Chapter 10

2D Graphics API

Note

This component library is available as a part of the TriMedia DTV software system. It is not included with the basic TriMedia SDE, but it is available under a separate licensing agreement. Please contact your TriMedia sales representative for more information. ◆

2D Graphics Library Overview

The 2D Graphics Library draws 2D lines, points, text, rectangles and polygons on a buffer that the user passes in. It is compliant with TriMedia Software Architecture (TSA). The 2D Graphics Library is decoupled from the hardware, hence, it does not have an instance setup function. It renders on the packet buffer passed in from the user. It supports eight buffer types, and they are: YUV422 planar, video-overlay sequence, DTVCM-YUV422 planar, DTVCM-overlay sequence, YUV422 planar with 4-bit alpha, RGB888, RGB565, and RGB555A.

The font renderer renders two font types: TMFont and TMFont2. It provides color conversion between RGB and YUV color spaces. The supported drawing primitives are: Point, Line, Text, Fill Rectangle, Fill Polygon, Image, and Blt.

Rectangle Coordinates Specification

The packet buffer size is derived from the imageWidth and imageHeight fields of the tmVideoFormat_t of the packet buffer. The 2D Graphics Library draws only within the packet boundary. The origin (0,0) of the rectangle is at the top left corner of the buffer. Therefore the upper left coordinate of the tsa2DRect_t structure is defined to be less than or equal, in both X and Y, to the bottom right coordinate. All API functions that draw rectangles generate an upper left and bottom right point from user-specified arguments. Since this kind of min/max box can be derived from any two points, no ordering is assumed for points supplied as arguments to the 2D Graphics Library.

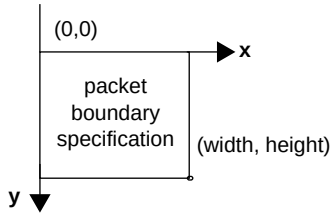

Figure 10-1 Rectangle Coordinates

Supported Buffer Types

2D Library supports the following buffer types:

- vdfYUV422Planar
- vdfYUV422Sequence
- vdfDTVCMPlanar
- vdfDTVCMSequence
- vdfRGB24
- vdfRGB16
- vdfRGB555A
- vdfYUV422PlanarAlpha4

Buffer type are specified through the *dataSubtype* entry of the video format. For example, to specify `vdfDTVCMPlanar`, use the following code entry shown below:

```
yuvFmt.dataSubtype = vdfYUV422Planar
```

Graphics and Video Images Blending Specification in the DTV Buffer Types

In the DTV environment, the `vdfDTVCMPlanar` and `vdfDTVCMSequence` buffer types are the corresponding YUV422 and overlay buffer types with the consideration of color multiplexing between Graphics and Video.

Blending Graphics and Video Streams

The following flags are used to specify the blending between Graphics and Video:

- `vdfDTVCM_0Video`
- `vdfDTVCM_25Video`
- `vdfDTVCM_50Video`
- `vdfDTVCM_75Video`
- `vdfDTVCM_DontCare`

The blending factor are specified through the description entry of the video format. For example, to specify a blending factor ratio of 25% of video and 75% of graphics, enter the following:

```
yuvFmt.description = vdfDTVCM_25Video; /* 25% Video, 75% graphics */
```

Note

When the graphics buffers are filled with color key values, it displays 100% of Video and 0% of Graphics. ◆

Blending of Anti-Aliased Text and Video Streams

The following two additional flags are used to specify the blending between anti-aliased Text and Video streams:

- `vdfDTVCM_MAP_GAtoVA_W_FC`
- `vdfDTVCM_MAP_GAtoVA_W_FCBC`

`vdfDTVCM_MAP_GAtoVA_W_FC` maps the encoded alpha blending values (0-15) in the text to the color multiplexor blending values (that is, LSBs of UV: 00, 01, 10, 11) with the foreground color.

`vdfDTVCM_MAP_GAtoVA_W_FCBC` maps the encoded alpha blending values (0-15) in the text to the color multiplexor blending values (i.e. LSBs of UV: 00, 01, 10, 11) with the resulting color of alpha blended foreground and background colors.

For example:

```
((ptmVideoFormat_t)pYuvPkt->header->format)->description =
                                              vdfDTVCM_MAP_GAtoVA_W_FC;
```

Table 10-1 Blending Values

Flag	Graphics Alpha Blending Values	Video Alpha Blending Values
`vdfDTVCM_MAP_GAtoVA_W_FC`	0 to 15	00, 01, 10, 11
`vdfDTVCM_MAP_GAtoVA_W_FCBC`	0 to 15	00, 01, 10, 11

Drawing Primitives APIs

There are three sets of drawing primitive APIs:

- No Graphics Context APIs
- Poly APIs
- Graphics Context APIs

No Graphics Context APIs

The following drawing primitives API do not use graphics context:tsa2DPointNC, tsa2DLineNC, tsa2DFillRectNC, tsa2DImageNC, and tsa2DTextNC.

Instead, the required information is supplied through input arguments.

Poly APIs

The following poly APIs are: `tsa2DPolyPoints`, `tsa2DPolyLine`, `tsa2DPolyFillRect`, `tsa2DPolyImage`, `tsa2DPolyText`, and `tsa2DPolyBlt`.
These Poly functions do drawing on multiple packets (i.e. `numPkt`).
Within each packet or each set of packets, they can also draw multiple times (i.e. specify in `pNumPerPkt`).

`pPkList` is a pointer to an array of packet pointers. The number of packet pointers should equal to `numPkt`. `pPtList` is a pointer to an array of 2D coordinates. The number of coordinates should equal to:

```
(pNumPerPkt[0] + ... + pNumPerPkt[numPkt-1])
```

`pColor` is a pointer to `tsa2DColor_t`. The entry, *`pColor->pColorData`*, is a pointer to an array of 2D colors (ex: `tsaYUVColor_t`). The number of colors should be equal to:

2D API Data Structure Descriptions`(pNumPerPkt[0] + ... + pNumPerPkt[numPkt-1])`

Graphics Context APIs

The following drawing primitive APIs do use graphics context of the input parameter: tsa2DGetPixel, tsa2DSetPixel, tsa2DPoint, tsa2DLine, tsa2DText, tsa2DImage, gsa2DFillRect, tsa2DFillPoly, tsa2DBlt, and tsa2DBltRegion.

Clipping

The 2D Graphics Library supports clipping on all primitives. Only the portion of a primitive falling within the packet boundary, if any, is drawn. The clipping is pixel exact, meaning that the pixels generated for a clipped primitive are a subset of the pixels generated for the unclipped primitive.

Drawing Rules

The 2D Graphics Library uses the ‘upper left pixel in, bottom right pixel out’ rule when determining which pixels belong to filled rectangle, image, and BitBlt drawing primitives. This means that the bottom row and rightmost column of the primitives mentioned are not drawn. This rule ensures that in the case of adjacent primitives, pixels along shared borders belong to exactly one primitive.

Fonts: TMFont and TMFont2

2D Graphics Library supports two types of fonts, TMFont and TMFont2. They are both bitmap type of fonts with slight variation in the font information data structures.

TMFont

The information of a particular font is stored in (font.mtr and font.bit) files. When `tsa2DLoadFont` is called, it loads the information into library. You need to provide information regarding the path of font files and the library returns a fonID after it loads in the font. `tsa2DUnLoadFont` unloads the font specified in the fontID.

Font TM Font Files

Below is a picture description of the TMFont font files. The .mtr file contains information for the font and each character. The .bit file has character bitmaps information. Figure 10-3 provides a graphic example.

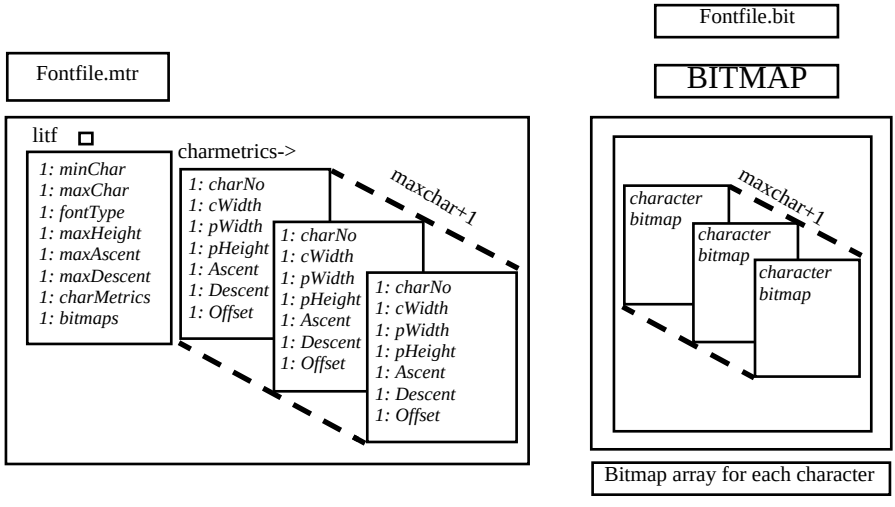

Figure 10-2 TMFont Font Files

TMFont2

The information of a particular font is stored in (font.tm and font.bit) files. When `tsa2DLoadFont` is called, it loads the information into the 2D library. You need to provide information regarding the path of font files and the library returns a fontID after it loads in the font. `tsa2DUnLoadFont` unloads the font specified in the fontID.

TMFont2 Character Metrics

Each pixel is represented with 4 bits of blending information (i.e. the color blending between text color and background color). 0xf shows the pixel with the text color. 0x0 shows the background color. The values in between are blended proportionally.

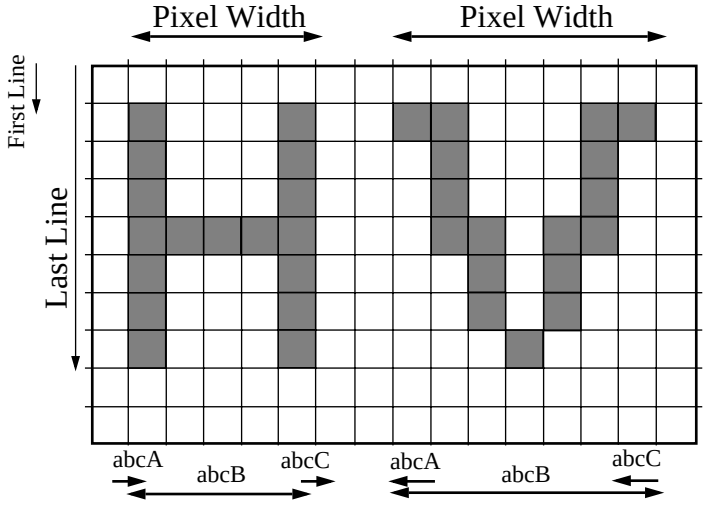

Figure 10-3 TMFont Character Metric Graphic Representation

TMFont2 Font Files

Below is a picture description of entries in the `tsaTMFont2CharMetrics`:

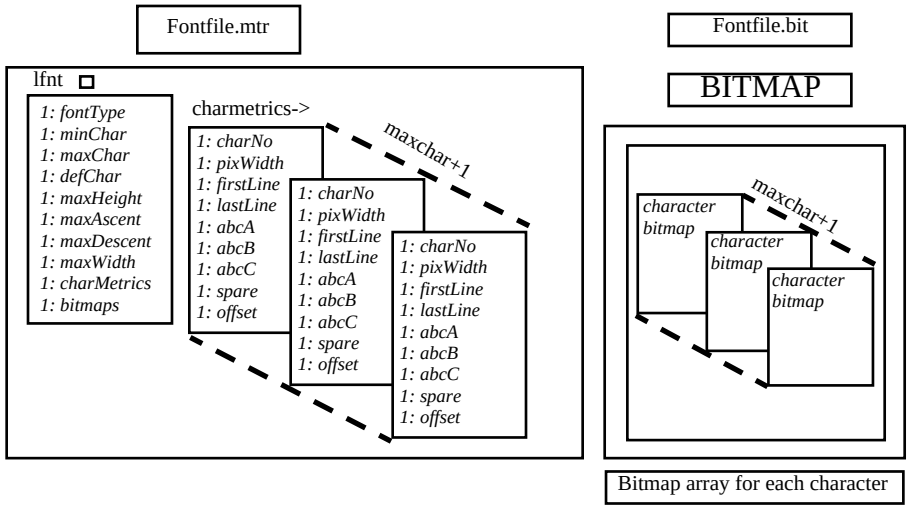

Figure 10-4 TMFont2 Font Files

How to Use the 2D Graphics Library

To use the 2D Graphics Library, you must use the specified hardware, and programs discussed in this section.

Necessary Items

The following items are necessary in order to use 2D Graphics:

1. TriMedia board with TV.
2. TriMedia Compilation System (TCS).
3. TriMedia Application Software (TAS) and specifically libtm2D.a.
4. Optional: Example program using 2D Library.

Programs that use 2D Graphics Library

An application program needs to get an instance ID from `tsa2DOpen` first, before using: font APIs, drawing APIs, and color conversion APIs. Use `tsa2DClose` when done.

1. Call `tsa2Dopen` to get an instance ID.
2. Use font APIs: `tsa2DGetStrWidth`, `tsa2DGetFontInfo`, `tsa2DLoadFont`, and `tsa2DUnLoadFont`.
3. Use color conversion APIs: `tsa2DRGBtoYUV`, `tsa2DYUVtoRGB`, `tsa2DLoadIndexColorLUT`, `tsa2DUnLoadIndexColorLUT`, and `tsa2DGetColorFmIndex`.
4. Use drawing APIs: `tsa2DLine` (NC), `tsa2Dpoint` (NC), `tsa2DFillRect` (NC), `tsa2Dimage` (NC) to draw to the YUV422 buffer or overlay buffer or DTVCM buffer.
5. Call `tsa2Dclose` to finish.

How to Load Fonts

There are two configurations that the user can load font in: PC host and standalone.

PC Host

In PC host configuration, the user calls tsa2DLoadFont to load font files. In TMFont type, they are file.bit and file.tm.

Standalone

In standalone configuration, the user only needs to do the following:

```
#include “plain16.h”
pFont->fontID = &plain16;
```

There is no need to load the font.

Technical Difficulties with 2D Graphics Library

1. In the YUV422 image, two Y pixels share one set of U and V. It is difficult to render exactly two colors for two neighboring pixels and have two sharp colors next to each other.
2. For the Overlay image (422 interleave:Y1 V0 Y0 U0), it only writes or reads overlay buffer in a 4-byte packet (equivalent to 2 pixels on screen).
3. For the DTVCM buffer, it uses the two least significant bits (LSBs) of U and V to indicate the blending level of video and graphics. This results in loss of colors.

Returned Error Messages

The following error messages are returned for the corresponding API.

Error code	API
TWOD_ERR_COLOR_TYPE	All the drawing APIs
TWOD_ERR_INDCOLOR_ALLOC	tsa2DLoadIndexColorLUT
TWOD_ERR_TMFONT_ALLOC	tsa2DLoadFont, tsa2DGetFontInfo
TWOD_ERR_TMFONT_MTR_FILE	tsa2DLoadFont, tsa2DGetFontInfo
TWOD_ERR_TMFONT_BIT_FILE	tsa2DLoadFont
TWOD_ERR_TMFONT_GETSTRWIDTH	tsa2DGetStrWidth
TWOD_ERR_TMFONT_NULL	tsa2DTextNC, tsa2DPolyText, tsa2DText
TWOD_ERR_TMFONT2_ALLOC	tsa2DLoadFont, tsa2DGetFontInfo
TWOD_ERR_TMFONT2_TM_FILE	tsa2DLoadFont, tsa2DGetFontInfo
TWOD_ERR_TMFONT2_BIT_FILE	tsa2DLoadFont
TWOD_ERR_TMFONT2_GETSTRWIDTH	tsa2DGetStrWidth
TWOD_ERR_TMFONT2_NULL	tsa2DTextNC, tsa2DPolyText, tsa2DText
TWOD_ERR_ALLOC	tsa2DOpen, tsa2DFillPoly
TWOD_ERR_ERR_NOT_SUPPORTED	tsa2DTextNC, tsa2DPolyText, tsa2DText, tsa2DBlt, tsa2DPolyBlt, tsa2DBltRegion, tsa2DImageNC, tsa2DPolyImage, tsa2DImage
TWOD_ERR_INVALID_RECT	tsa2DTextNC, tsa2DPolyText, tsa2DText, tsa2DFillRectNC, tsa2DPolyFillRect, tsa2DFillRect
TWOD_ERR_INVALID_POINTER	All the drawing APIs
TWOD_ERR_INVALID_FLAG	tsa2DGetFontInfo
TWOD_ERR_ODD_STRIDE	none
TWOD_ERR_INVALID_POLYGON	tsa2DFillPoly
TWOD_ERR_BLIT_INVALID_OPS_STRING	none

2D API Data Structure Descriptions

This section describes the 2D graphics API data structures. These data structures are defined in the tsa2D.h header file

tsa2DCapabilities_t

```
typedef struct tsa2DCapabilities_t {
   ptsaDefaultCapabilities_t   defaultCapabilities;
   tmVideoRGBYUVFormat_t       supportedBufferFormats;
} tsa2DCapabilities_t; *ptsa2DCapabilities_t;
```

Fields

defaultCapabilities	Default capabilities.

Description

tsa2DCapabilities_t holds a list of capabilities. The 2D maintains a structure of this type to describe itself. The user can retrieve the address of this structure by calling tsa2DGetCapabilities().

tsaYUVAColor_t

```
typedef struct tsaYUVAColor_t {
   UInt8   Y;
   UInt8   U;
   UInt8   V;
   UInt8   reserved;
} tsaYUVAColor_t, *ptsaYUVAColor_t;
```

Fields

Y	Y value.
U	U value.
V	V value.
reserved	Reserved.

Description

For the 2D display color value represent YUV values, each value takes up 8 bits of the integer value in the following order:

Integer Value	31-24	23-16	15-8	7-0
YUV Value	—	V	U	Y

tsaYUVColor_t

```
typedef struct tsaYUVColor_t {
   UInt8   V;
   UInt8   U;
   UInt8   Y;
} tsaYUVColor_t, *ptsaYUVColor_t;
```

Fields

V	V value.
U	U value.
Y	Y value.

Description

For the 2D display color value represent YUV values.

tsaRGBColor_t

```
typedef struct {
   UInt8   R;
   UInt8   G;
   UInt8   B;
} tsaRGBColor_t, *ptsaRGBColor_t;
```

Fields

R	Red color.
G	Green color.
B	Blue color.

Description

This structure describes RGB color.

tsa2DColorType_t

```
typedef enum {
   noColor        = 0,
   indexColor     = 1,
   YUVColor       = 2,
   YUVAColor      = 4,
   RGB888Color    = 8
   RGB565Color    = 16
   RGB555AColor   = 32
    YUVA4Color    = 64
} tsa2DColorType_t;
```

Description

This enum describes the available color type. According to *colorType* specified, *pColorData* points to particular color data. If it is *indexColor*, *pColorData* specifies the index color (i.e. an index number) of the current loaded and active index color LUT.

tsa2DColor_t

```
typedef struct {
   tmal2DColor_t   ColorType;
   Pointer         pColorData;
} tsa2DColor_t, *ptsa2DColor_t;
```

Fields

ColorType	Color specified.
pColorData	Pointer to particular color data.

Description

According to the *ColorType* specified, *pColorData* points to particular color data. If it is `indexColor`, *pColorData* specifies the index color (for example, an index number) of the current loaded and active index color LUT.

tsaYUVA4Color_t

```
typedef struct {
   UInt8    Y;
   UInt8    U;
   UInt8    V;
   UInt8    A;
} tsaYUVA4Color_t, * ptsaYUVA4Color_t;
```

Fields

Y	Y value.
U	U value.
V	V value.
A	Alpha value: only the 4 least-significant bits are used.

Description

This structure describes a YUV color with 4-bit alpha value.

tsa2DIndexColorLUT_t

```
typedef struct tsa2DIndexColorLUT_t {
   Int32           numEntry;
   tmal2DColor_t   LUTColorType;
   Pointer         pLUTColorData;
   UInt32          indexColorLUTID;
} tsa2DIndexColorLUT_t, *ptsa2DIndexColorLUT_t;
```

Fields

numEntry	Entry number.
LUTColorType	Color specified.
pLUTColorData	Color specified.
indexColorLUTID	Pointer to particular color data.

Description

This is the data structure used in loading the index color LUT. `numEntry` specifies the number of index colors in this LUT. *LUTColorType* specifies the color type in the look up table. *pLUTColorData* is a pointer, points to the corresponding colors in the look up table. Library fills in the *indexColorLUTID* after loading it successfully.

tsa2DCoordinate_t

```
typedef struct tsa2DCoordinate_t {
   Int   X;
   Int   Y;
} tsa2DCoordinate_t, *ptsa2DCoordinate_t;
```

Fields

X	X coordinate.
Y	Y coordinate.

Description

X and Y represent the cartesian coordinates in a 2D plane.

tsa2DRect_t

```
typedef struct tsa2DRect_t {
   tmal2DCoordinate_t   upLt;
   tmal2DCoordinate_t   btRt;
} tsa2DRect_t, *ptsa2DRect_t;
```

Fields

upLt	Specifies the (x,y) coordinates of the upper left position of the rectangle.
btRt	Specifies the (x,y) coordinates of the bottom right position of the rectangle.

Description

This data structure describes a rectangle through the positions of the upper left and bottom right coordinates.

tsa2DImageType_t

```
typedef enum {
   noImage           =  0,
   YUV422Image       =  1,
   YUV420Image       =  2,
   OverlayImage      =  4,
   BMP8BPPCLUTImage  =  8,
   PPMImage          = 16,
   GIFImage          = 32,
   RGB888Image       = 33,
   RGB565Image       = 34,
   RGB555AImage      = 35,
   YUV422A4Image     = 36
} tsa2DImageType_t;
```

Description

This type definition enumerates the available image types. Only `YUV422Imagetype` is currently being supported.

tsa2DImage_t

```
typedef struct tsa2DImage_t {
   tsa2DImageType_t  imageType;
   Int               iWidth;
   Int               iHeight;
   Int               iStride;
   Pointer           pHeader;
   Pointer           pData1;
   Pointer           pData2;
   Pointer           pData3;
   Pointer           pData4;
} tsa2DImage_t, *ptsa2DImage_t;
```

Fields

imageType	Specifies the image type.
iWidth	Specifies the width of the image.
iHeight	Specifies the height of the image.
iStride	Specifies the stride of the image.
pHeader	Pointer to the header information of the image.
pData1	Pointer to the first data of the image.
pData2	Pointer to the second data of the image.
pData3	Pointer to the third data of the image.
pData4	Pointer to the fourth data of the image.

Description

This data structure provides information regarding various images. First, the user specifies image type. *pHeader* points to image header information. *pData1*, *pData2*, *pData3*, and *pData4* can be used flexibly, pointing to image data.

tsa2DTextStyle_t

```
typedef enum {
   noTextStyle     = 0,
   textOnly        = 1,
   textBackColor   = 2,
   textUnderline   = 4
} tsa2DTextStyle_t;
```

Description

This type definition enumerates the supported text styles.

Text style can be either of the following:

- `textOnly` draw text with foreground color.
- `textBackColor` draw text with foreground color and fill the background with the background color.
- `textUnderline` draw the text and underline with foreground color.

tsa2DFontInfoFlag_t

```
typedef enum {
   NOFONTINFOFLAG   = 0,
   MINCHAR          = 1,
   MAXCHAR          = 2,
   MAXHEIGHT        = 4,
   MAXASCENT        = 8,
   MAXDESCENT      = 16,
   MAXWIDTH        = 32,
   DEFCHAR         = 64
} tsa2DFontInfoFlag_t;
```

Description

This type definition enumerates the supported flags to get specific information regarding font, and is used in `tsa2DGetFontInfo`.

tsaFontTMCharMetrics_t

```
typedef struct tsaFontTMCharMetrics_t {
   UInt8    charNo;
   UInt8    chWidth;
   UInt8    pixWidth;
   UInt8    pixHeight;
   char     Ascent;
   char     Descent;
   UInt32   Offset;
} tsaFontTMCharMetrics_t, *ptsaFontTMCharMetrics_t;
```

Fields

charNo	Number of characters.
chWidth	Character width.
pixWidth	Pixel width.
pixHeight	Pixel height.
Ascent	Ascent.
Descent	Descent.
Offset	Offset to the corresponding bitmap in the bit file.

Description

TriMedia Font Character Metrics Specification of `tsaFontTM_t`.

tsaFontTM_t

```
typedef struct tsaFontTM_t {
   UInt8                      minChar;
   UInt8                      maxChar;
   UInt8                      fontType;
   UInt8                      maxHeight;
   UInt8                      maxAscent;
   UInt8                      maxDescent;
   tsaFontTMCharMetrics_t   **charMetrics;
   UInt8                     *bitmaps;
} tsaFontTM_t, *ptsaFontTM_t;
```

Fields

minChar	Minimum number of characters in this font set.
maxChar	Maximum number of characters in this font.
fontType	Font type.
maxHeight	Maximum height.
maxAscent	Maximum ascent.
maxDescent	Maximum descent.
charMetrics	Pointer to pointer of character metrics array.
bitmaps	Pointer to bitmap array.

Description

TriMedia font general data structure.

tsaTMFont2CharMetrics

```
typedef struct tsaFontTM_t {
   UInt8   charNo;
   UInt8   pixWidth;
   UInt8   firstLine;
   UInt8   lastLine;
   Int8    abcA;
   UInt8   abcB;
   Int8    abcC;
   UInt8   spare;
   UInt    offset;
} tsaTMFont2CharMetrics_t, *ptsaTMFont2CharMetrics_t;
```

Fields

charNo	Number of characters.
pixWidth	Pixel width.
firstLine	First line.
lastLine	Last line.
abcA	Point A.
abcB	Point B.
abcC	Point C.
spare	Spare.
offset	Offset.

Description

TriMedia font character metrics specification of `tsaTMFont2_t`.

tsaTMFont2

```
typedef struct tsaFontTM_t {
   UInt8                       fontType;
   UInt8                       minChar;
   UInt8                       maxChar;
   UInt8                       defChar;
   UInt8                       maxHeight;
   UInt8                       maxAscent;
   UInt8                       maxDescent;
   UInt8                       maxWidth;
   tsaTMFont2CharMetrics_t  **charMetrics;
   UInt8                      *bitmaps;
} tsaTMFont2_t, *ptsaTMFont2_t;
```

Fields

fontType	Font type.
minChar	Minimum number of characters in this font set.
maxChar	Maximum number of characters in this font.
defChar	Character definition.
maxHeight	Maximum height.
maxAscent	Maximum ascent.
maxDescent	Maximum descent.
maxWidth	Maximum width.
charMetrics	Pointer to pointer of character metrics array.
bitmaps	Pointer to bitmap array.

Description

TriMedia Font general data structure.

tsa2DFontType_t

```
typedef enum {
   NoFont    = 0,
   TMFont    = 1,
   TMFont2   = 2
} tsa2DFontType_t;
```

Description

This type definition enumerates the font types. Only `TMFont` is currently supported.

tsa2DFont_t

```
typedef struct tsa2DFont {
   tmal2DFontType_t   fontType;
   UInt32             fontID;
   Pointer            pFontPath;
} tsa2DFont_t, *ptsa2DFont_t;
```

Fields

fontType	Minimum number of characters in this font set.
fontID	Maximum number of characters in this font.
pFontPath	Font type.

Description

The user specifies *fontType* and *fpontPath* to locate the font file. Once this font is loaded, library fills in the fontID. Only TMFont *fontType* is currently supported.

tsa2DContext_t

```
typedef struct tsa2DContext {
   ptsa2DColor_t   pPointColor;
   ptsa2DColor_t   pLineColor;
   ptsa2DColor_t   pFillColor;
   ptsa2DColor_t   pTextColor;
   ptsa2DColor_t   pBgColor;
   UInt32          lineStyle;
   UInt32          textStyle;
   UInt32          fillStyle;
   UInt32          bltStyle;
} tsa2DContext_t, *ptsa2DContext_t;
```

Fields

pPointColor	Color used in drawing the point.
pLineColor	Color used in drawing the line.
pFillColor	Color used in drawing the fill the rectangle.
pTextColor	Color used in drawing the text.
pBgColor	Color used in drawing the background.
lineStyle	Line style.
textStyle	Text style.
fillStyle	Fill rectangle style.
bltStyle	Bullet style.

Description

This graphic context data structure contains graphic context information that is used in various APIs.

2D API Function Descriptions

This section describes the 2D API data functions. These data functions are defined in the tsa2D.h header file.

tsa2DGetCapabilities

```
tmLibappErr_t tsa2DGetCapabilities(
   ptsa2DCapabilities_t  *pCap
);
```

Parameters

pCap	Pointer to buffer receiving returned capabilities.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

Retrieves global and 2D capabilities.

tsa2DOpen

```
tmLibappErr_t tsa2DOpen(
   Int   *instance
);
```

Parameters

instance	Pointer to the returned instance value.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_ALLOC	Returned if the function failed to allocate memory.

Description

User calls tsa2DOpen to get an instance ID. This function assigns a unique 2D instance to the caller.

tsa2DClose

```
tmLibappErr_t tsa2DClose(
   Int   instance
);
```

Parameters

instance	Instance to close.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

User calls tsa2DClose when exit. This routine deallocates the 2D instance.

tsa2DRGBtoYUV

```
tmLibappErr_t tsa2DRGBtoYUV(
   Int     instance,
   UInt8   r,
   UInt8   g,
   UInt8   b,
   UInt8   *y,
   UInt8   *u,
   UInt8   *v
);
```

Parameters

instance	Instance.
r	Red value.
g	Green value.
b	Blue value.
**y*	Y value.
**u*	U value.
**v*	V value.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

The function takes in RGB color and converts it to YUV. The returned Y, U, V, values are placed in *y, *u, and *v, respectively. The values are restricted to the range 16–35.

tsa2DYUVtoRGB

```
tmLibappErr_t tsa2DYUVtoRGB(
   Int     instance,
   UInt8   y,
   UInt8   u,
   UInt8   v,
   UInt8   *r,
   UInt8   *g,
   UInt8   *b
);
```

Parameters

instance	Instance.
y	Y value.
u	U value.
v	V value.
**r*	Red value.
**g*	Green value.
**b*	Blue value.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

The function takes in YUV values and converts them to RGB. The returned R, G, B values are placed in *r, *g, and *b, respectively.

tsa2DLoadIndexColorLUT

```
tmLibappErr_t tsa2DLoadIndexColorLUT(
   Int                     instance,
   ptsa2DIndexColorLUT_t   pIndClr
);
```

Parameters

instance	Instance.
pIndClr	Pointer to the index color LUT. The user specifies: 1. Number of entries (`numEntry`) in the index color. 2. The corresponding LUT color type (`LUTColorType`). 3. Pointer to the corresponding array of colors (`pLUTColorData`). Library returns `indexColorLUTID`.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_INDCOLOR_ALLLOC`	Returned if the function failed in memory allocation.

Description

This routine loads user's index color Look Up Table (LUT) to the 2D Library.

tsa2DUnLoadIndexColorLUT

```
tmLibappErr_t tsa2DUnLoadIndexColorLUT(
   Int                     instance,
   ptsa2DIndexColorLUT_t   pIndClr
);
```

Parameters

instance	Instance.
pIndClr	Pointer to the index color LUT. Library unloads this index color LUT in the library.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

This routine unloads the specified index color Look Up Table (LUT) in the 2D Library.

tsa2DGetColorFmIndex

```
tmLibappErr_t tsa2DGetColorFmIndex(
   Int                     instance,
   Int                     index,
   ptsa2DIndexColorLUT_t   pIndexCLUT,
   ptsa2DColor_t           pColor
);
```

Parameters

instance	Instance.
index	Index in the index color look up table (LUT).
pIndexCLUT	Pointer to the index color LUT.
pColor	Pointer to `ptsa2DColor_t`.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.

Description

This function returns the color in `pColor` according to the specified index number in the index and index color look up table in the pIndexCLUT.

The user sets the index color number, specifies the index color look up table to be used, allocates space on pColor. The function gets the corresponding color from the CLUT and put those color values in `pColor`. Only YUV color type is currently supported.

tsa2DPointNC

```
tmLibappErr_t tsa2DPointNC(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPoint,
   ptsa2DColor_t        pColor
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header tmAvFormats.h and packet data.
pPoint	Pointer to coordinate of a point within the input buffer.
pColor	Color to draw the point.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the point specified is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a point in the input packet buffer with specified coordinate and color.

tsa2DLineNC

```
tmLibappErr_t tsa2DLineNC(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DVOColor_t      pColor
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header tmAvFormats.h and packet data.
pPt1	Pointer to point 1.
pPt2	Pointer to point 2.
pColor	Color to draw the line.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_UPLT_BTRT	Returned if error in the upper left and buttom right coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the line specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a line from point 1 to point 2, with color specified by *pColor*, into the input packet buffer.

tsa2DFillRectNC

```
tmLibappErr_t tsa2DFillRectNC(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DVOColor_t      pColor
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header tmAvFormats.h and packet data.
pPt1	Pointer to upper left point.
pPt2	Pointer to bottom right point.
pColor	Color to fill the rectangle.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_INVALID_RECT	Returned if rectangle specified is invalid.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function fills a rectangle in the input packet buffer according to the rectangle specification of the upper left and the bottom right coordinates, and the color to fill the rectangle.

tsa2DImageNC

```
tmLibappErr_t tsa2DImageNC(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DImage_t        pImage
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header tmAvFormats.h and packet data.
pPt1	Pointer to top left point.
pPt2	Pointer to bottom right point.
pImage	Pointer to image.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_UPLT_BTRT	Returned if error in the upper left and buttom right coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function displays an image to the input packet buffer (*pPacket*) according to the rectangle specified in the upper left and bottom right coordinates.

tsa2DTextNC

```
tmLibappErr_t tsa2DTextNC(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt,
   const char          *string,
   ptsa2DFont_t         pFont,
   ptsa2DColor_t        pFColor,
   ptsa2DColor_t        pBColor,
   ptsa2DTextStyle_t    textStyle
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header *tmAvFormats.h* and packet data.
pPt	Pointer to (x, y) of starting position.
**string*	A string of characters to be drawn; can be one single character.
pFont	Pointer to font structure being used.
pFColor	Character string is drawn with this foreground color.
pBColor	Background is filled with this color. User should supply a valid background color, even if it is not used.
textStyle	Character string is drawn with this text style. See tsa2DTextStyle_t. 1. `textOnly`—draw text with foreground color. 2. `textBackColor`—draw text with the foreground color and fill the background with the background color. 3. `textUnderline`—draw the text and underline with the foreground color.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_TMFONT_NULL`	Returned if the TMFont pointer is null.

TWOD_ERR_TMFONT2_NULL	Returned if the TMFont2 pointer is null.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported.
TWOD_ERR_INVALID_RECT	Returned if error in the rectangle coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a string of characters in the input buffer *pPacket* by specifying the (x,y) coordinate *pPt*. The specified starting position is the base point (point between ascent and descent of a character) of the first character in the string. It supports two font types (TMFont and TMFont2), and three text drawing styles (textOnly, textBackColor, textUnderline). User also specifies the desired background and foreground color.

tsa2DSetPixel

```
tmLibappErr_t tsa2DSetPixel(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPixelSet,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to the input buffer packet header and packet data.
pPixelSet	Pointer to coordinate of a pixel.
pContext	Pointer to 2D context. pPointColor of pContext is the color to be used to set the pixel color.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the point specified is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function sets a pixel in the packet buffer with the `pPointColor` of the `pContext`.

tsa2DGetPixel

```
tmLibappErr_t tsa2DGetPixel(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPixelGet,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to the input buffer packet header and packet data.
pPixelGet	Pointer to coordinate of a pixel.
pContext	Return color in the pPointColor of pContext when success.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the point specified is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function gets a pixel color of a specified position in the packet buffer.

tsa2DPoint

```
tmLibappErr_t tsa2DPoint(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to buffer information, type, and data.
pPt1	Pointer to 2D point.
pContext	Pointer to 2D context.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the point specified is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a point to the specified position in the packet buffer with the pPointColor of the pContext.

tsa2DLine

```
tmLibappErr_t tsa2DLine(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to buffer information, type, and data.
pPt1	Pointer to first 2D point.
pPt2	Pointer to end 2D point.
pContext	Pointer to 2D context line color.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_UPLT_BTRT	Returned if error in the upper left and buttom right coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the line specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a line in the input packet buffer with the `pLinrColor` of the `pContext`.

tsa2DFillRect

```
tmLibappErr_t tsa2DFillRect(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to buffer info, type and data.
pPt1	Pointer to top left point.
pPt2	Pointer to bottom right point.
pContext	Pointer to 2D context.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_INVALID_RECT	Returned if rectangle specified is invalid.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function fills a rectangle in the input packet buffer according to the rectangle specification, and the `pFillColor` of the `pContext`.

tsa2DFillPoly

```
tmLibappErr_t tsa2DFillPoly(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPolyPoints,
   Int                  numPoints,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to buffer info, type and data.
pPolyPoints	Pointer to a list of points that form a polygon.
numPoints	Number of points in the polygon.
pContext	Pointer to 2D context.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_ALLOC	Returned if the function failed in memory allocation.
TWOD_ERR_INVALID_POLYGON	Returned if polygon specified is invalid.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function fills a convex polygon in the input packet buffer according to the polygon specification, and the `pFillColor` of the `pContext`.

tsa2DImage

```
tmLibappErr_t tsa2DImage(
   Int                  instance,
   ptmAvPacket_t        pPacket,
   ptsa2DCoordinate_t   pPt1,
   ptsa2DCoordinate_t   pPt2,
   ptsa2DImage_t        pImage,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to buffer info, type, and data.
pPt1	Pointer to top left point.
pPt2	Pointer to bottom right point.
pImage	Pointer to image.
pContext	Pointer to 2D context.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_UPLT_BTRT	Returned if error in the upper left and buttom right coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function copies an image to the input packet buffer (pPacket) according to the rectangle specified in the upper left and bottom right coordinates.

tsa2DText

```
tmLibappErr_t tsa2DText(
   Int                  instance,
   ptmAvPacket_         pPacket,
   ptsa2DCoordinate_t   pPt,
   const char           *str,
   ptsa2DFont_t         pFont,
   ptsa2DContext_t      pContext
);
```

Parameters

instance	Instance.
pPacket	Pointer to input buffer packet header and packet data.
pPt	Pointer to (x, y) of starting position.
**str*	Pointer to a string of characters to be drawn; can be one single character.
pFont	Pointer to a valid font.
pContext	Pointer to the 2D Context text and background color.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_TMFONT_NULL	Returned if the TMFont pointer is null.
TWOD_ERR_TMFONT2_NULL	Returned if the TMFont2 pointer is null.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported .
TWOD_ERR_INVALID_RECT	Returned if error in the rectangle coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws a string of characters in the input buffer `pPacket` by specifying the (x,y) coordinate `pPt`. The specified starting position is the base point (point between ascent and descent of a character) of the first charcater in the string. It supports two font types (TMFont and TMFont2), and three text drawing styles (`textOnly`, `textBackColor`, `textUnderline`). It uses the `pTextColor` of the `pContext` as the text foreground color and it uses the `pBgColor` of the `pContext` as the background color.

tsa2DBlt

```
tmLibappErr_t tsa2DBlt(
   Int               instance,
   ptmAvPacket_t     pDstPacket,
   ptmAvPacket_t     pSrcPacket,
   ptmAvPacket_t     pDstStartPt,
   ptsa2DContext_t   pContext
);
```

Parameters

instance	Instance.
pDstPacket	Pointer to destination buffer.
pSrcPacket	Pointer to source buffer.
pDstStartPt	Pointer to start (x, y) in destination buffer.
pContext	Pointer to context information.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported .
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function copies the entire source buffer `pSrcPacket` to the specified location in the destination packet buffer `pDstPacket`.

tsa2DBltRegion

```
tmLibappErr_t tsa2DBlt(
   Int               instance,
   ptmAvPacket_t     pDstPacket,
   ptmAvPacket_t     pSrcPacket,
   ptsaCoordinate_t  pDstStartPt,
   ptsaCoordinate_t  pSrcStartPt,
   Int               width,
   Int               height,
   ptsa2DContext_t   pContext
   Int               ops
);
```

Parameters

instance	Instance.
pDstPacket	Pointer to destination buffer.
pSrcPacket	Pointer to source buffer.
pDstStartPt	Pointer to start (x, y) in destination buffer.
pSrcStartPt	Pointer to start (x, y) in source buffer.
width	Width of the region to be BLT'd.
height	Height of the region to be BLT'd.
pContext	Pointer to context information. Only the YUV422 buffer type is supported and pContext is not currently used.
ops	Logical operation to be performed on the source and destination pixels.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported .
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.

TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function copies the source buffer with the specified starting position and (width, height) to the destination packet buffer at the sepcified destination starting position.

tsa2DPolyPoint

```
tmLibappErr_t tsa2DPolyPoint(
   Int                  instance,
   ptmAvPacket_        *pPktList,
   Int                  numPkt,
   ptsa2DCoordinate_t   pPtList,
   Int                 *pNumPerPk,
   ptsa2DColor_t        pColor
);
```

Parameters

instance	Instance.
**pPktList*	Pointer to an array of packet pointers.
numPkt	Number of packets to pass in.
pPtList	Pointer to an array of 2D points.
**pNumPerPk*	Pointer to array of `Int` which specifies the number of points to be drawn in each packet.
pColor	Color pointer, the *pColor->pColorData* is a pointer to an array of 2D colors.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_OUT_OF_BOUNDARY`	Returned if the point specified is out of the packet boundary.
`TWOD_ERR_INVALID_POINTER`	Returned if the function encounters an invalid pointer.
`TWOD_ERR_COLOR_TYPE`	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws multiple numbers of points on multiple numbers of packets according to the supplied positions and colors.

tsa2DPolyLine

```
tmLibappErr_t tsa2DPolyLine(
   Int                  instance,
   ptmAvPacket_t       *pPktList,
   Int                  numPkt,
   ptsa2DCoordinate_t   pPt1List,
   ptsa2DCoordinate_t   pPt2List,
   Int                  pNumPerPkt,
   ptsa2DColor_t        pColor
);
```

Parameters

instance	Instance.
pPktList	Pointer to an 'array' of packet pointers.
numPkt	Number of packets to pass in.
pPt1List	Pointer to an array of beginning 2D points.
pPt2List	Pointer to an array of ending 2D points.
pNumPerPkt	Pointer to array of `Int` that specifies number of lines to be drawn in each packet.
pColor	Color pointer, the *pColor->pColorData* is a pointer to an array of 2D colors.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_UPLT_BTRT`	Returned if error in the upper left and buttom right coordinates specification.
`TWOD_ERR_OUT_OF_BOUNDARY`	Returned if the line specification is out of the packet boundary.
`TWOD_ERR_INVALID_POINTER`	Returned if the function encounters an invalid pointer.
`TWOD_ERR_COLOR_TYPE`	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws multiple numbers of lines on multiple numbers of packets according to the supplied lines and colors.

tsa2DPolyFillRect

```
tmLibappErr_t tsa2DPolyFillRect(
   Int                  instance,
   ptmAvPacket_t       *pPktList,
   Int                  numPkt,
   ptsa2DCoordinate_t   pPt1List,
   ptsa2DCoordinate_t   pPt2List,
   Int                 *pNumPerPkt,
   ptsa2DColor_t    pColor
);
```

Parameters

instance	Instance.
pPktList	Pointer to an array of packet pointers.
numPkt	Number of packets to pass in.
pPt1List	Pointer to an array of upper left 2D points.
pPt2List	Pointer to an array of bottom right 2D points.
pNumPerPkt	Pointer to array of `Int` which specifies number of fill-rectangles to be drawn in each packet.
pColor	Color pointer, the *pColor->pColorData* is a pointer to an array of 2D colors.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_INVALID_RECT	Returned if rectangle specified is invalid.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function fills multiple numbers of rectangles on multiple numbers of packets according to the supplied rectangles and colors.

tsa2DPolyImage

```
tmLibappErr_t tsa2DPolyImage(
   Int                  instance,
   ptmAvPacket_t       *pPktList
   Int                  numPkt
   ptsa2DCoordinate_t   pPt1List
   ptsa2DCoordinate_t   pPt2List
   Int                 *pNumPerPkt
   ptsa2DImage_t    *pImageList
);
```

Parameters

instance	Instance.
**pPktList*	Pointer to an array of packet pointers.
numPkt	Number of packets to pass in.
pPt1List	Pointer to an array of beginning 2D points..
pPt2List	Pointer to an array of ending 2D points.
**pNumPerPkt*	Pointer to array of `Int` which specifies number of lines to be drawn in each packet.
**pImageList*	Pointer to an array of image pointers.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_UPLT_BTRT`	Returned if error in the upper left and buttom right coordinates specification.
`TWOD_ERR_OUT_OF_BOUNDARY`	Returned if the rectangle specification is out of the packet boundary.
`TWOD_ERR_INVALID_POINTER`	Returned if the function encounters an invalid pointer.
`TWOD_ERR_COLOR_TYPE`	Returned if the color type is not consistent with the packet buffer type.

Description

The function copies multiple numbers of images to multiple numbers of packets according to the supplied images and destination rectangle locations.

tsa2DPolyText

```
tmLibappErr_t tsa2DPolyText(
   Int                  instance,
   ptmAvPacket_t       *pPktList,
   ptsa2DCoordinate_t   pPtList,
   const char         **string,
   ptsa2DFont_t        *pFontList,
   ptsa2DColor_t        pFColor,
   ptsa2DColor_t        pBColor,
   tsa2DTextStyle_t    *textStyle,
   Int                 *pNumPerPkt
);
```

Parameters

instance	Instance.
pPktList	Pointer to an array of packet pointers.
pPktList	Number of packets to pass in.
pPtList	Pointer to an array of starting positions.
string	Pointer to an array of string of characters to be drawn.
pFontList	Pointer to an array of loaded fonts.
pFColor	foreground (or text) color pointer, the *pColor->pColorData* is a pointer to an array of 2D colors.
pBColor	Background is filled with this color. User should supply a valid background color, even if it is not used.
textStyle	Pointer to an array of text styles. It can be either: *textOnly* (drawstext with foreground color), *textBackColor* (draws text with foreground color and fill the back with background color), or *textUnderline* (draws the text and underline with foreground color).
**pNumPerPkt*	Pointer to array of `Int` which specifies number of lines to be drawn in each packet.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_TMFONT_NULL	Returned if the TMFont pointer is null.
TWOD_ERR_TMFONT2_NULL	Returned if the TMFont2 pointer is null.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported .
TWOD_ERR_INVALID_RECT	Returned if error in the rectangle coordinates specification.
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.
TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function draws multiple strings of characters to multiple numbers of packets according to the supplied input information.

tsa2DPolyBlt

```
tmLibappErr_t tsa2DPolyBlt(
   Int                 instance,
   ptmAvPacket_t      *pDstPktList,
   ptmAvPacket_t      *pSrcPktList,
   int                 numPkt,
   ptsa2DCoordinate_t  pDstStartPtList,
   ptsa2DCoordinate_t  pSrcStartPtList,
   Int                *pNumPerPkt,
   Int                *pWidthList,
   Int                *pHeightList,
   ptsa2DContext_t   pContext
);
```

Parameters

instance	Instance.
pDstPktList	Pointer to an array of dst packet pointers.
pSrcPktList	Pointer to an array of src packet pointers.
numPkt	number of packet pass in.
pDstStartPtList	Pointer to an array of destination (*dst*) starting points.
pSrcStartPtList	Pointer pointer to an array of source (*src*) starting points.
pNumPerPkt	Pointer to array of int which specifies number of fill-rectangles to be drawn in each packet.
pWidthList	Pointer to an array of width.
pHeightList	Pointer to an array of height.
pContext	Pointer to context. This is not used currently.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_NOT_SUPPORTED	Returned if the specified font type or text style are not supported .
TWOD_ERR_OUT_OF_BOUNDARY	Returned if the rectangle specification is out of the packet boundary.

TWOD_ERR_INVALID_POINTER	Returned if the function encounters an invalid pointer.
TWOD_ERR_COLOR_TYPE	Returned if the color type is not consistent with the packet buffer type.

Description

The function copies a number of rectangles from the source to the destination. User specifies the source and destination starting points and width and height for each Blt.

tsa2DGetStrWidth

```
tmLibappErr_t tsa2DGetStrWidth(
   Int           instance,
   const char   *string,
   Int          *width,
   ptsa2DFont_t  pFont
);
```

Parameters

instance	Instance.
string	String for which to get the pixel width.
width	The calculated pixel width, returning to caller.
pFont	Pointer to a valid font.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_TMFONT_GETSTRWIDTH	Returned if the function failed on TMFont font type.
TWOD_ERR_TMFONT2_GETSTRWIDTH	Returned if the function failed on TMFont2 font type.

Description

The function gets the width, in pixels, of the passed string.

tsa2DGetFontInfo

```
tmLibappErr_t tsaGetFontInfo(
   Int                  instance,
   tsa2DFontInfoFlag_t  flag,
   Int                 *retVal,
   ptsa2DFont_t         pFont
);
```

Parameters

instance	Instance.
flag	Flag to indicate the requested font entry.
**retVal*	Return value to caller.
pFont	Pointer to a valid font.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.
TWOD_ERR_TMFONT_ALLOC	Returned when failed in TMFont alloc.
TWOD_ERR_TMFONT2_ALLOC	Returned when failed in TMFont2 alloc.
TWOD_ERR_TMFONT_MTR_FILE	Returned when failed in reading information from *.mtr file on TMFont type of font.
TWOD_ERR_TMFONT2_TM_FILE	Returned when failed in reading information from *.tm file on TMFont2 type of font.
TWOD_ERR_INVALID_FLAG	Returned when invalid flag passed in.

Description

The function gets the specific font information according to the specified flag value.

tsa2DTMFontSetCharSpacingInString

```
tmLibappErr_t tsa2DTMFontSetCharSpacingInString(
   Int   instance,
   Int   spacingTMFont,
);
```

Parameters

instance	Instance.
spacingTMFont	Value of spacing to be set on the TMFont.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

Sets the spacing between characters in a string. The default is 2. This applies only to the TMFont type of fonts.

tsa2DTMFontGetCharSpacingInString

```
tmLibappErr_t tsa2DTMFontGetCharSpacingInString(
   Int    instance,
   Int   *spacingTMFont,
);
```

Parameters

instance	Instance.
spacingTMFont	Value of spacing to be retrieved on the TMFont.

Return Codes

TMLIBDEV_OK	Returned if the function completes successfully.

Description

Gets the spacing between characters in a string. The default is 2. This applies only to the TMFont type of fonts.

tsa2DLoadFont

```
tmLibappErr_t tsaLoadFont(
   Int            instance,
   ptsa2DFont_t   pFont
);
```

Parameters

instance	Instance.
pFont	Pointer to `tsa2DFont_t`. The user provides information regarding font type and font path. Library loads in the specified font and return a `fontID` in the `tsa2DFont_t` structure.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.
`TWOD_ERR_TMFONT_ALLOC`	Returned when failed in TMFont alloc.
`TWOD_ERR_TMFONT2_ALLOC`	Returned when failed in TMFont2 alloc.
`TWOD_ERR_TMFONT_MTR_FILE`	Returned when failed in reading information from *.mtr file on TMFont type of font.
`TWOD_ERR_TMFONT2_TM_FILE`	Returned when failed in reading information from *.tm file on TMFont2 type of font.
`TWOD_ERR_TMFONT_BIT_FILE`	Returned when failed in reading information from *.bit file on TMFont type of font.
`TWOD_ERR_TMFONT2_BIT_FILE`	Returned when failed in reading information from *.bit file on TMFont2 type of font.

Description

The function loads the font specified in the font path to the 2D Library.

tsa2DUnLoadFont

```
tmLibappErr_t tsaUnLoadFont(
   Int            instance,
   ptsa2DFont_t   pFont
);
```

Parameters

instance	Instance.
pFont	Pointer to `tsa2DFont_t`. Library looks up the font type on the `fontID` in the `tsa2DFont_t` struct and unloads it.

Return Codes

`TMLIBDEV_OK`	Returned if the function completes successfully.

Description

The function unloads the font specified by `pFont` from the 2D Library.

