E-paper Display COG
Driver Interface Timing
for 1.44”, 1.9”, 2”, 2.6” and 2.7” EPD with G2 COG and Aurora Mb Film

<table>
<thead>
<tr>
<th>Description</th>
<th>Detailed information to design a timing controller for 1.44”, 1.9”, 2”, 2.6” and 2.7” EPD with G2 COG and Aurora Mb film</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>2015/07/27</td>
</tr>
<tr>
<td>Doc. No.</td>
<td>4P018-00</td>
</tr>
<tr>
<td>Revision</td>
<td>03</td>
</tr>
</tbody>
</table>

Design Engineering

<table>
<thead>
<tr>
<th>Approval</th>
<th>Check</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>朝玄</td>
<td>朝玄</td>
<td>丁</td>
</tr>
<tr>
<td>2015.07.27</td>
<td>2015.07.27</td>
<td>2015.07.27</td>
</tr>
</tbody>
</table>

4F, No. 28, Chuangye Rd., Tainan Science Park, Tainan City 74144, Taiwan (R.O.C.)
Tel: +886-6-279-5399      Fax: +886-6-505-5300
Copyright

Pervasive Displays Incorporated All rights reserved.

This document is the exclusive property of Pervasive Displays Inc. (PDI) and shall not be reproduced or copied or transformed to any other format without prior permission of PDI. (PDI Confidential)

本資料為龍亭新技股份有限公司專有之財產，非經許可，不得複製、翻印或轉變成其他形式使用。

龍亭新技股份有限公司 Pervasive Displays Inc.

4F, No. 28, Chuangye Rd., Tainan Science Park, Tainan City 74144, Taiwan (R.O.C.)

Tel: +886-6-279-5399

http://www.pervasivedisplays.com
Table of Contents

1. General Description ........................................................................................................ 6
   1.1 Overview .................................................................................................................. 6
   1.2 Input Terminal Pin Assignment ............................................................................... 7
   1.3 Reference Circuit ...................................................................................................... 9
      1.3.1 1.44 inch EPD Reference Circuit ................................................................. 9
      1.3.2 2 inch EPD Reference Circuit ........................................................................... 10
      1.3.3 2.7 inch EPD Reference Circuit ....................................................................... 11
      1.3.4 1.9 inch EPD Reference Circuit ....................................................................... 12
      1.3.5 2.6 inch EPD Reference Circuit ....................................................................... 13
   1.4 EPD Driving Flow Chart ........................................................................................... 14
   1.5 Controller .................................................................................................................. 15
   1.6 SPI Timing Format .................................................................................................... 16
2. Write to the Memory ........................................................................................................... 21
3. Power On G2 COG Driver ............................................................................................... 22
4. Initialize G2 COG Driver .................................................................................................. 23
5. Write Data from the Memory to the EPD ......................................................................... 25
   5.1 Data Structure .......................................................................................................... 25
   5.2 Overall Update Flow ................................................................................................. 26
   5.3 Store a Line of Data in the Buffer ............................................................................ 26
   5.4 Writing to the Display in Stages ................................................................................. 32
6. Power Off G2 COG Driver ............................................................................................... 34
Appendix: Use G1’s PCBA to drive the EPD with G2 Driver IC ............................................... 35
## I. Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Page (New)</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver. 01</td>
<td>2014/11/20</td>
<td>All</td>
<td>All</td>
<td>First issued</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1.2</td>
<td>Modify the description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>1.3.3</td>
<td>Modify the pin description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>1.3.6</td>
<td>Modify the description of Note number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>1.3.6</td>
<td>Add description about hardware suggestion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>5.2</td>
<td>Add the description of “Turn on OE”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
<td>6</td>
<td>Modify the description BORDER -&gt; /BORDER_CONTROL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>4</td>
<td>Modify OE setting (0x02,0x40) -&gt; (0x02,0x04)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
<td>6</td>
<td>Remove OE off flow (0x02,0x05)</td>
</tr>
<tr>
<td>Ver. 02</td>
<td>2015/02/16</td>
<td>All</td>
<td>All</td>
<td>Modify PDi company address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All</td>
<td>All</td>
<td>Revise some descriptions in document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>1.2</td>
<td>Modify Pin.8 assignment</td>
</tr>
</tbody>
</table>
|         |              | 17-19      | 1.6     | Register Index -> Command Index
Register Data -> Command Data
Modify the delay time from 10us to 80ns between /CS high and low |
|         |              | 23         | 4       | Modify delay time of power on sequence
Modify OE setting (0x02,0x04) -> (0x02,0x06)                                  |
|         |              | 27-31      | 5.3     | Remove checking BUSY when sending each data byte                             |
|         |              | 32         | 5.4     | Add notes                                                                    |
|         |              | 35         | 7       | Section 1.3.6 move to Appendix                                               |
|         |              | 34         | 6       | Modify delay time of VGL voltage                                             |
### II. Glossary of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPD</td>
<td>Electrophoretic Display (e-Paper Display)</td>
</tr>
<tr>
<td>EPD Panel</td>
<td>EPD</td>
</tr>
<tr>
<td>Tcon</td>
<td>Timing Controller</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>FPL</td>
<td>Front Plane Laminate (e-Paper Film)</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>COG</td>
<td>Chip On Glass, Driver IC. There are two models: Gen1 (G1) is EOL and Gen2 (G2) which is used in this document</td>
</tr>
<tr>
<td>PDI, PDi</td>
<td>Pervasive Displays Incorporated</td>
</tr>
</tbody>
</table>
1. General Description

1.1 Overview

This document explains the interface to the G2 COG Driver to operate the EPD for a Timing Controller based solution using two pages of memory buffer. G2 COG Driver is the most recent EPD driving technology from PDi that offers new features such as breakage detection, lower inrush current, and a lower operation voltage.

The procedure to update display is:
1. Store new pattern in memory buffer
2. Power on G2 COG Driver
3. Initialize G2 COG Driver
4. Update display stage by stage
5. Power off G2 COG Driver

Refer to the EPD controller in section 1.5 for the complete update cycle from Power On, Initialize, Update and Power off. To operate the EPDs for the best sharpness and performance, each update of the panel is divided into a series of stages before the display of the new image pattern is completed. During each stage, frame updates with intermediate image patterns are repeated for a specified period of time. The number of repeated frame updates during each stage is dependent on the Timing Controller speed. After the final stage, the new pattern is displayed.

Section 1 is an overview and contains supporting information such as the overall theory for updating an EPD, SPI timing for PDI’s EPDs, as well as current profiles.

Section 2 describes a method to write to memory buffer. Previously updated and new patterns are stored in the memory buffer to compare the old and new image patterns during the update.

Section 3 describes how to power on the G2 COG Driver which consists of applying a voltage and generating the required signals for /CS and /RESET.

Section 4 describes the steps to initialize the G2 COG Driver.

Section 5 describes the details on how to update the EPD from the memory buffer, create a line of data, update in stages.

Section 6 describes how to power off the G2 COG Driver, and discharge voltage from EPD to ground, make sure there is no voltage remains in EPD.
# 1.2 Input Terminal Pin Assignment

<table>
<thead>
<tr>
<th>No</th>
<th>Signal</th>
<th>I/O</th>
<th>Connected to</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/CS</td>
<td>I</td>
<td>TCon</td>
<td>Chip Select. Low enable</td>
</tr>
<tr>
<td>2</td>
<td>BUSY</td>
<td>O</td>
<td>TCon</td>
<td>When BUSY = HIGH, EPD stays in busy state that EPD ignores any input data from SPI.</td>
</tr>
<tr>
<td>3</td>
<td>ID</td>
<td>I</td>
<td>Ground</td>
<td>Set SPI interface</td>
</tr>
<tr>
<td>4</td>
<td>SCLK</td>
<td>I</td>
<td>TCon</td>
<td>Clock for SPI</td>
</tr>
<tr>
<td>5</td>
<td>SI</td>
<td>I</td>
<td>TCon</td>
<td>Serial input from host Timing Controller to EPD</td>
</tr>
<tr>
<td>6</td>
<td>SO</td>
<td>O</td>
<td>TCon</td>
<td>Serial output from EPD to host Timing Controller</td>
</tr>
<tr>
<td>7</td>
<td>/RESET</td>
<td>I</td>
<td>TCon</td>
<td>Reset signal. Low enable</td>
</tr>
<tr>
<td>8</td>
<td>BORDER_DRV ER or PWRON</td>
<td>-</td>
<td>BORDER or NC</td>
<td>For 1.44” &amp; 2”, connect to BORDER. For 1.9”, 2.6” &amp; 2.7”, keep open.</td>
</tr>
<tr>
<td>9</td>
<td>Vcl</td>
<td>C</td>
<td>Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>C42P</td>
<td>C</td>
<td>NC</td>
<td>Not connected. These two pins are used only with G1 COG Drive IC.</td>
</tr>
<tr>
<td>11</td>
<td>C42M</td>
<td>C</td>
<td>NC</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>C41P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>C41M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>C31M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>C31P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>C21M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>C21P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>C16M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>C16P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>C15M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>C15P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>C14M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>C14P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td>-</td>
</tr>
<tr>
<td>No</td>
<td>Signal</td>
<td>I/O</td>
<td>Connected to</td>
<td>Function</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>-----</td>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>24</td>
<td>C13M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>C13P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>C12M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>C12P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>C11M</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>C11P</td>
<td>C</td>
<td>Charge-Pump Capacitor</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>VCOM_DRIVER</td>
<td>RC</td>
<td>Resistor &amp; Capacitor</td>
<td>The duty cycle of VCOM_DRIVER can adjust VCOM voltage from source driver IC</td>
</tr>
<tr>
<td>31</td>
<td>VCC</td>
<td>P</td>
<td>VCC</td>
<td>Power supply for analog part of source driver</td>
</tr>
<tr>
<td>32</td>
<td>VDD</td>
<td>P</td>
<td>VDD</td>
<td>Power supply for digital part of source driver</td>
</tr>
<tr>
<td>33</td>
<td>VSS</td>
<td>P</td>
<td>Ground</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>VGH</td>
<td>C</td>
<td>Capacitor</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>VGL</td>
<td>C</td>
<td>Capacitor</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>VDH</td>
<td>C</td>
<td>Capacitor</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>VDL</td>
<td>C</td>
<td>Capacitor</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>BORDER</td>
<td>I</td>
<td>-</td>
<td>For 1.44” &amp; 2”, connect to BORDER_DRIVER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For 2.7”, connect to VDL via control circuit for white frame border</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For 1.9” &amp; 2.6”, not connected</td>
</tr>
<tr>
<td>39</td>
<td>VST</td>
<td>P</td>
<td>VCOM_PANEL</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>VCOM_PANEL</td>
<td>C</td>
<td>Capacitor</td>
<td>VCOM to panel</td>
</tr>
</tbody>
</table>

**Note:**

1. **I:** Input, **O:** Output, **C:** Capacitor, **RC:** Resistor and Capacitor, **P:** Power, **NC:** Not connected (Do not SMT)
2. Recommend to use an independent SPI bus to control the EPD.
3. Around the active area of the EPD is a 0.5mm width blank area called the BORDER. It’s connected to VDL (-13V ~ -14V) to keep the border white. After EPD updates with the constant voltage, the border color may degrade to a gray level that is not as white as the active area. Reset the Border per screen update to avoid this phenomenon.
1.3 Reference Circuit

1.3.1 1.44 inch EPD Reference Circuit

Note:

1. VDD and VCC must be discharged promptly after power off
2. 1.44" Pin.1 location
3. R1 is connected.

Pin.1 /CS

Pin.40 VCOM_PANEL
1.3.2 2 inch EPD Reference Circuit

INPUT

Note:
1. VDD and VCC must be discharged promptly after power off
2. 2" Pin.1 location
3. R1 is connected.
1.3.3 2.7 inch EPD Reference Circuit

**INPUT**

Connect to Timing Controller SPI
Connect to Timing Controller GPIO
Connect to Power Ground
Connect to Switch
Connect to Timing Controller GPIO
Connect to Timing Controller SPI
Connect to Power Ground

**Note:**

1. VDD and VCC must be discharged promptly after power off
2. 2.7" Pin.1 location

**Note:**

- DISCHARGE: set high for EPD discharge when EPD power off
- BORDER Control: Square Pulse when power on

**Connectors/Pads:**

- Pin.1 /CS
- pin.40 VCOM_PANEL

---

This document is the exclusive property of PDI and shall not be reproduced or copied or transformed to any other format without prior permission of PDI. (PDI Confidential) 本資料為龍亭新技專有之財產，非經許可，不得複製、翻印或轉變成其他形式使用。
1.3.4 1.9 inch EPD Reference Circuit

**INPUT**

Connect to Timing Controller SPI
Connect to Timing Controller SPI
Connect to Timing Controller SPI
Connect to Timing Controller SPI
Connect to Timing Controller GPIO

Connect to Power Ground

Connect to Power Switch

Connect to a MOS Switch to prevent leakage current

DISCHARGE: set high for EPD discharge when EPD power off

Note:
1. V_DD and V_CC must be discharged promptly after power off
2. 1.9” Pin.1 location

![Diagram](image)
1.3.5 2.6 inch EPD Reference Circuit

Note:
1. V_DD and V_CC must be discharged promptly after power off
2. 2.6" Pin.1 location

If you are still using our old COG driver G1 and previous circuit, please refer to the Appendix.
1.4 EPD Driving Flow Chart

The flowchart below provides an overview of the actions necessary to update the EPD. We call this is Global Update or Full Update. The steps below refer to the detailed descriptions in the respective sections.

Start

Section 2
Write to the memory

Section 3
Power On G2 COG Driver

Section 4
Initialize G2 COG Driver

Section 5
Write Data from the Memory to the EPD

Update Complete?

Yes

Section 6
Power Off G2 COG Driver

End
1.5 Controller

The diagram below provides a signal control overview during an EPD update cycle.
The diagram is divided into
"3. Power On G2 COG Driver",
"4. Initialize G2 COG Driver",
"5. Write Data from the Memory to the EPD",
"6. Power Off G2 COG Driver",
The number and title matches a section title in this document.

Note:
1. /BORDER_CONTROL:
   /BORDER_CONTROL is used to keep a sharp border while taking care of the electronic ink particles.
   For implement this function, developer needs to use a Timing Controller pin (GPIO) to control this signal.
   (This function is only used for 2.7" EPD Panel)
1.6 SPI Timing Format

SPI commands are used to communicate between the Timing Controller and the G2 COG Driver. The SPI format differs from the standard in that two way communications, and /CS is pulled high then low between Command Index and Command Data. When setting up the SPI timing, please follow the SPI command format and timing in this section and verify the control signals in Section 1.5 then.

The maximum SPI clock speed for G2 COG Driver is 20MHz.

The SPI mode is 0.

Below is a description of the SPI Format:

SPI (0xI, 0xD1, 0xD2, 0xD3 ...)

Where:

I is the Command Index and the length is 1 byte
D1-n is the Command Data. The Command Data length is from 1 to 110 bytes depending on which Command Index is selected.

<table>
<thead>
<tr>
<th>Command Index</th>
<th>Number Bytes of Command Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>8</td>
</tr>
<tr>
<td>0x02</td>
<td>1</td>
</tr>
<tr>
<td>0x03</td>
<td>1</td>
</tr>
<tr>
<td>0x04</td>
<td>1</td>
</tr>
<tr>
<td>0x05</td>
<td>1</td>
</tr>
<tr>
<td>0x07</td>
<td>1</td>
</tr>
<tr>
<td>0x08</td>
<td>1</td>
</tr>
<tr>
<td>0x09</td>
<td>1</td>
</tr>
<tr>
<td>0x0A</td>
<td>&lt; 110</td>
</tr>
<tr>
<td>0x0B</td>
<td>1</td>
</tr>
<tr>
<td>0x0F</td>
<td>1</td>
</tr>
</tbody>
</table>

- Before sending the Command Index, the SPI (SI) must send a 0x70 (header of Command Index).
- Likewise, the SPI (SI) must send a 0x72 (header of Command Data) prior to the Command Data. The flow chart and detailed description can be found in the next page.
- Number Bytes of Command Index (0x0A) depends on which panel size is used.
SPI write command signals and flowchart (SPI):

Example:
Send two continue SPI write commands
SPI(0x08, 0x03) and SPI(0x09, 0xA1)

If Command Data is larger than two bytes, you must input data continuously without setting Command Index again.
SPI read command signals and flowchart (SPI_R):

Example:
Send one SPI read command
SPI_R(0x0F,0x00)

Send Dummy Data (0x00)
SPI read COG Driver ID and flowchart(SPI_RID):

SCLK ———— Header ————
    \                    / 0x00
        \                /
        SI ———— 7 1 ————

SO ———— COG Driver ID ————

/CS ———— ———— ———— ———— ————

SPI_RID

/CS = 1

Delay ≥ 80ns

/CS = 0

Header

0x71

send Dummy Data

(0x00)

Dummy Data Read Complete?

Yes

/CS = 1

No
- SPI command timing (mode 0)

![SPI timing diagram](image)

<table>
<thead>
<tr>
<th>Item</th>
<th>Signal</th>
<th>Symbol</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial clock cycle</td>
<td>SCLK</td>
<td>tcYS</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SCLK high pulse width</td>
<td>SCLK</td>
<td>twHS</td>
<td>25</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SCLK low pulse width</td>
<td>SCLK</td>
<td>twLS</td>
<td>25</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Data setup time</td>
<td>SI</td>
<td>tdSS</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>Data hold time</td>
<td>SI</td>
<td>tdHS</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>CSB setup time</td>
<td>/CS</td>
<td>tcSS</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>CSB hold time</td>
<td>/CS</td>
<td>tCHS</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

VCC = 2.3 to 3.6V  Temp = 0 to 50°C
2. Write to the Memory

Before powering on G2 COG Driver, developer should write the new pattern to image buffer, either SRAM or flash memory. The image pattern must be converted to a 1 bit bitmap format (Black/White) prior to writing.

Two buffer spaces should be allocated to store both previous and new patterns. The previous pattern is the currently displayed pattern. The new pattern will be written to the EPD. The G2 COG Driver will compare both patterns before updating the EPD. The table below lists the buffer space size required for each EPD size.

<table>
<thead>
<tr>
<th>EPD size</th>
<th>Image resolution(pixels)</th>
<th>Previous image + New image Buffer (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44”</td>
<td>128 x 96</td>
<td>3,072</td>
</tr>
<tr>
<td>2”</td>
<td>200 x 96</td>
<td>4,800</td>
</tr>
<tr>
<td>2.7”</td>
<td>264 x 176</td>
<td>11,616</td>
</tr>
<tr>
<td>1.9”</td>
<td>144 x 128</td>
<td>4,608</td>
</tr>
<tr>
<td>2.6”</td>
<td>232 x 128</td>
<td>7,424</td>
</tr>
</tbody>
</table>
3. Power On G2 COG Driver

This flowchart describes power on sequence for the G2 COG Driver.

1. Start:
   - Initial State:
     - $V_{CC}/V_{DD} = 0$
     - $/RESET, /CS, /BORDER\_CONTROL^{*2}, SI, SCLK = 0$

2. /BORDER\_CONTROL:
   - GPIO pin for 2.7” EPD Panel to reset the Border.
4. Initialize G2 COG Driver

Section 3
Power On

Busy == 1

Check COG Driver ID
SPI_RID == 0x12

Yes

Power Saving Mode
SPI(0x0B, 0x02)

No

Check Breakage
SPI_R(0x0F, 0x00) & 0x80 == 0x80

Yes

Driver latch on
SPI(0x03, 0x01)

Driver latch off
SPI(0x03, 0x00)

Delay ≥ 5ms

Chargepump Start
i = 1

Delay ≥ 150ms

Start chargepump positive voltage
VGH & VDH on
SPI(0x05, 0x01)

Delay ≥ 90ms

Start chargepump negative voltage
VGL & VDL on
SPI(0x05, 0x03)

Delay ≥ 40ms

Set chargepump Vcom on
SPI(0x05, 0x0F)

Check Breakage
SPI_R(0x0F, 0x00) & 0x40 == 0x40

Yes

No

Output enable to disable
SPI(0x02, 0x06)

Set Vcom level
SPI(0x09, 0xC2)

Delay ≥ 40ms

Power Setting
SPI(0x04, 0x03)

Write data from the Memory to the EPD

Delay ≥ 40ms

Check DC/DC
SPI_R(0x0F, 0x00) & 0x40 == 0x40

Yes

No

Delay ≥ 40ms

Power Setting
SPI(0x02, 0x40)

Delay ≥ 5ms

Driver latch on
SPI(0x03, 0x01)

Driver latch off
SPI(0x03, 0x00)

Delay ≥ 5ms

Chargepump Start
i = 1

Delay ≥ 150ms

Start chargepump positive voltage
VGH & VDH on
SPI(0x05, 0x01)

Delay ≥ 90ms

Start chargepump negative voltage
VGL & VDL on
SPI(0x05, 0x03)

Delay ≥ 40ms

Set chargepump Vcom on
SPI(0x05, 0x0F)

Check Breakage
SPI_R(0x0F, 0x00) & 0x80 == 0x80

Yes

No

Power Setting
SPI(0x08, 0x02)

Delay ≥ 40ms

Check COG Driver ID
SPI_RID == 0x12

Yes

[Flowchart diagram of initialization process]
Note:

1 SPI timing format (Refer to “1.6 SPI Timing Format” for detail)
   - SPI read COG Driver ID command: SPI_RID
     - G1 COG Driver ID is 0x11
     - G2 COG Driver ID is 0x12

2 SPI(0x01, Data):
   - Different by each size
     - 1.44”: SPI(0x01, (0x0000,0000,000F,FF00))
     - 2”: SPI(0x01, (0x0000,0000,01FF,E000))
     - 2.7”: SPI(0x01, (0x0000,007F,FFFE,0000))
     - 1.9”: SPI(0x01, (0x0000,0003,FC00,00FF))
     - 2.6”: SPI(0x01, (0x0000,1FE0,0000,00FF))
   - Take 2” for example, to send first byte protocol (0x70) before Command Index (0x01), and then send second byte protocol (0x72) before Command Data (0x0000,0000,01FF,E000).

3 Should measure VGH >12V and VDH >8V

4 Should measure VGL <-12V and VDL <-8V
5 Write Data from the Memory to the EPD

5.1 Data Structure

This section describes how data should be sent to the G2 COG Driver which will update the display. The G2 COG Driver uses a buffer to store a line of data and then writes to the display.

- EPD Resolutions

<table>
<thead>
<tr>
<th>EPD size</th>
<th>Image resolution(pixels)</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44”</td>
<td>128 x 96</td>
<td>128</td>
<td>96</td>
</tr>
<tr>
<td>2”</td>
<td>200 x 96</td>
<td>200</td>
<td>96</td>
</tr>
<tr>
<td>2.7”</td>
<td>264 x 176</td>
<td>264</td>
<td>176</td>
</tr>
<tr>
<td>1.9”</td>
<td>144 x 128</td>
<td>144</td>
<td>128</td>
</tr>
<tr>
<td>2.6”</td>
<td>232 x 128</td>
<td>232</td>
<td>128</td>
</tr>
</tbody>
</table>

- Data components
  - One Bit – A bit can be W (White), B (Black) or N (Nothing) bit. Using the N bit to keep the pixel for the same color if new bit is same as previous one and also avoid overdriving the pixel to reduce ghosting phenomenon.
  - One Dot/pixel is comprised of 2 bits.
  - One Line is the number of dots in a line.

For example:
- The 1.44” uses 128 Dots to represent 1 Line.
- The 2” uses 200 Dots to represent 1 Line.
- The 2.7” uses 264 Dots to represent 1 Line.
- The 1.9” uses 144 Dots to represent 1 Line.
- The 2.6” uses 232 Dots to represent 1 Line.
- The G2 COG Driver uses a buffer to write one line of data (Mapping) – interlaced. The table below is a Line data of 2” including Odd data bytes, Scan bytes and Even data bytes.
### 5.2 Overall Update Flow

<table>
<thead>
<tr>
<th>Previous Image</th>
<th>Inverse Previous Image (Compensate)</th>
<th>White</th>
<th>Inverse New Image</th>
<th>New Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- One frame of data is the number of lines * rows.

For example:
- 1.44”: One frame of data is 96 lines * 128 dots.
- 2”: One frame of data is 96 lines * 200 dots.
- 2.7”: One frame of data is 176 lines * 264 dots.
- 1.9”: One frame of data is 128 lines * 144 dots.
- 2.6”: One frame of data is 128 lines * 232 dots.

### 5.3 Store a Line of Data in the Buffer

This section describes the details of how to send data to the G2 COG Driver. The G2 COG Driver uses a buffer to update the display line by line.

The Nothing data byte can be either 0x00 or 0x01 where 0x00 is the default Nothing data for G2 driver.

---

- This document is the exclusive property of PDI and shall not be reproduced or copied or transformed to any other format without prior permission of PDI. (PDI Confidential)
1.44” Input Data Order

Note:
1. Turn on OE:
   Output data from COG driver to panel.

2. **Data**

<table>
<thead>
<tr>
<th>x,y</th>
<th>bit1</th>
<th>bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Black (B)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>White (W)</td>
</tr>
<tr>
<td>y = 1 – 96</td>
<td>0</td>
<td>0</td>
<td>Nothing (N)</td>
</tr>
</tbody>
</table>

Example:
- D(127,y) = Black (B) = 11
- D(125,y) = White (W) = 10
- D(123,y) = Nothing (N) = 00
- D(121,y) = Black (B) = 11

→ 1st Data Byte = 11,10,00,11

**Scan**

<table>
<thead>
<tr>
<th>y = 1</th>
<th>bit1</th>
<th>bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Scan on</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Scan off</td>
</tr>
</tbody>
</table>

Example:
- When y = 2,
  → Only S(2) is Scan on (11) while others are Scan off (00). The image represented by Data Bytes will be displayed on 2nd horizontal line (i.e. Dot(1,2) ~ Dot(128,2)).

| S(1) | Scan off = 00 |
| S(2) | Scan on = 11  |
| S(3) | Scan off = 00 |
| S(4) | Scan off = 00 |
| ...  |               |
| S(96) | Scan off = 00 |

→ 1st ~ 23rd Scan Byte = 00,00,00,00
→ 24th Scan Byte = 00,00,11,00

---

```
(1,1) (2,1) (3,1) (128,1)
(1,2) (2,2) (3,2) (128,2)
(1,96) (128,96)
```

---

Frame start

Line start

y = 1

Sending Data

SPI(0x0A, Data)

1st Data Byte

D(127,y), D(125,y), D(123,y), D(121,y)

24th Data Byte

D(122,y), D(124,y), D(126,y), D(128,y)

1st Scan Byte

S(96), S(95), S(94), S(93)

24th Scan Byte

S(4), S(3), S(2), S(1)

17th Data Byte

D(2,y), D(4,y), D(6,y), D(8,y)

32th Data Byte

D(122,y), D(124,y), D(126,y), D(128,y)

Border Byte

0x00

Turn on OE

SPI(0x02, 0x07)

Output data from COG driver to panel.

Line end

y++

Yes

Frame end

Note:
1. Turn on OE:
   Output data from COG driver to panel.
2. Add Border Byte to complete a line.
• **2” Input Data Order**

**Note:**

1. Turn on OE:
   Output data from COG driver to panel.

2. **Data**

<table>
<thead>
<tr>
<th>Data</th>
<th>bit1 bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(x,y)</td>
<td>1 1</td>
<td>Black (B)</td>
</tr>
<tr>
<td>x = 1~200</td>
<td>0 0</td>
<td>White (W)</td>
</tr>
<tr>
<td>y = 1~96</td>
<td>0 0</td>
<td>Nothing (N)</td>
</tr>
</tbody>
</table>

Example:
D(199,y) = Black (B) = 11
D(197,y) = White (W) = 10
D(195,y) = Nothing (N) = 00
D(193,y) = Black (B) = 11
→ 1st Data Byte = 11,10,00,11

**Scan**

<table>
<thead>
<tr>
<th>Scan</th>
<th>bit1 bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(1)~S(96)</td>
<td>1 1</td>
<td>Scan on</td>
</tr>
<tr>
<td></td>
<td>0 0</td>
<td>Scan off</td>
</tr>
</tbody>
</table>

Example:
When y = 2,
→ Only S(2) is Scan on (11) while others are Scan off (00). The image represented by Data Bytes will be displayed on 2nd horizontal line (i.e. Dot(1,2) ~ Dot(200,2)).

S(1) = Scan off = 00
S(2) = Scan on = 11
S(3) = Scan off = 00
S(4) = Scan off = 00
S(96) = Scan off = 00
→ 1st ~ 23rd Scan Byte = 00,00,00,00
→ 24th Scan Byte = 00,00,11,00

---

**Turn on OE**

Output data from COG driver to panel.

**Frame end**
2.7” Input Data Order

Note:

1. Turn on OE:
   Output data from COG driver to panel.

2.

<table>
<thead>
<tr>
<th>Data</th>
<th>bit1 bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(x,y)</td>
<td>1 1</td>
<td>Black (B)</td>
</tr>
<tr>
<td>x = 1-264</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y = 1-176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>White (W)</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>Nothing (N)</td>
<td></td>
</tr>
</tbody>
</table>

Example:
D(263,y) = Black (B) = 11
D(261,y) = White (W) = 10
D(259,y) = Nothing (N) = 00
D(257,y) = Black (B) = 11
\(\rightarrow 1^{st} \) Data Byte = 11,10,00,11

<table>
<thead>
<tr>
<th>Scan</th>
<th>bit1 bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(1) ~ S(176)</td>
<td>1 1</td>
<td>Scan on</td>
</tr>
<tr>
<td>0 0</td>
<td>Scan off</td>
<td></td>
</tr>
</tbody>
</table>

Example:
When y = 2,
\(\rightarrow \) Only S(2) is Scan on (11) while others are Scan off (00). The image represented by Data Bytes will be displayed on 2nd horizontal line (i.e. Dot(1,2) ~ Dot(264,2)).

S(1) = Scan off = 00
S(2) = Scan on = 11
S(3) = Scan off = 00
S(4) = Scan off = 00
\(\vdots\)
S(176) = Scan off = 00
\(\rightarrow 1^{st} - 43^{rd} \) Scan Byte = 00,00,00,00
\(\rightarrow 44^{th} \) Scan Byte = 00,00,11,00

\[ (1,1) \quad (2,1) \quad (3,1) \quad \ldots \quad (264,1) \]
\[ (1,2) \quad (2,2) \quad (3,2) \quad \ldots \quad (176) \]
\[ (2,3) \quad \ldots \quad (264,176) \]

Frame start

Line start
\( y = 1 \)

Sending Data
SPI(0x0A, Data)

0x00

1st Data Byte
(D(263,y), D(261,y), D(259,y), D(257,y))

33rd Data Byte
(D(7,y), D(5,y), D(3,y), D(1,y))

44th Data Byte
(D(258,y), D(260,y), D(262,y), D(264,y))

1st Scan Byte
(S(176), S(175), S(174), S(173))

44th Scan Byte
(S(4), S(3), S(2), S(1))

16th Data Byte
(D(2,y), D(4,y), D(6,y), D(8,y))

Turn on OE*
SPI(0x02, 0x07)

Output data from COG driver to panel.

Line end
\( y ++ \)

Add eight bits 0 to complete a line.

Total displayed data in a line: \((264+176)\times2 \) bits

Frame end

\( y == 177 \)
1.9” Input Data Order

Note:

1. Turn on OE:
   Output data from COG driver to panel.

2. The following table provides the input data order for each scan byte and data byte.

<table>
<thead>
<tr>
<th>Data</th>
<th>bit1</th>
<th>bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(x,y)</td>
<td>1</td>
<td>1</td>
<td>Black (B)</td>
</tr>
<tr>
<td>x = 1~144</td>
<td>1</td>
<td>0</td>
<td>White (W)</td>
</tr>
<tr>
<td>y = 1~128</td>
<td>0</td>
<td>0</td>
<td>Nothing (N)</td>
</tr>
</tbody>
</table>

Example:

D(1,y) = Black (B) = 11
D(2,y) = White (W) = 10
D(3,y) = Nothing (N) = 00
D(4,y) = Black (B) = 11
→ 1st Data Byte = 11,10,00,11

Scan | bit1 | bit0 | Input |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S(1) ~ S(128)</td>
<td>1</td>
<td>1</td>
<td>Scan on</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Scan off</td>
<td></td>
</tr>
</tbody>
</table>

Example:

When y = 2,
→ Only S(2) is Scan on (11) while others are Scan off (00). The image represented by Data Bytes will be displayed on 2nd horizontal line (i.e. Dot(1,2) ~ Dot(144,2)).

<table>
<thead>
<tr>
<th>D(x,y)</th>
<th>S(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(2)</td>
<td>= Scan off = 11</td>
</tr>
<tr>
<td>(3)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(4)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(5)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(6)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(7)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(8)</td>
<td>= Scan off = 00</td>
</tr>
<tr>
<td>(128)</td>
<td>= Scan off = 00</td>
</tr>
</tbody>
</table>

→ 1st Scan Byte = 11,00,00,00
→ 2nd ~ 32nd Scan Byte = 00,00,00,00

Each Stage inputs different Border Byte.

Stage 1 ~ Stage 4 : Data

<table>
<thead>
<tr>
<th>Stage</th>
<th>Border Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>0xAA</td>
</tr>
</tbody>
</table>

Total displayed data in a line: (144+128)x2 bits

Output data from COG driver to panel.
2.6” Input Data Order

Note:

1. Turn on OE:
   Output data from COG driver to panel.

2. Data | bit1 bit0 | Input
       |       |       |
D(x,y) | 1 1   | Black (B) |
x = 1~232
y = 1~128 | 1 0 | White (W) |
0 0     | Nothing (N) 

Example:
D(1,y) = Black (B) = 11
D(2,y) = White (W) = 10
D(3,y) = Nothing (N) = 00
D(4,y) = Black (B) = 11
→ 1st Data Byte = 11,10,00,11

Scan | bit1 bit0 | Input
     |       |       |
S(1) ~ S(128) | 1 1 | Scan on |
0 0     | Scan off |

Example:
When y = 2,
→ Only S(2) is Scan on (11) while others are Scan off (00). The image represented by Data Bytes will be displayed on 2nd horizontal line (i.e. Dot(1,2) ~ Dot(232,2)).

S(1) = Scan off = 00
S(2) = Scan on = 11
S(3) = Scan off = 00
S(4) = Scan off = 00
S(5) = Scan off = 00
S(6) = Scan off = 00
S(7) = Scan off = 00
S(8) = Scan off = 00
...
S(128) = Scan off = 00
→ 1st Scan Byte = 11,00,00,00
→ 2nd ~ 32nd Scan Byte = 00,00,00,00

x

y

1st Scan Byte
(S(2), S(4), S(6), S(8))

16th Scan Byte
(S(122), S(124), S(126), S(128))

1st Data Byte
(D(232,y), D(231,y), D(230,y), D(229,y))

58th Data Byte
(D(4,y), D(3,y), D(2,y), D(1,y))

17th Scan Byte
(S(127), S(125), S(123), S(121))

32nd Scan Byte
(S(7), S(5), S(3), S(1))

Border Byte
Stage 1 ~ Stage 4: Data

Stage 1,2,3 0x00
Stage 4 0xAA

Turn on OE*
SPI(0x02, 0x07)

Line end
y ++

| y = 129 |

Frame end
5.4 Writing to the Display in Stages

This section contains the method to write to the display in stages. Each of the 4 stages should be the same use the same number of frames. Rewrite the frame during each stage.

The following flow chart describes how to update an image from a previous displayed image stored in memory buffer to a new image also stored in memory buffer. See the sample previous and new images below.

<table>
<thead>
<tr>
<th>Previous Display</th>
<th>New Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Panel Size</th>
<th>Stage Time (ms)</th>
<th>Frame Time of Timing Controller (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44”</td>
<td>480</td>
<td>&lt; 50ms</td>
</tr>
<tr>
<td>2”</td>
<td>480</td>
<td>&lt; 50ms</td>
</tr>
<tr>
<td>1.9”</td>
<td>480</td>
<td>&lt; 50ms</td>
</tr>
<tr>
<td>2.7”</td>
<td>630</td>
<td>&lt; 80ms</td>
</tr>
<tr>
<td>2.6”</td>
<td>630</td>
<td>&lt; 70ms</td>
</tr>
</tbody>
</table>

Note:

1. Stage Time is the number of frames used to write an intermediate pattern. This can vary based on the Frame Time of Timing Controller (or MCU).
   - Example: Use 2 inch for example
     - If the Frame Time of Timing Controller is 40ms, it needs 12 frames at least of data per stage. (480/40 = 12)
     - If the Frame Time of Timing Controller is 36ms, it cannot divisible the Stage Time, users can choose one of the two methods as below to determine the number of frames
       1. Frame Time = 36ms (480/36 = 13.33), users can drive 14 frames of data per stage at least.
       2. Frame Time = 36ms (480/36 = 13.33), users can drive 13 frames of data per stage. And use equal delay time in each frame to meets the Stage Time.
          Example: 36ms * 13 + delay time= 480ms.

2. Frame Time is based on the MCU that PDi used to recommend a maximum limit value of Frame time.

3. This table is tested with PDi’s jig.
1. The previous image stored in memory is used to determine how to write the data for both Stage 1 and Stage 2.

2. The new image stored in memory is used to determine how to write the data for both Stage 3 and Stage 4.

3. Optional: The optical performance is dependent on Stage Time. If the ghosting is at unacceptable level, the EPD can be rewritten and then Stage 4 repeated to write the New image.

4. It needs (Stage Time * TF) ms to finish a stage.

5. The TF below 0°C is for reference only. PDI does not guarantee the performance and functionality below 0°C.

6. If you use Flash memory for the Section 2, please erase the buffer when Stage 4 is completed.

---

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>TF*5</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ -10</td>
<td>17</td>
</tr>
<tr>
<td>-5 ≤ T &gt; -10</td>
<td>12</td>
</tr>
<tr>
<td>5 ≥ T &gt; -5</td>
<td>8</td>
</tr>
<tr>
<td>10 ≥ T &gt; 5</td>
<td>4</td>
</tr>
<tr>
<td>15 ≥ T &gt; 10</td>
<td>3</td>
</tr>
<tr>
<td>20 ≥ T &gt; 15</td>
<td>2</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>0.7</td>
</tr>
</tbody>
</table>

---

**Stage 1**

Compensate
(Stage Time * TF) ms

**Stage 2**

White
(Stage Time * TF) ms

**Stage 3**

Inverse
(Stage Time * TF) ms

**Stage 4**

Normal
(Stage Time * TF) ms

---

**Initialize Driver**

Sense Temperature to determine Temperature Factor (TF)

**Stage 1**

```
<table>
<thead>
<tr>
<th>bit1</th>
<th>bit0</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Black (B)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>White (W)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Nothing (N)</td>
</tr>
</tbody>
</table>
```

**Stage R**

```
<table>
<thead>
<tr>
<th>New*2</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>W</td>
</tr>
</tbody>
</table>
```

**Display**

Current/Previous Display

---

**Stage 2**

```
| Previous*1 | B     | W |
| Display     | W     | B |
```

**Stage 3**

```
| New*2      | B     | W |
| Display     | N     | B |
```

**Stage 4**

```
| New*2      | B     | W |
| Display     | W     | B |
```

---

**Power Off**
6 Power Off G2 COG Driver

1. Nothing Frame:
   Write a frame data
<table>
<thead>
<tr>
<th>Panel size</th>
<th>Scan Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44&quot;</td>
<td>96</td>
</tr>
<tr>
<td>2&quot;</td>
<td>96</td>
</tr>
<tr>
<td>2.7&quot;</td>
<td>176</td>
</tr>
<tr>
<td>1.9&quot;</td>
<td>128</td>
</tr>
<tr>
<td>2.6&quot;</td>
<td>128</td>
</tr>
</tbody>
</table>
   Whose Data Bytes are 0x00. Scan Bytes operate normally.
   Scan lines are still turned on sequentially. This frame will make the image more uniform.
   Turn on OE SPI(0x02, 0x07) at the end of each line.
   For 1.44", 1.9", 2" & 2.6", need to set Border Byte to complete 1st Data Byte.
   Border Byte = 0x00 for 1.44" & 2"
   0xAA for 1.9" & 2.6"

2. Dummy Line:
   A line whose all Data Bytes are 0x00 and Scan Bytes are 0x00. Turning on OE SPI(0x02, 0x07) to complete this Dummy Line. Clear the register data before power off. Detail of data input is on page 27 – page 31.
   (This function is only used in 2.7")

3. /BORDER_CONTROL:
   When = 0, the /BORDER_CONTROL is ON and write to white. When = 1, the /BORDER_CONTROL is OFF.
   (This function is only used in 2.7")

4. External Discharge:
   For implement this function, users need to use a DISCHARGE pin from microcontroller to control.(refer to the reference circuit)
   This is important to avoid vertical lines.

5. If you use the flash memory for pattern store, please recheck flash in this phase and verify the old image flash is erased.

6. BORDER Dummy Line:
   Set Border Byte = 0x00A and write to white. A line whose all Data Bytes are 0x00 and Scan Bytes are 0x00. Then must to set SPI(0x002, 0x07) in the end of line for turn on output enable of COG Driver to control border and clear the register data before power off. Detail of data input is on page 27 – page 31.
   (This function is only used in 1.44", 1.9", 2" & 2.6")

7. Delay time of VGL and VDL:
   Please ensure to place the accurate delay time here until VGL is discharged to GND by measured by oscilloscope.
Appendix: Use G1’s PCBA to drive the EPD with G2 Driver IC

Below is the reference circuit if you have arranged the PCBA that drove the EPD with G1 Driver IC already.

**Note:**

1. Hardware setting for different size:

<table>
<thead>
<tr>
<th>Size</th>
<th>R1</th>
<th>Q3, Q4, R3, R4, R5</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.44 inch &amp; 2 inch</td>
<td>Mounted</td>
<td>No Mounted</td>
<td>Mounted</td>
</tr>
<tr>
<td>2.7 inch</td>
<td>No Mounted</td>
<td>Mounted</td>
<td>No Mounted</td>
</tr>
<tr>
<td>1.9 inch &amp; 2.6 inch</td>
<td>No Mounted</td>
<td>No Mounted</td>
<td>No Mounted</td>
</tr>
</tbody>
</table>
If users want to drive the EPD with G2 Driver IC by the current PCBA (i.e. the reference circuit above). Below items are the steps needed to do.

- Keep hardware unchanged as above.
  - Keep Resistor R1 open.
  - Keep BORDER CONTROL (Q3, Q4, R3, R4, and R5) circuit mounted.
- Modify SPI data as the following sections described.
- Disable the Timing Controller GPIO pin, PWM. Keep PWM signal as either 1 or 0.
- No matter what size EPD is, use same power off sequence of section 6.