E-paper Display COG Driver Interface Timing for Room Temperature of 2”, and 2.7” with G2 COG and v230 Film

Description	**Detailed information to design a timing controller for room temperature 2” and 2.7” with G2 COG and v230 film**
Date	**2014/01/28**
Doc. No.	**4P015-00**
Revision	**01**

	Design Engineering		
	Approval	**Check**	**Design**
	李 2014.01.28 欣達	李 2014.01.28 欣達	黃 2014.01.28 朝玄

No.71, Delun Rd., Rende Dist., Tainan City 71743, Taiwan (R.O.C.)
Tel: +886-6-279-5399 Fax: +886-6-279-5300

Copyright

龍亭新技股份有限公司 Pervasive Displays Inc.

No.71, Delun Rd., Rende Dist., Tainan City 71743, Taiwan (R.O.C.)

Tel: +886-6-279-5399

http://www.pervasivedisplays.com

Table of Contents

Revision History

Version	Date	Page (New)	Section	Description
Ver. 01	2014/01/28	All	All	First issued

Glossary of Acronyms

EPD	Electrophoretic Display (e-Paper Display)
EPD Panel	EPD
TCon	Timing Controller
FPL	Front Plane Laminate (e-Paper Film)
SPI	Serial Peripheral Interface
COG	Chip on Glass
PDI, PDi	Pervasive Displays Incorporated

1 General Description

1.1 Overview

This document explains the interface to the G2 COG Driver to operate the EPD for a MCU based solution using one page of memory buffer. G2 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. This document applies to 2.0" and 2.7" EPDs.

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 to see the complete procedure. 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 during each stage is dependent on the MCU speed. After the final stage, the new pattern is displayed.

Around the active area of the EPD is a 0.5mm width blank area called the border. When connected to V_{DL} (-13V ~ -14V) to keep the border white. After approximately 10,000 updates with the constant voltage, the border color may degrade to a gray level that is not as white as the active area. To prevent this phenomenon, PDI recommends turn on and off border to avoid the degradation.

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. New pattern is stored in the memory buffer and update image in displays.

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, and also power down housekeeping steps.

Section 6 describes how to power off the G2 COG Driver, and discharge voltage from EPD to ground, make sure there is not any voltage keep in EPD.

1.2 Input Terminal Pin Assignment

No	Signal	I/O	Connected to	Function
1	/CS	I	TCon	Chip Select. Low enable
2	BUSY	O	TCon	When BUSY = High, EPD stays in busy state that EPD ignores any input data from SPI.
3	ID	I	Ground	Set SPI interface
4	SCLK	I	TCon	Clock for SPI
5	SI	I	TCon	Serial input from host Timing Controller to EPD
6	SO	O	TCon	Serial output from EPD to host Timing Controller
7	/RESET	I	TCon	Reset signal. Low enable
8	BORDER_DRIVER or PWNON	-	BORDER or TCon	For 2”, connect to BORDER For 2.7”, Power ON Switching Pin. - Non-connected or connect to Timing Controller - Low : Power OFF High : Power ON - It has an internal pull-up resistor. It’s ok to float it.
9	V_{CL}	C	Capacitor	-
10	C42P	-	NC	Not connected. These two pins are used only with G1 COG Drive IC.
11	C42M	-		
12	C31M	C	Charge-Pump Capacitor	-
13	C31P	C		-
14	C21M	C	Charge-Pump Capacitor	-
15	C21P	C		-
16	C16M	C	Charge-Pump Capacitor	-
17	C16P	C		-
18	C15M	C	Charge-Pump Capacitor	-
19	C15P	C		-
20	C14M	C	Charge-Pump Capacitor	-
21	C14P	C		-

No	Signal	I/O	Connected to	Function
22	C14M	C	Charge-Pump Capacitor	-
23	C14P	C		-
24	C13M	C	Charge-Pump Capacitor	-
25	C13P	C		-
26	C12M	C	Charge-Pump Capacitor	-
27	C12P	C		-
28	C11M	C	Charge-Pump Capacitor	-
29	C11P	C		-
30	V_{COM_DRIVER}	RC	Resistor & Capacitor	The duty cycle of V_{COM_DRIVER} can adjust V_{COM} voltage from source driver IC
31	V_{CC}	P	V_{CC}	Power supply for analog part of source driver
32	V_{DD}	P	V_{DD}	Power supply for digital part of source driver
33	V_{SS}	P	Ground	-
34	V_{GH}	C	Capacitor	-
35	V_{GL}	C	Capacitor	-
36	V_{DH}	C	Capacitor	-
37	V_{DL}	C	Capacitor	-
38	BORDER	I	-	For 2”, connect to driver IC pin. 8 (BORDER_DRIVER) For 2.7”, connect to V_{DL} via control circuit for white frame border
39	V_{ST}	P	V_{COM_PANEL}	-
40	V_{COM_PANEL}	C	Capacitor	V_{COM} to panel

Note:

I: Input

O: Output

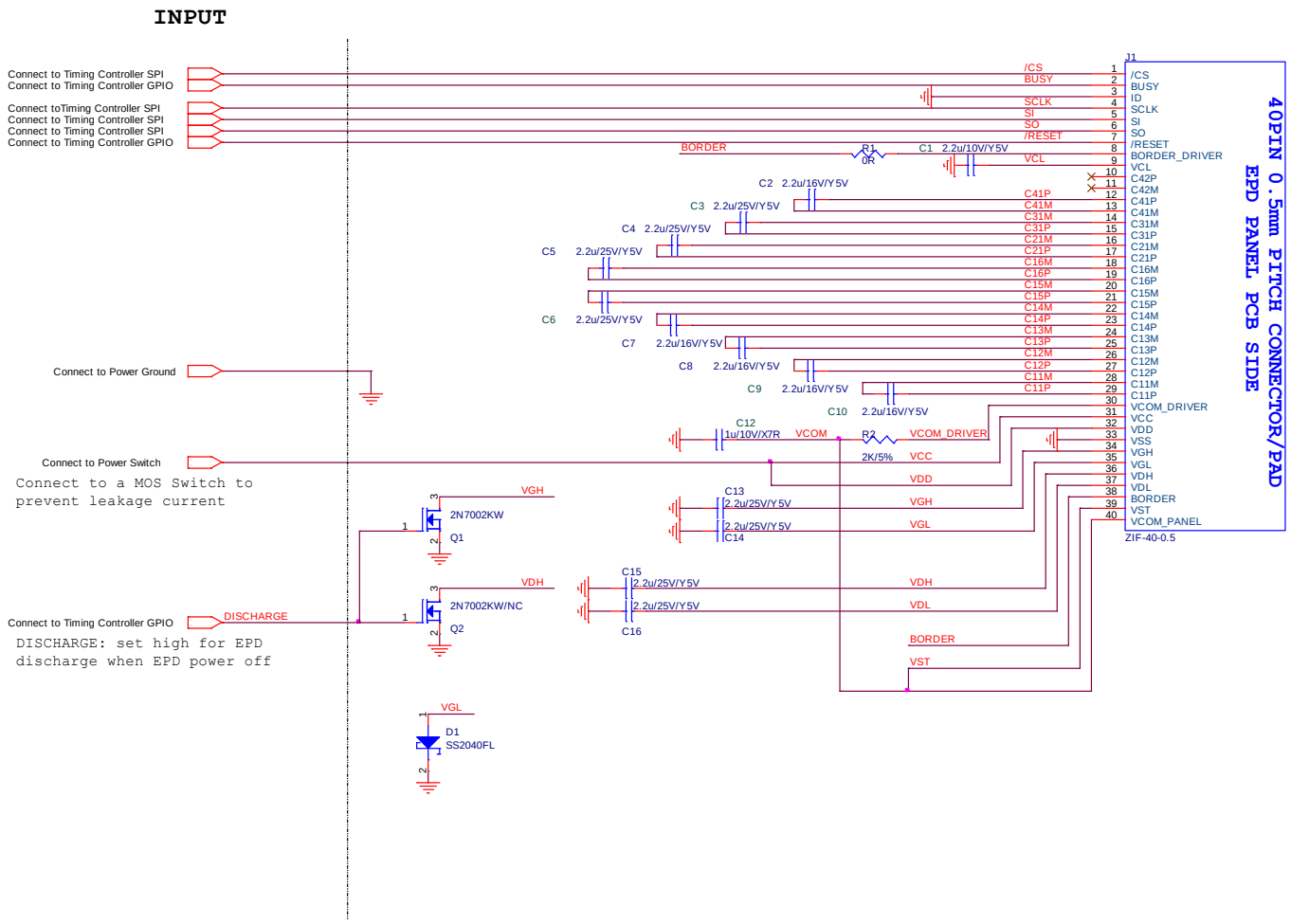
C: Capacitor

RC: Resistor and Capacitor

P: Power

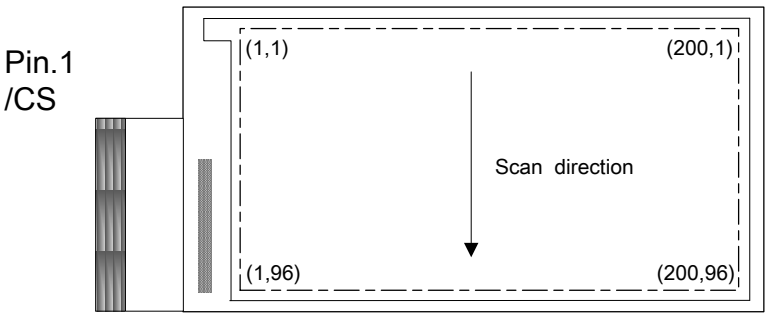
1.3 Reference Circuit

1.3.1 2 inch EPD Panel Reference Circuit

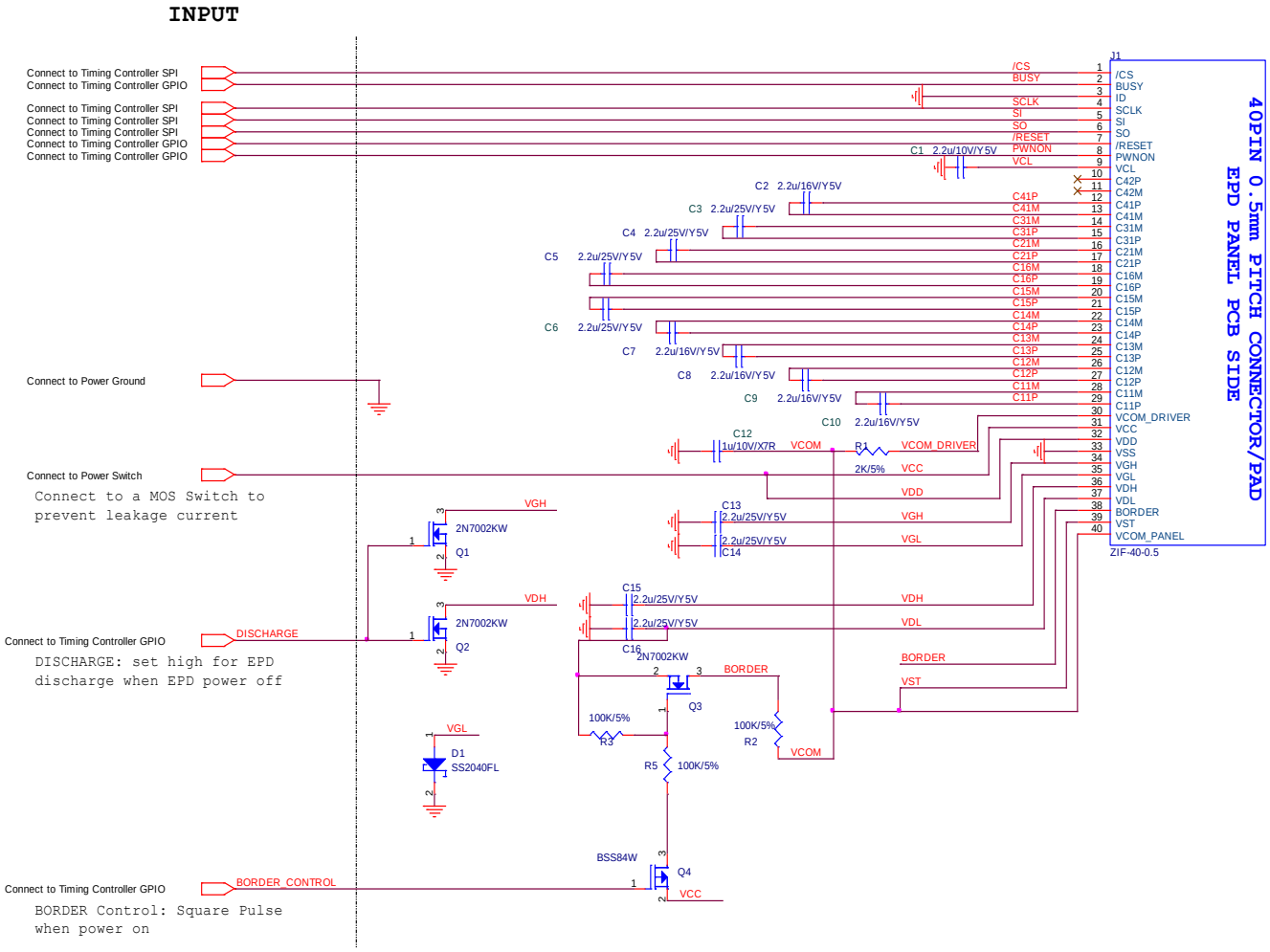


Note :

2" EPD Panel Pin.1 location

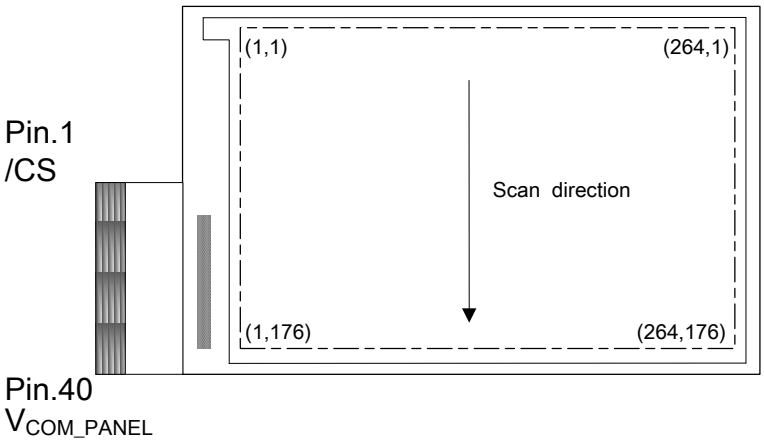


1.3.2 2.7 inch EPD Panel Reference Circuit



Note :

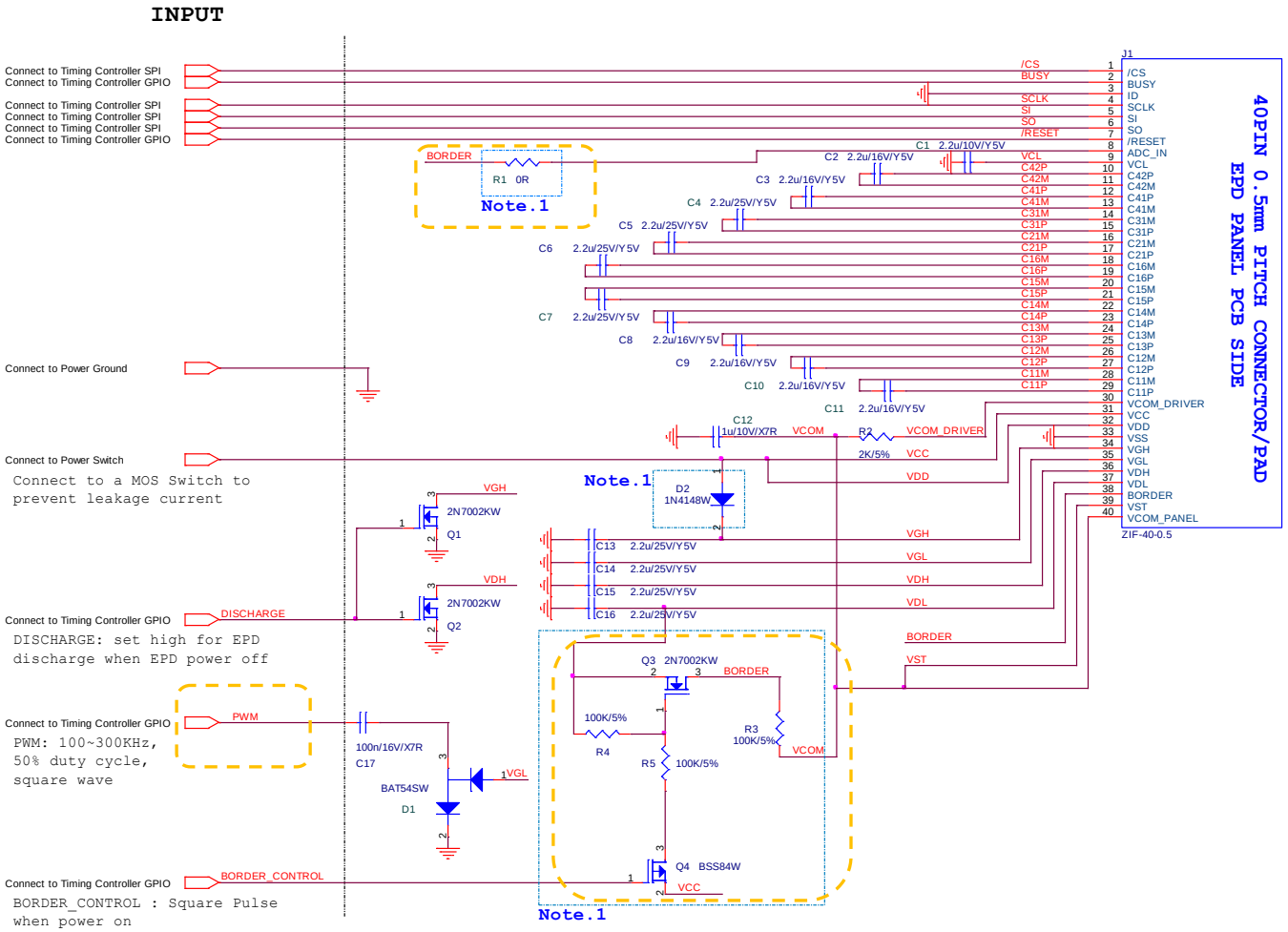
1. 2.7" EPD Panel Pin.1 location



2. Pin. 8 (PWNON), it has an internal pull-up resistor. It's ok to float it.
   (2.7" EPD Panel Only)

1.3.3 Use G1's PCBA to drive the EPD with G2 Driver IC

Below is the reference circuit for the EPD with G1 Driver IC.



Hardware setting for different size :

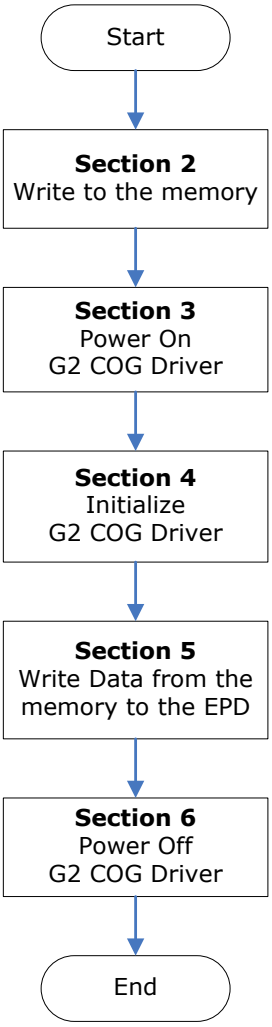
	1.44"	2" & 2.7"
R_1	Mounted	No Mounted
Q_3,Q_4 R_3,R_4,R_5	No Mounted	Mounted
D_2	Mounted	No Mounted

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 MCU GPIO pin, PWM. Keep PWM signal as either 1 or 0.
- No matter what size EPD is, use 2.7” Power Off Sequence as following section.

1.4 EPD Driving Flow Chart

The flowchart below provides an overview of the actions necessary to update the EPD. The steps below refer to the detailed descriptions in the respective sections.



1.5 Controller

The diagram below provides a signal control overview during an EPD update cycle. The diagram is divided into

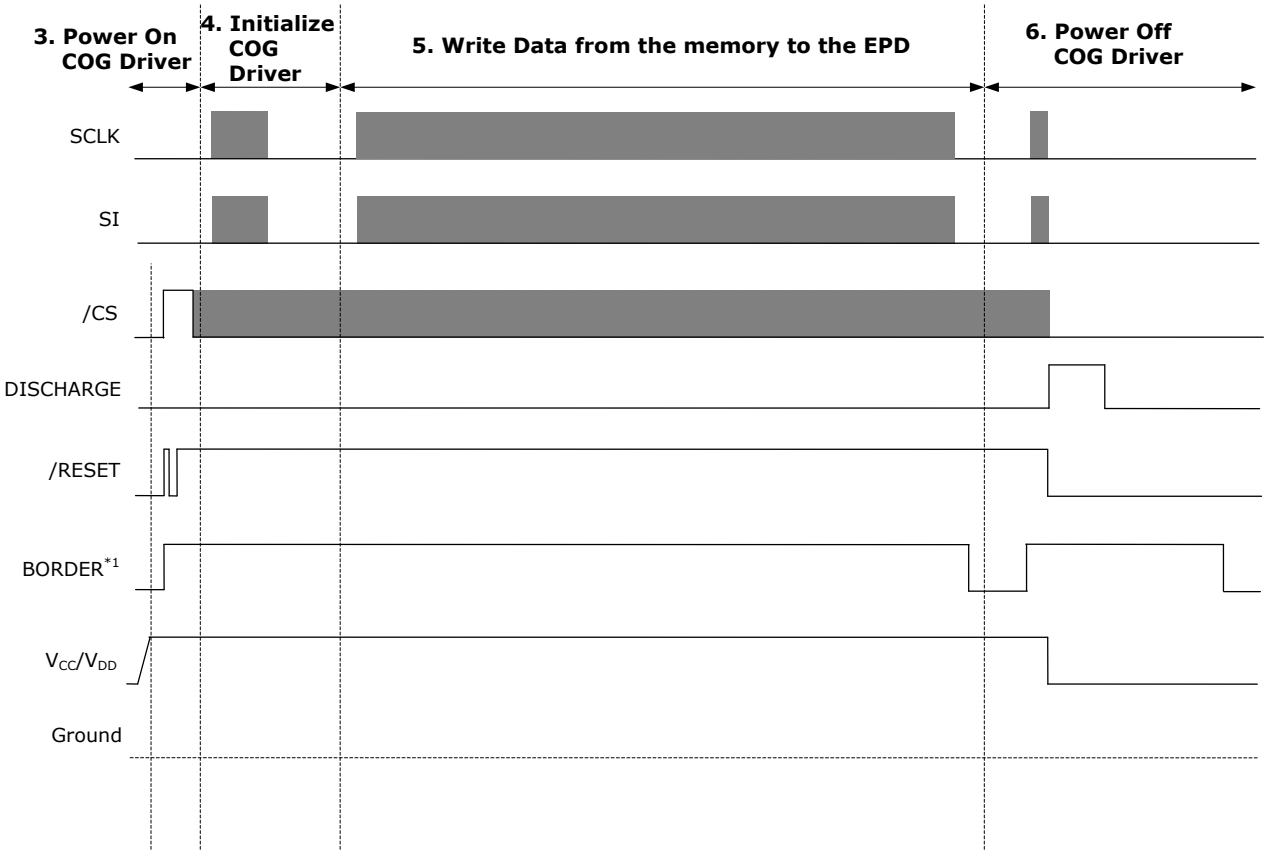
"3. Power On G2 COG Driver",

"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:
   BORDER is used to keep a sharp border while taking care of the electronic ink particles. For implement this function, Developer needs to use a MCU pin to control this signal.

   (This function is only used in 2.7" EPD Panel)

1.6 SPI Timing Format

SPI commands are used to communicate between the MCU and the G2 COG Driver. The SPI format used differs from the standard in that two way communications are not used, and CS is pulled high then low between clocks. When setting up the SPI timing, PDI recommends verifying the control signals for the overall waveform in Section 1.5, next verify the SPI command format and SPI command timing both in this section.

The maximum SPI clock speed that the G2 COG Driver can accept is 20MHz.

The SPI mode is 0.

- Below is a description of the SPI Format:

$$SPI(0xI\ ,\ 0xD_1,\ 0xD_2,\ 0xD_3,\ \ldots)$$

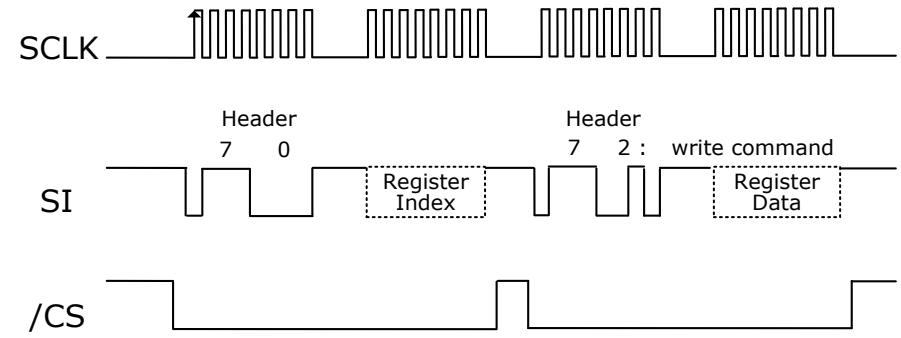
Where:

I is the Register Index and the length is 1 byte
$D_{1\sim n}$ is the Register Data. The Register Data length is 1 or 110 bytes depending on which Register Index is selected.

Register Index	Number Bytes of Register Data
0x01	8
0x02	1
0x03	1
0x04	1
0x05	1
0x07	1
0x08	1
0x09	1
0x0A	110
0x0B	1
0x0F	1

- Before sending the Register Index, the SPI (SI) must send a 0x**70** header command.
- Likewise, the SPI (SI) must send a 0x**72** is the header command prior to the Register Data. The flow chart and detailed description can be found on the next page.

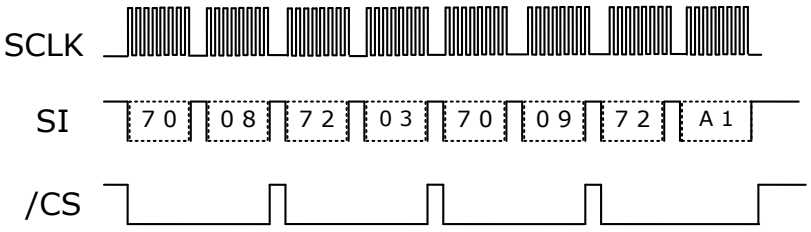
- SPI write command signals and flowchart(SPI):



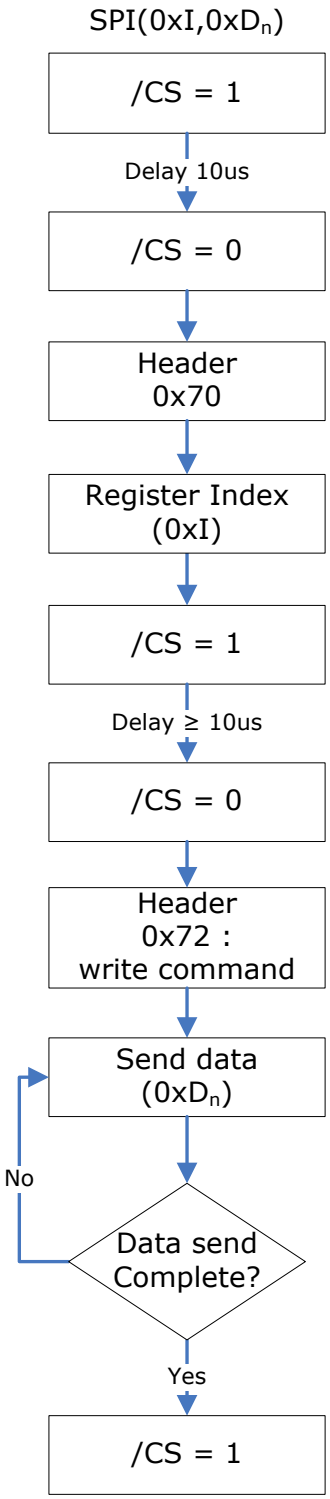
/CS must be set High then Low between Register Index and Register Data

For example:

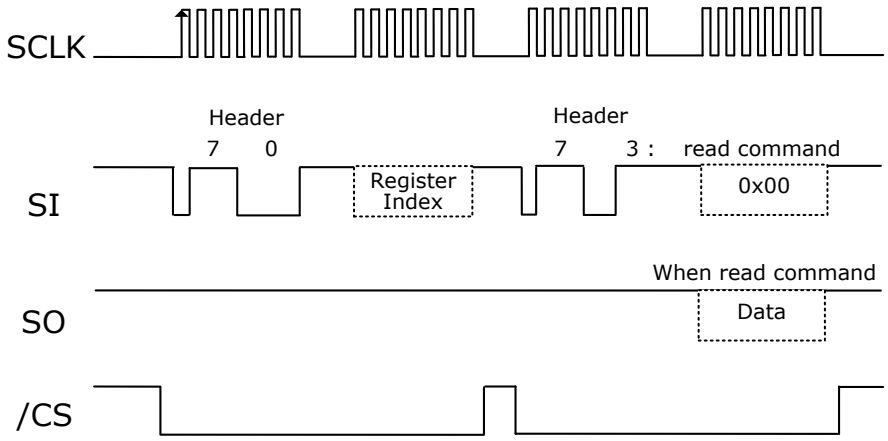
To send two continue SPI write commands:
SPI(0x08,0x03) and SPI(0x09, 0xA1)



If register data is larger than two bytes, you must input data continuously without setting Register Index again.



- SPI read command signals and flowchart(SPI_R):

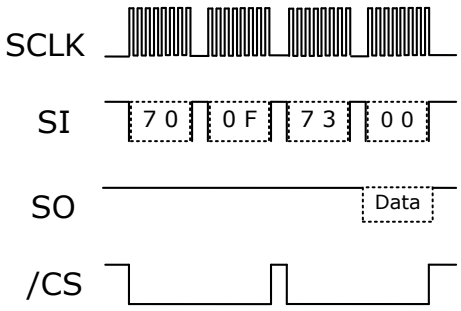


/CS must be set High then Low between Register Index and Register Data

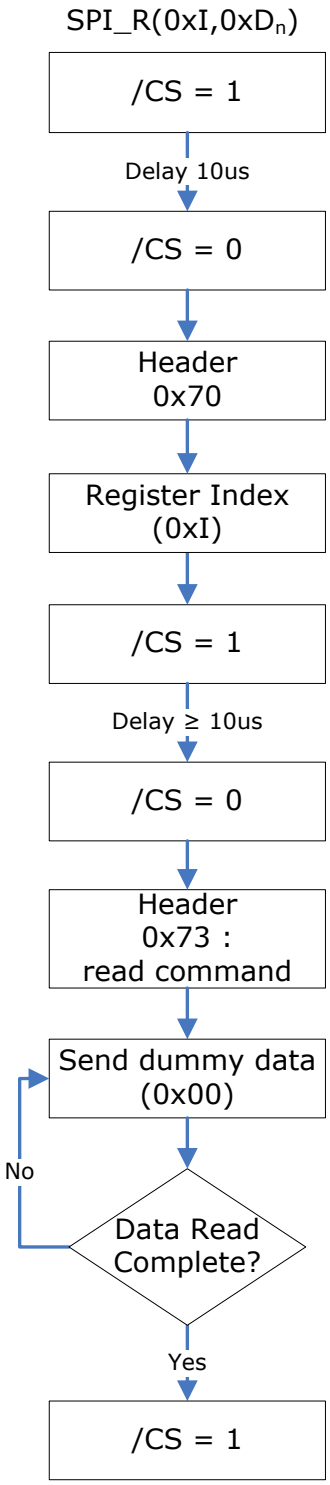
For example:

To send one SPI read command:

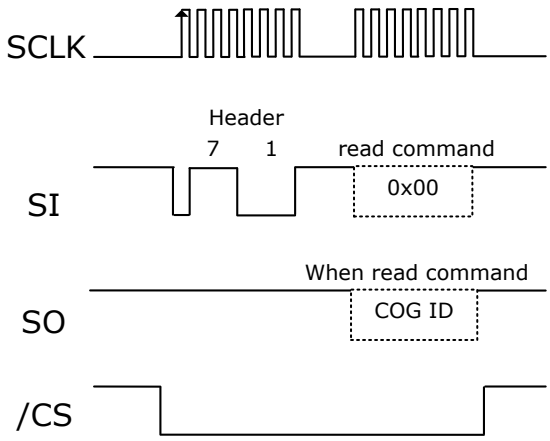
SPI_R(0x0F,0x00)

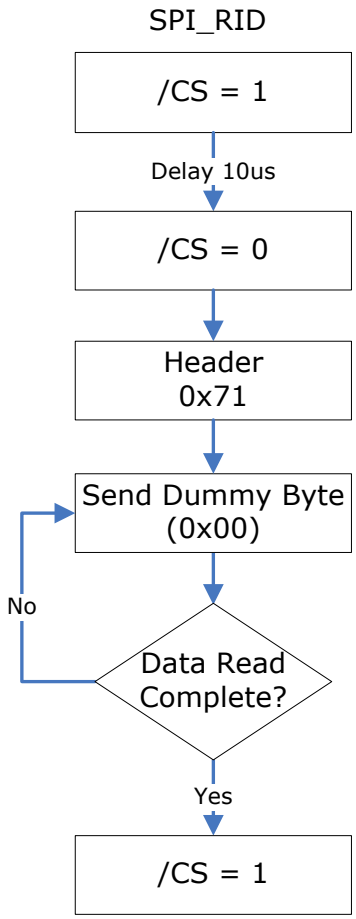


If register data is larger than two bytes, you must input data continuously without setting Register Index again.



- SPI read COG ID and flowchart(SPI_RID):

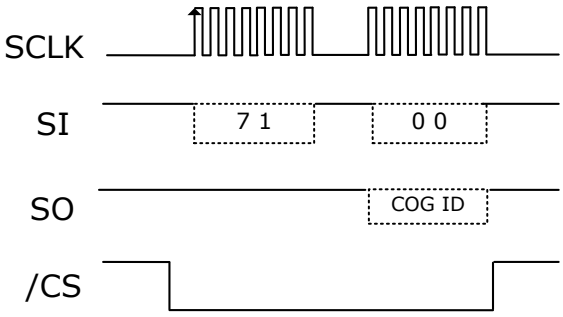




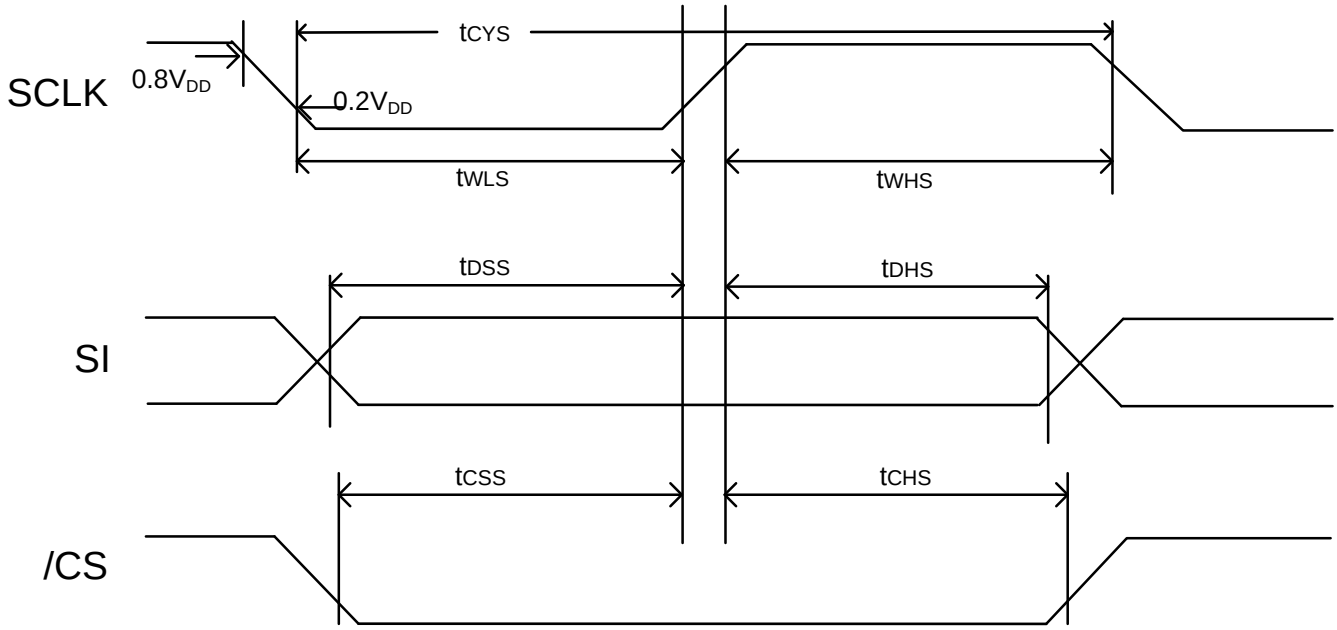
/CS must be set High then Low between Register Index and Register Data

For example:

To send one SPI read Command to COG ID(SPI_RID):



SPI command timing



VCC = 2.3 to 3.6V **Temp = -25 to 0℃**

Item	Signal	Symbol	Min.	Typ.	Max.	Unit	Remark
Serial clock cycle	SCLK	tCYS	50	-	-	ns	
SCLK high pulse width	SCLK	tWHS	25	-	-	ns	
SCLK low pulse width	SCLK	tWLS	25	-	-	ns	
Data setup time	SI	tDSS	12	-	-	ns	
Data hold time	SI	tDHS	12	-	-	ns	
CSB setup time	/CS	tCSS	12	-	-	ns	
CSB hold time	/CS	tCHS	20	-	-	ns	

2 Write to the Memory

Before powering on G2 COG Driver, the 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) in prior to writing.

One buffer space should be allocated to store new patterns. The new pattern will be written to the EPD. The table below list the buffer space size required for each EPD size.

EPD size	Image resolution(pixels)	image Buffer (bytes)
2"	200 x 96	2,400
2.7”	264 x 176	5,808

3 Power On G2 COG Driver

This flowchart describes power sequence for the COG Driver.

1. Start :

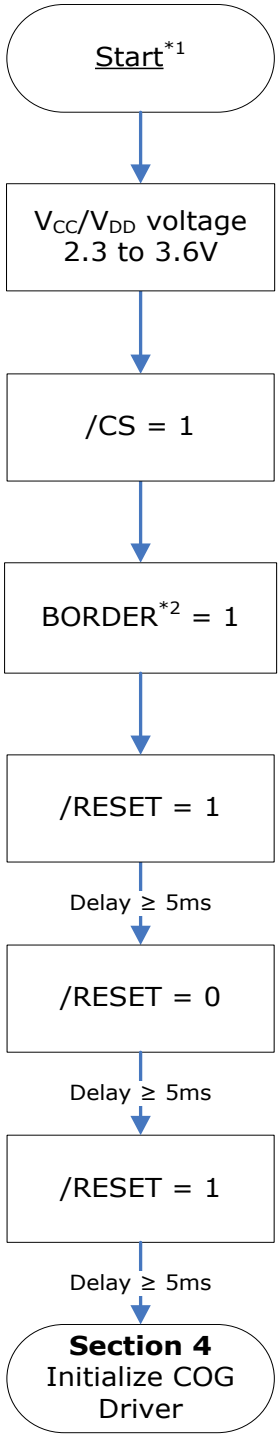
Initial State:

V_{CC}/V_{DD} = 0

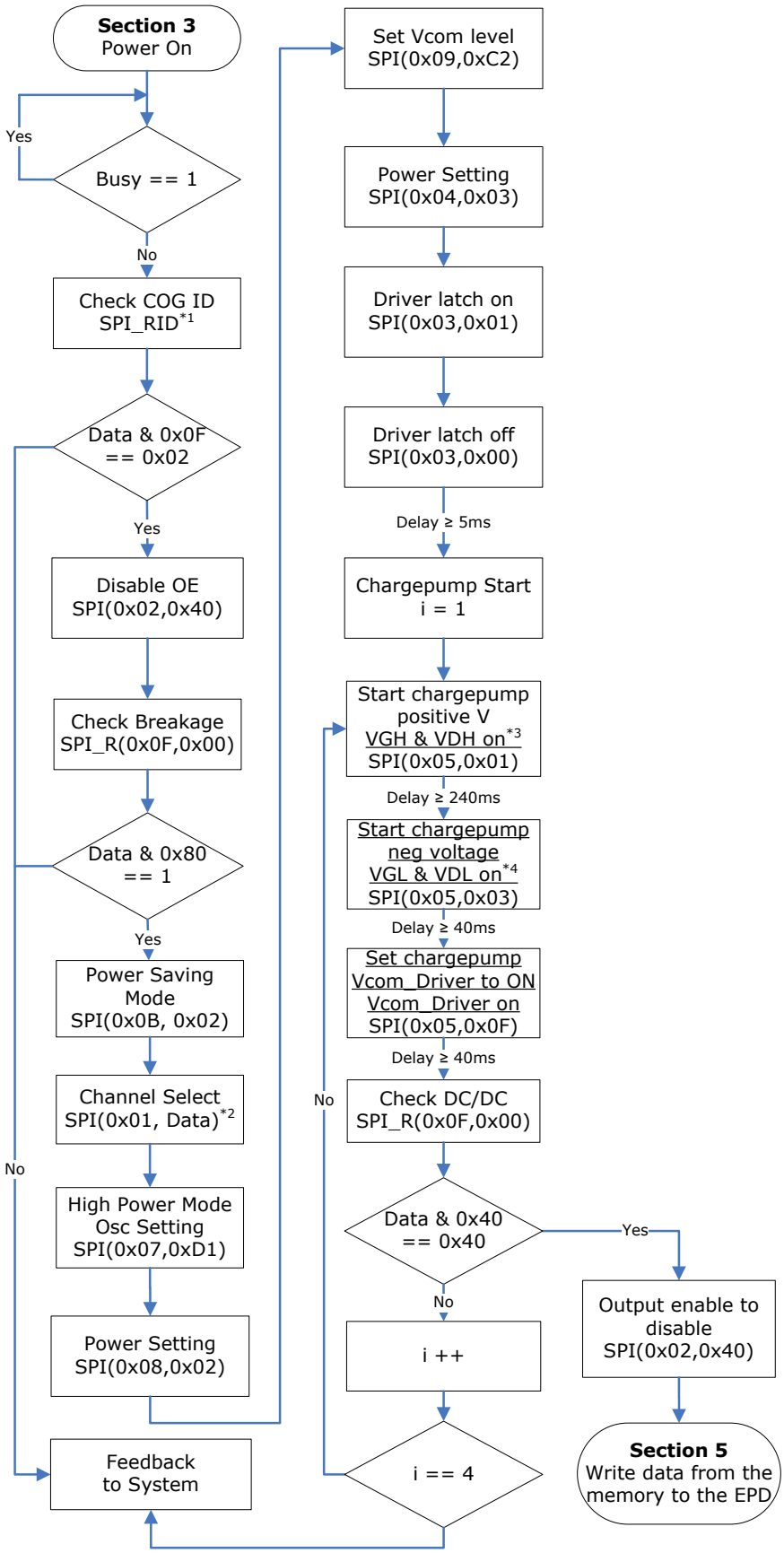
/RESET, /CS, BORDER, SI, SCLK = 0

2. BORDER:

For implement this function, Developer needs to use a MCU pin to control. /BORDER is used to keep a sharp border while taking care of the electronic ink particles.
(This function is only used in 2.7" EPD)



4 Initialize G2 COG Driver



Note:

1. SPI has two modes: SPI write command (SPI) and SPI read command (SPI_R), please refer ”1.6 SPI Timing Format” for detail. SPI_R(0x72,0x00) is used to check the COG Driver ID.
   - G1 COG Driver ID is 0x01.
   - G2 COG Driver ID is 0x02
2. SPI(0x01, Data):
   - Different by each size
     - 2”: SPI(0x01, (0x0000,0000,01FF,E000))
     - 2.7”: SPI(0x01, (0x0000,007F,FFFE,0000))
   - To send first byte protocol (0x70) before Register Index (0x01), and then send second byte protocol (0x72) before Register 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

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.

5.1 Data Structure

- EPD Resolutions

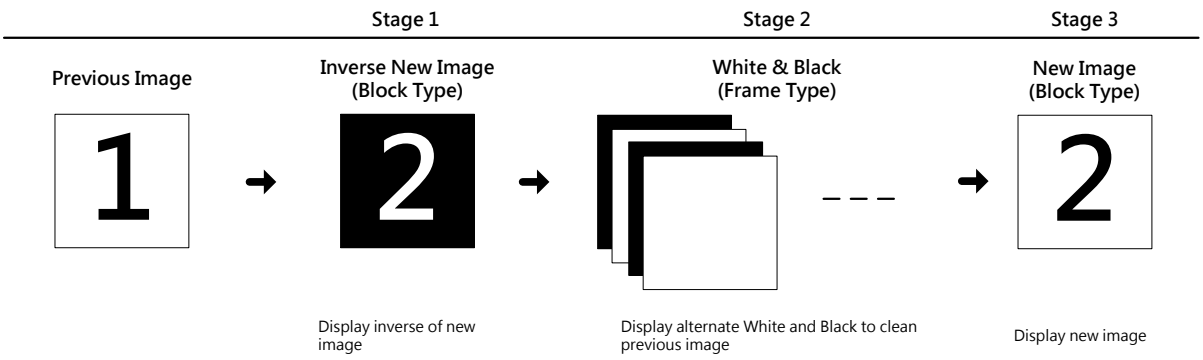
EPD size	Image resolution(pixels)	X	Y
2"	200 x 96	200	96
2.7”	264 x 176	264	176

- Data components
  - One Bit – A bit can be W (White), B (Black) or N (Nothing) bits. Using the N bit mitigates ghosting.
  - One Dot/pixel is comprised of 2 bits.
  - One line is the number of dots in a line. For example:
    - The 2” uses 200 Dots to represent 1 Line.
    - The 2.7” uses 264 Dots to represent 1 Line.
    - The G2 COG Driver uses a buffer to write one line of data (Mapping) - interlaced

Data Bytes	Scan bytes	Data Bytes
1^{st} – 25^{th} (Odd)	1^{st} - 24^{th}	26^{th} – 50^{th} (Even)
2” Example: Because method to write is interlaced, write the even data bytes for a line {D(199,y),D(197,y), D(195,y), D(193,y)} …. {D(7,y),D(5,y), D(3,y), D(1,y)}	2” Example: Write bytes for every scan line {S(96),S(95), S(94), S(93)}…. {S(4),S(3), S(2), S(1)}	2” Example: Write the odd data bytes for a line {D(2,y),D(4,y), D(6,y), D(8,y)}…{D(194,y),D(196,y), D(198,y), D(200,y)}

  - One frame of data is the number of lines * rows. For example:
    - The 2” frame of data is 96 lines * 200 dots.
    - The 2.7” frame of data is 176 lines * 264 dots.

5.2 Overall Update Flow



5.3 Store a Line of Data in the Buffer

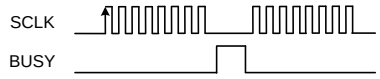
This section describes the details of how to send data to the G2 COG Driver. The G2 COG Driver uses a buffer to update the display line by line.

- 2" Input Data Order

Note :

1. When start transfer each Data Byte, users need to check BUSY pin.

Example :



2. If users cannot check BUSY pin, use delay at least 1 usec (10^{-6} second) Between byte-byte data for transfer image data.

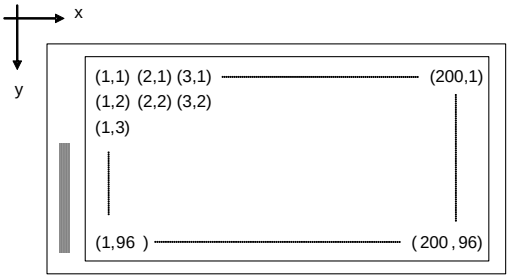
Data	bit1	bit0	Input	
D(x,y) x = 1~200 y = 1~96	1	1	Black	(B)
	1	0	White	(W)
	0	0	Nothing	(N)

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

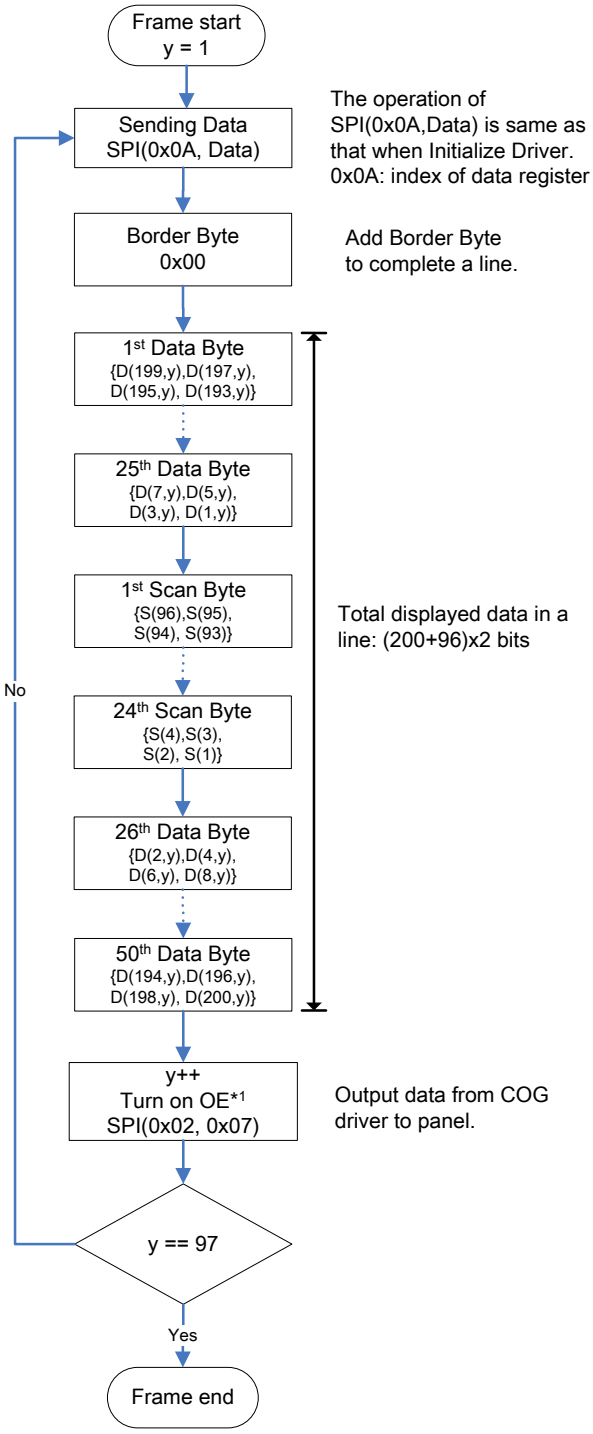
Scan	bit1	bit0	Input
S(1) ~S(96)	1	1	Scan on
	0	0	Scan off

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(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



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

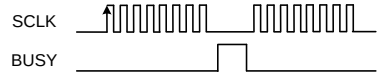


- 2.7" Input Data Order

Note :

1. When start transfer each Data Byte, users need to check BUSY pin.

Example :



2. If users cannot check BUSY pin, use delay at least 1 usec (10^{-6} second) Between byte-byte data for transfer image data.

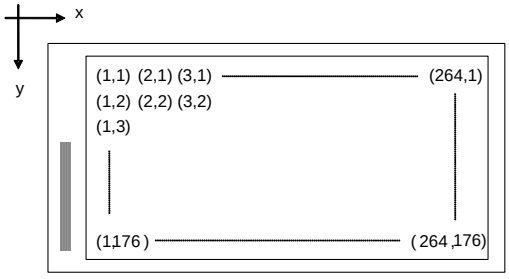
Data	bit1	bit0	Input	
D(x,y) x = 1~264 y = 1~176	1	1	Black	(B)
	1	0	White	(W)
	0	0	Nothing	(N)

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
→ 1st Data Byte= 11,10,00,11

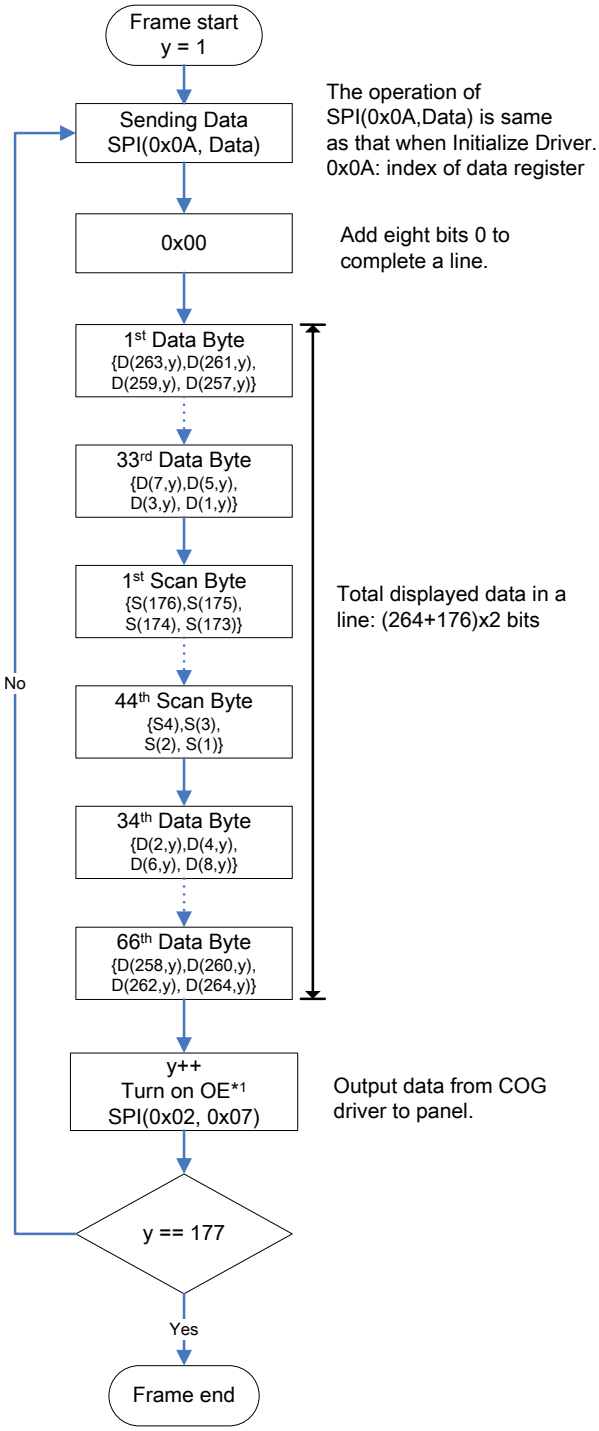
Scan	bit1	bit0	Input
S(1) ~S(176)	1	1	Scan on
	0	0	Scan off

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(264,2)).

S(1) = Scan off = 00
S(2) = Scan on = 11
S(3) = Scan off = 00
S(4) = Scan off = 00
:
S(176) = Scan off = 00
→ 1st ~ 43rd Scan Byte = 00,00,00,00
→ 44th Scan Byte = 00,00,11,00



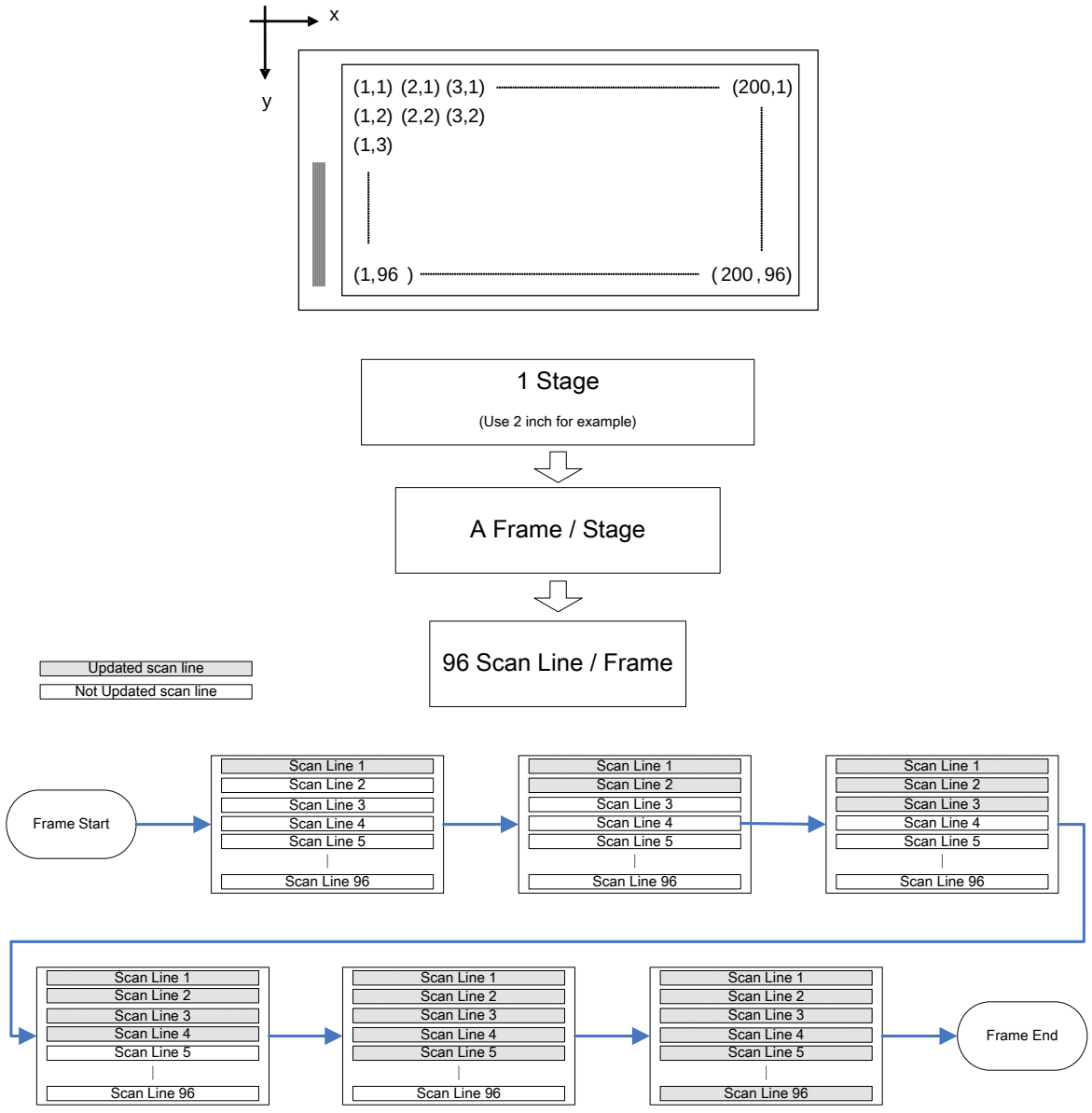
1. Turn on OE :
Output data from COG driver to panel.



5.4 Writing to the Display in Stages

This section contains the method to write to the display in stages. Rewrite the frame during each stage. There are two different ways, Block Type and Frame type, to scan the display.

5.4.1 Frame Type



Note :

Frame Type means turn on EPD scan lines line by line continuously to complete a frame. In a Frame Type Stage, you need to turn on first scan line (others scan lines are off), then turn on second scan line (the others scan lines are off) Until last scan line turn on. If you need to do more than one frame, you only do above action (i.e scanning from first line to last line) again.

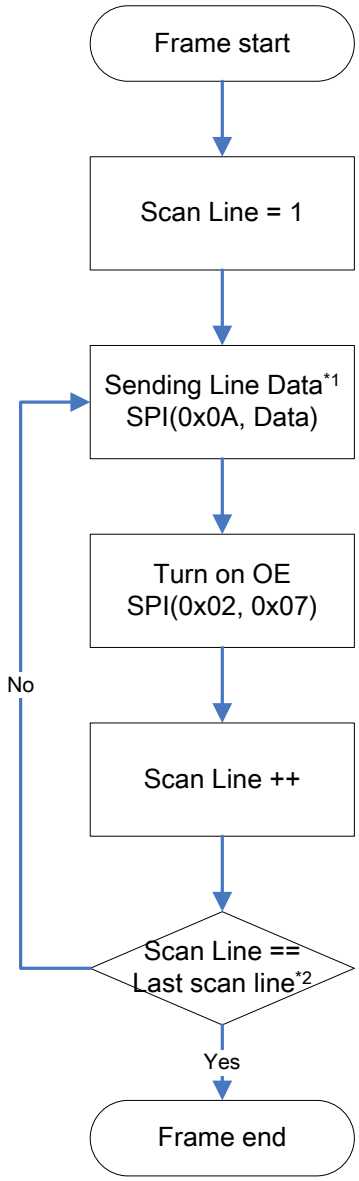
- **Frame Type Flow**

Note :

1. Line Data :
   One line data of 2" EPD Panel has 75 Bytes
   (include a Border Byte)

   One line data of 2.7" EPD Panel has 111 Bytes
   (include a Border Byte)

2. Last scan line :
   Total scan of 2" EPD Panel → 96 scan line
   Total scan of 2.7" EPD Panel → 176 scan line



5.4.2 Block Type

This type drives the portion lines of an EPD. It can reduce the refresh or recharge time of the line. Therefore need to shift the portion lines (*Block*) in a fixed number of lines (*Step*) until finish all lines (*Frame number*). The last Frame needs to drive "Nothing" at the last *Step* in each *Block*. The Nothing is included in *Block*. In this type of scan, you need to decide three variables. *Frame number*, *Block* and *Step*, use these three variables to complete a stage.

***Frame number*:**

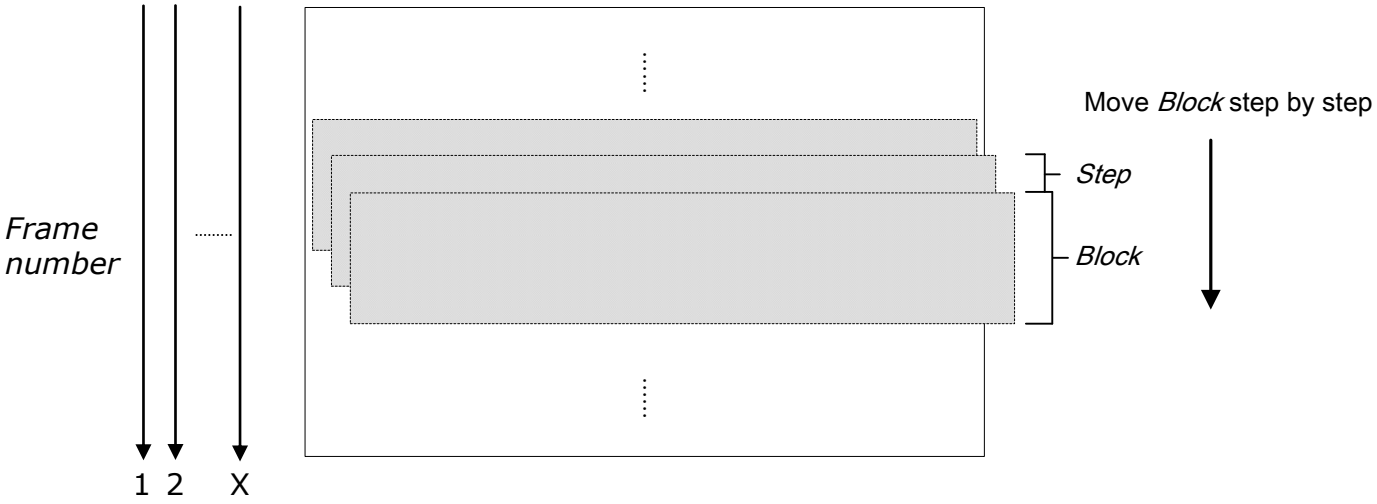
Frame number is number of *Block* moving from top to bottom of display. The last frame needs to drive nothing at the last *Step* in each *Block*.

***Block*:**

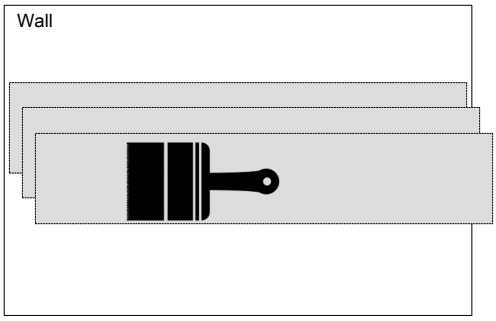
Block is a portion of total lines of EPD. It can decide the refresh or recharge time of each line. A smaller *Block* gets the fast refresh or recharge time of each line, whereas a bigger *Block* would be slower. And the number of *Block* should be divisible by the number of *Step*.

***Step*:**

Step means the *Block* will be shifted how many lines. It decides each line will be scanned how many times. A smaller *Step* gets more scan times, whereas a bigger *Step* would be less. And the number of total lines should be divisible by the number of *Step*.



Block Type update is similar to painting a wall by a brush. The width of brush is *Block*. Every time, you paint horizontally then move down certain distance, *Step.* Then paint again until finish the whole wall. If the painting is not thick enough, you will repeat above action again until getting thick enough painting. The repetition is *Frame*.



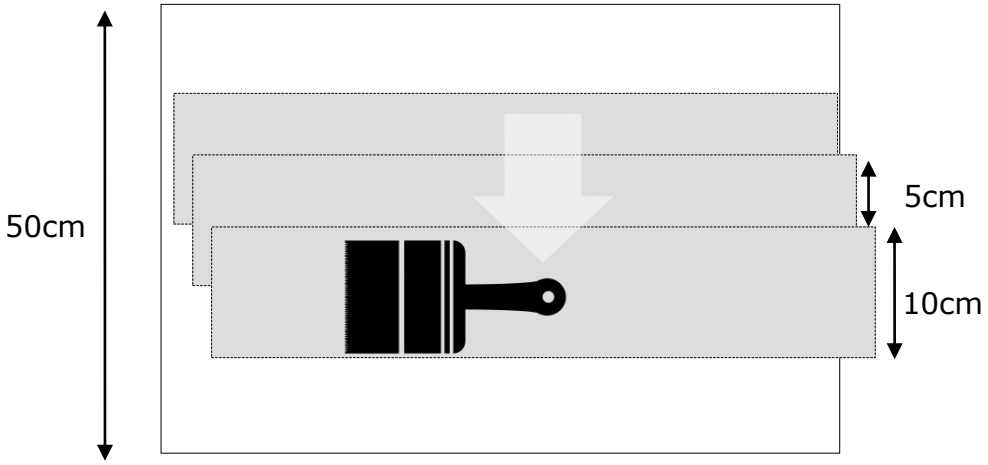
Example:

A 50cm height wall, a 10cm width brush, we move down brush in 5cm step. We must ensure the painting on the whole wall is same thickness.

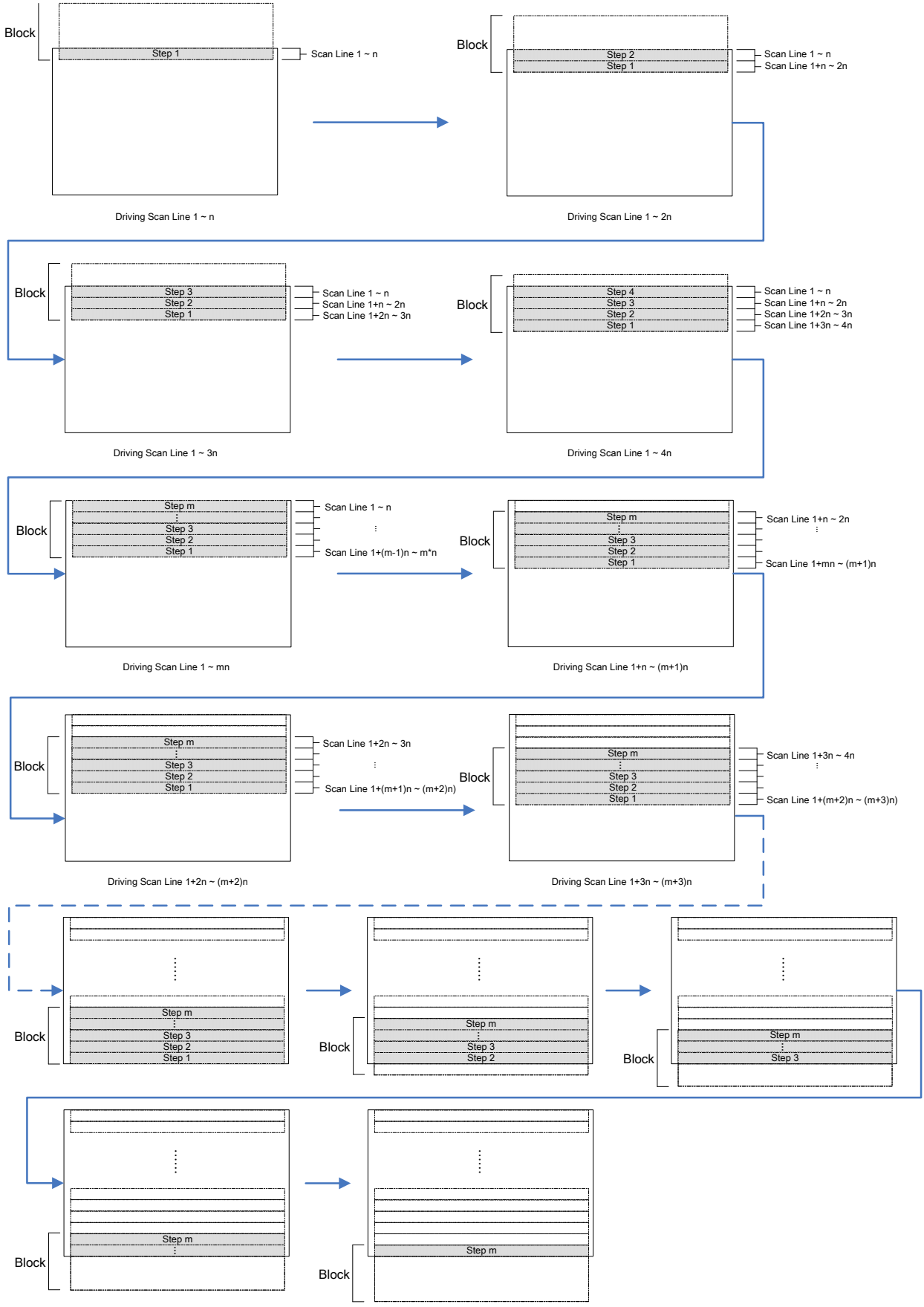
We paint the wall by this order.

Top 0~5cm → 0~10cm → 5~15cm → … → 40~50cm → 45~50cm (i.e. bottom 0~5cm).

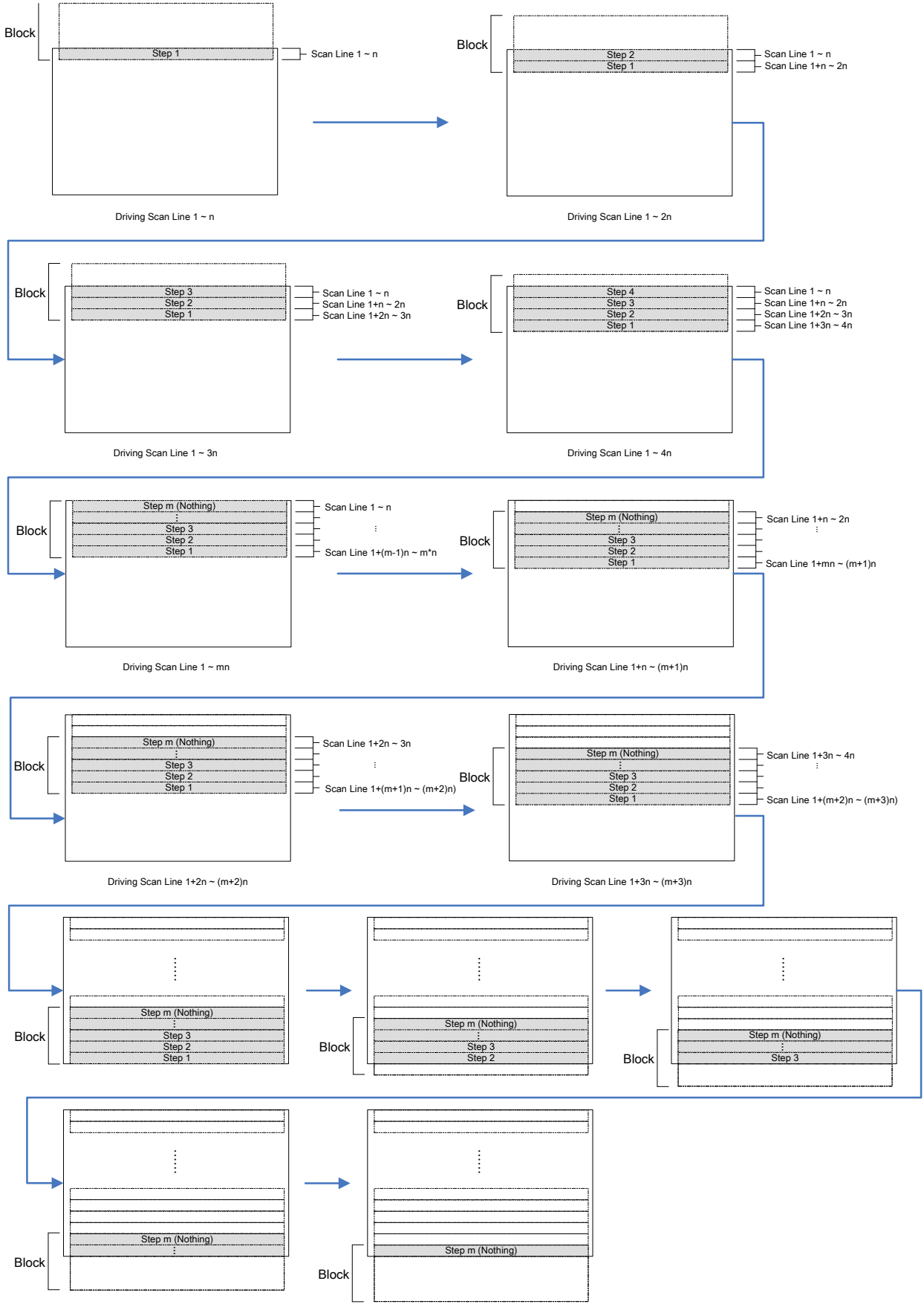
In this case, every cm of this wall is painted twice. If twice is not enough. We can repeat this top to bottom procedure one more time to get sufficient thickness.



- **Frame 1 ~ (X-1)**



- **Frame X (i.e. Last Frame)**



- **Block Type Flow**

Note :

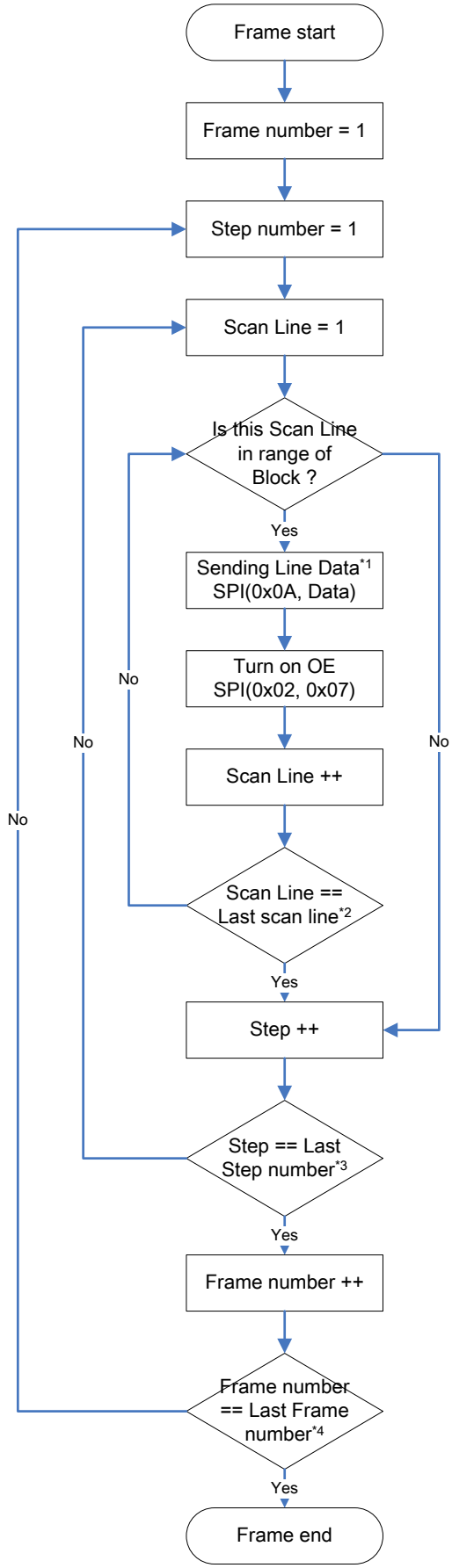
1. Line Data :
   One line data of 2" EPD Panel has 75 Bytes
   (include a Border Byte)

   One line data of 2.7" EPD Panel has 111 Bytes
   (include a Border Byte)

2. Last scan line :
   Total scan of 2" EPD Panel → 96 scan line
   Total scan of 2.7" EPD Panel → 176 scan line

3. Last Step number :
   Total Step number is determine by Step and Block which user set.

4. Last Frame number :
   Total Step number is determine by user st.



- **Method to Select a suitable *Frame/Block/Step***

1. Measure Line Time of your TCon.

   Line Time: The time needed to finish the update of one scan line. i.e. The time to finish the flowchart in "Store a Line of Data in the Buffer".

   Typically, Line Time is 0.3~1ms.

2. Determine the suitable width of *Block* (i.e. # of scan line in a *Block*)

   Width of *Block* = Block Time / Line time

   Typically, **Block Time = 5~25ms**. 10ms is suggested in most of cases.
   If Line Time = 0.5ms
   We can set
   Width of *Block* = 10ms / 0.5ms = 20 scan lines

3. Determine width of Step

   Width of *Block* is multiple of width of *Step*.
   Typically, **width of *Step* can be 2, 3, or 4 scan lines.**

4. Determine *Frame*
   In one frame, the time every scan line driven is

   Block time * Width of *Block*/ Width of *Step*

   If Block time = 10ms, Width of Block = 20, Width of Step = 2.

   10ms * 20 / 2 = 100ms

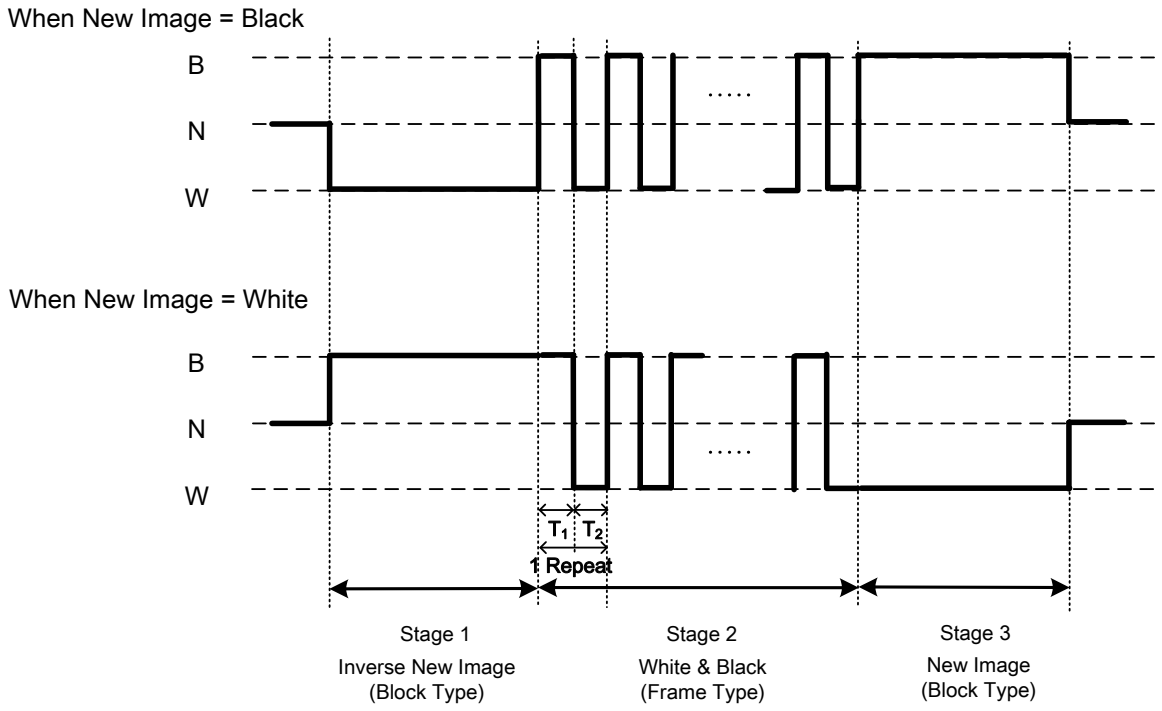
   In typical case, at room temperature, **the total time every scan line driven should be 200~400ms.** So in this example,

   Frame = 200~400ms / 100ms = 2~4.

   The larger *Frame*, the higher Contrast Ratio EPD has.

   Next page is the setting of PDi Jig.

Driving Waveform and Temperature Factor



Temperature Range (℃)	Panel Size	Stage 1			Stage 2			Stage 3			Total time (sec)
		Frame number (frame)	Block time (ms)	Step (Scan line)	T_1 (ms)	T_2 (ms)	Repeat	Frame number (frame)	Block time (ms)	Step (Scan line)	
50 ≧ T > 40	2.7 inch	4	7.4	2	196	196	4	4	7.4	2	5
	2 inch	4	22	2	196	196	4	4	22	2	5
40 ≧ T > 10	2.7 inch	2	7.4	6	196	196	4	2	7.4	6	2.5
	2 inch	2	22	6	196	196	4	2	22	6	2.5
10 ≧ T ≧ 0	2.7 inch	2	7.4	6	392	392	4	2	7.4	6	4.1
	2 inch	2	22	6	392	392	4	2	22	6	4.1

Note:

1: Line time of PDi Jig is 0.462ms. (@2.7 inch EPD Panel)

2: Detailed temperature guaranteed operation range is listed in PDI EPD datasheets. The temperature range listed in this document is only for timing controller programming reference.

3: This table is tested with PDI jig.

6 Power off G2 COG Driver

2.0" Power Off Sequence

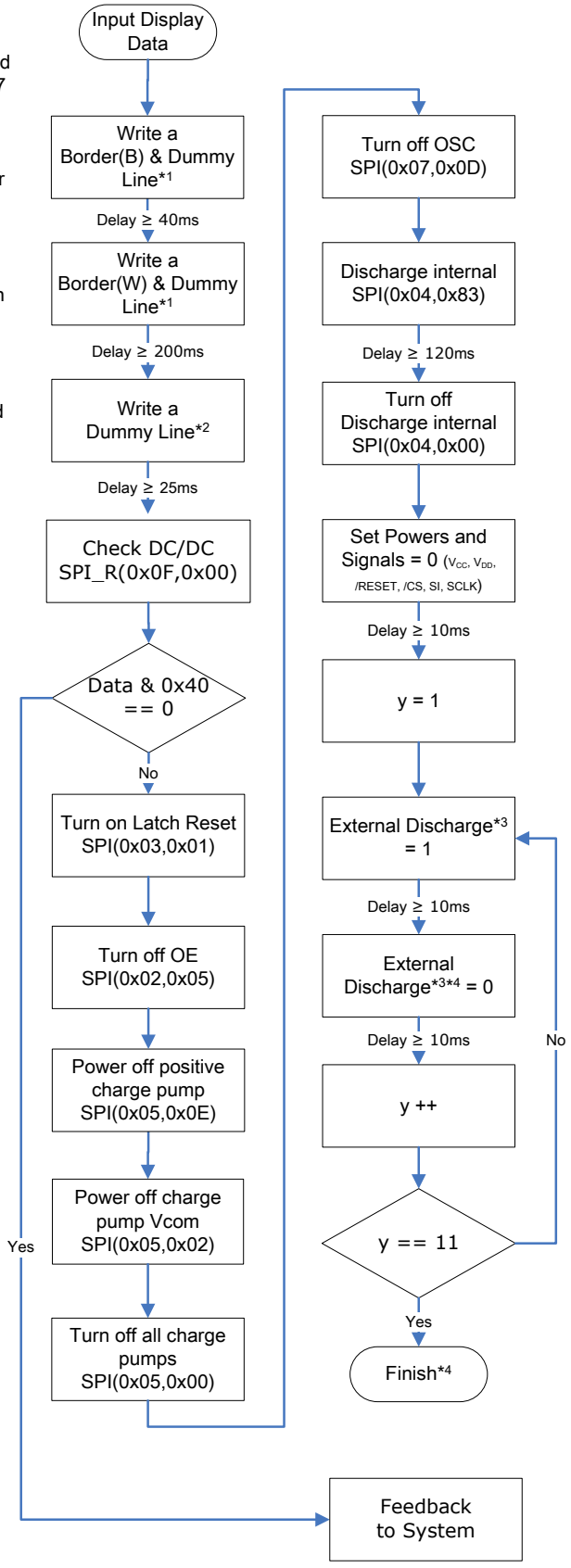
1. Border(Input) & Dummy Line :
   Set Border Byte = 0xFF to write black(B) and Border Byte = 0xAA to write white(W). A Border and Dummy line whose detail flow be shown on page 37 & 38.

2. Dummy Line :
   A line whose all Data Bytes, Scan Bytes and Border Byte are 0x00 and turn on OE. Clear the register data before power off.

3. External Discharge :
   For implement this function, users need to use a pin from Microcontroller to control. This is important to avoid vertical lines.

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

5. After finish whole update, power and signals status as below:
   V_{CC}/V_{DD} =0V.
   /RESET, /CS = 0.
   SI, SCLK = 0.



2.7" Power Off Sequence

1. BORDER :
   For implement this function, users need to use a pin to control from Microcontroller. When = 0, the BORDER is ON and write to white. When = 1,the BORDER is OFF. The reason for using BORDER is to keep a sharp border and not have a charge on the Eink particles . Voltage too long on these will produce a gray effect which is the optimal for long term operation. BORDER is needed in film.
   For 2.7", need to use this function.

2. External Discharge :
   For implement this function, users need to use a pin from Microcontroller to control. This is important to avoid vertical lines.

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

4. After finish whole update, power and signals status as below:
   V_{CC}/V_{DD} = 0.
   /RESET, /CS, BORDER*1 = 0.
   SI, SCLK = 0.