Technology Consulting Company
Research, Development &
Global Standard

How DirectFB Adopted Market Specific Requirements

Takanari Hayama, Hisao Munakata, and
Denis Oliver Kropp

Technology Consulting Company IGEL Co.,Ltd.

What is DirectFB?

- Thin Graphics Library
  - Light weight and small footprint (< 700KB Library for SH4)
  - Not server/client model like X11
- Hardware Abstraction Layer for Hardware Graphics Acceleration
  - Anything not supported by Hardware still supported by software, i.e. utilize hardware where possible
- Multi-process support
- And others

DirectFB for Japanese Digital TV

- Denis and Renesas have been working to enhance DirectFB to meet particular requirements by Japanese Broadcasting Standard, ARIB.
- Requirements by ARIB includes:
  - Clip region support
  - AYUV support
  - AYUV CLUT support
  - 2bpp fonts support w/ CLUT2
  - SJIS + ARIB Character encoding support

Additional Requirements to Support Digital TV Platform

- Following Requirements needed to be met due to low cost hardware design and Japanese specific situation
  - Efficient multiple layer composition
  - Efficient use of BLIT
  - Efficient Font Cache Management

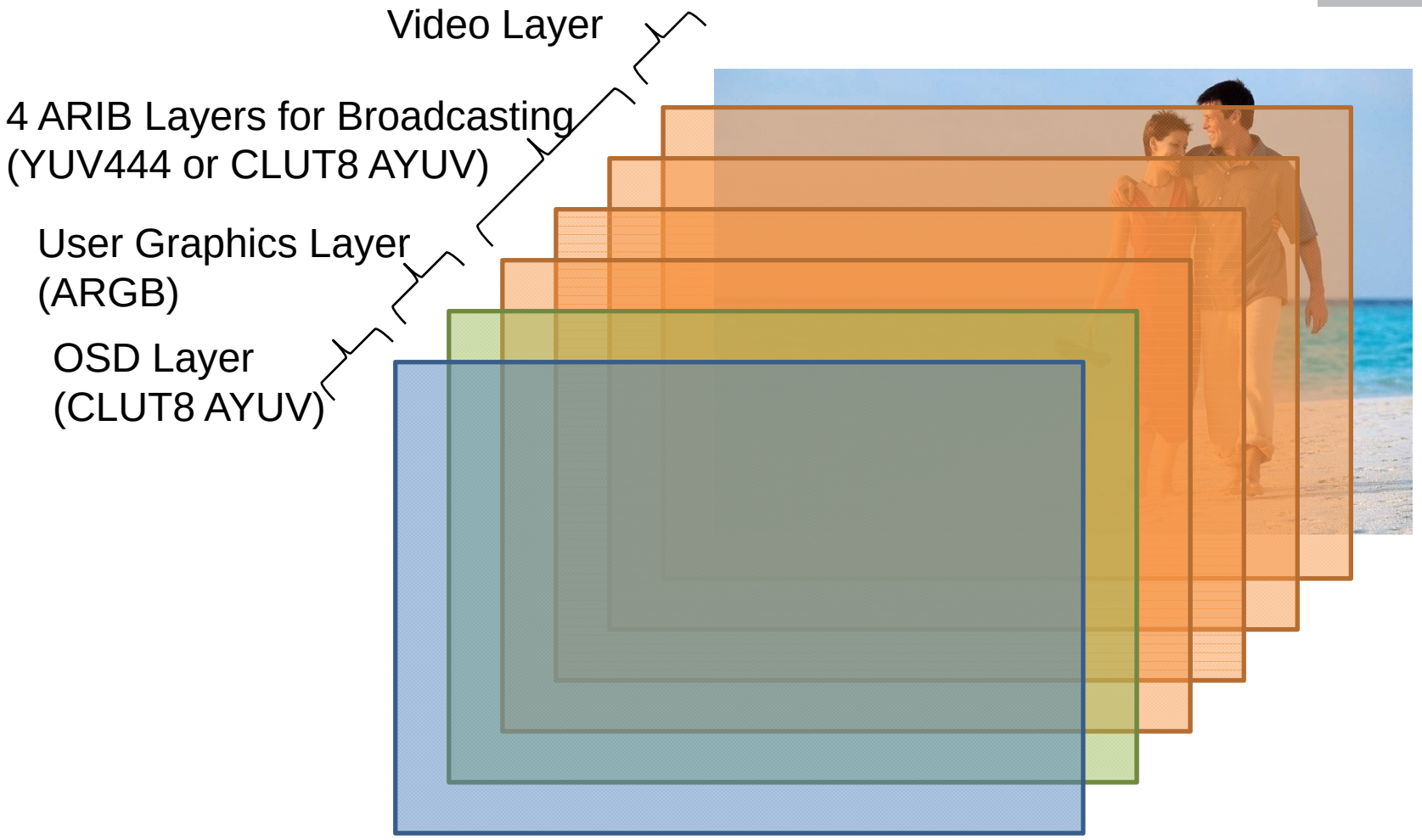
Our Approach to meet ARIB Digital TV requirements

- In general, our approach was not to jeopardize the evolution of existing DirectFB framework.
  - Work closely with Code Maintainer
- Denis had worked out to capture our requirements and where needed, added new API for the particular purpose, but not to be specific only to ARIB.
  - Newly designed API are useful in general.
- Anything that are platform dependent were hidden in gfxdriver, as there was no needs to exposed APIs.

DirectFB 1.0+ meets ARIB

- Clip region support
- AYUV support
- 2bpp fonts support w/ CLUT2
- ARIB Character Encoding support

ARIB Digital TV Graphic Layer Organization

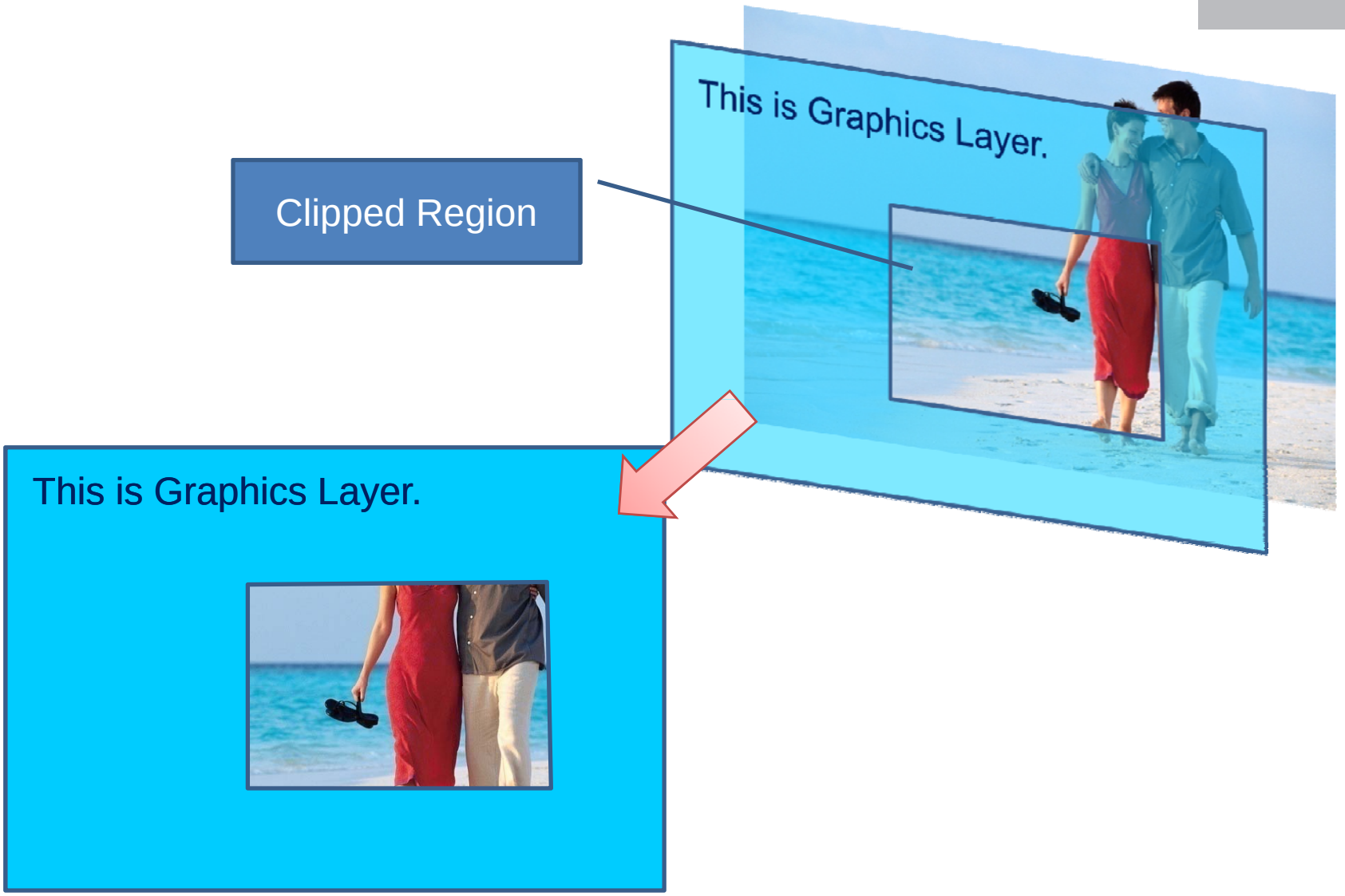


Note: Video Layer is not handled by DirectFB

Clip Region Support

- ARIB Requires to support clipping of layer to make some part of video visible.
- This has to be supported via new API.
  - *API | IDirectFBDisplayLayer [1.0.0-rc1]*
    Added SetClipRegions() that, if supported by hardware, sets the clipping regions that are used to enable or disable visibility of parts of the layer. The number of regions is stated in the display layer description. The layer will be shown only in these regions or except in these regions.
    http://www.directfb.org/index.php?path=Main%2FNews

Clip Region Support (contd.)



DirectFB 1.0+ meets ARIB

- Clip region support
- AYUV support
- 2bpp fonts support w/ CLUT2
- ARIB Character Encoding support

AYUV Support

- All Colors are specified in AYUV.
- This required new API and definition in DirectFB
  - *API [0.9.25]*
    Added DSPF_AYUV, a 32bit packed AYUV format for graphics, being the counterpart of ARGB in the YUV color space.

AYUV CLUT Support

- Upcoming DirectFB 1.1 will support AYUV CLUT as well.
  - New methods in `IDirectFBPalette`: `SetEntriesYUV()`, `GetEntriesYUV()` and `FindBestMatchYUV()`.
  - New `DFBColorYUV` type.
  - Support of YUV CLUT in `DFBSurfaceDescription` and `DFBPaletteDescription`.
- DirectFB maintains 2 sets of CLUT; One in RGB and one in YUV. Setting color index for either one will set another one automatically. Gfxdriver can pick appropriate one depending on the layer's pixel format.

Example for Creating Surface with AYUV CLUT


```
static const DFBSurfaceDescription pngtest3_png_desc = {
  flags    : DSDESC_WIDTH | DSDESC_HEIGHT |
             DSDESC_PIXELFORMAT | DSDESC_PREALLOCATED |
             DSDESC_PALETTE,
  width    : 175,
  height   : 200,
  pixelformat  : DSPF_LUT8,
  preallocated : {{  data : (void *) pngtest3_png_data,
                     pitch : 176  }},
  palette :    {  entries_yuv : pngtest3_png_palette,
                  entries     : NULL,
                     size     : 256  }
};
```

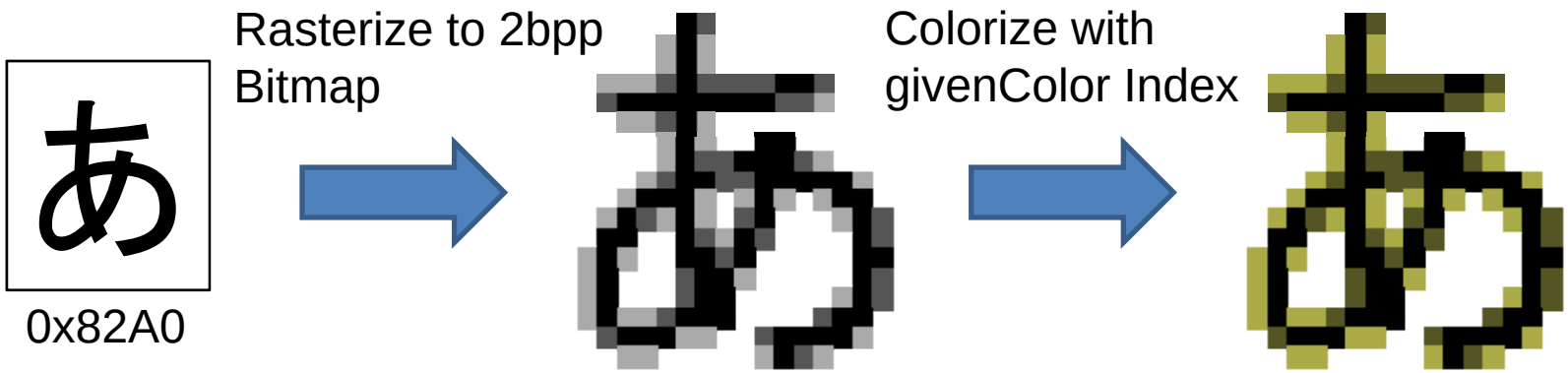
DirectFB 1.0+ meets ARIB

- Clip region support
- AYUV support
- 2bpp fonts support w/ CLUT2
- ARIB Character Encoding support

2bpp and CLUT2 Support

- ARIB requires the font to be rasterized in 2bpp format. 2bpp bitmap is then colorized with specified indexed color.
  - No anti-aliasing allowed. Everything is particularly directed by broadcaster. No exception.

Font Rendering in ARIB



- To realize this process, 2bpp pixel format and CLUT conversions are added.

LUT2 Support

- New pixel format in DFBSurfacePixelFormat.
  - `API [1.0.0-rc2]`
    `Added DSPF_LUT2 being indexed 2 bit per pixel packed, i.e. four pixel in one byte.`

CLUT Conversion Support

- CLUT to CLUT translation
  - *API | IDirectFBSurface [1.0.0 rc2]*
    - *Added SetIndexTranslation() that sets a translation table used for fast indexed to indexed pixel format conversion. Negative or undefined values in the table will result in no pixel being written.*
    - *Added DSBLIT_INDEX_TRANSLATION to do fast indexed to indexed translation, this flag is mutual exclusive with all others.*
    - *So far only LUT2 to LUT8 is supported, used for fast anti-aliased text rendering to indexed format. The translation table contains "-1, 36, 37, 38" for example. First entry makes the background transparent and the other three are shaded versions of the text color.*

CLUT Translation Example


```
int indices[4] = { -1, 20, 21, 22 };

surface->SetBlittingFlags( surface,
  DSBLIT_INDEX_TRANSLATION );
surface->SetIndexTranslation( surface,
  &indices, 4 );

surface->SetFont( surface, font2bpp );
surface->DrawString( surface, ... );
```

DirectFB 1.0+ meets ARIB

- Clip region support
- AYUV support
- 2bpp fonts support w/ CLUT2
- ARIB Character Encoding support

ARIB Character Encoding Support

Encodings	Description
JIS	Used in Internet mostly, e.g. Email. Aka ISO-2022-JP
ShiftJIS	Used in legacy PC including old Windows.
EUC	Used in UNIX World of course.
Unicode	Used in Windows, UNIX, Web etc.
ShiftJIS + Proprietary Extension	Used in Digital Broadcasting, Mobile Handsets etc.

- ARIB Character Sets falls into “ShiftJIS + Proprietary Extension”
  - No standard way to map proprietary chars to Unicode
  - ShiftJIS Unicode mapping has quite tricky issue
    - 0x5c in SJIS is not usually translated into 0x005c in Unicode.

Arbitrary Character Encoding Support

- API | IDirectFBSurface [1.0.0-rc1]
  - Added SetEncoding() to choose an encoding for the text routines, will be overwritten by SetFont() which takes the default encoding of the new font, also see Fonts section below.
- API | IDirectFBFont [1.0.0-rc1]
  - Added EnumEncodings() enumerating all provided encodings, also see Fonts section below.
  - Added FindEncoding() to look up an encoding directly by its name.
  - Added SetEncoding() for choosing the encoding for local interface methods and as a default for the surfaces.

Arbitrary Character Encoding Support (contd.)

- Fonts [1.0.0-rc1]
  - Added support for other encodings than UTF8. Encodings are (or can be) provided with each font implementation. This model reduces code sharing slightly, but allows higher efficiency via optimized combination of decoding and translation.
  - Every encoding just has a name and an ID, where the name can be chosen freely, except for DTEID_UTF8 which is always available and has the name "UTF8".
  - Implemented UTF8 and "Latin1" encoding in FT2 font loader. Nice demonstration how different encodings can be used, while still having a single glyph cache, which is no longer based on character codes, but on their raw indices.

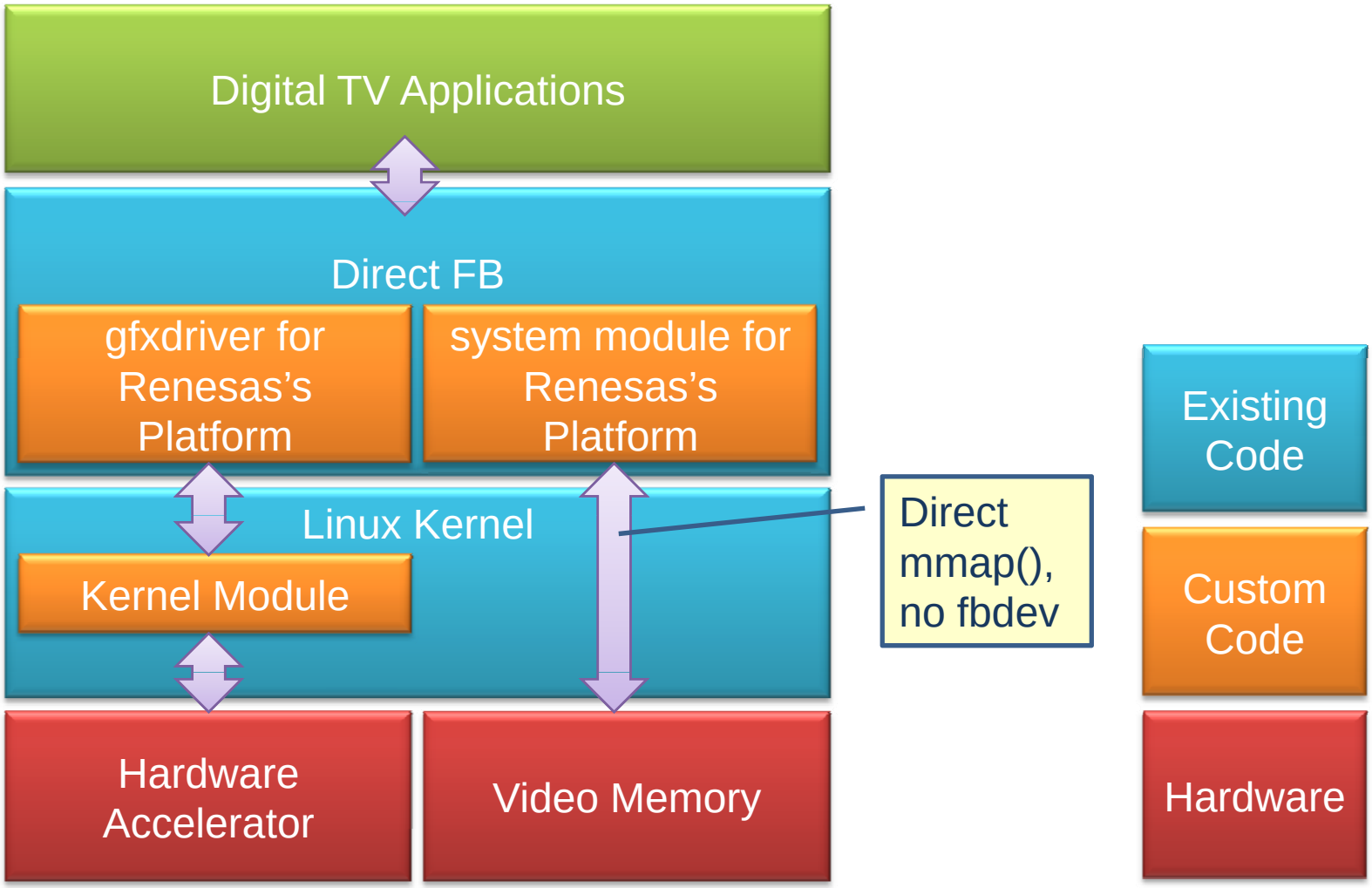
Selection of Character Encoding Example


```
IDirectFB *dfb;
IDirectFBFont*font;
IDirectFB *surface;
DFBTextEncoding ID enc_id;

/* … */

dfb->CreateFont(dfb, ”ARIB_font”, &font_desc,
  &font);
font->FindEncoding(font, ”ARIB”, &enc_id);
font->SetEncoding(font, enc_id);
surface->SetFont(surface, font);
```

Other Enhancements

- Efficient Font Cache Management
  - Original DirectFB cached the rasterized font until the font is destroyed without any limitation.
  - SJIS defines more than 11,000 characters, thus can easily use up memory.
  - `Graphics Core [1.0.0-rc1]`
    - `Implemented font cache limit, currently hardcoded to five rows, each row stores a number of glyphs. When the maximum number is reached it will reuse the least recently used row, kicking out all glyphs that have been on it.`
- 32bit Memory Access
  - Some CPU requires 32bit aligned access, e.g. SH, ARM
  - `Software Renderer [1.0.0-rc1]`
    - `Fix unaligned 32 bit accesses in color keying code.`

DirectFB Software Architecture for Our Platform



Lessons Learned

- Typically in Japan, people tries to add new APIs to meet their requirements.
  - Although LGPL mandates to distribute the modified code, depending on the design, it could not well fit to original API design.
- This would lead us to diversion or fragmentation.
  - Lose opportunity to use the community's latest code.
  - Need to maintain the code by yourself.
  - Lesser merit to use Open Source Assets

Lessons Learned (contd.)

- We believe our case was success, we got everything we need with the functionality in main tree.
  - Win-win situation.
  - Minimum impact to main tree.
  - Not always work like this. ☺
- But, this is one of many approaches industry can take, when they need to add additional requirements to original open source.