**ZI font format version 6 specification**

❗ This is an unfinished reversed engineered specification the TJC USART HMI ZI font format (version 6). ❗ Use at own risk.

The differences between version 5 and version 6 are minimal.

Version 6 can support larger files by using an 8 **byte** alignment of the characterdata. A bit is set in the header when needed and all offsets are divided by 8.

Version 6 also supports subsets of a codepage, allowing for arbitrary sets of characters instead of the full characterset. This way files can be reduced in size and drawing times speeded up while still supporting a large number of characters and UTF-8.

**General information**

The Nextion Font Format is a proprietary font format used by the Nextion/USART Editor HMI software. The editor has a built in "Font Generator"-tool which converts standard fonts into .zi files that is compatible with the Nextion and TJC HMI displays.

|  |  |
| --- | --- |
| Information |  |
| **Software** | Nextion Editor V0.58 |
| **File extension** | .zi |
| **File format version** | ZI version 5 & 6 |
| **Example file** | Consolas\_16\_ascii.zi |

**Code pages / character encoding reference**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code page / encoding | Value | Multibyte | Number of characters | Language |
| ASCII | 0x01 |  | 95 | English |
| [GB2312](https://en.wikipedia.org/wiki/GB_2312) | 0x02 | 01 | 8273 | Simplified Chinese |
| ISO-8859-1 | 0x03 |  | 224 | Western European |
| ISO-8859-2 | 0x04 |  | 224 | Central European |
| ISO-8859-3 | 0x05 |  | 224 | Latin 3 |
| ISO-8859-4 | 0x06 |  | 224 | Baltic |
| ISO-8859-5 | 0x07 |  | 224 | Cyrillic |
| ISO-8859-6 | 0x08 |  | 224 | Arabic |
| ISO-8859-7 | 0x09 |  | 224 | Greek |
| ISO-8859-8 | 0x0A |  | 224 | Hebrew |
| ISO-8859-9 | 0x0B |  | 224 | Turkish |
| ISO-8859-13 | 0x0C |  | 224 | Estonian |
| ISO-8859-15 | 0x0D |  | 224 | Latin 9 |
| ISO-8859-11 | 0x0E |  | 224 | Thai |
| [KS\_C\_5601-1987](https://en.wikipedia.org/wiki/KS_X_1001#Wansung_code_charts) | 0x0F | 01 | 3855 | Korean |
| BIG5 | 0x10 | 01 | 14225 | Traditional Chinese |
| WINDOWS-1255 | 0x11 |  | 224 | Hebrew |
| WINDOWS-1256 | 0x12 |  | 224 | Arabic |
| WINDOWS-1257 | 0x13 |  | 224 | Baltic |
| WINDOWS-1258 | 0x14 |  | 224 | Vietnamese |
| WINDOWS-874 | 0x15 |  | 224 | Thai |
| [KOI8-R](https://en.wikipedia.org/wiki/KOI8-R) | 0x16 |  | 224 | Cyrillic |
| [SHIFT-JIS](https://en.wikipedia.org/wiki/Shift_JIS) | 0x17 | 01 | 8931 | Japanese |
| [UTF-8](https://en.wikipedia.org/wiki/UTF-8) | 0x18 | 01 | 65536 | Unicode |

**File format structure**

**Fixed header**

**Length:** 0x2C (44)

0x00000000: 04 FF 00 0A 01 00 14 28 00 00 20 7E 5F 00 00 00

0x00000010: 03 0E 0E 00 2A 25 00 00 00 00 00 00

| **Offset** | **Length** | **Data** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 0x00000000 | 1 | 04 | byte | 4 | File signature |
| 0x00000001 | 1 | FF | byte | 255 | Start low byte to Skip (FF=none) |
| 0x00000002 | 1 | 00 | byte | 0 | Number of characters to skip |
| 0x00000003 | 1 | 0A | byte | 10 | Orientation (10 = Vertical) |
| 0x00000004 | 1 | 01 | byte | 1 | Encoding |
| 0x00000005 | 1 | 00 | byte | 0 | Multi-byte mode ( 0=single, 1=double, 2=subset) |
| 0x00000006 | 1 | 00 | byte | 0 | Character width = 0 for variable width fonts |
| 0x00000007 | 1 | 28 | byte | 40 | Character height |
| 0x00000008 | 1 | 00 | byte | 0 | Code page start - multibyte first byte |
| 0x00000009 | 1 | 00 | byte | 0 | Code page end - multibyte first byte |
| 0x0000000A | 1 | 20 | byte | 32, ' ' (ASCII) | Code page start - multibyte second byte |
| 0x0000000B | 1 | 7E | byte | 126, '~' (ASCII) | Code page end - multibyte second byte |
| 0x0000000C | 4 | 5F 00 00 00 | uint32 | 95 | Number of characters in encoding |
| 0x00000010 | 1 | 06 | byte | 6 | Font File Version |
| 0x00000011 | 1 | 11 | byte | 17 | Length of description (fontname + encodingname) |
| 0x00000012 | 2 | 00 00 | uint16 | 0 |  |
| 0x00000014 | 4 | 2A 25 00 00 | uint32 | 9514 | Total length of font name and character data |
| 0x00000018 | 4 | 2C 00 00 00 | uint32 | 44 | Start of Data Address (= Font Name location) |
| 0x0000001C | 1 | FF | byte | 255 | Start high byte to Skip (FF=none) |
| 0x0000001D | 1 | 00 | byte | 0 | Number of characters to skip |
| 0x0000001E | 1 | 01 | byte | 1 | Anti-alias |
| 0x0000001F | 1 | 01 | byte | 1 | Variable Width (0=monospace, 1=variable width) |
| 0x00000020 | 1 | 05 | byte | 5 | Length of font name |
| 0x00000021 | 1 | 00 | byte | 0 | 0x01 = Align Chardata offsets to 8 bytes (files >= 16MB) |
| 0x00000022 | 2 | 00 00 | uint16 | 0 | Reserved |
| 0x00000024 | 4 | 00 00 00 00 | uint32 | 0 | Actual Number of Characters of the subset |
| 0x00000028 | 4 | 00 00 00 00 | uint32 | 0 | Reserved |
| 0x0000002C | # | `` | byte[] | 0 | Description: Fontname + Encodingname |
| 0x0000003D | 2 | 20 00 | uint16 | space | Start of Character Map **(see below)** |

**\* The file signature/magic bytes for a .zi file containing the BIG5 code page is different from all other files. For BIG5 the magic bytes are 04 7E 22 0A. Which means that the second and third byte might be variable and have some meaning beyond being magic numbers.** The meaning of byte 2 and 3 are: the start byte to skip and the number of bytes to skip in Big5.

**Description name**

Variable length fontname + encodingname. In this case the font name is 0x11 (17) bytes long as seen in offset 0x00000011. The start of the fontname is indicated by offset 0x00000018 and should be 0x0000002C since the header size is fixed.

0x00000010: 03 0E 0E 00 2A 25 00 00 00 00 00 00 41 72 69 61

0x00000020: 6C 5F 34 30 5F 61 73 63 69 69 00 00 00 00 00 00

| **Offset** | **Length** | **Data** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 0x0000001C | 14 | 41 72 69 61 6C 5F 34 30 5F 61 73 63 69 69 | string | Arial\_40\_ascii | Font name |

**Character Map**

Next the file contains a fixed size lookup table for each character with pointers to the actual location of the pixel data. The Character map length is 10 \* **<Number of characters in file>** as found in offset 0x0000000C. Each character entry is 10 bytes long:

| **Offset** | **Length** | **Data** | **Type** | **Value** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 0x0000003D | 2 | 20 00 | uint16 | space | Character of the Code page |
| 0x0000003F | 1 | 08 | byte | 8 | Character width |
| 0x00000040 | 1 | 00 | byte | 0 | Character kerning left |
| 0x00000041 | 1 | 00 | byte | 0 | Character kerning right |
| 0x00000042 | 3 | B6 03 00 | byte[3] | 950 | Start byte of the character data, as an offset from the start of the charactermap (in this case 0x3D) |
| 0x00000045 | 2 | 06 00 | uint16 | 6 | Length of the character data |
|  |  |  |  |  |  |
| 0x00000047 | # | 21 00 | uint16 | ! | **next character in lookup table** |

Version 6 introduces a flag to align Chardata offsets to 8 bytes. This means the 3 bytes representing the Start byte of the character data is trimmed by 3 bits (divided by 8). When the flag is set, you need to multiply the Start byte value by 8 to get the actual data location within the .zi file. This (ugly hack) accomodates bigger font files.

**Character Data**

The rest of the file contains the binary representation of each character with variable lengths. The start position and length of each character can be derived from the character map above.

The **first** byte of the character data indicates the encoding of the rest of that character data: 0x01 for black/white or 0x03 for 3-bit anti-aliased. Following this start byte are the actual compressed bits defining the appearance of the character.

**Anti-aliased - 0x03 start byte**

There are 8 possible shades of alpha represented by 3 bits: 0b000 being completely transparent to 0b111 being completely opaque.

Each byte in the character data *-excluding the first start byte-* contains 8 bits in the format YZdddddd, where YZ is the drawing mode followed by 6 data bits.

* 4 Black & White drawing modes : Y = 0
  +  YZ = 00 0 xxxxx : Repeat transparent pixel xxxxx times
  +  YZ = 00 1 xxxxx : Repeat opaque pixel xxxxx times
  +  YZ = 01 0 xxxxx : Repeat transparent pixel xxxxx times, followed by **1** opaque pixel
  +  YZ = 01 1 xxxxx : Repeat transparent pixel xxxxx times, followed by **2** opaque pixels
* 2 Alpha drawing modes : Y = 1
  +  YZ = 10 xxx ccc : Repeat transparent pixel xxx times, followed by 1 alpha pixel defined by 3 bits
  +  YZ = 11 ccc ddd : 2 different alpha pixels, each 3 bits

**Black & White**

0x01 start byte.

For Black & White only characters, there is a slight change in the drawing modes to optimize for more black pixels.

* 4 Black & White drawing modes : Y = 0 (same as Anti-alias mode)
  +  YZ = 00 0 xxxxx : Repeat transparent pixel xxxxx times
  +  YZ = 00 1 xxxxx : Repeat opaque pixel xxxxx times
  +  YZ = 01 0 xxxxx : Repeat transparent pixel xxxxx times, followed by **1** opaque pixel
  +  YZ = 01 1 xxxxx : Repeat transparent pixel xxxxx times, followed by **2** opaque pixels
* 3 Extra Black & White drawing modes : Y = 1
  +  YZ = 10 0 xxxxx : Repeat transparent pixel xxxxx times, followed by **3** opaque pixel
  +  YZ = 10 1 xxxxx : Repeat transparent pixel xxxxx times, followed by **4** opaque pixels
  +  YZ = 11 www bbb : www times white pixels followed by bbb opaque pixels, each 3 bits

**Character data example for a 16 pixels tall exclamation mark (!) character**

To be investigated.