

# BDF Font translation

## ■ Initialization

```
In[1]:= Off[General::spell, General::spell1]
```

## ■ Function to expand hex numbers into bits

```
In[2]:= HexExpand = Function[{hex}, Sign[BitAnd[ToExpression["16^^" <> hex], #]] & /@  
Table[2^b, {b, 4 StringLength[hex] - 1, 0, -1}]]
```

```
Out[2]= Function[{hex}, (Sign[BitAnd[ToExpression[16^^<> hex], #1]] &) /@  
Table[2^b, {b, 4 StringLength[hex] - 1, 0, -1}]]
```

## ■ Function to compress bit array into words

```
In[3]:= WordBuild = Function[{bitarray}, Fold[2 #1 + #2 &, 0, bitarray]]
```

```
Out[3]= Function[{bitarray}, Fold[2 #1 + #2 &, 0, bitarray]]
```

---

## Import and process

```
In[4]:= SetDirectory["../src"]
```

```
Out[4]= /home/cmaier/Hackerspaces/sinobit/zpix-pixel-font/src
```

```
In[5]:= FileNames[]
```

```
Out[5]= {charcode.c.snippet, glyph.c.snippet, TGSCC-Unicode.txt, Zpix.bdf, Zpix.vfb}
```

```
In[6]:= Dimensions[bdflines = StringSplit /@ Import["Zpix.bdf", "Lines"]]
```

```
Out[6]= {393 838}
```

```
In[7]:= Dimensions[rawunicodetable = Import["TGSCC-Unicode.txt", "Table"]]
```

```
Out[7]= {8107}
```

```
In[8]:= Take[bdflines, 200]
Out[8]= {{STARTFONT, 2.1}, {FONT, -FontForge-Zpix-Normal-R-Normal--12-120-75-75-P-119-ISO10646-1}, {SIZE, 12, 75, 75}, {FONTPADDINGBOX, 19, 13, -5, -2}, {COMMENT, "Generated by, fontforge,, http://fontforge.sourceforge.net"}, {STARTPROPERTIES, 39}, {FOUNDRY, "FontForge"}, {FAMILY_NAME, "Zpix"}, {WEIGHT_NAME, "Normal"}, {SLANT, "R"}, {SETWIDTH_NAME, "Normal"}, {ADD_STYLE_NAME, ""}, {PIXEL_SIZE, 12}, {POINT_SIZE, 120}, {RESOLUTION_X, 75}, {RESOLUTION_Y, 75}, {SPACING, "P"}, {AVERAGE_WIDTH, 119}, {CHARSET_REGISTRY, "ISO10646"}, {CHARSET_ENCODING, "1"}, {FONTNAME_REGISTRY, ""}, {CHARSET_COLLECTIONS, "ASCII, ISOLatin1Encoding, ISO8859-5, JISX0208.1997, GB2312.1980, BIG5, ISO10646-1"}, {FONT_NAME, "Zpix"}, {FACE_NAME, "ZpixEX2"}, {FONT_VERSION, "2.0"}, {FONT_ASCENT, 10}, {FONT_DESCENT, 2}, {UNDERLINE_POSITION, -1}, {UNDERLINE_THICKNESS, 1}, {X_HEIGHT, 6}, {CAP_HEIGHT, 8}, {RAW_ASCENT, 796}, {RAW_DESCENT, 203}, {NORM_SPACE, 5}, {RELATIVE_WEIGHT, 40}, {RELATIVE_SETWIDTH, 50}, {SUPERSCRIPT_X, 0}, {SUPERSCRIPT_Y, 6}, {SUPERSCRIPT_SIZE, 6}, {SUBSCRIPT_X, 0}, {SUBSCRIPT_Y, 0}, {SUBSCRIPT_SIZE, 6}, {FIGURE_WIDTH, 6}, {AVG_LOWERCASE_WIDTH, 89}, {AVG_UPPERCASE_WIDTH, 95}, {ENDPROPERTIES}, {CHARS, 22043}, {STARTCHAR, space}, {ENCODING, 32}, {SWIDTH, 333, 0}, {DWIDTH, 4, 0}, {BBX, 1, 1, 13, -2}, {BITMAP}, {00}, {ENDCHAR}, {STARTCHAR, exclam}, {ENCODING, 33}, {SWIDTH, 300, 0}, {DWIDTH, 6, 0}, {BBX, 1, 9, 2, 0}, {BITMAP}, {80}, {80}, {80}, {80}, {80}, {80}, {80}, {00}, {80}, {80}, {ENDCHAR}, {STARTCHAR, quotedbl}, {ENCODING, 34}, {SWIDTH, 378, 0}, {DWIDTH, 6, 0}, {BBX, 3, 2, 1, 7}, {BITMAP}, {A0}, {A0}, {ENDCHAR}, {STARTCHAR, numbersign}, {ENCODING, 35}, {SWIDTH, 601, 0}, {DWIDTH, 6, 0}, {BBX, 5, 9, 0, 0}, {BITMAP}, {50}, {50}, {F8}, {50}, {50}, {50}, {F8}, {50}, {50}, {ENDCHAR}, {STARTCHAR, dollar}, {ENCODING, 36}, {SWIDTH, 628, 0}, {DWIDTH, 6, 0}, {BBX, 5, 11, 0, -1}, {BITMAP}, {20}, {70}, {A8}, {A0}, {60}, {20}, {30}, {28}, {A8}, {70}, {20}, {ENDCHAR}, {STARTCHAR, percent}, {ENCODING, 37}, {SWIDTH, 968, 0}, {DWIDTH, 6, 0}, {BBX, 6, 8, 0, 0}, {BITMAP}, {48}, {A8}, {B0}, {50}, {28}, {34}, {54}, {48}, {ENDCHAR}, {STARTCHAR, ampersand}, {ENCODING, 38}, {SWIDTH, 769, 0}, {DWIDTH, 6, 0}, {BBX, 5, 9, 0, 0}, {BITMAP}, {60}, {90}, {90}, {60}, {40}, {A8}, {A8}, {90}, {68}, {ENDCHAR}, {STARTCHAR, quotesingle}, {ENCODING, 39}, {SWIDTH, 203, 0}, {DWIDTH, 6, 0}, {BBX, 1, 2, 2, 7}, {BITMAP}, {80}, {80}, {ENDCHAR}, {STARTCHAR, parenleft}, {ENCODING, 40}, {SWIDTH, 359, 0}, {DWIDTH, 6, 0}, {BBX, 3, 11, 2, -1}, {BITMAP}, {20}, {40}, {40}, {80}, {80}, {80}, {80}, {40}, {40}, {20}, {ENDCHAR}, {STARTCHAR, parenright}, {ENCODING, 41}, {SWIDTH, 359, 0}, {DWIDTH, 6, 0}, {BBX, 3, 11, 1, -1}, {BITMAP}, {80}, {40}, {40}, {20}, {20}, {20}, {20}, {40}, {40}, {80}, {ENDCHAR}, {STARTCHAR, asterisk}, {ENCODING, 42}, {SWIDTH, 437, 0}, {DWIDTH, 6, 0}, {BBX, 5, 5, 0, 2}, {BITMAP}, {20}, {20}, {F8}, {20}}
```

```
In[9]:= Take[rawunicodetable, 32]
Out[9]= {#, Table, of, General, Standard, Chinese, Characters, mapped, to, Unicode},
{index, number, codepoint}, {1, U+4E00}, {2, U+4E59}, {3, U+4E8C}, {4, U+5341},
{5, U+4E01}, {6, U+5382}, {7, U+4E03}, {8, U+535C}, {9, U+516B}, {10, U+4EBA},
{11, U+5165}, {12, U+513F}, {13, U+5315}, {14, U+51E0}, {15, U+4E5D},
{16, U+5201}, {17, U+4E86}, {18, U+5200}, {19, U+529B}, {20, U+4E43},
{21, U+53C8}, {22, U+4E09}, {23, U+5E72}, {24, U+4E8E}, {25, U+4E8F},
{26, U+5DE5}, {27, U+571F}, {28, U+58EB}, {29, U+624D}, {30, U+4E0B}}
```

## ■ Split font file in character descriptions

```
In[10]:= Transpose[Flatten[Position[bdflines, #]] & /@ {"STARTCHAR", _, {"ENDCHAR"}]];
Dimensions[charlines = Take[bdflines, #]] & /@ %
Out[11]= {22 043}
In[12]:= charlines[[1]]
Out[12]= {{STARTCHAR, space}, {ENCODING, 32}, {SWIDTH, 333, 0},
{DWIDTH, 4, 0}, {BBX, 1, 1, 13, -2}, {BITMAP}, {00}, {ENDCHAR}}
In[13]:= charlines[[2]]
Out[13]= {{STARTCHAR, exclam}, {ENCODING, 33}, {SWIDTH, 300, 0},
{DWIDTH, 6, 0}, {BBX, 1, 9, 2, 0}, {BITMAP}, {80}, {80},
{80}, {80}, {80}, {80}, {00}, {80}, {80}, {ENDCHAR}}
```

## ■ Extract character codes

```
In[14]:= Dimensions[
charcodes = Last[Flatten[Select[#, First[#] == "ENCODING" &]]] & /@ charlines]
Out[14]= {22 043}
```

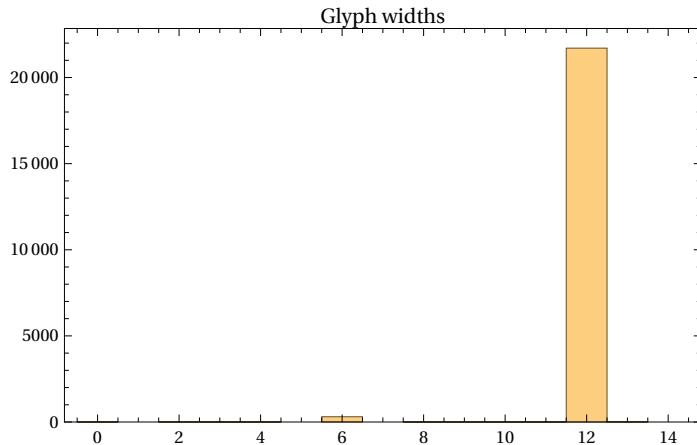
## ■ Extract character widths

```
In[15]:= Dimensions[widths = Flatten[Select[#, First[#] == "DWIDTH" &]]][[2]] & /@ charlines]
Out[15]= {22 043}
```

## ■ Character width statistics

```
In[16]:= TableForm[Sort[Tally[ToExpression /@ widths], #1[[1]] < #2[[1]] &]]
Out[16]//TableForm=
0      2
2      6
3      2
4      1
6      305
8      2
9      3
10     3
11     3
12     21 706
13     10
```

```
In[17]:= Print[Histogram[ToExpression/@widths, {-1/2, 29/2, 1},
  Frame → True, PlotRange → All, PlotLabel → "Glyph widths"]];
```



#### ■ Indices of characters with widths ≠ 12

```
In[18]:= nonstdwidth = Flatten[Position[ToExpression/@widths, w_ /; w ≠ 12]]
```

```
Out[18]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 104, 105, 106, 107,
 108, 109, 110, 113, 114, 115, 117, 119, 120, 121, 122, 123, 124, 125, 126, 127,
 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
 144, 145, 146, 147, 148, 149, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160,
 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176,
 177, 178, 179, 180, 181, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,
 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209,
 210, 211, 212, 325, 326, 329, 332, 333, 336, 339, 345, 346, 347, 349, 350, 351,
 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 364, 365, 366, 369, 370,
 372, 373, 374, 375, 376, 387, 388, 403, 404, 405, 407, 412, 416, 418, 419, 420,
 437, 441, 442, 451, 452, 667, 668, 1354, 21834, 21835, 21836, 21837, 21838,
 21839, 21840, 21841, 21842, 21843, 21855, 21856, 21857, 21858, 21859, 21860,
 21861, 21891, 21974, 21975, 21976, 21977, 21978, 21979, 21980, 21981, 21982,
 21983, 21984, 21985, 21986, 21987, 21988, 21989, 21990, 21991, 21992, 21993,
 21994, 21995, 21996, 21997, 21998, 21999, 22000, 22001, 22002, 22003, 22004,
 22005, 22006, 22007, 22008, 22009, 22010, 22011, 22012, 22013, 22014, 22015,
 22016, 22017, 22018, 22019, 22020, 22021, 22022, 22023, 22024, 22025, 22026,
 22027, 22028, 22029, 22030, 22031, 22032, 22033, 22034, 22035, 22036}
```

#### ■ Extract bounding box dimensions

```
In[19]:= Dimensions[boundingboxes = Flatten[Select[#, First[#] == "BBX" &]] & /@ charlines]
```

```
Out[19]= {22 043, 5}
```

#### ■ Extract bounding box sizes

```
In[20]:= Dimensions[sizes = Take[#, {2, 3}] & /@ ToExpression[boundingboxes]]
```

```
Out[20]= {22 043, 2}
```

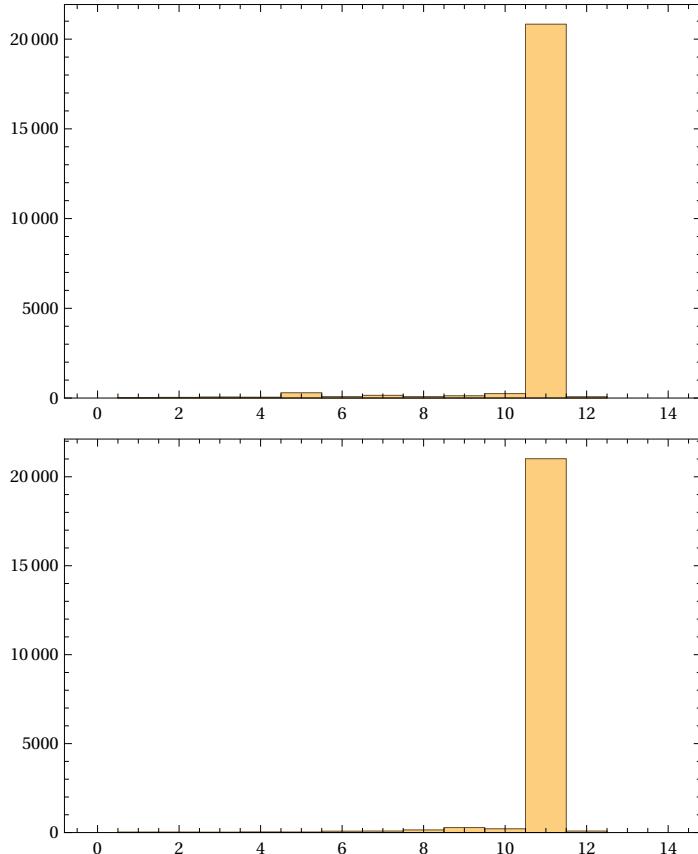
## ■ Maximum bounding box size

```
In[21]:= maxsize = MapThread[Apply, {{Max, Max}, Transpose[sizes]}]
```

```
Out[21]= {12, 12}
```

## ■ Bounding box size statistics

```
In[22]:= Print[Histogram[#, {-1/2, 29/2, 1}, Frame → True, PlotRange → All]] & /@ Transpose[sizes];
```

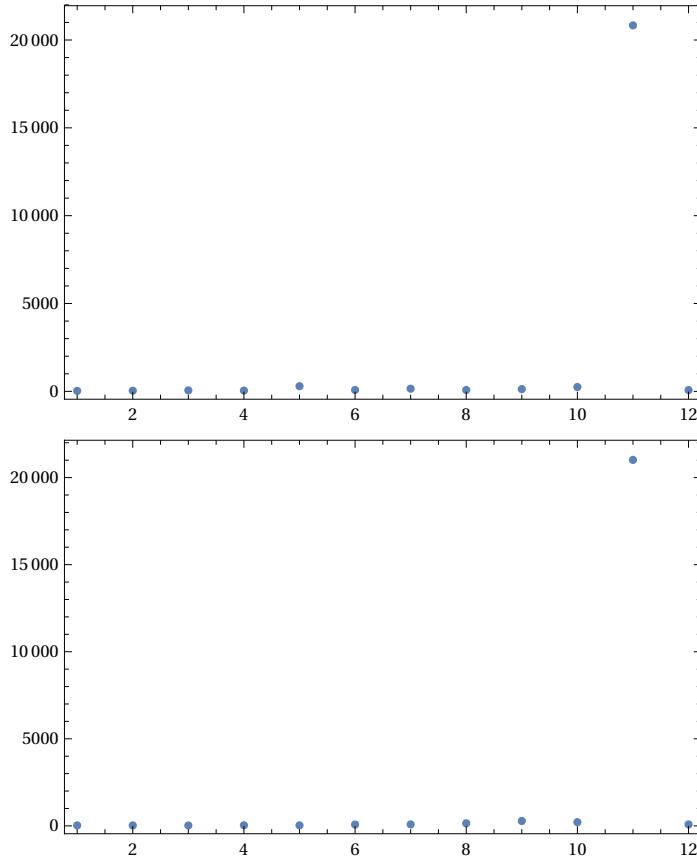


```
In[23]:= TableForm[Sort[Tally[#, #1[[1]] < #2[[1]] &] & /@ Transpose[sizes]]]
```

```
Print[ListPlot[#, Frame → True, PlotRange → All]] & /@ %;
```

```
Out[23]//TableForm=
```

1	2	3	4	5	6	7	8	9	10	11	12
25	36	56	45	294	79	150	79	127	247	20832	73
1	2	3	4	5	6	7	8	9	10	11	12
27	28	23	35	31	76	84	147	281	212	21015	84



#### ■ Extract bounding box positions

```
In[25]:= bounds = Flatten[{Take[#, -2], Take[#, -2] + Take[#, {2, 3}]}] & /@
    ToExpression[boundingboxes];
Dimensions[bounds]
```

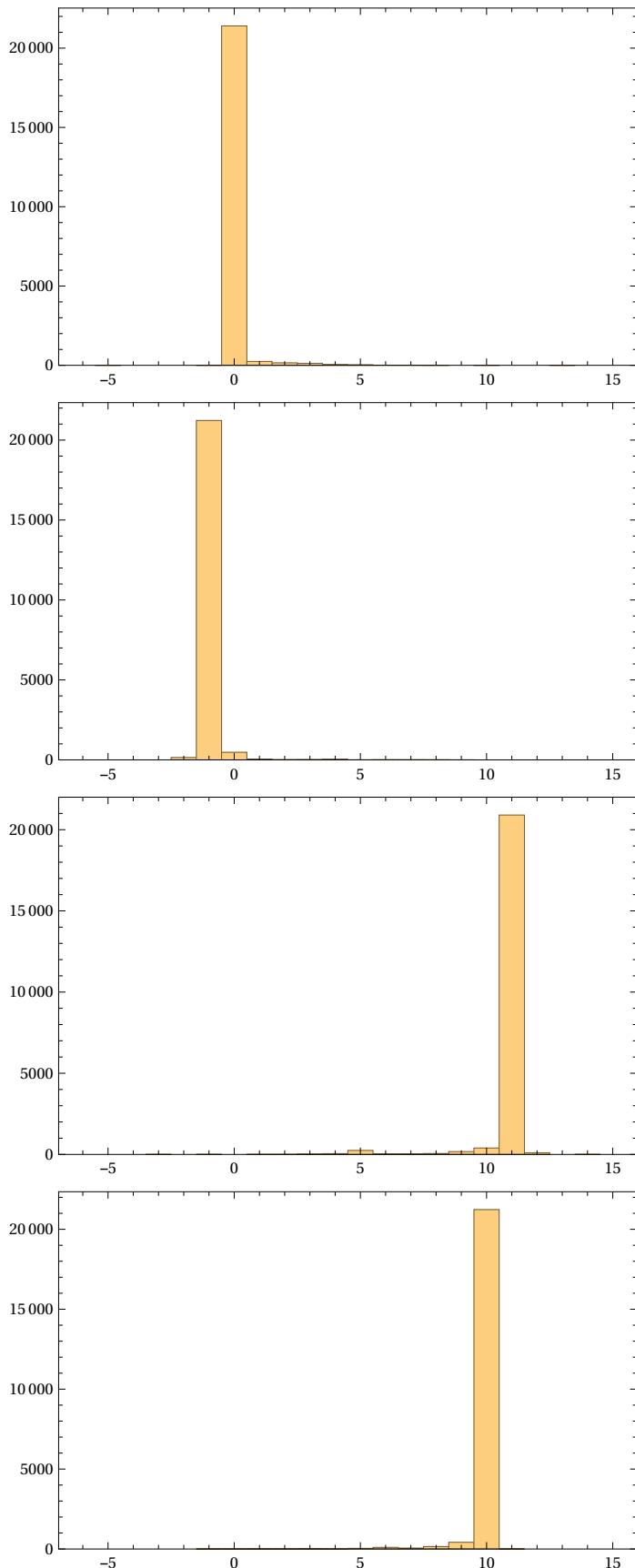
```
Out[26]= {22 043, 4}
```

#### ■ Envelope of all bounding boxes

```
In[27]:= bigbox = MapThread[Apply, {{Min, Min, Max, Max}, Transpose[bounds]}]
Out[27]= {-5, -2, 14, 11}
```

#### ■ Bounding box statistics

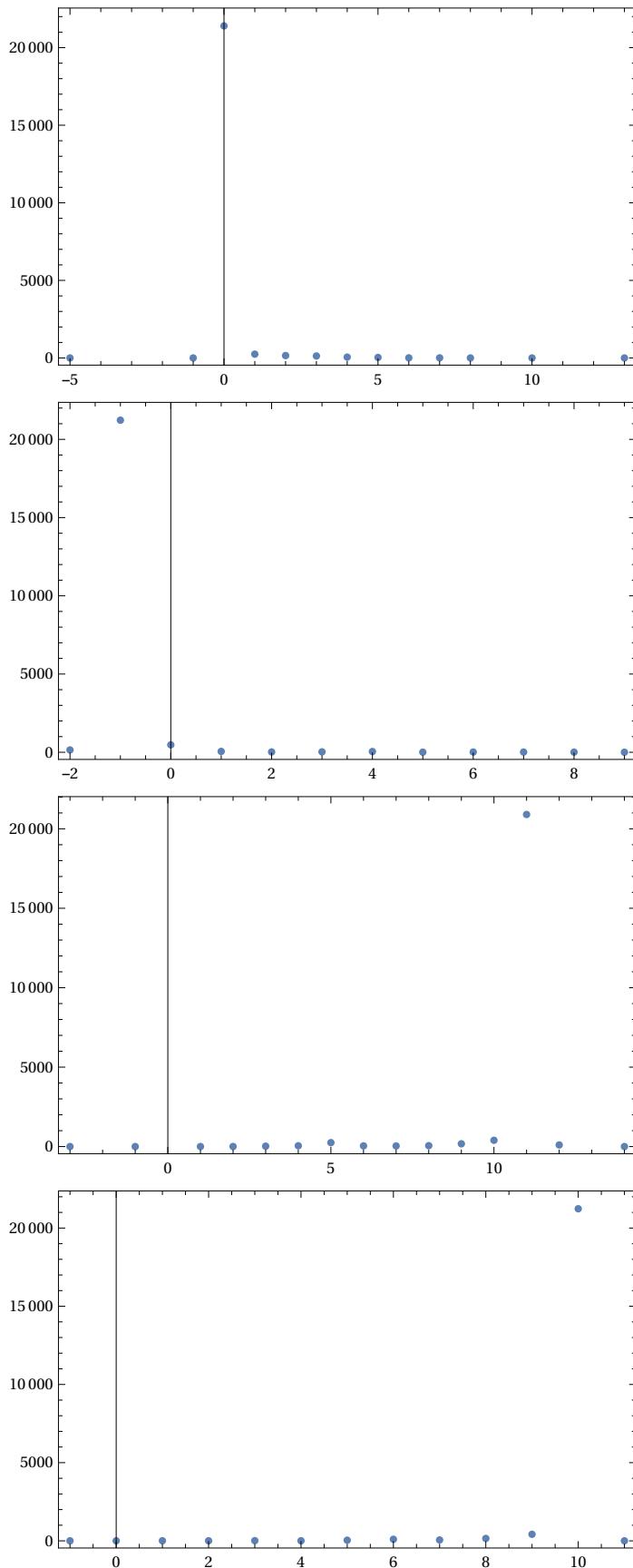
```
In[28]:= Print[Histogram[#, {- $\frac{13}{2}$ ,  $\frac{31}{2}$ , 1}, Frame → True, PlotRange → All]] & /@
    Transpose[bounds];
```



```
In[29]:= TableForm[Sort[Tally[#, #1[[1]] < #2[[1]] &] &/@Transpose[bounds]]]
Print[ListPlot[#, Frame → True, PlotRange → All]] &/@%;
```

Out[29]:= TableForm=

-5	-1	0	1	2	3	4	5	6	7	8	10
2	1	21400	246	155	128	55	35	9	8	2	1
-2	-1	0	1	2	3	4	5	6	7	8	9
149	21225	475	54	20	26	47	6	15	13	10	3
-3	-1	1	2	3	4	5	6	7	8	9	10
1	1	3	4	24	50	247	43	38	56	174	395
-1	0	1	2	3	4	5	6	7	8	9	10
3	3	7	4	13	5	41	102	56	155	419	21232



## ■ Indices of bounding box position outliers

```
In[31]:= outliers = Flatten[Position[bounds, {l_, b_, r_, t_} /; l < 0 || b < -1 || r > 11 || t > 11]]
```

```
Out[31]= {1, 13, 64, 93, 101, 116, 134, 166, 236, 237, 240, 241, 247, 250, 254, 255, 256, 263, 265, 281, 284, 295, 307, 310, 311, 313, 316, 327, 328, 329, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 646, 658, 659, 661, 668, 675, 1033, 2300, 3362, 5390, 6077, 6760, 6787, 8423, 9419, 13735, 13789, 14038, 15261, 15906, 16400, 17659, 17900, 20505, 21427, 21576, 21590, 21706, 21829, 21831, 21840, 21858, 21877, 21891, 21942, 21950, 21953, 21959, 21960, 21968, 21971, 22040}
```

## ■ Extend glyph bitmaps to full font bounding box dimensions

```
In[32]:= fills = # - bigbox & /@ bounds;
```

```
In[33]:= Take[fills, 16]
```

```
Out[33]= {{18, 0, 0, -12}, {7, 2, -11, -2}, {6, 9, -10, -2}, {5, 2, -9, -2}, {5, 1, -9, -1}, {5, 2, -8, -3}, {5, 2, -9, -2}, {7, 9, -11, -2}, {7, 1, -9, -1}, {6, 1, -10, -1}, {5, 4, -9, -4}, {5, 4, -9, -4}, {5, 0, -12, -10}, {5, 6, -9, -6}, {7, 2, -11, -9}, {6, 2, -10, -2}}
```

```
In[34]:= Dimensions[bitmaps = Function[{line},
```

```
Block[{bounds = ToExpression[Drop[Flatten[Select[line, First[#] == "BBX" &]], 1]]},
```

```
1 - Take[HexExpand[#], bounds[[1]]] & /@ Flatten[Take[line,
```

```
{1, -1} + Flatten[Position[line, #] & /@ {"BITMAP", "ENDCHAR"}]]]]
```

```
]
] /@ charlines]
```

```
Out[34]= {22 043}
```

```
In[35]:= Take[bitmaps, 16]

Out[35]= {{{1}}, {{0}, {0}, {0}, {0}, {0}, {1}, {0}, {0}}, {{0, 1, 0}, {0, 1, 0}}, {{1, 0, 1, 0, 1}, {1, 0, 1, 0, 1}, {0, 0, 0, 0, 0}, {1, 0, 1, 0, 1}, {1, 0, 1, 0, 1}, {1, 0, 1, 0, 1}, {0, 0, 0, 0, 0}, {1, 0, 1, 0, 1}, {1, 0, 1, 0, 1}}, {{1, 1, 0, 1, 1}, {1, 0, 0, 0, 1}, {0, 1, 0, 1, 0}, {0, 1, 0, 1, 1}, {1, 0, 0, 1, 1}, {1, 1, 0, 1, 1}, {1, 0, 0, 0, 1}, {0, 1, 0, 1, 0}, {0, 1, 0, 1, 1}, {1, 0, 0, 1, 1}, {1, 1, 0, 1, 1}, {1, 0, 0, 0, 1}, {1, 1, 0, 0, 1}, {1, 1, 0, 1, 0}, {0, 1, 0, 1, 0}, {1, 0, 0, 0, 1}, {1, 1, 0, 1, 1}}, {{1, 0, 1, 1, 0, 1}, {0, 1, 0, 1, 0, 1}, {0, 1, 0, 0, 1, 1}, {1, 0, 1, 0, 1, 1}, {1, 1, 0, 1, 0, 1}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 1, 0, 1}, {0, 1, 1, 0, 1, 0}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1}, {0, 1, 1, 0, 1, 1}, {1, 0, 0, 1, 0, 1}, {1, 0, 1, 1, 1}, {1, 0, 0, 1, 1, 1}, {0, 1, 0, 1, 0, 0}, {0, 1, 1, 0, 1, 0}, {1, 0, 0, 1, 0, 0}, {{0}, {0}}, {{1, 1, 0}, {1, 0, 1}, {0, 1, 1}, {0, 1, 1}, {0, 1, 1}, {0, 1, 1}, {0, 1, 1}, {0, 1, 1}, {1, 0, 1, 1, 0, 1}, {1, 1, 0, 1, 0, 1}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 1}, {0, 1, 1, 0, 1, 0}, {0, 1, 1, 0, 1, 1}, {1, 0, 0, 1, 0, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 0}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 1, 0, 1, 0, 1}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 1}, {0, 1, 1, 0, 1, 0}, {0, 1, 1, 0, 1, 1}, {1, 0, 0, 1, 0, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 0}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 1, 0, 1, 0, 1}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 1}, {0, 1, 1, 0, 1, 0}, {0, 1, 1, 0, 1, 1}, {1, 0, 0, 1, 0, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 0}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 1, 0, 1, 0, 1}, {1, 0, 1, 0, 1, 0}, {1, 0, 1, 0, 1, 1}, {1, 0, 1, 1, 0, 1}, {1, 0, 0, 1, 1, 1}}}

In[36]:= BitmapExtend = Function[{bitmap, fill},
  Block[{bmp, xl = Table[1, {0, fill[[1]]}], xr = Table[1, {0, -fill[[3]]}], yt, yb},
    bmp = Join[xl, #, xr] & /@ bitmap;
    yb = Table[Table[1, {0, Length[First[bmp]]}], {0, fill[[2]]}];
    yt = Table[Table[1, {0, Length[First[bmp]]}], {0, -fill[[4]]}];
    Join[yt, bmp, yb]
  ]
]

Out[36]= Function[{bitmap, fill},
  Block[{bmp, xl = Table[1, {0, fill[[1]]}], xr = Table[1, {0, -fill[[3]]}], yt, yb},
    bmp = (Join[xl, #1, xr] &) /@ bitmap;
    yb = Table[Table[1, {0, Length[First[bmp]]}], {0, fill[[2]]}];
    yt = Table[Table[1, {0, Length[First[bmp]]}], {0, -fill[[4]]}];
    Join[yt, bmp, yb]]]

In[37]:= Dimensions[extbitmaps = MapThread[BitmapExtend, {bitmaps, fills}]]]

Out[37]= {22 043, 13, 19}

In[38]:= bigbox

Out[38]= {-5, -2, 14, 11}

■ Truncate glyphs to 12×12 bitmap

In[39]:= BitmapTruncate = Function[bitmap, Take[#, {6, 17}] & /@ Take[bitmap, {2, 13}]]

Out[39]= Function[bitmap, (Take[#, {6, 17}] &) /@ Take[bitmap, {2, 13}]]

In[40]:= Dimensions[trbitmaps = BitmapTruncate /@ extbitmaps]

Out[40]= {22 043, 12, 12}
```

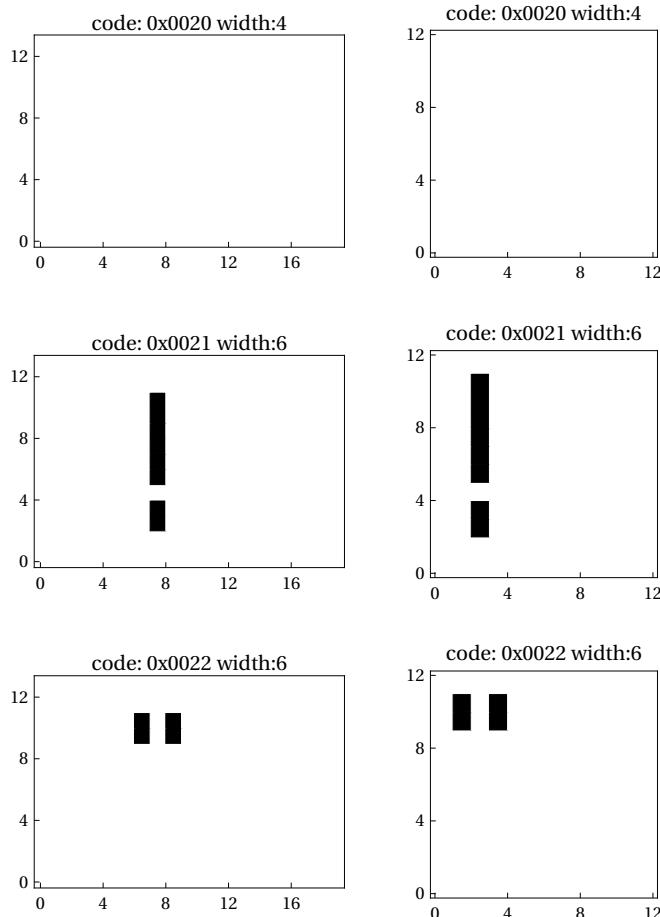
## Display glyphs

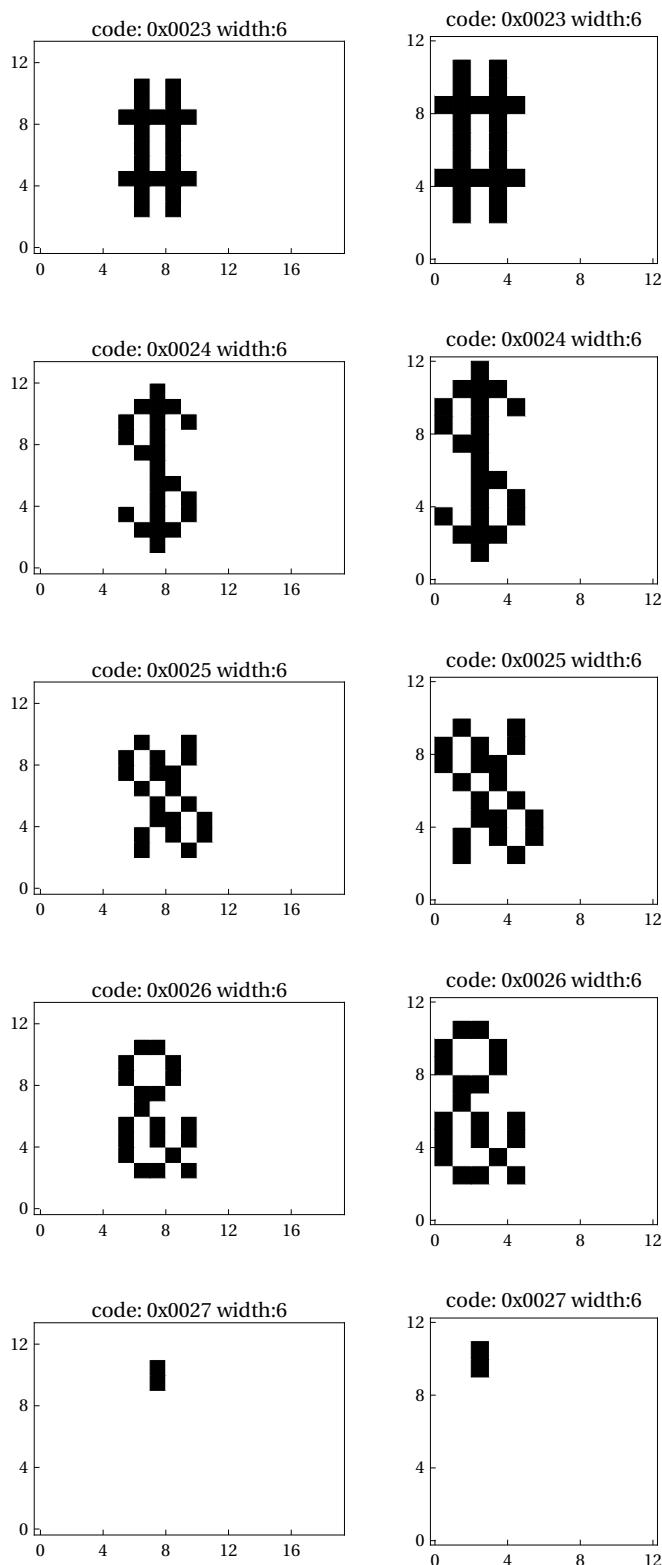
### ■ Non standard width glyphs

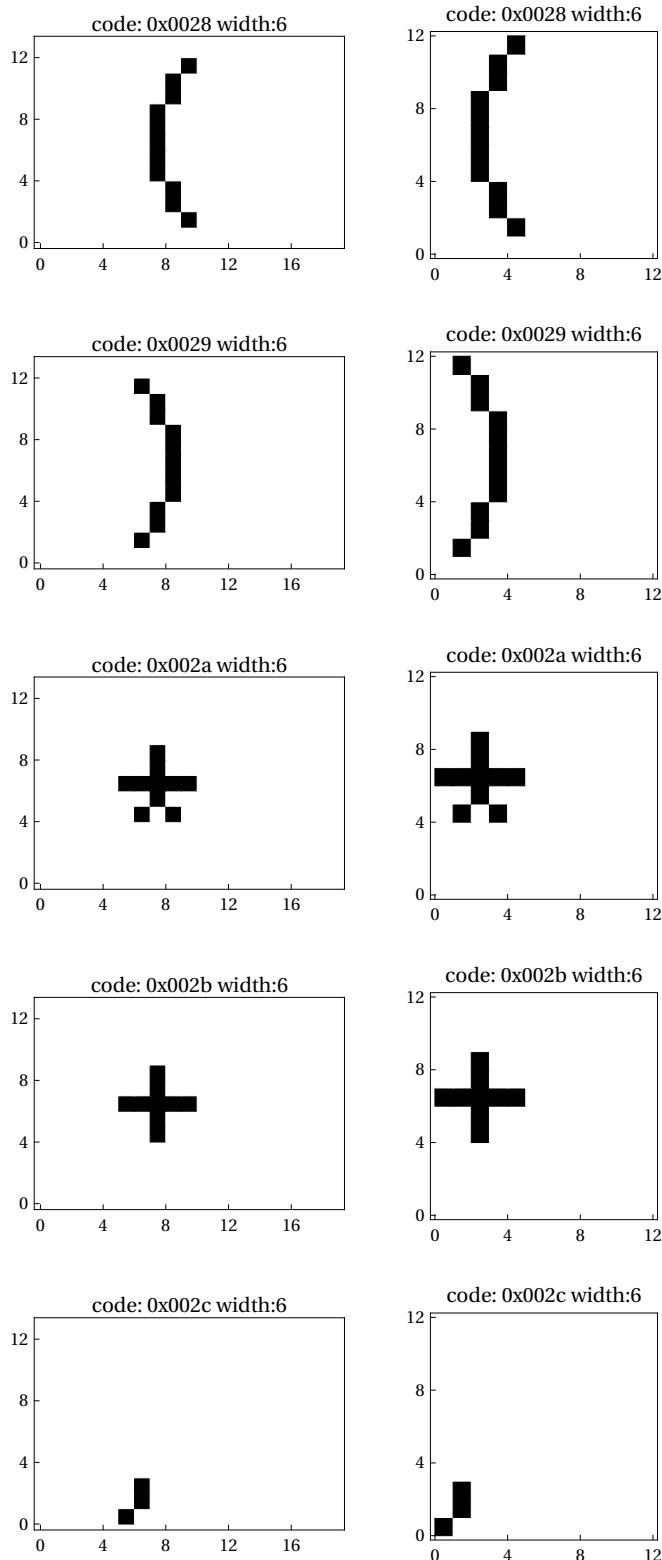
```
In[41]:= nonstdwidthgraphics =
MapThread[Graphics[Raster[Reverse[#1]], Frame → True, AspectRatio →
Divide @@ Dimensions[#1], FrameTicks → ({#, #, {}, {}} & [Range[0, 16, 4]]),
PlotLabel → ("code: " <> IntegerString[ToExpression[#2], 16, 4] <> " width:" <>
ToString[#3])] &, #[[nonstdwidth]] & /@ {extbitmaps, charcodes, widths}];

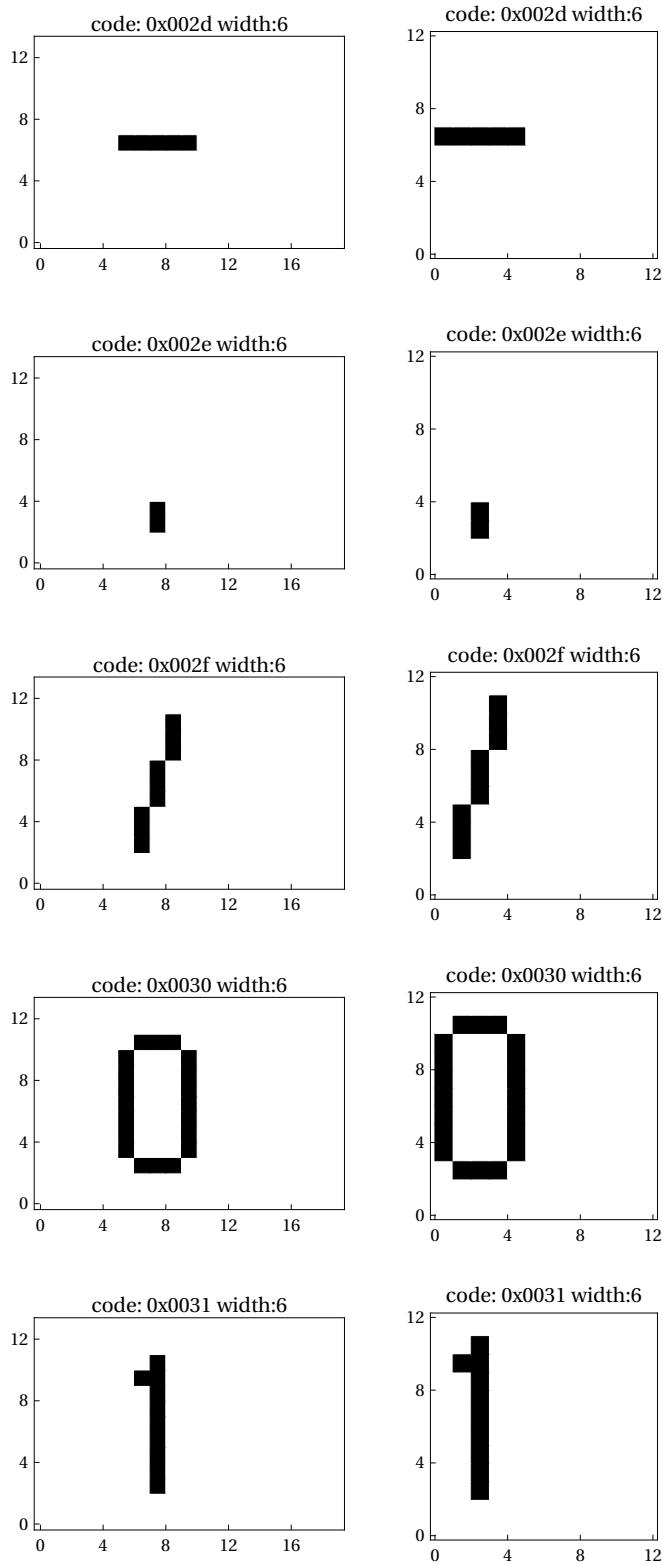
In[42]:= nonstdwidthtruncgraphics =
MapThread[Graphics[Raster[Reverse[#1]], Frame → True, AspectRatio →
Divide @@ Dimensions[#1], FrameTicks → ({#, #, {}, {}} & [Range[0, 16, 4]]),
PlotLabel → ("code: " <> IntegerString[ToExpression[#2], 16, 4] <> " width:" <>
ToString[#3])] &, #[[nonstdwidth]] & /@ {trbitmaps, charcodes, widths}];

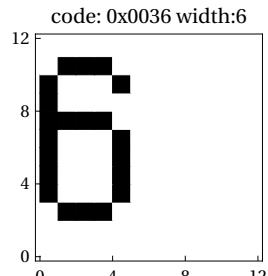
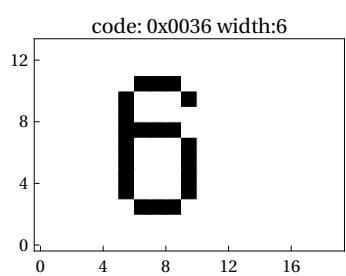
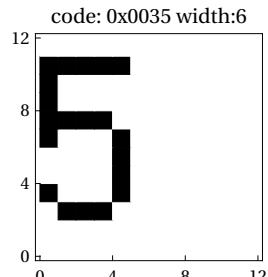
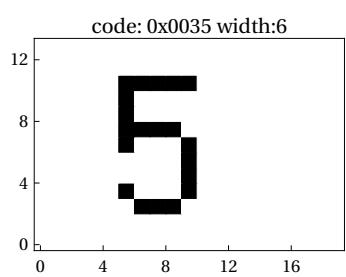
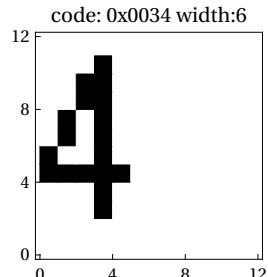
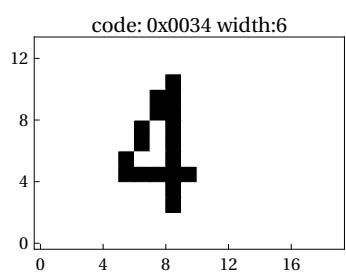
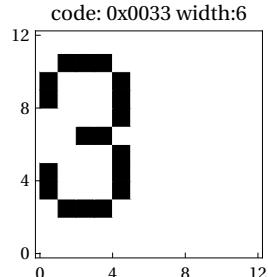
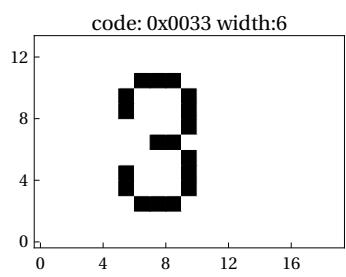
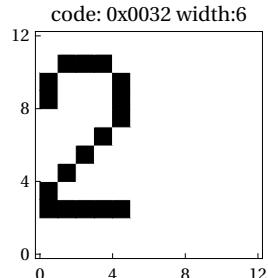
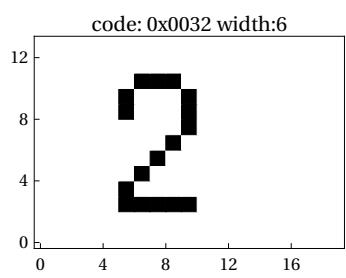
In[43]:= MapThread[Print[GraphicsGrid[{{##}}]]] &,
{nonstdwidthgraphics, nonstdwidthtruncgraphics}]
```

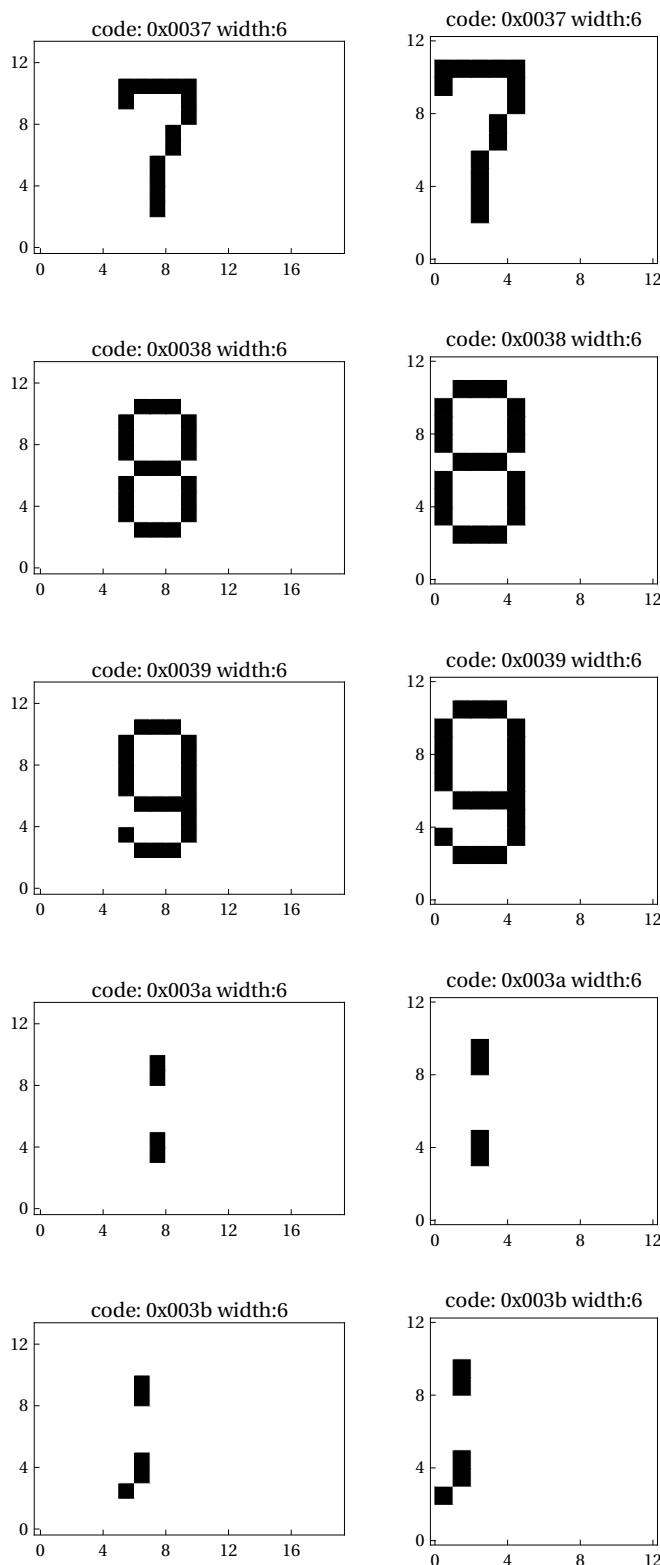


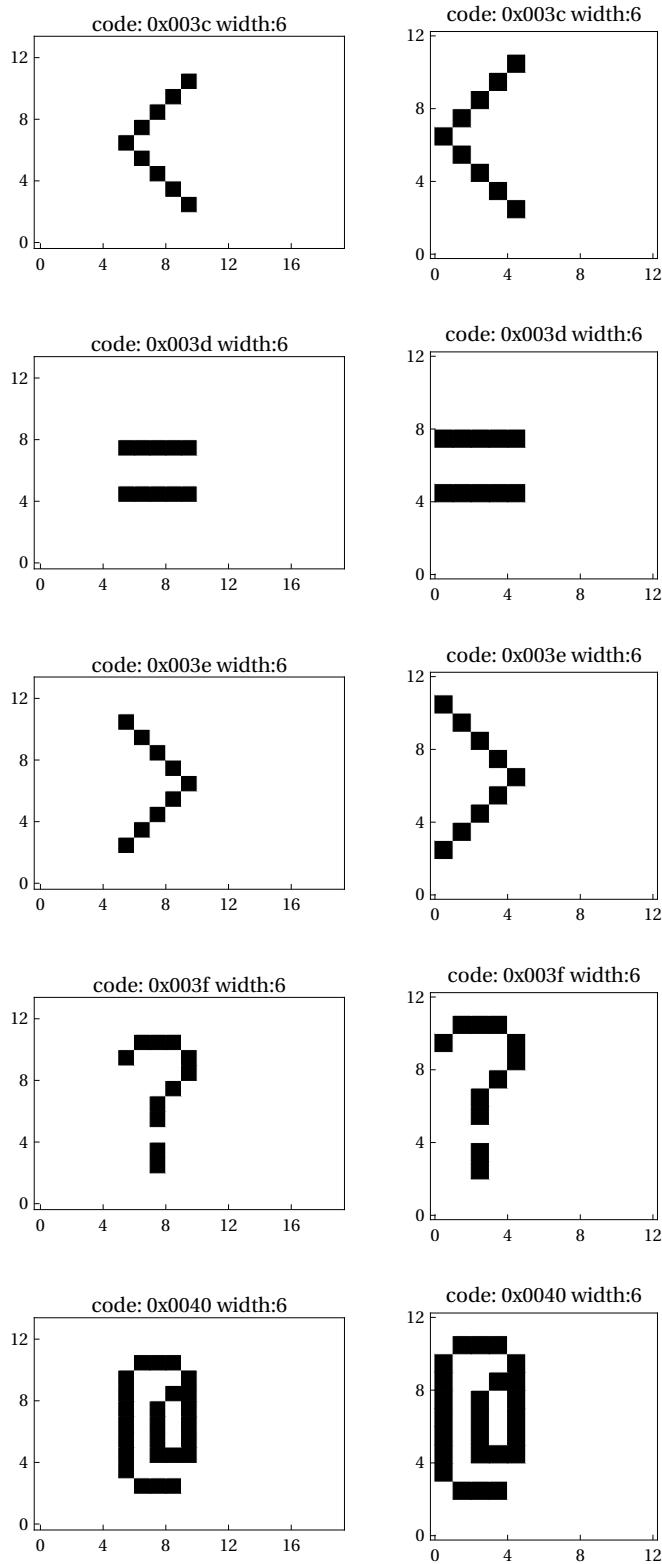


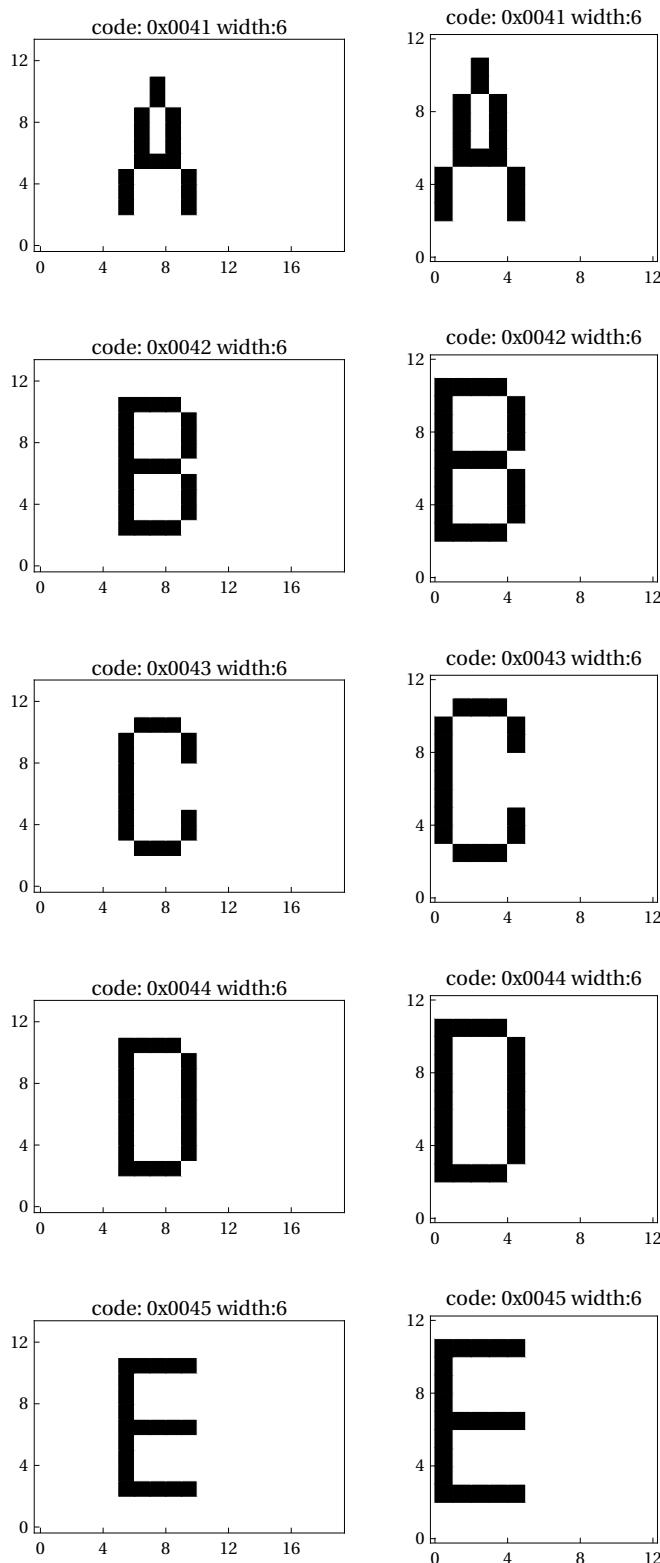


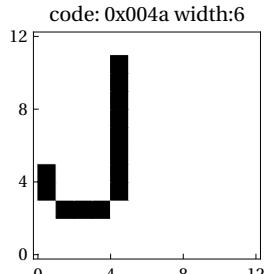
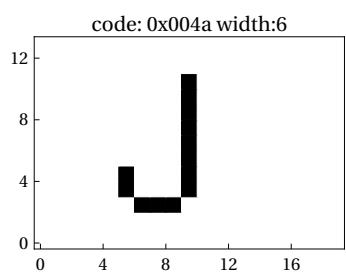
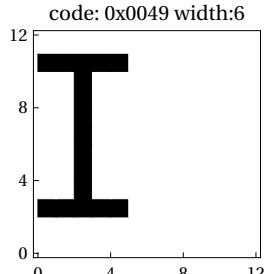
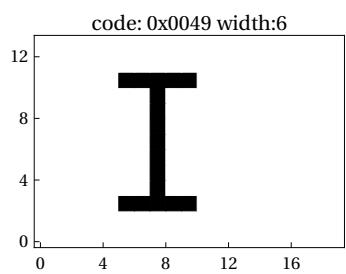
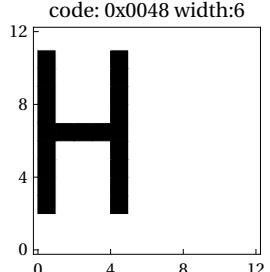
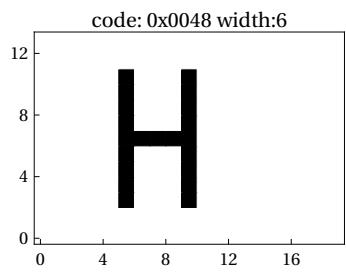
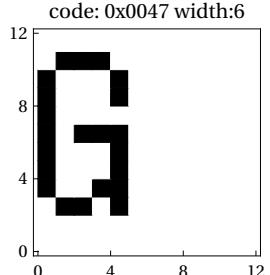
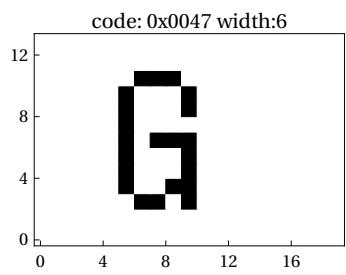
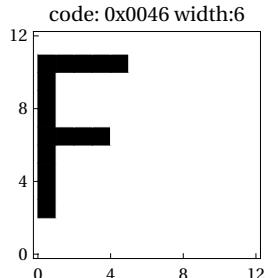
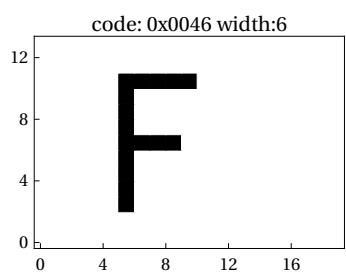


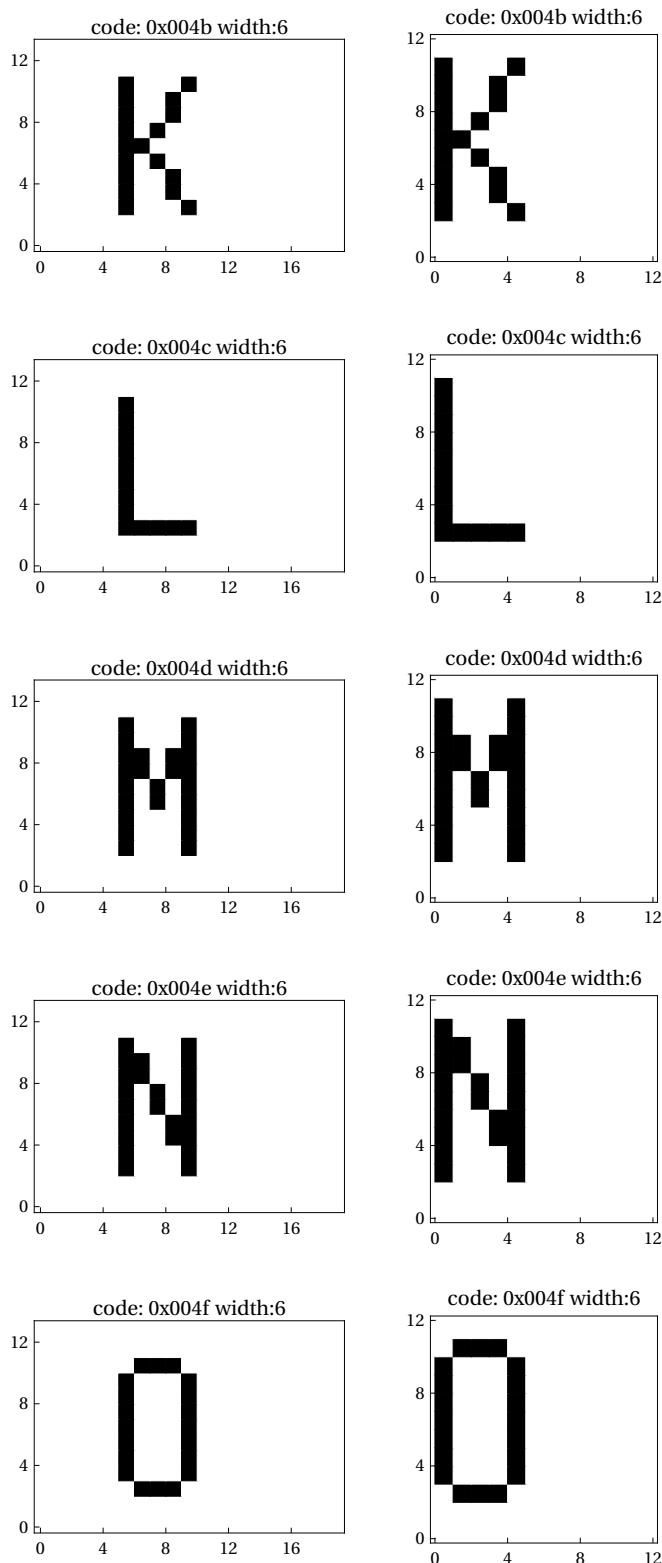


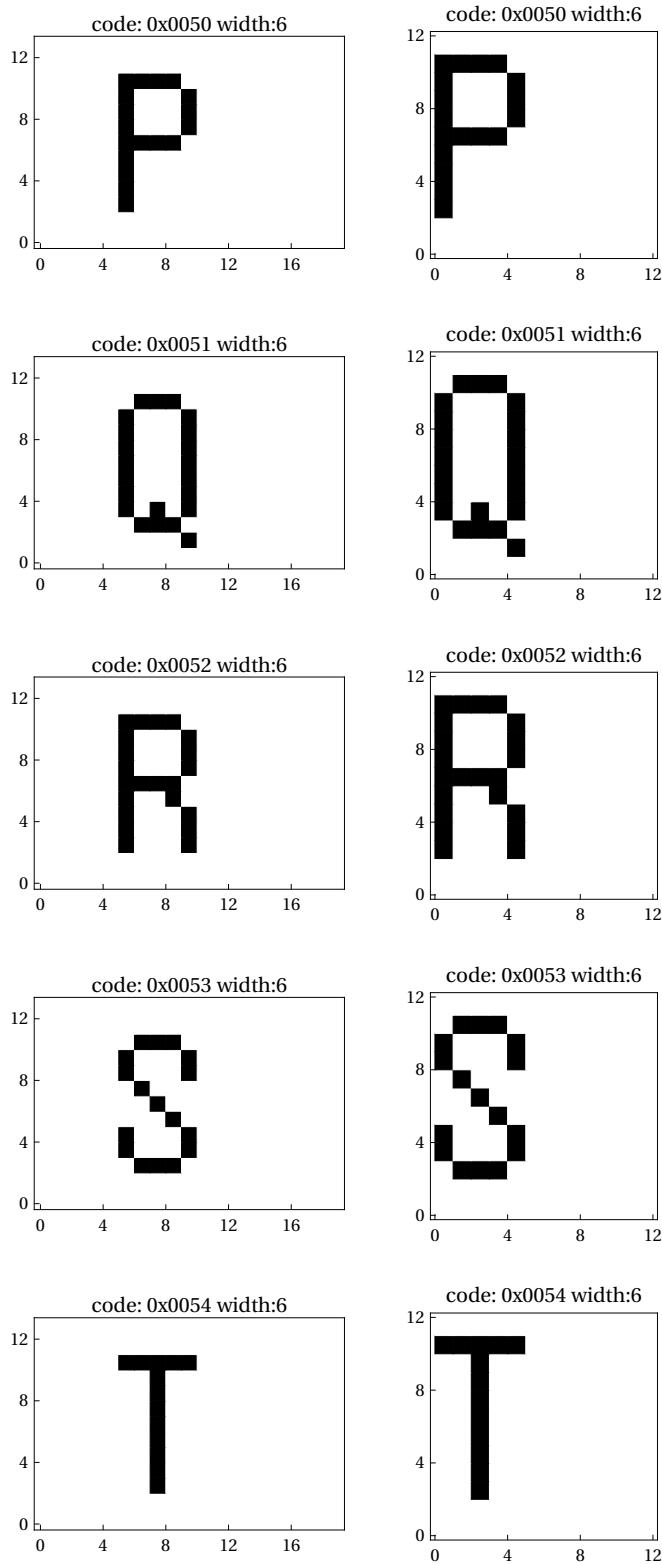


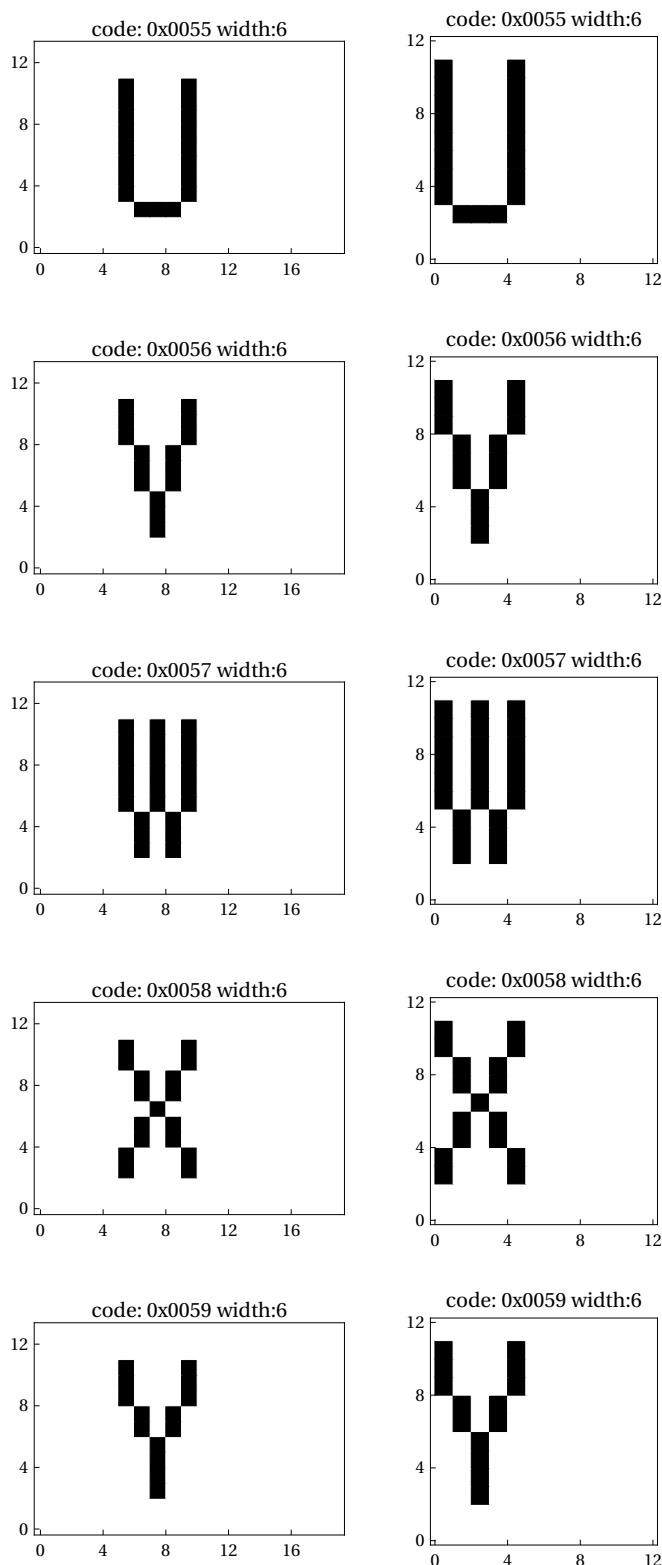


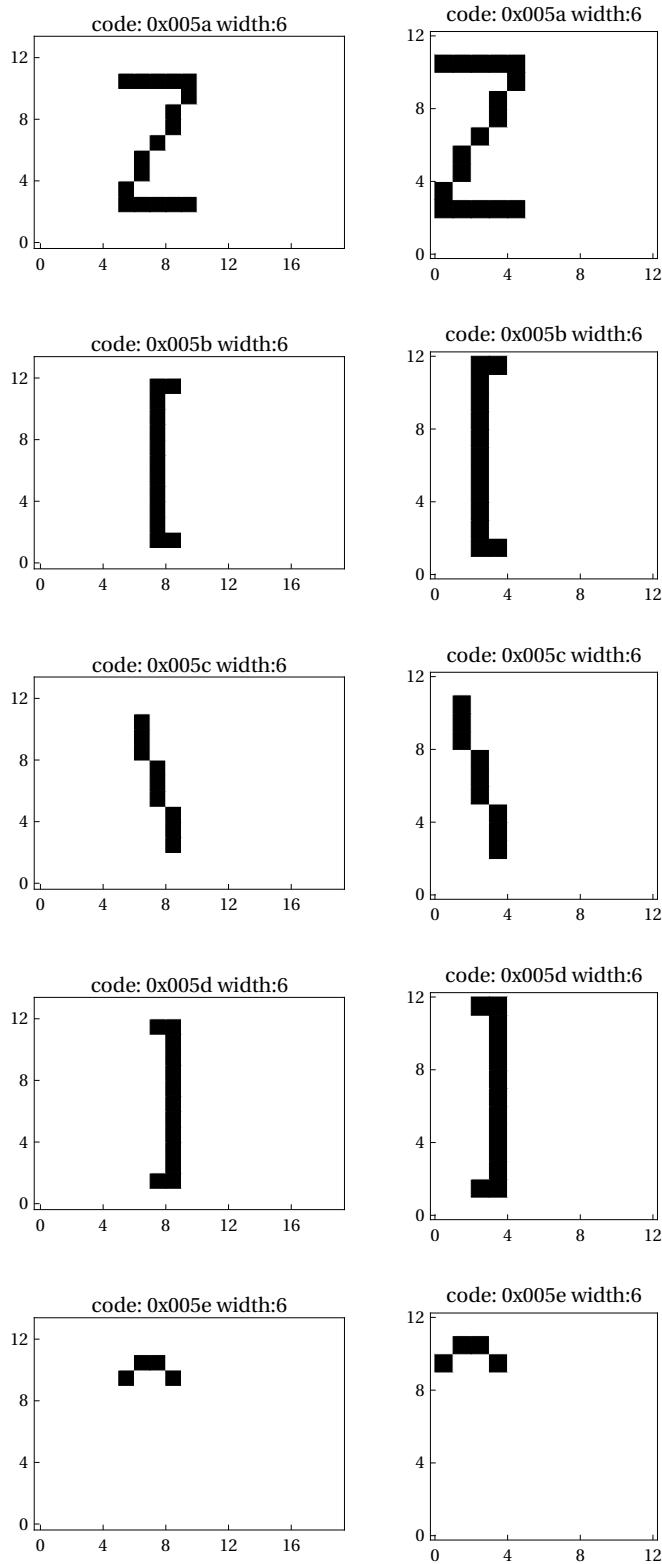


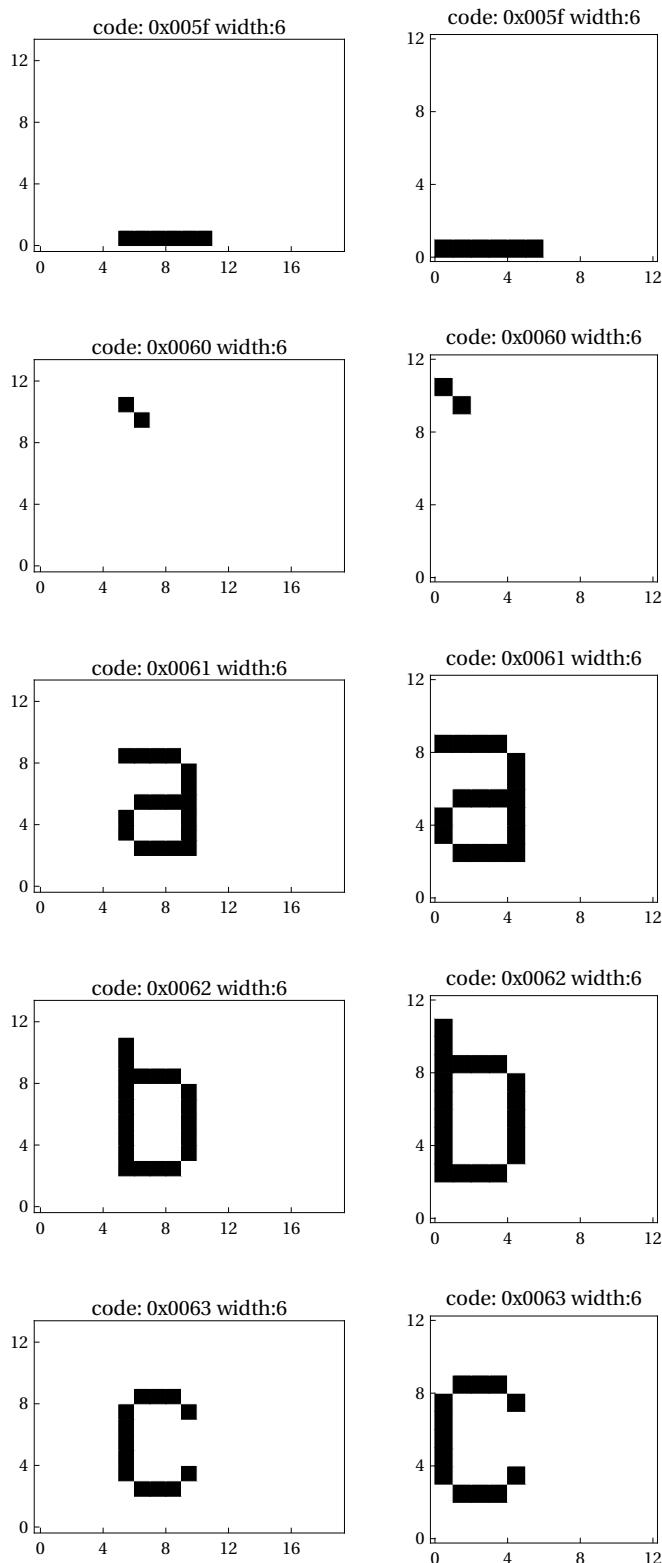


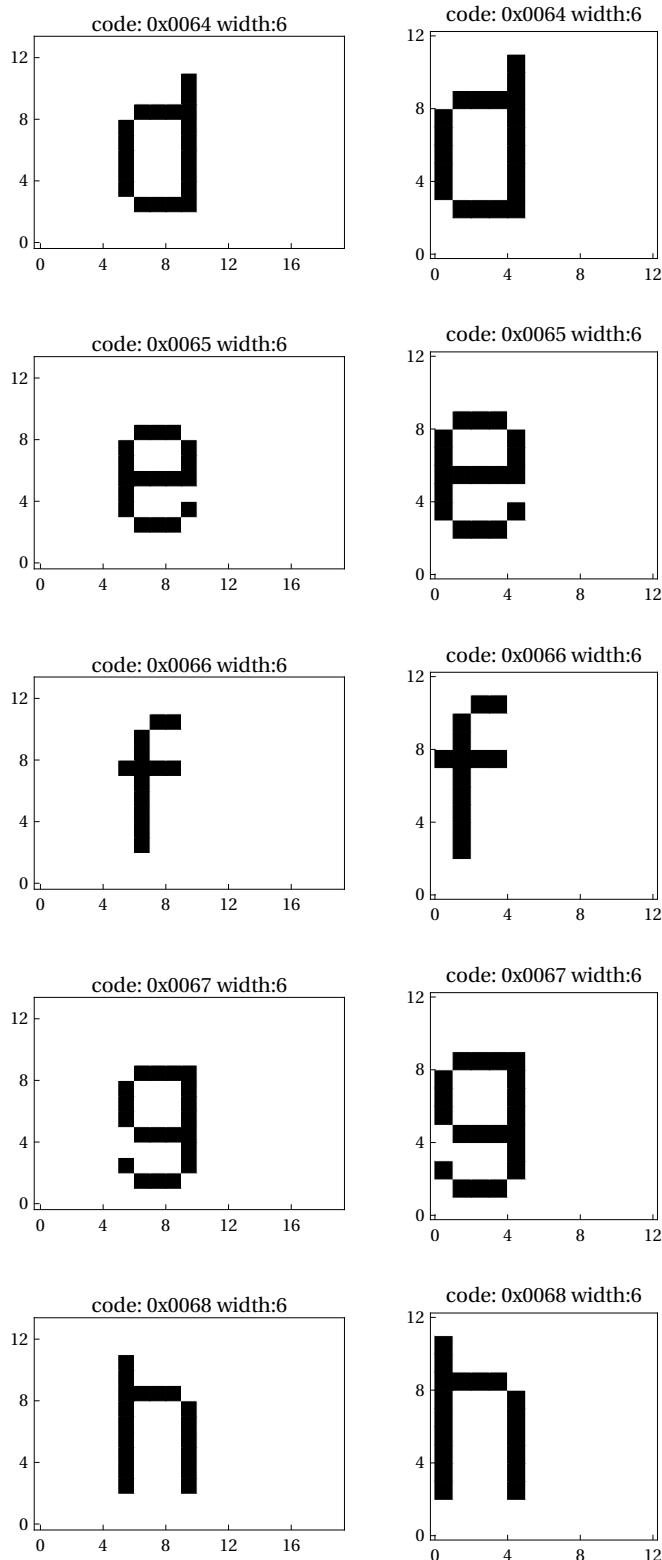


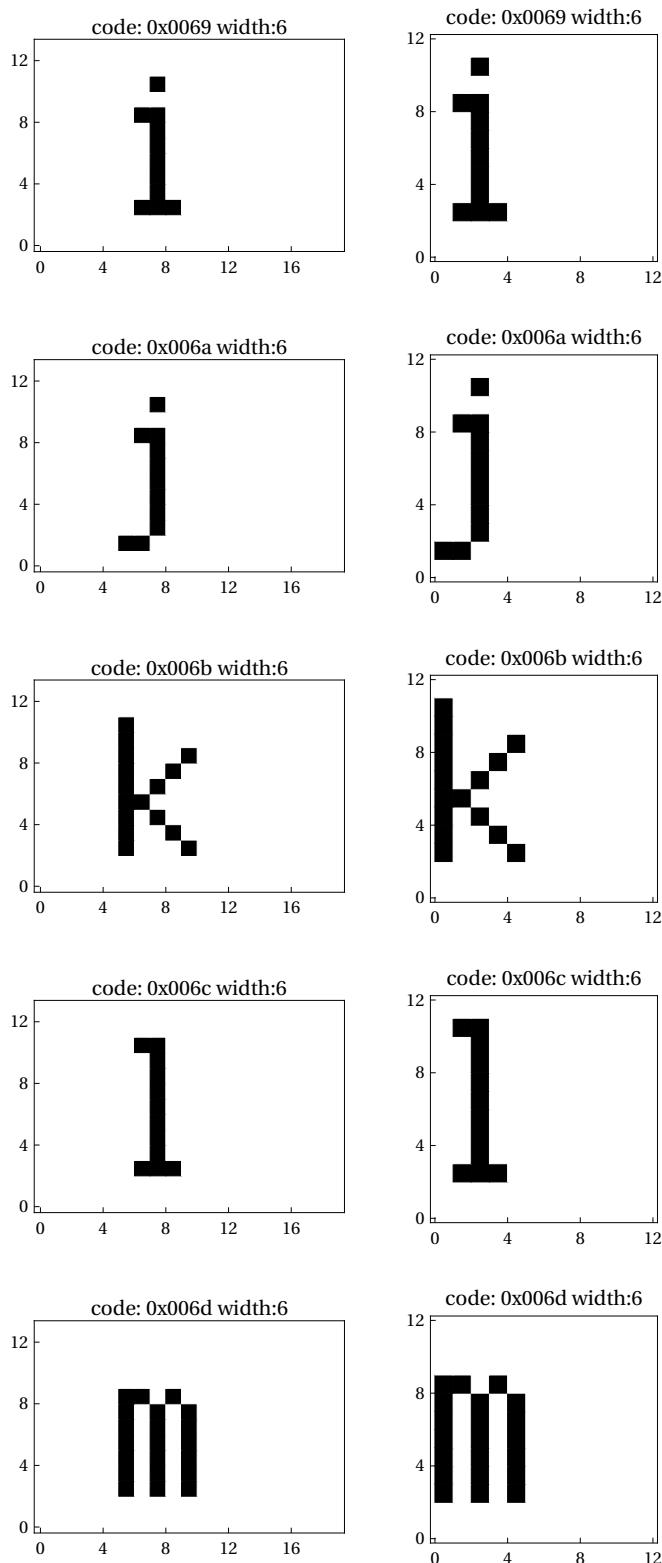


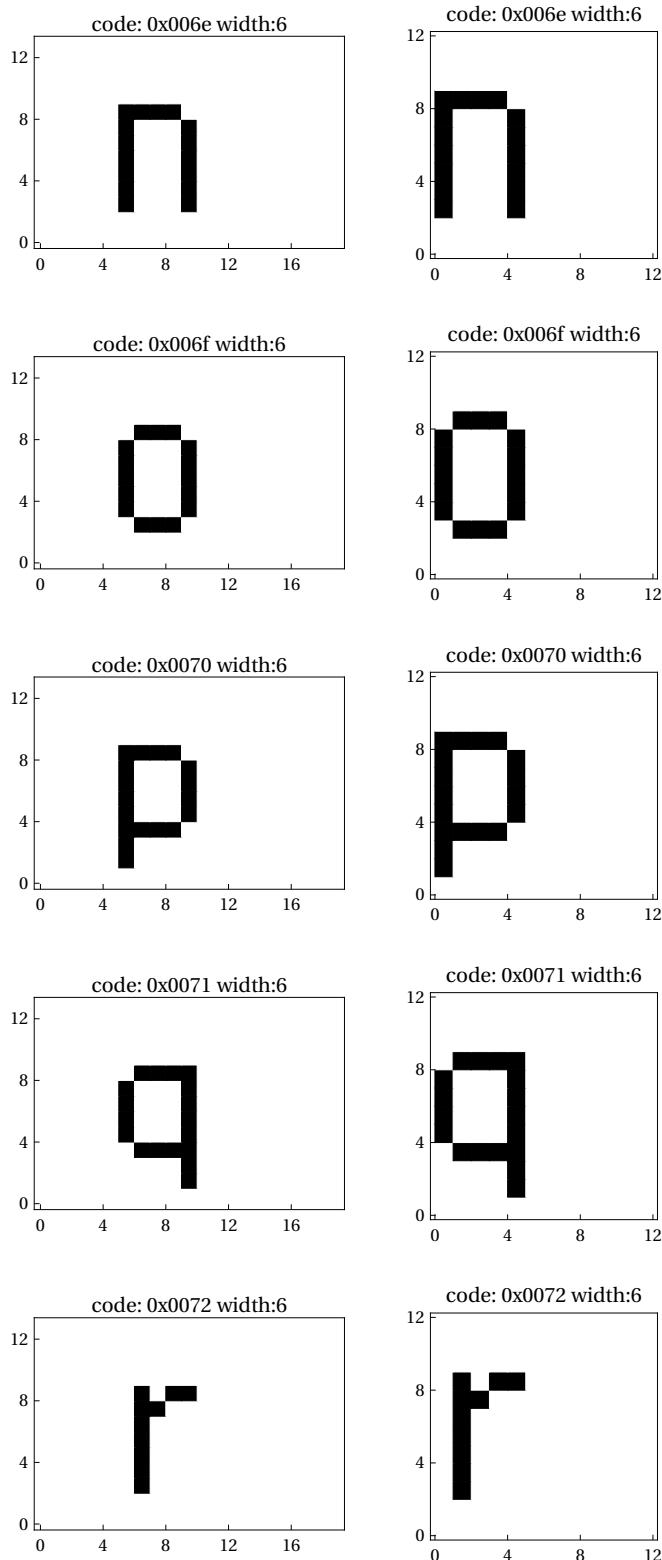


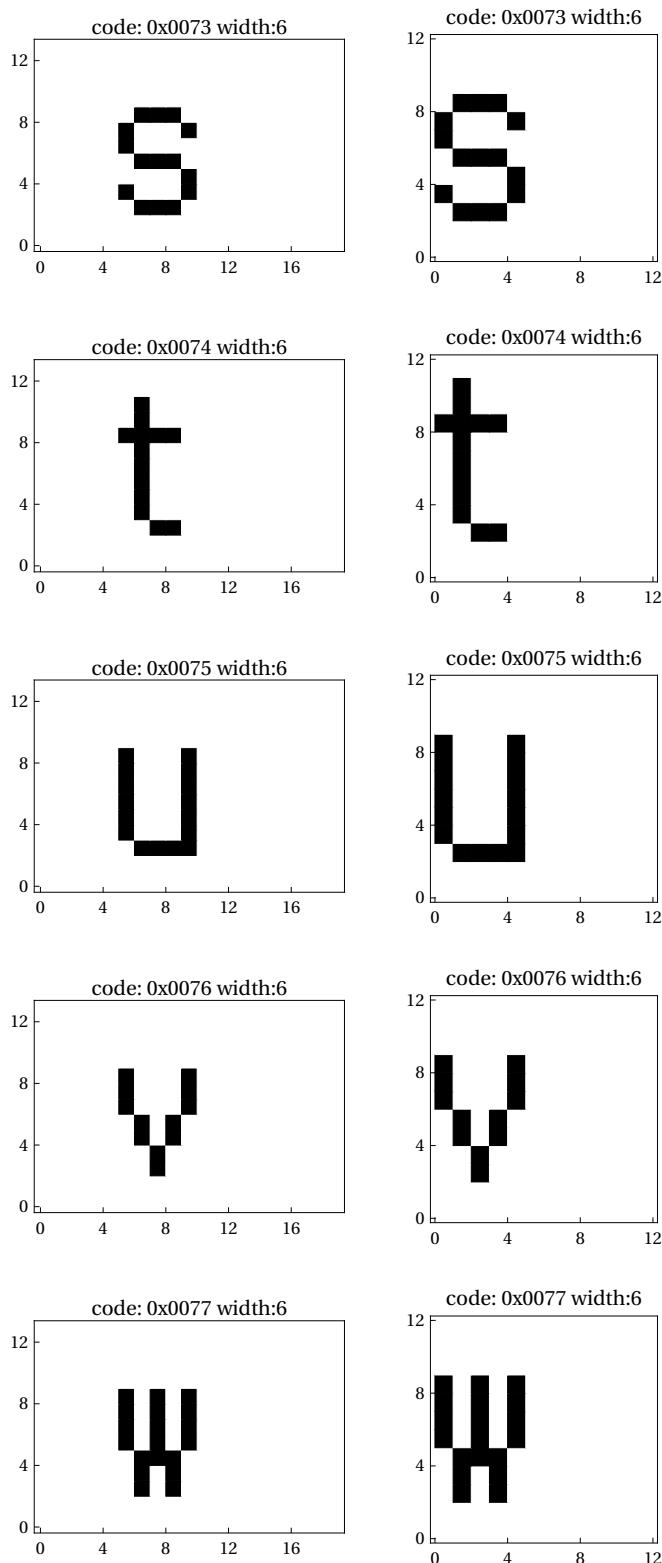


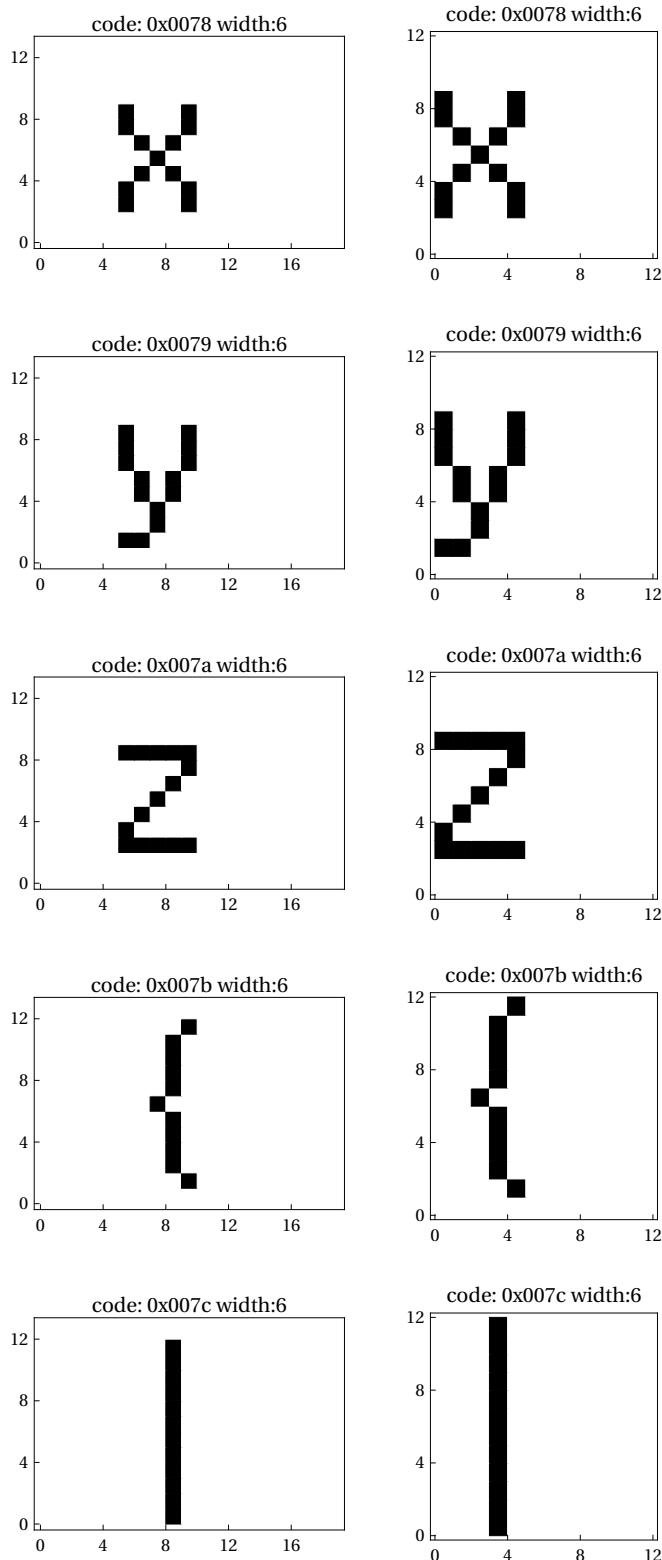


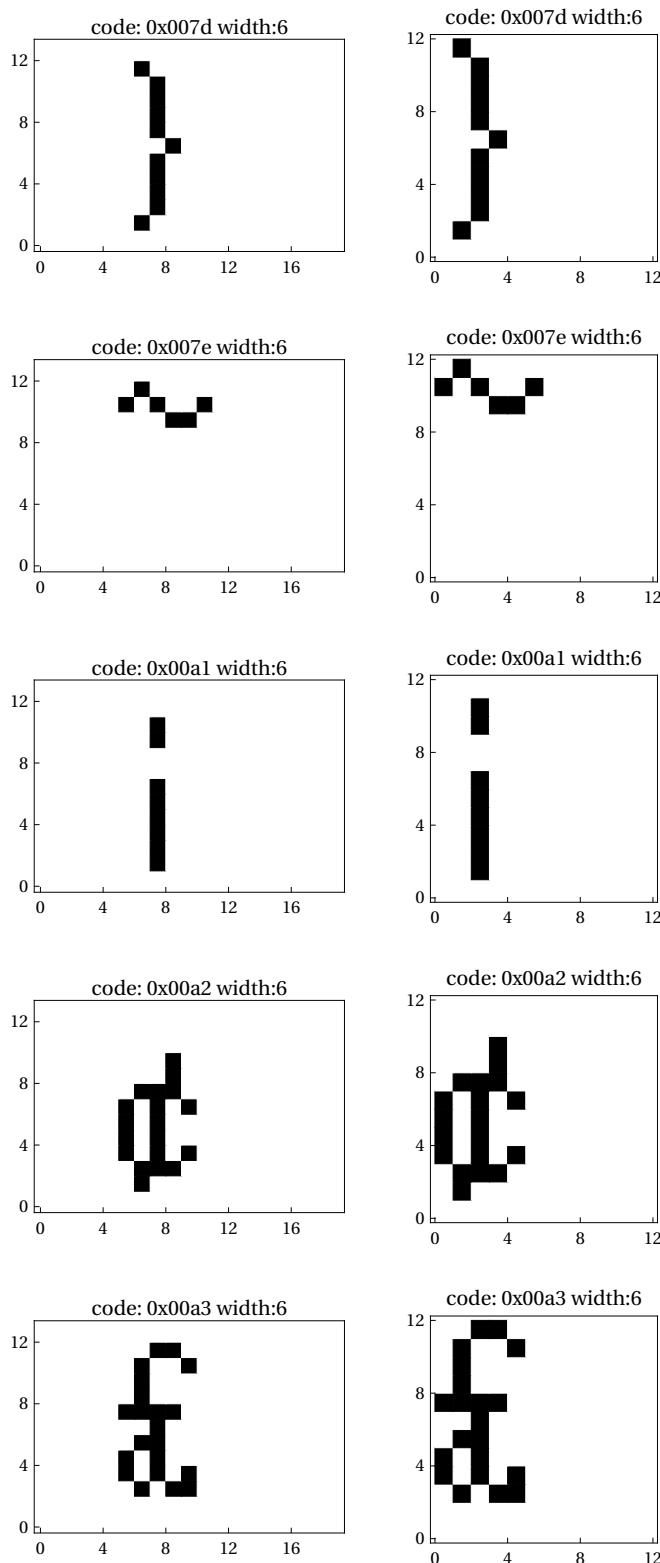


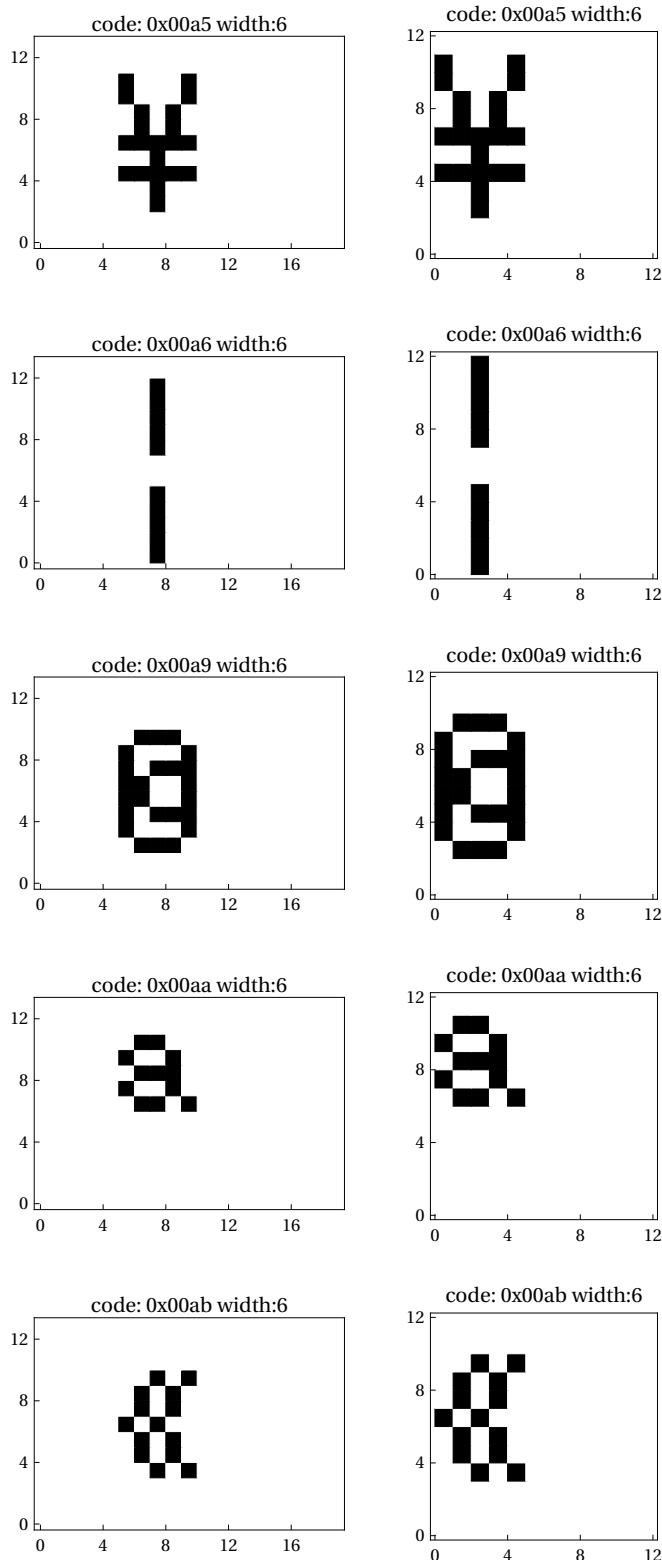


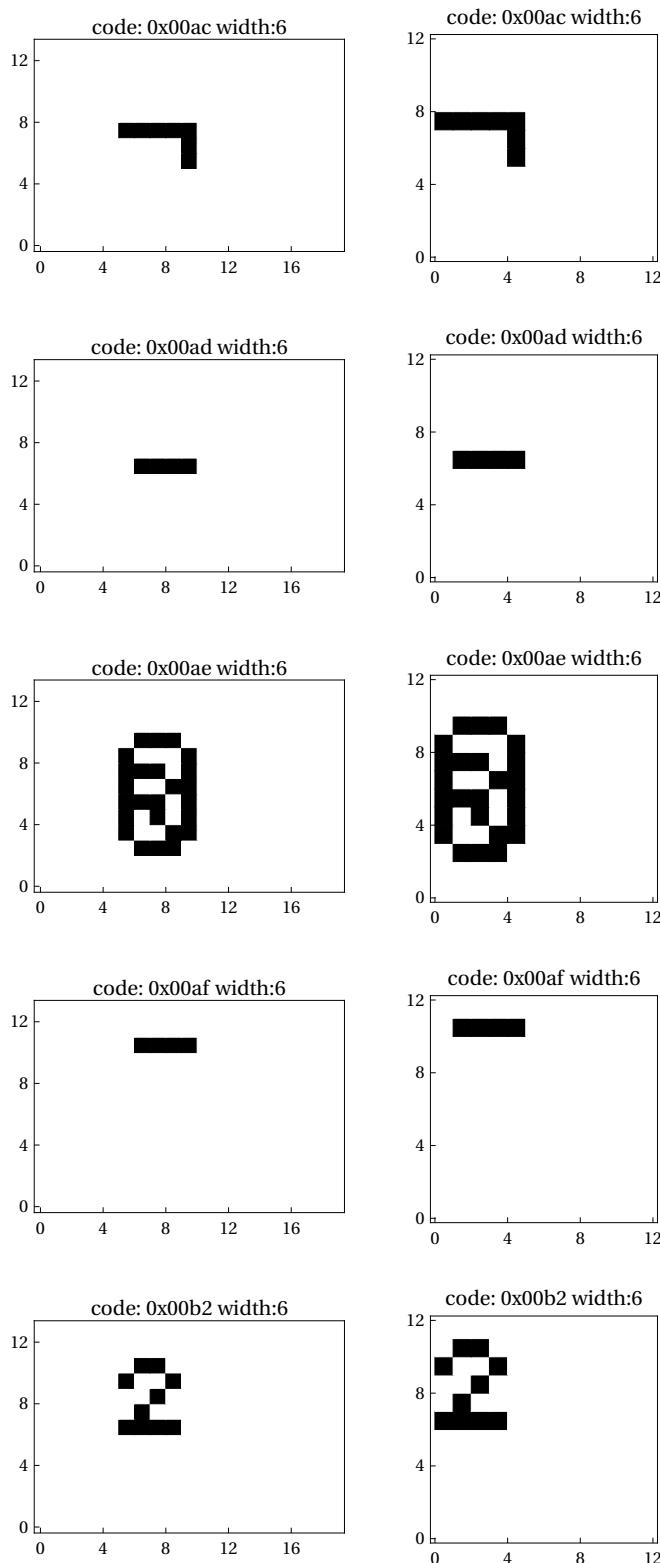


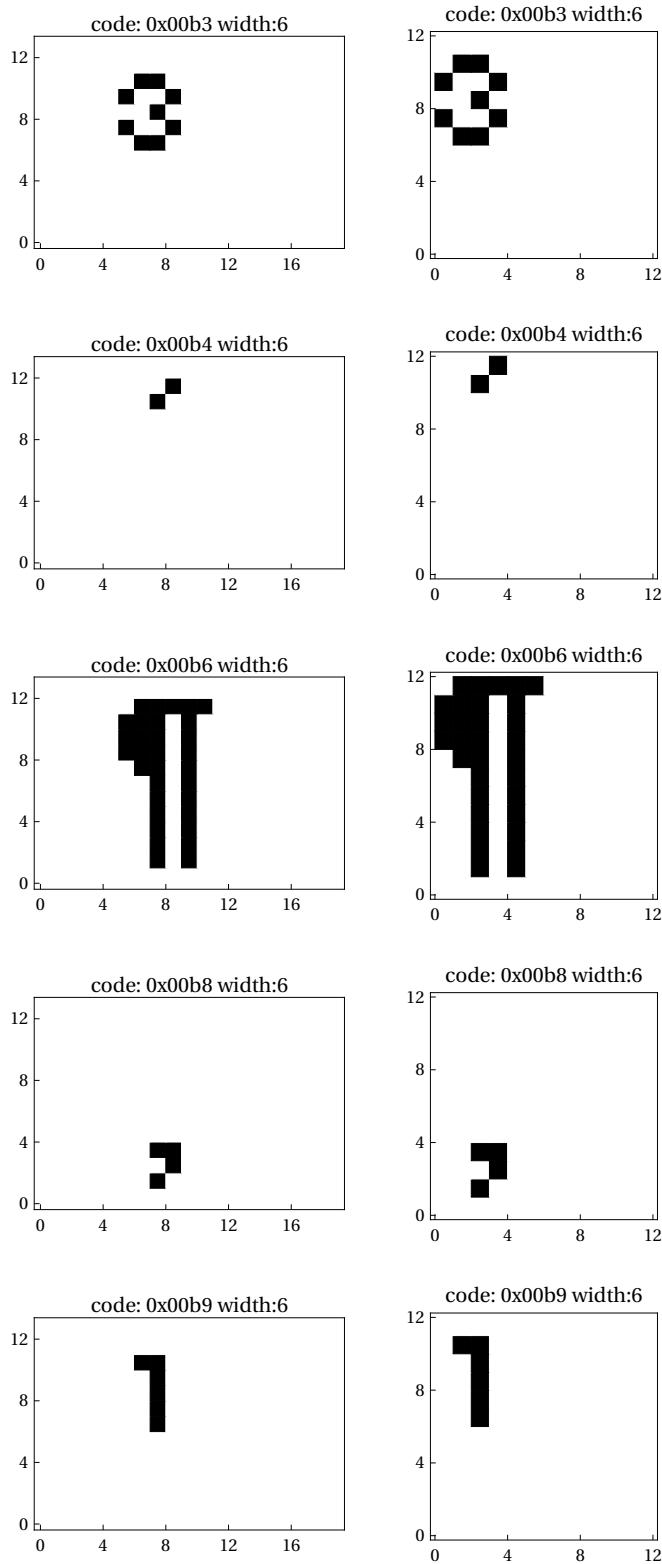


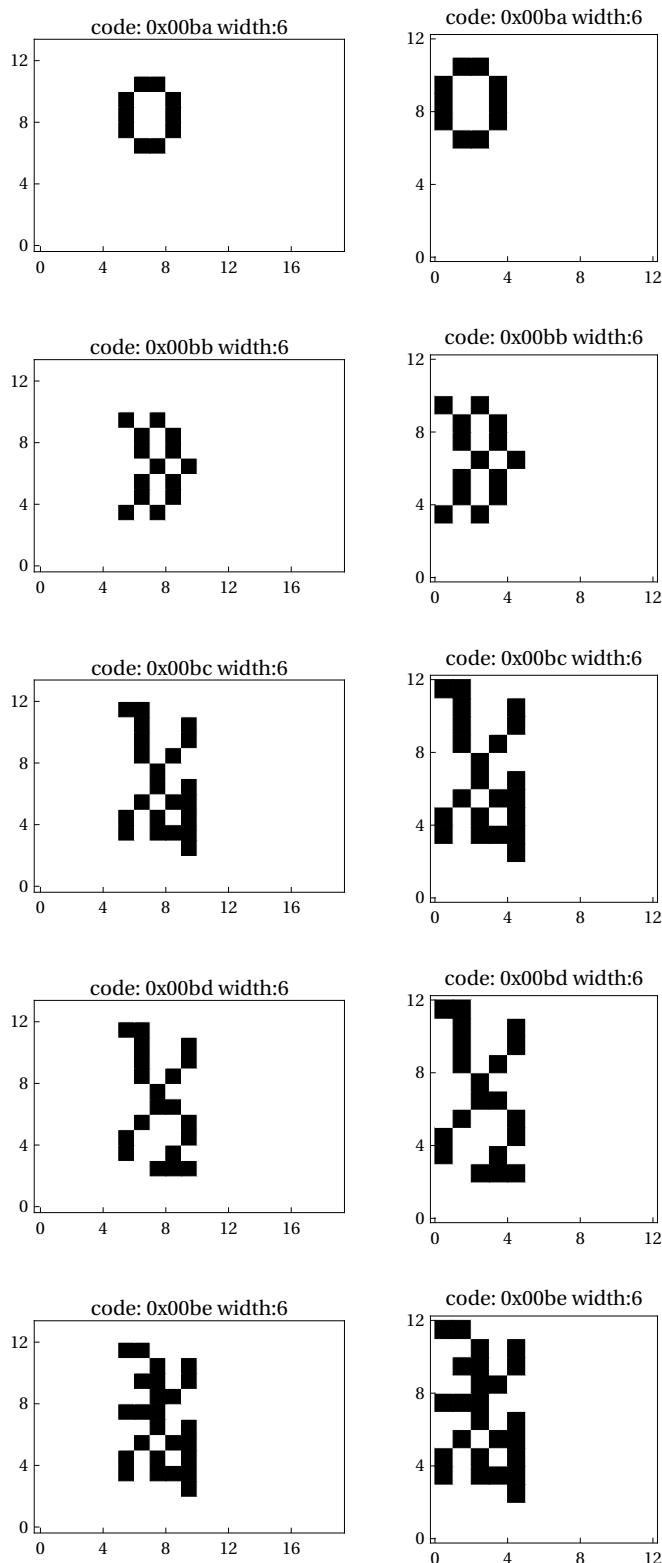


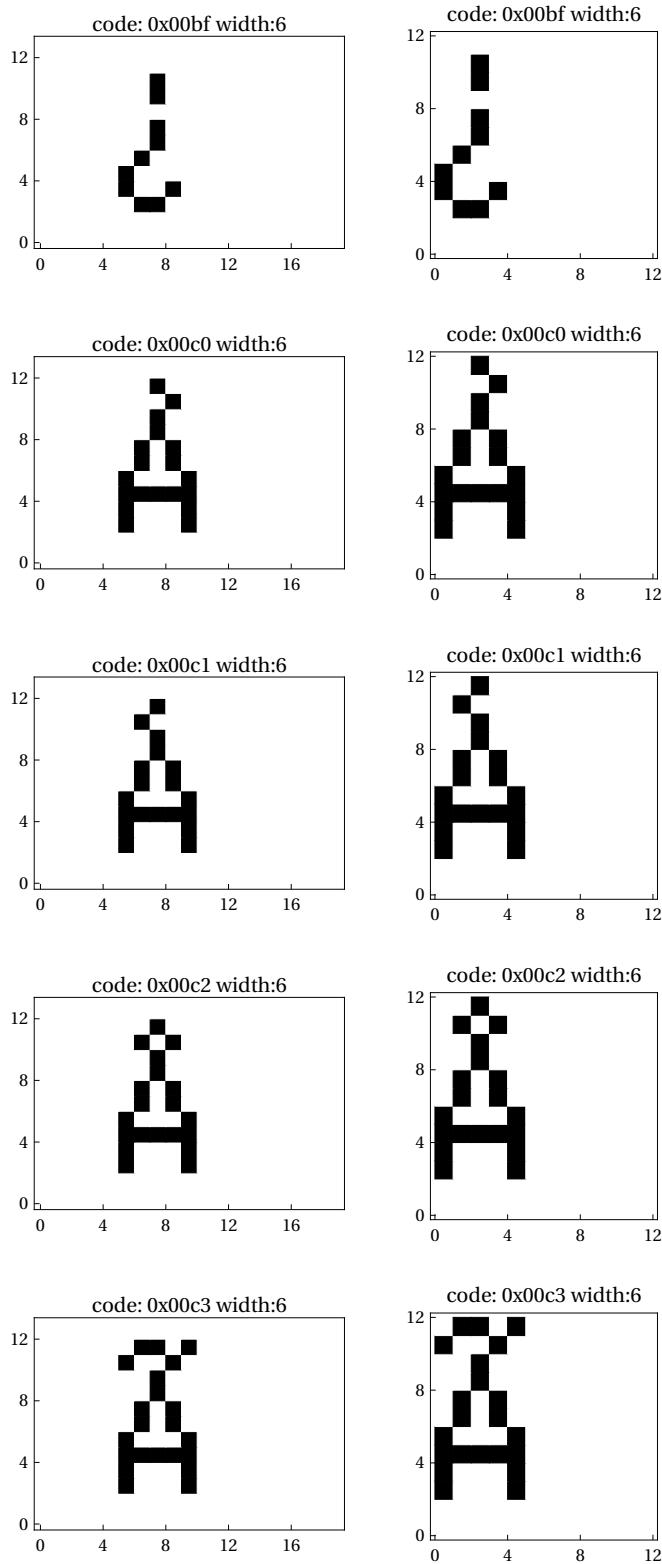


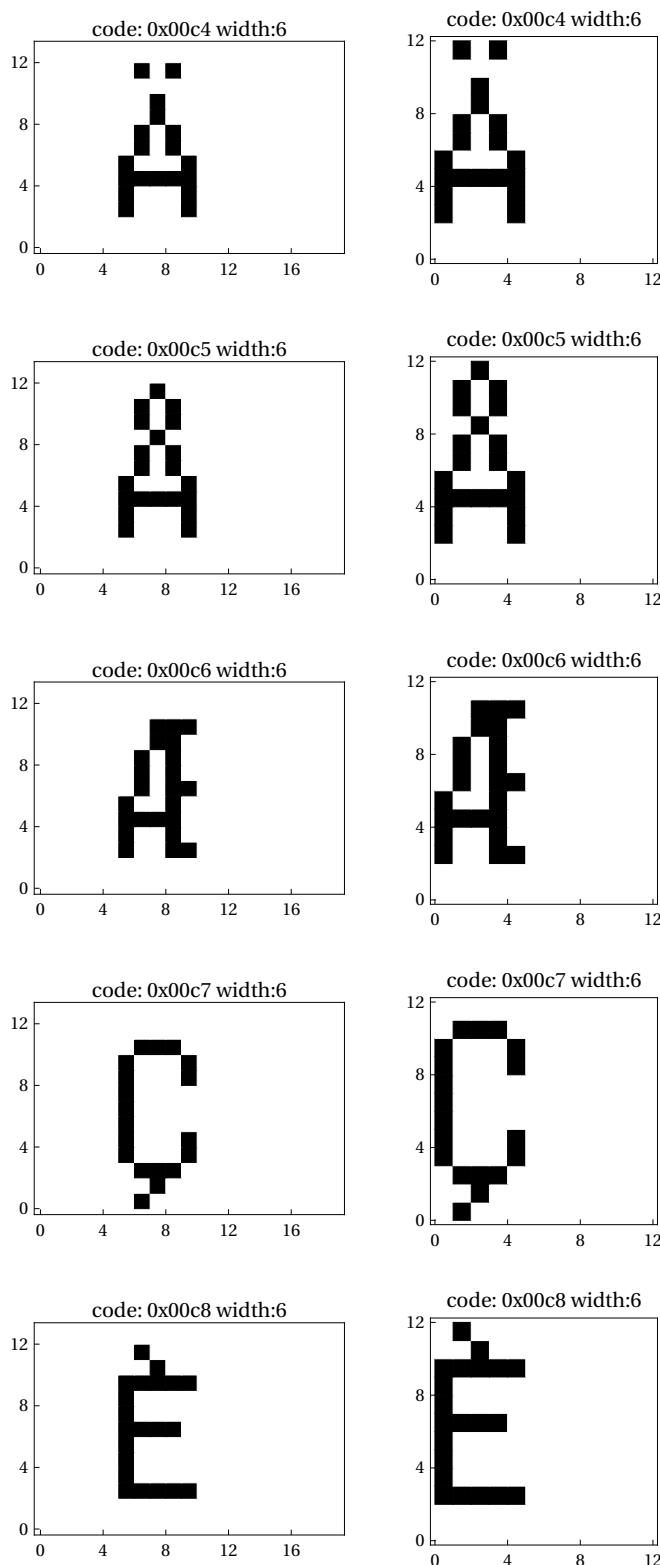


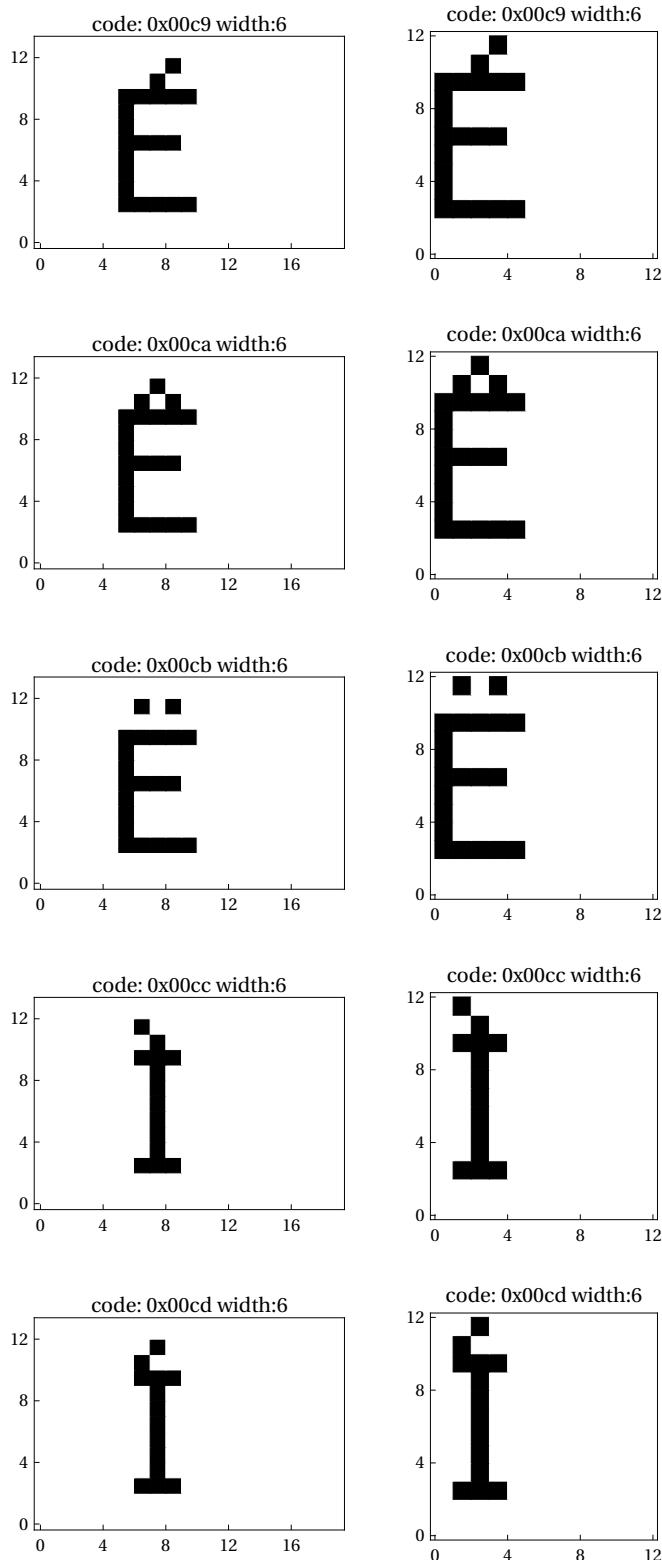


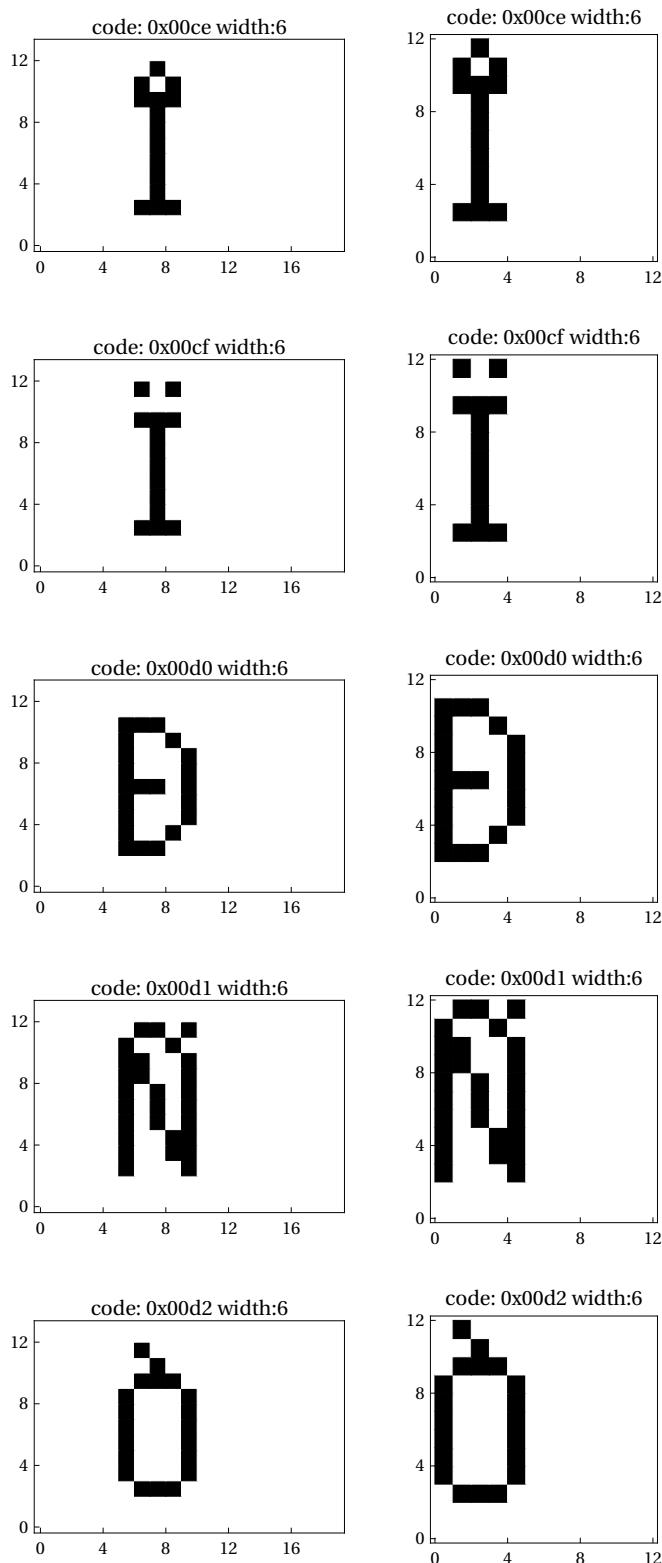


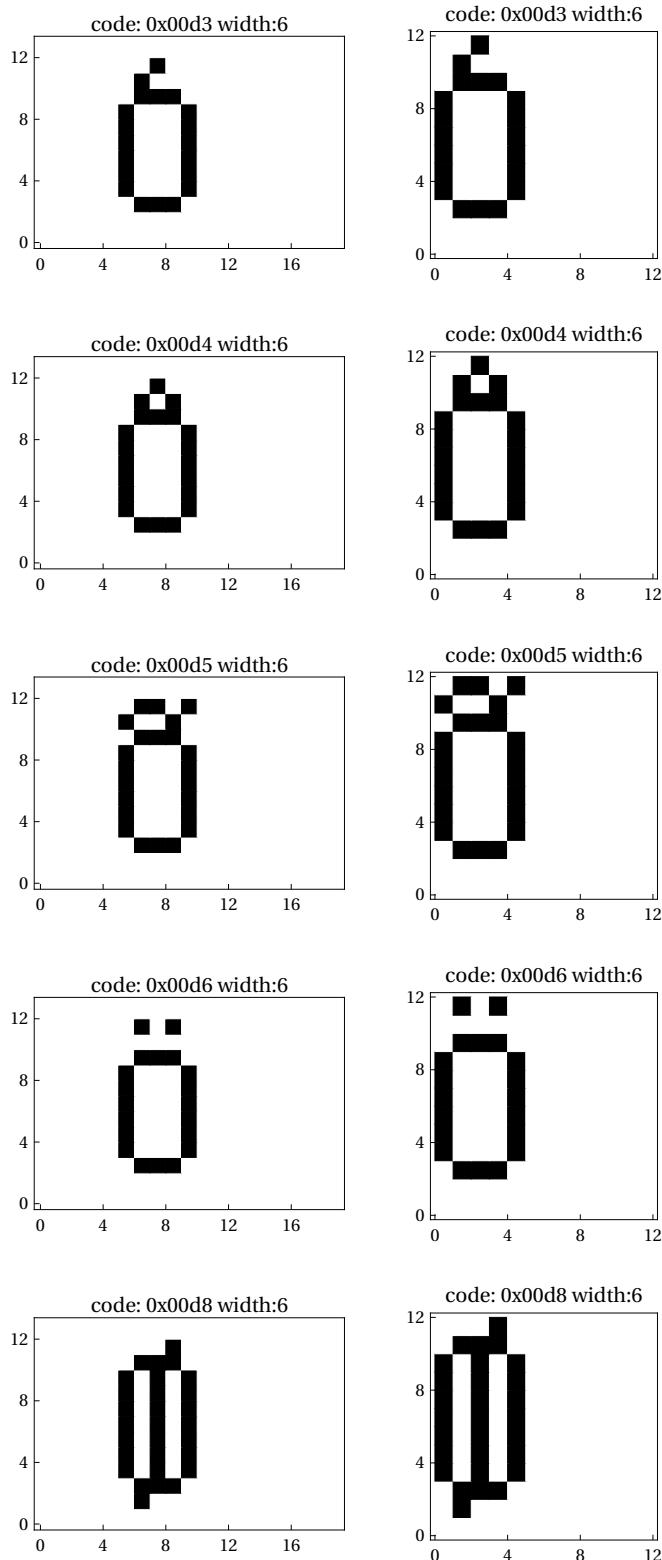


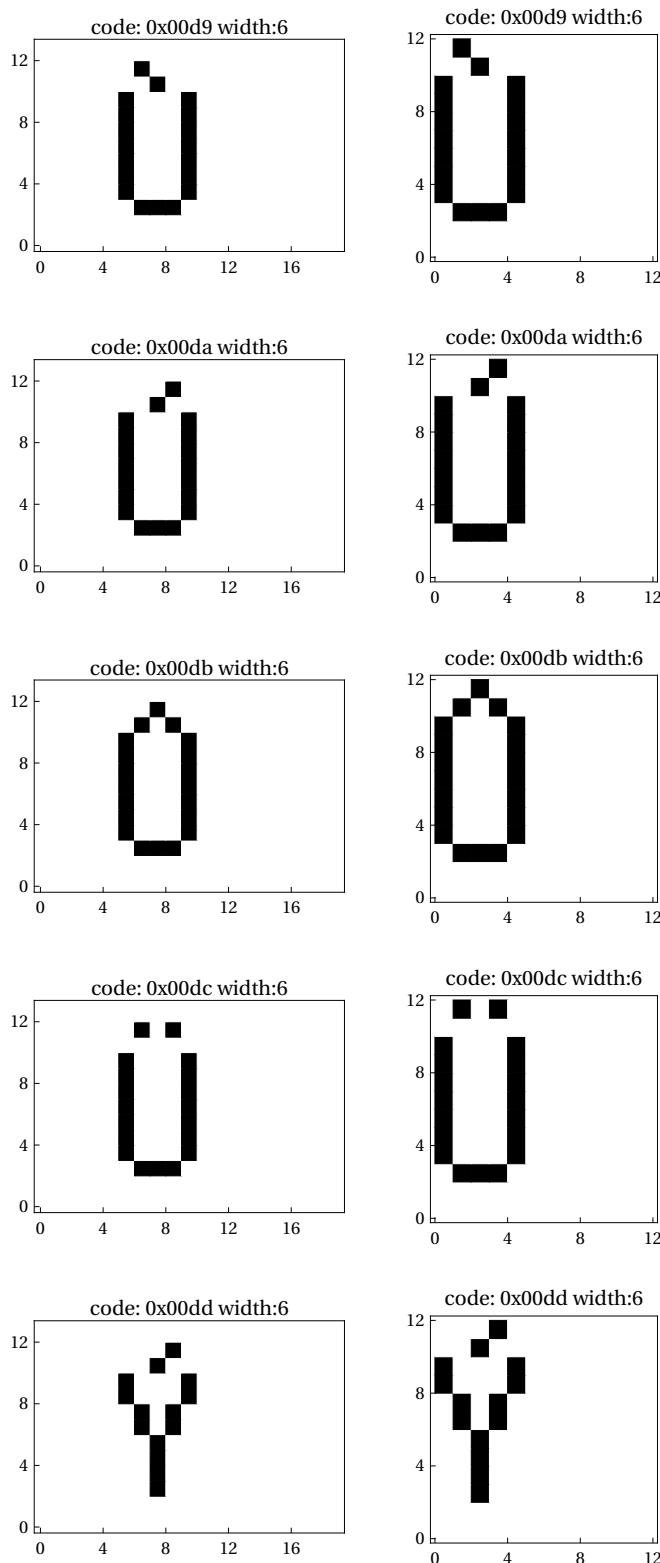


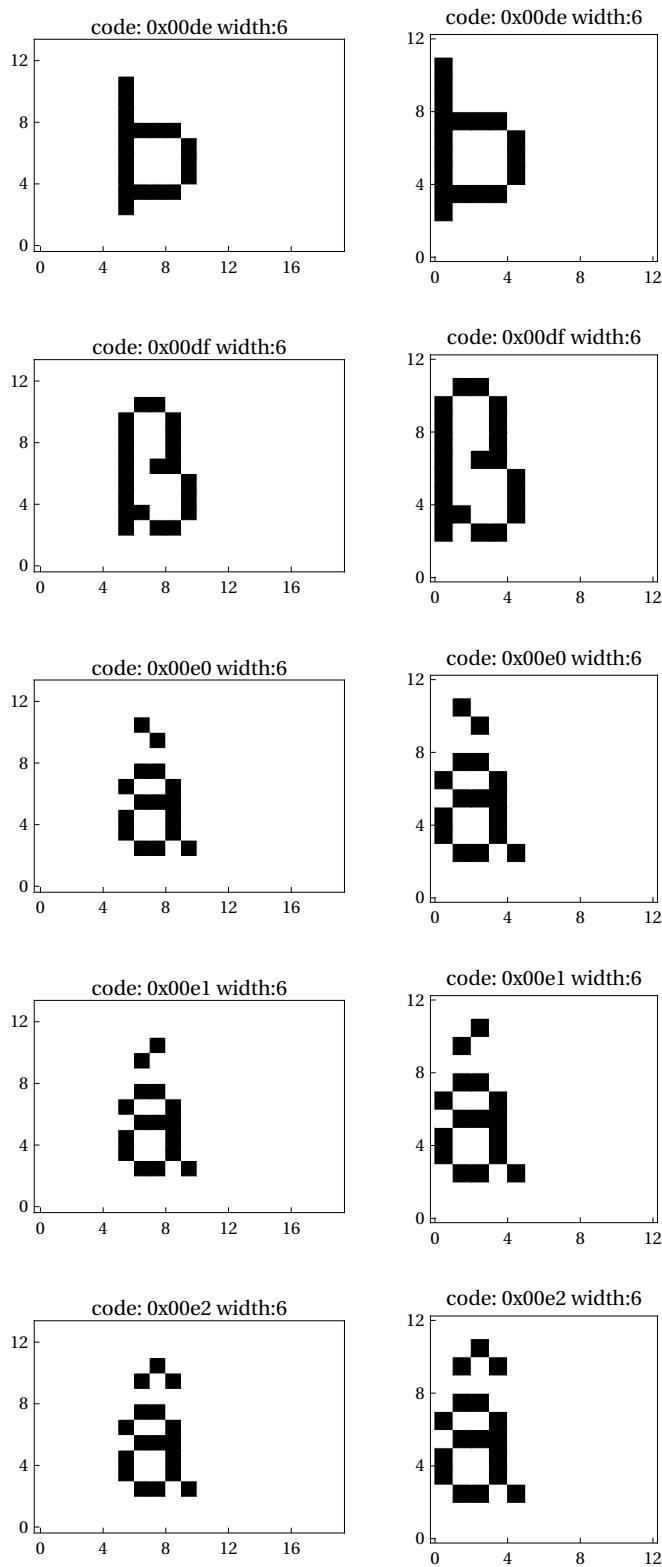


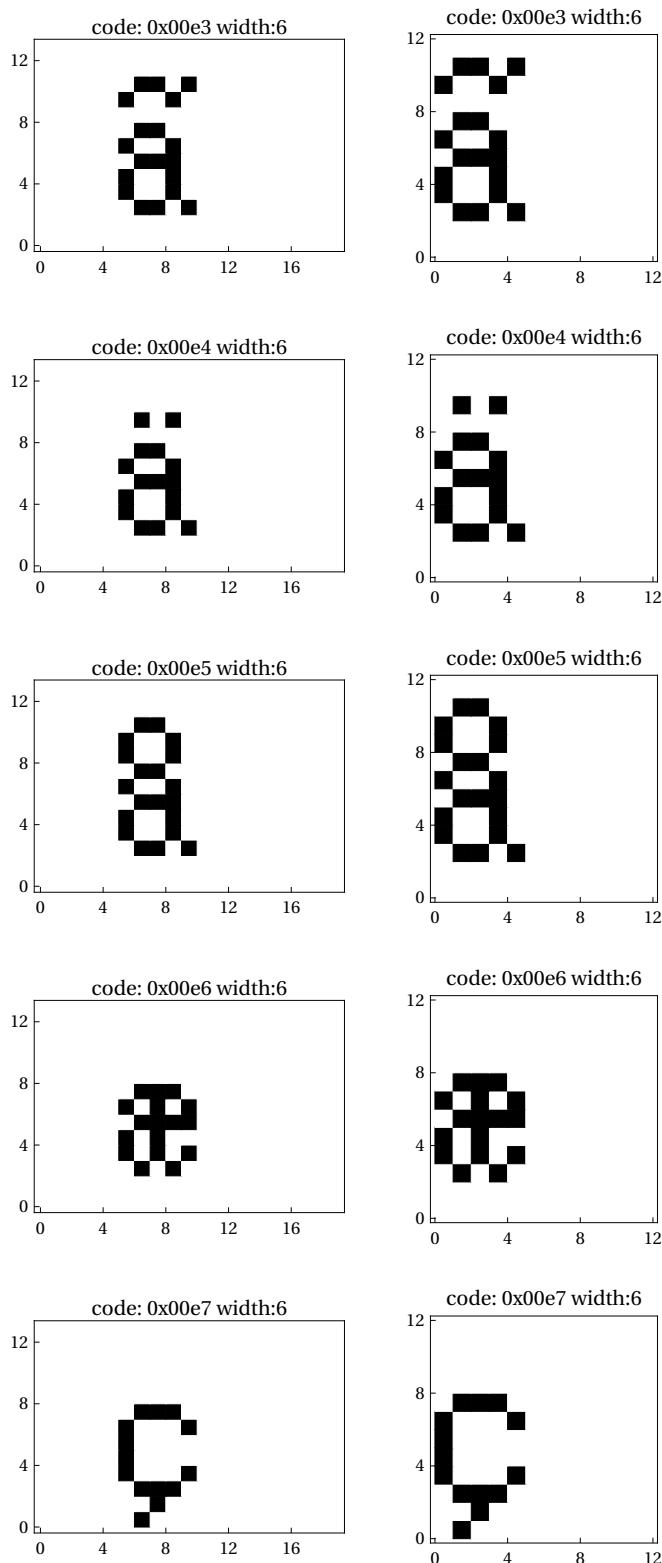


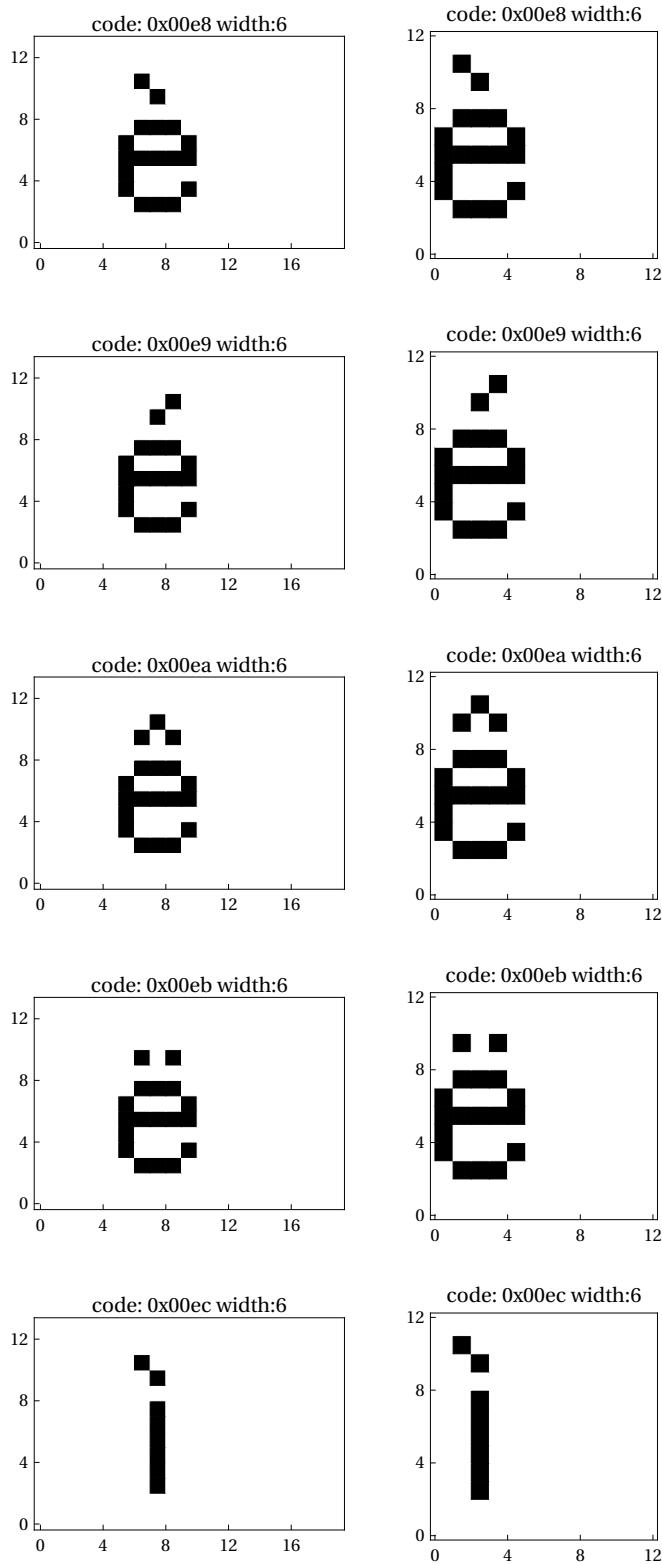


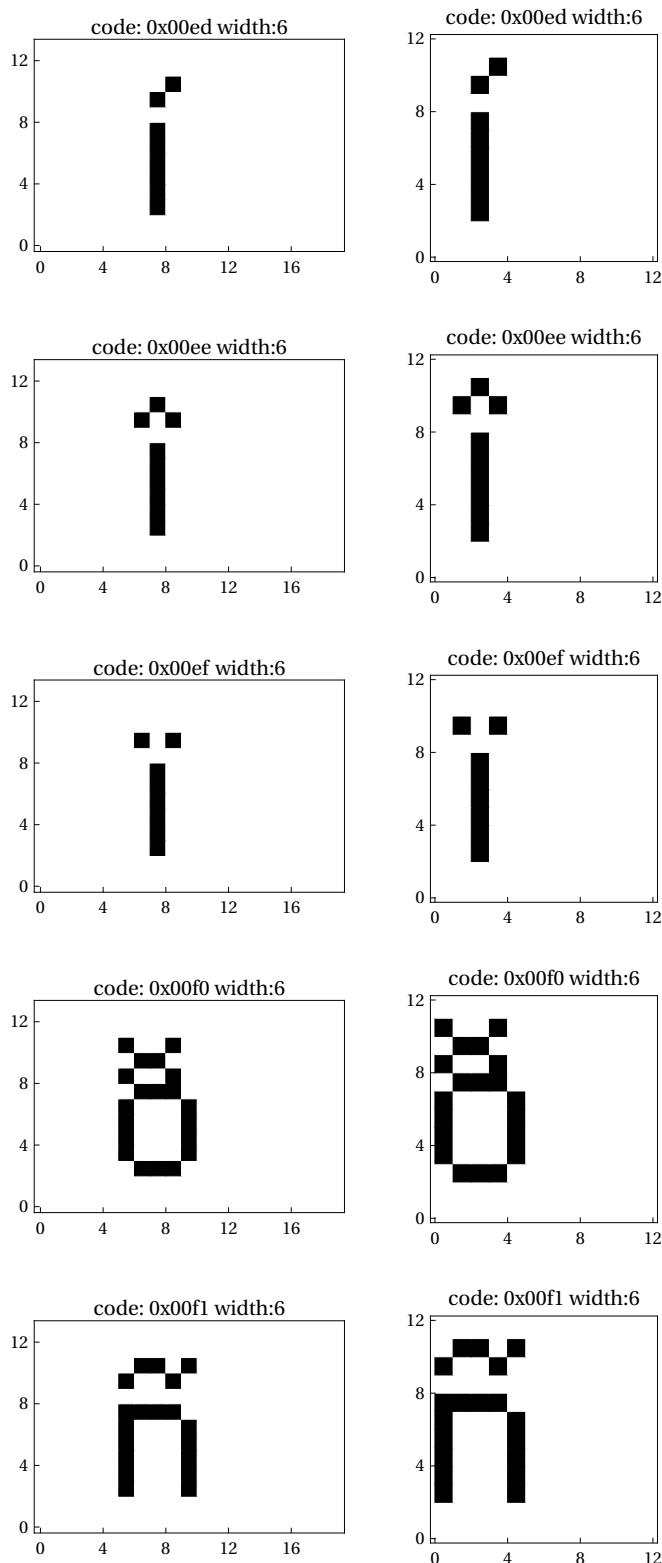


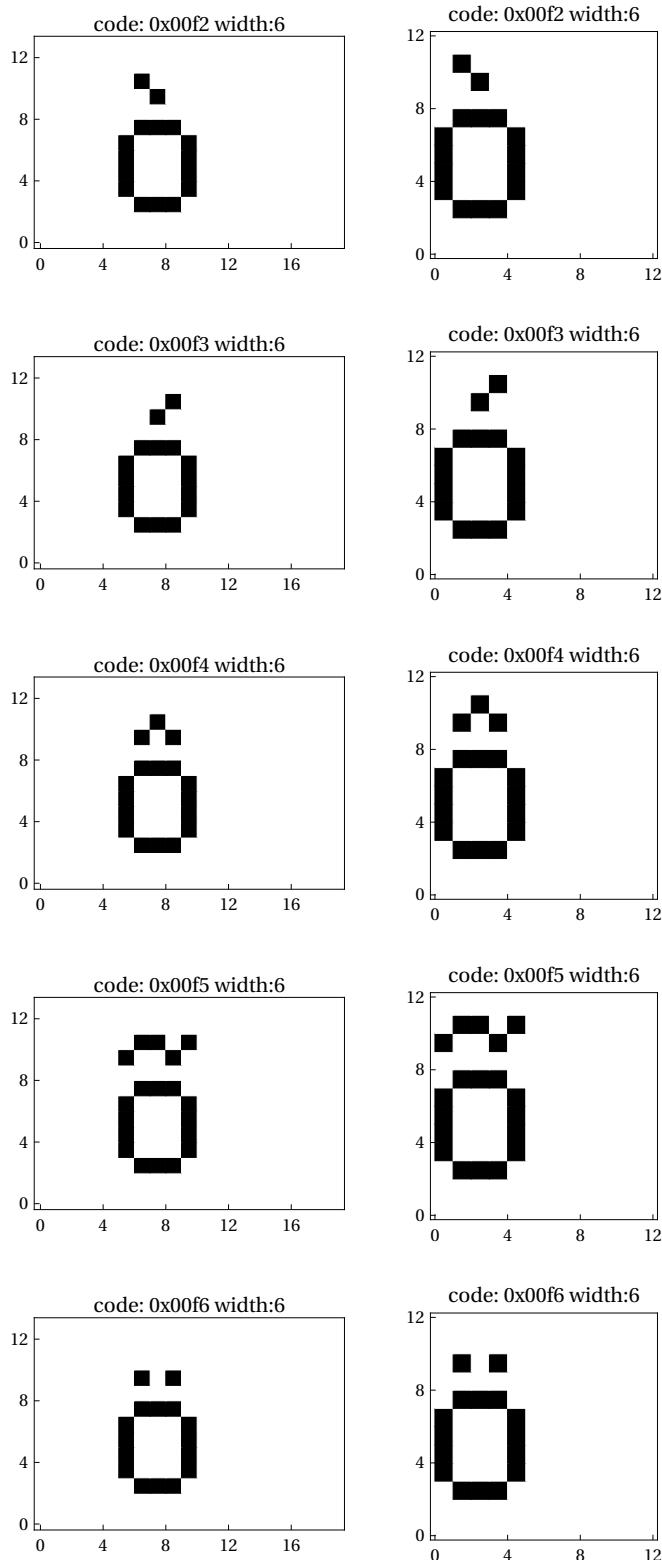


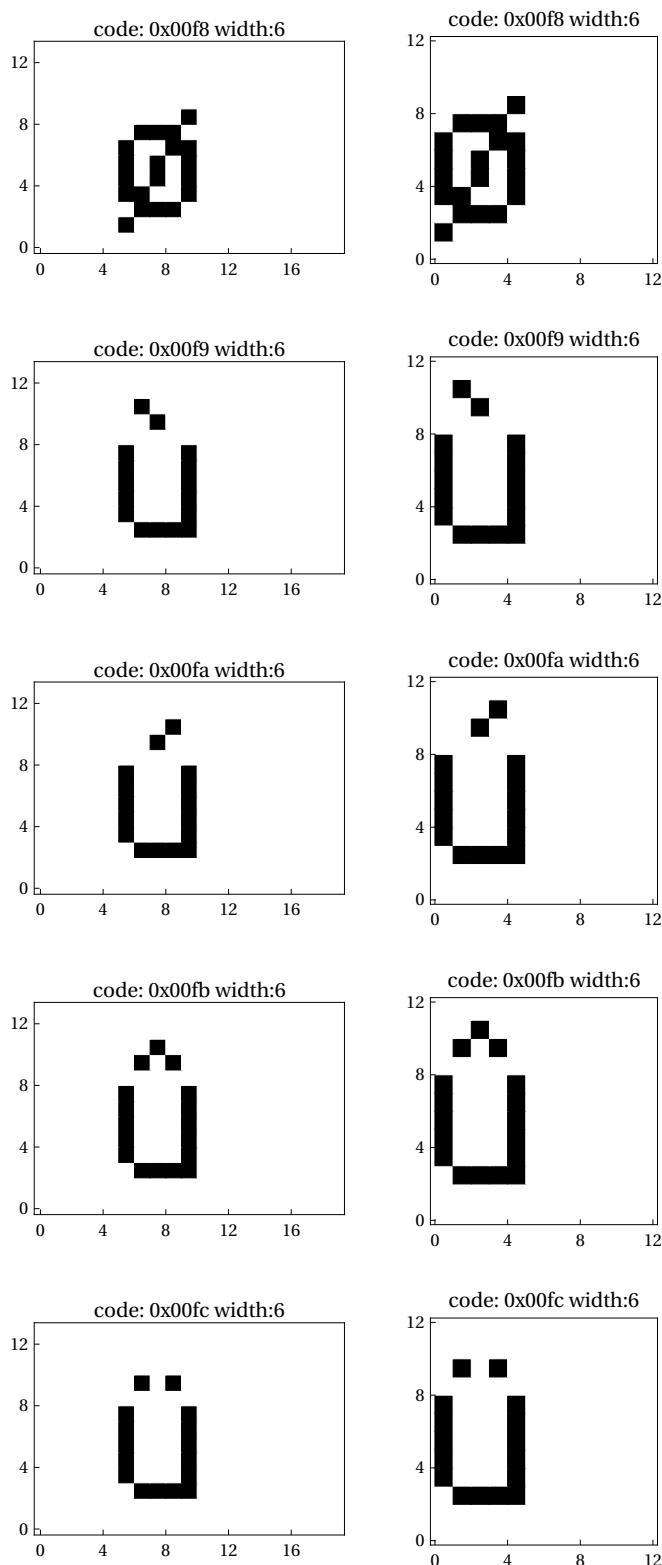


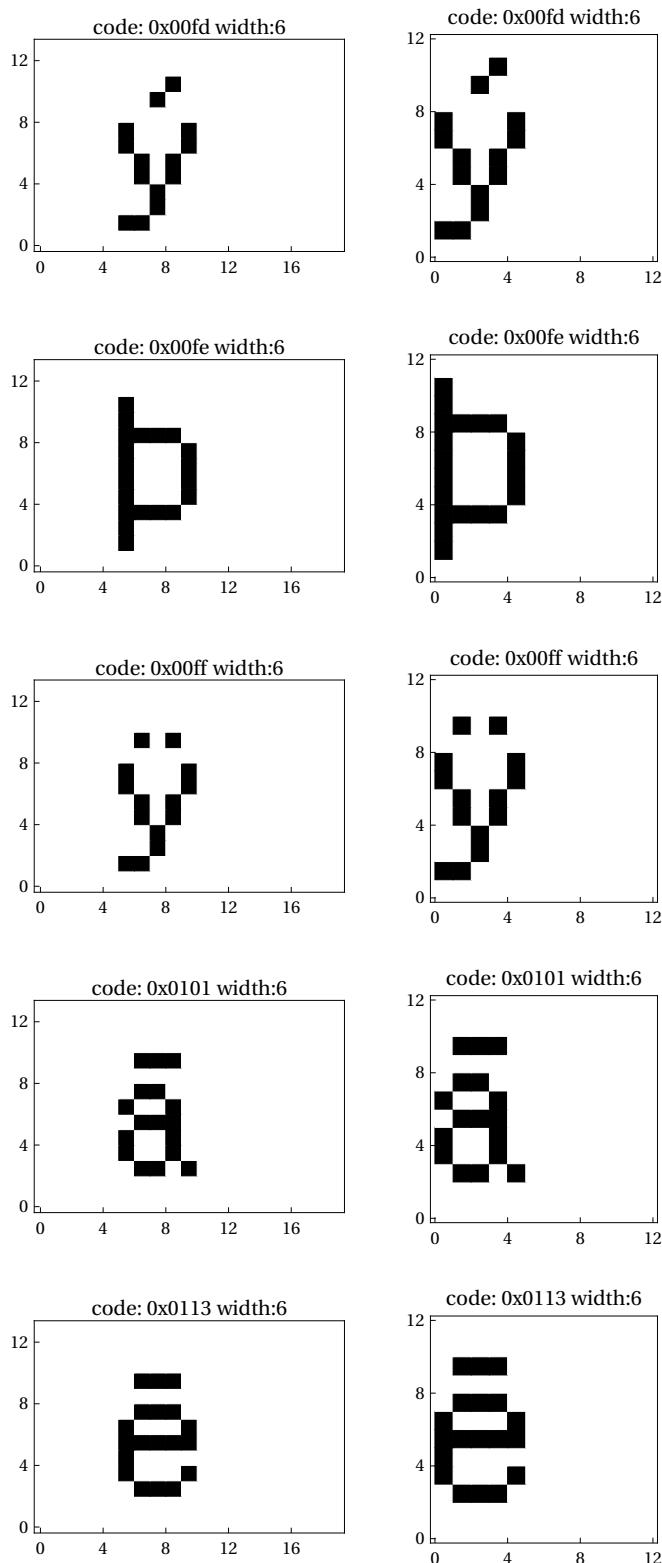


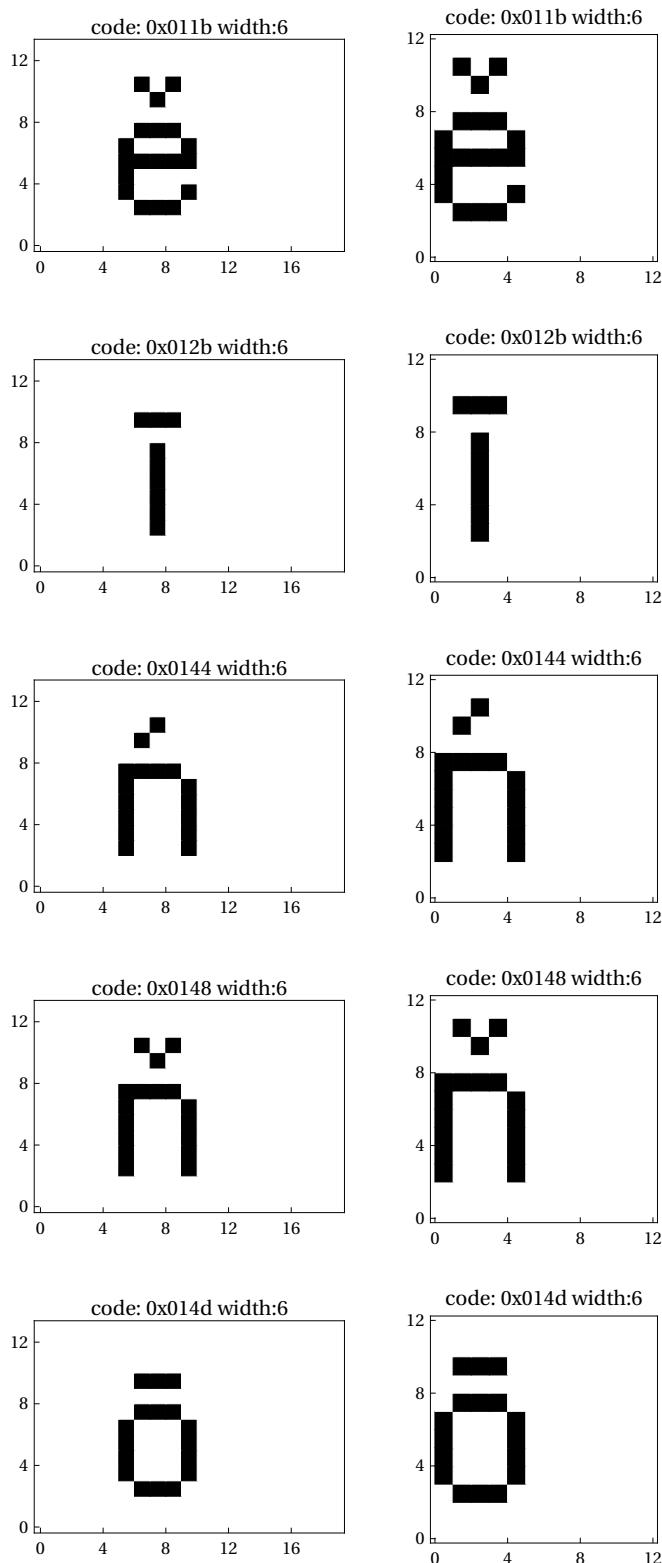


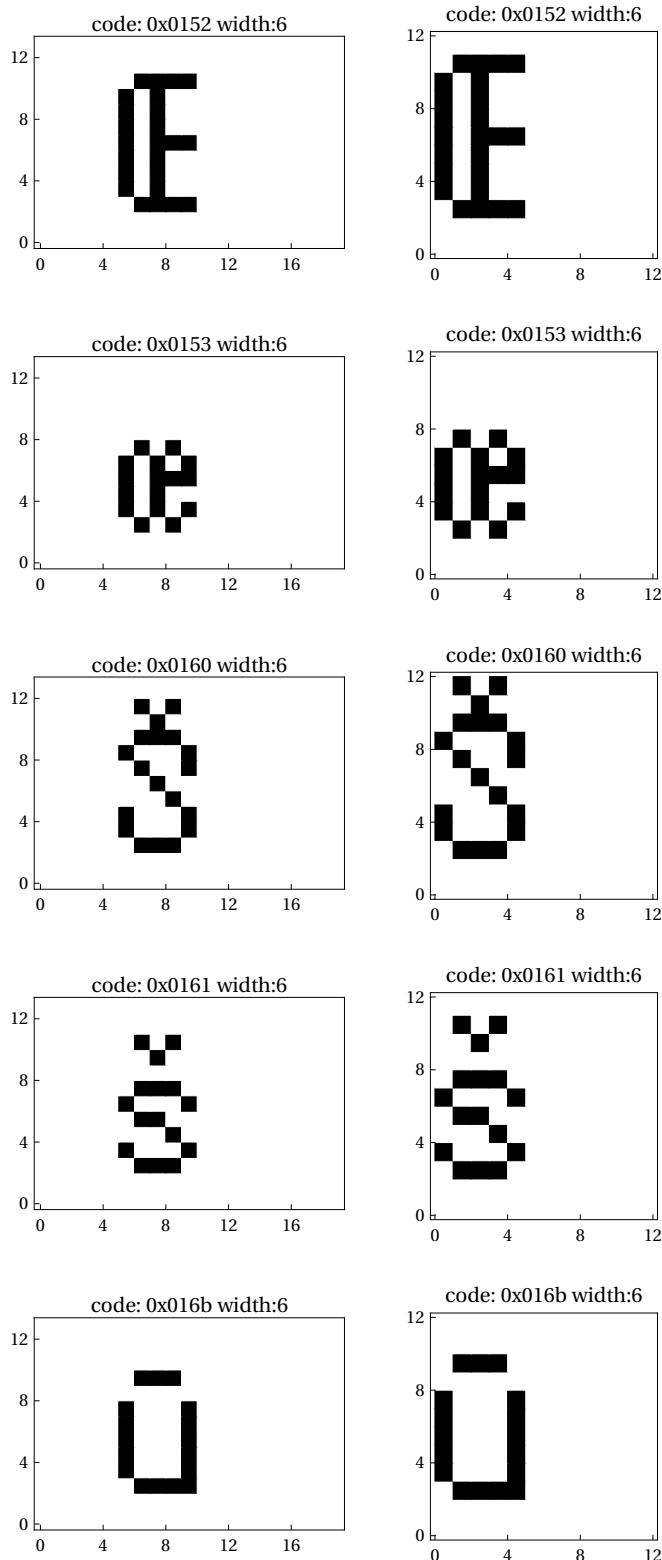


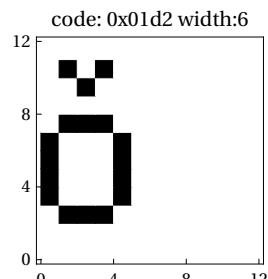
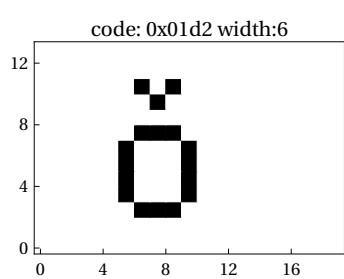
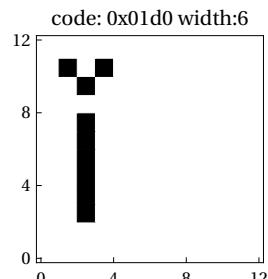
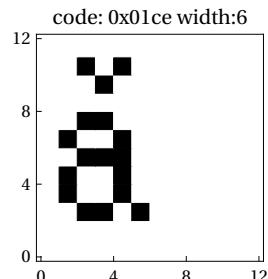
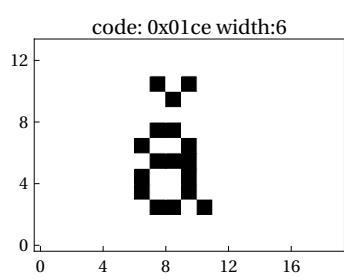
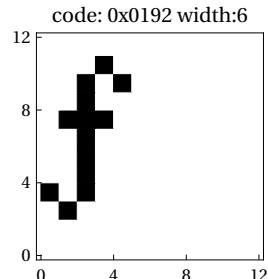
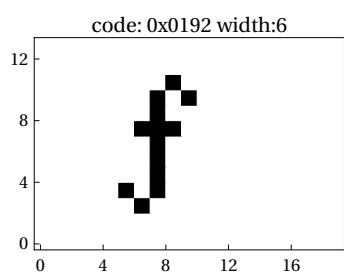
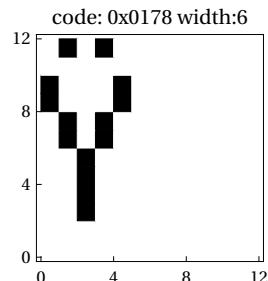
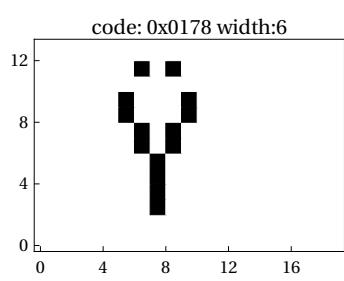


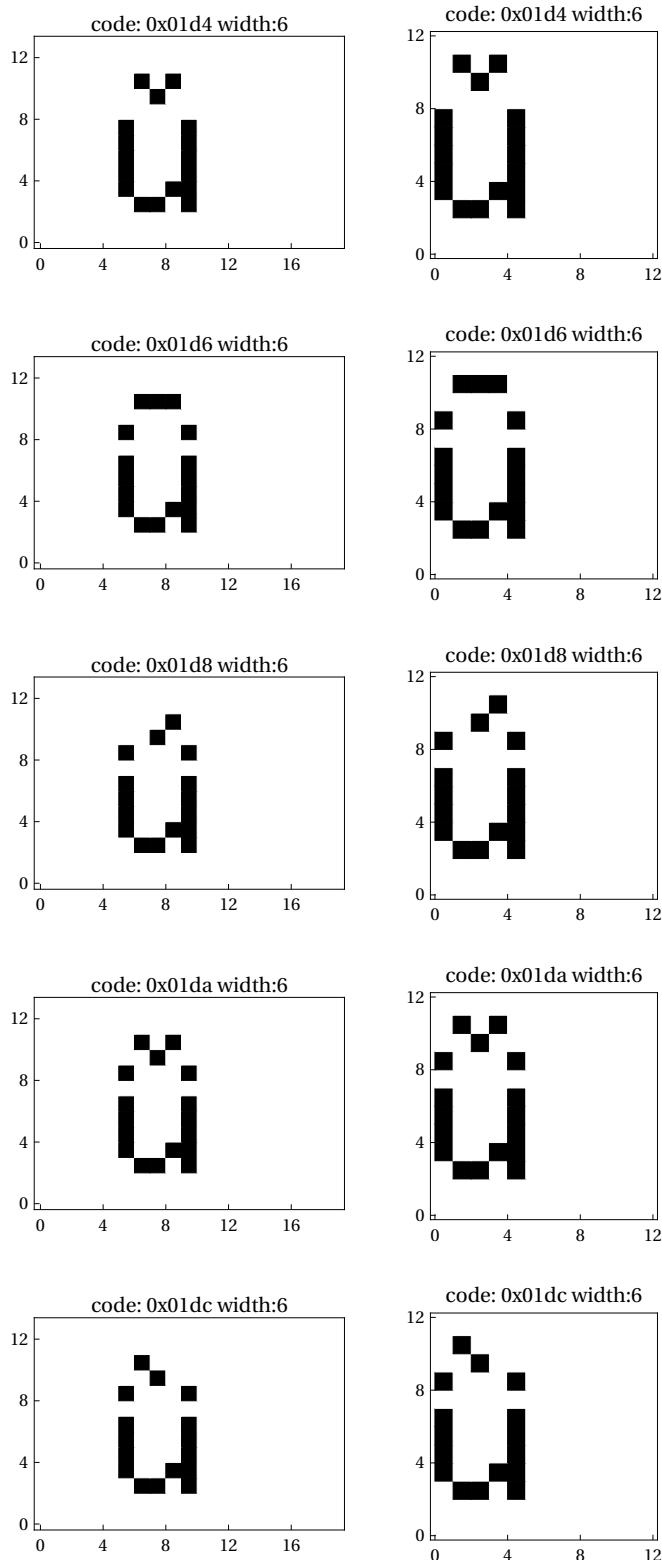


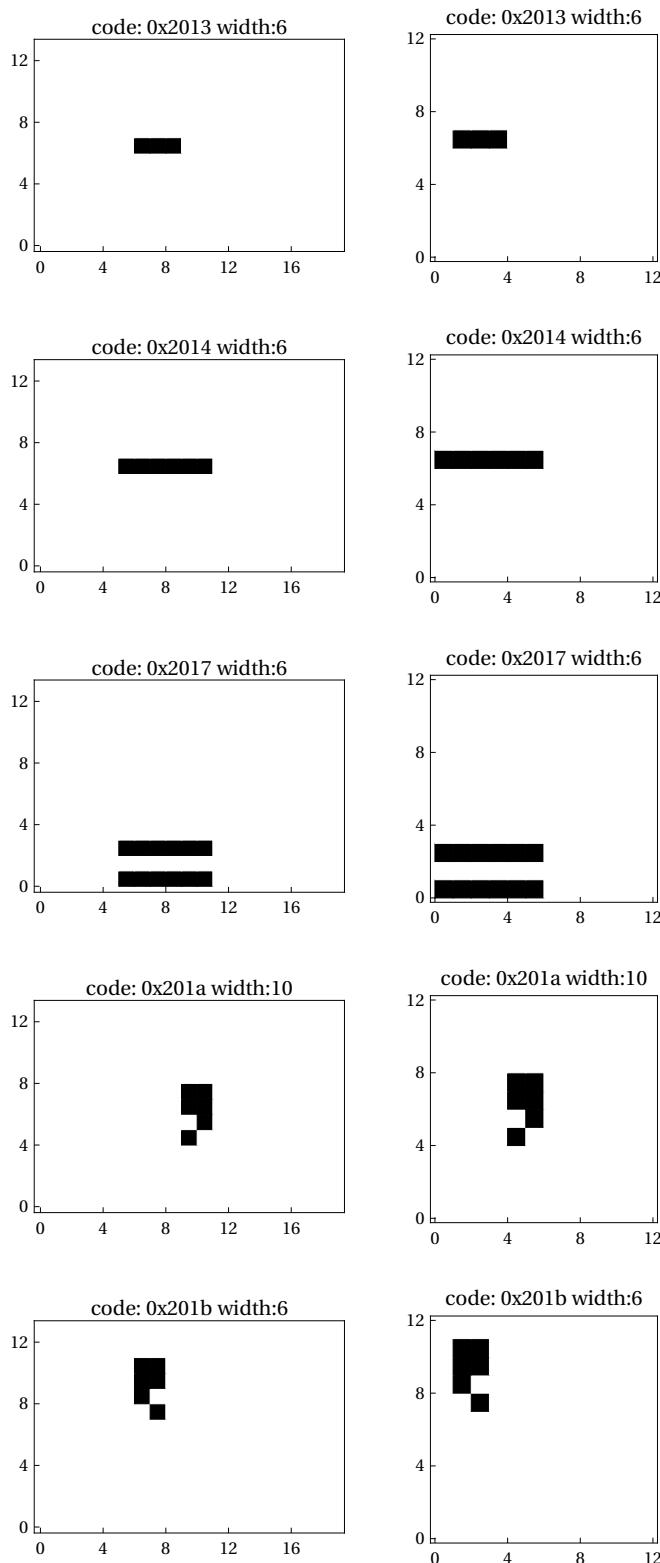


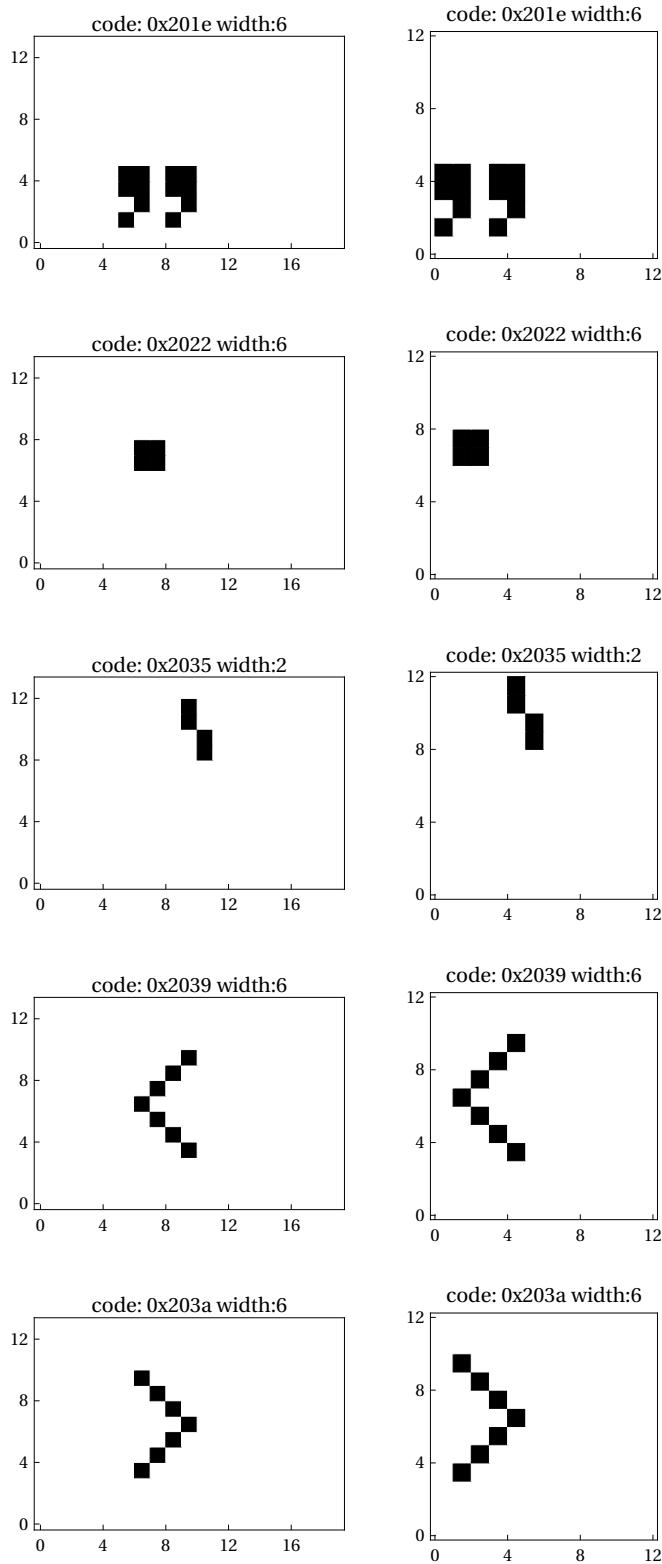


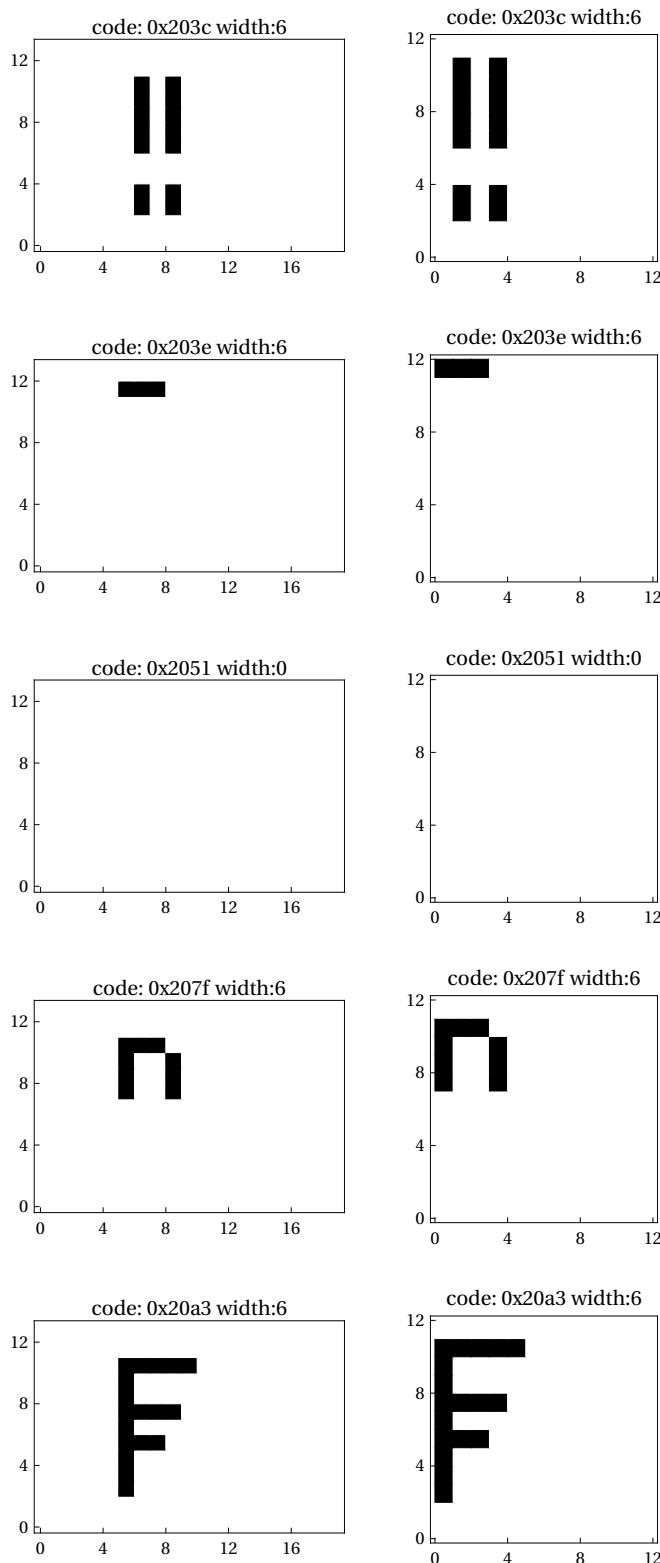


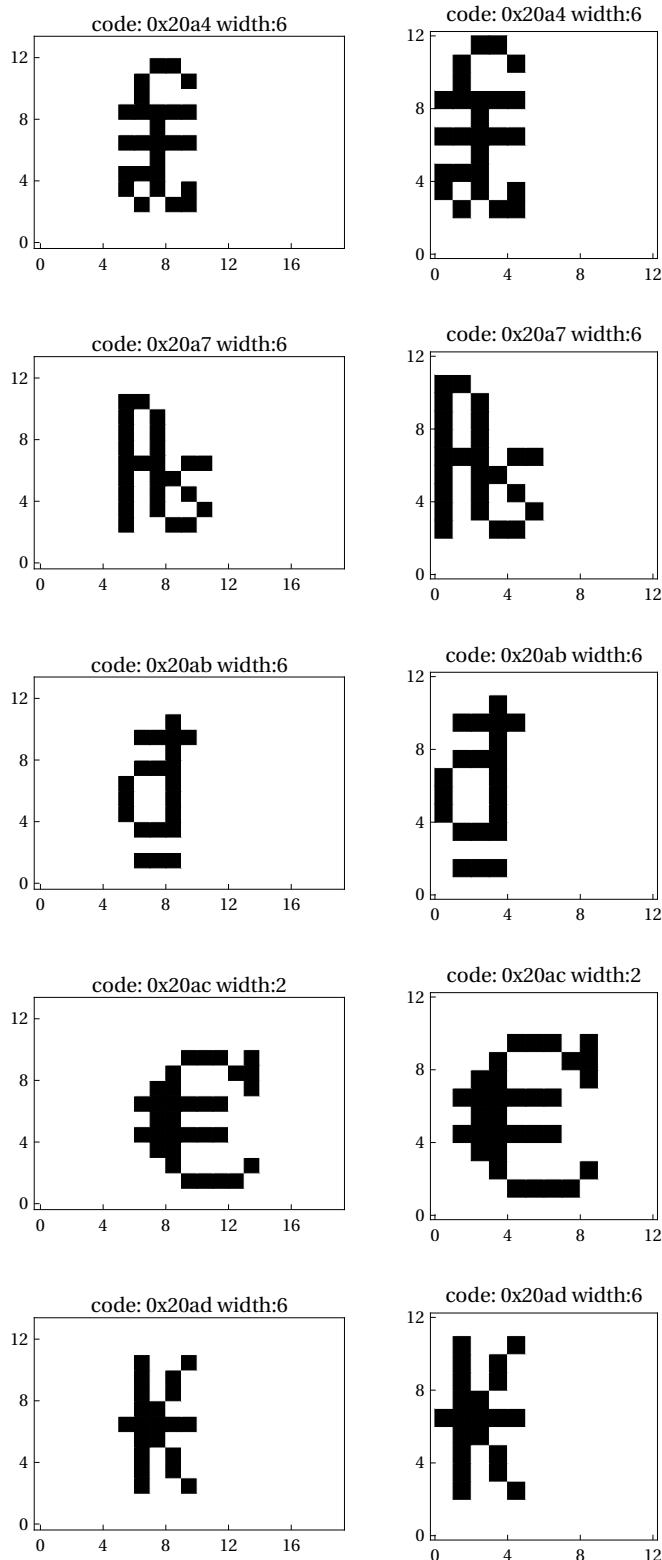


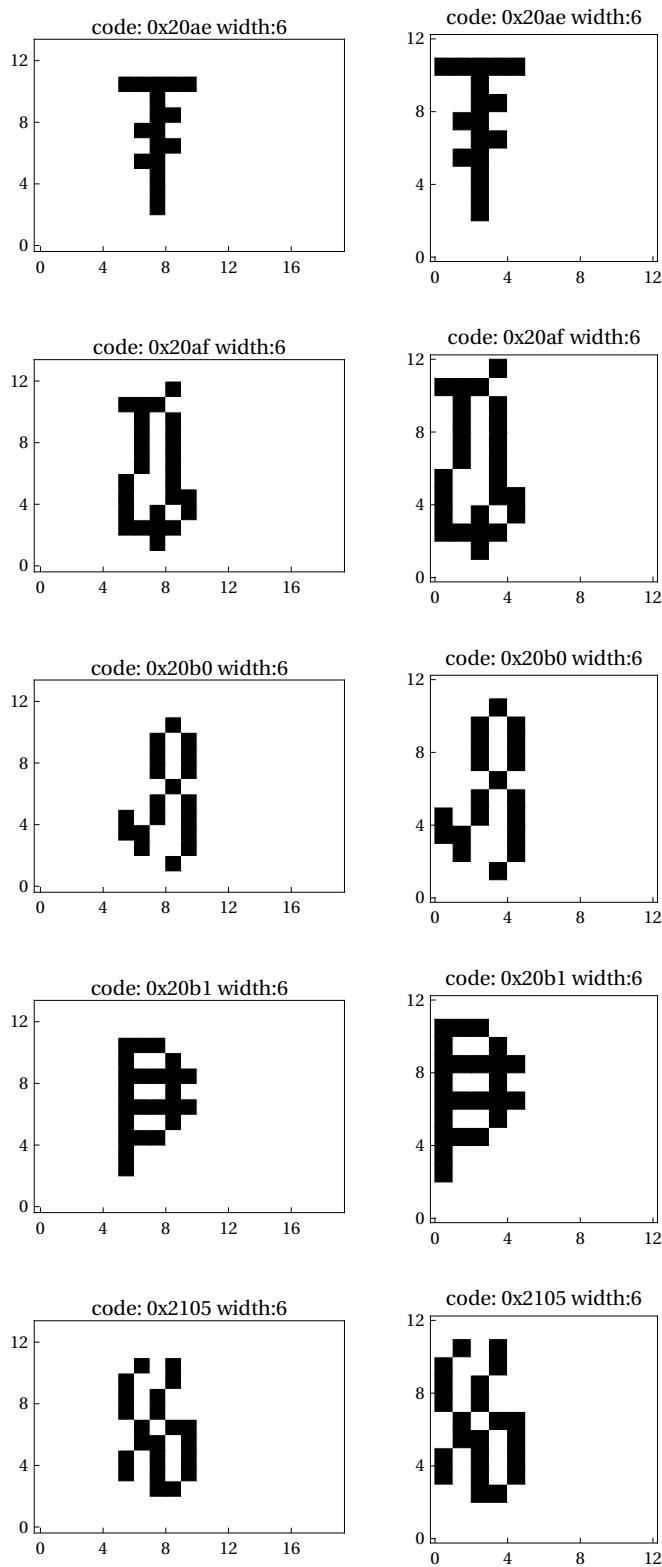


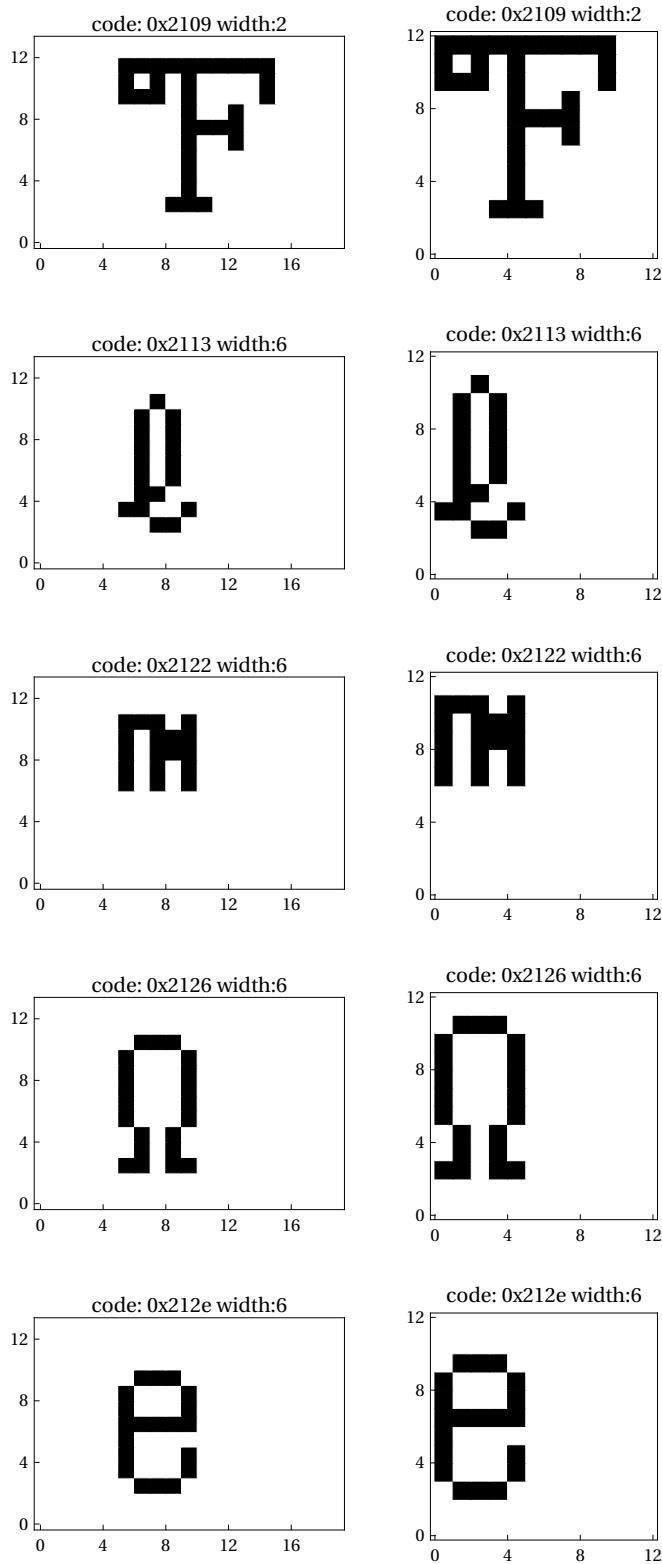


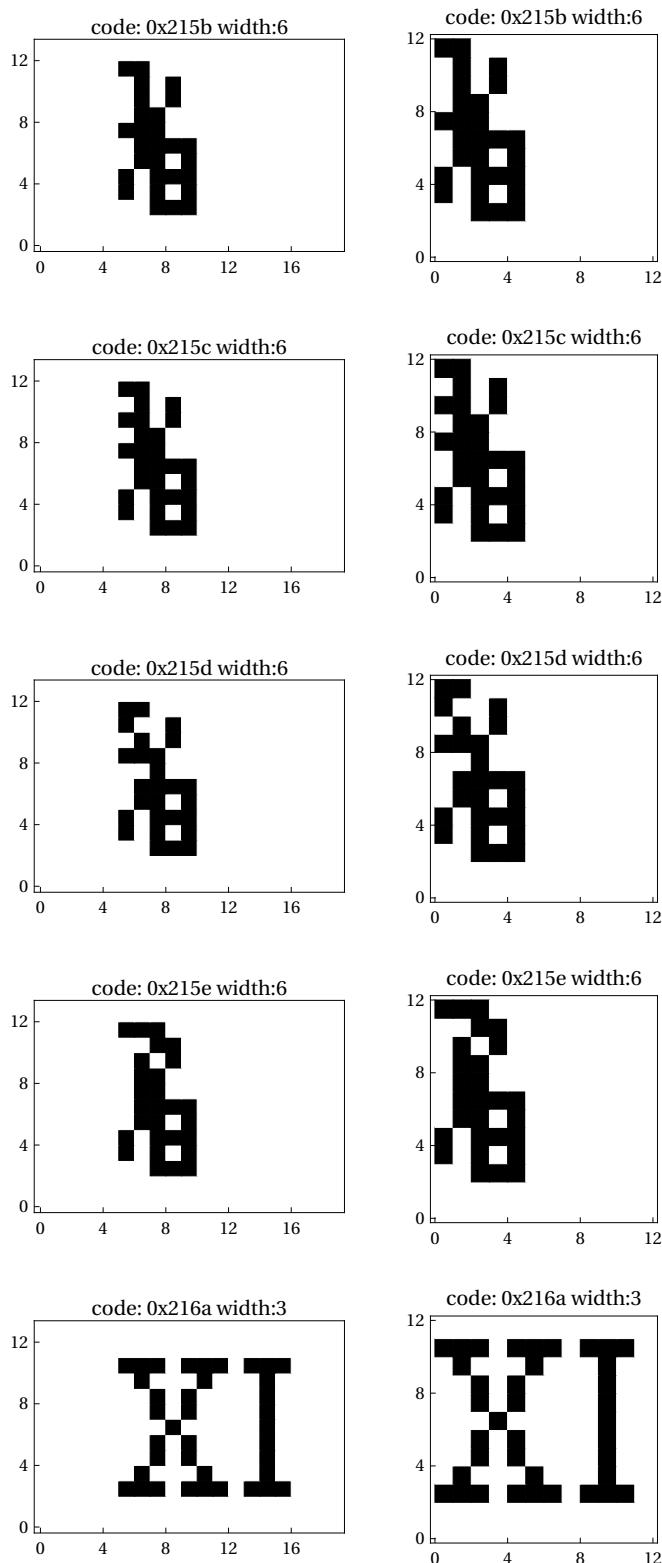


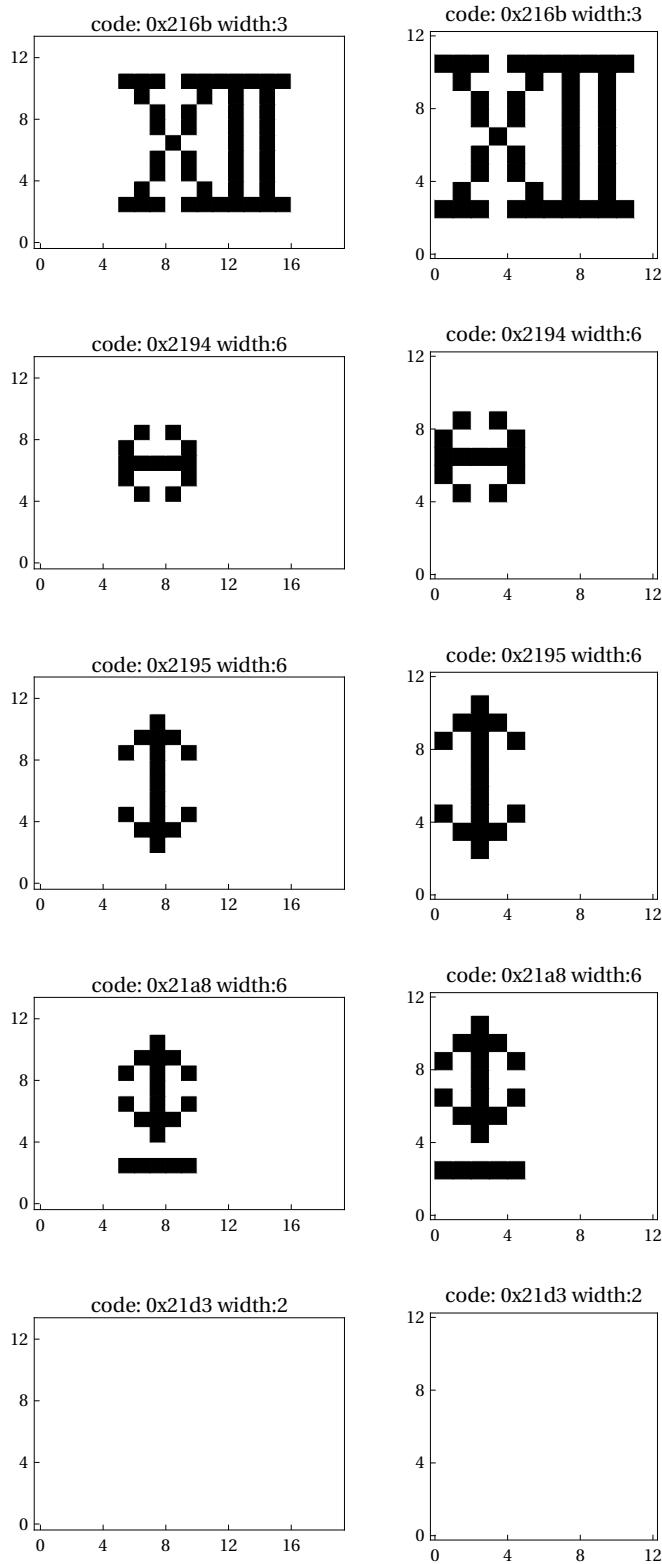


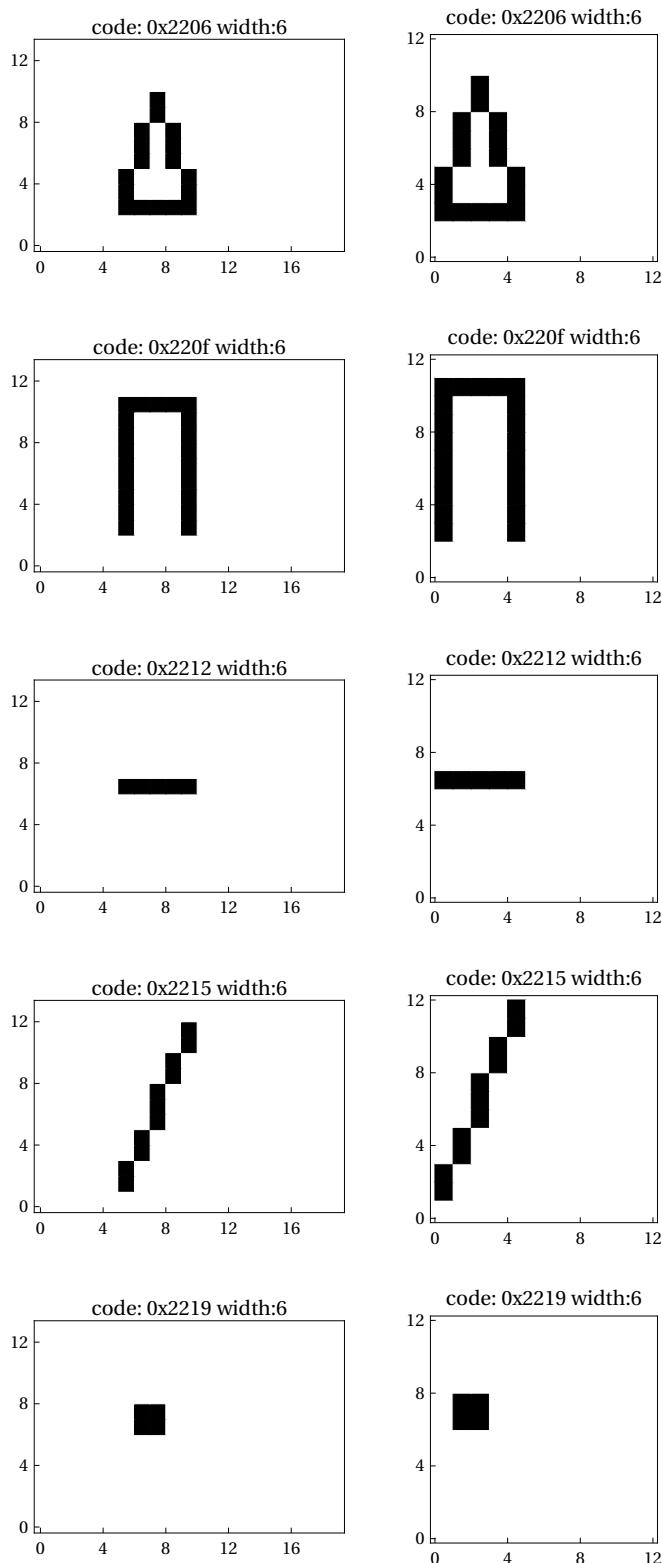


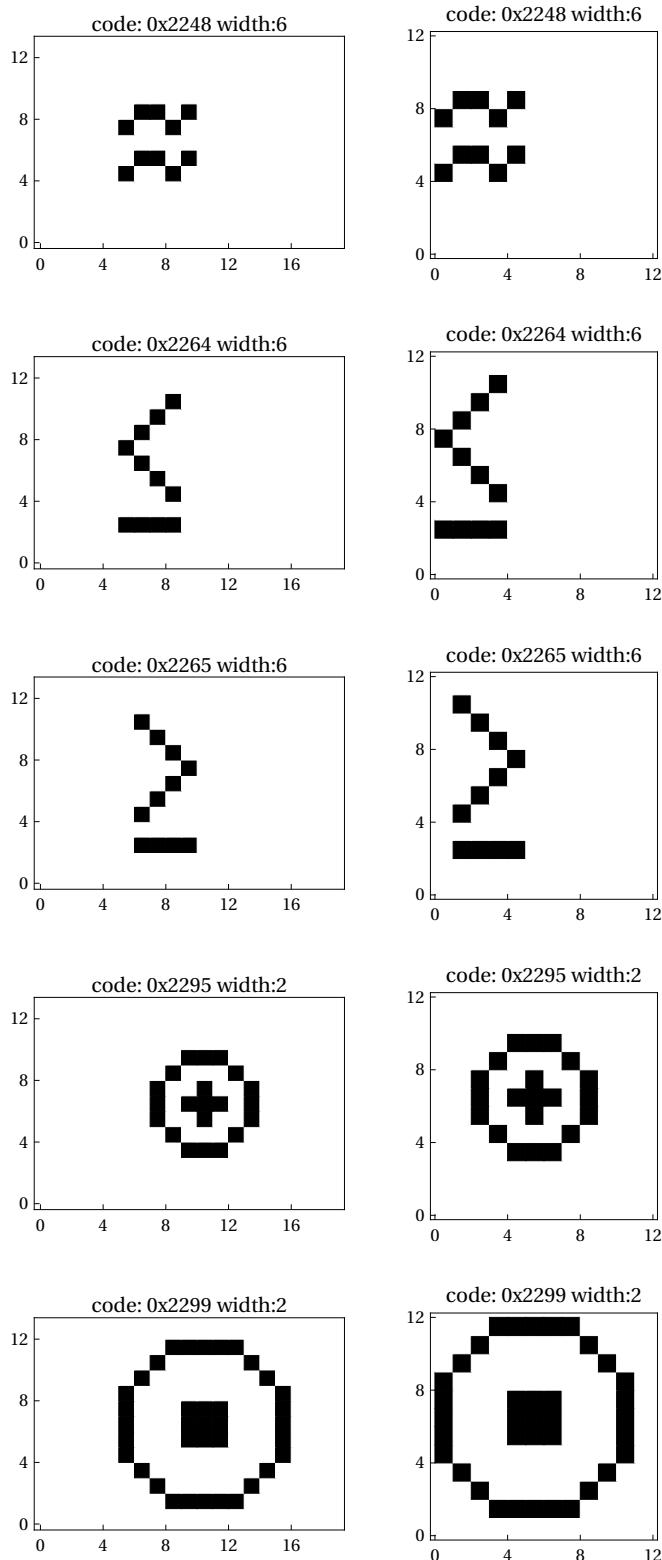


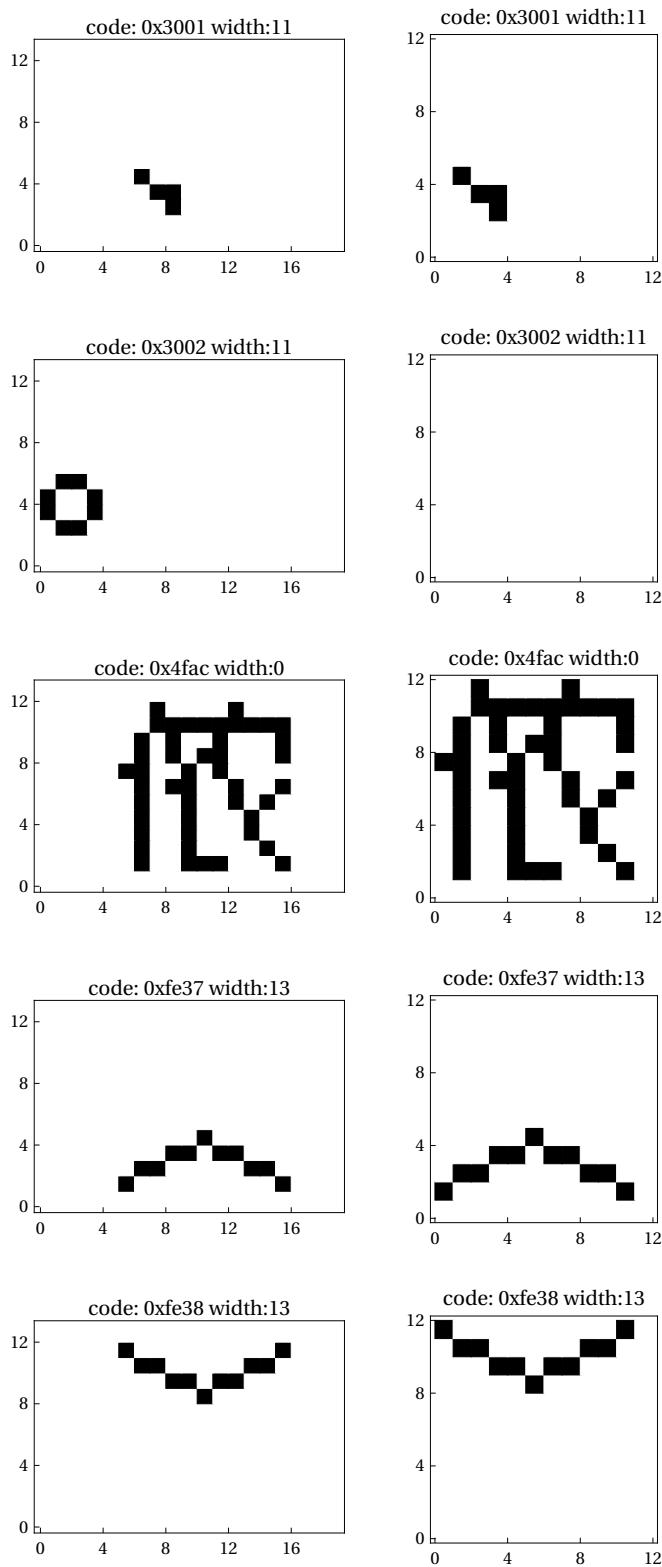


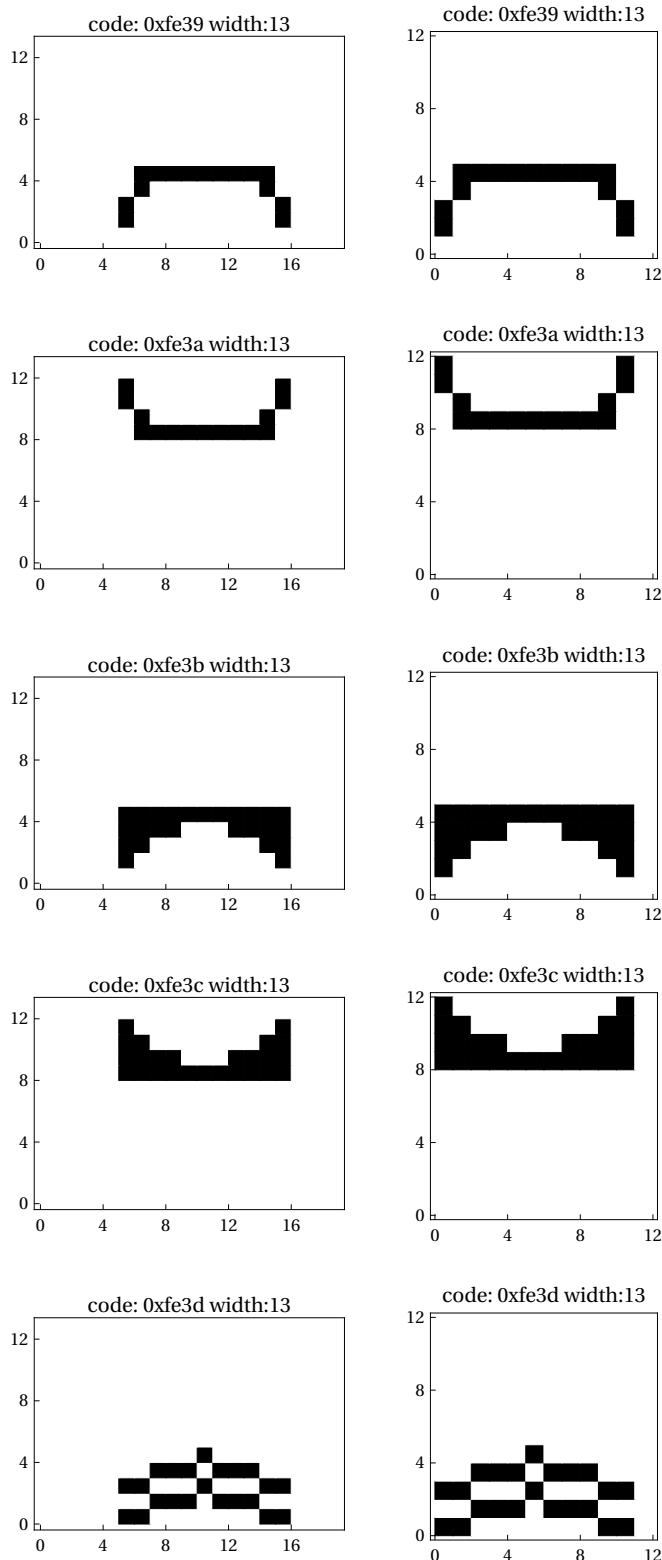


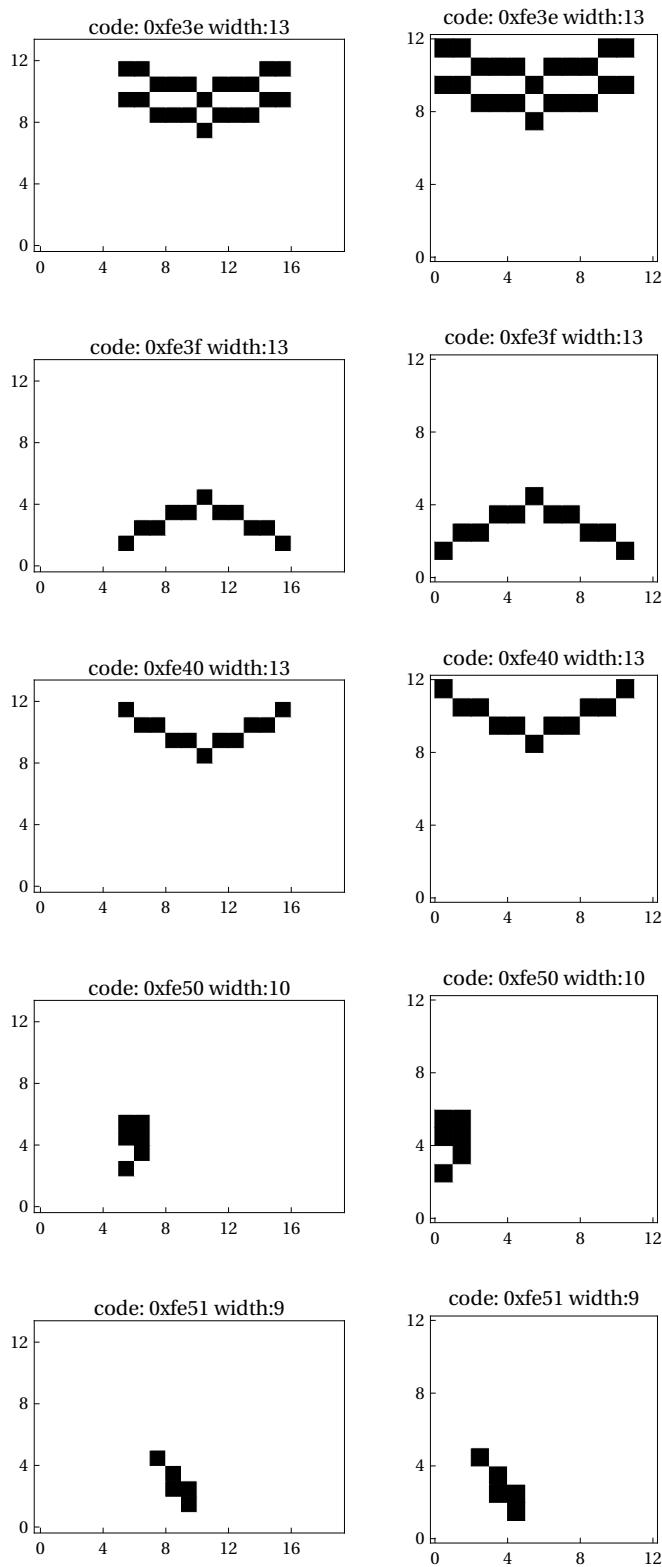


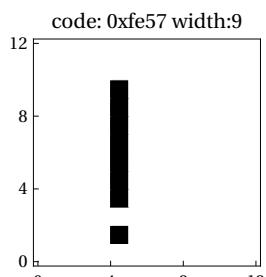
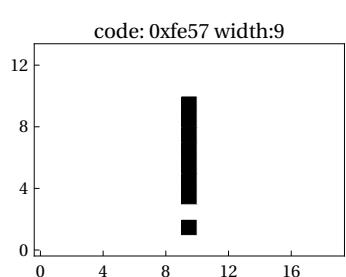
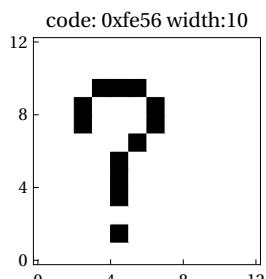
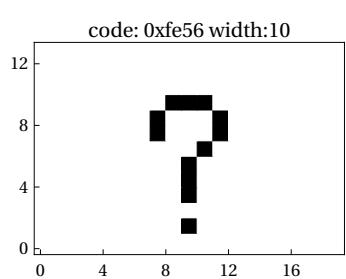
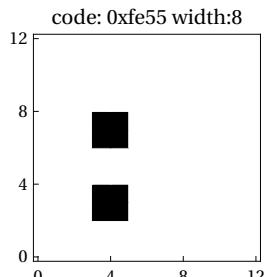
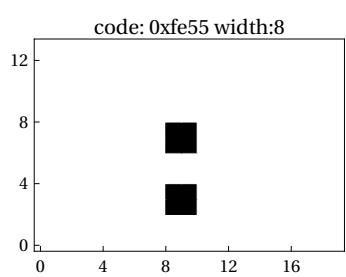
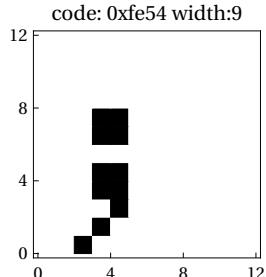
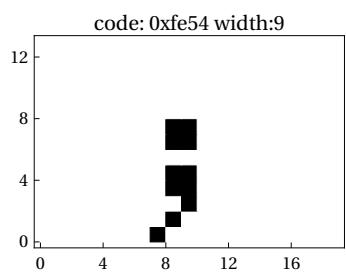
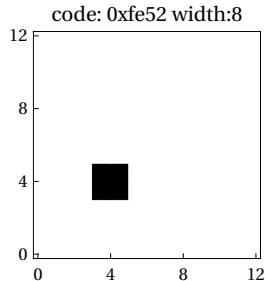
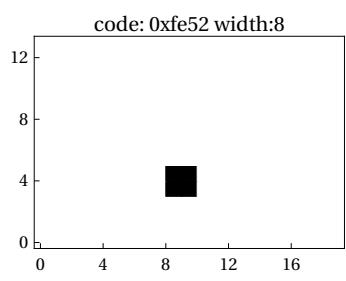


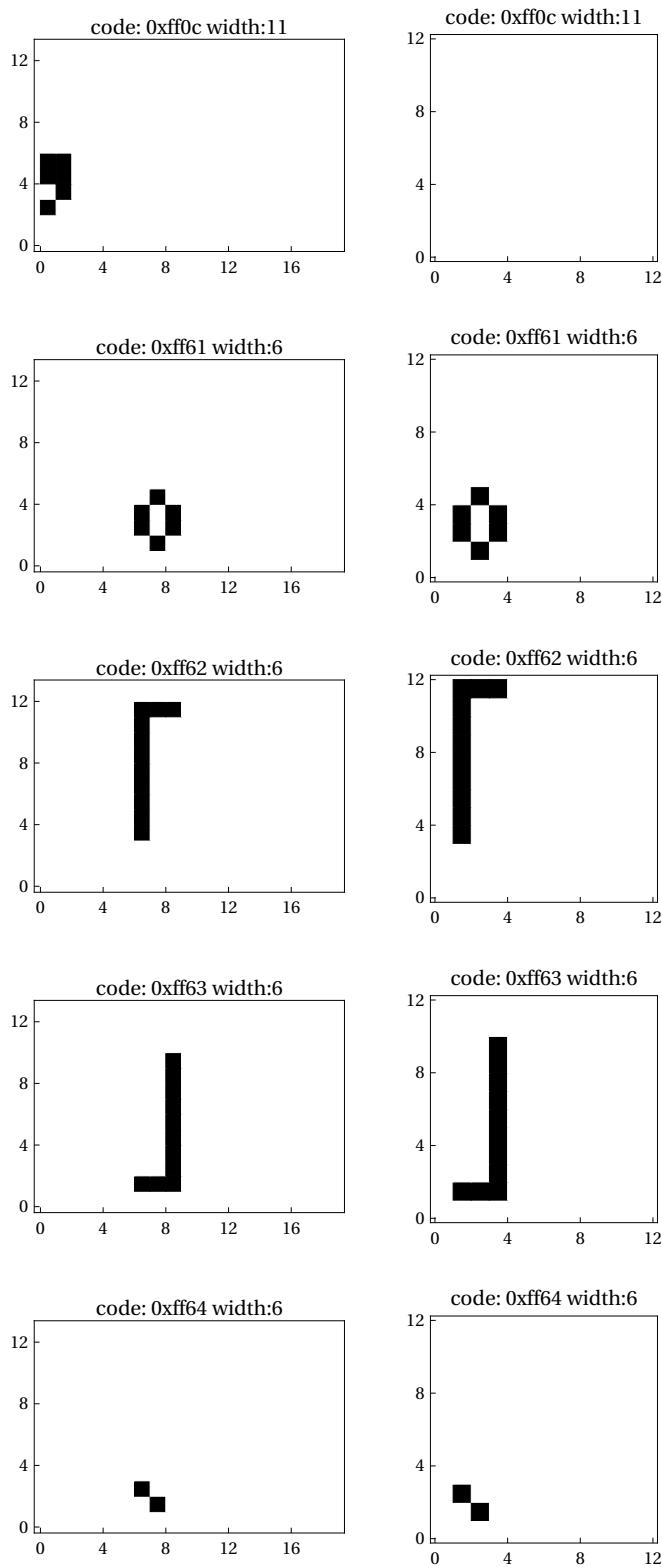


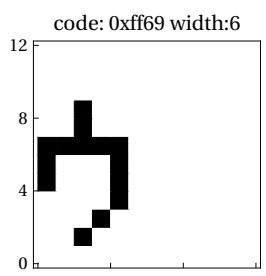
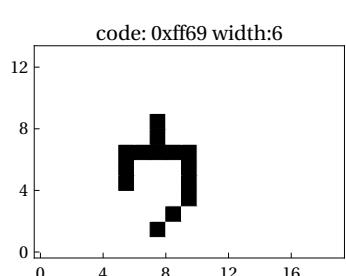
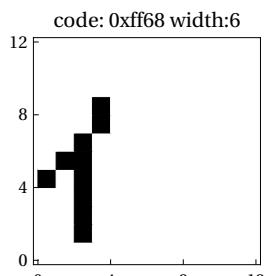
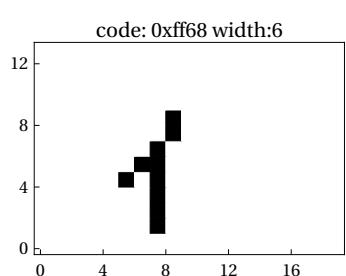
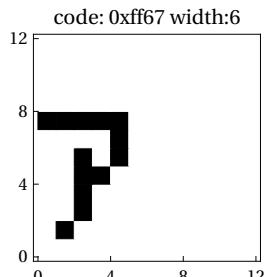
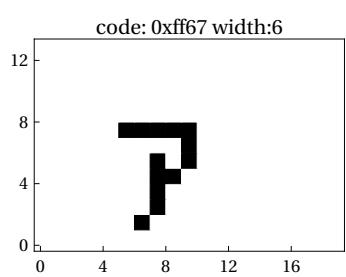
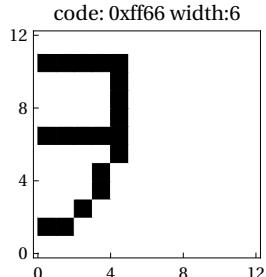
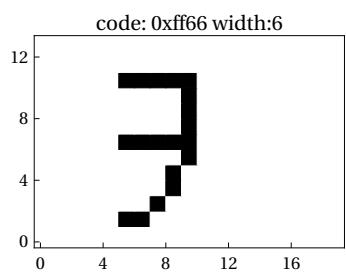
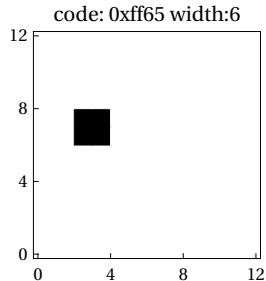
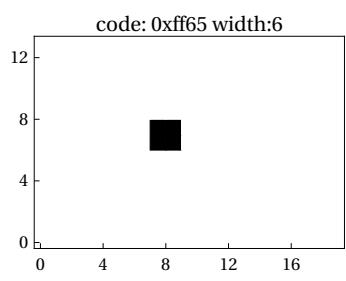


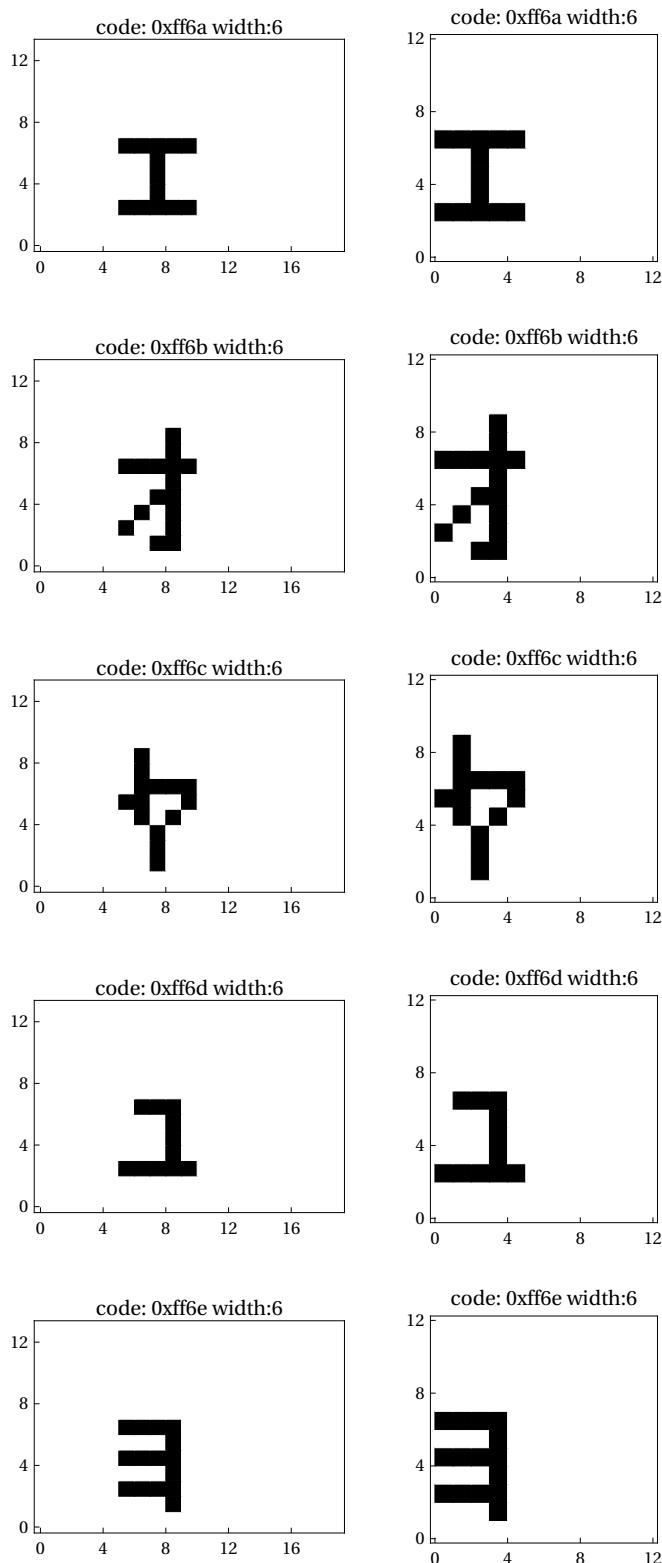


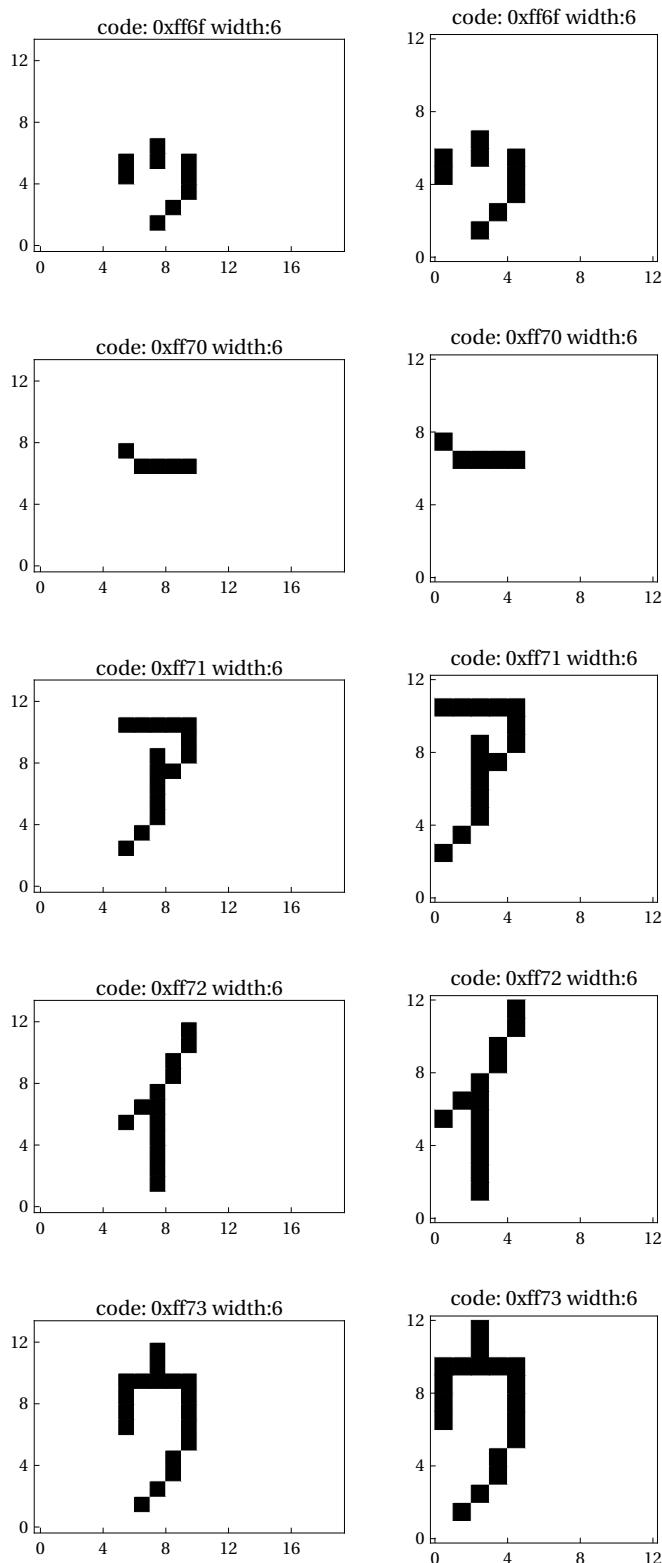


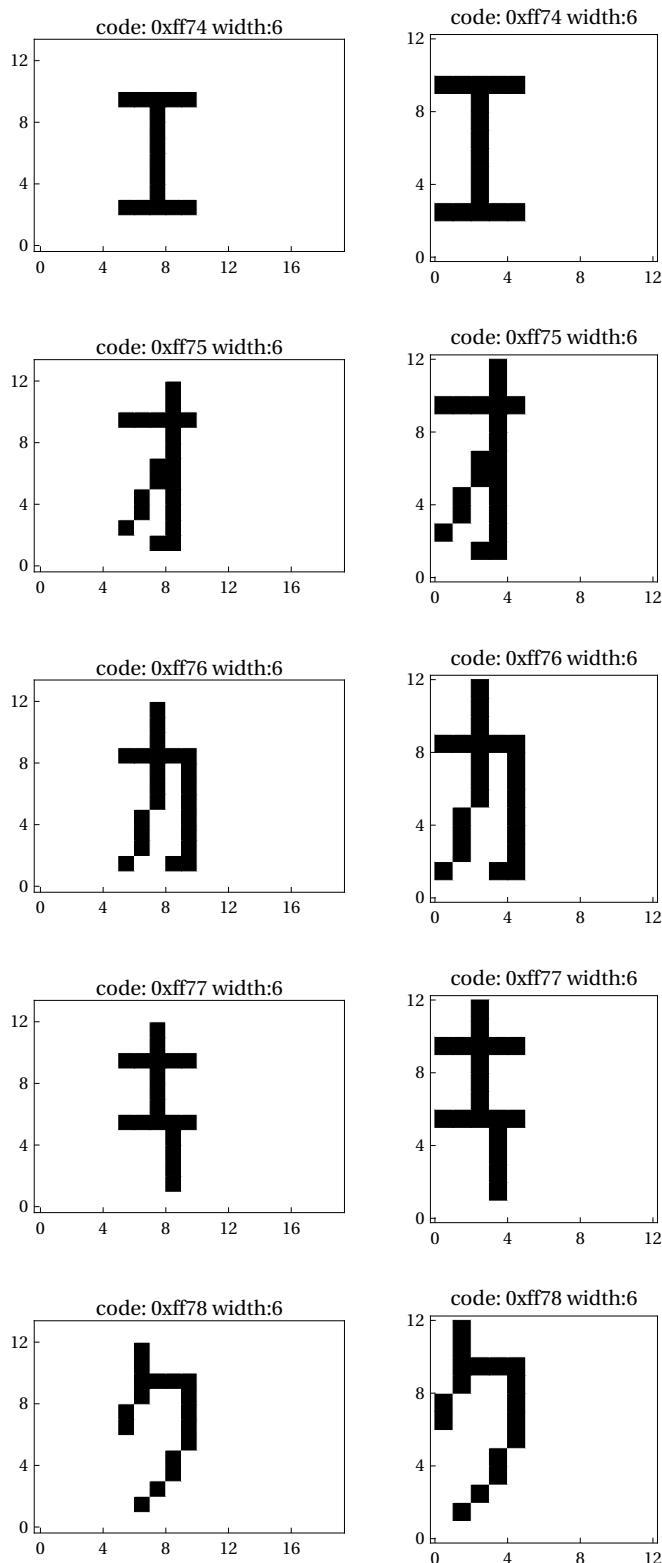


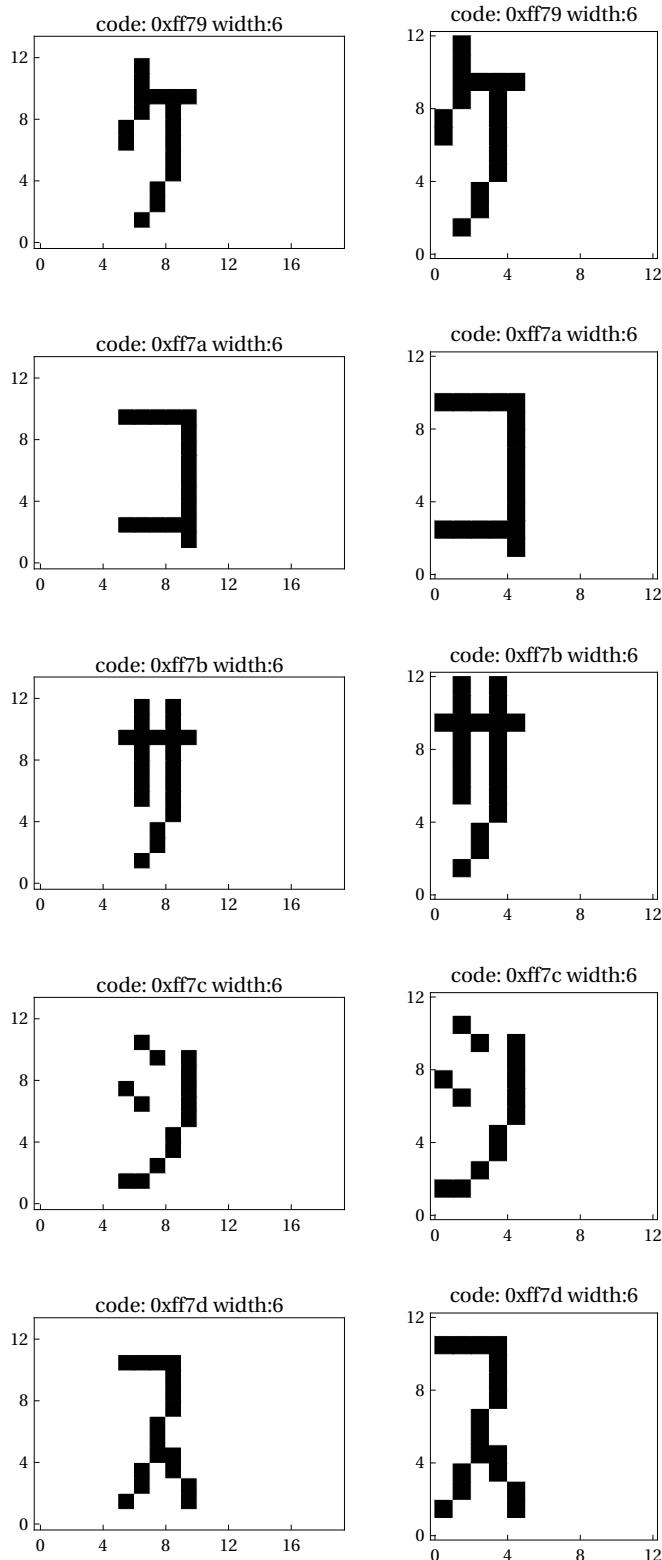


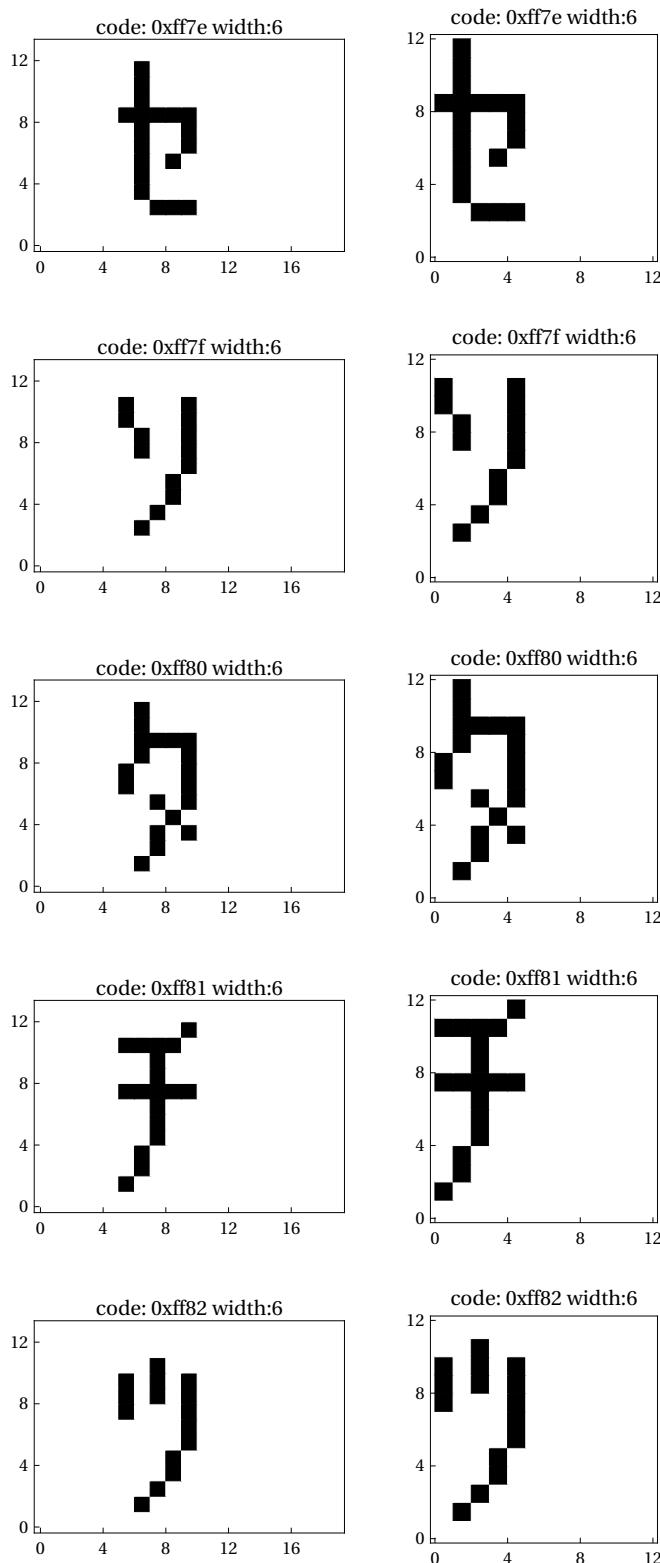


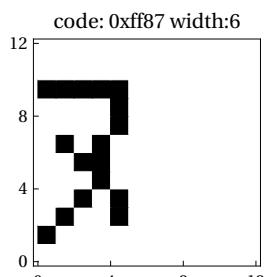
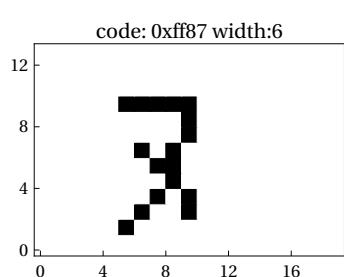
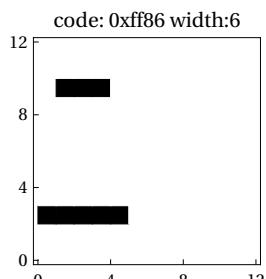
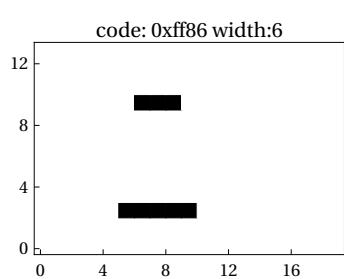
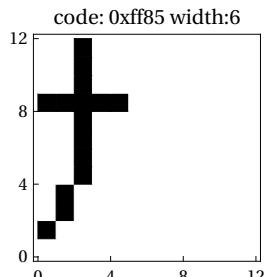
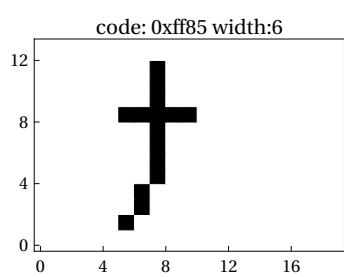
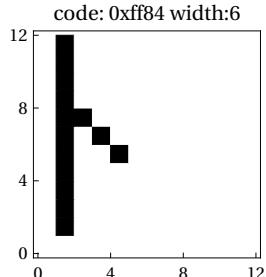
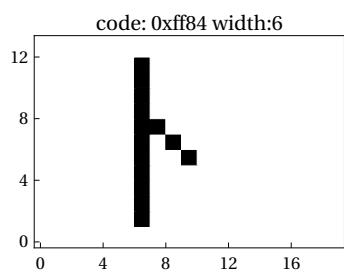
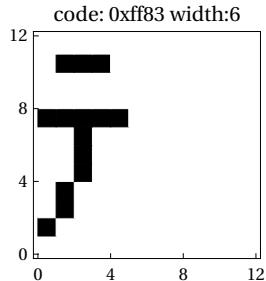
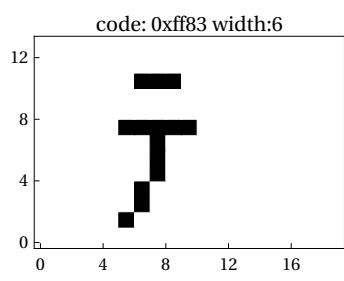


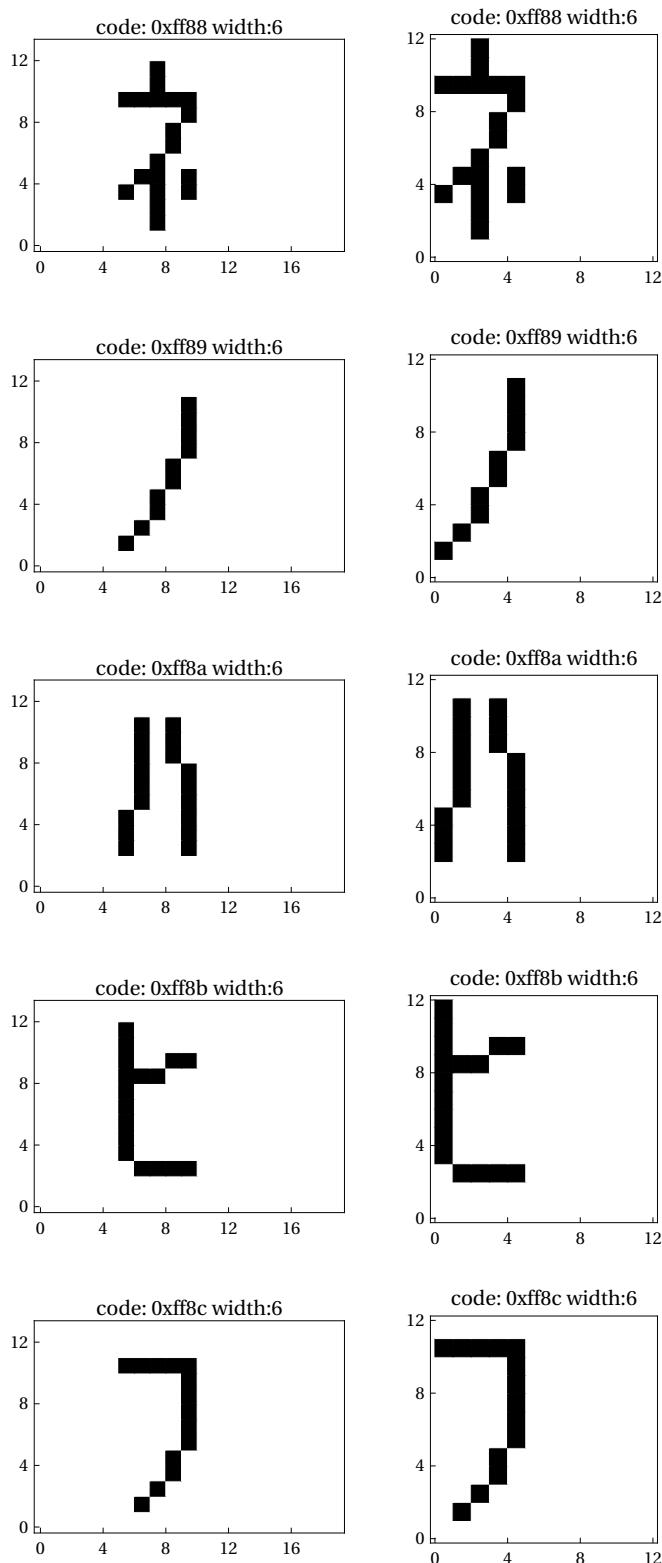


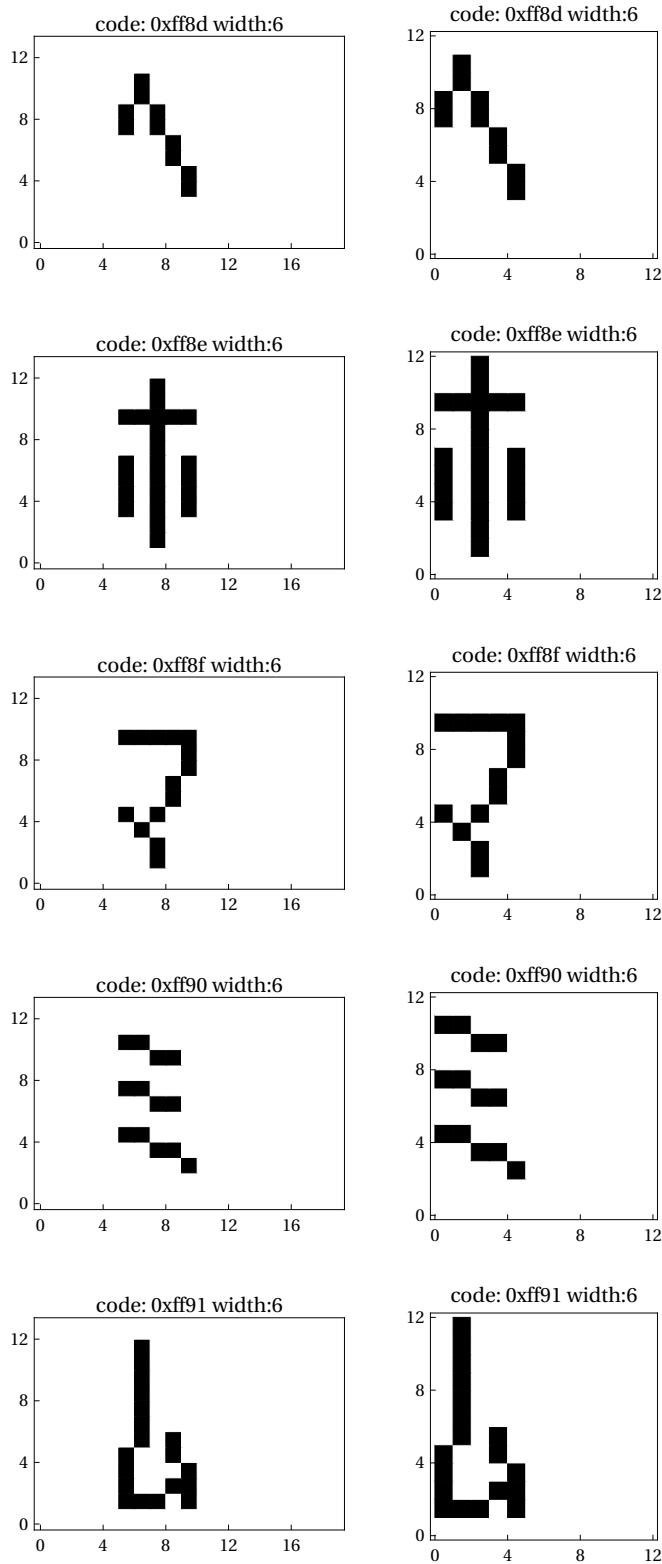


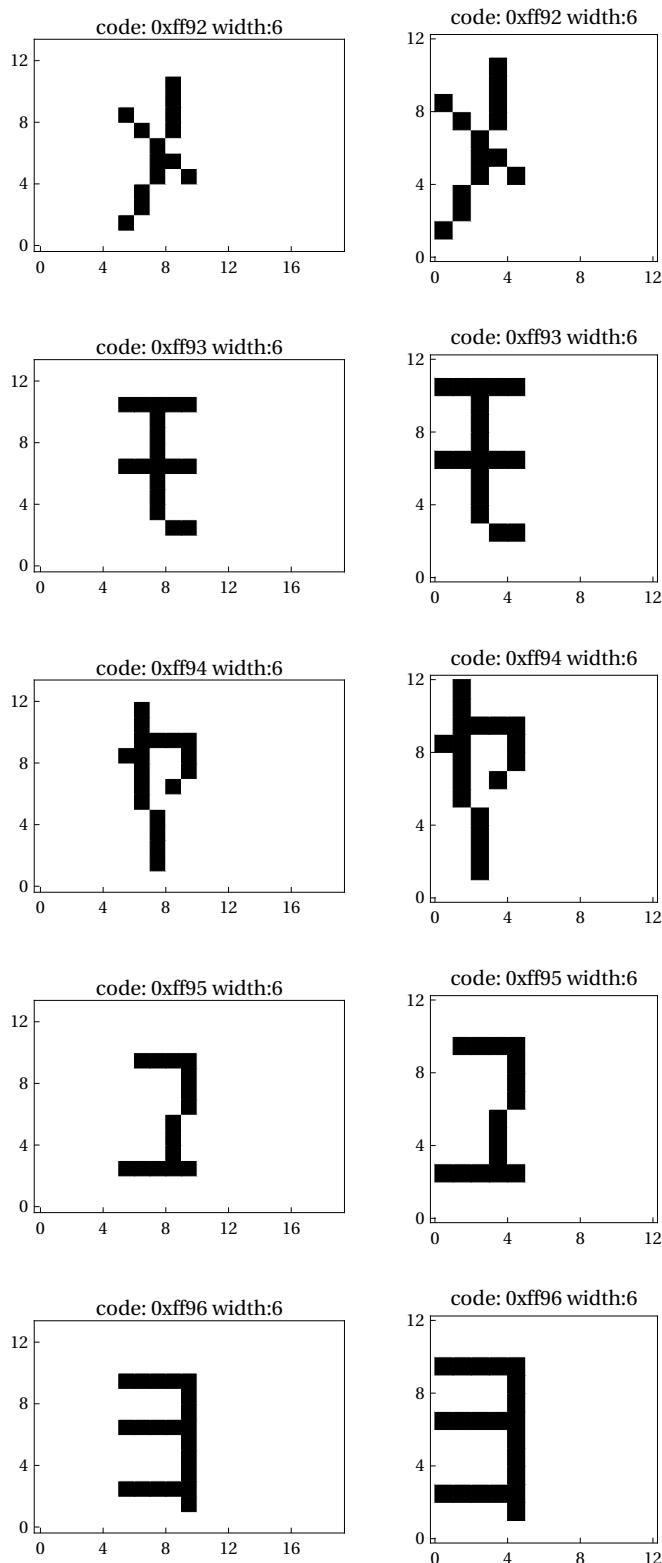


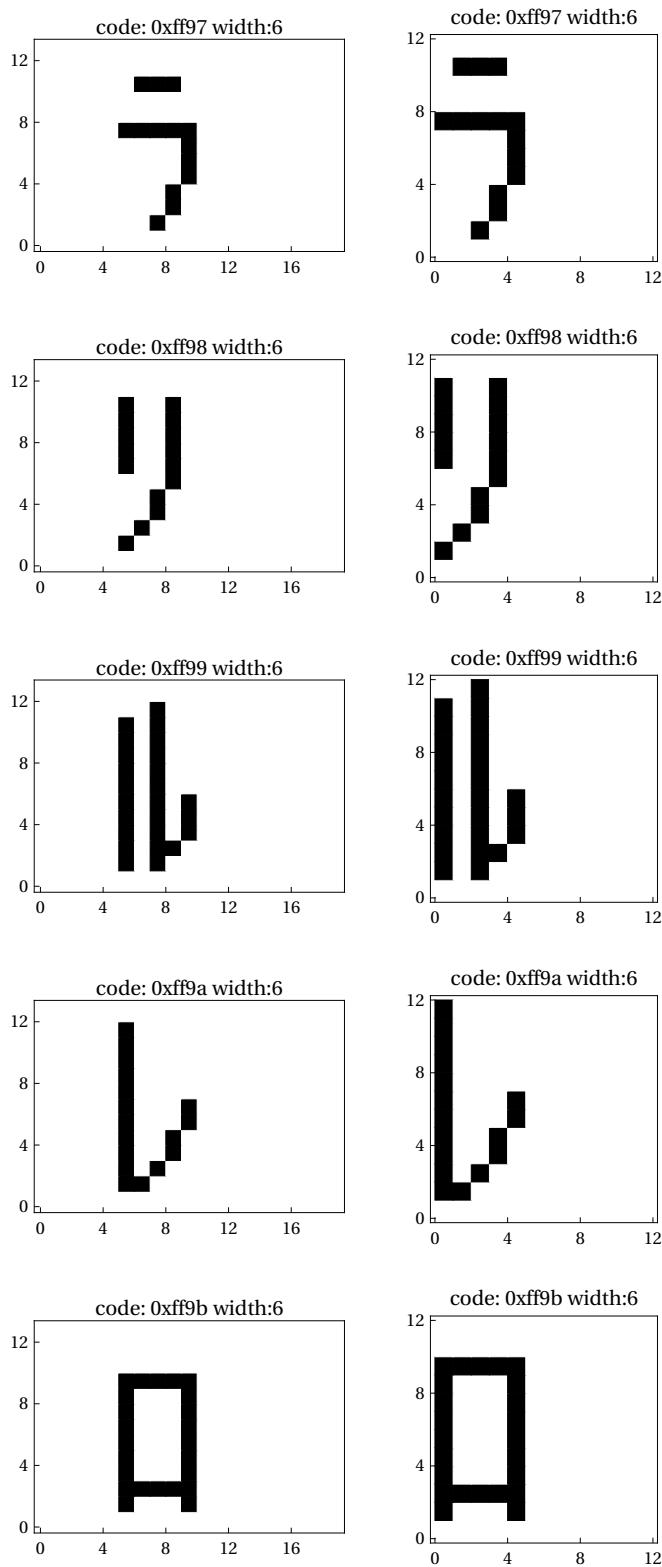


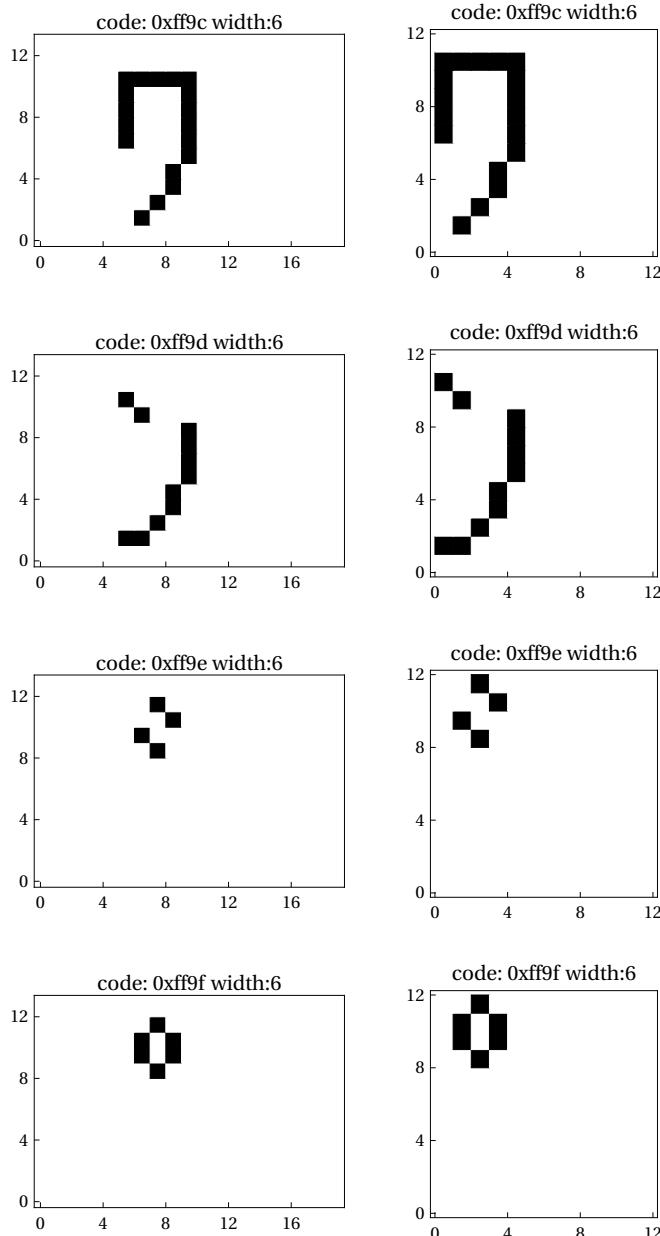






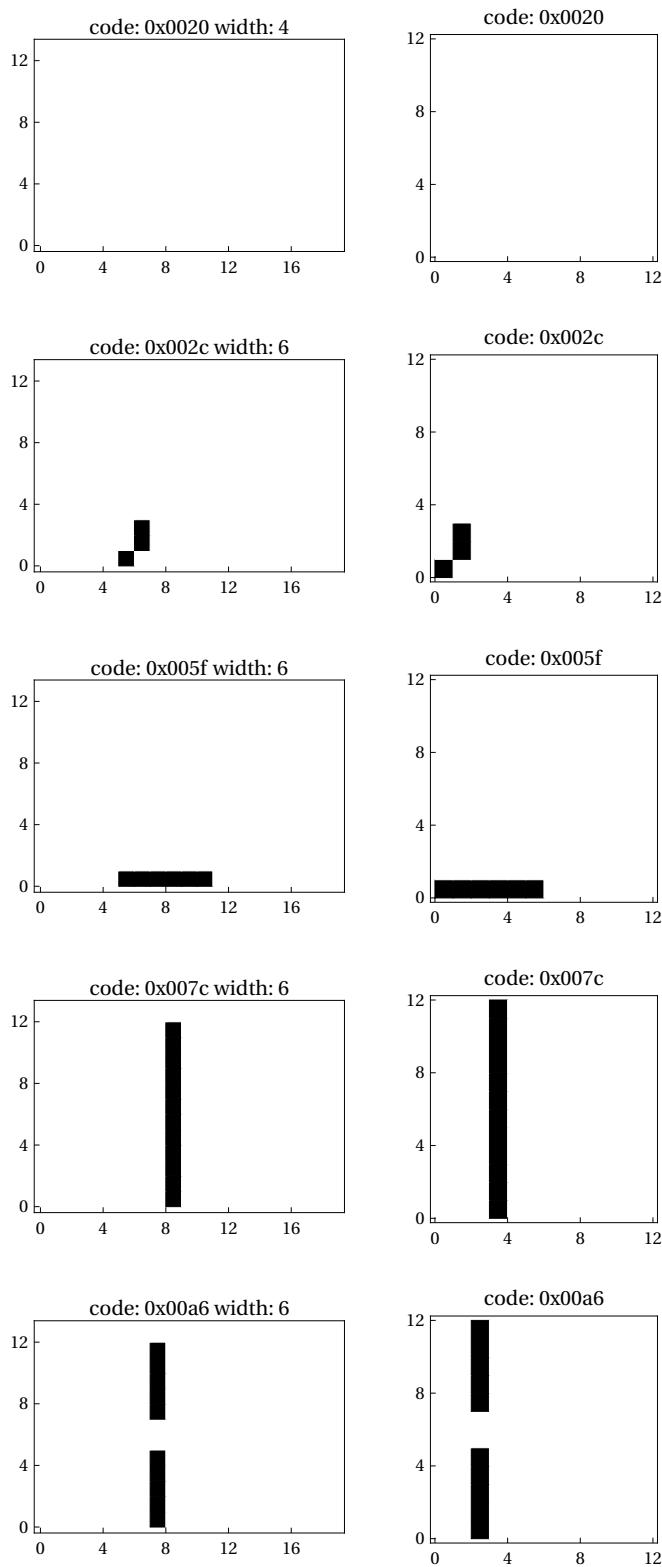


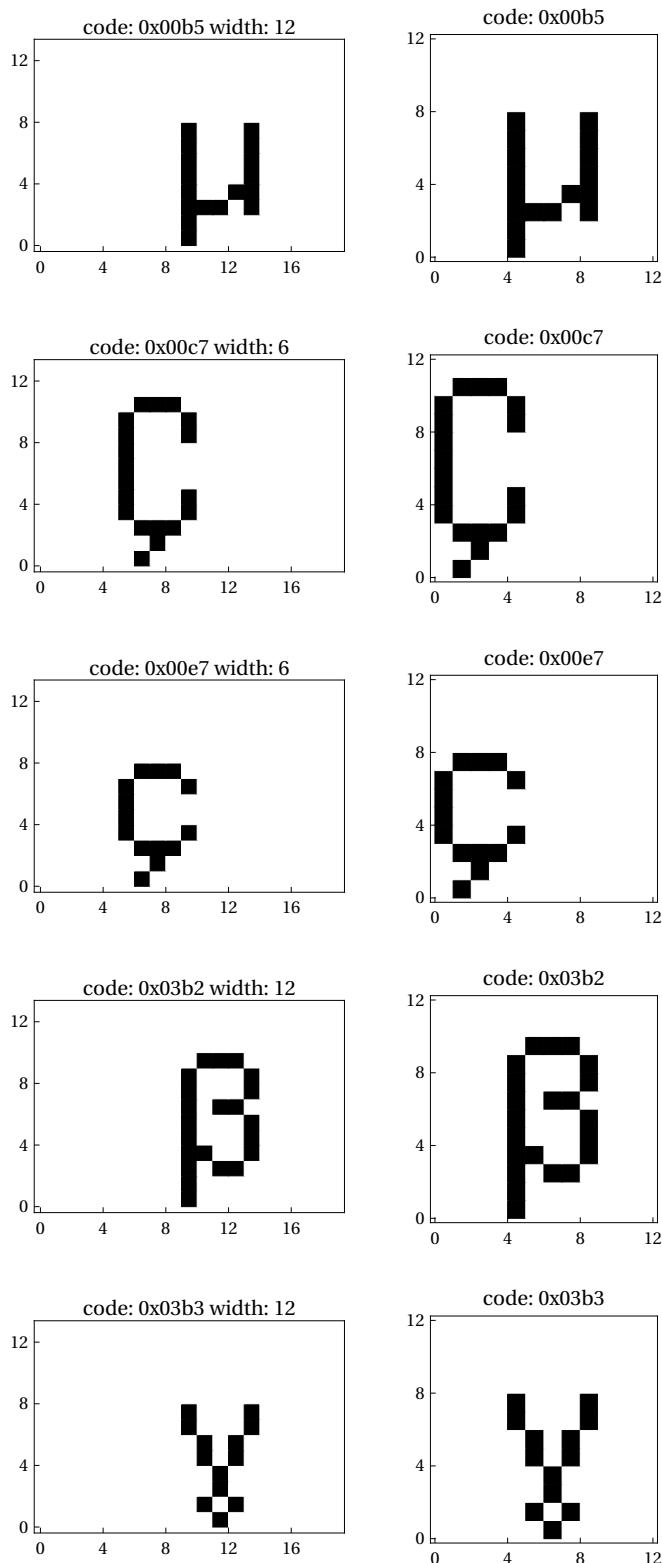


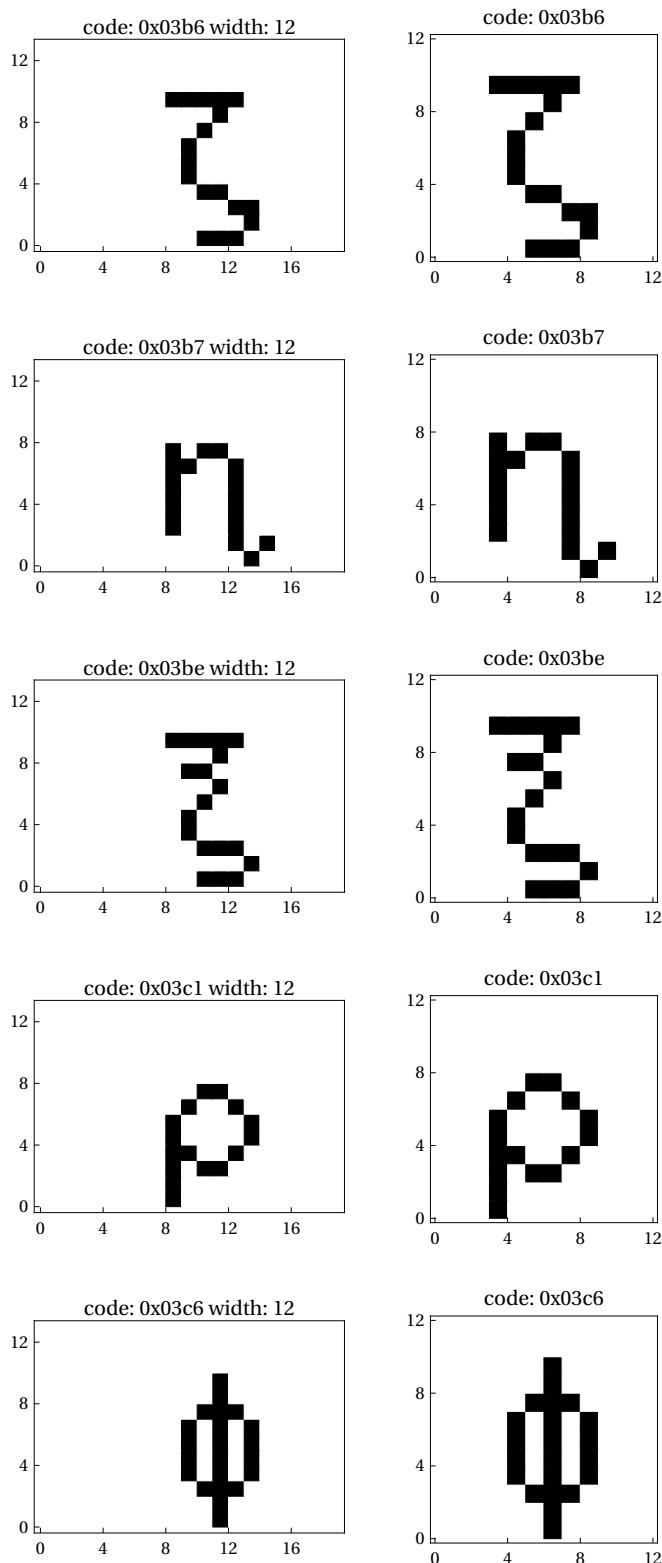


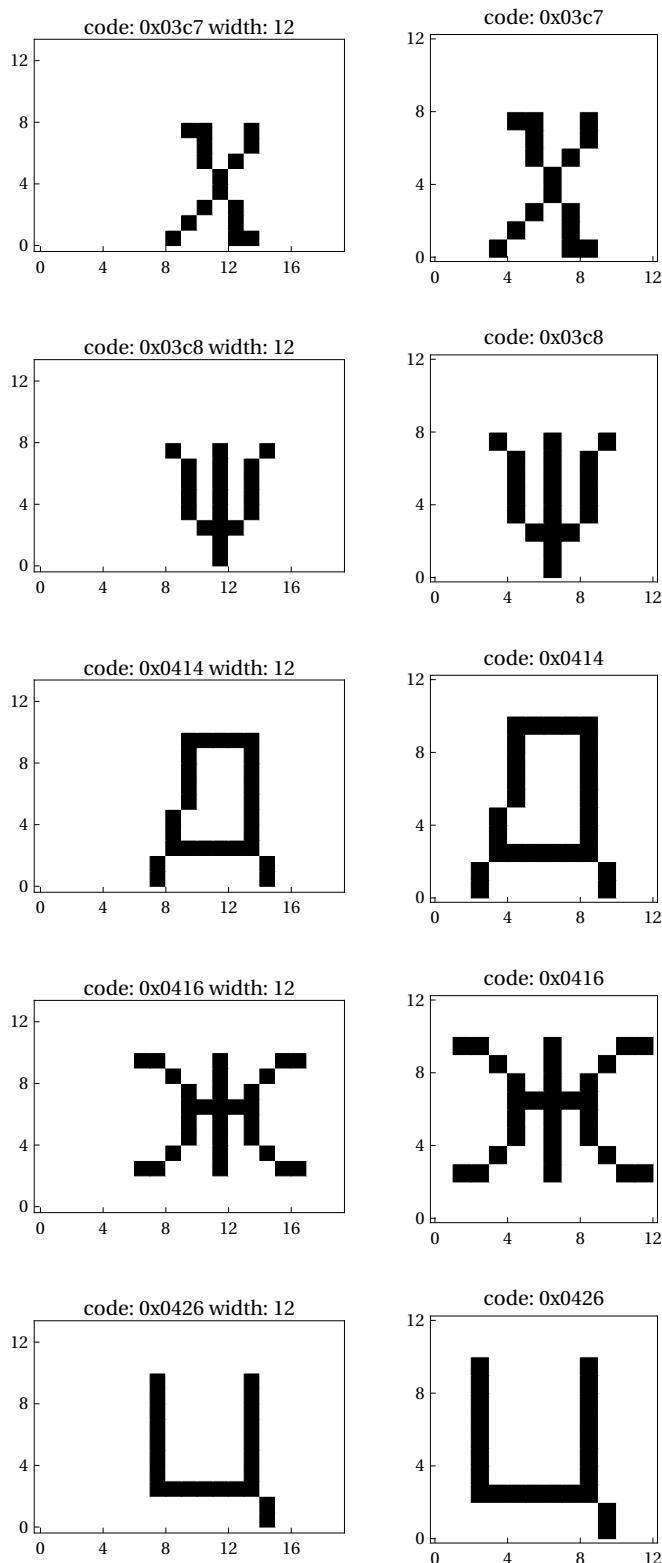
## ■ Bounding box outlier glyphs

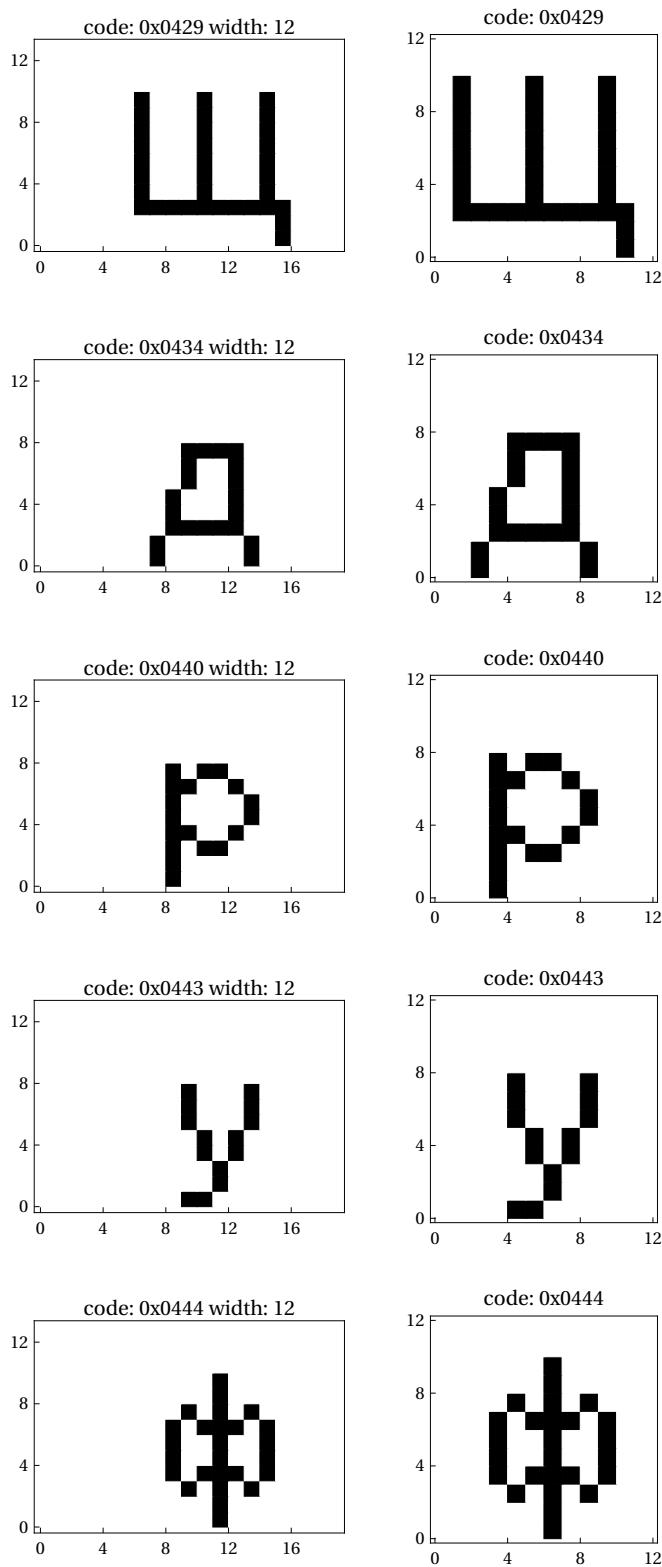
```
In[44]:= outliergraphics = MapThread[Graphics[Raster[Reverse[#1]],  
Frame -> True, AspectRatio -> Divide @@ Dimensions[#1],  
FrameTicks -> ({#, #, {}, {}} & [Range[0, 16, 4]]), PlotLabel -> ("code: 0x" <>  
IntegerString[ToExpression[#2], 16, 4] <> " width: " <> ToString[#3])] &,  
#\[Outliers] & /@ {extbitmaps, charcodes, widths}] ;  
  
In[45]:= outliertruncgraphics =  
MapThread[Graphics[Raster[Reverse[#1]], Frame -> True, AspectRatio ->  
Divide @@ Dimensions[#1], FrameTicks -> ({#, #, {}, {}} & [Range[0, 16, 4]]),  
PlotLabel -> ("code: 0x" <> IntegerString[ToExpression[#2], 16, 4])] &,  
#\[Outliers] & /@ {trbitmaps, charcodes, widths}] ;  
  
In[46]:= MapThread[Print[GraphicsGrid[{{##}}]] &, {outliergraphics, outliertruncgraphics}] ;
```

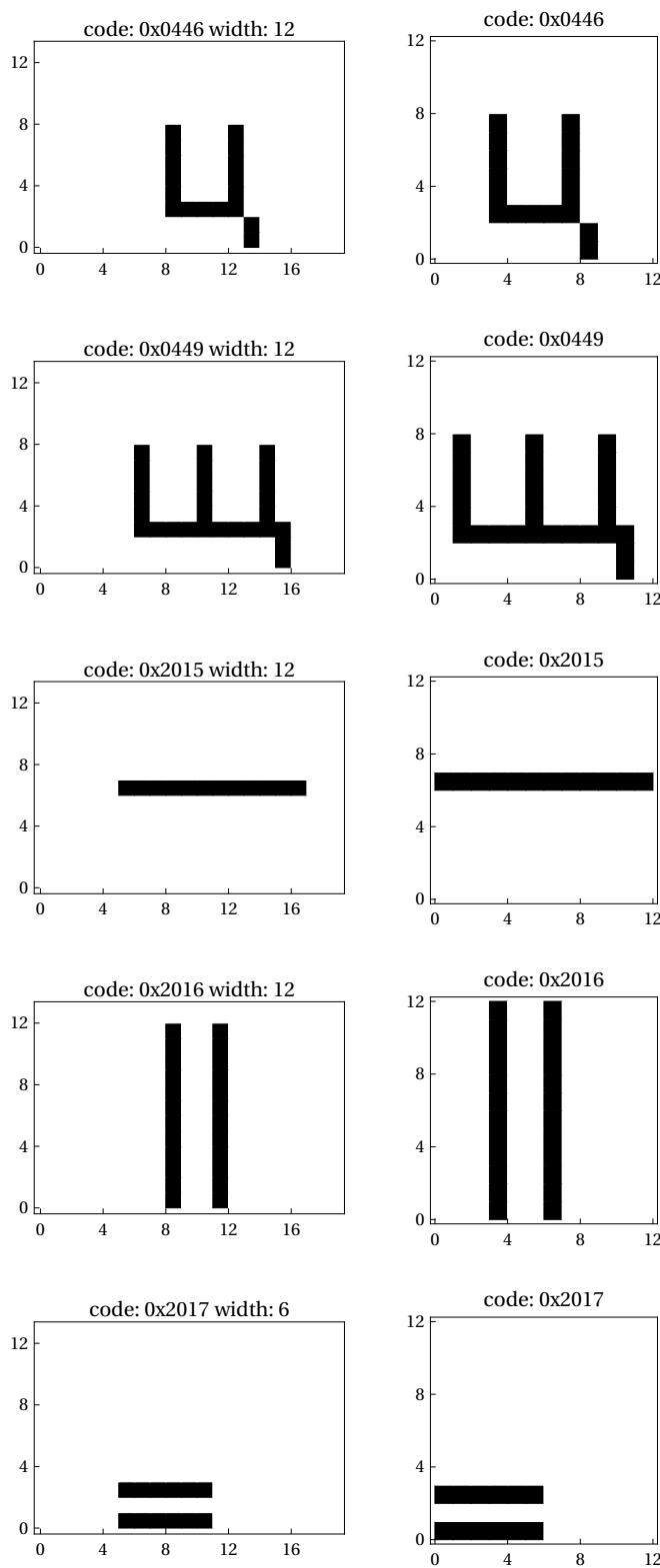


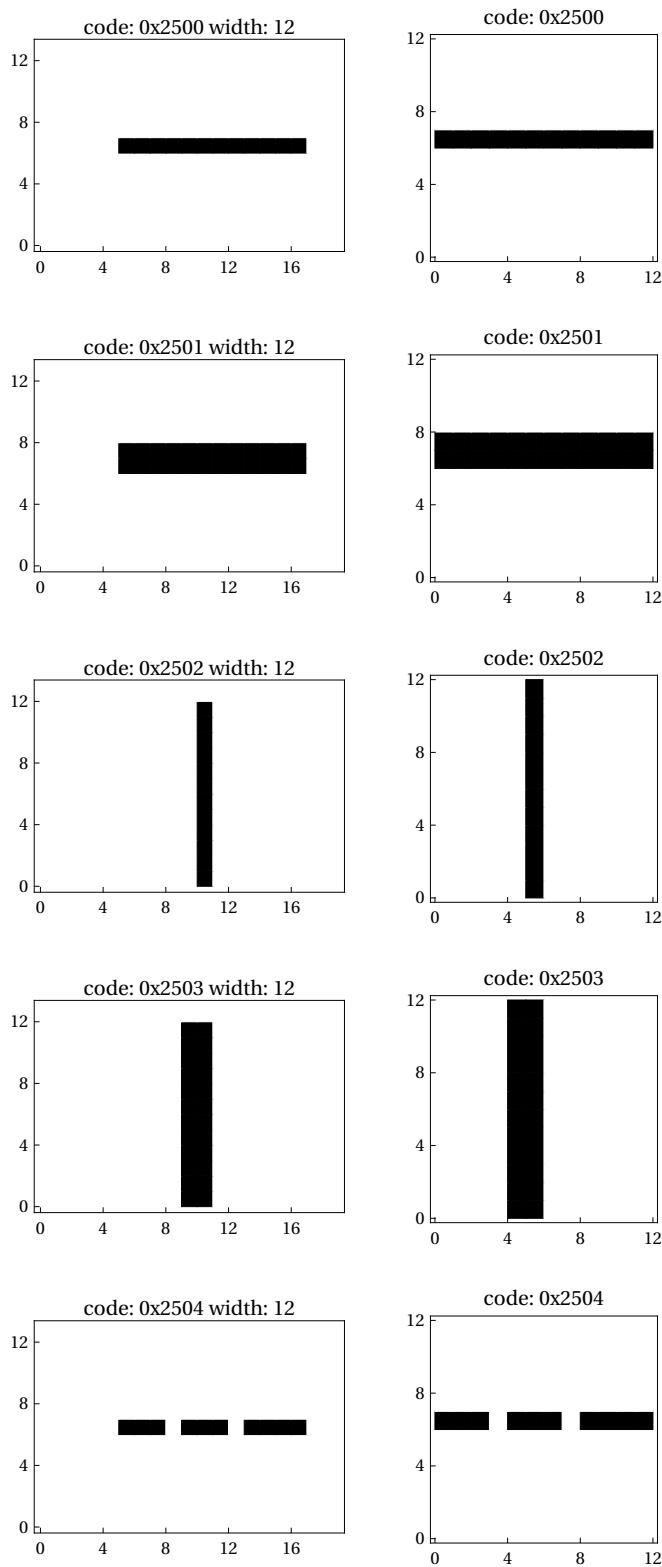


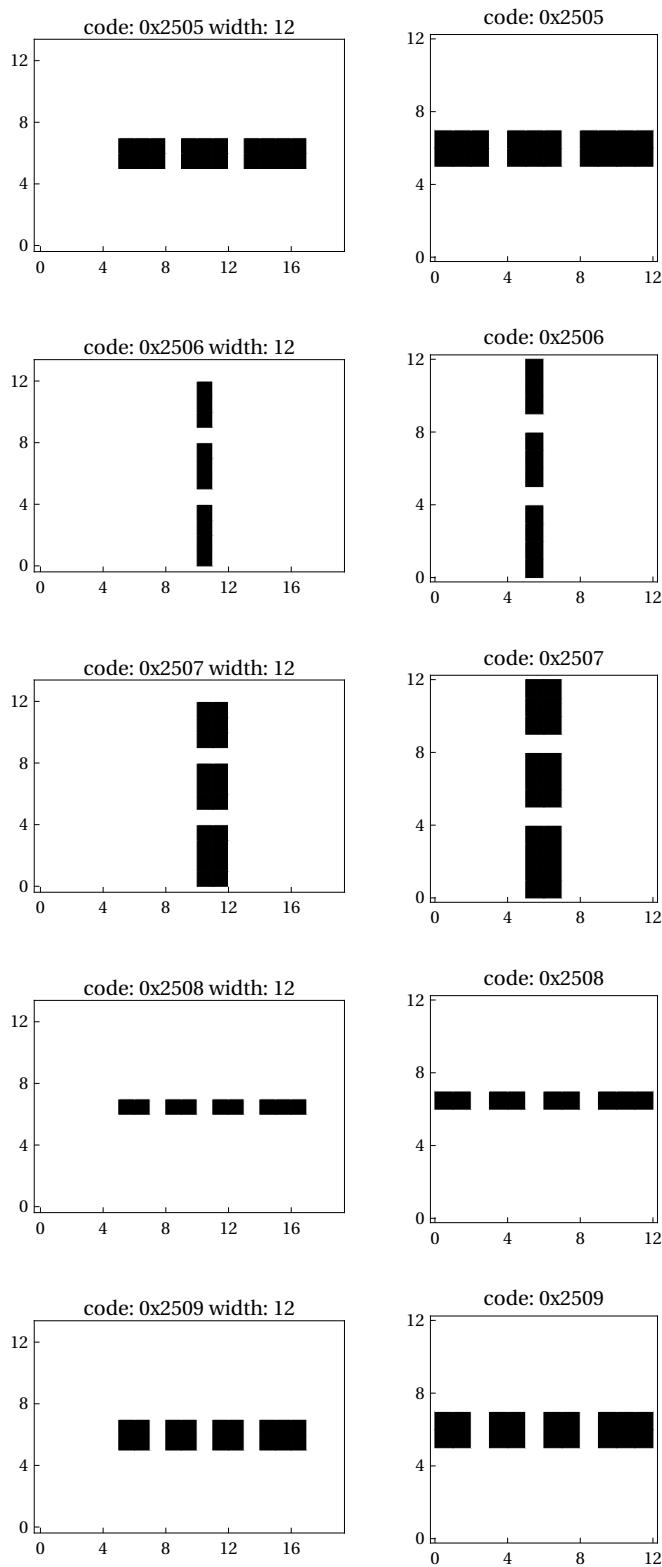


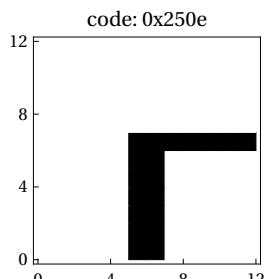
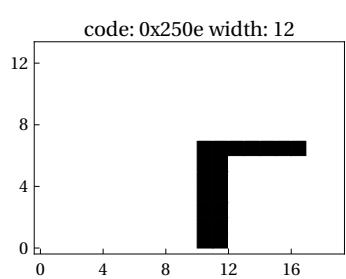
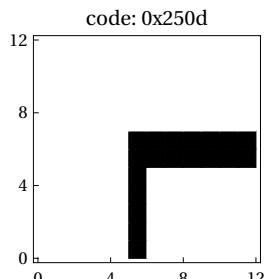
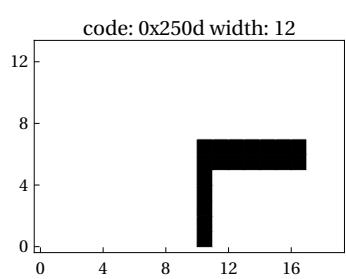
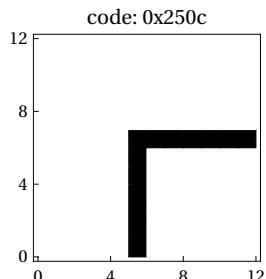
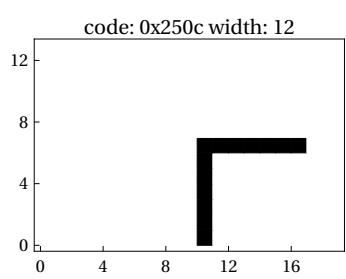
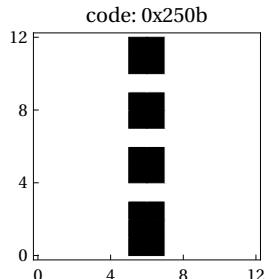
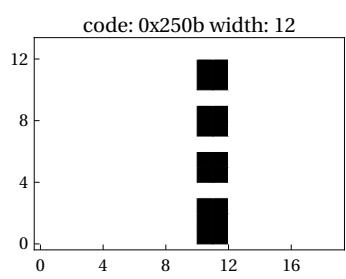
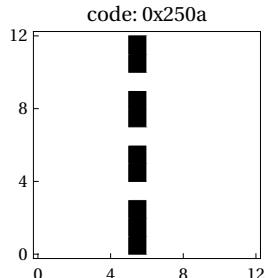
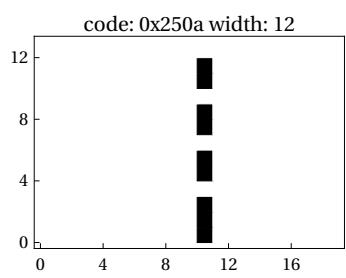


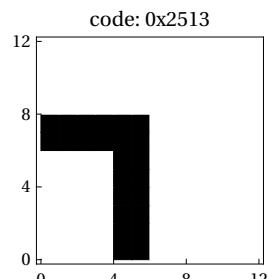
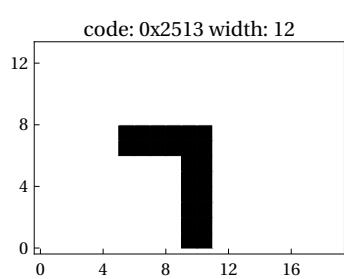
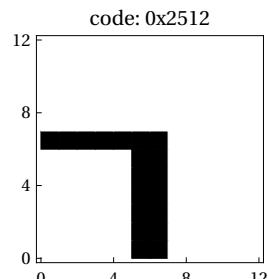
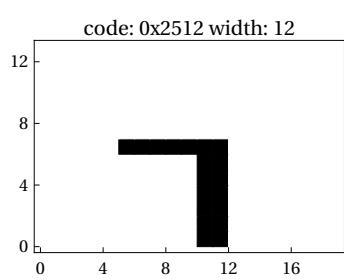
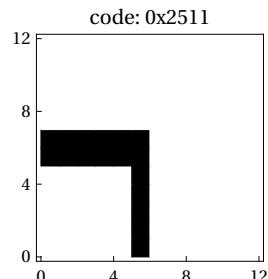
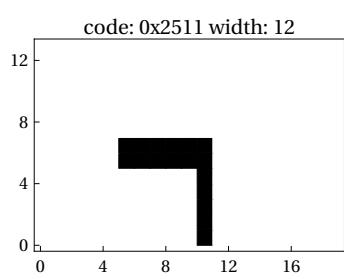
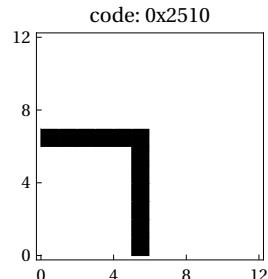
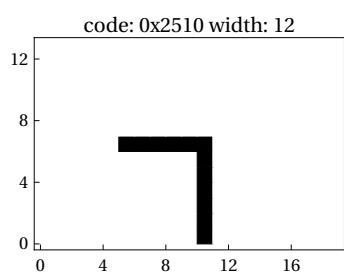
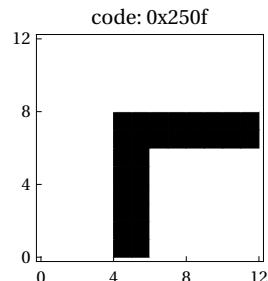
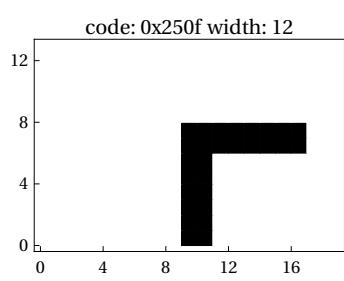


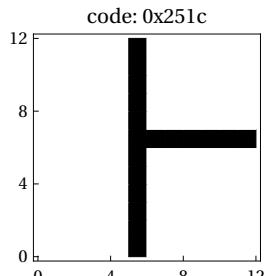
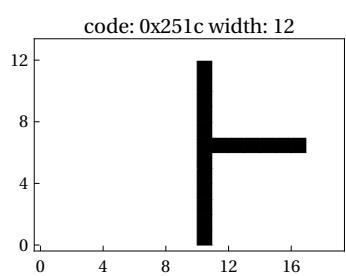
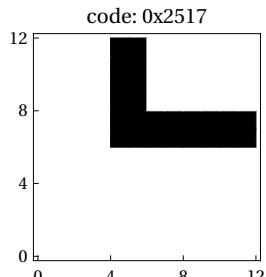
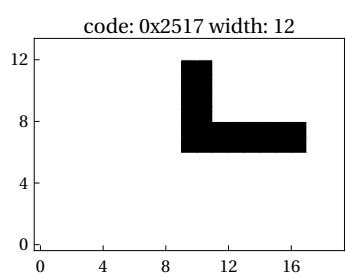
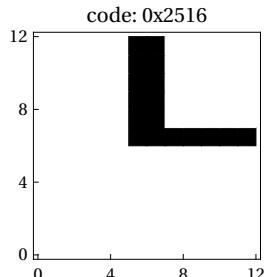
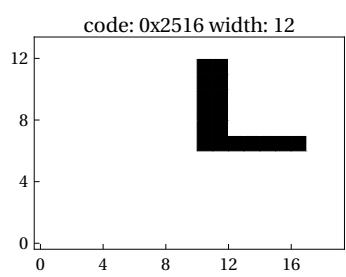
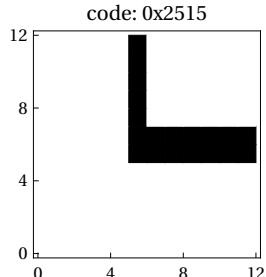
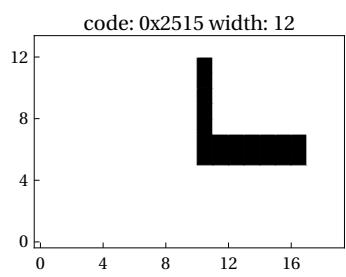
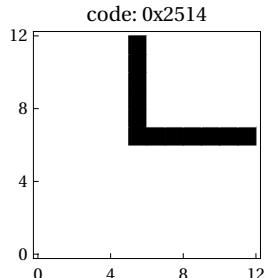
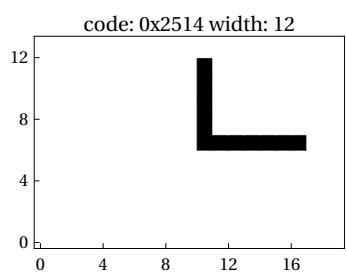


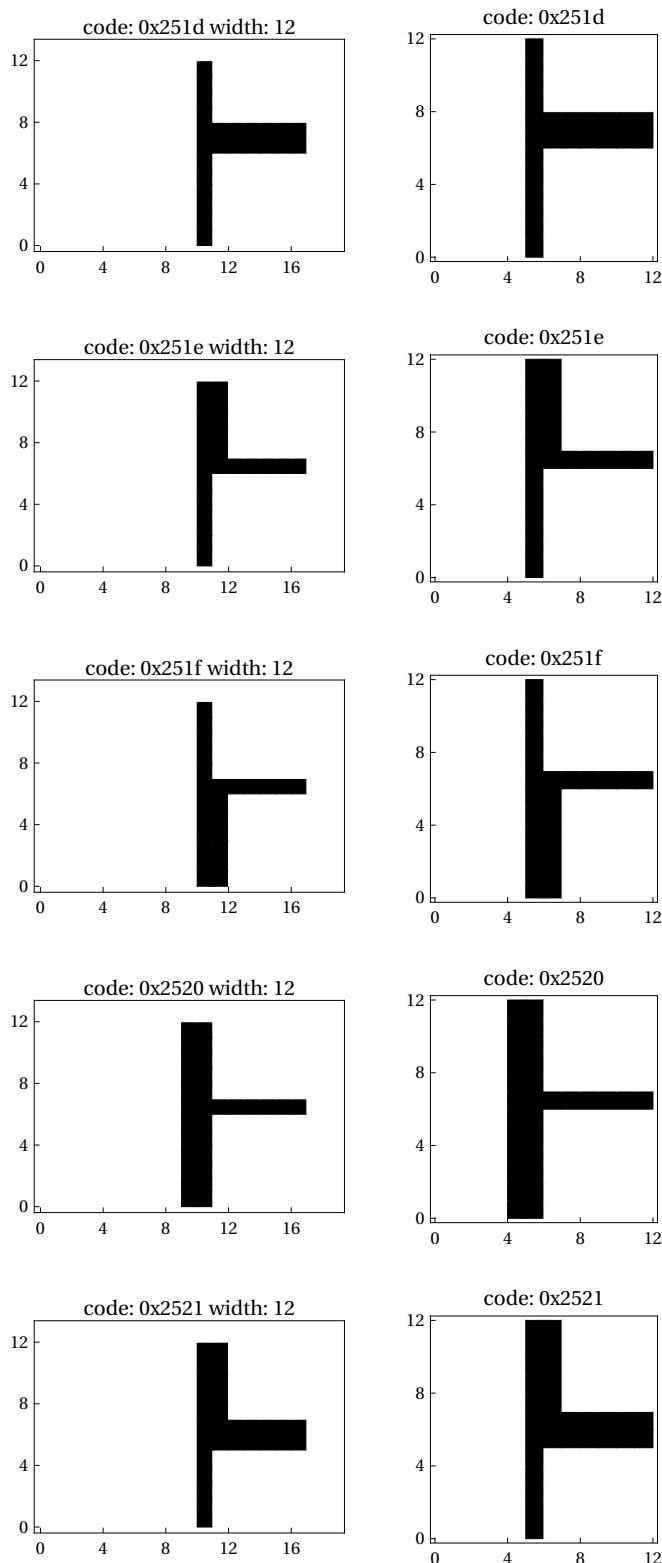


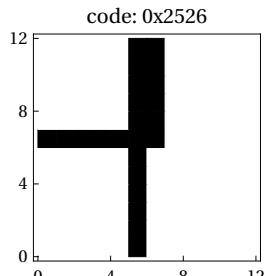
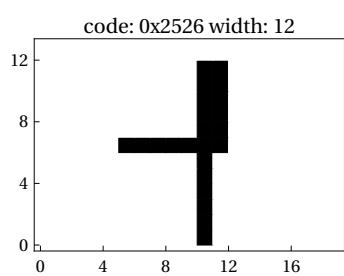
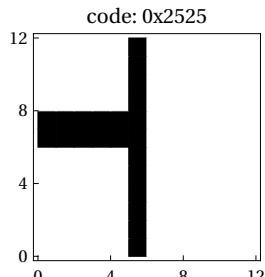
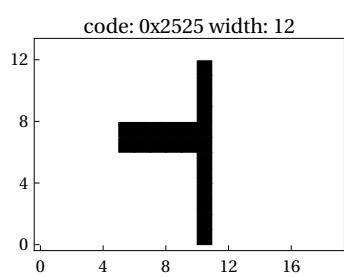
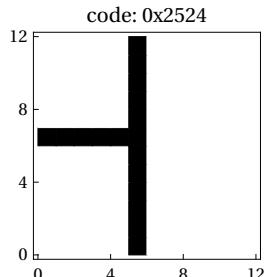
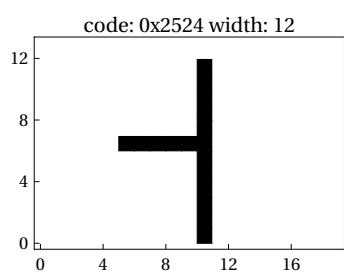
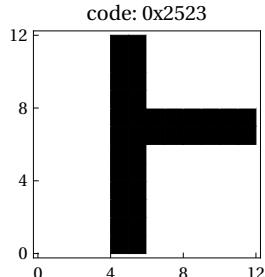
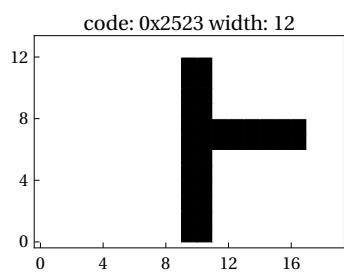
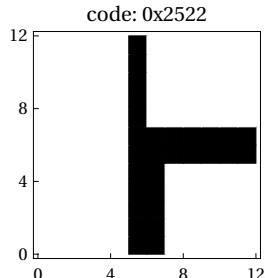
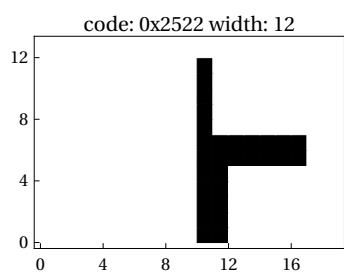


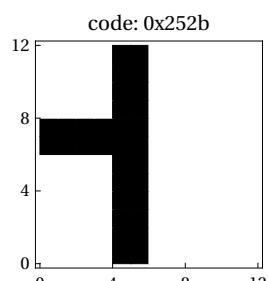
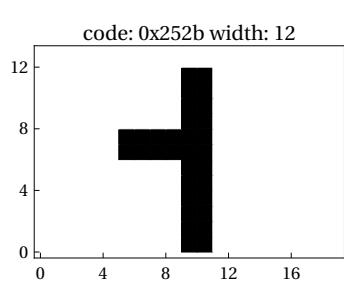
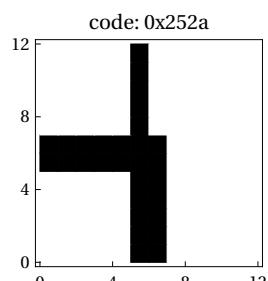
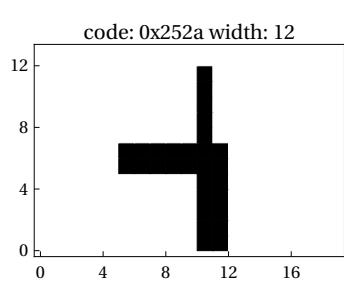
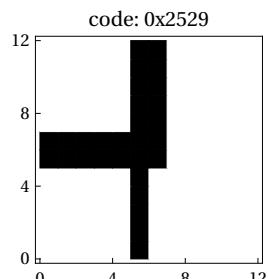
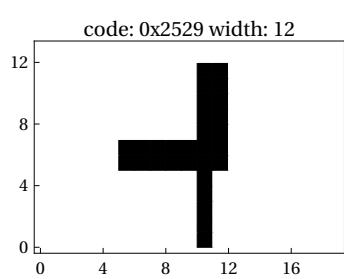
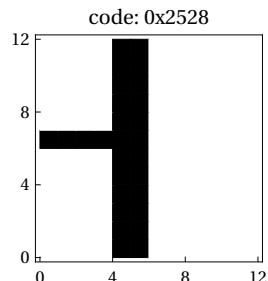
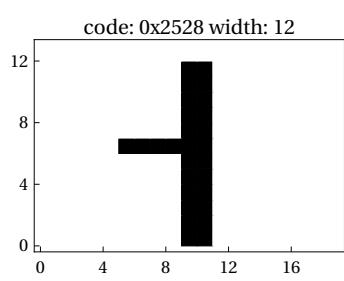
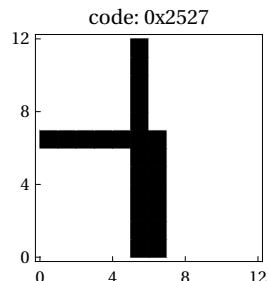
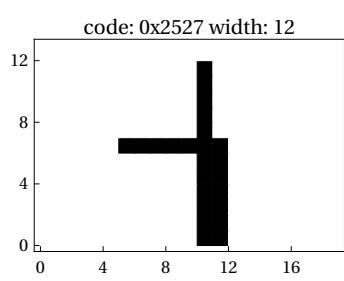


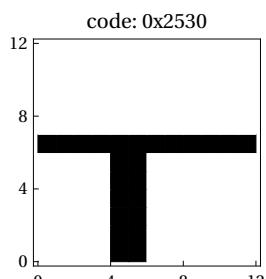
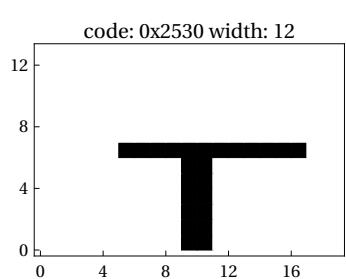
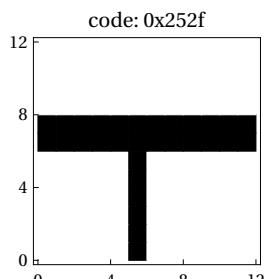
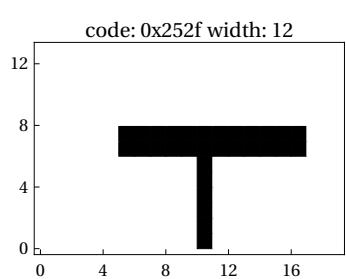
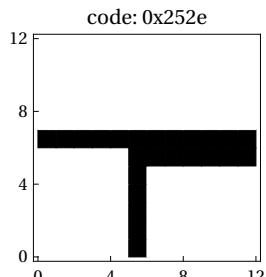
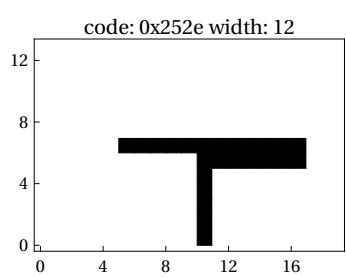
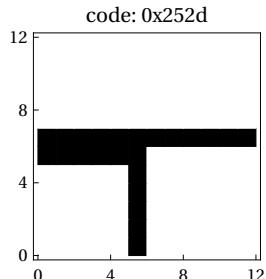
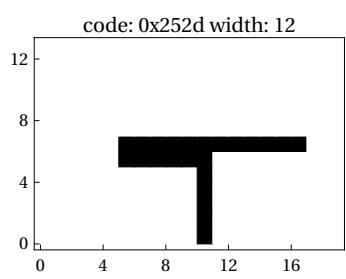
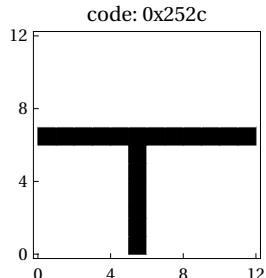
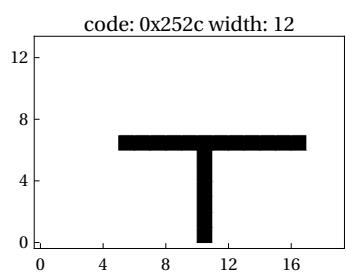


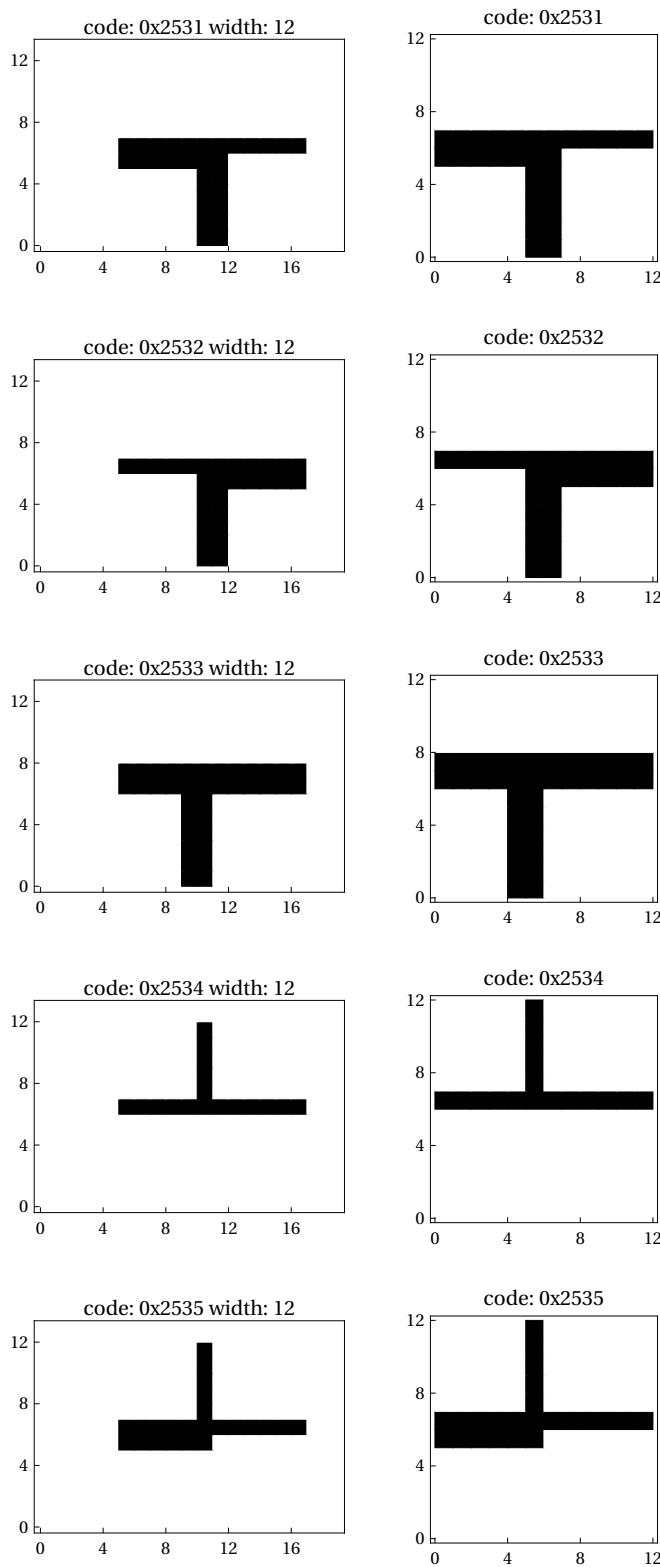


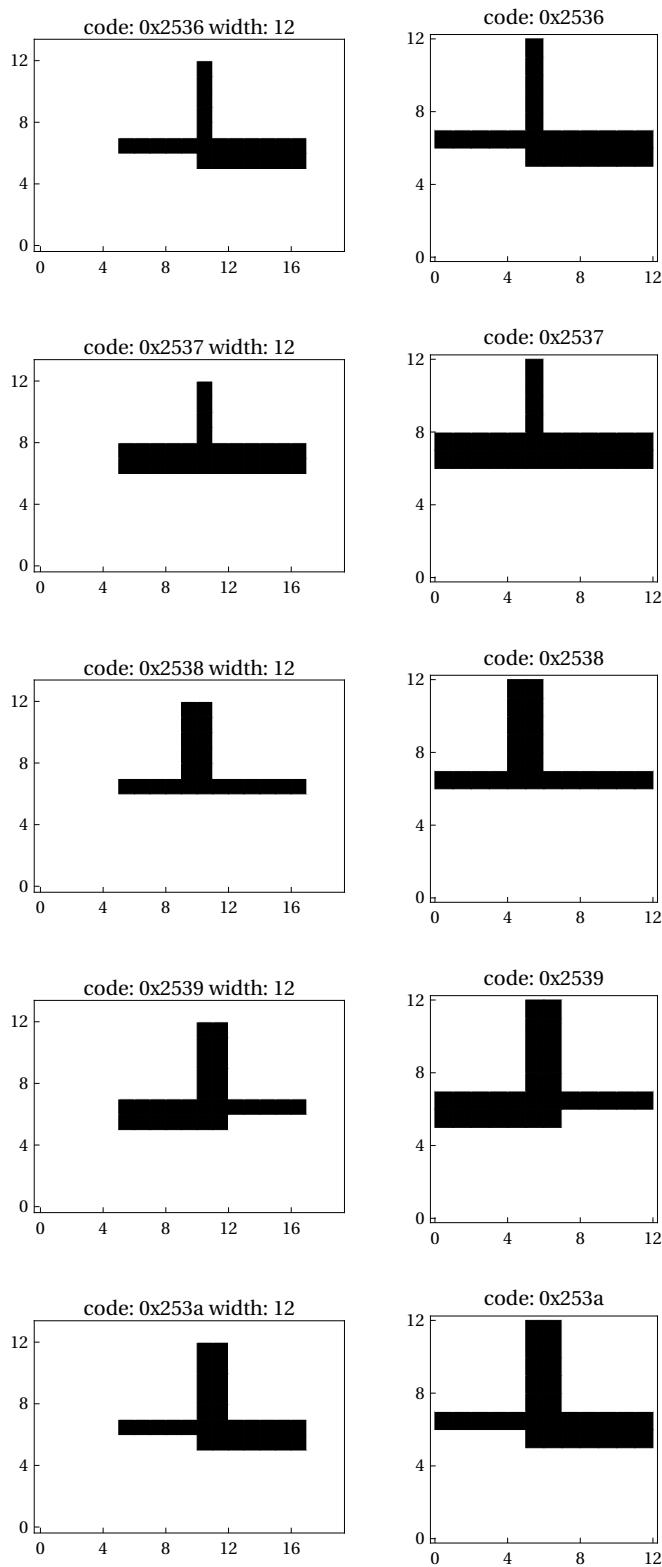


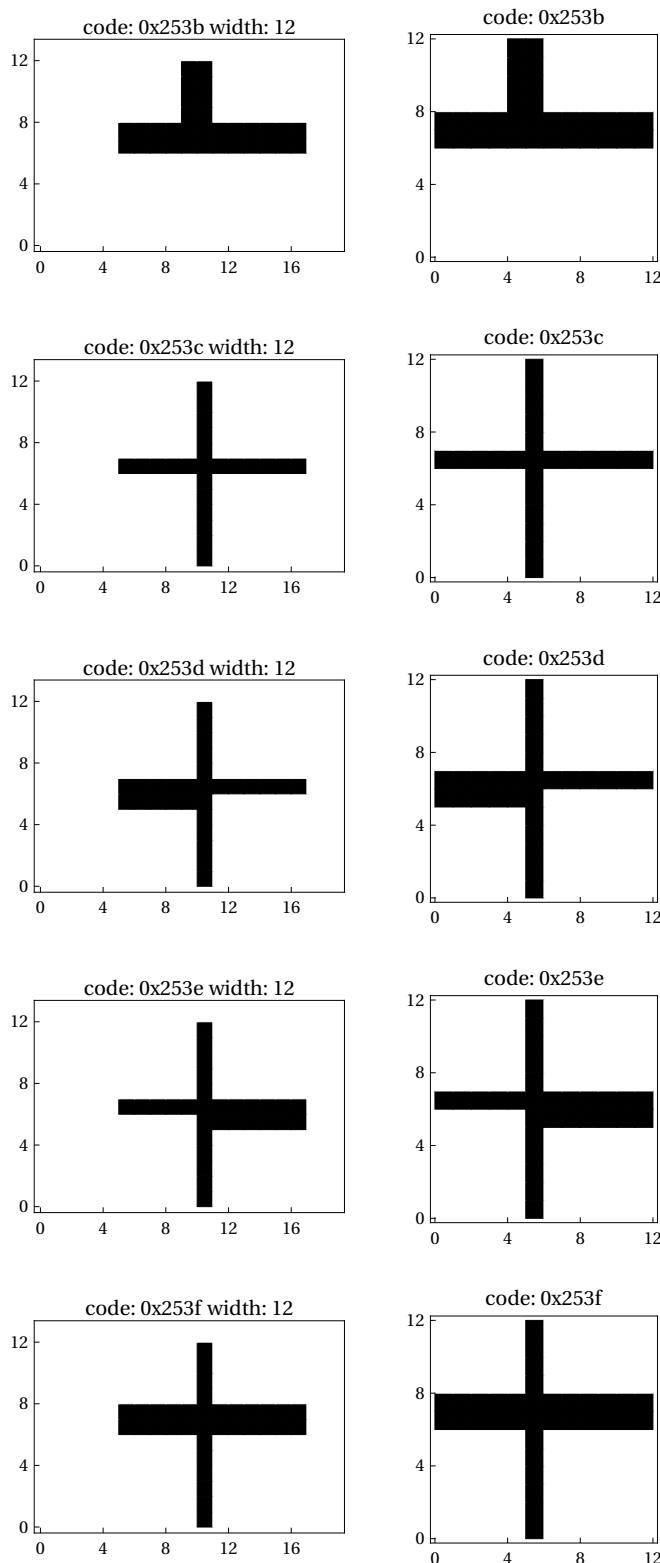


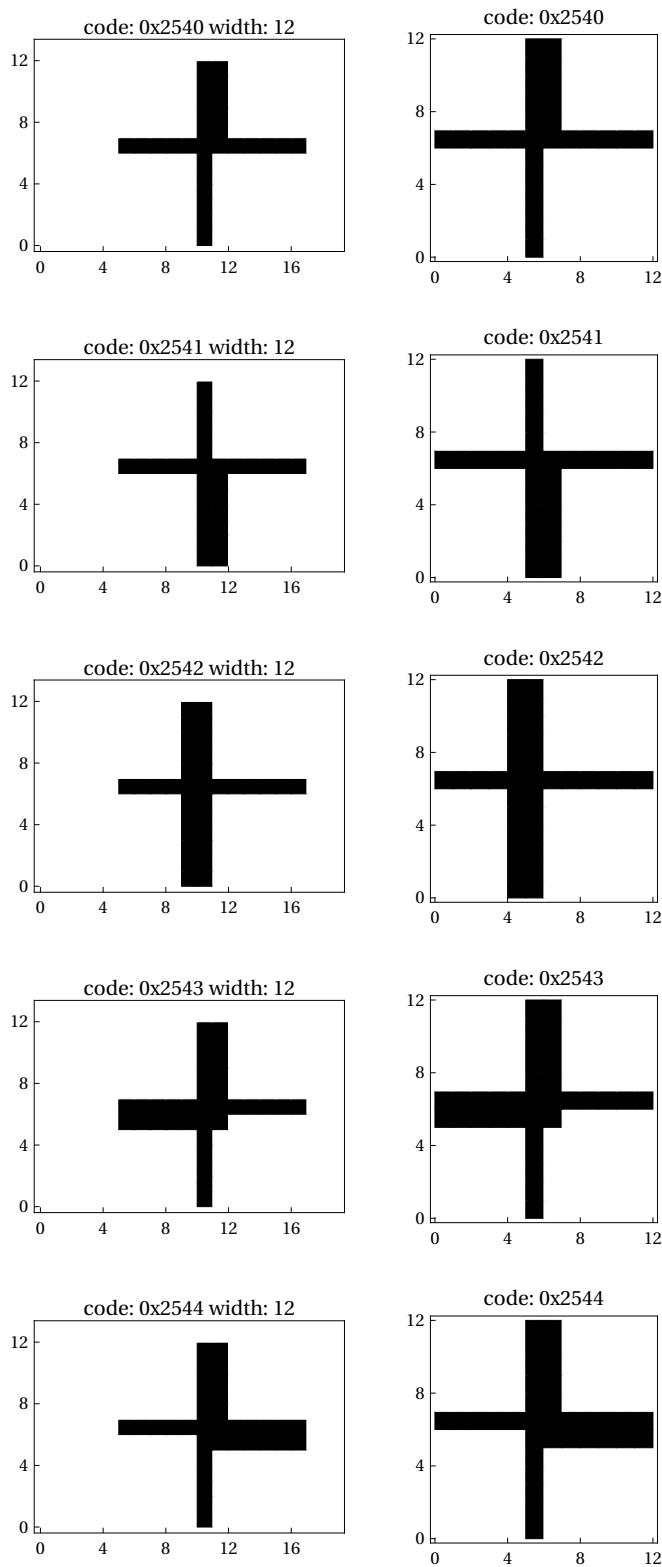


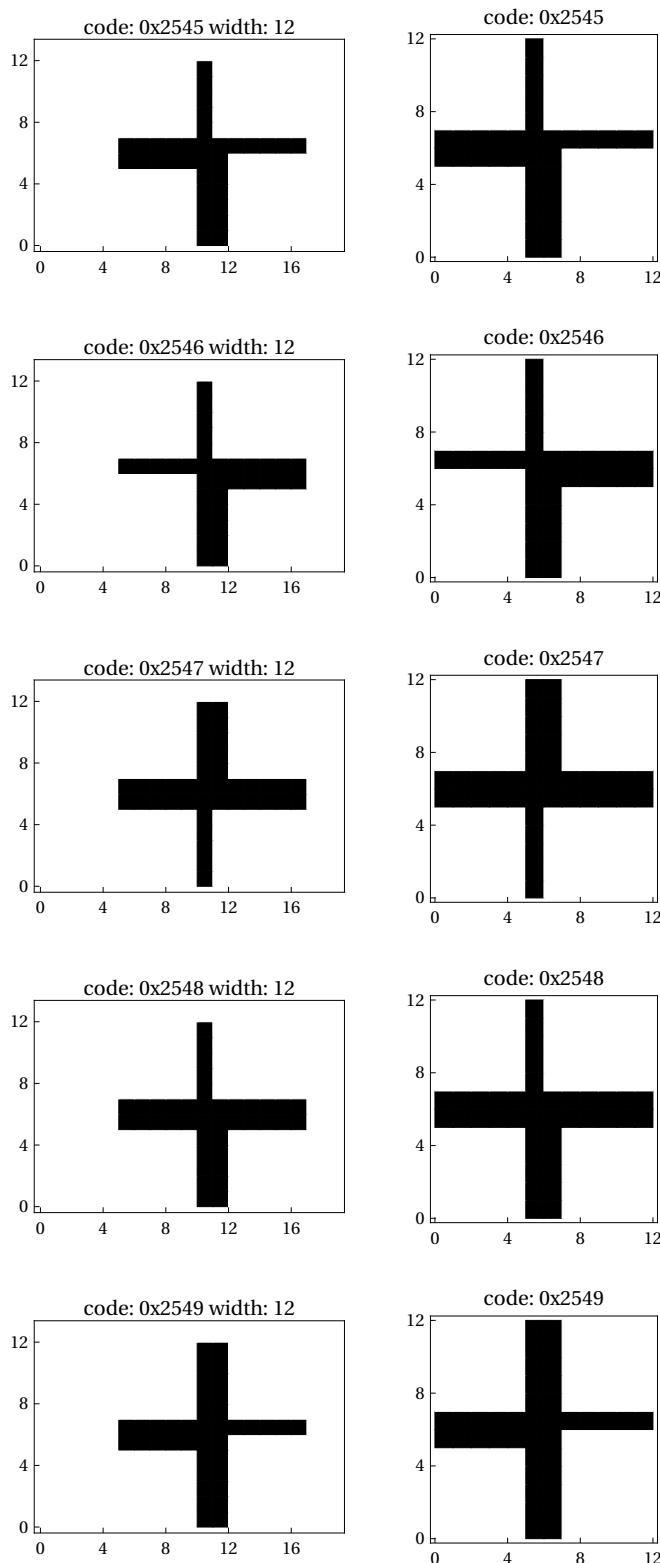


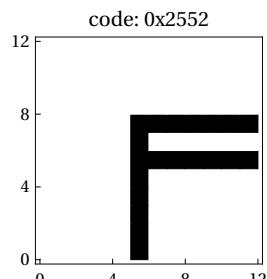
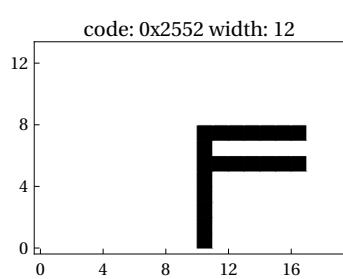
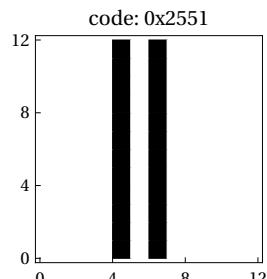
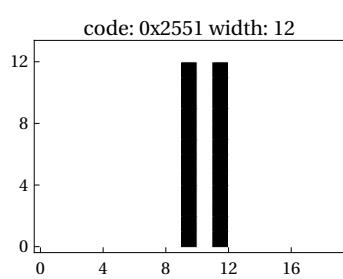
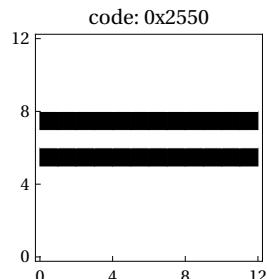
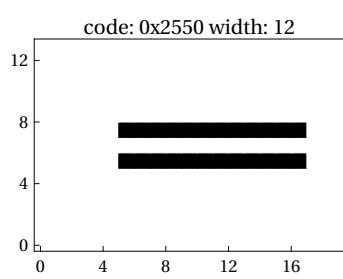
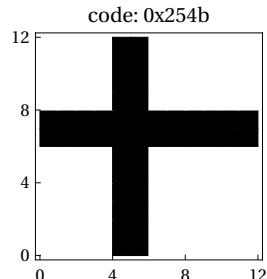
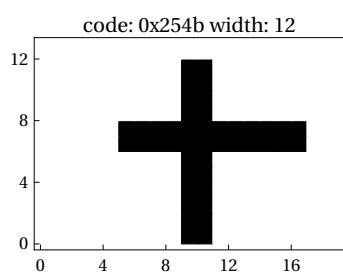
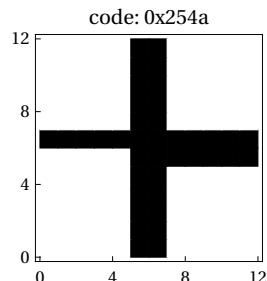
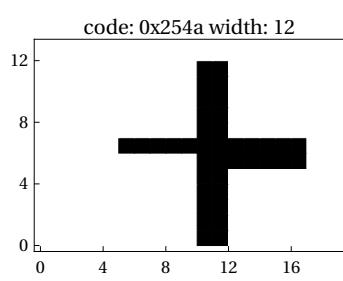


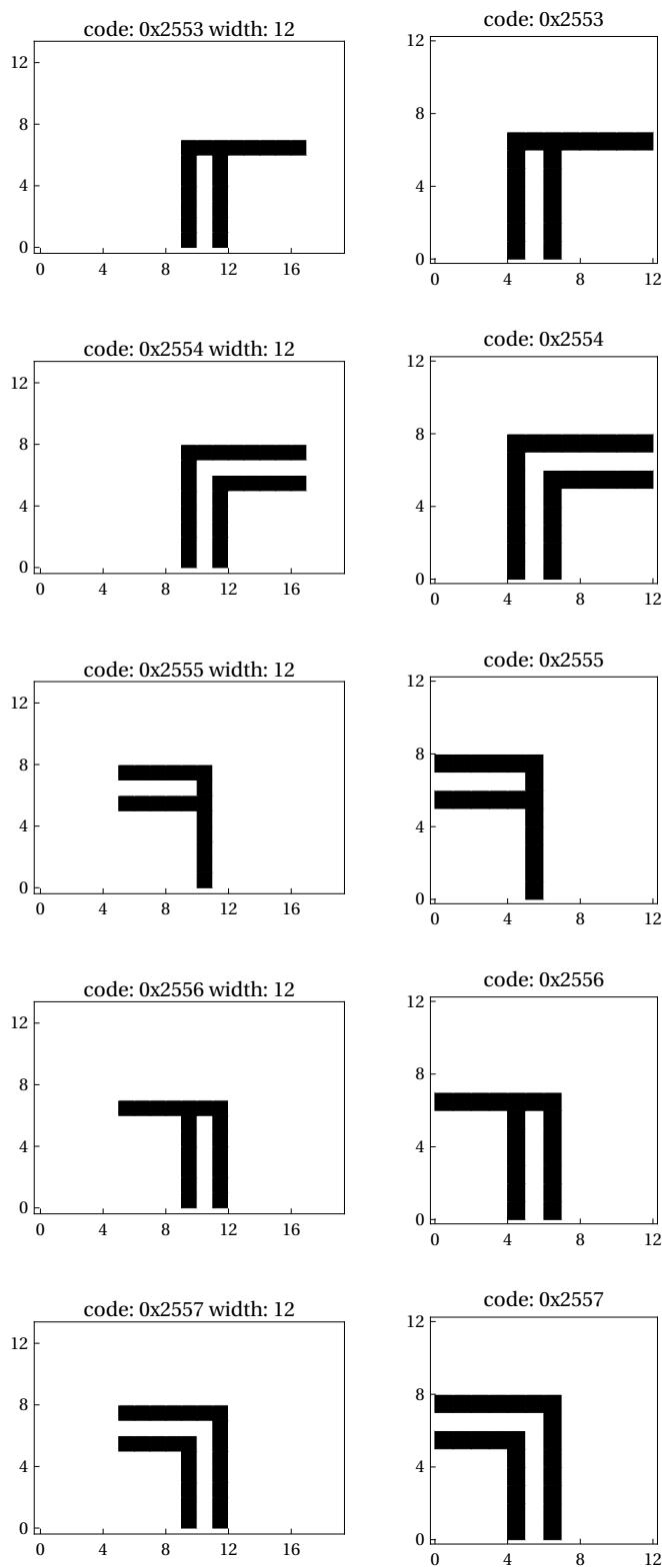


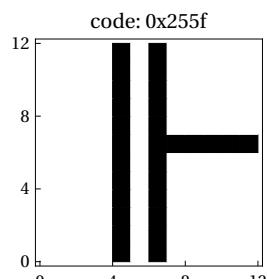
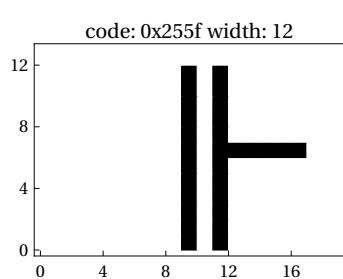
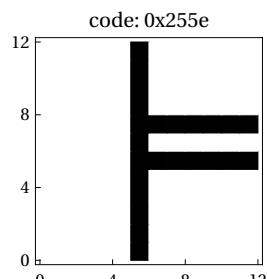
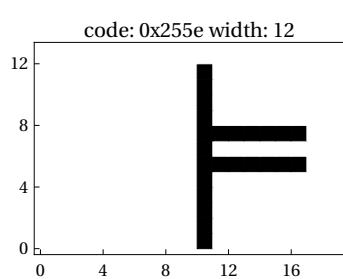
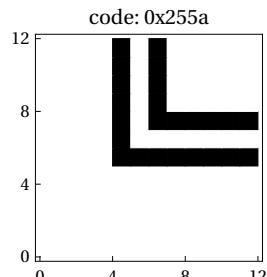
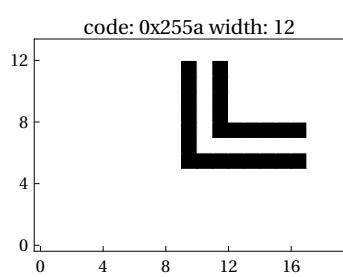
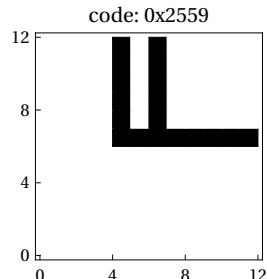
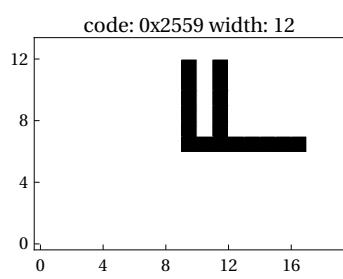
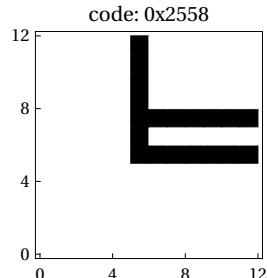
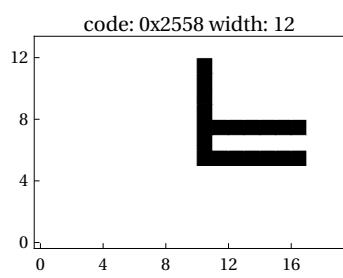


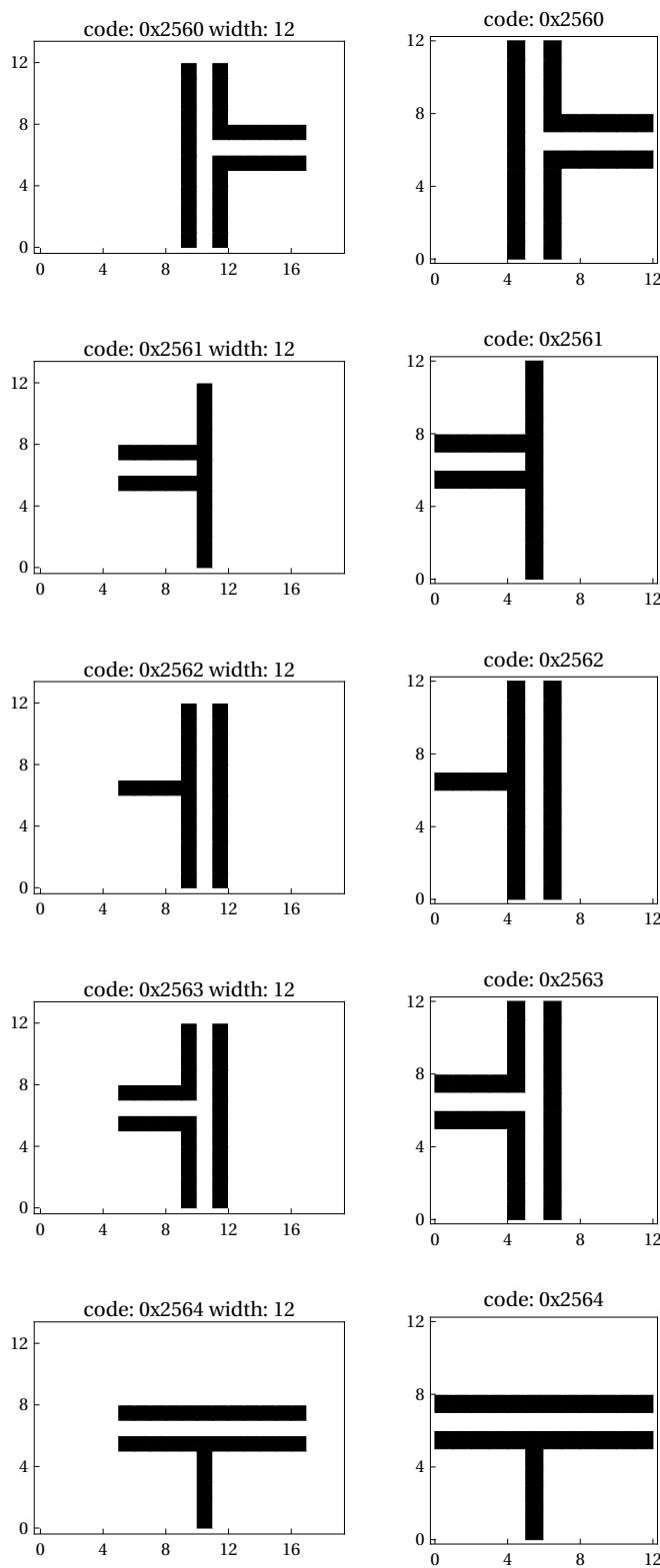


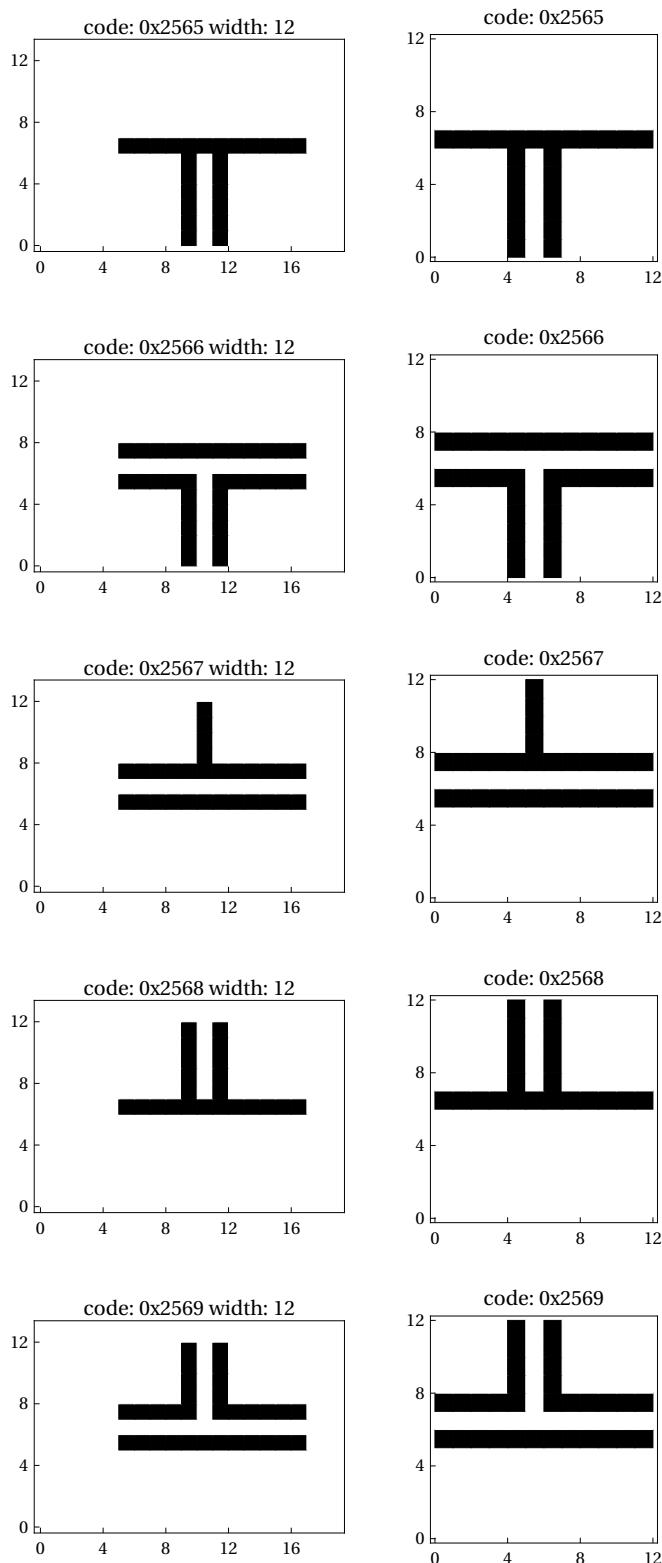


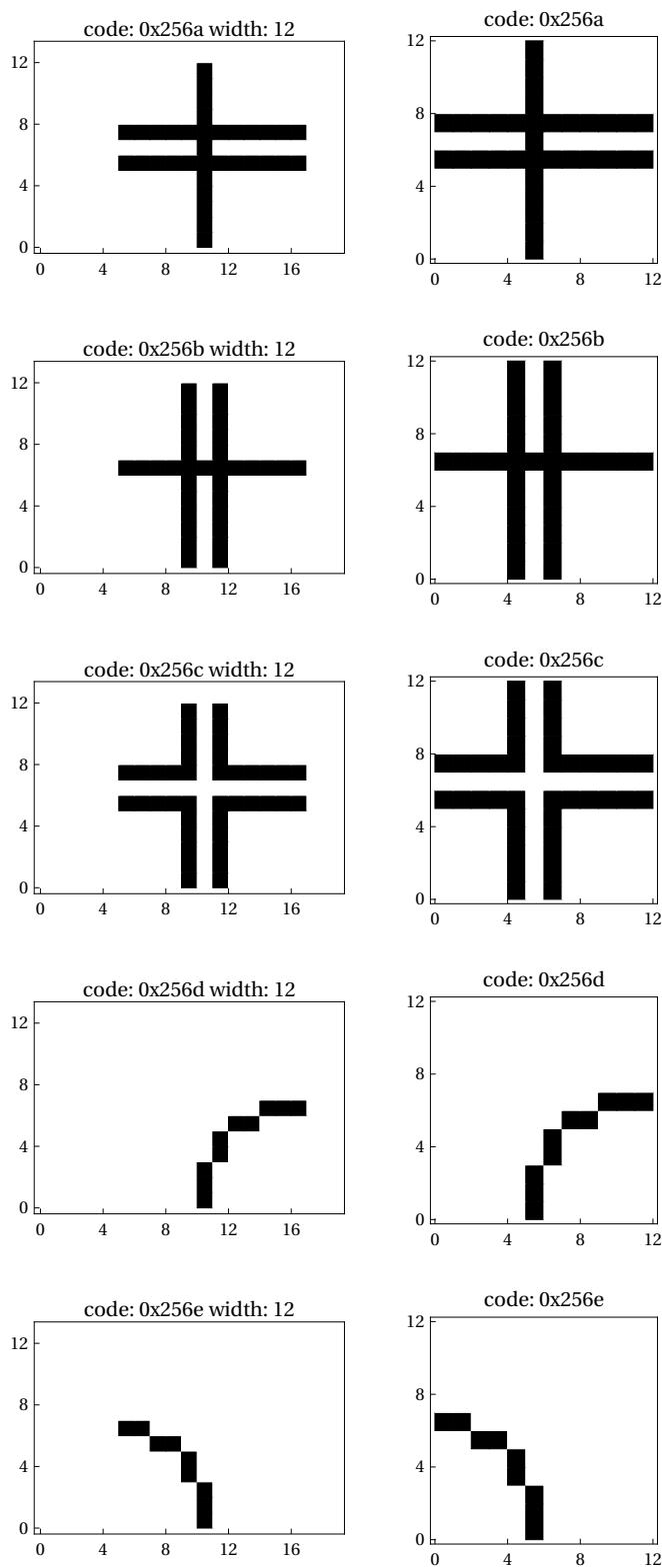


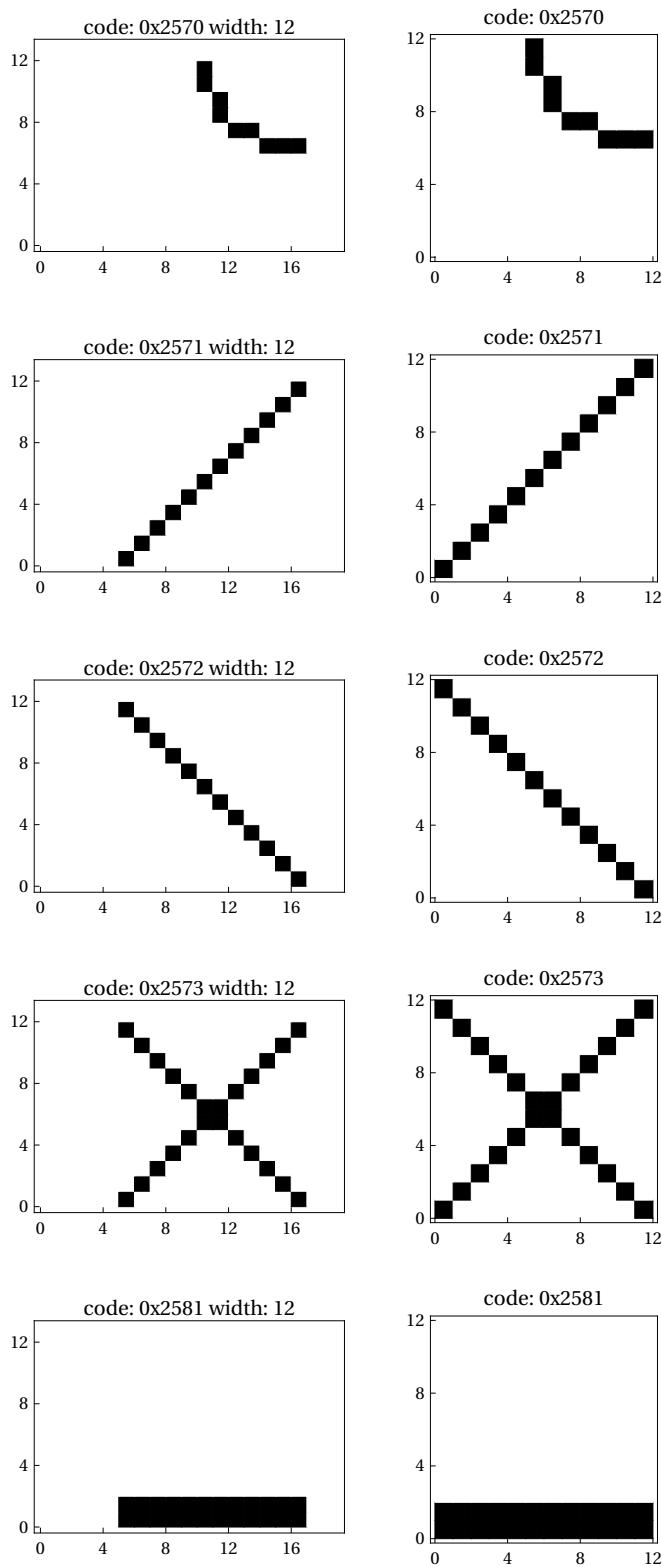


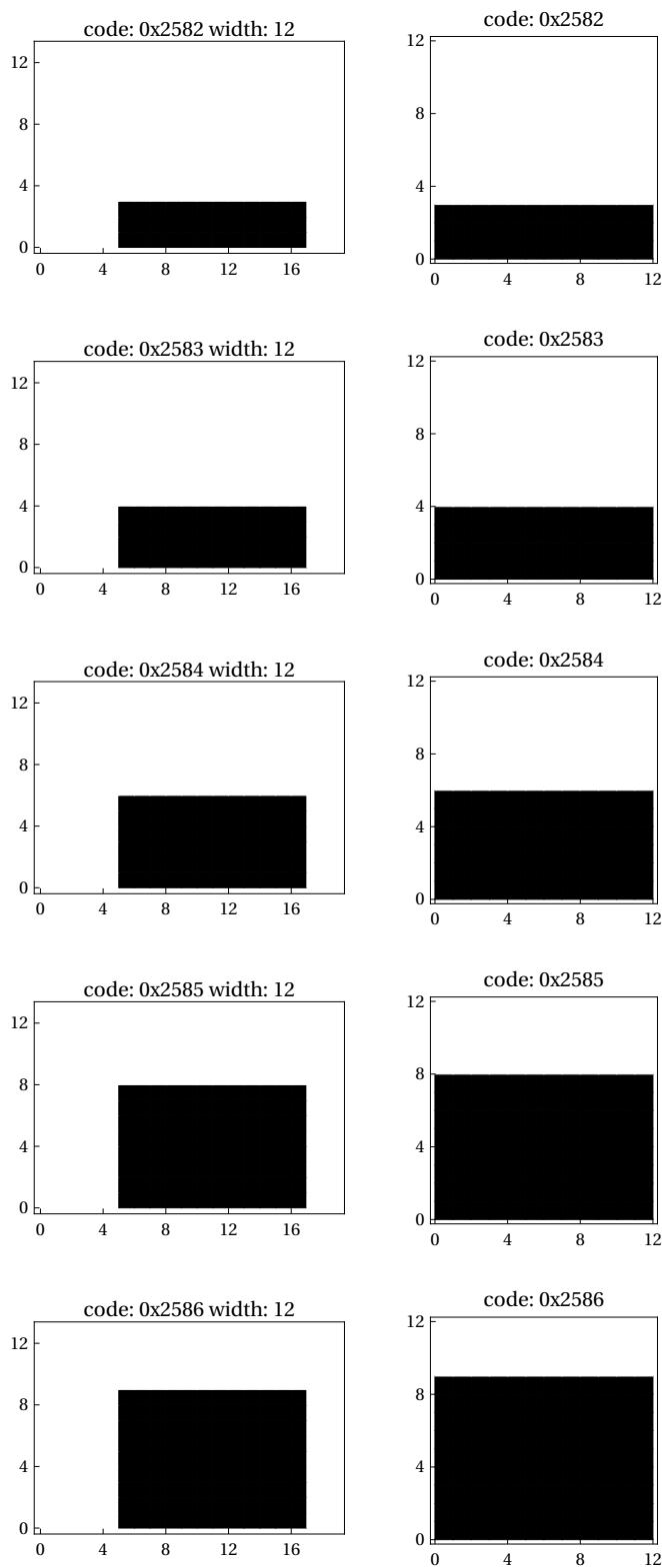


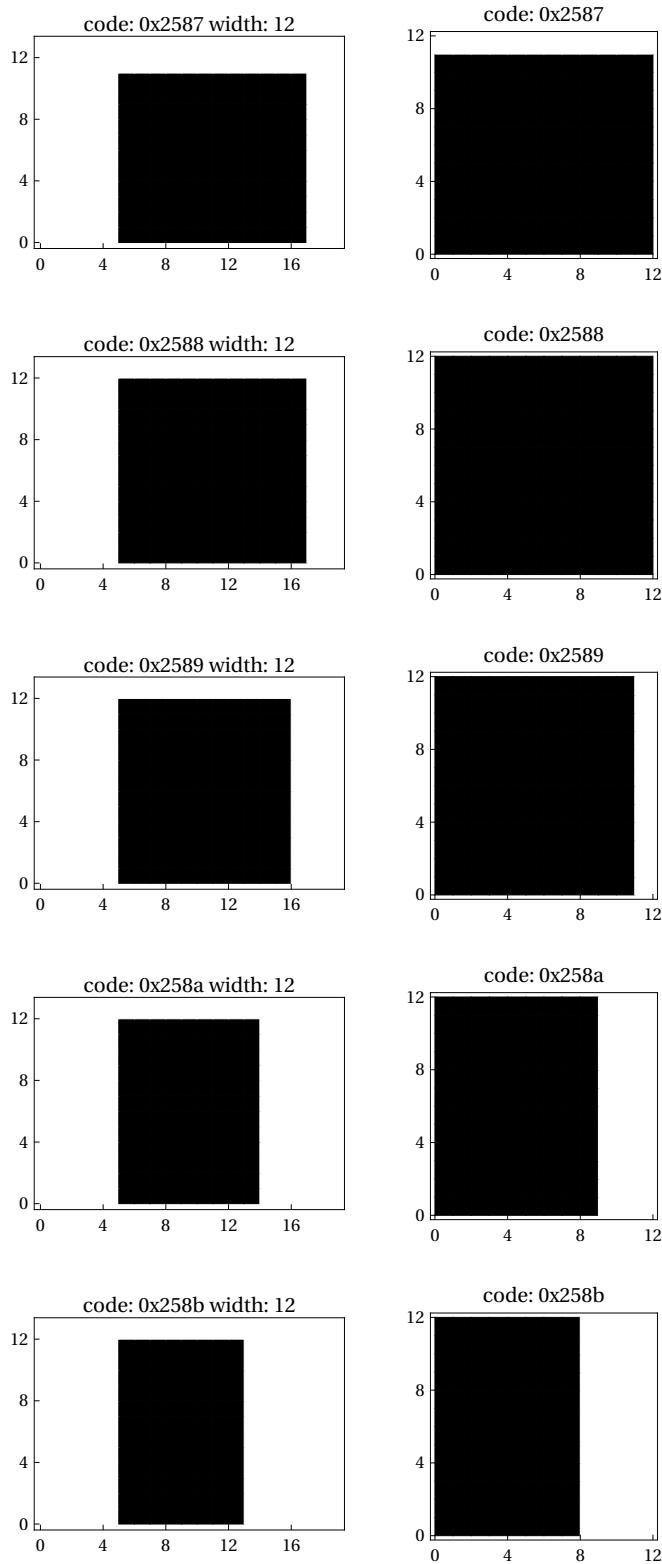


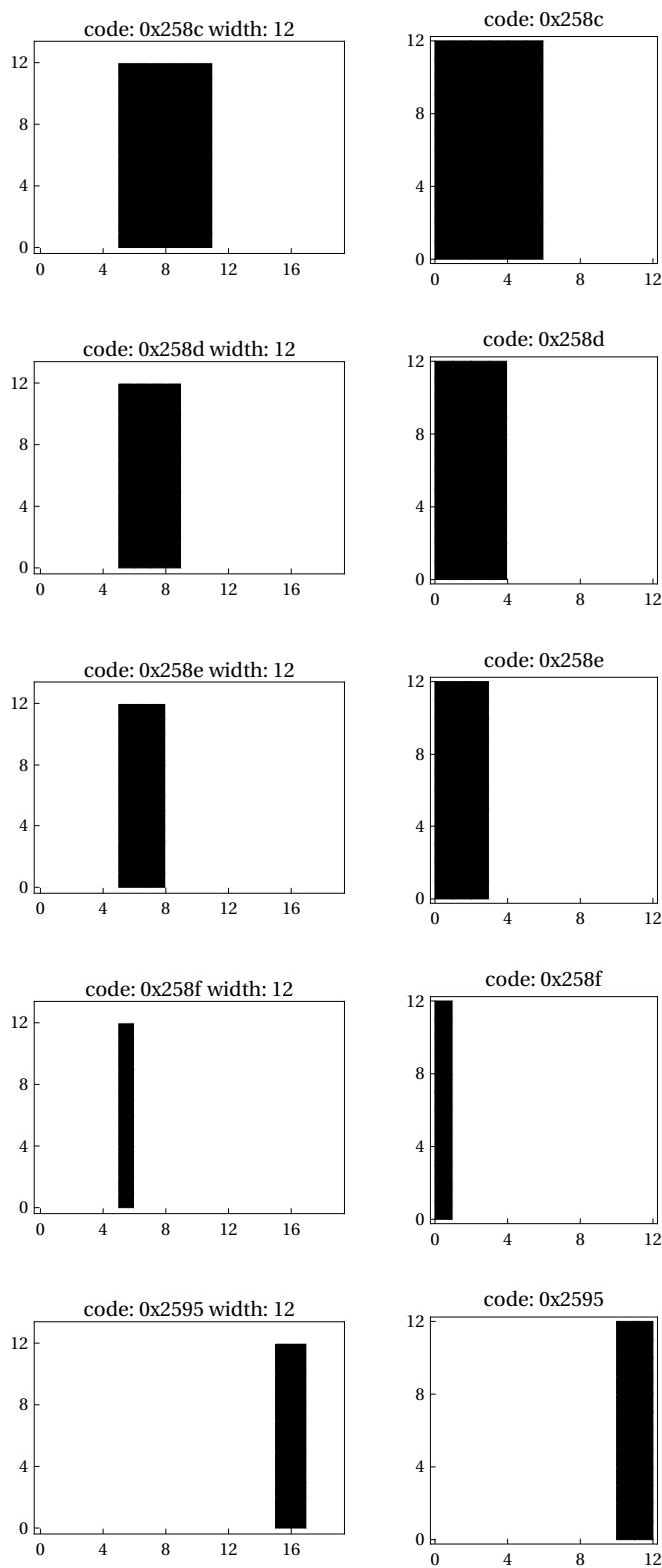


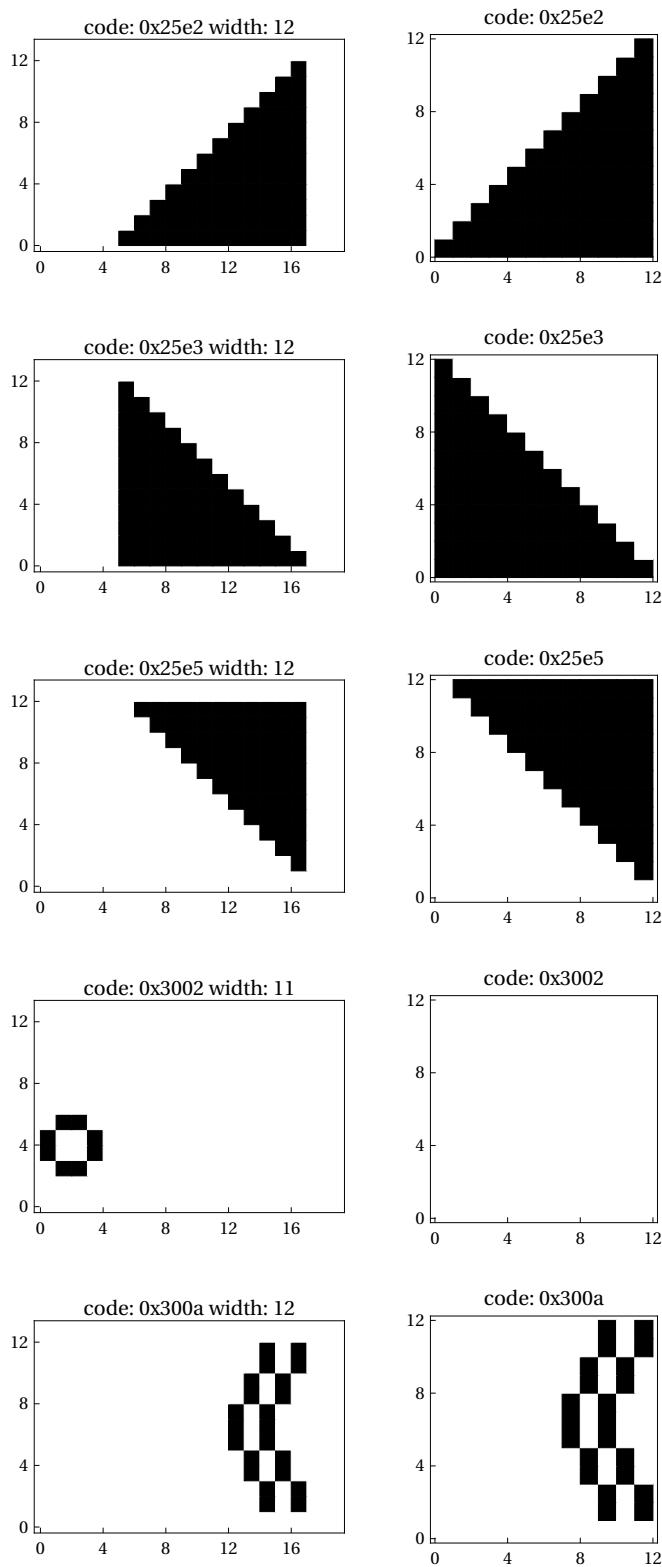


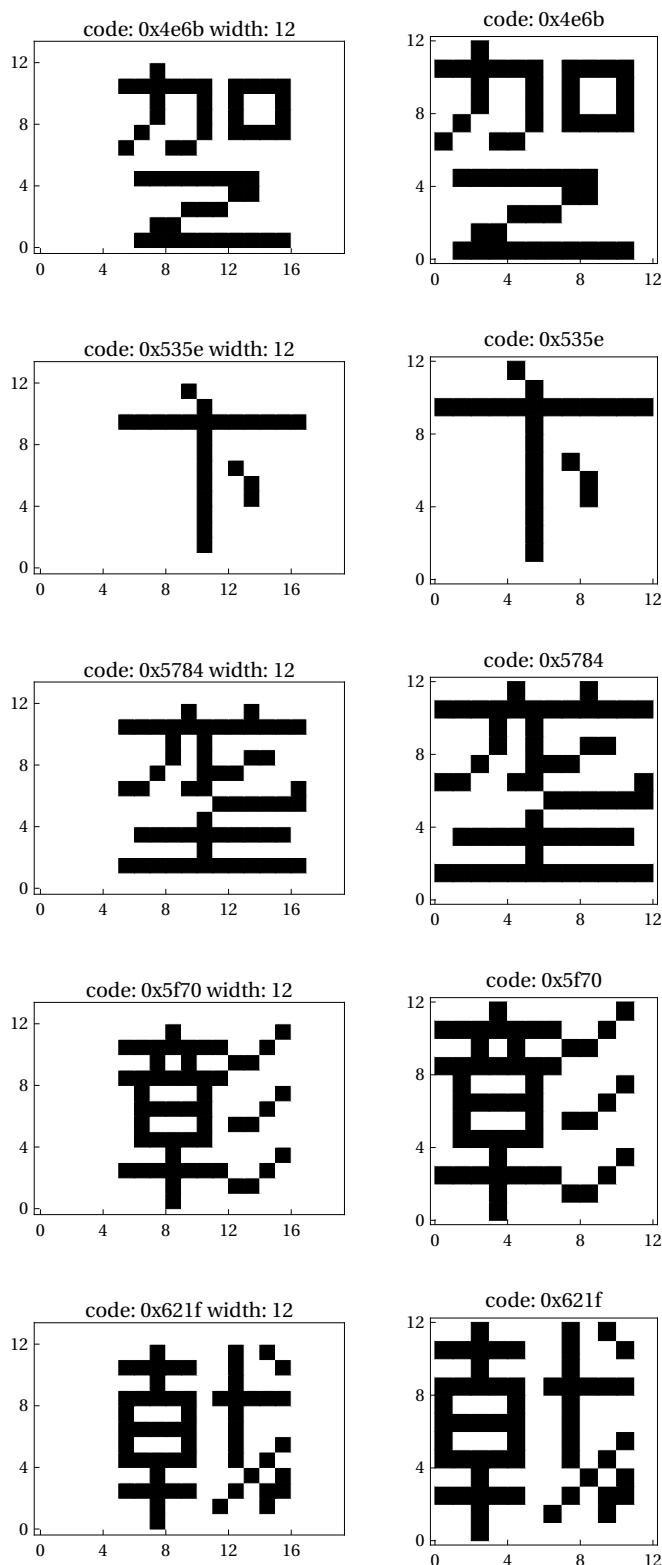


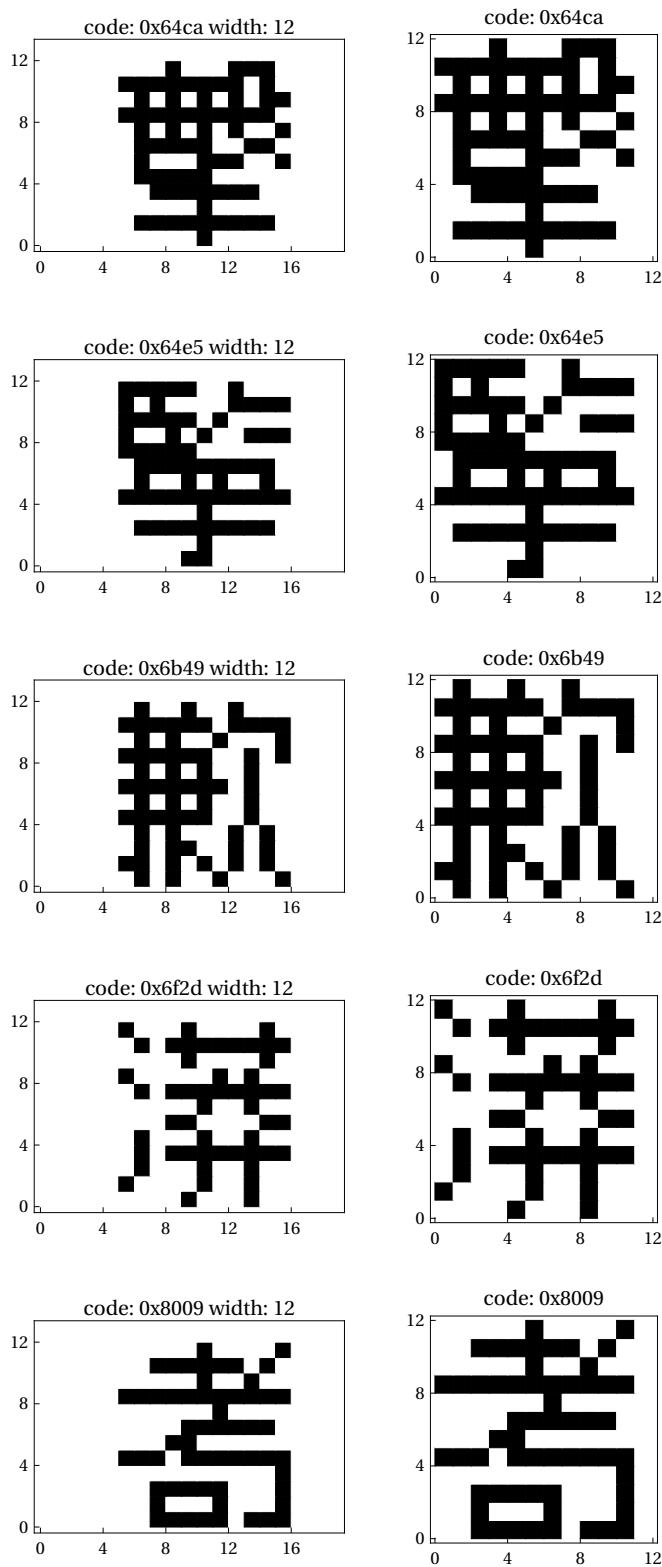


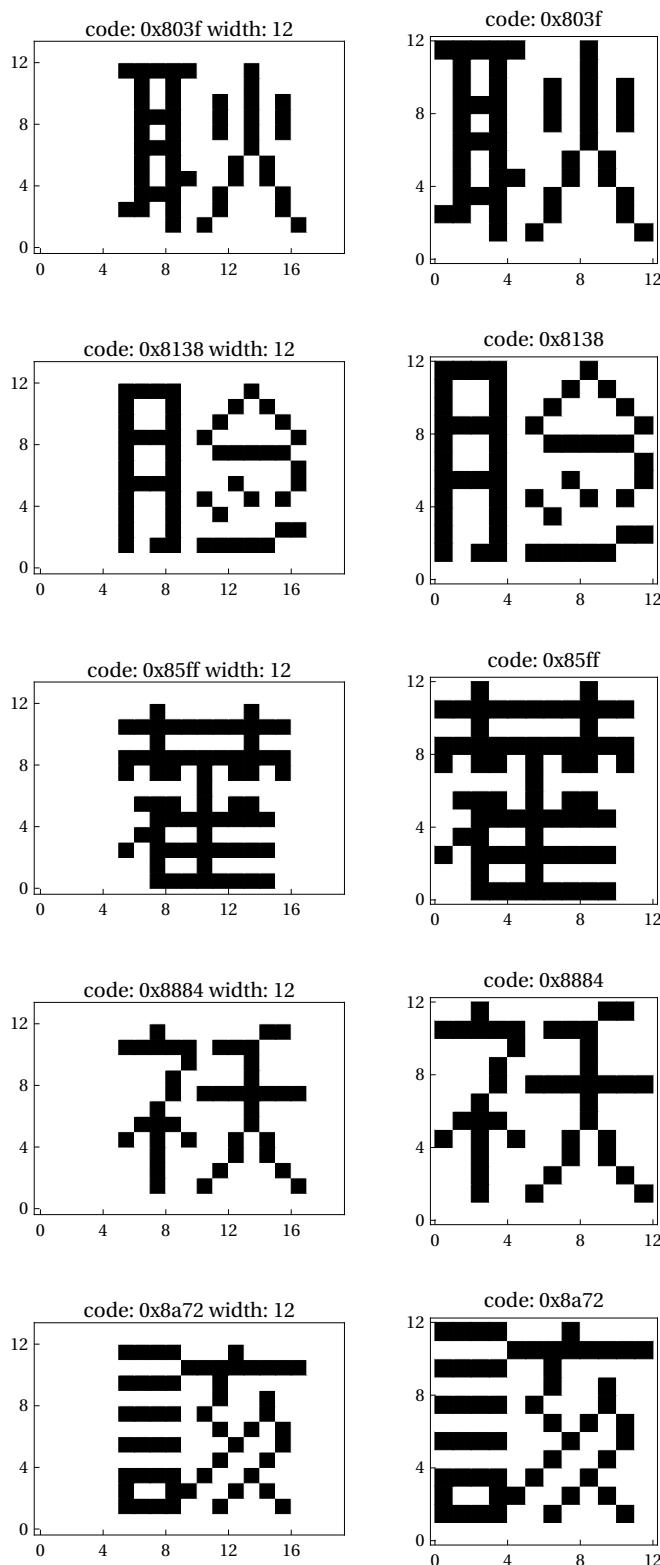


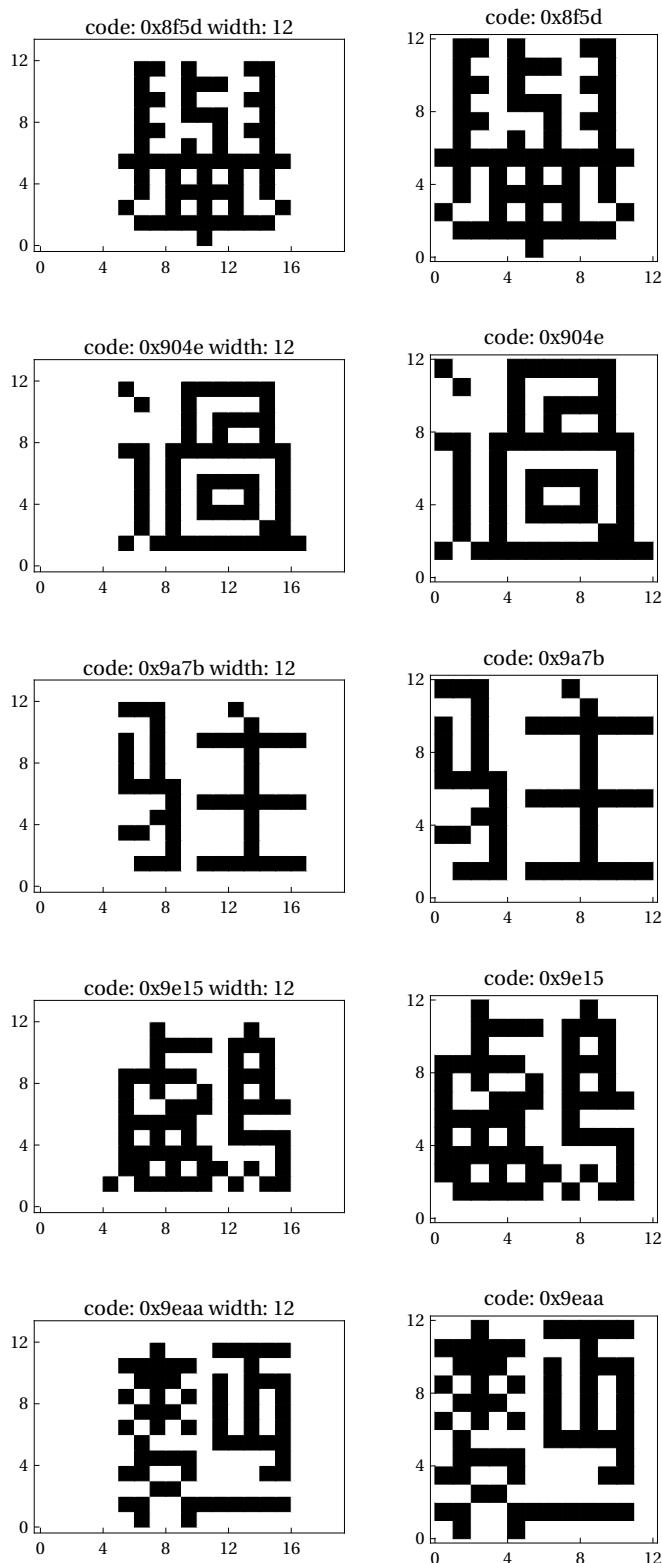


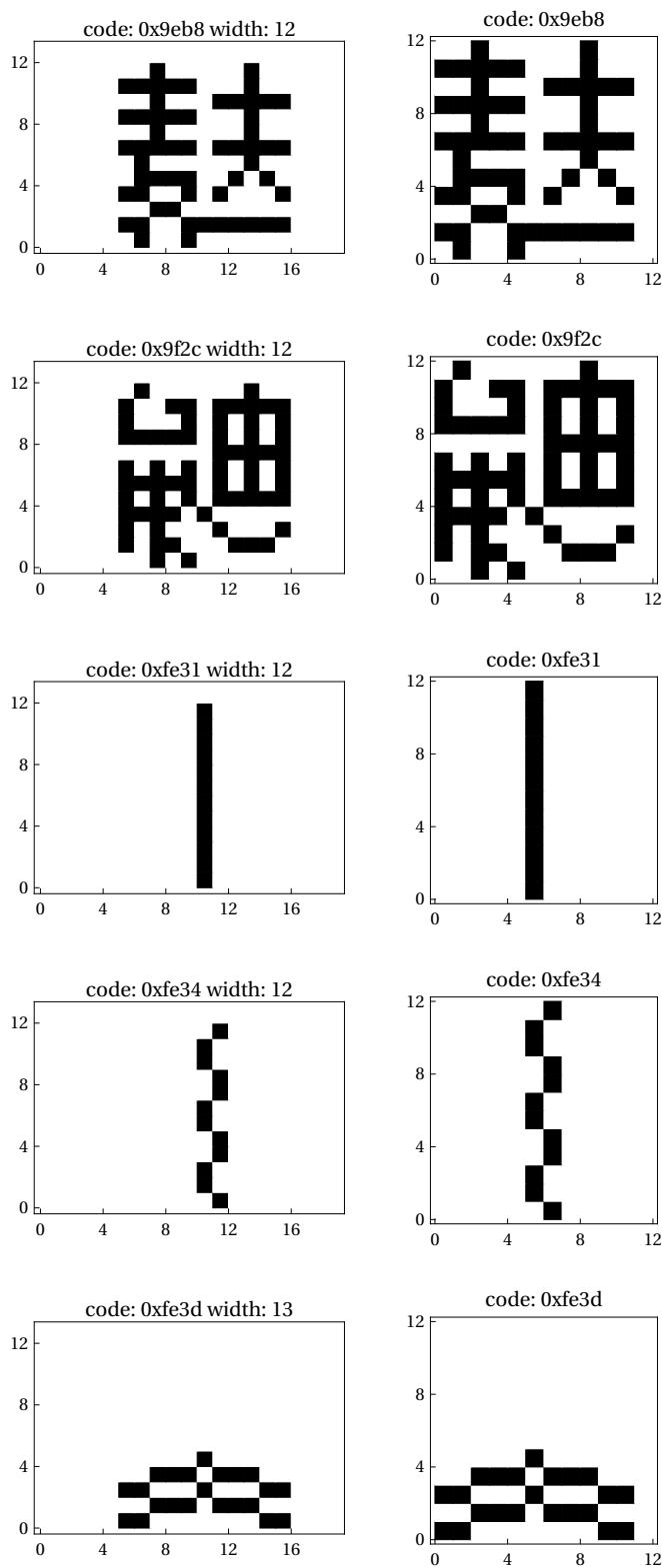


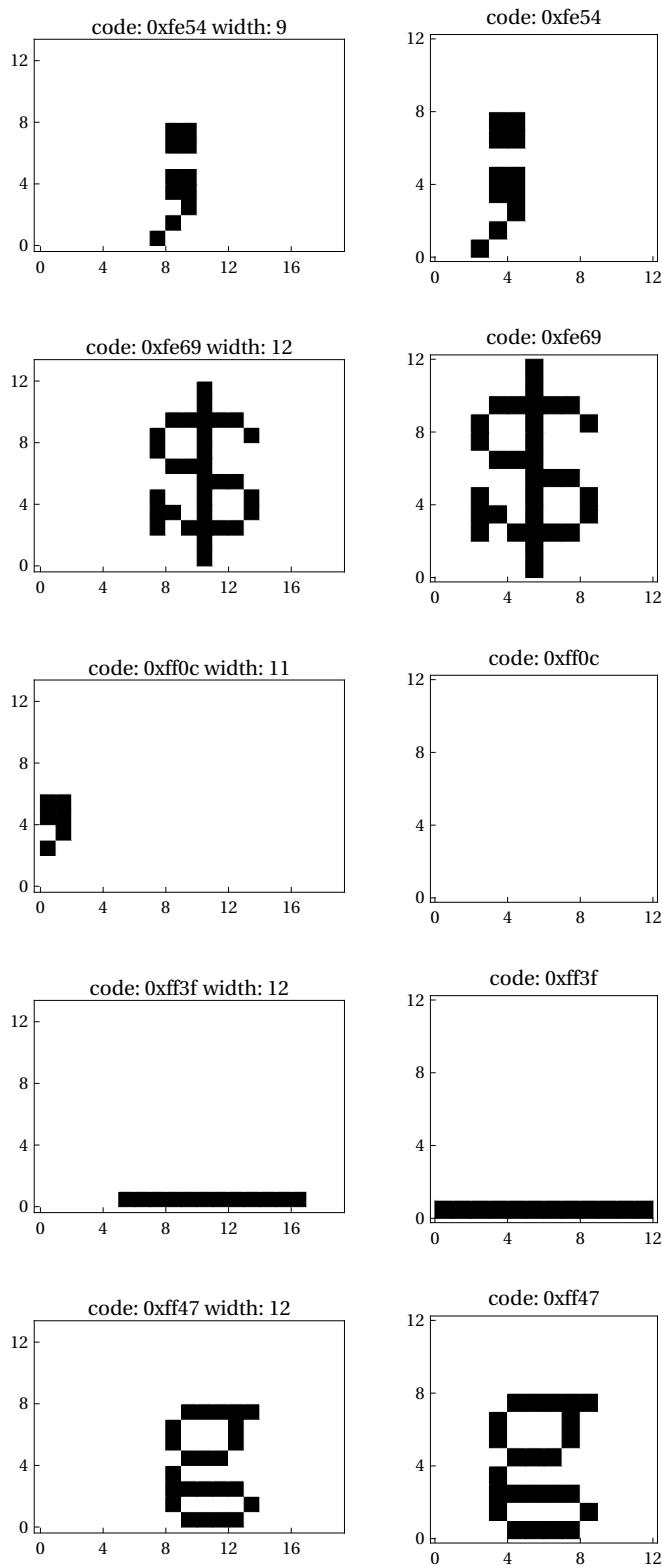


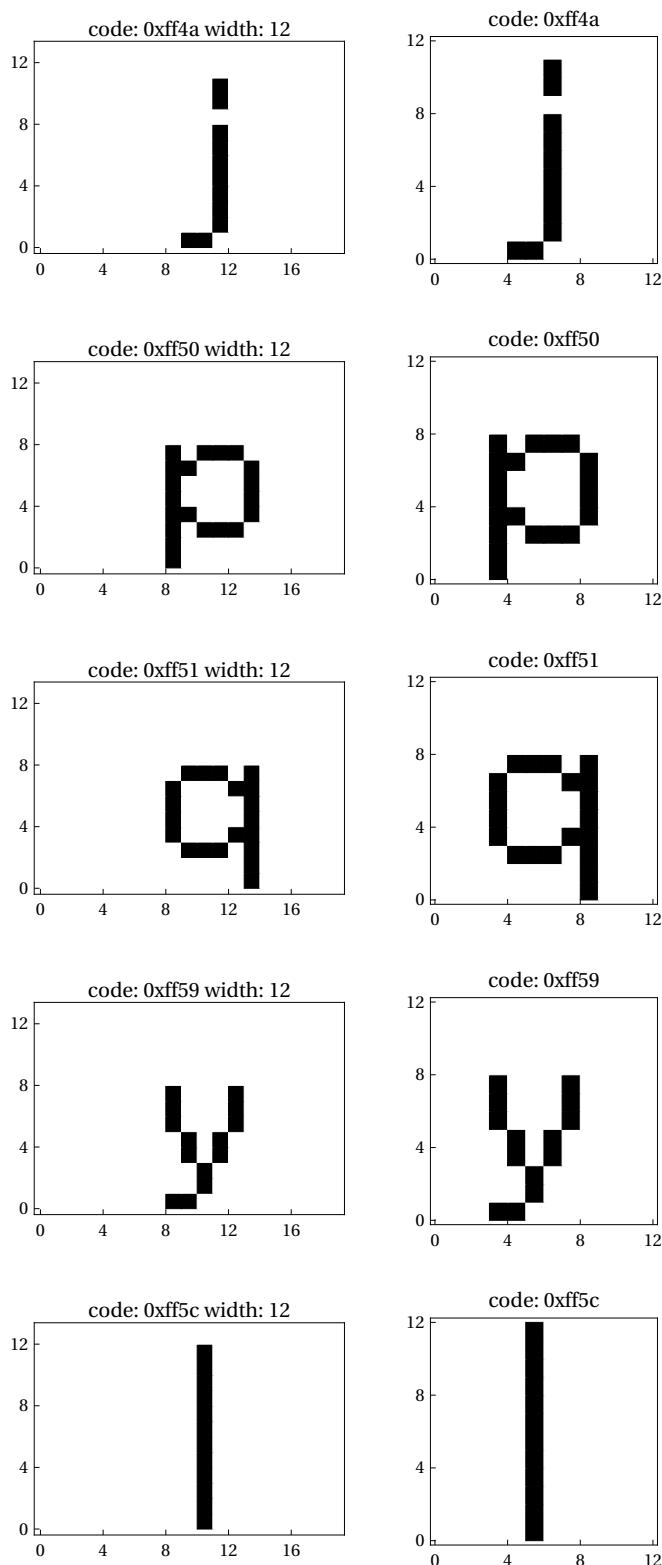


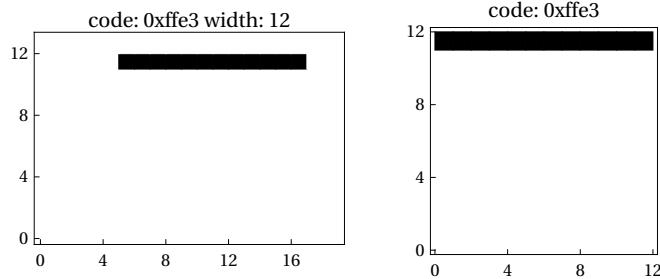










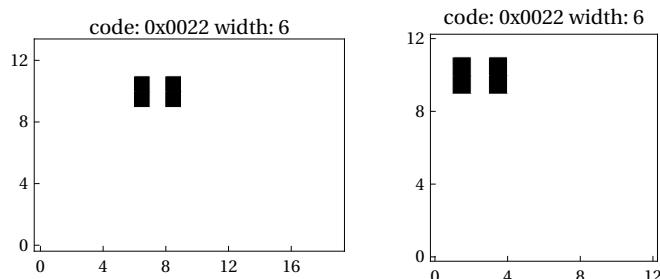
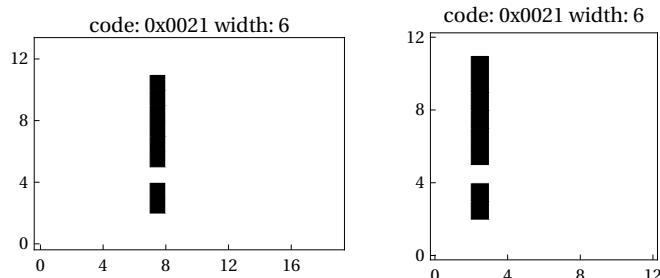
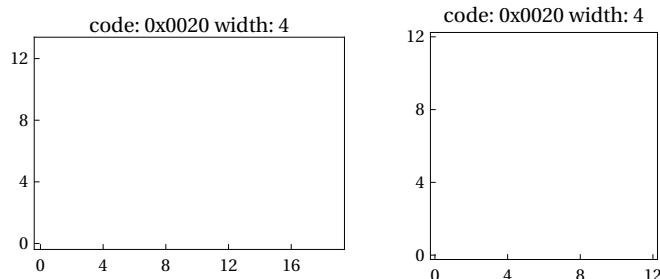


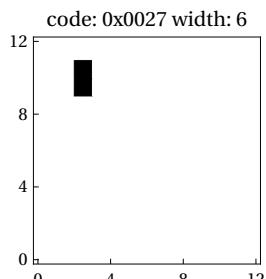
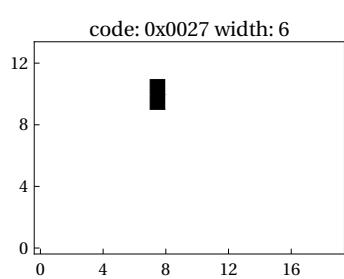
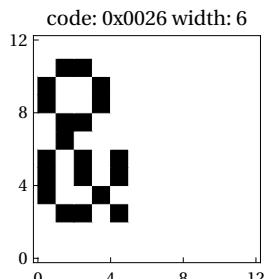
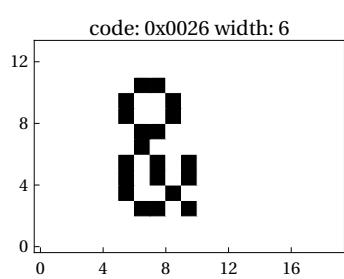
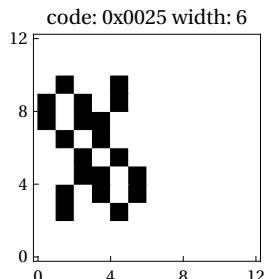
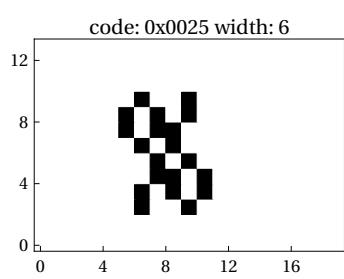
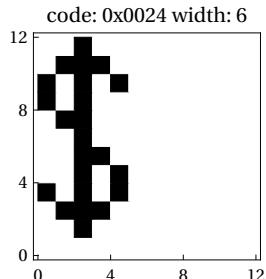
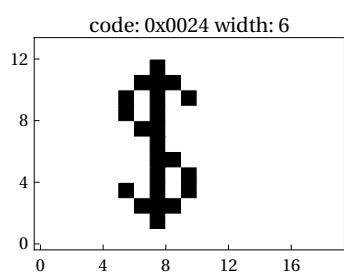
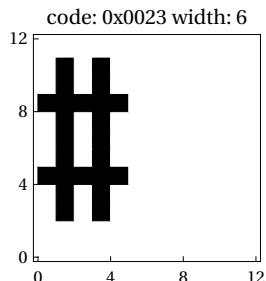
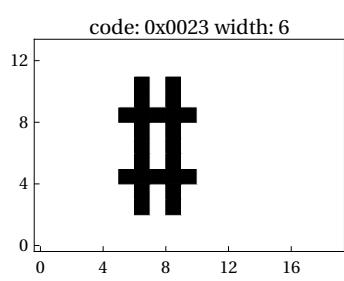
## ■ All glyphs

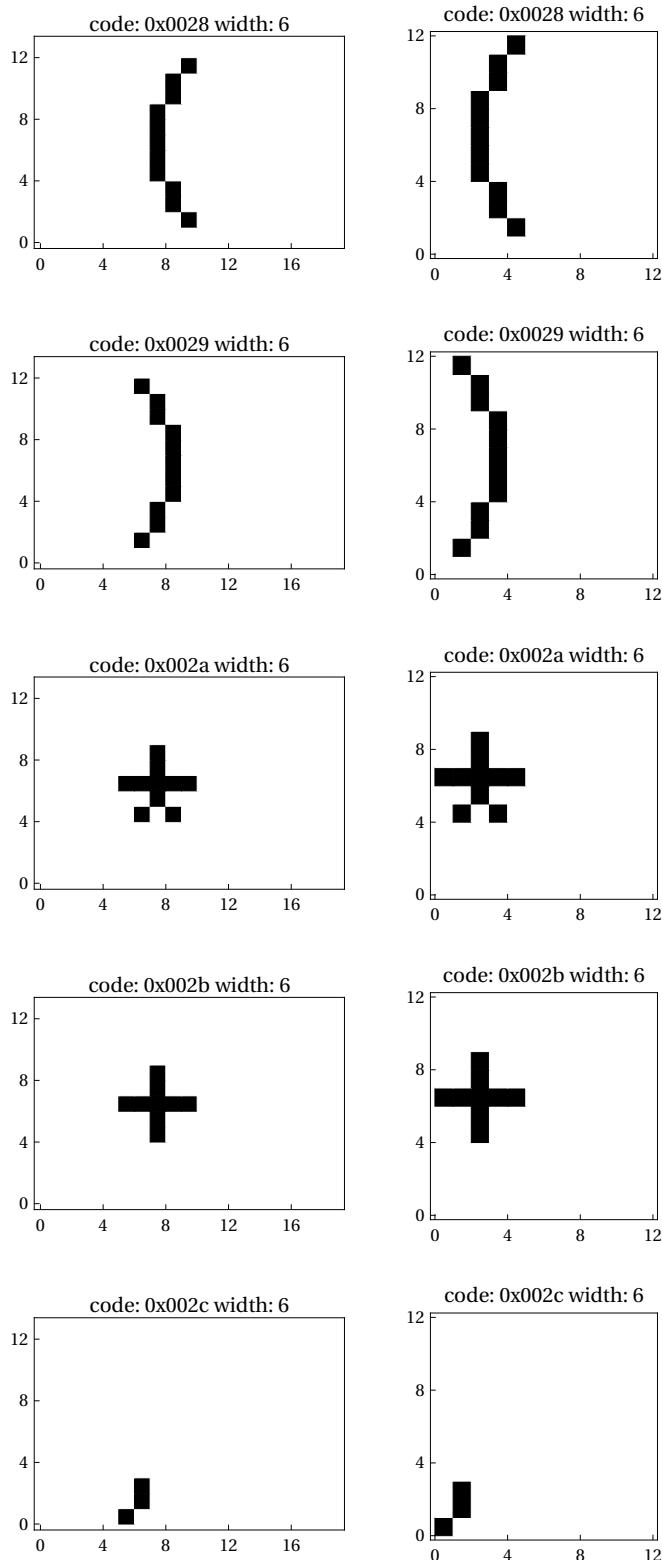
```
In[47]:= bitmapgrafix = MapThread[Graphics[Raster[Reverse[#1]], Frame -> True, AspectRatio ->
    Divide @@ Dimensions[#1], FrameTicks -> ({#, #, {}, {}} &[Range[0, 16, 4]]),
    PlotLabel -> ("code: " <> IntegerString[ToExpression[#2], 16, 4] <>
    " width: " <> ToString[#3])] &, {extbitmaps, charcodes, widths}];

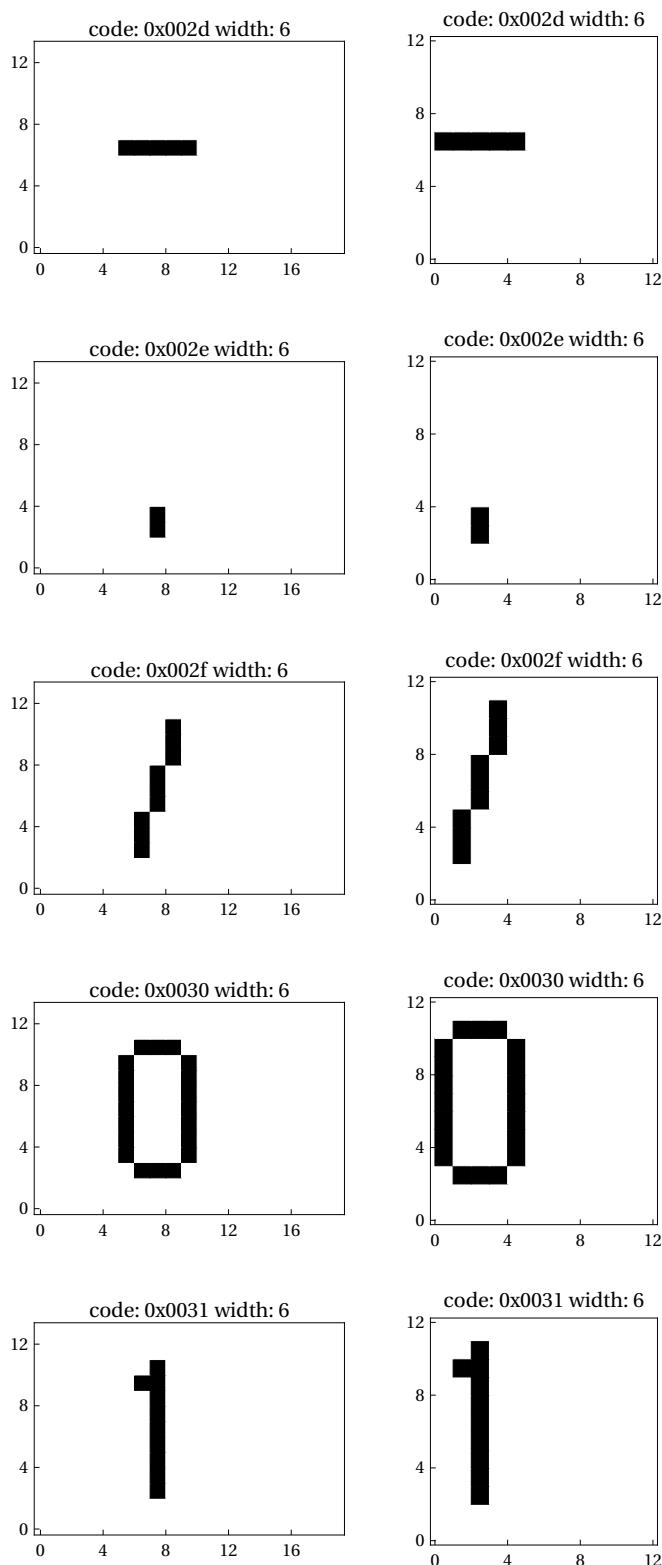
In[48]:= bitmaprgrafix = MapThread[Graphics[Raster[Reverse[#1]], Frame -> True, AspectRatio ->
    Divide @@ Dimensions[#1], FrameTicks -> ({#, #, {}, {}} &[Range[0, 16, 4]]),
    PlotLabel -> ("code: " <> IntegerString[ToExpression[#2], 16, 4] <>
    " width: " <> ToString[#3])] &, {trbitmaps, charcodes, widths}];

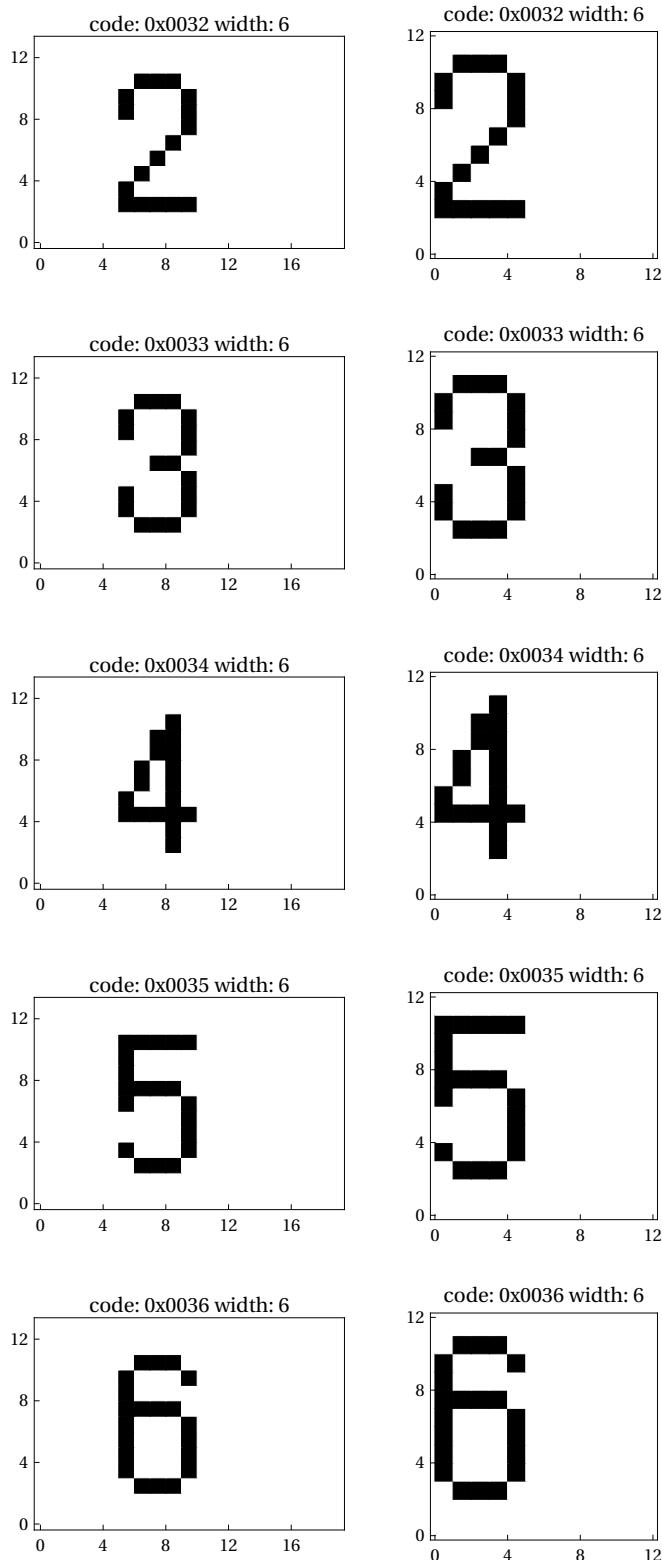
In[49]:= MapThread[Print[GraphicsGrid[{{##}}]] &,
    Take[#, 1024] &/@{bitmapgrafix, bitmaprgrafix}];
```

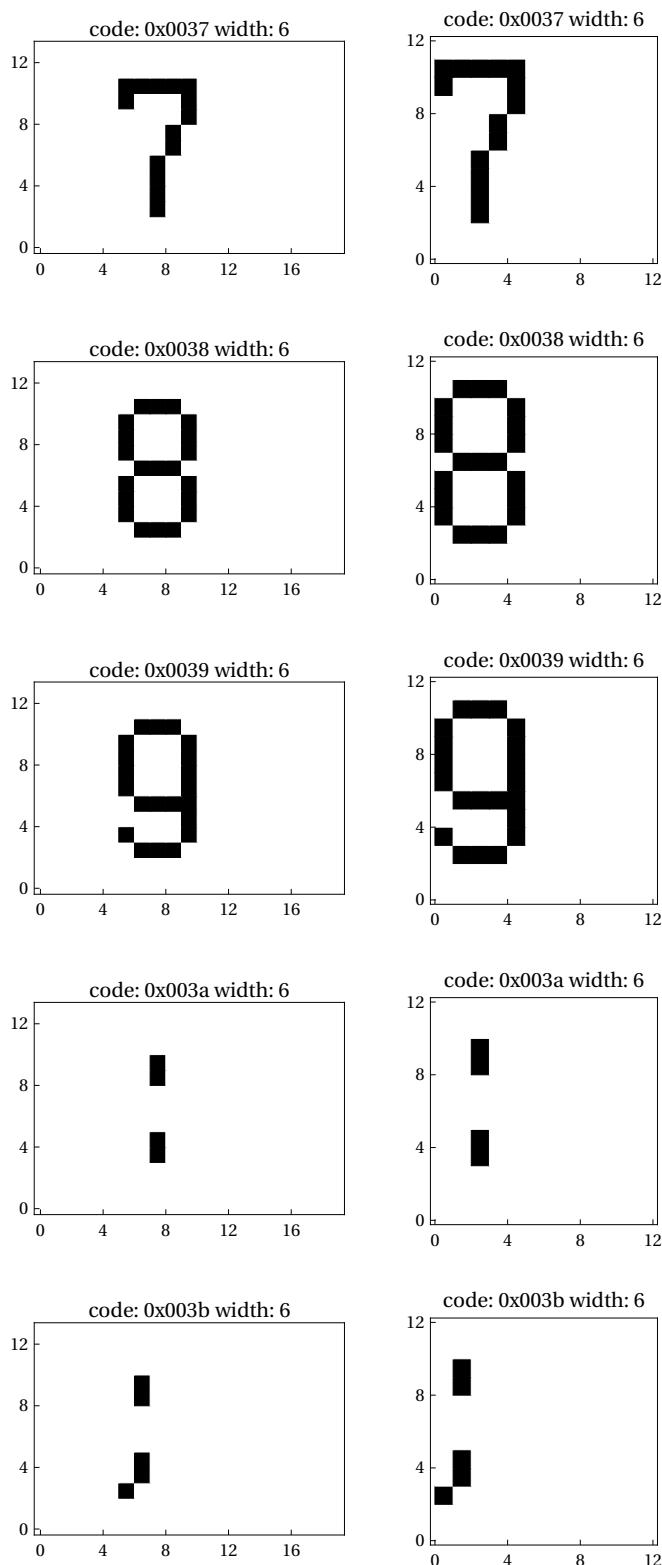


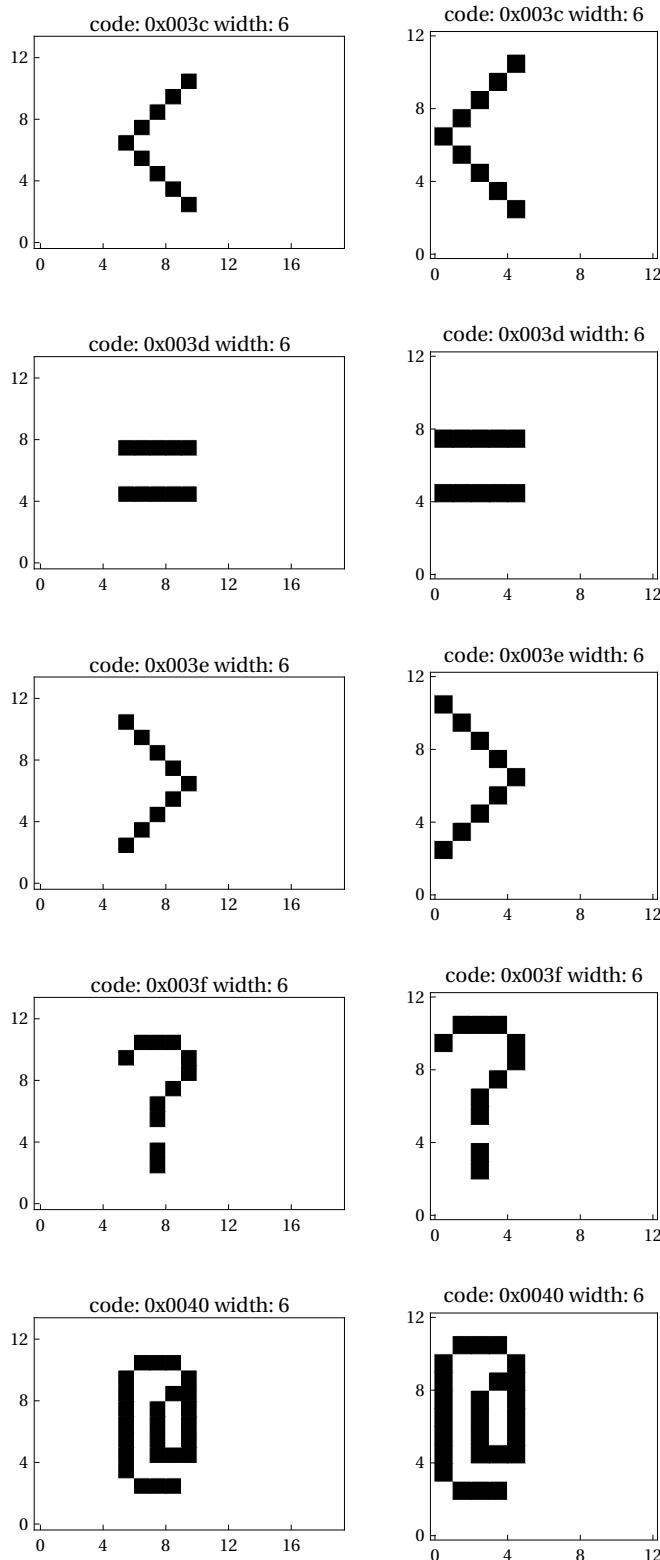


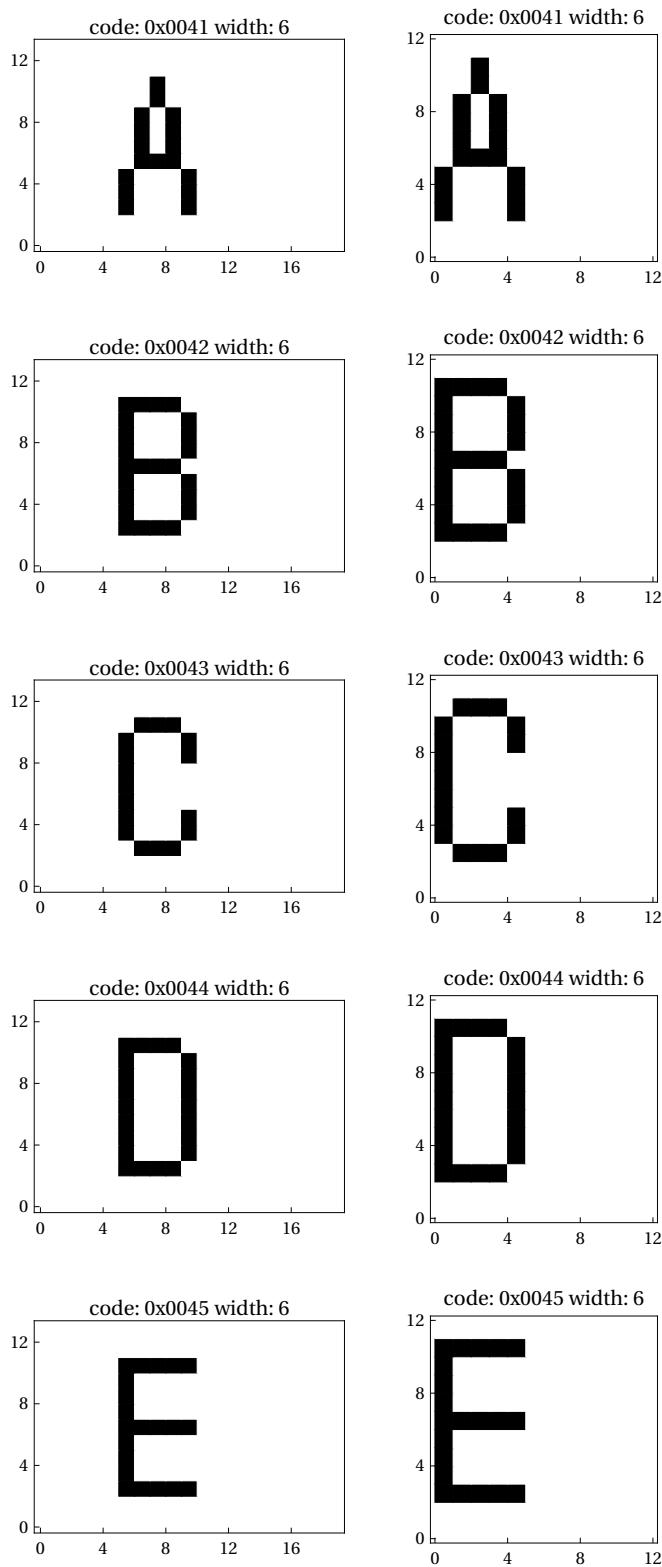


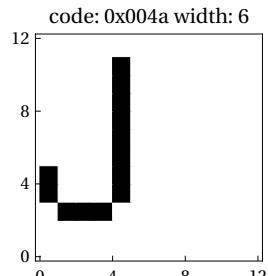
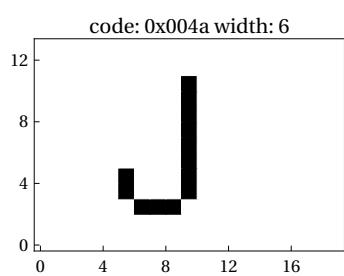
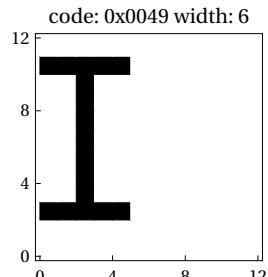
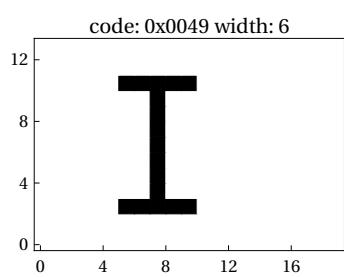
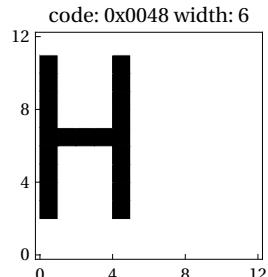
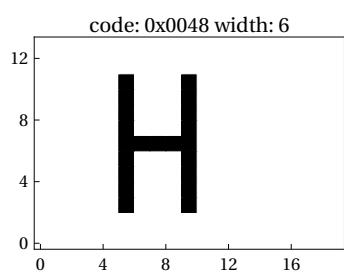
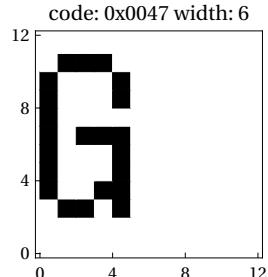
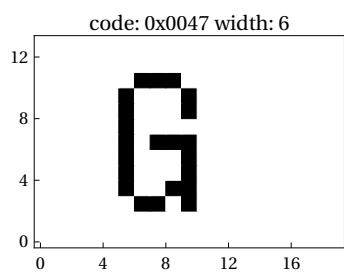
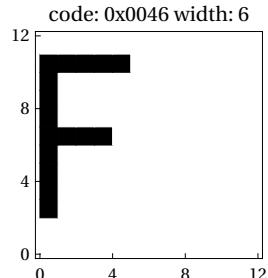
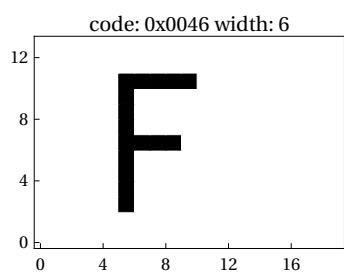


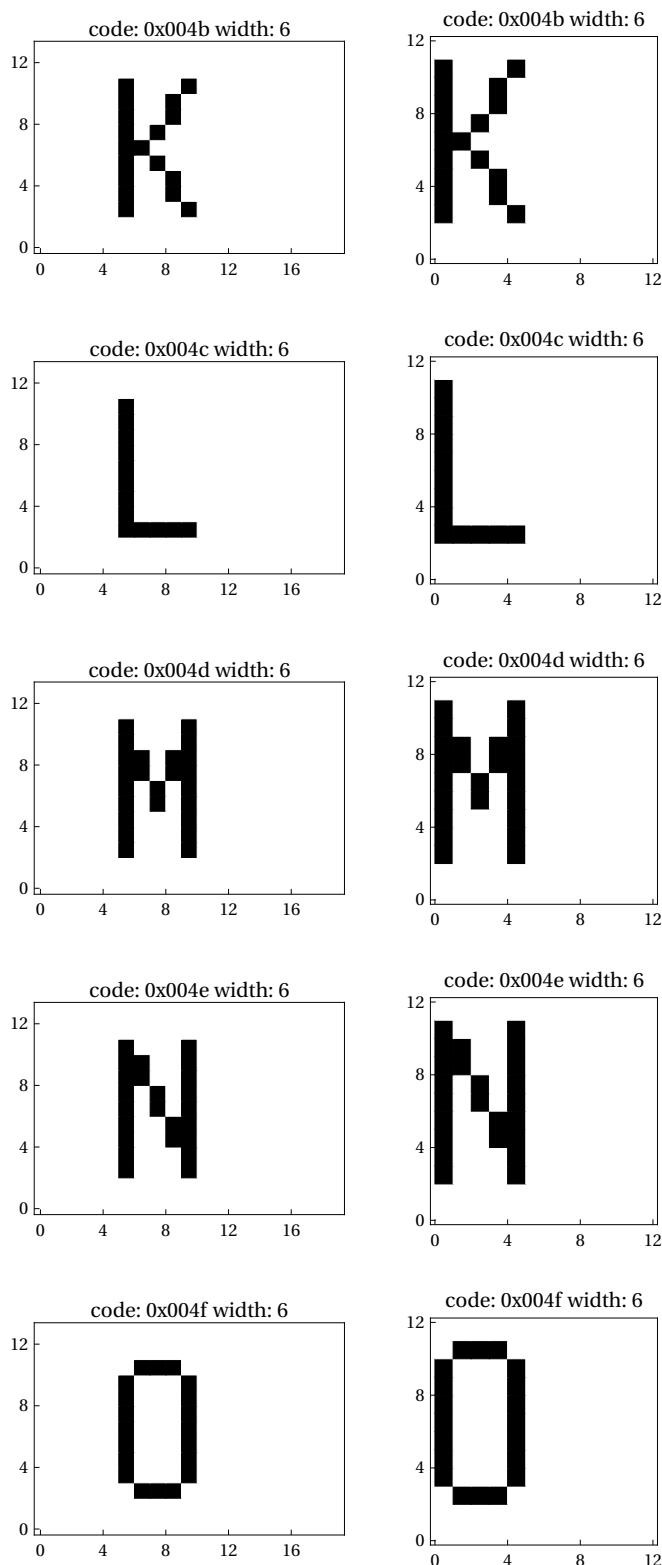


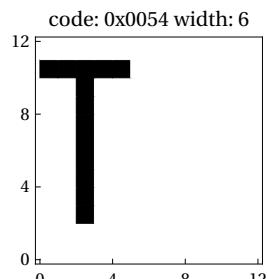
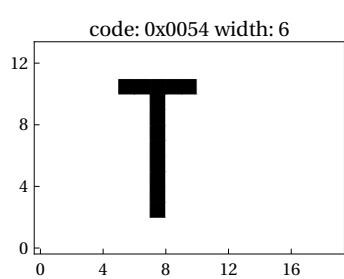
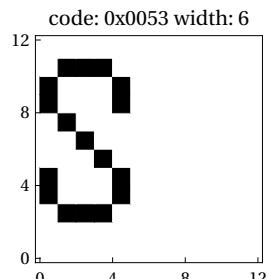
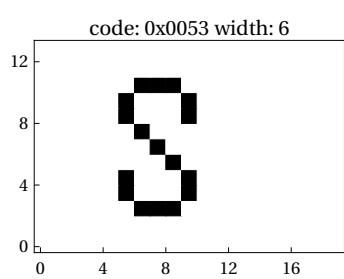
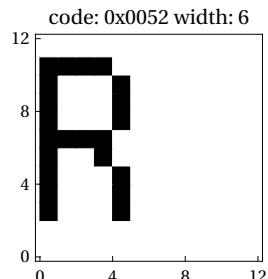
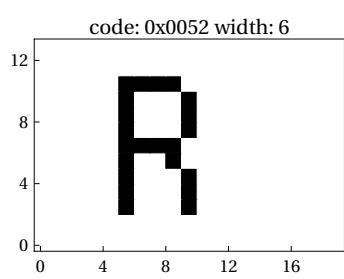
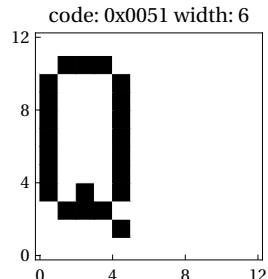
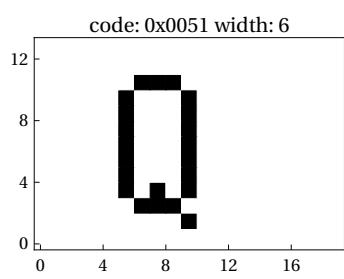
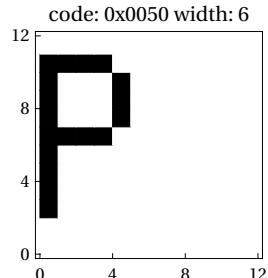
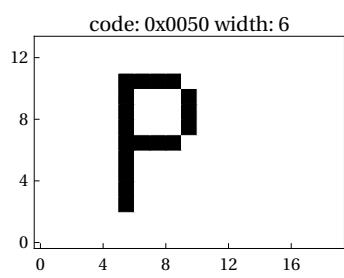


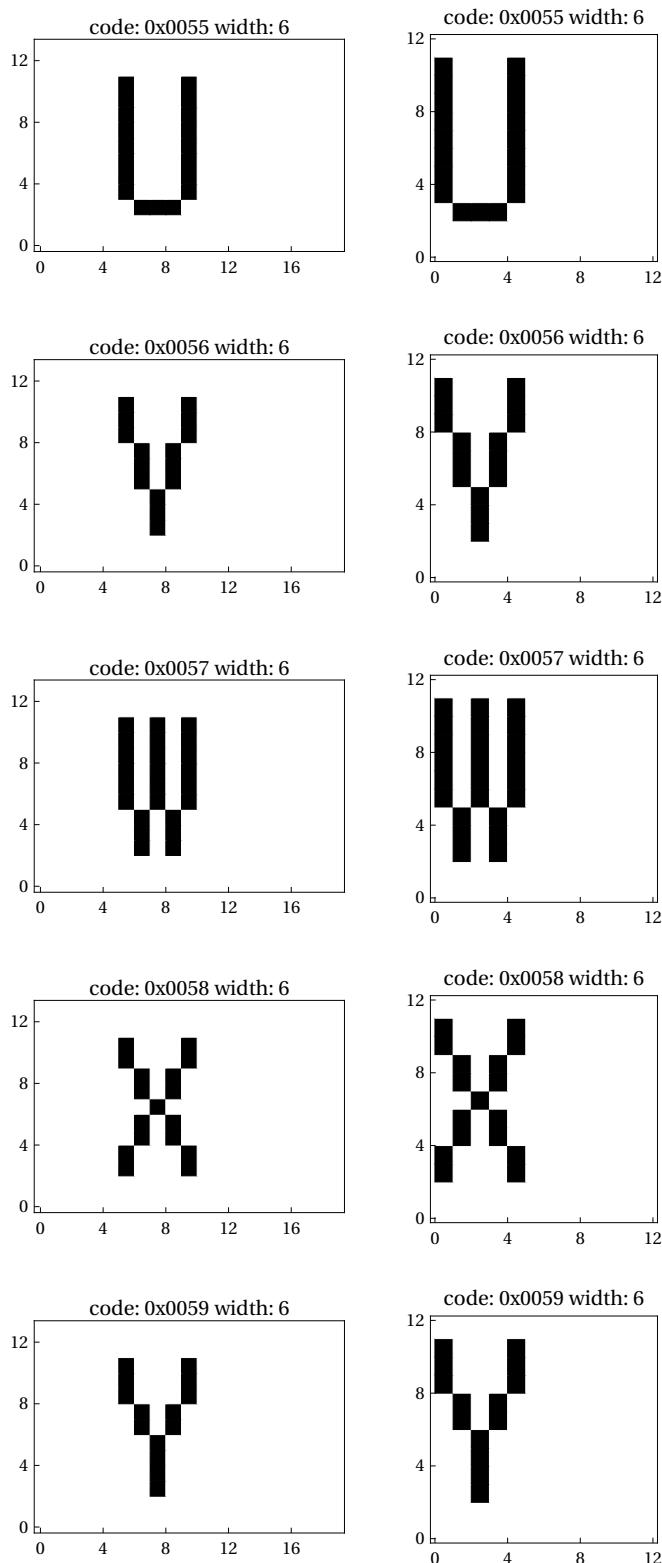


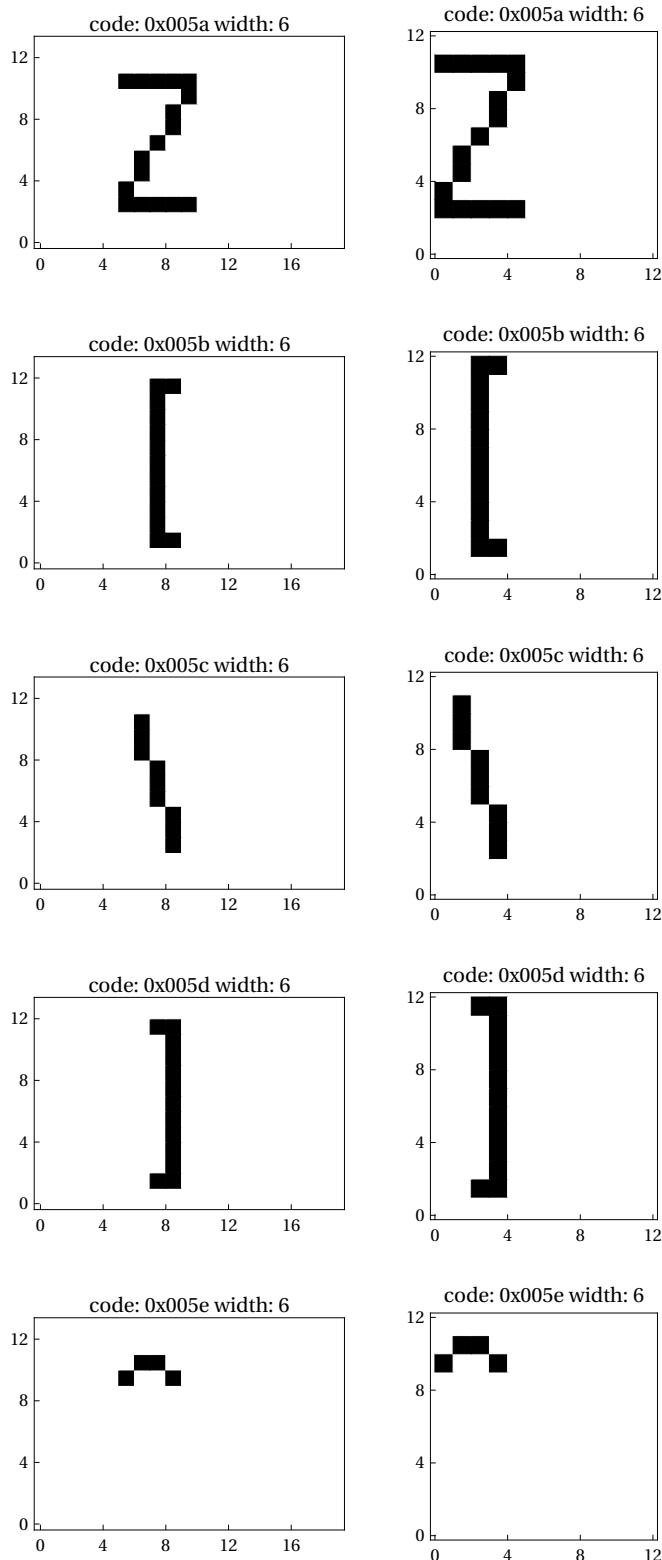


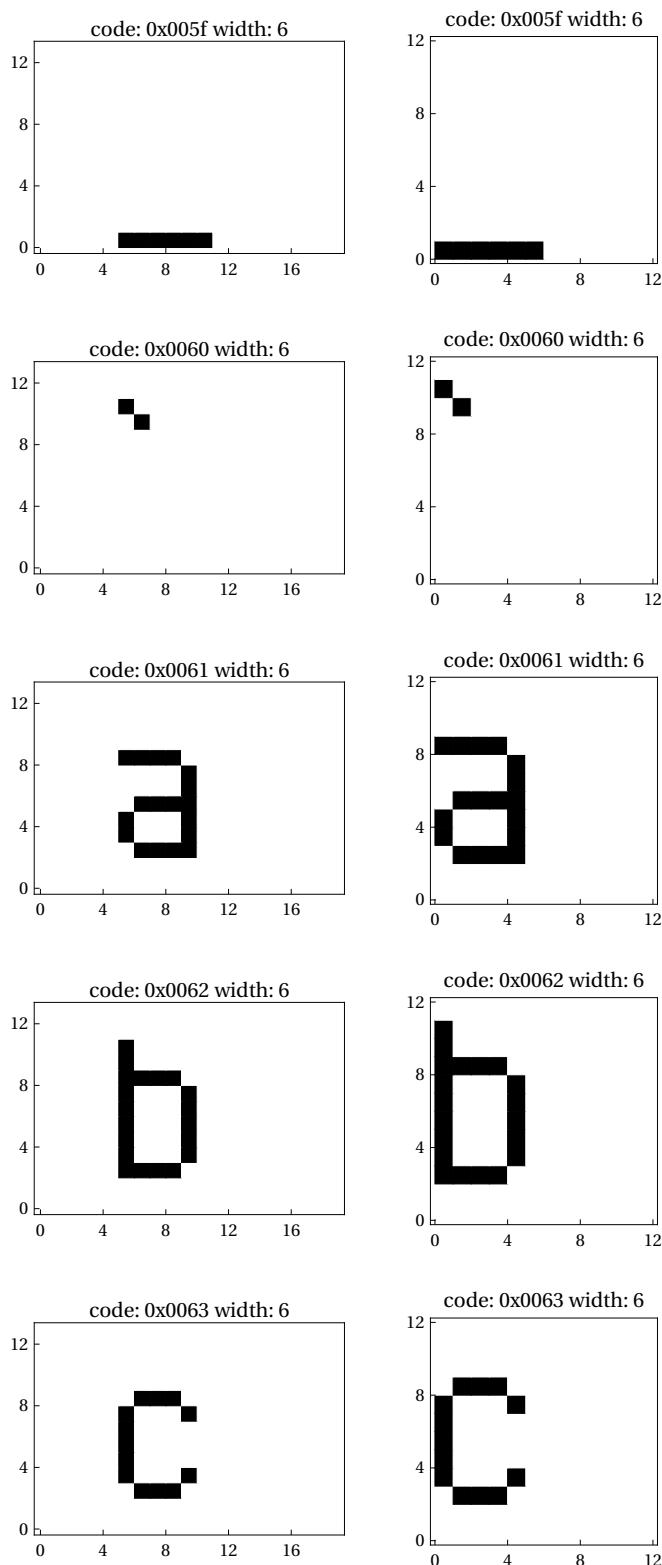


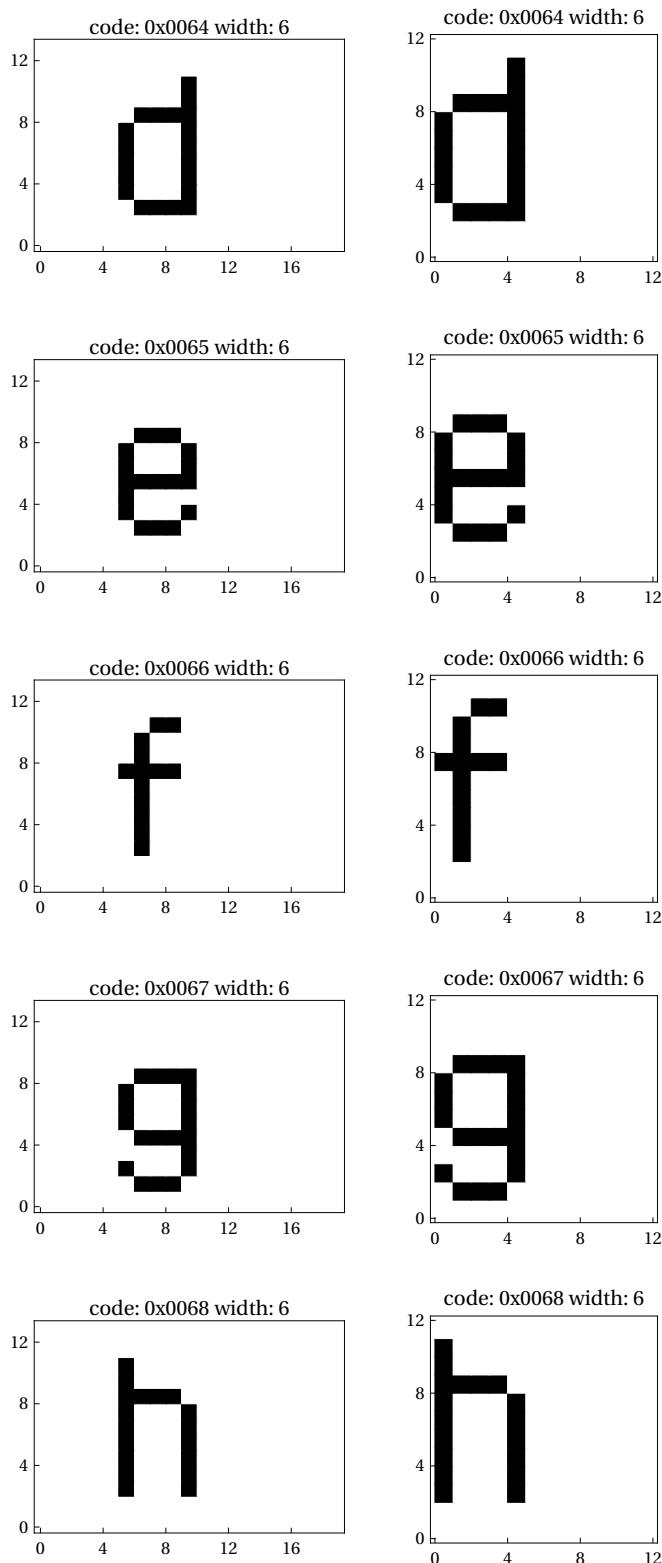


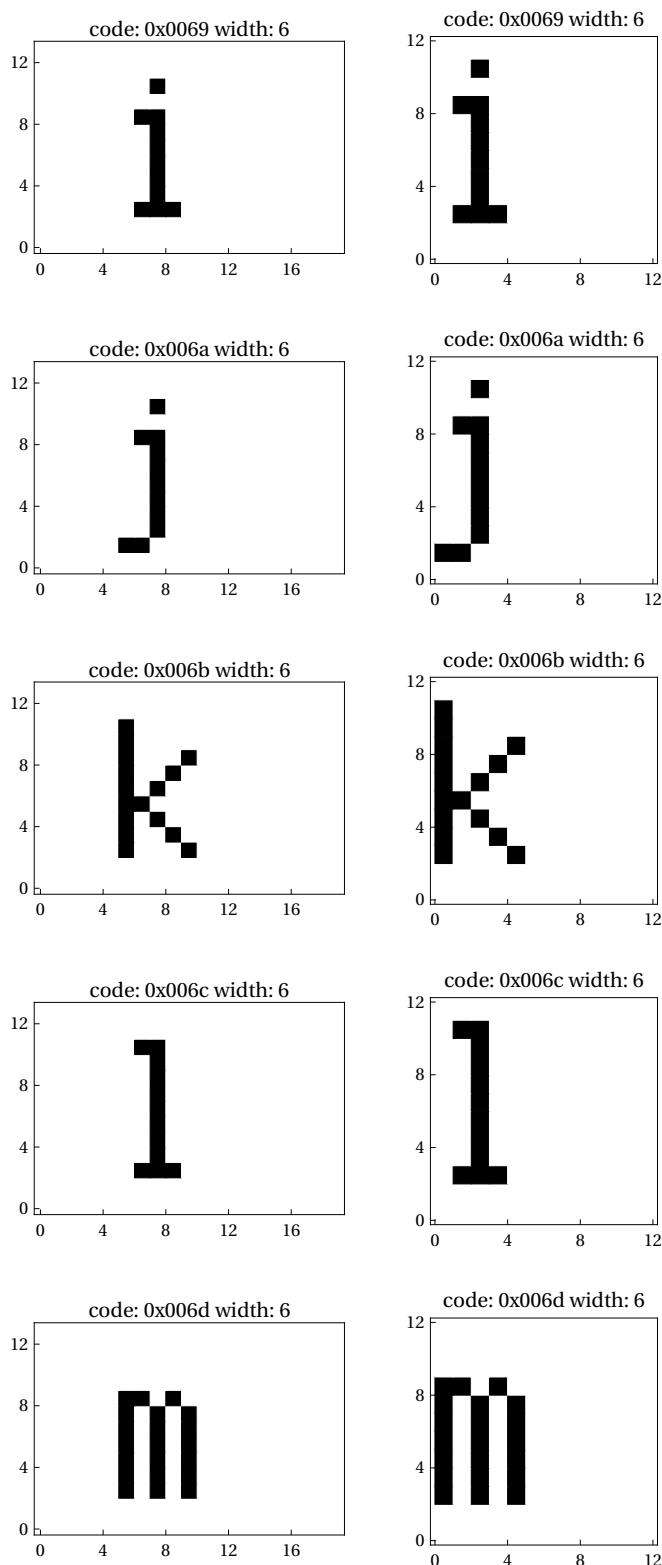


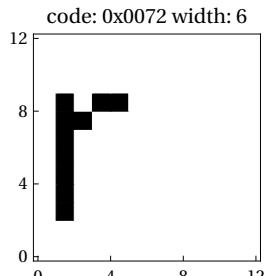
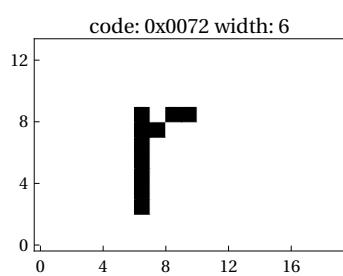
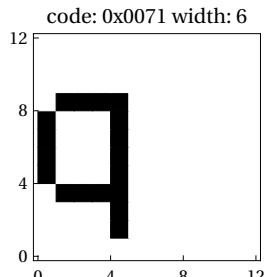
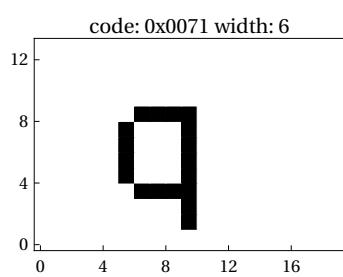
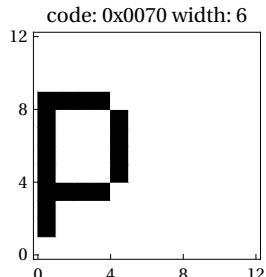
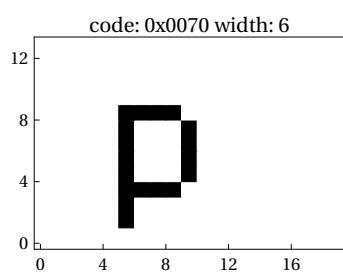
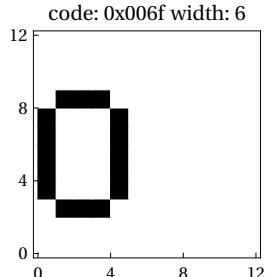
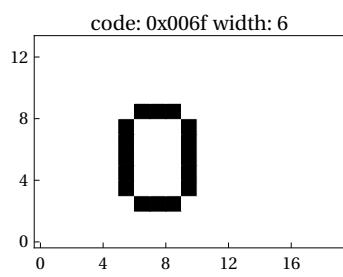
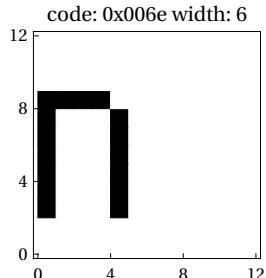
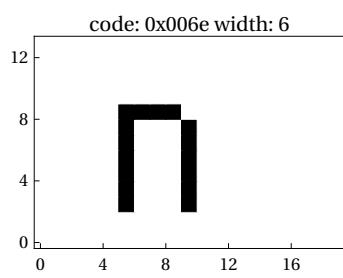


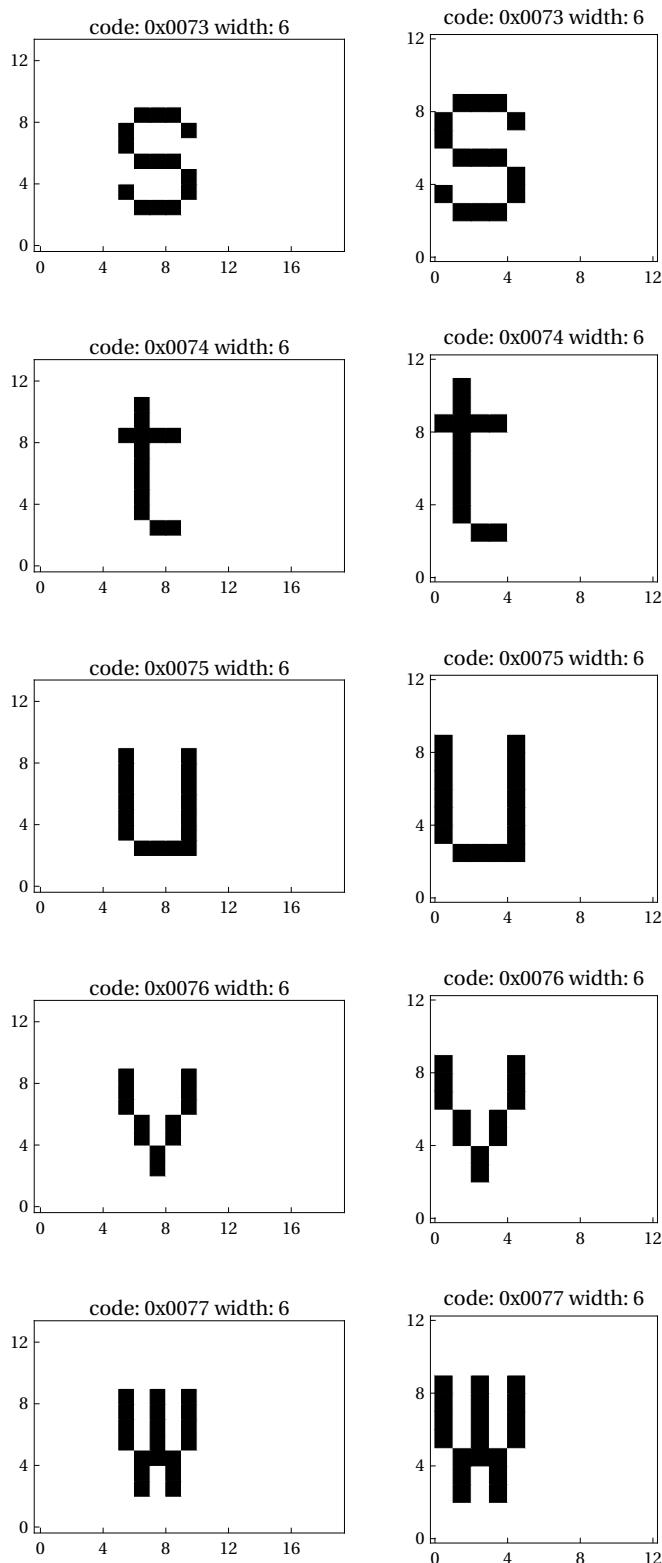


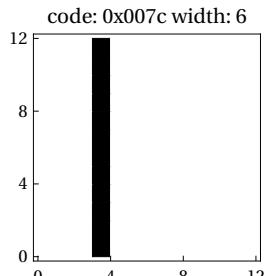
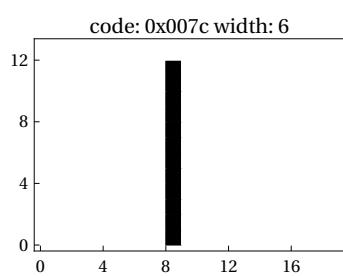
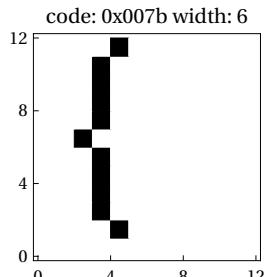
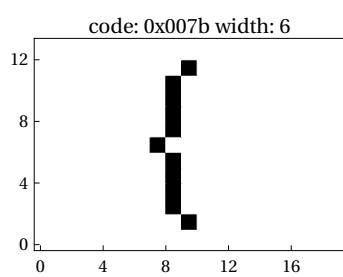
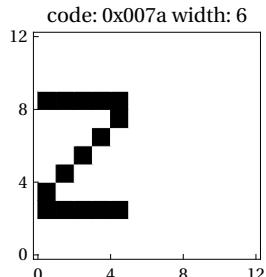
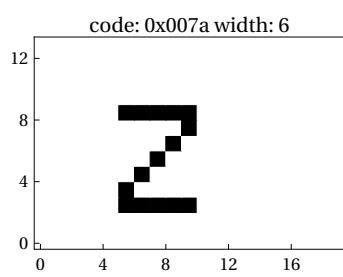
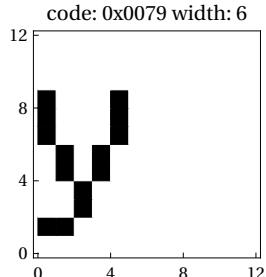
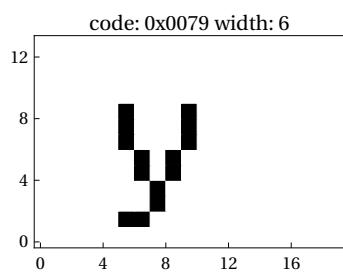
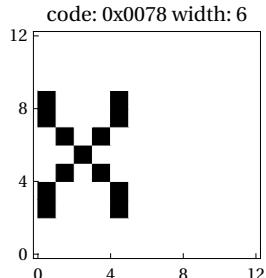
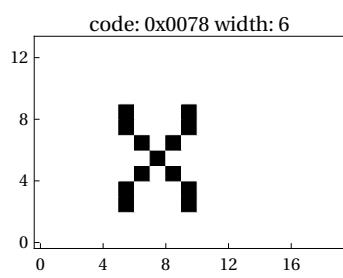


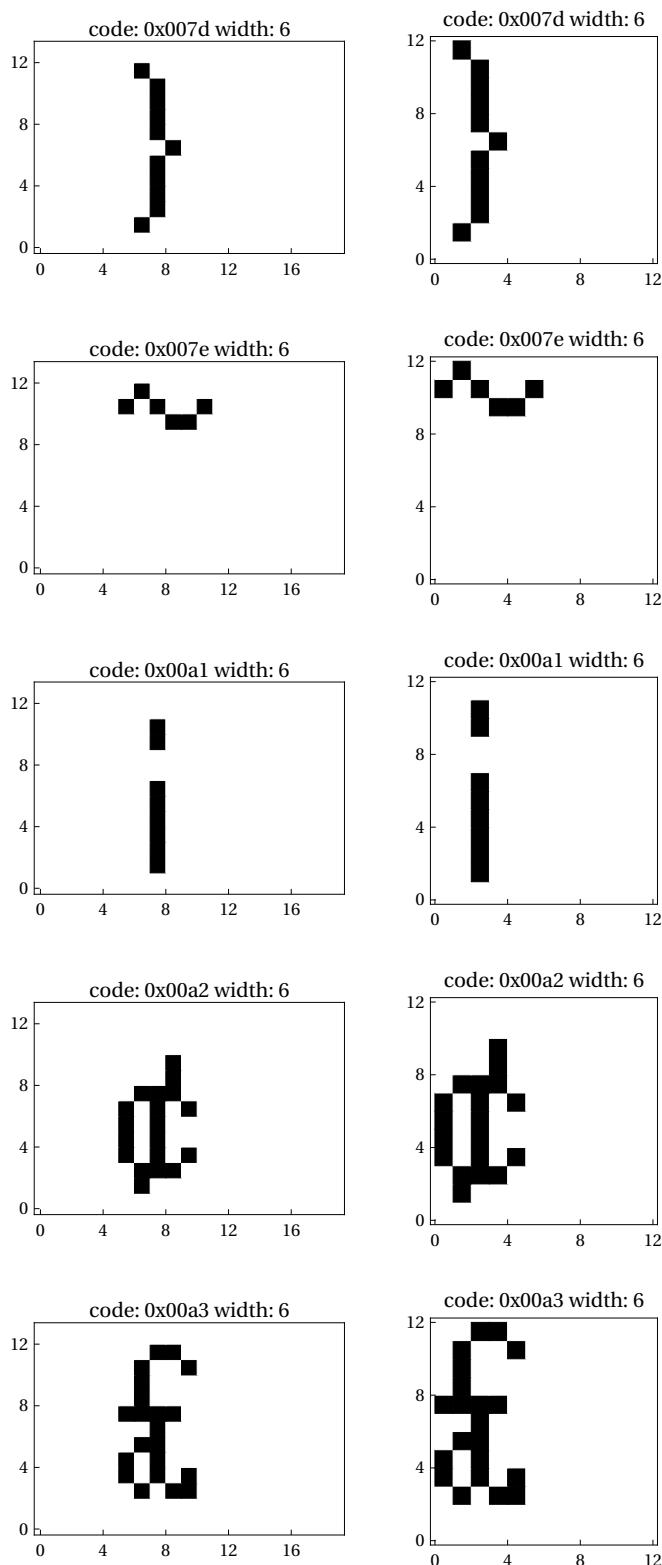


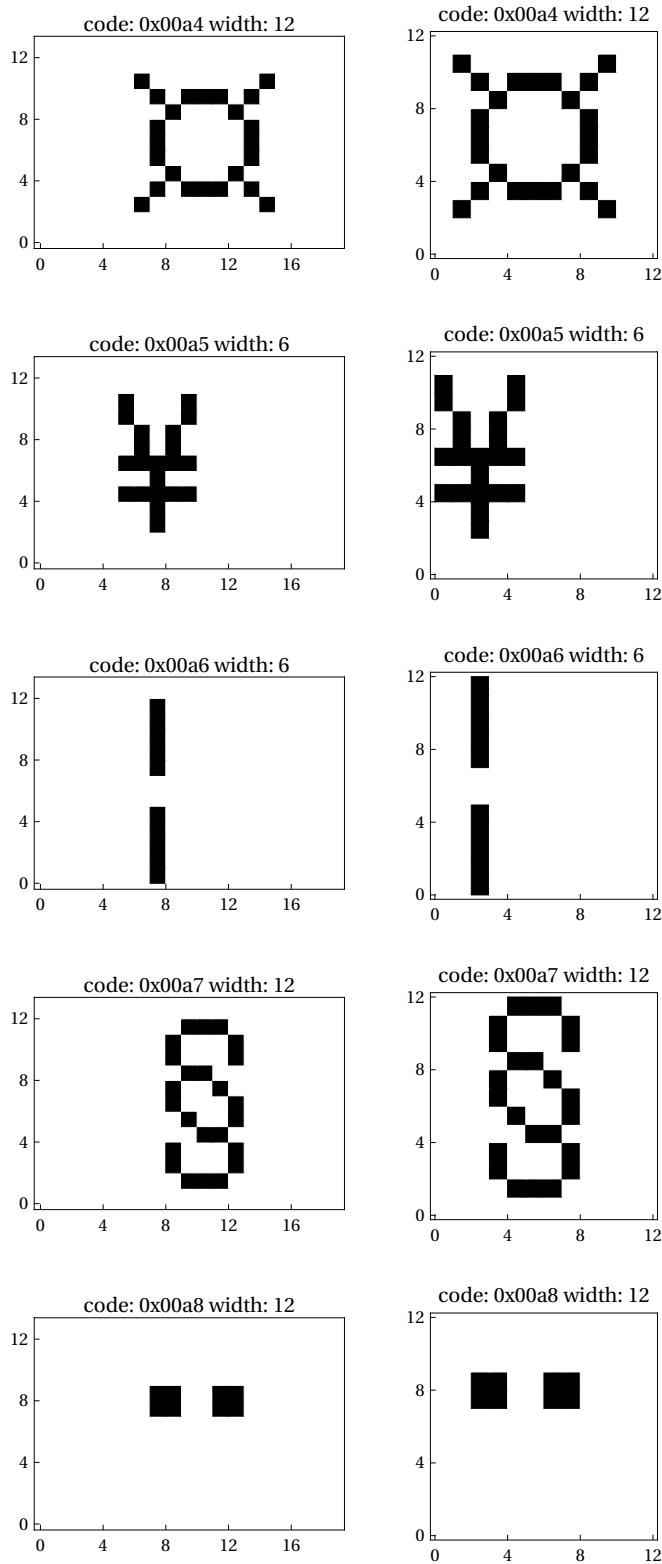


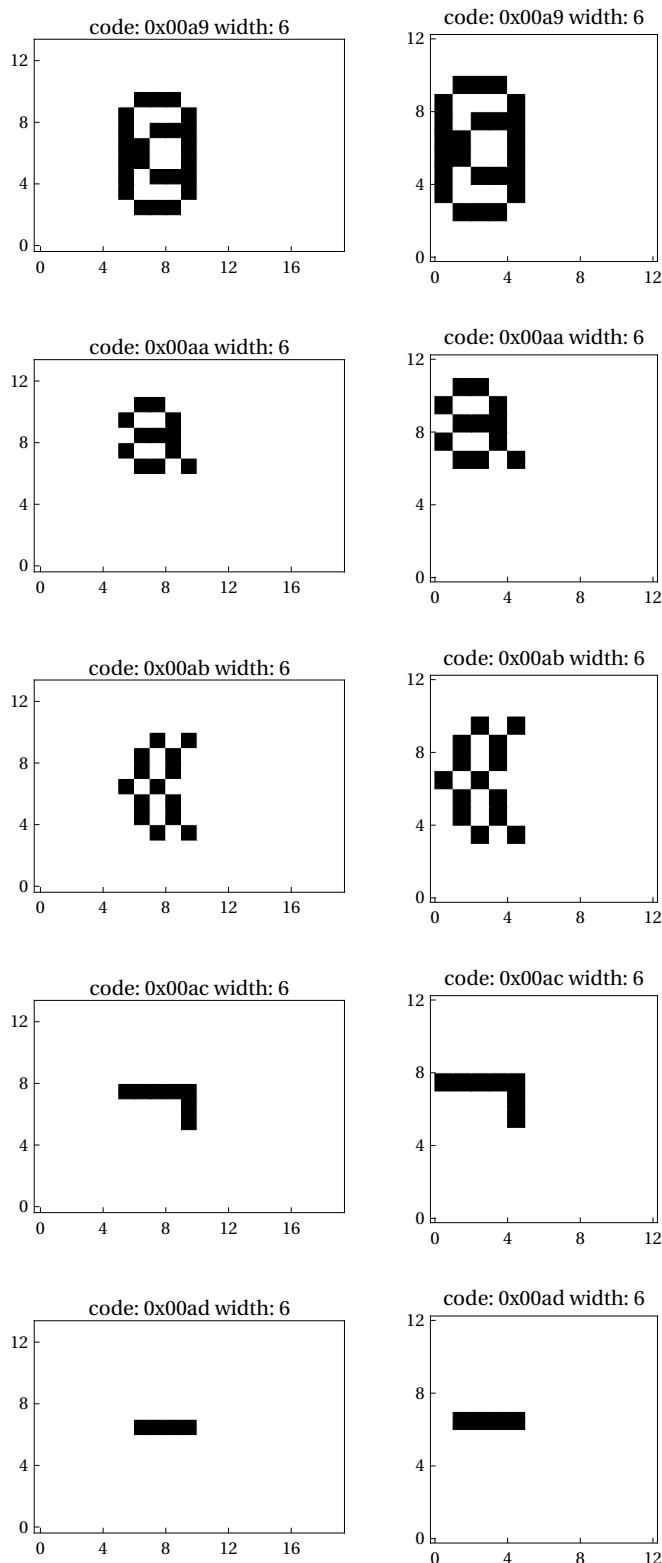


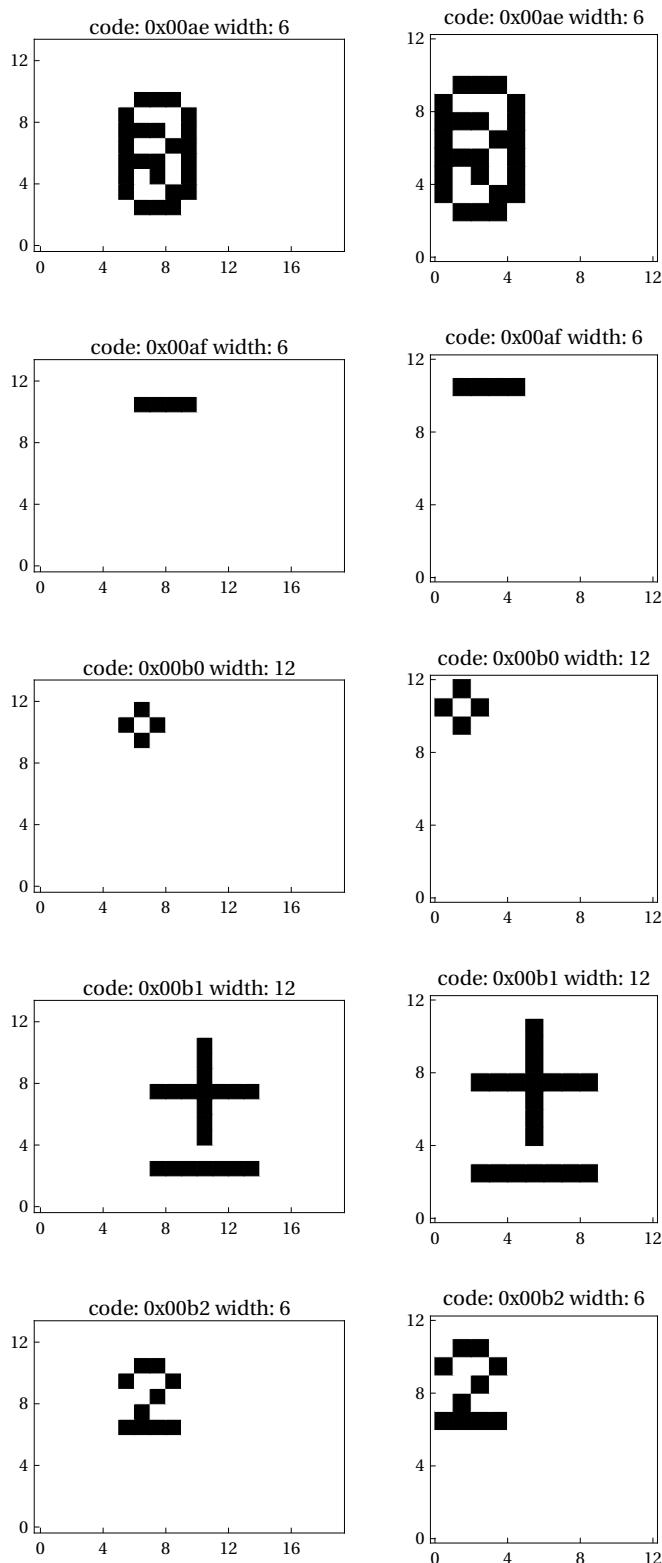


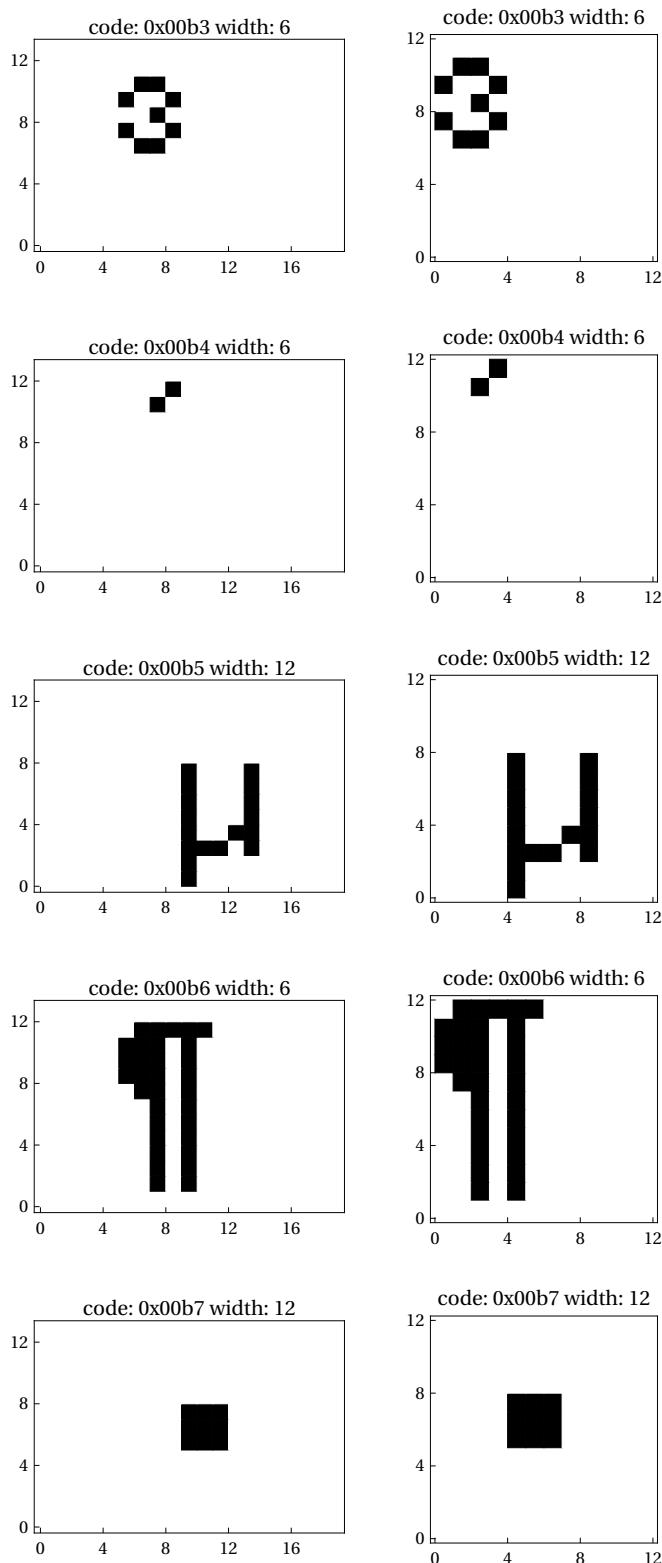


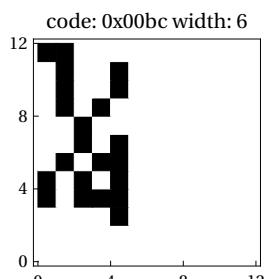
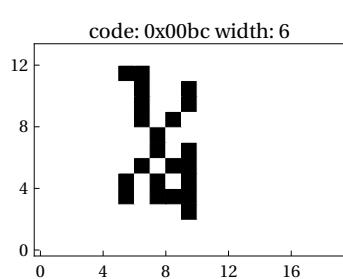
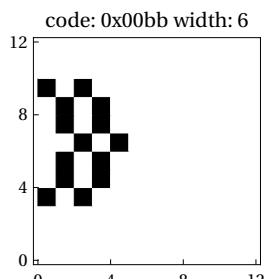
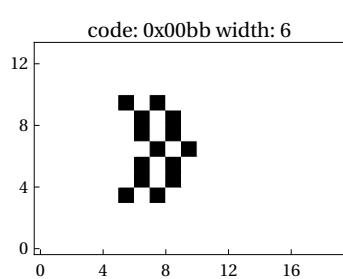
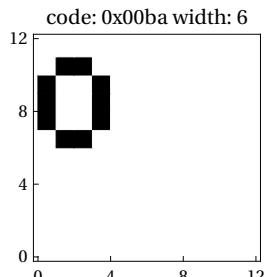
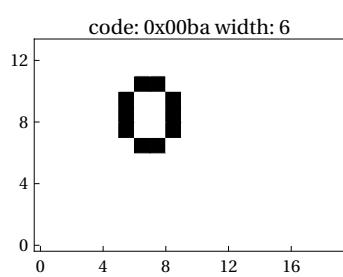
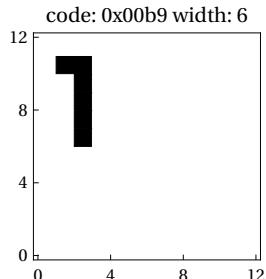
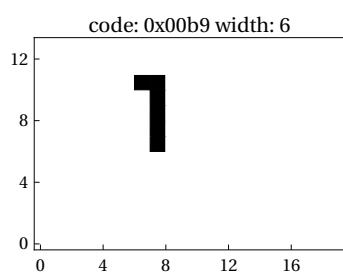
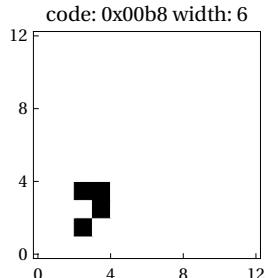
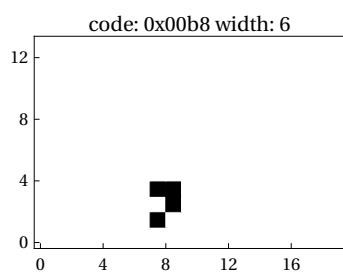


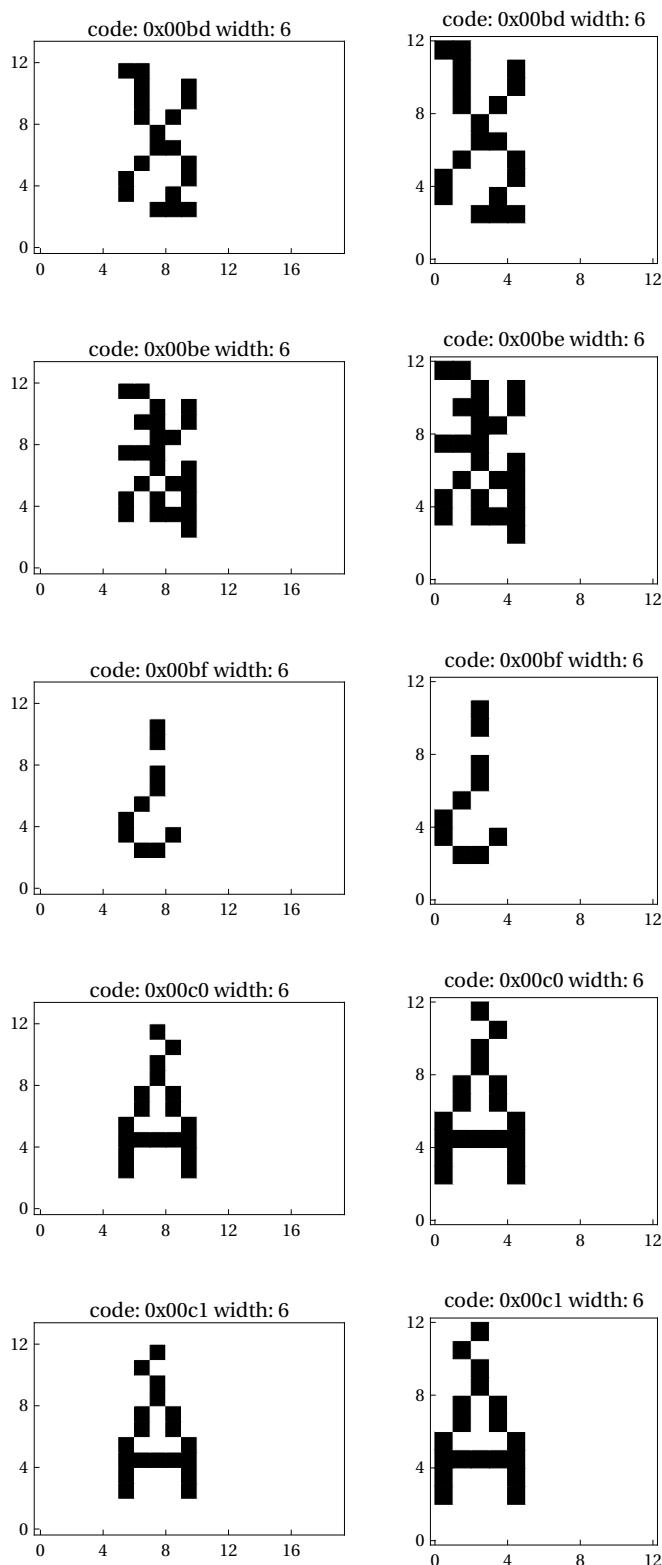


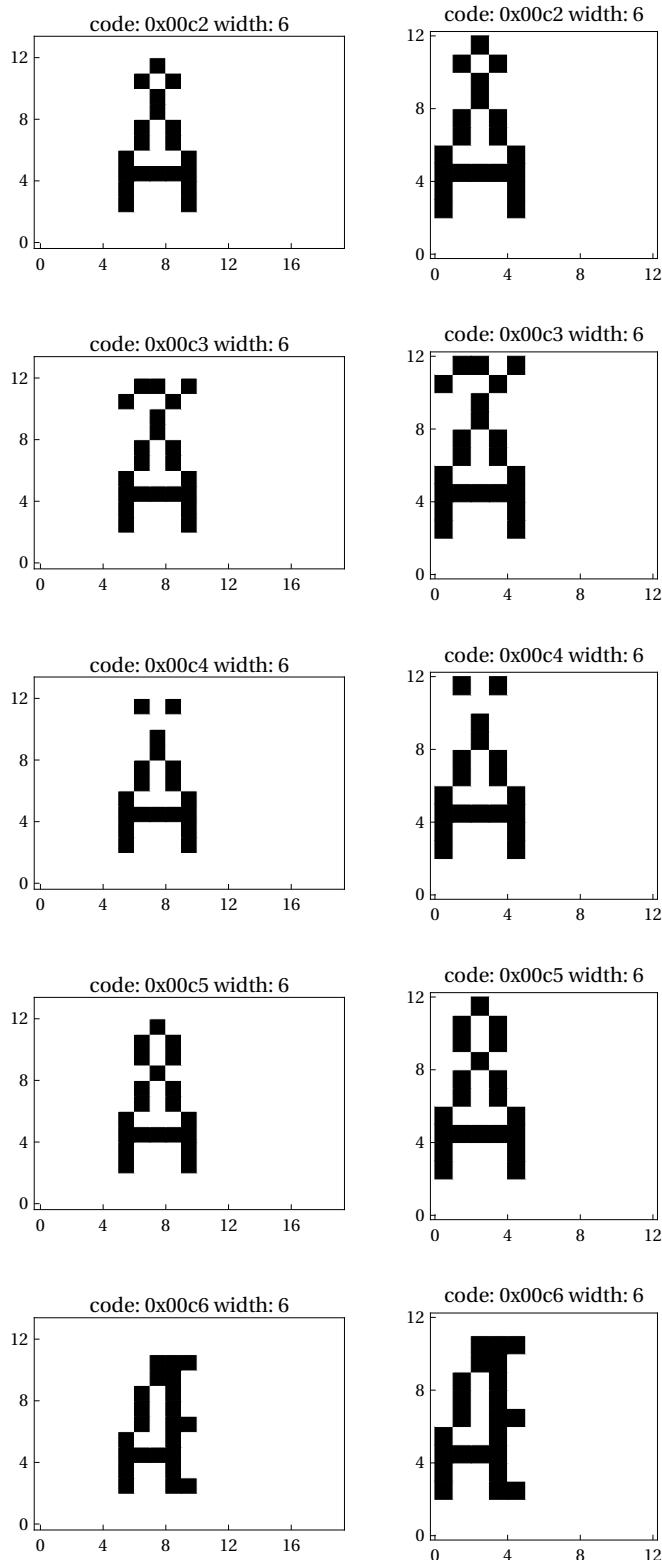


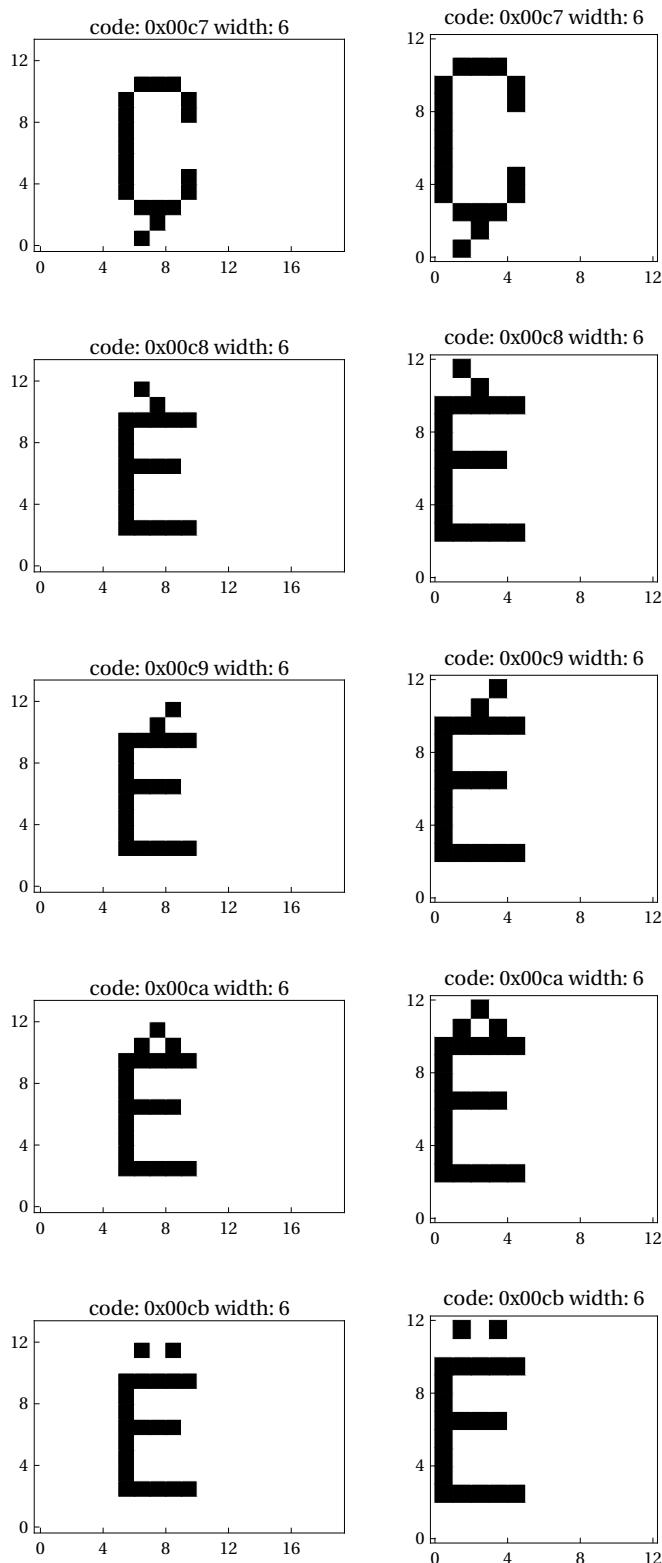


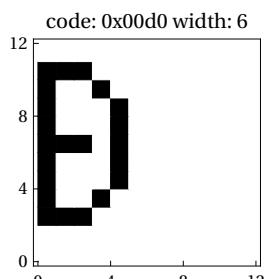
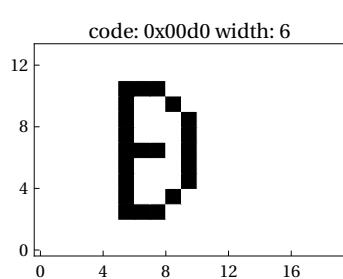
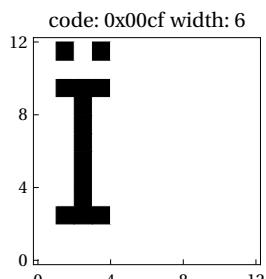
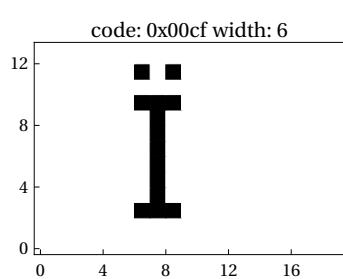
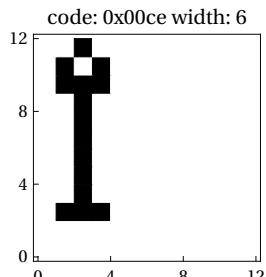
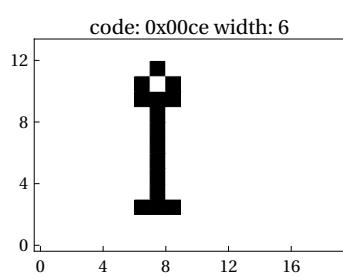
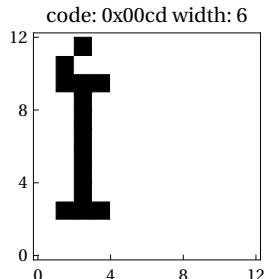
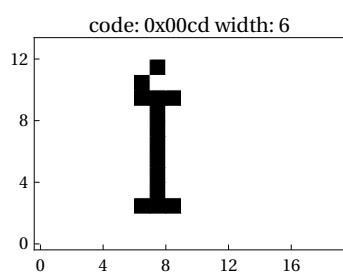
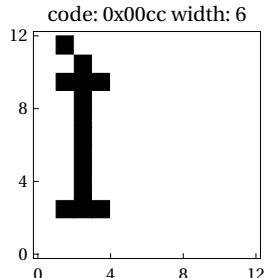
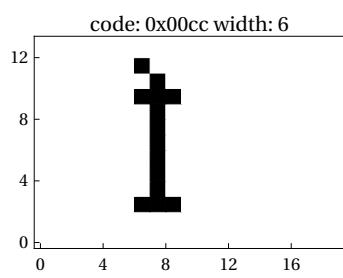


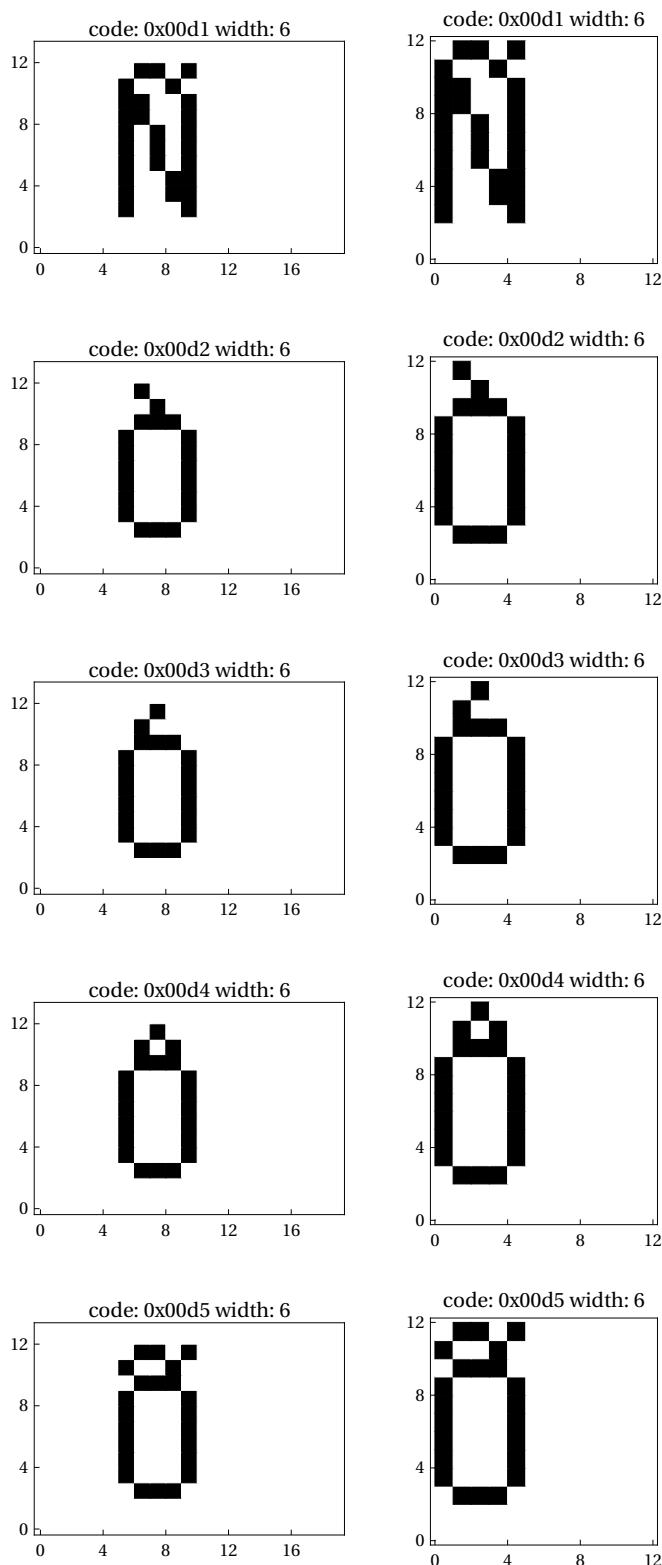


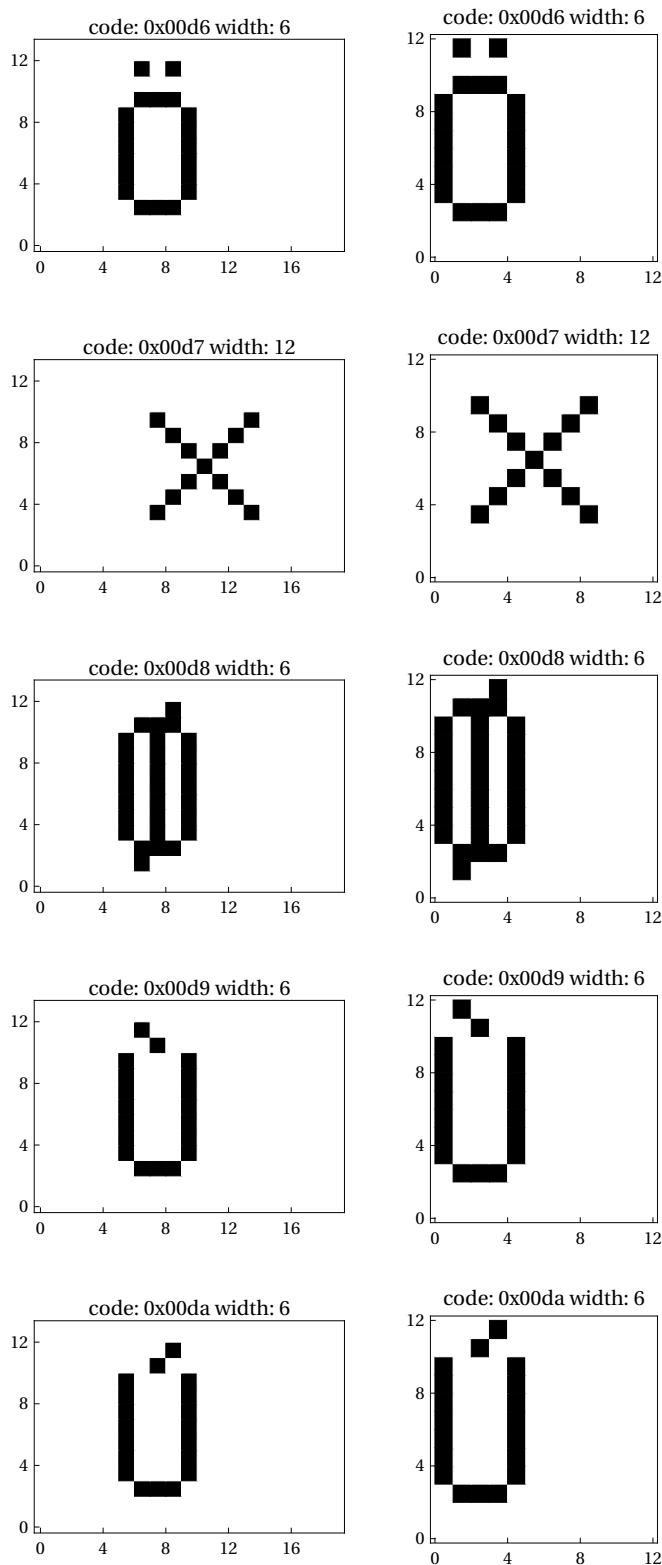


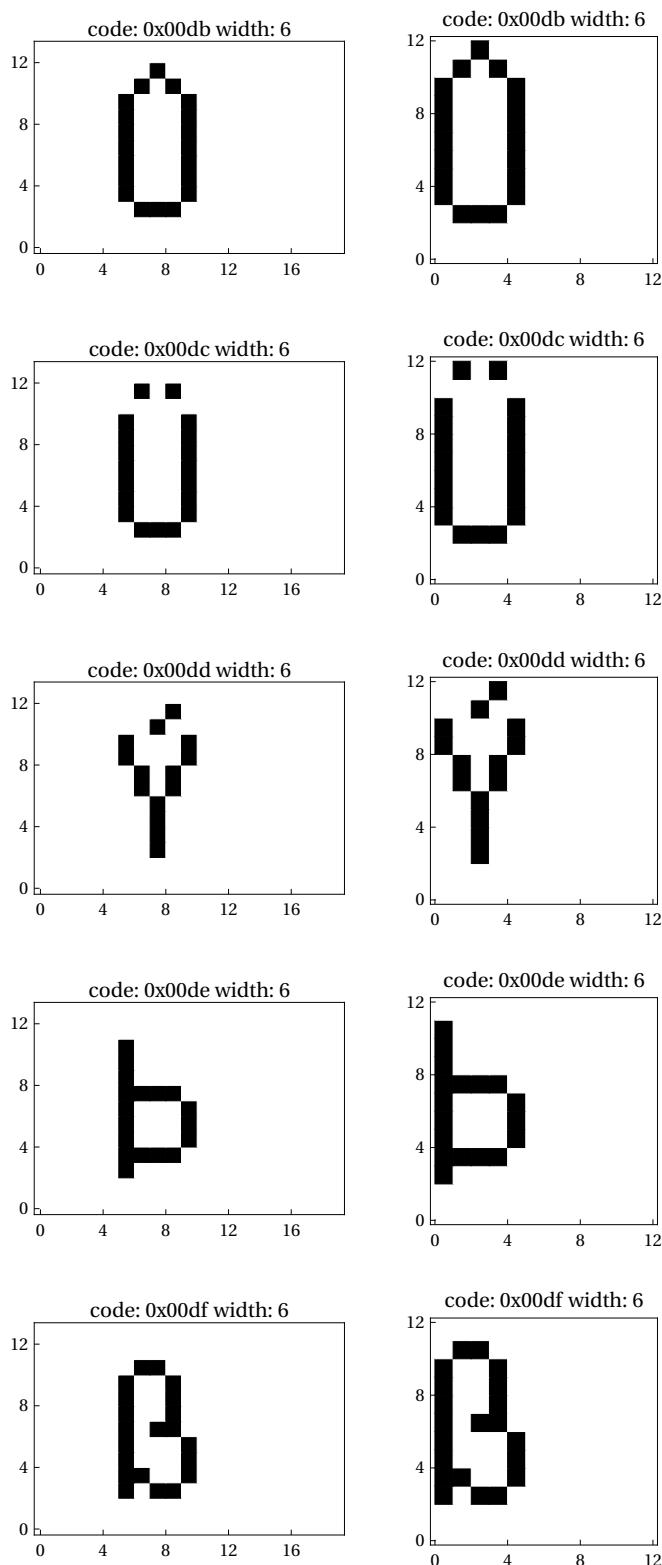


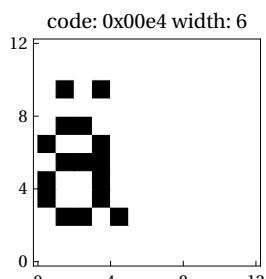
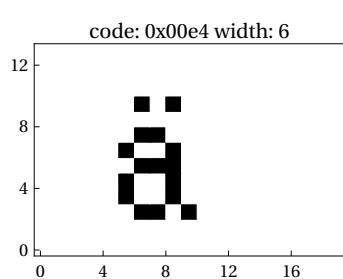
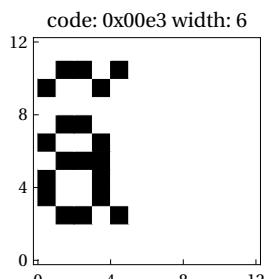
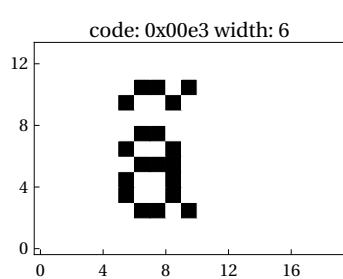
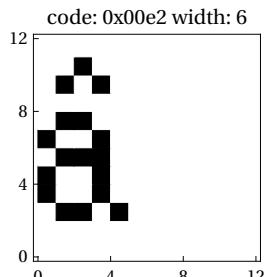
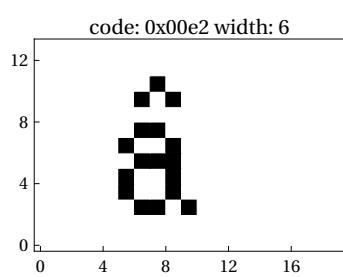
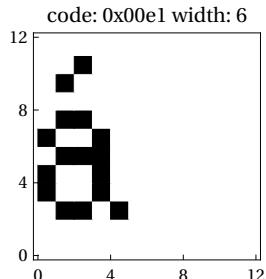
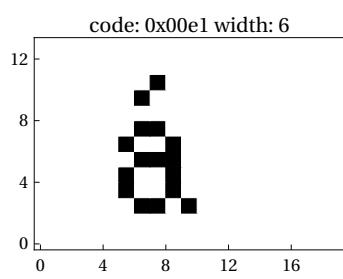
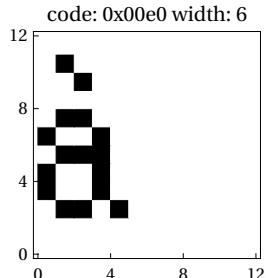
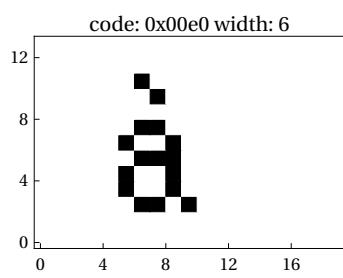


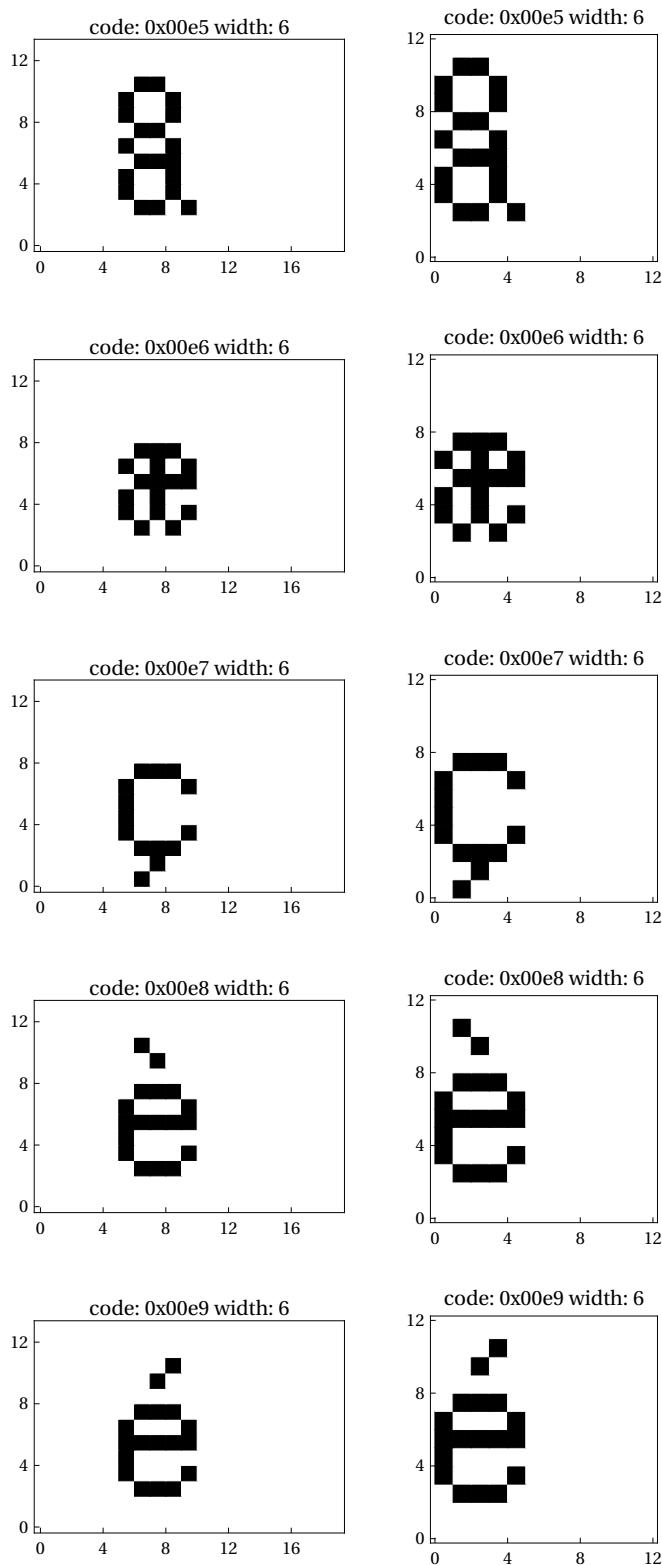


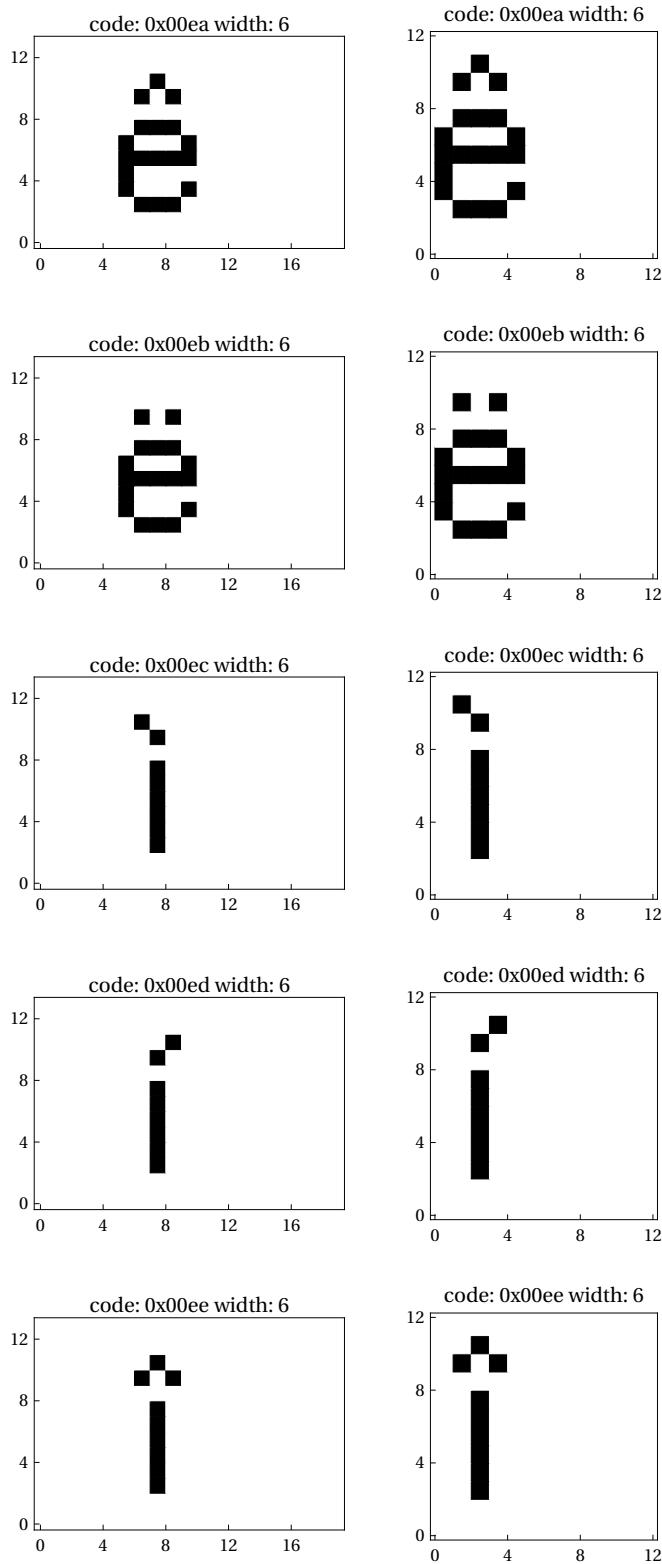


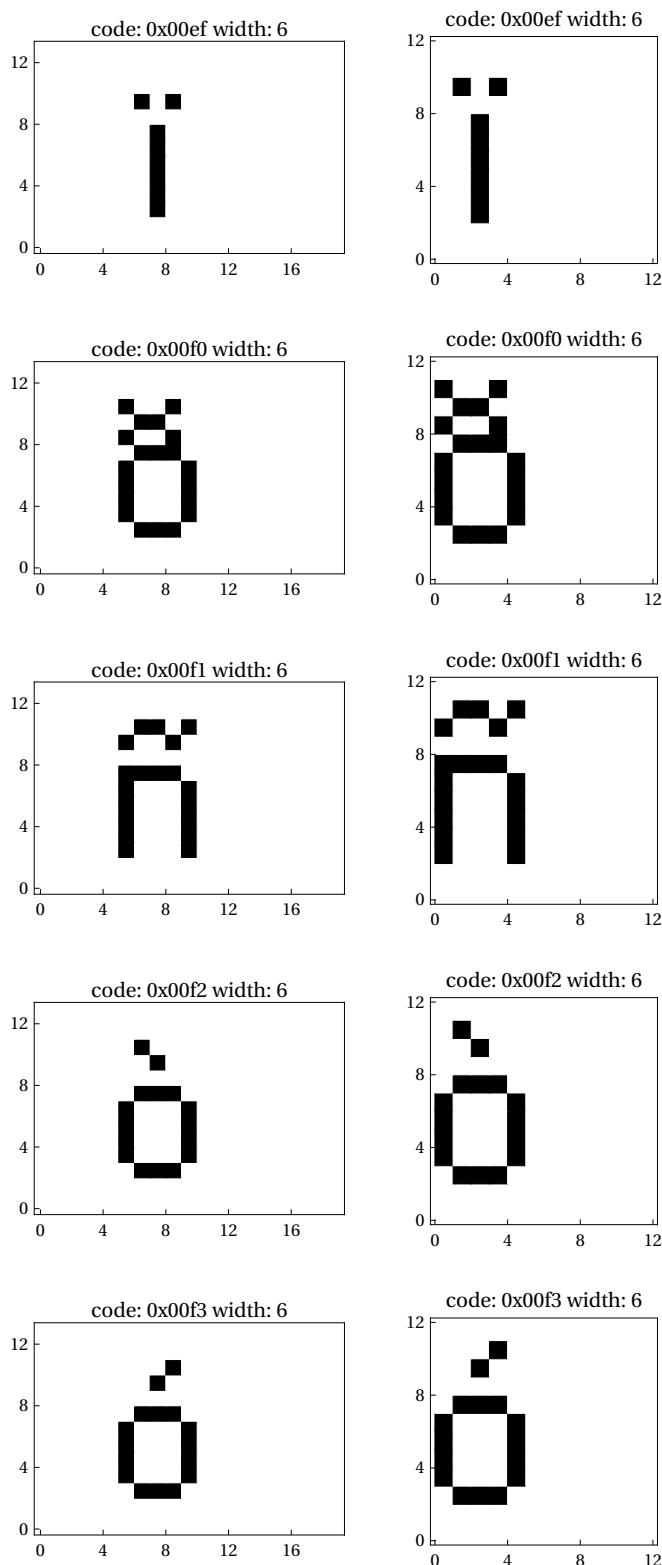


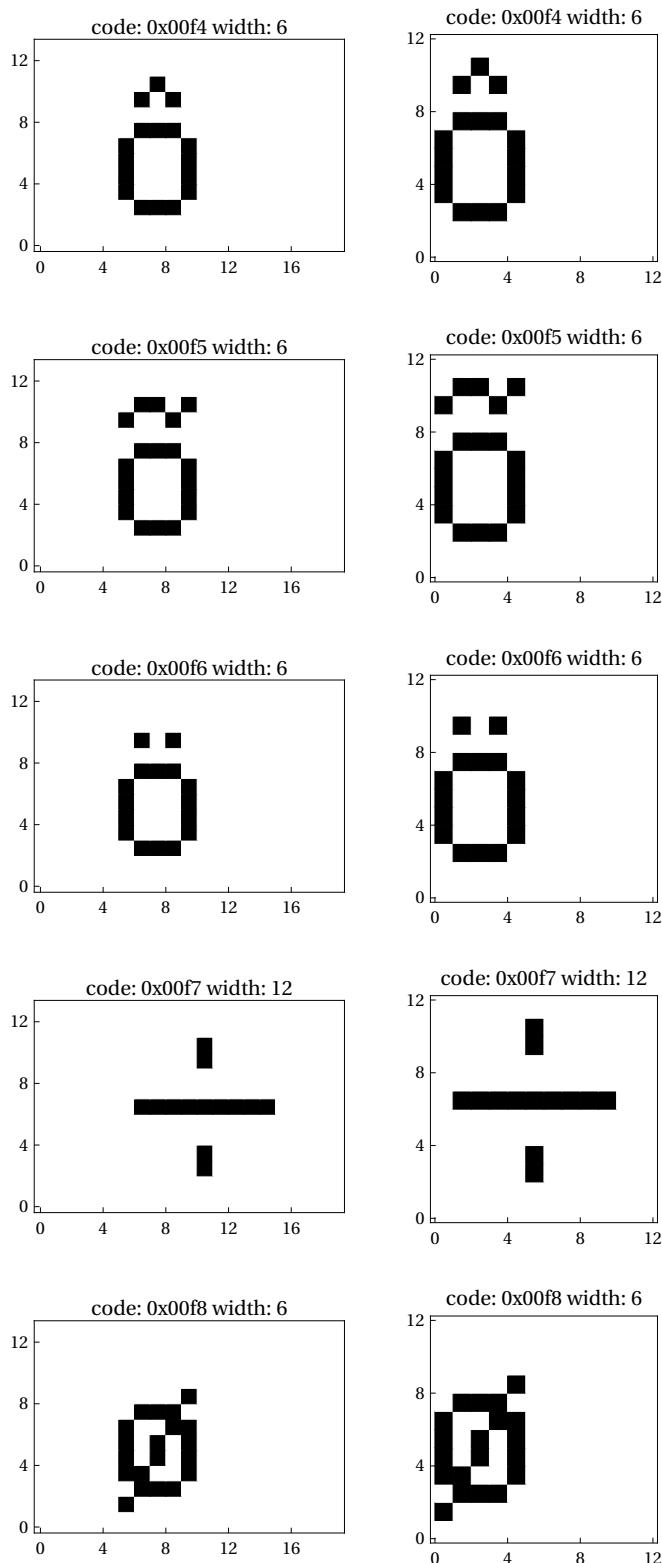


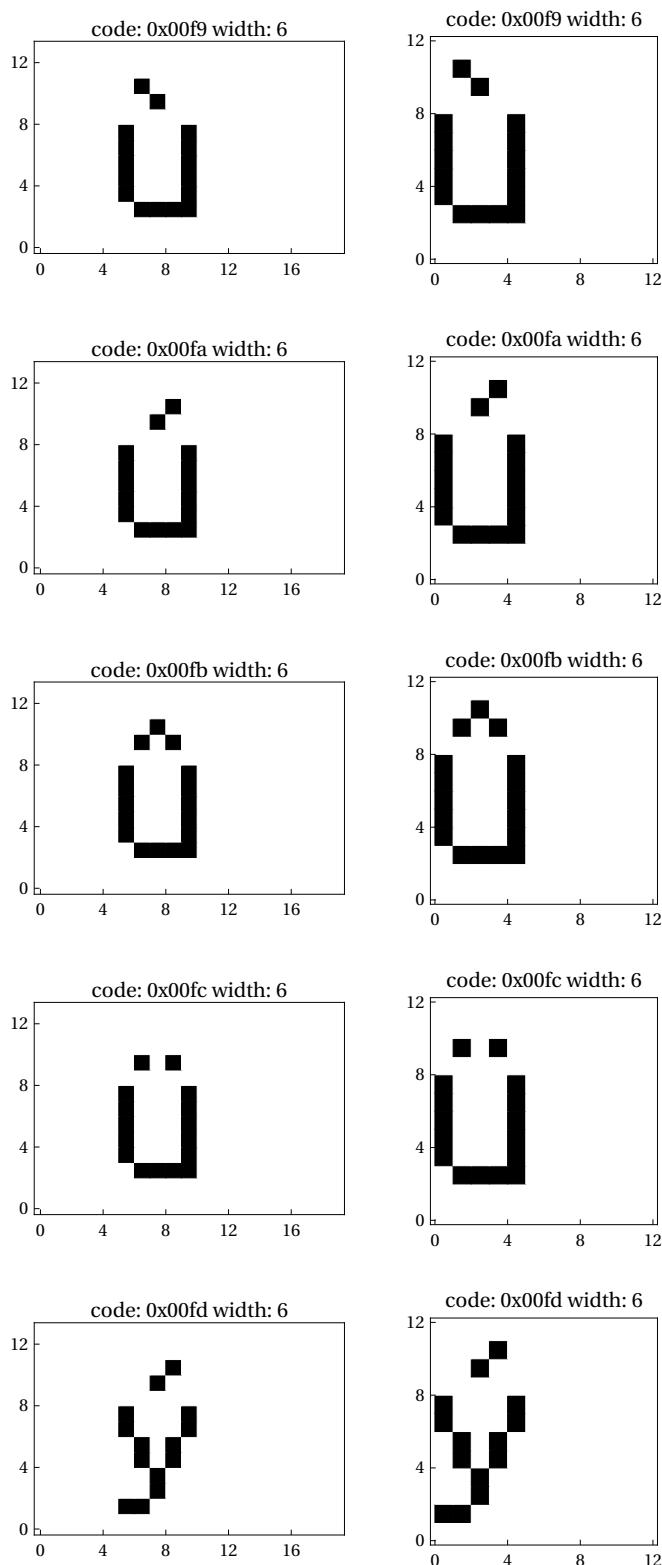


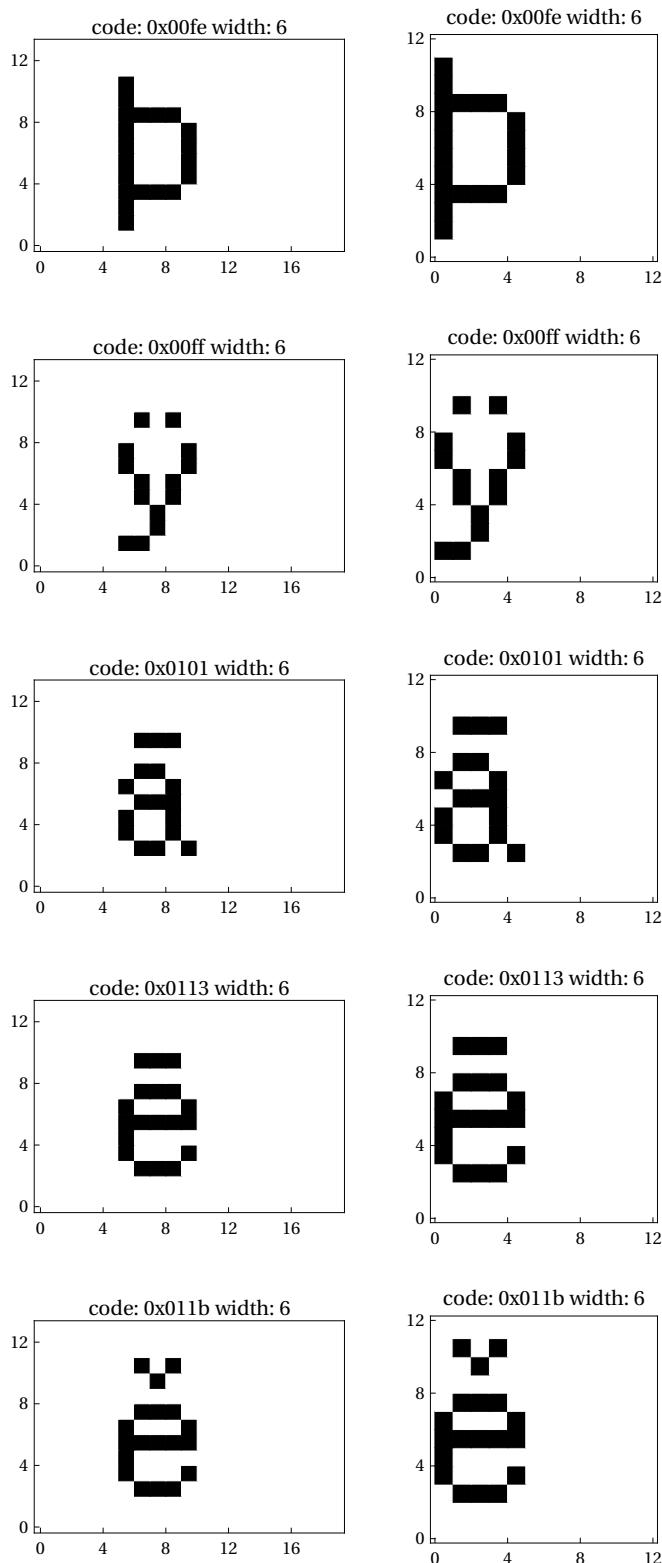


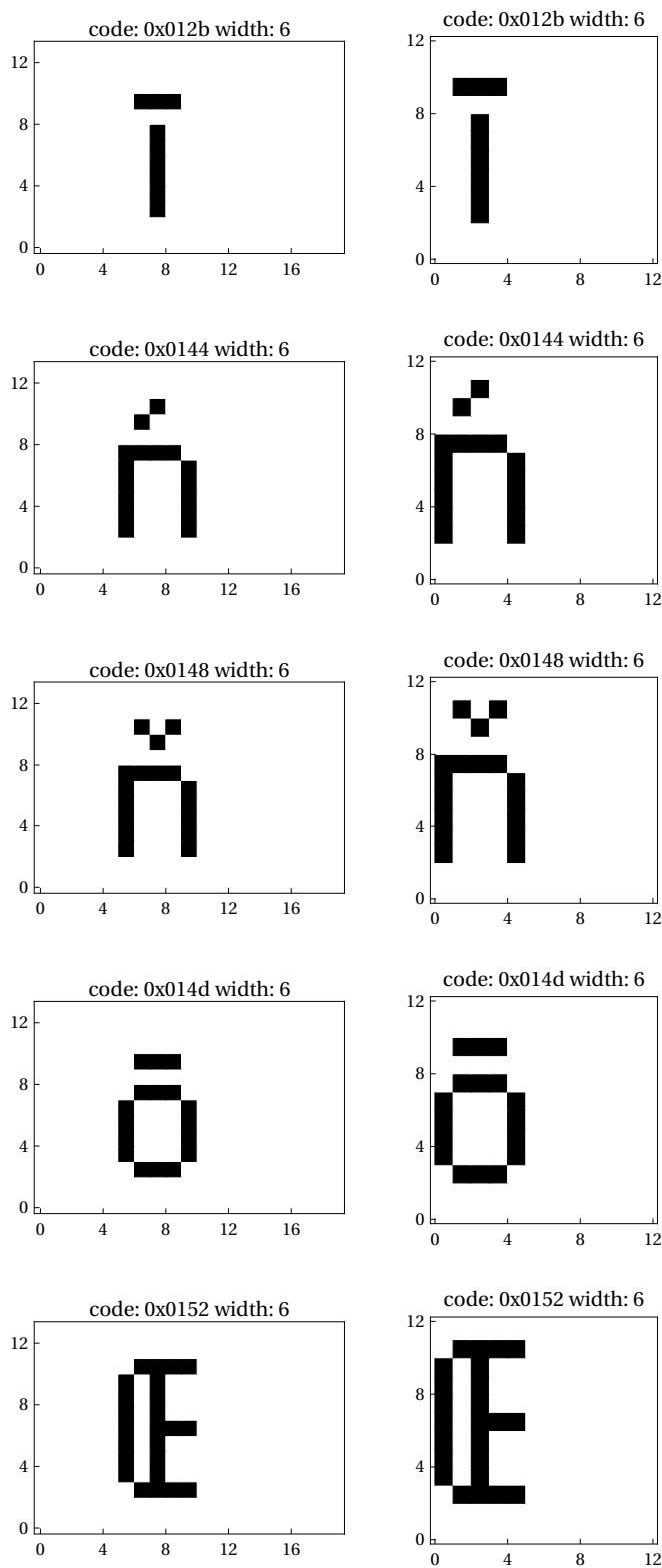


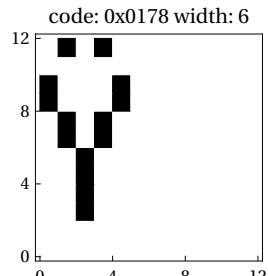
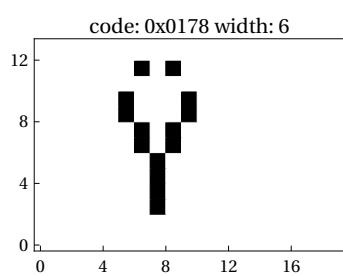
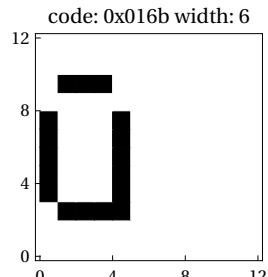
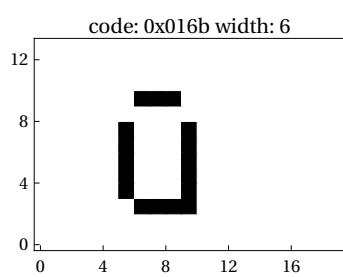
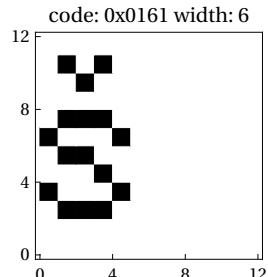
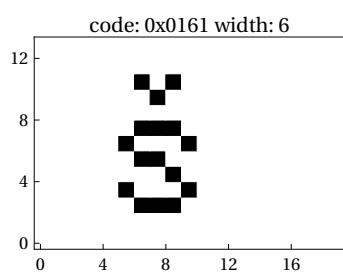
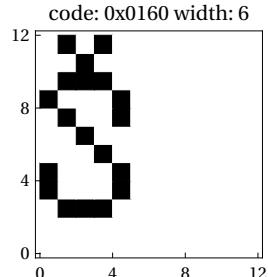
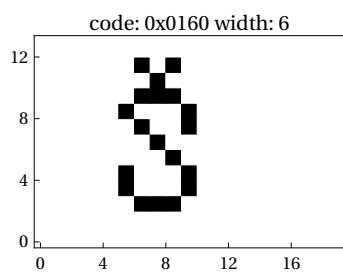
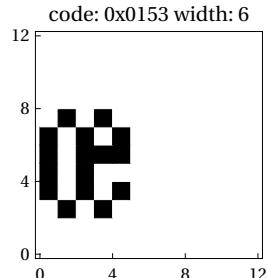
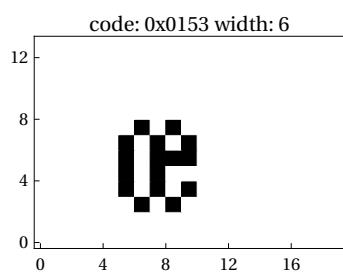


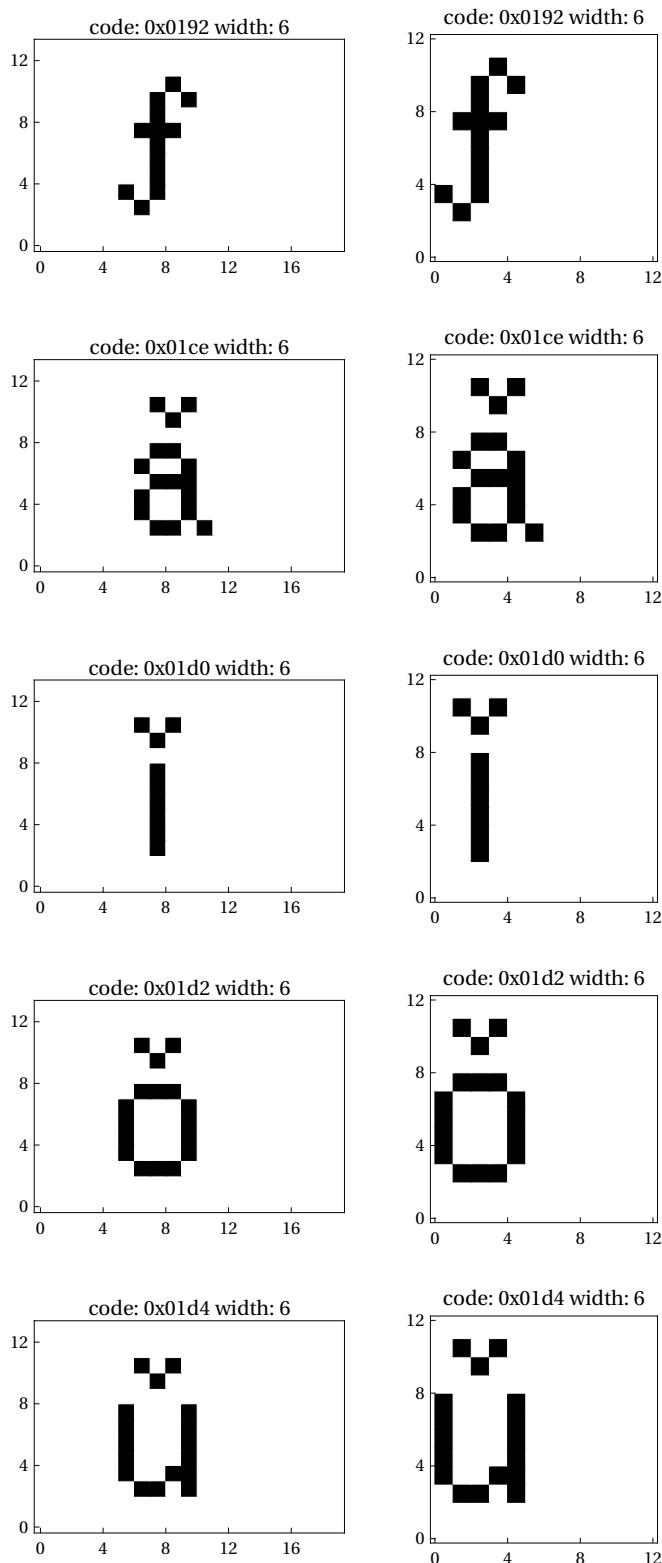


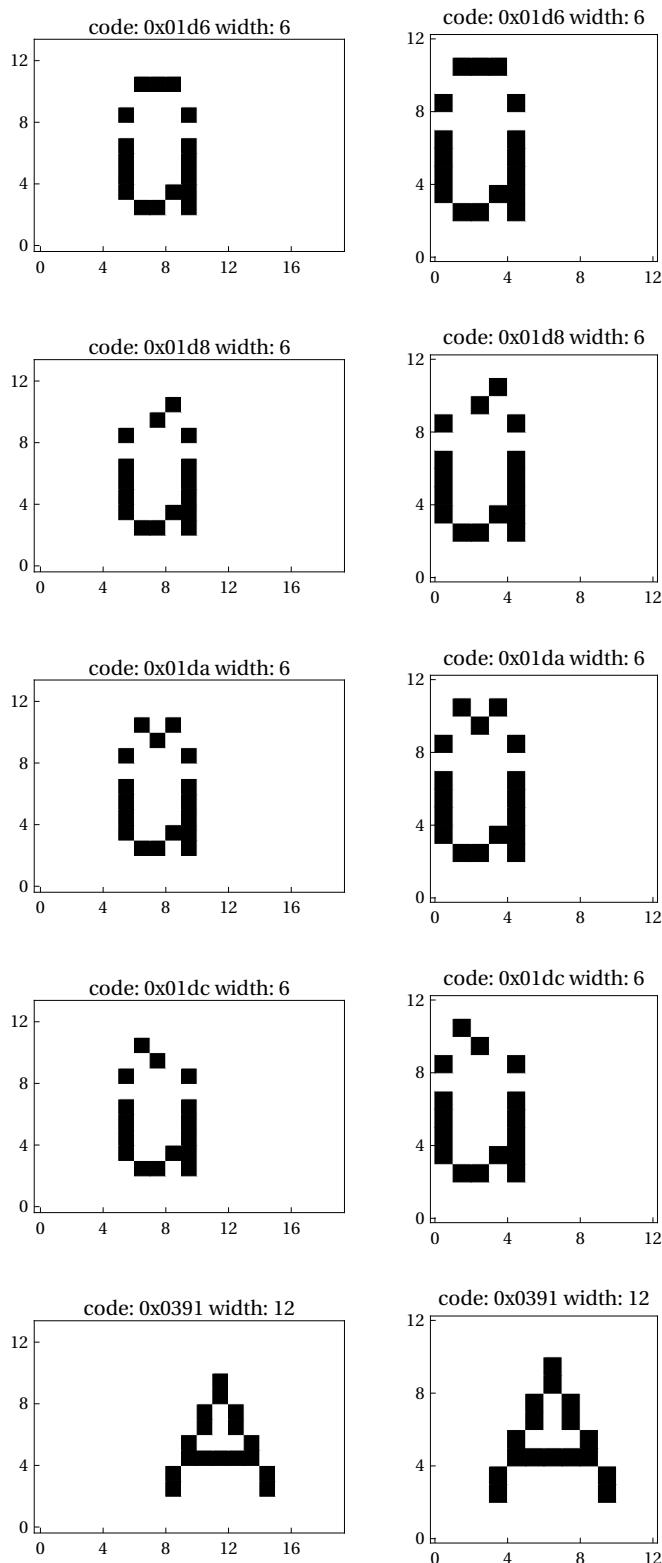


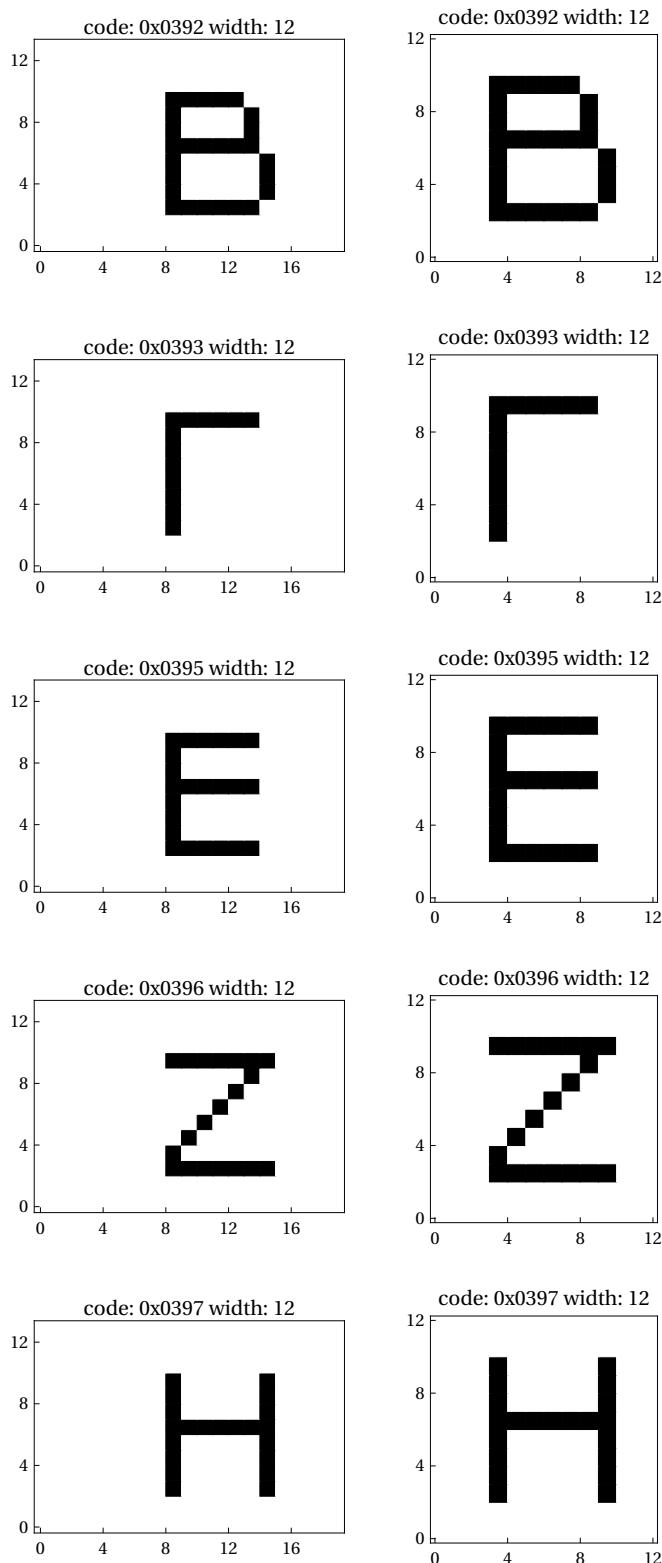


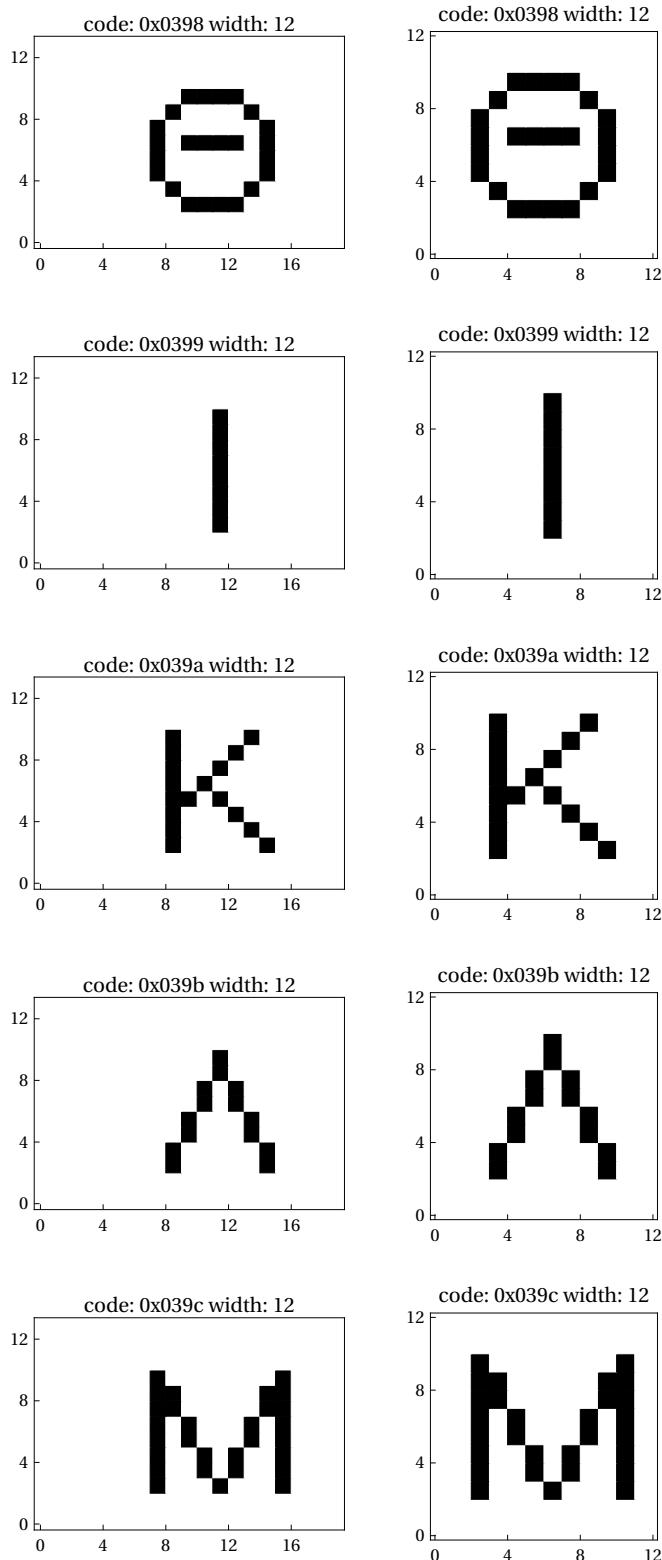


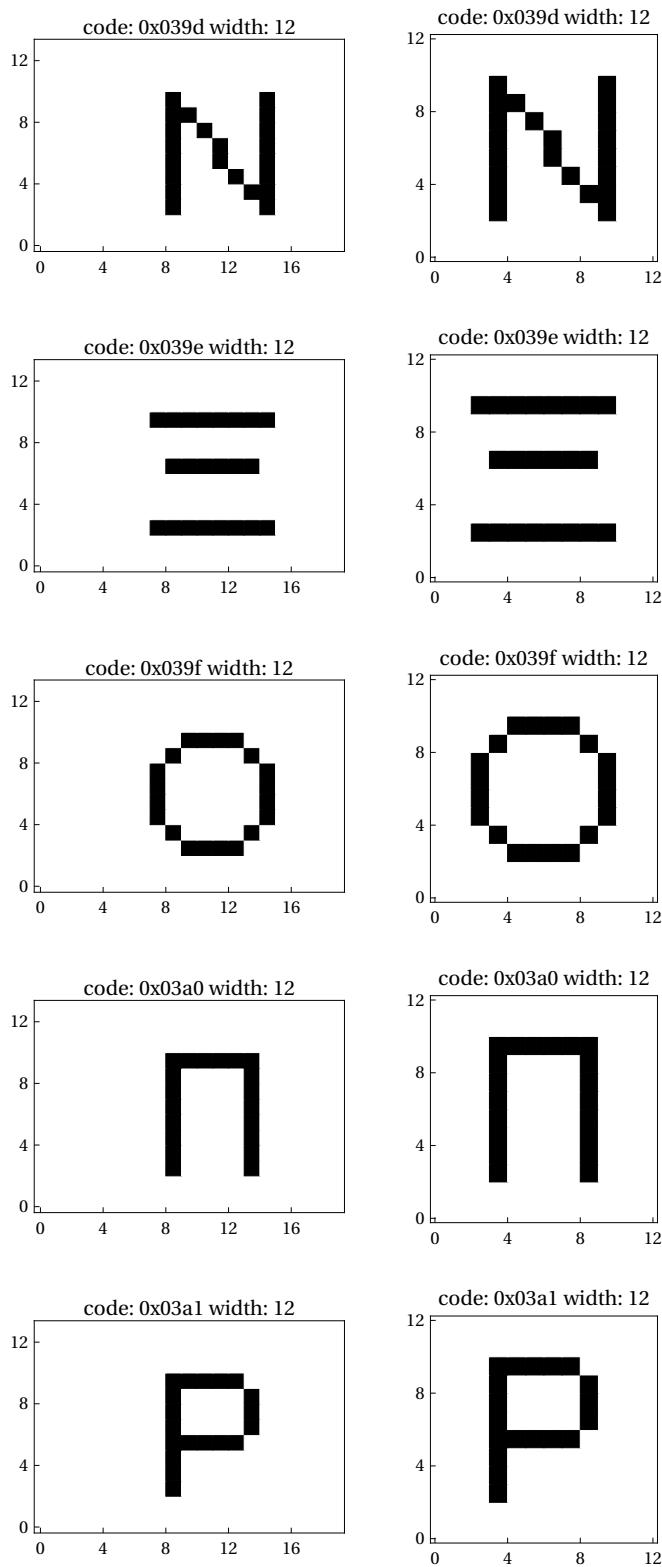


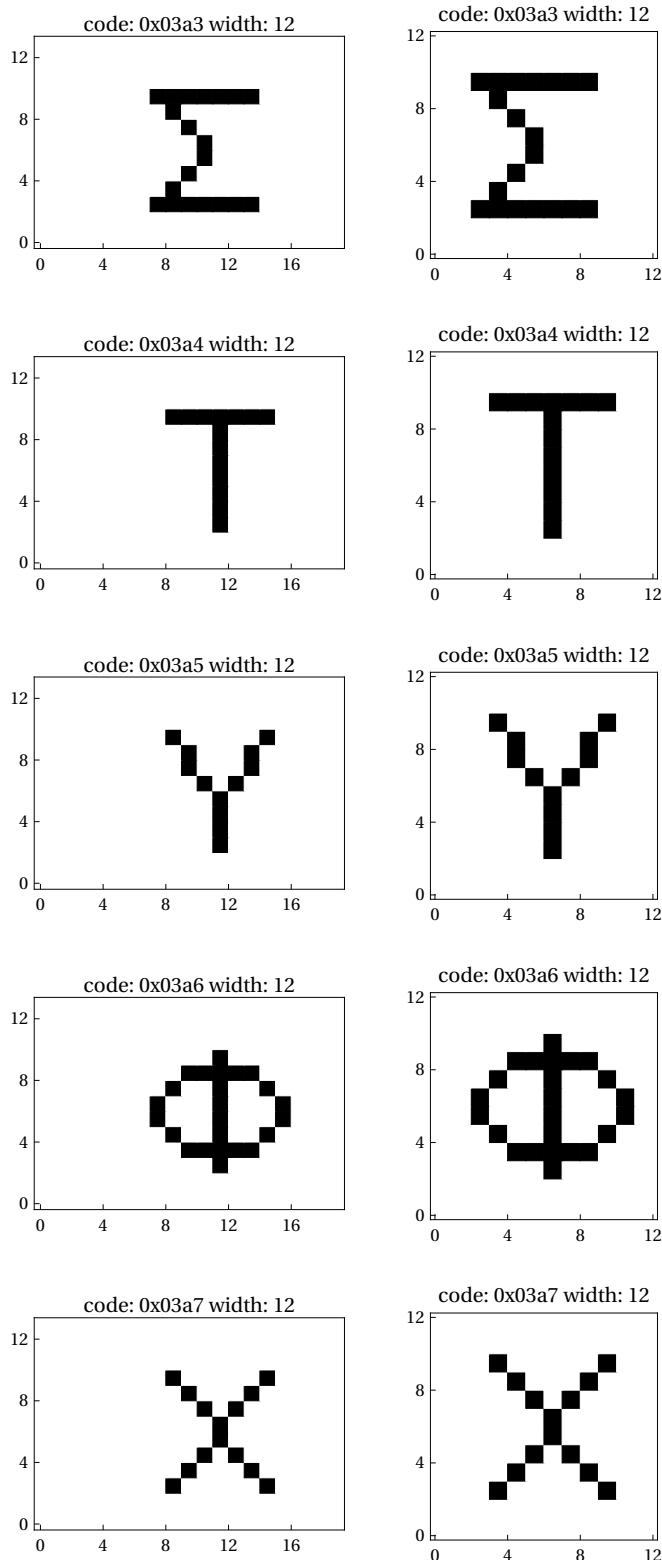


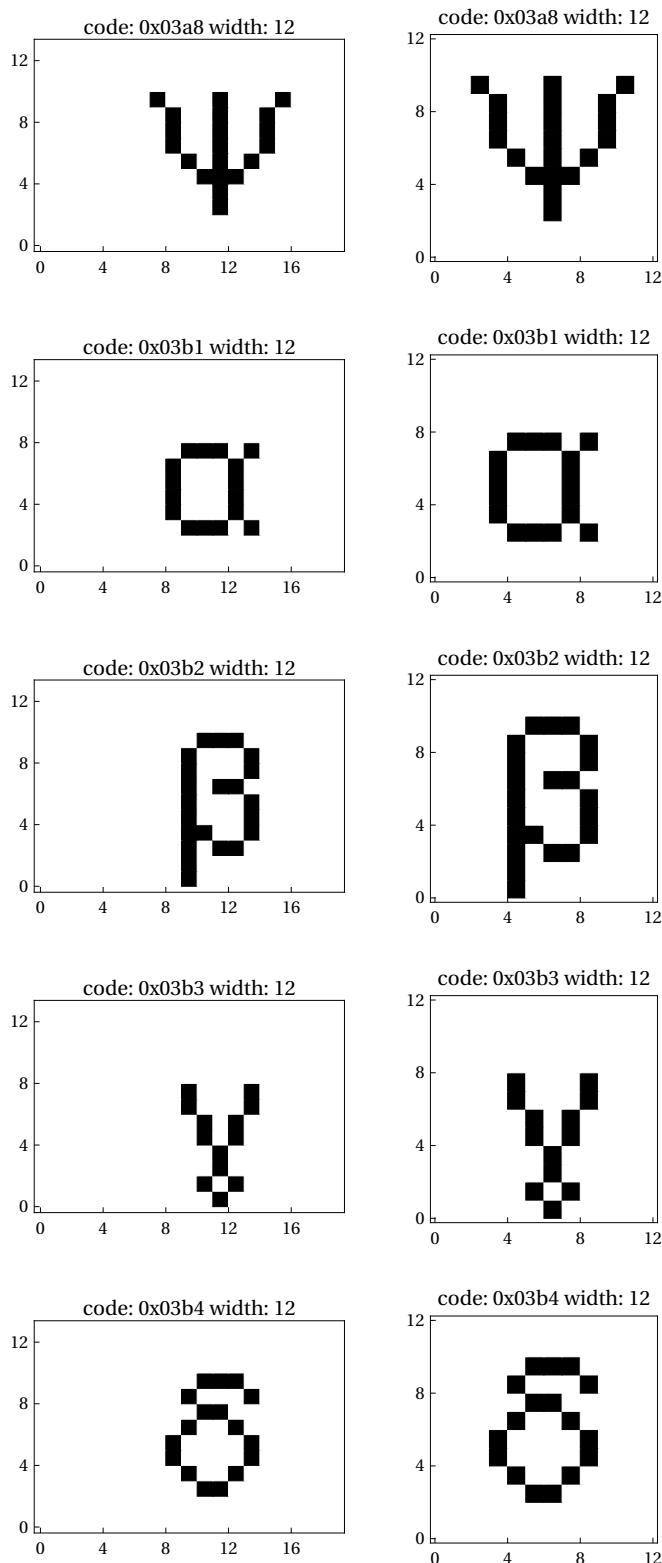


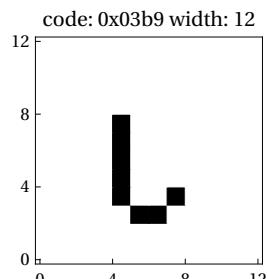
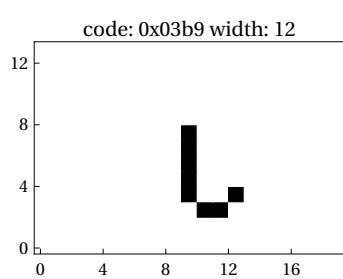
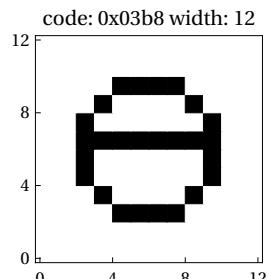
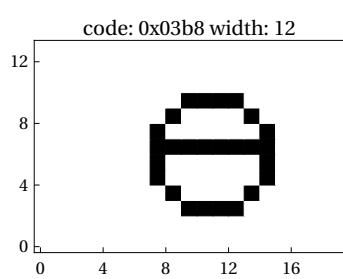
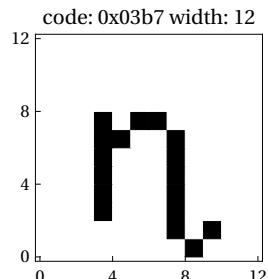
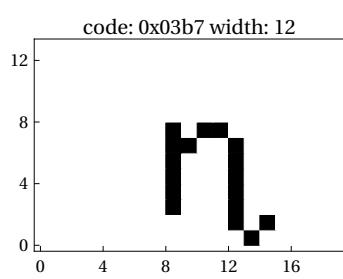
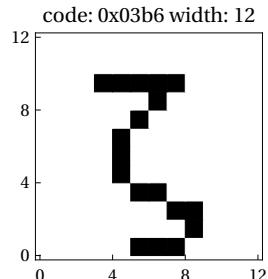
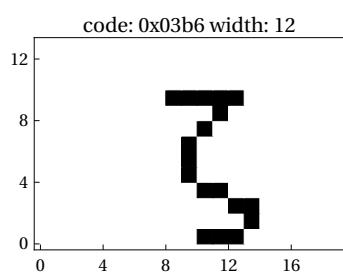
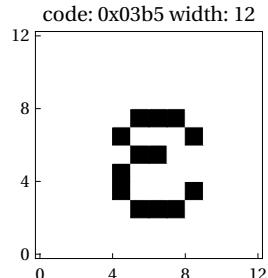
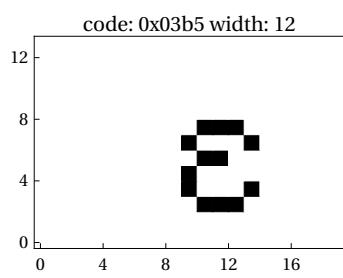


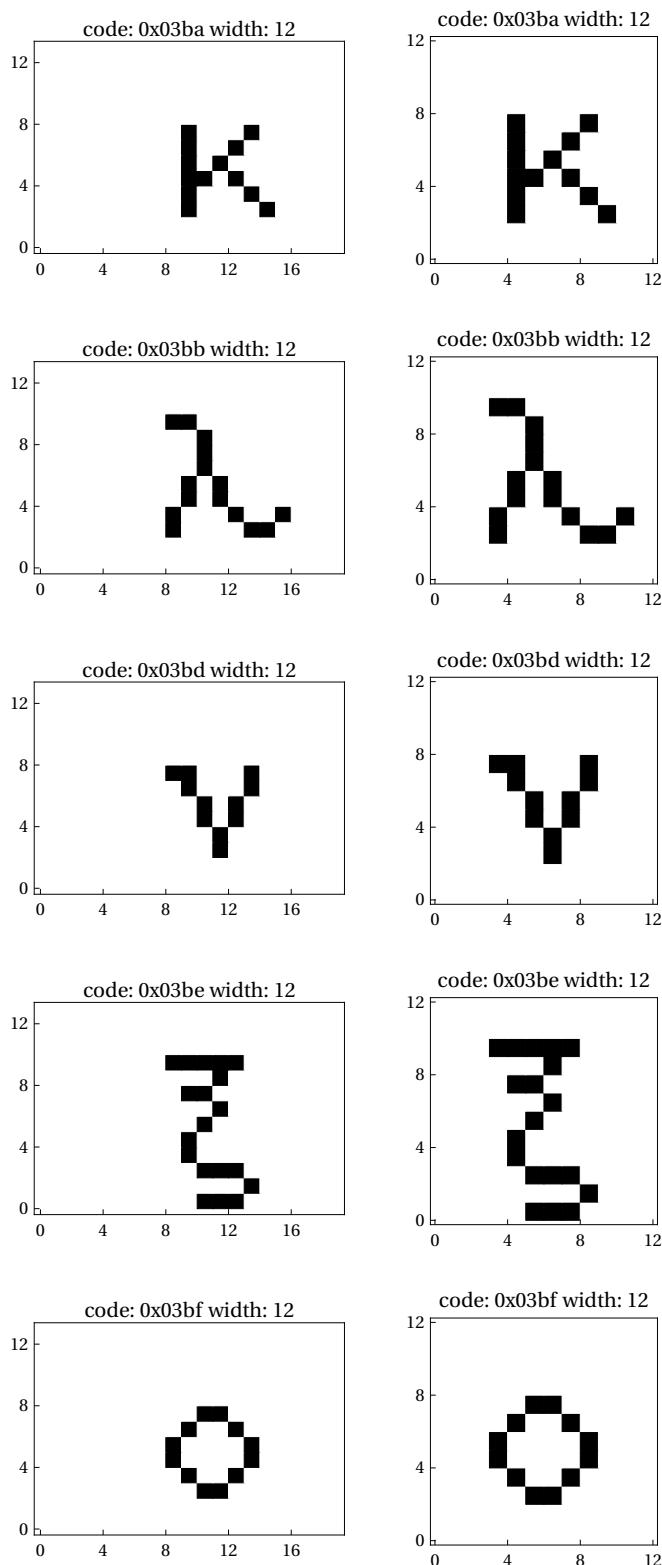


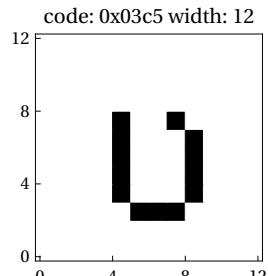
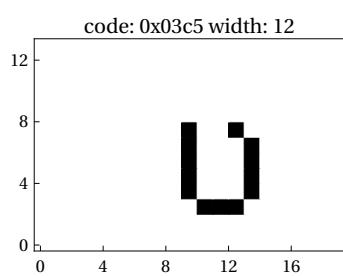
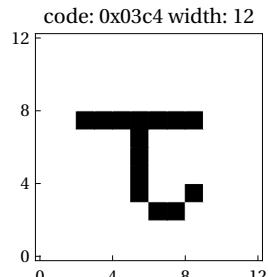
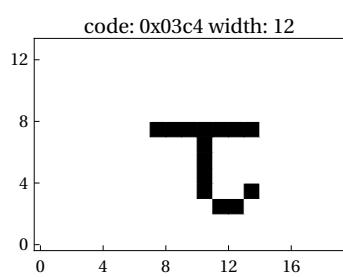
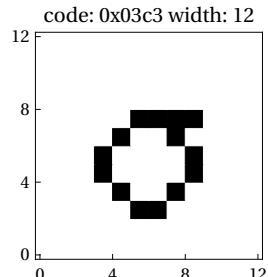
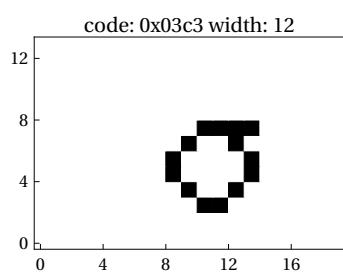
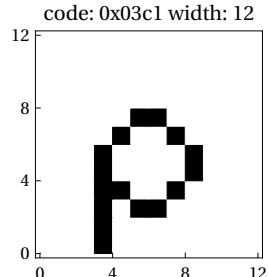
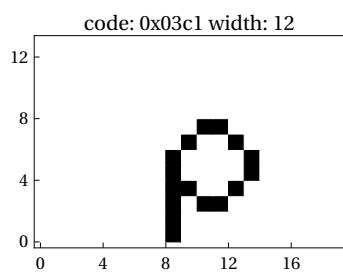
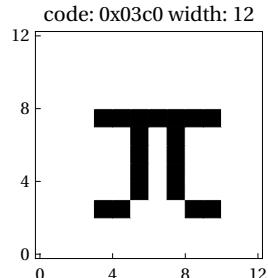
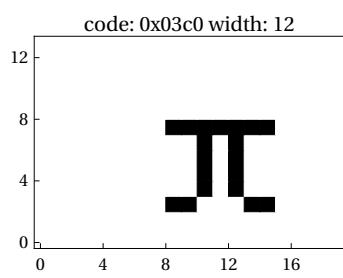


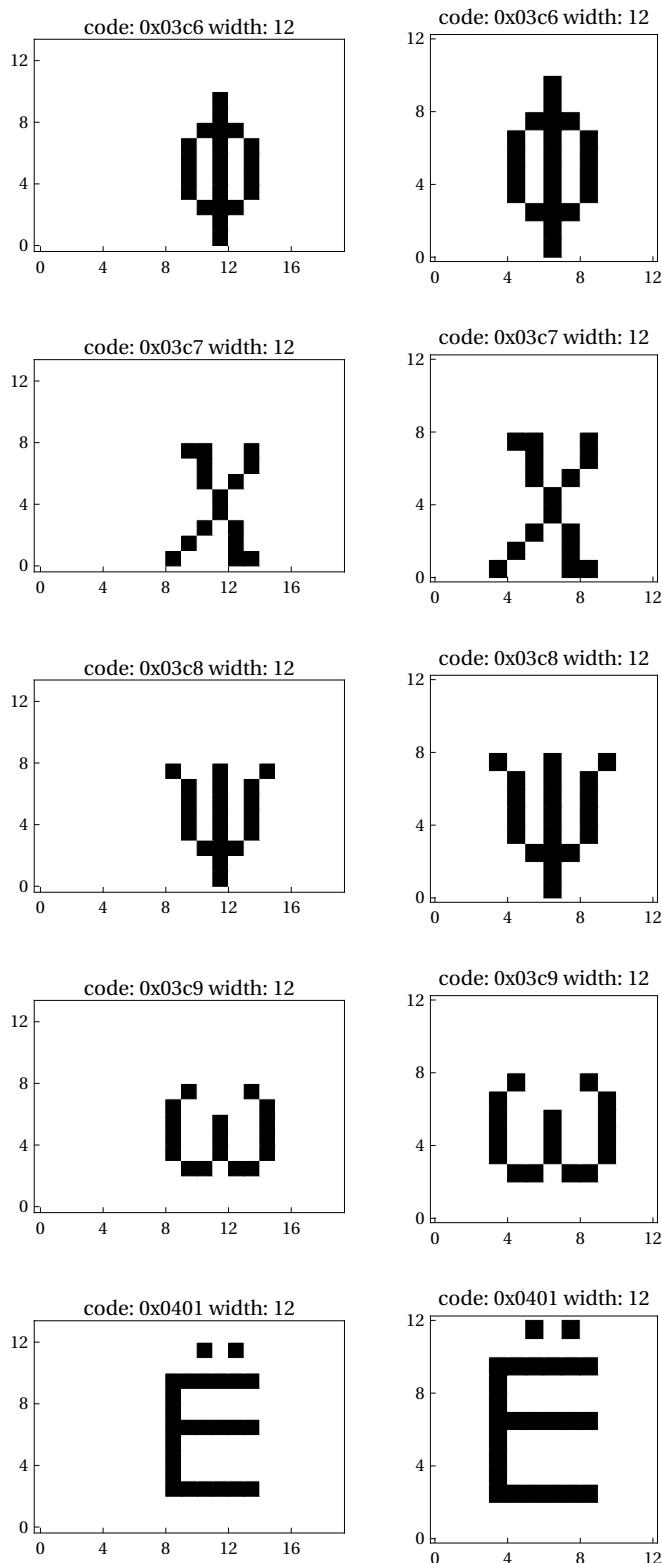


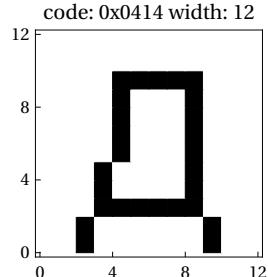
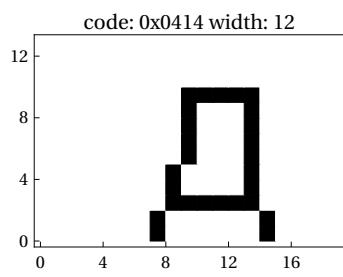
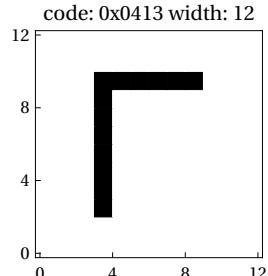
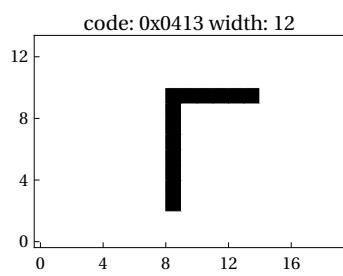
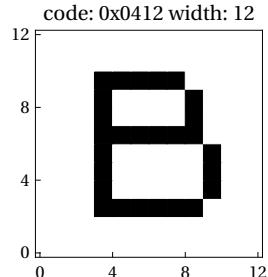
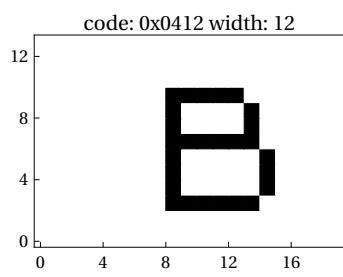
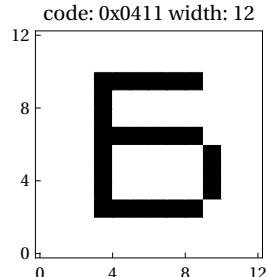
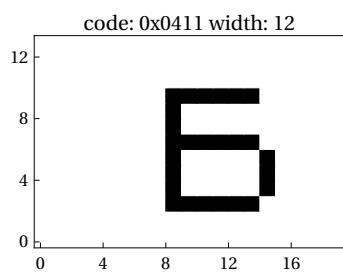
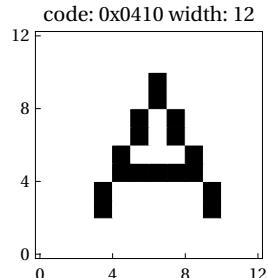
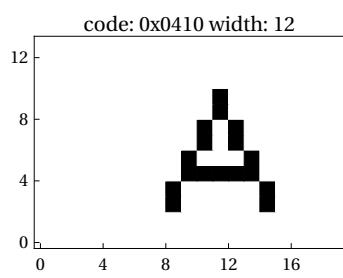


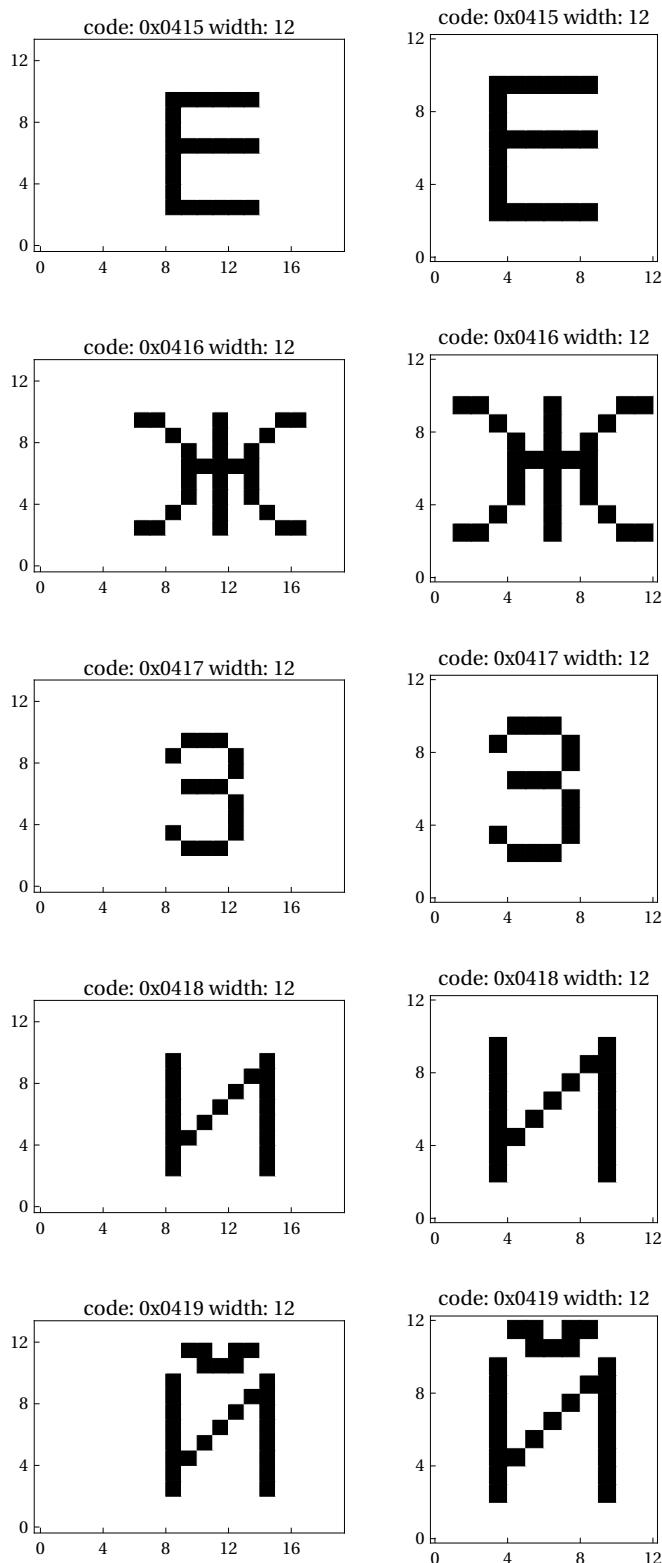


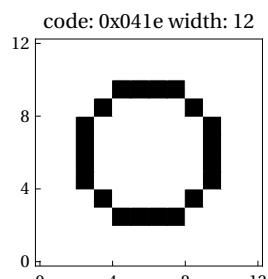
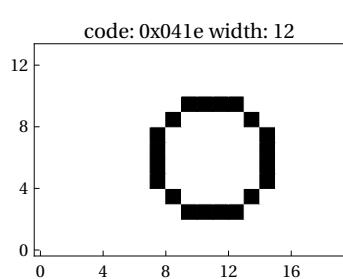
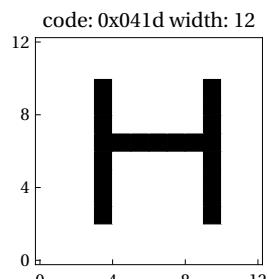
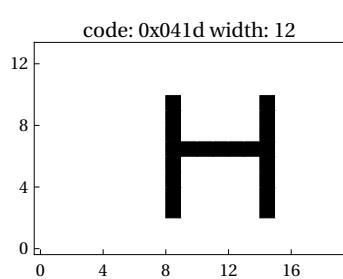
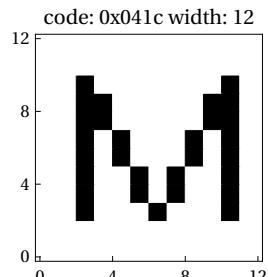
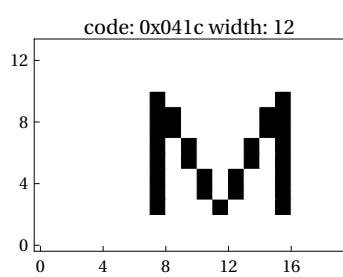
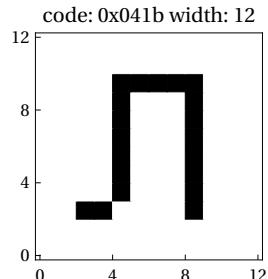
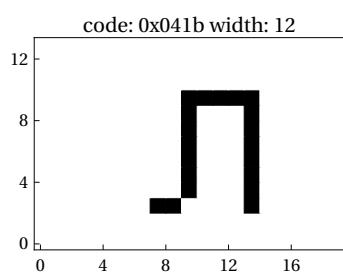
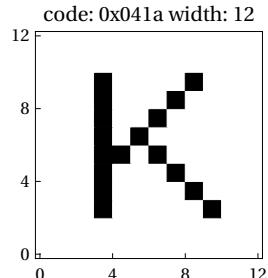
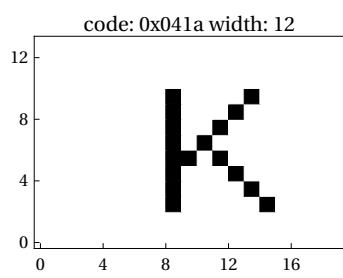


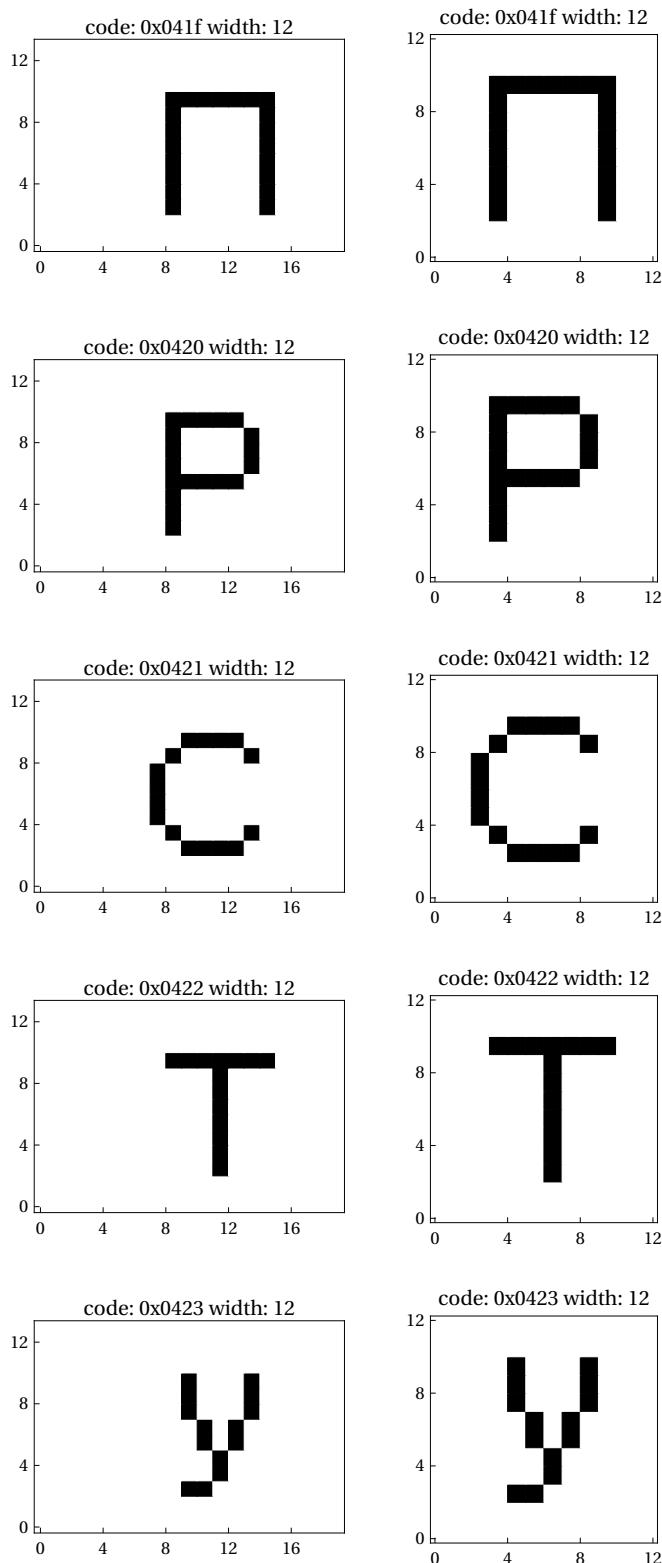


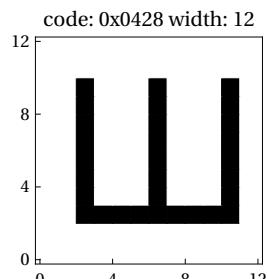
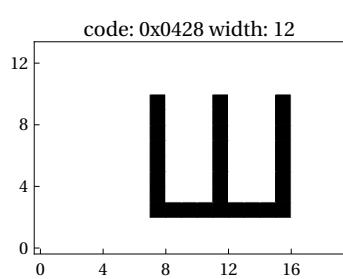
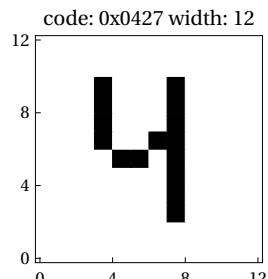
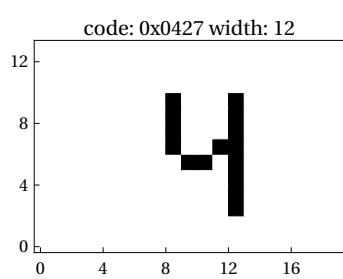
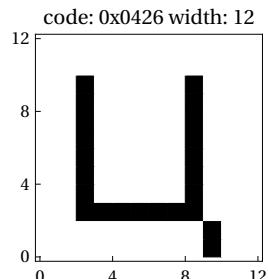
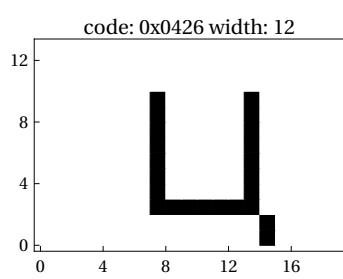
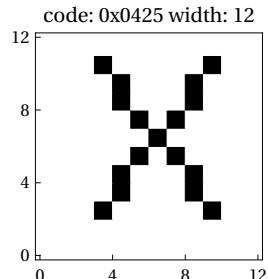
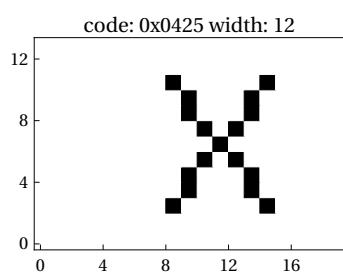
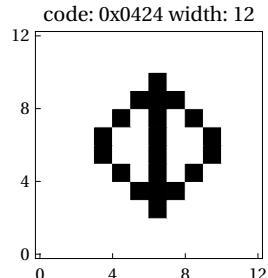
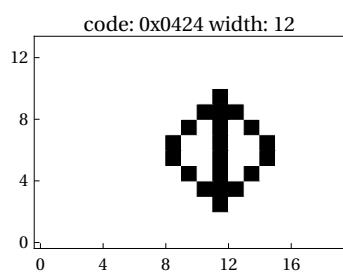


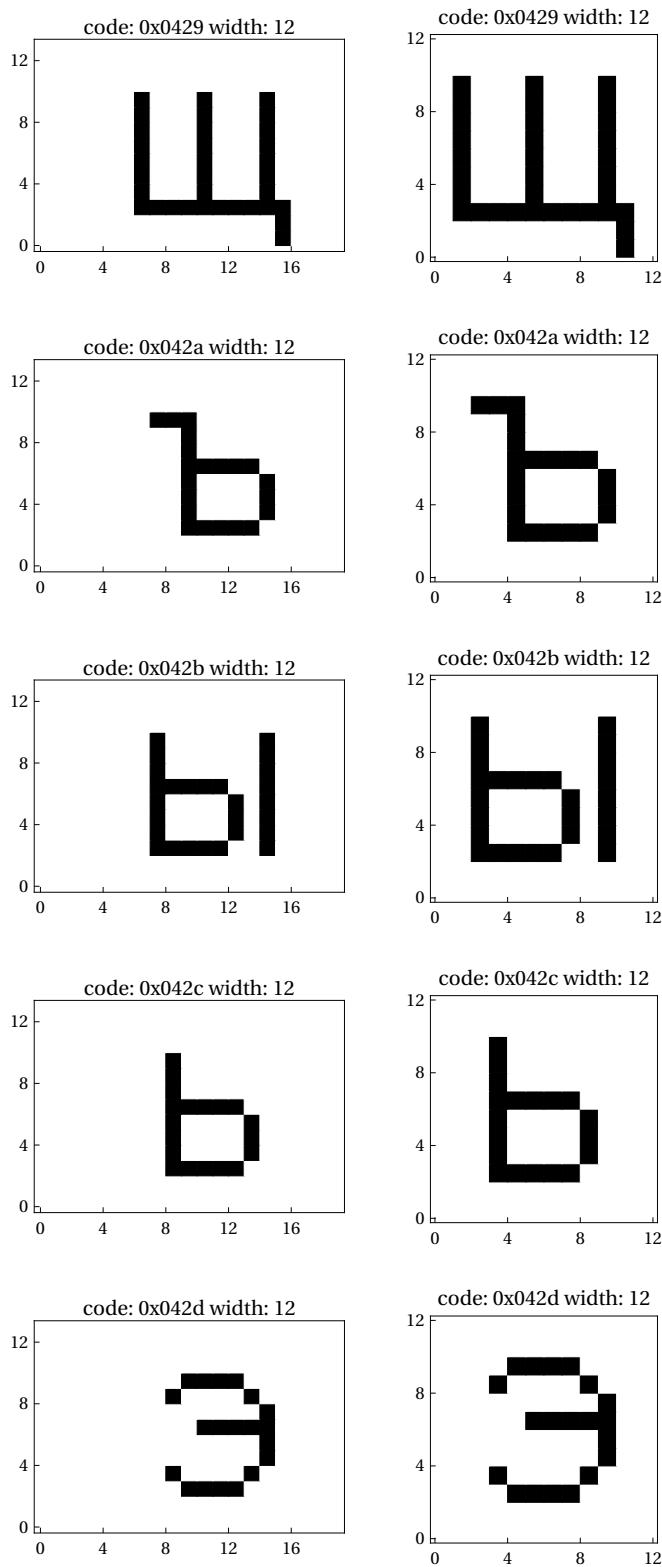


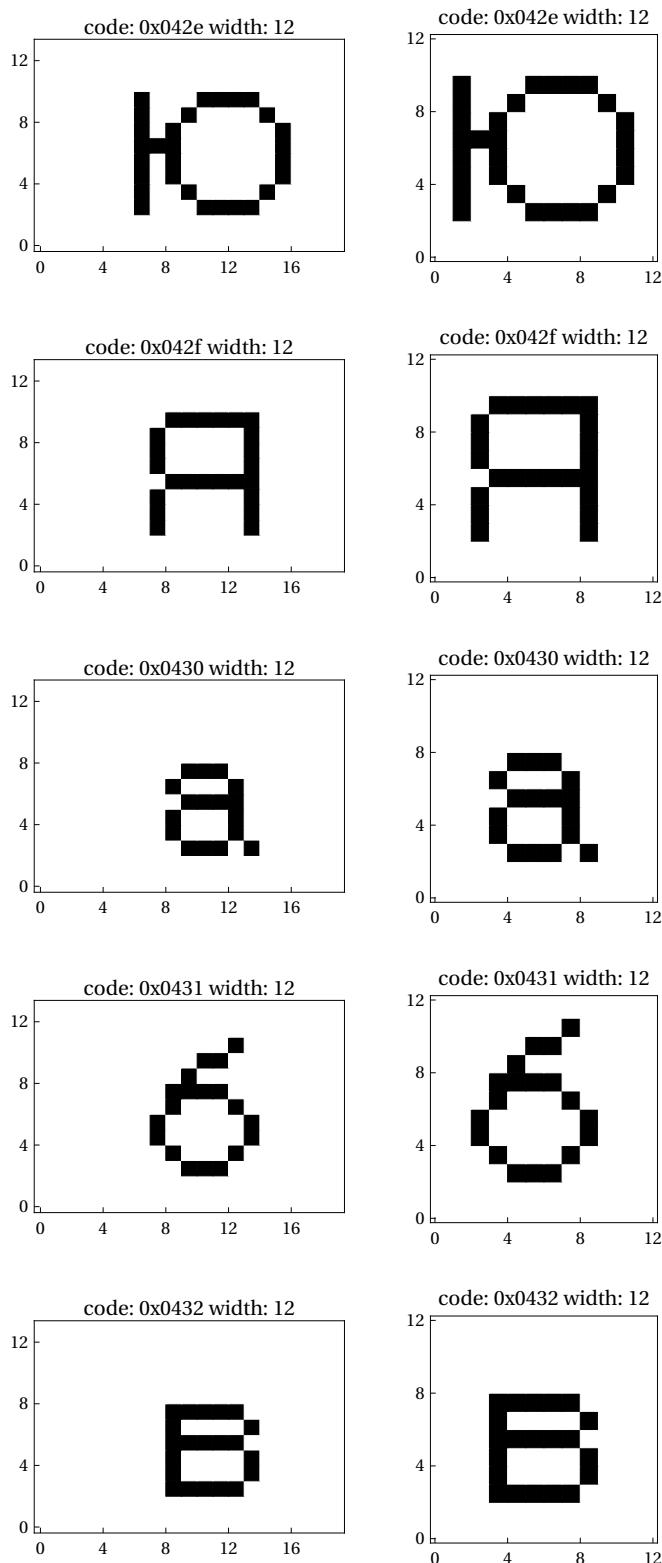


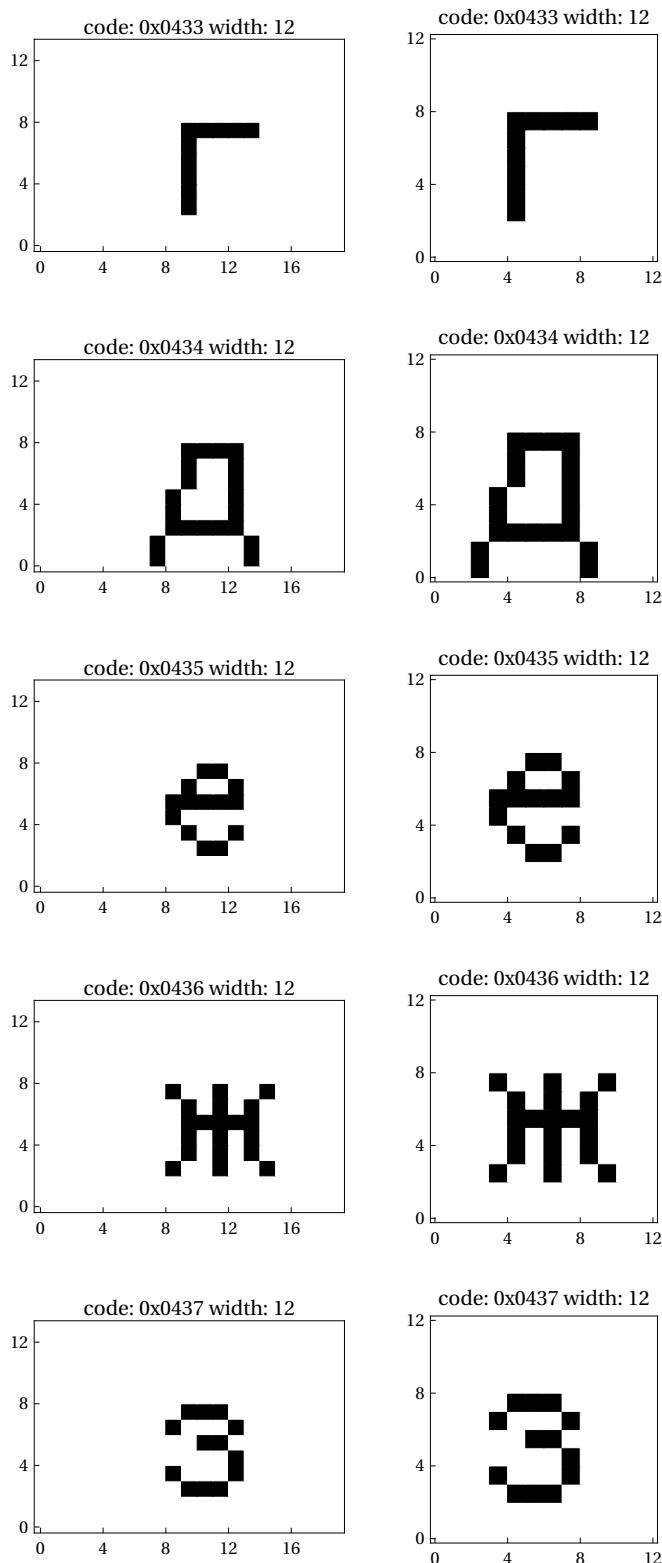


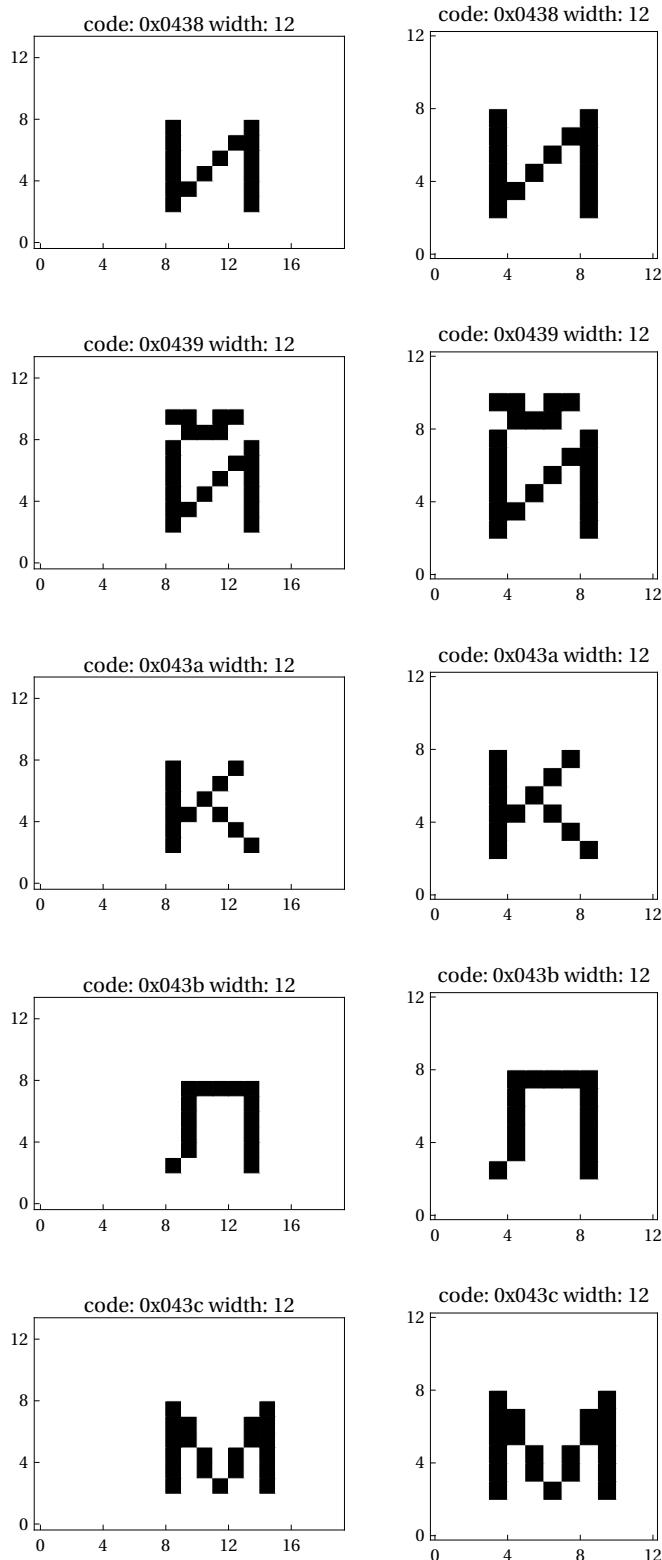


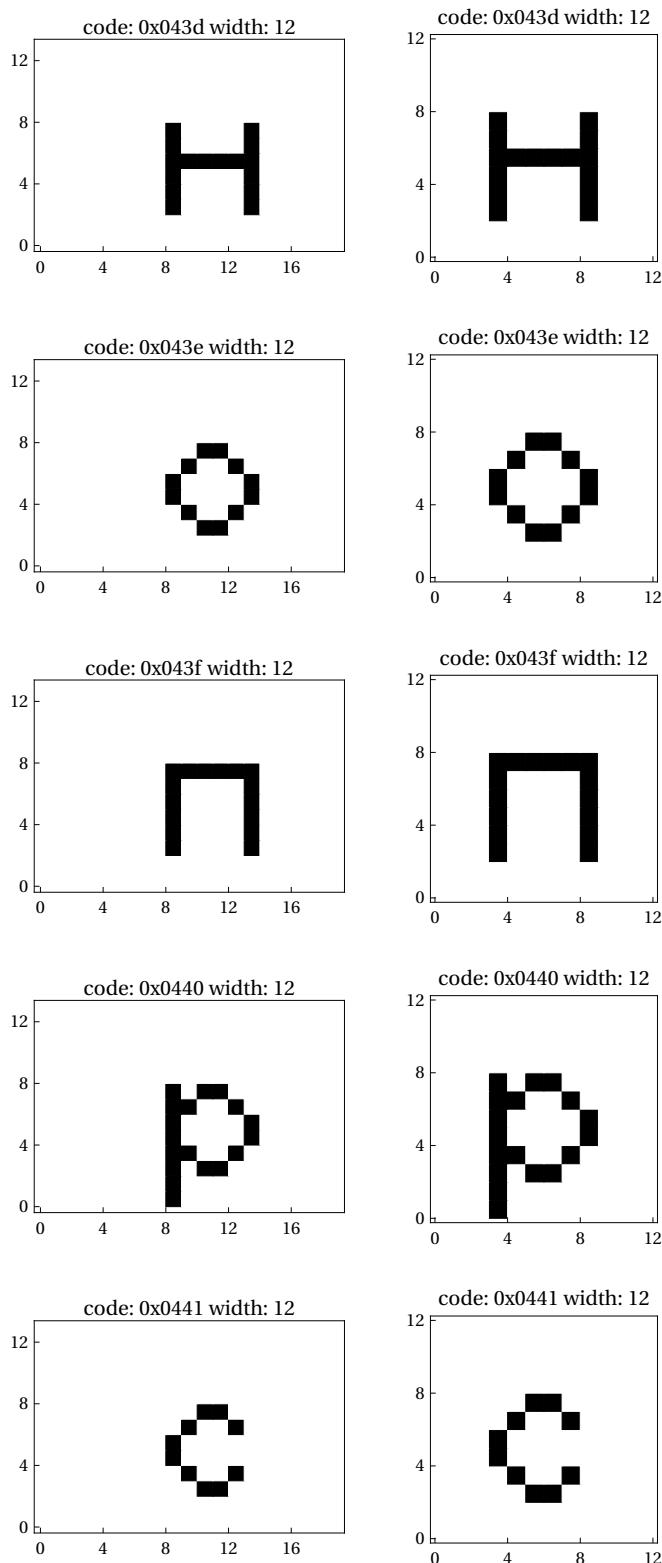


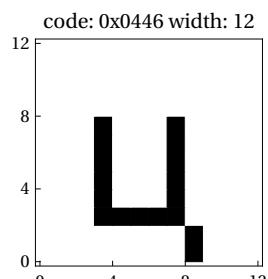
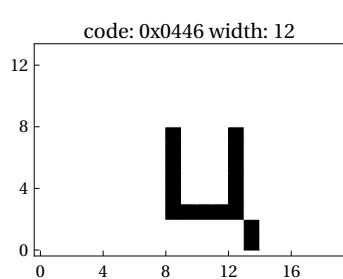
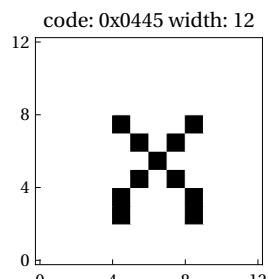
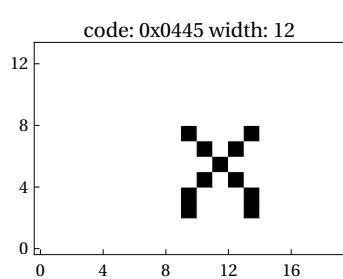
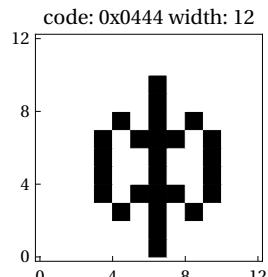
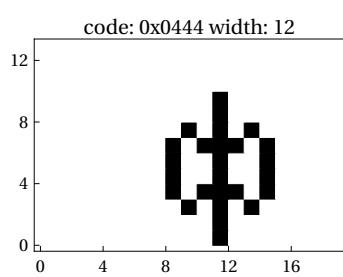
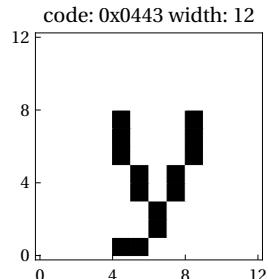
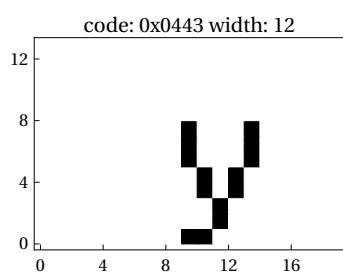
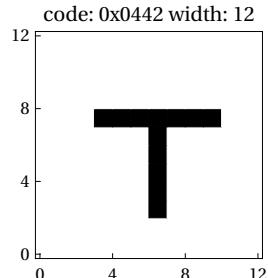
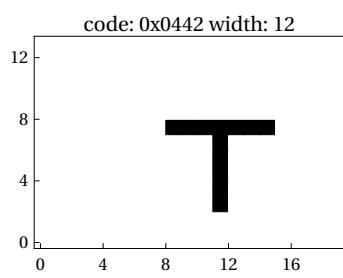


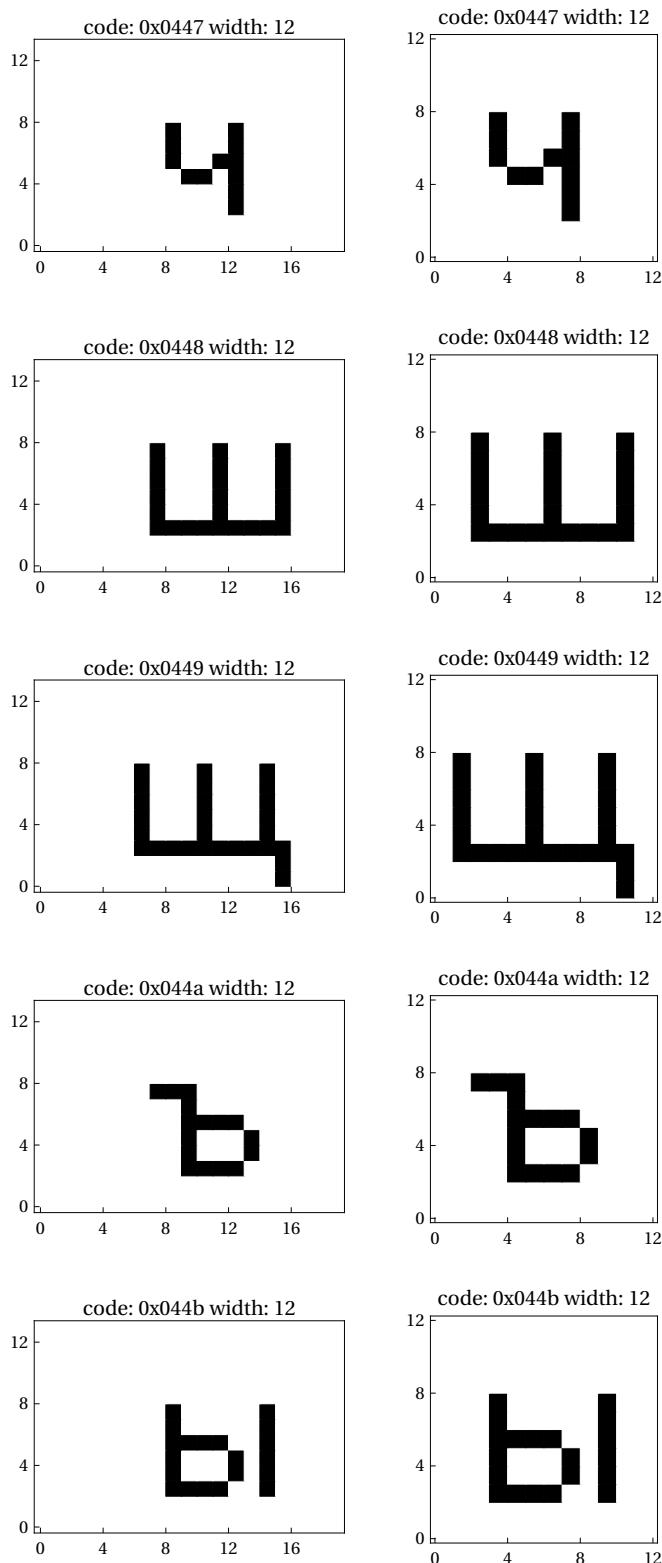


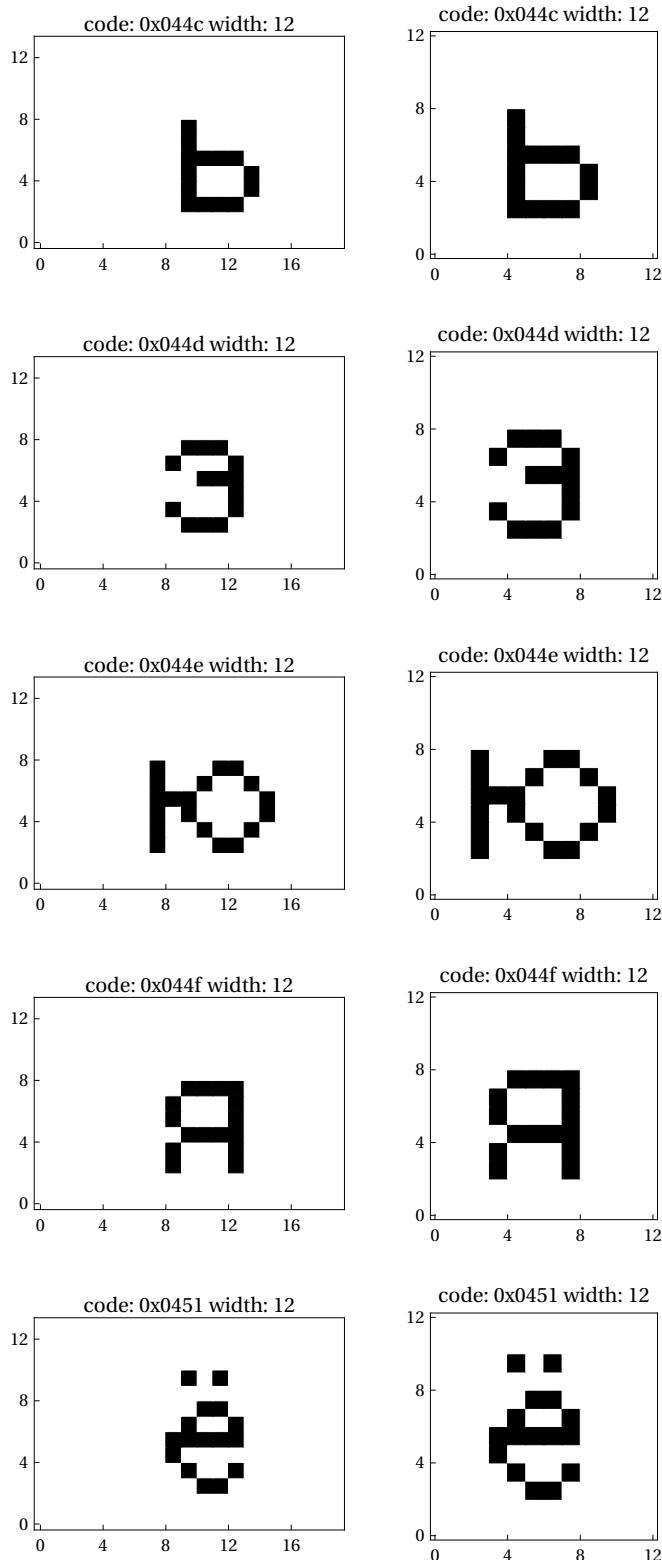


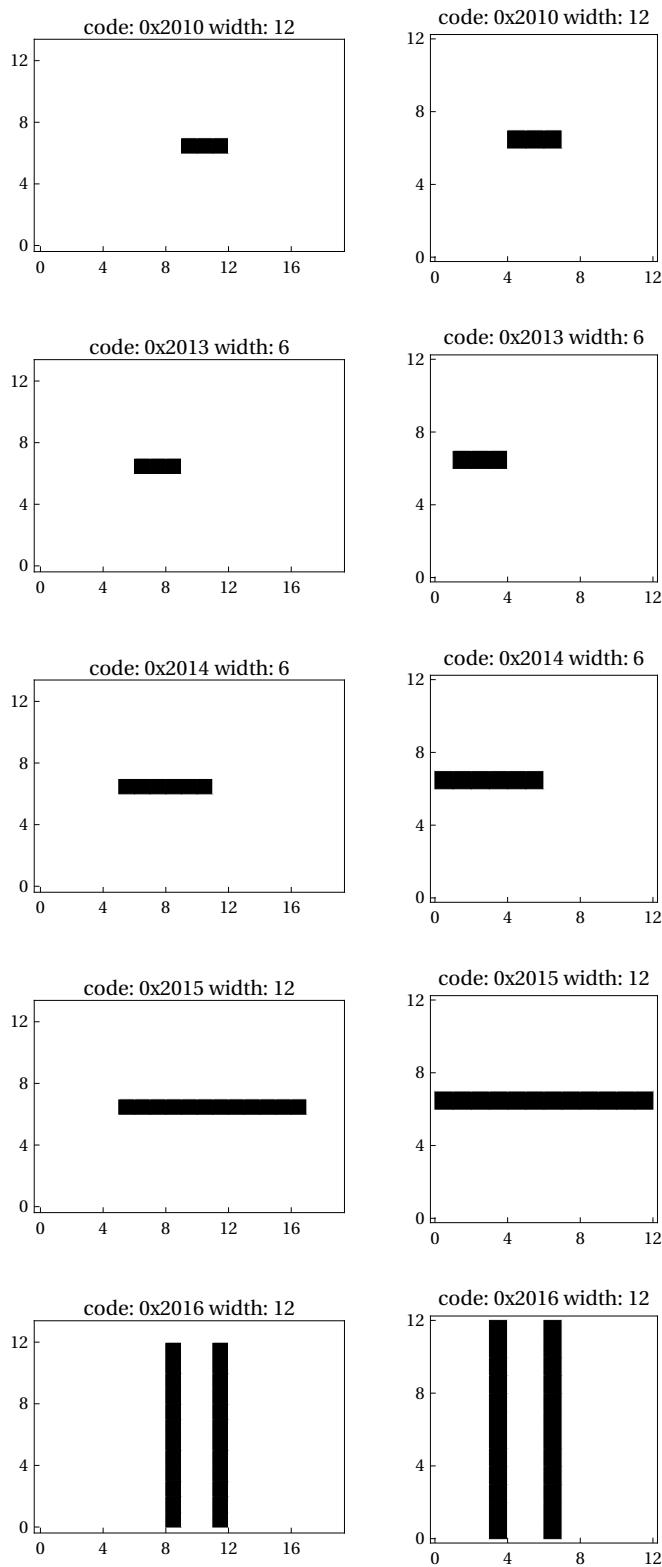


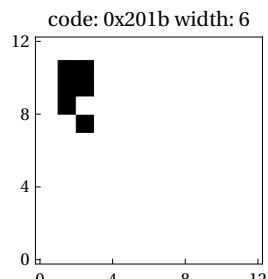
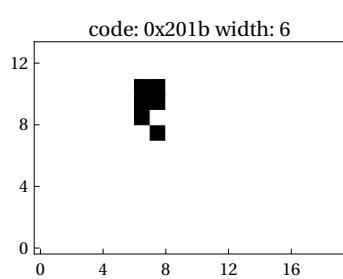
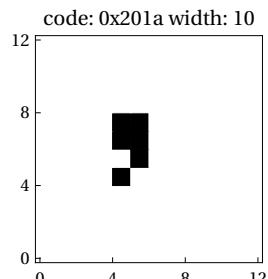
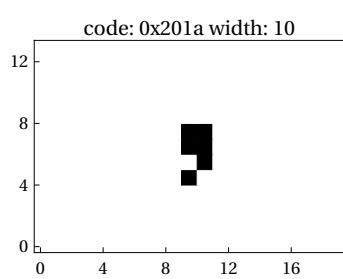
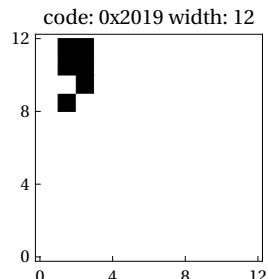
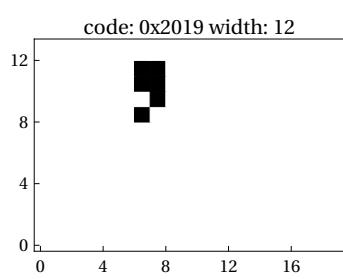
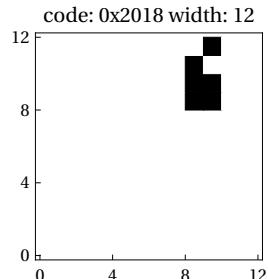
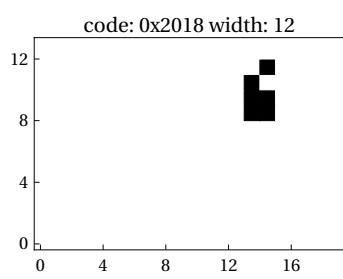
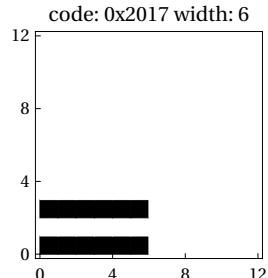
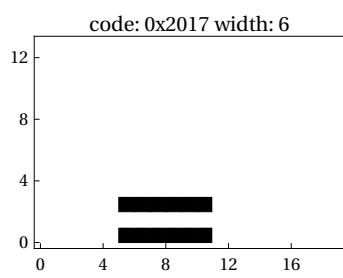


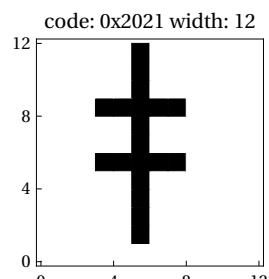
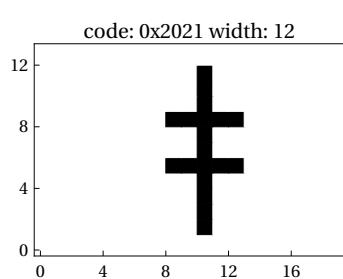
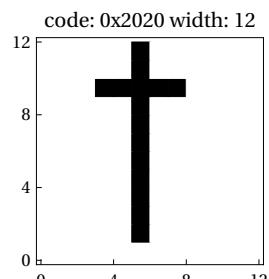
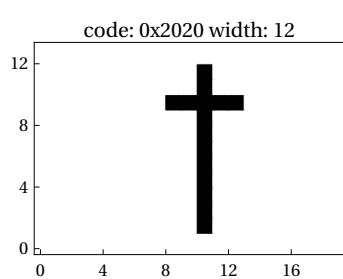
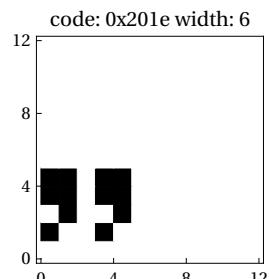
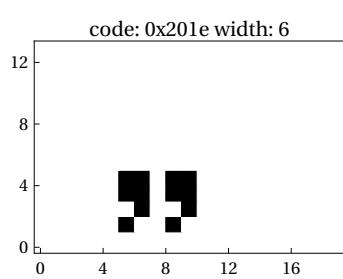
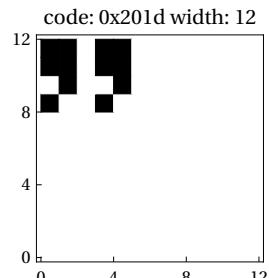
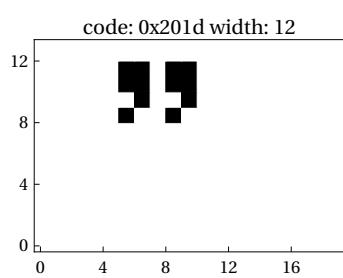
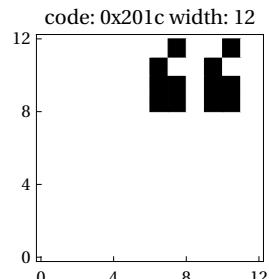
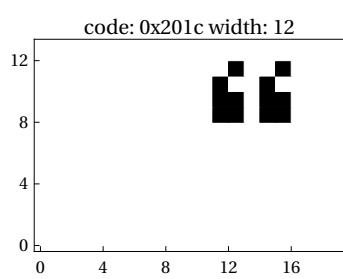


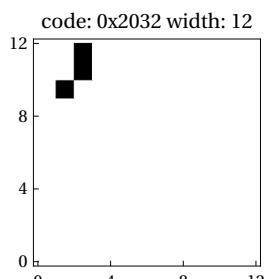
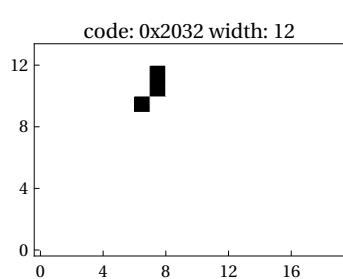
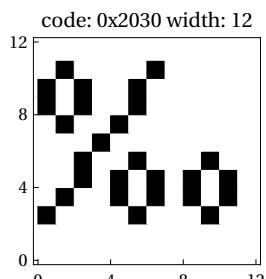
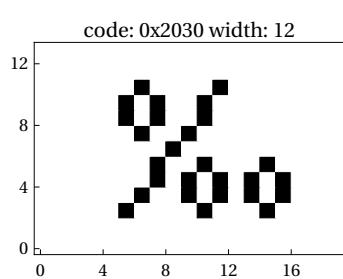
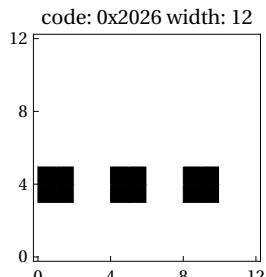
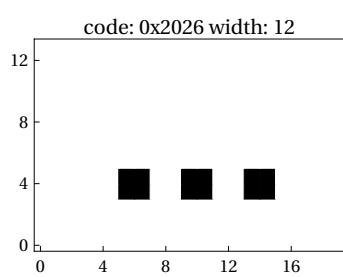
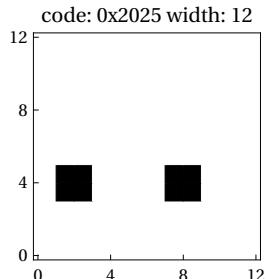
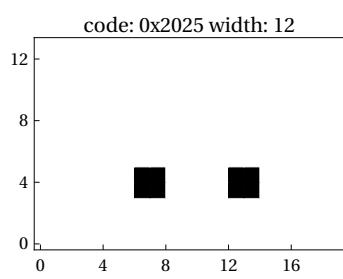
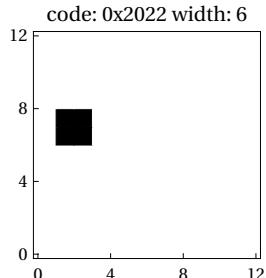
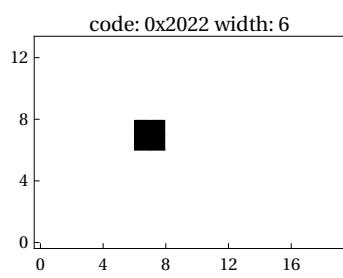


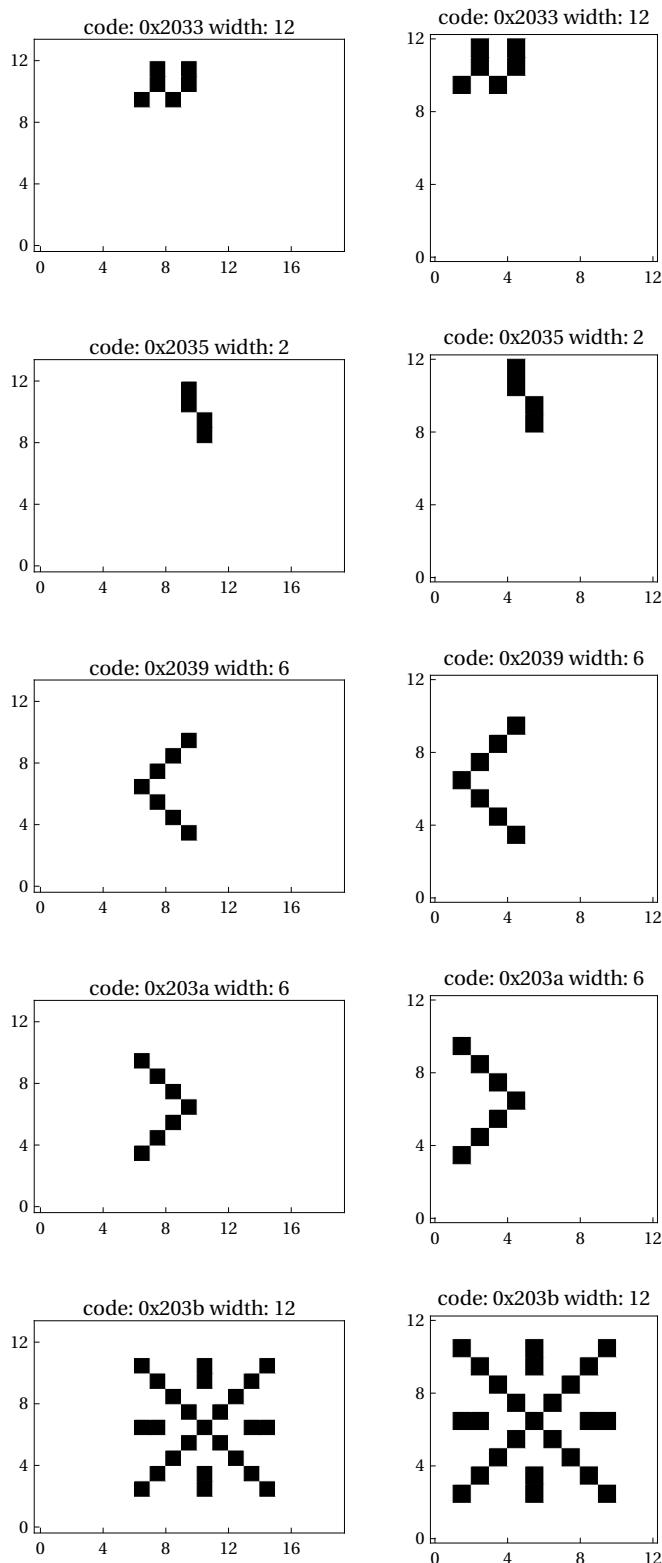


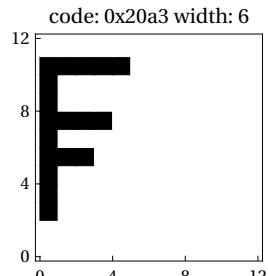
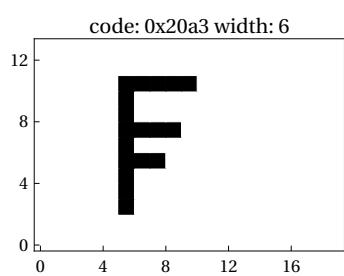
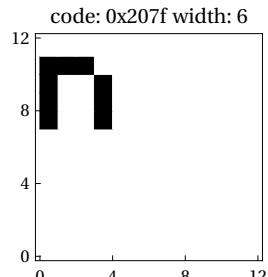
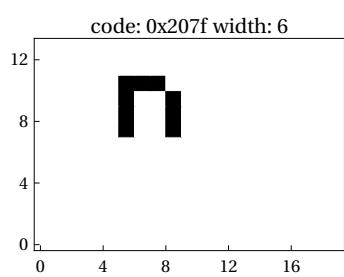
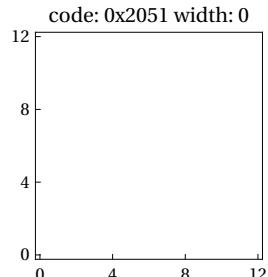
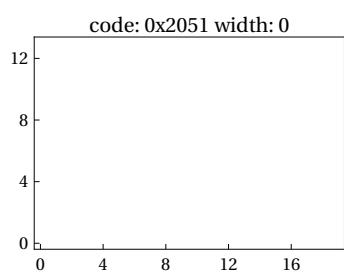
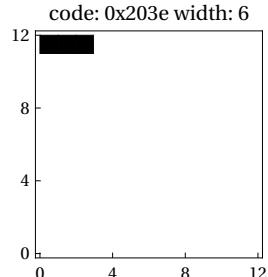
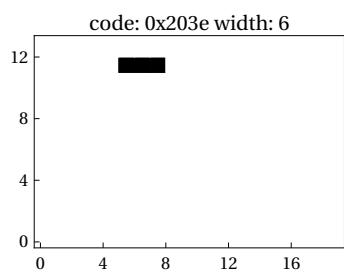
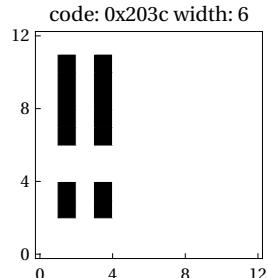
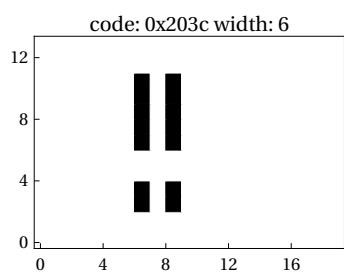


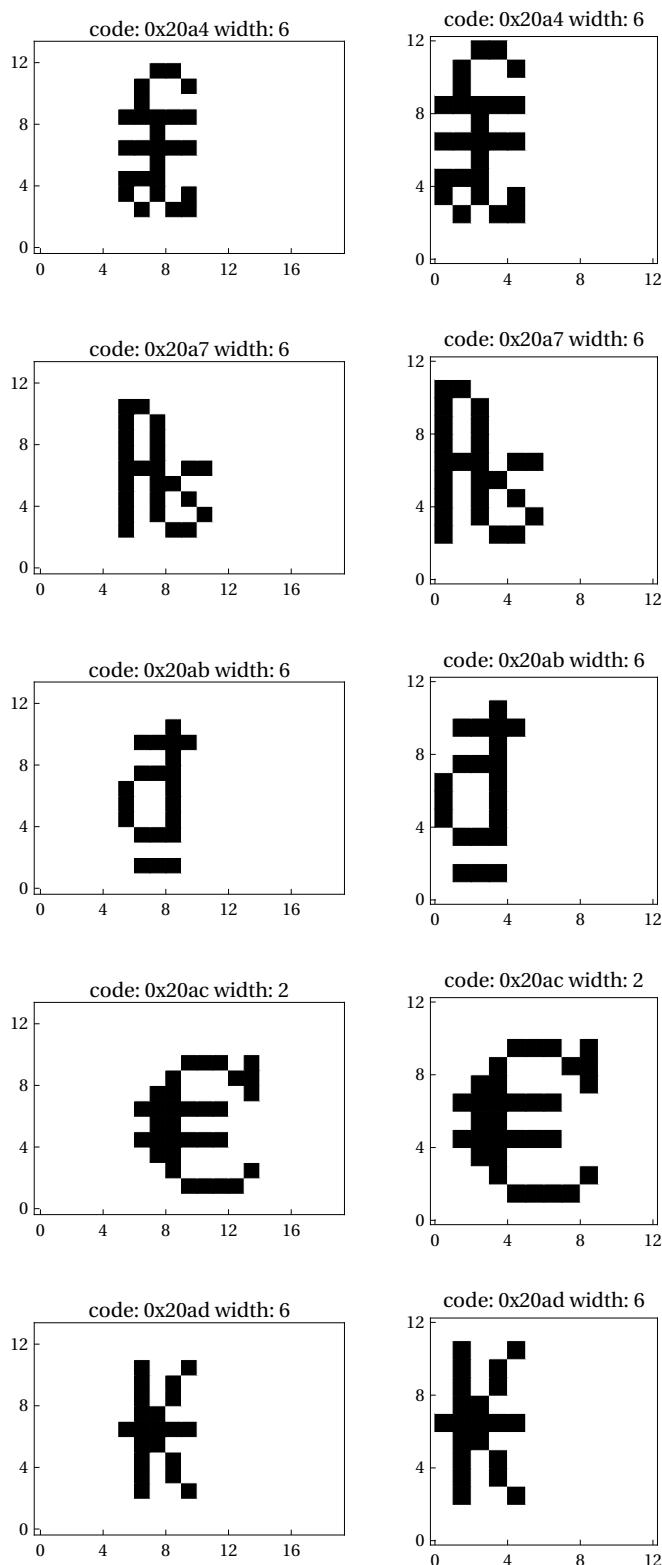


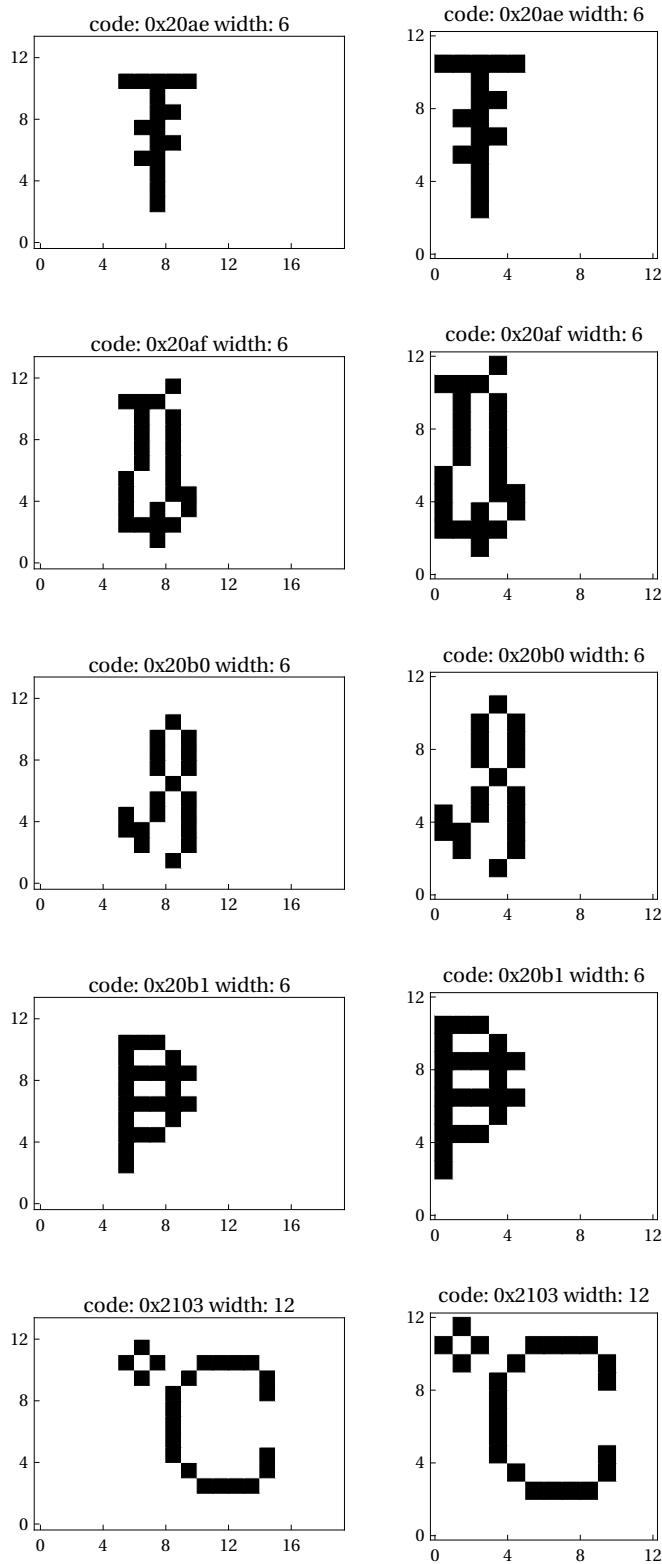


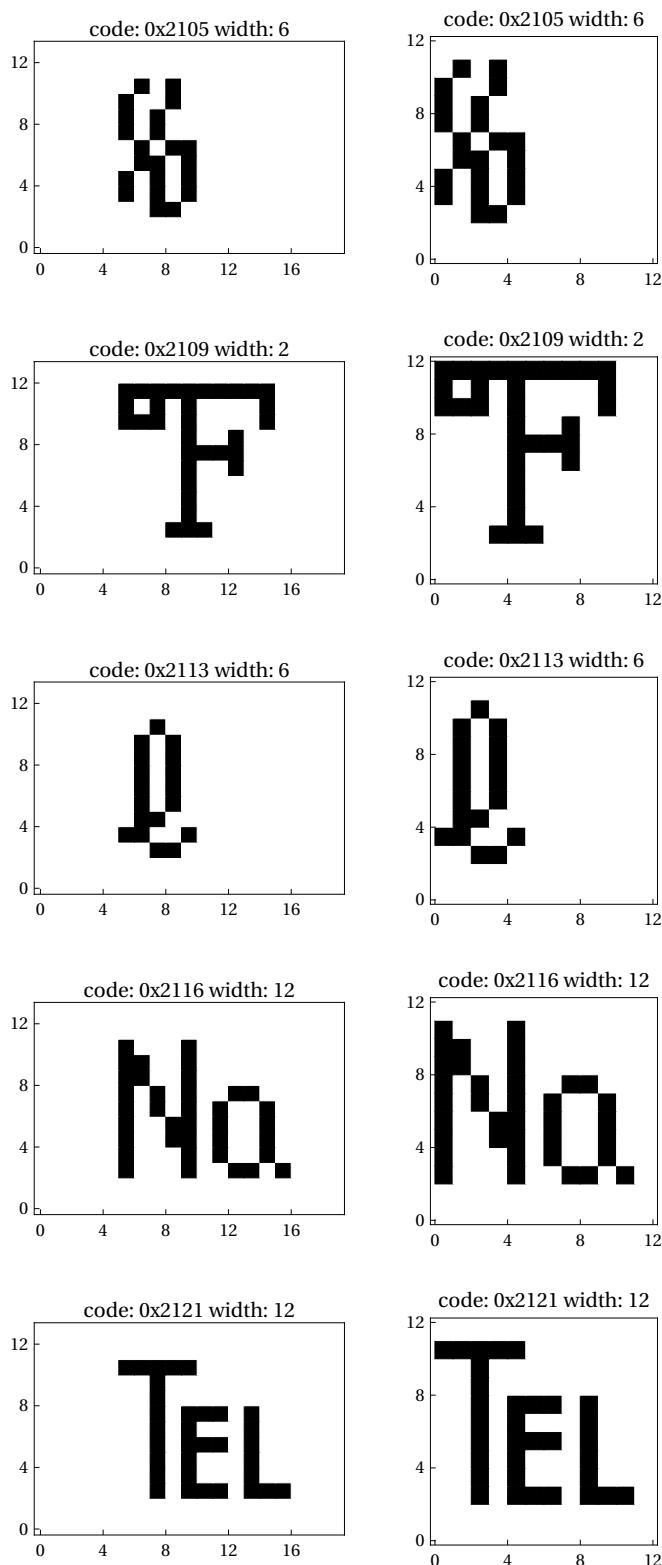


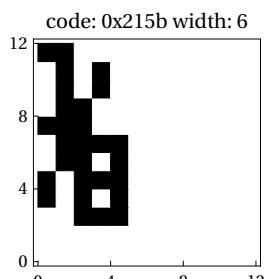
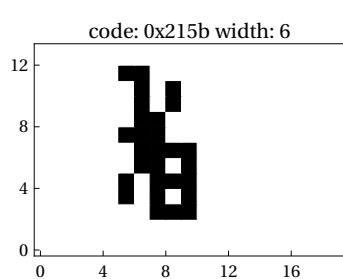
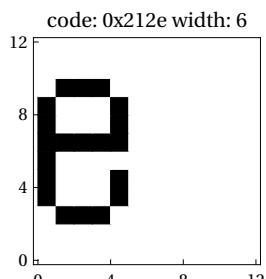
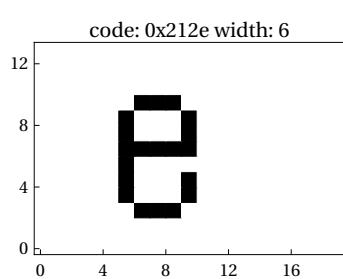
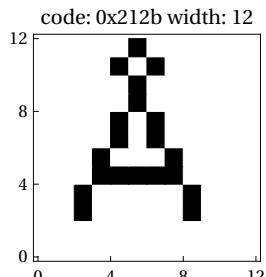
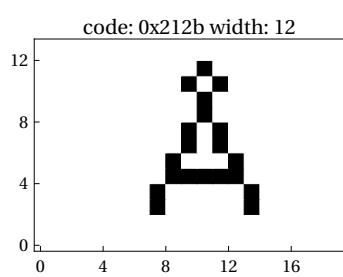
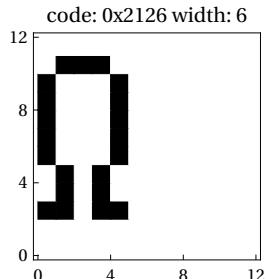
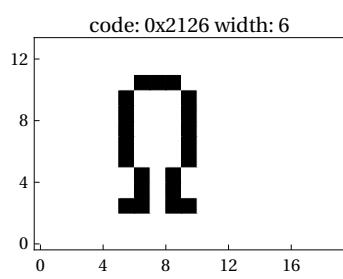
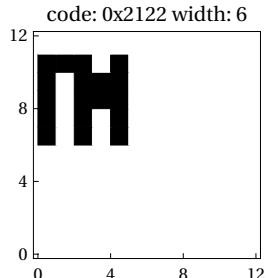
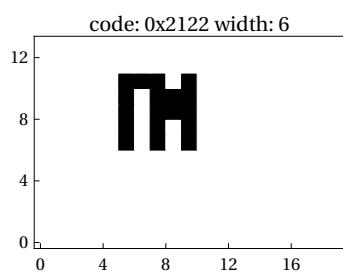


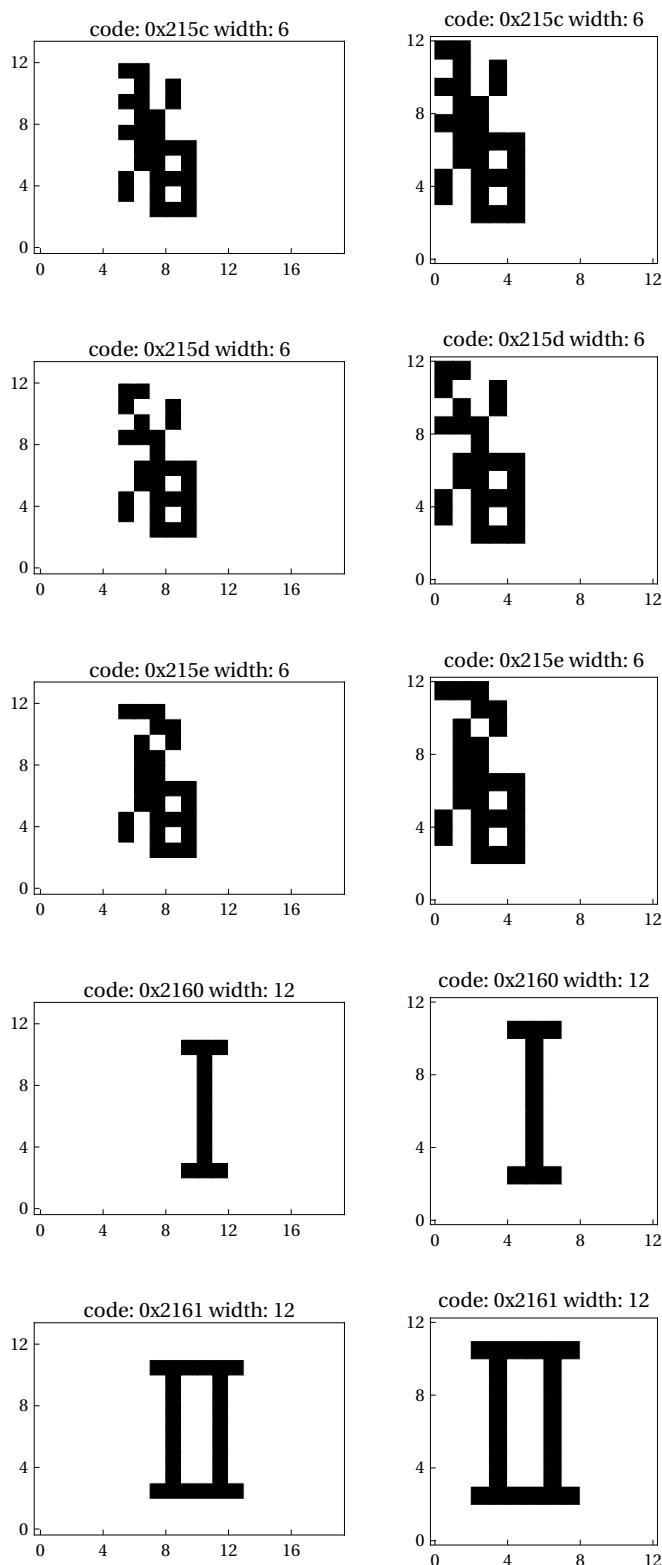


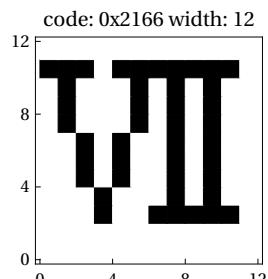
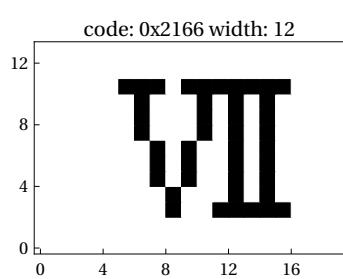
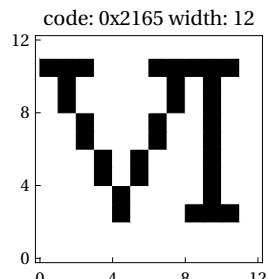
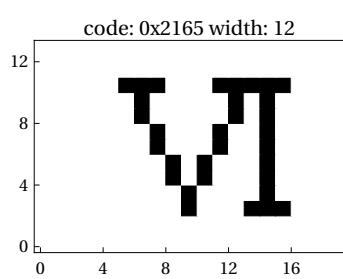
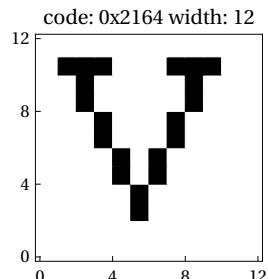
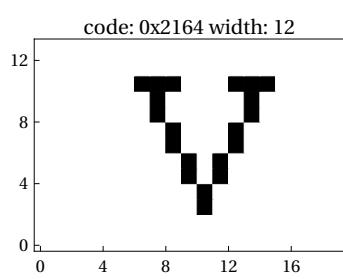
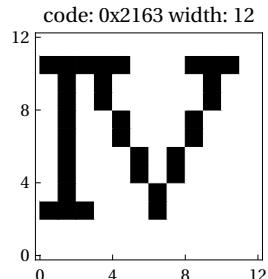
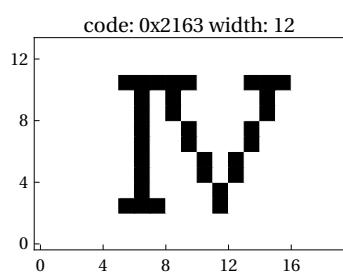
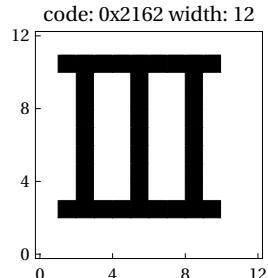
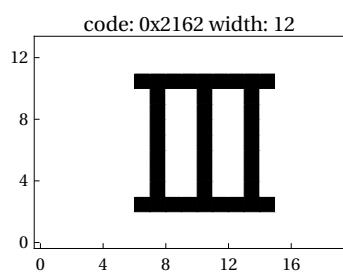


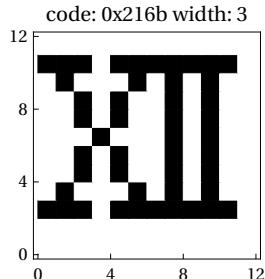
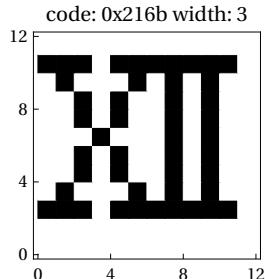
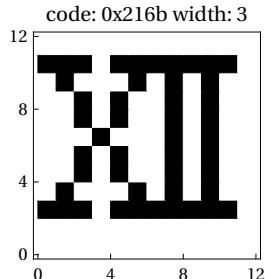
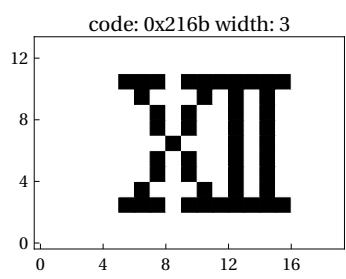
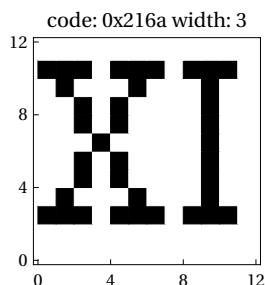
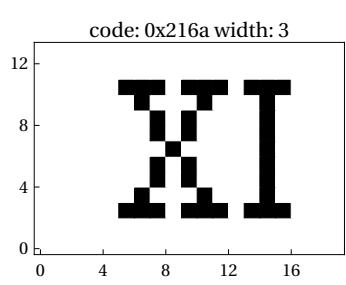
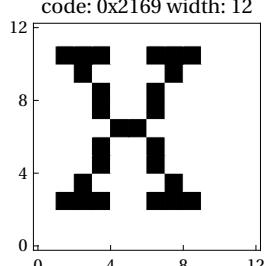
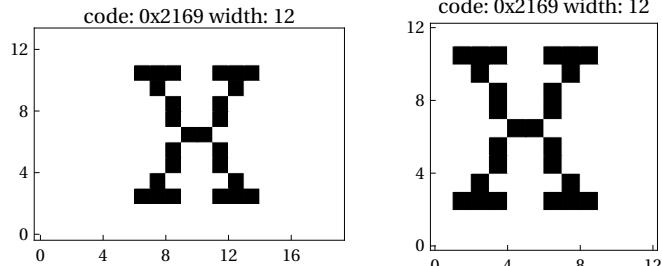
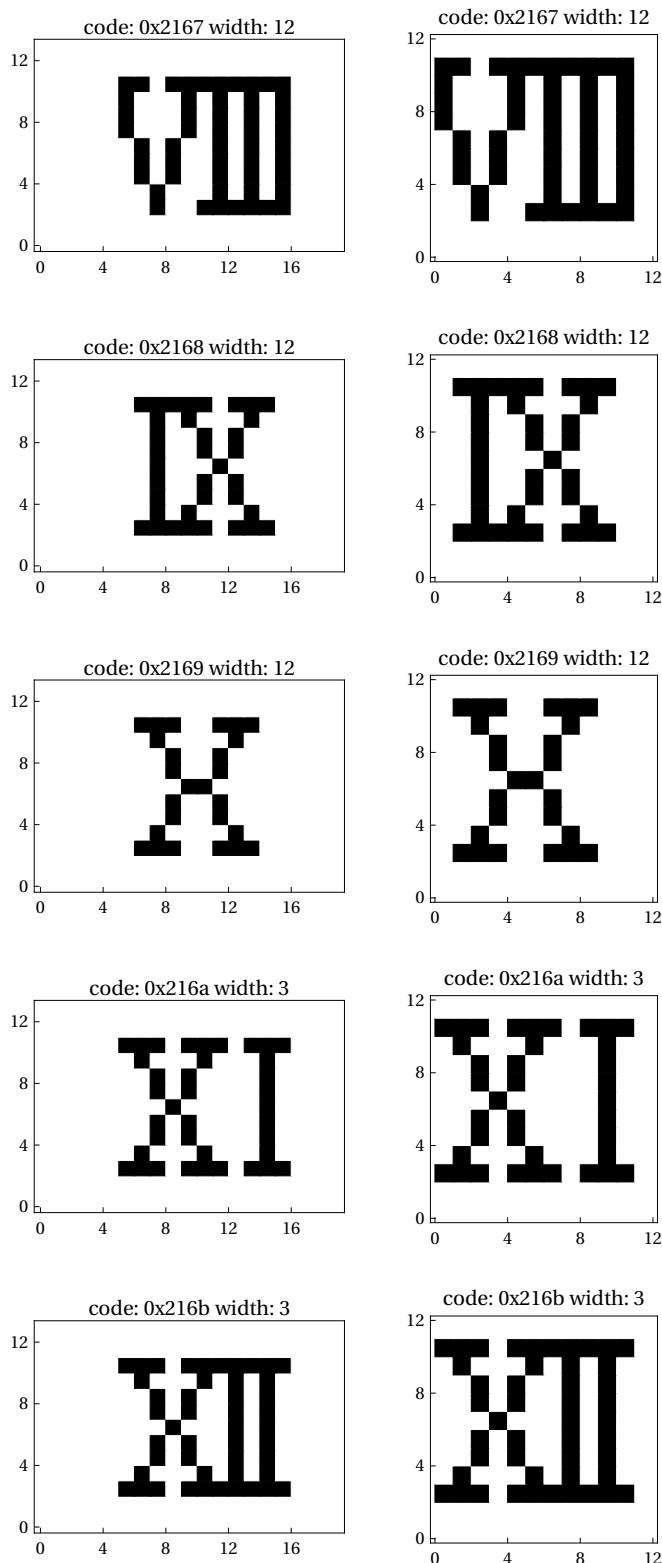


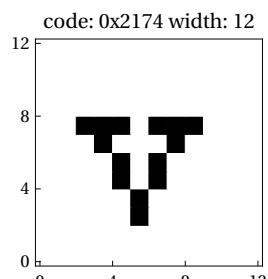
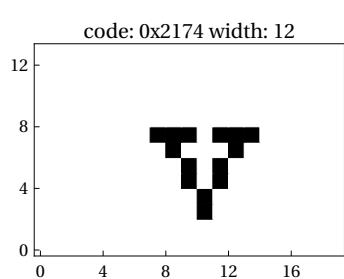
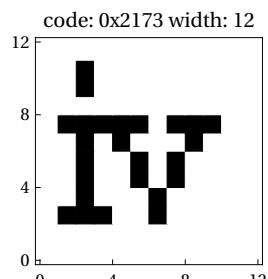
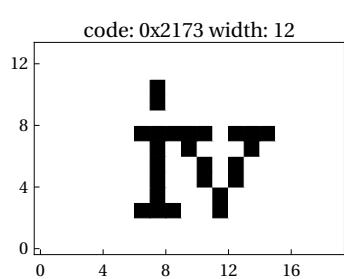
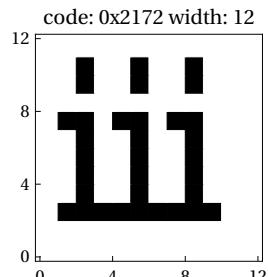
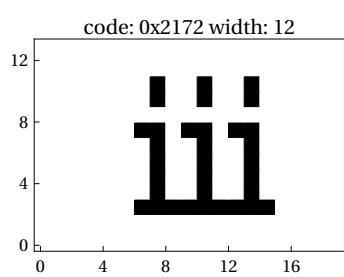
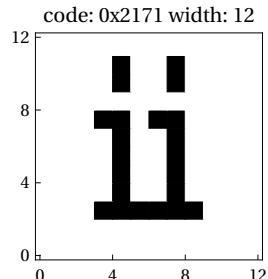
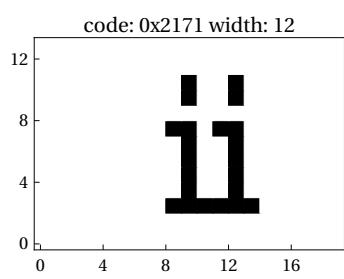
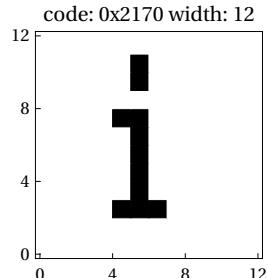
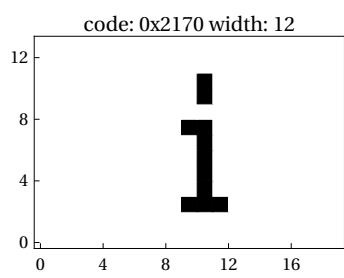


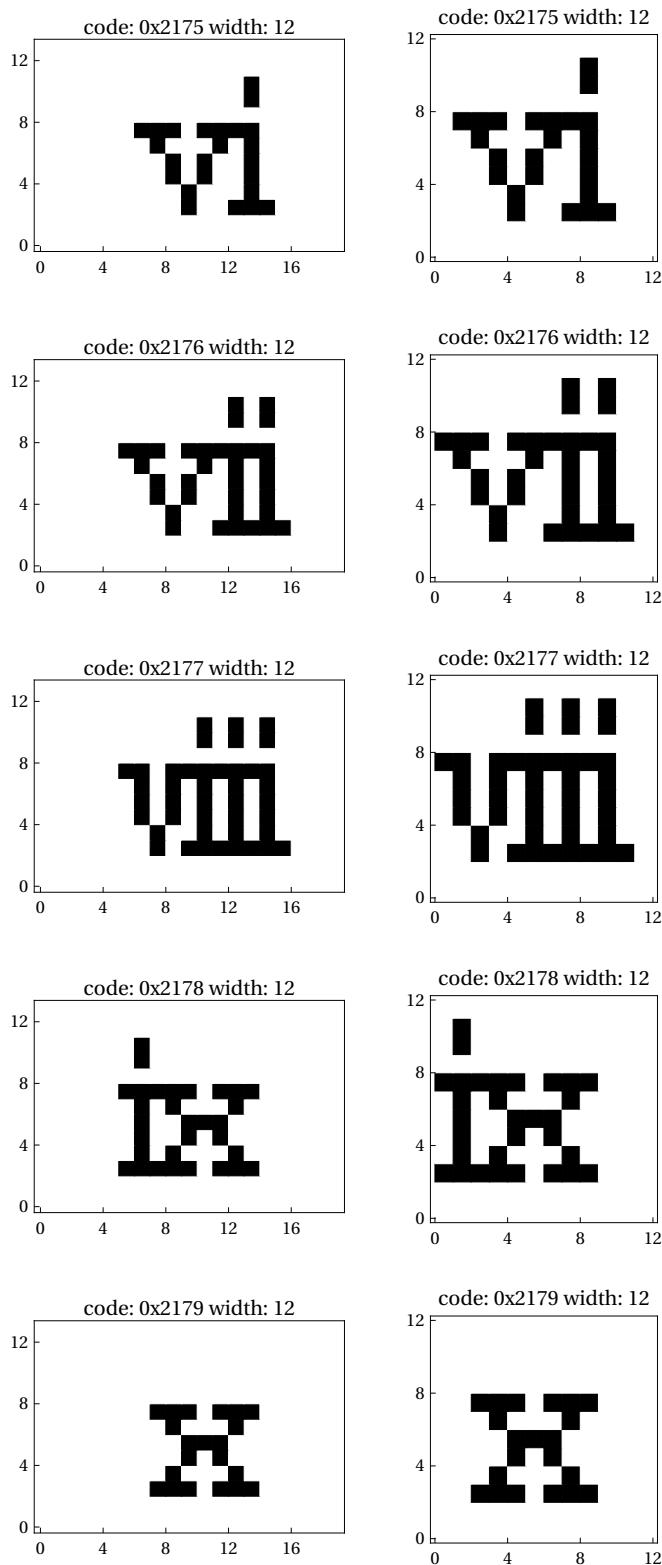


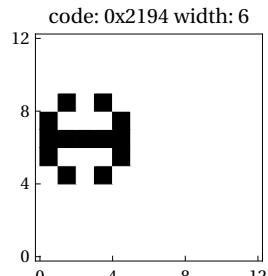
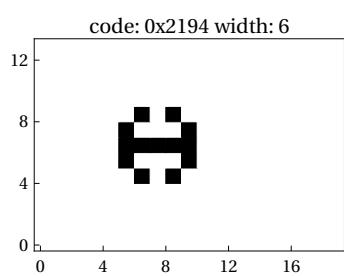
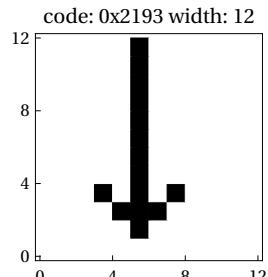
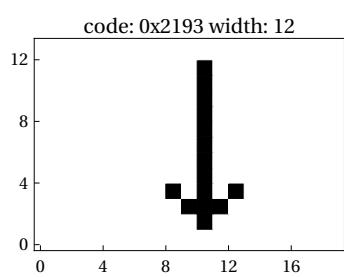
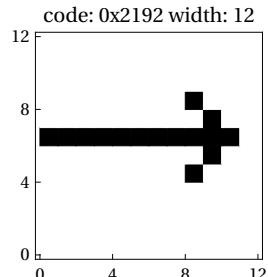
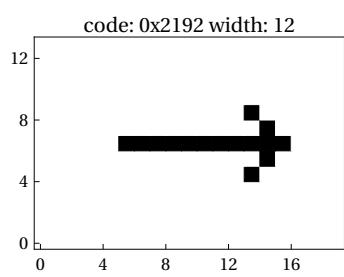
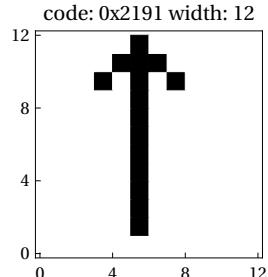
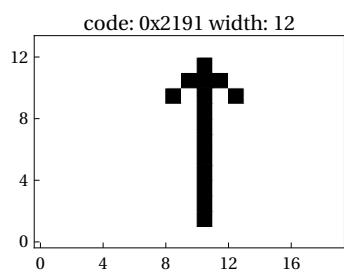
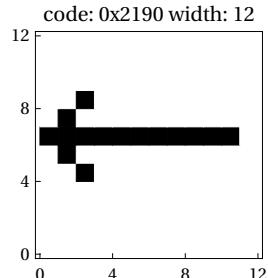
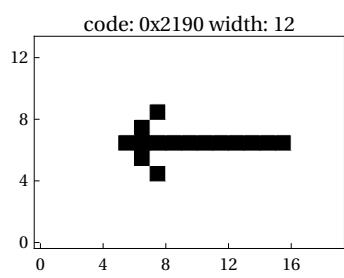


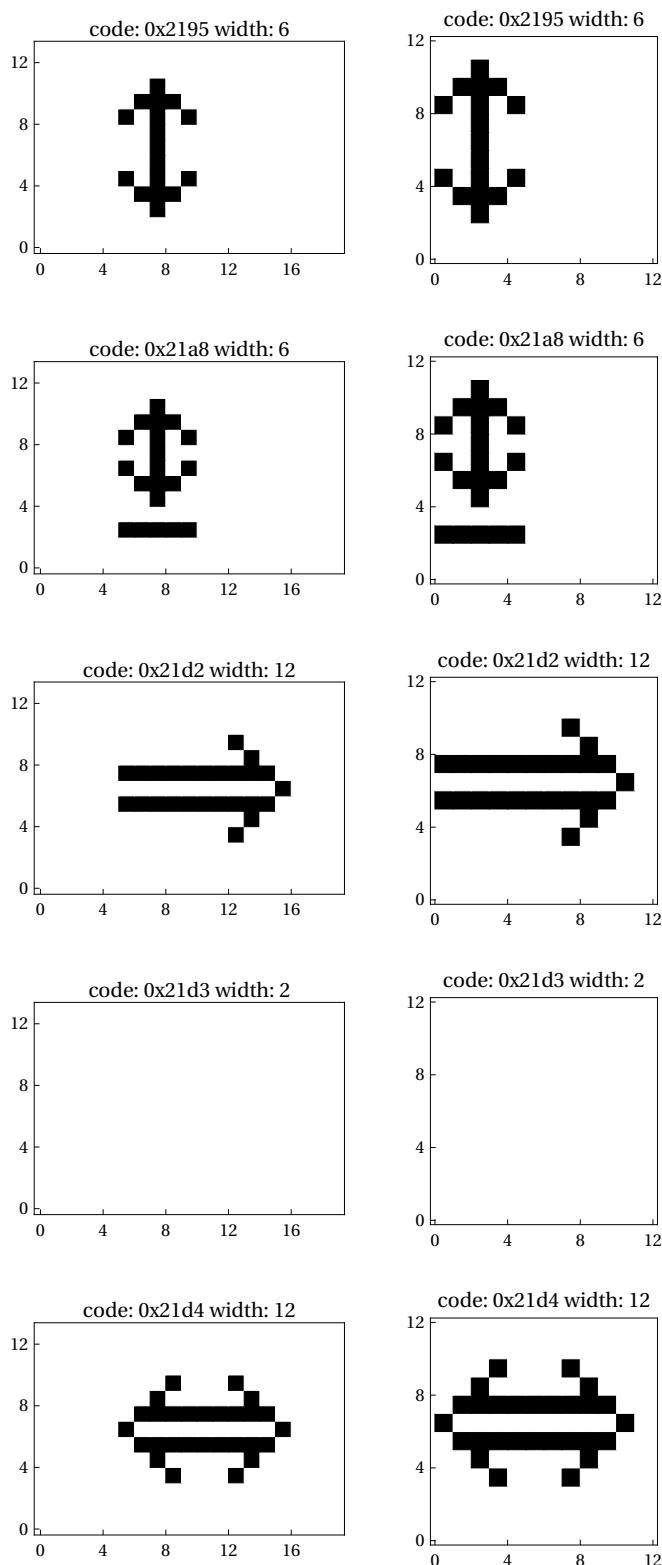


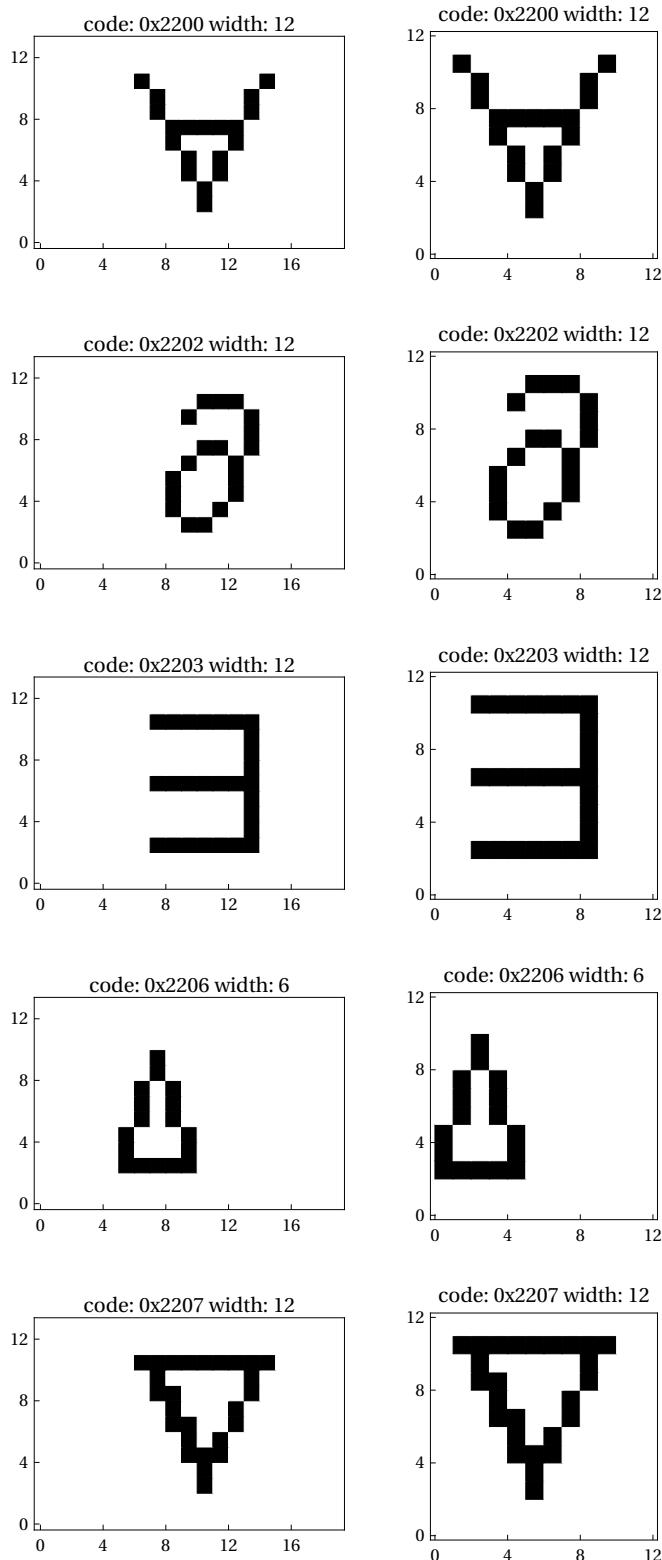


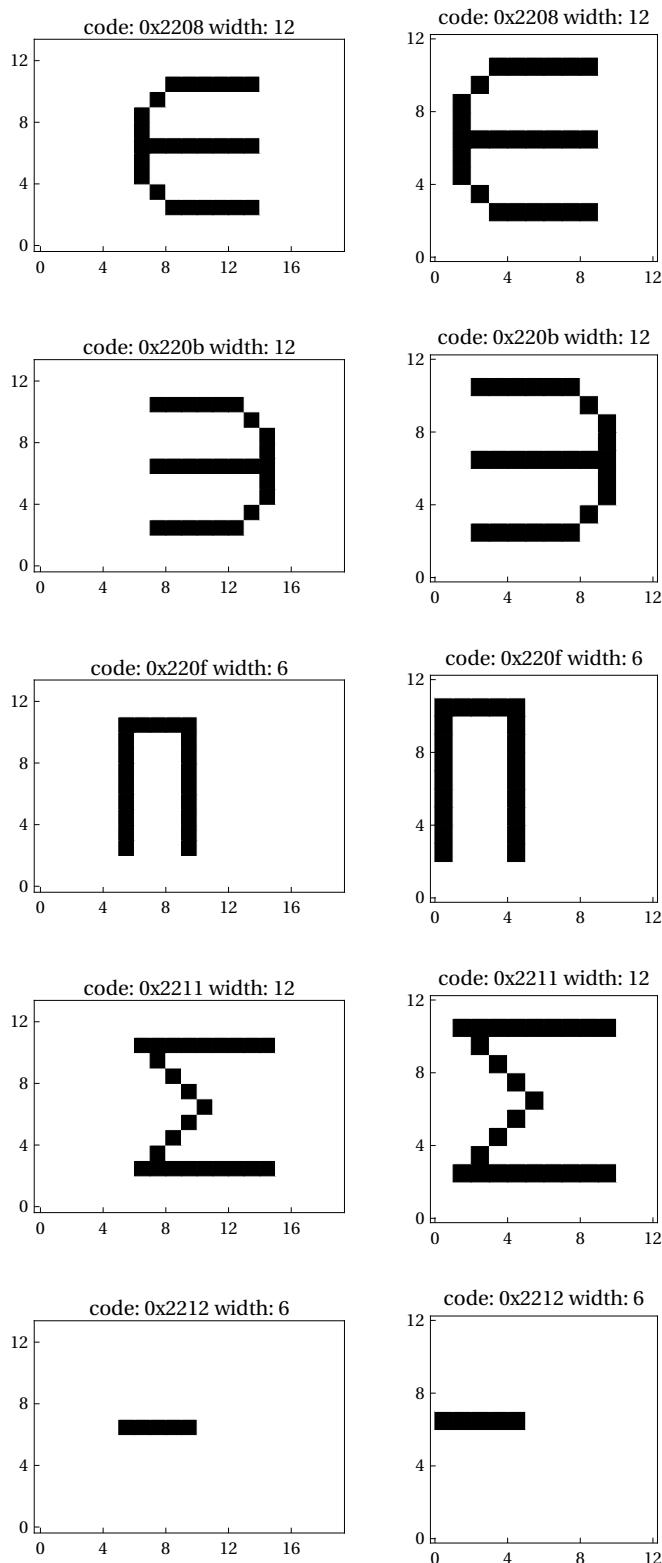


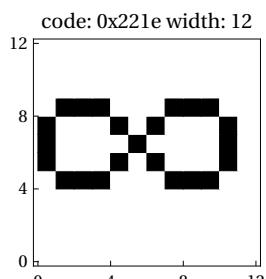
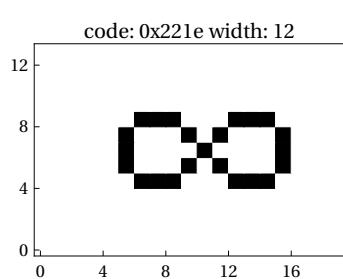
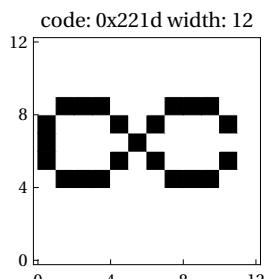
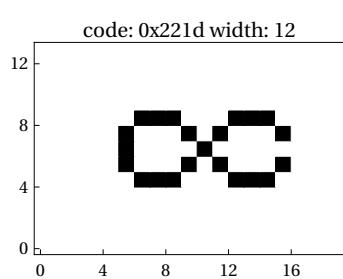
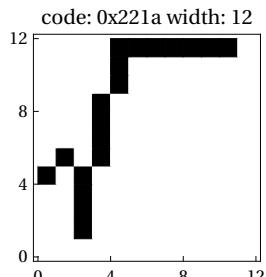
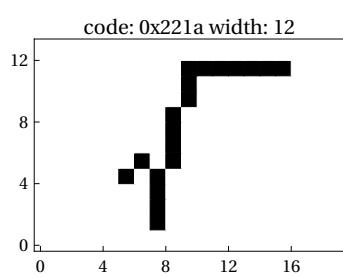
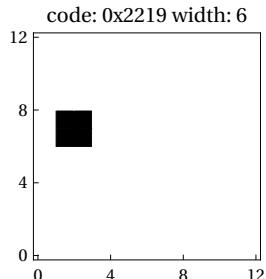
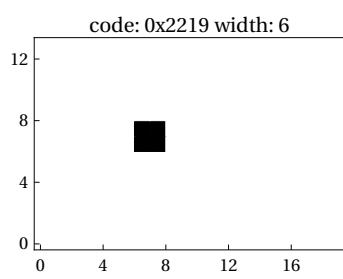
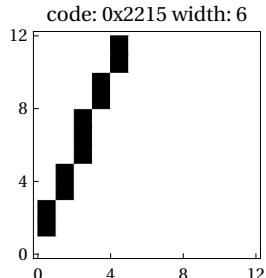
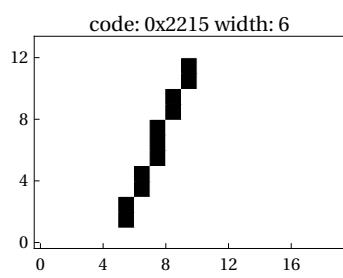


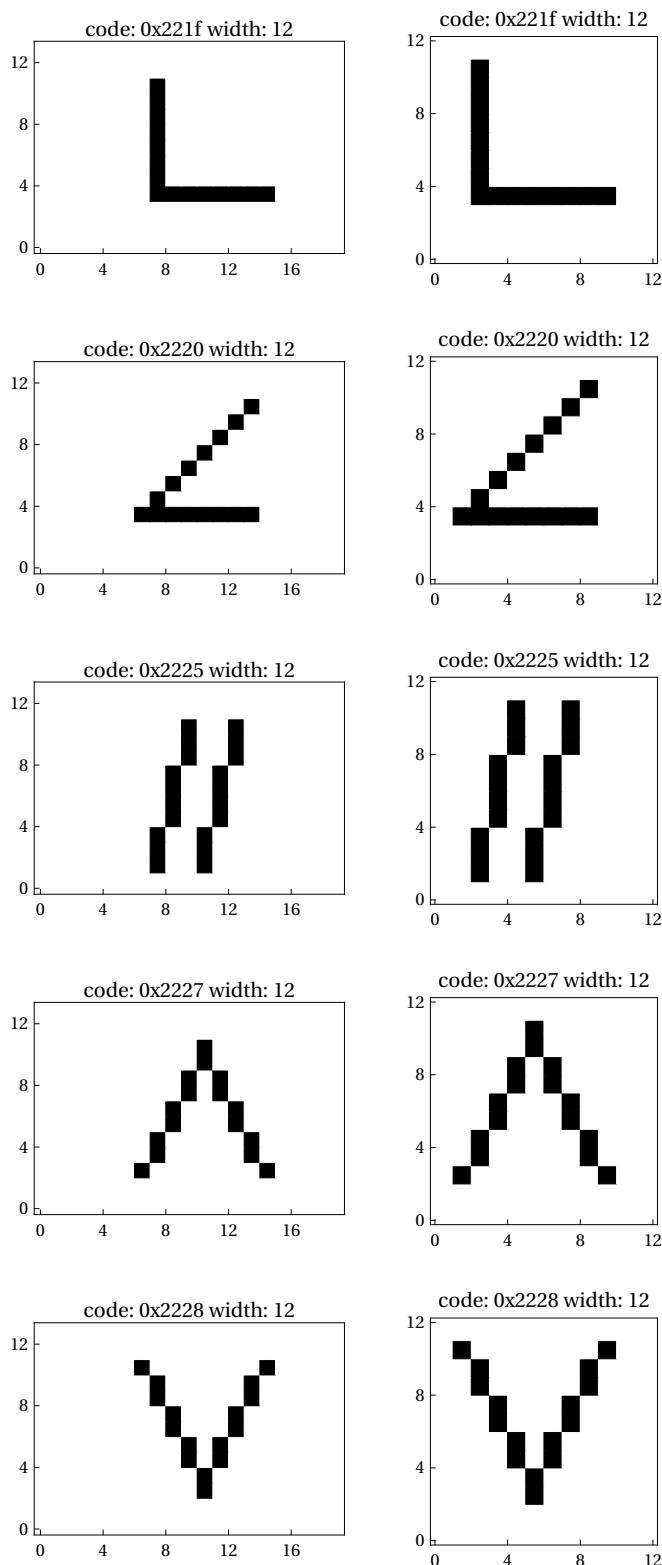


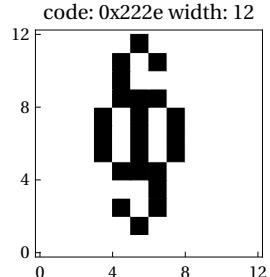
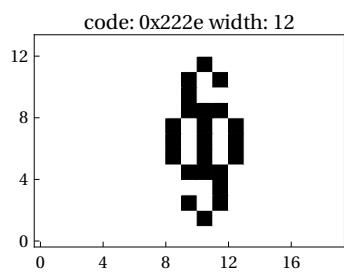
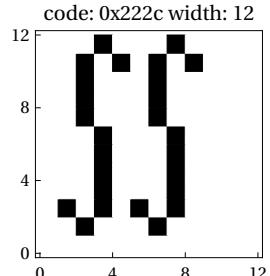
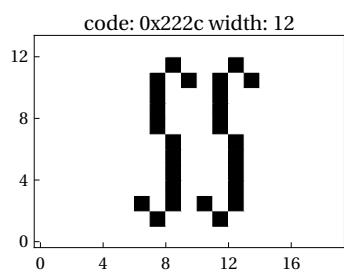
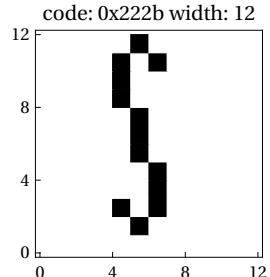
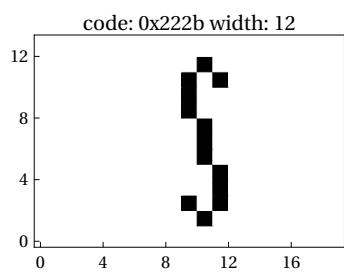
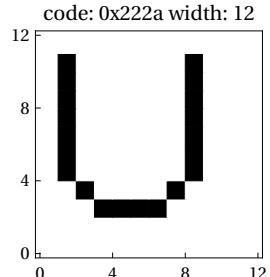
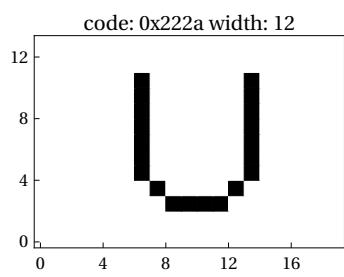
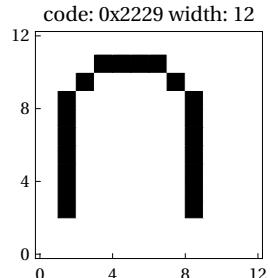
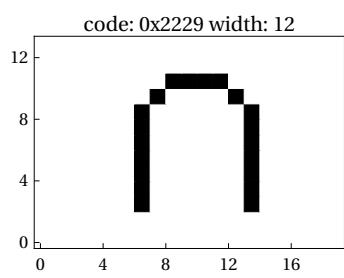


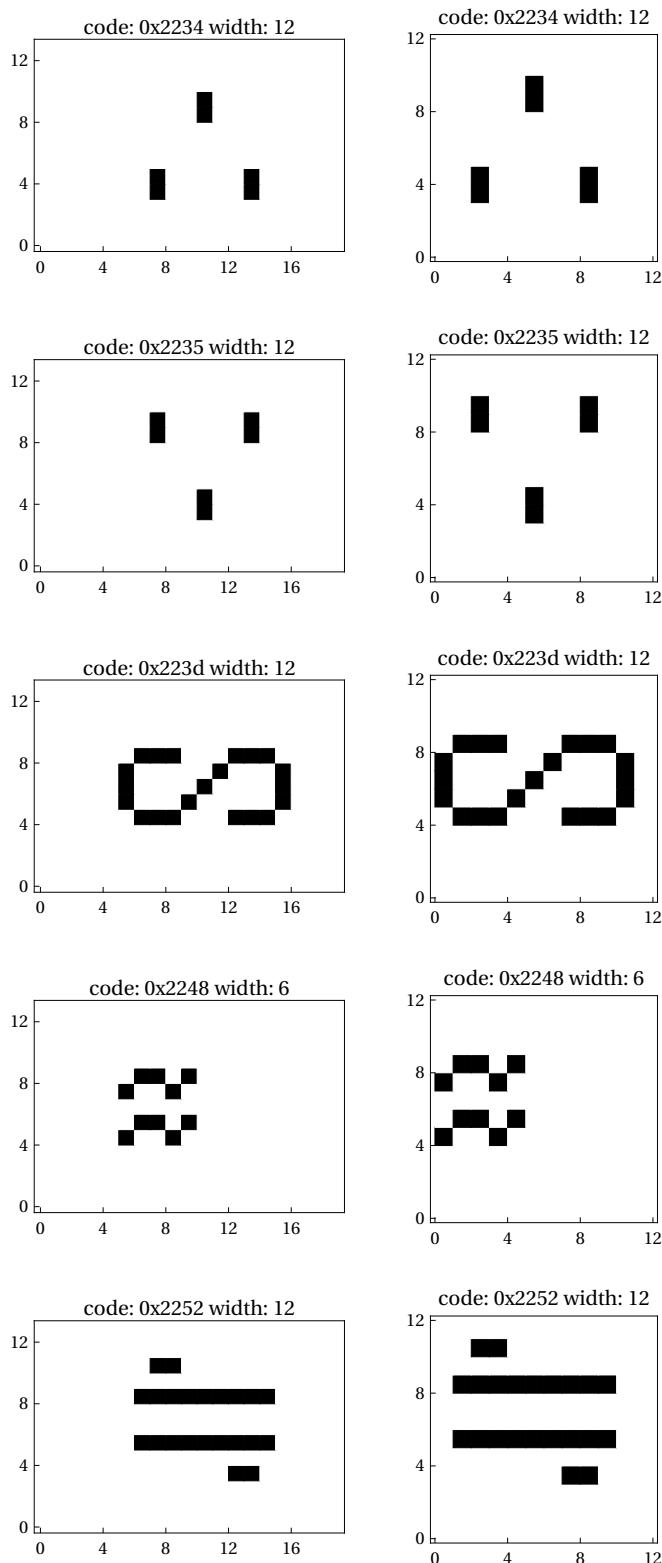


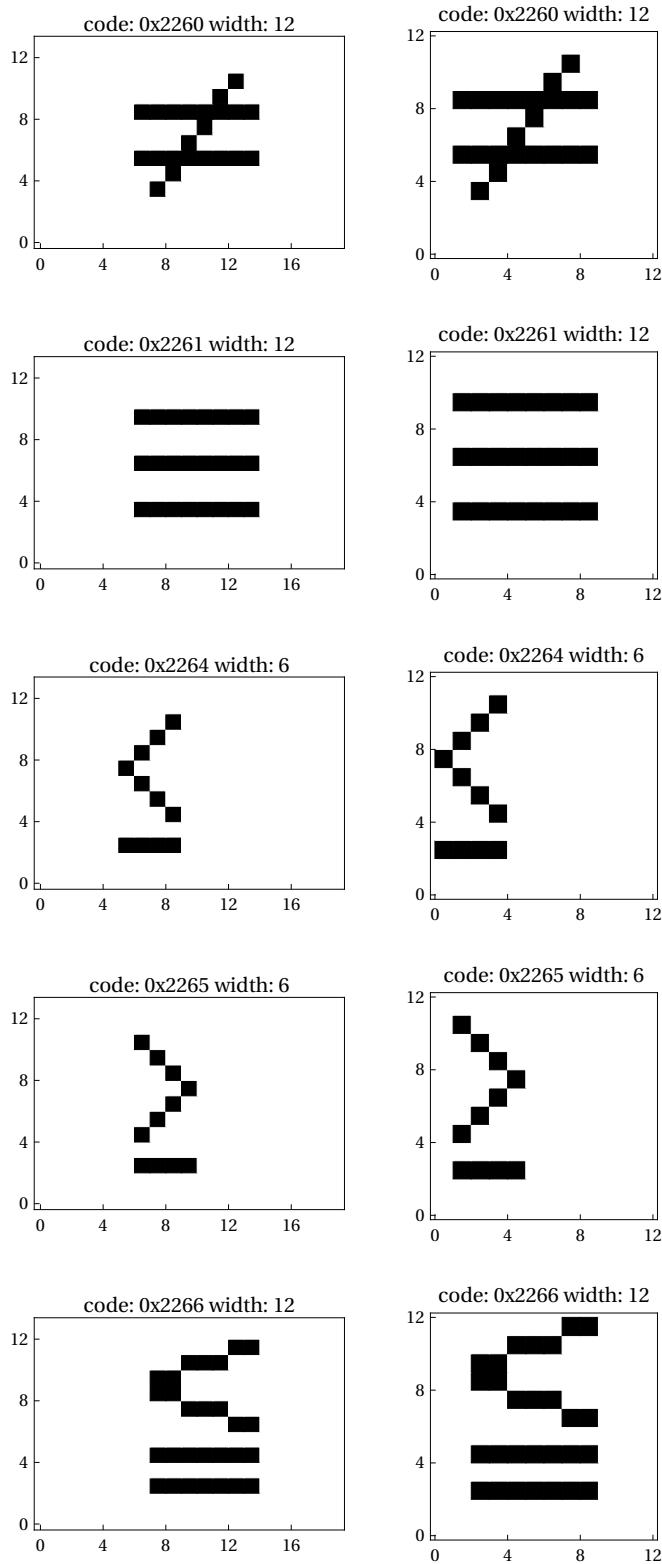


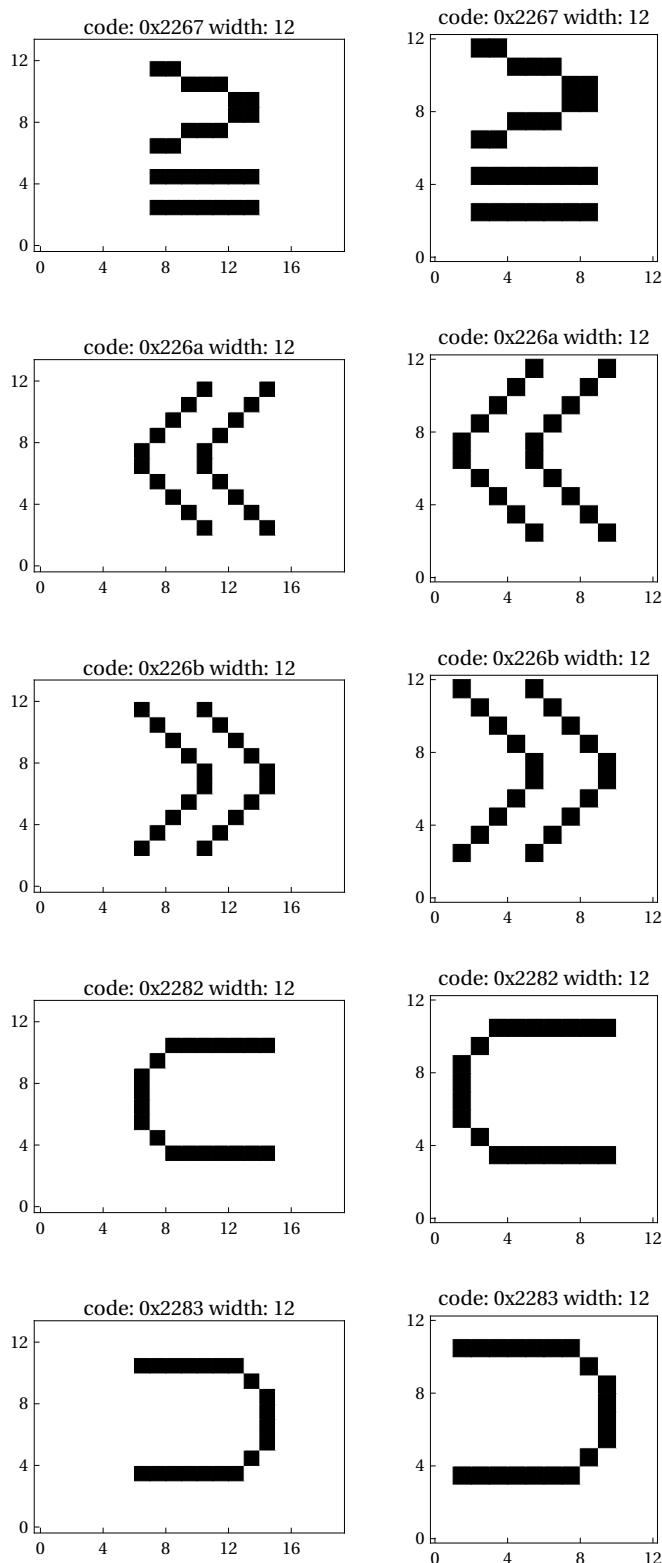


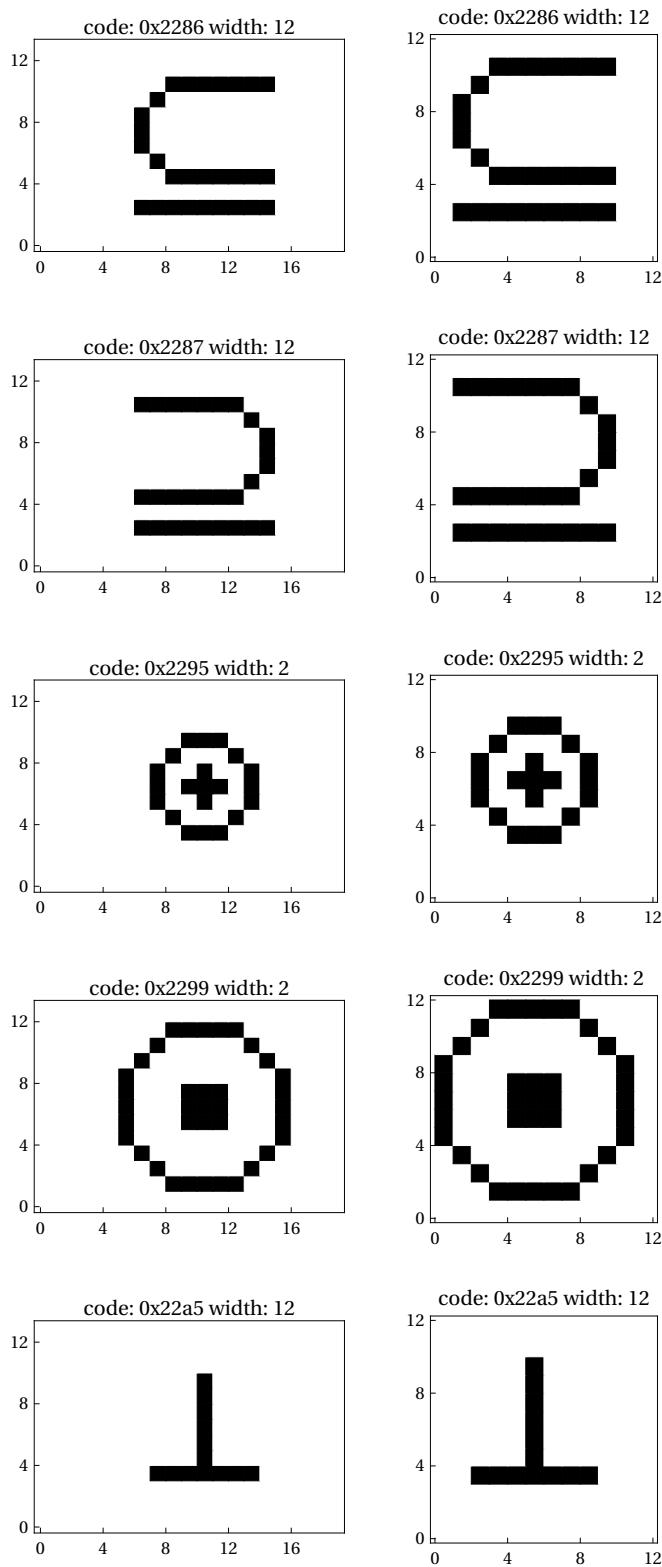


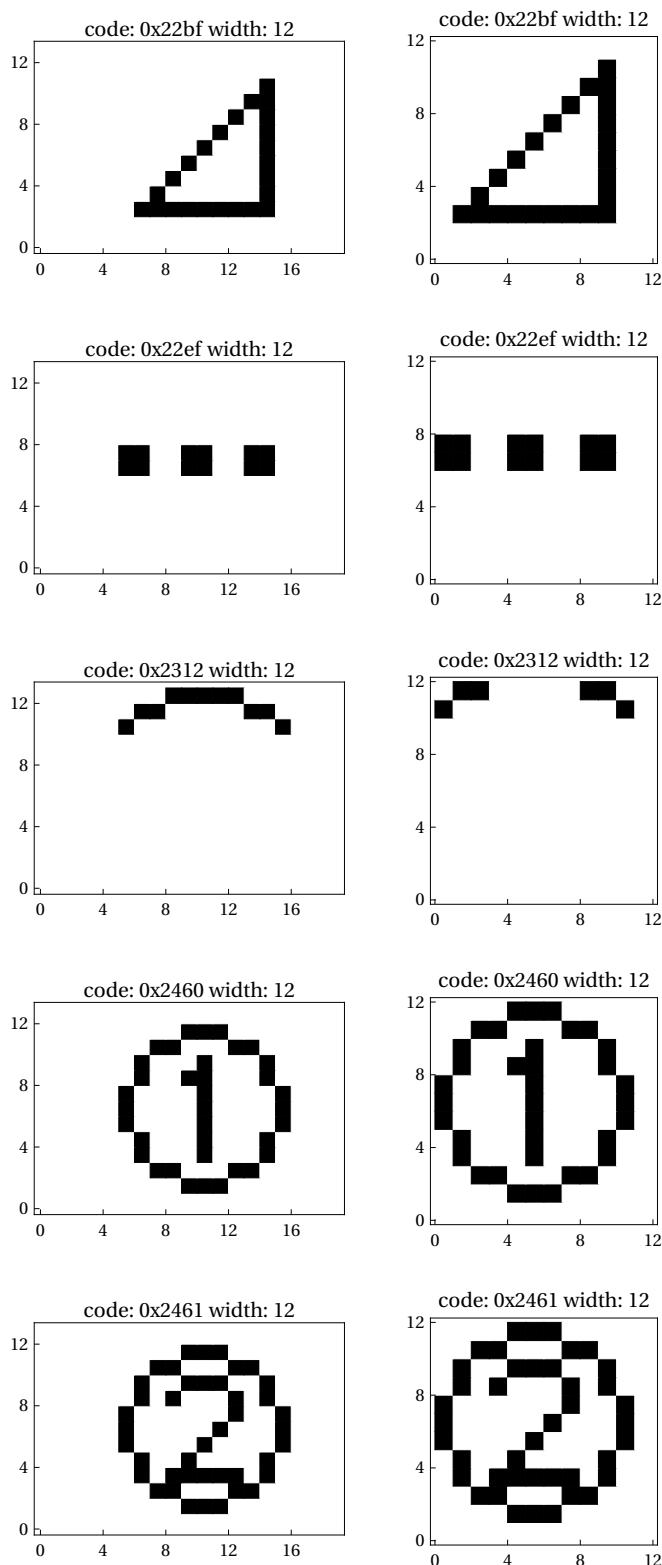


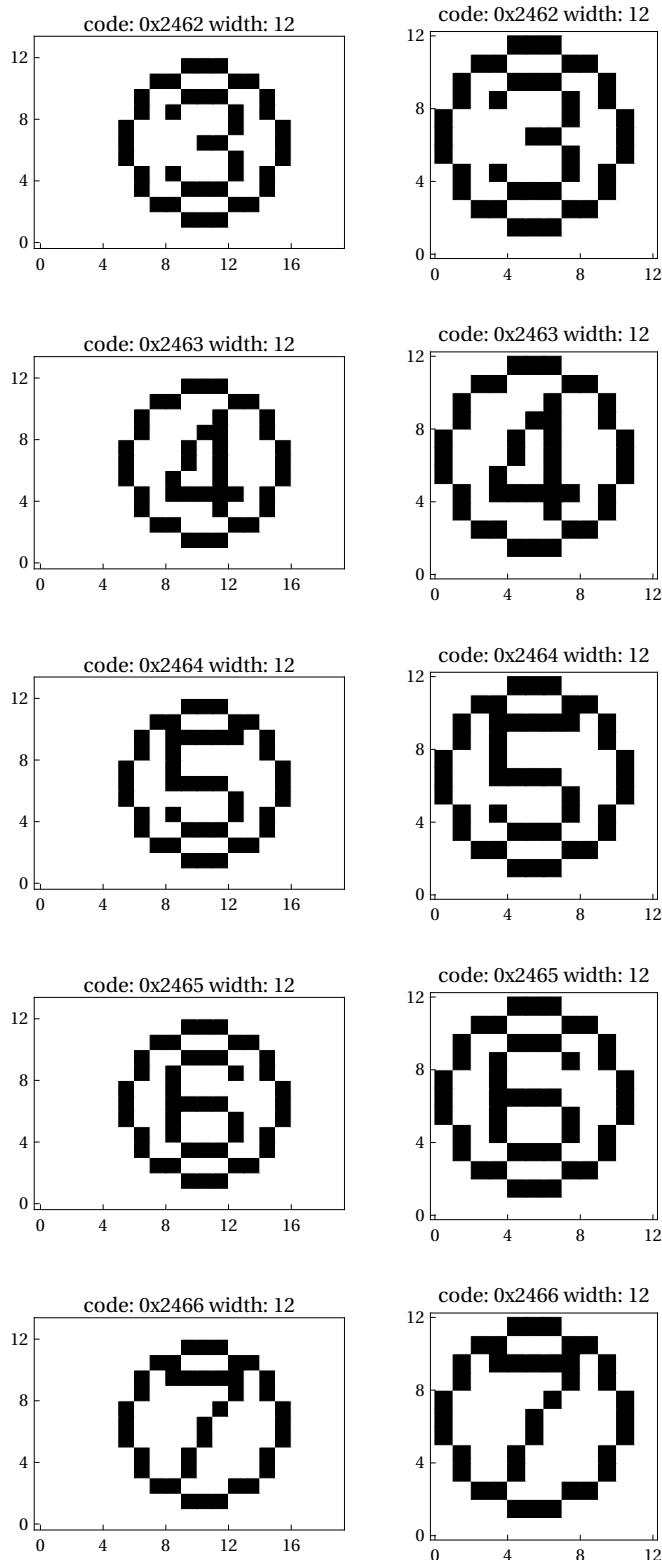


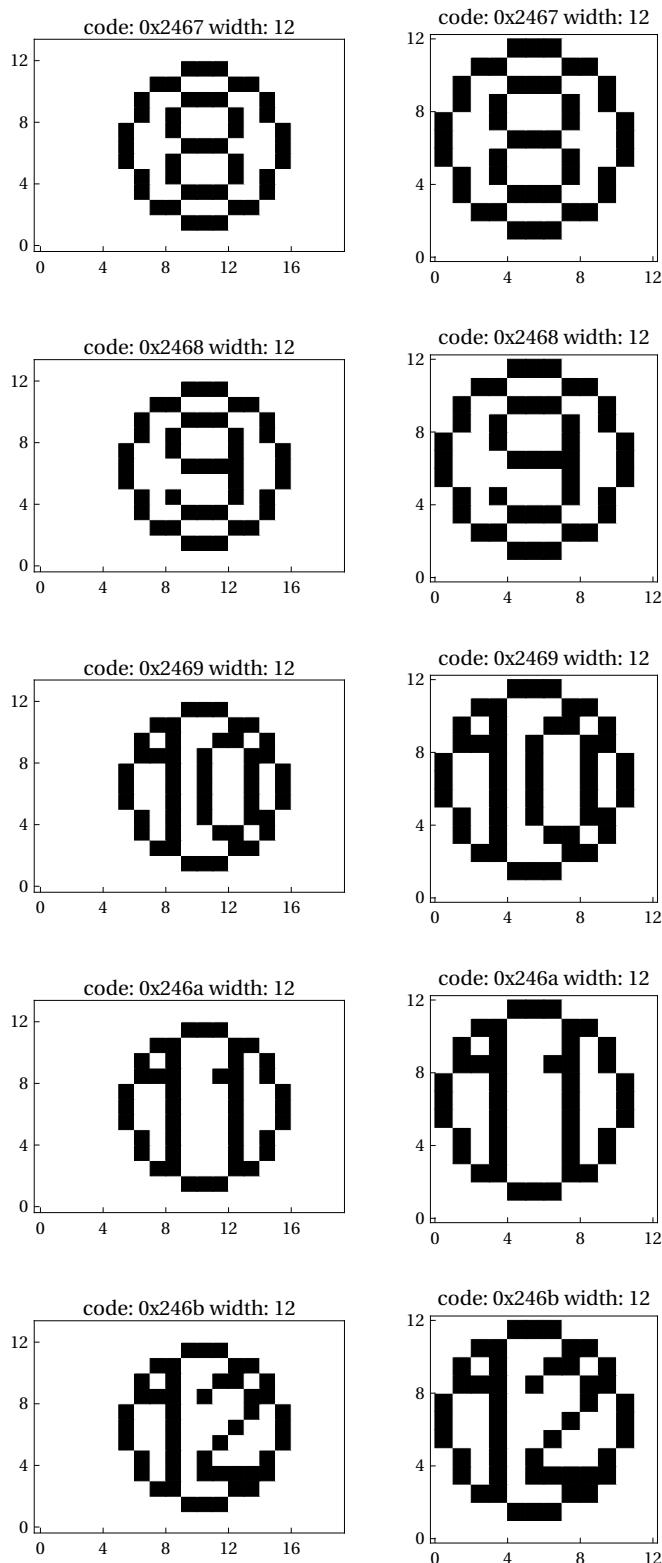


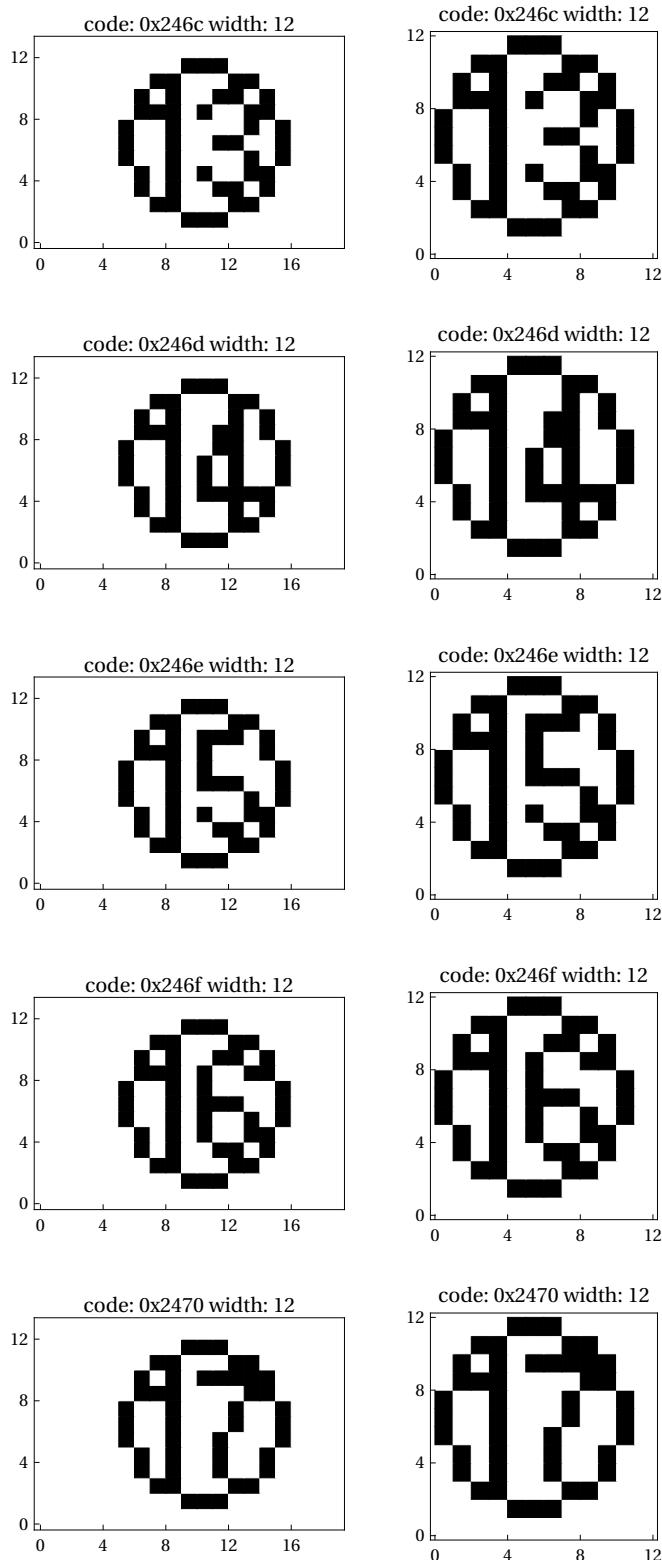


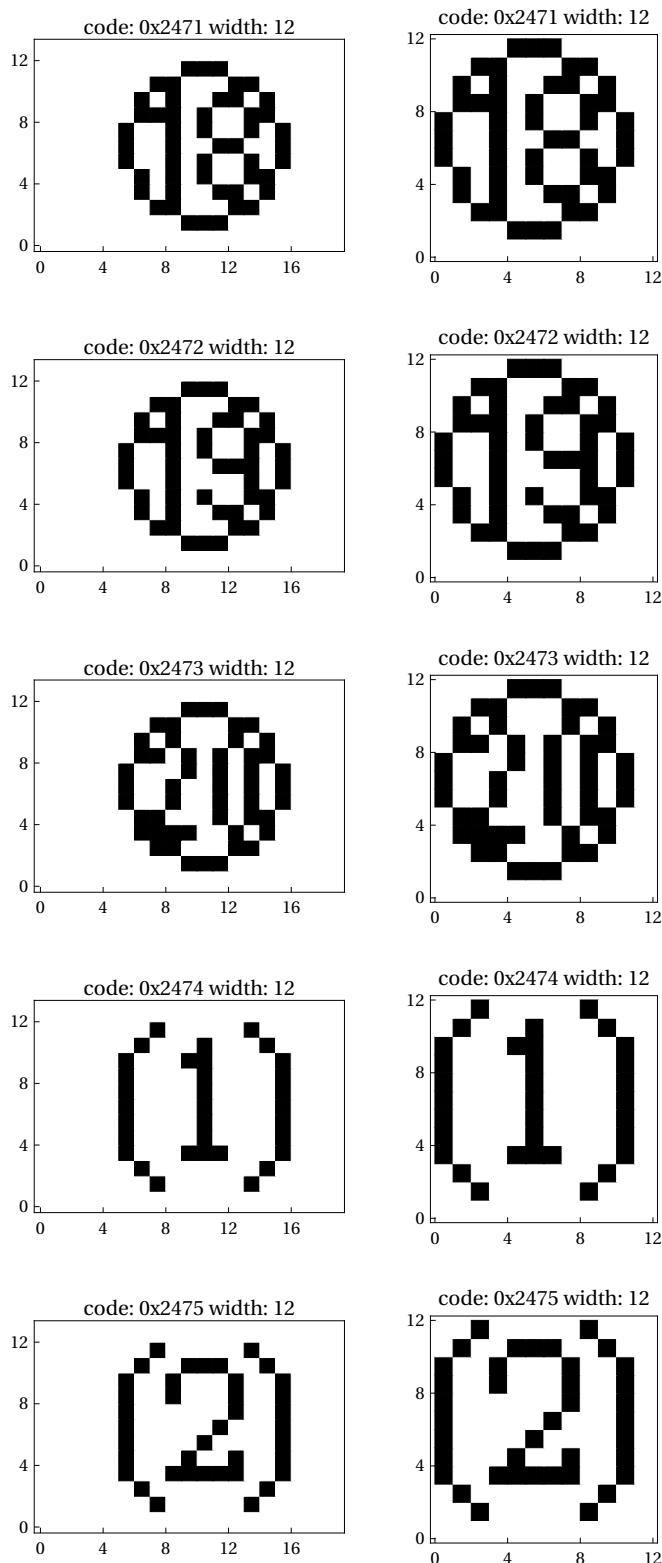


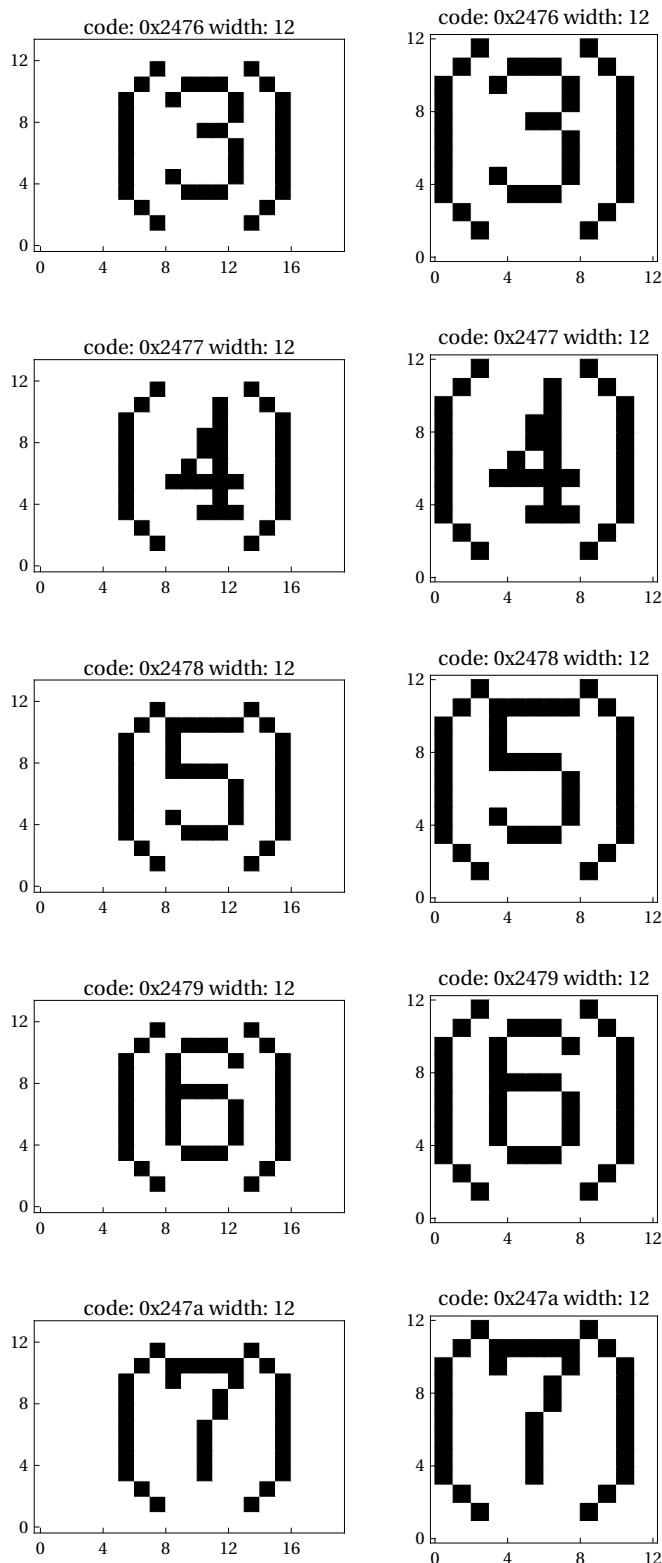


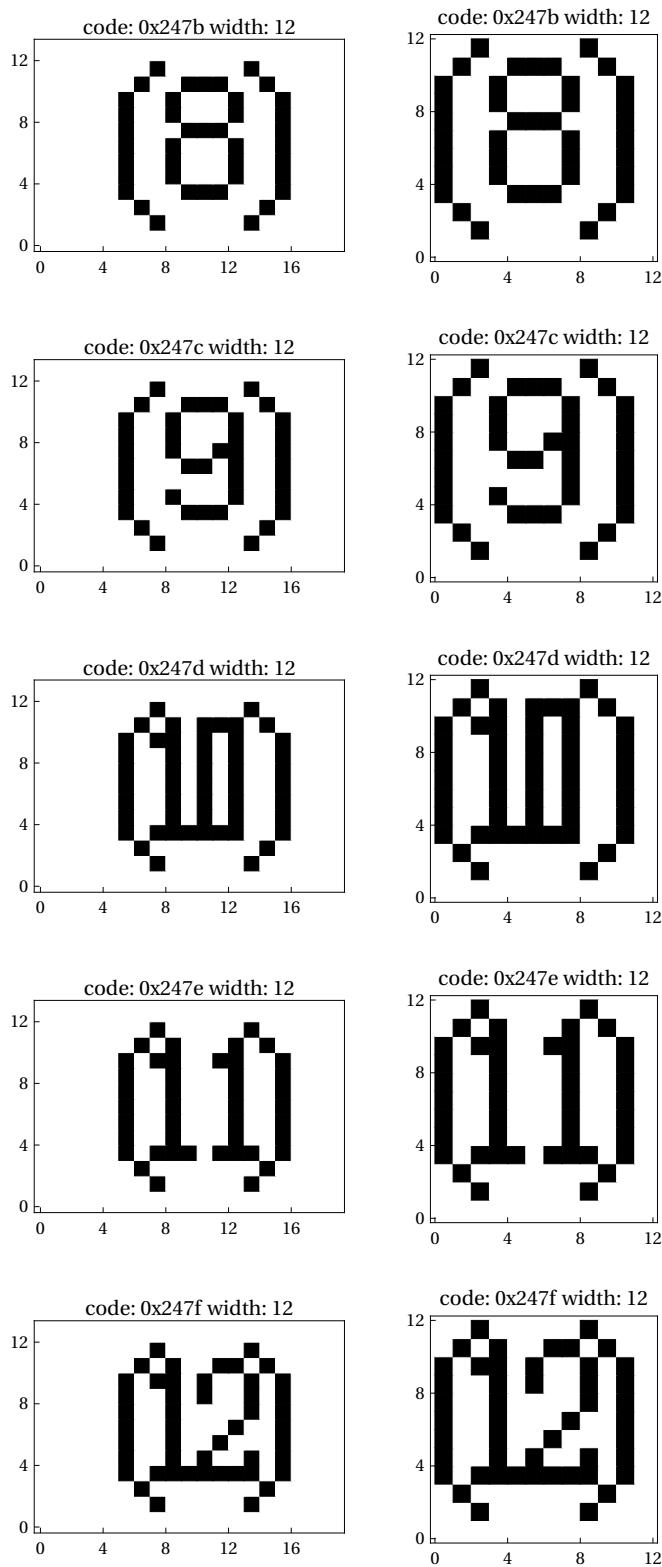


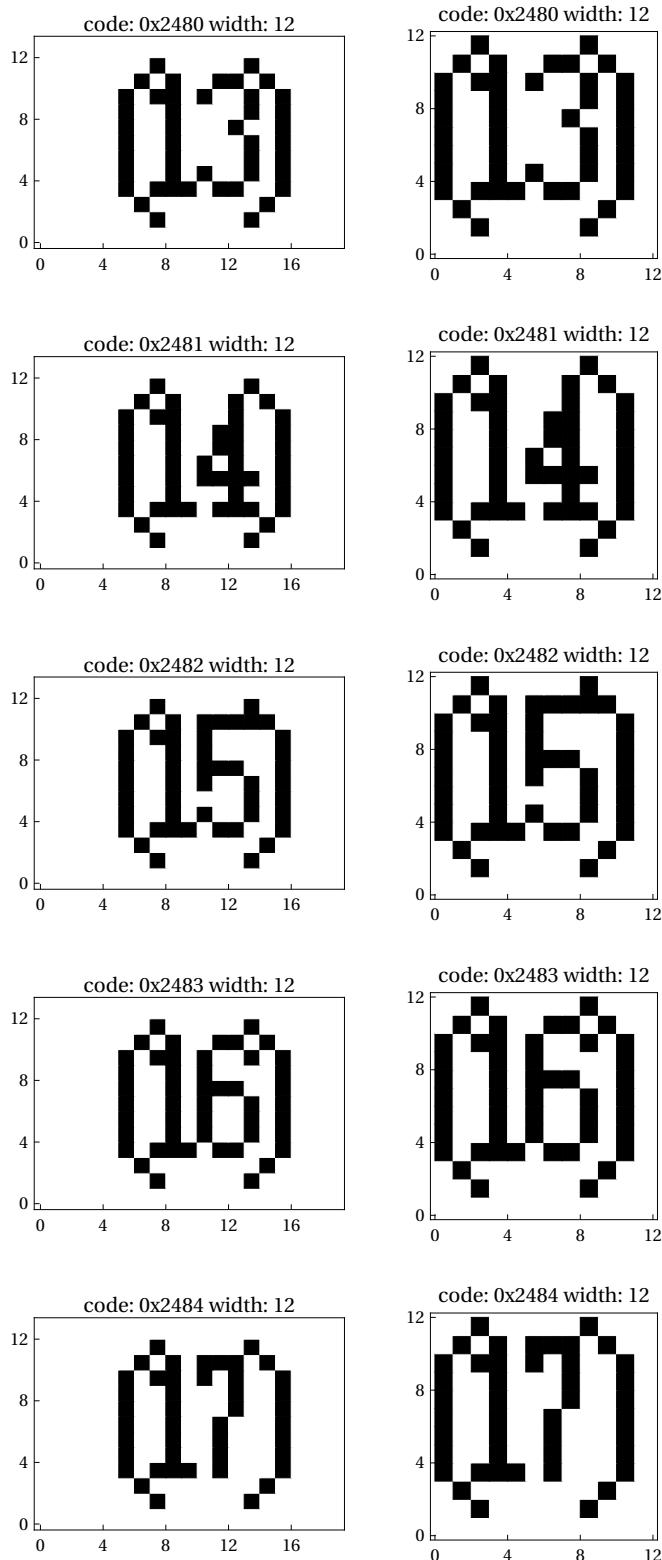


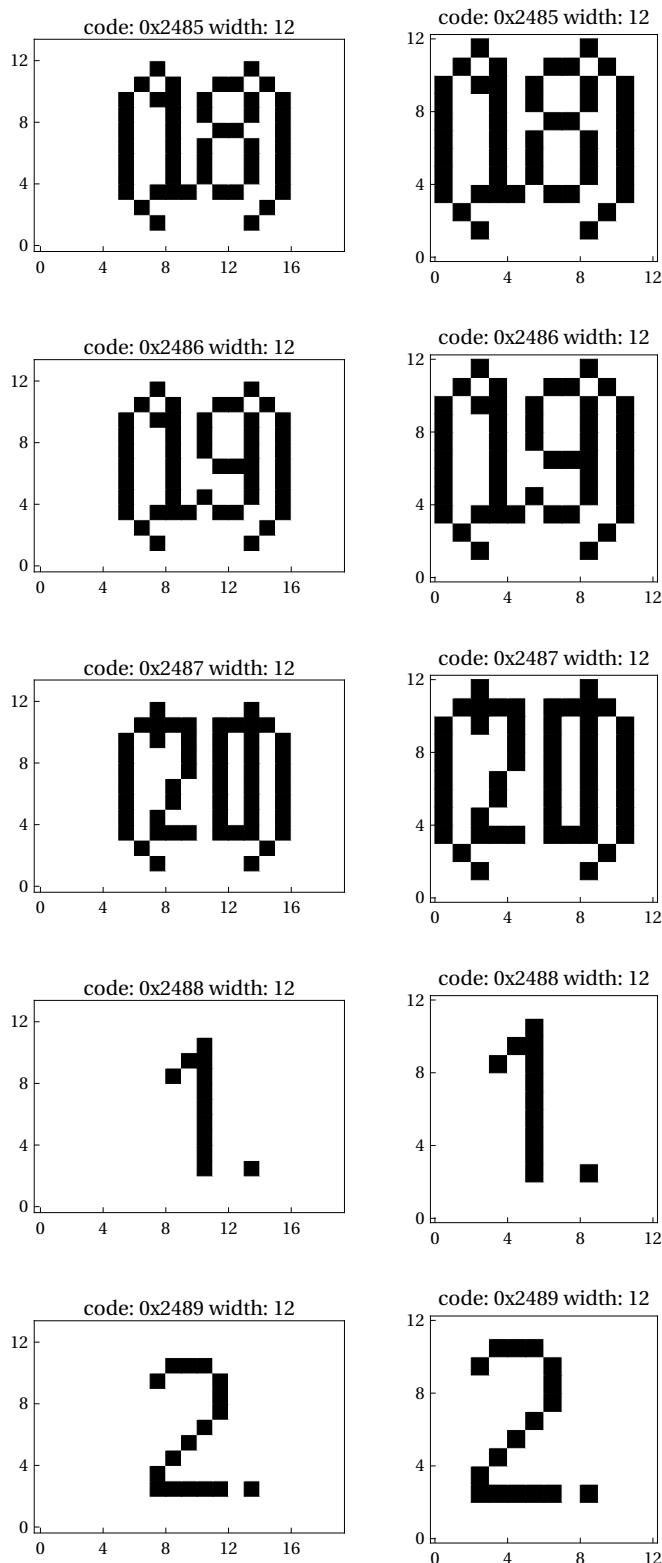


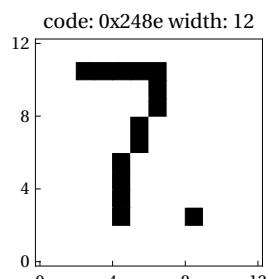
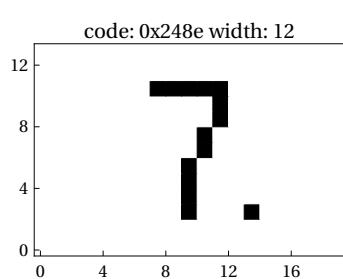
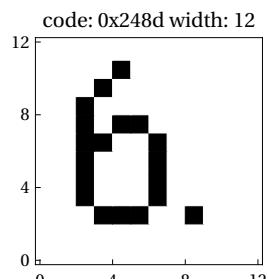
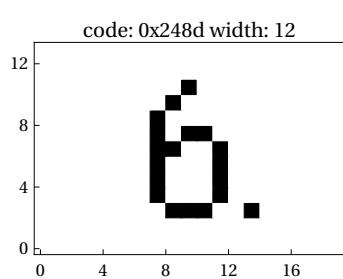
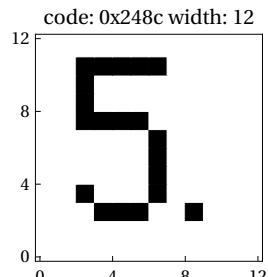
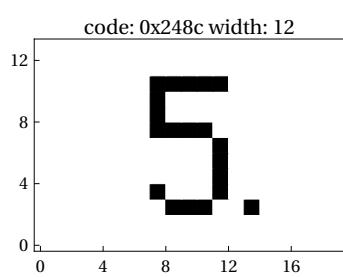
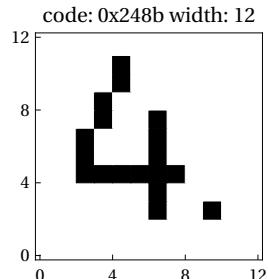
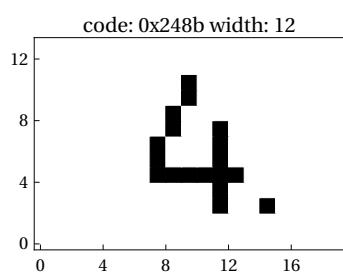
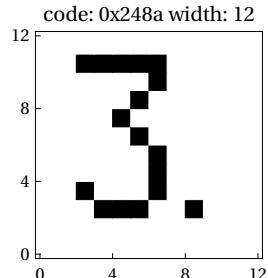
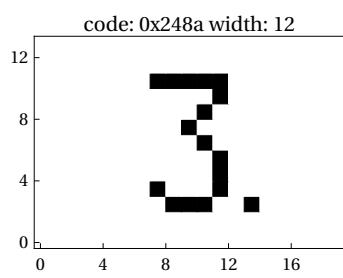


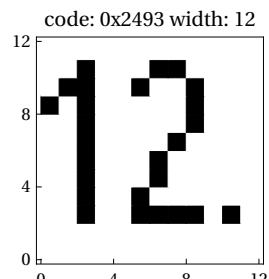
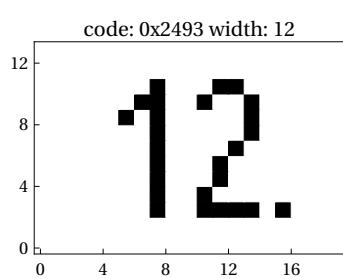
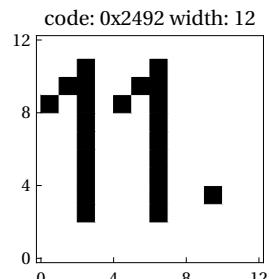
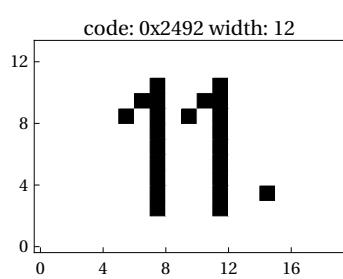
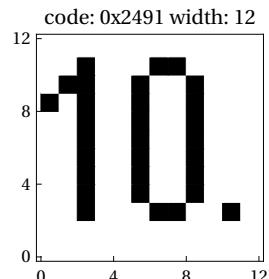
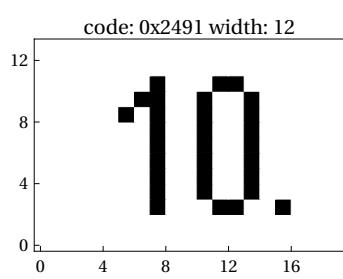
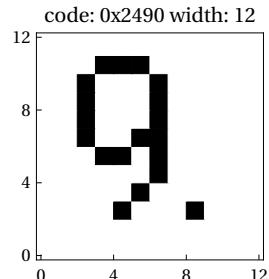
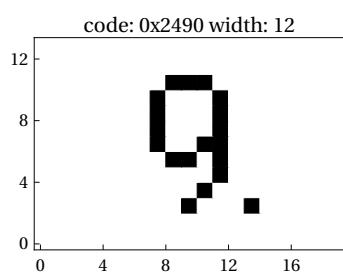
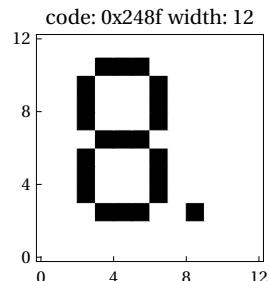
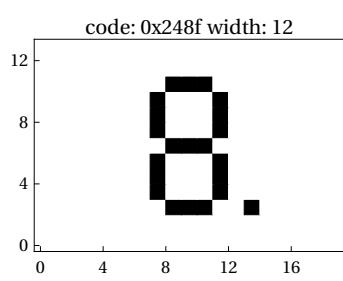


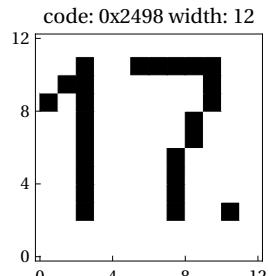
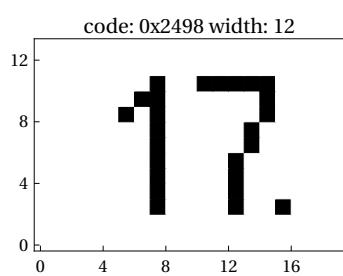
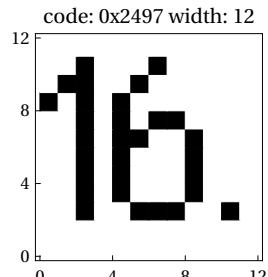
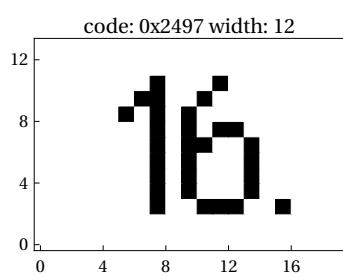
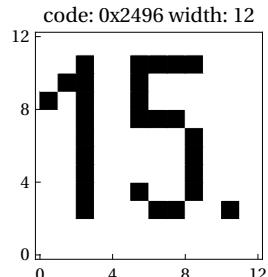
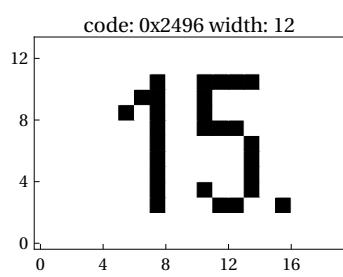
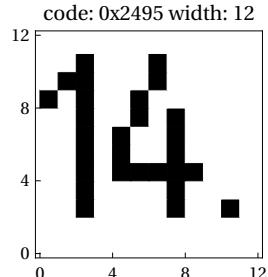
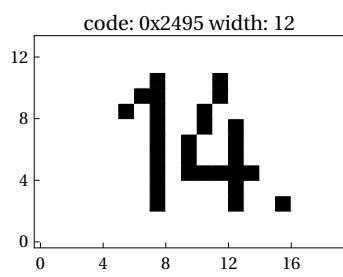
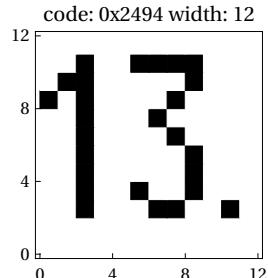
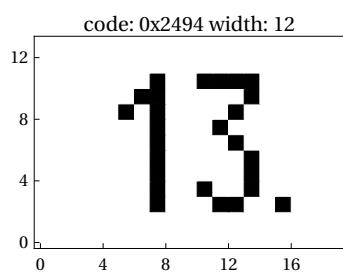


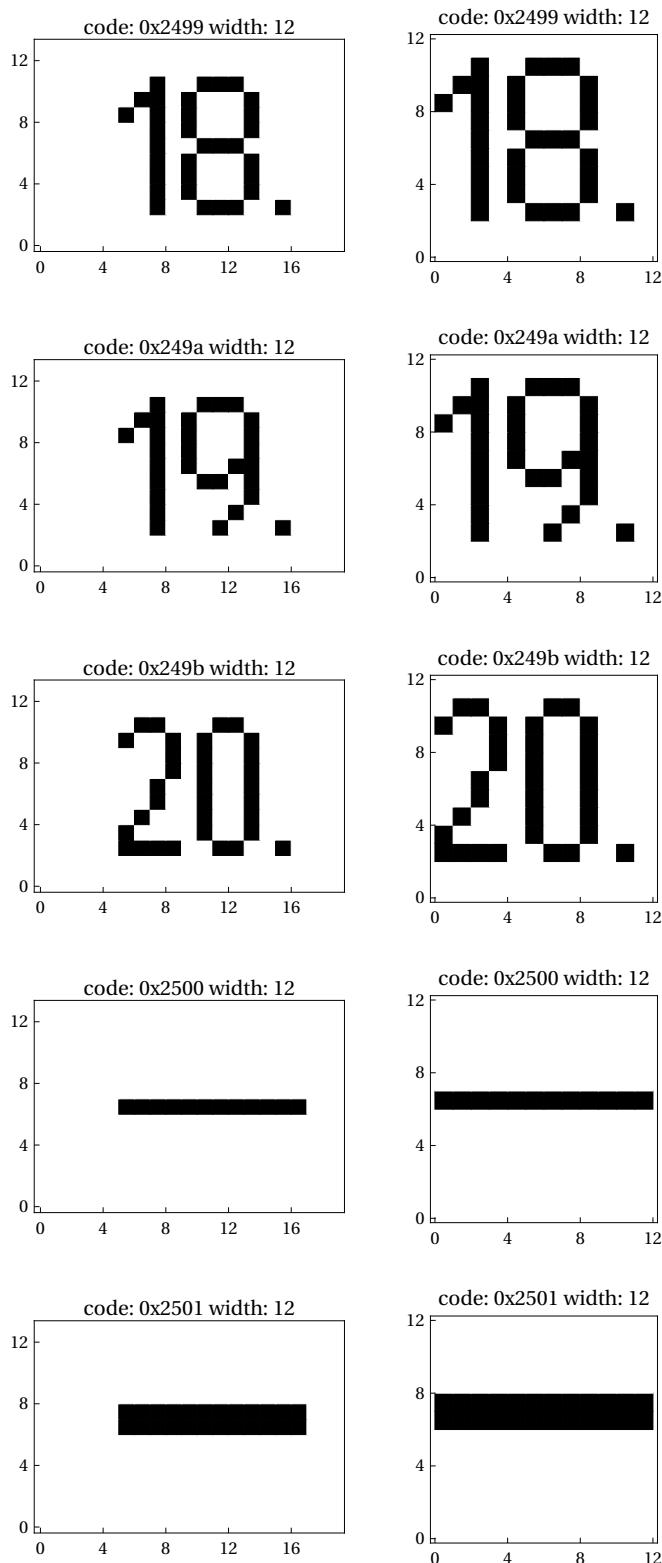


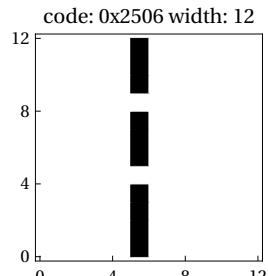
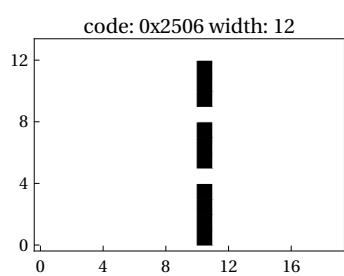
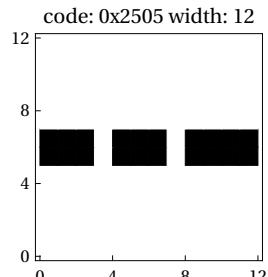
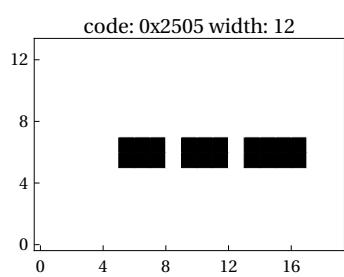
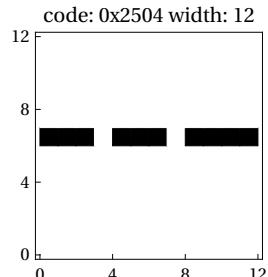
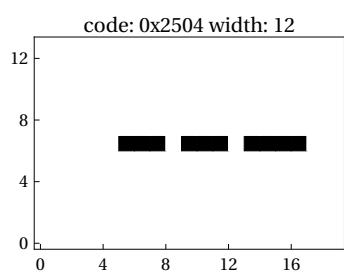
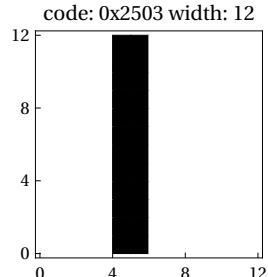
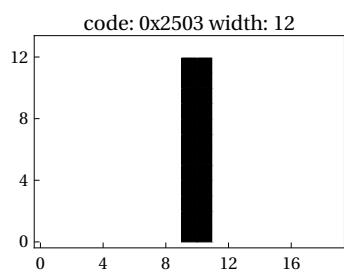
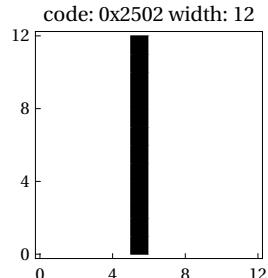
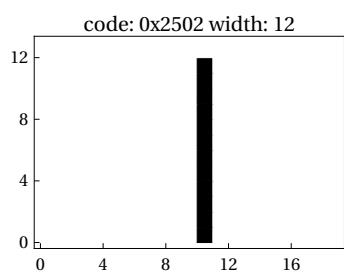


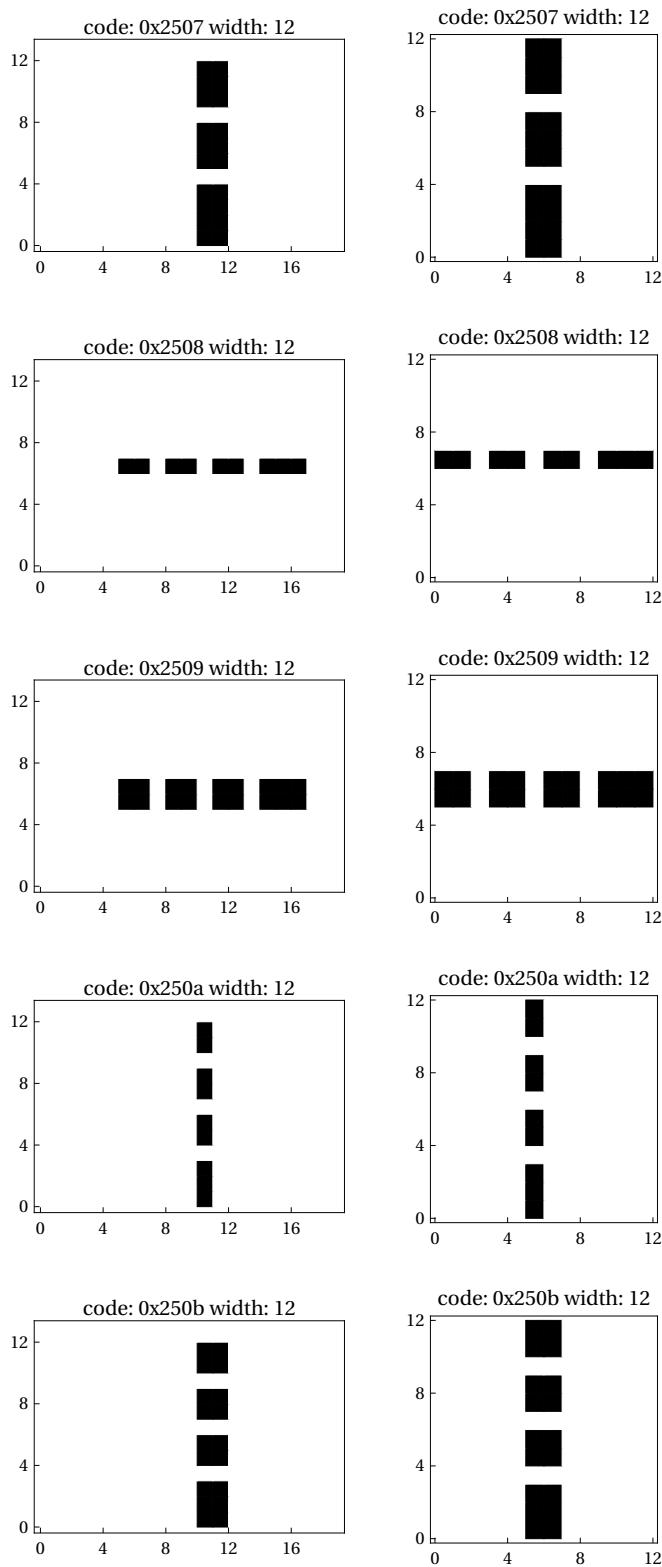


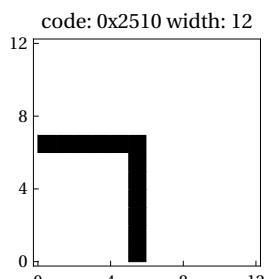
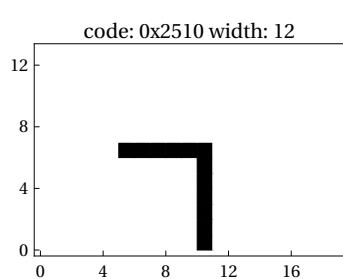
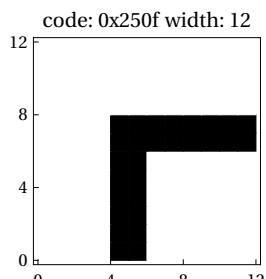
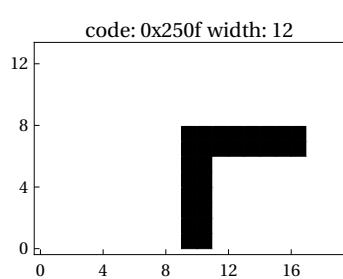
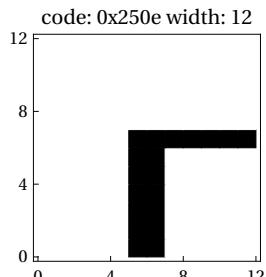
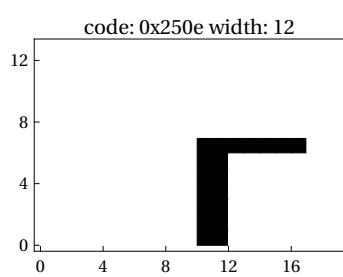
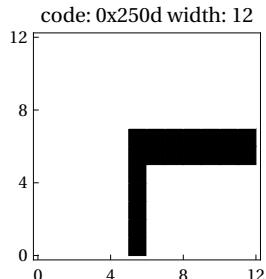
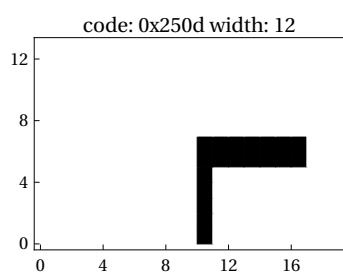
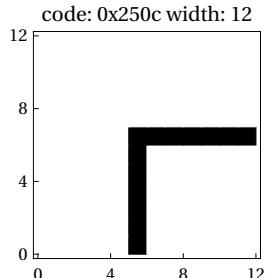
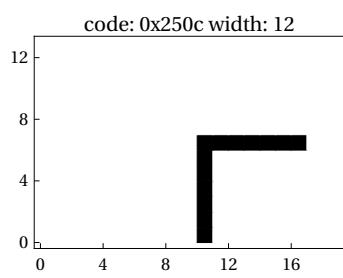


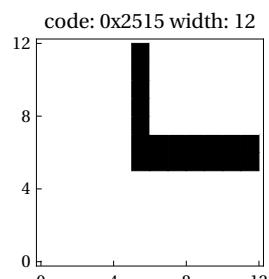
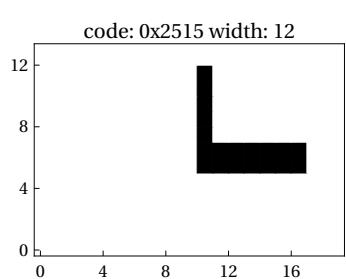
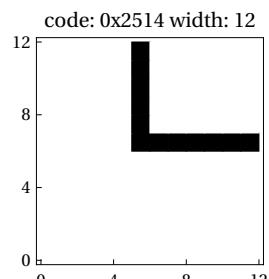
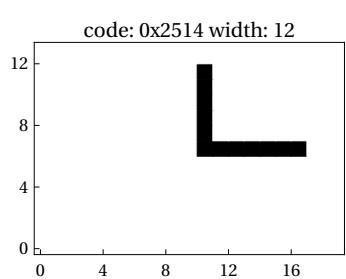
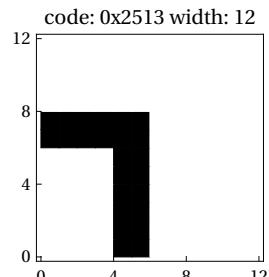
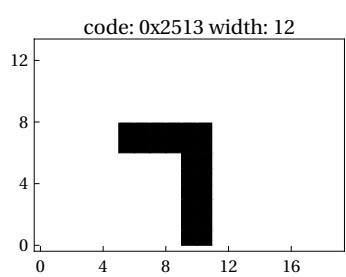
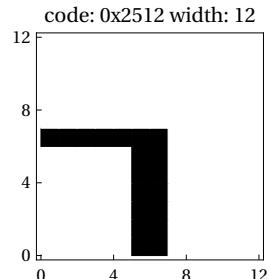
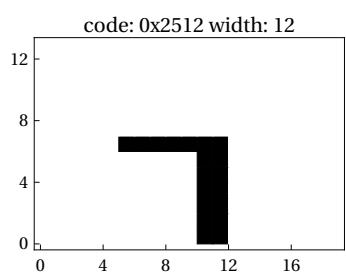
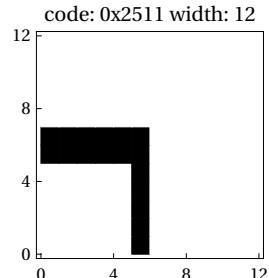
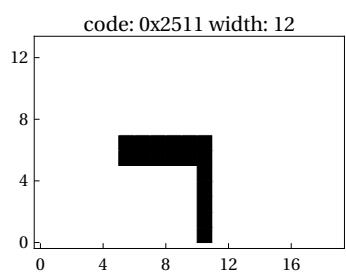


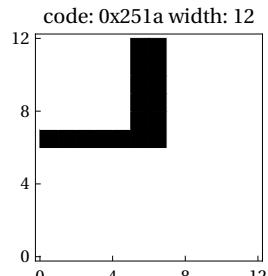
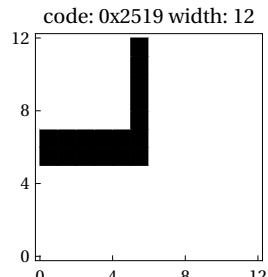
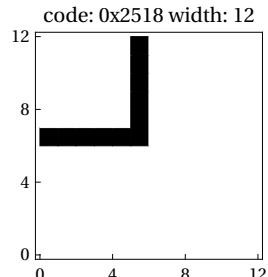
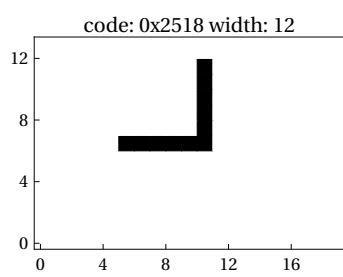
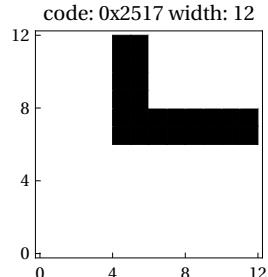
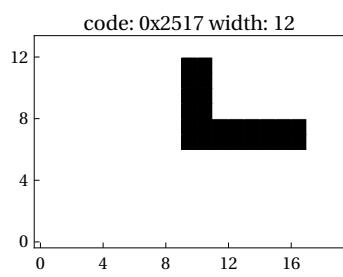
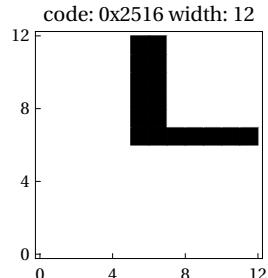
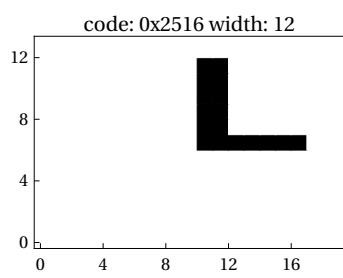


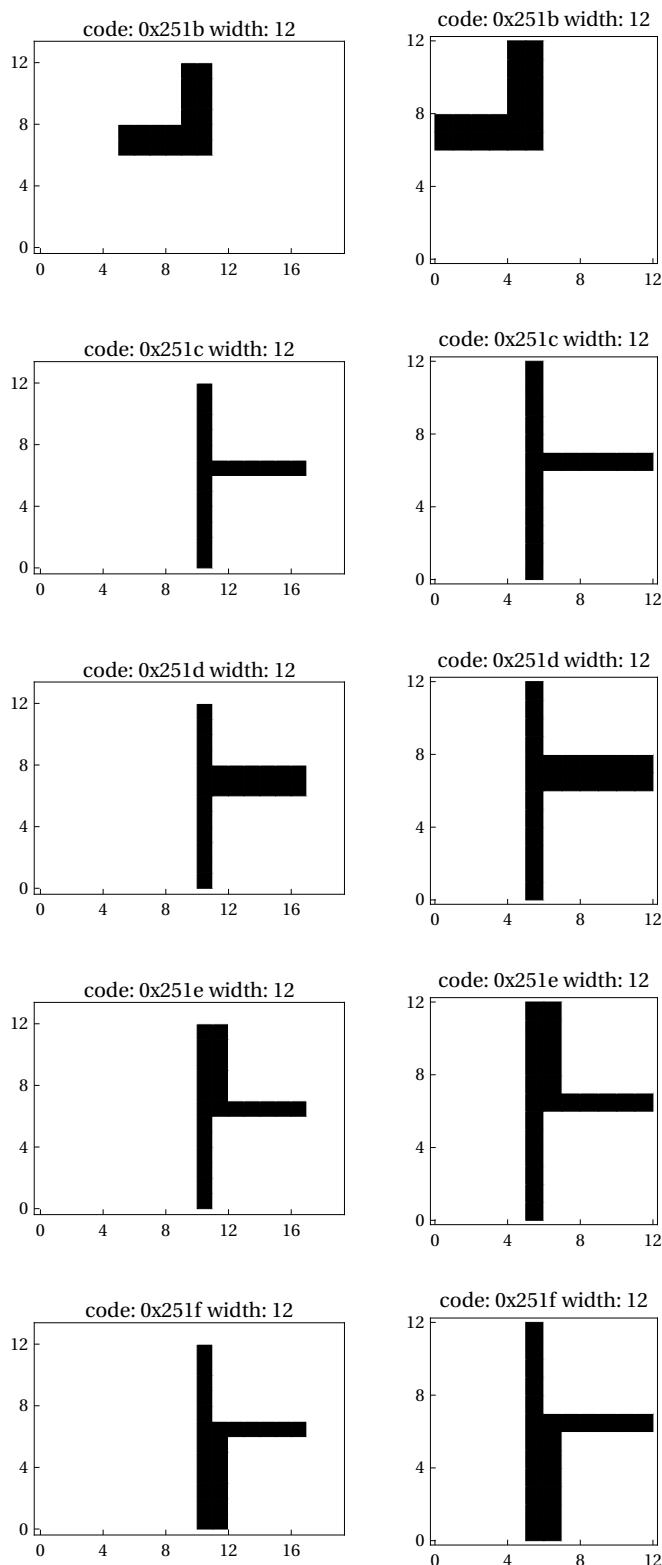


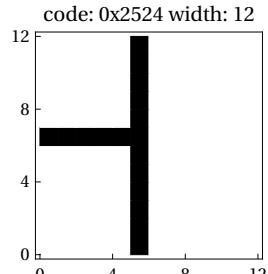
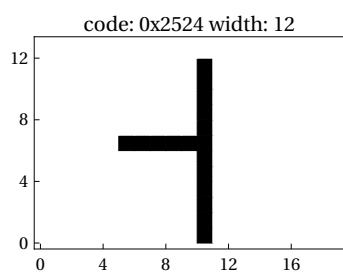
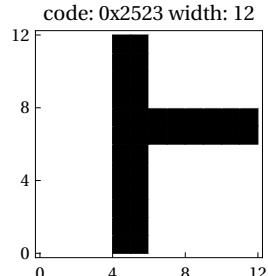
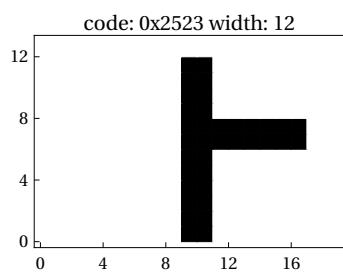
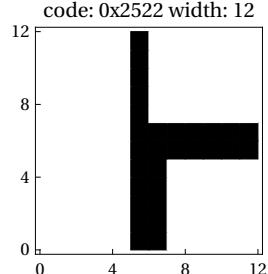
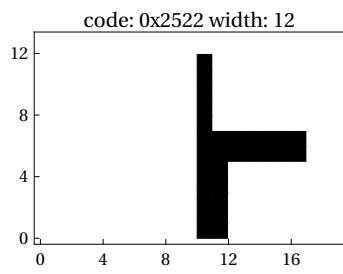
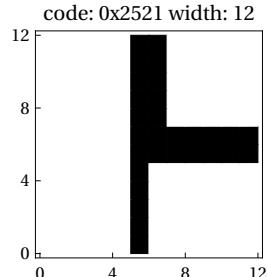
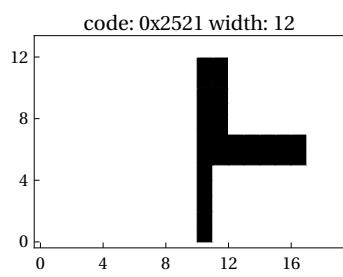
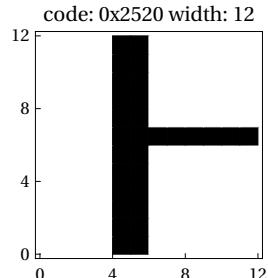
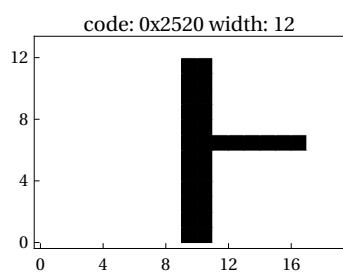


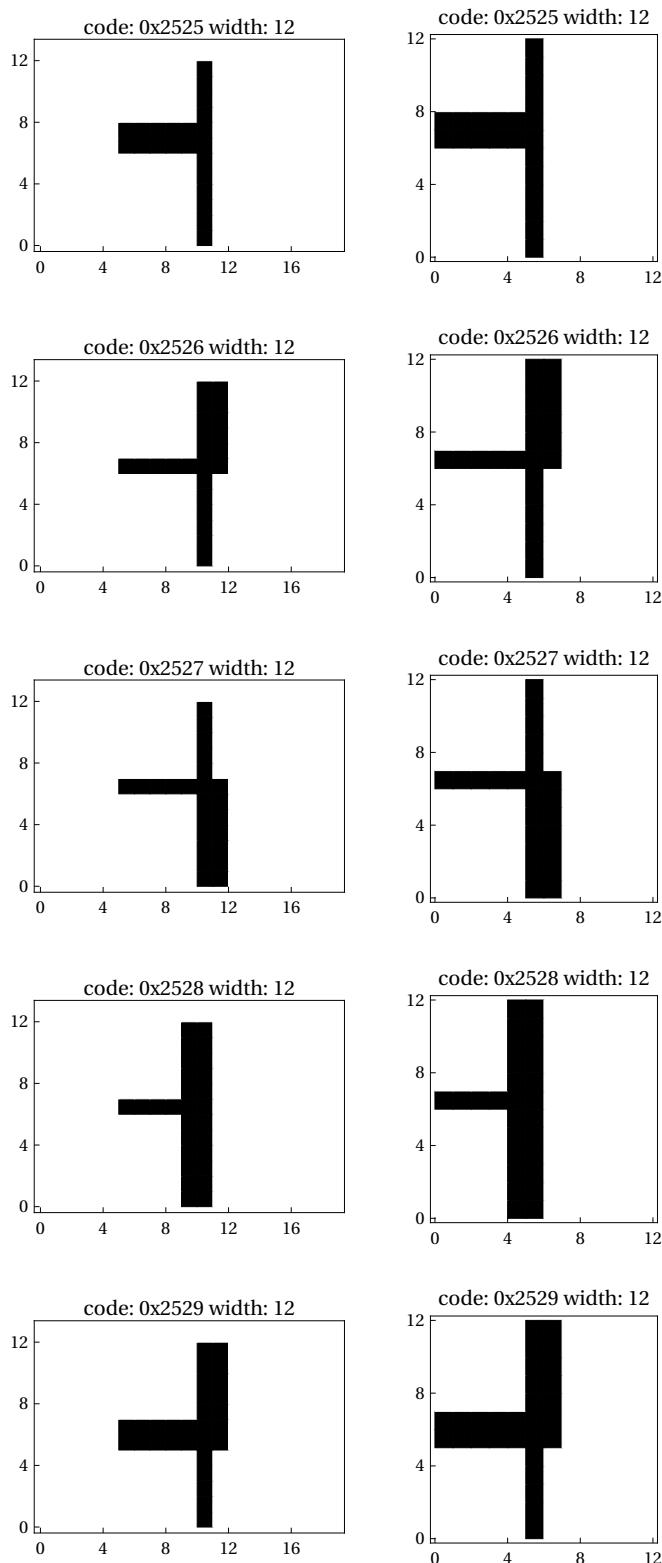


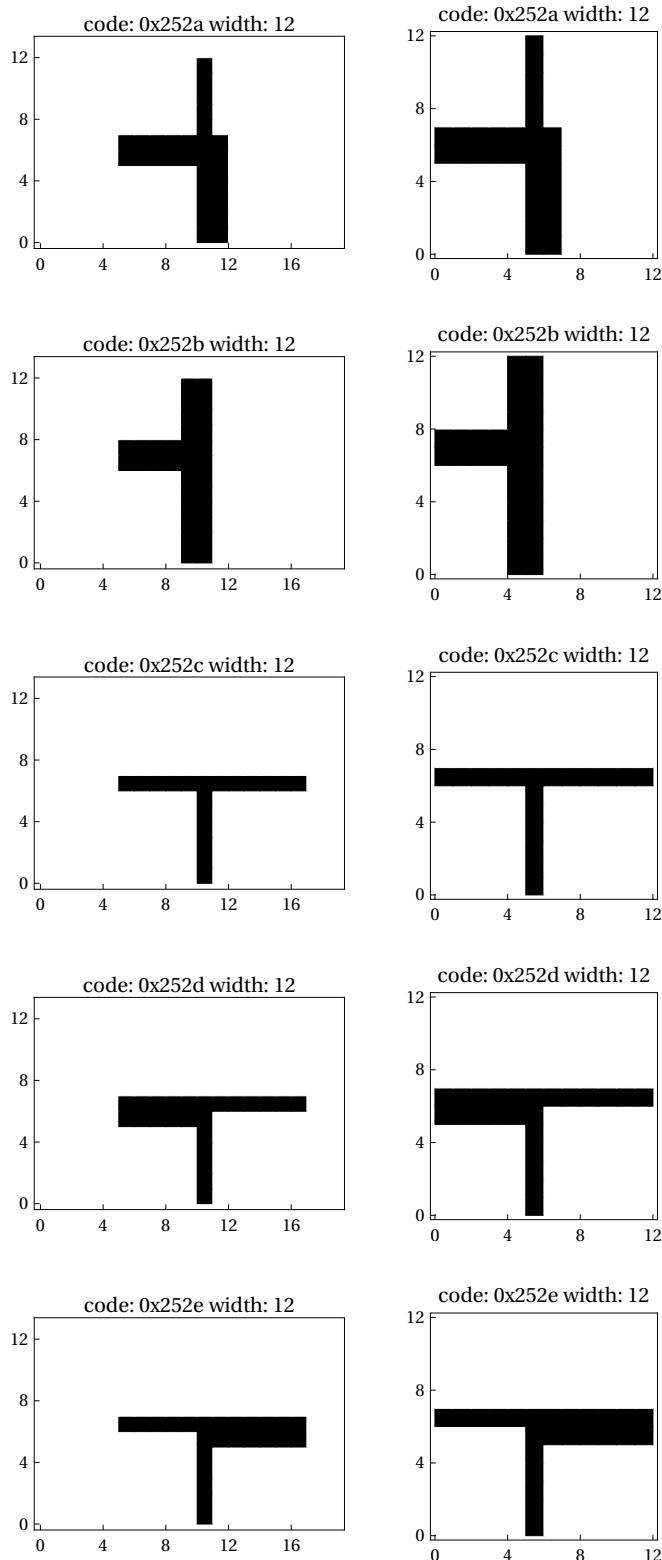


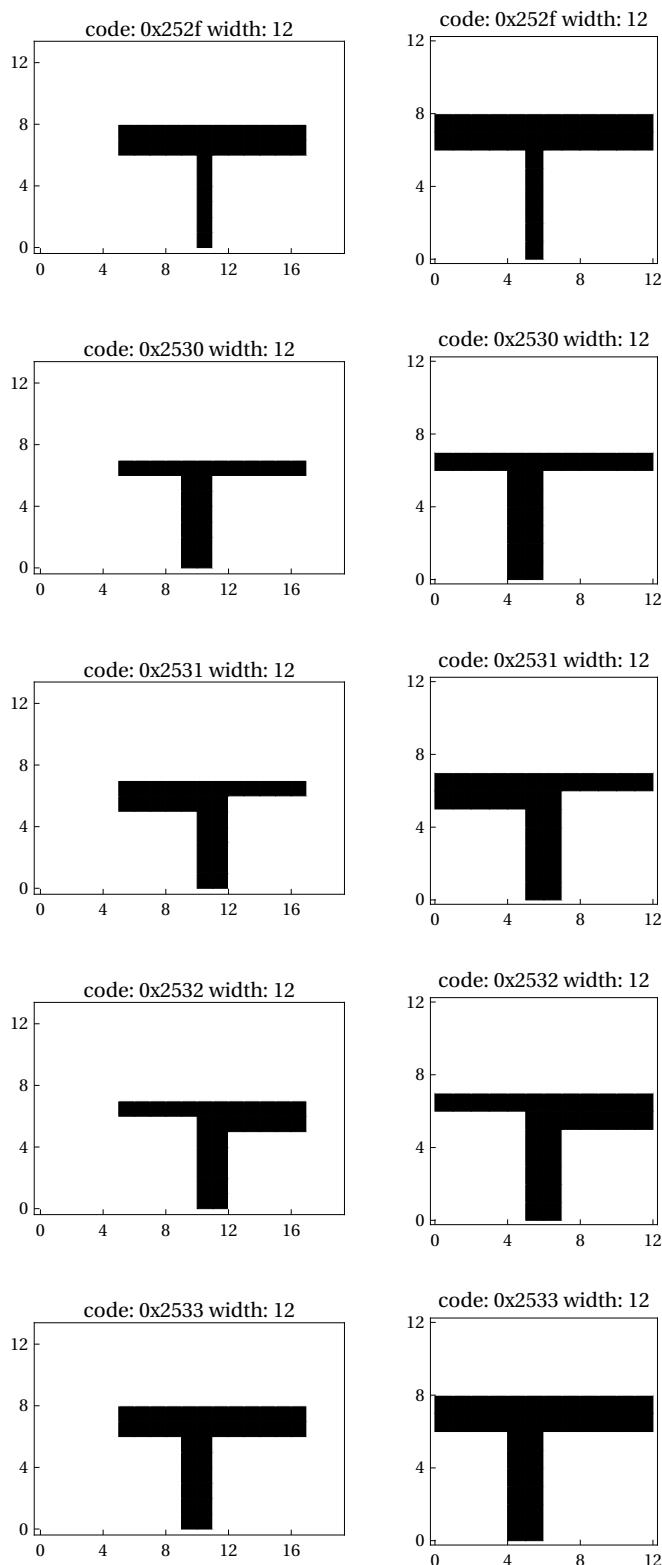


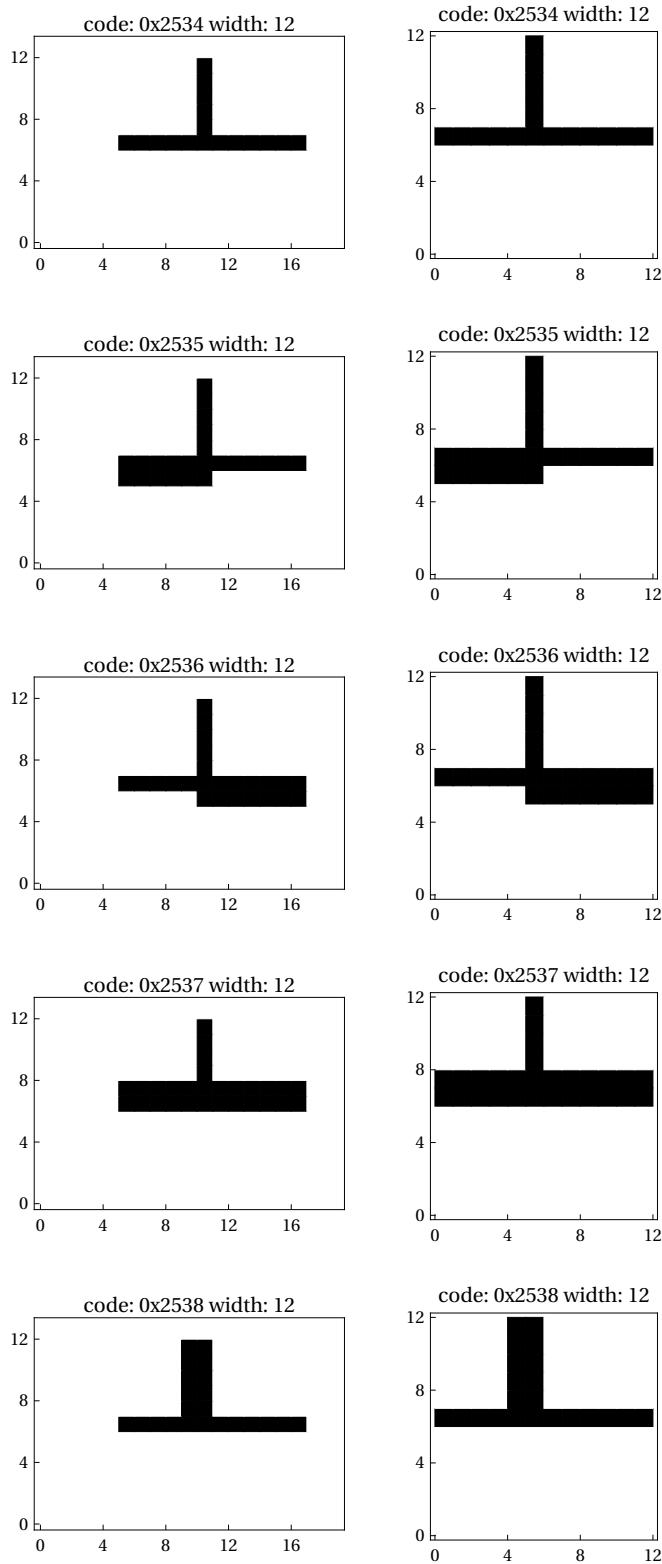


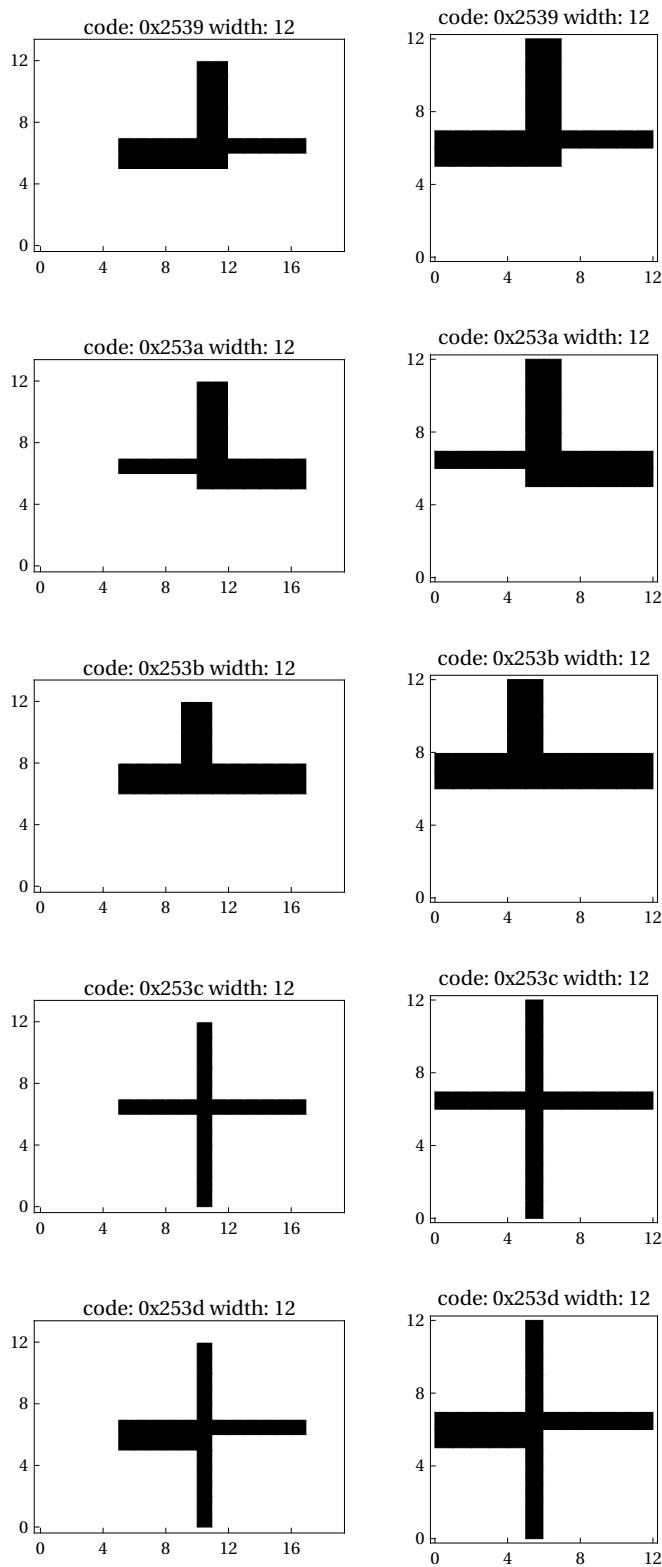


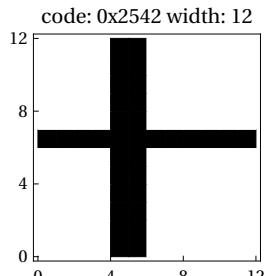
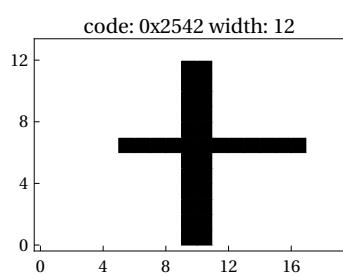
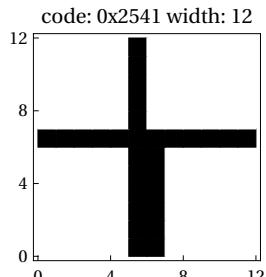
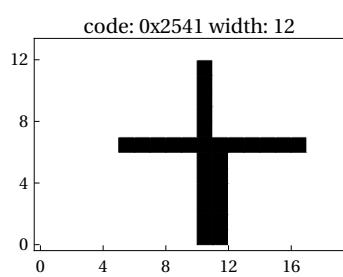
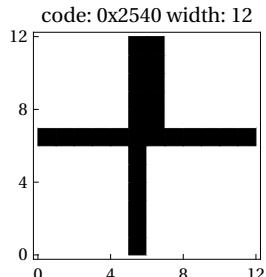
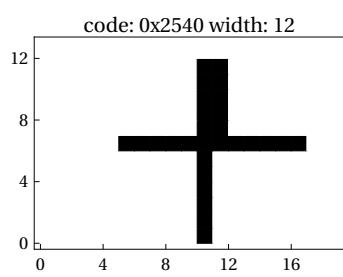
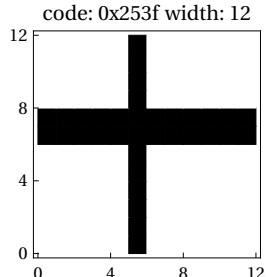
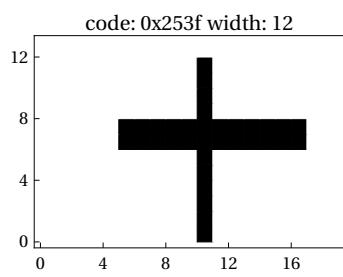
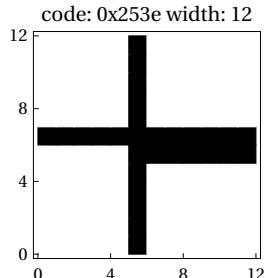
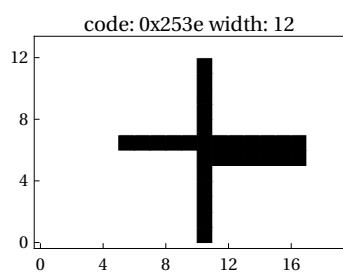


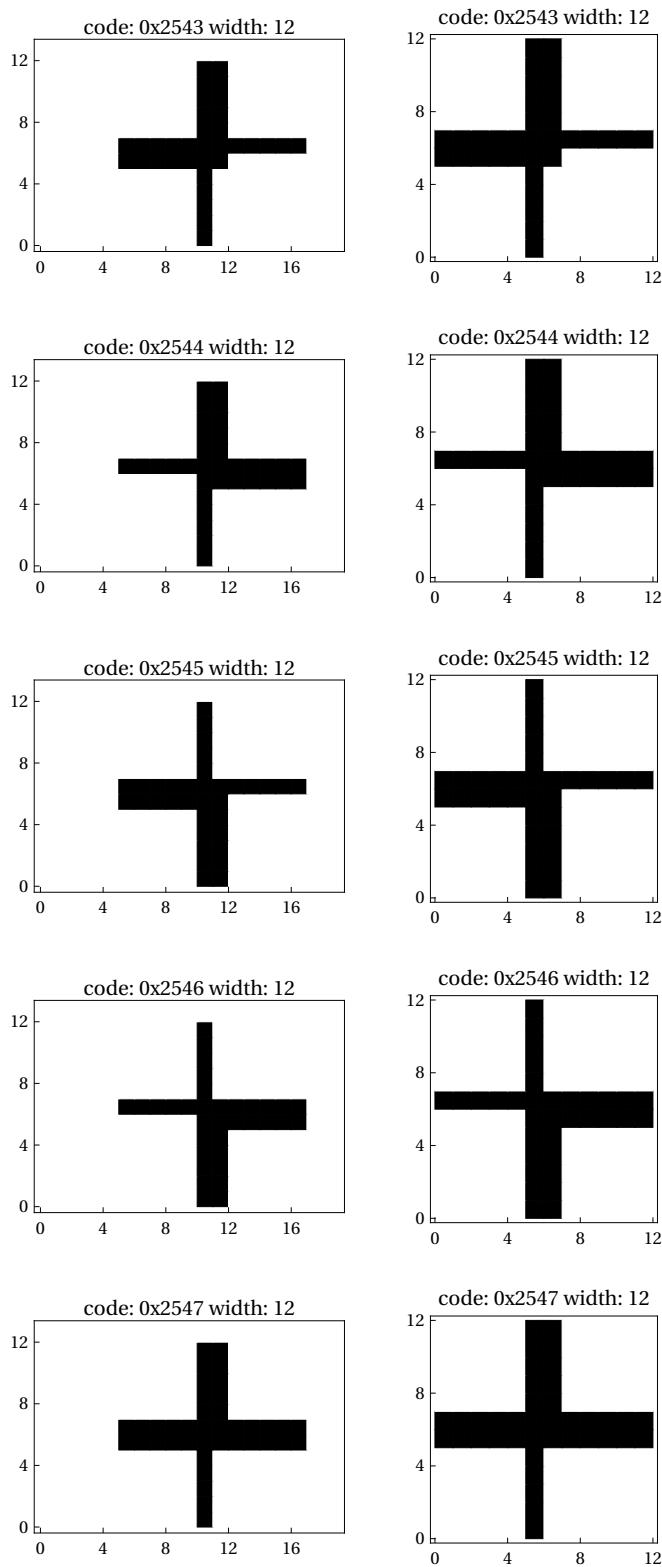


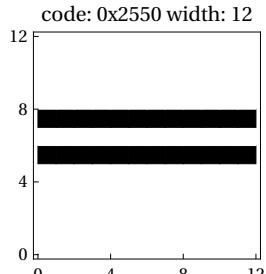
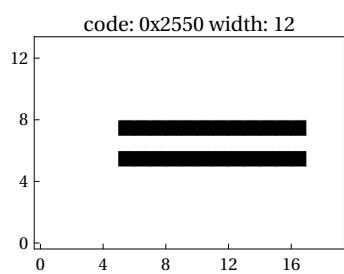
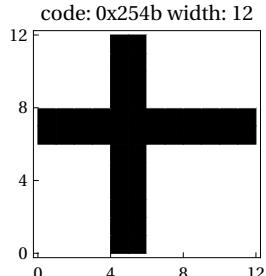
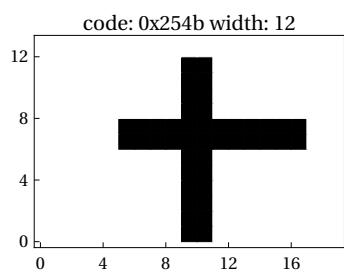
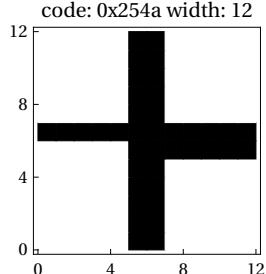
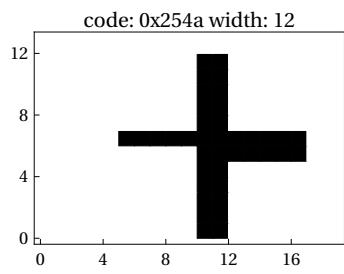
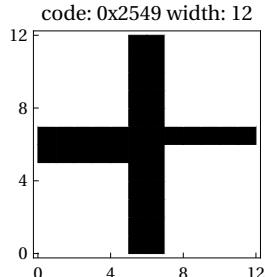
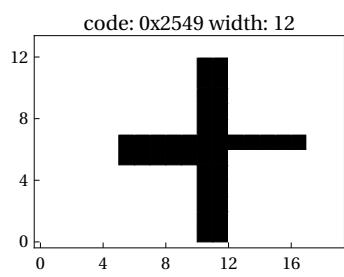
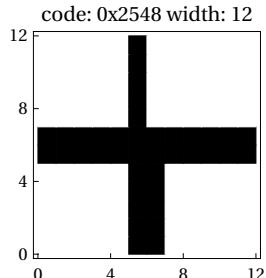
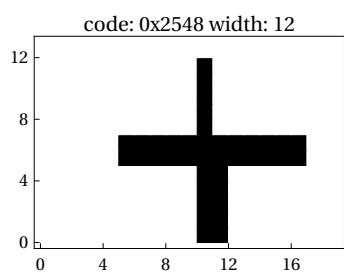


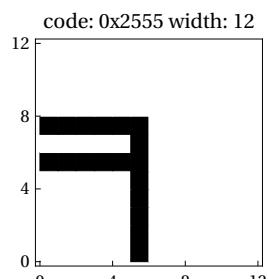
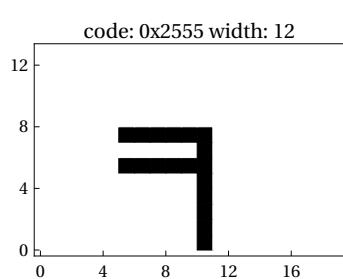
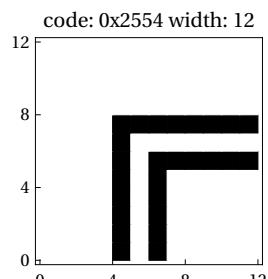
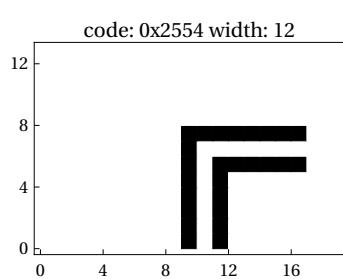
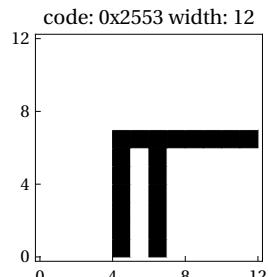
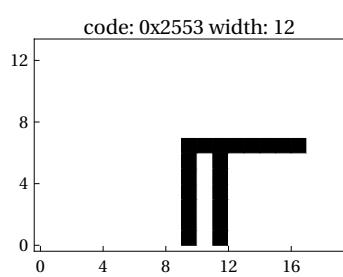
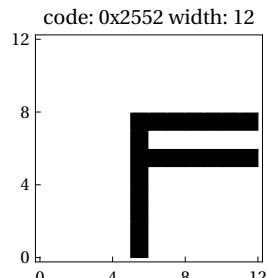
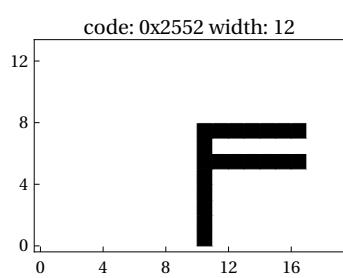
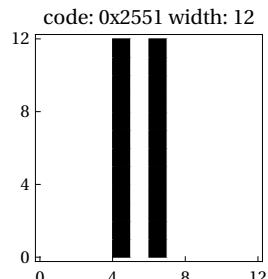
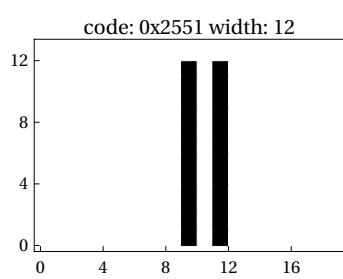


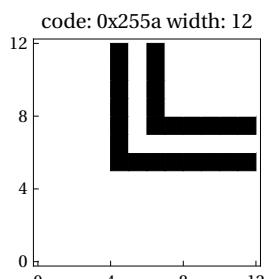
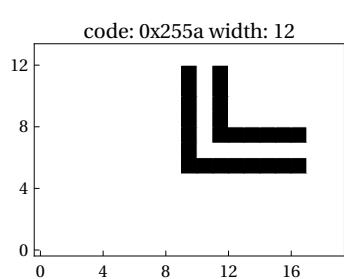
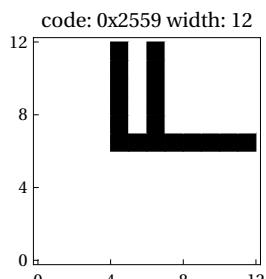
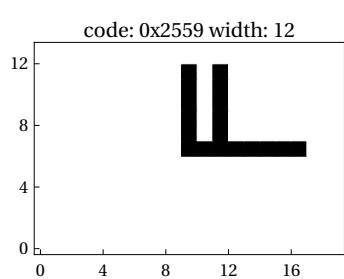
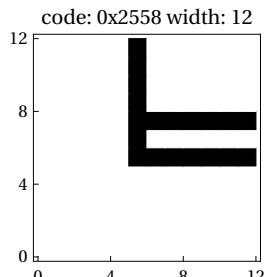
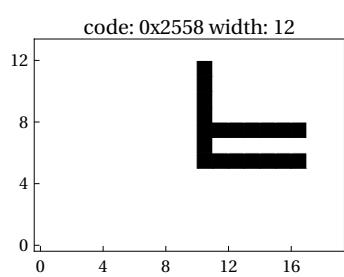
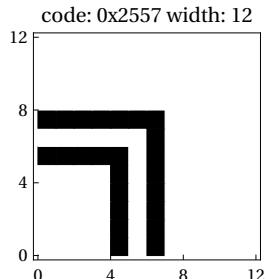
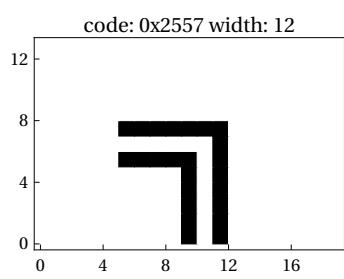
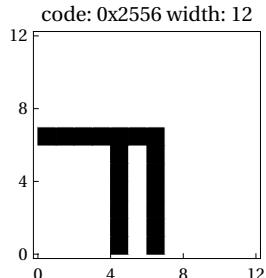
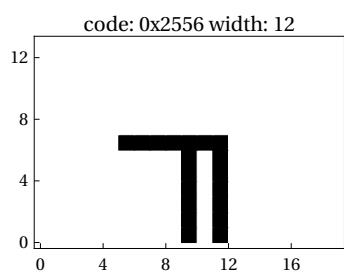


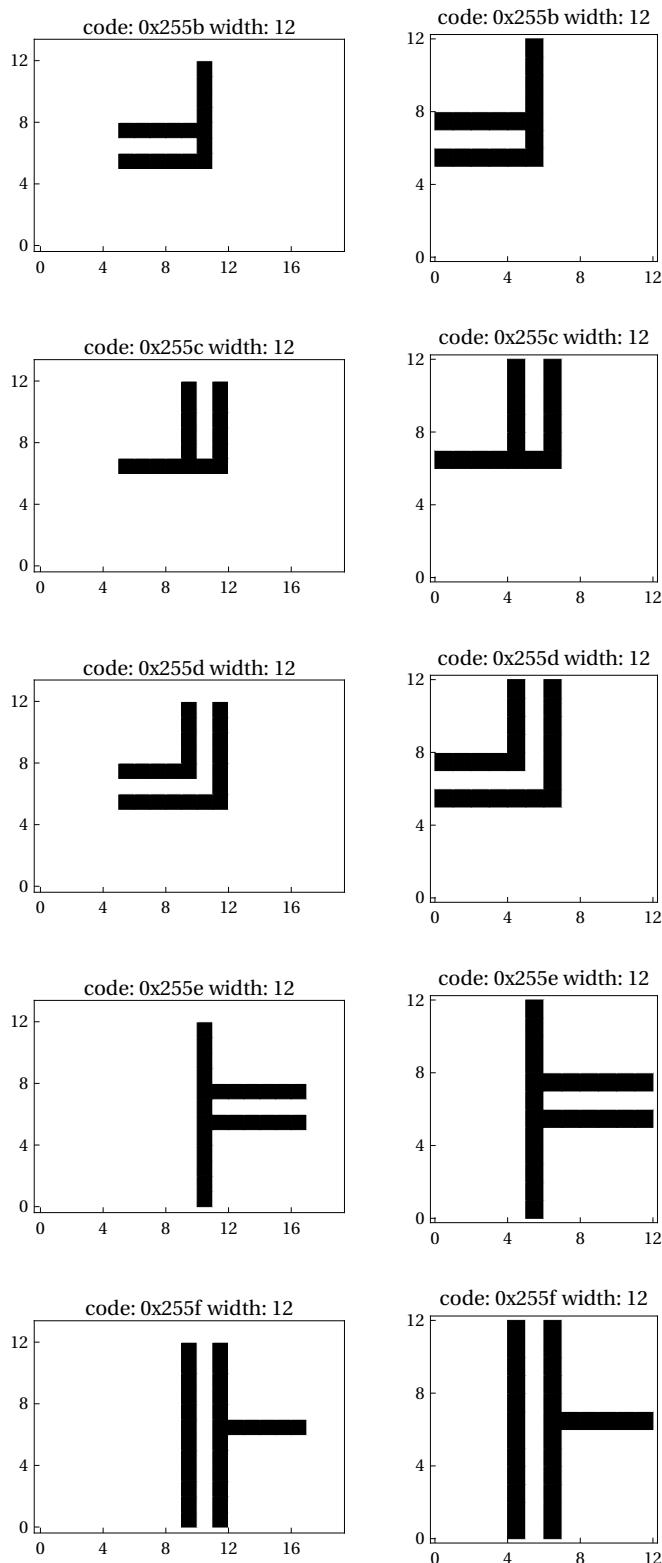


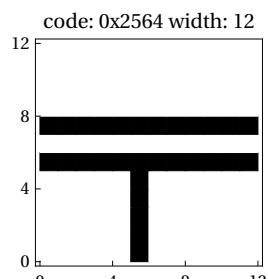
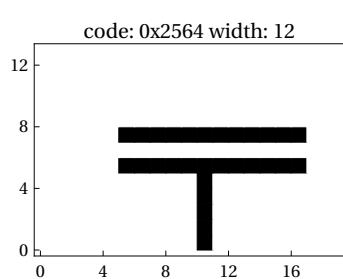
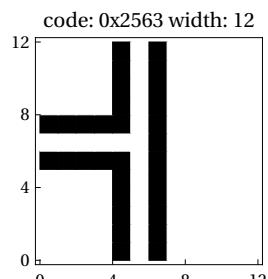
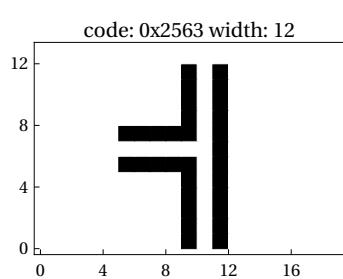
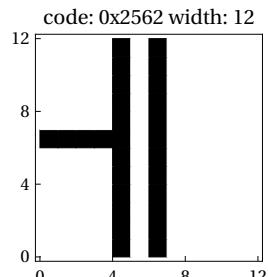
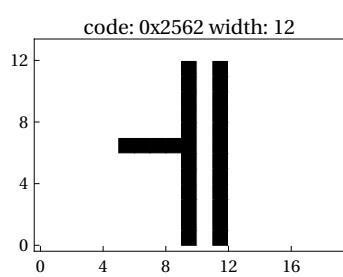
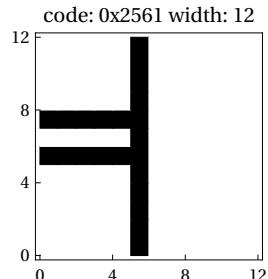
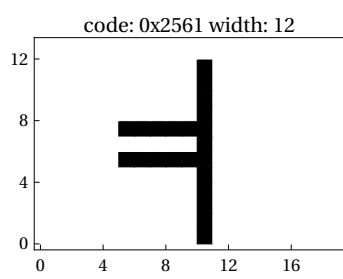
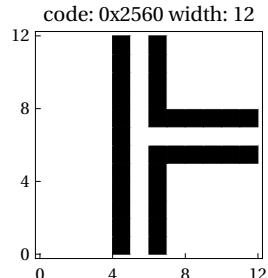
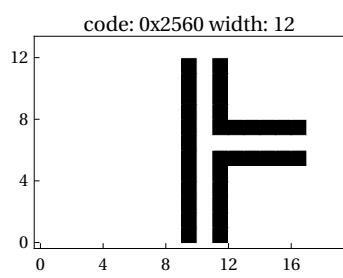


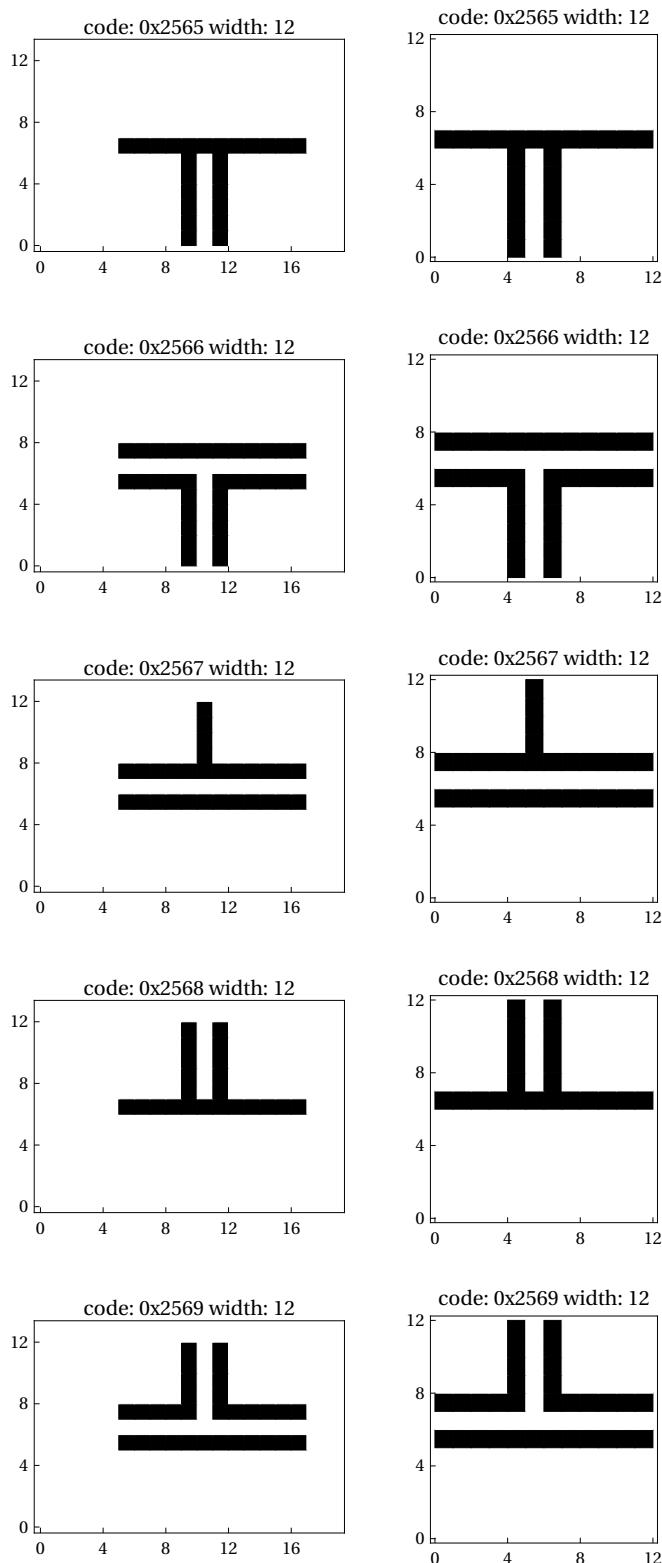


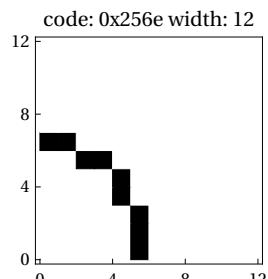
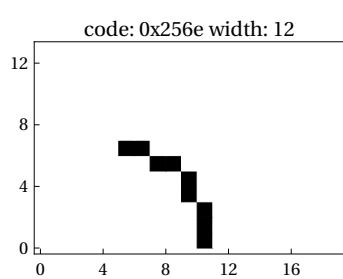
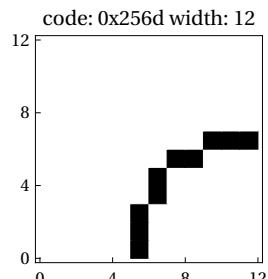
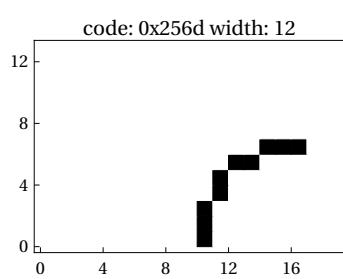
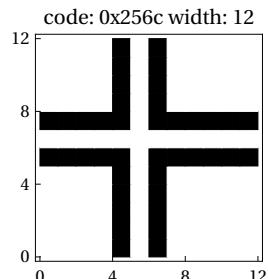
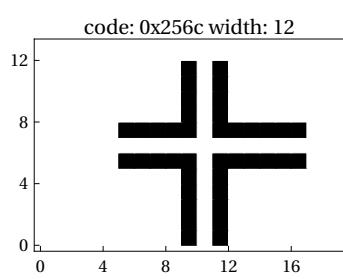
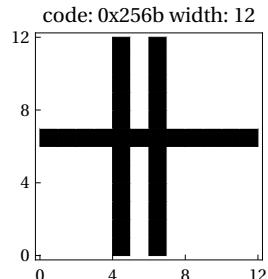
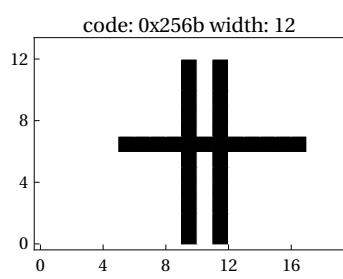
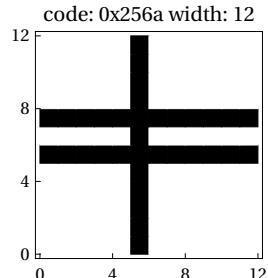
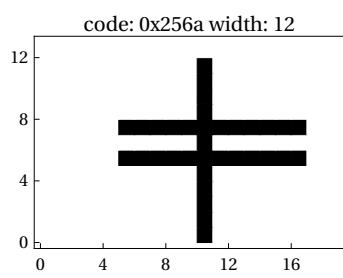


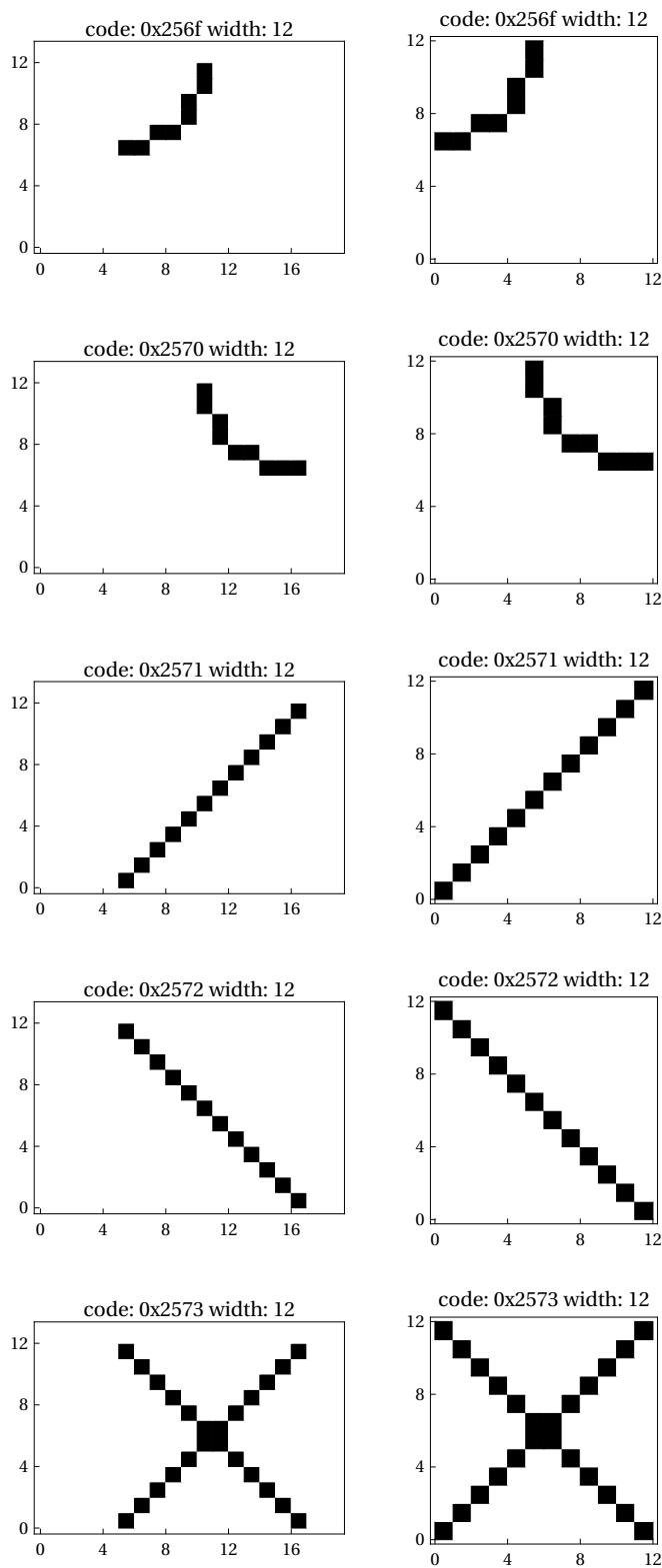


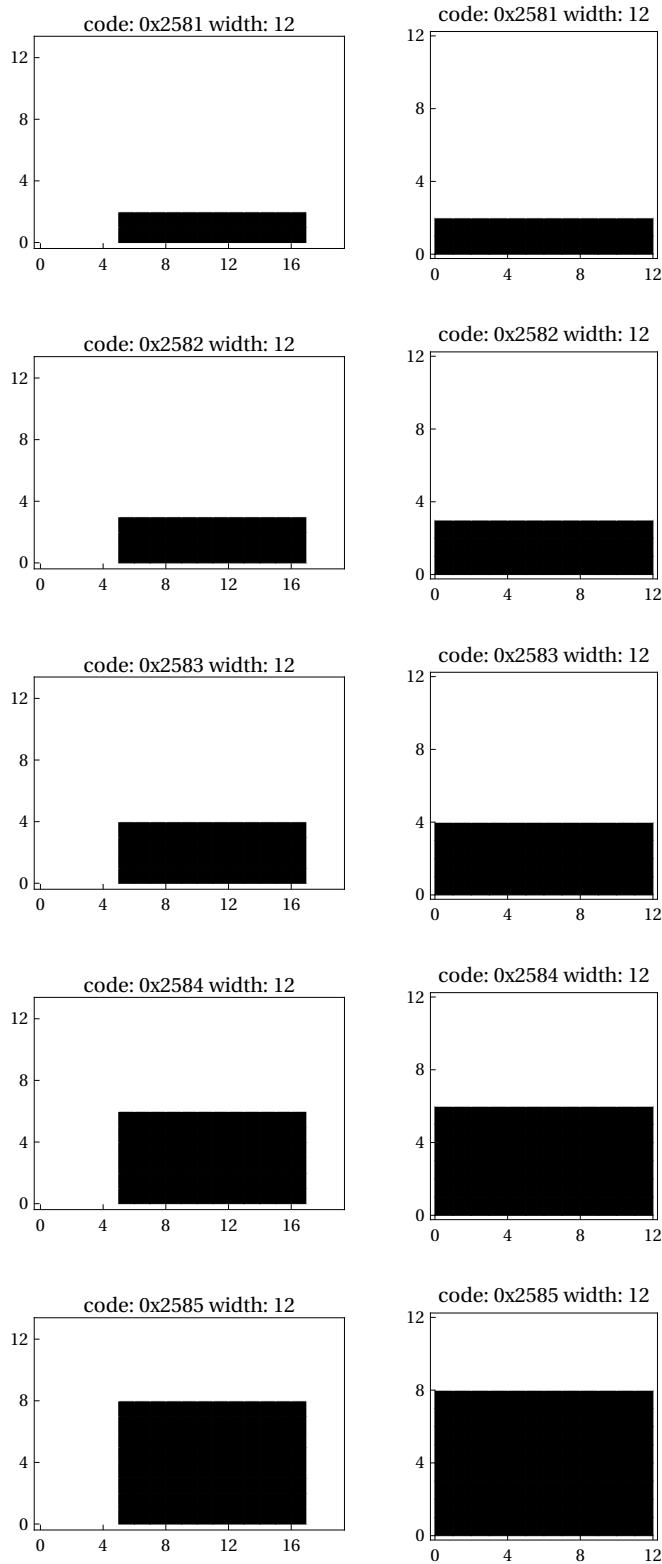


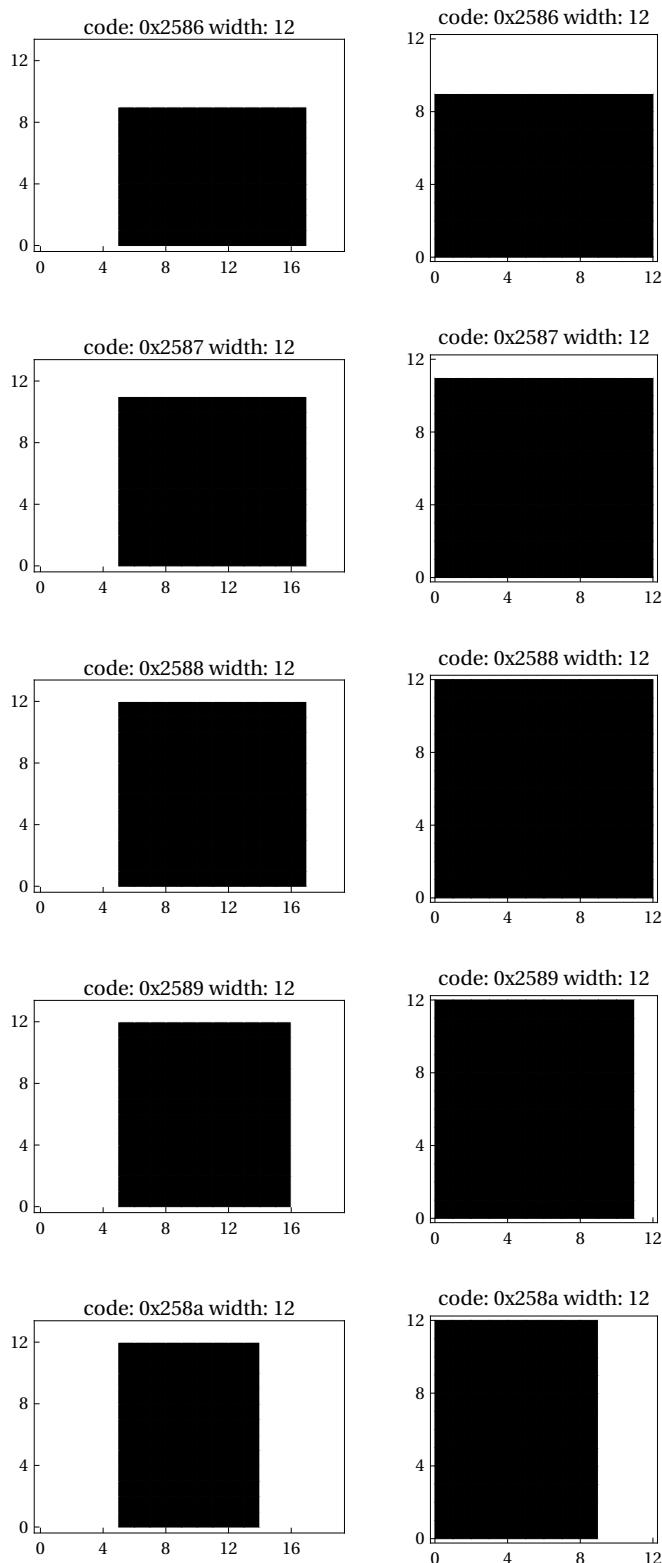


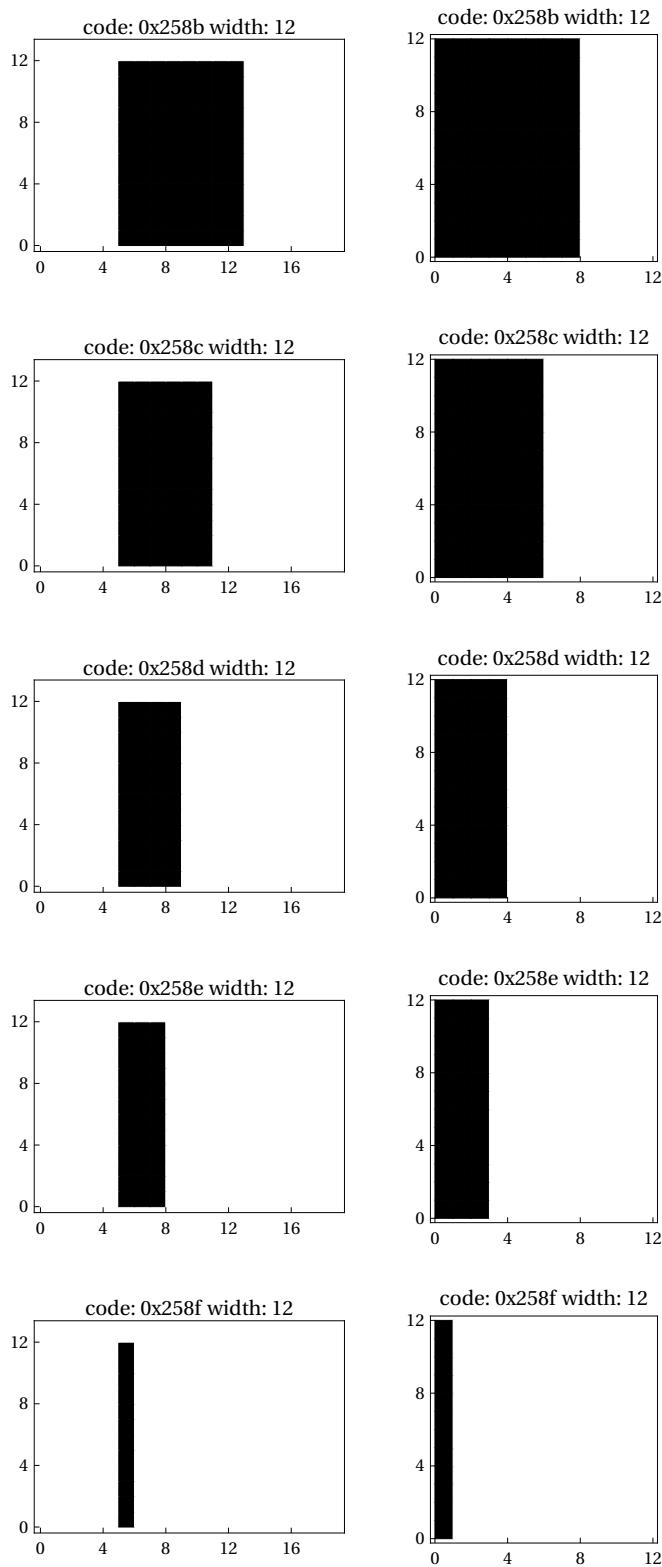


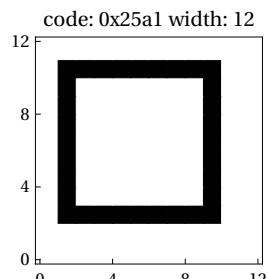
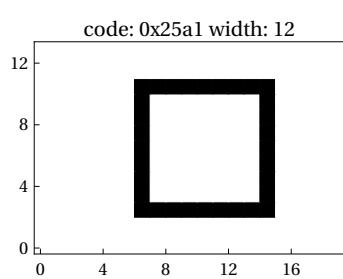
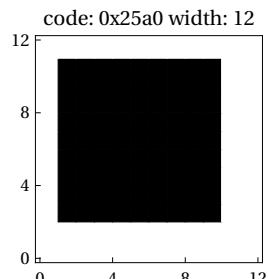
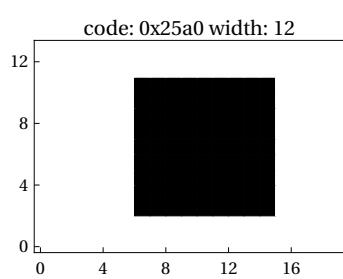
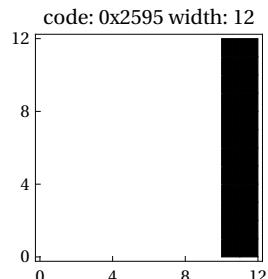
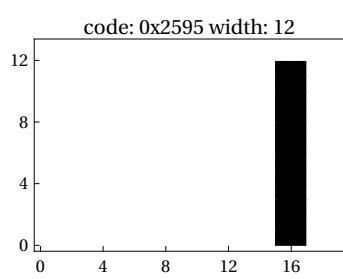
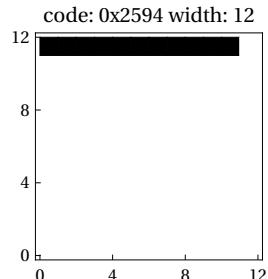
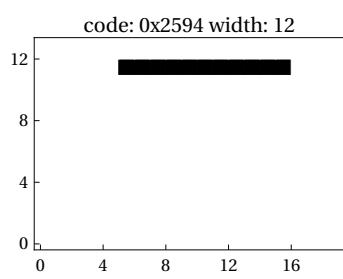
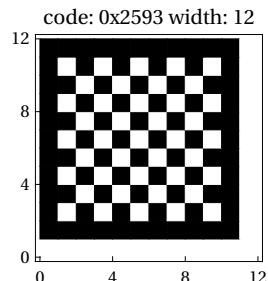
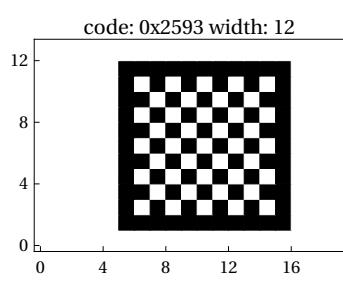


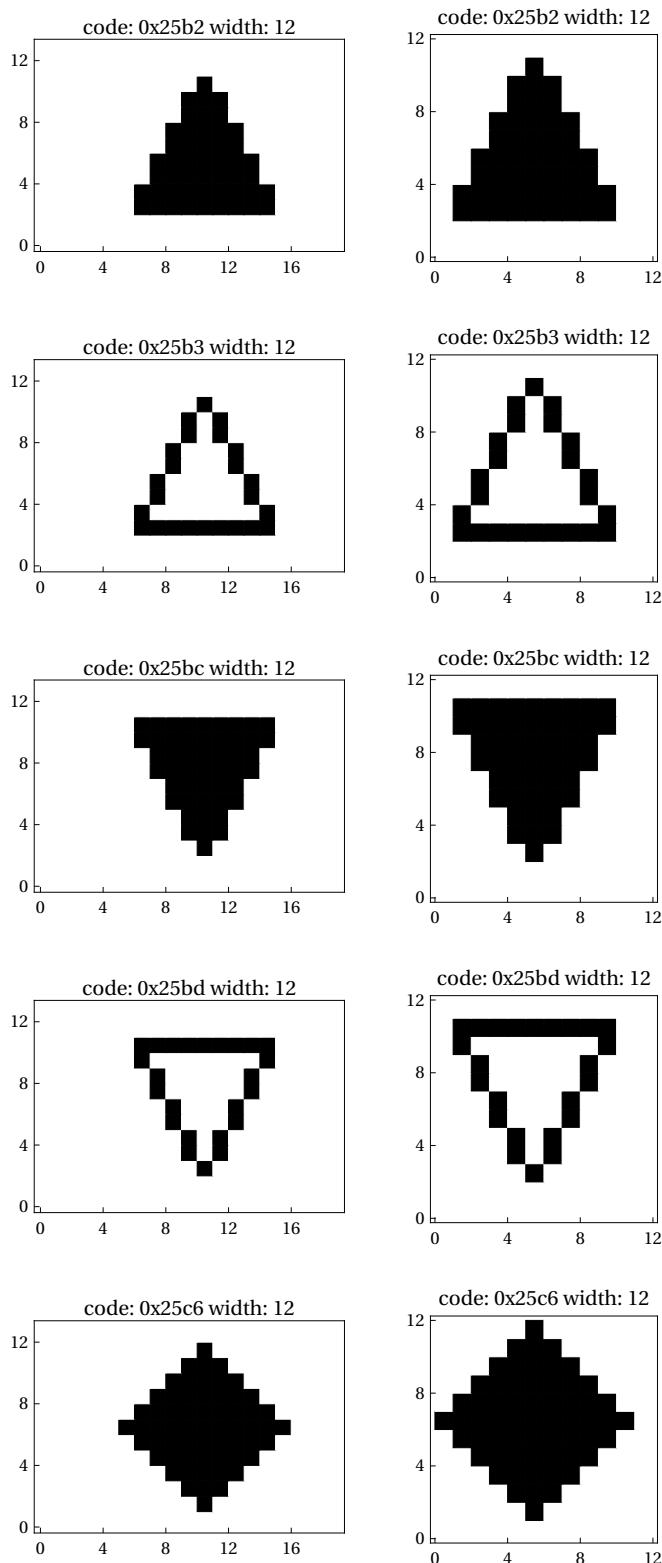


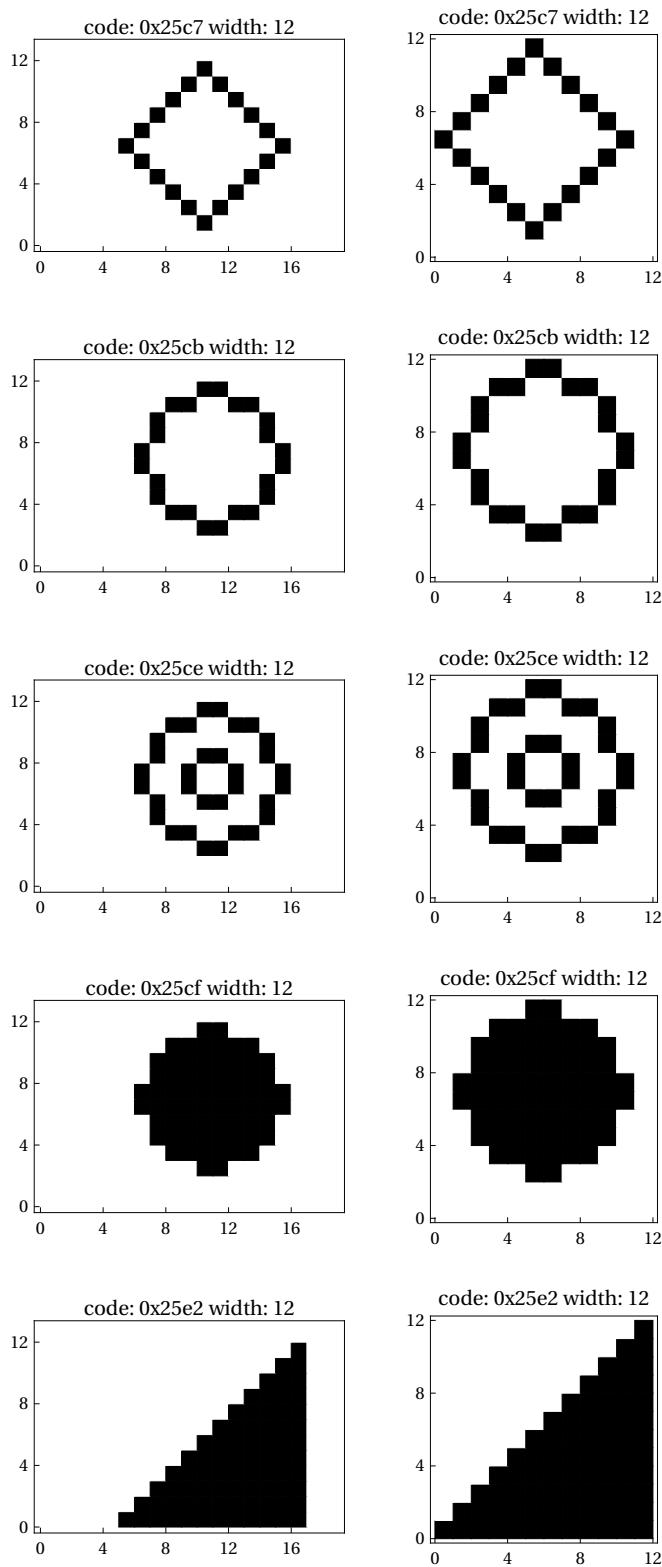


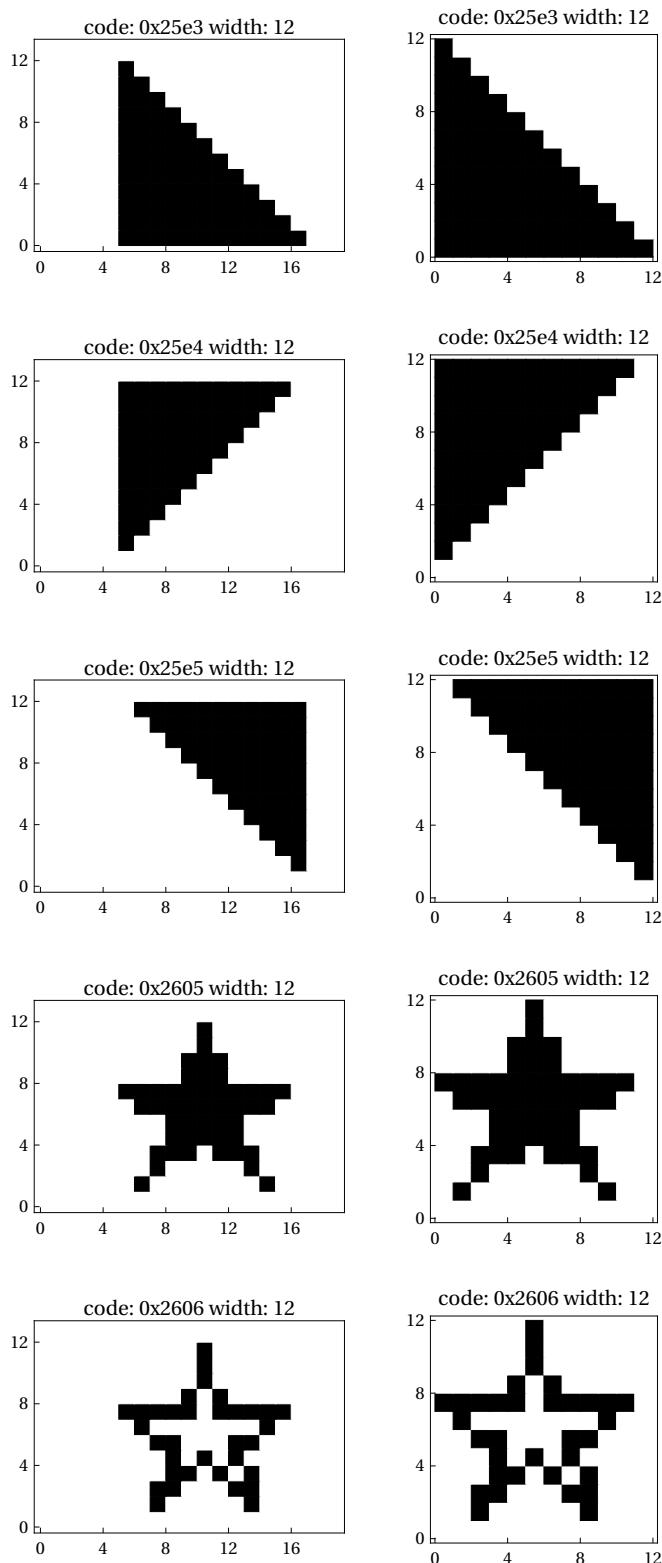


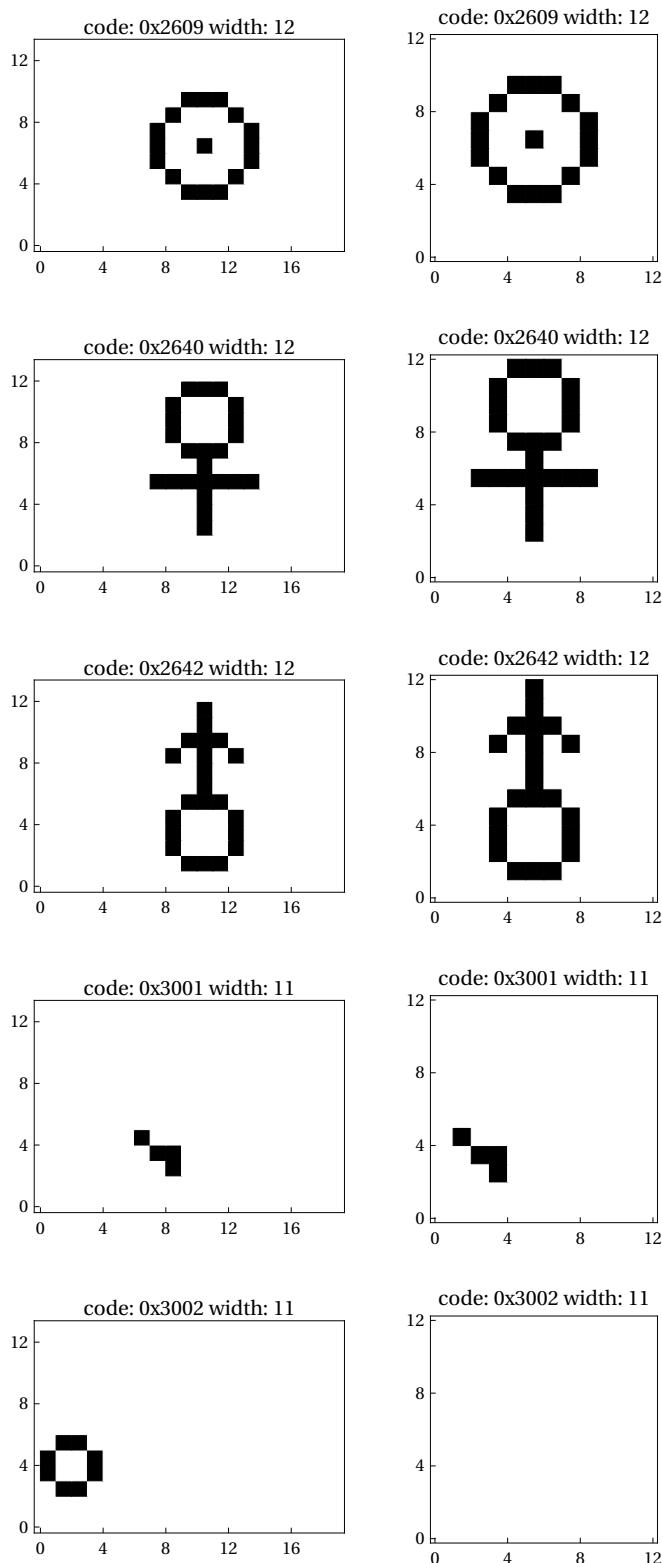


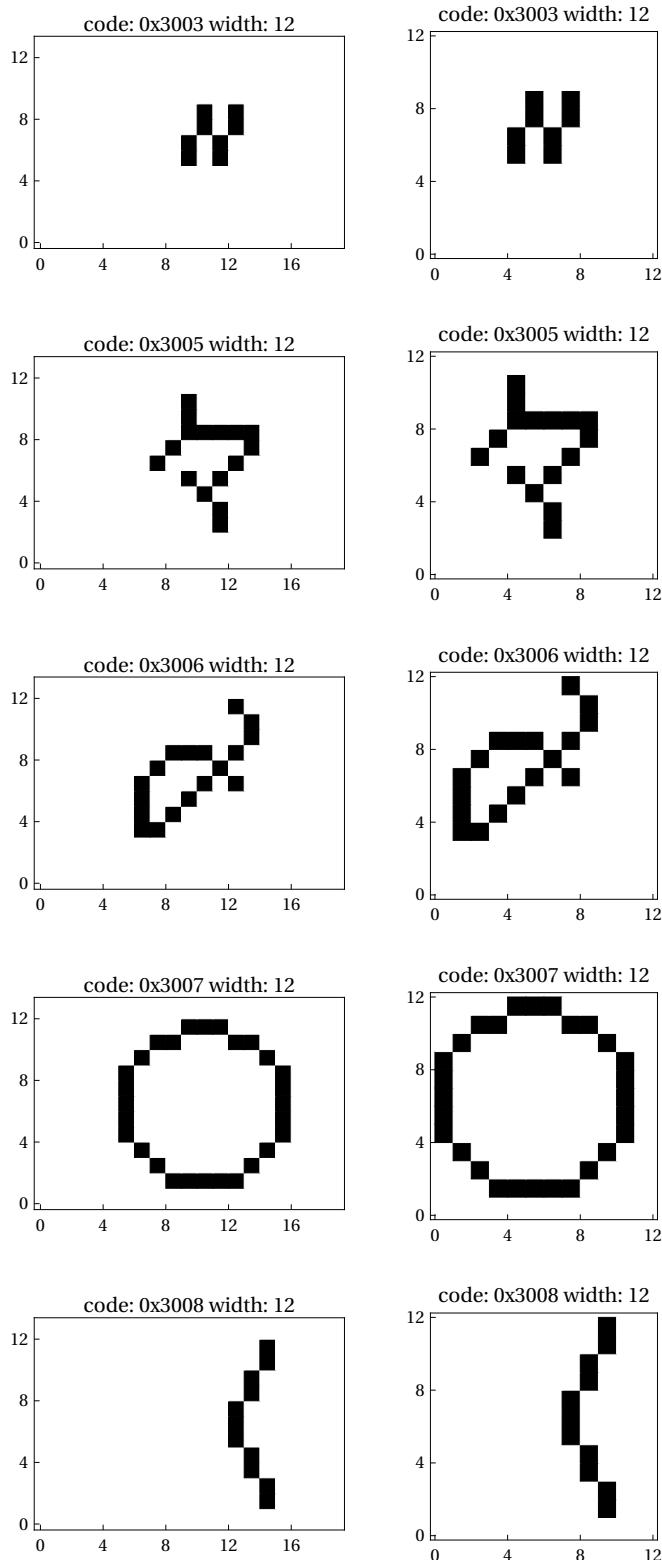


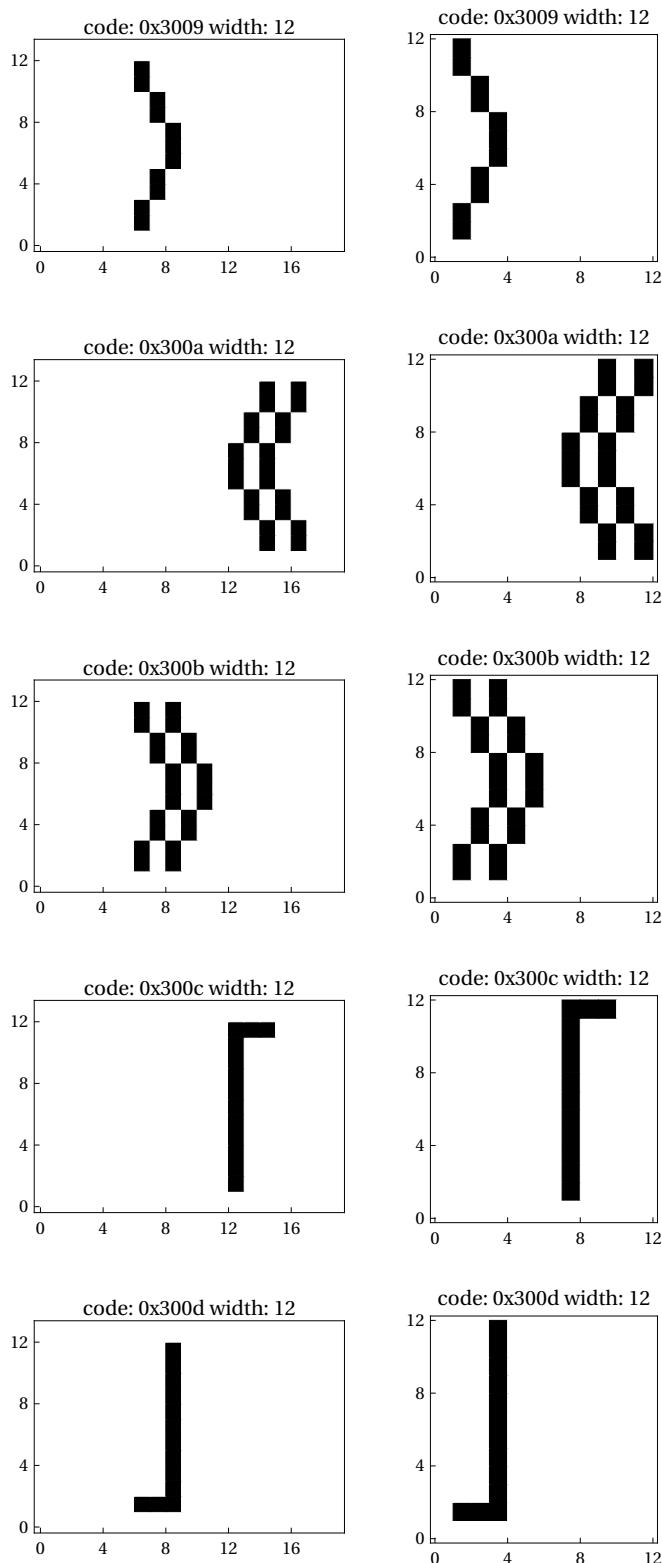


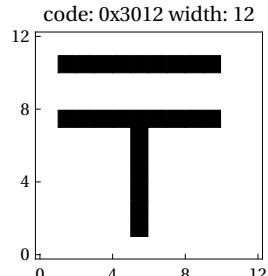
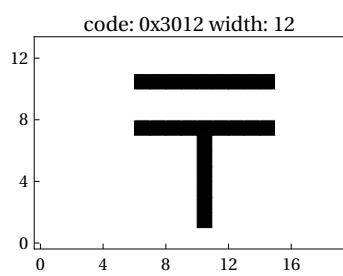
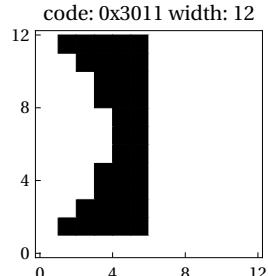
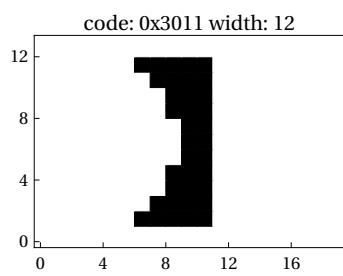
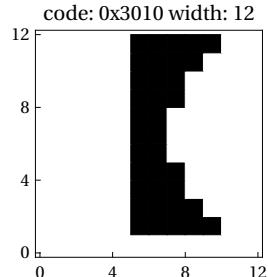
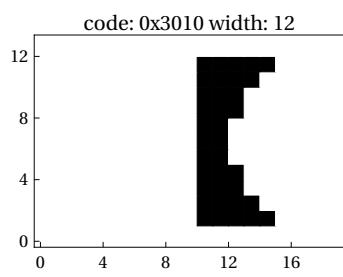
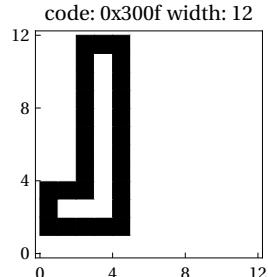
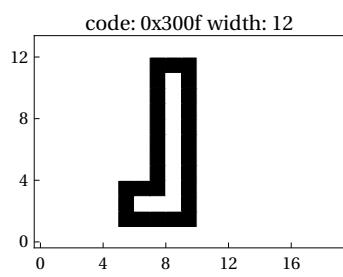
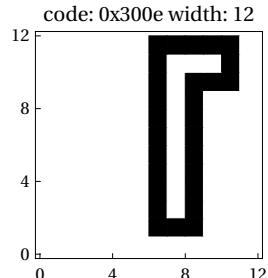
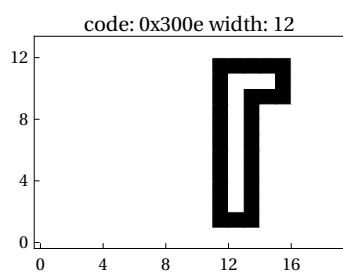


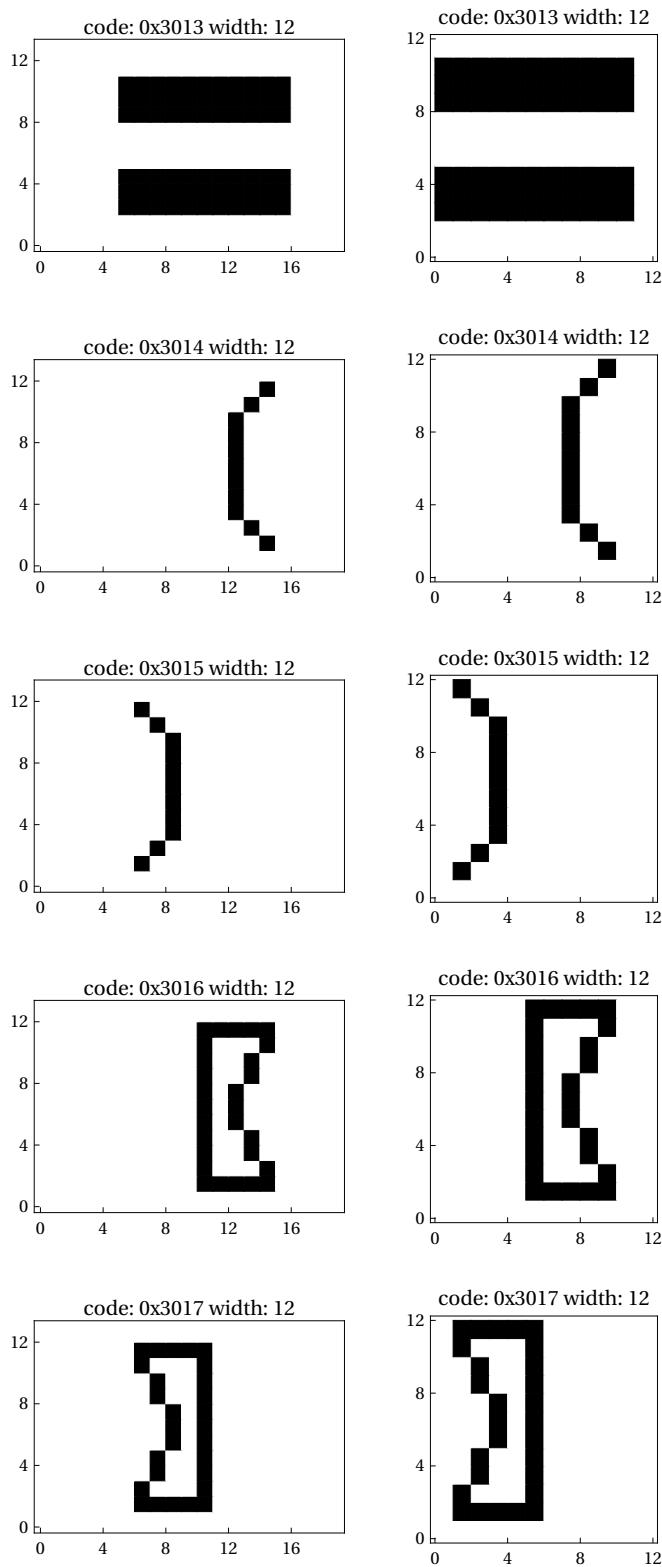


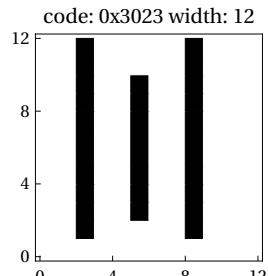
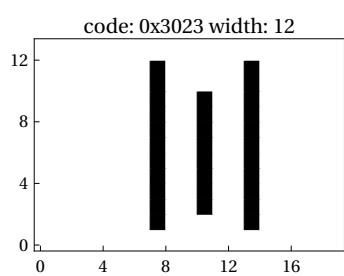
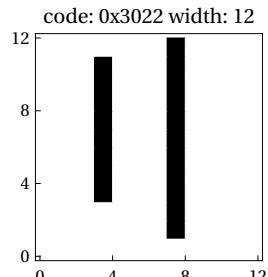
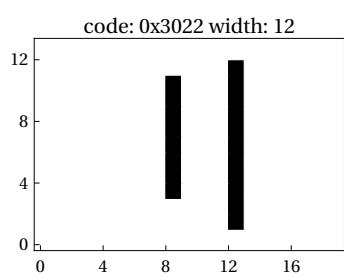
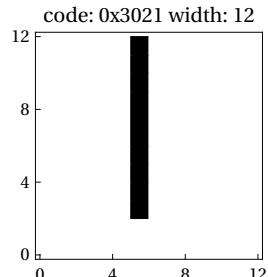
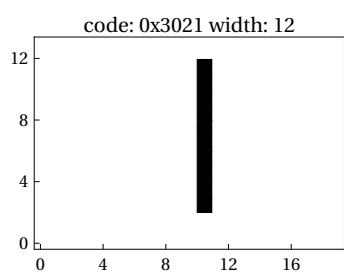
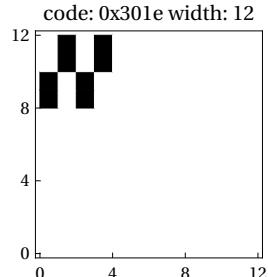
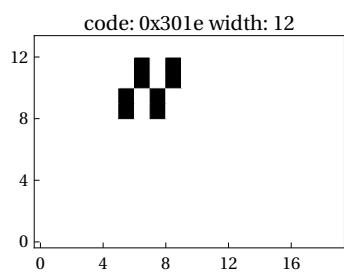
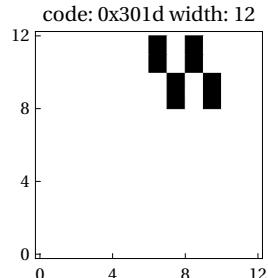
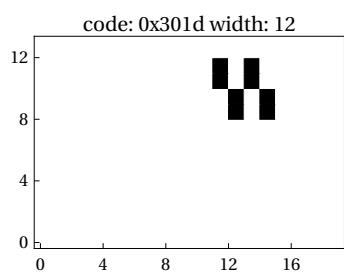


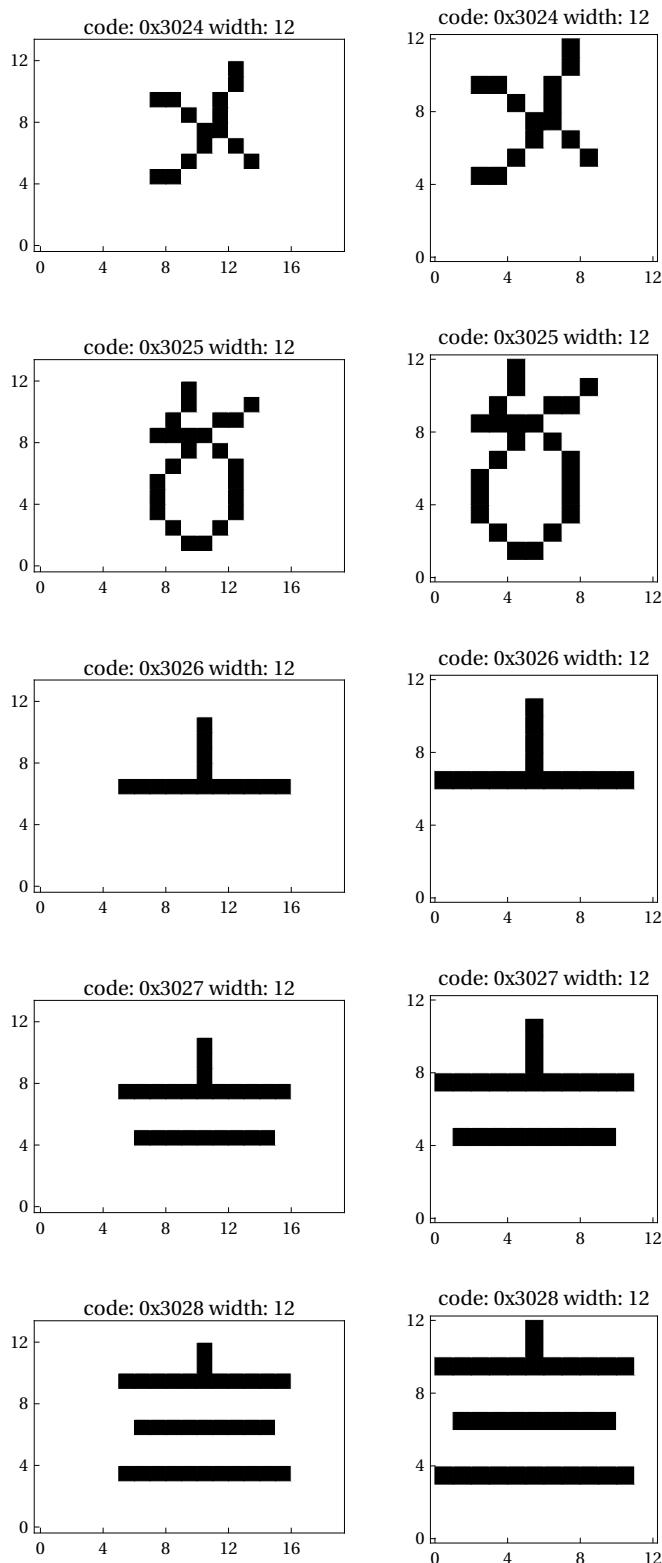


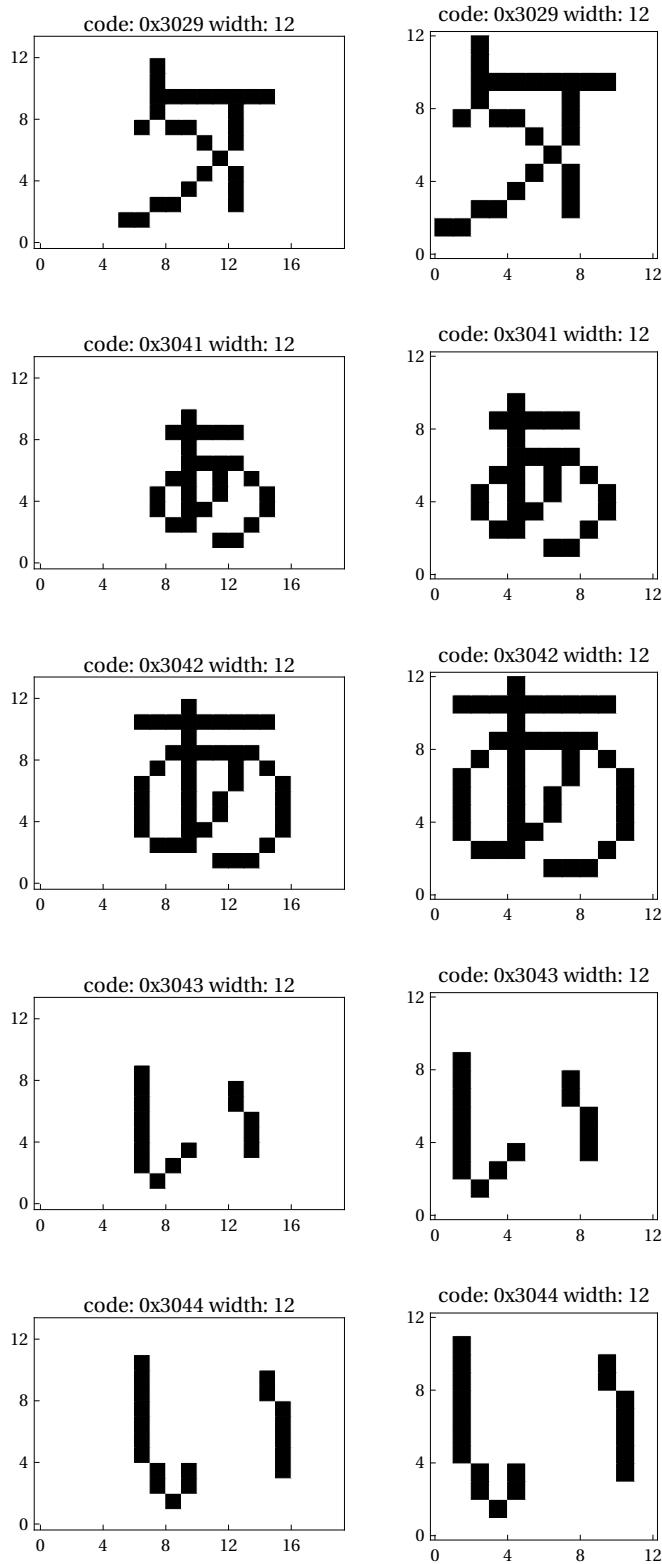


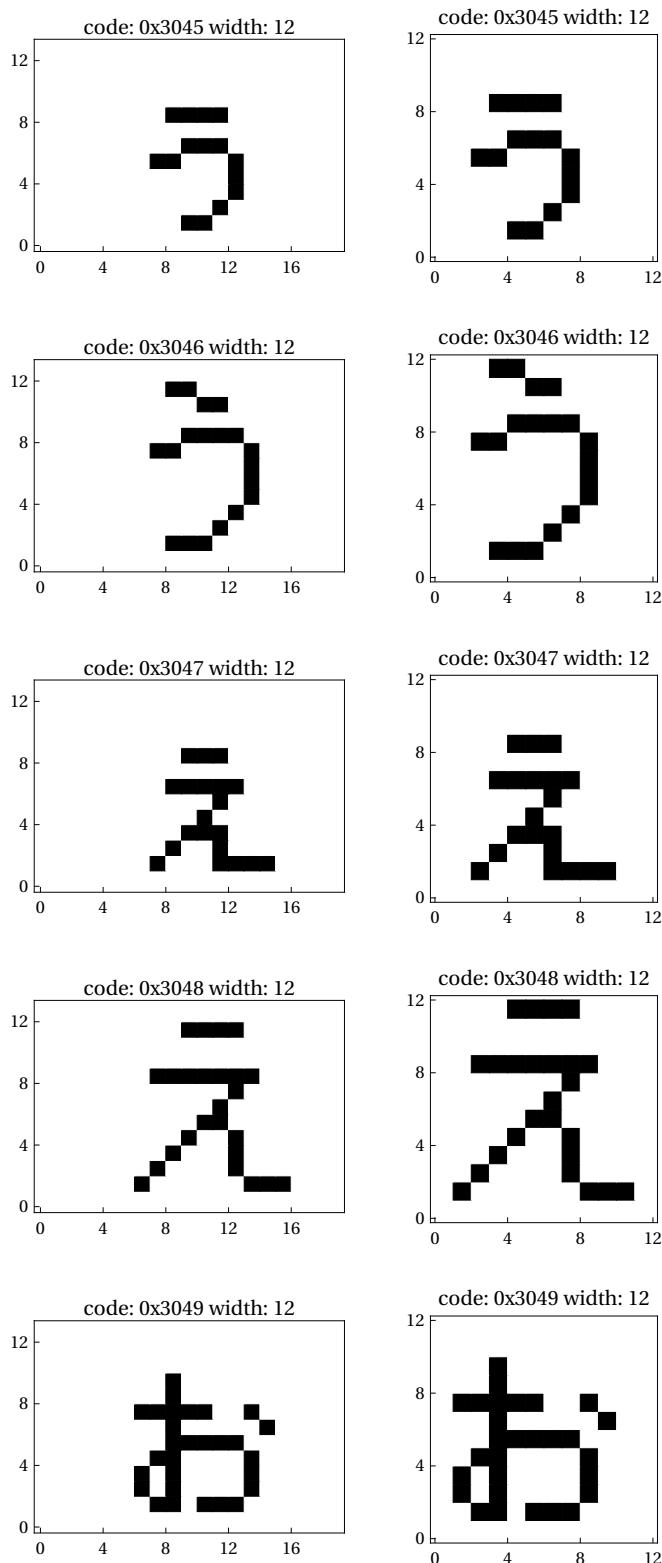


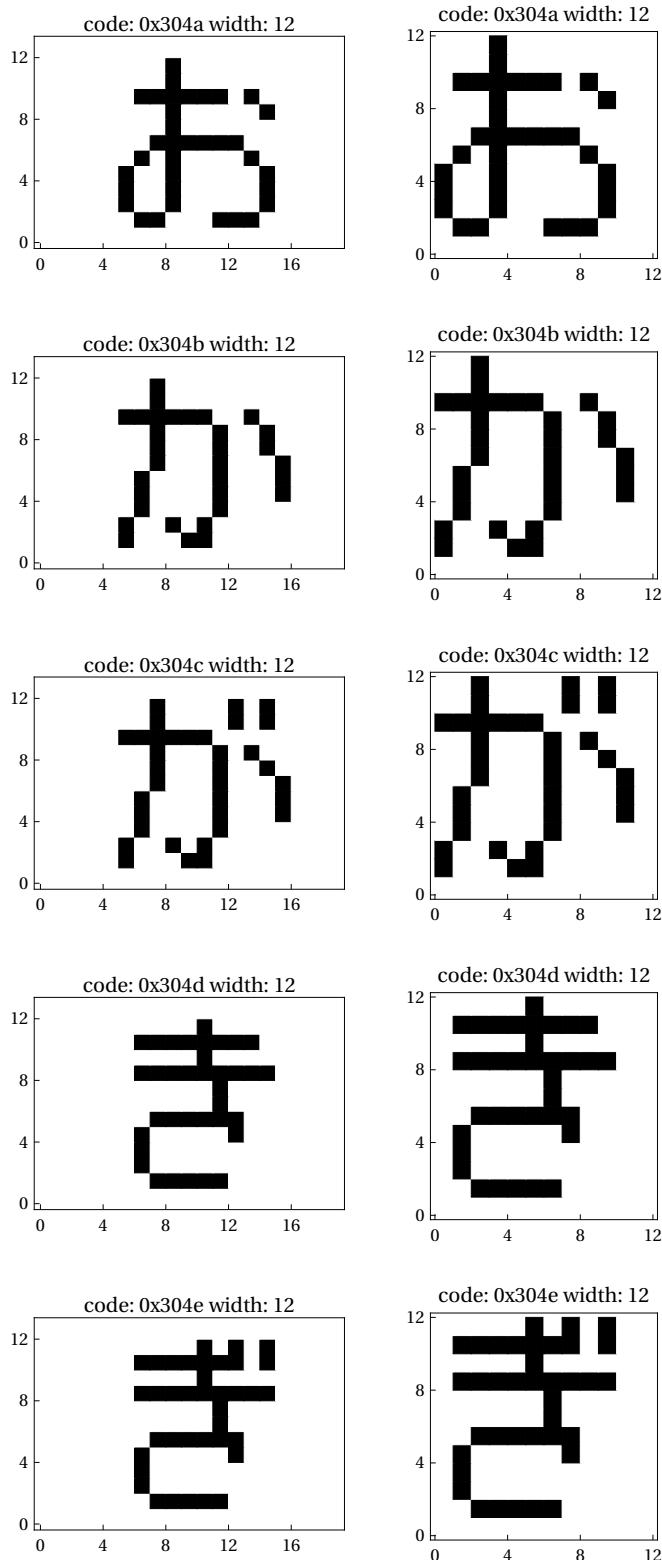


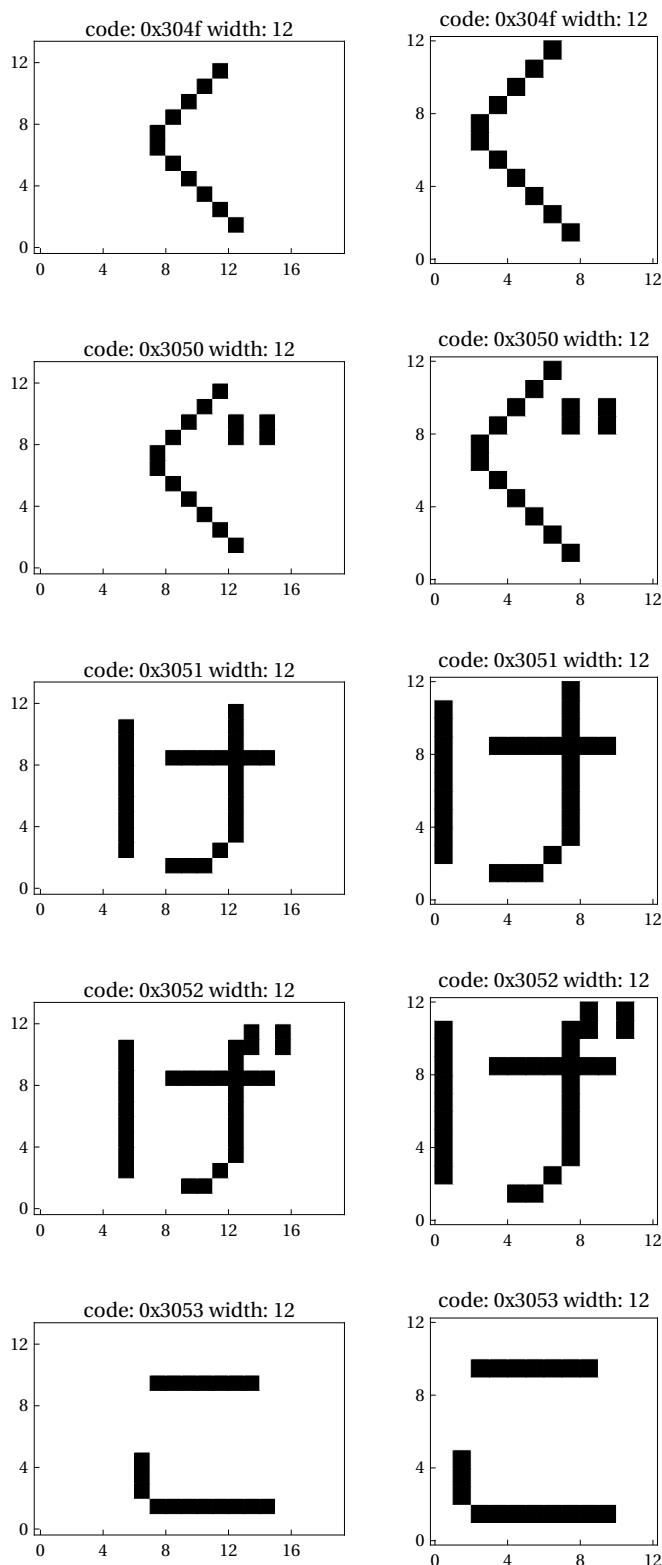


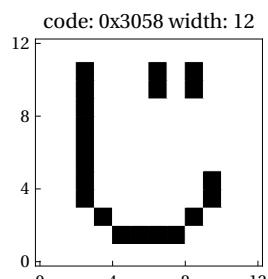
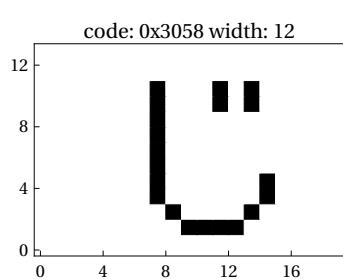
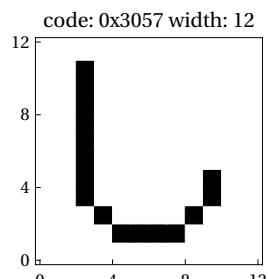
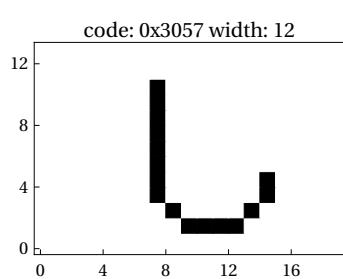
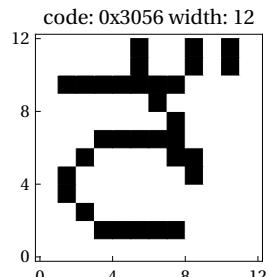
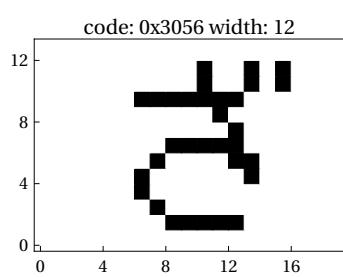
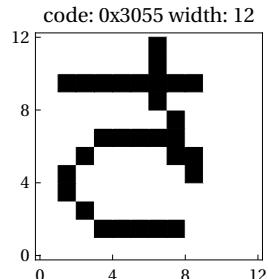
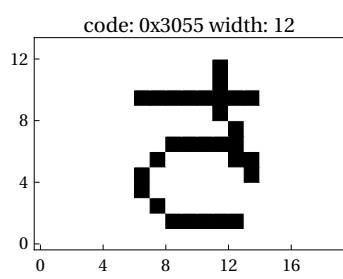
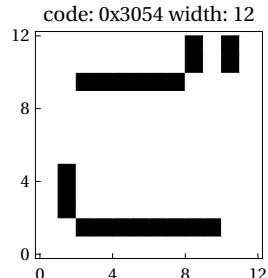
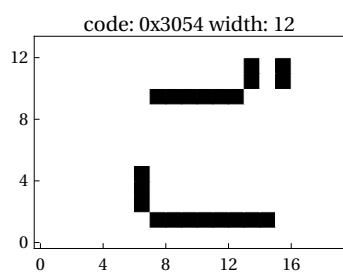


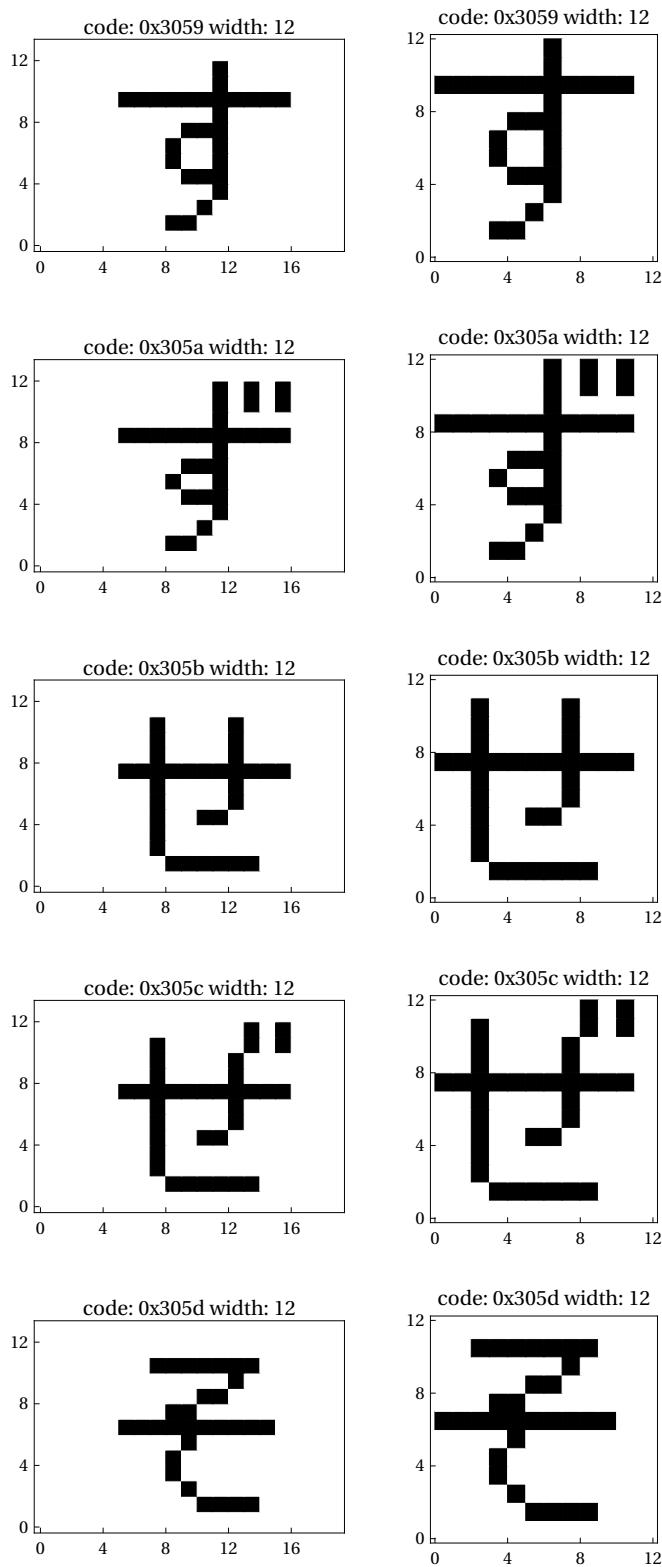


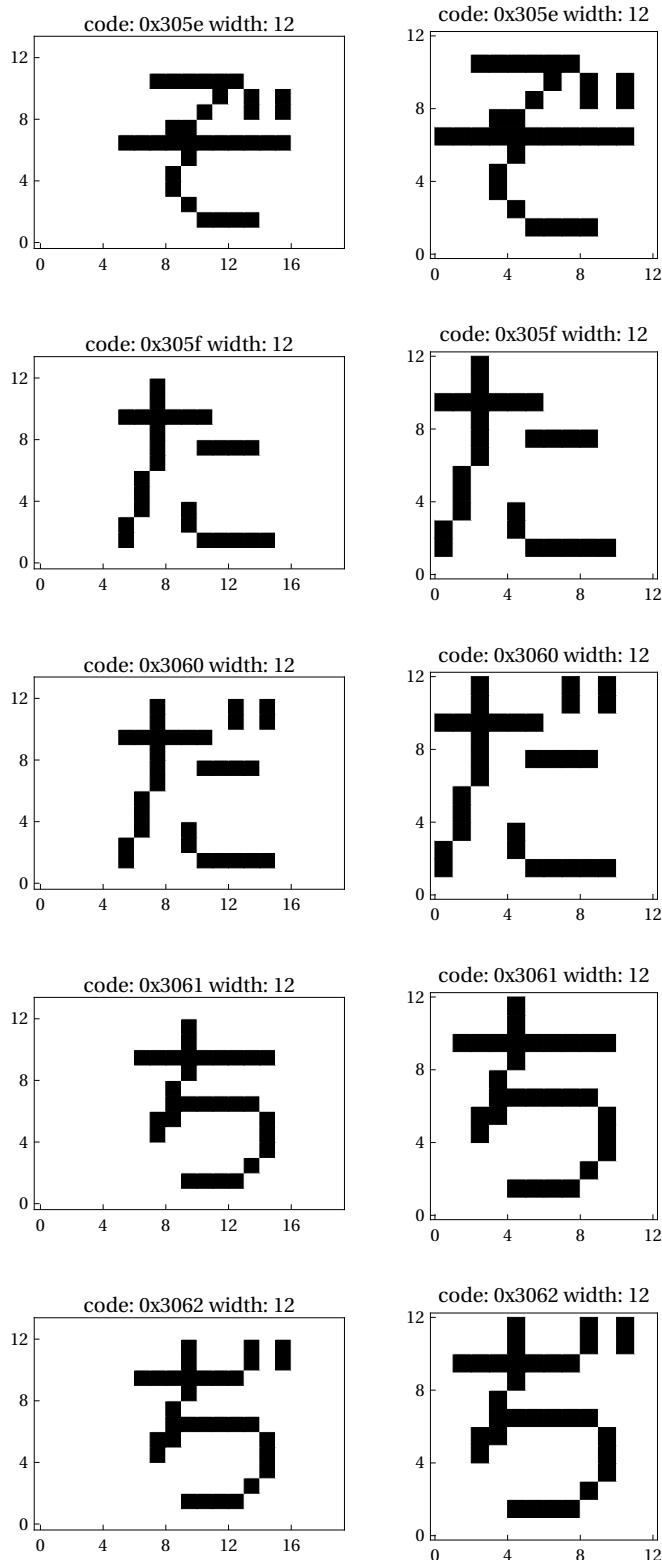


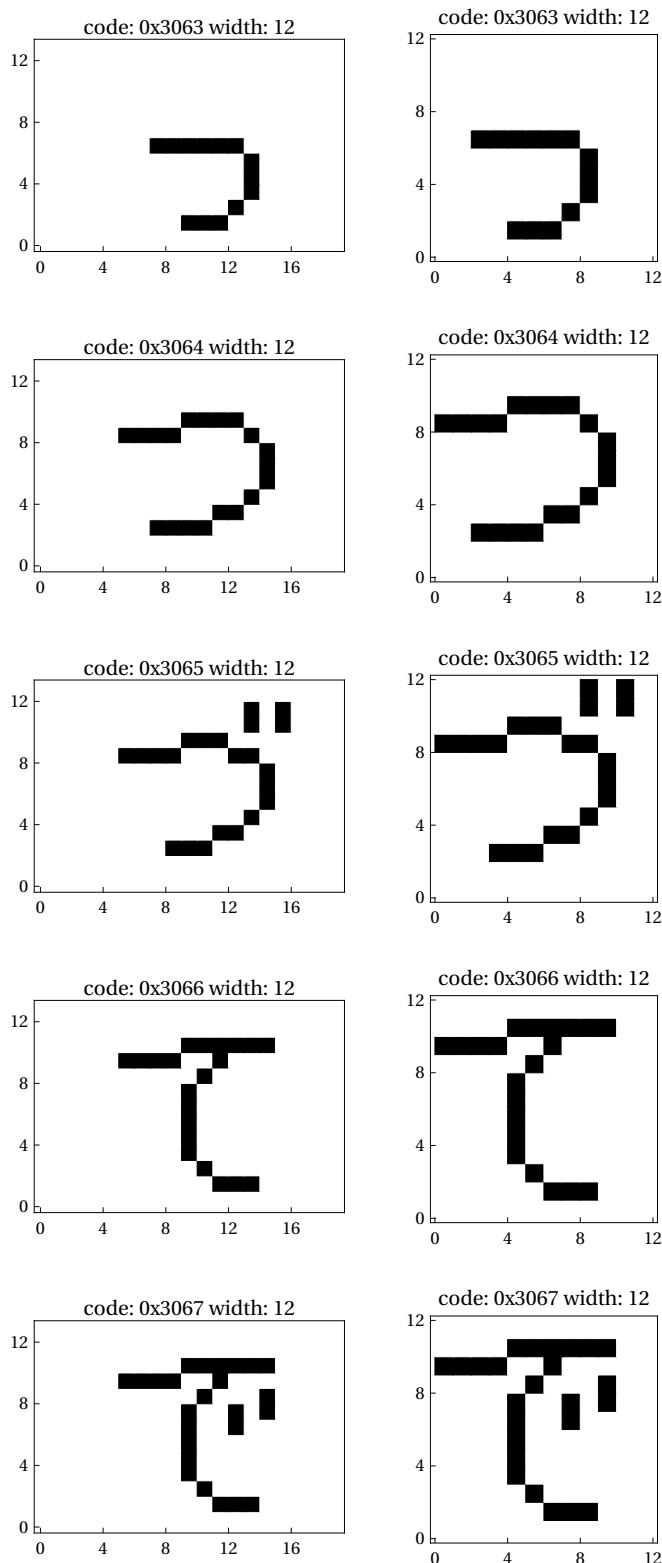


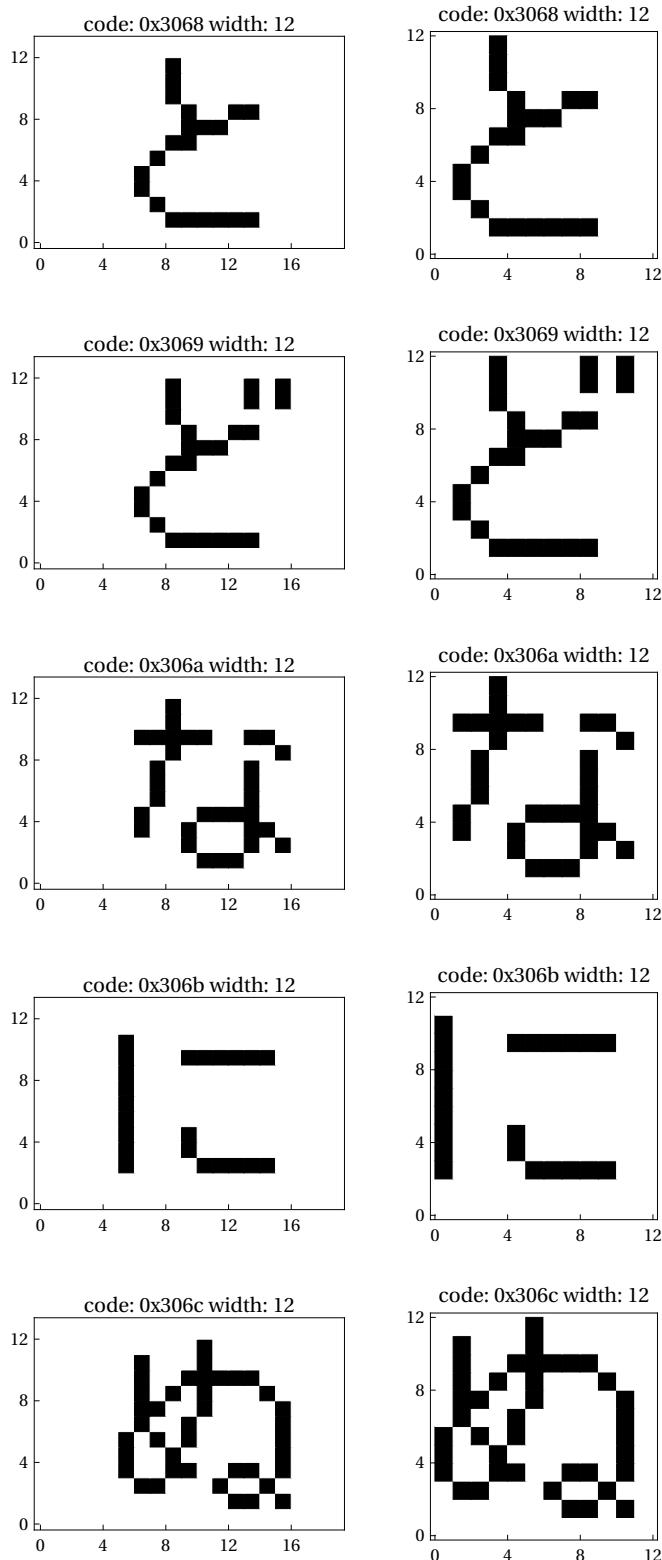


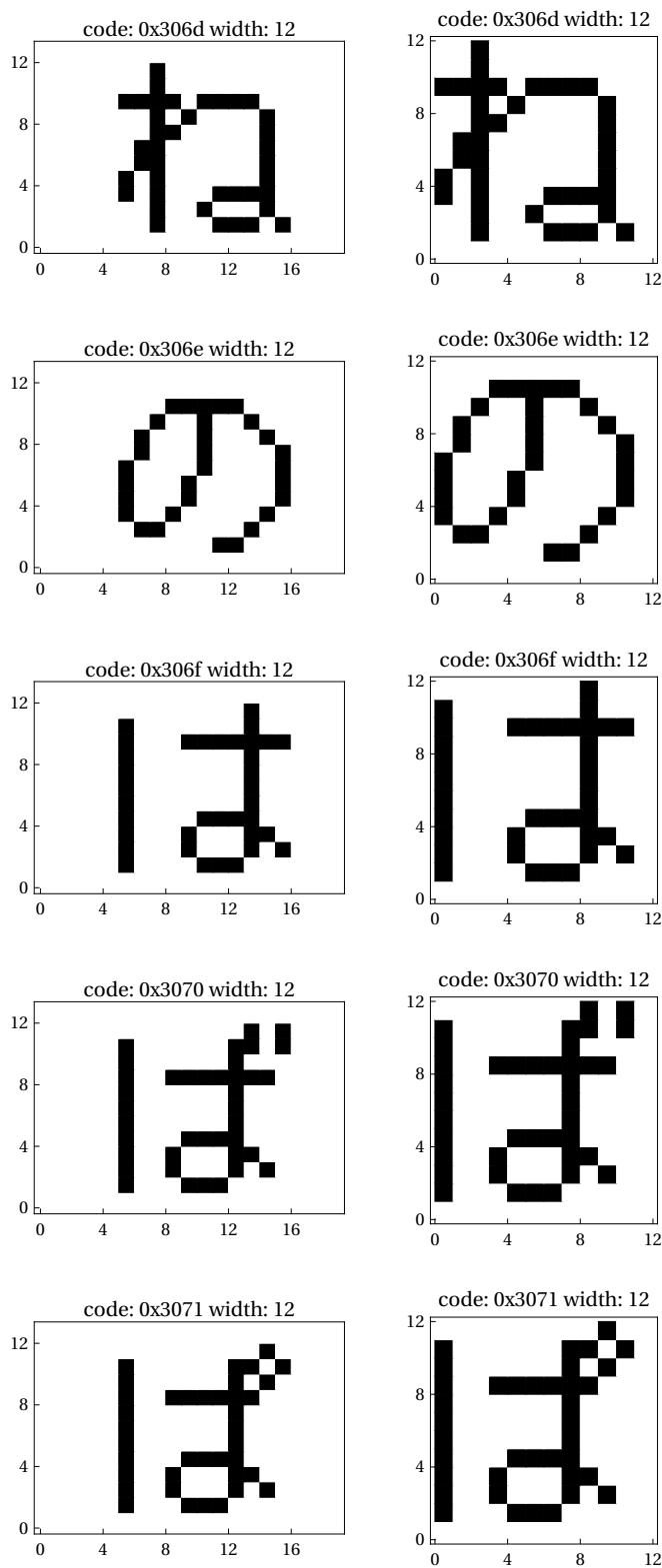


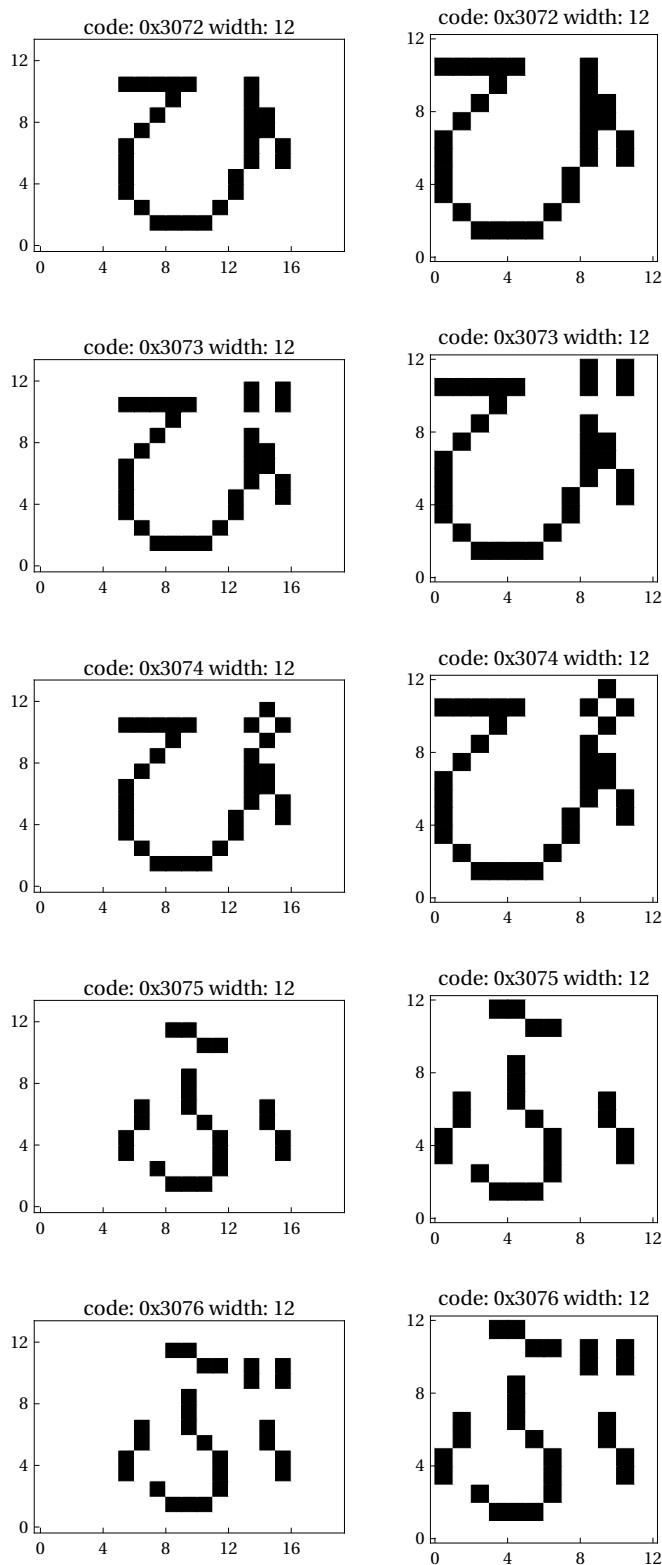


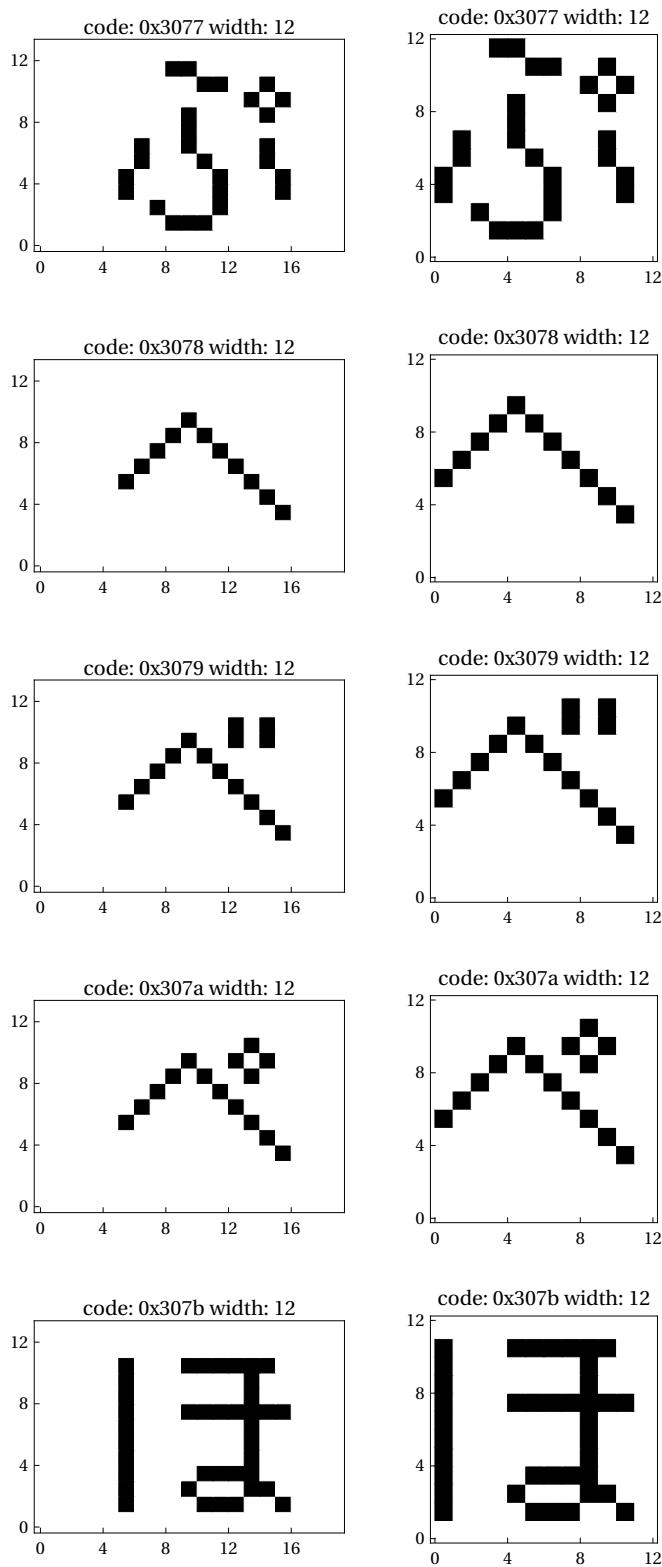


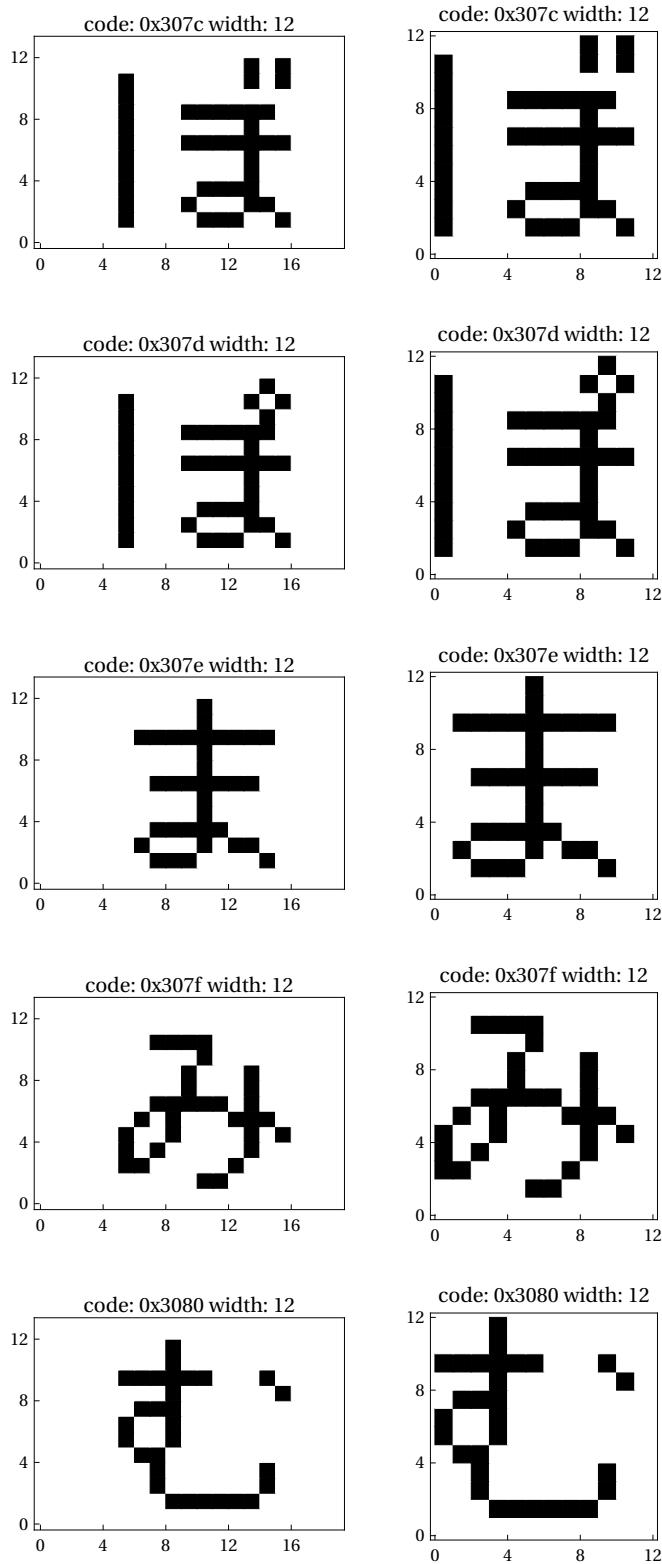


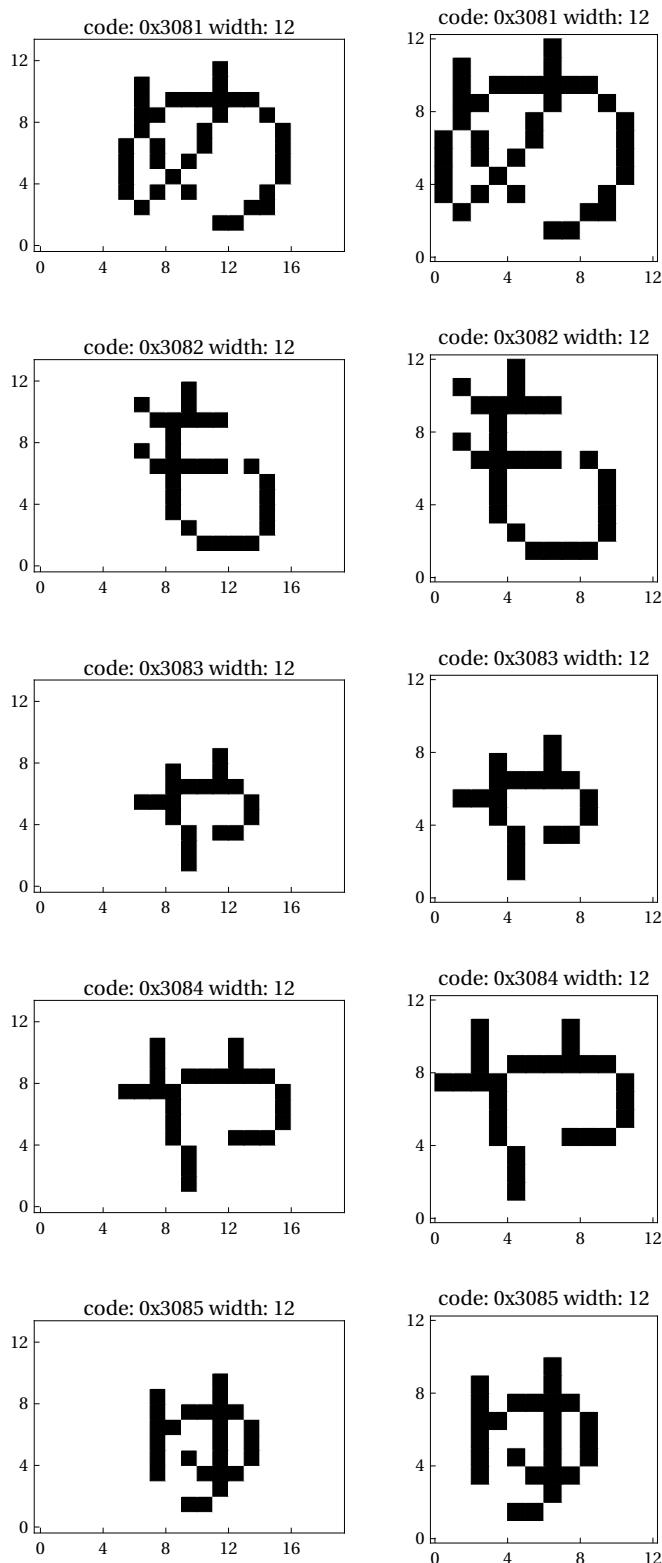


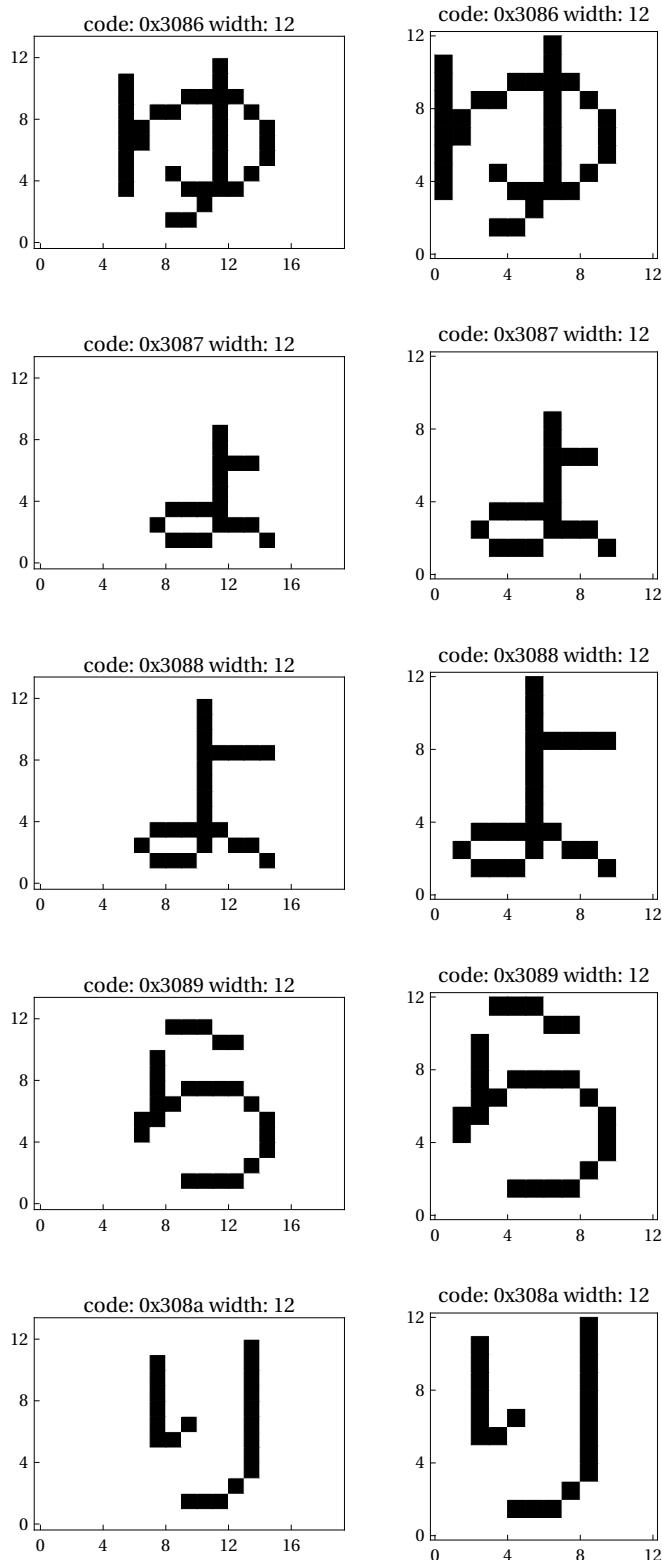


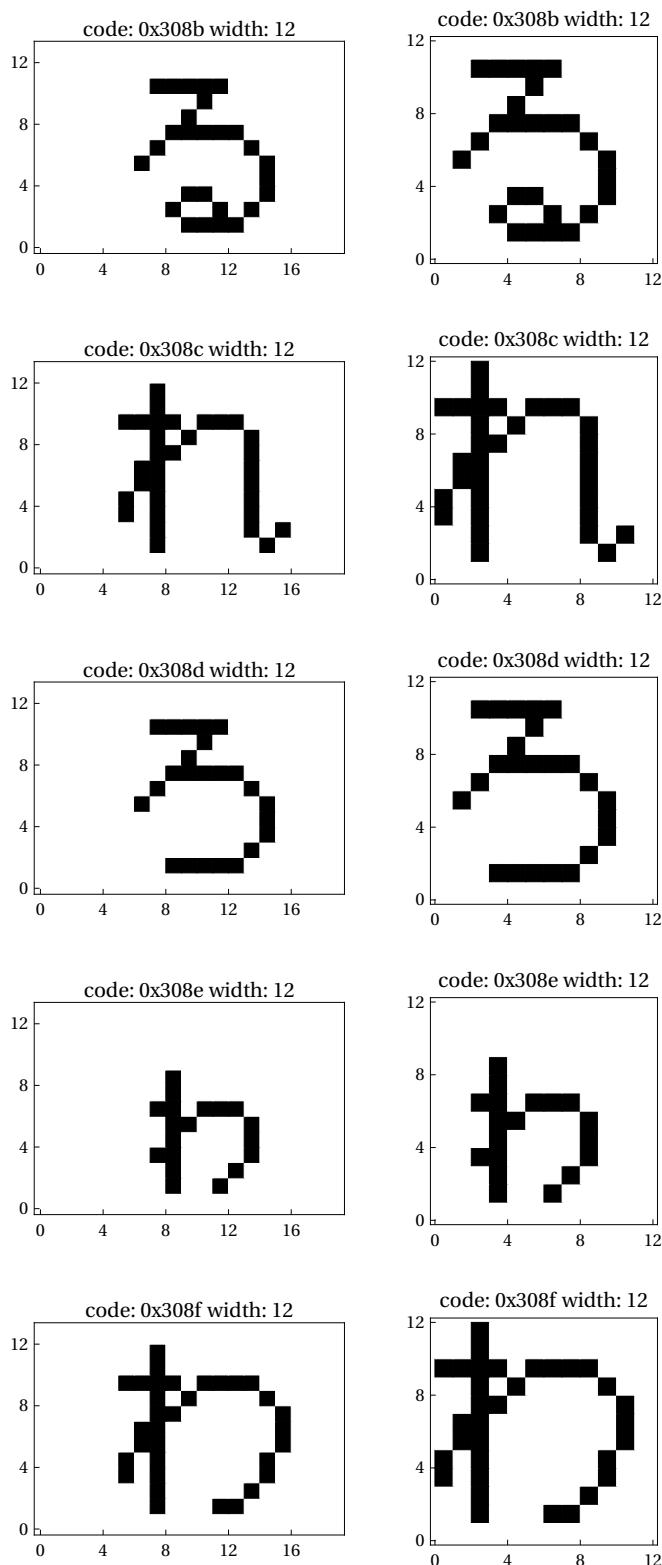


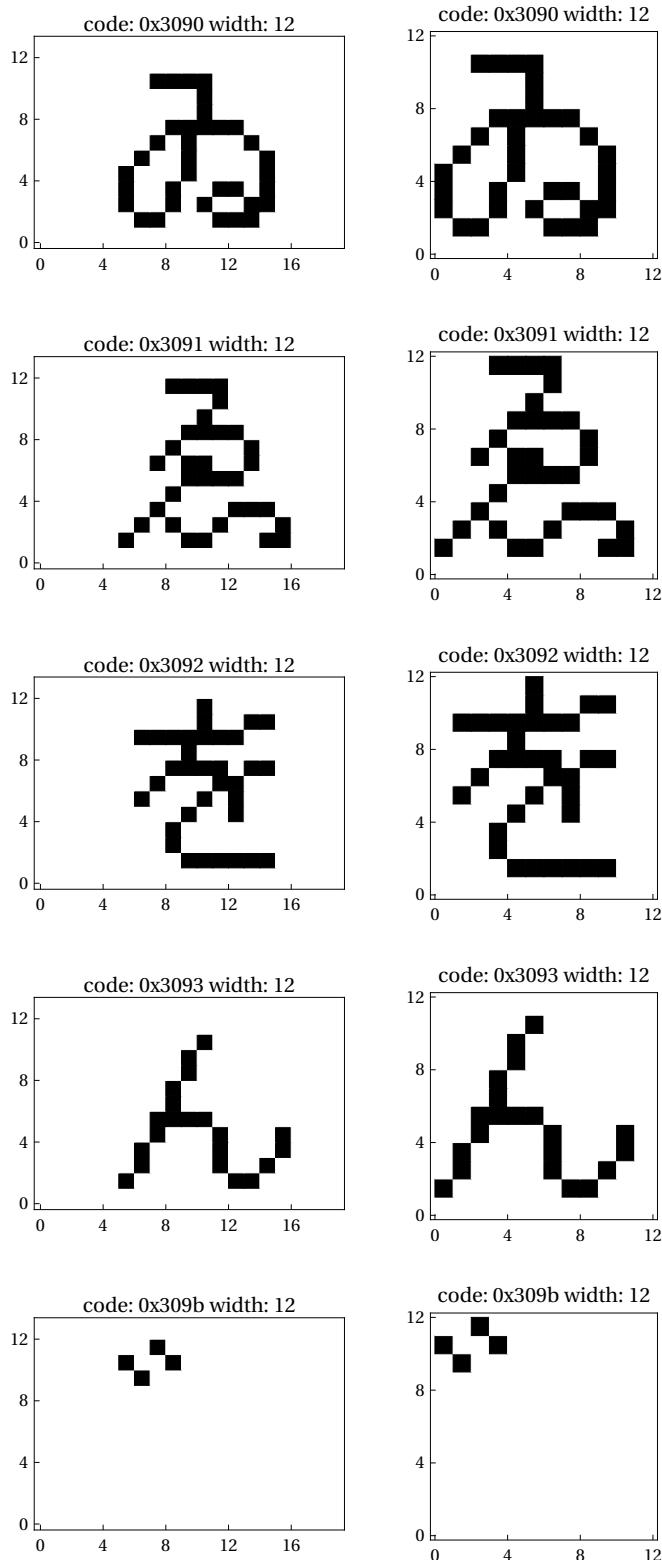


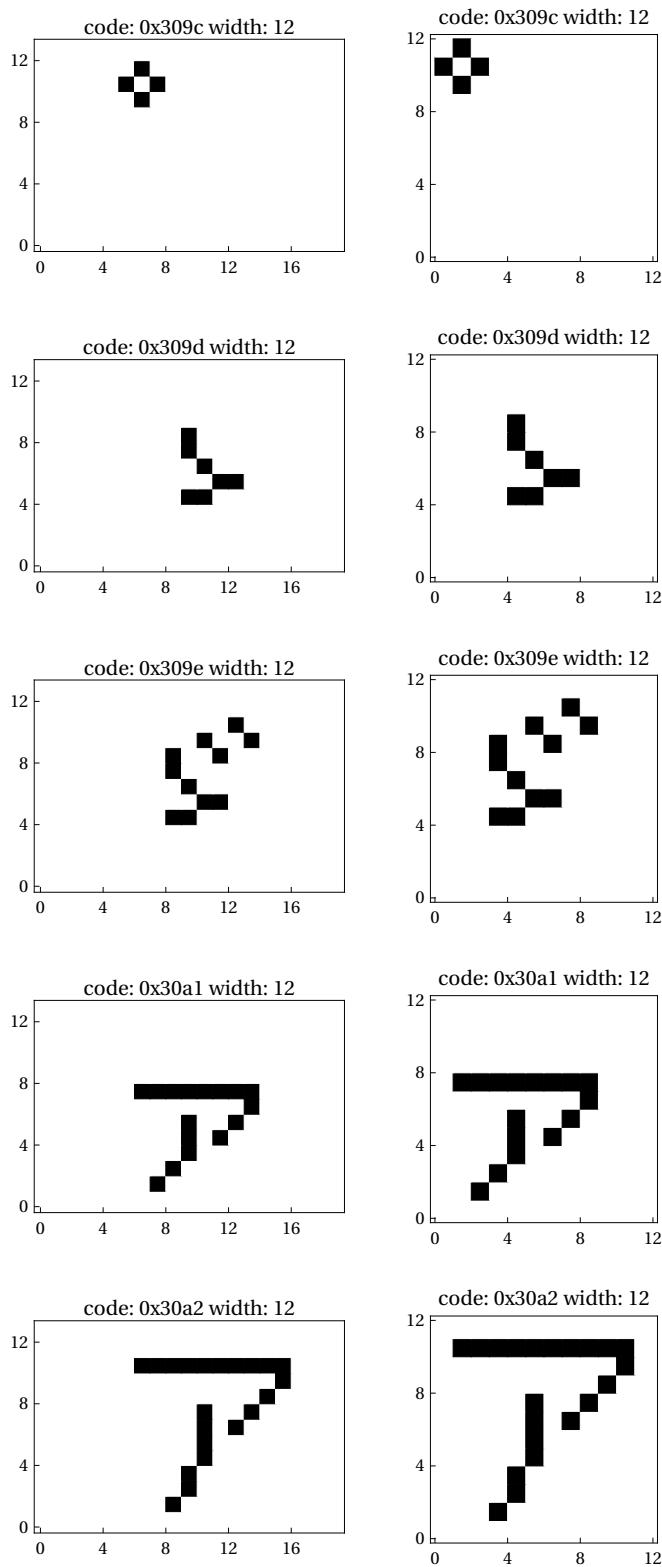


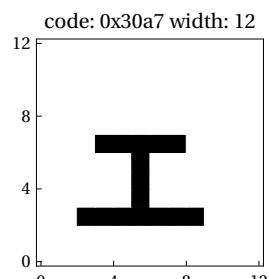
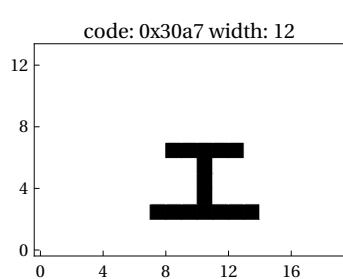
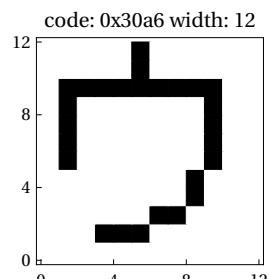
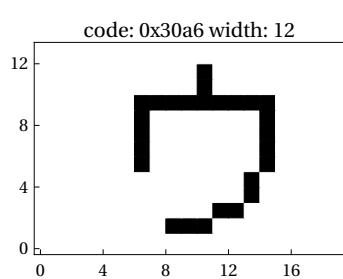
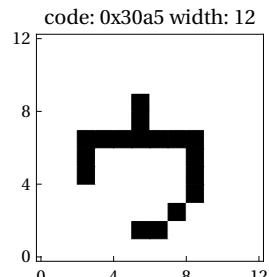
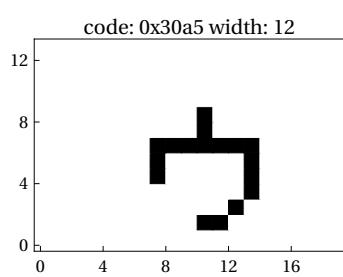
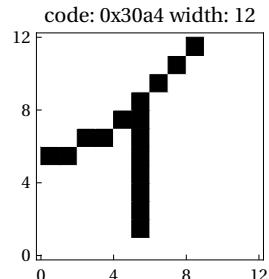
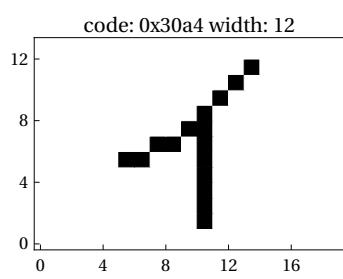
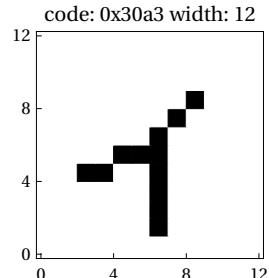
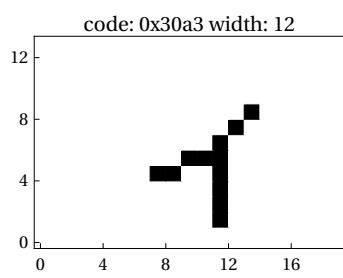


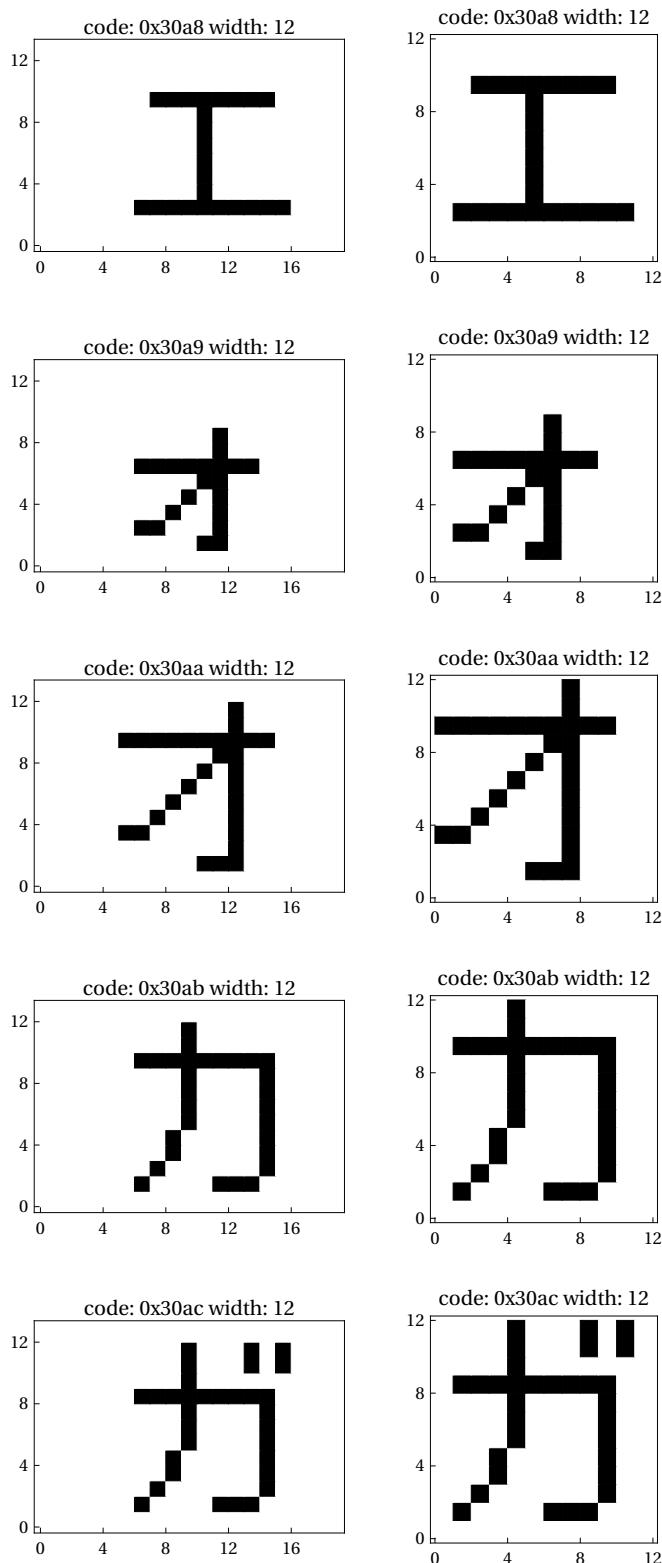


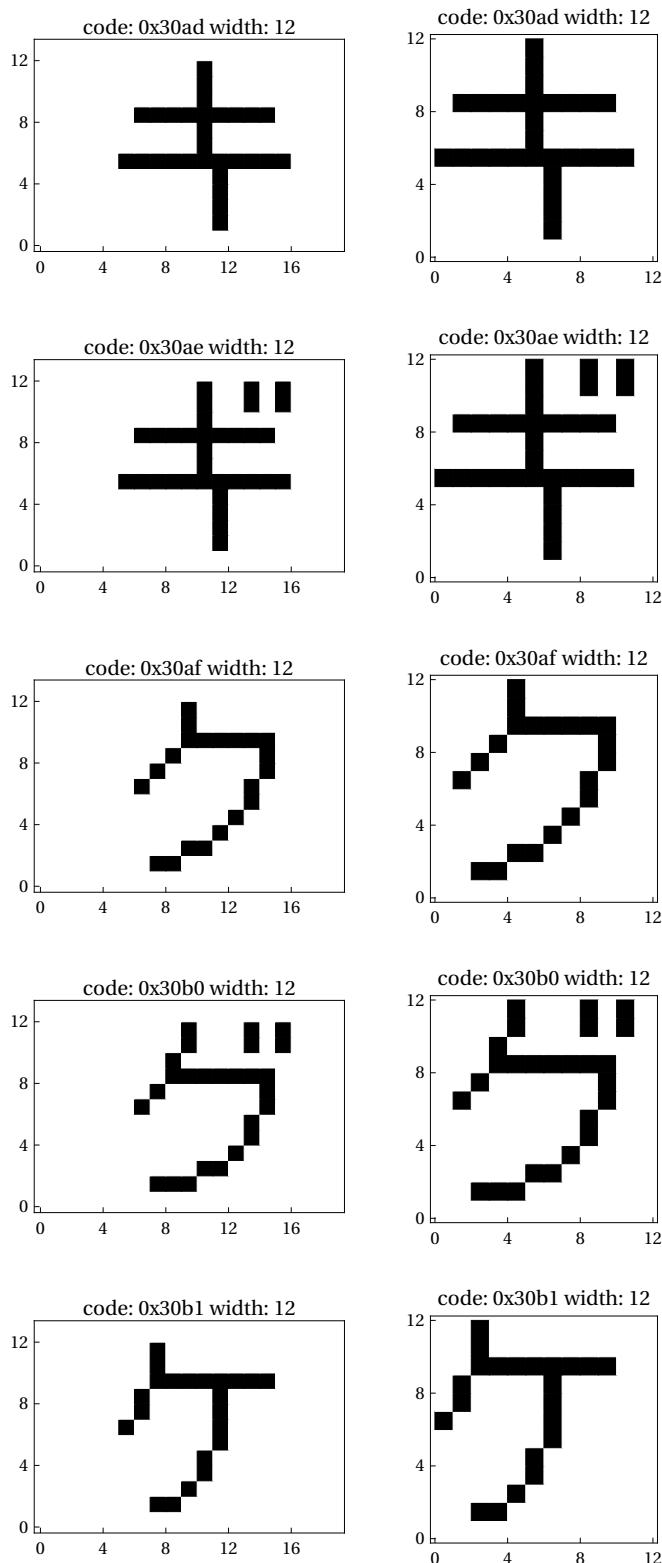


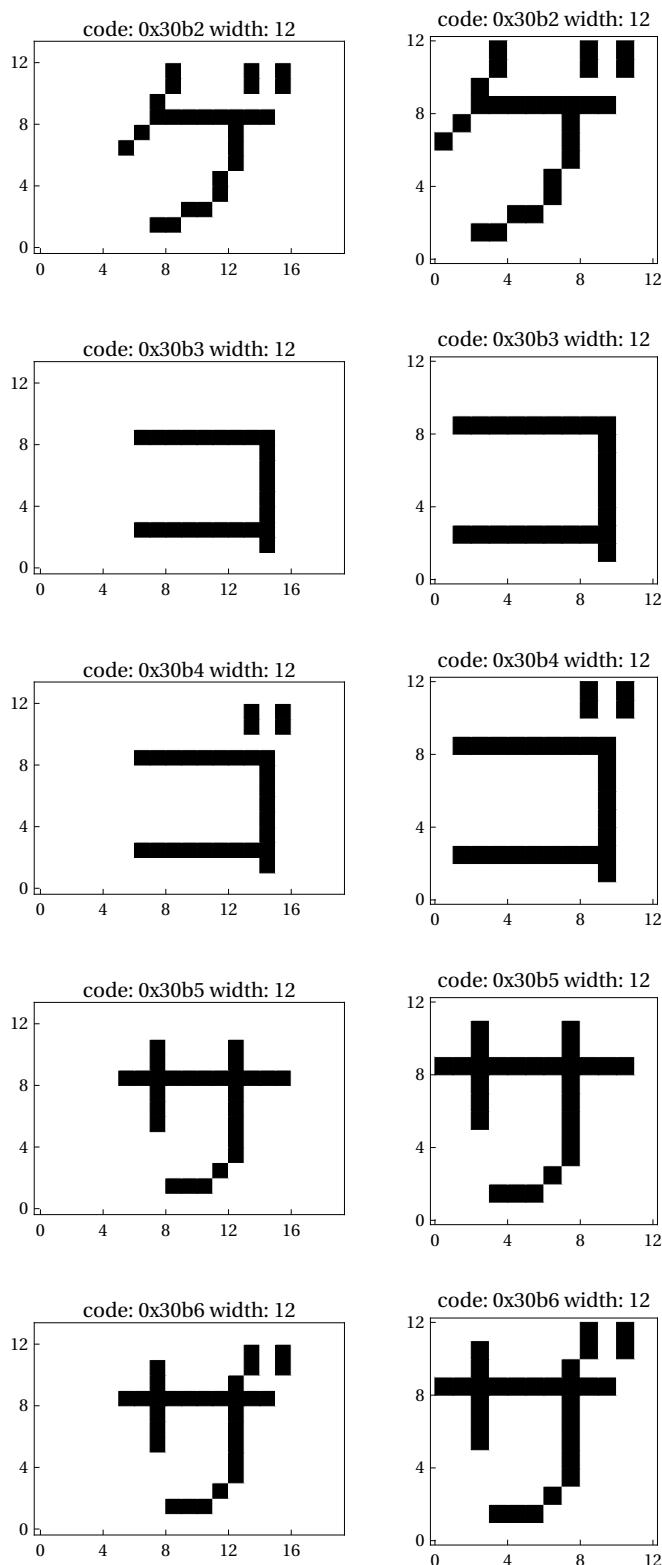


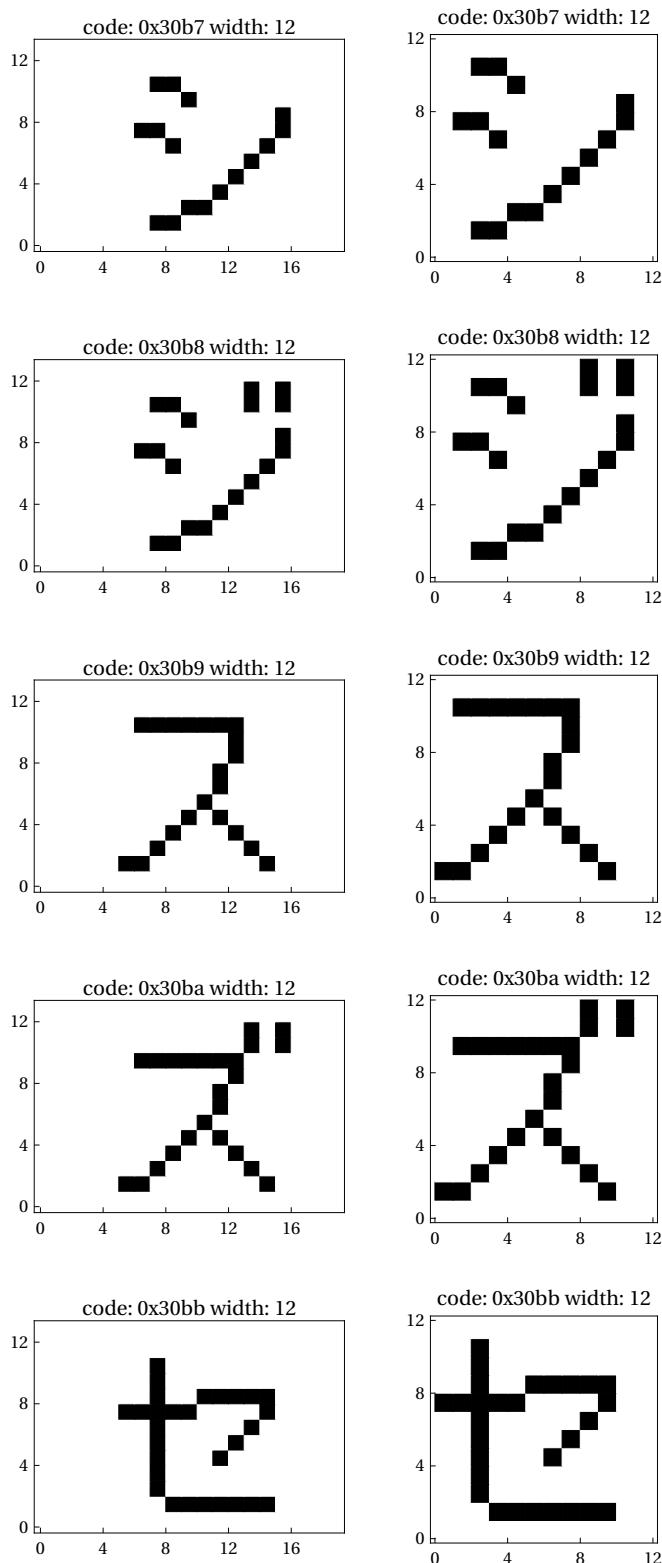


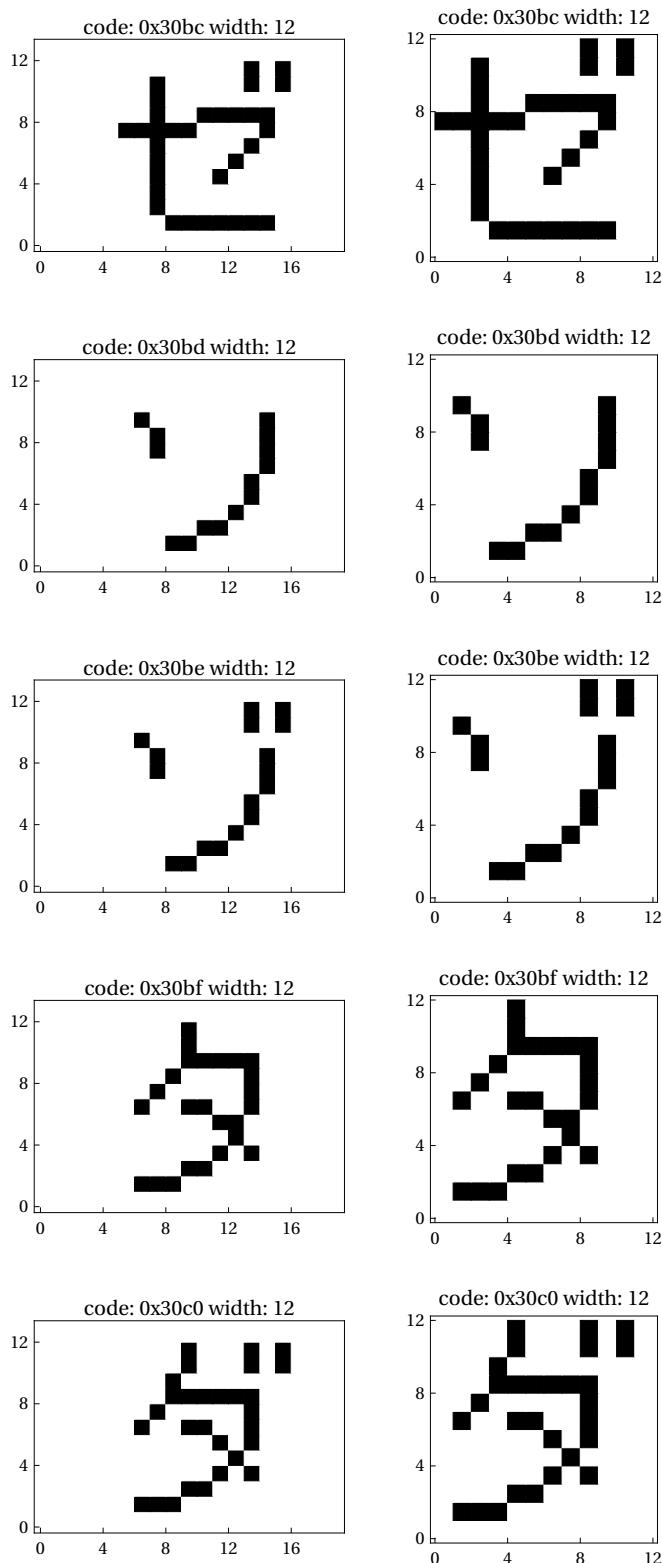


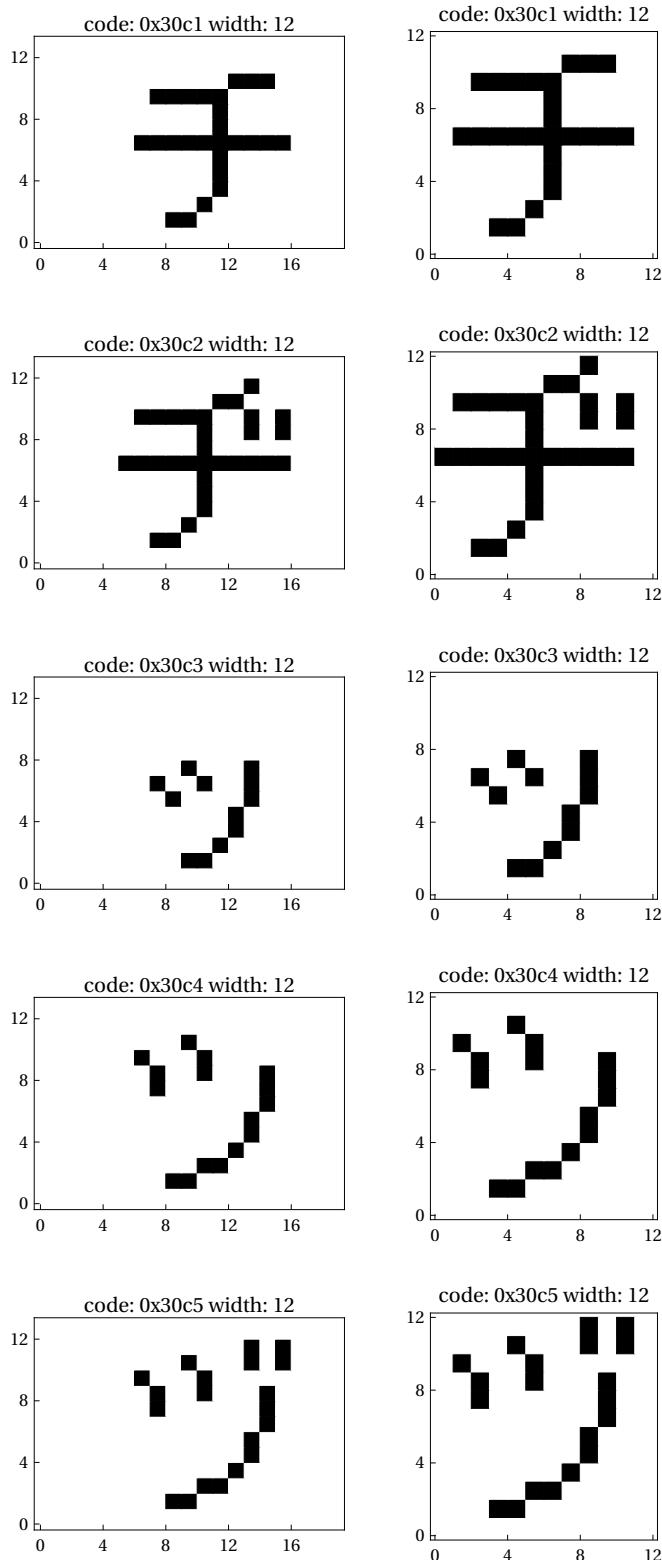


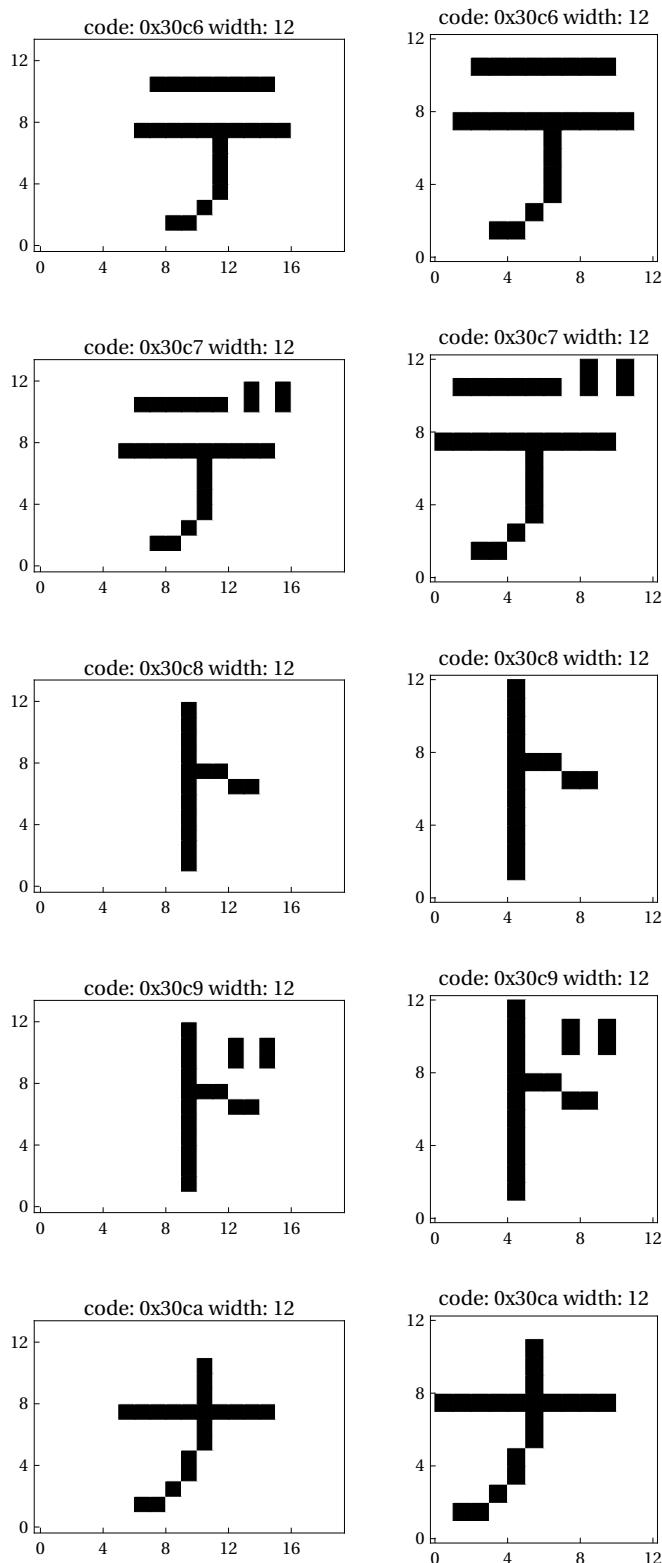


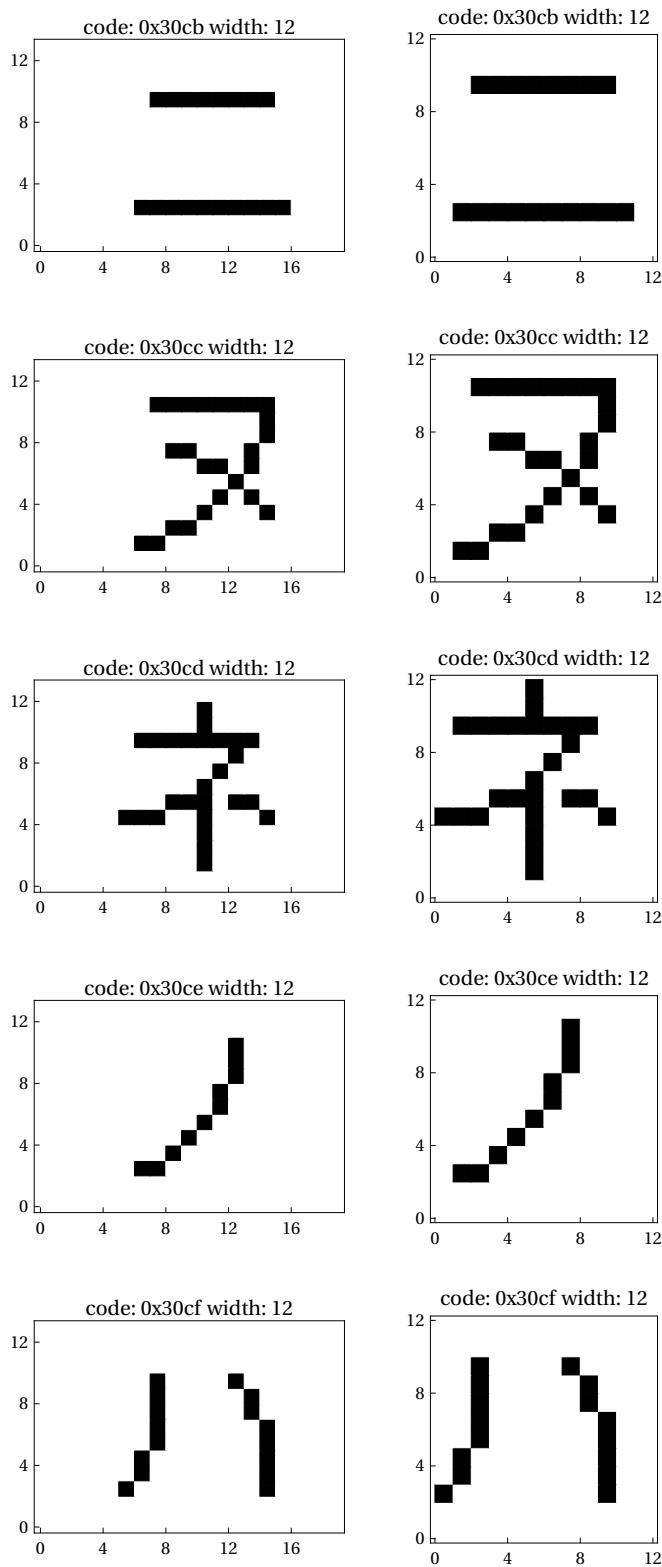


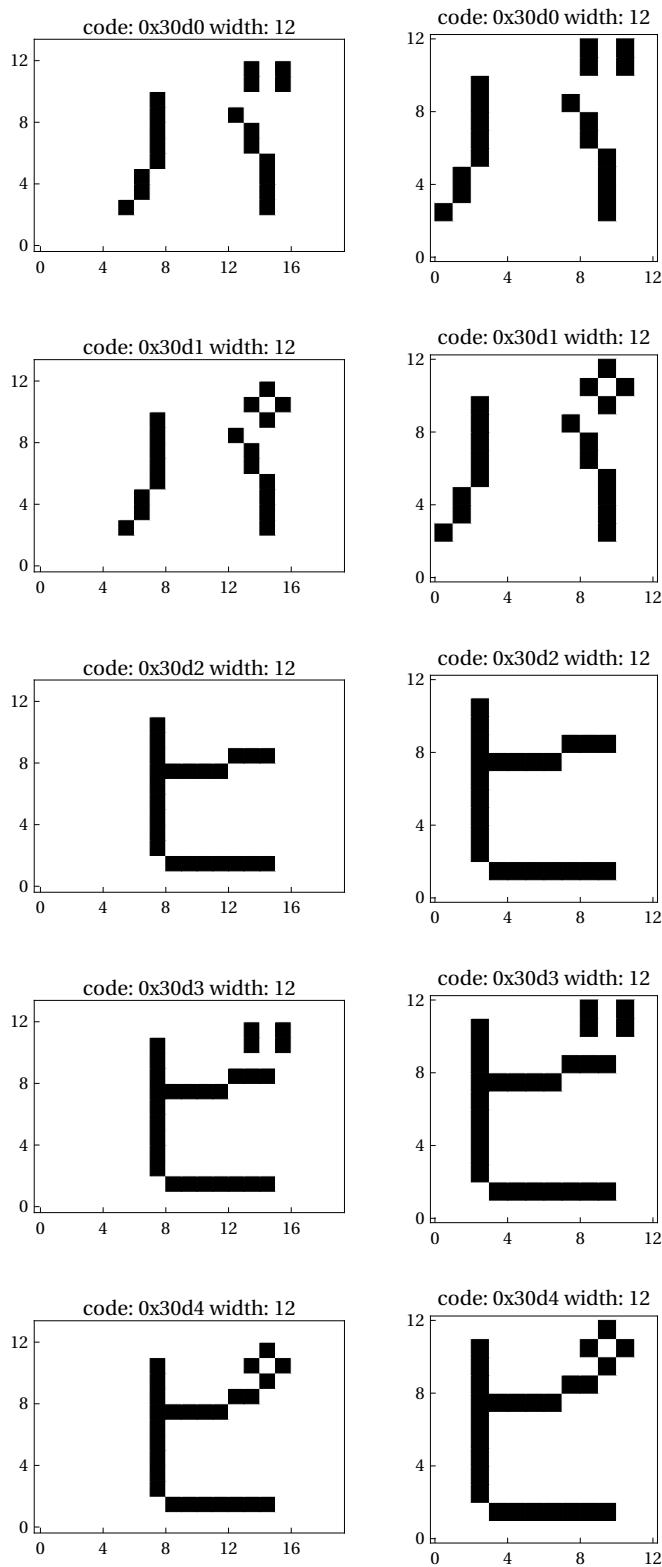


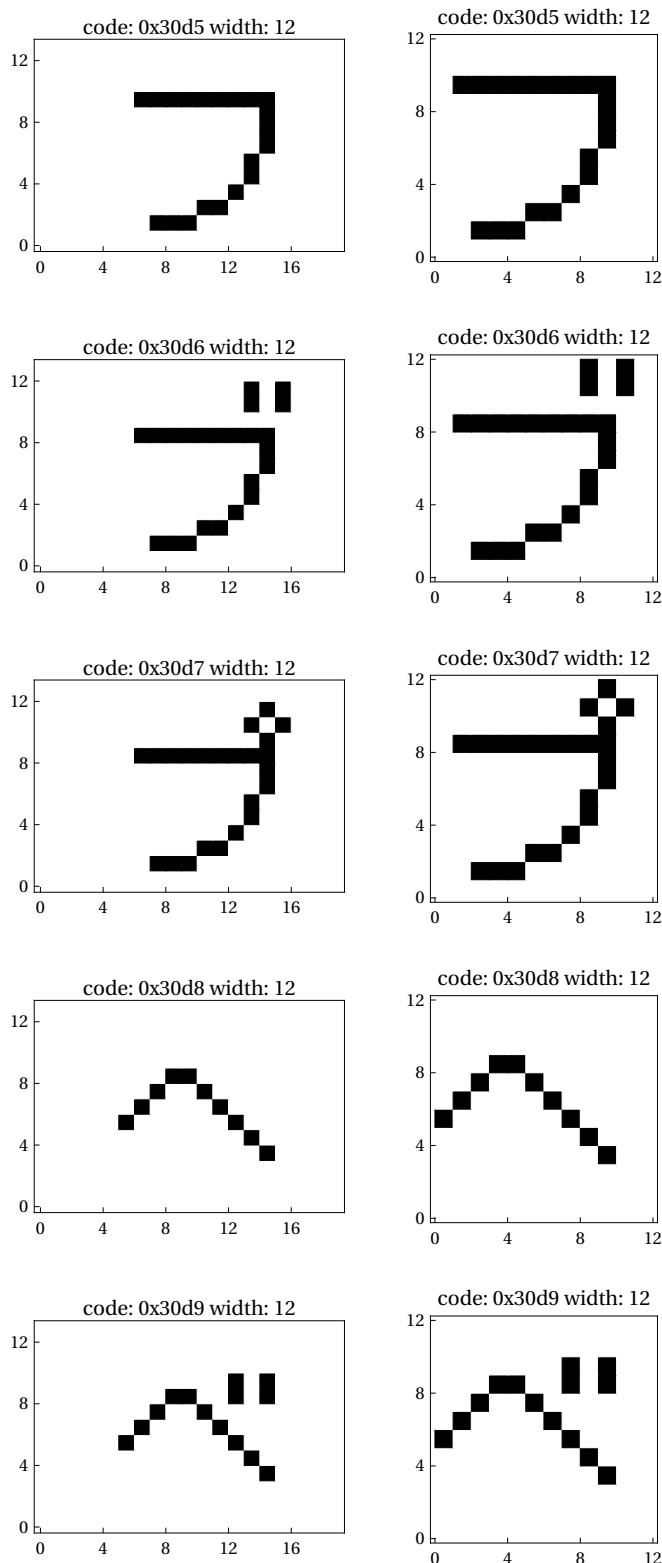


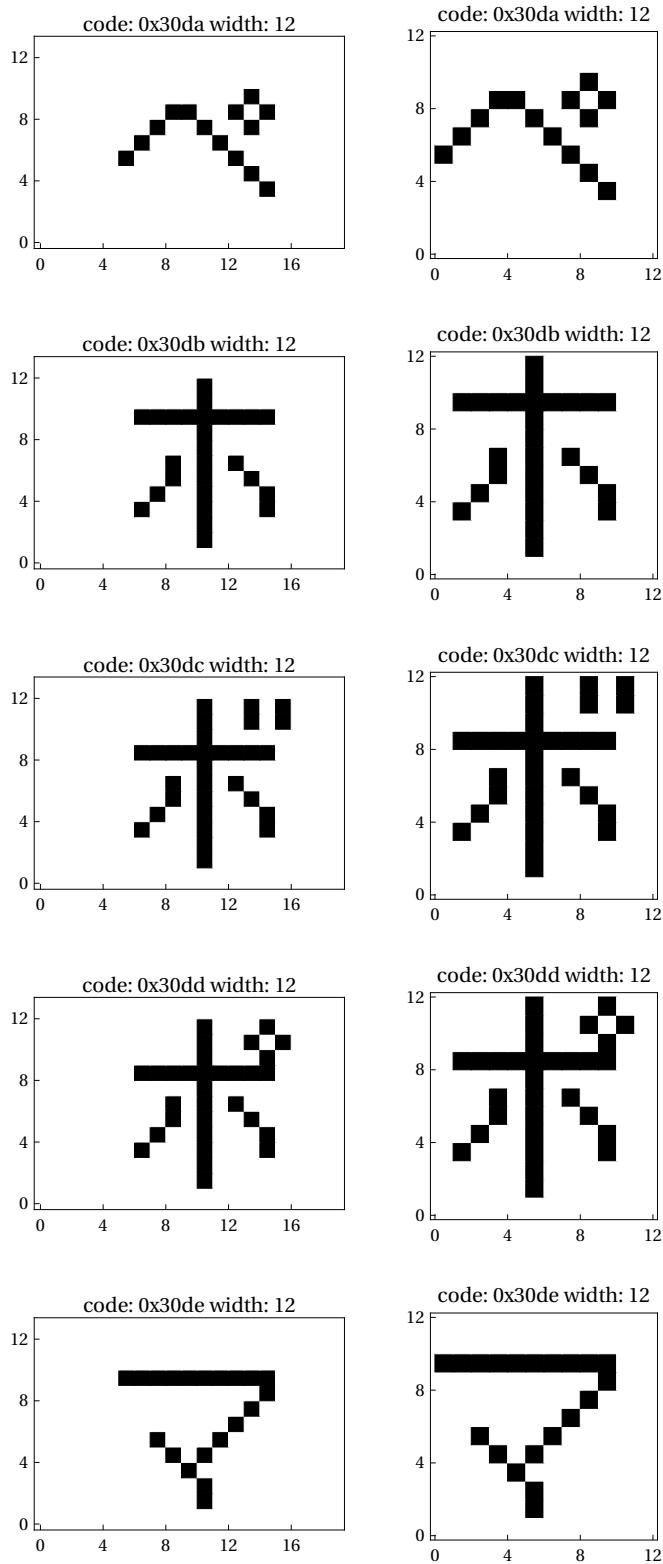


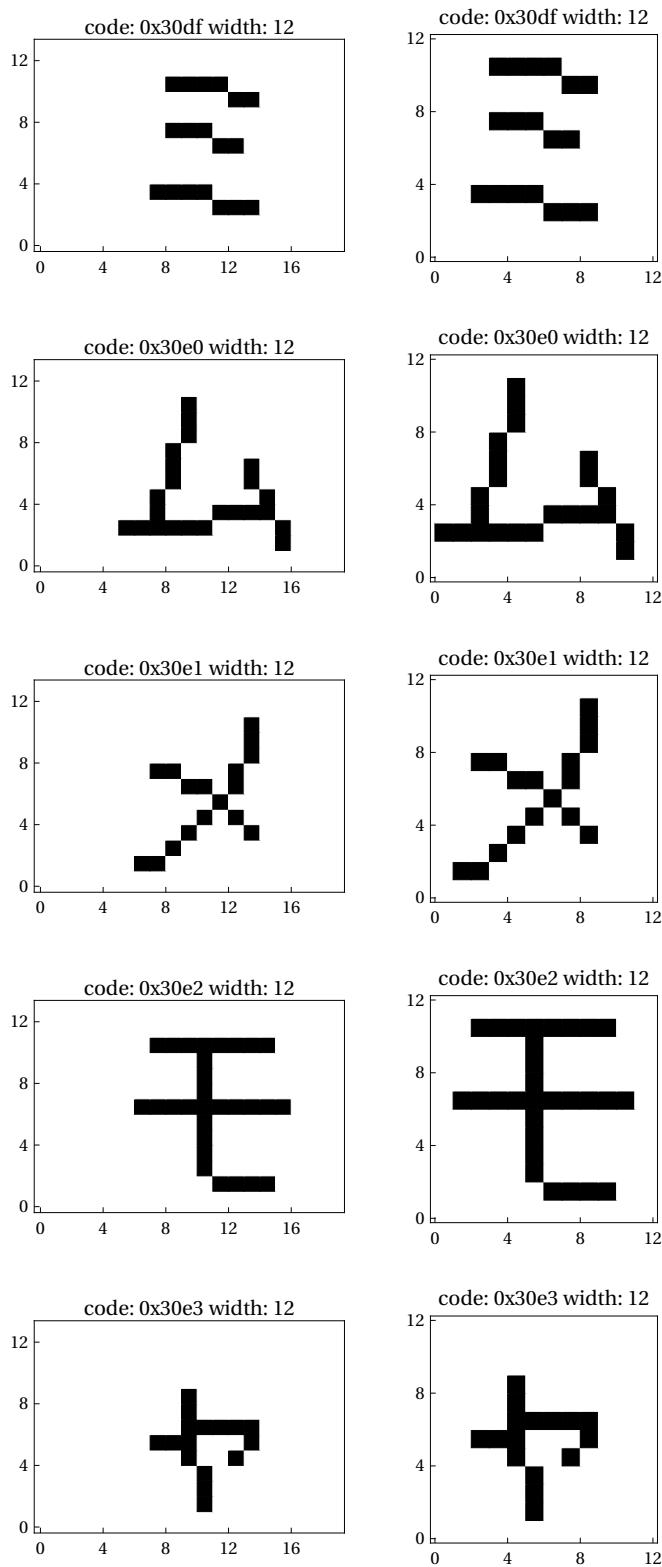


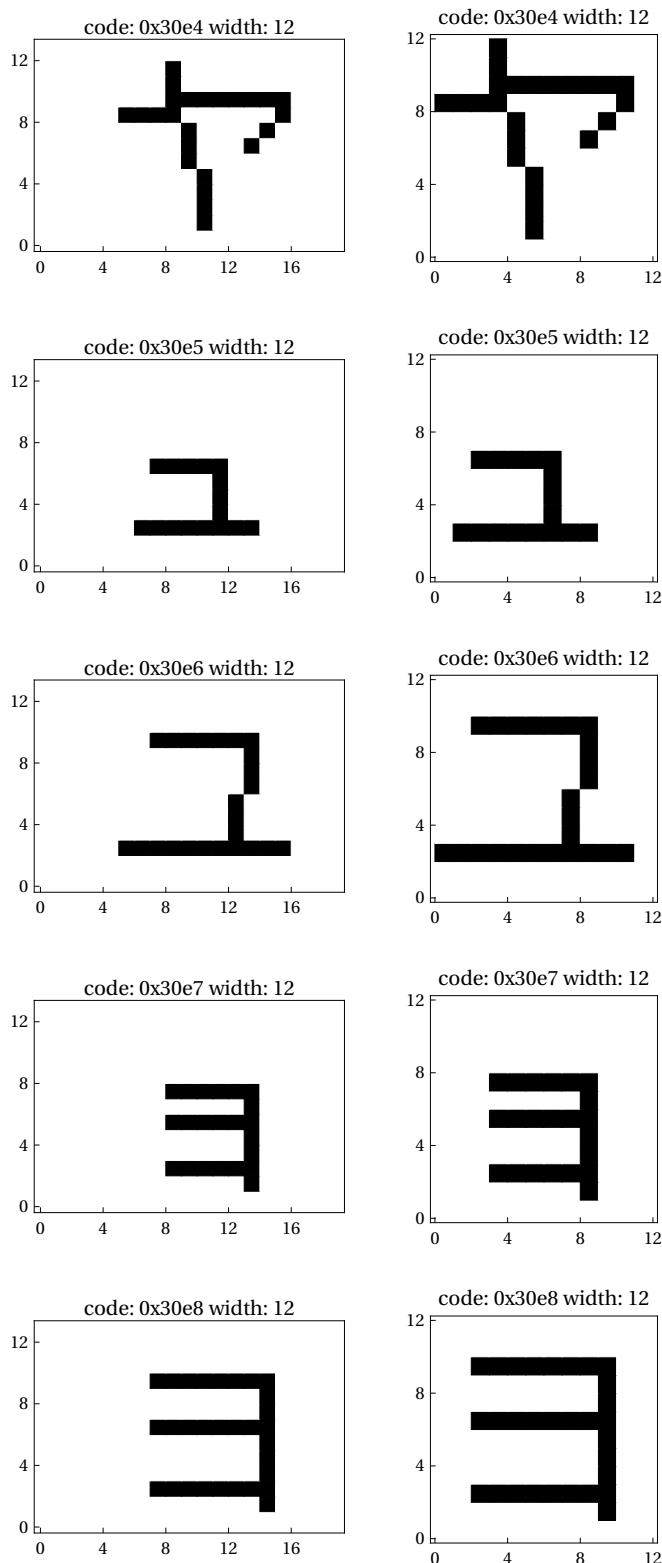


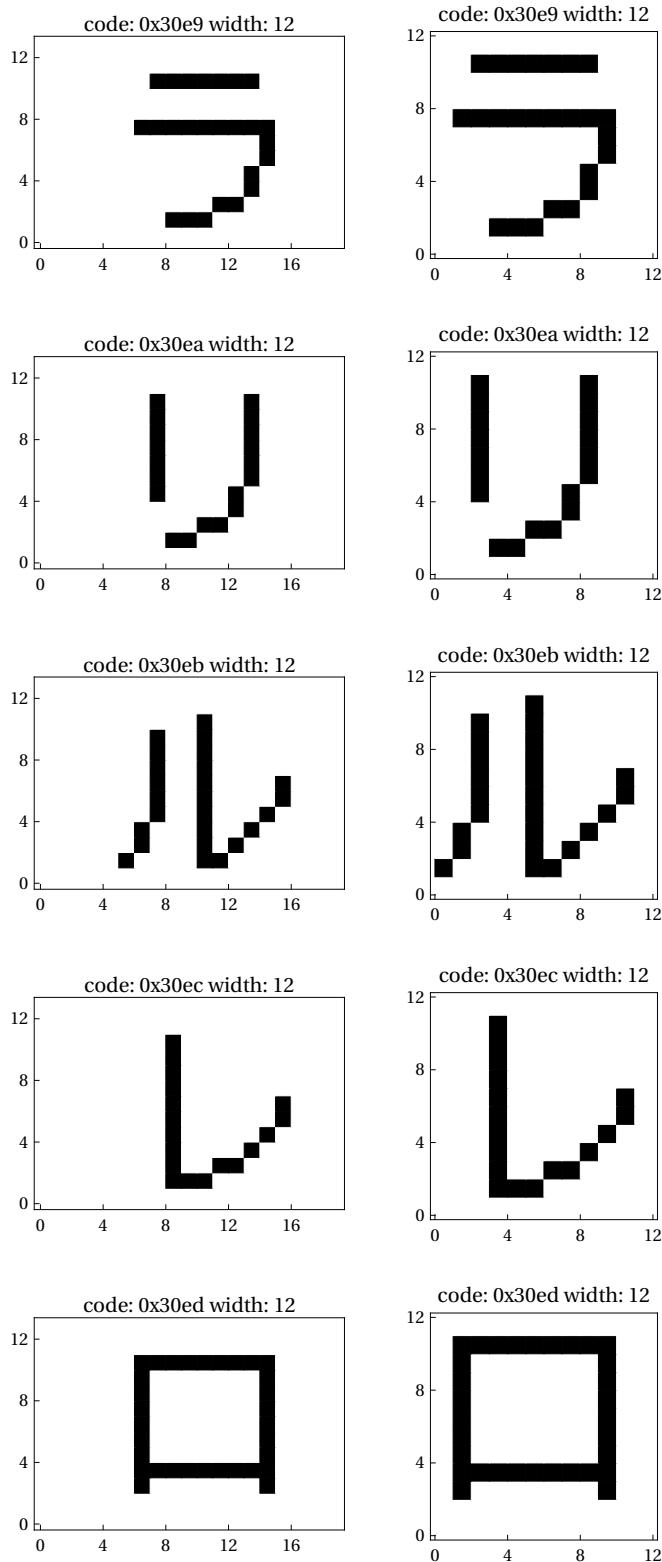


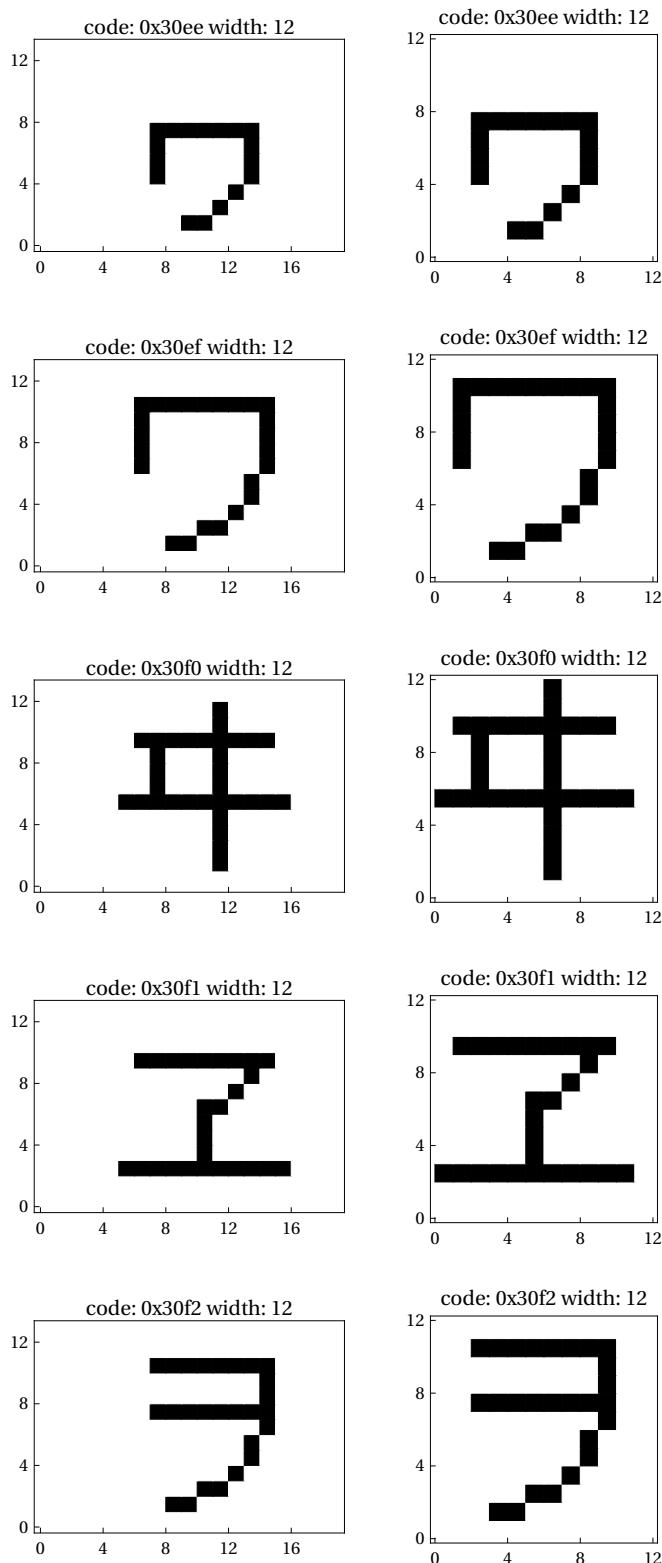


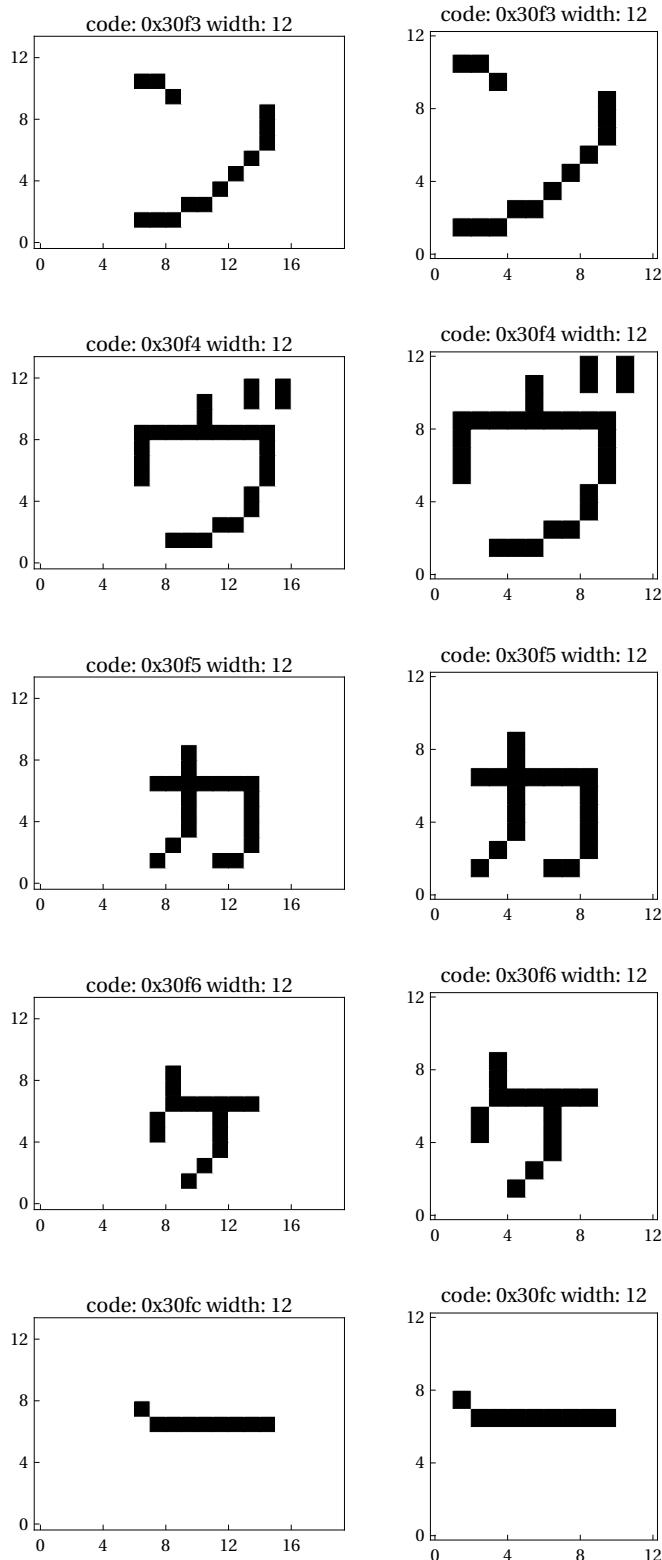


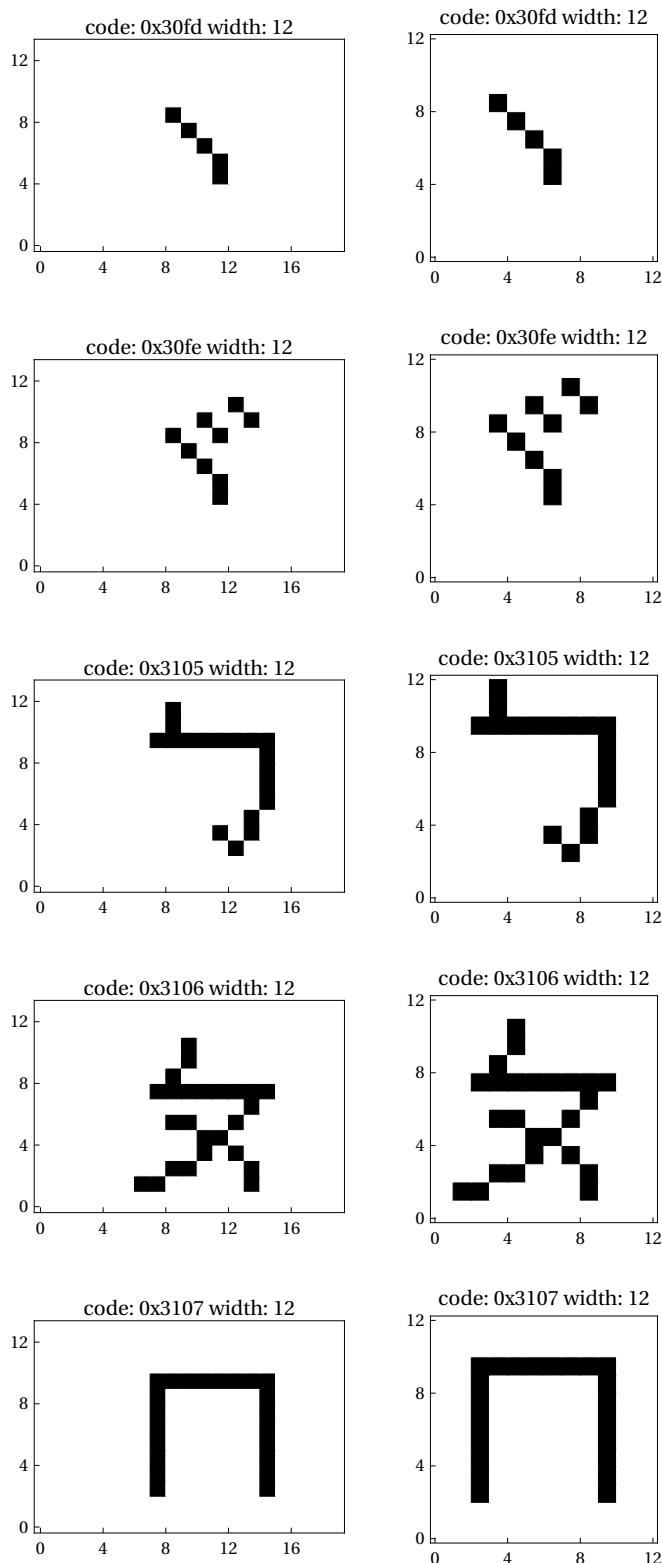


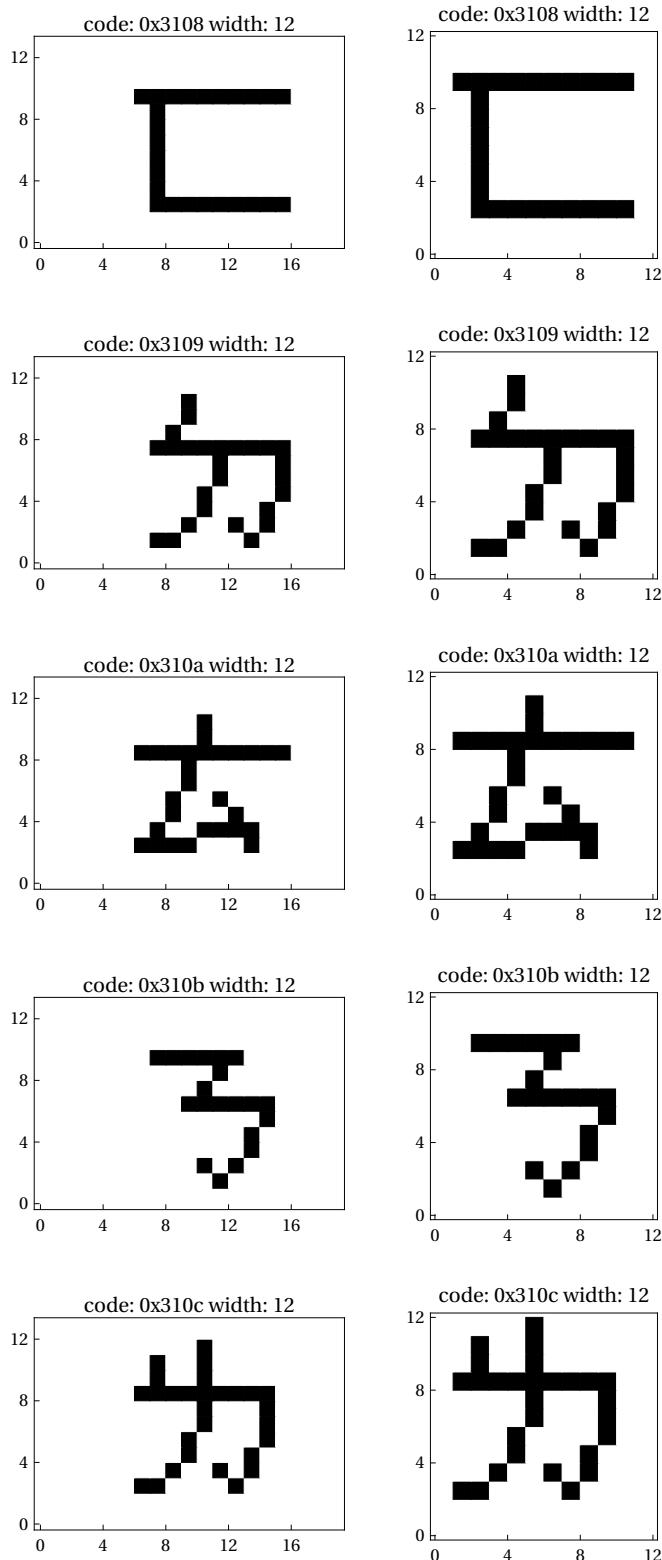


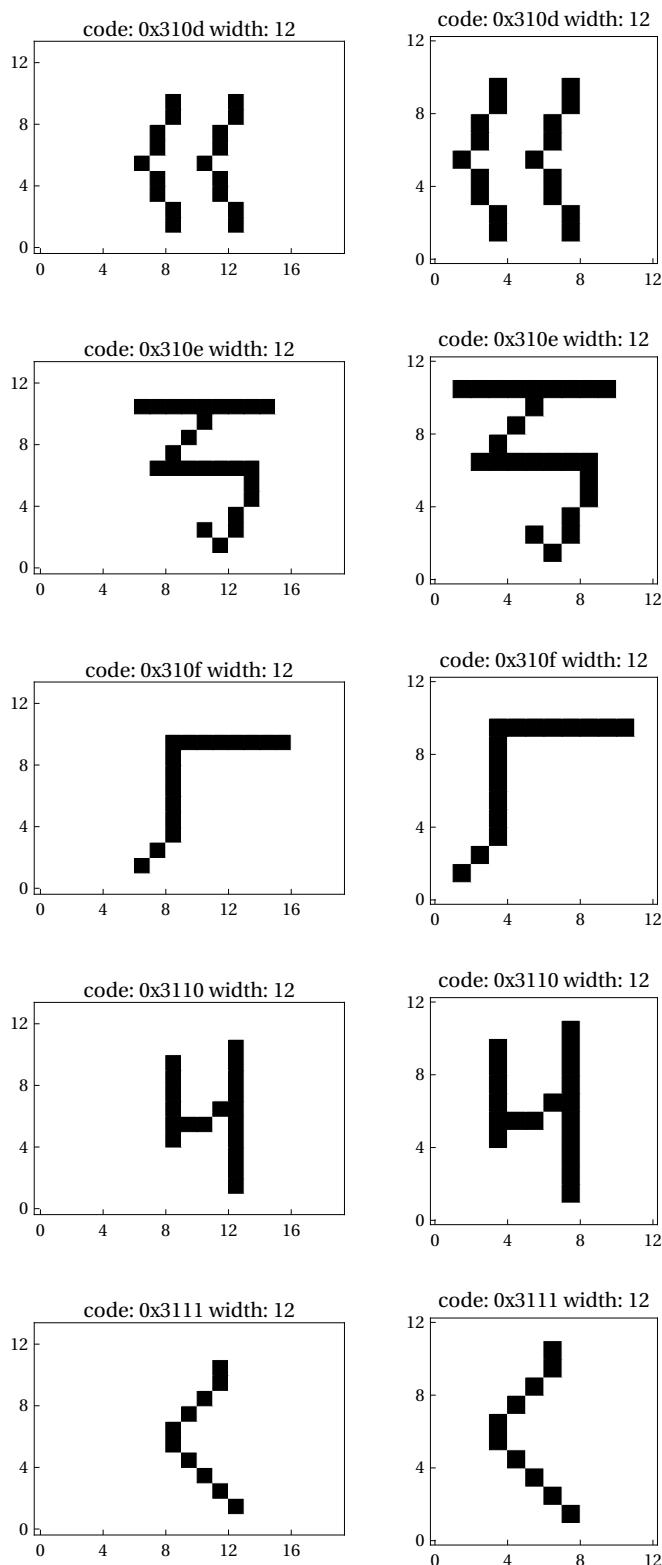


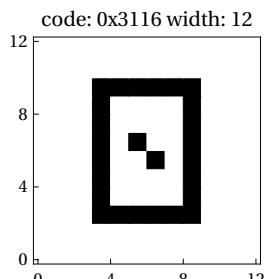
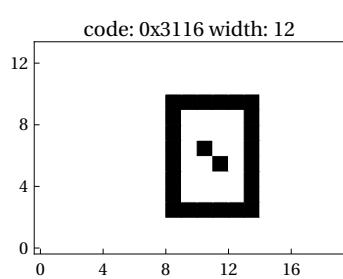
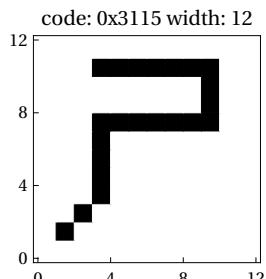
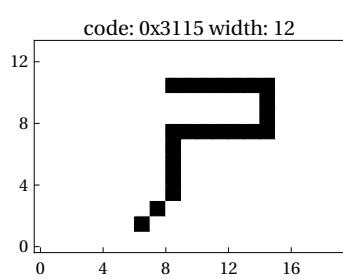
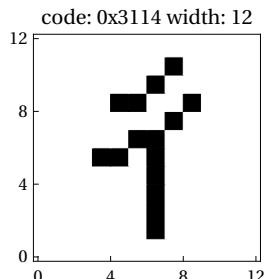
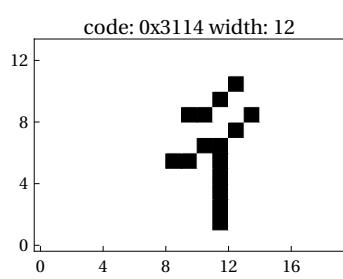
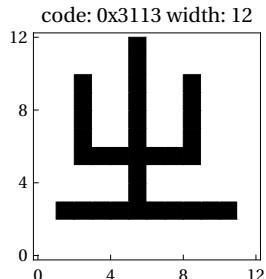
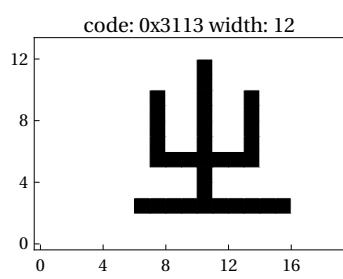
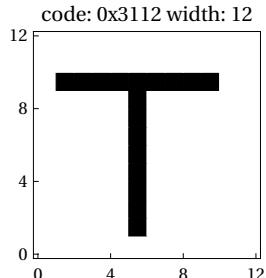
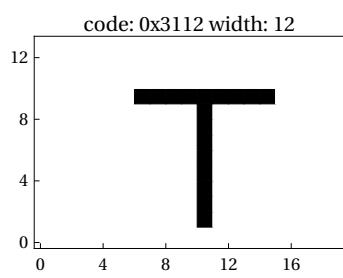


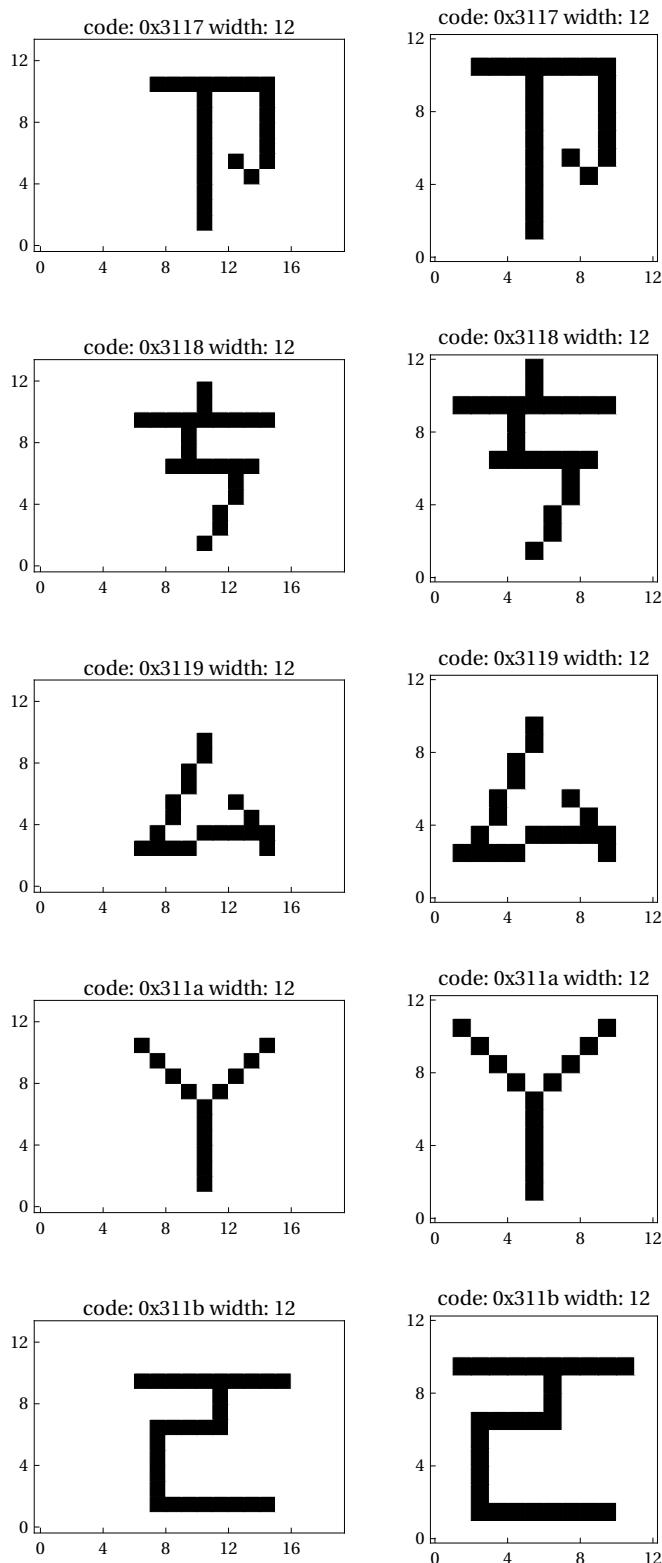


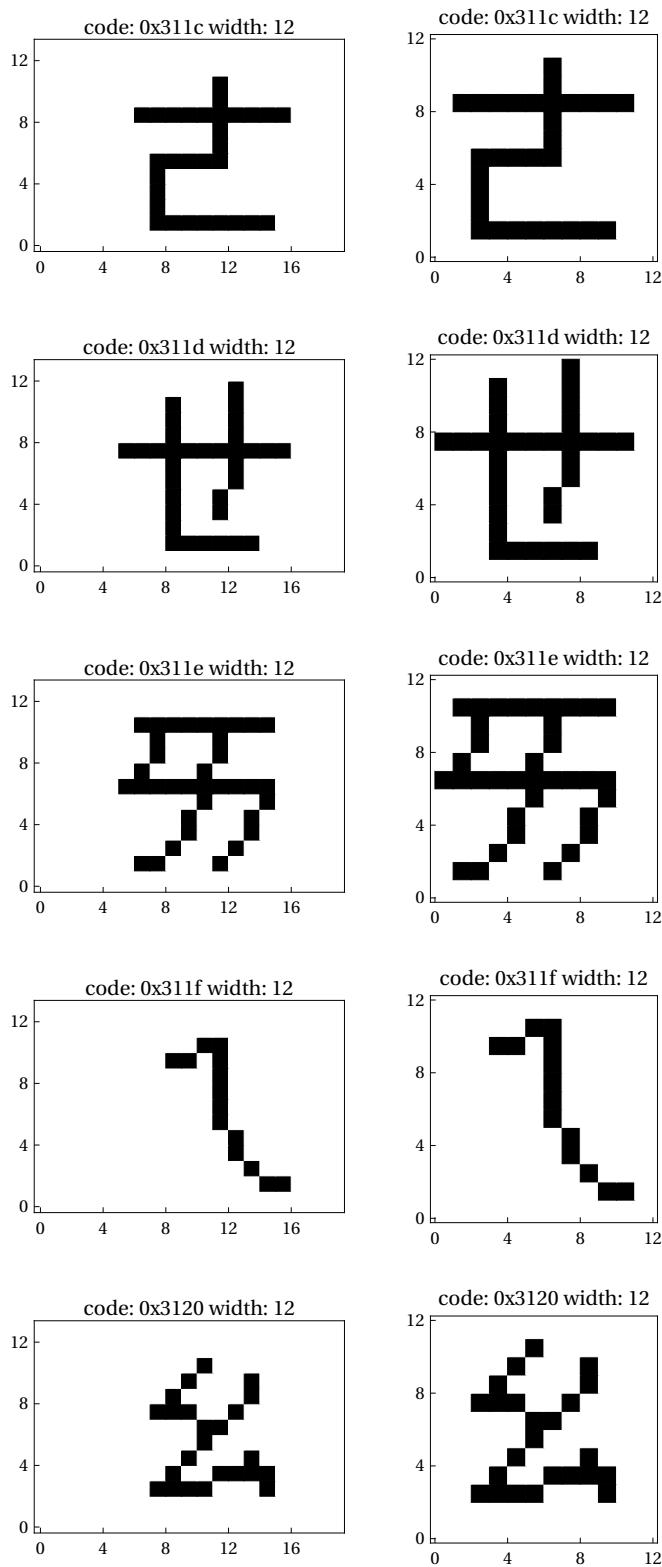


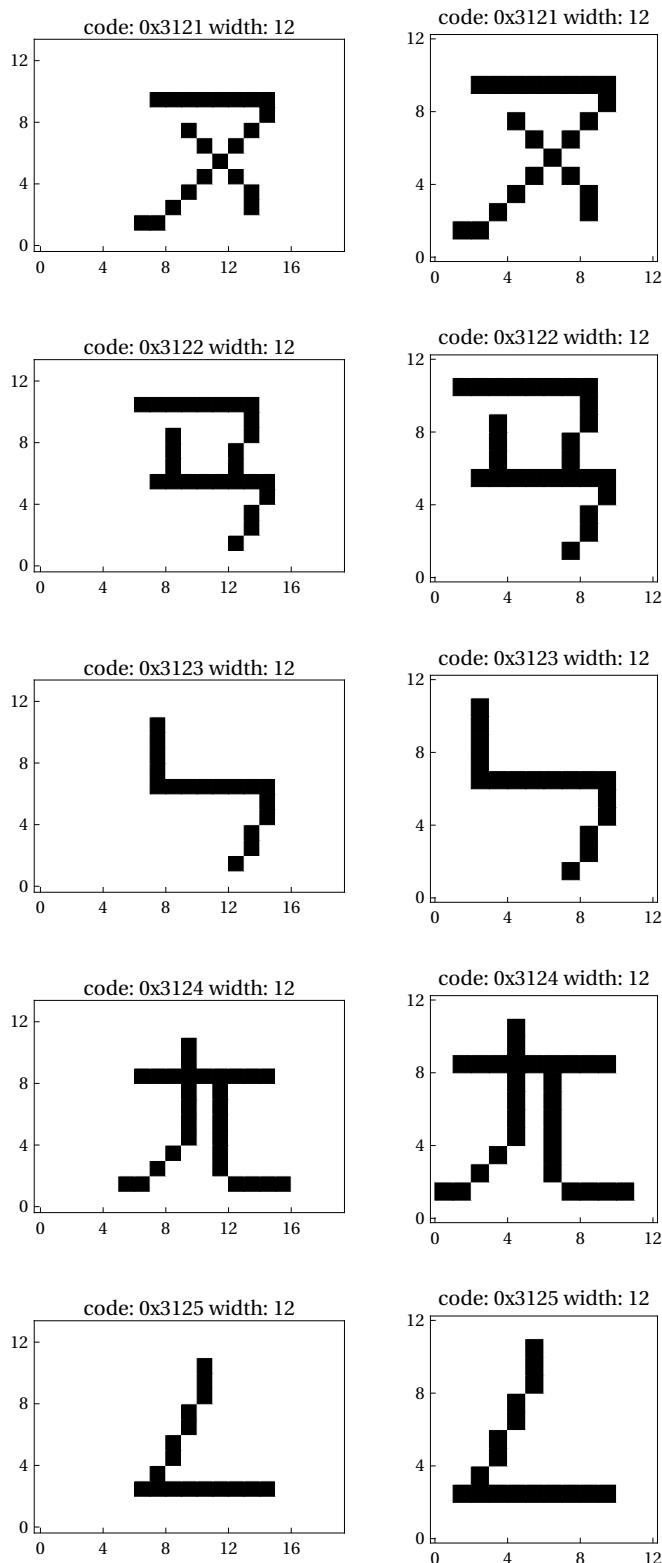


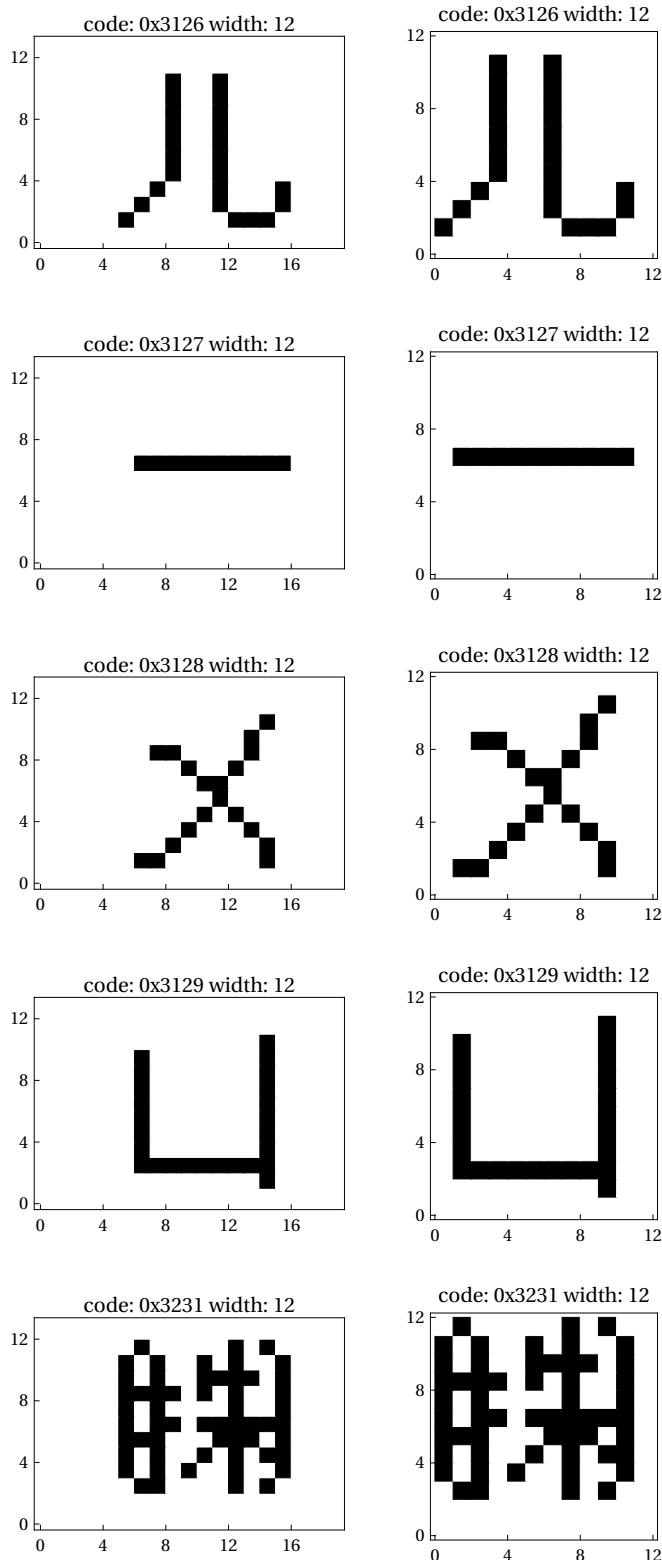


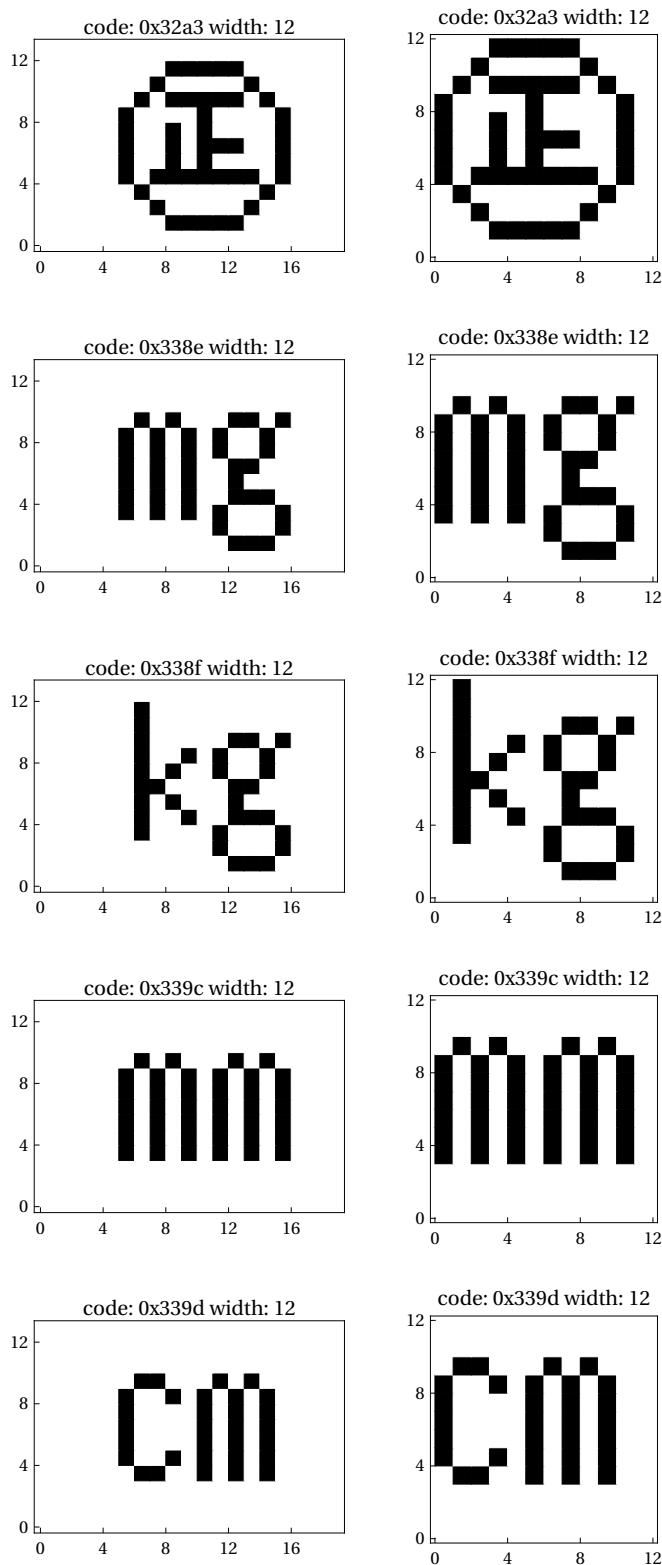


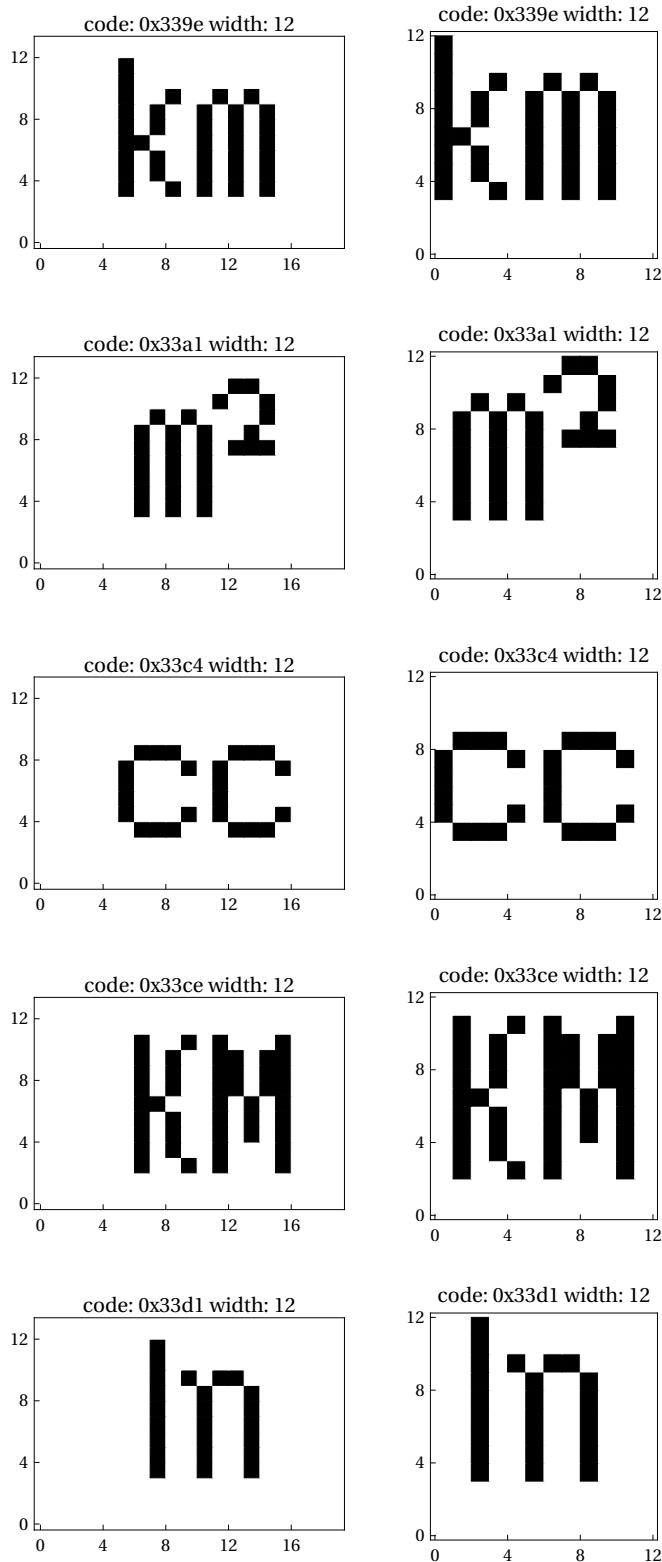


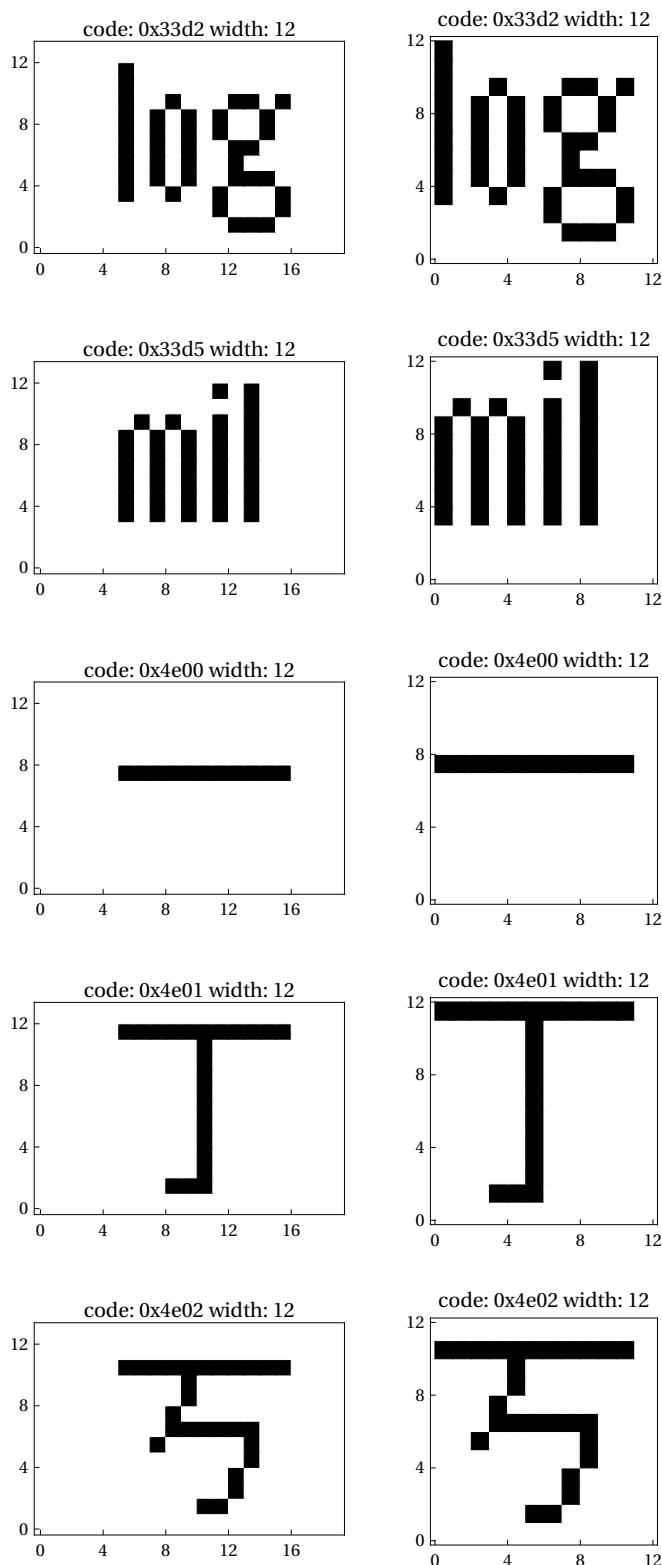


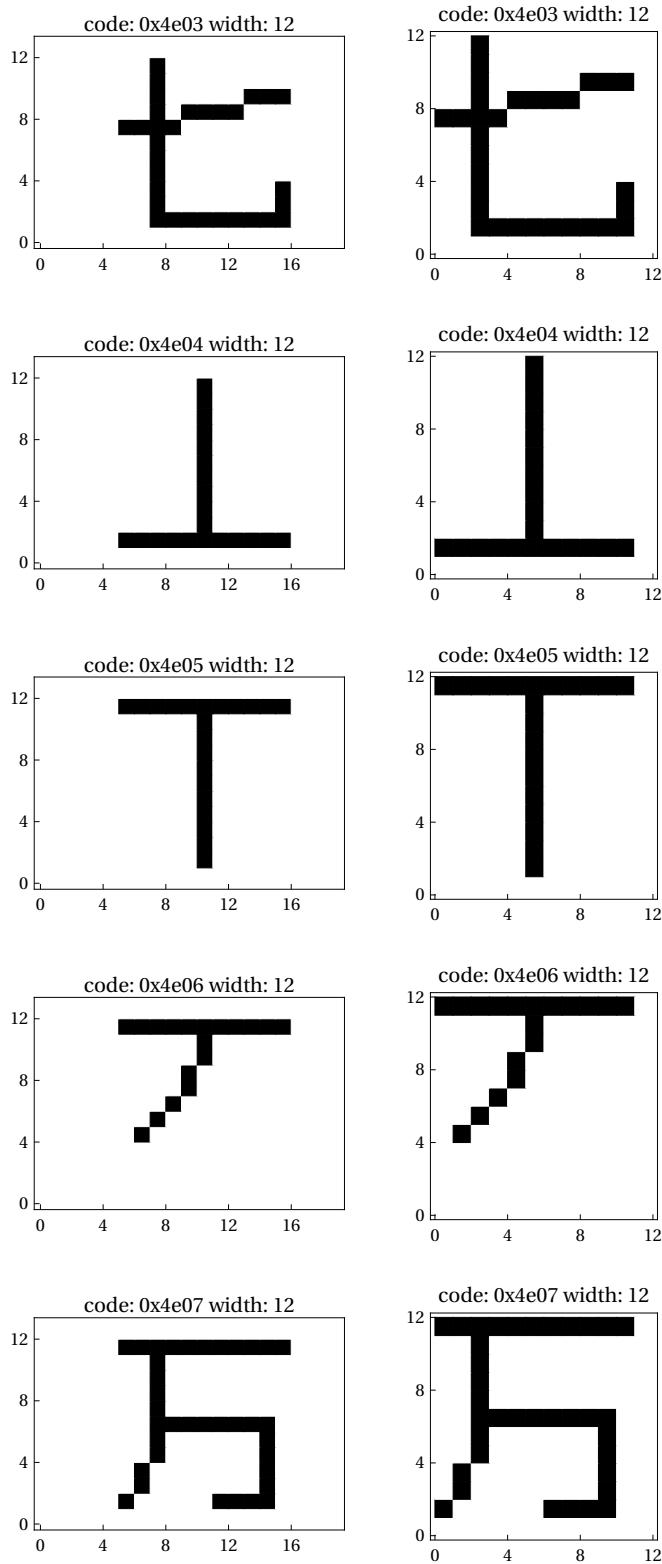


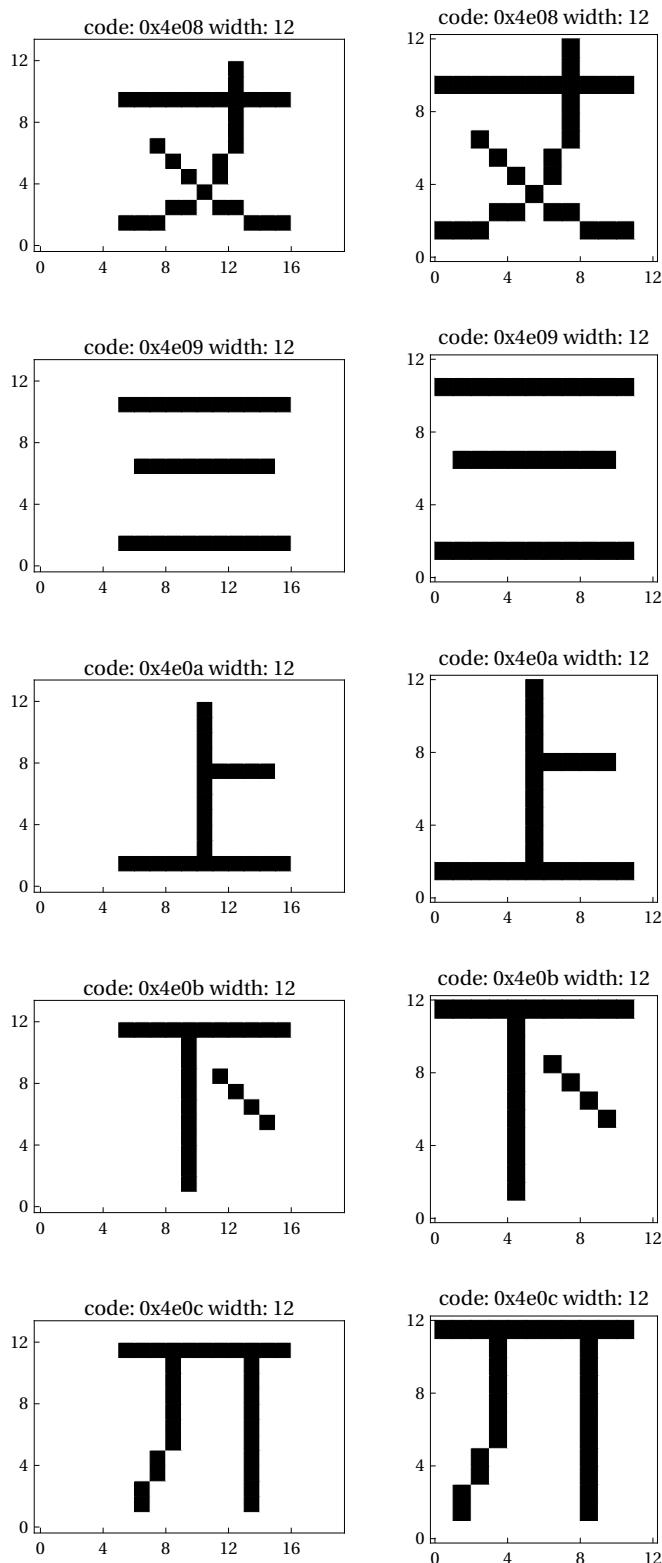


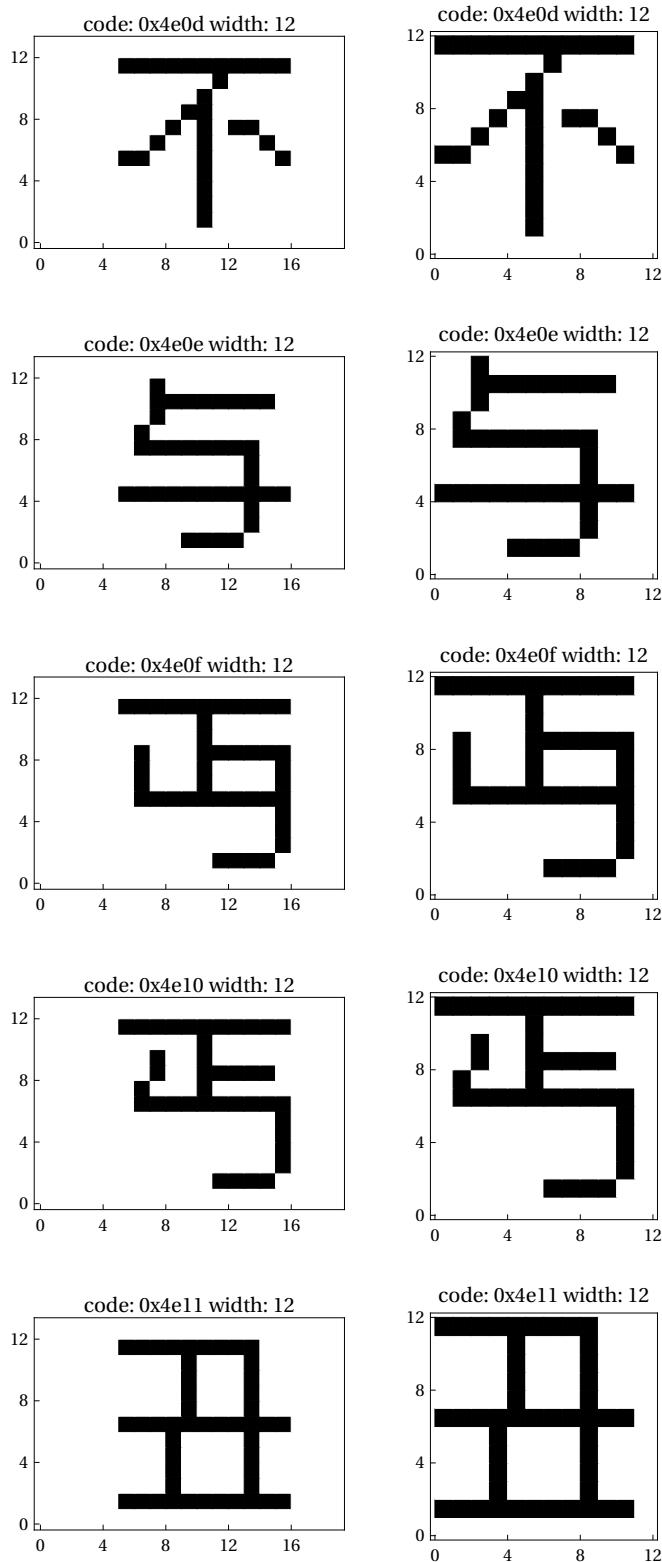


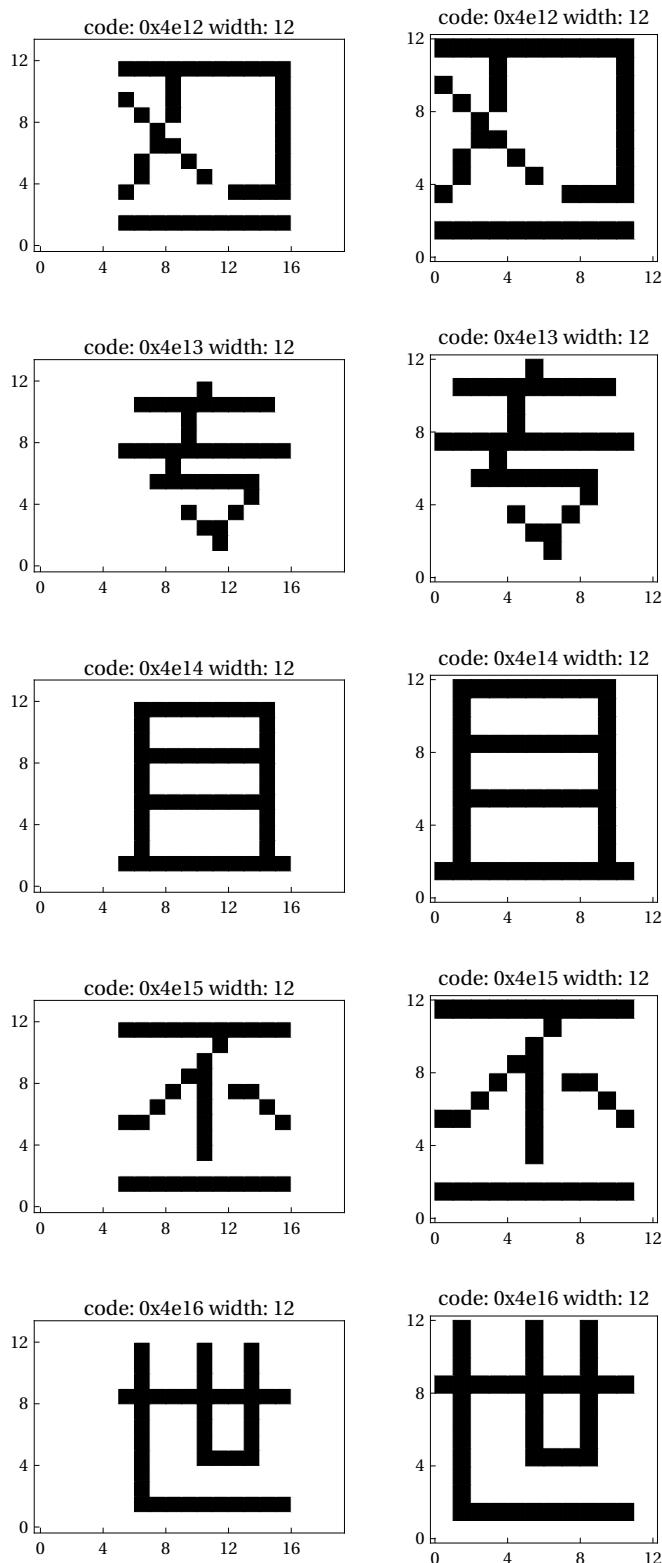


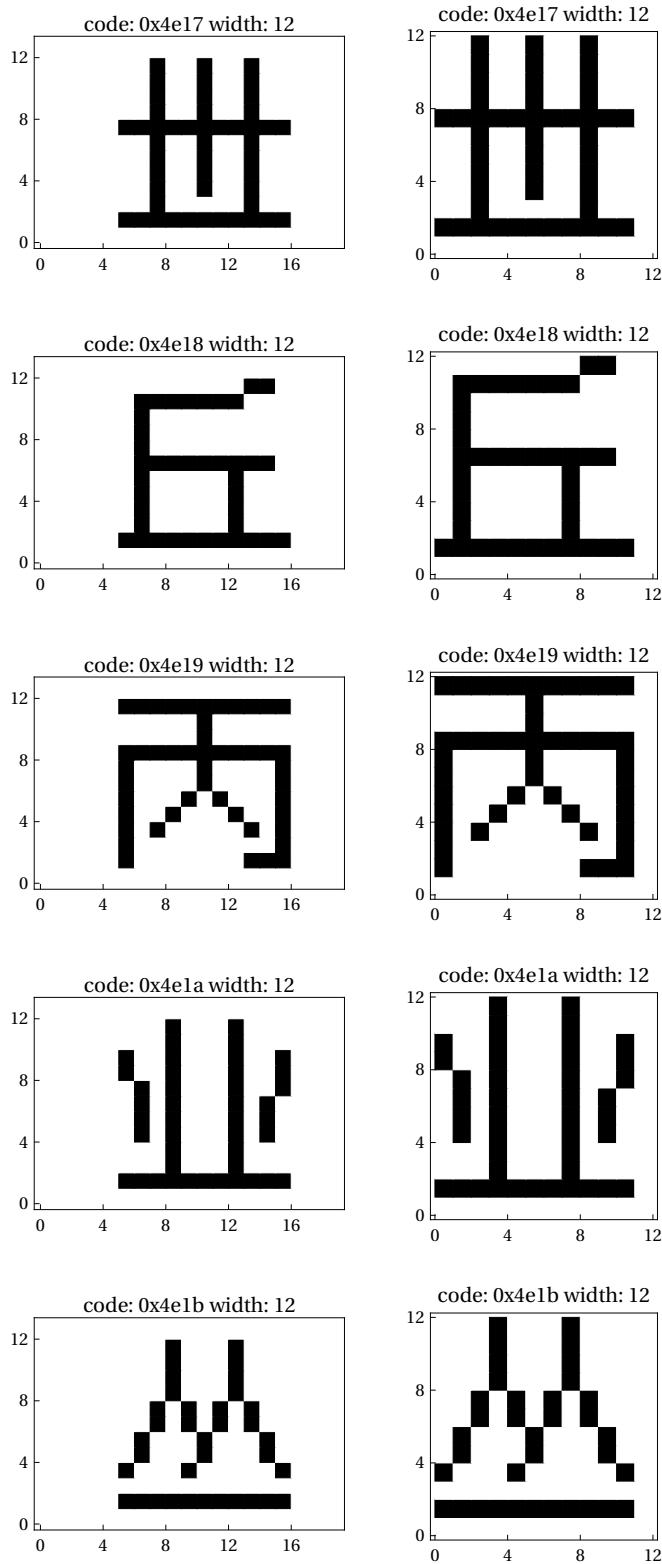


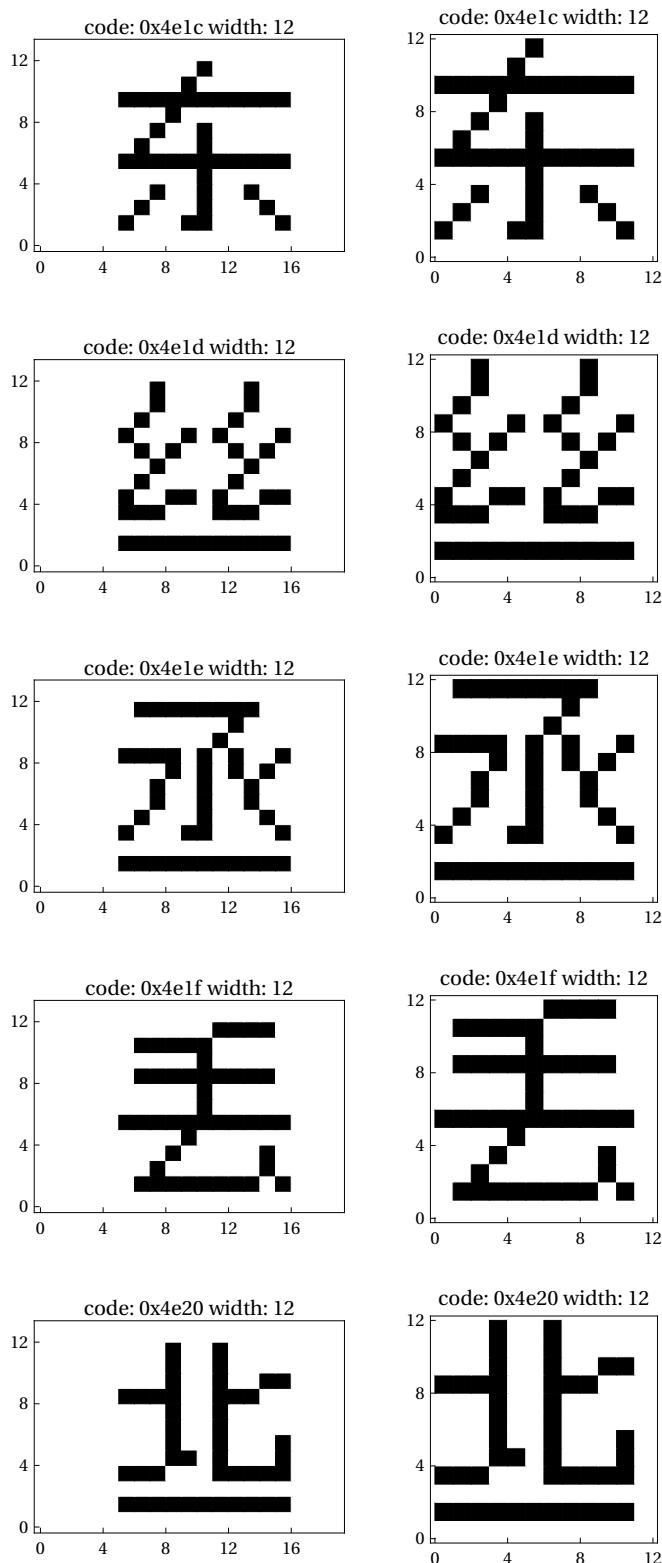


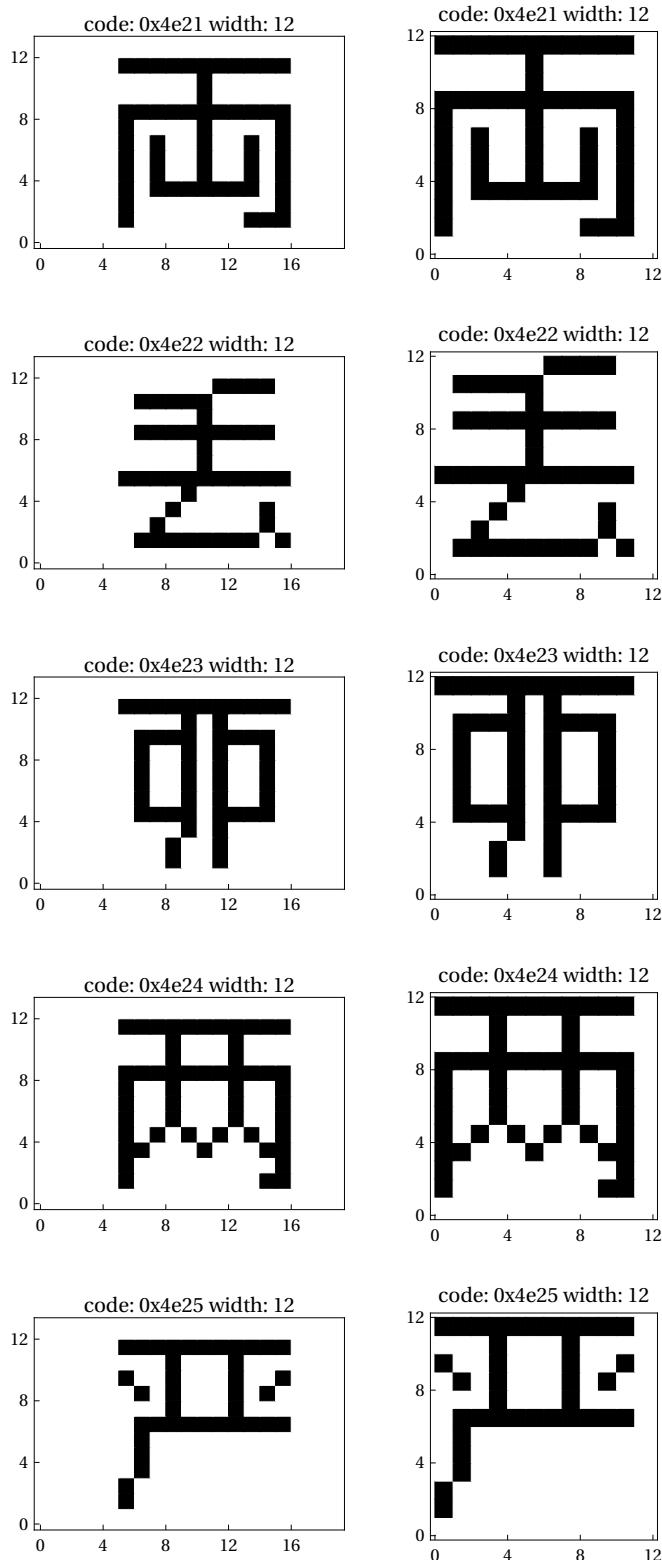


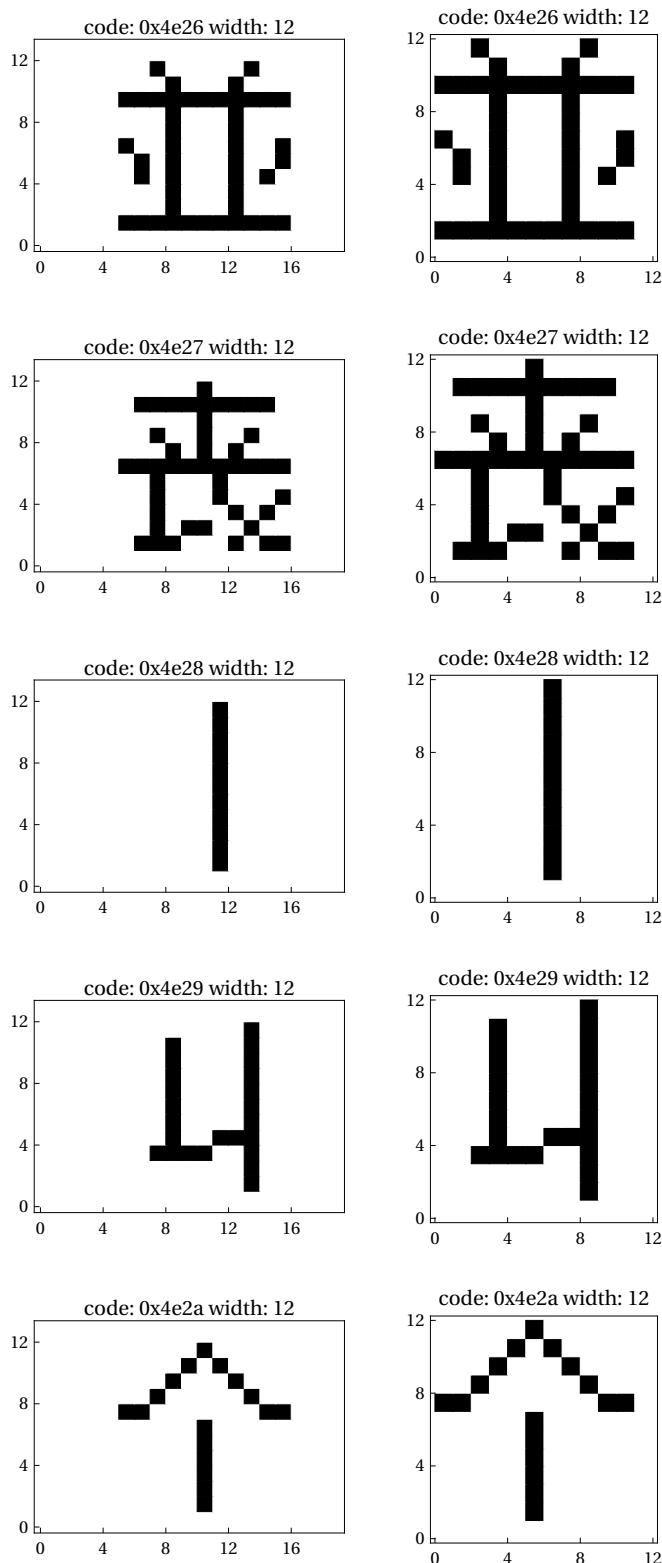


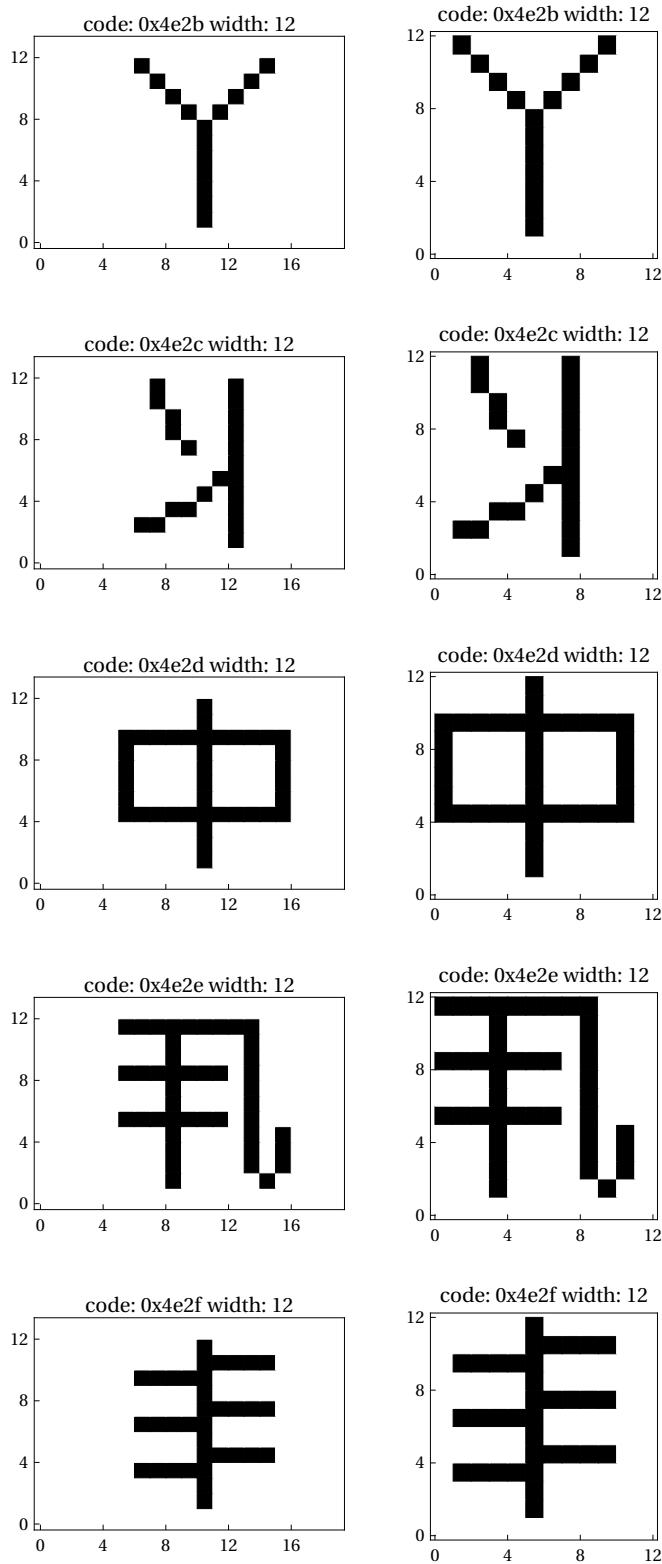


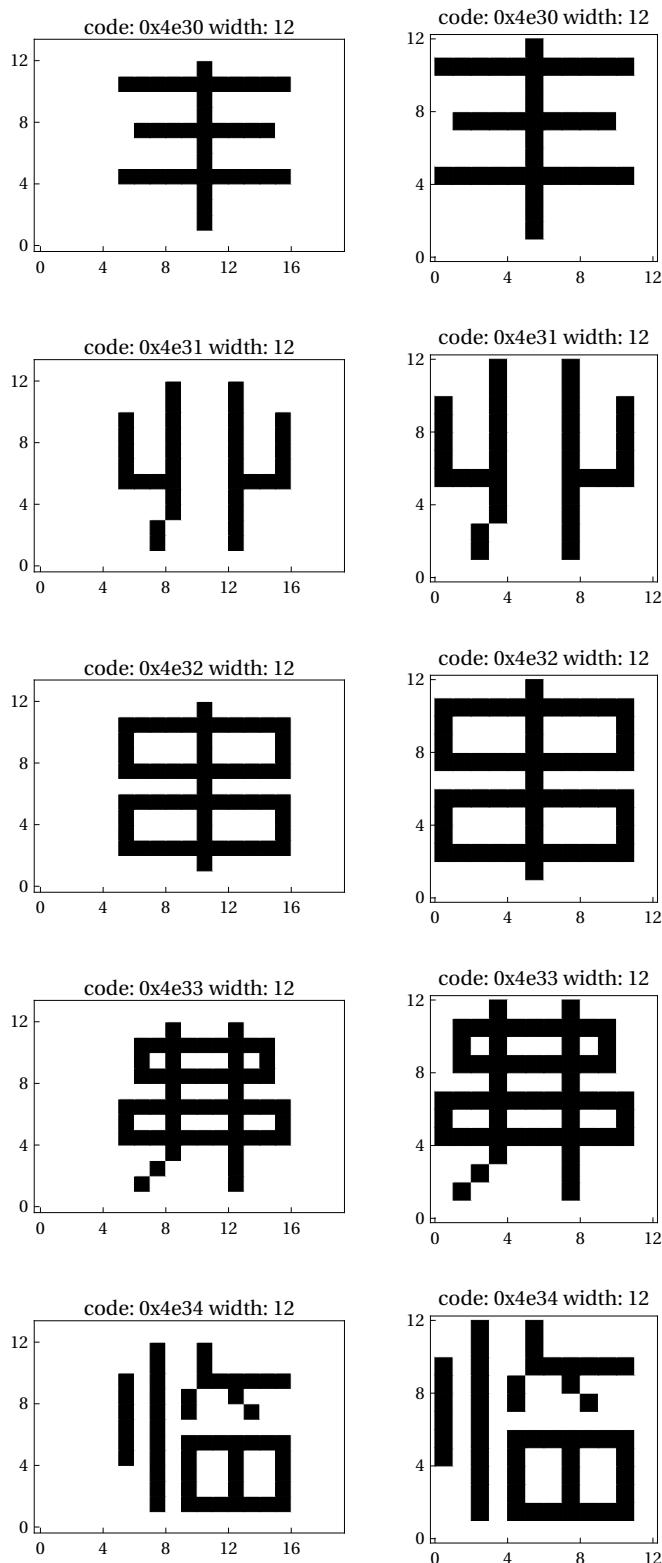


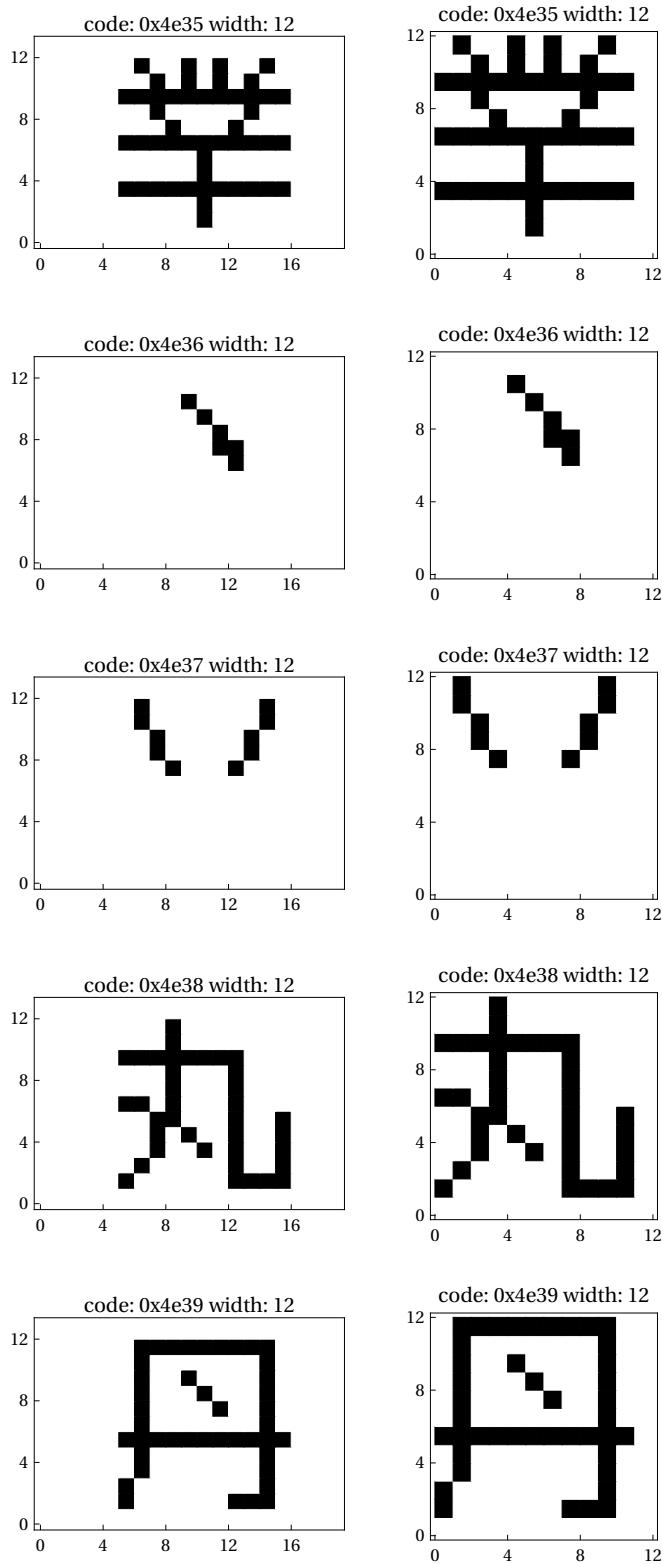


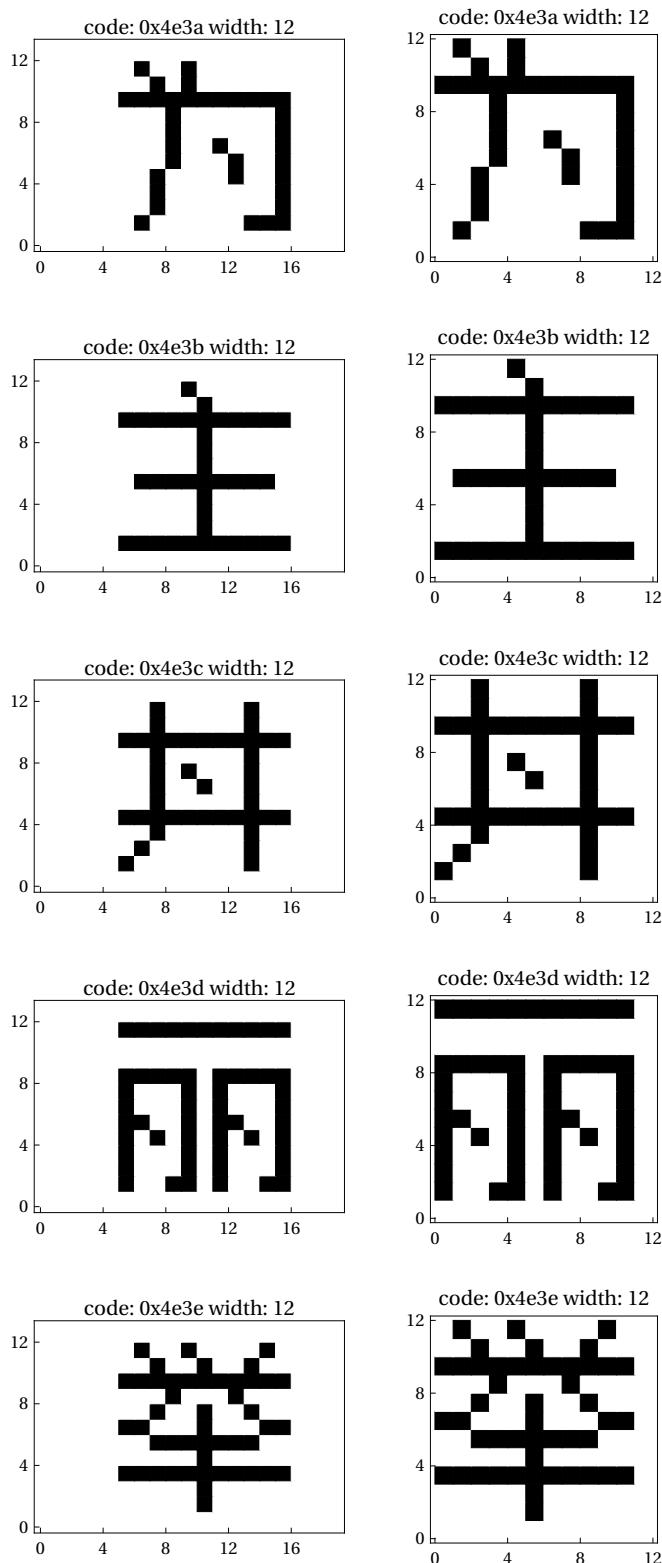


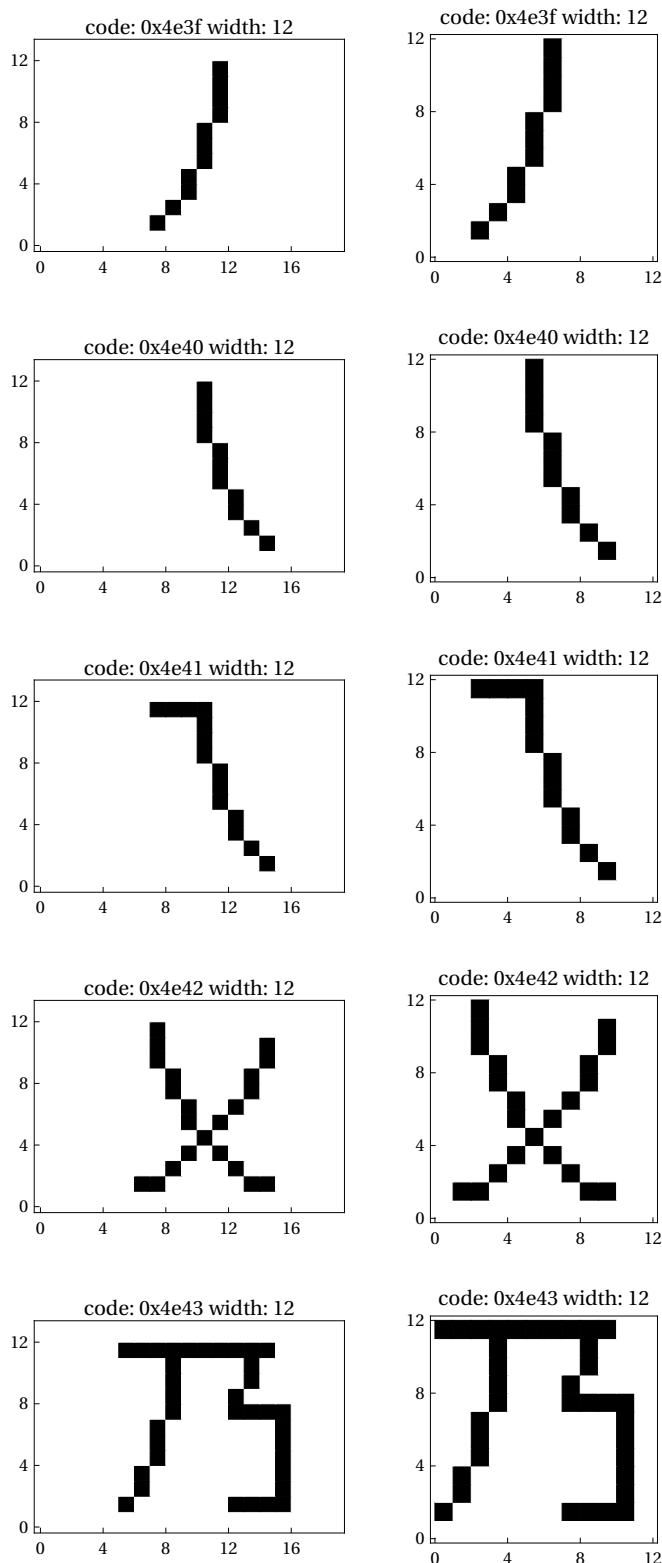


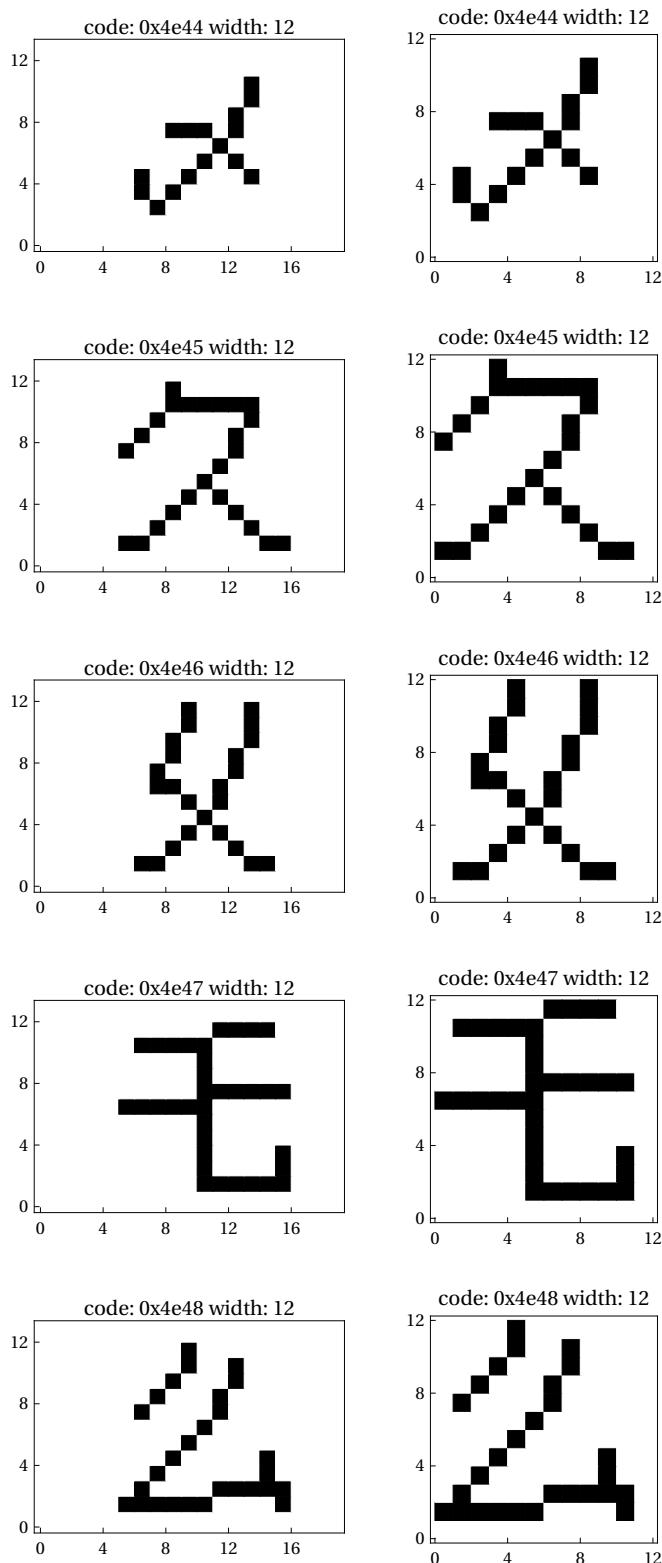


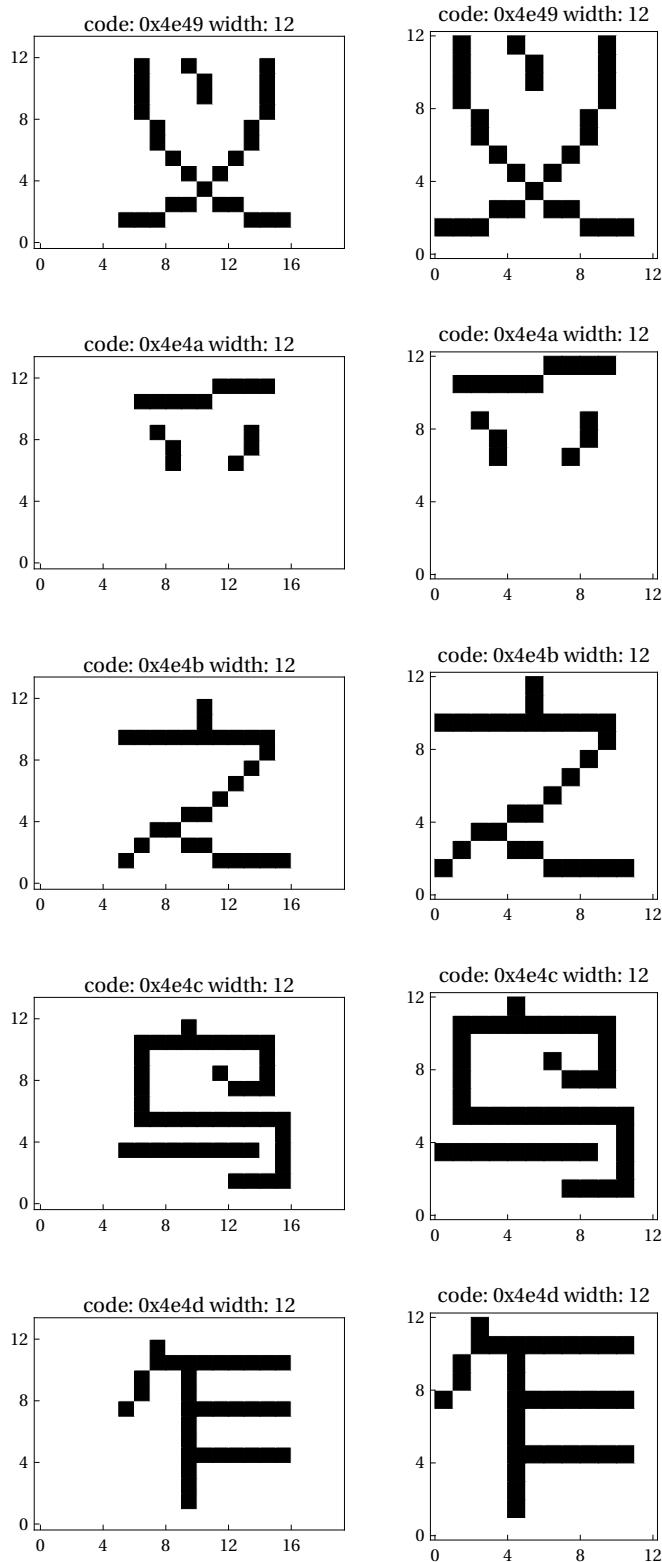


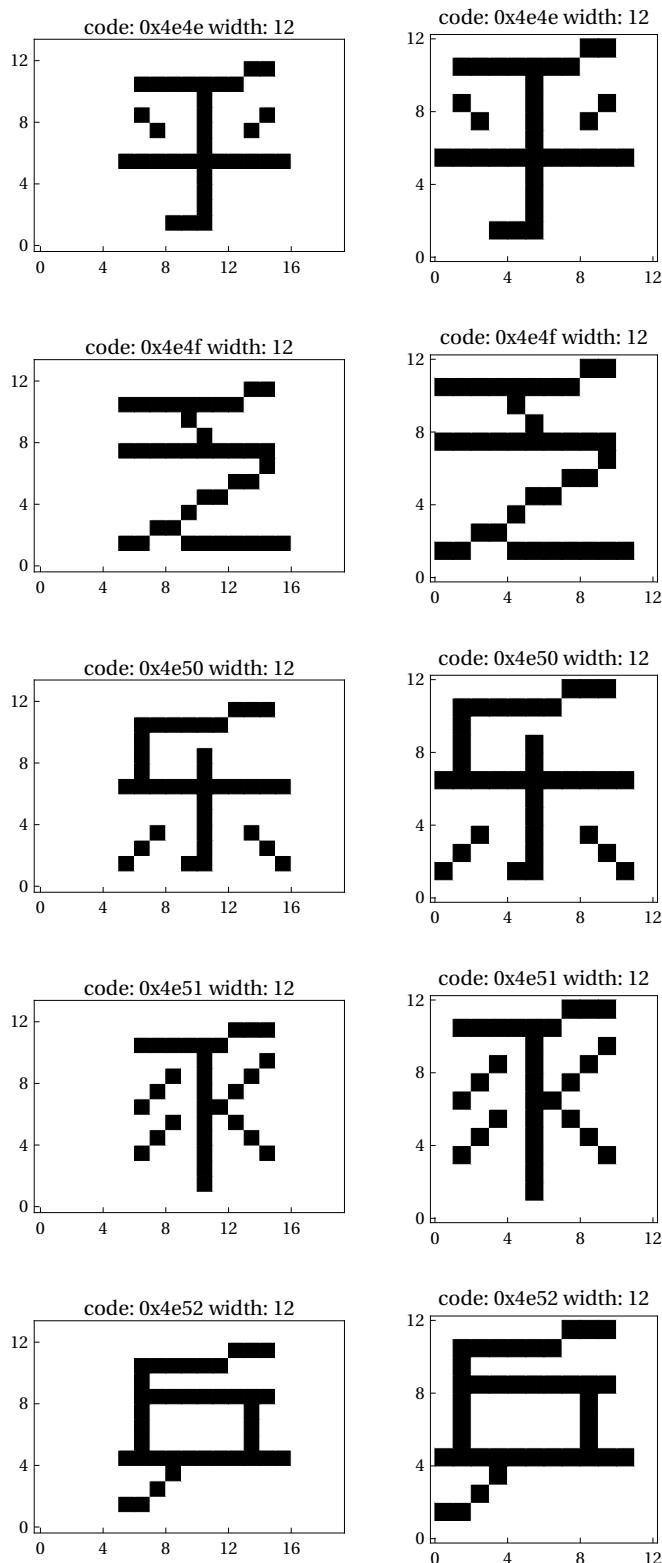


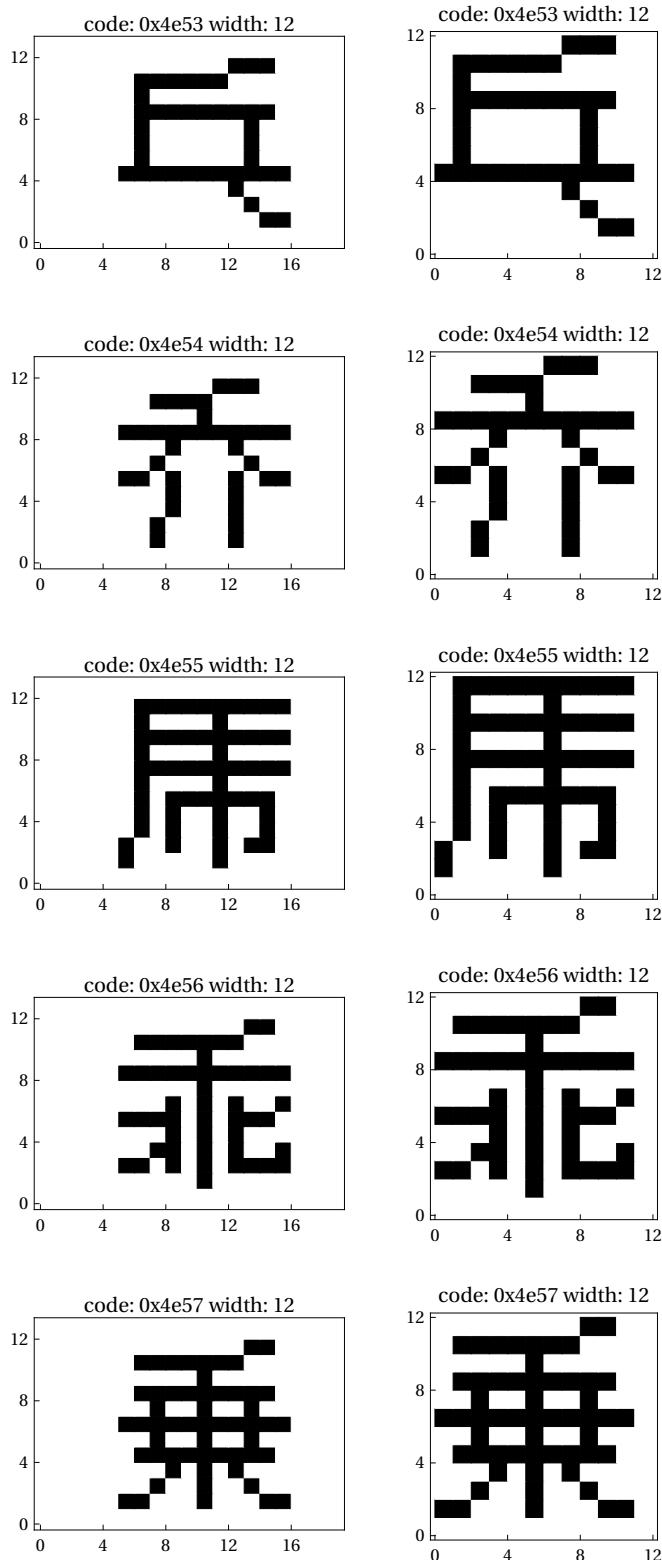


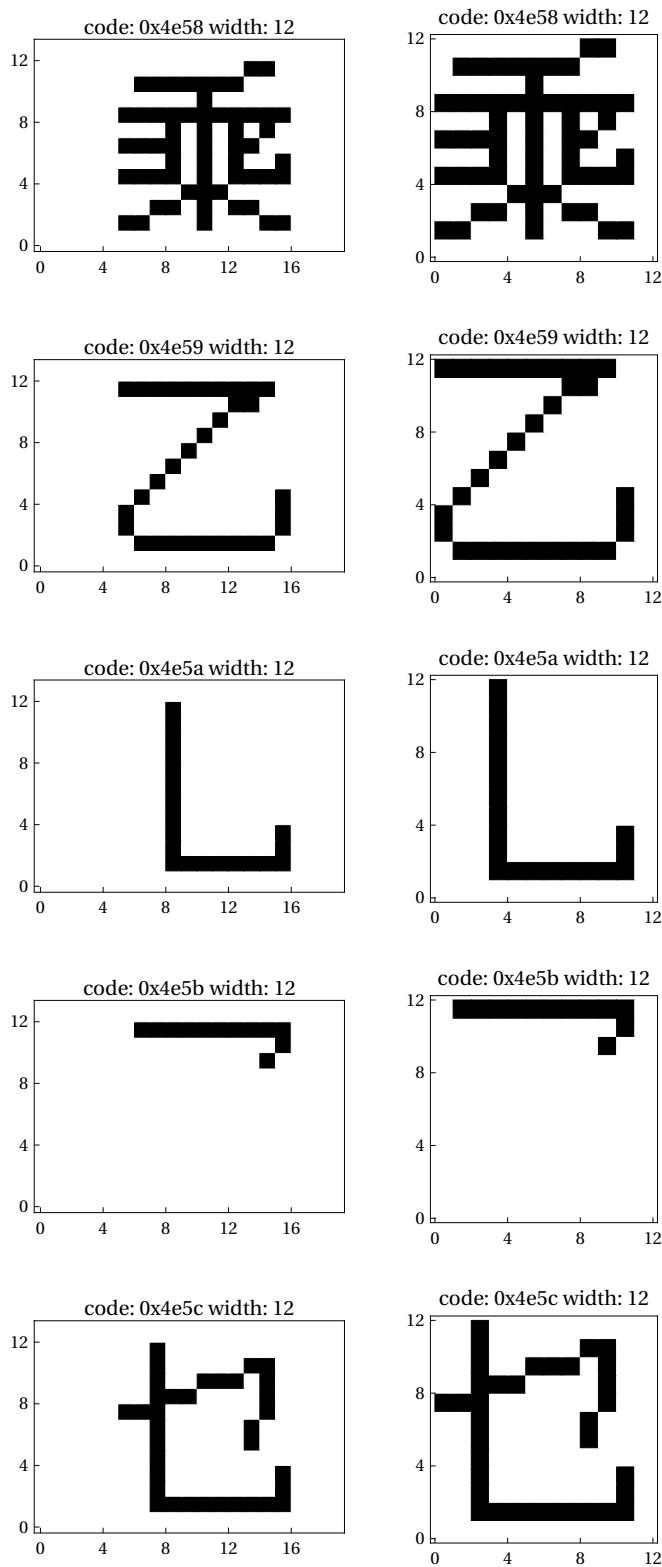


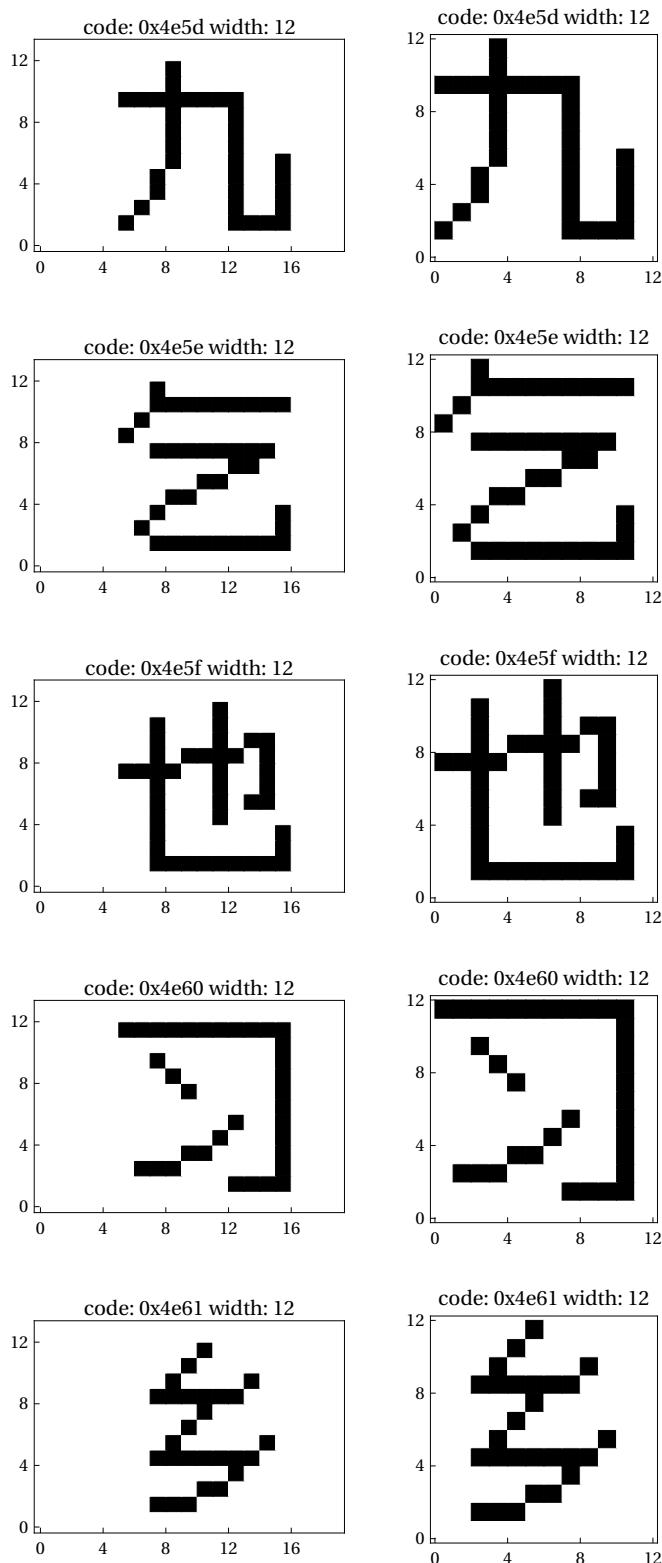


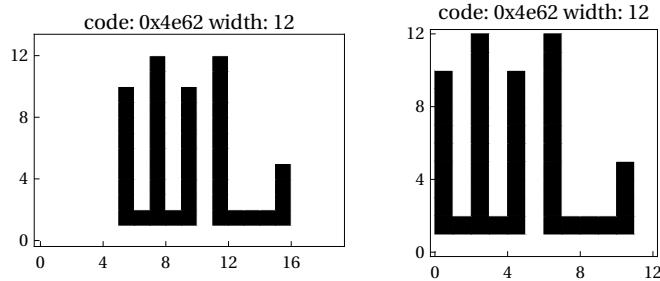












## Extract and sort glyphs by Table of General Standard Chinese Characters

```
In[50]:= Take[rawunicodetable, 16]
```

```
Out[50]= {#, Table, of, General, Standard, Chinese, Characters, mapped, to, Unicode}, {index, number, codepoint}, {1, U+4E00}, {2, U+4E59}, {3, U+4E8C}, {4, U+5341}, {5, U+4E01}, {6, U+5382}, {7, U+4E03}, {8, U+535C}, {9, U+516B}, {10, U+4EBA}, {11, U+5165}, {12, U+513F}, {13, U+5315}, {14, U+51E0}}
```

### ■ Check if the TGSCC is contiguous

```
In[51]:= {Max[#, Min[#]} &[ListConvolve[{1, -1}, First /@ Drop[rawunicodetable, 2]]]]
```

```
Out[51]= {1, 1}
```

Yes, it is.

```
In[52]:= Take[TGSCCcodes = ToExpression[StringReplace[Last[#], "U+" → "16^&"]] & /@ Drop[rawunicodetable, 2], 16]
```

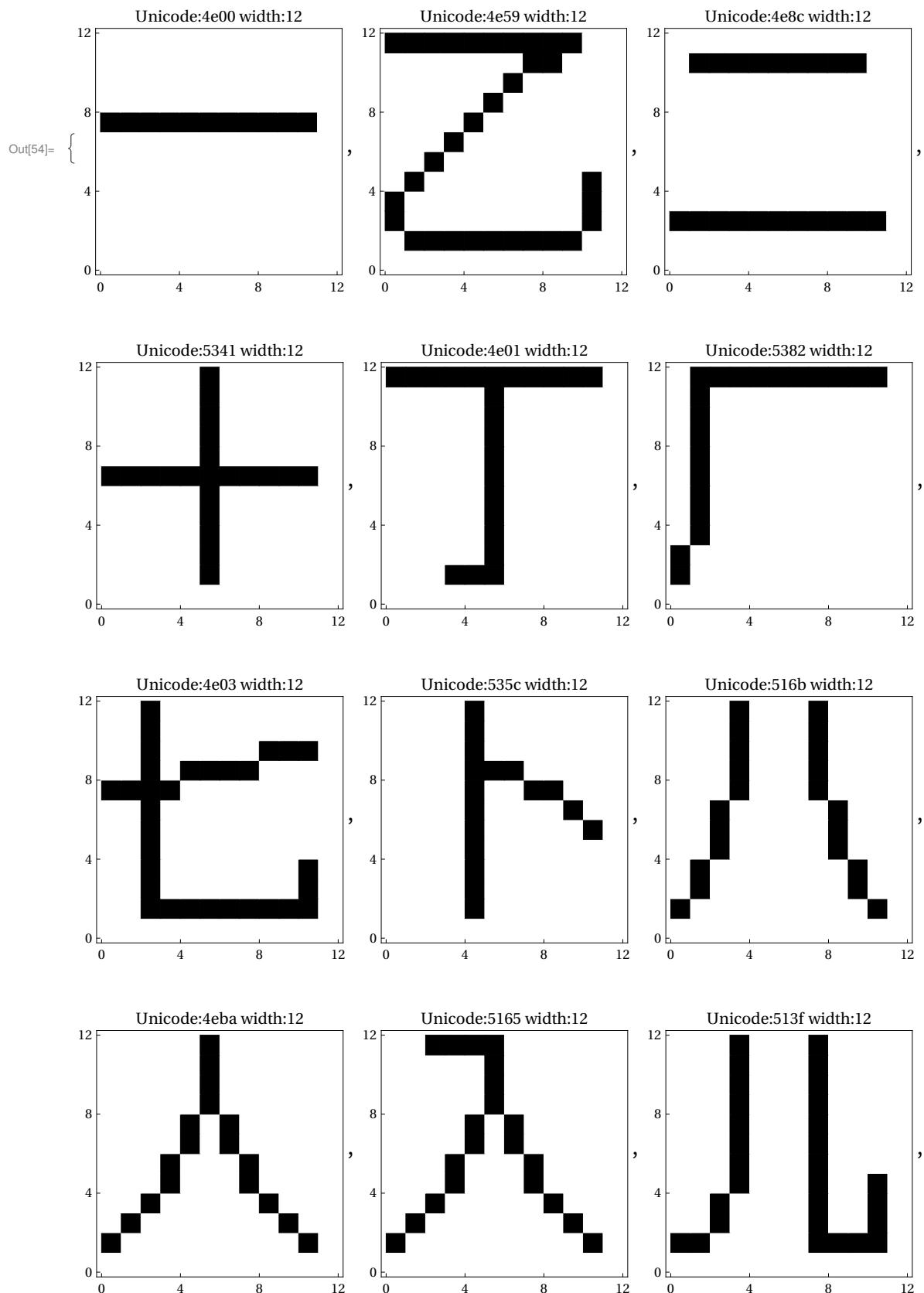
```
Out[52]= {19 968, 20 057, 20 108, 21 313, 19 969, 21 378, 19 971, 21 340, 20 843, 20 154, 20 837, 20 799, 21 269, 20 960, 20 061, 20 993}
```

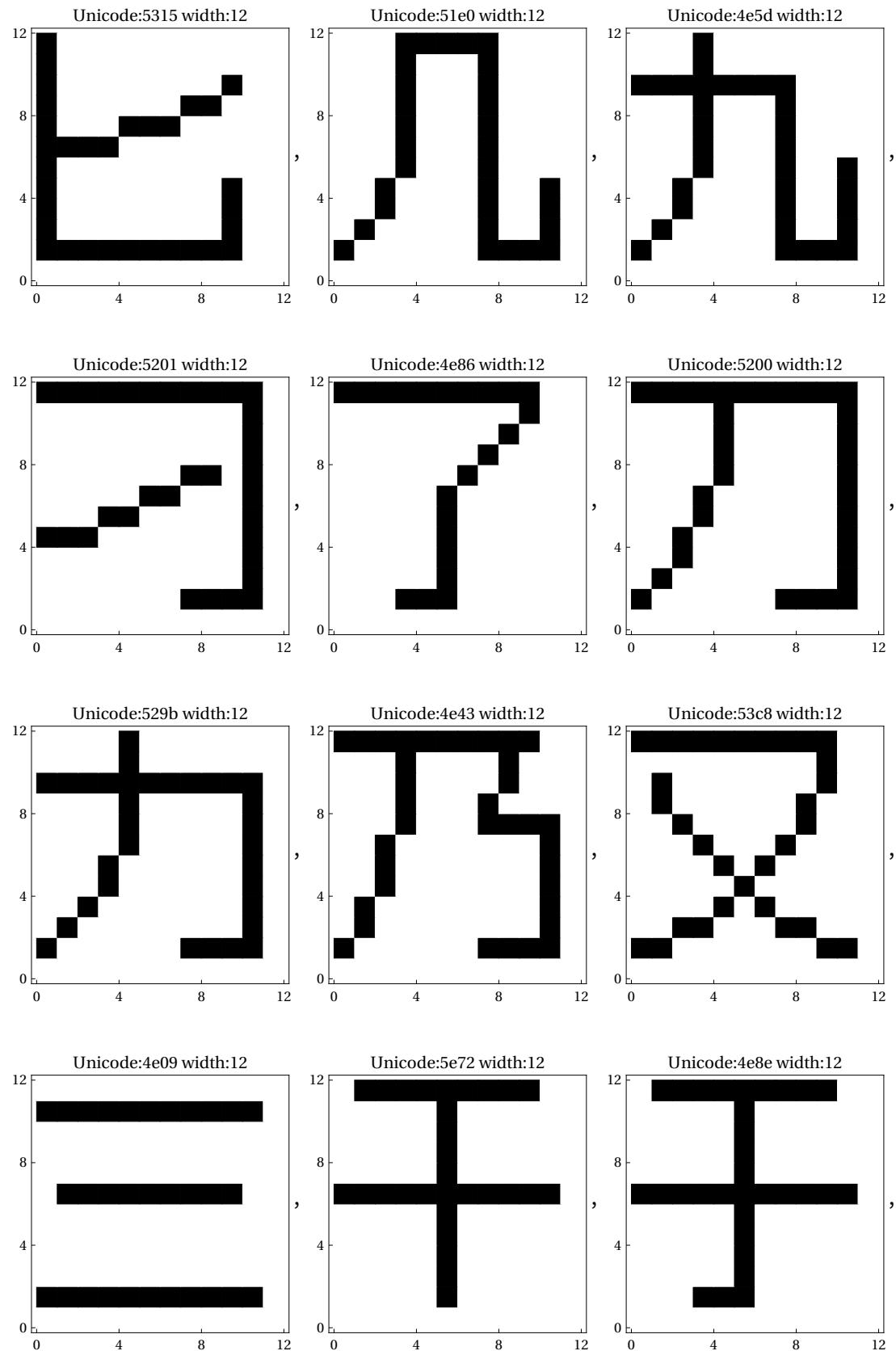
```
In[53]:= Dimensions[
```

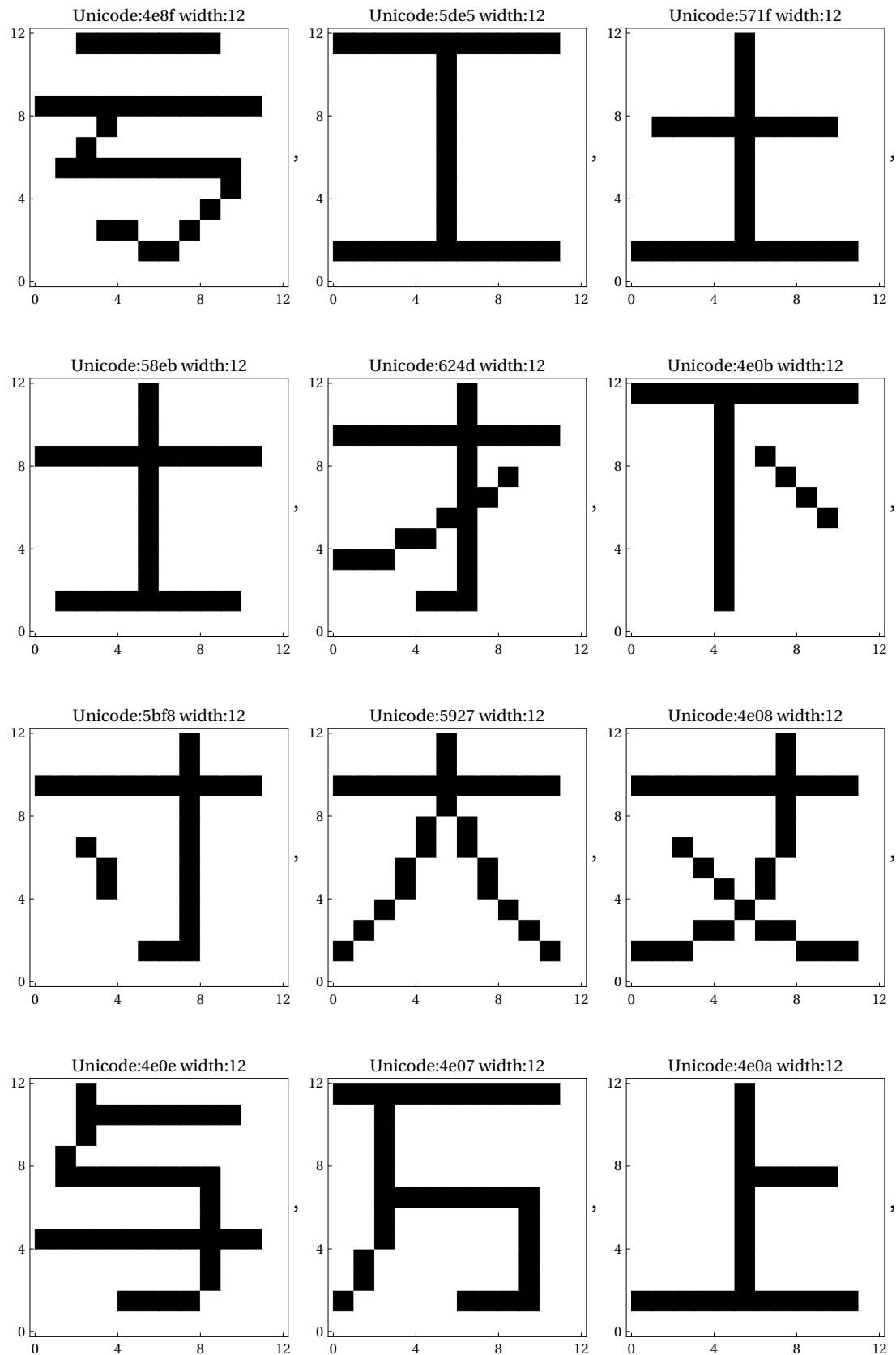
```
lookuptable = Association[MapThread[Function[{code, trbitmap, extbitmap, width}, ToExpression[code] → {trbitmap, extbitmap, width}], {charcodes, trbitmaps, extbitmaps, widths}]]
```

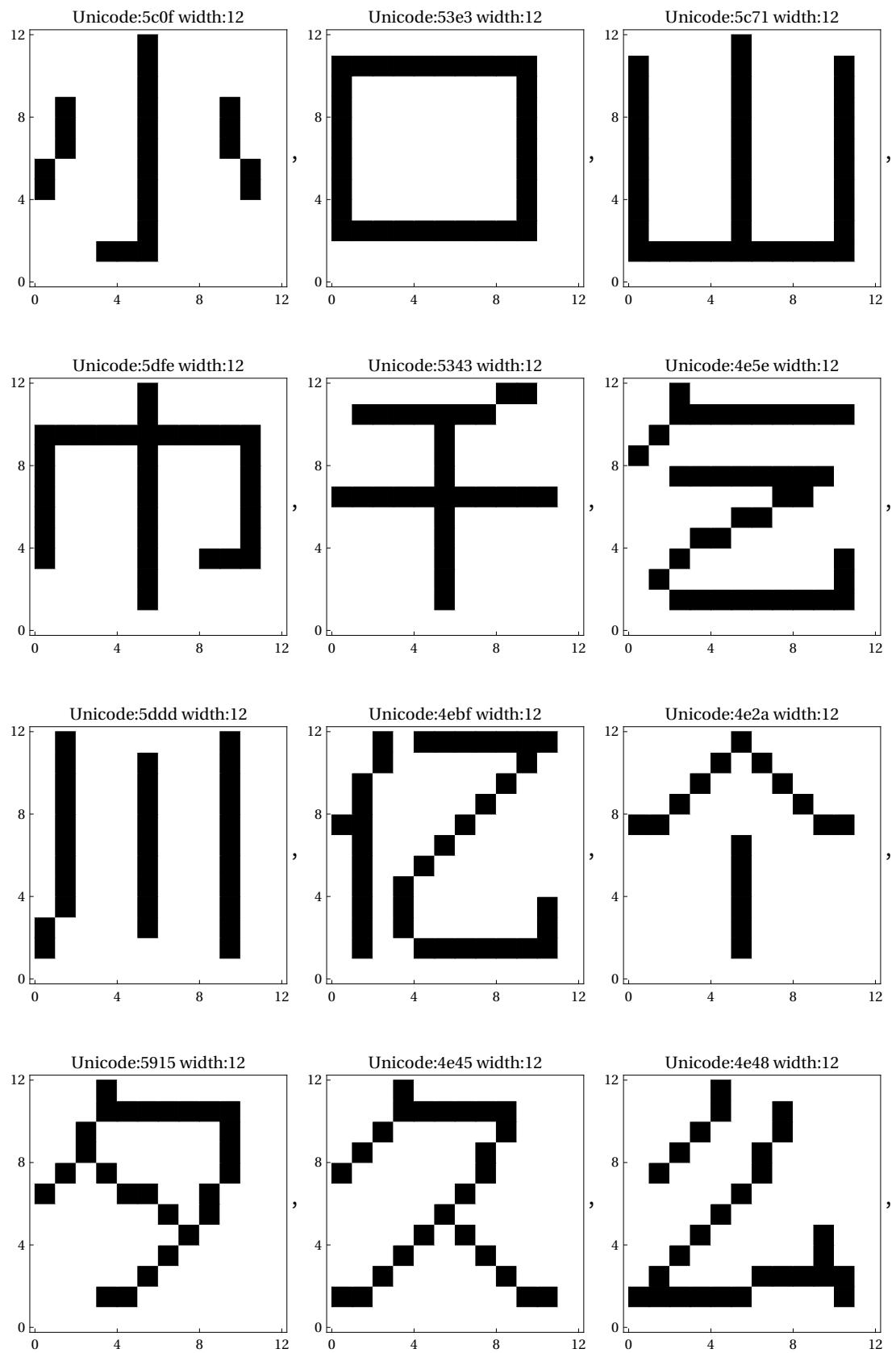
```
Out[53]= {22 043}
```

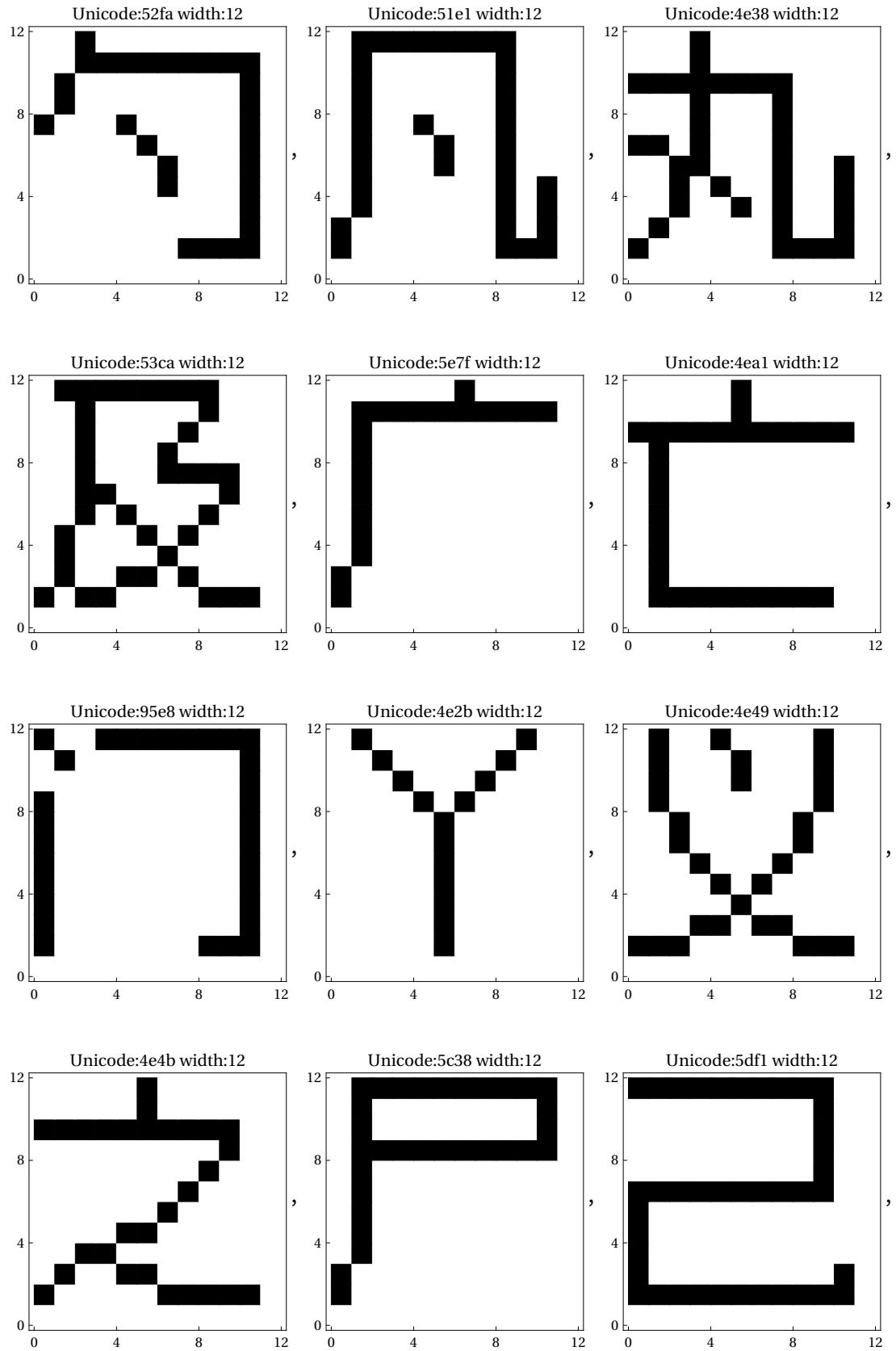
```
In[54]:= Graphics[Raster[Reverse[#1[[2]]]], Frame → True, AspectRatio → Divide @@ Dimensions[#1[[2]]], PlotLabel → "Unicode:" <> IntegerString[#1[[1]], 16, 4] <> " width:" <> ToString[#1[[1]]], FrameTicks → ({#, #, {}, {}} & [Range[0, 16, 4]])] & /@ Take[Prepend[Lookup[lookuptable, #], #] & /@ TGSCCcodes, 1500]
```

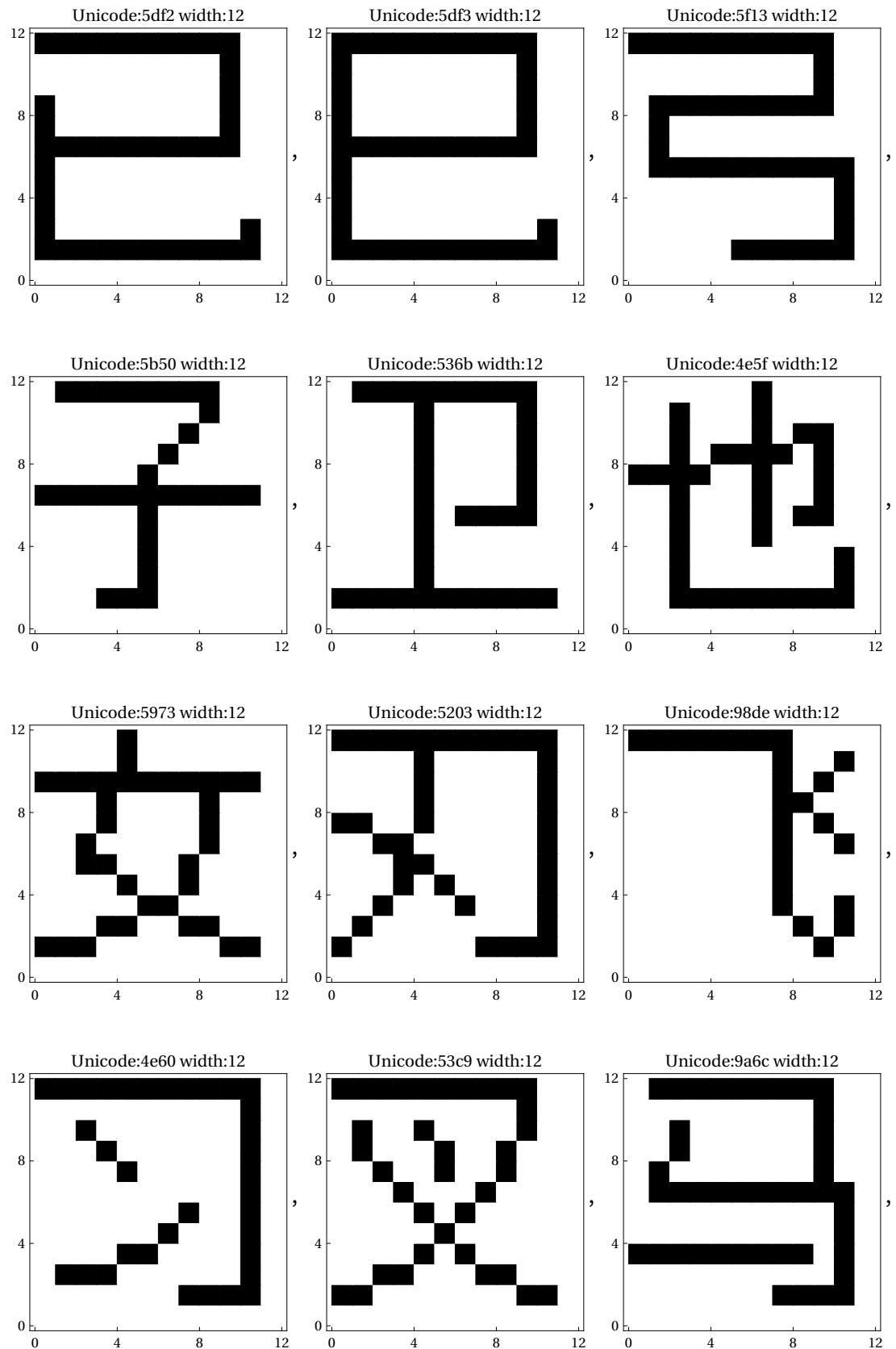


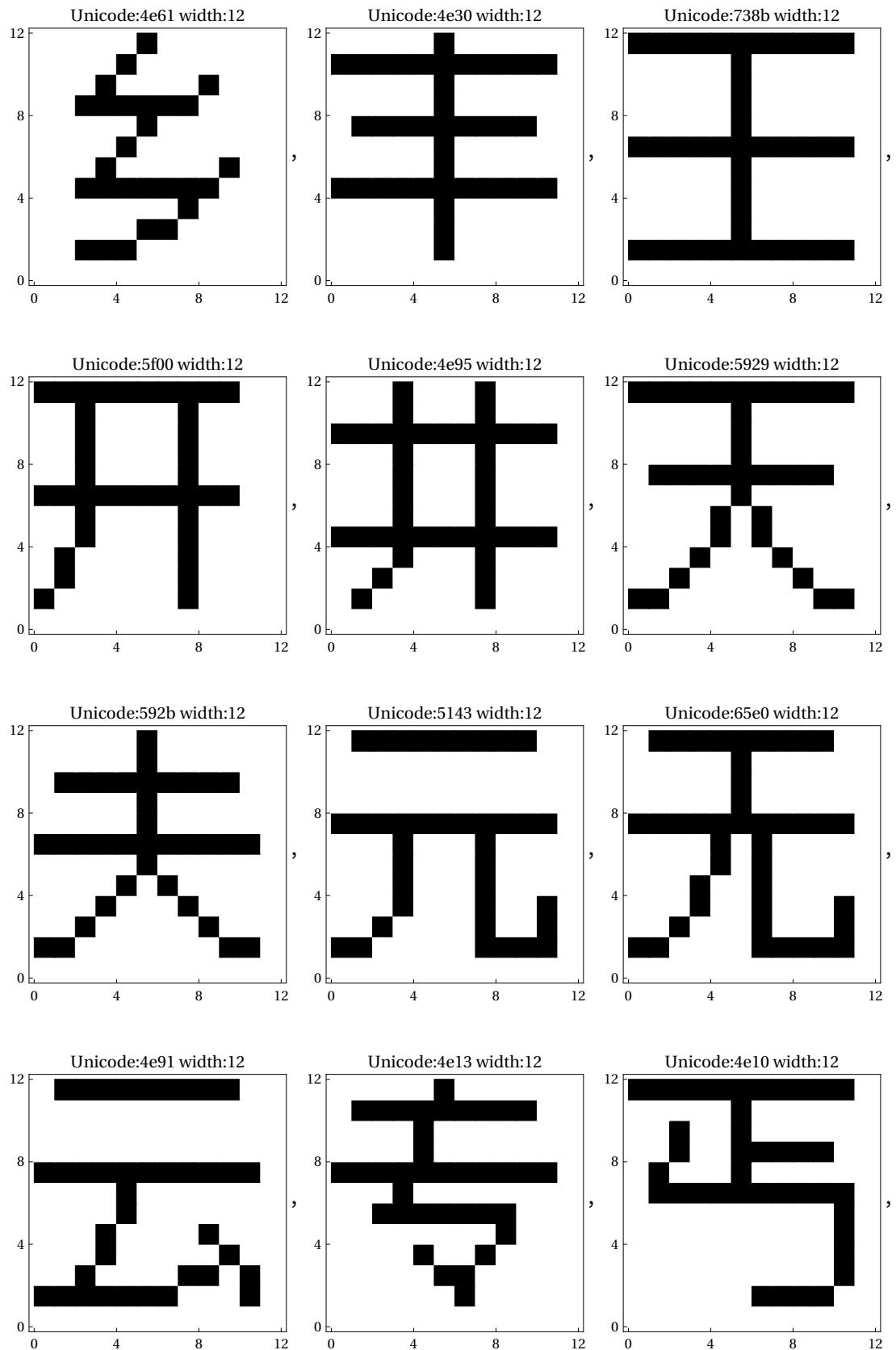


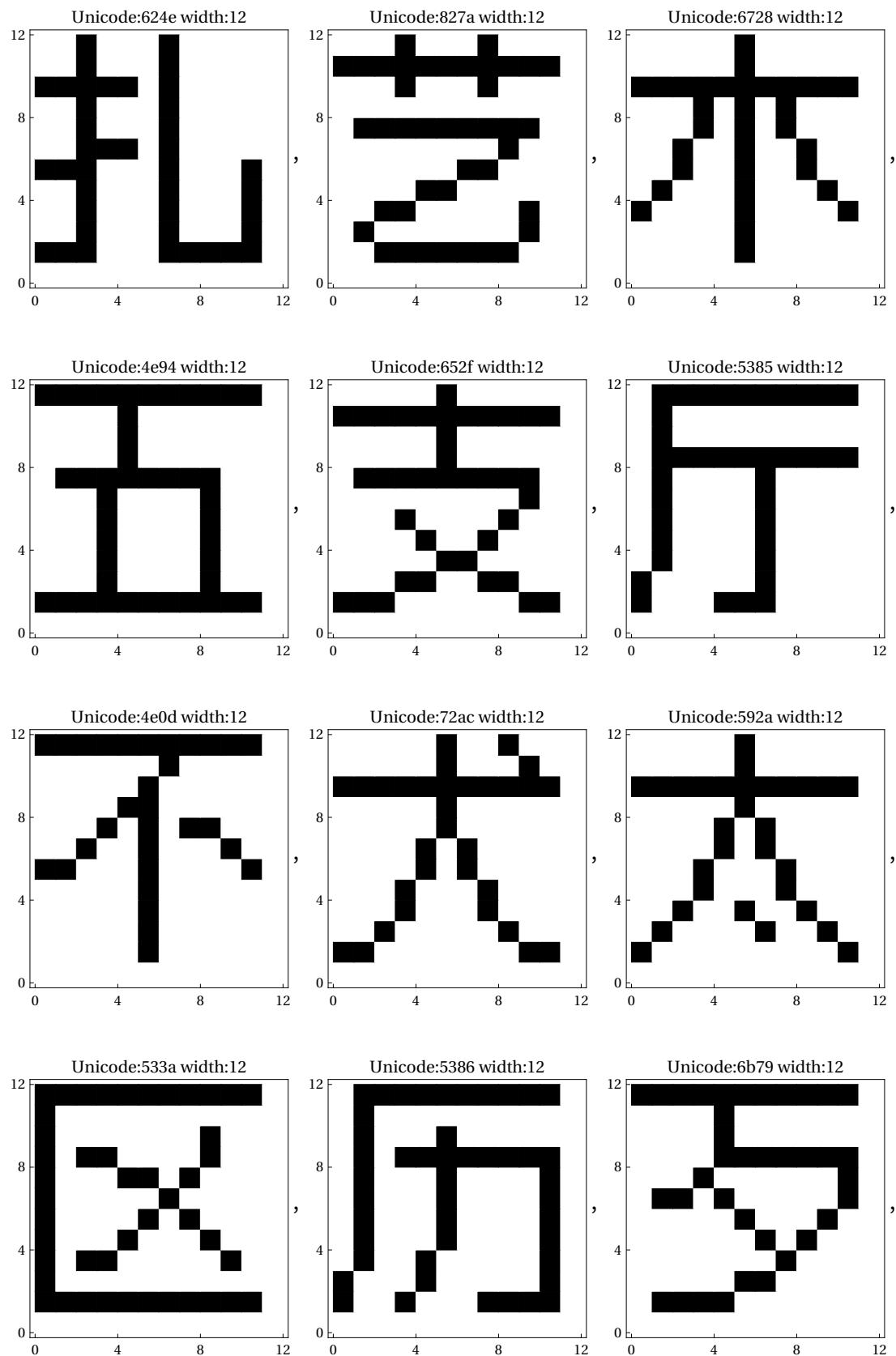


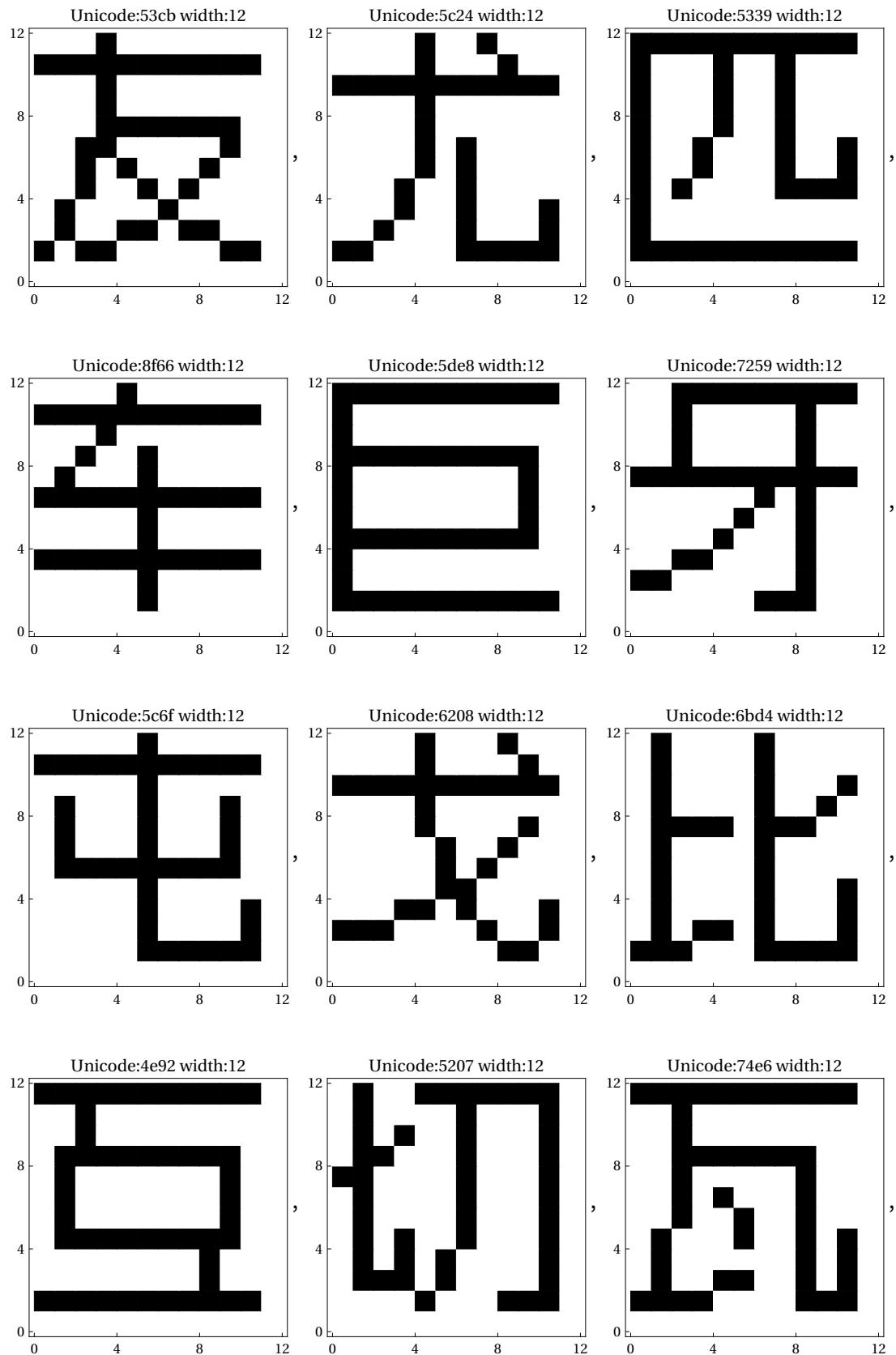


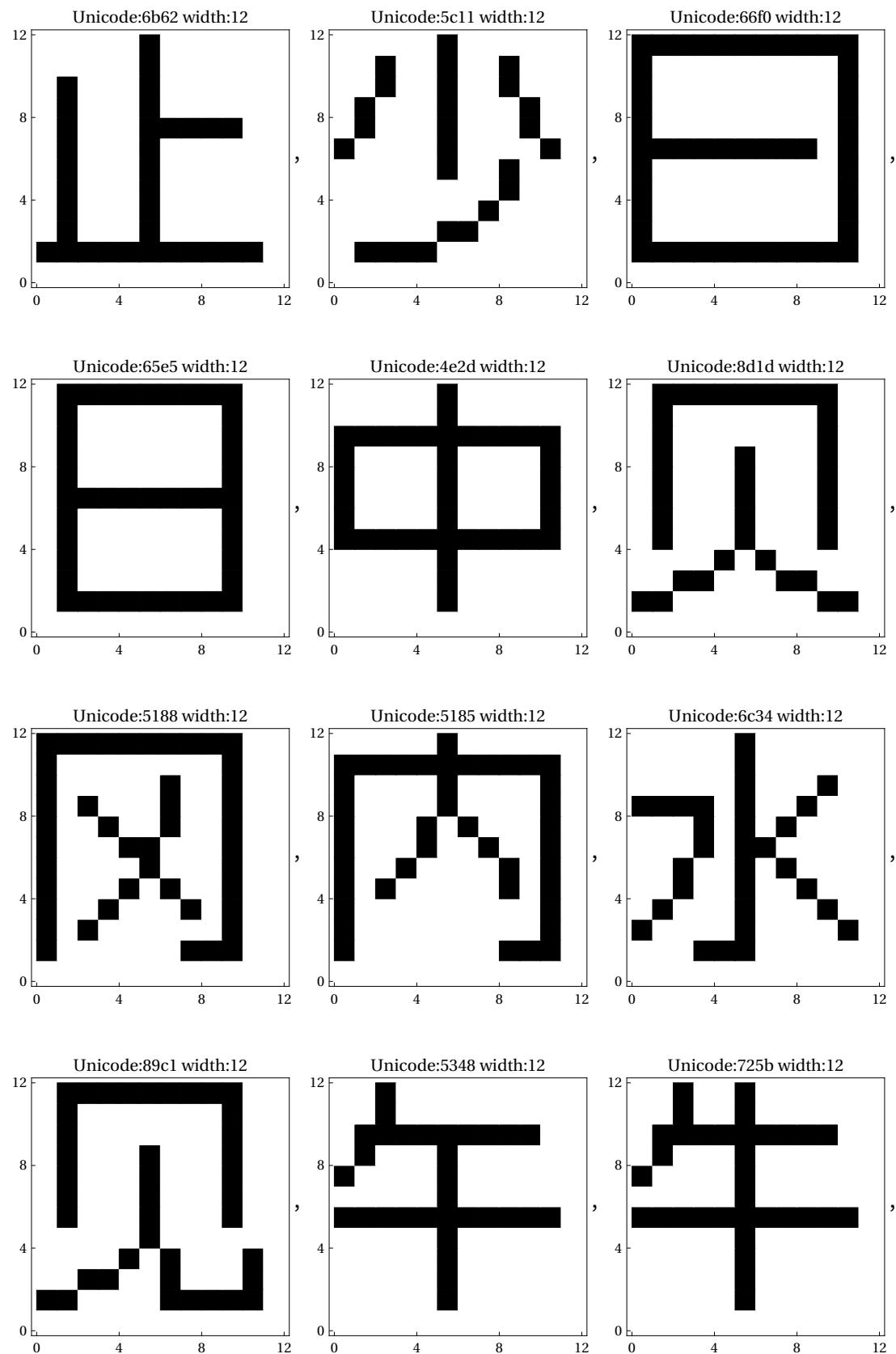


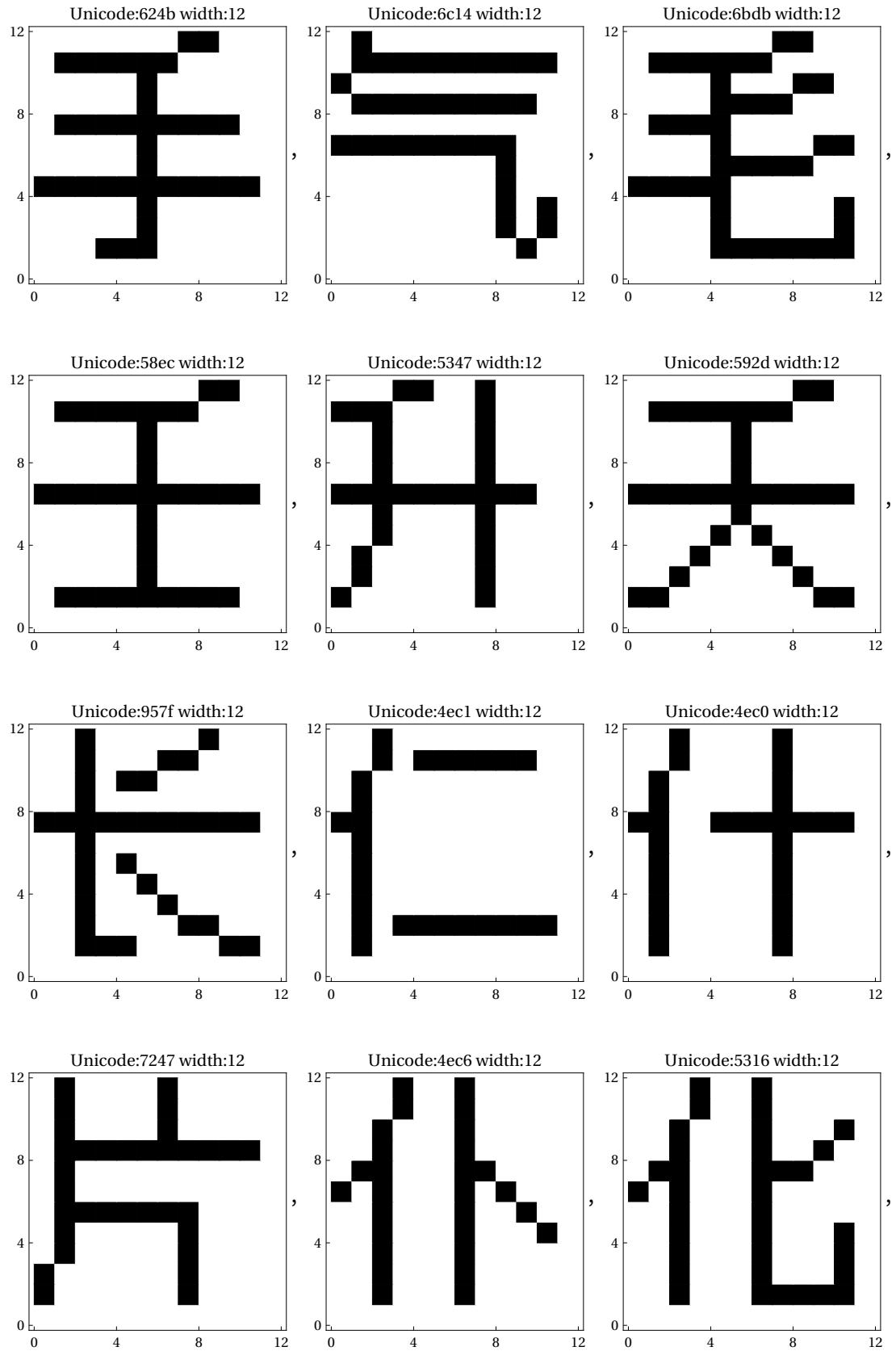


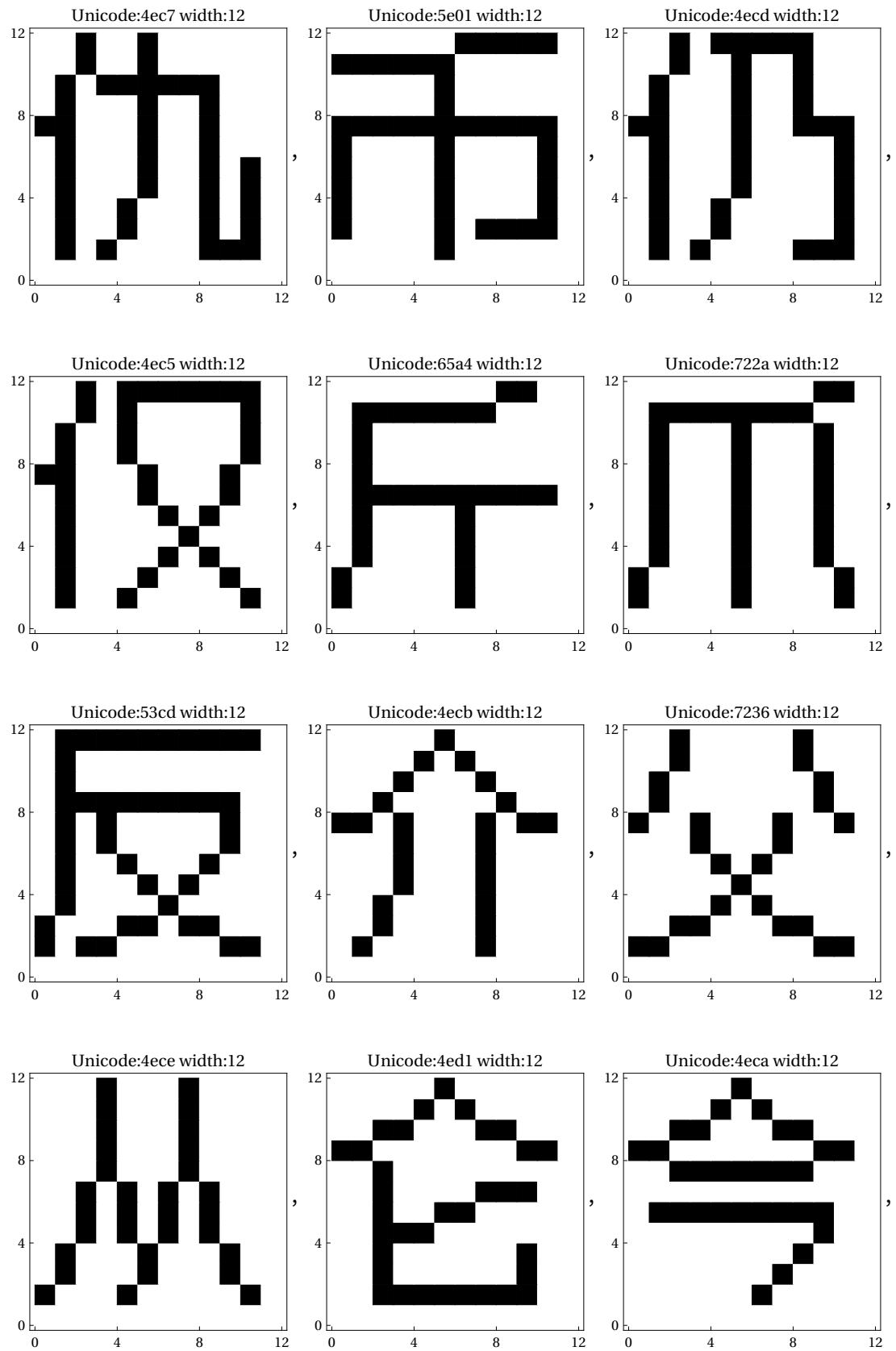


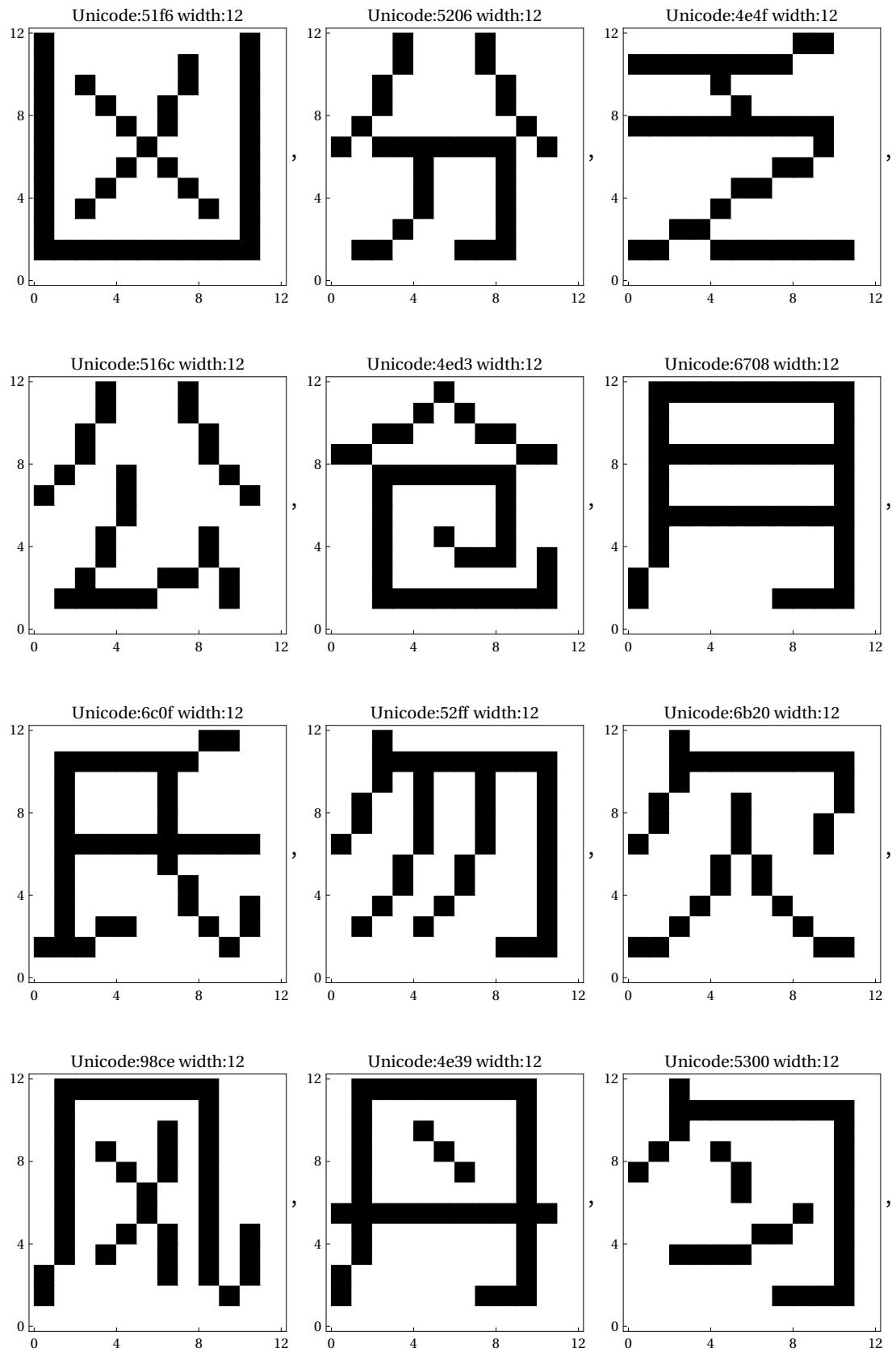


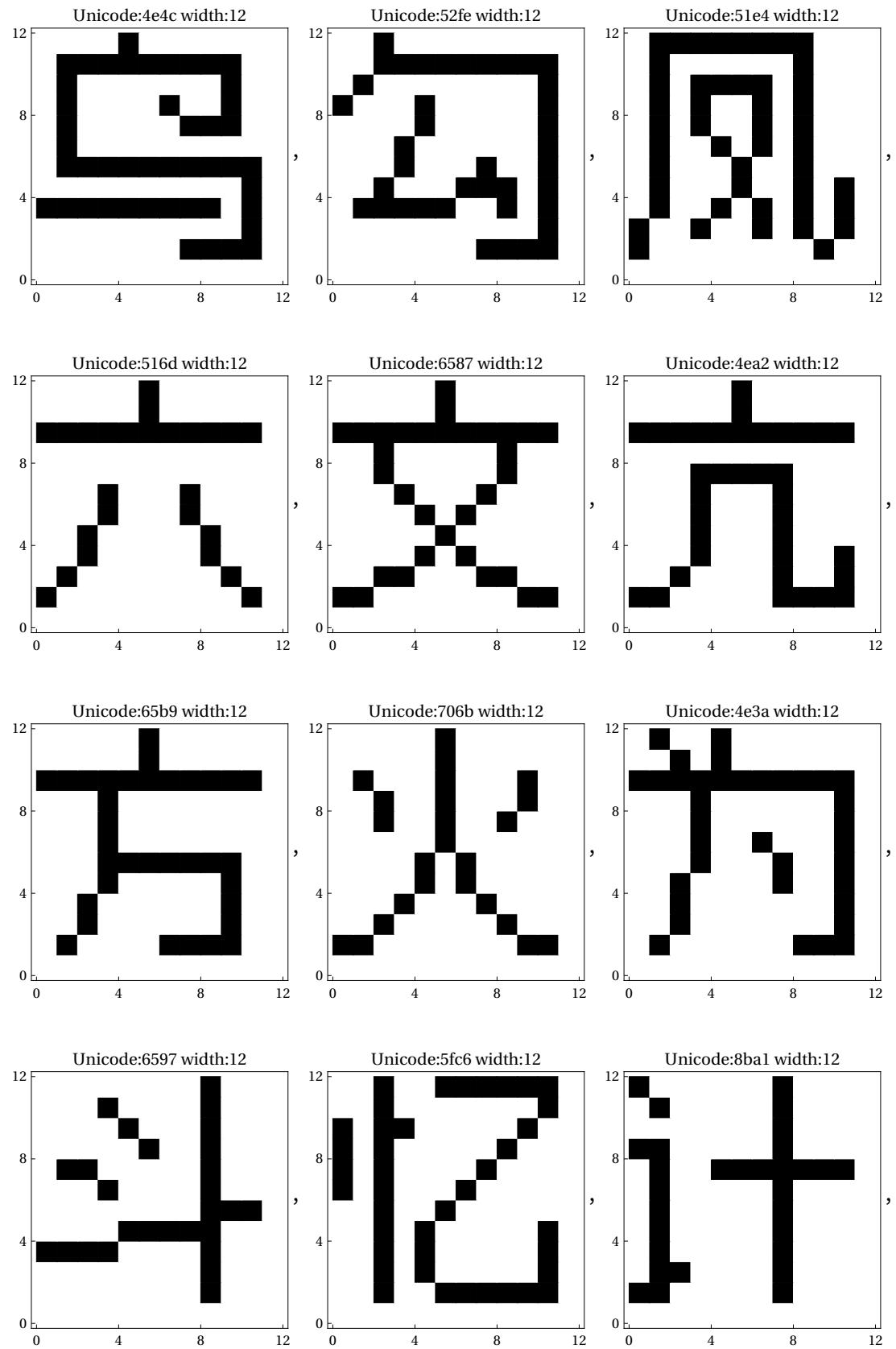


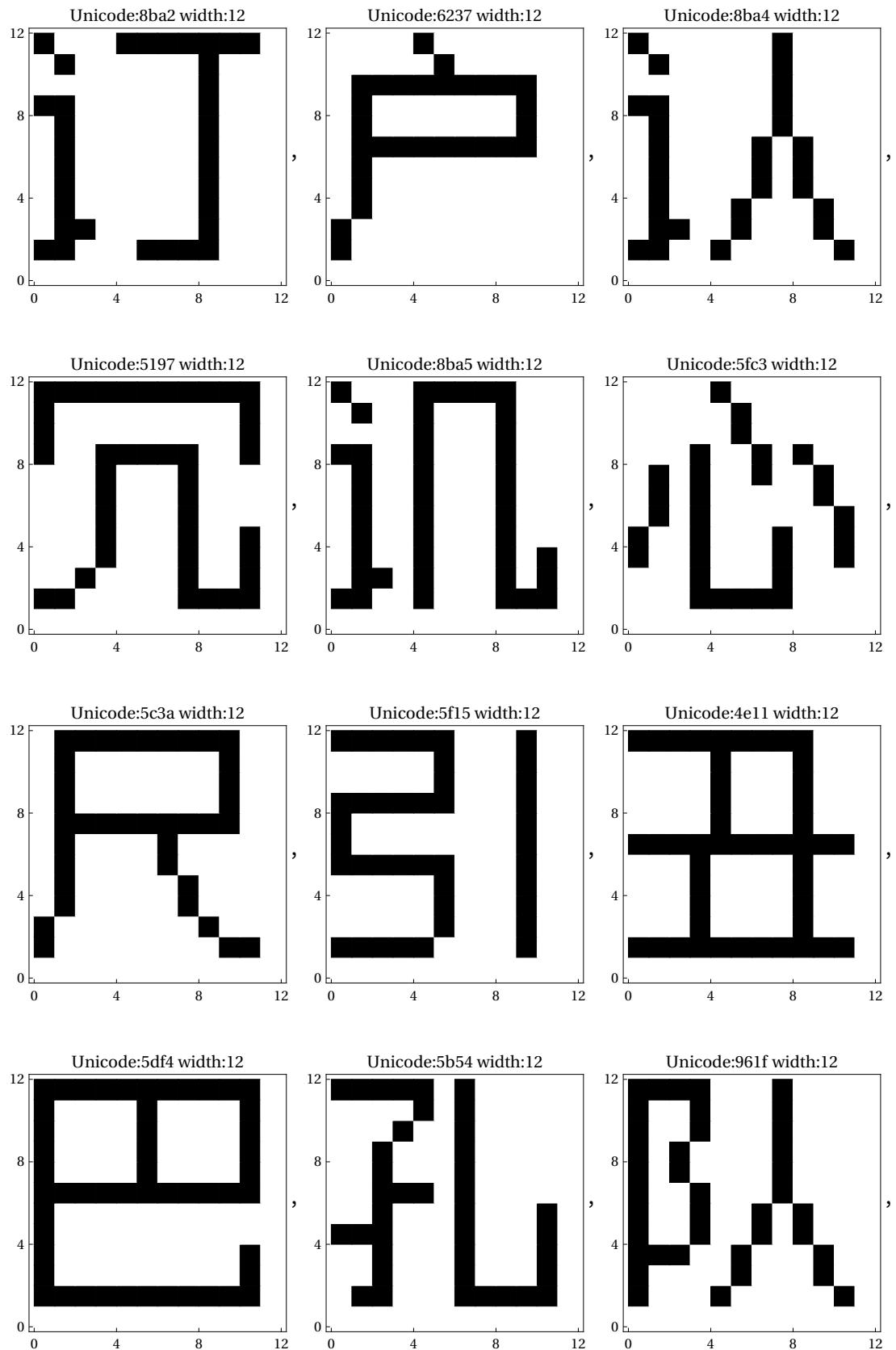


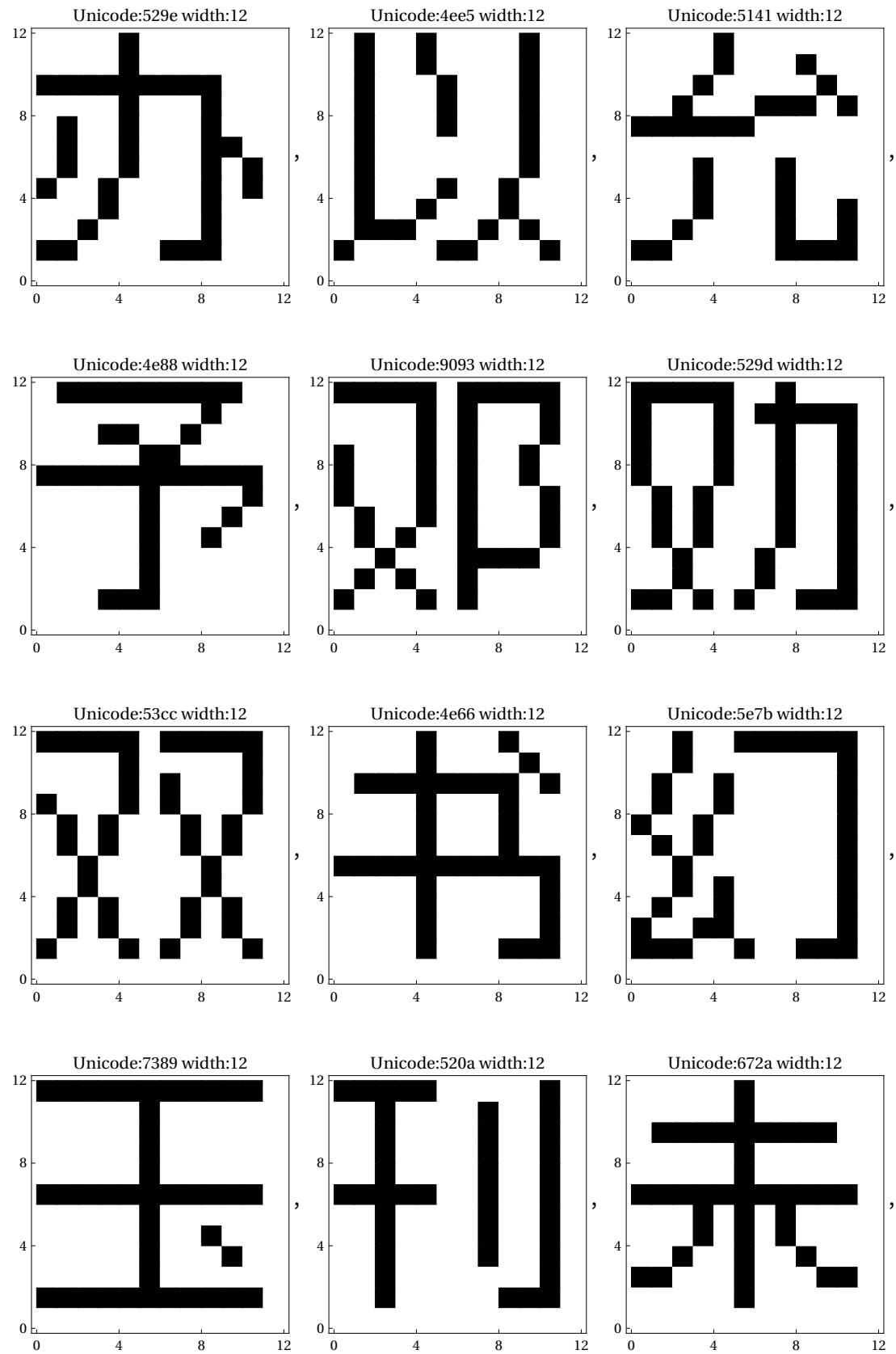


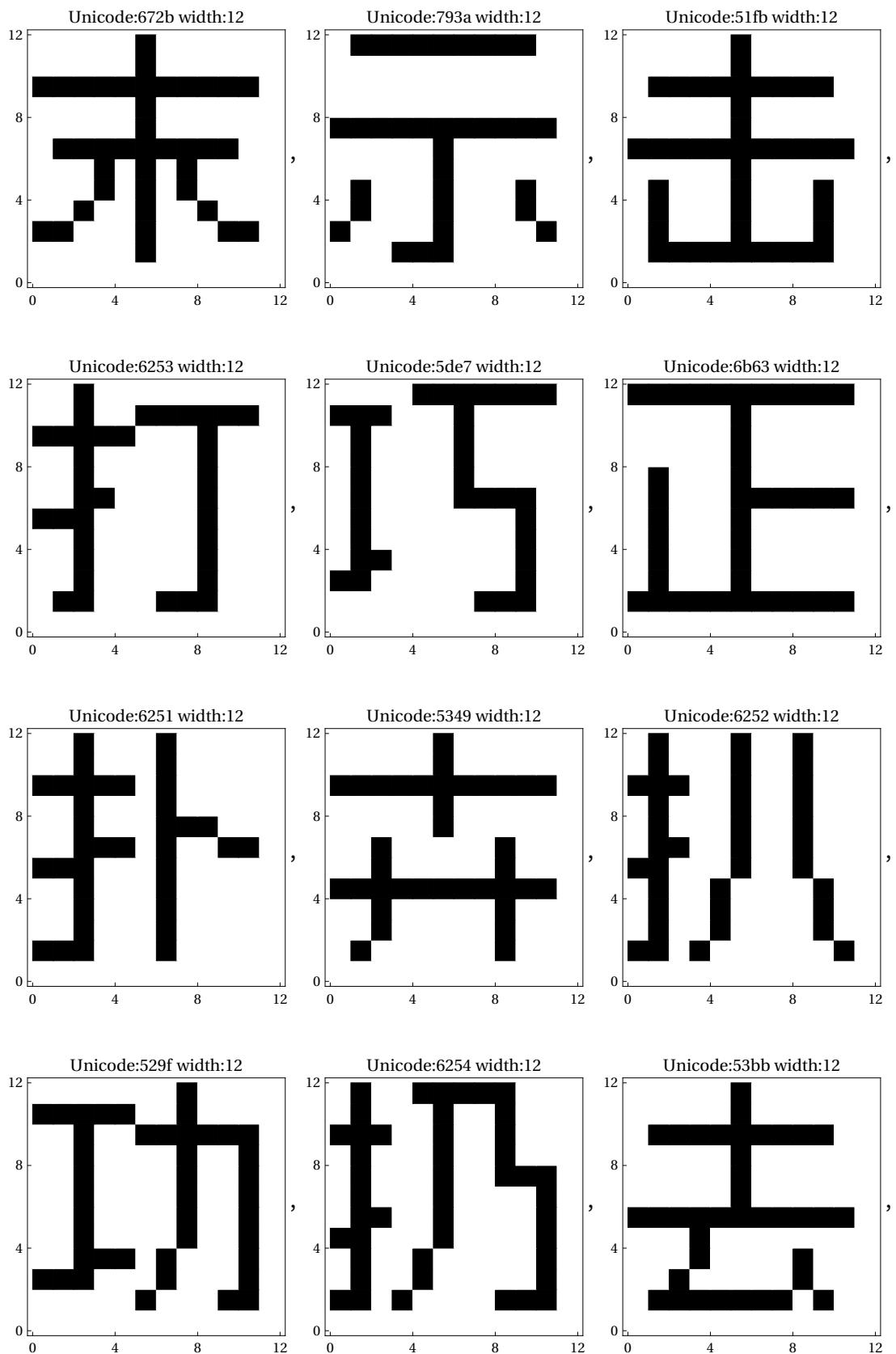


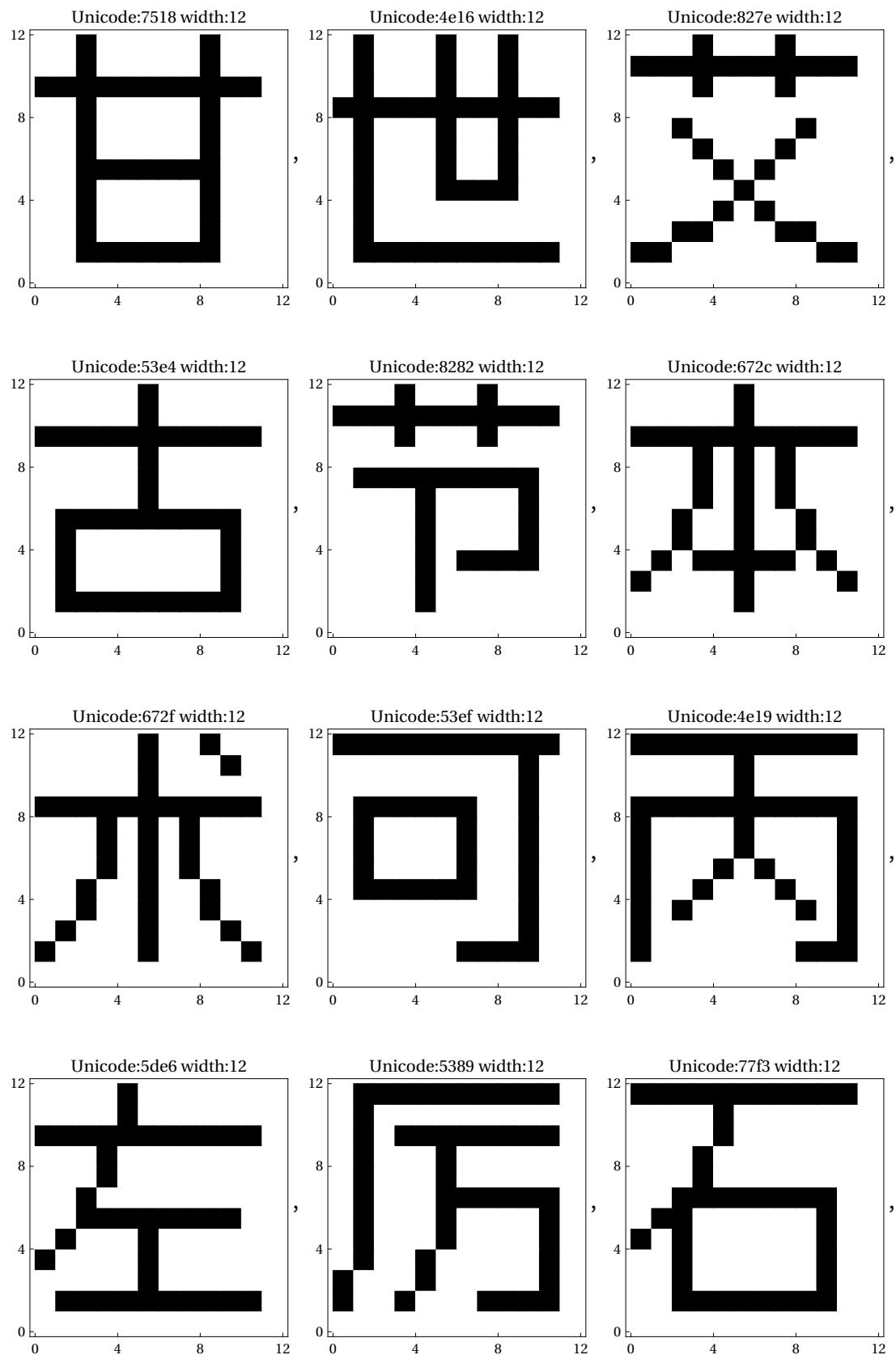


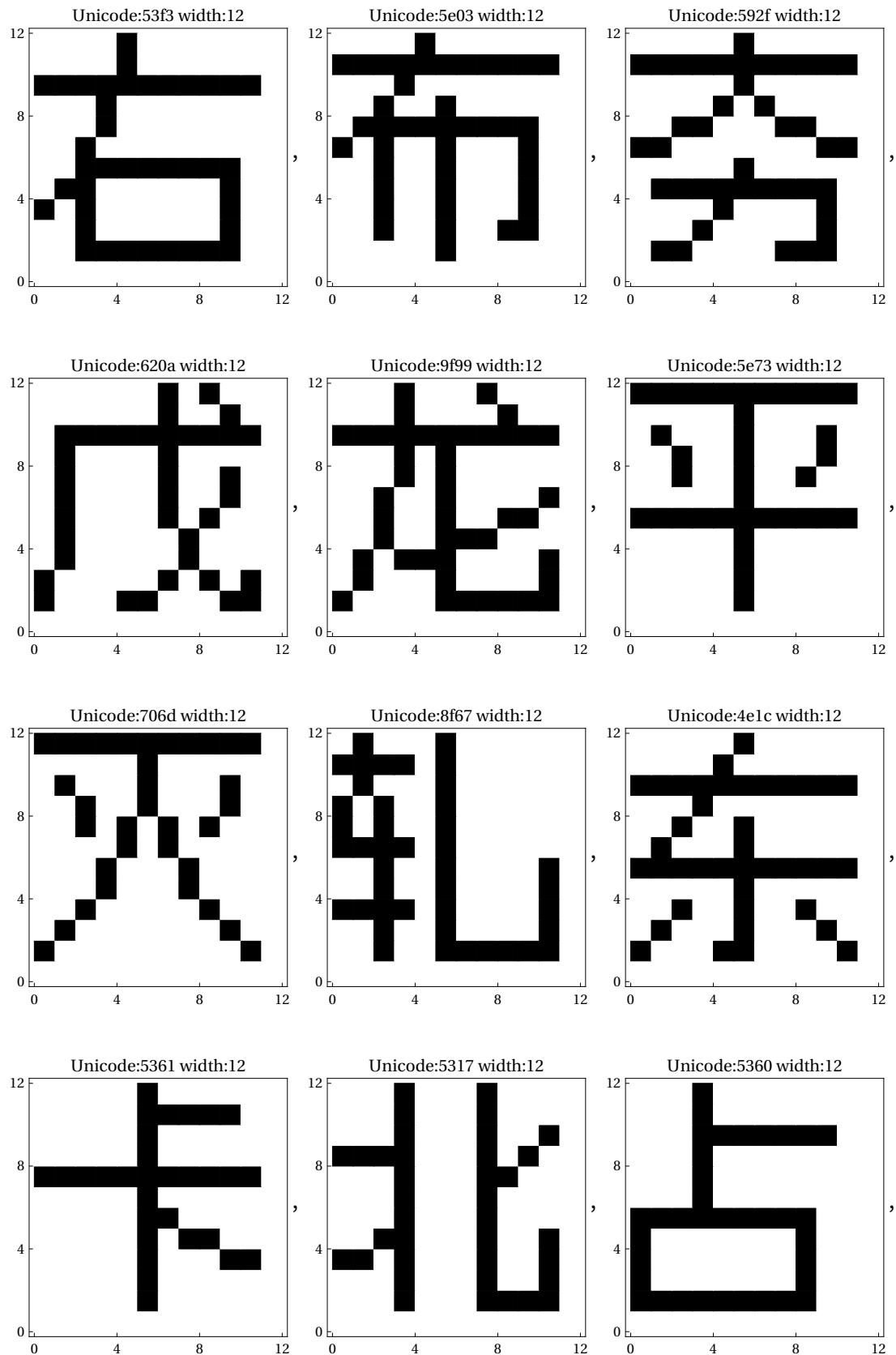


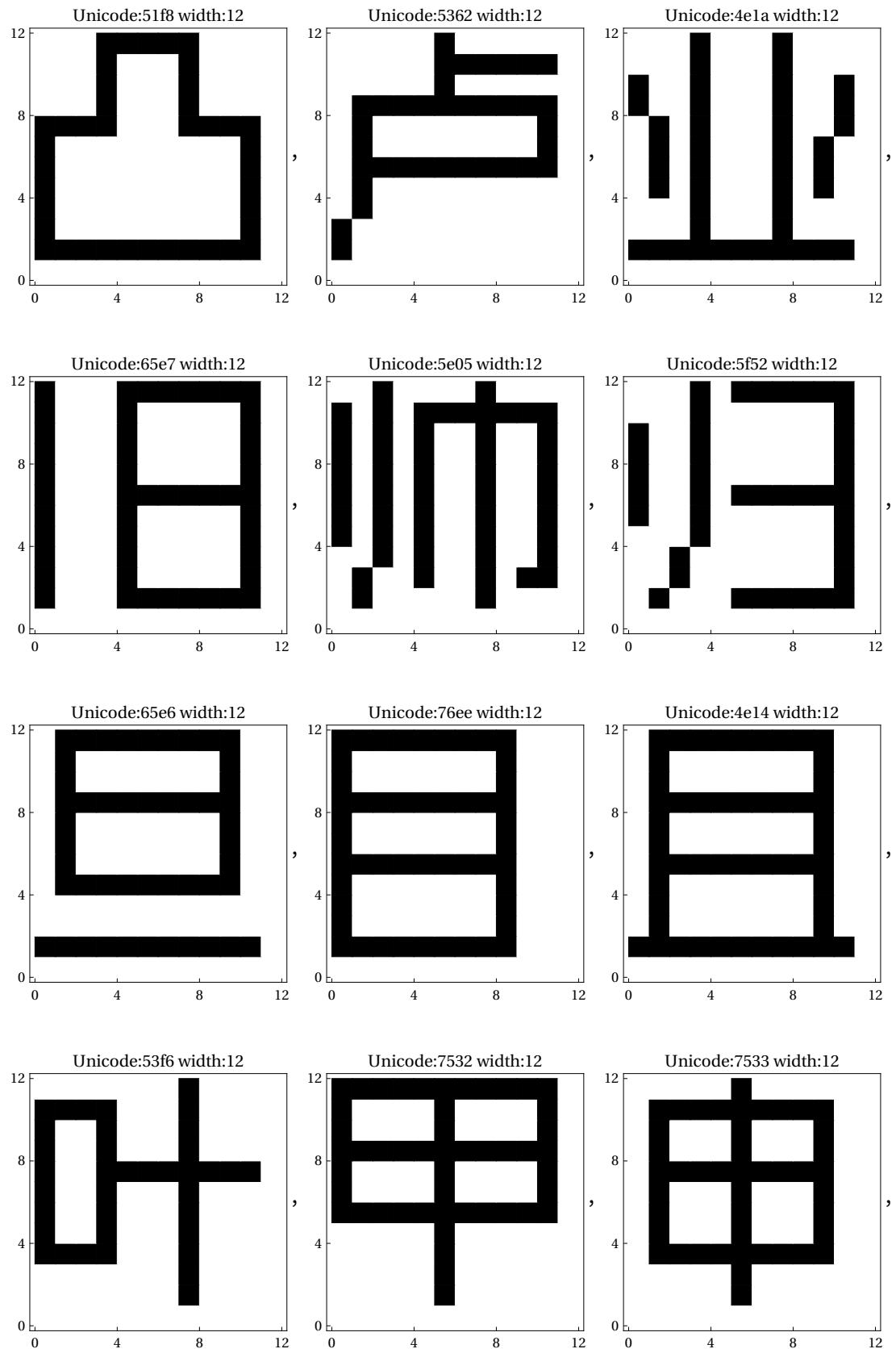


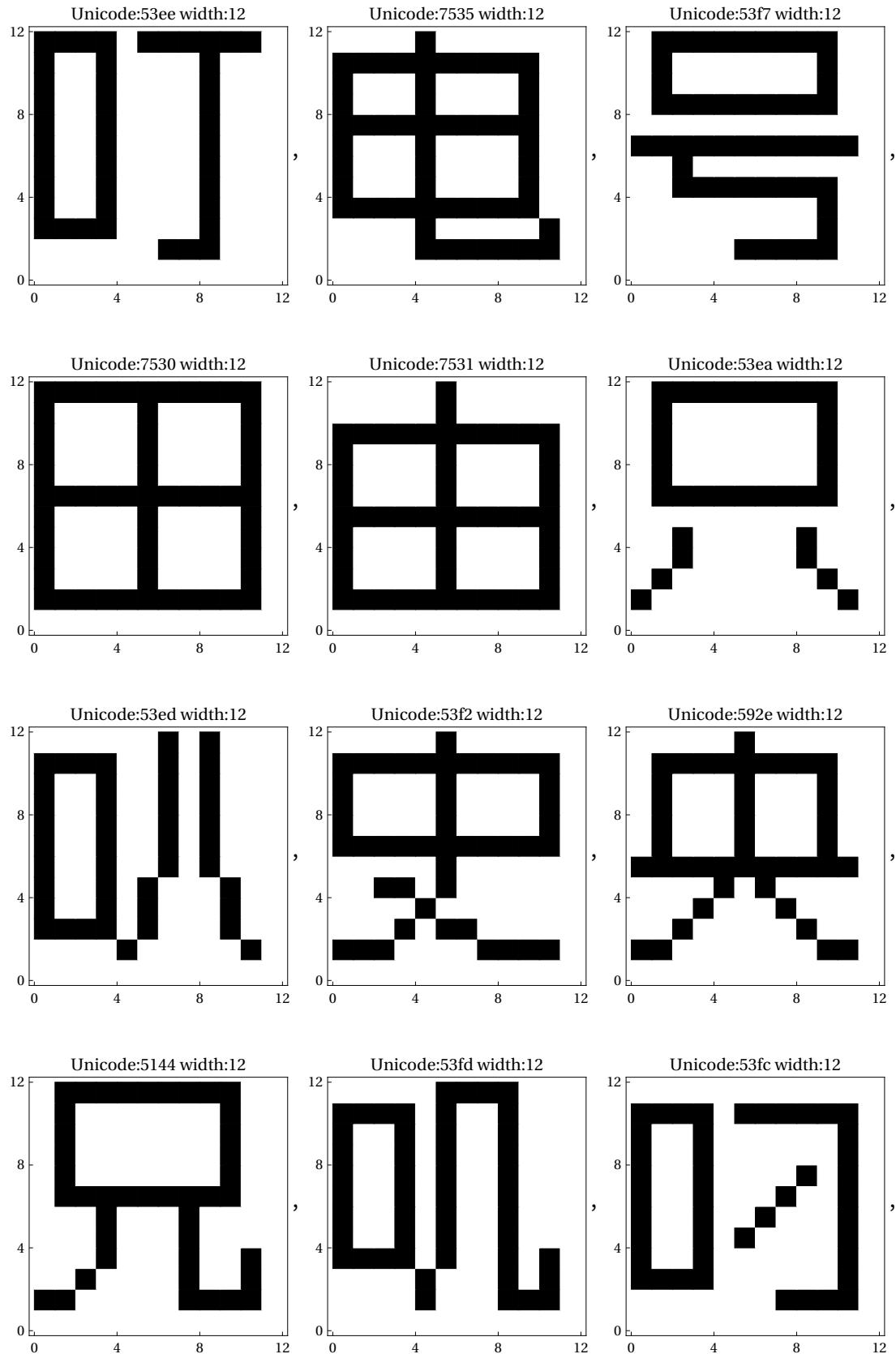


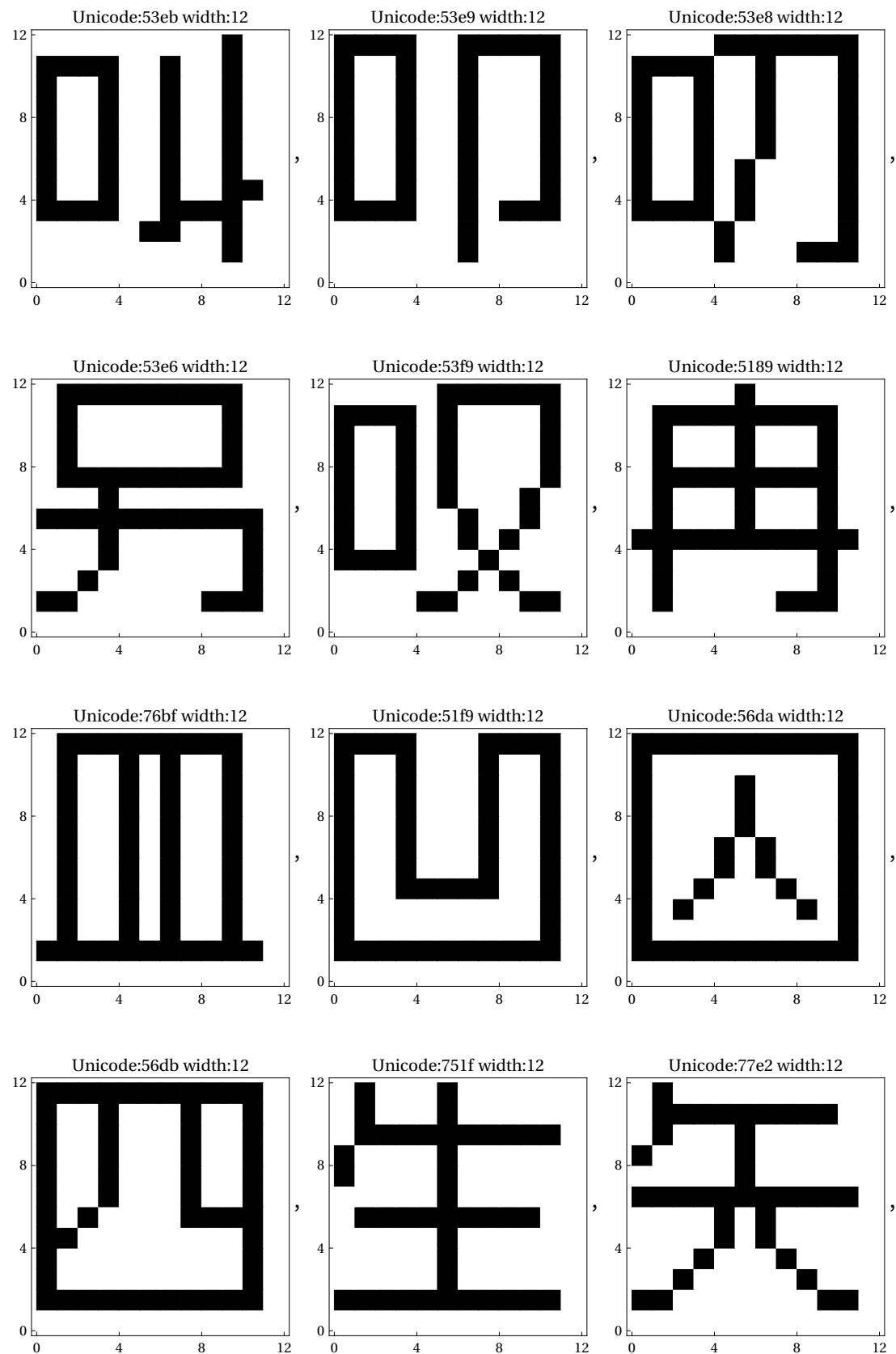


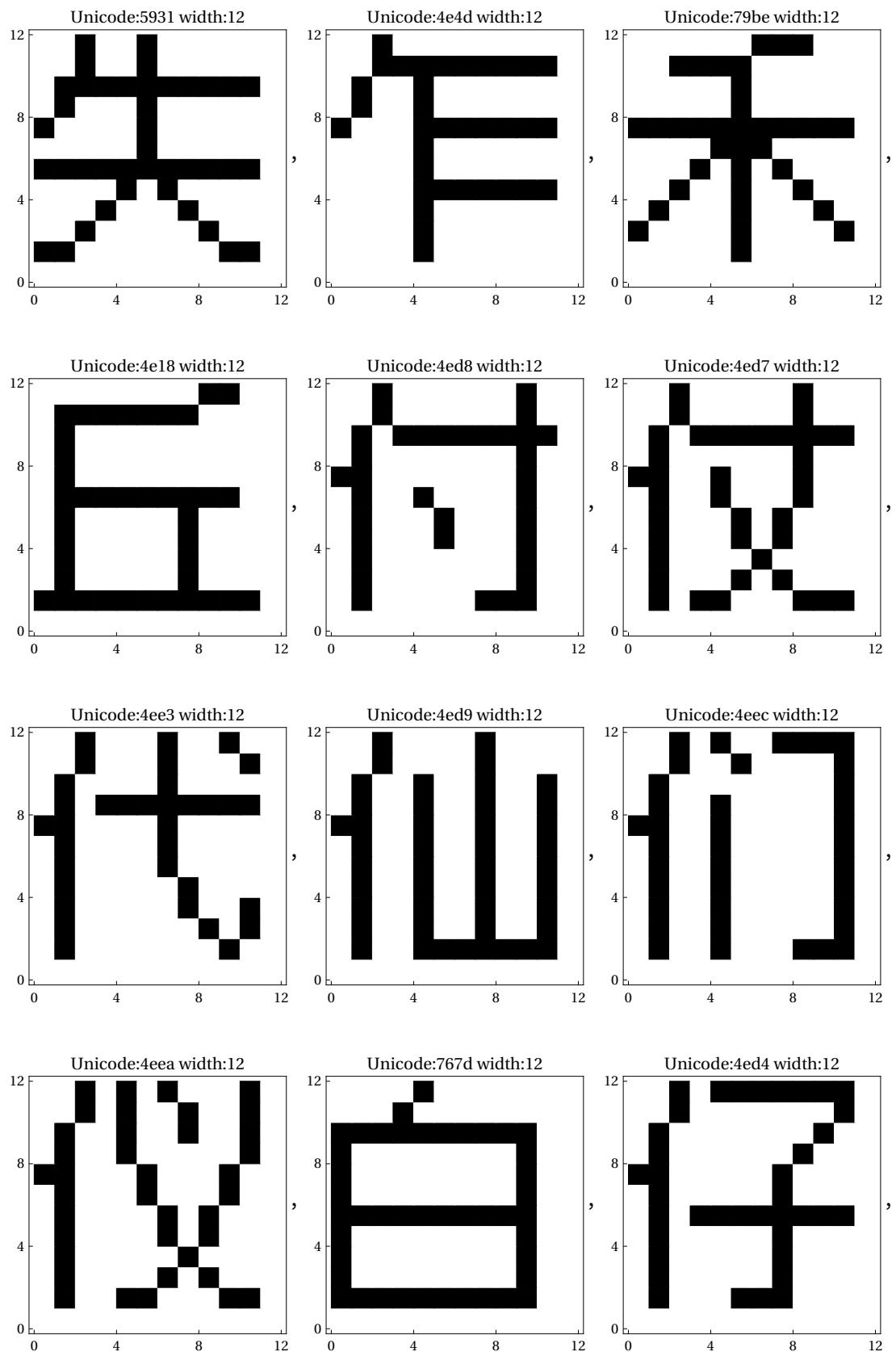


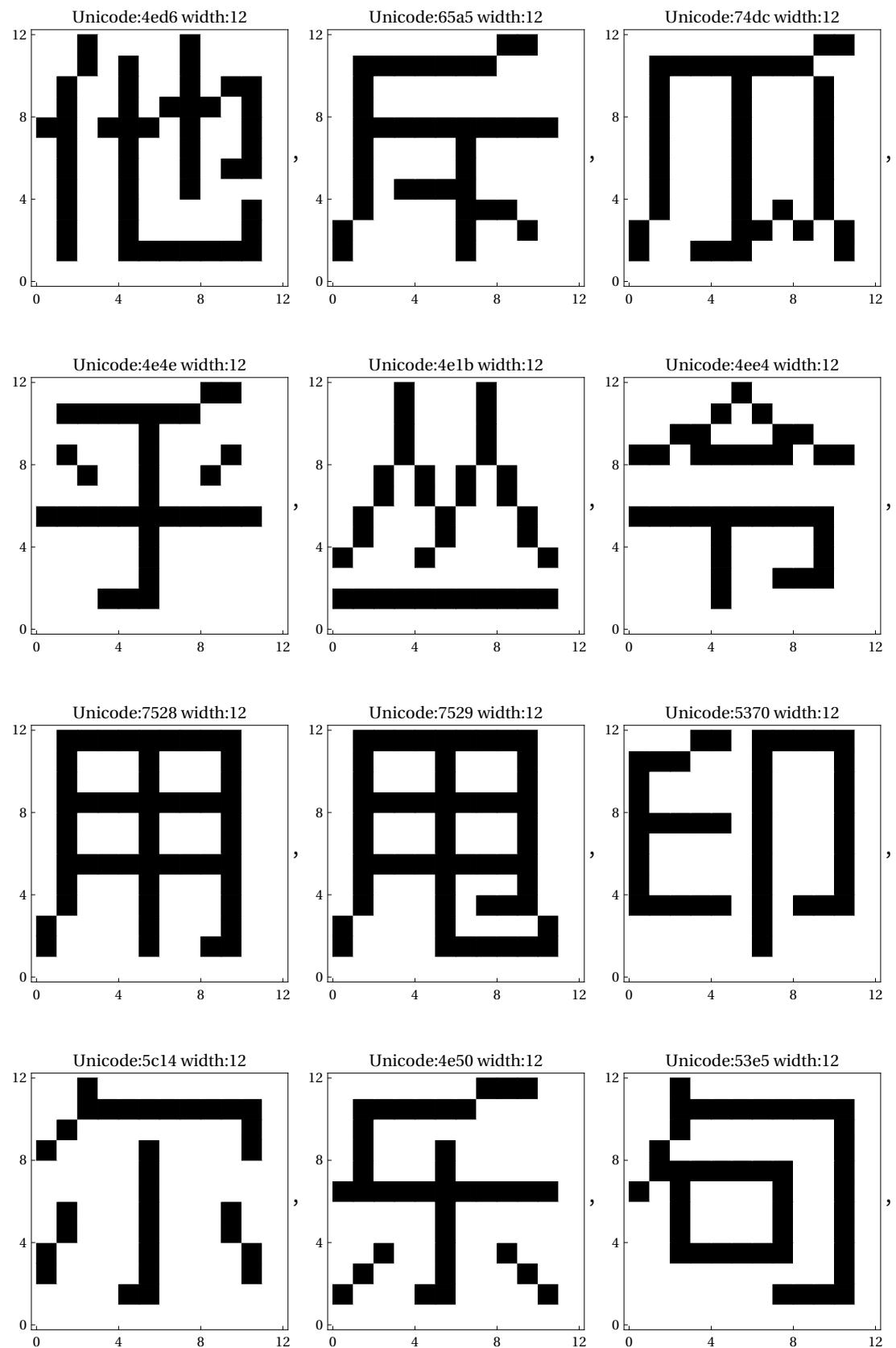


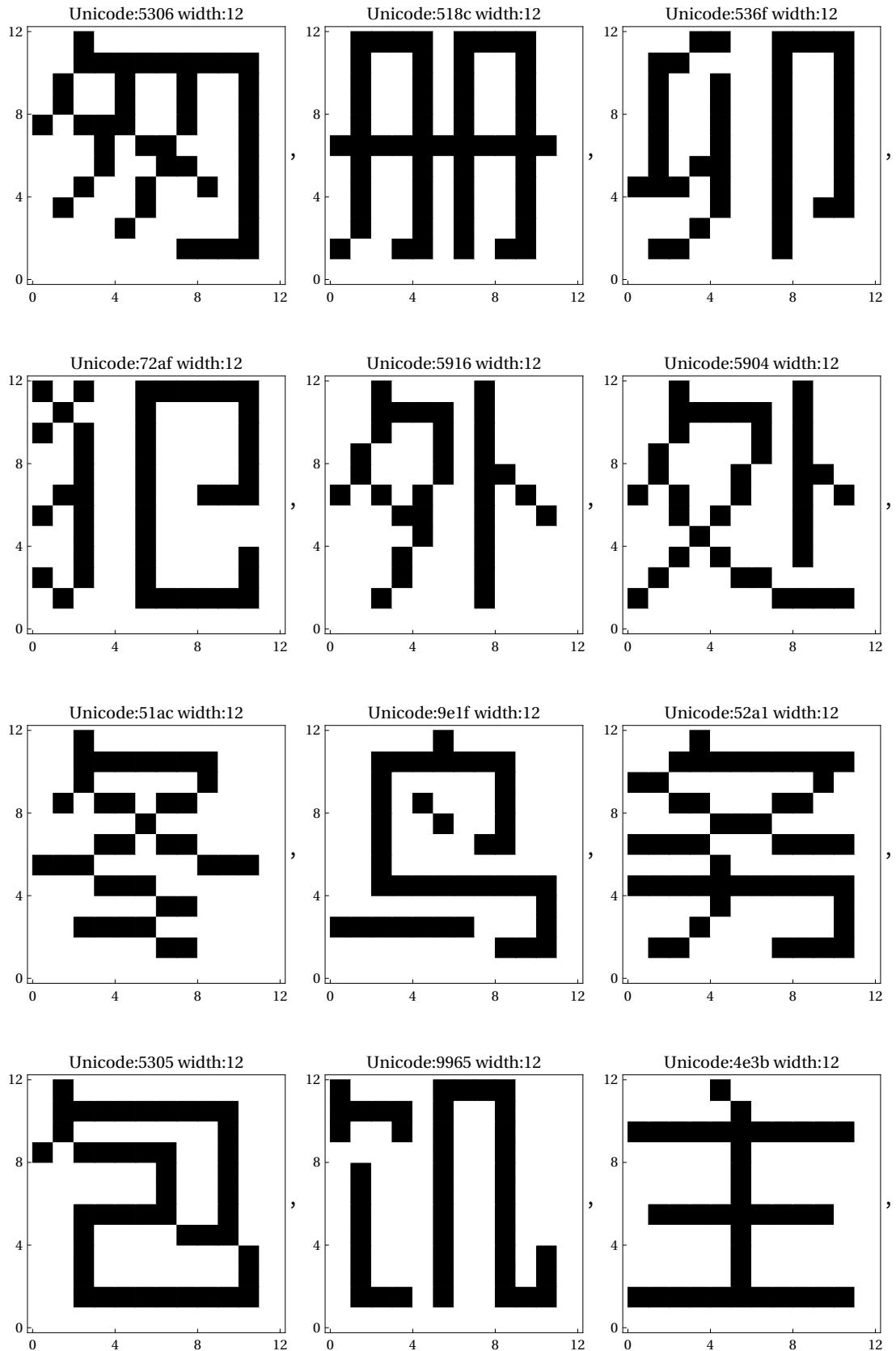


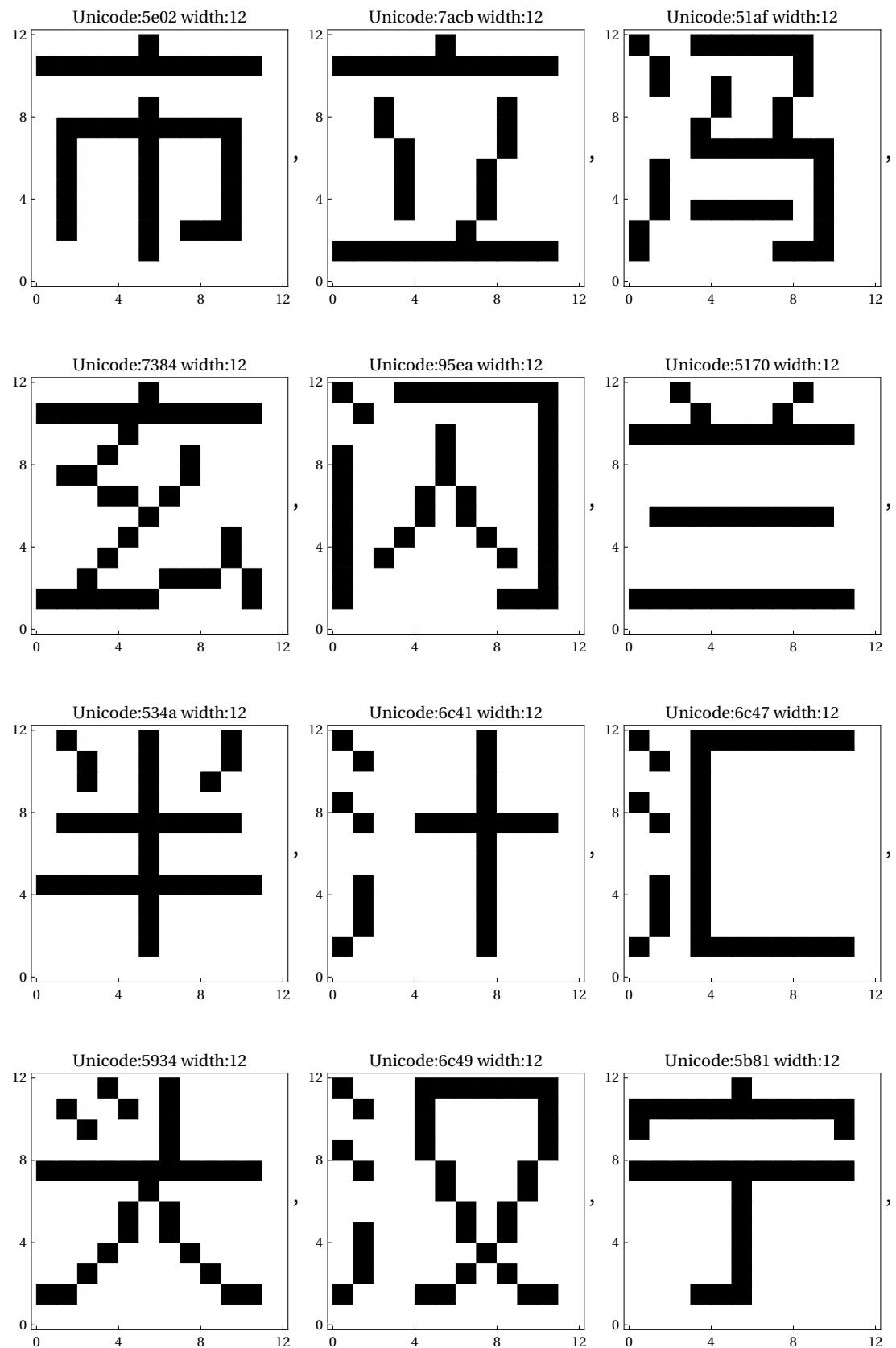


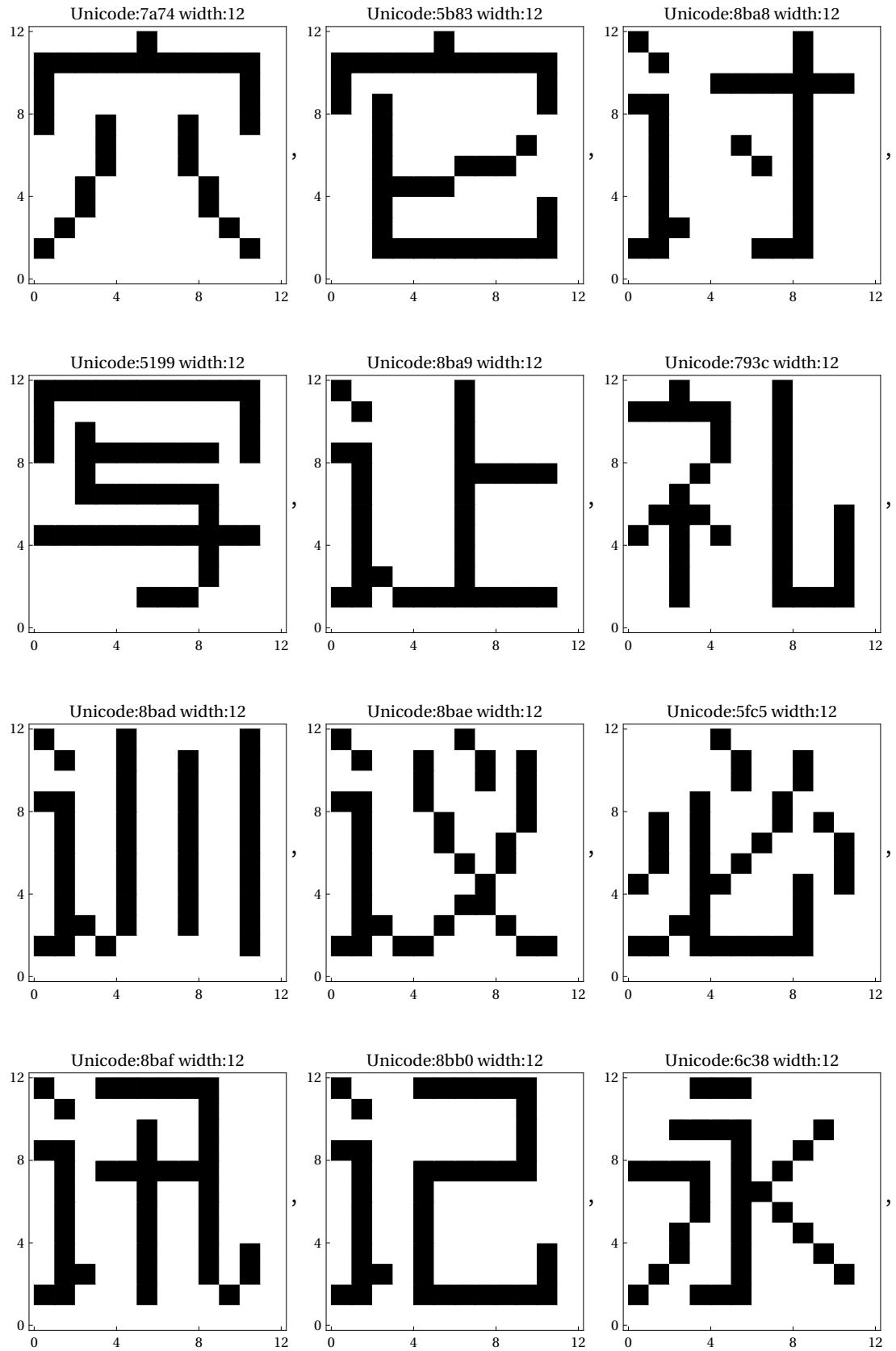


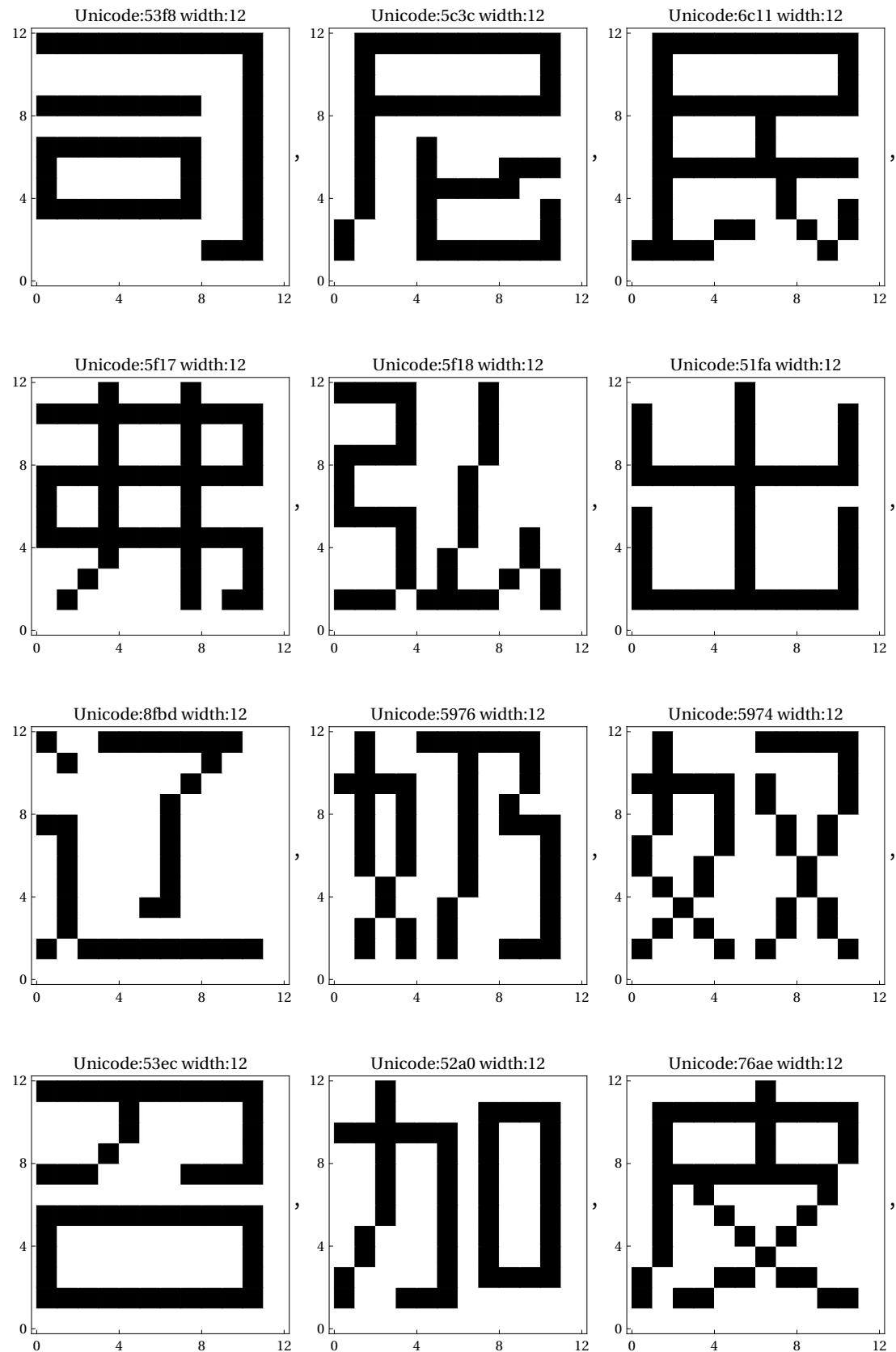


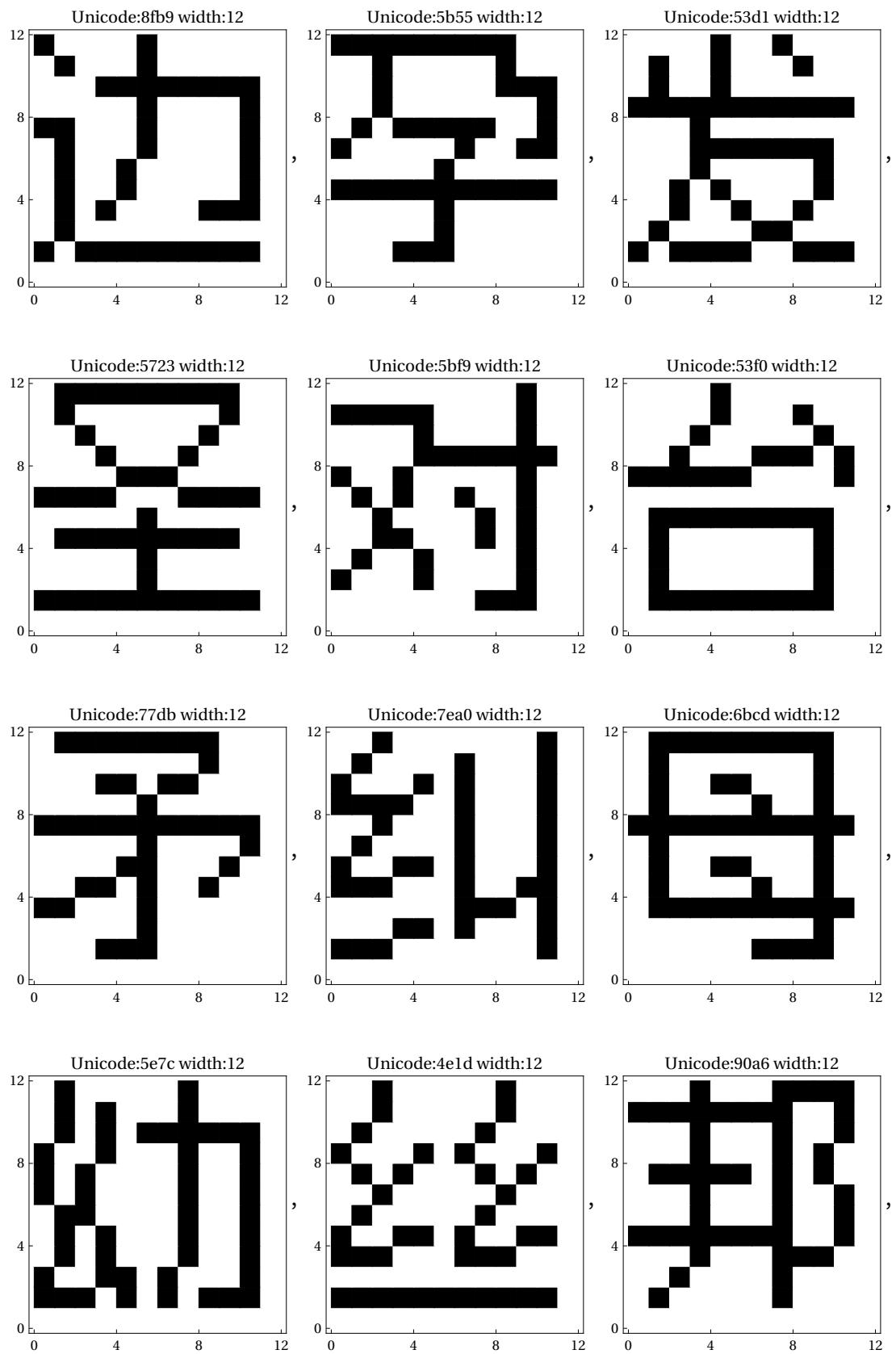


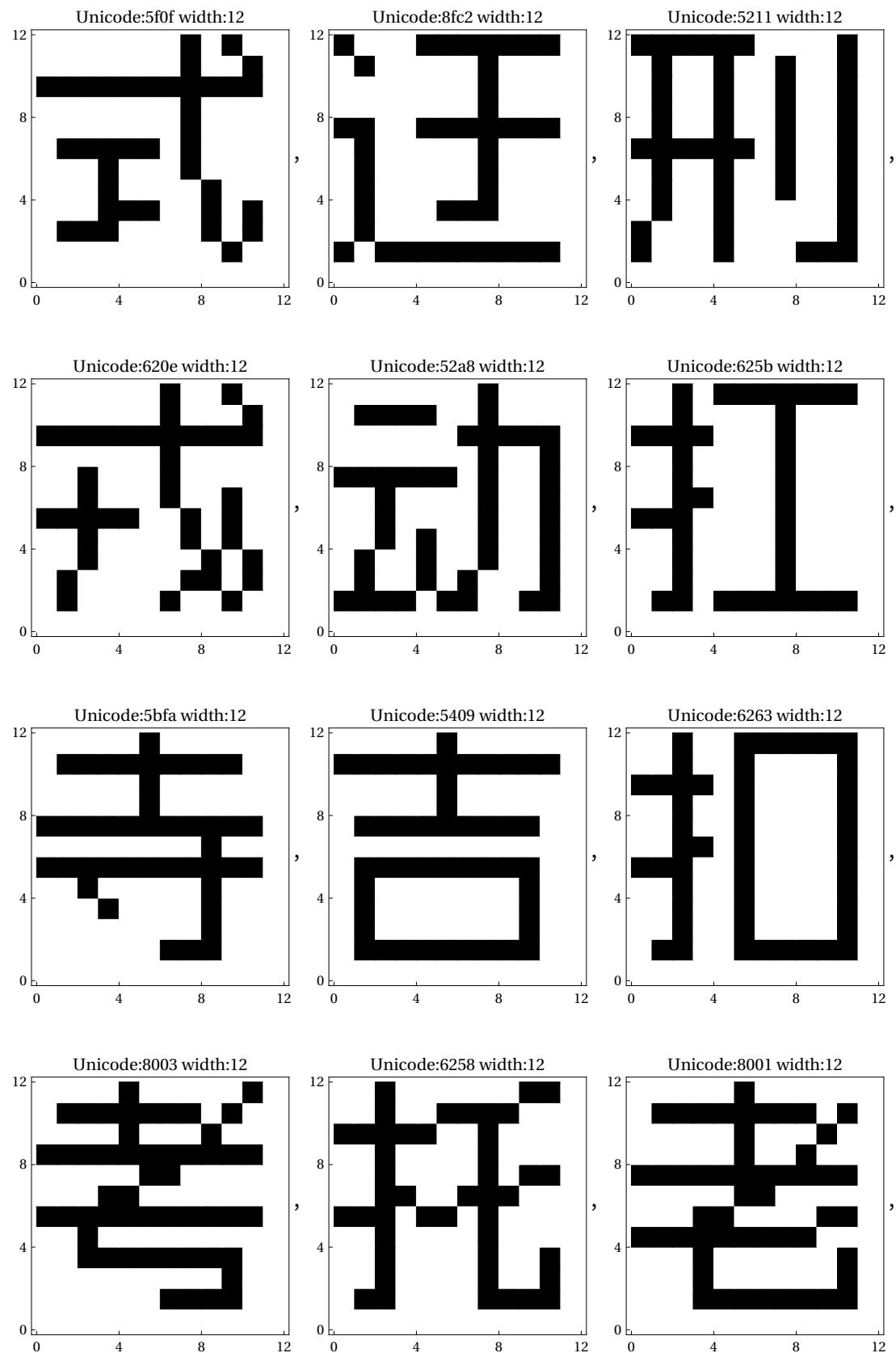


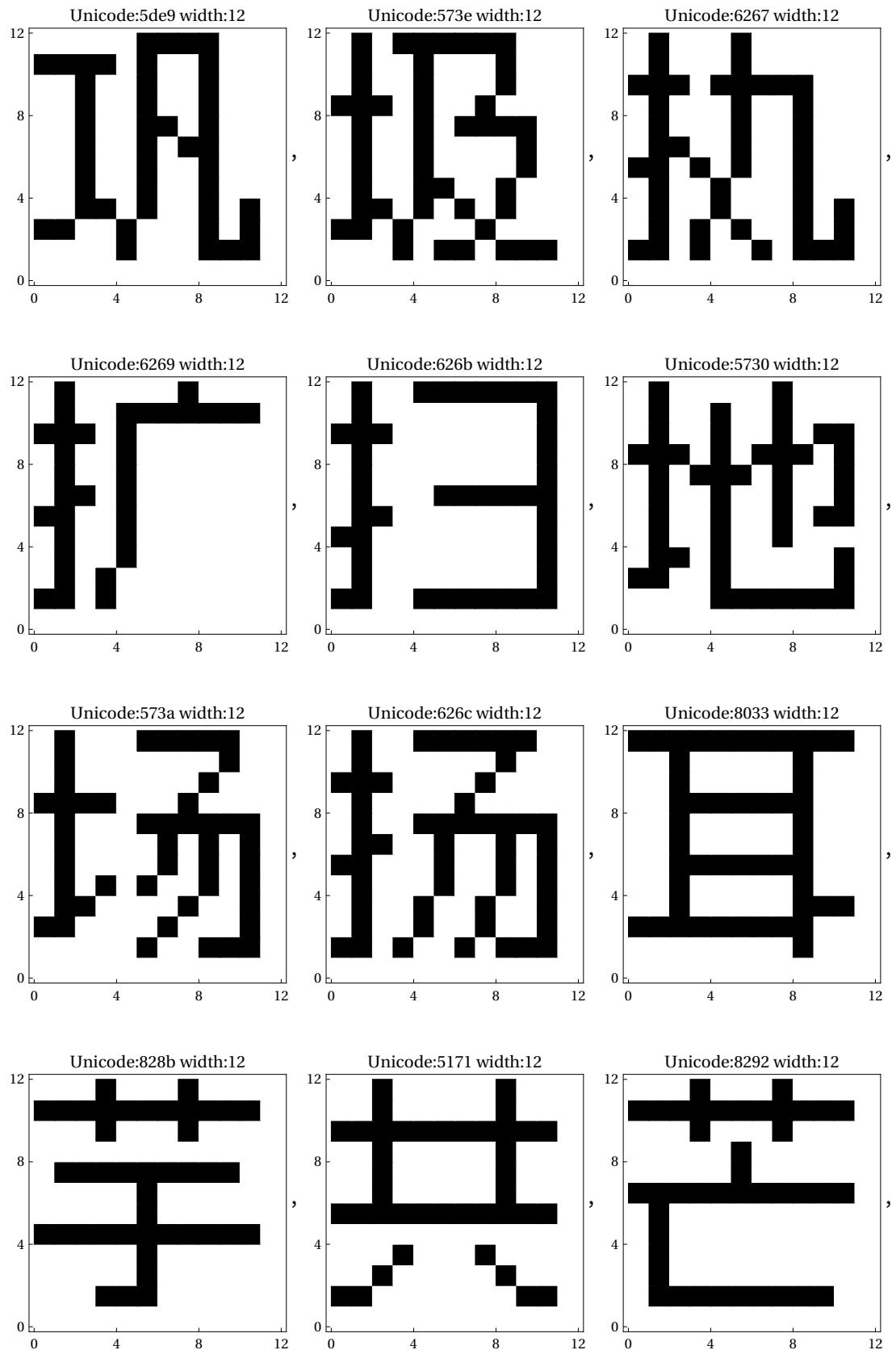


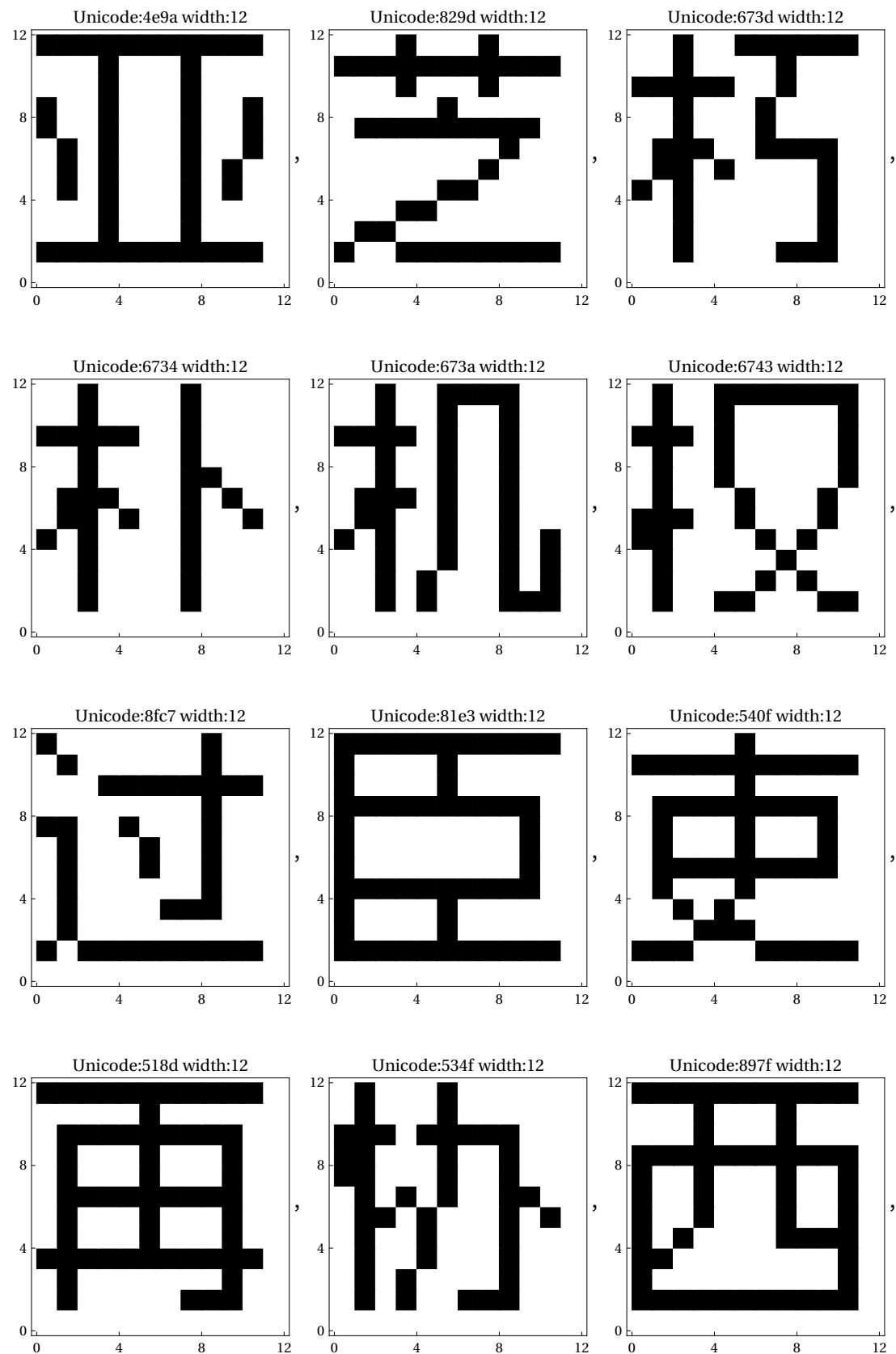


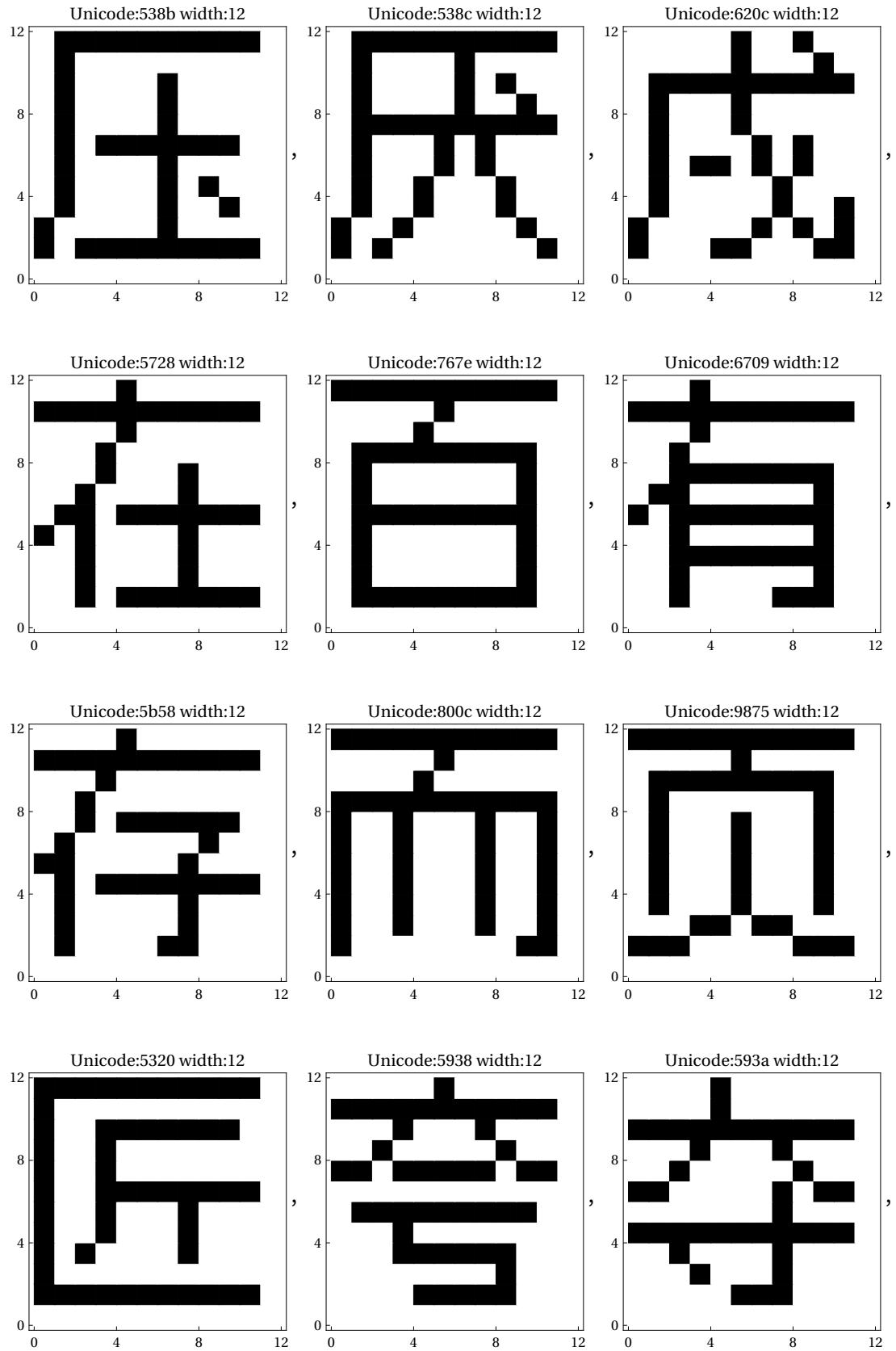


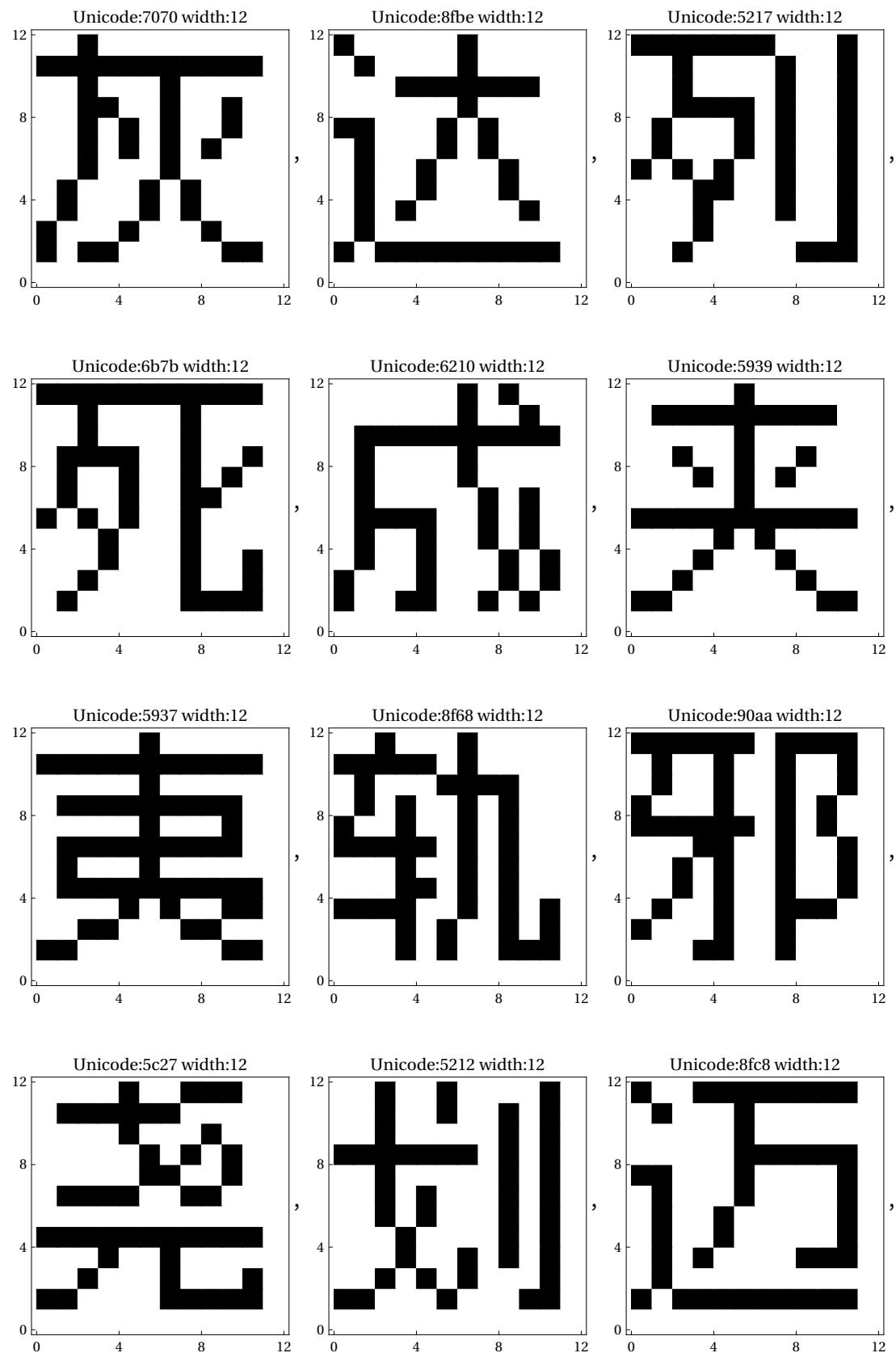


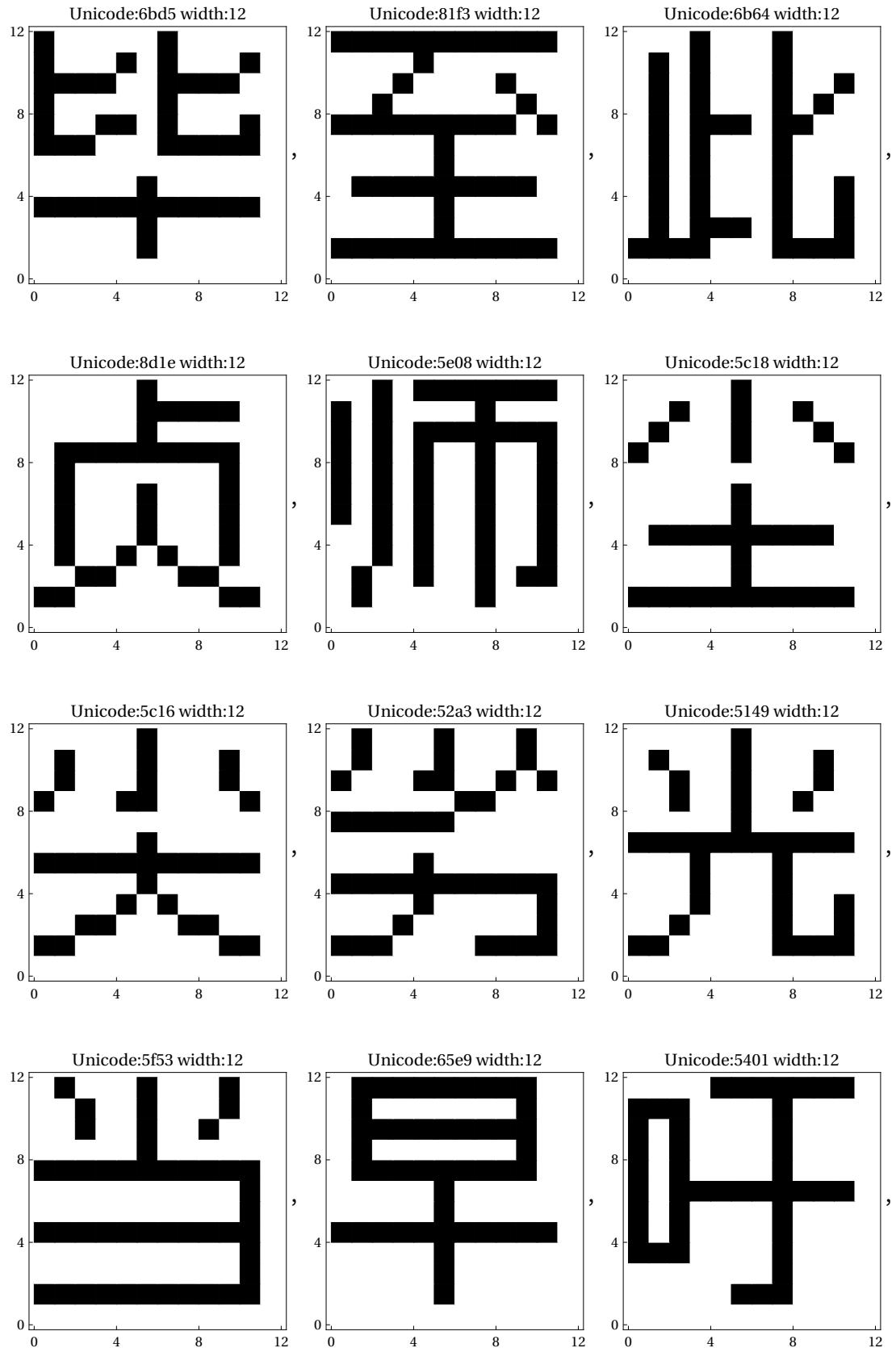


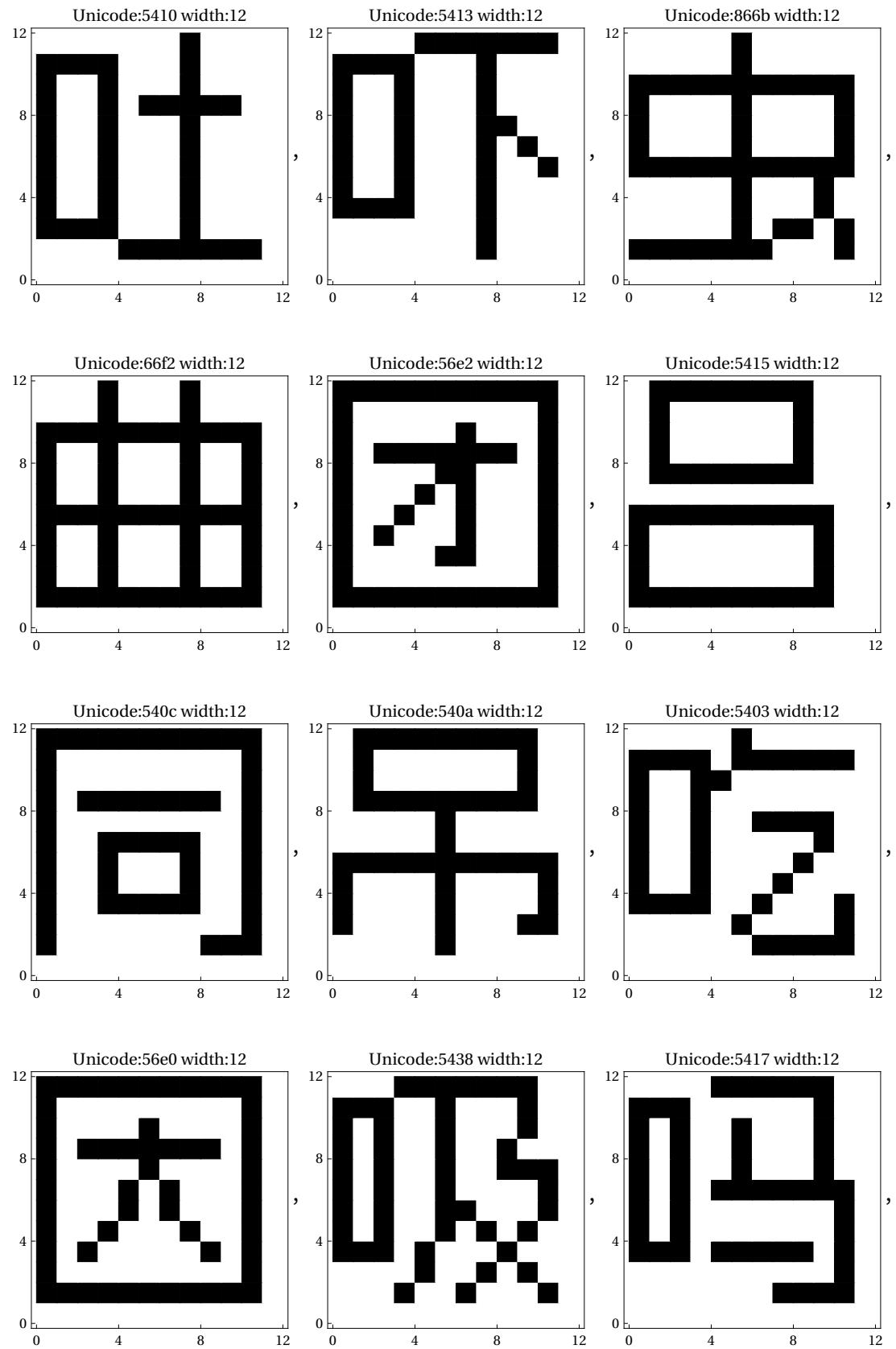


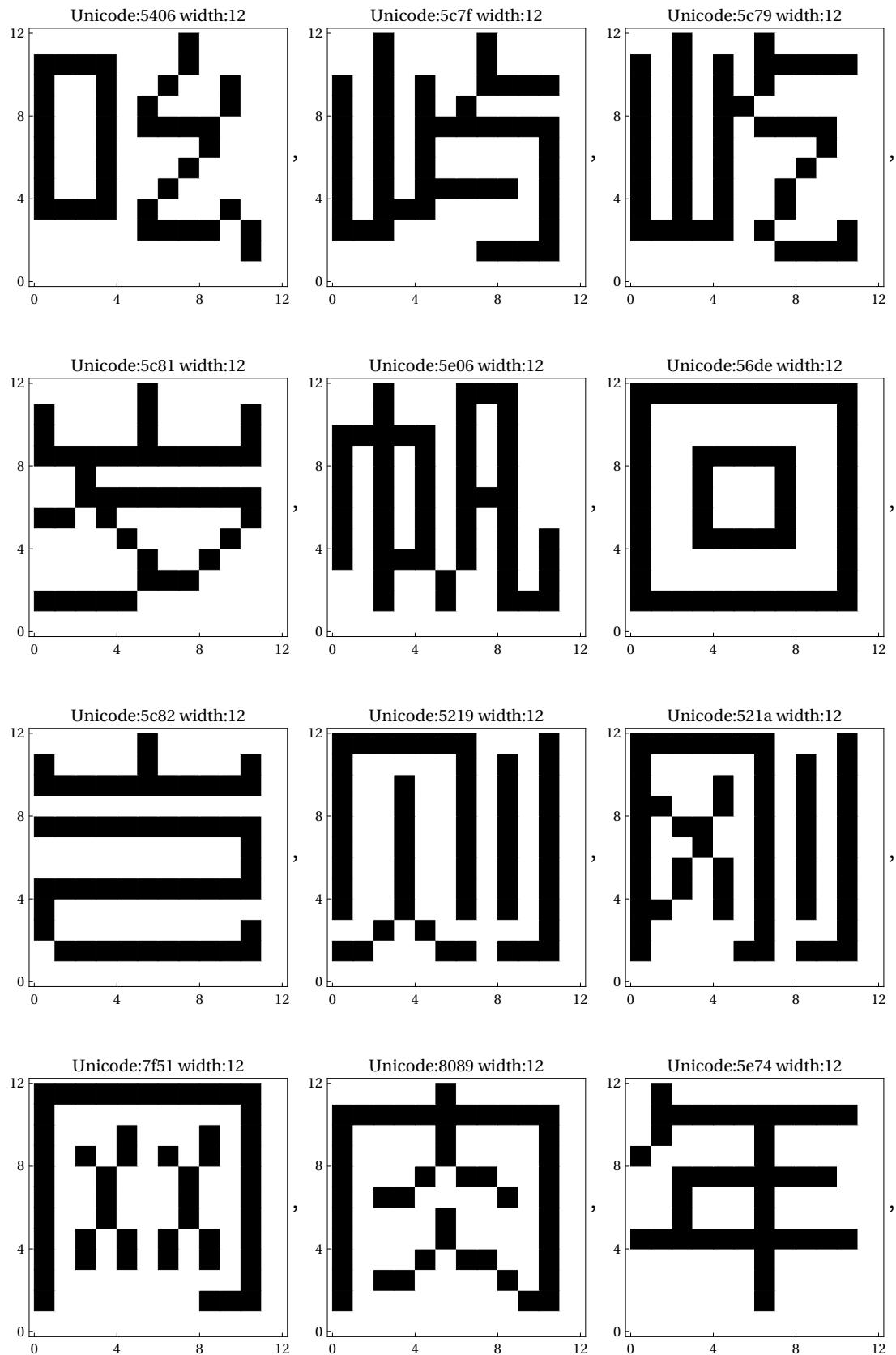


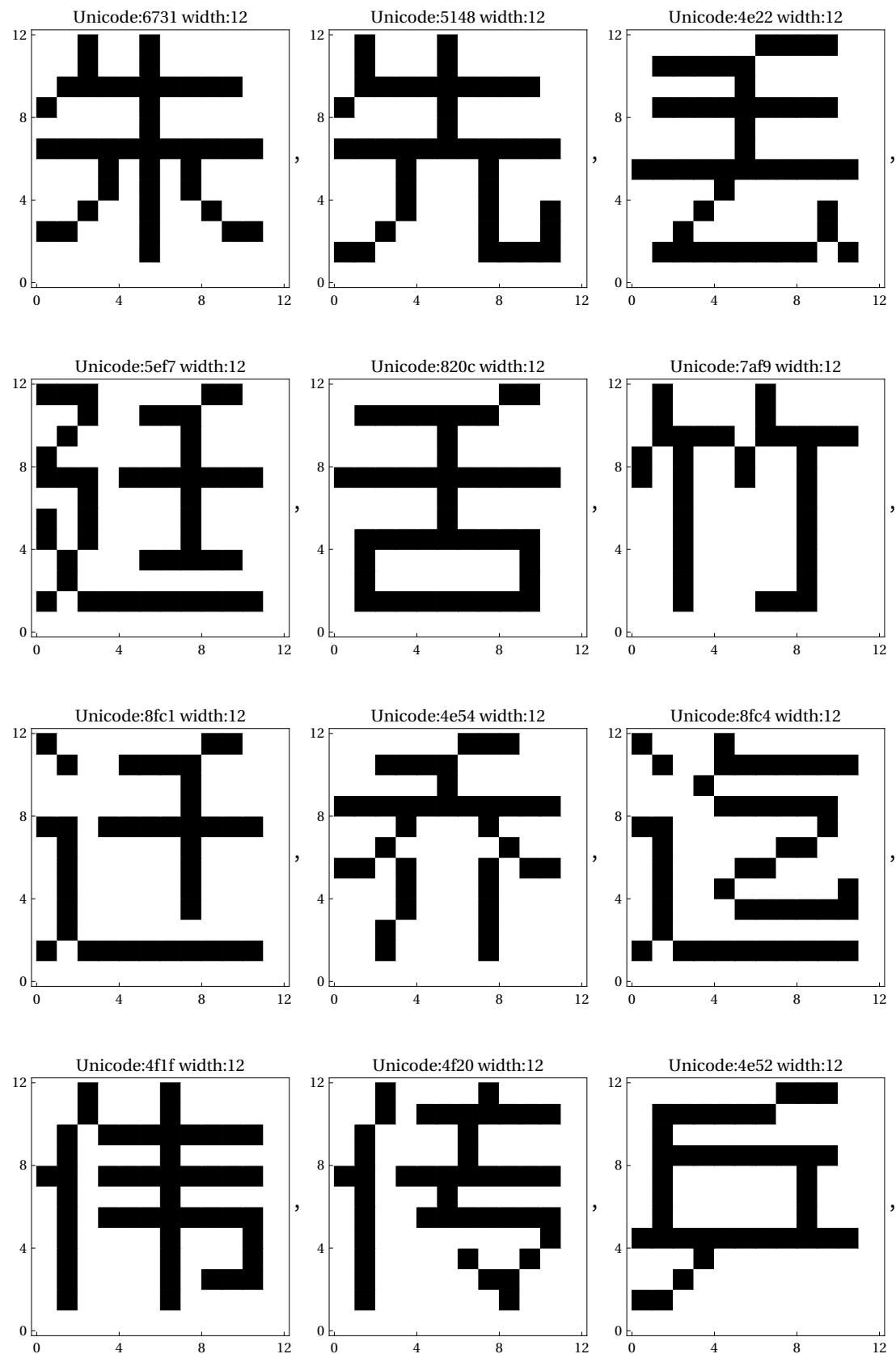


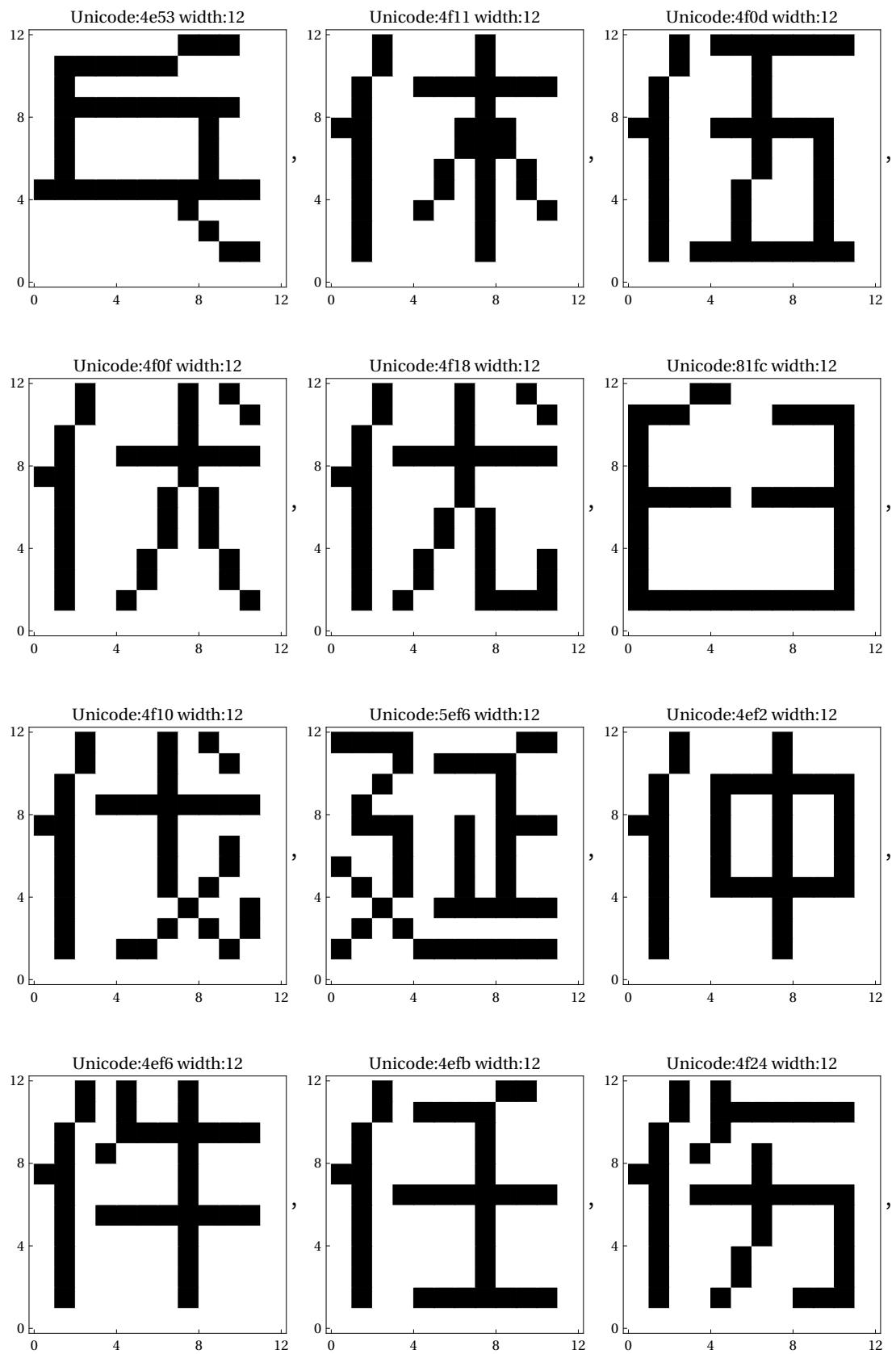


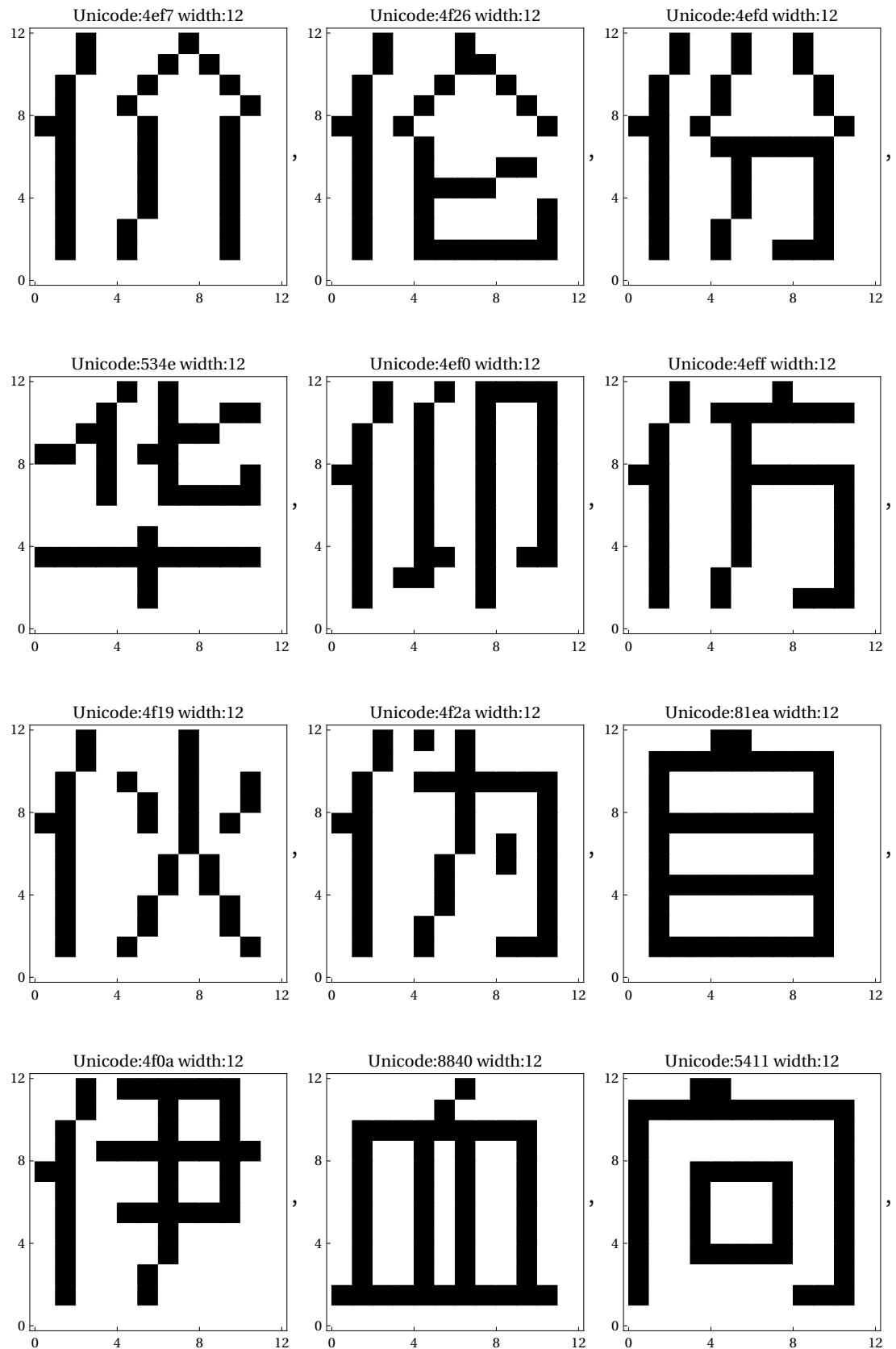


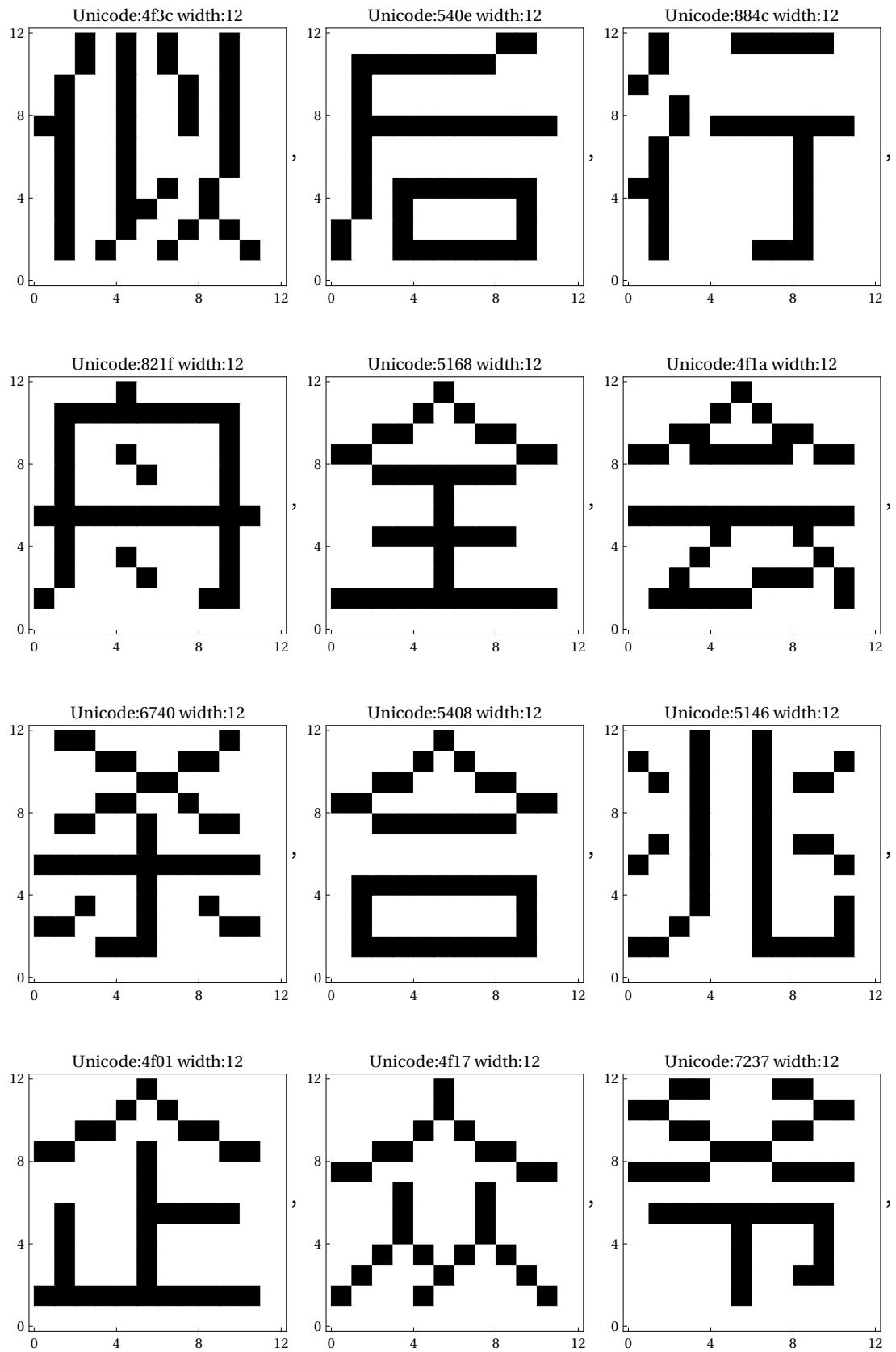


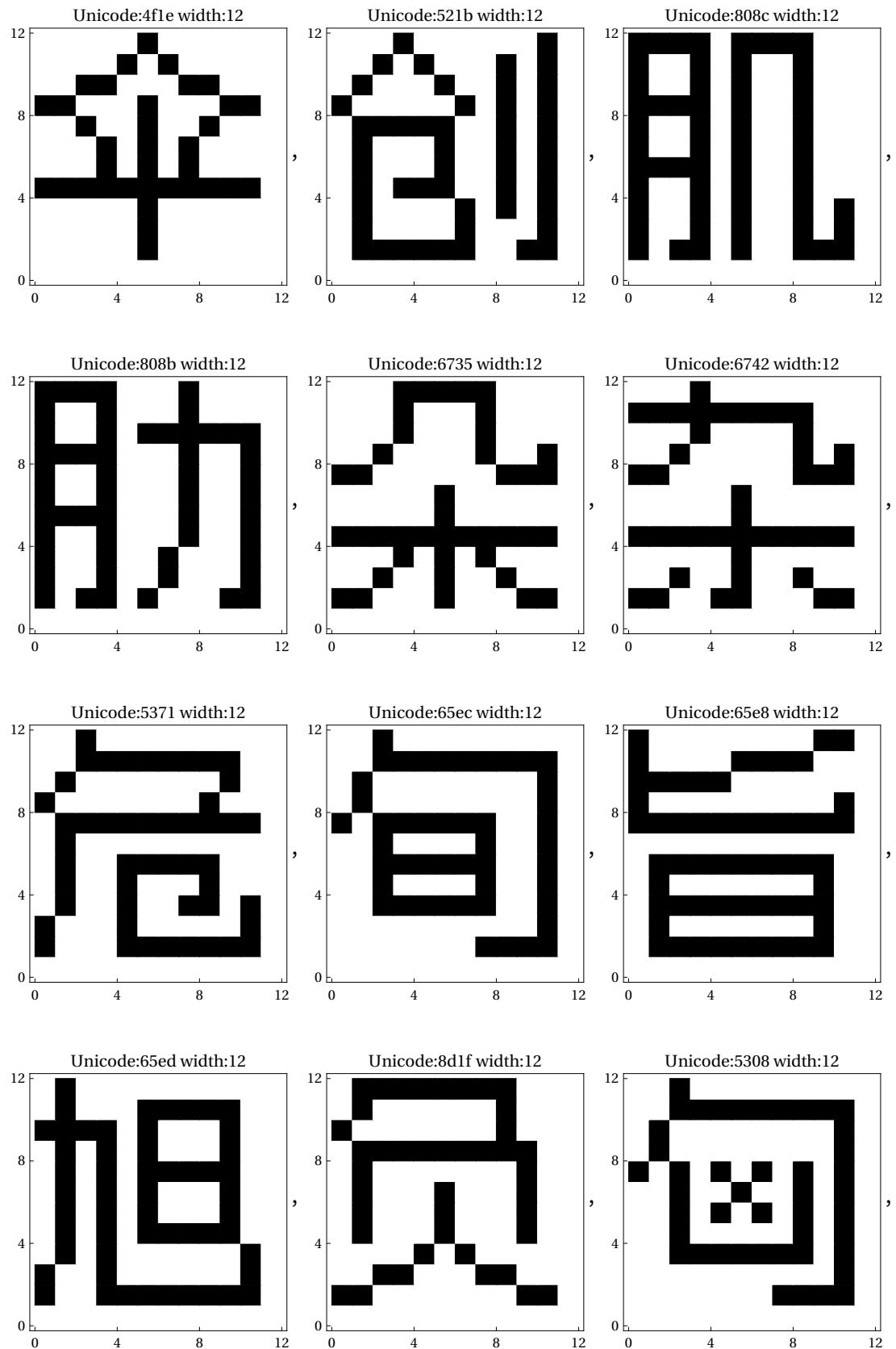


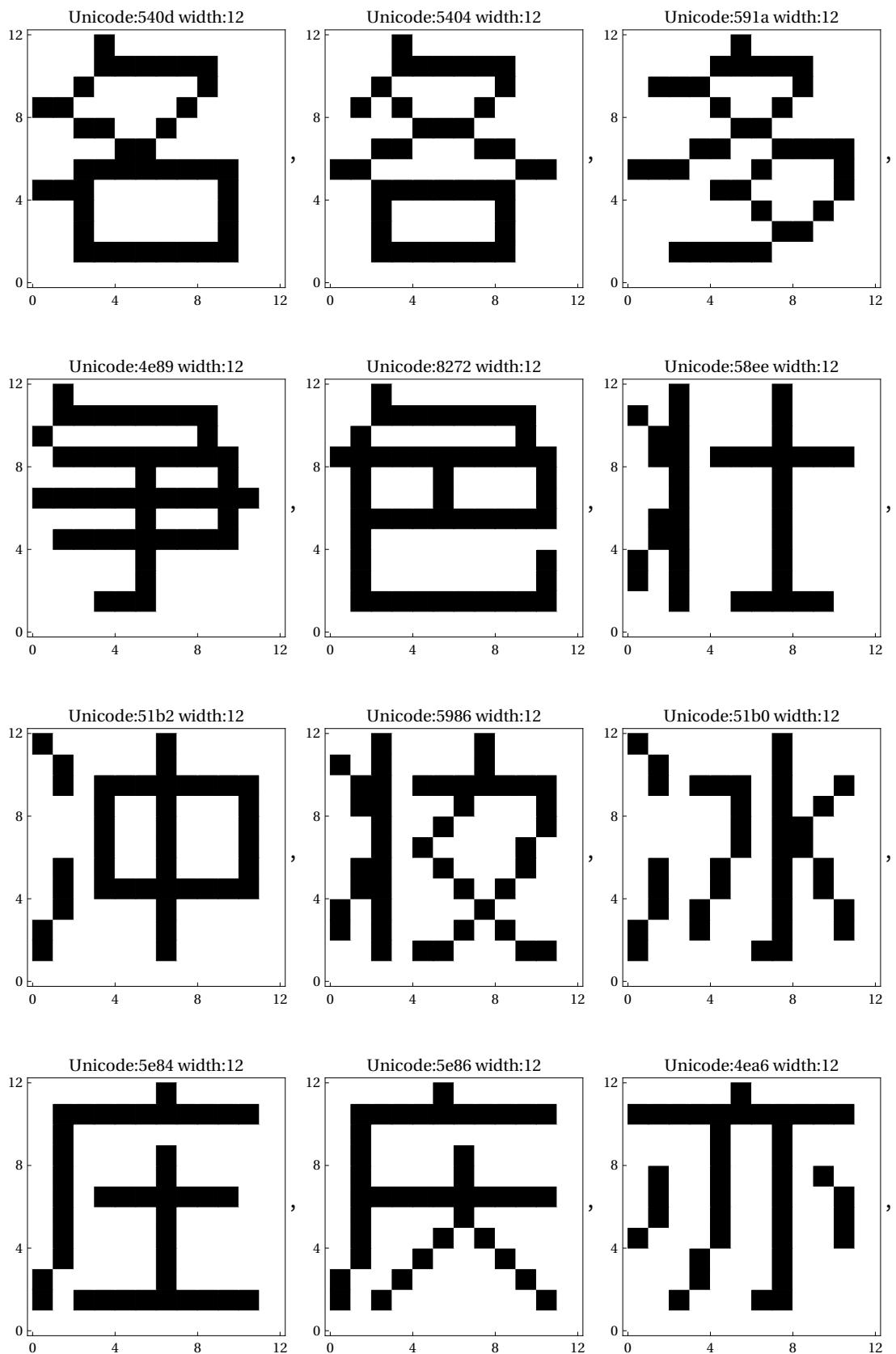


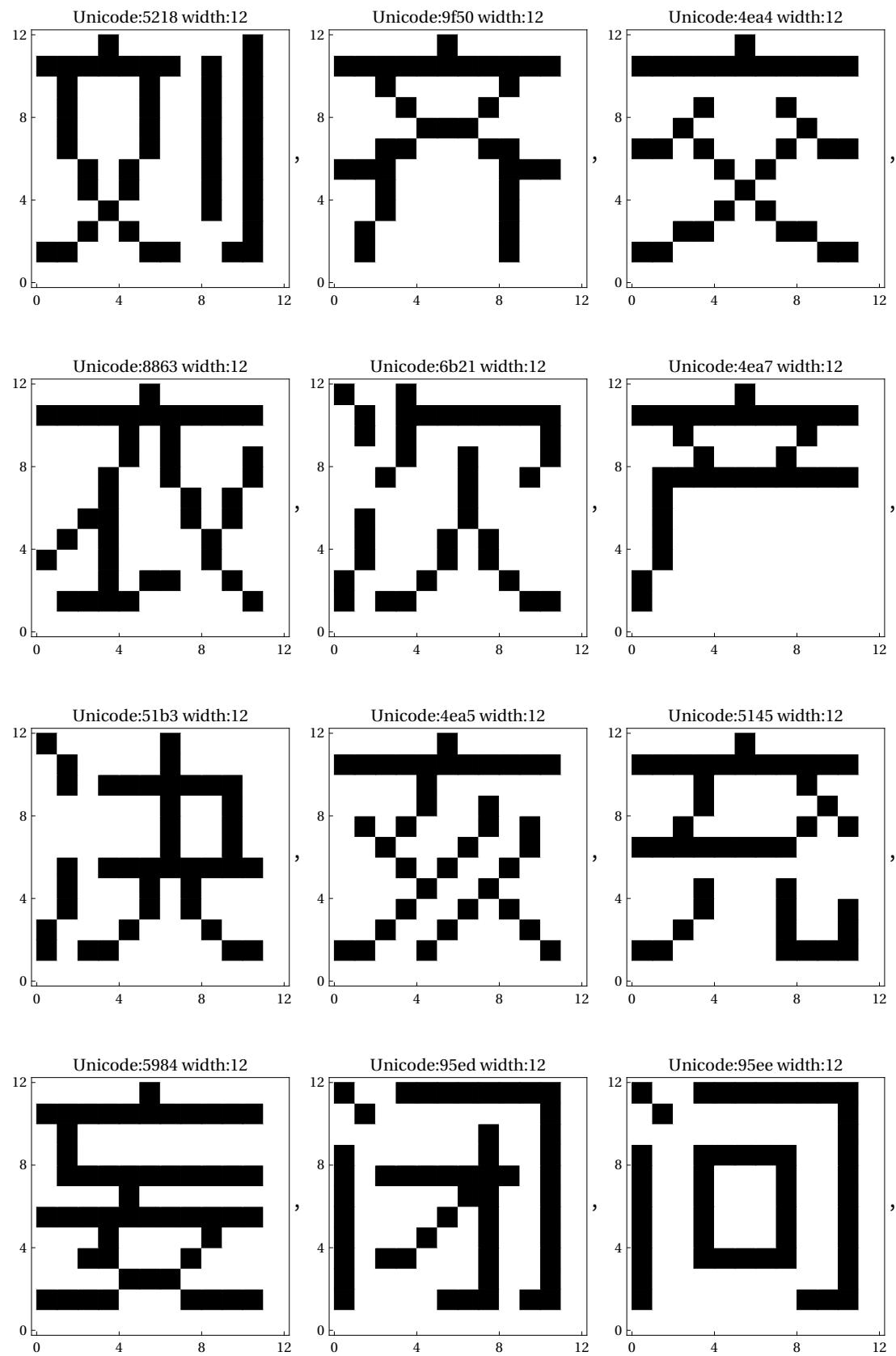


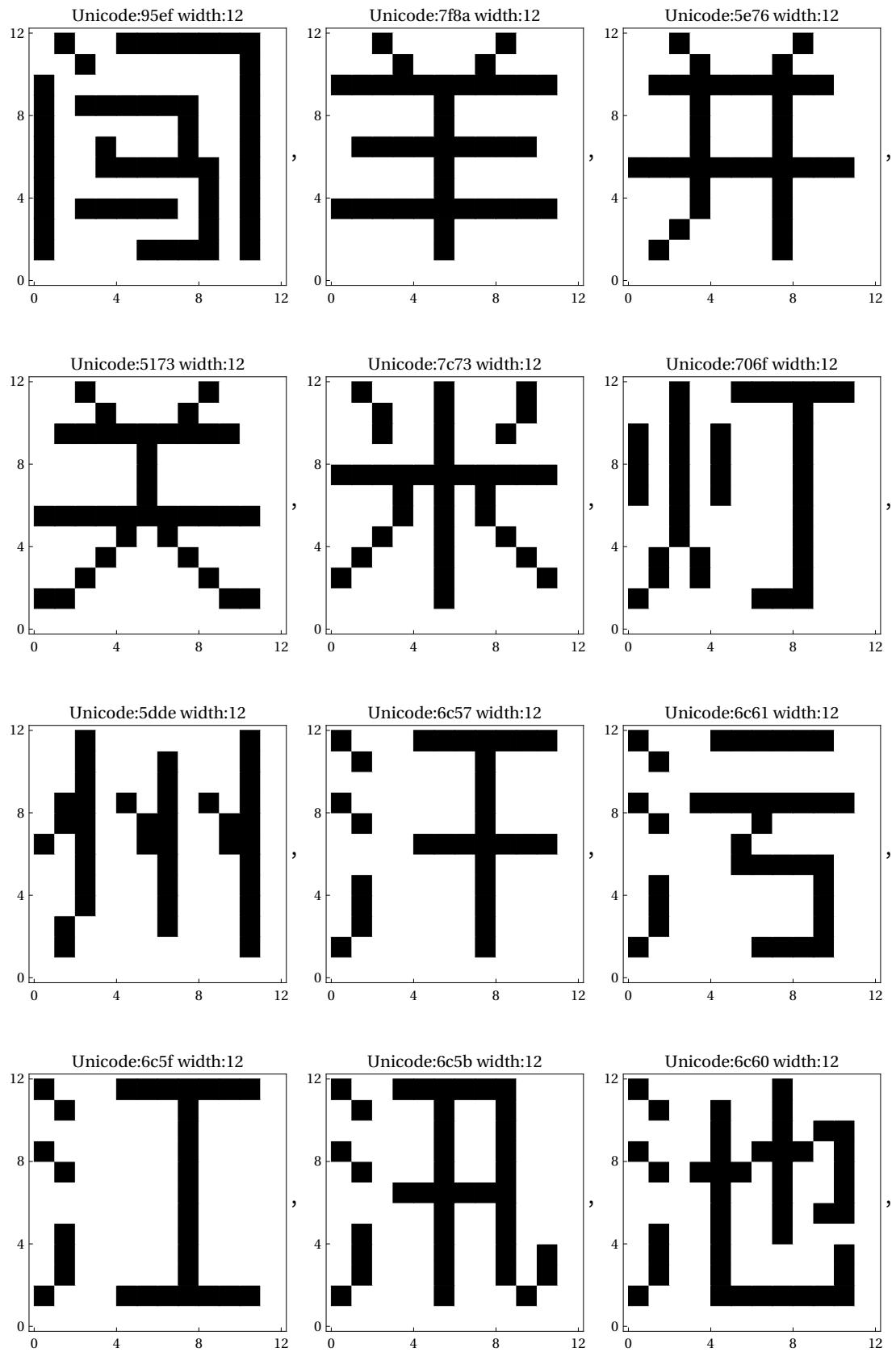


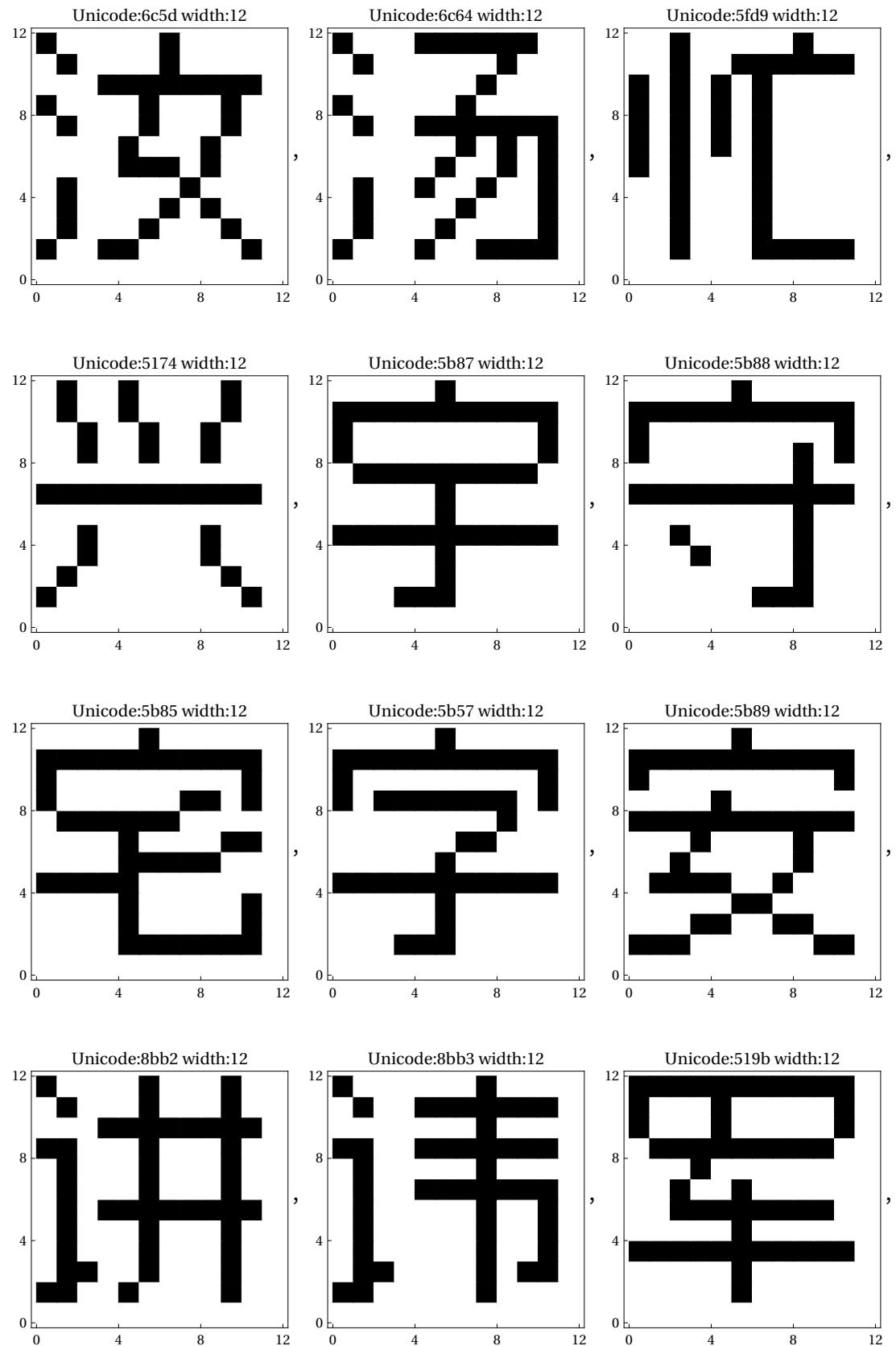


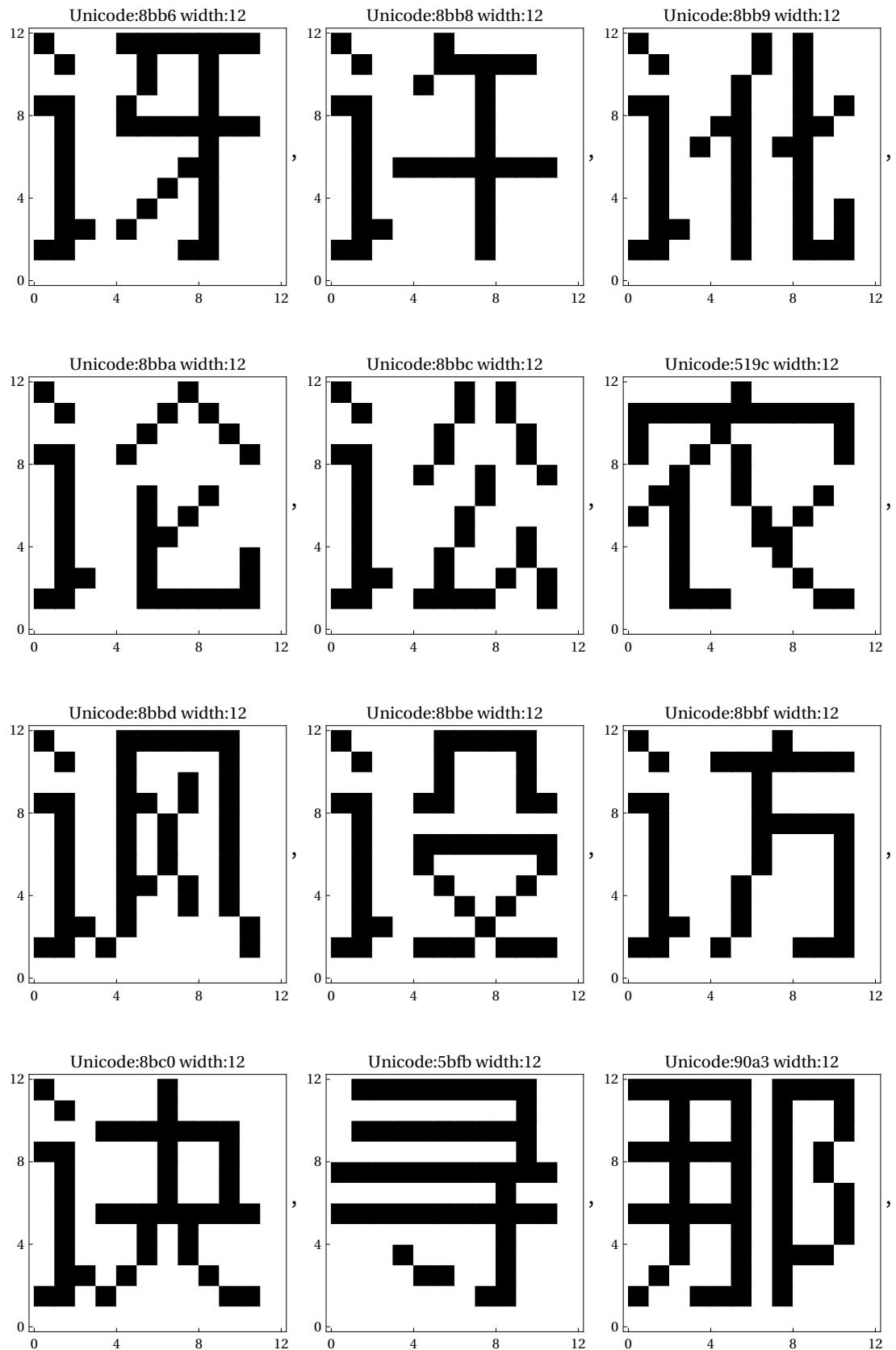


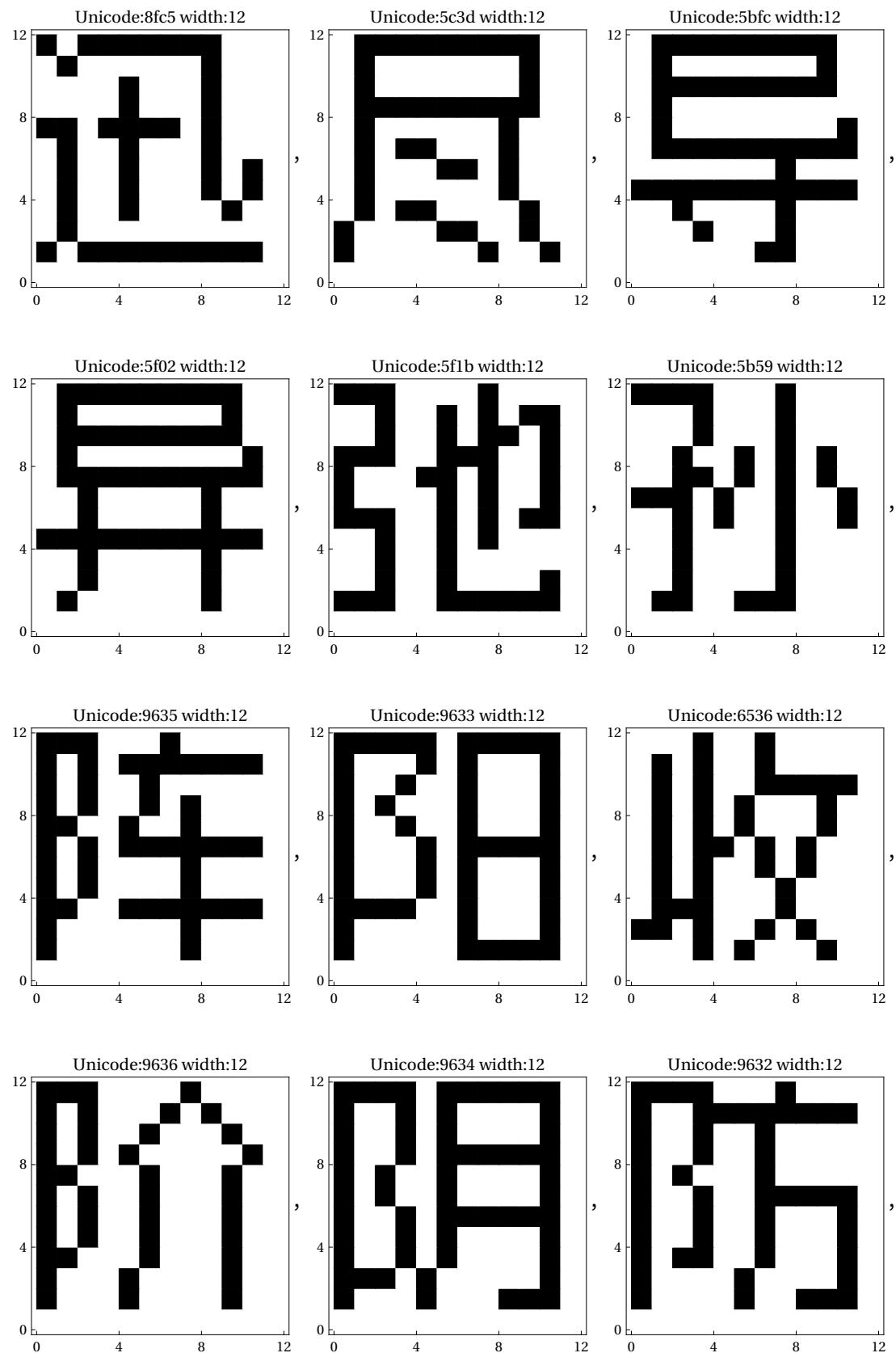


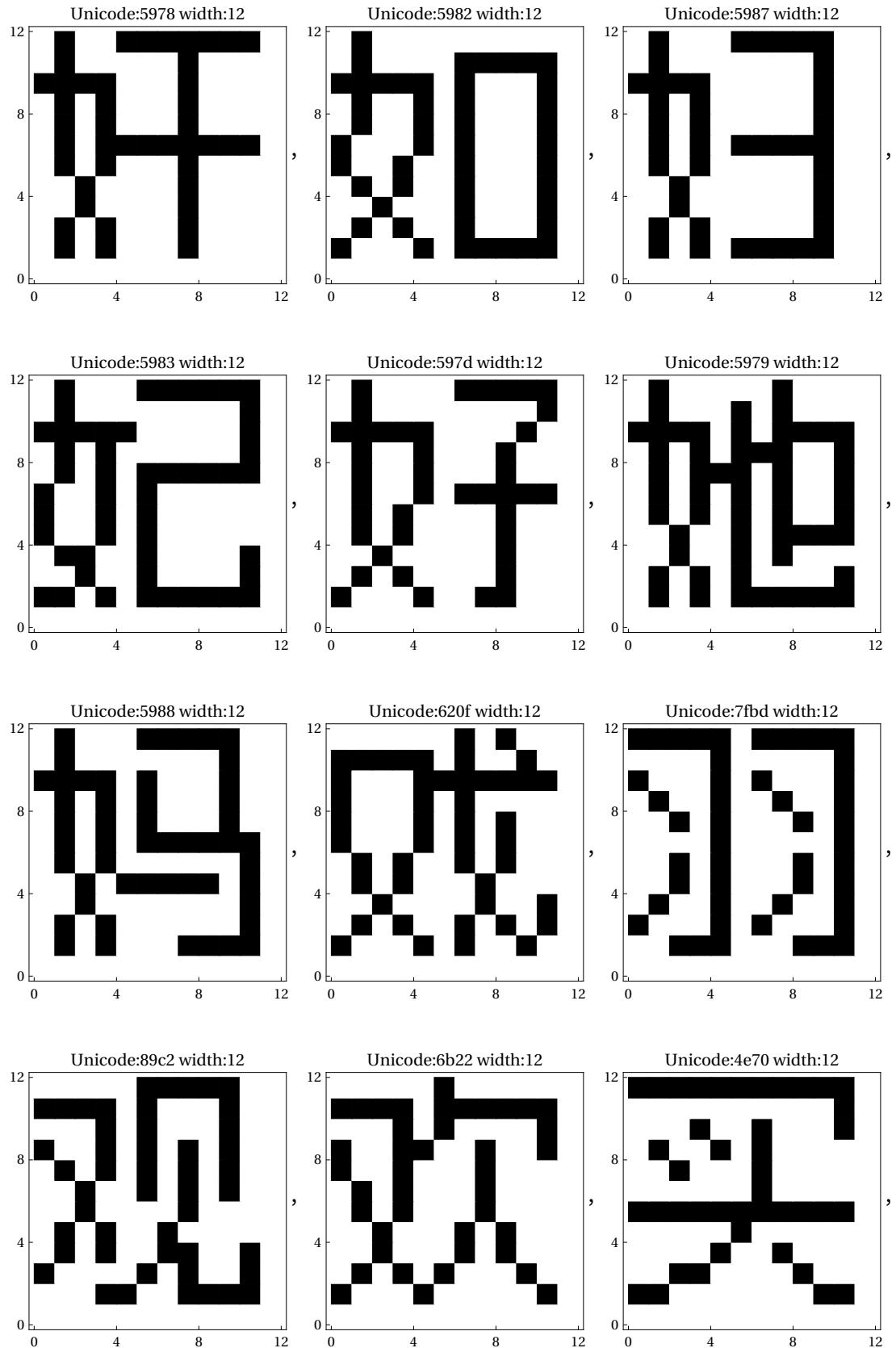


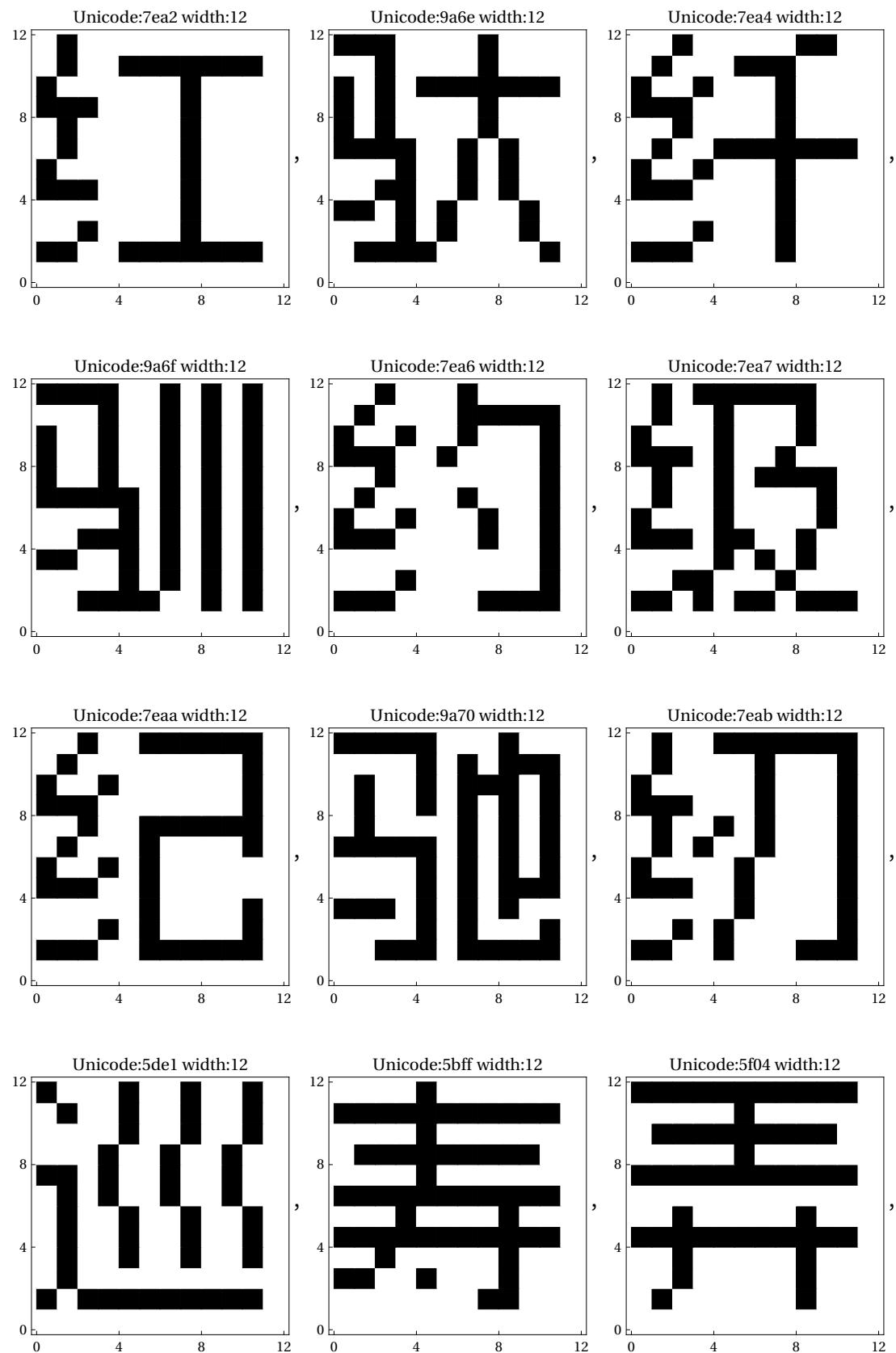


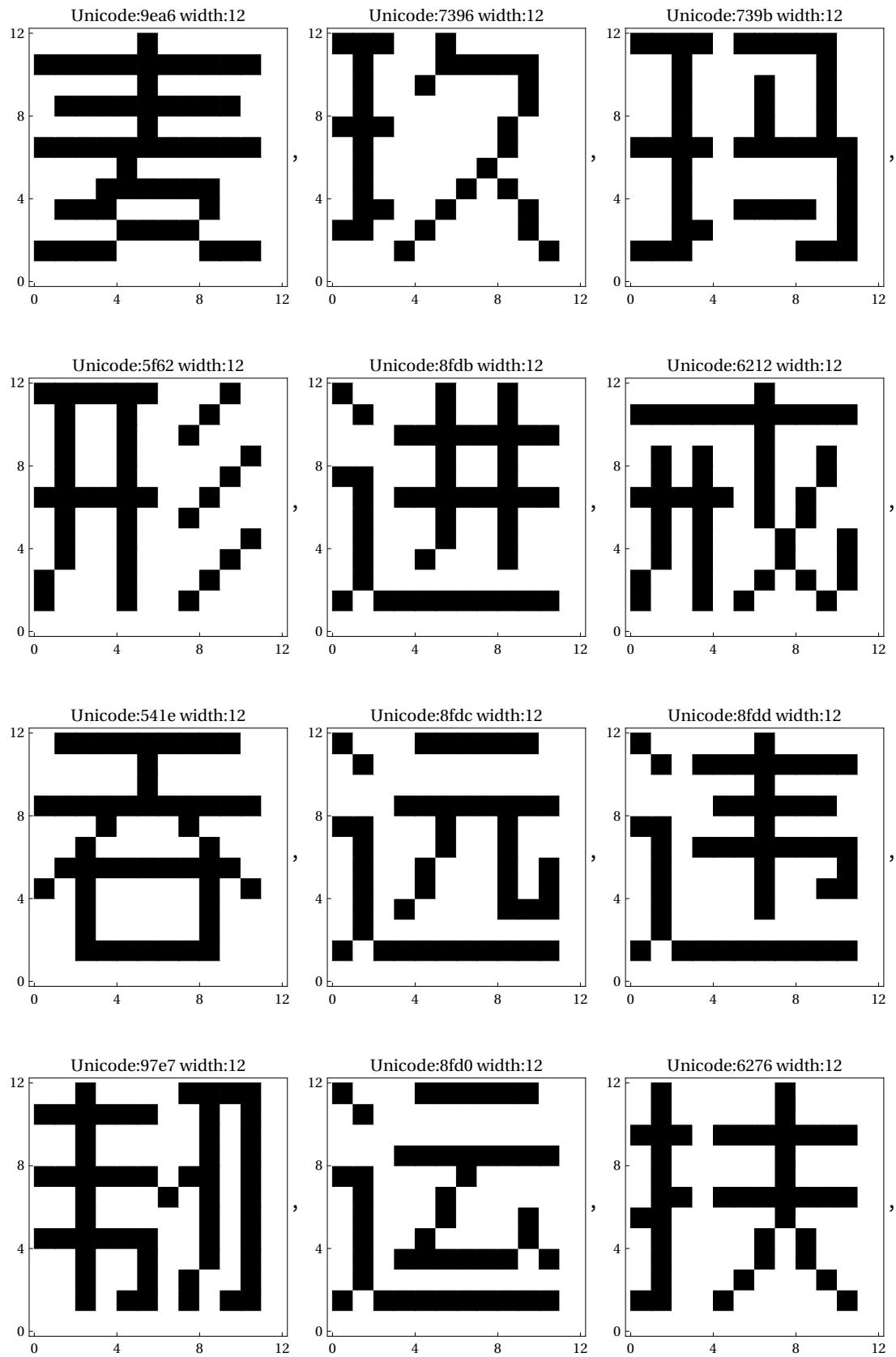


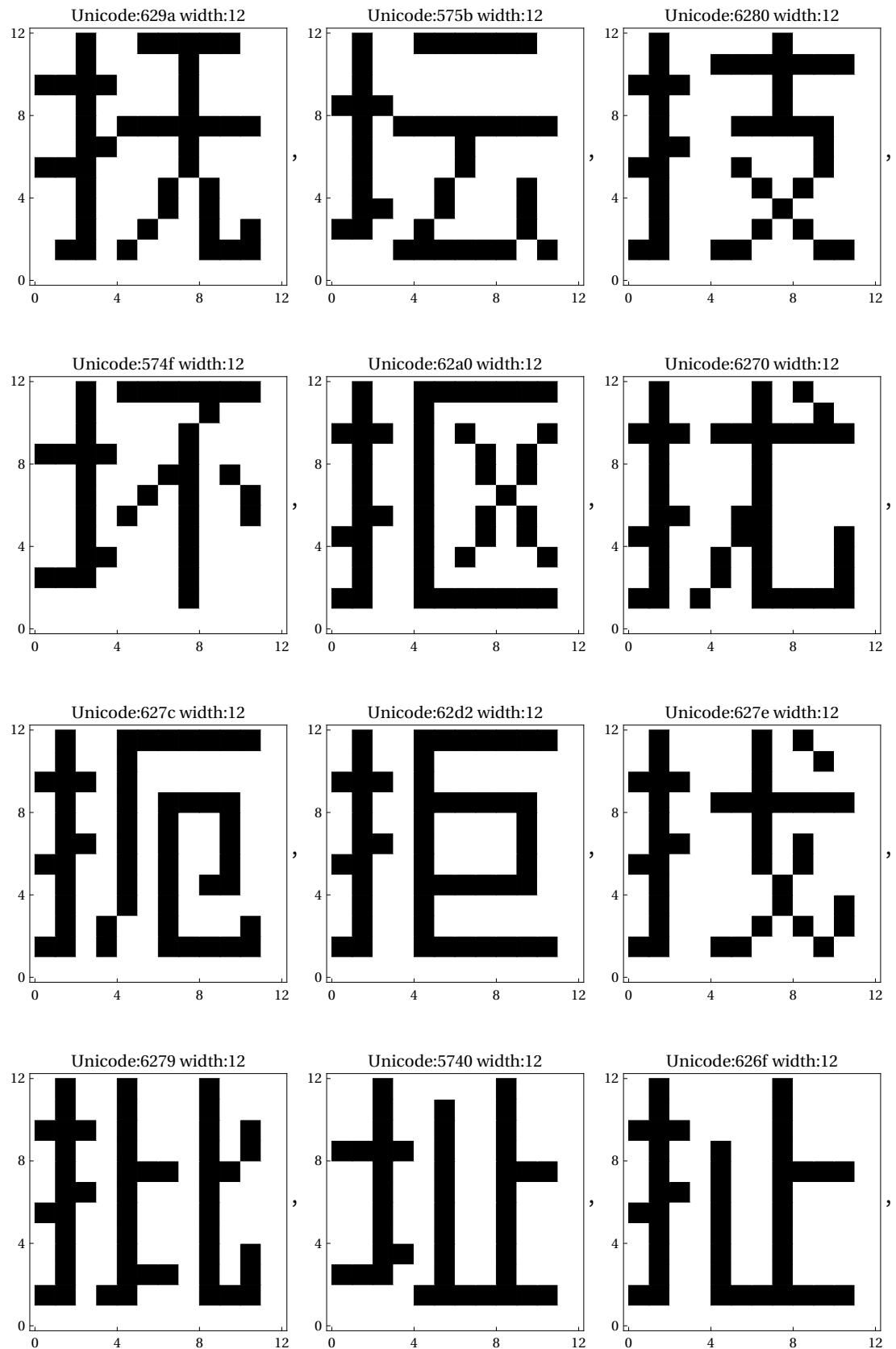


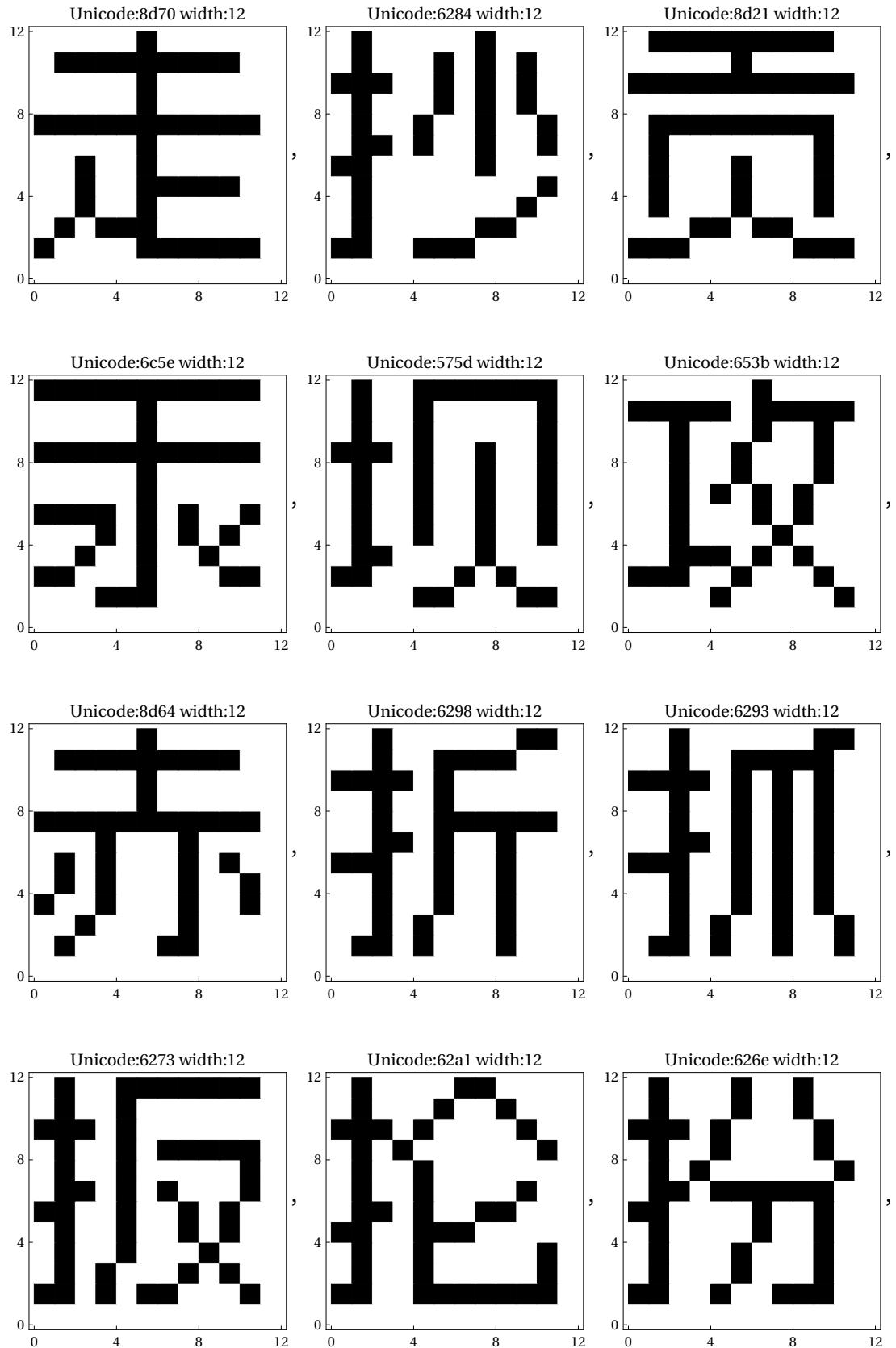


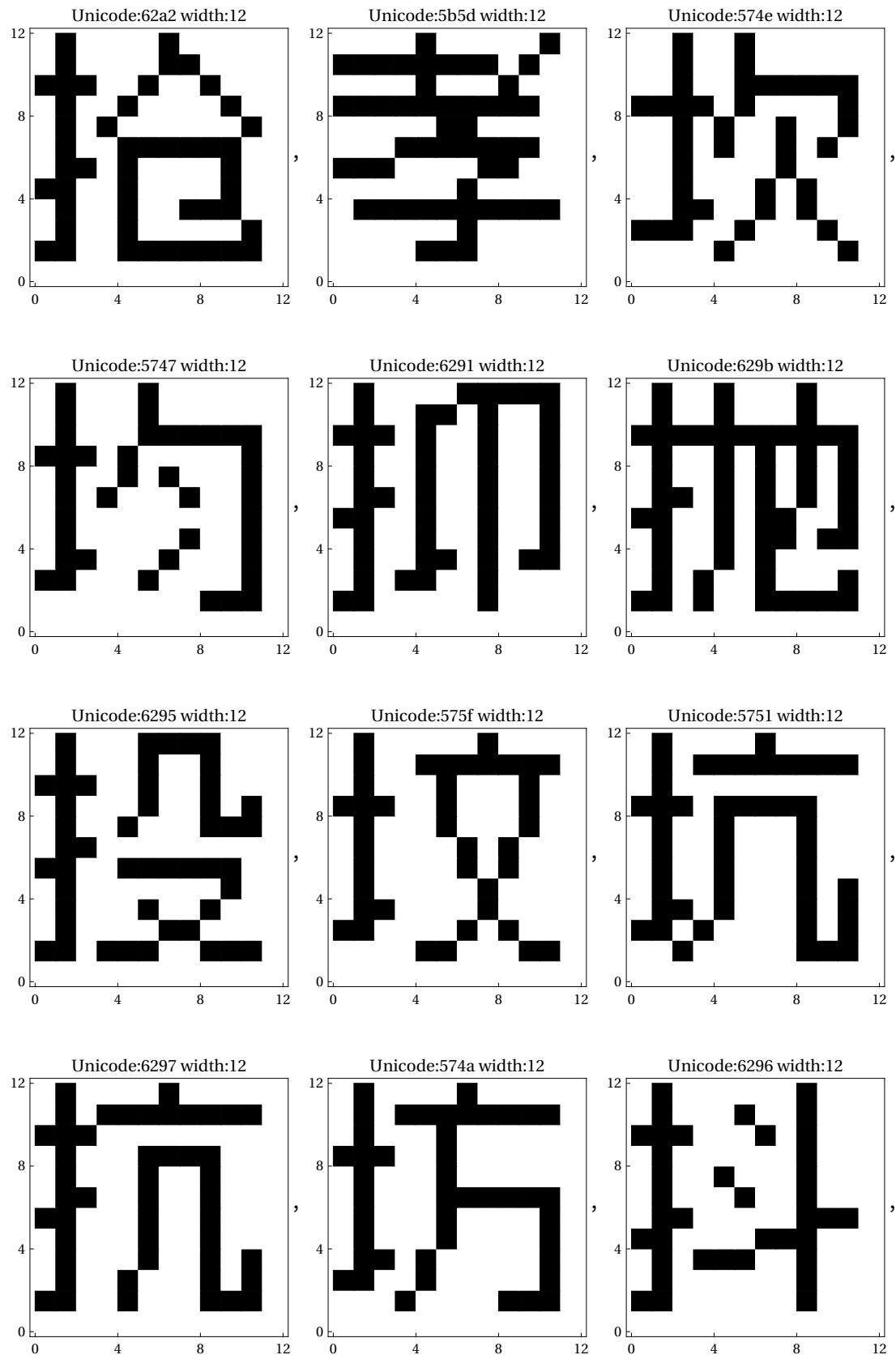


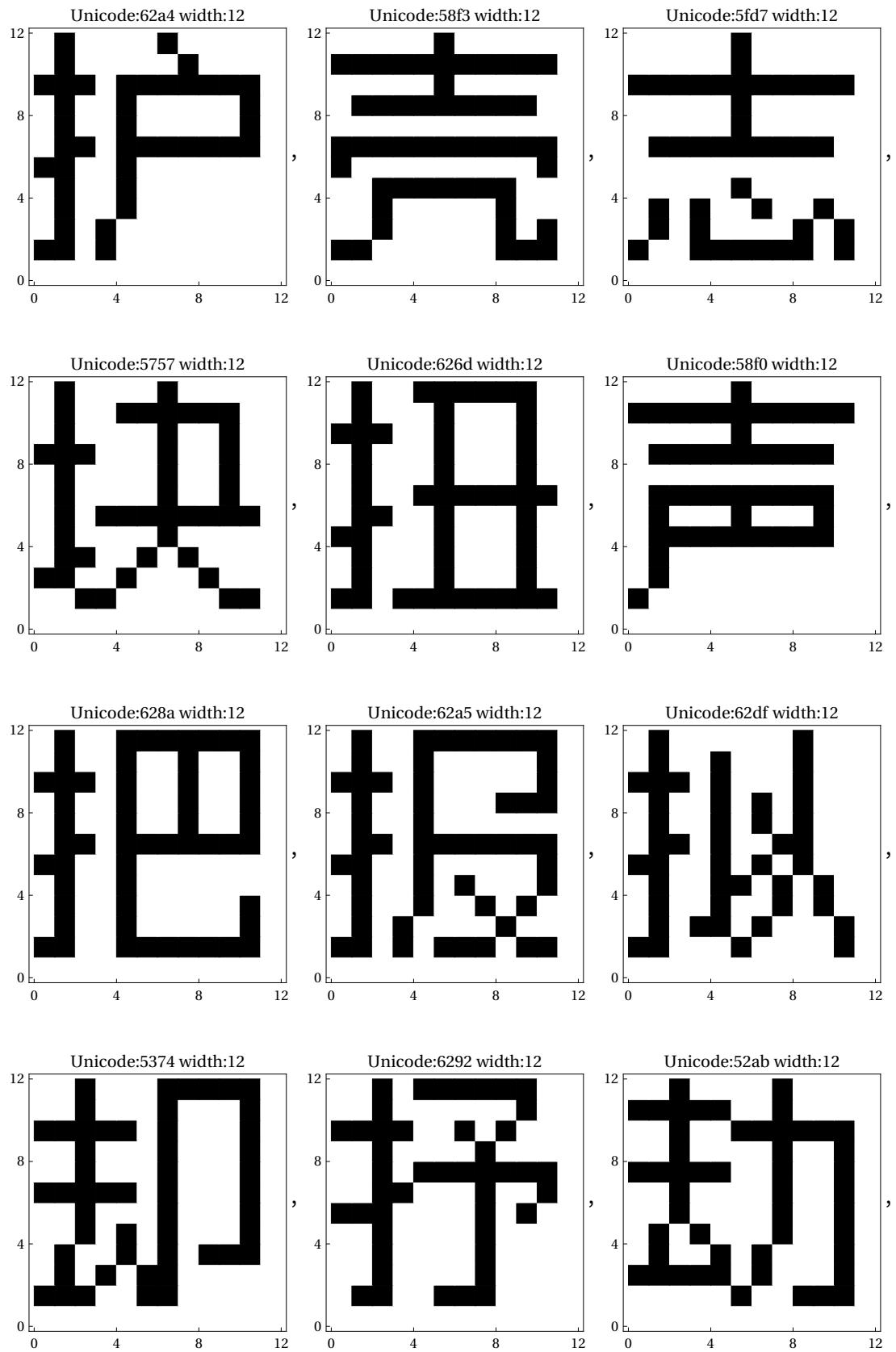


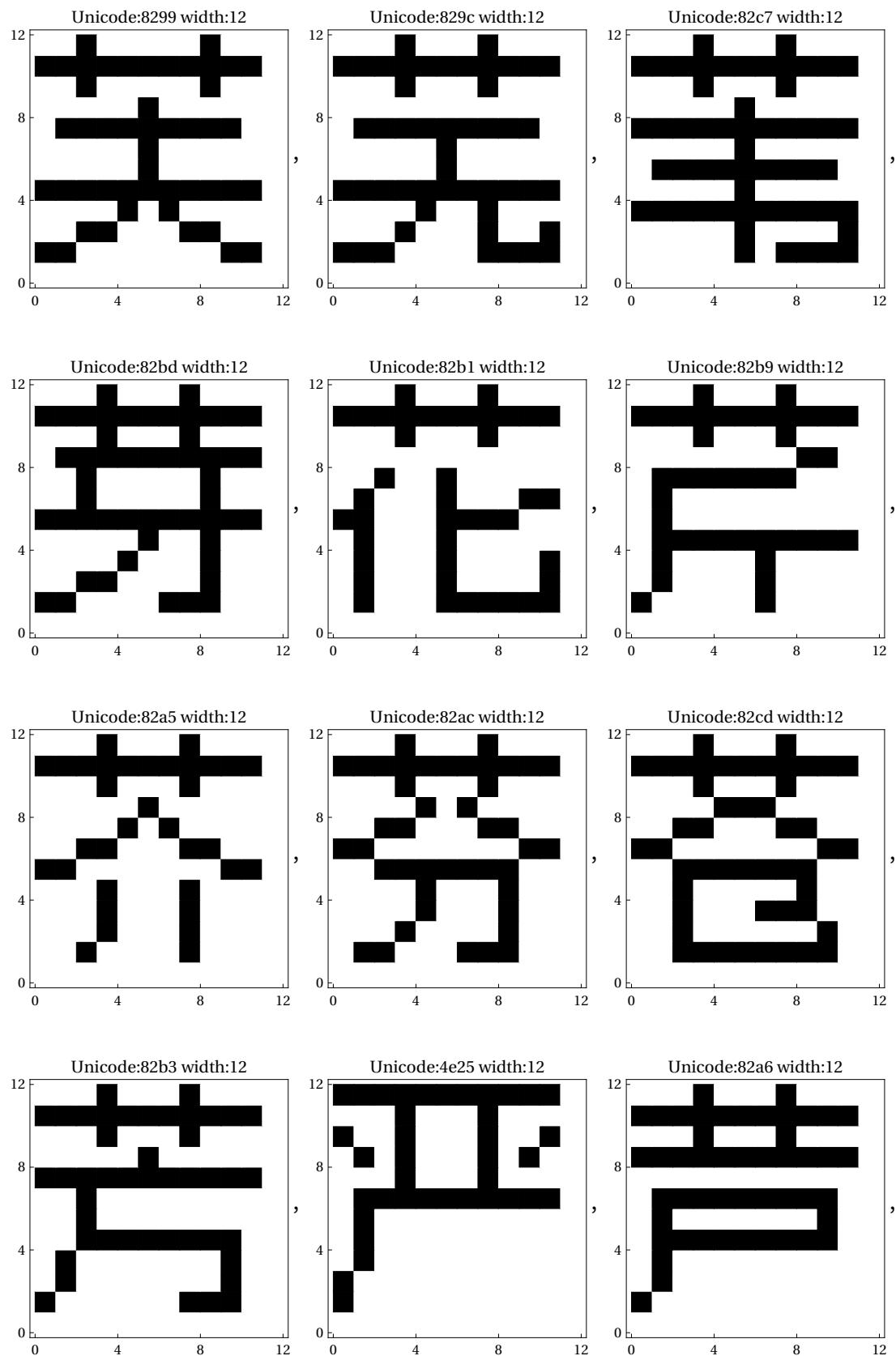


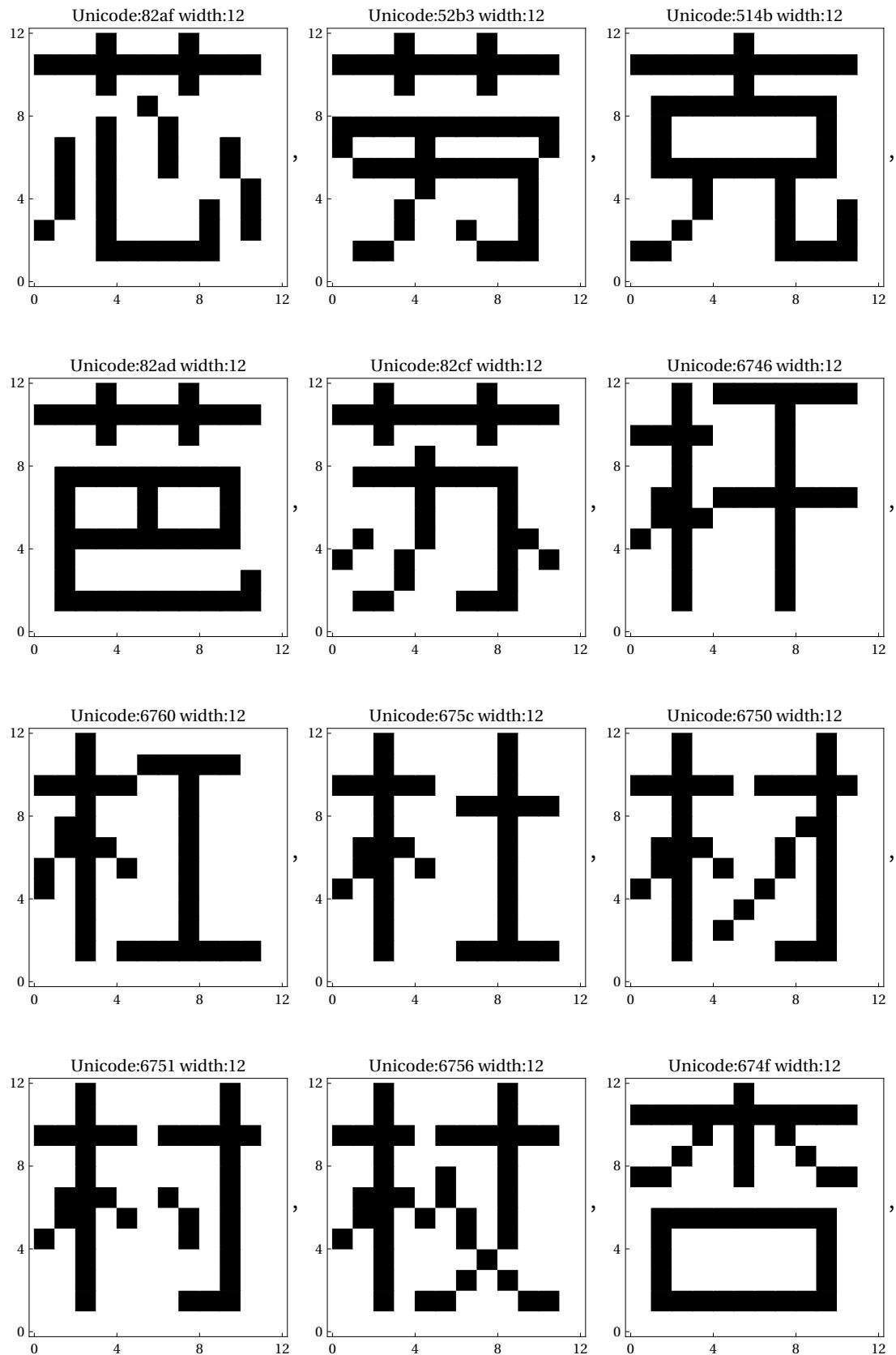


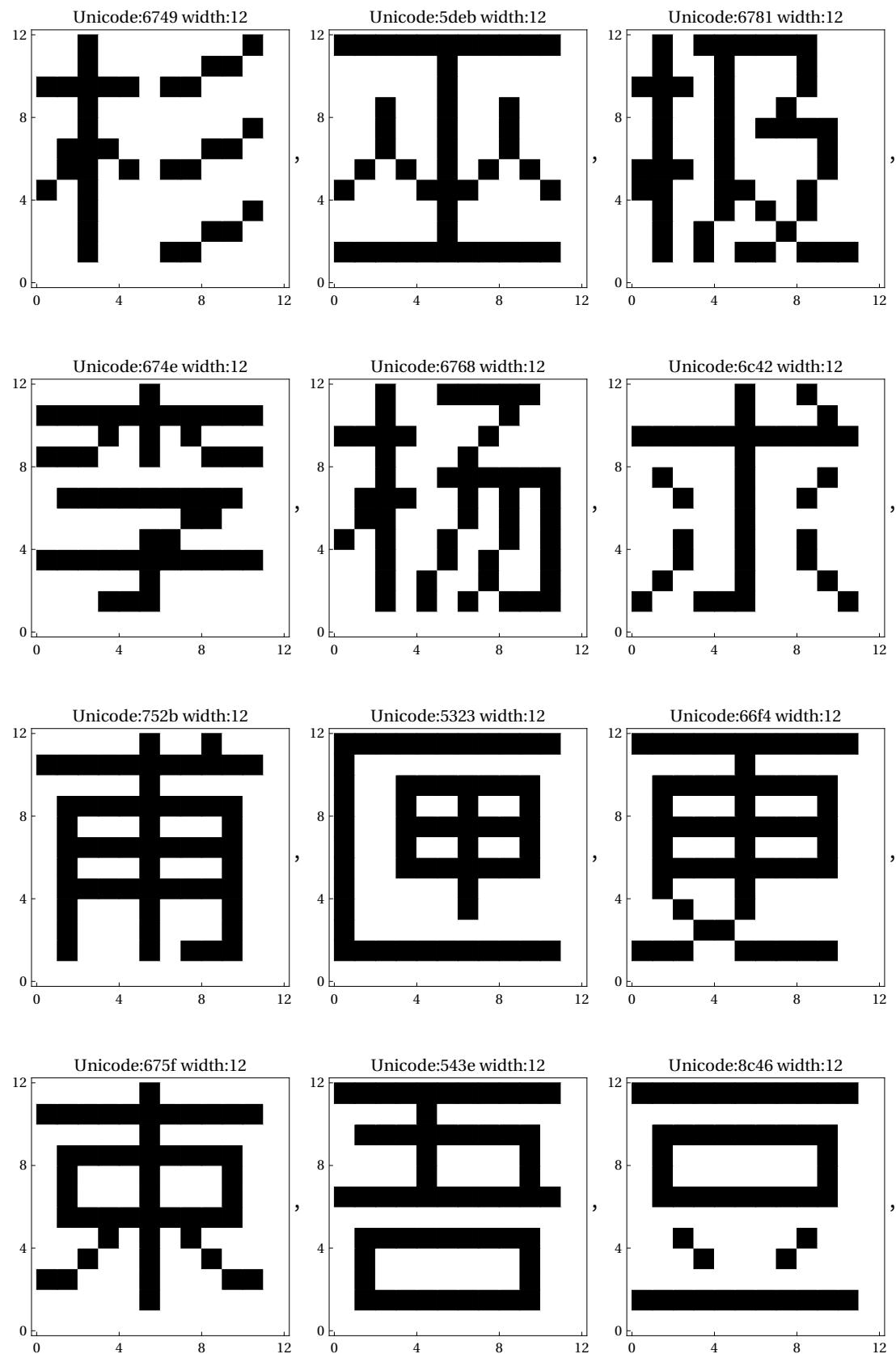


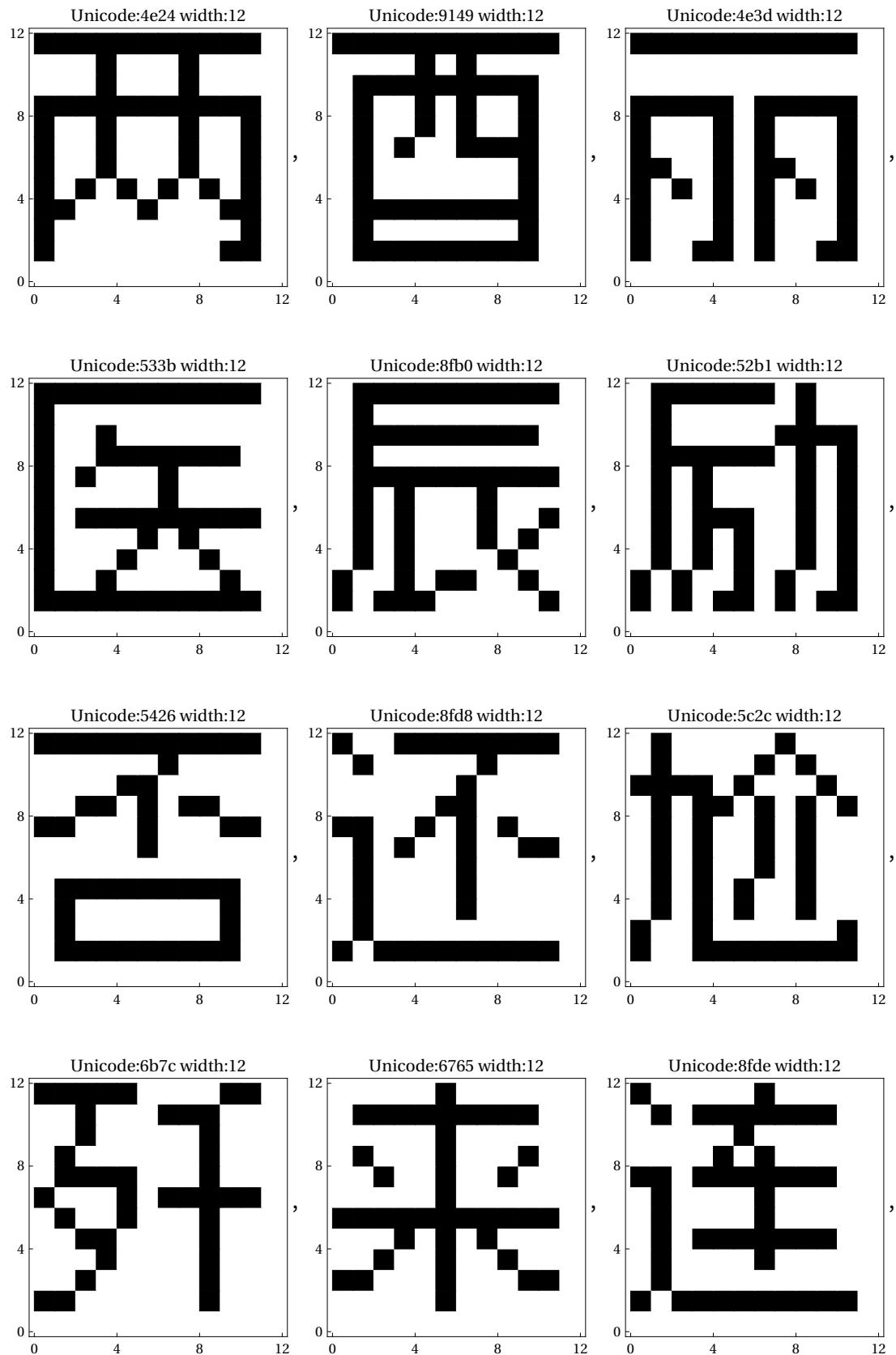


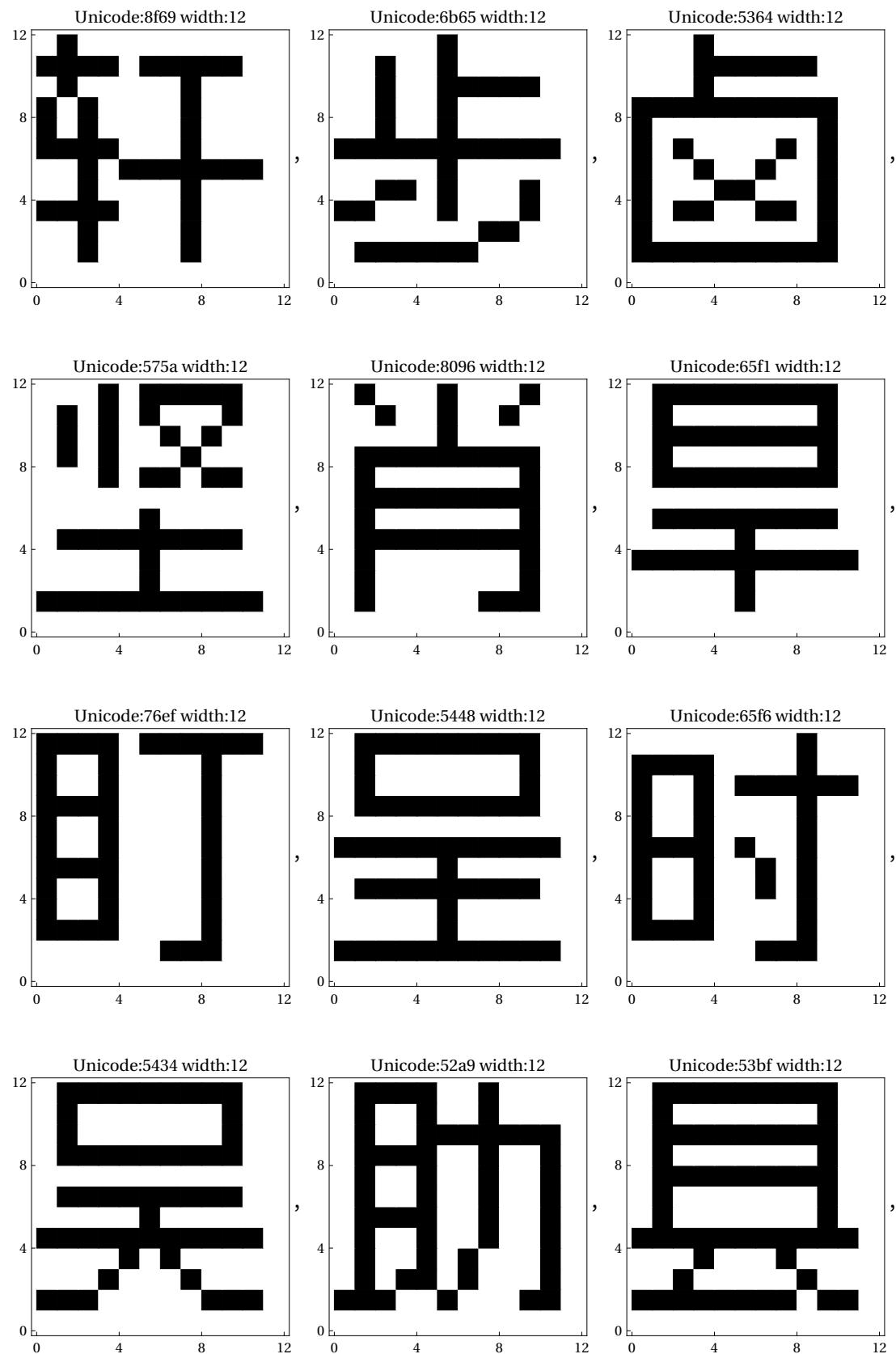


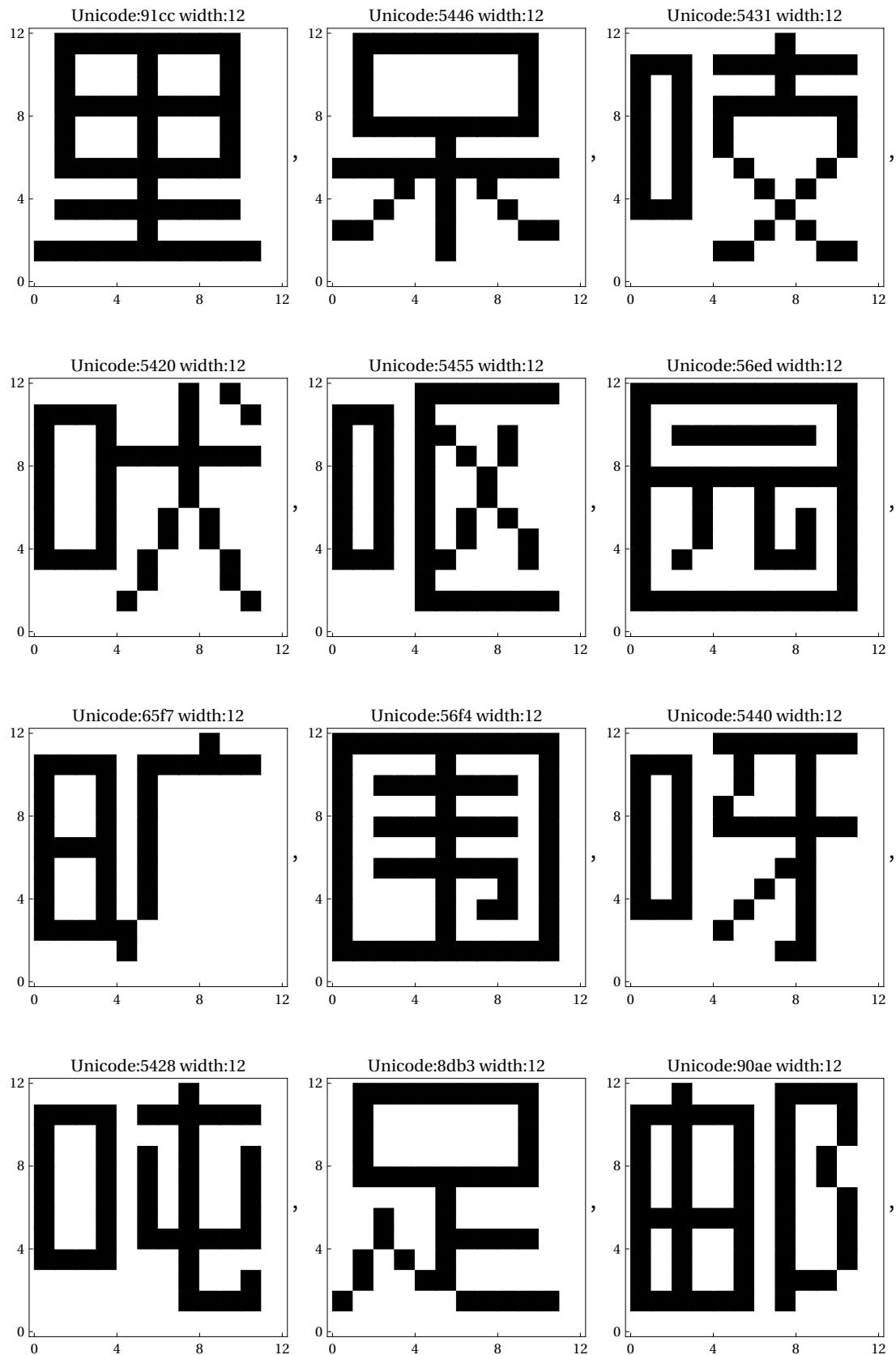


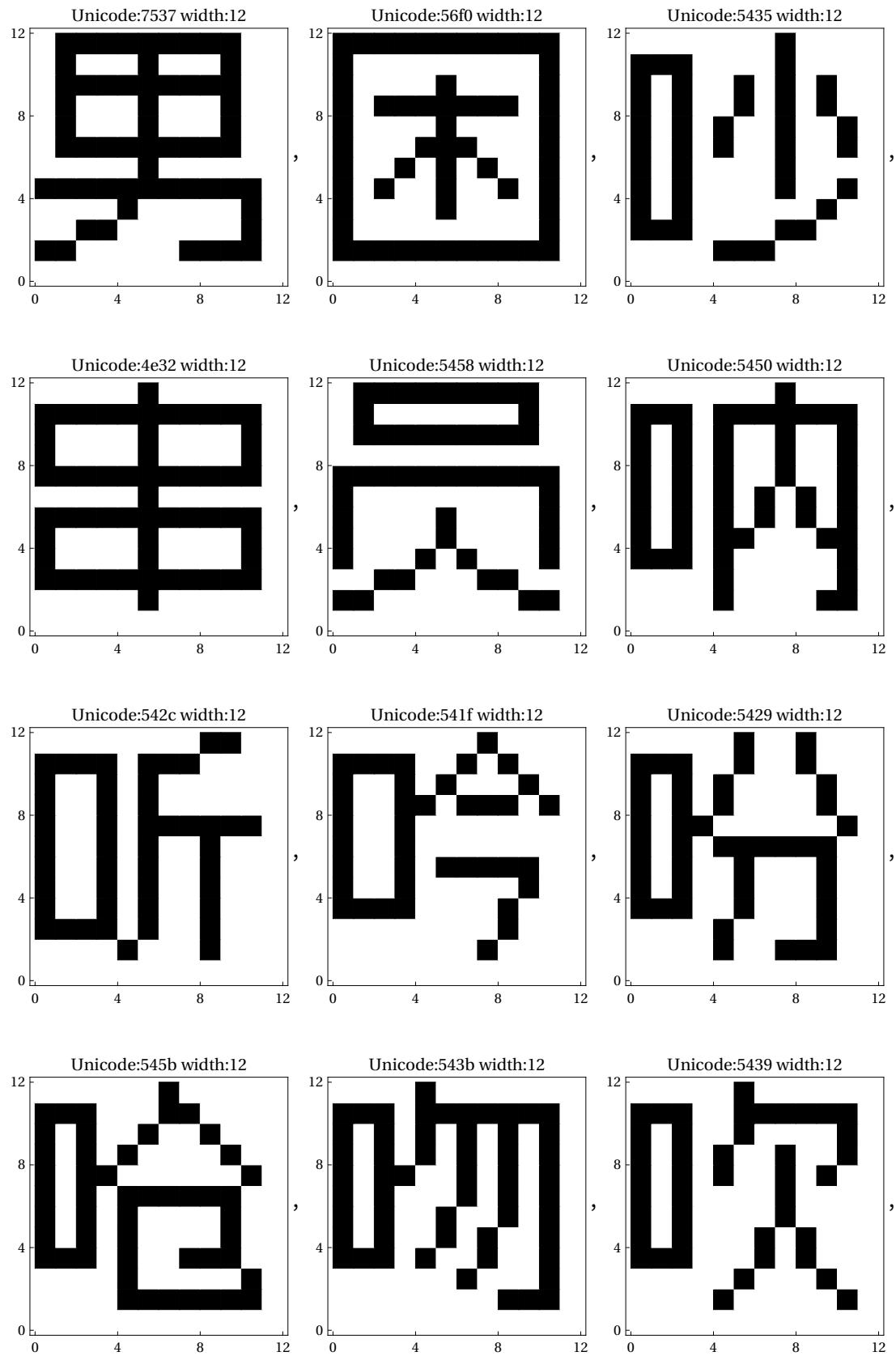


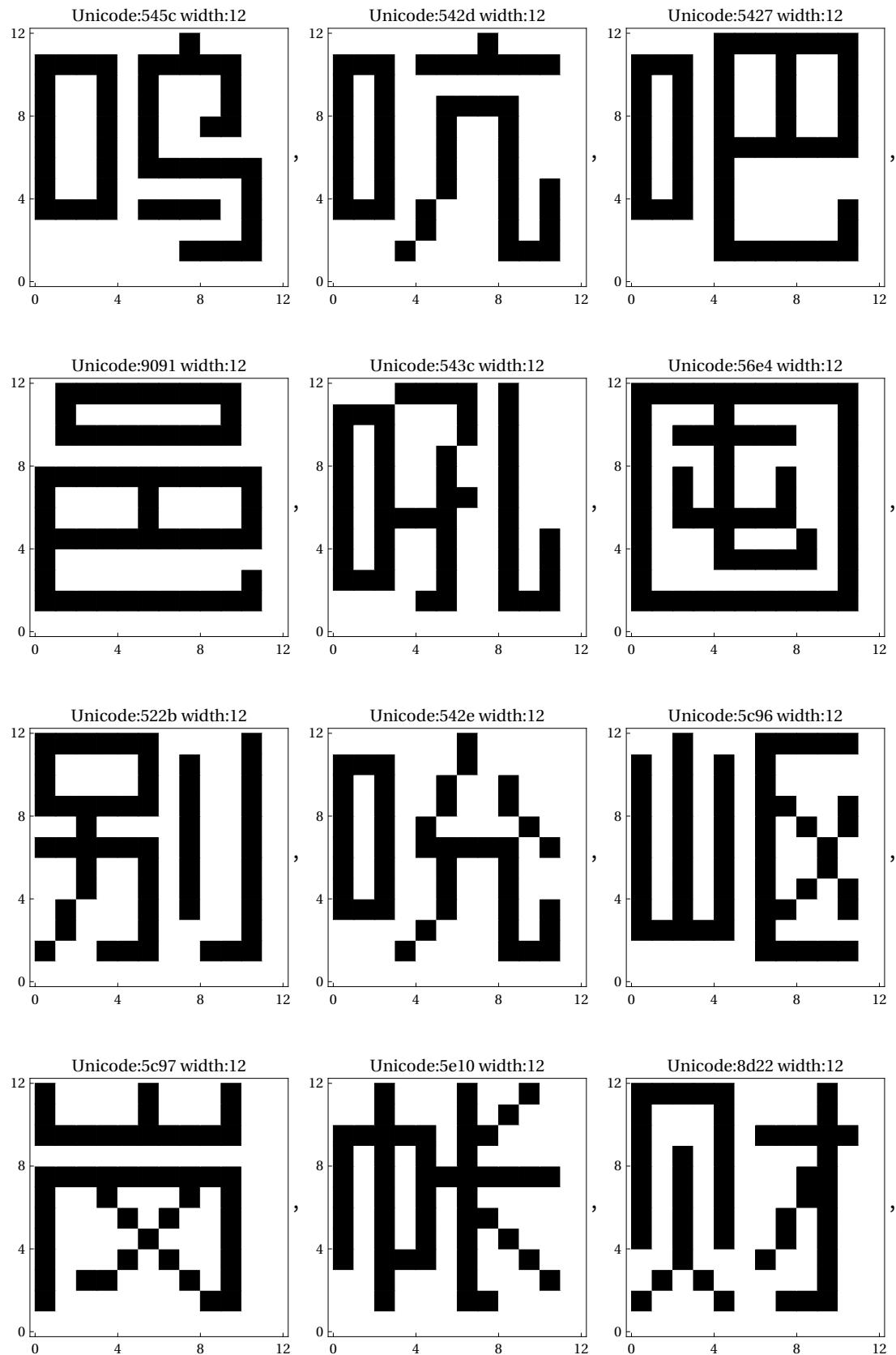


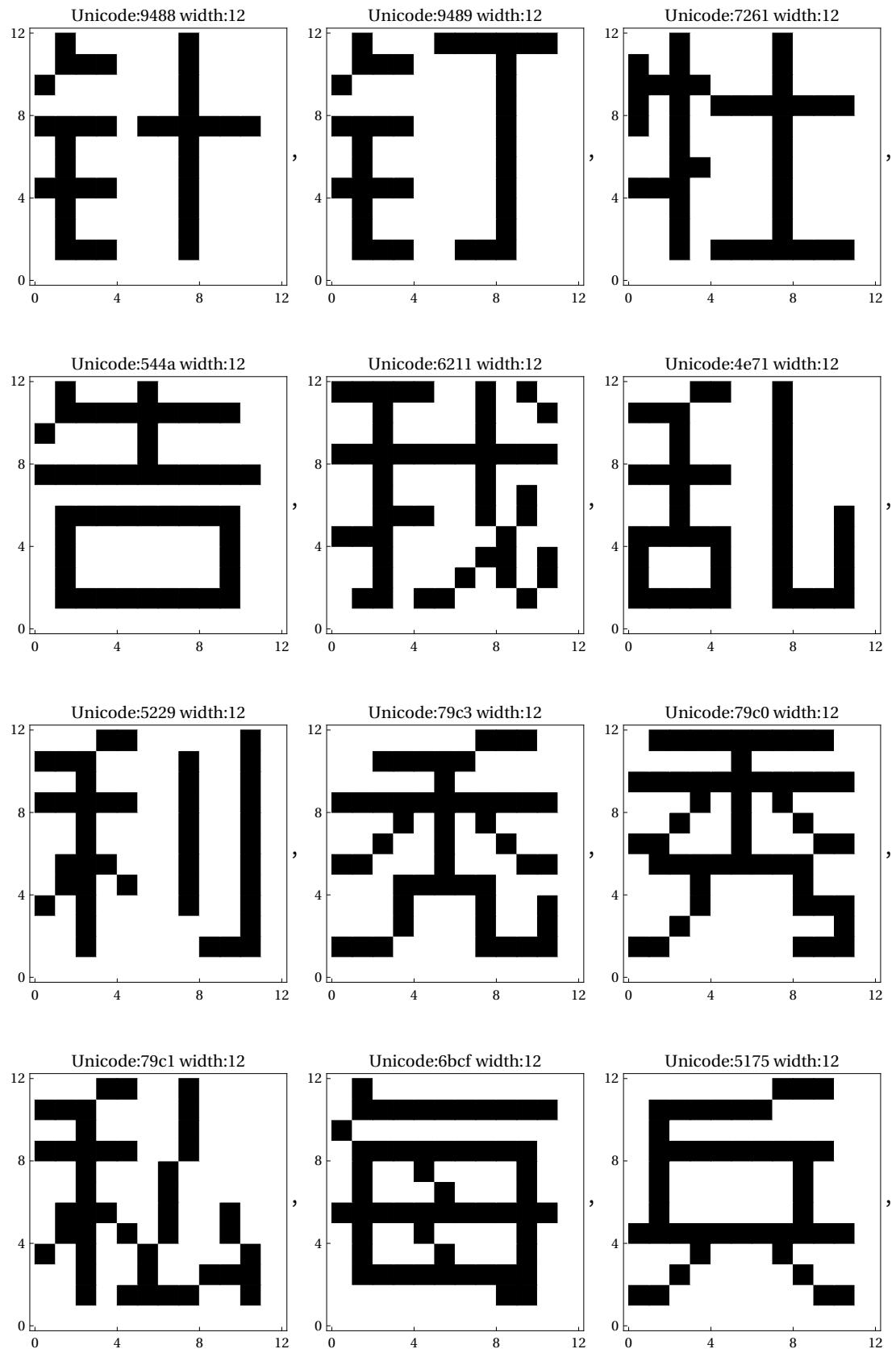


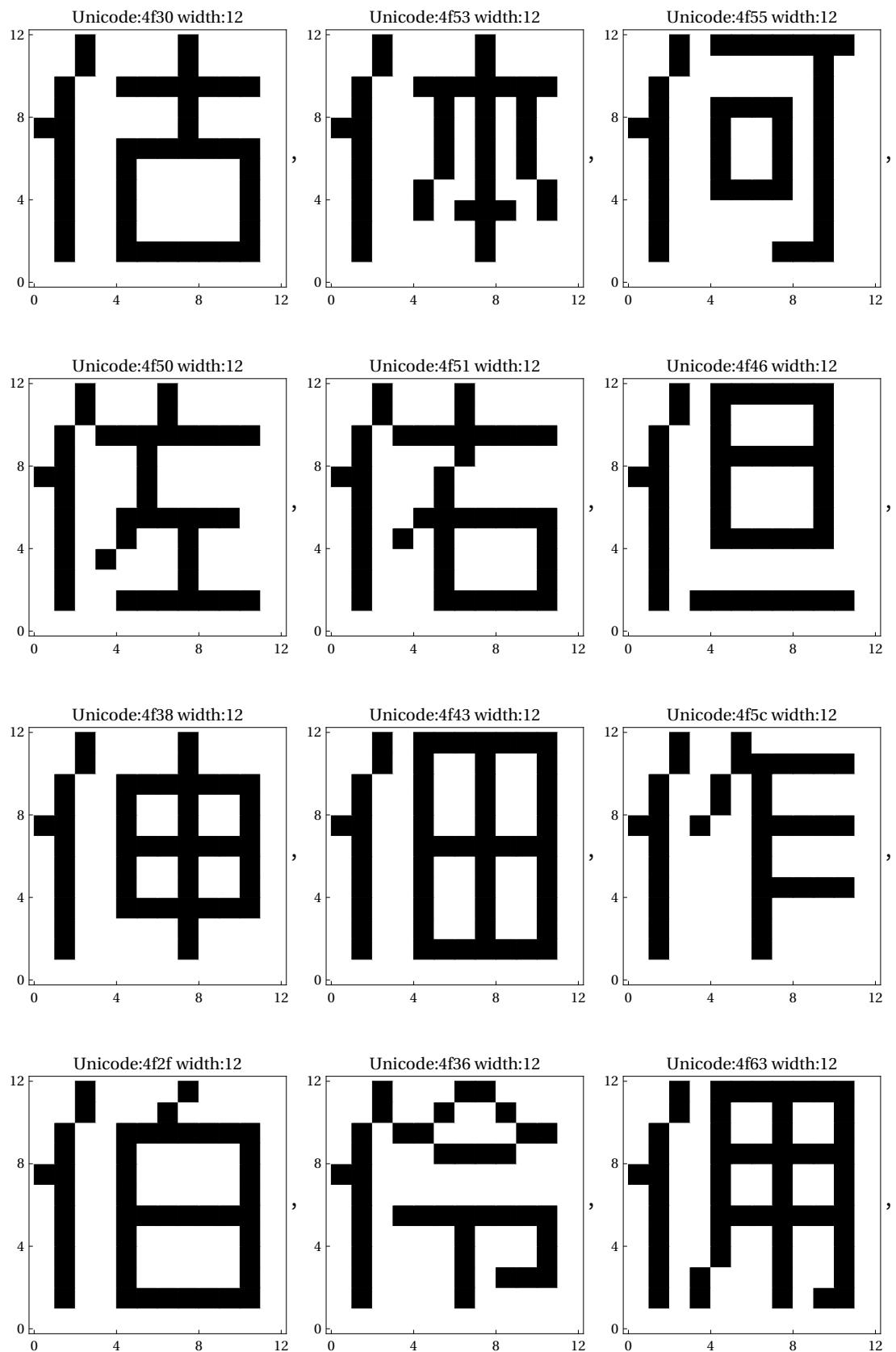


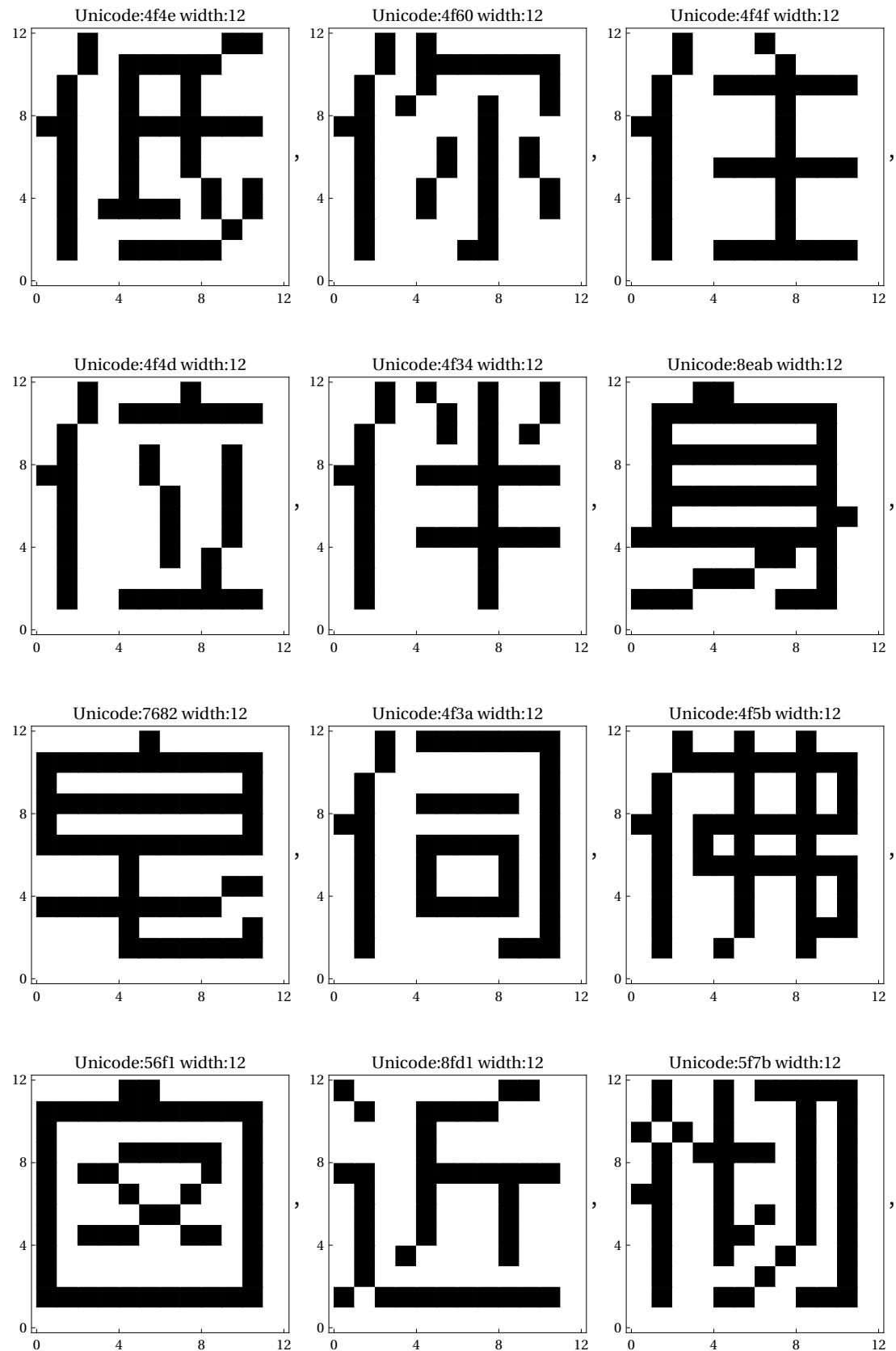


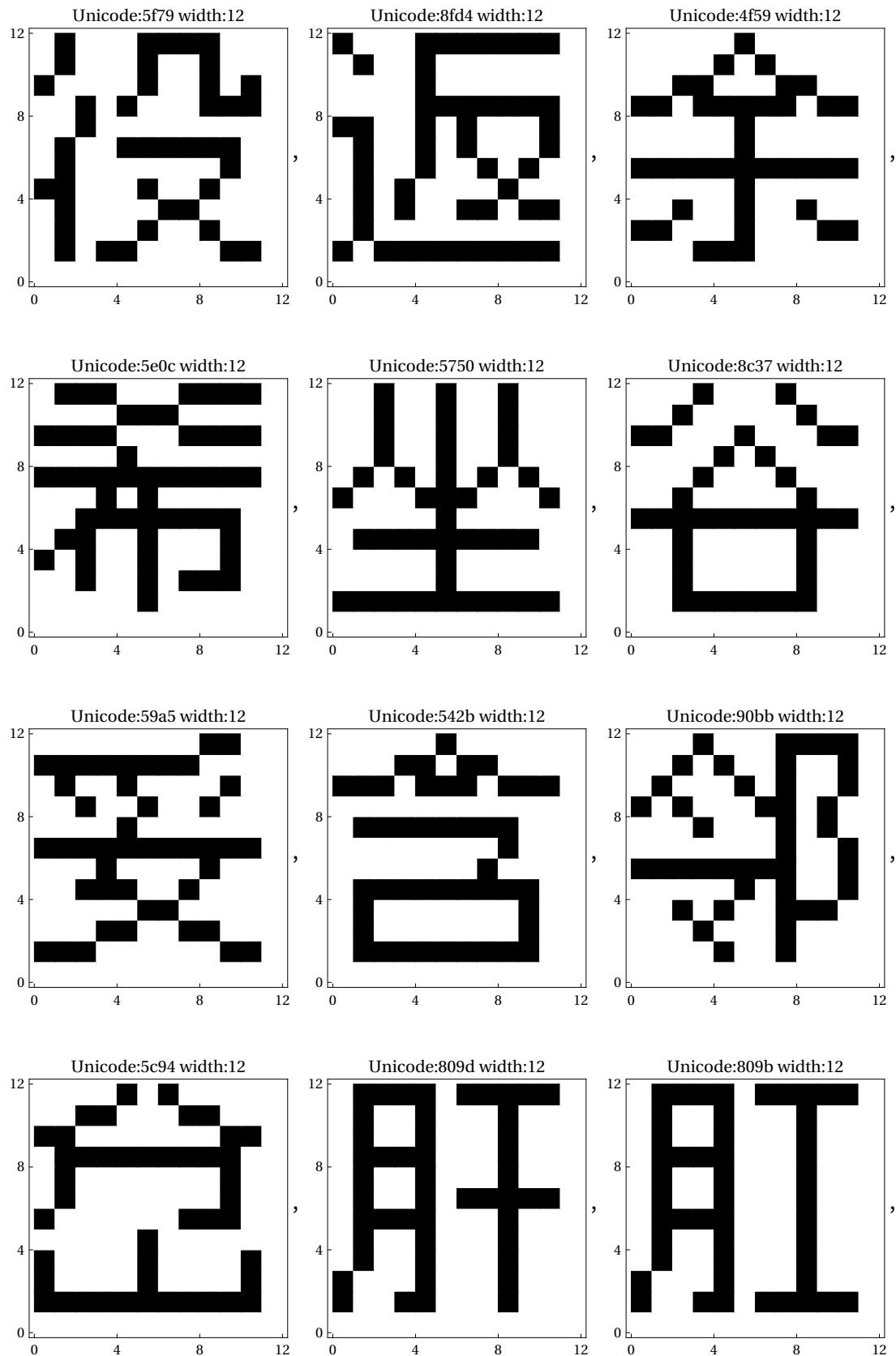


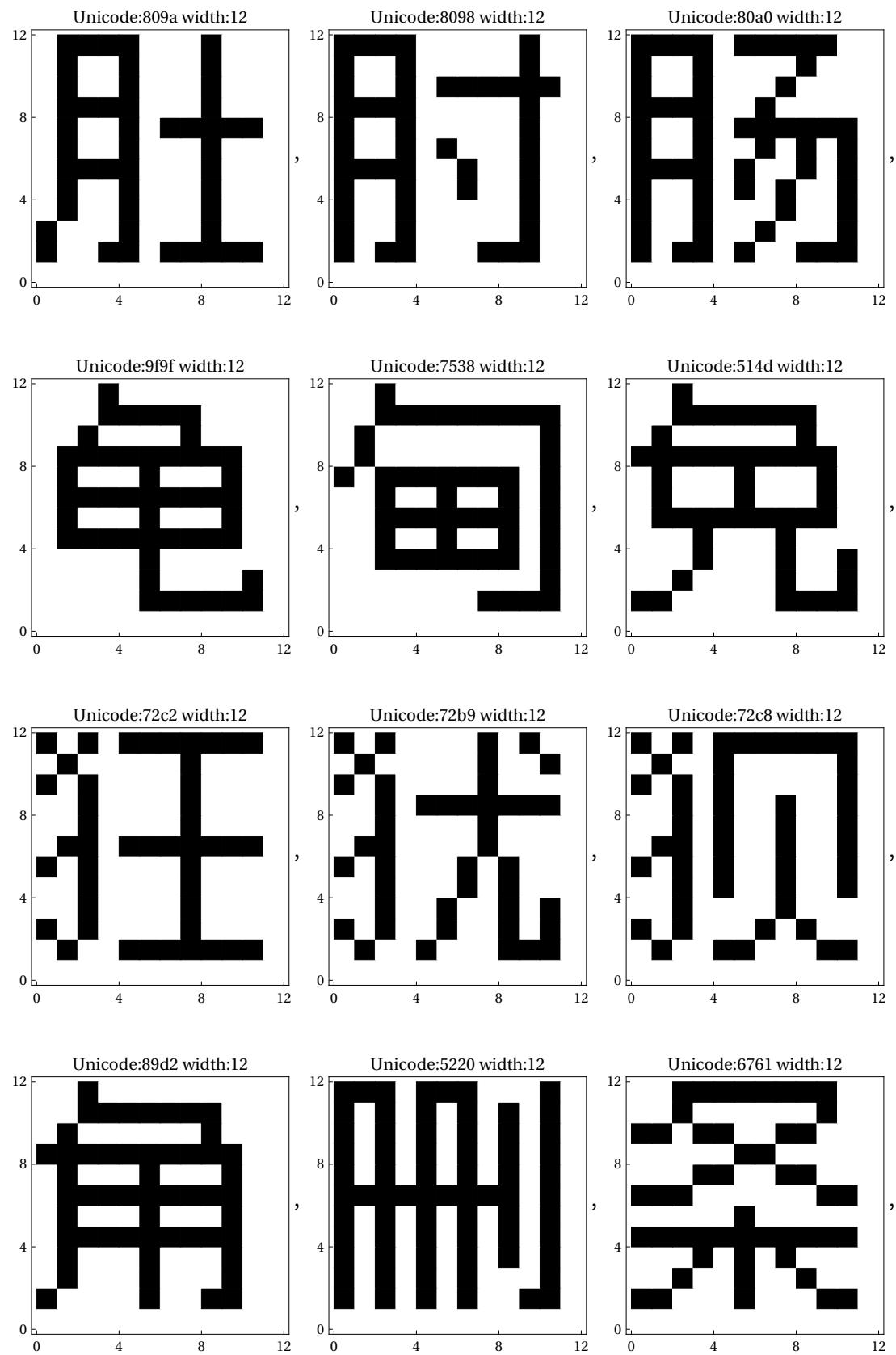


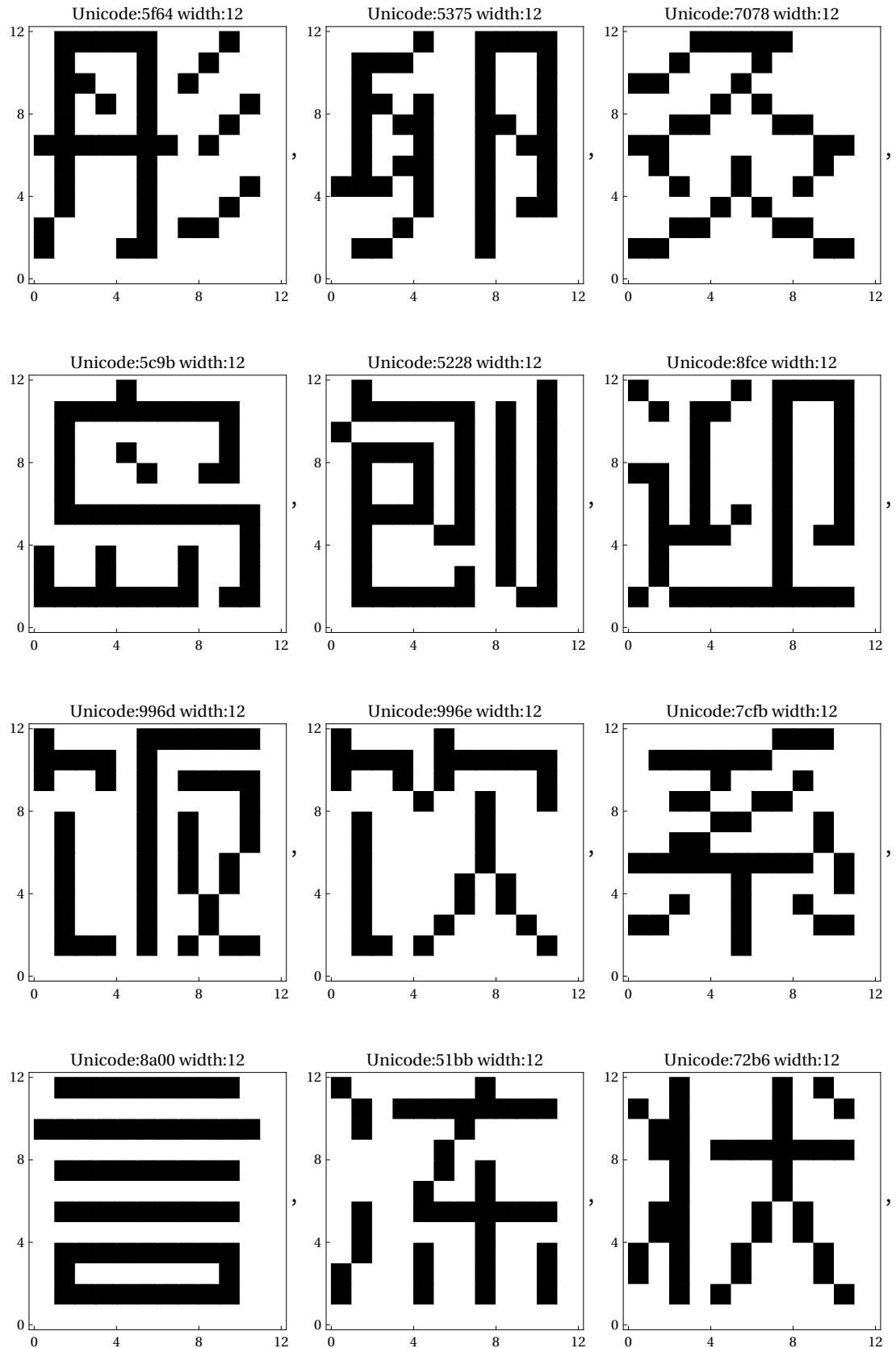


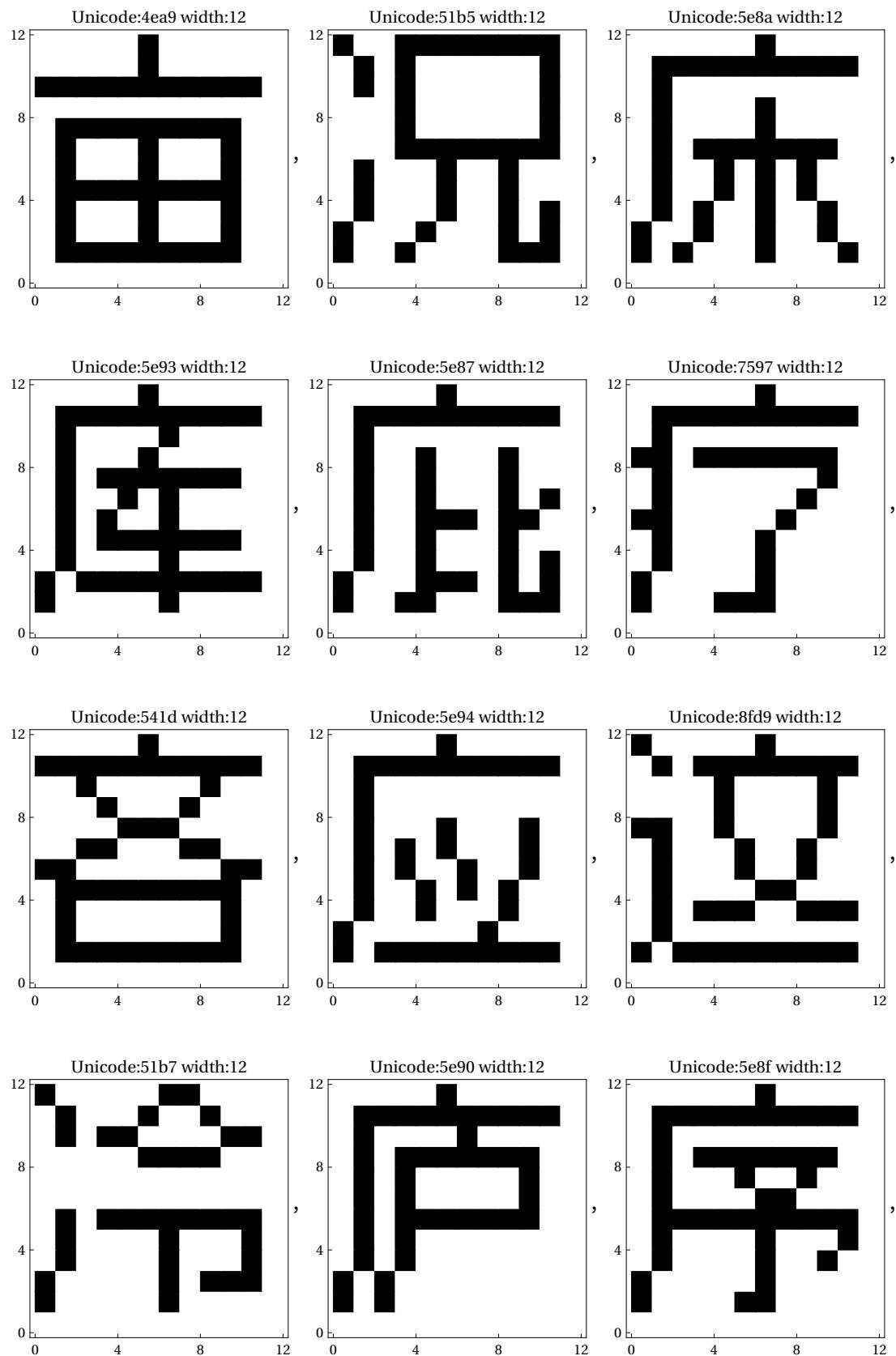


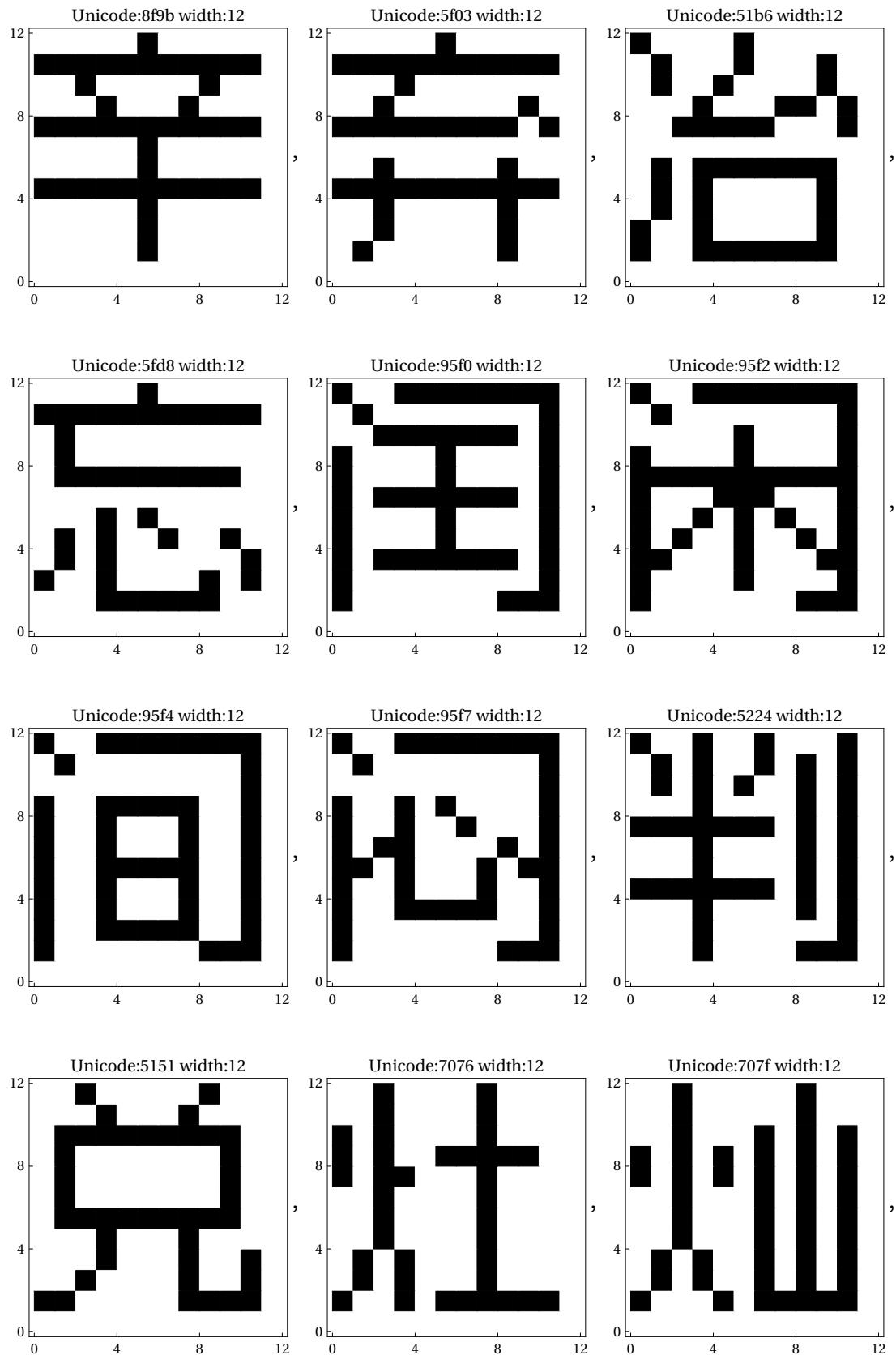


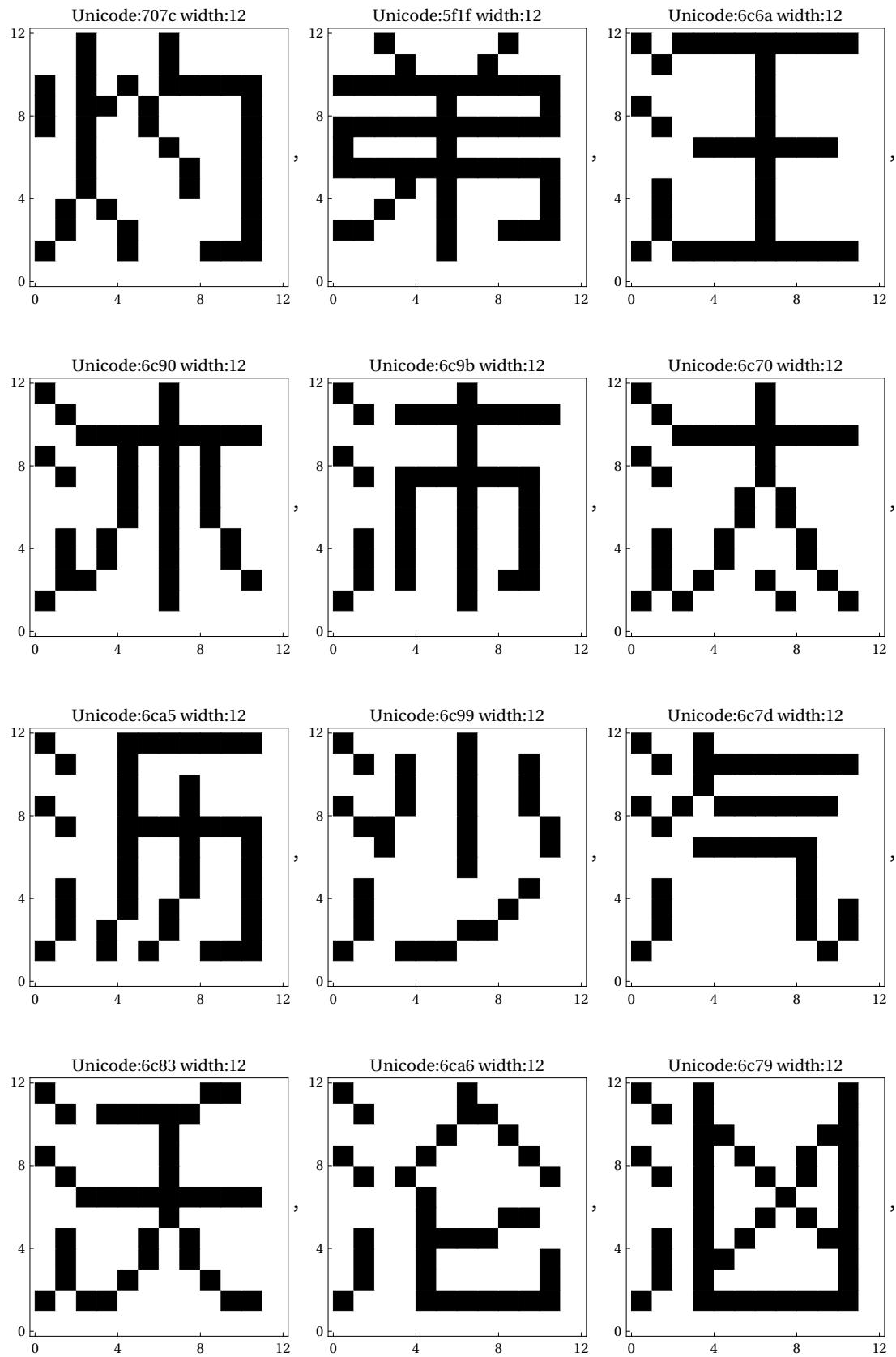


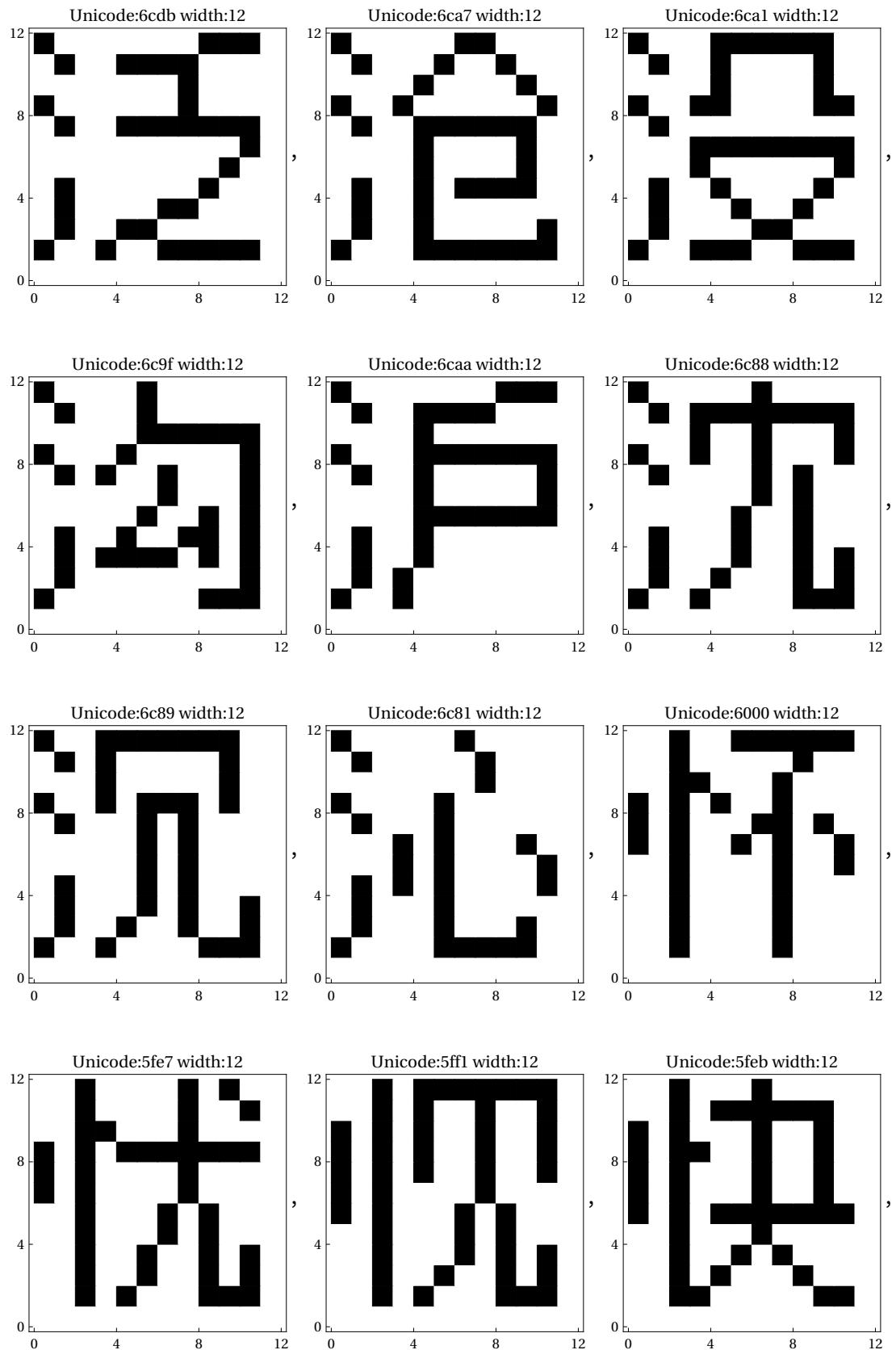


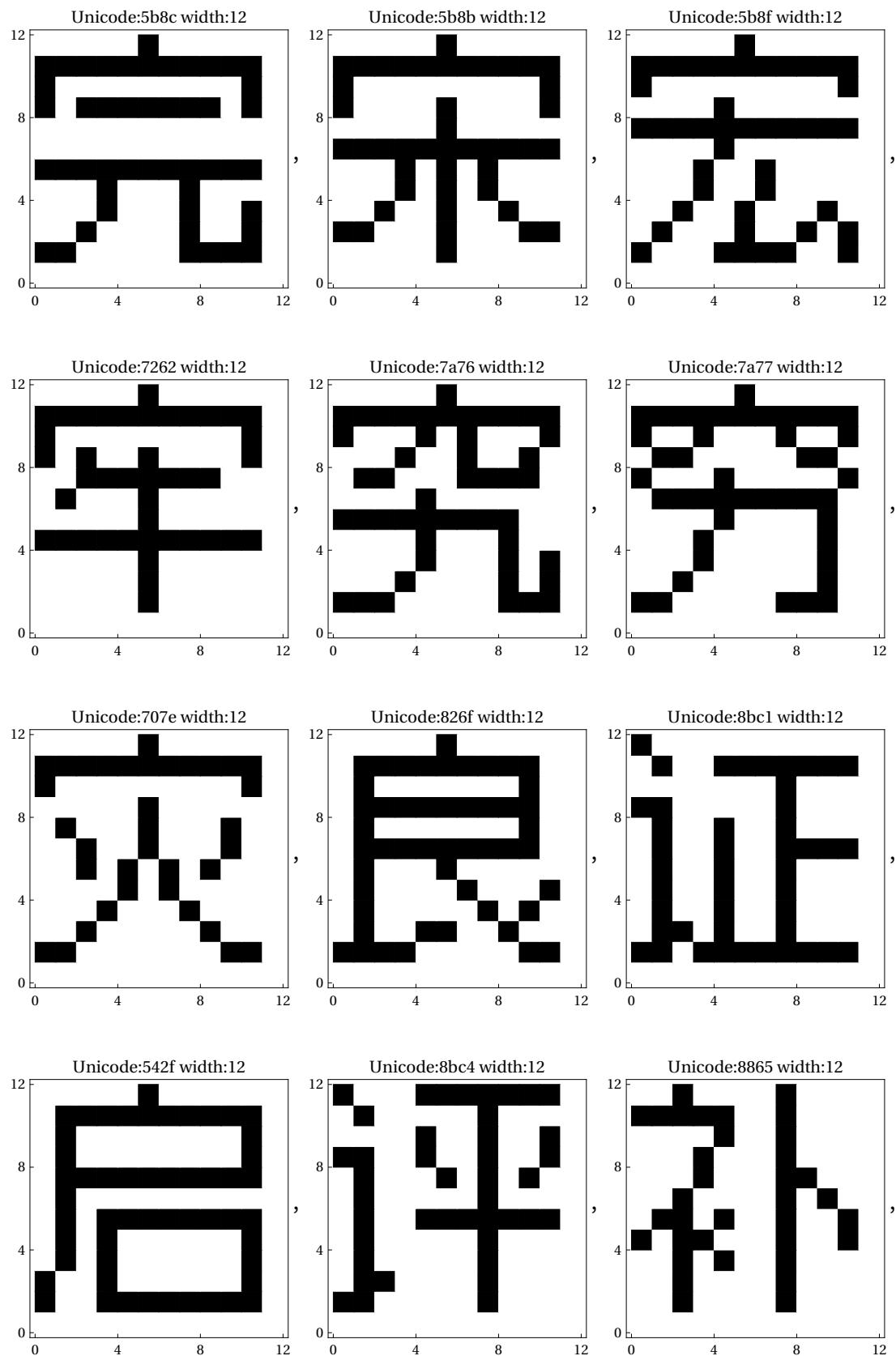


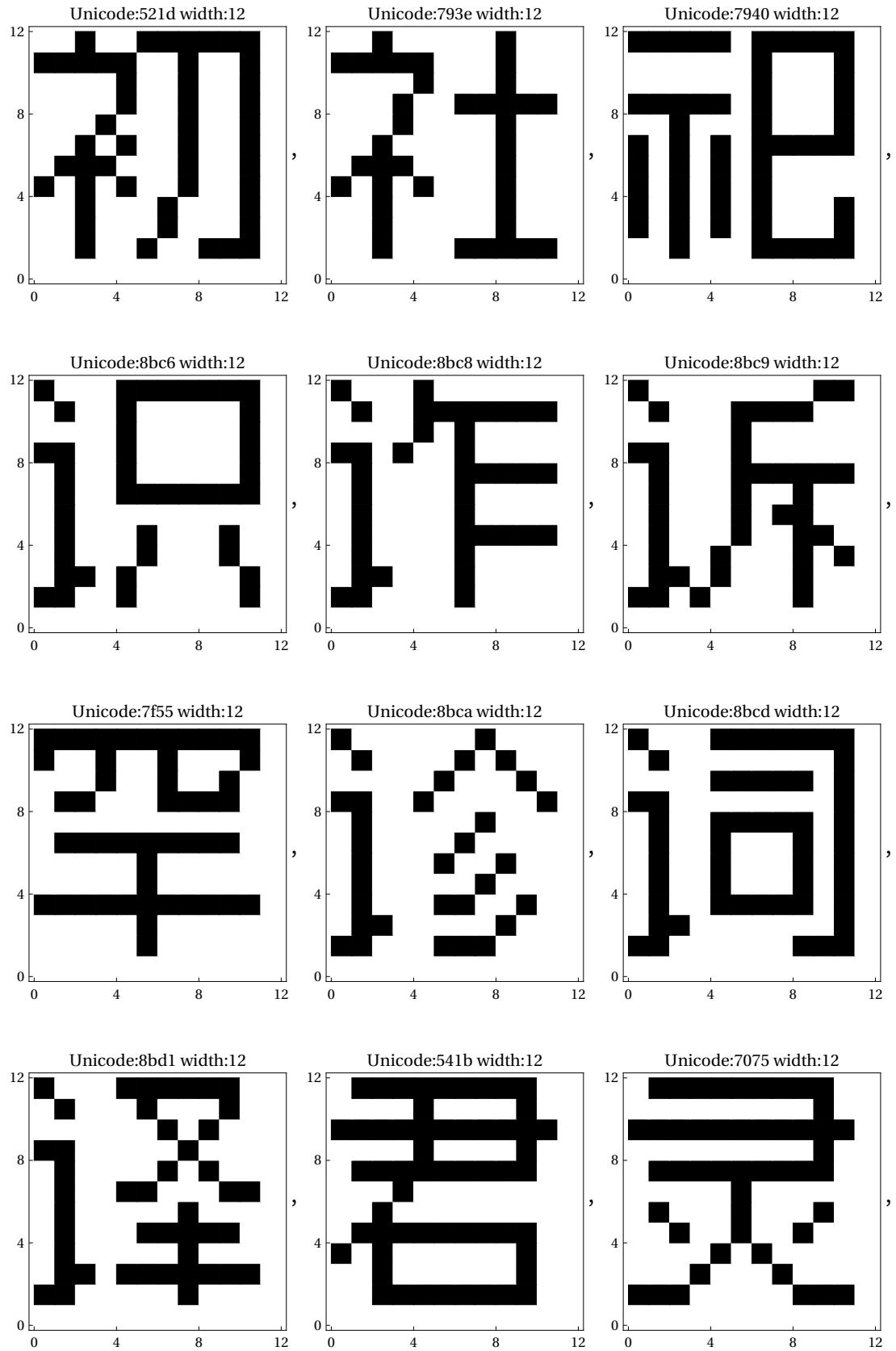


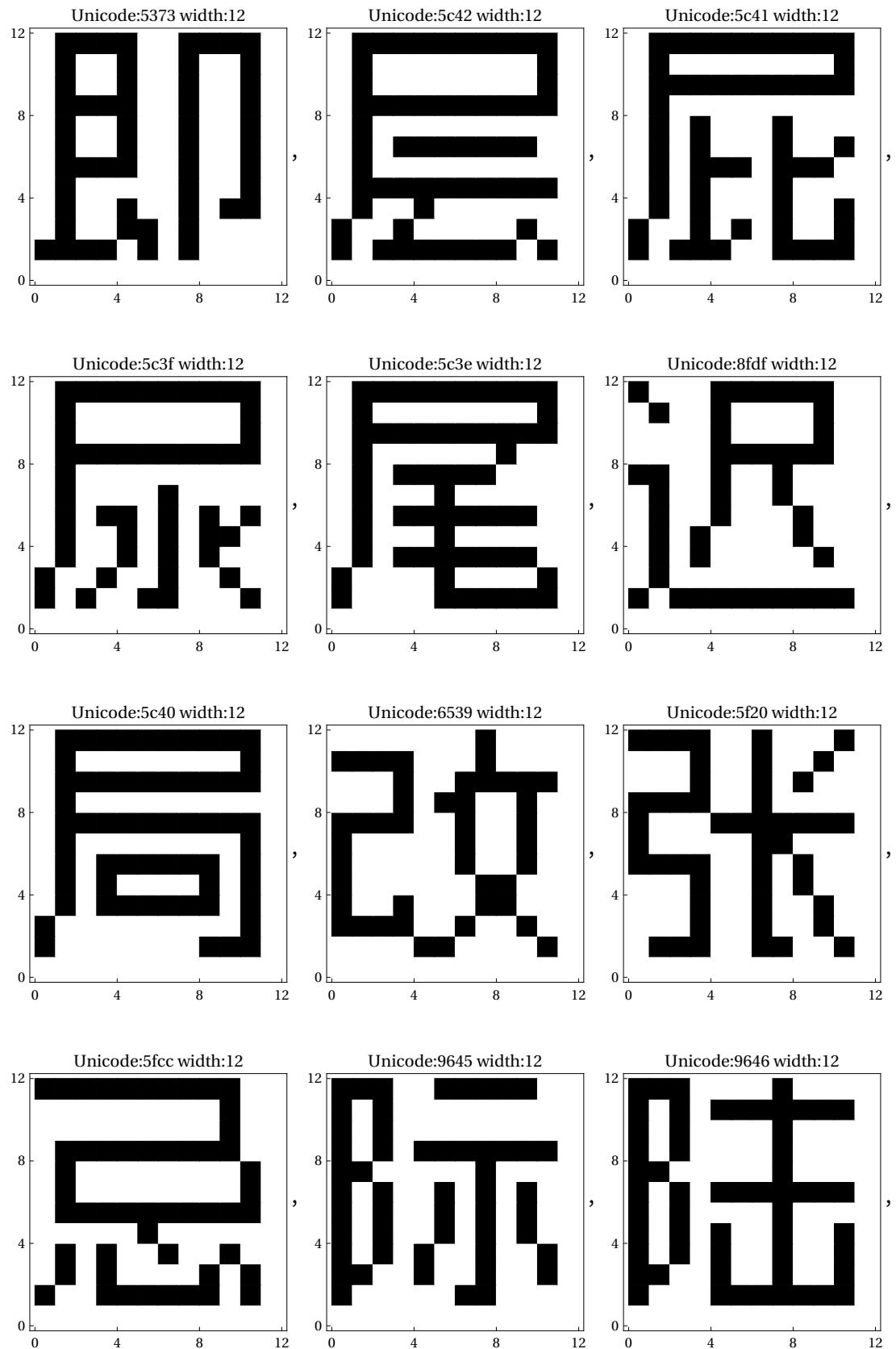


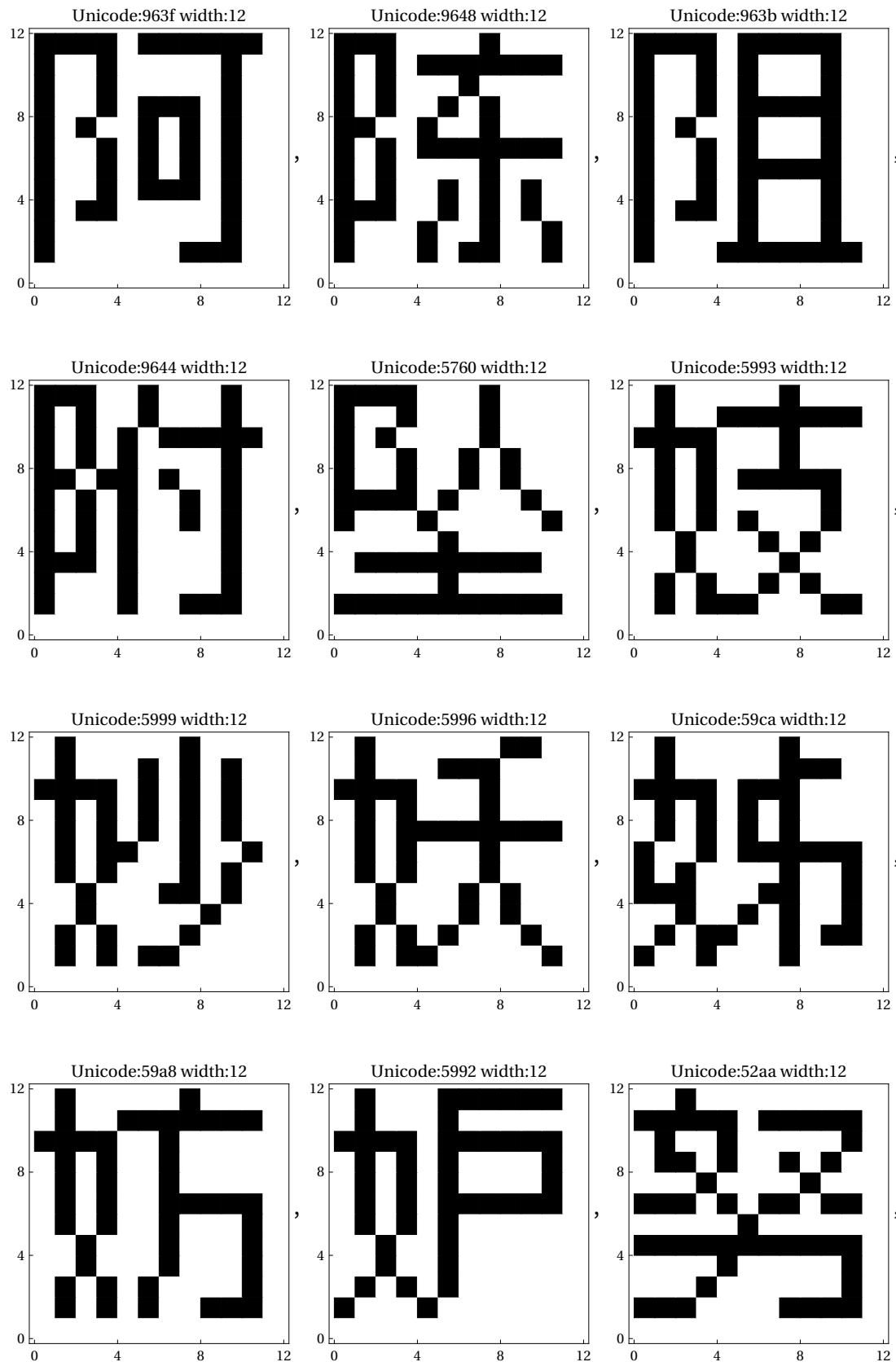


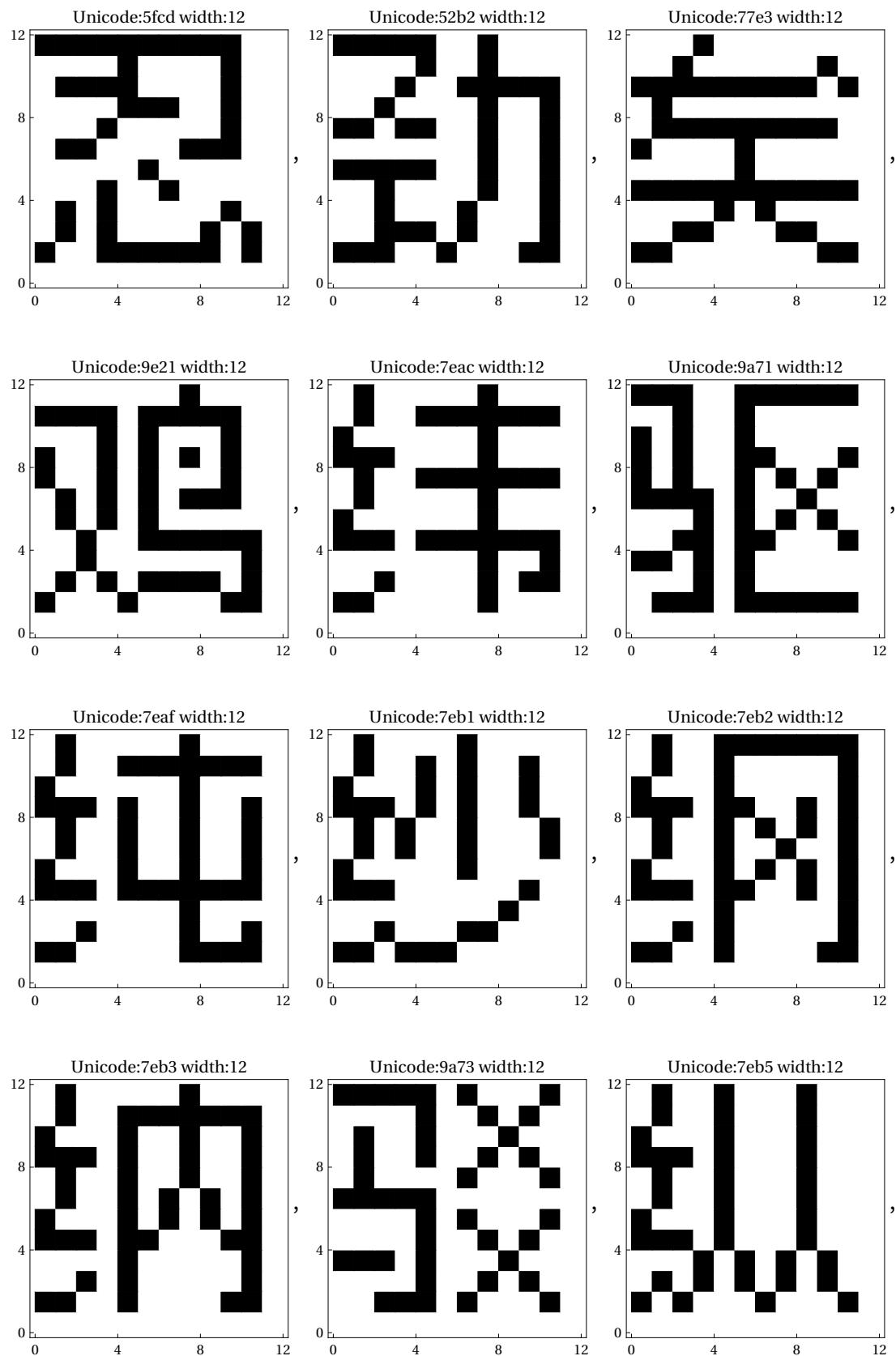


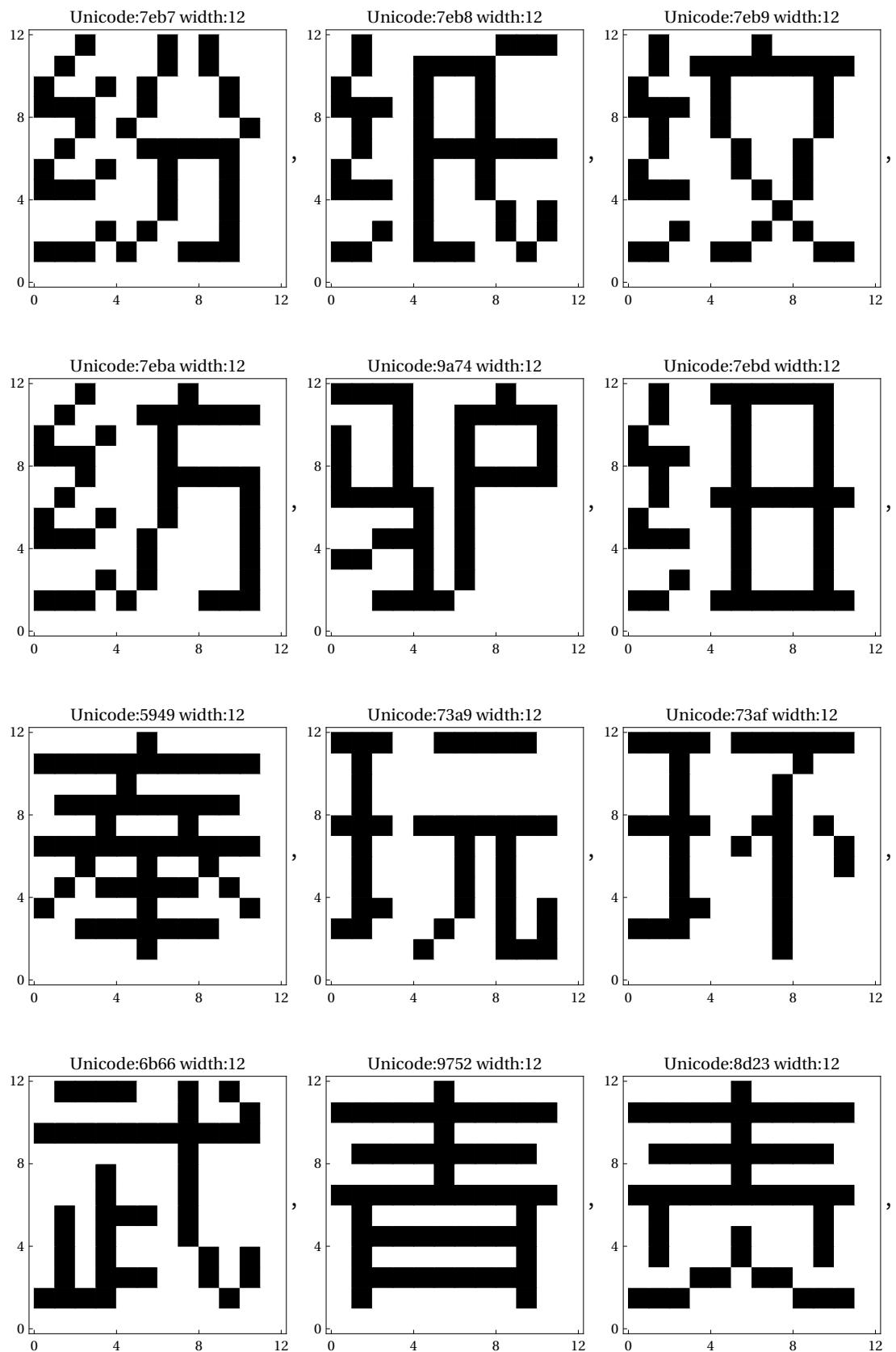


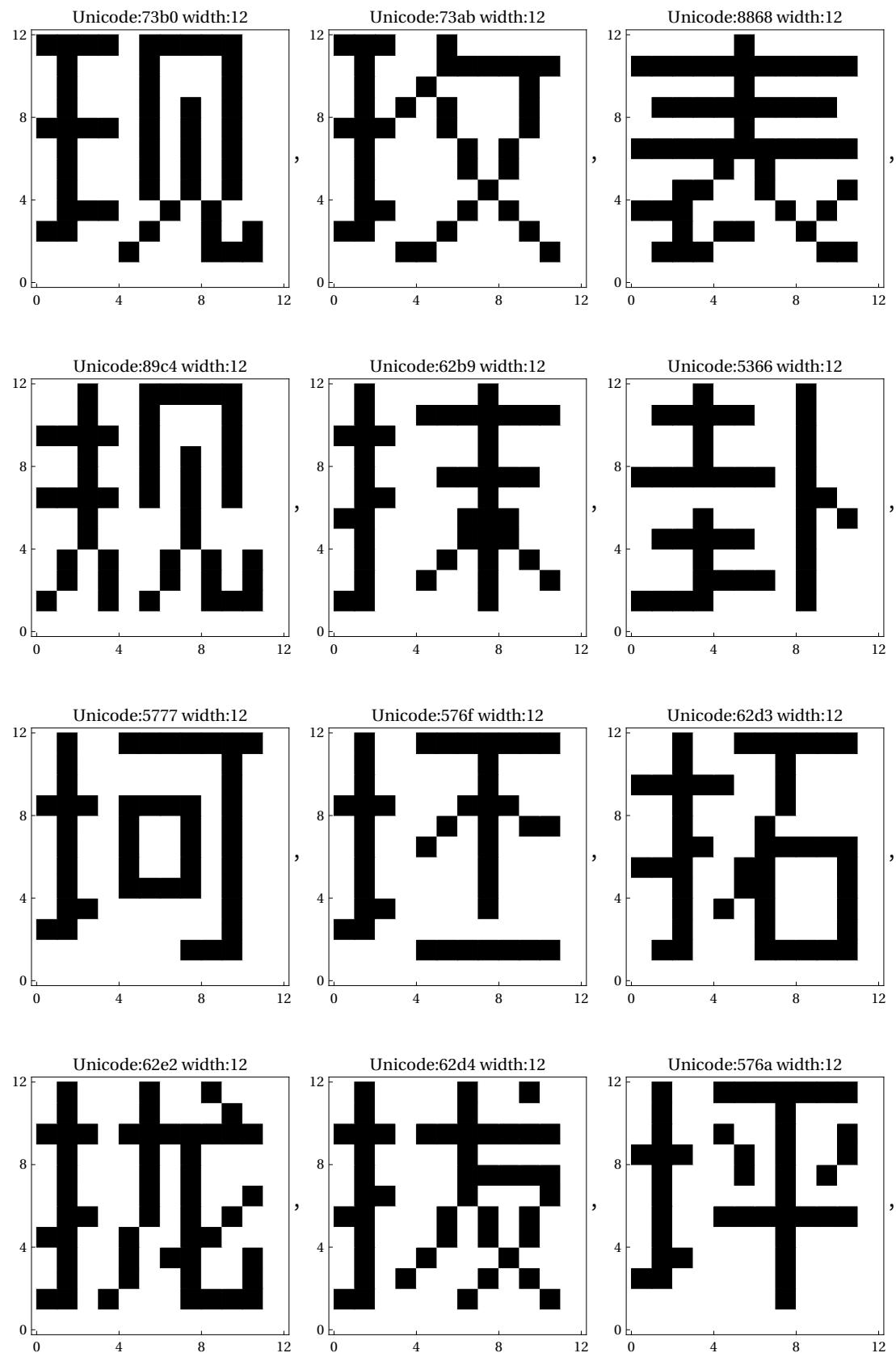


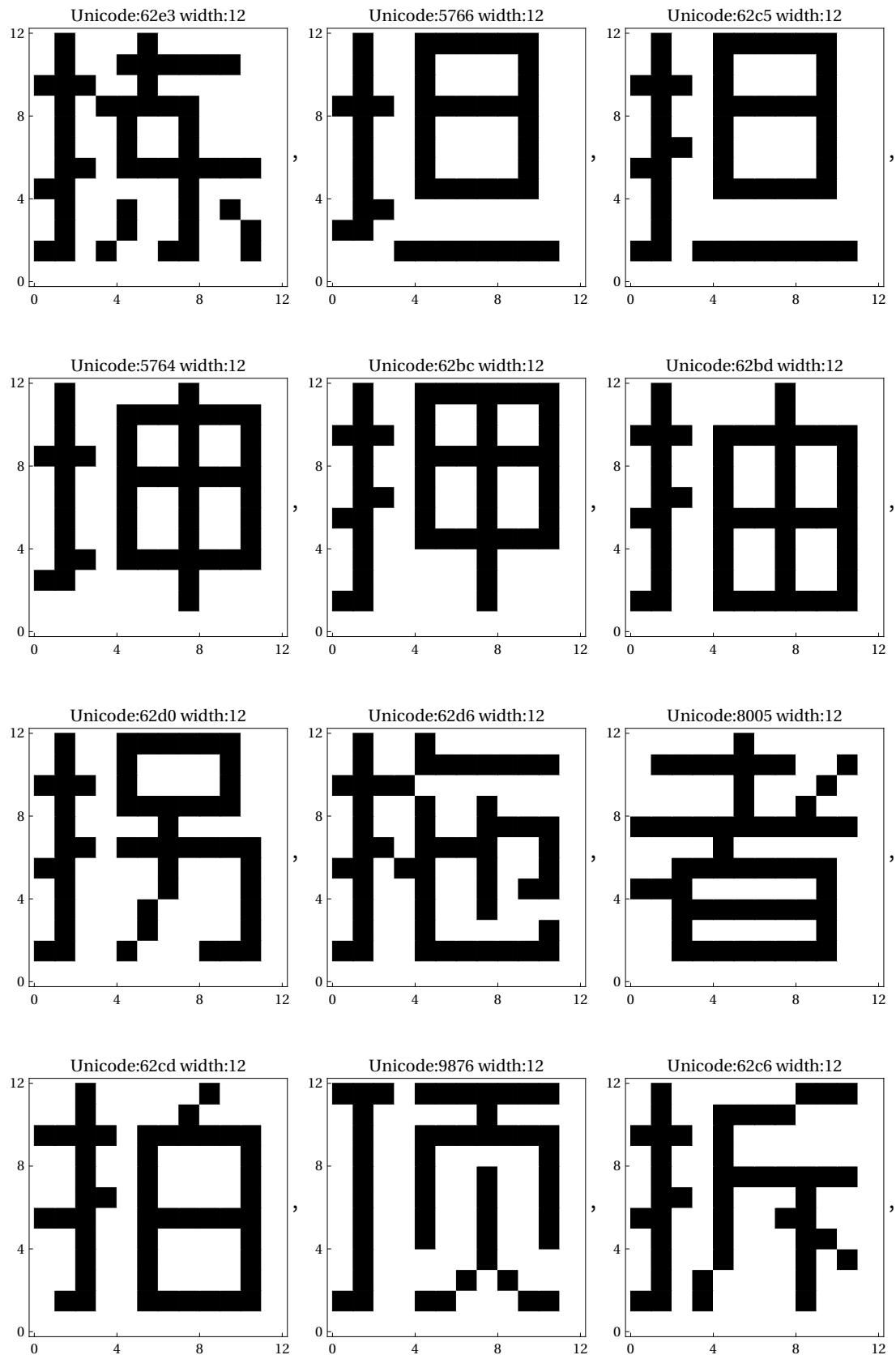


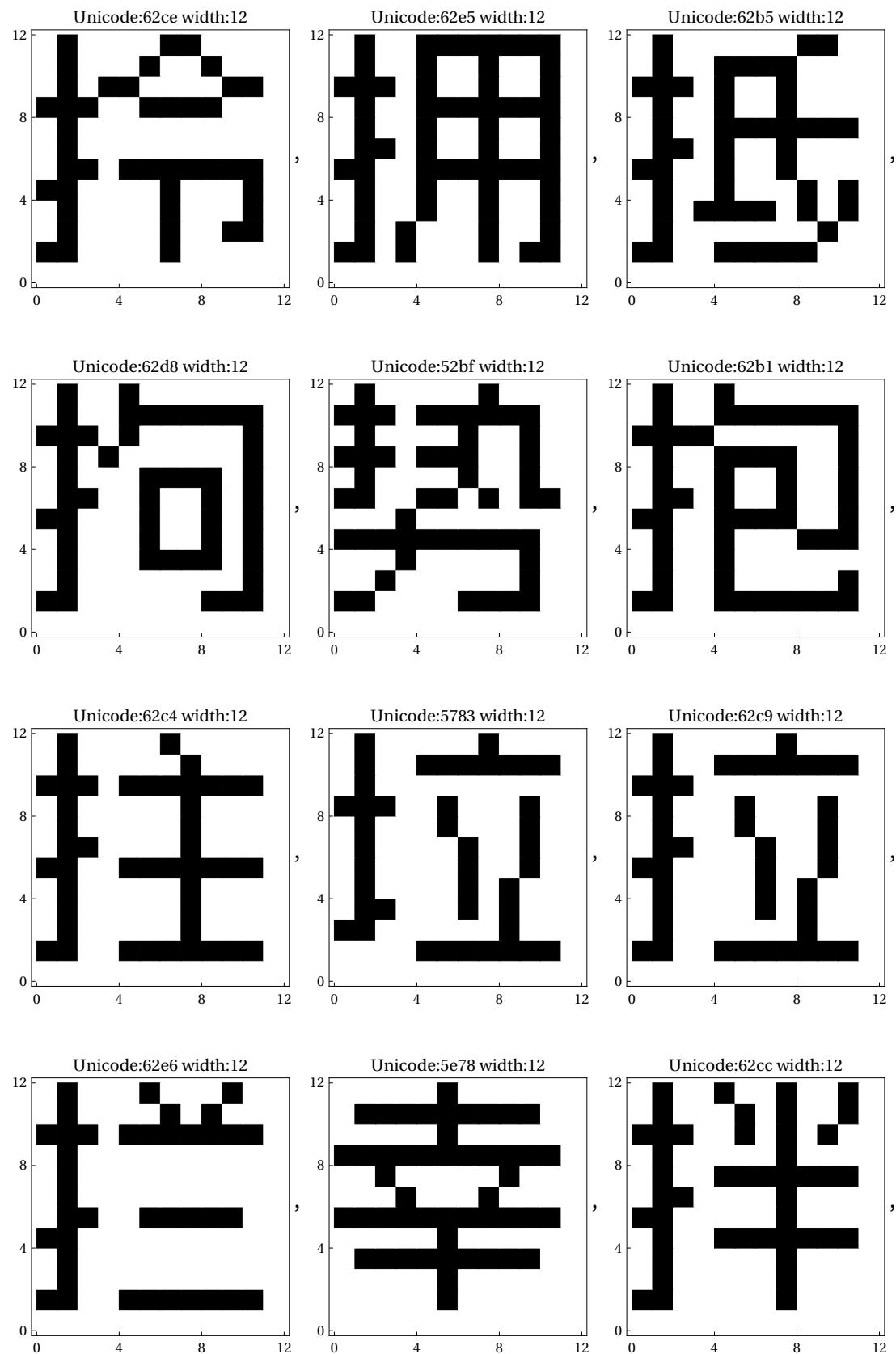


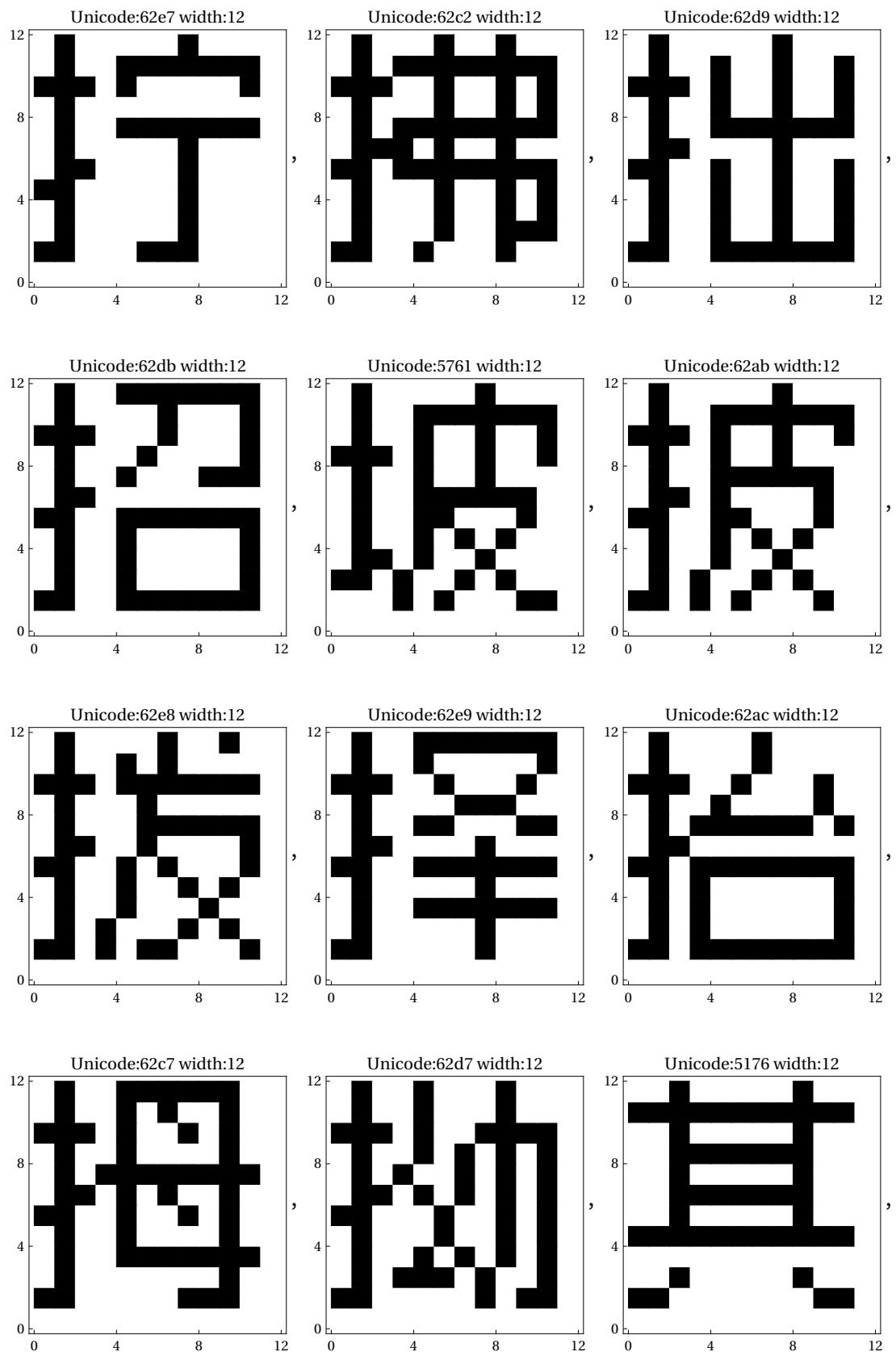


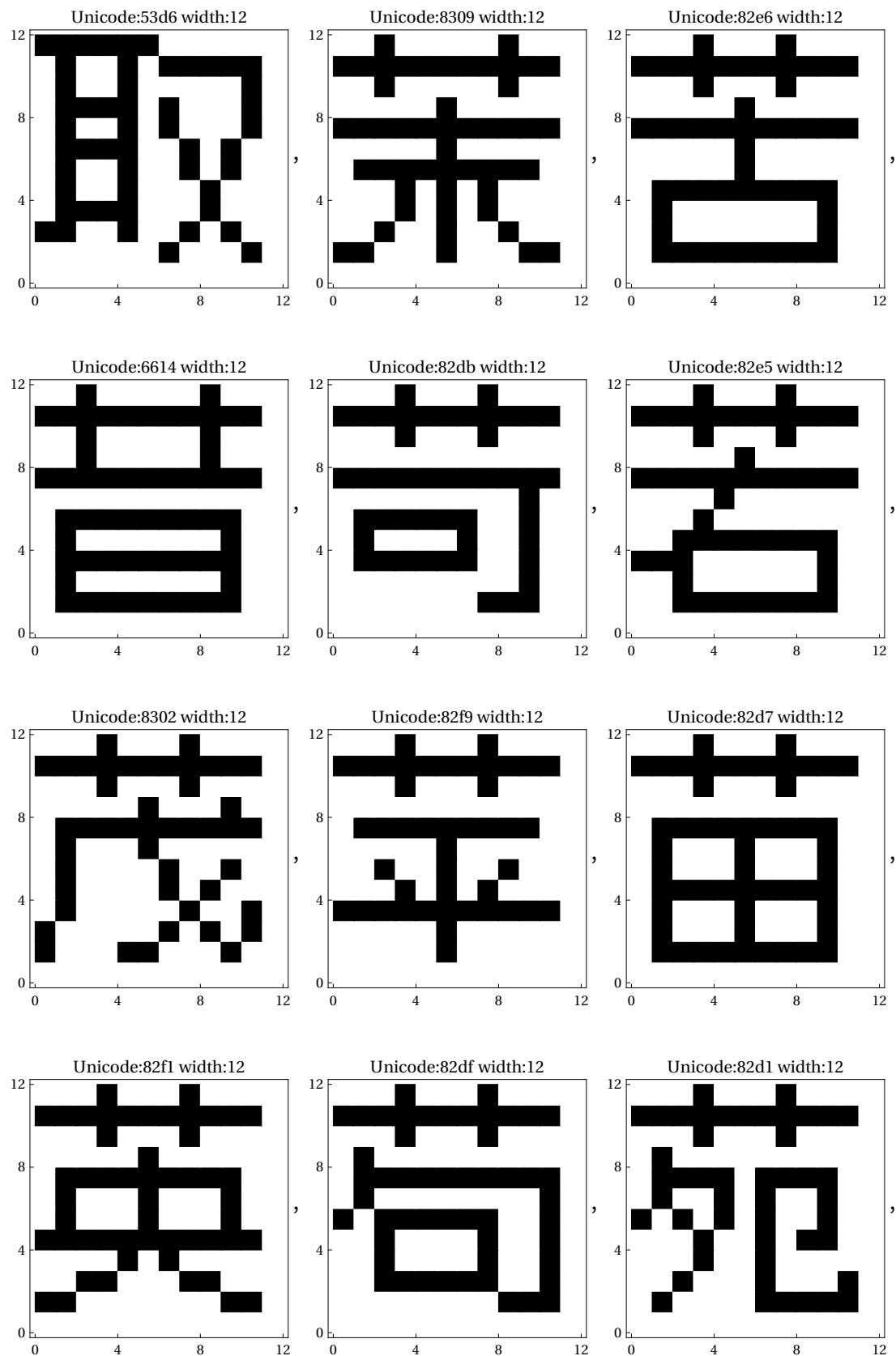


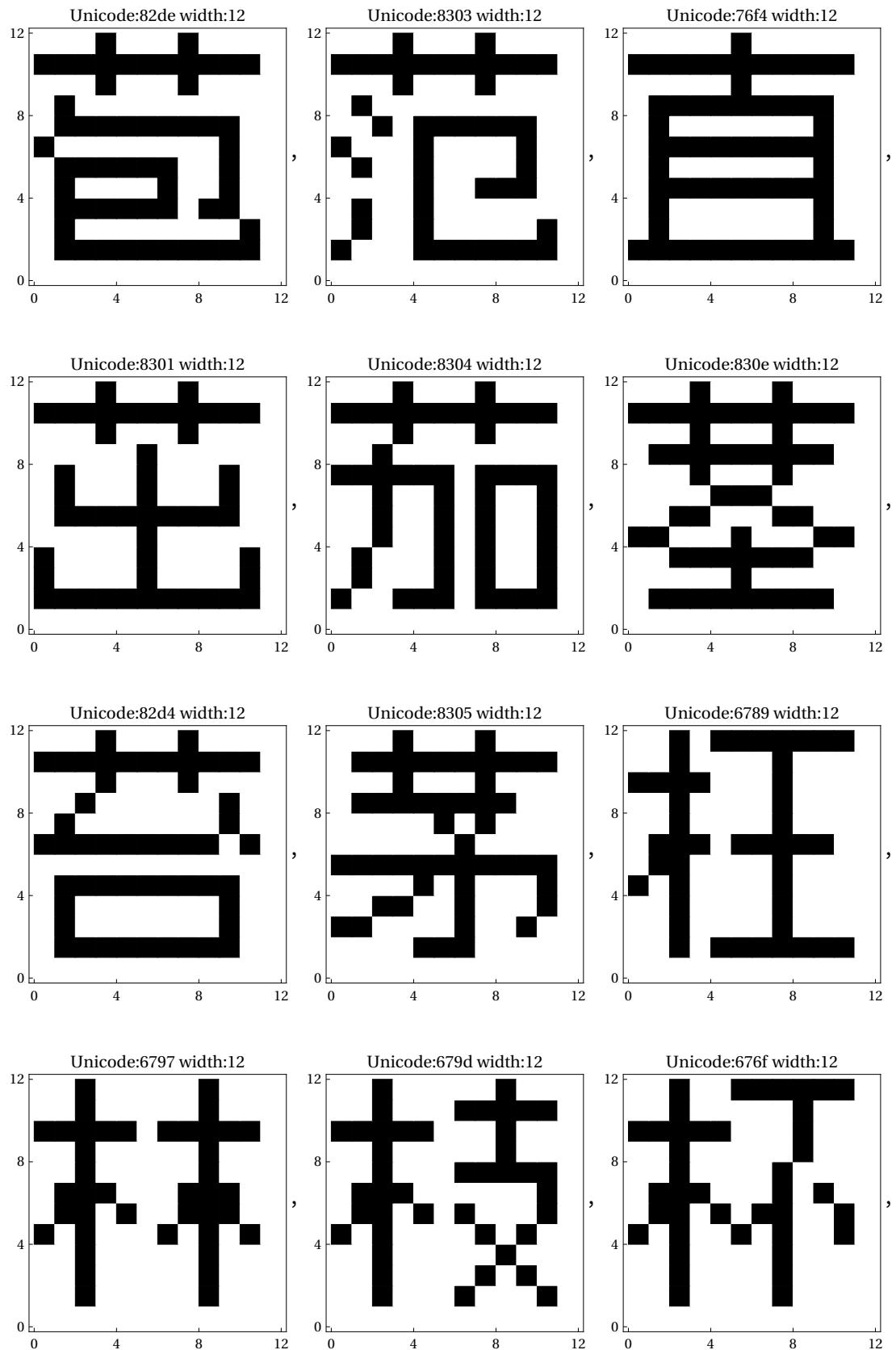


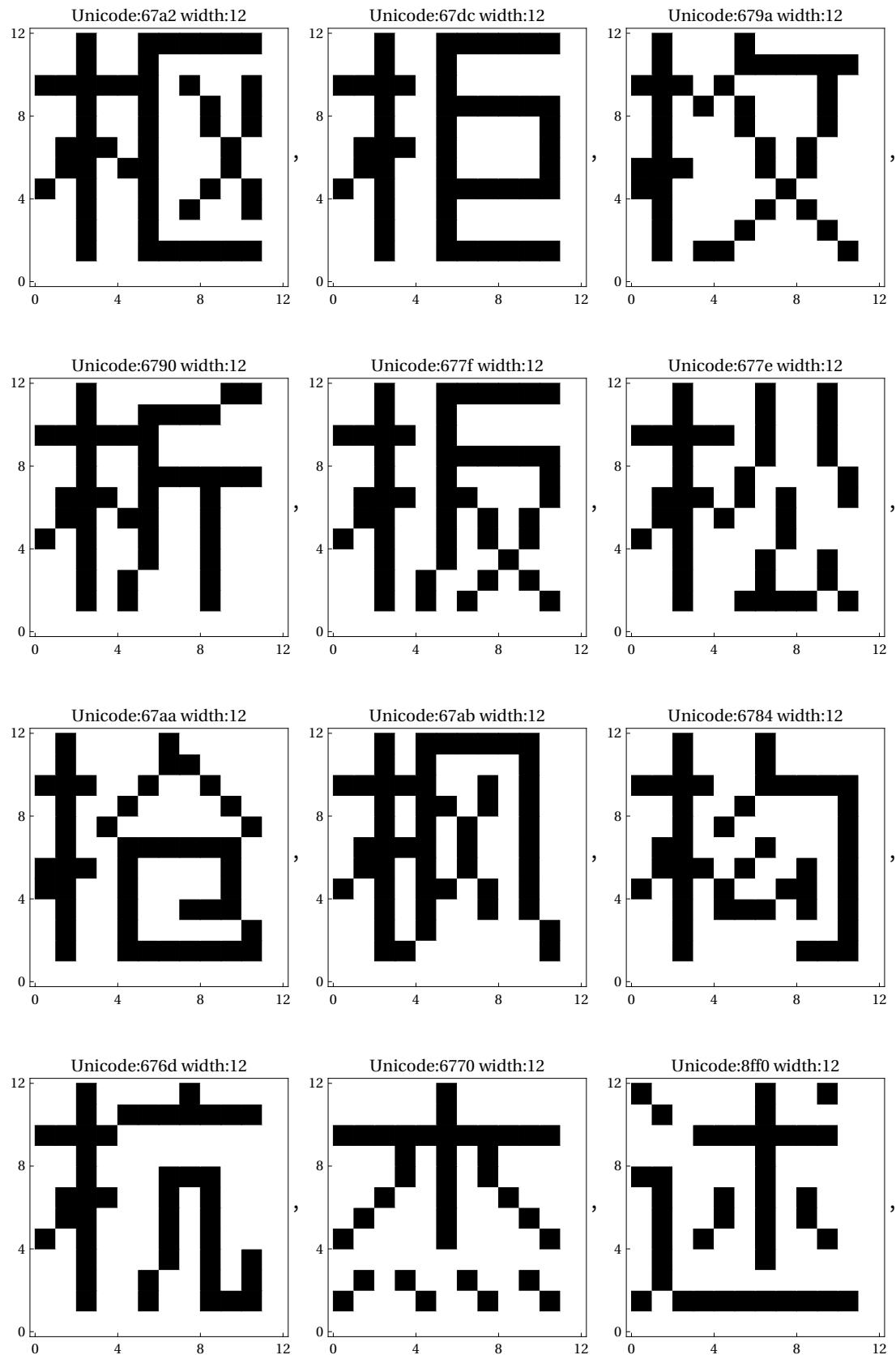


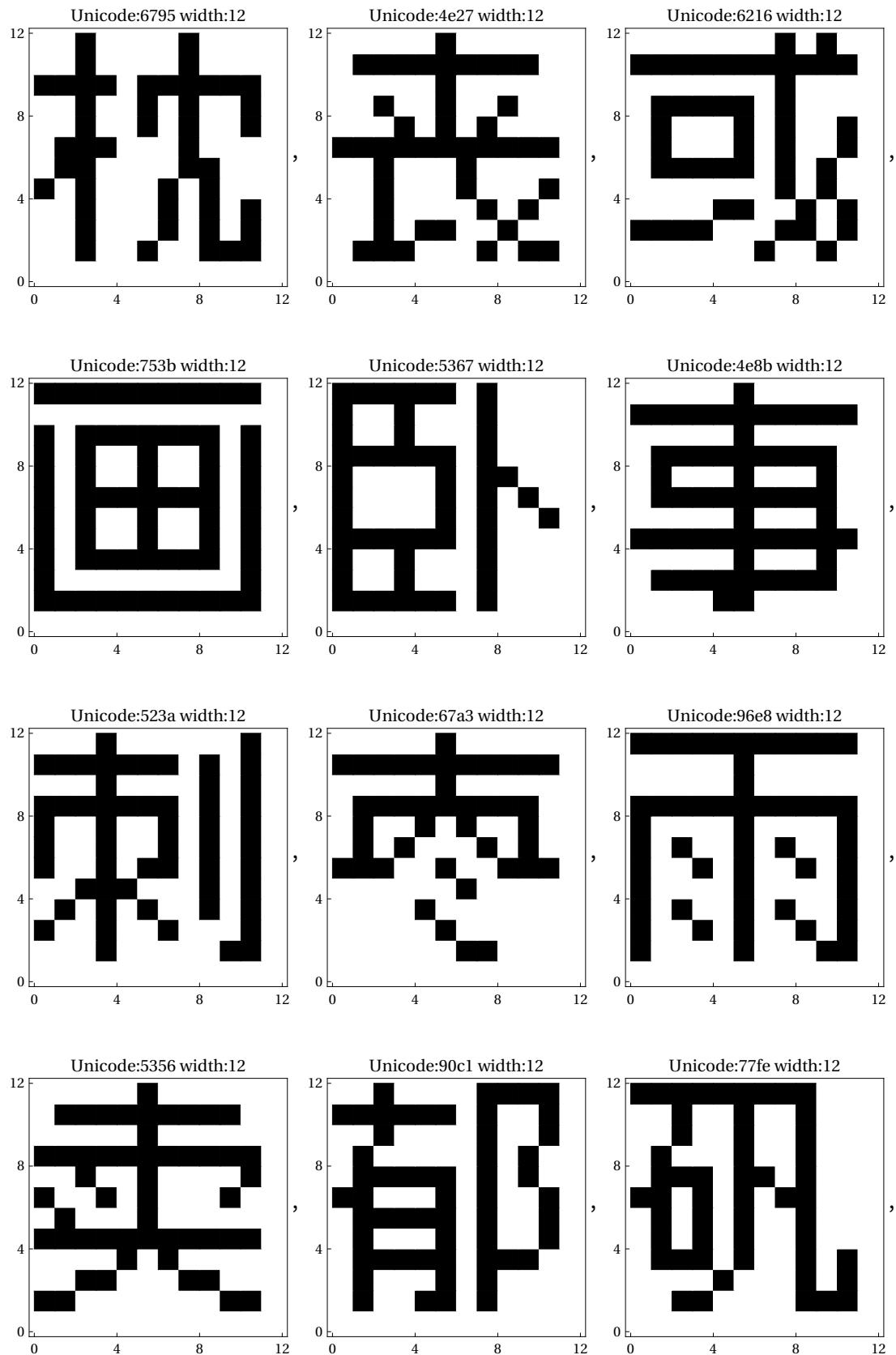


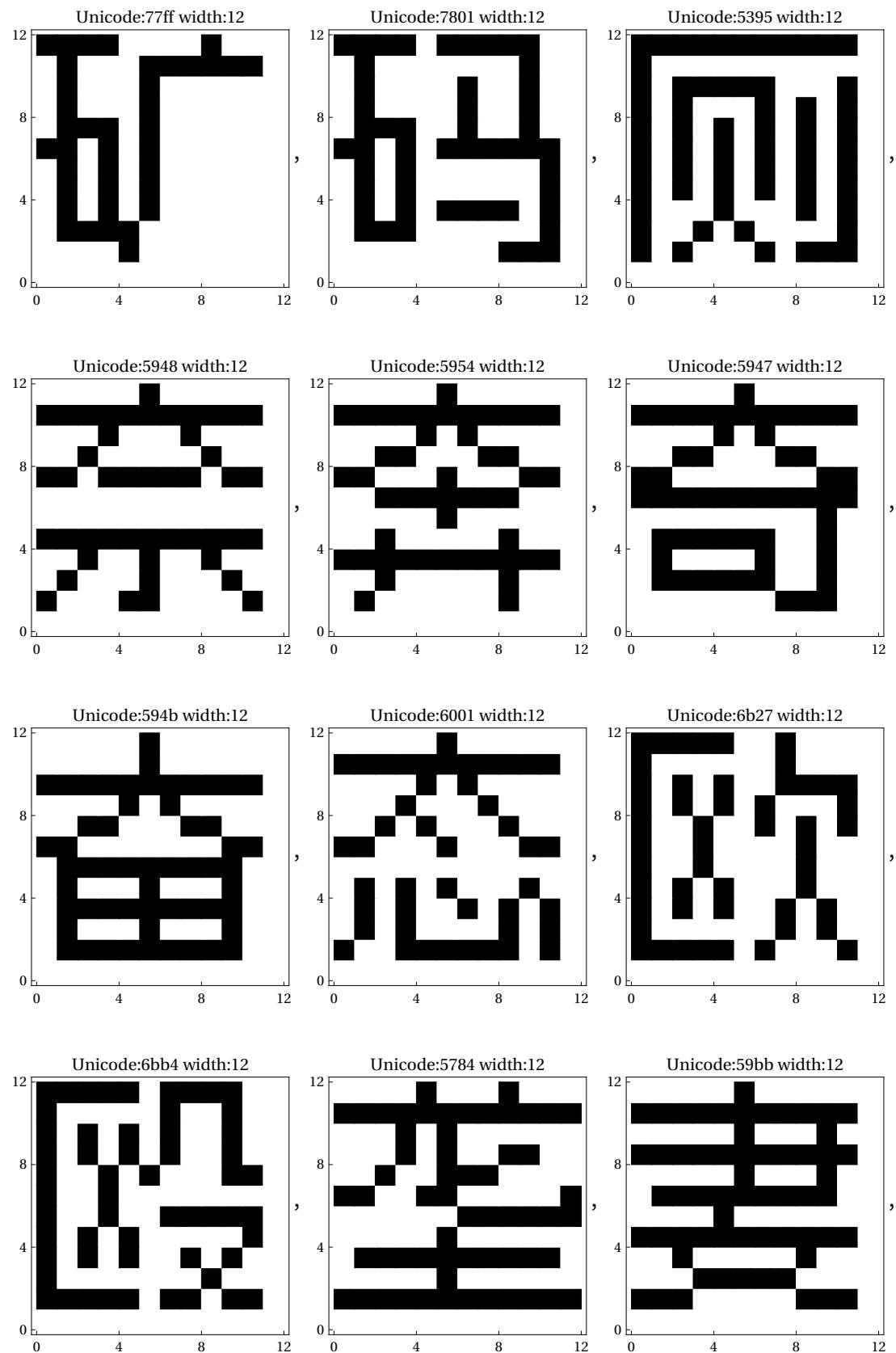


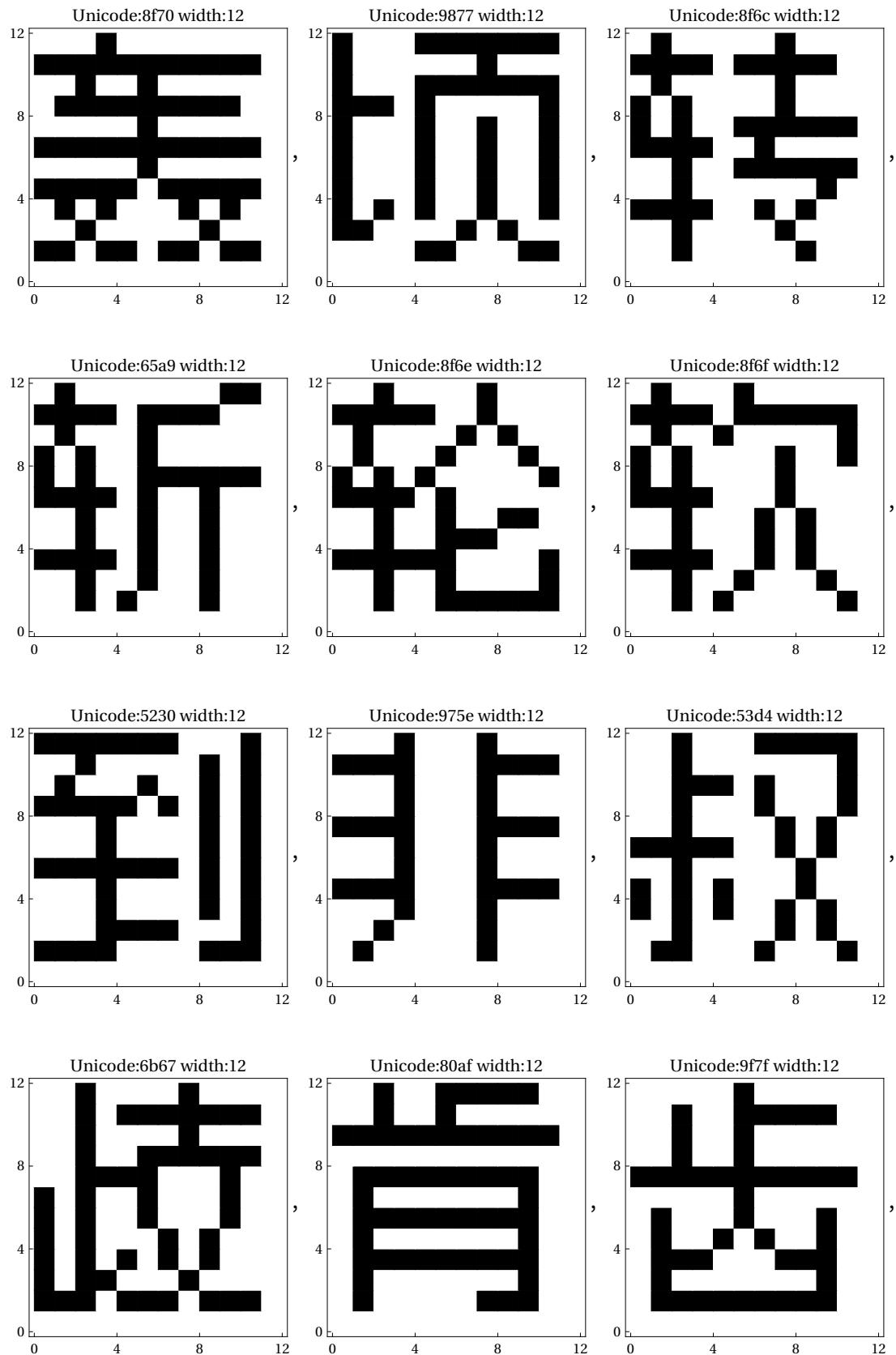


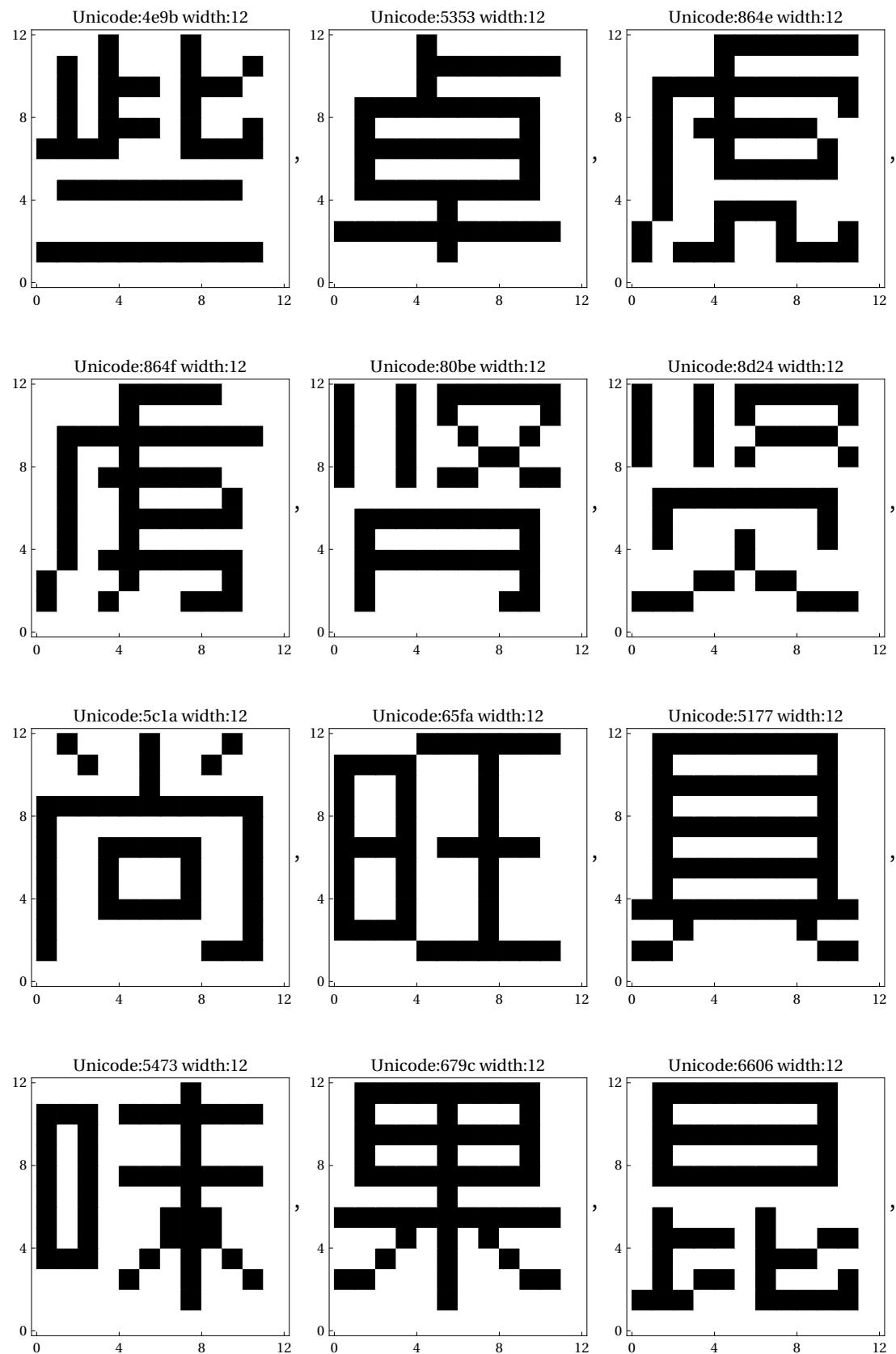


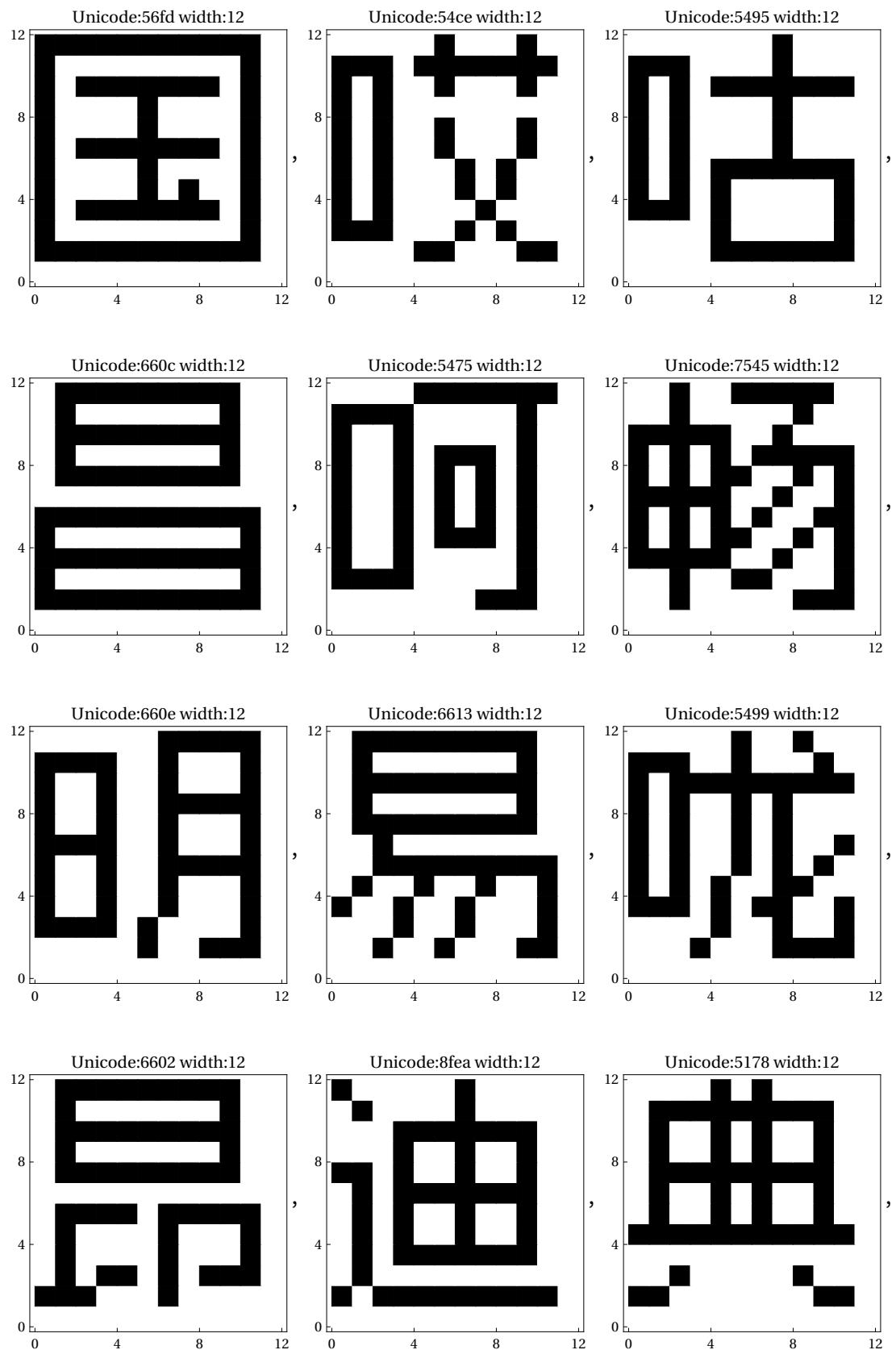


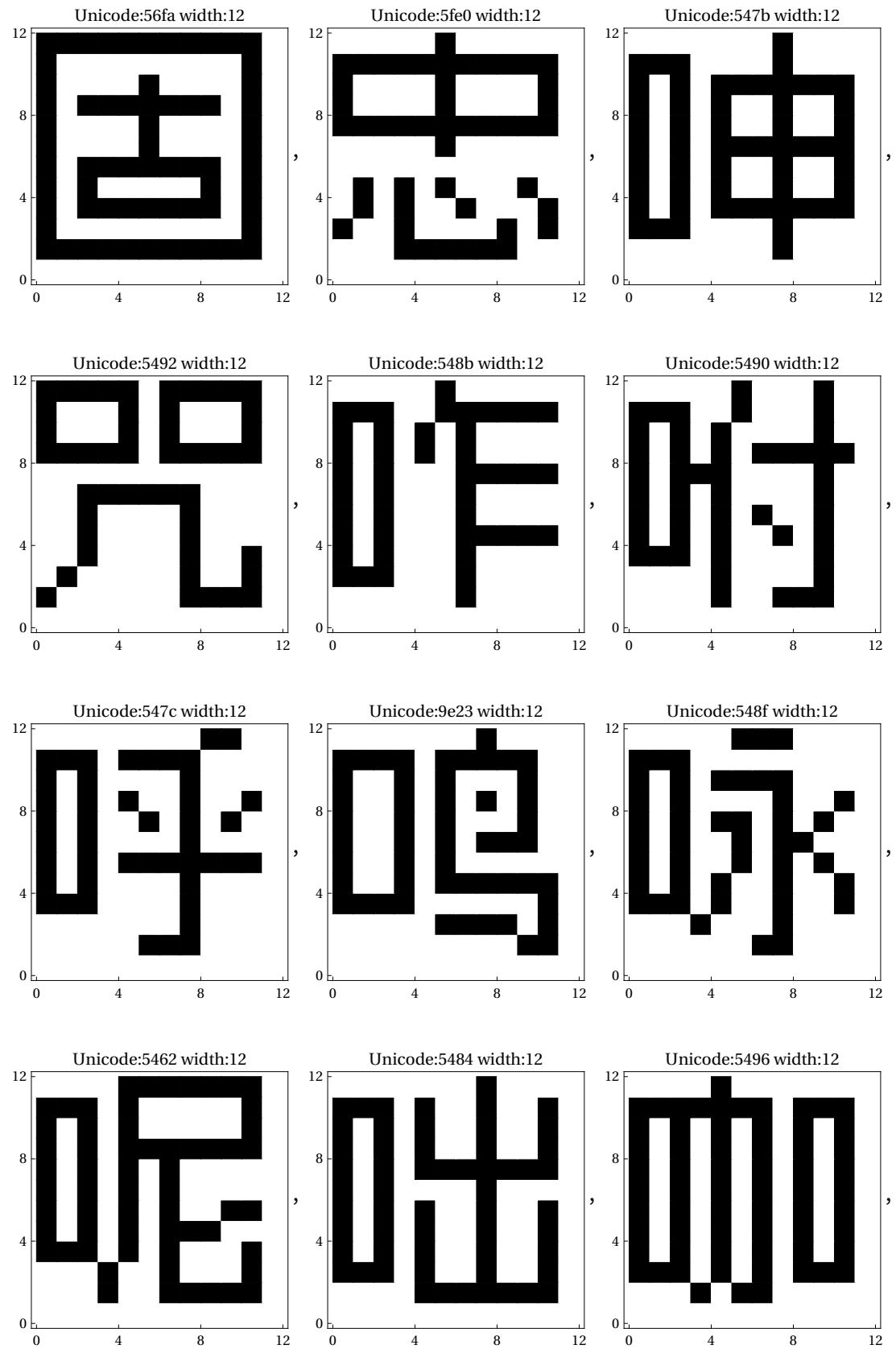


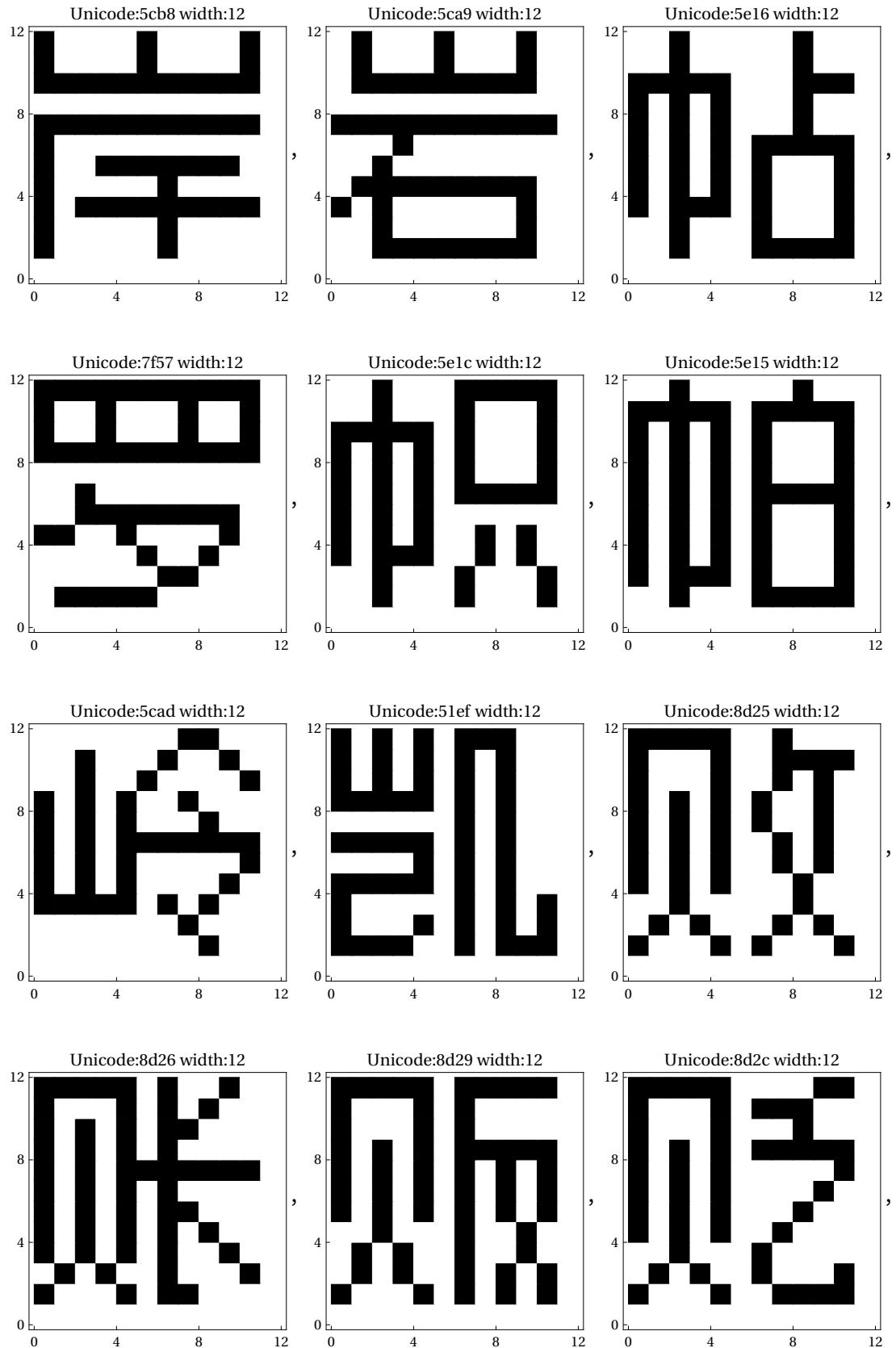


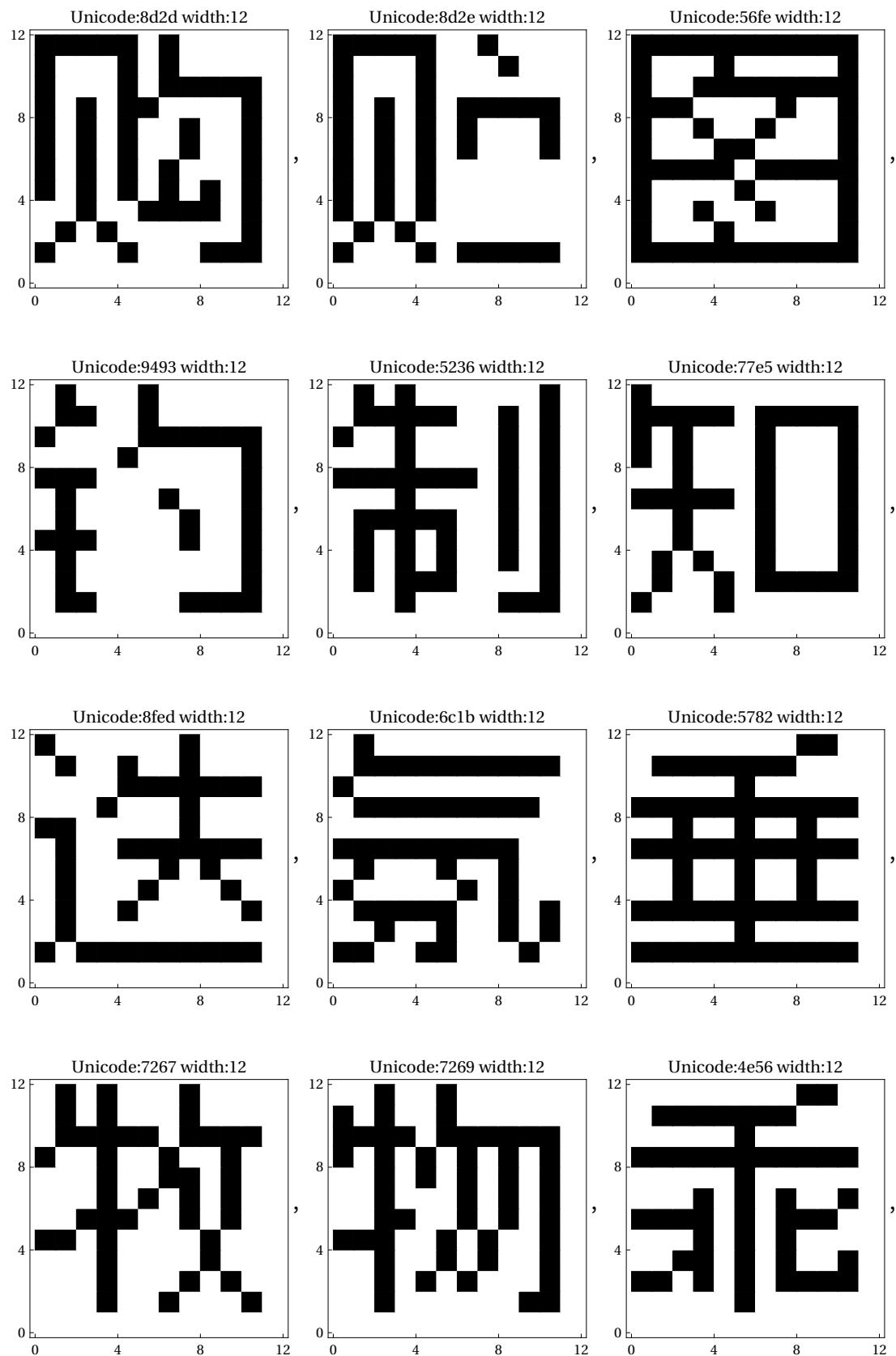


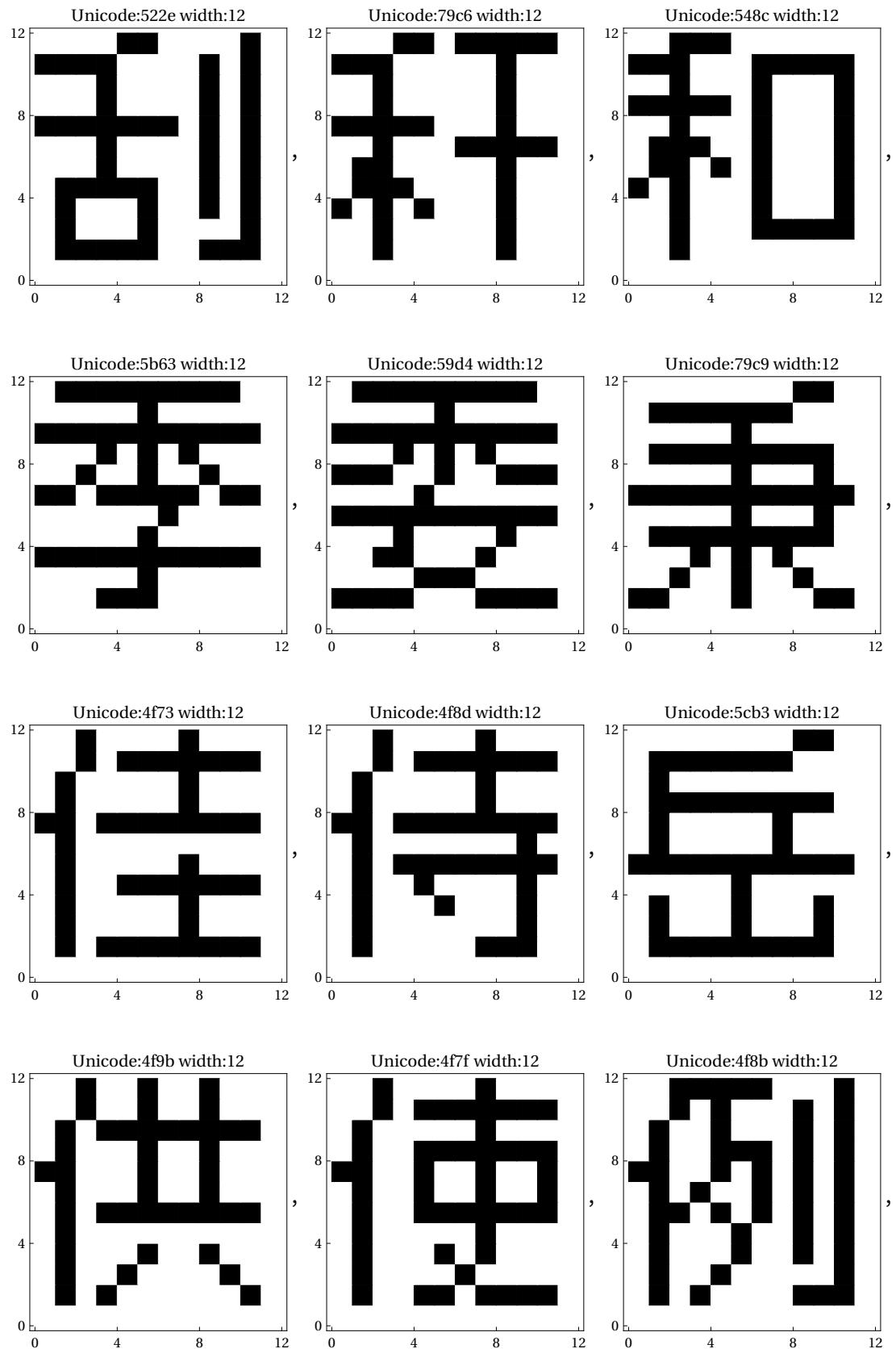


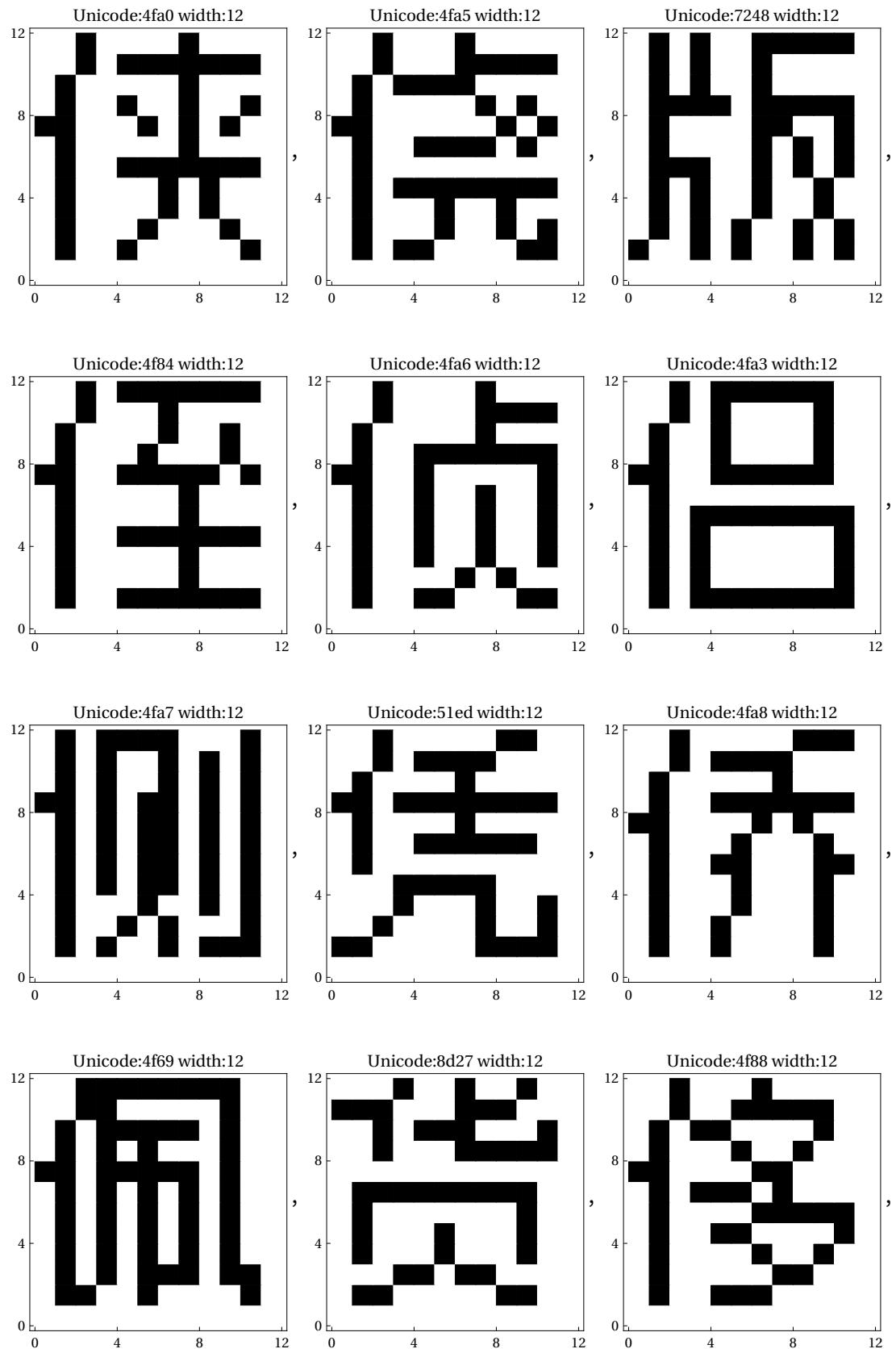


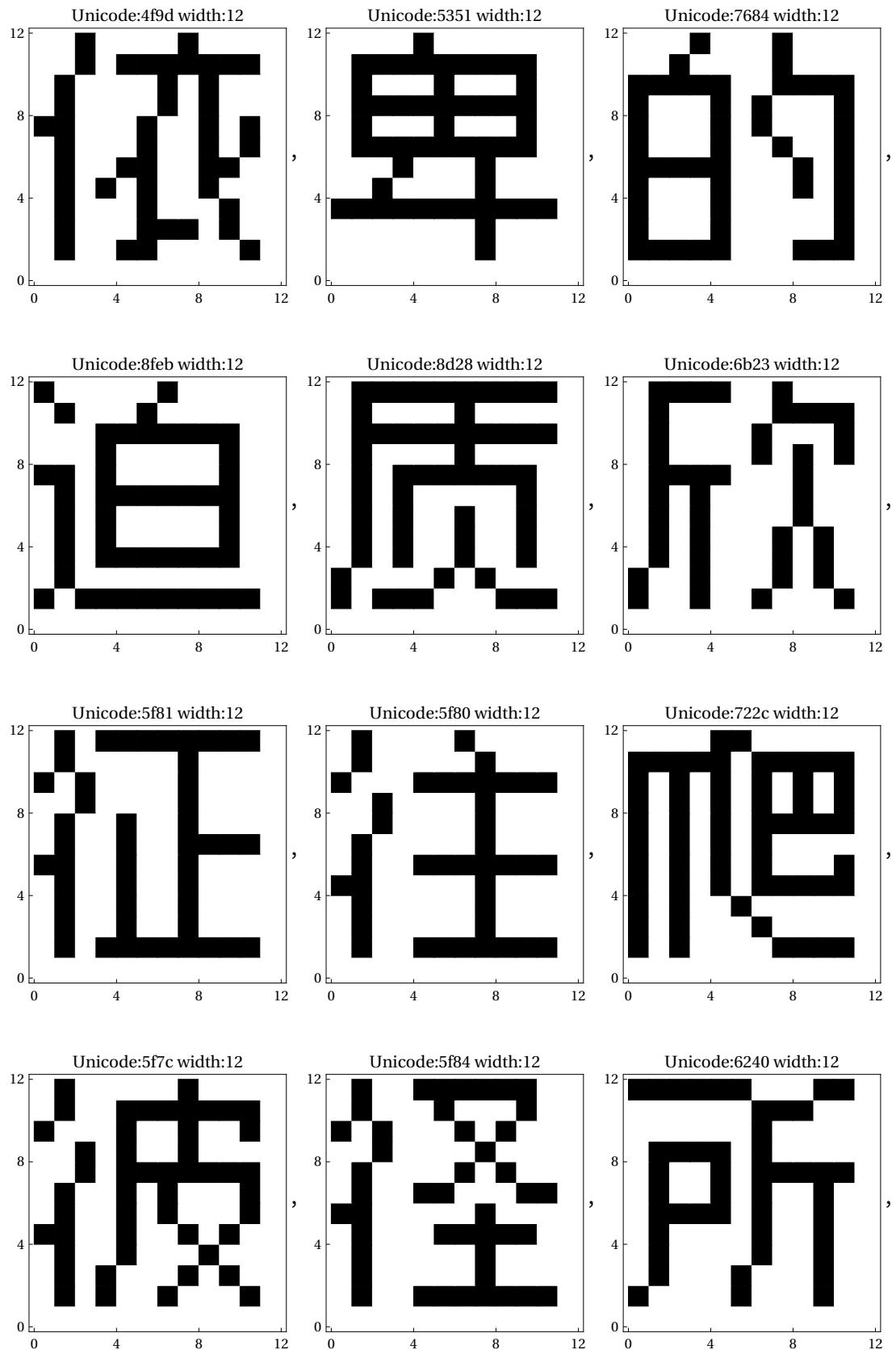


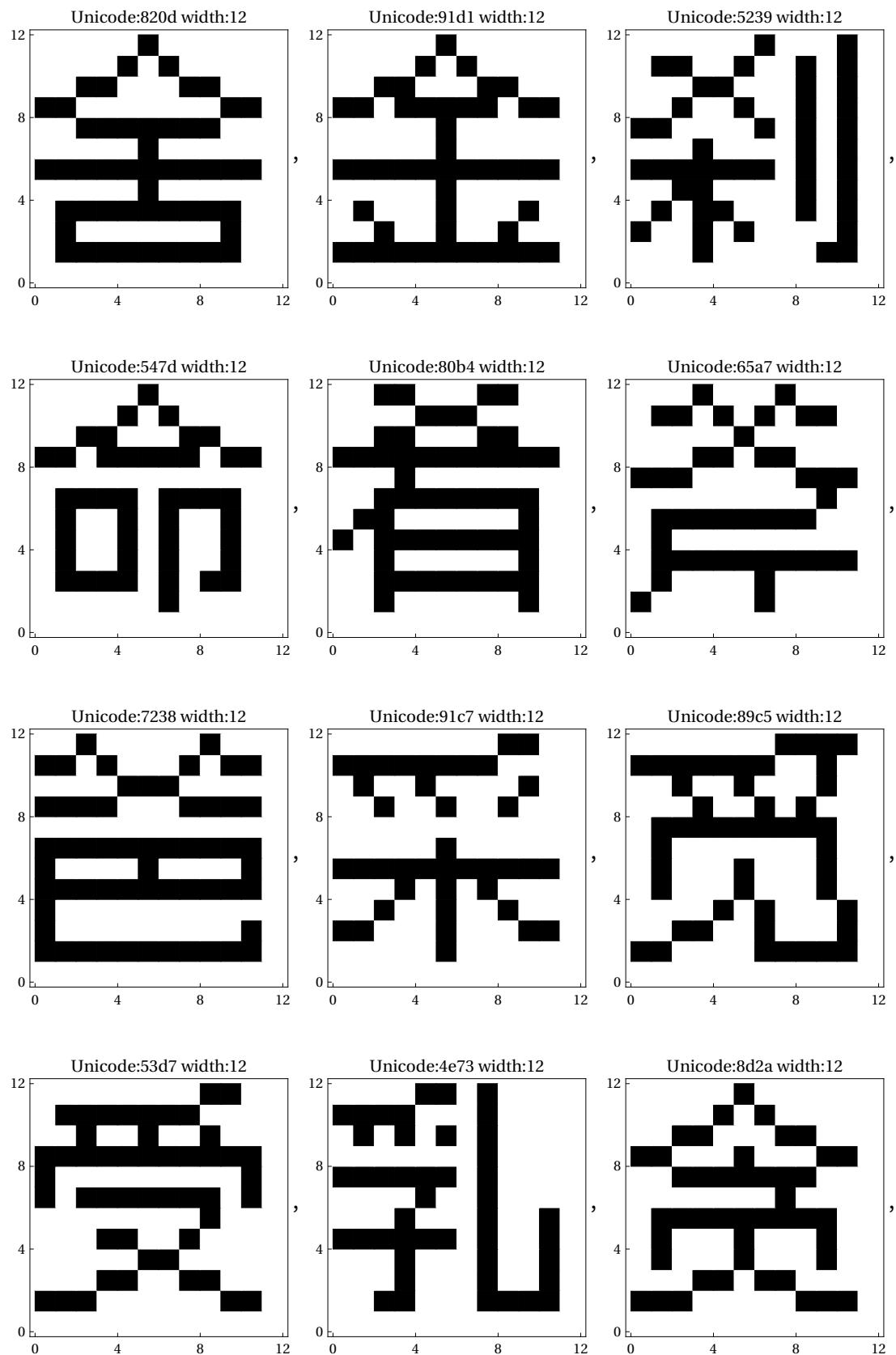


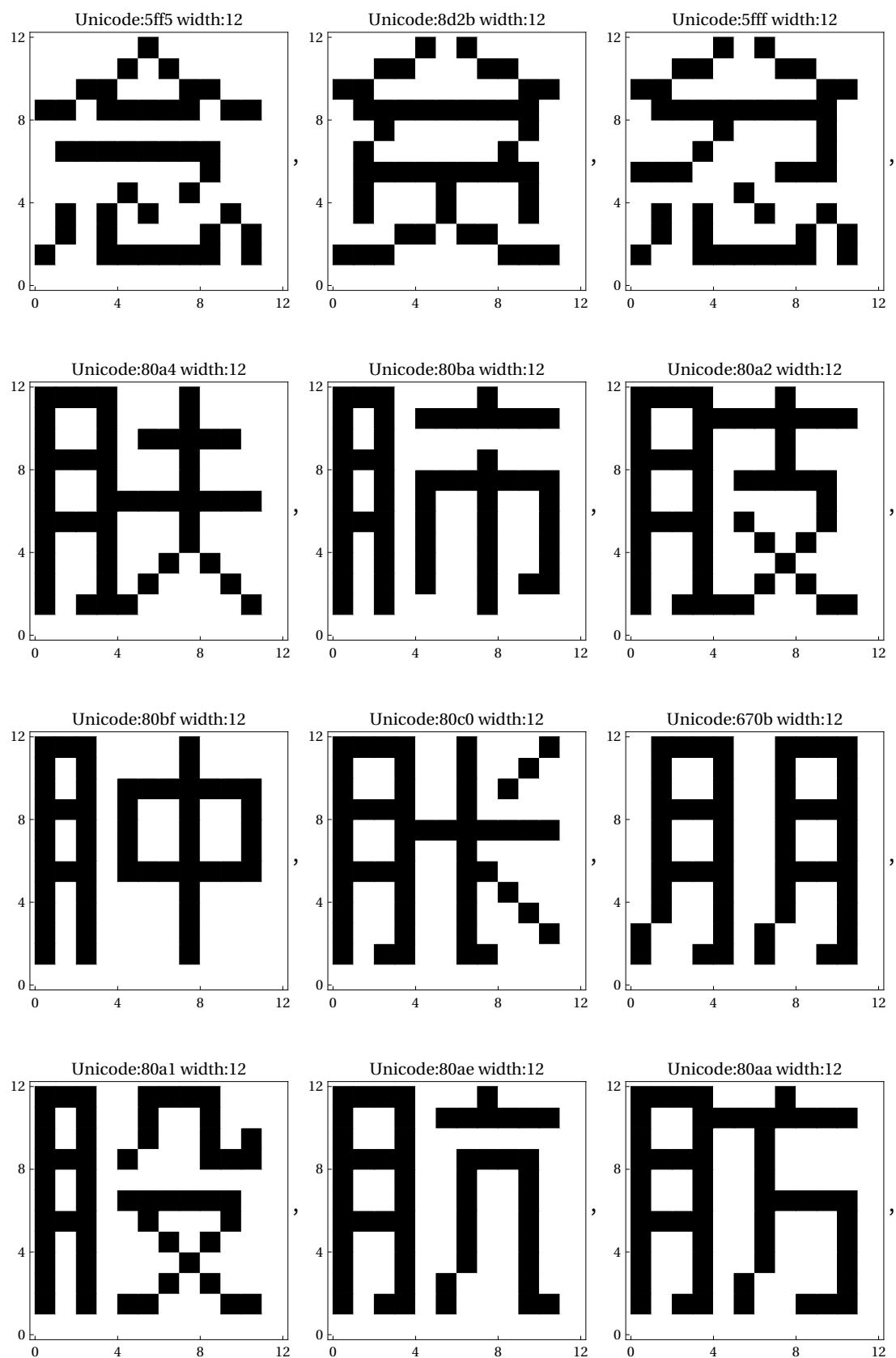


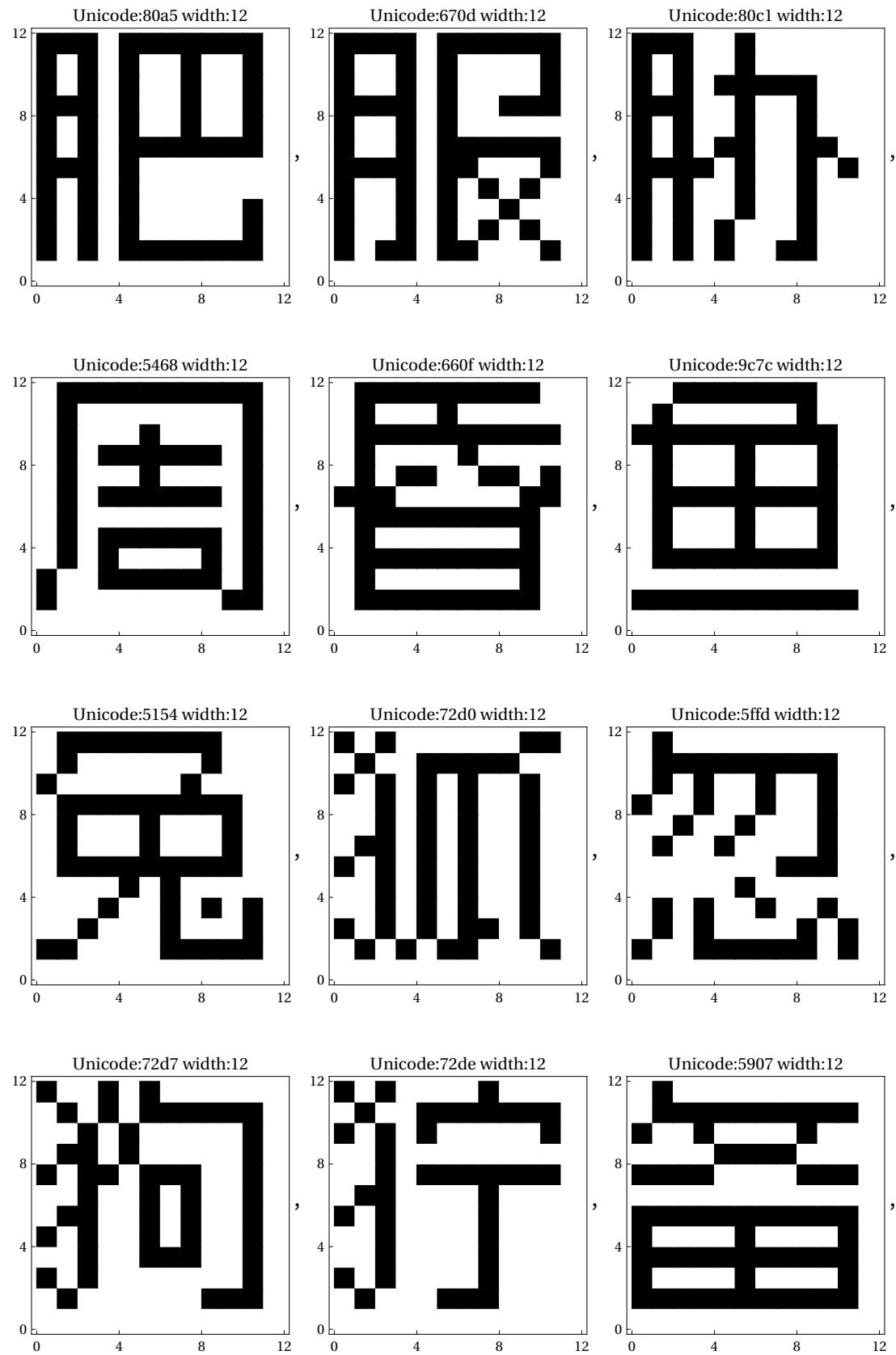


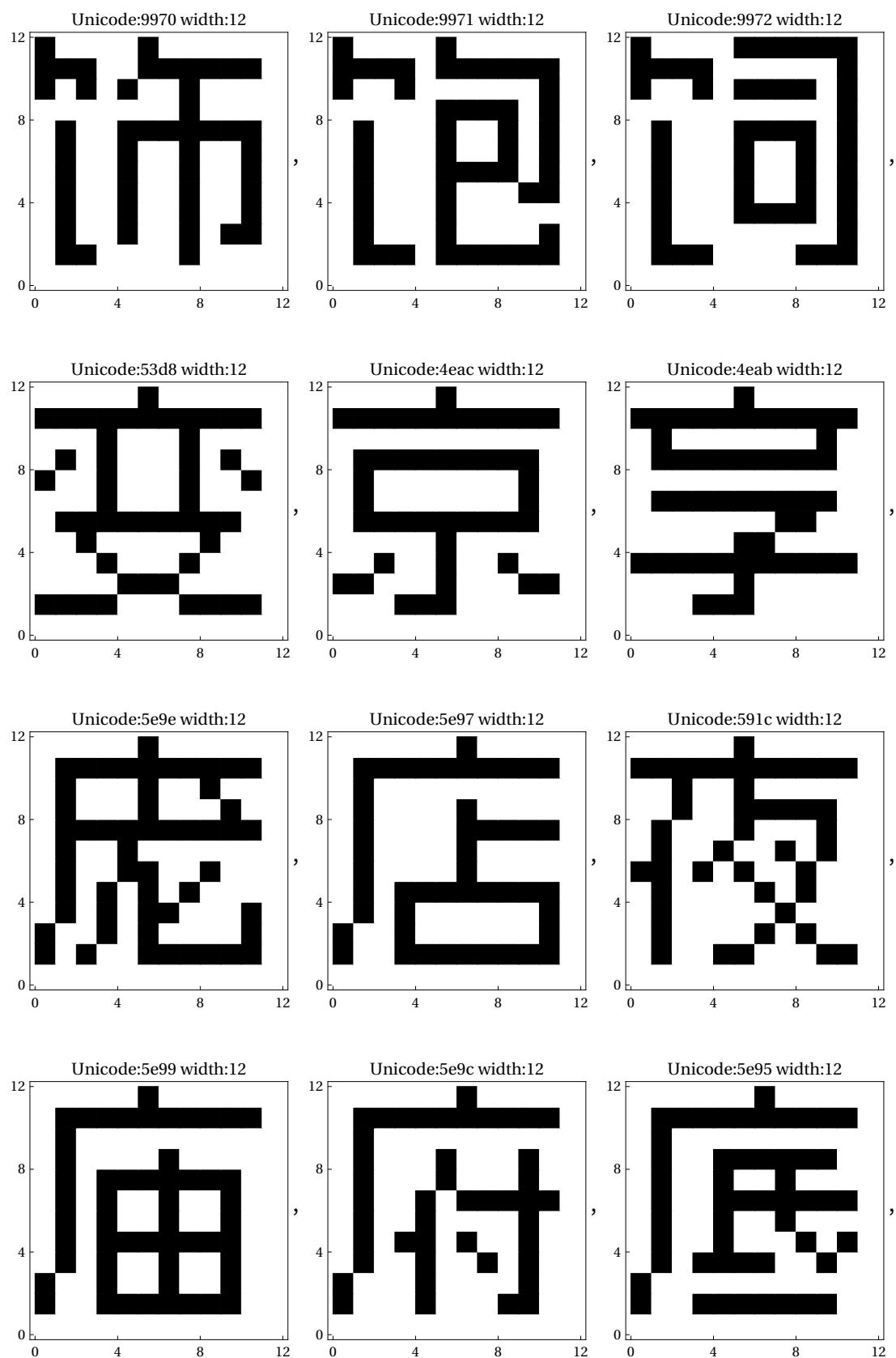


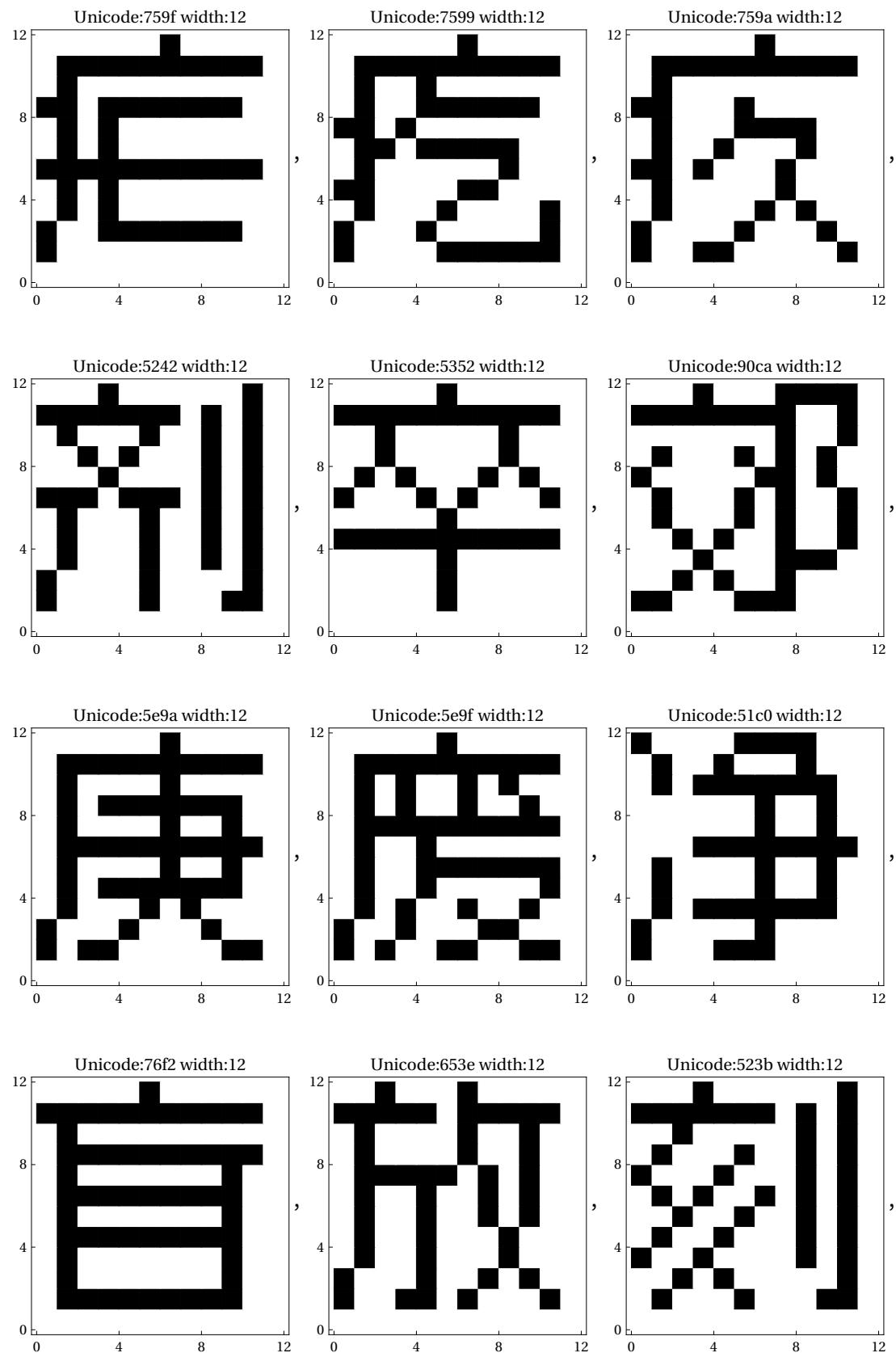


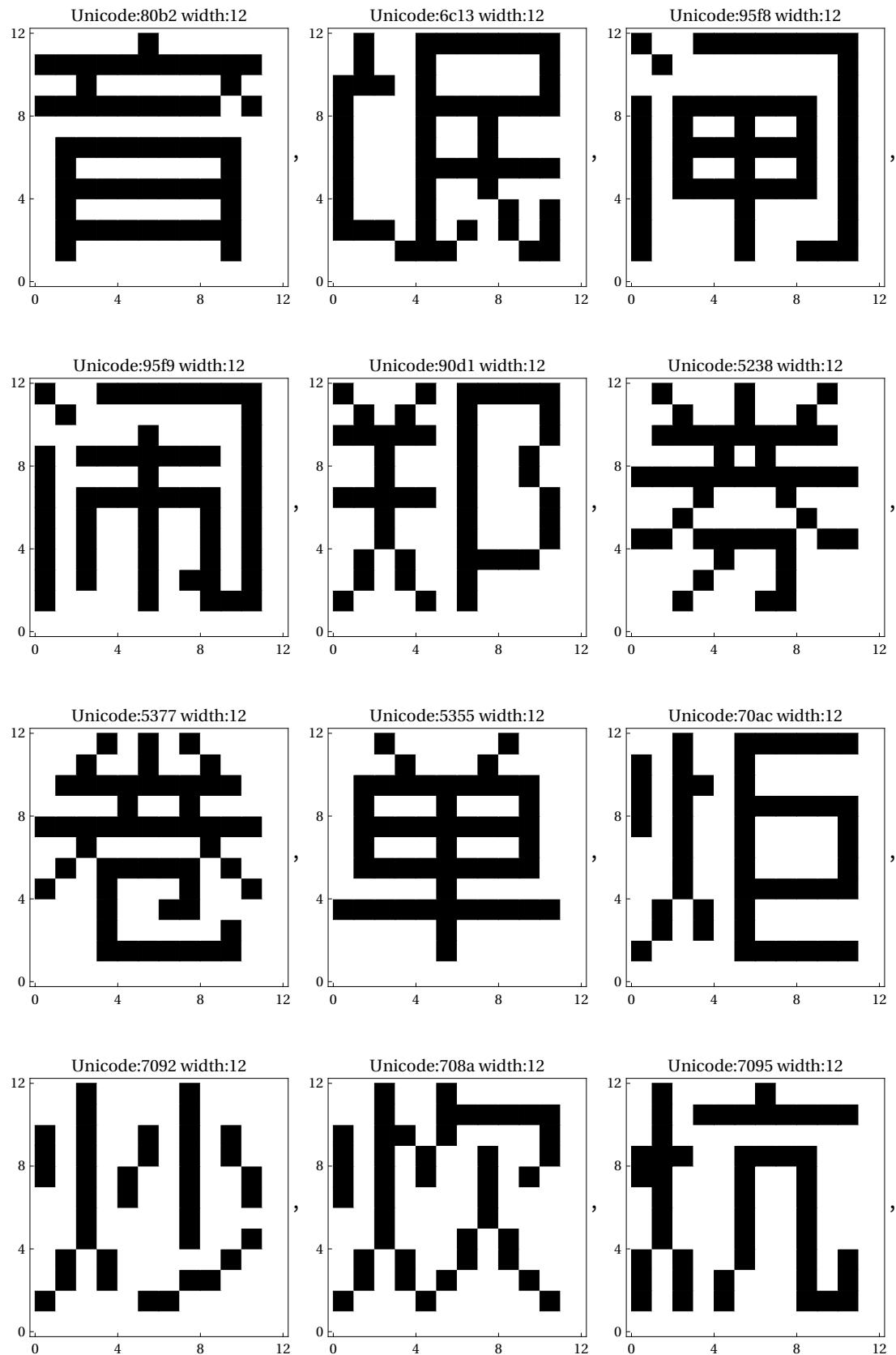


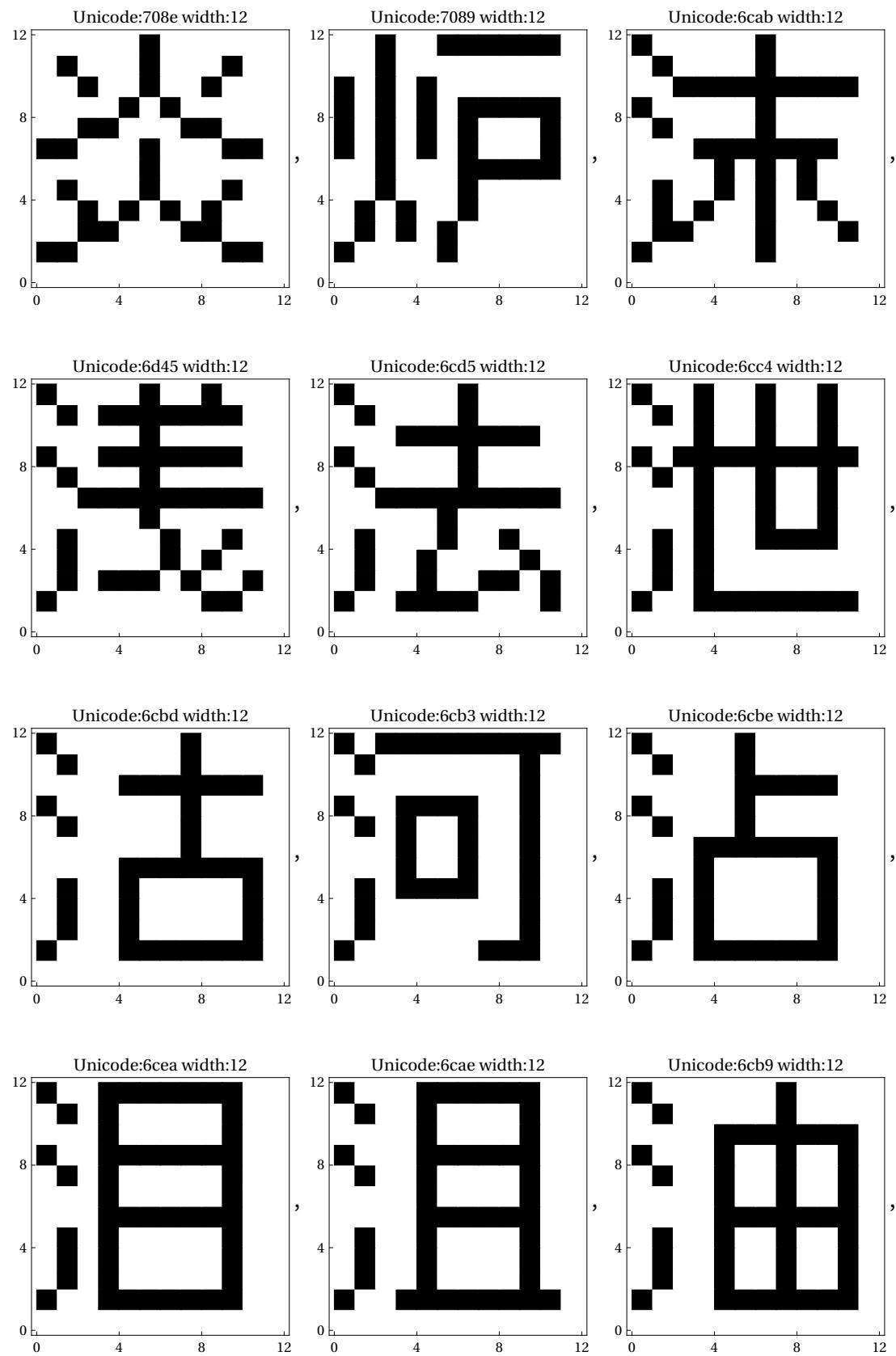


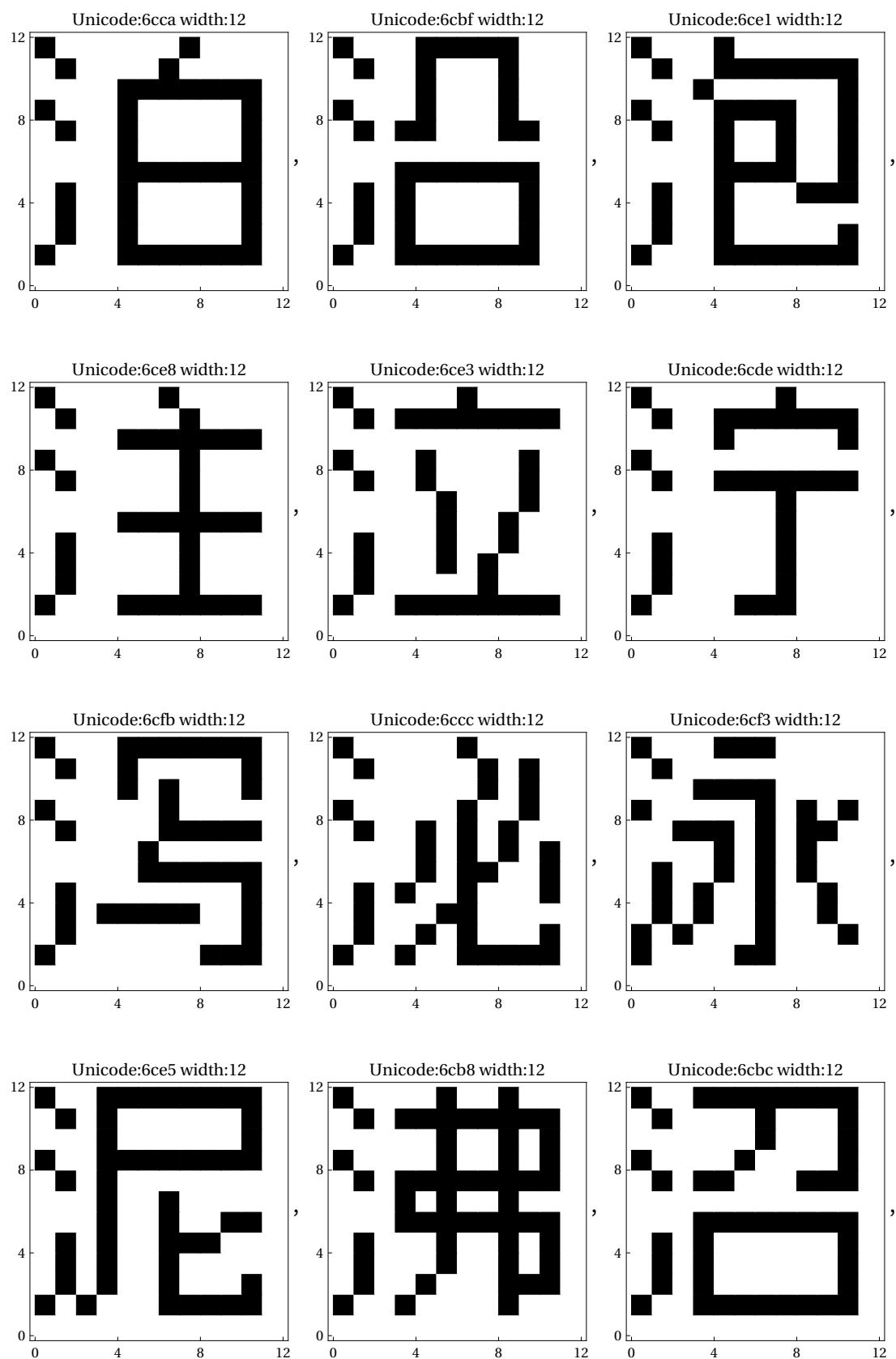


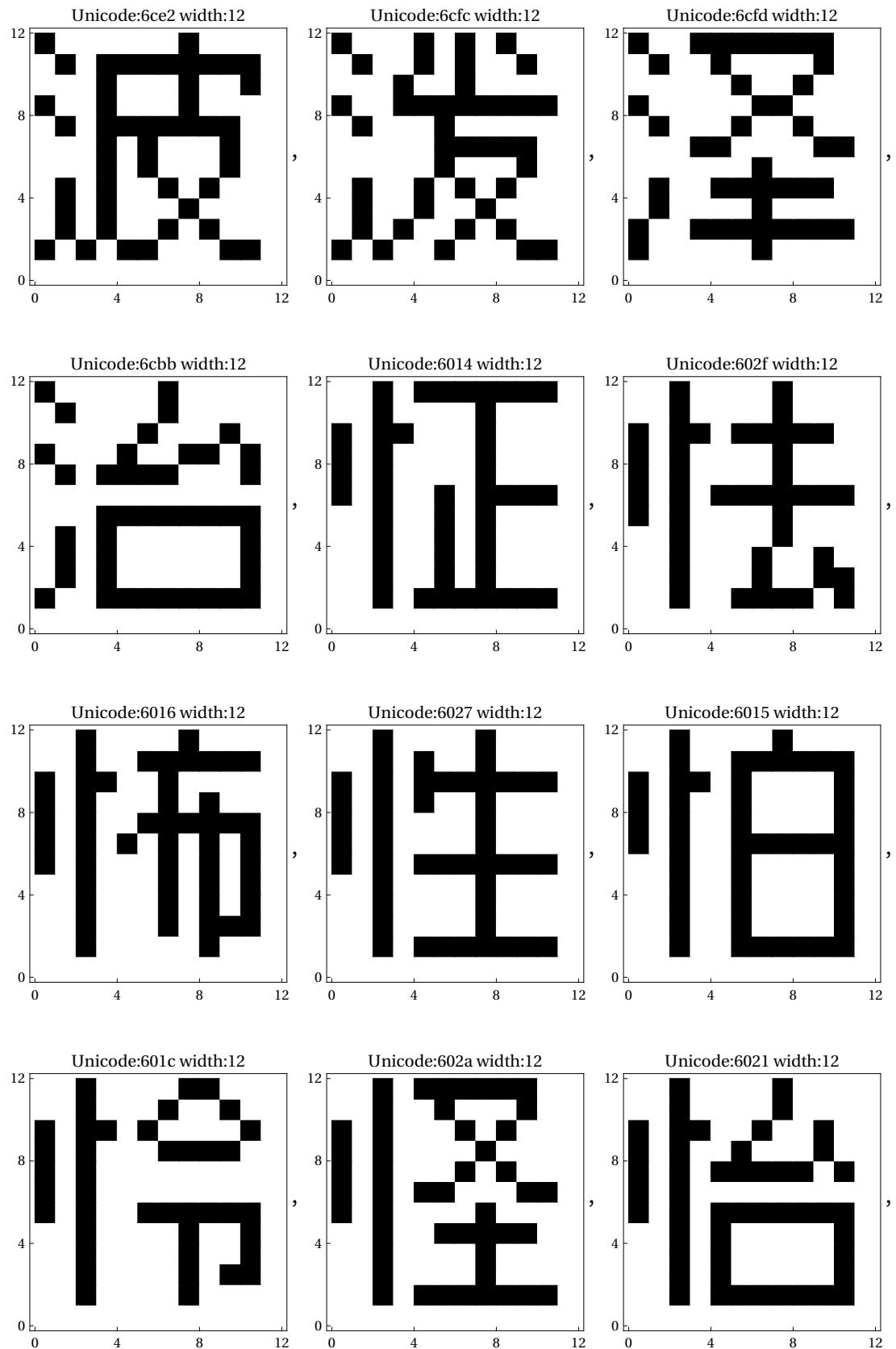


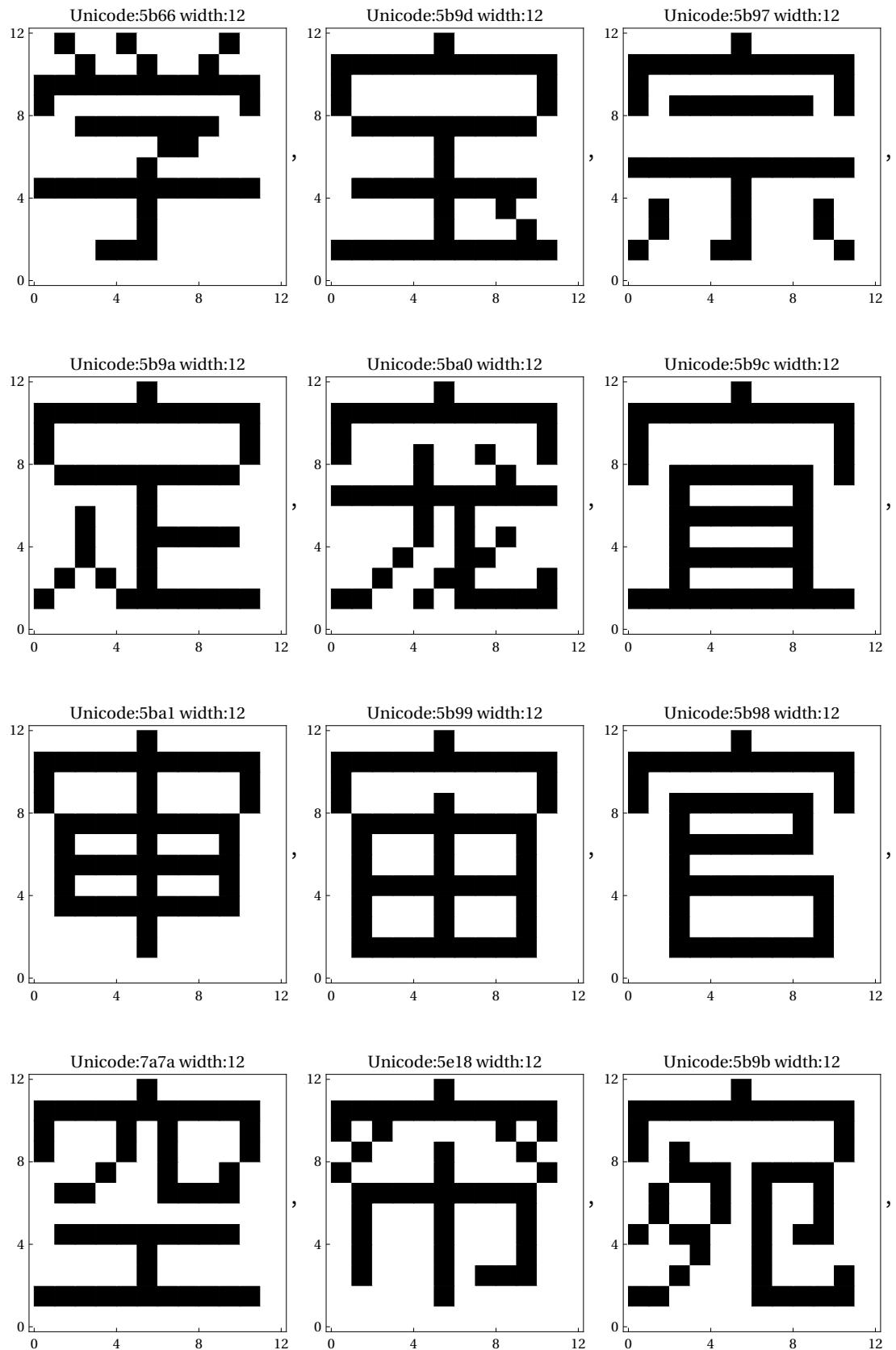


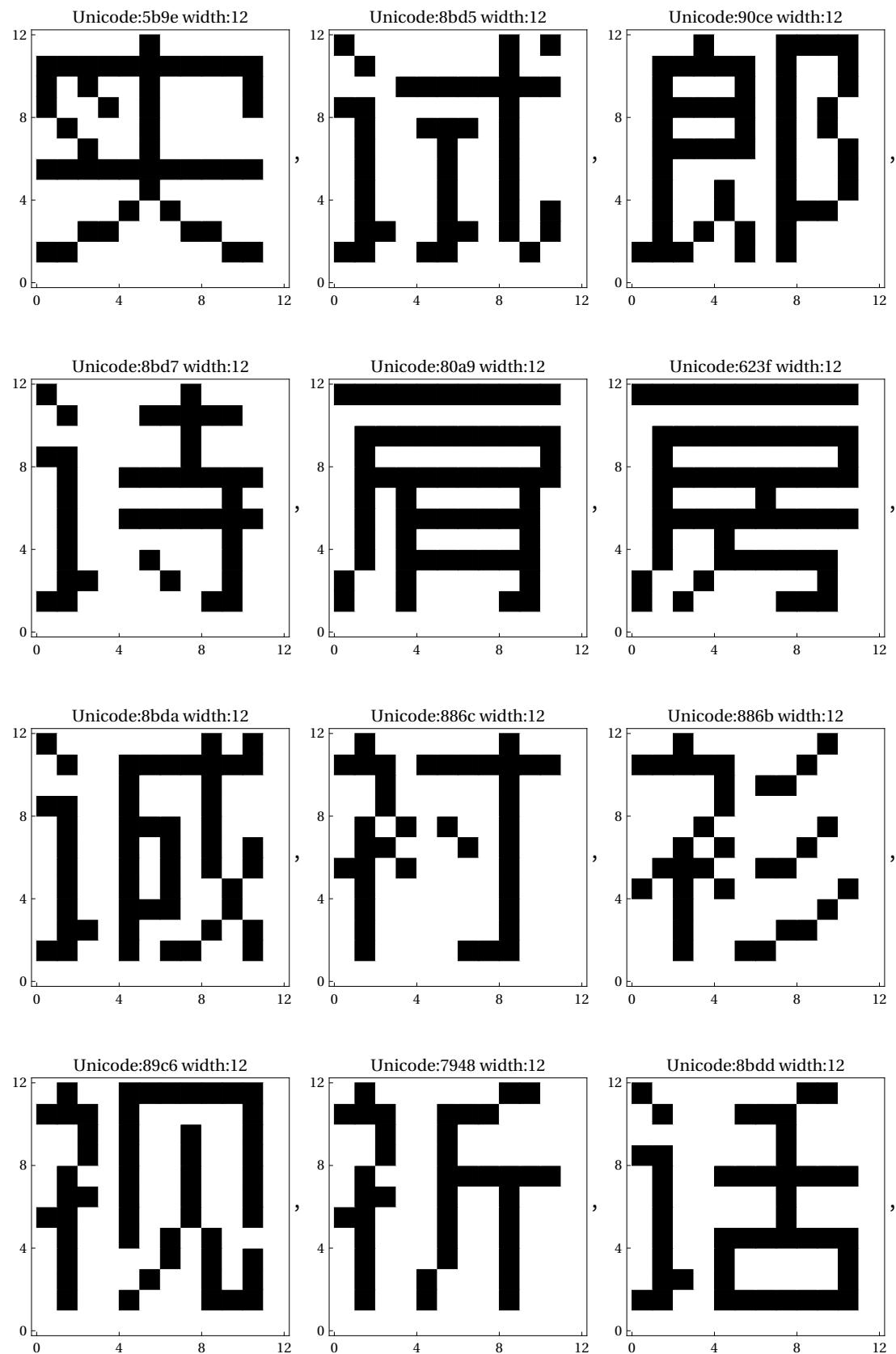


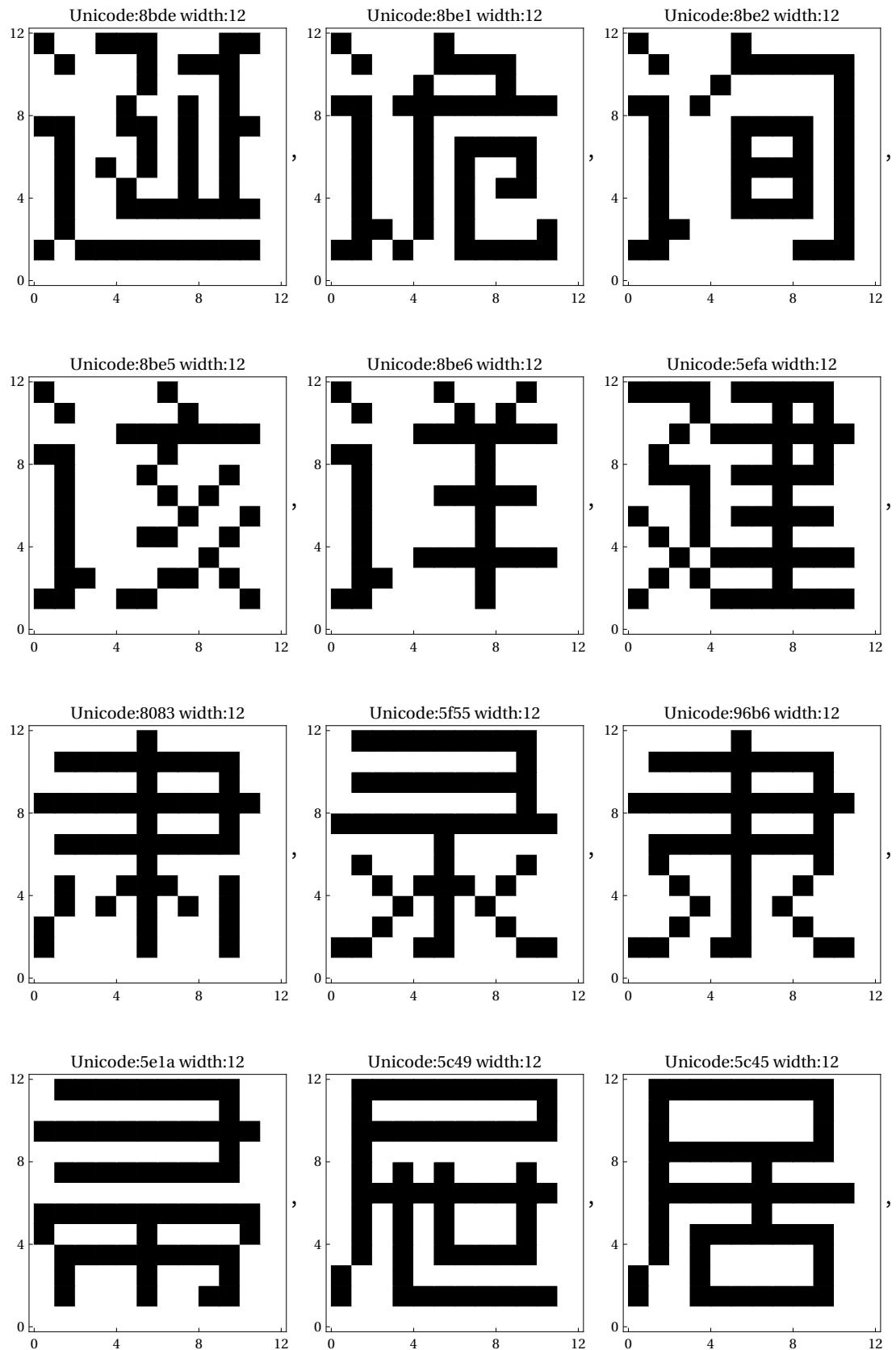


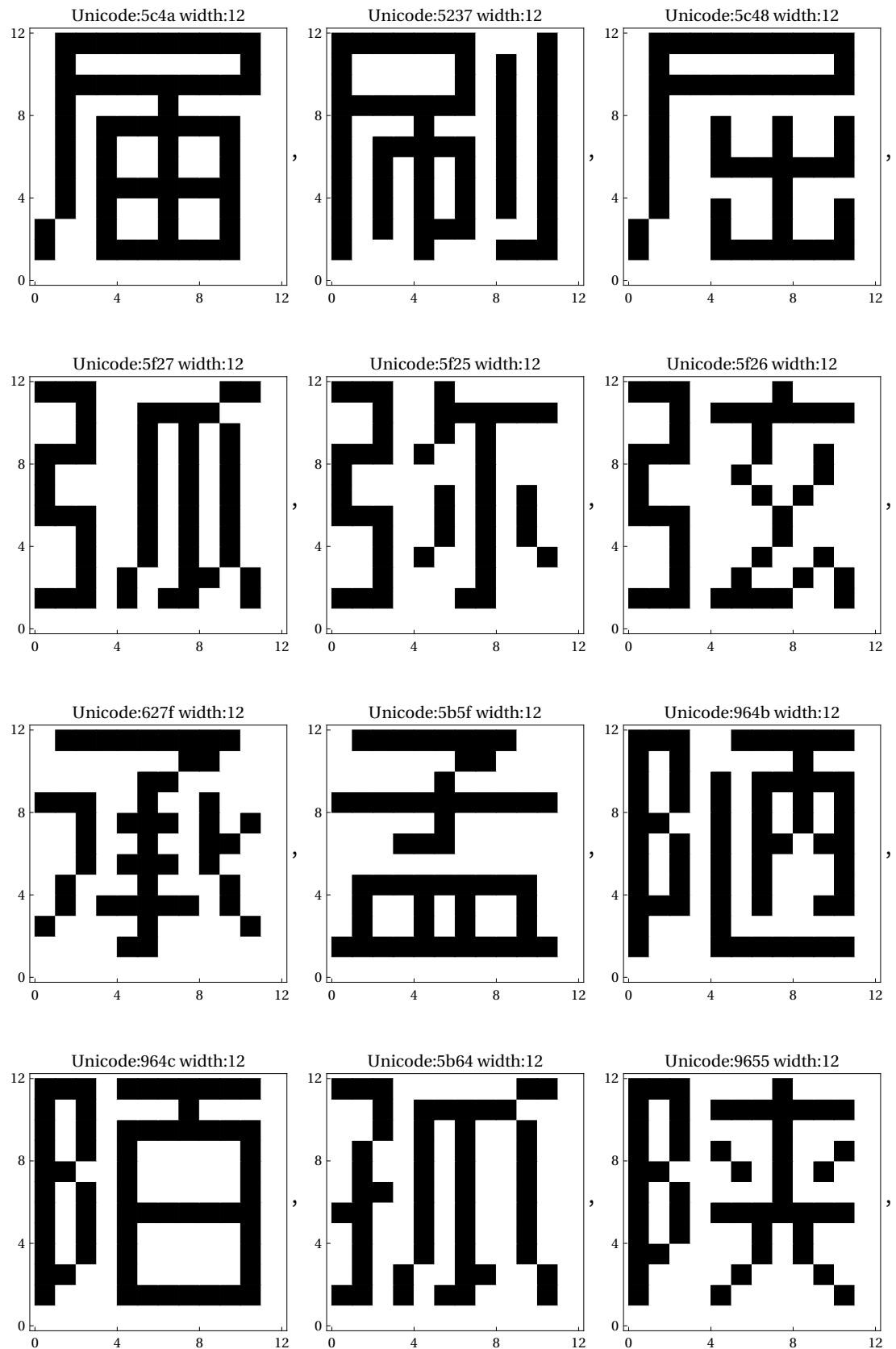


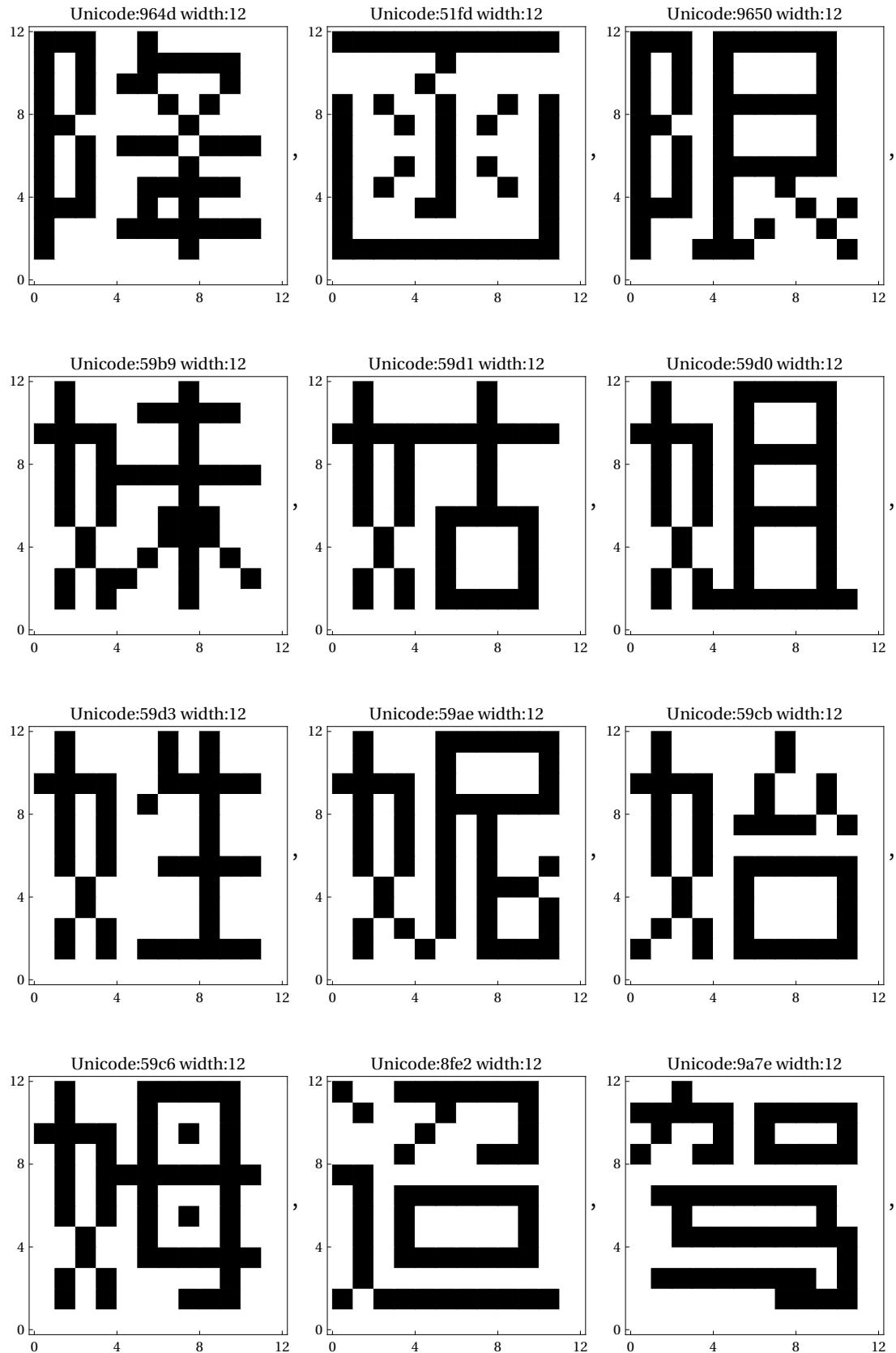


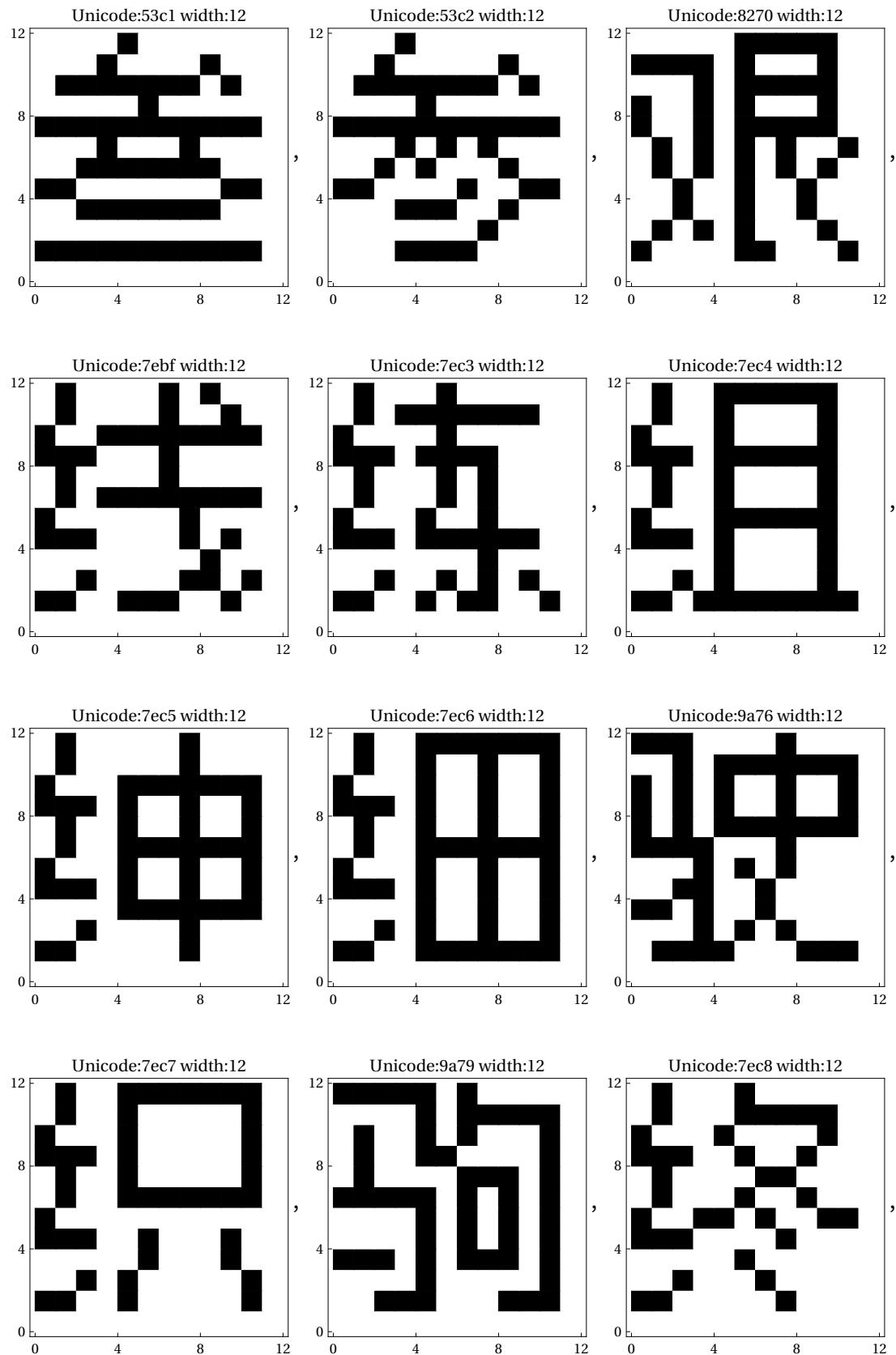


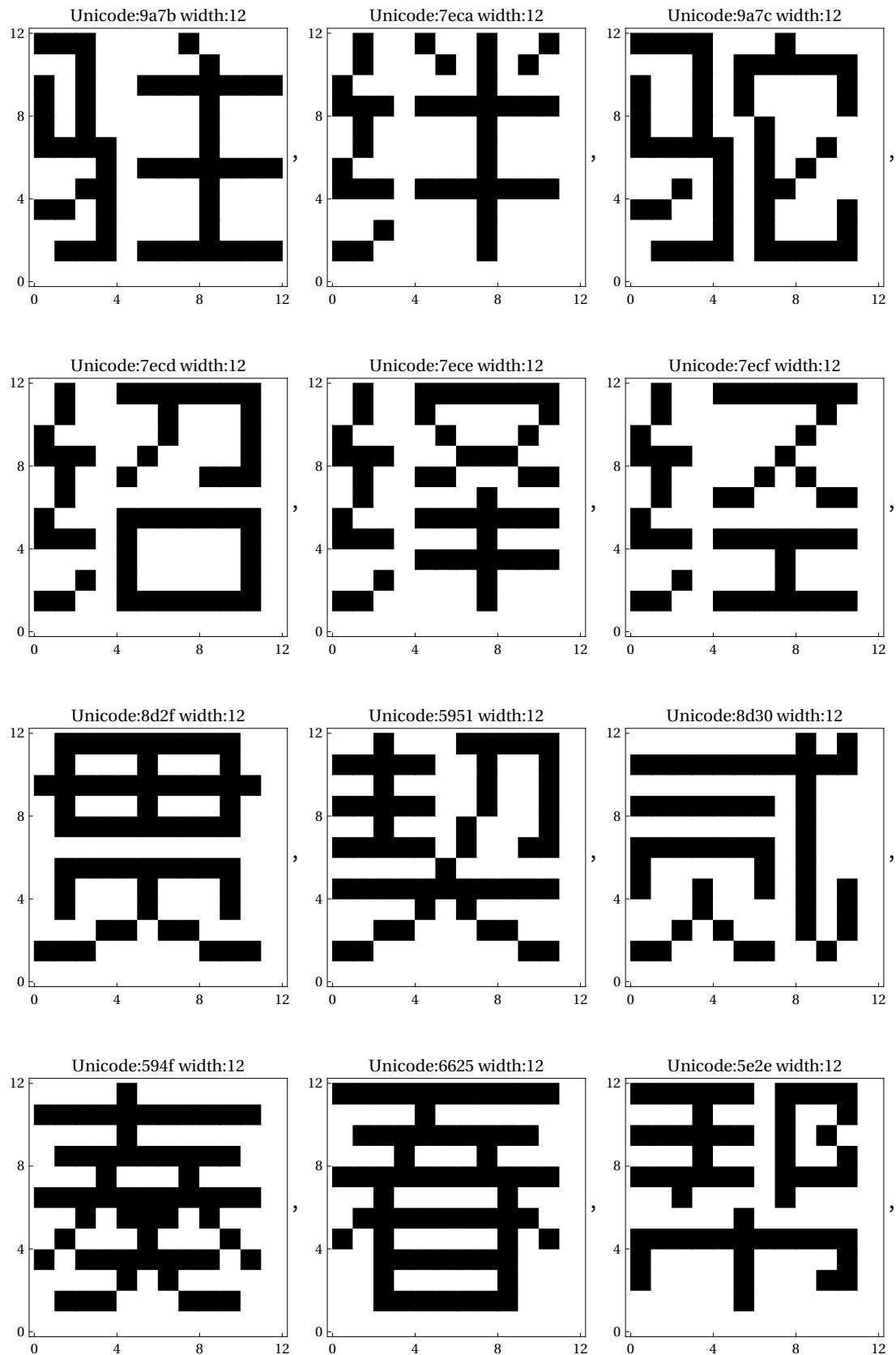


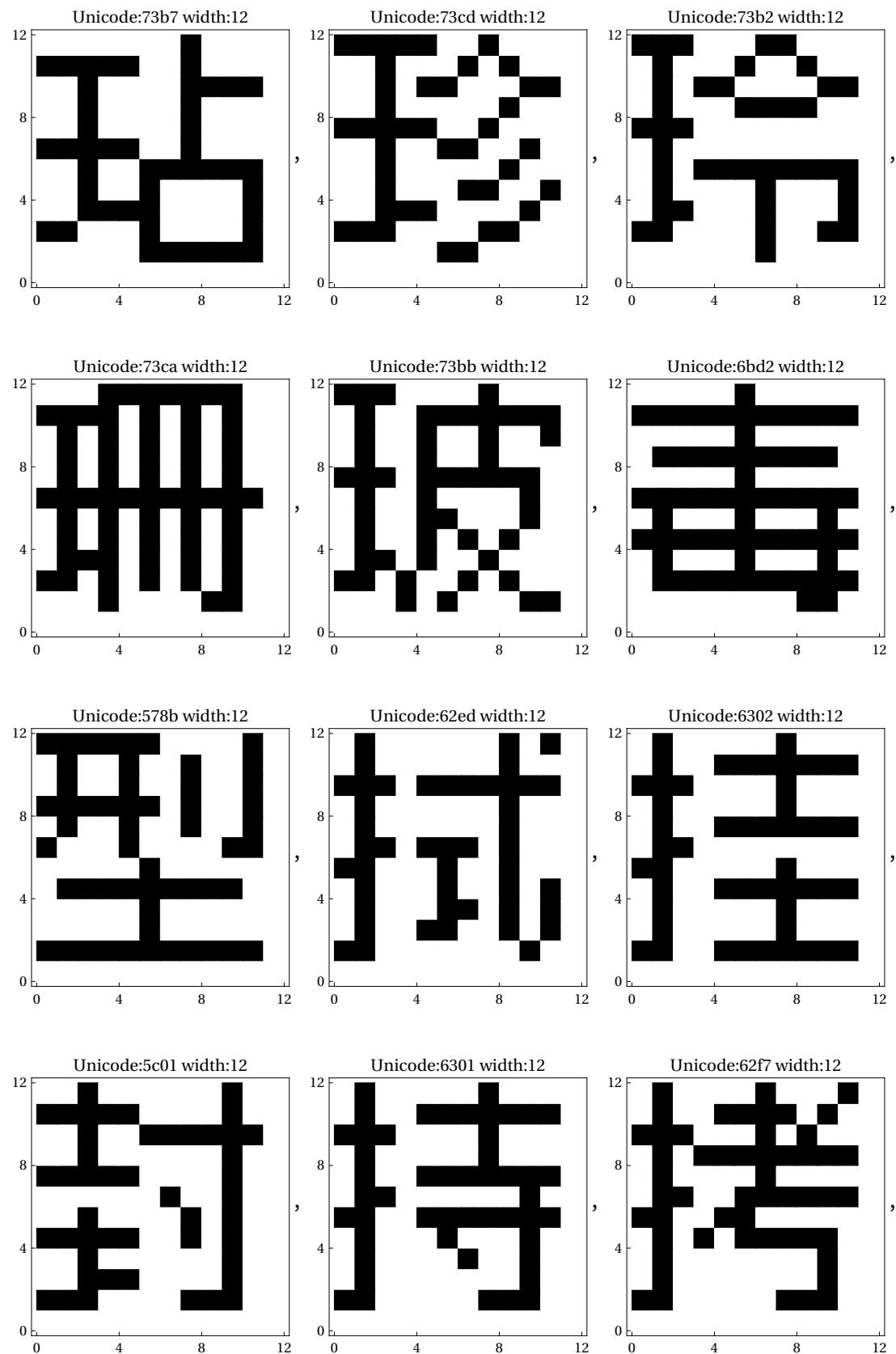


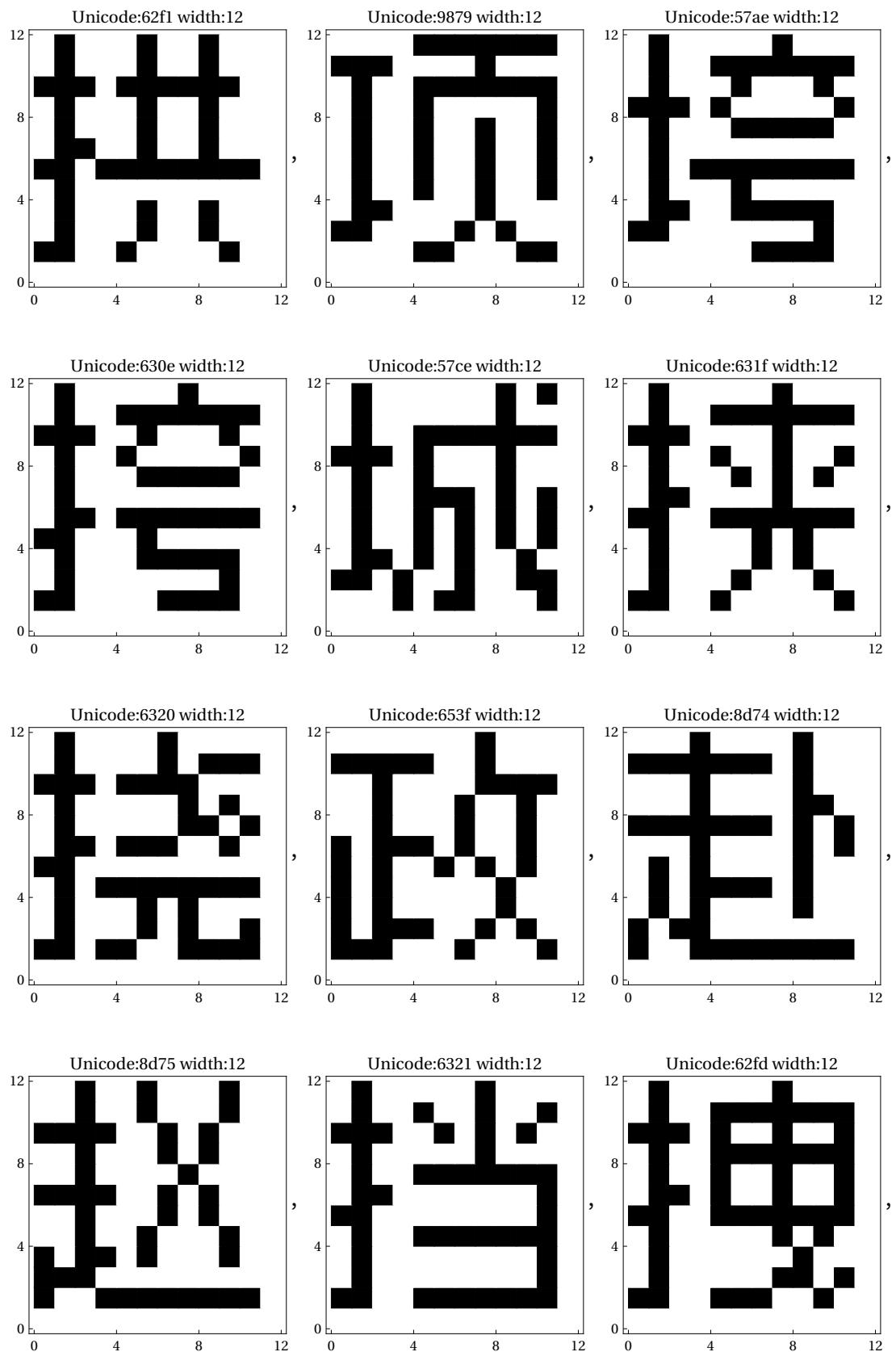


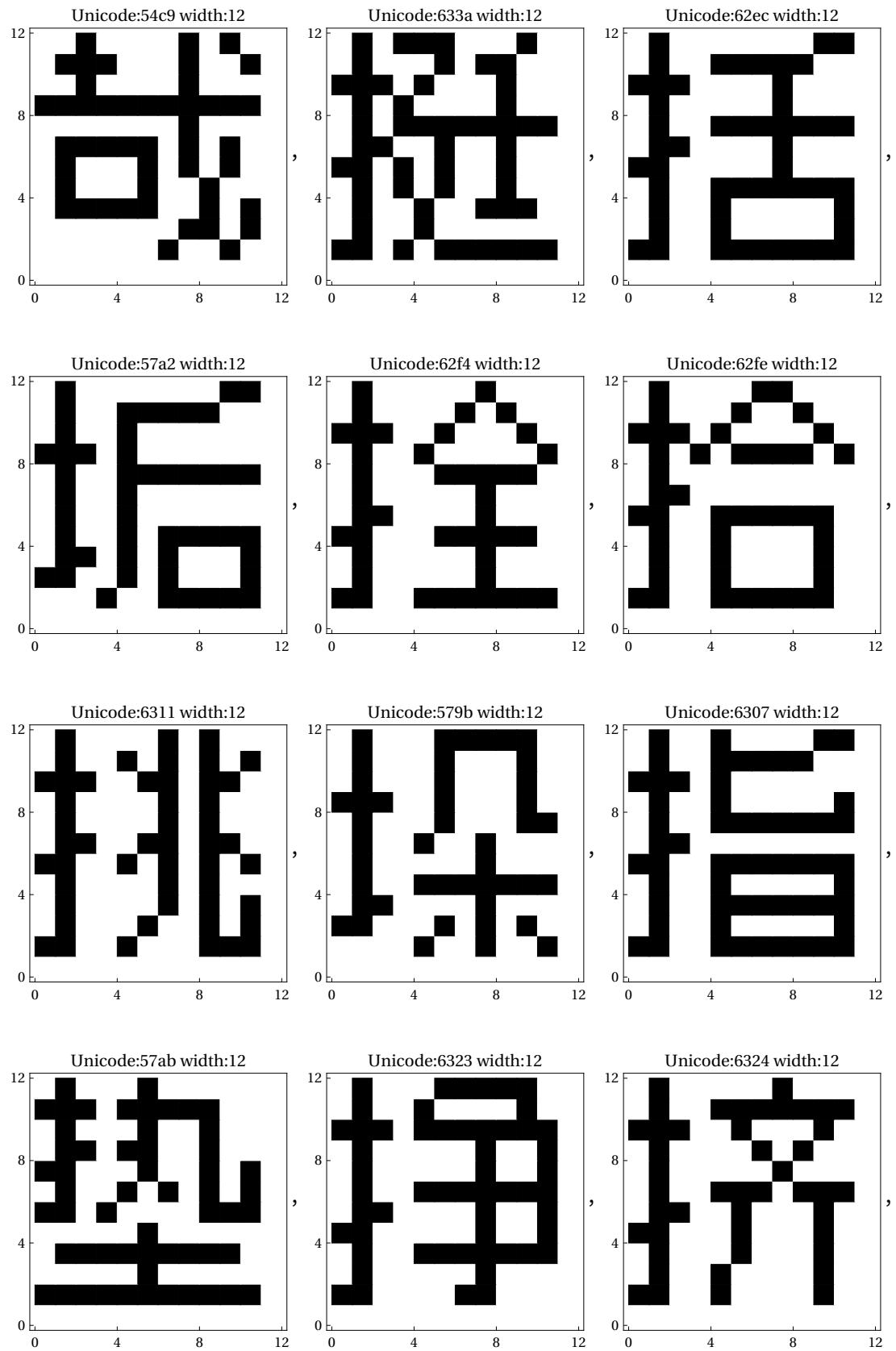


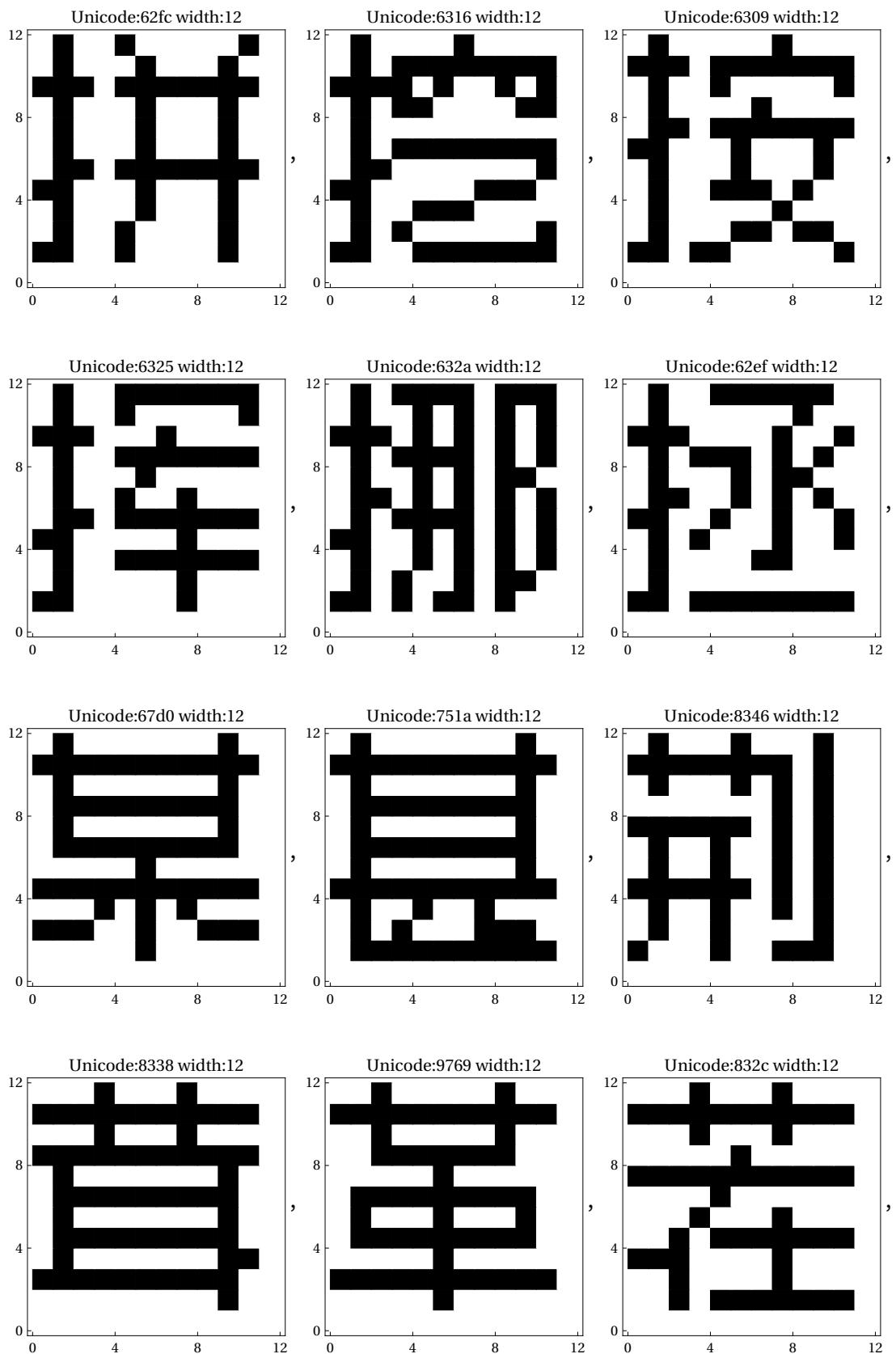


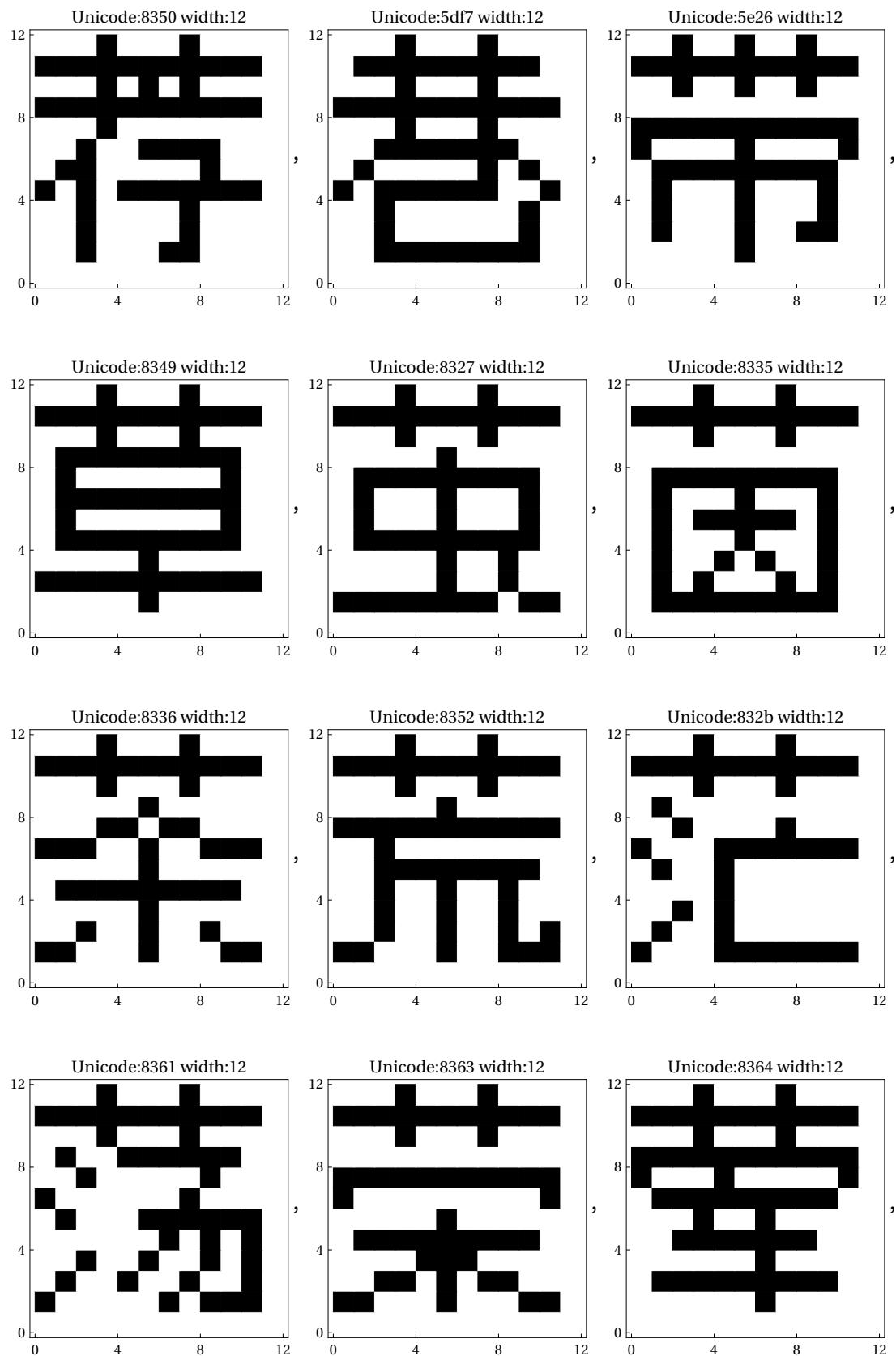


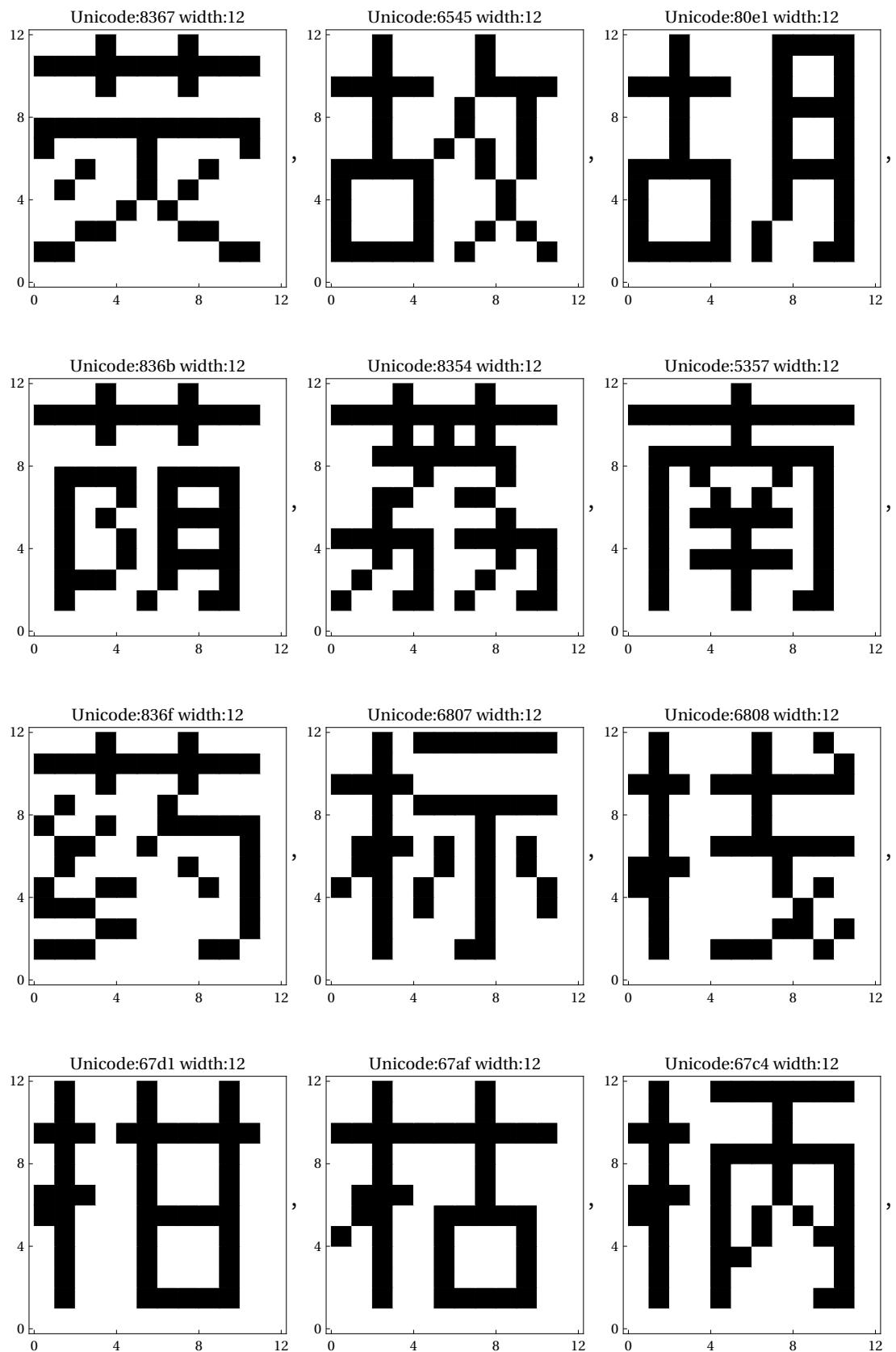


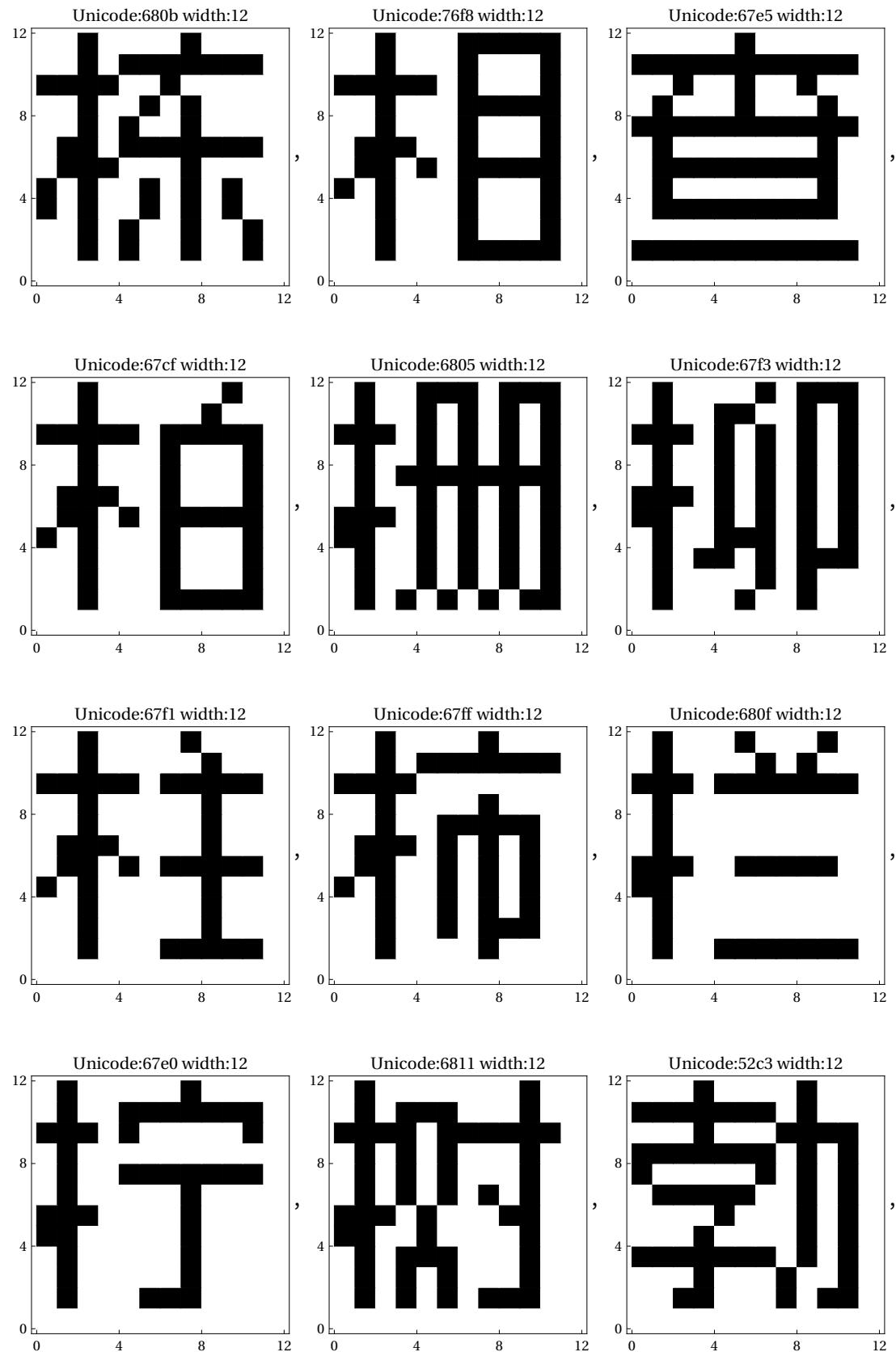


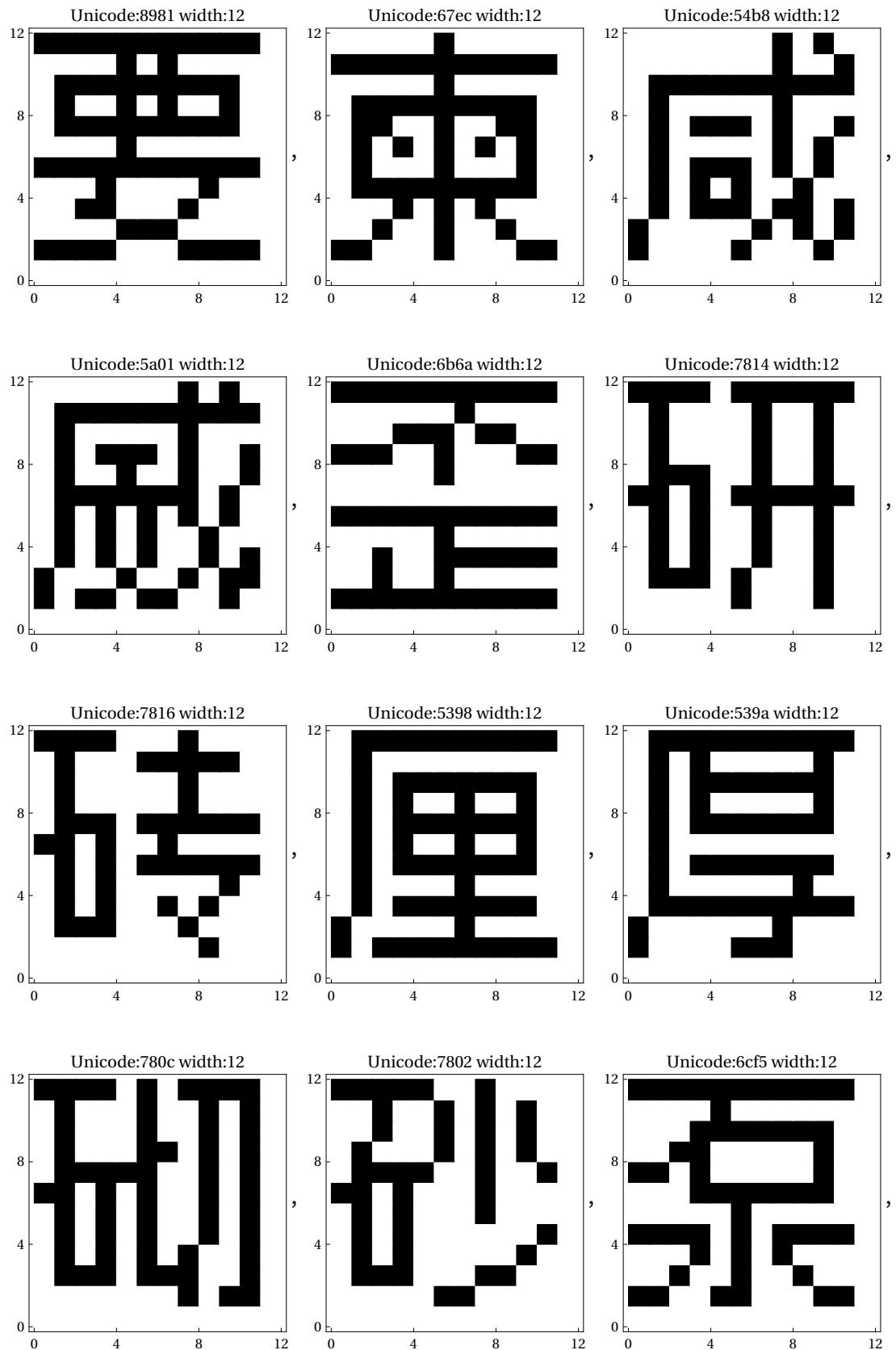


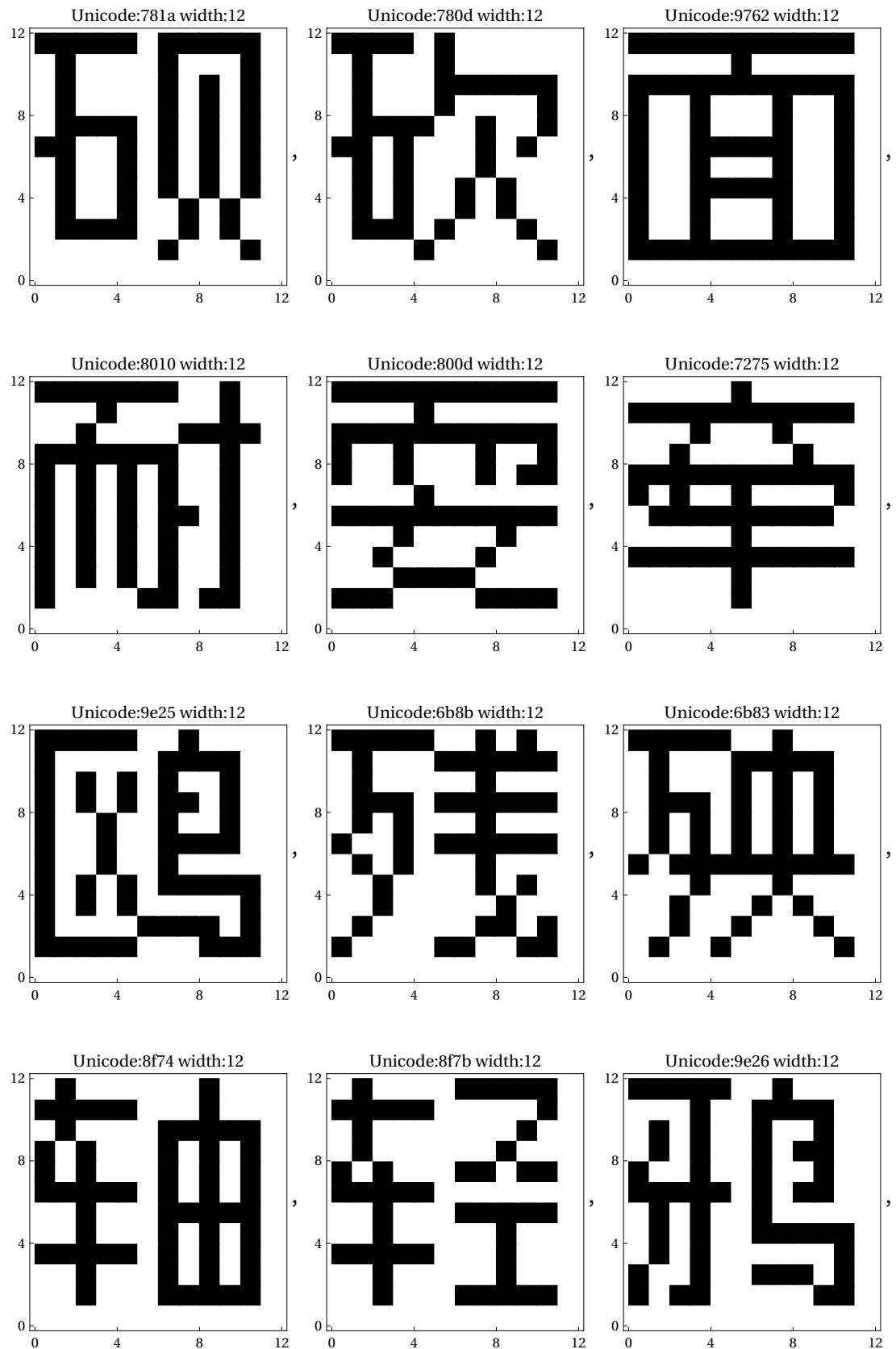


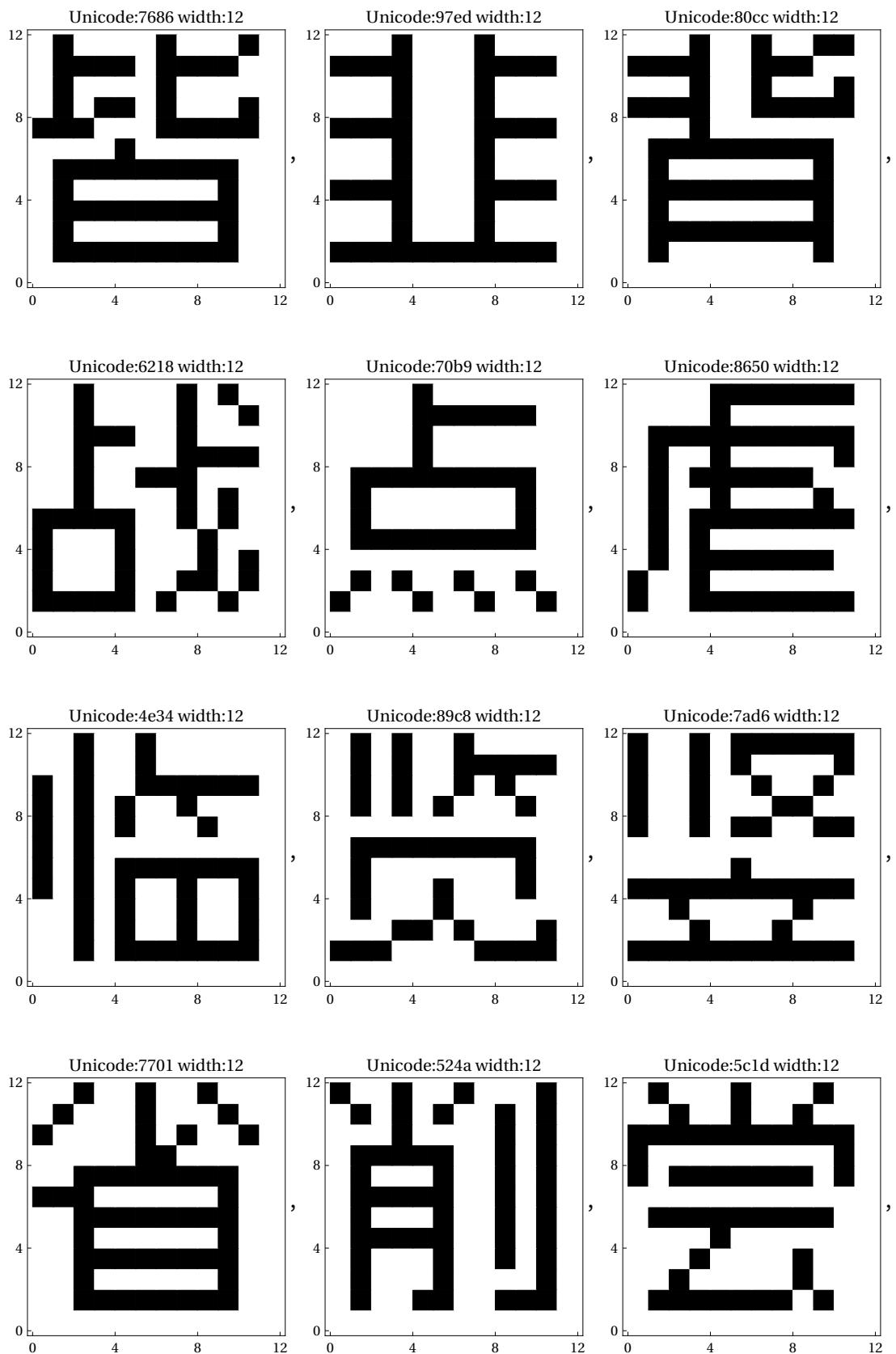


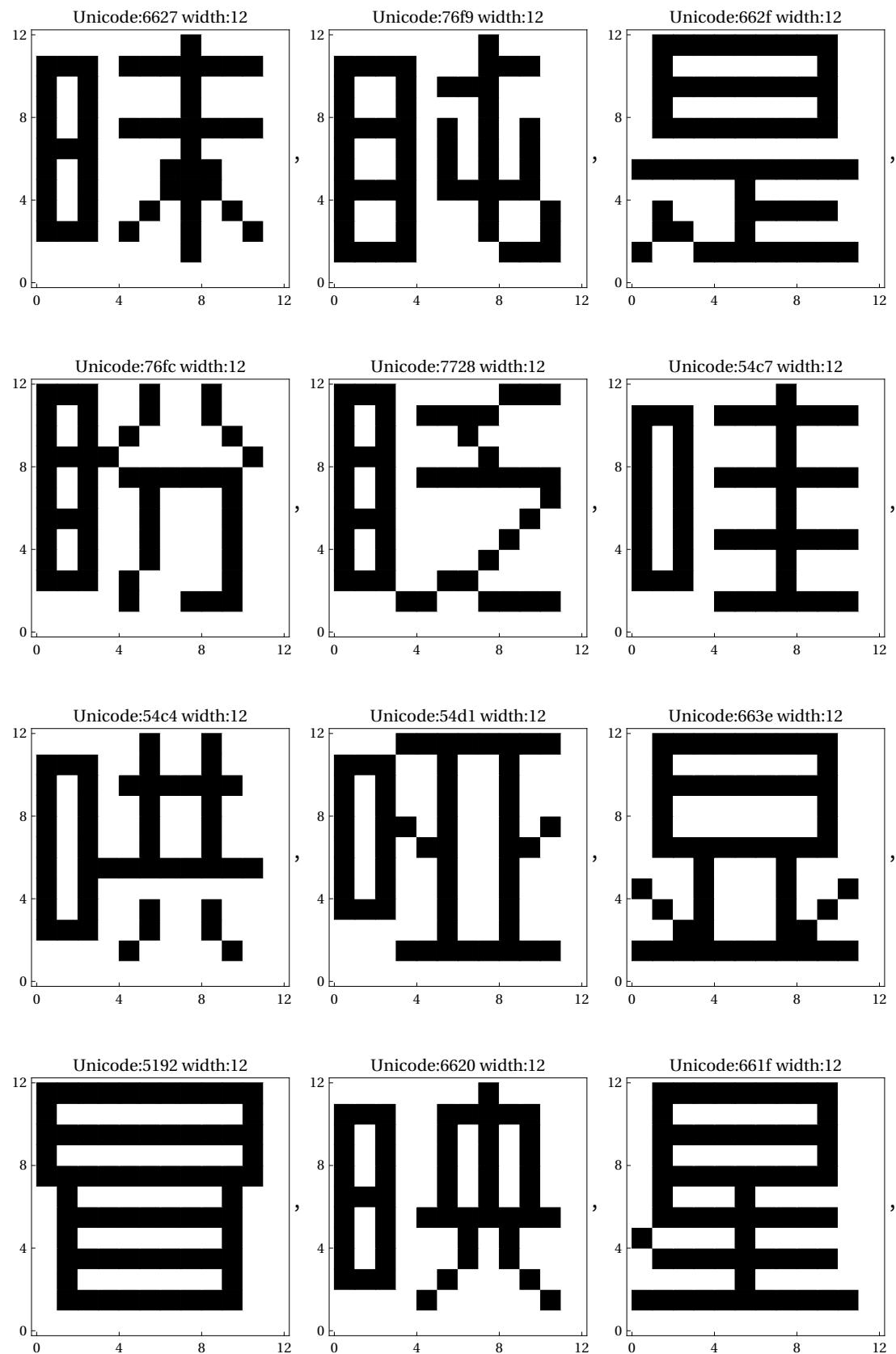


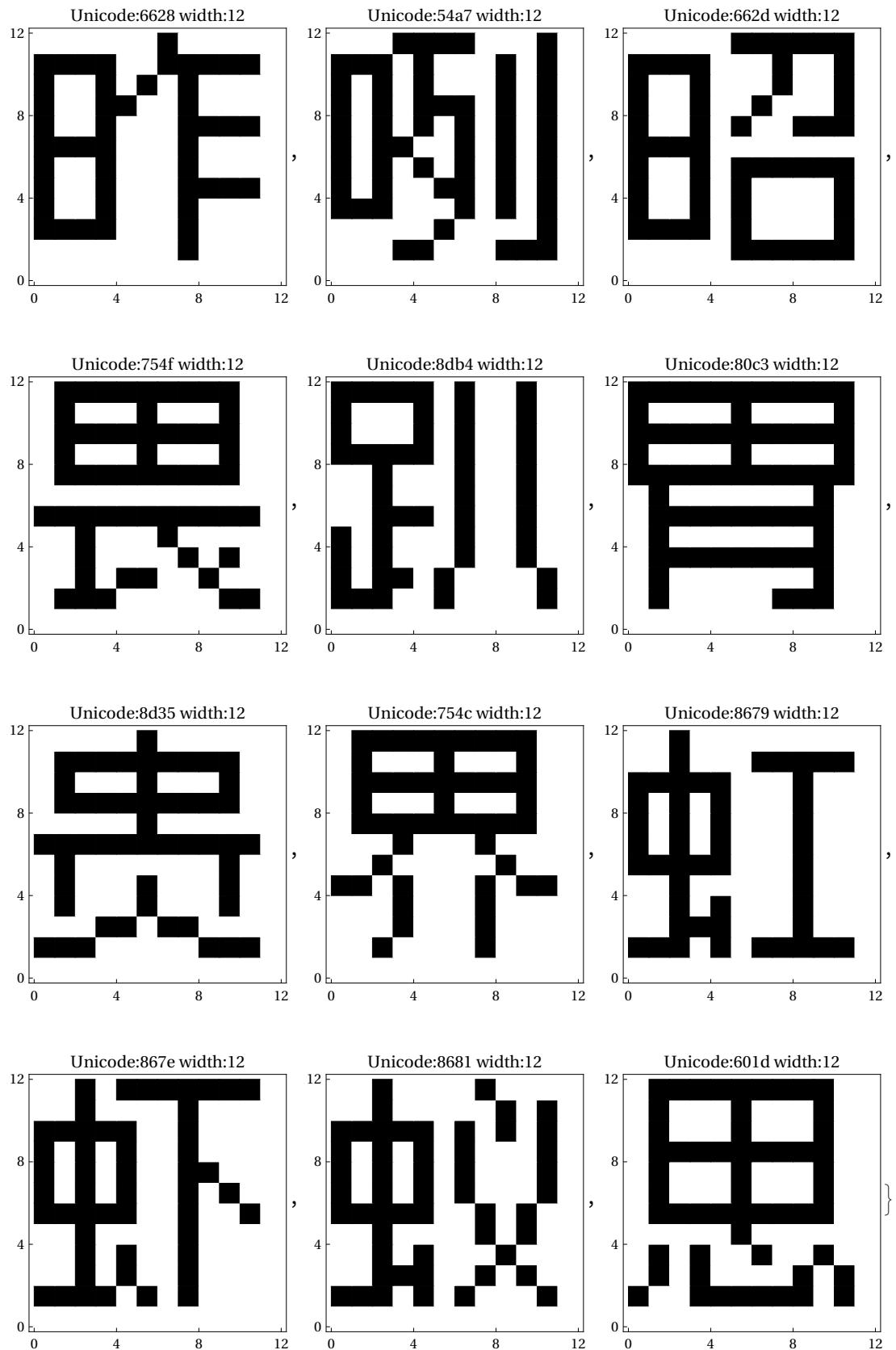













---

**Export bitmaps into C files**

## Build an extraction function, step by step

```
In[55]:= MatrixForm[#[[51]]] & /@ {trbitmaps, charcodes, widths}

Out[55]= { $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, 82, 6}$ 
```

- The sino-bit display is organized in columns, so transpose the  $12 \times 12$  array

```
In[56]:= MatrixForm[1 - Transpose[trbitmaps[[51]]]]
```

```
Out[56]//MatrixForm=  $\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 
```

- The columns get assembled into 12-bit words for starters

```
In[57]:= BaseForm[WordBuild /@ (1 - Transpose[trbitmaps[[51]]]), 2]
%
```

```
Out[57]//BaseForm=
{111111111002, 100010000002, 100010000002,
 100011000002, 11100111002, 02, 02, 02, 02, 02, 02}
```

```
Out[58]= {2044, 1088, 1088, 1120, 924, 0, 0, 0, 0, 0, 0, 0}
```

- For more efficient use of memory, the 12-bit words get split into 8+4 bits in two different arrays

```
In[59]:= BaseForm[Block[{t = (1 - Transpose[trbitmaps[[51]]])},
  {WordBuild[Take[#, 8]] & /@ t, WordBuild[Take[#, -4]] & /@ t}], 2]
%
```

```
Out[59]//BaseForm=
{{11111112, 10001002, 10001002, 10001102, 1110012, 02, 02, 02, 02, 02, 02, 02, 02, 02, 02, 02, 02},
 {11002, 02, 02, 02, 11002, 02, 02, 02, 02, 02, 02, 02, 02}}
```

```
Out[60]= {{127, 68, 68, 70, 57, 0, 0, 0, 0, 0, 0}, {12, 0, 0, 0, 12, 0, 0, 0, 0, 0, 0}}
```

■ The 4-bit array gets compressed into a byte array

```
In[61]:= BaseForm[Block[{t = (1 - Transpose[trbitmaps[[51]]])}, {WordBuild[Take[#, 8]] & /@ t, WordBuild /@ Partition[Flatten[Take[#, -4] & /@ t], 8]}], 2]
%
Out[61]//BaseForm=
{{111111112, 10001002, 10001002, 10001102, 1110012, 02, 02, 02, 02, 02, 02, 02}, {110000002, 02, 110000002, 02, 02, 02}}
Out[62]= {{127, 68, 68, 70, 57, 0, 0, 0, 0, 0, 0, 0}, {192, 0, 192, 0, 0, 0}}
```

■ The arrays get concatenated;  $12 \times 12$  bits get mapped into  $12 + \frac{12}{2} = 18$  bytes

```
In[63]:= BaseForm[
Block[{t = (1 - Transpose[trbitmaps[[51]]])}, Flatten[{WordBuild[Take[#, 8]] & /@ t, WordBuild /@ Partition[Flatten[Take[#, -4] & /@ t], 8]}]], 2]
%
Out[63]//BaseForm=
{111111112, 10001002, 10001002, 10001102, 1110012, 02, 02, 02, 02, 02, 110000002, 02, 110000002, 02, 02}
Out[64]= {127, 68, 68, 70, 57, 0, 0, 0, 0, 0, 0, 192, 0, 192, 0, 0, 0}
```

■ Now we have a function

```
In[65]:= ht1632bitmap = Function[bitmap12x12,
Block[{t = 1 - Transpose[bitmap12x12]}, Flatten[{WordBuild[Take[#, 8]] & /@ t, WordBuild /@ Partition[Flatten[Take[#, -4] & /@ t], 8]}]]]
Out[65]= Function[bitmap12x12,
Block[{t = 1 - Transpose[bitmap12x12]}, Flatten[{(WordBuild[Take[#, 8]] &) /@ t, WordBuild /@ Partition[Flatten[(Take[#, -4] &) /@ t], 8]}]]]
In[66]:= ht1632bitmap[trbitmaps[[51]]]
Out[66]= {127, 68, 68, 70, 57, 0, 0, 0, 0, 0, 0, 192, 0, 192, 0, 0, 0}
```

■ Export compressed glyph data

```
In[67]:= missingcode = Reverse[IntegerDigits[#, 2, 12] & @@
{2111111111111, 2111000000111, 2110111111011, 2101111111101,
2101011110101, 2101100001101, 2101111111101, 2101111111101,
2101101101101, 2110111111011, 2111000000111, 2111111111111}]
Out[67]= {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1},
{1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1}, {1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1},
{1, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},
{1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1}, {1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1},
{1, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1},
{1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}}
```

```
In[68]:= glyphstatement =
  Function[{vartype, varname, bitmaps}, vartype <> " " <> varname <> "[]" [18]= {\n    "
  <> StringRiffle[ToString[ht1632bitmap[#]] & /@ bitmaps, ",\n    "] <> "\n} ;\n"
Out[68]= Function[{vartype, varname, bitmaps}, vartype <> " " <> varname <> "[]" [18]= {
  <> StringRiffle[(ToString[ht1632bitmap[#1]] &) /@ bitmaps, ,
  ] <>
} ;
]
In[69]:= glyphstatement["const uint8_t", "glyph", {missingcode}]
Out[69]= const uint8_t glyph[] [18]= {
  {0, 31, 32, 65, 82, 66, 66, 82, 65, 32, 31, 0, 8, 66, 34, 34, 36, 128}
};
```

```
In[70]:= glyphstatement["const uint8_t", "glyph",
  Take[Lookup[lookupTable, #, {missingcode, {}, 12}] [[1]] & /@ TGSCCodes, 32]]

Out[70]= const uint8_t glyph[] [18] = {
  {8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0},
  {128, 129, 130, 132, 136, 144,
  160, 192, 192, 128, 1, 0, 194, 34, 34, 34, 34, 192},
  {0, 64, 64, 64, 64, 64, 64, 64, 64, 64, 0, 0, 68, 68, 68, 68, 68, 64},
  {4, 4, 4, 4, 4, 255, 4, 4, 4, 4, 0, 0, 0, 14, 0, 0, 0},
  {128, 128, 128, 128,
  255, 128, 128, 128, 128, 0, 0, 2, 46, 0, 0, 0},
  {0, 255, 128, 128, 128, 128, 128, 128, 128, 128, 0, 104, 0, 0, 0, 0, 0, 0},
  {8, 8, 255, 8, 16, 16, 16, 16, 32, 32, 32, 0, 0, 226, 34, 34, 34, 224},
  {0, 0, 0, 0, 255, 16, 16, 8, 8, 4, 2, 0, 0, 0, 224, 0, 0, 0},
  {0, 0, 7, 248, 0, 0, 0, 248, 7, 0, 0, 0, 44, 0, 0, 0, 12, 32},
  {0, 0, 0, 3, 12, 240, 12, 3, 0, 0, 0, 0, 36, 128, 0, 0, 132, 32},
  {0, 0, 128, 131, 140, 240, 12, 3, 0, 0, 0, 0, 36, 128, 0, 0, 132, 32},
  {0, 0, 0, 255, 0, 0, 0, 255, 0, 0, 1, 0, 34, 192, 0, 14, 34, 224},
  {255, 4, 4, 4, 8, 8, 8, 16, 16, 33, 0, 0, 0, 226, 34, 34, 34, 46, 0},
  {0, 0, 1, 254, 128, 128, 255, 0, 0, 1, 0, 36, 128, 0, 14, 34, 224},
  {32, 32, 33, 254, 32, 32, 32, 63, 0, 0, 3, 0, 36, 128, 0, 14, 34, 224},
  {129, 129, 129, 130, 130,
  132, 136, 136, 128, 255, 0, 0, 0, 0, 2, 34, 224},
  {128, 128, 128, 128, 128, 135, 136, 144, 160, 192, 0, 0, 0, 2, 46, 0, 0, 0},
  {128, 128, 129, 134, 248, 128,
  128, 128, 128, 128, 255, 0, 36, 128, 0, 2, 34, 224},
  {32, 32, 32, 35, 252, 32, 32, 32, 32, 63, 0, 36, 128, 0, 2, 34, 224},
  {128, 128, 135, 248, 128, 128,
  128, 152, 232, 136, 15, 0, 44, 0, 0, 2, 34, 224},
  {128, 176, 136, 132, 130, 129, 130, 132, 152,
  224, 0, 0, 34, 68, 128, 132, 66, 32},
  {64, 68, 68, 68, 68, 68, 68, 68, 68, 68, 64, 0, 34, 34, 34, 34, 34, 32},
  {4, 132, 132, 132, 132, 255, 132, 132, 132, 132, 4, 0, 0, 0, 0, 14, 0, 0, 0},
  {4, 132, 132, 132, 132, 255, 132, 132, 132, 132, 4, 0, 0, 2, 46, 0, 0, 0},
  {16, 18, 150, 154, 146, 146, 146, 146, 146, 19, 16, 0, 0, 4, 66, 36, 128, 0},
  {128, 128, 128, 128, 255,
  128, 128, 128, 128, 128, 0, 34, 34, 46, 34, 34, 32},
  {0, 8, 8, 8, 255, 8, 8, 8, 8, 0, 0, 34, 34, 46, 34, 34, 32},
  {16, 16, 16, 16, 16, 255, 16, 16, 16, 16, 0, 2, 34, 46, 34, 34, 0},
  {32, 32, 32, 33, 33, 34, 255, 36, 40, 32, 32, 0, 136, 128, 34, 224, 0, 0},
  {128, 128, 128, 128, 255,
  144, 136, 132, 130, 128, 0, 0, 0, 0, 224, 0, 0, 0},
  {32, 32, 36, 35, 32, 32, 32, 255, 32, 32, 32, 0, 0, 0, 2, 46, 0, 0},
  {32, 32, 32, 35, 44, 240, 44, 35, 32, 32, 0, 36, 128, 0, 0, 132, 32}
};
```

## ■ Export character codes

```
In[71]:= codestatement =
Function[{vartype, varname, codes}, vartype <> " " <> varname <> "[] [3] = {\n      {0x"
<> StringRiffle[
  StringRiffle[StringPartition[IntegerString[ToExpression[#], 16, 6], 2],
  ", 0x"] & /@ codes, "\n      {0x"} <> "}\n};\n"
]

Out[71]= Function[{vartype, varname, codes}, vartype <> " " <> varname <> "[] [3] = {
{0x" <> StringRiffle[
  (StringRiffle[StringPartition[IntegerString[ToExpression[#1], 16, 6], 2],
  ", 0x"] &) /@ codes, },
{0x} ] <> }
};

]
```

```
In[72]:= codestatement["const uint8_t", "codes", Take[TGSCCcodes, 32]]
```

```
Out[72]= const uint8_t codes[] [3] = {
    {0x00, 0x4e, 0x00},
    {0x00, 0x4e, 0x59},
    {0x00, 0x4e, 0x8c},
    {0x00, 0x53, 0x41},
    {0x00, 0x4e, 0x01},
    {0x00, 0x53, 0x82},
    {0x00, 0x4e, 0x03},
    {0x00, 0x53, 0x5c},
    {0x00, 0x51, 0x6b},
    {0x00, 0x4e, 0xba},
    {0x00, 0x51, 0x65},
    {0x00, 0x51, 0x3f},
    {0x00, 0x53, 0x15},
    {0x00, 0x51, 0xe0},
    {0x00, 0x4e, 0x5d},
    {0x00, 0x52, 0x01},
    {0x00, 0x4e, 0x86},
    {0x00, 0x52, 0x00},
    {0x00, 0x52, 0x9b},
    {0x00, 0x4e, 0x43},
    {0x00, 0x53, 0xc8},
    {0x00, 0x4e, 0x09},
    {0x00, 0x5e, 0x72},
    {0x00, 0x4e, 0x8e},
    {0x00, 0x4e, 0x8f},
    {0x00, 0x5d, 0xe5},
    {0x00, 0x57, 0x1f},
    {0x00, 0x58, 0xeb},
    {0x00, 0x62, 0x4d},
    {0x00, 0x4e, 0x0b},
    {0x00, 0x5b, 0xf8},
    {0x00, 0x59, 0x27}
};
```

## ■ Export the files

```
In[73]:= Export["charcode.c.snippet",
  codestatement["const uint16_t", "charcode", TGSCCcodes], "String"]
```

```
Out[73]= charcode.c.snippet
```

```
In[74]:= Export["glyph.c.snippet", glyphstatement["const uint8_t", "glyph",
  Lookup[lookuptable, #, {missingcode, {}, 12}] [1] & /@ TGSCCcodes], "String"]
```

```
Out[74]= glyph.c.snippet
```

## TGSCC codes missing in zpix font

### ■ Find glyphs missing from font

```
In[75]:= missingcodes = Select[TGSCCcodes, Not[KeyExistsQ[lookupTable, #]] &]
Out[75]= {14 800, 14 815, 17 373, 180 860, 14 616, 134 352, 18 813, 16 470, 155 351, 18 819, 15 878,
132 726, 182 489, 165 496, 180 693, 18 211, 182 494, 181 779, 179 039, 179 042, 14 801,
176 621, 177 156, 13 678, 179 575, 13 383, 146 584, 146 583, 180 697, 15 559, 182 497,
179 753, 18 618, 165 525, 177 692, 181 784, 177 693, 18 620, 158 556, 15 182, 182 786,
180 265, 180 266, 14 275, 183 081, 13 386, 176 995, 182 515, 182 909, 183 231, 183 232,
179 227, 183 541, 183 542, 17 209, 178 186, 14 019, 181 083, 40 909, 182 060, 158 753,
15 189, 19 731, 19 886, 176 034, 174 640, 177 383, 183 086, 183 089, 183 085, 17 377,
157 302, 15 794, 15 578, 15 576, 181 793, 183 549, 181 801, 176 423, 174 045, 181 092,
179 068, 182 063, 176 22, 180 393, 178 182, 164 872, 181 015, 177 249, 174 646, 183 096,
183 099, 183 097, 183 103, 183 105, 178 360, 177 639, 180 872, 181 384, 167 577,
14 042, 183 551, 176 424, 181 803, 16 558, 179 075, 13 901, 17 643, 17 644, 17 640,
17 627, 40 910, 19 733, 15 088, 183 382, 182 269, 17 283, 177 391, 178 169, 178 172,
16 352, 18 265, 17 394, 131 428, 19 732, 180 729, 181 089, 14 660, 182 535, 177 007,
182 538, 177 010, 183 246, 181 804, 181 805, 183 554, 177 702, 178 117, 181 807,
177 703, 13 912, 174 331, 136 090, 180 426, 183 765, 177 168, 183 932, 163 833,
179 591, 177 582, 40 911, 183 114, 166 983, 177 398, 183 118, 177 583, 17 411, 183 391,
177 901, 16 004, 179 703, 183 217, 180 900, 146 979, 15 636, 14 073, 183 555, 177 704,
19 734, 149 979, 182 953, 181 396, 16 581, 182 798, 177 171, 18 648, 178 887, 15 114,
177 401, 183 130, 183 131, 16 735, 181 570, 19 615, 183 691, 183 693, 157 564, 177 462,
176 896, 177 587, 177 706, 177 708, 178 089, 136 211, 174 359, 144 843, 183 832,
16 590, 181 399, 152 882, 177 814, 19 735, 15 118, 166 991, 183 140, 13 926, 177 813,
183 696, 183 695, 171 902, 183 834, 16 207, 15 294, 182 557, 181 826, 16 108, 17 878,
14 343, 183 145, 19 736, 171 907, 15 696, 183 843, 177 021, 183 562, 182 175, 152 930,
178 150, 183 944, 178 204, 15 130, 17 883, 14 355, 174 680, 183 148, 166 993, 183 151,
177 652, 183 846, 176 439, 150 141, 13 838, 183 155, 183 158, 177 421, 183 160, 166 996,
183 164, 177 422, 183 850, 19 616, 183 711, 183 712, 17 898, 183 720, 156 193, 176 440,
182 209, 153 000, 17 908, 19 618, 156 813, 17 302, 19 737, 181 834, 15 360, 183 725,
171 916, 165 856, 183 726, 166 729, 15 884, 181 835, 150 217, 183 955, 177 837}

In[76]:= Dimensions[missingcodes]
Out[76]= {276}

In[77]:= Flatten[{Position[TGSCCcodes, #], #}] & /@ missingcodes
Out[77]= {{3658, 14 800}, {3848, 14 815}, {3978, 17 373}, {4004, 180 860}, {4031, 14 616},
{4190, 134 352}, {4546, 18 813}, {5693, 16 470}, {5989, 155 351}, {6111, 18 819},
{6411, 15 878}, {6506, 132 726}, {6520, 182 489}, {6529, 165 496}, {6547, 180 693},
{6549, 18 211}, {6551, 182 494}, {6553, 181 779}, {6560, 179 039}, {6564, 179 042},
{6567, 14 801}, {6576, 176 621}, {6586, 177 156}, {6592, 13 678}, {6594, 179 575},
{6602, 13 383}, {6612, 146 584}, {6613, 146 583}, {6616, 180 697}, {6618, 15 559},
{6623, 182 497}, {6630, 179 753}, {6633, 18 618}, {6637, 165 525}, {6640, 177 692},
{6642, 181 784}, {6643, 177 693}, {6657, 18 620}, {6658, 158 556}, {6664, 15 182},
{6670, 182 786}, {6671, 180 265}, {6672, 180 266}, {6686, 14 275}, {6688, 183 081},
{6700, 13 386}, {6730, 176 995}, {6731, 182 515}, {6732, 182 909}, {6737, 183 231},
{6739, 183 232}, {6744, 179 227}, {6747, 183 541}, {6749, 183 542}, {6750, 17 209},
{6752, 178 186}, {6757, 14 019}, {6761, 181 083}, {6774, 40 909}, {6791, 182 060},
```

```
{6794, 158 753}, {6797, 15 189}, {6806, 19 731}, {6814, 19 886}, {6820, 176 034},
{6839, 174 640}, {6844, 177 383}, {6846, 183 086}, {6847, 183 089},
{6848, 183 085}, {6865, 17 377}, {6867, 157 302}, {6879, 15 794}, {6884, 15 578},
{6896, 15 576}, {6919, 181 793}, {6920, 183 549}, {6921, 181 801}, {6922, 176 423},
{6932, 174 045}, {6937, 181 092}, {6941, 179 068}, {6951, 182 063},
{6955, 17 622}, {6959, 180 393}, {6967, 178 182}, {6976, 164 872}, {6979, 181 015},
{6995, 177 249}, {6999, 174 646}, {7004, 183 096}, {7006, 183 099},
{7007, 183 097}, {7008, 183 103}, {7009, 183 105}, {7019, 178 360}, {7039, 177 639},
{7053, 180 872}, {7070, 181 384}, {7084, 167 577}, {7088, 14 042}, {7093, 183 551},
{7094, 176 424}, {7097, 181 803}, {7098, 16 558}, {7114, 179 075}, {7118, 13 901},
{7124, 17 643}, {7126, 17 644}, {7132, 17 640}, {7134, 17 627}, {7146, 40 910},
{7152, 19 733}, {7158, 15 088}, {7161, 183 382}, {7166, 182 269}, {7169, 17 283},
{7177, 177 391}, {7178, 178 169}, {7180, 178 172}, {7198, 16 352}, {7209, 18 265},
{7213, 17 394}, {7219, 131 428}, {7224, 19 732}, {7234, 180 729}, {7241, 181 089},
{7242, 14 660}, {7249, 182 535}, {7250, 177 007}, {7254, 182 538}, {7255, 177 010},
{7260, 183 246}, {7273, 181 804}, {7274, 181 805}, {7275, 183 554}, {7276, 177 702},
{7278, 178 117}, {7279, 181 807}, {7281, 177 703}, {7292, 13 912}, {7299, 174 331},
{7300, 136 090}, {7326, 180 426}, {7334, 183 765}, {7343, 177 168}, {7347, 183 932},
{7354, 163 833}, {7361, 179 591}, {7367, 177 582}, {7373, 40 911}, {7375, 183 114},
{7377, 166 983}, {7378, 177 398}, {7382, 183 118}, {7399, 177 583}, {7402, 17 411},
{7405, 183 391}, {7408, 177 901}, {7410, 16 004}, {7414, 179 703}, {7421, 183 217},
{7423, 180 900}, {7425, 146 979}, {7431, 15 636}, {7450, 14 073}, {7456, 183 555},
{7457, 177 704}, {7465, 19 734}, {7470, 149 979}, {7503, 182 953}, {7506, 181 396},
{7508, 16 581}, {7512, 182 798}, {7513, 177 171}, {7516, 18 648}, {7519, 178 887},
{7520, 15 114}, {7529, 177 401}, {7534, 183 130}, {7537, 183 131}, {7540, 16 735},
{7541, 181 570}, {7560, 19 615}, {7561, 183 691}, {7562, 183 693}, {7568, 157 564},
{7580, 177 462}, {7602, 176 896}, {7610, 177 587}, {7617, 177 706}, {7618, 177 708},
{7622, 178 089}, {7635, 136 211}, {7638, 174 359}, {7650, 144 843}, {7654, 183 832},
{7660, 16 590}, {7661, 181 399}, {7663, 152 882}, {7667, 177 814}, {7668, 19 735},
{7672, 15 118}, {7678, 166 991}, {7682, 183 140}, {7698, 13 926}, {7701, 177 813},
{7706, 183 696}, {7707, 183 695}, {7708, 171 902}, {7712, 183 834},
{7729, 16 207}, {7736, 15 294}, {7737, 182 557}, {7746, 181 826}, {7747, 16 108},
{7772, 17 878}, {7782, 14 343}, {7789, 183 145}, {7795, 19 736}, {7799, 171 907},
{7815, 15 696}, {7824, 183 843}, {7828, 177 021}, {7831, 183 562}, {7841, 182 175},
{7851, 152 930}, {7854, 178 150}, {7855, 183 944}, {7856, 178 204}, {7862, 15 130},
{7865, 17 883}, {7868, 14 355}, {7870, 174 680}, {7872, 183 148}, {7873, 166 993},
{7874, 183 151}, {7890, 177 652}, {7894, 183 846}, {7916, 176 439}, {7918, 150 141},
{7936, 13 838}, {7937, 183 155}, {7939, 183 158}, {7940, 177 421}, {7943, 183 160},
{7944, 166 996}, {7945, 183 164}, {7946, 177 422}, {7958, 183 850}, {7960, 19 616},
{7961, 183 711}, {7963, 183 712}, {7970, 17 898}, {7974, 183 720}, {7979, 156 193},
{7980, 176 440}, {7987, 182 209}, {7996, 153 000}, {8000, 17 908}, {8014, 19 618},
{8020, 156 813}, {8024, 17 302}, {8029, 19 737}, {8030, 181 834}, {8032, 15 360},
{8042, 183 725}, {8043, 171 916}, {8057, 165 856}, {8062, 183 726}, {8063, 166 729},
{8064, 15 884}, {8073, 181 835}, {8075, 150 217}, {8079, 183 955}, {8100, 177 837}}
```

In[78]:= BaseForm[missingcodes, 16]

```

Out[78]:= BaseForm=
{39d016, 39df16, 43dd16, 2c27c16, 391816, 20cd016, 497d16, 405616, 25ed716, 498316, 3e0616,
2067616, 2c8d916, 2867816, 2c1d516, 472316, 2c8de16, 2c61316, 2bb5f16, 2bb6216,
39d116, 2b1ed16, 2b40416, 356e16, 2bd7716, 344716, 23c9816, 23c9716, 2c1d916, 3cc716,
2c8e116, 2be2916, 48ba16, 2869516, 2b61c16, 2c61816, 2b61d16, 48bc16, 26b5c16, 3b4e16,
2ca0216, 2c02916, 2c02a16, 37c316, 2cb2916, 344a16, 2b36316, 2c8f316, 2ca7d16, 2cbbf16,
2cbc016, 2bc1b16, 2ccf516, 2ccf616, 433916, 2b80a16, 36c316, 2c35b16, 9fc16, 2c72c16,
26c2116, 3b5516, 4d1316, 4dae16, 2afa216, 2aa3016, 2b4e716, 2cb2e16, 2cb3116, 2cb2d16,
43e116, 2667616, 3db216, 3cda16, 3cd816, 2c62116, 2ccfd16, 2c62916, 2b12716, 2a7dd16,
2c36416, 2bb7c16, 2c72f16, 44d616, 2c0a916, 2b80616, 2840816, 2c31716, 2b46116,
2aa3616, 2cb3816, 2cb3b16, 2cb3916, 2cb3f16, 2cb4116, 2b8b816, 2b5e716, 2c28816,
2c48816, 28e9916, 36da16, 2ccff16, 2b12816, 2c62b16, 40ae16, 2bb8316, 364d16, 44eb16,
44ec16, 44e816, 44db16, 9fce16, 4d1516, 3af016, 2cc5616, 2c7fd16, 438316, 2b4ef16,
2b7f916, 2b7fc16, 3fe016, 475916, 43f216, 2016416, 4d1416, 2c1f916, 2c36116, 394416,
2c90716, 2b36f16, 2c90a16, 2b37216, 2cbce16, 2c62c16, 2c62d16, 2cd0216, 2b62616,
2b7c516, 2c62f16, 2b62716, 365816, 2a8fb16, 2139a16, 2c0ca16, 2cdd516, 2b41016,
2ce7c16, 27ff916, 2bd8716, 2b5ae16, 9fcf16, 2cb4a16, 28c4716, 2b4f616, 2cb4e16,
2b5af16, 440316, 2cc5f16, 2b6ed16, 3e8416, 2bdf716, 2cbb116, 2c2a416, 23e2316,
3d1416, 36f916, 2cd0316, 2b62816, 4d1616, 249db16, 2caa916, 2c49416, 40c516, 2ca0e16,
2b41316, 48d816, 2bac716, 3b0a16, 2b4f916, 2cb5a16, 2cb5b16, 415f16, 2c54216, 4c9f16,
2cd8b16, 2cd8d16, 2677c16, 2b53616, 2b30016, 2b5b316, 2b62a16, 2b62c16, 2b7a916,
2141316, 2a91716, 235cb16, 2ce1816, 40ce16, 2c49716, 2553216, 2b69616, 4d1716,
3b0e16, 28c4f16, 2cb6416, 366616, 2b69516, 2cd9016, 2cd8f16, 29f7e16, 2ce1a16,
3f4f16, 3bbe16, 2c91d16, 2c64216, 3eec16, 45d616, 380716, 2cb6916, 4d1816, 29f8316,
3d5016, 2ce2316, 2b37d16, 2cd0a16, 2c79f16, 2556216, 2b7e616, 2ce8816, 2b81c16,
3b1a16, 45db16, 381316, 2aa5816, 2cb6c16, 28c5116, 2cb6f16, 2b5f416, 2ce2616,
2b13716, 24a7d16, 360e16, 2cb7316, 2cb7616, 2b50d16, 2cb7816, 28c5416, 2cb7c16,
2b50e16, 2ce2a16, 4ca016, 2cd9f16, 2cda016, 45ea16, 2cda816, 2622116, 2b13816,
2c7c116, 255a816, 45f416, 4ca216, 2648d16, 439616, 4d1916, 2c64a16, 3c0016, 2cdad16,
29f8c16, 287e016, 2cdae16, 28b4916, 3e0c16, 2c64b16, 24ac916, 2ce9316, 2b6ad16}

In[79]:= Export["..../transmogrify/missingcodes.UTF32LE", missingcodes, "UnsignedInteger32"]

Out[79]= .../transmogrify/missingcodes.UTF32LE

In[80]:= Flatten[{#, 9, ToCharacterCode[IntegerString[Flatten[Position[TGSCCcodes, #]], 10]], 9, ToCharacterCode["U+" <> IntegerString[#, 16, 5]], 10}] & /@ missingcodes
Export["..../transmogrify/missingTGSCCcodes.UTF32LE", %, "UnsignedInteger32"]

Out[80]= {{14800, 9, 51, 54, 53, 56, 9, 85, 43, 48, 51, 57, 100, 48, 10}, {14815, 9, 51, 56, 52, 56, 9, 85, 43, 48, 51, 57, 100, 102, 10}, {17373, 9, 51, 57, 55, 56, 9, 85, 43, 48, 52, 51, 100, 100, 10}, {180860, 9, 52, 48, 48, 52, 9, 85, 43, 50, 99, 50, 55, 99, 10}, {14616, 9, 52, 48, 51, 49, 9, 85, 43, 48, 51, 57, 49, 56, 10}, {134352, 9, 52, 49, 57, 48, 9, 85, 43, 50, 48, 99, 100, 48, 10}, {18813, 9, 52, 53, 52, 54, 9, 85, 43, 48, 52, 57, 55, 100, 10}, {16470, 9, 53, 54, 57, 51, 9, 85, 43, 48, 52, 48, 53, 54, 10}, {155351, 9, 53, 57, 56, 57, 9, 85, 43, 50, 53, 101, 100, 55, 10}, {18819, 9, 54, 49, 49, 49, 9, 85, 43, 48, 52, 57, 56, 51, 10}, {15878, 9, 54, 52, 49, 49, 9, 85, 43, 48, 51, 101, 48, 54, 10}, {132726, 9, 54, 53, 48, 54, 9, 85, 43, 50, 48, 54, 55, 54, 10}, {182489, 9, 54, 53, 50, 48, 9, 85, 43, 50, 99, 56, 100, 57, 10}}

```

```

{165496, 9, 54, 53, 50, 57, 9, 85, 43, 50, 56, 54, 55, 56, 10},
{180693, 9, 54, 53, 52, 55, 9, 85, 43, 50, 99, 49, 100, 53, 10},
{18211, 9, 54, 53, 52, 57, 9, 85, 43, 48, 52, 55, 50, 51, 10},
{182494, 9, 54, 53, 53, 49, 9, 85, 43, 50, 99, 56, 100, 101, 10},
{181779, 9, 54, 53, 53, 51, 9, 85, 43, 50, 99, 54, 49, 51, 10},
{179039, 9, 54, 53, 54, 48, 9, 85, 43, 50, 98, 98, 53, 102, 10},
{179042, 9, 54, 53, 54, 52, 9, 85, 43, 50, 98, 98, 54, 50, 10},
{14801, 9, 54, 53, 54, 55, 9, 85, 43, 48, 51, 57, 100, 49, 10},
{176621, 9, 54, 53, 55, 54, 9, 85, 43, 50, 98, 49, 101, 100, 10},
{177156, 9, 54, 53, 56, 54, 9, 85, 43, 50, 98, 52, 48, 52, 10},
{13678, 9, 54, 53, 57, 50, 9, 85, 43, 48, 51, 53, 54, 101, 10},
{179575, 9, 54, 53, 57, 52, 9, 85, 43, 50, 98, 100, 55, 55, 10},
{13383, 9, 54, 54, 48, 50, 9, 85, 43, 48, 51, 52, 52, 55, 10},
{146584, 9, 54, 54, 49, 50, 9, 85, 43, 50, 51, 99, 57, 56, 10},
{146583, 9, 54, 54, 49, 51, 9, 85, 43, 50, 51, 99, 57, 55, 10},
{180697, 9, 54, 54, 49, 54, 9, 85, 43, 50, 99, 49, 100, 57, 10},
{15559, 9, 54, 54, 49, 56, 9, 85, 43, 48, 51, 99, 99, 55, 10},
{182497, 9, 54, 54, 50, 51, 9, 85, 43, 50, 99, 56, 101, 49, 10},
{179753, 9, 54, 54, 51, 48, 9, 85, 43, 50, 98, 101, 50, 57, 10},
{18618, 9, 54, 54, 51, 51, 9, 85, 43, 48, 52, 56, 98, 97, 10},
{165525, 9, 54, 54, 51, 55, 9, 85, 43, 50, 56, 54, 57, 53, 10},
{177692, 9, 54, 54, 52, 48, 9, 85, 43, 50, 98, 54, 49, 99, 10},
{181784, 9, 54, 54, 52, 50, 9, 85, 43, 50, 99, 54, 49, 56, 10},
{177693, 9, 54, 54, 52, 51, 9, 85, 43, 50, 98, 54, 49, 100, 10},
{18620, 9, 54, 54, 53, 55, 9, 85, 43, 48, 52, 56, 98, 99, 10},
{158556, 9, 54, 54, 53, 56, 9, 85, 43, 50, 54, 98, 53, 99, 10},
{15182, 9, 54, 54, 54, 52, 9, 85, 43, 48, 51, 98, 52, 101, 10},
{182786, 9, 54, 54, 55, 48, 9, 85, 43, 50, 99, 97, 48, 50, 10},
{180265, 9, 54, 54, 55, 49, 9, 85, 43, 50, 99, 48, 50, 57, 10},
{180266, 9, 54, 54, 55, 50, 9, 85, 43, 50, 99, 48, 50, 97, 10},
{14275, 9, 54, 54, 56, 54, 9, 85, 43, 48, 51, 55, 99, 51, 10},
{183081, 9, 54, 54, 56, 56, 9, 85, 43, 50, 99, 98, 50, 57, 10},
{13386, 9, 54, 55, 48, 48, 9, 85, 43, 48, 51, 52, 52, 97, 10},
{176995, 9, 54, 55, 51, 48, 9, 85, 43, 50, 98, 51, 54, 51, 10},
{182515, 9, 54, 55, 51, 49, 9, 85, 43, 50, 99, 56, 102, 51, 10},
{182909, 9, 54, 55, 51, 50, 9, 85, 43, 50, 99, 97, 55, 100, 10},
{183231, 9, 54, 55, 51, 55, 9, 85, 43, 50, 99, 98, 98, 102, 10},
{183232, 9, 54, 55, 51, 57, 9, 85, 43, 50, 99, 98, 99, 48, 10},
{179227, 9, 54, 55, 52, 52, 9, 85, 43, 50, 98, 99, 49, 98, 10},
{183541, 9, 54, 55, 52, 55, 9, 85, 43, 50, 99, 99, 102, 53, 10},
{183542, 9, 54, 55, 52, 57, 9, 85, 43, 50, 99, 99, 102, 54, 10},
{17209, 9, 54, 55, 53, 48, 9, 85, 43, 48, 52, 51, 51, 57, 10},
{178186, 9, 54, 55, 53, 50, 9, 85, 43, 50, 98, 56, 48, 97, 10},
{14019, 9, 54, 55, 53, 55, 9, 85, 43, 48, 51, 54, 99, 51, 10},
{181083, 9, 54, 55, 54, 49, 9, 85, 43, 50, 99, 51, 53, 98, 10},
{40909, 9, 54, 55, 55, 52, 9, 85, 43, 48, 57, 102, 99, 100, 10},
{182060, 9, 54, 55, 57, 49, 9, 85, 43, 50, 99, 55, 50, 99, 10},
{158753, 9, 54, 55, 57, 52, 9, 85, 43, 50, 54, 99, 50, 49, 10},
{15189, 9, 54, 55, 57, 55, 9, 85, 43, 48, 51, 98, 53, 53, 10},

```

{19731, 9, 54, 56, 48, 54, 9, 85, 43, 48, 52, 100, 49, 51, 10},  
{19886, 9, 54, 56, 49, 52, 9, 85, 43, 48, 52, 100, 97, 101, 10},  
{176034, 9, 54, 56, 50, 48, 9, 85, 43, 50, 97, 102, 97, 50, 10},  
{174640, 9, 54, 56, 51, 57, 9, 85, 43, 50, 97, 97, 51, 48, 10},  
{177383, 9, 54, 56, 52, 52, 9, 85, 43, 50, 98, 52, 101, 55, 10},  
{183086, 9, 54, 56, 52, 54, 9, 85, 43, 50, 99, 98, 50, 101, 10},  
{183089, 9, 54, 56, 52, 55, 9, 85, 43, 50, 99, 98, 51, 49, 10},  
{183085, 9, 54, 56, 52, 56, 9, 85, 43, 50, 99, 98, 50, 100, 10},  
{17377, 9, 54, 56, 54, 53, 9, 85, 43, 48, 52, 51, 101, 49, 10},  
{157302, 9, 54, 56, 54, 55, 9, 85, 43, 50, 54, 54, 55, 54, 10},  
{15794, 9, 54, 56, 55, 57, 9, 85, 43, 48, 51, 100, 98, 50, 10},  
{15578, 9, 54, 56, 56, 52, 9, 85, 43, 48, 51, 99, 100, 97, 10},  
{15576, 9, 54, 56, 57, 54, 9, 85, 43, 48, 51, 99, 100, 56, 10},  
{181793, 9, 54, 57, 49, 57, 9, 85, 43, 50, 99, 54, 50, 49, 10},  
{183549, 9, 54, 57, 50, 48, 9, 85, 43, 50, 99, 99, 102, 100, 10},  
{181801, 9, 54, 57, 50, 49, 9, 85, 43, 50, 99, 54, 50, 57, 10},  
{176423, 9, 54, 57, 50, 50, 9, 85, 43, 50, 98, 49, 50, 55, 10},  
{174045, 9, 54, 57, 51, 50, 9, 85, 43, 50, 97, 55, 100, 100, 10},  
{181092, 9, 54, 57, 51, 55, 9, 85, 43, 50, 99, 51, 54, 52, 10},  
{179068, 9, 54, 57, 52, 49, 9, 85, 43, 50, 98, 98, 55, 99, 10},  
{182063, 9, 54, 57, 53, 49, 9, 85, 43, 50, 99, 55, 50, 102, 10},  
{17622, 9, 54, 57, 53, 53, 9, 85, 43, 48, 52, 52, 100, 54, 10},  
{180393, 9, 54, 57, 53, 57, 9, 85, 43, 50, 99, 48, 97, 57, 10},  
{178182, 9, 54, 57, 54, 55, 9, 85, 43, 50, 98, 56, 48, 54, 10},  
{164872, 9, 54, 57, 55, 54, 9, 85, 43, 50, 56, 52, 48, 56, 10},  
{181015, 9, 54, 57, 55, 57, 9, 85, 43, 50, 99, 51, 49, 55, 10},  
{177249, 9, 54, 57, 57, 53, 9, 85, 43, 50, 98, 52, 54, 49, 10},  
{174646, 9, 54, 57, 57, 57, 9, 85, 43, 50, 97, 97, 51, 54, 10},  
{183096, 9, 55, 48, 48, 52, 9, 85, 43, 50, 99, 98, 51, 56, 10},  
{183099, 9, 55, 48, 48, 54, 9, 85, 43, 50, 99, 98, 51, 98, 10},  
{183097, 9, 55, 48, 48, 55, 9, 85, 43, 50, 99, 98, 51, 57, 10},  
{183103, 9, 55, 48, 48, 56, 9, 85, 43, 50, 99, 98, 51, 102, 10},  
{183105, 9, 55, 48, 48, 57, 9, 85, 43, 50, 99, 98, 52, 49, 10},  
{178360, 9, 55, 48, 49, 57, 9, 85, 43, 50, 98, 56, 98, 56, 10},  
{177639, 9, 55, 48, 51, 57, 9, 85, 43, 50, 98, 53, 101, 55, 10},  
{180872, 9, 55, 48, 53, 51, 9, 85, 43, 50, 99, 50, 56, 56, 10},  
{181384, 9, 55, 48, 55, 48, 9, 85, 43, 50, 99, 52, 56, 56, 10},  
{167577, 9, 55, 48, 56, 52, 9, 85, 43, 50, 56, 101, 57, 57, 10},  
{14042, 9, 55, 48, 56, 56, 9, 85, 43, 48, 51, 54, 100, 97, 10},  
{183551, 9, 55, 48, 57, 51, 9, 85, 43, 50, 99, 99, 102, 102, 10},  
{176424, 9, 55, 48, 57, 52, 9, 85, 43, 50, 98, 49, 50, 56, 10},  
{181803, 9, 55, 48, 57, 55, 9, 85, 43, 50, 99, 54, 50, 98, 10},  
{16558, 9, 55, 48, 57, 56, 9, 85, 43, 48, 52, 48, 97, 101, 10},  
{179075, 9, 55, 49, 49, 52, 9, 85, 43, 50, 98, 98, 56, 51, 10},  
{13901, 9, 55, 49, 49, 56, 9, 85, 43, 48, 51, 54, 52, 100, 10},  
{17643, 9, 55, 49, 50, 52, 9, 85, 43, 48, 52, 52, 101, 98, 10},  
{17644, 9, 55, 49, 50, 54, 9, 85, 43, 48, 52, 52, 101, 99, 10},  
{17640, 9, 55, 49, 51, 50, 9, 85, 43, 48, 52, 52, 101, 56, 10},  
{17627, 9, 55, 49, 51, 52, 9, 85, 43, 48, 52, 52, 100, 98, 10},

```

{40910, 9, 55, 49, 52, 54, 9, 85, 43, 48, 57, 102, 99, 101, 10},
{19733, 9, 55, 49, 53, 50, 9, 85, 43, 48, 52, 100, 49, 53, 10},
{15088, 9, 55, 49, 53, 56, 9, 85, 43, 48, 51, 97, 102, 48, 10},
{183382, 9, 55, 49, 54, 49, 9, 85, 43, 50, 99, 99, 53, 54, 10},
{182269, 9, 55, 49, 54, 54, 9, 85, 43, 50, 99, 55, 102, 100, 10},
{17283, 9, 55, 49, 54, 57, 9, 85, 43, 48, 52, 51, 56, 51, 10},
{177391, 9, 55, 49, 55, 55, 9, 85, 43, 50, 98, 52, 101, 102, 10},
{178169, 9, 55, 49, 55, 56, 9, 85, 43, 50, 98, 55, 102, 57, 10},
{178172, 9, 55, 49, 56, 48, 9, 85, 43, 50, 98, 55, 102, 99, 10},
{16352, 9, 55, 49, 57, 56, 9, 85, 43, 48, 51, 102, 101, 48, 10},
{18265, 9, 55, 50, 48, 57, 9, 85, 43, 48, 52, 55, 53, 57, 10},
{17394, 9, 55, 50, 49, 51, 9, 85, 43, 48, 52, 51, 102, 50, 10},
{131428, 9, 55, 50, 49, 57, 9, 85, 43, 50, 48, 49, 54, 52, 10},
{19732, 9, 55, 50, 50, 52, 9, 85, 43, 48, 52, 100, 49, 52, 10},
{180729, 9, 55, 50, 51, 52, 9, 85, 43, 50, 99, 49, 102, 57, 10},
{181089, 9, 55, 50, 52, 49, 9, 85, 43, 50, 99, 51, 54, 49, 10},
{14660, 9, 55, 50, 52, 50, 9, 85, 43, 48, 51, 57, 52, 52, 10},
{182535, 9, 55, 50, 52, 57, 9, 85, 43, 50, 99, 57, 48, 55, 10},
{177007, 9, 55, 50, 53, 48, 9, 85, 43, 50, 98, 51, 54, 102, 10},
{182538, 9, 55, 50, 53, 52, 9, 85, 43, 50, 99, 57, 48, 97, 10},
{177010, 9, 55, 50, 53, 53, 9, 85, 43, 50, 98, 51, 55, 50, 10},
{183246, 9, 55, 50, 54, 48, 9, 85, 43, 50, 99, 98, 99, 101, 10},
{181804, 9, 55, 50, 55, 51, 9, 85, 43, 50, 99, 54, 50, 99, 10},
{181805, 9, 55, 50, 55, 52, 9, 85, 43, 50, 99, 54, 50, 100, 10},
{183554, 9, 55, 50, 55, 53, 9, 85, 43, 50, 99, 100, 48, 50, 10},
{177702, 9, 55, 50, 55, 54, 9, 85, 43, 50, 98, 54, 50, 54, 10},
{178117, 9, 55, 50, 55, 56, 9, 85, 43, 50, 98, 55, 99, 53, 10},
{181807, 9, 55, 50, 55, 57, 9, 85, 43, 50, 99, 54, 50, 102, 10},
{177703, 9, 55, 50, 56, 49, 9, 85, 43, 50, 98, 54, 50, 55, 10},
{13912, 9, 55, 50, 57, 50, 9, 85, 43, 48, 51, 54, 53, 56, 10},
{174331, 9, 55, 50, 57, 57, 9, 85, 43, 50, 97, 56, 102, 98, 10},
{136090, 9, 55, 51, 48, 48, 9, 85, 43, 50, 49, 51, 57, 97, 10},
{180426, 9, 55, 51, 50, 54, 9, 85, 43, 50, 99, 48, 99, 97, 10},
{183765, 9, 55, 51, 51, 52, 9, 85, 43, 50, 99, 100, 100, 53, 10},
{177168, 9, 55, 51, 52, 51, 9, 85, 43, 50, 98, 52, 49, 48, 10},
{183932, 9, 55, 51, 52, 55, 9, 85, 43, 50, 99, 101, 55, 99, 10},
{163833, 9, 55, 51, 53, 52, 9, 85, 43, 50, 55, 102, 102, 57, 10},
{179591, 9, 55, 51, 54, 49, 9, 85, 43, 50, 98, 100, 56, 55, 10},
{177582, 9, 55, 51, 54, 55, 9, 85, 43, 50, 98, 53, 97, 101, 10},
{40911, 9, 55, 51, 55, 51, 9, 85, 43, 48, 57, 102, 99, 102, 10},
{183114, 9, 55, 51, 55, 53, 9, 85, 43, 50, 99, 98, 52, 97, 10},
{166983, 9, 55, 51, 55, 55, 9, 85, 43, 50, 56, 99, 52, 55, 10},
{177398, 9, 55, 51, 55, 56, 9, 85, 43, 50, 98, 52, 102, 54, 10},
{183118, 9, 55, 51, 56, 50, 9, 85, 43, 50, 99, 98, 52, 101, 10},
{177583, 9, 55, 51, 57, 57, 9, 85, 43, 50, 98, 53, 97, 102, 10},
{17411, 9, 55, 52, 48, 50, 9, 85, 43, 48, 52, 52, 48, 51, 10},
{183391, 9, 55, 52, 48, 53, 9, 85, 43, 50, 99, 99, 53, 102, 10},
{177901, 9, 55, 52, 48, 56, 9, 85, 43, 50, 98, 54, 101, 100, 10},
{16004, 9, 55, 52, 49, 48, 9, 85, 43, 48, 51, 101, 56, 52, 10},

```

{179703, 9, 55, 52, 49, 52, 9, 85, 43, 50, 98, 100, 102, 55, 10},  
{183217, 9, 55, 52, 50, 49, 9, 85, 43, 50, 99, 98, 98, 49, 10},  
{180900, 9, 55, 52, 50, 51, 9, 85, 43, 50, 99, 50, 97, 52, 10},  
{146979, 9, 55, 52, 50, 53, 9, 85, 43, 50, 51, 101, 50, 51, 10},  
{15636, 9, 55, 52, 51, 49, 9, 85, 43, 48, 51, 100, 49, 52, 10},  
{14073, 9, 55, 52, 53, 48, 9, 85, 43, 48, 51, 54, 102, 57, 10},  
{183555, 9, 55, 52, 53, 54, 9, 85, 43, 50, 99, 100, 48, 51, 10},  
{177704, 9, 55, 52, 53, 55, 9, 85, 43, 50, 98, 54, 50, 56, 10},  
{19734, 9, 55, 52, 54, 53, 9, 85, 43, 48, 52, 100, 49, 54, 10},  
{149979, 9, 55, 52, 55, 48, 9, 85, 43, 50, 52, 57, 100, 98, 10},  
{182953, 9, 55, 53, 48, 51, 9, 85, 43, 50, 99, 97, 97, 57, 10},  
{181396, 9, 55, 53, 48, 54, 9, 85, 43, 50, 99, 52, 57, 52, 10},  
{16581, 9, 55, 53, 48, 56, 9, 85, 43, 48, 52, 48, 99, 53, 10},  
{182798, 9, 55, 53, 49, 50, 9, 85, 43, 50, 99, 97, 48, 101, 10},  
{177171, 9, 55, 53, 49, 51, 9, 85, 43, 50, 98, 52, 49, 51, 10},  
{18648, 9, 55, 53, 49, 54, 9, 85, 43, 48, 52, 56, 100, 56, 10},  
{178887, 9, 55, 53, 49, 57, 9, 85, 43, 50, 98, 97, 99, 55, 10},  
{15114, 9, 55, 53, 50, 48, 9, 85, 43, 48, 51, 98, 48, 97, 10},  
{177401, 9, 55, 53, 50, 57, 9, 85, 43, 50, 98, 52, 102, 57, 10},  
{183130, 9, 55, 53, 51, 52, 9, 85, 43, 50, 99, 98, 53, 97, 10},  
{183131, 9, 55, 53, 51, 55, 9, 85, 43, 50, 99, 98, 53, 98, 10},  
{16735, 9, 55, 53, 52, 48, 9, 85, 43, 48, 52, 49, 53, 102, 10},  
{181570, 9, 55, 53, 52, 49, 9, 85, 43, 50, 99, 53, 52, 50, 10},  
{19615, 9, 55, 53, 54, 48, 9, 85, 43, 48, 52, 99, 57, 102, 10},  
{183691, 9, 55, 53, 54, 49, 9, 85, 43, 50, 99, 100, 56, 98, 10},  
{183693, 9, 55, 53, 54, 50, 9, 85, 43, 50, 99, 100, 56, 100, 10},  
{157564, 9, 55, 53, 54, 56, 9, 85, 43, 50, 54, 55, 55, 99, 10},  
{177462, 9, 55, 53, 56, 48, 9, 85, 43, 50, 98, 53, 51, 54, 10},  
{176896, 9, 55, 54, 48, 50, 9, 85, 43, 50, 98, 51, 48, 48, 10},  
{177587, 9, 55, 54, 49, 48, 9, 85, 43, 50, 98, 53, 98, 51, 10},  
{177706, 9, 55, 54, 49, 55, 9, 85, 43, 50, 98, 54, 50, 97, 10},  
{177708, 9, 55, 54, 49, 56, 9, 85, 43, 50, 98, 54, 50, 99, 10},  
{178089, 9, 55, 54, 50, 50, 9, 85, 43, 50, 98, 55, 97, 57, 10},  
{136211, 9, 55, 54, 51, 53, 9, 85, 43, 50, 49, 52, 49, 51, 10},  
{174359, 9, 55, 54, 51, 56, 9, 85, 43, 50, 97, 57, 49, 55, 10},  
{144843, 9, 55, 54, 53, 48, 9, 85, 43, 50, 51, 53, 99, 98, 10},  
{183832, 9, 55, 54, 53, 52, 9, 85, 43, 50, 99, 101, 49, 56, 10},  
{16590, 9, 55, 54, 54, 48, 9, 85, 43, 48, 52, 48, 99, 101, 10},  
{181399, 9, 55, 54, 54, 49, 9, 85, 43, 50, 99, 52, 57, 55, 10},  
{152882, 9, 55, 54, 54, 51, 9, 85, 43, 50, 53, 53, 51, 50, 10},  
{177814, 9, 55, 54, 54, 55, 9, 85, 43, 50, 98, 54, 57, 54, 10},  
{19735, 9, 55, 54, 54, 56, 9, 85, 43, 48, 52, 100, 49, 55, 10},  
{15118, 9, 55, 54, 55, 50, 9, 85, 43, 48, 51, 98, 48, 101, 10},  
{166991, 9, 55, 54, 55, 56, 9, 85, 43, 50, 56, 99, 52, 102, 10},  
{183140, 9, 55, 54, 56, 50, 9, 85, 43, 50, 99, 98, 54, 52, 10},  
{13926, 9, 55, 54, 57, 56, 9, 85, 43, 48, 51, 54, 54, 54, 10},  
{177813, 9, 55, 55, 48, 49, 9, 85, 43, 50, 98, 54, 57, 53, 10},  
{183696, 9, 55, 55, 48, 54, 9, 85, 43, 50, 99, 100, 57, 48, 10},  
{183695, 9, 55, 55, 48, 55, 9, 85, 43, 50, 99, 100, 56, 102, 10},

```

{171902, 9, 55, 55, 48, 56, 9, 85, 43, 50, 57, 102, 55, 101, 10},  

{183834, 9, 55, 55, 49, 50, 9, 85, 43, 50, 99, 101, 49, 97, 10},  

{16207, 9, 55, 55, 50, 57, 9, 85, 43, 48, 51, 102, 52, 102, 10},  

{15294, 9, 55, 55, 51, 54, 9, 85, 43, 48, 51, 98, 98, 101, 10},  

{182557, 9, 55, 55, 51, 55, 9, 85, 43, 50, 99, 57, 49, 100, 10},  

{181826, 9, 55, 55, 52, 54, 9, 85, 43, 50, 99, 54, 52, 50, 10},  

{16108, 9, 55, 55, 52, 55, 9, 85, 43, 48, 51, 101, 101, 99, 10},  

{17878, 9, 55, 55, 55, 50, 9, 85, 43, 48, 52, 53, 100, 54, 10},  

{14343, 9, 55, 55, 56, 50, 9, 85, 43, 48, 51, 56, 48, 55, 10},  

{183145, 9, 55, 55, 56, 57, 9, 85, 43, 50, 99, 98, 54, 57, 10},  

{19736, 9, 55, 55, 57, 53, 9, 85, 43, 48, 52, 100, 49, 56, 10},  

{171907, 9, 55, 55, 57, 57, 9, 85, 43, 50, 57, 102, 56, 51, 10},  

{15696, 9, 55, 56, 49, 53, 9, 85, 43, 48, 51, 100, 53, 48, 10},  

{183843, 9, 55, 56, 50, 52, 9, 85, 43, 50, 99, 101, 50, 51, 10},  

{177021, 9, 55, 56, 50, 56, 9, 85, 43, 50, 98, 51, 55, 100, 10},  

{183562, 9, 55, 56, 51, 49, 9, 85, 43, 50, 99, 100, 48, 97, 10},  

{182175, 9, 55, 56, 52, 49, 9, 85, 43, 50, 99, 55, 57, 102, 10},  

{152930, 9, 55, 56, 53, 49, 9, 85, 43, 50, 53, 53, 54, 50, 10},  

{178150, 9, 55, 56, 53, 52, 9, 85, 43, 50, 98, 55, 101, 54, 10},  

{183944, 9, 55, 56, 53, 53, 9, 85, 43, 50, 99, 101, 56, 56, 10},  

{178204, 9, 55, 56, 53, 54, 9, 85, 43, 50, 98, 56, 49, 99, 10},  

{15130, 9, 55, 56, 54, 50, 9, 85, 43, 48, 51, 98, 49, 97, 10},  

{17883, 9, 55, 56, 54, 53, 9, 85, 43, 48, 52, 53, 100, 98, 10},  

{14355, 9, 55, 56, 54, 56, 9, 85, 43, 48, 51, 56, 49, 51, 10},  

{174680, 9, 55, 56, 55, 48, 9, 85, 43, 50, 97, 97, 53, 56, 10},  

{183148, 9, 55, 56, 55, 50, 9, 85, 43, 50, 99, 98, 54, 99, 10},  

{166993, 9, 55, 56, 55, 51, 9, 85, 43, 50, 56, 99, 53, 49, 10},  

{183151, 9, 55, 56, 55, 52, 9, 85, 43, 50, 99, 98, 54, 102, 10},  

{177652, 9, 55, 56, 57, 48, 9, 85, 43, 50, 98, 53, 102, 52, 10},  

{183846, 9, 55, 56, 57, 52, 9, 85, 43, 50, 99, 101, 50, 54, 10},  

{176439, 9, 55, 57, 49, 54, 9, 85, 43, 50, 98, 49, 51, 55, 10},  

{150141, 9, 55, 57, 49, 56, 9, 85, 43, 50, 52, 97, 55, 100, 10},  

{13838, 9, 55, 57, 51, 54, 9, 85, 43, 48, 51, 54, 48, 101, 10},  

{183155, 9, 55, 57, 51, 55, 9, 85, 43, 50, 99, 98, 55, 51, 10},  

{183158, 9, 55, 57, 51, 57, 9, 85, 43, 50, 99, 98, 55, 54, 10},  

{177421, 9, 55, 57, 52, 48, 9, 85, 43, 50, 98, 53, 48, 100, 10},  

{183160, 9, 55, 57, 52, 51, 9, 85, 43, 50, 99, 98, 55, 56, 10},  

{166996, 9, 55, 57, 52, 52, 9, 85, 43, 50, 56, 99, 53, 52, 10},  

{183164, 9, 55, 57, 52, 53, 9, 85, 43, 50, 99, 98, 55, 99, 10},  

{177422, 9, 55, 57, 52, 54, 9, 85, 43, 50, 98, 53, 48, 101, 10},  

{183850, 9, 55, 57, 53, 56, 9, 85, 43, 50, 99, 101, 50, 97, 10},  

{19616, 9, 55, 57, 54, 48, 9, 85, 43, 48, 52, 99, 97, 48, 10},  

{183711, 9, 55, 57, 54, 49, 9, 85, 43, 50, 99, 100, 57, 102, 10},  

{183712, 9, 55, 57, 54, 51, 9, 85, 43, 50, 99, 100, 97, 48, 10},  

{17898, 9, 55, 57, 55, 48, 9, 85, 43, 48, 52, 53, 101, 97, 10},  

{183720, 9, 55, 57, 55, 52, 9, 85, 43, 50, 99, 100, 97, 56, 10},  

{156193, 9, 55, 57, 55, 57, 9, 85, 43, 50, 54, 50, 50, 49, 10},  

{176440, 9, 55, 57, 56, 48, 9, 85, 43, 50, 98, 49, 51, 56, 10},  

{182209, 9, 55, 57, 56, 55, 9, 85, 43, 50, 99, 55, 99, 49, 10},

```

```
{153 000, 9, 55, 57, 57, 54, 9, 85, 43, 50, 53, 53, 97, 56, 10},  
{17908, 9, 56, 48, 48, 48, 9, 85, 43, 48, 52, 53, 102, 52, 10},  
{19618, 9, 56, 48, 49, 52, 9, 85, 43, 48, 52, 99, 97, 50, 10},  
{156813, 9, 56, 48, 50, 48, 9, 85, 43, 50, 54, 52, 56, 100, 10},  
{17302, 9, 56, 48, 50, 52, 9, 85, 43, 48, 52, 51, 57, 54, 10},  
{19737, 9, 56, 48, 50, 57, 9, 85, 43, 48, 52, 100, 49, 57, 10},  
{181834, 9, 56, 48, 51, 48, 9, 85, 43, 50, 99, 54, 52, 97, 10},  
{15360, 9, 56, 48, 51, 50, 9, 85, 43, 48, 51, 99, 48, 48, 10},  
{183725, 9, 56, 48, 52, 50, 9, 85, 43, 50, 99, 100, 97, 100, 10},  
{171916, 9, 56, 48, 52, 51, 9, 85, 43, 50, 57, 102, 56, 99, 10},  
{165856, 9, 56, 48, 53, 55, 9, 85, 43, 50, 56, 55, 101, 48, 10},  
{183726, 9, 56, 48, 54, 50, 9, 85, 43, 50, 99, 100, 97, 101, 10},  
{166729, 9, 56, 48, 54, 51, 9, 85, 43, 50, 56, 98, 52, 57, 10},  
{15884, 9, 56, 48, 54, 52, 9, 85, 43, 48, 51, 101, 48, 99, 10},  
{181835, 9, 56, 48, 55, 51, 9, 85, 43, 50, 99, 54, 52, 98, 10},  
{150217, 9, 56, 48, 55, 53, 9, 85, 43, 50, 52, 97, 99, 57, 10},  
{183955, 9, 56, 48, 55, 57, 9, 85, 43, 50, 99, 101, 57, 51, 10},  
{177837, 9, 56, 49, 48, 48, 9, 85, 43, 50, 98, 54, 97, 100, 10}}}
```

```
Out[81]= .../transmogrify/missingTGSCCcodes.UTF32LE
```