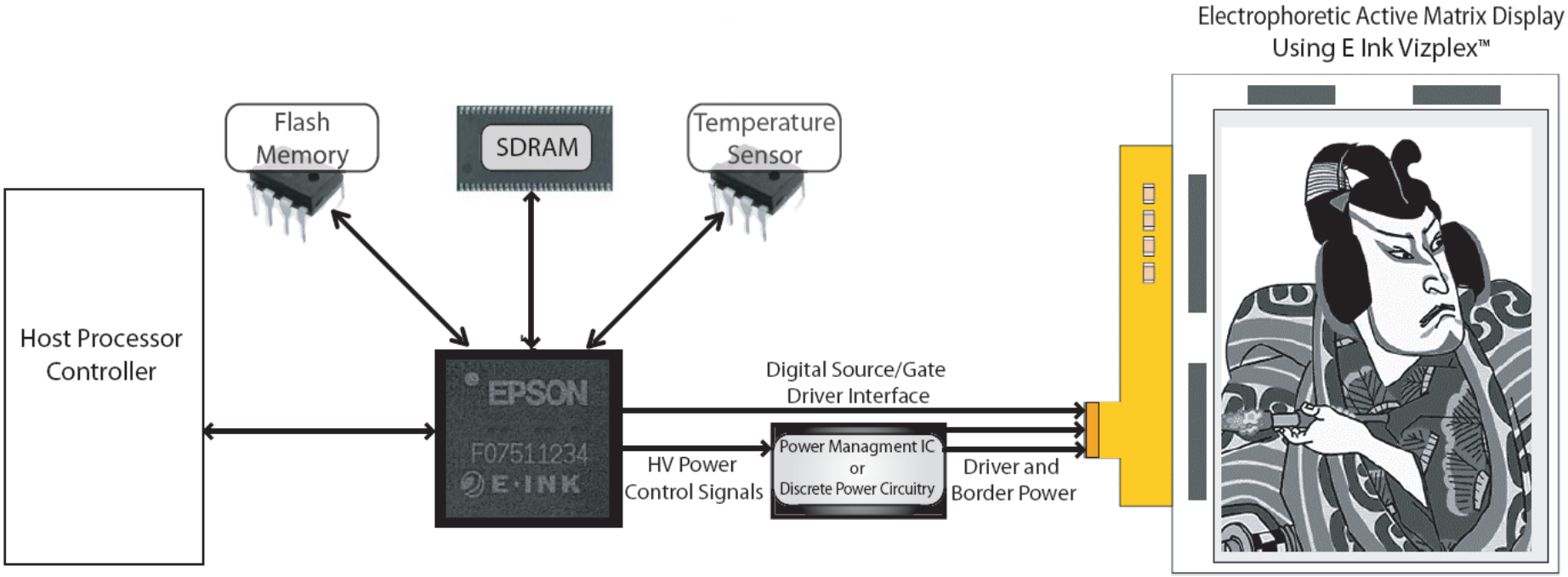
# What are waveforms?

- Waveforms define ink movement
  - Predefined voltage impulse sequence, stored in a Waveform Look-Up Table (LUT), used by the display controller to manipulate ink.
  - Paired to display by production lots. Waveforms reduce lot-to-lot variation, by enabling precise placement of pigment to achieve accurate graytones.
  - Offer sequences that enable pleasing UI transitions by managing material tradeoffs.

# Broadsheet Active Matrix Display System




**Host Processor Controller** - Device micro-controller unit (MCU)

**Flash Memory** - Contains host interface commands and waveform.

**SDRAM** - Image frame buffer storage location

**Temperature Sensor** - Input for waveform temperature compensation

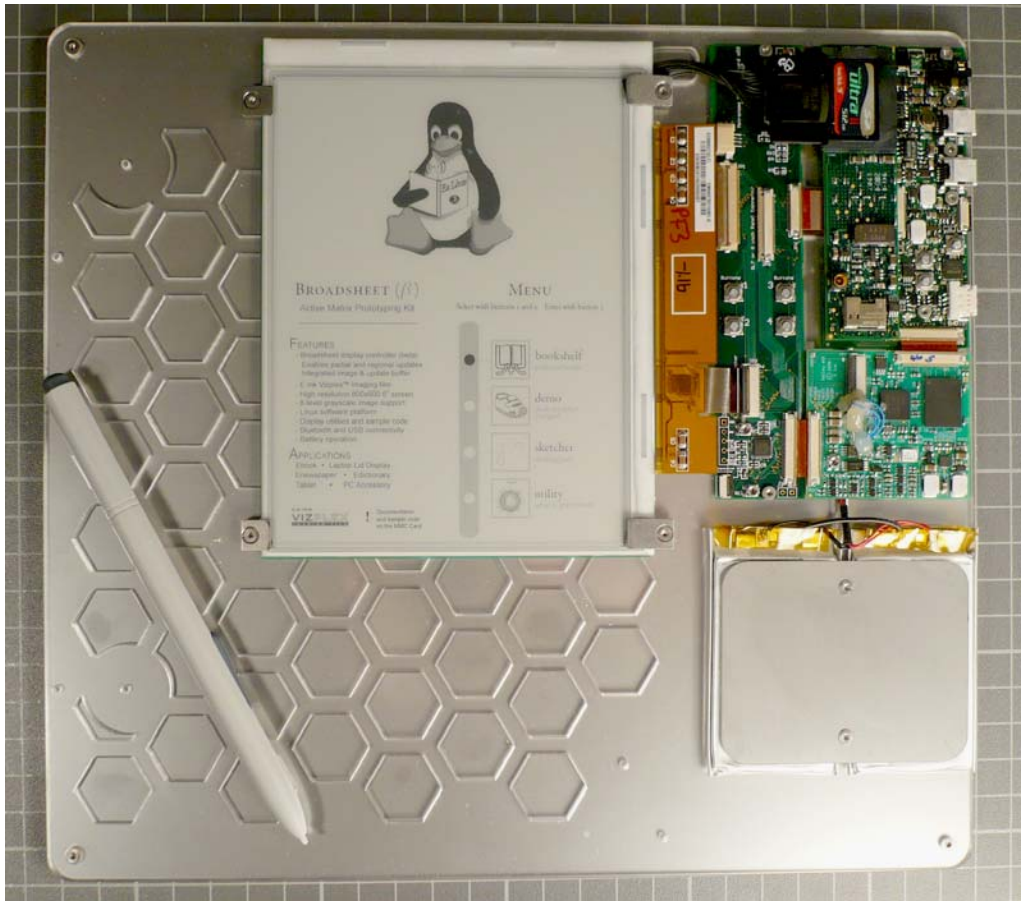
**S1D13521** - Electrophoretic display controller

**HV Power Supplies** - Provides display drivers and border analog voltage signals

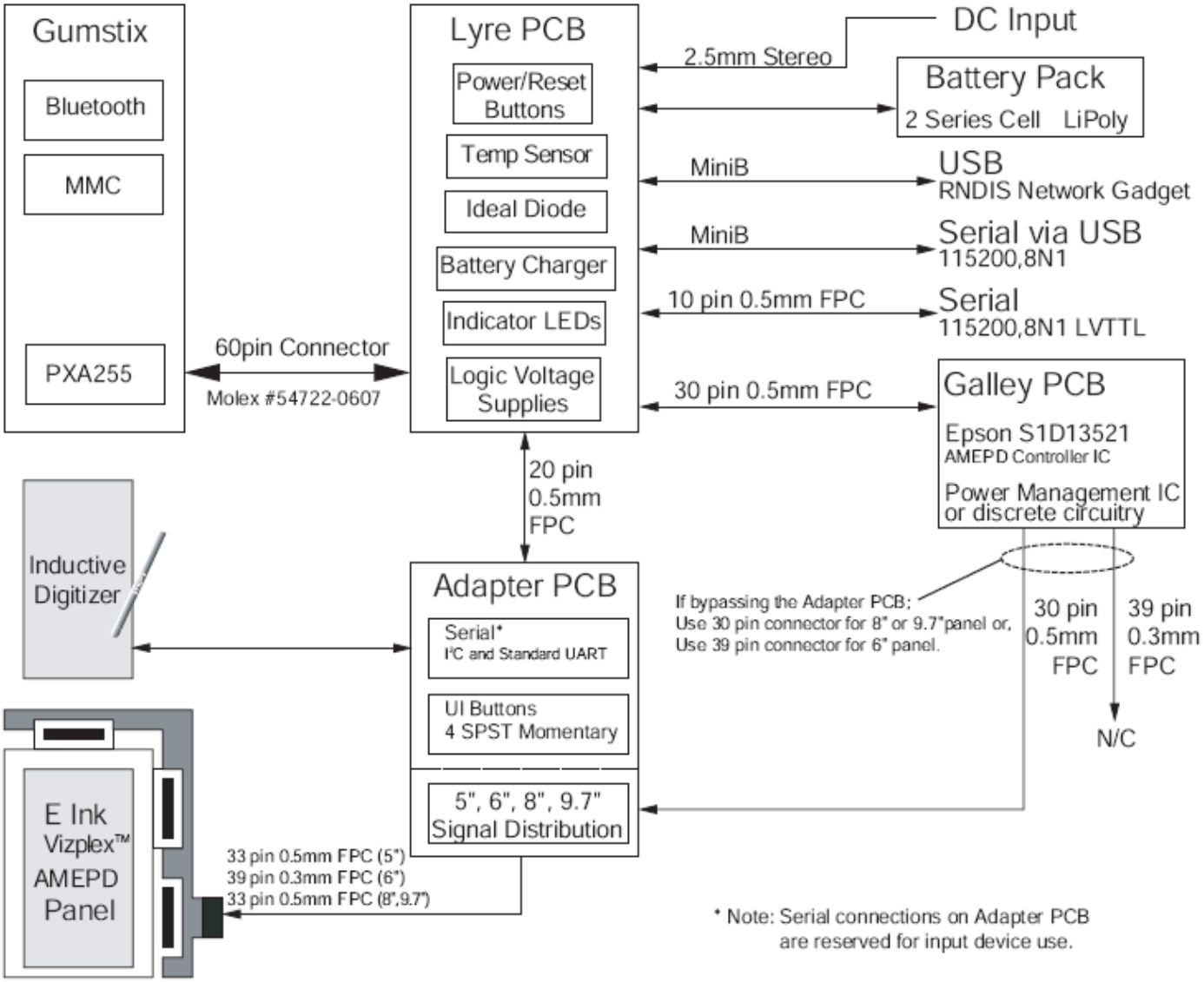
# Kit Overview

- *Fastest* way to start working with E Ink technology featuring Epson's S1D13521 EPD controller.
- *Create* functional, low-profile product mock-ups, using the modular kit design.
- *Compatible* with the 5", 6", 8", 9.7" active matrix displays.

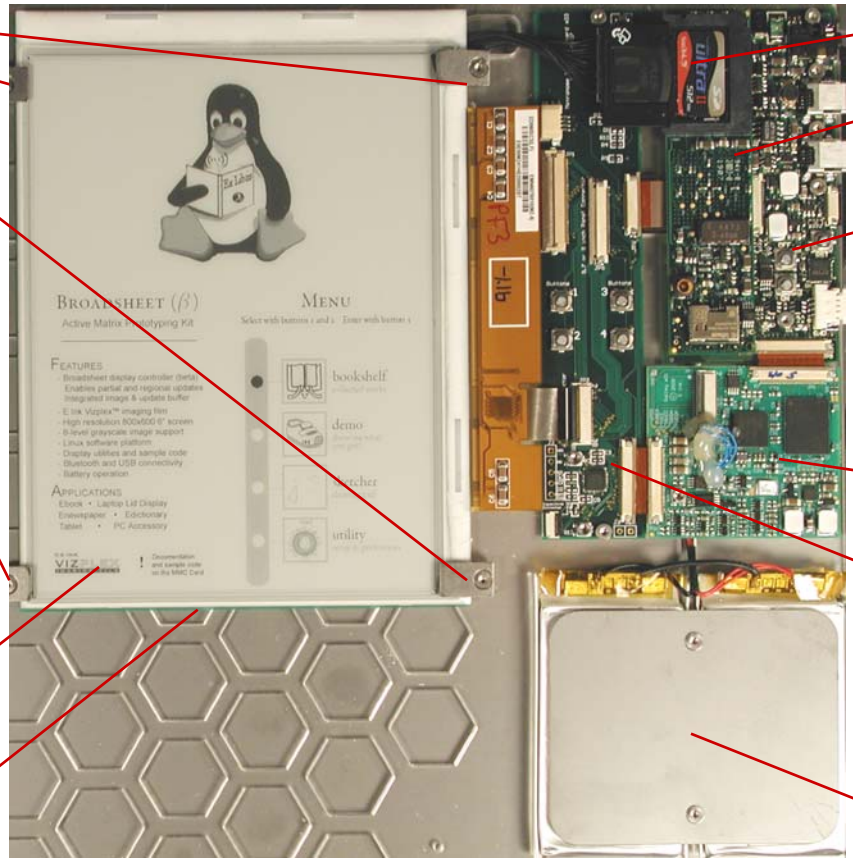

Order online at ***www.eink.com***



# AM300 Block Diagram



# Layout and Components



# Host and Interface

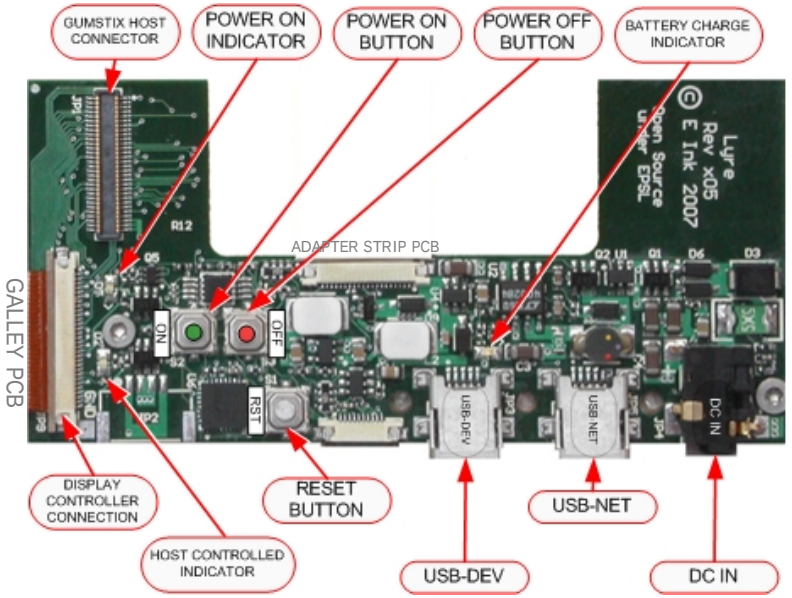
The Gumstix host is a single board computer running Linux on a Marvel PXA255 processor.

www.gumstix.org


IMPORTANT NOTE:
AM300 User's Guide is located on the MMC/USB card.

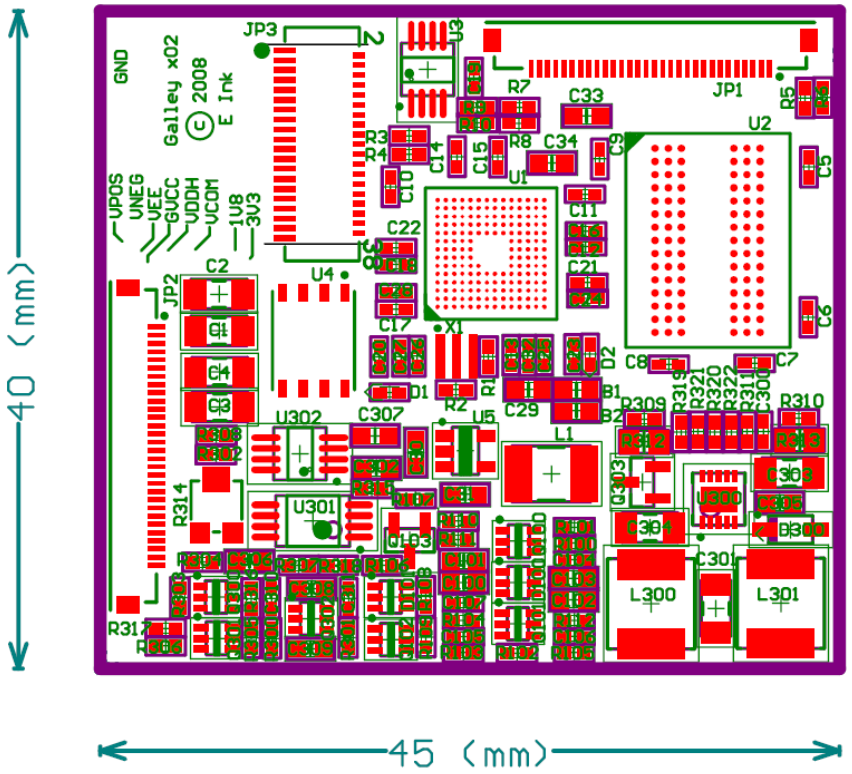
1) Fold MMC card to expose USB connector.
2) Insert USB connector into PC.
3) Go to ...\Documentation\AM300_USERS_GUIDE.pdf


The Lyre PCB is an I/O board between the Gumstix and Galley. This PCB has the AM300, on, off and reset buttons.




Developers can access the Gumstix host via the USB-DEV connection on the Lyre PCB.

# Galley PCB



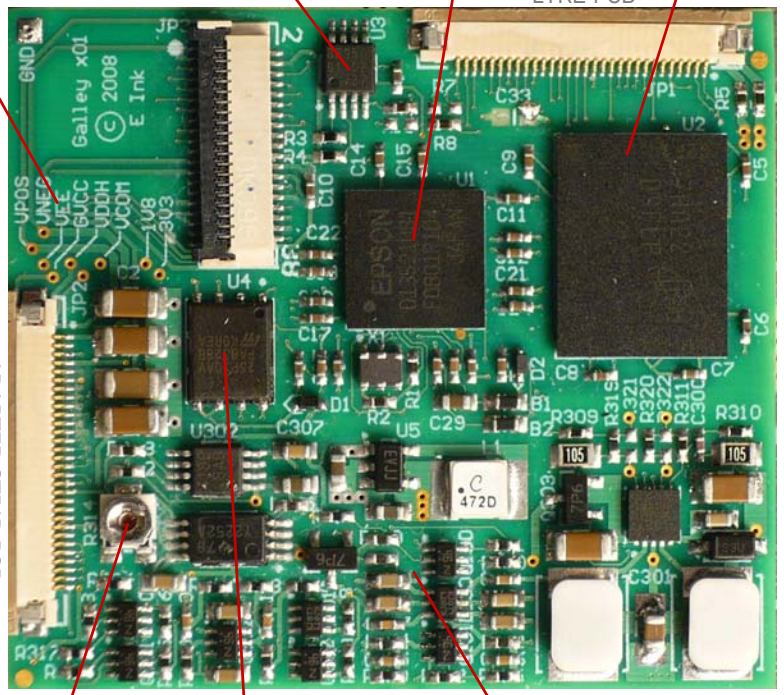

Test points

Temp sensor

S1D13521

SDRAM

LYRE PCB



ADAPTER STRIB PCB

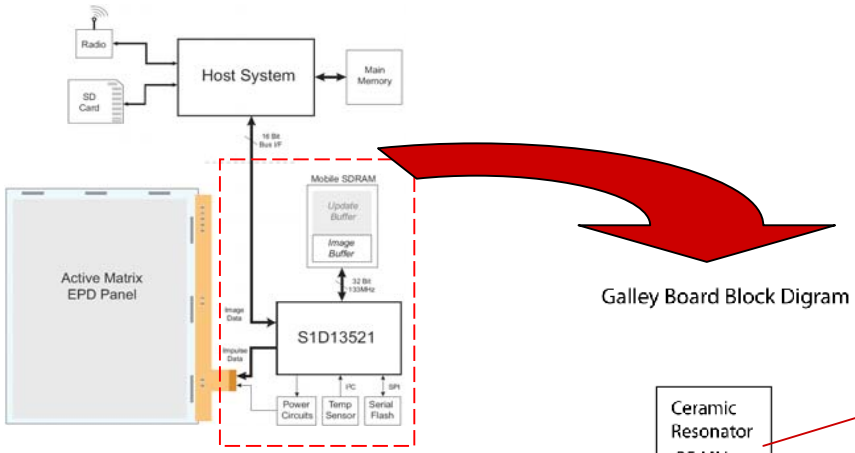
R314 VCOM Potentiometer

Flash memory

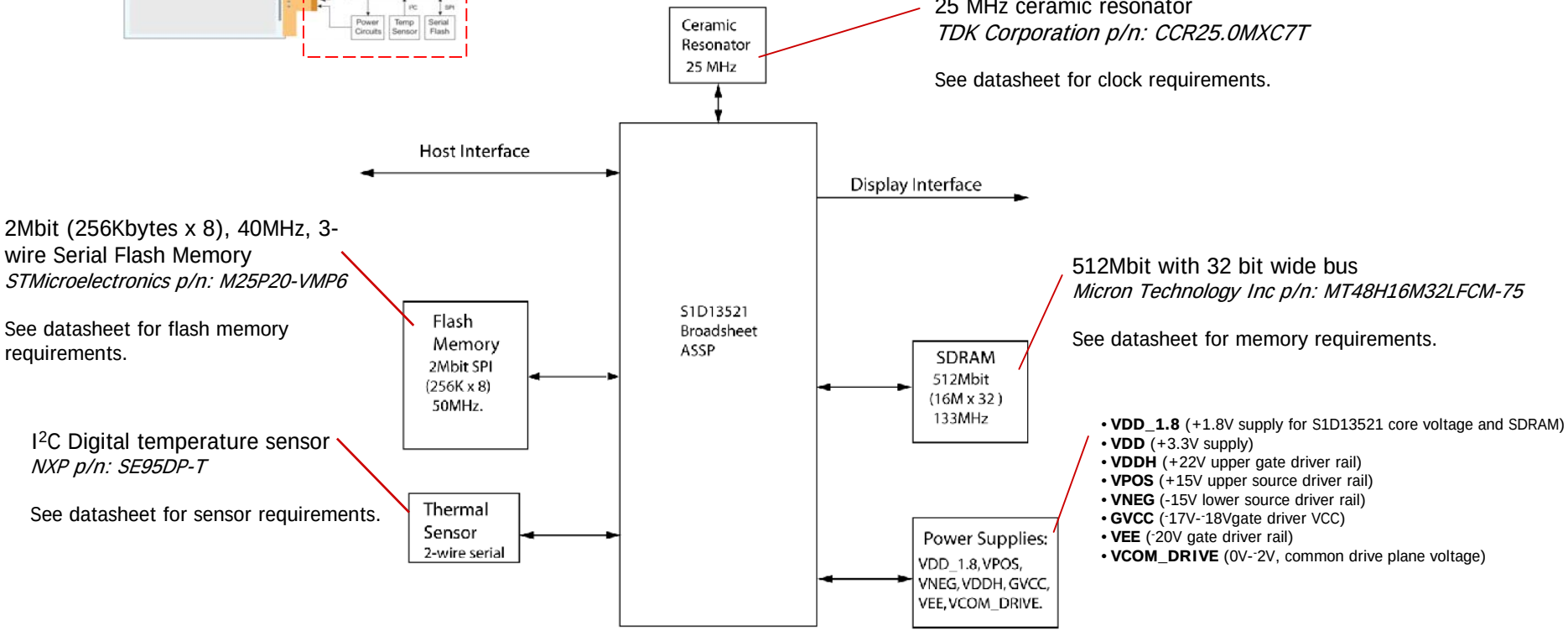
Power circuitry

VDD_1.8, VDD, VDDH, VPOS, VNEG, GVCC, VEE, and VCOM_DRIVE

# Galley Details




Galley Board Block Digram



# Determining SDRAM Size

***Determining the correct SDRAM size based on application:***

The following calculation should be used to determine the correct SDRAM size needed for your design independent of the waveform bit depth:

There are two parts to the SDRAM that you have to consider for S1D13521:

The **update buffer** which usually inhabits the lowest address space in SDRAM, and
The **image buffer**, which can be anywhere else in SDRAM.

The update buffer uses two bytes for each pixel, and the image buffer uses one byte for each pixel. Which gives a total of 3 bytes per pixel.

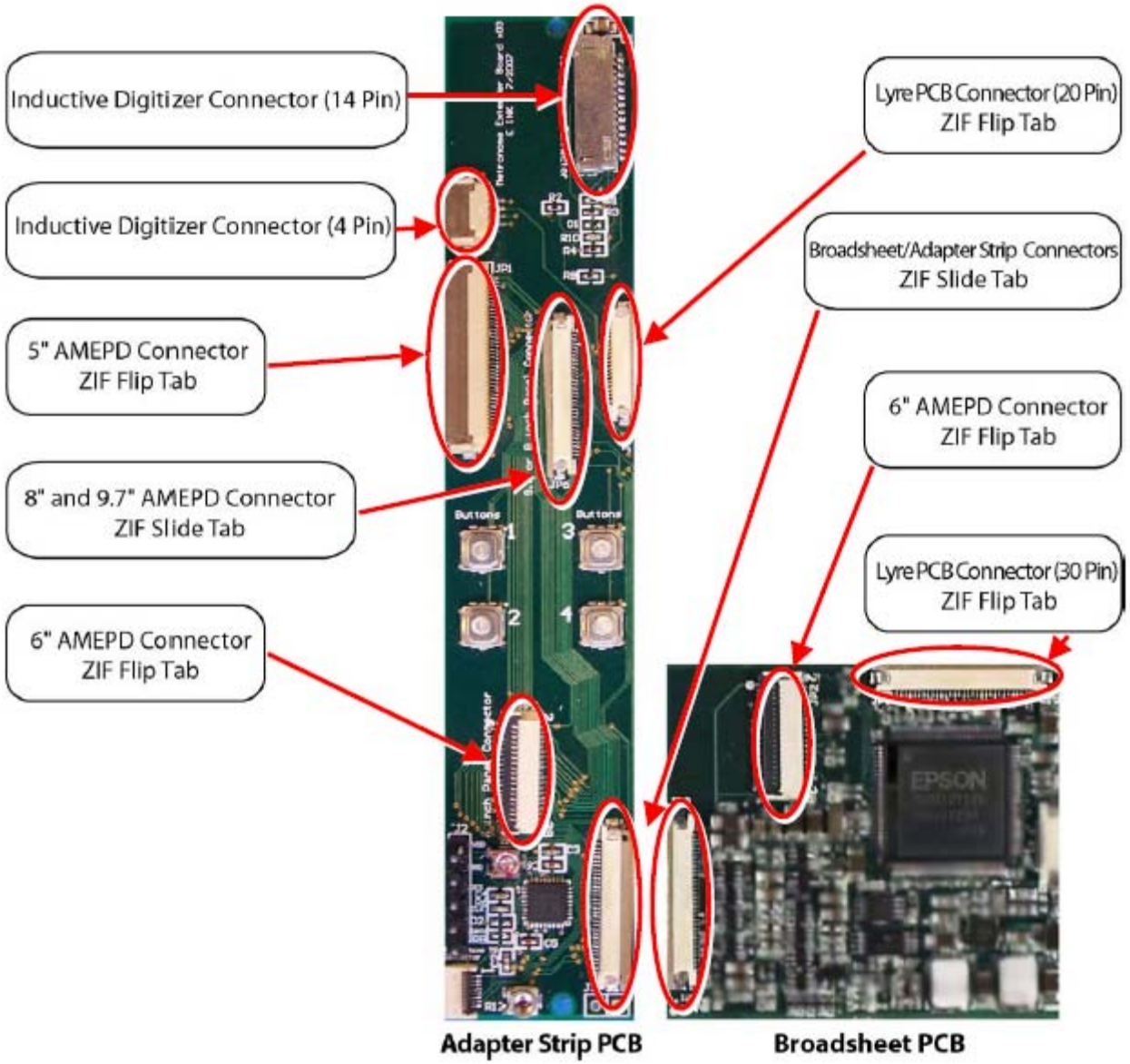
*Minimum SDRAM size needed* = **PIXELS** x **LINES** x **3 BYTES**

*Example:*

What is the minimum SDRAM requirement for a 6", 800x600 AMEPD?

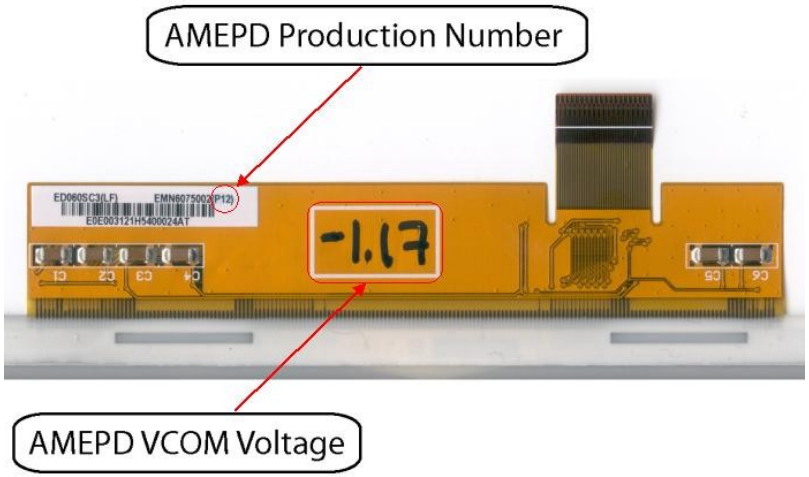
Minimum SDRAM size needed = 800 x 600 x 3 bytes = 1,440,000 bytes
= 1.44Mbytes
= 11.52Mbits

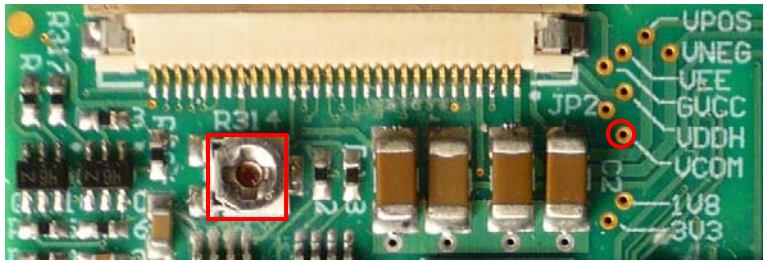
# Adaptor Strip PCB



# Identifying Panels and Waveforms

- Changing panels requires retuning VCOM, and updating the waveform.




V110_Paaa_60_WDxxyy_BTC.wbf

- Temperature compensated
- Controller: B = Epson S1D13521
- WF subversion
- WF version
- WF type
- Display Size
- FPL lot number
- Run type
- FPL Platform

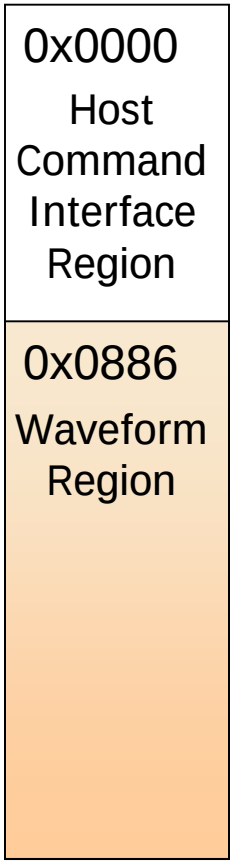
•AMEPD Production number

There are two waveform types used on Broadsheet.

- WD: 3-Bit (8 graytone) Waveform
- **WE: 4-Bit (16 graytone) Waveform**

# Programming Waveform

Flash Memory Address map



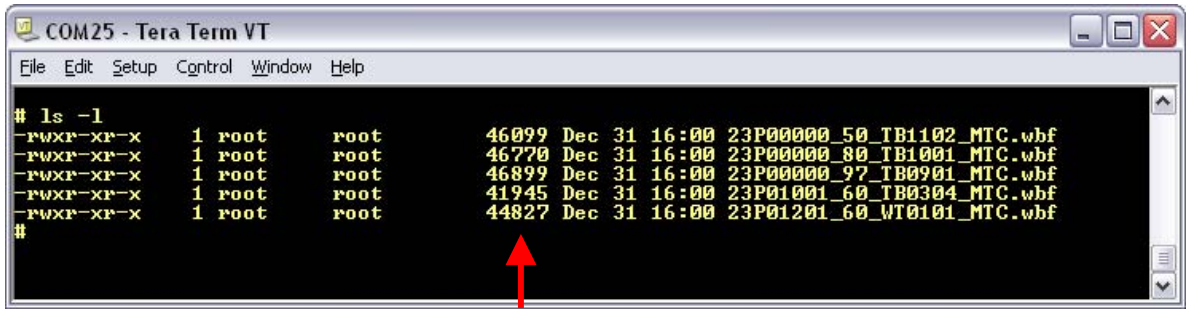

## Broadsheet requires the flash to contain both Host Command Interface and Waveform data.

To update the waveform file on the flash memory, use the bs_sfm command

**bs_sfmrw_rm_m25P20 <read/write> <address> <byte_count> <bin_file>**

*<read/write>* - read or writes to serial flash interface
*<address>* - location in flash memory where to write the waveform file.
*<byte_count>* - waveform file byte size.
*<bin_file>* - name of the waveform file.

This example writes the waveform file to address 0x886 on flash memory.
**bs_sfmrw_rm_m25P20 write 0x886 44827 /mnt/mmc/root/wfms/23P01201_60_WT0101_MTC.wbf**




**To determine the waveform's byte count:**

In the terminal window, go to the the …/mnt/mmc/root/wfms directory and type "ls –l" . The following similar data displays.

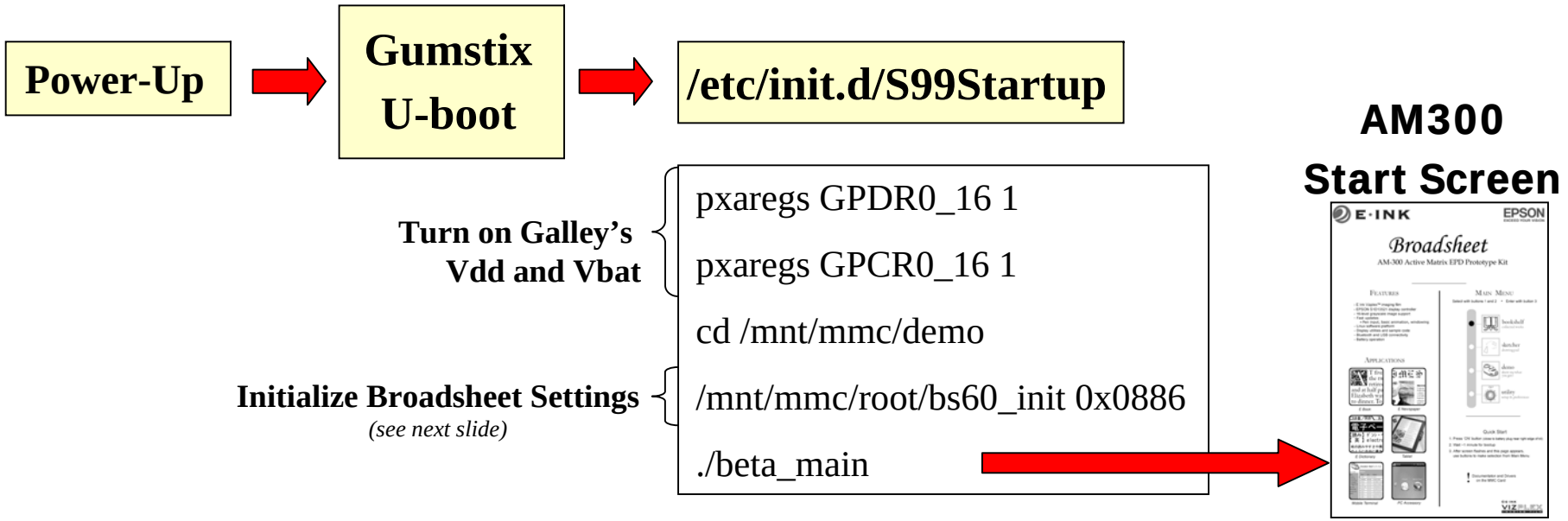
The byte_count value displays in the 5th column.

**NOTE:**
If SFM=M25P10 then use **bs_sfmrw_rm_m25P10**
If SFM=M25P20 then use **bs_sfmrw_rm_m25P20**

# Host Interface Command

- Located in serial flash memory at 0x000
- Back-up of HIC binary is on the MMC card:
  **/mnt/mmc/root/wfms/23P01201_60_WT0107_MTC_withcmdifV0007_wfmAdr0x886.wbf**
- Follow the same process as waveform programming, but change the start address location.
- This example writes the HIC file to address 0x000 on flash memory.

**bs_sfmrw_rm_m25P20 write 0x0000 49183 /mnt/mmc/root/wfms/ 23P01201_60_WT0107_MTC_withcmdifV0007_wfmAdr0x886.wbf**

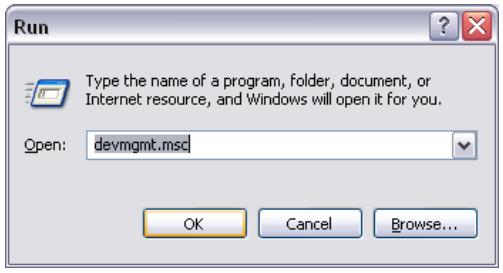


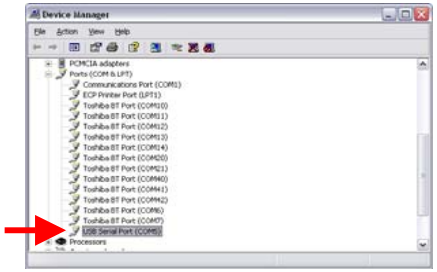
# Connecting to a PC to the AM300

- Connect USB-DEV to USB port on PC with included USB cable.
- Press the Lyre PCB's "ON" button.
- Follow the PC's prompts to automatically detect and install new hardware.

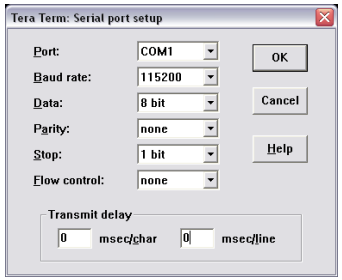


- Determine which serial port the USB connection uses.



- Connect using Terminal Emulator (TeraTerm or HyperTerminal)



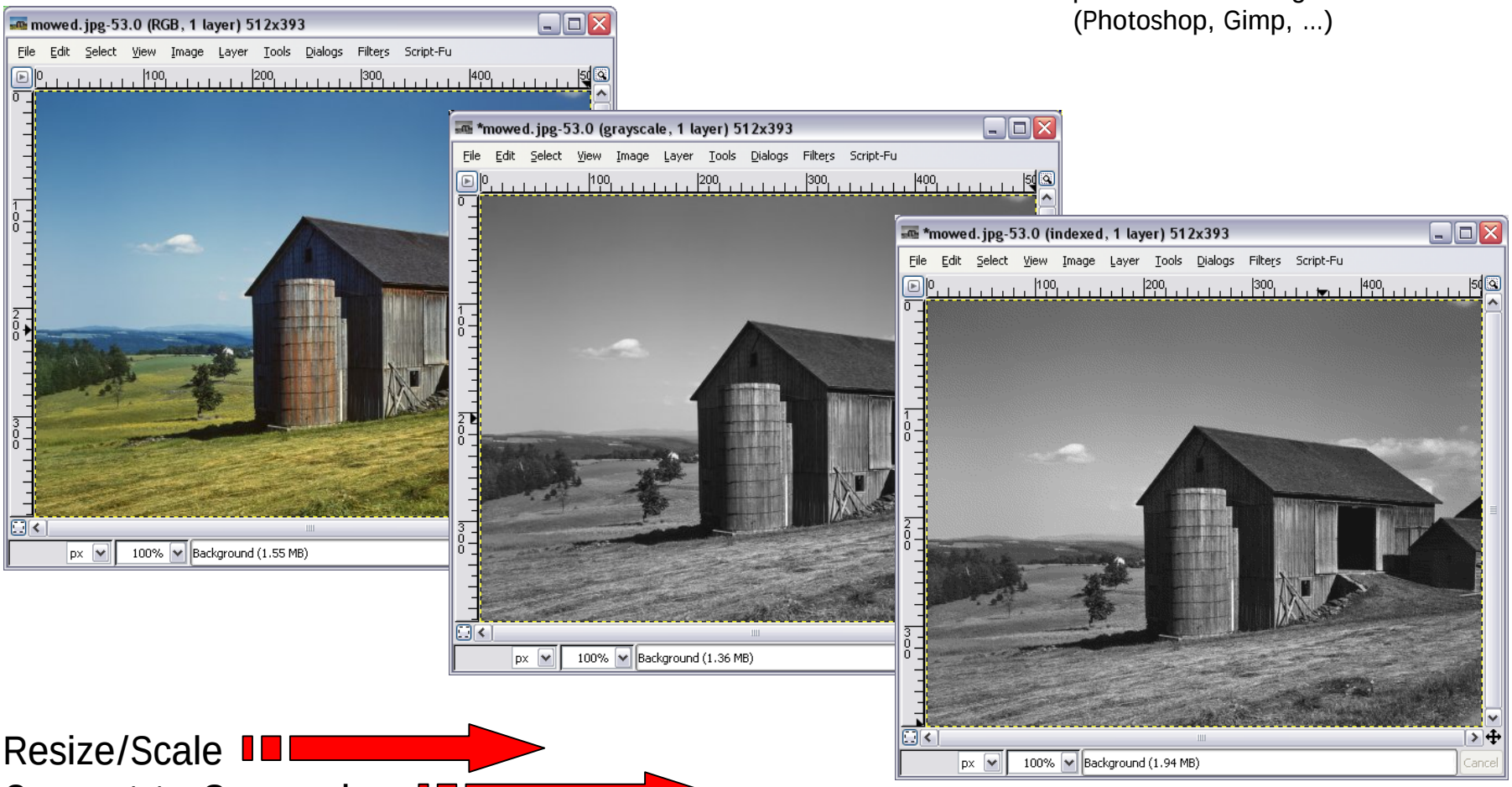

```
Login: root

Password: gumstix
```

# Formatting Images for Display.cpp

Source Image File

Most graphic editing programs can perform these image conversions. (Photoshop, Gimp, ...)



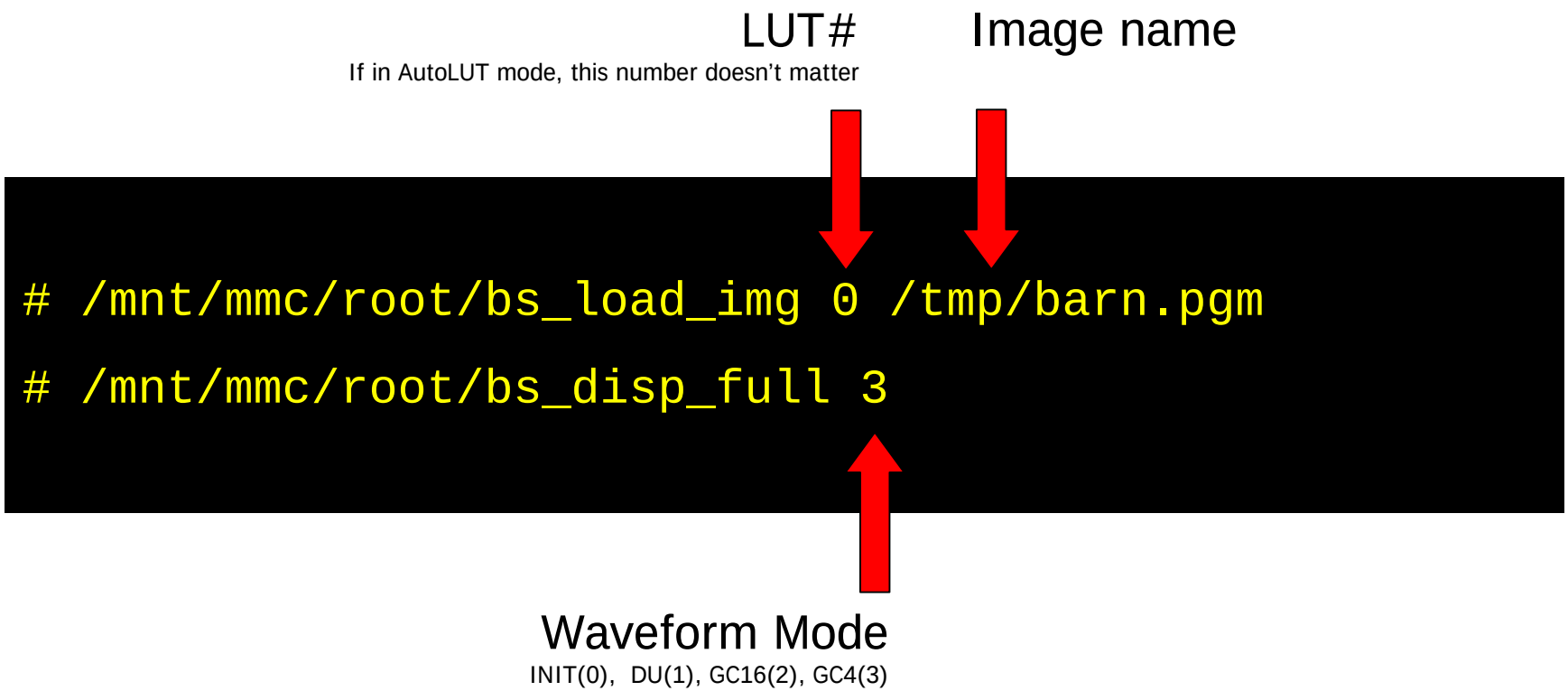

Resize/Scale

Convert to Grayscale

Index to desired grayscale levels

PPM or PGM Image File

# Displaying an Image using the Command Line



# Broadsheet Waveform

- *WD Waveform* Specification
  *3-bit Broadsheet Waveform using Vizplex 110*

- *WE Waveform* Specification
  *4-bit Broadsheet Waveform using Vizplex 110*

- WD and WE Specs are still preliminary
  *Locked specification is scheduled for late April*

E Ink currently supplies the WD 3-bit waveform
In May we will switch over to the WE 4-bit waveform

# Waveform Suite

The *.wbf waveform file contains the following WF suite

- **INIT** (Global update WF) **AVAILABLE IN BOTH WD *AND* WE**
  ***Initialize*** is used to completely clear the display, if it's left in an unknown state (i.e. if the previous image has been lost by a re-boot)

- **DU** (Local update WF) **AVAILABLE IN BOTH WD *AND* WE**
  ***Direct update*** Non-flashing waveform that can be used to update. It can update any changed graytone pixel to black or white. This waveform can be used for pen or other fast menu updates.

- **GU** (Local update WF) **ONLY AVAILABLE IN WD**
  ***Grayscale update*** Non-flashing waveform that can be used to update. It only updates changed pixels.

- **GC** (Global update WF) **AVAILABLE IN BOTH WD *AND* WE**
  ***Grayscale clear*** is the "flashy" waveform that is used by default in the kit and gives the best update performance. All the pixels are updated or cleared.

All waveforms are tuned to a 50Hz (20ms) frame rate

# Local / Global Update Tradeoff

- Local Updates (Broadsheet *Partial* Update)
  - ☺ Updates only changing pixels (Less screen flashing)
  - ☹ More Ghosting

- Global Updates (Broadsheet *Full* Update)
  - ☹ Updates all pixels (More screen flashing)
  - ☺ Less Ghosting

# Standard Waveform Modes

## **WD** (3-bit) typical performance

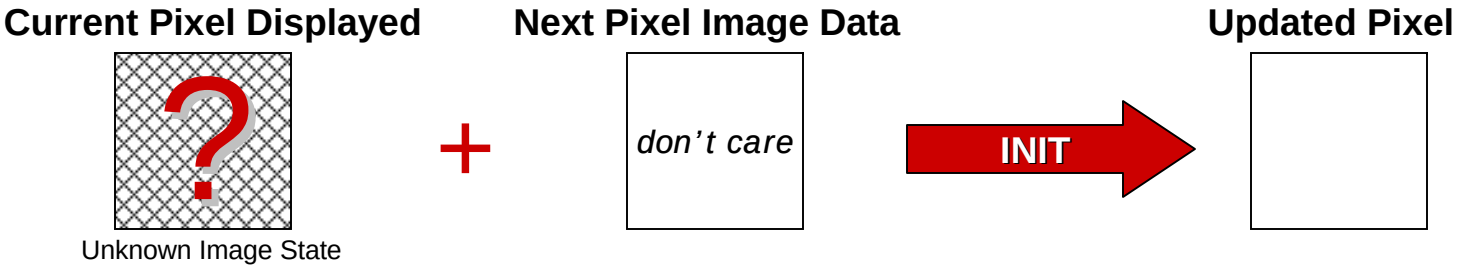
| Mode | Mode | Type | Gray Levels | Transition Appearance | Ghosting | Usage | Target Update time at 25 C (ms) |
|---|---|---|---|---|---|---|---|
| INIT | 0 | Global | White only | High Flash | N/A | Display initialization | ~4000 |
| DU | 1 | Local | 8 → Black and White | No Flash | Low | Monochrome menu, text input, and touch screen/pen input | 260 |
| GU | 2 | Local | 8 | Low Flash | Medium | High quality images and anti-aliased text | 780 |
| GC | 3 | Global[#] | 8 | High Flash | Low | High quality images and anti-aliased text | 780 |

## **WE** (4-bit) typical performance

| Mode | Mode | Supported Gray Levels transitions | Ghosting | Usage | Typical Update time at 25 C (ms) |
|---|---|---|---|---|---|
| INIT | 0 | White only | N/A | Display initialization | ~2000 |
| DU | 1 | 16 → Black/white | Low for Black and white. Medium for graytone to black or white | Pen input and cursors | 260 |
| GC16 | 2 | 16→ 16 | Low | High quality images | 800 |
| GC4 | 3 | 4→4<br>16→4 | Low | Anti-aliased text | 500 |

The released waveform specification can be communicated to end-customers who are under non-disclosure agreements with E Ink Corporation.

# Init Waveform (WE waveform)




- The INIT waveform resets the display
- *Use only* if current screen information is in an unknown state
- This waveform mode is unique because it *always* drives the display to white, after multiple black-white flashes, and not to the 'Next Screen Image Data' display information

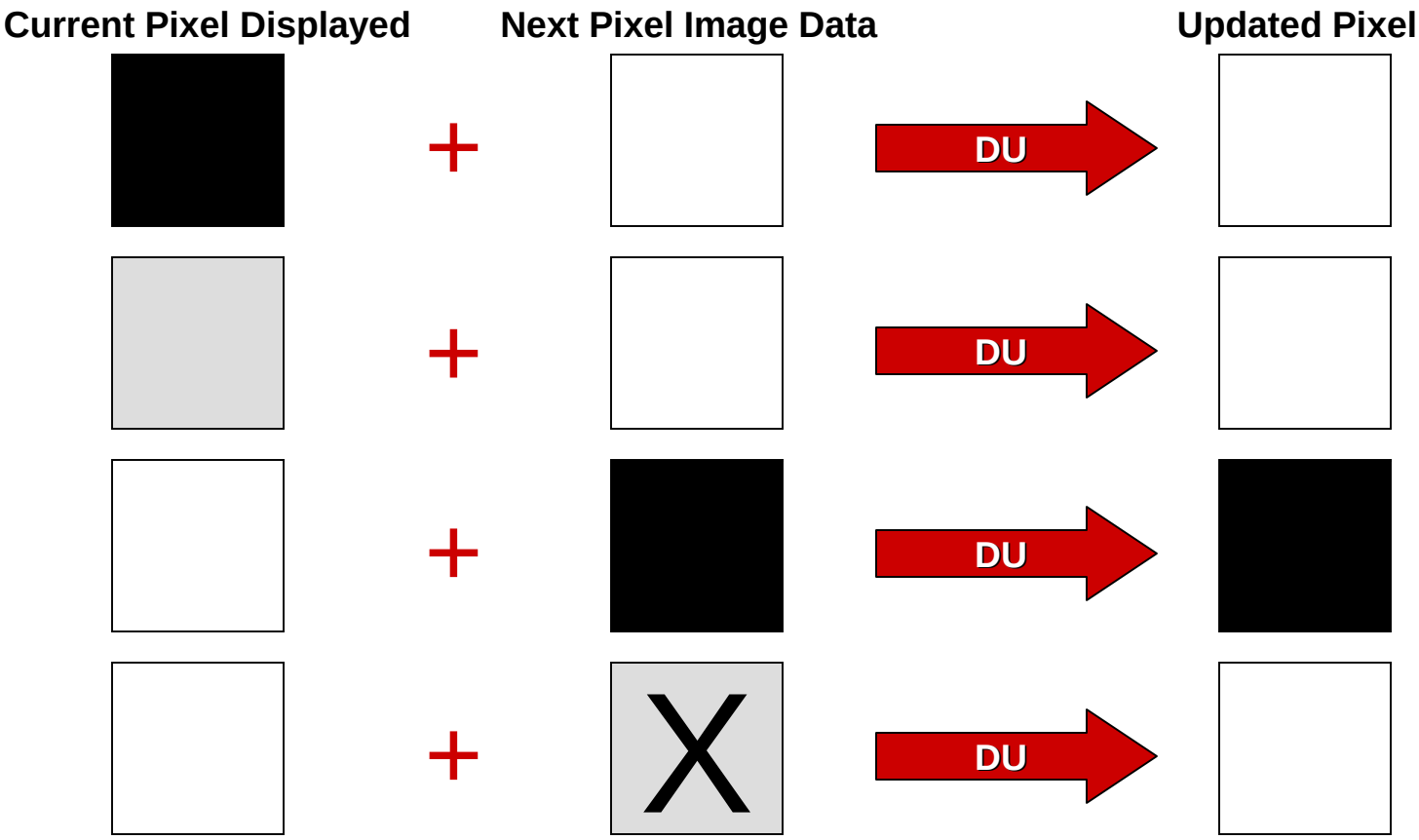
**WF Mode 0**

**Transition time: ~2000ms at 25C**

**Global Update – Flashing**

Example usage case:
*Device Power Up*

# Direct Update (WE waveform)



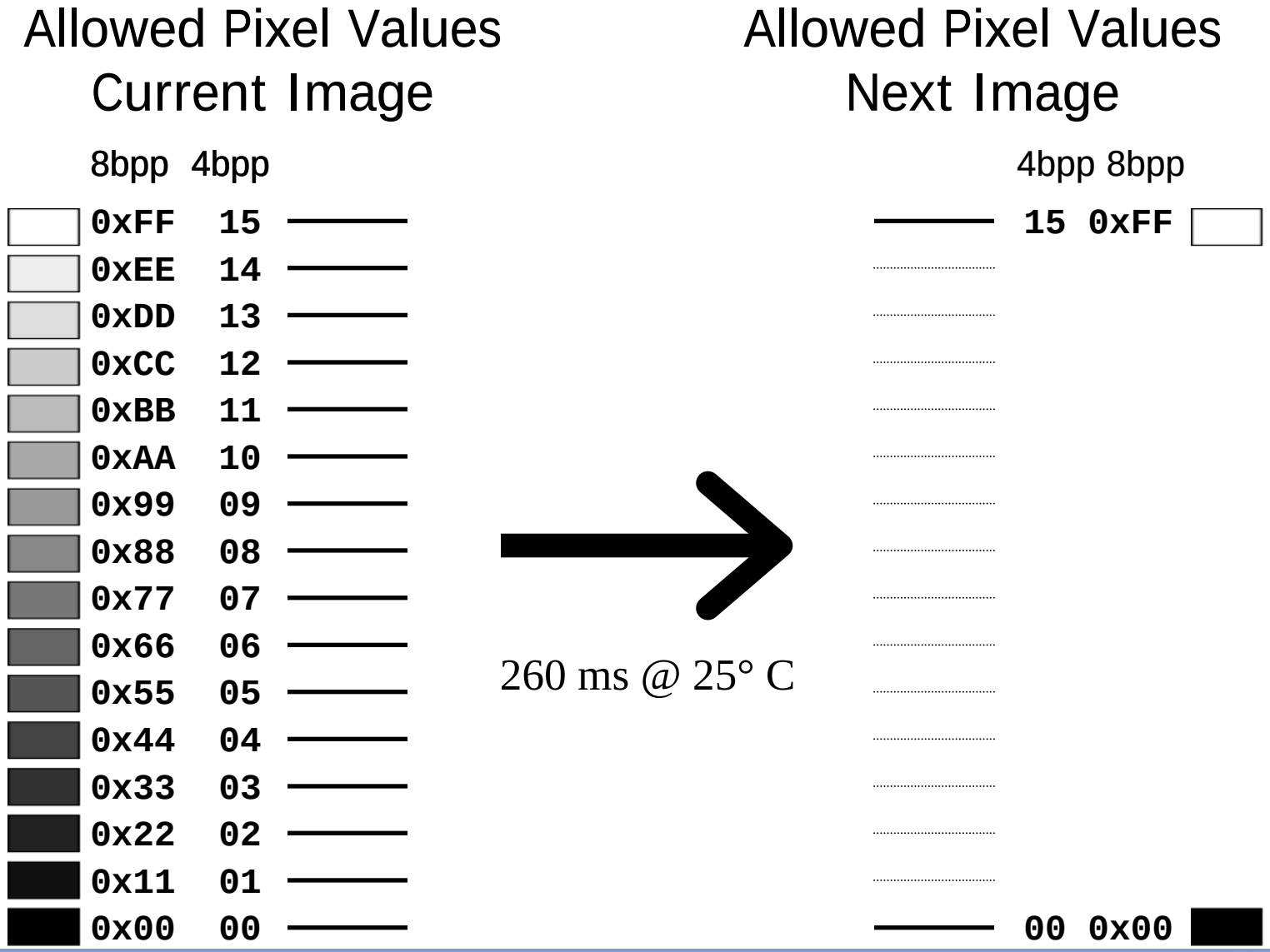

**WF Mode 1**

**Transition time: ~260ms at 25C**

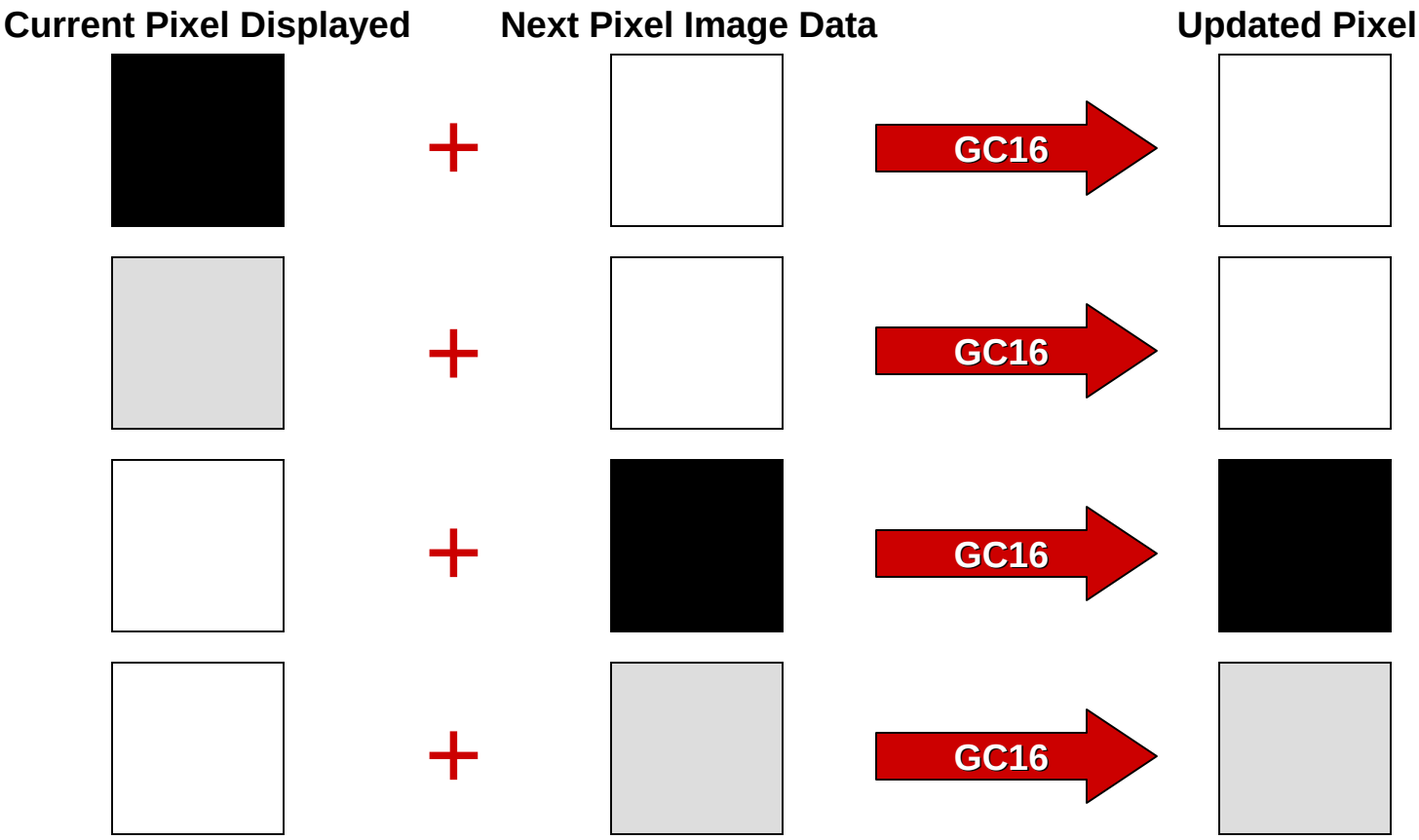
**Local Update – No Flashing**

Example usage case:
*Fast cursor movement*

# Available DU State Transitions



# Grayscale Clear 16 (WE waveform)



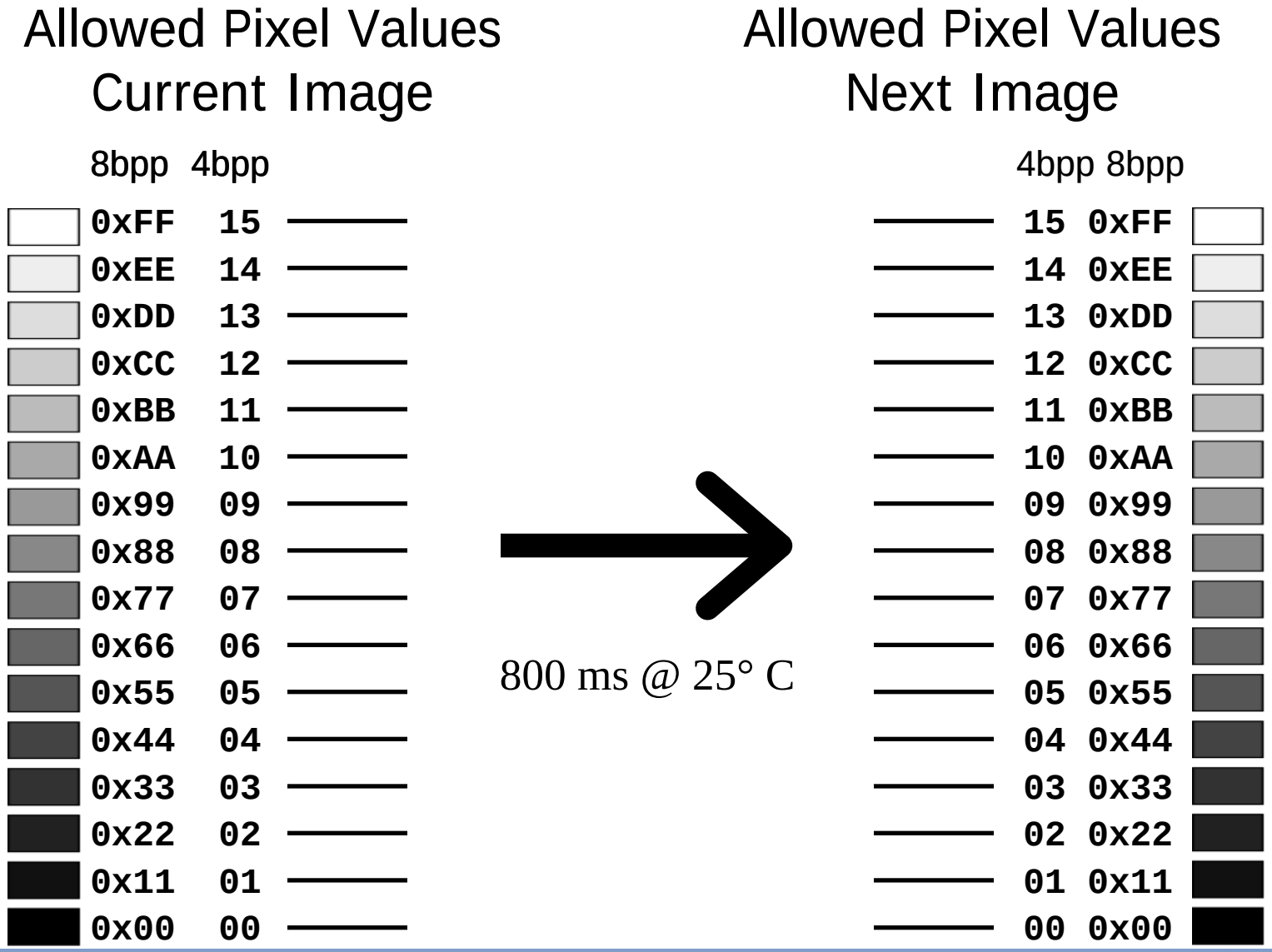

**WF Mode 2**

**Transition time: ~800ms at 25C**

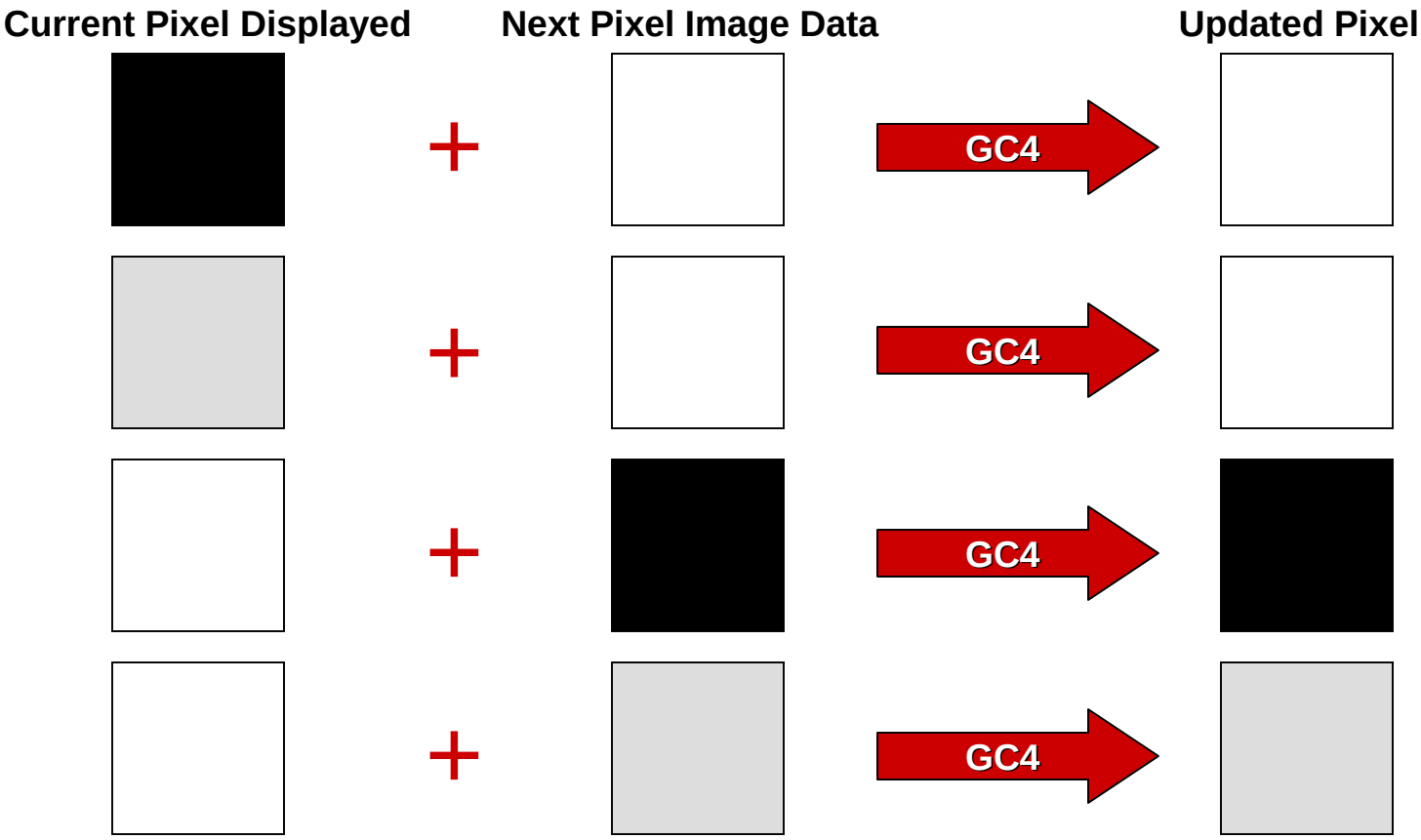
**Global Update – Flashing**

Example usage case:
*Displaying an image*

# Available GC16 State Transitions



# Grayscale Clear 4 (WE waveform)



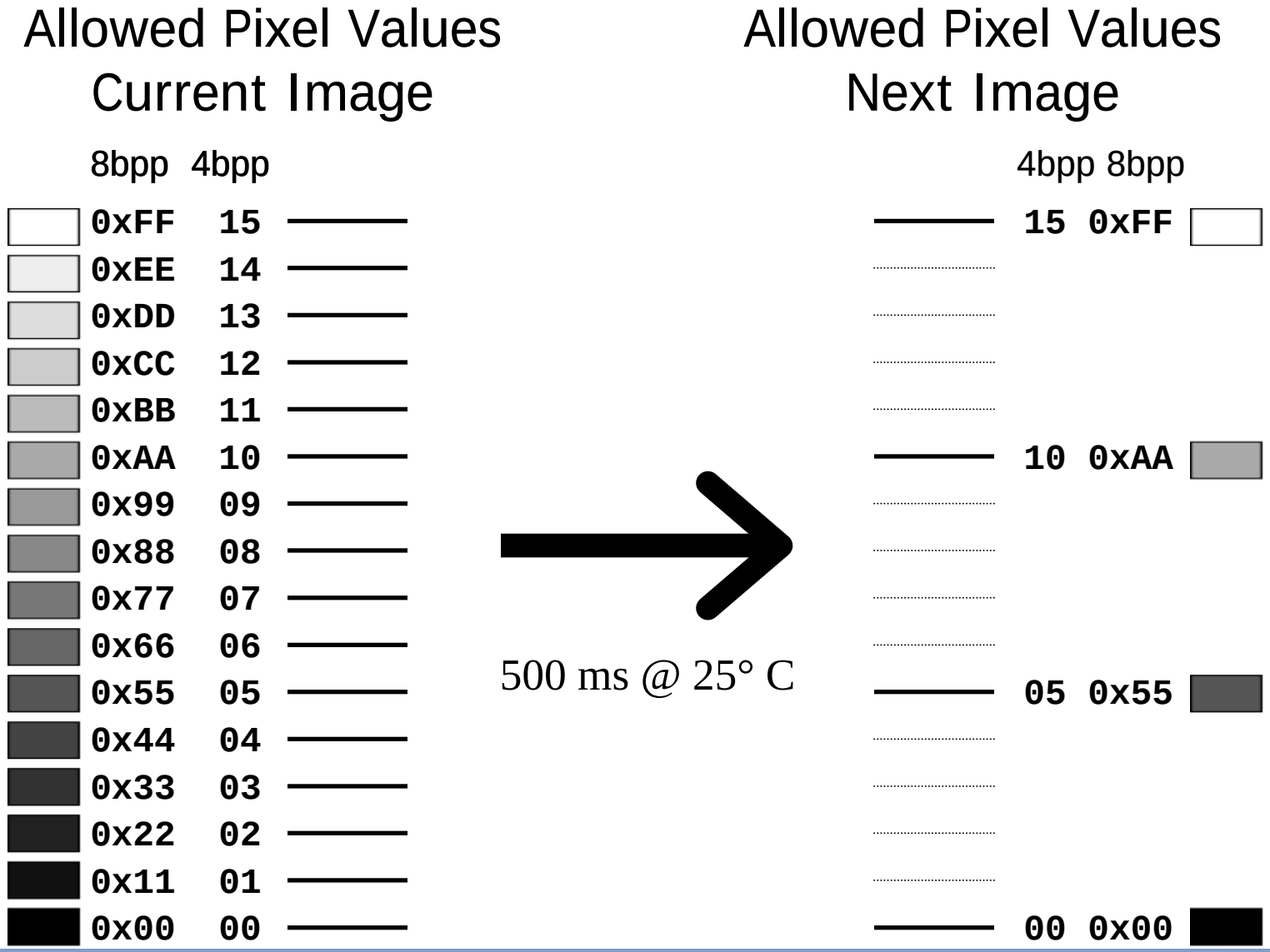

**WF Mode 3**

**Transition time: ~500ms at 25C**

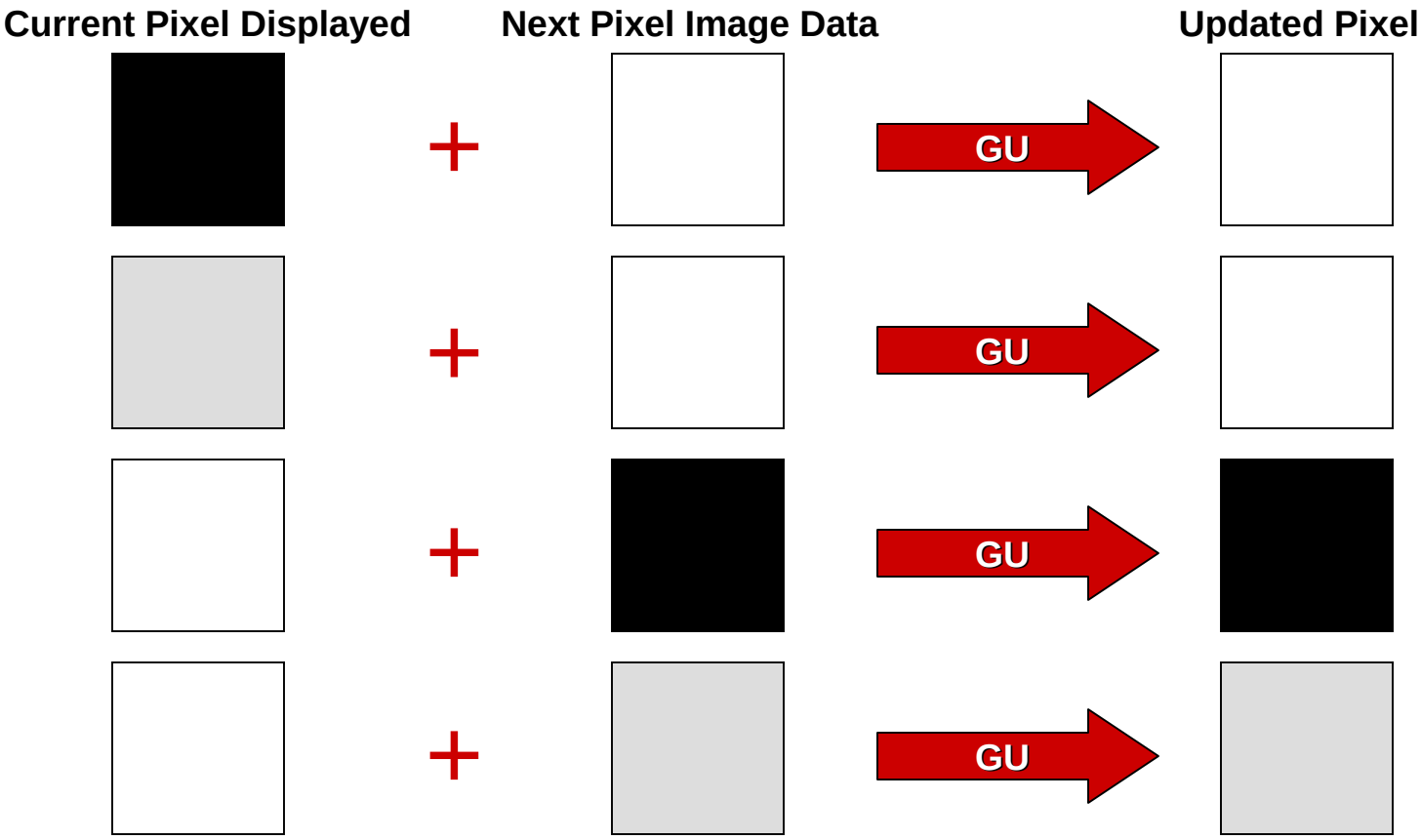
**Global Update – Flashing**

Example usage case:
*Displaying anti-aliased text*

# Available GC4 State Transitions



# Grayscale Update (WD waveform)



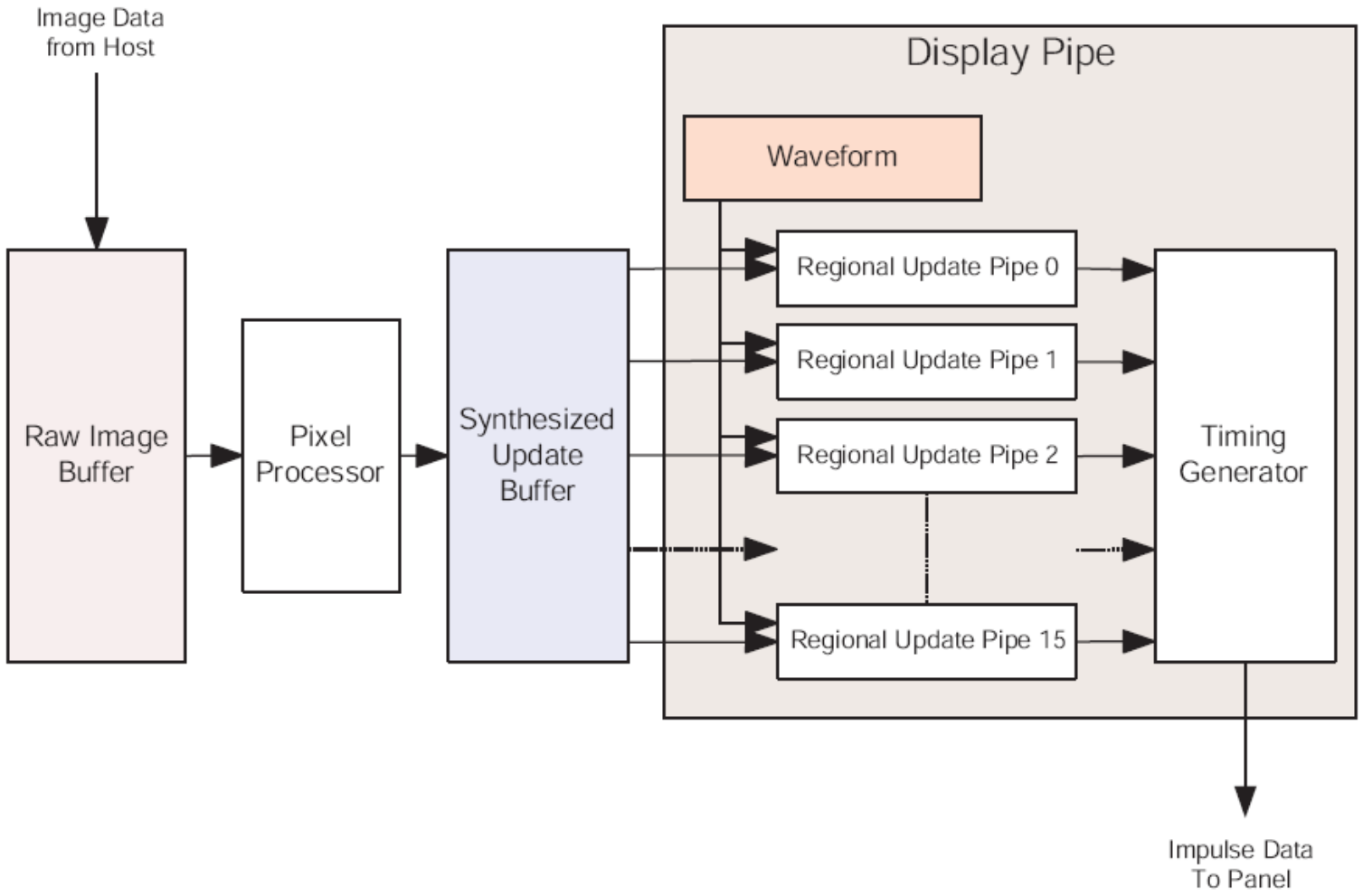

**WF Mode 2**

**Transition time: ~780ms at 25C**

**Local Update – No Flashing**

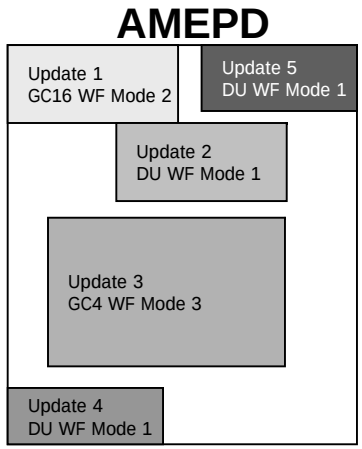
Example usage case:
*Display anti-aliased text*

# Image Data Management

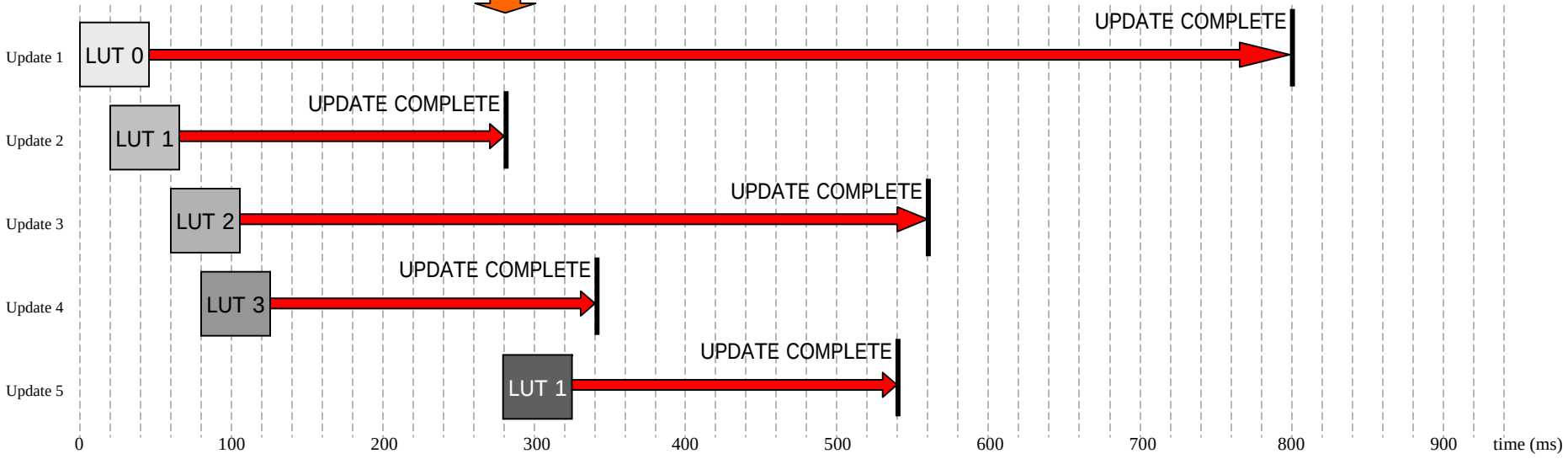


# 16 Regional Pipeline Updates

- Each update can uses its own WF mode
- One update can initiate every 20ms (based on frame clock)
- Updates cannot be interrupted until complete
  - i.e. LUT1 running DU cannot be used for 260ms




**Snapshot at 280ms**
- LUT 0, LUT 2 and LUT 4 are in use
- 13 LUTs are available
- Image data in LUT1 can be overwritten




NOTE: Example uses S1D13521 Auto-LUT mode

# Updating the Display

## Load Image Data

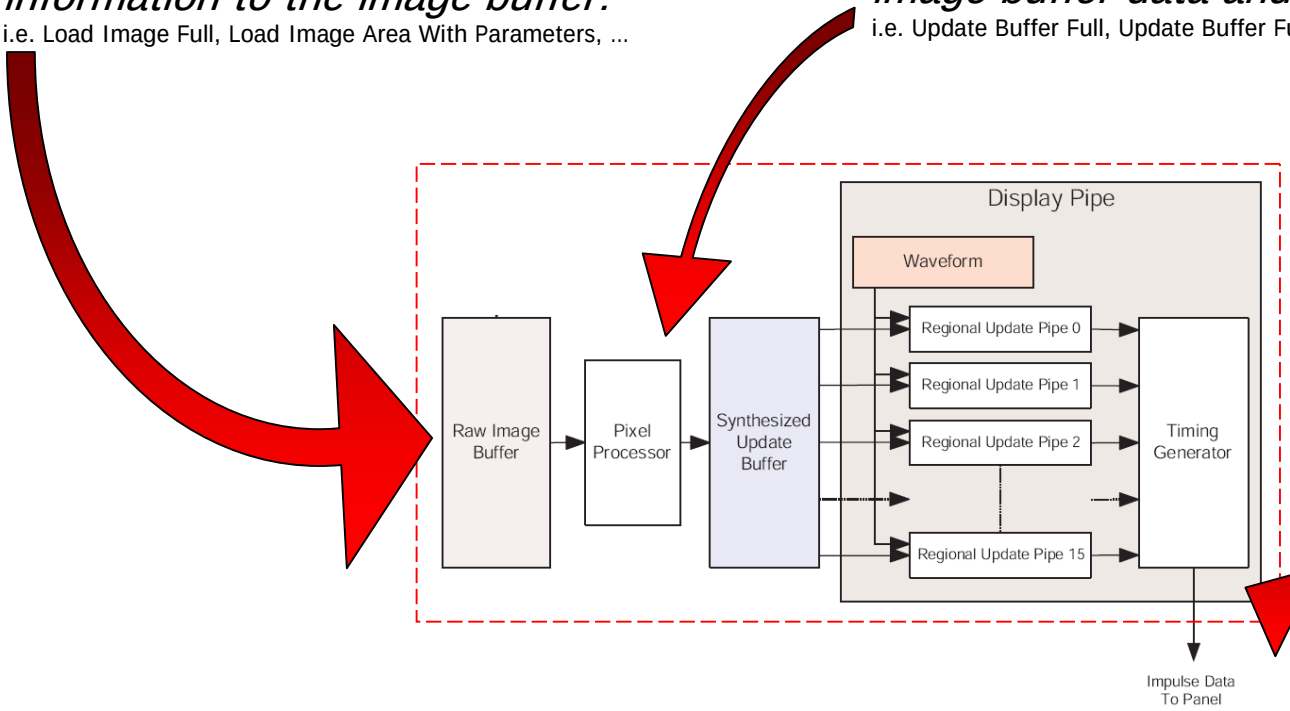
*Use image loading commands to post information to the image buffer:*

i.e. Load Image Full, Load Image Area With Parameters, ...

## Update Image Data

*Use image update commands to process image buffer data and load it on the display:*

i.e. Update Buffer Full, Update Buffer Full Area, Update Buffer Partial, ...



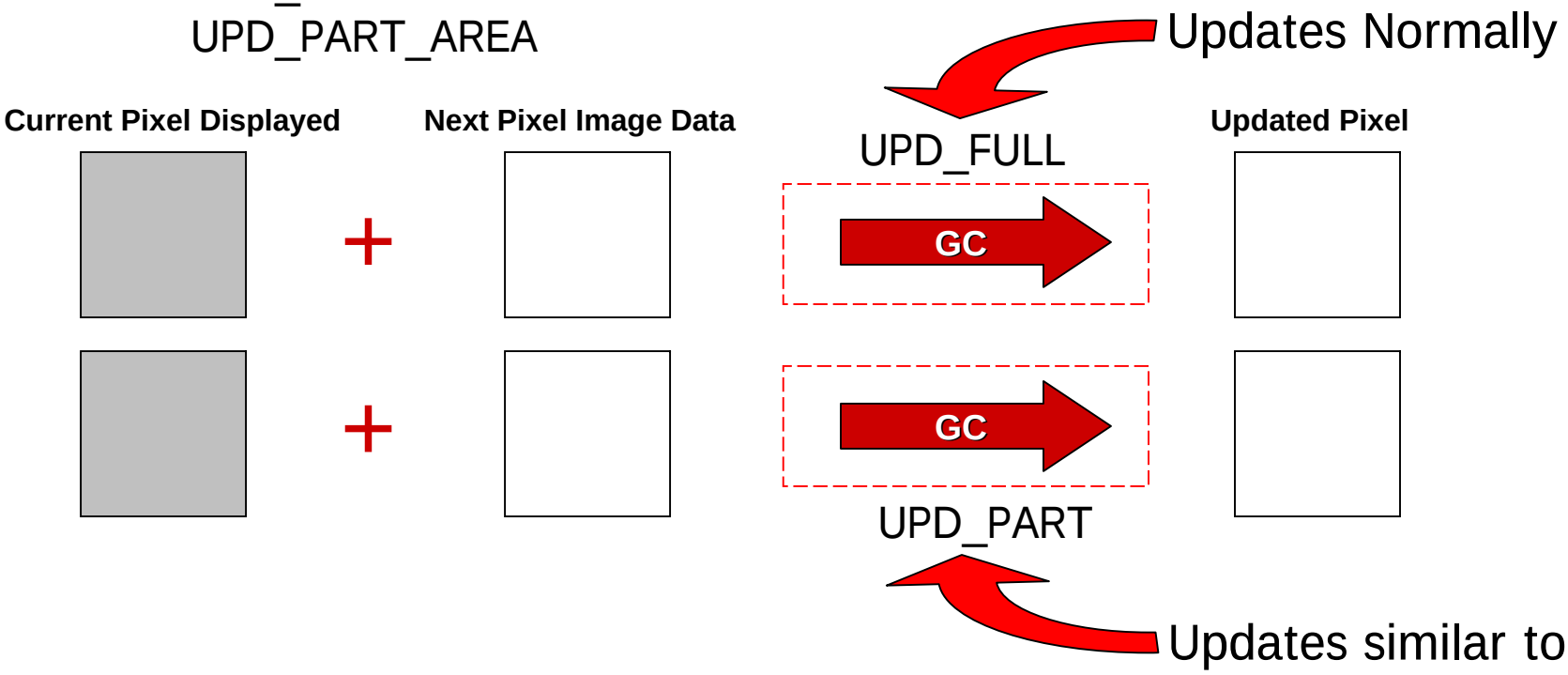

## Monitor Update Status

*Using Polling commands to determine display pipeline status:*

i.e. Wait for Display Engine At Least 1 LUT is Free,...

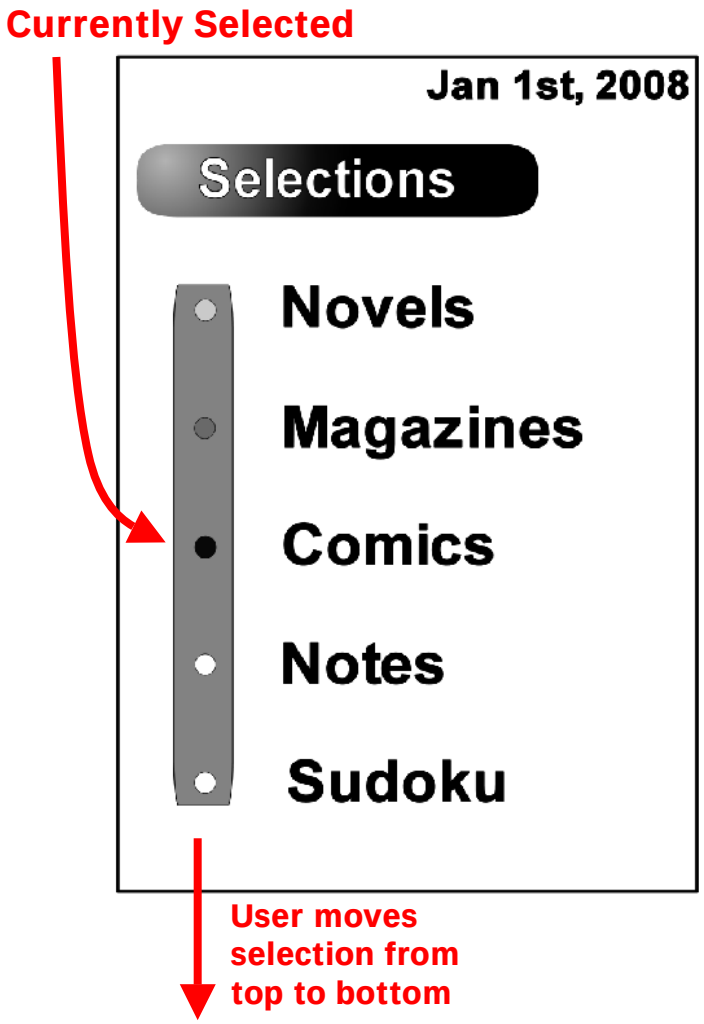
# Full and Partial Waveform Update Commands

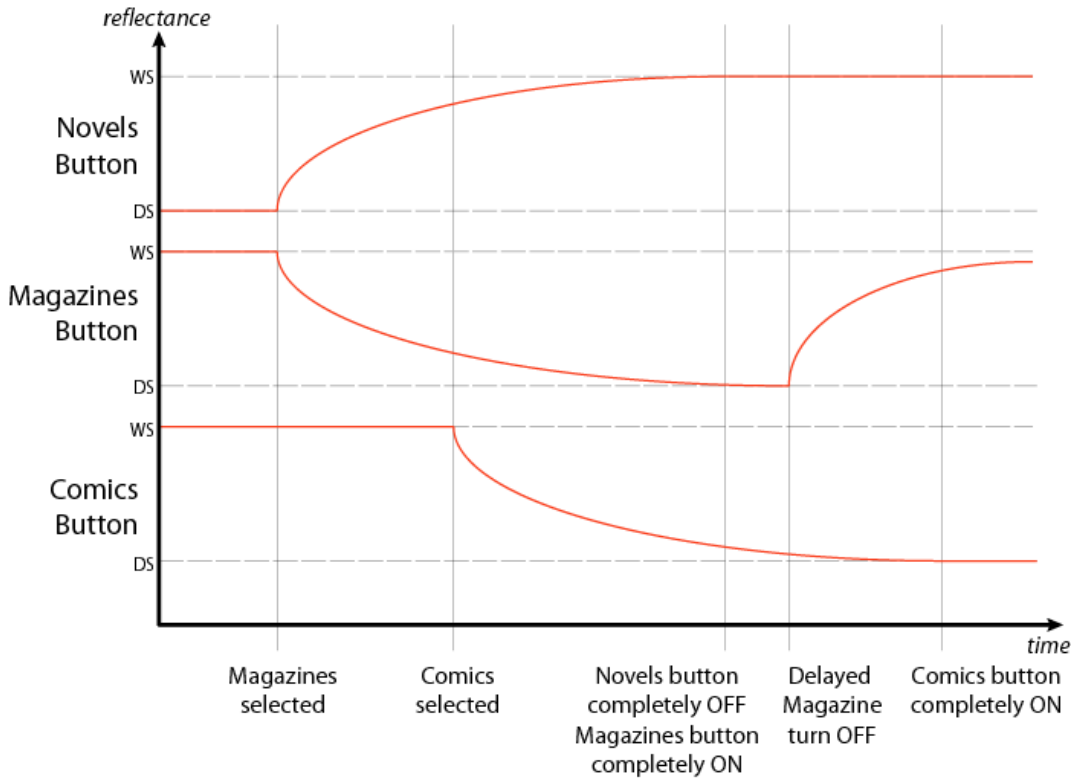
- Full update, updates the display based on selected waveform
  - UPD_FULL
  - UPD_FULL_AREA
- Part updates, updates only changing pixels using the selected waveform
  - UPD_PART
  - UPD_PART_AREA



# Waveform Human Factors

- Transitions are perceivable at ~40-60ms.



# Broadsheet AutoLUT Mode

### 10.3.17 Display Engine: Control/Trigger Registers

**REG[0330h] Update Buffer Configuration Register**
Default = 0000h — Read/Write

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Display Engine Software Reset (WO) | n/a | | | Reserved | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| LUT Auto Select Enable | Reserved | n/a | | | LUT Index Format Select bits 2-0 | | |

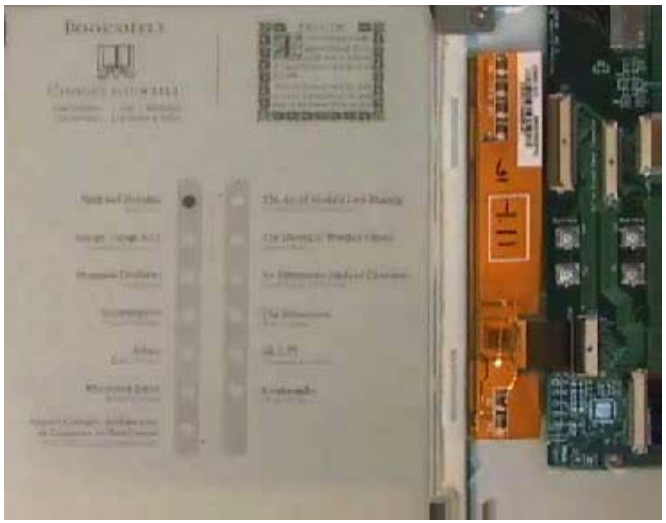
// Turn off auto-lut mode - ***Manually assign and track LUTs***

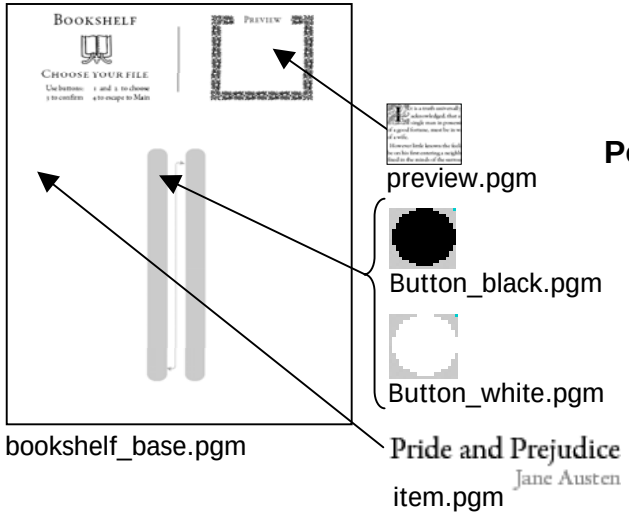
bs_cmd_wr_reg(0x0330, bs_cmd_rd_reg(0x0330) & ~(1<<7));

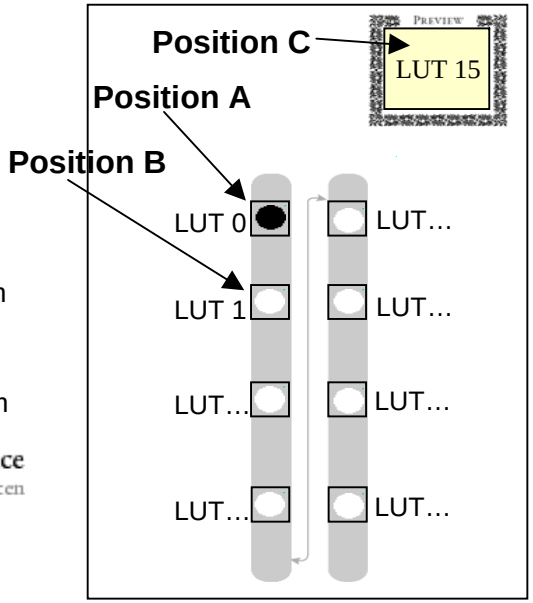
// Turn on auto-lut mode - ***Broadsheet will assign and manage LUTs***

bs_cmd_wr_reg(0x0330, bs_cmd_rd_reg(0x0330) | (1<<7));

# Menu Bar Updates






- Button Erase - Load White button data to image buffer (DU)

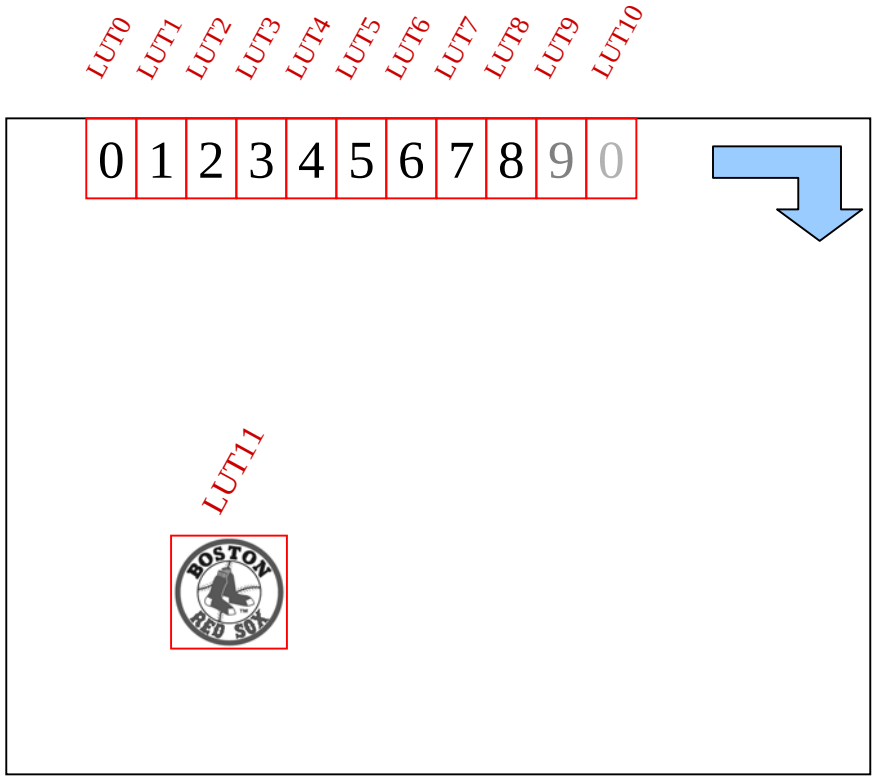
  Update to specific LUT
- Button Write - Load Black button data to image buffer (DU)

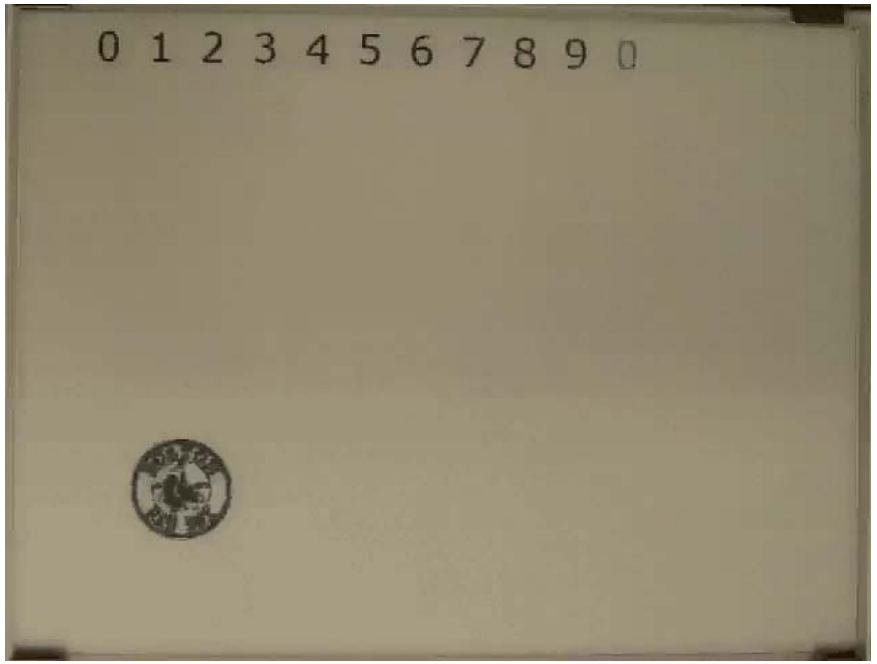
  Update to specific LUT
- Preview Write – Load Preview image data to image buffer (GC16)

  Update to specific LUT
- Check for LUT conflict using Polling commands

See *bookshelf.c* for sample code

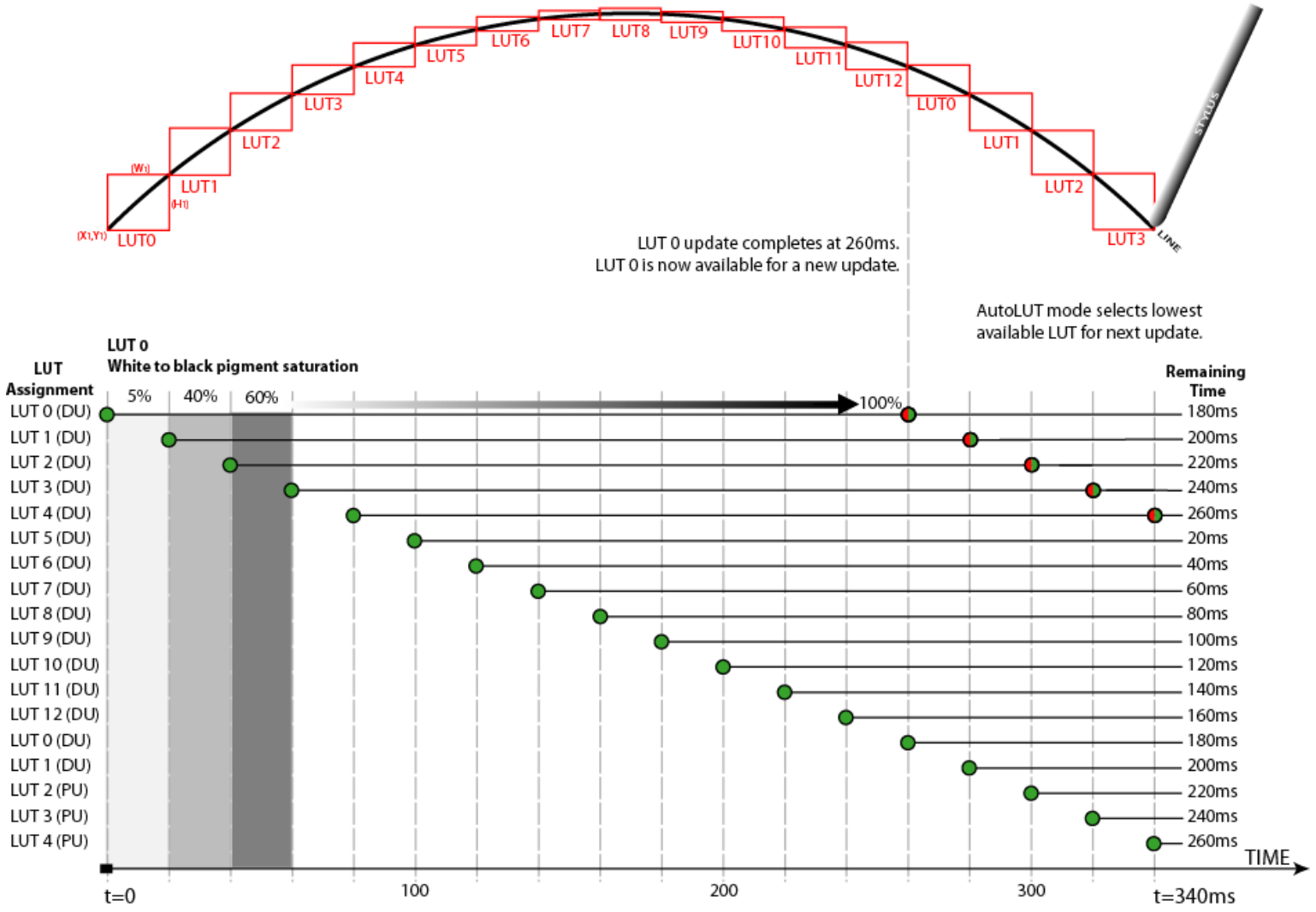
# 16 Partial Image LUT




- Numbers around perimeter use DU 260ms update mode
- Baseball logos use GC16 800ms update mode
- **What is the maximum number of LUT's this update can scheme can use?**

# Pen Input




Notes:
20ms = 1 frame
DU waveform updates require 260ms (or 13 frames)
Image transition is perceivable at ~40-60ms

# Tips to Reduce Ghosting

- Maximize use of Global and Full updates
- Add a Global update to successive Local updates
  - Ghosting accumulates over time with successive partial updates
- Load pop-ups quickly with Local update, then cleanly erase window with a Global update
- Less ghosting is visible when transitioning to white compared to gray
- Minimize use of images with sharp edges
- Vcom must be tuned to panel
- Use the proper waveform