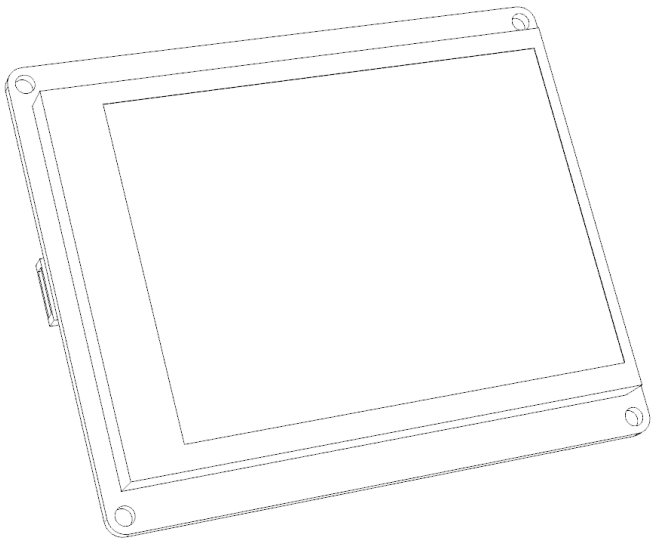
RoboPeak Mini USB Display
USB Interface Protocol Specification

2013-12-11
Rev. 3



Contents:

1. Overview

RoboPeak Mini USB display is a standard USB device follows the USB2.0 Full Speed (FS) specification. Any host system with a USB host compatible with the USB2.0 FS standard can communicate with the display.

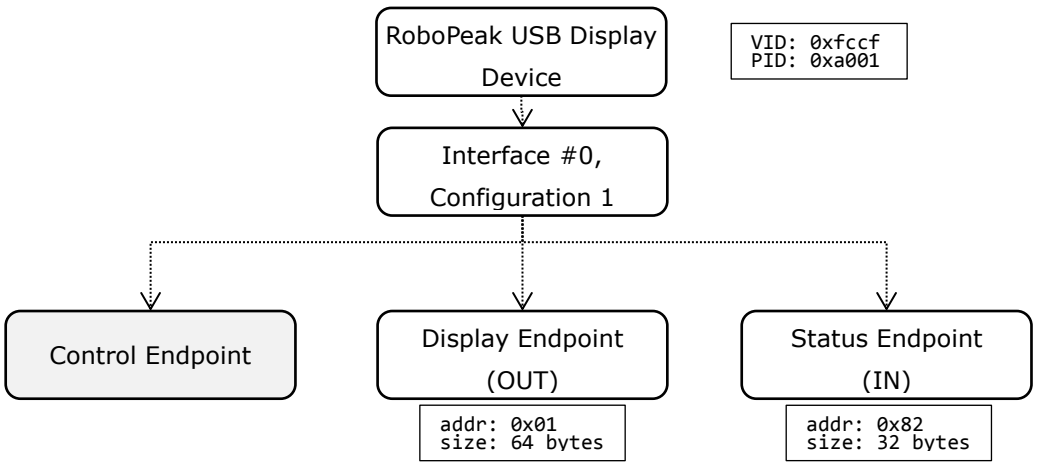
In this document, the detailed USB protocol of the display is provided.

2. Basic Protocol

USB Device Descriptor

The USB display provides one USB interface with two uni-directional endpoints to handle display commands and status query from the host side.

The following figure depicts the interface structure:



The USB Display device uses the following VID/UID pair to be identified by the host system:

```
VID: 0xfccf PID: 0xa001
```

Other pre-defined values in the USB descriptor are listed below:

Field Type	Predefined Value
bDeviceClass	0
bDeviceSubClass	0
bInterfaceClass	0xFF
bInterfaceSubClass	0xC1
Manufacturer String	"RoboPeak"
Product String	"RPusbdisp"
Interface Descriptor String	"RPUSBDisplay Channel"

Display Endpoint

All display commands and the related data are transferred via the Display Endpoint. This endpoint is implemented as bulk-type, output only with the address 0x01. The maximum transfer data size for each USB package cannot exceed 64 bytes.

Item	Value
Endpoint Direction	Out
Address	0x01
MaxPacketSize	64
Type	Bulk

RoboPeak USB Display uses a packet-based, stateless protocol via this endpoint. Please refer to the following section in the document for details.

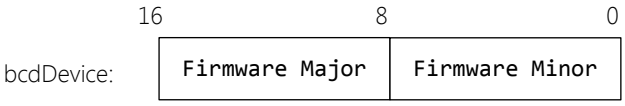
Status Endpoint

The Status Endpoint is a interrupt-type, Input only channel with address 0x82. Host systems query the build-in touch screen events and the display status via this endpoint. The maximum transfer size is 32 bytes.

Item	Value
Endpoint Direction	In
Address	0x82
MaxPacketSize	32
Type	Interrupt

Display Firmware Version

The display firmware version info can be retrieved from the `bcdDevice` field of the USB descriptor directly.



The `Firmware major` represents the integer part of the version number, whereas the `Firmware Minor` is the decimal part.

e.g. if the bcdDevice value is 0x0102, then the actual firmware version number is 1.02.

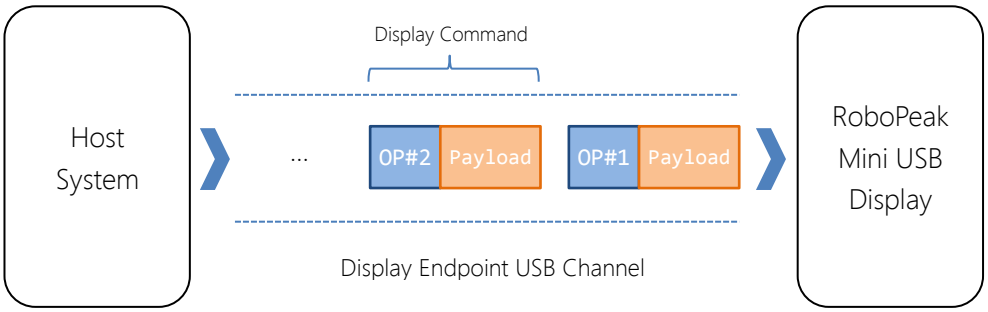
Data Endianness

All the data transferred via the USB interface is based on little-endian. For the host architectures like MIPS which use big-endian, the endianness transformation is required otherwise the USB display will receive corrupted data.

3. Display Commands

Host system controls the RoboPeak Mini USB Display to show specified image on the screen via a set of Display Commands provided by the USB display.

Each Display Command is a predefined data packet transmitted through the Display Endpoint which contains an `operation` field that describes what operation to be done on the screen and related `payload` data to describe how the operation to be performed.



RoboPeak Mini USB Display provides the following Display Commands for host systems to use:

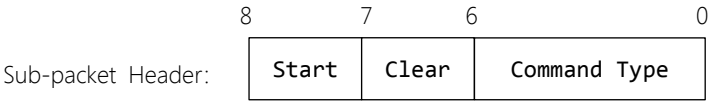
Command	Description
Fill	Fill the screen with the color specified in the payload section
Rect	Draw a solid rectangle on the screen with the given position and color specified in the payload section
Bitblt	Display a given image on the screen with the specified location.
Copyarea	Copy an area of image on the screen to another place.

Basic Packet Format

As the USB channel is a diagram based (packet based) data bus with the maximum packet size limit. In order to transmit arbitrary size of command packets, all the display command packets will be split into sub-packets with a pre-defined common sub-packet header.

It is the host system driver's responsibility to ensure each sub-packet won't exceeds the 64bytes as defined by the display endpoint.

The pre-defined common sub-packet header is a byte (8bit) flag with the following definition:



- `Command Type`:

The 6-bit `Command Type` defines which display command this sub-packet belongs to. The following value can be used:

Value	Description
0x1	This sub-packet belongs to a Fill command
0x2	This sub-packet belongs to a Bitblt command
0x3	This sub-packet belongs to a Rect command
0x4	This sub-packet belongs to a Copyarea command

- `Start bit`:

The `Start` bit is used to indicate whether the current sub-packet is the first sub-packet of the display command sub-packets sequence.

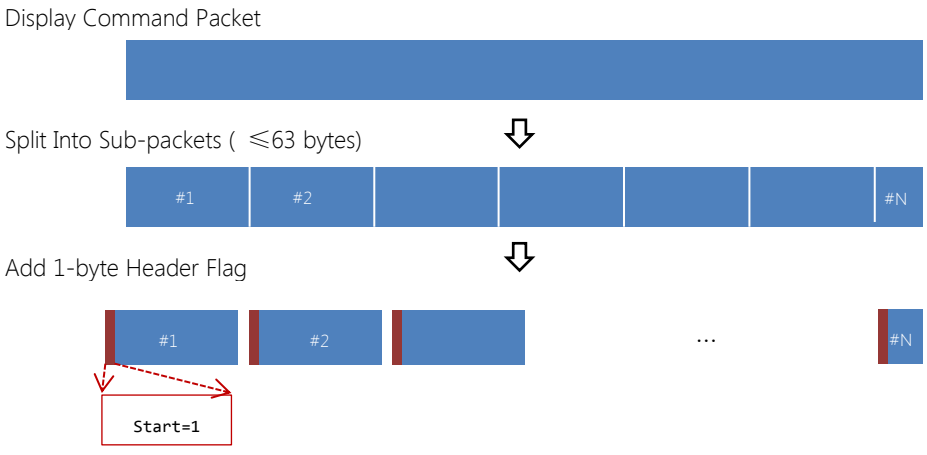
When this bit is set to 1, the USB display will treat the current received sub-packet is a new display command.

- `Clear bit`:

When set this bit, the USB display will clear this internal dirty flag. Please refer to the 4.Touch Screen and Display Status section for details.

When sending a Display Command packet, the host driver should generate a sequence of sub-packets within which the sub-packet header should always be added in the header of the sub-packets. Then, the remaining 63 bytes data of the sub-packet can be filled with the data of the original Display Command packet.

For the first sub-packet of the sequence, the `Start bit` must be set by the host driver. All of the sub-packets belong to a Display Command should have the same `Command Type`.

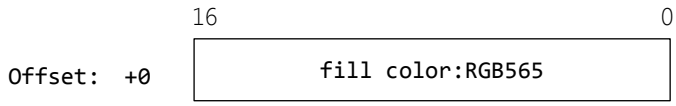


Fill Command

The Fill Command requests the USB Display to fill the full screen with the specified color. The color is represented in RGB565 16bit format.

The Fill Command Packet has the following format:

Fill Command Packet:



- `fill color`

Color in RGB565 (16bit) format to be used to fill the screen.

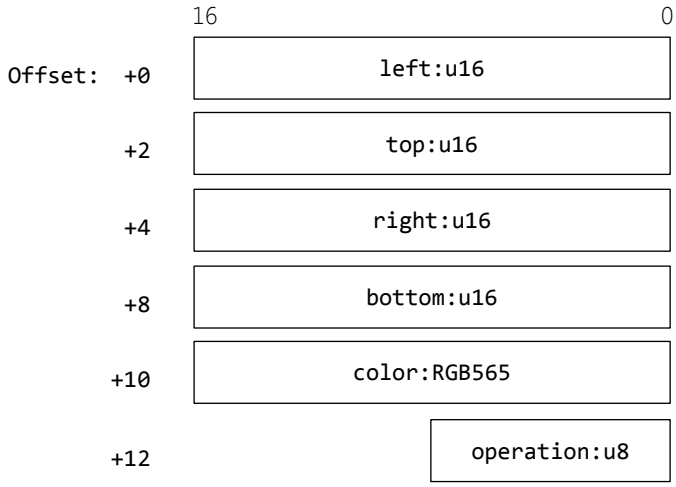
The Fill command can be used to clear the screen.

Rect Command

The Rect Command makes the USB Display to draw a solid rectangle on the screen with the given position (left,top,right,bottom) and the given fill color.

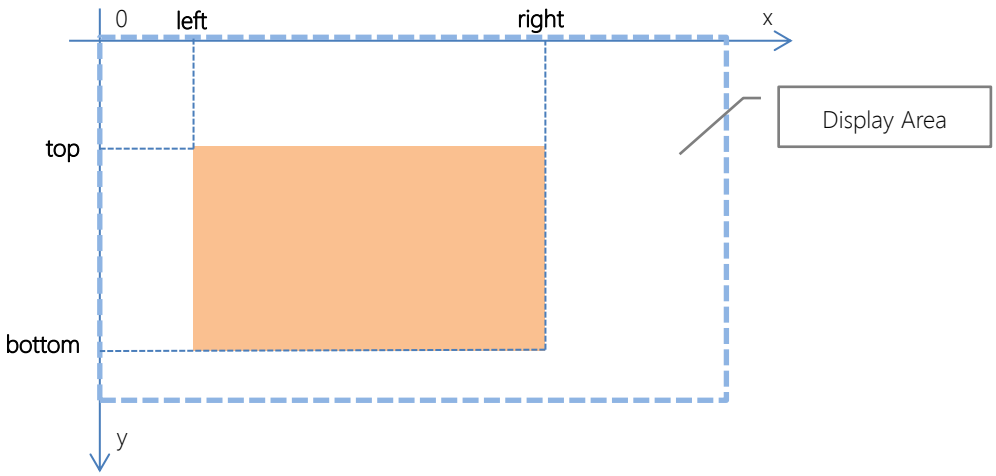
The Rect Command Packet has the following format:

Fill Command Packet:



- `left, top, right, bottom`

These fields specify the position and the size of the rectangle to draw. All of them are 16bit unsigned integers. The following coordination system is used by the USB display to interpret these values.



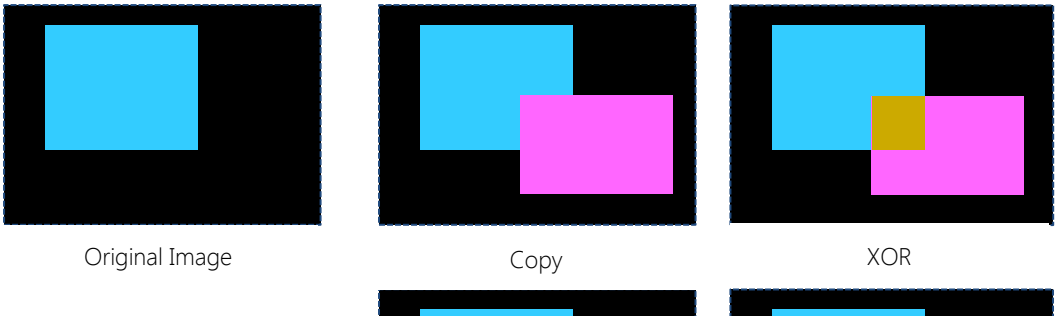
- `color`

The RGB565 color used to fill the rectangle.

- `operation`

Specify the pixel operation to be performed with the previous pixel color of the same location. The following values can be used:

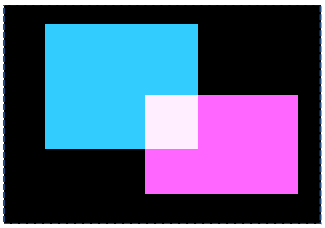
Value	Operation	Description
0x0	Copy	The new pixel value is set to the fill color
0x1	XOR	The new pixel value is the result of the XOR operation of the previous value and the fill color.
0x2	OR	The new pixel value is the result of the OR operation of the previous value and the fill color.
0x3	AND	The new pixel value is the result of the AND operation of the previous value and the fill color.



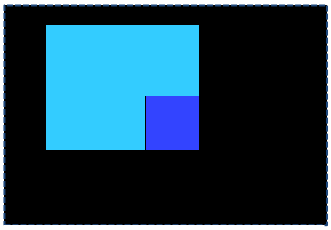
Original Image

Copy

XOR



OR



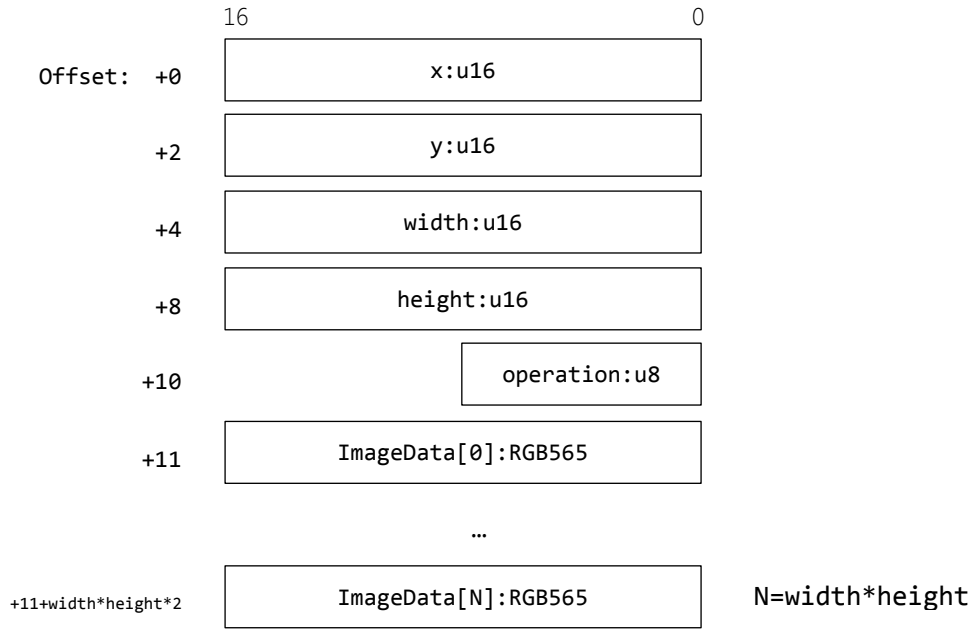
AND

Bitblt Command

The Bitblt Command can be used to transmit an image and make it to be displayed on the USB display with the specified location. The data of image to be displayed should be appended to the command packet as the payload. The image pixels are 16bit RGB565 format.

The Bitblt Command packet has the following format:

Bitblt Command Packet:

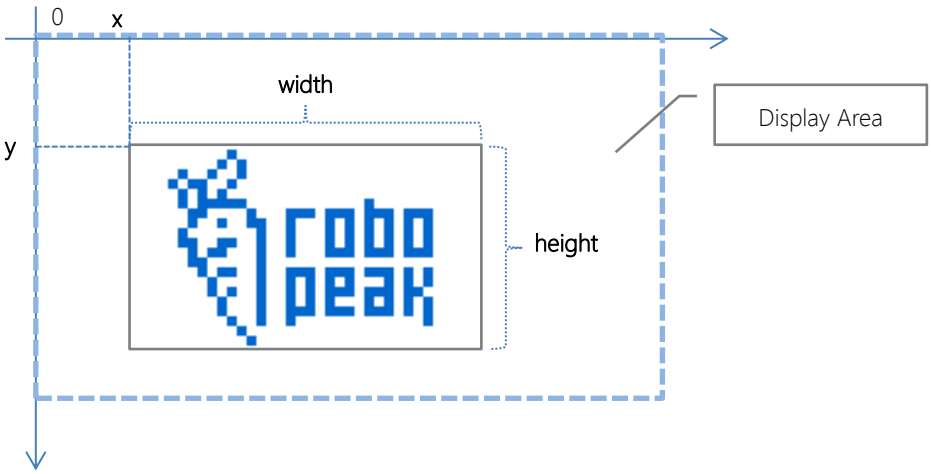


- `x,y`

The position where the image to be displayed on the screen. The (x,y) is the upper-left corner of the image.

- `width, height`

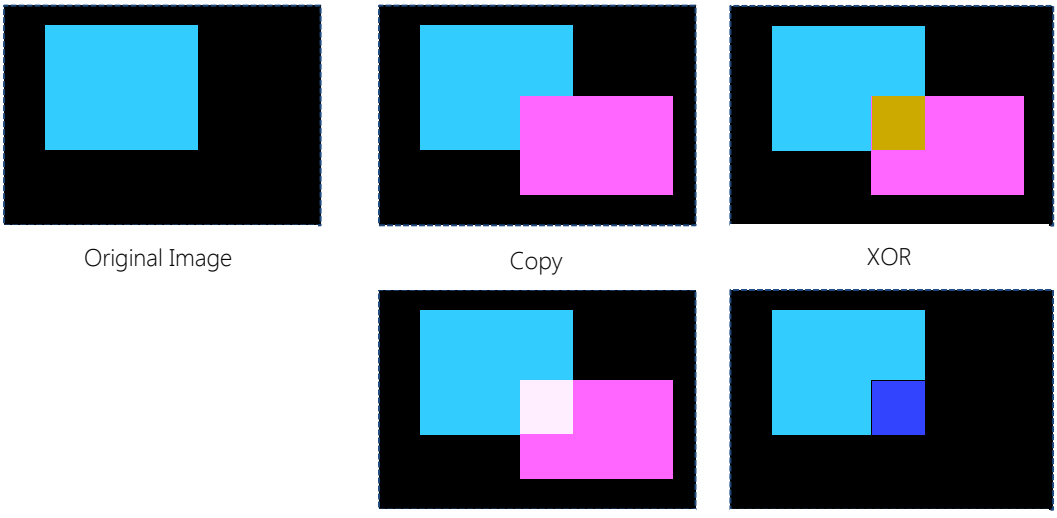
The size of the image to be displayed, in pixel unit.



- `operation`

Specify the pixel operation to be performed with the previous pixel color of the same location. The following values can be used:

Value	Operation	Description
0x0	Copy	The new pixel value is set to the fill color
0x1	XOR	The new pixel value is the result of the XOR operation of the previous value and the fill color.
0x2	OR	The new pixel value is the result of the OR operation of the previous value and the fill color.
0x3	AND	The new pixel value is the result of the AND operation of the previous value and the fill color.



- `ImageData[0..n]`

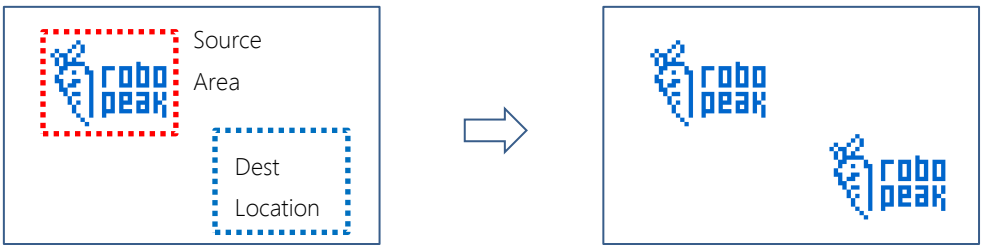
Pixel data of the image to be displayed represents in RGB565 format. The data is organized in a left-to-right, top-to-bottom fashion as the following figure depicts:

Pixel Data Organization of a N*M Image:

P[0]	P[1]	P[2]	...	P[N-1]
P[N]	P[N+1]	P[N+2]	...	P[2*N-1]
...				
...				P[M*N-1]

Copyarea Command

The Copyarea Command requests the USB Display to copy an area of specified image block on the screen to another location. This command can be used to accelerate the animations like scrolling or moving effect.

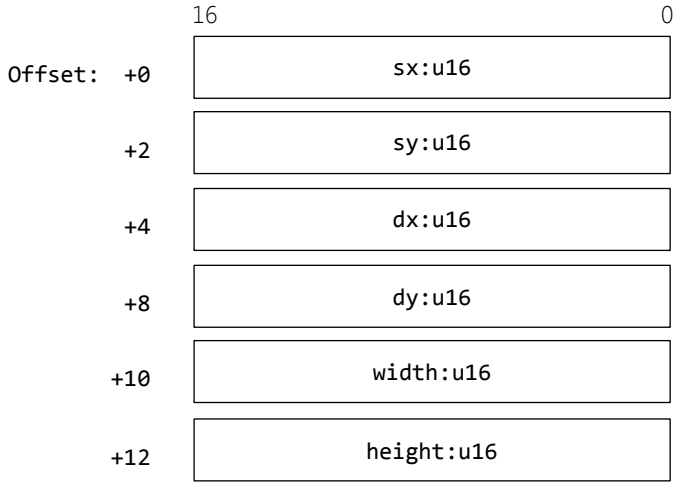


Before Copyarea

After Copyarea

The Copyarea Command Packet has the following format:

Copyarea Command Packet:



- `sx,sy,width,height`

The rectangle area of the source image to copy.

- `dx, dy`

The upper-left corner of the destination where the source image to be copied to.

4. Touch Screen and Display Status

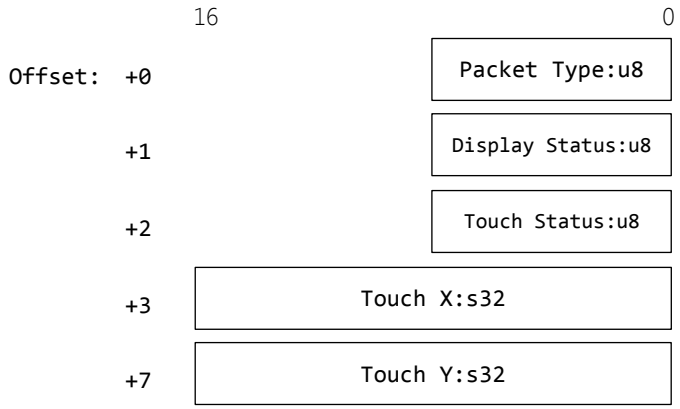
Host systems can retrieve the build-in touch screen events and the display status via polling data from the Status Endpoint. Once the following events occur, the USB Display will send a status packet via the Status Endpoint.

Event Type	Description
Finger Pressed	The user finger has pressed the screen
Finger Moving	The user finger is moving on the screen
Finger Leaved	The user finger has left the screen
Display dirty flag set	The display has detected it is not of sync of the host system

Status Packet

The USB Display always sends the status packet with the same format no matter what event has happened. The Status Packet format is defined as below:

Status Packet:



- `Packet Type`

The packet type of the Status Packet. It must be 0 for the current USB Display firmware.

- `Display Status`

The status of the display. It can be set with the following bit flag with OR operation:

Bit Flag	Value	Description
Dirty Flag	0x80	Indicates the screen image may be out of sync with the host system

Once the Dirty Flag has been set, the host system should take actions to sync the image on the screen with its internal version, e.g. flush the whole image from the frame-buffer to the screen. Also, the host system should set the clear

bit in the Display Command Sub-packet header to prevent the display to report the event again.

- `Touch Status`

The event type of the build-in touch screen when a touch event has occurred. The value can be the following ones:

Touch Status	Value	Description
Pressed	1	The user finger currently presses on the screen.
Not touched	0	The user finger currently doesn't touch the screen

The host driver should keep a copy of the previous touch status to detect the exact touch event. The following table can be used as a reference:

Current Status	Previous Status	Exact Touch Event
Not touched	Touched	Pressed Event
Touched	Touched	Moving Event
Touched	Not Touched	Leave Event

- `Touch X, Touch Y`

The finger position data. It is reported corresponding to the current screen pixel position.

5. Revision History

Date	Description
2013-10-29	Initial version
2013-11-01	Fixed several typos
2013-12-04	Refined the descriptions
2013-12-11	Fixed bugs in the Status Packet structure define