



## **XPOS XP-36xx USER MANUAL**

## Autorské právo

Tato publikace obsahuje informace chráněné autorským právem. Žádná část této publikace nesmí být kopírována ani reprodukována v jakékoliv formě nebo jakýmkoliv způsobem bez svolení vlastníků.

Dovozce si vyhrazuje právo na změny v této publikaci, změny specifikace a dále změny na výrobku bez předchozího upozornění. Dovozce nenese zodpovědnost za technické a tiskové chyby uvedené v této publikaci, ani za škody vzniklé v souvislosti s nesprávným použitím této publikace.

Názvy výrobků a výrobní značky mohou být obchodní značkou a/nebo registrovanou obchodní značkou jejich vlastníků. Ostatní chráněné značky jsou vlastnictvím příslušných majitelů. Informace a specifikace v této příručce jsou předmětem průběžných aktualizací. Dovozce proto neručí za chyby

a nesrovnalosti, které mohu v této publikaci vzniknout.

© 2020 Všechna práva vyhrazena

# TABLE OF CONTENTS

### **1. INTRODUCTION**

XP-3685 Basic Introduction	005 - 009
Dimensions	010 - 010
Specification	011 - 011
Packing List	012 - 012
2. SOFTWARE INSTALLATION AND SETUP	
Installation Order	012 - 012

	012 - 012
Chipset	013 - 014
Audio	014 - 014
Graphics	015 - 015
Intel ME	016 - 016
Serial IO	017 - 017
USB LAN	018 - 018
Windows Driver	020 - 020
Set COM	021 – 021
Sensor	020 - 020
Intel® Rapid Storage Technology	022 - 023
ADB Android Debug Bridge	023 - 023

## 3. Hardware

Access Storage Device Adding or Removing M.2 PCBA from M.2 Tray	023 - 024 025 - 025
Access the Motherboard	026 - 027
Install or Remove Memory	027 - 028

## 4. IO Board SDK Instruction

Overview	028 - 028
System Requirements & Installation	029 - 030
DLL Interface for FEC Io Board SDK	030 - 032
Example of DLL Interface	033 - 033
EXE Interface for FEC Cash Drawer SDK	033 - 034

### 5. Software

Developers Guide for Serial Communications 034 -	041
Ambient Light and Proximity Sensor 041 -	043
Control the Sensor Timing 044 -	044
Cash Drawer Command 045 -	045

# **XP-3685 Basic Introduction**

XPOS XP-3685 is a fifteen inch all in one fanless point of sales system powered by Intel® 7<sup>th</sup> generation Celeron, i3 and i5 processors. By utilizing slim key components and enclosing them in an aluminum die-casting chassis, XP-3685 is able to achieve both a slim and strong design. The IO interface is connected with an USB cable and installed inside the stand base.

The dual hinge stand provides the user the ability to adjust the stand and display angles making it optimal for all environments.

On the bottom of the display is a standard 2in1 sensor. The proximity sensor can detect a presence infront of the sensor and wake the system up from an S1 and S3 sleep state allowing users to save on power consumption when the store traffic is low.

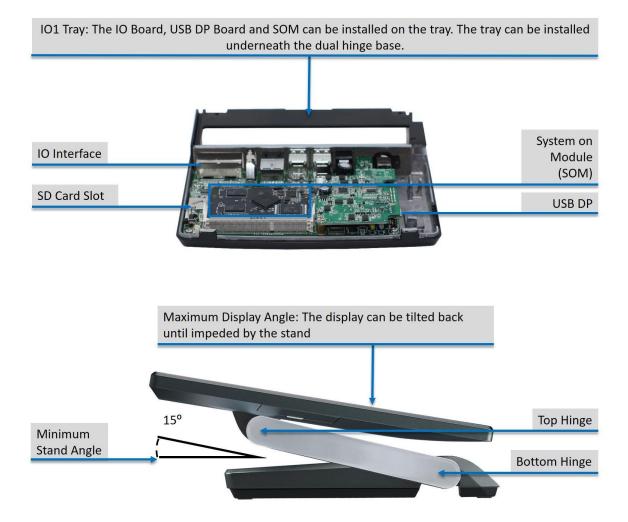
- Slim Panel PC 2.4cm
- Fanless Design
- 2in1 Sensor: Ambient Light Sensor & Proximity Sensor
- Dual Hinge Stand: High Profile and Low Profile
- Integrated and Extended 2<sup>nd</sup> Display Options





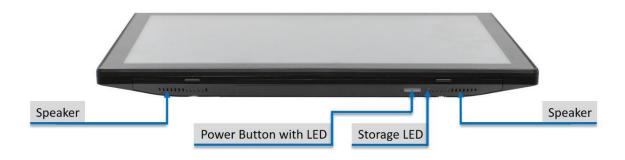


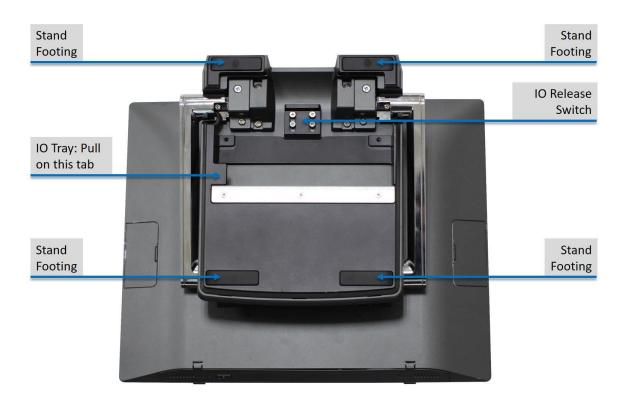




Power Button with LED:

- Red = Power adaptor connected but system is off Blue = System is on
- Storage LED
   Orange = Flashing during storage activity





## Galaxy Gray with Black Silver with Black



## **Dimensions**



## **Display Dimension**

		XP-3685	XP-3685W
Α	Display Height	273.3mm	235.7mm
В	Display Length	342.5mm	368.2mm



## **Side Profile Dimension**

		XP-3685	XP-3685W
A	Panel PC Thickness	24mm	24mm
В	System	332mm	293mm
С	Base Depth	231.75mm	231.75mm

# **SPECIFICATION**

Panel PC			
Description	XP-3685		
Processor	Intel® Celeron® Processor 3965U, 2M Cache, 2.20 GHz Intel® Core™ i3-7100U Processor, 3M Cache, 2.40 GHz Intel® Core™ i5-7300U Processor, 3M Cache, up to 3.50 GHz		
System Memory	4GB Standard, Maximum 320	GB (2 x 260-pin DDR4)	
Storage Device		Celeron: 1 x M.2 (B+M Key) SATA III and PCIE, 1 x M.2 (BM Key) SATA III i3 & i5: 2 x M.2 (B+M Key) SATA III and PCIE	
Speaker	2 x 2W Internal Speaker		
Construction	Aluminum Die-casting + Plastic + Glass		
Housing Color	Galaxy Gray + Black Silver + Black		
Touch LCD Display			
Size / Resolution	15″ TFT-LCD / 1024 x 768	15.6" TFT-LCD / 1920 x 1080	
Brightness / Backlight	400nits (LED) PCAP	400nits (LED) PCAP	
Panel PC IO			
USB Port	1 x USB 1.0 Type A, 1 x USB 2.0 Type A 1 x USB DP (Reserved for 2nd Display)		
FEC DP Port	1 x FEC DP Port (Customer Display or Integrated 2 <sup>nd</sup> Display)		

## **IO Options**



IO1	
System on Module (SOM)	ARM Cortex A7 Quad Core
Optional	1G DDR3 On Board
	8GB eMMC 5.0 On Board
SD Card Slot	1 x SD Card Slot (Must Have SOM)
Video Port	1 x Video Port (Must Have SOM)

USB DP	2 x USB DP Port (1Reserved to Connect to Panel PC)	
USB Port	4 x USB 2.0 Type A	
LAN Port	1 x LAN (Green Light Mega LAN, Orange Light Giga LAN)	
Serial Port	2 x RJ45 (RS232)	
DC-in	1 x DC-in for 90W (20V/4.5A) Adaptor	

# **Packing List**

Standard	Optional
XP-3685 x 1	WiFi Module x 1
M.2 Tray x 1	M.2 Tray x 1
COM RJ45 to DB9 Cable x 2	Add-on Device x1
90W Adaptor x 1	Customer Display x 1
Power Cord x 1	2 <sup>nd</sup> Display x 1
Driver and Manual CD x 1	

# Software Installation and Setup: Motherboard

Follow the below order if installation for the Motherboard:

O1 Chipset O2 Audio O3 Graphics O4 Intel® ME

O5 Setup Serial IO O6 USB LAN O7 Windows Drivers O8 Set COM

O9 Intel<sup>®</sup> Rapid Storage Technology (RST) (Applicable to i3 & i5) O10 ADB Driver (Applicable to SOM)

← → ∽ ↑ <mark> </mark> → T	his PC → Local	Disk (C:) > New folder > 主機 > WIN10	✓ Ö Sear	h WIN10	Q
		Name	Date modified	Туре	Size
Quick access     Desktop     Doumloads     Documents     Pictures     This PC     Desktop     Documents     Documents     Documents     Music     Pictures     Videos     Local Disk (C:)     Network	**	<ul> <li>Audio_8425</li> <li>Chipset_10.1.1.45</li> <li>CP210x_Windows_Drivers.rar</li> <li>ME_11.8.50.3425_SOL</li> <li>RST_16.50.1021</li> <li>SerialIO_30.100.1746.4</li> <li>USB_LAN_Win10_10023_03012018</li> <li>VGA_4973</li> </ul>	4/25/2018 4:02 AM 4/25/2018 3:59 AM 10/30/2018 8:45 PM 5/29/2018 8:51 PM 5/27/2018 8:49 PM 4/25/2018 4:02 AM 4/25/2018 4:02 AM 4/25/2018 4:04 AM	Compressed (zipp Compressed (zipp RAR File Compressed (zipp Compressed (zipp Compressed (zipp Compressed (zipp	295,( 3, 6,1 104,( 17,( 2,,2 9,4 338,(
3 items	<				

#### □ Un-compress files

### 1. Chipset

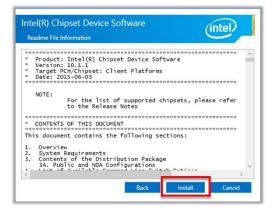
#### Locate chipset folder and double click on [ SetupChipset ]



#### □ Click [ Next ]



#### □ Click [ Install ]



#### □ Click [ Accept ]



#### Click [ Yes ]





#### Audio

## Double click [ Setup ]

🖓 📒 🖛		Compressed Folder Tools	Audio_8425		- 0	×
File Home Share	View	Extract				~ 0
← → ~ 个 🦲 « 主機 )	WIN10	> Audio_8425 > Audio_	8425 >	v Ö	Search Audio_8425	ø
		Name ^		Туре	Compressed size	Pa *
🖈 Quick access		(a) 0x0412		Configuration setti		Nc
Desktop	1	(i) 0x0413		Configuration setti	ngs 6 KB	Nc
Downloads		DxD414		Configuration setti	ings 5 KB	Nc
Cocuments	+	@ 0x0415		Configuration setti	ngs 6 KB	No
Pictures		0x0416		Configuration setti		Nc
New folder	~	6 0x0419		Configuration setti	ngs 6 KB	Nc
New folder		(a) 0x0421		Configuration setti	ngs 6 KB	Ne
ConeDrive		@ 0x0424		Configuration setti	ngs 6 KB	Nc
		6 0x0604		Configuration setti	ngs 5 KB	Ne
This PC		6 0x0616		Configuration setti	ngs 6 KB	Nc
Desktop		ChCfg		Application	35 KB	Ne
B Documents		🗮 data1		Cabinet File	5,989 KB	Ne
Downloads		data1.hdr		HDR File	9 K B	No
b Music		🛗 data2		Cabinet File	1 KB	Nc
Pictures		Setup.dll		Application extensi	ion 701 KB	Ne
Videos		] layout.bin		BIN File	1 KB	Ne
	_	RtlExUpd.dll		Application extensi		Nc
Local Disk (C:)		Setup		Application	512 KB	No
Network		(a) setup		Configuration setti	ngs 2 kB	No
		setup.inx		INX File	361 KB	Ne
		setup.isn		ISN File	59 KB	Ne
		setup.iss		ISS File	1 KB	No
		USetup.iss		ISS File	1 KB	Ne .
	<					>
16 items						11 1

### Click [ Next ]



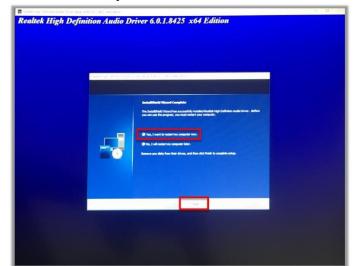
#### □ Click [ Restart Now ]



#### Click [ Yes ]

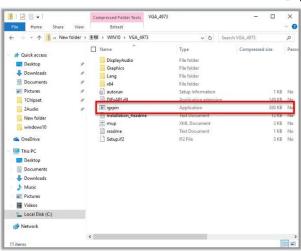


# Click [ Yes, I want to restart my computer now ], [ Finish ]



#### Graphics

#### Double Click [ igxpin ]



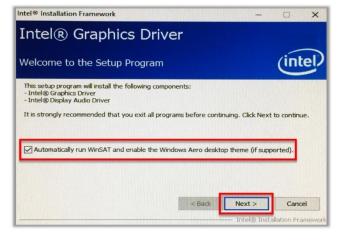
٠

#### ] 🛛 🛛 Click **[ Yes ]**

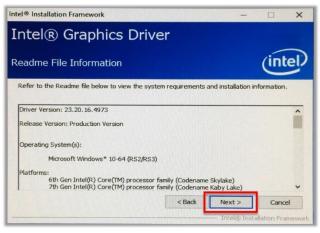


Click [ Next ]

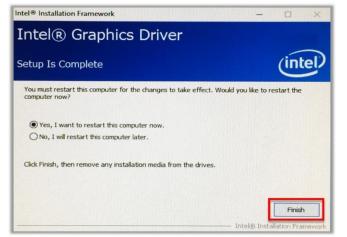
٠



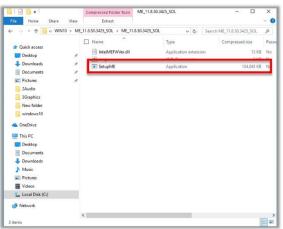
#### Click [ Next ]







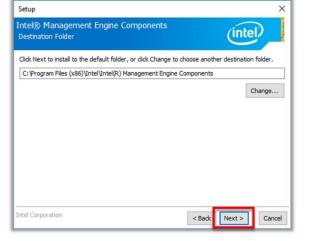
#### • Double Click [ SetupME ] Click [ Next ]



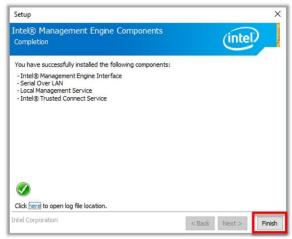


• Accept Terms then click [ Next ] 
Click [ Next ]





Click [ Finish ]



Click [ Yes ]

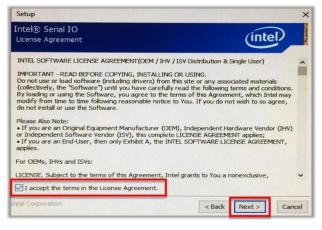


#### Serial IO

Double Click [ SetupSerialIO ]



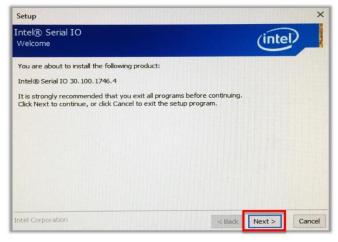
□ Accept terms then click [ Next ]



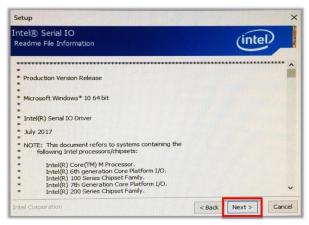
Click [ Next ]

Setup			×
Intel® Serial IO Confirmation		(inte	D
You are about to install the following components:			
- Intel® Serial IO GPIO Driver - Intel® Serial IO 12C Driver			
Intel Corporation	< Back	Next >	Cancel

#### □ Click [ Next ]



Click [ Next ]



Select Yes, I want to restart this computer now the click [ Finish ]

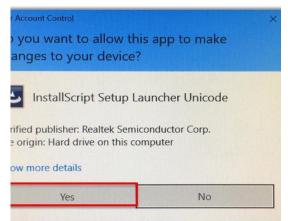


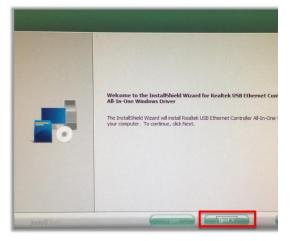
## 6. USB LAN

#### Double Click [ Setup]

File Home Share	View	Application Tools New folder Manage		- 0	×
← → × ↑ 📑 > TI	his PC → Loc	al Disk (C:) > New folder > New folde	r ~ 0	Search New folder	p
📌 Quick access		□ Name ^ (1) 0x0414	Date modified 11/6/2017 4:59	Type PM Configuration sett	Size
Desktop	*	6 0x0415	11/6/2017 4:59	PM Configuration sett	
Downloads	*	🗟 0x0416	11/6/2017 4:59	PM Configuration sett	
Documents	*	6 0x0418	11/6/2017 4:59	PM Configuration sett	
E Pictures	*	6 0x0419	11/6/2017 4:59	PM Configuration sett	
ZAudio		6 0x0421	11/6/2017 4:59	PM Configuration sett	
		6 0x0424	11/6/2017 4:59		
3Graphics		6 0x0804	11/6/2017 4:59	PM Configuration sett	
4ME		6 0x0816	11/6/2017 4:59		
New folder		🔡 data1	11/6/2017 4:59		6
G OneDrive		data1.hdr	11/6/2017 4:59		
Chebrive		data2	11/6/2017 4:59		
This PC		EngLangID	11/6/2017 4:59		
Desktop		InstCtrl	11/6/2017 4:59		
Documents		InstOpt	11/6/2017 4:59	The second second second	
Downloads		ISSetup.dll	11/6/2017 4:59	PM Application extens	
h Music		(j) to a state of the state of	14/6/0017-160	01.1 OH10%-	
E Pictures		🗹 💽 setup	11/6/2017 4:59		- 1
		gj setup	11/6/2017 4:59		
Videos	_	setup.inx	11/6/2017 4:59		
Local Disk (C:)		setup.isn	11/6/2017 4:59		
A Network		setup.iss	11/6/2017 4:59 11/6/2017 4:59		
and the second		setupctrl	11/6/2017 4:59	PM Text Document	
58 items 1 item selected		5			

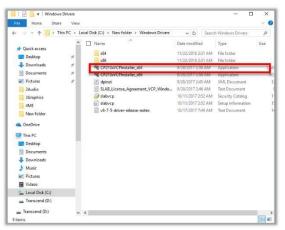
Click [ Yes ]



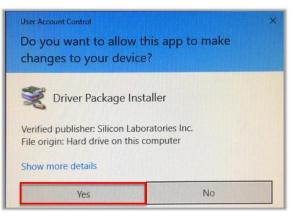


## 7. Windows Driver

#### Double Click [ CP210XVCPInstaller\_x64 ]



#### Click [ Yes ]



#### 8. Set COM and COM Address

#### 1. Double click [ SetXPOSCOM ]

□ You will see the COM is set in Device Manager

🔄 📑 =		Application Tools	20180910_SetXPOSCOM v1.0.1			- 0	
File Home	Share	View Manage					~
> • 🛧 📙	> This	s PC > Desktop > 20180910_	SetXPOSCOM v1.0.1	ō	Search 201809	10_SetXPOSCO	1
		Name ^	Date modified		Туре	Size	
Quick access		🔳 devcon_64	12/18/2015 1:44 A	M	Application	80 KB	
Desktop	Ħ	SetXPOSCOM	9/11/2018 6:21 AM	Л	Application	15 KB	I
👆 Downloads	*	SetXPOSCOM.exe.cor	nfig 8/8/2018 4:38 PM		CONFIG File	1 KB	
🔮 Documents	*	SetXPOSCOM.pdb	9/11/2018 6:21 AM	Л	PDB File	20 KB	
Pictures	*	SetXPOSCOM.vshost	9/12/2018 1:17 AM	Λ	Application	23 KB	í.
sleeper		SetXPOSCOM.vshost.	exe.config 8/8/2018 4:38 PM		CONFIG File	1 KB	r.
🛆 OneDrive		SetXPOSCOM.vshost.	exe.manifest 8/10/2018 4:29 AN	1	MANIFEST File	4 KB	

### **COM Address**

You can check Ports (COM & LPT) in the Device Manager

#### Y 🙀 Ports (COM & LPT)

- Communications Port (COM1)
- Communications Port (COM2)
- Silicon Labs Quad CP2108 USB to UART Bridge: Interface 0 (COM98)
- Silicon Labs Quad CP2108 USB to UART Bridge: Interface 1 (COM4)
- Silicon Labs Quad CP2108 USB to UART Bridge: Interface 2 (COM97)
- Silicon Labs Quad CP2108 USB to UART Bridge: Interface 3 (COM3)

#### After the SET COM you should see the below items in the Device Manager:

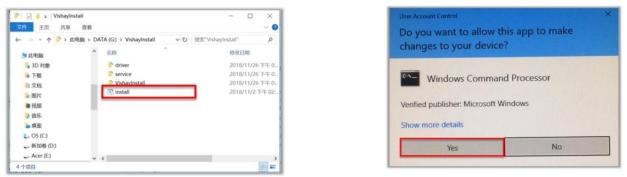
- □ Communications Port (COM1): FEC DP interface reserved for customer display
- Communications Port (COM2): Reserved on motherboard

- □ Silicon Labs Quad CP2108 USB to UARD Bridge: Interface 0 (COM98): GPIO Control for IO1
- □ Silicon Labs Quad CP2108 USB to UARD Bridge: Interface 1 (COM3): RJ45 Interface on IO for Devices
- □ Silicon Labs Quad CP2108 USB to UARD Bridge: Interface 1 (COM4): RJ45 Interface on IO for Devices
- □ Silicon Labs Quad CP2108 USB to UARD Bridge: Interface 1 (COM3): RJ45 Interface on IO for Devices
- □ Intel iAMT: Available on i3 and i5 models

#### 9. Sensor

Double Click [ install ]

### Click [ Yes ]



Press any key to continue ...

C\Windows\System32\cmd.exe		-	
logfile = C:\Windows\Hi	crosoft.NET\Framework64\v4.0.30319\VishaySensorService.InstallLog		
The Commit phase completed			
The transacted install has [SC] ChangeServiceConfig S			
SERVICE_NAME: SensrSvc TYPE STATE WIN32_EXIT_CODE SERVICE_EKIT_CODE CHECKPDINT WAIT_HINT PID FLAGS [SC] ChangeServiceConfig S	: 0 (0x0) : 0x0/00 : 0x7/00 : 5332 :		
SERVICE NAME: VishaySensor	Service		
TYPE STATE	: 10 WIN32_OWN_PROCESS : 2 START_PENDING (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)		
WIN32_EXIT_CODE SERVICE_EXIT_CODE CHECKPOINT WAIT_HINT PID FLAGS			
Press any key to continue			

### 9. Intel<sup>®</sup> Rapid Storage Technology (RST)

For additional information about Intel RST: <u>https://downloadcenter.intel.com/product/55005/IntelRapid-</u> <u>Storage-Technology-Intel-RST-</u>

Double Click [ SetupRST] 
 Dick [ Yes ]

<ul> <li>Image: RST_16.5.0.1021</li> <li>文件 主页 共享 查看</li> </ul>					□ × ~ 0
	> XP-3685 > User Manual > RST 16.5.0.1021	~ U	搜索*RST	16.5.0.102	
3 User_Manual	^ 名称	修改日期		美型	1
windows10	1 mup	2018/4/27	FF 10	XML 21	
Creative Cloud Files	🦛 SetupRST	2018/4/27	下午 10二	应用程序	
6 OneDrive	WixLicense	2018/4/27	FT 10_	文本文相	
<ul> <li>30 対象</li> <li>下載</li> <li>文档</li> <li>型 投稿</li> <li>2 投稿</li> <li>2 投稿</li> <li>2 段稿</li> <li>2 自乐</li> <li>■ 桌面</li> <li>2 OS (C)</li> </ul>					
→ 新加巻 (D:) → pm (\\192.168.0.5) (Z:)					



• Accept terms then click [ Next ] 

Click [ Next ]



• Choose save location and click [ Next ]





#### Click [ Next ]



• Click [ Install ] Choose to restart then click [ Finish ]





## 10. ADB (Android Debug Bridge)

This driver is only applicable the system on module (SOM) is installed

#### 📕 🛛 🔋 🔹 🛛 DriverAssitant\_v4.5 × 件 主页 共享 查看 C → → ↑ 📒 = User\_Manual > Links > windows10 > DriverAssitant\_v4.5 > ✓ ひ 提索\*DriverAssitant\_v4.5 P 名称 修改日期 类型 . 此电脑 ADBDriver 2018/11/25 下午 1\_ 文件夹 5 3D 对象 ▶ 下载 ■ 文档 bin 2018/11/25 下午 1\_ 文件夹 2018/11/25下午1\_ 2018/11/25下午1\_ 2018/11/25下午1\_ 文件夹文件夹 Driver Log 🍡 图片 2016/10/27上午1\_ 应用程序 ■ 视频 🤐 Dri nstall ● 音乐 ● 桌面 5- OS (C) \_\_ 新加卷 (D:) OVD RW 驱动器 (E:) Disc ≈x pm (\\192.168.0.5) (Z:) ② DVD RW 驱动器 (E:) Disc IO Board 主機 ~ < 7 个项目

#### Double Click on [ DriverInstall ]







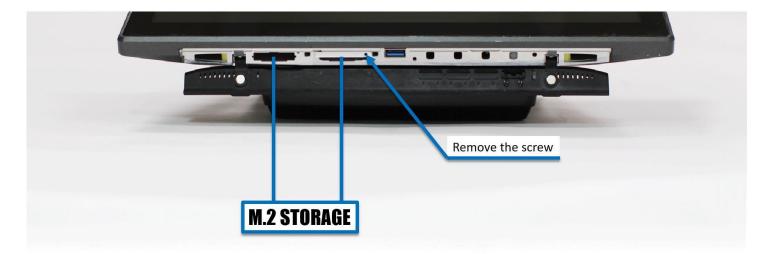
# Hardware

## **Access Storage Device**

- 1. Make sure the system is turned off (If using RAID system can be on)
- 2. Flip open the cover which is held together by magnets



3. Remove the screw(s)



If you are using RAID, follow the below step. If not, skip this step. Make sure to flip the small switch to the right ► to turn off the power to the storage device. After the M.2 is inserted, flip the switch back to on ◄.

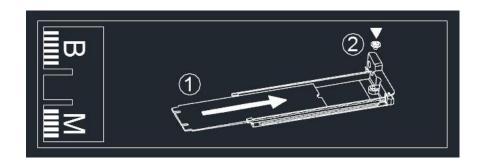


5. Flip open the handle and slowly pull out the M.2 Tray

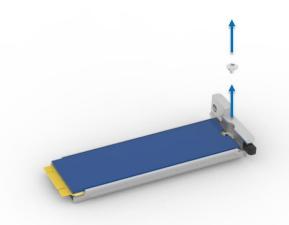


## Adding or removing M.2 from tray

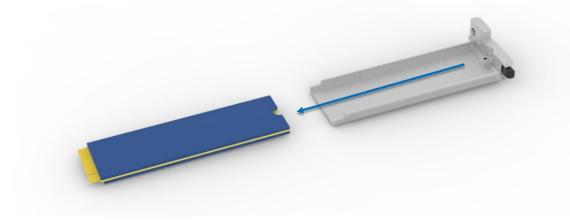
Note: This image is taped on the M.2 tray for users to identify which direction to place the M.2 module as well as how to install the M.2.



#### 1. Remove Screw



2. Slowly slide out the M.2 module



### Accessing the Motherboard

1. Remove the 3 screws as shown below



2. Remove the bottom plastic away from the aluminum chassis.



3. Slide the display module ~1cm as shown below

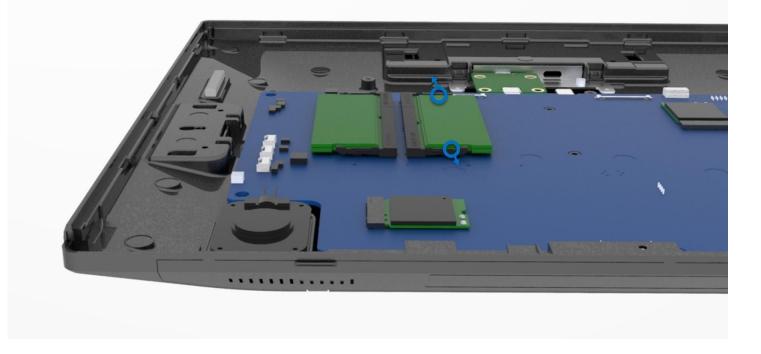


4. Lift the display up. There will be cables between the panel and motherboard. Reach your hand in and disconnect the cables on the motherboard.



## Memory

1. Slightly pull to the memory socket to the left and right. The memory will pop up





2. Remove the memory



# **3. IO Board SDK Instruction**

### 1. Overview

This document describes about how to use the FEC IO Board SDK to control IO functions on FEC IO Board via serial interface. The FEC IO Board SDK support the DLL and EXE interface controlling IO on windows application

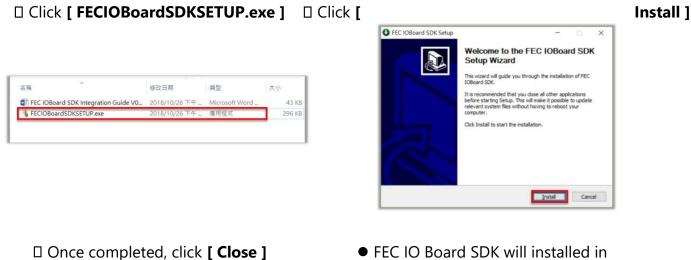
The IO functions are:

- COM A: RS232 Port A Enable/Disable, 5V/12V Setting
- COM B: RS232 Port B Enable/Disable, 5V/12V Setting
- Cash Drawer: Open, Get status
- USB Smart COM: Enable/Disable
- Reset SOM (Android System)
- Reset CCG4 USB Type-C Control IC

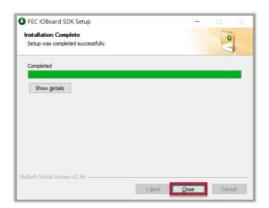
#### 2. System Requirement & Installation

Supported Operating System(OS) Microsoft<sup>®</sup> Windows 10 IOT Enterprise LTSB

Installation



 FEC IO Board SDK will installed in C:\Program Files(x86)\FEC\IOBoardSDK



< 本核	機磁碟 (C:) > Program Files (x86) > FEC >	IOBoardSDK v (	り 授尋 IOBoards	DK ,
^	名稱	僻改日期	類型	大小
	FEC_IOBoard_Utility.exe	2018/11/7 下午 0	應用程式	16 KB
	fec_xpos_ioboard_dll.dll	2018/9/25 下午 0	應用程式擴充	12 KB
	FEC_XPOS_IOBoard_Tester.exe	2018/9/25 下午 0	應用程式	15 KB
	😵 Uninstall.exe	2018/11/19下午	應用程式	160 KB

### 3. DLL Interface for FEC IO Board SDK

FEC IO Board SDK provide DLL interface to control IO Board, the DLL name is fec\_xpos\_ioboard\_dll.dll. API Functions & definitions

#define CTL_COM_MODE_RI	0x00
#define CTL_COM_MODE_DC	0x01
#define CTL_COM_PWR_5V	0x02
#define CTL_COM_PWR_12V	0x03
#define CTL_CASH_OUT_LOW	0x04
#define CTL_CASH_OUT_HIGH	0x05
#define CTL_CASH_OUT_HIGH_200MS	0x06
#define CTL_CASH_PWRSEL_12V	0x07
#define CTL_CASH_PWRSEL_24V	0x08
#define CTL_SOM_RST_LOW	0x09
#define CTL_SOM_RST_HIGH	0x10
#define CTL_TYPEC_FW_RS_LOW	0x11
#define CTL_TYPEC_FW_RS_HIGH	0x12
#define CTL_SMART_COM_BYPASS	0x13
#define CTL_SMART_COM_SMART_COM	0x14
DLLExport int SetComAMo	de(int mode)

This function enable/disable the COM Port A power supply

Parameter: int mode: CTL\_COM\_MODE\_RI (0x00): Disable COM A Power supply CTL\_COM\_MODE\_DC (0x01): Enable COM A Power supply Return Value: Fail: 1 Success: 0

• DLLExport int SetComAPwr(int pwrmode) This function set the COM Port A power level to 5V or 12V

Parameter: int pwrmode: CTL_COM_PWR_5V (0x02): Set to 5V CTL_COM_PWR_12V (0x03) : Set to 12V Return Value: Fail: 1 Success: 0
DLLExport int SetComBMode(int mode) This function enable/disable the COM Port B power supply
Parameter: int mode: CTL_COM_MODE_RI (0x00): Disable COM B Power supply CTL_COM_MODE_DC (0x01): Enable COM B Power supply Return Value: Fail: 1 Success: 0
DLLExport int SetComBPwr(int pwrmode) This function set the COM Port B power level to 5V or 12V
Parameter: int pwrmode: CTL_COM_PWR_5V (0x02): Set to 5V CTL_COM_PWR_12V (0x03) : Set to 12V Return Value: Fail: 1 Success: 0
DLLExport int SetCashDrawer1(int mode) This function opens the Cash Drawer GPIO 1
Parameter: int mode: CTL_CASH_OUT_HIGH_200MS (0x06): Open the cash drawer Return Value: Fail: 1 Success: 0
DLLExport int SetCashDrawer2(int mode)
This function opens the Cash Drawer GPIO 2
Parameter: int mode: CTL_CASH_OUT_HIGH_200MS (0x06): Open the cash drawer Return Value: Fail: 1 Success: 0
DLLExport int SetCashDrawerPwrSel(int pwrmode)

This function set the cash drawer power level

Parameter: int pwrmode: CTL\_CASH\_PWRSEL\_12V (0x07): Set to 12V CTL\_CASH\_PWRSEL\_24V (0x08) : Set to 24V Return Value: Fail: 1 Success: 0

DLLExport int GetCashDrawerStatus(BYTE \*byStatus) This function gets the cash drawer

status (Open or Close)

Parameter: BYTE \*byStatus: \*byStatus = 0: Close \*byStatus = 1: Open Return Value: Fail: 1 Success: 0

DLLExport int SetSomReset(int mode) This function reset the SOM Android system

Parameter:

int mode: Don't care (Set to 0~255) Return Value:

Fail: 1 Success: 0

DLLExport int SetSmartCom (int mode)

This function set the Smart COM enable or bypass

Parameter: int

mode:

CTL\_SMART\_COM\_BYPASS (0x13): Set Smart COM to bypass

CTL\_SMART\_COM\_SMART\_COM (0x14): Set Smart COM enable

Return Value:

Fail: 1

Success: 0

DLLExport int GetAllStatus(BYTE \*byStatus) This function gets all the GPIO pin status on

IO Board

Parameter:

BYTE \*byStatus: 0 = low, 1 = high byStatus [0], bit0 --> COMA\_MODE byStatus [0], bit1 --> COMA\_PWR byStatus [0], bit2 --> COMB\_MODE byStatus [0], bit3 --> COMB\_PWR byStatus [0], bit4 --> CashDrawer\_GPIO0 byStatus [0], bit5 --> CashDrawer\_GPIO1 byStatus [0], bit6 --> CASH\_PWRSEL byStatus [0], bit7 --> CASH\_IN byStatus [1], bit0 --> SOM\_RST byStatus [1], bit1 --> TYPEC\_FW\_RS byStatus [1], bit2 --> SMART\_COM Return Value: Fail: 1

□ DLLExport int GetFwVersion(BYTE \* byVersion) This function gets the FW version for IO Board FW.

byStatus [0]  $\rightarrow$  High byte of the FW version number byStatus [1]  $\rightarrow$  Low byte of the FW version number Return Value: Fail: 1

Success: 0

### 4. Example for DLL Interface

FEC\_IOBoard\_Utility: Please refer the sample workspace: "FEC\_IOBoard\_Utility" create by Visual Studio 2015 for the sample code.

The screenshot for FEC\_IOBoard\_Utility:

EC IOBoard Utility			- 0
lease check th	e IO status	ř	
Get FW	Get All Status		
COMA On	COMA Off		
COMA 5V	COMA 12V		
COMB On	COMB Off		
COMB 5V	COMB 12V		
USB COM to F	Port		
USB COM to	c		
CashDrawer 1	2V		
CashDrawer 2	4V		
CashDrawer1	Open		
CashDrawer2	Open		
CashDrawer S	tatus		
E	xit		

C#:

[DllImport("fec\_xpos\_ioboard\_dll", CharSet = CharSet.Unicode)] public static extern int SetComAMode(int mode); const int CTL\_COM\_MODE\_RI = 0x00; const int CTL\_COM\_MODE\_DC = 0x01; SetComAMode(CTL\_COM\_MODE\_RI); SetComAMode(CTL\_COM\_MODE\_DC);

#### 5. EXE Interface for FEC Cash Drawer SDK

Enable COM Port A Power

Set the COM Port A Power Enable with the parameter 1: "SetComAMode" & parameter 2: "enable" or "disable"

> FEC\_XPOS\_IOBoard\_Tester.exe SetComAMode enable

> FEC\_XPOS\_IOBoard\_Tester.exe SetComAMode disable

Enable / Disable the COM Port B Power

Set the COM Port B Power Enable / Disable with the parameter 1: "SetComBMode" & parameter 2:

"enable" or "disable"

> FEC\_XPOS\_IOBoard\_Tester.exe SetComBMode enable

> FEC\_XPOS\_IOBoard\_Tester.exe SetComBMode disable

Set the COM Port A Power level

Set the COM Port A Power to 5v / 12v with the parameter 1: "SetComAPwr" & parameter 2: "5v" or "12v"

> FEC XPOS IOBoard Tester.exe SetComAPwr 5v > FEC XPOS IOBoard Tester.exe SetComAPwr 12v Open the Cash Drawer 1 Open the Cash Drawer 1 with the parameter 1: "SetCashDrawer1" & parameter 2: "activate" > FEC XPOS IOBoard Tester.exe SetCashDrawer1 activate Open the Cash Drawer 2 Open the Cash Drawer 2 with the parameter 1: "SetCashDrawer2" & parameter 2: "activate" > FEC\_XPOS\_IOBoard\_Tester.exe SetCashDrawer2 activate Set the Cash Drawer Power Level Set the Cash Drawer Power Level to 12v/24v with the parameter 1: "SetCashDrawerPwrSel" & parameter 2: "12v" or "24v" > FEC XPOS IOBoard Tester.exe SetCashDrawerPwrSel 12v > FEC\_XPOS\_IOBoard\_Tester.exe SetCashDrawerPwrSel 24v Get the Cash Drawer Status Get the Cash Drawer Status with the parameter : "GetCashDrawerStatus" > FEC XPOS IOBoard Tester.exe GetCashDrawerStatus CashDrawer Status = 0 means close

CashDrawer Status = 1 means open

Reset the SOM Reset the SOM Android system with the parameter : "SetSomReset" > FEC XPOS IOBoard Tester.exe SetSomReset

Set the Smart COM enable/bypass

Set the Smart COM enable/bypass with the parameter 1: "SetSmartCom" & parameter 2: "enable" or "disable"

> FEC\_XPOS\_IOBoard\_Tester.exe SetSmartCom enable

> FEC\_XPOS\_IOBoard\_Tester.exe SetSmartCom disable

# 4. Software

## **Developers Guide for Serial Communications**

This document is intended for developers creating products based on the CP210x USB to UART Bridge Controller. It provides information about serial communications and how to obtain the port number for a specific CP210x device. Code samples are provided for opening, closing, configuring, reading, and writing to a COM port. Also included is a Get PortNum function that can be copied and used to determine the port number on a CP210x device by using its Vendor ID (VID), Product ID (PID), and serial number.

### **Opening a COM Port**

Before configuring and using a COM port to send and receive data, it must first be opened. When a COM port is opened, a handle is returned by the CreateFile() function that is used from then on for all communication. Here is example code that opens COM3:

```
HANDLE hMasterCOM = CreateFile("\\\\.\COM3",
GENERIC_READ | GENERIC_WRITE,
0,
0,
0,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
0);
```

The first parameter in the **CreateFile()** function is a string that contains the COM port number to use. This string will always be of the form **\\\\.\\COMX** where 'X' is the COM port number to use. The second parameter contains flags describing access, which will be **GENERIC\_READ** and **GENERIC\_WRITE** for the example in this document, and allows both read and write access. Parameters three and four must always be 0, and the flag in parameter five must always be **OPEN\_EXISTING** when using **CreateFile()** for COM applications. The sixth parameter should always contain the **FILE\_ATTRIBUTE\_NORMAL** flag. In addition, the **FILE\_FLAG\_OVERLAPPED** is an optional flag that is used when working with asynchronous transfers (this option is used for the example in this document). If overlapped mode is used, functions that read and write to the COM port must specify an OVERLAPPED structure identifying the file pointer, which is demonstrated in the sections **Purging the COM Port** and **Saving the COM Port's Original State** (more information on overlapped I/O is located at <u>https://msdn.microsoft.com/en-us/library/windows/desktop/ms686358(v=vs.85).aspx</u>). The seventh, and last, parameter must always be 0.

If this function returns successfully, then a handle to the COM port will be assigned to the HANDLE variable. If the function fails, **then INVALID\_HANDLE\_VALUE** will be returned. Upon return, check the handle and if it is valid, then prepare the COM port for data transmission.

### Preparing an Open COM Port for Data Transmission

Once a handle is successfully assigned to a COM port, several steps must be taken to set it up. The COM port must first be purged and its initial state should be retrieved. Then the COM port's new settings can be assigned and set up by a device control block (DCB) structure (more information is provided on the DCB structure in the section **Setting up a DCB Structure to Set the New COM State** and at <u>https://msdn.microsoft.com/en-us/library/windows/desktop/aa363214(v=vs.85).aspx</u>).

#### **Purging the COM Port**

First, the COM port should be purged to clear any existing data going to or from the COM port using the **PurgeComm()** function:

PurgeComm(hMasterCOM, PURGE\_TXABORT | PURGE\_RXABORT | PURGE\_TXCLEAR | PURGE\_RXCLEAR);

The first parameter in the **PurgeComm()** function is a handle to the open COM port that will be purged. The second parameter contains flags that further describe what actions should be taken. All four flags, **PURGE\_TXABORT, PURGE\_RXABORT, PURGE\_TXCLEAR,** and **PURGE\_RXCLEAR** should always be used. The first two flags terminate overlapped write and read operations, and the last two flags clear the output and input buffers.

If this function returns successfully then a non-zero value is returned. If the function fails, then it returns 0. Upon return, check the return value; if it is non-zero, continue to set up the COM port (more information on the PurgeComm() function is located at <a href="https://msdn.microsoft.com/enus/library/windows/desktop/aa363428(v=vs.85).aspx">https://msdn.microsoft.com/enus/library/windows/desktop/aa363428(v=vs.85).aspx</a> ).

#### Saving the COM Port's Original State

Since the COM port settings can be modified to meet different needs, it is good practice to obtain the COM port's current state and store it so that when the COM port is closed, the COM port can be restored back to its original state. This can be done using the **GetCommState()** function:

DCB dcbMasterInitState; GetCommState(hMasterCOM, &dcbMasterInitState);

The first parameter in the **GetCommState()** function is a handle to the open COM port to obtain settings from. The second parameter is an address to a DCB structure to store the COM port's settings. This DCB structure should also be used as the initial state when specifying new settings for the COM port (see section **Setting up a DCB Structure to Set the New COM State**). If this function returns successfully then a non-zero value is returned. If the function fails, then it returns 0. Upon return, check the return value; if it is non-zero, continue to set up the COM port (more information on

the **GetCommState()** function is located at <a href="https://msdn.microsoft.com/enus/library/windows/desktop/aa363260(v=vs.85).aspx">https://msdn.microsoft.com/enus/library/windows/desktop/aa363260(v=vs.85).aspx</a> ).

#### Setting up a DCB Structure to Set the New COM State

All of a COM port's settings are stored in a DCB structure. In section Saving the COM Port's Original State a DCB structure was retrieved that contained the initial settings of the COM port by using the **GetCommState()** function. To change a COM port's settings, a DCB structure must be created and filled out with the desired settings. Then the **SetCommState()** function can be used to activate those settings:

DCB dcbMaster = dcbMasterInitState;

```
dcbMaster.BaudRate = 57600; dcbMaster.Parity = NOPARITY; dcbMaster.ByteSize = 8; dcbMaster.StopBits = ONESTOPBIT;
```

SetCommState(hMasterCOM, &dcbMaster);

Delay(60);

Here a new DCB structure dcbMaster has been initialized to **dcbMasterInitState**, which are the current settings of the COM port. After it has been initialized to the current settings, new settings can be assigned.

#### **Baud Rate**

The baud rate property is set to 57600 bps, but can be set to any of the baud rates supported by the CP210x. (See the current datasheet for the list of supported baud rates for the CP210x.)

#### Parity

The parity is set to NOPARITY, however it can also be set to ODDPARITY, EVENPARITY, SPACEPARITY, and MARKPARITY if supported by the CP210x. (See the current data sheet for the list of supported parities for the CP210x.)

#### **Byte Size**

The byte size is set to 8, so there are 8 data bits in every byte of data sent. This can also be set to 5, 6, or 7 if supported by the

CP210x. (see the data sheet for the list of supported byte sizes for the CP210x.)

#### Stop Bits

The stop bits are set to ONESTOPBIT, but could also be set to TWOSTOPBITS or ONE5STOPBITS (1.5). (See the current data sheet for the list of supported stop bits for the CP210x.) All combinations of data and stop bits can be used except for the combination of 5 data bits with 2 stop bits and the combination of 6, 7, or 8 data bits with 1.5 stop bits. After each of these settings is set to the desired value, the **SetCommState()** function can be called to set up the COM port. The first parameter in the **SetCommState()** function is a handle to the open COM port to change the settings on. The second parameter is an address to a DCB structure containing the COM port's new settings (more information on serial settings using DCB structures is located at <a href="https://msdn.microsoft.com/enus/library/windows/desktop/aa363214(v=vs.85).aspx">https://msdn.microsoft.com/enus/library/windows/desktop/aa363214(v=vs.85).aspx</a>

If this function returns successfully, a non-zero value is returned. If the function fails, it returns 0. Upon return, check the return value; if it is non-zero, delay for 60 ms to allow time for the settings to change and then continue to set up the COM port. This delay is not required; however, a conservative time of 60 ms is good practice to ensure that the settings are changed before any other operations take place.

#### **Transmitting Data Across the COM Port**

Once the COM port is successfully opened and configured, data can be written or read.

#### Writing Data

There are several things that need to happen in a write, so it is a good idea to create a function for the writes to be called whenever a write must occur. Here is an example of a write function:

```
bool WriteData(HANDLE handle, BYTE* data, DWORD length, DWORD* dwWritten)
        bool success = false;
        OVERLAPPED o = \{0\};
        o.hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
        if (!WriteFile(handle, (LPCVOID)data, length, dwWritten, &o))
        {
                if (GetLastError() = = ERROR_IO_PENDING)
                                                               if
                (WaitForSingleObject(o.hEvent, INFINITE) == WAIT_OBJECT_0)
                       if (GetOverlappedResult(handle, &o, dwWritten, FALSE))
                                success = true;
        }
        Else
                success = true;
        if (*dwWritten != length)
                success = false:
        CloseHandle(o.hEvent);
        return success;
```

The parameters passed in to this function are the handle to an open COM port, a pointer to an array of bytes that will be written, the number if bytes that are in the array, and a pointer to a variable to store and return the number of bytes written. Two local variables are declared at the beginning of the function: a bool named success that will store the success of the write (this is initialized to false, and only set true when the write succeeds) and an overlapped object o which is passed to the **WriteFile()** function and alerts if the transfer is complete or not (this is always initialized to {0} before the **hEvent** is assigned). Creating an event with the **CreateEvent (NULL, FALSE, FALSE, NULL)** function sets the **hEvent** property of o to prepare it to be passed to the **WriteFile()** function (more information on CreateEvent() is located at

https://msdn.microsoft.com/enus/library/windows/desktop/ms682396(v=vs.85).aspx ).

Next, the **WriteFile()** function is called with the handle, data, length of the data, and variable to store the amount of data that was written (more information on **WriteFile()** is located at <u>https://msdn.microsoft.com/en-us/library/windows/desktop/aa365747(v=vs.85).aspx</u>). If this function returns successfully, a non-zero value is returned. If the function fails, it returns 0. The if statement will determine if the write succeeded and if it did not, the last error is retrieved to see if there really was an error or the write just wasn't finished. If **ERROR\_IO\_PENDING** is returned then object o is then

waited on until either the write finishes or fails (if something other than **ERROR\_IO\_PENDING** is returned by the **GetLastError()** function, then there is the possibility of surprise removal; see section **Application Design Notes** for comments on surprise removal). When the wait is over, the result is obtained so that the amount of bytes written is updated. The success variable will then be assigned with the appropriate value, and the handle of **o.hEvent** is closed. Then the amount of bytes written is checked, and finally the function returns the success of the write, which will be true if the write successfully completed.

#### **Reading Data**

There are several things that need to happen in a read, so it is a good idea to create a function for the reads to be called whenever a read must occur. Here is an example of a read function:

```
bool ReadData(HANDLE handle, BYTE* data, DWORD length, DWORD* dwRead, UINT timeout)
       bool success = false;
       OVERLAPPED o = \{0\};
       o.hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
       if (!ReadFile(handle, data, length, dwRead, &o))
       {
               if (GetLastError() == ERROR_IO_PENDING)
                                                                             if
(WaitForSingleObject(o.hEvent, timeout) == WAIT_OBJECT_0)
                               success = true:
                       GetOverlappedResult(handle, &o, dwRead, FALSE);
       }
       else
       success = true;
       CloseHandle(o.hEvent);
       return success;
```

The parameters passed in to this function are the handle to an open COM port, a pointer to an array of bytes that will be read, the number if bytes that are in the array, a pointer to a variable to store and return the number of bytes read, and a timeout value. Two local variables are declared at the beginning of the function: a bool named success that will store the success of the read (this is initialized to false, and only set true when the read succeeds), and an overlapped object o which is

passed to the **ReadFile()** function and alerts if the transfer is complete or not (this is always initialized to {0} before the **hEvent** is assigned). Creating an event with the **CreateEvent(NULL, FALSE, FALSE, NULL)** function sets the **hEvent** property of o to prepare it to be passed to the ReadFile() function (more information on **CreateEvent()** is located at

https://msdn.microsoft.com/enus/library/windows/desktop/ms682396(v=vs.85).aspx ).

Next, the **ReadFile()** function is called with the handle, data, length of the data, and variable to store the amount of data that was written (more information on the **ReadFile()** function is located at <u>https://msdn.microsoft.com/en-us/library/windows/desktop/aa365467(v=vs.85).aspx</u>). If this function returns successfully then a non-zero value is returned. If the function fails, then it will return 0. The if statement will determine if the write succeeded and if it didn't, the last error is retrieved to see if there really was an error or the write just wasn't finished. If **ERROR\_IO\_PENDING** is returned then object o is then waited on until either the write finishes or fails (if something other than **ERROR\_IO\_PENDING** is returned by the **GetLastError()** function, then there is the possibility of surprise removal; see section **Application Design Notes** for comments on surprise removal). When the wait is over, the result is obtained so that the amount of bytes read is updated. The success variable will then be assigned with the appropriate value, and the handle of **o.hEvent** is closed. Finally, the function returns the success of the read, which will be true if the read successfully completed.

### **Closing the COM Port**

After all communication is finished, then the COM port should then be closed. First, the COM port should be set back to its initial state, and then the handle to the COM port should be closed and set to an invalid handle. Example code is shown below:

SetCommState(hMasterCOM, &dcbMasterInitState);

Delay(60);

CloseHandle(hMasterCOM); hMasterCOM = INVALID\_HANDLE\_VALUE;

**The SetCommState()** function works the same as described in section **Setting up a DCB Structure to Set the New COM State**. A delay of 60 ms is used to make sure the settings have time to be set. Finally the device is closed using the **CloseHandle()** function. This function just takes in the handle of the COM port. After this function is called, it is important to set the variable to an **INVALID\_HANDLE\_VALUE**.

#### **Sample Program to Demonstrate Serial Communications**

Included in the AN197 software package is a directory named CP210xSerialTest which contains the source code and executables for a Visual Studio project that makes use of all the serial communication functions described in section **Preparing an Open COM Port for Data Transmission**, section **Transmitting Data Across the COM Port**, and section **Closing the COM Port**. The program is a basic dialog based application that accepts two COM port numbers, and then will send a test array of 64 bytes of data back and forth between them.

#### **Application Design Notes**

The functions used in sections Preparing an Open COM Port for Data Transmission, Transmitting Data Across the COM Port, and section Closing the COM Port are Windows COMM API functions. The examples provided are just samples of the recommended way of dealing with serial communication. For more specific information on these functions, see the MSDN website at:

https://msdn.microsoft.com/enus/library/ff802693.aspx.

It should also be noted that the **SetCommState()** function does not save the settings between opening and closing the COM port. As stated before, it is good practice to get the current settings after the COM port is opened, and then restore them before it is closed. All of the functions here will return an error code. It is a good idea to nest these functions in order to catch errors if they occur by using the **GetLastError()** function. This will also solve any surprise removal problems by allowing the discovery of an invalid handle to be found and dealt with. The example application (CP210xSerialTest) has several cases that will detect surprise removal. In this example, there are checks on every function to make sure that the return code is true. If it is not, then it will display where the error occurred in the output window. As long as correct and supported settings are passed to the functions they should execute normally. Most failures can occur from having an INVALID\_HANDLE\_VALUE, however, the handles must be set to this value after a surprise removal occurs. Because regular COM ports will always be visible, then data can always be written to them successfully, even if there is no way to read it. However, because the CP210x is a virtual COM port, if the device is removed, then the handle that it uses becomes invalid when trying to write to it. If for some reason the CP210x device is unplugged the write will fail and ERROR\_OPERATION\_ABORTED will be returned by GetLastError(). When this happens, the handle needs to be closed and then set to **INVALID\_HANDLE\_VALUE**. Alternatively, a regular COM port can always be read from, but if there is no data then it will time out. When using the CP210x as the virtual COM port and it is removed before a read occurs, then the read will fail and ERROR\_ACCESS\_DENIED will be returned by GetLastError().

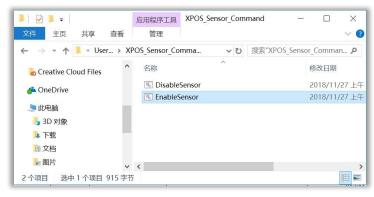
Again when this happens, the handle needs to be closed and then set to **INVALID\_HANDLE\_VALUE**.

## **Enable and Disable Ambient Light and Proximity Sensor**

There are three ways to enable or disable the 2in1 sensor.

#### 1. Utility Batch File

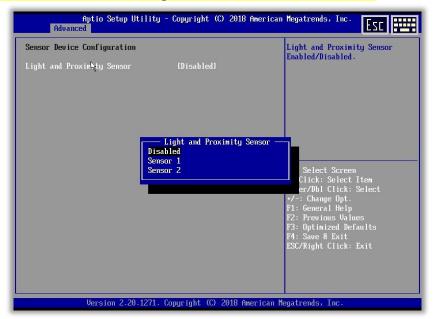
File is located in your Driver CD



### 2. BIOS

- 1. Start up the system
- 2. Press [ Delete ] during startup to enter BIOS
- 3. Under Advanded > Sensor Device Configuration click Disabled to turn off, Sensor 1 to turn on proximity and anmbient light sensor. Sensor 2 has no function.

Note: Sensor Service and BIOS Sensor settings both need to be on to work



#### 3. Windows 10 OS

1. Under Computer Management go under Services and Applications to click on [ Services ] 2. Click on [ Sensor Service ]

Computer Management (Local	O. Services	2				Actions	
<ul> <li>System Tools</li> <li>Task Scheduler</li> </ul>	Sensor Service	Name	Description	Status	Startu ^	Services	
> I lask Scheduler		Retail Demo Service	The Retail D		Manu	More Actions	
> pi Shared Folders	Start the service	Routing and Remote Access	Offers routi		Disabl	The second second second	
> A Local Users and Groups		RPC Endpoint Mapper	Resolves RP	Running	Autor	Sensor Service	
> ( Performance	Description:	Secondary Logon	Enables star		Manu	More Actions	
🛃 Device Manager	A service for sensors that manages	Secure Socket Tunneling Pr	Provides su		Manu		
Storage	different sensors' functionality. Manages Simple Device Orientation	Security Accounts Manager	The startup	Running	Autor		
📅 Disk Management	(SDO) and History for sensors. Loads	Security Center	The WSCSV	Running	Autor		
Services and Applications	the SDO sensor that reports device	🧠 Sensor Data Service	Delivers dat		Manu		
Services	orientation changes. If this service is stopped or disabled, the SDO sensor	🧟 Sensor Monitoring Service	Monitors va		Autor		
i WMI Control	will not be loaded and so auto-	Sensor Service	A service fo		Manu		
	rotation will not occur. History	Q Server	Supports fil	Running	Autor		
	collection from Sensors will also be	🥋 Shared PC Account Manager	Manages pr		Disabl		
	stopped.	Shell Hardware Detection	Provides no	Running	Autor		
		Smart Card	Manages ac		Disabl		
		Smart Card Device Enumera	Creates soft		Manu		
		Smart Card Removal Policy	Allows the s		Manu		
		Q, SNMP Trap	Receives tra		Manu		
		Software Protection	Enables the		Autor		
		Spot Verifier	Verifies pote		Manu		
		SSDP Discovery	Discovers n	Running	Manu		
		State Repository Service	Provides re	Running	Manu		
		Still Image Acquisition Events			Manu		
		Storage Service	Provides en	Running	Manu		
		Storage Tiers Management	Optimizes t		Manu		
		Superfetch	Maintains a	Running	Autor		
		Sync Host_2ebd6	This service	Running	Autor		
		System Event Notification S	Monitors sy	Running	Autor 🗸		

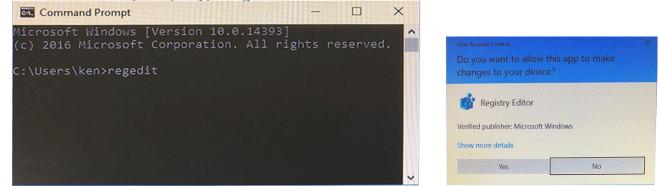
3. Sensor Services Properties make sure the "Startup type" is set to Automatic. Click

### [ Start ] [ Apply ] [ OK ]

General	Log On	Recovery	Dependencies		
Service	name:	SensorSer	vice		
Display	name:	Sensor Se	vice		
Descrip	tion:		or sensors that manages differ ty. Manages Simple Device Ori		
Path to	executab	le:			
C:\Win	dows\syst	em32\svcho	st.exe + LocalSystemNetwork	Restricted	
			St. CAO IN EDGGIOJSterini Chinon		
Startup	type:	Automatic		~	I
Startup	type:	Automatic		~	
Startup	type:	Automatic		~	
	type:	Automatic		~	
Service	e status:	Stopped		~	
Service		L		∼ Resume	
Service You ca	e status: Start n specify t	Stopped Stop		Resume	
Service	e status: Start n specify t	Stopped Stop	p Pause	Resume	
Service You ca from he	e status: Start n specify t	Stopped Stop	p Pause	Resume	

## **Control the Sensor Timing**

1. In the command prompt, type regedit and click enter



- Locate the following folder through this path HKEY\_LOCAL\_MACHINE > SOFTWARE > Microsoft > Windows NT > CurrentVersion > AdaptiveDisplayBrightness > {23B44AF278CE-4943-81DF-89817E8D23FD}
- 3. Click on CRI then the radio head decimal and change the number to the desired timing (ex 3000 is approximately 3 seconds, 10000 is around 10 seconds)

Edit DWORD (32-bit	t) Value 🛛 🗙
Value name:	
CRI	
Value data:	Base
3000	OHexadecimal
	Decimal
	OK Cancel
	OK Cancel

4. The sensor needs to be deactivated then reactivated under Computer Management > Sensor Services

le Action View Help								
🔿 🙋 📷 📷 🙆	🛙 📷 🕨 🖿 🖬 🕪						_	General Log On Recovery Dependencies
Computer Management (Local	O Services	4				Actions		and the second sec
Vistem Tools     Sensor Service     Sensor Service	Sensor Service	Name	Description	Status	Startu ^	Services		Service name: SensorService
		Retail Demo Service	The Retail D	J Mr	Manu	More Actions	•	Display name: Sensor Service
	Start the service	Routing and Remote Access	Resolves RP Running Enables star Provides su		Disabl		1000	Display name: Sensor Service
		RPC Endpoint Mapper		Running	Autor	Sensor Service	<b>•</b>	
No Performance	Description:	Secondary Logon		Manu	More Actions	- F	Description: A service for sensors that manages different sensors'	
🛃 Device Manager				Manu		functio	functionality. Manages Simple Device Orientation	
Storage		Security Accounts Manager	The startup		Autor			
<ul> <li>Services and Applications</li> <li>Services</li> <li>WMI Control</li> <li>WMI Control</li> <li>WMI control</li> </ul>		Security Center	The WSCSV Running	Autor			Path to executable:	
		Sensor Data Service	Delivers dat		Manu			C:\Windows\system32\svchost.exe + LocalSystemNetworkRestricted
	stopped or disabled, the SDO sensor	Sensor Monitoring Service	Monitors va		Autor			C. Windows Systemsz Svenost.exe ik Ebealoysteminetwork nesticied
	will not be leaded and so auto- rotation will not ecur. History collection from Sensors will also be stopped.	Server	A service fo	n :	Autor		Sta	Startup type: Automatic
		Shared PC Account Manager		nages pr vides no Running nages ac	Disabl			Startup type: Automatic ~
		Shell Hardware Detection			Autor			
		Smart Card	Manages ac		Disabl			
		Smart Card Device Enumera			Manu			
		Smart Card Removal Policy	Allows the s		Manu			
		SNMP Trap	Receives tra	Manu		Service status:	Service status: Stopped	
		Software Protection	Enables the		Autor			
		Spot Verifier	Verifies pote		Manu			Start Stop Pause Resume
		SSDP Discovery	Discovers n	Running	Manu			
		State Repository Service	Provides re	Running	Manu			You can specify the start parameters that apply when you start the service
		Still Image Acquisition Events	Launches a		Manu			from here.
		🗟 Storage Service	Provides en	Running	Manu			
		Storage Tiers Management	Optimizes t		Manu		_	Start parameters:
		Superfetch	Maintains a	Running	Autor			Stat parameters.
		Sync Host_2ebd6	This service	Running	Autor			
		System Event Notification S	Monitors sy	Running	Autor 🗸			
	Extended (Standard /	<			>			

## **Cash Drawer Command**

Note: It is recommended that developers use the chapter 3: IO Board SDK Instruction

Command:

CashDrawer output 1: A5 01 05 02 6B Return A5 01 08 00 5B is Open A5 01 08 01 5B is Closed

CashDrawer output 2: A5 01 06 02 6B Return A5 01 08 00 6B is Open A5 01 08 01 6B is Closed