AN11128

PTN3460 Programming Guide

Rev. 1.3 - 8/3/12

Document information

Info	Content
Keywords	PTN3460, DisplayPort, eDP, LVDS, bridge
Abstract	This document describes the PTN3460 programming guide which is the supporting document for the DisplayPort to LVDS bridge device PTN3460

AN11128

PTN3460 Programming Guide

Rev. 1.3 - 8/3/12

Revision history

Rev	Date	Description
v.1	20111215	Application note; initial version.
v.1.1	20120320	Revise
v.1.2	20120331	Revise
v.1.3	20120803	Revise

AN11128
PTN3460 Programming Guide

Rev. 1.3 - 8/3/12

1. Introduction

PTN3460 is built in a configuration table in internal SRAM, which allows user to program seven EDID and 128 configuration registers through M/S I2C-bus. It can change PTN3460 functionality by changing and storing the configuration table. This application note provides detailed explanations for the configuration table, programming method, and how to program EDID and configuration setting into PTN3460 internal flash.

2. Configuration table

2.1. Configuration table in internal SRAM memory

The configuration table is stored in the SRAM of the PTN3460 and is 1K bytes (1024 bytes) long. It consists 896 bytes to store seven EDID (128 × 7 = 896 bytes) and 128 bytes of configuration registers used to control PTN3460 functionality. This configuration table is located at a fixed address in XDATA memory (x:0x0C00 - -0x0F00) of the PTN3460.

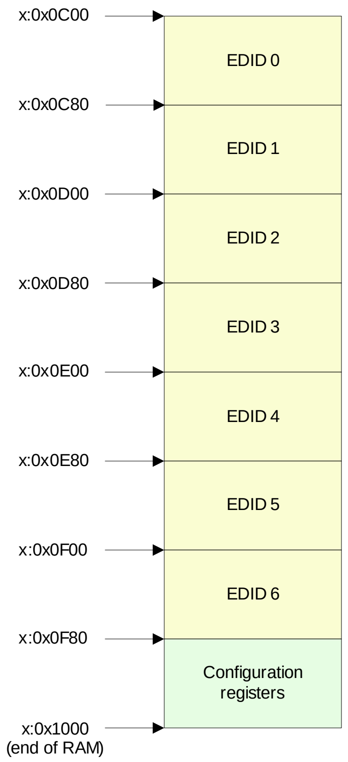


Fig 1. Configuration table location in XDATA memory

2.2. Configuration table initialization

PTN3460 offers two ways to initialize the configuration table

- Read it from internal flash memory.
- Read it from an external EEPROM.

PTN3460 firmware will first check DEV_CFG pin (pin 12 of PTN3460) status during initialization stage, and it will read 1K bytes configuration table from external EEPROM if DEV_CFG pin is high, otherwise it will read 1K bytes configuration table from internal flash memory.

PTN3460 firmware uses all zero to initialize configuration table in SRAM in following two cases

- If DEV_CFG pin = high, but no external EEPROM is existed.
- If DEV_CFG pin = open or low, but magic number in configuration table in internal flash is incorrect.

2.3. Configuration table access through I2C bus

• PTN3460 is in I2C slave mode (DEV_CFG pin = open or low)

The configuration table is a 1024 bytes table which consists of 7 EDID and 128 bytes of configuration registers. However, PTN3460 supports 8 bit I2C addressing mode which allows external MCU or PCH to access entire 1024 bytes of configuration table through M/S I2C interface of PTN3460.

When accessing PTN3460 M/S I2C as an I2C slave, it can only be addressed up to 256 bytes with standard I2C protocols, so the layout of the configuration table is modified by firmware to fit in these 256 bytes. The 256 bytes configuration table is shown in Fig 2, the Offset 0 - 255 means I2C register address 0x00 – 0xFF.

If an external MCU or PCH wants to access EDID2 which the data is physically located in PTN3460 internal SRAM location 0x0D00 – 0x0D80, then it needs to set configuration register 0x85[2:0] = 0x03 (EDID ROM access control register) and reads 128 byte EDID data from I2C register 0x00 – 0x7F over PTN3460 M/S I2C bus. And firmware will move EDID2 data from internal SRAM location 0x0D00 – 0x0D80 to I2C register 0x00 – 0x7F while external MCU or PCH access to it.

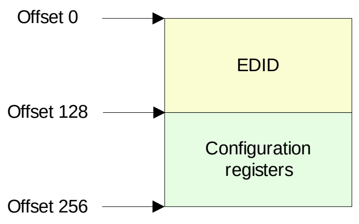


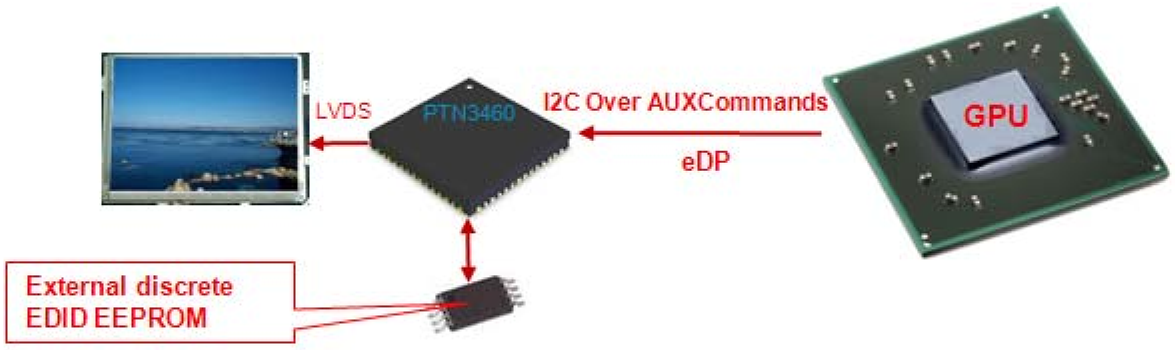
Fig 2. Configuration table layout modified by firmware to fit 256 bytes

PTN3460 M/S I2C slave address is set 0xC0 while DEV_CFG pin is set to open, and set to 0x40 while DEV_CFG pin is set to low.

PTN3460 M/S I2C pins are located at pin 24 (MS_SDA) and pin 25 (MS_SCL).

- **PTN3460 is in I2C master mode (DEV_CFG pin = high)**

PTN3460 I2C master mode application block diagram is shown as figure below.



PTN3460 reads 256 bytes of configuration table from external EEPROM when it's set as I2C master mode (DEV_CFG pin = 1), the configuration table includes 128 bytes EDID and 128 bytes of configuration register bytes.

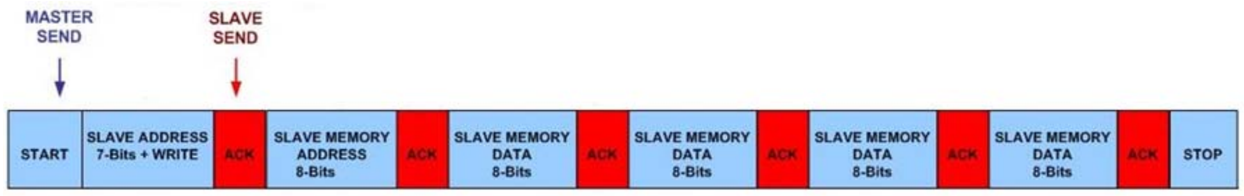
When PTN3460 is in I2C master mode, it reads 256 bytes configuration table from external EEPROM to internal SRAM in system initialization phase. It only supports one EDID storing in external EEPROM, however the system designer can change the EDID in external EEPROM easier by using EEPROM programmer to program it.

PTN3460 I2C slave mode function is disabled when it is set to I2C master mode, so PTN3460 would not be able listening to I2C command from PCH or external MCU which means that the accessing to PTN3460 I2C registers would not be supported through SMBus. The functional control such as brightness control and backlight control would be required through AUX commands instead of SMBus commands.

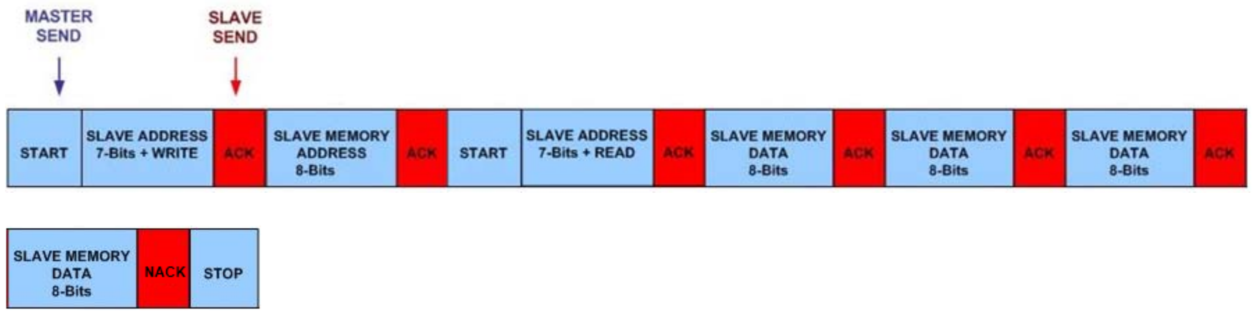
PTN3460 I2C master mode application only requires 256 bytes space in external EEPROM which the I2C slave address is 0xA0, the rest space of EEPROM can be shared with other device in application.

- **I2C Programming Protocol**

I2C master (PCH or MCU) WSRITES to I2C salve (PTN3460) protocol



I2C master (PCH or MCU) READS to I2C slave (PTN3460) protocol



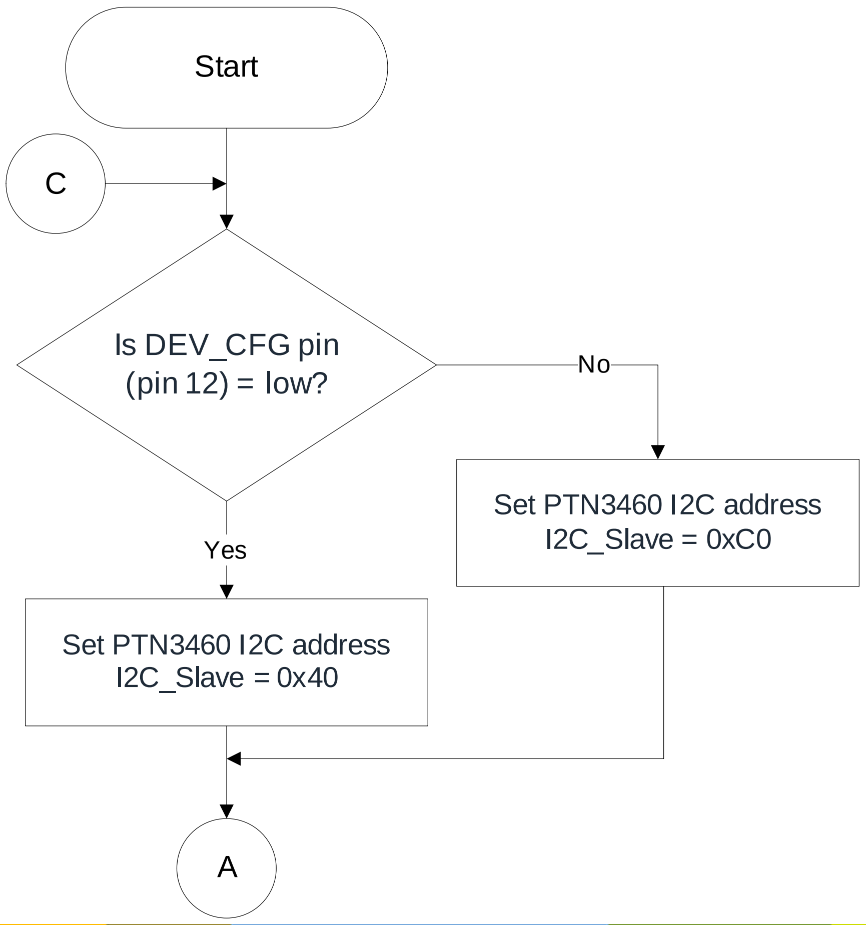
3. EDID Table

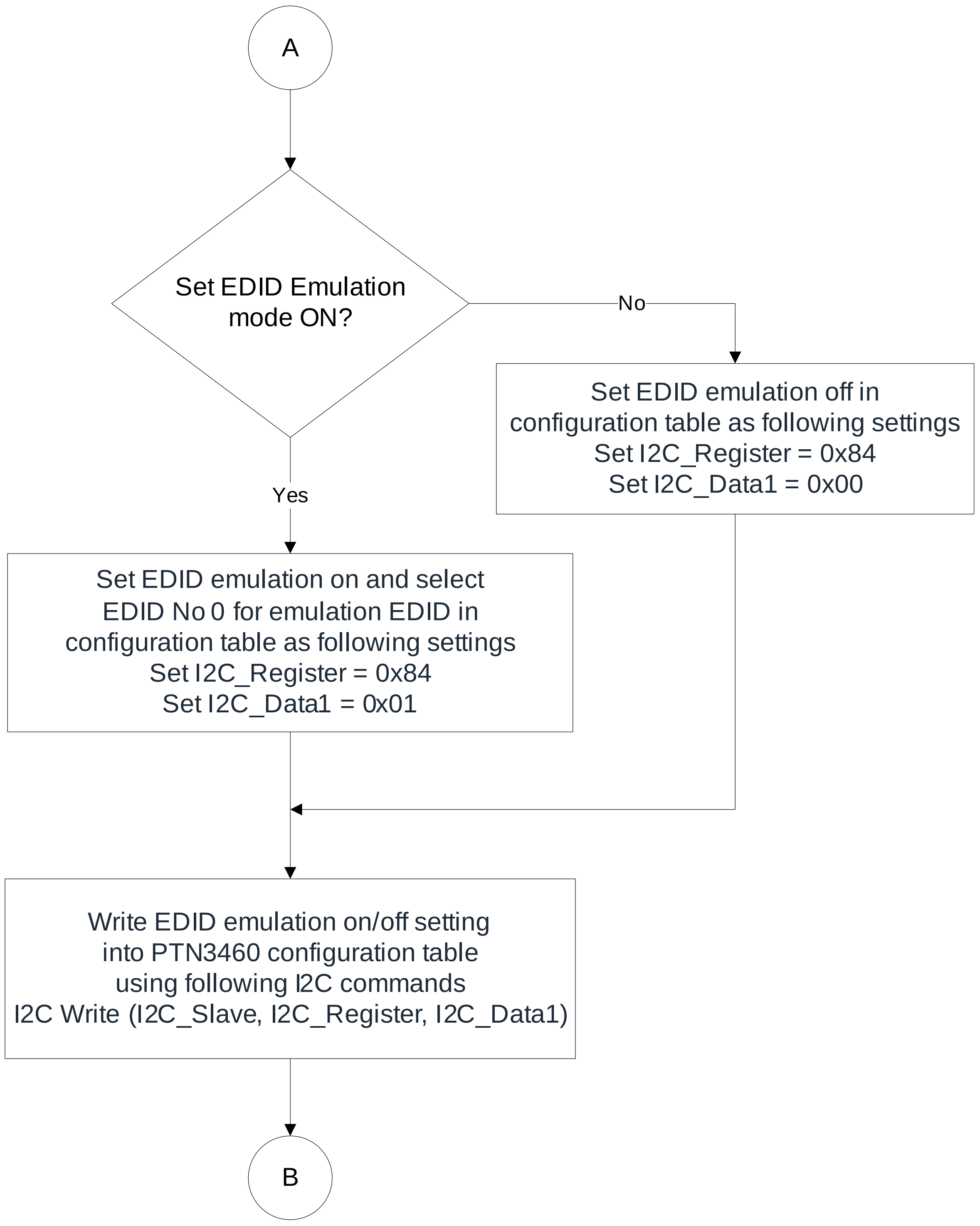
PTN3460 stores 7 EDID in configuration table in SRAM shown as fig 3.

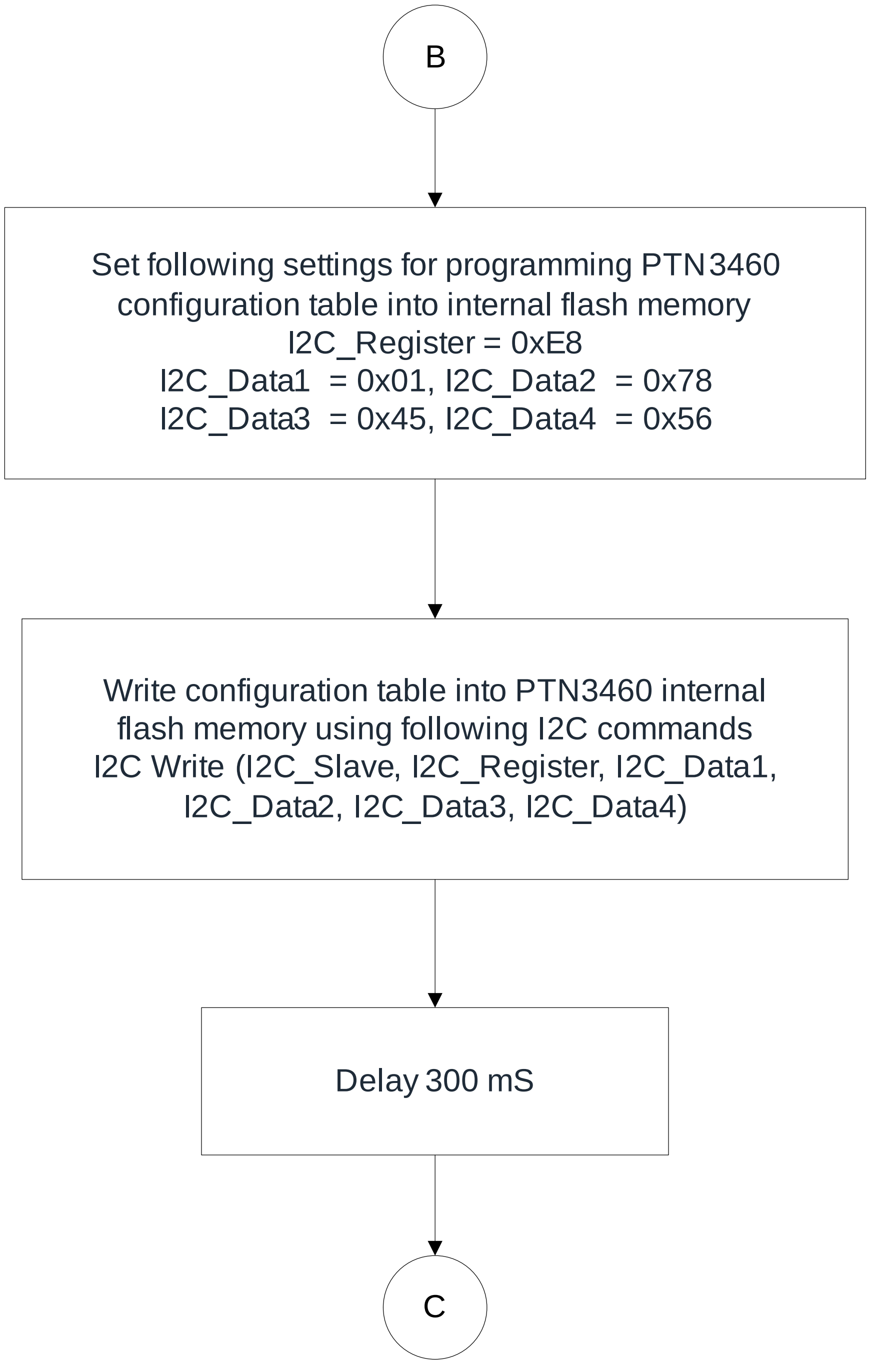
EDID N0	Resolution	EDID Description
0	1024 x 768 @60Hz	NXP Generic
1	1920 x 1080 @60Hz	NXP Generic
2	1920 x 1080 @60Hz	NXP Generic
3	1600 x 900 @60Hz	Samsung LTM200KT
4	1920 x 1080 @60Hz	Samsung LTM230HT
5	1366 x 768 @60Hz	NXP Generic
6	1600 x 900 @60Hz	ChiMei M215HGE

4. Programming Flow Chart

4.1. EDID Emulation ON/OFF Setting Control Flow

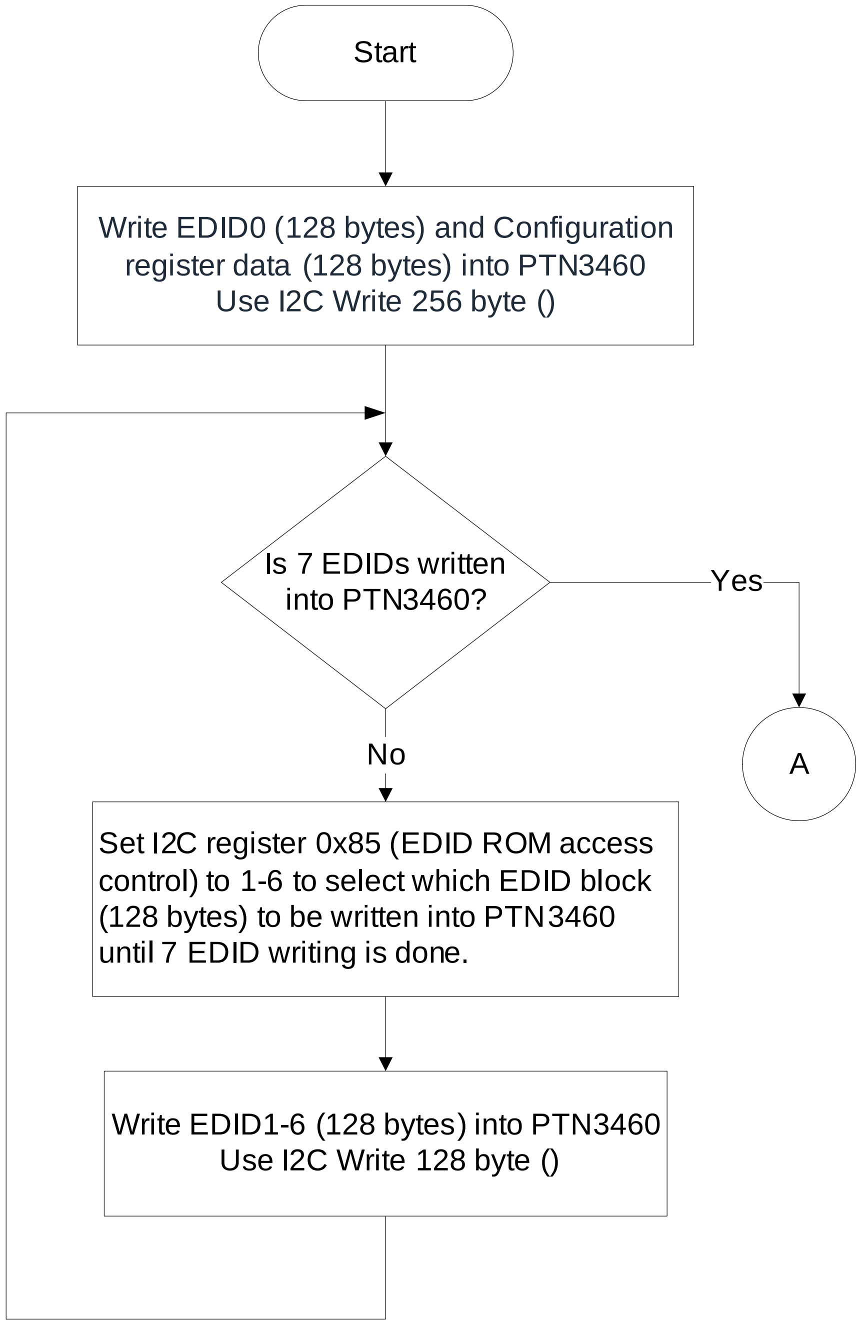

Start
C
Is DEV_CFG pin (pin 12) = low?
No
Yes
Set PTN3460 I2C address I2C_Slave = 0xC0
Set PTN3460 I2C address I2C_Slave = 0x40
A

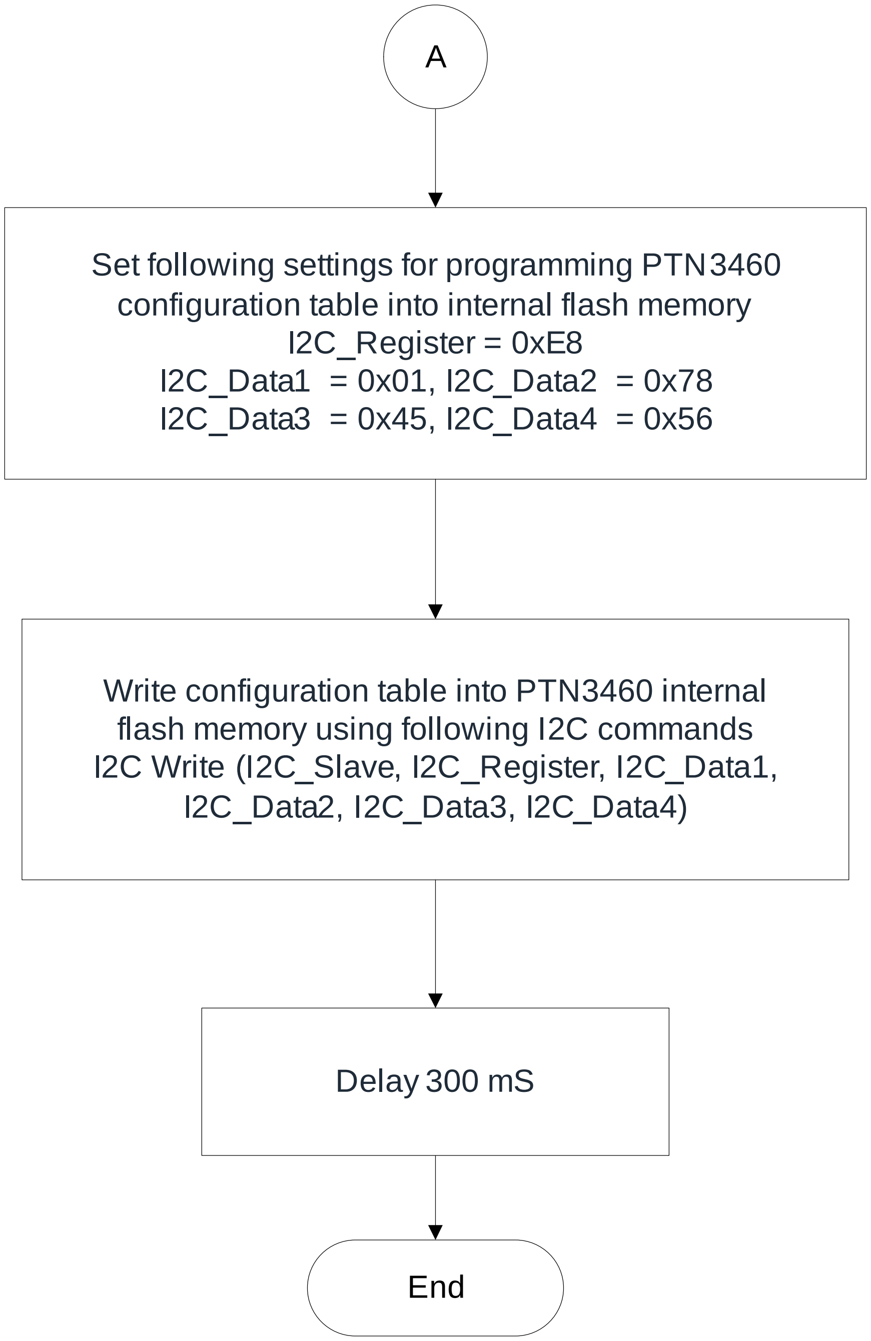

A
Set EDID Emulation mode ON?
No
Yes
Set EDID emulation off in configuration table as following settings
Set I2C_Register = 0x84
Set I2C_Data1 = 0x00
Set EDID emulation on and select EDID No 0 for emulation EDID in configuration table as following settings
Set I2C_Register = 0x84
Set I2C_Data1 = 0x01
Write EDID emulation on/off setting into PTN3460 configuration table using following I2C commands
I2C Write (I2C_Slave, I2C_Register, I2C_Data1)
B


B
Set following settings for programming PTN3460 configuration table into internal flash memory
I2C_Register = 0xE8
I2C_Data1 = 0x01, I2C_Data2 = 0x78
I2C_Data3 = 0x45, I2C_Data4 = 0x56
Write configuration table into PTN3460 internal flash memory using following I2C commands
I2C Write (I2C_Slave, I2C_Register, I2C_Data1, I2C_Data2, I2C_Data3, I2C_Data4)
Delay 300 mS
C

4.2. Program 1K configuration bytes into PTN3460 flow chart

PTN3460 M/S I2C bus supports I2C 8-bit addressing, so that it can only be written 256 bytes data into PTN3460 at one time ($2^8 = 256$). Following flow chart and operation steps describe how to write op_edid.bin into PTN3460.


Start
Write EDID0 (128 bytes) and Configuration register data (128 bytes) into PTN3460 Use I2C Write 256 byte ()
Is 7 EDIDs written into PTN3460?
Yes
A
No
Set I2C register 0x85 (EDID ROM access control) to 1-6 to select which EDID block (128 bytes) to be written into PTN3460 until 7 EDID writing is done.
Write EDID1-6 (128 bytes) into PTN3460 Use I2C Write 128 byte ()


A
Set following settings for programming PTN3460 configuration table into internal flash memory
I2C_Register = 0xE8
I2C_Data1 = 0x01, I2C_Data2 = 0x78
I2C_Data3 = 0x45, I2C_Data4 = 0x56
Write configuration table into PTN3460 internal flash memory using following I2C commands
I2C Write (I2C_Slave, I2C_Register, I2C_Data1, I2C_Data2, I2C_Data3, I2C_Data4)
Delay 300 mS
End

5. Configuration Registers

There are 128 bytes configuration registers in configuration table. Some of configuration registers have been defined and used in firmware, and some of them were reserved for further usage. Following are the detail explanations for the used configuration registers.

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0	EDID	128			EDID stored for emulation (7 EDID can be accessed from here, EDID selection is made through configuration register 0x85[2:0] - EDID ROM access control register.)
0x80	DisplayPort interface control	1	00	7:2	Reserved
				1	0 = AUX P/N swapping disabled. 1 = AUX P/N swapping enabled
				0	0 = Main link P/N swapping disabled. 1 = Main link P/N swapping enabled
0x81	LVDS interface control 1	1	0x0B	7:6	Reserved
				5:4	Color depth and data packing format 00 — VESA 24 bpp 01 — JEIDA 24 bpp 10 — VESA and JEIDA18 bpp 11 — reserved
				3	Dual LVDS mode 0 — Single LVDS bus mode 1 — Dual LVDS bus mode
				2	Data enable polarity 0 — active LOW 1 — active HIGH
				1:0	Clock output for dual LVDS mode. 00 — valid clock output on even bus only 01 — valid clock output on odd bus only 10 — reserved 11 — valid clock output on both buses

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0x82	LVDS interface control 2	1	0x03	7:6	Reserved
				5:3	LVDS clock frequency center spreading depth 000 — no spreading 001 — 0.5 % 010 — 1.0 % 011 — 1.5 % 100 — 2.0 % 101 — 2.5 % 110 to 111 — reserved
				2:0	LVDS differential output swing level 000 — 150 mV 001 — 200 mV 010 — 250 mV 011 — 300 mV 100 — 350 mV 101 — 400 mV 110 — 450 mV 111 — reserved
0x83	LVDS interface control 3	1	0x00	7:3	Reserved
				2	Differential pair (P/N) swapping 0 — normal (as indicated by pin name) 1 — swapped (P ↔ N)
				1	Channel differential pairs order swapping 0 — normal (as indicated by pin names) 1 — channel swapped (A ↔ D, B ↔ CLK, C ↔ C)
				0	LVDS bus swapping 0 — normal (as indicated by pin name) 1 — swapped (odd bus ↔ even bus)

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0x84	EDID ROM emulation control	1	0x09	7:4	Reserved
				3:1	Emulated EDID selection. Legal values are [0..6]
				0	Enable EDID emulation 0 — emulation OFF, EDID is read from DDC bus 1 — emulation ON, EDID is read from internal flash
0x85	EDID ROM access control	1	0x00	7:3	Reserved
				2:0	EDID to be accessed through offset 0..127 legal values are [0..6]
0x86	PWM minimum frequency	3	0x000001		24-bit PWM minimum allowed frequency (big endian)
0x89	PWM maximum frequency	3	0xFE0000		24-bit PWM maximum allowed frequency (big endian)
0x8C	Fast link training control	1	0x08	7:4	Reserved
				3:1	Default equalizer setting for fast link training (default 4) Legal values are 0 (strong EQ) to 7 (no EQ)
				0	Enable fast link training (default 0 = disabled) 0 - FALT is disabled 1 - FALT is enabled

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0x8D	Pin config control 1	1	0x00	7	Use CFG2 for Panel Self Test. 0 - Don't use CFG2 for Panel Self Test input pin. 1 - Use CFG2 for Panel Self Test input pin.
				6	Use CFG1 & CFG2 for brightness control (CFG1 = Br+, CFG2 = Br-) 0 - Don't use CFG1 & 2 for brightness control 1 - Use CFG1 & 2 for brightness control
				5	Use CFG3 & CFG4 for EDID selection (CFG3:CFG4 = EDID ROM Emulation control) 0 - Don't use CFG3 & 4 for EDID selection 1 - Use CFG3 & 4 for EDID selection
				4	Enable PWMI passtrough 0 - PWMI is disabled, PWM is internally generated (eDP) 1 - PWMI is enabled and used as PWM source
				3	PWMO disable (default 0 = enabled) 0 - PWMO output is enabled 1 - PWMO output is disabled (high impedance)
				2	Pin 33 behavior (default 0 = PVCCEN) 0 - PVCCEN 1 - GPIO
				1	Pin 31 & 32 behavior (default 0 = LVDSE) 0 - LVDSE_P and LVDSE_N pins 1 - BKLTEN and PWMO pins
				0	Pin 23 behavior (default 0 = CFG3) 0 - CFG3 configuration input 1 - BKLTEN (overrides pin 32 configuration)
0x8E	Pin config control 2	1	0x00	7:4	Reserved
				3	Override CFG4 pin 0 - CFG4 pin has priority on I2C/Flash configuration table 1 - I2C/Flash configuration table overrides CFG1 pin
				2	Override CFG3 pin 0 - CFG3 pin has priority on I2C/Flash configuration table 1 - I2C/Flash configuration table overrides CFG1 pin
				1	Override CFG2 pin 0 - CFG2 pin has priority on I2C/Flash configuration table 1 - I2C/Flash configuration table overrides CFG1 pin

				0	Override CFG1 pin 0 - CFG1 pin has priority on I2C/Flash configuration table 1 - I2C/Flash configuration table overrides CFG1 pin

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0x8F	PWM default bitcount	1	0x0C	7:0	Default bitcount for PWM control value in DPCD register 0x724
0x90-91	PWM value	2	0x07FF	7:0	Actual value of PWM in DPCD registers 0x722 & 0x723
0x92	PWM default freq	1	0x00	7:0	Default value for PWM frequency in DPCD register 0x728
0x93	Panel T3 timing	1	0x0A	7:0	Minimum T3 timing of panel power sequence to enforce (expressed in units of 50ms)
0x94	Panel T12 timing	1	0x14	7:0	Minimum T12 timing of panel power sequence to enforce (expressed in units of 50ms)
0x95	Backlight control	1	0x00	7:1	Reserved
				0	Backlight disable 0 - Backlight enabled 1 - Backlight disabled
0x96	Panel T2 delay	1	0x01	7:1	Reserved
				0	Enable T2 delay 0 - T2 is not delayed 1 - T2 is delayed by 16 ms
0x97	Panel T4 timing	1	0x02	7:0	Minimum T4 timing of panel power sequence to enforce (expressed in units of 50ms)
0x98	Panel T5 delay	1	0x01	7:1	Reserved
				0	Enable T5 delay 0 - T5 is not delayed 1 - T5 is delayed by 16 ms

Offset (HEX)	Name	Size (bytes)	Default Value	Bit field	Comment
0x99 - 0xD6	*Reserved*		*0x00*		
0xD7 - E7	FoA Mailbox	17	0x00		Byte area mapped to the FoA mailbox
0xE8	Flash command	1	0x00	7:0	Flash operation to be executed (default 0) 0 - Erase only 1 - Erase and flash
0xE9	Flash magic number	2	0x00	7:0	Must be equal to 0x7845 for the flash command to be actually processed (big endian) (default 0)
0xEB	Flash trigger	1	0x00	7:0	If 0x56 is written in this register and if Flash magic number is correct, then the flash command will be executed (default 0)
0xEC	Configuration magic number	4	0x12345678	7:0	Must be equal to 0x12345678 for PTN3460 to consider flashed configuration table (big endian) (default 0)

6. Programming Examples

6.1. Panel Brightness Control

// Register 0x90, 0x91 in configuration table are used for controlling PWM output for panel brightness control.

```
i2c_start();           // Generates I2C START condition
i2c_write(0x40);       // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x90);       // Select I2C Register 0x90
i2c_write(0x07);       // Register 0x90 0x91: PWM value registers
i2c_write(0xFF);       // Write PWM value 0x07FF to register 0x90, 0x91
i2c_stop();            //  Sends I2C stop-condition
```

6.2. Panel LVDS channel and data format setting

// Register 0x80 in configuration table is used for LVDS channel and data format setting

```
i2c_start();           // Generates I2C START condition
i2c_write(0x40);       // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x80);       // Register 0x80: LVDS interface control 1 register
i2c_write(0x0B);       // Set dual LVDS channel, VESA 24 bpp (bit per pixel)
i2c_stop();            //  Sends I2C stop-condition
```

6.3. LVDS clock frequency spreading depth setting

// Register 0x82 bit [5:3] in configuration table is used for LVDS clock frequency spreading depth setting

```
i2c_start();           // Generates I2C START condition
i2c_write(0x40);       // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x82);       // Select register 0x82: LVDS interface control 2
```

```
i2c_write(0x10);    // Set Register 0x82 bit [5:3] = 0b010: Set 1% spreading
i2c_stop();         //  Sends I2C stop-condition
```

6.4. LVDS differential output swing level

```
// Register 0x82 bit [2:0] in configuration table is used for LVDS differential output swing level
i2c_start();        // Generates I2C START condition
i2c_write(0x40);    // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x82);    // Select register 0x82: LVDS interface control 2
i2c_write(0x10);    // Set Register 0x82 bit [2:0] = 0b010: Set 250 mV swing level
i2c_stop();         //  Sends I2C stop-condition
```

6.5. Panel backlight control

```
// Register 0x95 bit [0] in configuration table is used for panel backlight control
i2c_start();        // Generates I2C START condition
i2c_write(0x40);    // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x95);    // Select register 0x95: Backlight control
i2c_write(0x00);    // Set Register 0x95 bit [0] = 0: Set panel backlight off
i2c_stop();         //  Sends I2C stop-condition
```

6.6. PWM output frequency setting

```
// Register 0x8F, 0x92 in configuration table are used for PWM output frequency setting
// PWM output frequency formula is Fpwm = 27000000 / (2^Pn * Fd)
// Pn is register 0x8F, Fd is register 0x92
i2c_start();        // Generates I2C START condition
```

```
i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x8F);     // Register 0x8F: PWM default bitcount

i2c_write(0x0C);     // Register 0x8F = 0x0C

i2c_stop();          //  Sends I2C stop-condition

i2c_start();         // Generates I2C START condition

i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x92);     // Register 0x92: PWM default freq

i2c_write(0x2e);     // Register 0x92 = 0x1e,

                     // Fpwm = 27000000 / (2^12 * 0x1e) = 27000000 / (4096 * 30) =~ 220Hz

i2c_stop();          //  Sends I2C stop-condition
```

6.7. Program CFG3, CFG4 pins as panel selection pins

```
// Register 0x8D bit 5 (Pin config control 1) in configuration table is used to set CFG3 and CFG4
// pins for EDID (panel) selection.

i2c_start();         // Generates I2C START condition

i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x8D);     // Register 0x8D: Pin config control 1 register

i2c_write(0x20);     // Register 0x8D = 0x20: set CFG3 and CFG4 pins for EDID selection

i2c_stop();          //  Sends I2C stop-condition

// Set CFG 3pin = 0, CFG4 pin = 0 to select emulation EDID no. 0 (Panel A)

// Set CFG 3pin = 0, CFG4 pin = 1 to select emulation EDID no. 1 (Panel B)

// Set CFG 3pin = 1, CFG4 pin = 0 to select emulation EDID no. 2 (Panel C)

// Set CFG 3pin = 1, CFG4 pin = 1 to select emulation EDID no. 3 (Panel D)
```

6.8. Write EDID No 0 – 6 to configuration table in internal SRAM

// Register 0x85 bit [2:0] (EDID ROM access control) is used to set where the EDID is going to be written to EDID 0 – 6 in 1k bytes configuration table in PTN3460 internal SRAM.

```
i2c_start();            // Generates I2C START condition
i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x85);        // Register 0x85: EDID ROM access control
i2c_write(0x02);        // Register 0x85 = 0x02, the EDID writes to I2C register 0x00 – 0x7F will be
                        // written to EDID 2 in configuration table in internal SRAM
i2c_stop();             //  Sends I2C stop-condition
i2c_start();            // Generates I2C START condition
i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0x00);        // Set EDID writing from I2C Register 0x00
i2c_write(128);       // Write 128 bytes EDID to EDID 2 into configuration table
i2c_stop();             //  Sends I2C stop-condition
```

6.9. Program configuration table into PTN3460 internal flash memory

// Register 0xE8 – 0xEB configuration table are used for programming configuration table into PTN3460 internal flash memory

```
i2c_start();            // Generates I2C START condition
i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)
i2c_write(0xE8);        // Register 0xE8: Flash operation register
i2c_write(0x01);        // Register 0xE8 = 0x01: Perform erase and then flash operation
i2c_write(0x78);        // Register 0xE9, EA = 0x7845: Flash magic number. Only flash magic
i2c_write(0x45);        // number is correct, then flash operation can be executed
i2c_write(0x56);        // Register 0xEB = 0x56: Flash trigger register. The 1K configuration table
                        // will be written into PTN3460 internal flash memory.
```

i2c_stop();   // Sends I2C stop-condition

6.10.Select emulation EDID No 0

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID
// selection

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);   // Register 0x84: EDID ROM emulation control register

i2c_write(0x01);   // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 0: Select EDID no 0
// for EDID emulation data.

i2c_stop();   // Sends I2C stop-condition

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 0
// from PTN3460.
// Emulation EDID no 0 in configuration table in PTN3460 firmware is **1024x768 @ 60Hz**
// and the EDID data is as following.

// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x3B, 0x10, 0x60, 0x34, 0x00, 0x00, 0x00, 0x00,

// 0x26, 0x15, 0x01, 0x03, 0x68, 0x1E, 0x16, 0x78, 0xEE, 0x37, 0x25, 0x9E, 0x58, 0x4A, 0x97, 0x26,

// 0x19, 0x50, 0x54, 0xAD, 0xEE, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x64, 0x19, 0x00, 0x40, 0x41, 0x00, 0x26, 0x30, 0x18, 0x88,

// 0x36, 0x00, 0x30, 0xE4, 0x10, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0xFD, 0x00, 0x32, 0x4C, 0x1E,

// 0x3F, 0x08, 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x41,

// 0x49, 0x4F, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFF,

// 0x00, 0x4E, 0x58, 0x50, 0x20, 0x50, 0x54, 0x4E, 0x33, 0x34, 0x36, 0x30, 0x20, 0x20, 0x00, 0xAF,

6.11.Select emulation EDID No 1

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID
selection

```
i2c_start();          // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);     // Register 0x84: EDID ROM emulation control register

i2c_write(0x03);      // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 1: Select EDID no 1
// for EDID emulation data.

i2c_stop();           //  Sends I2C stop-condition

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 1
// from PTN3460.
// Emulation EDID no 1 in configuration table in PTN3460 firmware is 1920 x 1080 @ 60Hz
// and the EDID data is as following.

// x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x3b, 0x10, 0x36, 0x92, 0x89, 0x25, 0x71, 0x02,

// 0x20, 0x13, 0x01, 0x04, 0xa5, 0x34, 0x20, 0x78, 0x26, 0x6e, 0xa1, 0xa7, 0x55, 0x4c, 0x9d, 0x25,

// 0x0e, 0x50, 0x54, 0x00, 0x00, 0x00, 0xd1, 0xc0, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x02, 0x3a, 0x80, 0x18, 0x71, 0x38, 0x2d, 0x40, 0x58, 0x2c,

// 0x45, 0x00, 0xe0, 0x0e, 0x11, 0x00, 0x00, 0x1a, 0x00, 0x00, 0x00, 0xff, 0x00, 0x32, 0x41, 0x39,

// 0x33, 0x32, 0x30, 0x45, 0x41, 0x30, 0x4b, 0x30, 0x0a, 0x20, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x50,

// 0x54, 0x4e, 0x33, 0x34, 0x36, 0x30, 0x20, 0x20, 0x20, 0x0a, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00,

// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xCD,
```

6.12.Select emulation EDID No 2

```
// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID
// selection

i2c_start();          // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);     // Register 0x84: EDID ROM emulation control register

i2c_write(0x05);      // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 2: Select EDID no 2
for EDID emulation data.

i2c_stop();           //  Sends I2C stop-condition
```

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 2
// from PTN3460.
// Emulation EDID no 2 in configuration table in PTN3460 firmware is **1920 x 1080 @ 60Hz**
// and the EDID data is as following.

// 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x3b, 0x10, 0x01, 0x10, 0x01, 0x00, 0x00, 0x00,

// 0x27, 0x14, 0x01, 0x03, 0x80, 0x33, 0x1d, 0x78, 0xee, 0x0c, 0xb0, 0xa2, 0x57, 0x51, 0x9f, 0x27,

// 0x0a, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9d, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1f, 0x40, 0x1e, 0x14,

// 0x35, 0x00, 0xfd, 0x1e, 0x11, 0x00, 0x00, 0x1e, 0x9d, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1f, 0x40,

// 0x28, 0x14, 0xc7, 0x00, 0xfd, 0x1e, 0x11, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x41,

// 0x49, 0x4f, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfd,

// 0x00, 0x32, 0x4b, 0x37, 0x53, 0x08, 0x00, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x0c,

6.13.Select emulation EDID No 3

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID selection

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);        // Register 0x84: EDID ROM emulation control register

i2c_write(0x07);        // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 3: Select EDID no 3
//for EDID emulation data.

i2c_stop();             //  Sends I2C stop-condition
```

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 3 from // PTN3460.

// Emulation EDID no 3 in configuration table in PTN3460 firmware is **1600 x 900 @ 60Hz**
// and the EDID data is as following.

// 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x3B, 0x10, 0x4a, 0x31, 0x00, 0x00, 0x00, 0x00,

// 0x00, 0x14, 0x01, 0x01, 0x80, 0x33, 0x1d, 0x78, 0x0a, 0x87, 0xf5, 0x94, 0x57, 0x4f, 0x8c, 0x27,

// 0x27, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x3c, 0x2e, 0x40, 0x00, 0x62, 0x84, 0x22, 0x30, 0x58, 0xa8,

// 0x35, 0x00, 0xbb, 0xf9, 0x10, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x00,

// 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x50, 0x0c, 0x13, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x41,

// 0x49, 0x4f, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xfe,

// 0x00, 0x31, 0x38, 0x34, 0x48, 0x54, 0x30, 0x33, 0x2d, 0x30, 0x30, 0x31, 0x0a, 0x20, 0x00, 0x30,

6.14.Select emulation EDID No 4

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID selection

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);        // Register 0x84: EDID ROM emulation control register

i2c_write(0x09);        // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 4: Select EDID no 4
for EDID emulation data.

i2c_stop();             //  Sends I2C stop-condition
```

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 4
// from PTN3460.
// The emulation EDID no 4 in configuration table in PTN3460 firmware is **1920 x 1080 @ 60Hz**
// and the data is as following.

// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x3B, 0x10, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,

// 0x2B, 0x15, 0x01, 0x03, 0x80, 0x35, 0x1F, 0x78, 0xEE, 0x9A, 0x80, 0xA3, 0x58, 0x51, 0x9E, 0x25,

// 0x0E, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x40, 0x38, 0x80, 0x00, 0x71, 0x38, 0x14, 0x40, 0x58, 0x2C,

// 0x45, 0x00, 0xFE, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x4F, 0x70, 0x74,

// 0x69, 0x50, 0x6C, 0x65, 0x78, 0x20, 0x39, 0x30, 0x31, 0x30, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,

// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,

// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x1D,

6.15.Select emulation EDID No 5

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID selection

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);   // Register 0x84: EDID ROM emulation control register

i2c_write(0x0B);   // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 5: Select EDID no 5 for EDID emulation data.

i2c_stop();   // Sends I2C stop-condition

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 5
// from PTN3460.
// Emulation EDID no 5 in configuration table in PTN3460 firmware is 1366 x 768 @ 60Hz
// and the EDID data is as following.

// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x3B, 0x10, 0x60, 0x34, 0x84, 0x0D, 0x00, 0x00,

// 0x26, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0x0A, 0x87, 0xF5, 0x94, 0x57, 0x4F, 0x8C, 0x27,

// 0x27, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x3E, 0x1C, 0x56, 0xA0, 0x50, 0x00, 0x16, 0x30, 0x30, 0x20,

// 0x25, 0x00, 0x99, 0xE6, 0x10, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x00,

// 0x00, 0x00, 0x00, 0x00, 0x00, 0x1B, 0xE8, 0x04, 0x5A, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x00, 0x4E,

// 0x58, 0x50, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFE,

// 0x00, 0x41, 0x49, 0x4F, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0xC6,

6.16.Select emulation EDID No 6

// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID selection

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);        // Register 0x84: EDID ROM emulation control register

i2c_write(0x0D);        // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 6: Select EDID no 6
for EDID emulation data.

i2c_stop();             //  Sends I2C stop-condition

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 6
// from PTN3460.
// Emulation EDID no 6 in configuration table in PTN3460 firmware is 1600 x 900 @ 60Hz
// and the data is as following.

// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x3B, 0x10, 0xBF, 0x01, 0x01, 0x00, 0x50, 0x63,

// 0x0C, 0x14, 0x01, 0x04, 0xA5, 0x30, 0x1B, 0x78, 0x22, 0x35, 0x85, 0xA6, 0x56, 0x48, 0x9A, 0x24,

// 0x12, 0x50, 0x54, 0x21, 0x08, 0x00, 0x81, 0xC0, 0x95, 0xC0, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x2F, 0x26, 0x40, 0xA0, 0x60, 0x84, 0x1A, 0x30, 0x30, 0x20,

// 0x35, 0x00, 0xDD, 0x0C, 0x11, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0xFD, 0x00, 0x38, 0x4B, 0x1E,

// 0x53, 0x11, 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x41,

// 0x49, 0x4F, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFF,

// 0x00, 0x4C, 0x35, 0x36, 0x30, 0x35, 0x31, 0x37, 0x39, 0x34, 0x33, 0x30, 0x32, 0x0A, 0x01, 0xAD
```

6.17.Select emulation EDID No7

```
// Register 0x84 in configuration table is used for EDID ROM emulation control and 7 EDID
selection

i2c_start();            // Generates I2C START condition

i2c_write(0x40);        // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x84);        // Register 0x84: EDID ROM emulation control register

i2c_write(0x0D);        // 0x84 bit [0] = 1: EDID emulation is on. 0x84 bit [3:1] = 6: Select EDID no 6
for EDID emulation data.

i2c_stop();             //  Sends I2C stop-condition
```

// When GPU uses I2C-Over-Aux to read EDID from PTN3460, it will get emulation EDID no 6
// from PTN3460.
// Emulation EDID no 6 in configuration table in PTN3460 firmware is **1600 x 900 @ 60Hz**
// and the data is as following.

// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x3B, 0x10, 0xBF, 0x01, 0x01, 0x00, 0x50, 0x63,

// 0x0C, 0x14, 0x01, 0x04, 0xA5, 0x30, 0x1B, 0x78, 0x22, 0x35, 0x85, 0xA6, 0x56, 0x48, 0x9A, 0x24,

// 0x12, 0x50, 0x54, 0x21, 0x08, 0x00, 0x81, 0xC0, 0x95, 0xC0, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,

// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x2F, 0x26, 0x40, 0xA0, 0x60, 0x84, 0x1A, 0x30, 0x30, 0x20,

// 0x35, 0x00, 0xDD, 0x0C, 0x11, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0xFD, 0x00, 0x38, 0x4B, 0x1E,

// 0x53, 0x11, 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x41,

// 0x49, 0x4F, 0x20, 0x50, 0x43, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFF,

// 0x00, 0x4C, 0x35, 0x36, 0x30, 0x35, 0x31, 0x37, 0x39, 0x34, 0x33, 0x30, 0x32, 0x0A, 0x01, 0xAD

6.18. Program 1K configuration bytes into PTN3460 steps

Step1: Write EDID No0 (128 bytes) and configuration register data (128 bytes) into PTN3460

i2c_start(); // Generates I2C START condition

i2c_write(0x40); // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00); // Set PTN3460 I2C register address = 0x00

i2c_write_256bytes(); // Write EDID No0 and configuration data (shown in below) into PTN3460

// EDID No0 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x40, 0xA7, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x50, 0x45, 0x47,
// 0x31, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x45,
// Configuration register data (128 bytes as below)
// 0x00, 0x0B, 0x2B, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0xFE, 0x00, 0x00, 0x08, 0x20, 0x00, 0x0C,
// 0x07, 0xFF, 0x00, 0x0A, 0x24, 0x00, 0x01, 0x02, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```
// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x34, 0x56, 0x78,
// 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF

i2c_stop();           //  Sends I2C stop-condition
```

<u>Step2: Write EDID No1 (128 bytes) into PTN3460</u>

```
i2c_start();          // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x85);      // Register 0x85: EDID ROM access control register

i2c_write(0x01);      // Register 0x85= 0x01: Set EDID No1 to be access through I2C register
0x00 – 0x7f

i2c_stop();           //  Sends I2C stop-condition

i2c_start();          // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);      // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();     // Write EDID No1 (shown in below) into PTN3460

// EDID No1 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x10, 0xAC, 0x48, 0x05, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0x9A, 0x80, 0xA3, 0x58, 0x51, 0x9E, 0x25,
// 0x0E, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x40, 0x38, 0x80, 0x00, 0x71, 0x38, 0x14, 0x40, 0x58, 0x2C,
// 0x45, 0x00, 0xFE, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x49, 0x4E, 0x53,
// 0x50, 0x49, 0x52, 0x4F, 0x4E, 0x20, 0x4F, 0x4E, 0x45, 0x0A, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x33,

i2c_stop();           //  Sends I2C stop-condition
```

Step3: Write EDID No2 (128 bytes) into PTN3460

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x85);     // Register 0x85: EDID ROM access control register

i2c_write(0x02);      // Register 0x85= 0x02: Set EDID No2 to be access through I2C register
0x00 – 0x7f

i2c_stop();           //  Sends I2C stop-condition

i2c_start();            // Generates I2C START condition

i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);     // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();      // Write EDID No2 (shown in below) into PTN3460

// EDID No2 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x10, 0xAC, 0x48, 0x05, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x49, 0x4E, 0x53,
// 0x50, 0x49, 0x52, 0x4F, 0x4E, 0x20, 0x4F, 0x4E, 0x45, 0x0A, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0xDE,

i2c_stop();           //  Sends I2C stop-condition
```

Step4: Write EDID No3 (128 bytes) into PTN3460

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x85);     // Register 0x85: EDID ROM access control register

i2c_write(0x03);      // Register 0x85= 0x03: Set EDID No3 to be access through I2C register
0x00 – 0x7f

i2c_stop();           //  Sends I2C stop-condition
```

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);     // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();      // Write EDID No3 (shown in below) into PTN3460

// EDID No3 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x40, 0xA7, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x50, 0x45, 0x47,
// 0x34, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x42,

i2c_stop();            //  Sends I2C stop-condition
```

Step5: Write EDID No4 (128 bytes) into PTN3460

```
i2c_start();            // Generates I2C START condition

i2c_write(0x40);      // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x85);     // Register 0x85: EDID ROM access control register

i2c_write(0x04);      // Register 0x85= 0x04: Set EDID No4 to be access through I2C register
0x00 – 0x7f

i2c_stop();            //  Sends I2C stop-condition

i2c_start();            // Generates I2C START condition

i2c_write(0x40);     // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);     // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();      // Write EDID No4 (shown in below) into PTN3460

// EDID No4 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x40, 0xA7, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
```

// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x50, 0x45, 0x47,
// 0x35, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x41,,

i2c_stop();   // Sends I2C stop-condition

Step6: Write EDID No5 (128 bytes) into PTN3460

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x85);   // Register 0x85: EDID ROM access control register

i2c_write(0x05);   // Register 0x85= 0x05: Set EDID No5 to be access through I2C register 0x00 – 0x7f

i2c_stop();   // Sends I2C stop-condition

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);   // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();   // Write EDID No5 (shown in below) into PTN3460

// EDID No5 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x40, 0xA7, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x50, 0x45, 0x47,
// 0x36, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x40,

i2c_stop();   // Sends I2C stop-condition

Step7: Write EDID No6 (128 bytes) into PTN3460

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low) i2c_write(0x85);   // Register 0x85: EDID ROM access control register

i2c_write(0x05);   // Register 0x85= 0x06: Set EDID No6 to be access through I2C register 0x00 – 0x7f

i2c_stop();   // Sends I2C stop-condition

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0x00);   // Set PTN3460 I2C register address = 0x00

i2c_write_128bytes();   // Write EDID No6 (shown in below) into PTN3460

// EDID No6 (128 bytes as below)
// 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x40, 0xA7, 0x11, 0x11, 0x01, 0x00, 0x00, 0x00,
// 0x1E, 0x15, 0x01, 0x03, 0x80, 0x33, 0x1D, 0x78, 0xEE, 0xEE, 0x20, 0xA3, 0x56, 0x51, 0x9E, 0x27,
// 0x0B, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x9D, 0x34, 0x80, 0x64, 0x70, 0x38, 0x1F, 0x40, 0x1E, 0x14,
// 0x35, 0x00, 0xFD, 0x1E, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x50, 0x45, 0x47,
// 0x37, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x0A,
// 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0xFC,
// 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x3F,

i2c_stop();   // Sends I2C stop-condition

Step8: Write 7 EDID and configuration register data into PTN3460 internal flash memory (1K bytes)

// Register 0xE8 – 0xEB configuration table are used for programming configuration table into PTN3460 internal flash memory

i2c_start();   // Generates I2C START condition

i2c_write(0x40);   // Select PTN3460 slave address (DEV_CFG pin = low)

i2c_write(0xE8);   // Register 0xE8: Flash operation register

i2c_write(0x01);   // Register 0xE8 = 0x01: Perform erase and then flash operation

i2c_write(0x78);   // Register 0xE9, EA = 0x7845: Flash magic number. Only flash magic

i2c_write(0x45);   // number is correct, then flash operation can be executed

i2c_write(0x56);   // Register 0xEB = 0x56: Flash trigger register. The 1K configuration table

```
                        // will be written into PTN3460 internal flash memory.
i2c_stop();             //  Sends I2C stop-condition
 wait (300ms);          // Wait 300mS to have PTN3460 internal flash memory programmed properly.
```

6.19.Read 1K configuration bytes from PTN3460

When reading out EDIDs and configuration register data from PTN3460 for verification, it should do same procedure like writing procedure just change write data command to read data command.