IT8951 USB Programming Guide
(USB Command Based)

2016/11/14

Version v.0.4

Revision history

Revision History			
Version	**Date**	**Description**	**Modified by**
0.1	2016/03/18	Create document	Eric Su
0.2	2016/05/04	Correct CDB table of Display command 0x94 Added CBW related reference	Eric Su
0.3	2016/05/19	Added Load image to indicated Index buffer feature Added Display image from indicated index buffer feature	Eric Su
0.4	2016/11/14	Added 0xA5 - Fast Write Memory command for load full image Added 0xA7 – Auto Reset command for IT8951	Eric Su

Table of Contents

■ figures

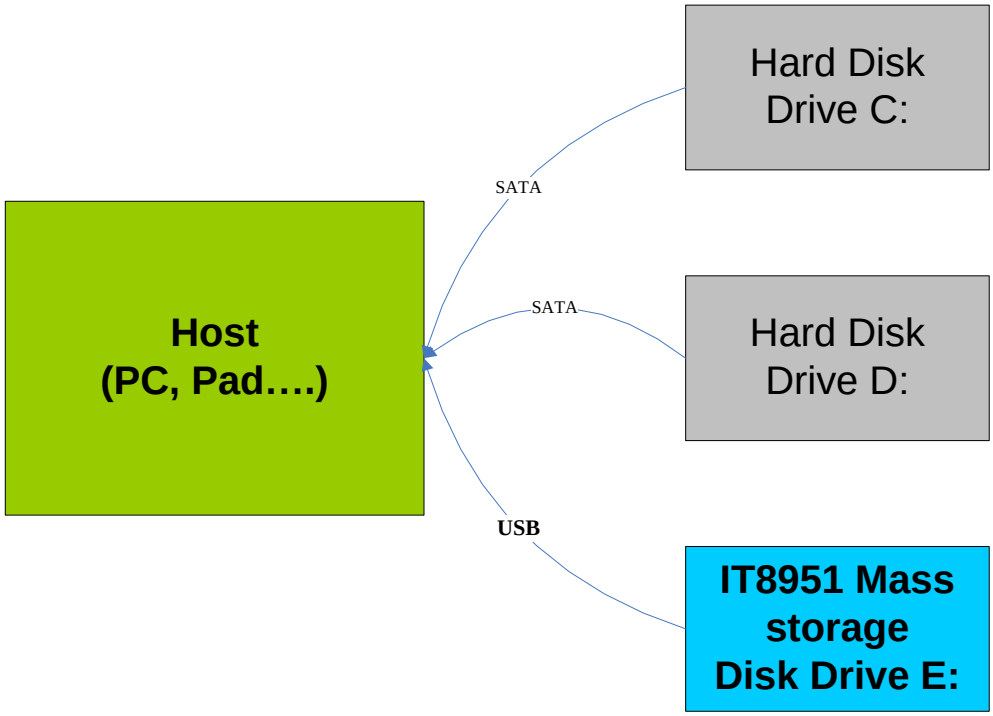
■ Tables

Chapter 1. IT8951 USB Device

1.1. Introduction for IT8951 Host programing through USB

IT8951 is a pure slave T-con chip and needs controlled by Host through specified host interface, it can support I80, SPI, I2C and also has USB interface when IT8951 OTG controller is configured to USB device mode, When IT8951 was plugged to Host via USB cable, it would report to a regular Mass storage device, therefore, Host programmer can send regular CDB command specified by SCSI to access IT8951 such as a general Disk of PC.

Figure 1 Block Diagram of IT8951 Device and Host



In this document, we will discuss how to program between Host and IT8951 by SCSI command via USB interface.

For some develop environment and platform (e.g. Android), the CDB[16] may need to be packed to CBW wrapper for Bulk transfer API as following table,

Command Block Wrapper

bit / Byte	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11 (08h-0Bh)	*dCBWDataTransferLength*							
12 (0Ch)	*bmCBWFlags*							
13 (0Dh)	Reserved (0)				*bCBWLUN*			
14 (0Eh)	Reserved (0)				*bCBWCBLength*			
15-30 (0Fh-1Eh)	*CBWCB* CDB[16]							

In this document, we will discuss and focus on IT8951 specified USB command in CDB[16] only, for more detailed description about CBW, please see http://www.usb.org/developers/docs/devclass_docs/usbmassbulk_10.pdf

e.g. - Android Bulk API (CBW needed)
http://developer.android.com/reference/android/hardware/usb/UsbDeviceConnection.html

For Example:

Send CBW[15] + CDB[16] + Write(Out) Data[28]

CBW[0]	0x55	Signature 'U'
CBW[1]	0x53	'S'
CBW[2]	0x42	'B'
CBW[3]	0x43	'C'
CBW[4]	0x00	Tag (Any value)
CBW[5]	0x00	
CBW[6]	0x00	
CBW[7]	0x00	
CBW[8]	**0x1C**	**Data Transfer Length[7:0](Bytes) (IN/OUT Data, CBW exclude)**
CBW[9]	**0x00**	**Data Transfer Length [15:8]**

CBW[10]	0x00	Data Transfer Length[23:16]
CBW[11]	0x00	Data Transfer Length[31:24]
CBW[12]	0x00	Direction – Bit[7] 0x80 -Bulk-In, 0x00 – Bulk out
CBW[13]	0x00	LUN – 0 (LUN of IT8951 is 0)
CBW[14]	0x10	CDB Length, we use 16 here
CDB[0]		See
CDB[1]		
……. ……		
CDB[14]		
CDB[15]		
WData[0]		
WData[1]		
…….. …...		
WData[27]		

1.1.1. Initial Flow of Host

As described above, Host will detect a Mass storage device when IT8951 connected to Host. Therefore, IT8951 will be assigned a Drive No. (e.g. **E:** , **F:** or…) ,

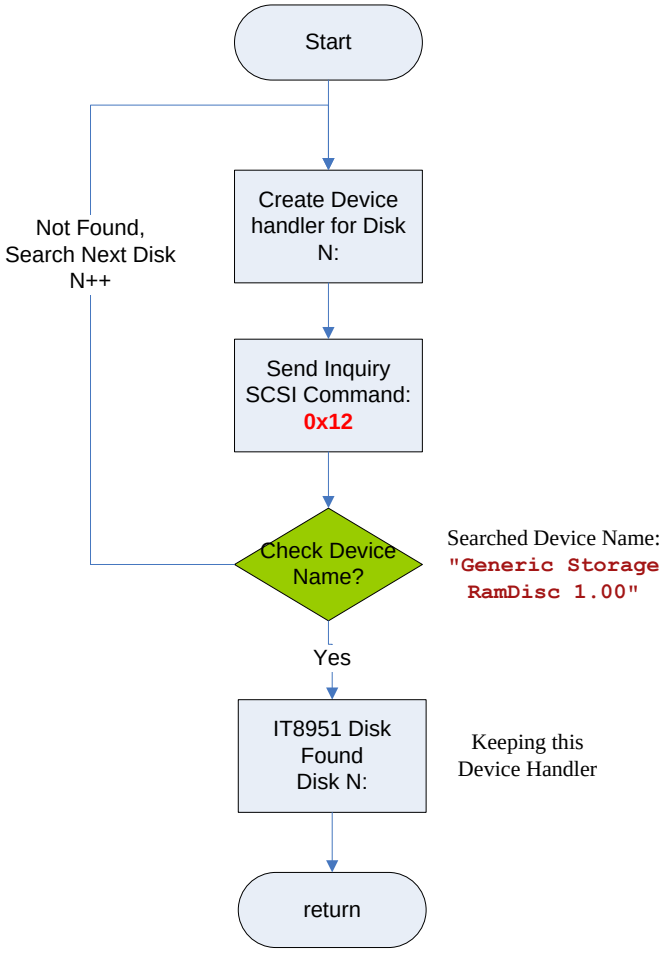
1 Searching Disk to find the Drive No of IT8951

2 Using inquiry command to check returned device name:

2.1 Send SCSI inquiry command

2.2 Comparing string from received buffer **recvbuf[8]~recvbuf[36]** is equal to `"Generic Storage RamDisc 1.00"`

Figure 2 Flow chart of Host Initial flow



■ SCSI Inquiry

- **Inquiry and get device name**
- **Direction: Bulk-In**
- **Programming flow**

1. Send CDB[16]

CDB[0]	**0x12**	Inquiry
CDB[1]	0x00	
CDB[2]	0x00	
CDB[3]	0x00	
CDB[4]	0x00	
CDB[5]	0x00	
CDB[6]	0x00	
CDB[7]	0x00	
CDB[8]	0x00	
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2. => parameters and data

No

3. <= Read inquiry data from IT8951

- Size : 40 bytes
- The data structure is defined as following:

Standard INQUIRY data format

bit Byte	7	6	5	4	3	2	1	0
0	Peripheral qualifier			Peripheral device type				
1	RMB	Device-type modifier						
2	ISO version		ECMA version			ANSI-approved version		
3	AENC	TrmIOP	Reserved		Response data format			
4	Additional length (n-4)							
5	Reserved							
6	Reserved							
7	RelAdr	WBus32	WBus16	Sync	Linked	Reserved	CmdQue	SftRe
8 15	(MSB) Vendor identification (LSB)							
16 31	(MSB) Product identification (LSB)							
32 35	(MSB) Product revision level (LSB)							
36 55	Vendor-specific							
56 95	Reserved							
96 n	Vendor-specific parameters Vendor-specific							

"Generic Storage RamDisc 1.00"

1.1.2. Get Device information from IT8951

Host need to do initial flow to get necessary information as following from IT8951 in initial step:

- Image Width
- Image Height
- Image Buffer Address (Index 0)
- Numbers of waveform modes and Temperature segments
- Numbers of Image buffer index (new added feature in **v.0.3**)

For more detailed description, please see USB command list in next chapter.

3 USB Command list

3.1 USB Host command introduction

In this chapter, we will introduce the IT8951 USB command in CDB format, host programmer can send the CDB with IT8951 specified command and data to control IT8951, for more detailed programming flow, please refer our sample code for Windows platform (without CBW)

3.2 New feature added in v.0.3

In previous version, there is only one image buffer address allocated in IT8951, In this version v.0.3, we added new feature for multiple image buffer for more flexible application for host programmer.

The numbers of maximum image buffer size are depends on Panel resolution and Memory available capacity.

3.3 New feature added in v.0.4

In this version, we added the following new commands:

- Fast write memory command **0xA5** which is for fast load full width image (continuous pixels data) and it the transfer rate may speed up to Max: 30MB/Secs.
- **0xA7** command for Auto reset IT8951
  - e.g. – After upgrading the IT8951 firmware via USB(PC) , host can send this command to Inform IT8951 auto reset that User has no need to reset IT8951 device manually.

■ **0x80 – GET_SYS**

- **Get System information**
- **Direction: Bulk-In**
- **Programming flow**

4. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	**0x38**	Signature[0] of "8951"
CDB[3]	**0x39**	Signature[1]
CDB[4]	**0x35**	Signature[2]
CDB[5]	**0x31**	Signature[3]

CDB[6]	0x80	**Get System information.**
CDB[7]	0x00	
CDB[8]	0x01	Version[8-11]: 0x00010002
CDB[9]	0x00	
CDB[10]	0x02	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

5. => parameters and data

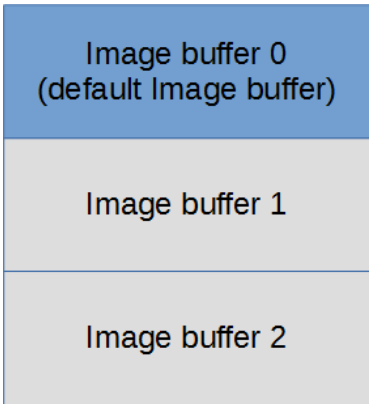
No

6. <= Read Device information data from IT8951

- Size : 112 bytes
- The data structure is defined as following:

```
typedef struct _TRSP_SYSTEM_INFO_DATA
{
       unsigned int uiStandardCmdNo; // Standard command number2T-con
       Communication Protocol
    unsigned int uiExtendCmdNo; // Extend command number
    unsigned int uiSignature; // 31 35 39 38h (8951)
    unsigned int uiVersion; // command table version
    unsigned int uiWidth; // Panel Width
    unsigned int uiHeight; // Panel Height
    unsigned int uiUpdateBufBase; // Update Buffer Address
    unsigned int uiImageBufBase; // Image Buffer Address(index 0)
    unsigned int uiTemperatureNo; // Temperature segment number
    unsigned int uiModeNo; // Display mode number
    unsigned int uiFrameCount[8]; // Frame count for each mode(8).
    unsigned int uiNumImgBuf; //Numbers of Image buffer
    unsigned int uiReserved[9]; // Don't care
    void* lpCmdInfoDatas[1]; // Command table pointer
} TRSP_SYSTEM_INFO_DATA;
```

For example, if we get uiNumImgBuf = 3
it means there are totally 3 image buffers (includes default image buffer) can be used currently, host programmer can load/display image to any one of these image buffers for some application such as preloading, previous page.., and the default image buffer (return by uiImageBufBase) would be allocated to index 0. For more detailed programming setting, please see IT8951 USB command 0x94(DisplayArea) and 0xA2(Load Image).



- **0x81 – READ_MEM**
  - **Read Memory function**
  - **Direction: <= Bulk-In**
  - **Programming flow**

1. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	**Addr[3]**	Memory Address[31:24] Big Endian for IT8951
CDB[3]	**Addr[2]**	Memory Address [23:16]
CDB[4]	**Addr[1]**	Memory Address [16:8]
CDB[5]	**Addr[0]**	Memory Address [7:0]
CDB[6]	**0x81**	Read Memory command
CDB[7]	**Len[1]**	Length[15:8]
CDB[8]	**Len[0]**	Length[7:0]
CDB[9]	0x00	
CDB[10]	0x00	

CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2. => parameters and data

No

3. <= Read Length: N of Memory data from IT8951

Size and Read Memory address are indicated by Host.

■ 0x82 – WRITE_MEM

- **Write Memory function**
- **Direction: => Bulk-Out**
- **Programming flow**

1. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	**Addr[3]**	Memory Address[31:24] Big Endian for IT8951
CDB[3]	**Addr[2]**	Memory Address [23:16]
CDB[4]	**Addr[1]**	Memory Address [16:8]
CDB[5]	**Addr[0]**	Memory Address [7:0]
CDB[6]	**0x82**	Write Memory command
CDB[7]	**Len[1]**	Length[15:8]
CDB[8]	**Len[0]**	Length[7:0]
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2. => Write N data (N= Length[15:0] in above)

WDataBuf[0]~ WDataBuf[N-1]

Size and Write Memory address are indicated by Host.

■ **0x94 – DPY_AREA**

- **Display Area function**
- **Direction: => Bulk-Out**
- **Programming flow**

1 Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	0x00	
CDB[3]	0x00	
CDB[4]	0x00	
CDB[5]	0x00	
CDB[6]	**0x94**	Display Area command
CDB[7]	0x00	
CDB[8]	0x00	
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2 Write Arguments - 7 Double Words (28 bytes)

- New Features

In this version, we added new feature that host programmer can select address mode(originally) or Index mode to indicate current loaded/displayed image buffer, for new Arguments setting, please refer as following:

- **If you want to use address mode as previous:**

Bit[31] = 0 – Address mode (originally)

Bit[30:0] – the Indicated Image buffer Address for displaying

e.g. – load image to address: 0x12345678

Set Arg[0~3] to Image Buffer Addr = 0x12345678 as usual for loading

image to indicated memory address.

Arg[0]	**MemAddr[3]**	**0x12**
Arg[1]	MemAddr[2]	0x34
Arg[2]	MemAddr[1]	0x56
Arg[3]	MemAddr[0]	0x78

- **If you want to use Index mode of Image buffer, please try to set image buffer Addr**

  Bit[31] = 1 – index mode

  Bit[30:4] - reserved

  Bit[3:0] – indicate index of Image buffer for displaying

  e.g. – load image to index 1 of Image buffer

  Set Arg[0~3] Image Buffer Addr = **0x80000001**

Arg[0]	**MemAddr[3]**	**0x80 – Enable Index mode**
Arg[1]	MemAddr[2]	0x00
Arg[2]	MemAddr[1]	0x00
Arg[3]	MemAddr[0]	0x01 – Index 1

Write Arguments - 7 Double Words (28 bytes)

Arg[0]	**MemAddr[3]**	**Image Buffer Addr [31:24] Big Endian**
Arg[1]	MemAddr[2]	Image Buffer Addr [23:16]
Arg[2]	MemAddr[1]	Image Buffer Addr [15:8]
Arg[3]	MemAddr[0]	Image Buffer Addr [7:0]
Arg[4]	**Mode[3]**	**Display Mode [31:24] Big Endian**
Arg[5]	Mode[2]	Display Mode [23:16]
Arg[6]	Mode[1]	Display Mode [15:8]
Arg[7]	Mode[0]	Display Mode [7:0]
Arg[8]	**DpyX[3]**	**Display X [31:24] Big Endian**
Arg[9]	DpyX[2]	Display X [23:16]
Arg[10]	DpyX[1]	Display X [15:8]
Arg[11]	DpyX[0]	Display X [7:0]
Arg[12]	**DpyY[3]**	**Display Y [31:24] Big Endian**
Arg[13]	DpyY[2]	Display Y [23:16]
Arg[14]	DpyY[1]	Display Y [15:8]
Arg[15]	DpyY[0]	Display Y [7:0]

Arg[16]	**DpyW[3]**	**Display Width [31:24] Big Endian**
Arg[17]	DpyW[2]	Display Width [23:16]
Arg[18]	DpyW[1]	Display Width [15:8]
Arg[19]	DpyH[0]	Display Width [7:0]
Arg[20]	**DpyH[3]**	**Display** Height **[31:24] Big Endian**
Arg[21]	DpyH[2]	Display Height [23:16]
Arg[22]	DpyH[1]	Display Height [15:8]
Arg[23]	DpyW[0]	Display Width [7:0]
Arg[24]	**EnWaitRdy[3]**	**EnWait Display Ready [31:24] Big Endian**
Arg[25]	**EnWaitRdy**[2]	**EnWaitRdy** [23:16]
Arg[26]	**EnWaitRdy**[1]	**EnWaitRdy** [15:8]
Arg[27]	**EnWaitRdy**[0]	**EnWaitRdy** [7:0]

■ 0xA2 – LD_IMG_AREA

- **Host Load Image Area function**
- **Direction: => Bulk-Out**
- All of the pixels need to be stored in 8bpp format by IT8951 as following:

0x00	Gray 0 (Black)	**0x40**	Gray 4	**0x80**	Gray 8	**0xC0**	Gray 12
0x10	Gray 1	**0x50**	Gray 5	**0x90**	Gray 9	**0xD0**	Gray 13
0x20	Gray 2	**0x60**	Gray 6	**0xA0**	Gray 10	**0xE0**	Gray 14
0x30	Gray 3	**0x70**	Gray 7	**0xB0**	Gray 11	**0xF0**	Gray 15 (white)

- **Programming flow**

1. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	0x00	
CDB[3]	0x00	
CDB[4]	0x00	
CDB[5]	0x00	
CDB[6]	**0xA2**	Load Image area command
CDB[7]	0x00	

CDB[8]	0x00	
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2. Write Argument - 5 Double Words (20 bytes)

Bit[31] = 0 – Address mode (originally)

Bit[30:0] – the Indicated Image buffer Address

e.g. – load image to address:0x12345678

Set Arg[0~3] to Image Buffer Addr = 0x12345678 as usual for loading image to indicated memory address.

Arg[0]	**Addr[3]**	**0x12**
Arg[1]	Addr[2]	0x34
Arg[2]	Addr[1]	0x56
Arg[3]	Addr[0]	0x78

Bit[31] = 1 – index mode

Bit[30:4] - reserved

Bit[3:0] – indicate index of Image buffer

e.g. – load image to index 1 of Image buffer

Set Arg[0~3] Image Buffer Addr = 0x80000001

Arg[0]	**Addr[3]**	**0x80 – Enable Index mode**
Arg[1]	Addr[2]	0x00
Arg[2]	Addr[1]	0x00
Arg[3]	Addr[0]	0x01 – Index 1

Arguments - 5 Double Words (20 bytes)

Arg[0]	**Addr[3]**	**Image buffer Addr [31:24] Big Endian**
Arg[1]	Addr[2]	Image buffer Addr [23:16]
Arg[2]	Addr[1]	Image buffer Addr [15:8]
Arg[3]	Addr[0]	Image buffer Addr [7:0]
Arg[4]	**DpyX[3]**	**Ld Image X [31:24] Big Endian**

Arg[5]	DpyX[2]	Ld Image X [23:16]
Arg[6]	DpyX[1]	Ld Image X [15:8]
Arg[7]	DpyX[0]	Ld Image X [7:0]
Arg[8]	**DpyY[3]**	**Ld Image Y [31:24] Big Endian**
Arg[9]	DpyY[2]	Ld Image Y [23:16]
Arg[10]	DpyY[1]	Ld Image Y [15:8]
Arg[11]	DpyY[0]	Ld Image Y [7:0]
Arg[12]	**DpyW[3]**	**Ld Image Width [31:24] Big Endian**
Arg[13]	DpyW[2]	Ld Image Width [23:16]
Arg[14]	DpyW[1]	Ld Image Width [15:8]
Arg[15]	DpyH[0]	Ld Image Width [7:0]
Arg[16]	**DpyH[3]**	**Ld Image** Height **[31:24] Big Endian**
Arg[17]	DpyH[2]	Ld Image Height [23:16]
Arg[18]	DpyH[1]	Ld Image Height [15:8]
Arg[19]	DpyW[0]	Ld Image Width [7:0]

3. Send Image (Size N = Image Width * Image Height)

The sent image will be collected by IT8951 whatever Host sends partial or full image.

Data[0]	**Pixel[0]**	**Image Pixel – 8bpp (Raw Data)**
Data[1]	Pixel[1]	**Image Pixel – 8bpp**
Data[2]	Pixel[2]	**Image Pixel – 8bpp**
Data[] ….	Pixel[]	**Image Pixel – 8bpp** …….
Data[N-1]	Pixel[N-1]	Last sent Pixel N =Image Width x Image height

■ 0xA3 – PMIC control

- **PMIC control for switching power on/off sequence**
- **Direction: => Bulk-Out**

- **Programming flow**

4. Send CDB[16]

CDB[0]	**0xFE**	Customer command

CDB[1]	0x00	
CDB[2]	0x00	
CDB[3]	0x00	
CDB[4]	0x00	
CDB[5]	0x00	
CDB[6]	**0xA3**	PMIC Control command
CDB[7]	0x00	Set VCom Value [15:8]
CDB[8]	0x00	Set VCom Value [7:0]
CDB[9]	0x00	Do Set VCom? (0 – no, 1 – Do)
CDB[10]	0x00	Do Power on/off? (0 –no, 1- Do)
CDB[11]	0x00	Power on/off 0 – off, 1 - on
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

- **Set VCom Value**
  - CDB[9] must be set to 1
  - Unit: mV
  - E.g. 2500 => -2500 mV = -2.5V
    - CDB[7] = 0x09
    - CDB[8] = 0xC4
- **Set Power on/off sequence**
  - CDB[10] must be set to 1
  - CDB[11]:
    - 0 – power off
    - 1 – power on
- e.g. Both set Power on and Set VCom
  - CDB[7] = 0x09
  - CDB[8] = 0xC4
  - CDB[9] = 1
  - CDB[10] = 1
  - CDB[11] = 1

- **0xA5 – FAST_WRITE_MEM**
  - **Fast Write Memory function**

- New enhanced command based on 0x82 Memory Write command

■ **Direction: => Bulk-Out**

■ **Programming flow**

1. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	**Addr[3]**	Memory Address[31:24] Big Endian for IT8951
CDB[3]	**Addr[2]**	Memory Address [23:16]
CDB[4]	**Addr[1]**	Memory Address [16:8]
CDB[5]	**Addr[0]**	Memory Address [7:0]
CDB[6]	**0xA5**	Fast Write Memory command
CDB[7]	**Len[1]**	Length[15:8]
CDB[8]	**Len[0]**	Length[7:0]
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

2. => Write N data (N= Length[15:0] in above)

WDataBuf[0]~ WDataBuf[N-1]

Size and Write Memory address are indicated by Host.

■ **0xA7 – AUTO_RESET**

■ **Auto reset IT8951 function**

- Host may send this command to inform IT8951 Auto Reset after fw upgrading process through USB.

■ **Direction: => Bulk-Out**

■ **Programming flow**

1. Send CDB[16]

CDB[0]	**0xFE**	Customer command
CDB[1]	0x00	
CDB[2]	0x00	

CDB[3]	0x00	
CDB[4]	0x00	
CDB[5]	0x00	
CDB[6]	**0xA7**	Auto Reset command
CDB[7]	0x00	
CDB[8]	0x00	
CDB[9]	0x00	
CDB[10]	0x00	
CDB[11]	0x00	
CDB[12]	0x00	
CDB[13]	0x00	
CDB[14]	0x00	
CDB[15]	0x00	

No parameters

4 Reference and Sample Code

4.1 USB Sample Code on Windows

Platform: Windows library

```
#include "winbase.h"
#include "Ntddscsi.h"
#include "winioctl.h"
```

https://msdn.microsoft.com/zh-tw/library/windows/desktop/aa363216(v=vs.85).aspx

```
#define SPT_BUF_SIZE  (60*1024)
```

Note: the Maximum transfer size is **60K** bytes for IT8951 USB
If the total transfer size would be over it, please divide them to multiple 60K transactions to complete the whole data transfer.

- **Initial Flow – Get Device information**
  - Command: 0x80
  - Return 112 bytes data

```
//*****************************************************
//  Structure of Get Device Information
//*****************************************************
typedef struct _TRSP_SYSTEM_INFO_DATA
{
    unsigned int uiStandardCmdNo; // Standard command number2T-con
Communication Protocol
    unsigned int uiExtendCmdNo; // Extend command number
    unsigned int uiSignature; // 31 35 39 38h (8951)
    unsigned int uiVersion; // command table version
    unsigned int uiWidth; // Panel Width
    unsigned int uiHeight; // Panel Height
    unsigned int uiUpdateBufBase; // Update Buffer Address
    unsigned int uiImageBufBase; // Image Buffer Address
    unsigned int uiTemperatureNo; // Temperature segment number
    unsigned int uiModeNo; // Display mode number
    unsigned int uiFrameCount[8]; // Frame count for each mode(8).
    unsigned int uiNumImgBuf; //Numbers of Image buffer
    unsigned int uiReserved[9]; // Don't care
```

```
    void* lpCmdInfoDatas[1]; // Command table pointer
} TRSP_SYSTEM_INFO_DATA;
```

```
DWORD IT8951GetSystemInfoAPI(_TRSP_SYSTEM_INFO_DATA* pstSystemInfo)
{
    DWORD dwReturnBytes = 0;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;
    BYTE bSuccess;
    unsigned int  i;
    unsigned int* pi = (unsigned int*)gSPTDataBuf;

    pstSPTDBuf->Length           = sizeof(SCSI_PASS_THROUGH_DIRECT);
    pstSPTDBuf->ScsiStatus        = 0;
    pstSPTDBuf->PathId           = 0;
    pstSPTDBuf->TargetId          = 0;
    pstSPTDBuf->Lun              = 0;
    pstSPTDBuf->CdbLength         = 16;
    pstSPTDBuf->SenseInfoLength    = 0;
    pstSPTDBuf->SenseInfoOffset    = 0;
    pstSPTDBuf->DataIn           = 1;
    pstSPTDBuf->DataTransferLength = 0x70;//112
    pstSPTDBuf->TimeOutValue      = 5;
    pstSPTDBuf->DataBuffer        = gSPTDataBuf;
    pstSPTDBuf->SenseInfoLength    = 0;

    //Copy Scsi Buffer to SPT Buf
    pstSPTDBuf->Cdb[0] = 0xFE;
    pstSPTDBuf->Cdb[1] = 0x00;
    pstSPTDBuf->Cdb[2] = 0x38;//Signature
    pstSPTDBuf->Cdb[3] = 0x39;
    pstSPTDBuf->Cdb[4] = 0x35;
    pstSPTDBuf->Cdb[5] = 0x31;
    pstSPTDBuf->Cdb[6] = 0x80;
    pstSPTDBuf->Cdb[7] = 0x00;//Version
```

```
    pstSPTDBuf->Cdb[8] = 0x01;
    pstSPTDBuf->Cdb[9] = 0x00;
    pstSPTDBuf->Cdb[10] = 0x02;
    pstSPTDBuf->Cdb[11] = 0x00;
    pstSPTDBuf->Cdb[12] = 0x00;
    pstSPTDBuf->Cdb[13] = 0x00;
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    bSuccess = DeviceIoControl(hDev,
                               IOCTL_SCSI_PASS_THROUGH_DIRECT,
                               &stSPTDBuf,
                               sizeof(SCSI_PASS_THROUGH_DIRECT),
                               &stSPTDBuf,
                               sizeof(SCSI_PASS_THROUGH_DIRECT),
                               &dwReturnBytes,
                               NULL );


    memcpy(pstSystemInfo, gSPTDataBuf,
sizeof(_TRSP_SYSTEM_INFO_DATA));
    //Endian Convert (Big => Little)
    pi = (unsigned int*)pstSystemInfo;
    for(i=0;i<sizeof(_TRSP_SYSTEM_INFO_DATA)/sizeof(unsigned
int);i++)
    {
       pi[i] = SWAP_32(pi[i]);
    }
    //Update Panel width and Height
    PANEL_W = pstSystemInfo->uiWidth;
    PANEL_H = pstSystemInfo->uiHeight;


    return 1;
}
```

```
_TRSP_SYSTEM_INFO_DATA gstIT8951SysInfo;
IT8951GetSystemInfoAPI(&gstIT8951SysInfo);//Get Device Info
```

- **Example 1 - IT8951 Write Memory**
  - Command: 0x82 – Regular Write Memory
  - Command: 0xA5 - Fast Write Memory (it needs IT8951 FW support)

```
DWORD IT8951WriteMemAPI(DWORD ulMemAddr, WORD usLength, BYTE* pSrcBuf)
{
    DWORD ulRetCode;
    DWORD dwReturnBytes = 0;
    BYTE  bSuccess;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;


    pstSPTDBuf->Length    = sizeof(SCSI_PASS_THROUGH_DIRECT);
    pstSPTDBuf->ScsiStatus        = 0;
    pstSPTDBuf->PathId           = 0;
    pstSPTDBuf->TargetId          = 0;
    pstSPTDBuf->Lun              = 0;
    pstSPTDBuf->CdbLength         = 16;
    pstSPTDBuf->SenseInfoLength    = 0;
    pstSPTDBuf->SenseInfoOffset    = 0;
    pstSPTDBuf->DataIn           = SCSI_IOCTL_DATA_OUT;//Out
    pstSPTDBuf->DataTransferLength = usLength;//DWORD R/W Access
    pstSPTDBuf->TimeOutValue      = 5;
    pstSPTDBuf->DataBuffer        = (void*)pSrcBuf;
    pstSPTDBuf->SenseInfoOffset    = 0;

    //Copy Scsi Buffer to SPTD Buf
    pstSPTDBuf->Cdb[0]  = 0xFE;
    pstSPTDBuf->Cdb[1]  = 0x00;
    pstSPTDBuf->Cdb[2]  = (BYTE)((ulMemAddr >> 24) & 0xFF); //Byte 2~5
Register Address BigEndian for IT8951
    pstSPTDBuf->Cdb[3]  = (BYTE)((ulMemAddr >> 16) & 0xFF);
    pstSPTDBuf->Cdb[4]  = (BYTE)((ulMemAddr >> 8) & 0xFF);
```

```
    pstSPTDBuf->Cdb[5]  = (BYTE)((ulMemAddr) & 0xFF);
#ifdef EN_FAST_WRITE_MEM//{IT8951_USB_OP_FAST_WRITE_MEM
    pstSPTDBuf->Cdb[6]  = IT8951_USB_OP_FAST_WRITE_MEM;//0xA5 for Fast
Write Memory(Need FW support);
#else
    pstSPTDBuf->Cdb[6]  = IT8951_USB_OP_WRITE_MEM;//0x82
#endif//}IT8951_USB_OP_FAST_WRITE_MEM
    pstSPTDBuf->Cdb[7]  = (BYTE)((usLength >> 8) & 0xFF);
    pstSPTDBuf->Cdb[8]  = (BYTE)((usLength) & 0xFF);
    pstSPTDBuf->Cdb[9]  = 0x00;
    pstSPTDBuf->Cdb[10] = 0x00;
    pstSPTDBuf->Cdb[11] = 0x00;
    pstSPTDBuf->Cdb[12] = 0x00;
    pstSPTDBuf->Cdb[13] = 0x00;
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    //Send SCSI Command
    bSuccess = DeviceIoControl(hDev,

IOCTL_SCSI_PASS_THROUGH_DIRECT,//IOCTL_SCSI_PASS_THROUGH_DIRECT,//IOC
TL_SCSI_PASS_THROUGH,
                           &stSPTDBuf,
                           sizeof(SCSI_PASS_THROUGH_DIRECT),
//sizeof( TSPTWBData),
                           &stSPTDBuf,

sizeof(SCSI_PASS_THROUGH_DIRECT),//sizeof( TSPTWBData),
                         &dwReturnBytes,
                           NULL );

    if(!bSuccess)
    {
        ulRetCode = GetLastError();
    }
    return bSuccess;
}
```

- **Example 2 - IT8951 Read Memory**
  - Command: 0x83

```
DWORD IT8951ReadMemAPI(DWORD ulMemAddr, WORD usLength, BYTE* RecvBuf)
{
    BYTE  bSuccess;
    DWORD dwReturnBytes = 0;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;

    pstSPTDBuf->Length           = sizeof(SCSI_PASS_THROUGH_DIRECT);
    pstSPTDBuf->ScsiStatus        = 0;
    pstSPTDBuf->PathId           = 0;
    pstSPTDBuf->TargetId          = 0;
    pstSPTDBuf->Lun              = 0;
    pstSPTDBuf->CdbLength         = 16;
    pstSPTDBuf->SenseInfoLength    = 0;
    pstSPTDBuf->SenseInfoOffset    = 0;
    pstSPTDBuf->DataIn           = 1;//Read - 1
    pstSPTDBuf->DataTransferLength = usLength;
    pstSPTDBuf->TimeOutValue      = 5;
    pstSPTDBuf->DataBuffer        = (void*)RecvBuf;
    pstSPTDBuf->SenseInfoOffset    = 0;

    //CDB - SCSI Request Sense Command
    pstSPTDBuf->Cdb[0]  = 0xFE;
    pstSPTDBuf->Cdb[1]  = 0x00;
    pstSPTDBuf->Cdb[2]  = (BYTE)((ulMemAddr >> 24) & 0xFF); //Byte 2~5
Register Address BigEndian for IT8951
    pstSPTDBuf->Cdb[3]  = (BYTE)((ulMemAddr >> 16) & 0xFF);
    pstSPTDBuf->Cdb[4]  = (BYTE)((ulMemAddr >> 8) & 0xFF);
    pstSPTDBuf->Cdb[5]  = (BYTE)((ulMemAddr) & 0xFF);
    pstSPTDBuf->Cdb[6]  = IT8951_USB_OP_READ_MEM;//0x81
    pstSPTDBuf->Cdb[7]  = (BYTE)((usLength >> 8) & 0xFF);
    pstSPTDBuf->Cdb[8]  = (BYTE)((usLength) & 0xFF);
```

```
    pstSPTDBuf->Cdb[9]  = 0x00;
    pstSPTDBuf->Cdb[10] = 0x00;
    pstSPTDBuf->Cdb[11] = 0x00;
    pstSPTDBuf->Cdb[12] = 0x00;
    pstSPTDBuf->Cdb[13] = 0x00;
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    bSuccess = DeviceIoControl(hDev,
                        IOCTL_SCSI_PASS_THROUGH_DIRECT,
                        &stSPTDBuf,
                        sizeof(SCSI_PASS_THROUGH_DIRECT),
                        &stSPTDBuf,
                        sizeof(SCSI_PASS_THROUGH_DIRECT),
                        &dwReturnBytes,
                        NULL );

    //Return 4 bytes Read Value in Big Endian format
    return 1;
}
```

- **Example 2 - IT8951 Load Image Area**
  - Command: 0xA2

```
typedef struct _LOAD_IMG_AREA_
{
    int     iAddress; //Bit[31] = 1, index Mode, 0 - Address mode
    int     iX;
    int     iY;
    int     iW;
    int     iH;
} LOAD_IMG_AREA;
```

```
DWORD IT8951LdImgAreaAPI(LOAD_IMG_AREA* pstLdImgArea, BYTE* pSrcBuf)
{
```

```
    DWORD ulRetCode;

    DWORD dwReturnBytes = 0;
    BYTE  bSuccess;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;
    WORD usLength;
    int i;

    //Set Image Length for this transfer
    usLength = (WORD)(pstLdImgArea->iW*pstLdImgArea->iH);

    pstSPTDBuf->Length           = sizeof(SCSI_PASS_THROUGH_DIRECT);
    pstSPTDBuf->ScsiStatus        = 0;
    pstSPTDBuf->PathId           = 0;
    pstSPTDBuf->TargetId          = 0;
    pstSPTDBuf->Lun              = 0;
    pstSPTDBuf->CdbLength         = 16;
    pstSPTDBuf->SenseInfoLength    = 0;
    pstSPTDBuf->SenseInfoOffset    = 0;
    pstSPTDBuf->DataIn           = SCSI_IOCTL_DATA_OUT;//Out
    pstSPTDBuf->DataTransferLength = sizeof(LOAD_IMG_AREA) +
usLength;//DWORD R/W Access
    pstSPTDBuf->TimeOutValue      = 5;
    pstSPTDBuf->DataBuffer        = (void*)gSPTDataBuf;
    pstSPTDBuf->SenseInfoOffset    = 0;

    //Little Endian to Big for IT8951/61
    for(i=0;i<sizeof(LOAD_IMG_AREA)/sizeof(int);i++)
    {
        ((int*)pstLdImgArea)[i] = SWAP_32(((int*)pstLdImgArea)[i]);
    }
    //Fill SPTD Buffer
    //    => Byte[0~19]: Load Image Arguments
    //    => Byte[20~x]: Image Pixel Data
    memcpy(gSPTDataBuf, pstLdImgArea, sizeof(LOAD_IMG_AREA));
    memcpy(&gSPTDataBuf[sizeof(LOAD_IMG_AREA)], pSrcBuf, usLength);
```

```
    //Copy Scsi Buffer to SPTD Buf
    pstSPTDBuf->Cdb[0]  = 0xFE;
    pstSPTDBuf->Cdb[1]  = 0x00;
    pstSPTDBuf->Cdb[2]  = 0x00;
    pstSPTDBuf->Cdb[3]  = 0x00;
    pstSPTDBuf->Cdb[4]  = 0x00;
    pstSPTDBuf->Cdb[5]  = 0x00;
    pstSPTDBuf->Cdb[6]  = IT8951_USB_OP_LD_IMG_AREA;//0xA2
    pstSPTDBuf->Cdb[7]  = 0x00;
    pstSPTDBuf->Cdb[8]  = 0x00;
    pstSPTDBuf->Cdb[9]  = 0x00;
    pstSPTDBuf->Cdb[10] = 0x00;
    pstSPTDBuf->Cdb[11] = 0x00;
    pstSPTDBuf->Cdb[12] = 0x00;
    pstSPTDBuf->Cdb[13] = 0x00;
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    //Send SCSI Command
    bSuccess = DeviceIoControl(hDev,
                               IOCTL_SCSI_PASS_THROUGH_DIRECT,
                              &stSPTDBuf,
                              sizeof(SCSI_PASS_THROUGH_DIRECT),
                              &stSPTDBuf,
                              sizeof(SCSI_PASS_THROUGH_DIRECT),
                              &dwReturnBytes,
                               NULL );

    if(!bSuccess)
    {
        ulRetCode = GetLastError();
    }
    return bSuccess;


}
```

- **Example of IT8951 Load Image**

  New Feature for loading image by Index

  Check the argument : `ulMemAddr`

  - **Address Mode(not changed):**

    `IT8951LoadImage(pSrcImgBuf, ulITEImageBufAddr, x,y,w,h);`

  - **Index Mode(change argument 2 for enable index mode)**

    `IT8951LoadImage(pSrcImgBuf, 0x80000000 | Index, x,y,w,h);`

```
DWORD IT8951LoadImage(BYTE* pSrcImgBuf, DWORD ulITEImageBufAddr, DWORD
ulX, DWORD ulY, DWORD ulW, DWORD ulH)
{
    unsigned int j;
    BYTE* pCurStartBuf;
    DWORD ulCurDevImgStartBuf;
    DWORD ulSendLineCnt = 0;

    if(ulW <= 2048)
    {
            //Using IT8951 New USB Command for Loading Image - it Needs
IT8951 F/W support
            LOAD_IMG_AREA stLdImgInfo;

            ulSendLineCnt = SPT_BUF_SIZE/ulW;

            for(j=0;j<ulH;j=j+ulSendLineCnt)
            {
                if( ulSendLineCnt > (ulH-j))
                {
                    ulSendLineCnt = ulH-j;
                }

                stLdImgInfo.iX = ulX;
                stLdImgInfo.iY = ulY + j;
                stLdImgInfo.iW = ulW;
```

```
                stLdImgInfo.iH = ulSendLineCnt;

                //Set Image Buffer Start Address
                stLdImgInfo.iAddress = ulITEImageBufAddr;
                pCurStartBuf        = pSrcImgBuf + (j*ulW);

                //We Send Multi Line for each Bulk Transfer
                IT8951LdImgAreaAPI(&stLdImgInfo, pCurStartBuf);

            }
    }
    else
    {
        //SW Memory Copy, Send Multi Lines for each transfer
        ulSendLineCnt = SPT_BUF_SIZE/ulW;
        for(j=0;j<ulH;j=j+ulSendLineCnt)
        {
            //Set Line Start Address
            ulCurDevImgStartBuf = ulITEImageBufAddr +
((ulY+j)*gulPanelW) + ulX;
            pCurStartBuf       = pSrcImgBuf + (j*ulW);

            if( ulSendLineCnt > (ulH-j))
            {
                ulSendLineCnt = ulH-j;
            }
            //We Send 1 Line for each Bulk Transfer
            IT8951WriteMemAPI(ulCurDevImgStartBuf,
(WORD)(ulW*ulSendLineCnt), pCurStartBuf);

        }
    }
    return 1;
}
```

- **Example 3 - IT8951 Display Area**
  - Command: 0x94
  - Parameters

```
//-----------------------------------------------------------
//    Display Area
//-----------------------------------------------------------

typedef struct  _TDRAW_UPD_ARG_DATA
{
    int     iMemAddr;
    int     iWavMode;
    int     iPosX;
    int     iPosY;
    int     iWidth;
    int     iHeight;
    int     iEnWaitDpyReady; //1-Enable Wait Ready
} TDRAW_UPD_ARG_DATA;
typedef TDRAW_UPD_ARG_DATA  TDrawUPDArgData;
typedef TDrawUPDArgData*    LPDrawUPDArgData;
```

```
DWORD IT8951DisplayAreaAPI(DWORD ulX, DWORD ulY, DWORD ulW, DWORD ulH,
DWORD ulMode, DWORD ulMemAddr, DWORD ulEnWaitReady)
{
    DWORD dwReturnBytes = 0;
    BYTE  bSuccess;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;
    TDRAW_UPD_ARG_DATA stDrawUPDArgData;

    //Set Display Area 0x94 Arguments
    stDrawUPDArgData.iPosX         = SWAP_32(ulX); //Conver Little to
Big for IT8951/61
```

```
    stDrawUPDArgData.iPosY         = SWAP_32(ulY);
    stDrawUPDArgData.iWidth        = SWAP_32(ulW);
    stDrawUPDArgData.iHeight       = SWAP_32(ulH);
    stDrawUPDArgData.iEnWaitDpyReady = SWAP_32(ulEnWaitReady);
    stDrawUPDArgData.iMemAddr      = SWAP_32(ulMemAddr);
    stDrawUPDArgData.iWavMode      = SWAP_32(ulMode);

     pstSPTDBuf->Length           = sizeof(SCSI_PASS_THROUGH_DIRECT);
     pstSPTDBuf->ScsiStatus        = 0;
     pstSPTDBuf->PathId           = 0;
     pstSPTDBuf->TargetId          = 0;
     pstSPTDBuf->Lun              = 0;
     pstSPTDBuf->CdbLength         = 16;
     pstSPTDBuf->SenseInfoLength    = 0;
     pstSPTDBuf->SenseInfoOffset    = 0;
     pstSPTDBuf->DataIn           = 0;
     pstSPTDBuf->DataTransferLength =
sizeof(TDRAW_UPD_ARG_DATA);//DWORD R/W Access
     pstSPTDBuf->TimeOutValue      = 5;
     pstSPTDBuf->DataBuffer        = (void*)&stDrawUPDArgData;
     pstSPTDBuf->SenseInfoOffset    = 0;

     //Copy Scsi Buffer to SPTD Buf
     pstSPTDBuf->Cdb[0]  = 0xFE;
     pstSPTDBuf->Cdb[1]  = 0x00;
     pstSPTDBuf->Cdb[2]  = 0x00;
     pstSPTDBuf->Cdb[3]  = 0x00;
     pstSPTDBuf->Cdb[4]  = 0x00;
     pstSPTDBuf->Cdb[5]  = 0x00;
     pstSPTDBuf->Cdb[6]  = IT8951_USB_OP_DPY_AREA;
     pstSPTDBuf->Cdb[7]  = 0x00;
     pstSPTDBuf->Cdb[8]  = 0x00;
     pstSPTDBuf->Cdb[9]  = 0x00;
     pstSPTDBuf->Cdb[10] = 0x00;
     pstSPTDBuf->Cdb[11] = 0x00;
     pstSPTDBuf->Cdb[12] = 0x00;
     pstSPTDBuf->Cdb[13] = 0x00;
```

```
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    //Send SCSI Command
    bSuccess = DeviceIoControl(hDev,
                                IOCTL_SCSI_PASS_THROUGH_DIRECT,
                            &stSPTDBuf,
                                sizeof(SCSI_PASS_THROUGH_DIRECT),
                                &stSPTDBuf,
                        sizeof(SCSI_PASS_THROUGH_DIRECT),
                                &dwReturnBytes,
                          NULL );

    return bSuccess;

}
```

- **New Feature for Displaying by Index**

  Check the argument : `ulMemAddr`

  - **Address Mode(not changed):**

    `IT8951DisplayAreaAPI` `(pSrcImgBuf, ulMemAddr, x,y,w,h);`

  - **Index Mode(change argument 2 for enable index mode)**

    `IT8951DisplayAreaAPI` `(pSrcImgBuf, 0x80000000 | Index, x,y,w,h);`

- **Example 4 - IT8951 Auto Reset**
  - Command: 0xA7
  - Parameters: Null

```
DWORD IT8951SoftwareResetAPI(void)
{
    DWORD dwReturnBytes = 0;
    SCSI_PASS_THROUGH_DIRECT stSPTDBuf;
    SCSI_PASS_THROUGH_DIRECT* pstSPTDBuf = &stSPTDBuf;
    BYTE bSuccess;

    pstSPTDBuf->Length             = sizeof(SCSI_PASS_THROUGH_DIRECT);
    pstSPTDBuf->ScsiStatus          = 0;
    pstSPTDBuf->PathId             = 0;
```

```
    pstSPTDBuf->TargetId           = 0;
    pstSPTDBuf->Lun                   = 0;
    pstSPTDBuf->CdbLength          = 16;
    pstSPTDBuf->SenseInfoLength     = 0;
    pstSPTDBuf->SenseInfoOffset     = 0;
    pstSPTDBuf->DataIn             = SCSI_IOCTL_DATA_OUT;
    pstSPTDBuf->DataTransferLength  = 0;
    pstSPTDBuf->TimeOutValue        = 5;
    pstSPTDBuf->DataBuffer          = (void*)gSPTDataBuf;
    pstSPTDBuf->SenseInfoLength     = 0;

    //Copy Scsi Buffer to SPT Buf
    pstSPTDBuf->Cdb[0] = 0xFE;
    pstSPTDBuf->Cdb[1] = 0x00;
    pstSPTDBuf->Cdb[2] = 0x00;
    pstSPTDBuf->Cdb[3] = 0x00;
    pstSPTDBuf->Cdb[4] = 0x00;
    pstSPTDBuf->Cdb[5] = 0x00;
    pstSPTDBuf->Cdb[6] = 0xA7;
    pstSPTDBuf->Cdb[7] = 0x00;
    pstSPTDBuf->Cdb[8] = 0x00;
    pstSPTDBuf->Cdb[9] = 0x00;
    pstSPTDBuf->Cdb[10] = 0x00;
    pstSPTDBuf->Cdb[11] = 0x00;
    pstSPTDBuf->Cdb[12] = 0x00;
    pstSPTDBuf->Cdb[13] = 0x00;
    pstSPTDBuf->Cdb[14] = 0x00;
    pstSPTDBuf->Cdb[15] = 0x00;

    bSuccess = DeviceIoControl(hDev,

IOCTL_SCSI_PASS_THROUGH_DIRECT,//IOCTL_SCSI_PASS_THROUGH_DIRECT,//IOC
TL_SCSI_PASS_THROUGH,
        &stSPTDBuf,
        sizeof(SCSI_PASS_THROUGH_DIRECT),//+sizeof(gSPTDataBuf),
//sizeof( TSPTWBData),
        &stSPTDBuf,
```

```
        sizeof(SCSI_PASS_THROUGH_DIRECT),//sizeof(gSPTDataBuf),
//sizeof( TSPTWBData),
        &dwReturnBytes,
        NULL);
    return 1;

}
```