause undesired image artifacts, shorten the life of your display or cause irreversible damage.

DC balance occurs when, for all transitions:

- the voltage impulse of all B->W transitions is equal magnitude but opposite sign of all W->B transitions;
- the voltage impulse of all W->W or B->B transitions individually equals zero.

## Overdriving:

For all E Ink segmented displays, make sure that the switching voltage is within specification: 5 to 15V (± 0.1V). The amount of time for which the display needs to be driven is inversely proportional to the voltage applied. When driving the display using 15V, the starting pulse length should be 240ms, and the recommended range is 200ms-480ms. Feel free to modify pulse lengths from the 240ms rule if you seek better optical performance. When driving the display using 5V, this time increases to a minimum of 720ms to a maximum of 2000ms. Your customized pulse length should remain within the recommended range.

If you drive the display so that the voltage impulse exceeds this range, you may produce what is know as overdriving, which should be avoided. Overdriving occurs when the display is driven too much in one direction and starts to behave erratically. Not only can it shorten the life of the display, it can also cause display artifacts (or “ghosting”), at temperatures above 35°C.

## Kickback

Kickback or self-erasing occurs when the optical reflectance moves away from the reflectance it was driven to and does within seconds after an update before leveling off. Visually, the image will appear then quickly fade. This can be caused by overdriving – driving the display too much in one direction. You can avoid kickback by making sure the voltage impulse remains within the recommended range of drive time for the selected voltage.

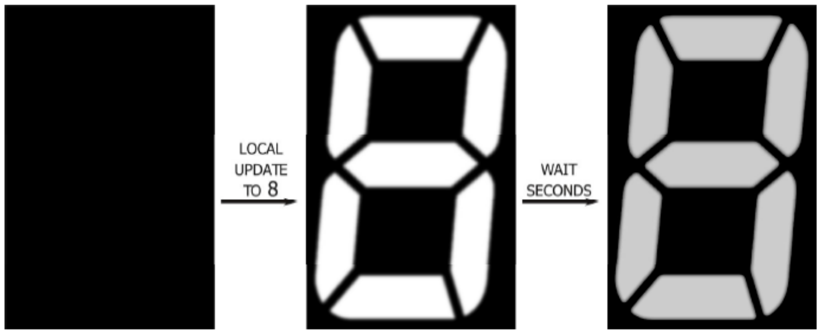


**Figure 12: An exaggerated example of the effects of Image Drift. A multiple minute wait causes whole display to drift slightly toward grey. While the information on the display is still readable, updating the display with a local update reveals the difference between the segments that have been recently updated, and those that have not.**

## Image Drift

Image Drift occurs when the reflectance changes over long time scales. This can be typically 2L* in the first ten minutes after a switch after which the change in reflectance levels off. Image Drift is one source of ghosting. Employing a Global Update Waveform will fix ghosting due to image drift.

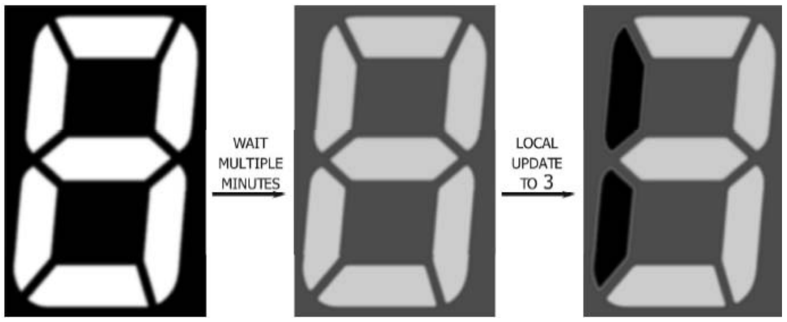


**Figure 13: An exaggerated example of Areal Ghosting. Here, the final image has areal ghosting. That is, there is a difference between the reflectance of the background and the two far left segments even though they are all in the black state.**

## Areal Ghosting

Areal Ghosting is the mismatch of two reflectances that should be in the same optical state. This can occur for a large number of reasons including kickback, image drift, and Dwell State Dependence (DSD). Adjust your pulse length to minimize this artifact.

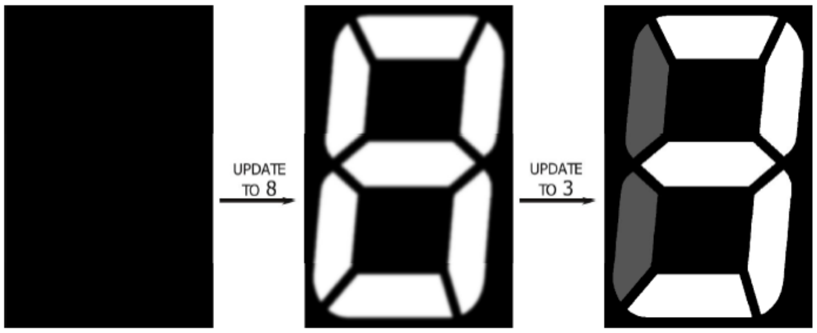


**Figure 14: An exaggerated example of Areal Ghosting. Here, the final image has areal ghosting. That is, there is a difference between the reflectance of the background and the two far left segments even though they are all in the black state.**

## Edge Ghosting

Edge Ghosting is a reflectance difference between two neighboring segments in the same state. It is most common when a subset of segments is frequently switched between states while their neighbors do not switch. Adjust the pulse length or use a global waveform periodically to minimize this artifact type.

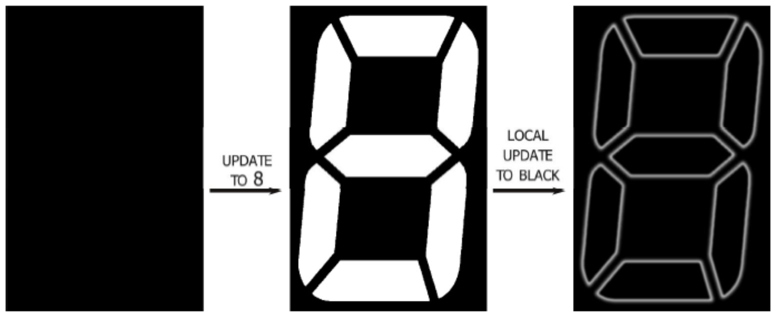


**Figure 15: An exaggerated example of Edge Ghosting. Here, the white outline of segments that can be seen in the final image is an edge ghost.**

## Best Practices:

1. Physically measure the signals sent to the display, and verify that they are what you expect.
2. When adjusting pulse length, do so in 10ms or 20ms increments. Generally, keep pulse length around 240ms for a 15V driver and 720ms for a 5V drive.
3. Take advantage of the different waveform types by incorporating and testing different sequences in with the design of the user interface.
4. If possible, leave about one second between transitions. This will help reduce image artifacts or ghosting.
5. The first time a device is powered up, the display should be cleared.
6. When testing power consumption, test at 25°C for typical and also at the upper limit of your device's operating temperature range (typically 50-60°C) for peak power consumption.
7. When a device is turned off, the controller should retain the image information on the display so it can be accounted for next time the device is turned on.
8. Test device in typical operating temperature.

## High Level Overview of Hardware

The Renesas RL78/G14 Group microcontrollers employ the RL78 core which realizes high processing performance while delivering the lowest power consumption in their class, and have enhanced computing power and peripheral functions suitable for driving E Ink displays, motor control and other applications.

In addition to a high-precision (±1%) on-chip oscillator enabling 32 MHz CPU operation and other standard on-chip functions from the RL78/G13, the RL78/G14 microcontrollers have multiply and divide and multiply-accumulate instructions capable of faster processing than the RL78/G13, and functions from the R8C Family with proven track records, such as timers RD/RG/RJ, data transfer controller, and event link controller, helping to lower the total cost of building a system and contributing to more compact size and lower power consumption.

The YLPDSKRL78G14 (shown in the figure below) employs the use of RL78G14 operating at 5 V for driving the E Ink display attached using a daughter card to the YRPBRL78G14. The display is driven directly by applying voltages from the MCU pins without the need for pull up resistors. The daughter board also has three switches for general use and provides all the necessary tools along with a sample project/driver/programming interface to demonstrate driving E Ink segmented displays while observing all the aforementioned constraints.

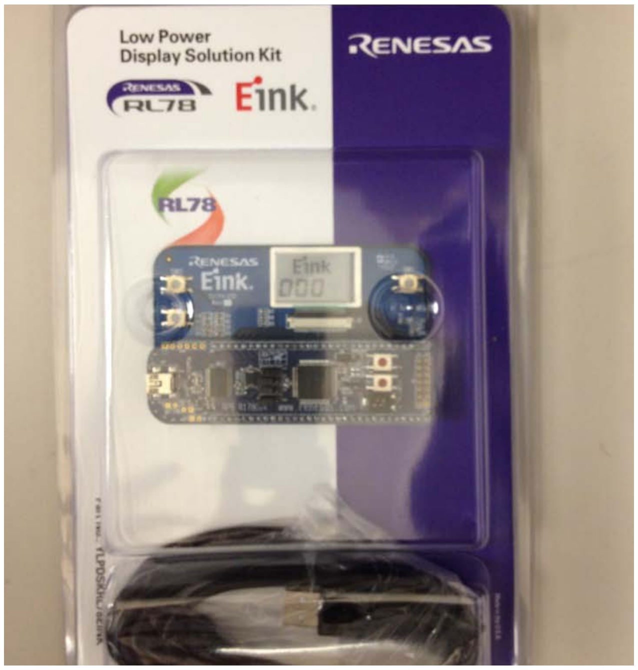


**Figure 16: YLPDSKRL78G14 kit**

The YLPDSKRL78G14 board has preloaded firmware with two demonstration projects for driving E Ink displays. When Powered ON, the display is refreshed and the MCU acts as a counter. Pressing switch SW1 increments the number displayed on the EInk screen, whereas SW2 decrements it.

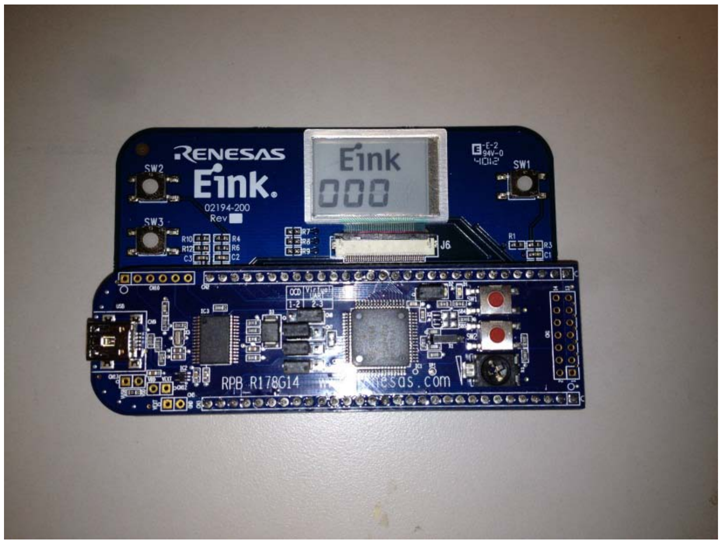


**Figure 17: The YRPBRL78G14 kit attached to the E Ink daughter board**

Pressing SW3 resets the displayed value to "000" and the MCU begins operating as a Timing unit, changing the value displayed by driving the display every 1 second for 720ms.

## Lower-level HW definition

The YLPDSKRL78G14 has a 36 position ZIF 0.5mm FPC connector which allows for connecting the E Ink segmented display. When looking at the display (top view), the connections from the FPC connector to the MCU port pins are listed from right to left are as listed below in :

| J6 Pin number | YRPBRL78G14 Connector name | MCU Port Name |
|---|---|---|
| 1 | CN1-17 | P60 |
| 2 | CN2-03 | P140 (TP1) |
| 3 | CN1-18 | P61 |
| 4 | CN1-19 | P62 |
| 5 | CN1-05 | P12 |
| 6 | CN2-05 | P00 |
| 7 | CN2-06 | P01 |
| 8 | CN2-07 | P02 |
| 9 | CN2-08 | P03 |
| 10 | CN2-09 | P04 |
| 11 | CN2-28 | P05 |
| 12 | CN2-10 | P130 |
| 13 | CN1-11 | P55 |
| 14 | CN1-12 | P54 |
| 15 | CN1-13 | P53 |
| 16 | CN2-20 | P120 |
| 17 | CN1-14 | P52 |
| 18 | CN2-21 | P43 |
| 19 | CN1-15 | P51 *(INTP2)* |

| | | |
|---|---|---|
| 20 | CN2-22 | P42 |
| 21 | CN2-23 | P41 |
| 22 | CN1-03 | P10 |
| 23 | CN2-25 | P146 |
| 24 | CN1-04 | P11 |
| 25 | CN2-26 | P147 |
| 26 | CN2-04 | P141 *(INTP7)* |
| 27 | CN2-27 | P06 |
| 28 | CN2-29 | P30 |
| 29 | CN1-10 | P17 |
| 30 | CN1-16 | P50 (INTP1) |
| 31 | CN1-24 | P75 |
| 32 | CN1-25 | P74 |
| 33 | CN1-26 | P73 |
| 34 | CN1-27 | P72 |
| 35 | CN1-28 | P71 |
| 36 | CN1-29 | P70 |

**Table 1: Pin connections for hardware**

Figure 18 is the schematic for the daughter board to the YRPBRL78G14.

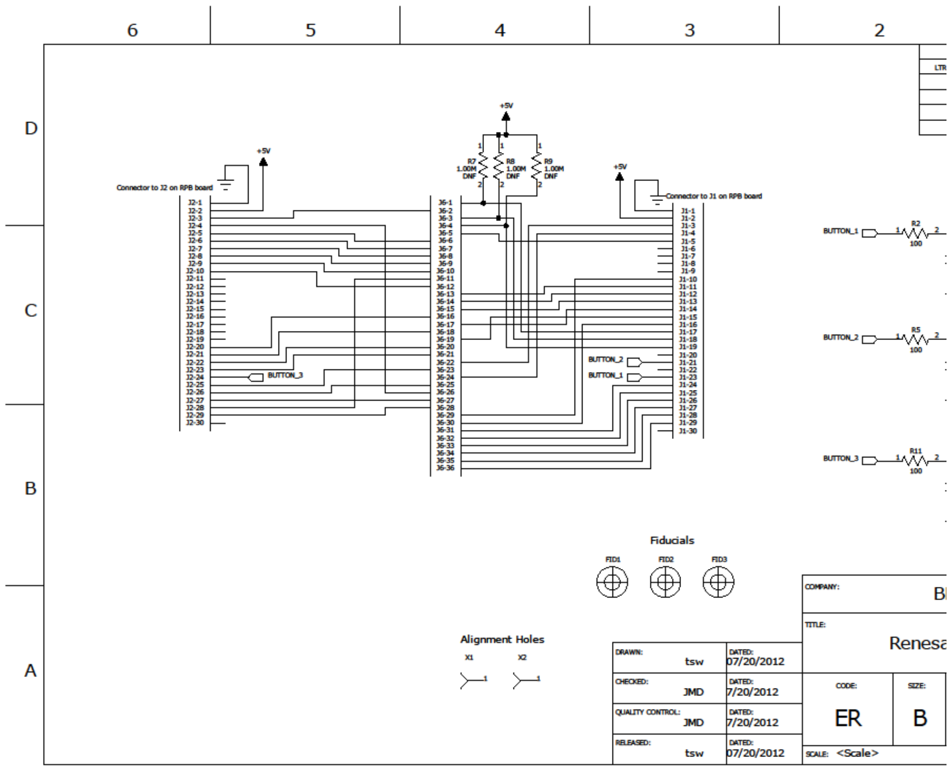


**Figure 18: Hardware Connections from YRPBRL78G14 to daughter board with 36 Pin Connector for inserting E Ink Display**

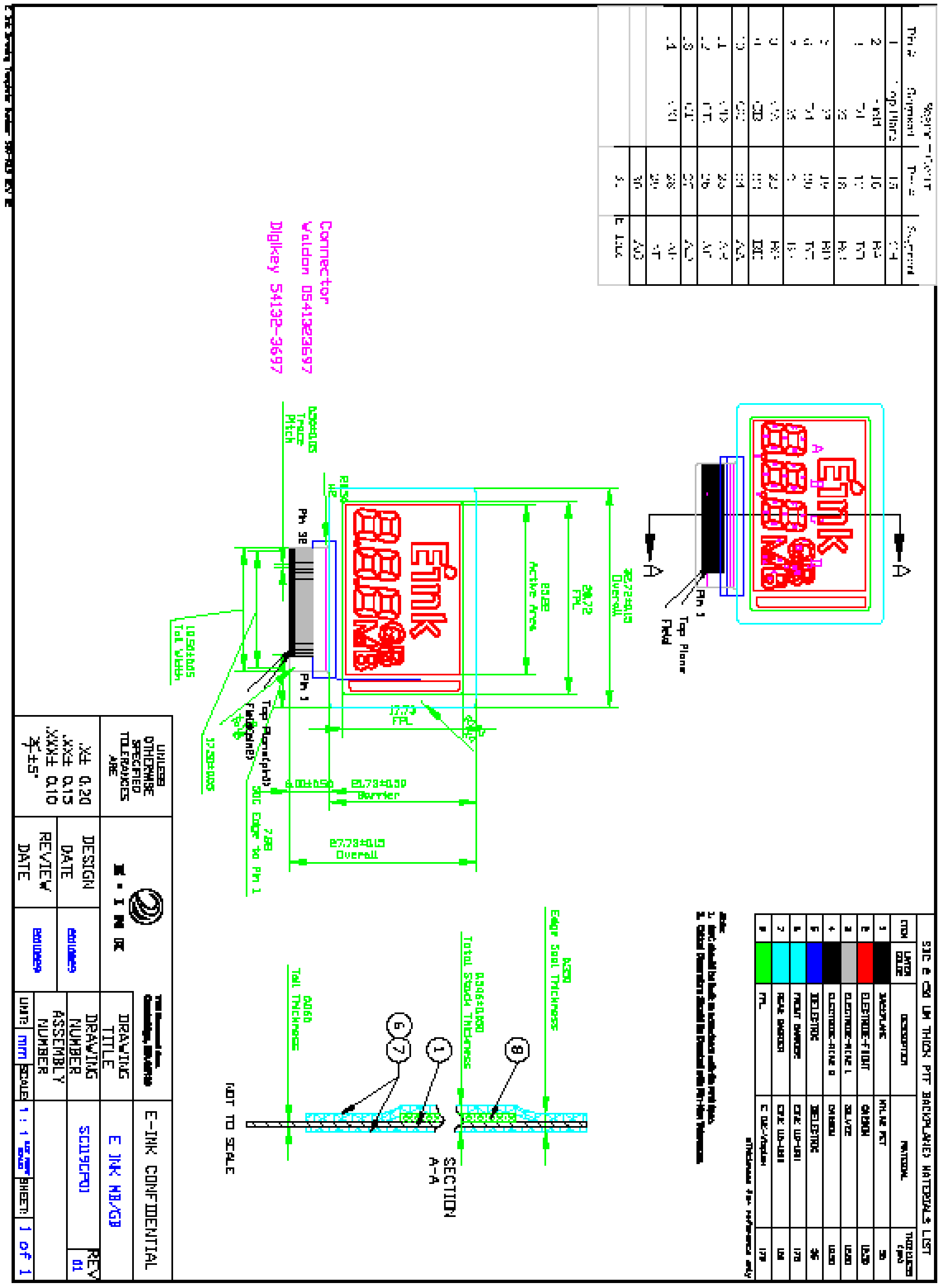
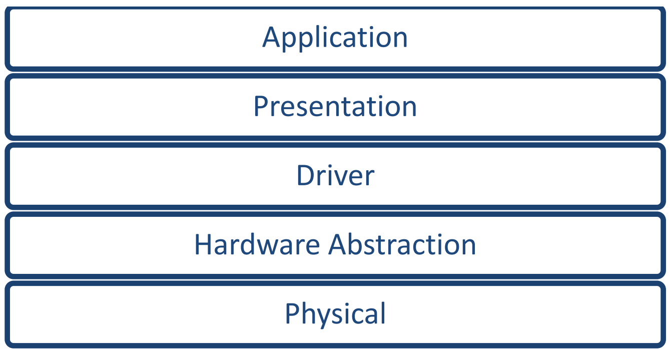


Figure 19: E Ink segmented screen pin connections provided with YLPDSKRL78G14

## Driver definition

The YLPDSKRL78G14 board has preloaded firmware with two demonstration projects for driving E Ink displays. The source code for the preloaded demos and code for driving the E Ink display segments is available royalty-free. By manipulating a few preprocessor definitions and a few variables, one can quickly adjust the driving time for the segments, adding a huge difference to the contrast of the screen. The API provided also allows the user to group segments together and make displaying characters and icons easier. Hardware abstracted port pin names allow the user to make changes to the firmware, making it easy to integrate the code into their project. The code for driving the E Ink segmented display is periodic and non-blocking (no 'while' loops present); allowing the processor to perform other duties while the screen is being updated. The source code is also very well documented to allow understanding the intricate details of driving E Ink displays. The 30 pin headers available on the YRPBRL78G14 kit are helpful for engineers to probe the signals being sent to the display when a display update occurs. If probing, it is helpful to have the top plane probed along with other segments.

The driver code is organized into different layers as shown below:



## The Physical Layer

The physical layer is a set of functions that manipulate the physical pin voltages for each segment of the EInk segmented display. This is done using the function "SetOutput" in the file rEInkDriver.c . To allow pin voltages to be toggled, the MCU port pins are initialized to the output direction using the function "initEInkHardware" also present in the rEInkDriver.c file. The user should add more statements to these functions in the already present format to extend functionality for the actual number of segments present.

The Hardware Abstraction Layer

The hardware abstraction layer is a simple set of preprocessor definitions that exist in the file rEInkHardwareAbstraction.h. This file is included in rEInkDrive.c and contains the names of each microcontroller port pin that maps to each segment of the EInk display. These definitions are replaced in the setOutput function when preprocessing of the source file occurs.

## The Driver Layer

The driver layer consists of the "einkDriver" function which is called periodically from a timer interrupt or from the main loop. "einkDriver" is organized as a non-blocking finite state machine. When not updating the screen, the driver is in the IDLE state checking if an update request is made to physically update the screen. When an update request occurs, it transitions to WAIT state. In the WAIT state, a call is made to "UpdateScreen" after a fixed delay. The "UpdateScreen" function selects the voltage for each segment depending upon the user's choice of waveform and the previously applied voltage. The selected voltage for each pin is applied using the "SetOutput" function.

### The Presentation Layer:

The presentation layer is useful when dealing with EInk displays that may have a lot of segments, digits, icons or combinations of the same. The "einkDriver" function reads the "nextDisplay" array (which defines the segments that need to be displayed next) and compares it with "currDisplay" (which defines the segments that are currently active) to determine which segments need transitions. This task of setting the elements of "nextDisplay" array can be difficult when there are a lot of segments in a display or if the segments are not grouped together in the standard 8, 9, 14 or 16 segment format.

This issue can be resolved by grouping the segments into logical digits and zones. A digit is a collection of segments. A zone is a collection of digits with the same number of segments. The information about this grouping is manually filled out by the user for a new display and passed to the function "displayInZone" which selects the correct mapping to use for displaying characters in that zone and sets the corresponding elements in "nextDisplay" to "1" depending upon the information to be displayed and the chosen zone passed as arguments to the function.

The information about the display is located in the file YLPDSKRL78EINK_display.c.

## Overview of using the driver and example code

The demonstration project is provided as an IAR Workspace/project. The main program running on the YLPDSKRL78EINK kit is as shown in Figure 20. Renesas Applilet Generator for RL78/G14 family was used to generate code for initializing and setting up the peripherals (port pins and Interval timer) for the MCU. The main while loop performs three tasks:

1. Checking if switches are pressed,
2. Performing the next step for the currently selected demo
3. Driving the screen if an update request is made.

Note that the function for driving the screen can be called through a periodic timer interrupt instead of the main while loop. Presently, the timer interrupt performs the task of providing an accurate delay.

The current value being displayed on screen is passed to the presentation layer function "displayInZone" from within the driver function "einkDriver" although this can be done in any other application layer function. The display is physically updated when the variable "updateRequest" is set to TRUE.

The user can use functions like "ClearDisplay" to clear the screen. Also, it is recommended for the program to wait for the state to finish the current update before setting "updateRequest" to TRUE again. This can be done by checking the variable “changingState”. For further documentation, see references item 2 below.

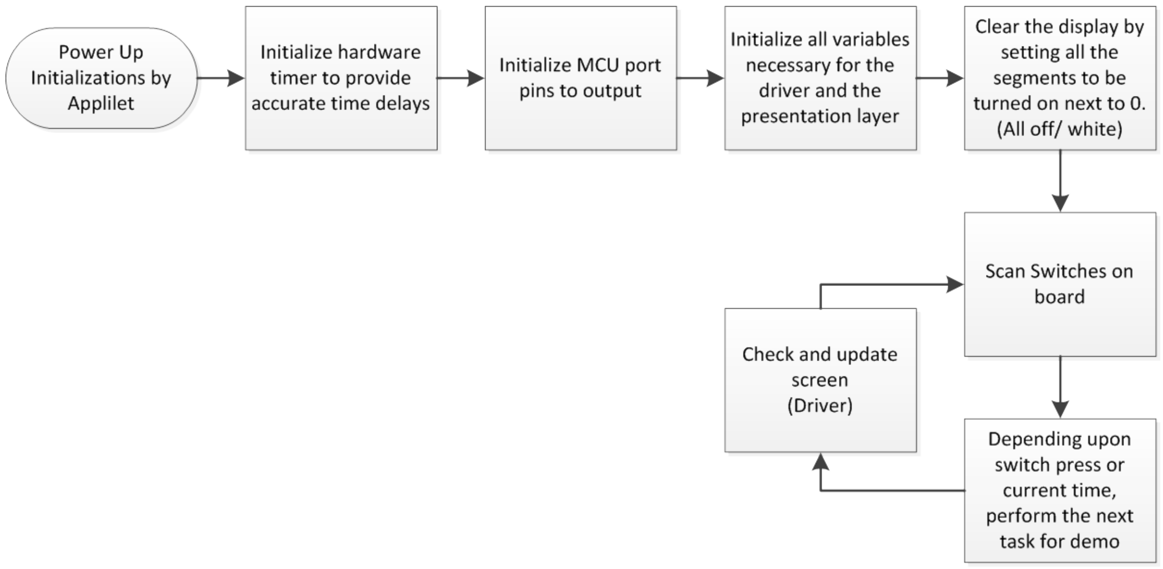

Power Up Initializations by Applilet
Initialize hardware timer to provide accurate time delays
Initialize MCU port pins to output
Initialize all variables necessary for the driver and the presentation layer
Clear the display by setting all the segments to be turned on next to 0. (All off/ white)
Scan Switches on board
Depending upon switch press or current time, perform the next task for demo
Check and update screen (Driver)

**Figure 20: Main program running on the YLPDSKRL78EINK**

## References:

1. YLPDSKRL78EINK main page: ********Link goes here********
2. YLPDSKRL78EINK sample project and driver workspace:********Link goes here********
3. Programming API and documentation manual:********Link goes here********
4. Applilet For RL78/G14:********Link goes here********
5. RL78/G14 family microcontrollers:********Link goes here********
6. Renesas Main Page:********Link goes here********
7. BNS Solutions:********Link goes here********