# NintendoWare for CTR

## FontConverter Formats

2010/07/01

Ver. 1.0.0

PROVISIONAL TRANSLATION

## Table of Contents

## Code

## Tables

## Figures

# 1 Introduction

## 1.1 About This Manual

This manual serves to explain the different file formats used with NW4C FontConverter (hereafter, FontConverter).

The four formats are:

- Text filter files (xllt), see Chapter 3 Text Filter Files
- Text ordering files (xlor), Chapter 4 Text Ordering Files
- Glyph group files (xggp), Chapter 5 Glyph Group Files
- Image files (bmp, tga), Chapter 6 Image Formats

## 1.2 Font Licenses

Although FontConverter can convert any font installed on a PC for use as a display font on the CTR system, a font license is required in order to market software that uses any of these fonts. Furthermore, appropriate font licenses must be obtained for each game title.

No font licenses are included with FontConverter or NintendoWare for CTR.

# 2 Common XML File Structures

## 2.1 Overview

The XML files used by FontConverter all have a common structure. The specific sections used in common for all files are explained here. FontConverter uses three types of XML files:

- Text filter files (xllt)
- Text ordering files (xlor)
- Glyph group files (xggp)

## 2.2 File Structure

A sample indicating the common sections used in XML files is shown in Code 2-1.

**Code 2-1 Common XML File Structures**

```
<?xml version="1.0" encoding="UTF-8" ?>
<highest-order element version="1.0">
  <head>
          <create user="creating user name" date="2005-02-18T10:51:13" />
          <title>document title</title>
          <comment>document comment</comment>
          <generator name="creating application name" version="1.0.0">
  </head>
  <body>
          optional content
  </body>
</highest-order element>
```

In the general structure, there is a single highest-order element, with a single head and body element within it. All of the structures within the head element are common regardless of the XML specification. The content of the body element varies depending on each XML specification. The name of the highest-order element is defined by each XML specification.

The head element can be omitted and can include as many as one each of the following elements: create, title, comment, or generator.

The create element stores information about the time of the file's creation as attributes. It has four attributes: user, host, date, and source. Each can be omitted. The user attribute stores the user name from the PC on which the file was created. The host attribute stores the name of the PC on which the file was created. The date attribute stores the date and time the file was created in the expanded format specified by ISO 8601. If the file was generated as a result of a conversion from another file, that file's name is stored as the source attribute.

The title and comment elements have no attributes. They store the title of and a comment about the file, respectively.

The generator element stores information about the application that generated the file as attributes. It has name and version attributes, both of which are required. The name attribute stores the name of the application. The version attribute stores the version of the application as a string.

The content of the body element is defined by each XML specification and stores the main part of the data.

**Table 2-1 Common XML File Elements**

| Element Name | Includable Element | Attribute | Description |
|---|---|---|---|
| (Defined separately) | head, body (required) | version (required) | The highest-order element for each XML file. The element name is specified in each XML specification, but the includable elements and attributes are common to all. |
| head | create, comment, title, generator | none | Stores information about the file itself.<br>The structure beneath the head element is common to all files. |
| body | (Defined separately) | none | Stores the primary data for each XML file. Internal specifications are defined in each XML specification. |
| create | none | user, host, date, source | Stores the information about when the file was created.<br>The user attribute stores the user name from the PC which created the file. The host attribute stores the name of the PC. The date attribute stores the date and time of creation. If a file exists that is the source of the data, the name of that file is stored in the source attribute.<br>The date and time is formatted using the extended format defined by ISO 8601. |
| title | none | none | Stores the title of the file. |
| comment | none | none | Stores comments created by the file's creator about the file. |
| generator | none | name (required), version (required) | Stores information about the application that generated the file.<br>A string identifying the application is stored in the name attribute and the version string is stored in the version attribute. |

# 3 Text Filter Files

## 3.1 Overview

The text filter file specifies which characters FontConverter outputs. Use of a text filter file allows only those characters required by the application to be stored as a font resource.

Text filter files are written in XML and use the .xllt extension.

## 3.2 Structure

The XML structure for text filter files is described in Chapter 2 Common XML File Structure. The following is a description of the root element and the structure within the body element not specified in Chapter 2.

Code 3-1 shows the code used in the `sample.xllt`, which is a sample text filter file. When this text filter file is used, the 24 characters “FontConverter a, i, u, e, ka, ki, ku, ke, Nin, ten, do” and a single-byte space character are output.

**Code 3-1 Listing of sample.xllt**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE letter-list SYSTEM "letter-list.dtd">

<letter-list version="1.0">
  <head>
          <create user="NW4C_test" date="2005-02-18T10:51:13" />
          <title>xllt sample</title>
          <comment>text filter file sample</comment>
  </head>

  <body>
          <letter>
                  FontConverter
                  a, i, u, e, ka, ki, ku, ke
                  Nintendo
          </letter>
  </body>

</letter-list>
```

The root element of the text filter is letter-list. This element contains only the head and body elements defined in Chapter 2.

The head element and included elements are not described here because they are all defined in Chapter 2.

The body element includes only the letter element. The characters included in the letter element represent those characters that will be output. Note that white space characters (single-byte space and tab characters) included in the letter element are ignored and that characters that are output always include a single-byte space.

**Table 3-1 Text Filter Defined Elements**

| Element Name | Includable Element | Attribute | Description |
|---|---|---|---|
| letter-list | head, body (required) | version (required) | Defines the text filter. This is the root element of the text filter file. Currently, 1.0 is specified for the required attribute, namely, version. |
| letter | none | none | Defines the text to be output. Characters are specified by directly entering the characters to be output.<br>Note that white space characters used in the specification are ignored. |

## 3.3 DTD

A Document Type Definition (DTD) follows the structural rules of an XML document and contains the items shown in Table 3-1.

The DTD for the text filter file is shown in `letter-list.dtd` located in the `xllt` folder. This folder is located in the same folder as `NW4C_FontConverter.exe`.

## 3.4 Usage

For more on using text filter files, see the GUI and command-line editions of the FontConverter manuals (`FontConverter_Manual.pdf` and `FontConverterConsole_Manual.pdf`, respectively).

## 3.5 Included Text Filter File

FontConverter includes one   text filter file (Table 3-2).

**Table 3-2 Included Text Filter Files**

| File Name | Description |
|---|---|
| `sample.xllt` | Sample file for text filter files. |

# 4 Text Ordering Files

## 4.1 Overview

Text ordering files define the order of characters in BMP files that are input and output by FontConverter. This file specifically defines the number of characters in the horizontal direction in the BMP image, the number of characters in the vertical direction in the BMP image, and the order in which characters are drawn. The text ordering file also functions as a character filter because characters that have no defined location in the text ordering file are not output.

Text ordering files are written in XML format and have the file extension .xlor.

## 4.2 Structure

The XML structure for text ordering files is shown in Chapter 2 Common XML File Structures.

Code 4-1 shows some of the contents of the supplied text ordering file, cp1252.xlor.

**Code 4-1 Listing for cp1252.xlor**

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE letter-order SYSTEM "letter-order.dtd">

<letter-order version="1.0">
  <head>
          <create user="xlor_test" date="2005-02-18T10:51:13" />
          <title>European Language (CodePage 1252 / Latin-1)</title>
          <comment>Windows Code ～(omitted)～ 8859-1 (Latin-1).</comment>
          </head>
  <body>
          <area width="16" />

          <order>
                  <sp/> ! &quot; # $ % &amp; &apos; ( ) * + , - . /
                  0 1 2 3 4 5 6 7 8 9 : ; &lt; = &gt; ?
                  @ A B C D E F G H I J K L M N O
                  P Q R S T U V W X Y Z [ ¥ ] ^ _
                  ` a b c d e f g h i j k l m n o
                  p q r s t u v w x y z { | } ~  <null/>

                  &#x20AC; <null/>   ～(omitted)～ &#x00FE; &#x00FF;
          </order>
  </body>
</letter-order>
```

The root element for text ordering is letter-order, which includes the head and body elements defined in Chapter 2 Common XML File Structures. The version attribute takes a value of either 1.0 or 1.1, which will determine the behavior of the order element.

The description of the head element and included elements are omitted since they are all defined in Chapter 2 Common XML File Structures.

The body element has only the area and order elements. The area element specifies the number of characters in the vertical and horizontal directions in an image, while the order element specifies the order in which characters are drawn.

The area element has no content. It also has two attributes, height and width, used to define the number of characters in the horizontal and vertical directions. Either or both of these attributes may be omitted. If width is omitted, a specification of 16 is assumed. If height is omitted, a value just sufficient for outputting all characters is assumed.

The order element has a list of characters as its content. The order of the characters in this list is the order of the characters in the image. The character in the upper left of the image is the first character, the character to its right is the second character, and so on, from left to right. When the right edge of the image is reached, letter order moves to the second row, where characters are similarly ordered left to right. Since white space characters (single-byte space and tab characters) are ignored, it is necessary to use an sp element instead in order to output a single-byte space. It is also possible to skip one output position by using the null element.

If 1.0 is specified for the version attribute of the letter-order element, characters having the same Unicode encoding may be specified more than once. This would result in an error if 1.1 is specified. We recommend specifying a version of 1.1 to avoid the problems that arise from the use of multiple identical characters as noted in Section 4.6 Cautions Specific to Text Ordering Files.

The preceding is summarized in Table 4-1.

**Table 4-1 Text Ordering Defined Elements**

| Element Name | Includable Element | Attribute | Description |
|---|---|---|---|
| letter-order | head, body (required) | version (required) | Defines the text ordering. This is the root element of the text ordering file. Specify either 1.0 or 1.1 for the required version attribute. |
| area | none | width, height | Defines the number of characters output in the horizontal and vertical directions in the image.<br>The width attribute defines the number of characters in the horizontal direction. If omitted, a value of 16 is assumed.<br>The height attribute defines the number of characters in the vertical direction. If omitted, a value just sufficient for outputting the characters specified by the order element is assumed. |
| order | sp, null | none | Defines the characters to be output. Characters are specified in the order they are to be output. |

| Element Name | Includable Element | Attribute | Description |
|---|---|---|---|
| sp | none | none | Specifies a single-byte space inside the order element. Use this element to specify a single-byte space since entering a literal space will be ignored. |
| null | none | none | Specifies that no character is to be output. As a result, one character space is skipped in the output. |

## 4.3 DTD

The DTD file for the text order file is `letter-order.dtd`, located in the `xlor` folder, which is located in the same folder as `NW4C_FontConverter.exe`.

## 4.4 Usage

### 4.4.1 Command-Line Version

For the command-line version of FontConverter, the text ordering file is specified as a command line option. For more information, see the command-line version of the FontConverter manual, `FontConverterConsole_Manual.pdf`.

### 4.4.2 GUI Version

When the GUI version of FontConverter starts, it reads the files with the `.xlor` extension in the `xlor` folder (`xlor¥*.xlor`). This folder is located in the same folder that contains `NW4C_FontConverter.exe`. FontConverter treats them as text ordering files, showing the title element content in each file as a list item under Text Ordering. Note that text ordering files must have a `.xlor` extension and must be saved in the `xlor` folder alongside `NW4C_FontConverter.exe` in order to be used as a text ordering file.

If you start FontConverter after adding a new text ordering file to the `xlor` folder, the new file will appear as a list item under Text Ordering. If you select the newly added file as an input and output image file, the new text ordering file will be used for input and output of the image file.

If the new text ordering file fails to load at startup due to a syntax error or other cause, the new file will not be displayed under Text Ordering and cannot be used. Be sure to edit the text ordering file based on the contents of the warning message that is displayed when this happens.

## 4.5 Included Text Ordering Files

The following text ordering files are supplied with FontConverter (see Table 4-2).

Text ordering files can include 16 or 94 characters in the horizontal direction. There are five text ordering files that use the basic format of 16 characters in the horizontal direction; the four that include kanji characters are also available in the text ordering file with 94 characters in the horizontal direction. Although space has been added to adjust for the different 16 and 94 character files, they are otherwise the same. Since some image editing applications cannot open files that are longer on any dimension than 32,768 pixels, the 94 character files were added.

**Table 4-2 Included Text Ordering Files**

| **16 Character Filename**<br>**94 Character Filename** | **Displayed by FontConverter** |
|---|---|
| **Description** | |
| `cp1252.xlor`<br>`None` | European Language (CodePage 1252 / Latin-1)<br>None |
| Outputs alphanumeric and some European language characters. This file is called Windows Code Page 1252. It includes all ISO 8859-1 (Latin-1) and ASCII characters. | |
| `JIS_X0201_X0208_01.xlor`<br>`JIS_X0201_X0208_01_94.xlor` | Japanese Level 1 (JIS X 0208)<br>Japanese Level 1 (JIS X 0208) 94 block width |
| Outputs Japanese characters through Level 1 Kanji. This includes single-byte alphanumeric characters and single-byte kana characters. | |
| `JIS_X0201_X0208_012.xlor`<br>`JIS_X0201_X0208_012_94.xlor` | Japanese Level 1,2 (JIS X 0208)<br>Japanese Level 1,2 (JIS X 0208) 94 block width |
| Outputs Level 2 Kanji in addition to the Level 1 Kanji in `JIS_X0201_X0208_01.xlor`. | |
| `cp1252_JIS_X0201_X0208_012.xlor`<br>`cp_1252_JIS_X0201_X0208_012_94.xlor` | European + Japanese Level 1, 2<br>European + Japanese level 1, 2 94 block width |
| Combines cp1252.xlor and JIS_X0201_X0208_012.xlor.<br>**Note:** Because this text ordering file contains sixteen duplicate non-ASCII characters from `cp1252.xlor` and `JIS_X0201_X0208_012.xlor`, this text ordering files replaces those characters from `JIS_X0201_X0208_012.xlor` with <null/>. For more information, see Section 4.6 Cautions Specific to Text Ordering Files. | |

## 4.6 Cautions Specific to Text Ordering Files

It is possible to create text ordering files that specify multiple characters for the same encoding by specifying 1.0 for the version attribute of the letter-order element. When this text ordering file is used to input an image file, multiple glyphs will correspond to a single encoding. In this case, FontConverter will use the first glyph specified within the text ordering file. It will ignore subsequent glyphs and issue a warning when those subsequent glyphs are different from the first one that was used.

When using a text ordering file where the same encoding is specified multiple times, you may encounter a situation where changes made to a glyph in a cell are not reflected in the output font. We therefore recommend that you avoid specifying multiple characters with the same encoding. Specifying 1.1 for the version attribute for the letter-order element will generate an error when multiple characters with the same encoding are specified.

Characters in the text ordering file are internally converted into Unicode because FontConverter uses

Unicode for internal processing. For example, when comparing the different encoding for cp1252 and Shift JIS, the same encoding exists for ASCII characters, European characters, and single-byte katakana. In Unicode, this problem is avoided because the encoding is different for European characters and single-byte katakana. Conversely, Table 4-3 lists the 16 characters that have a different encoding in the two other code sets but are mapped to the same encoding in Unicode. This problem is avoided by using the included text ordering files `cp1252_JIS_X0201_X0208_012.xlor` and `cp1252_JIS_X0201_X0208_012_94.xlor`, which replace the corresponding characters in Shift JIS with <null/>.

**Table 4-3 Characters Mapped to the Same Code in Unicode**

| Glyph | cp1252 Encoding | Shift JIS Encoding | Unicode Encoding |
|---|---|---|---|
| ´ | 0xB4 | 0x814C | U+00B4 |
| ¨ | 0xA8 | 0x814E | U+00A8 |
| … | 0x85 | 0x8163 | U+2026 |
| ‘ | 0x91 | 0x8165 | U+2018 |
| ’ | 0x92 | 0x8166 | U+2019 |
| “ | 0x93 | 0x8167 | U+201C |
| ” | 0x94 | 0x8168 | U+201D |
| ± | 0xB1 | 0x817D | U+00B1 |
| × | 0xD7 | 0x817E | U+00D7 |
| ÷ | 0xF7 | 0x8180 | U+00F7 |
| ° | 0xB0 | 0x818B | U+00B0 |
| § | 0xA7 | 0x8198 | U+00A7 |
| ‰ | 0x89 | 0x81F1 | U+2030 |
| † | 0x86 | 0x81F5 | U+2020 |
| ‡ | 0x87 | 0x81F6 | U+2021 |
| ¶ | 0xB6 | 0x81F7 | U+00B6 |

# 5 Glyph Group Files

## 5.1 Overview

Glyph group files define sets for glyph groups (group sets). They are specified when creating archive fonts in FontConverter and included in the archive fonts. NW4C allows you to specify the glyphs to read from archive fonts for each glyph group that is specified by a glyph group file. Characters not defined in a glyph group file are handled as part of a unique group named “*”.

Glyph group files are written in XML and have the file extension `.xggp`.

## 5.2 Structure

The XML structure for glyph group files is as indicated in Chapter 2 Common XML File Structures.

Code 5-1 shows some of the contents of the supplied glyph group file, `sample.xggp`.

**Code 5-1 Listing for sample.xggp**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE glyph-groups SYSTEM "glyph-groups.dtd">

<glyph-groups version="1.0">
  <head>
          <create user="xgpp_test" date="2006-09-05T14:50:23" />
          <title>sample</title>
  </head>

  <body>          <group name="all">
                  <group name="ascii">
                          <sp/> ! &quot; # $ % &amp; &apos; ( ) * + , - . /
                          0 1 2 3 4 5 6 7 8 9 : ; &lt; = &gt; ?
                          @ A B C D E F G H I J K L M N O
                          P Q R S T U V W X Y Z [ ¥ ] ^ _
                          ` a b c d e f g h i j k l m n o
                          p q r s t u v w x y z { | } ~
                  </group>

                  <group name="european">
                          &#x20AC; &#x201A; &#x0192; &#x201E;
                          -- (omitted) --
                          鼬鼾齊齒齔齣齟齠齡齦齧齬齪齷齲齶
                          龕龜龠堯槇遙瑤凜熙
                  </group>
```

```
        </group>
    </body>
</glyph-groups>
```

The root element for defining glyph groups is the glyph-groups element, which includes the head and body elements defined in Chapter 2 Common XML File Structures.

The head element and included elements are omitted since they are all defined in Chapter 2.

The body element includes only the group element, which specifies the glyph group.

The group element is a string that enumerates the name attribute and its contents. The name attribute indicates the name of the glyph group and the text enumeration that of the text included in the glyph group. Furthermore, the content of the group element can include nested group elements.

Because white space characters (single-byte space characters and tab characters) are ignored, the `sp` element must be used to include single-byte space characters in a glyph group. The only characters that can be used for the name attribute are single-byte alphanumeric characters (0—9, a—z, A—Z) and the underscore(_).

This information is summarized in Table 5-1.

**Table 5-1 Glyph Group Defined Elements**

| Element Name | Includable Element | Attribute | Description |
|---|---|---|---|
| glyph-groups | head, body (required) | version (required) | Defines the glyph group set. This is the root element for the glyph group file. Currently, 1.0 is specified for the required version attribute. |
| body | group | none | The main content of the glyph group file. Multiple group elements can be included. |
| group | sp, group | name (required) | Defines the glyph group. The required attribute, name, specifies the name. Characters included in the glyph group are specified by the enumeration of the characters. The group element can also be nested.<br>The only characters that can be used for the name attribute are single-byte alphanumeric characters (0—9, a—z, A—Z) and the underscore(_). |
| sp | none | none | Specifies a single-byte space inside the order element. Use this element to specify a single-byte space as direct entry of a literal single-byte space will be ignored. |

## 5.3 DTD

The DTD file for the glyph group file is `glyph-groups.dtd`, located in the `xggp` folder, which is in the same folder as `NW4C_FontConverter.exe`.

## 5.4 Supplied Glyph Group Files

The following glyph group files are supplied with FontConverter (Table 5-2).

**Table 5-2 Supplied Glyph Group Files**

| Filename | Displayed by FontConverter |
|---|---|
| **Description** | |
| `sample.xggp` | sample |
| This is a sample file of a glyph group file. Glyph groups are as follows. | |

| Glyph Group Name | | | Description |
|---|---|---|---|
| all | Ascii | | This glyph group consists of the ASCII characters. |
| | European | | This glyph group consists of cp1252 characters except ASCII characters. |
| | Hankana | | This glyph group consists of half-width katakana. |
| | jis_level_0 | jis_level_0a | This glyph group consists of the first half of non-kanji JIS characters.<br>Included are general symbols, numeric symbols, numeric characters, alphabetic characters, hiragana, and katakana. |
| | | jis_level_0b | This glyph group consists of the last half of non-kanji JIS characters.<br>Included are Greek characters, Cyrillic characters, line segments, circled numbers, and ligatures. |
| | jis_level_1 | | This glyph group consists of Level I JIS kanji. |
| | jis_level_2 | | This glyph group consists of Level II JIS kanji. |

## 5.5 Usage

### 5.5.1 Command Line Version

In the command-line version of FontConverter, the glyph group file is specified as a command line option. For more information, see the command line version of the FontConverter manual, `FontConverterConsole_Manual.pdf`.

### 5.5.2 GUI Version

When the GUI version of FontConverter starts, it reads the files with the `.xggp` extension (`xggp¥*.xggp`) from the `xggp` folder located in the same folder that contains `NW4C_FontConverter.exe`. FontConverter treats them as glyph group files, showing the title element content in each file as a selection tree under Group Sets. Note that glyph group files must have a `.xggp` extension and be saved in the `xggp` folder alongside `NW4C_FontConverter.exe` in order to be used as glyph group files by FontConverter.

If you start FontConverter after adding a glyph group file to the `xggp` folder, the new file will appear as a list item under Group Sets. If you select the newly added file during output of a bcfna file, the new glyph group file will be used to output the bcfna file.

If the glyph group file fails to load at startup due to a syntax error or other cause, the new file will not be displayed under Group Sets and cannot be used. When this happens, be sure to edit the glyph group file based on the content of the warning message that is displayed.

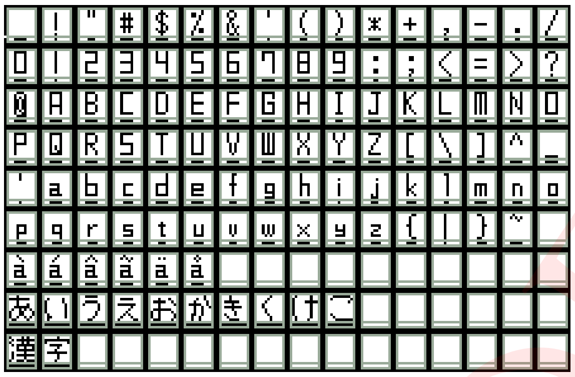
# 6 Image Formats

Images handled by FontConverter have two structures called a block and cell structure. The image consists of a grid of blocks, each containing cells. There are no spaces between blocks.

## 6.1 Blocks

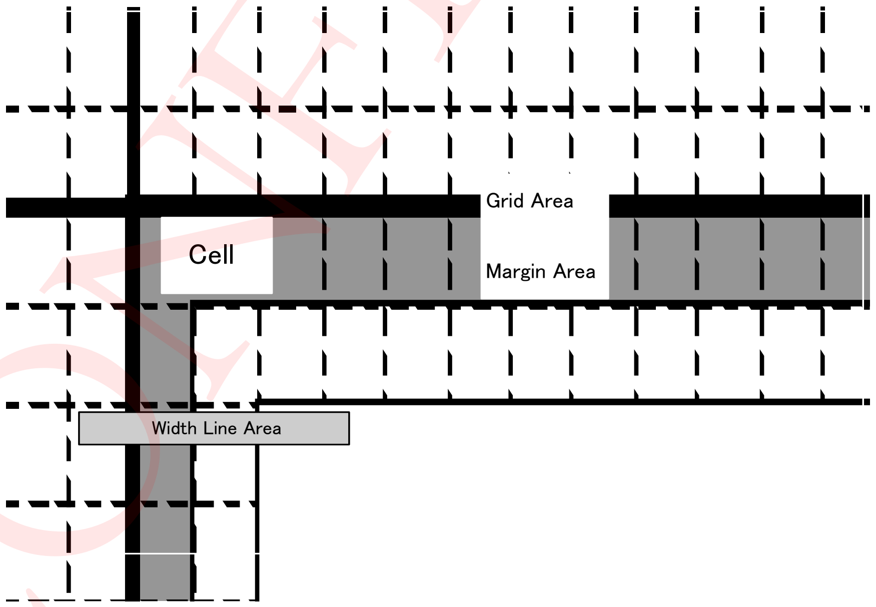
Image blocks are arranged without any spaces between the blocks. The number of blocks in the horizontal and vertical axes is defined in the text ordering file. (See Figure 6-1.)

**Figure 6-1 Sample BMP Image with 16 x 9 Blocks**



Each block includes a cell and a width line. The blocks shown in Figure 6-2 have a width of 14 pixels and a height of 16 pixels. The thick solid line indicates the boundaries of the block while the thin dashed lines indicate the boundaries of each pixel.

**Figure 6-2 Block Schematic**



### 6.1.1 Grid Area

The "grid area" is the one-pixel portion around the inside boundary of the block. In Figure 6-2, the grid area is shown filled in dark grey along the block boundary.

If the Render Grid option is selected when Image is set for output, the grid area will be filled with the grid color and neighboring blocks will create what appears as a 2-pixel grid. If the Render Grid option is not selected, the grid area is filled with the background color instead.

When an image is input, this grid area is simply ignored.

### 6.1.2 Margin Area

The margin area is the one-pixel portion around the inside boundary of the grid area. This area is constructed such that the grid, cell, and width lines do not touch. There is also a one-pixel margin area between the cell and width lines.

If the Render Grid option is selected when Image is output, the margin area is filled with the margin color. If it is not selected, the margin area is filled with the background color.

When an image is input, the margin area is checked if it is filled with solid color. If two or more colors are detected, it is assumed that the glyph may be protruding, so a warning is displayed.

### 6.1.3 Cells

The larger block contained by the margin area is called the cell. The glyph image is drawn within the cell. Since the height of the width line is fixed at one pixel, the cell size is always as follows:

Cell width = Block width - 4 pixels

Cell height = Block height - 6 pixels

### 6.1.4 Width Line Area

The width line is a horizontal line with a height of one pixel drawn under the cell inside the margin area. It defines the width of the character and the relative location of the glyph image.

## 6.2 Cells and Width Lines

The left image in Figure 6-3 shows the pattern diagram of a block containing a 10-by-10 pixel cell and its associated width line.

### 6.2.1 Glyph Images

Only the glyph image is located in the cell. FontConverter detects the smallest rectangle inside the cell that includes either non-white pixels or pixels with alpha values other than 0 and treats that as the glyph image. Therefore, even if the cell is large, white, transparent parts around the glyph image are ignored and have no effect on the output font.

For example, in Figure 6-3 (at left), the glyph image is extracted as a rectangle seven pixels high and five pixels wide—see Figure 6-3 (at right). Furthermore, if the image includes no alpha channel, the smallest rectangle including non-white pixels is treated as the glyph image.

When the relative position of the glyph versus the width line are the same, the font output does not change even if the glyph image is moved left or right inside the cell. For example, the same font as shown in Figure 6-3 is output whether it is drawn as shown on the left or the right in Figure 6-4.

**Figure 6-3 Cell and Width Line Schematic**

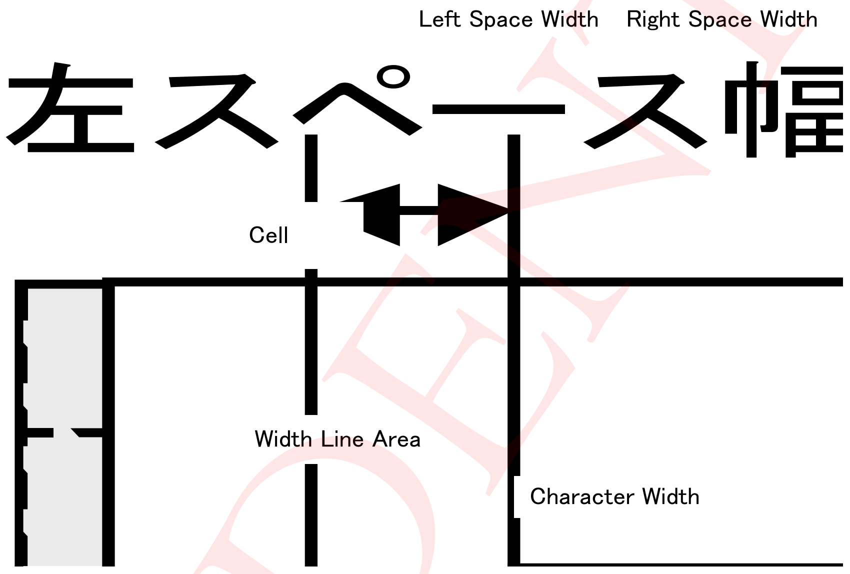
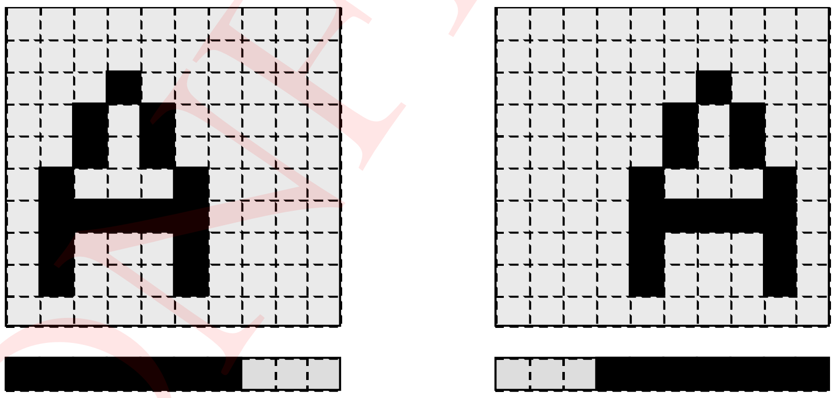


**Figure 6-4 Left-Right Positioning in the Cell**



## 6.2.2 Width Lines

Only a single line called a width line is drawn in the width line area. The width line specifies the character width and the width of spaces to the left and right of the character. If the line is broken into two segments, an error will result.

The horizontal width of the width line represents the character width as is. Since the character width can be made smaller than the glyph image width, it will sometimes be displayed on the CTR system superimposed on top of the previous character. In Figure 6-3, the character width is seven pixels.

The left space width of the character is the difference between the left end of the width line and the left edge of the glyph image. When characters are drawn using the character drawing library, the glyphs are drawn with a space equal to this left space width. In Figure 6-3, with a left space width of one pixel, a five-pixel-wide glyph is drawn one pixel to the right of the specified coordinate.

As with the left space width, the right space width of a character is the difference between the right end of the width line and the right edge of the glyph image. In Figure 6-3, the right space width is also one pixel.

### 6.2.3 Glyphs That Are Not Output

If the glyph image size is 0 and the width line is 0, or if the cell is filled with a single color and the width line is 0, then the glyph is not passed to the output. This can be used to control glyph output without using a character filter. However, if a character is specified for output with a character filter, but the glyph is not passed to the output, a warning message is displayed, as the character might have been dropped unintentionally.

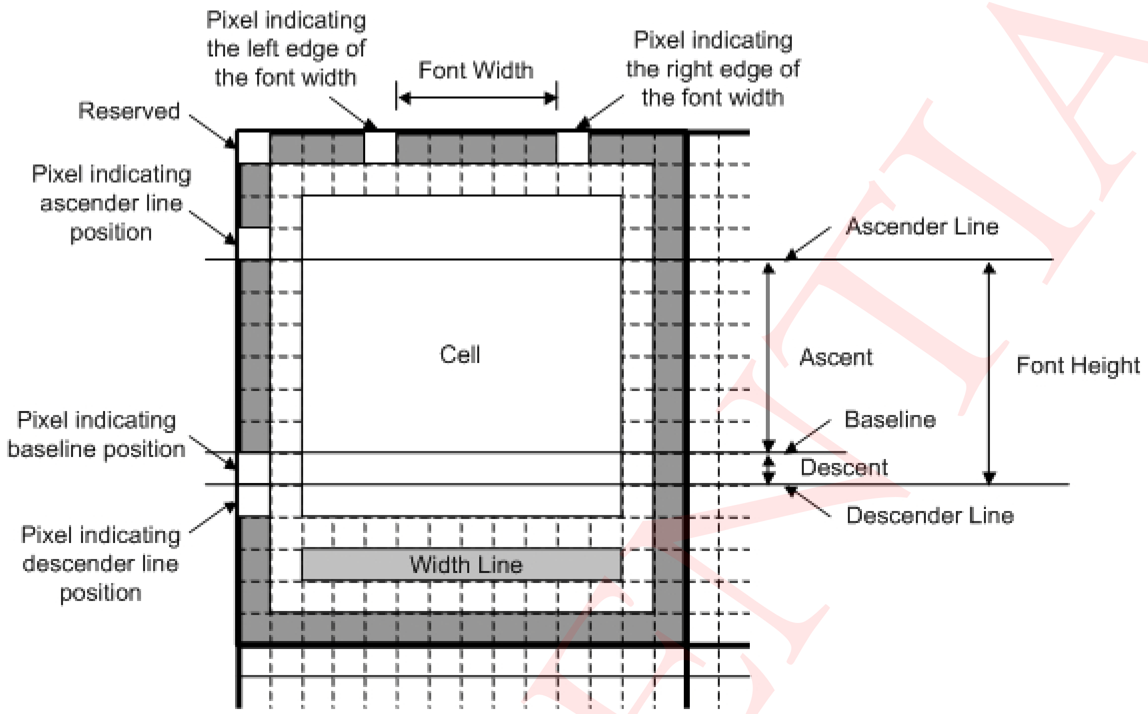
## 6.3 Location Information

Several single pixel points within the image's upper left block serve as base values when drawing strings. There are four base values: baseline, ascender and descender lines, and font width. These four values are expressed with five pixels. The values they express are shared by the entire font and cannot be specified on a letter level.

All of the points fall within the grid area. They are white (255, 255, 255) when the grid is rendered and the grid color when it is not. Although the pixel indicating the baseline position must exist, the other four pixels may be omitted. In addition, the pixel at the upper left of the image is reserved for future use and must therefore always be white (255, 255, 255). This pixel is also used as the transparent color. If this pixel includes an alpha value, it determines that the transparent color includes alpha values. A block including all six points is shown in Figure 6-5.

The following is a description of each of the points.

**Figure 6-5 All Location Information Pixels**

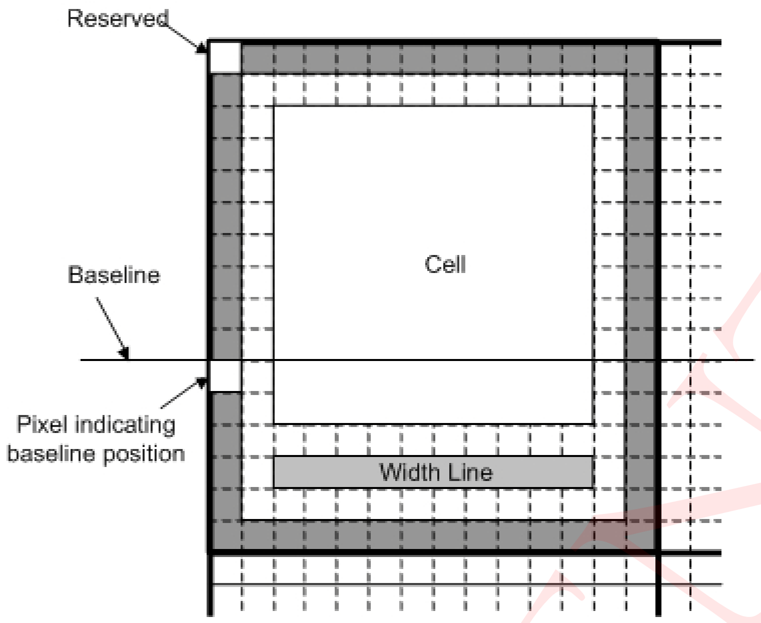

Pixel indicating the left edge of the font width
Font Width
Pixel indicating the right edge of the font width
Reserved
Pixel indicating ascender line position
Ascender Line
Cell
Ascent
Font Height
Baseline
Pixel indicating baseline position
Descent
Descender Line
Pixel indicating descender line position
Width Line

## 6.3.1 Baseline

The baseline position is indicated by a single point at the left of the grid area. The upper edge of this pixel defines the position of the baseline. In other words, the baseline lies between pixels. (See Figure 6-6.)

The baseline serves as the base position when mixing different fonts or when increasing and decreasing font size.

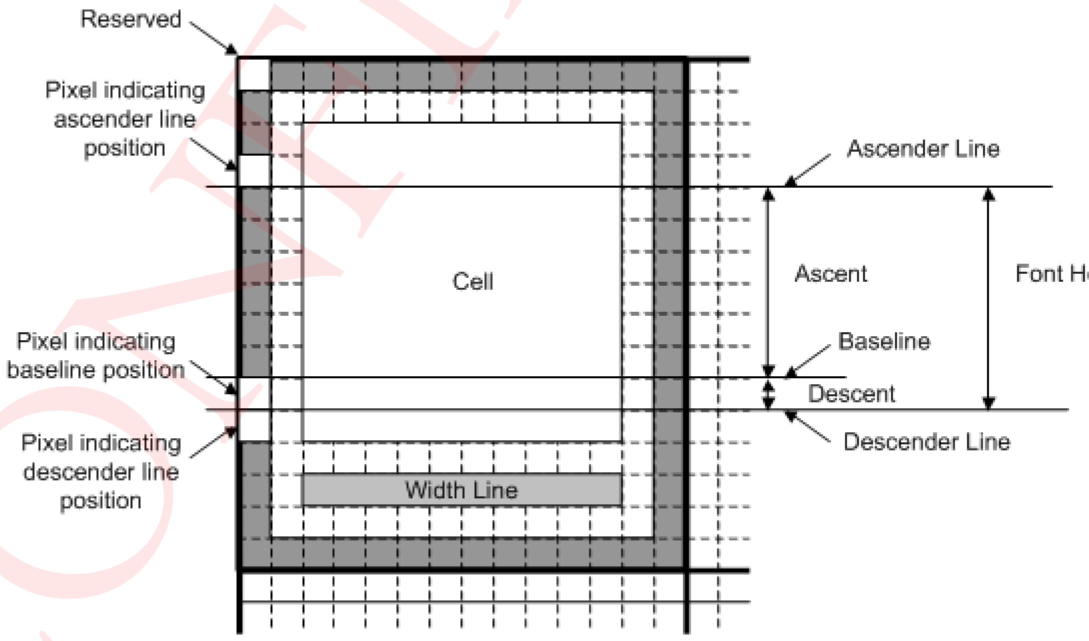
**Figure 6-6 Baseline Position Schematic**



## 6.3.2 Ascender and Descender Lines

The ascender and descender lines are represented by a single point each on the left side of the grid area. Both points for the ascender and descender lines lie between pixels. The position of the ascender line corresponds to the lower edge of the pixel indicating the ascender. The position of the descender line corresponds to the upper edge of the pixel indicating descender. (See Figure 6-7.)

**Figure 6-7 Ascender and Descender Line Position Schematic**


Reserved
Pixel indicating ascender line position
Ascender Line
Ascent
Font Height
Cell
Pixel indicating baseline position
Baseline
Descent
Pixel indicating descender line position
Descender Line
Width Line

The ascender line is always located above the baseline, while the descender line is always located below the baseline. The same pixel can indicate both the descender line and the baseline, in which case the descent is zero. Both the ascent and descent must have values greater than or equal to zero.

These points may be omitted. Table 6-1 indicates how they are to be interpreted based on the number of points available. When there is one point, and therefore no ascender or descender line has been specified, the upper edge of the cell is taken as the ascender line and the lower edge as the descender line.

The image output by FontConverter will not always have pixels rendered to indicate the positions of the ascender line and the descender line. If the input is a Windows font, the pixels are never rendered upon conversion. If the input is bcfnt, these pixels are always rendered upon conversion. If the input is an image file, the pixels indicating the positions of the ascender line and the descender line are rendered if they exist in the input image, but not otherwise. The size of the output image is enlarged automatically if it is too small to depict the pixels that get rendered to indicate the positions of the ascender line and the descender line.

The ascender line is used to represent the top edge of the text; the descender line is used to represent the bottom edge of the text. The distance between these two lines is called the font height. This is the base size used for vertical scaling.

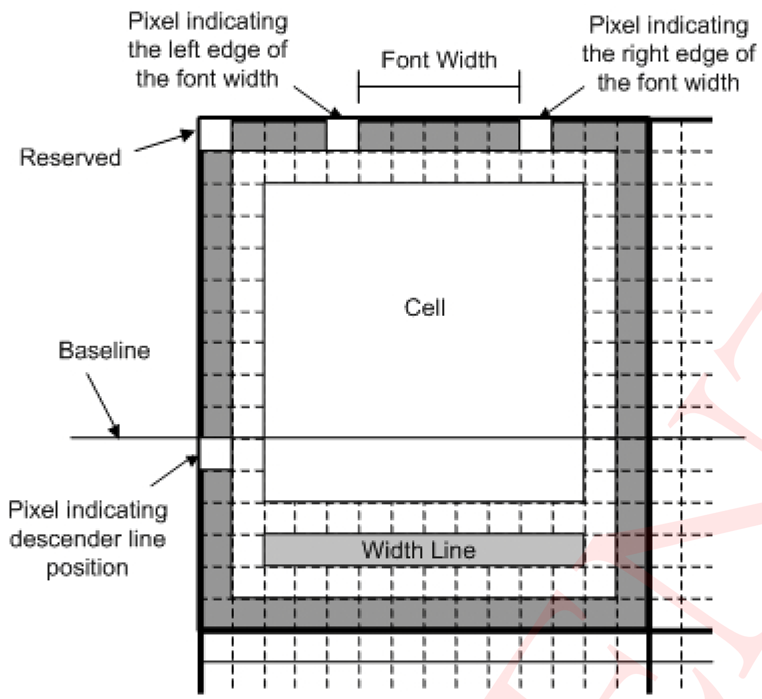
**Table 6-1 Understanding Placement Information Points**

| Number of Points | First Point from Top | Second Point from Top | Third Point from Top |
|---|---|---|---|
| 0 | Error | Error | Error |
| 1 | Point indicating baseline | N/A | N/A |
| 2 | Point indicating ascender line | Point indicating baseline and descender line | N/A |
| 3 | Point indicating ascender line | Point indicating baseline | Point indicating descender line |
| 4 or more | Error | Error | Error |

## 6.3.3 Font Width

The font width is represented by two points at the top of the grid area. They indicate the left and right edges of the font. The number of pixels sandwiched between these two points is equal to the font width. (See Figure 6-8.) It does not matter where in the block these two points are located; so long as the number of pixels between them remains the same, the font width does not change.

**Figure 6-8 Font Width Schematic**


Pixel indicating the left edge of the font width
Font Width
Pixel indicating the right edge of the font width
Reserved
Cell
Baseline
Pixel indicating descender line position
Width Line

If no pixels exist to represent the font width, the cell width is taken to be the font width. If there are pixels representing the font width, there must be two of them.

The image output by FontConverter does not always have pixels rendered to indicate the font width. If the input is a Windows font, the pixels are never rendered upon conversion. If the input is bcfnt, these pixels are always rendered. If the input is Image, the pixels indicating the font width are rendered if they exist in the input image, but not otherwise. The size of the output image is enlarged automatically if it is too small to display the pixels that get rendered to indicate the font width.

The font width is used as the base size for the tab and for horizontal scaling.

# 7 Image File Formats

## 7.1 Image File Output

### 7.1.1 BMP

FontConverter outputs in direct color BMP format at 32 bits per pixel (RGBA8). Resolution is set at 72 ppi.

### 7.1.2 TGA

FontConverter outputs in run length-compressed TrueColor TGA format at 32 bits per pixel (RGBA8). The 16 bytes “FontConverter output.” are set as the image ID.

## 7.2 Image Files Imports

The following common restrictions are placed on image files passed to FontConverter:

Block width × Number of Blocks in Horizontal Direction = Image Width
Block height × Number of Blocks in the Vertical Direction = Image Height

Essentially, there must be no spaces between blocks in the image and no margins.

In reality, the number of blocks in the horizontal and vertical directions is determined by the text ordering file, while the image width (height) is determined based on the image file. The block width and height are calculated from these values. The block width and height must both be integer values.

### 7.2.1 BMP

The following types of BMP files can be input to FontConverter:

- Index color BMP files with 1, 4, or 8 bits per pixel (for 2, 16, or 256 colors)
- Direct color BMP files with 24 bits per pixel (RGB8)
- Direct color BMP files with 32 bits per pixel (RGBA8)

### 7.2.2 TGA

The following types of TGA files can be input to FontConverter:

- Index color (ColorMap) TGA files having 24-bit (RGB8) or 32-bit (RGBA8) palettes with 8 or 16 bits per pixel (for 265 or 65536 colors)
- Direct color (TrueColor) format TGA files with 24 bits per pixel (RGB8) or 32 bits per pixel (RGBA8)

Both run-length compressed and uncompressed formats are supported.

## 7.3 Handling Intensity Format Fonts

For intensity format fonts, the luminosity values for image file glyph images are reversed.

Darker pixels have higher luminosity values and lighter pixels have lower luminosity values.

# 8 Revision History

| Version | Revision Date | Description |
|---|---|---|
| 1.0.0 | 2010/07/01 | Changed format. |
| | 2010/01/15 | Revised figure content. |
| | 2009/10/30 | Initial version. |