

# 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]= {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}
```

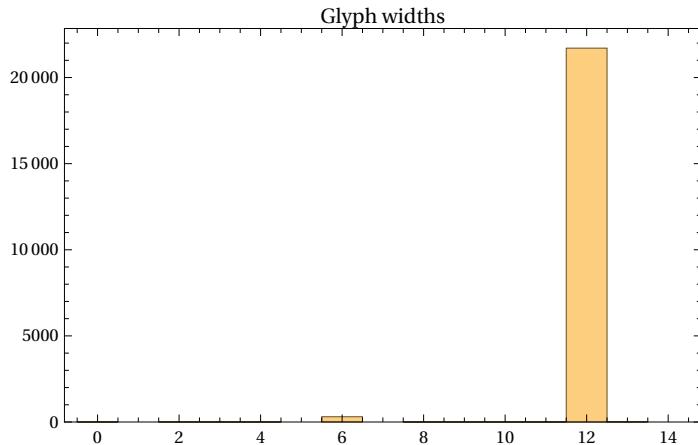
## ■ Extract character widths

```
In[15]:= Dimensions[widths = Flatten[Select[#, First[#] == "DWIDTH" &]]] & /@ 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}
```

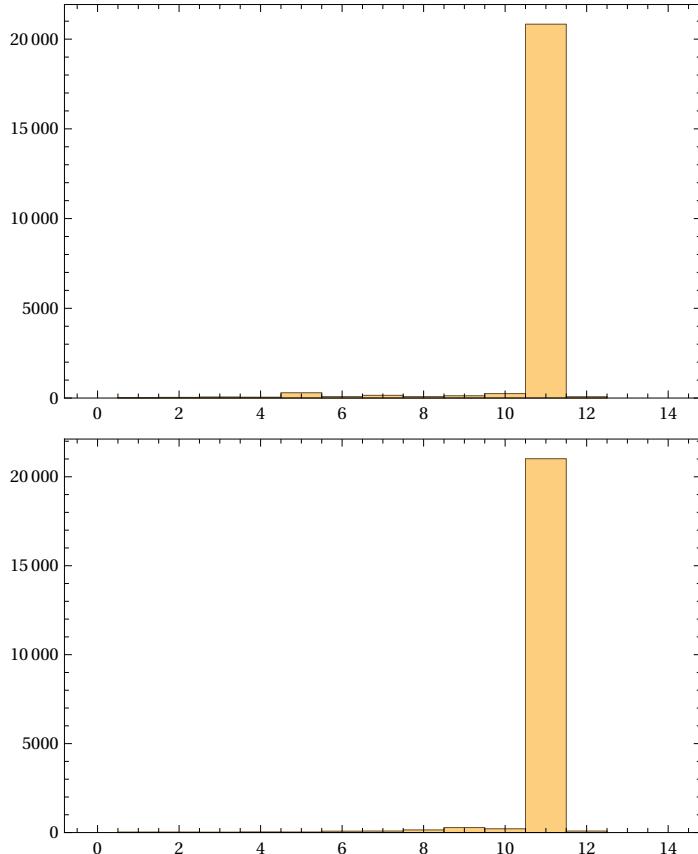
## ■ Maximum bounding box size

```
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];
```

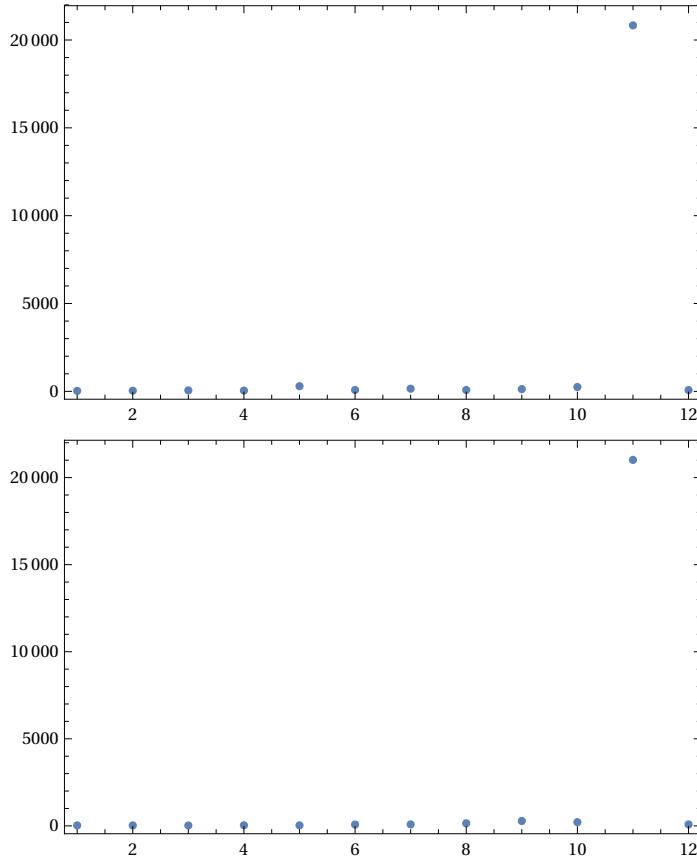


```
In[23]:= TableForm[Sort[Tally[#, #1[[1]] < #2[[1]] &] & /@ Transpose[sizes]]]
```

```
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]
```

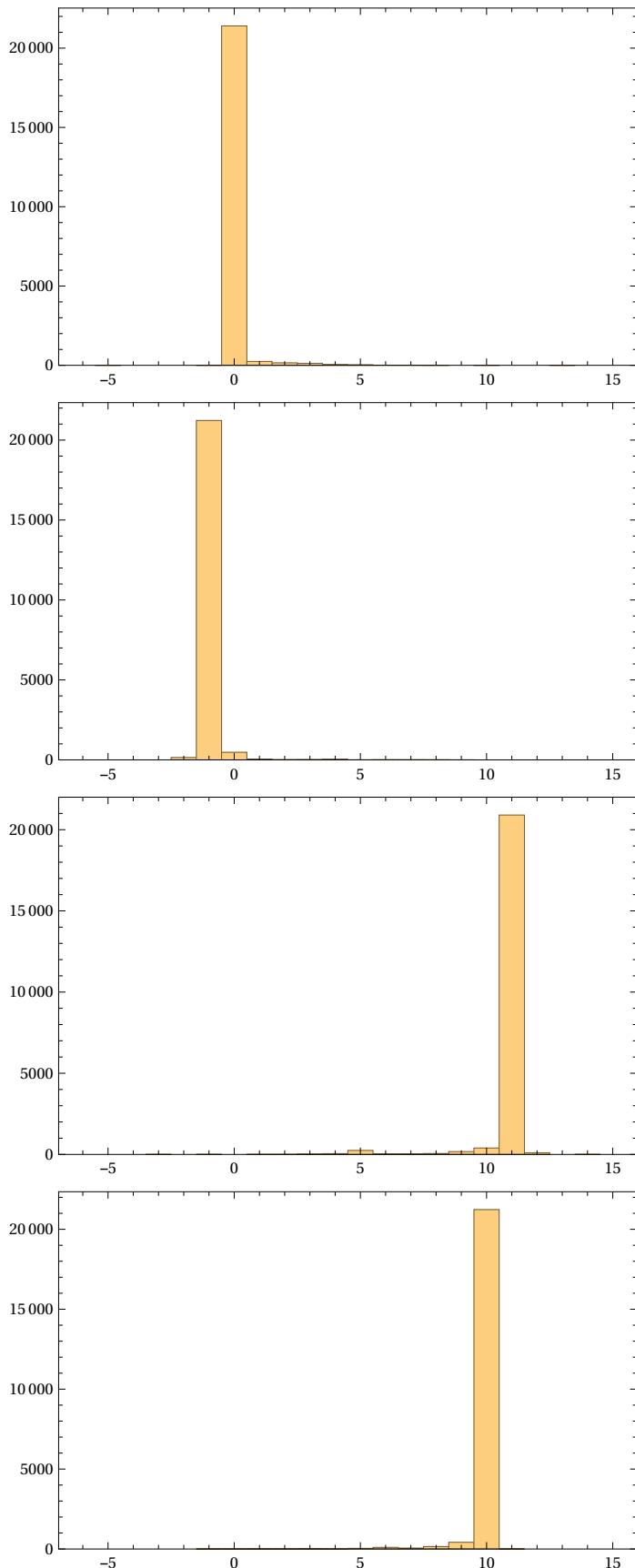
```
Out[26]= {22 043, 4}
```

#### ■ 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

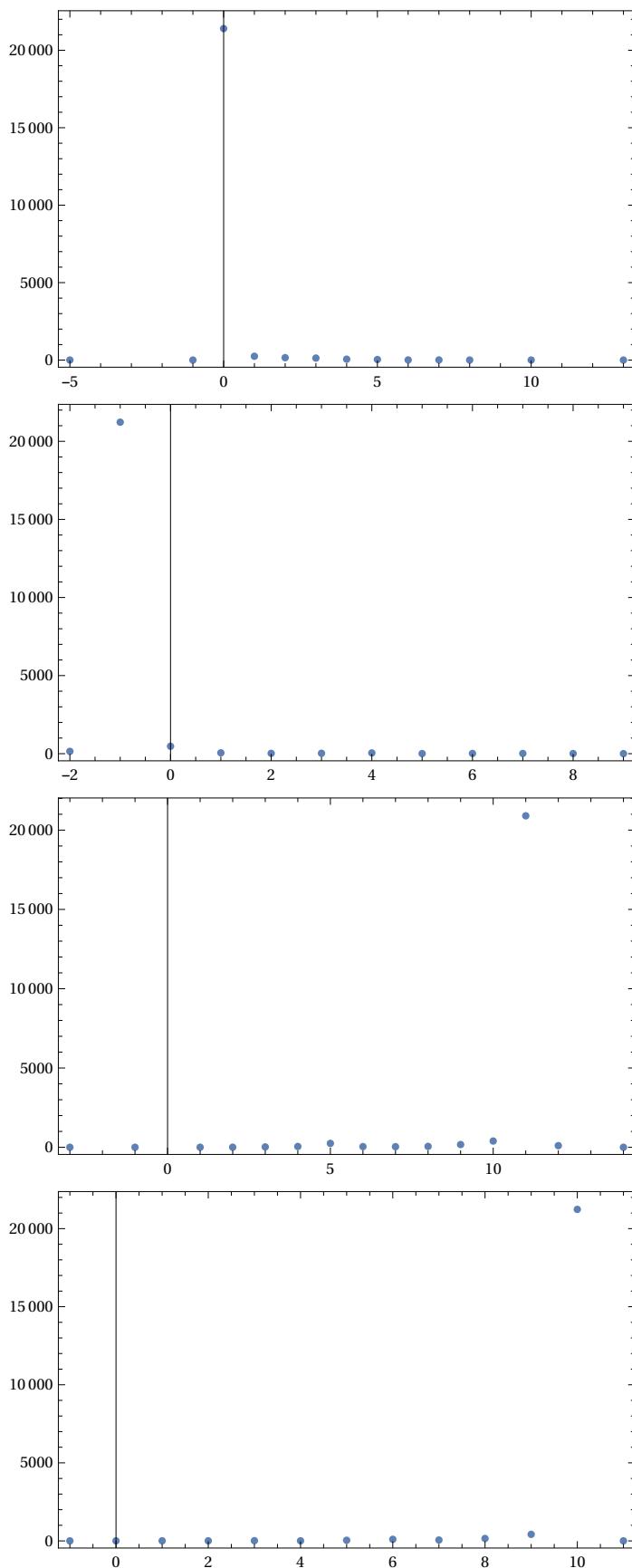
```
In[28]:= Print[Histogram[#, {- $\frac{13}{2}$ ,  $\frac{31}{2}$ , 1}, Frame → True, PlotRange → All]] & /@
  Transpose[bounds];
```



```
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}
```



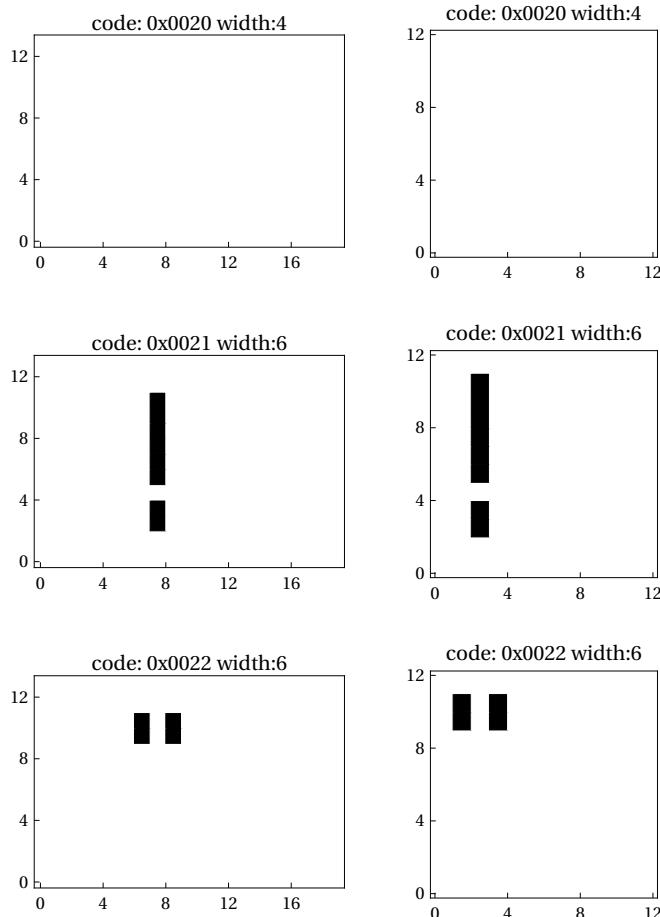
## Display glyphs

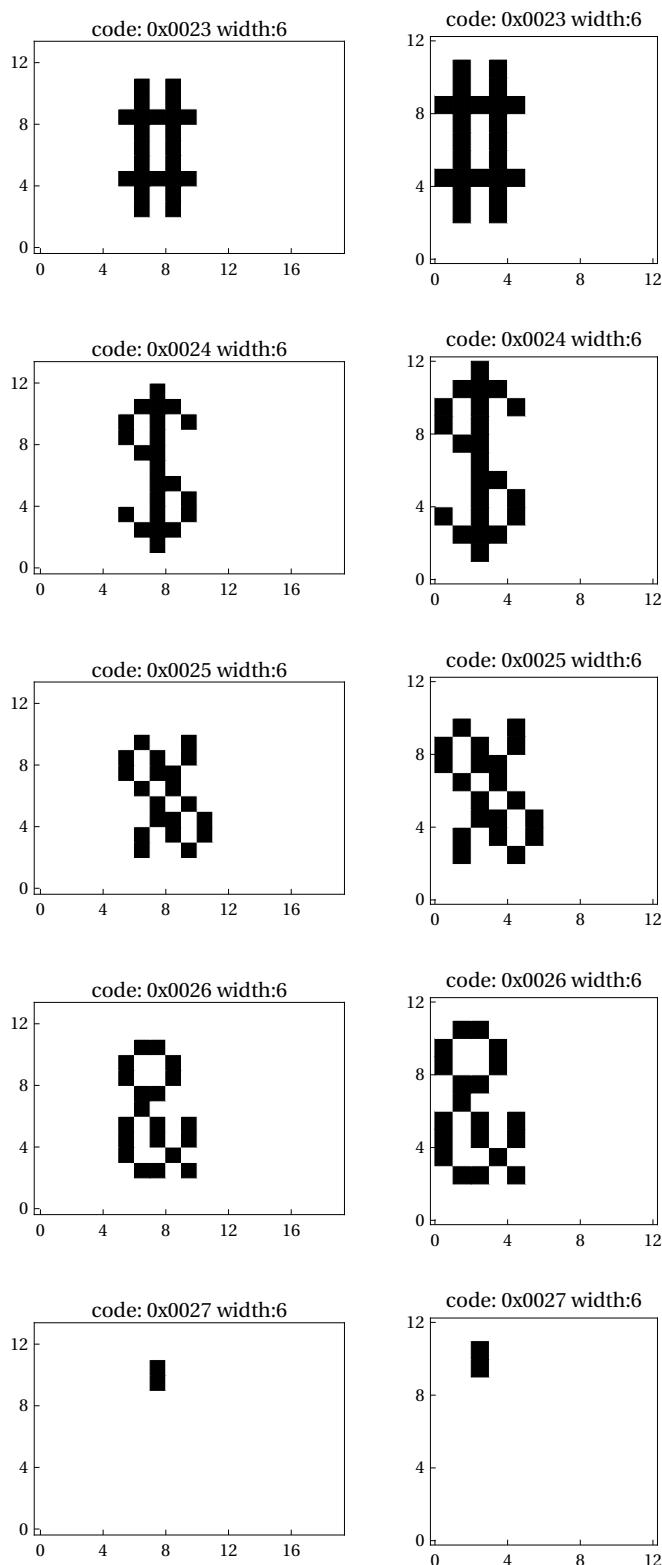
### ■ Non standard width glyphs

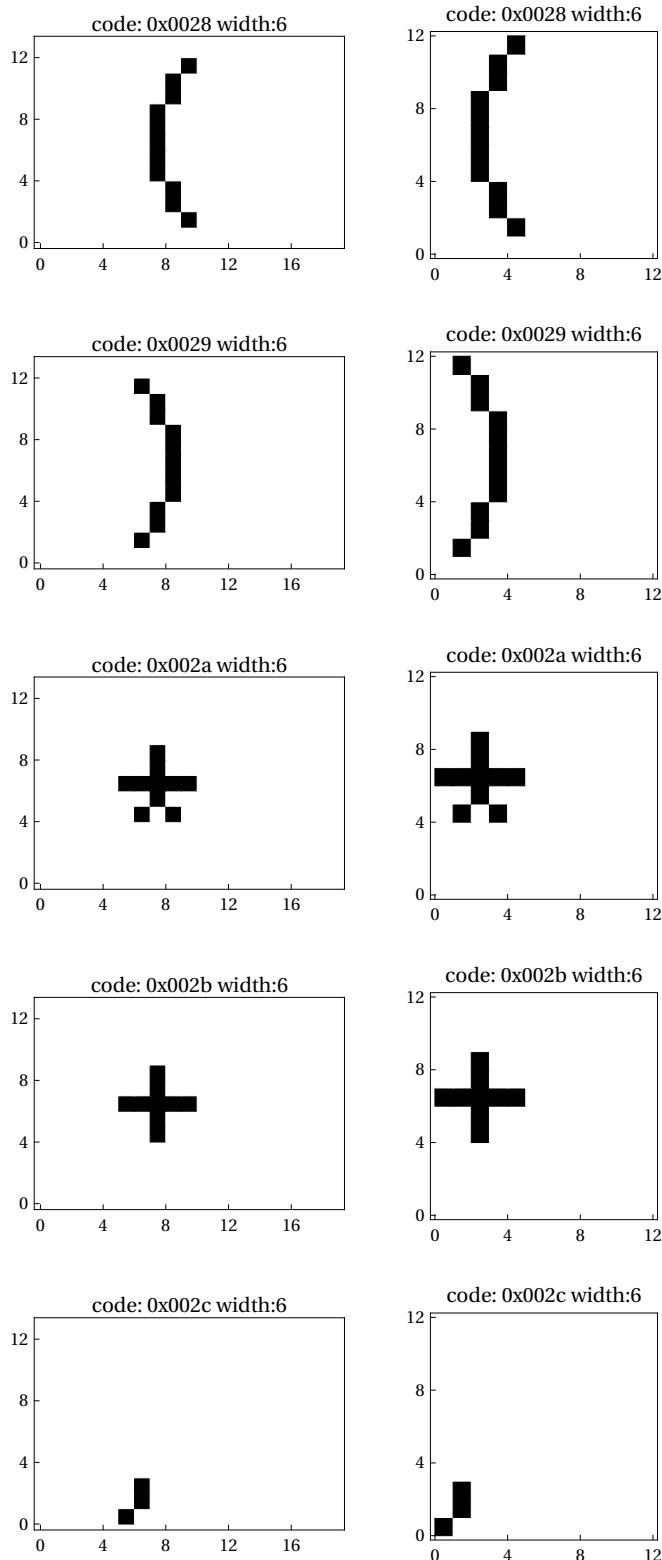
```
In[41]:= nonstdwidthgraphics =
MapThread[Graphics[Raster[Reverse[#1]], Frame → True, AspectRatio →
Divide @@ Dimensions[#1], FrameTicks → ({#, #, {}, {}} & [Range[0, 16, 4]]),
PlotLabel → ("code: " <> IntegerString[ToExpression[#2], 16, 4] <> " width:" <>
ToString[#3])] &, #[[nonstdwidth]] & /@ {extbitmaps,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

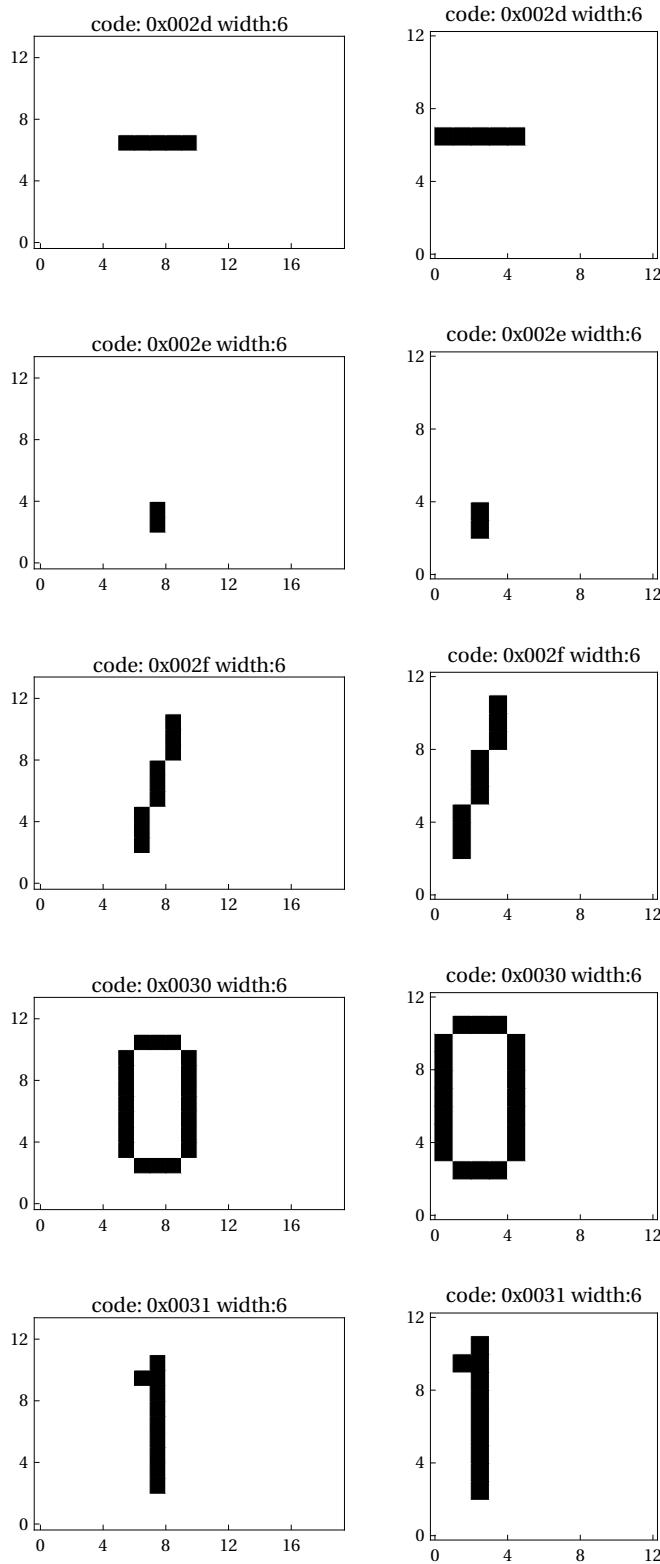
In[42]:= nonstdwidthtruncgraphics =
MapThread[Graphics[Raster[Reverse[#1]], Frame → True, AspectRatio →
Divide @@ Dimensions[#1], FrameTicks → ({#, #, {}, {}} & [Range[0, 16, 4]]),
PlotLabel → ("code: " <> IntegerString[ToExpression[#2], 16, 4] <> " width:" <>
ToString[#3])] &, #[[nonstdwidth]] & /@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

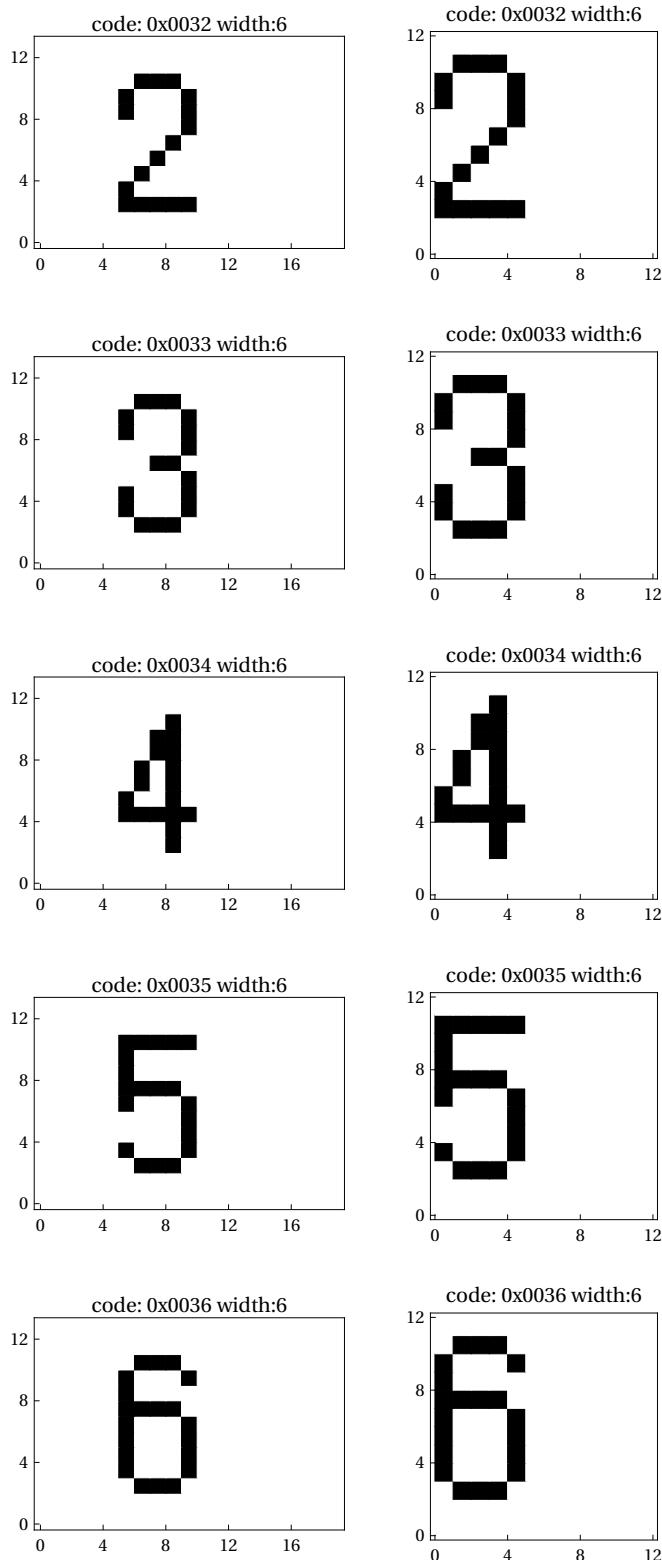
In[43]:= MapThread[Print[GraphicsGrid[{{##}}]]] &,
{nonstdwidthgraphics, nonstdwidthtruncgraphics}]
```

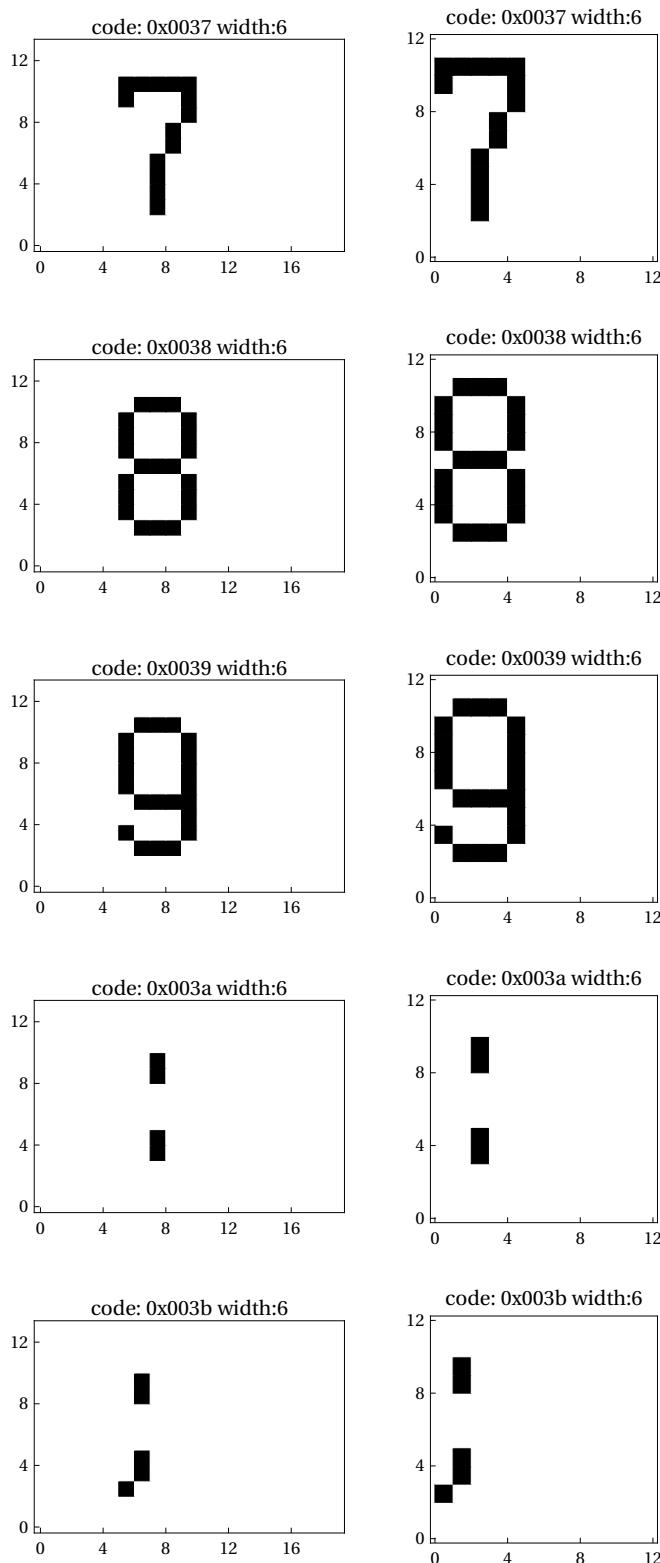


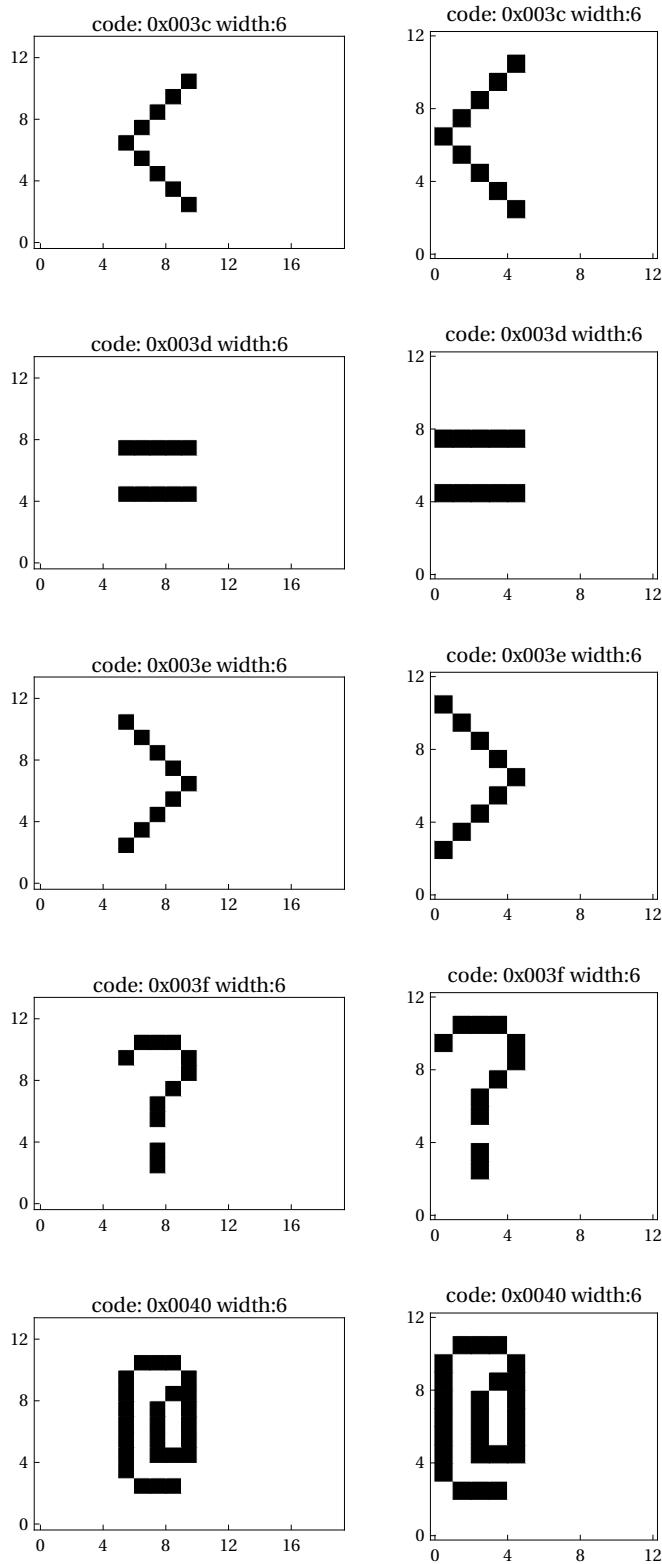


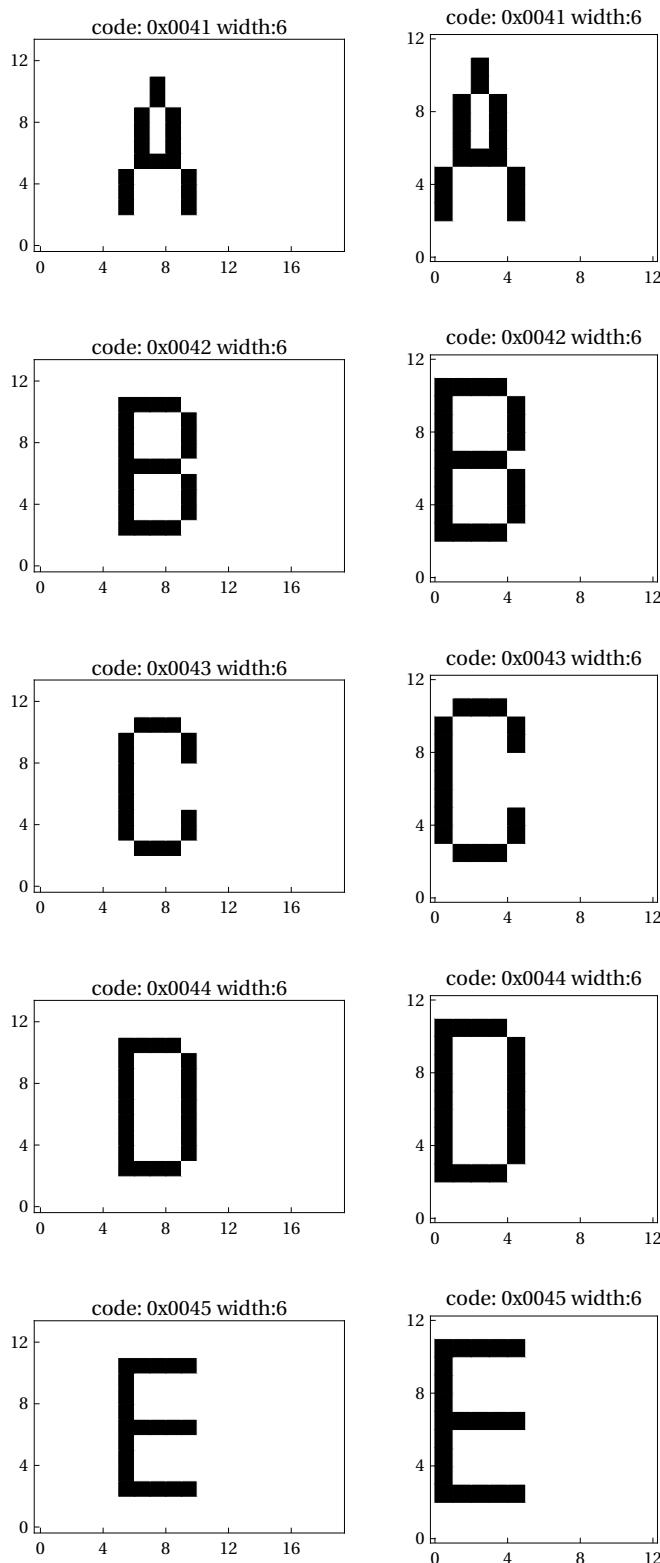


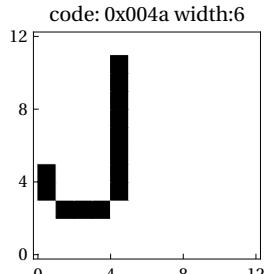
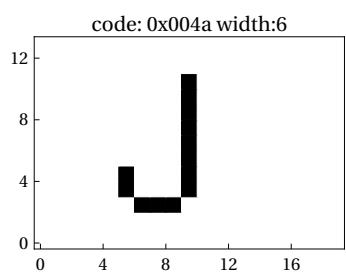
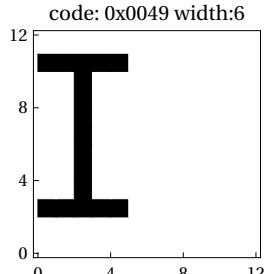
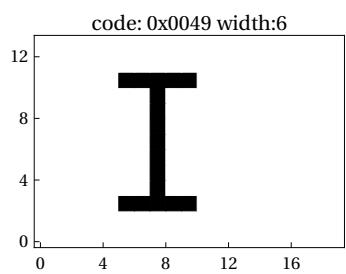
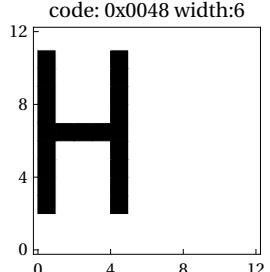
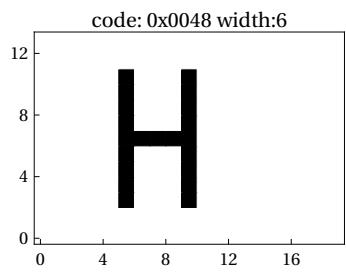
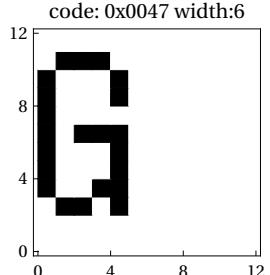
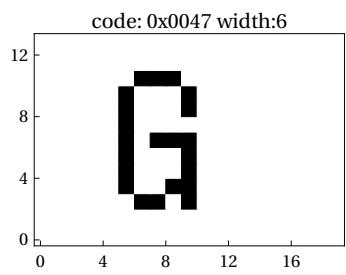
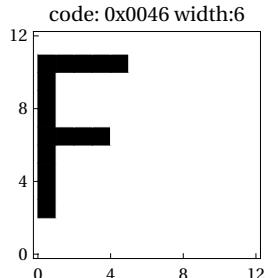
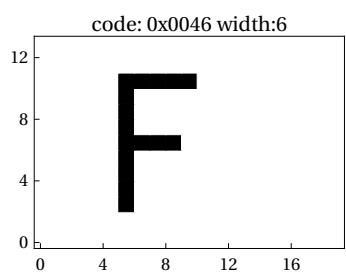


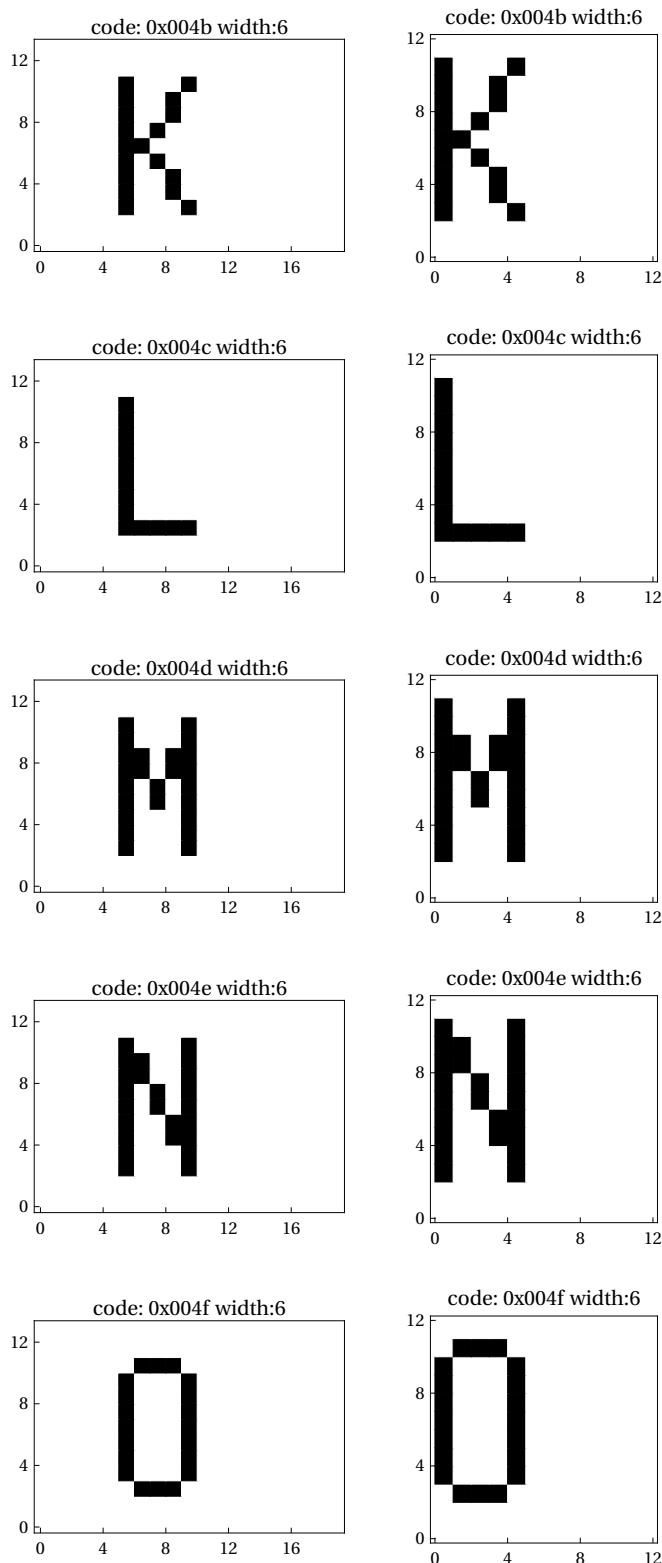


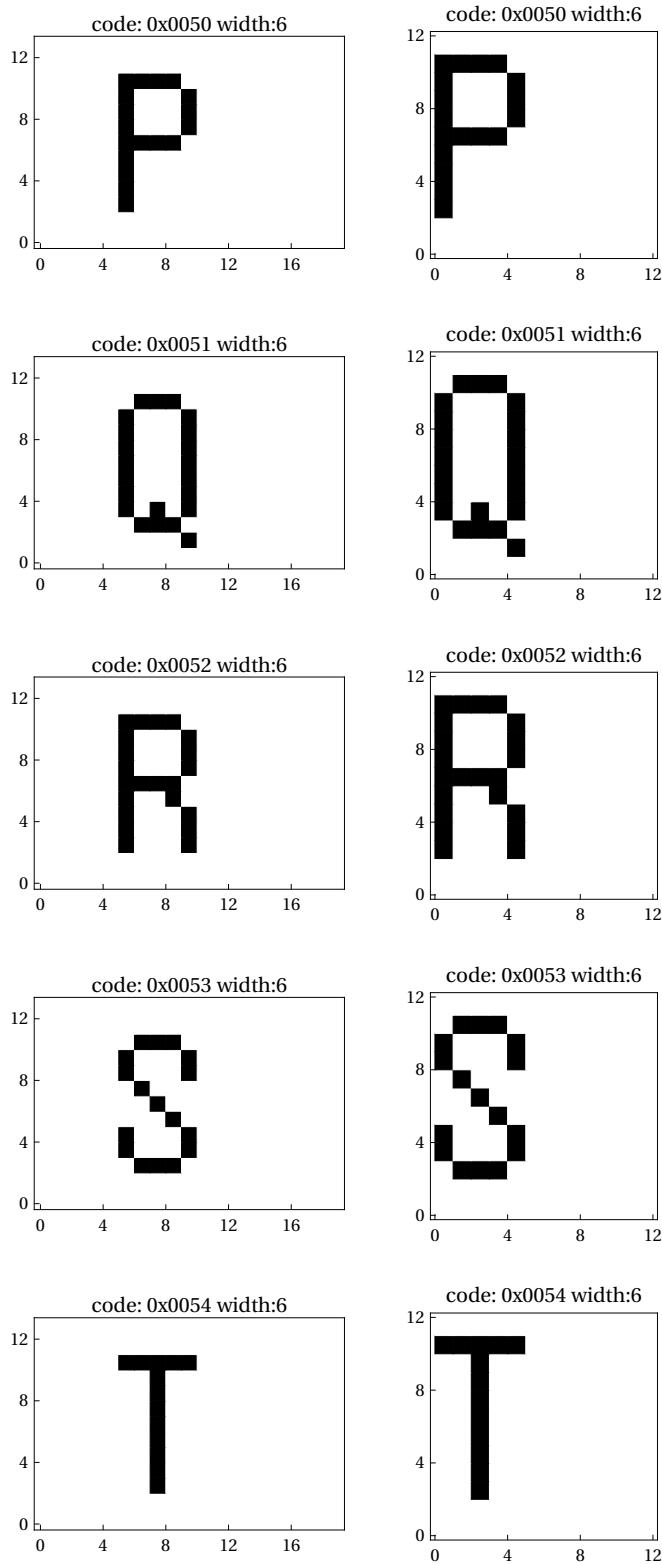


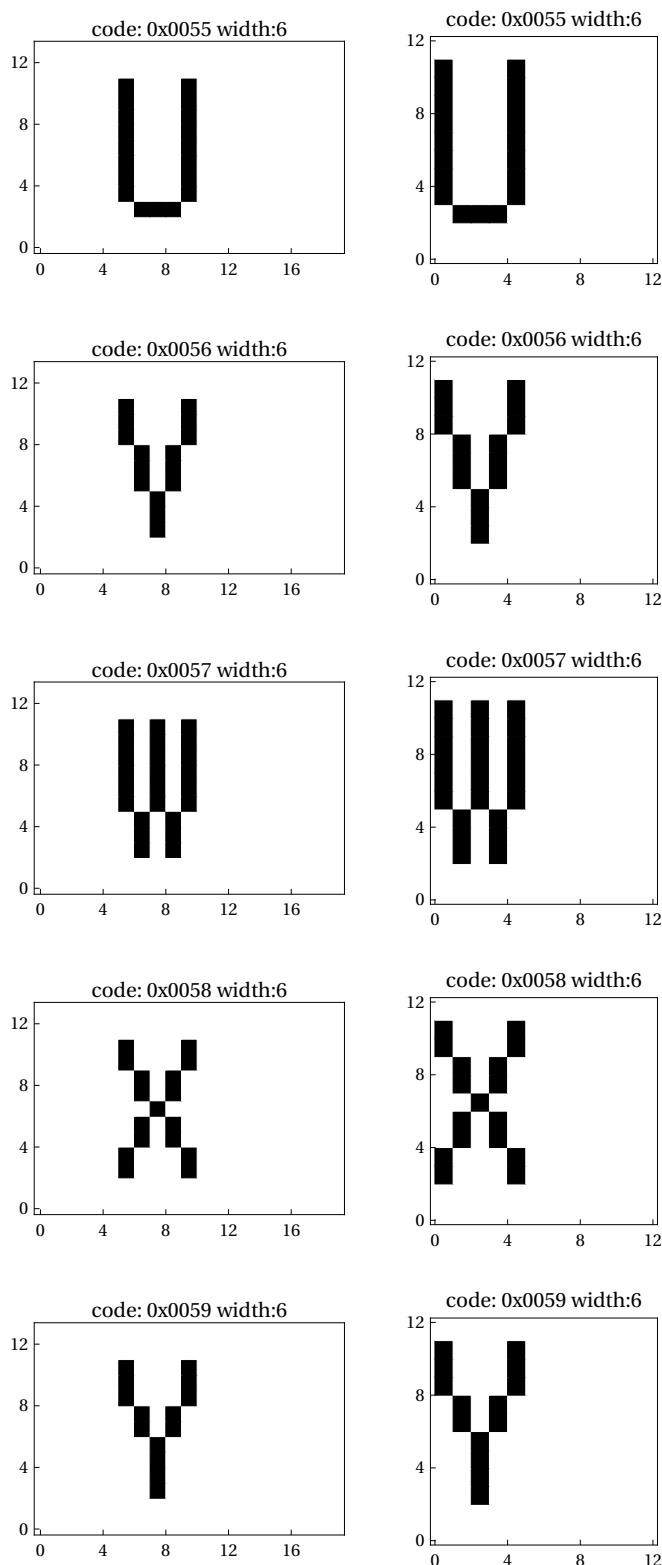


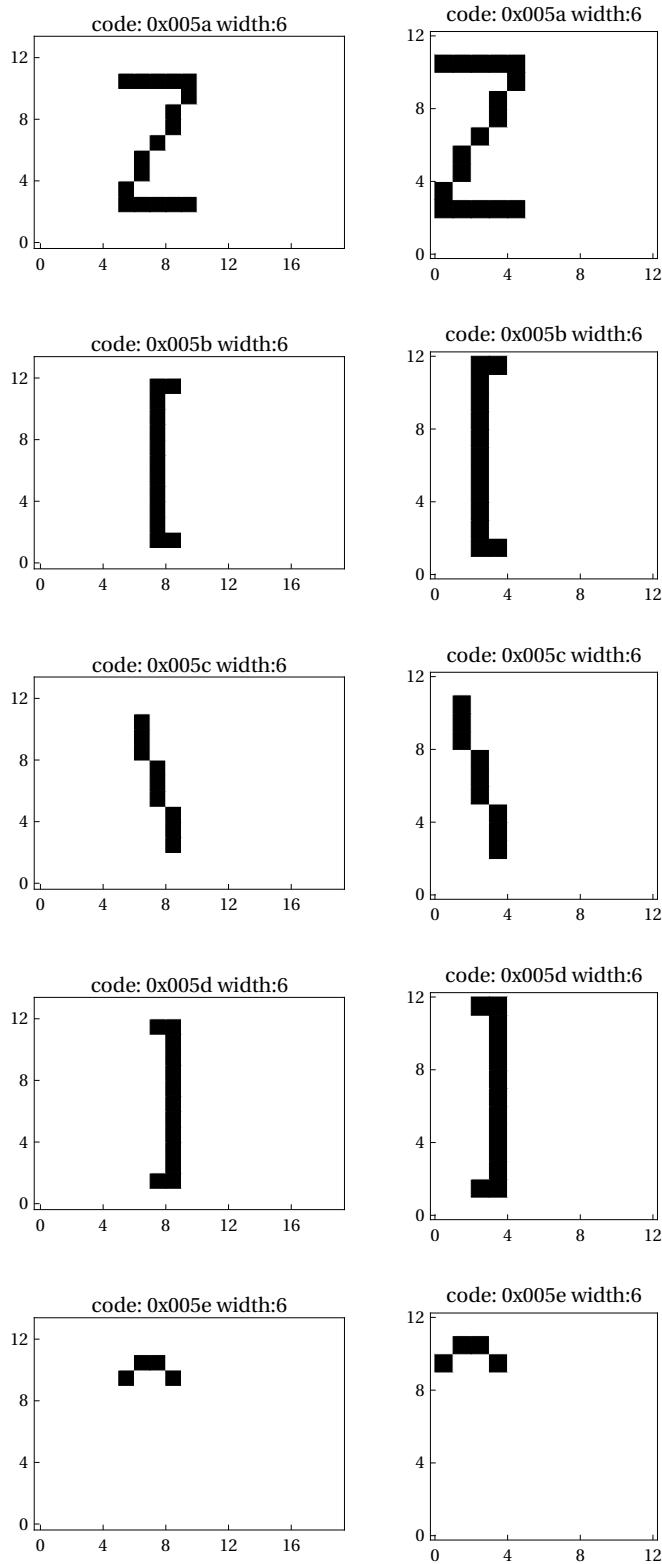


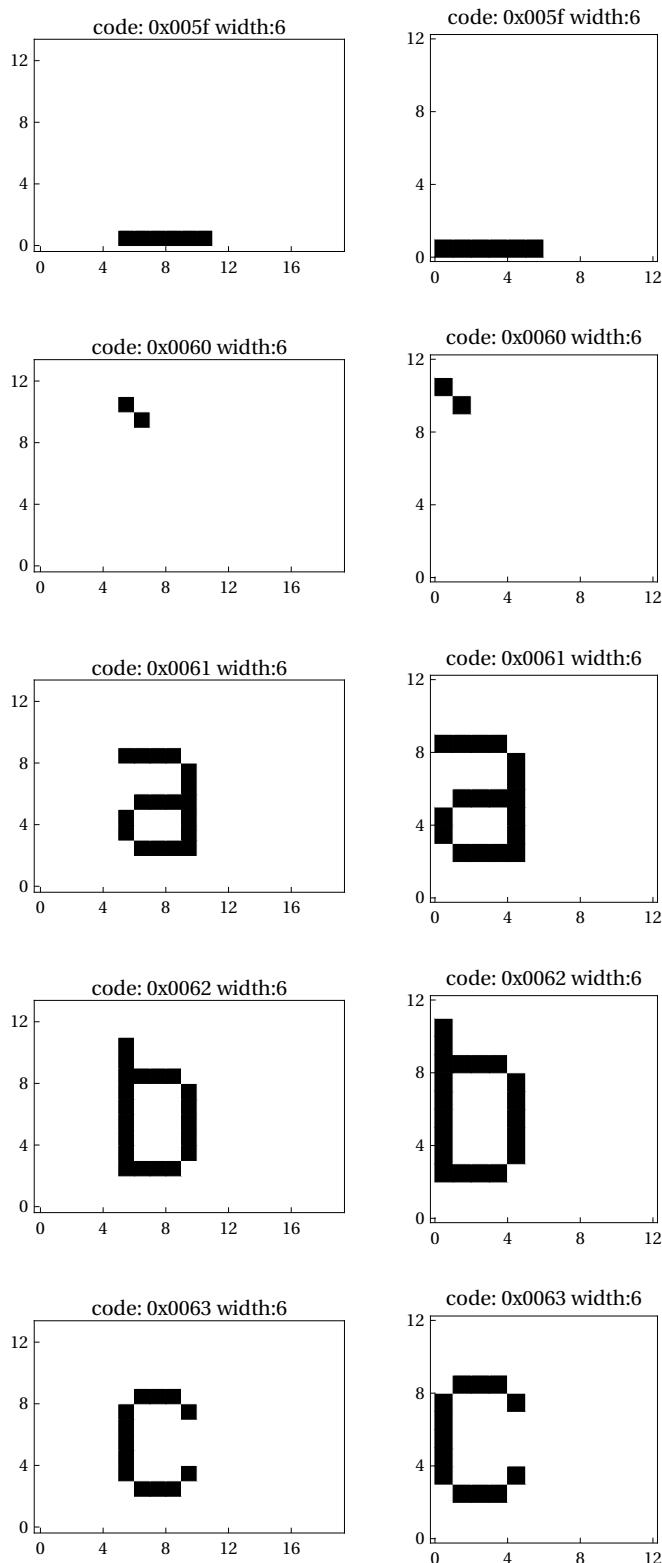


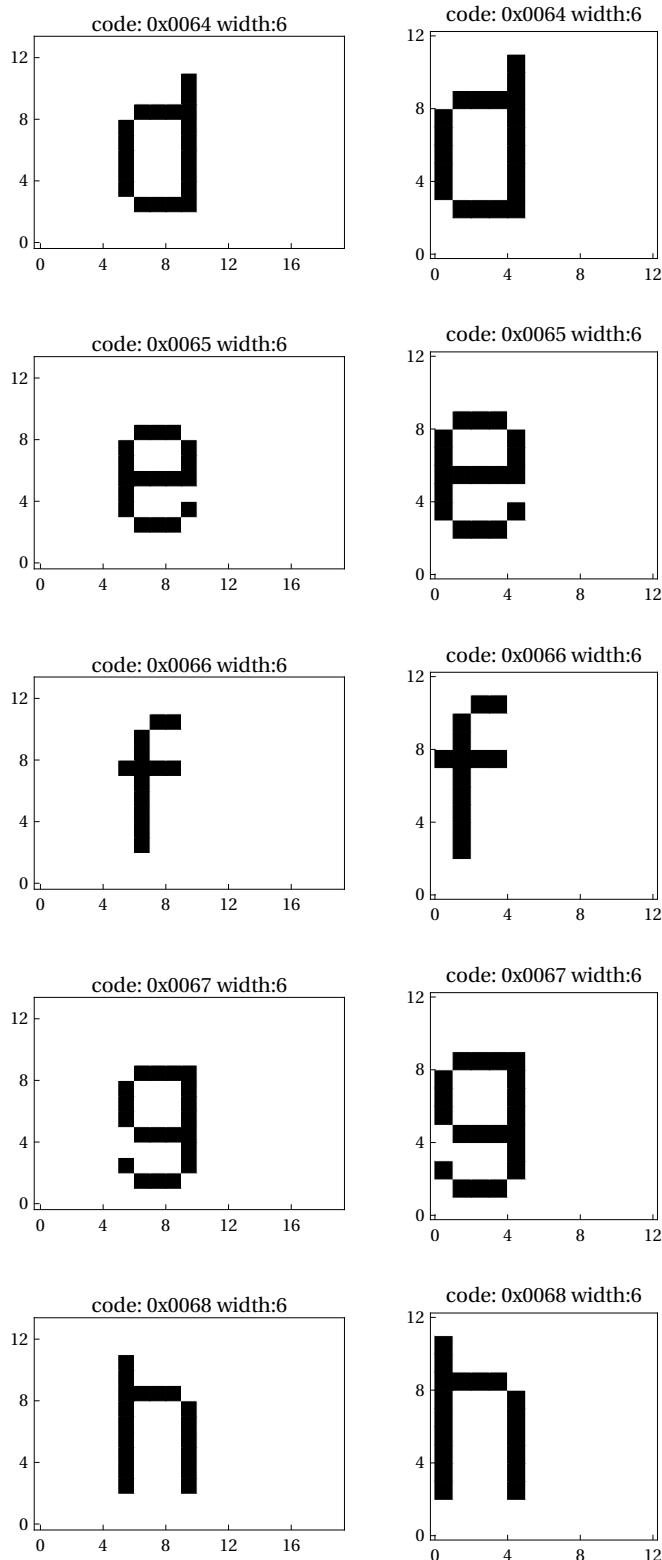


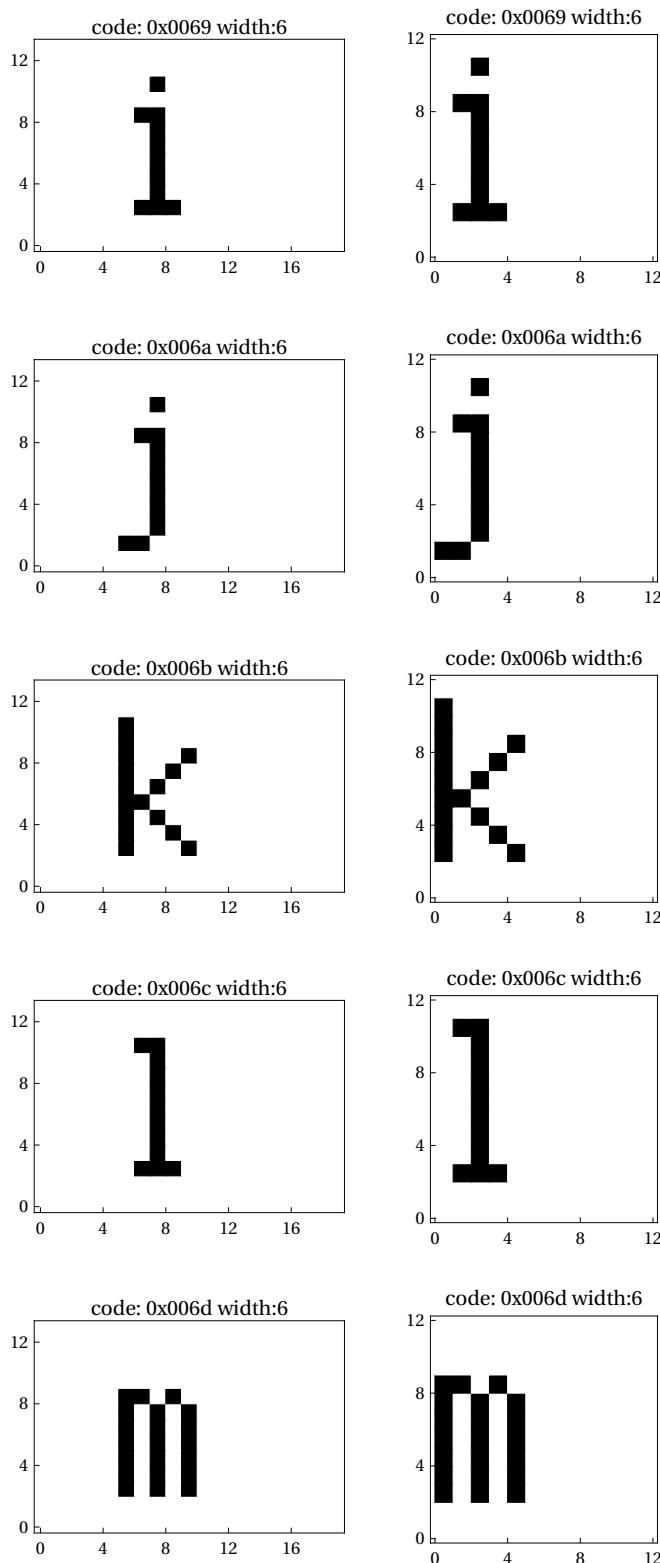


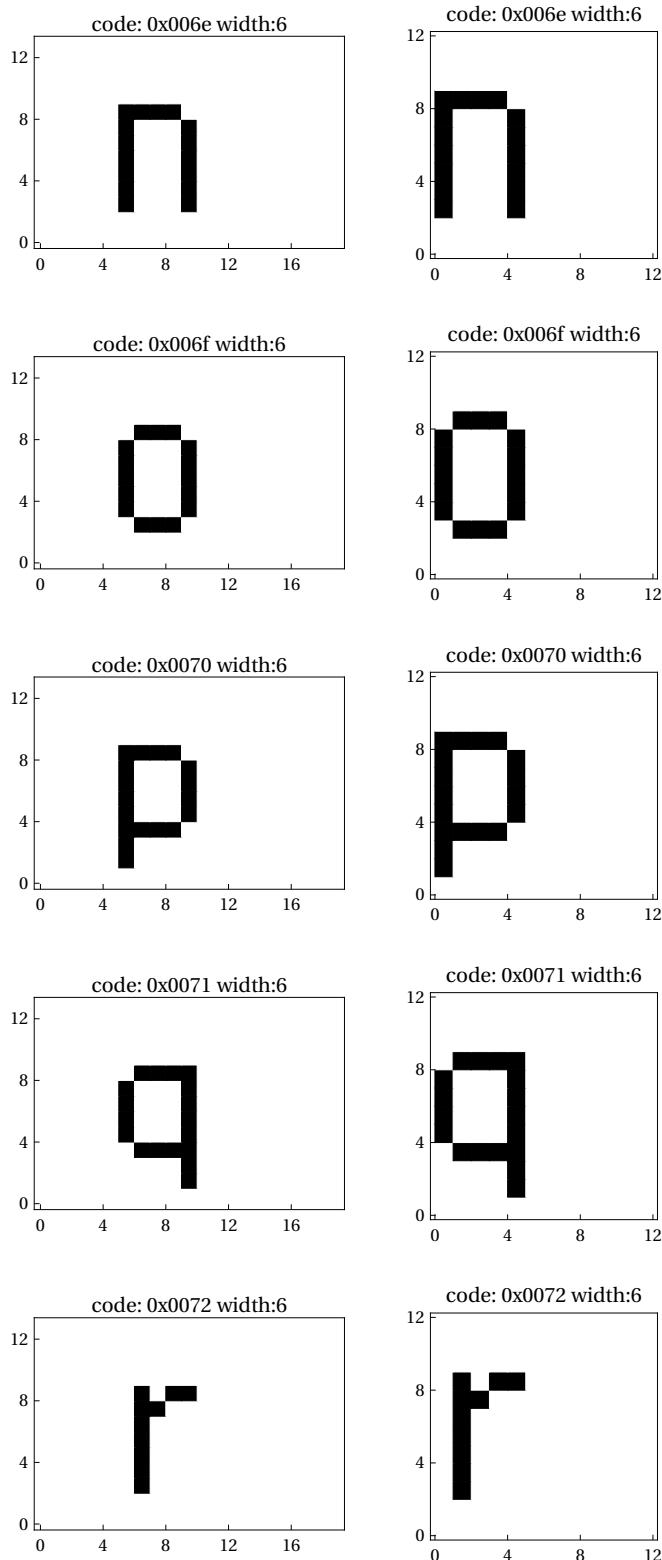


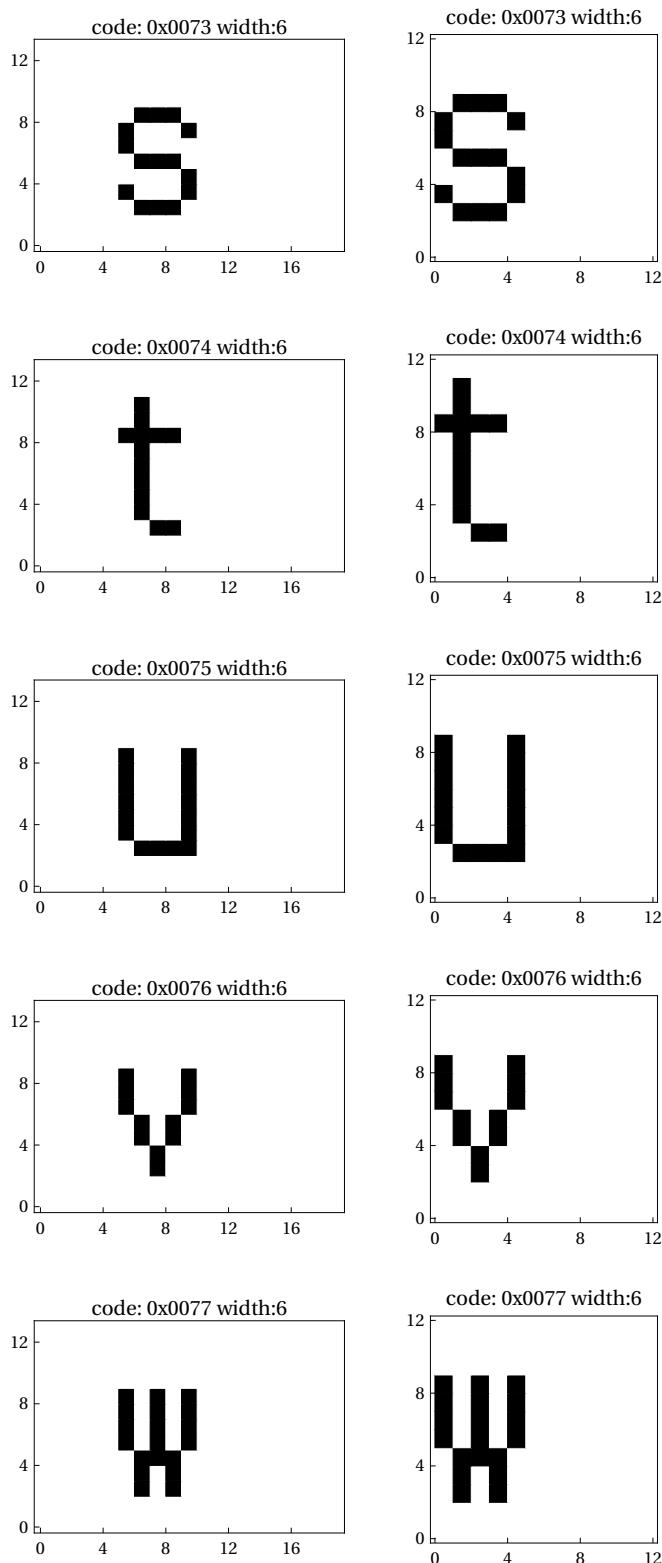


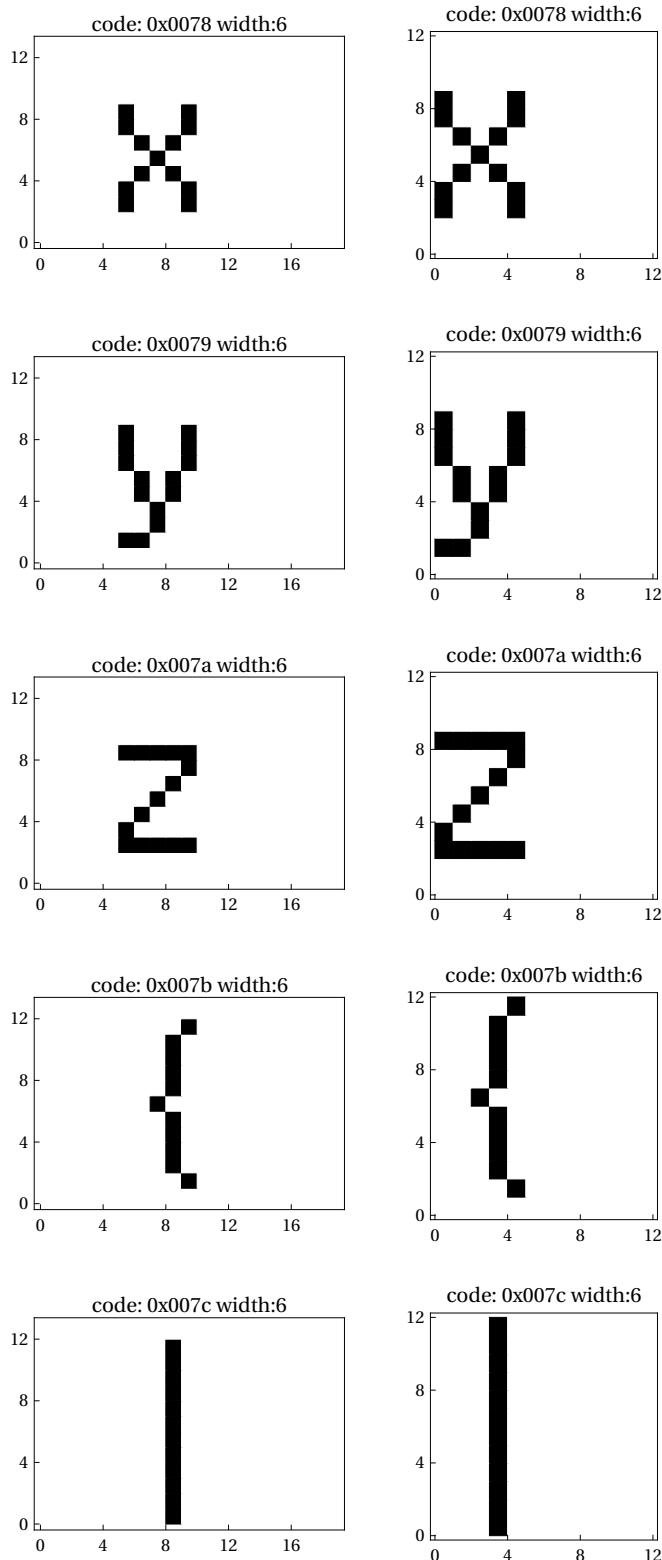


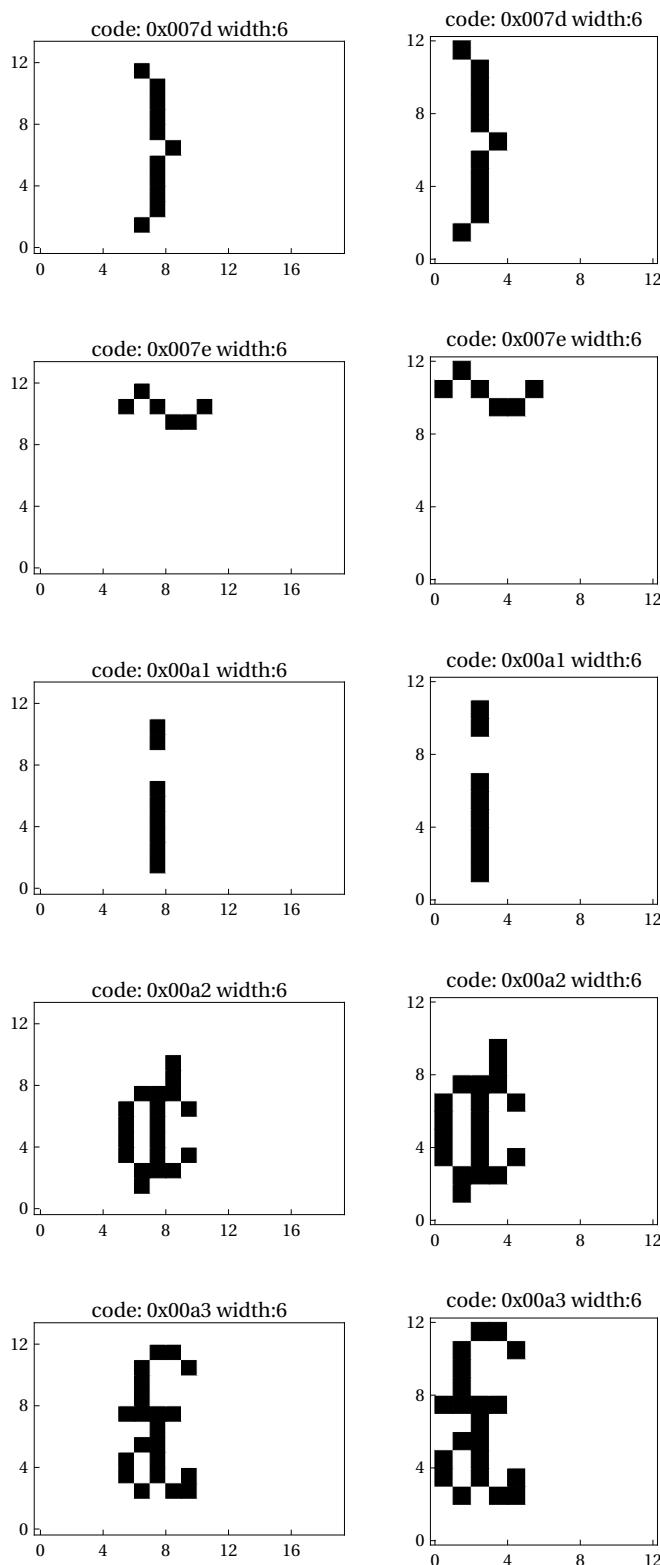


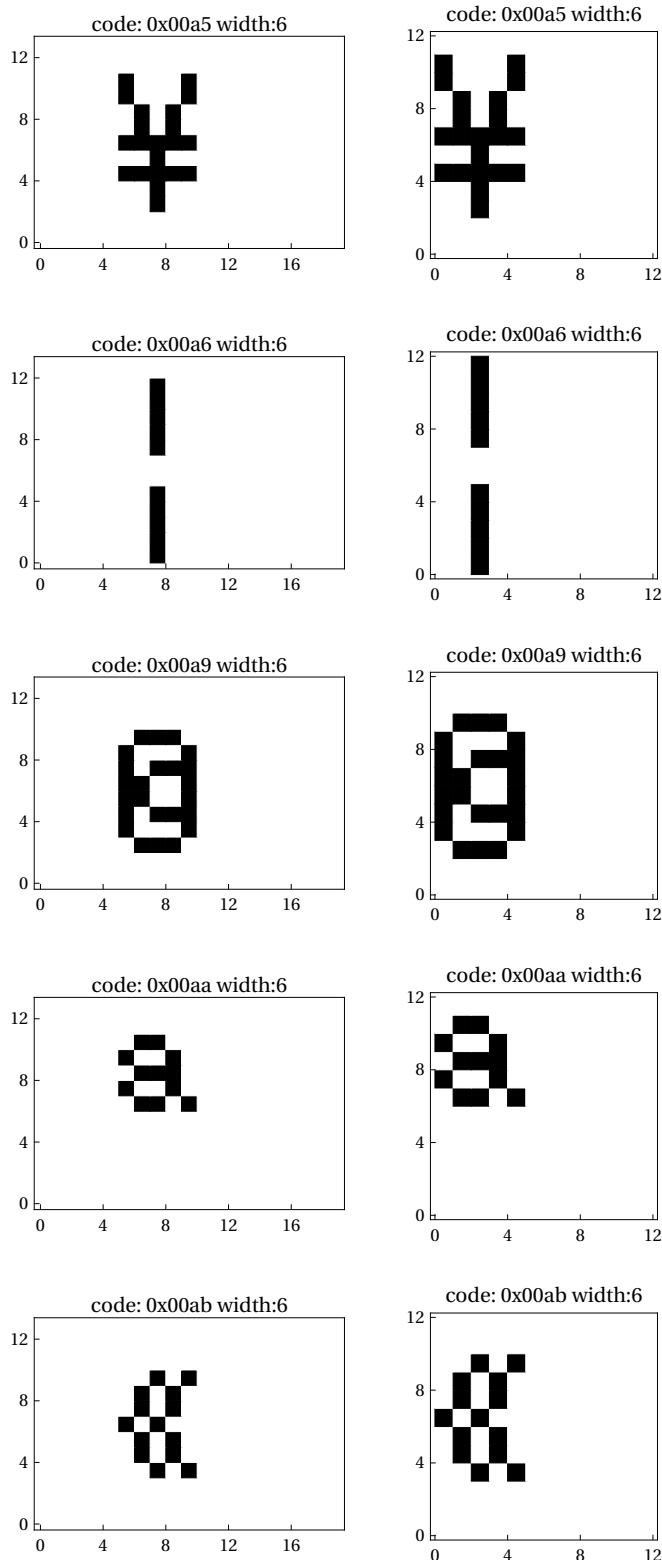


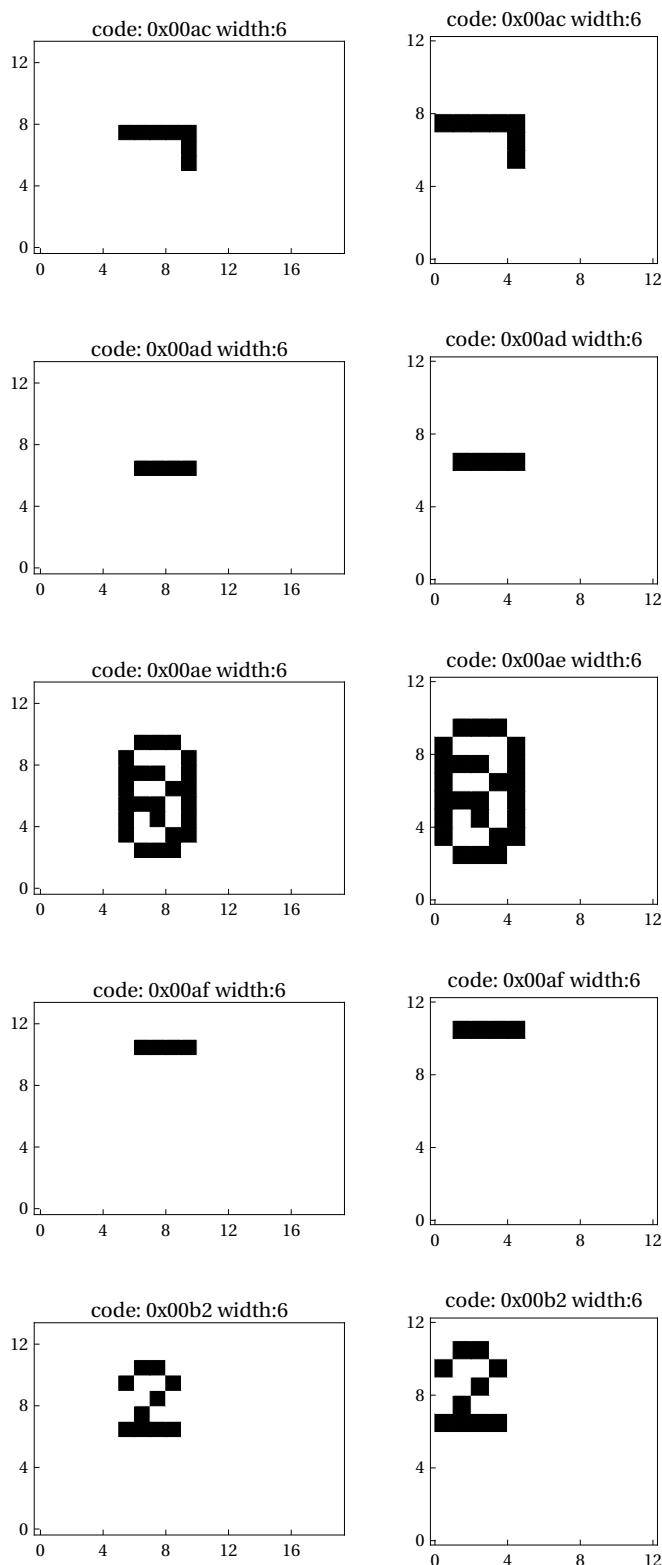


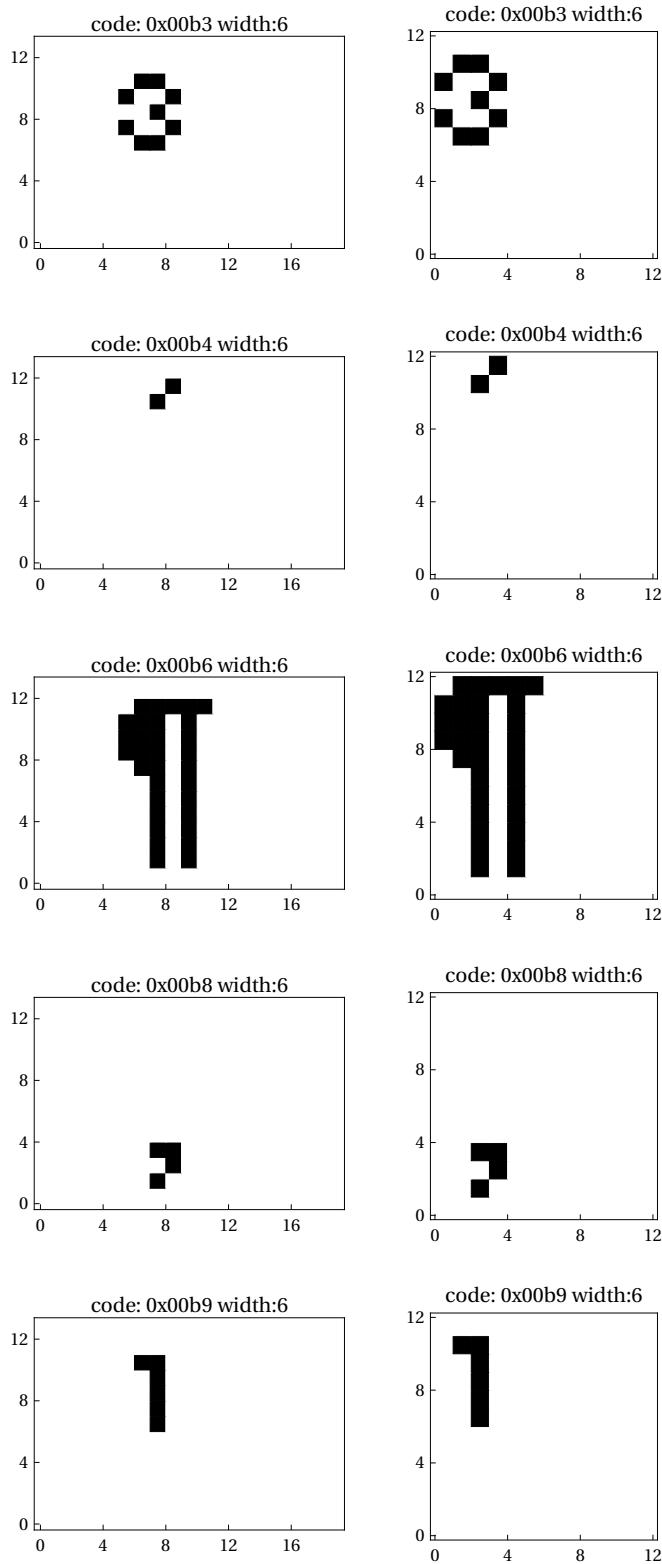


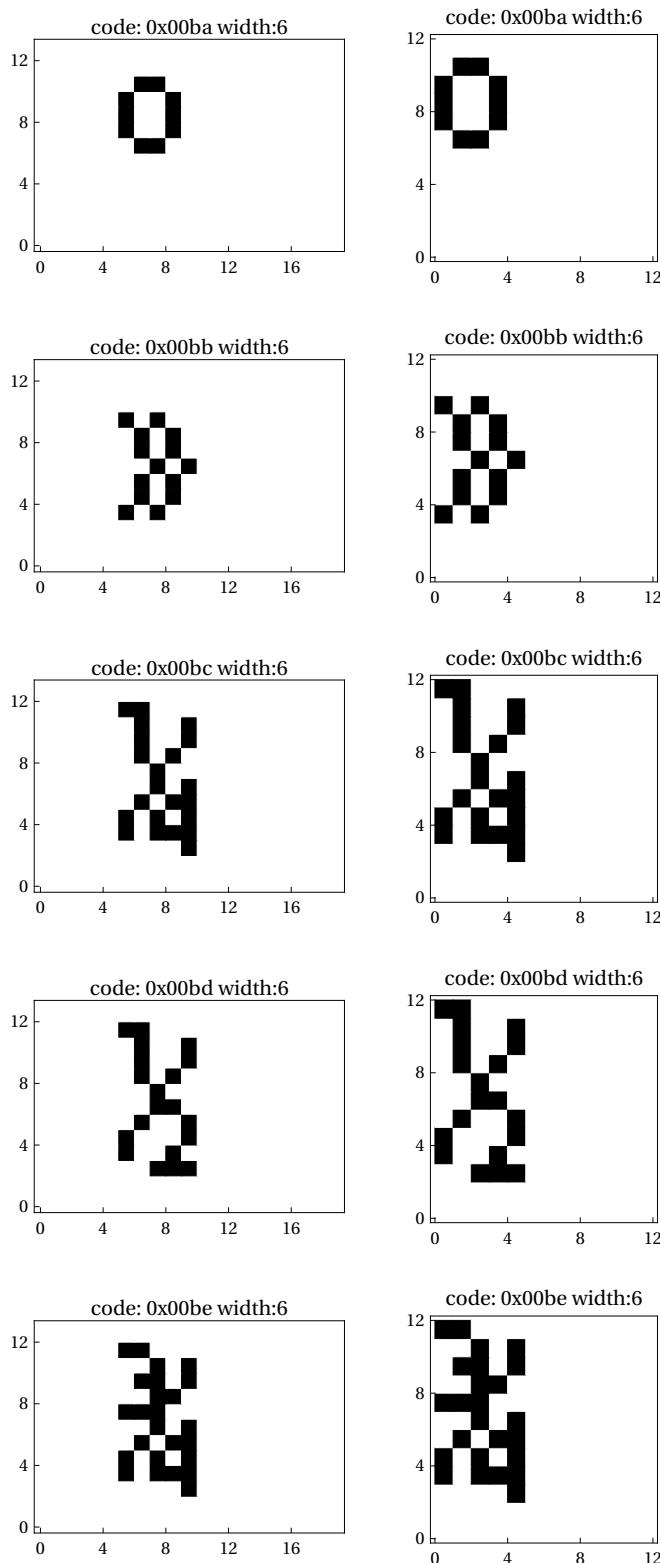


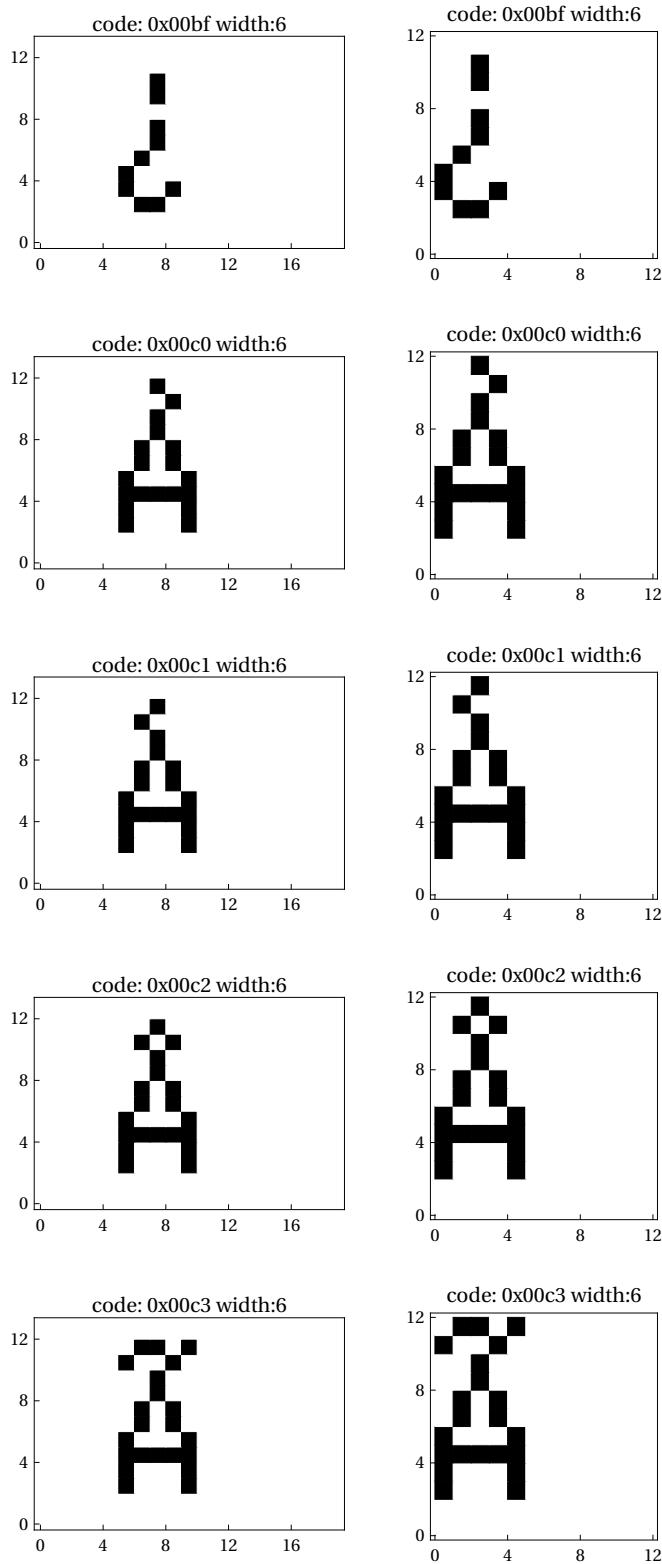


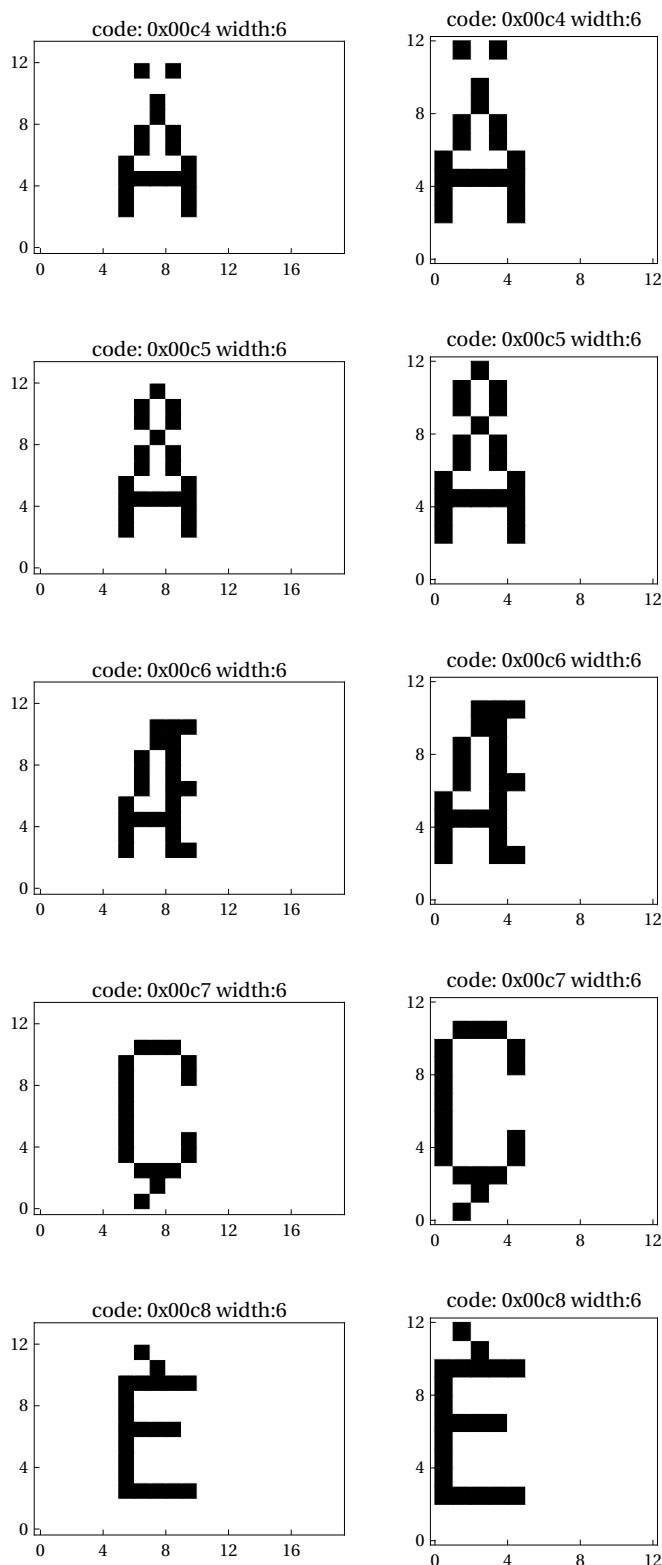


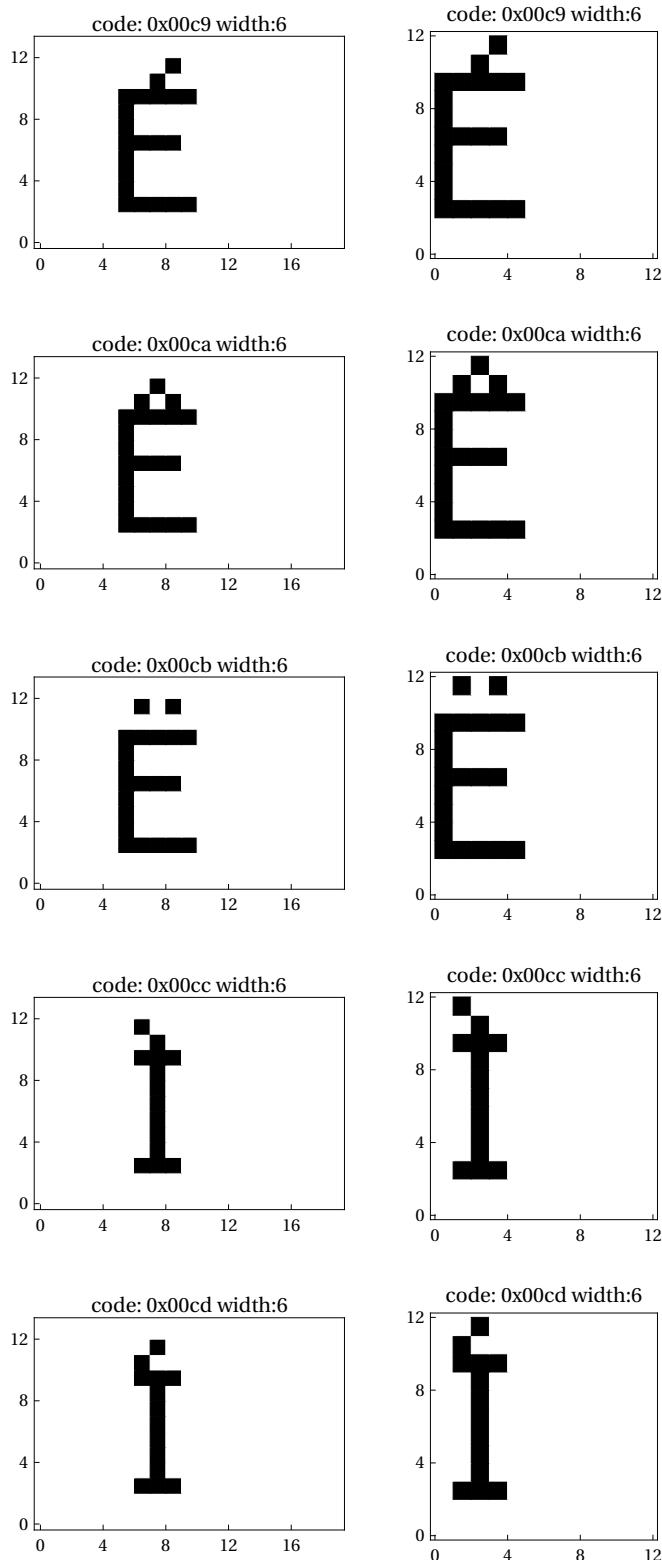


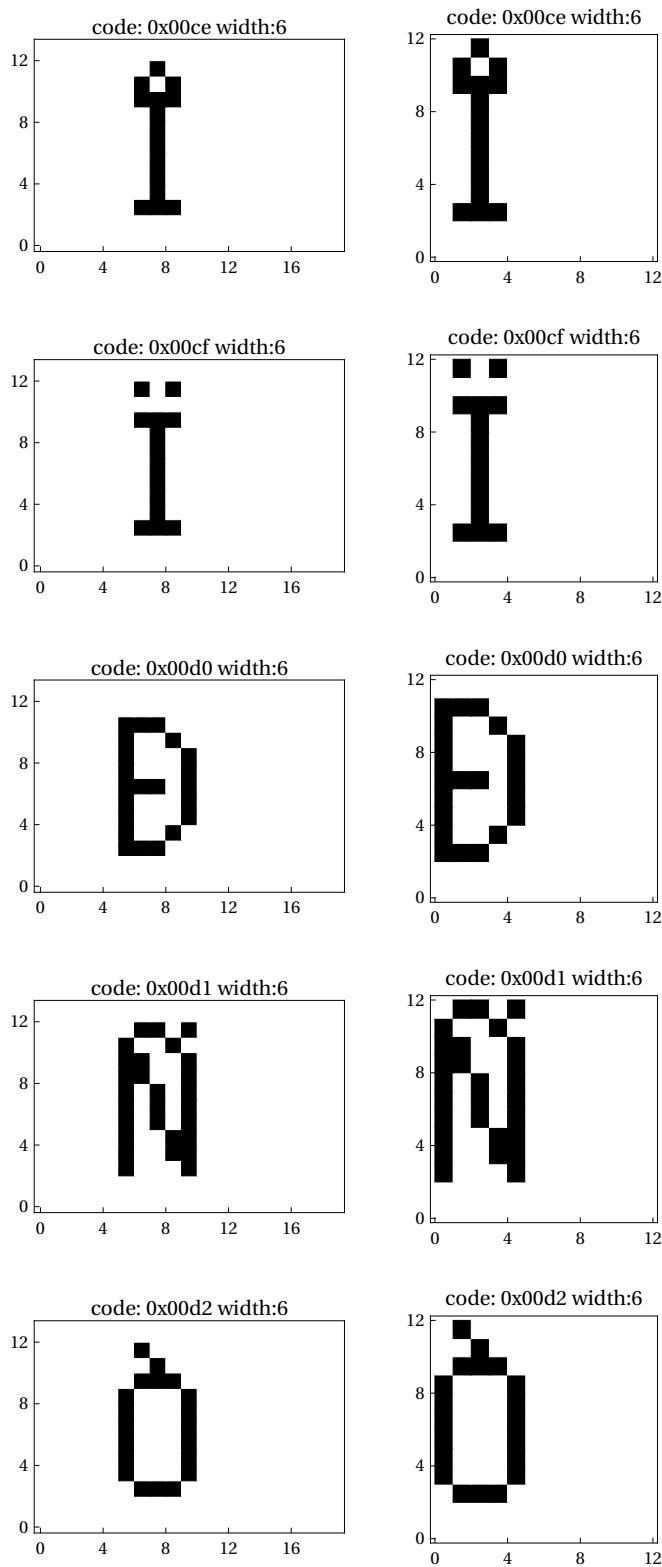


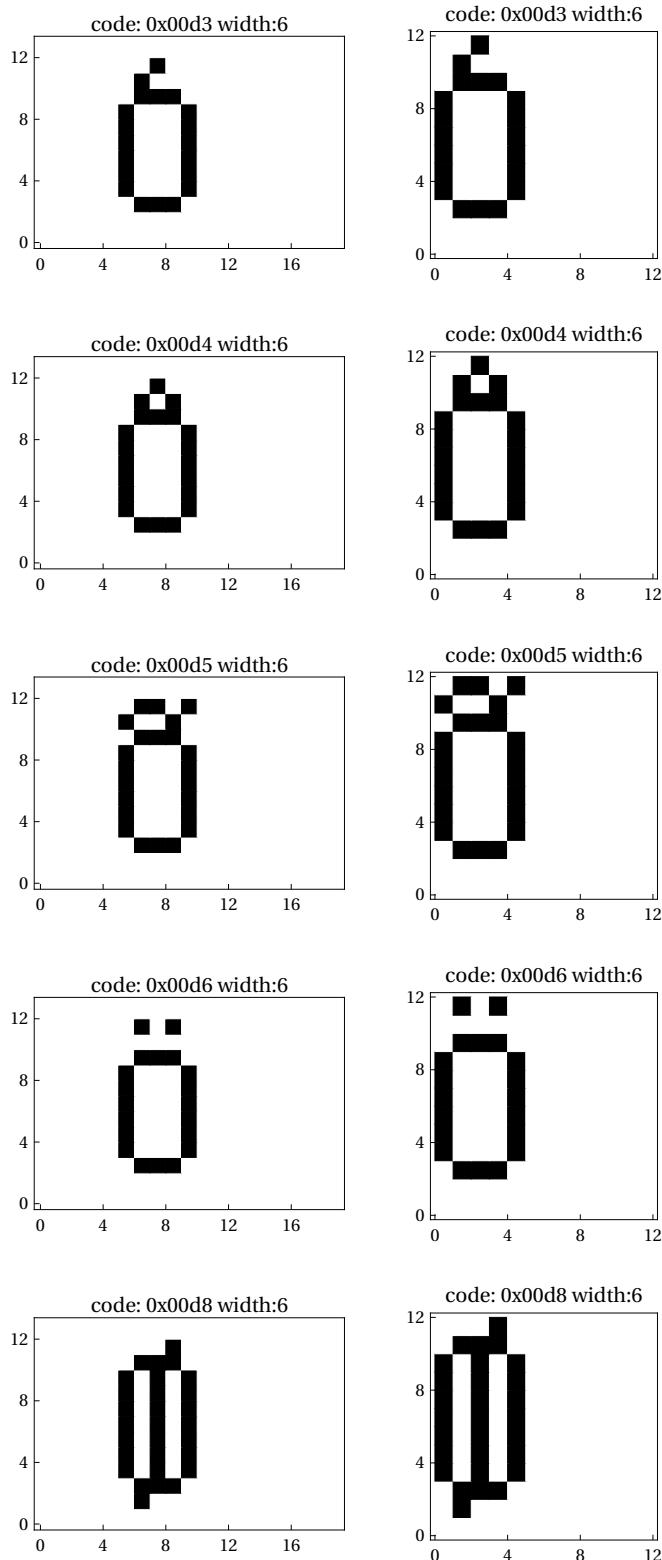


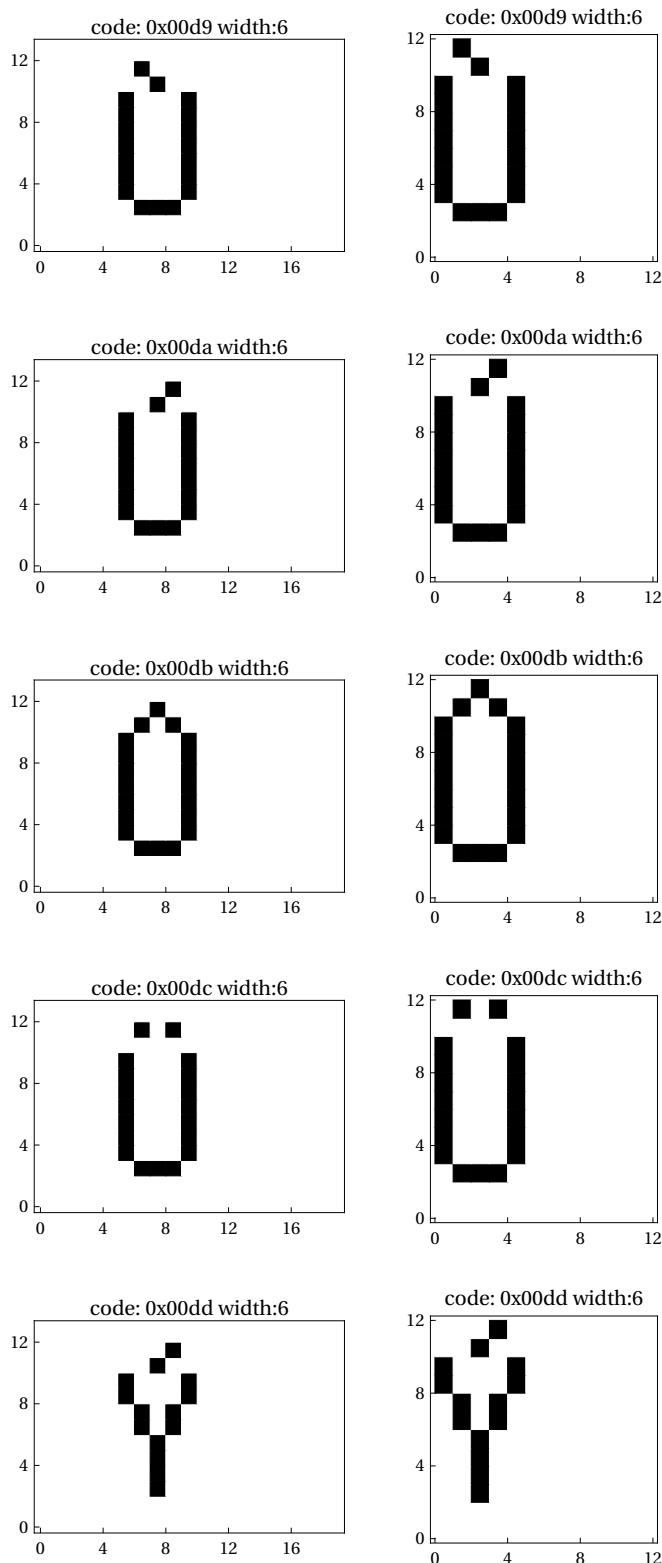


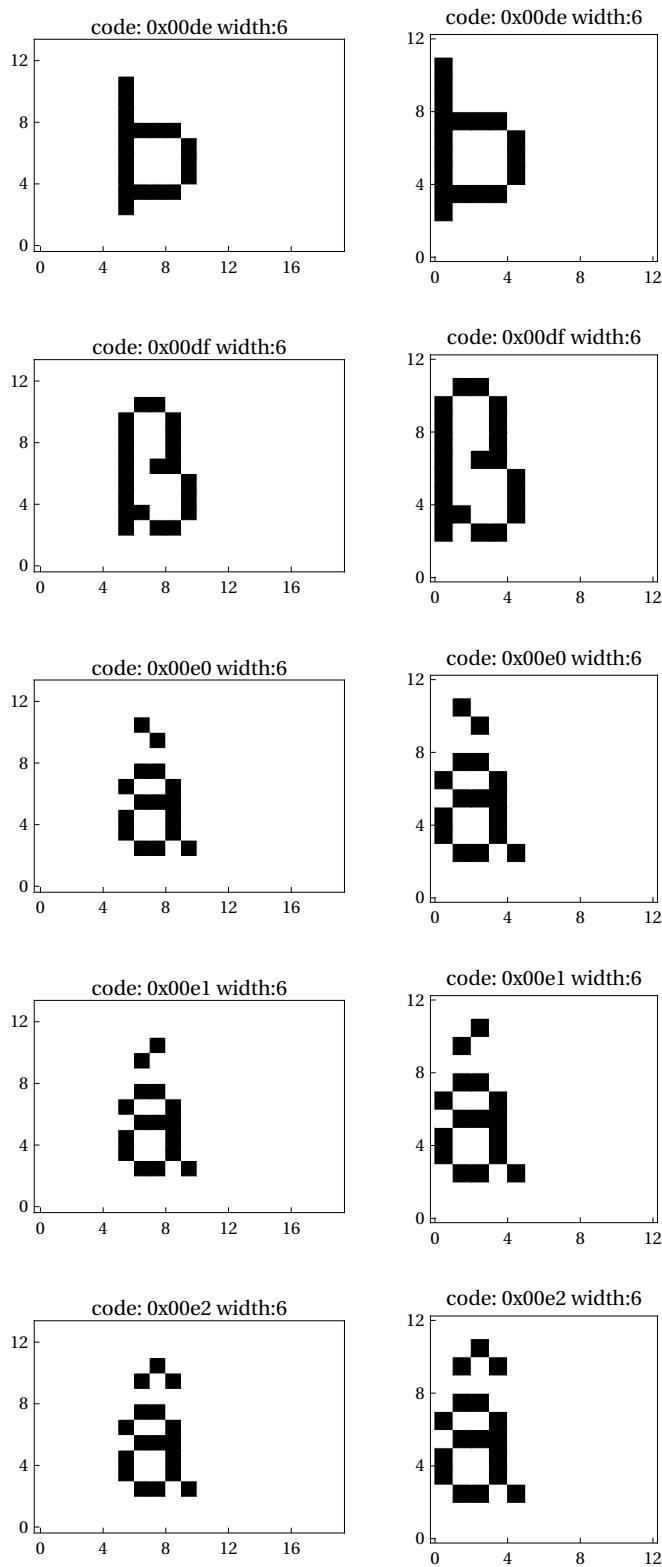


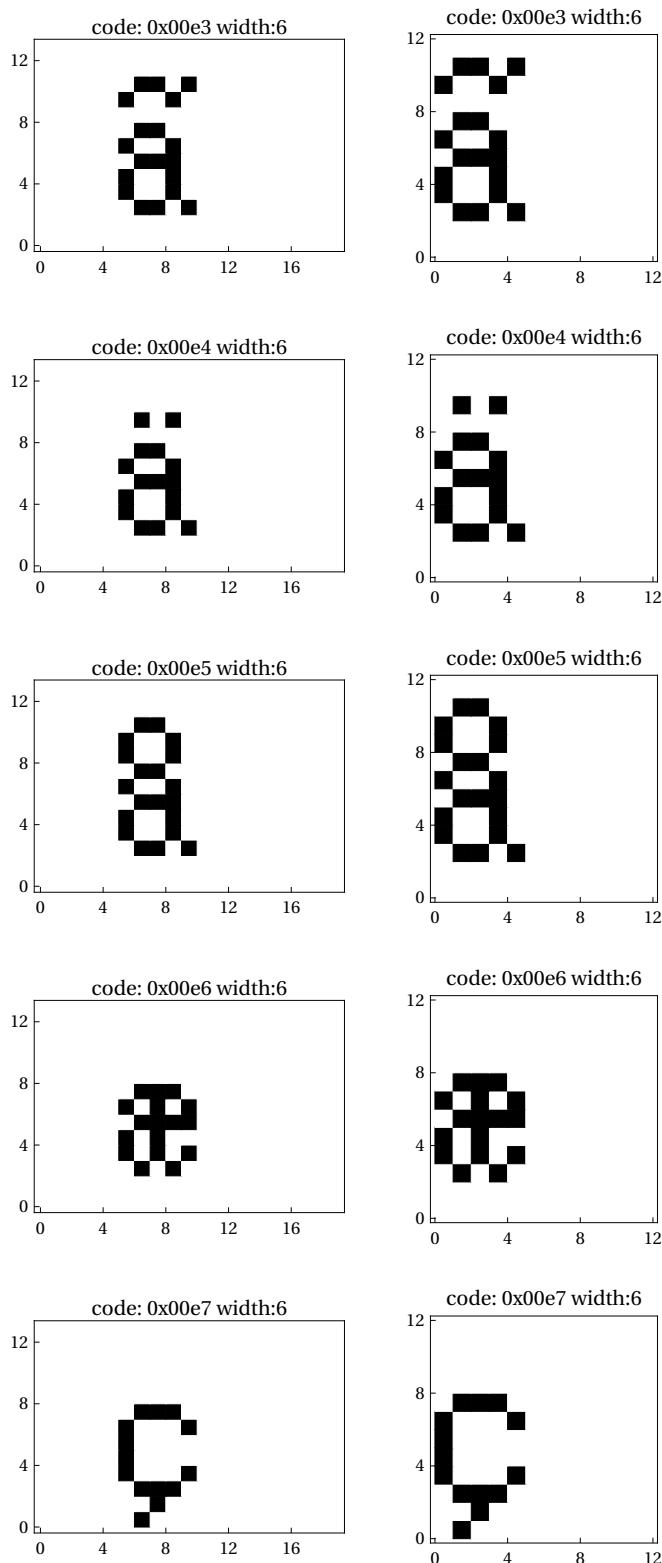


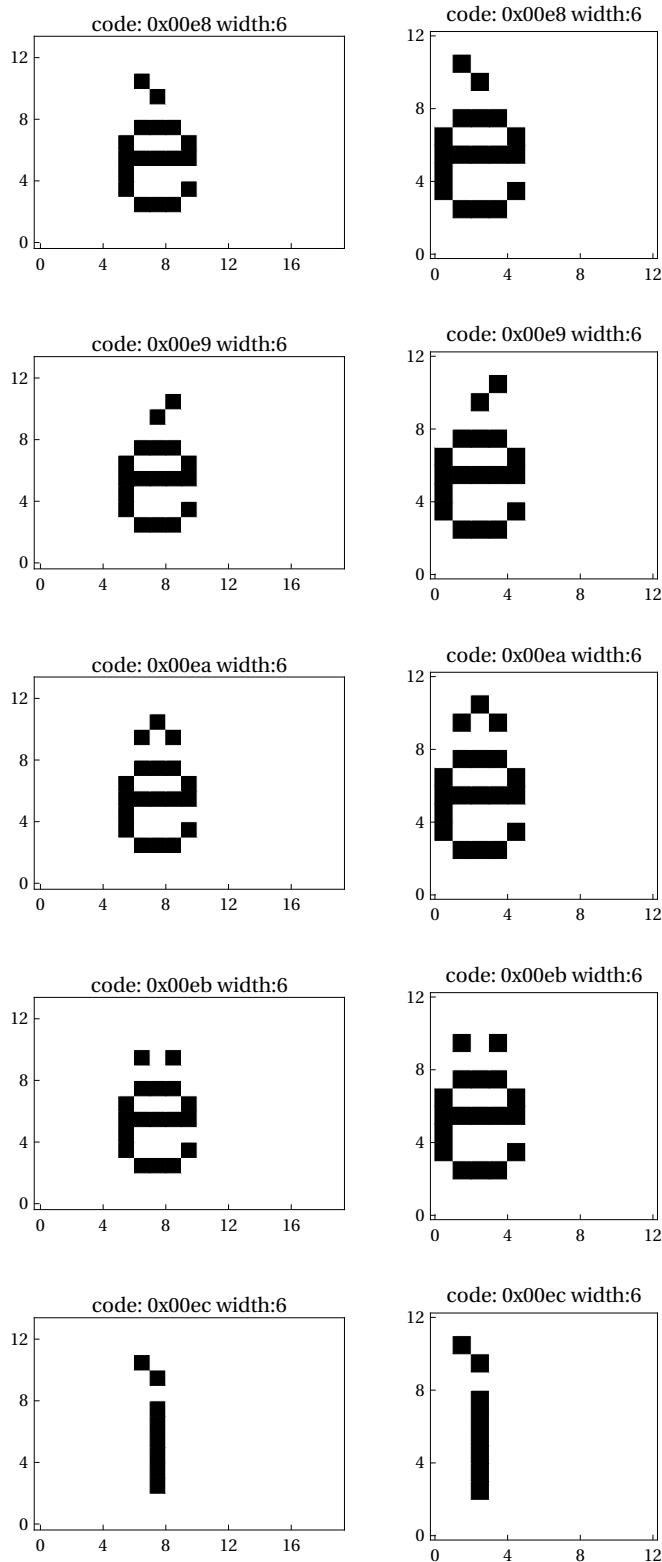


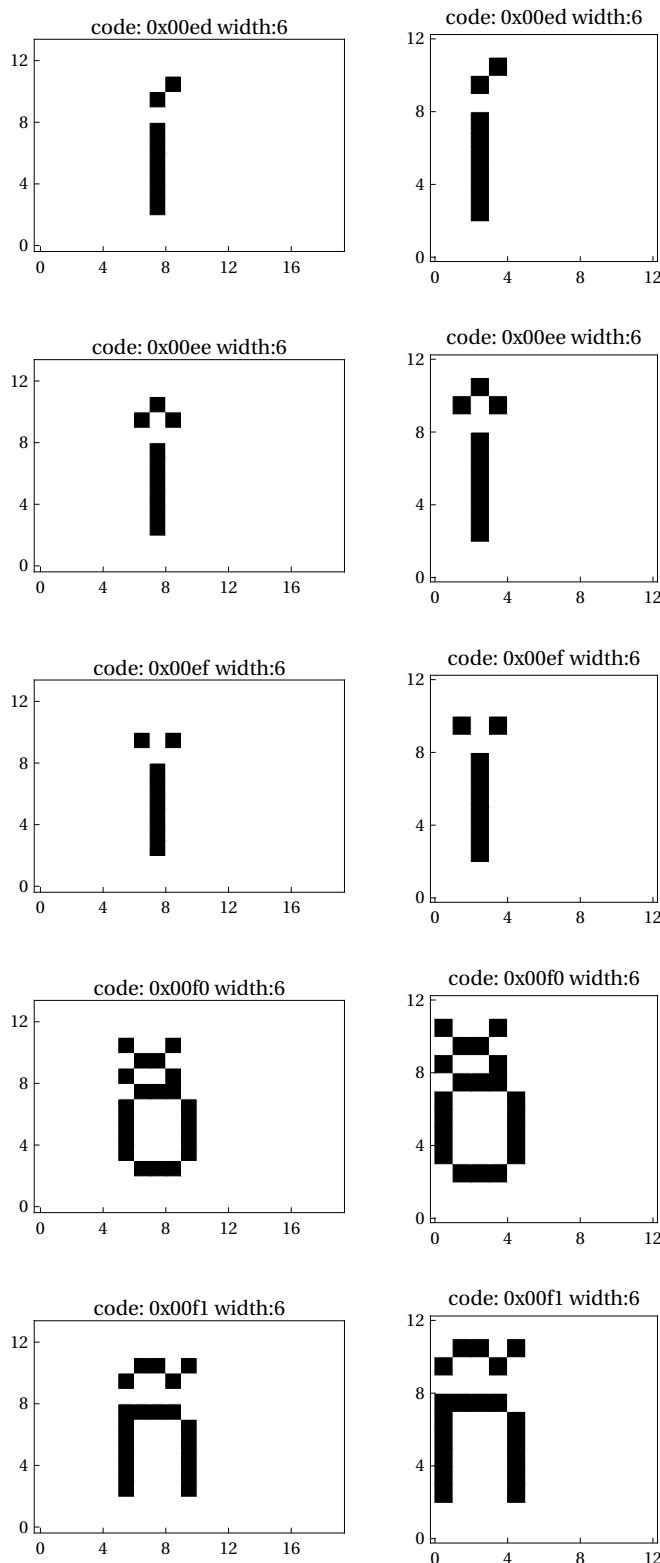


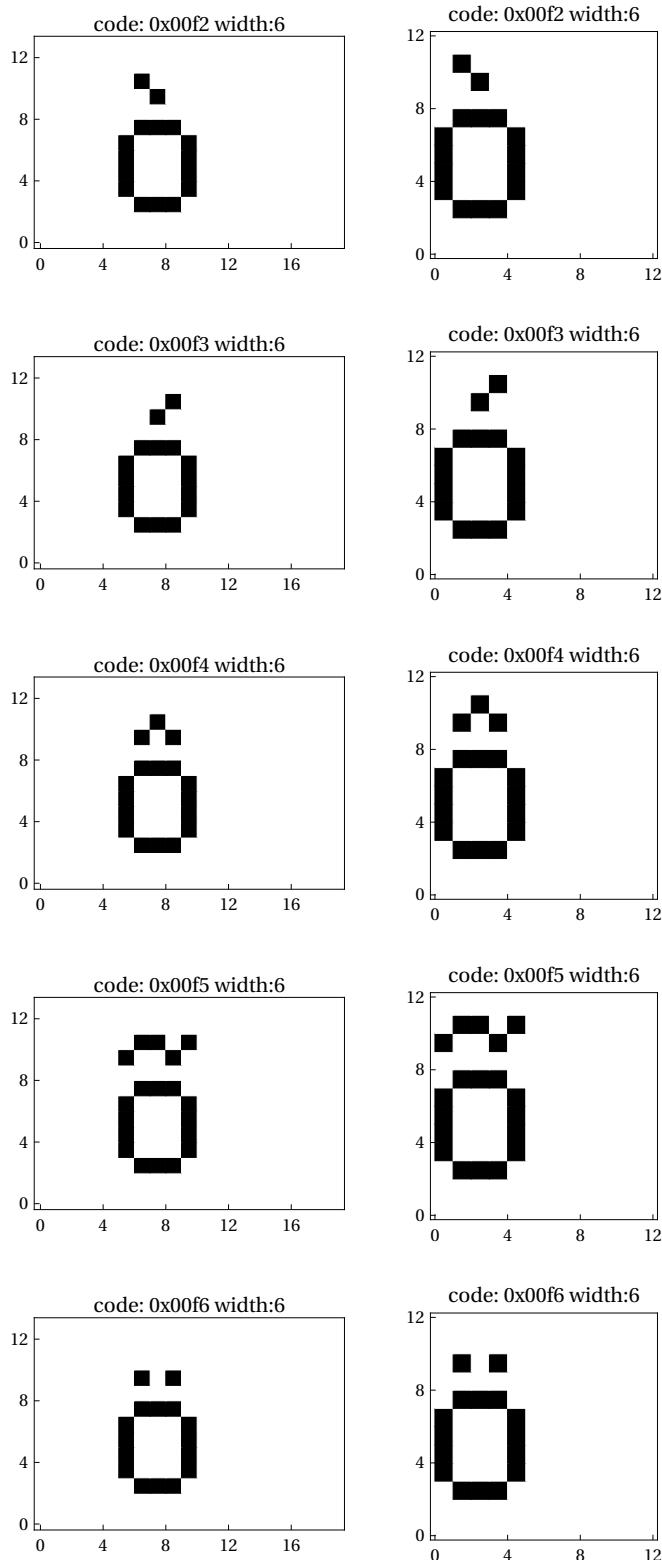


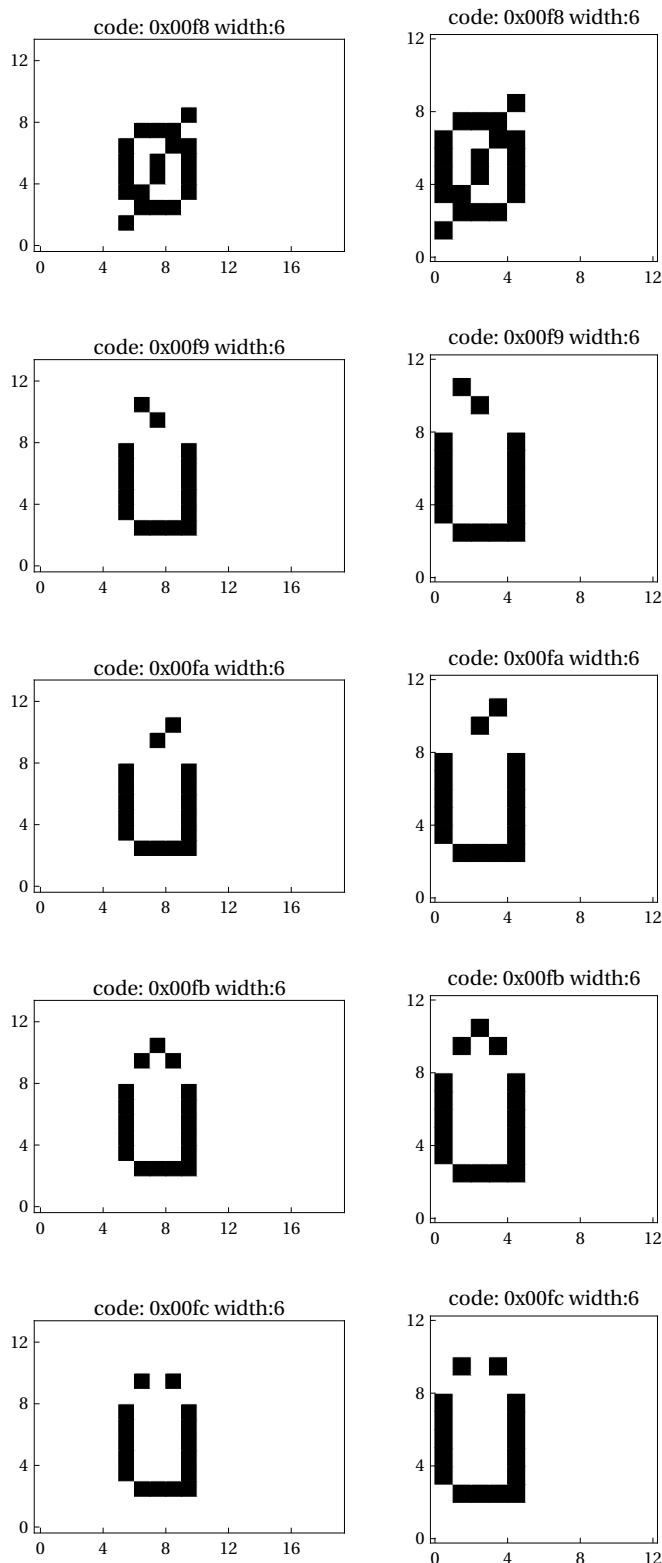


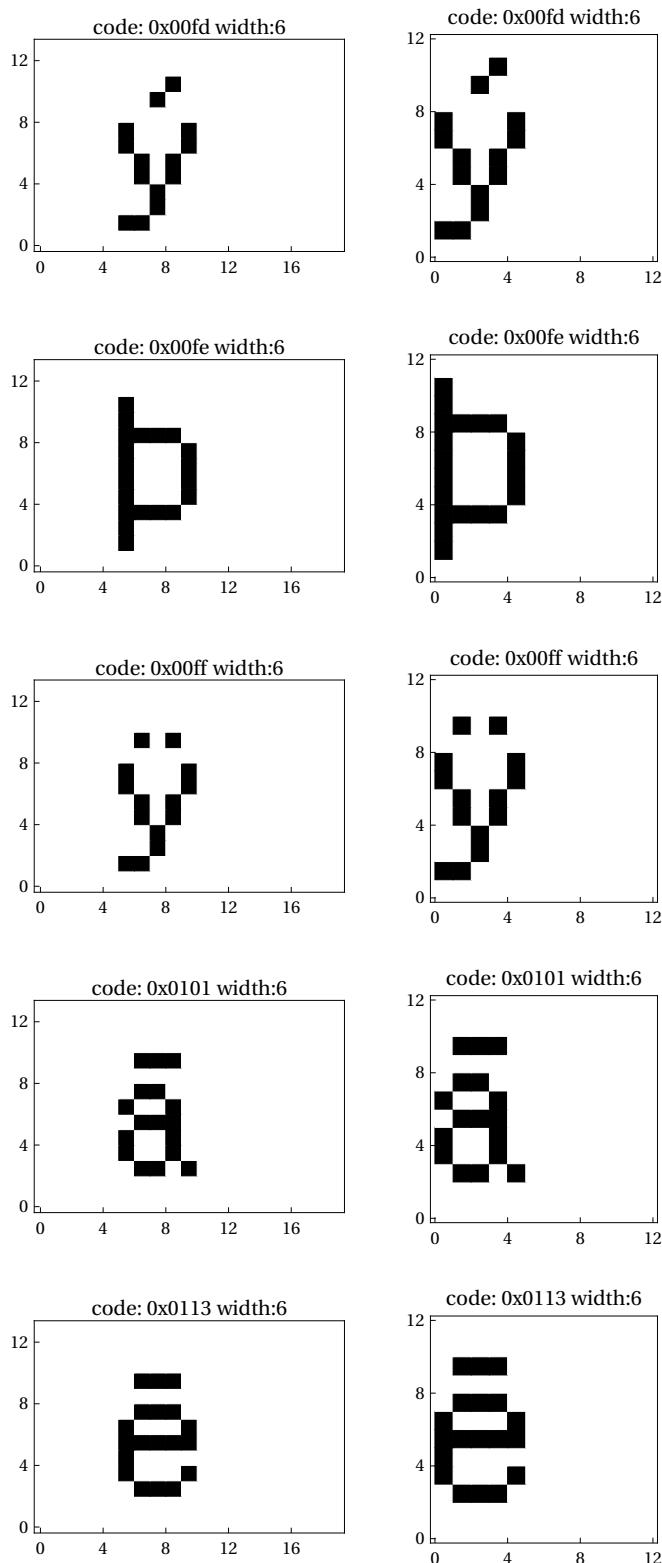


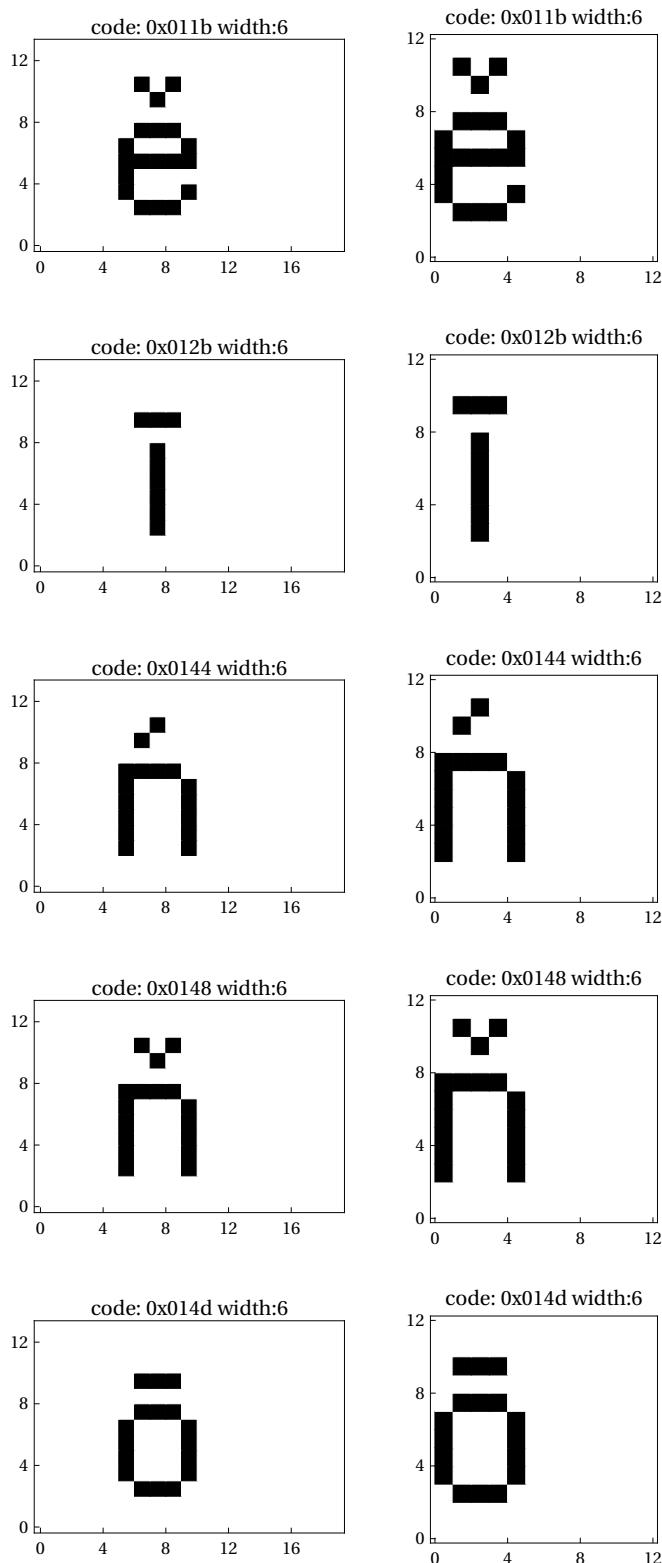


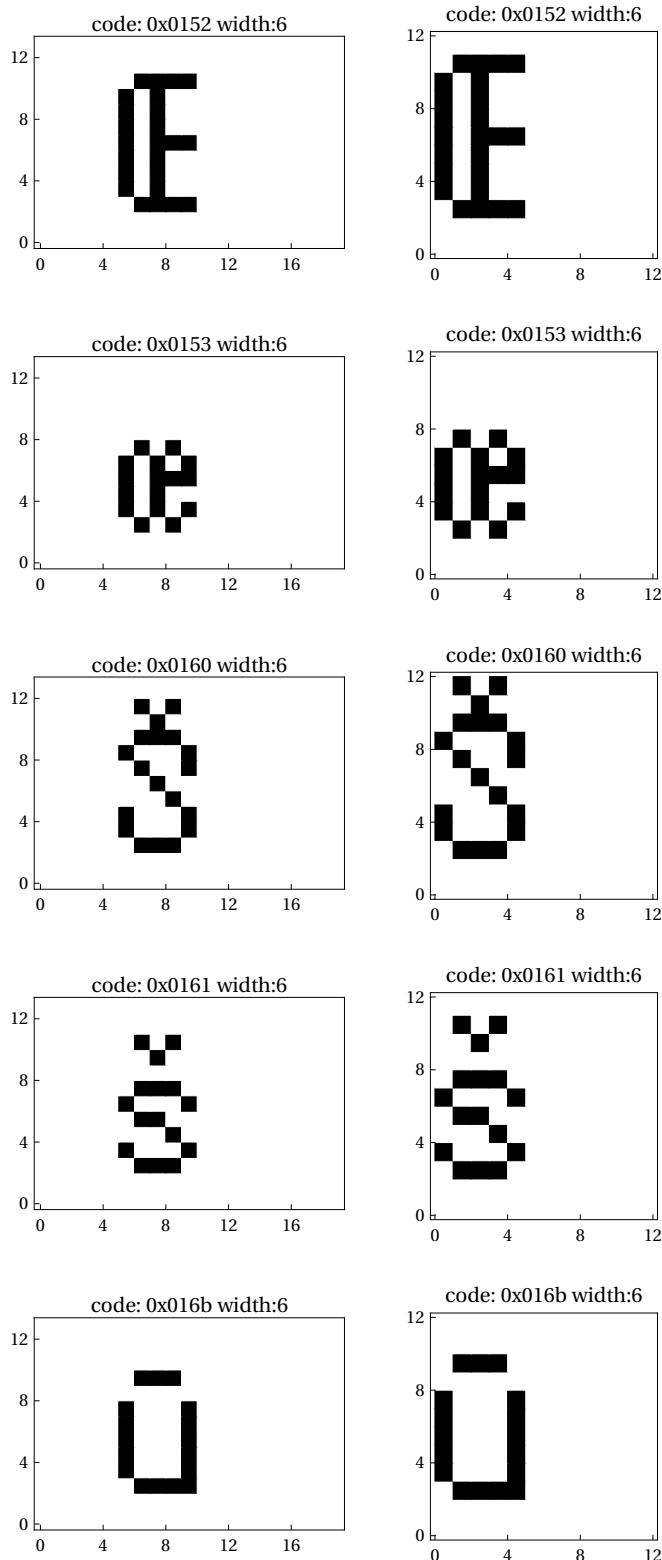


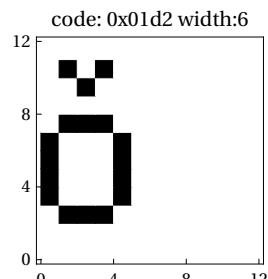
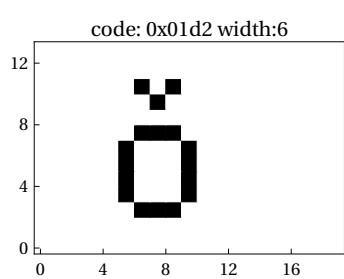
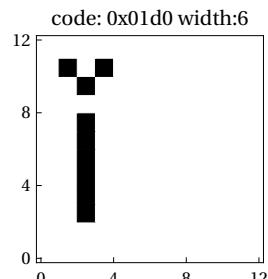
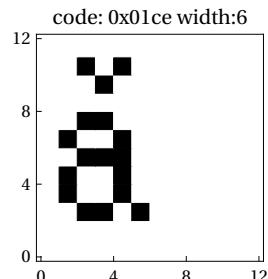
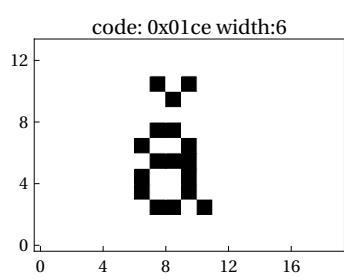
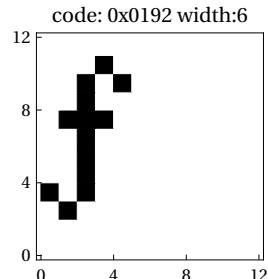
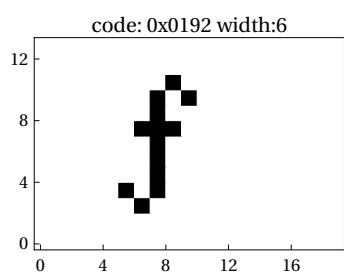
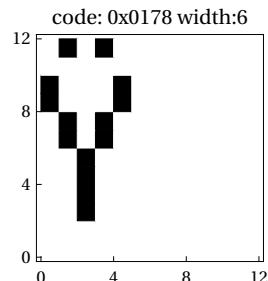
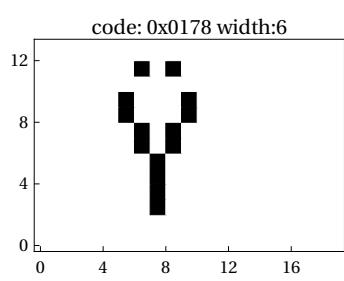


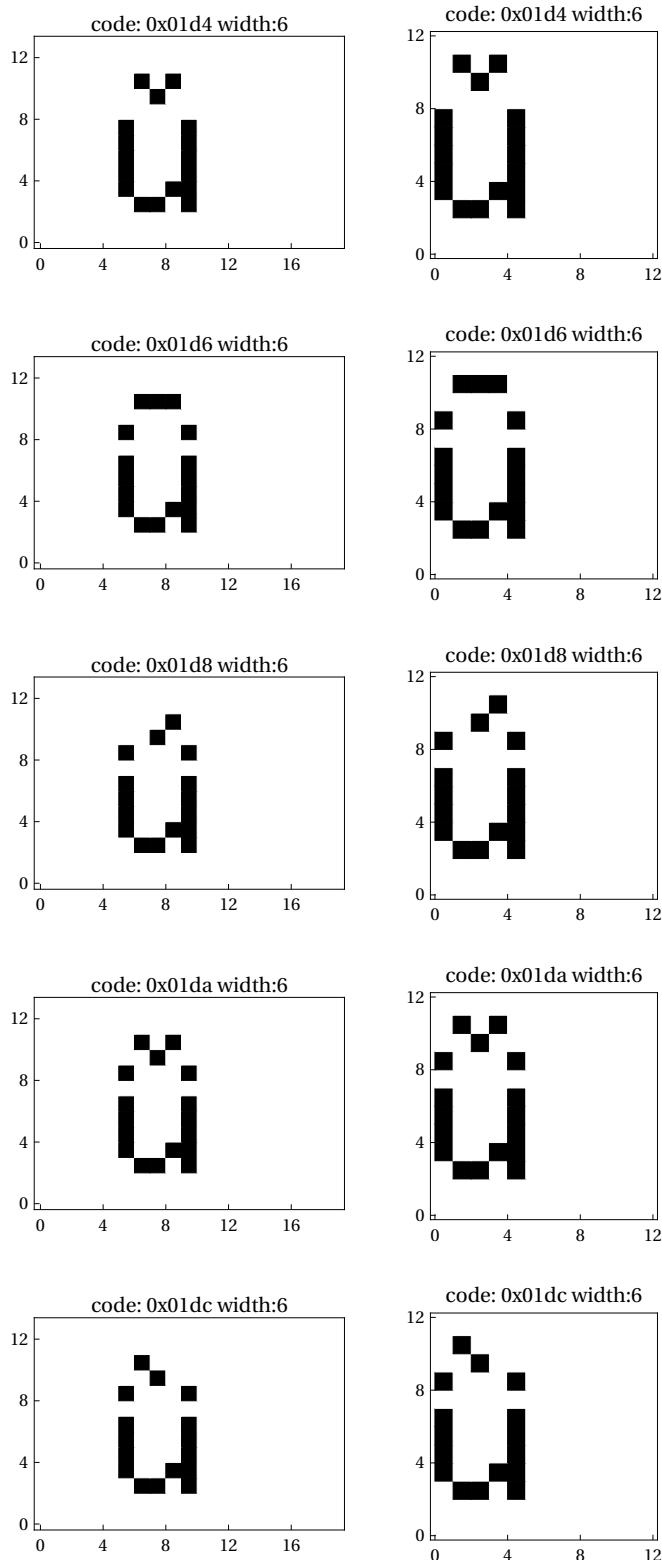


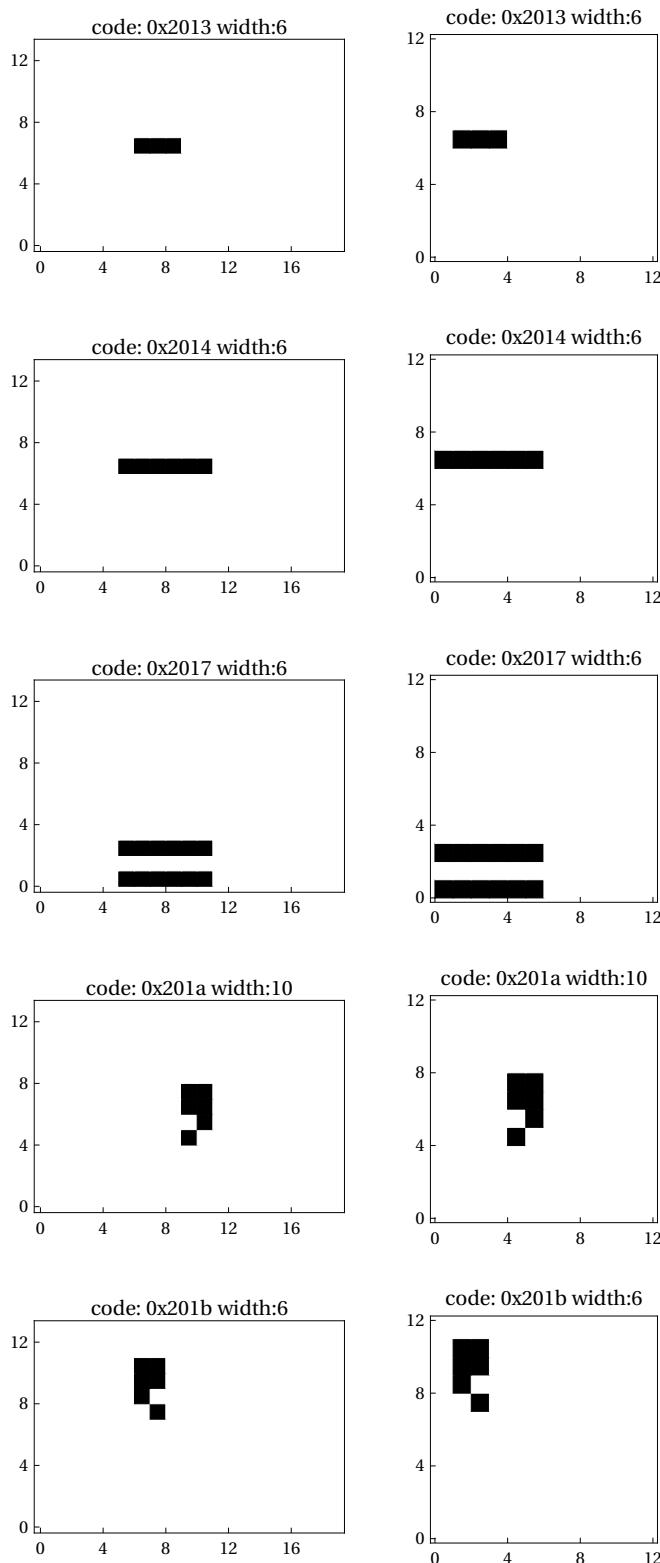


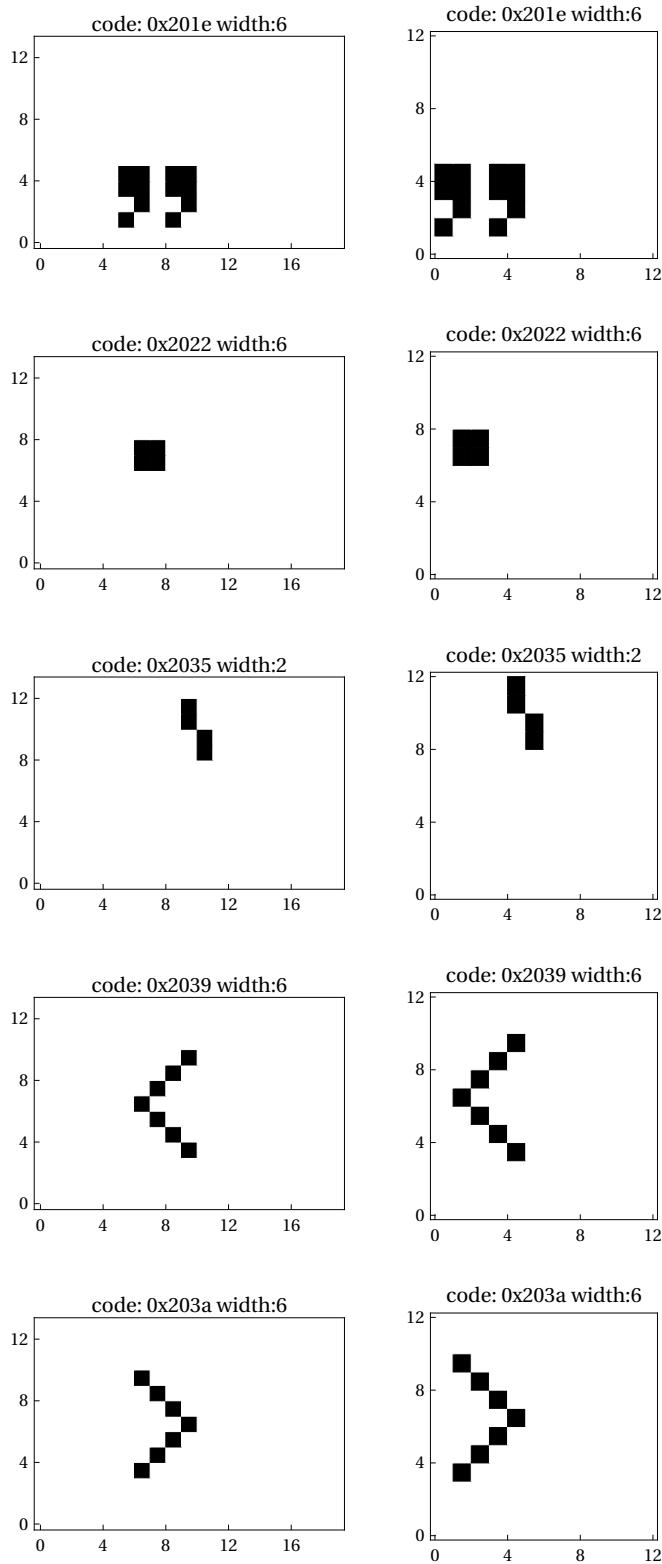


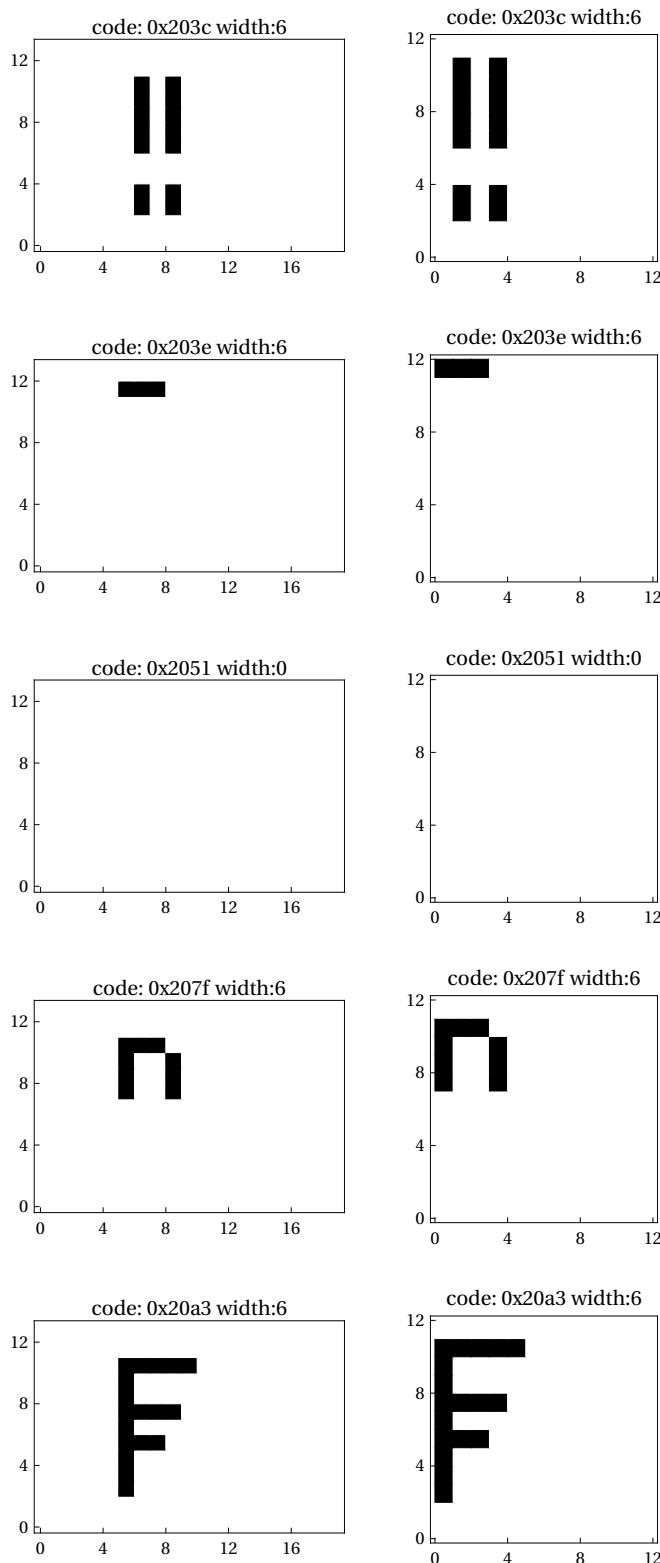


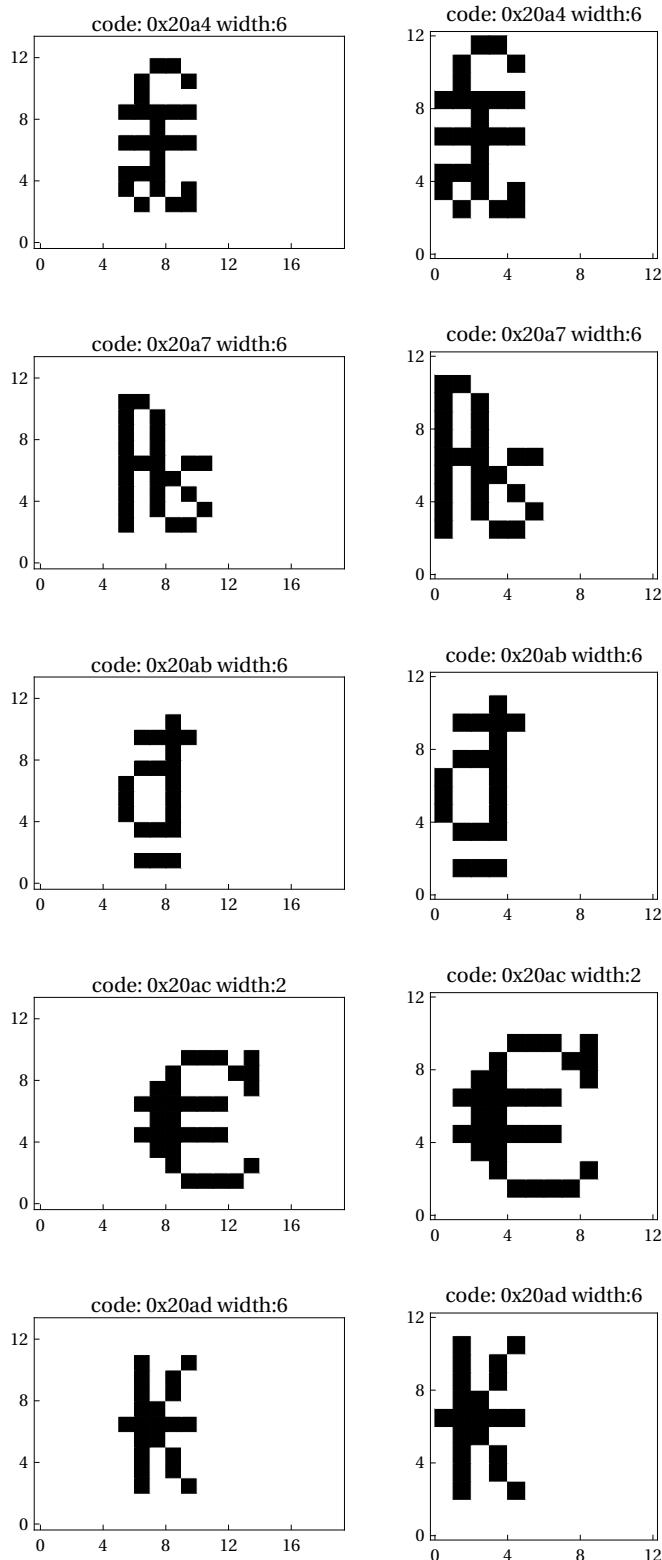


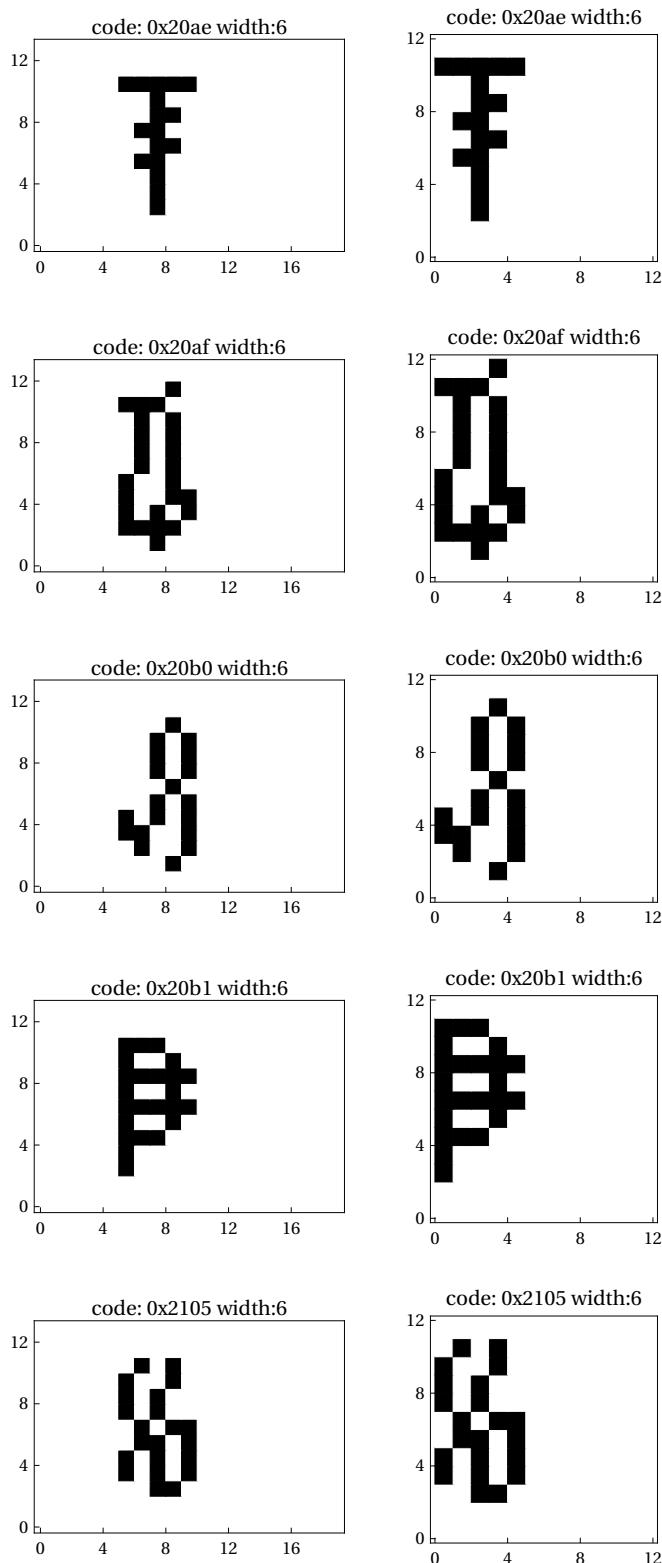


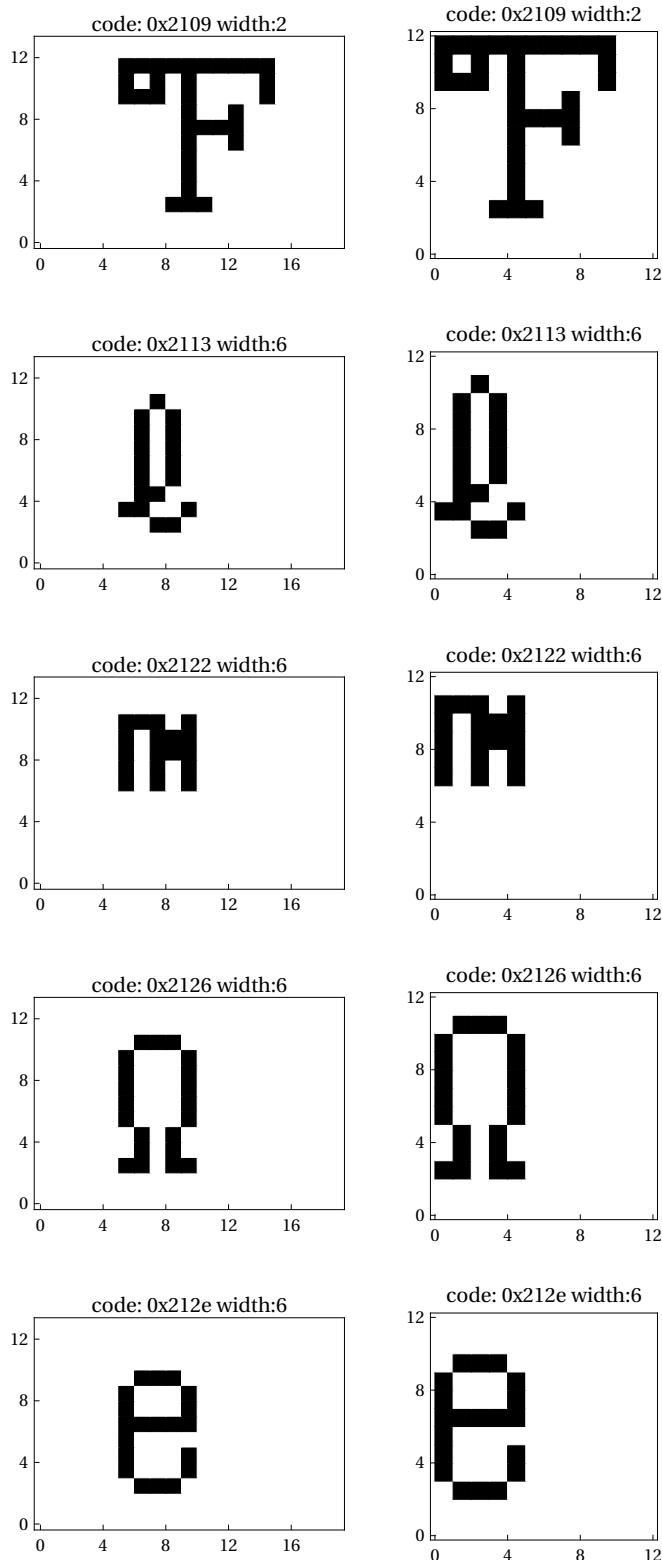


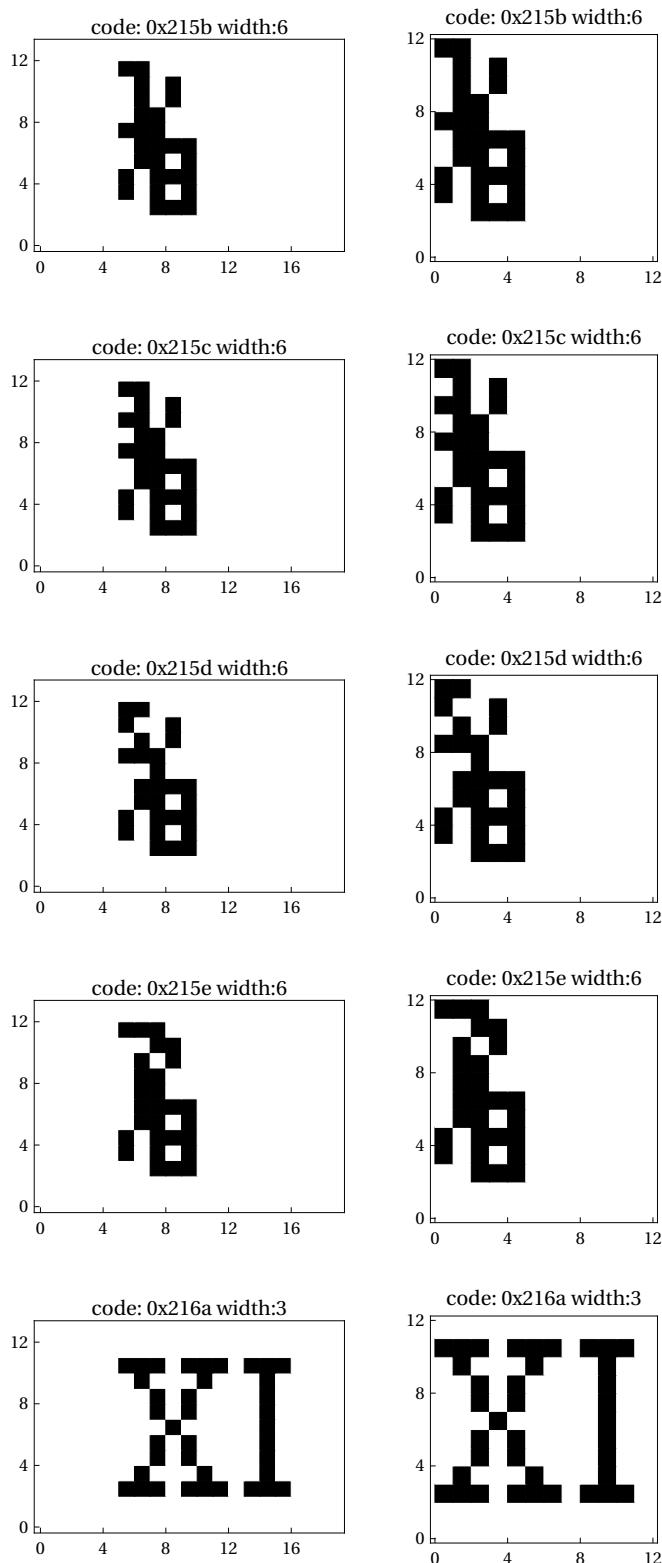


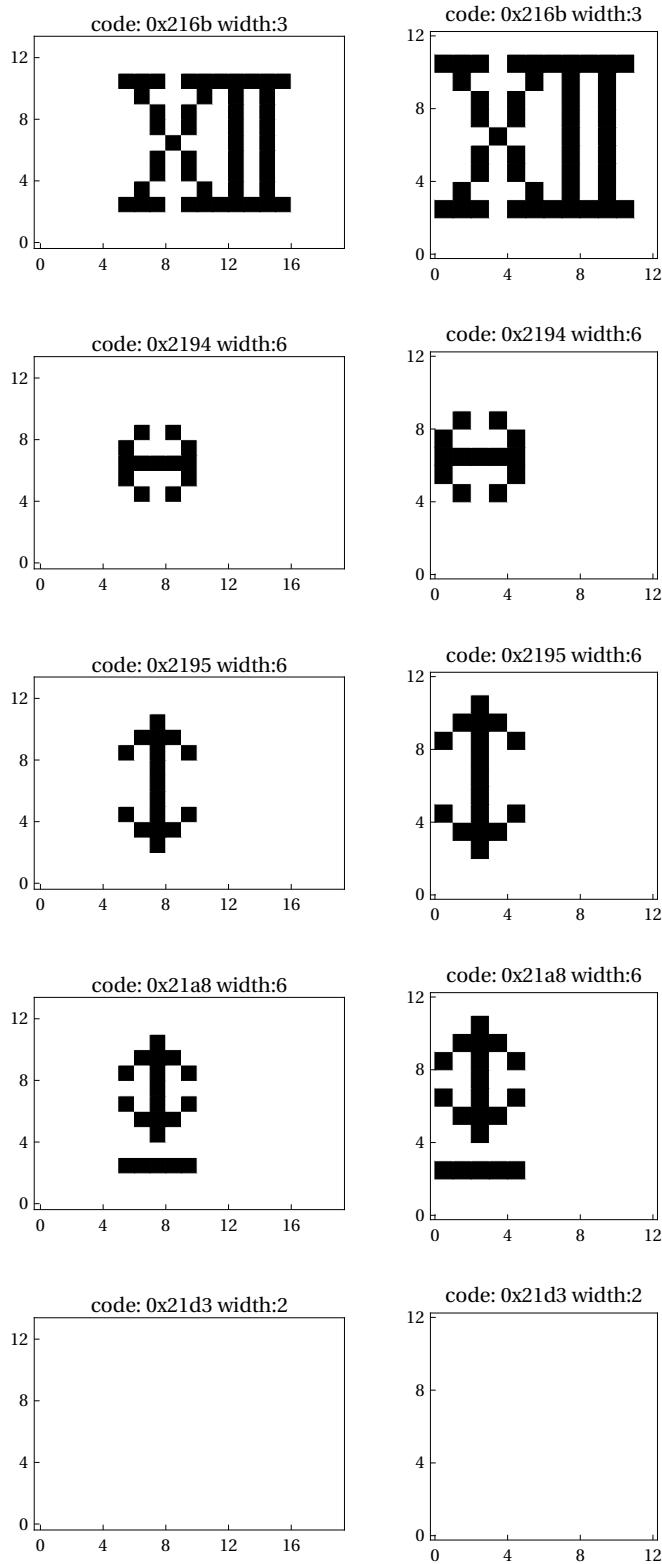


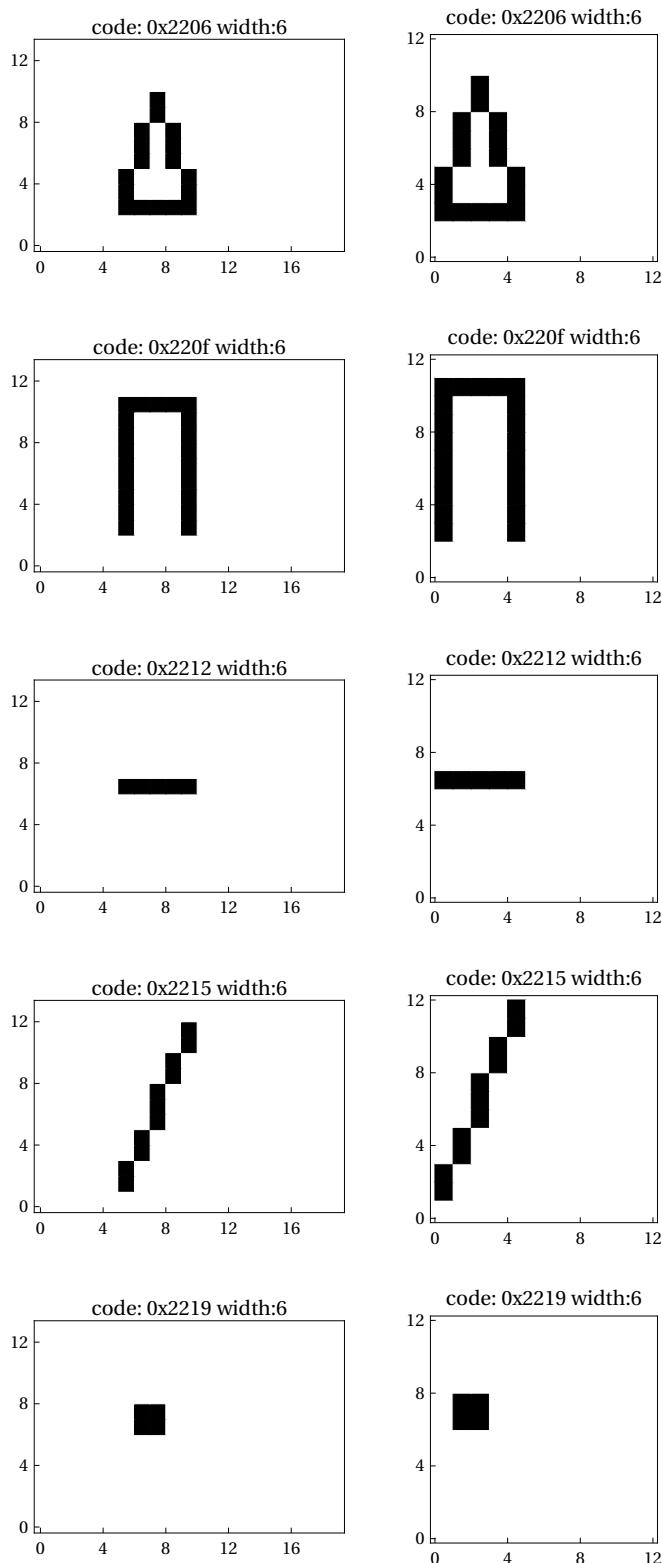


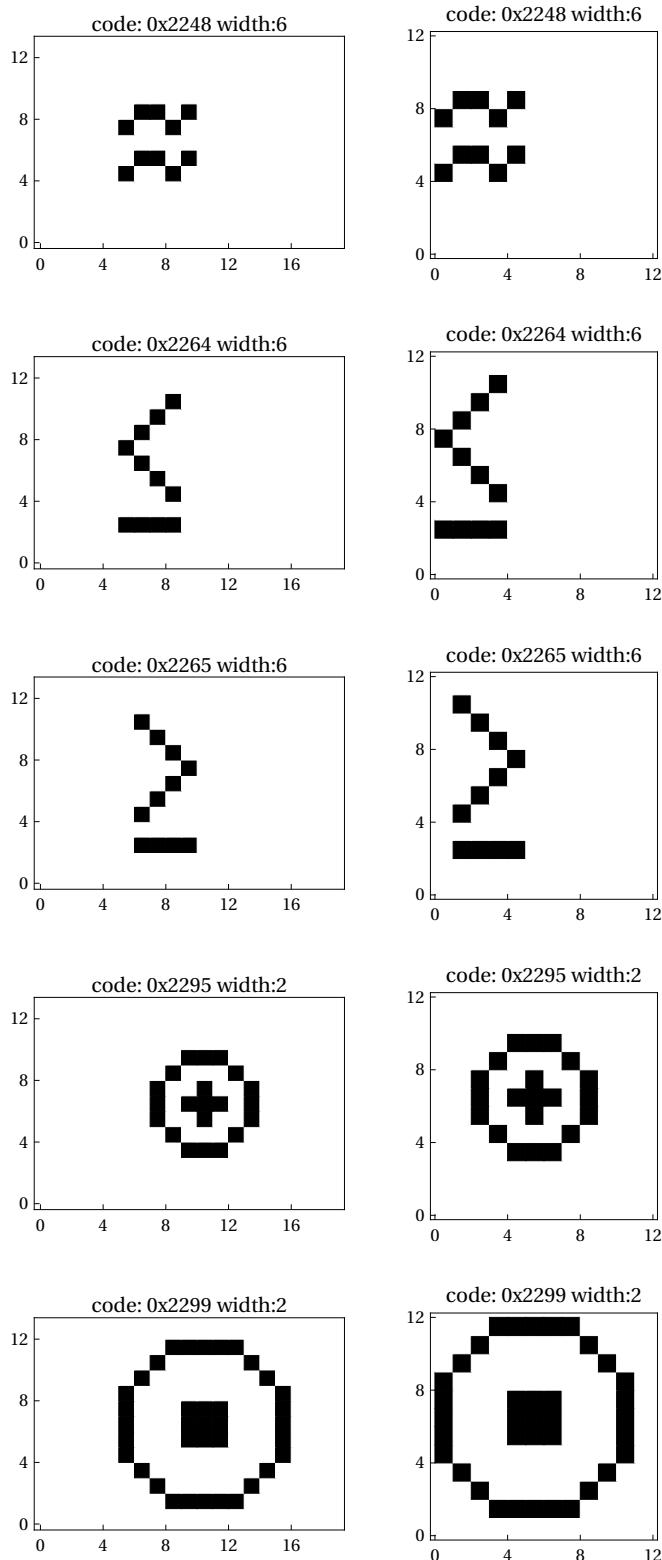


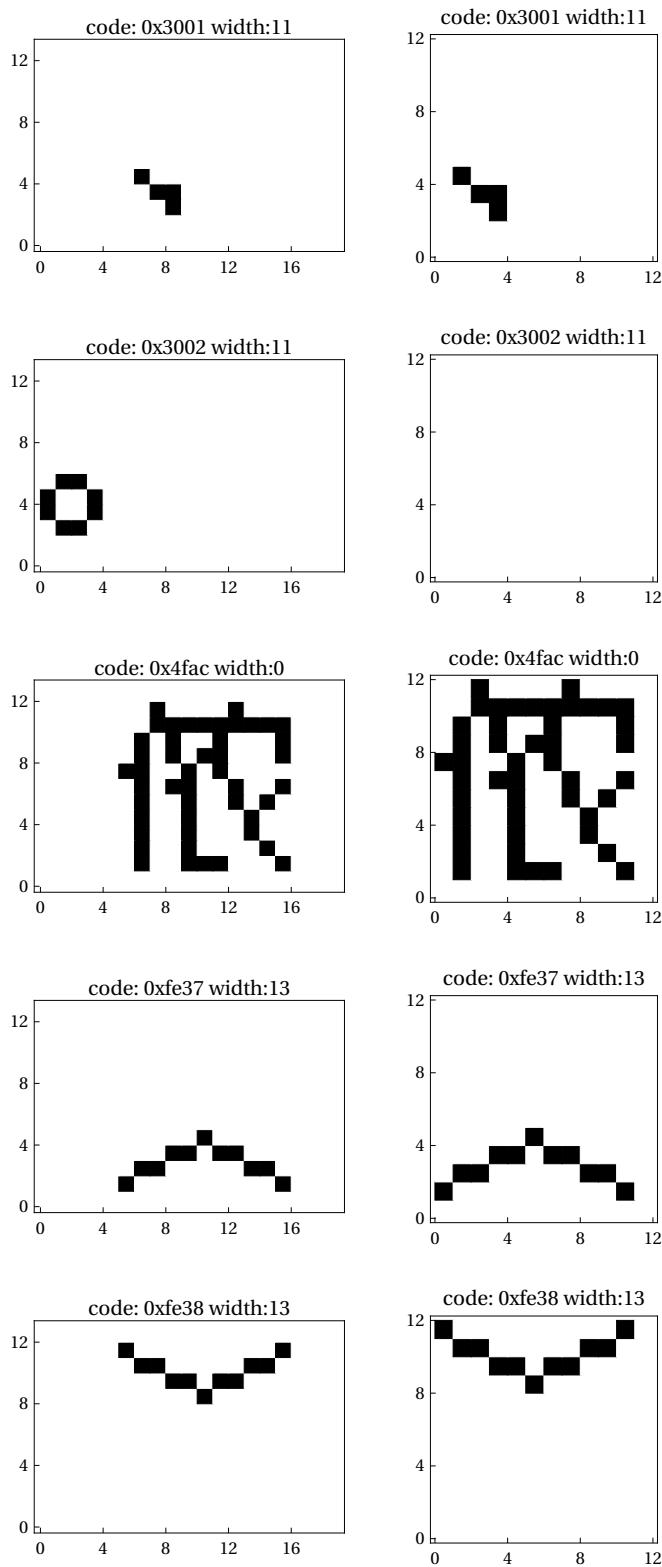


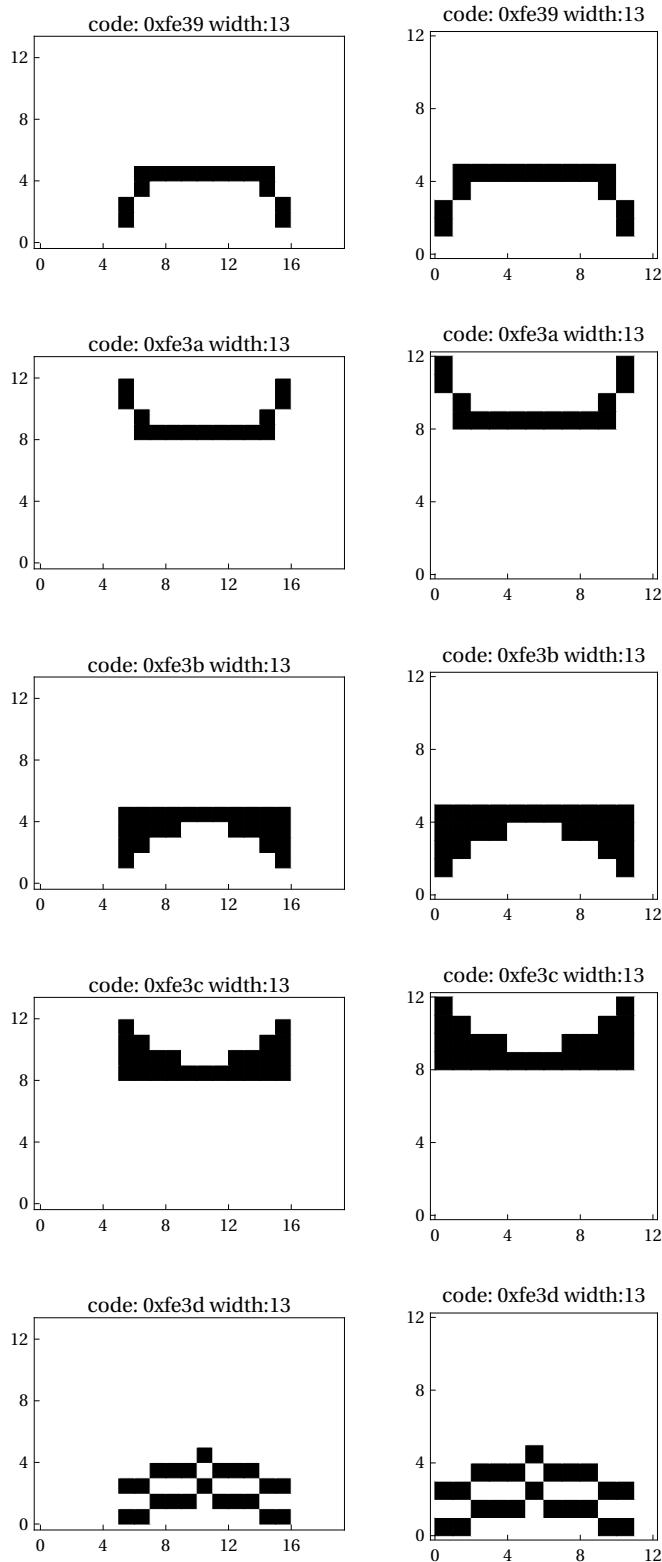


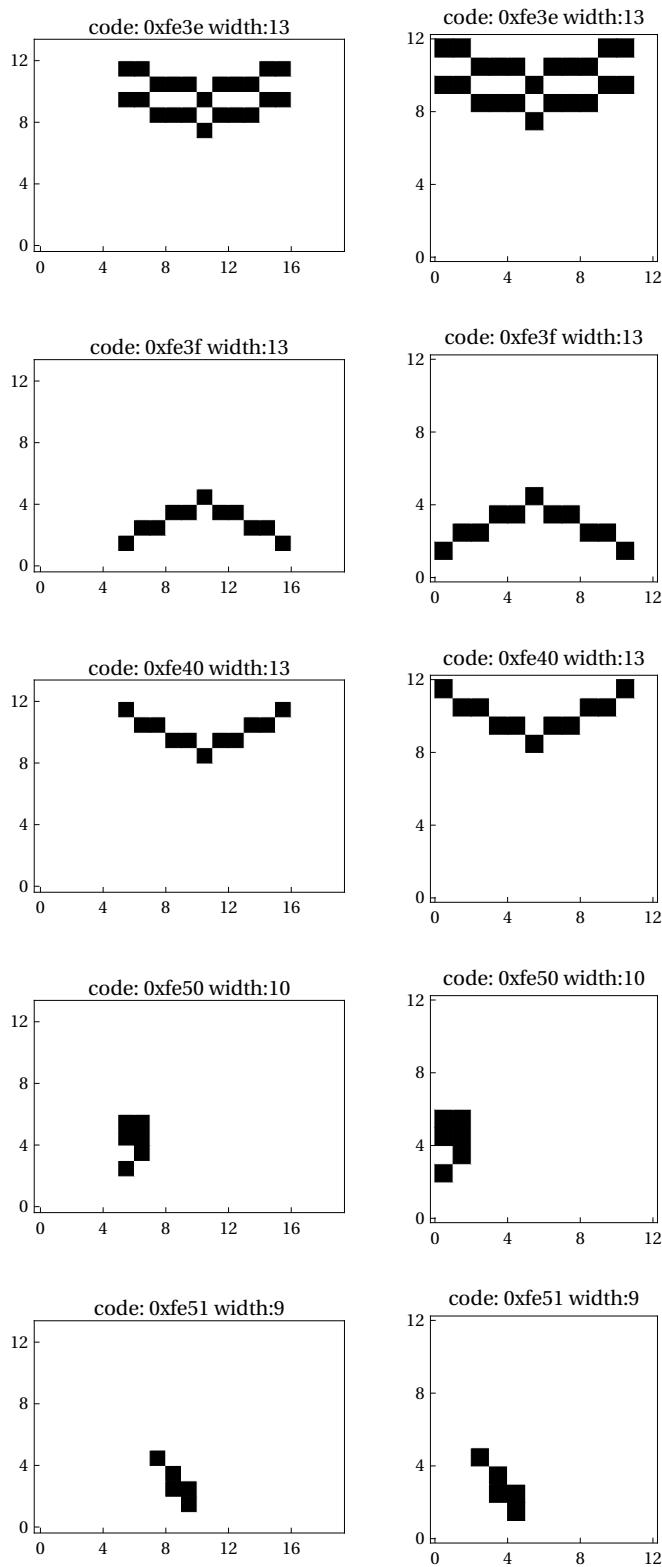


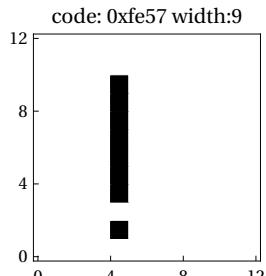
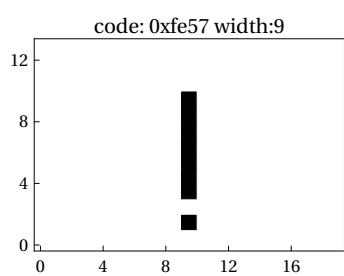
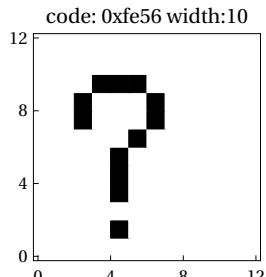
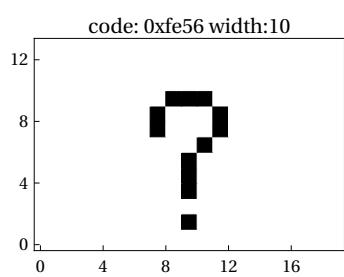
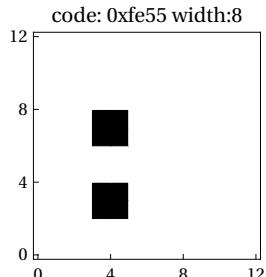
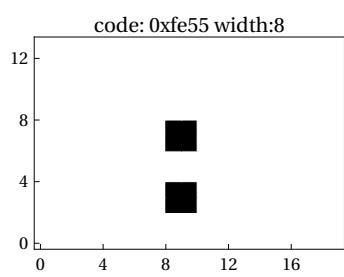
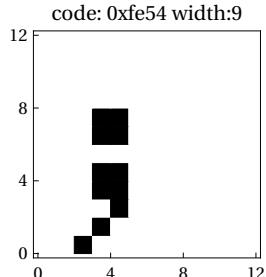
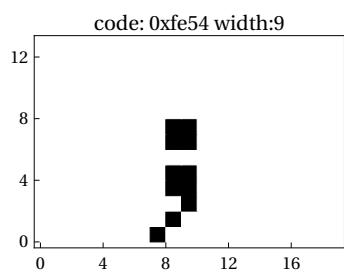
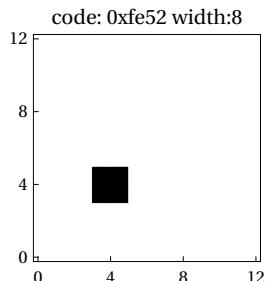
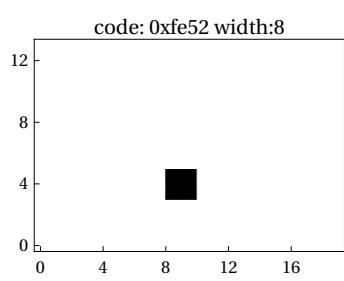


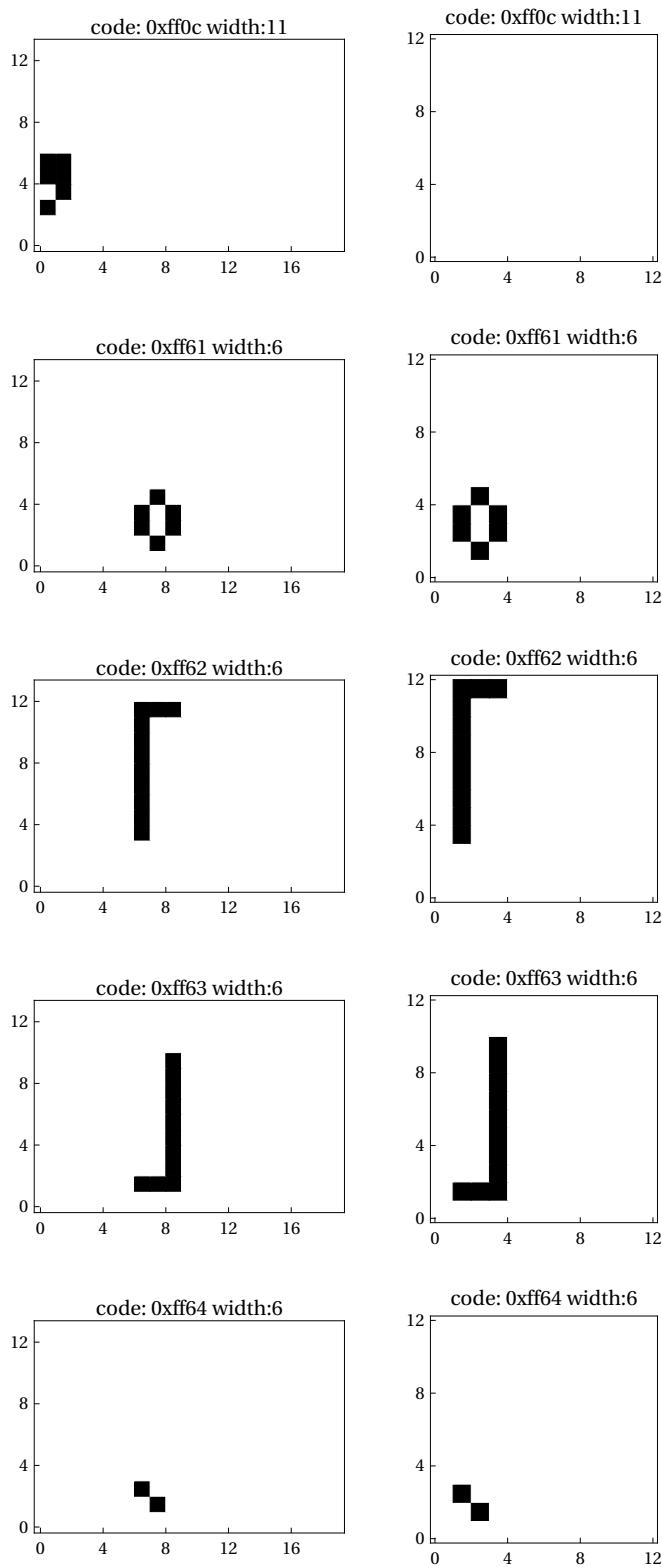


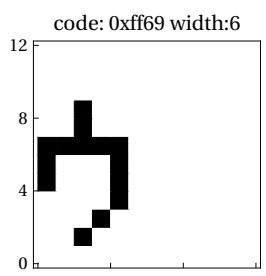
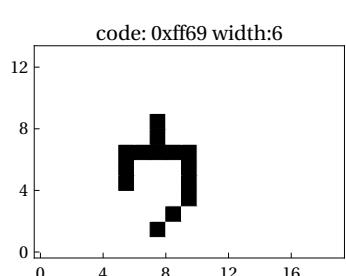
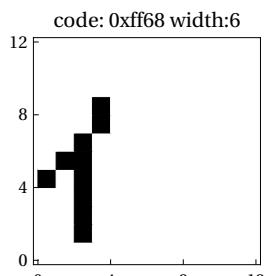
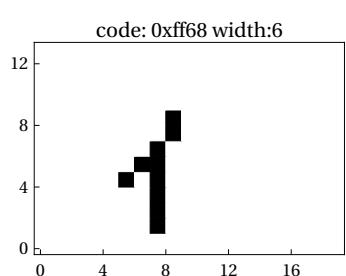
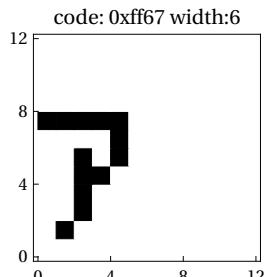
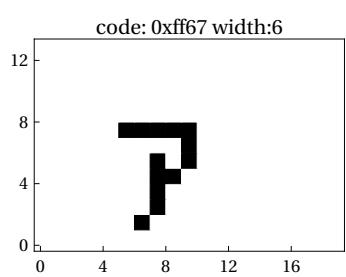
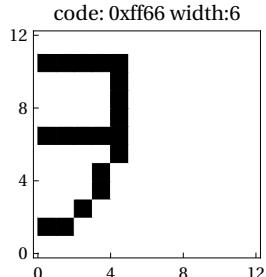
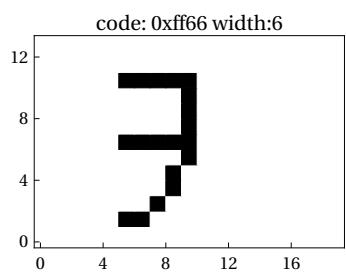
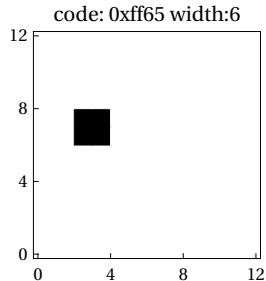
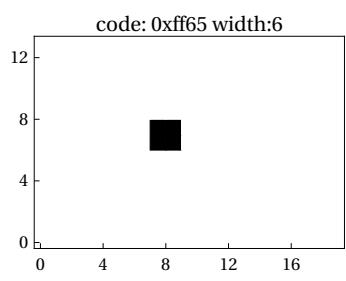


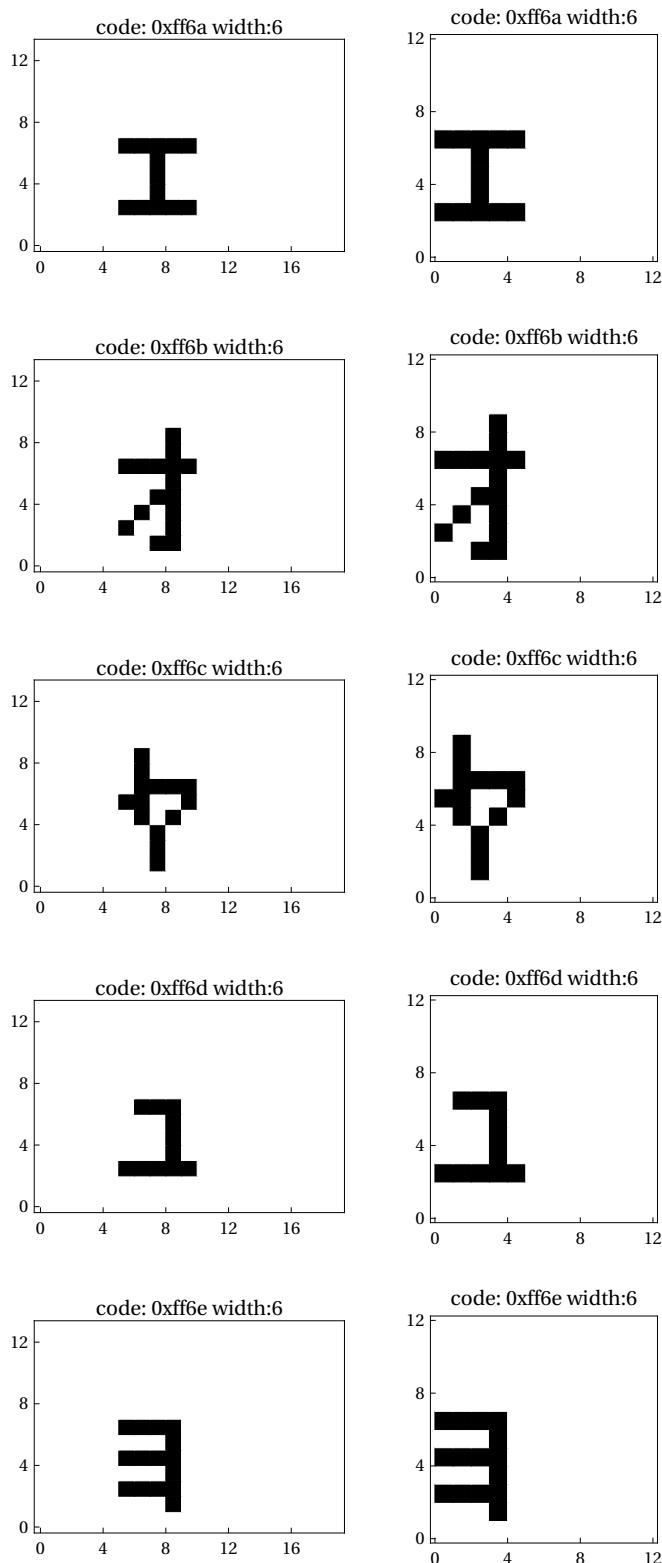


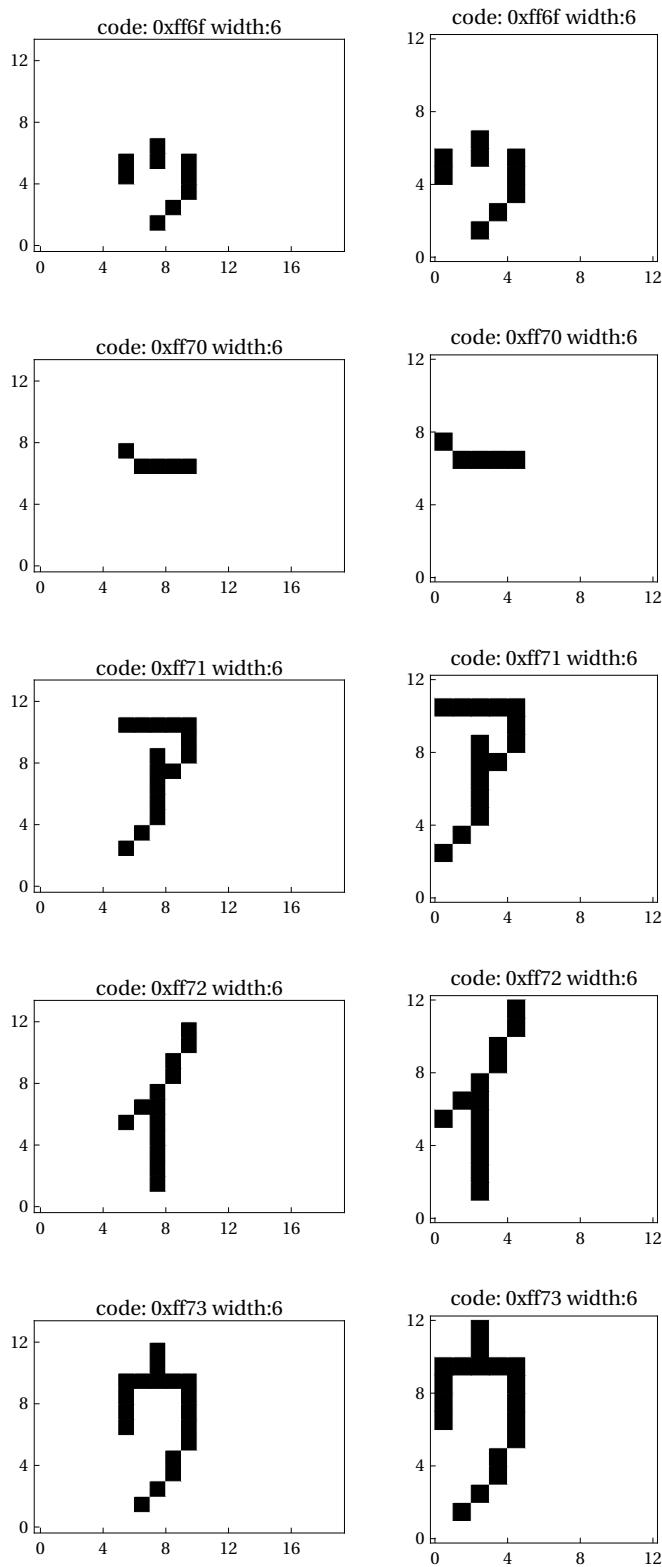


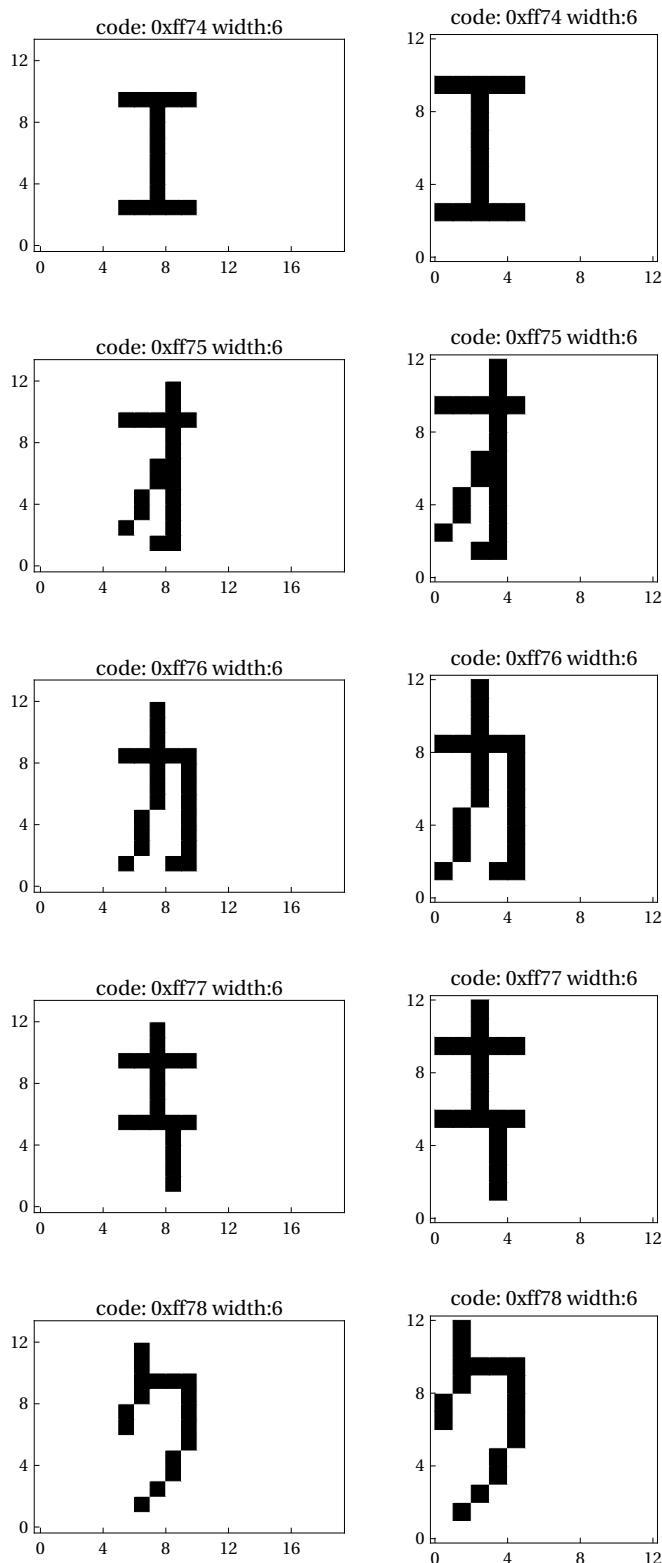


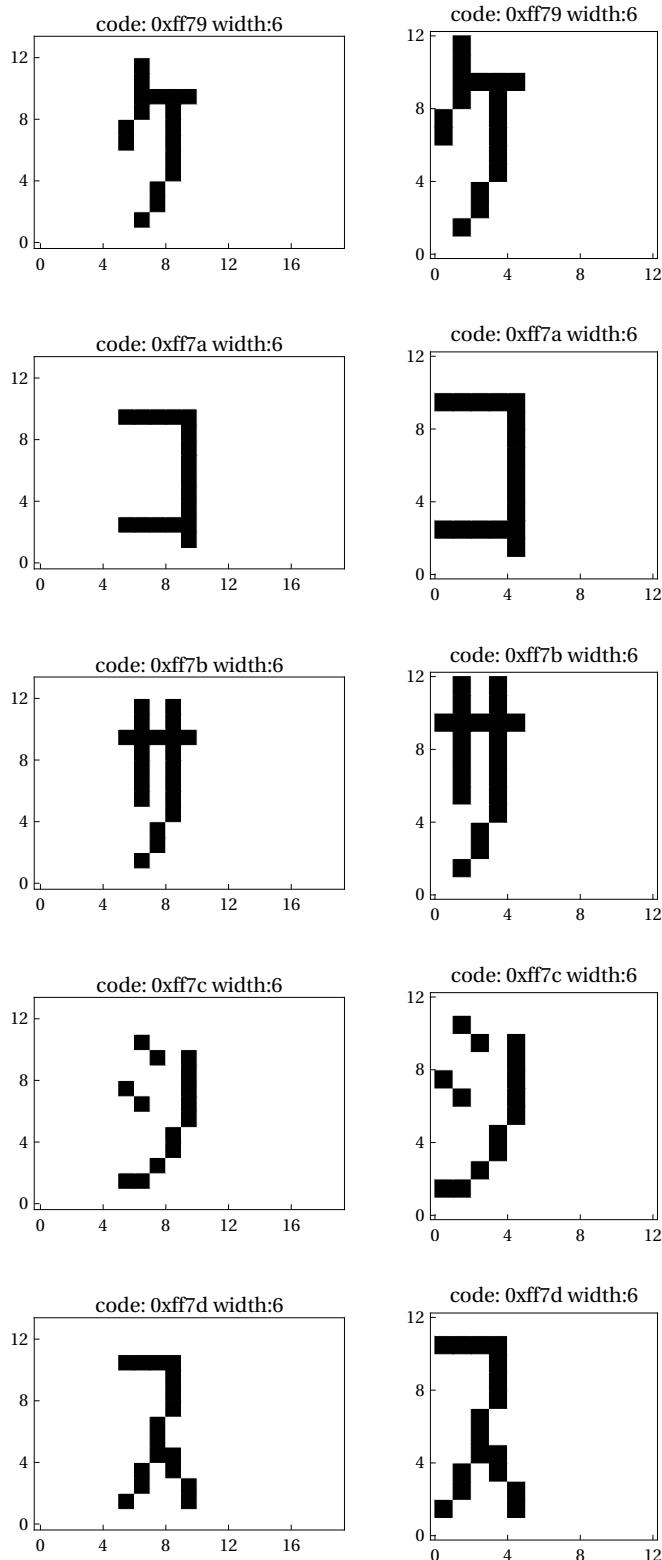


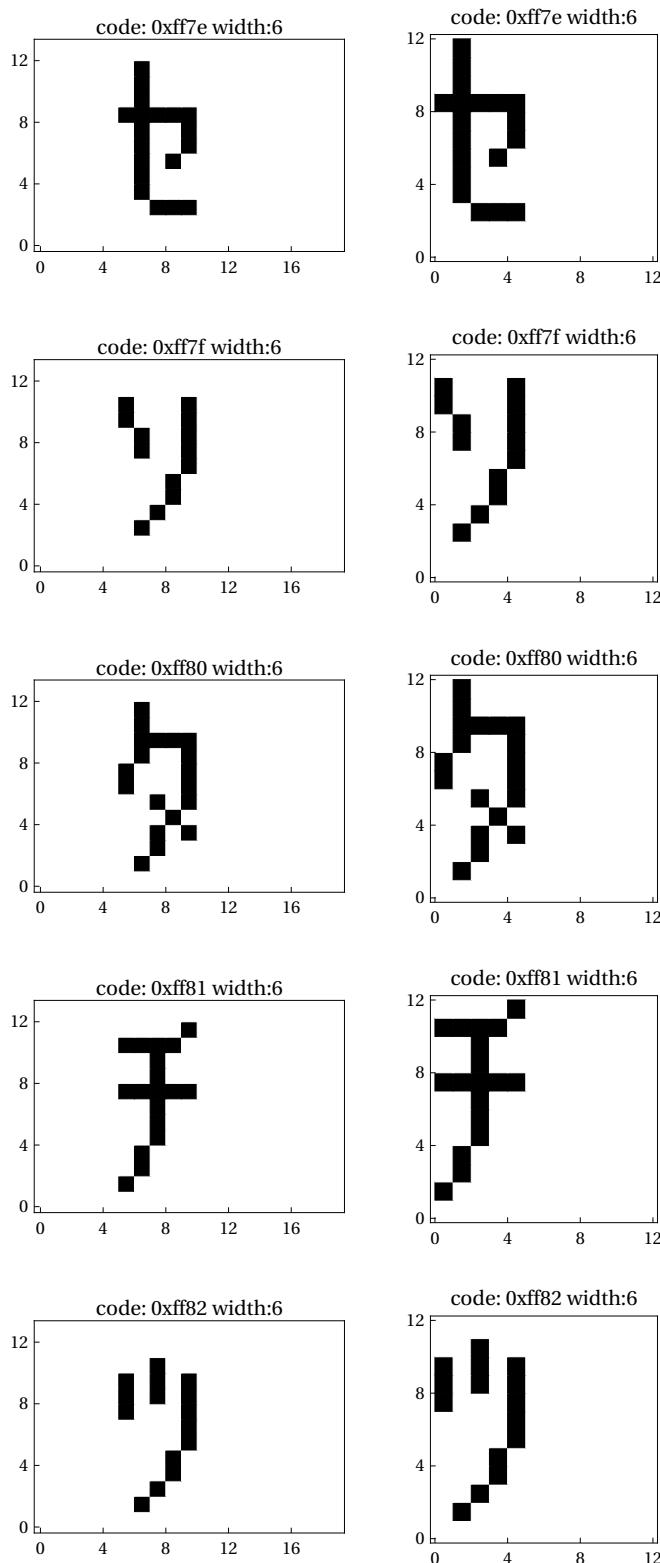


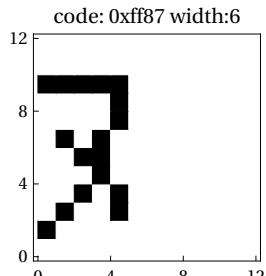
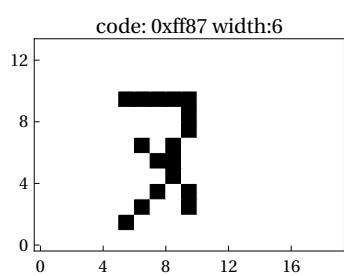
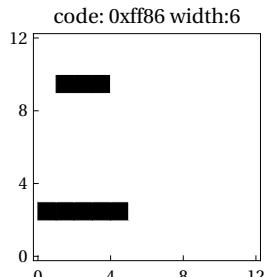
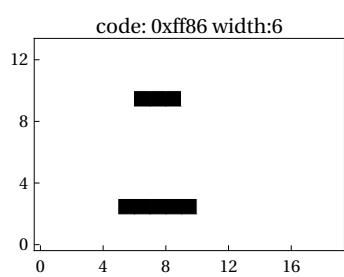
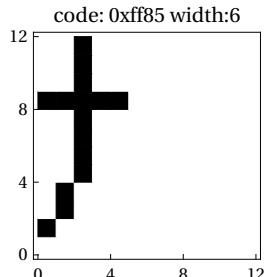
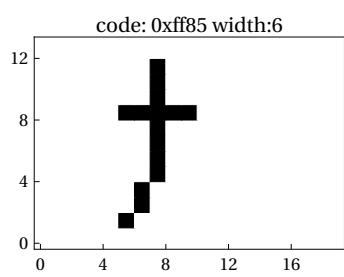
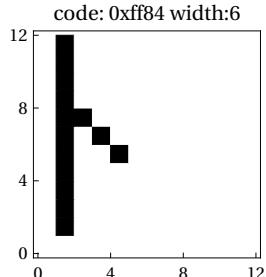
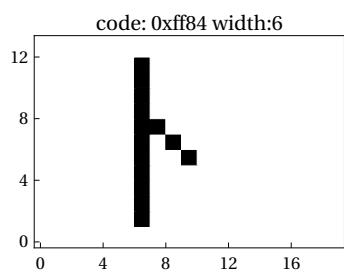
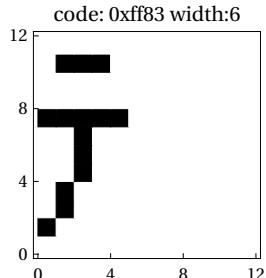
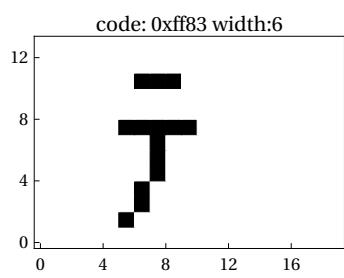


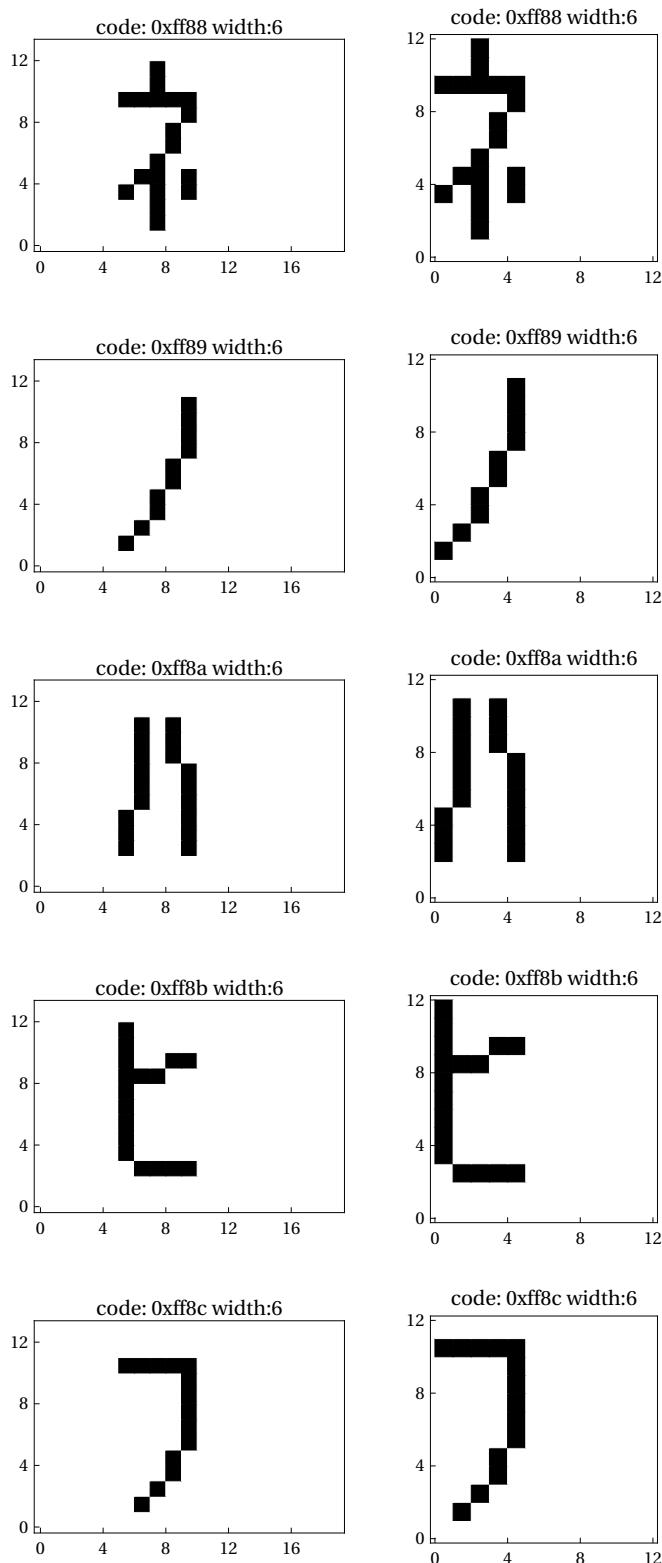


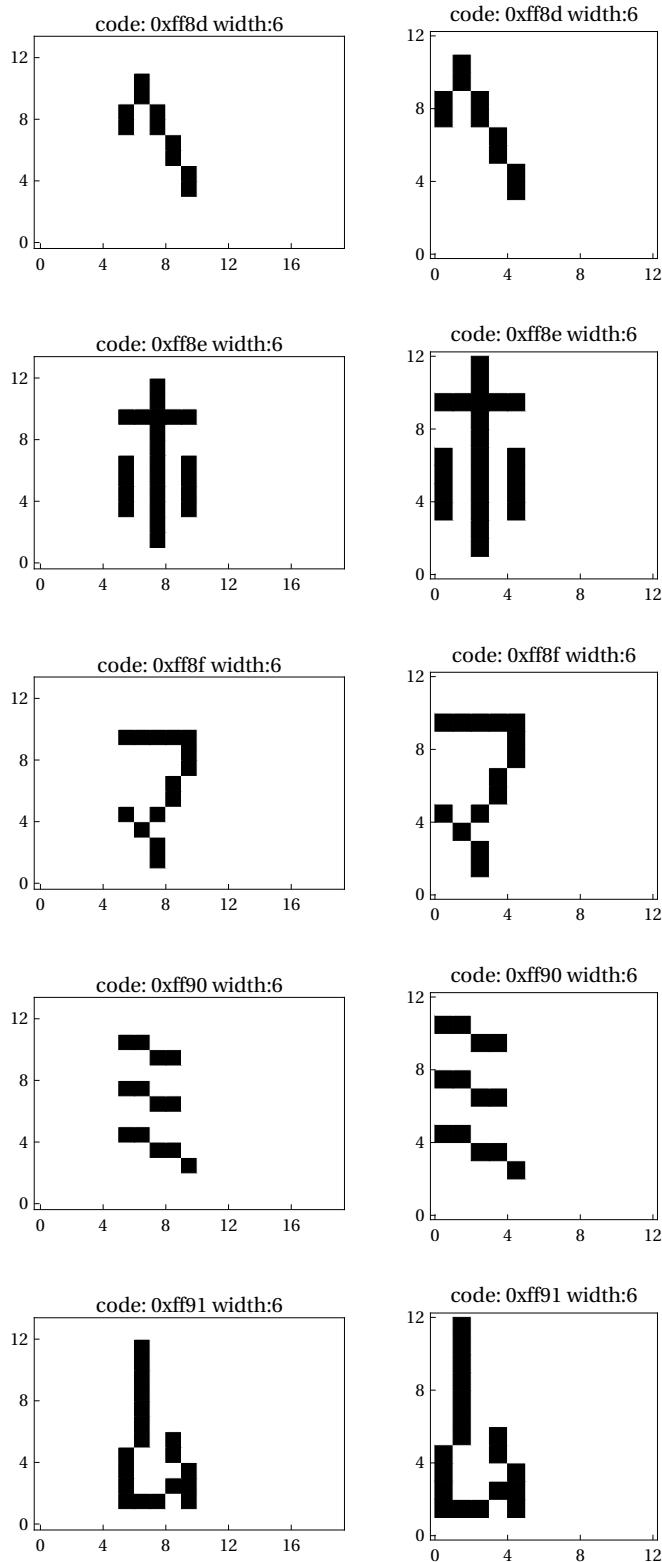


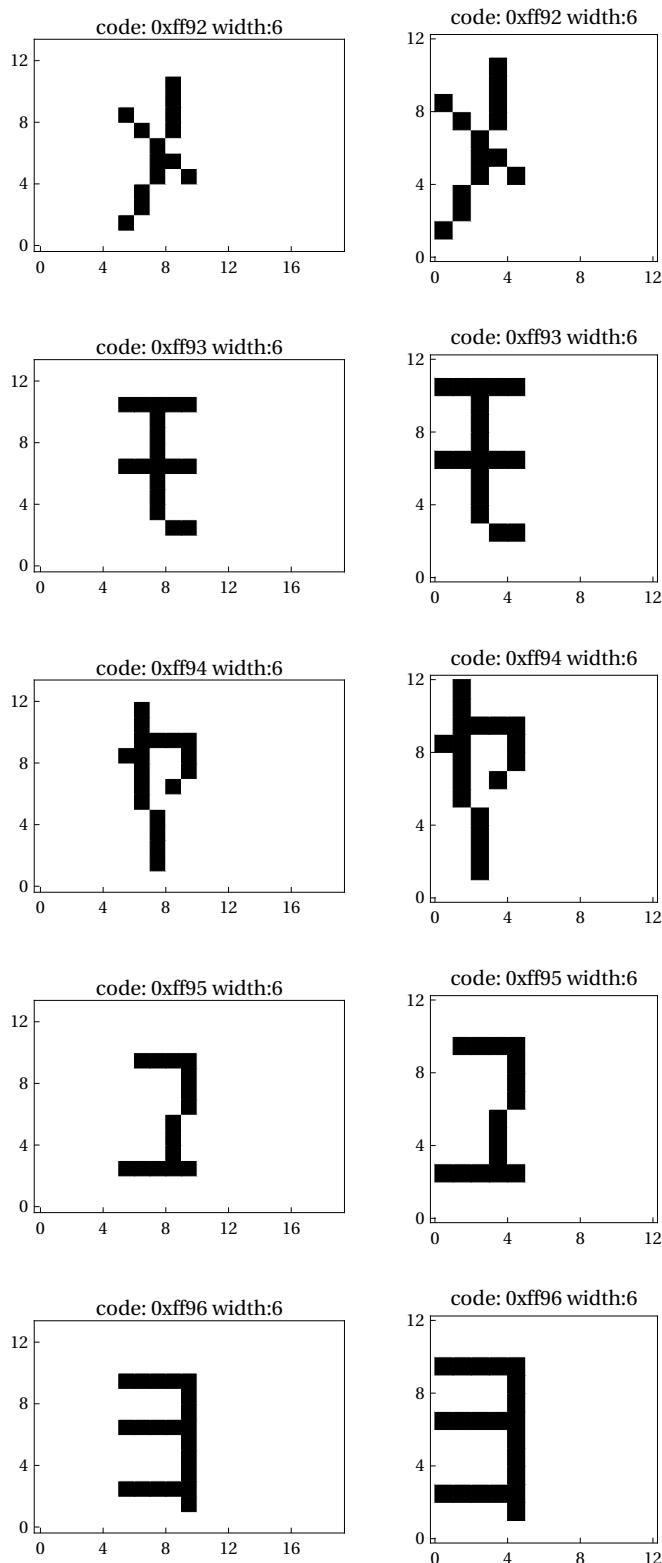


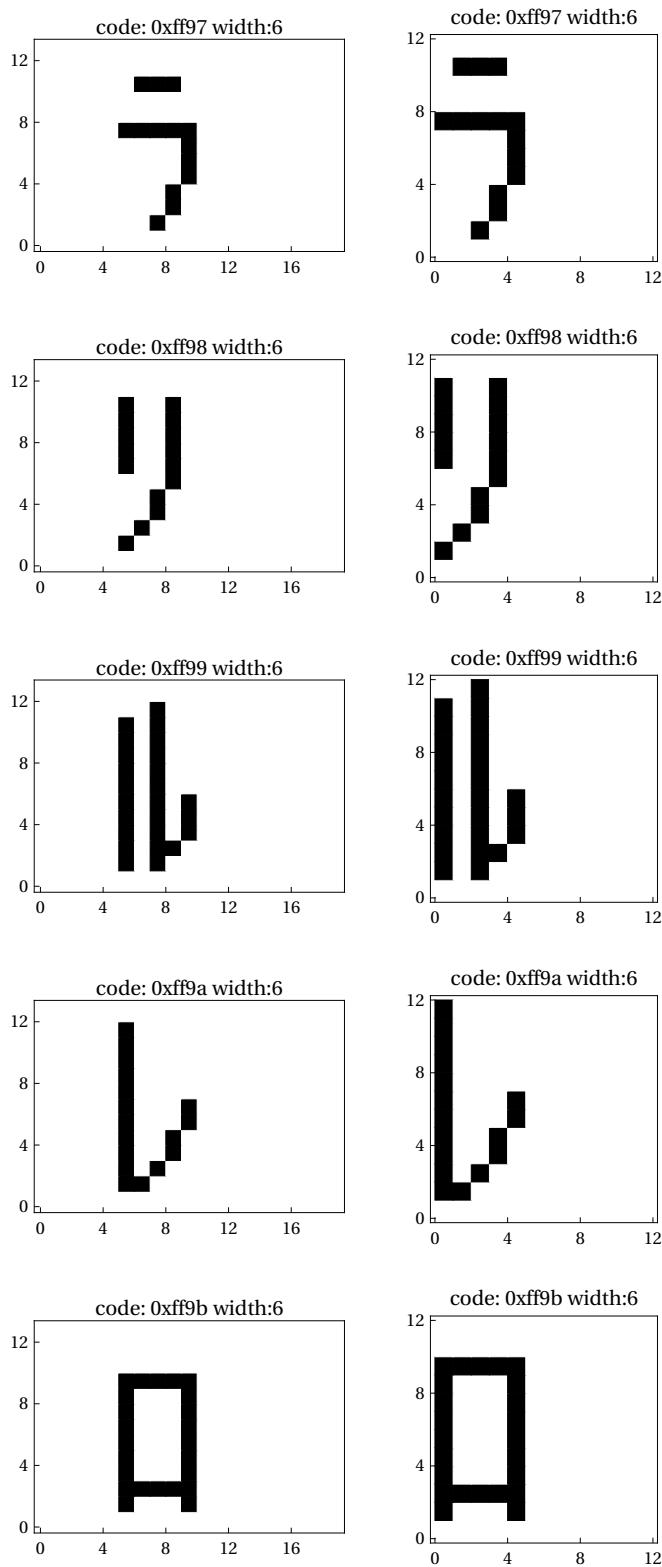


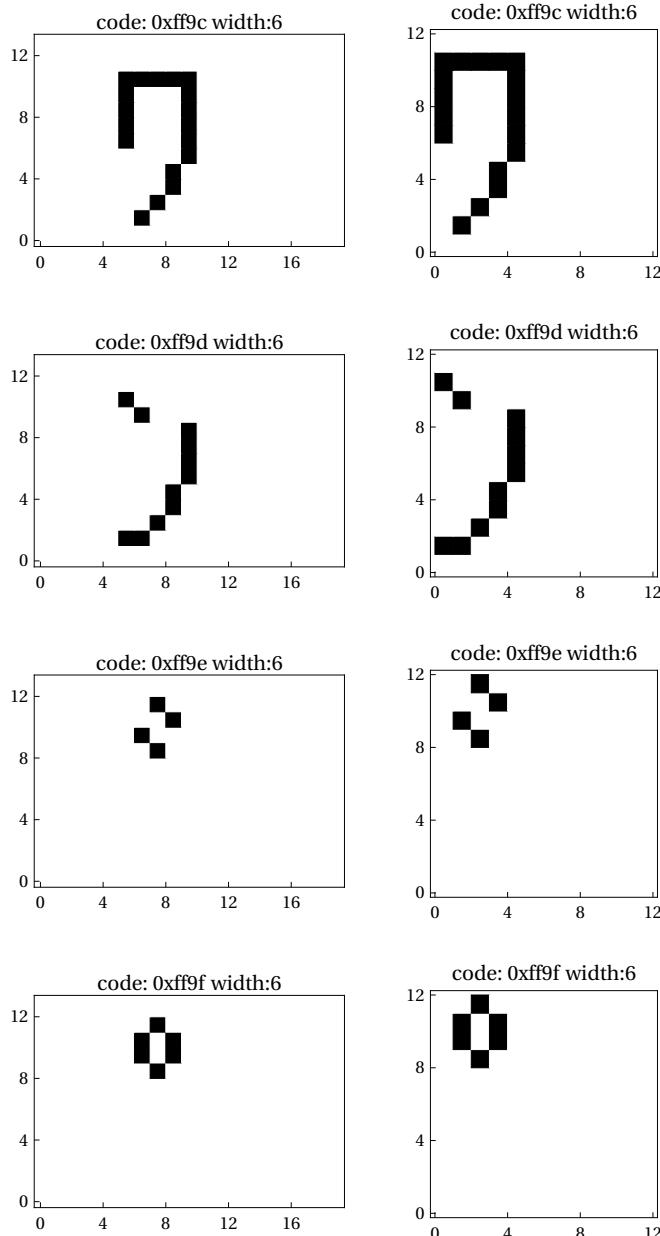






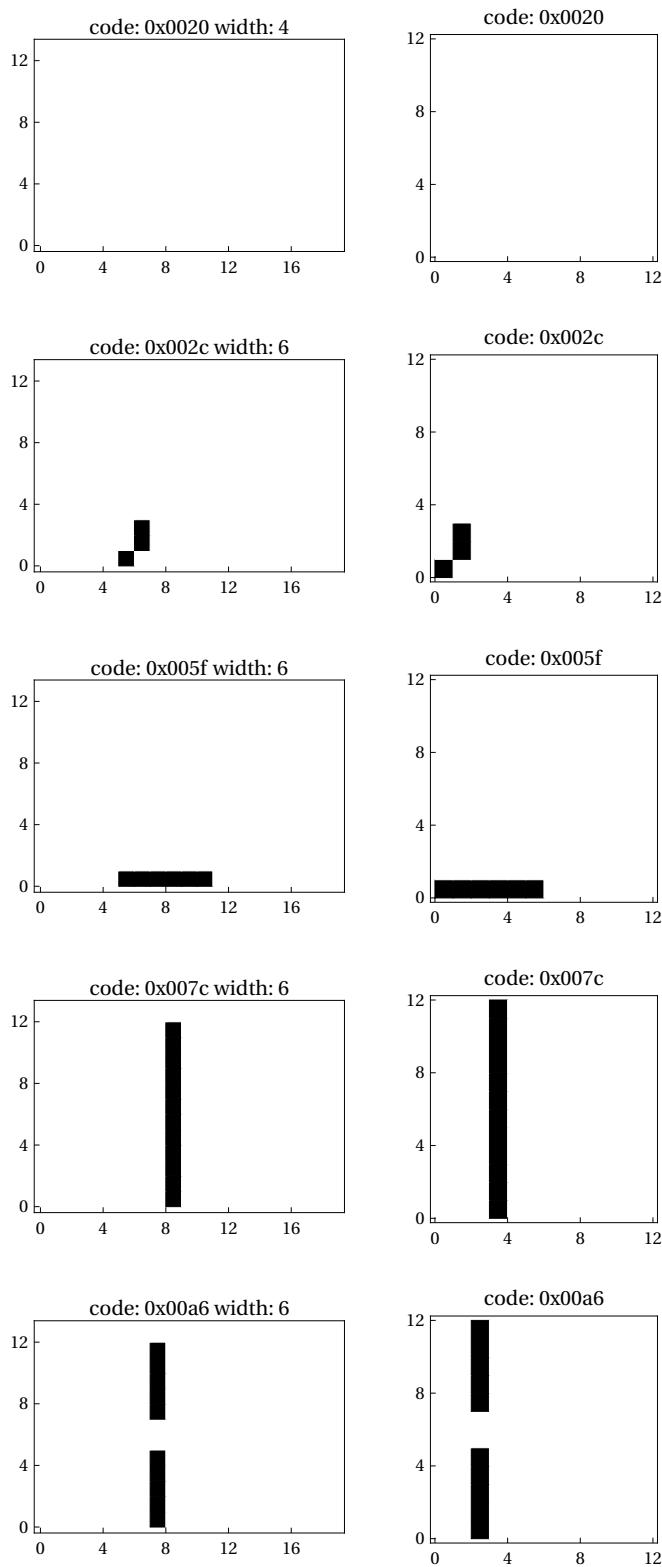


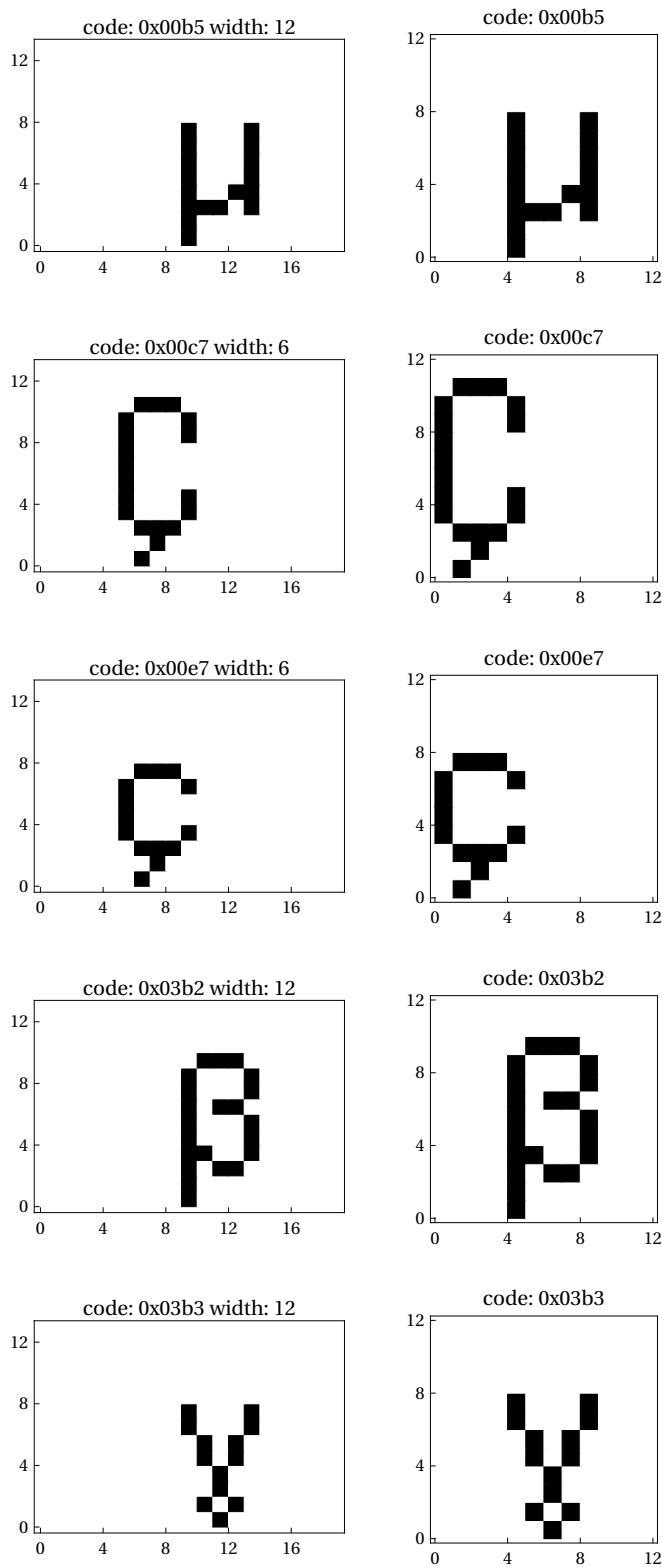


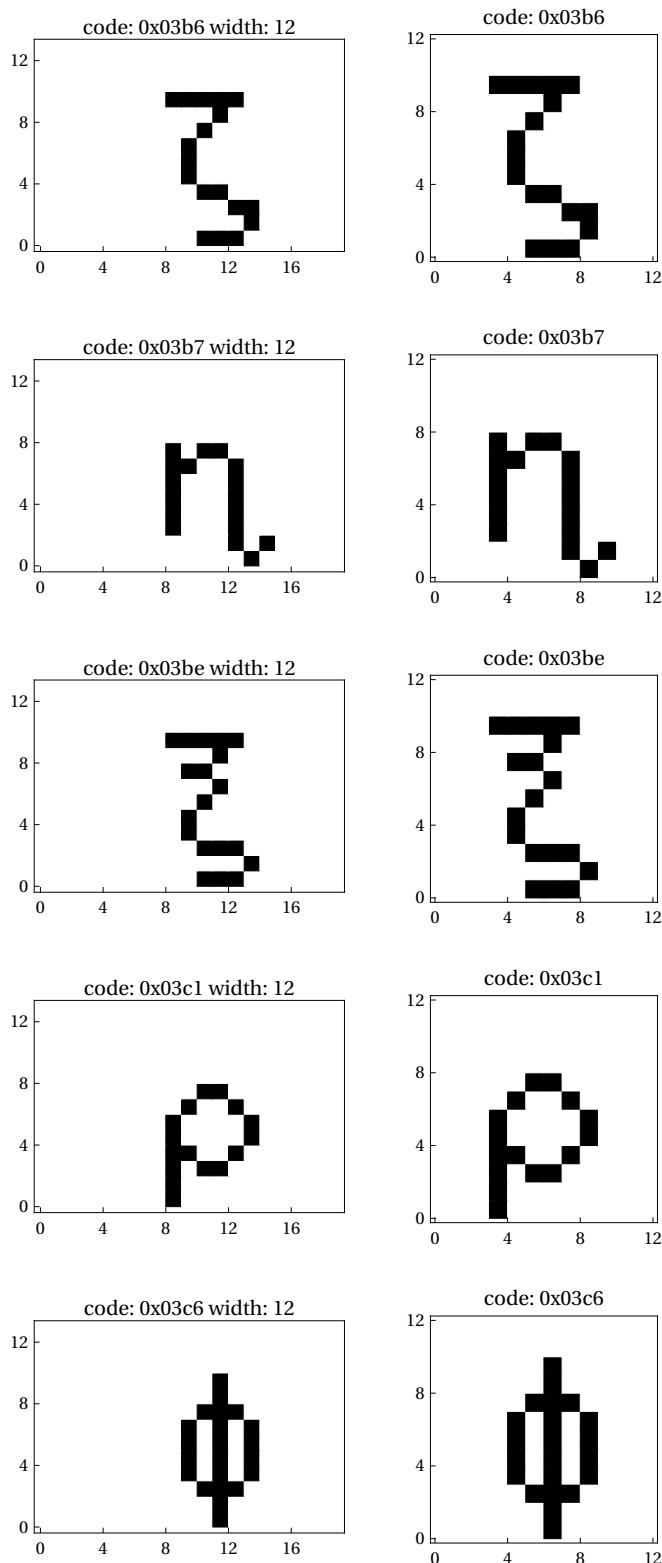


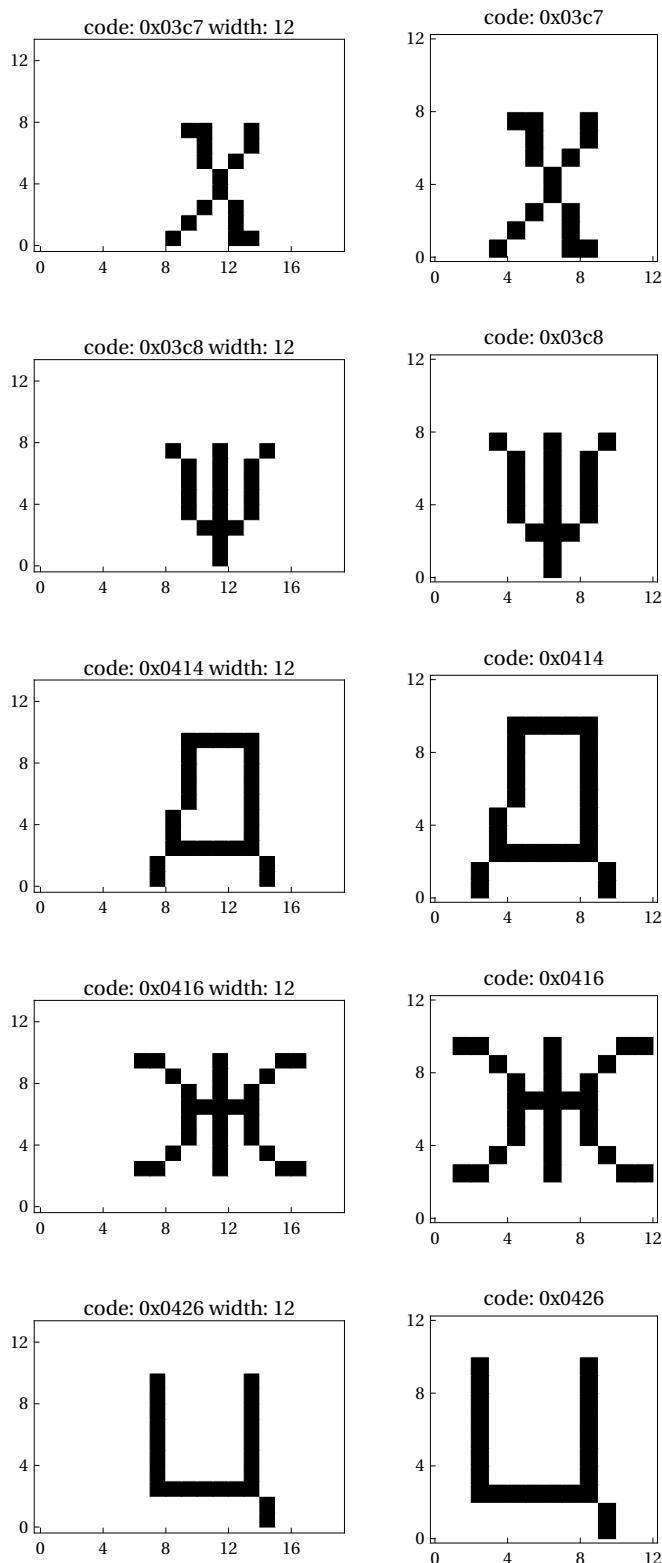
## ■ Bounding box outlier glyphs

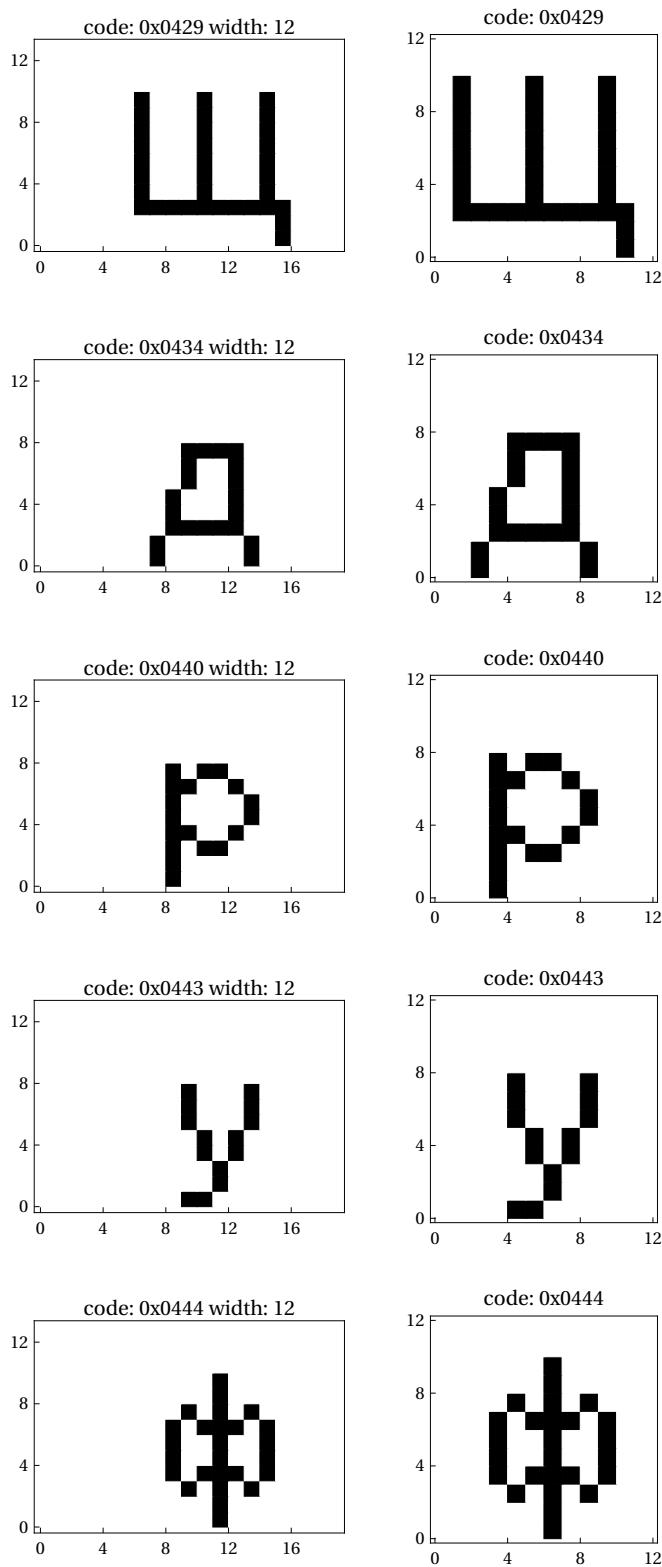
```
In[44]:= outliergraphics = MapThread[Graphics[Raster[Reverse[#1]],  
Frame -> True, AspectRatio -> Divide @@ Dimensions[#1],  
FrameTicks -> ({#, #, {}, {}} & [Range[0, 16, 4]]), PlotLabel -> ("code: 0x" <>  
IntegerString[ToExpression[#2], 16, 4] <> " width: " <> ToString[#3])] &,  
#\[Outliers]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

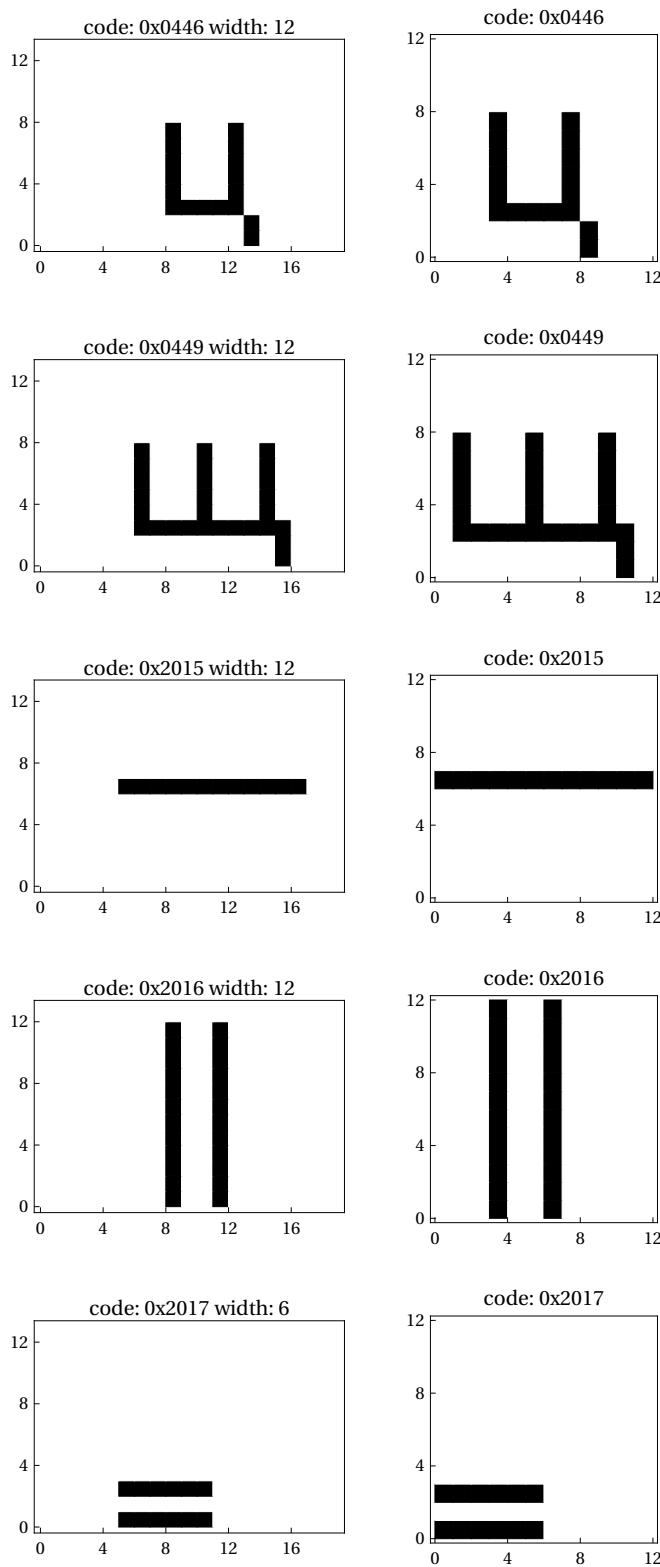


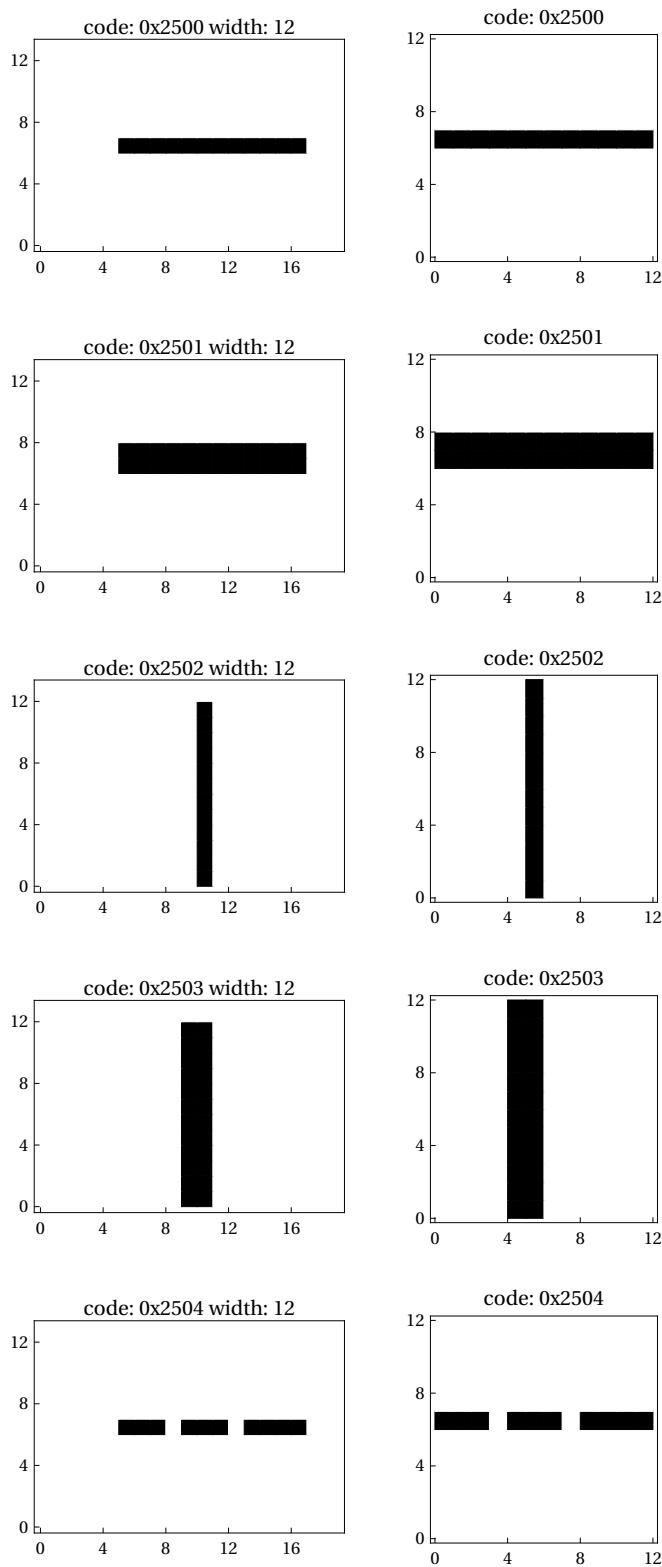


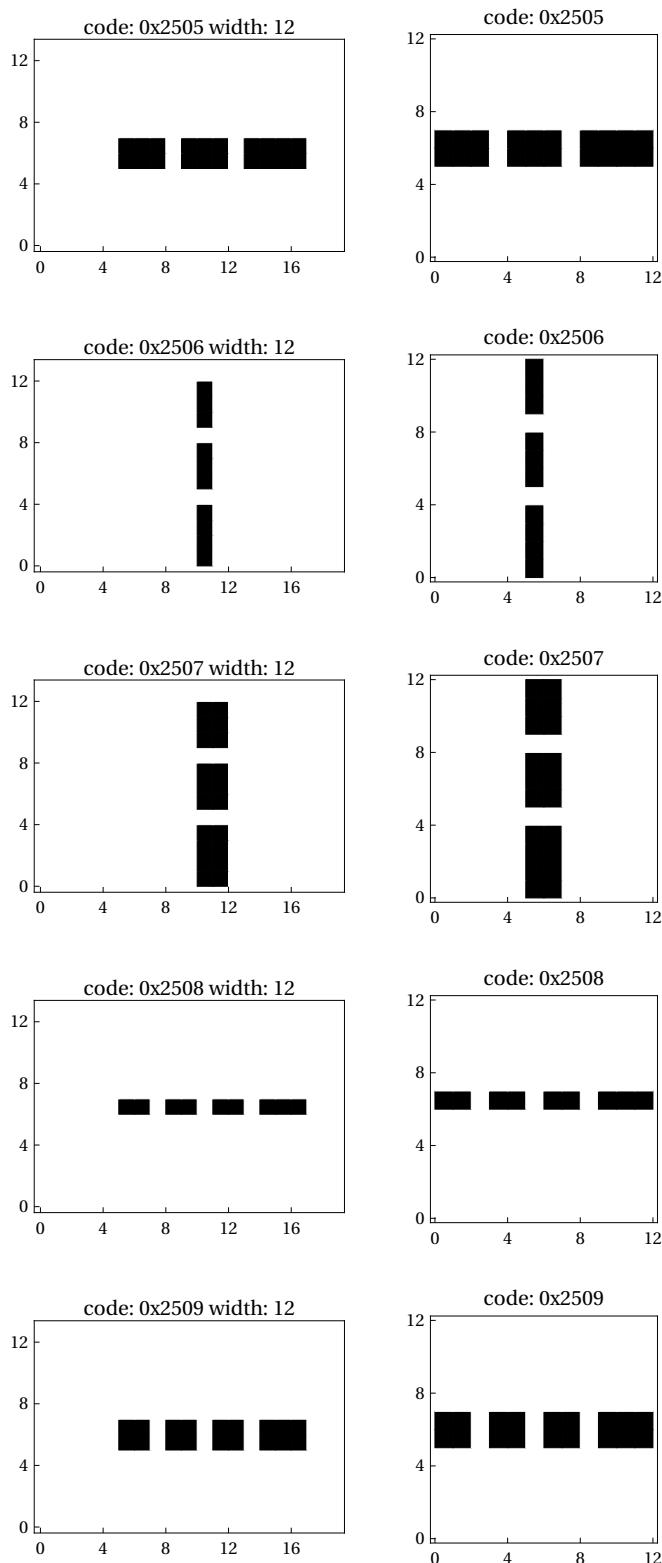


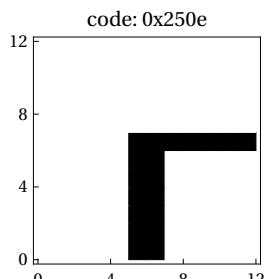
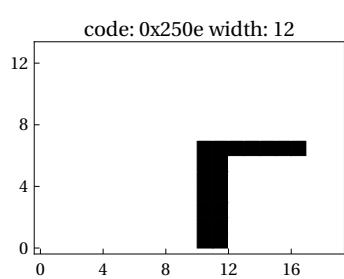
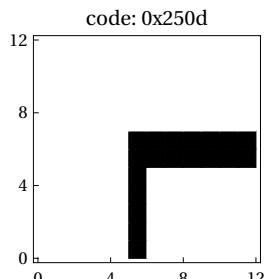
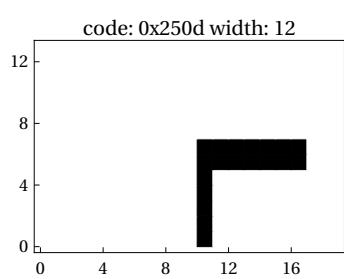
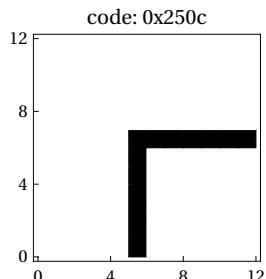
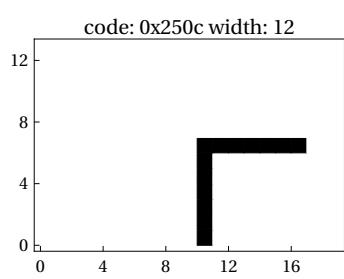
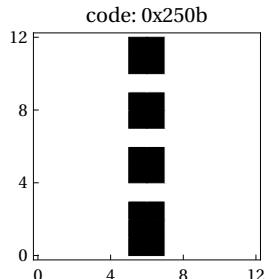
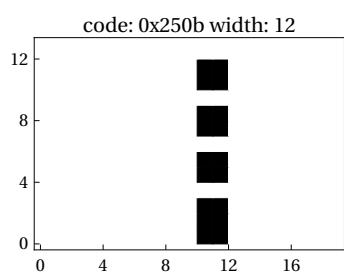
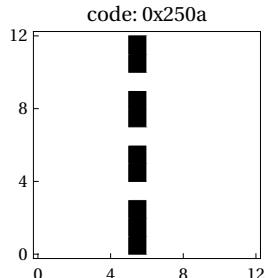
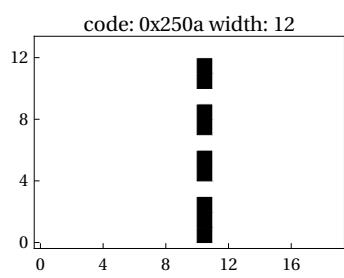


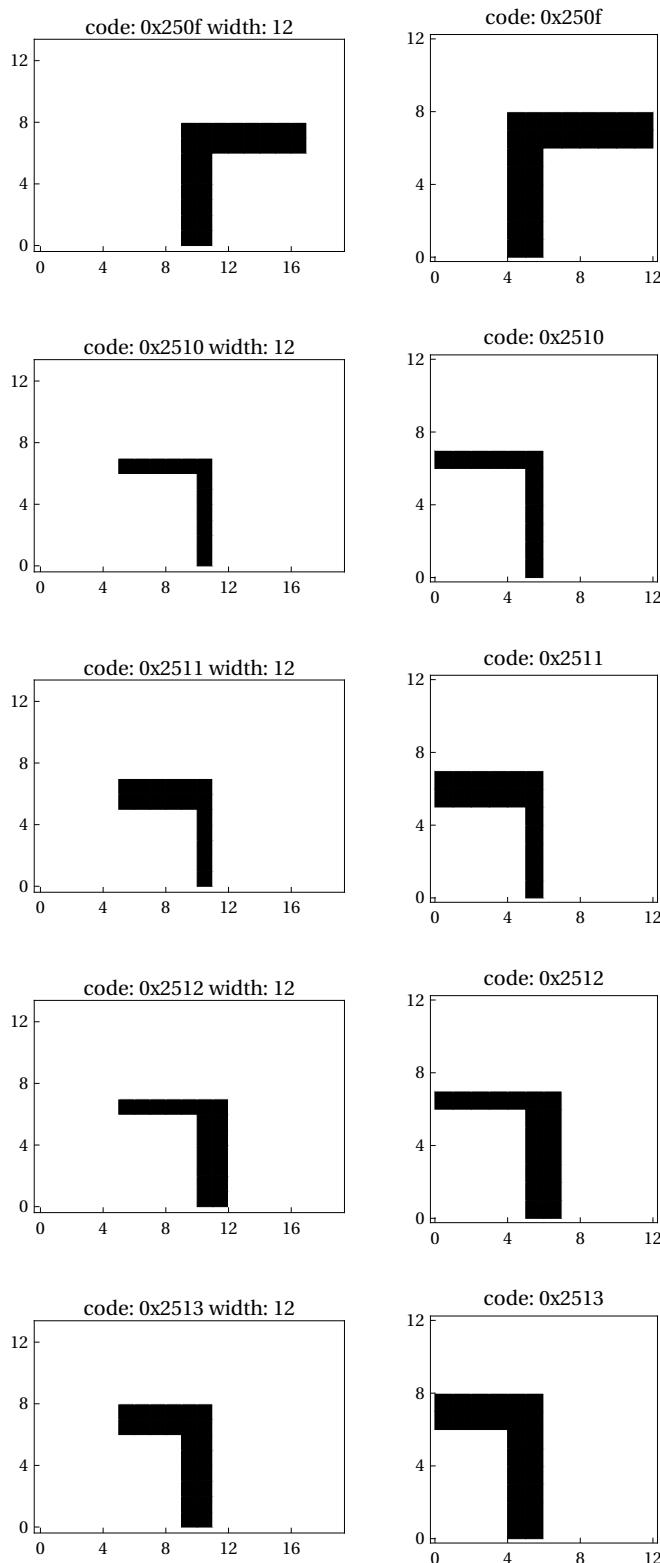


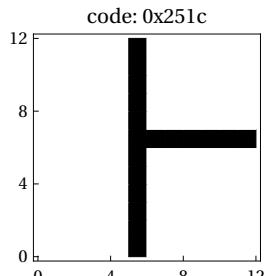
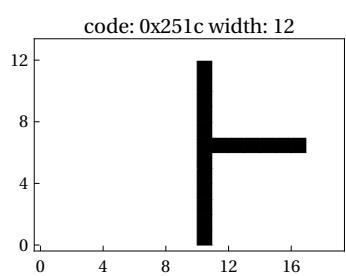
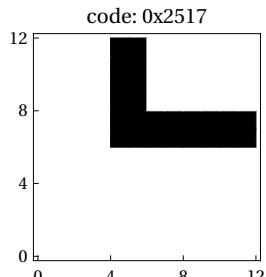
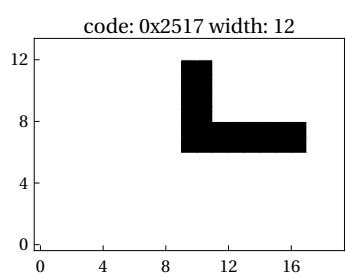
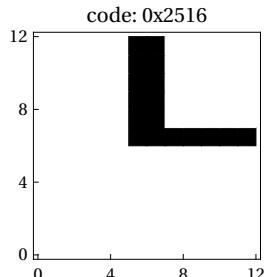
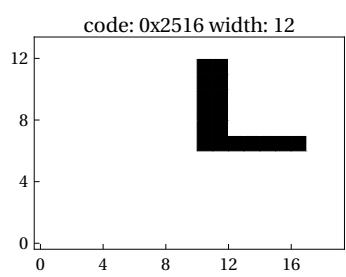
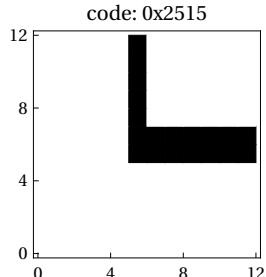
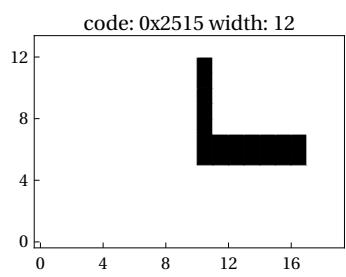
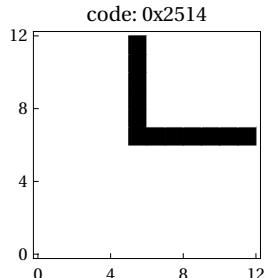
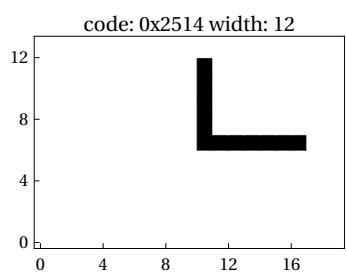


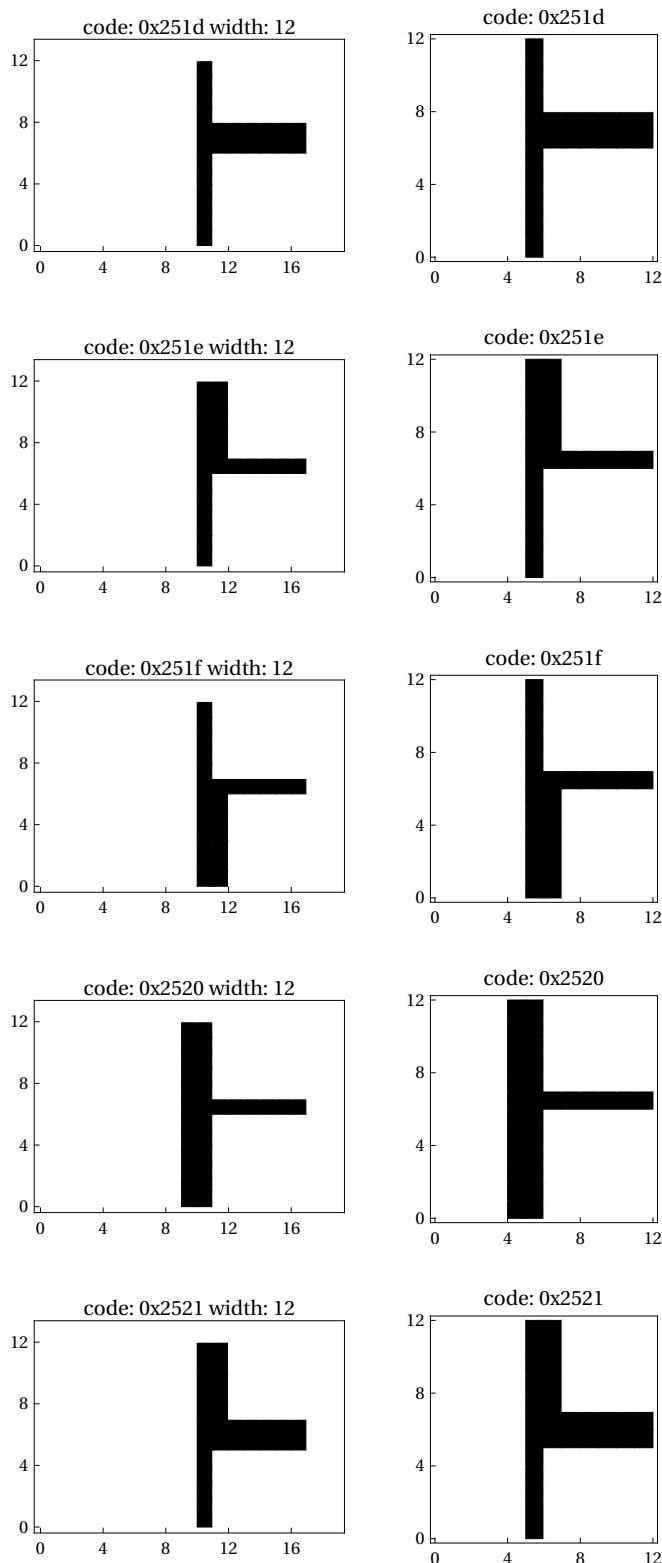


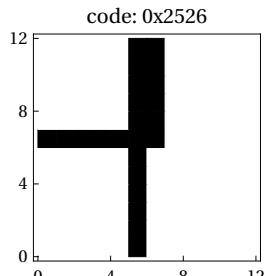
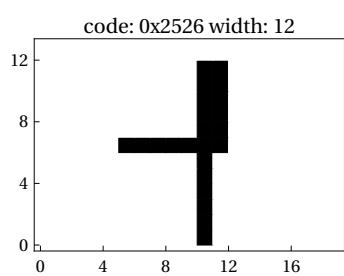
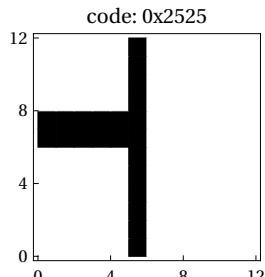
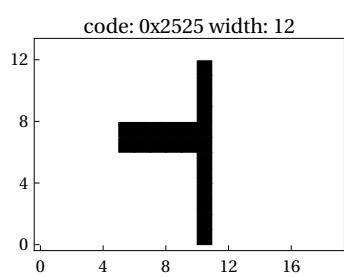
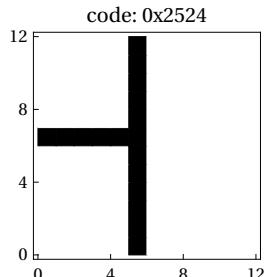
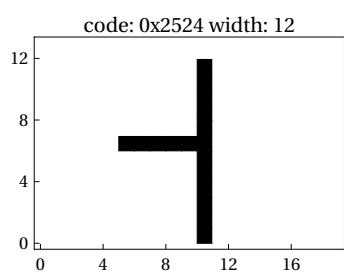
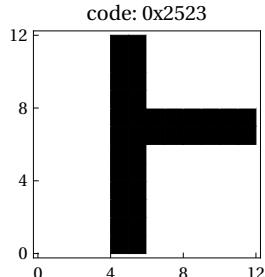
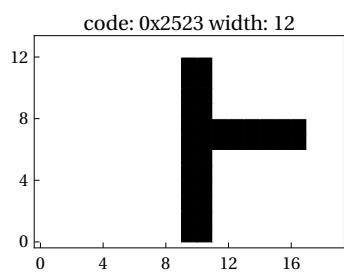
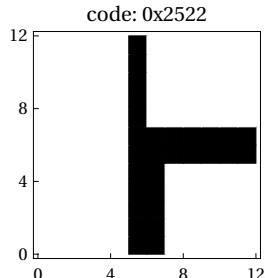
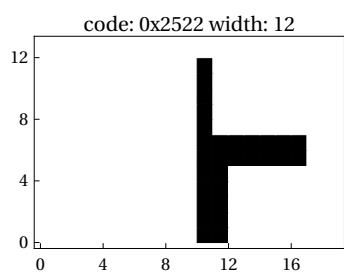


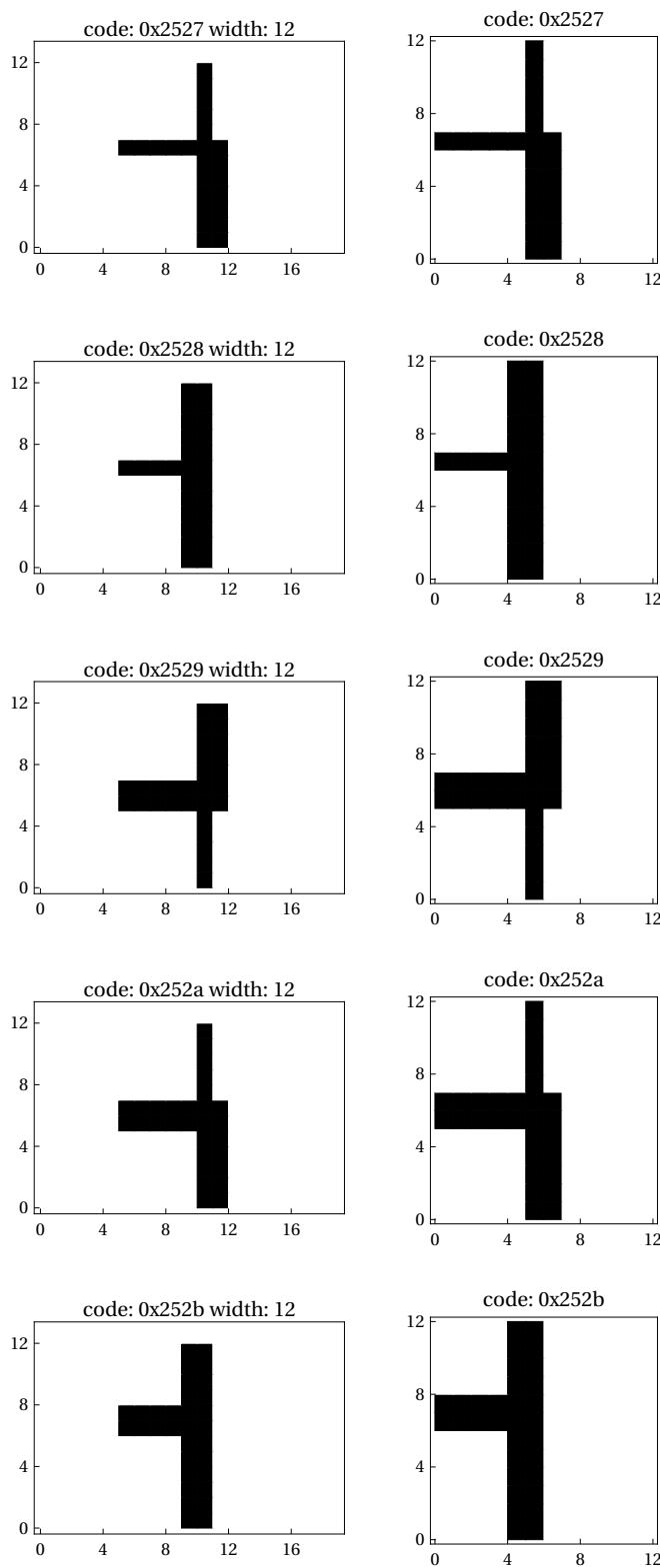


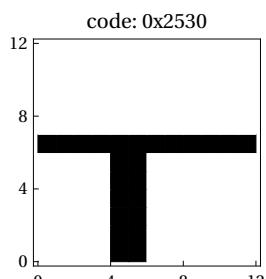
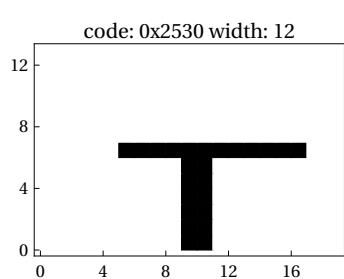
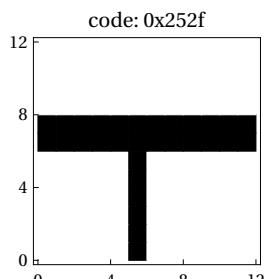
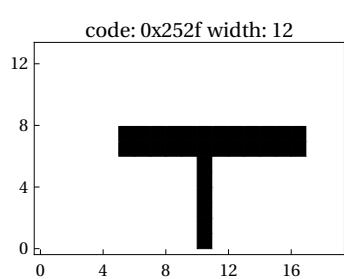
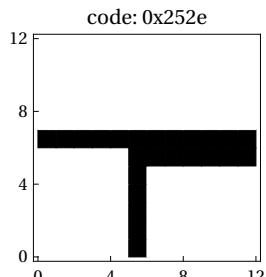
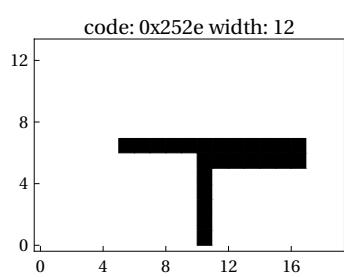
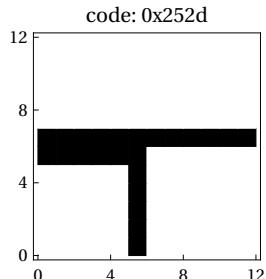
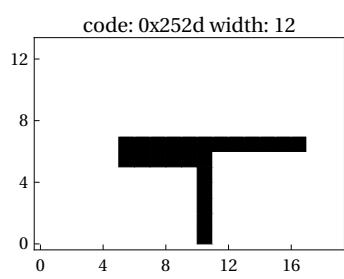
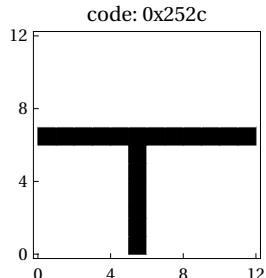
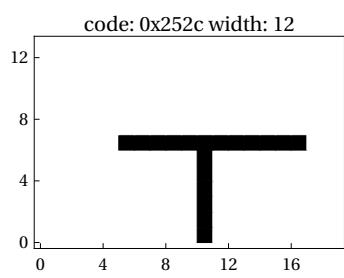


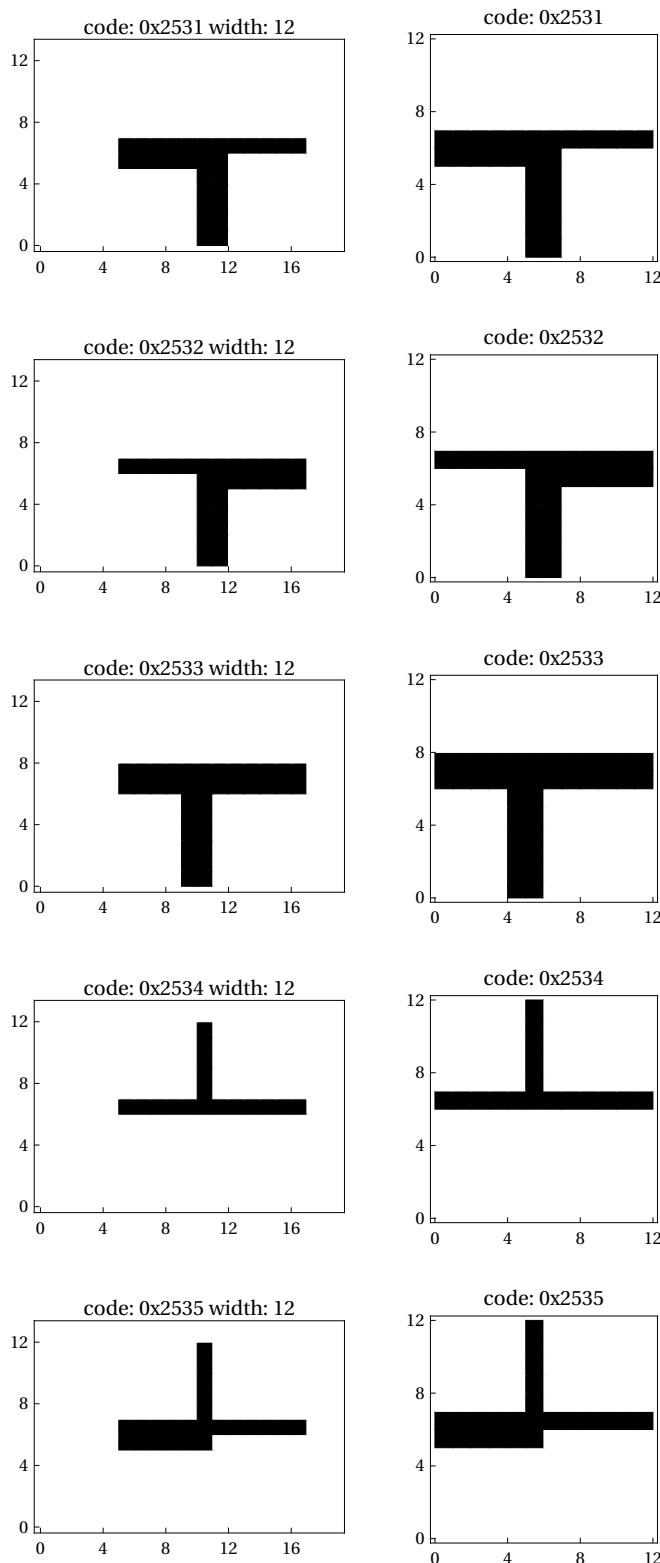


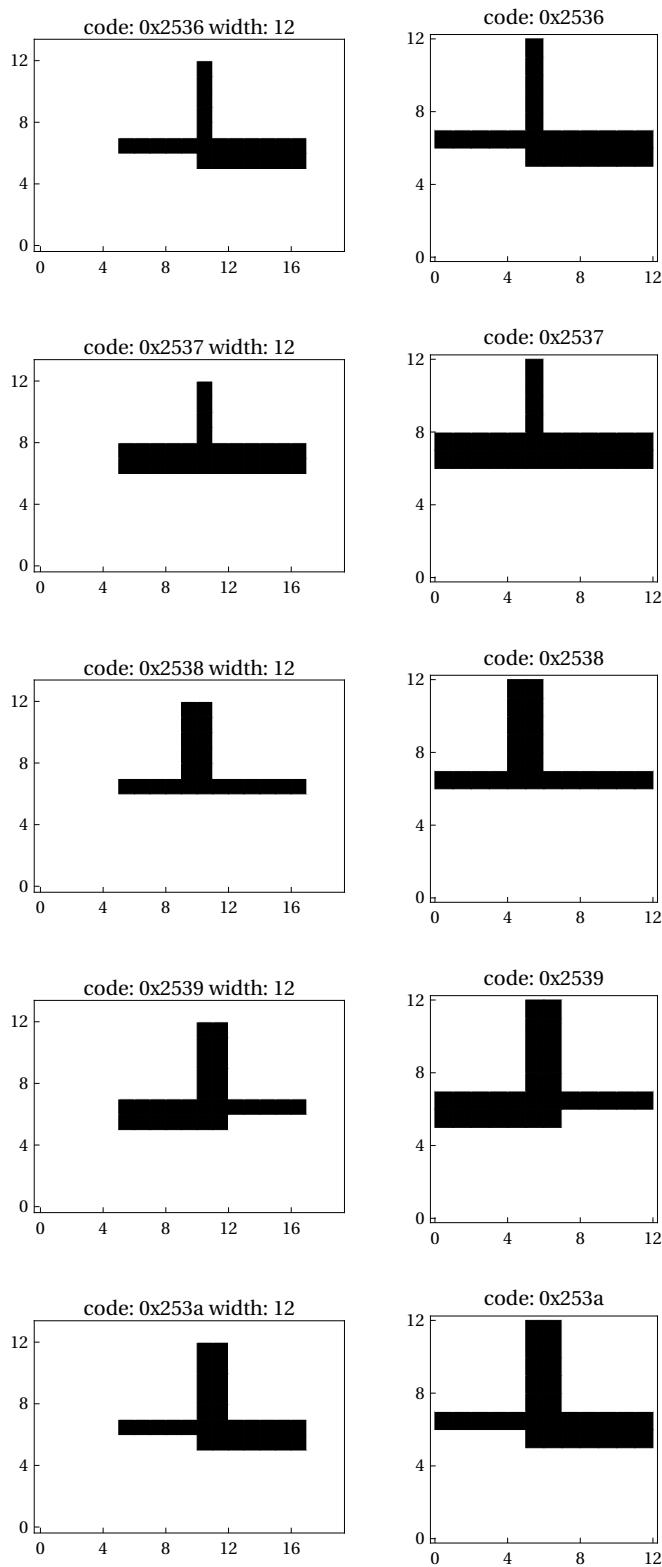


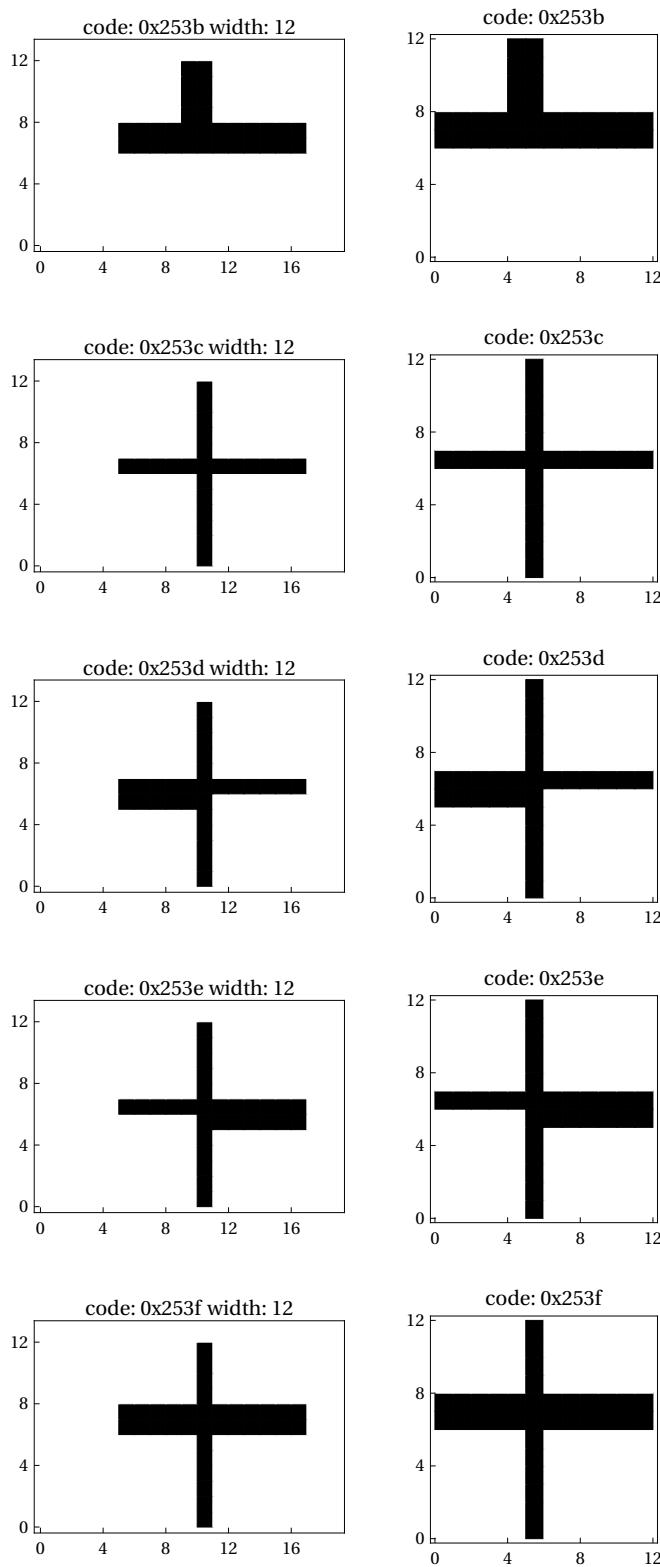


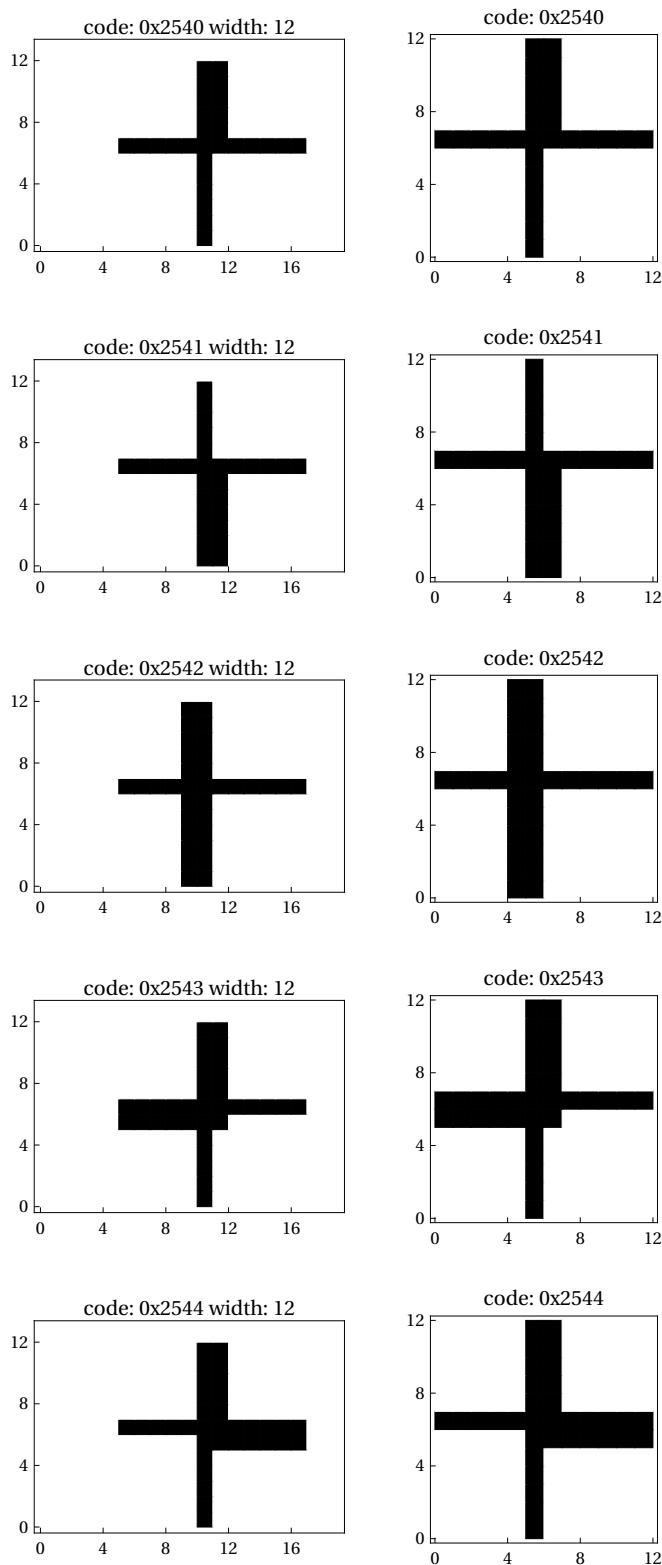


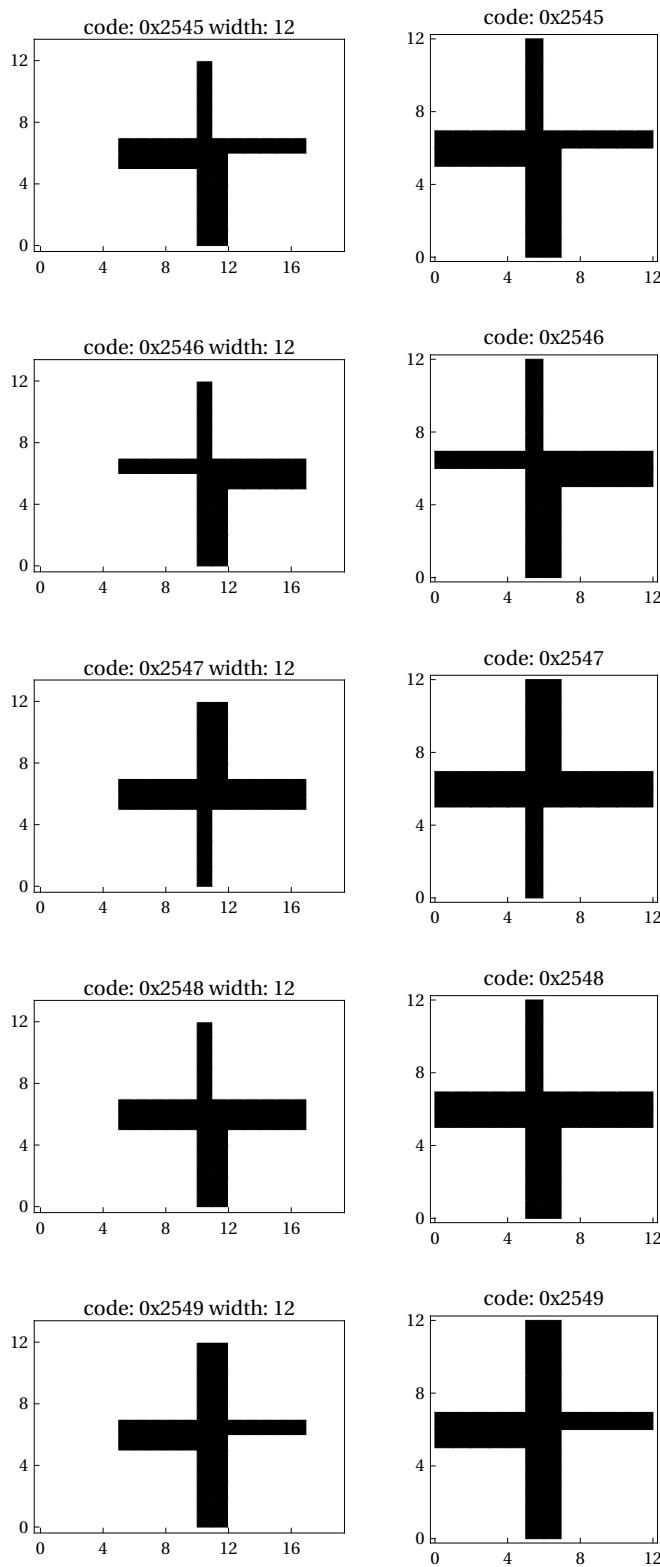


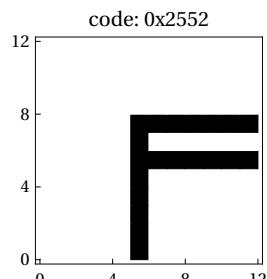
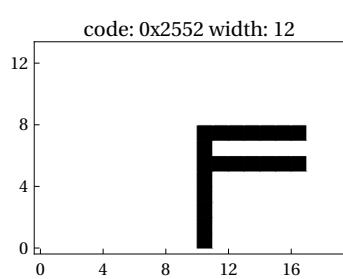
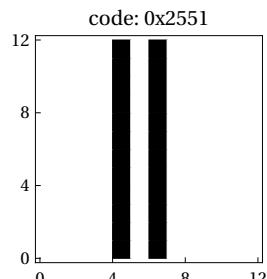
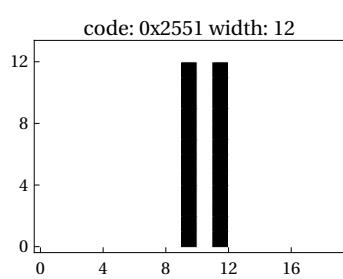
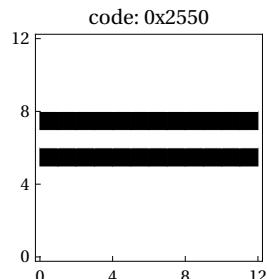
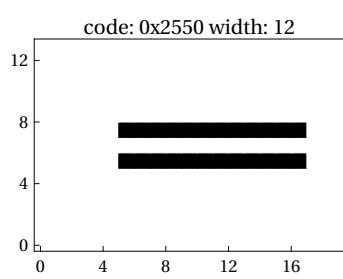
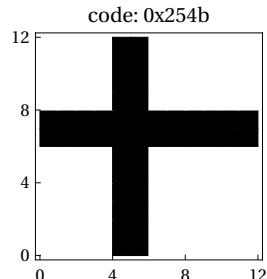
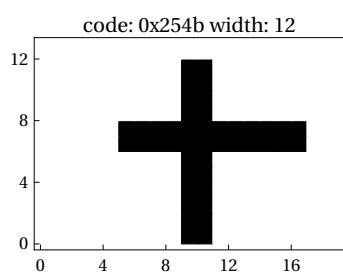
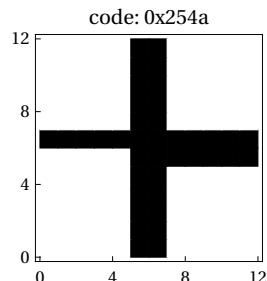
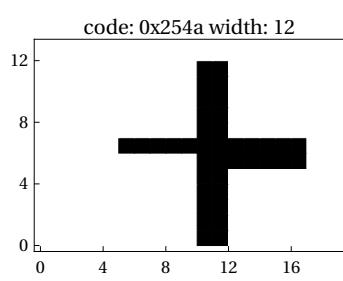


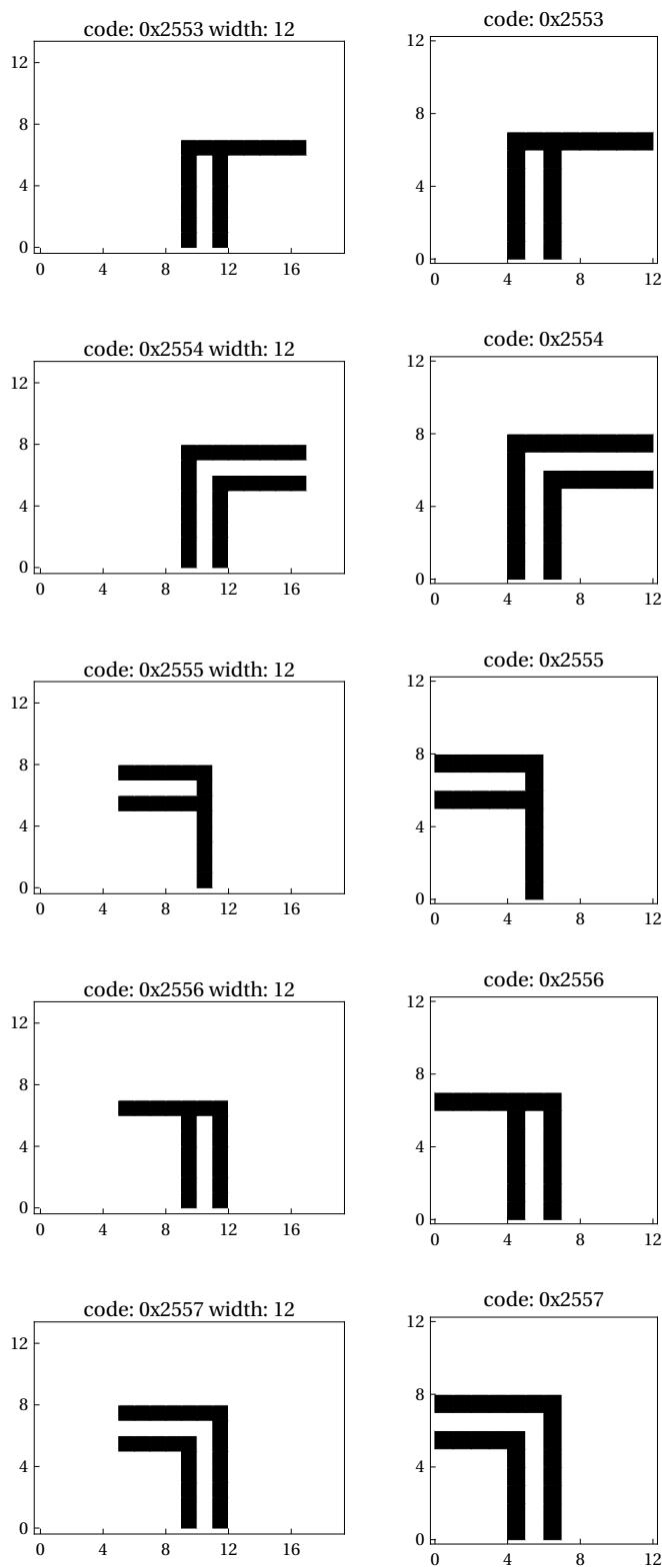


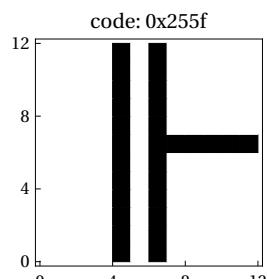
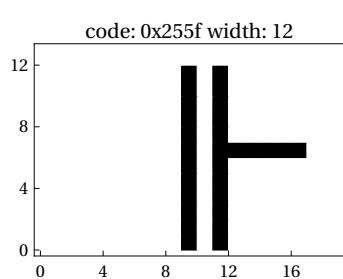
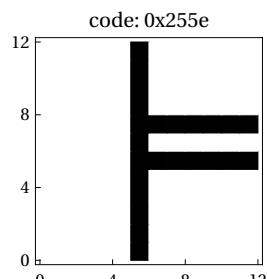
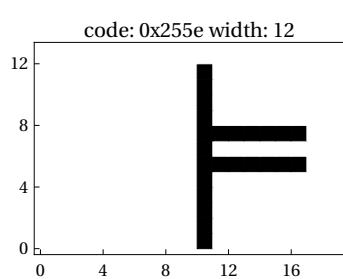
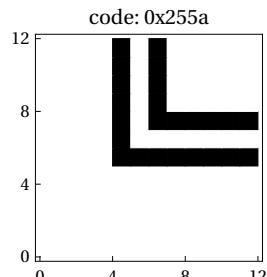
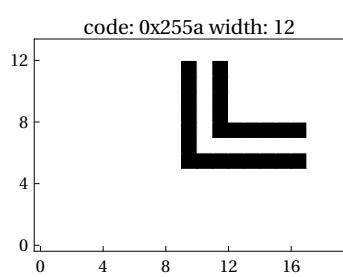
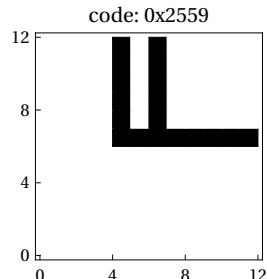
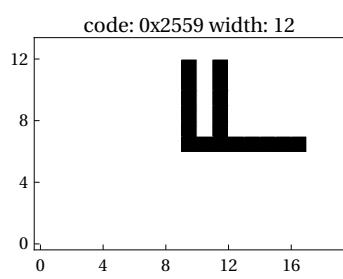
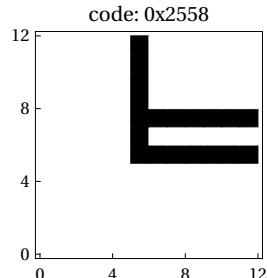
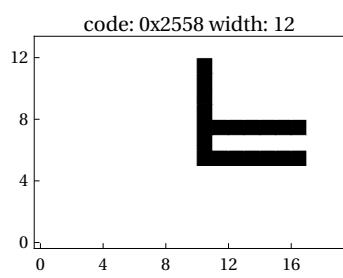


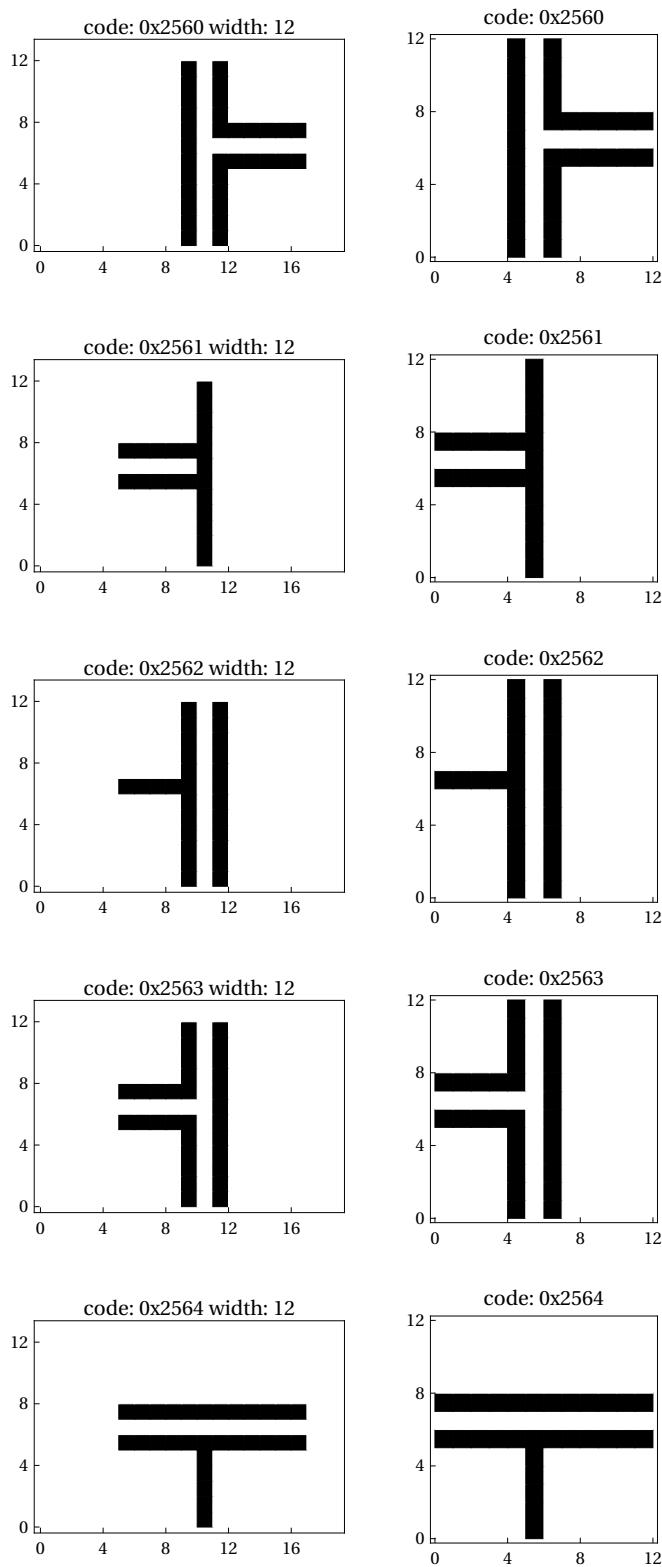


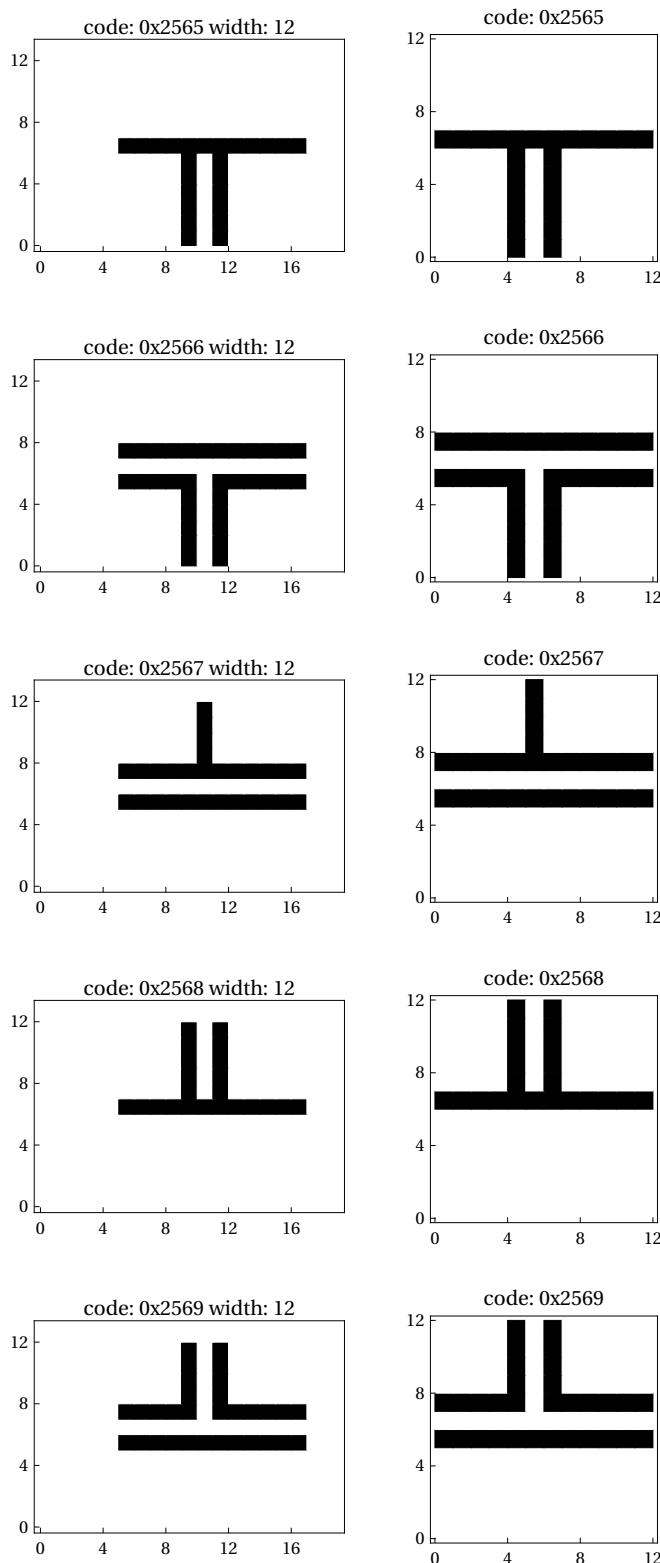


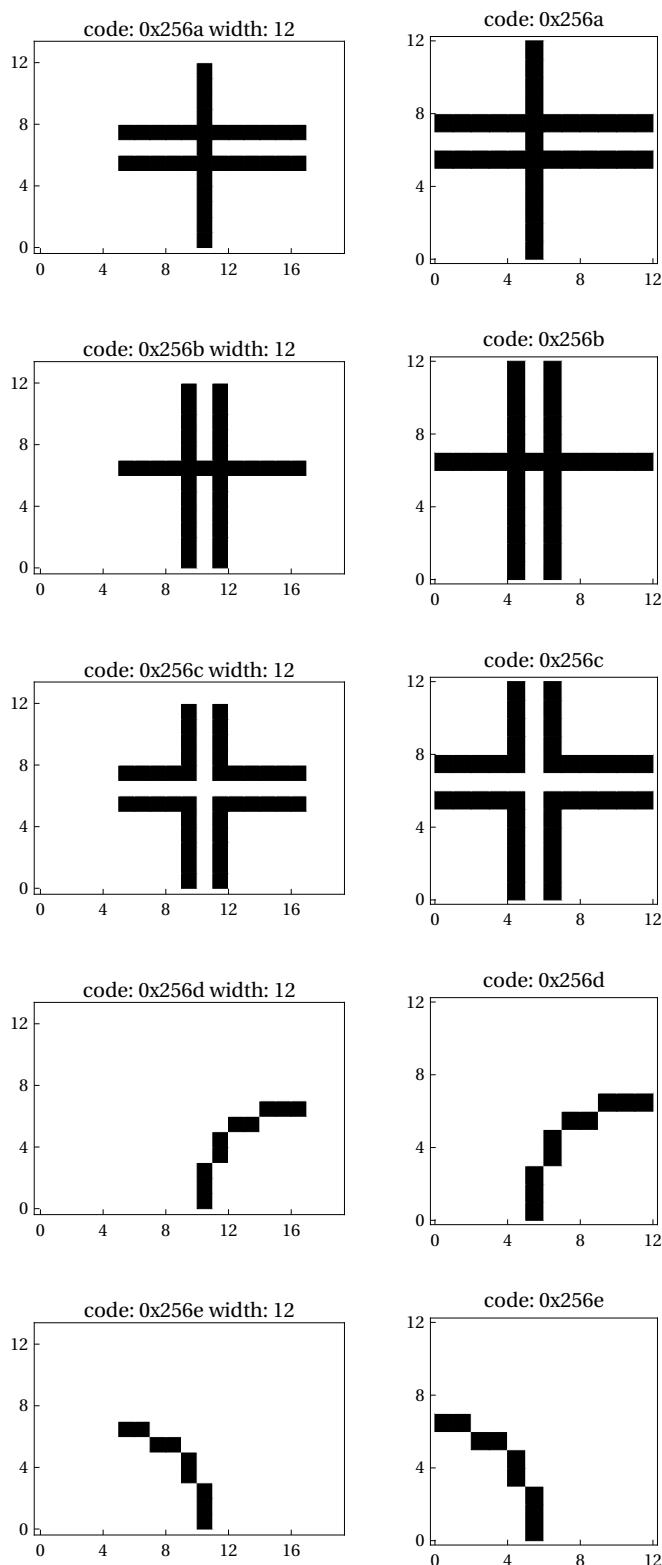


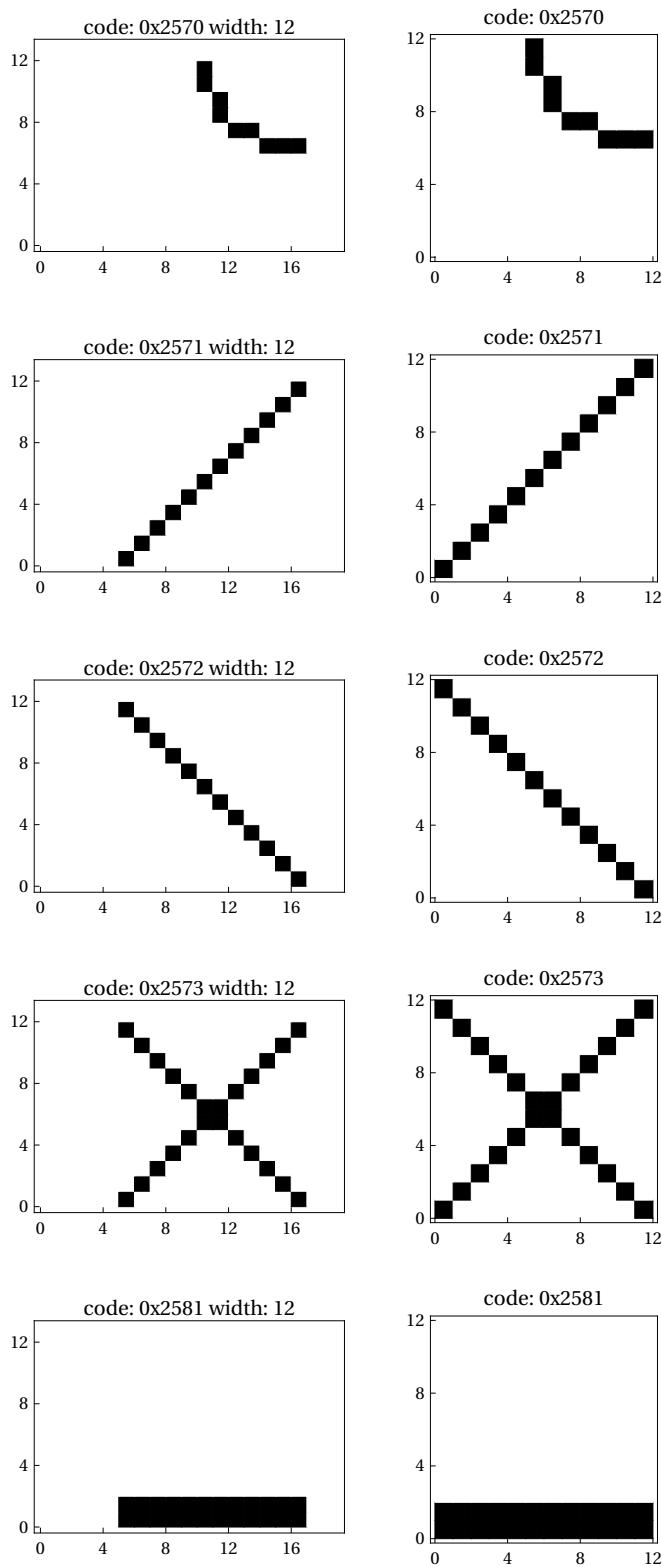


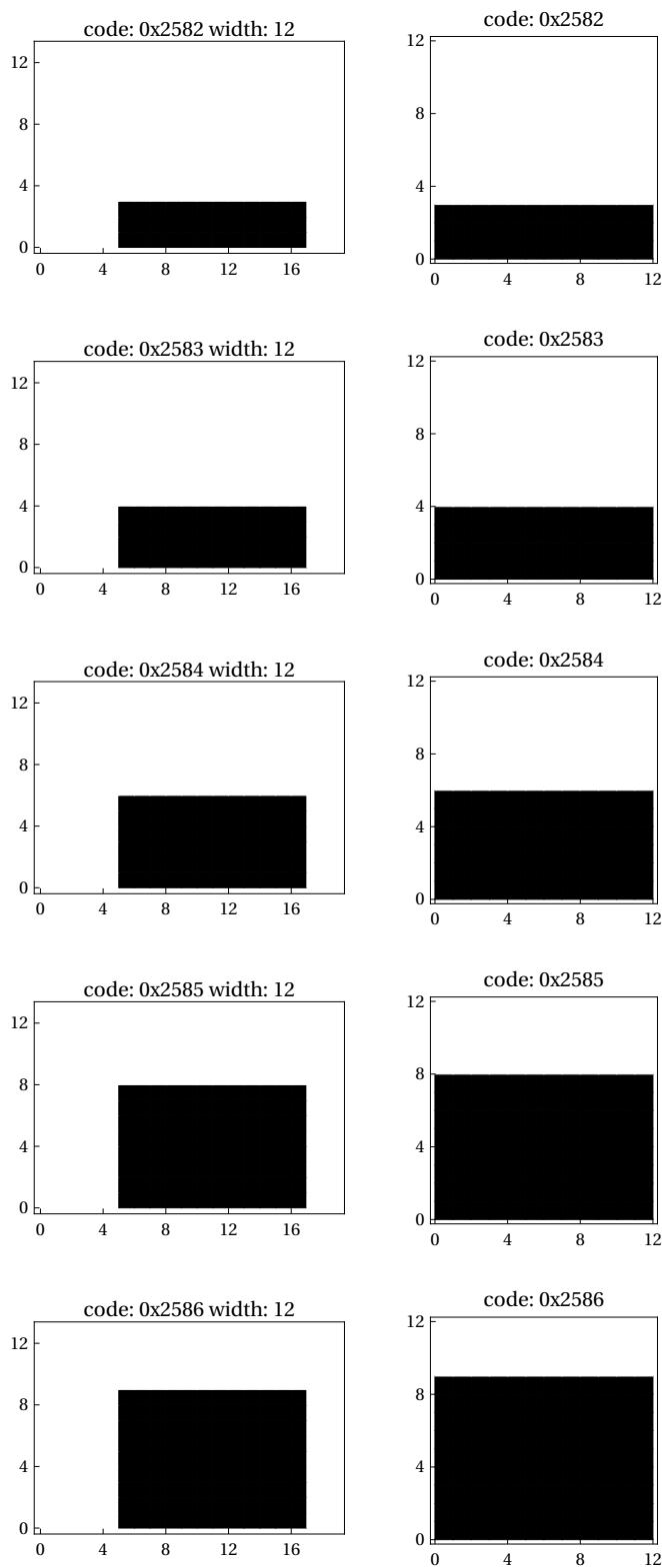


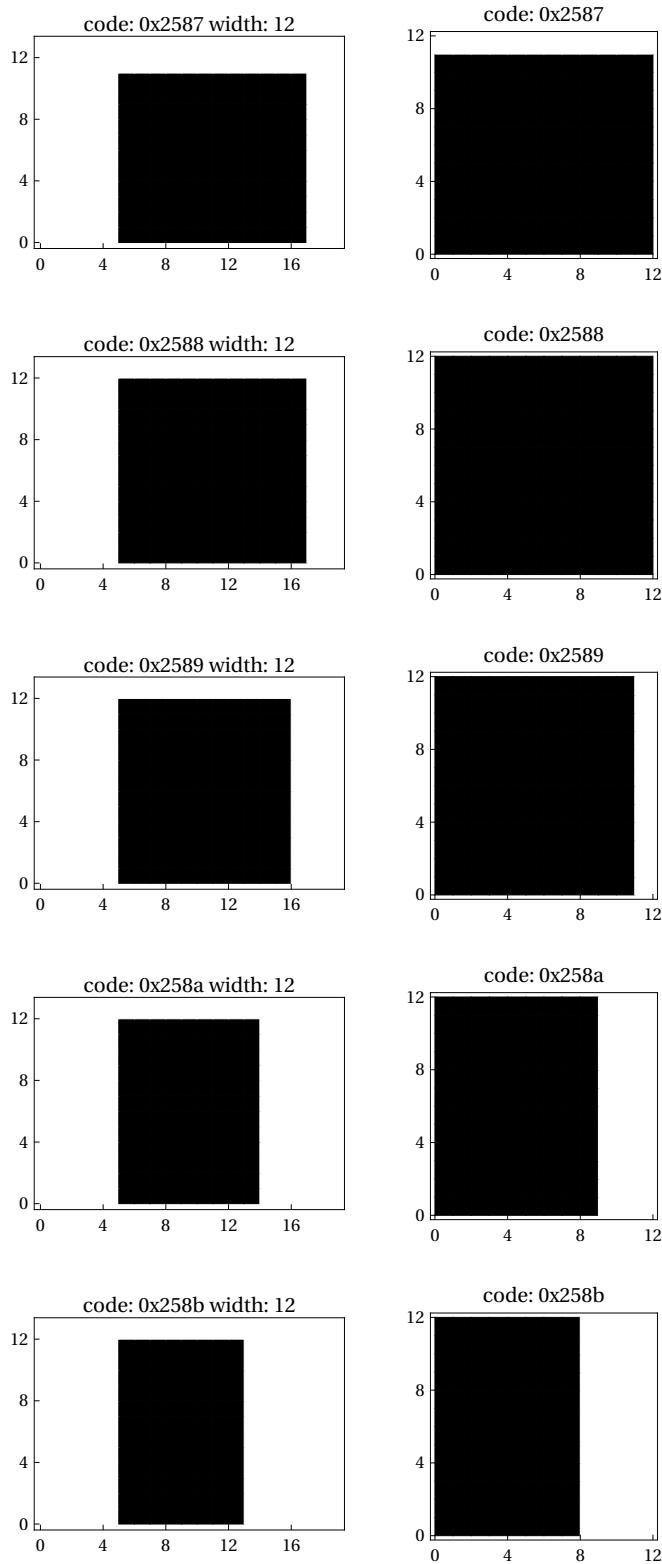


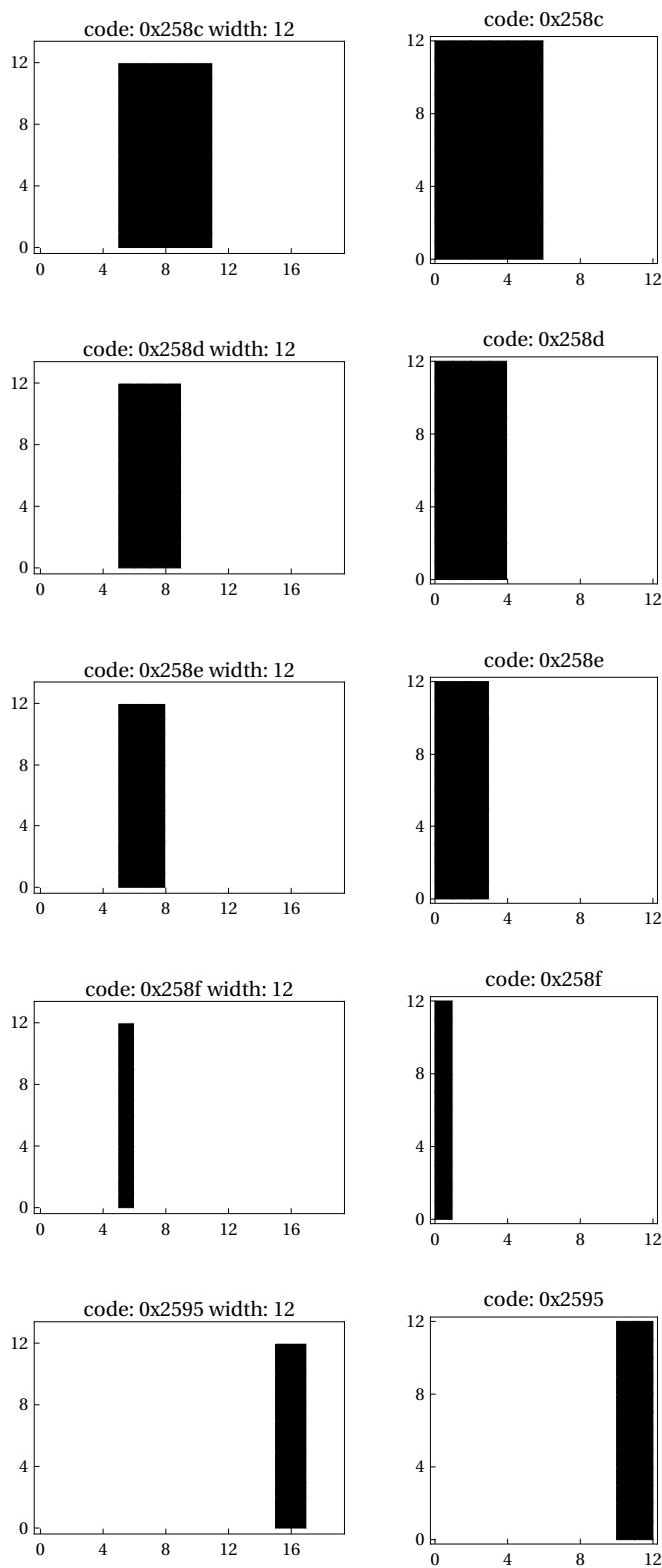


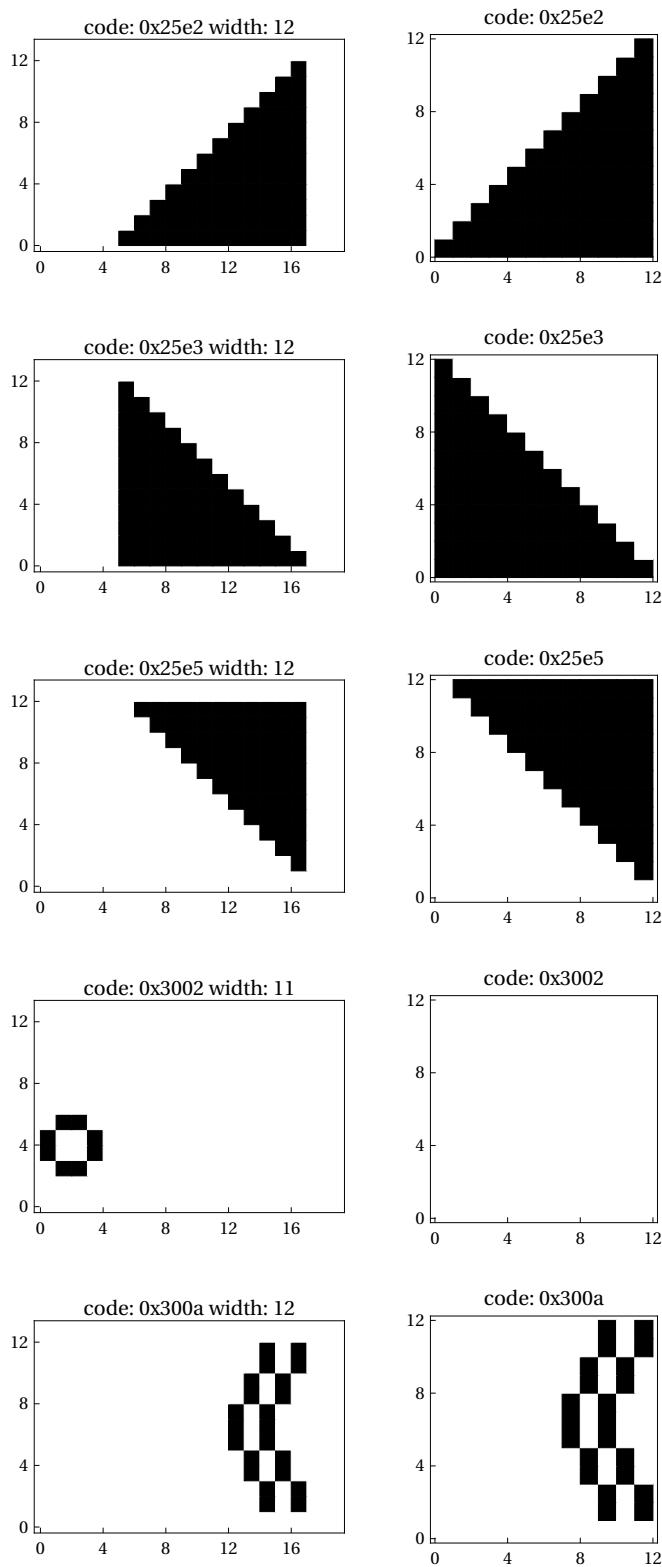


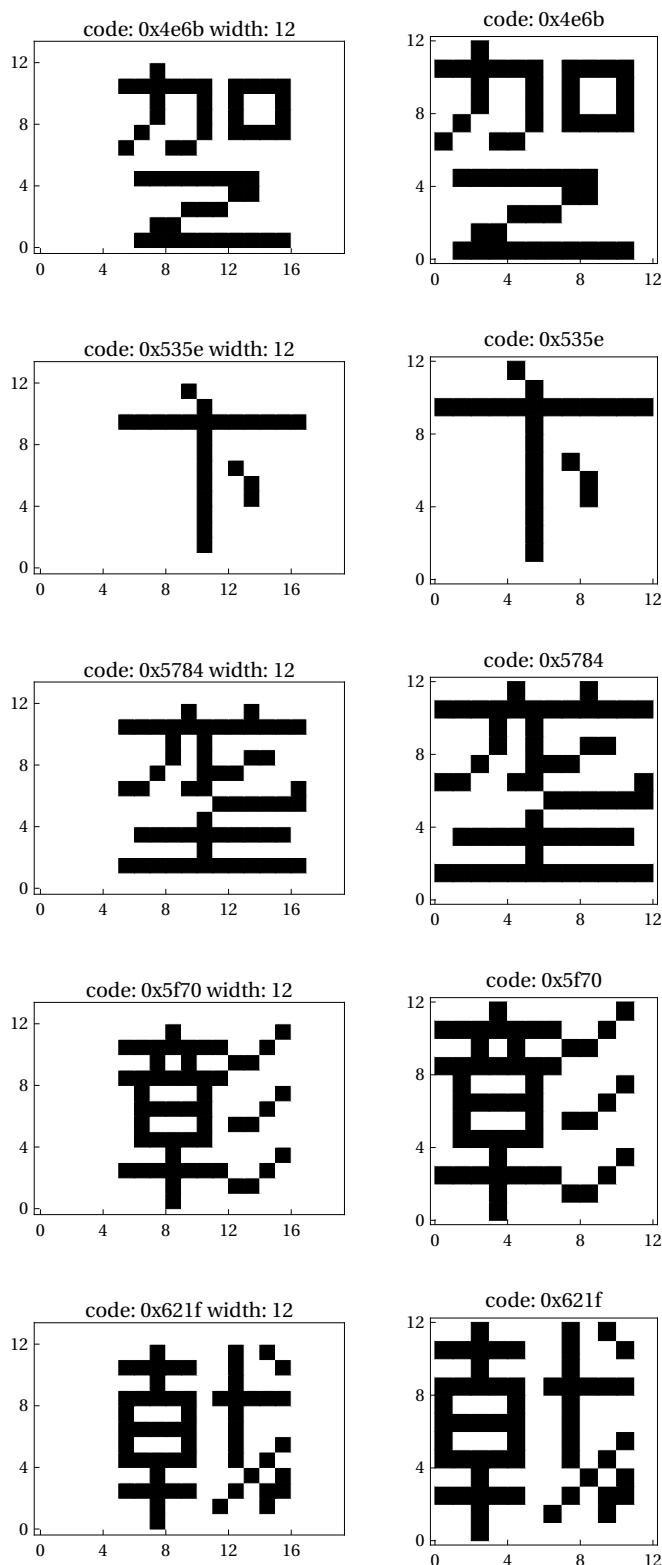


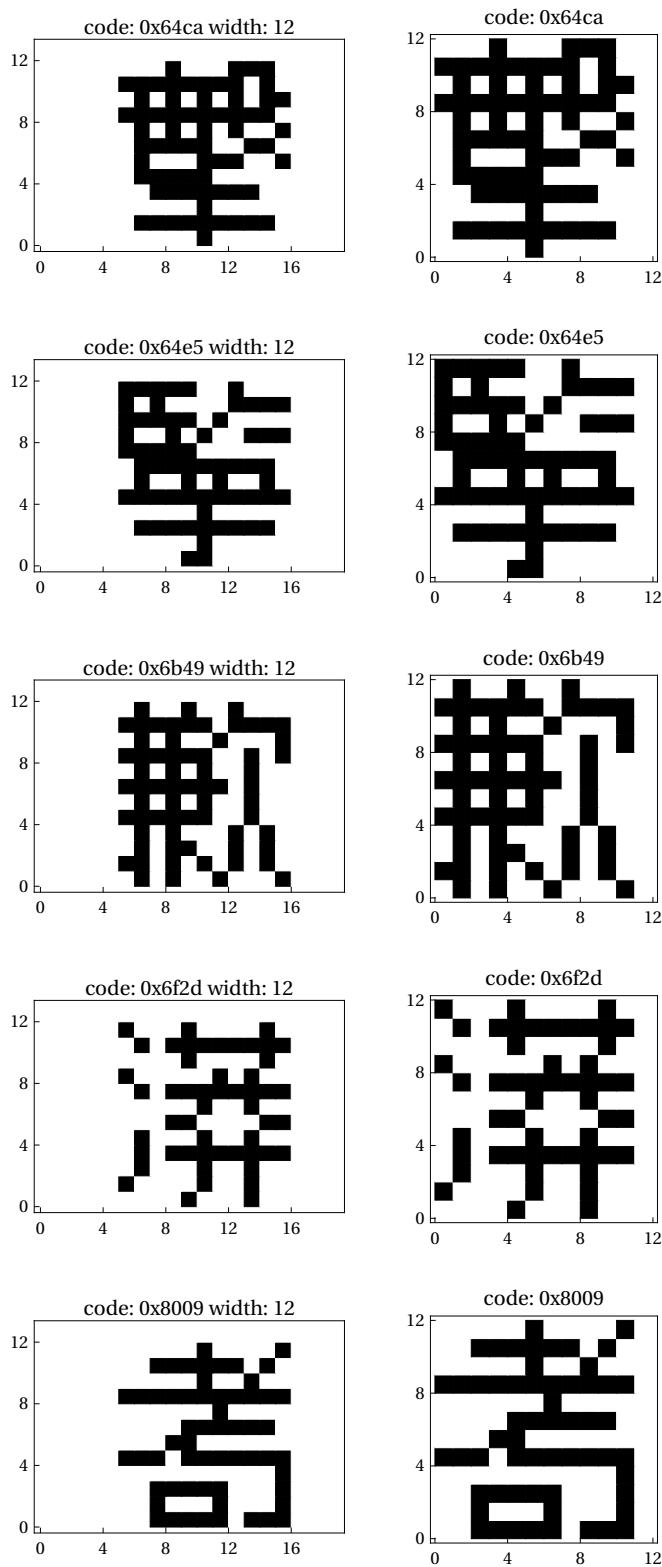


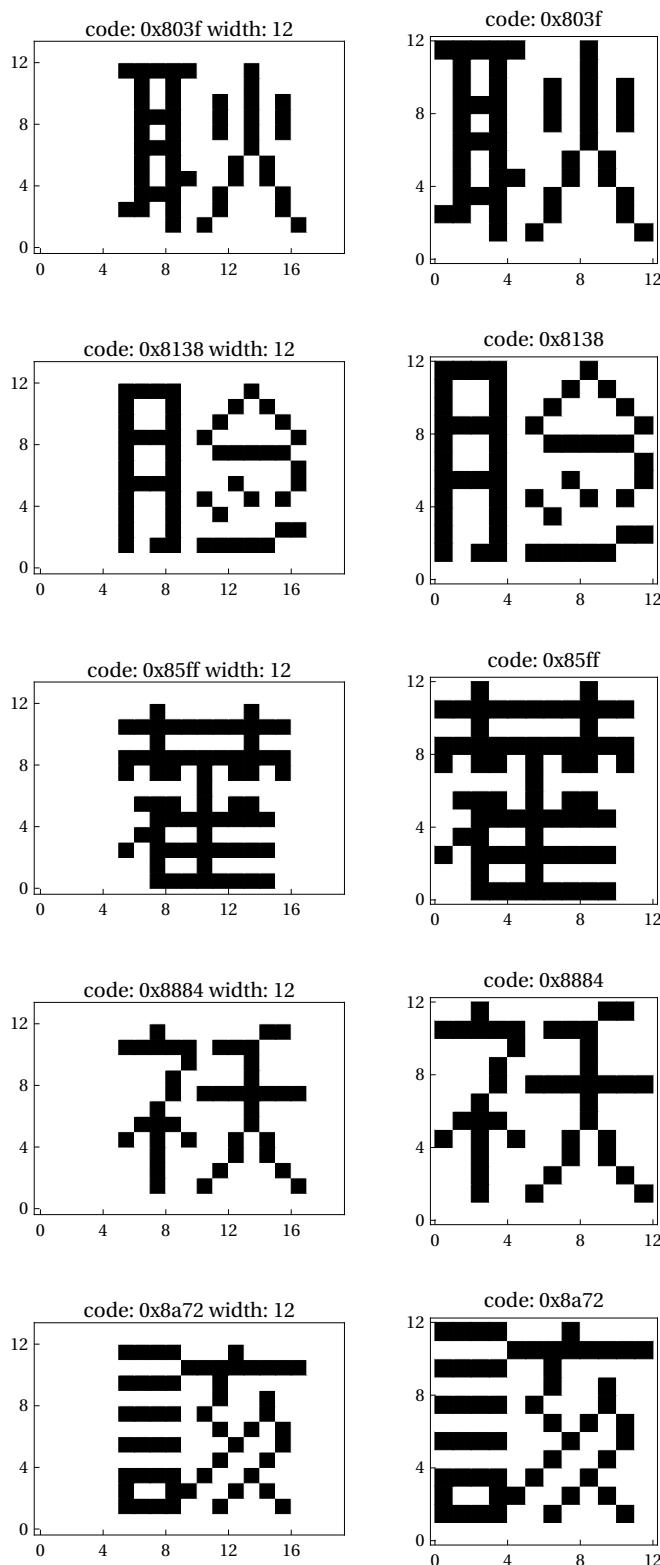


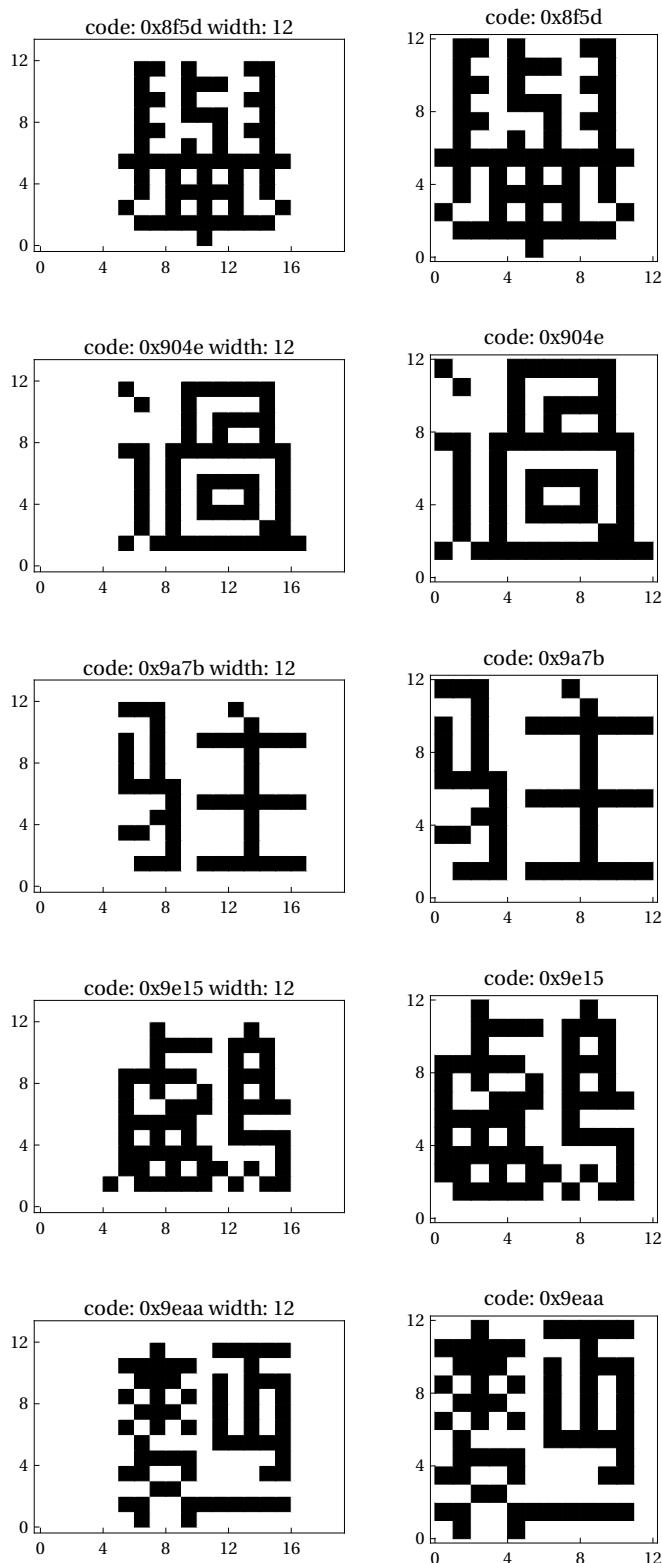


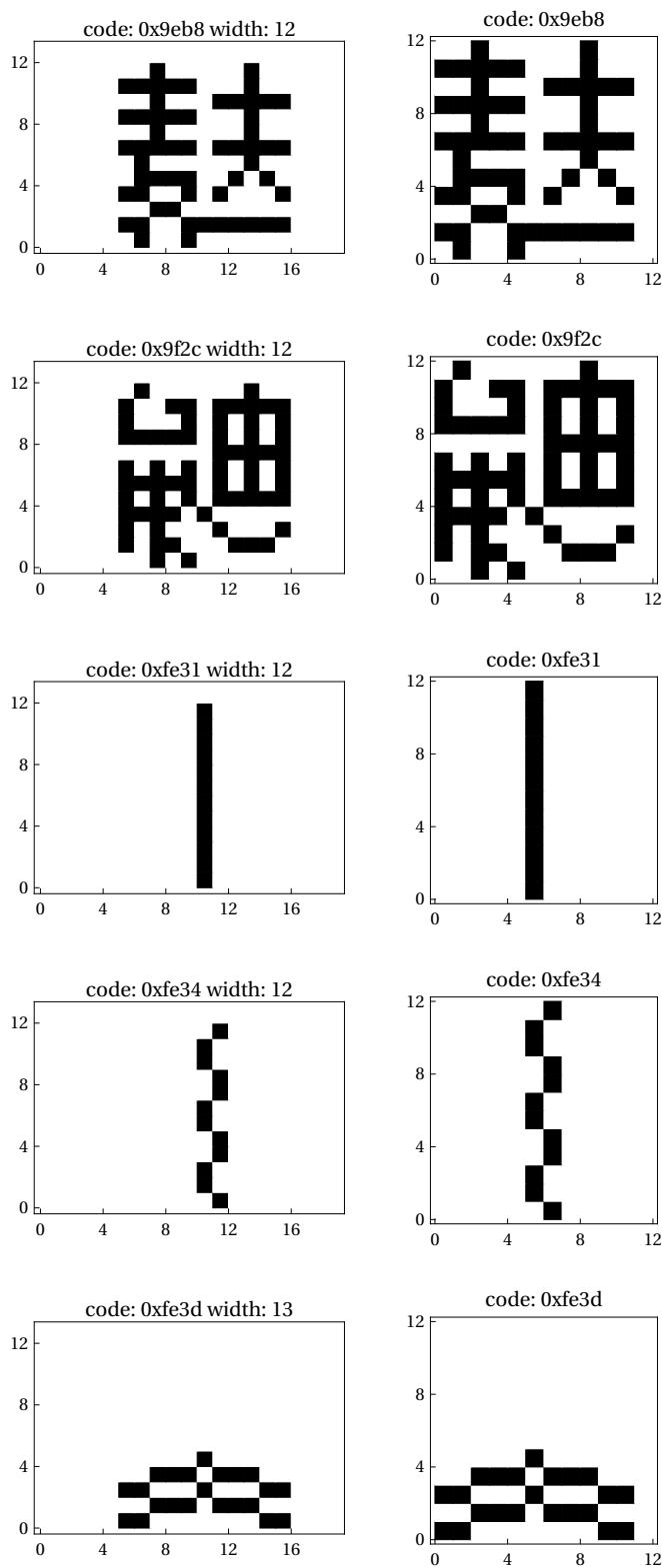


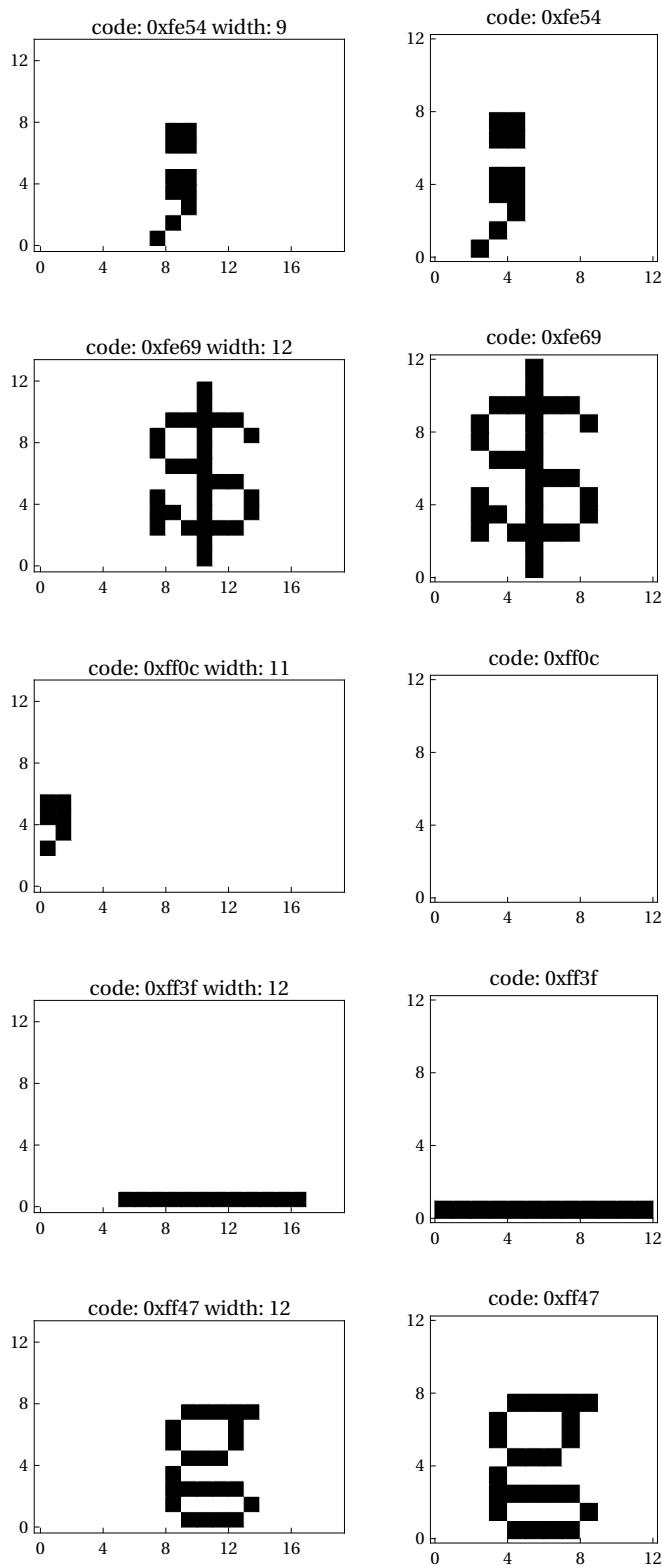


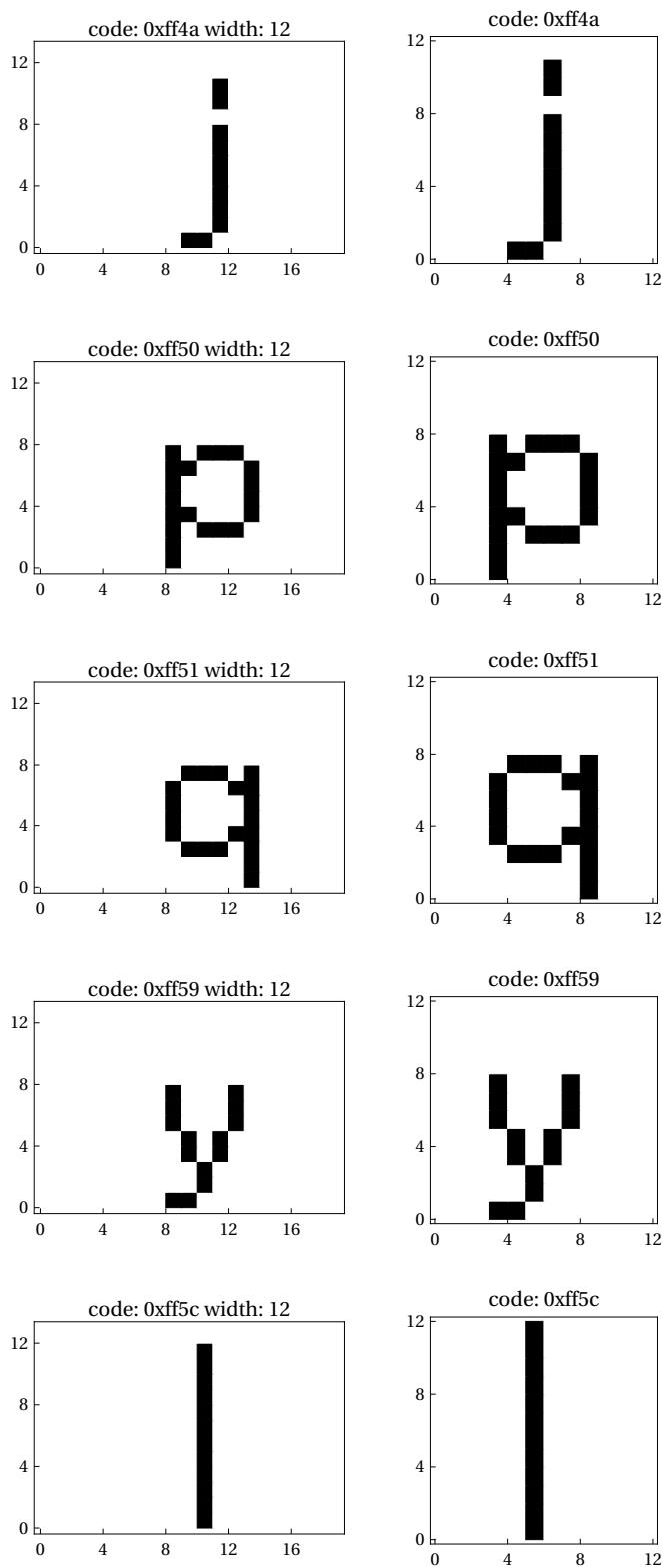


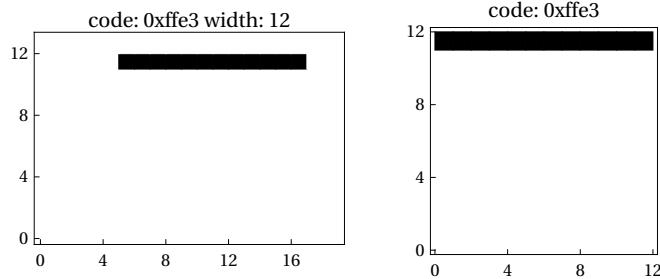










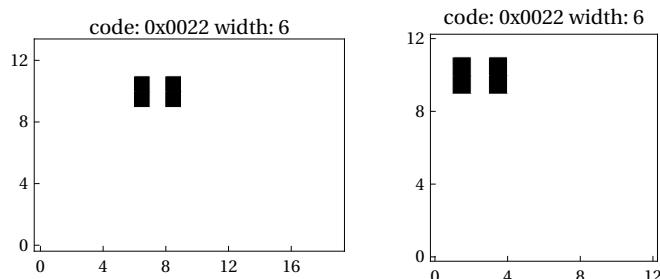
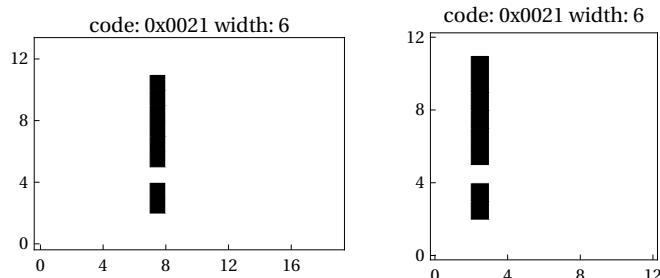
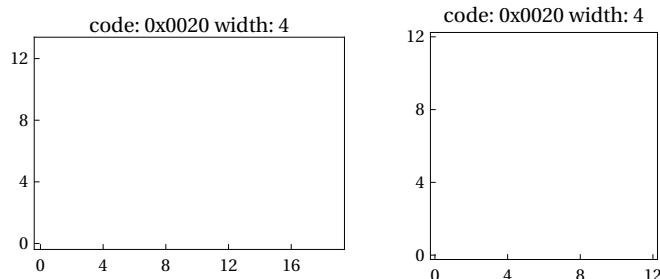


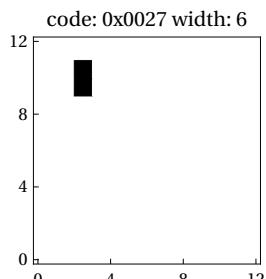
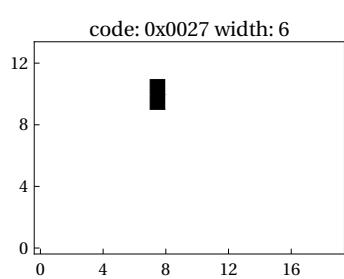
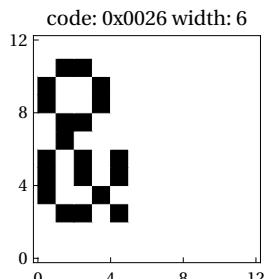
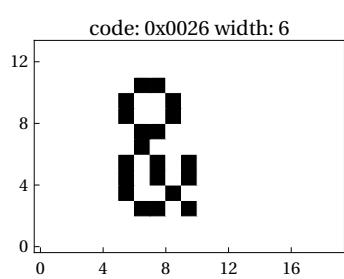
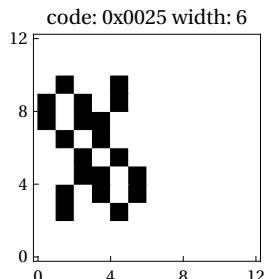
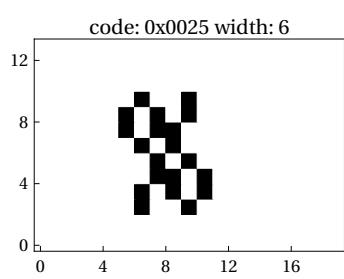
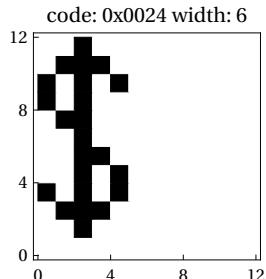
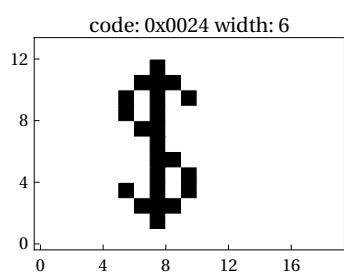
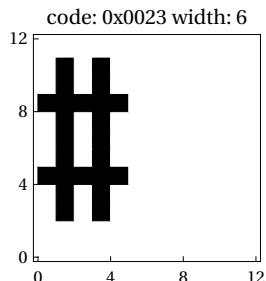
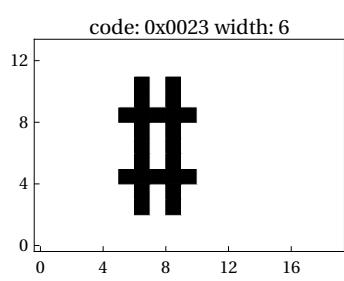
## ■ All glyphs

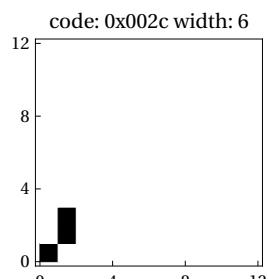
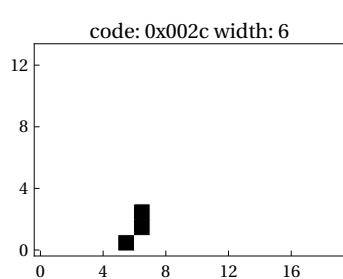
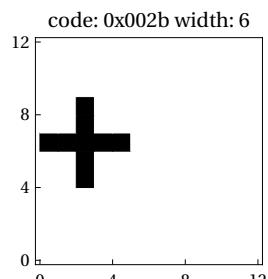
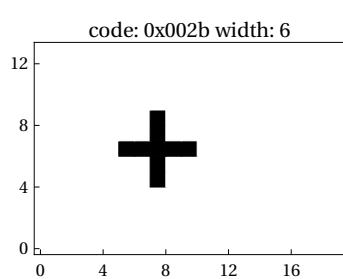
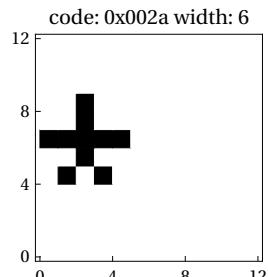
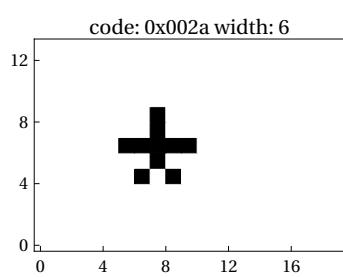
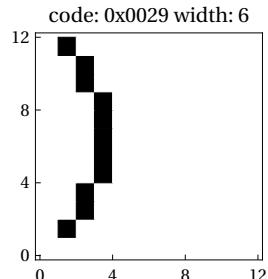
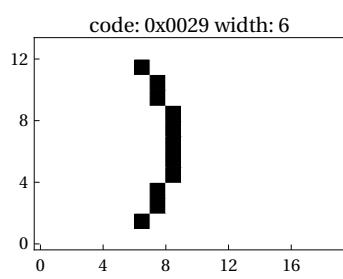
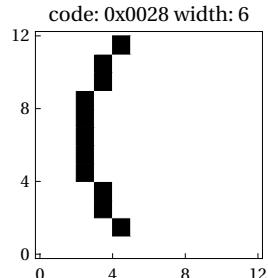
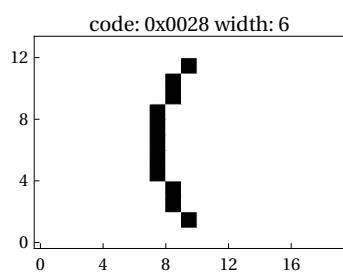
```
In[47]:= bitmapgrafix = MapThread[Graphics[Raster[Reverse[#1]], Frame -> True, AspectRatio ->
    Divide @@ Dimensions[#1], FrameTicks -> ({#, #, {}, {}} &[Range[0, 16, 4]]),
    PlotLabel -> ("code: " <> IntegerString[ToExpression[#2], 16, 4] <>
    " width: " <> ToString[#3])] &, {extbitmaps, charcodes,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

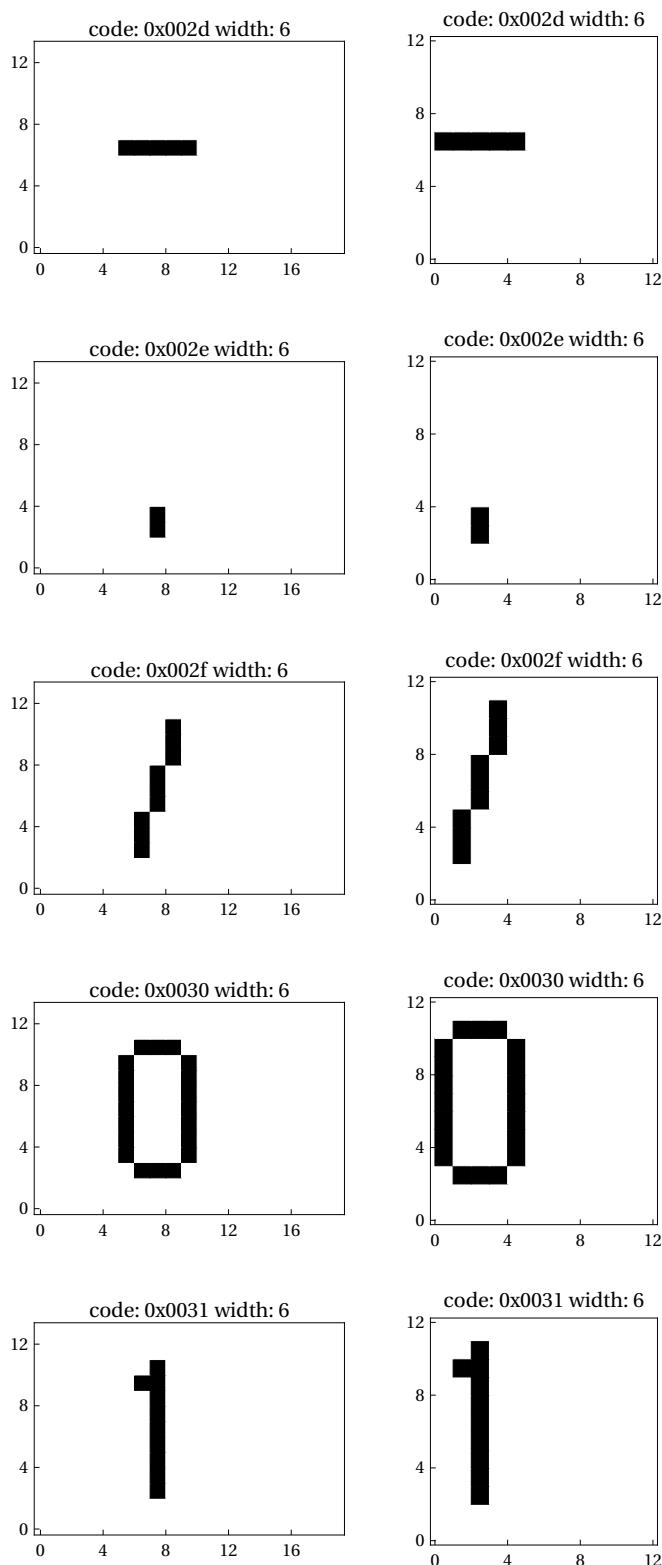
In[48]:= bitmaprgrafix = MapThread[Graphics[Raster[Reverse[#1]], Frame -> True, AspectRatio ->
    Divide @@ Dimensions[#1], FrameTicks -> ({#, #, {}, {}} &[Range[0, 16, 4]]),
    PlotLabel -> ("code: " <> IntegerString[ToExpression[#2], 16, 4] <>
    " width: " <> ToString[#3])] &, {trbitmaps, charcodes,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

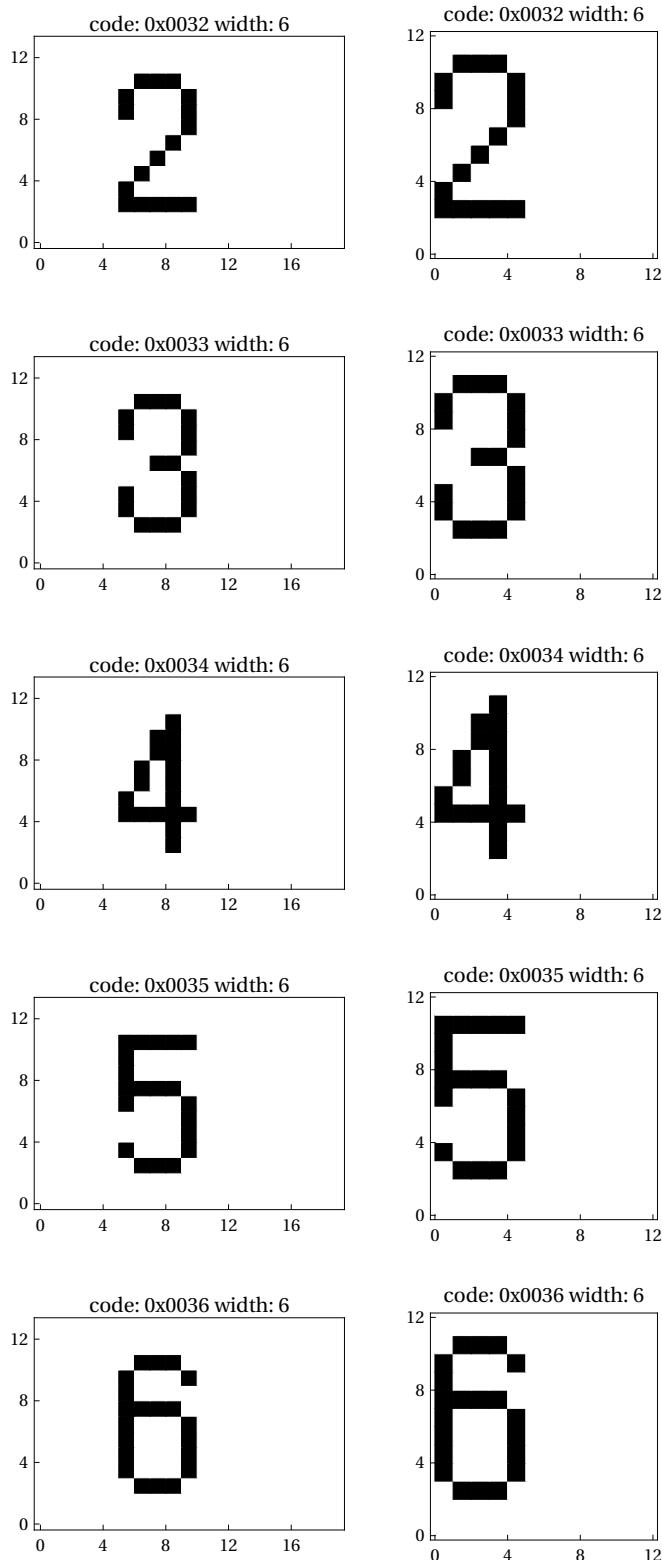
In[49]:= MapThread[Print[GraphicsGrid[{{##}}]] &,
    Take[#, 1024] &/@{bitmapgrafix, bitmaprgrafix}];
```

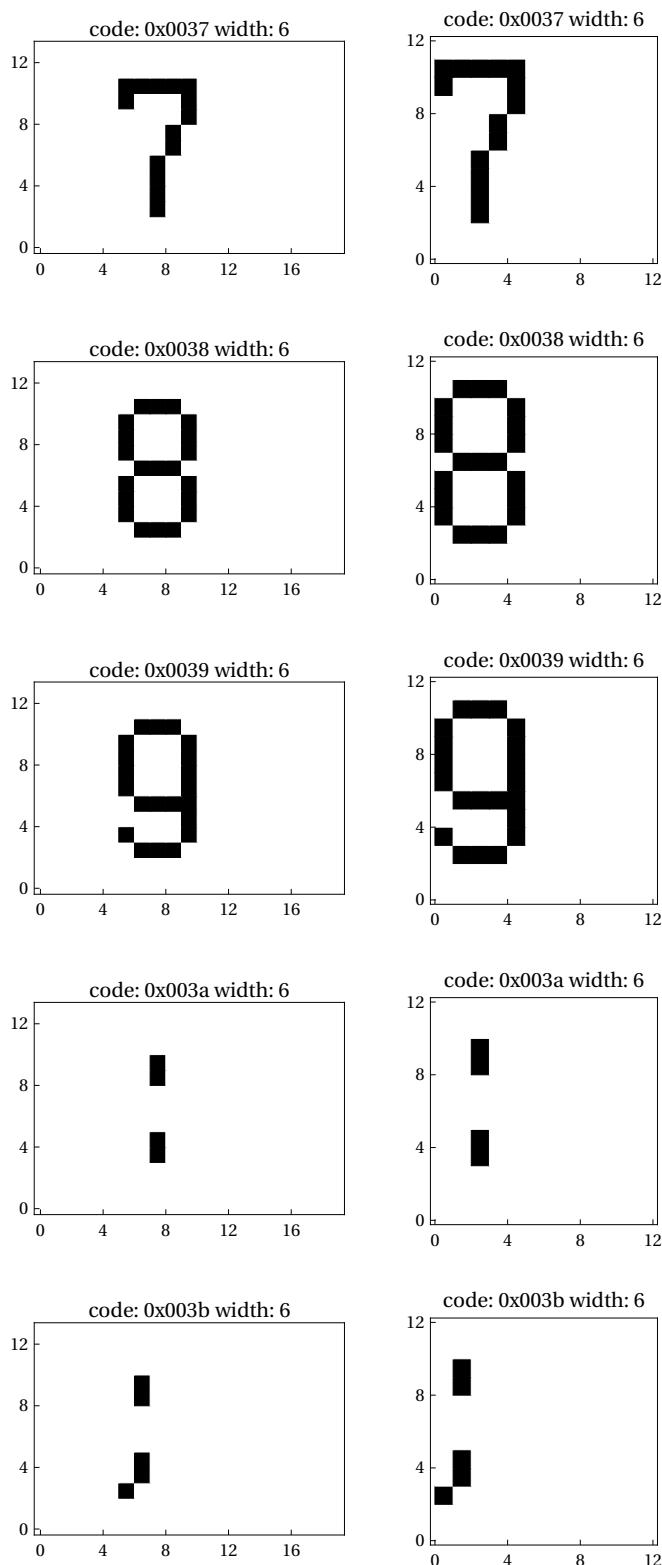


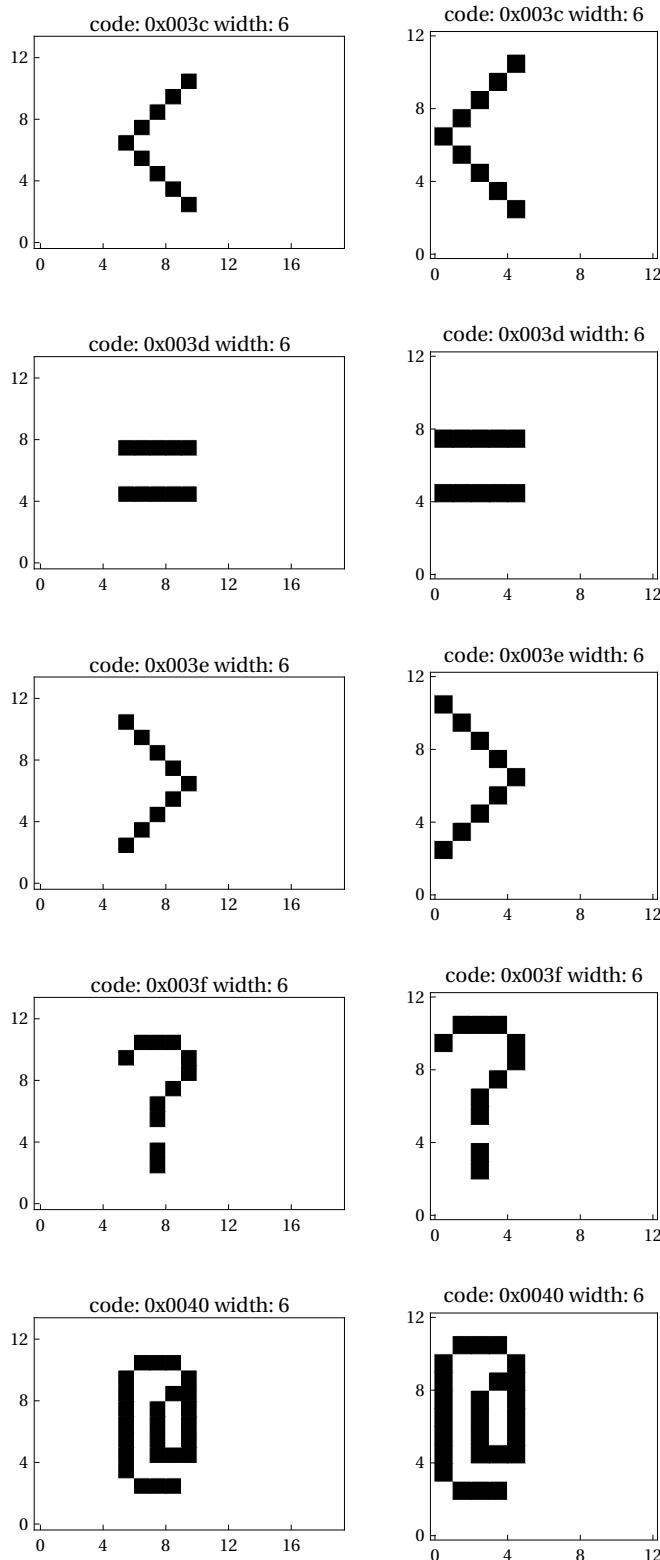


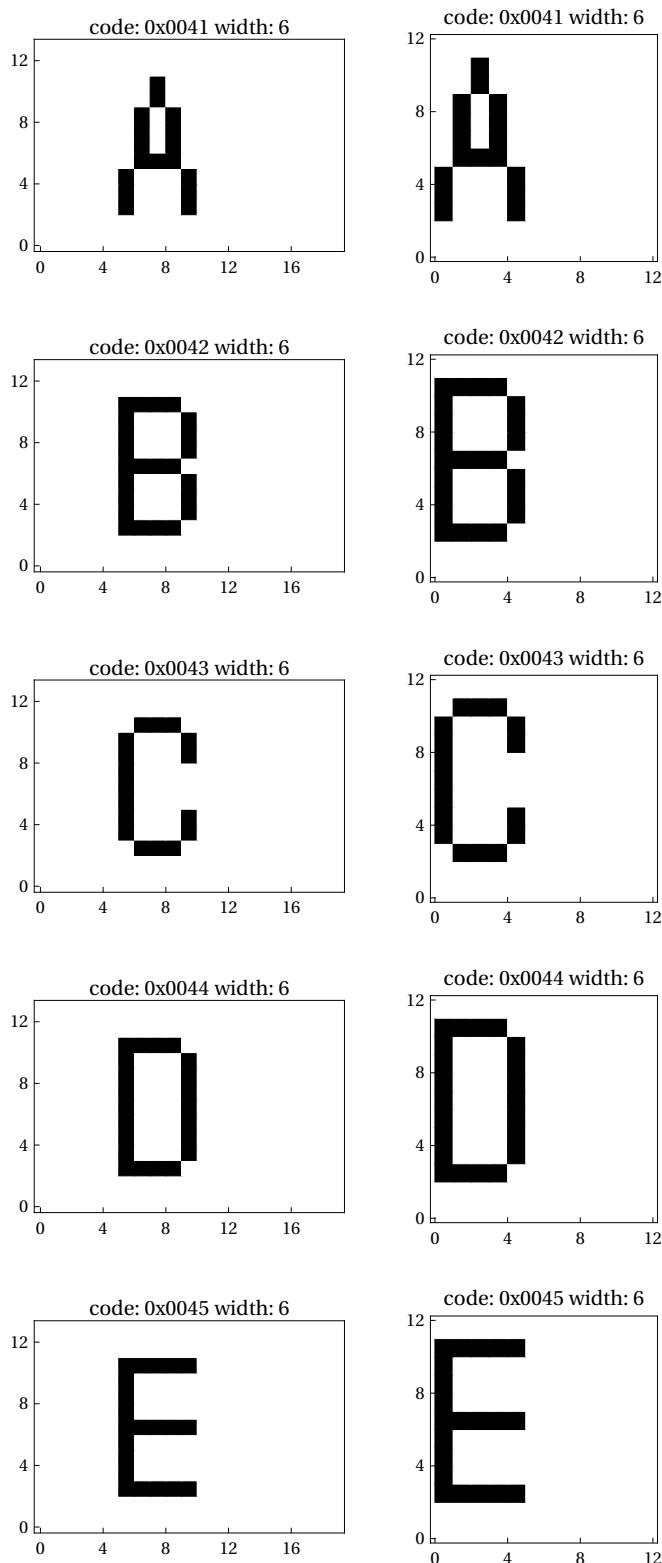


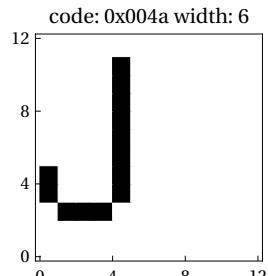
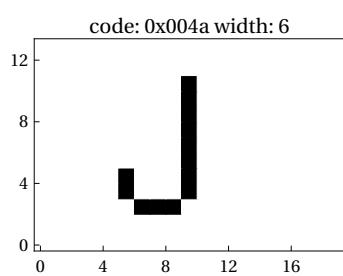
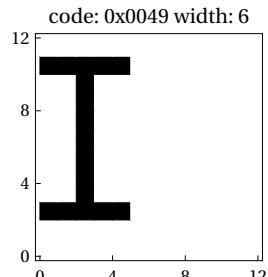
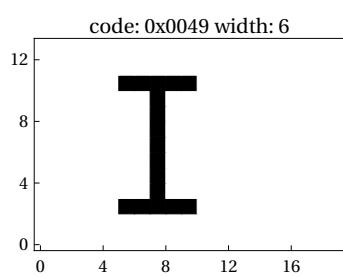
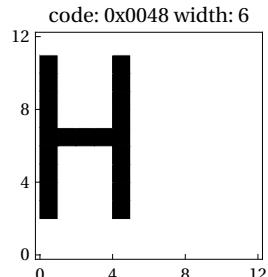
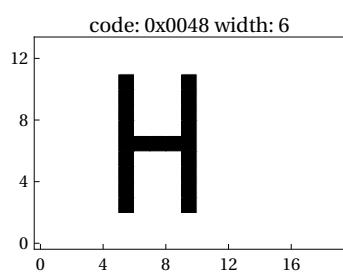
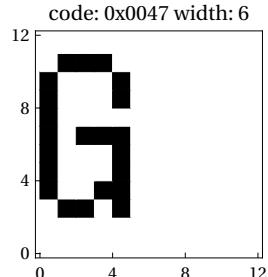
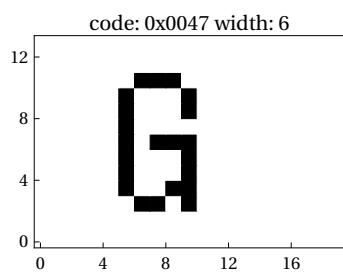
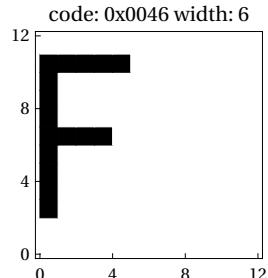
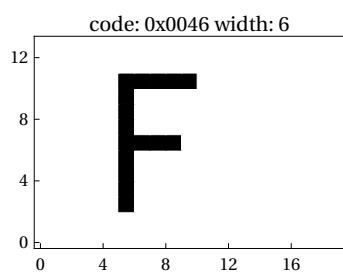


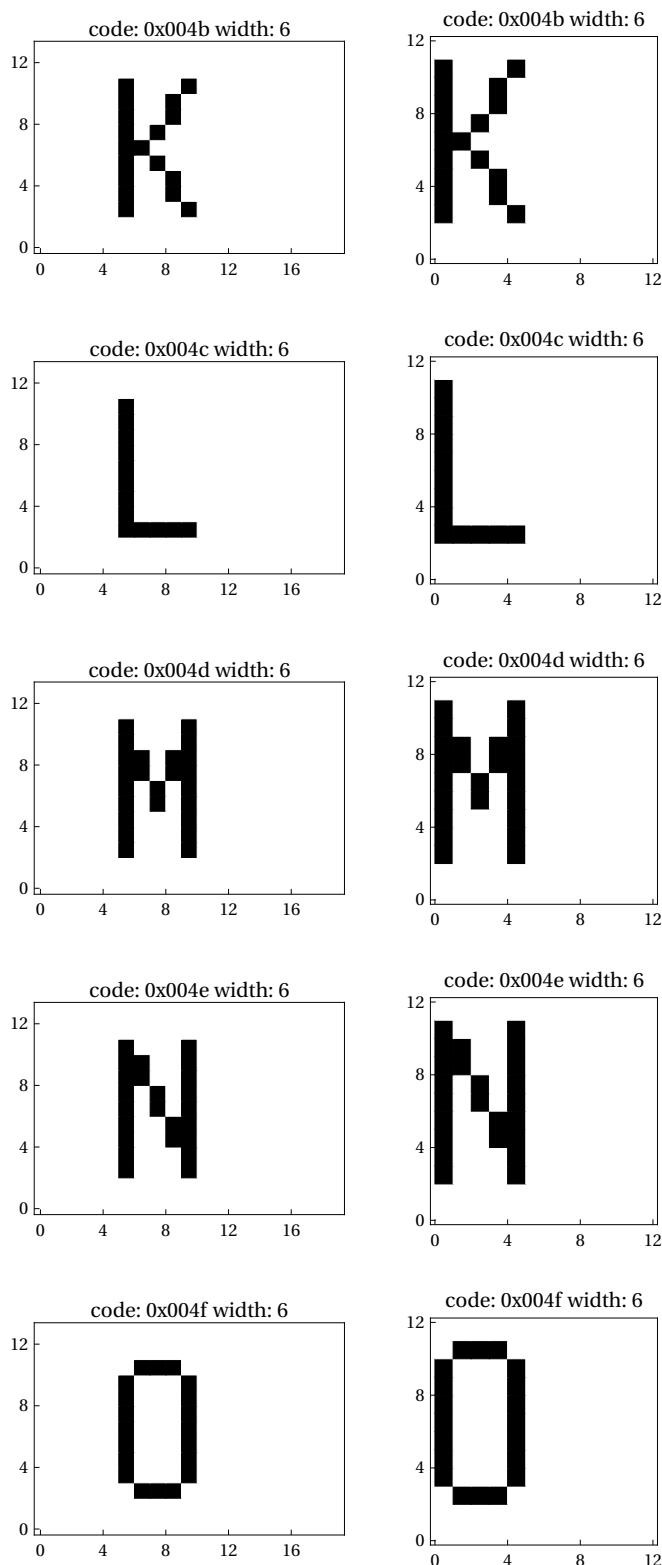


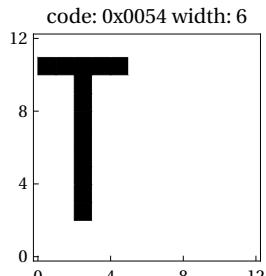
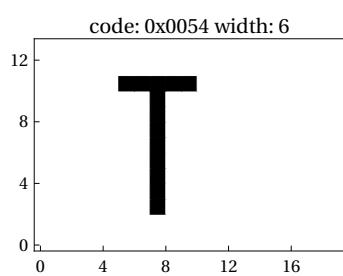
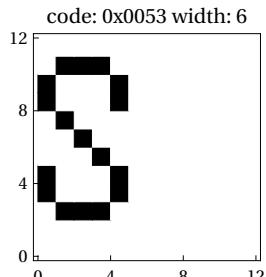
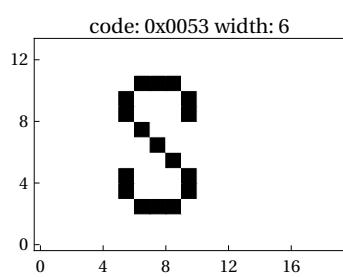
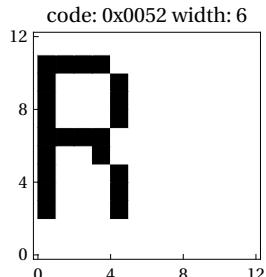
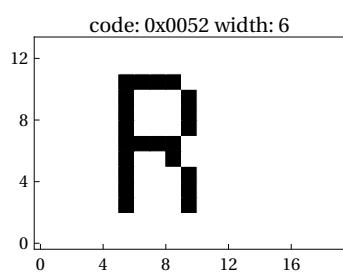
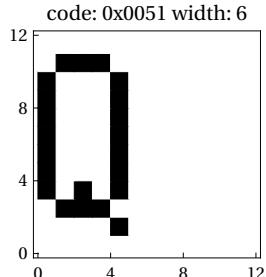
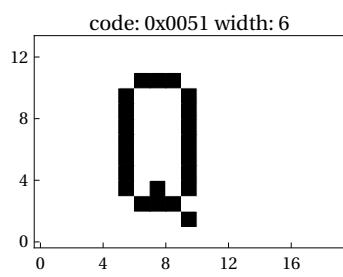
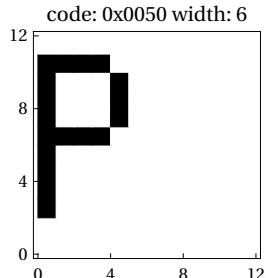
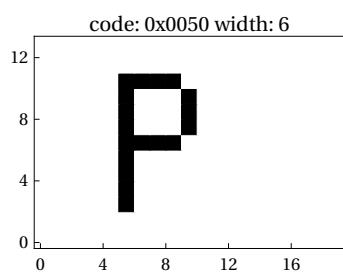


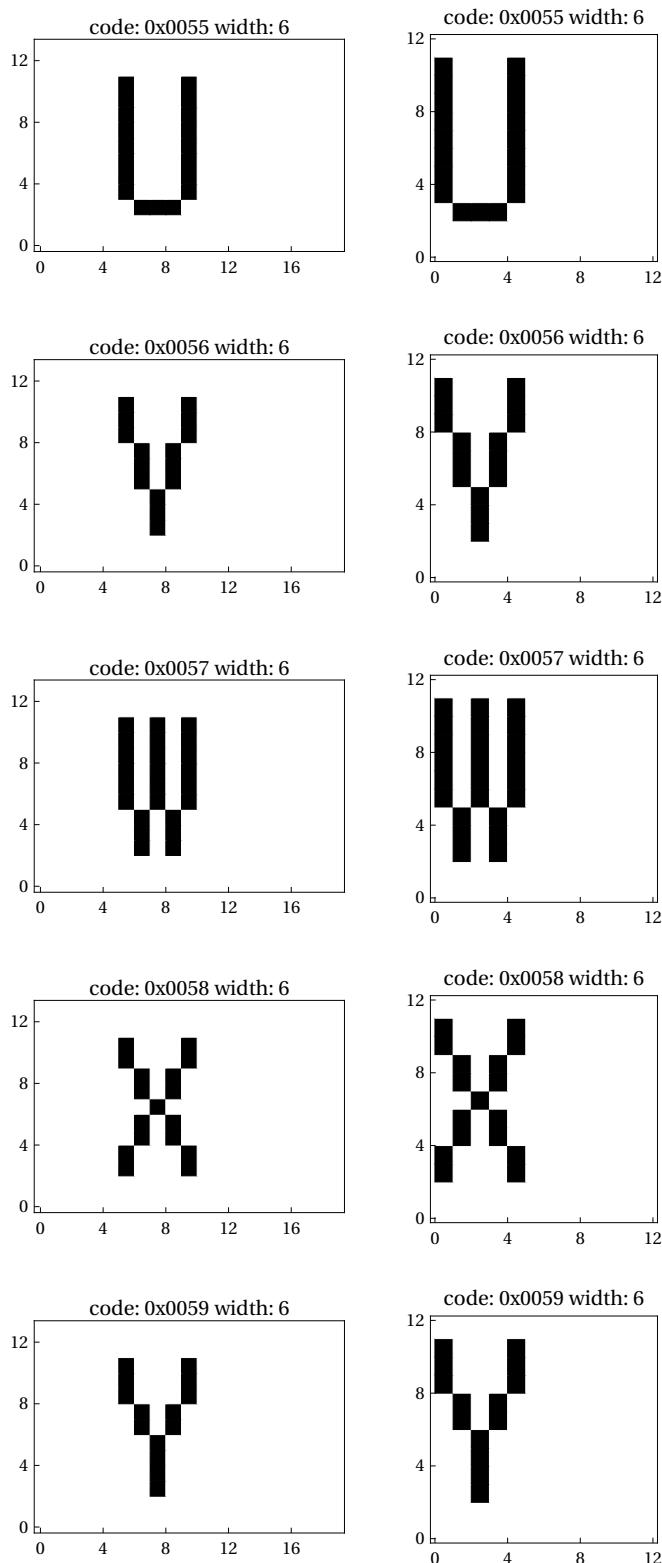


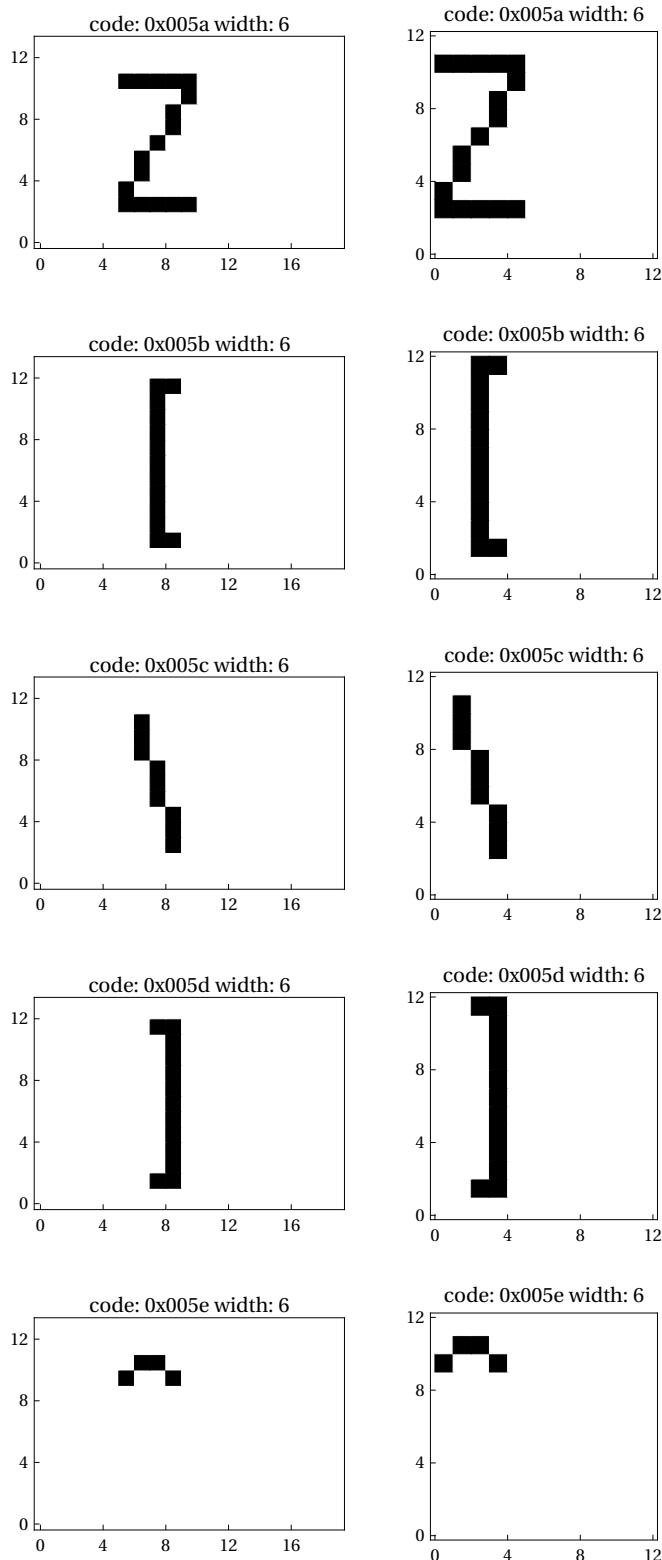


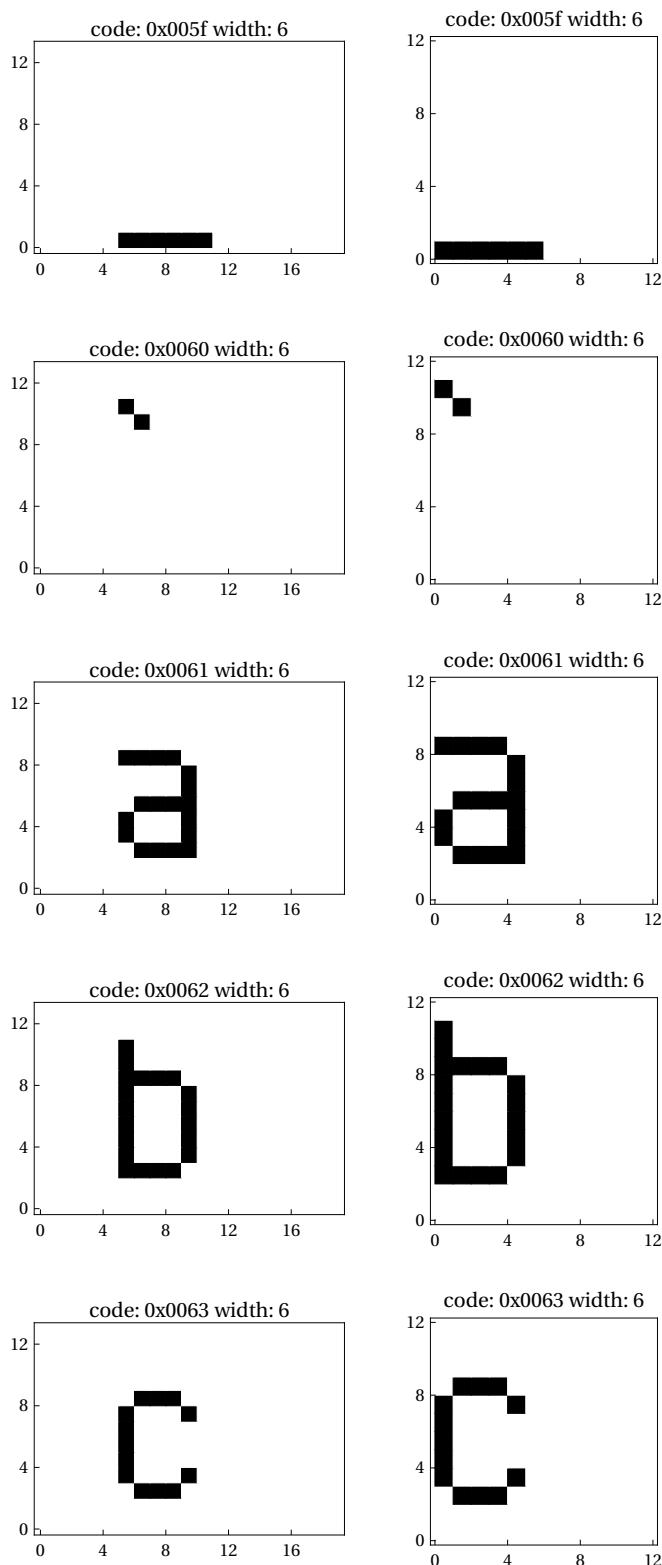


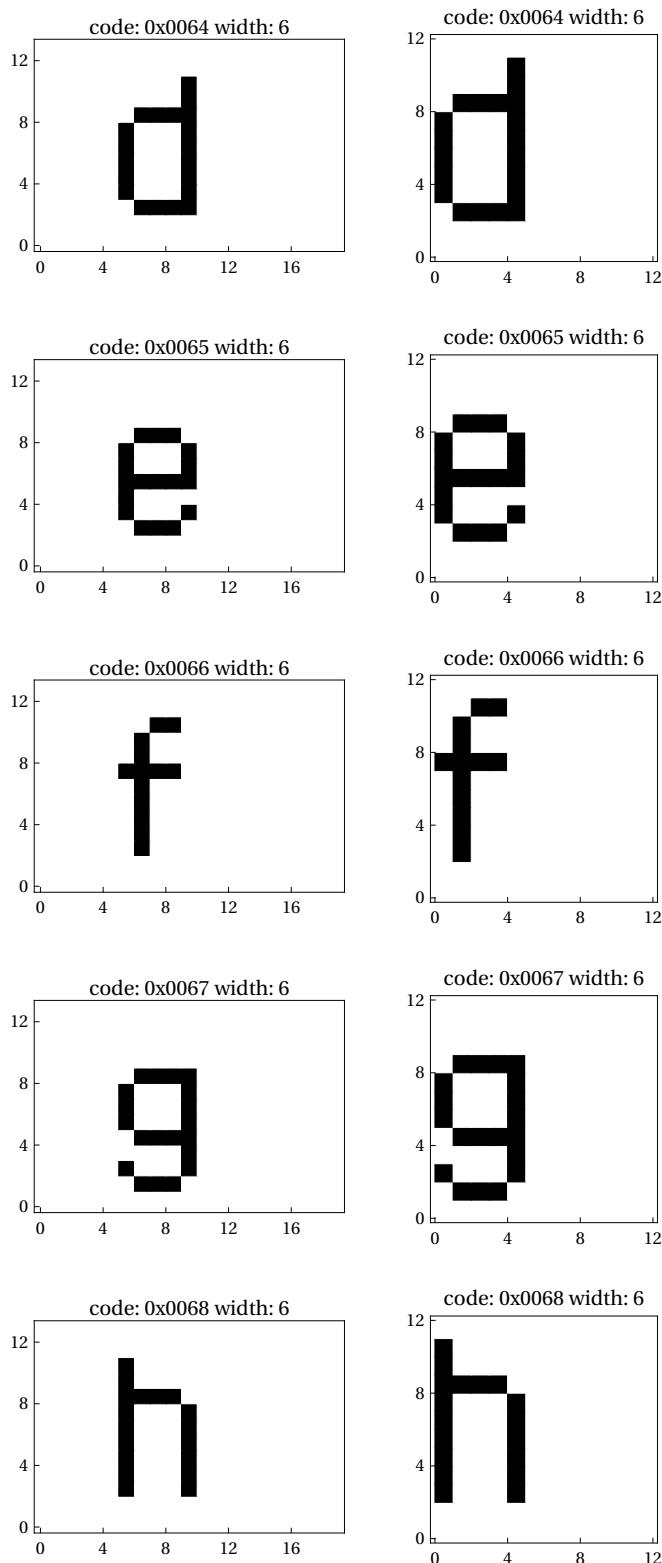


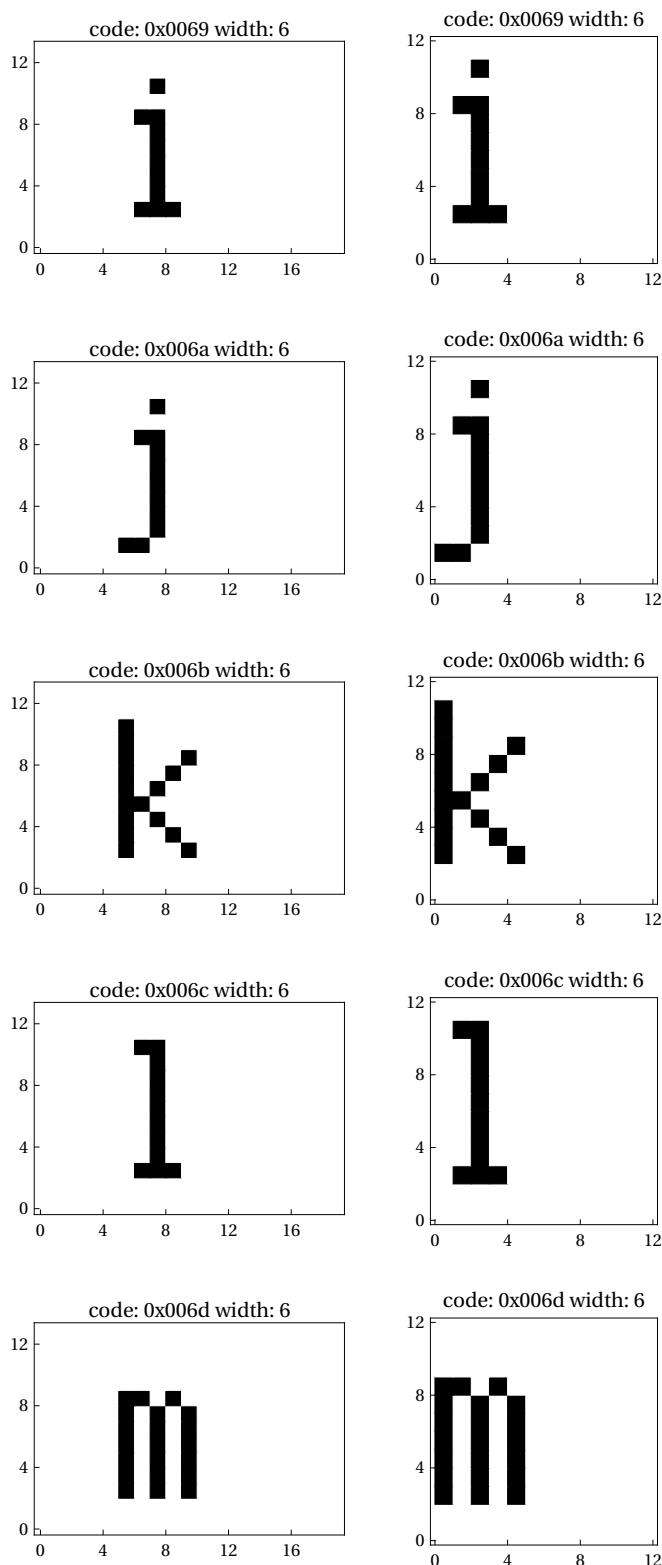


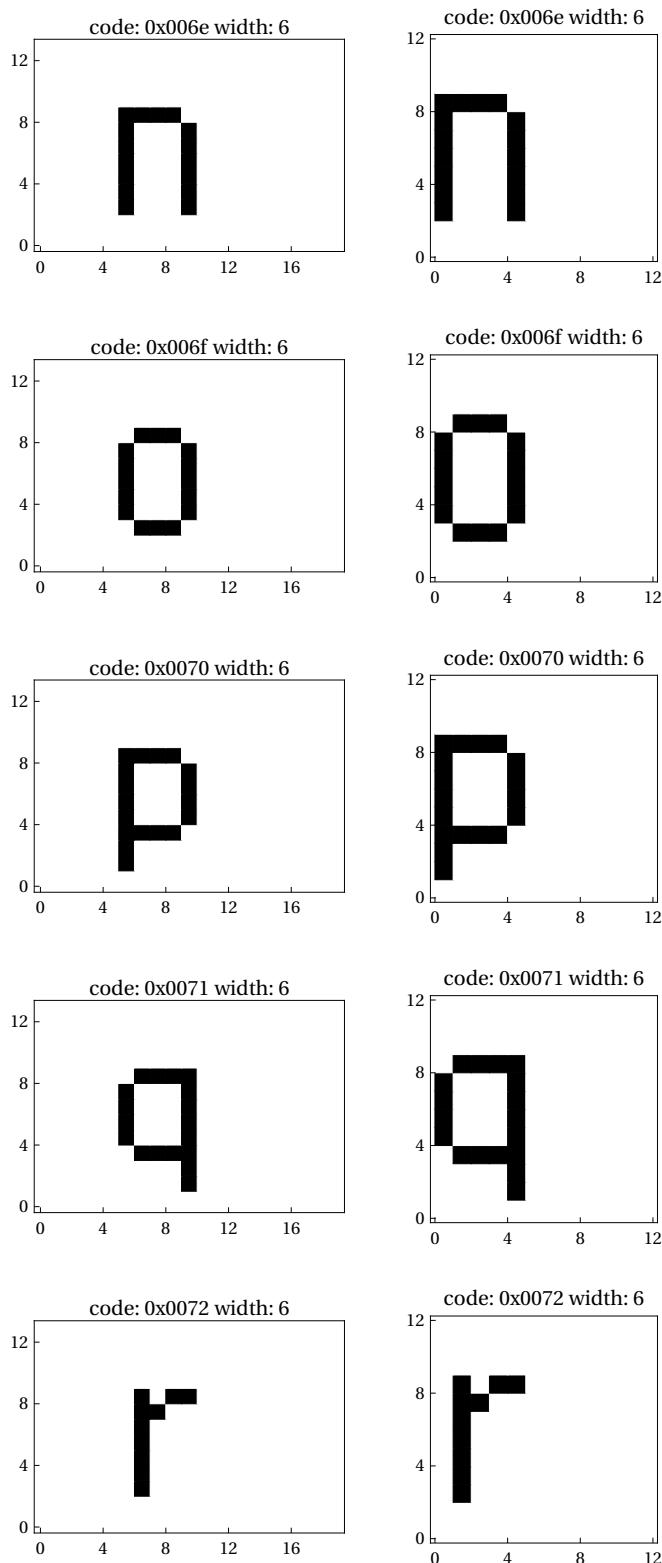


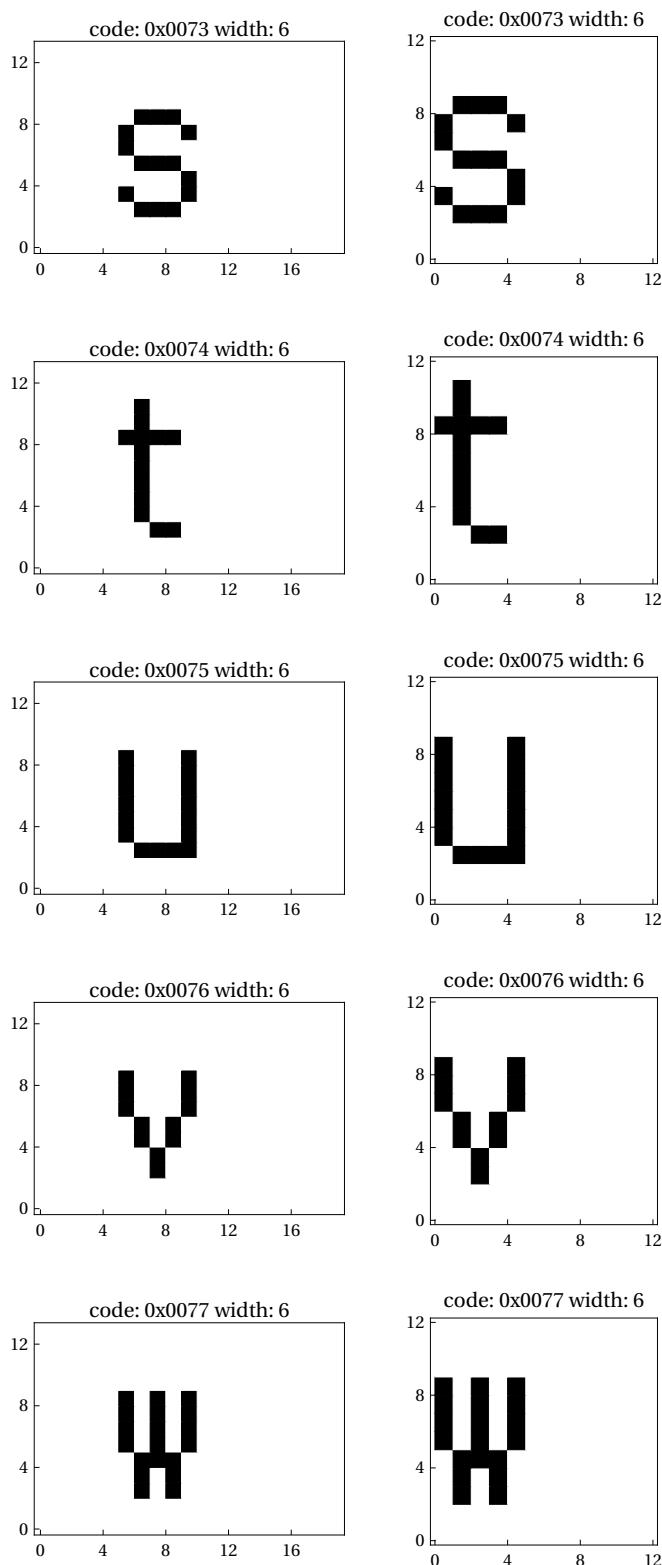


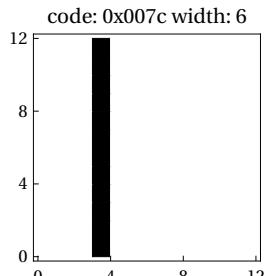
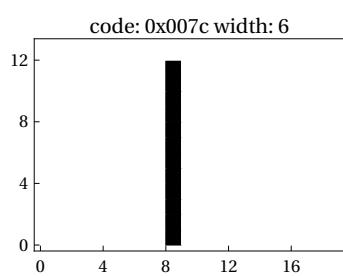
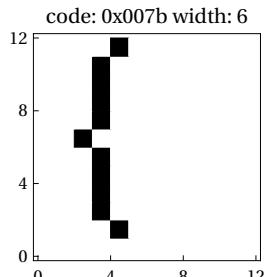
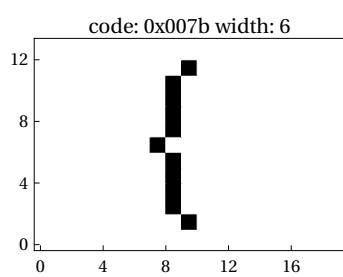
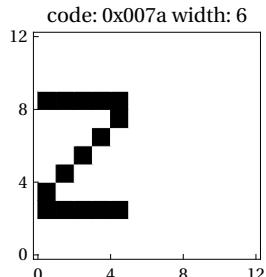
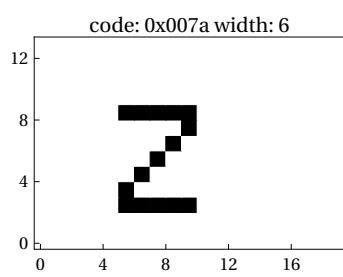
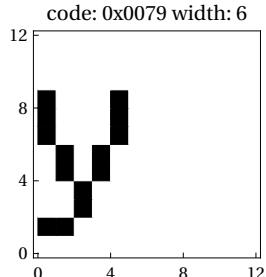
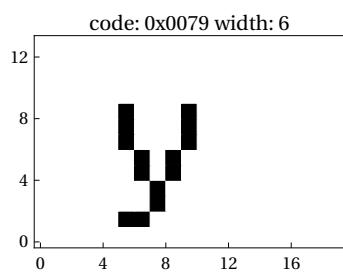
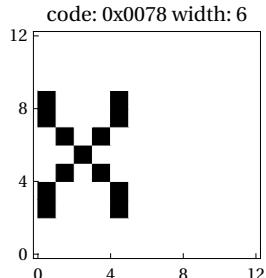
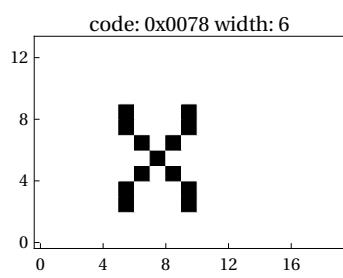


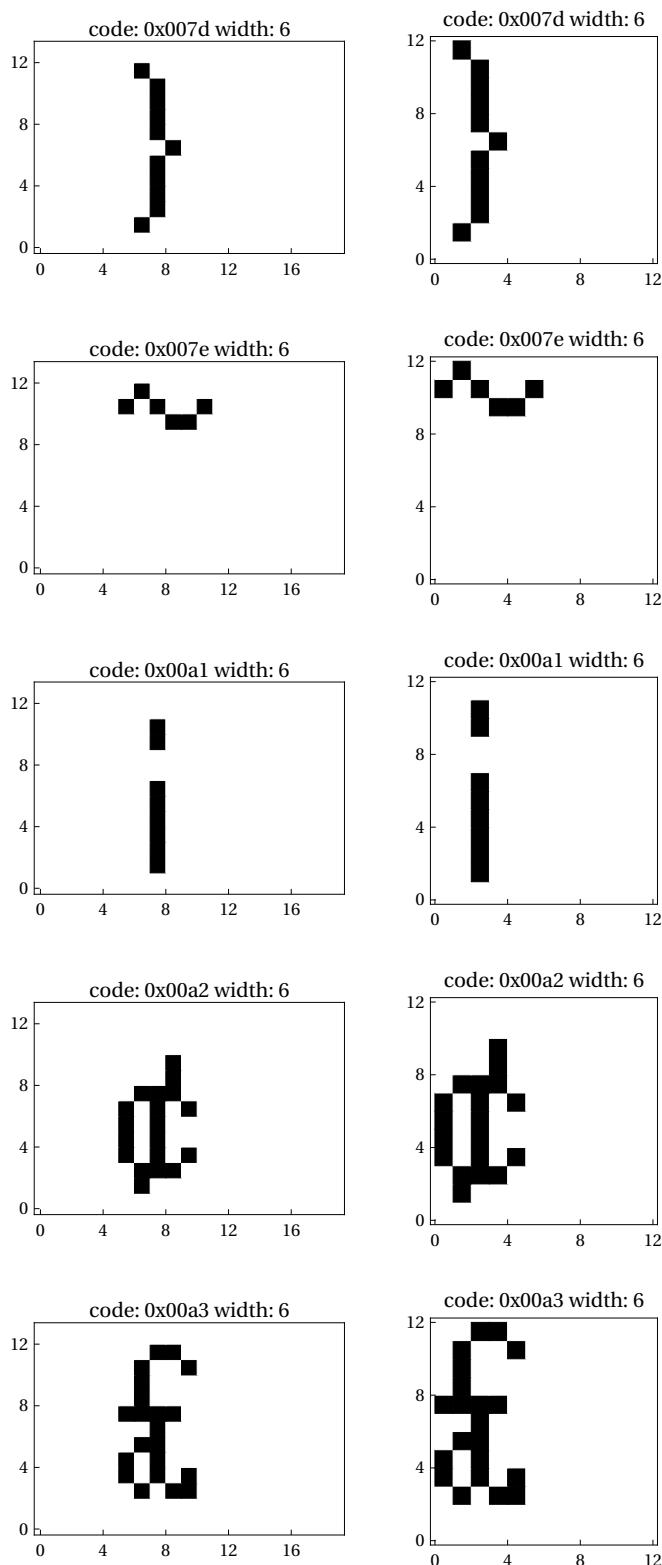


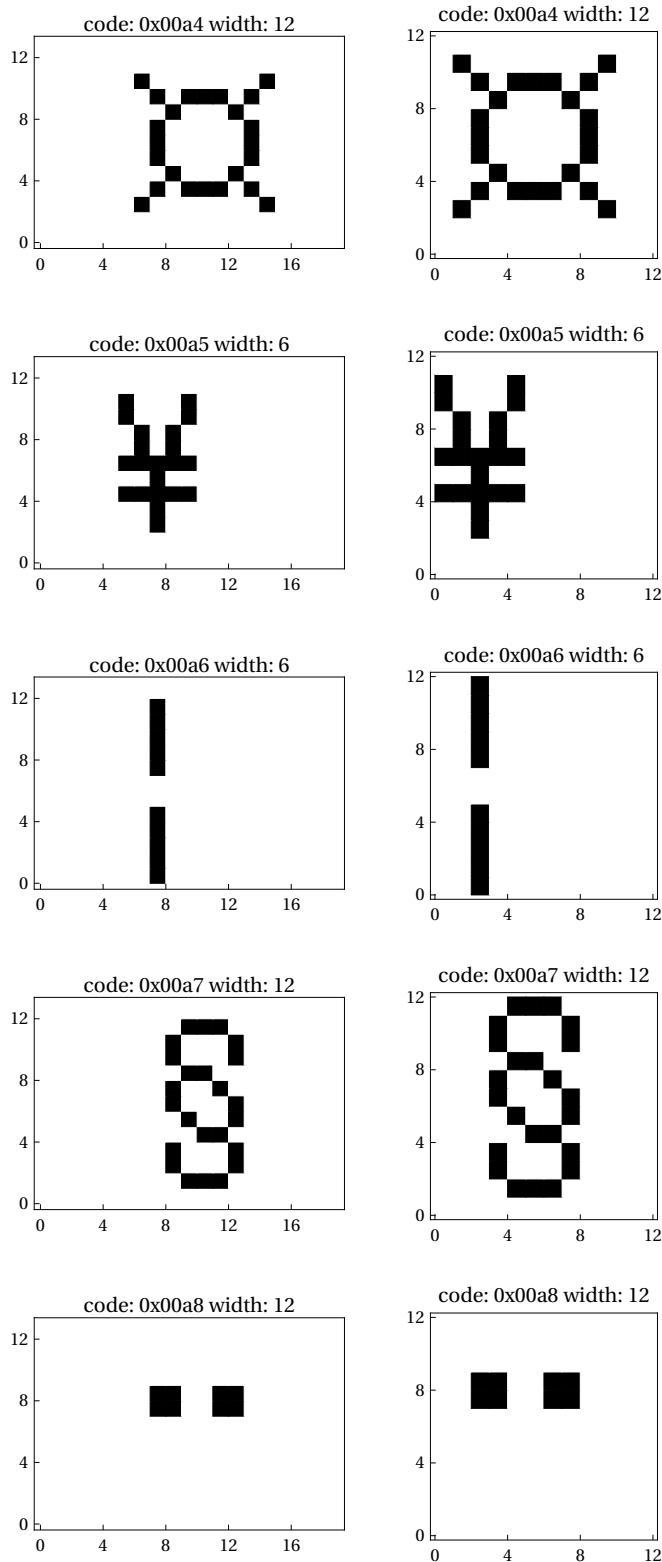


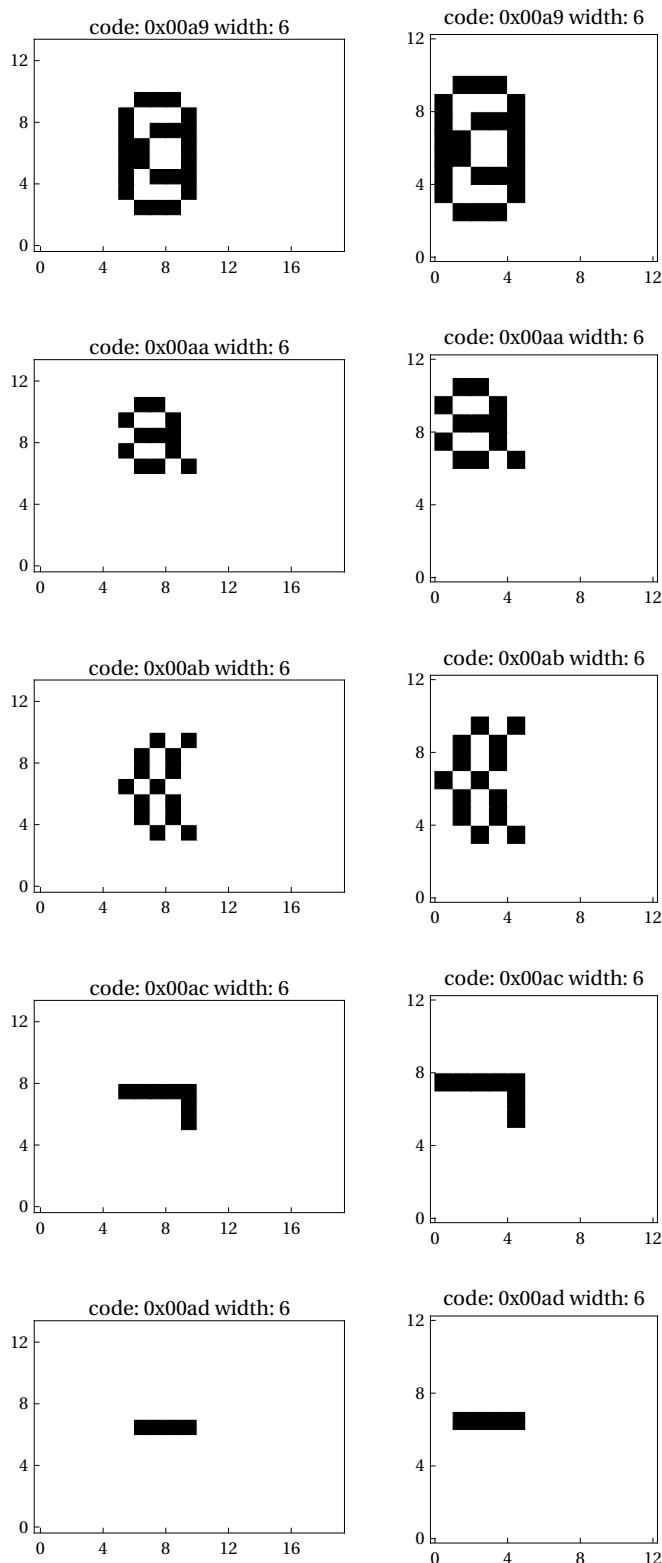


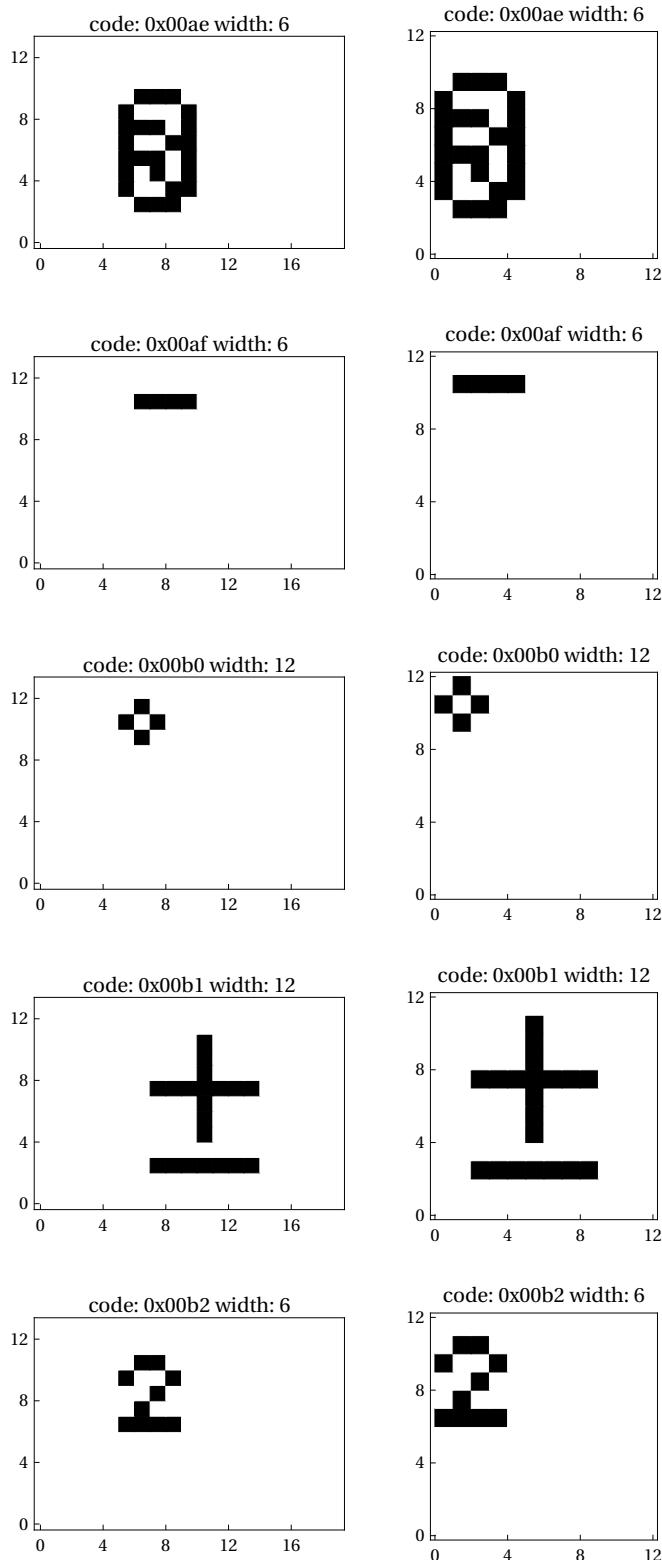


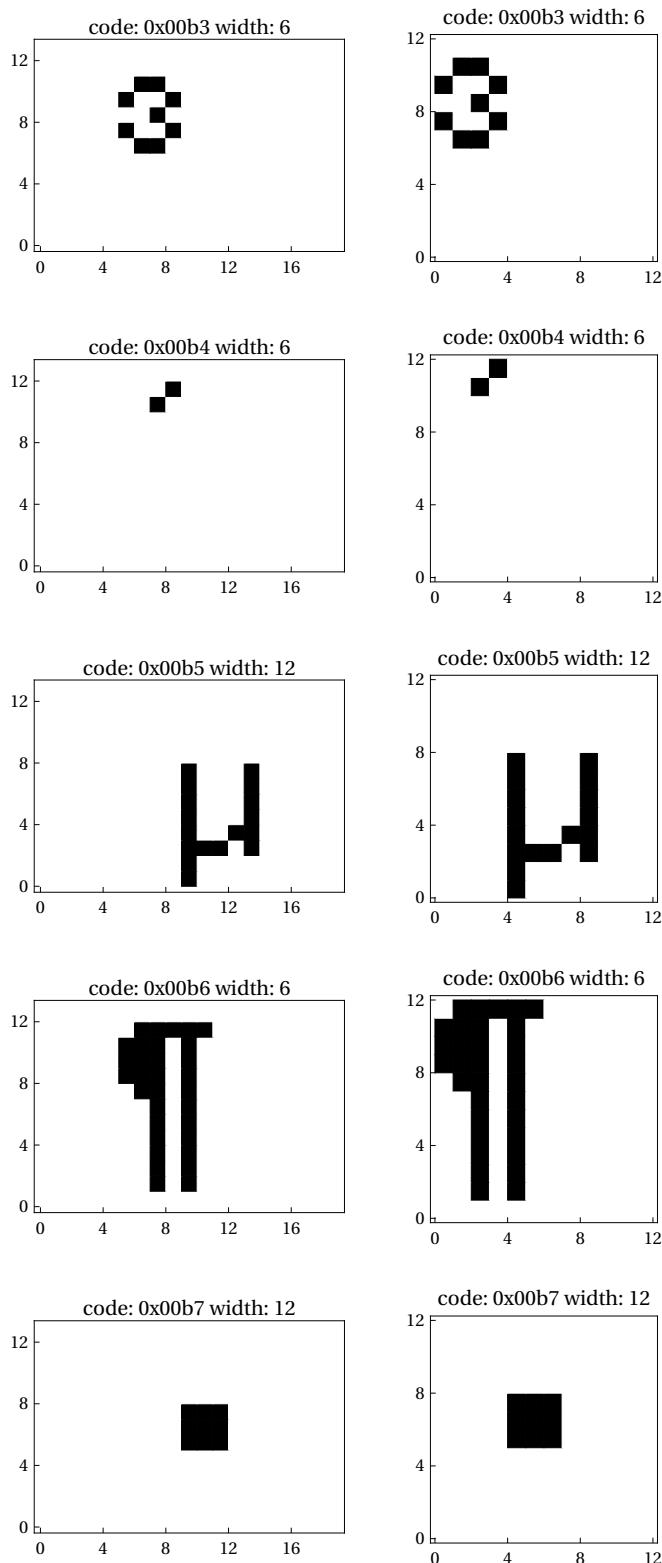


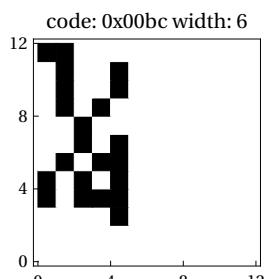
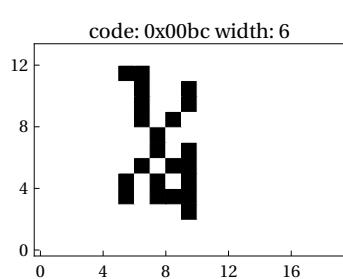
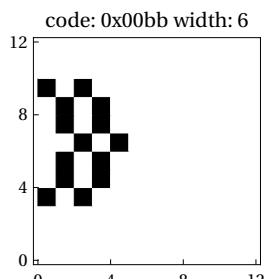
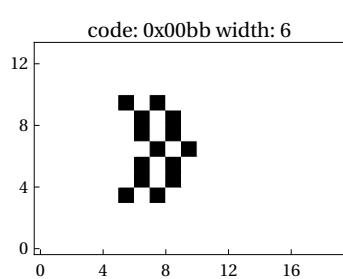
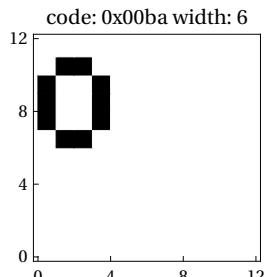
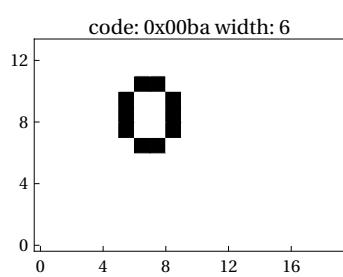
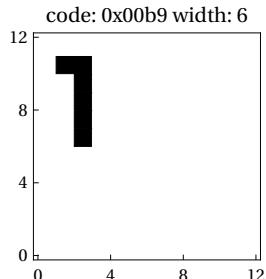
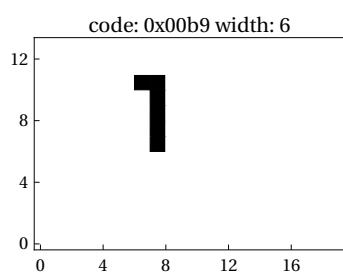
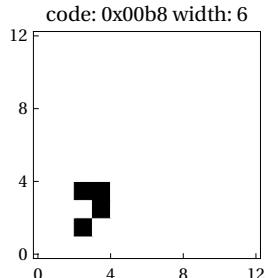
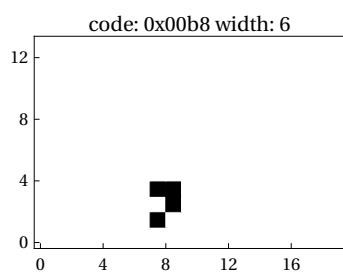


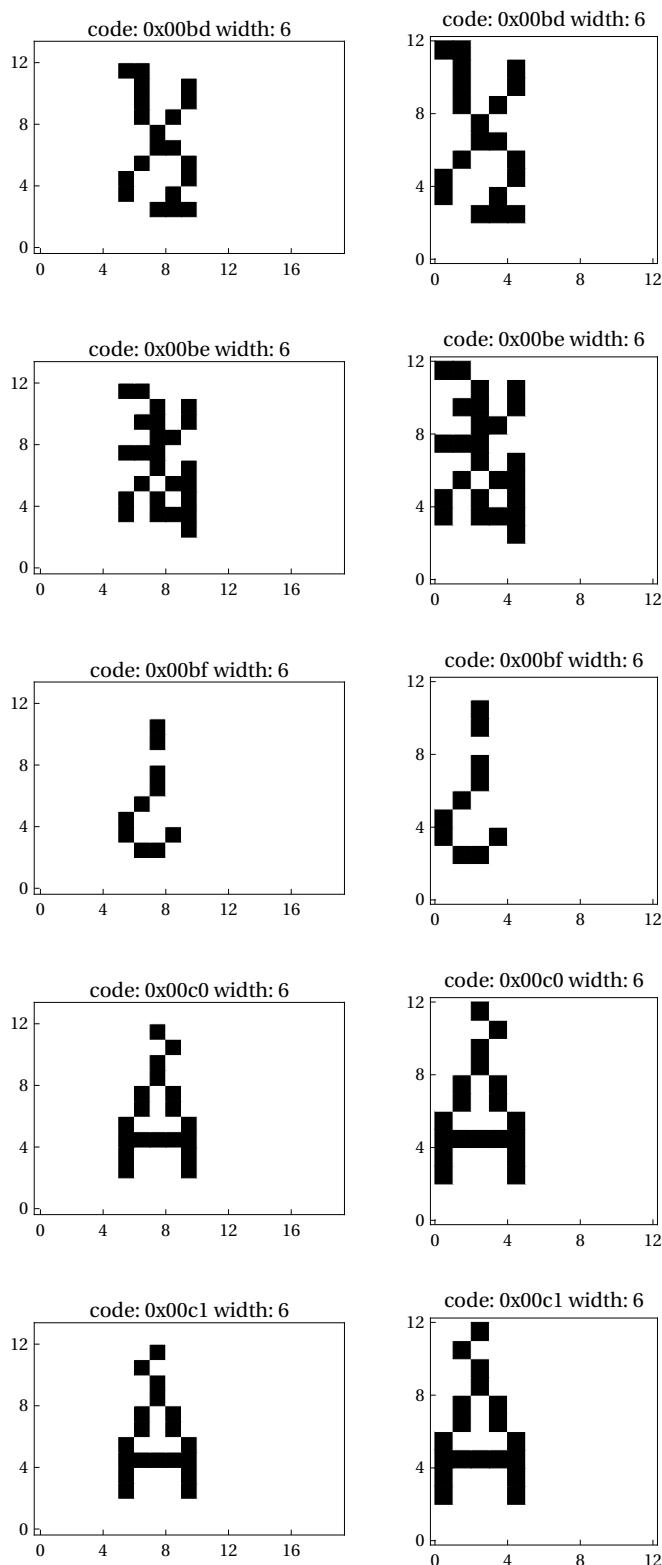


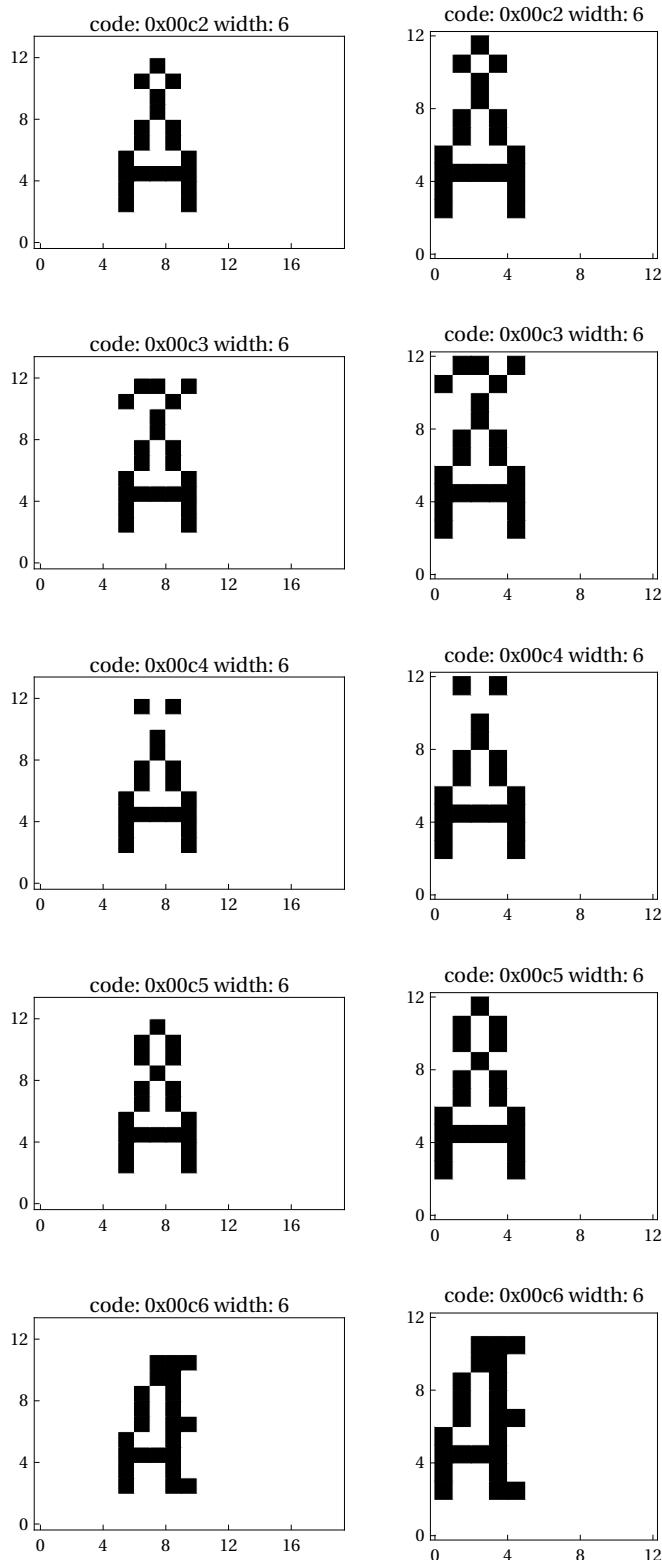


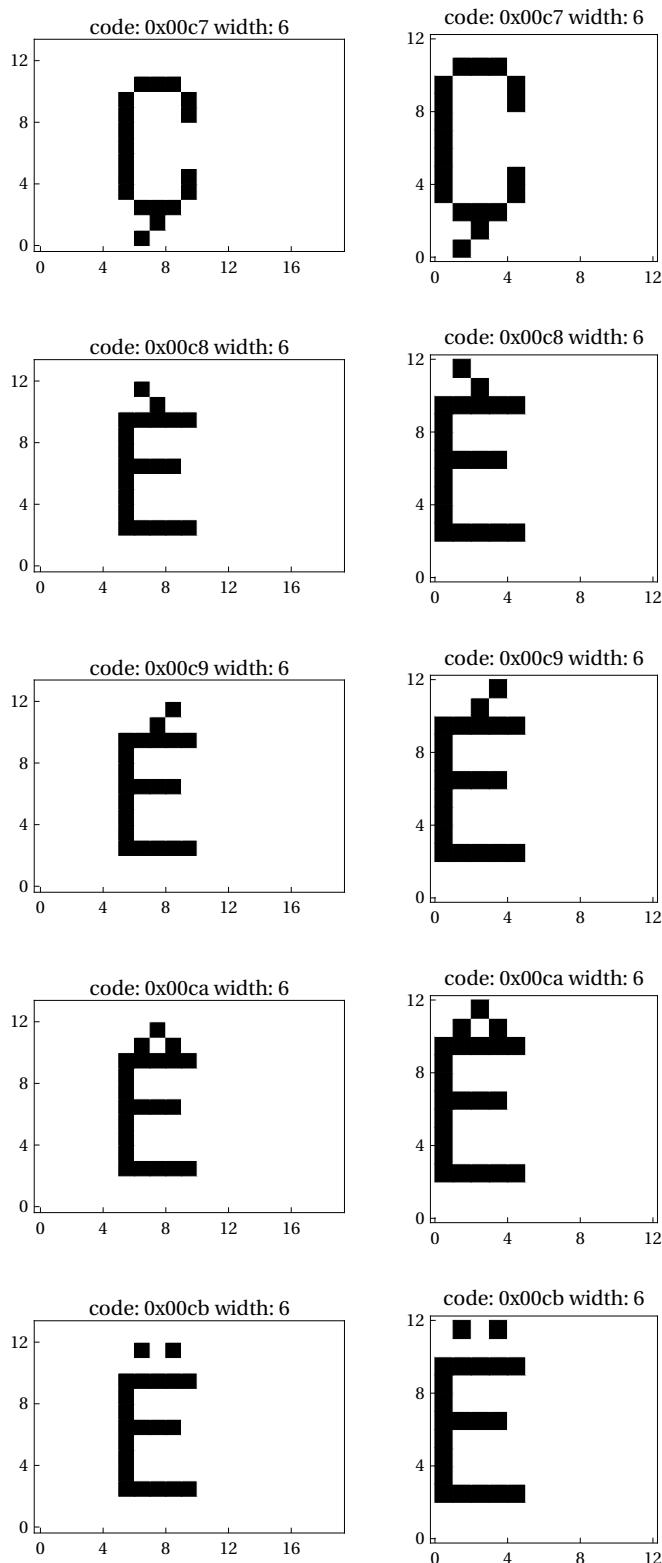


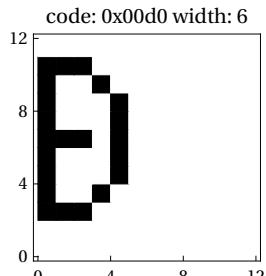
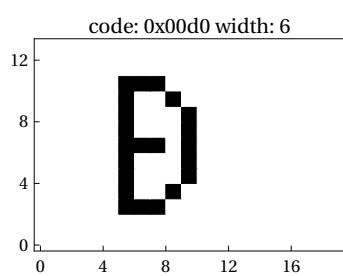
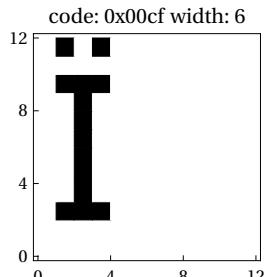
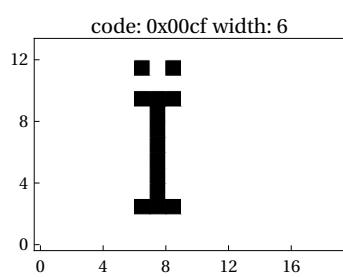
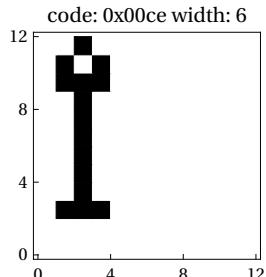
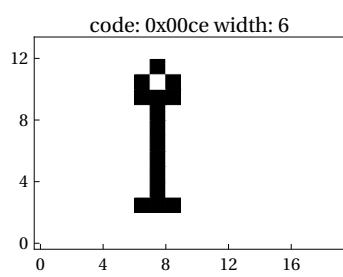
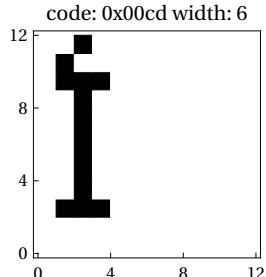
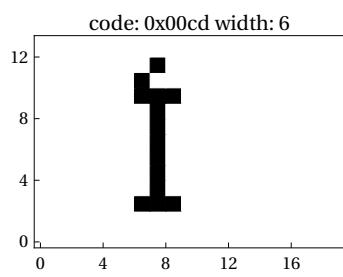
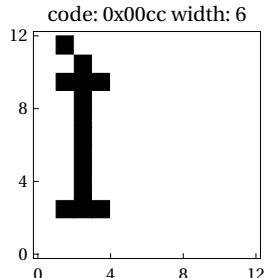
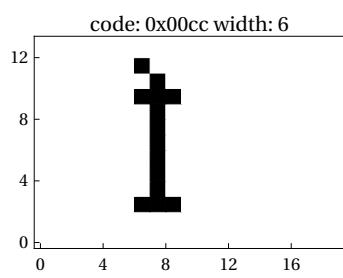


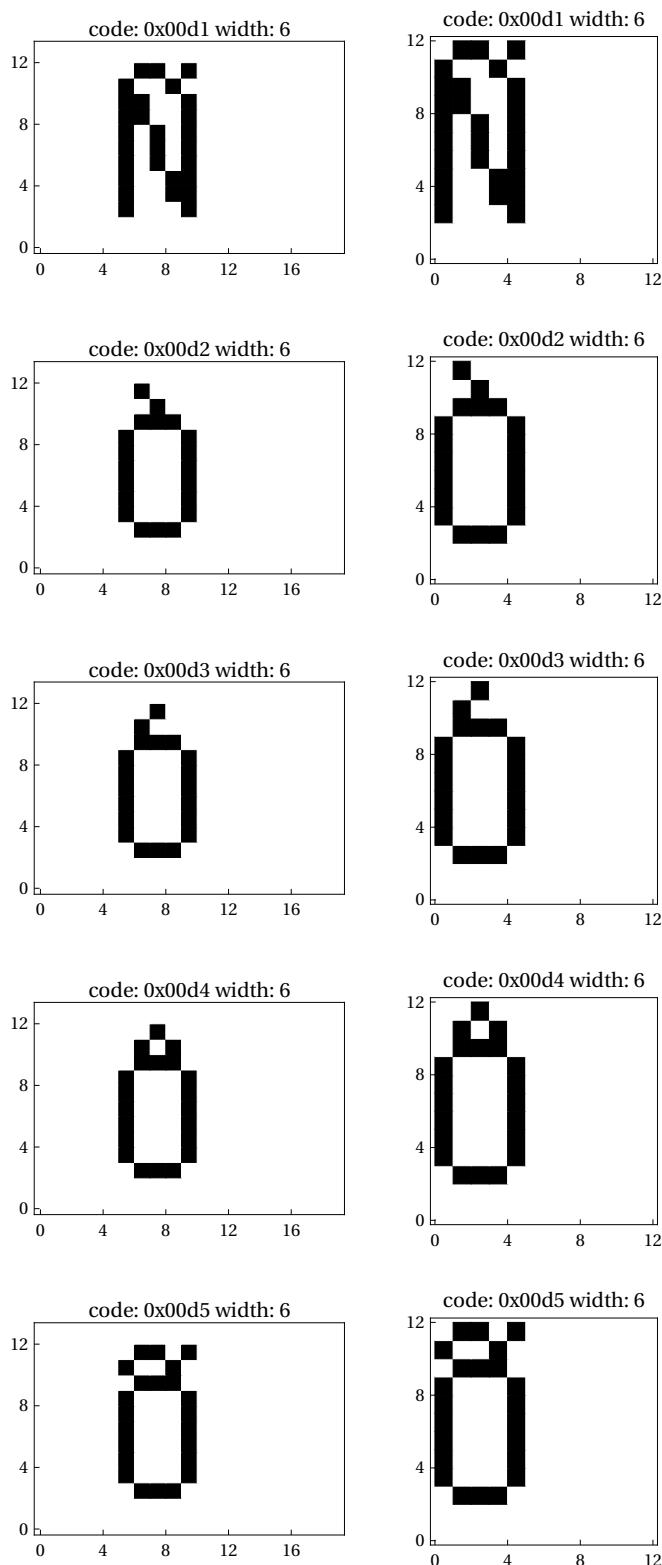


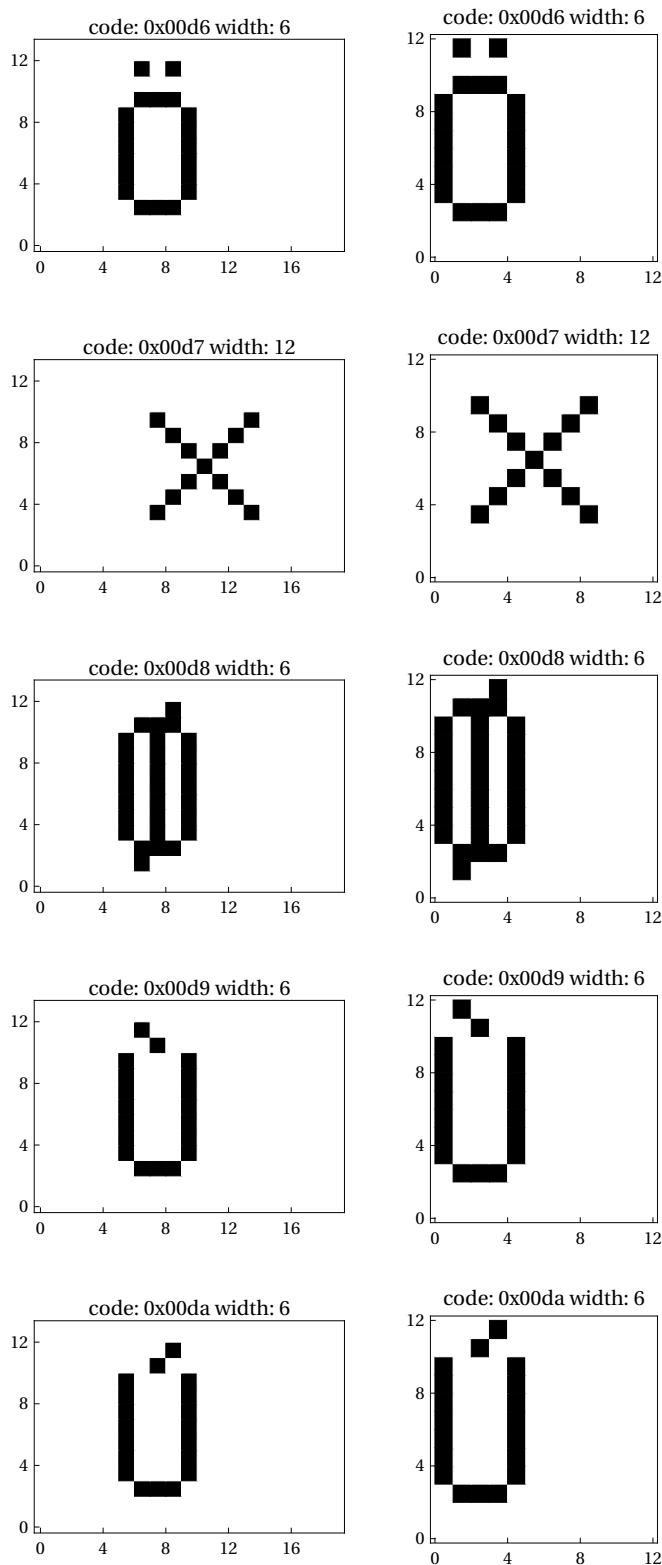


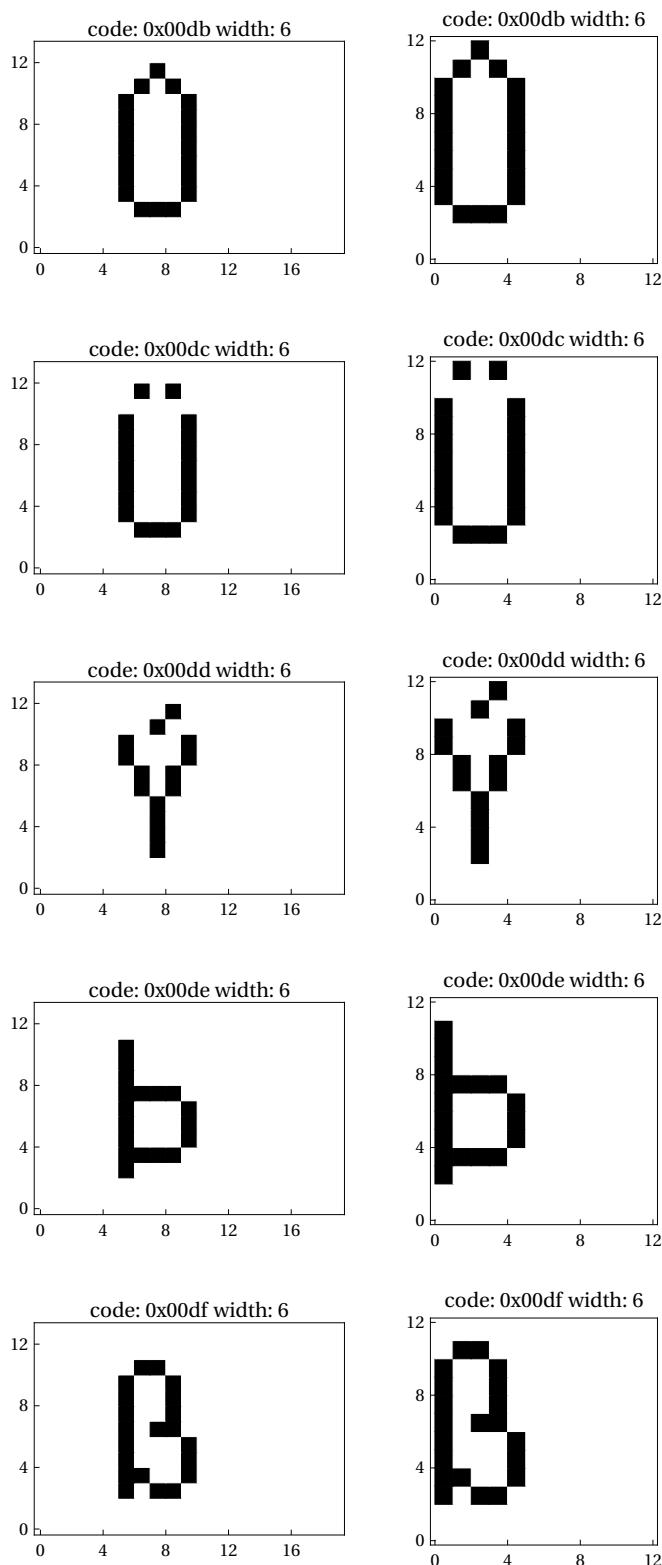


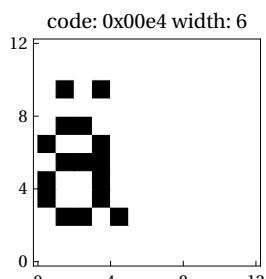
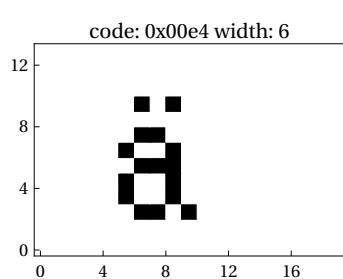
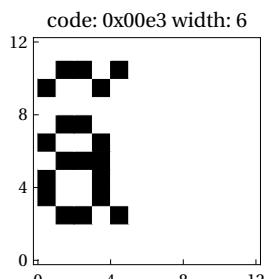
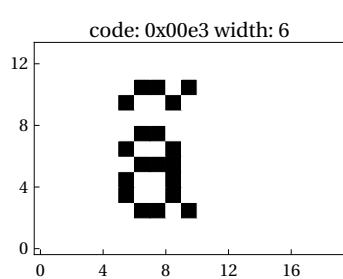
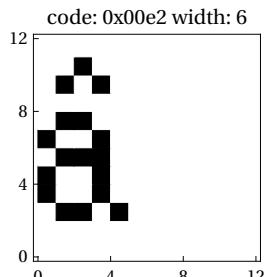
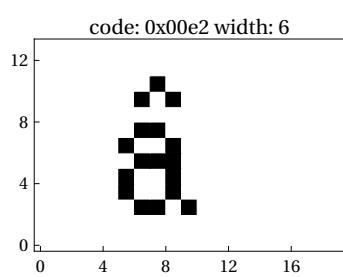
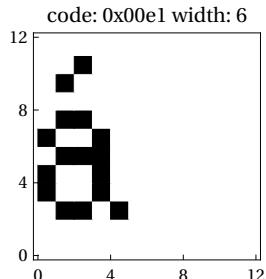
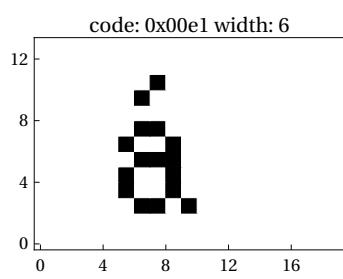
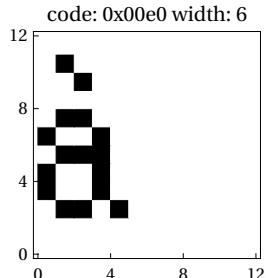
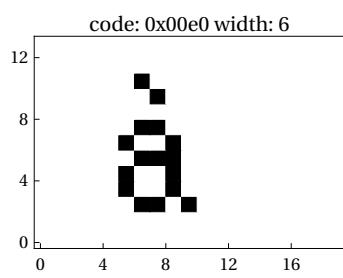


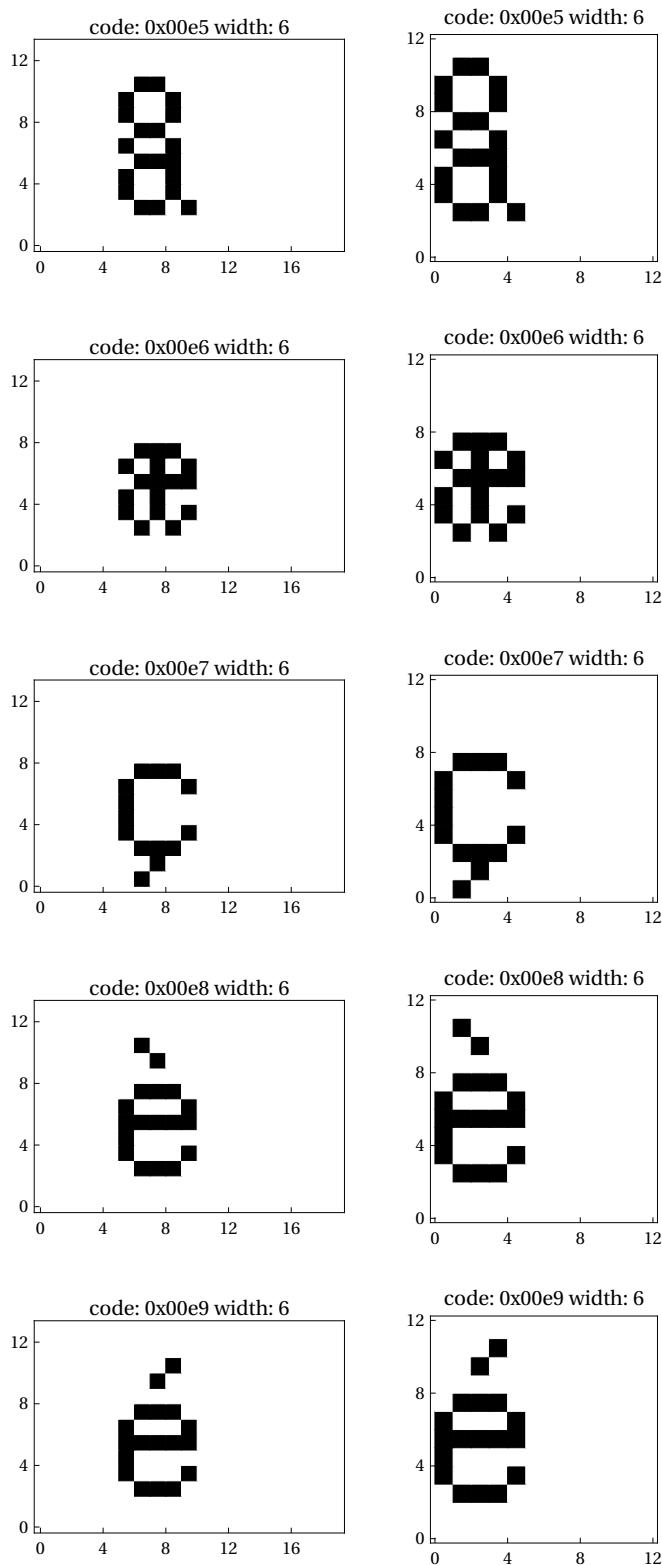


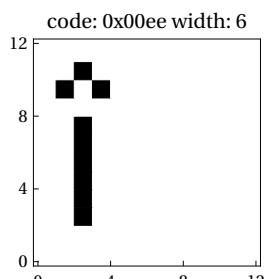
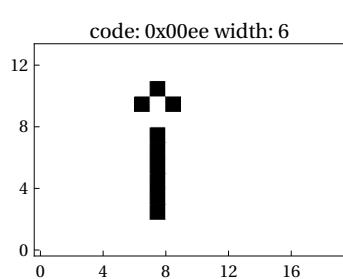
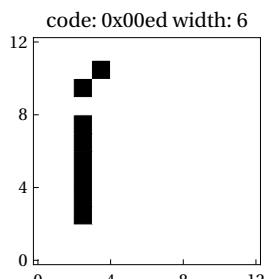
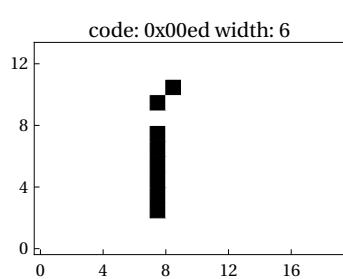
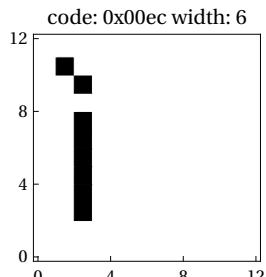
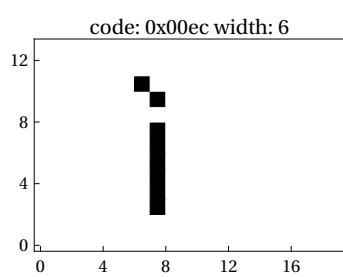
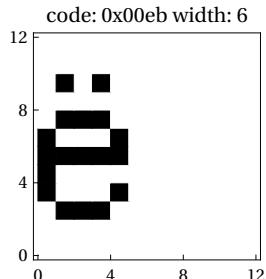
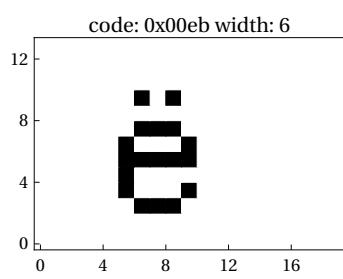
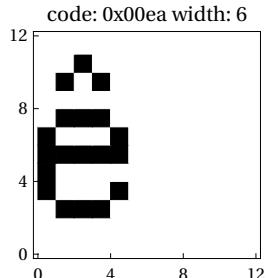
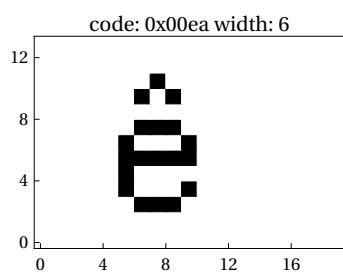


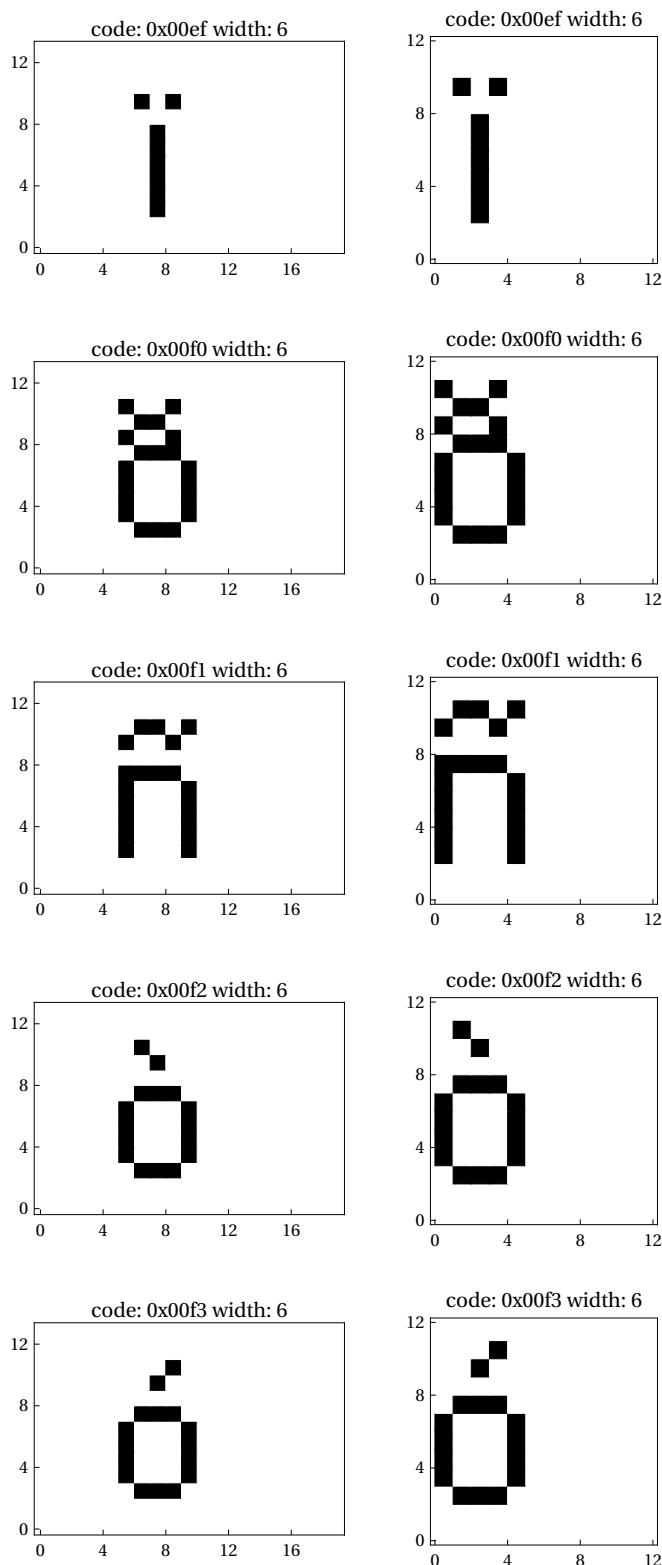


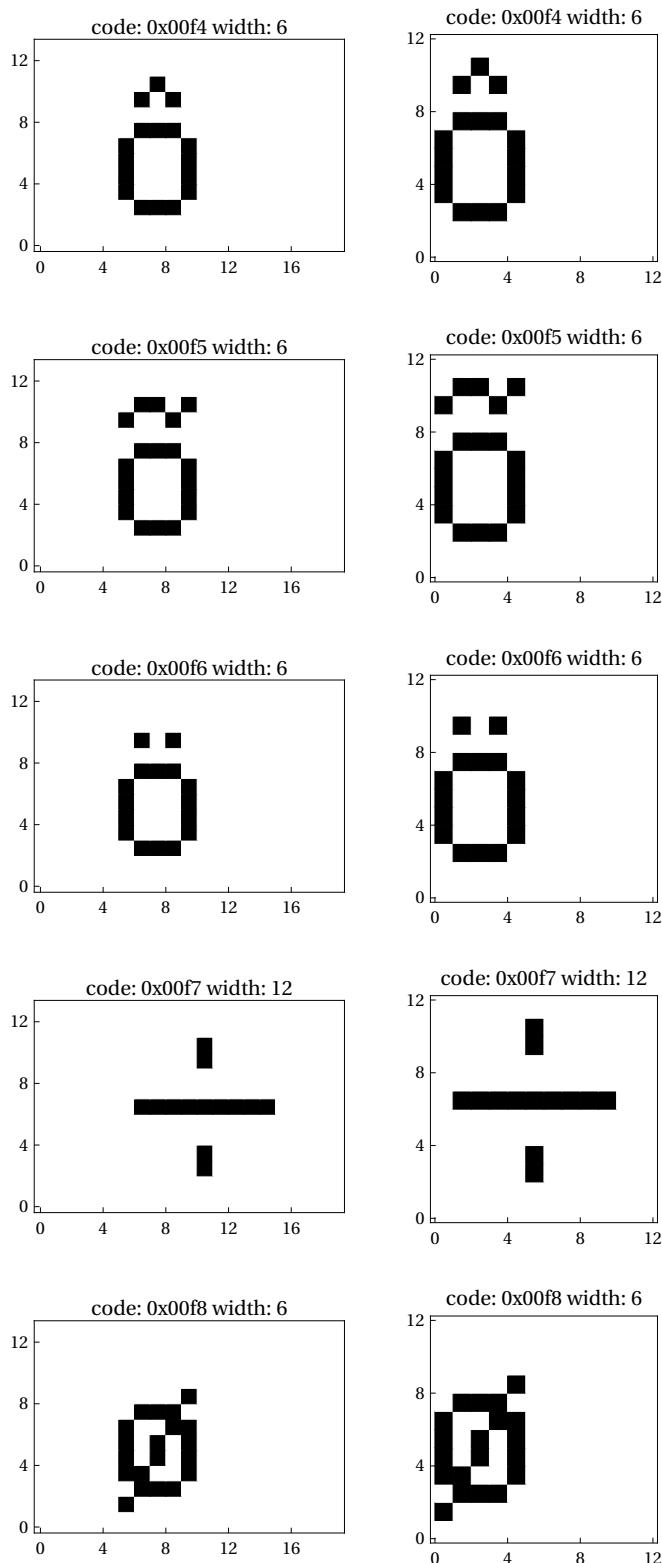


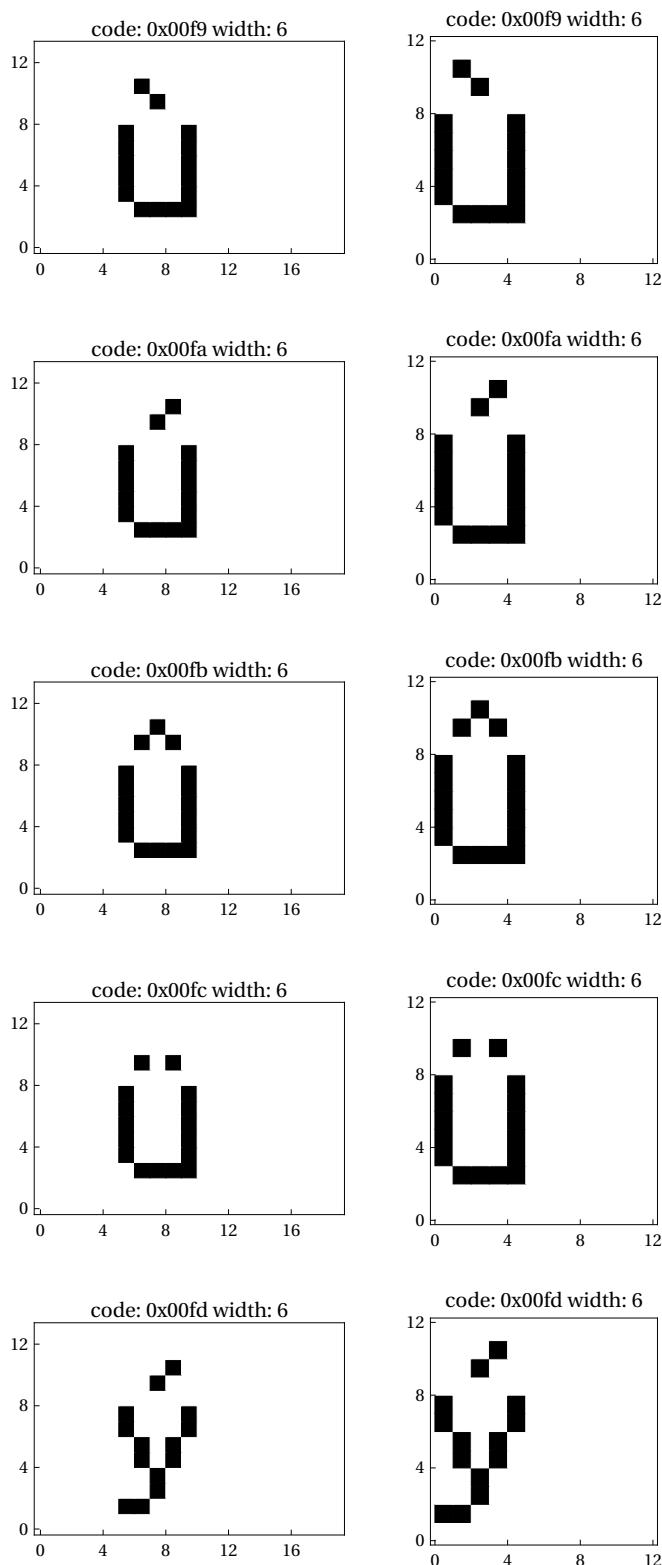


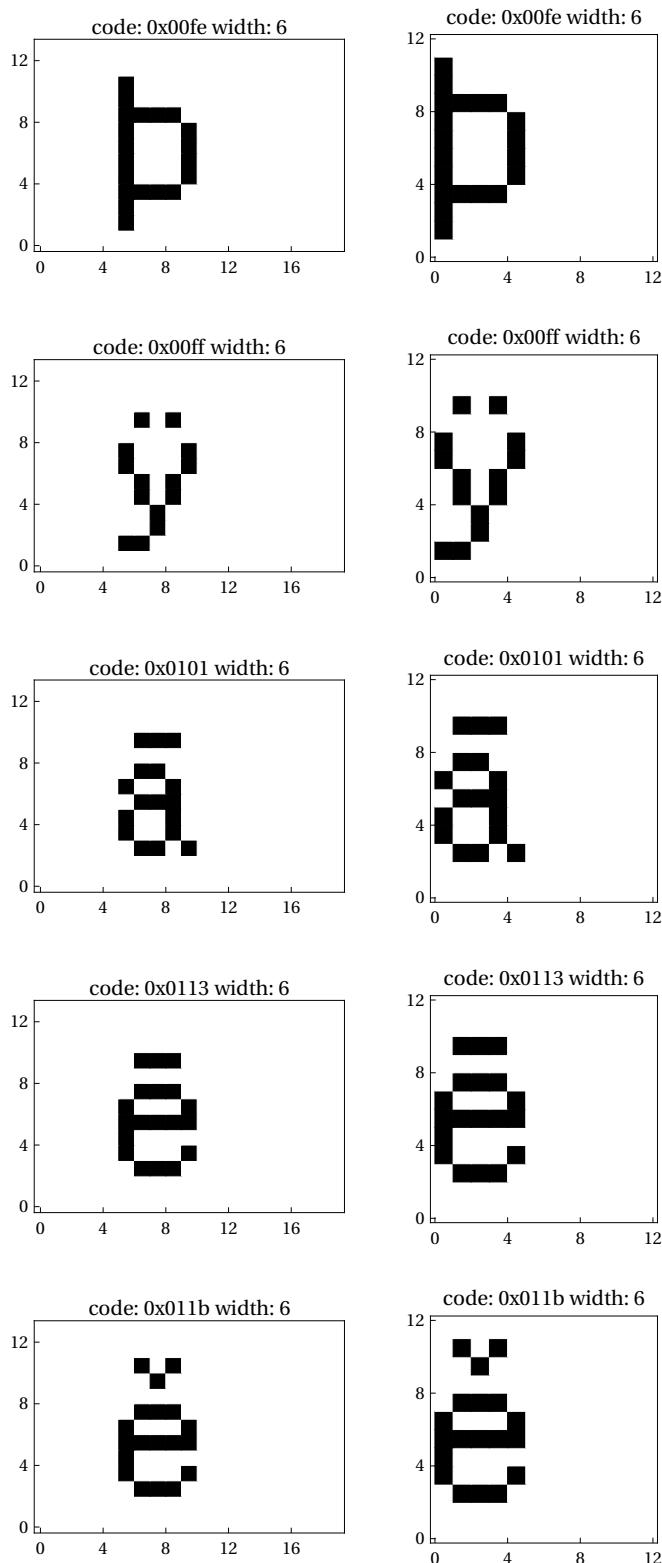


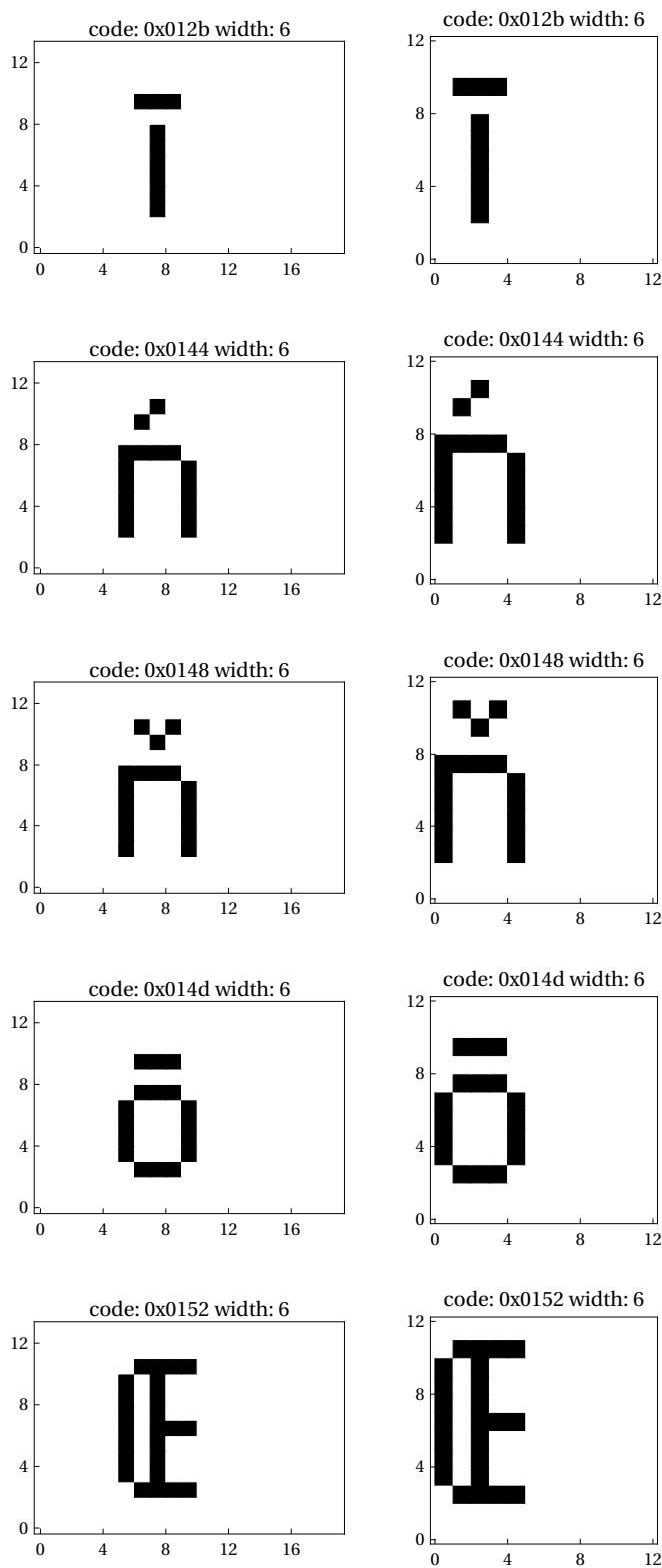


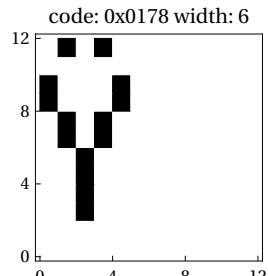
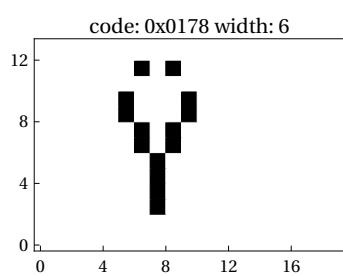
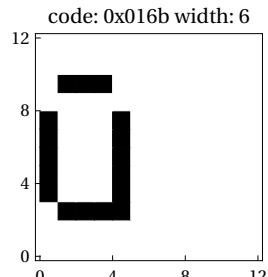
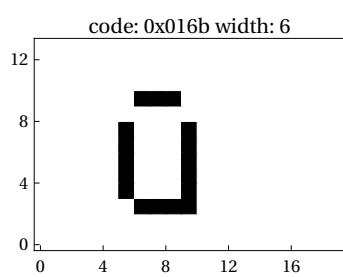
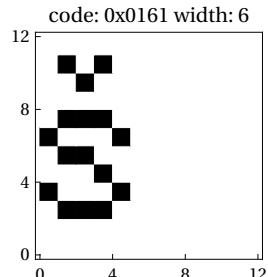
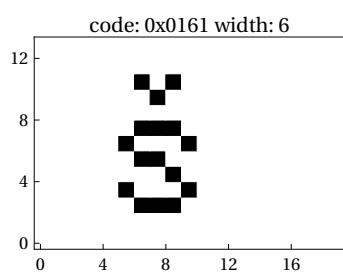
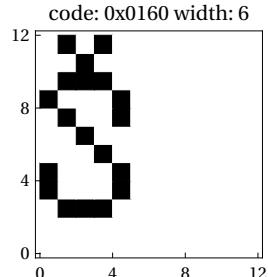
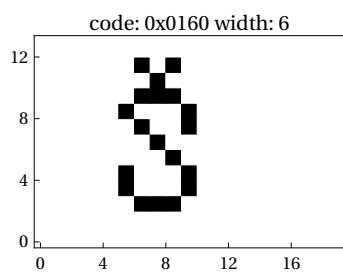
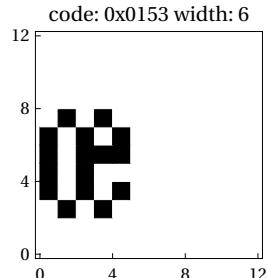
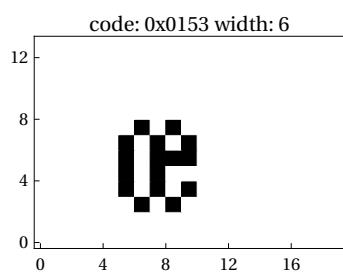


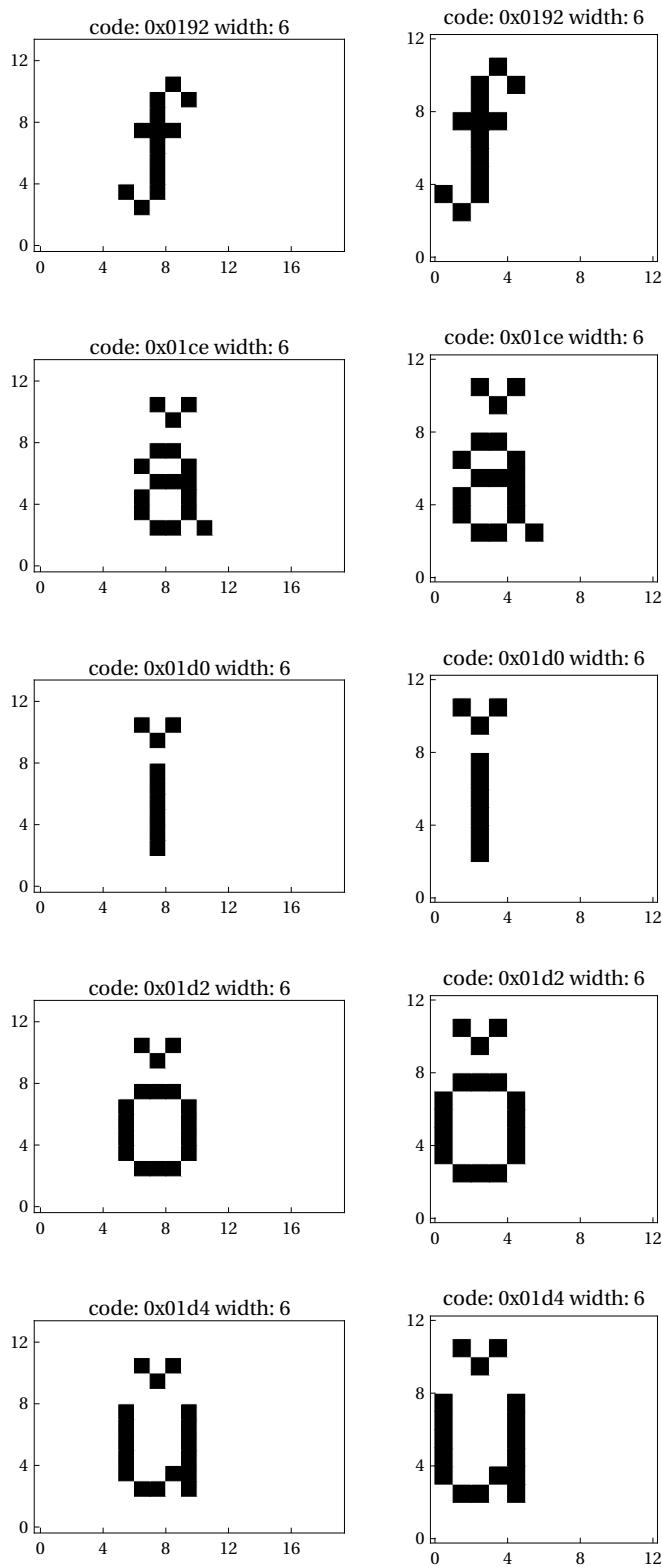


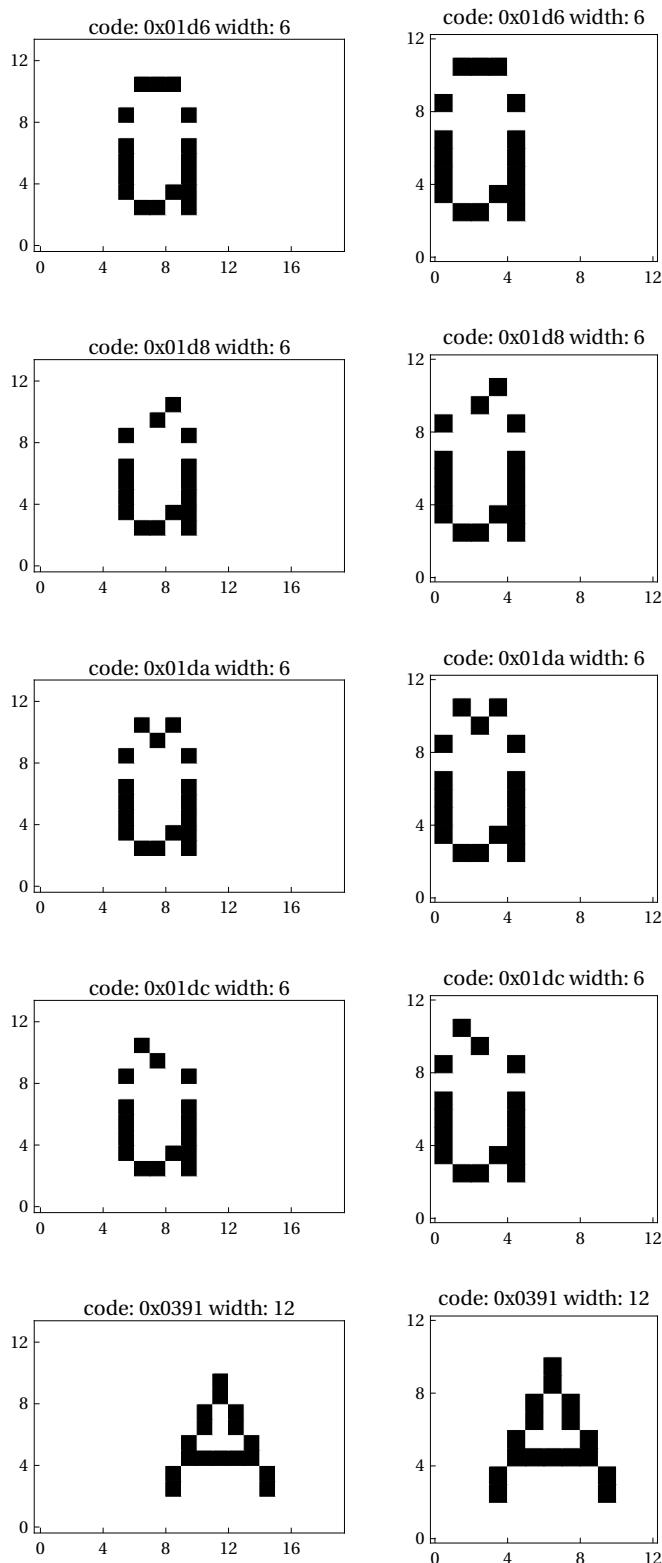


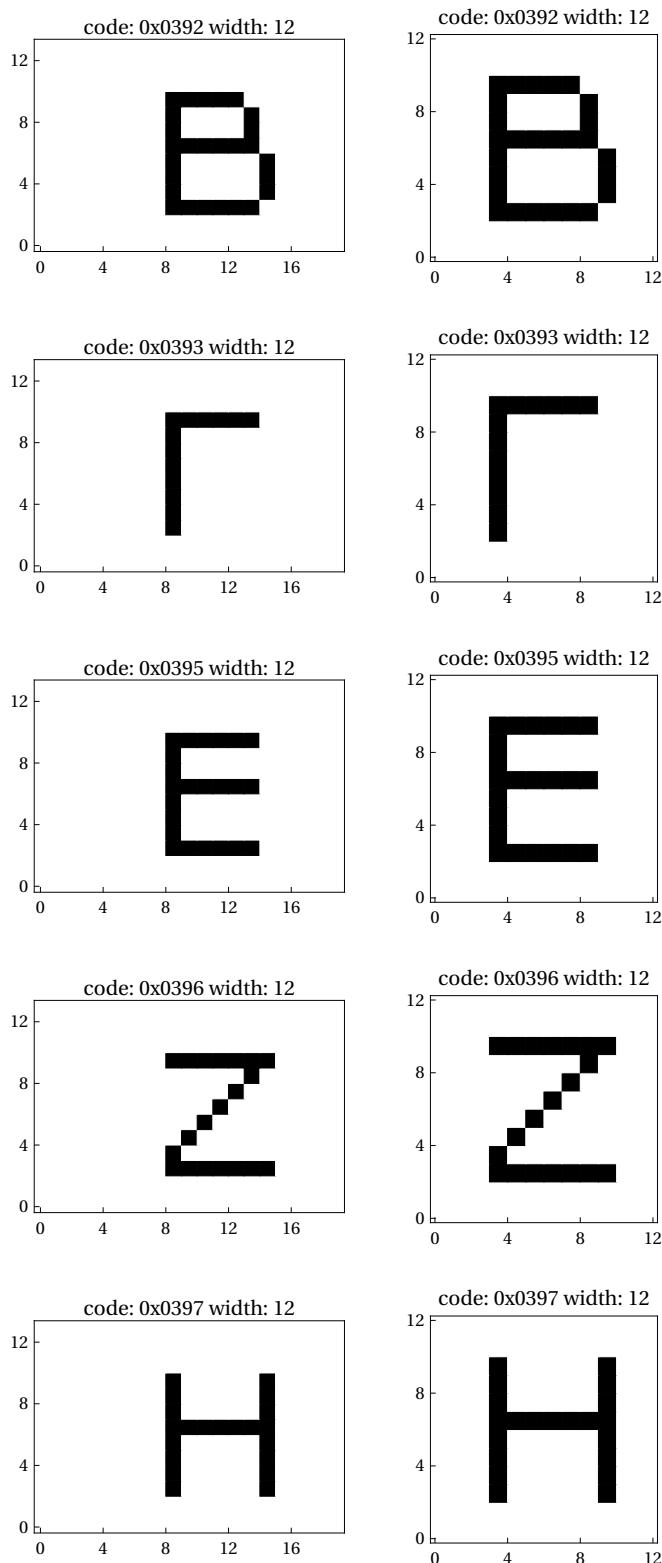


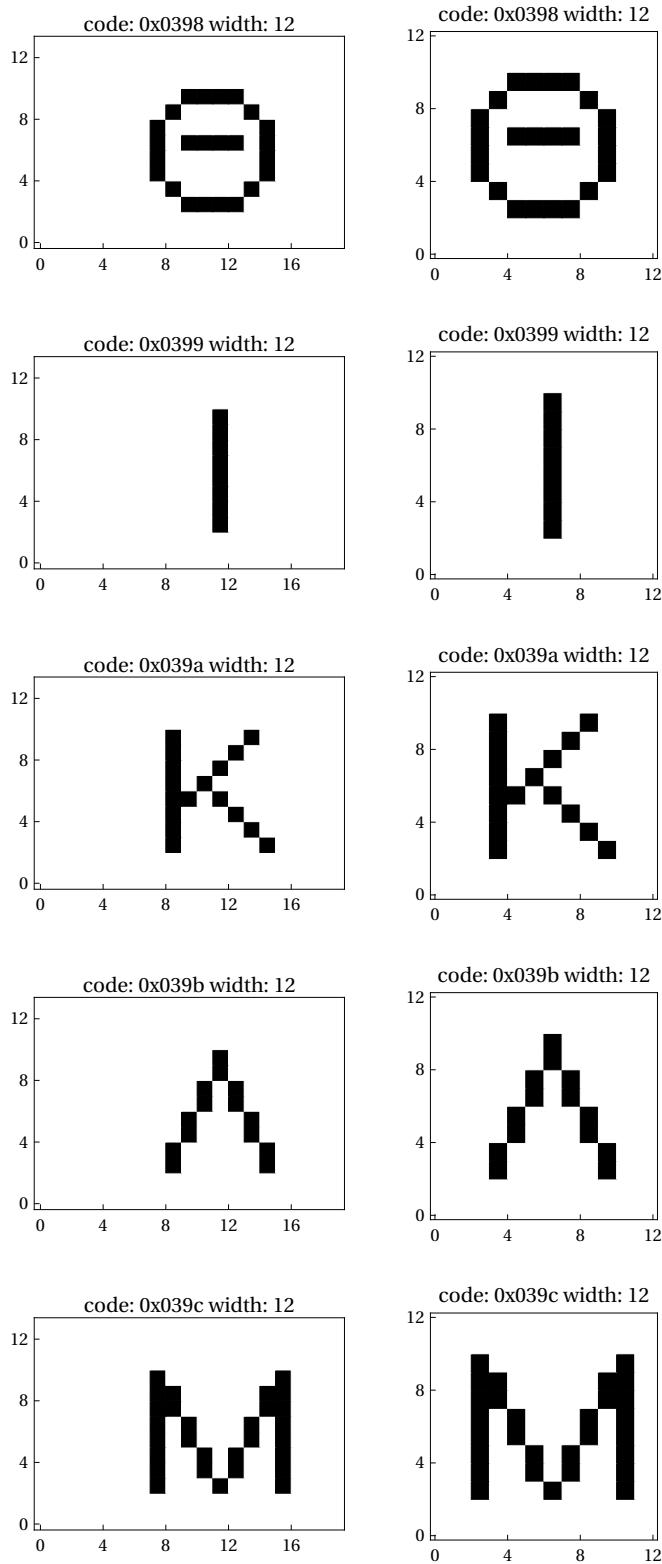


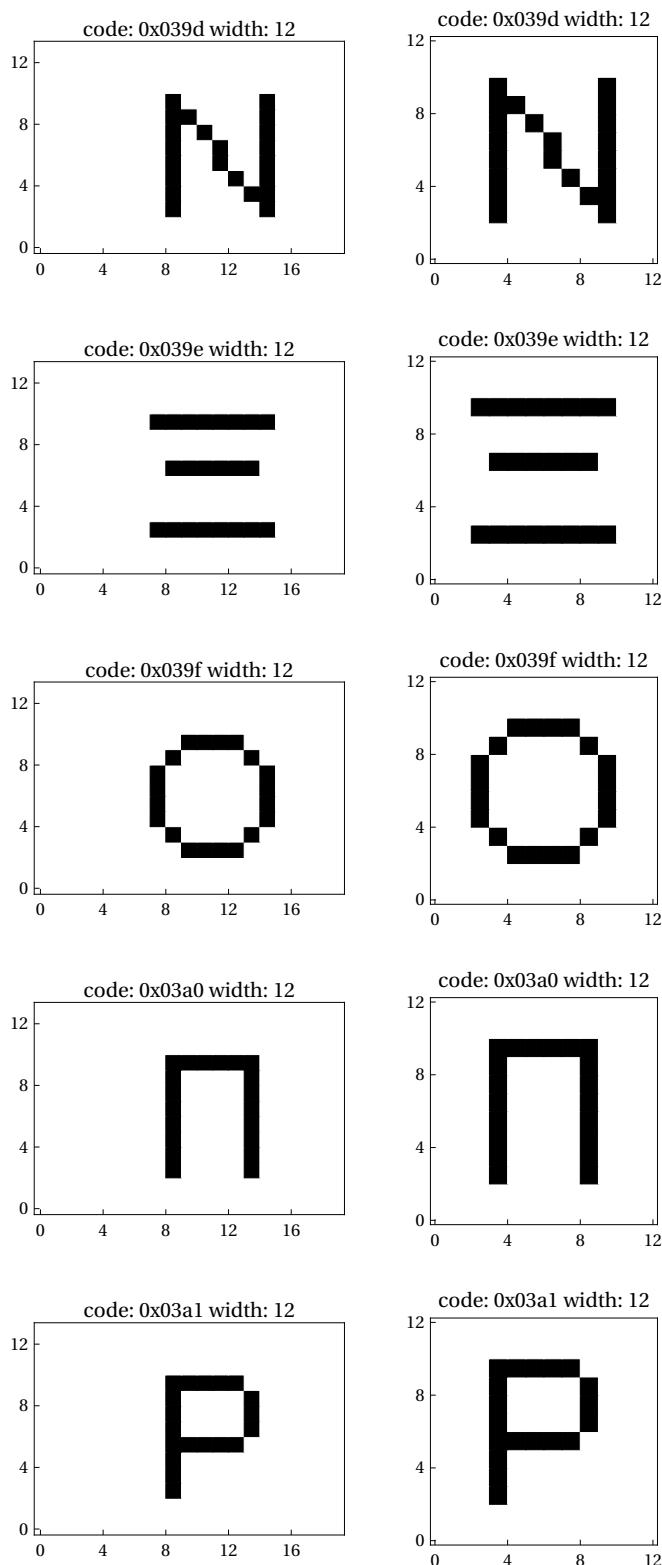


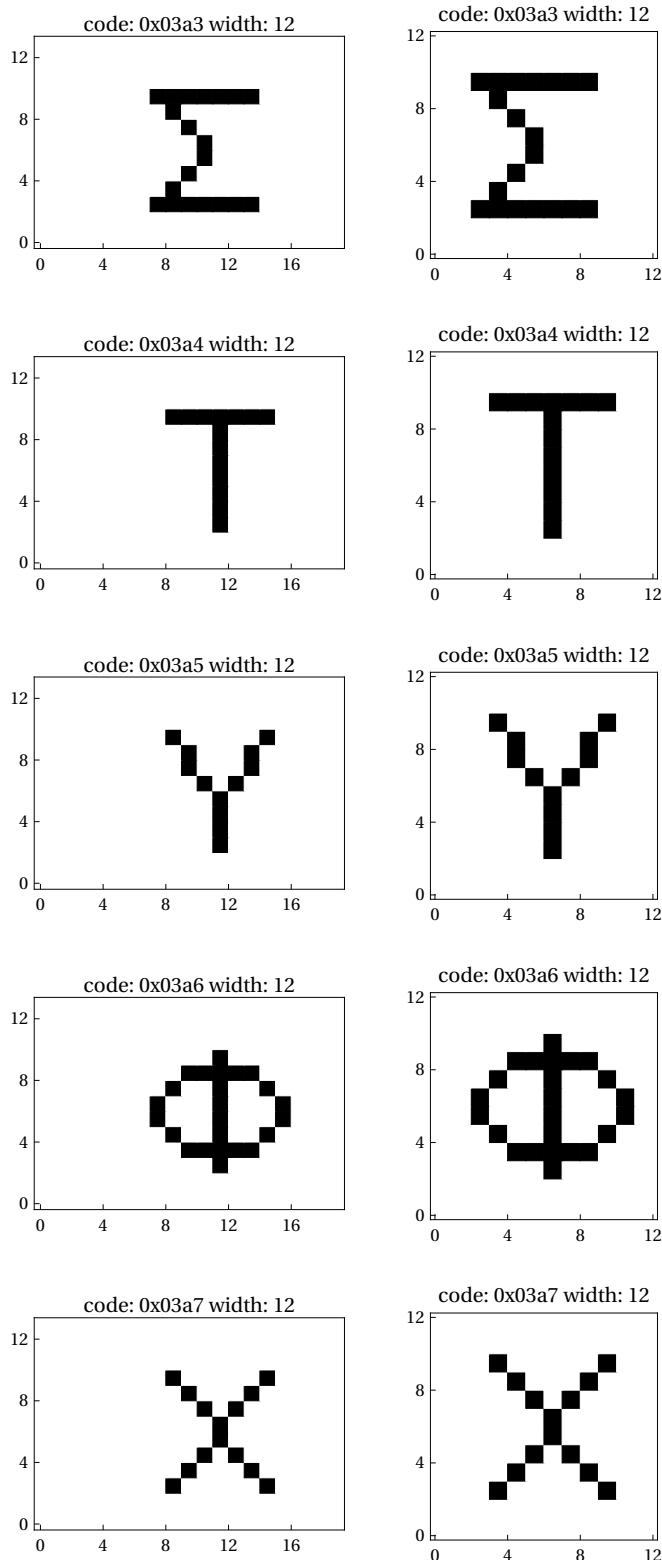


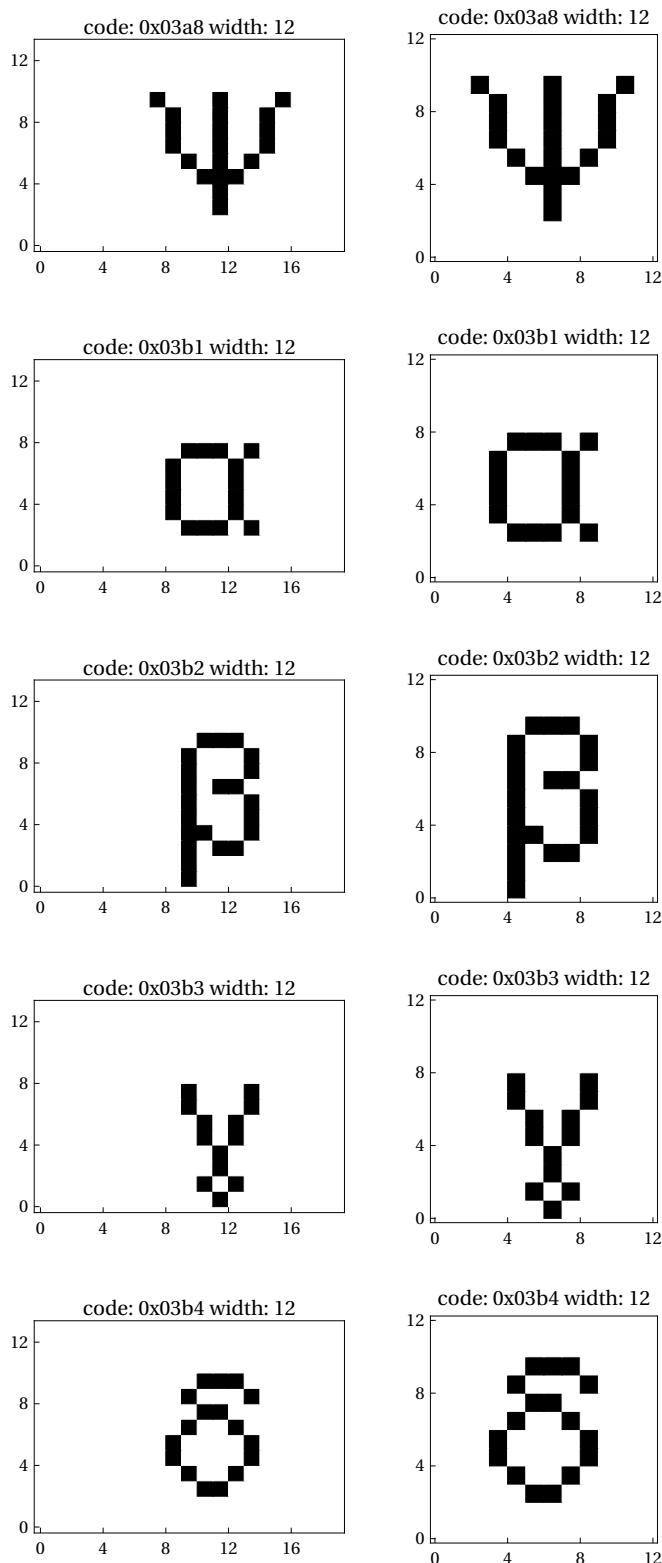


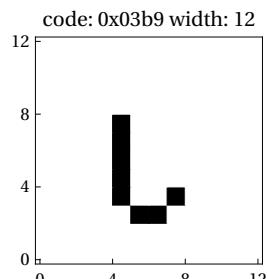
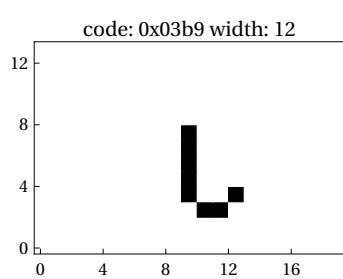
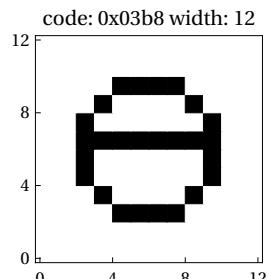
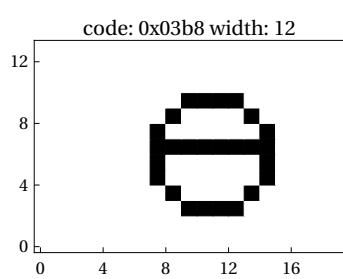
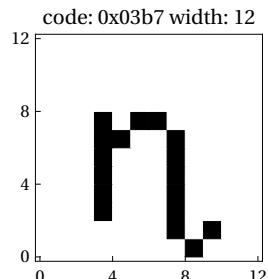
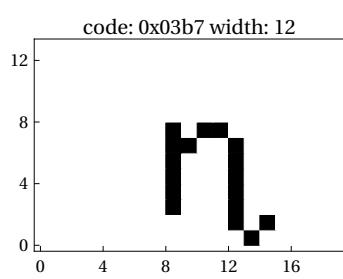
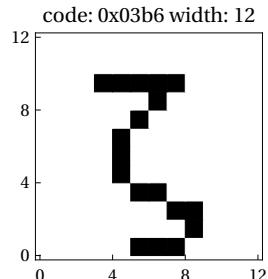
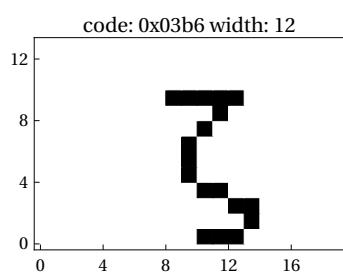
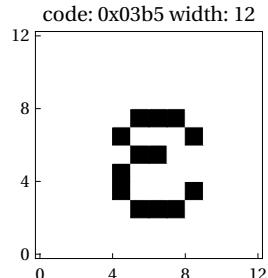
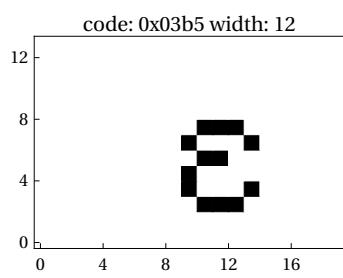


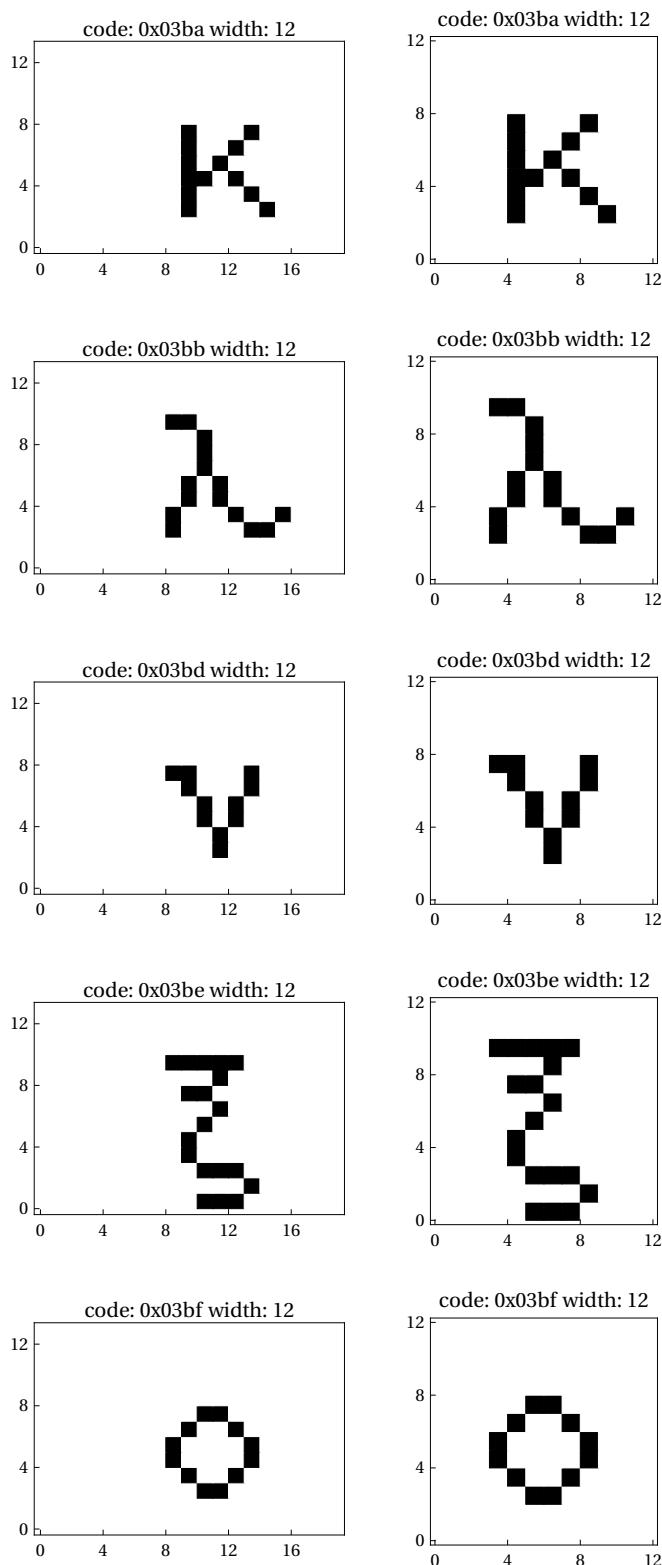


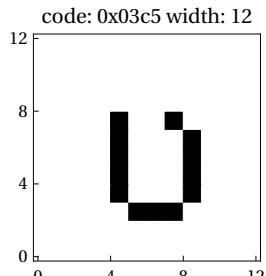
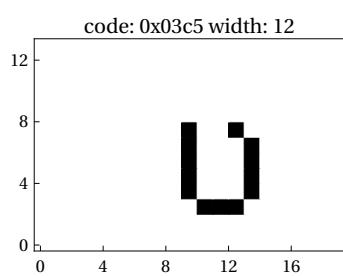
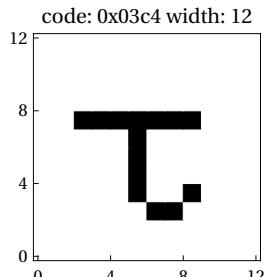
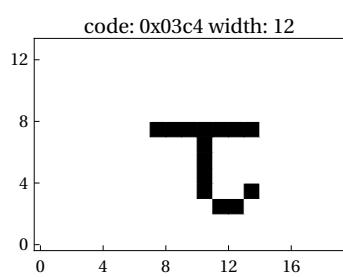
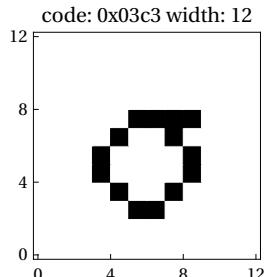
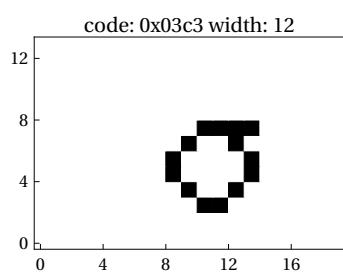
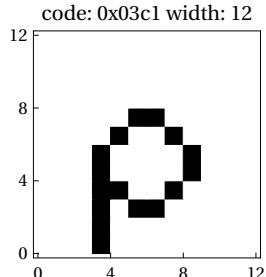
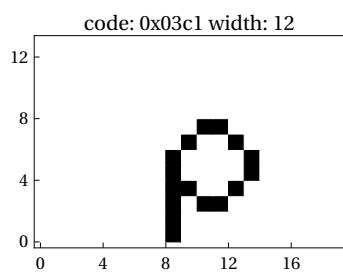
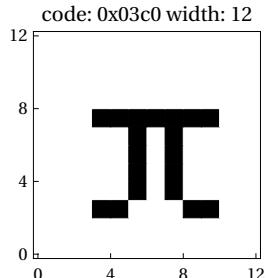
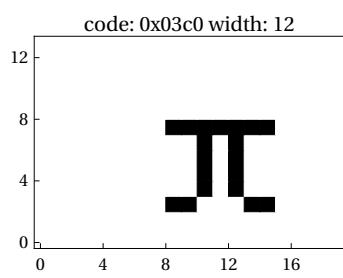


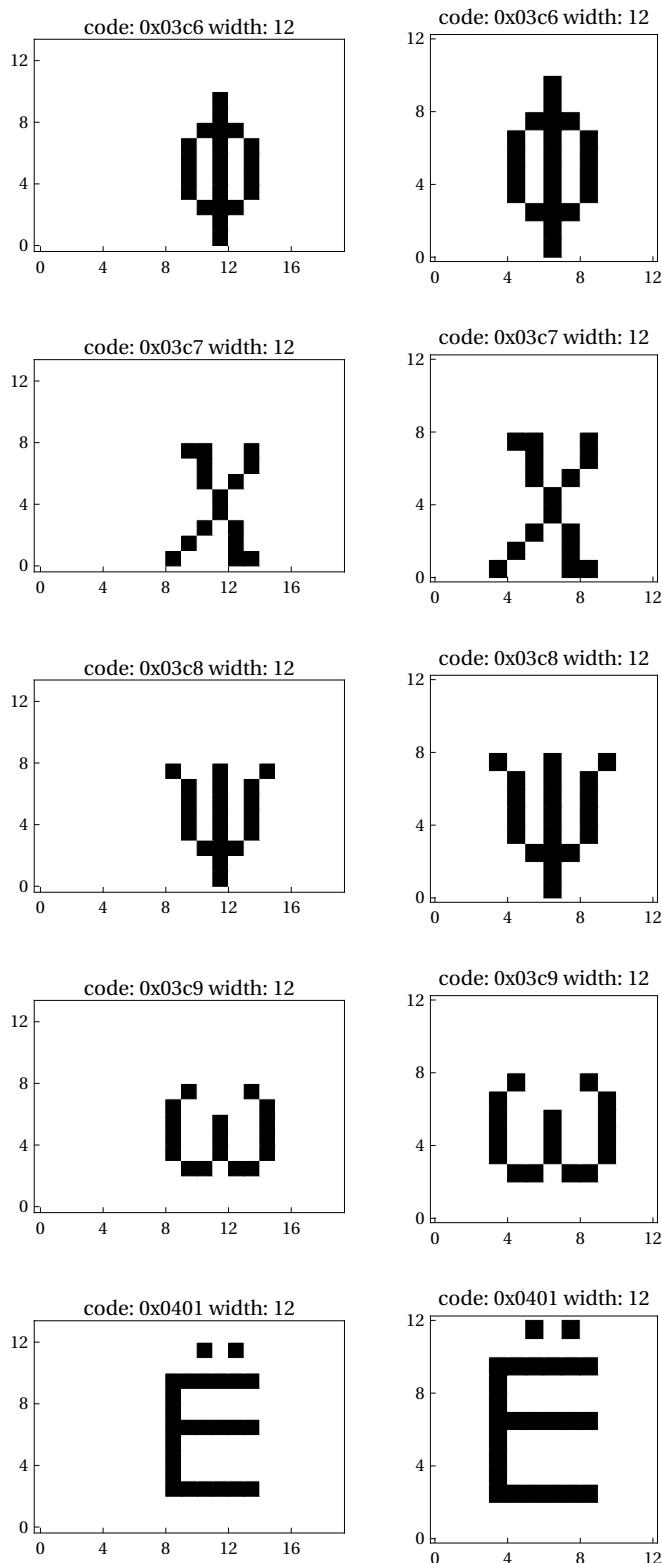


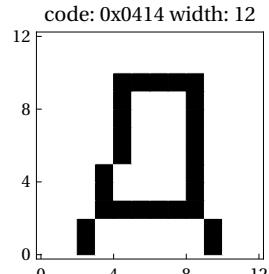
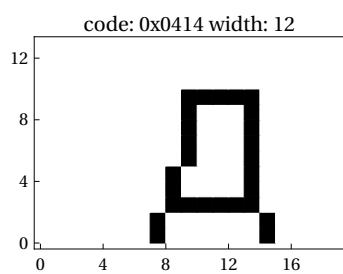
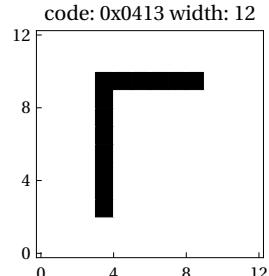
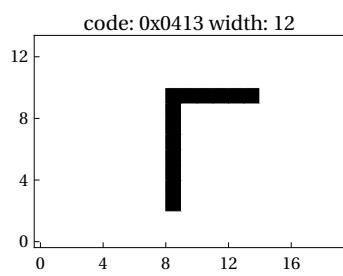
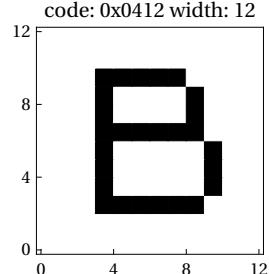
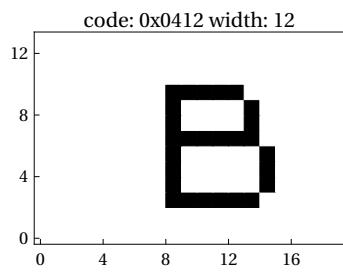
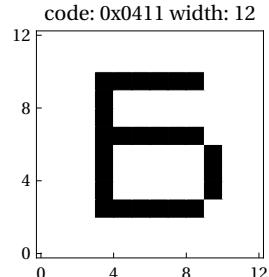
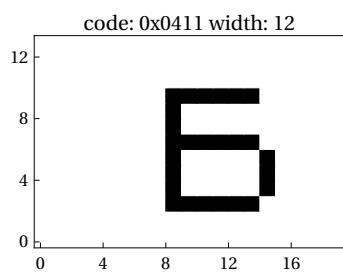
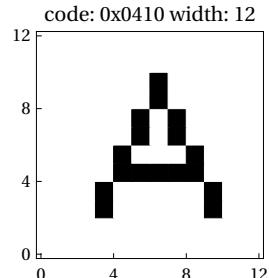
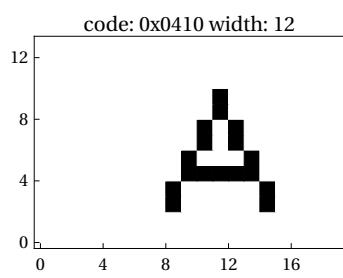


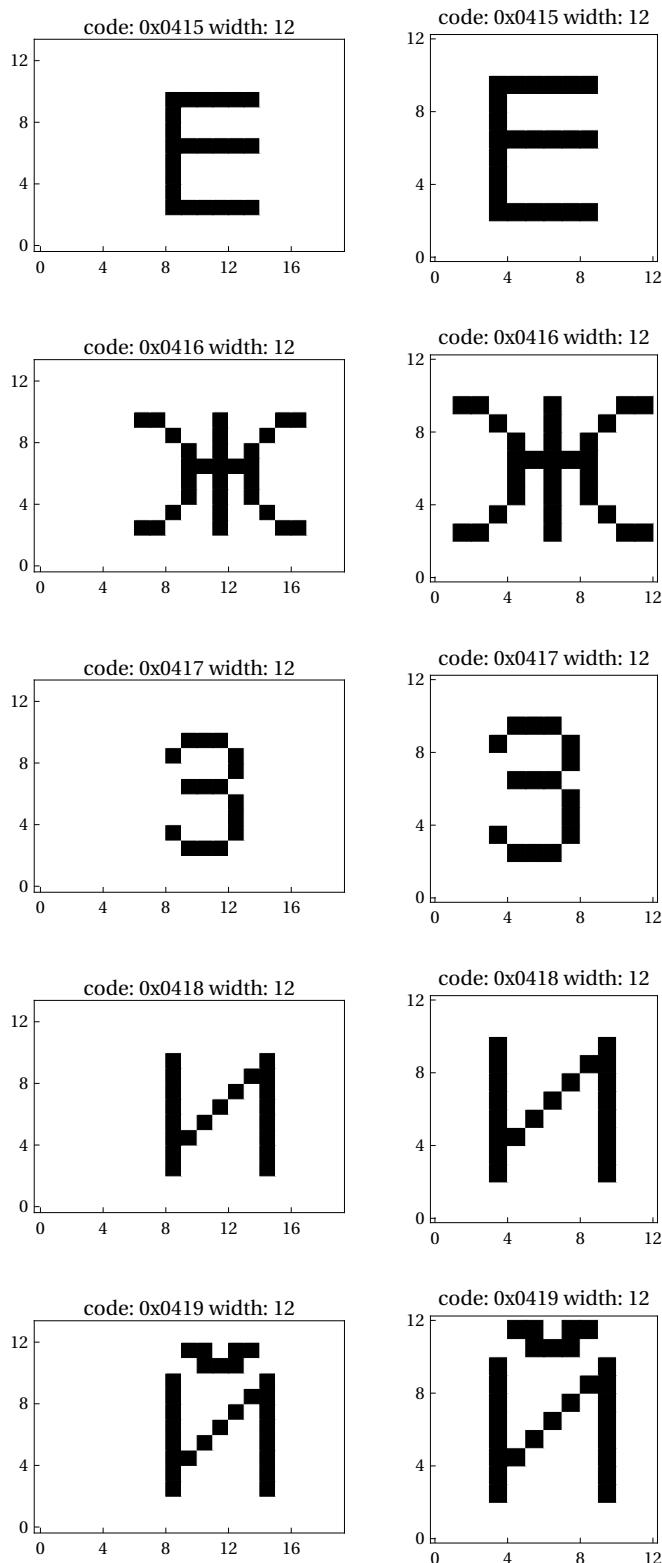


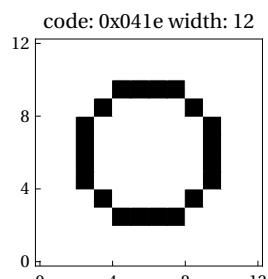
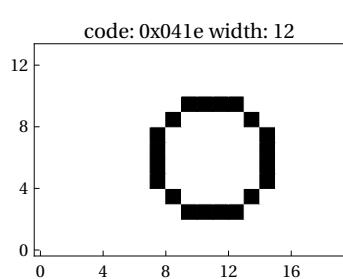
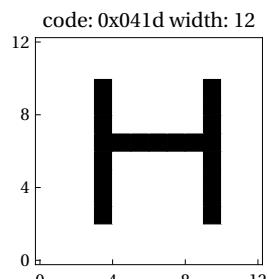
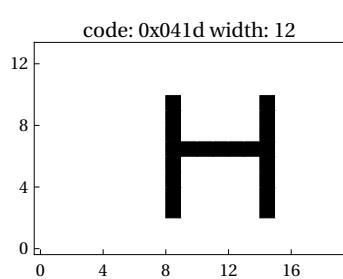
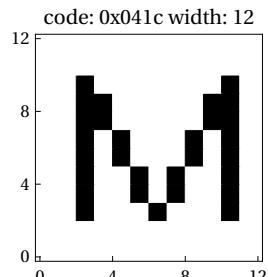
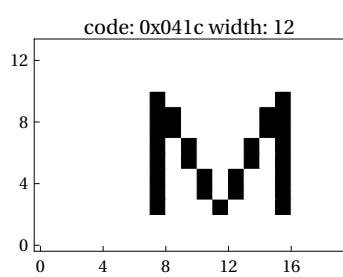
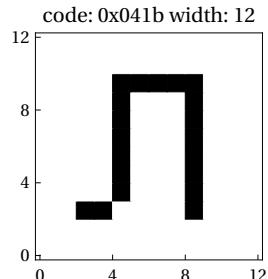
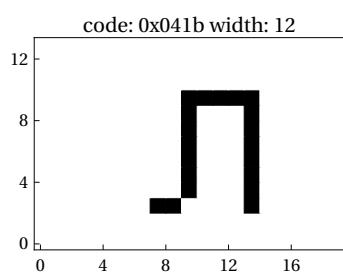
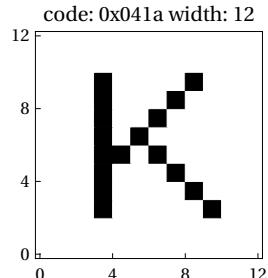
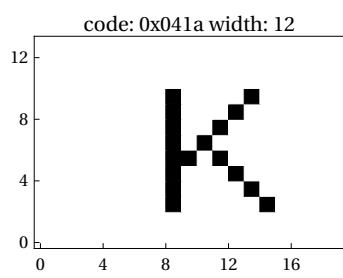


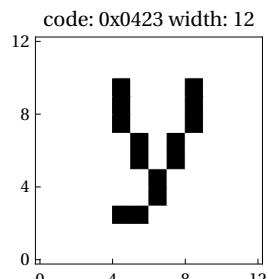
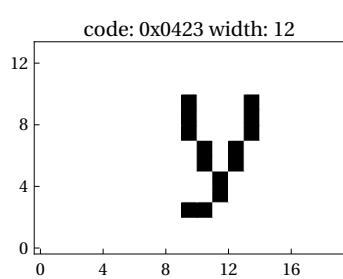
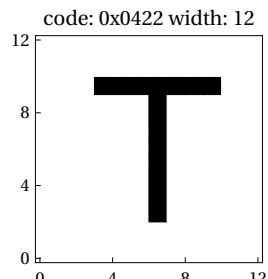
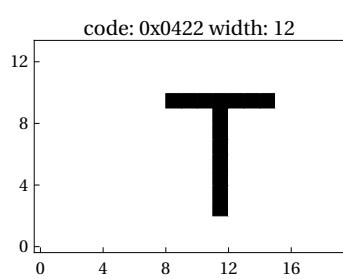
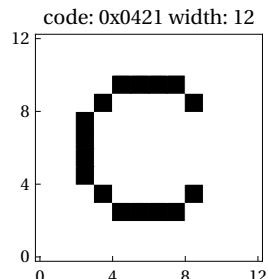
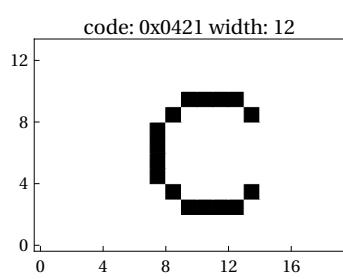
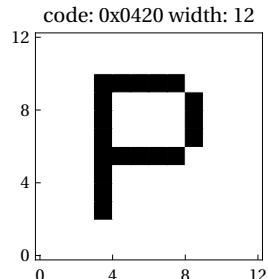
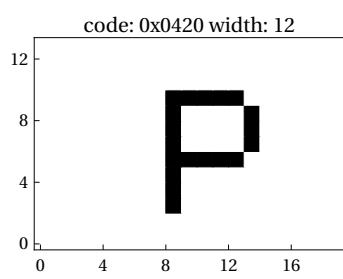
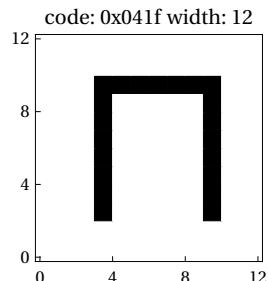
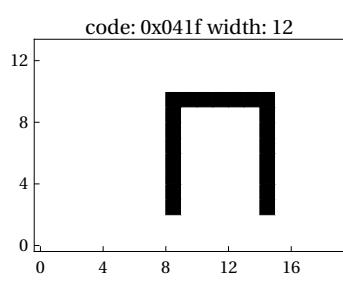


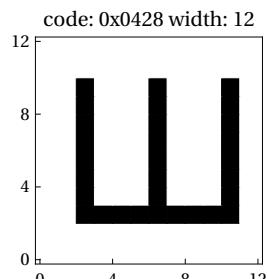
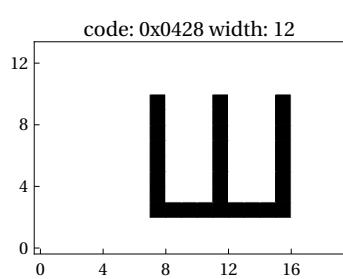
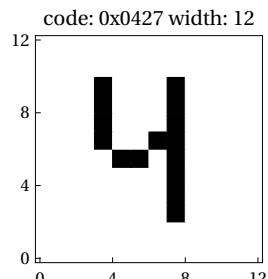
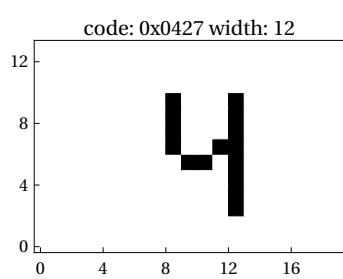
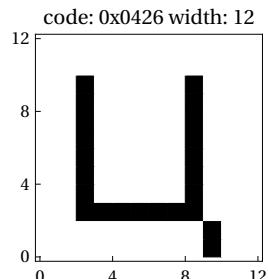
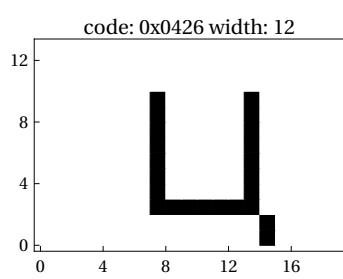
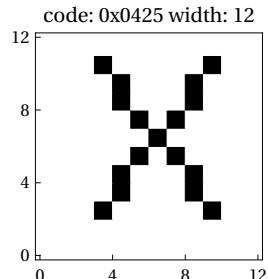
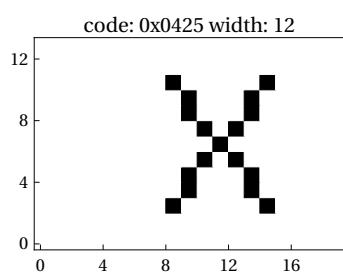
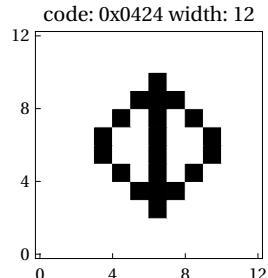
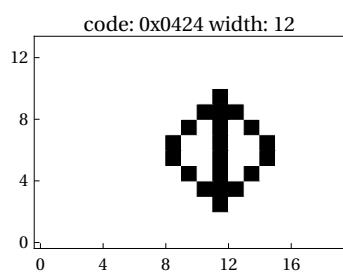


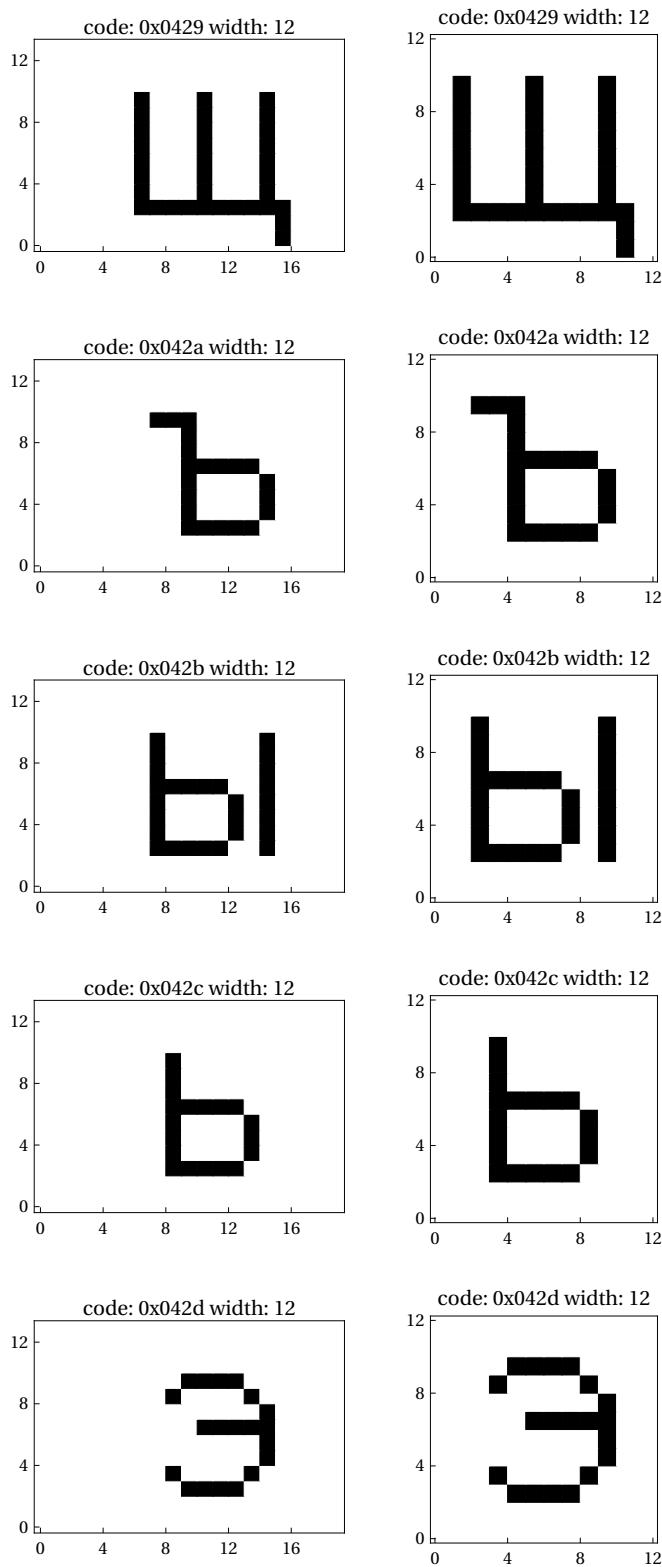


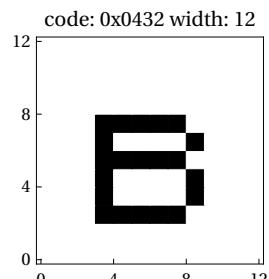
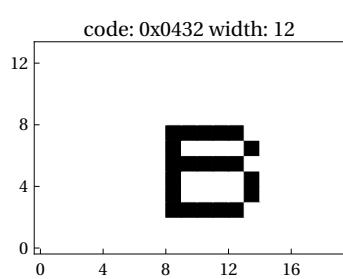
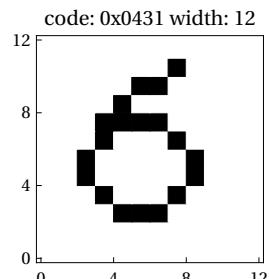
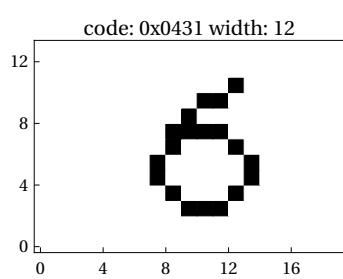
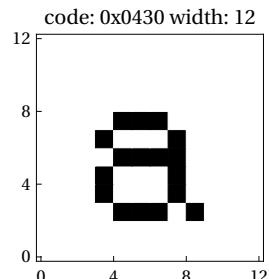
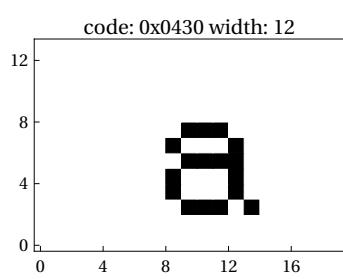
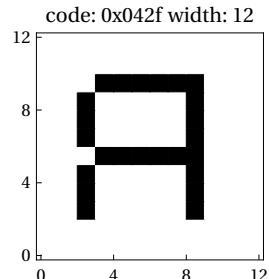
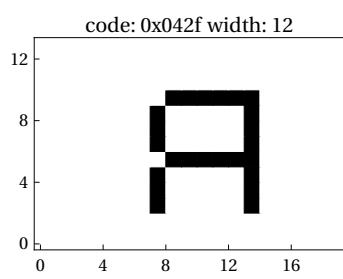
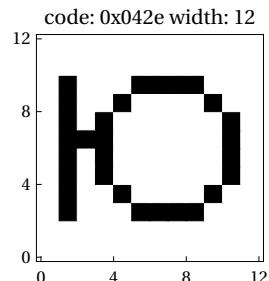
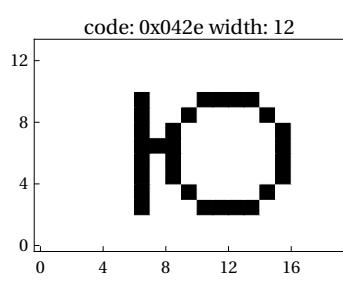


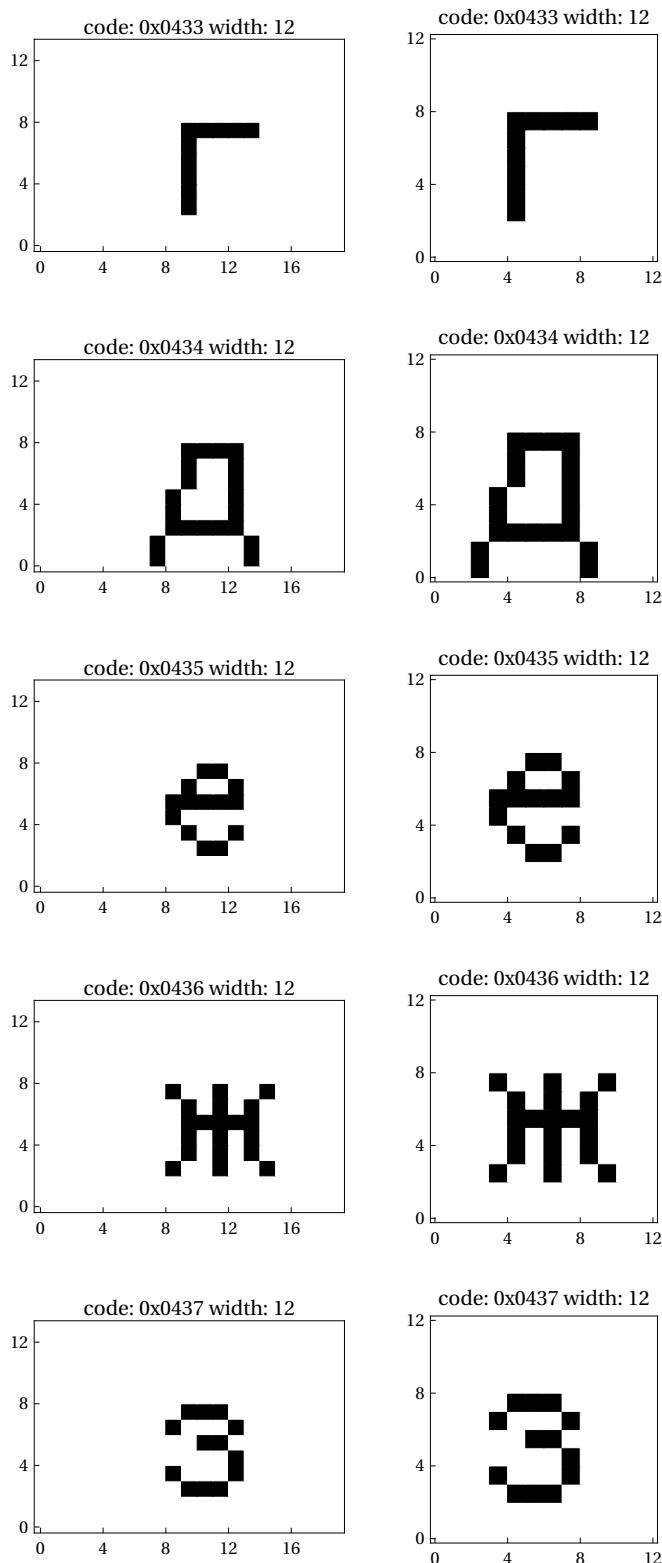


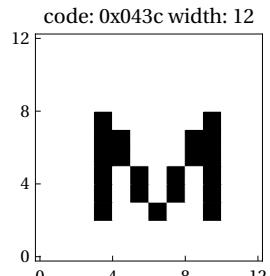
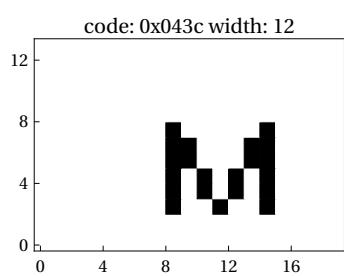
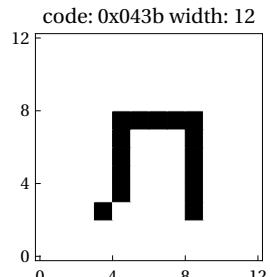
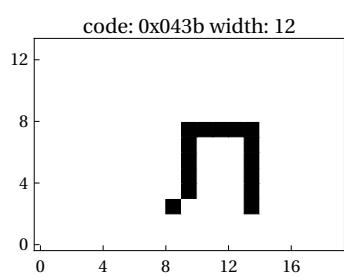
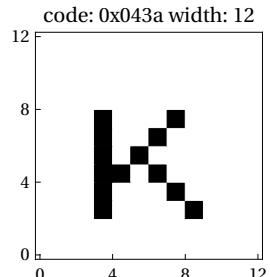
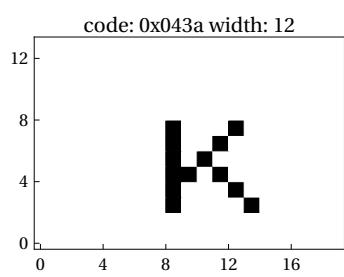
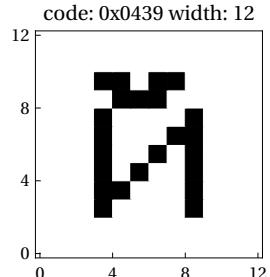
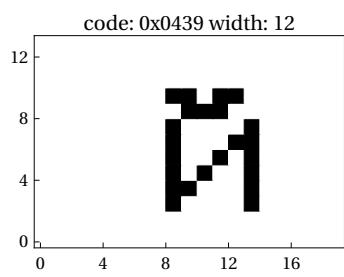
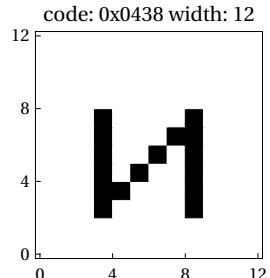
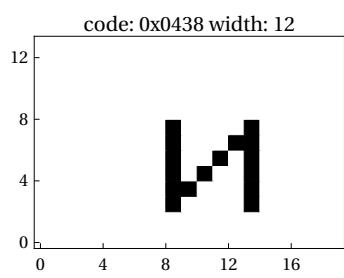


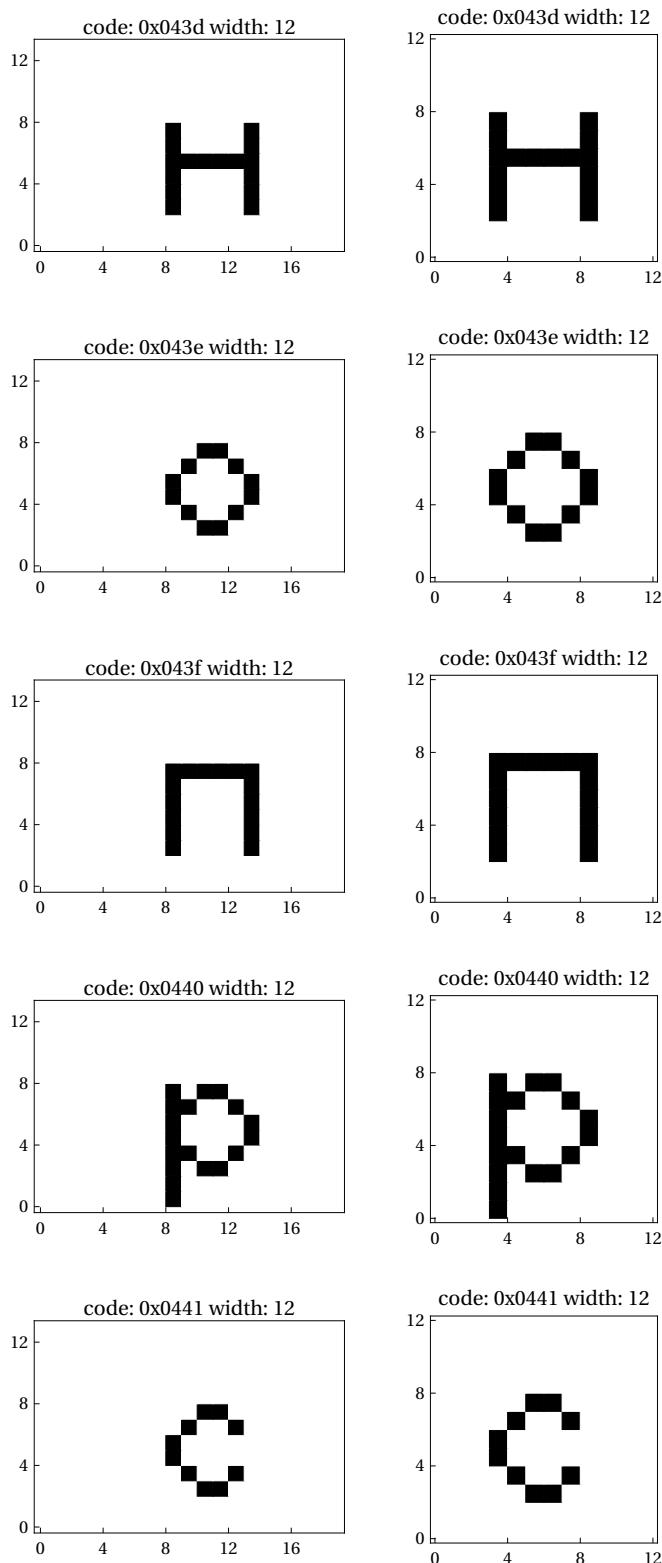


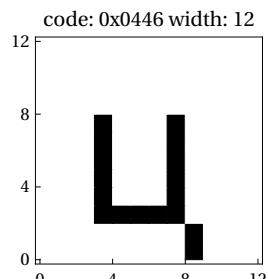
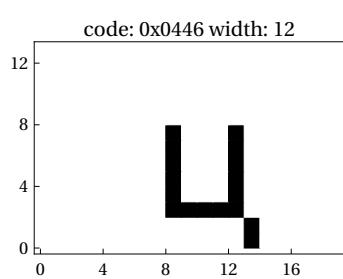
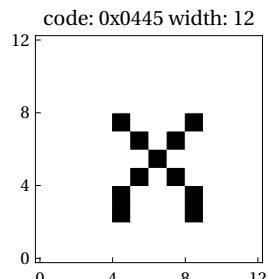
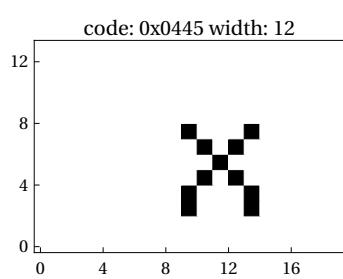
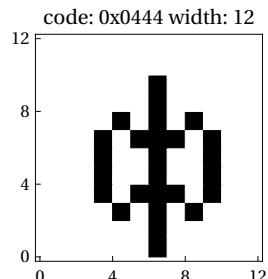
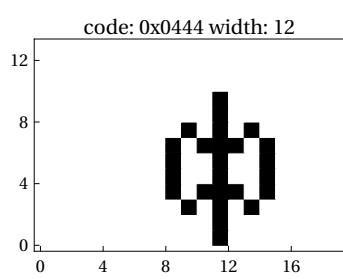
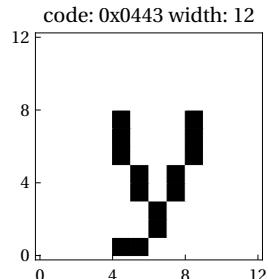
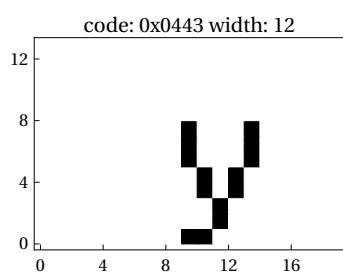
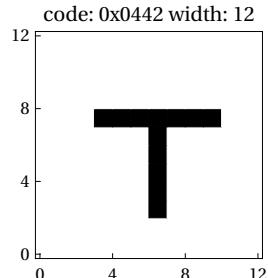
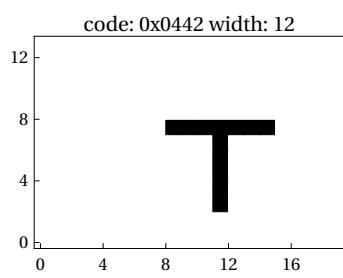


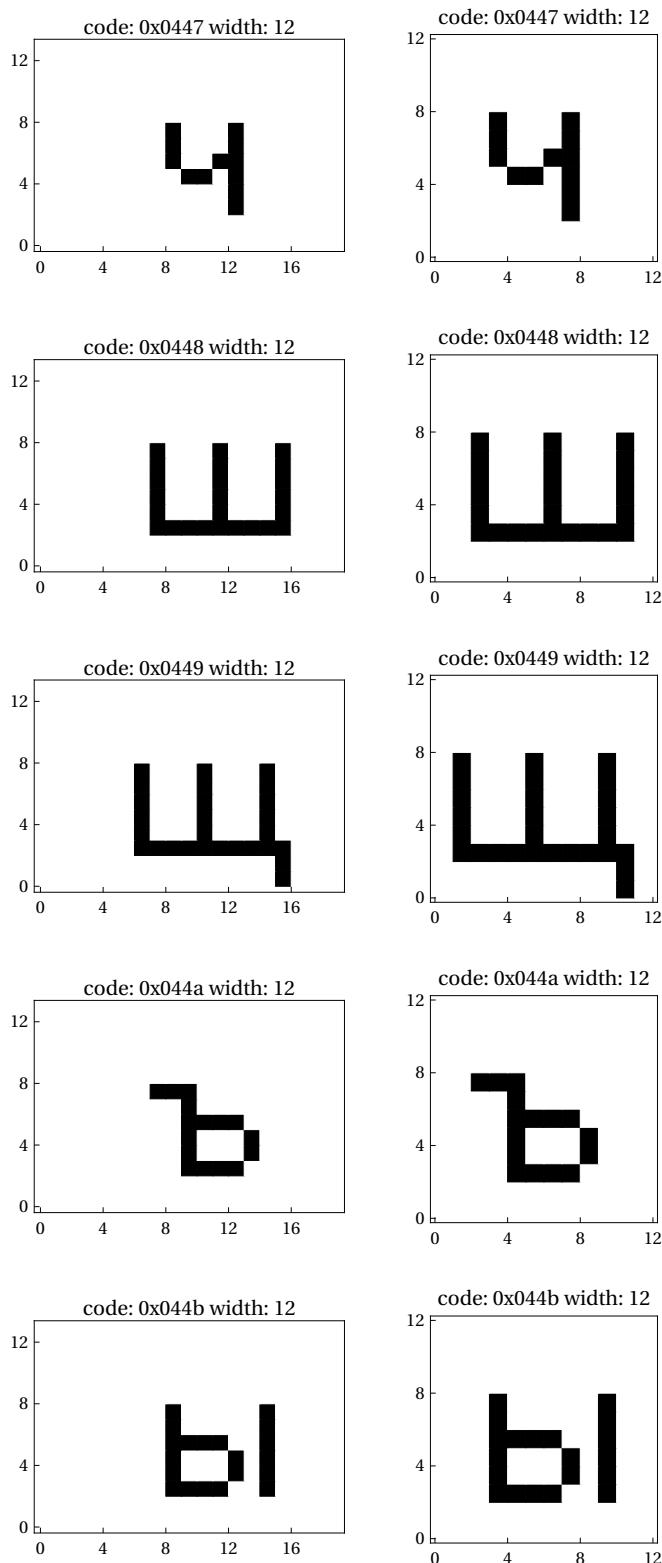


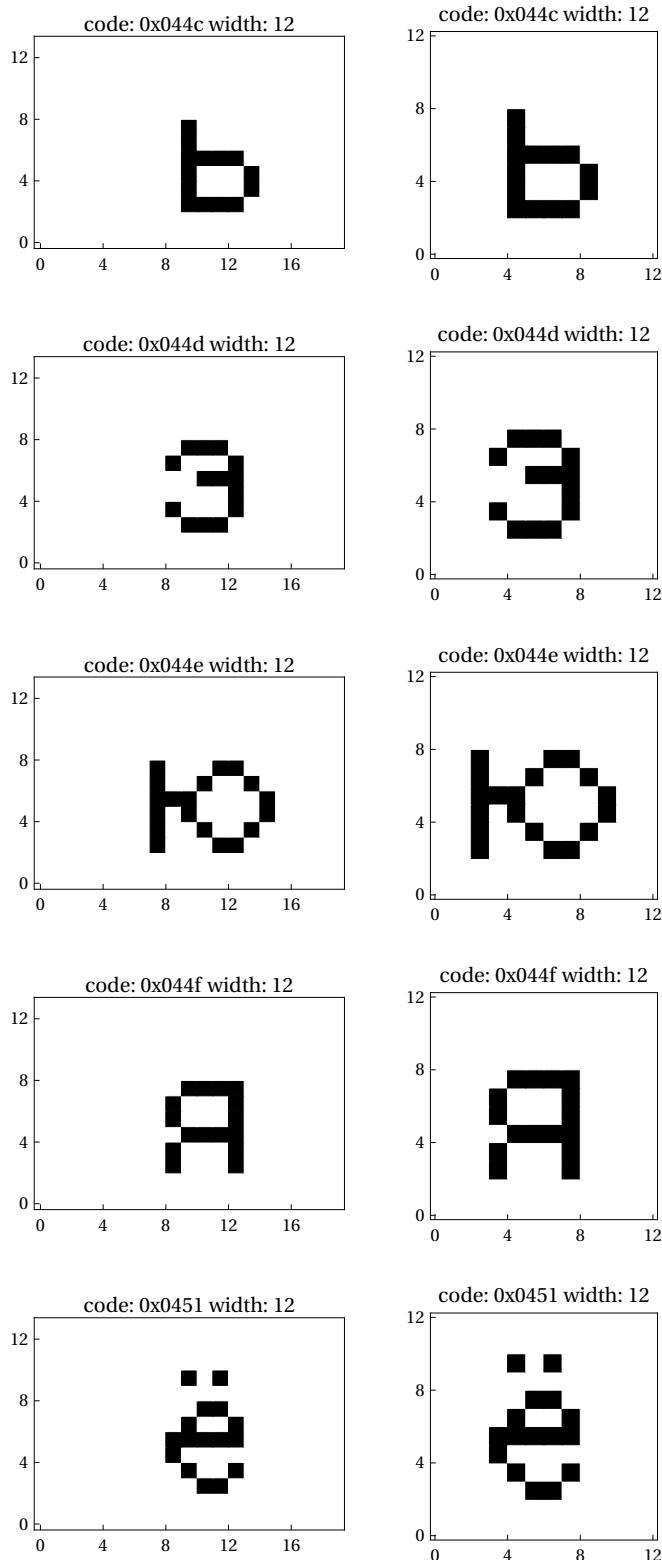


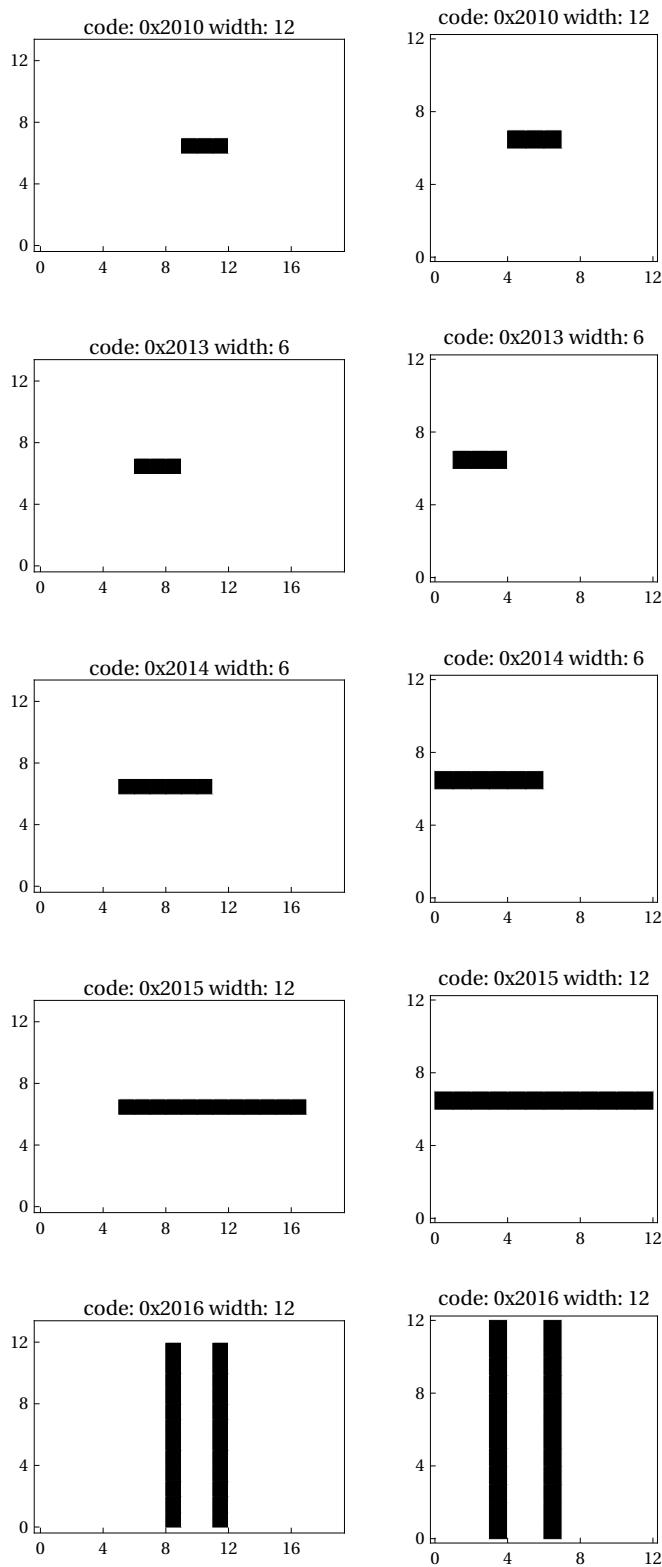


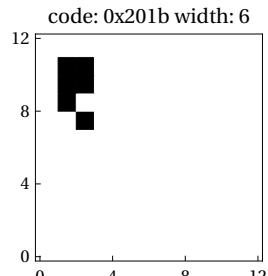
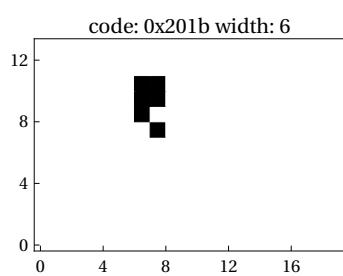
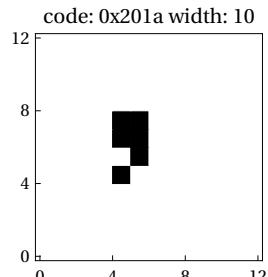
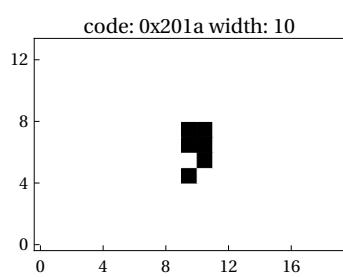
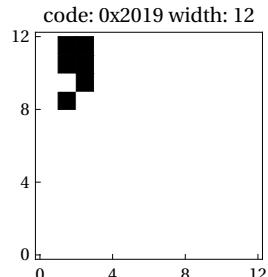
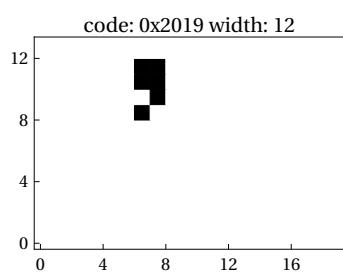
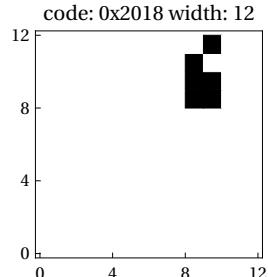
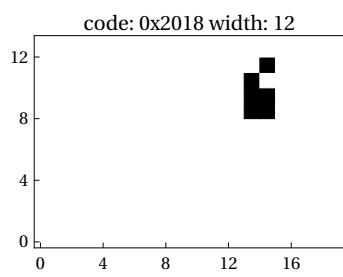
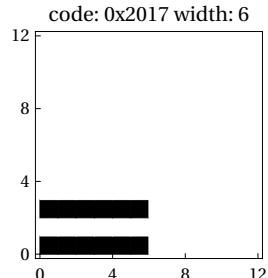
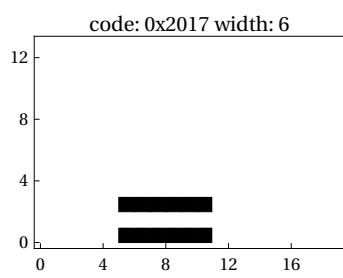


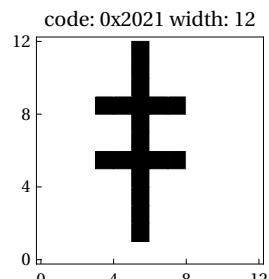
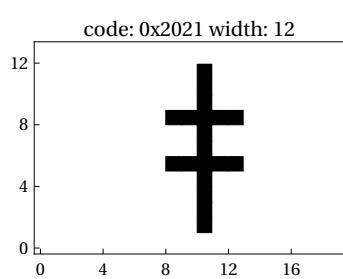
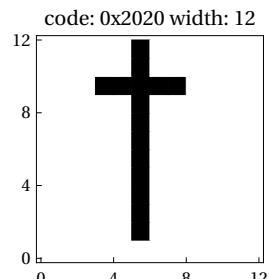
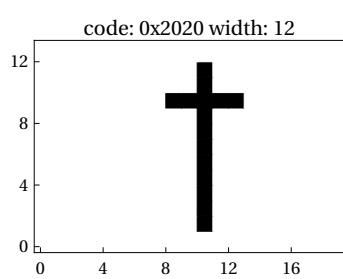
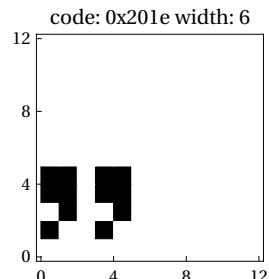
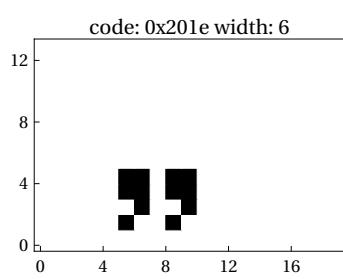
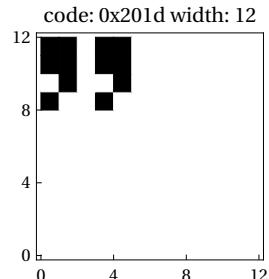
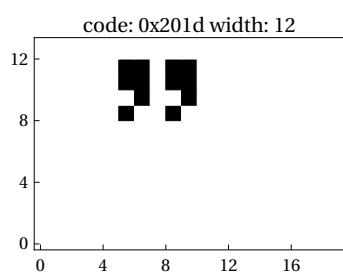
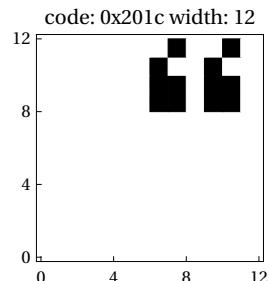
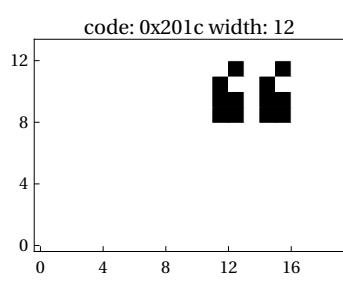


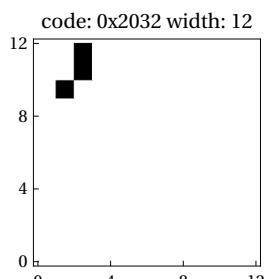
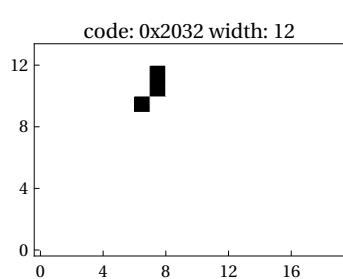
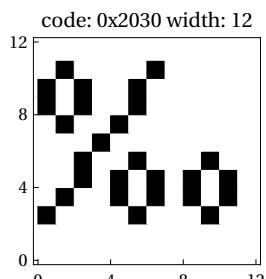
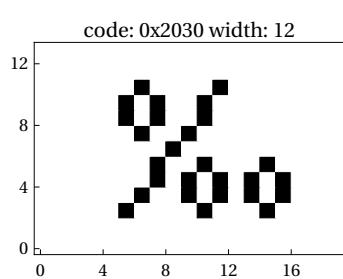
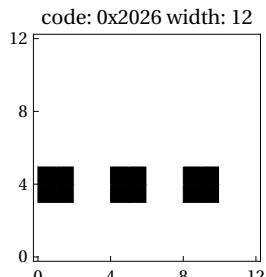
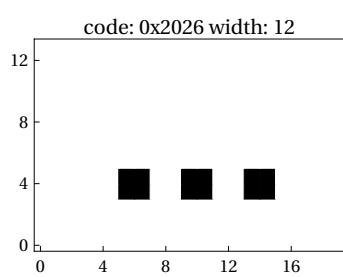
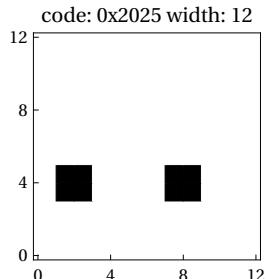
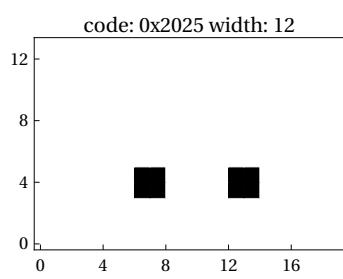
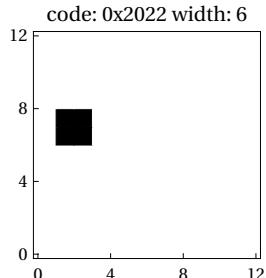
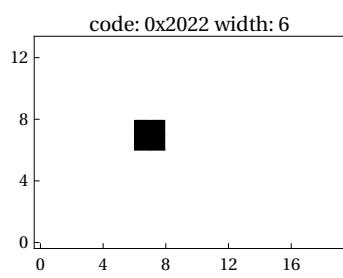


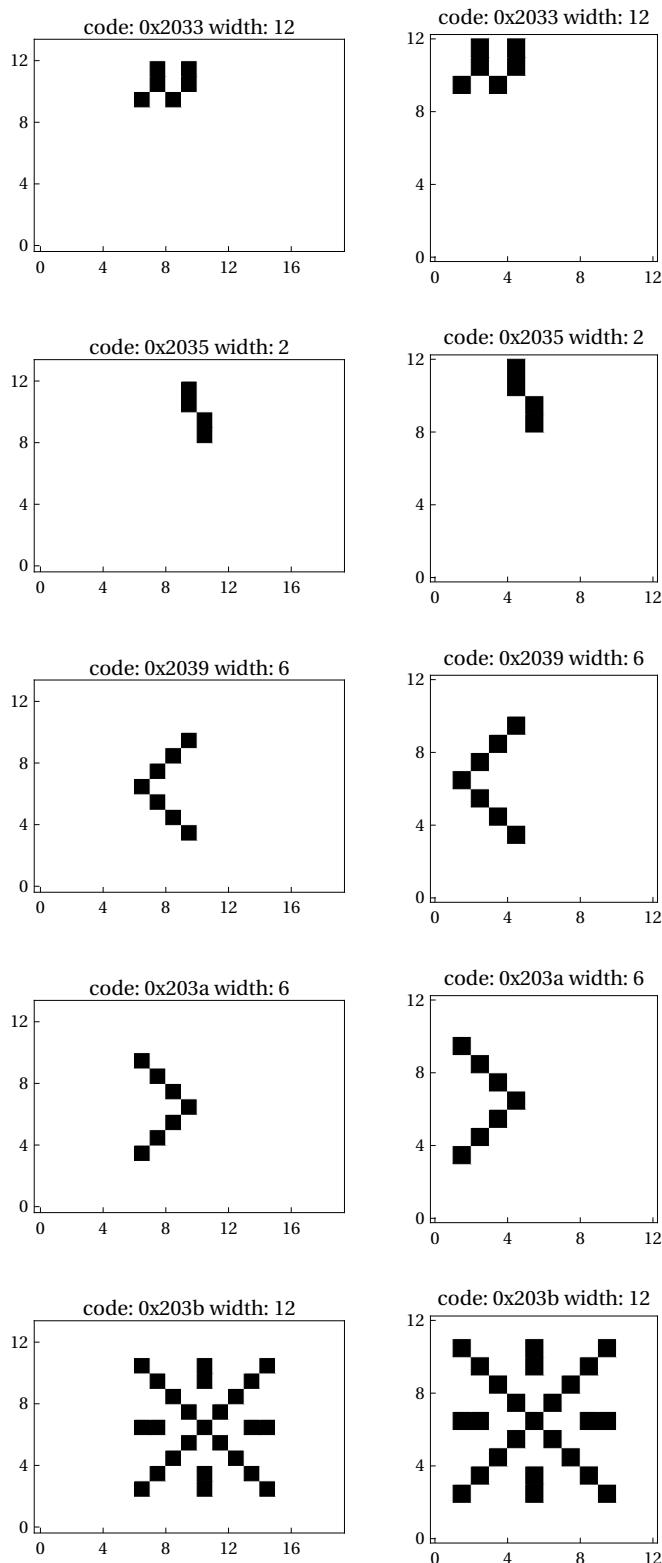


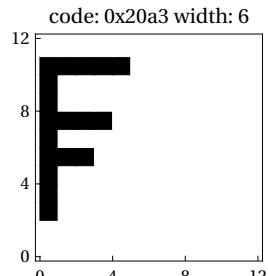
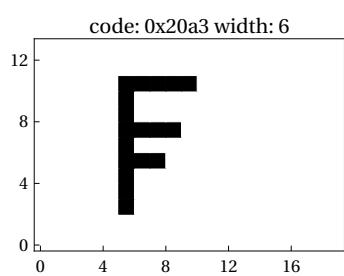
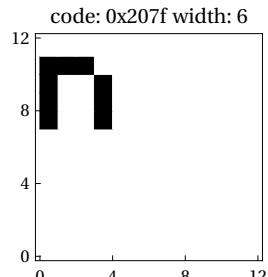
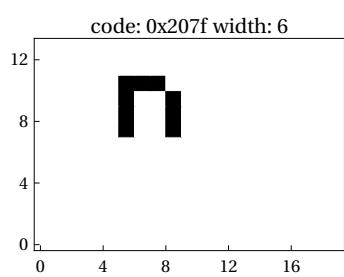
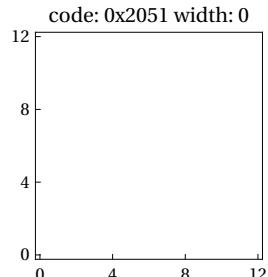
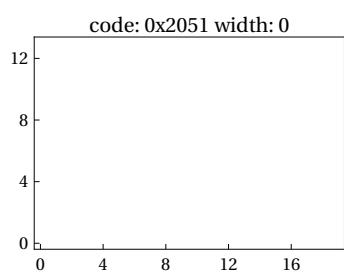
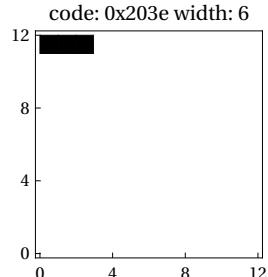
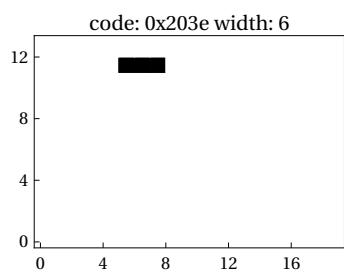
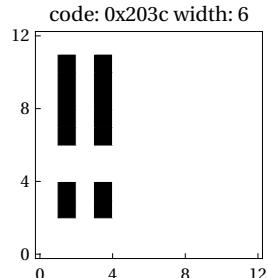
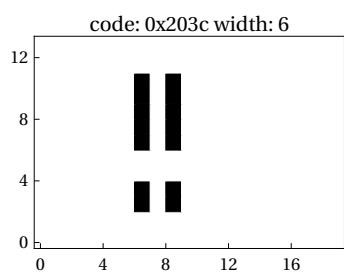


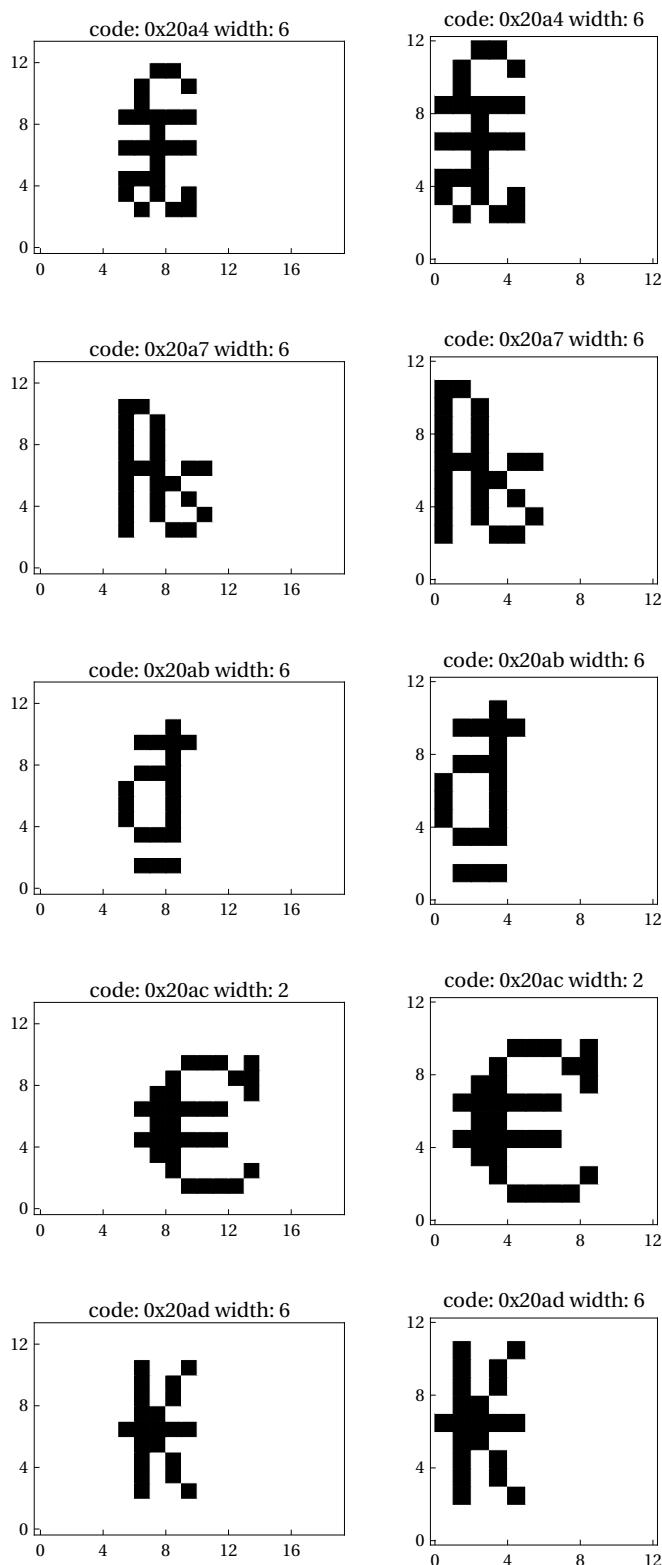


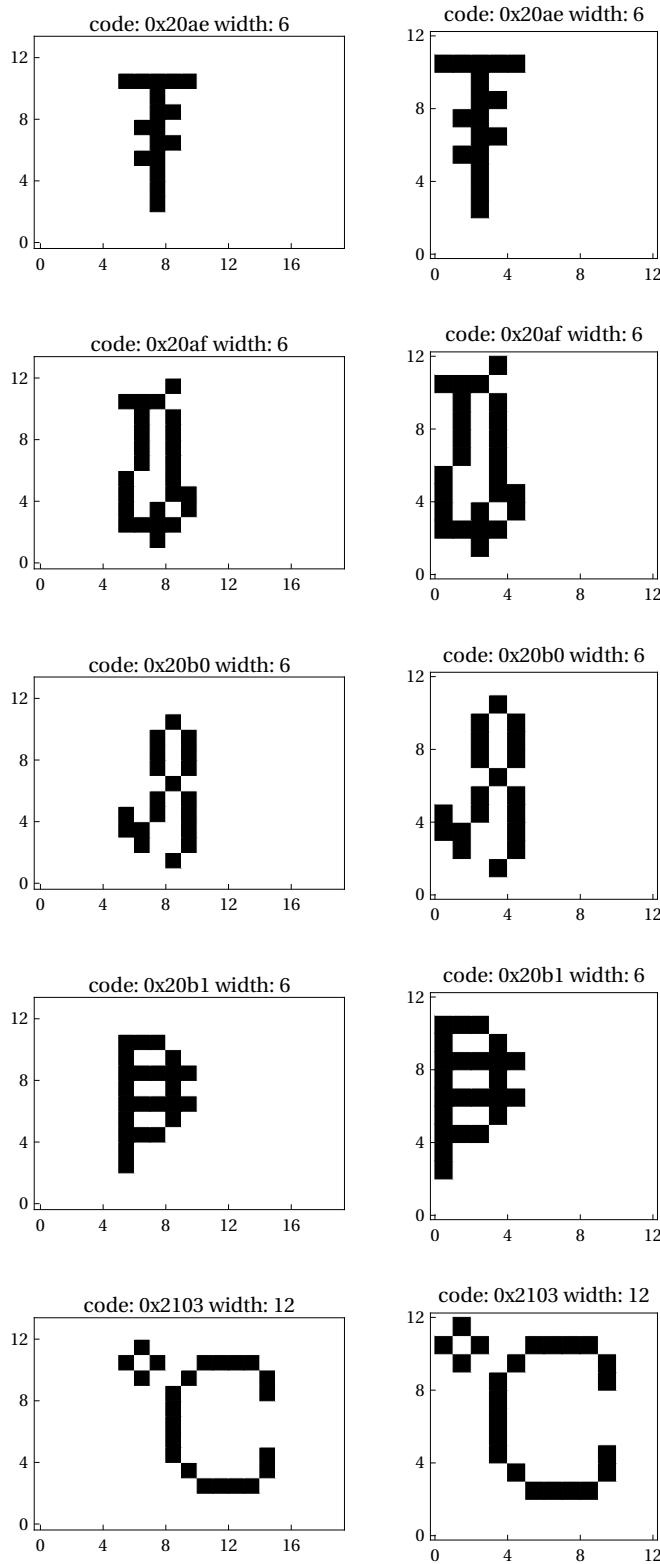


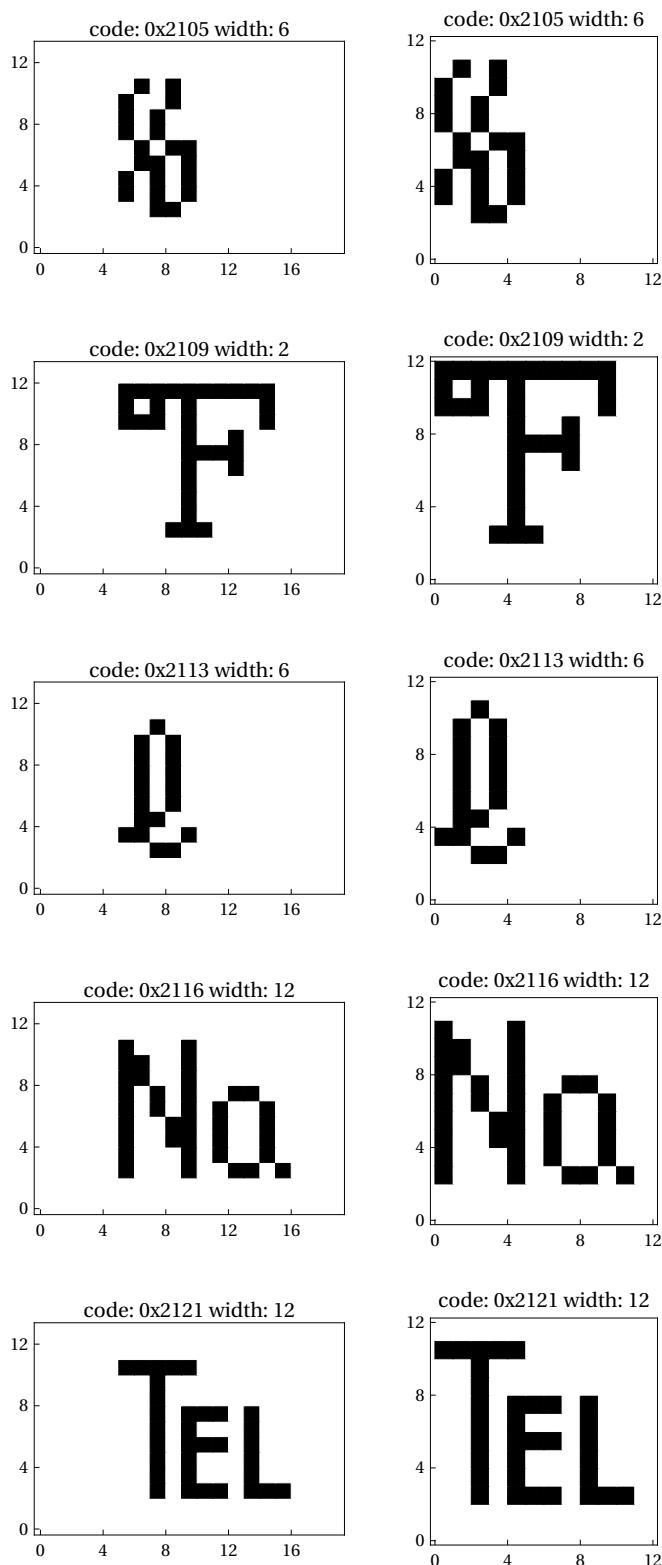


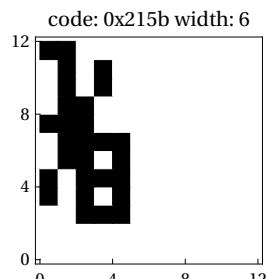
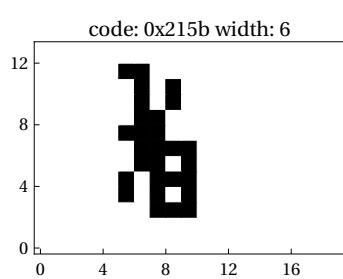
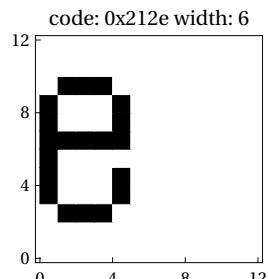
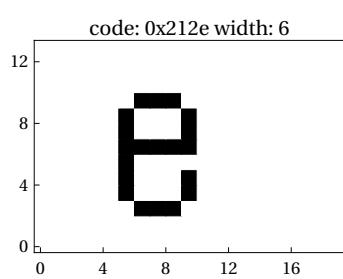
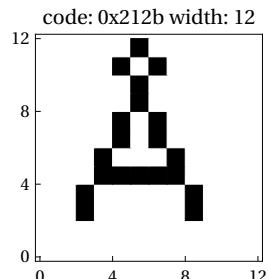
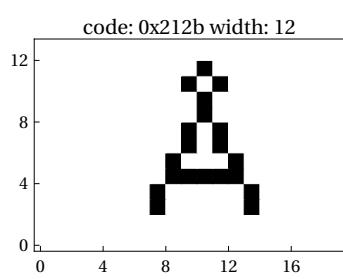
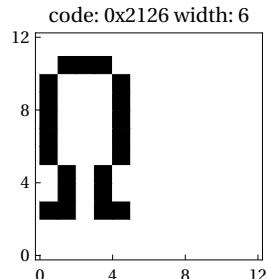
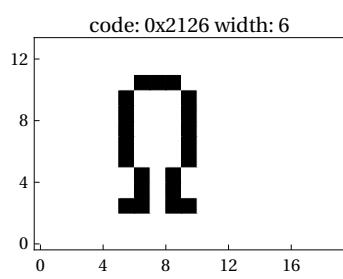
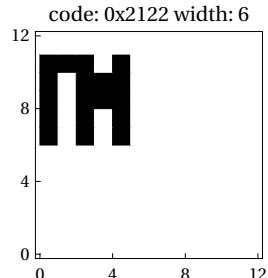
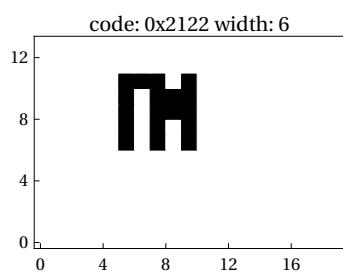


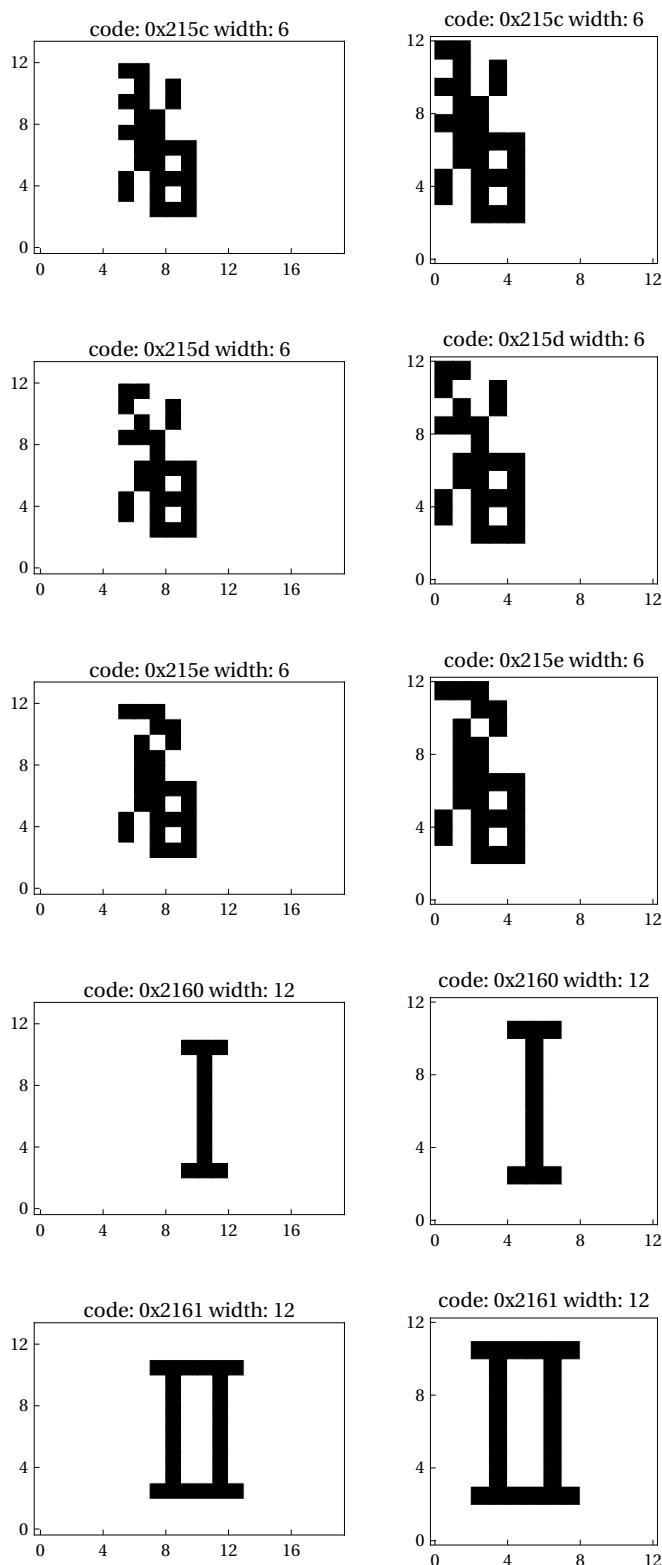


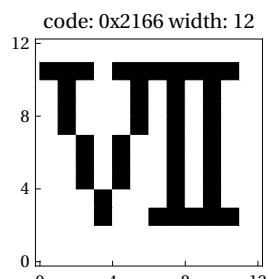
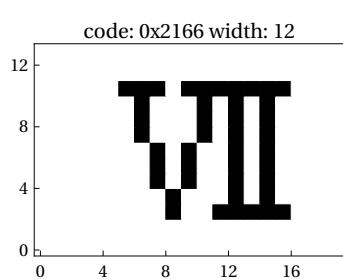
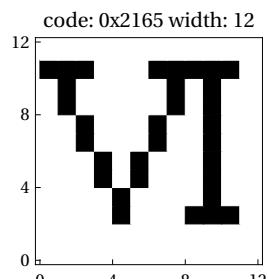
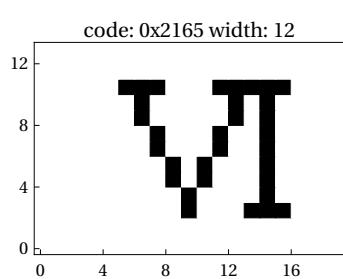
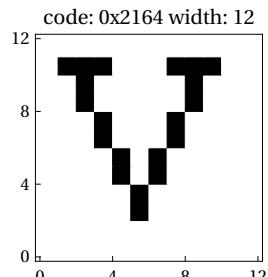
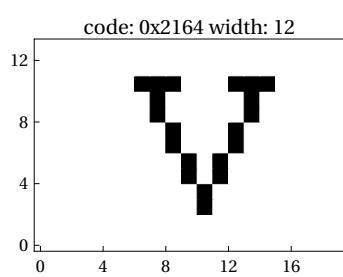
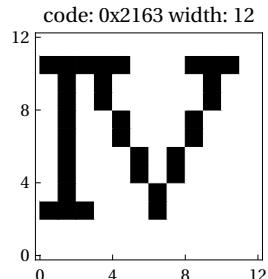
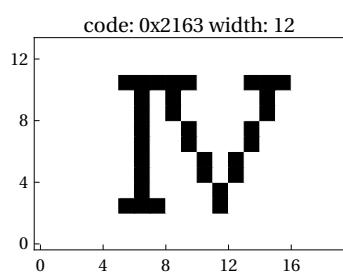
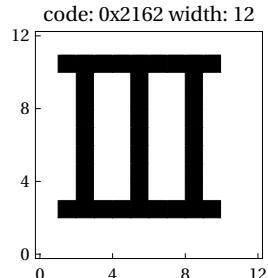
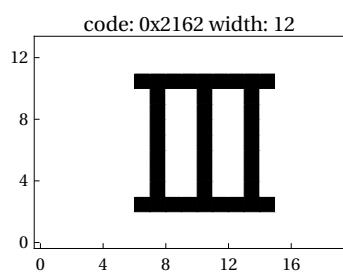


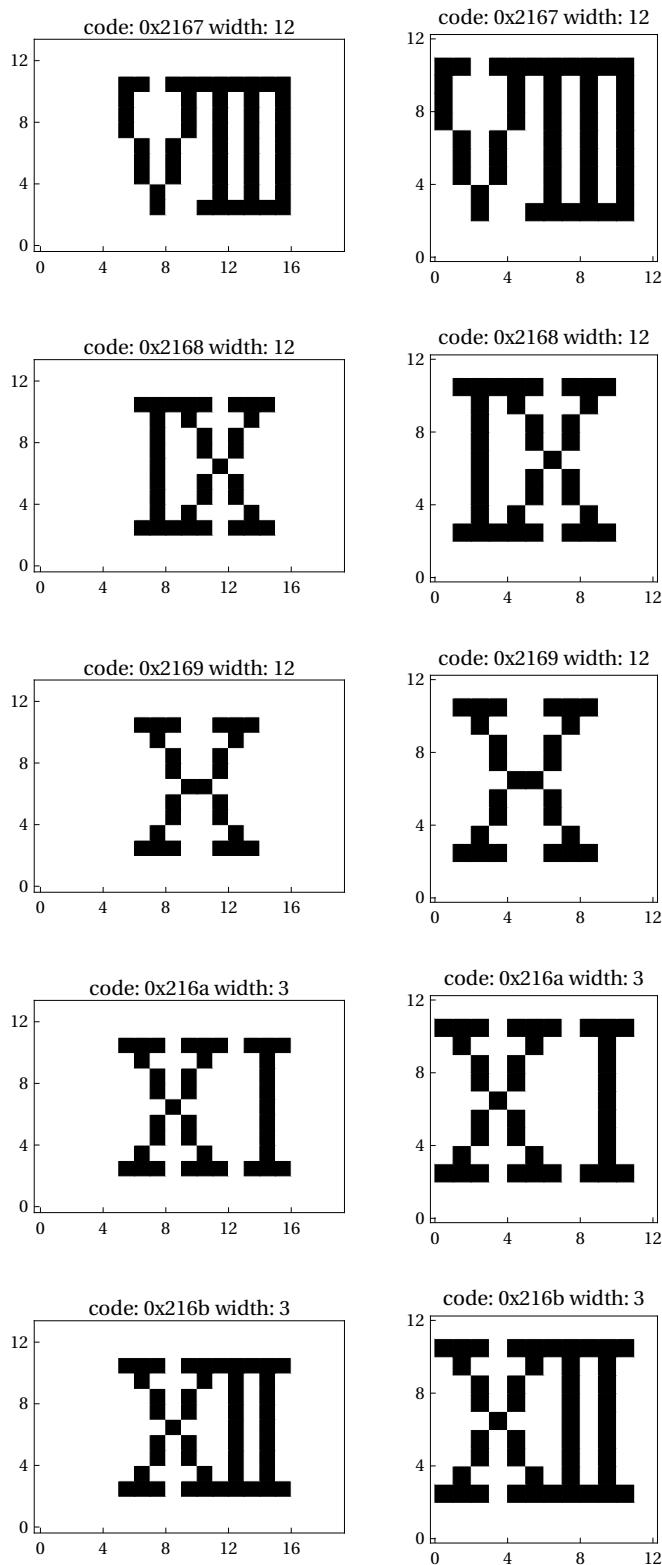


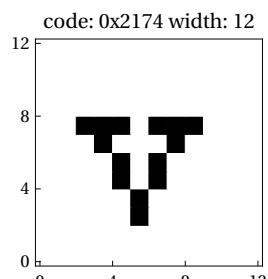
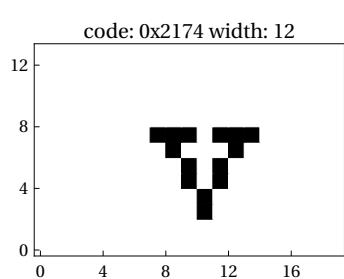
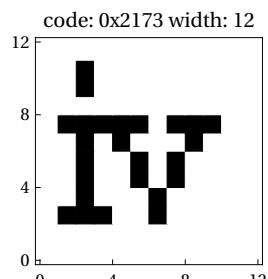
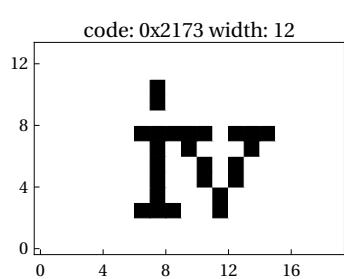
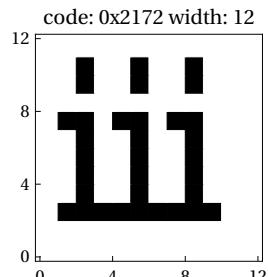
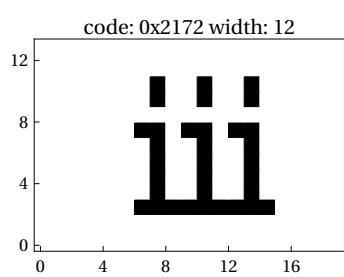
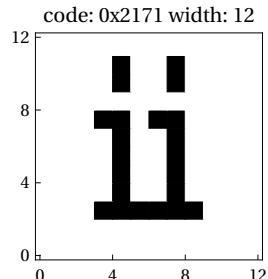
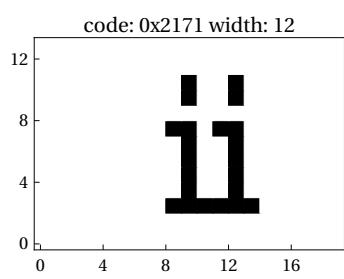
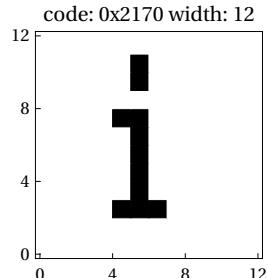
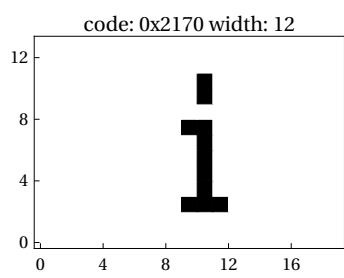


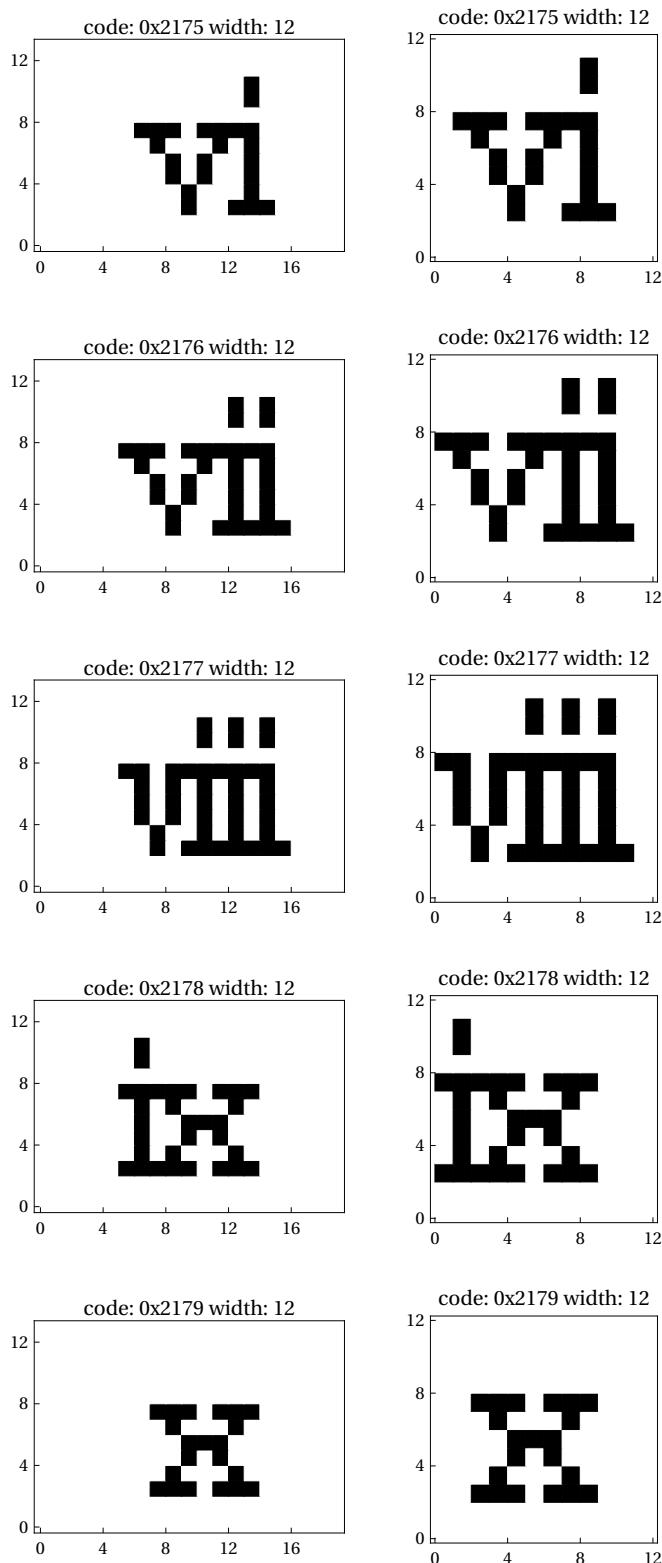


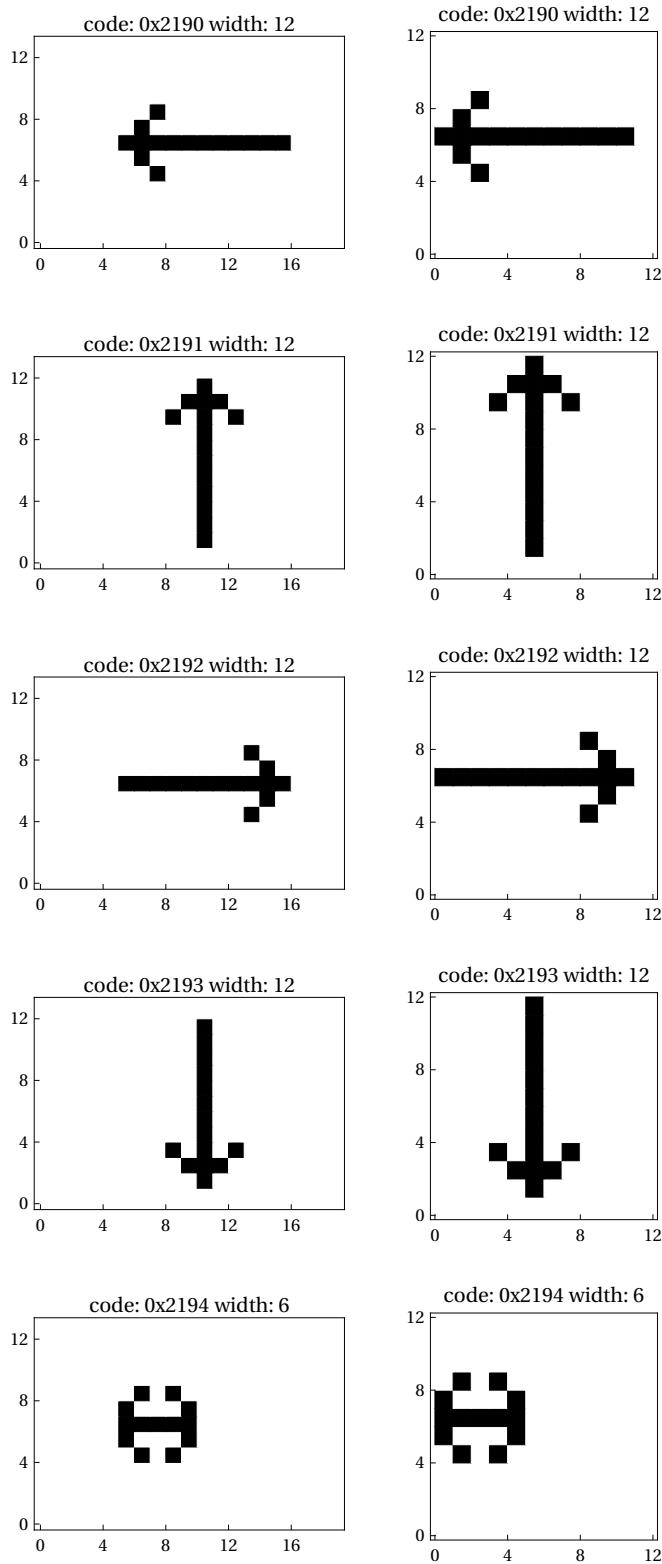


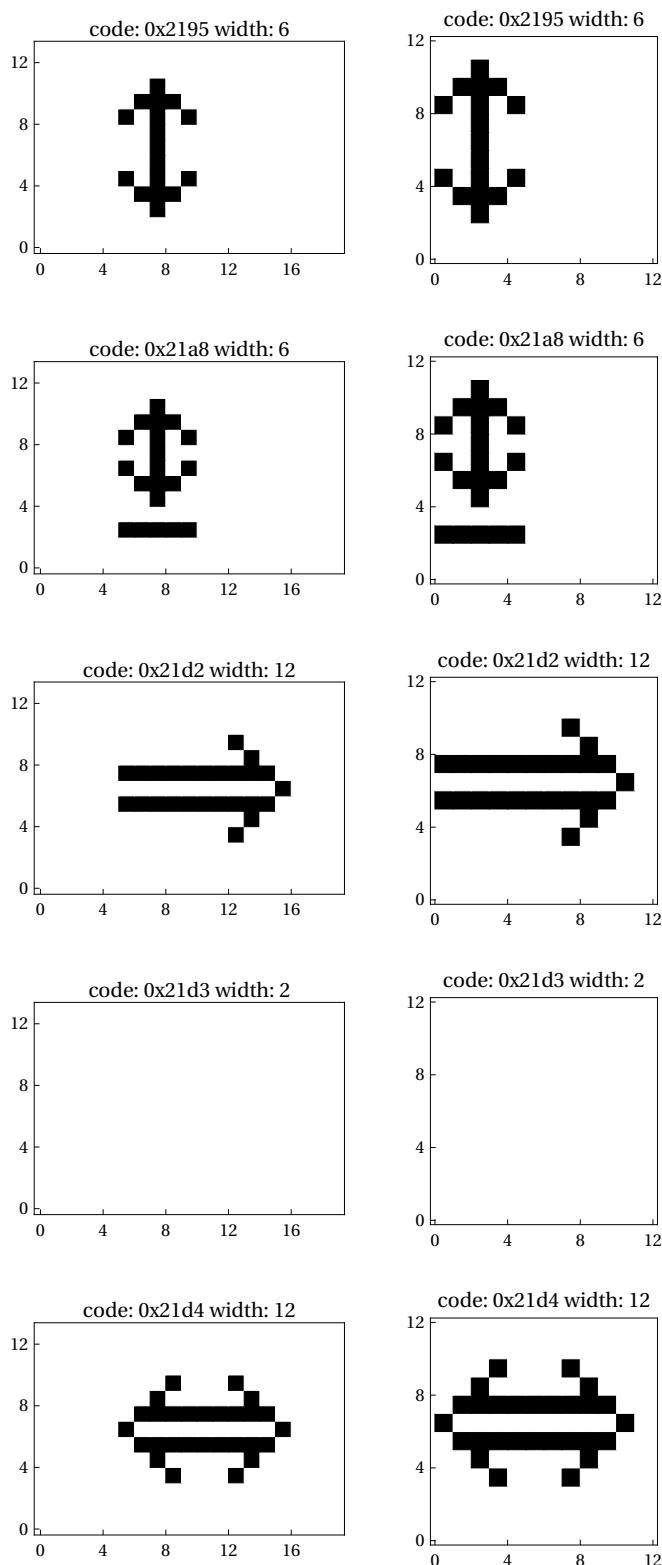


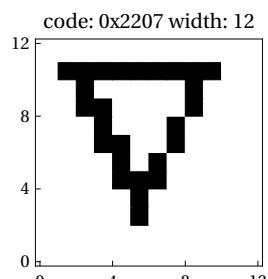
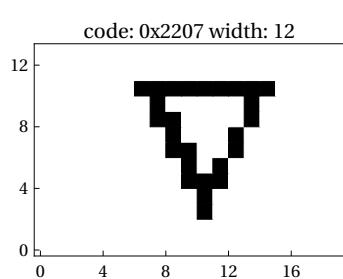
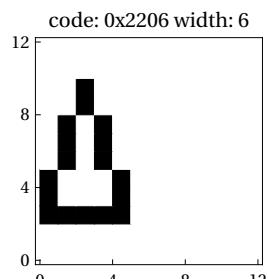
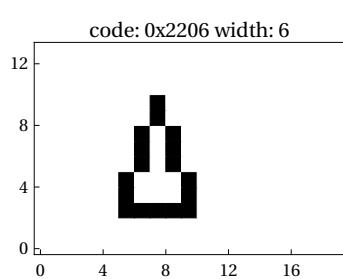
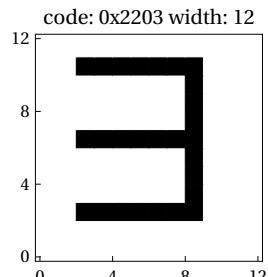
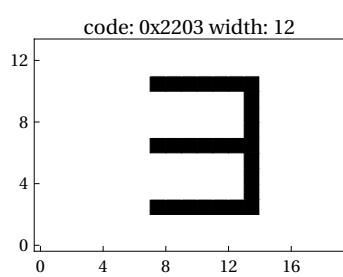
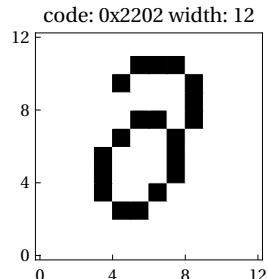
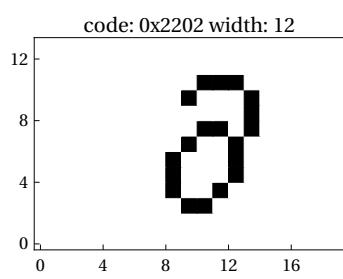
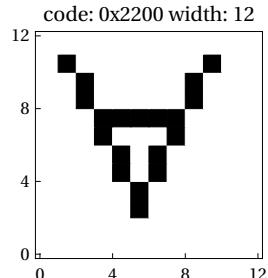
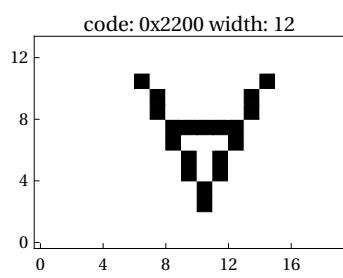


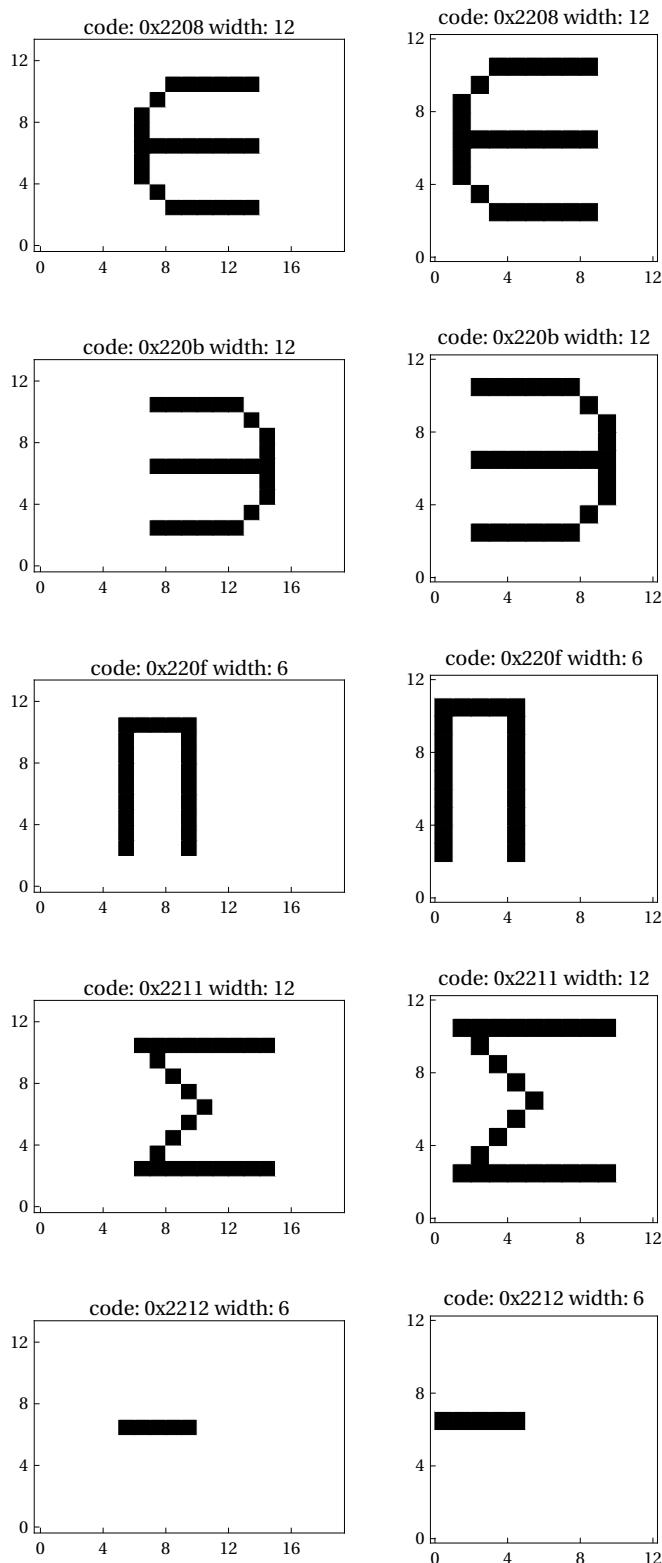


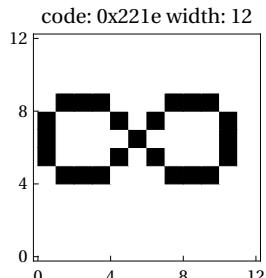
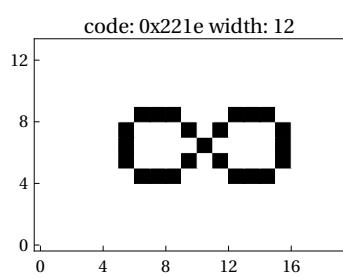
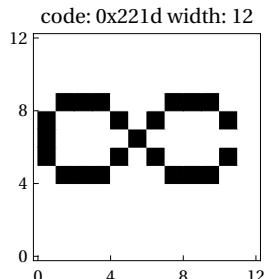
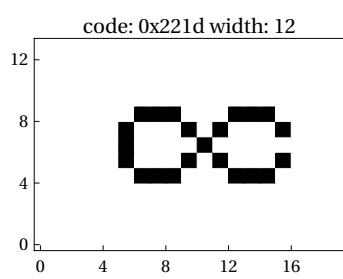
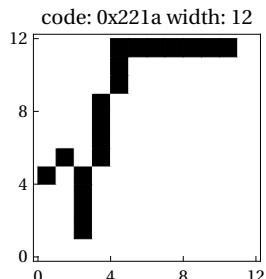
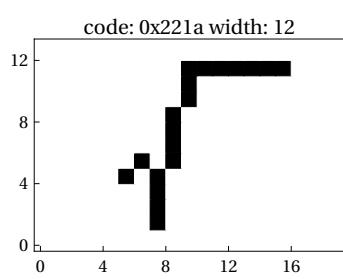
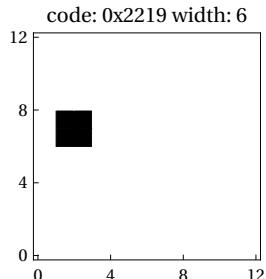
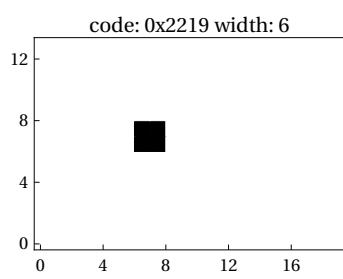
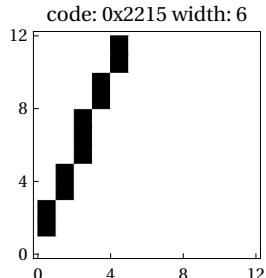
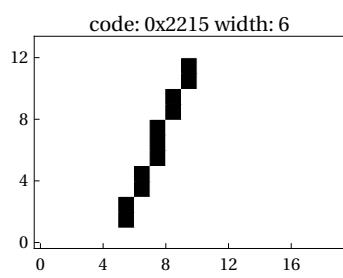


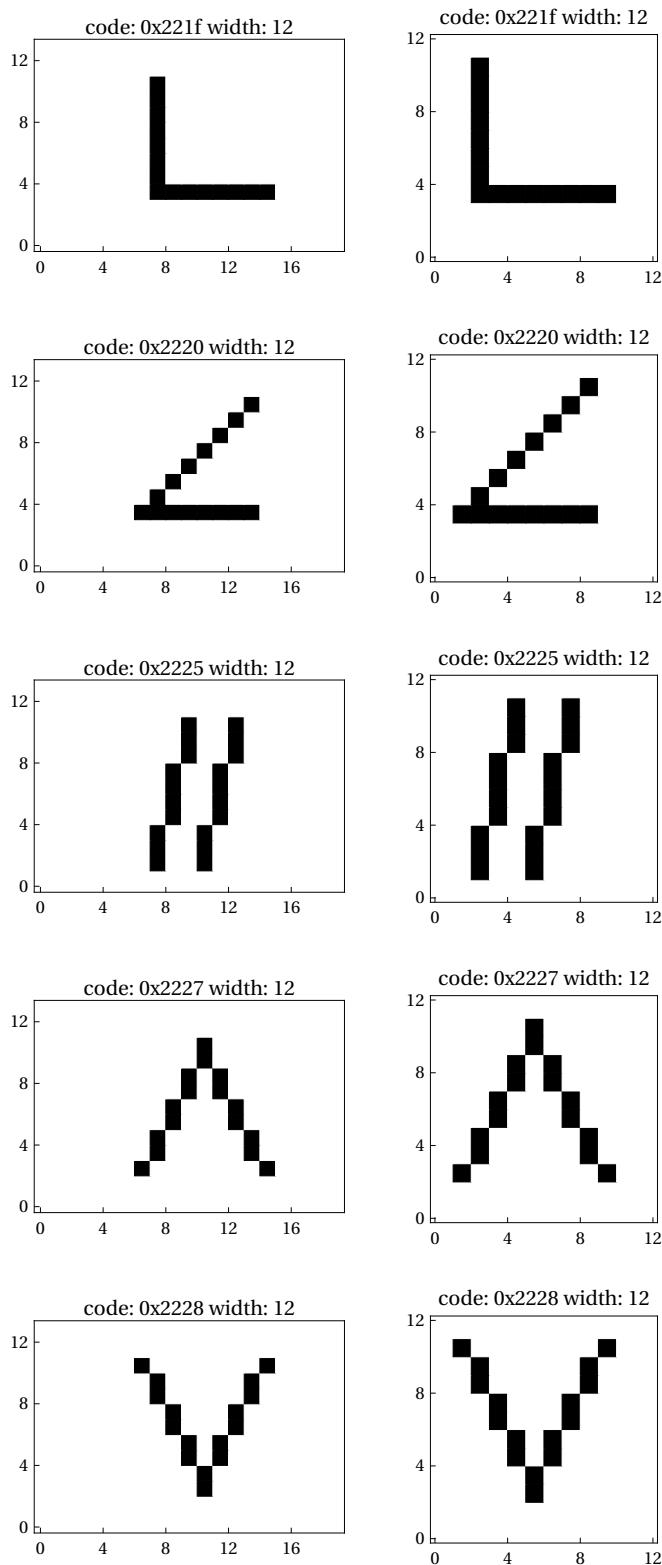


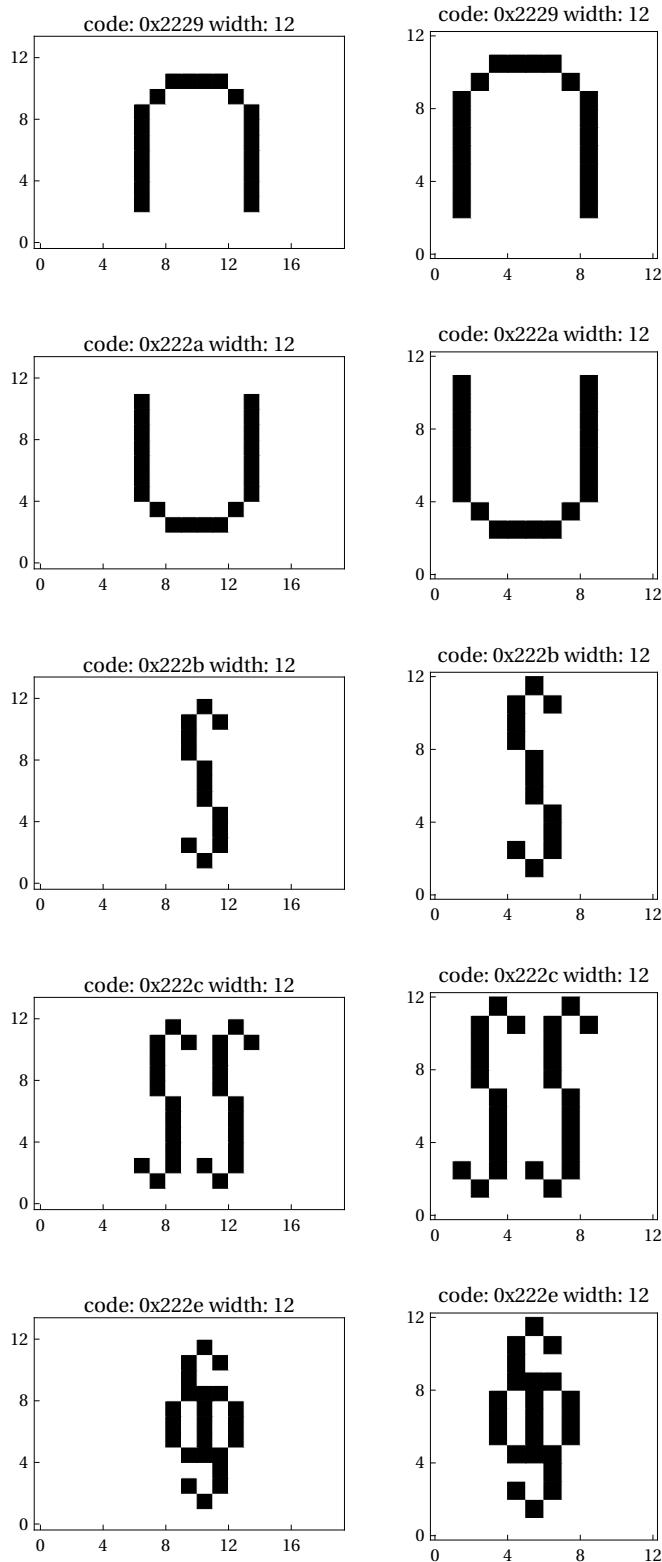


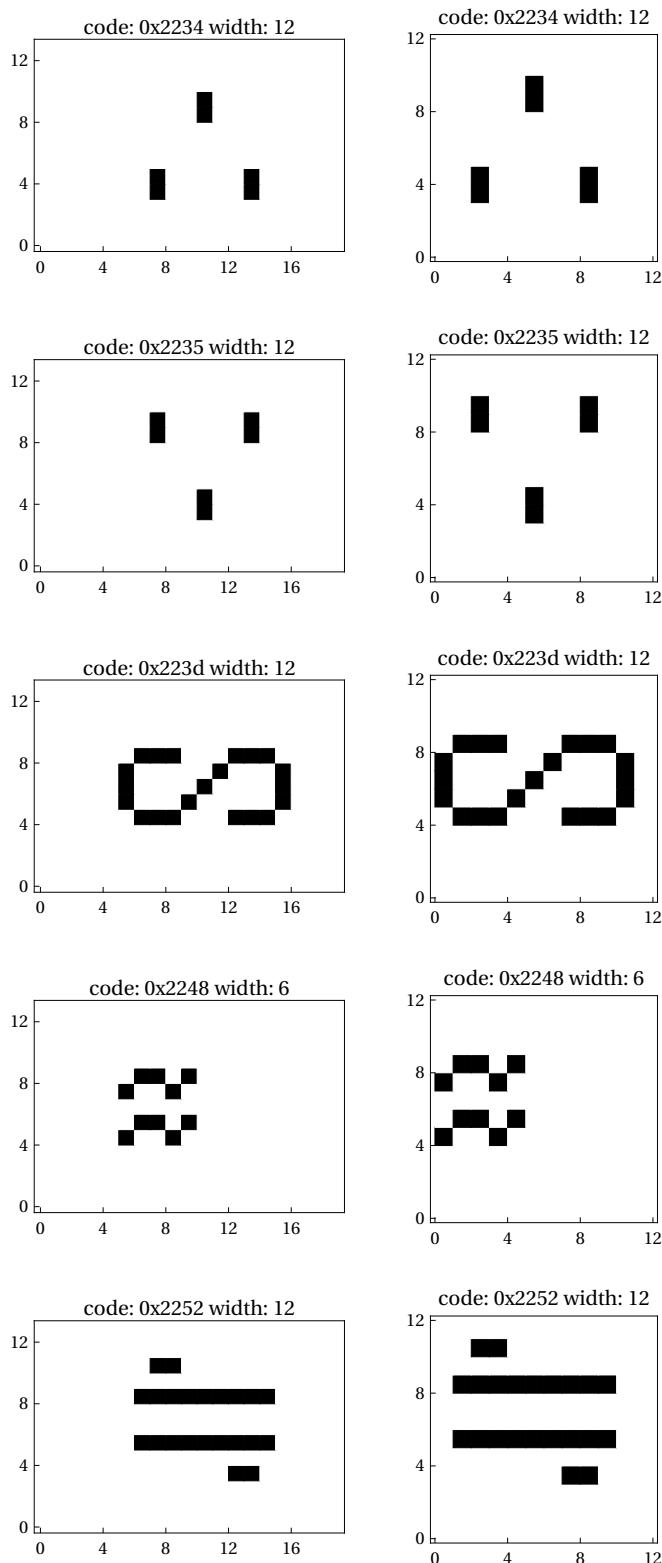


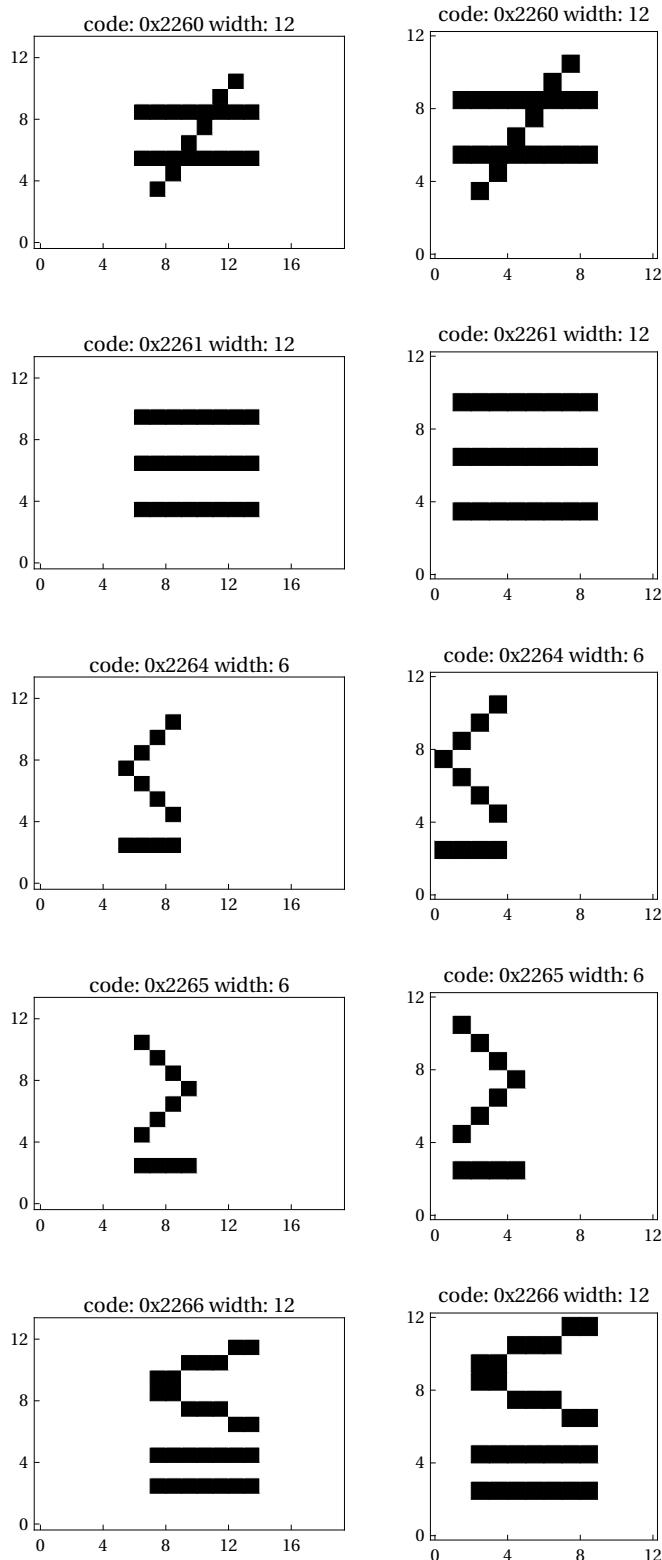


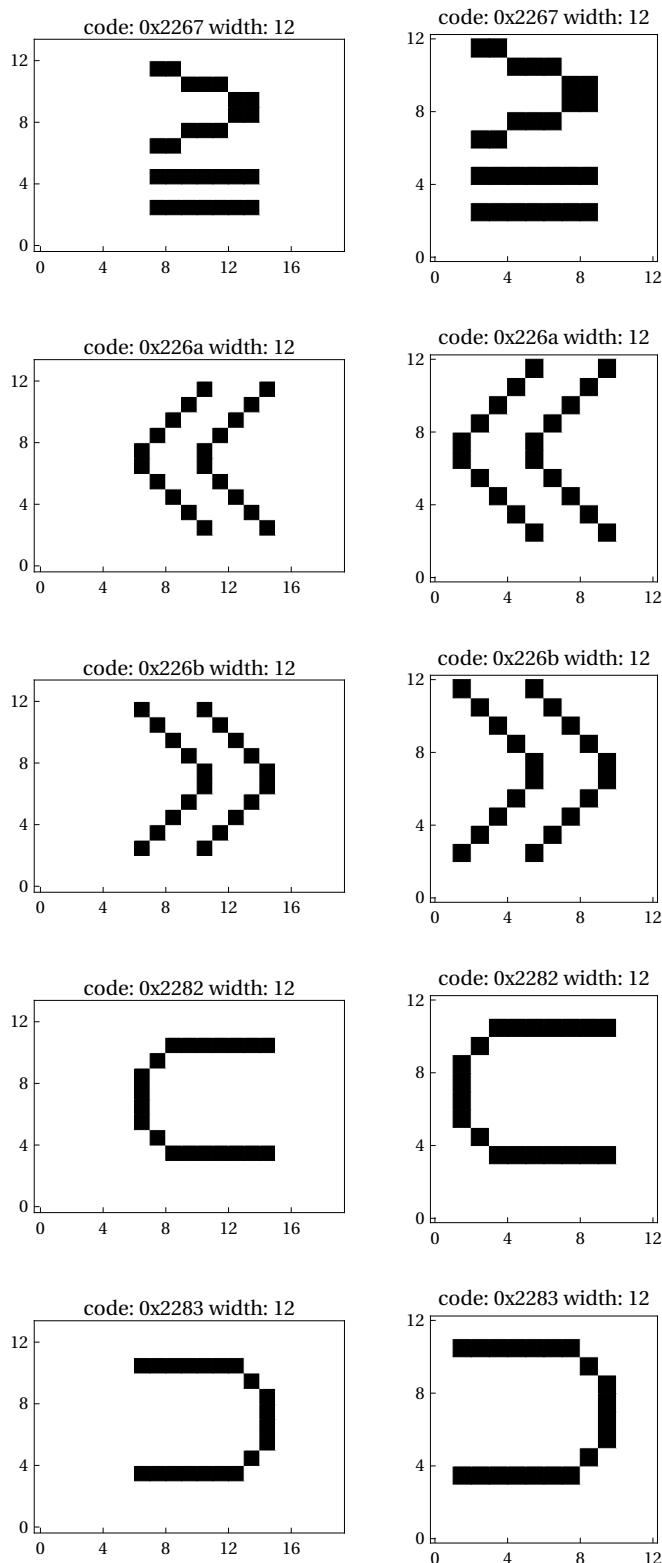


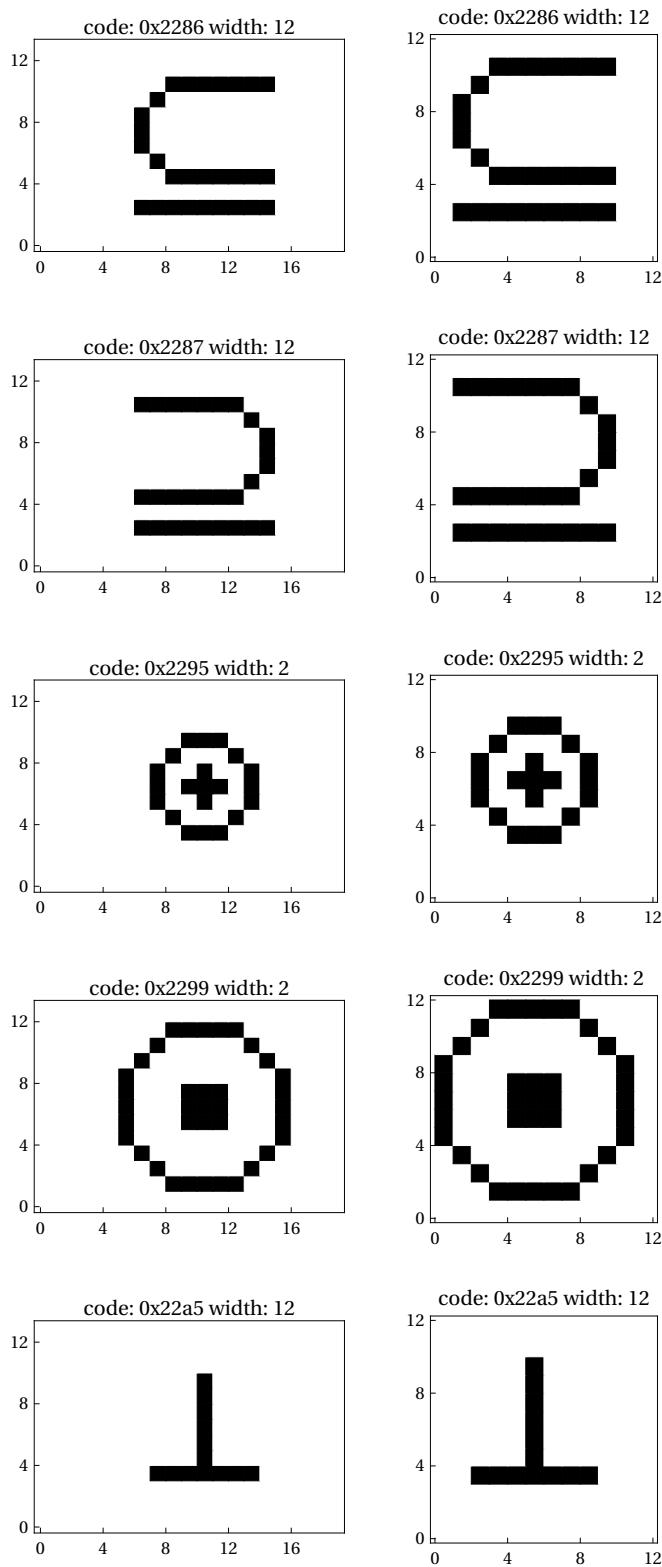


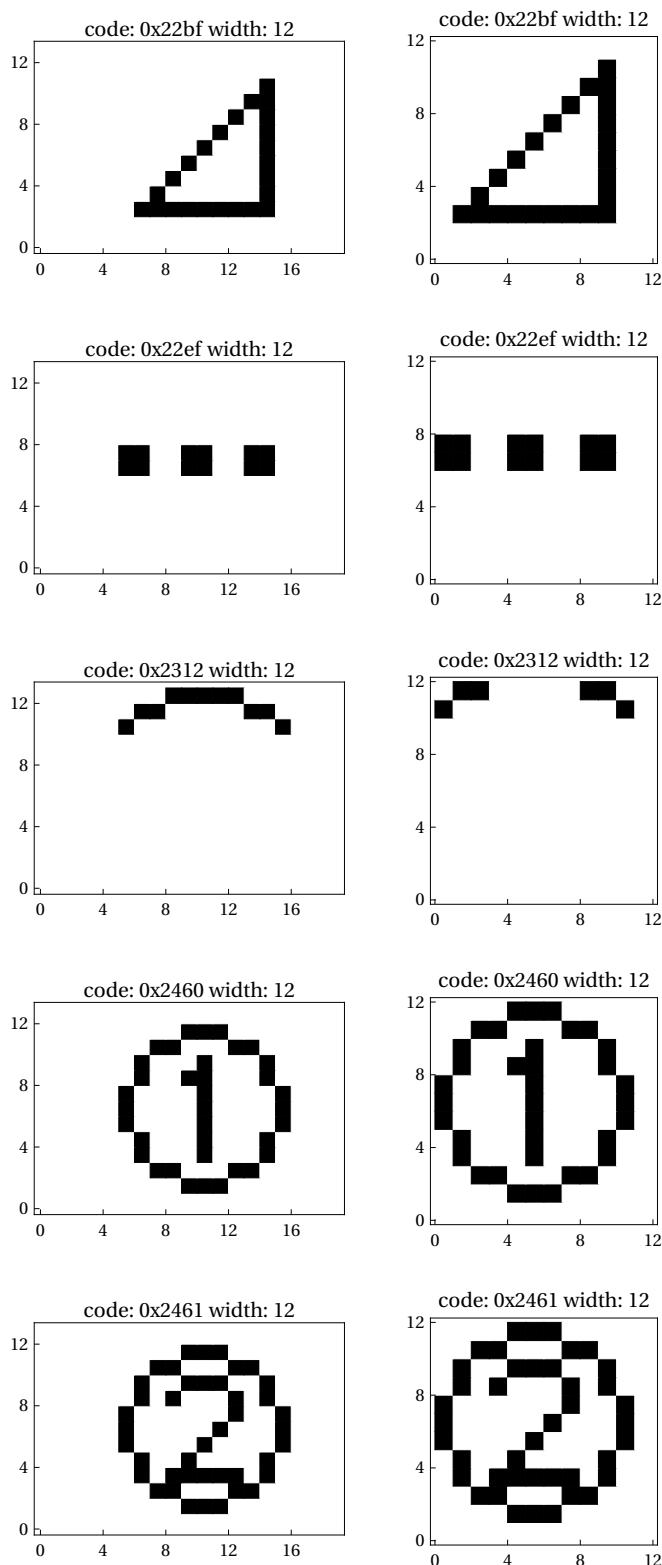


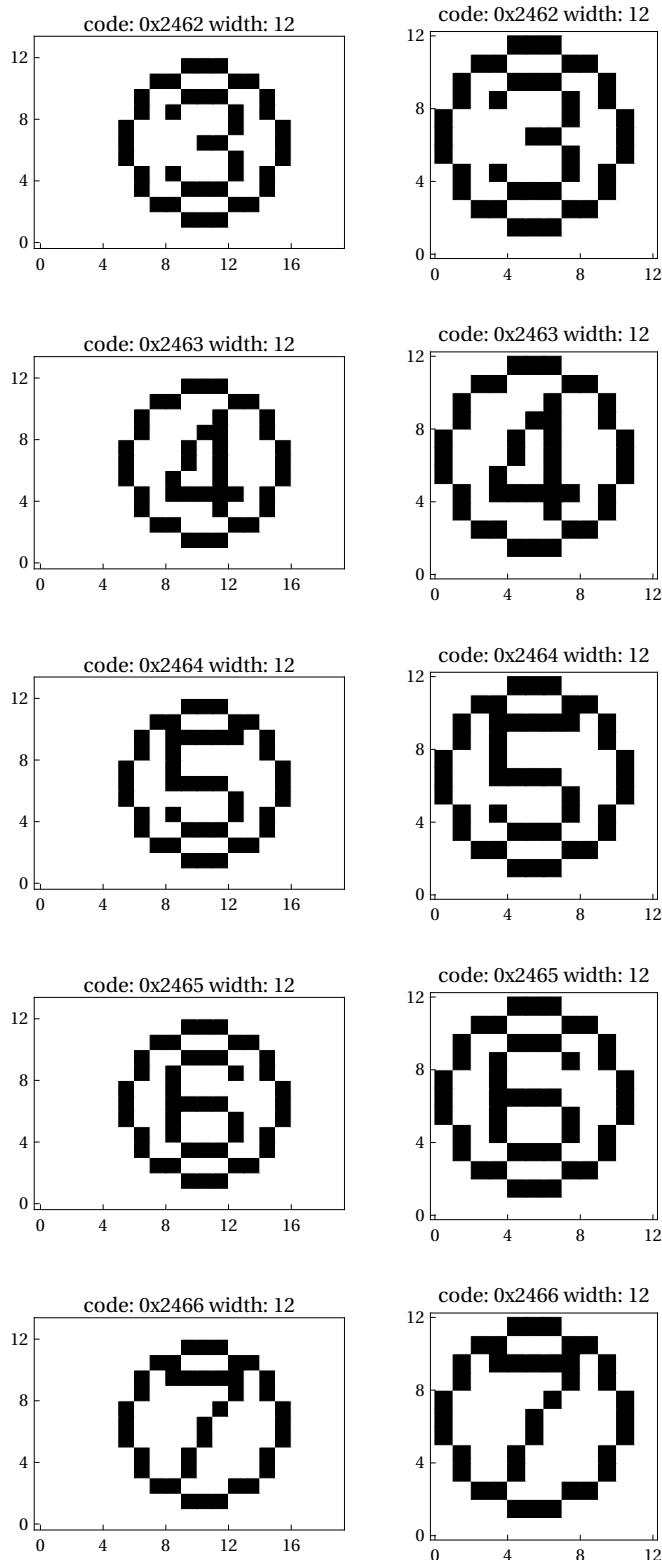


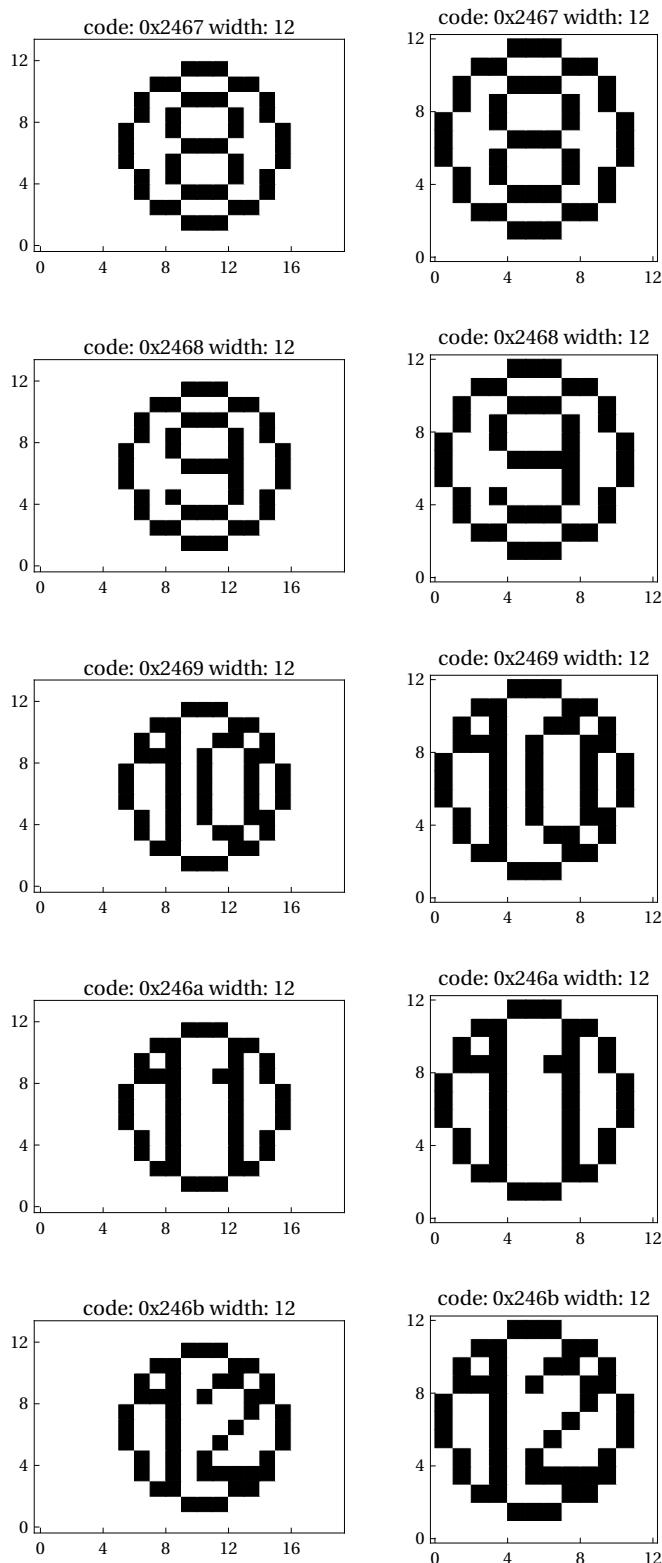


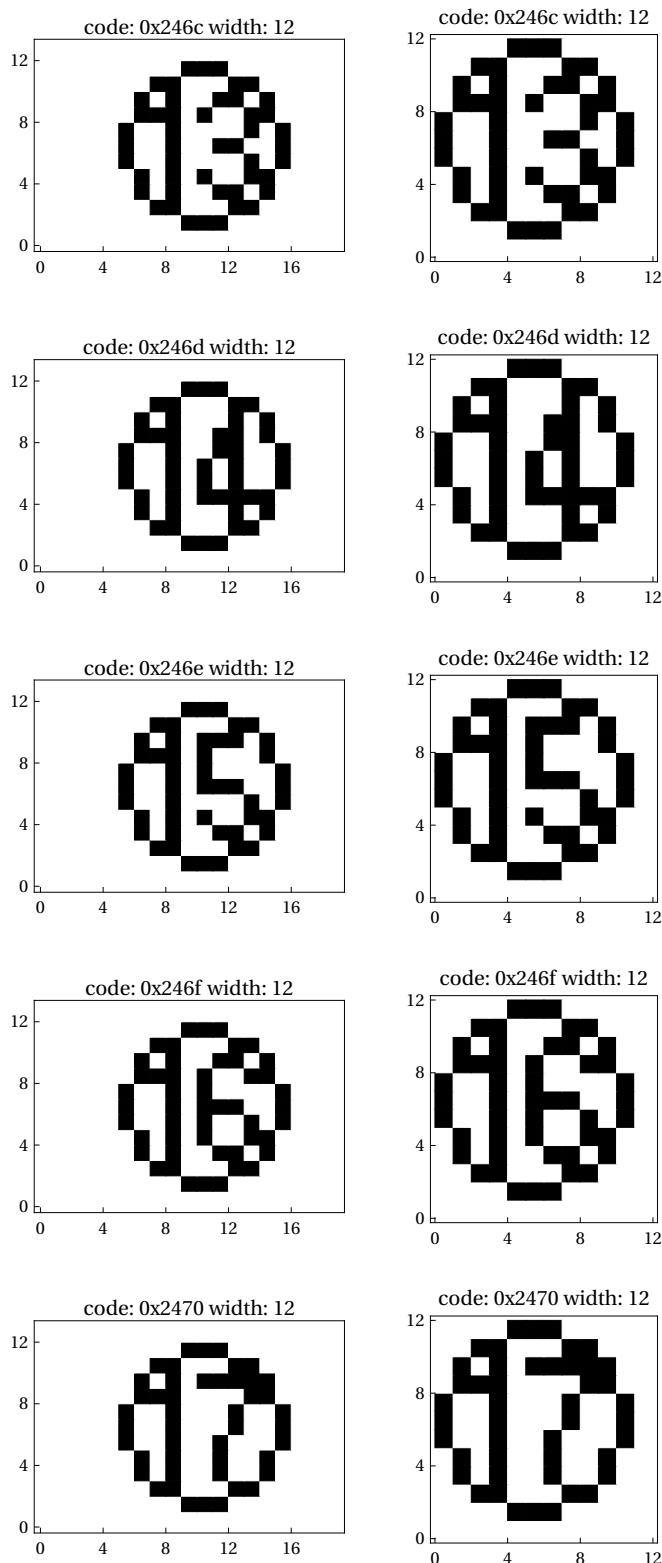


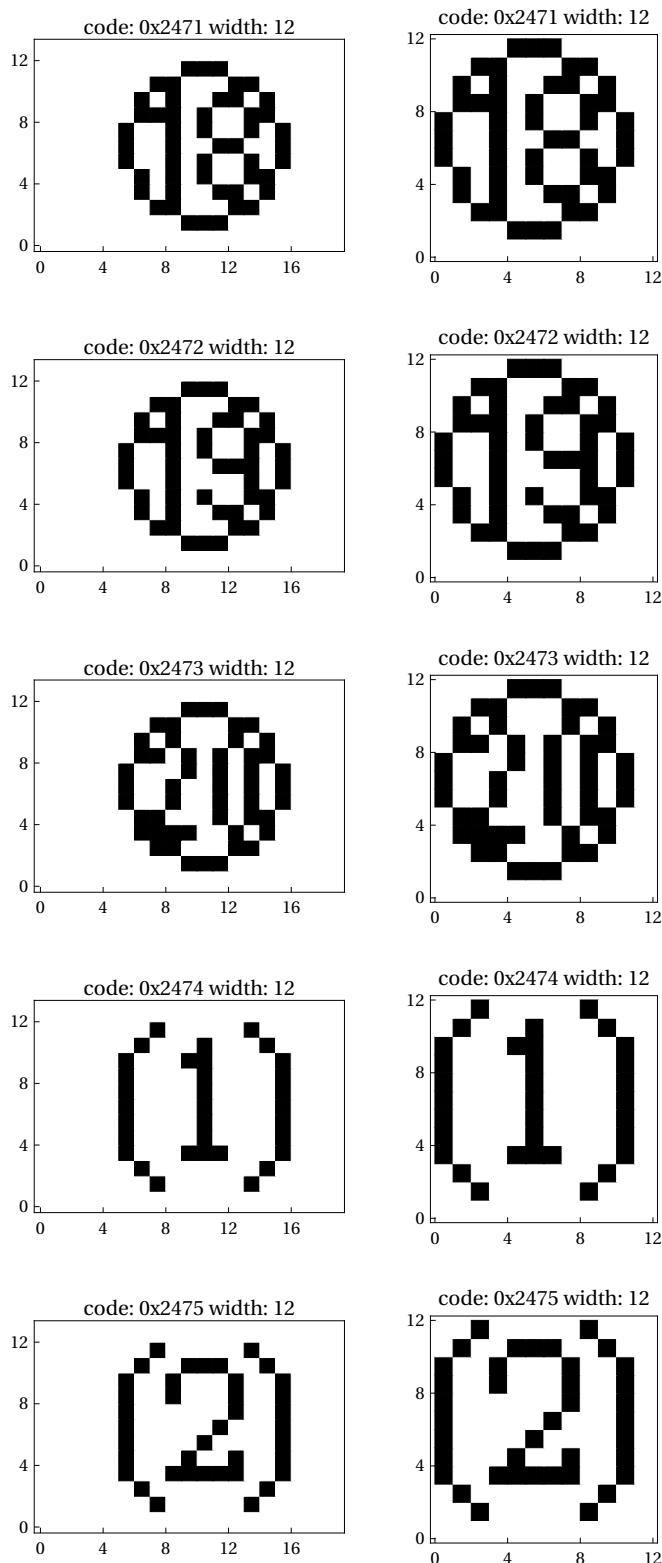


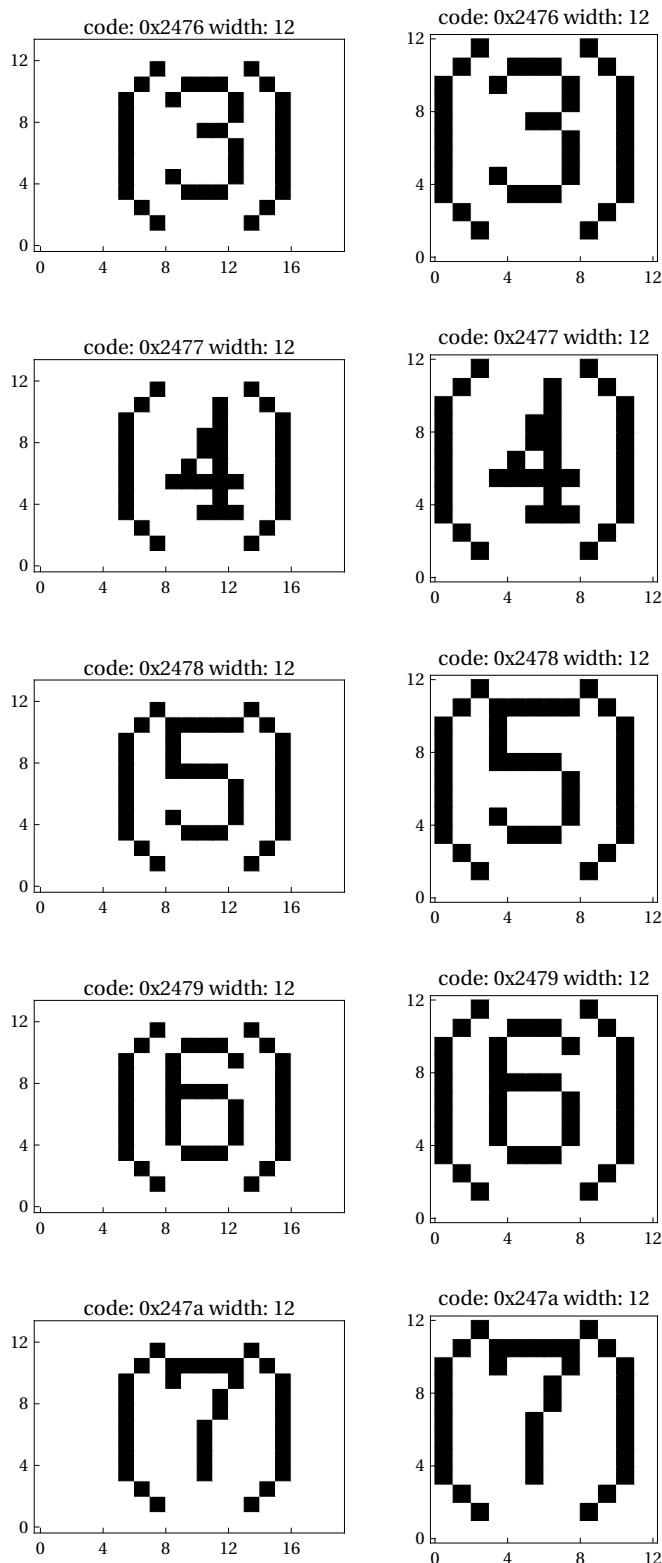


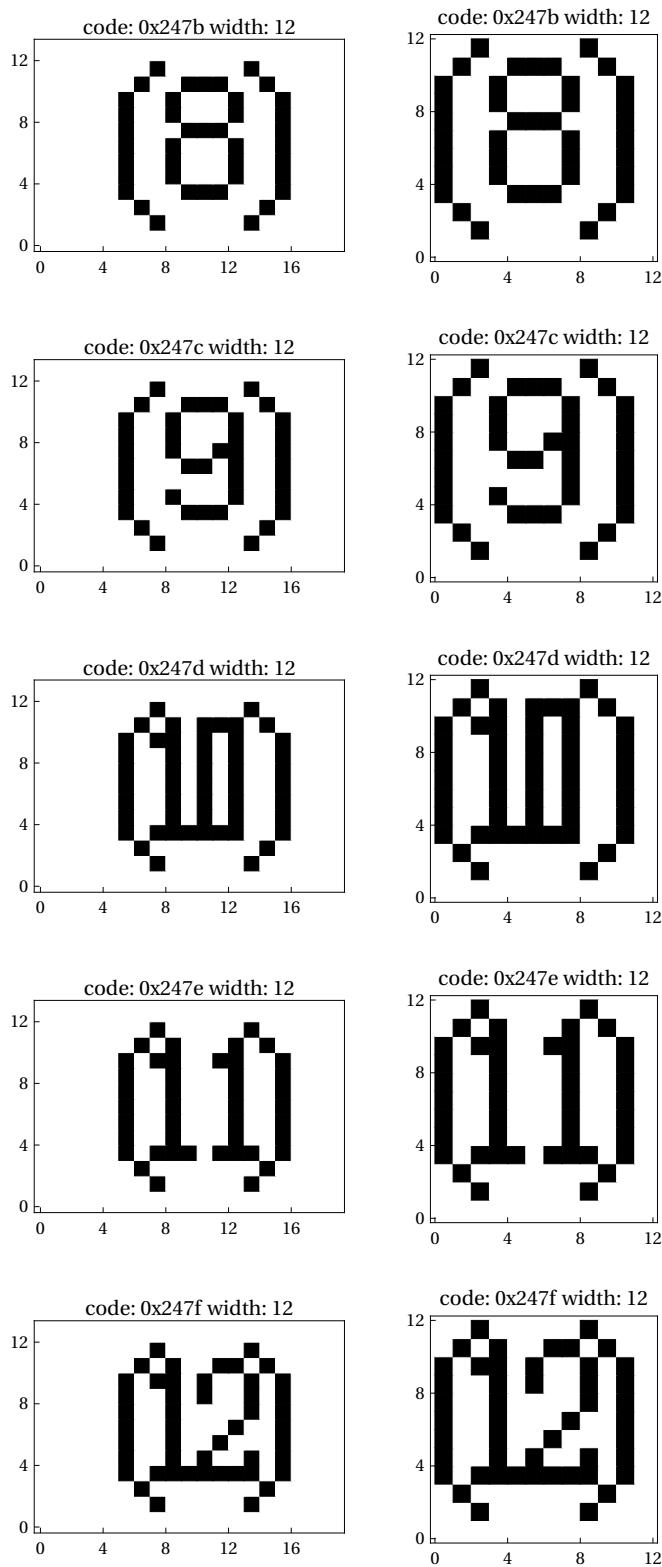


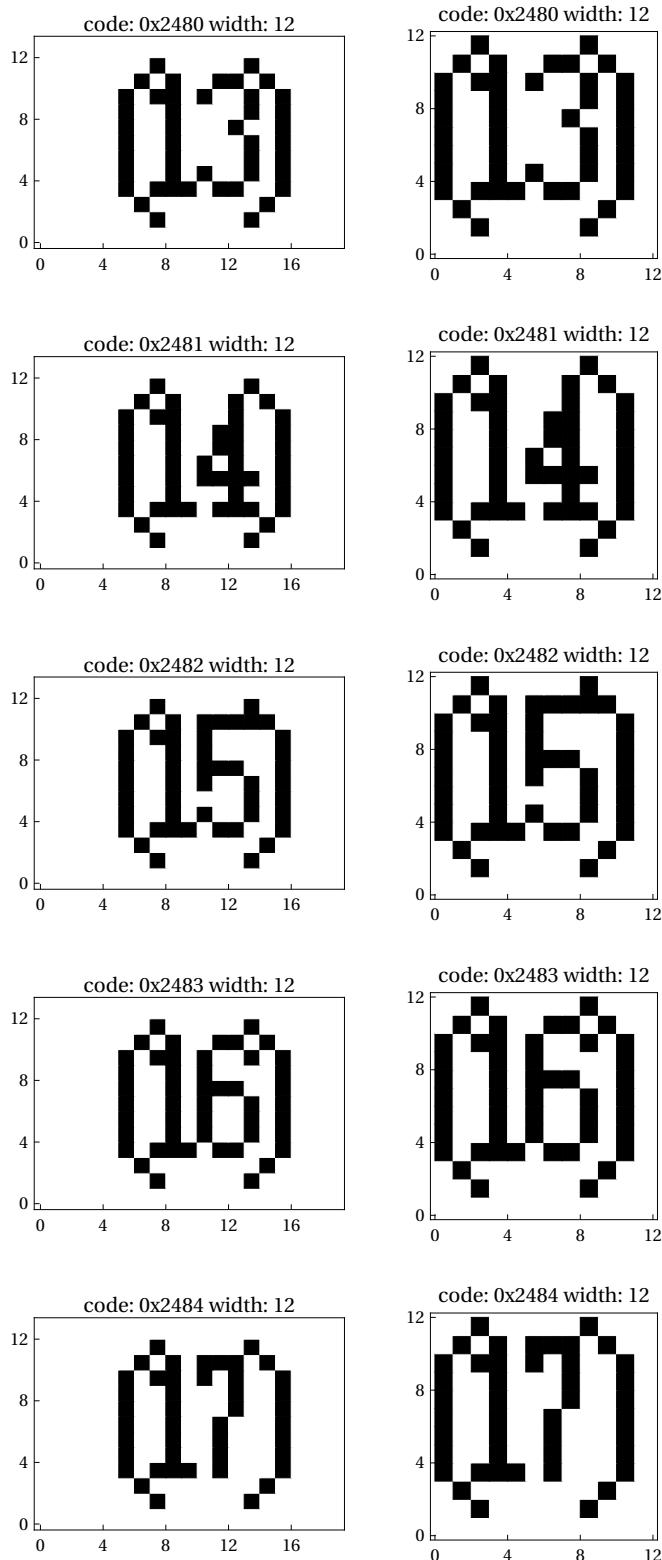


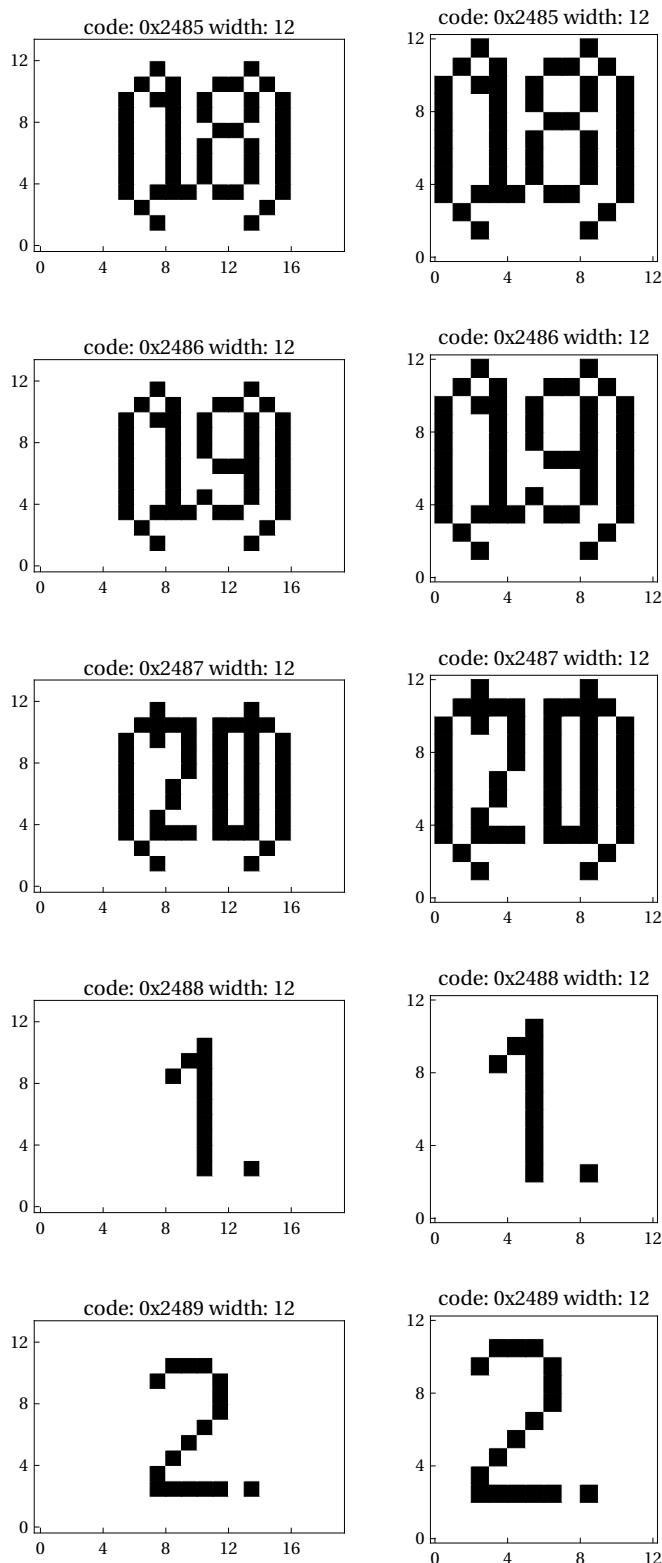


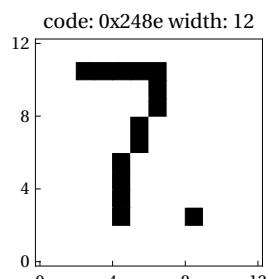
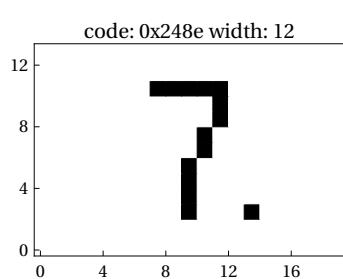
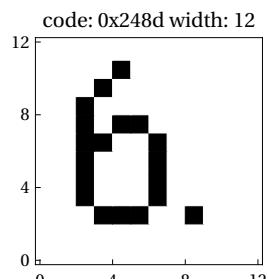
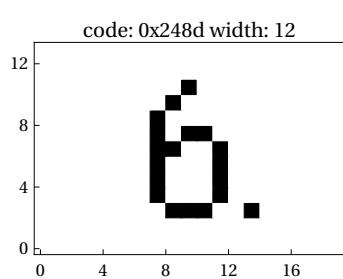
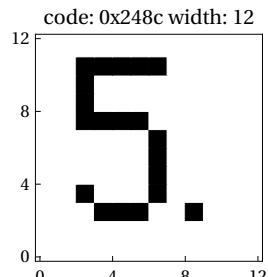
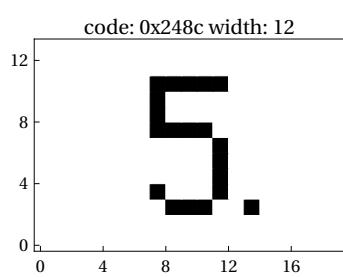
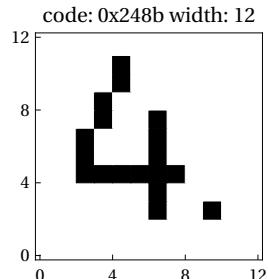
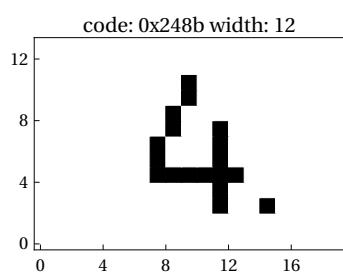
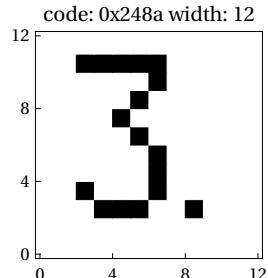
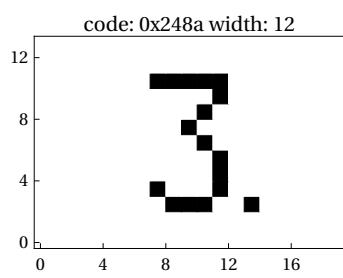


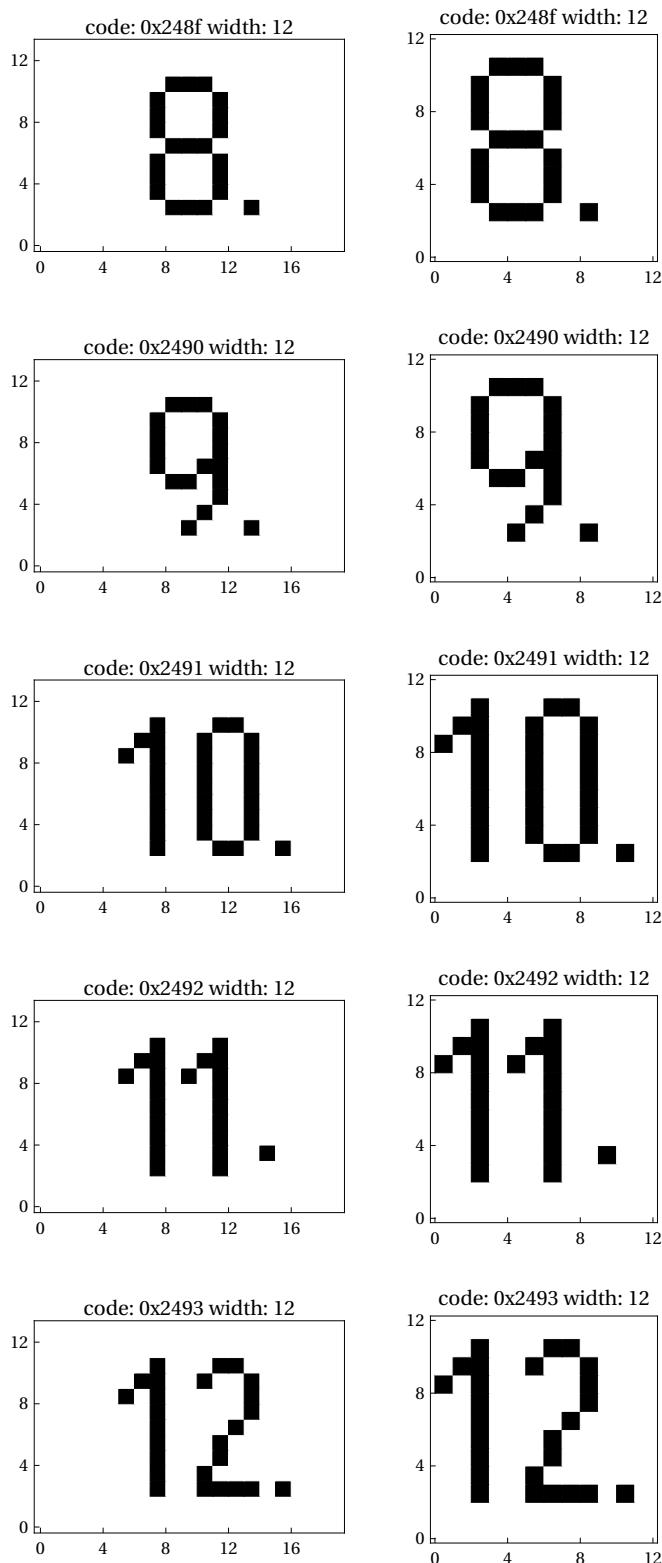


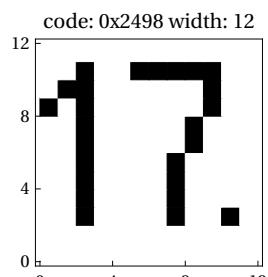
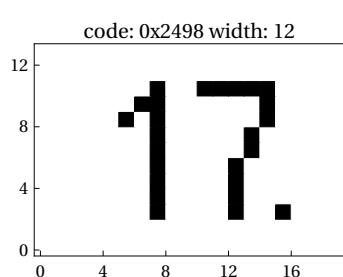
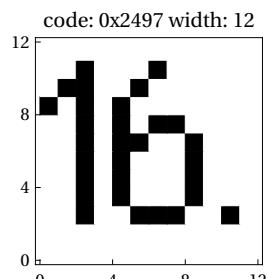
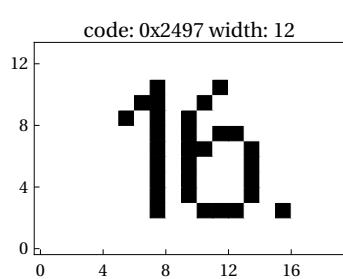
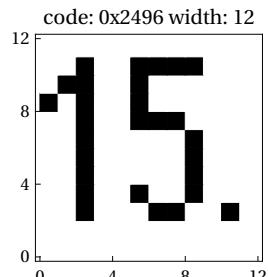
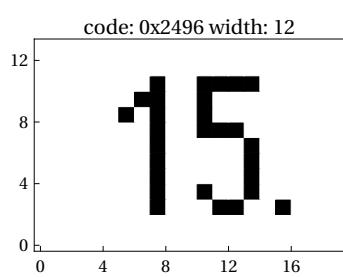
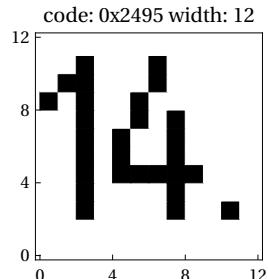
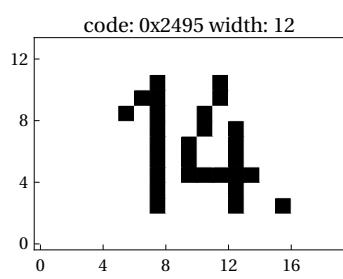
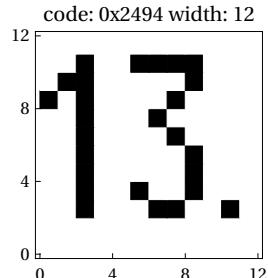
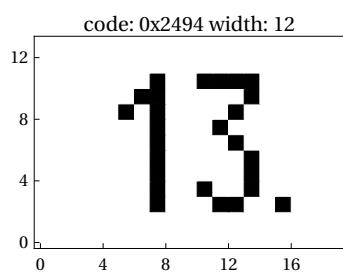


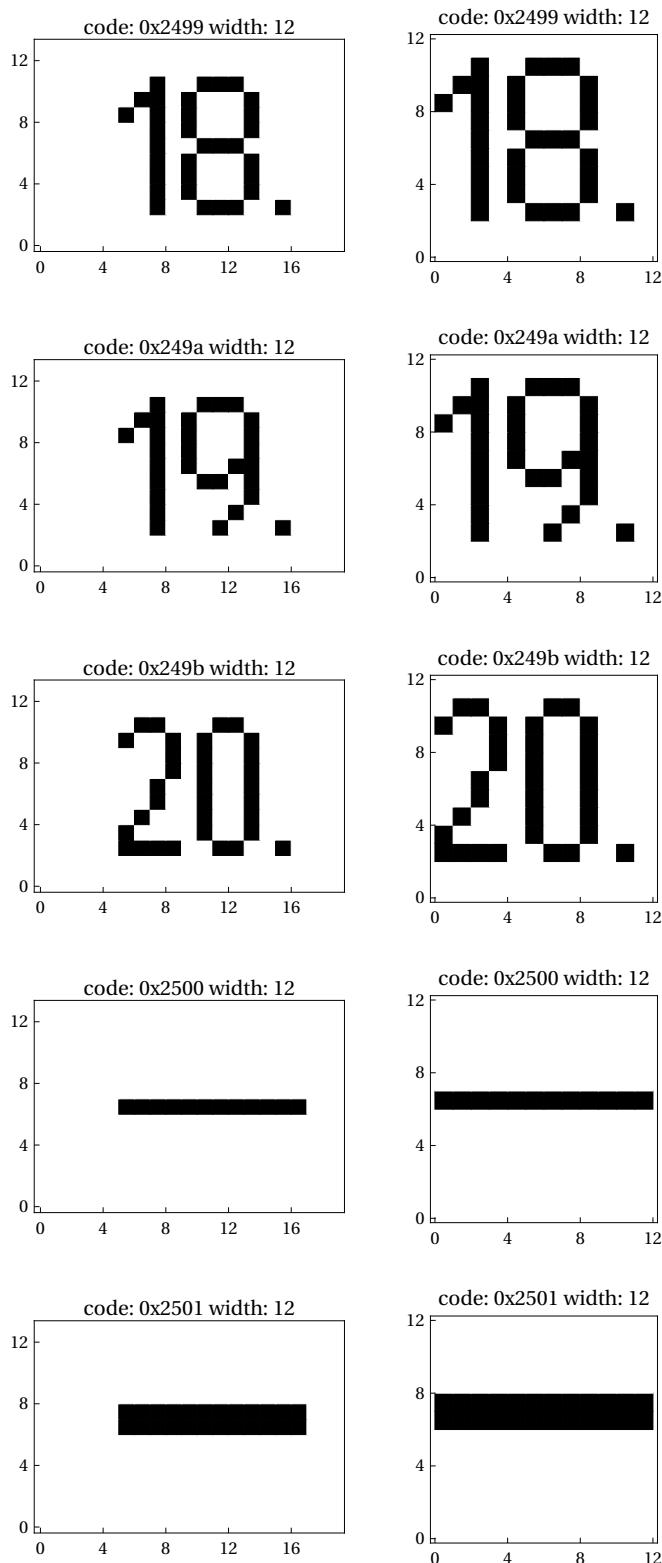


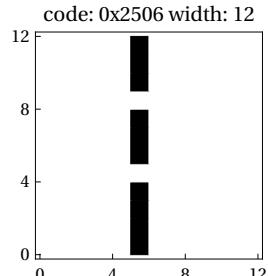
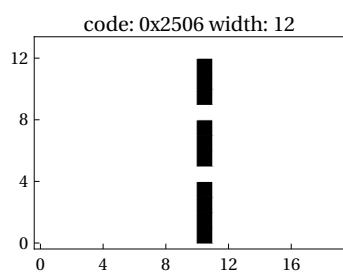
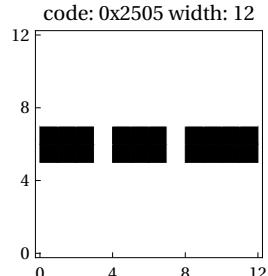
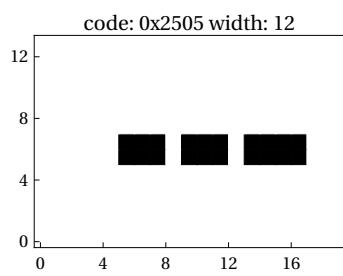
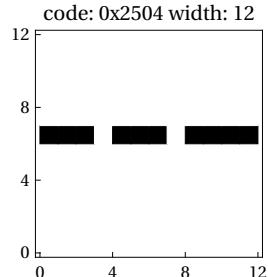
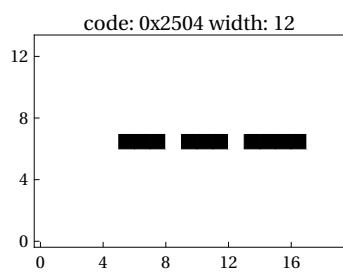
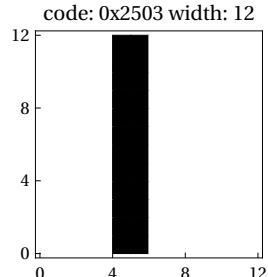
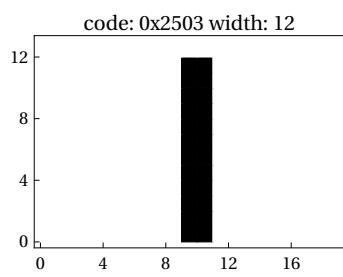
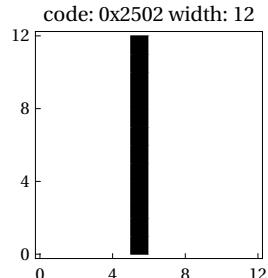
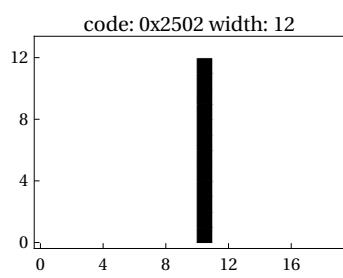


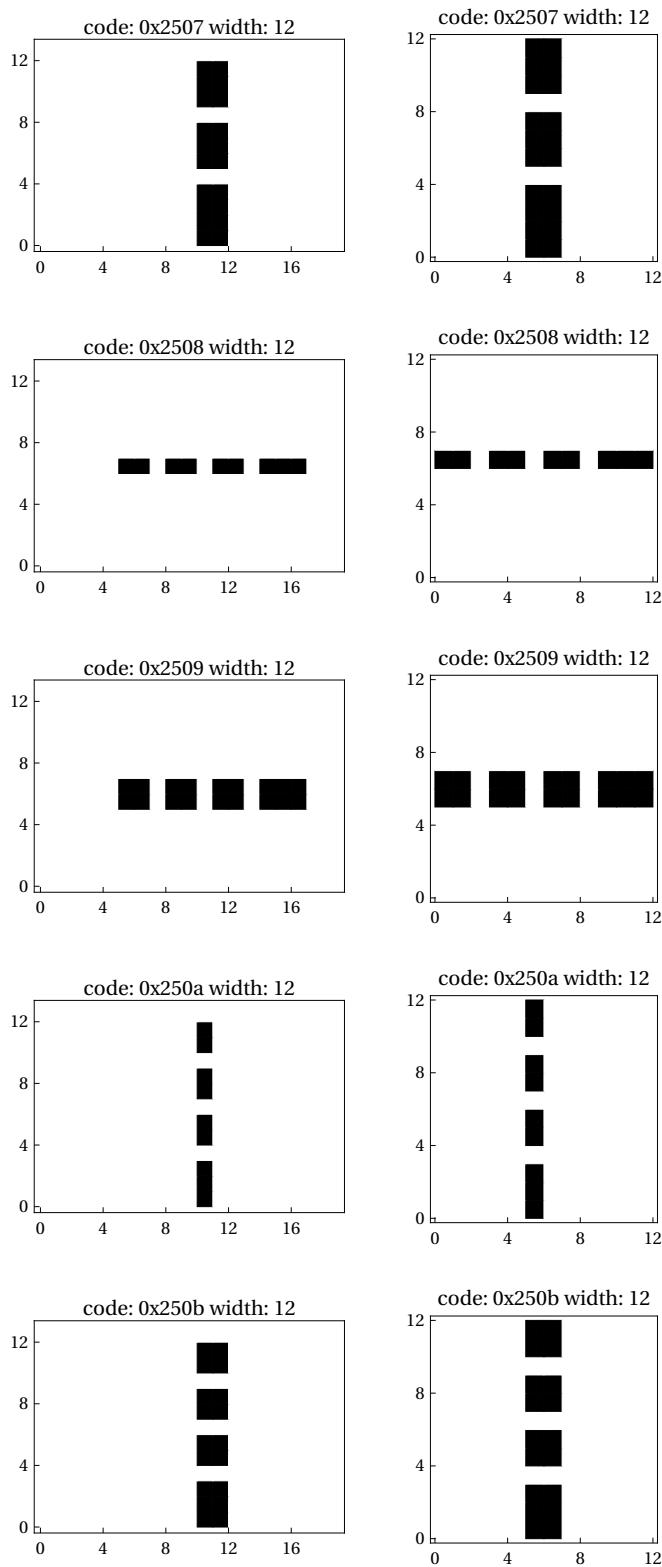


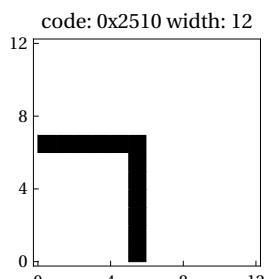
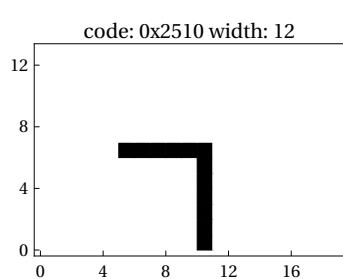
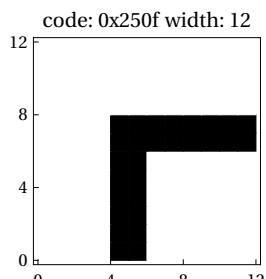
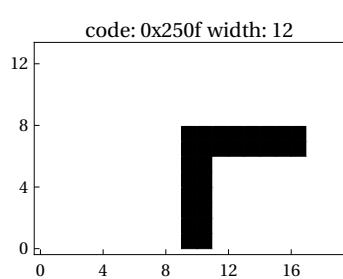
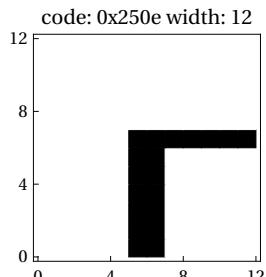
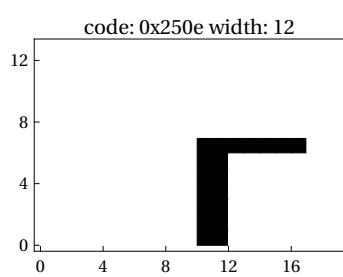
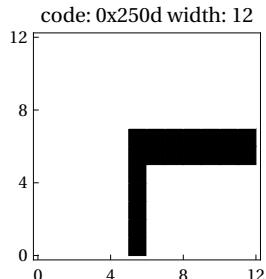
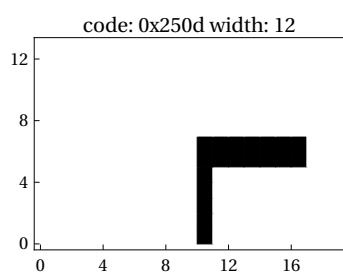
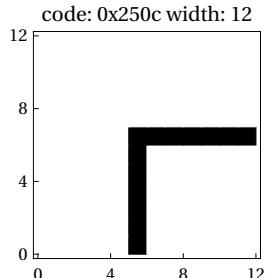
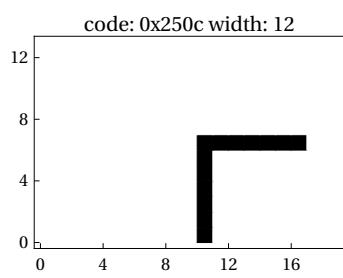


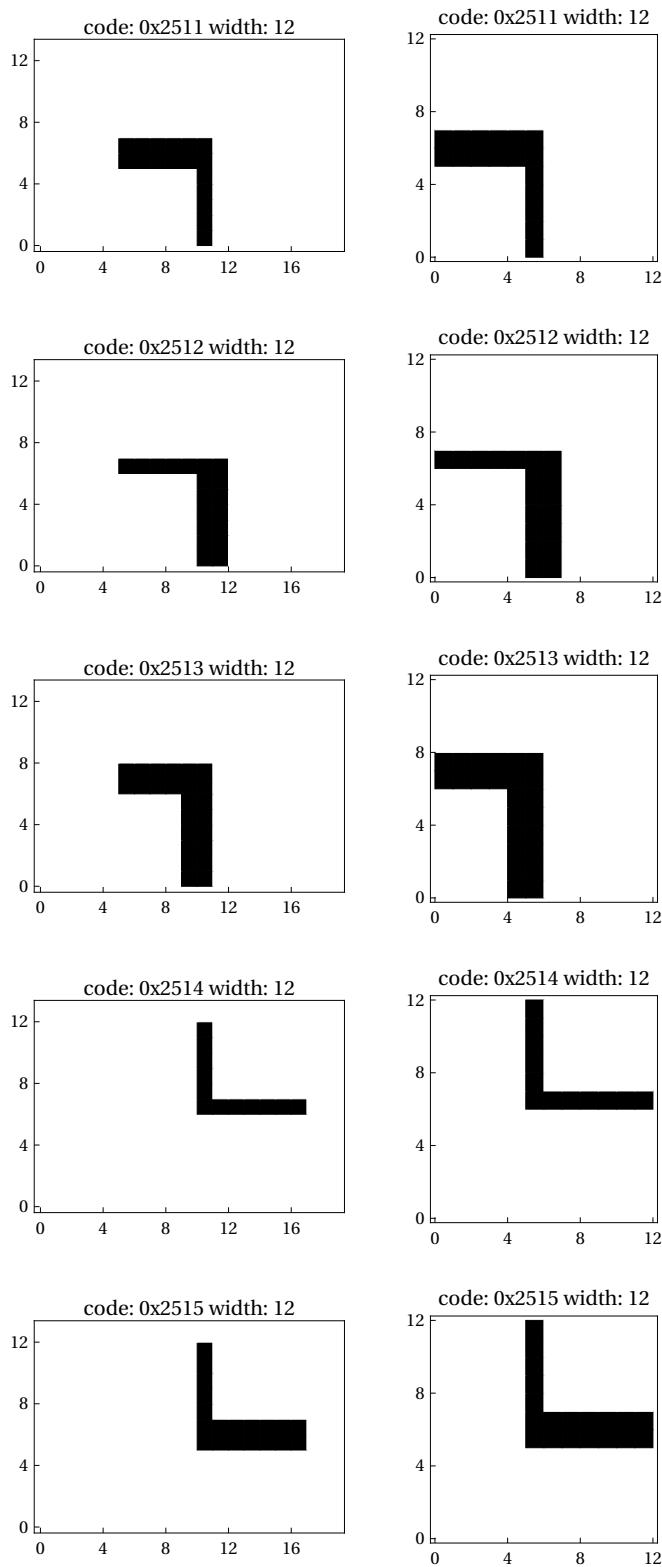


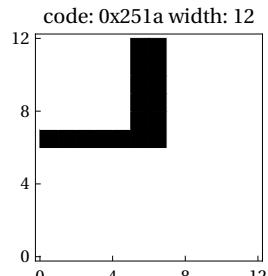
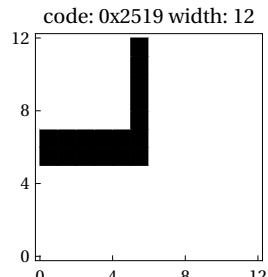
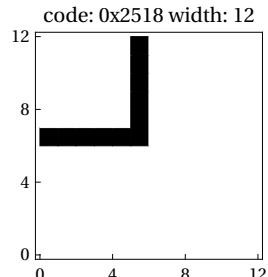
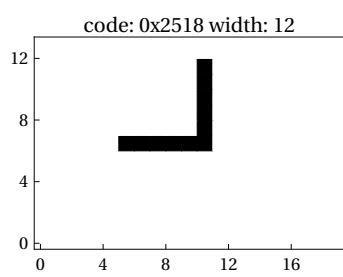
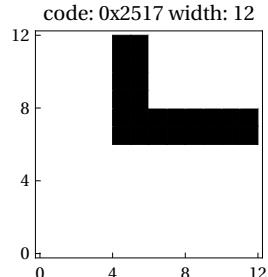
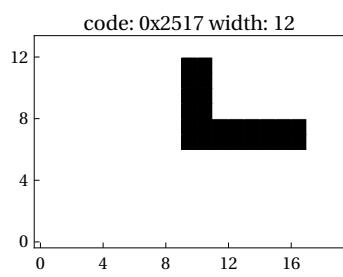
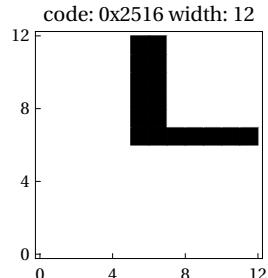
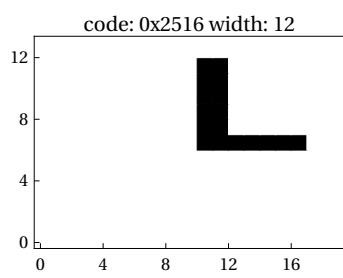


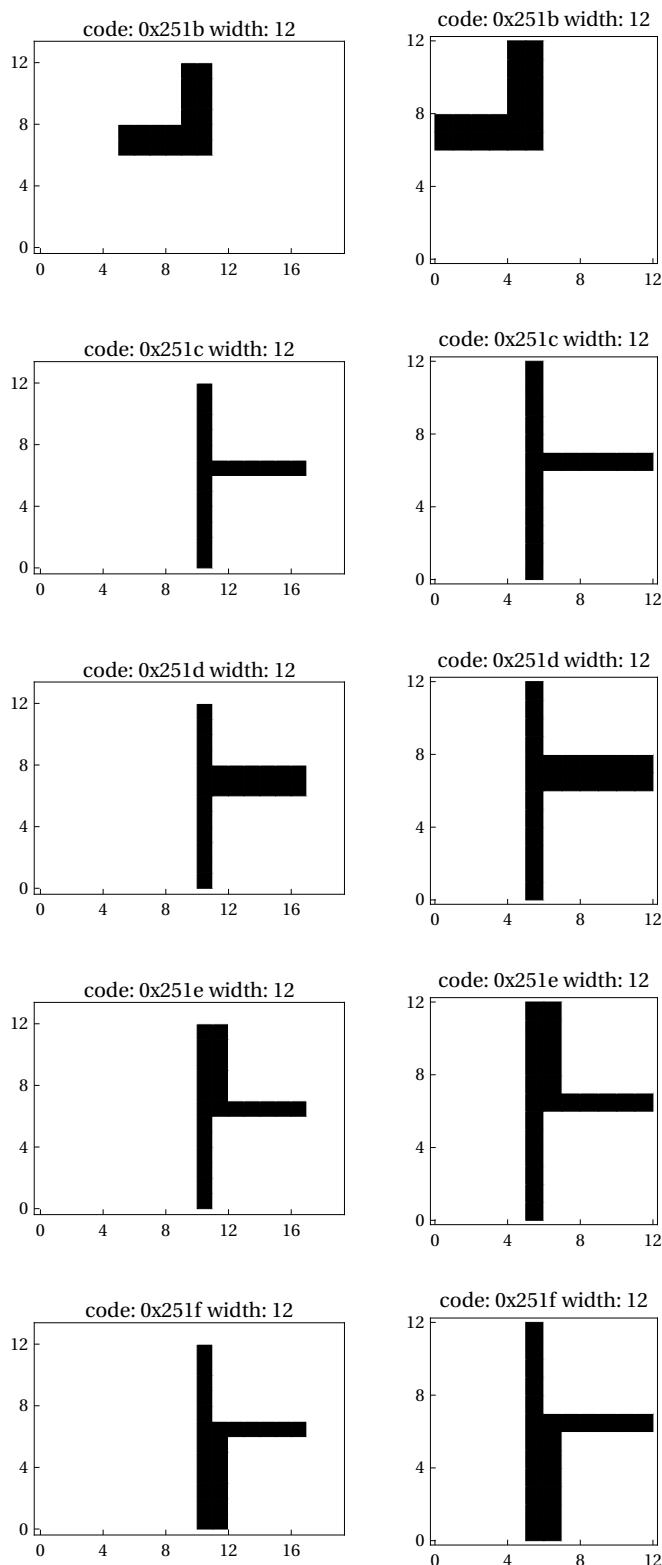


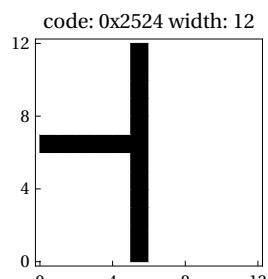
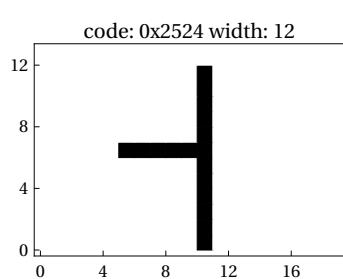
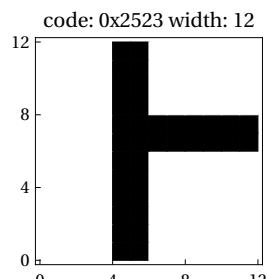
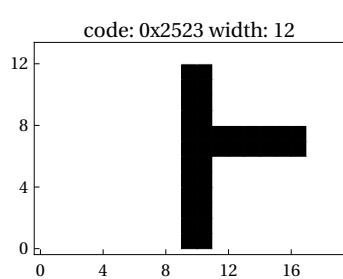
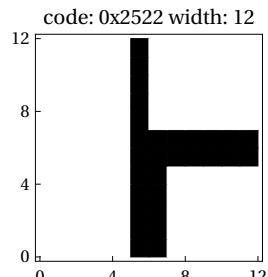
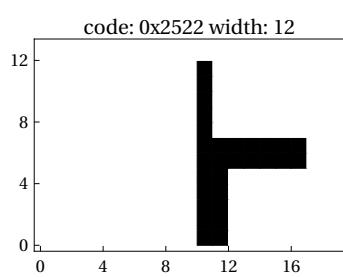
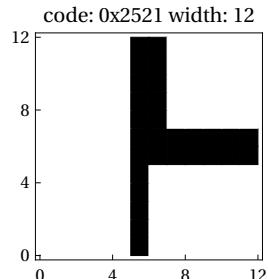
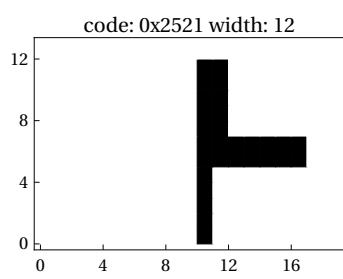
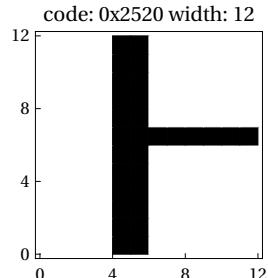
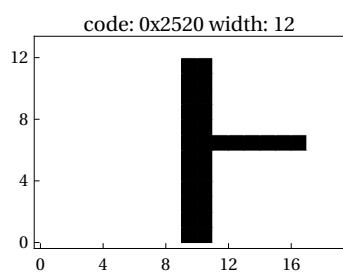


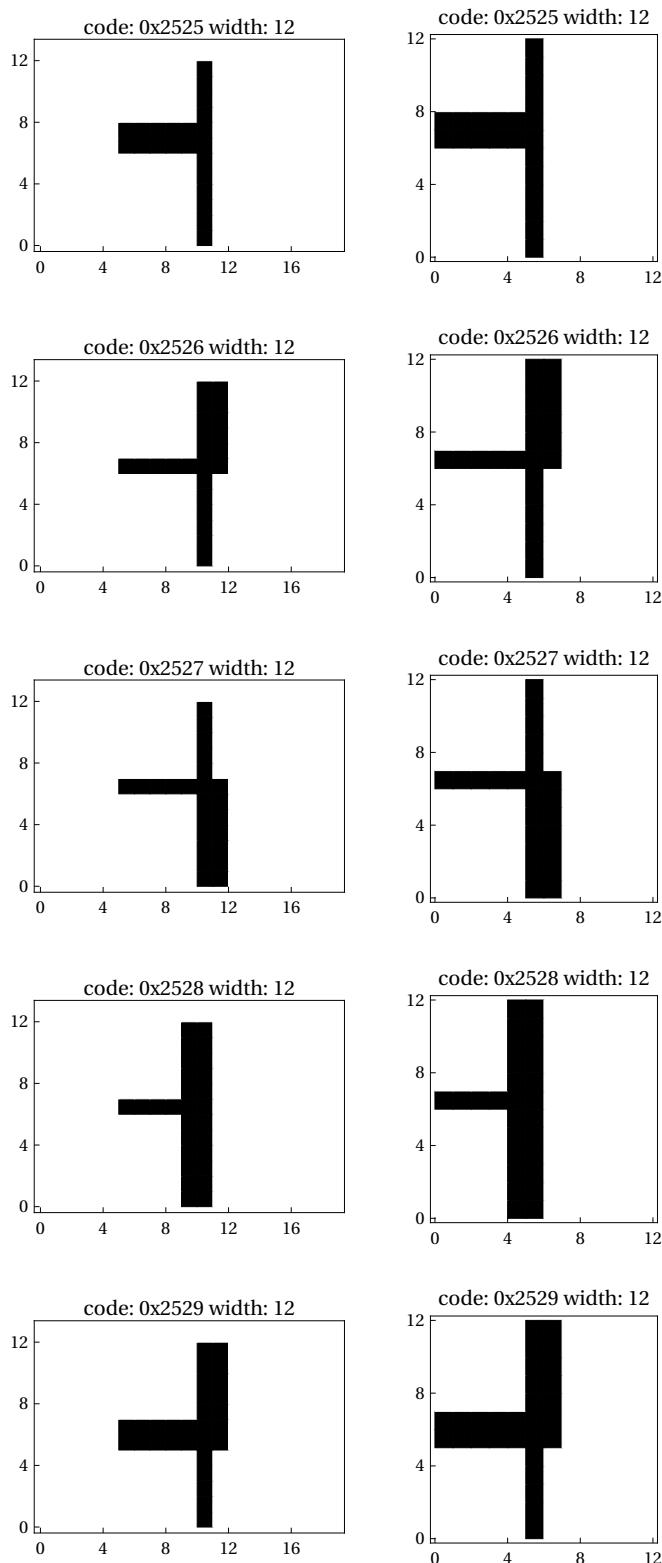


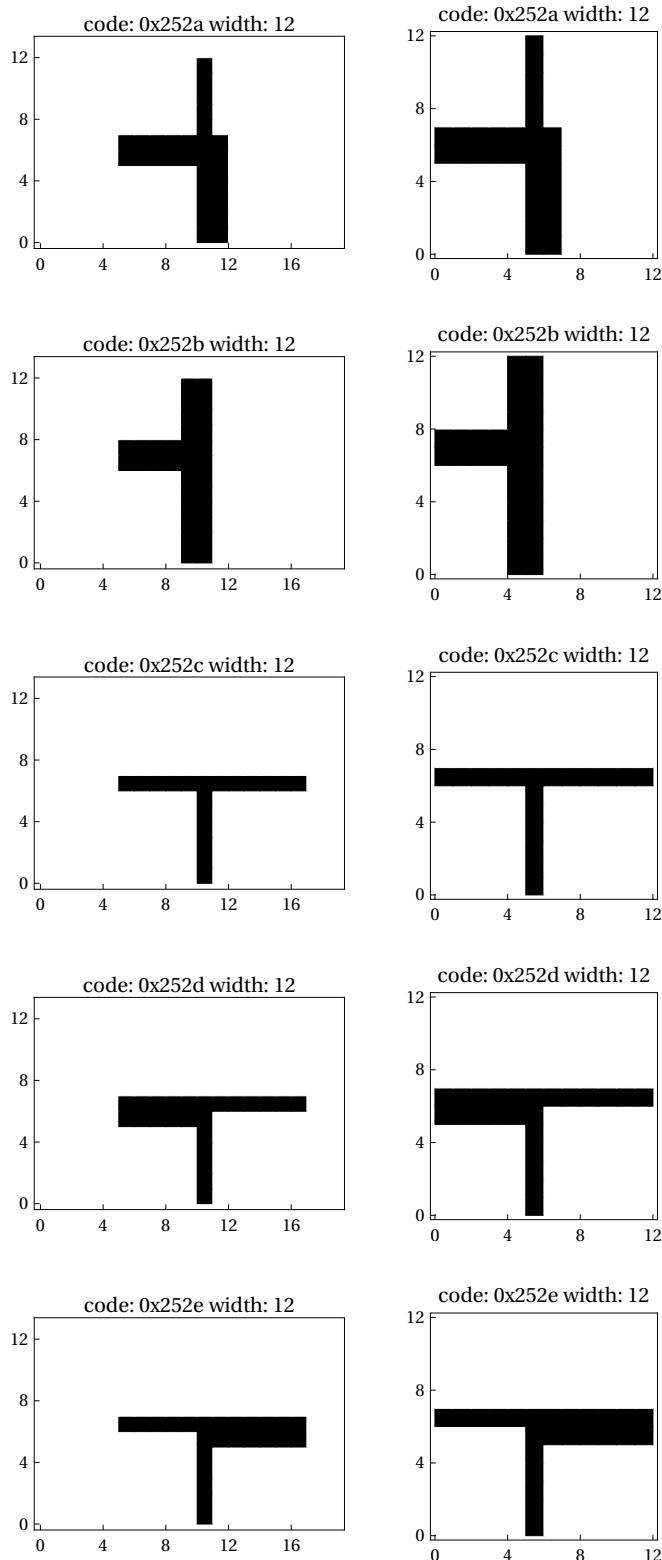


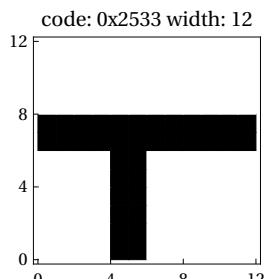
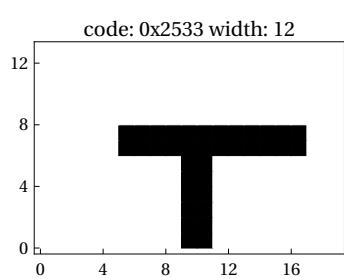
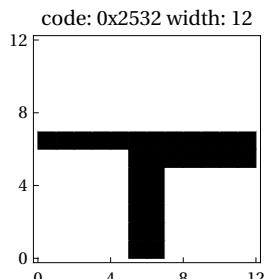
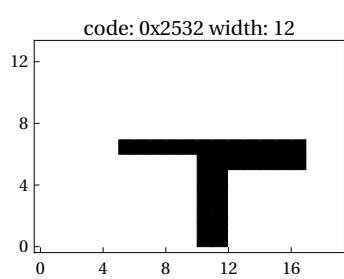
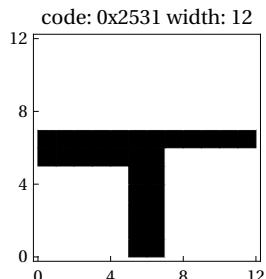
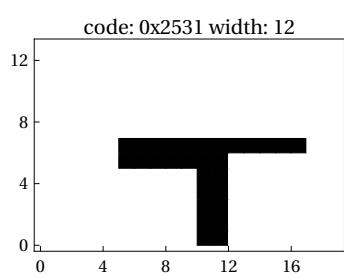
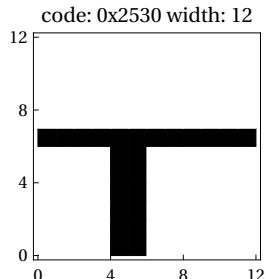
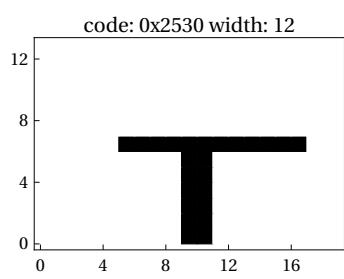
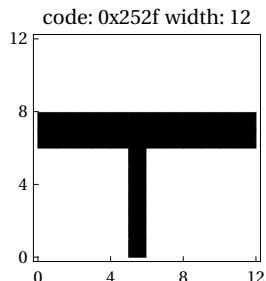
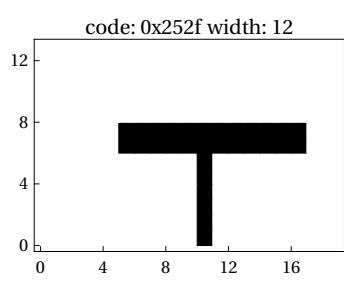


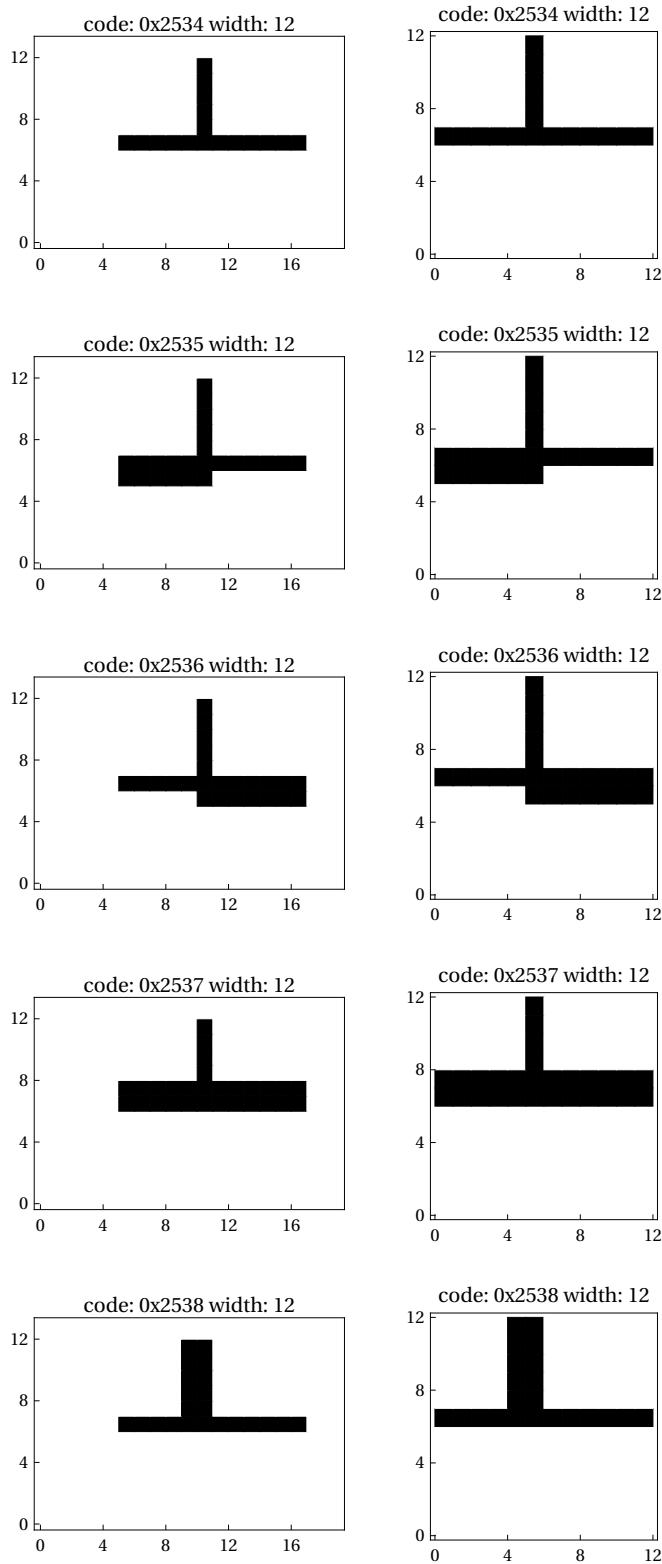


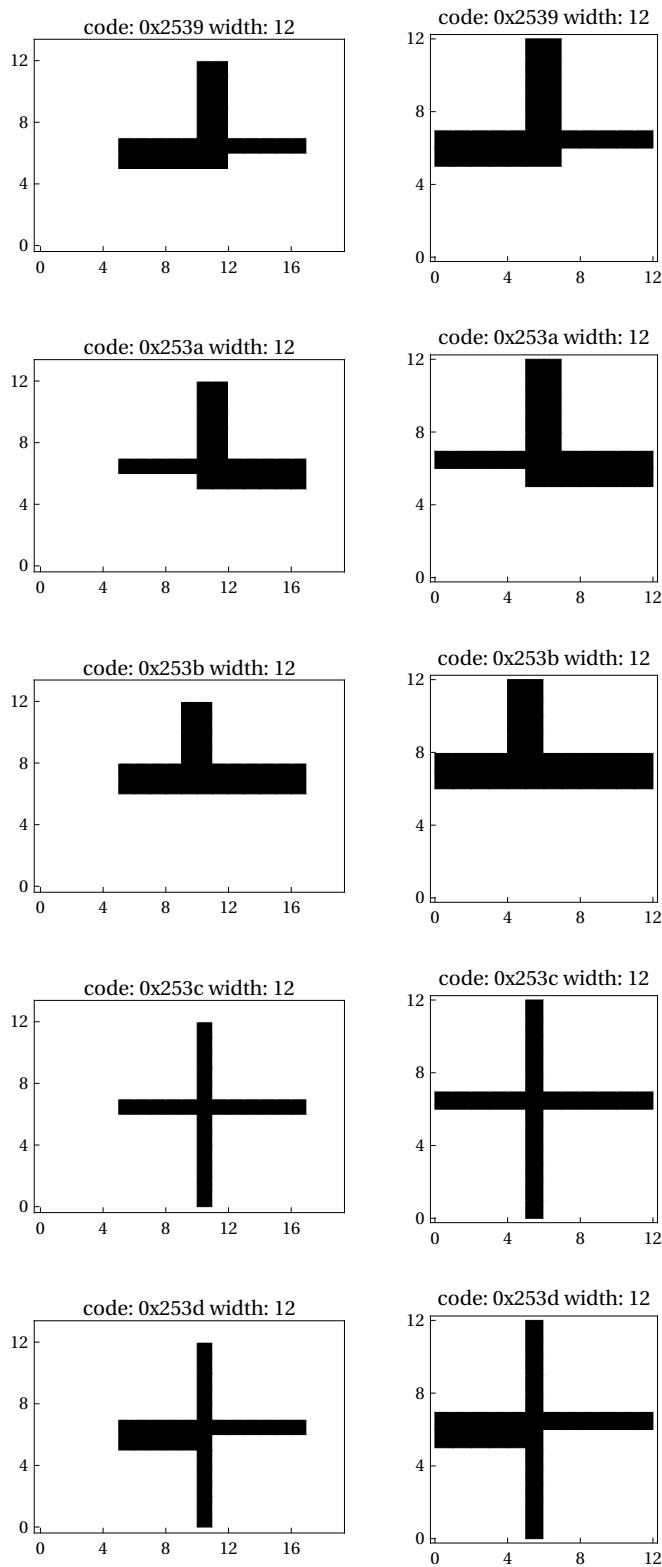


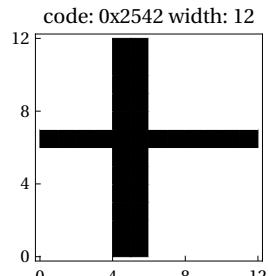
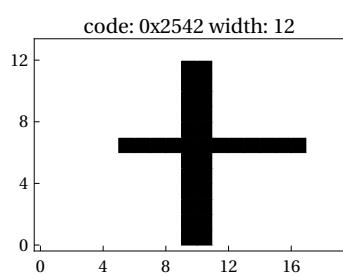
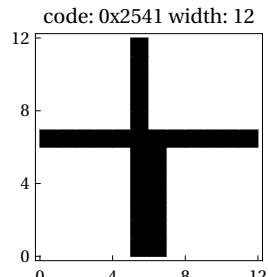
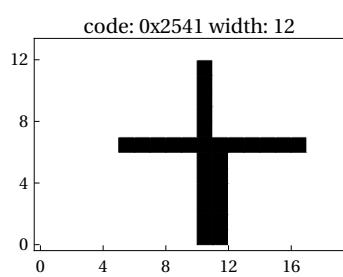
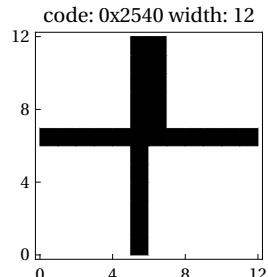
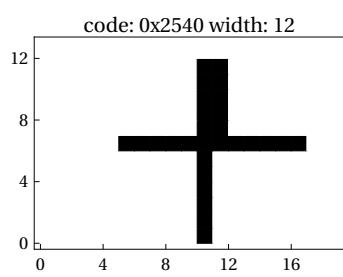
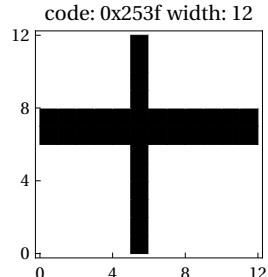
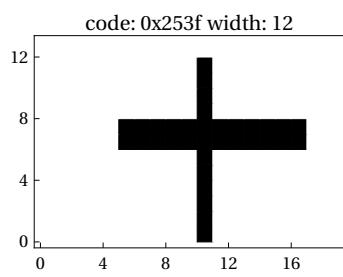
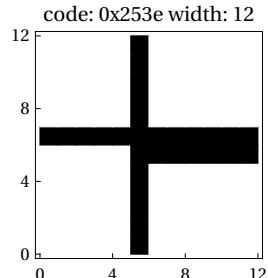
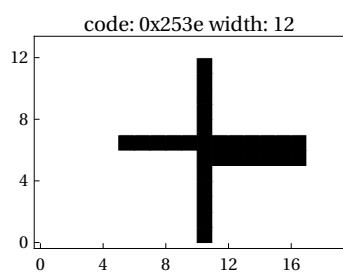


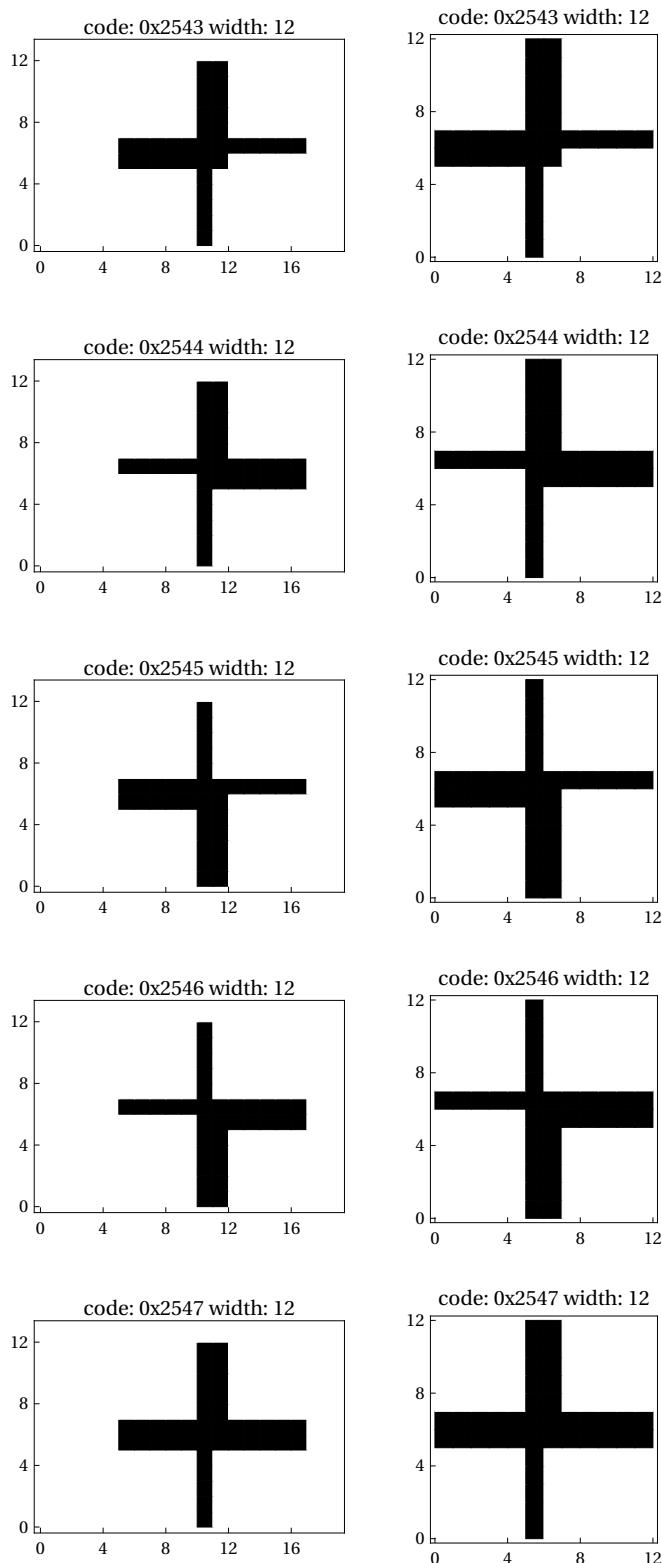


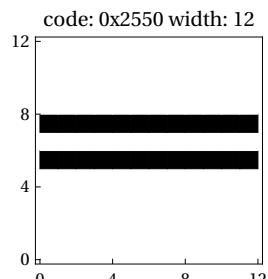
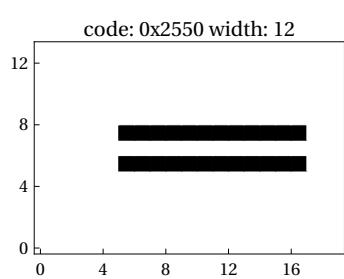
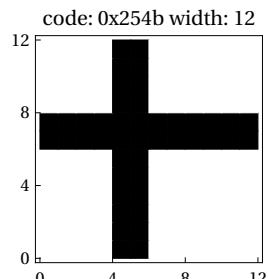
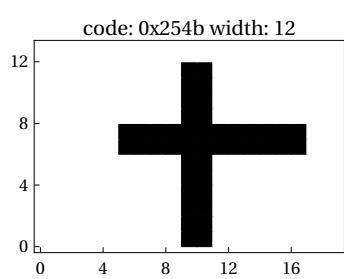
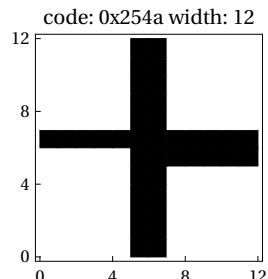
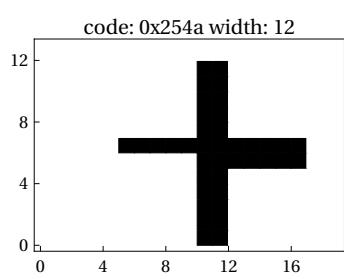
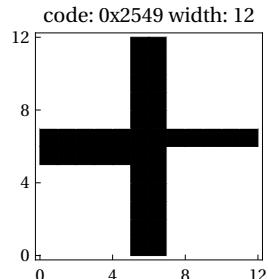
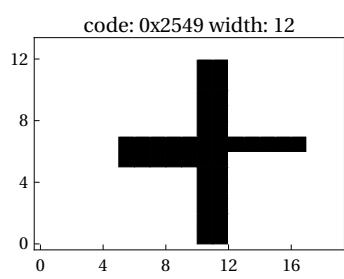
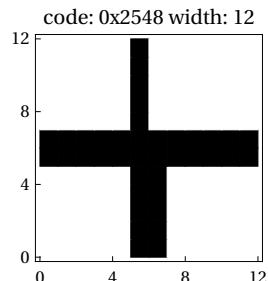
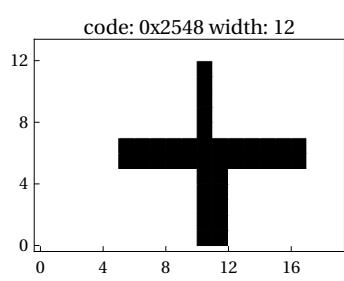


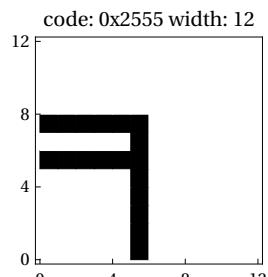
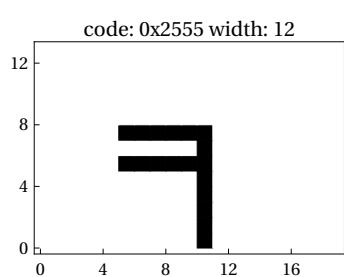
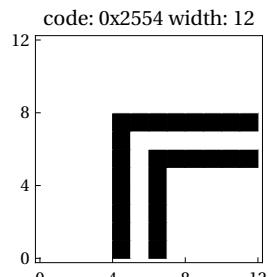
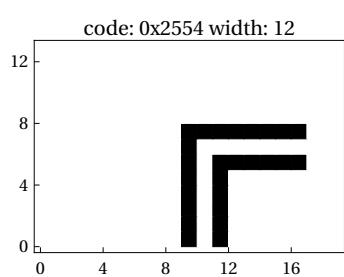
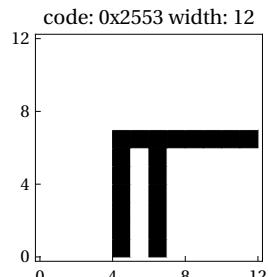
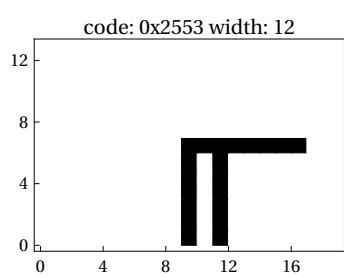
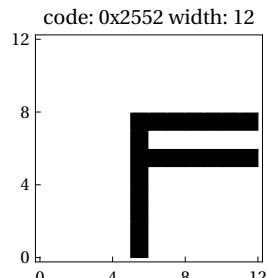
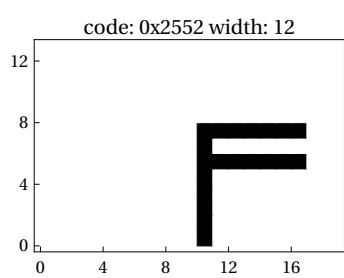
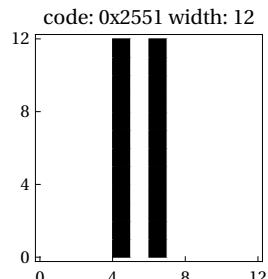
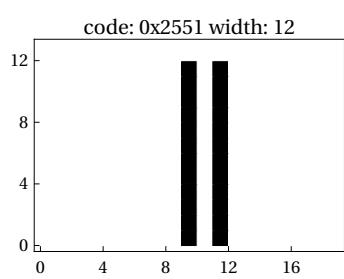


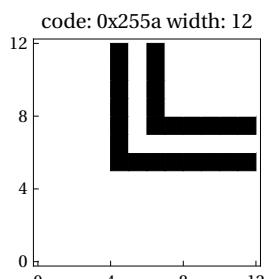
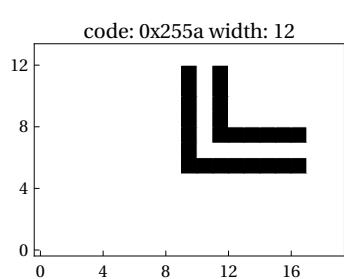
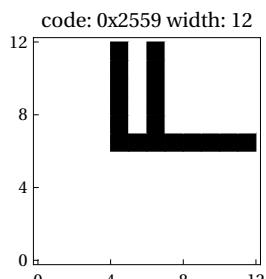
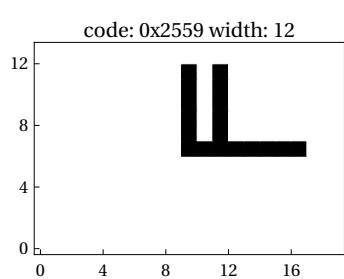
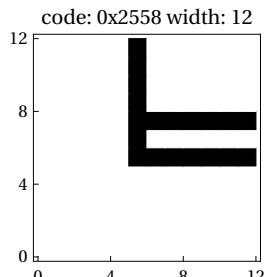
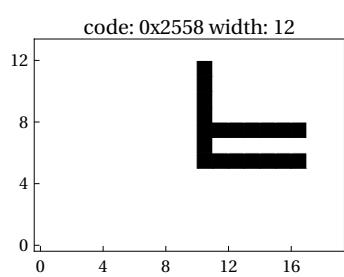
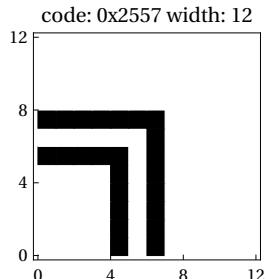
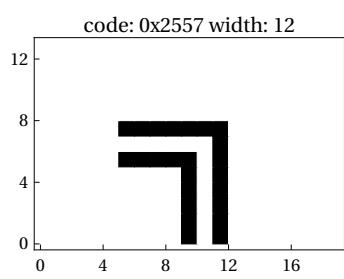
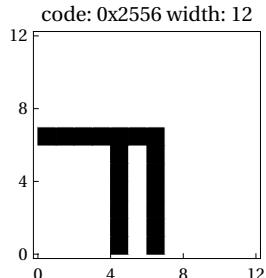
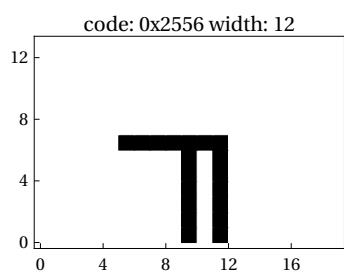


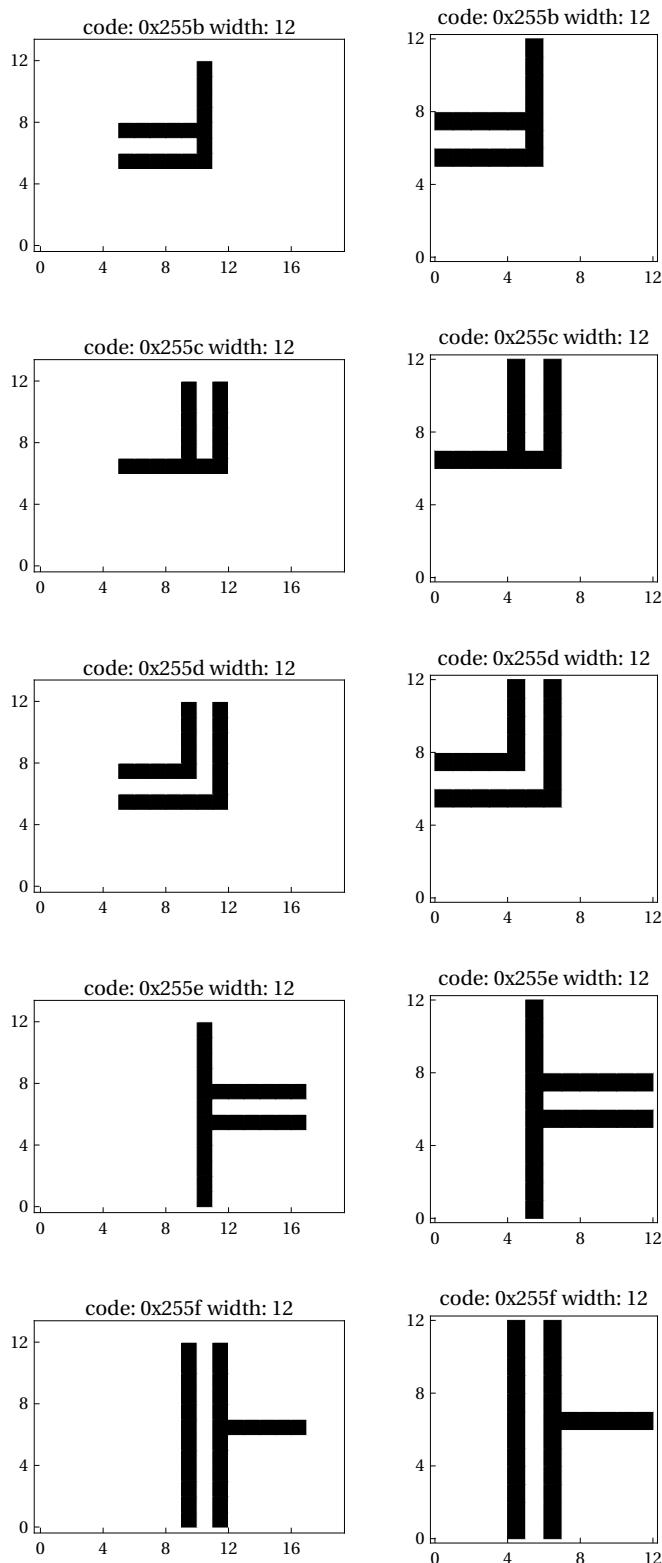


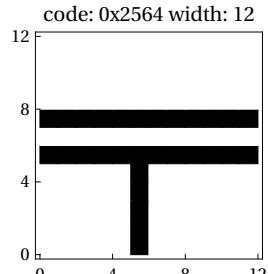
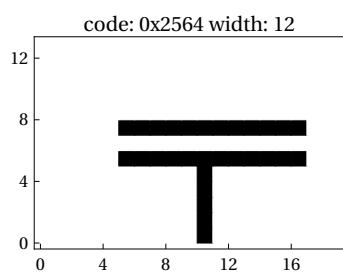
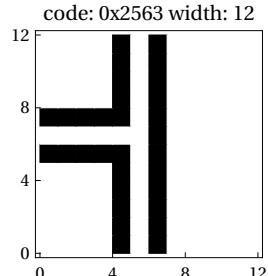
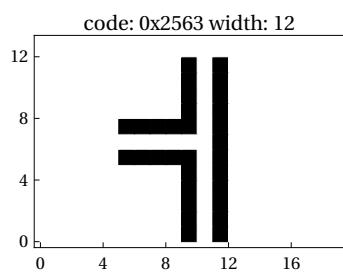
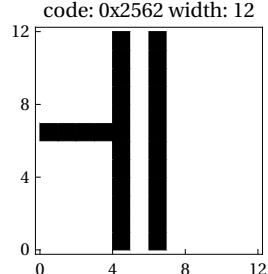
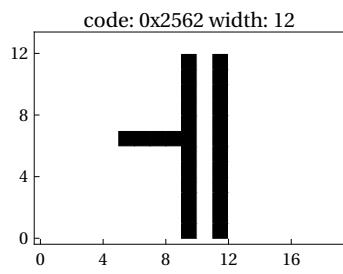
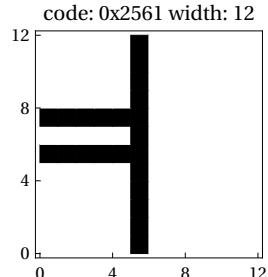
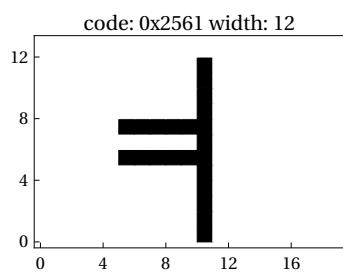
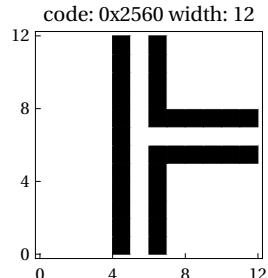
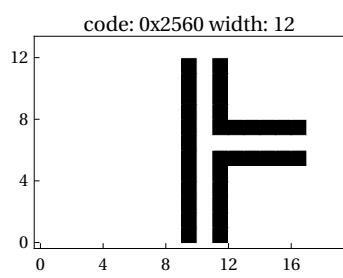


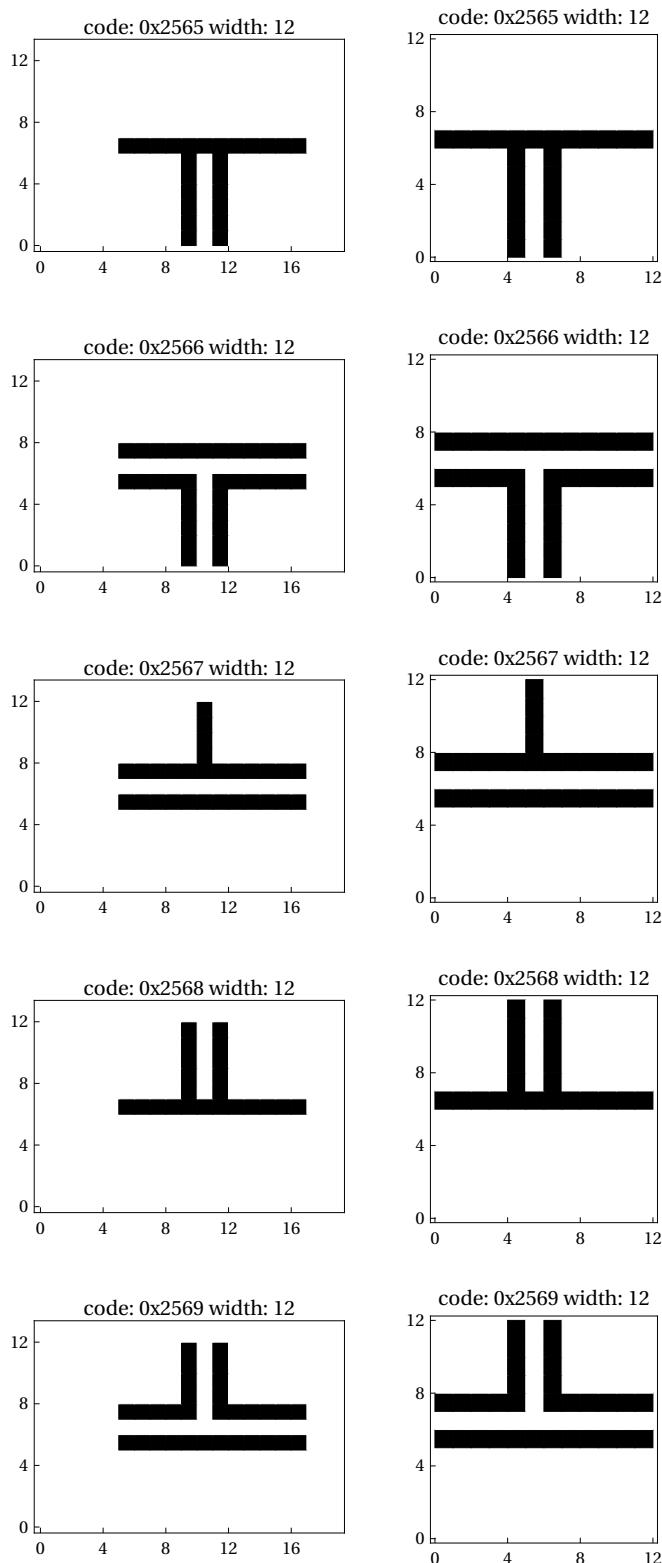


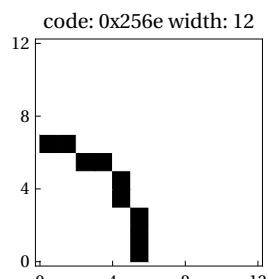
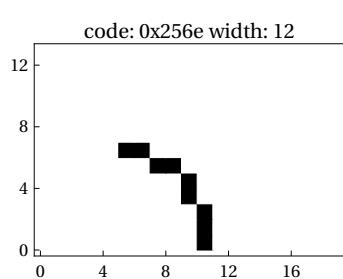
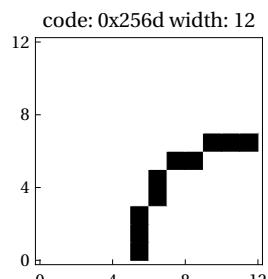
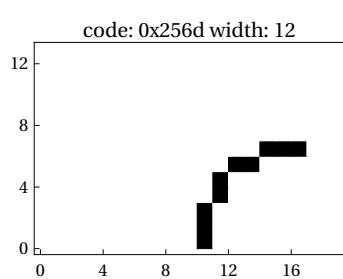
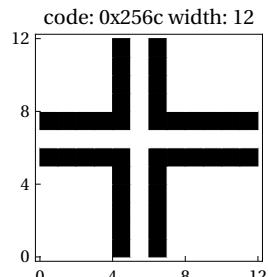
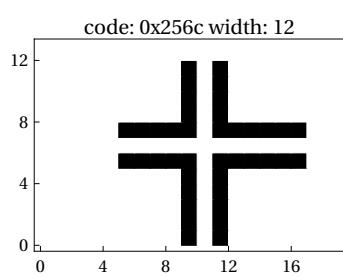
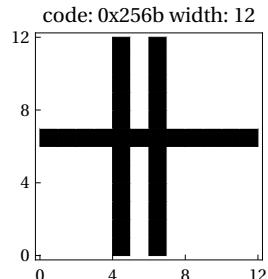
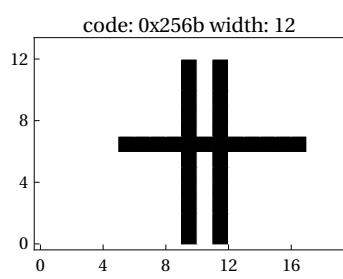
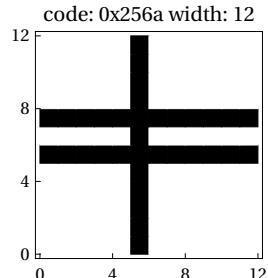
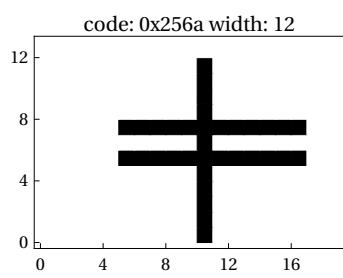


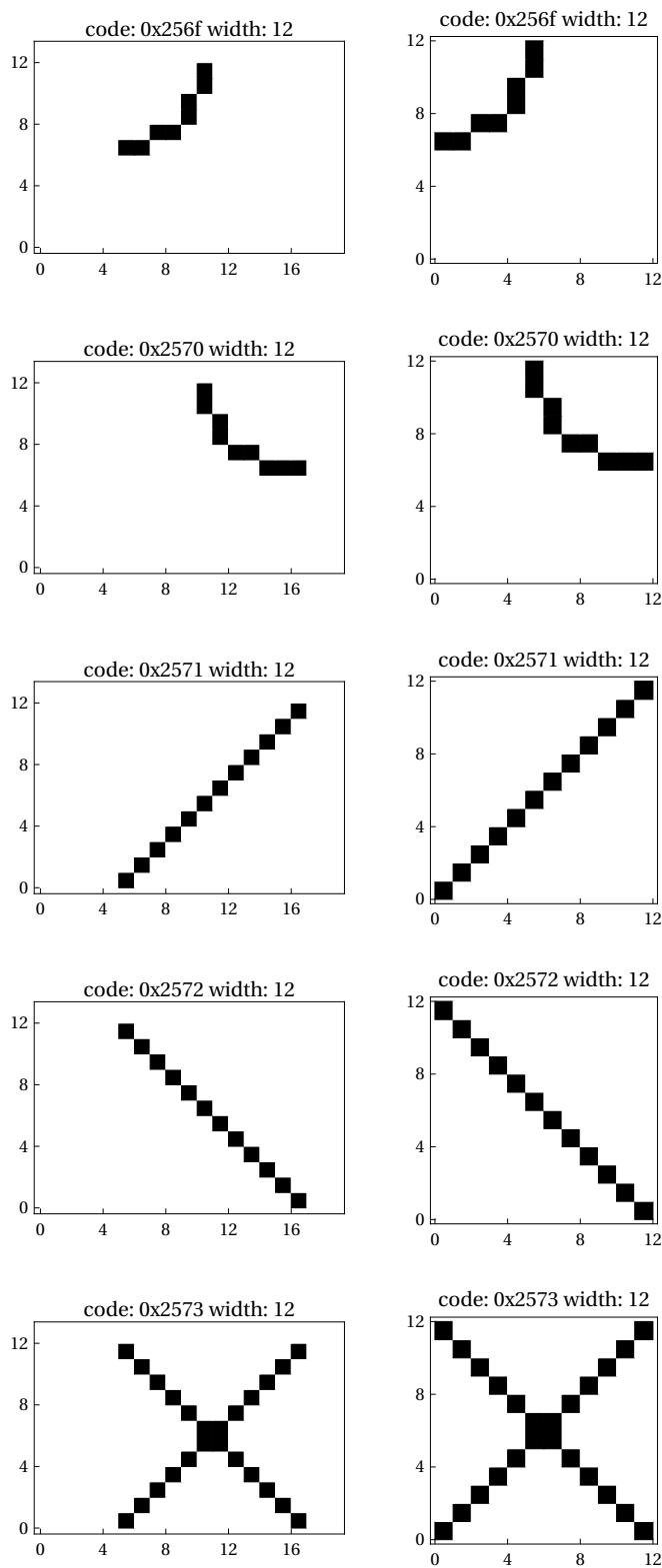


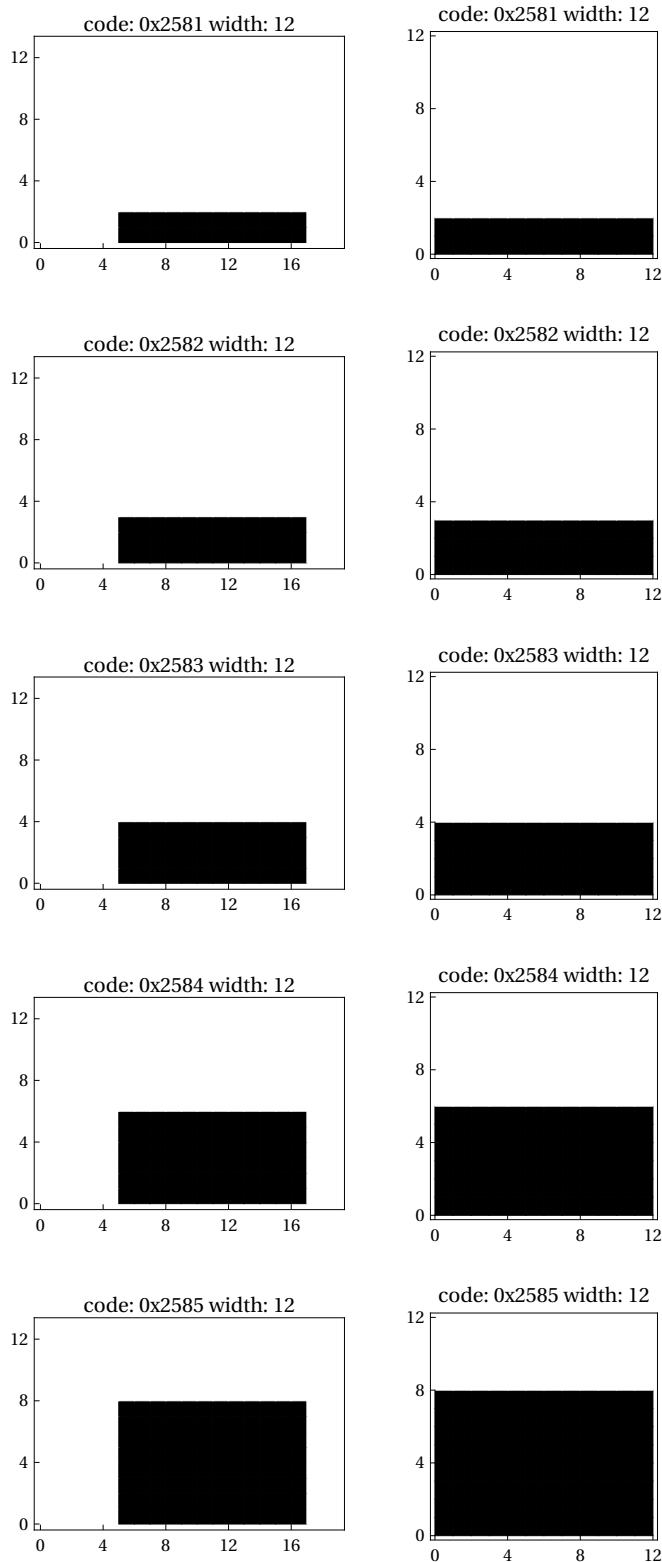


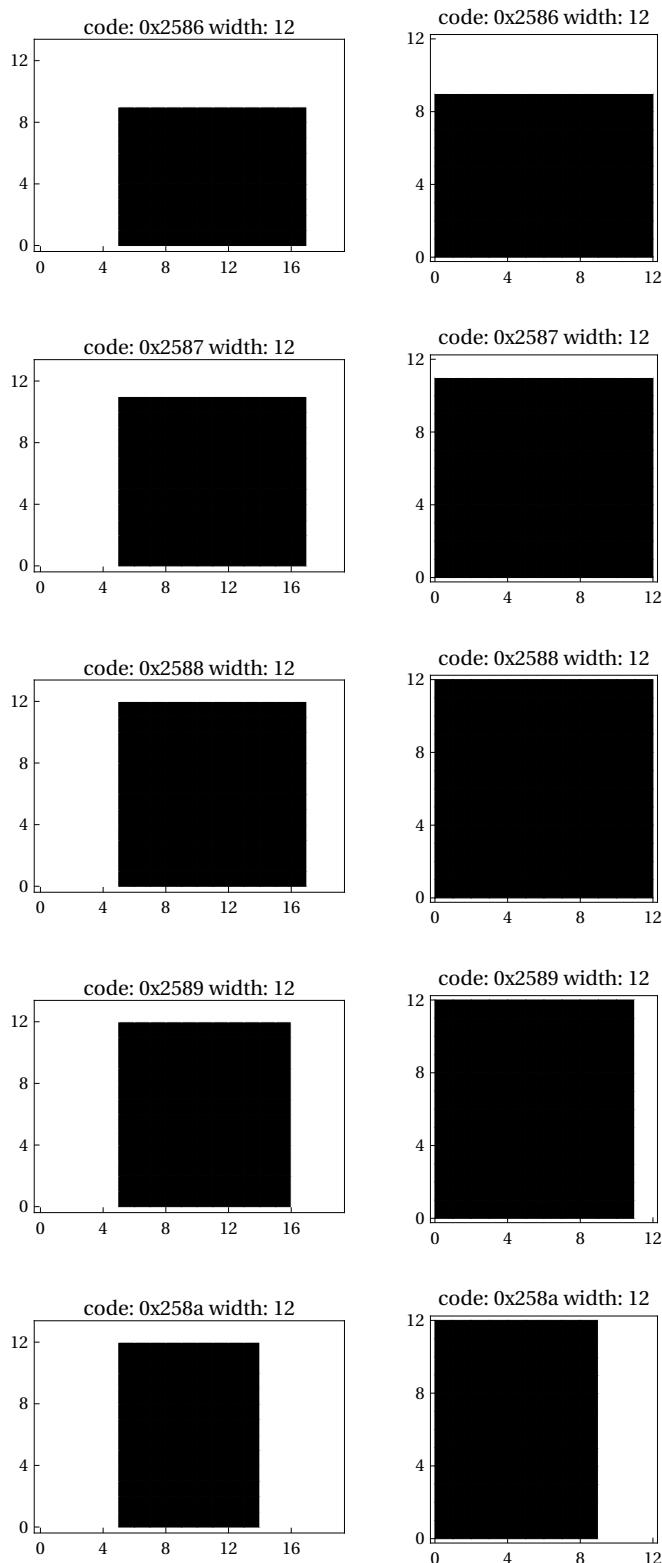


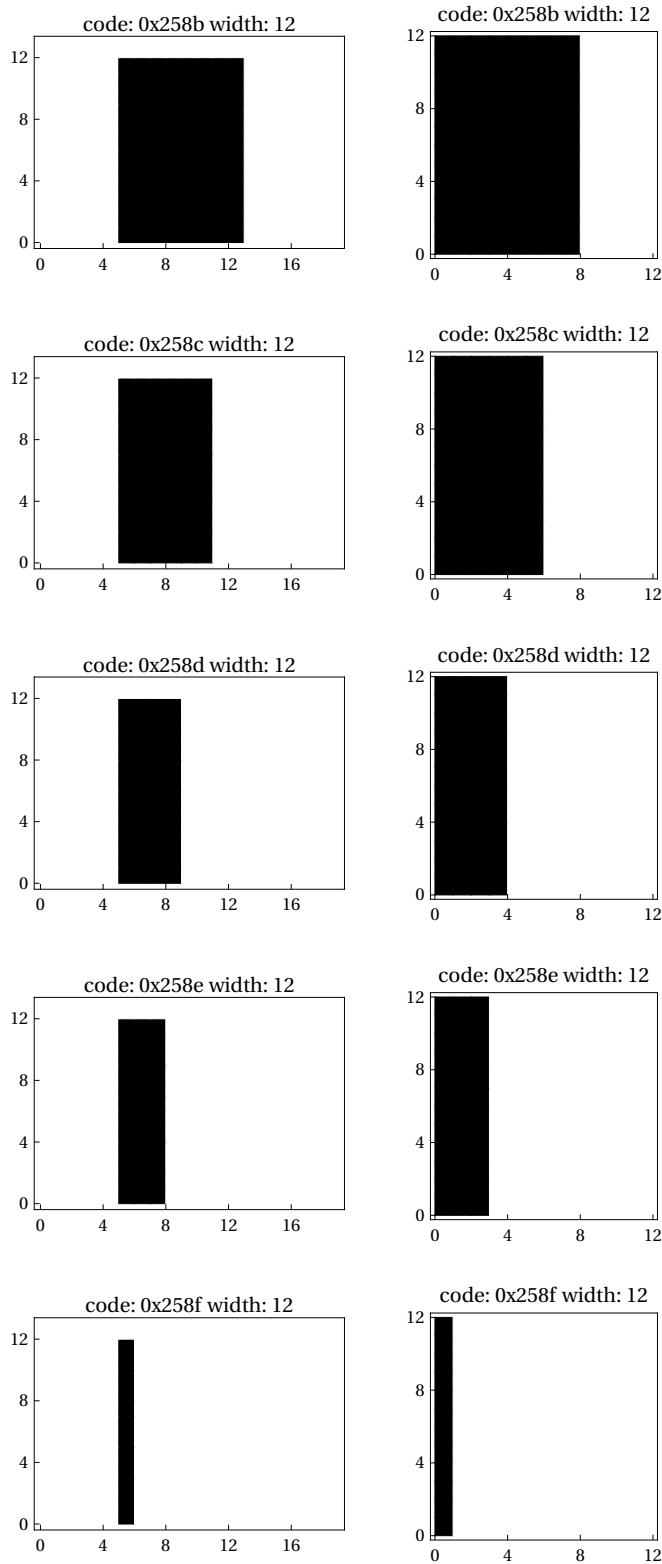


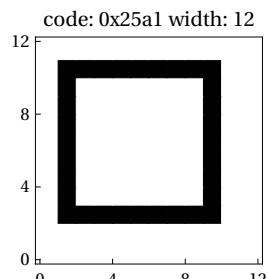
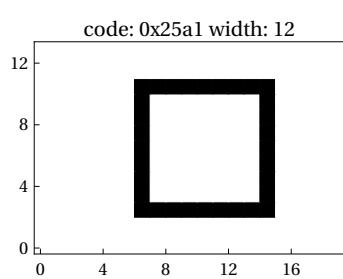
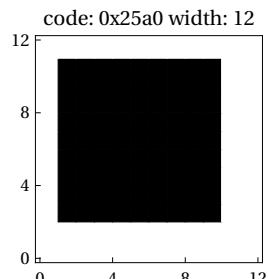
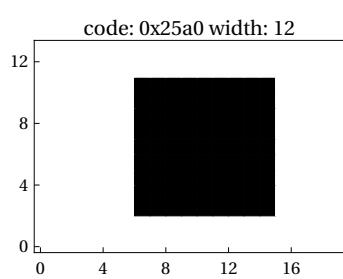
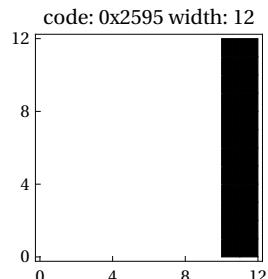
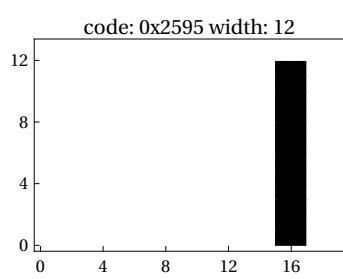
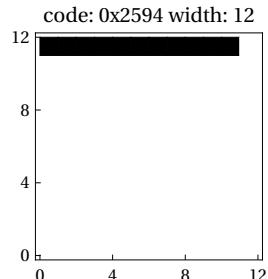
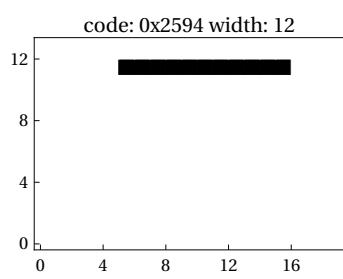
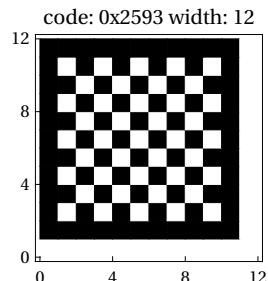
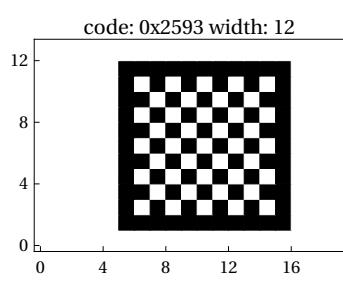


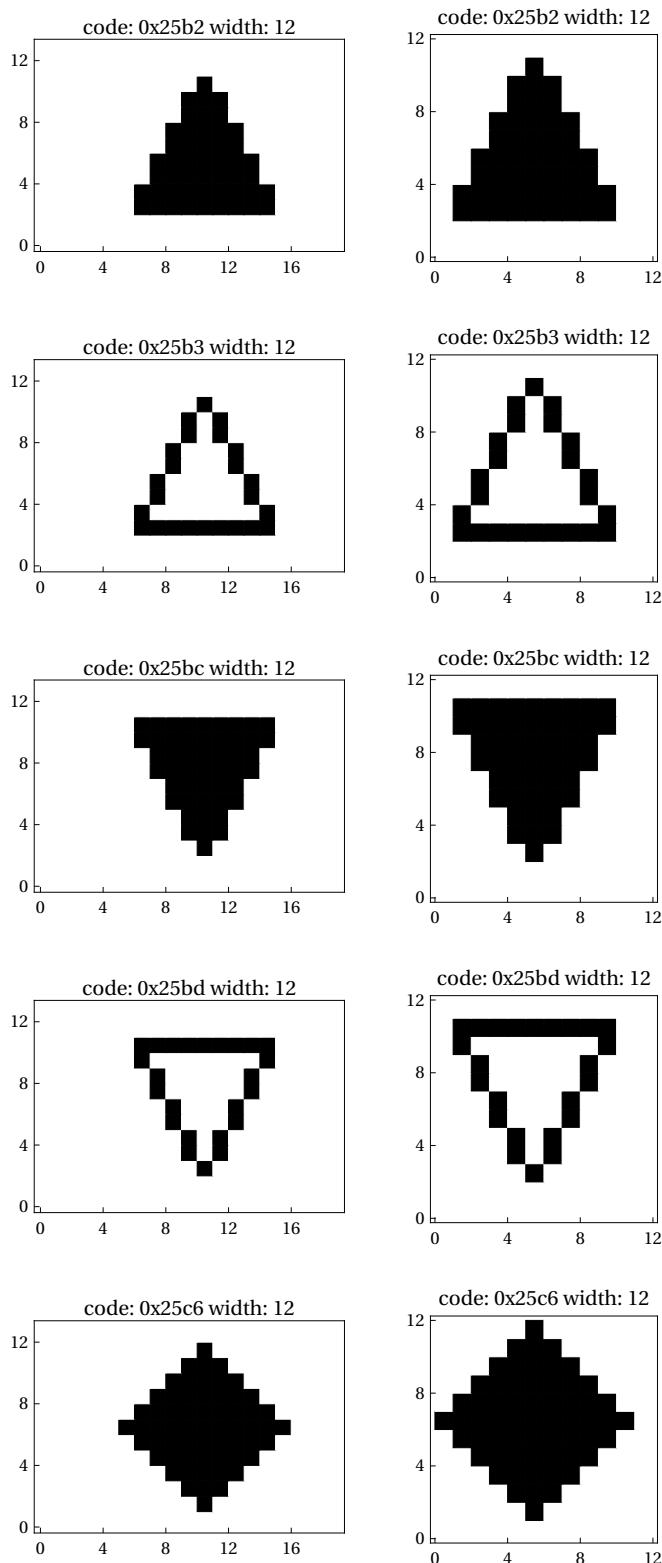


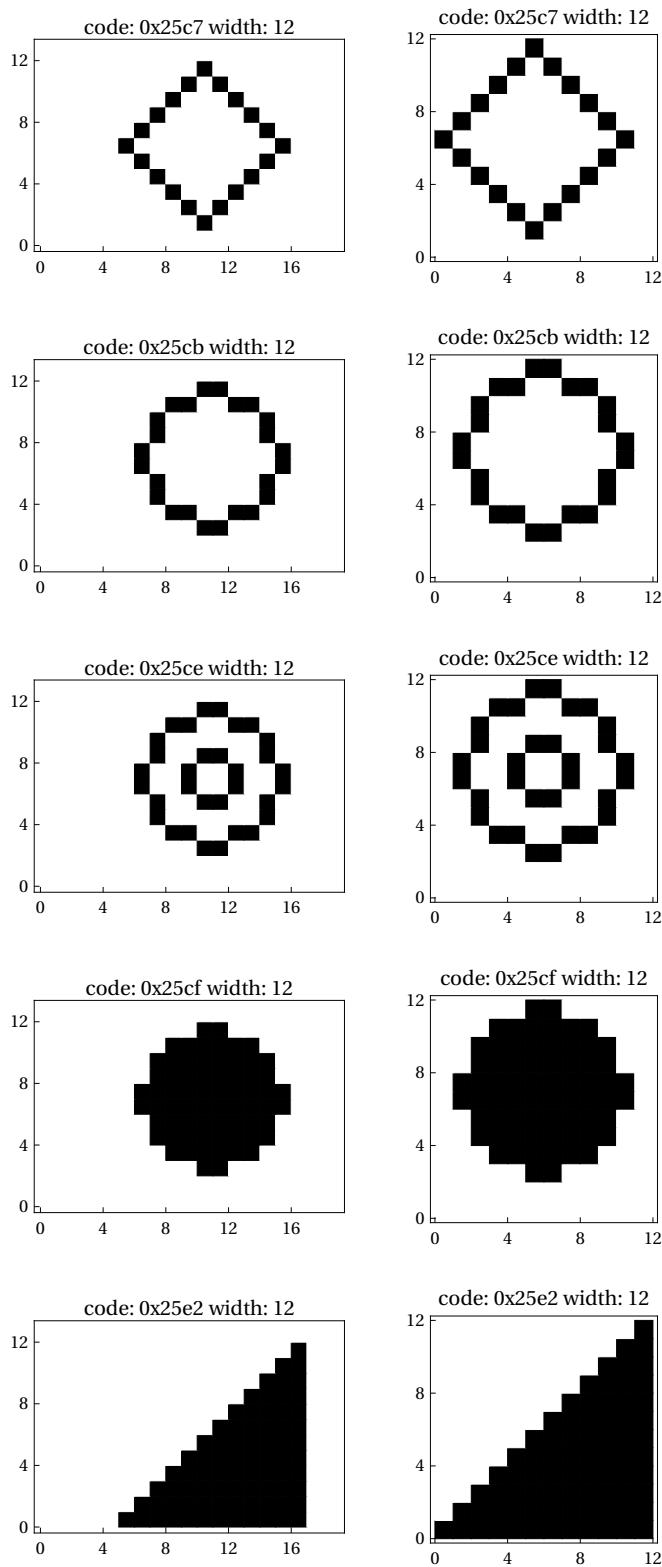


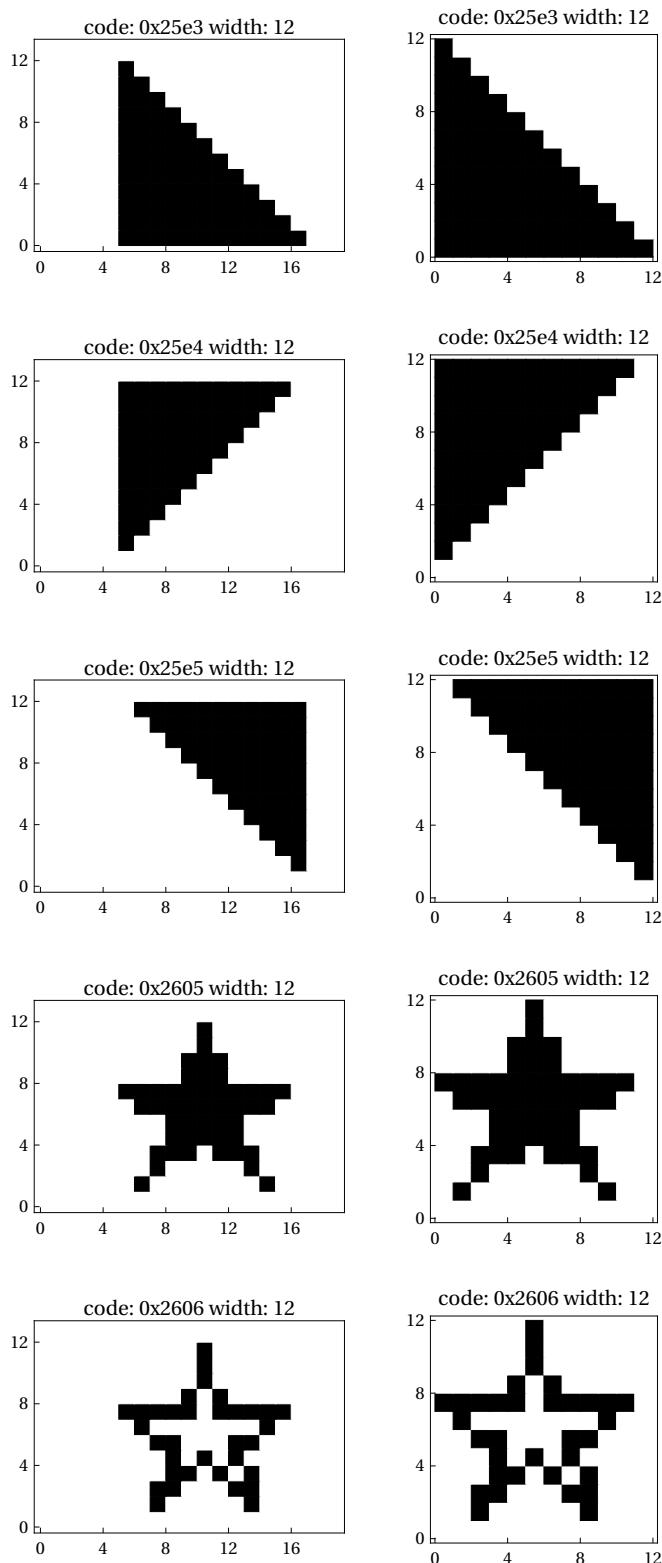


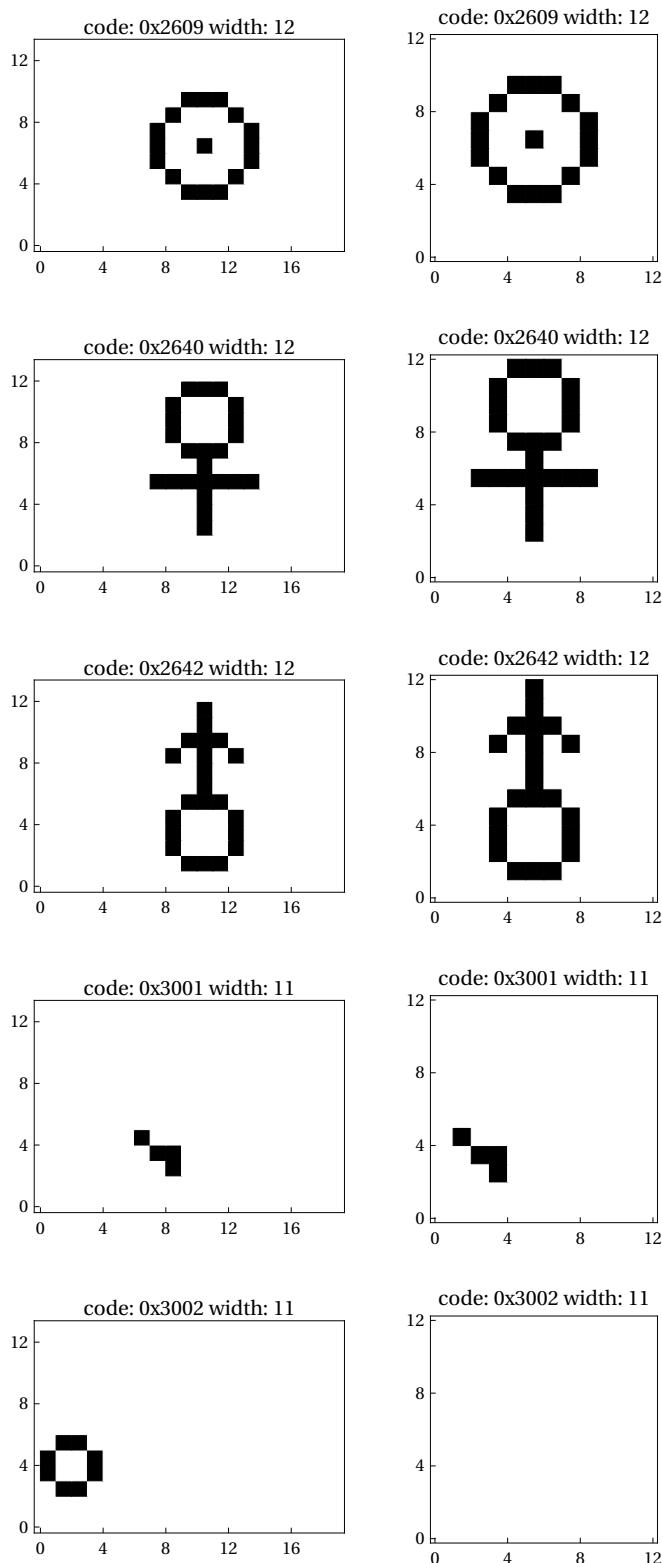


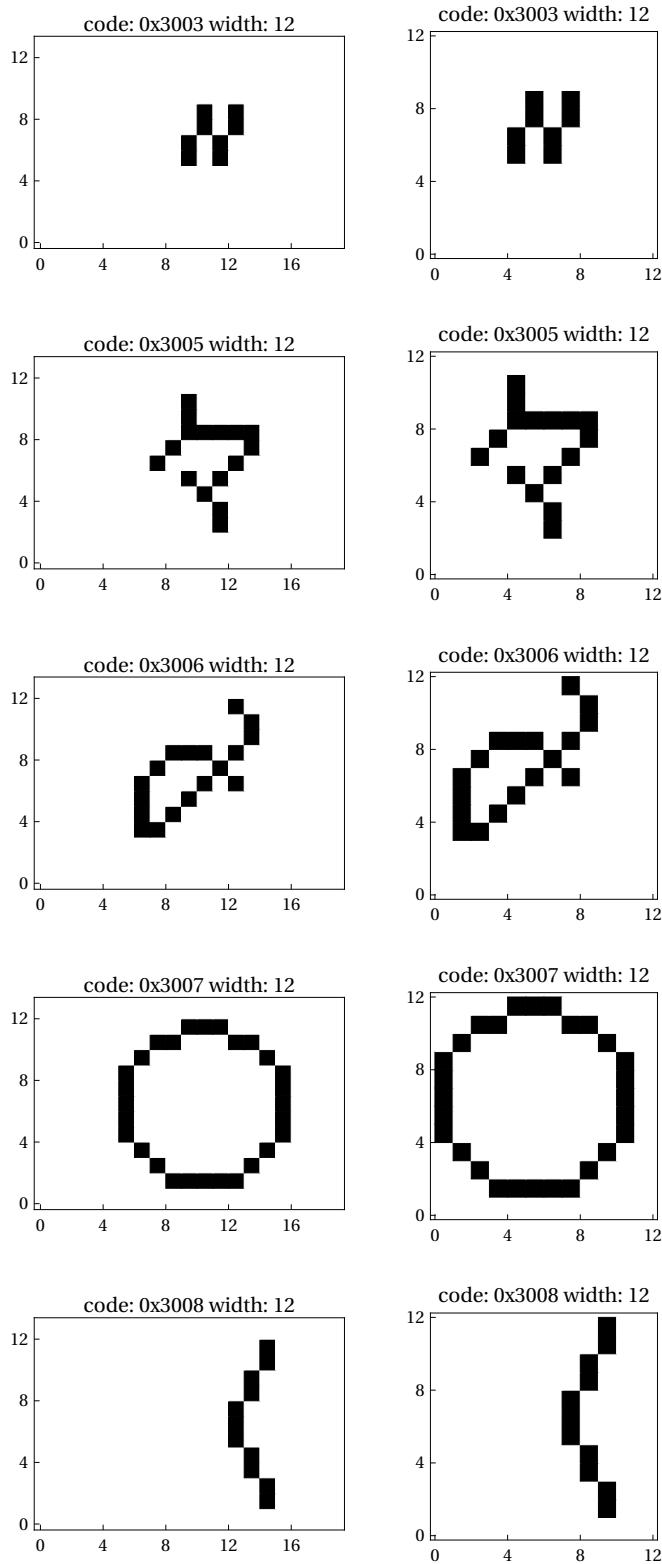


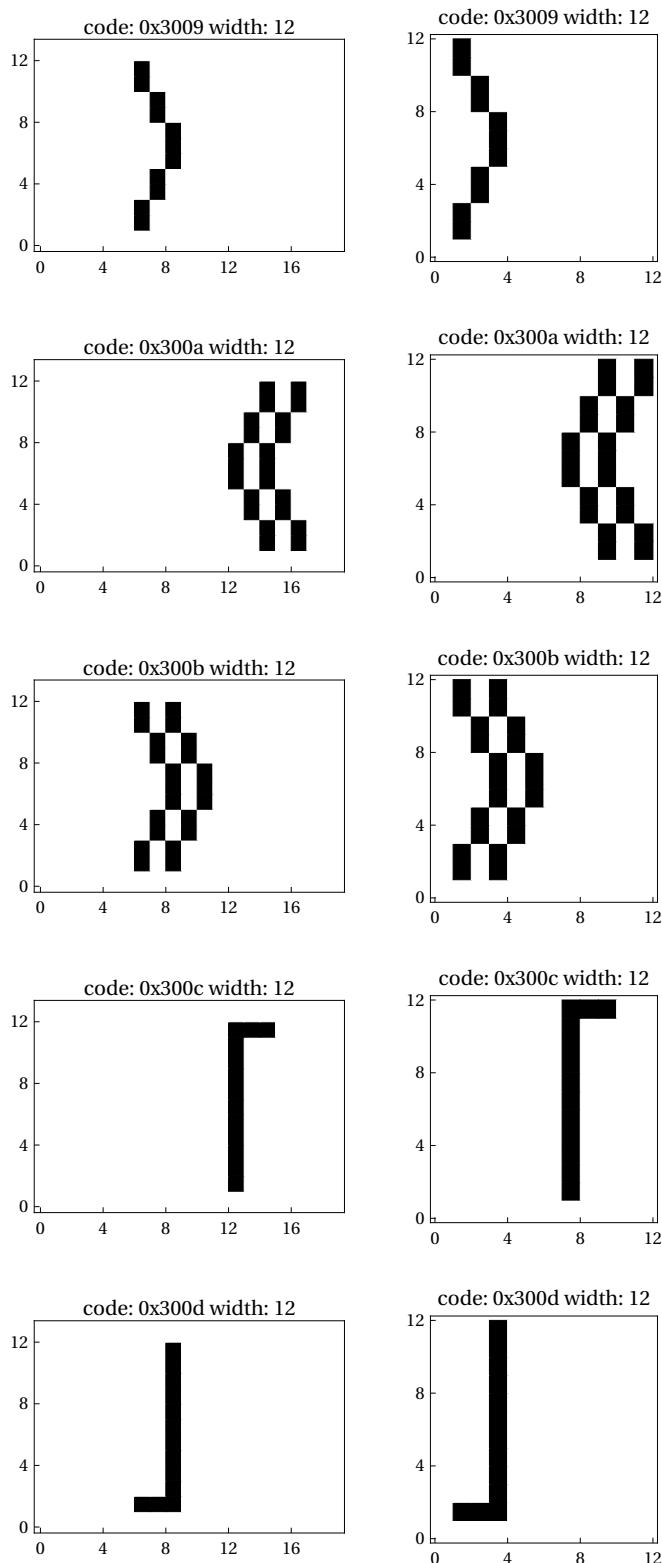


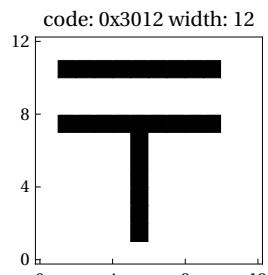
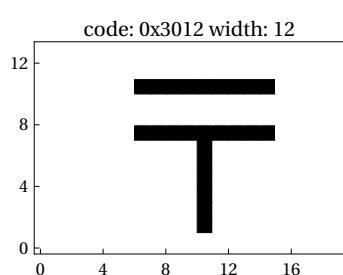
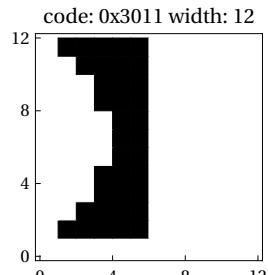
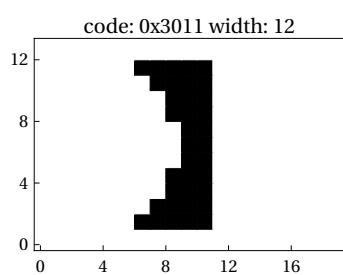
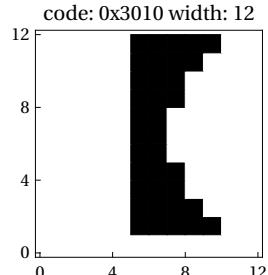
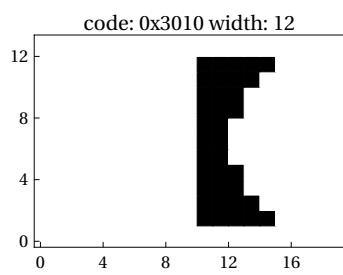
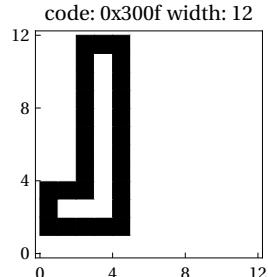
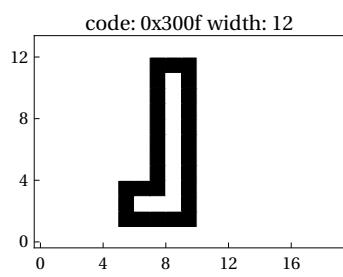
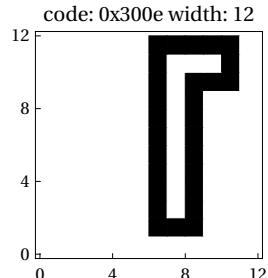
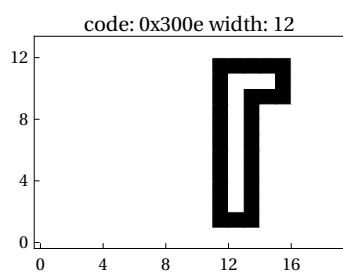


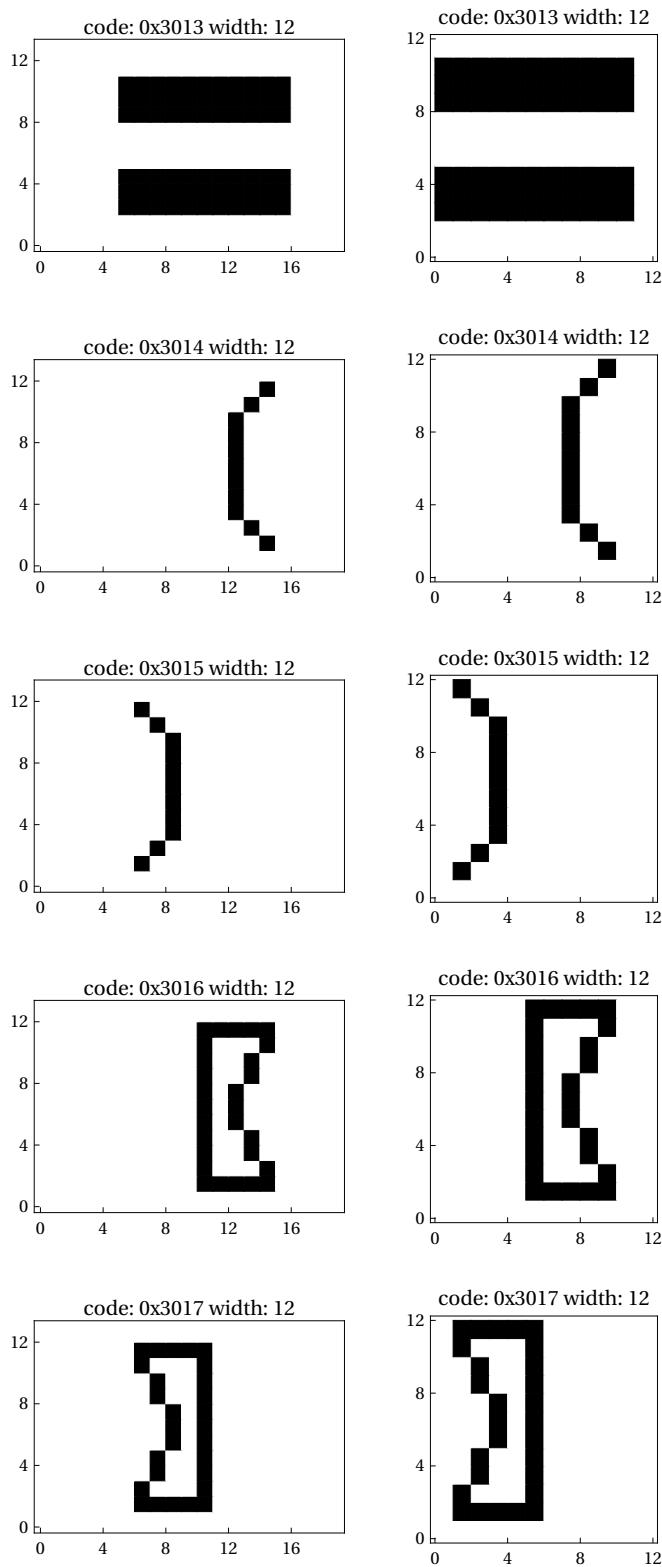


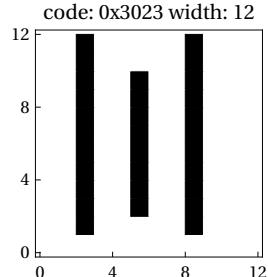
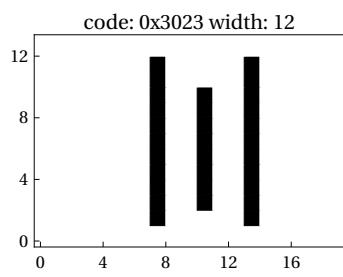
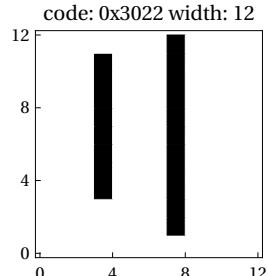
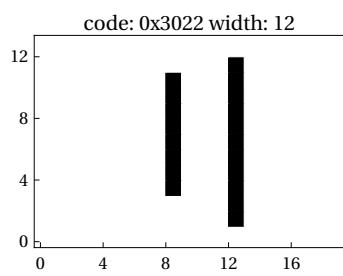
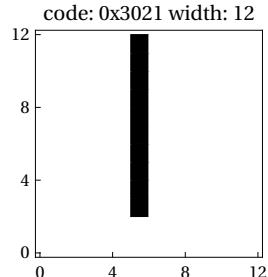
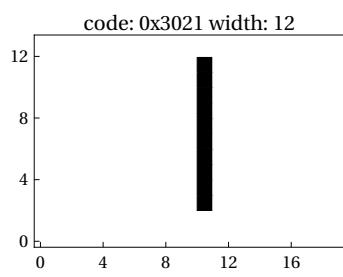
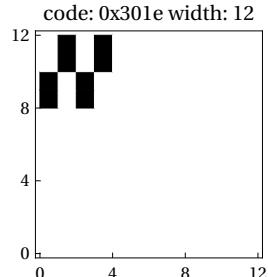
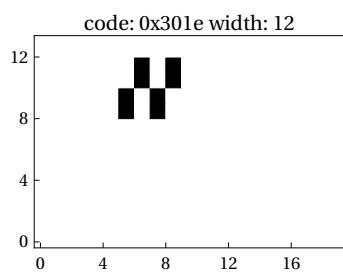
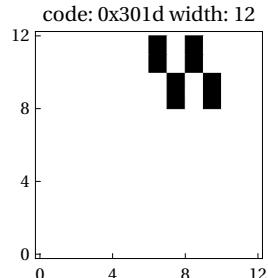
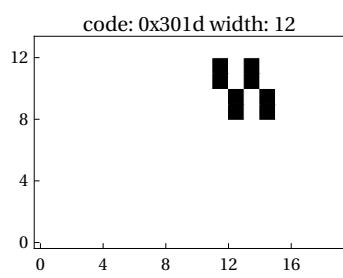


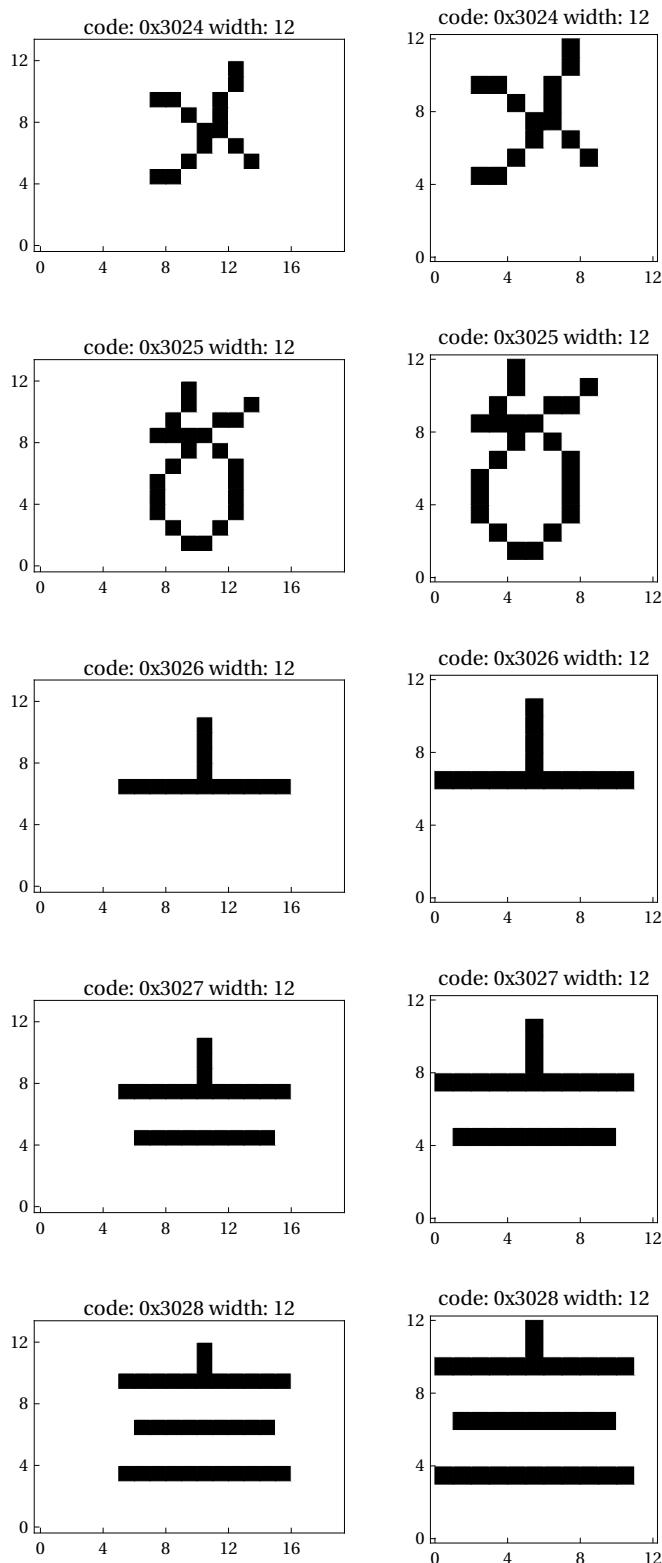


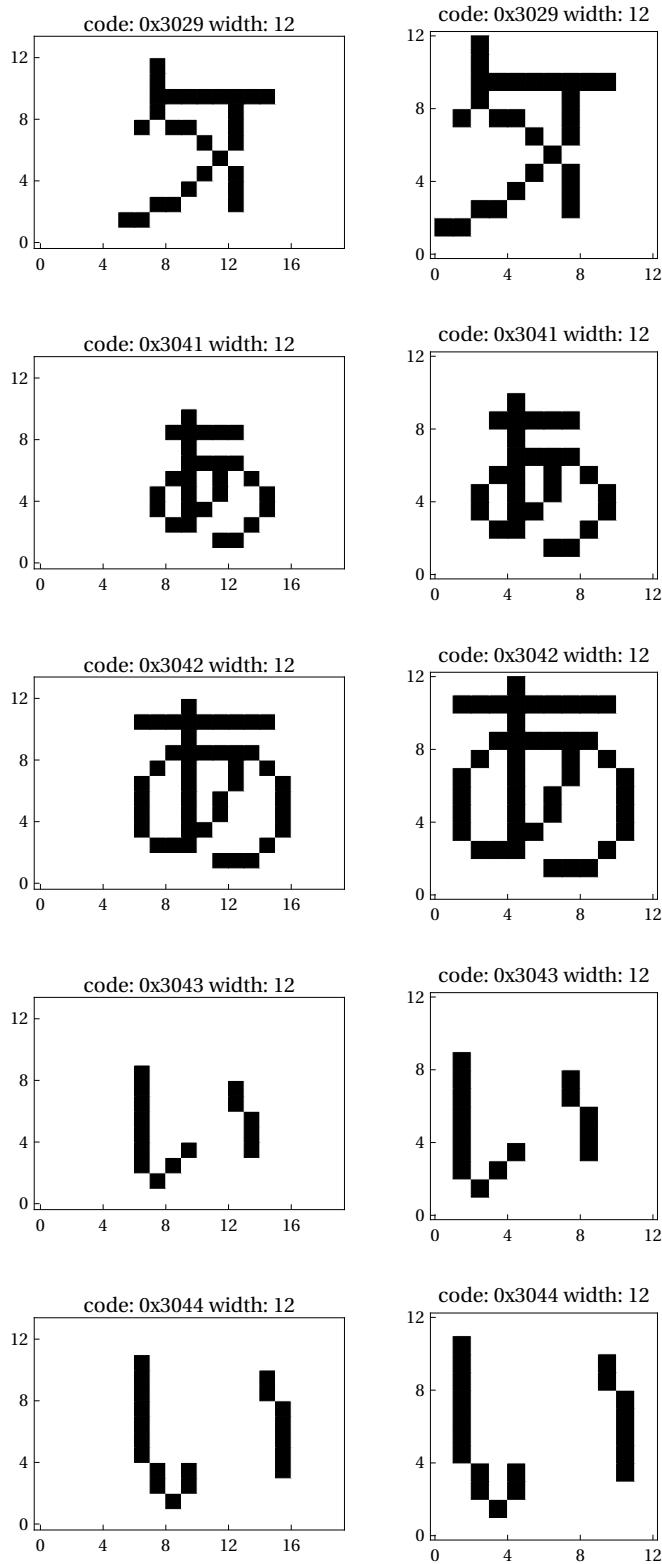


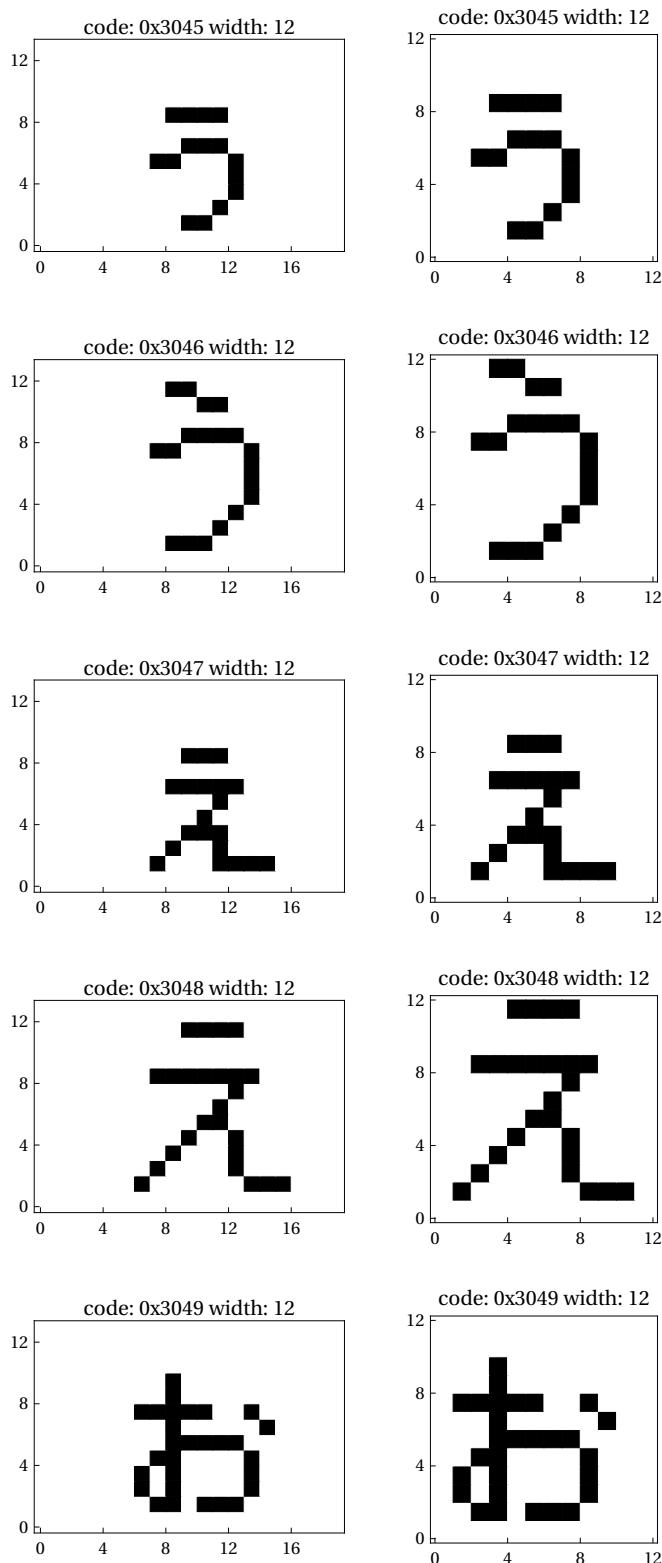


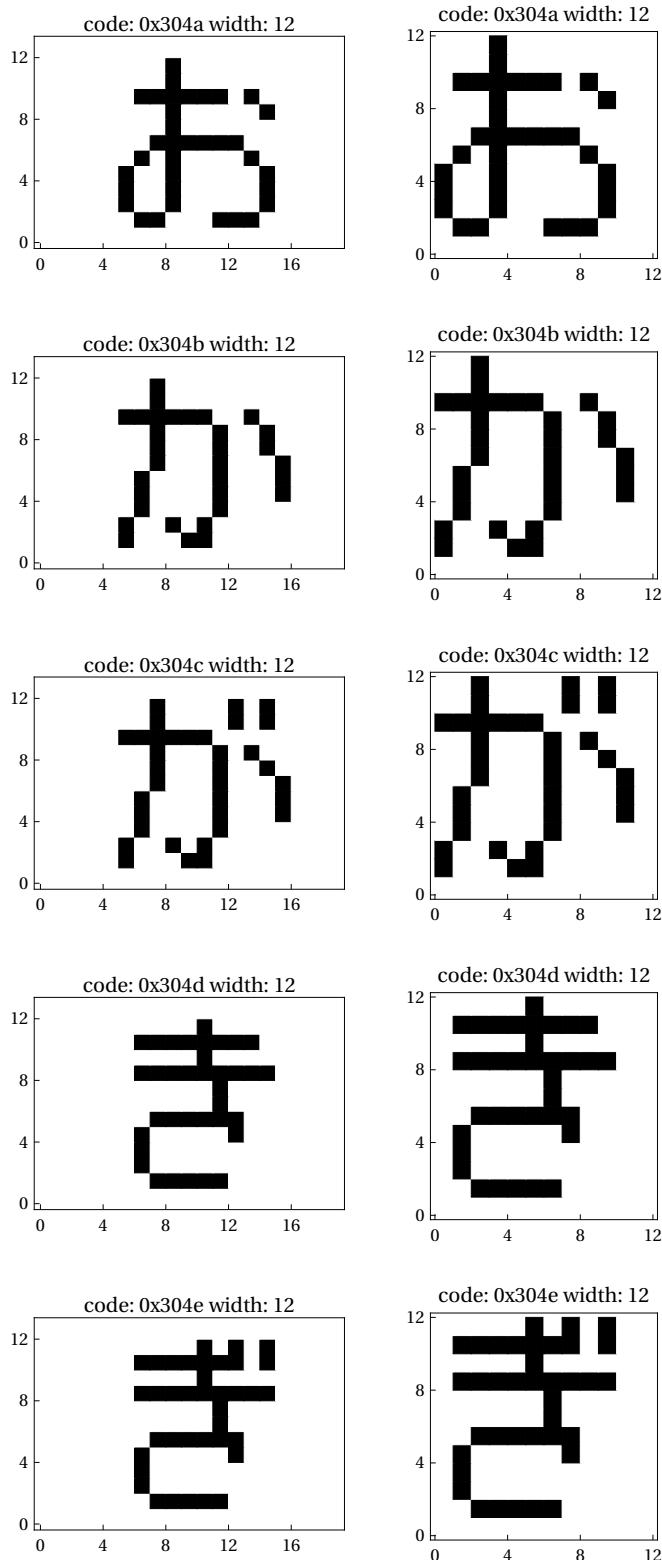


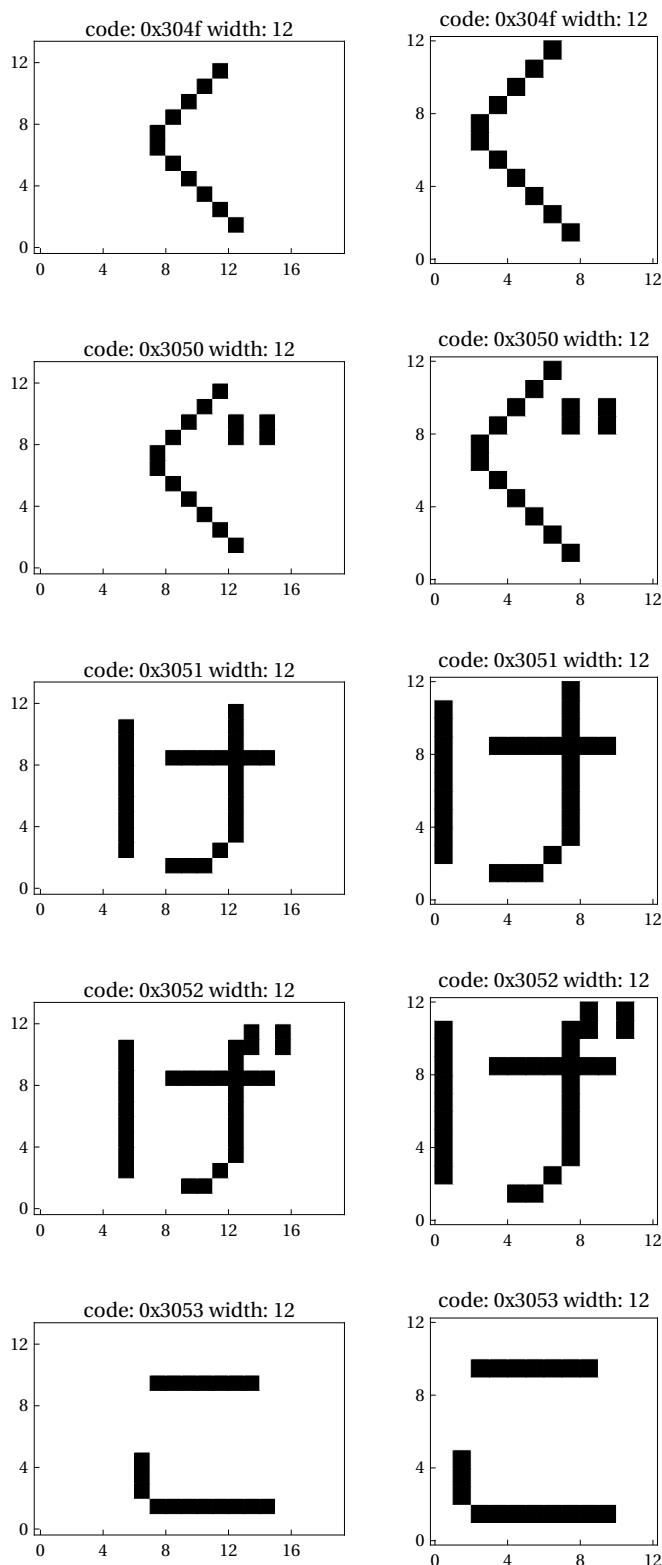


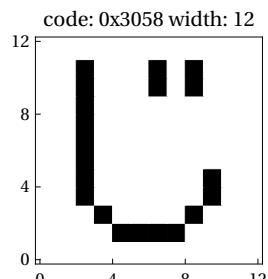
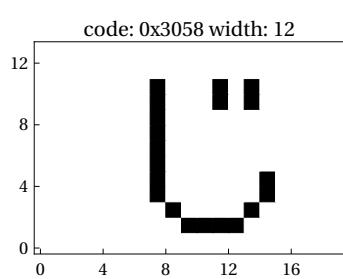
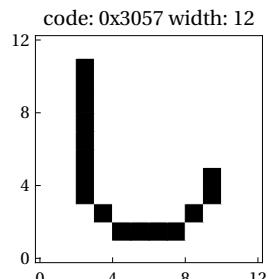
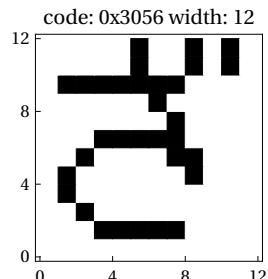
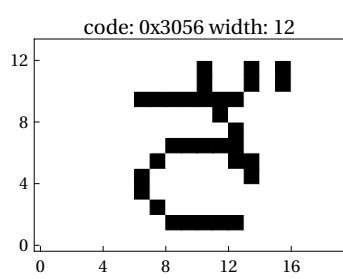
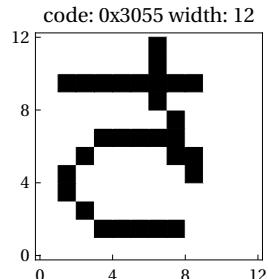
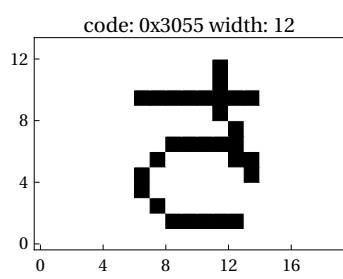
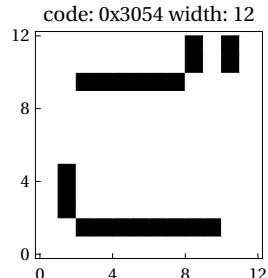
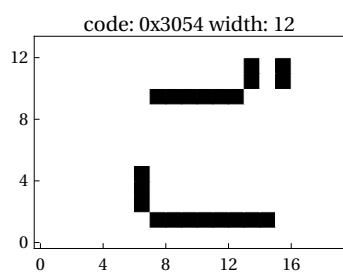


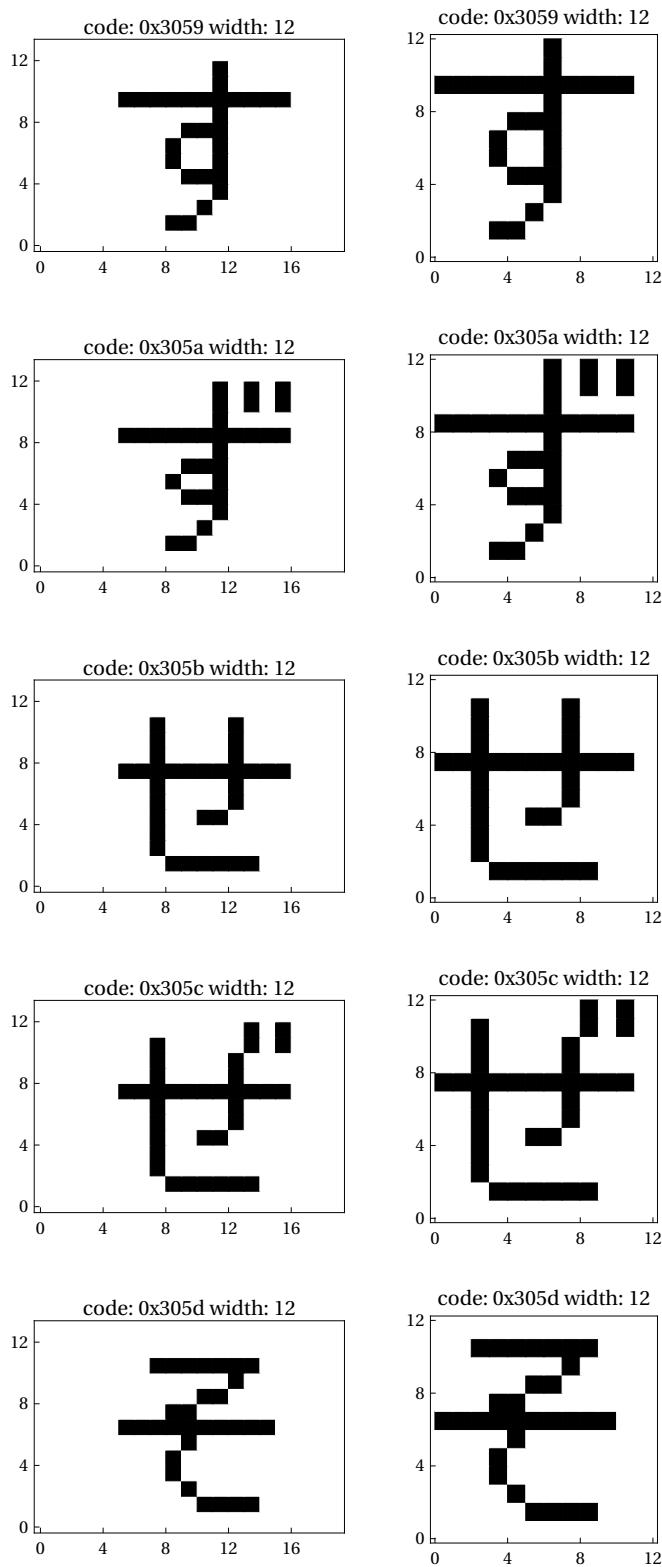


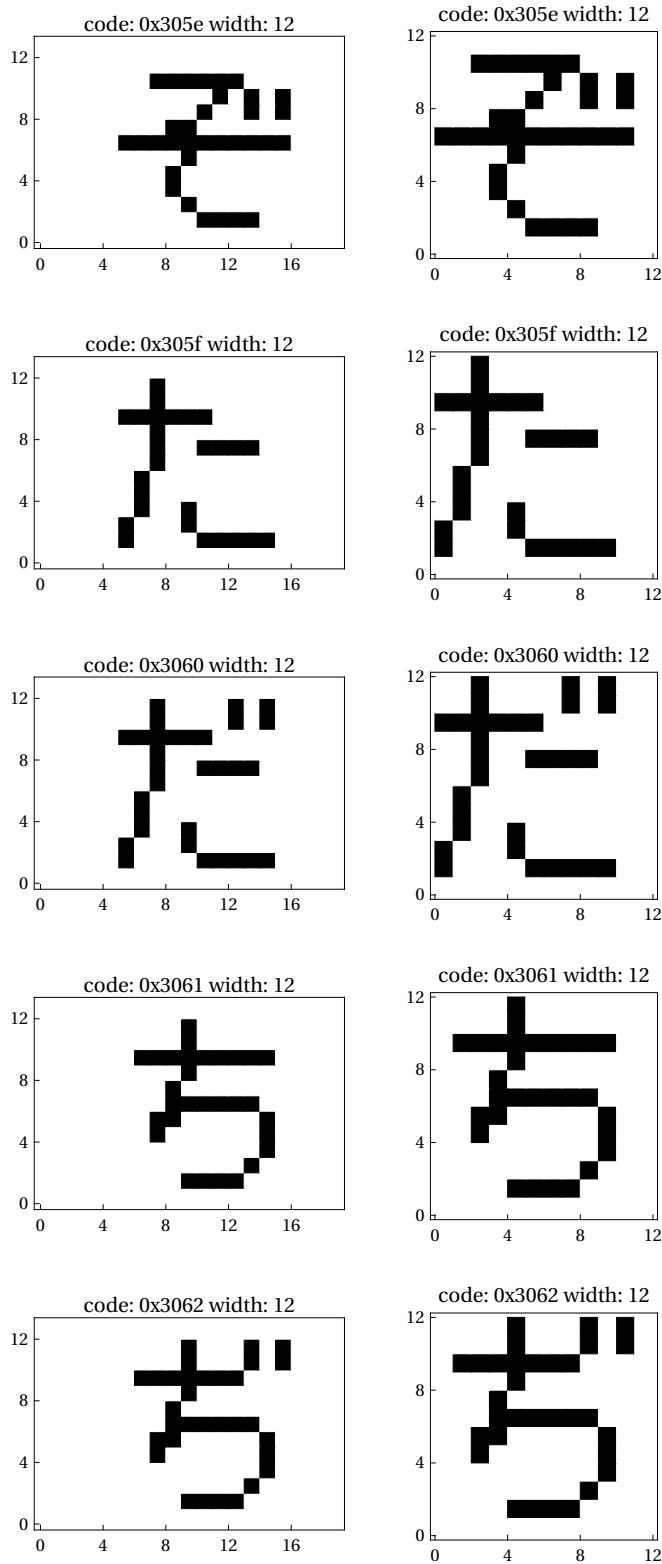


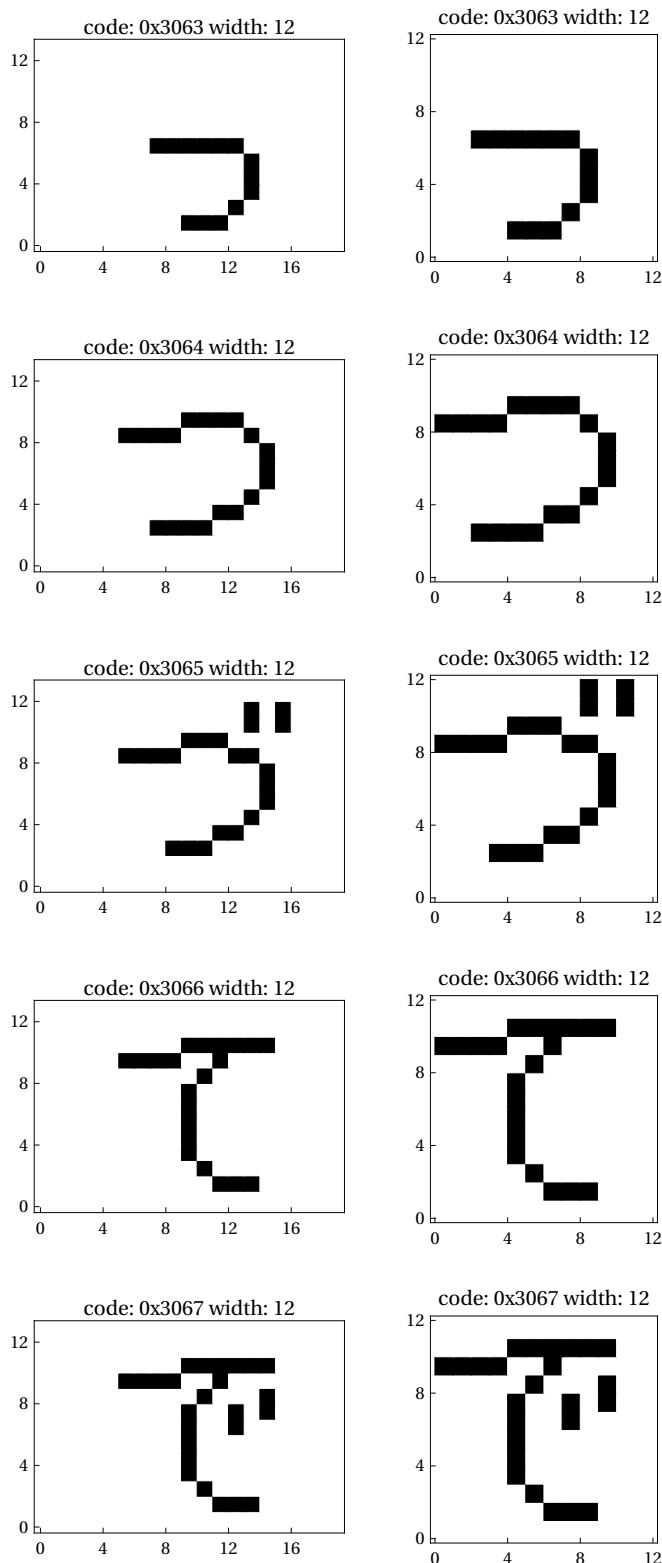


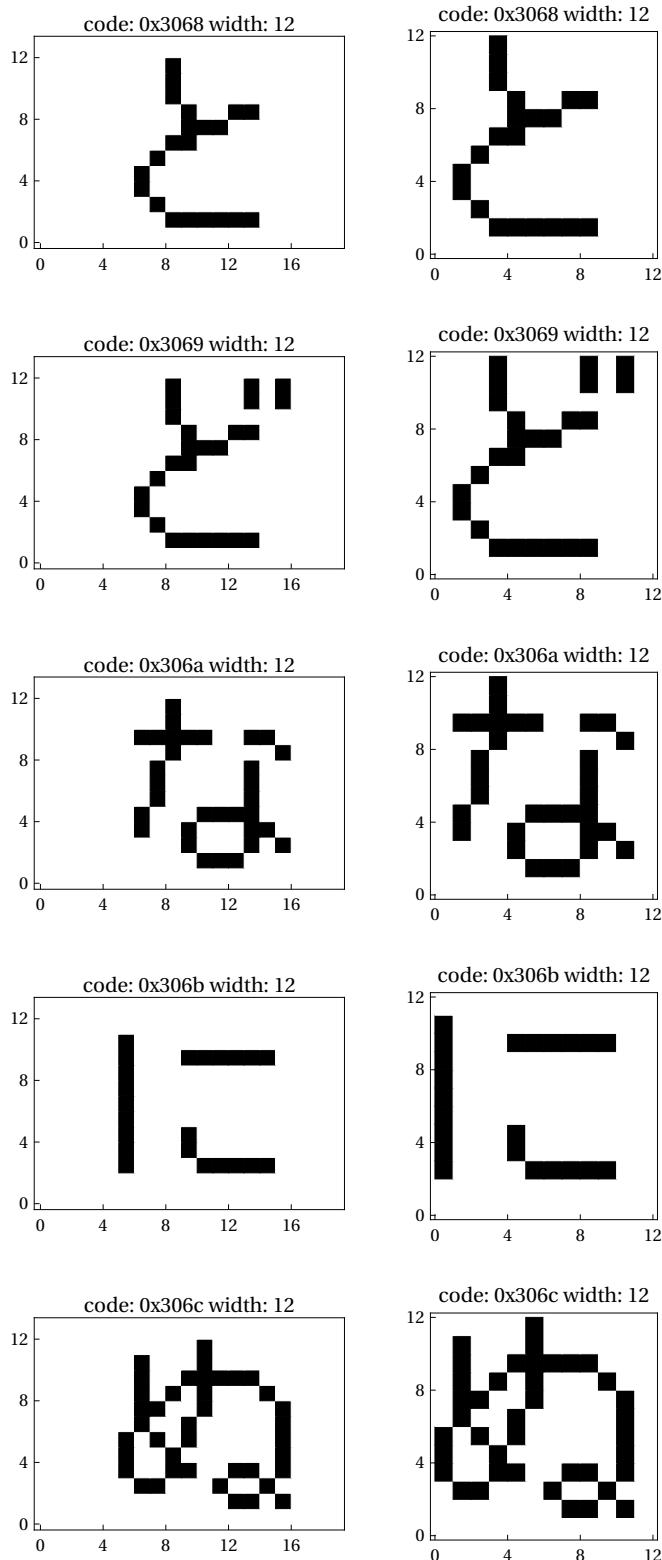


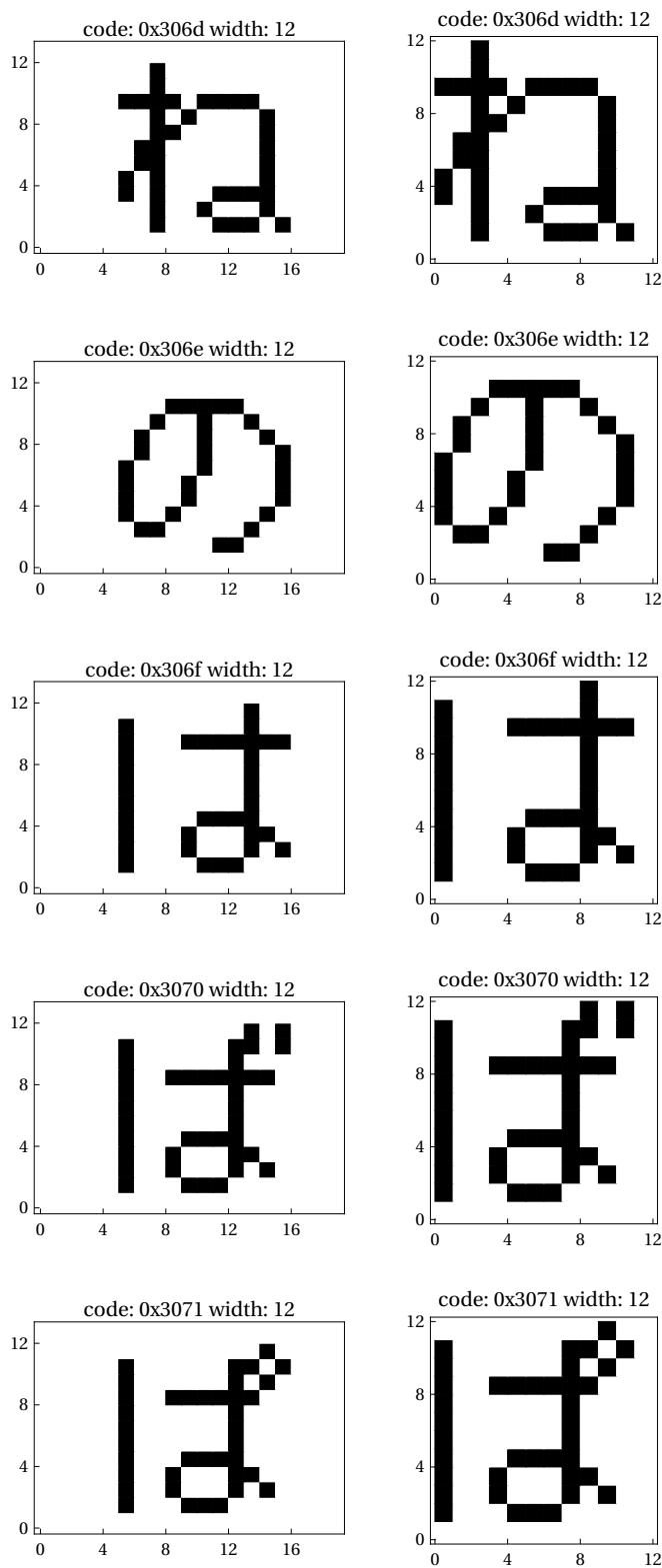


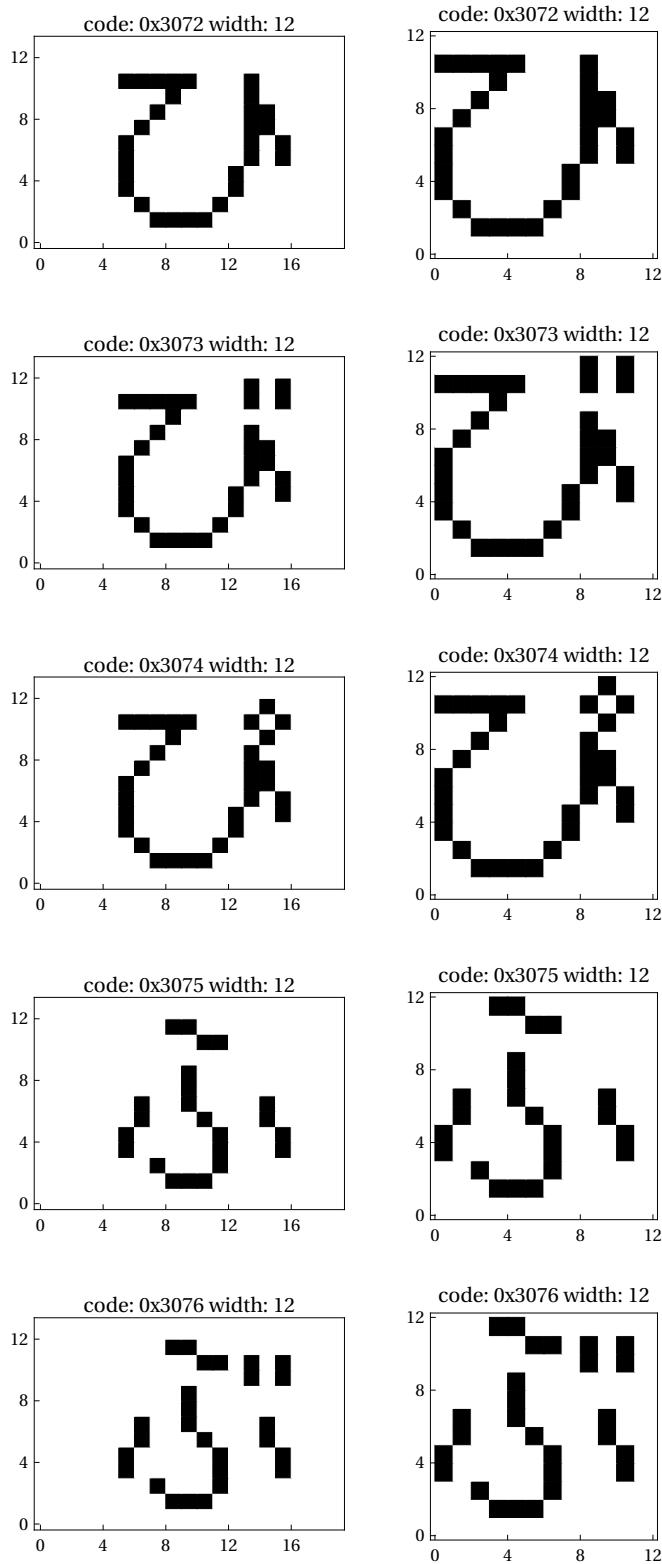


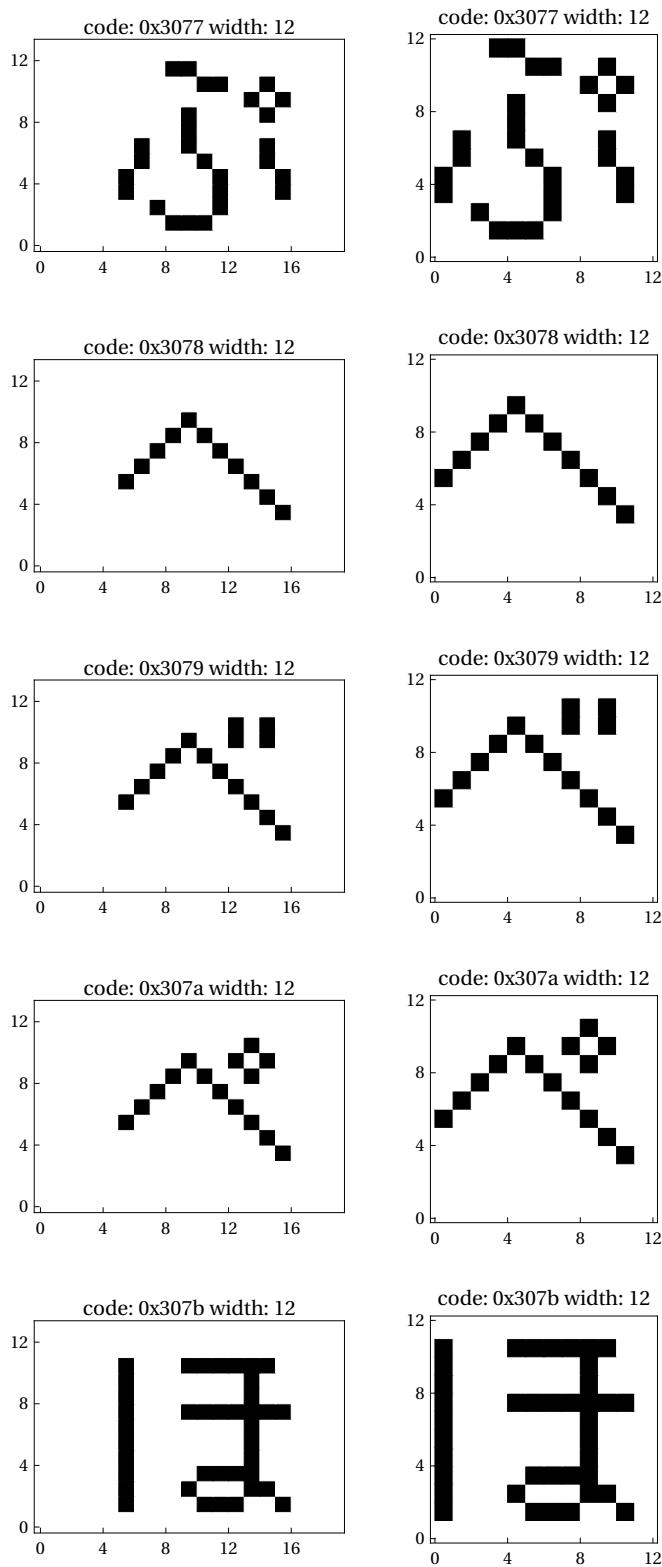


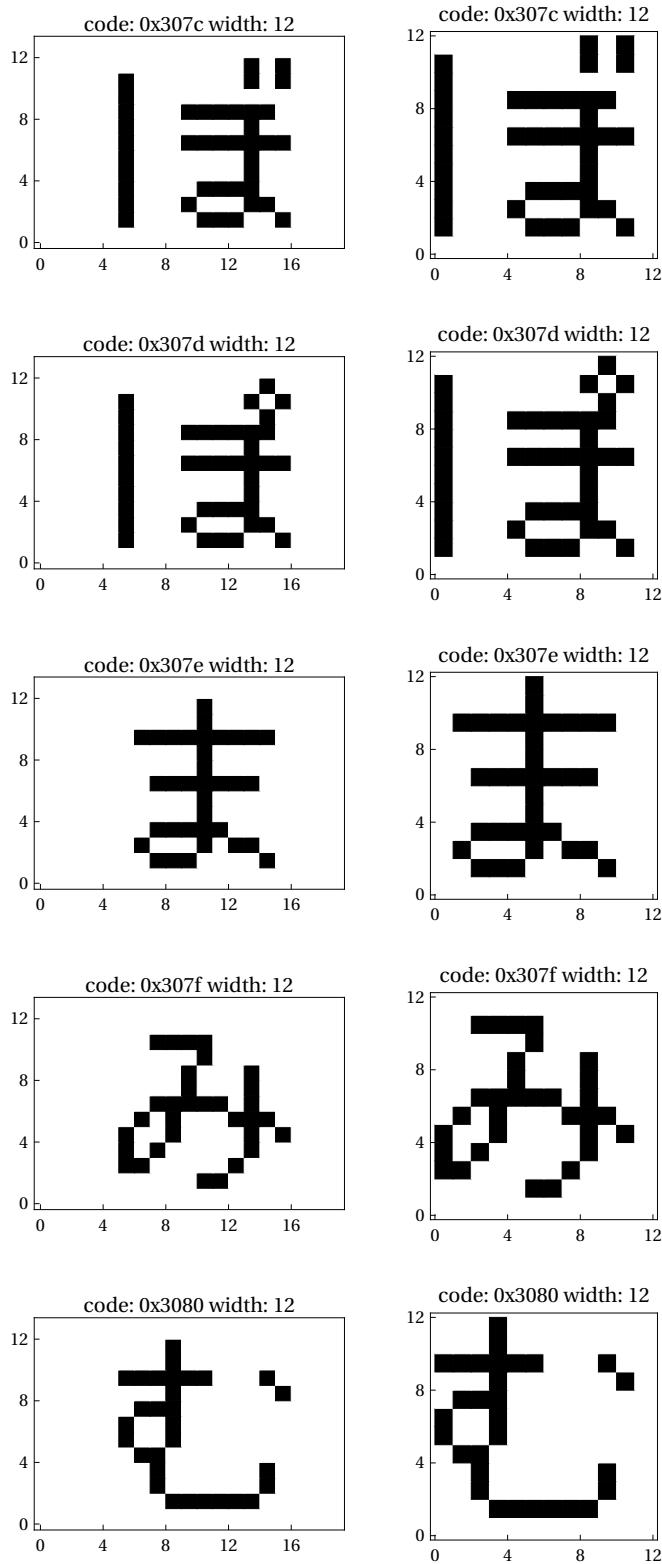


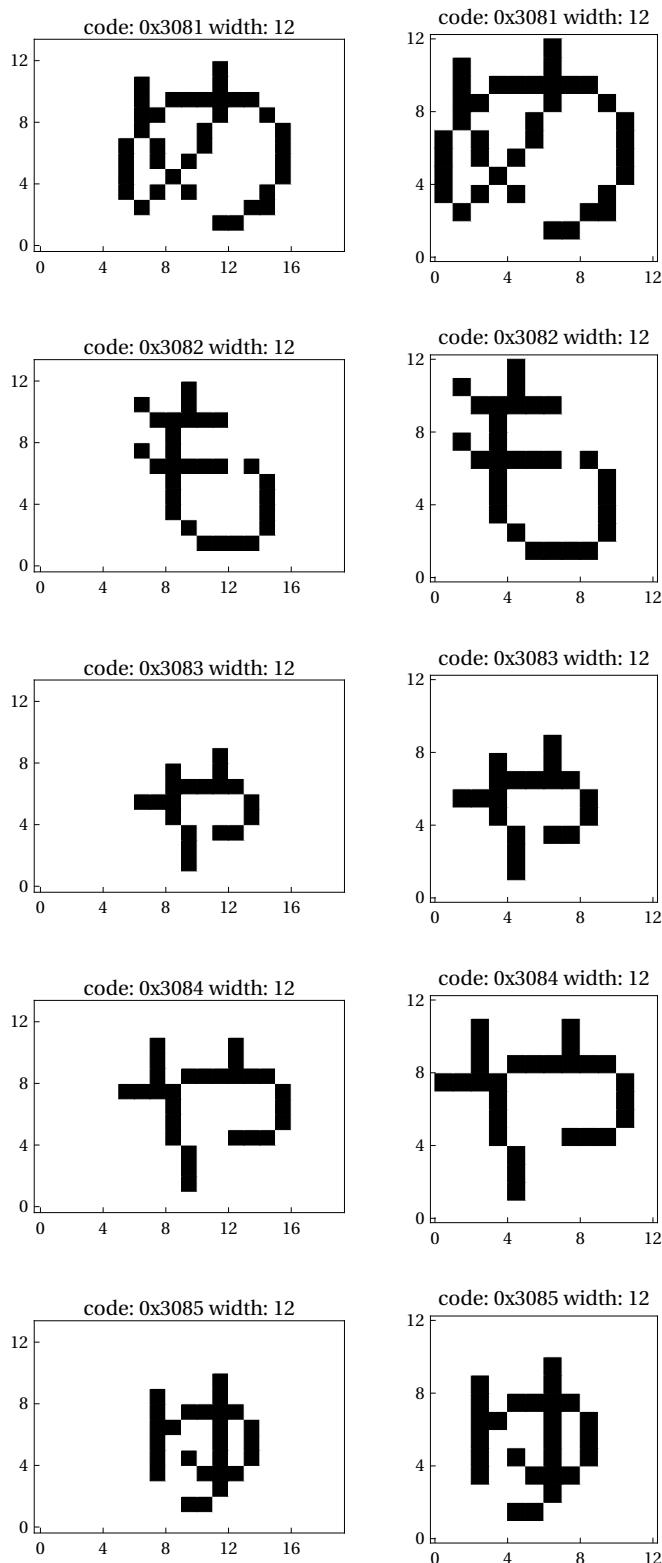


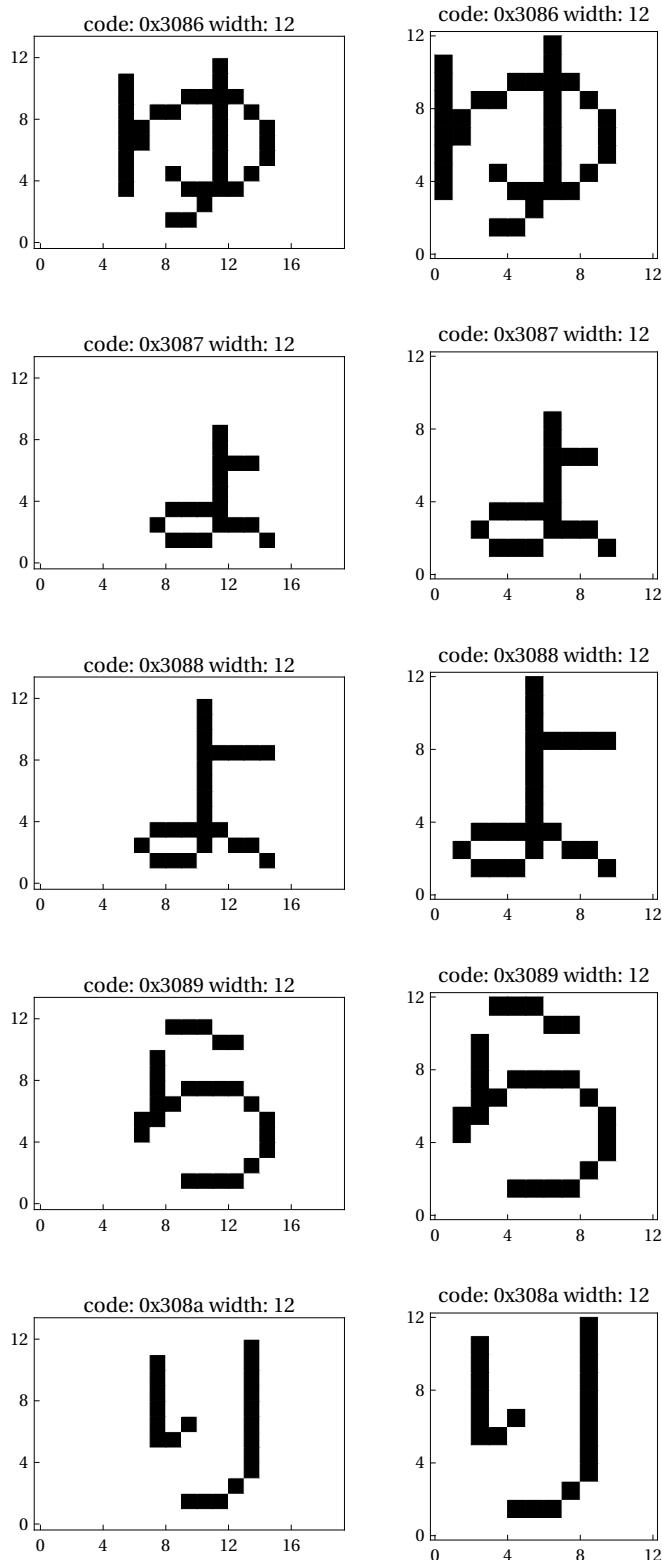


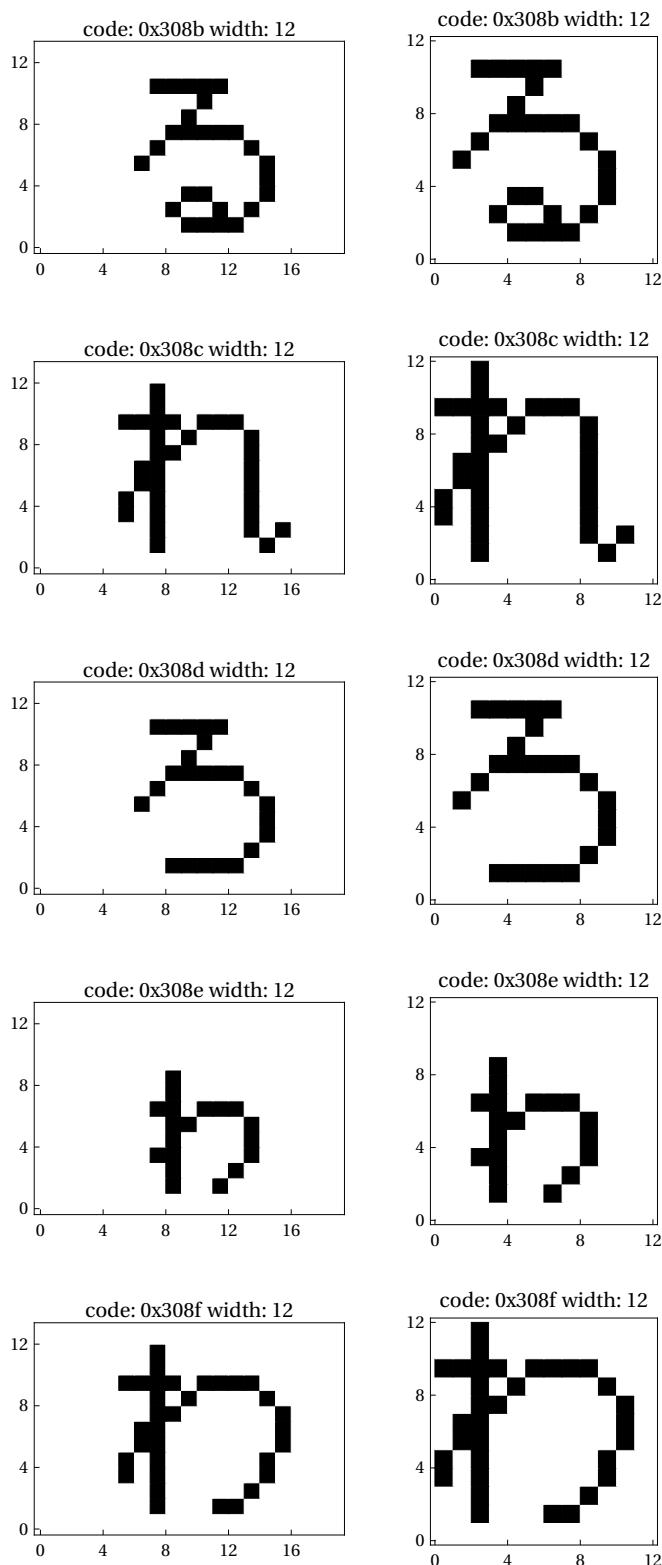


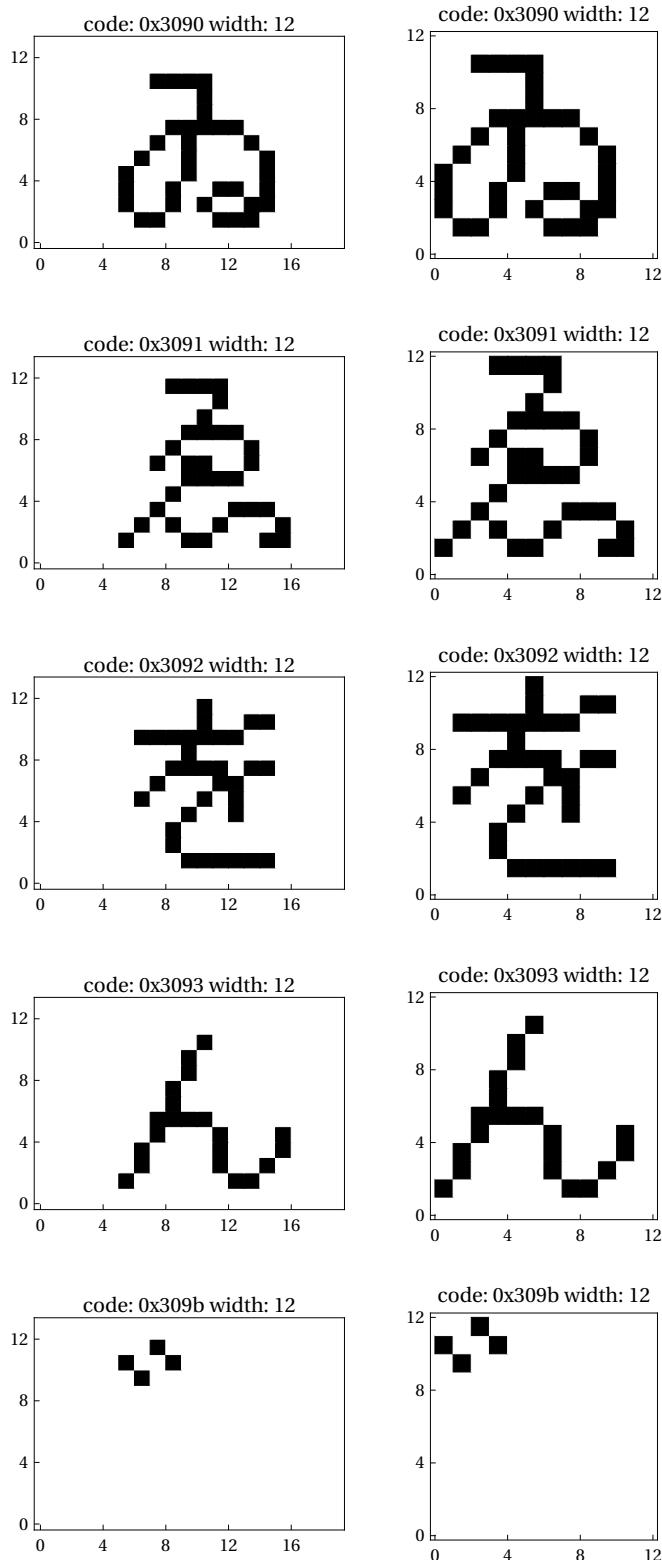


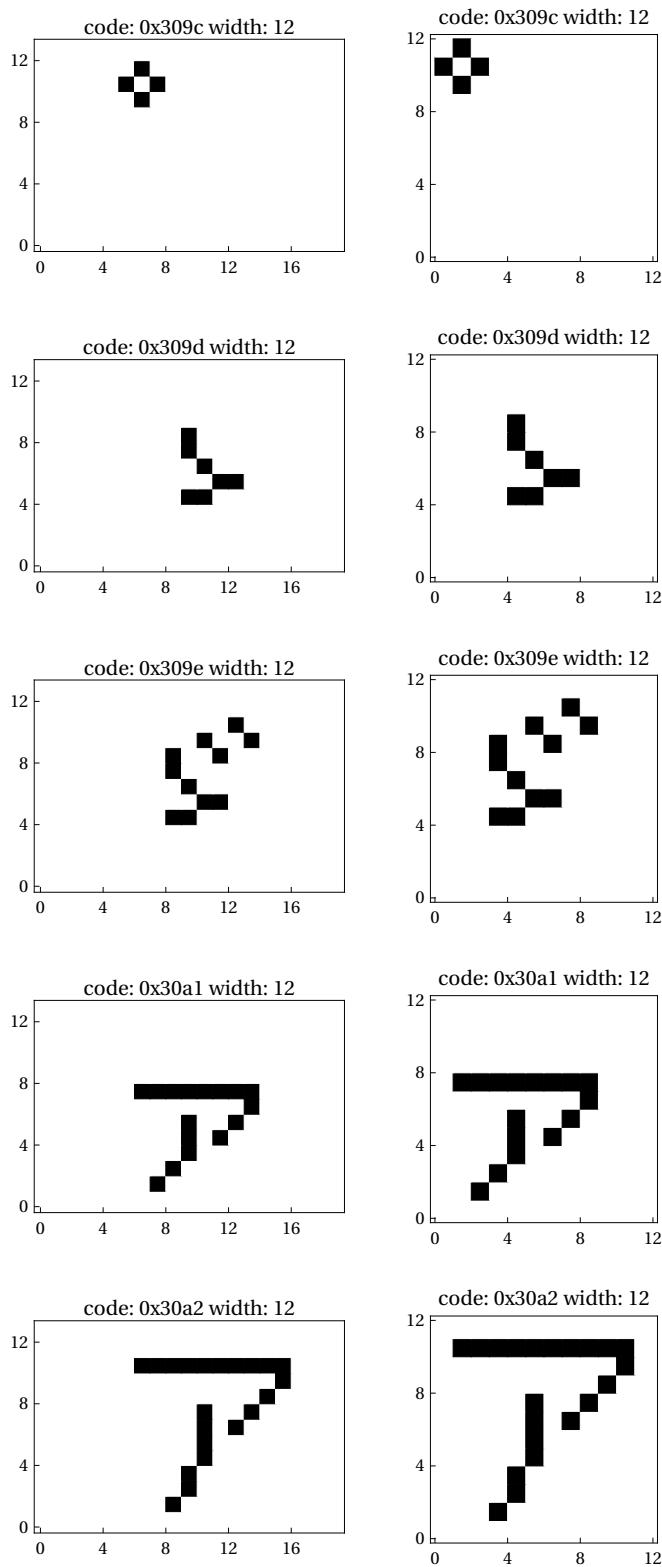


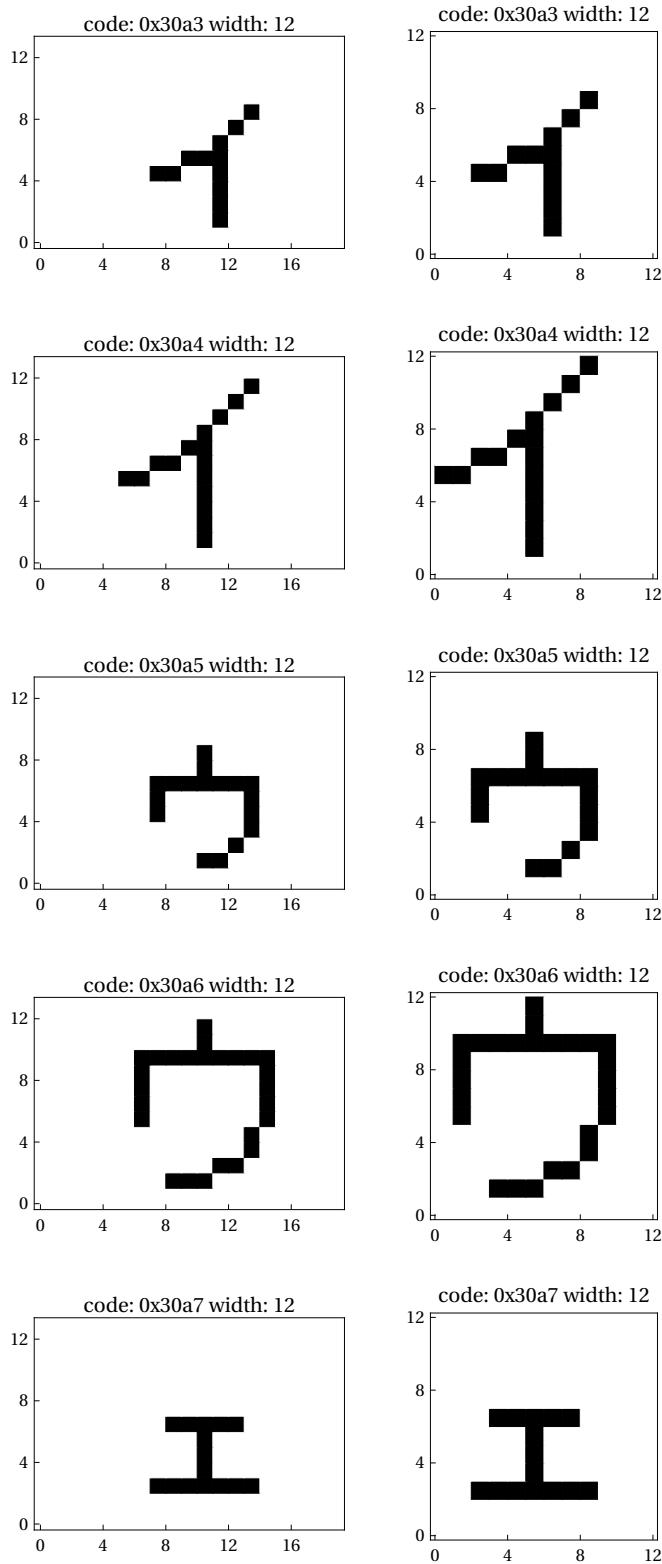


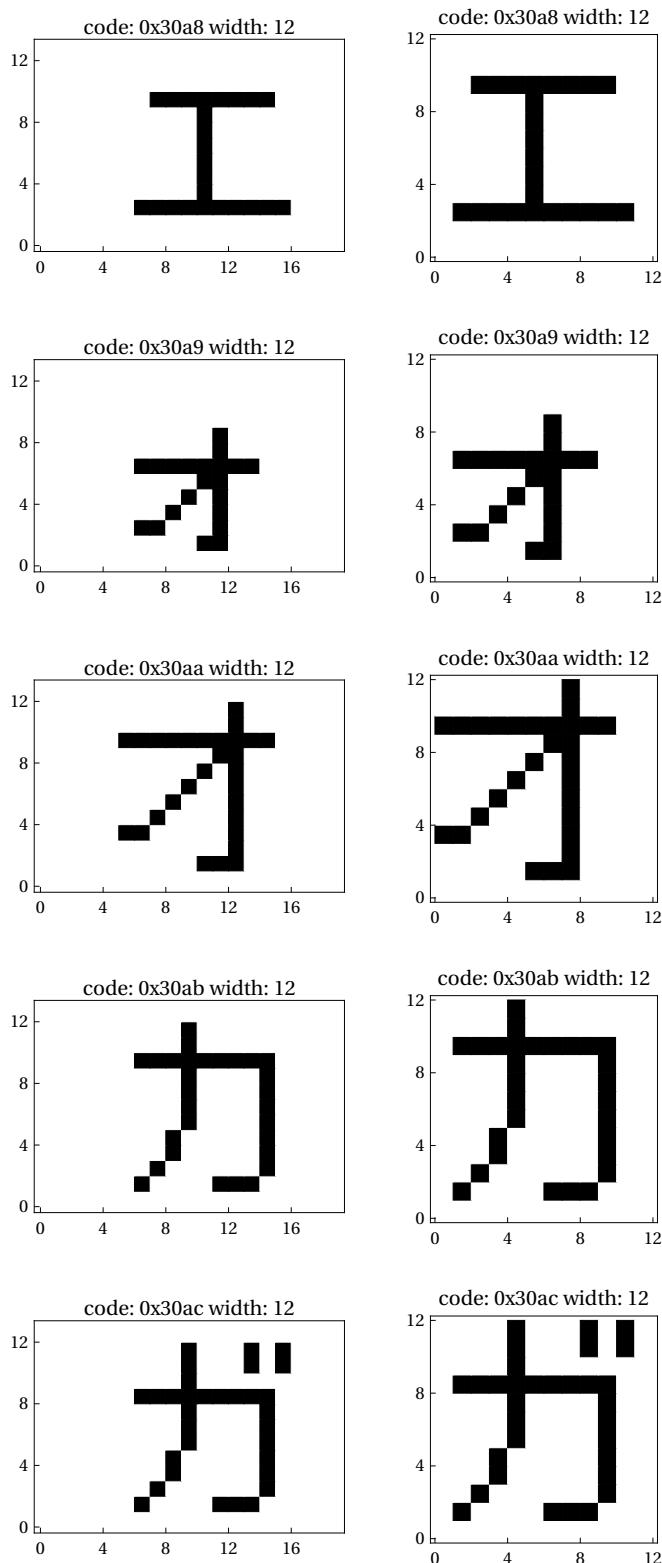


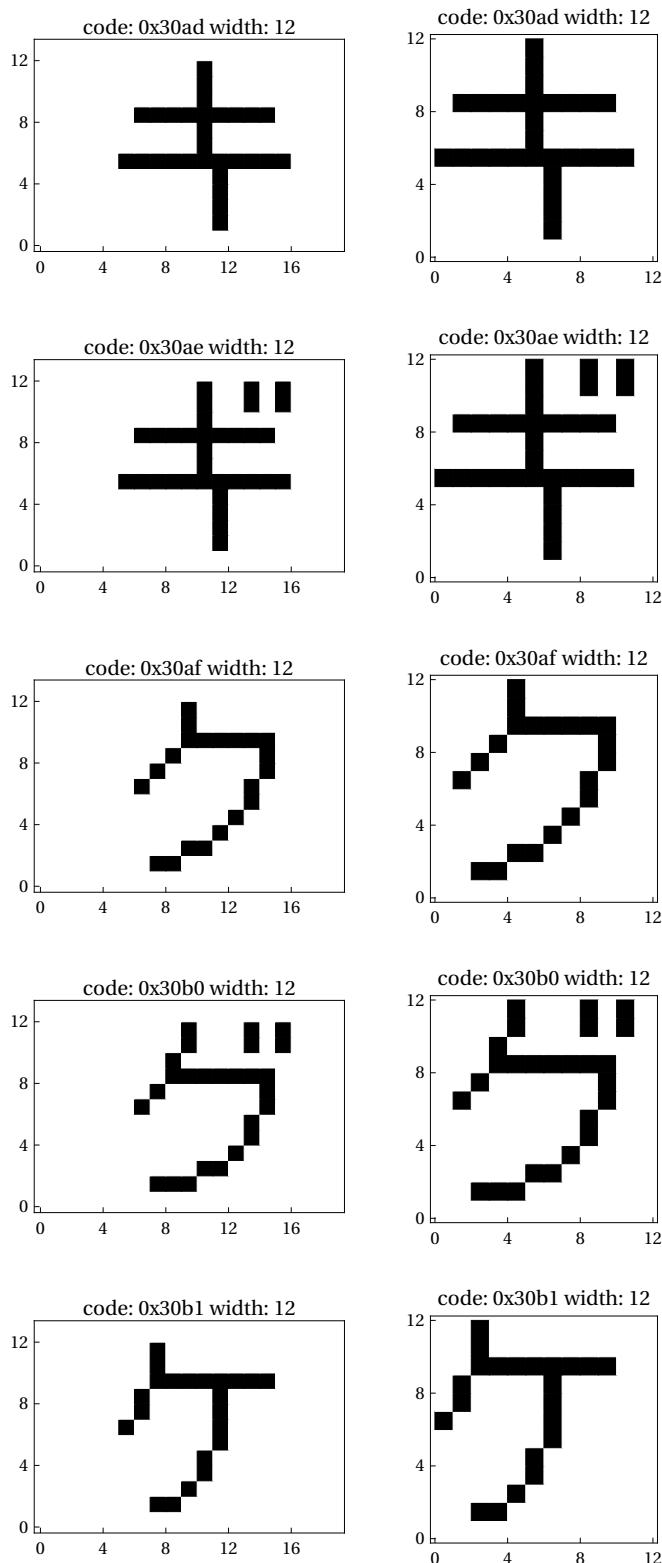


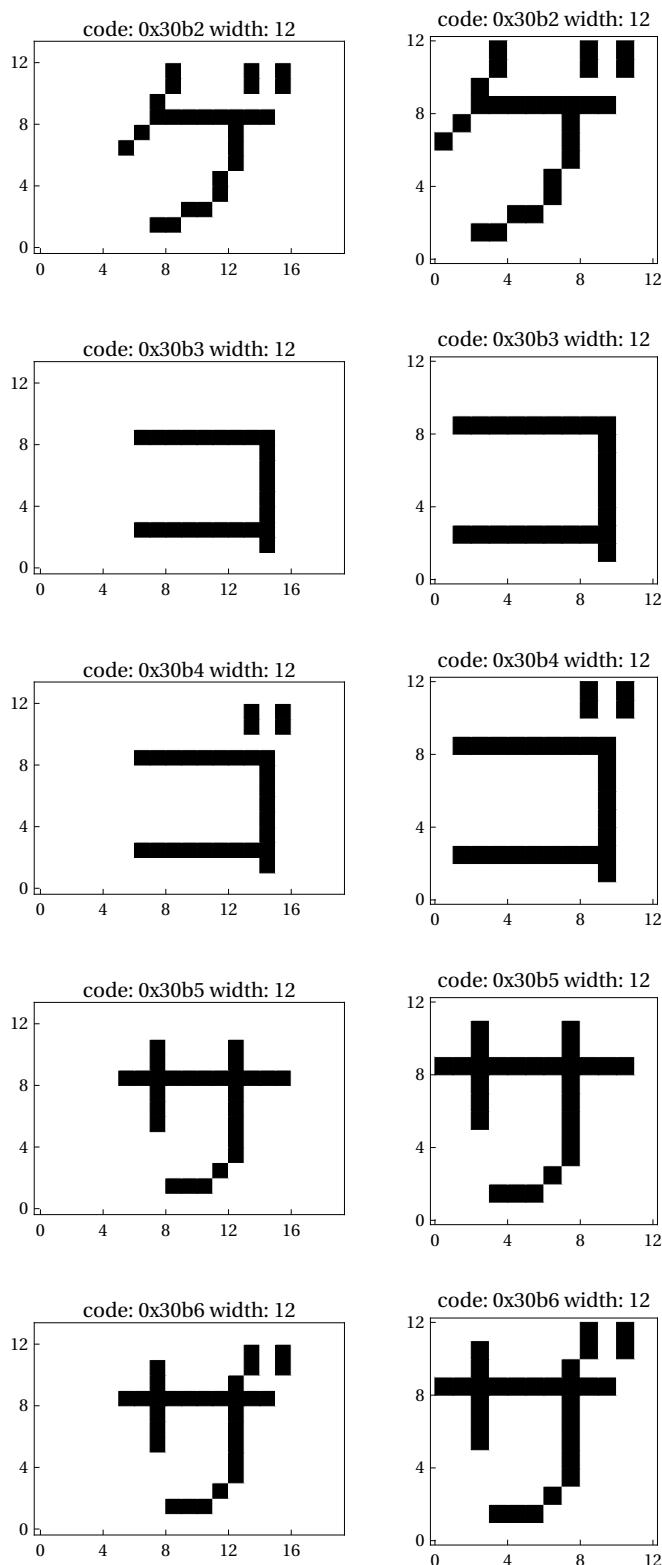


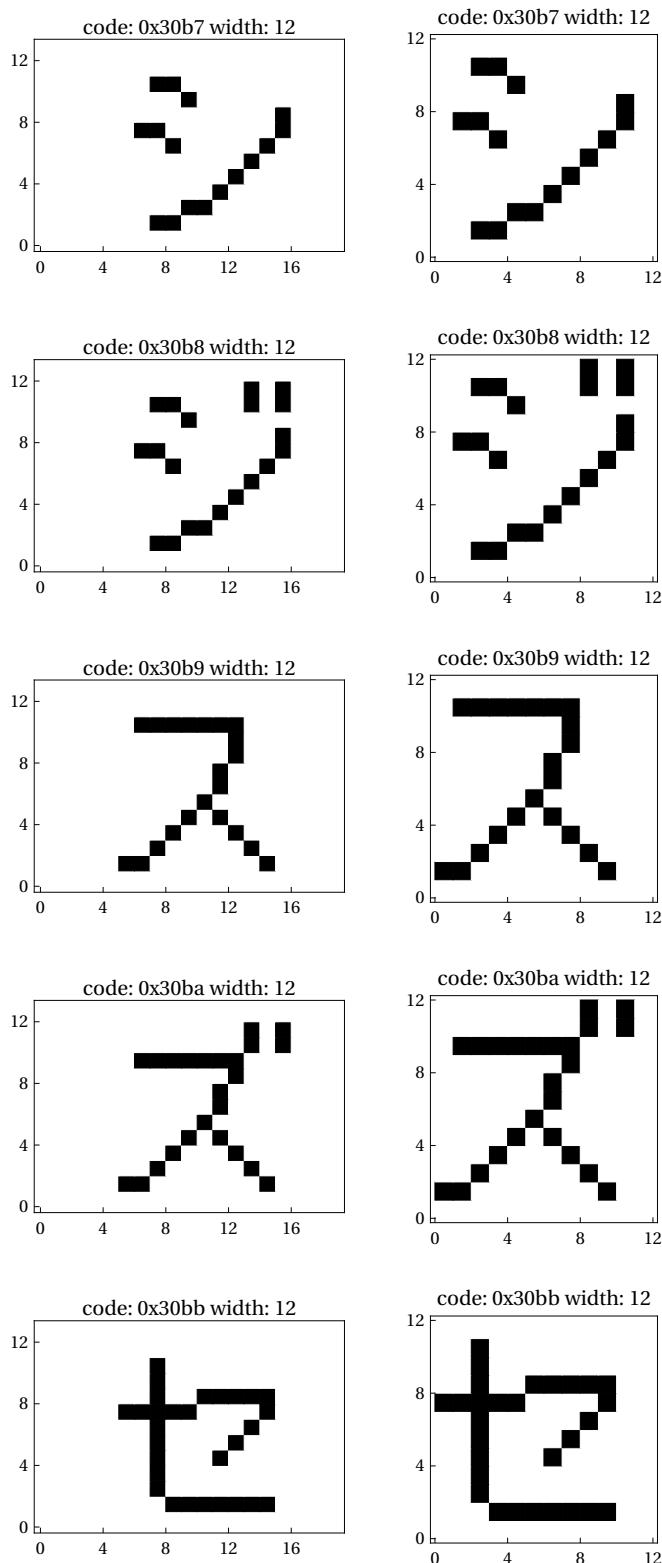


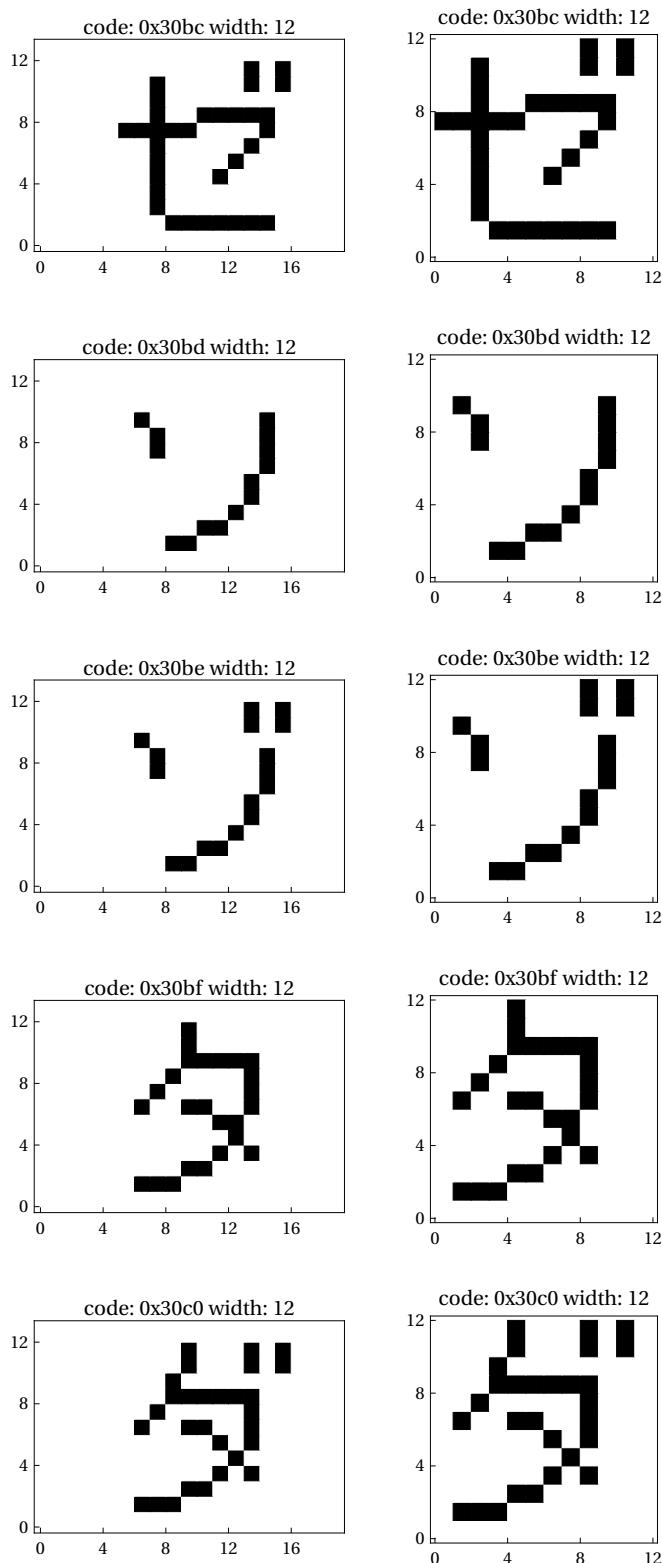


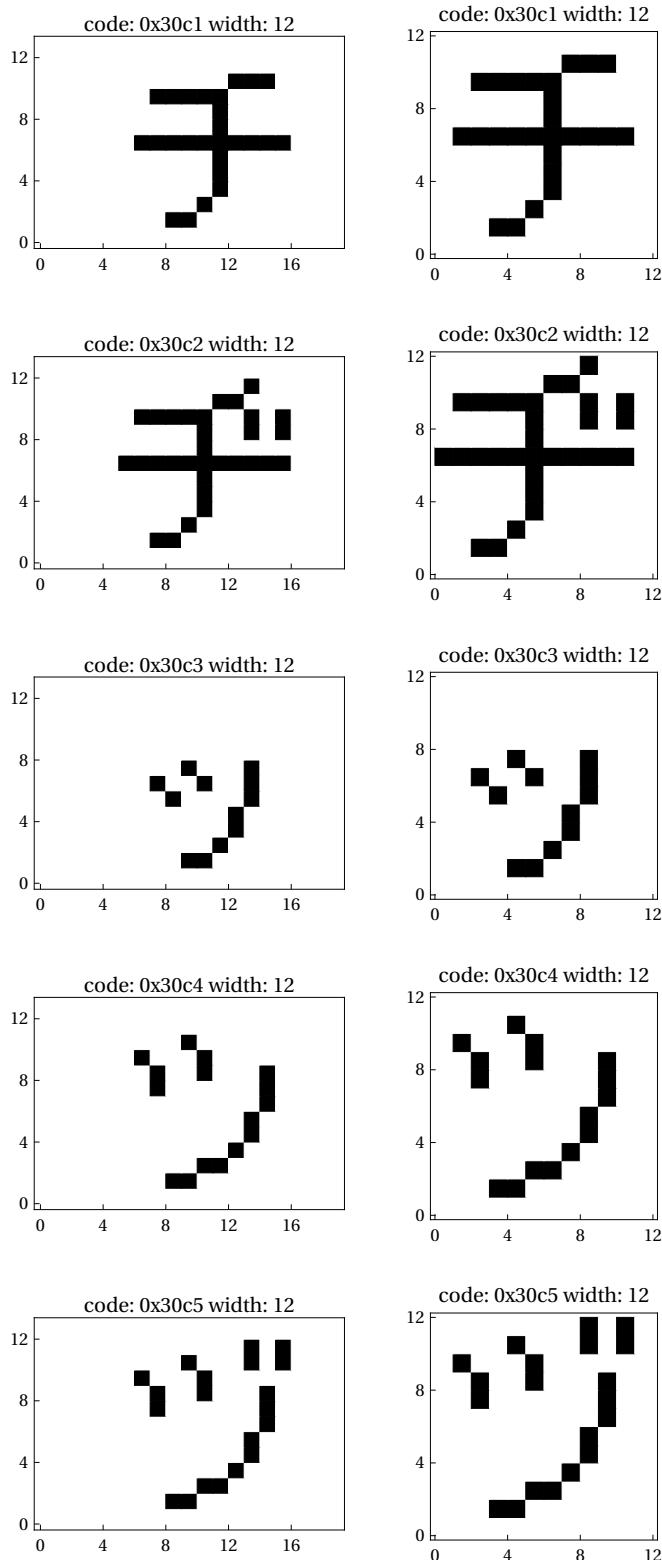


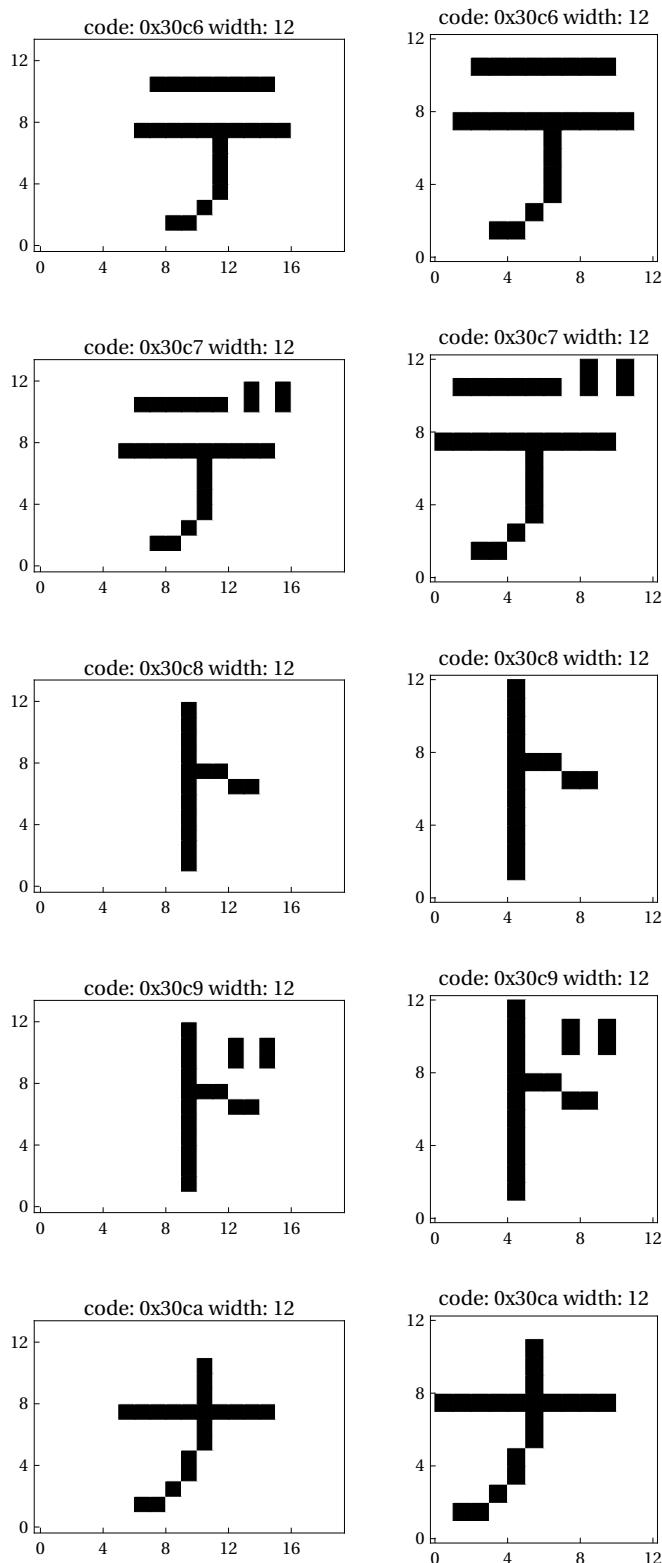


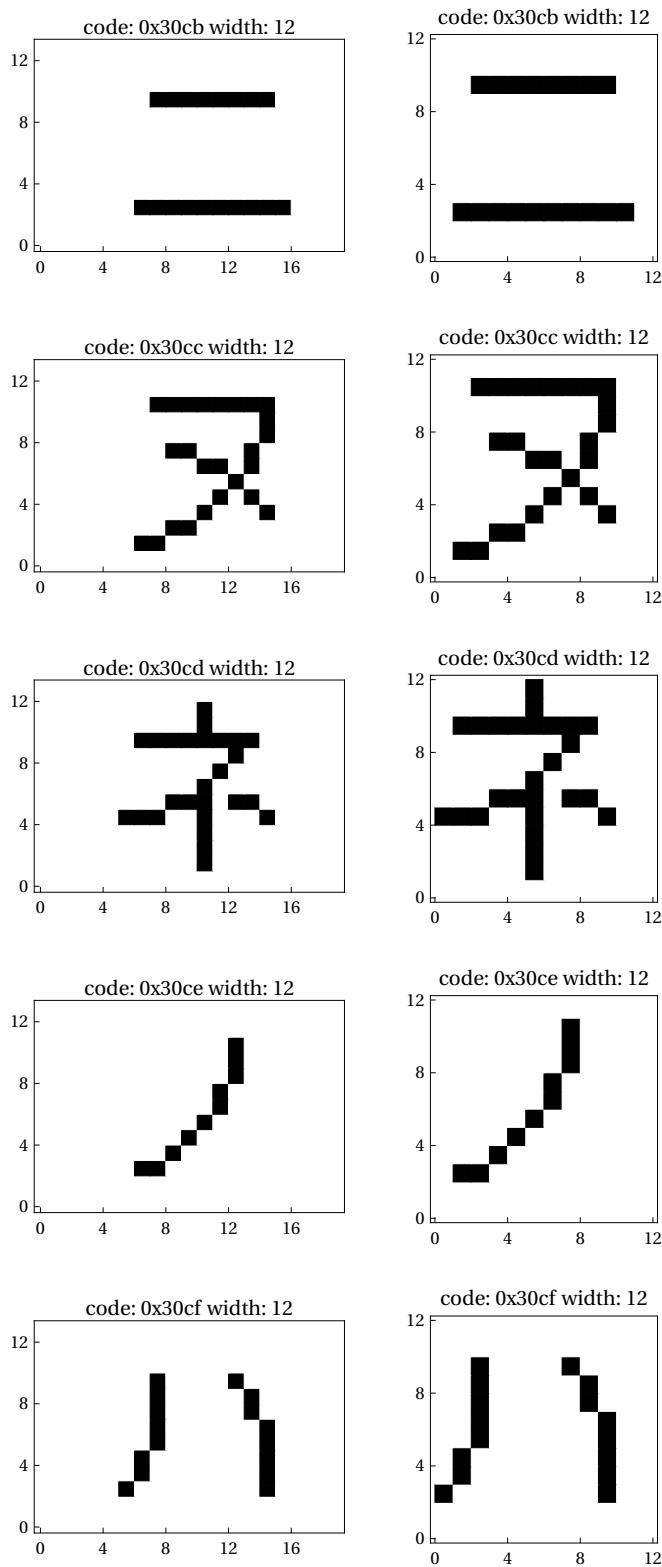


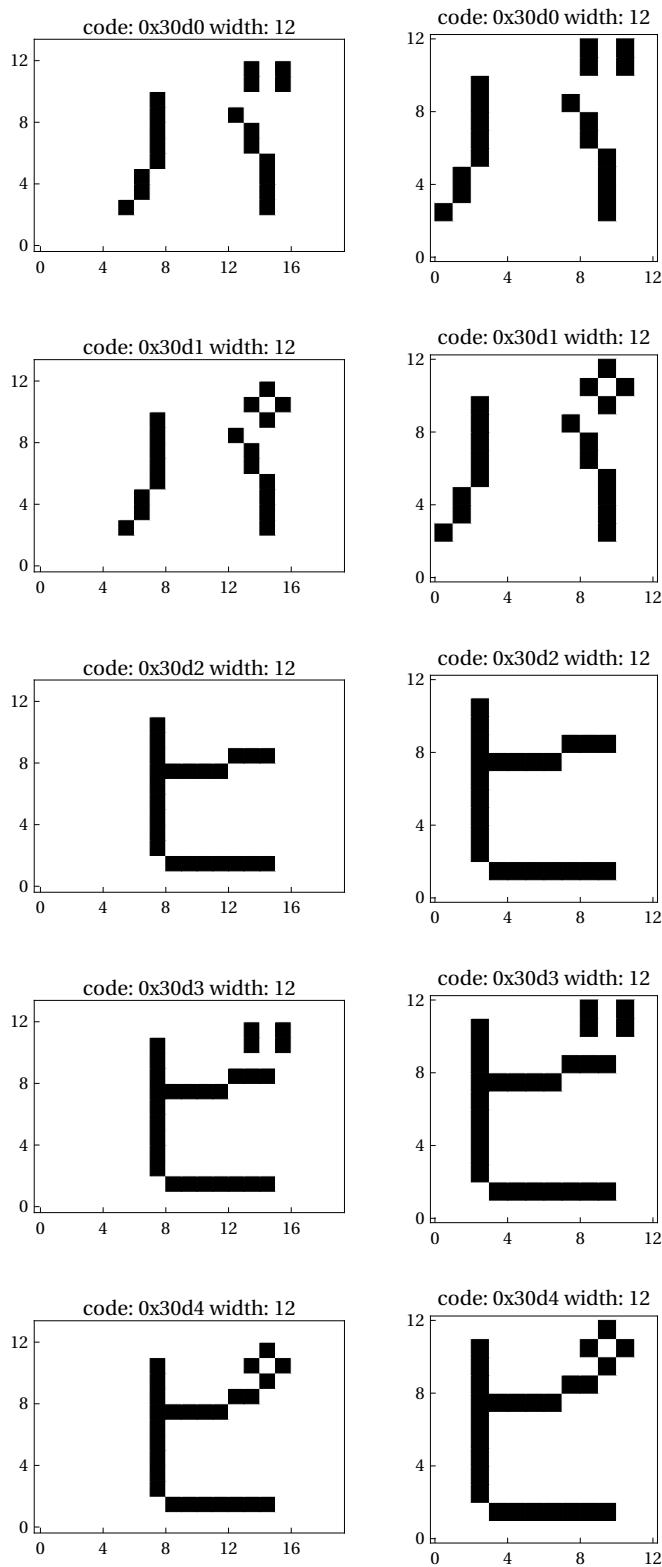


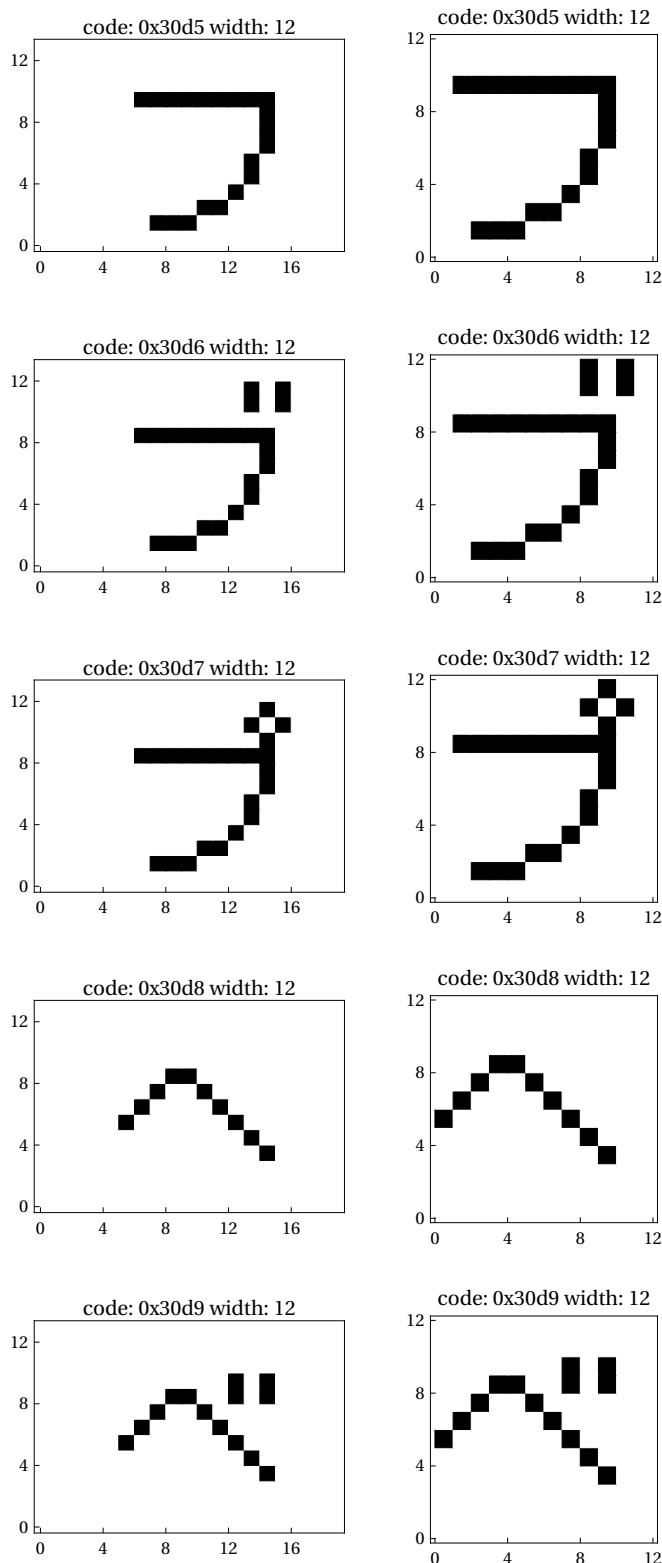


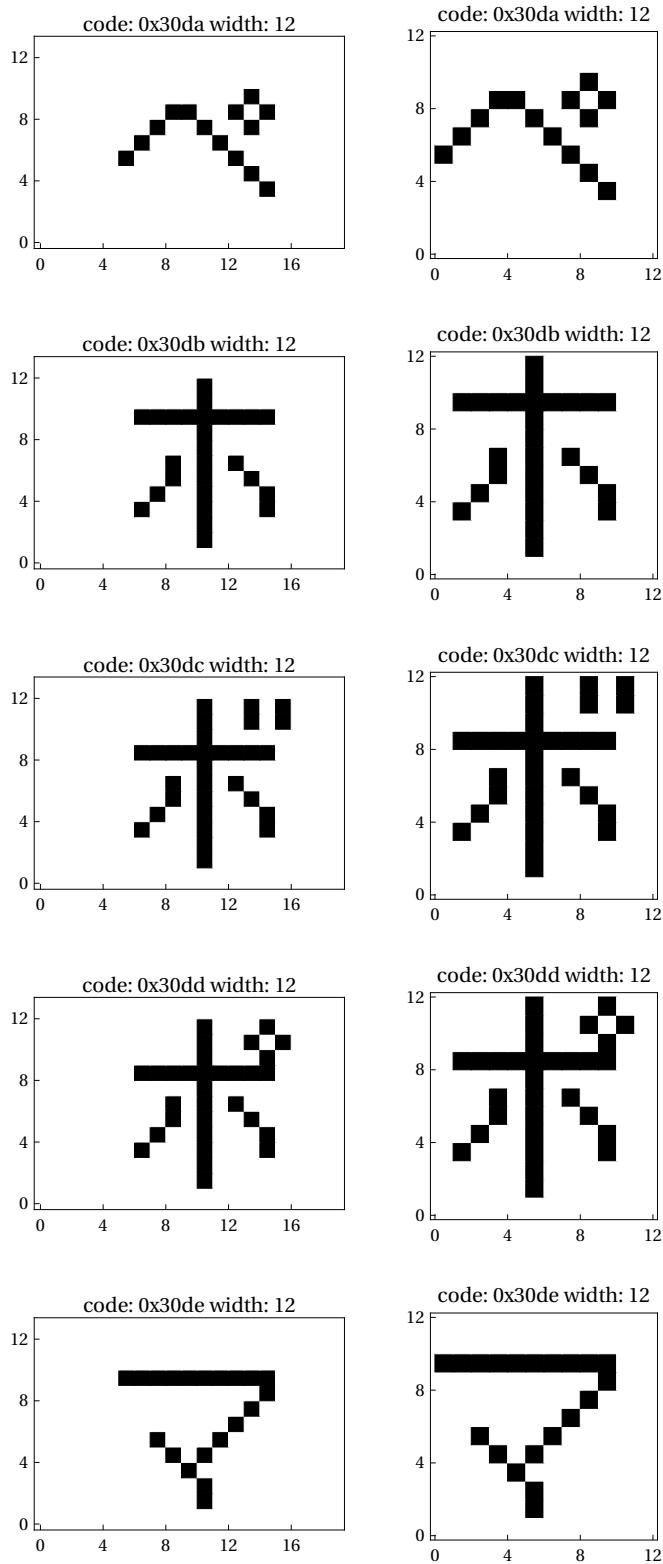


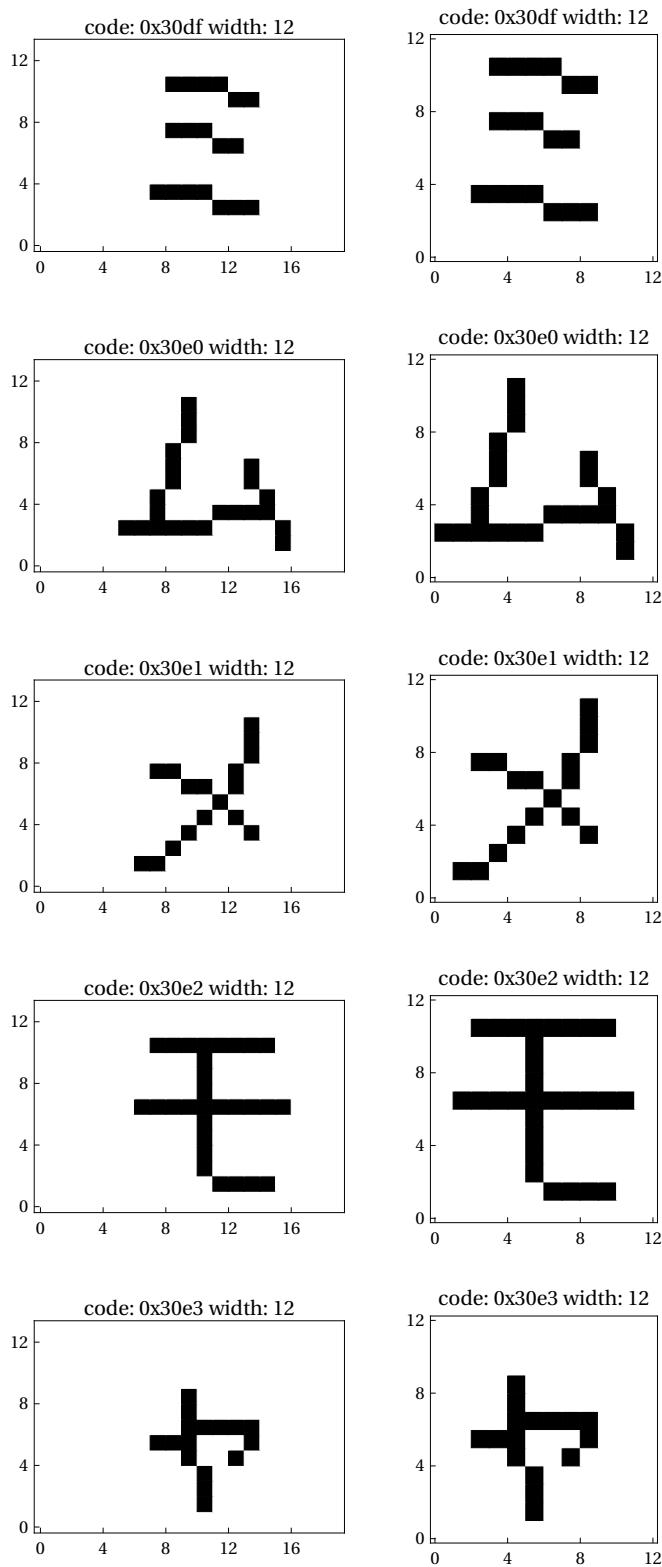


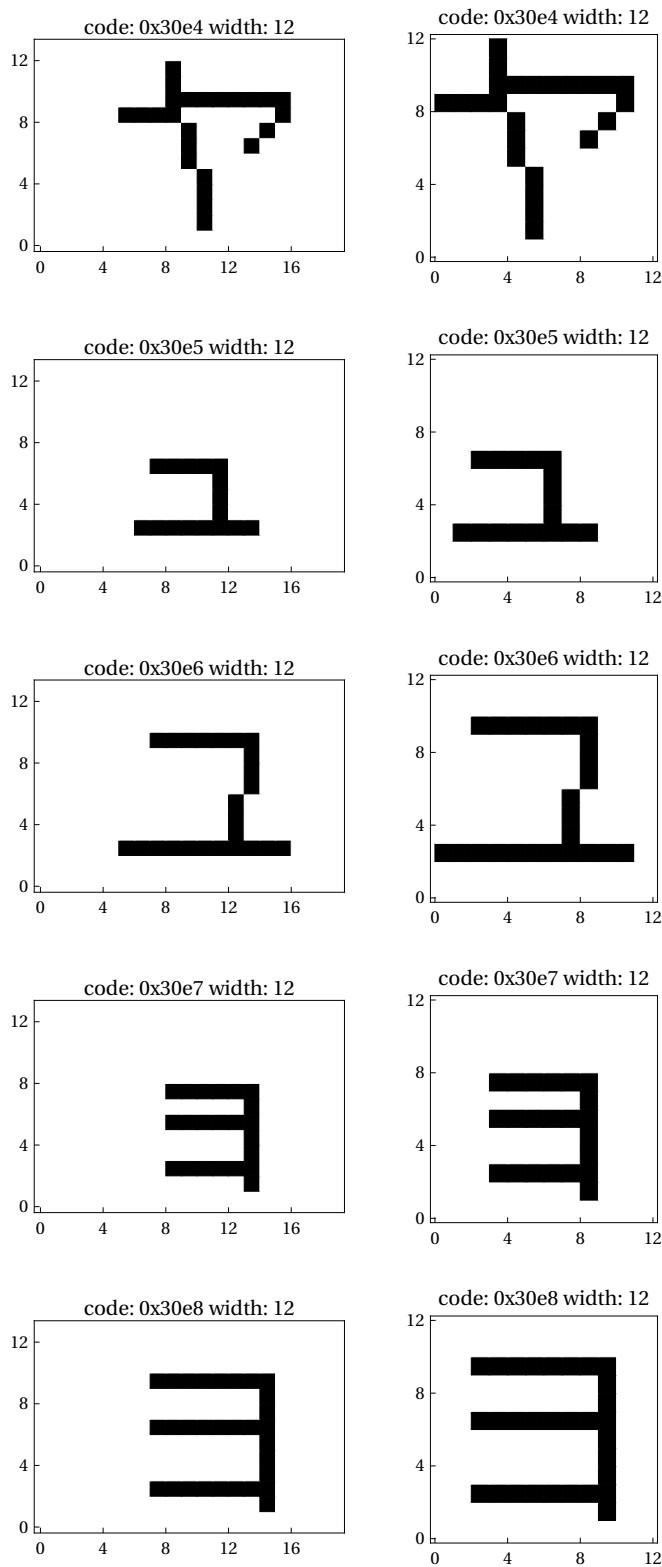


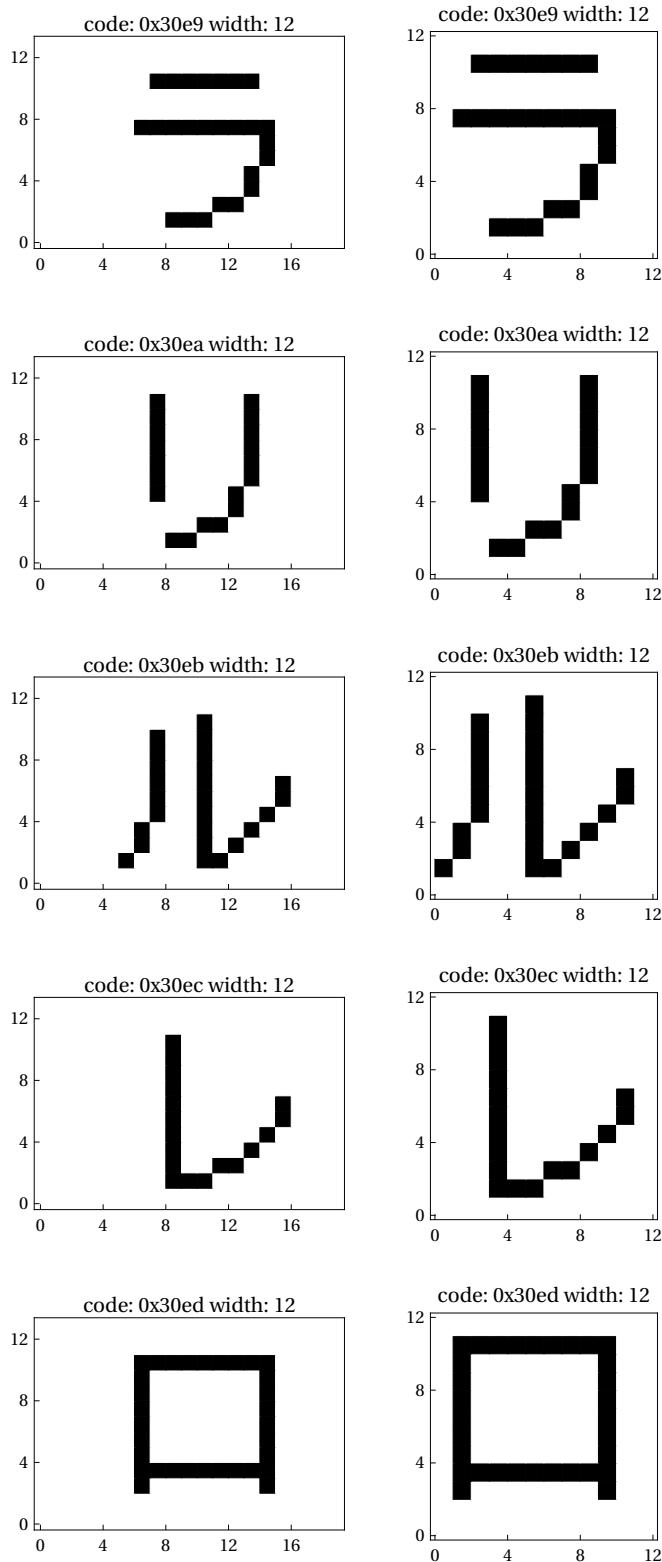


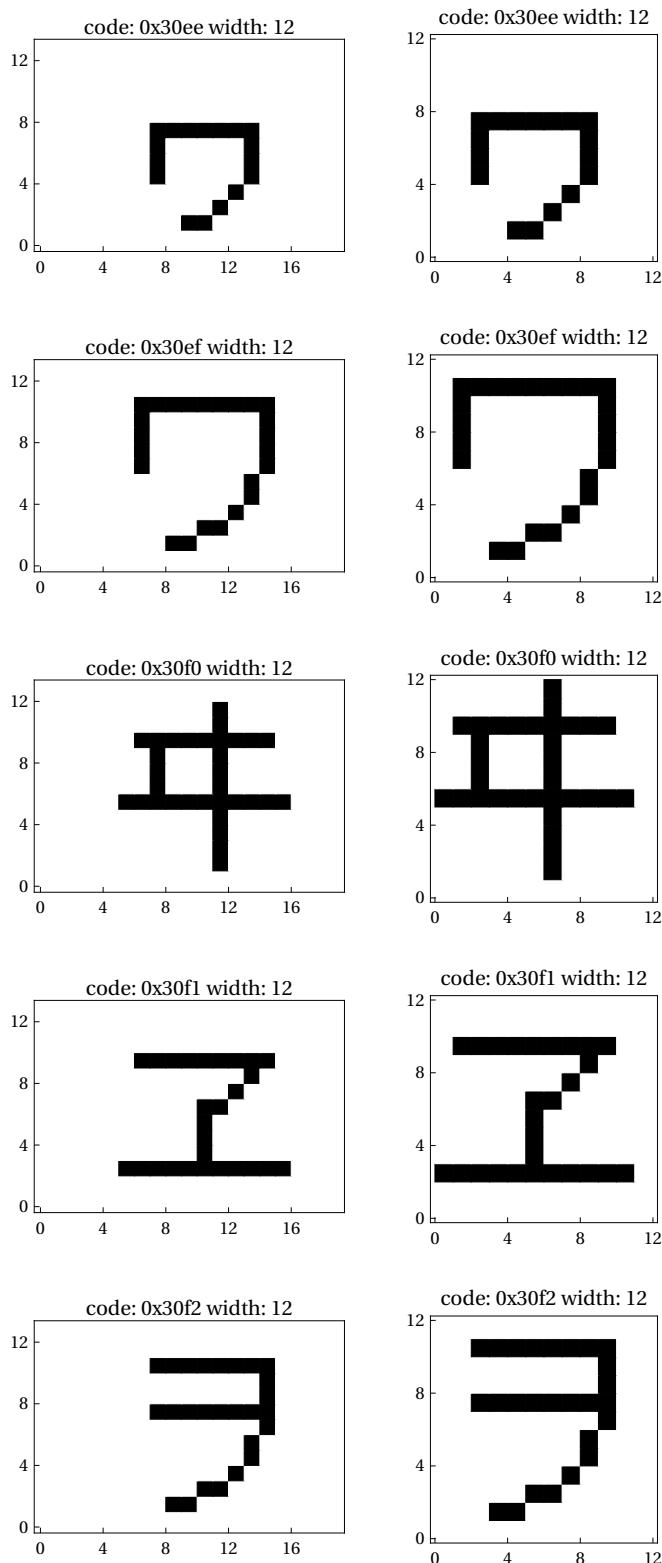


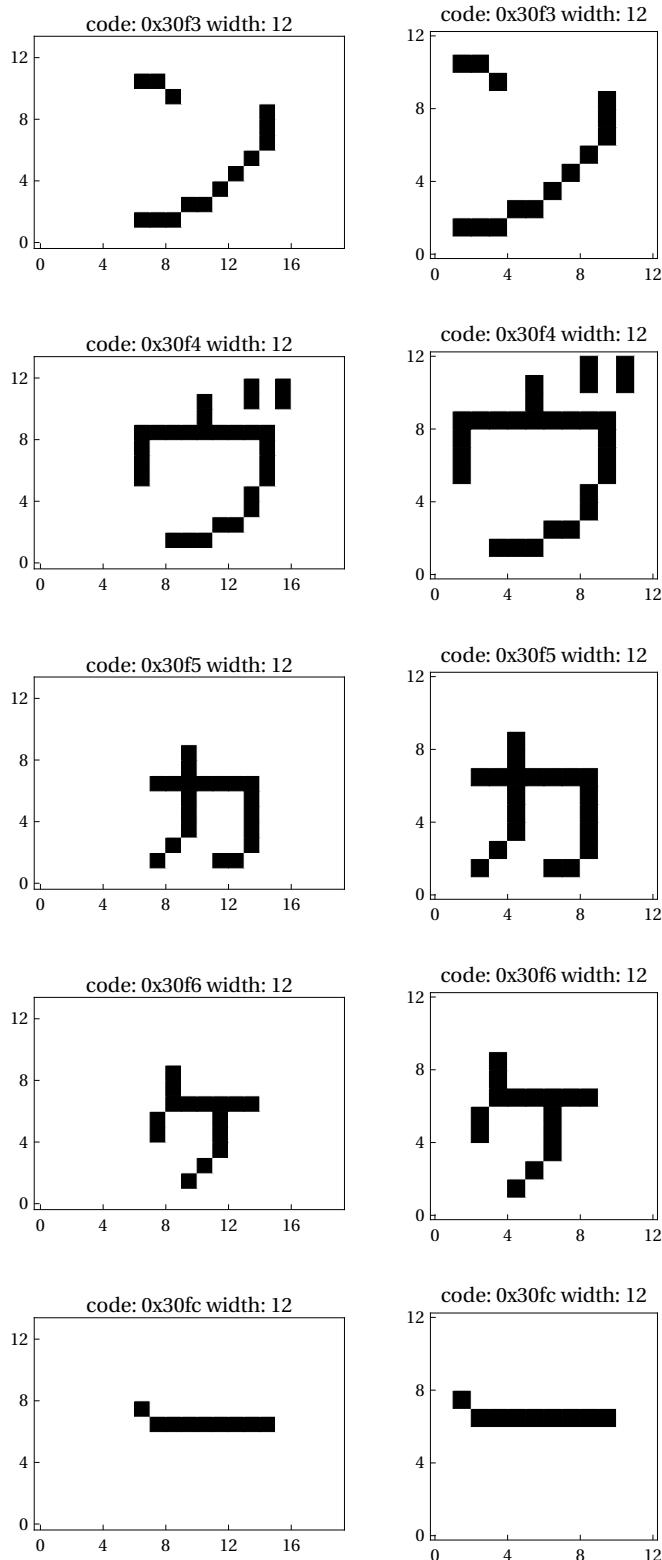


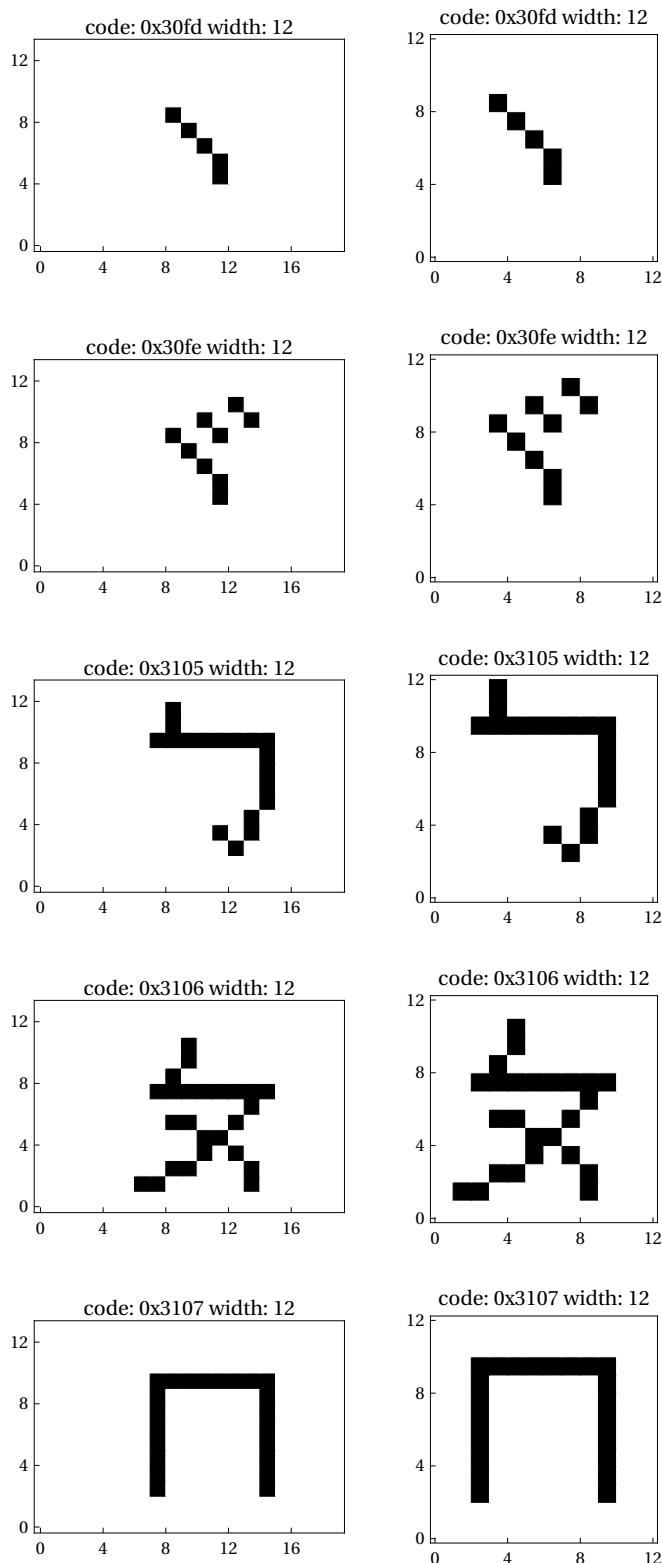


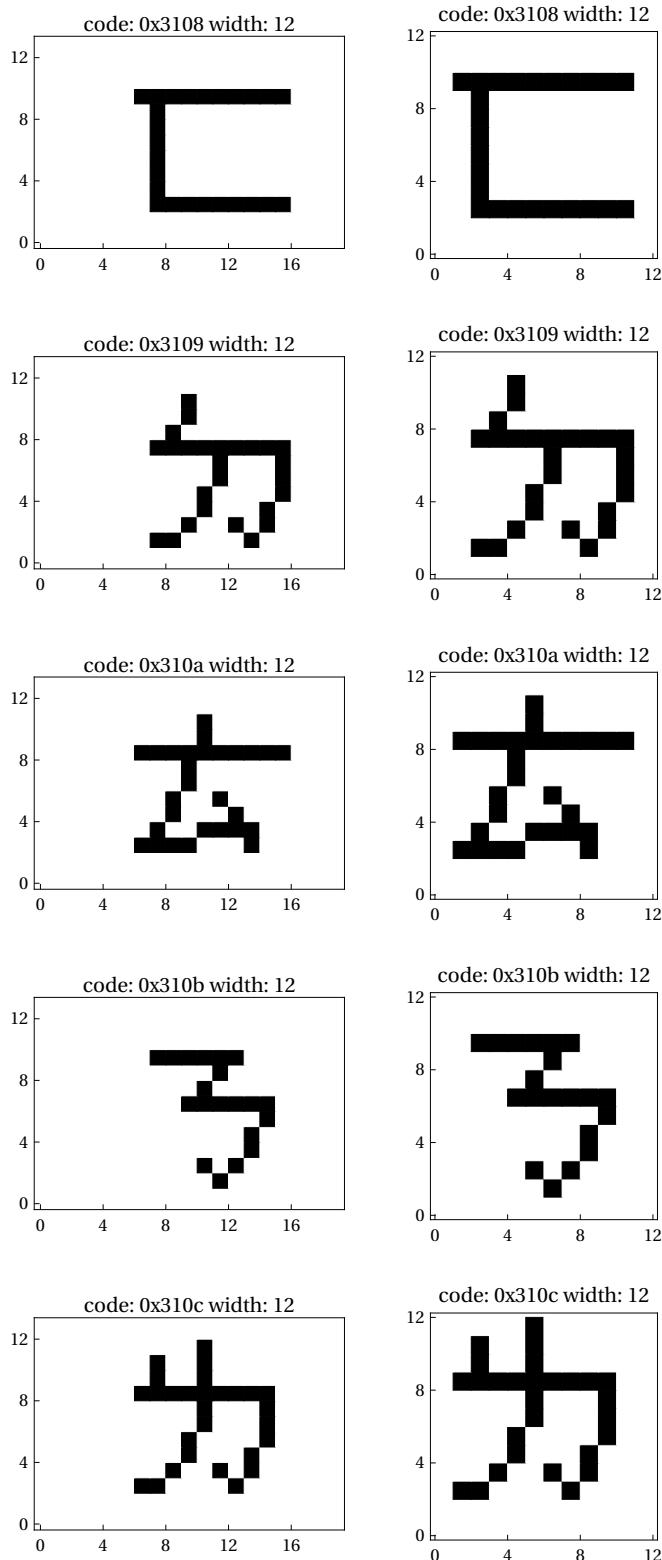


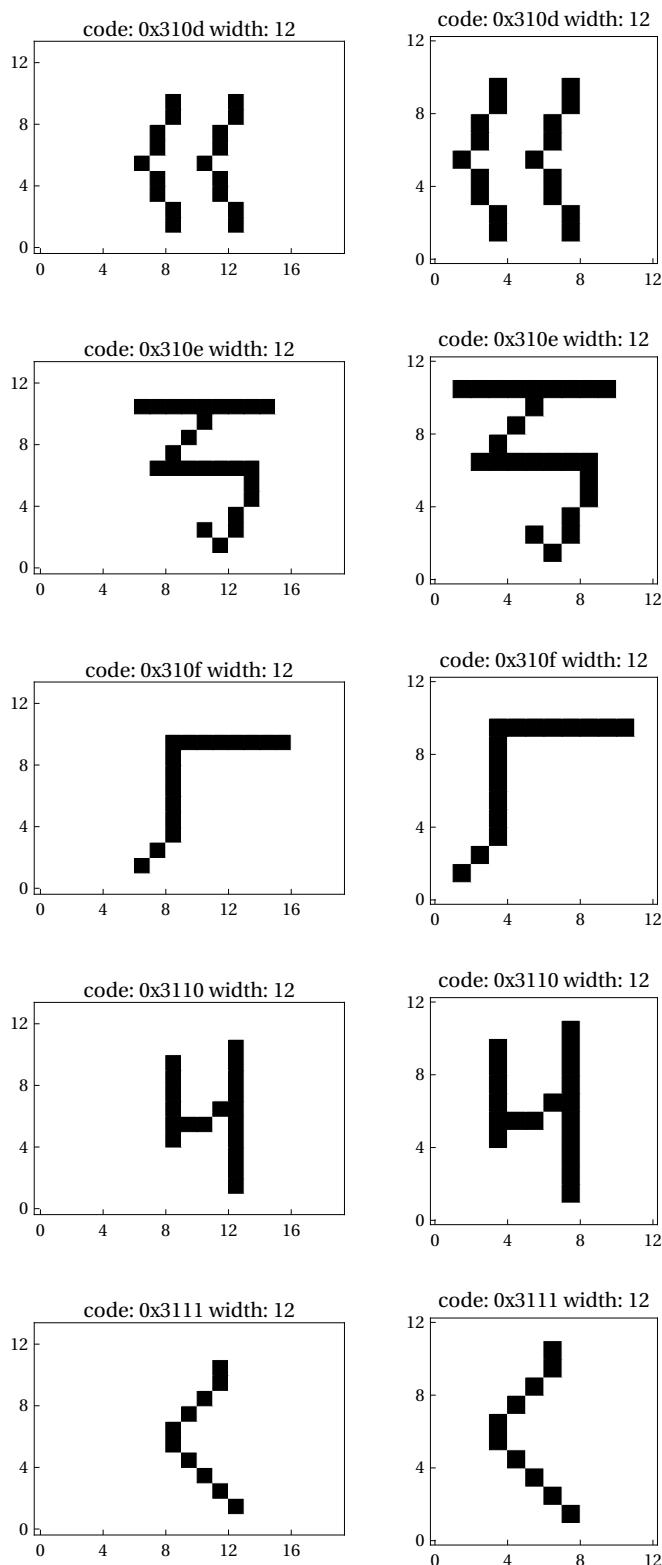


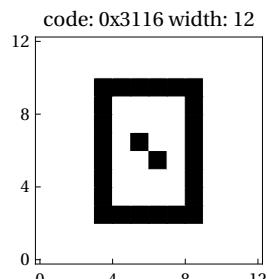
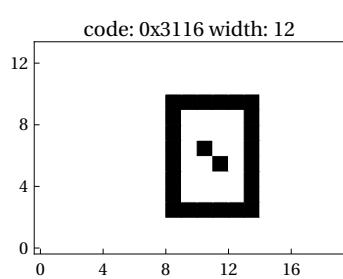
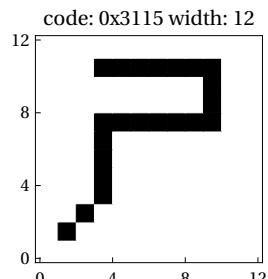
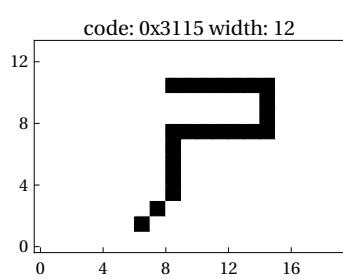
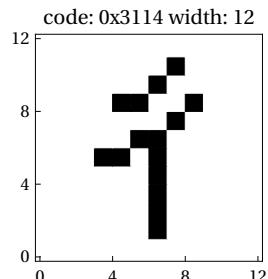
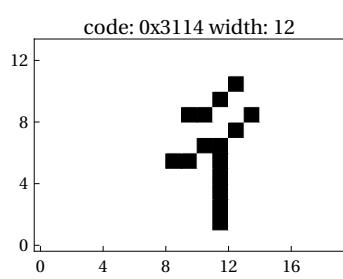
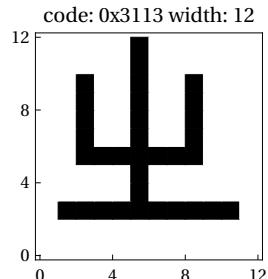
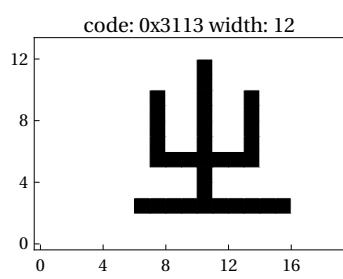
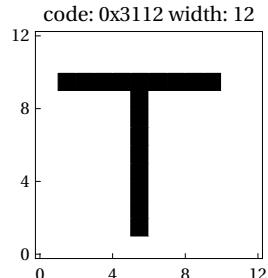
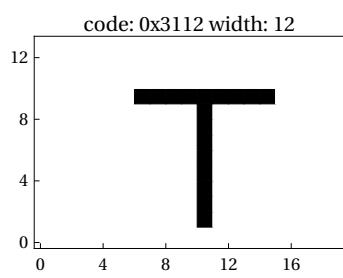


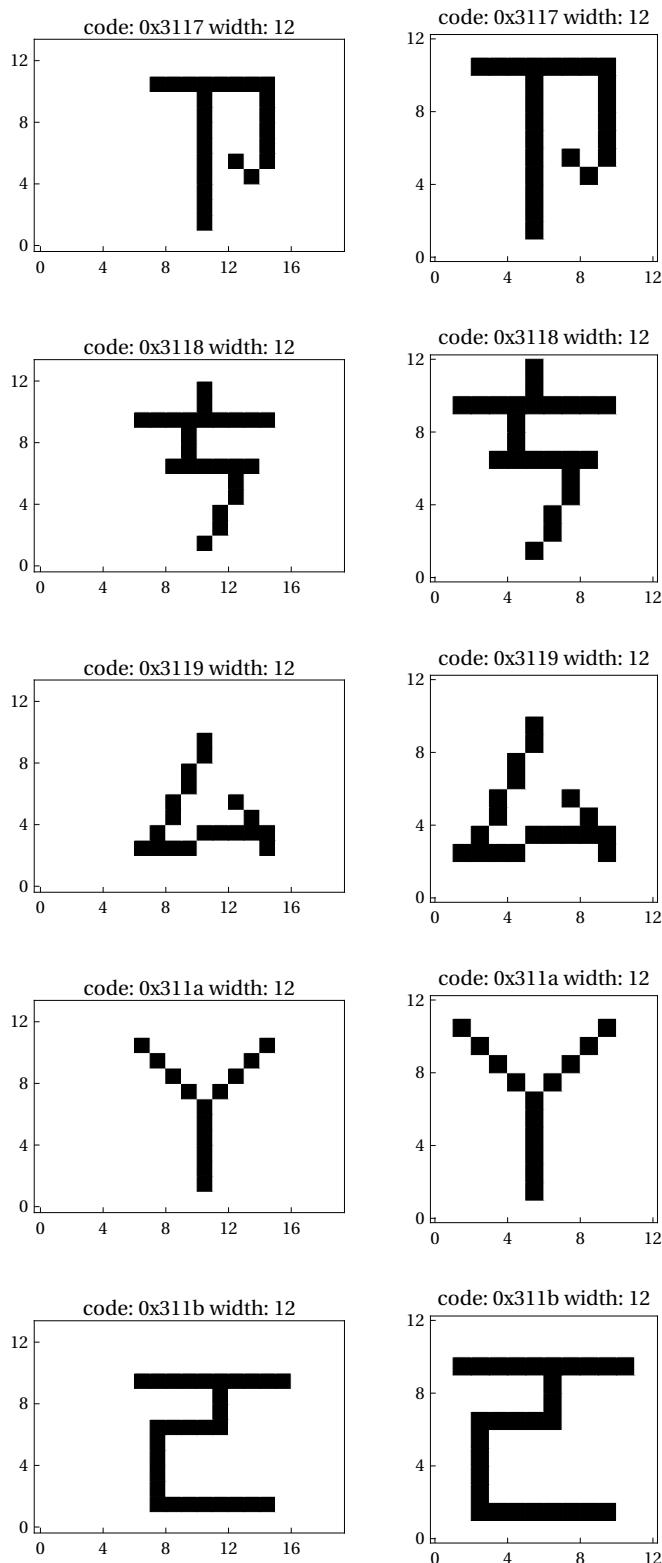


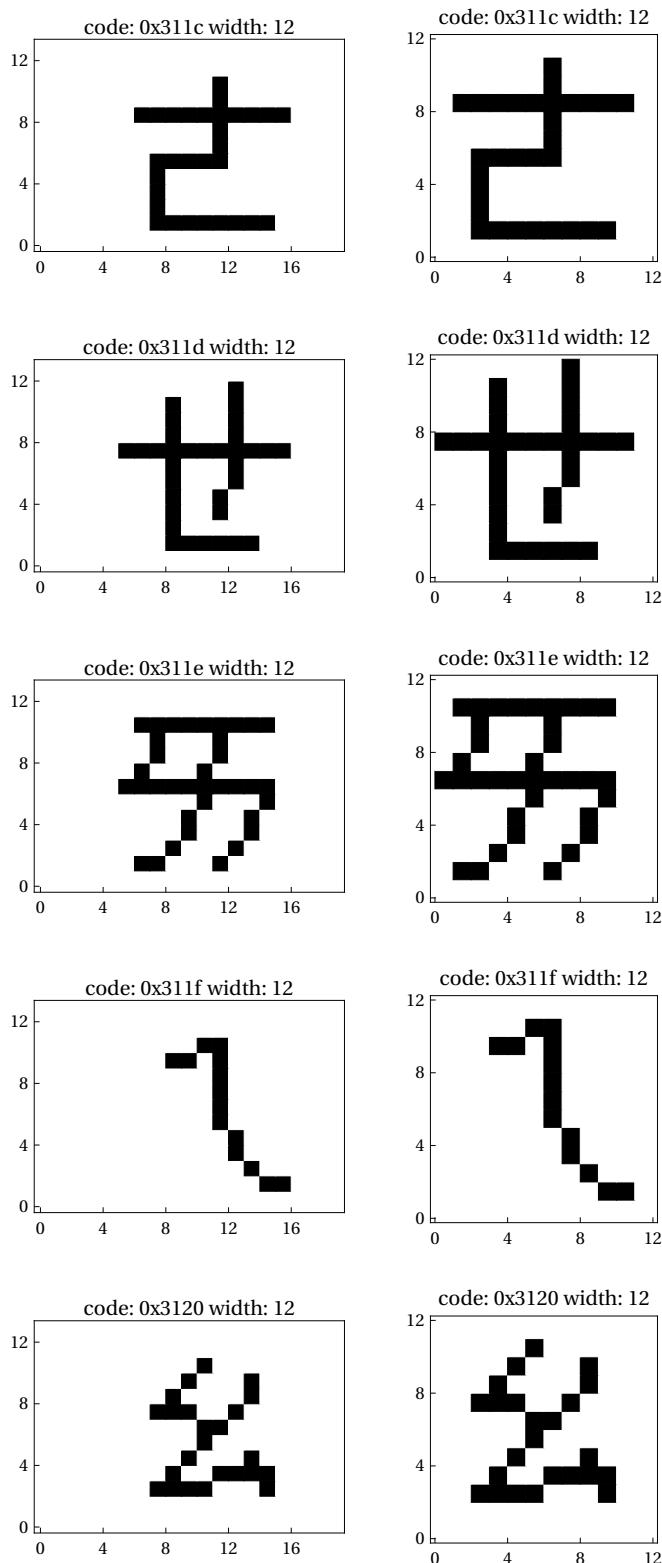


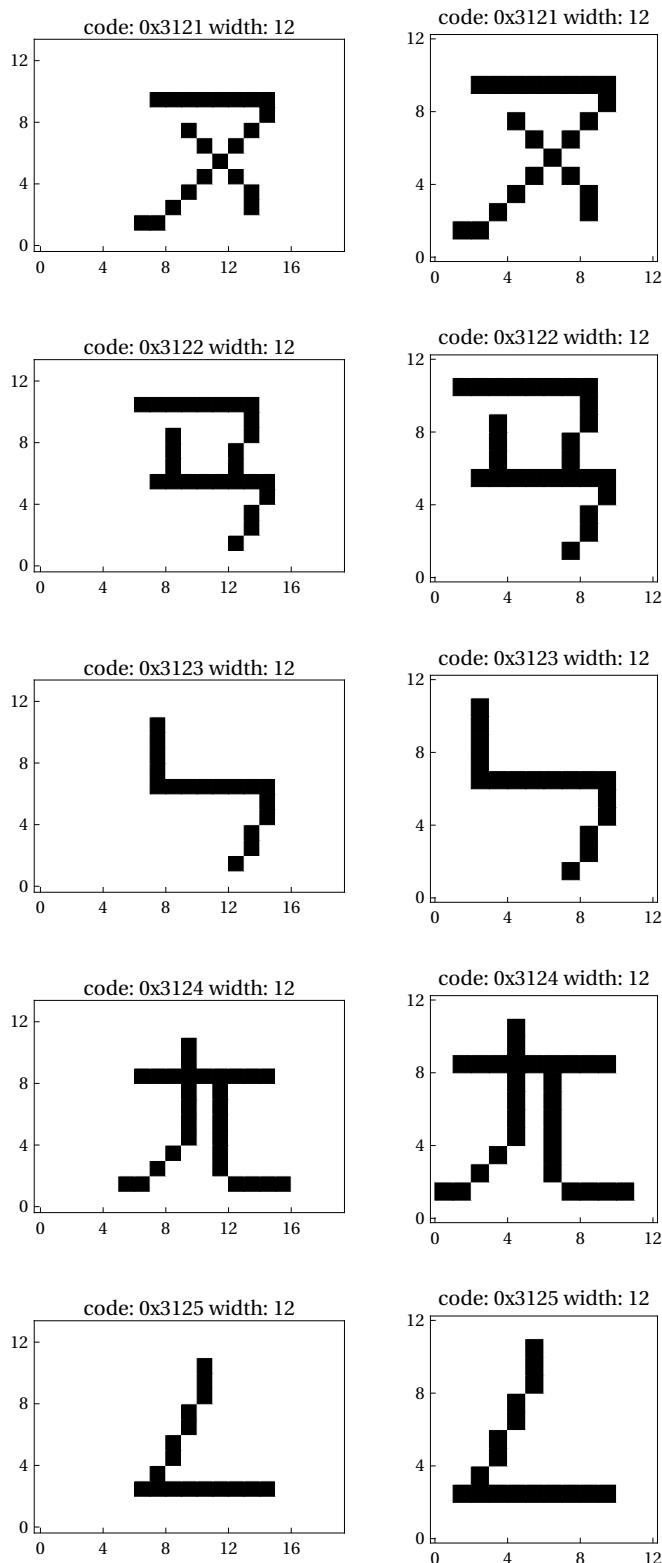


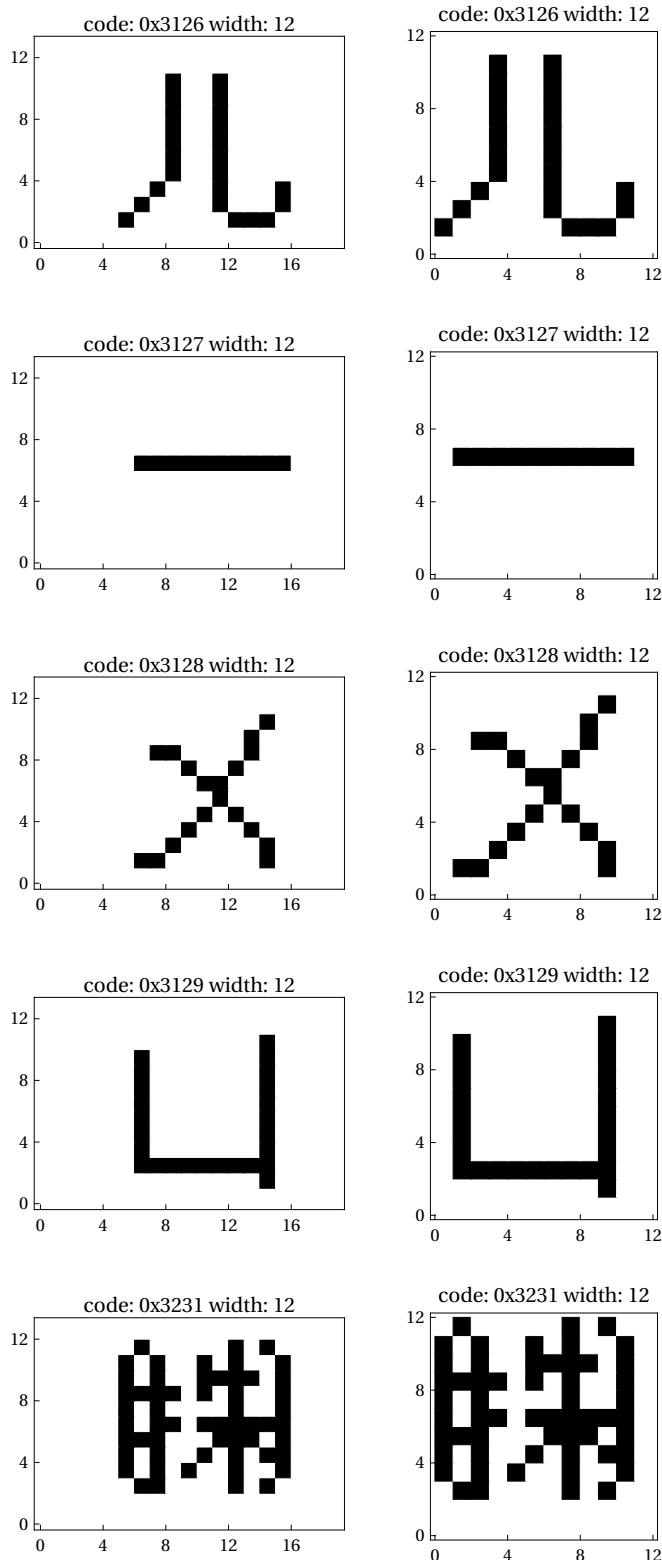


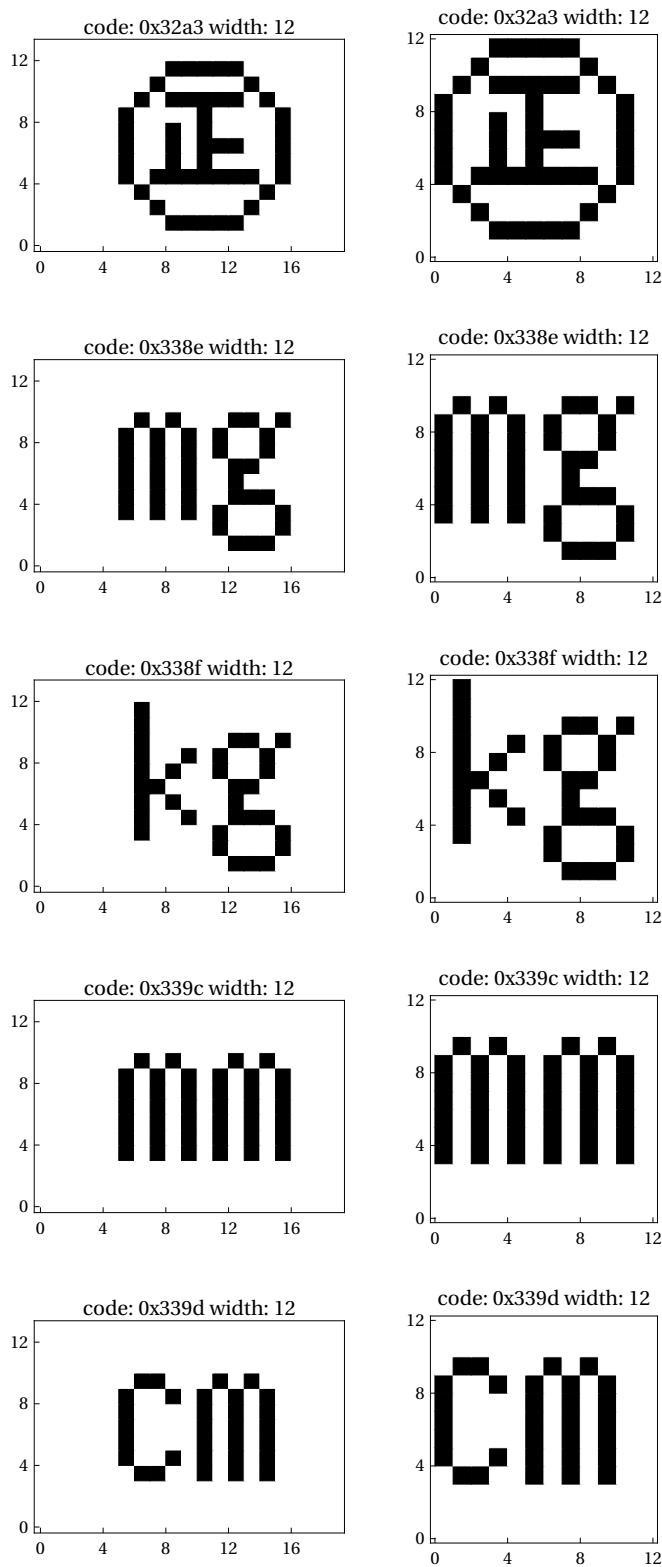


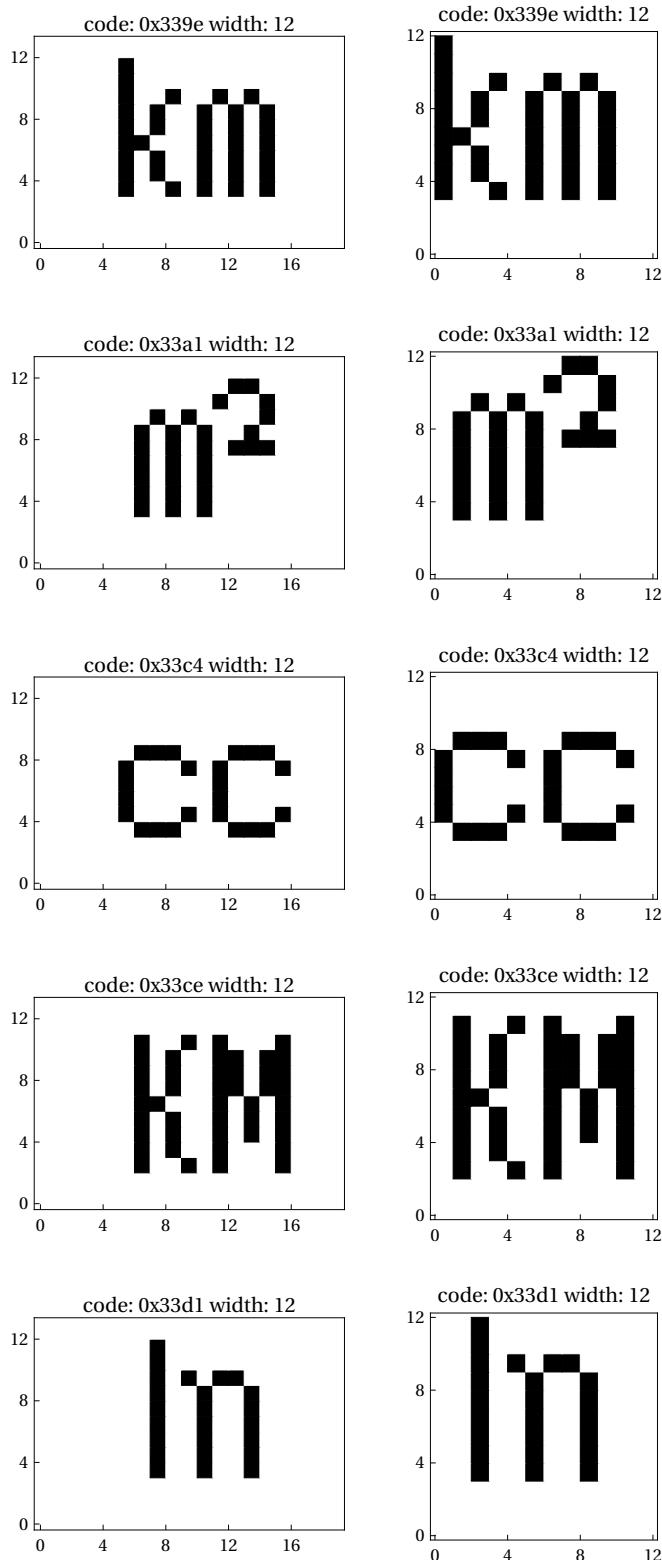


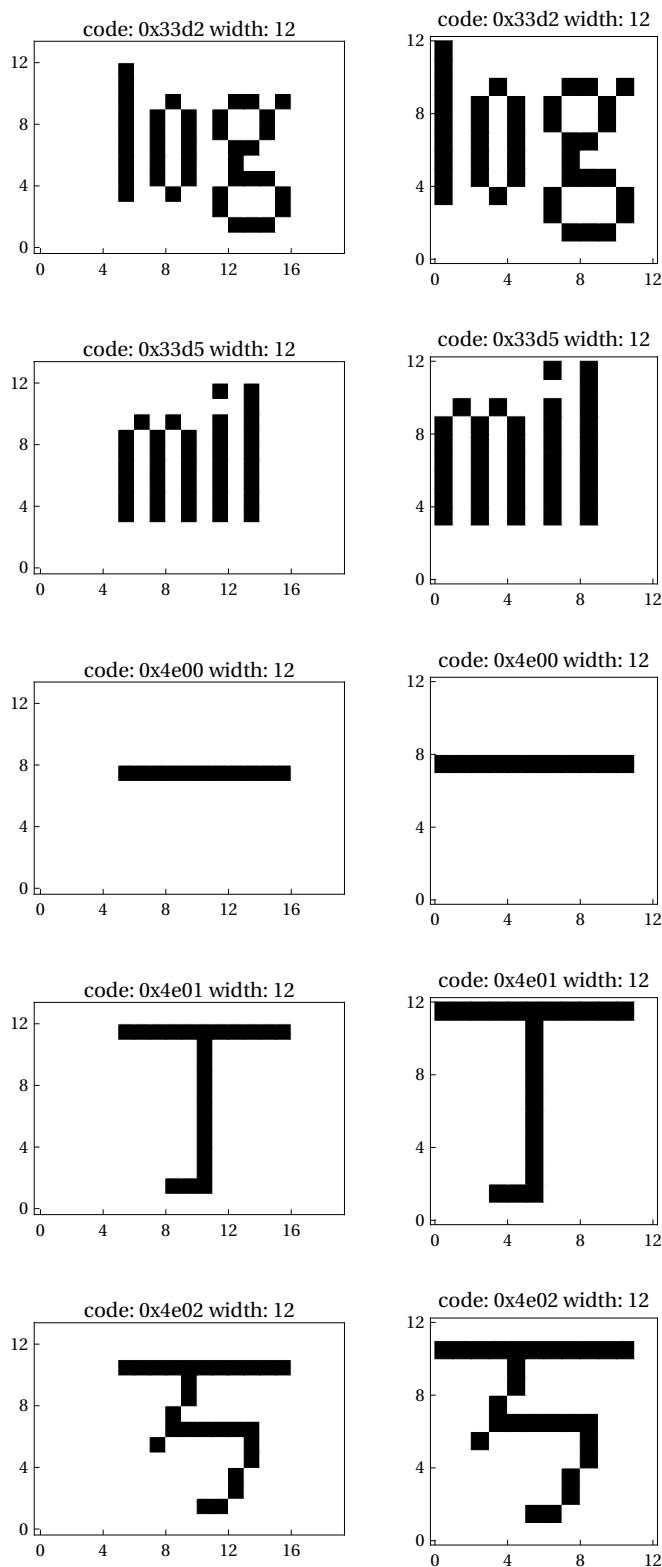


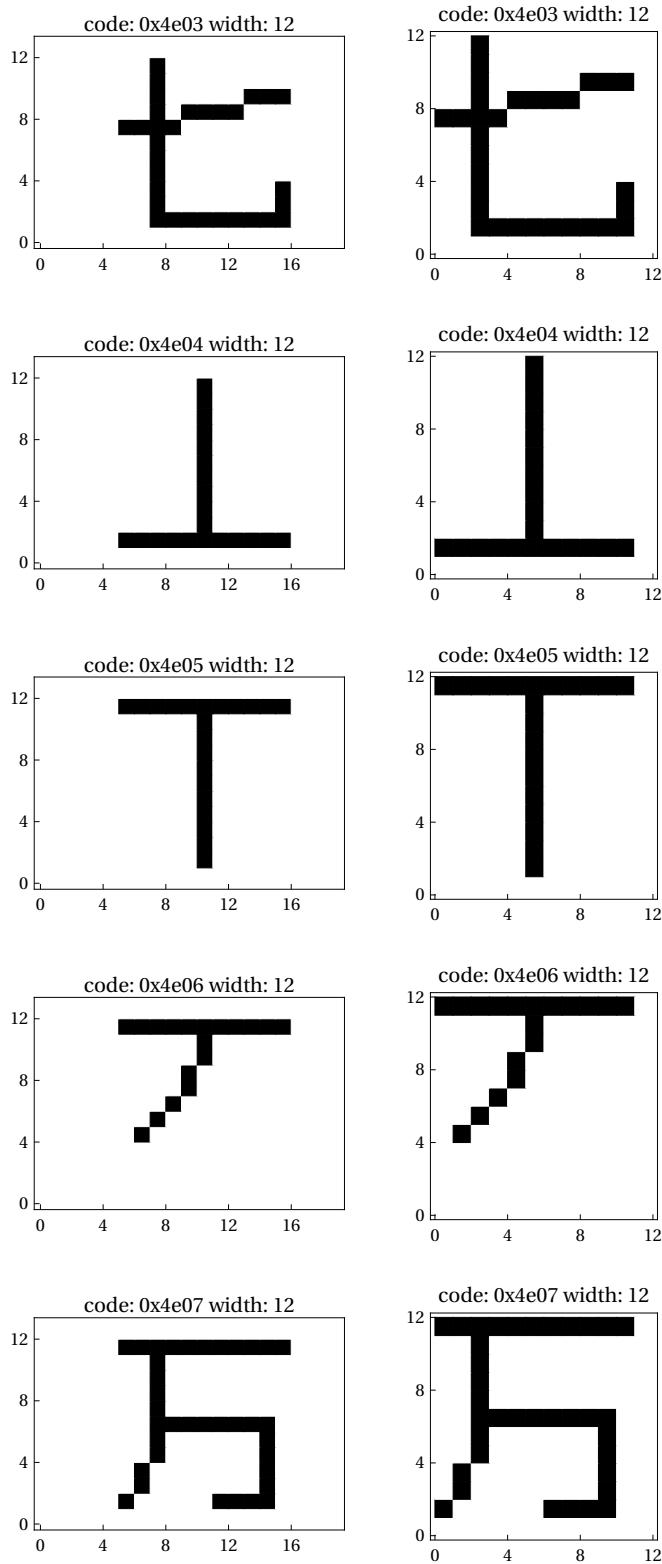


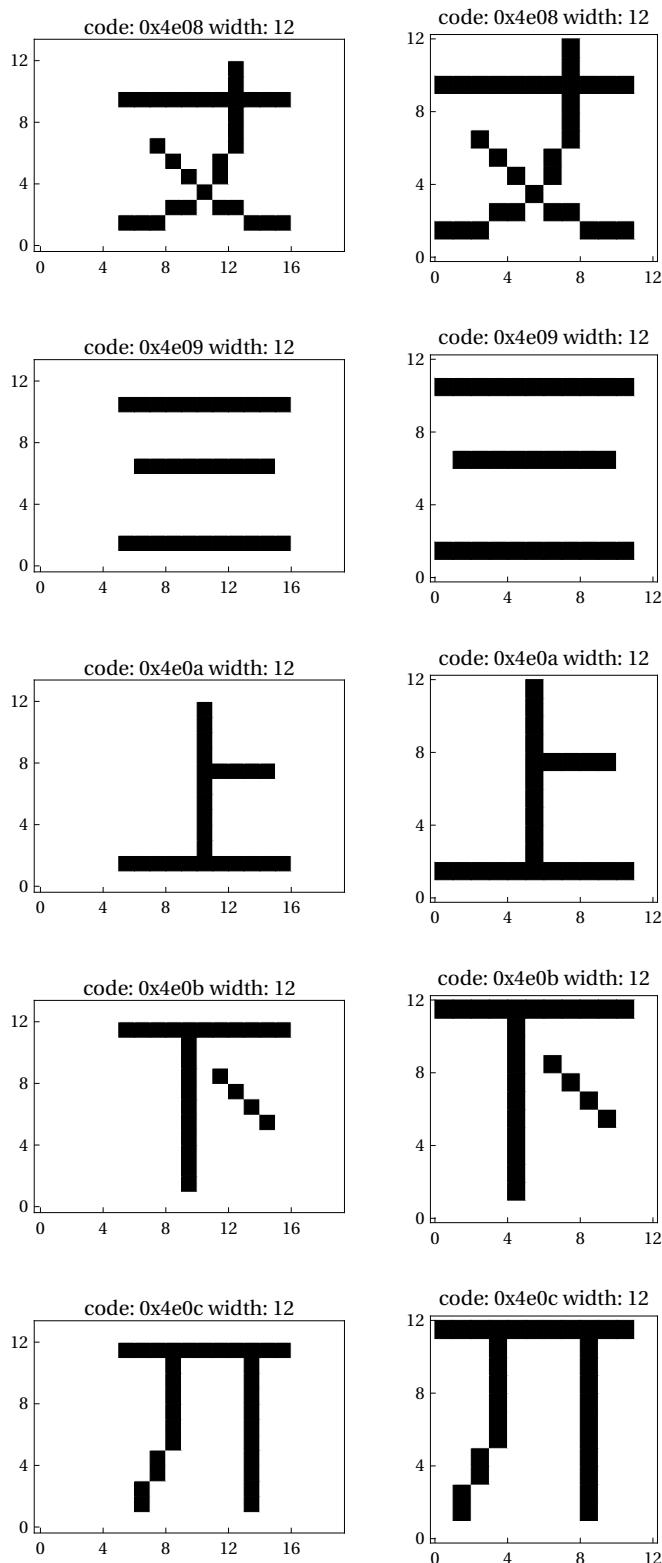


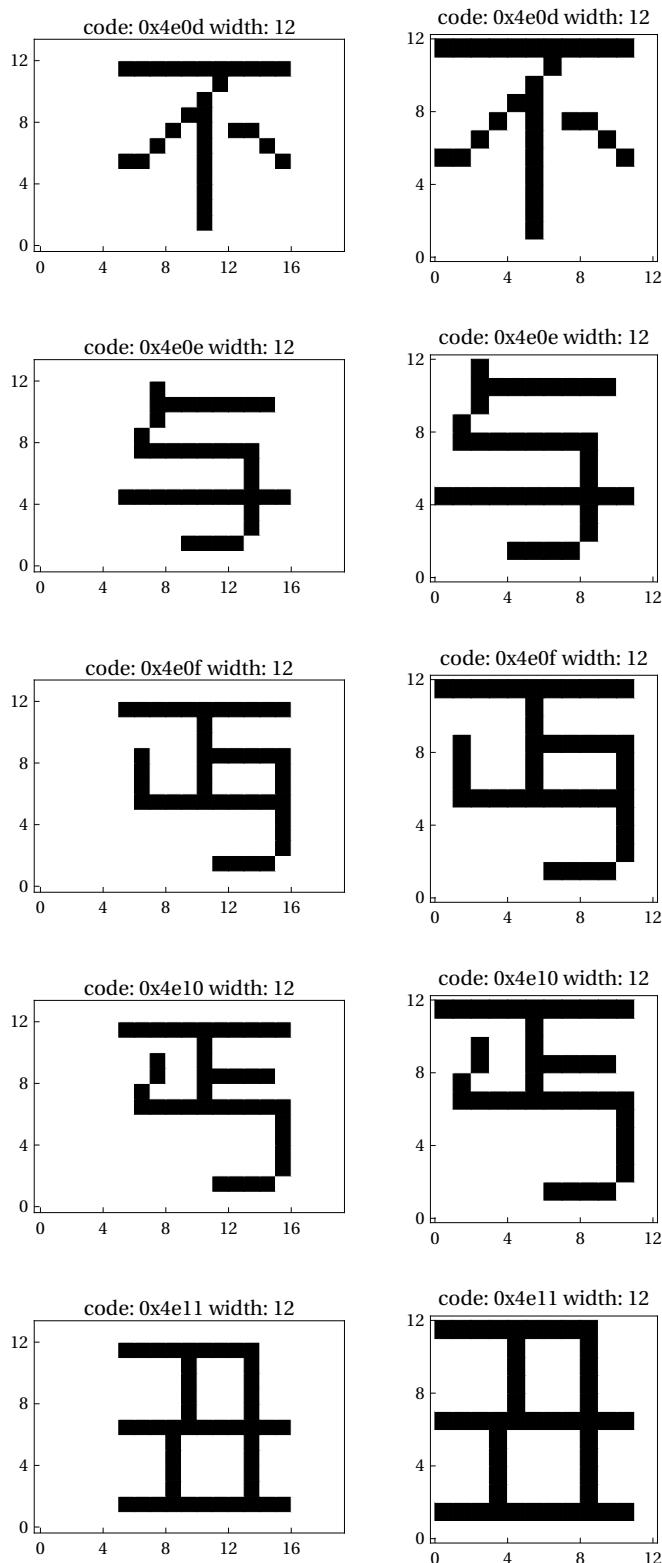


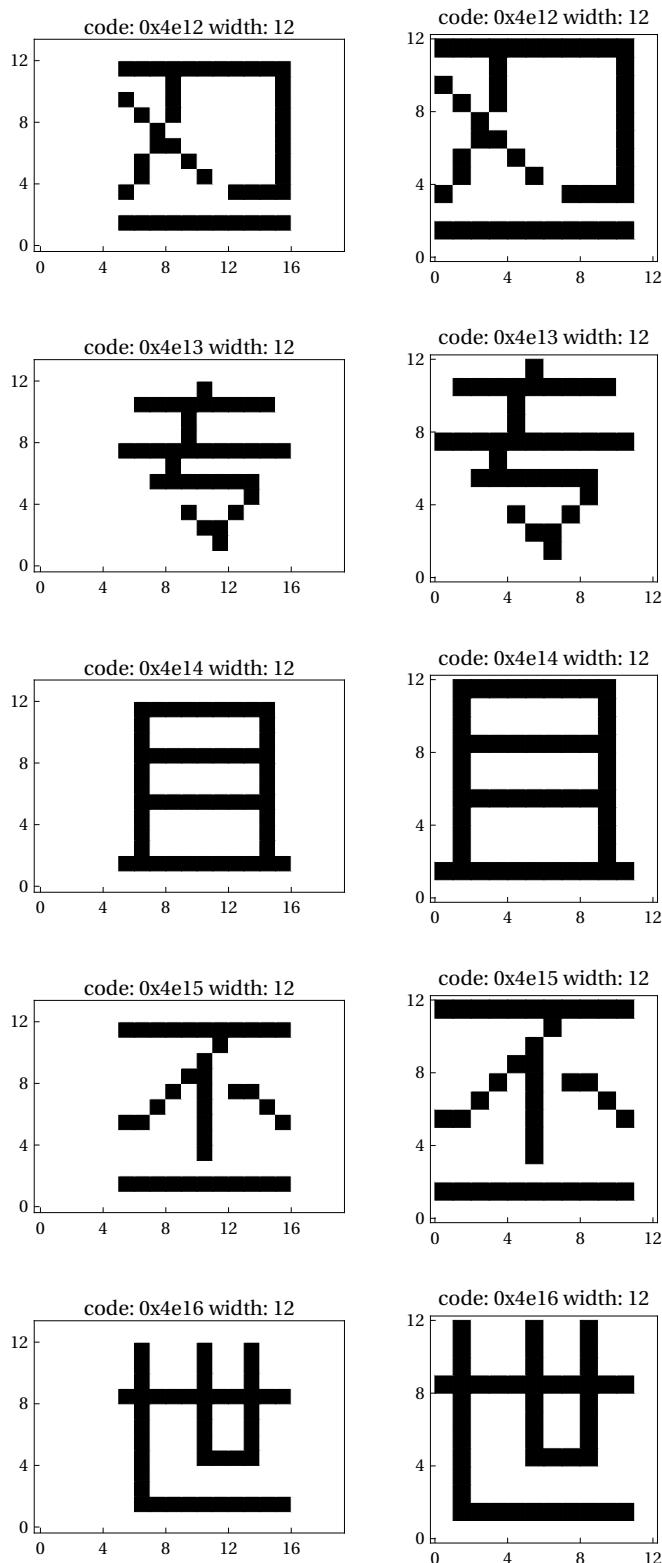


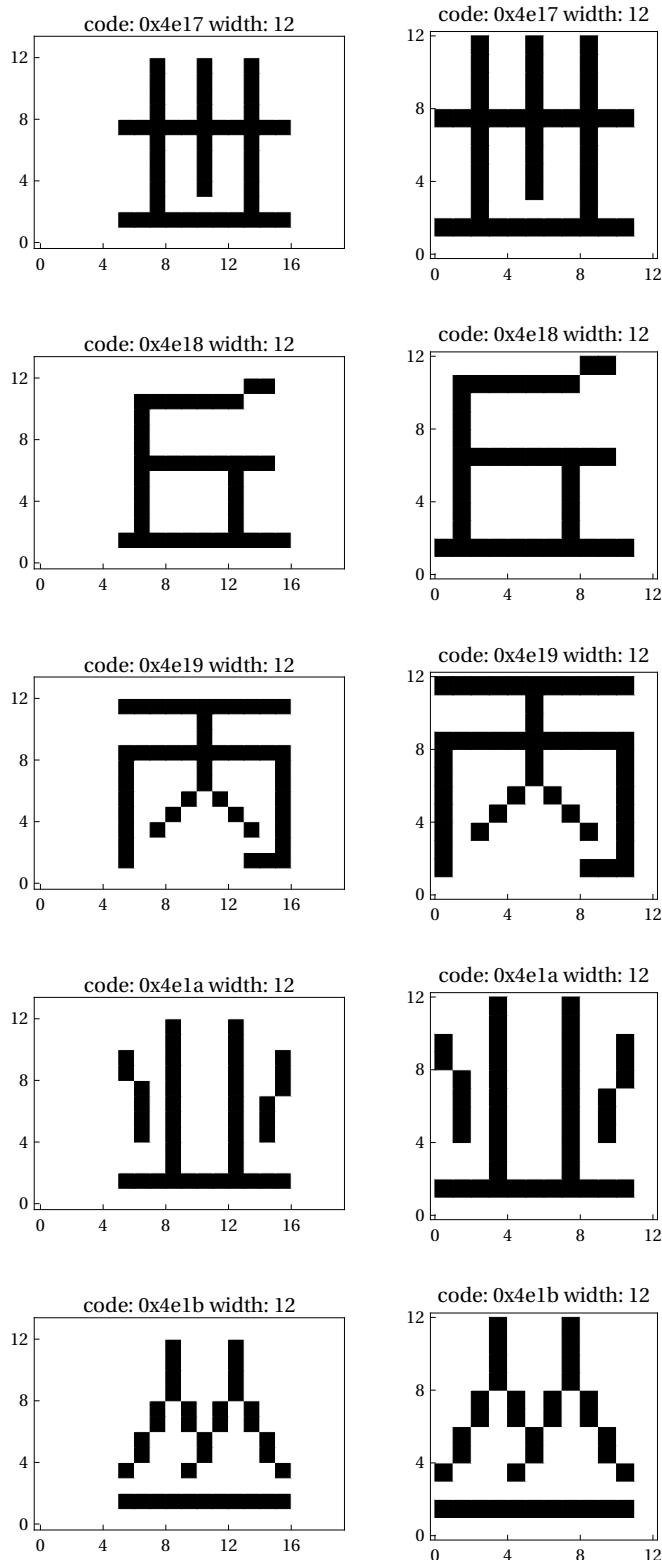


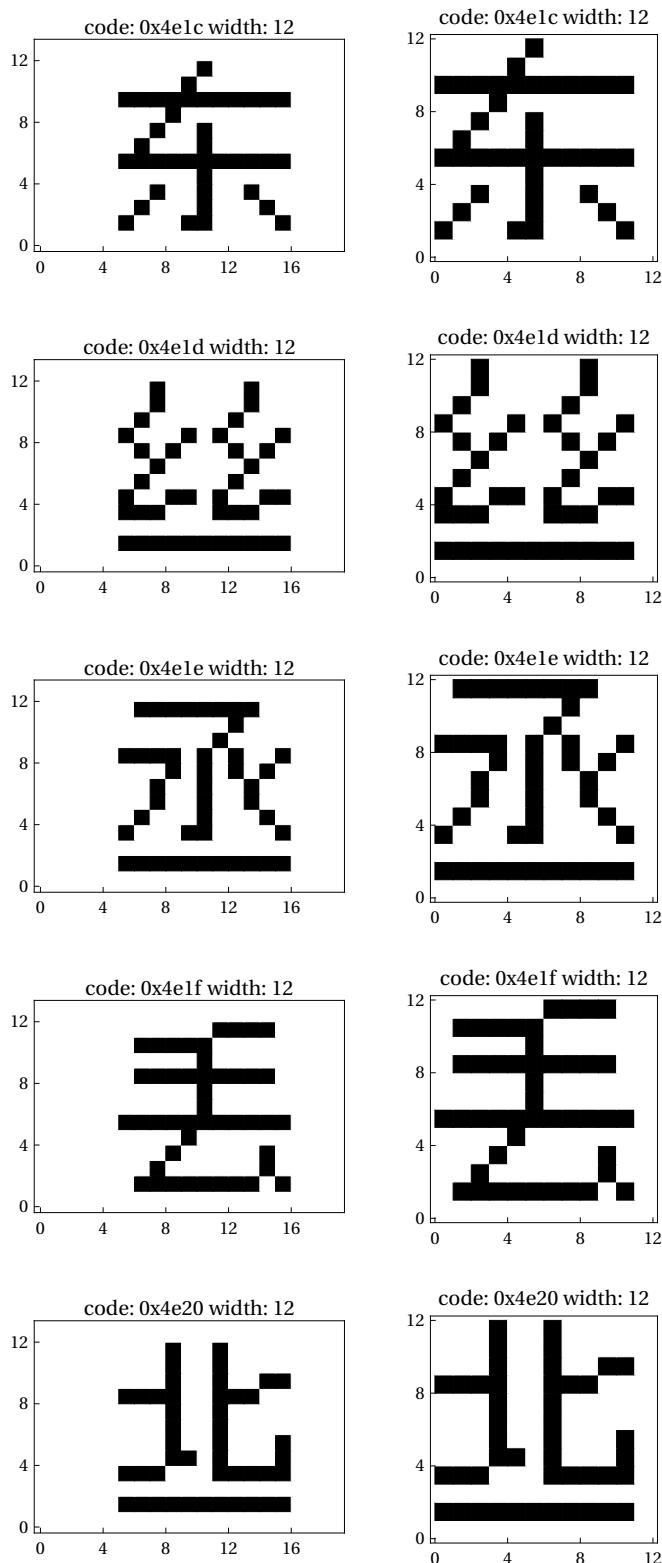


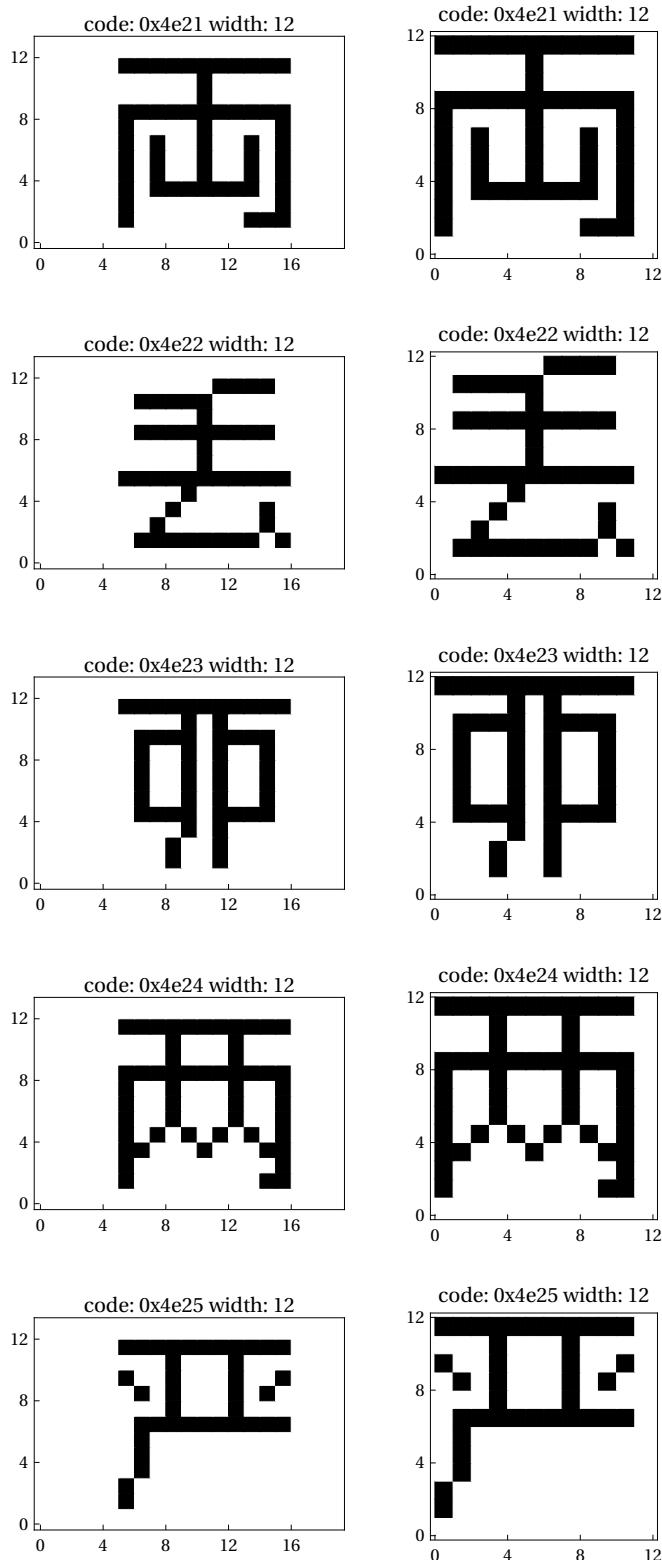


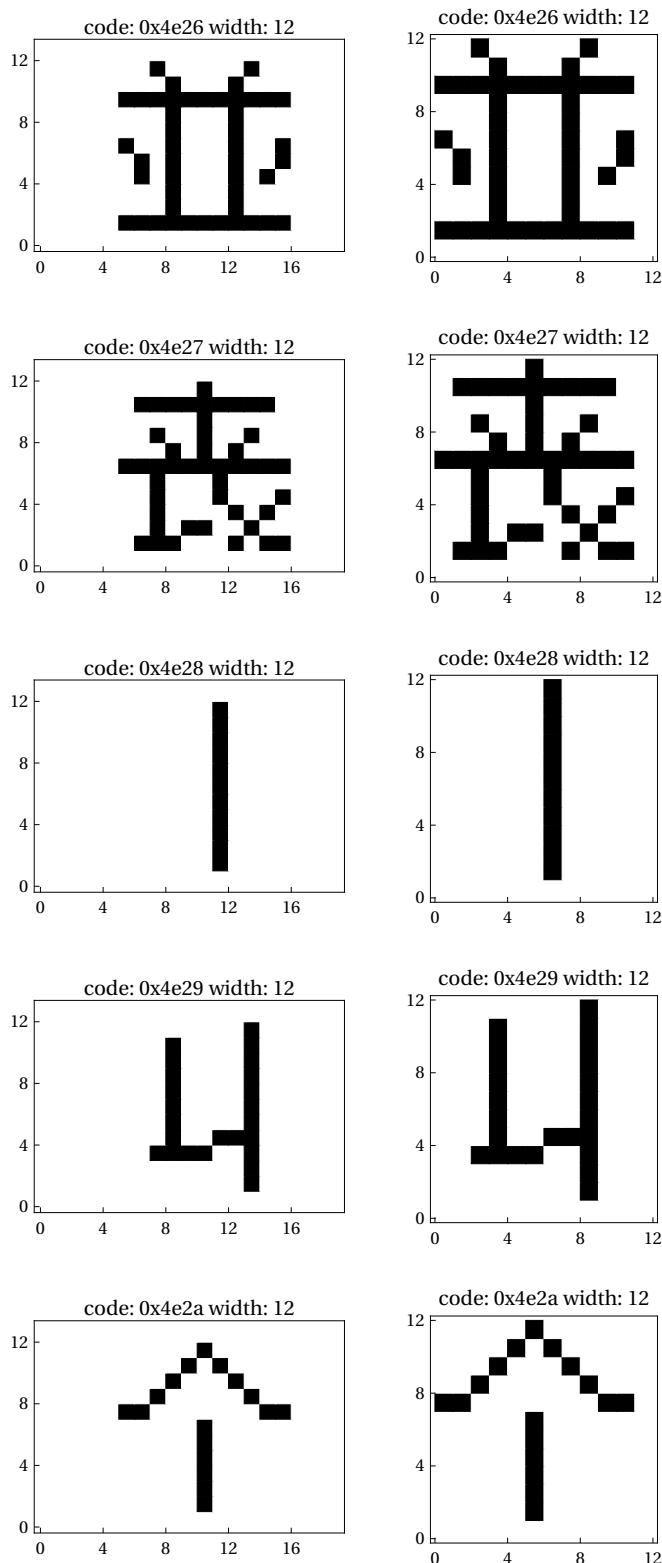


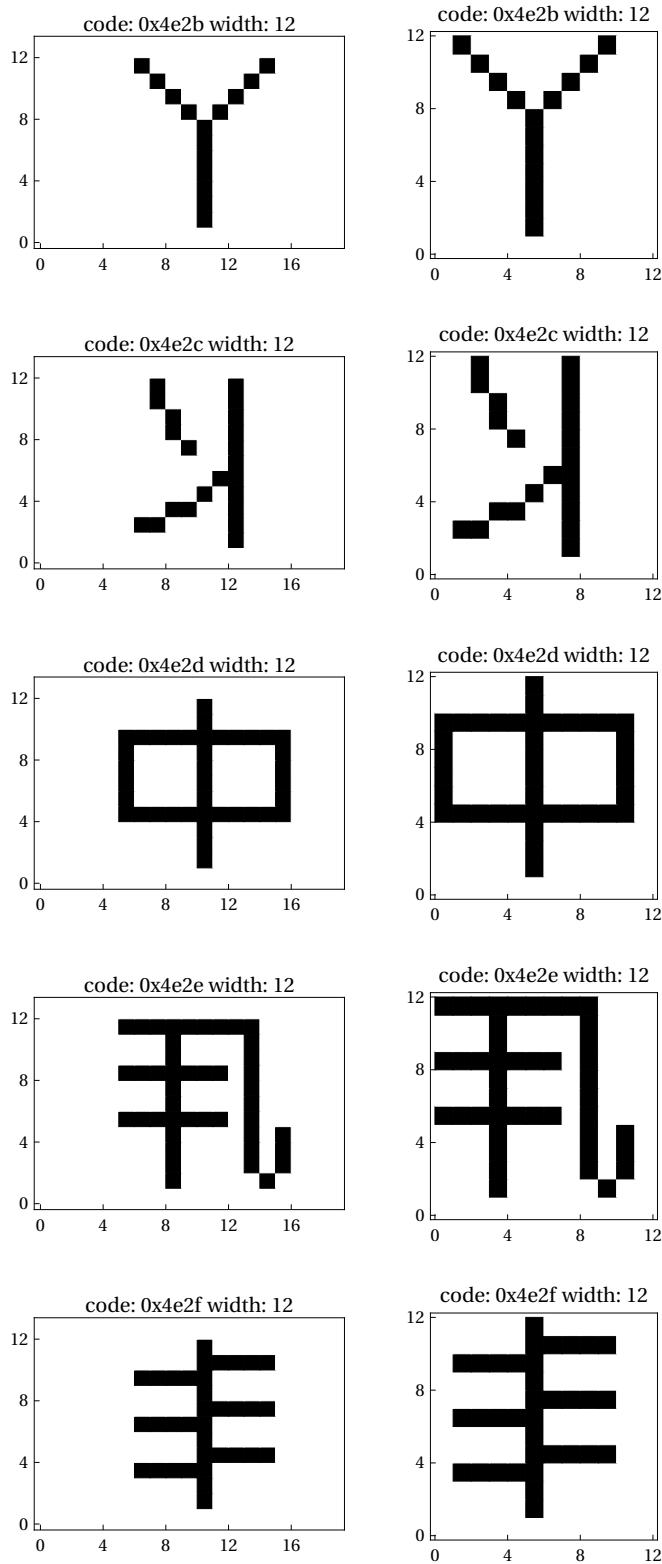


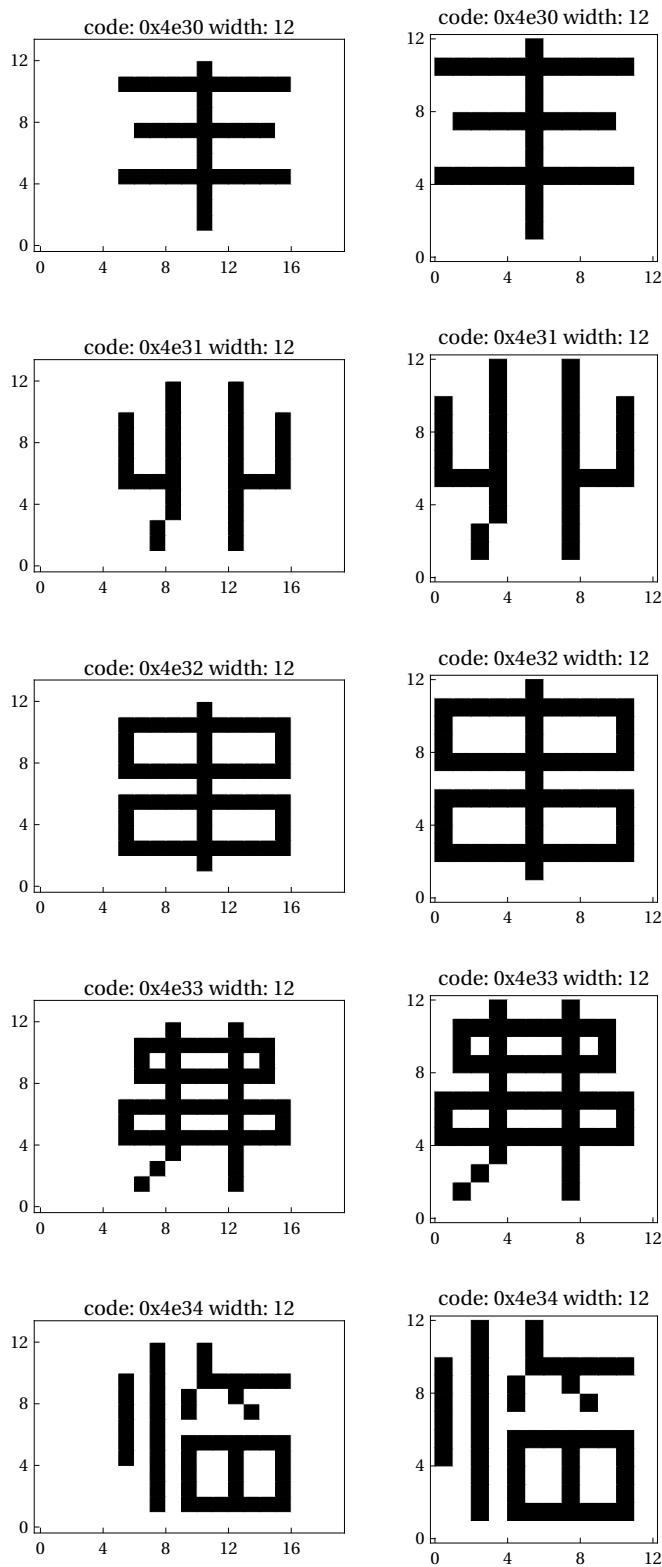


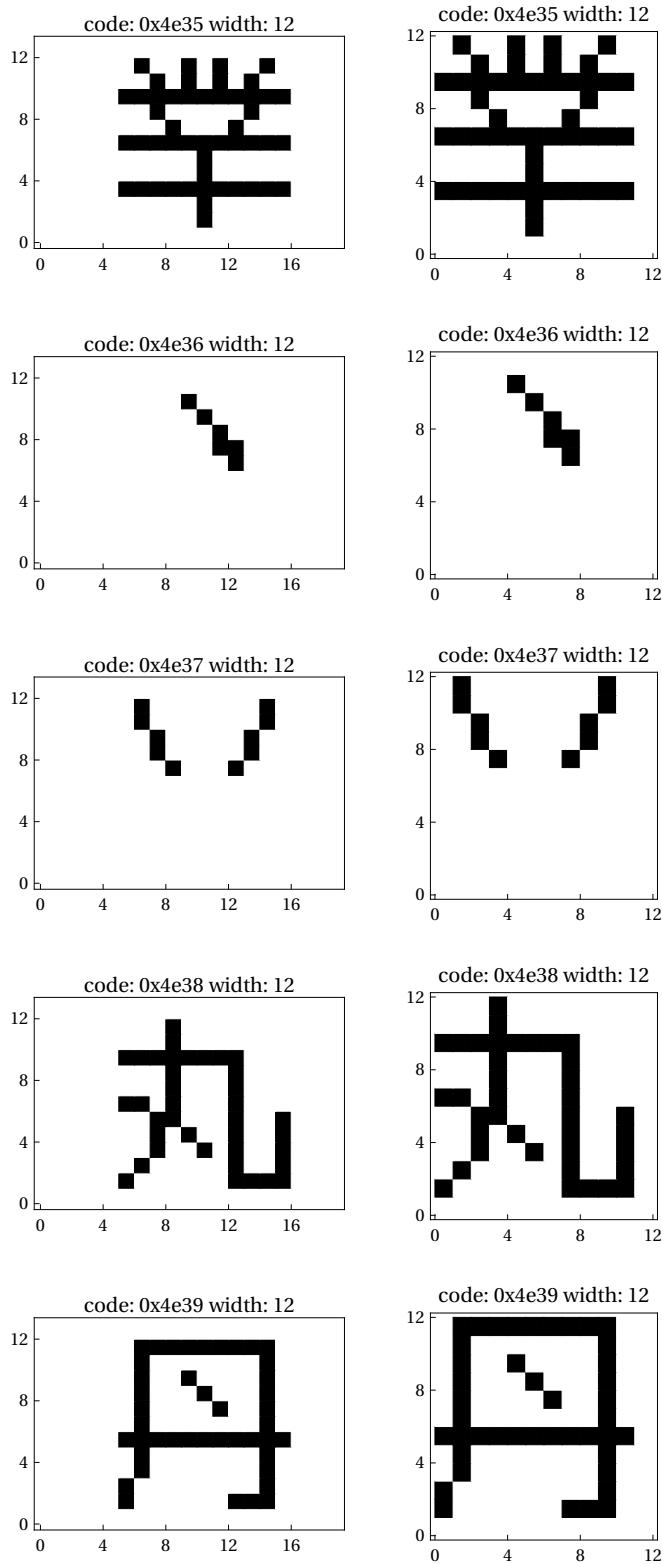


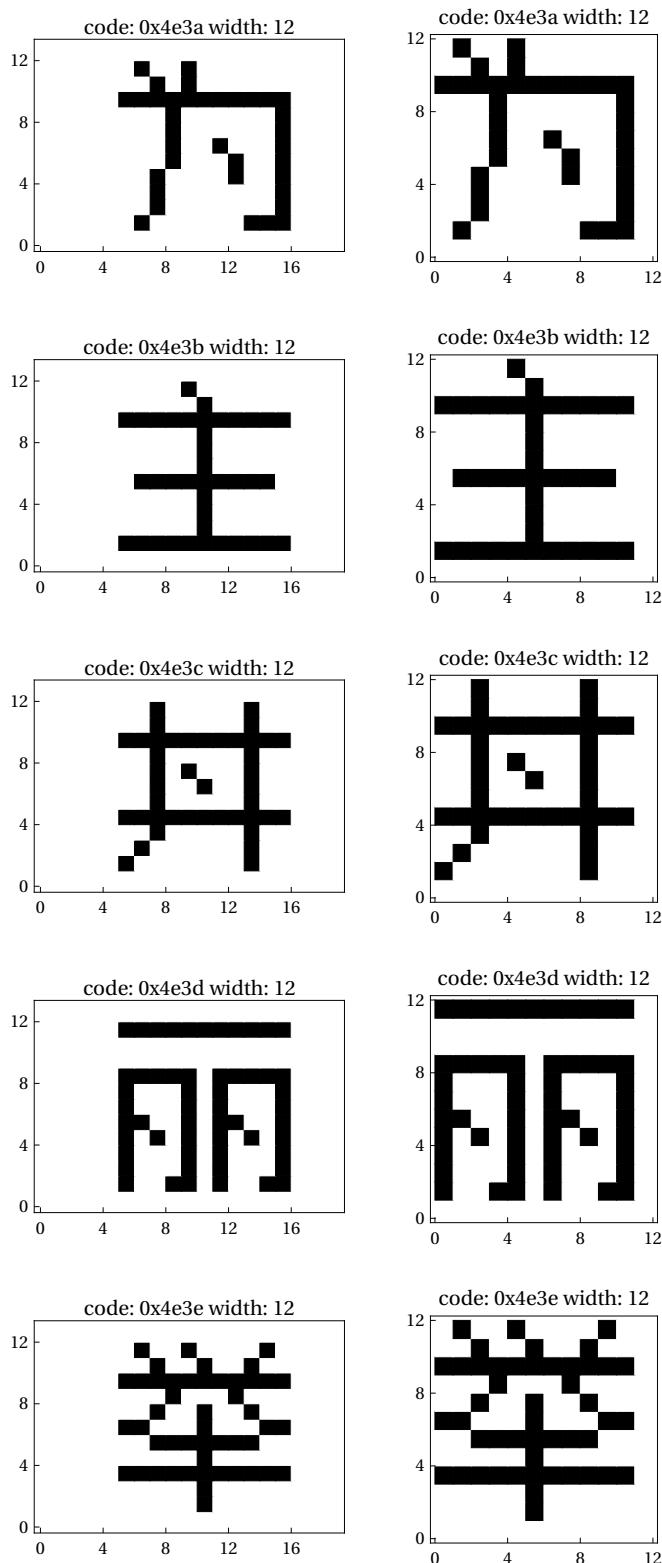


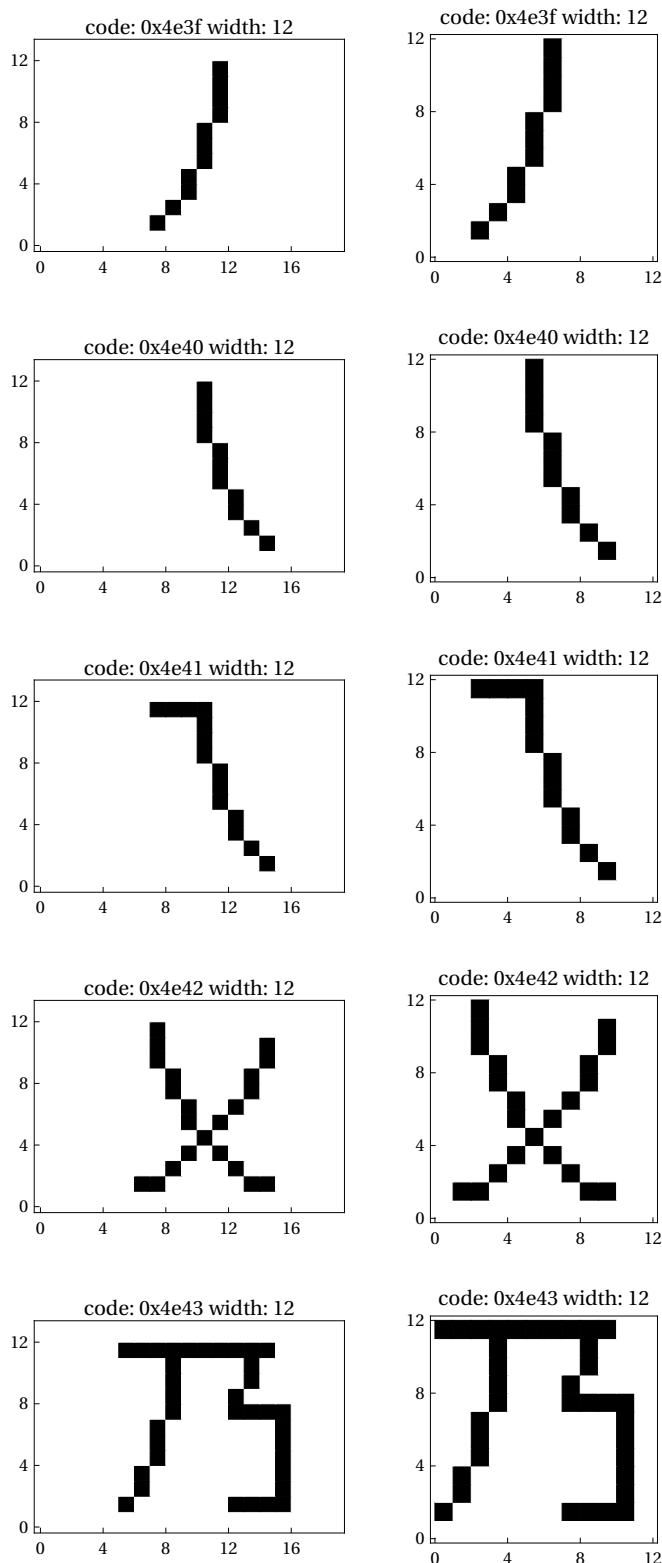


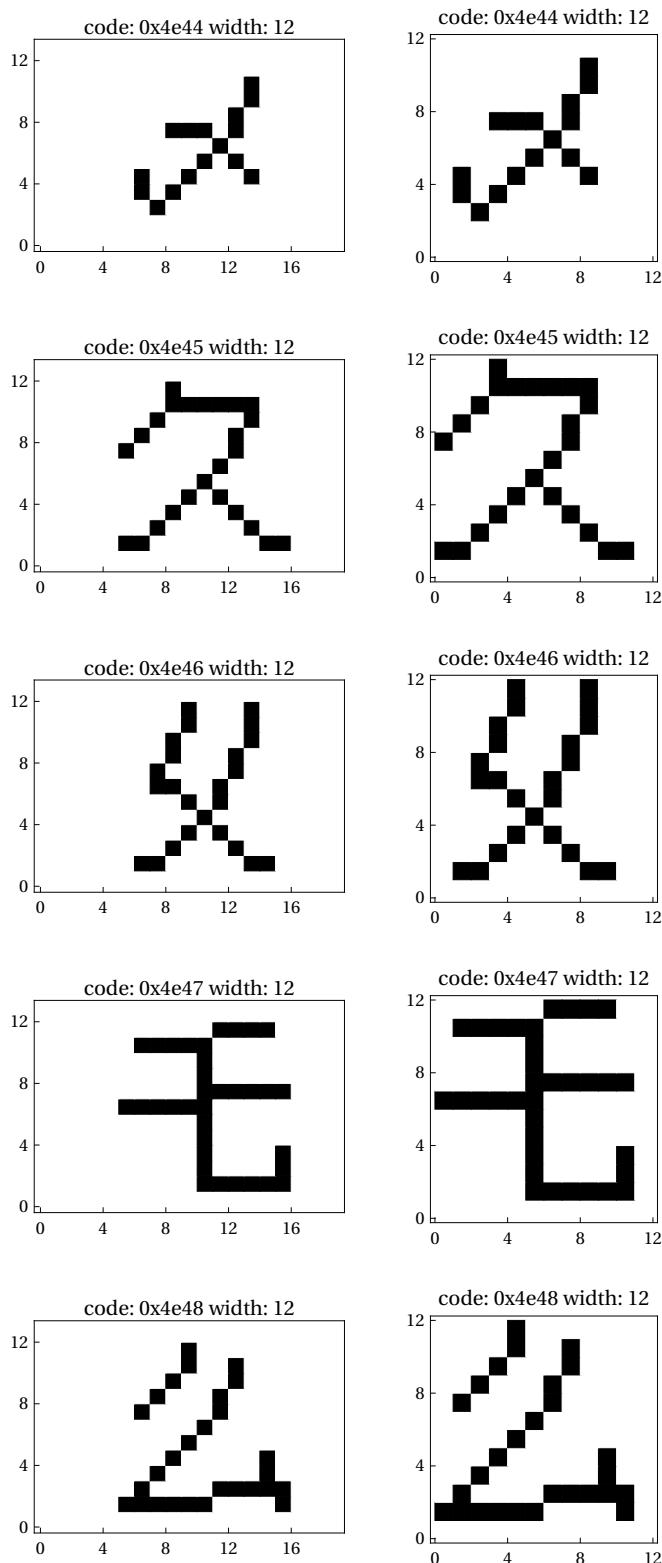


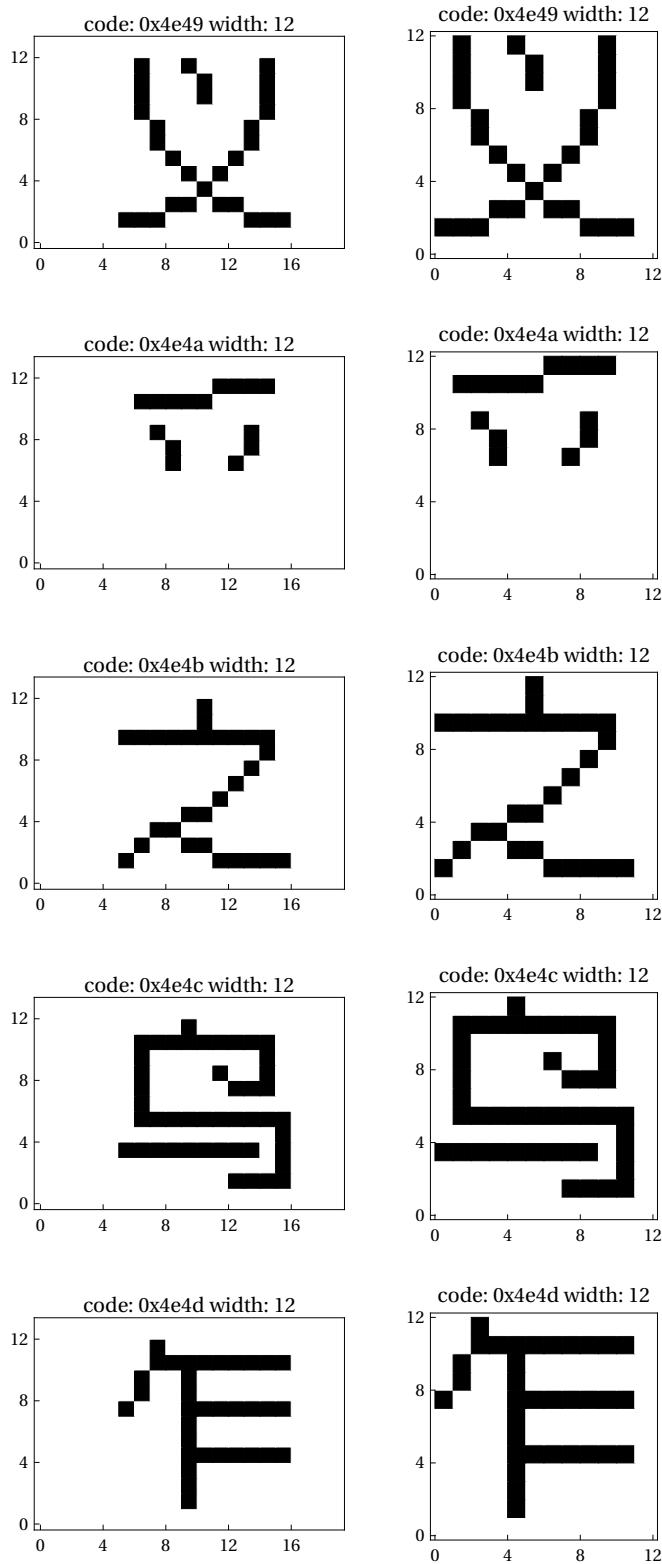


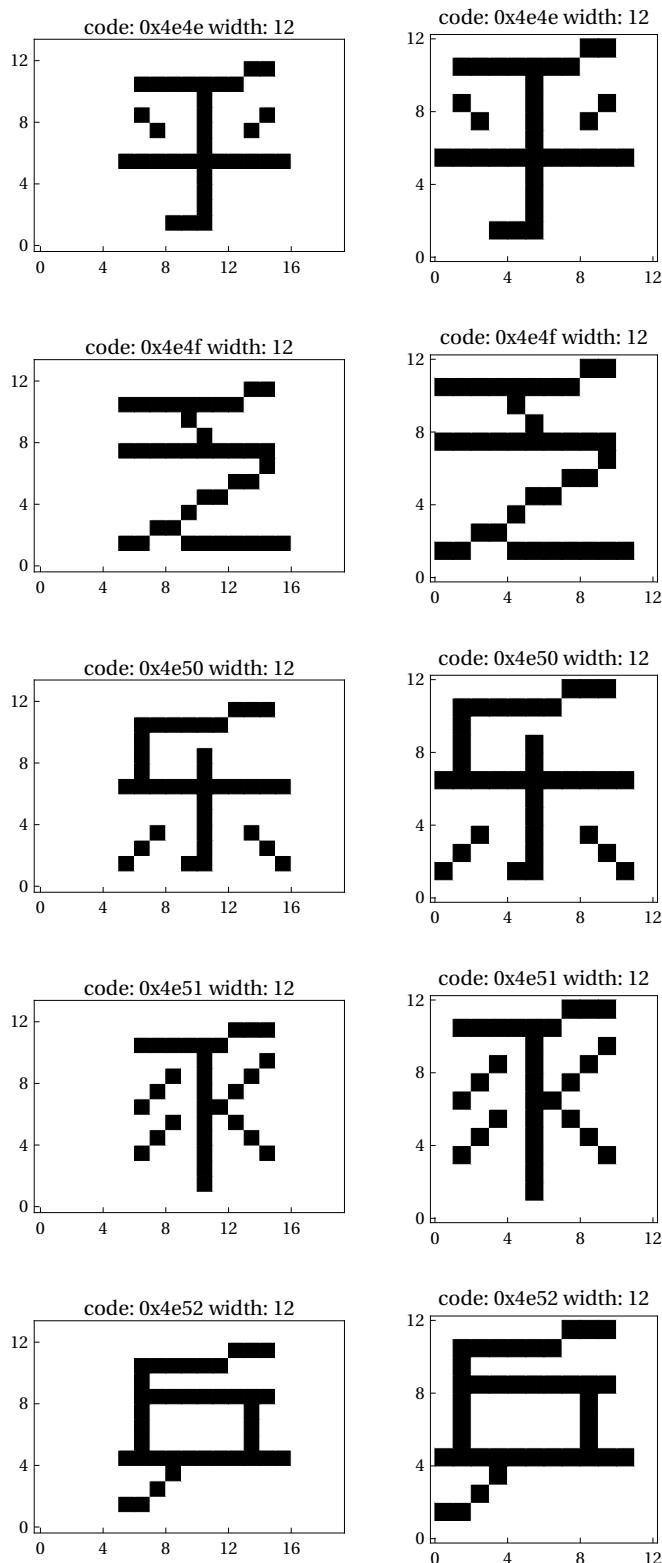


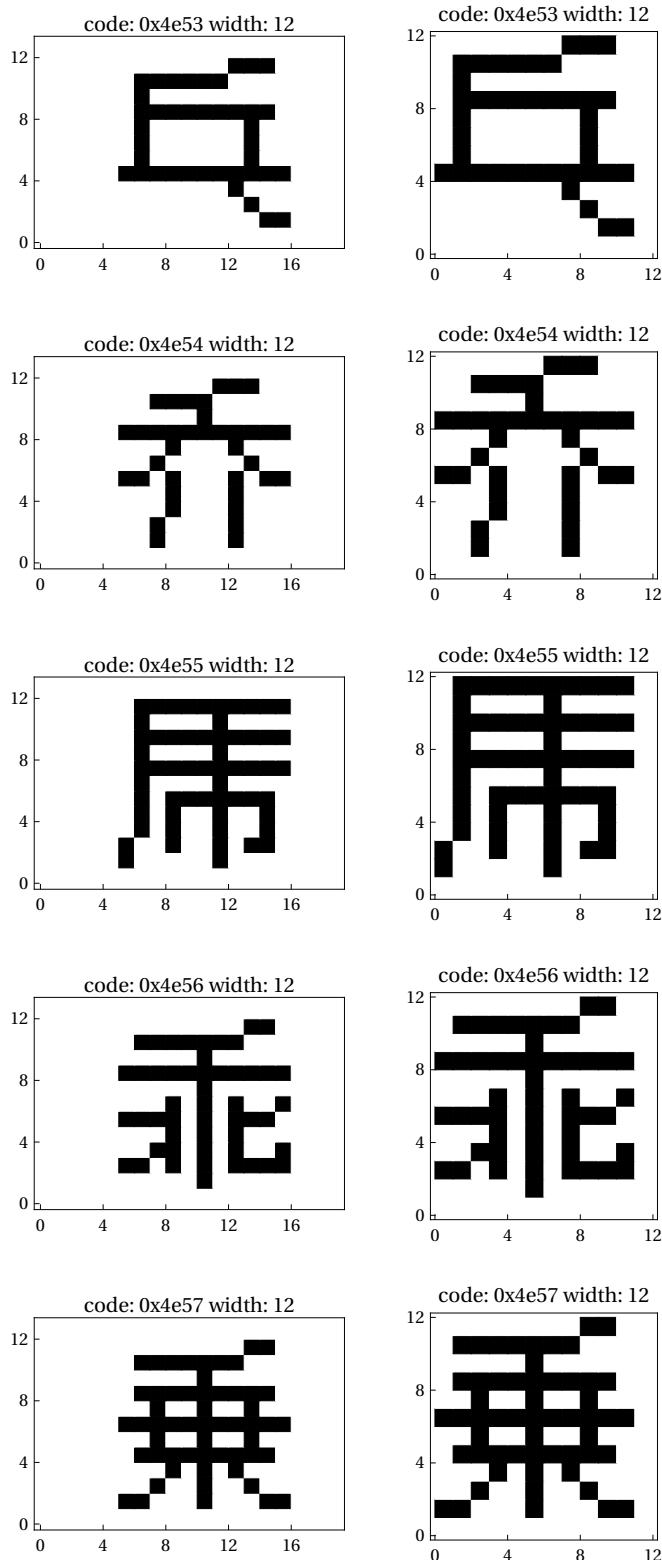


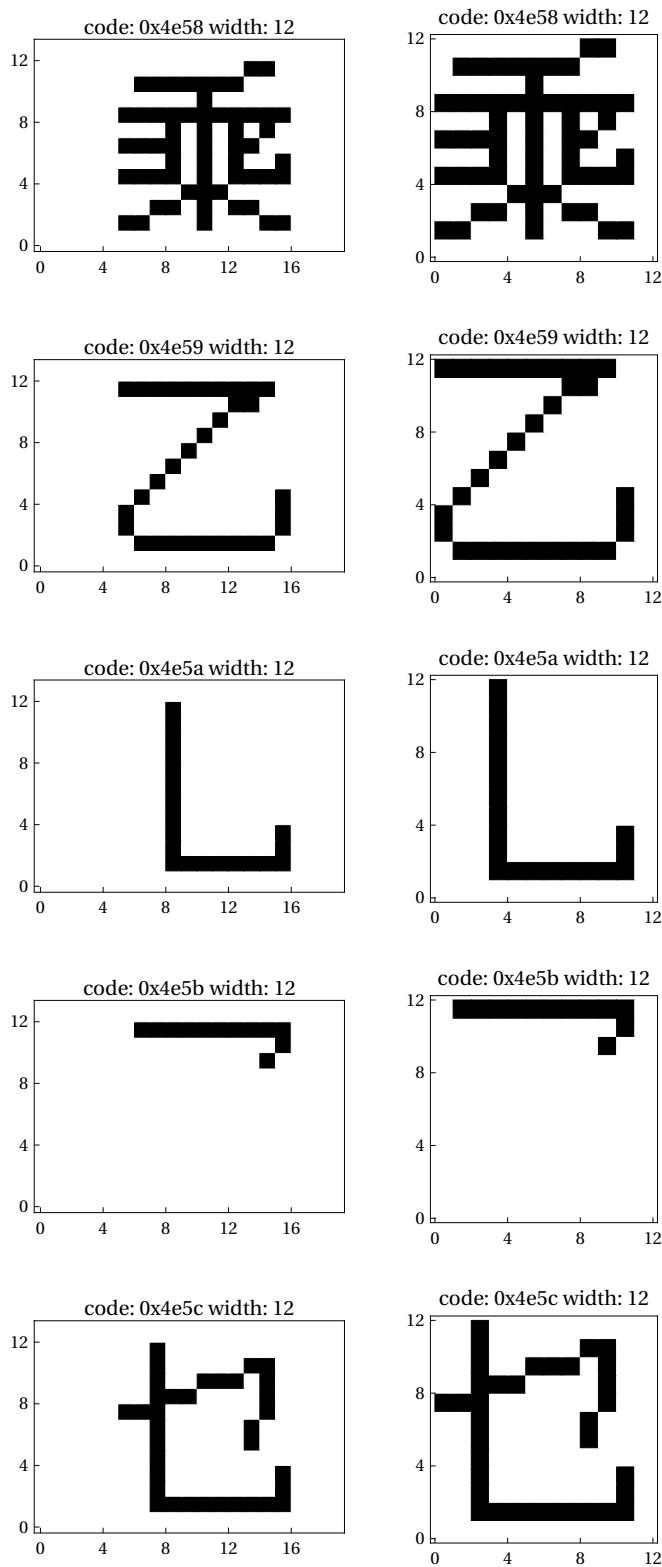


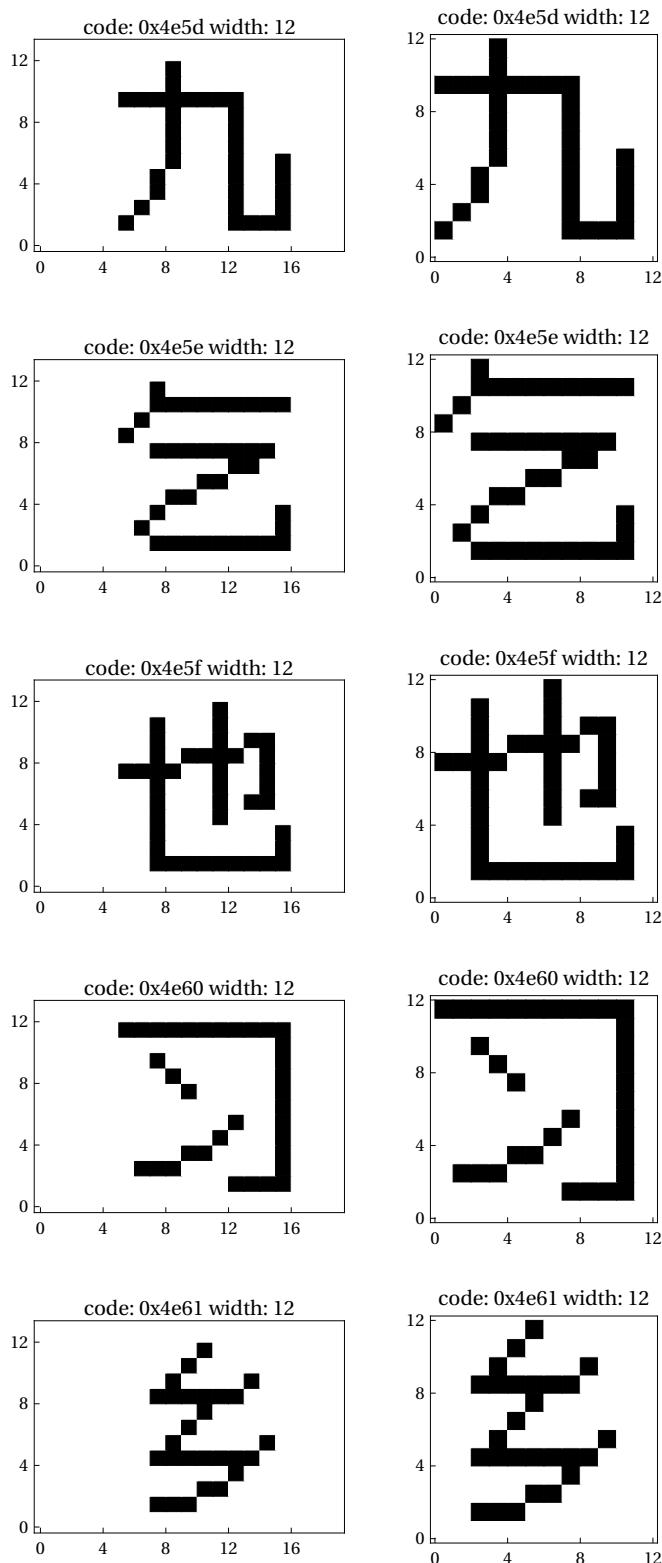


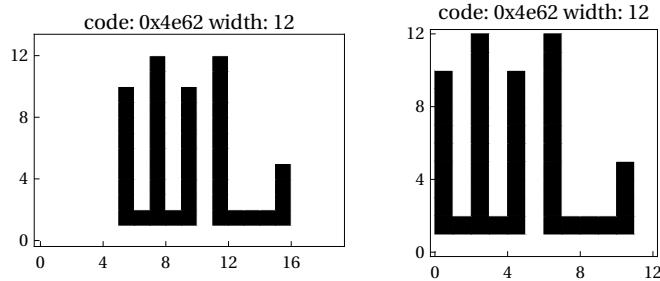












## Extract and sort glyphs by Table of General Standard Chinese Characters

```
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}}
```

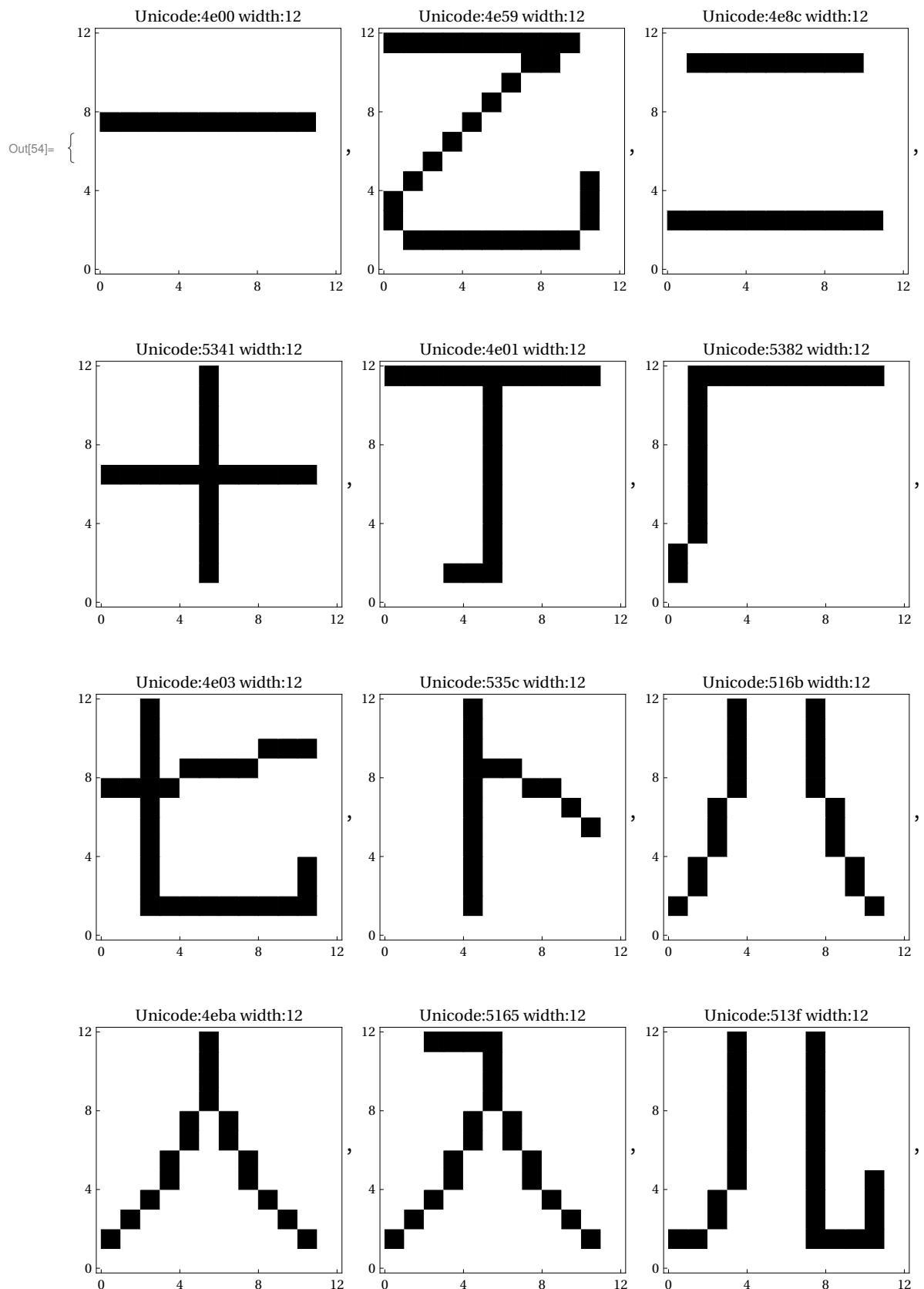
### ■ Check if the TGSCC is contiguous

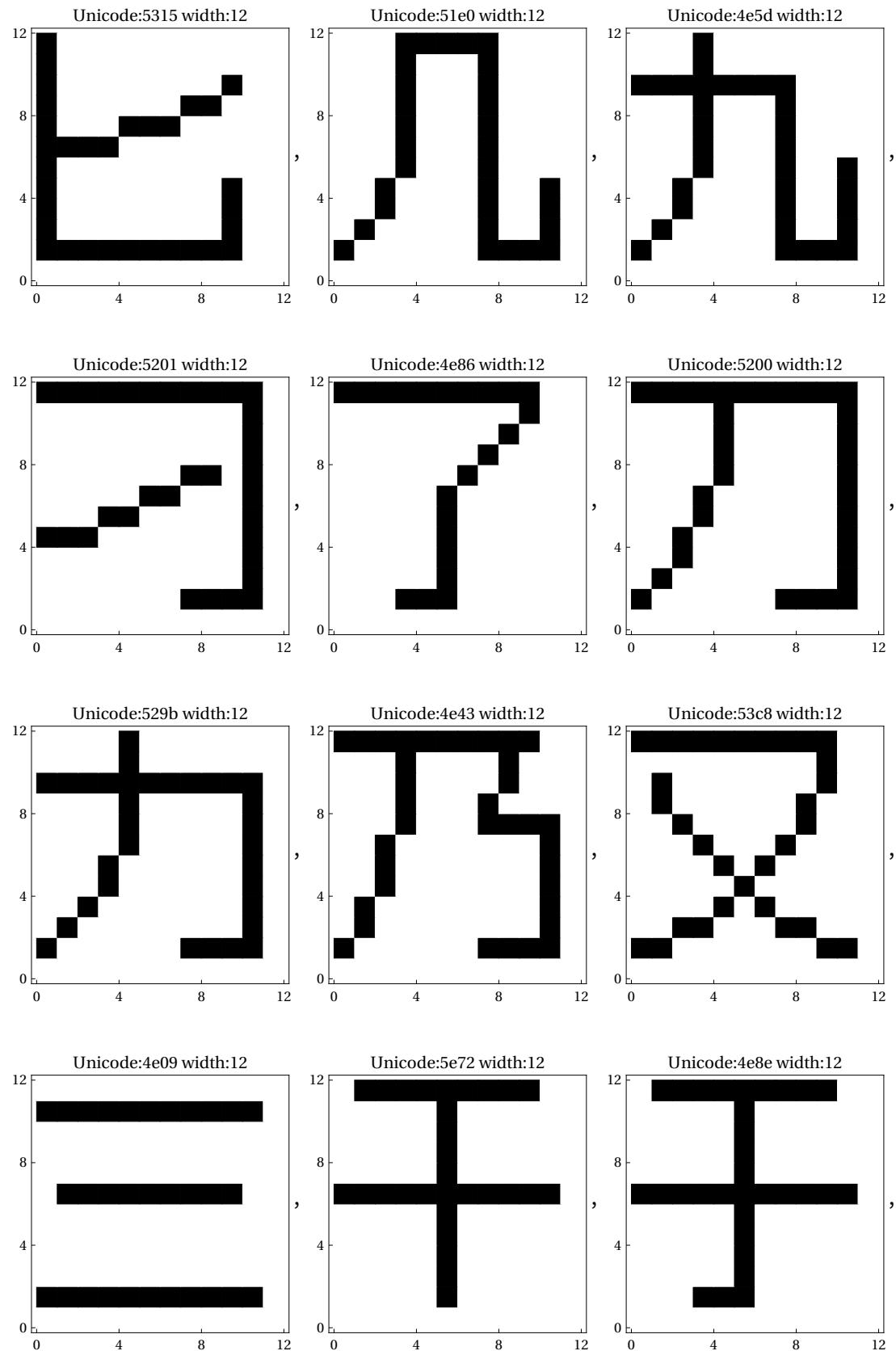
```
In[51]:= {Max[#, Min[#]} &[ListConvolve[{1, -1}, First /@ Drop[rawunicodetable, 2]]]]
```

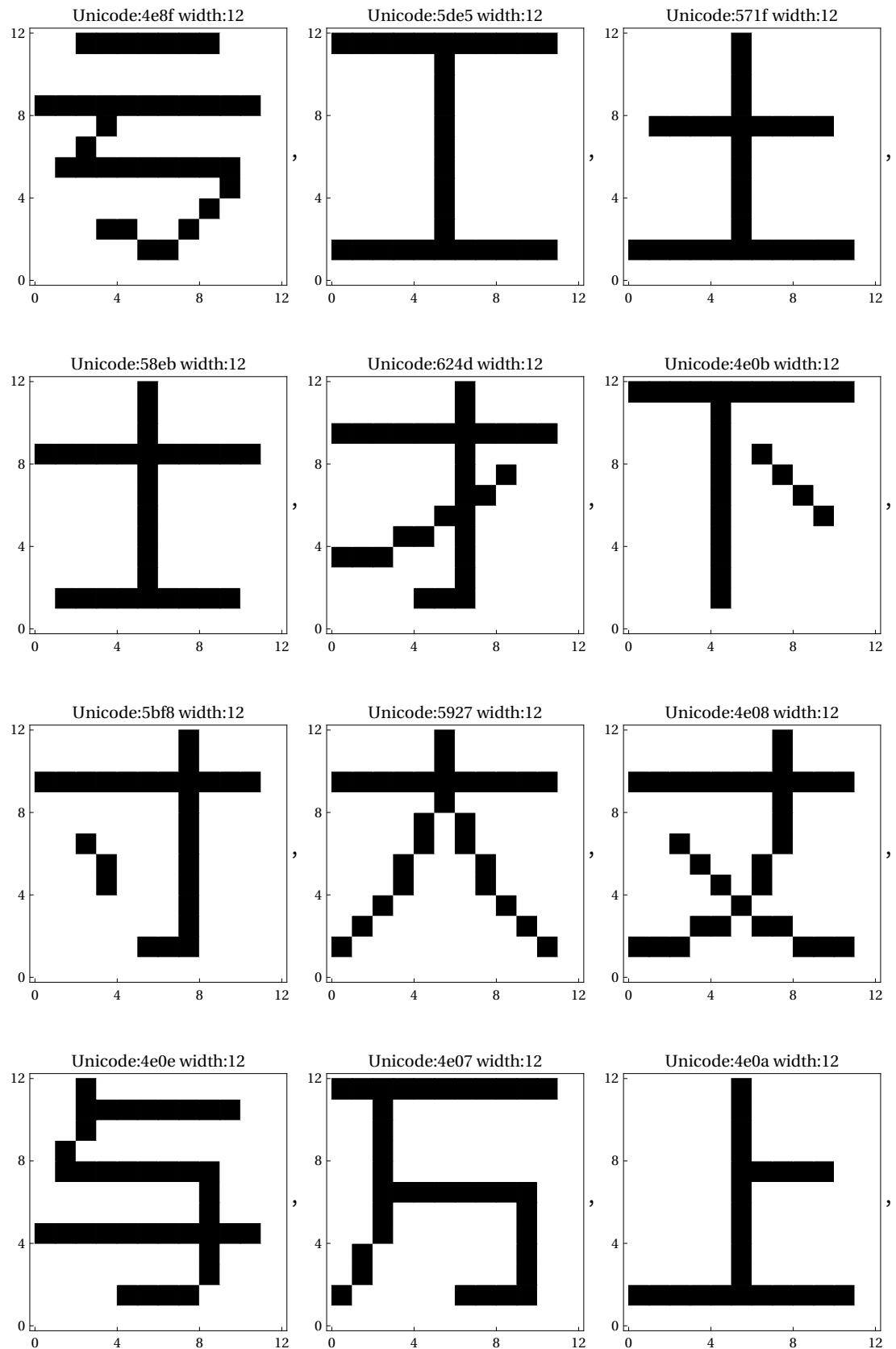
```
Out[51]= {1, 1}
```

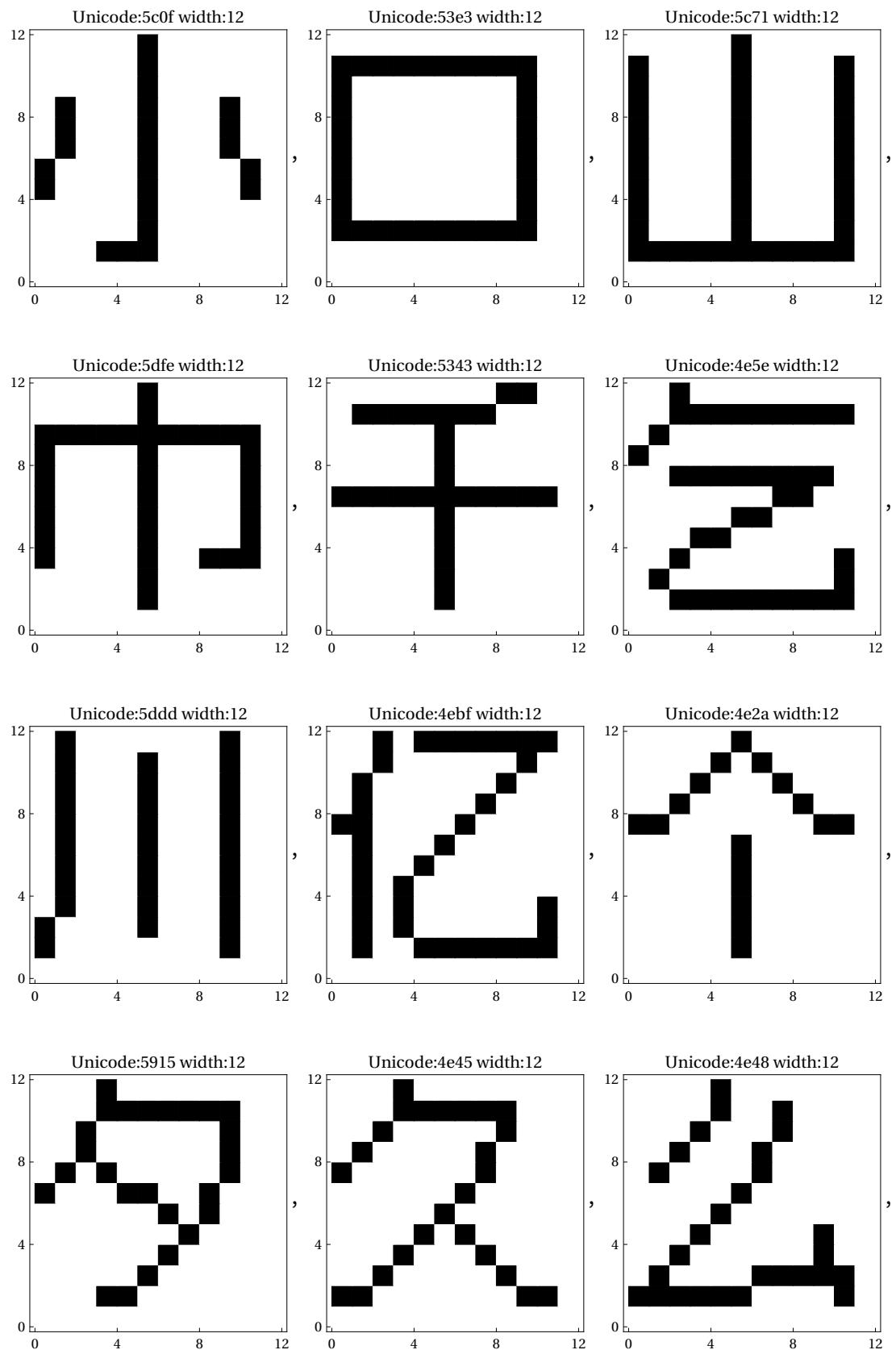
Yes, it is.

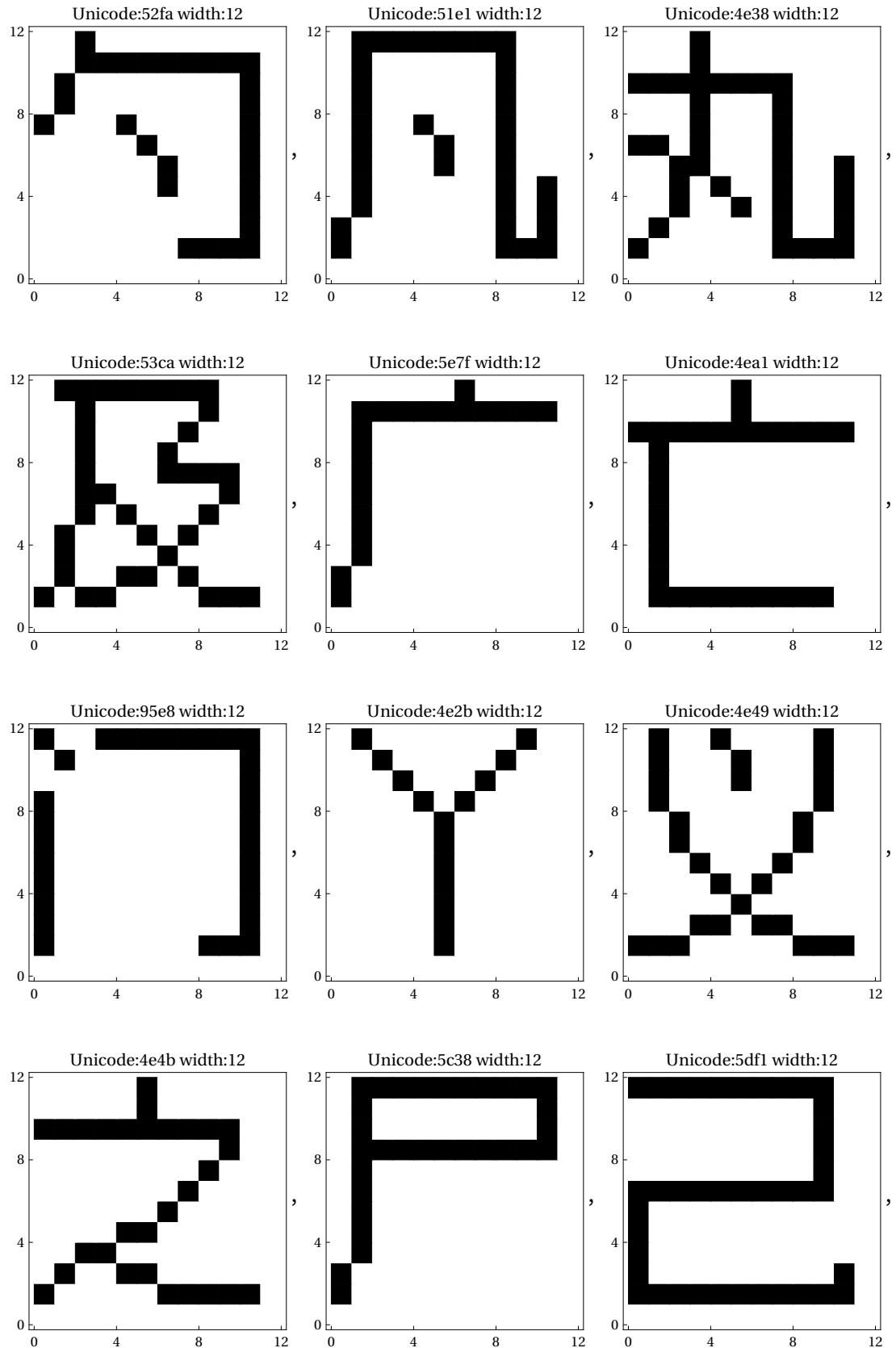
```
In[52]:= Take[TGSCCcodes = ToExpression[StringReplace[Last[#], "U+" → "16^&"]] & /@
Drop[rawunicodetable, 2], 16]
Out[52]= {19 968, 20 057, 20 108, 21 313, 19 969, 21 378, 19 971,
21 340, 20 843, 20 154, 20 837, 20 799, 21 269, 20 960, 20 061, 20 993}
In[53]:= Dimensions[
lookupable = Association[MapThread[Function[{code, 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[lookupable, #], #] & /@ TGSCCcodes, 1500]
```

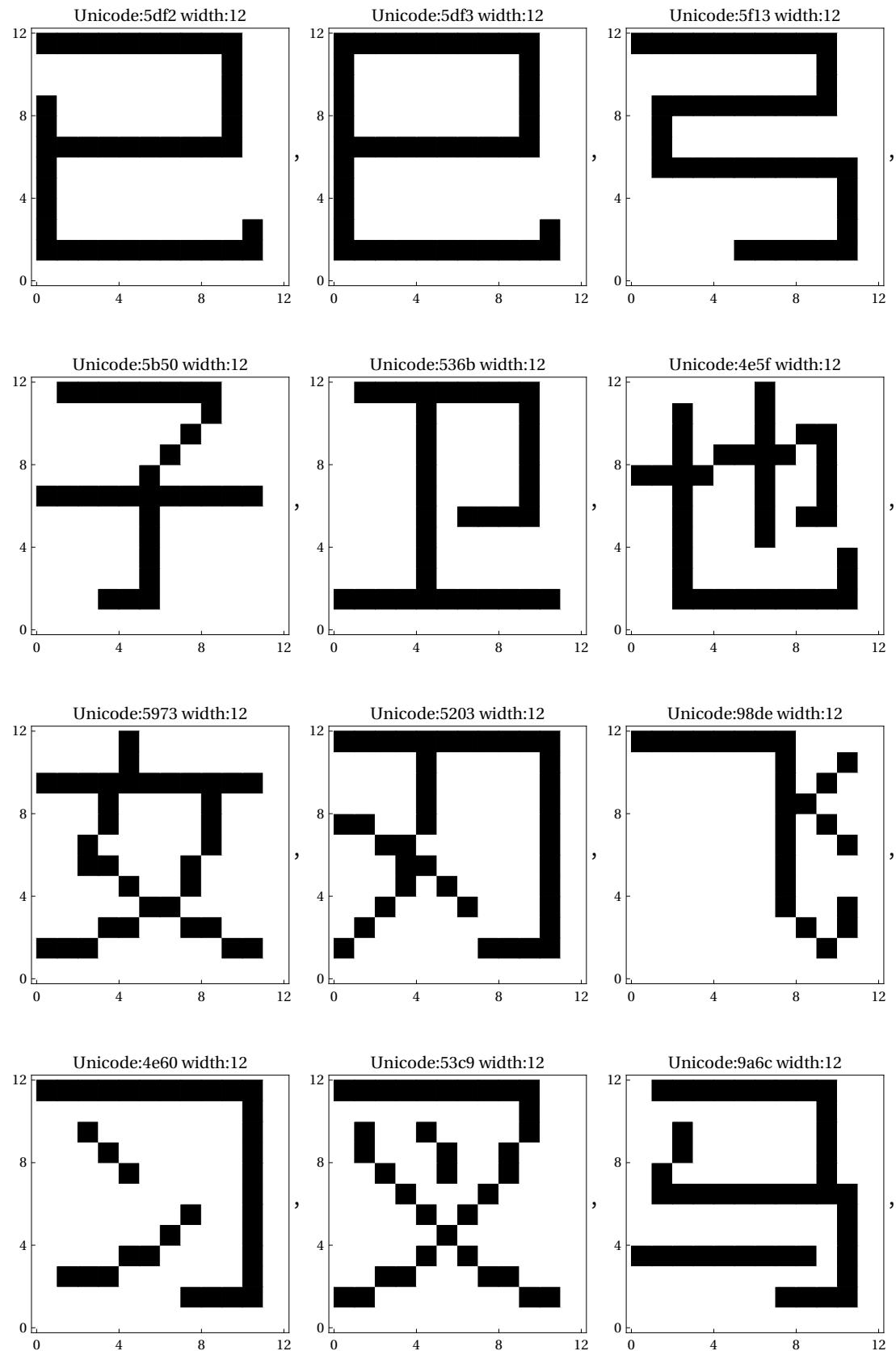


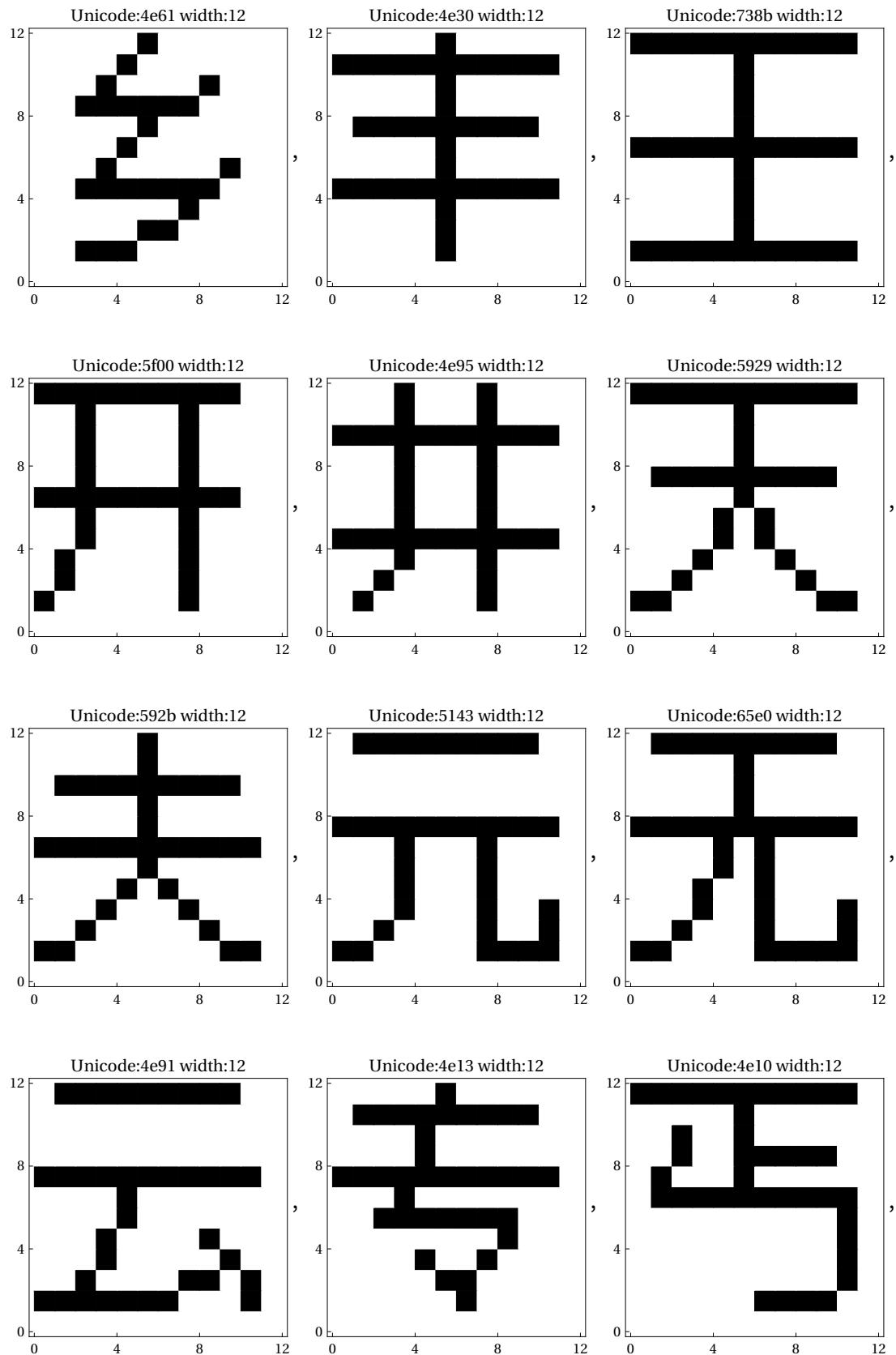


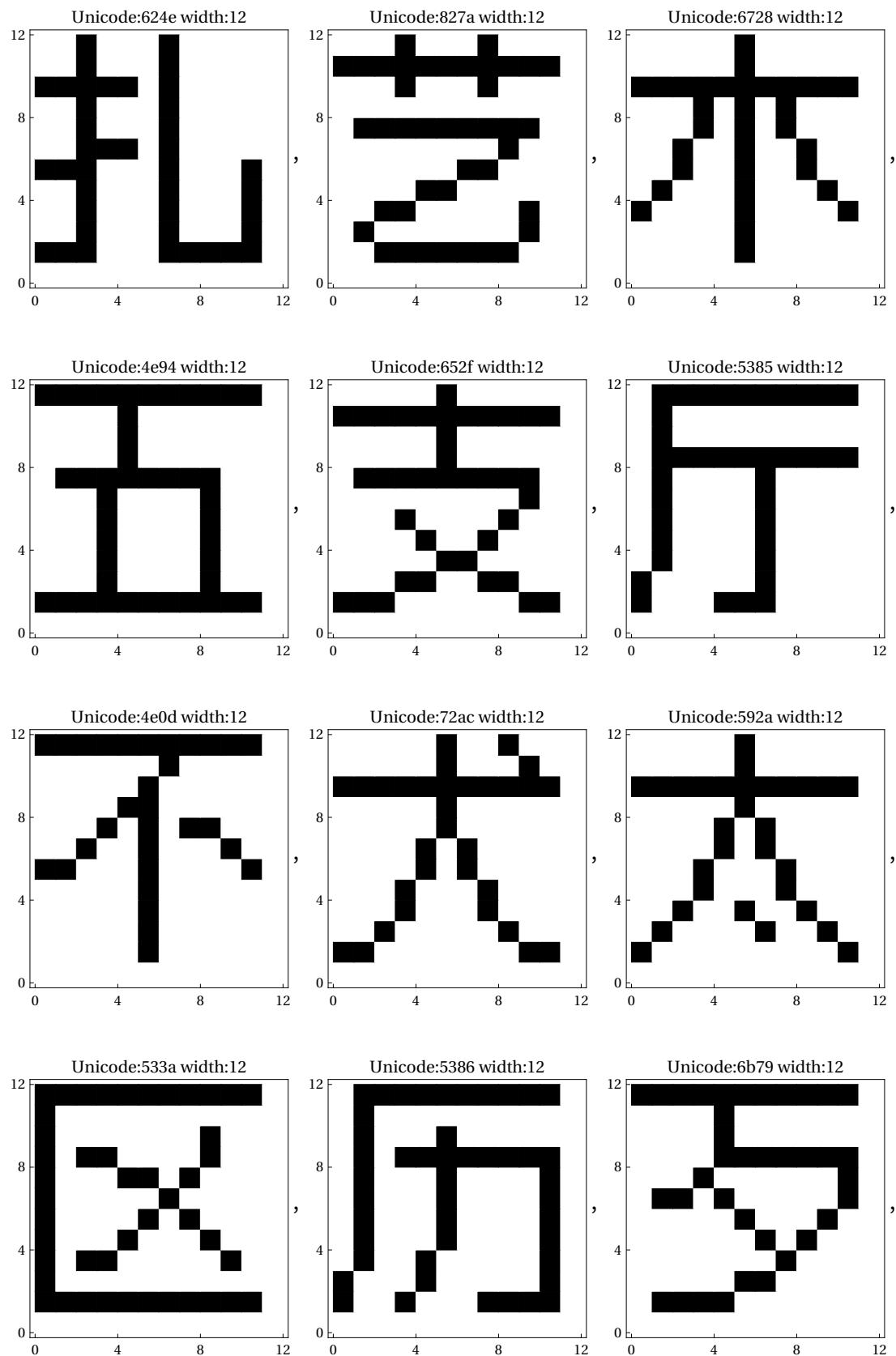


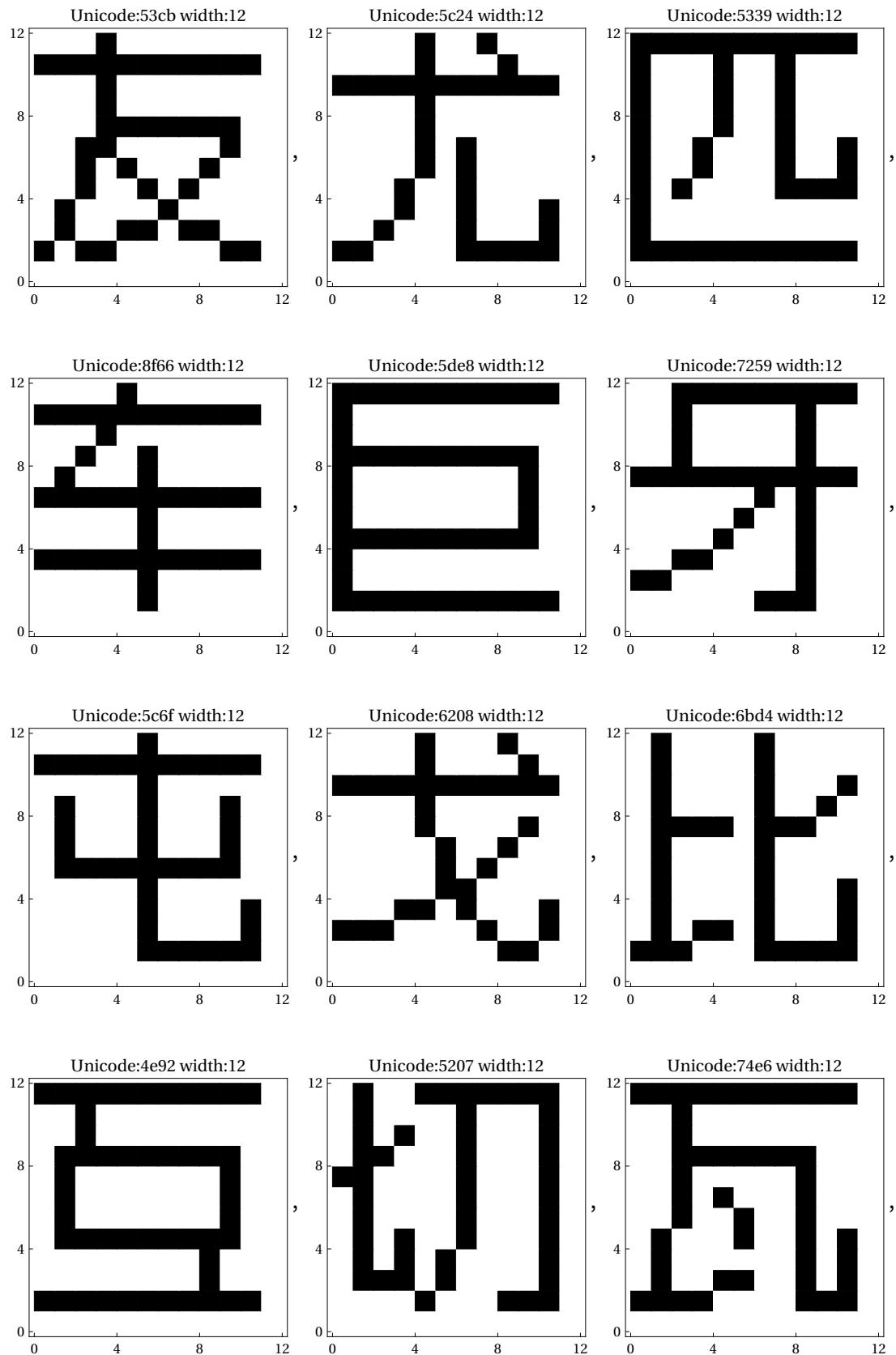


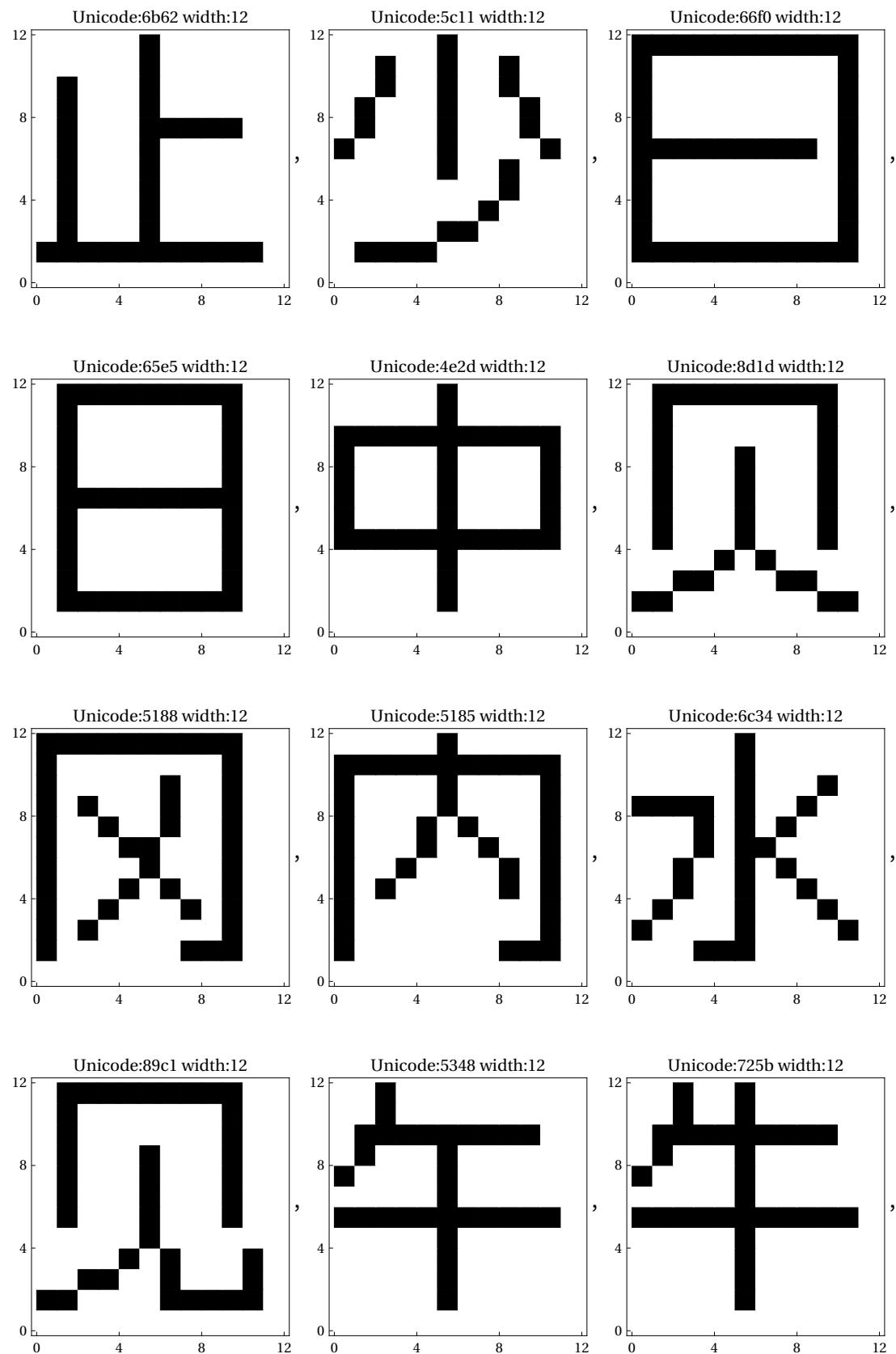


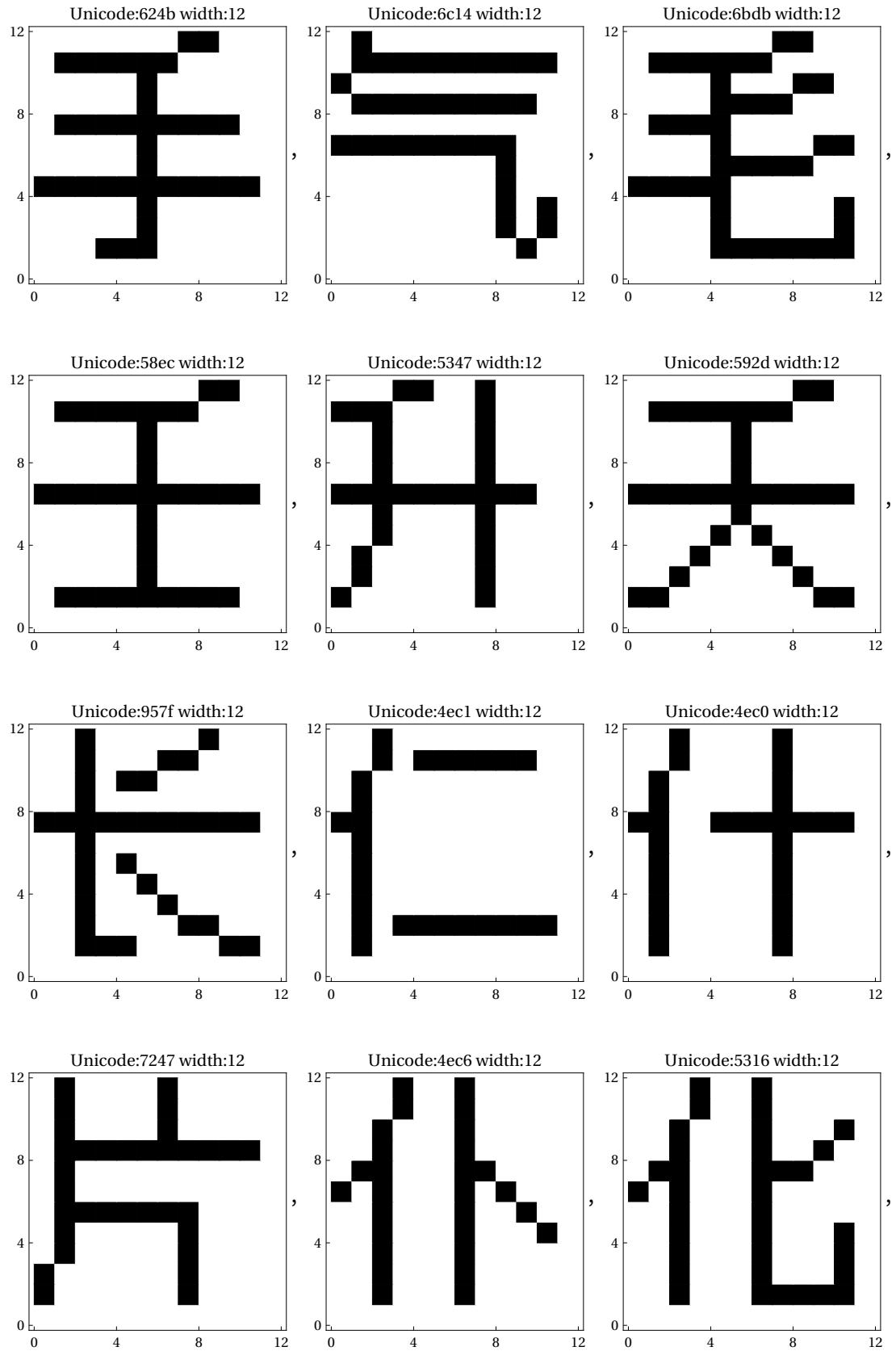


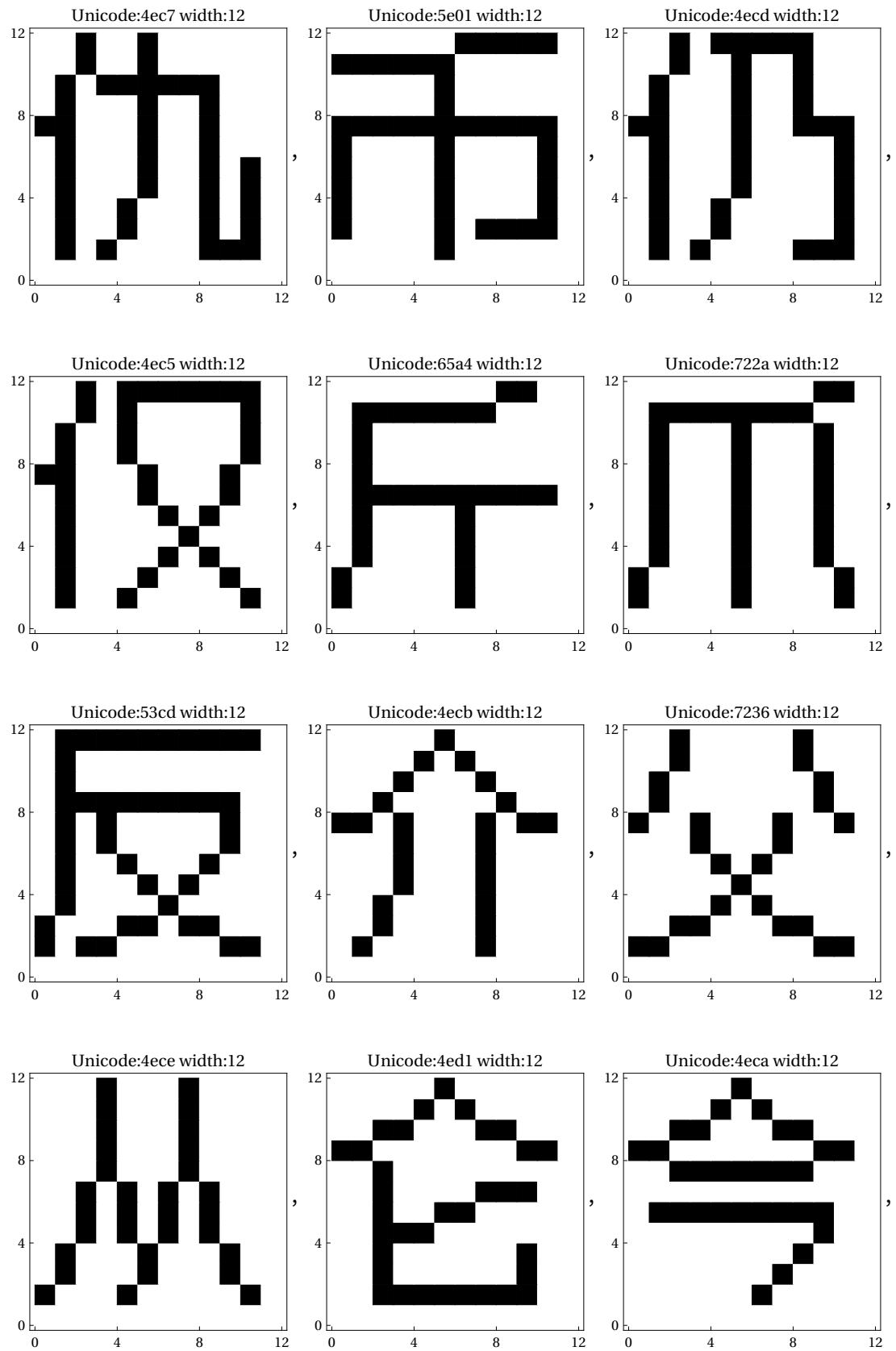


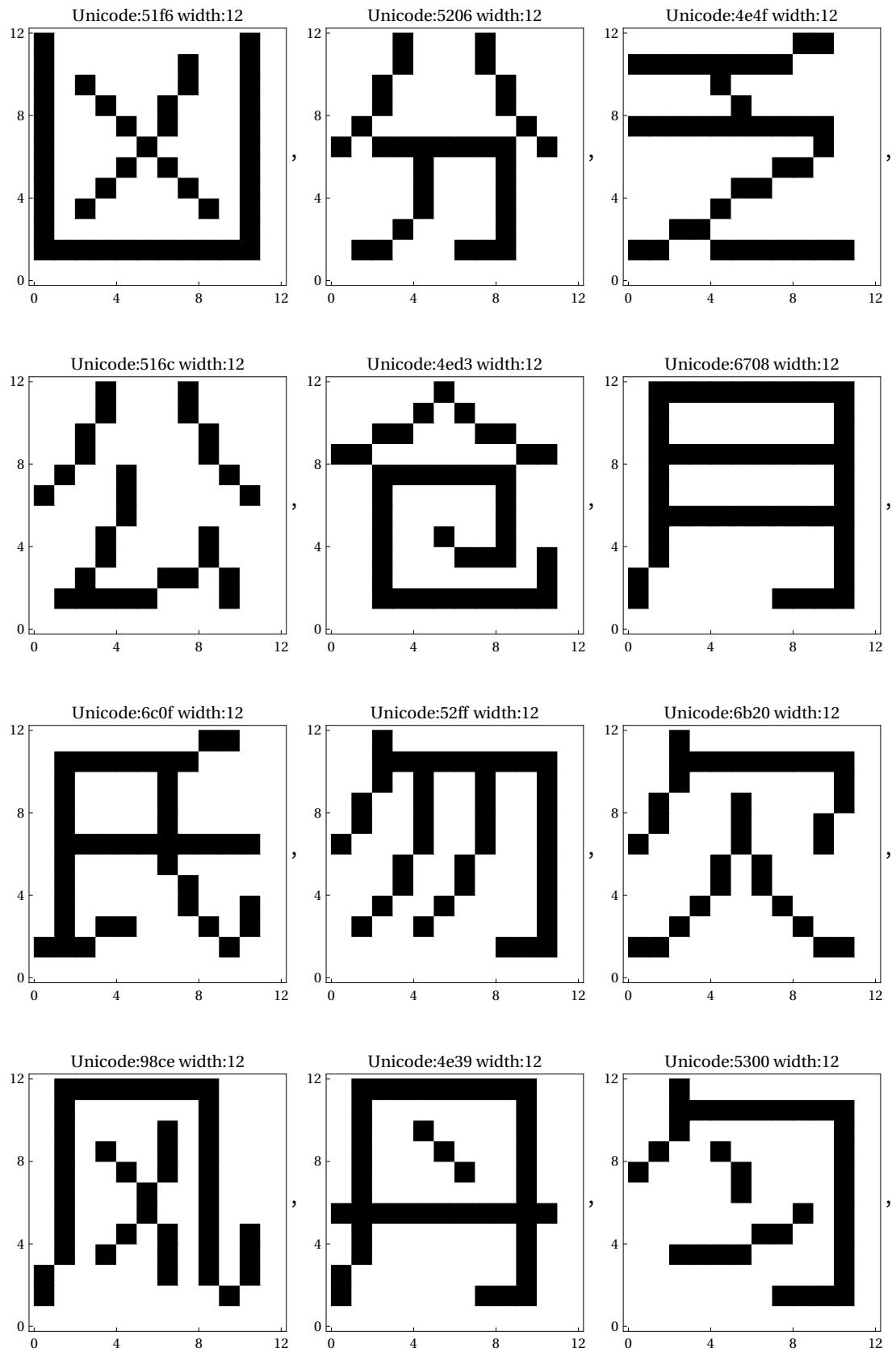


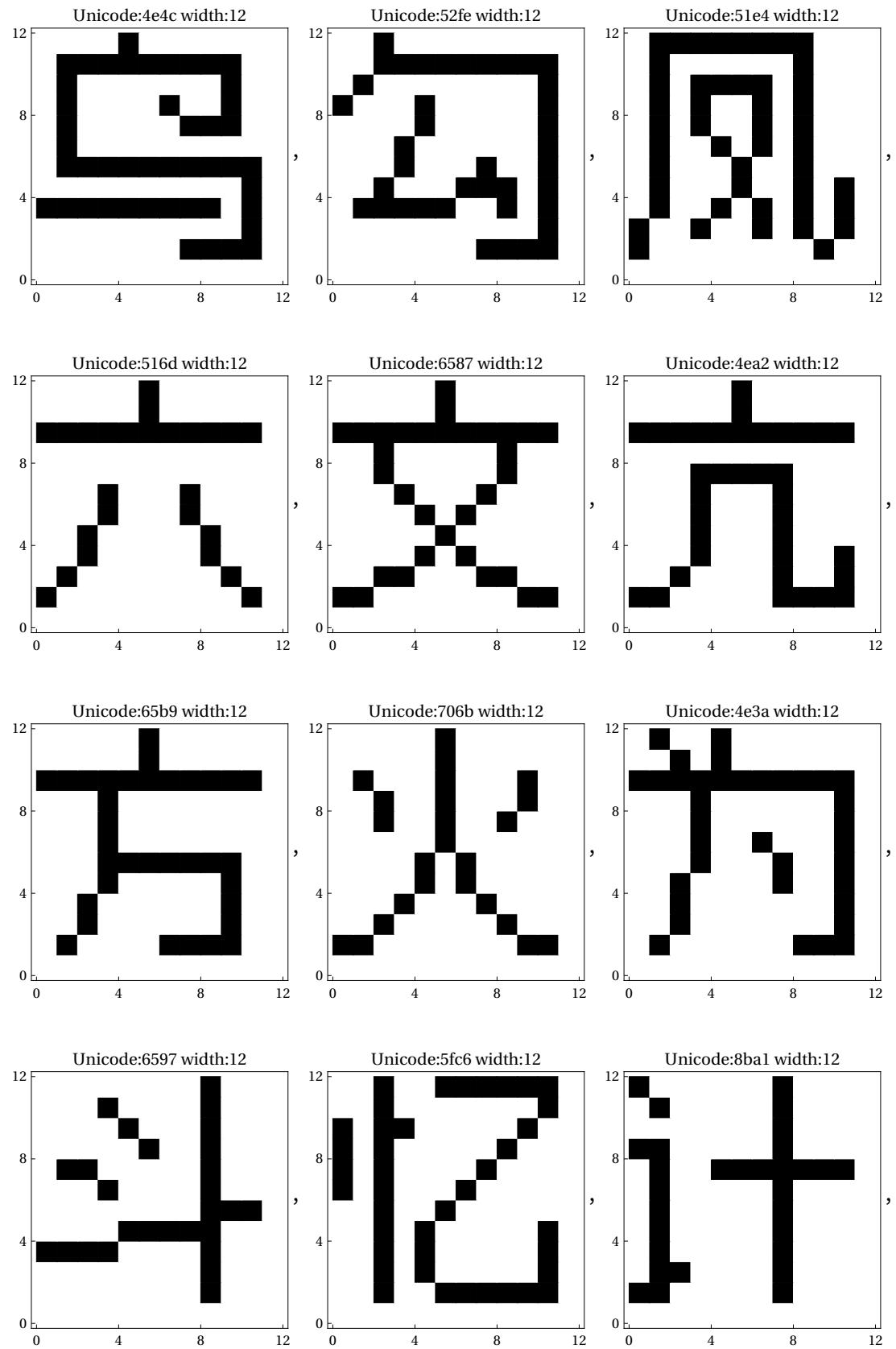


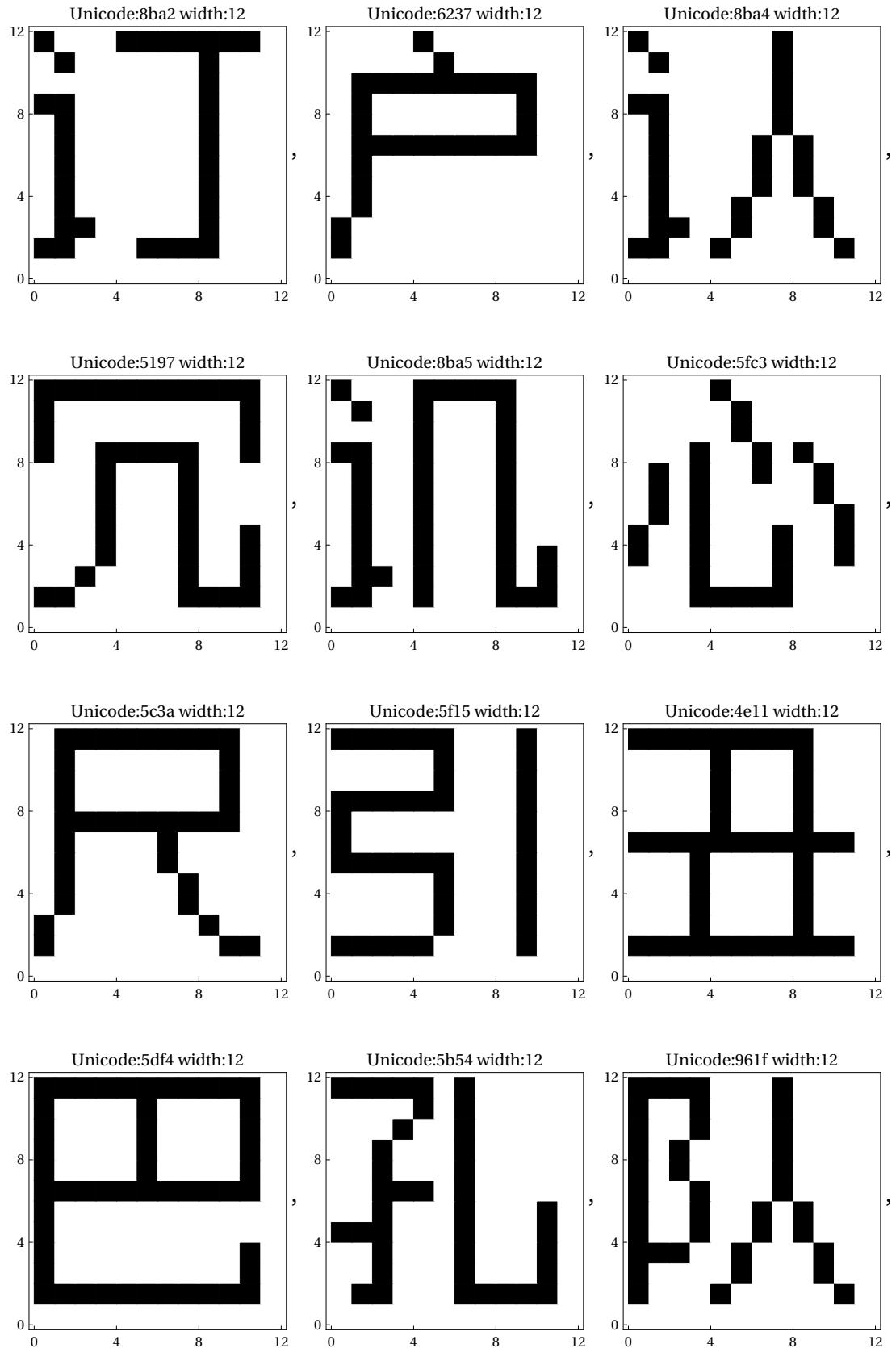


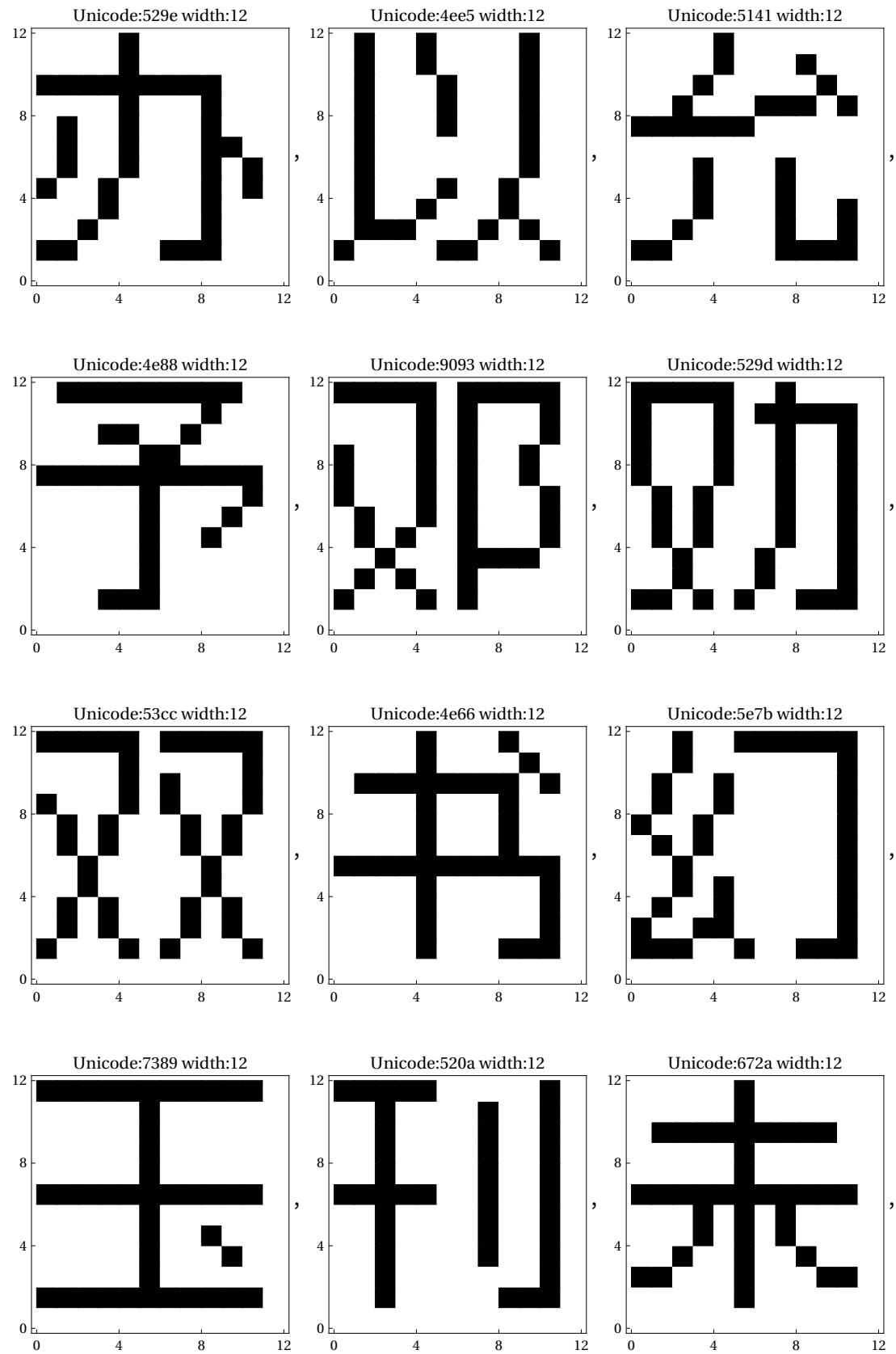


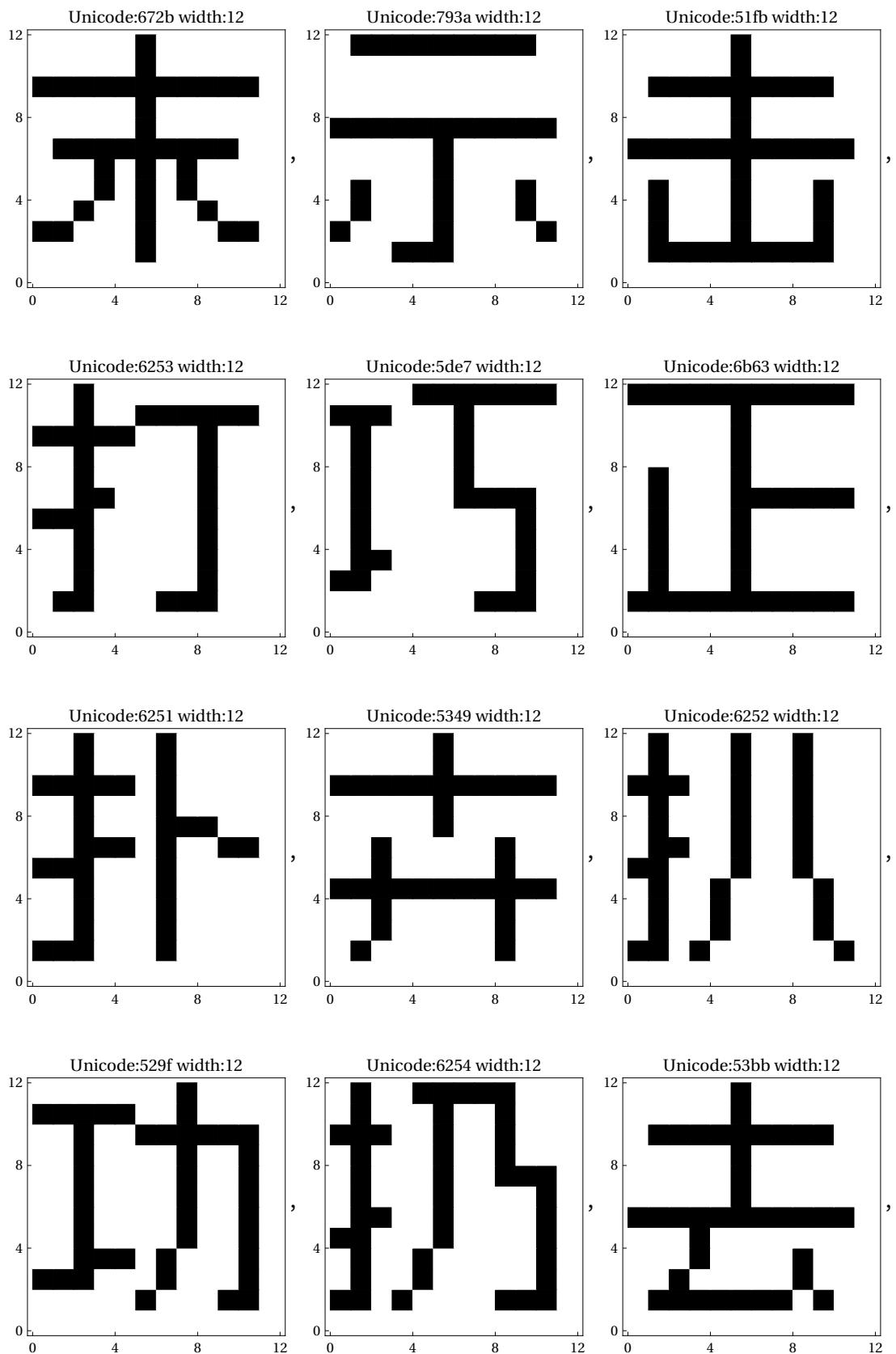


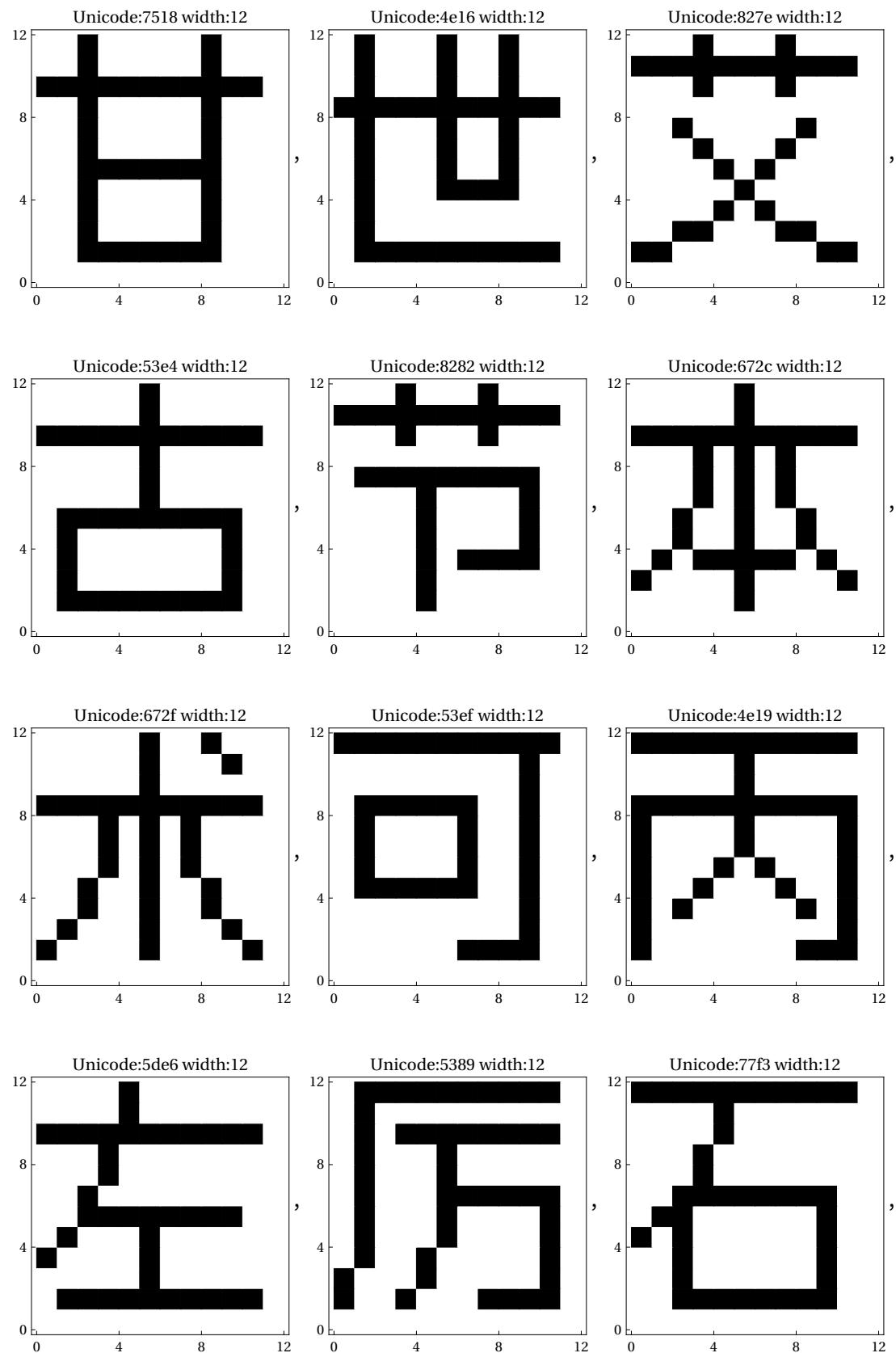


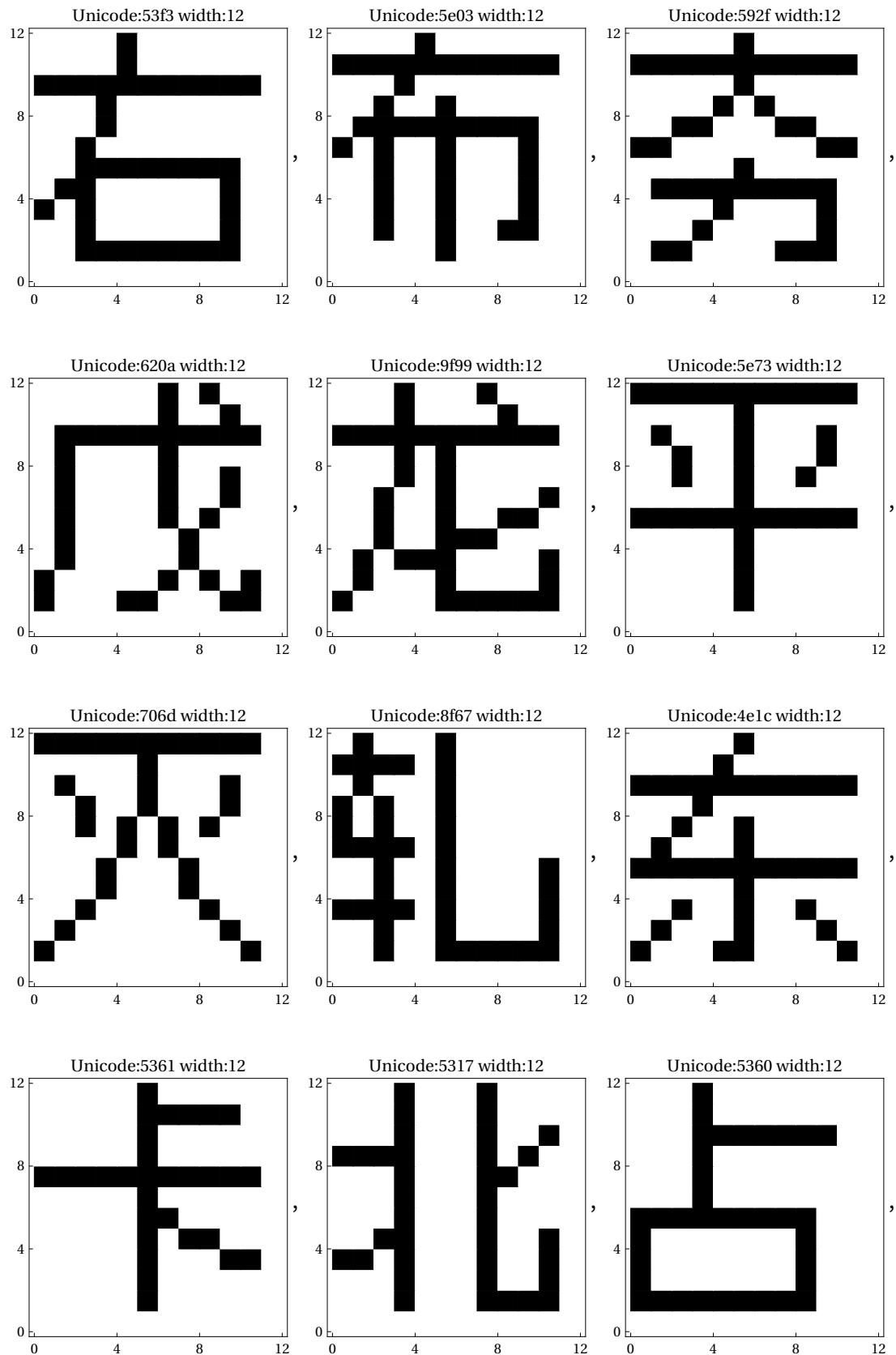


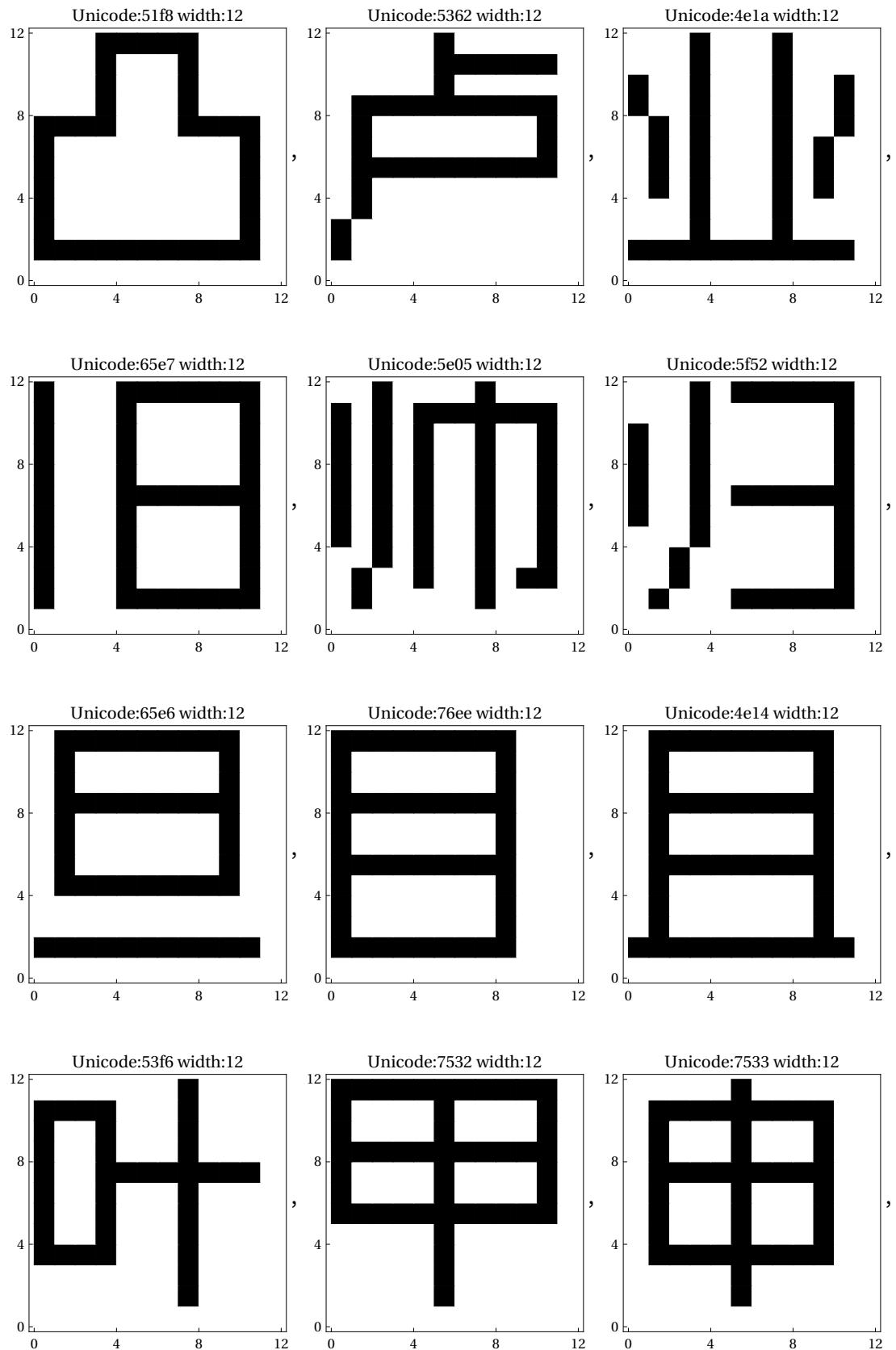


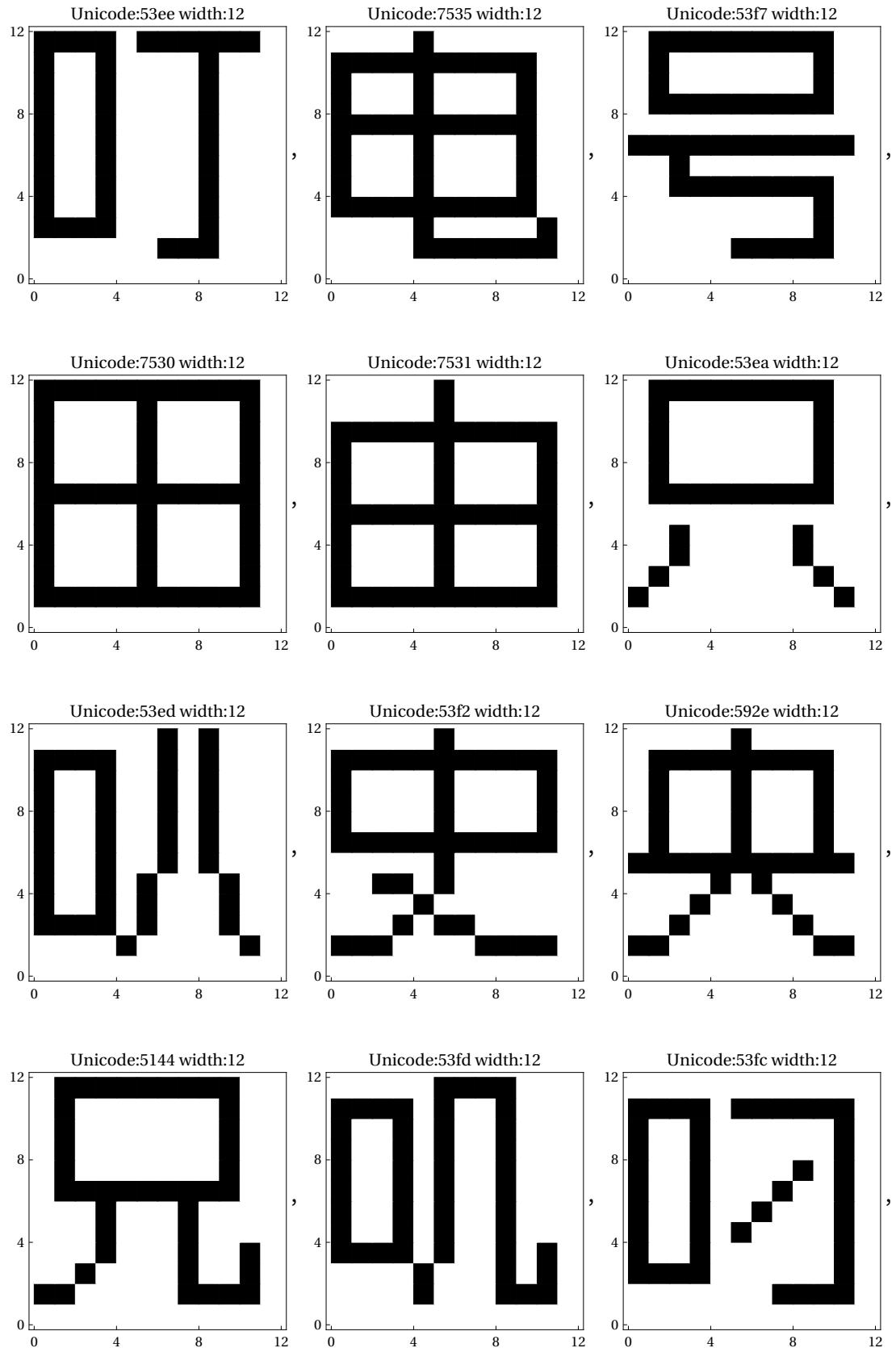


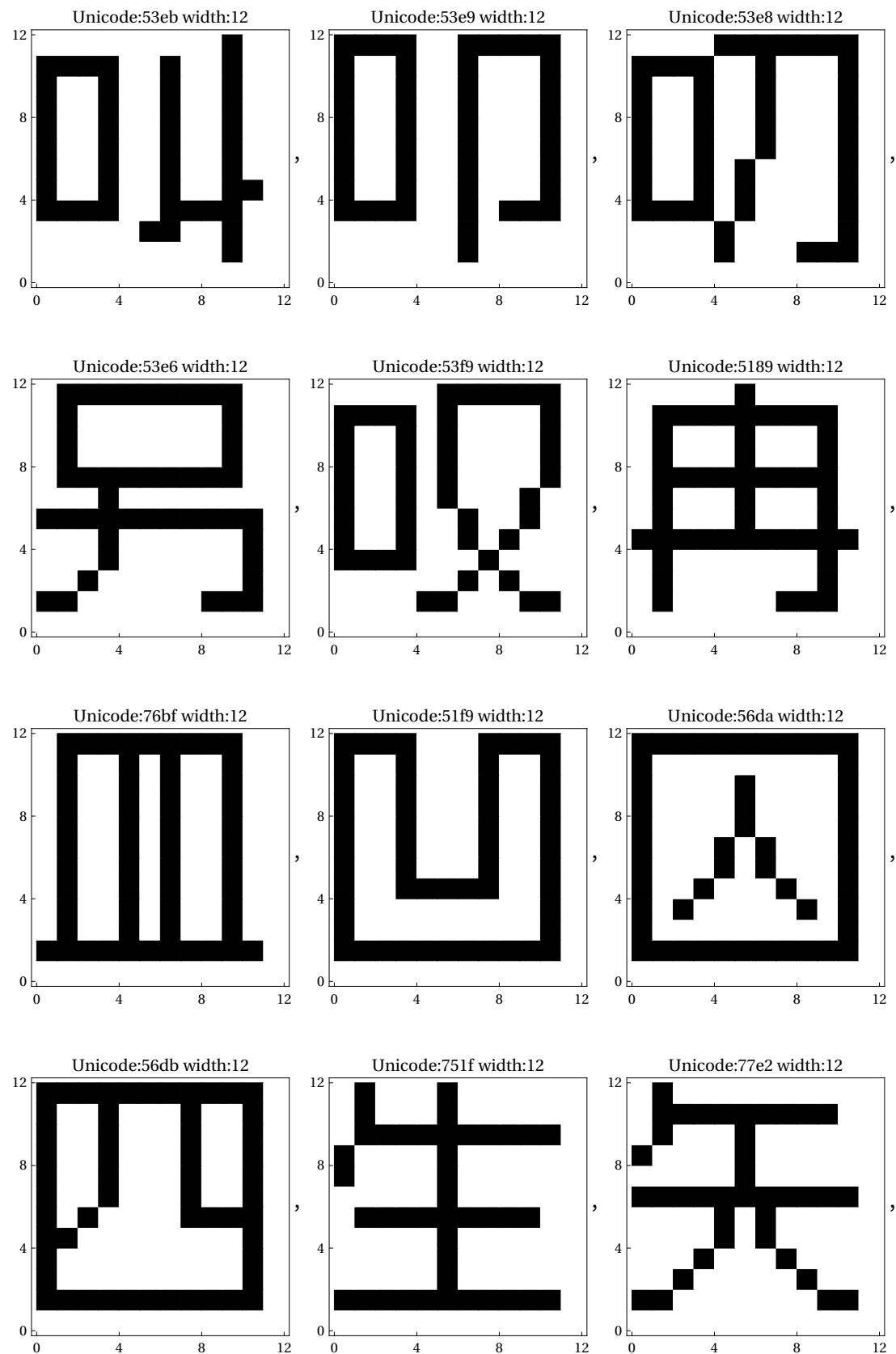


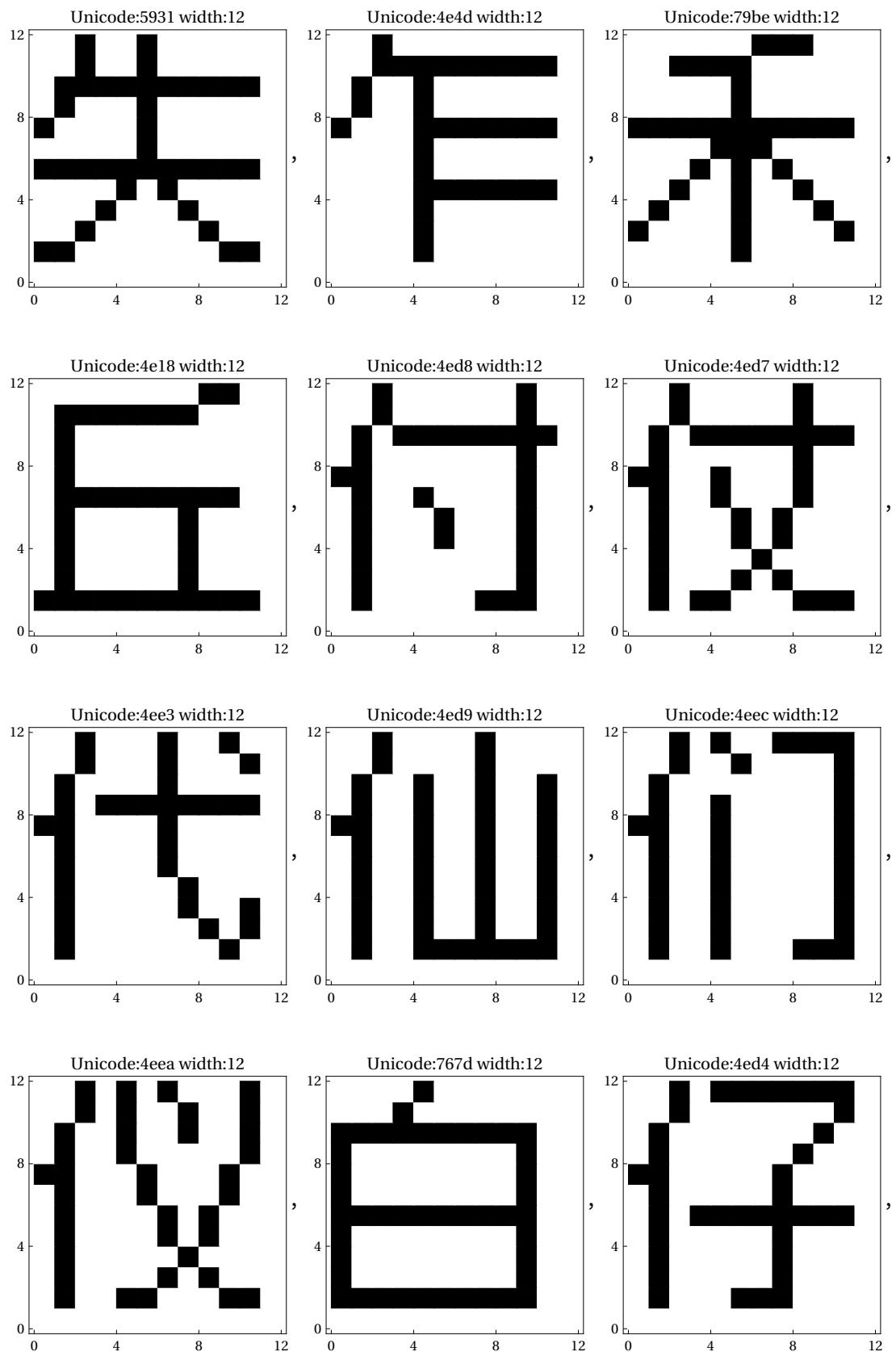


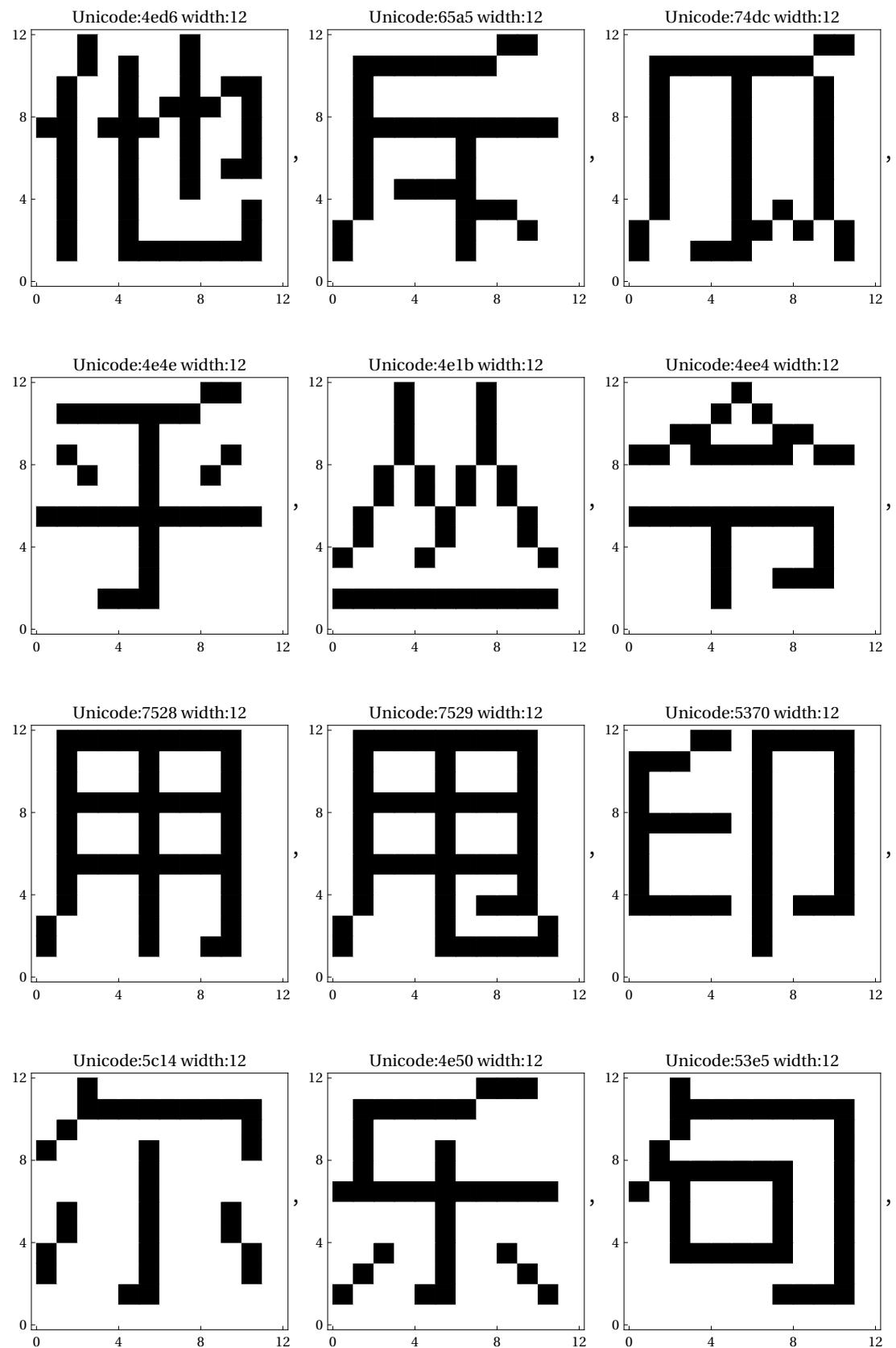


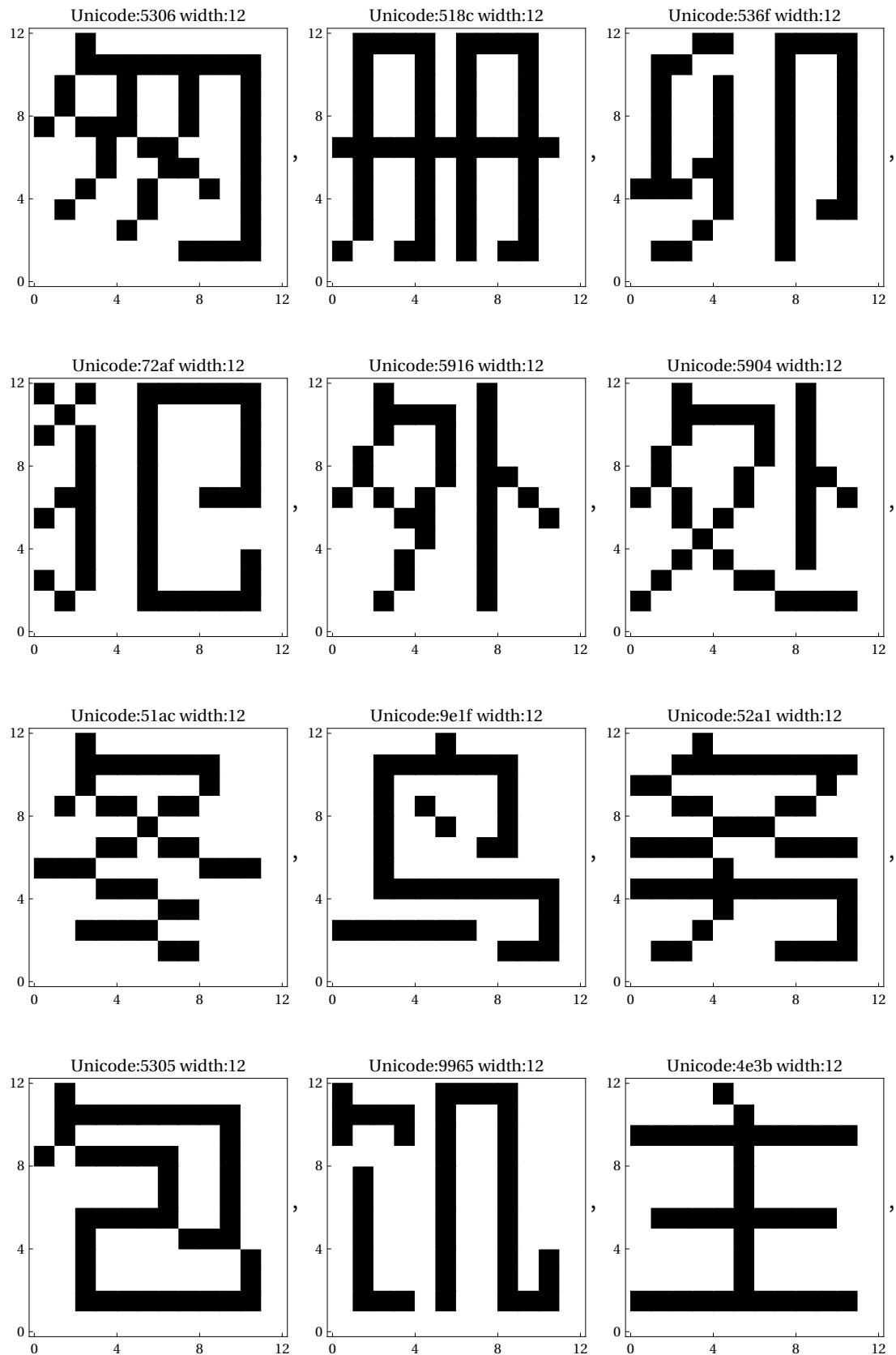


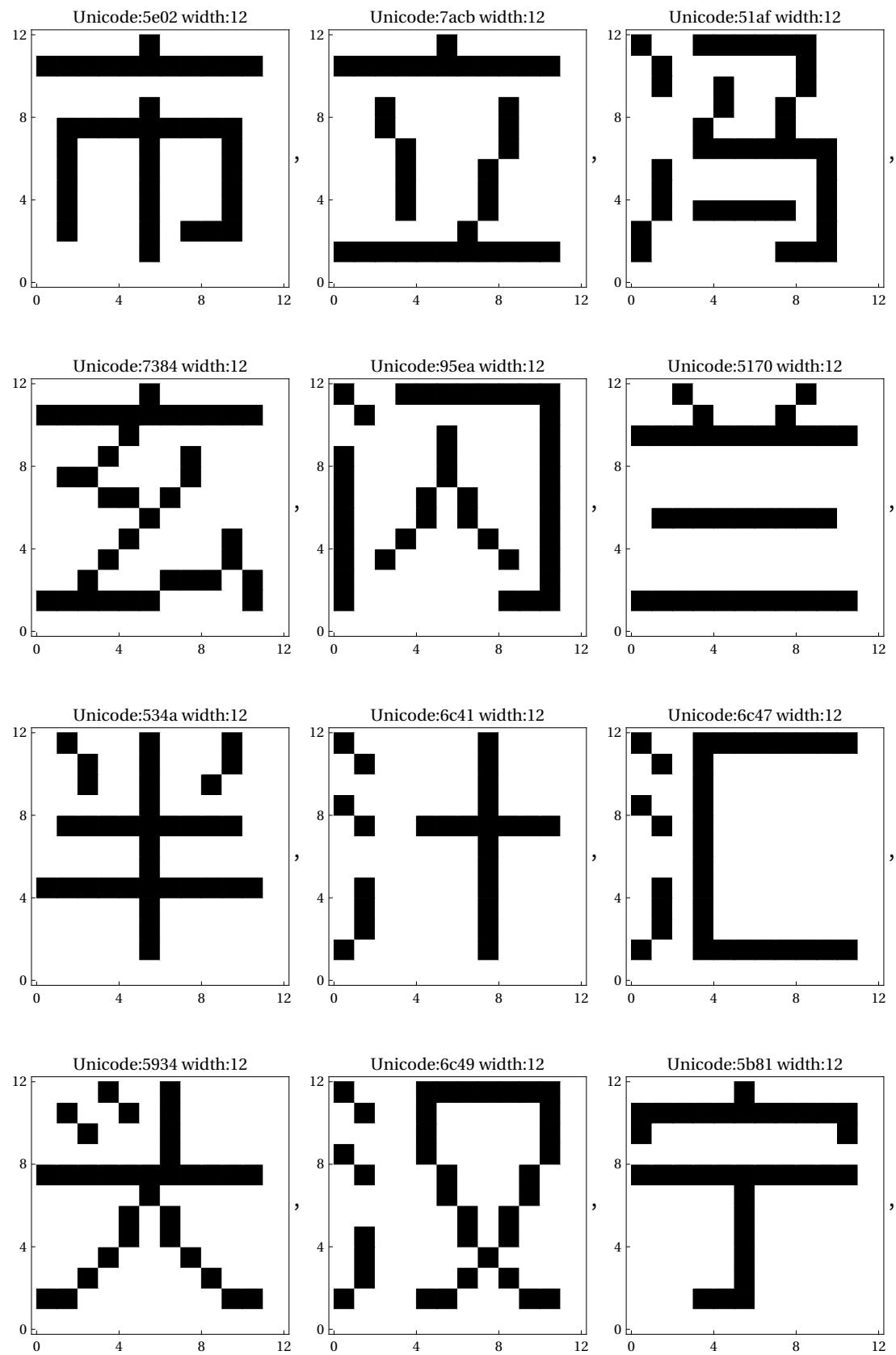


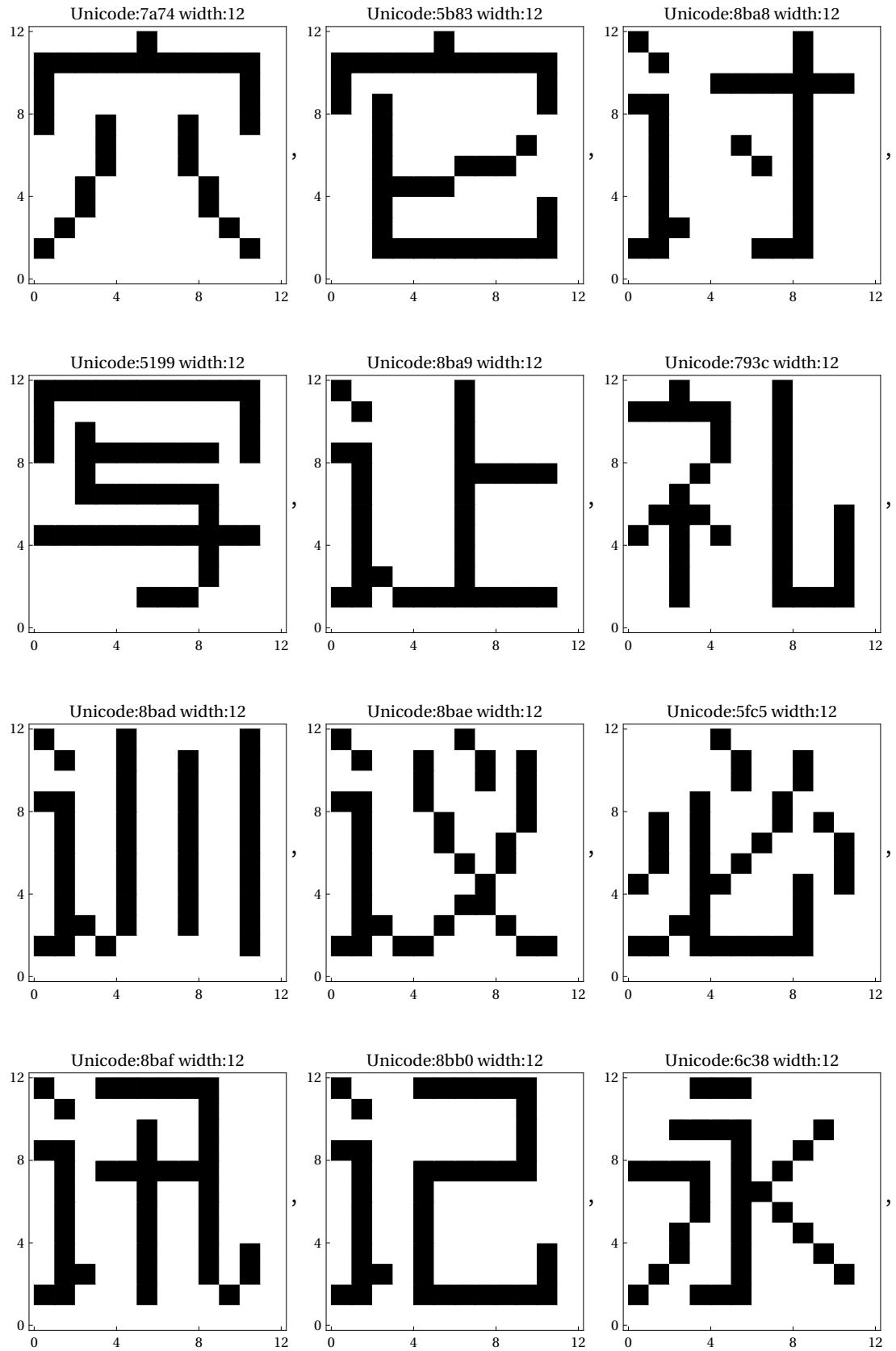


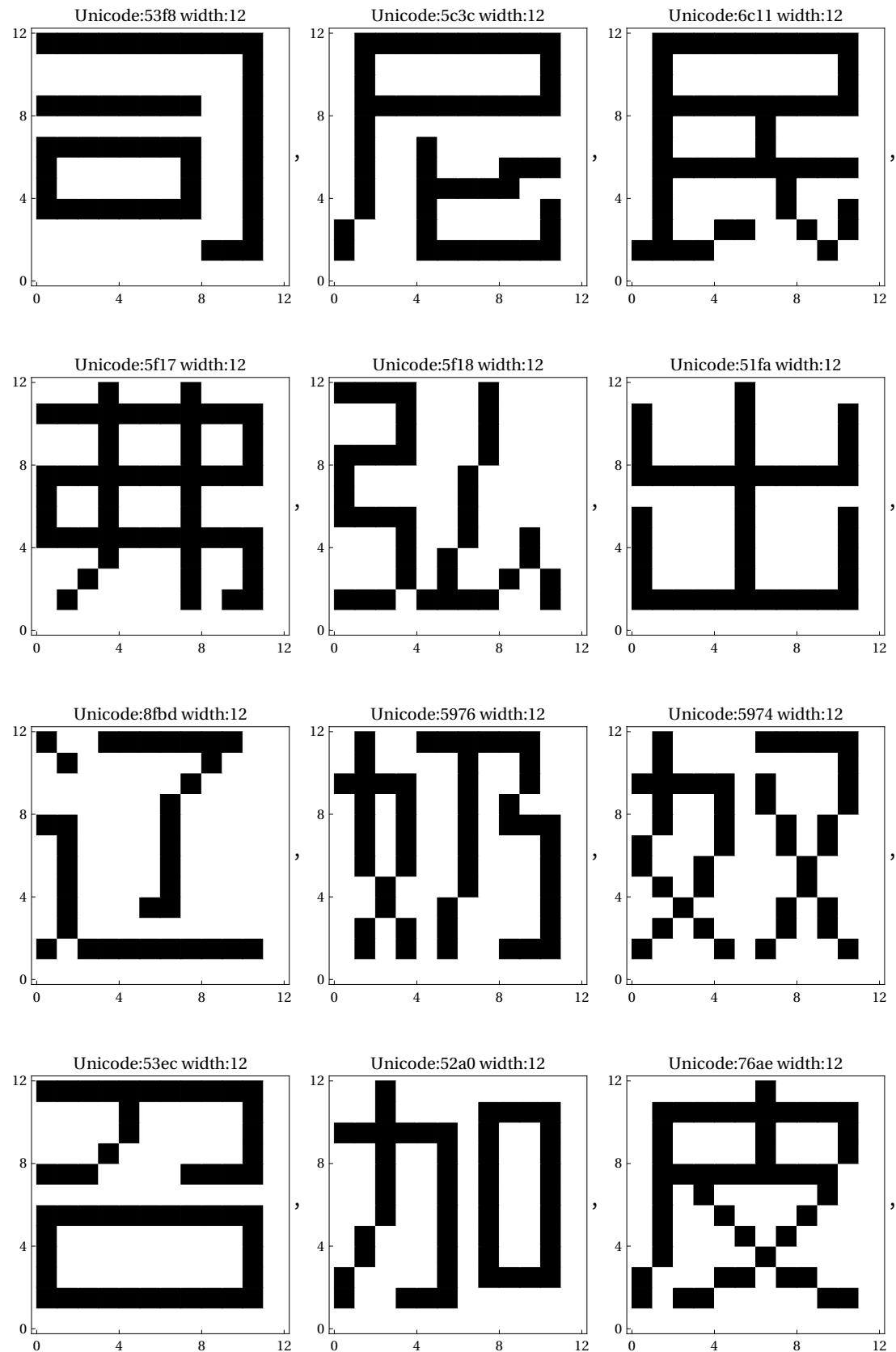


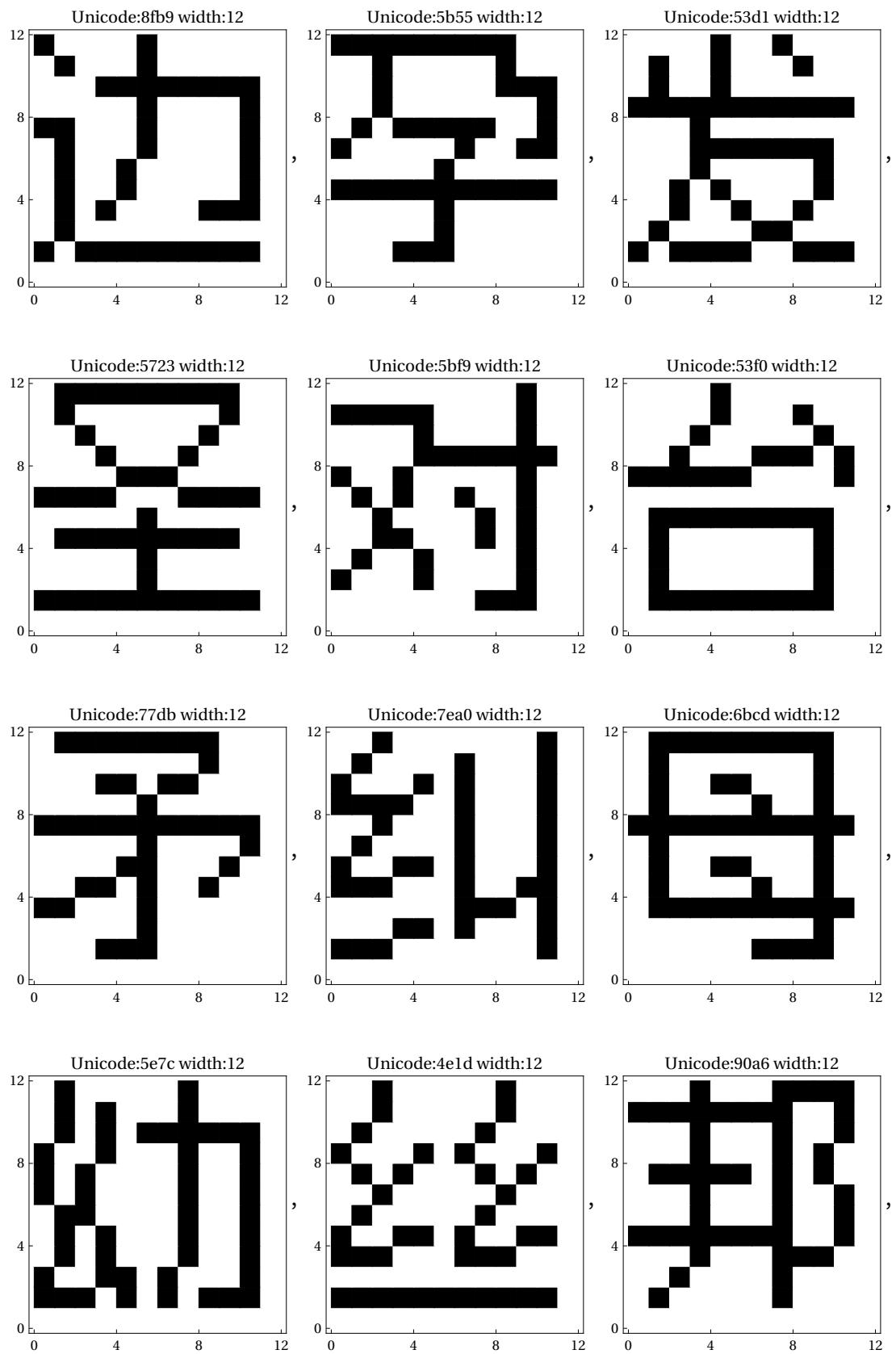


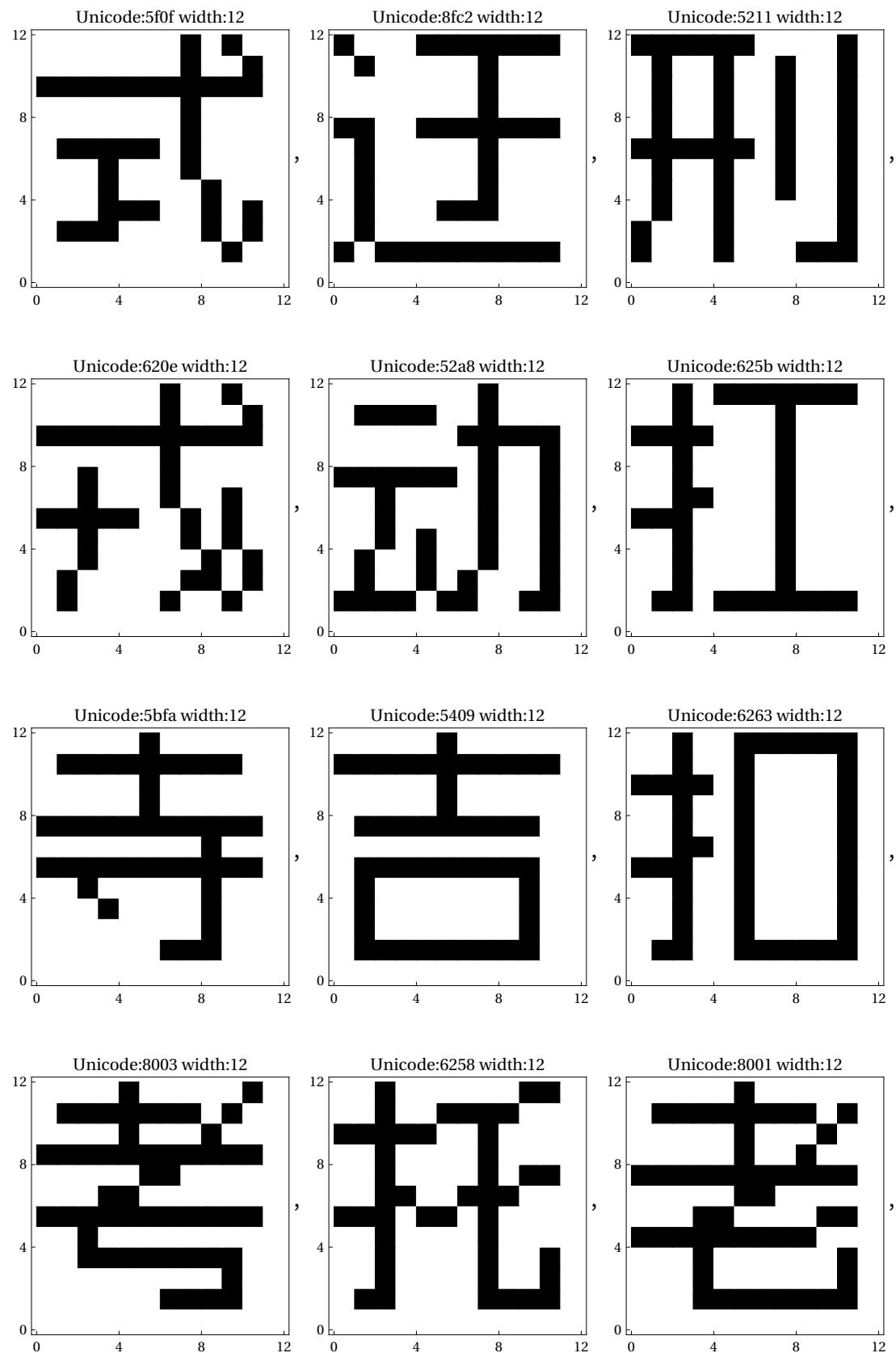


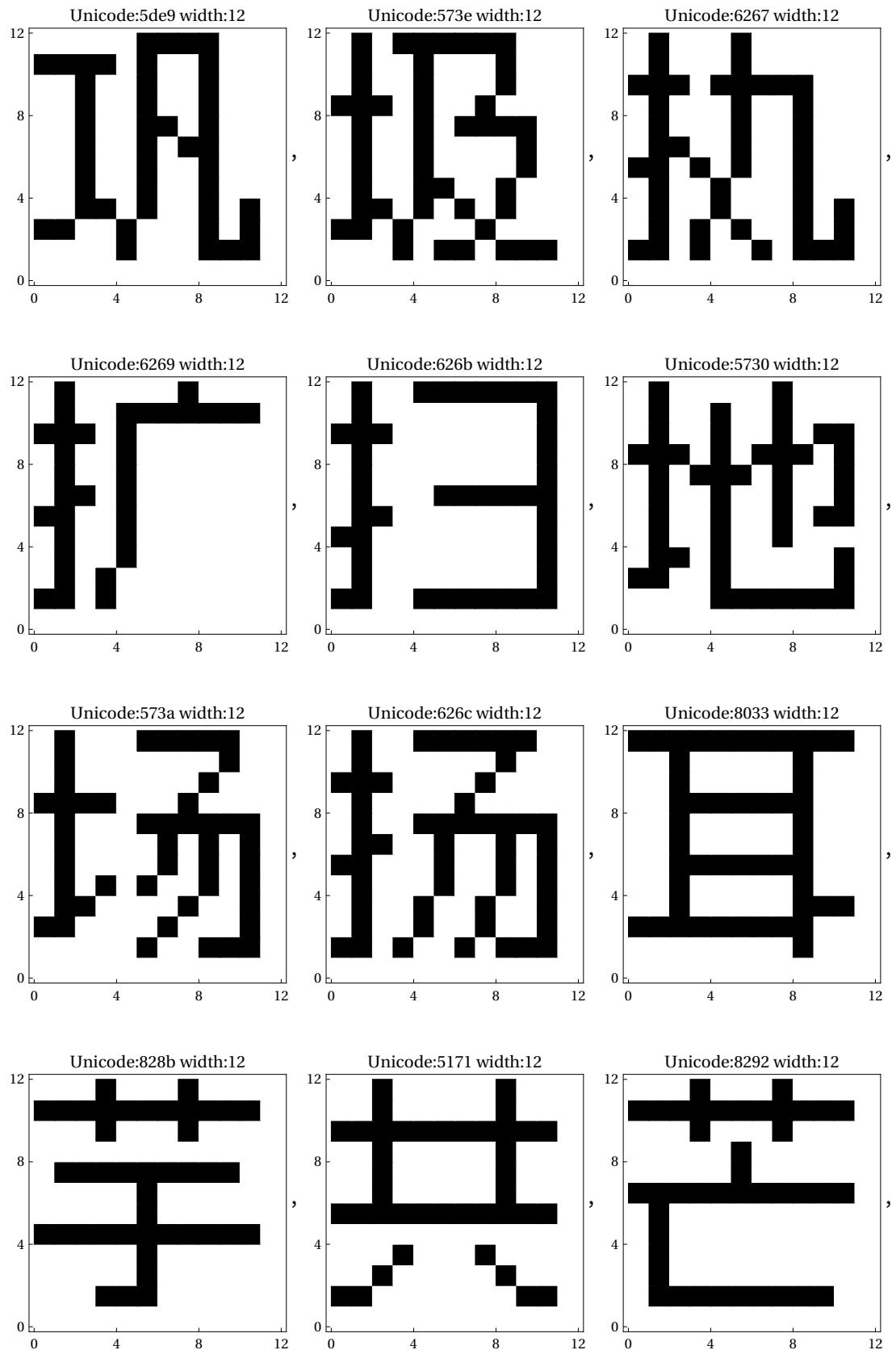


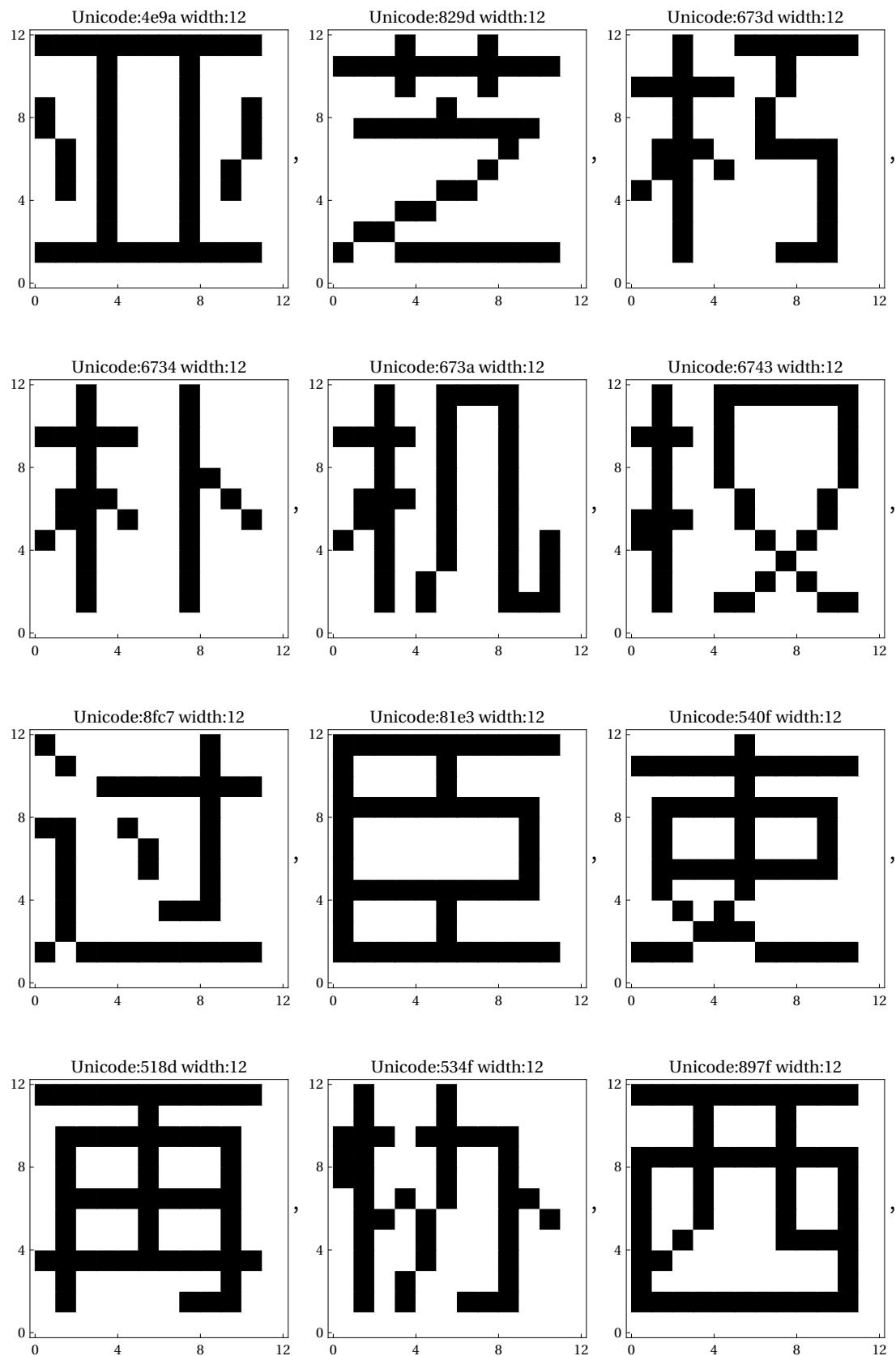


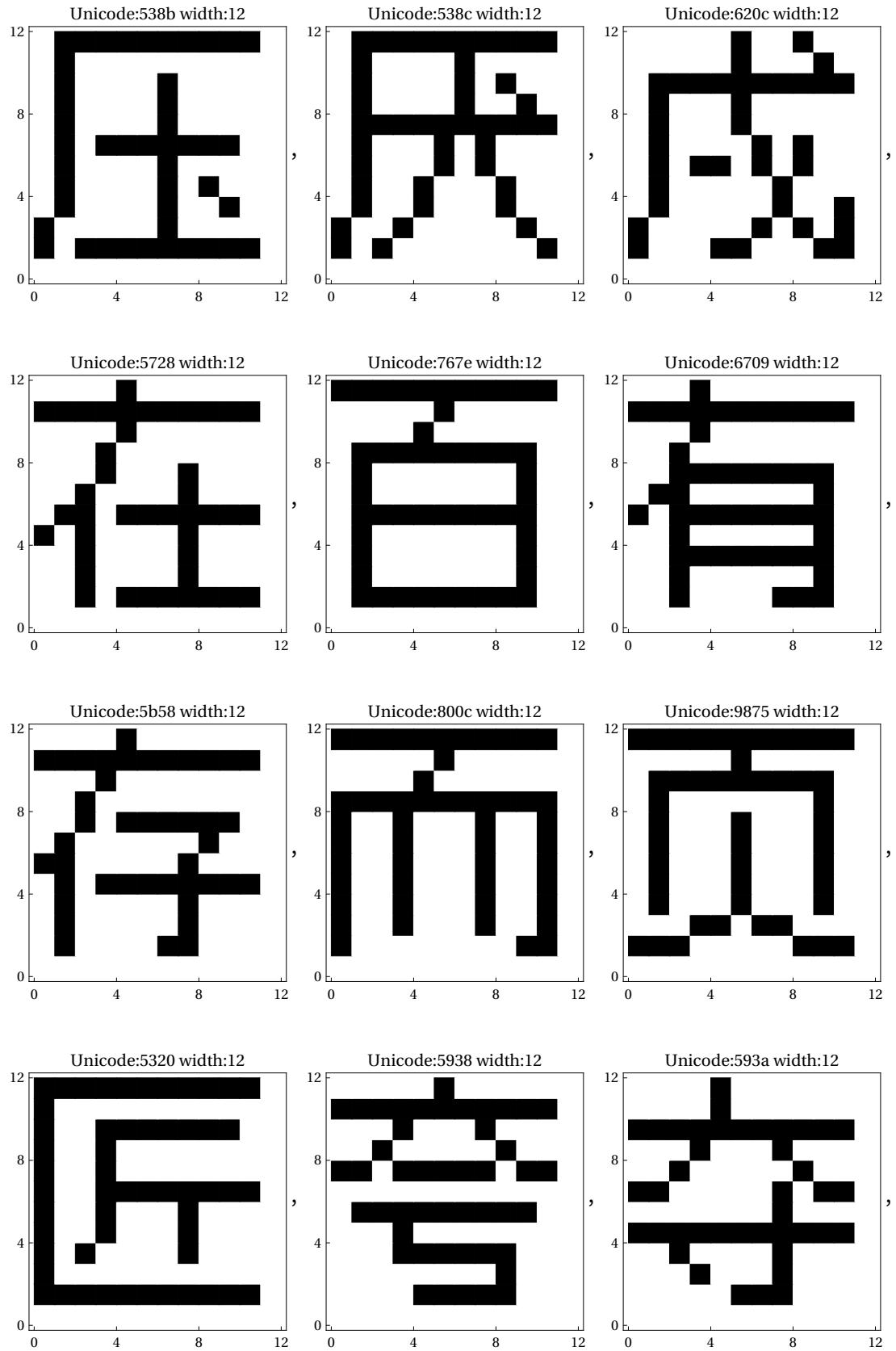


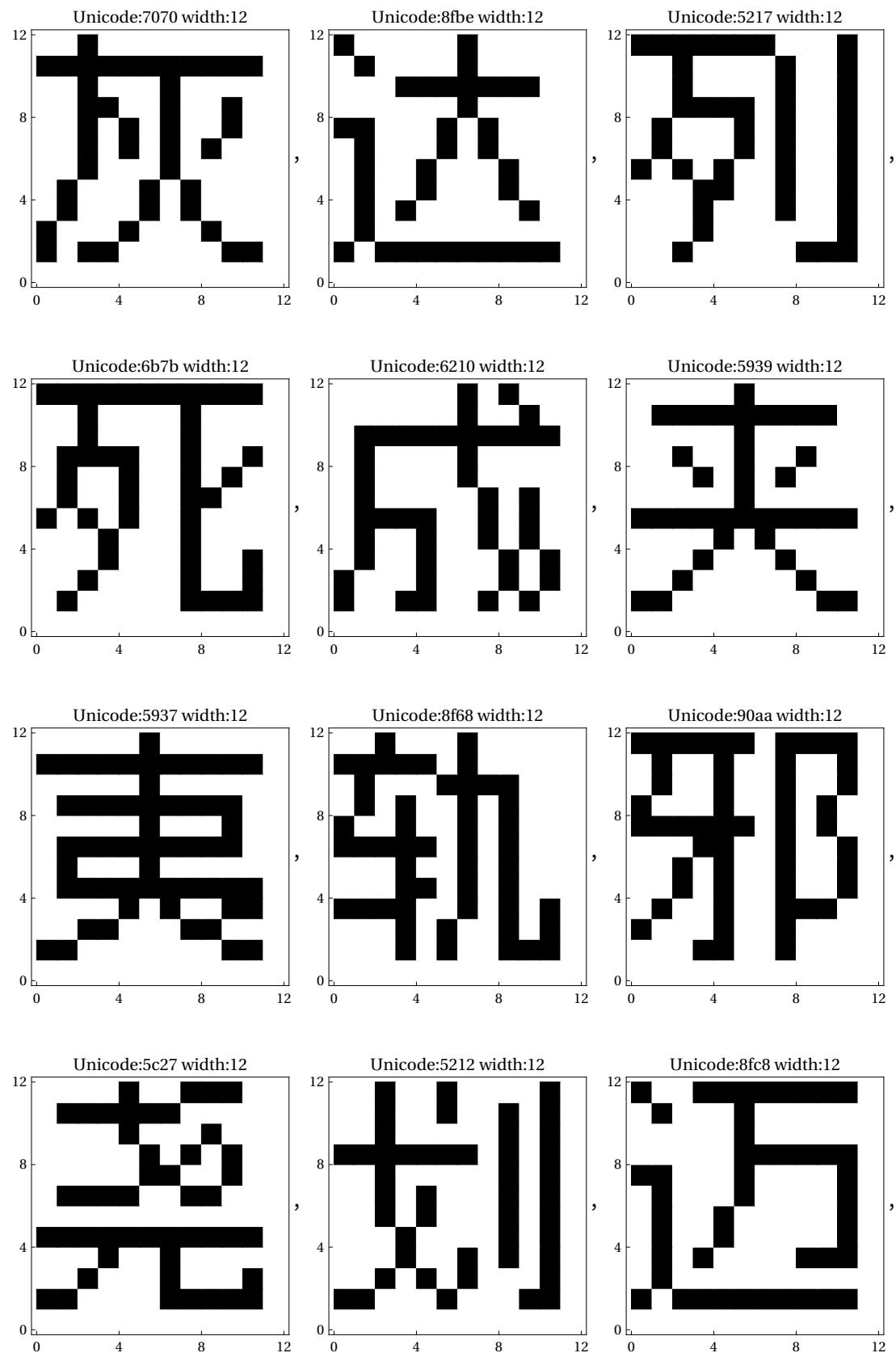


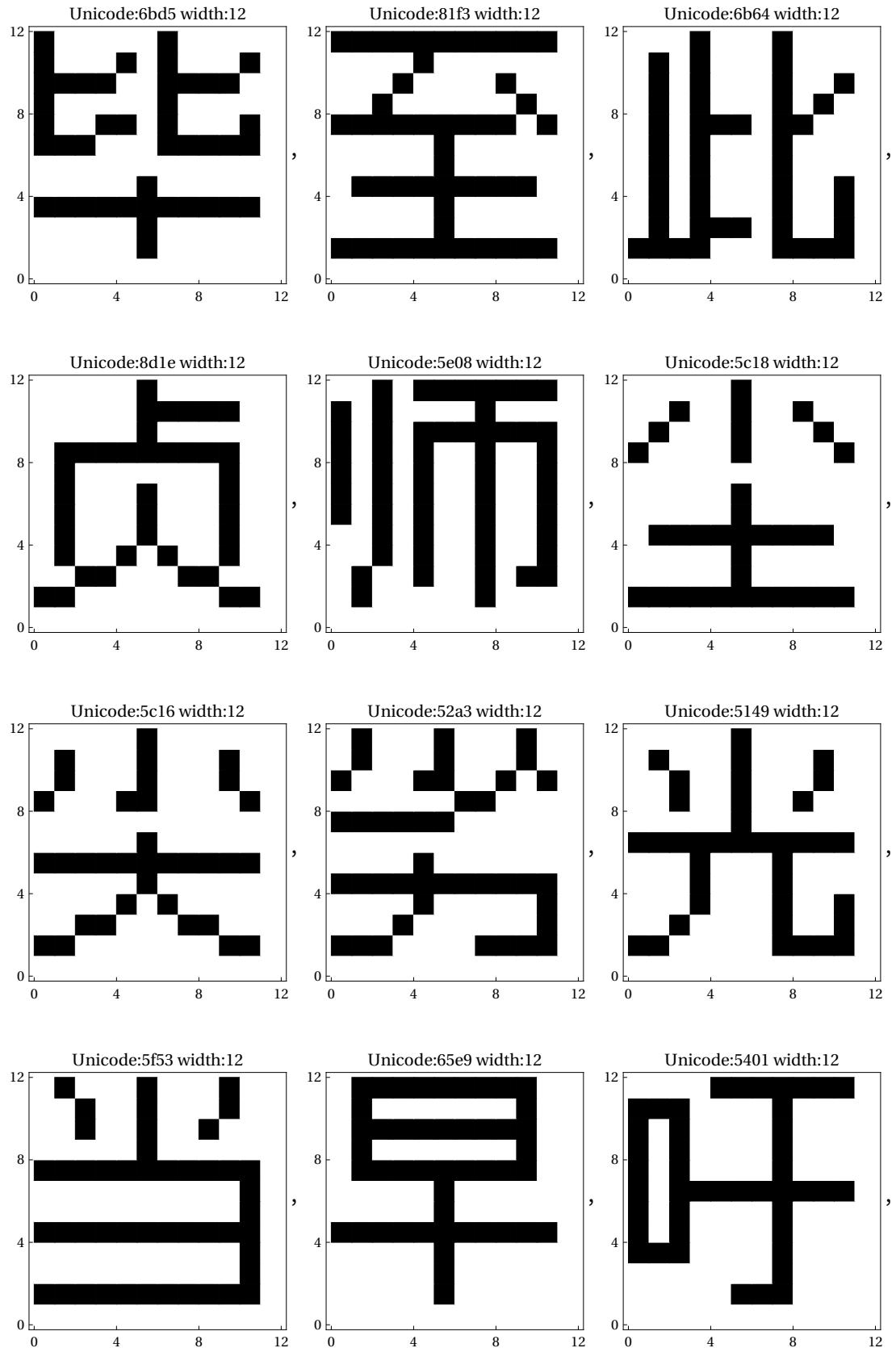


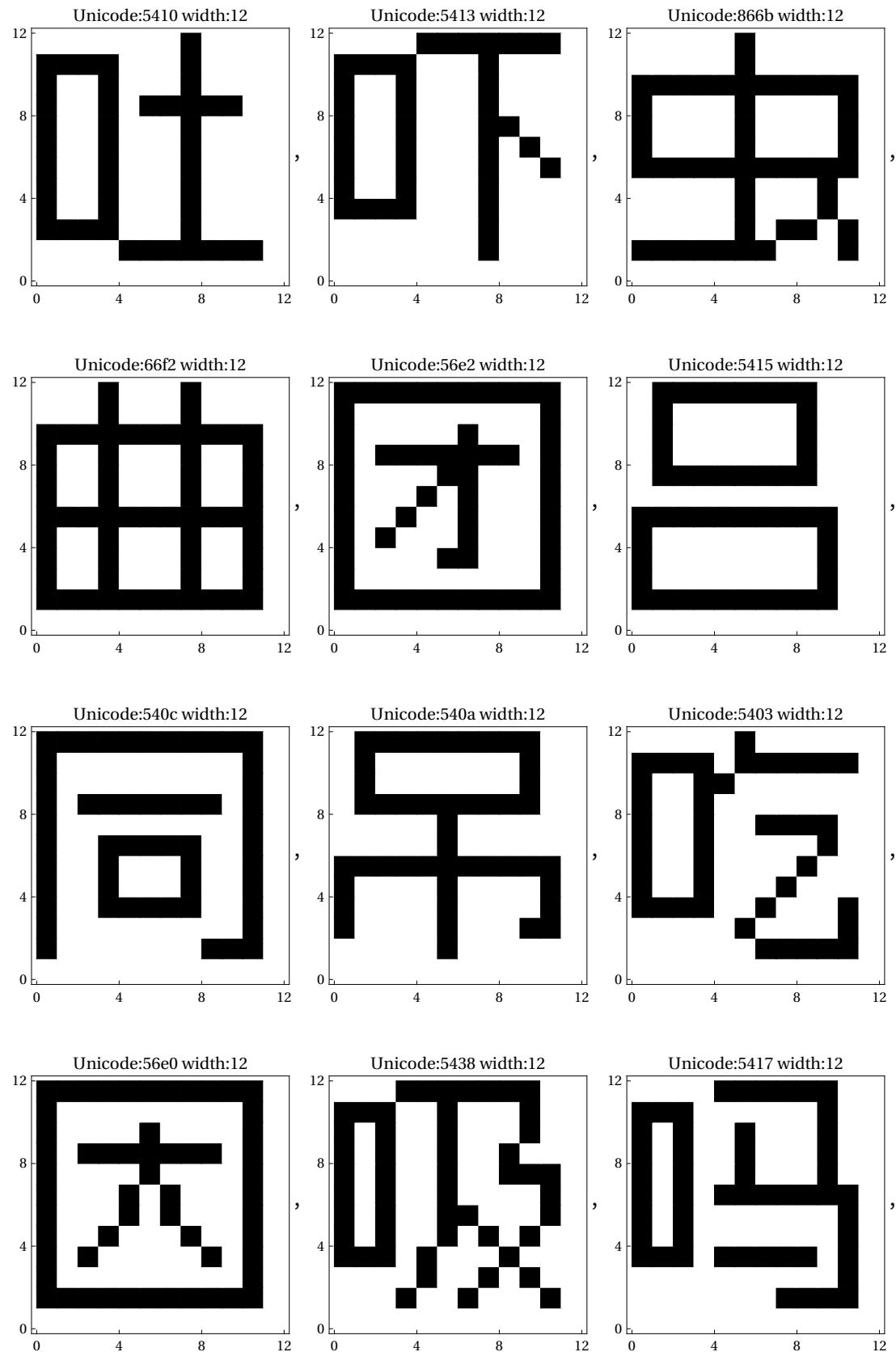


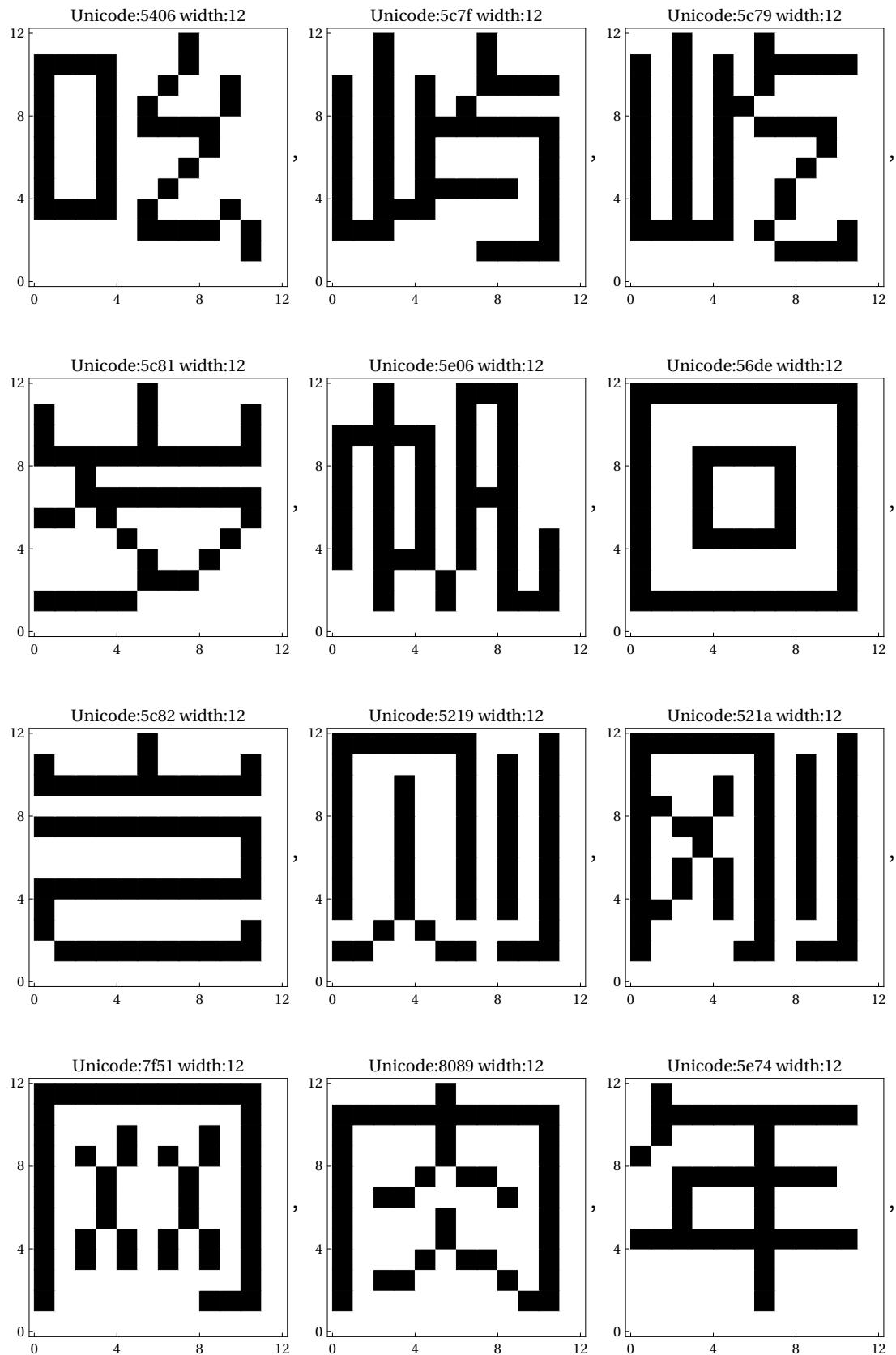


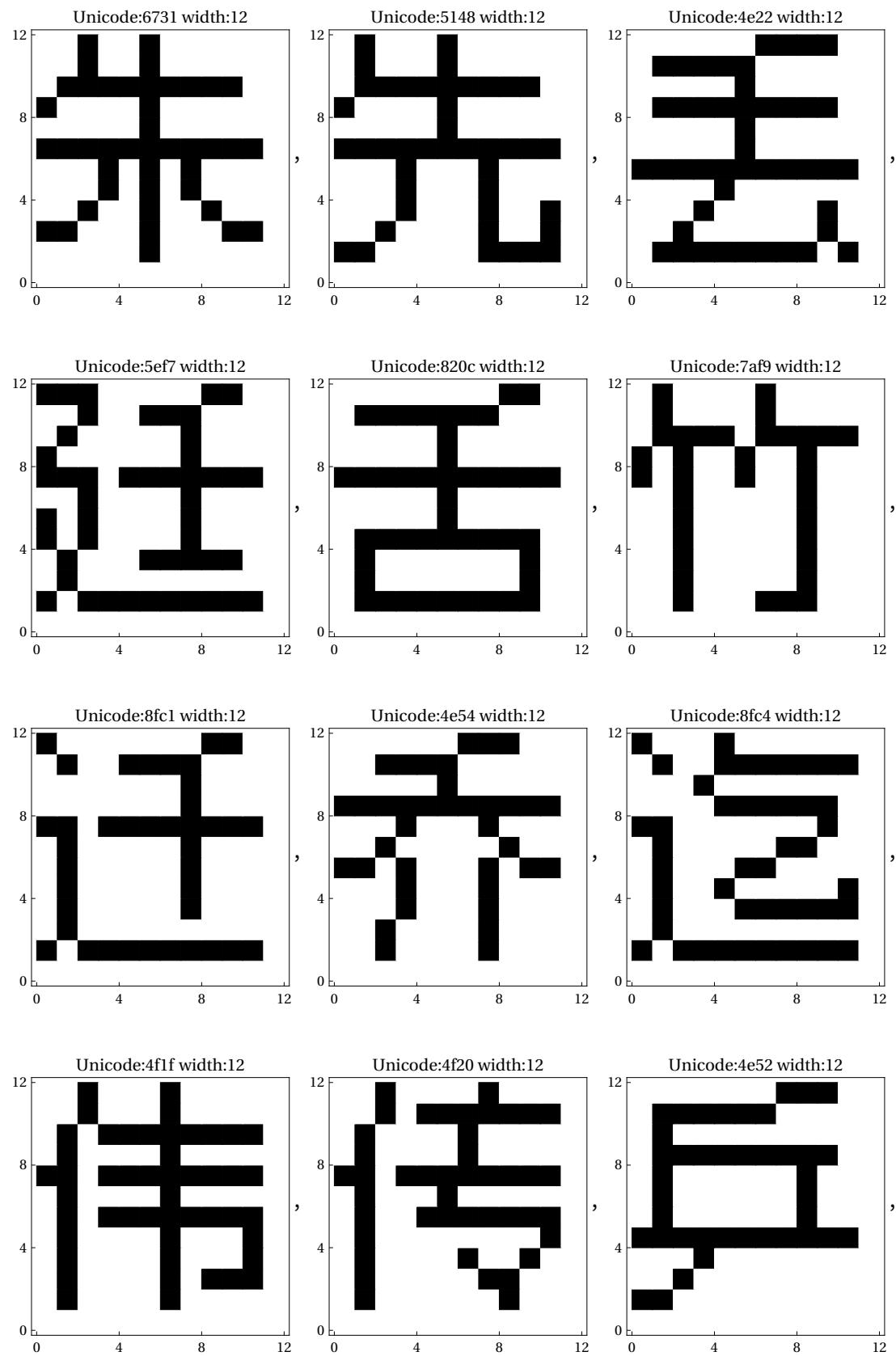


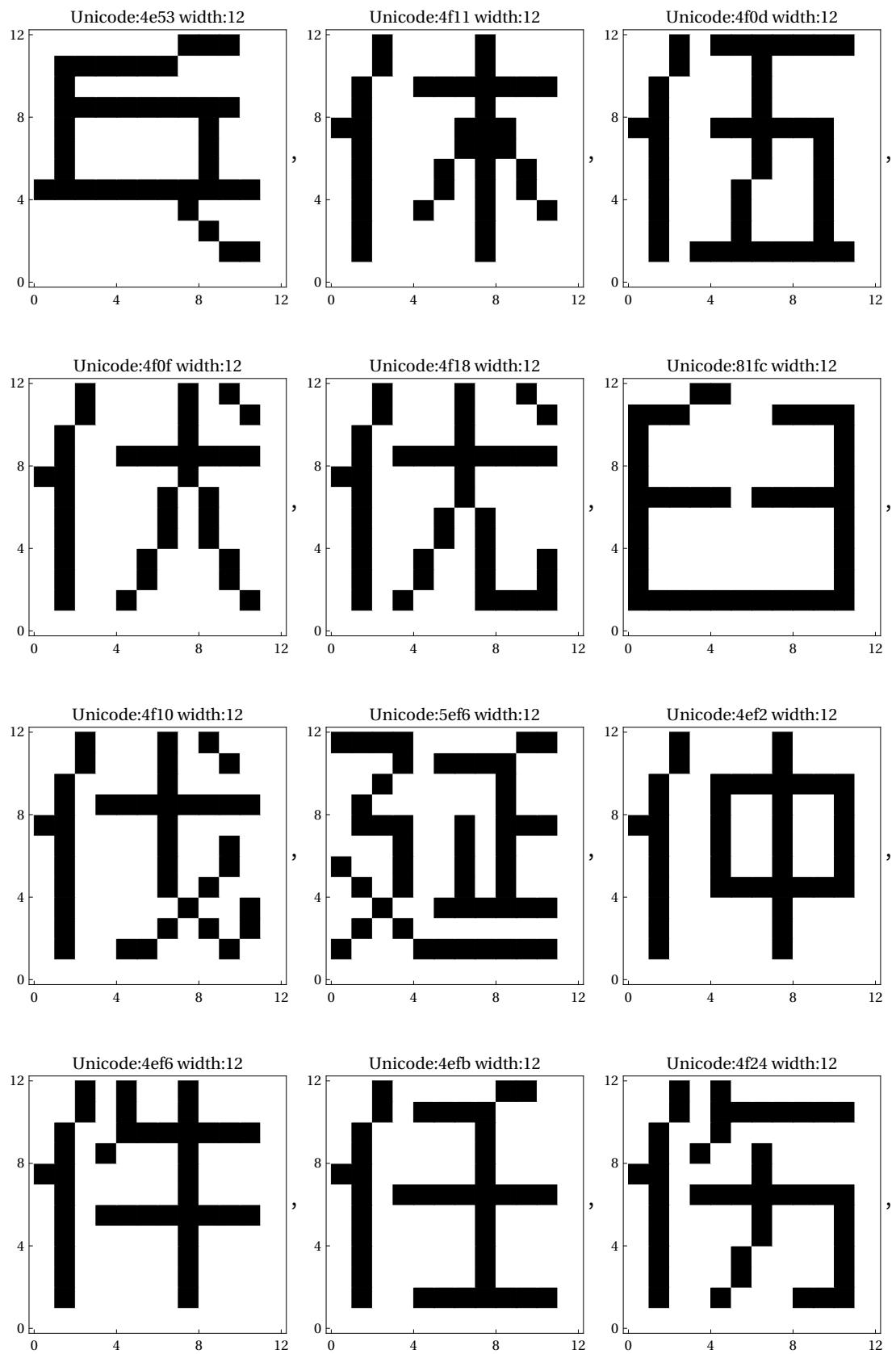


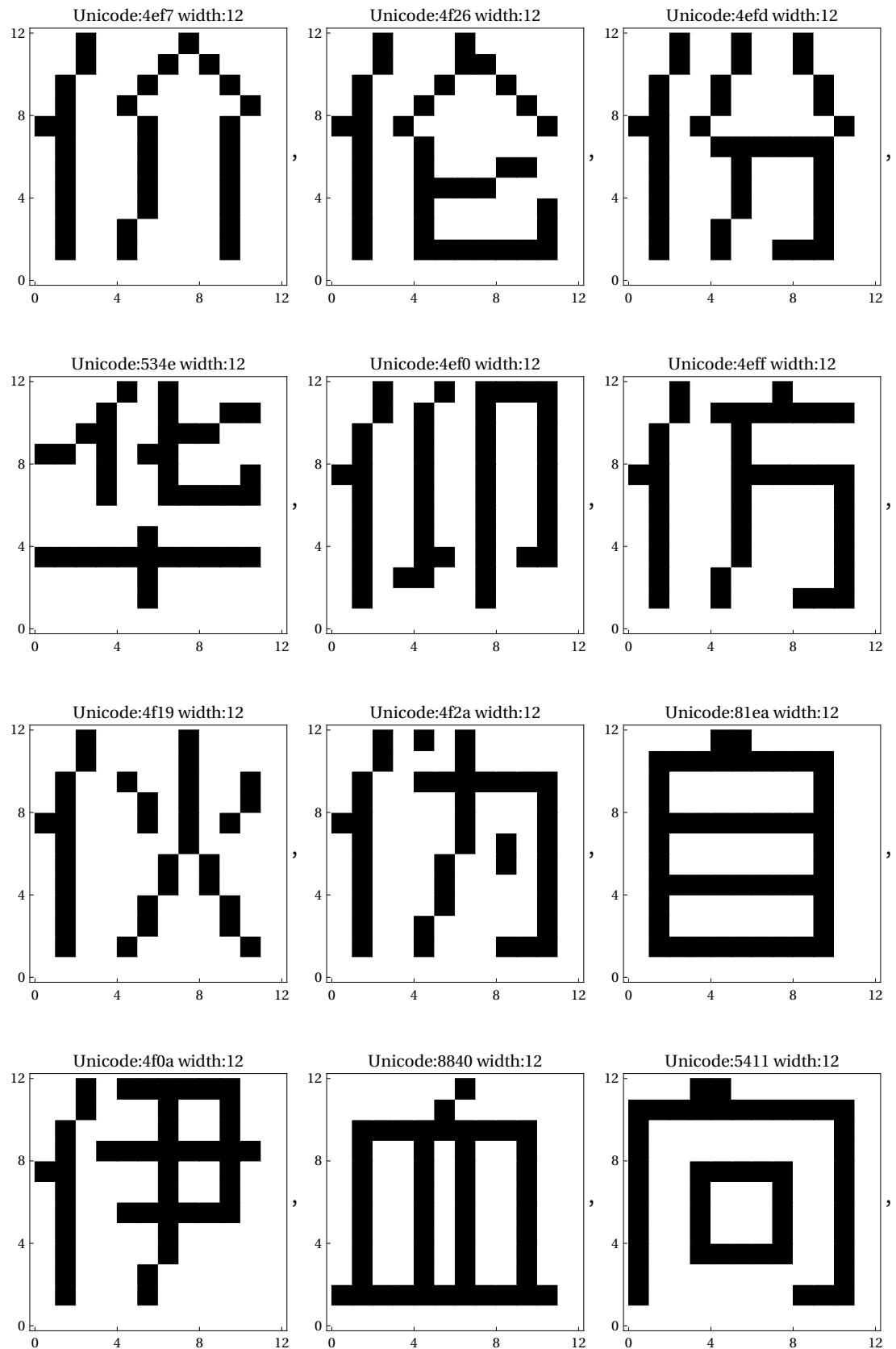


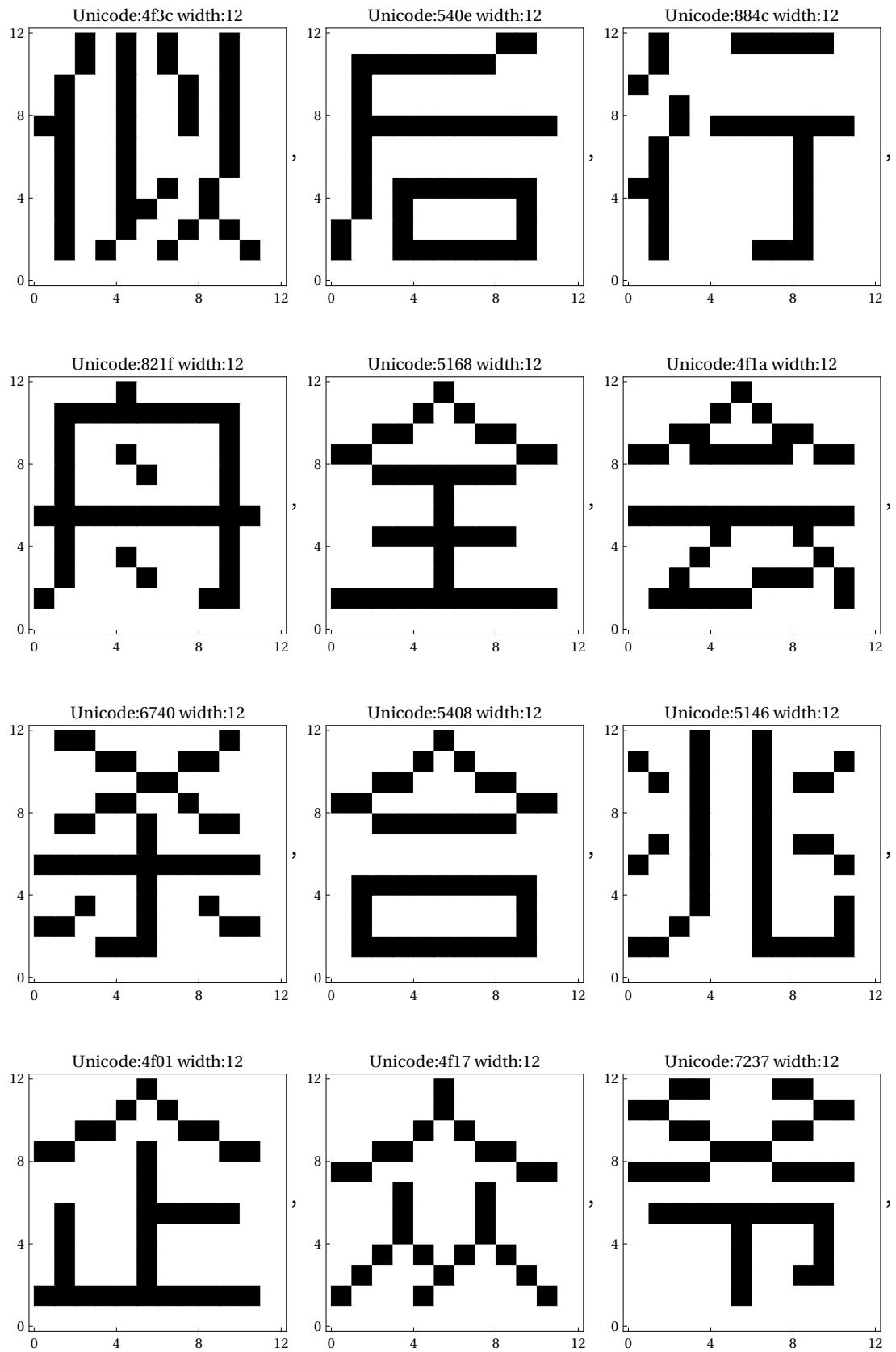


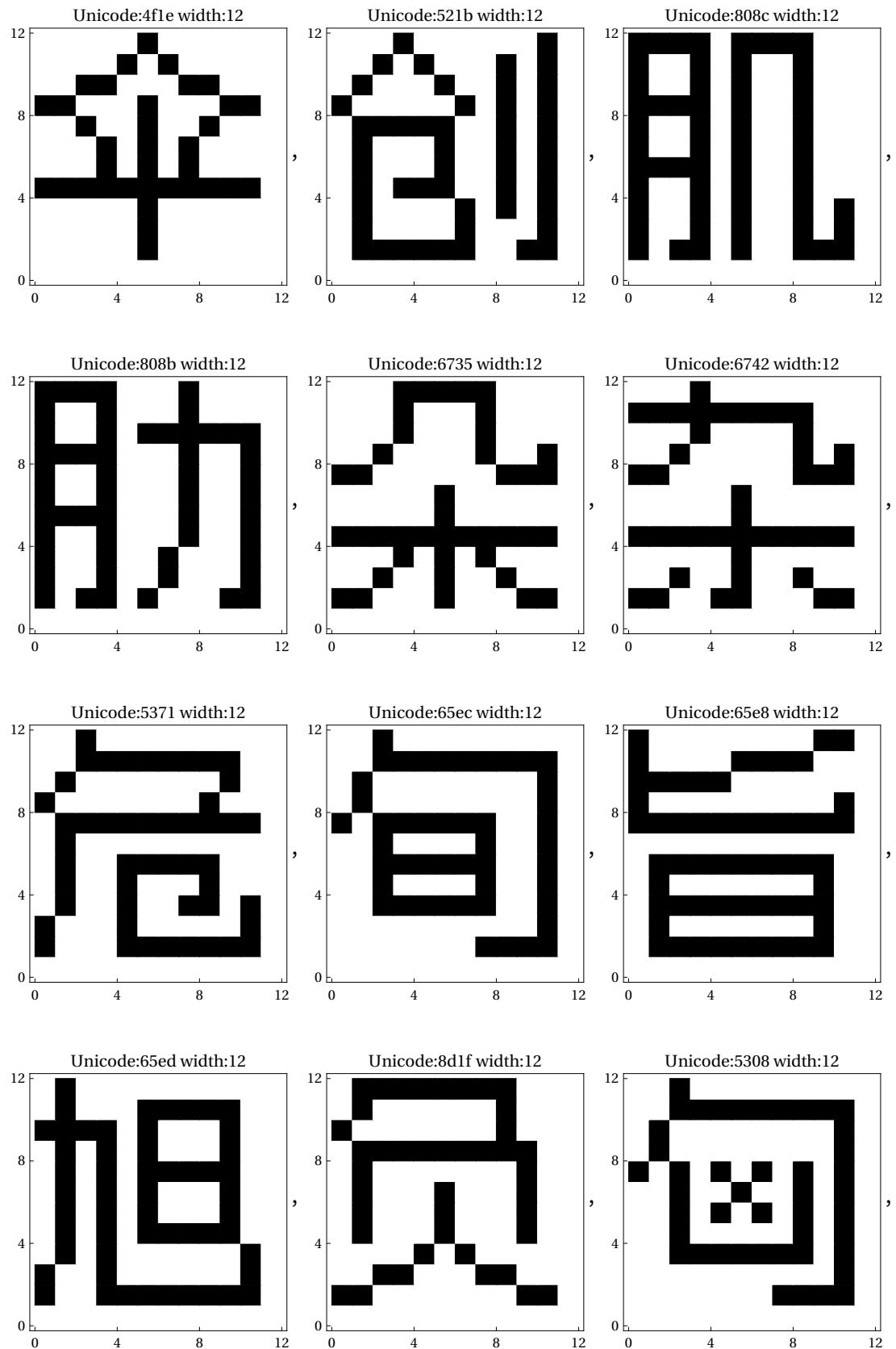


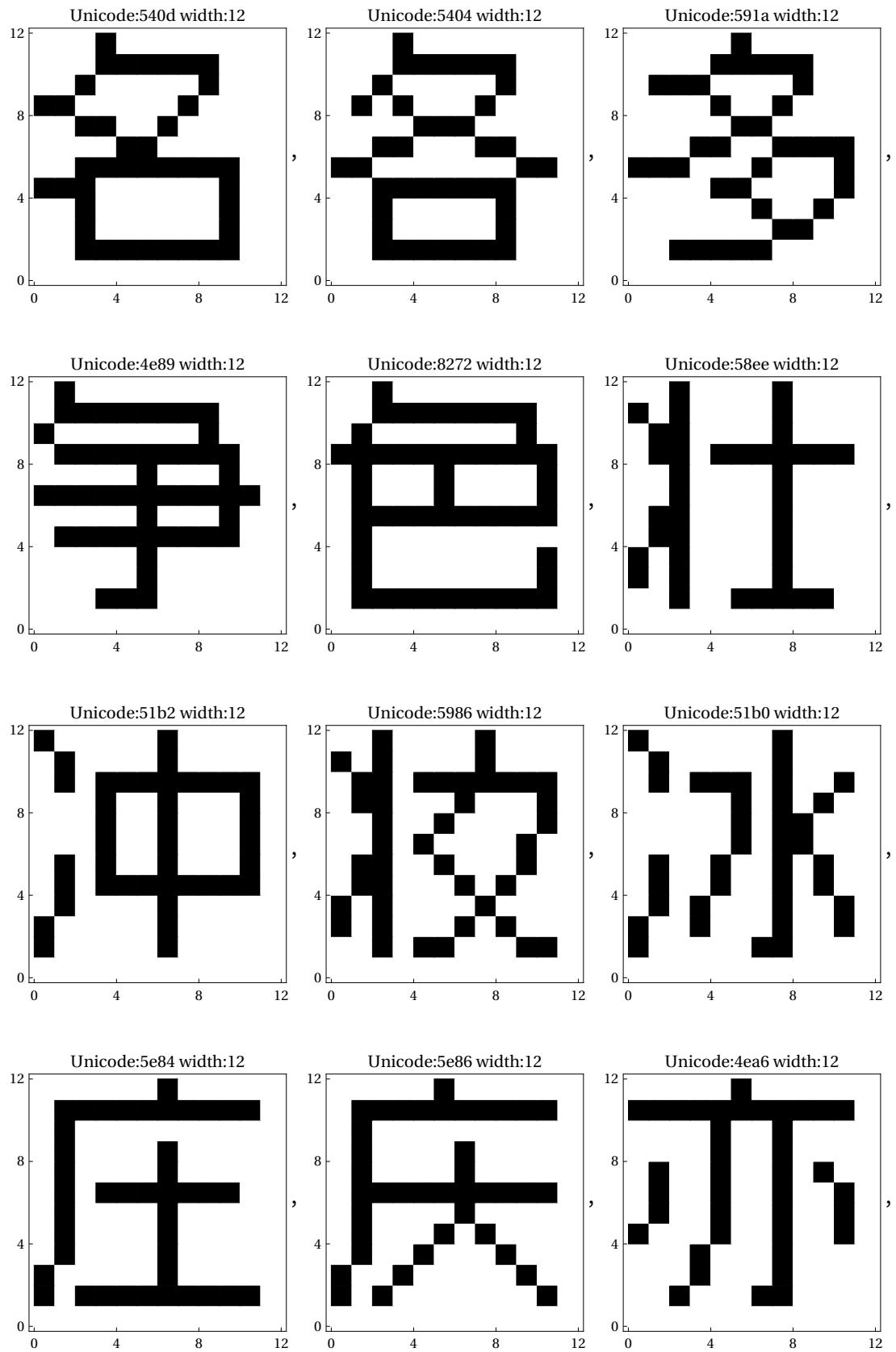


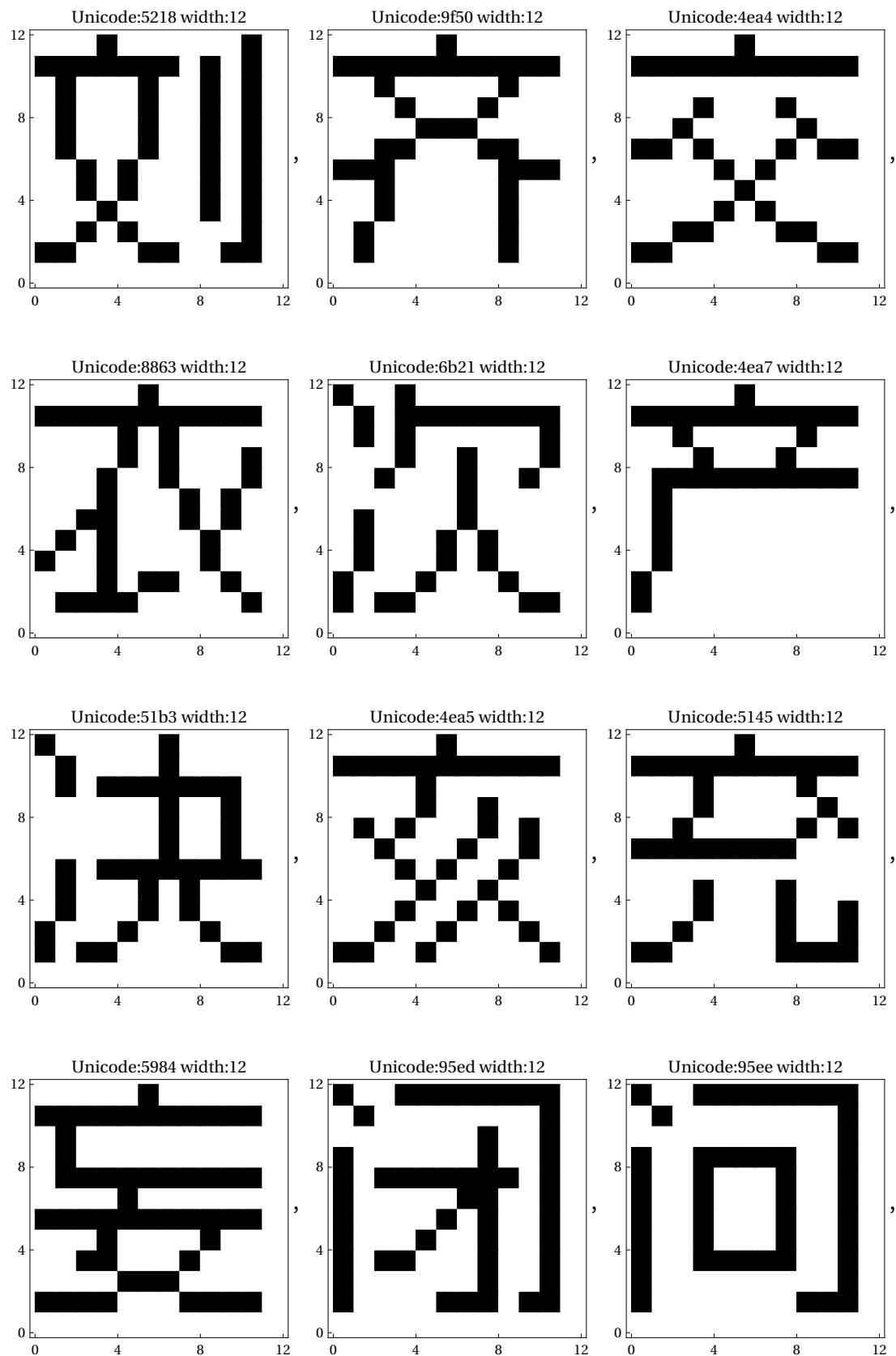


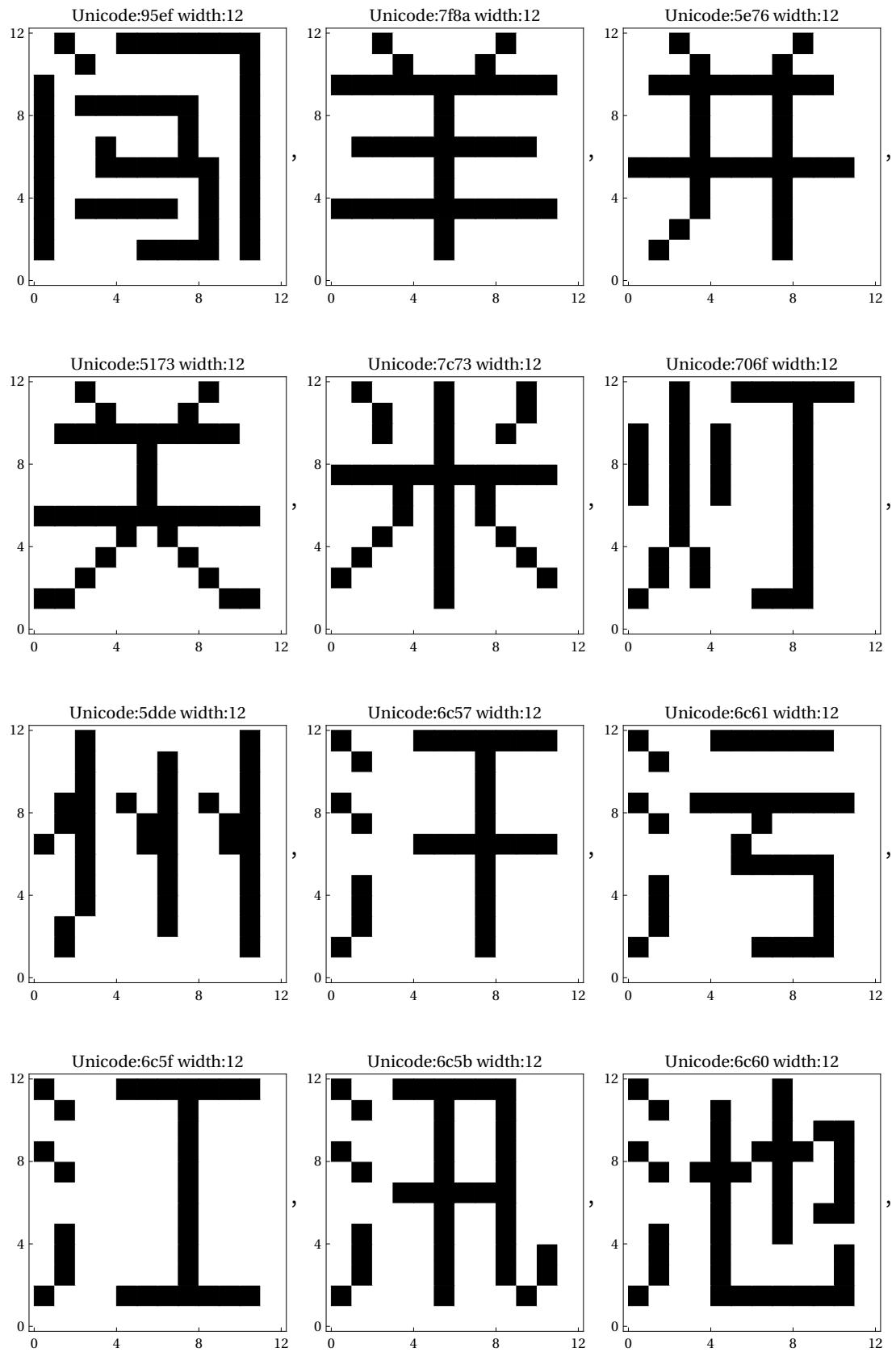


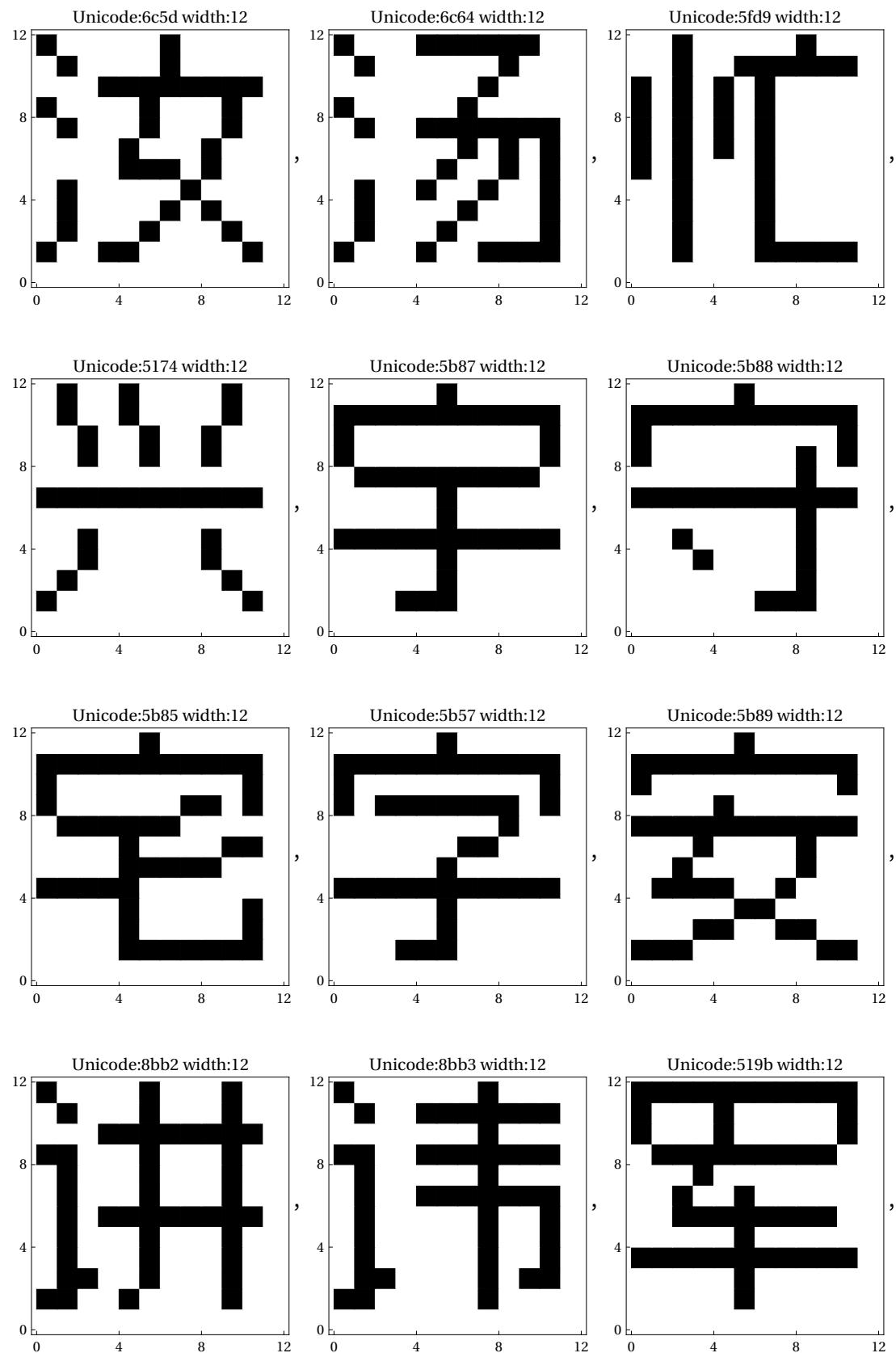


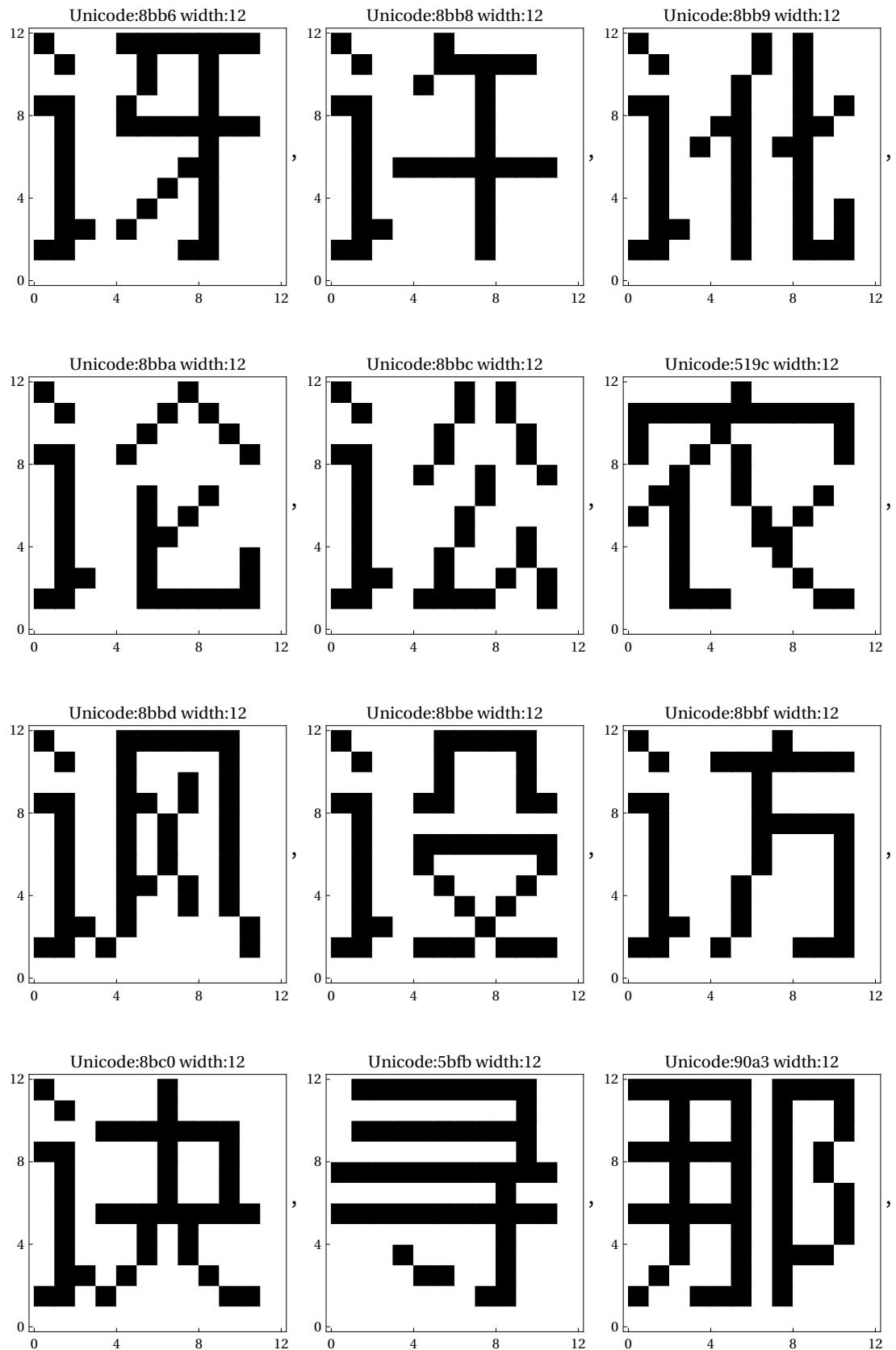


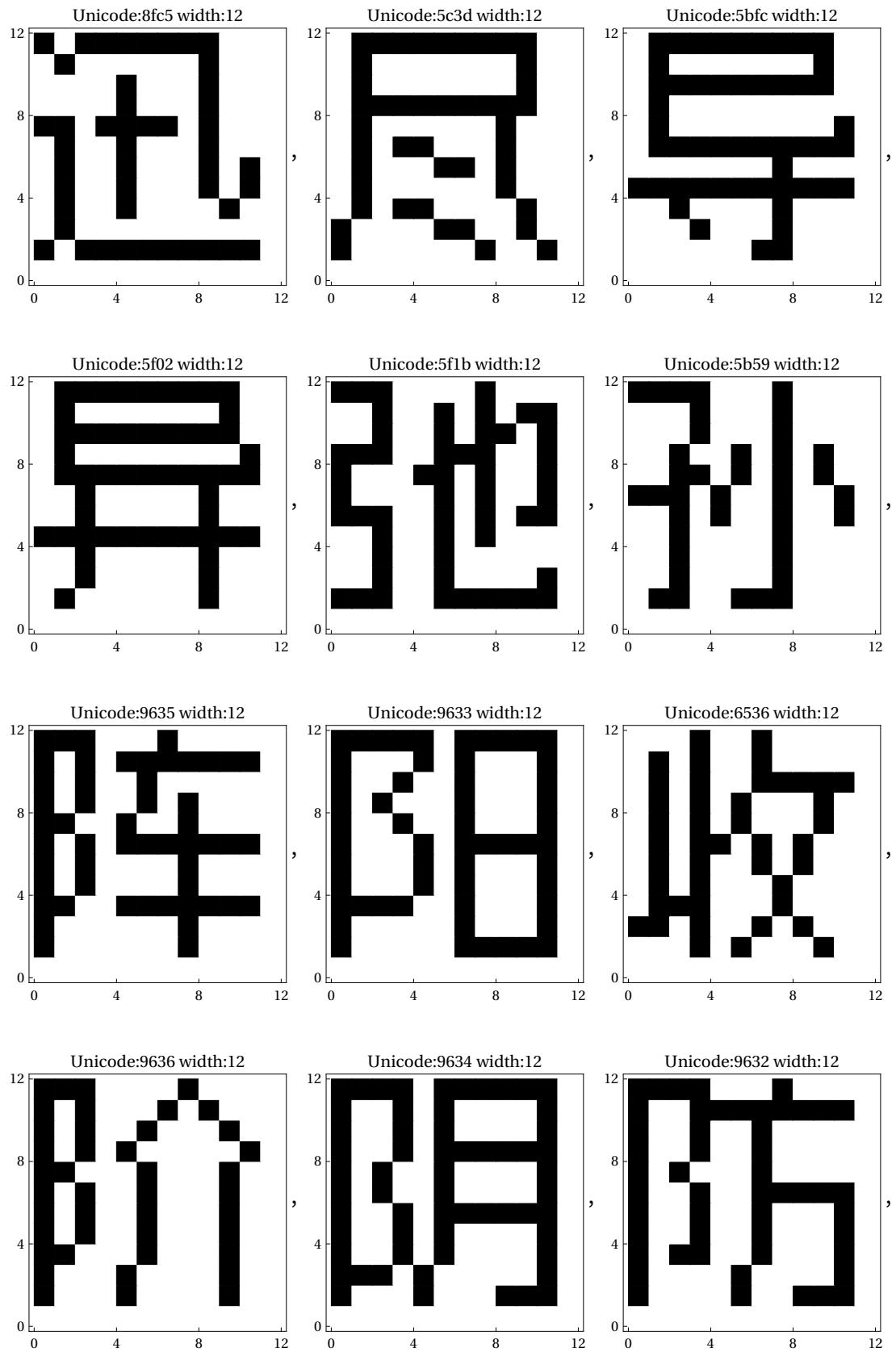


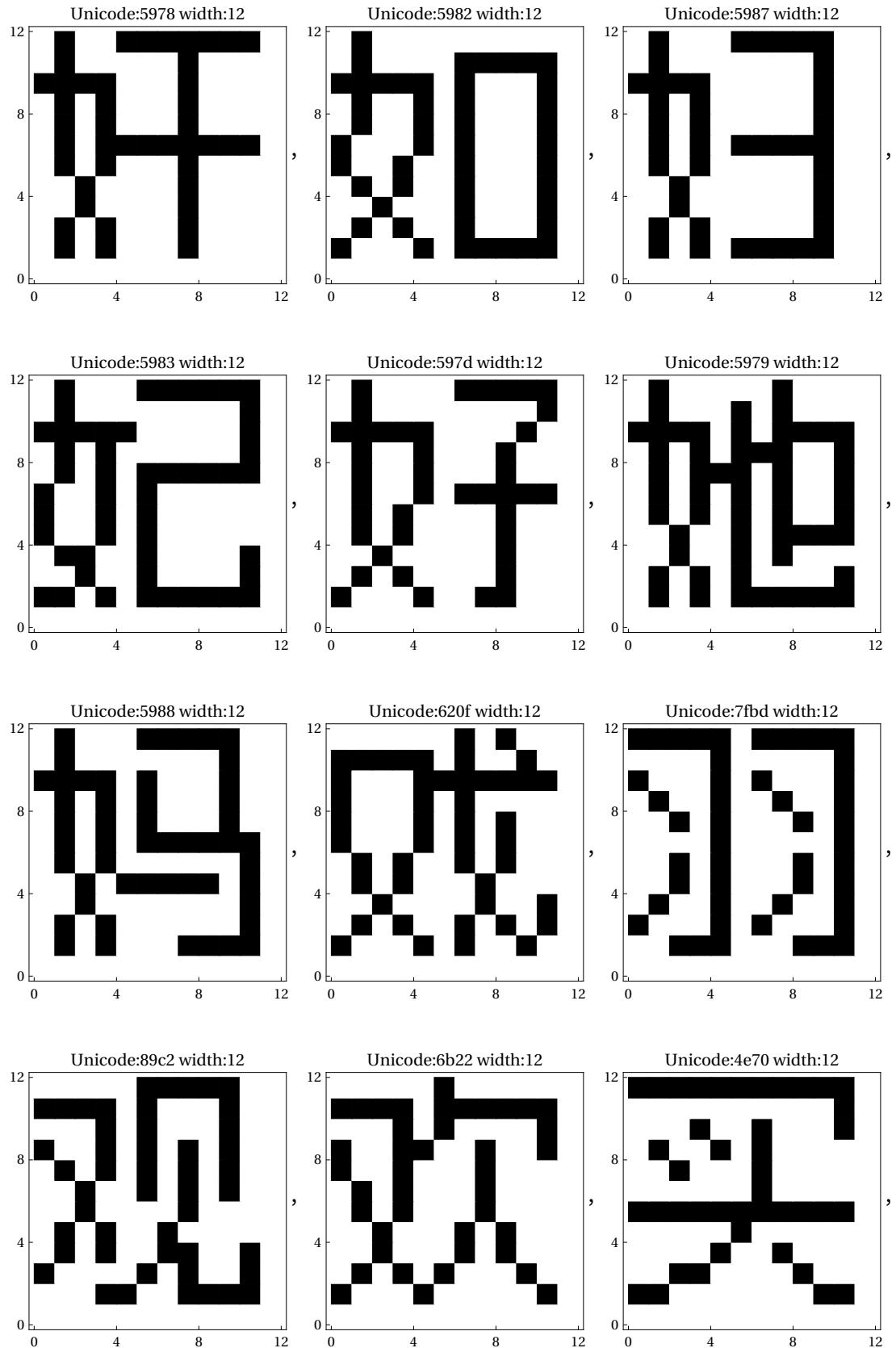


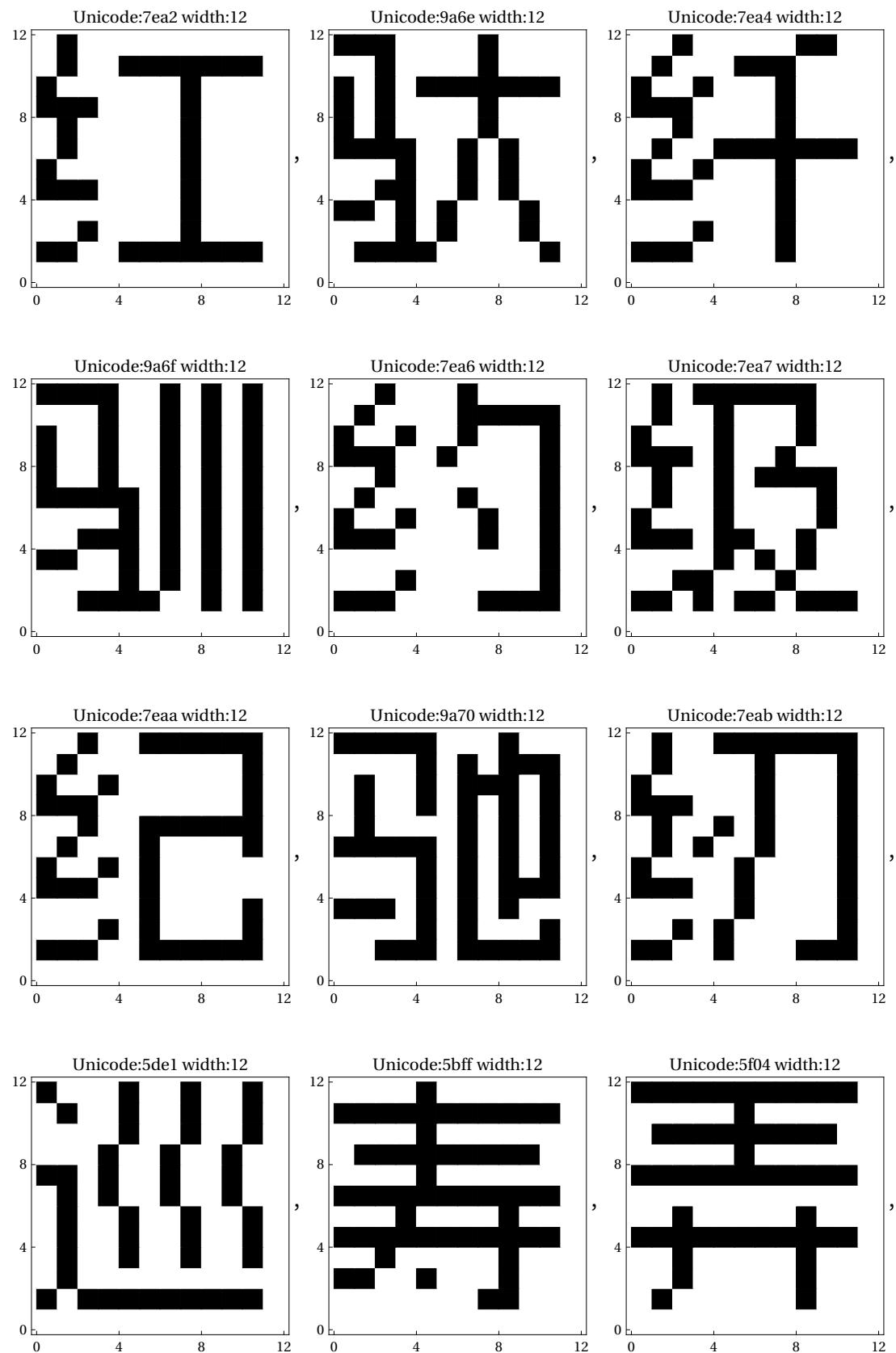


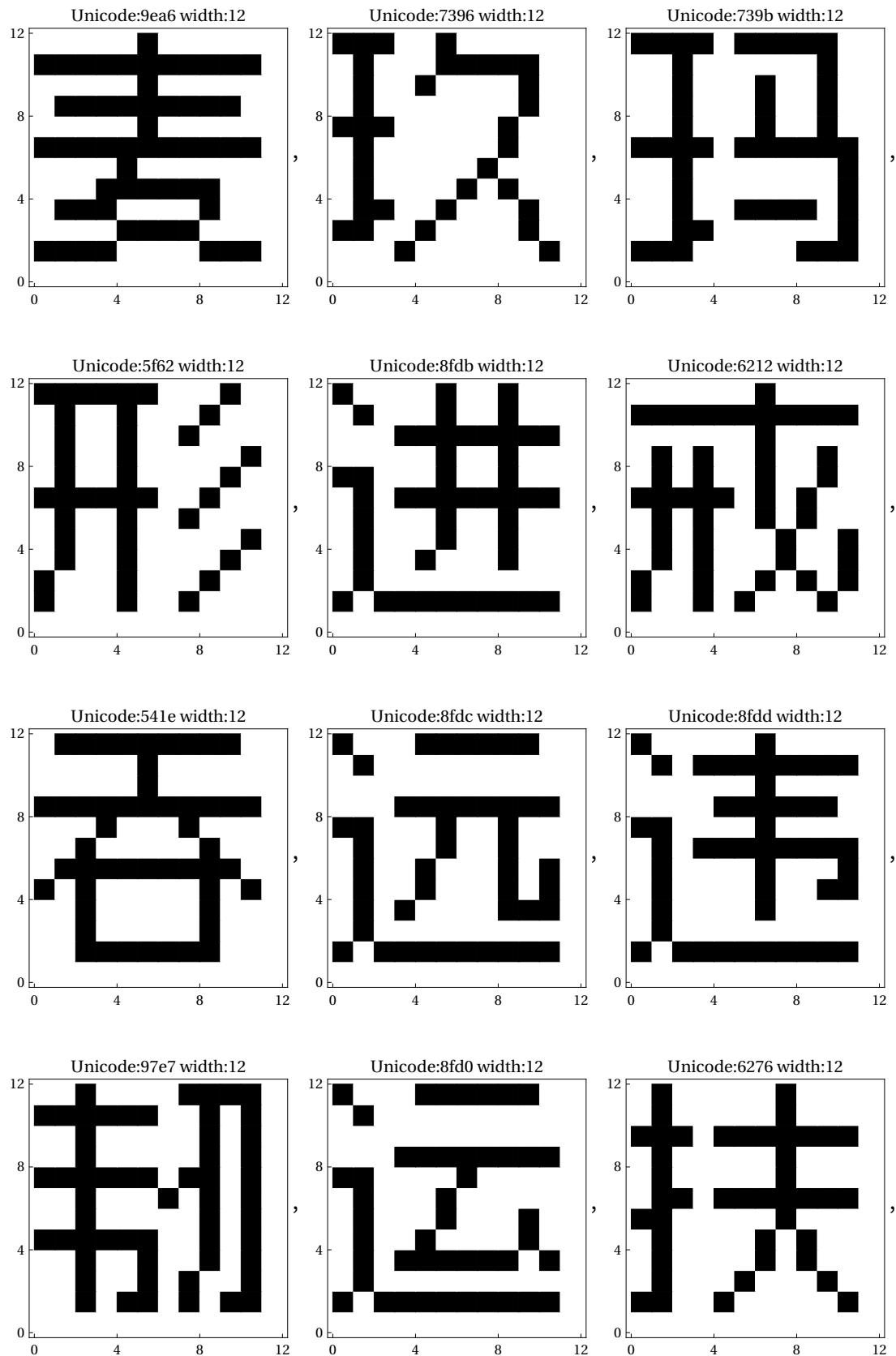


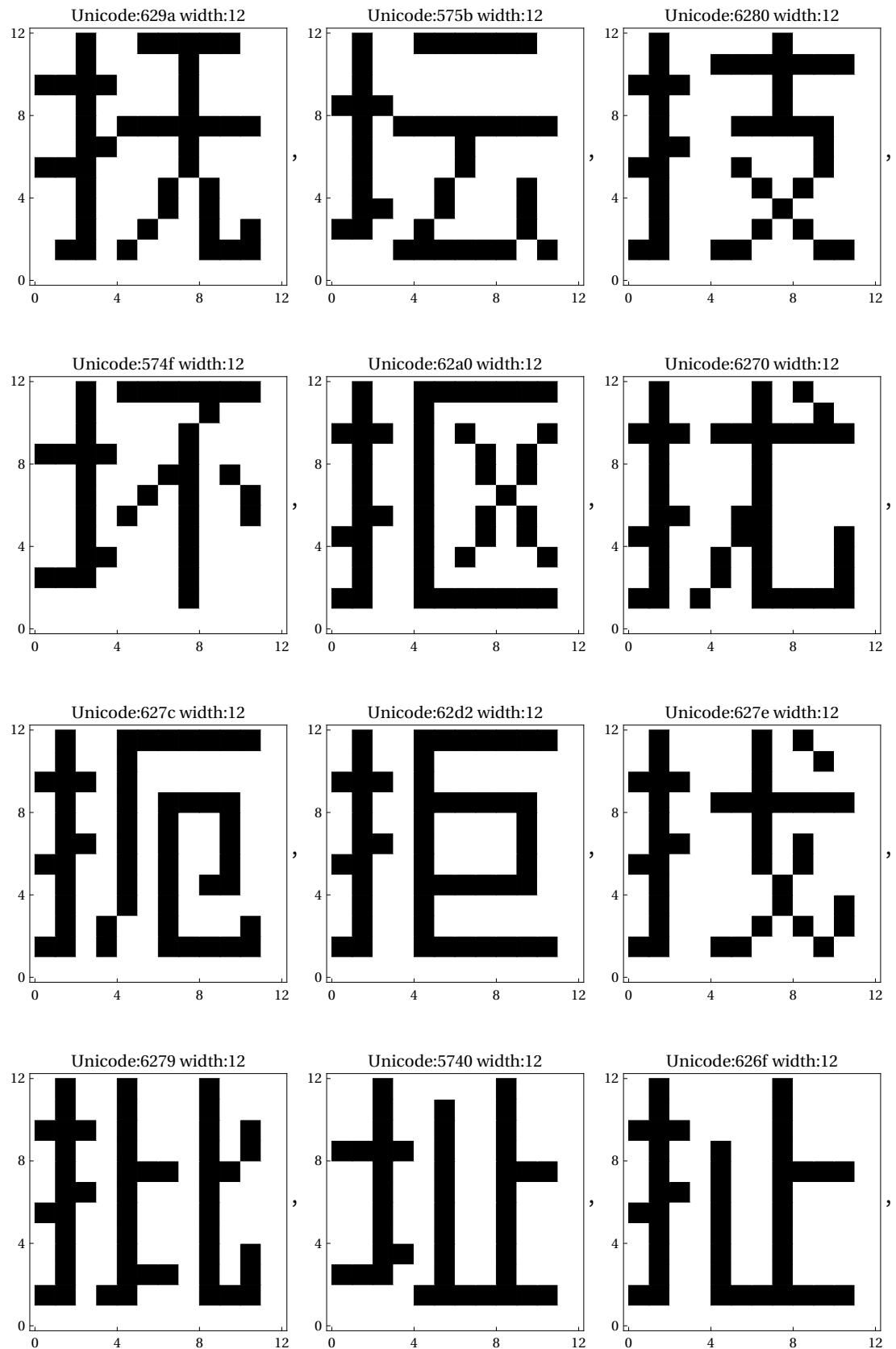


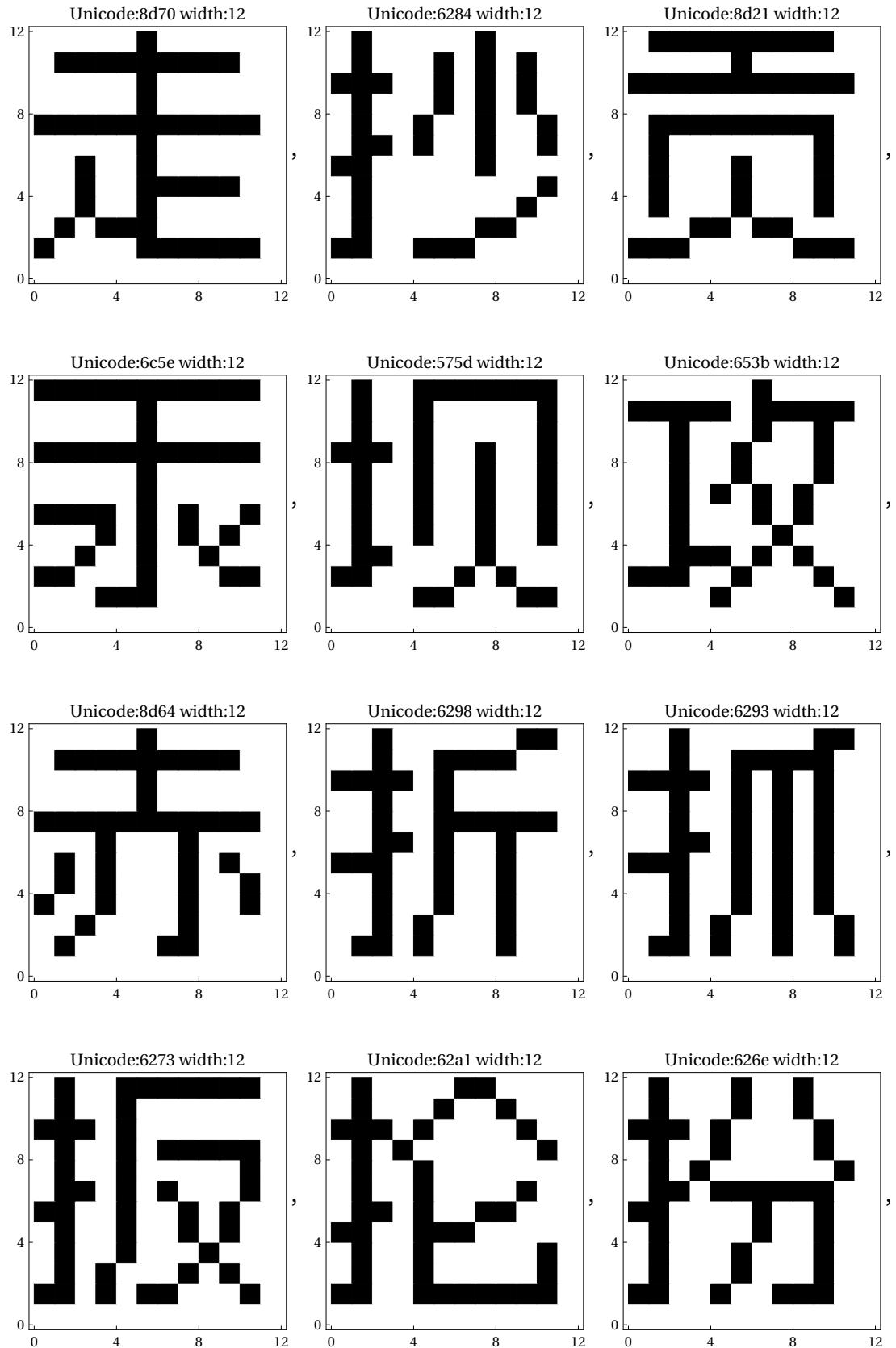


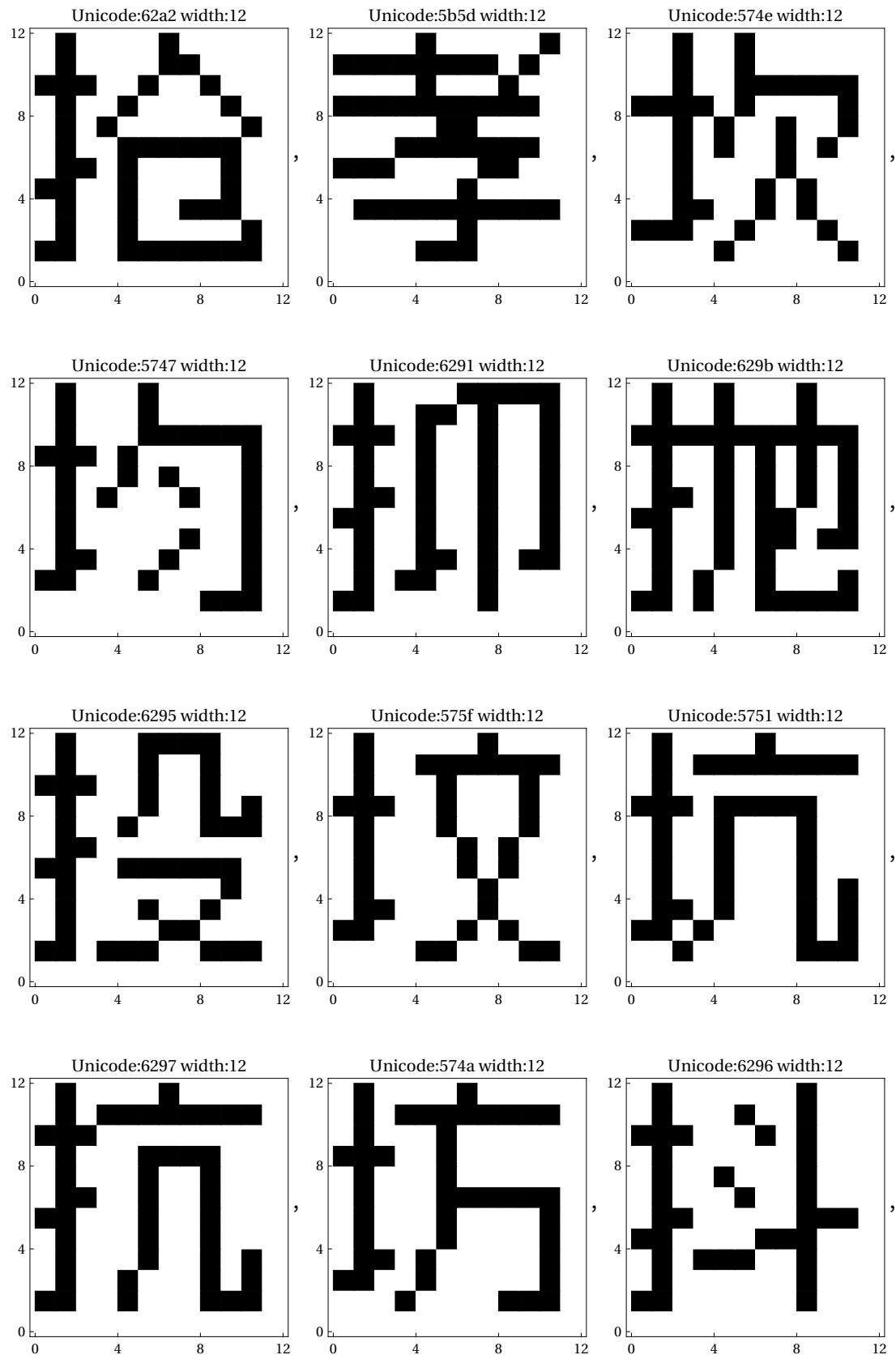


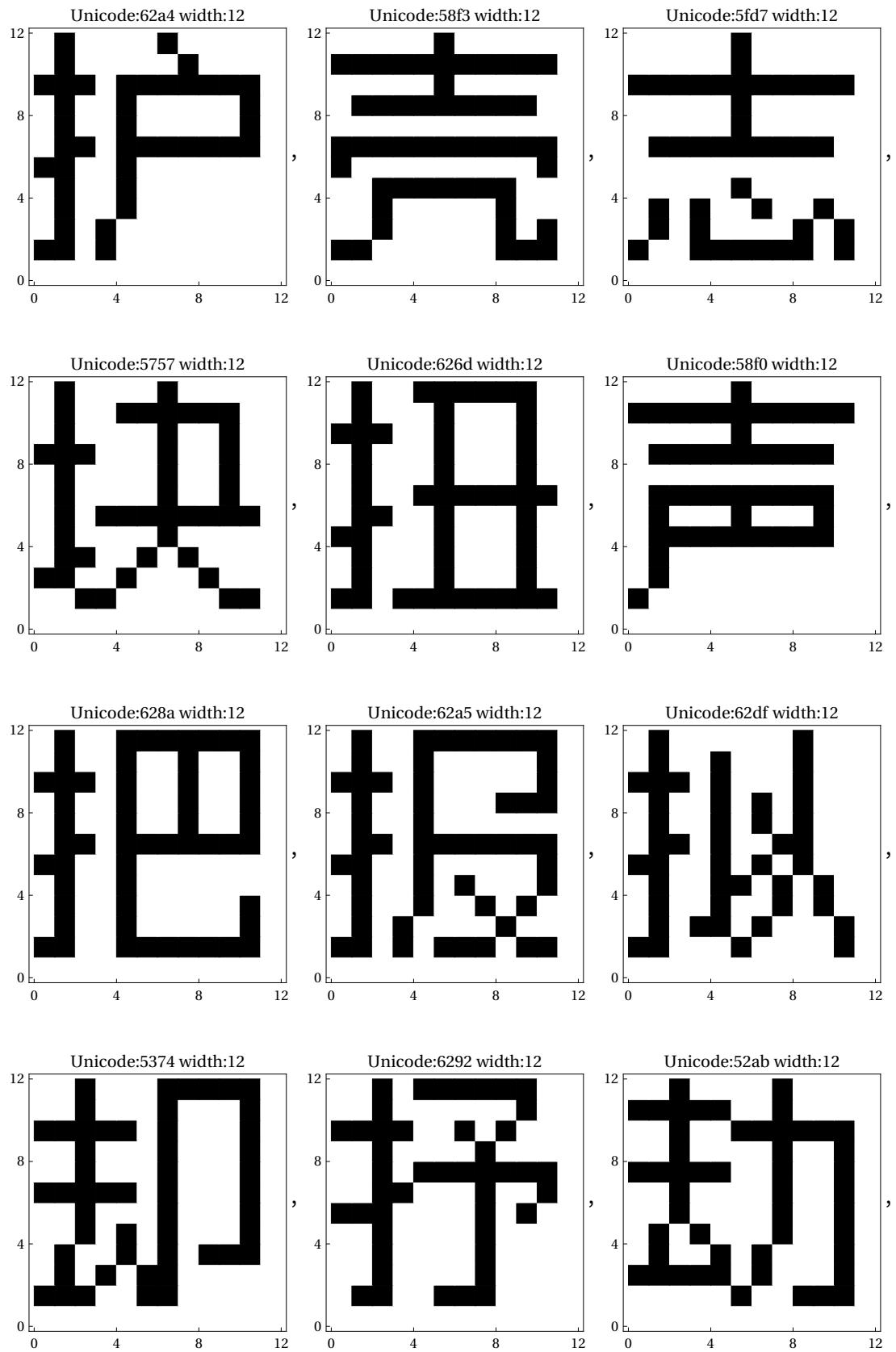


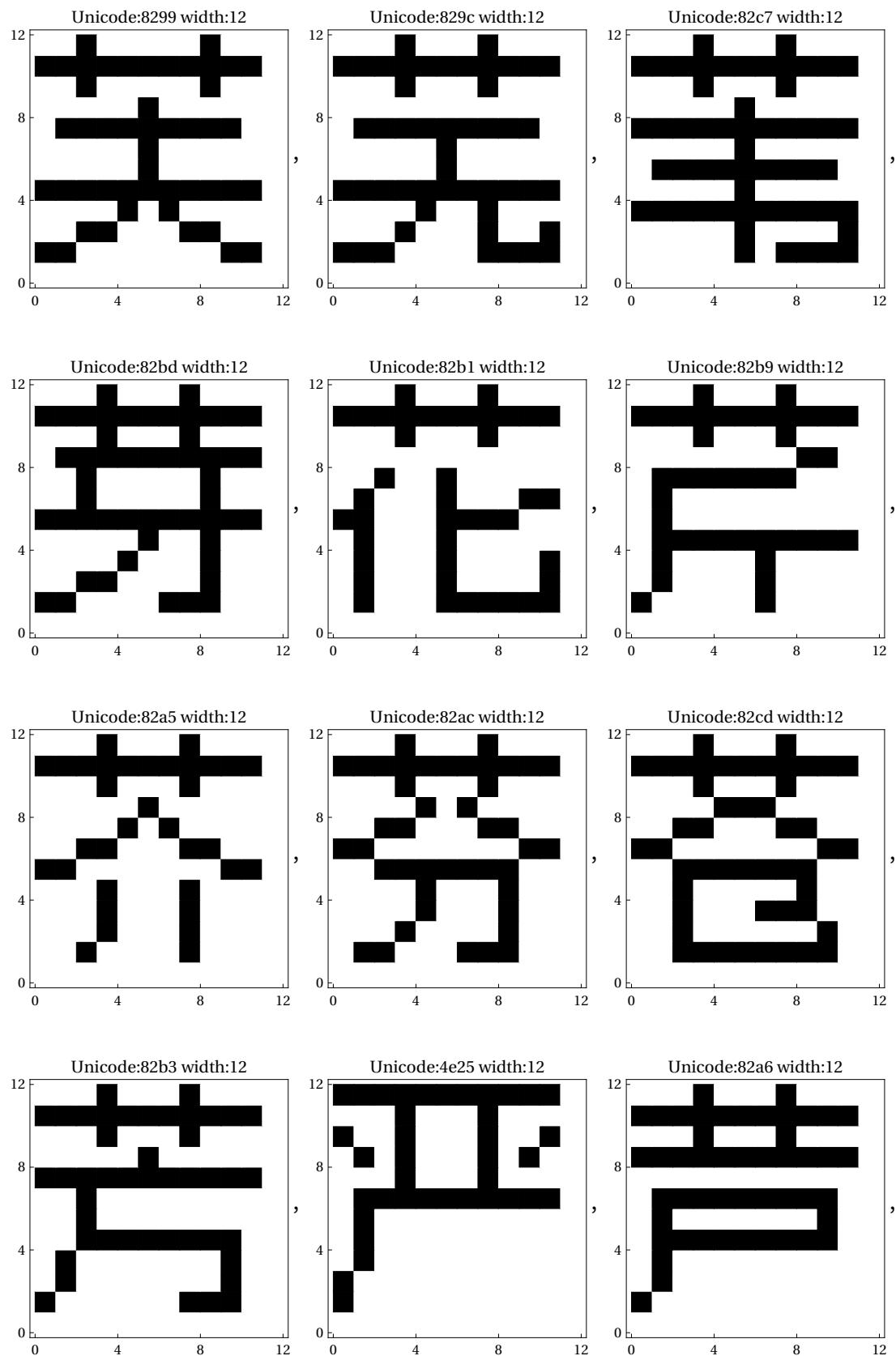


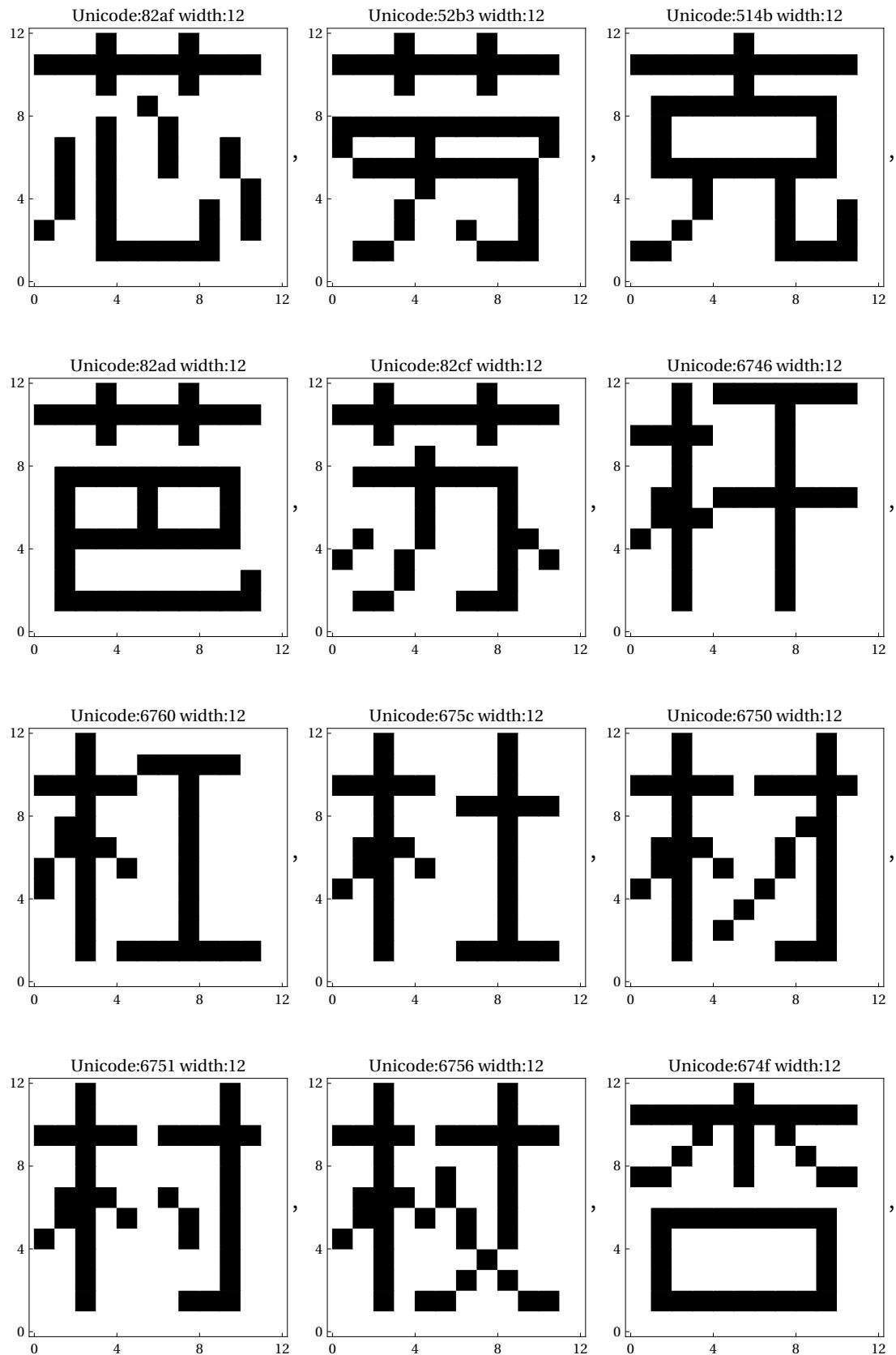


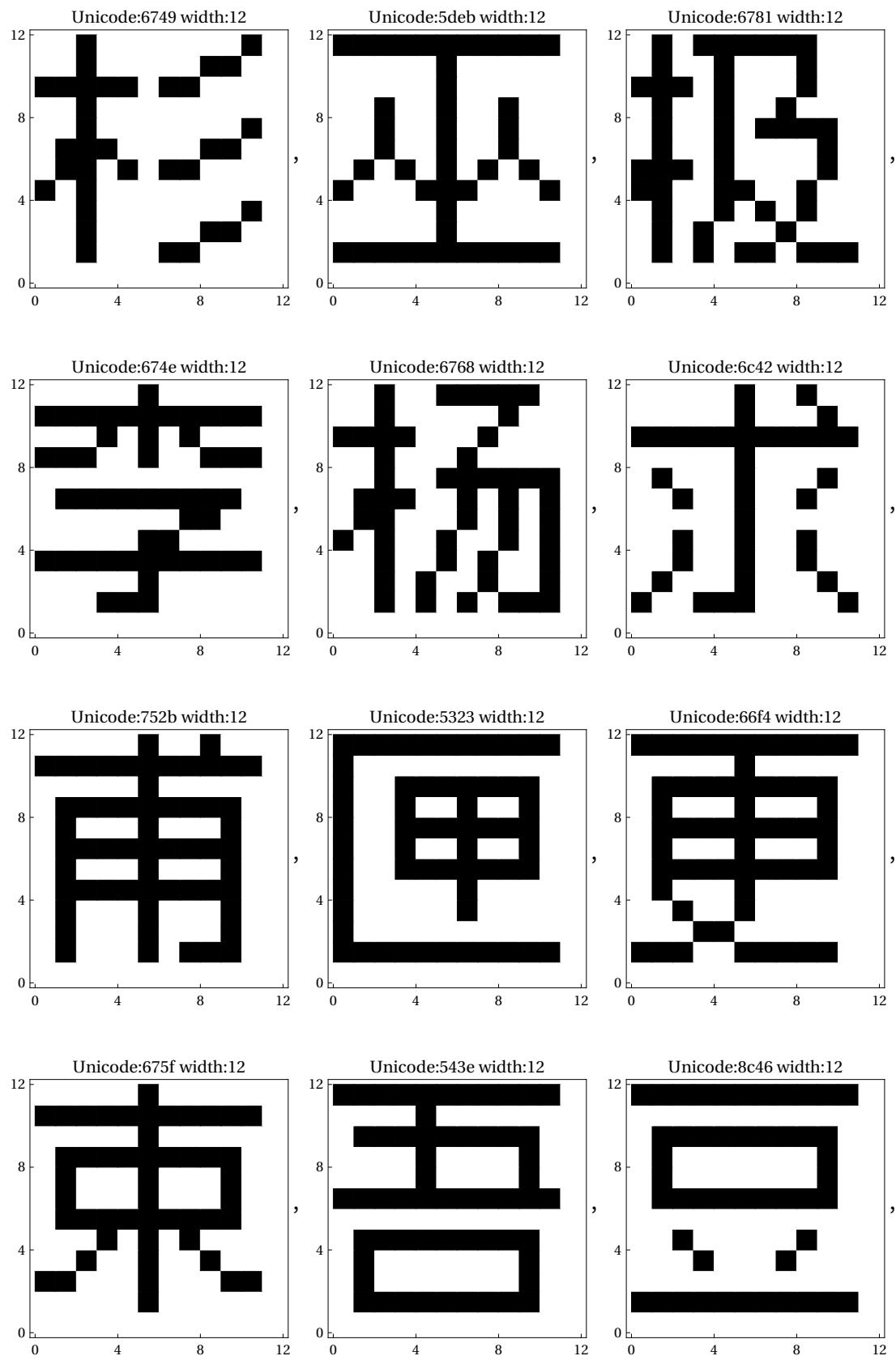


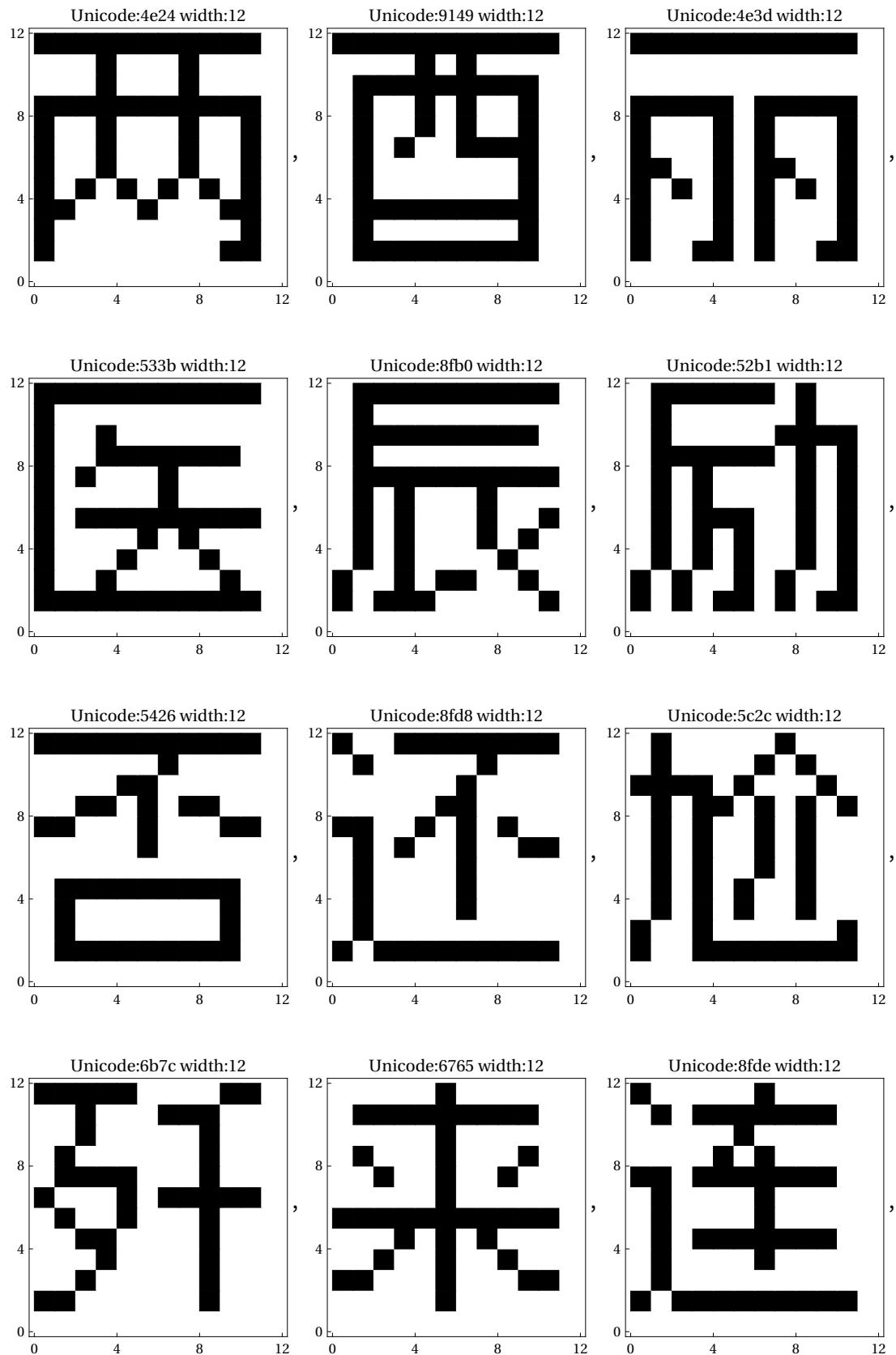


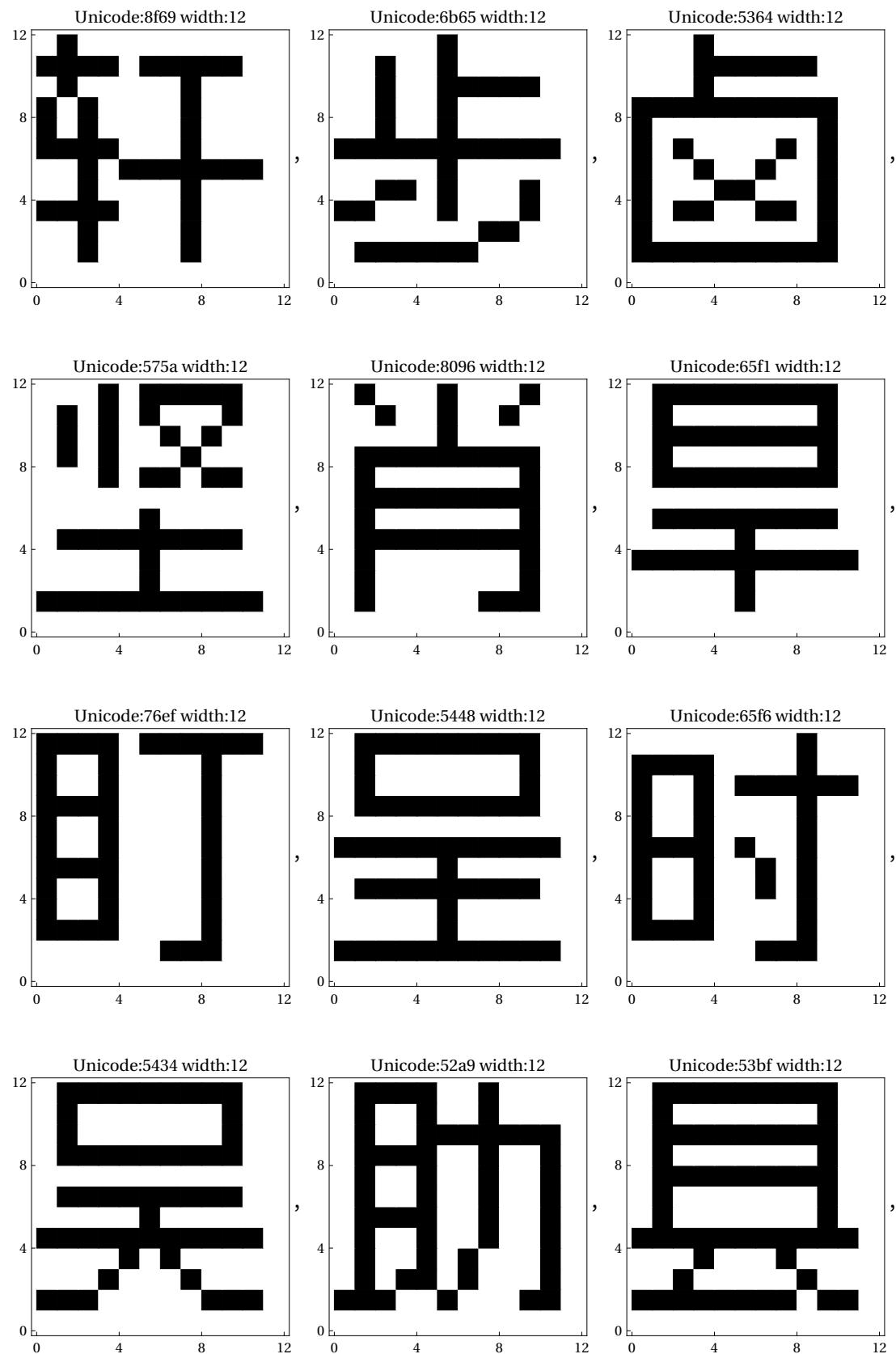


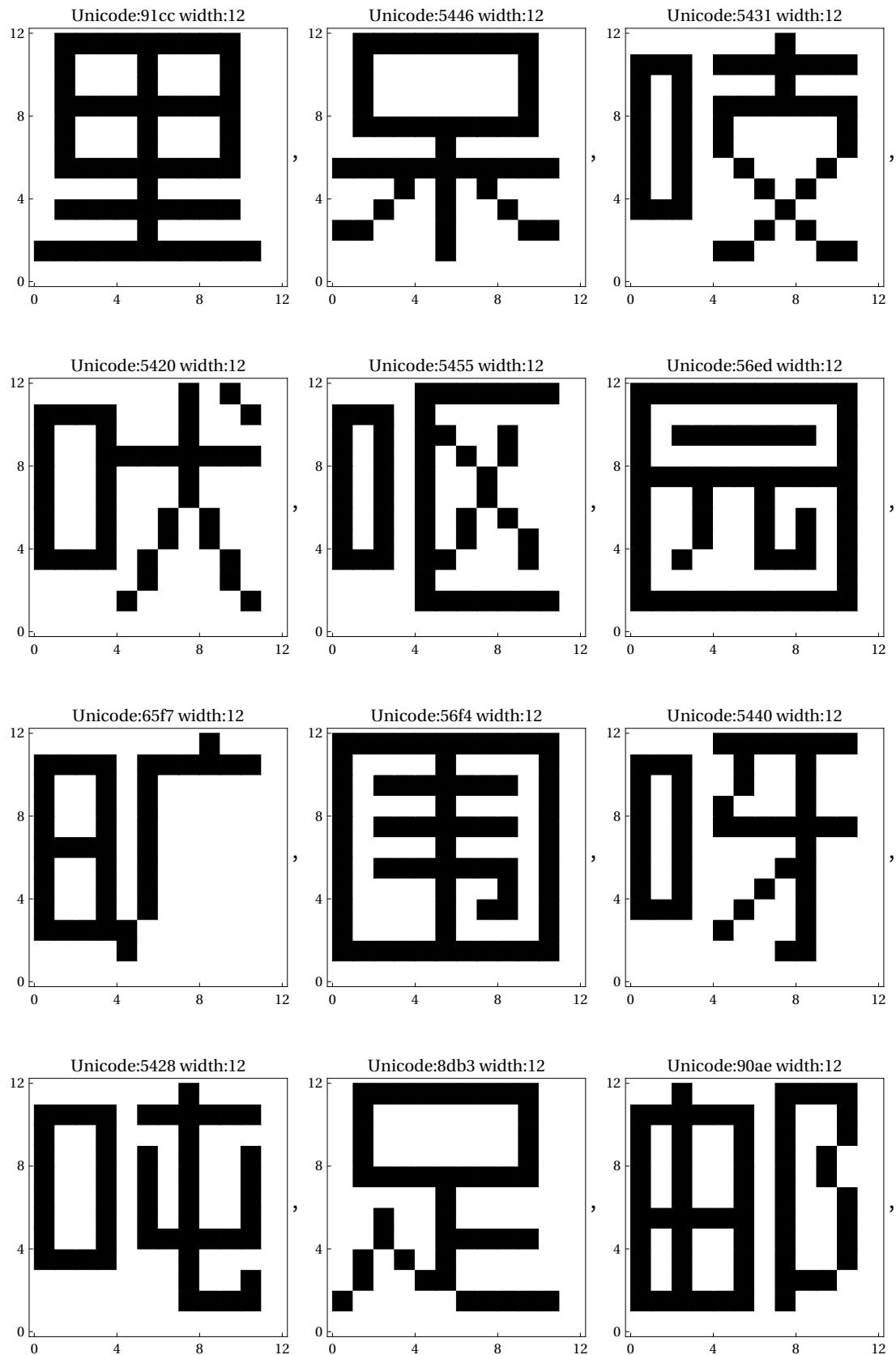


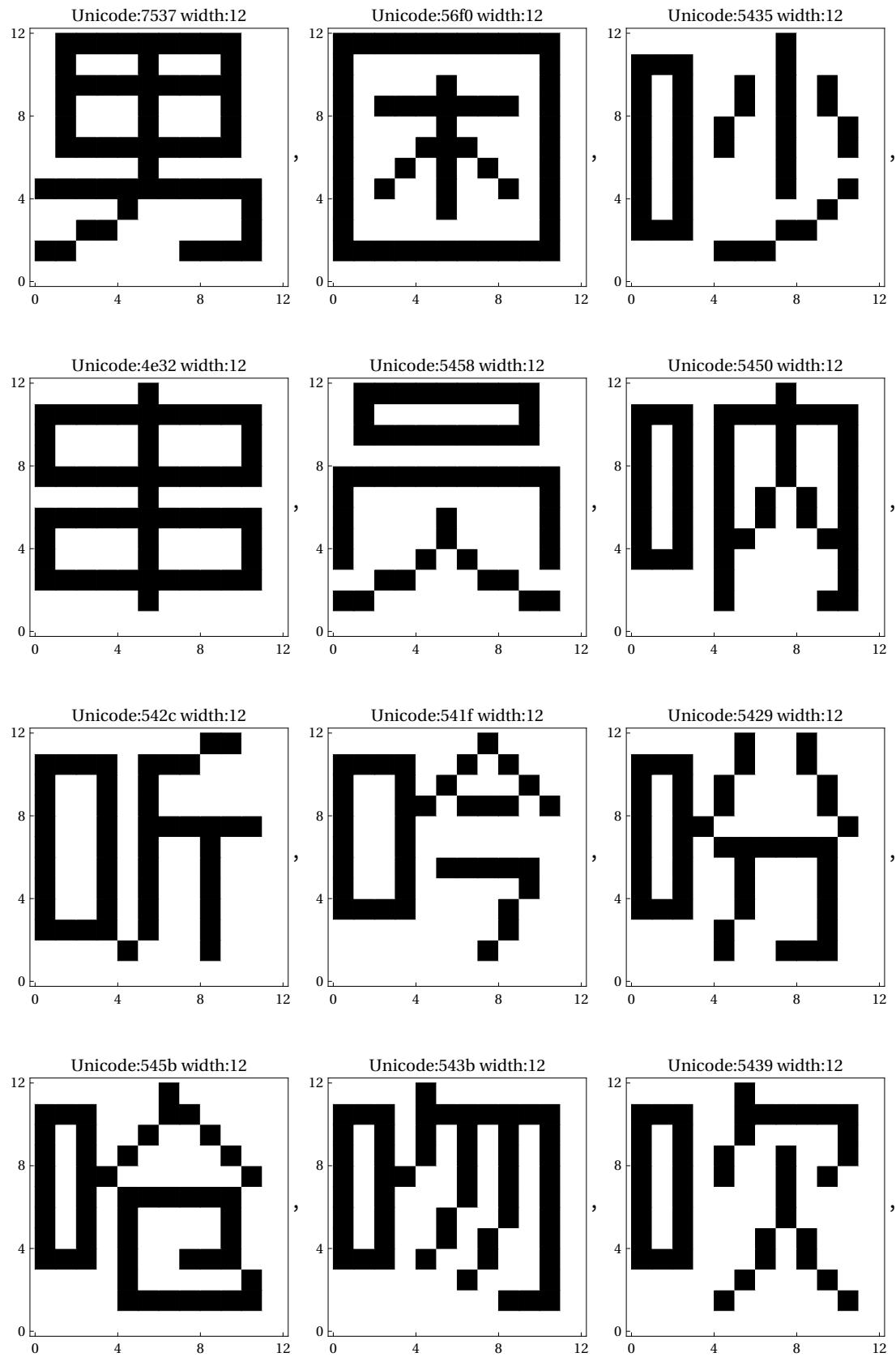


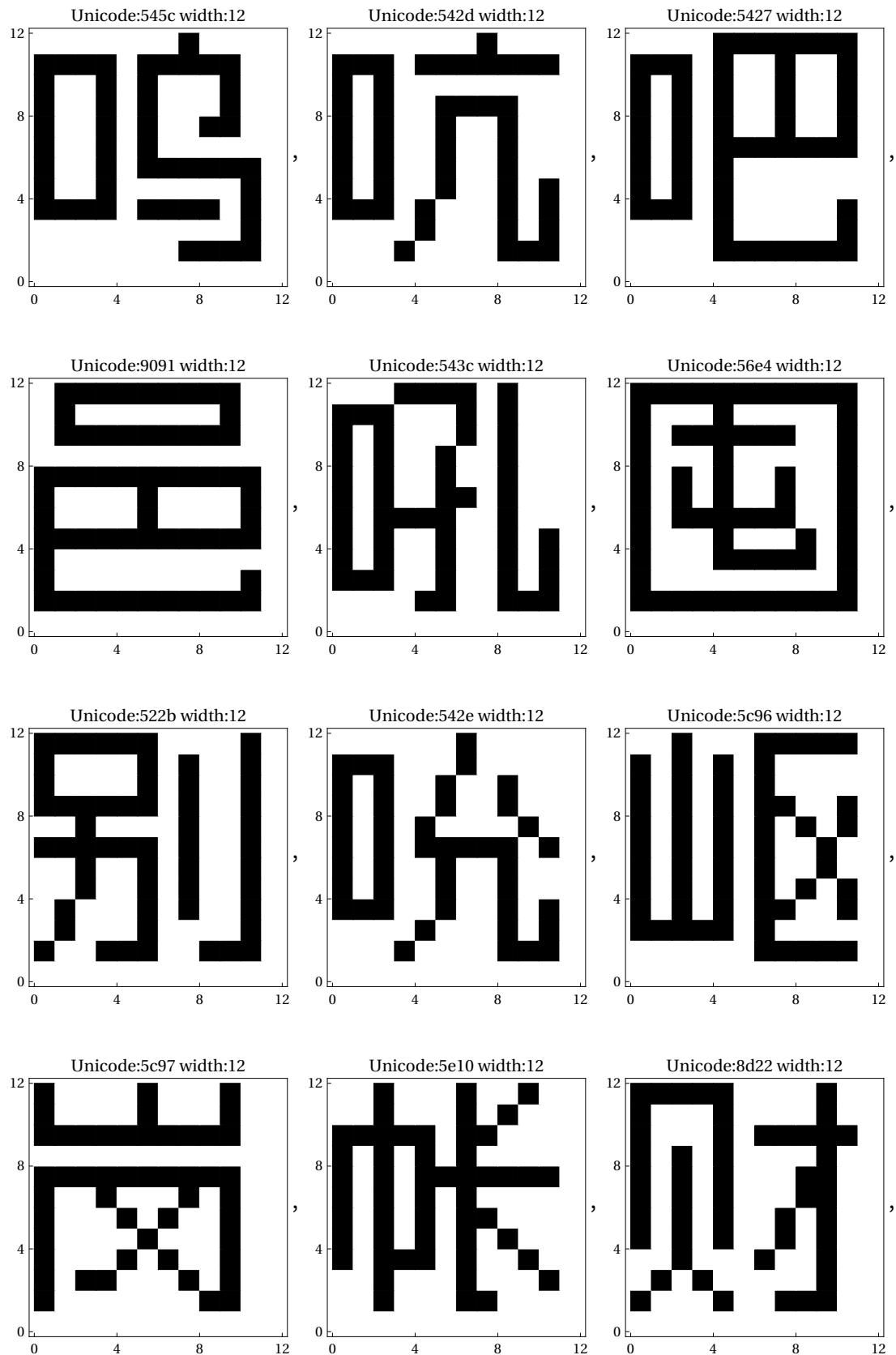


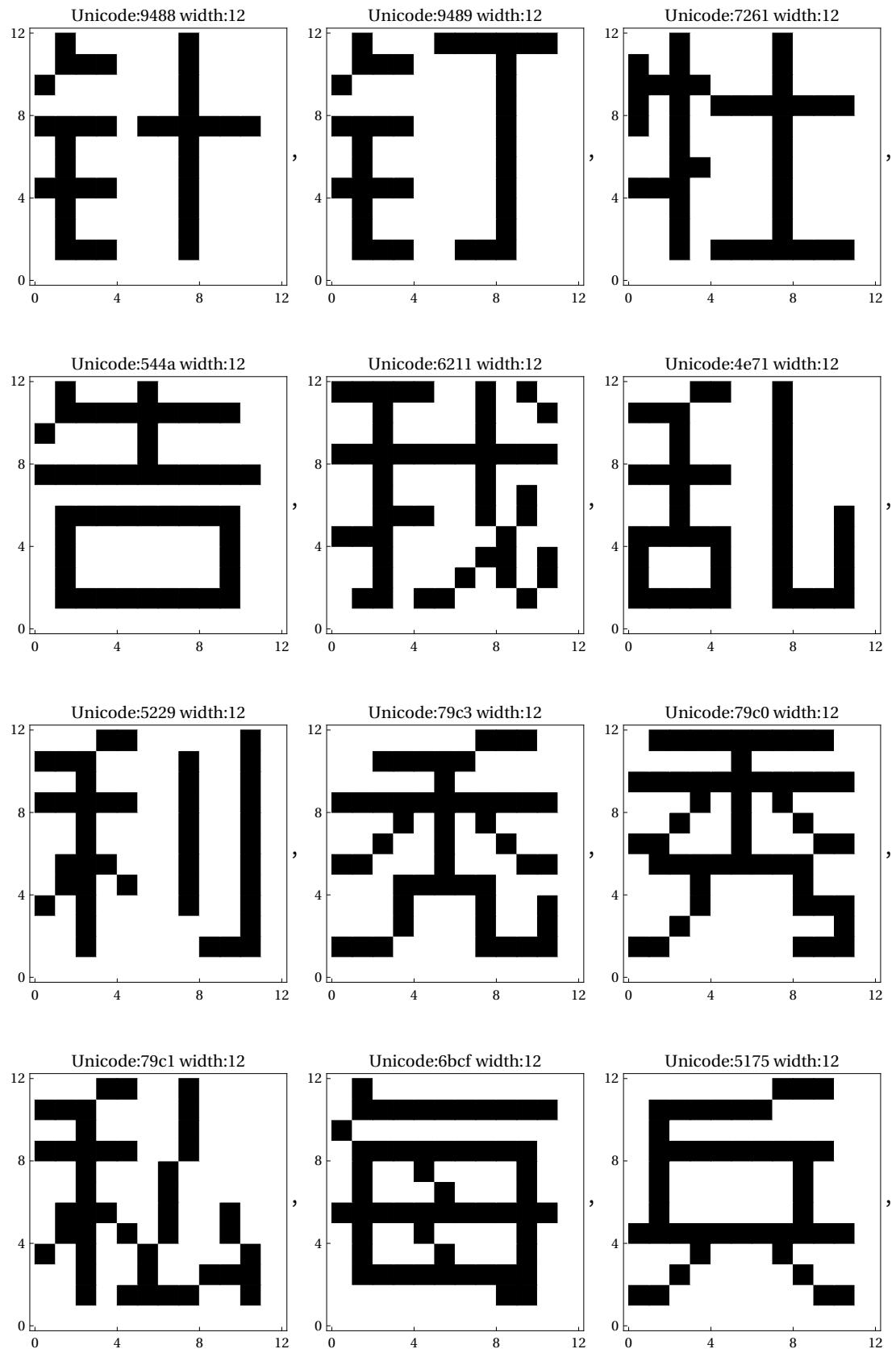


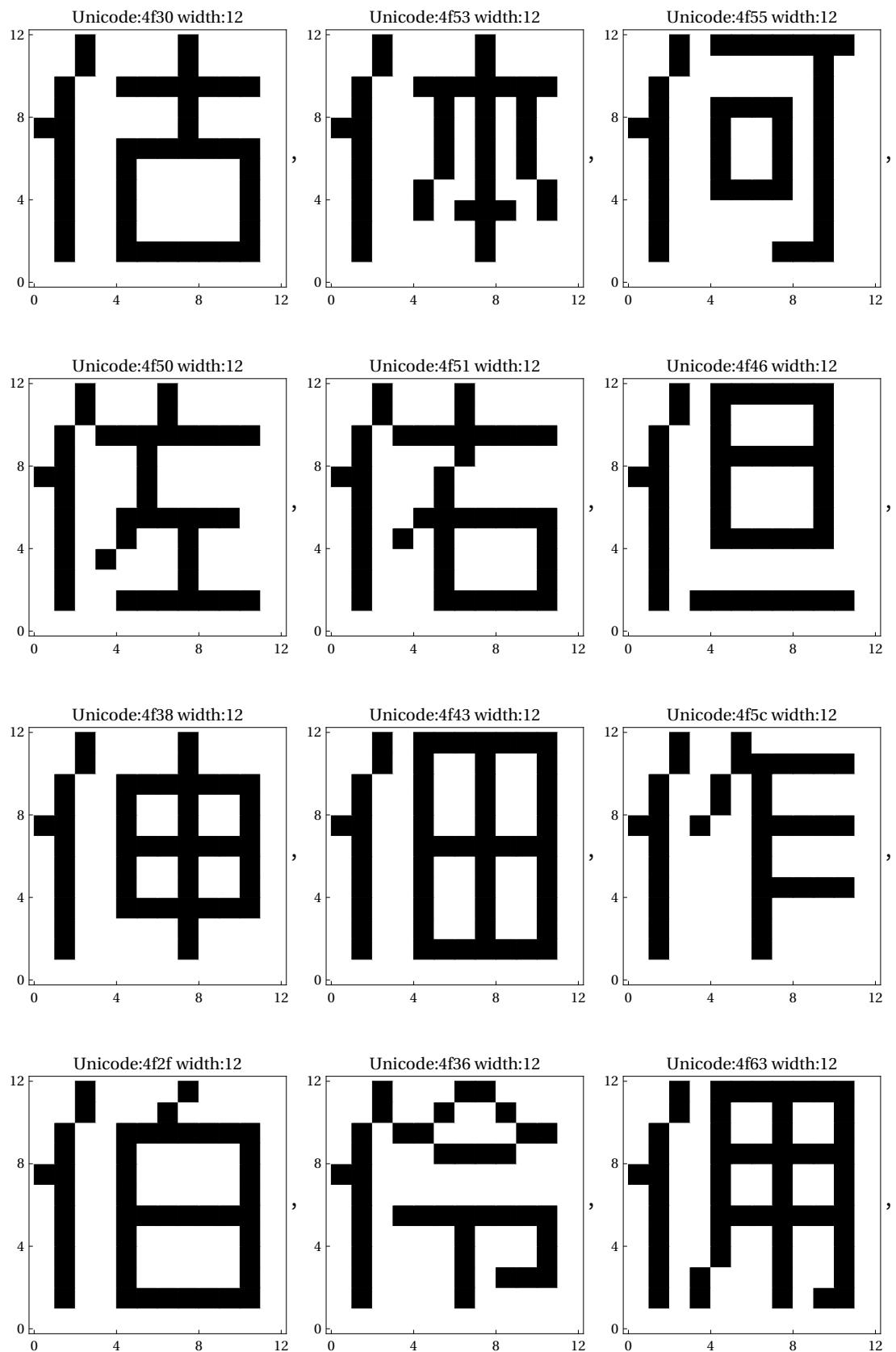


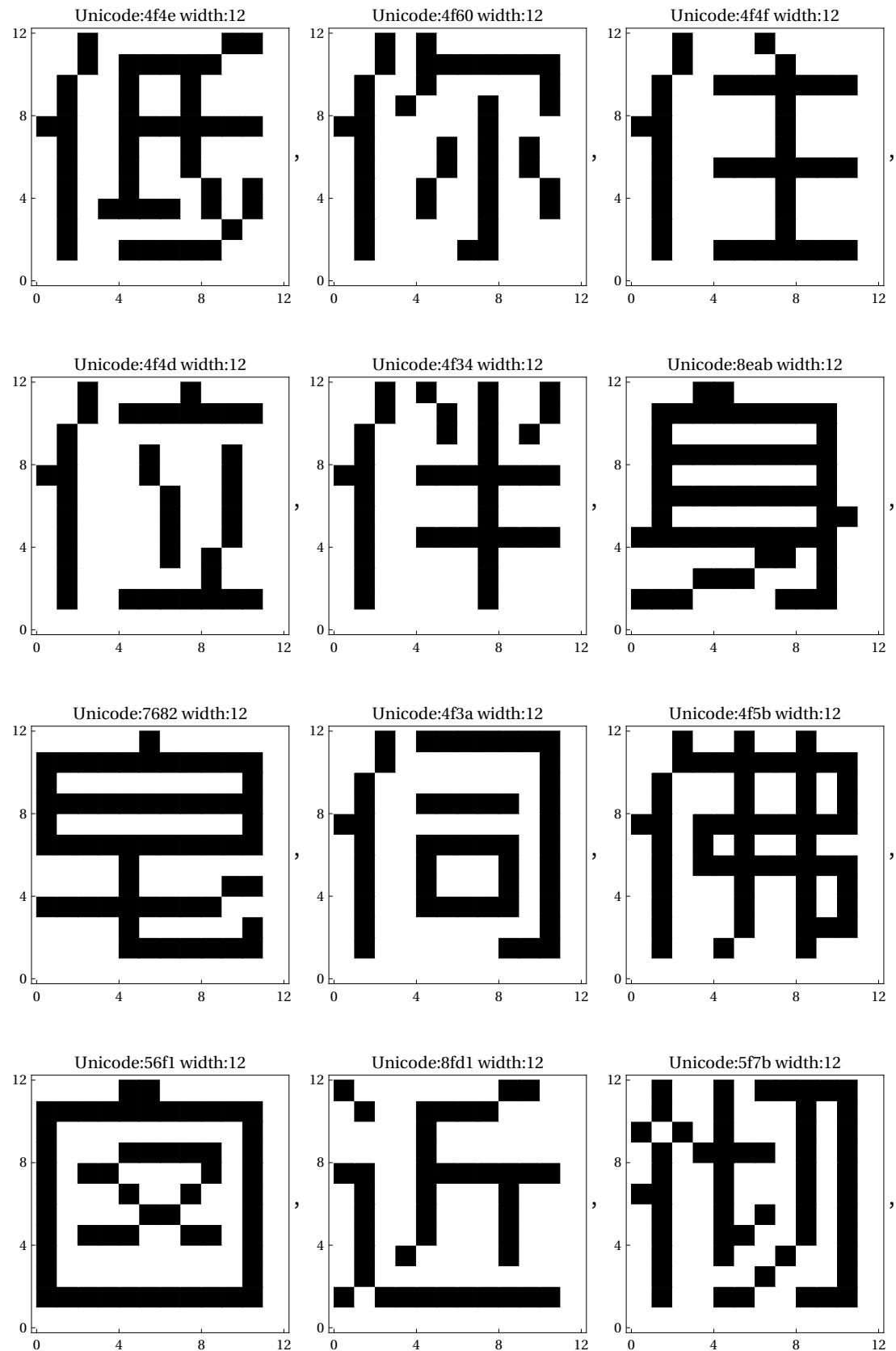


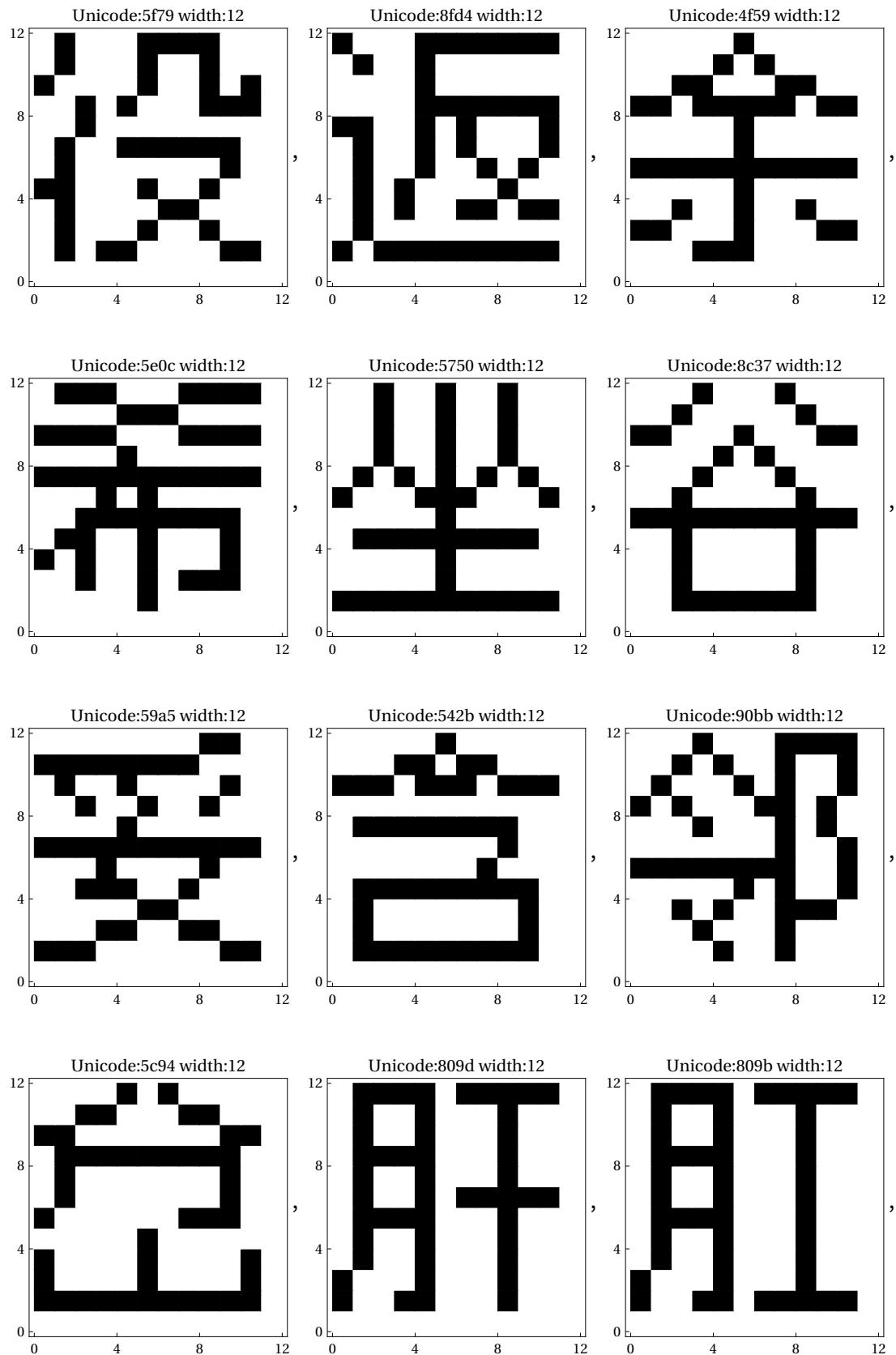


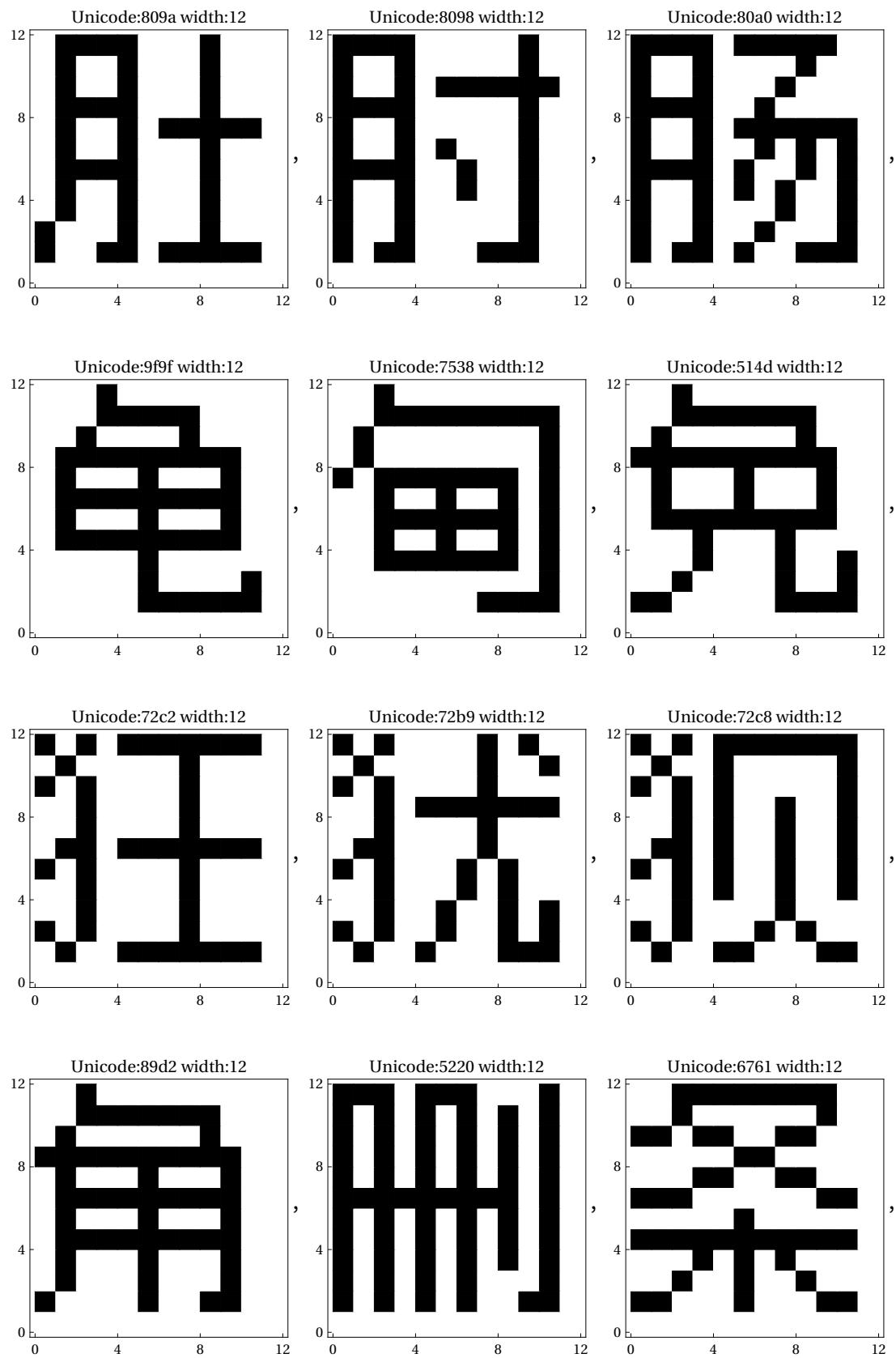


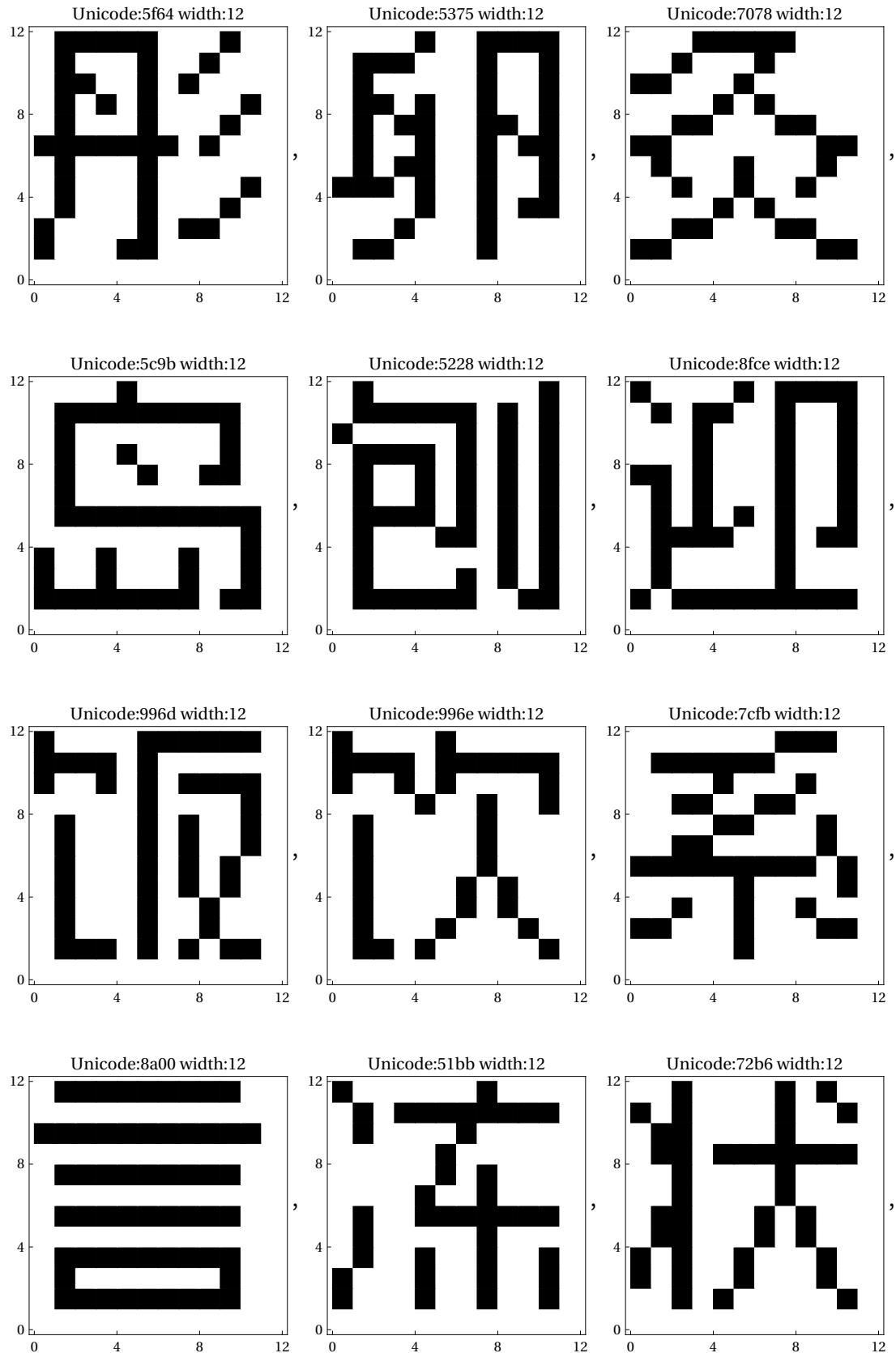


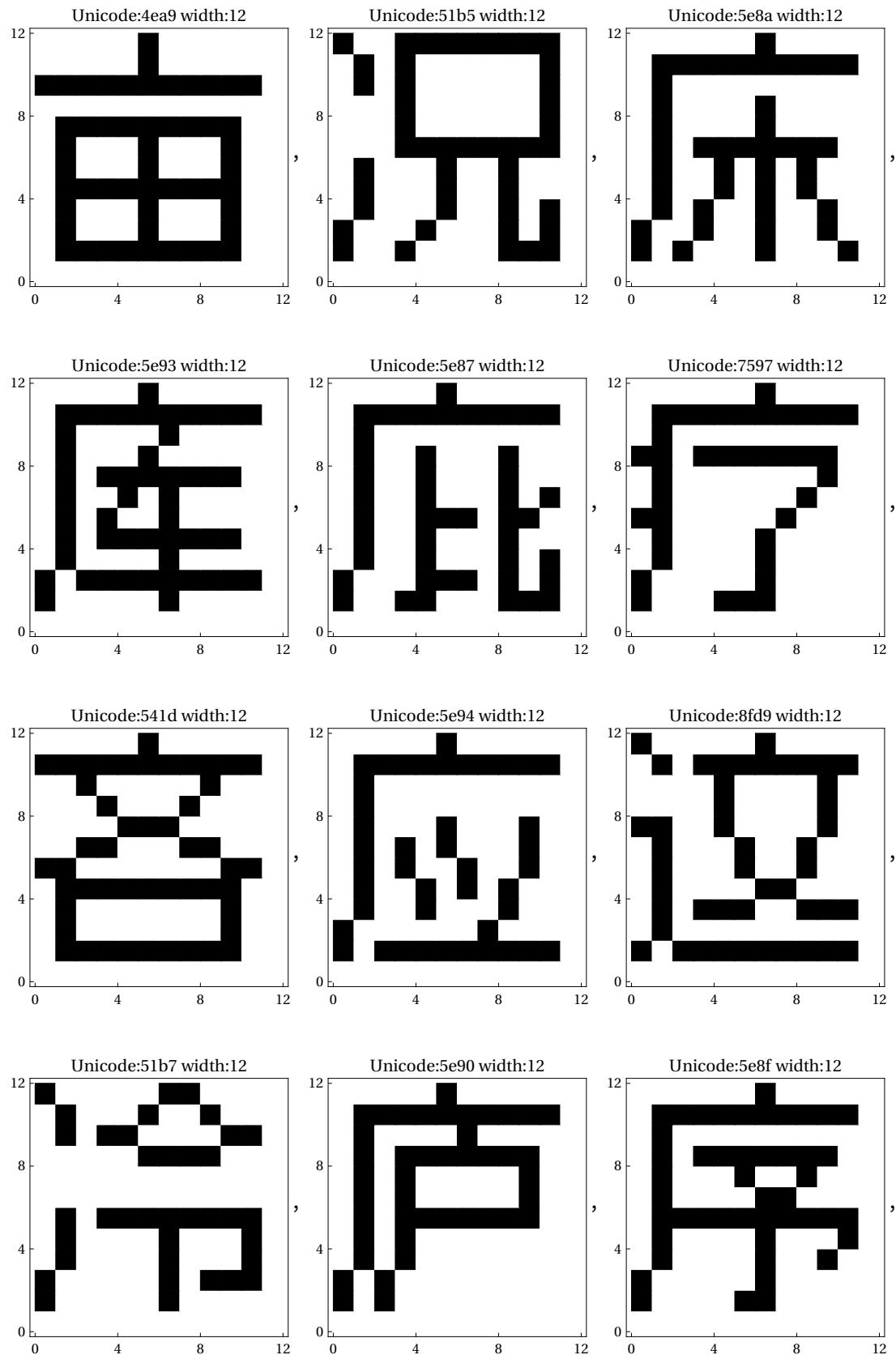


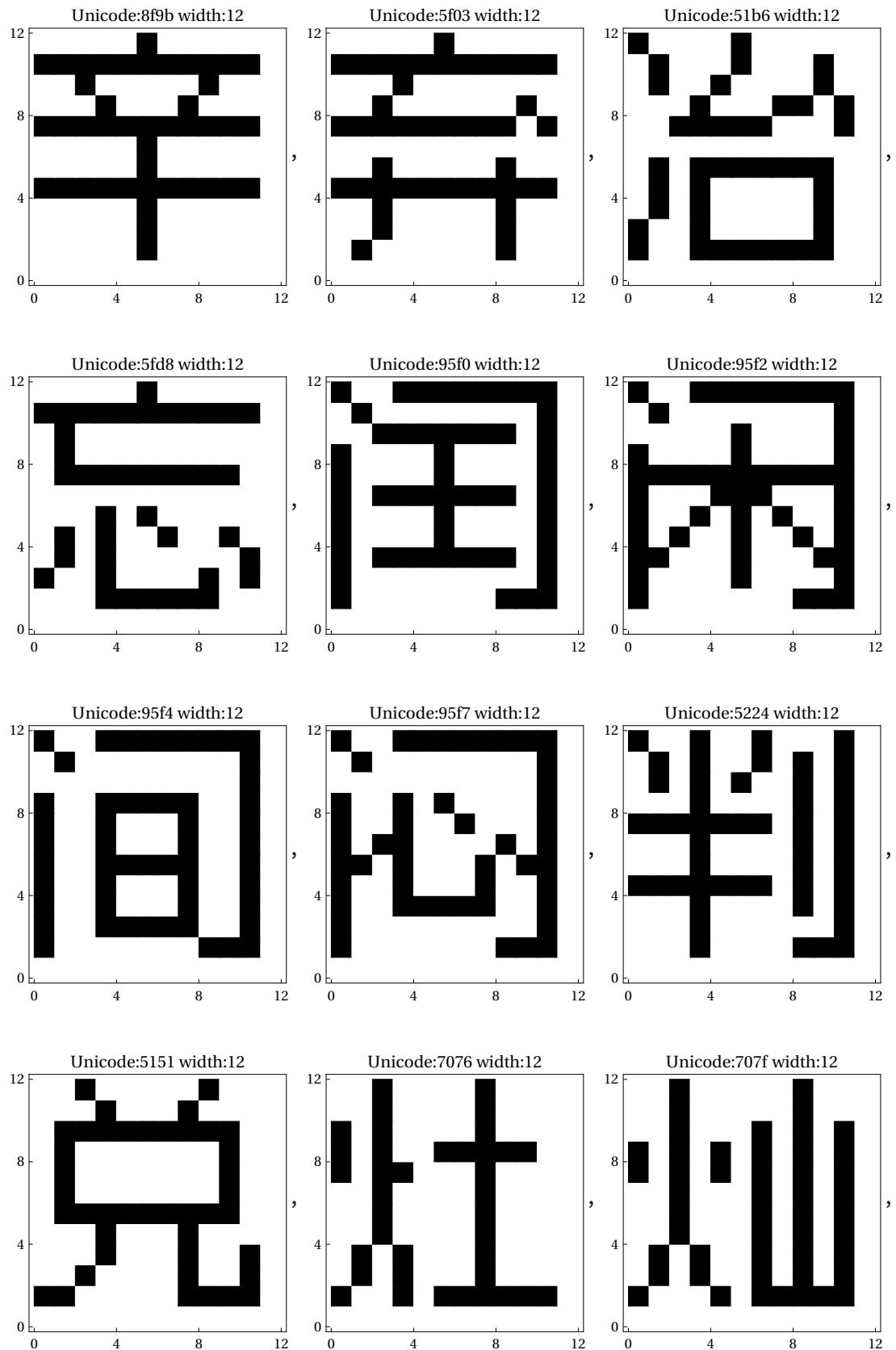


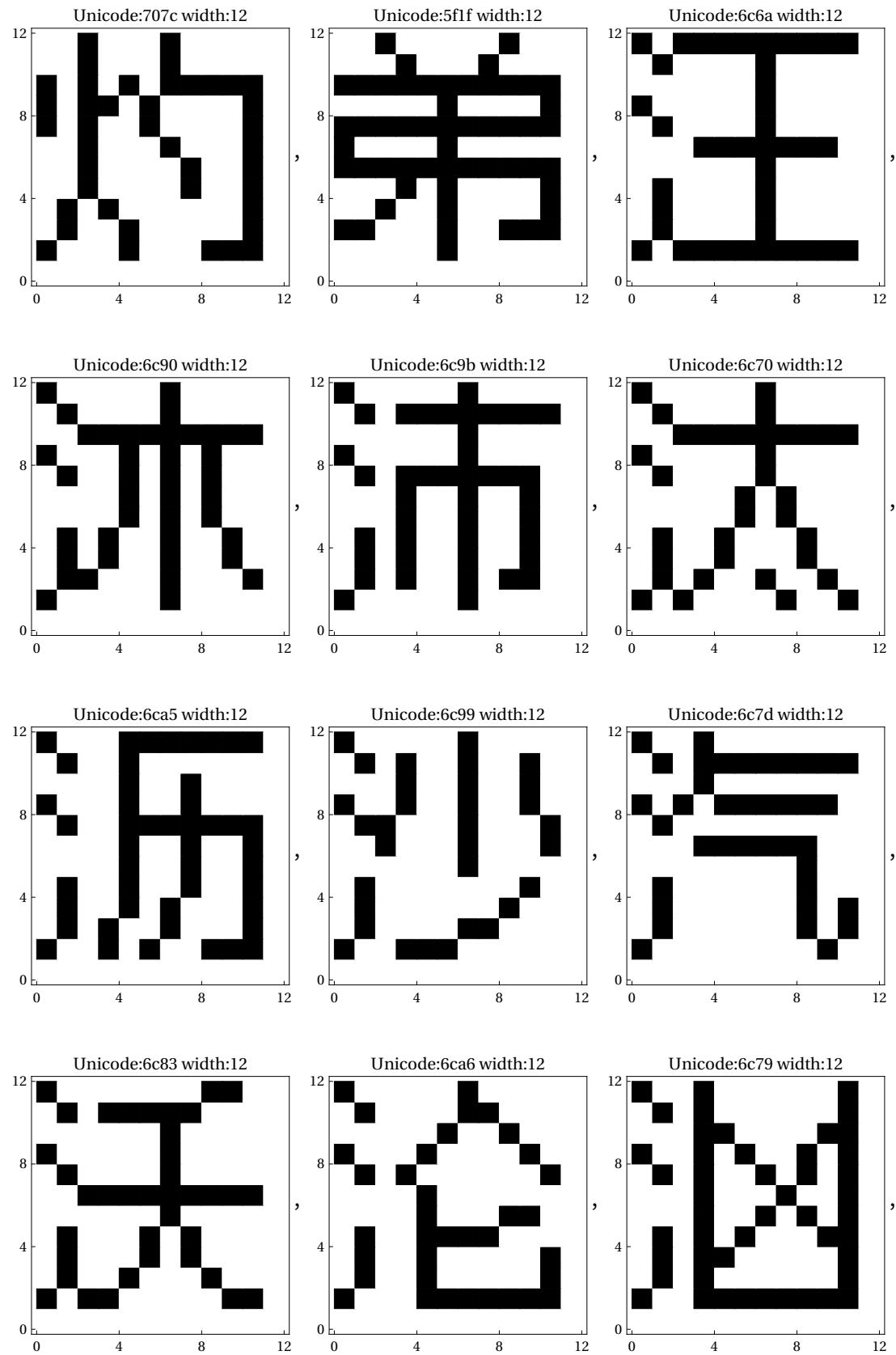


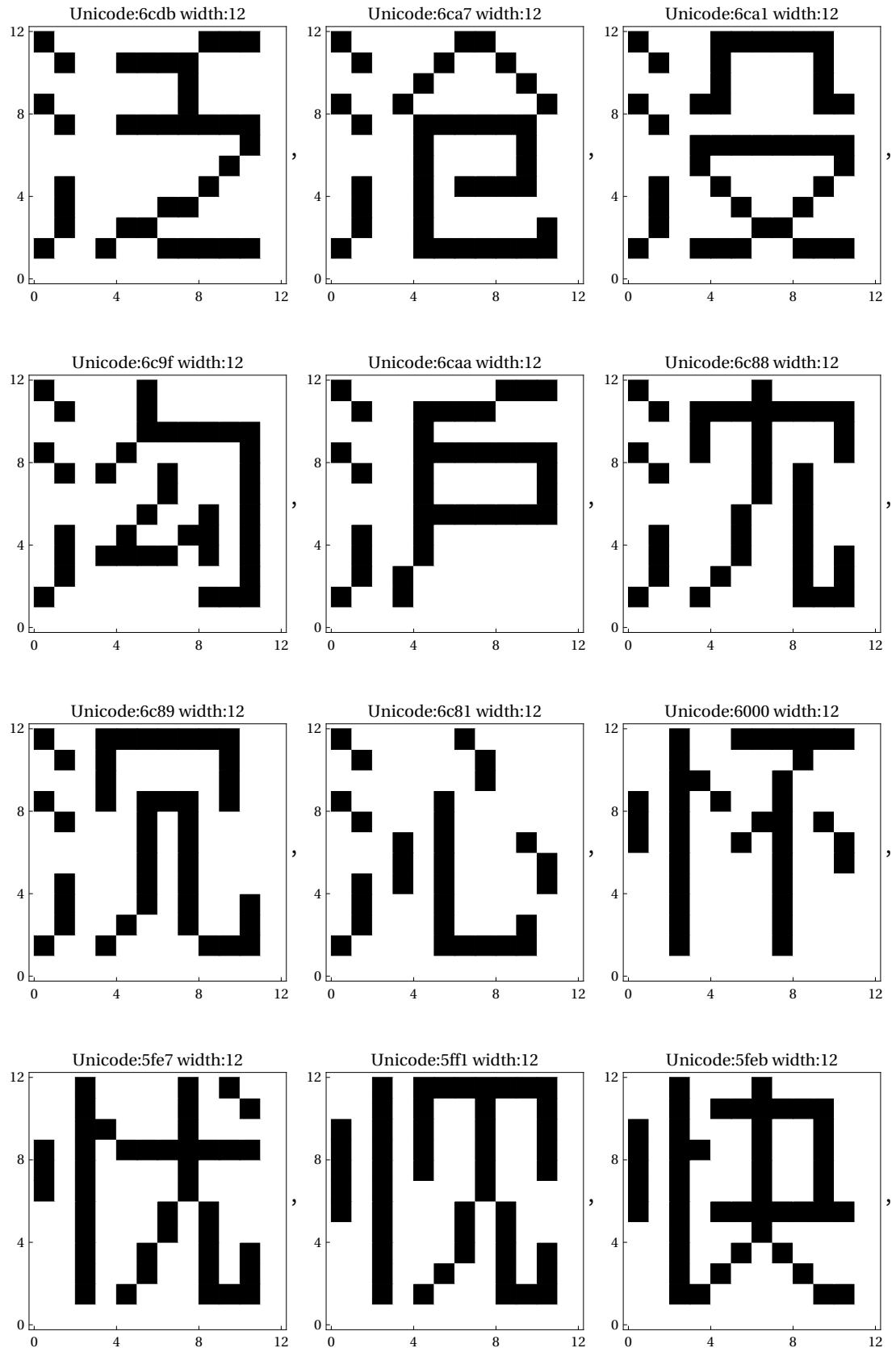


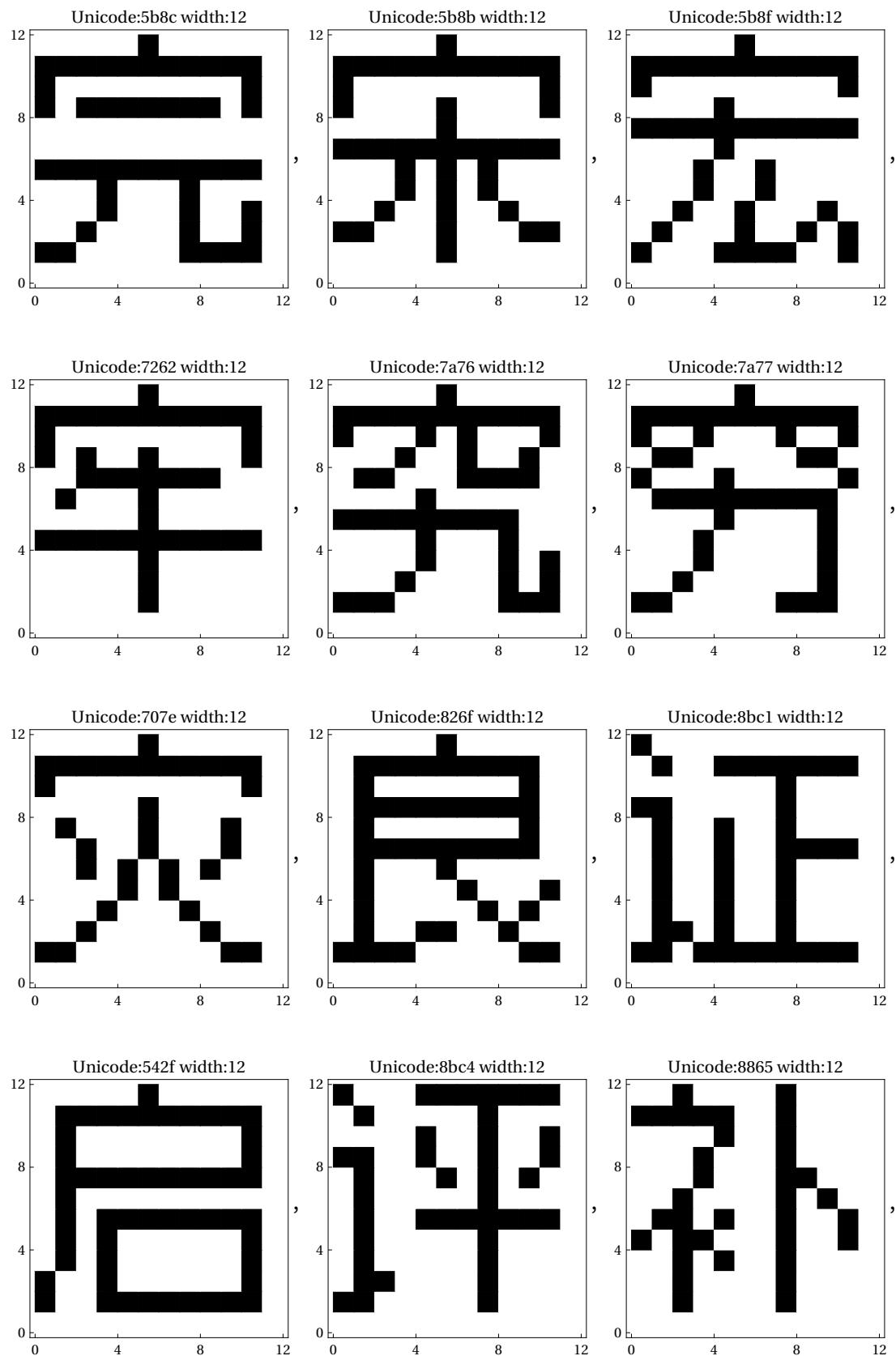


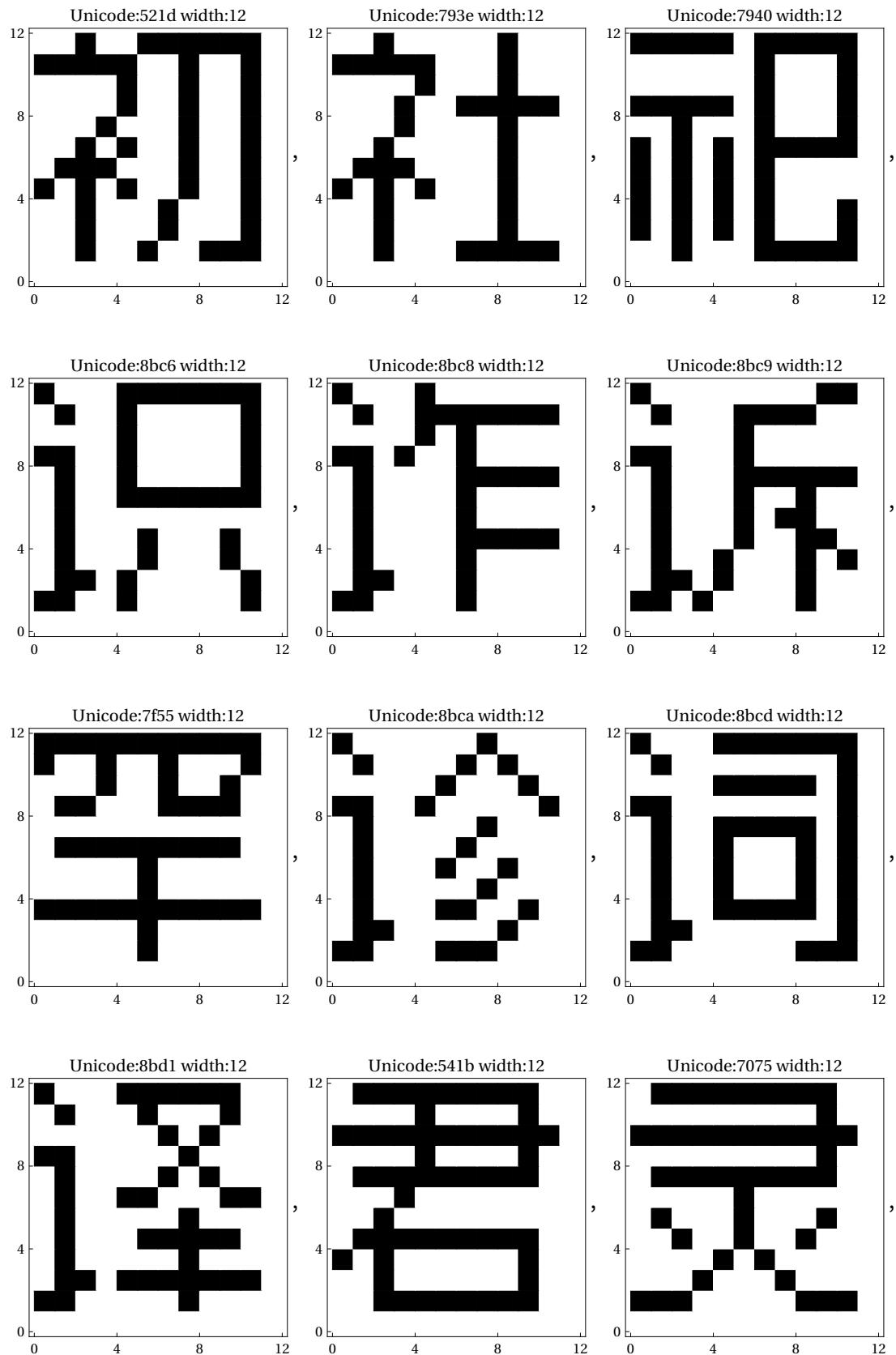


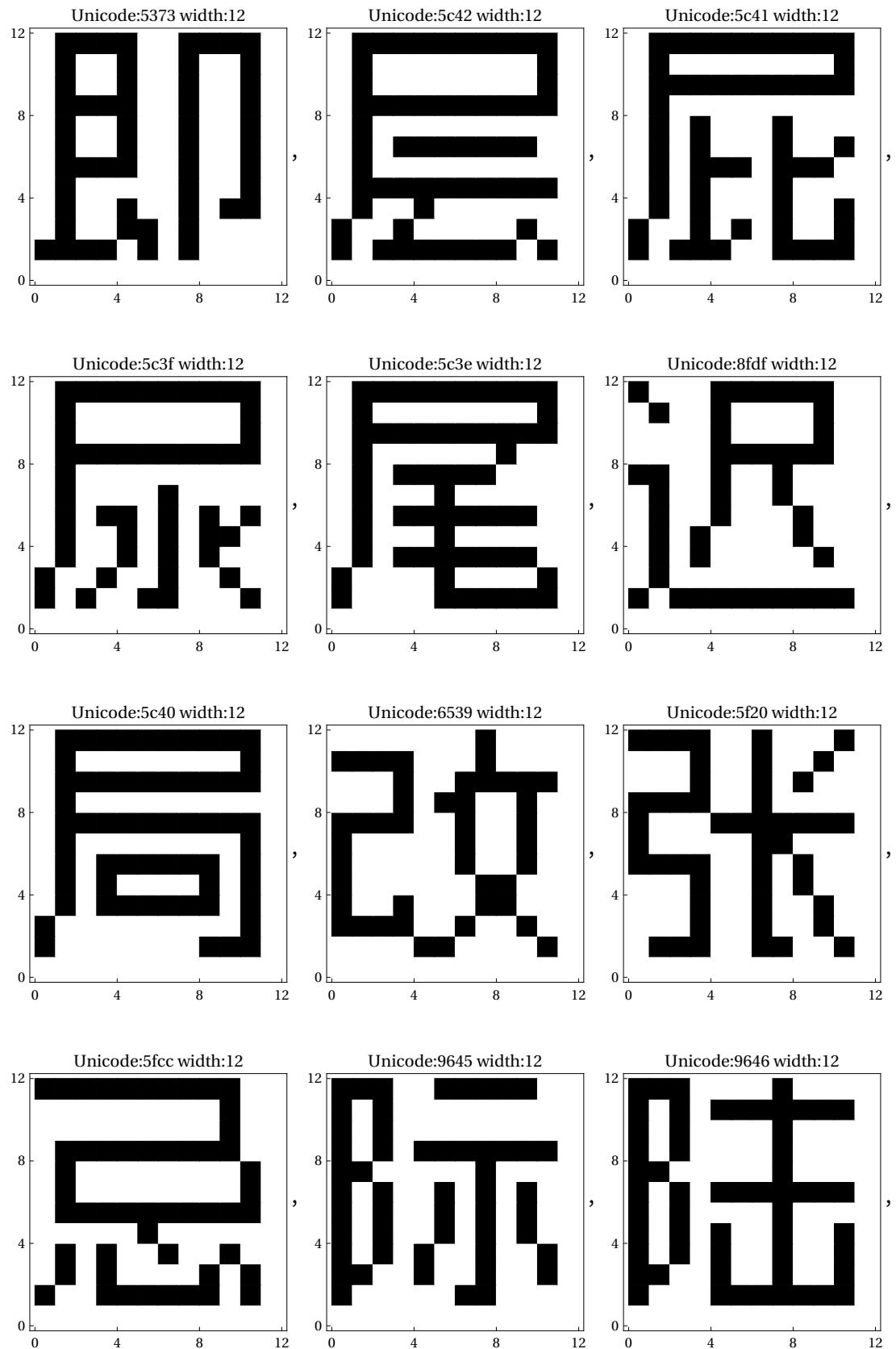


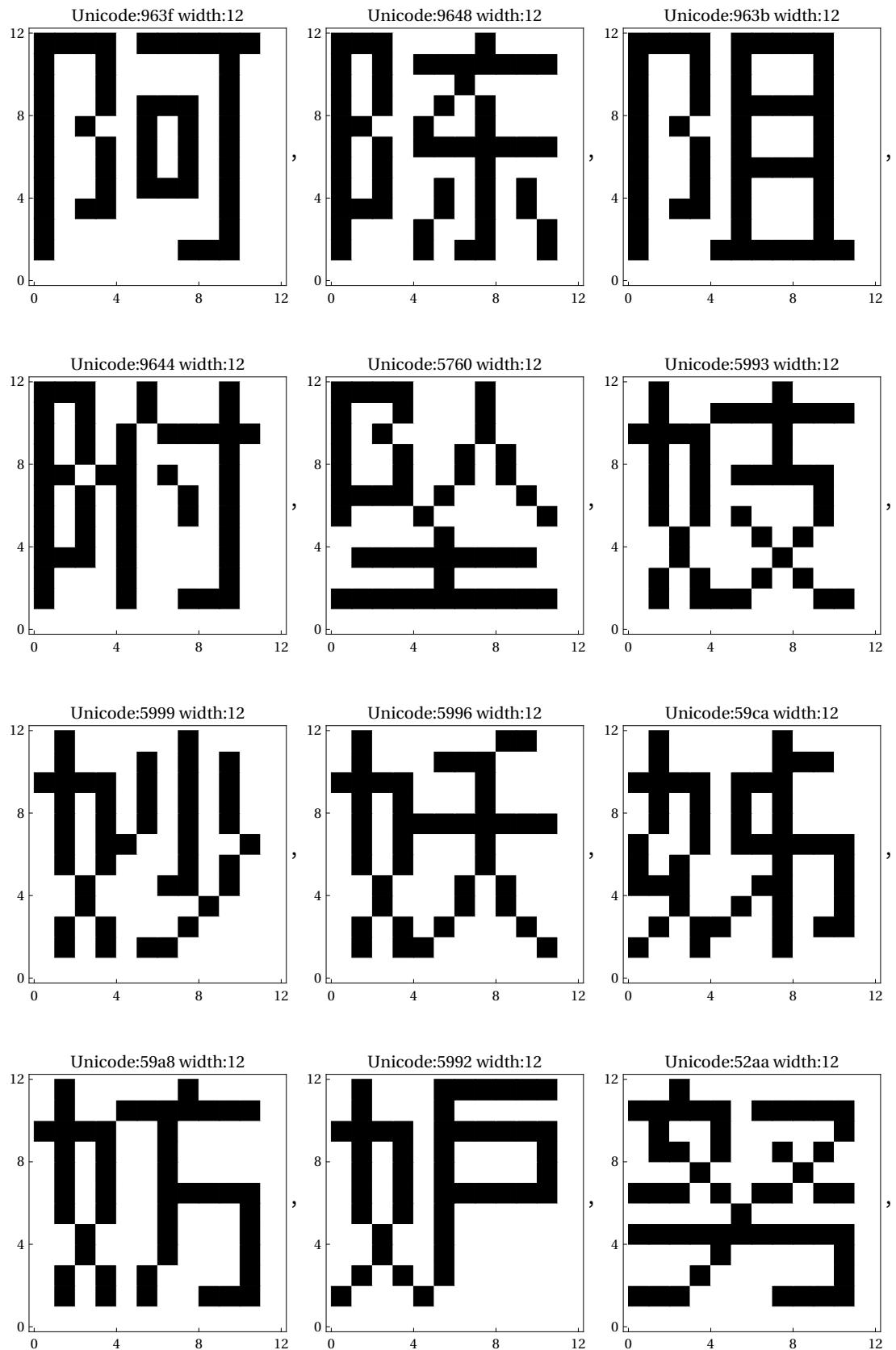


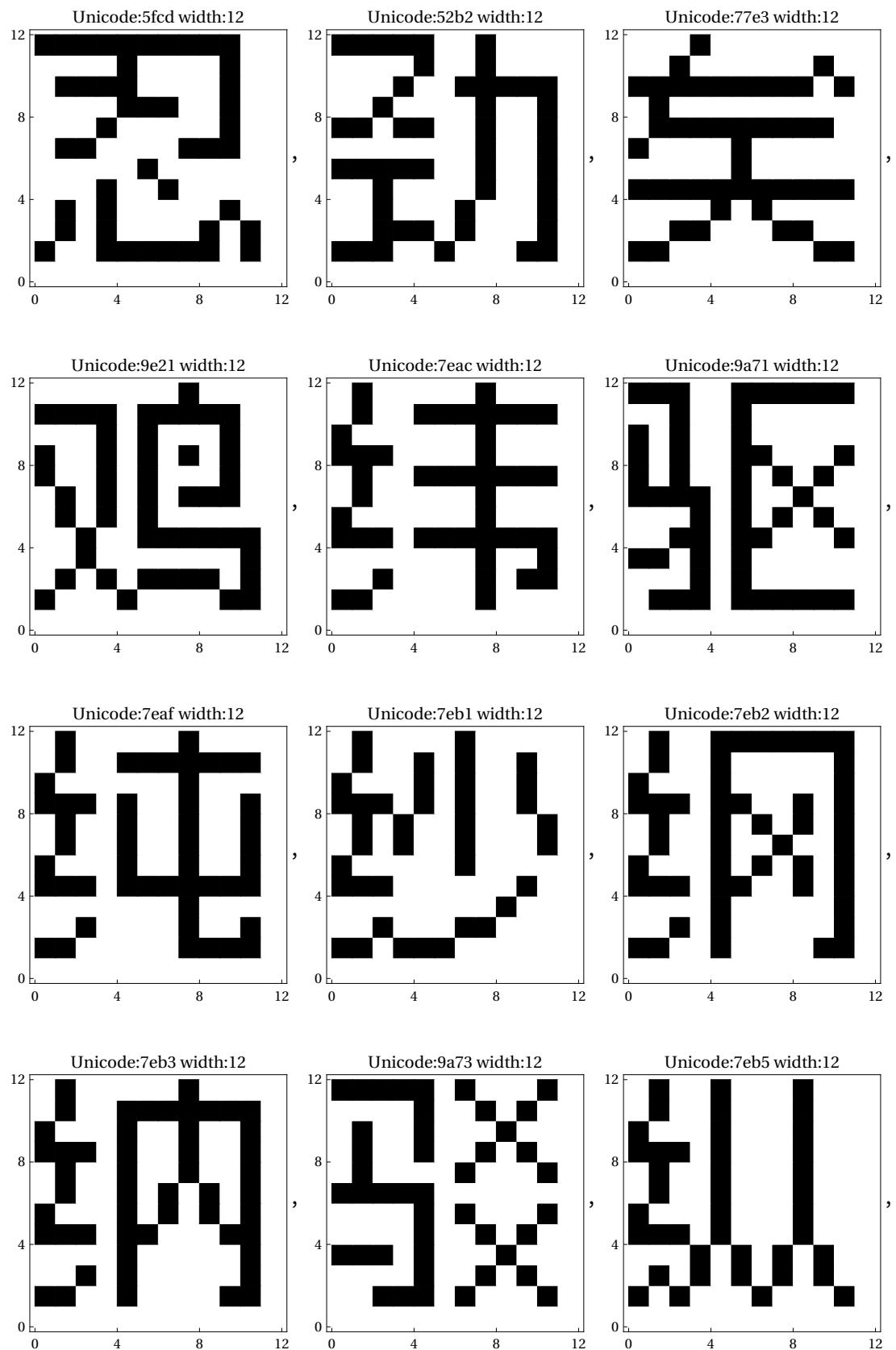


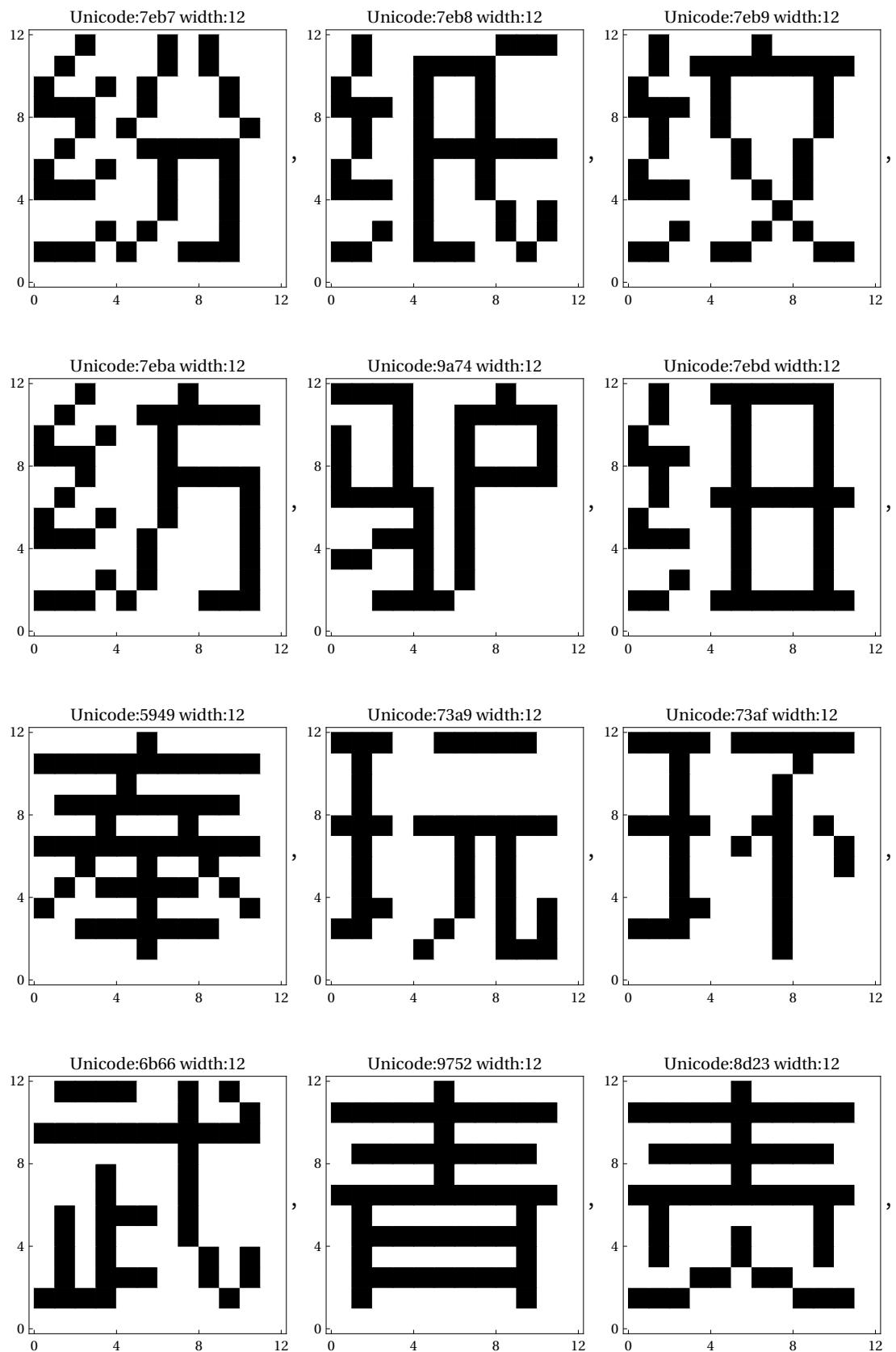


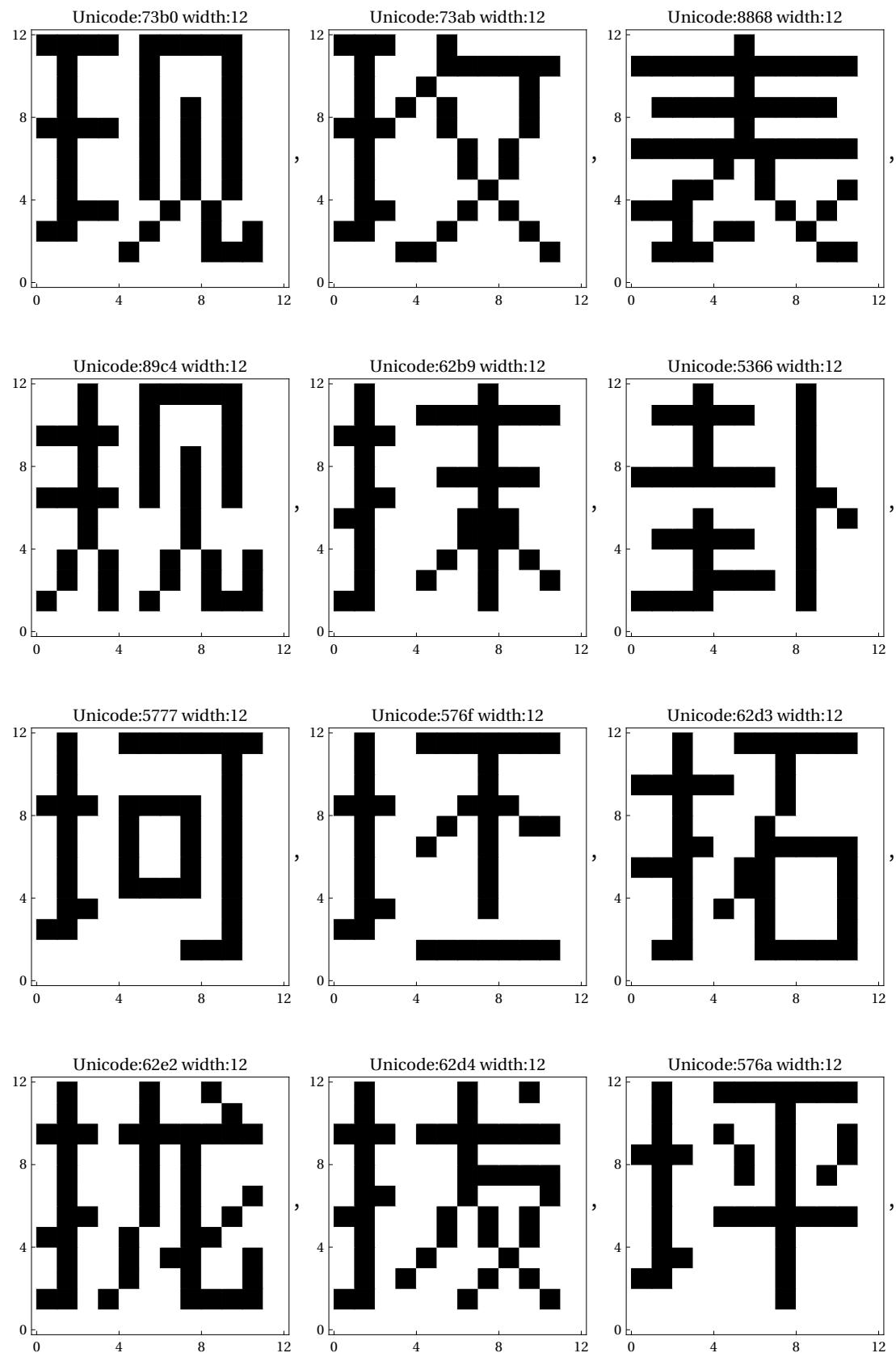


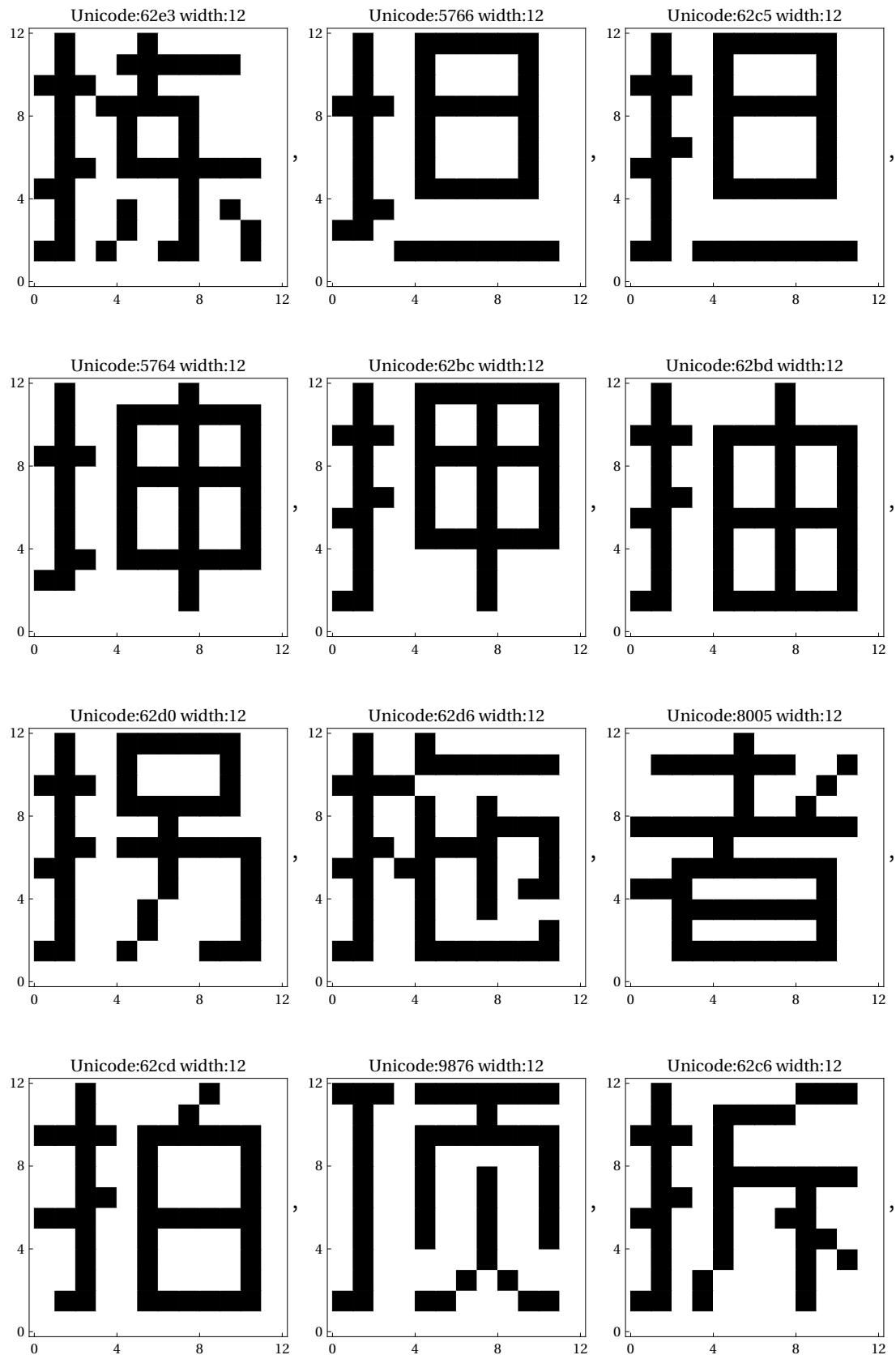


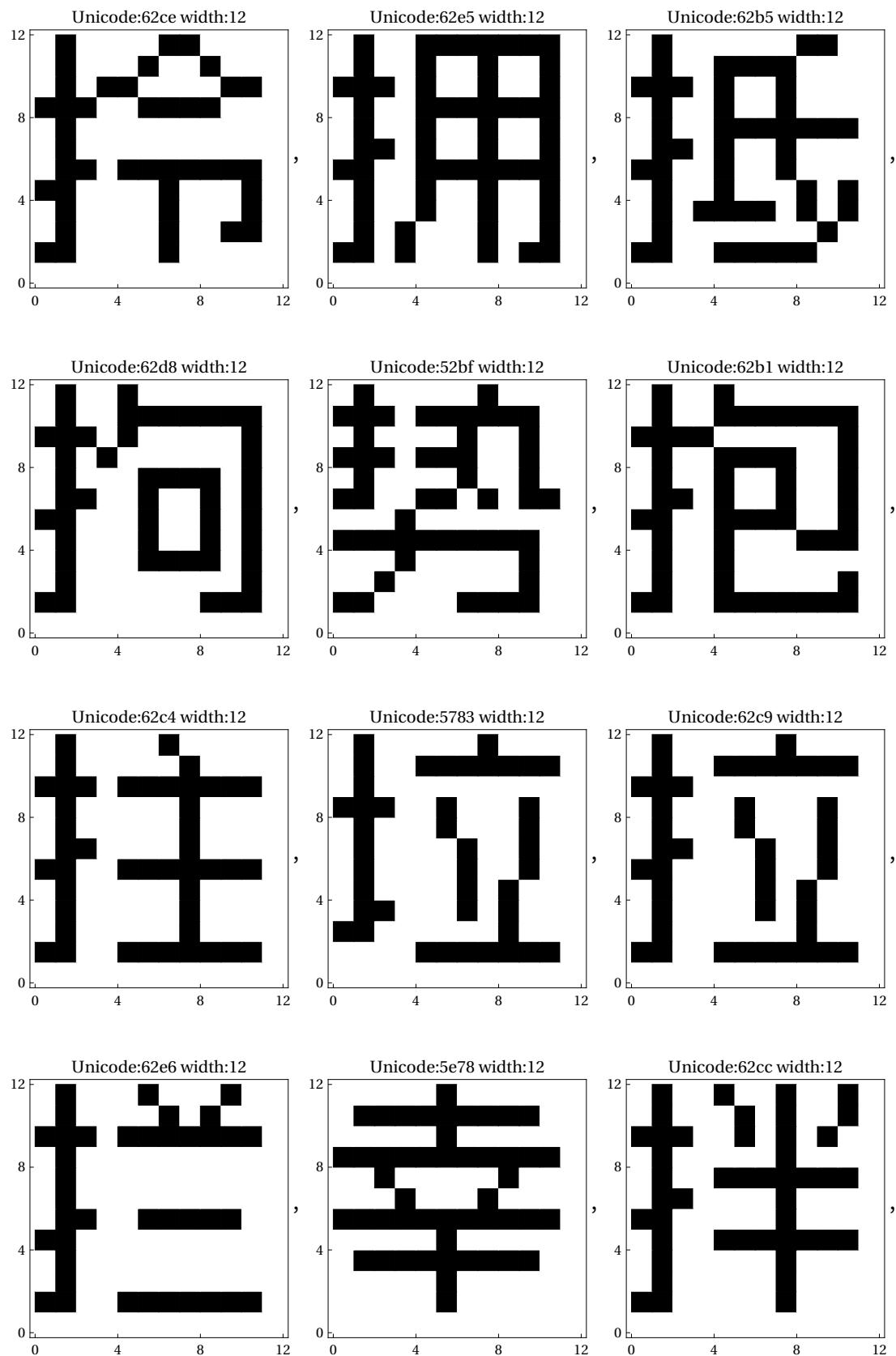


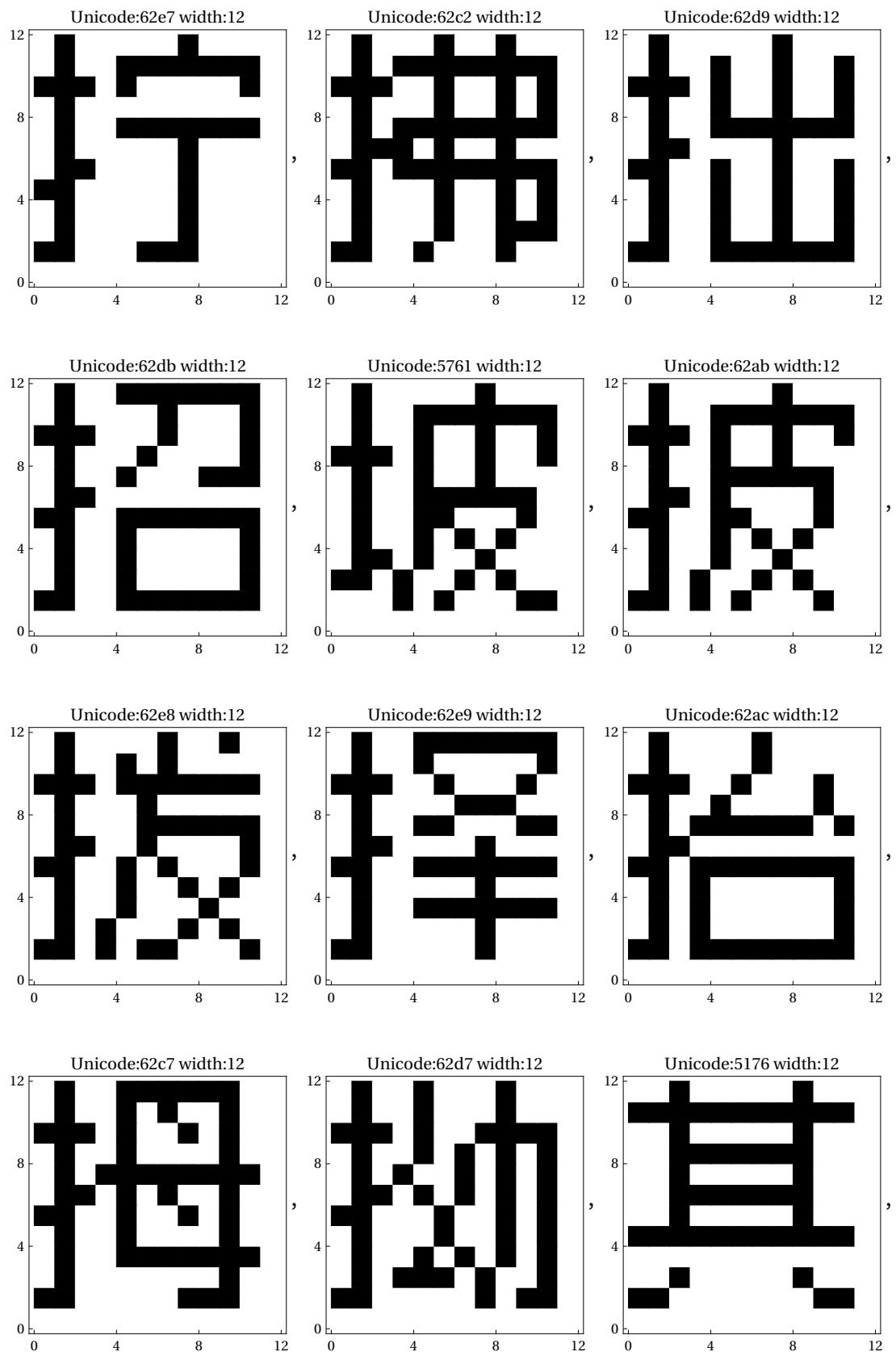


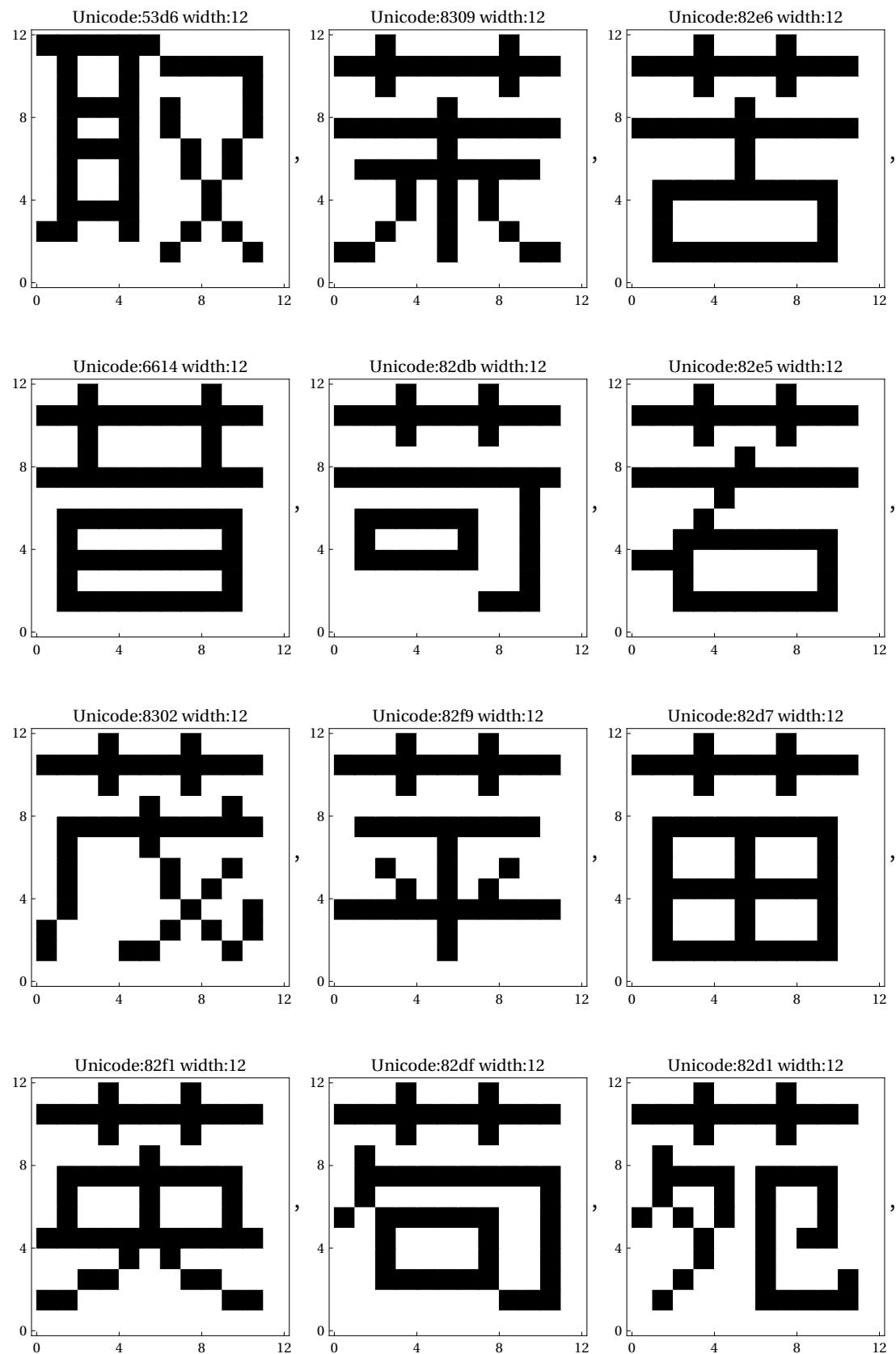


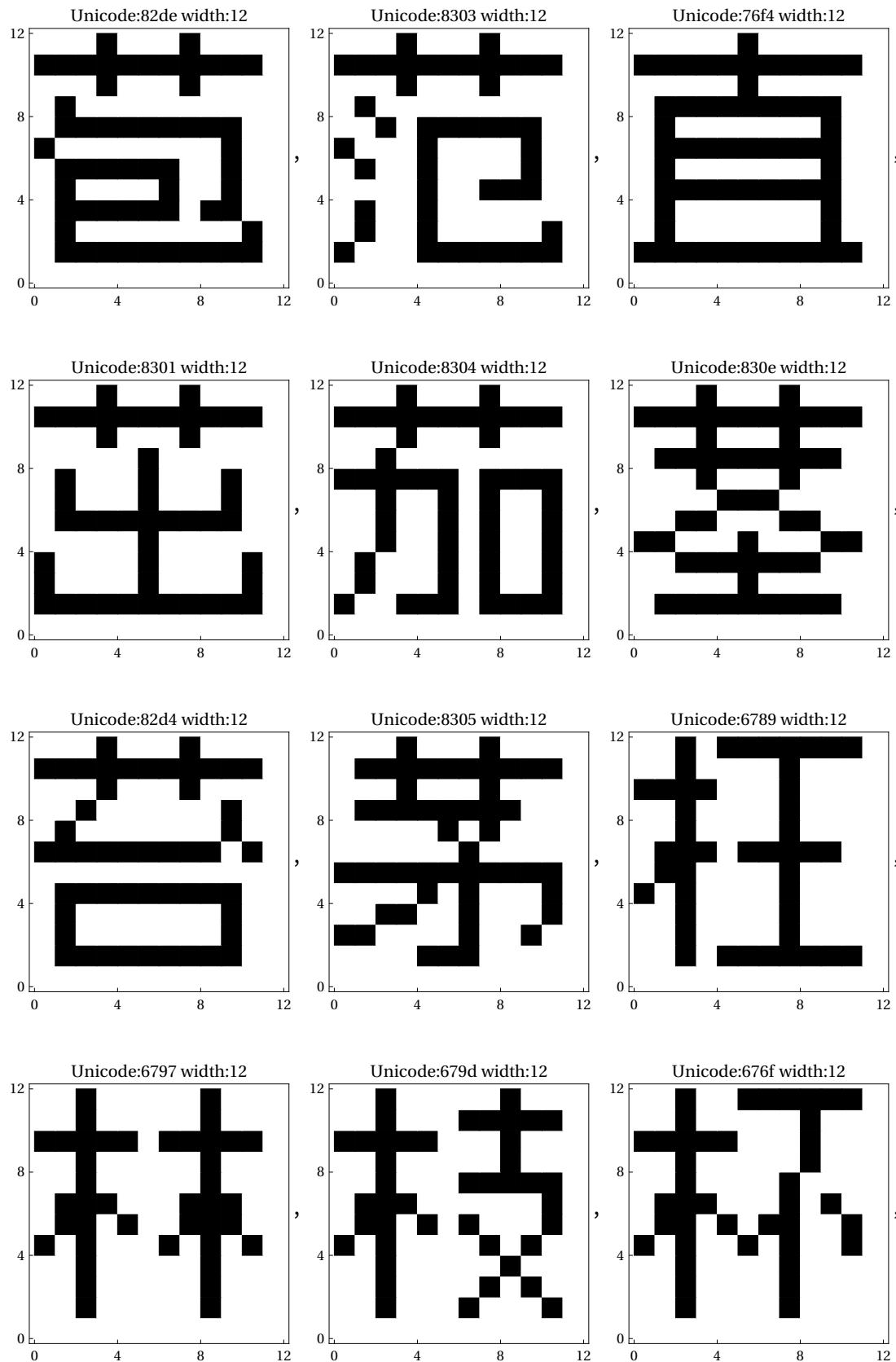


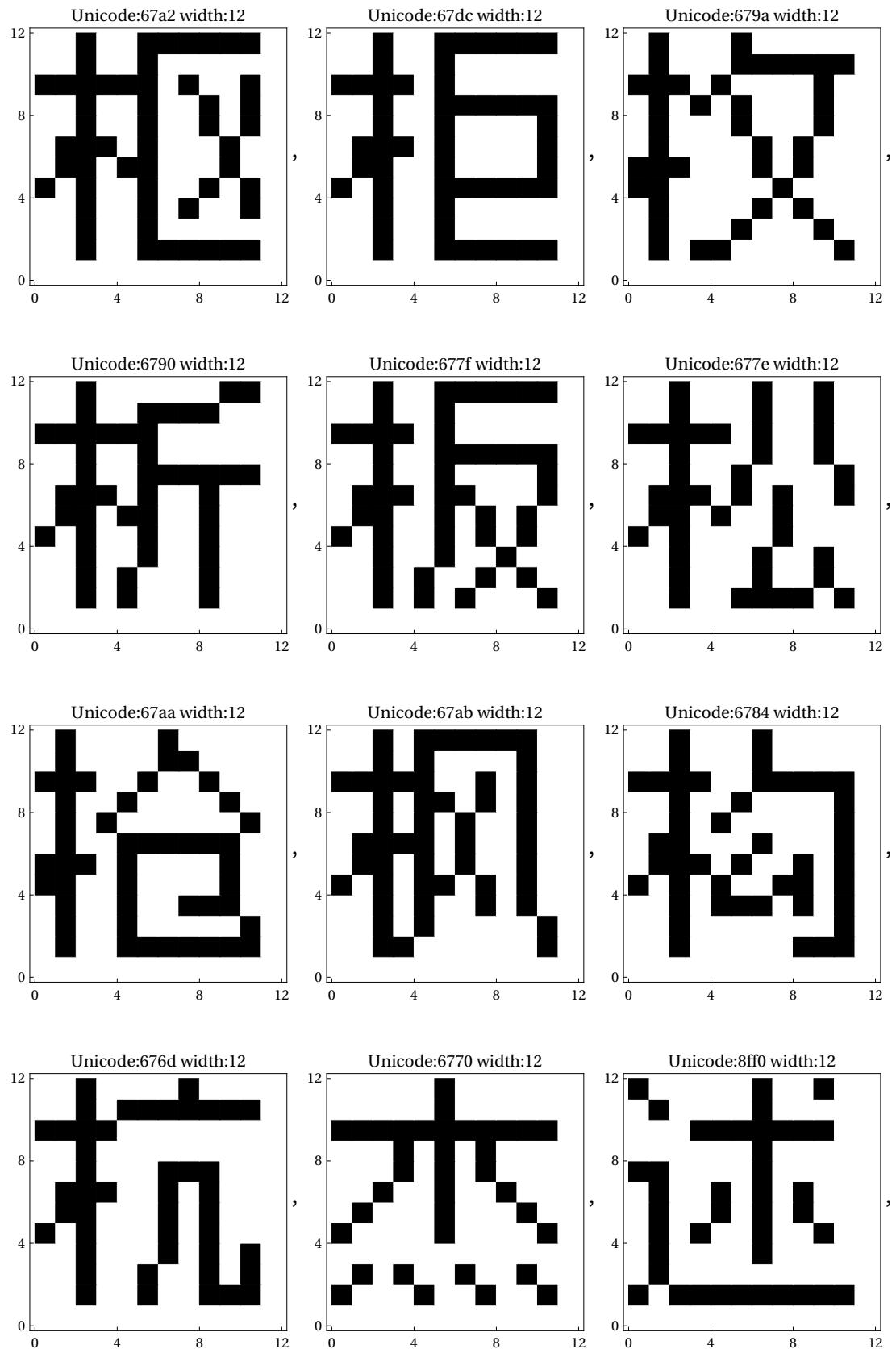


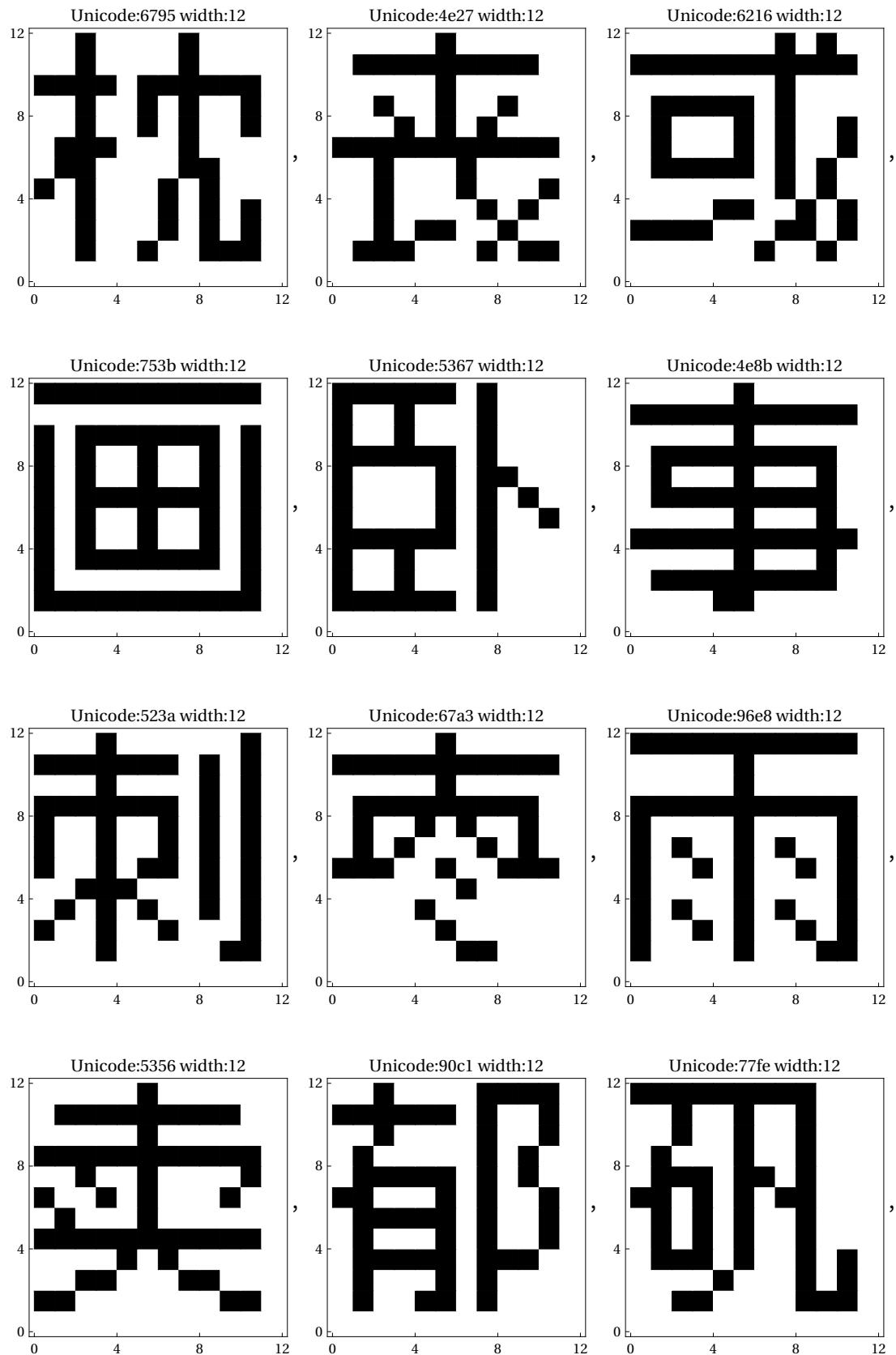


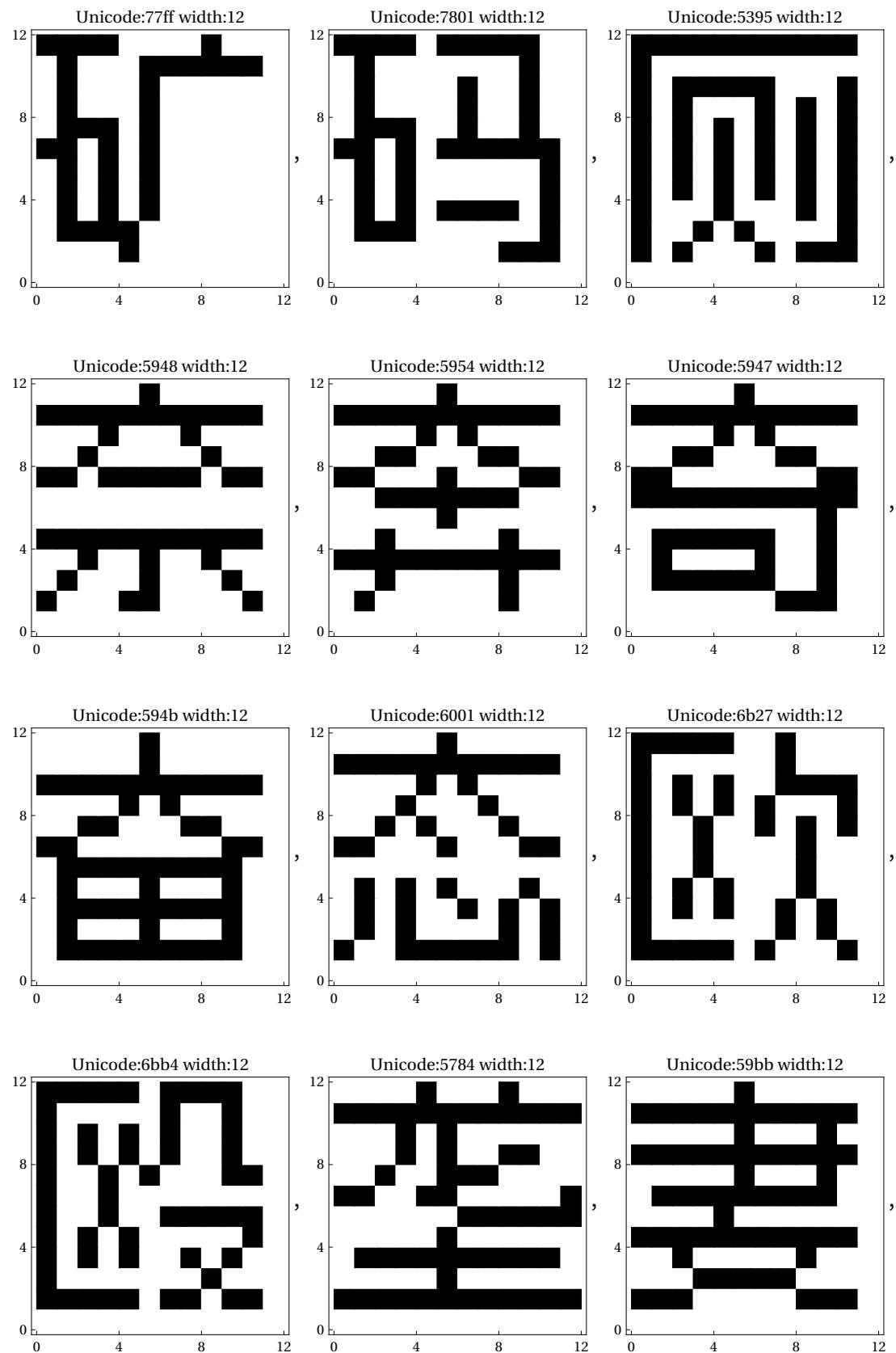


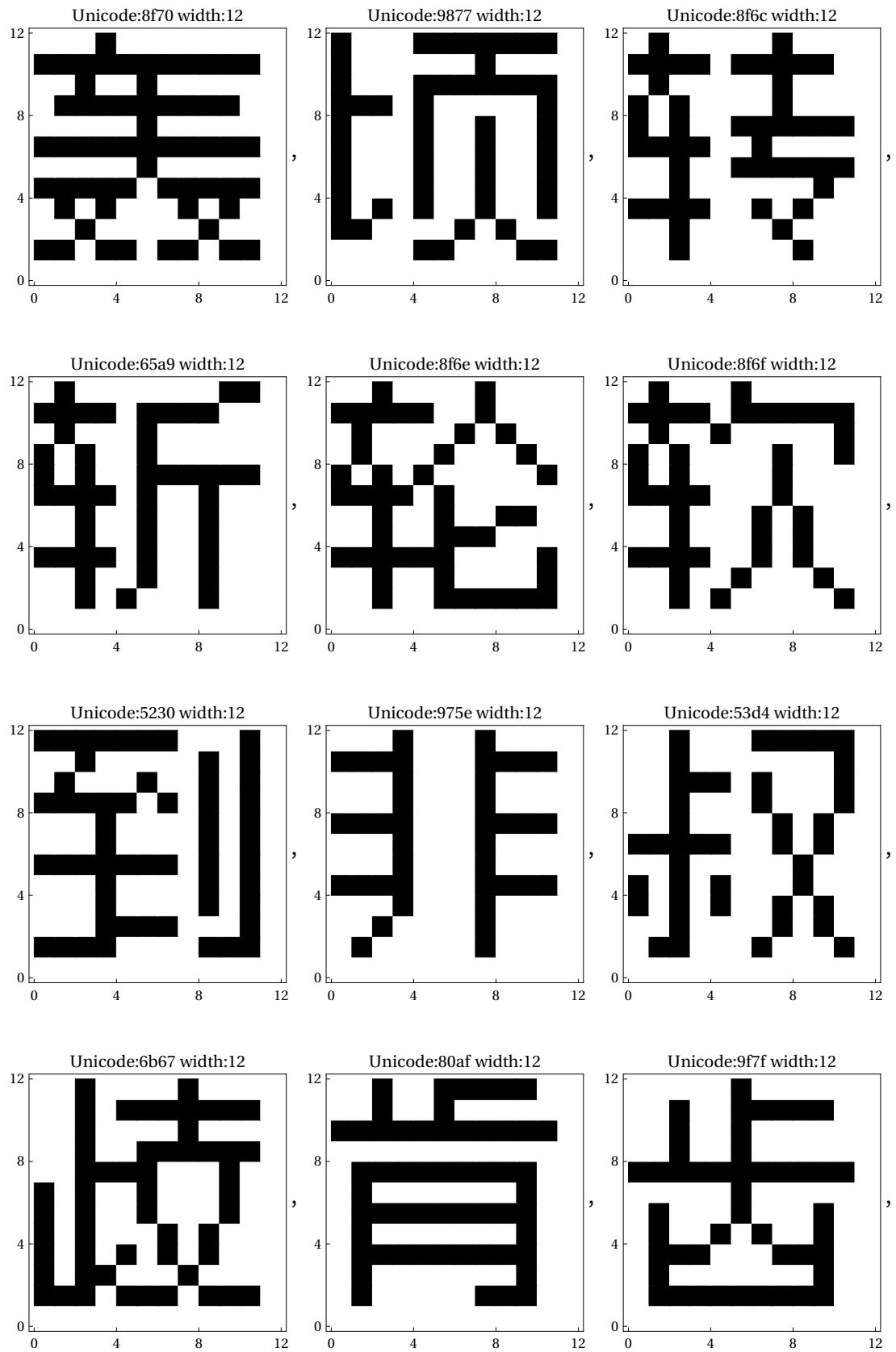


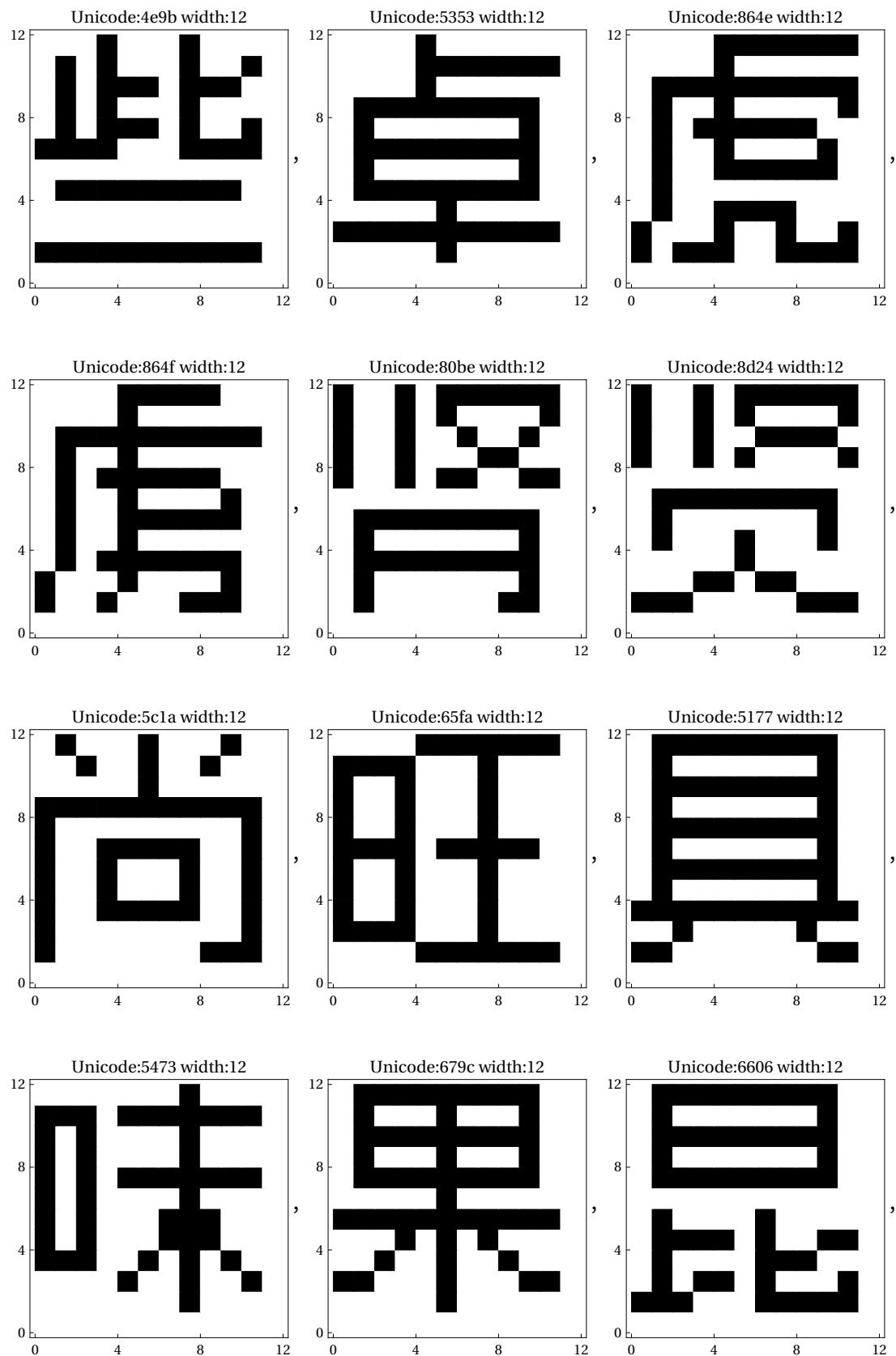


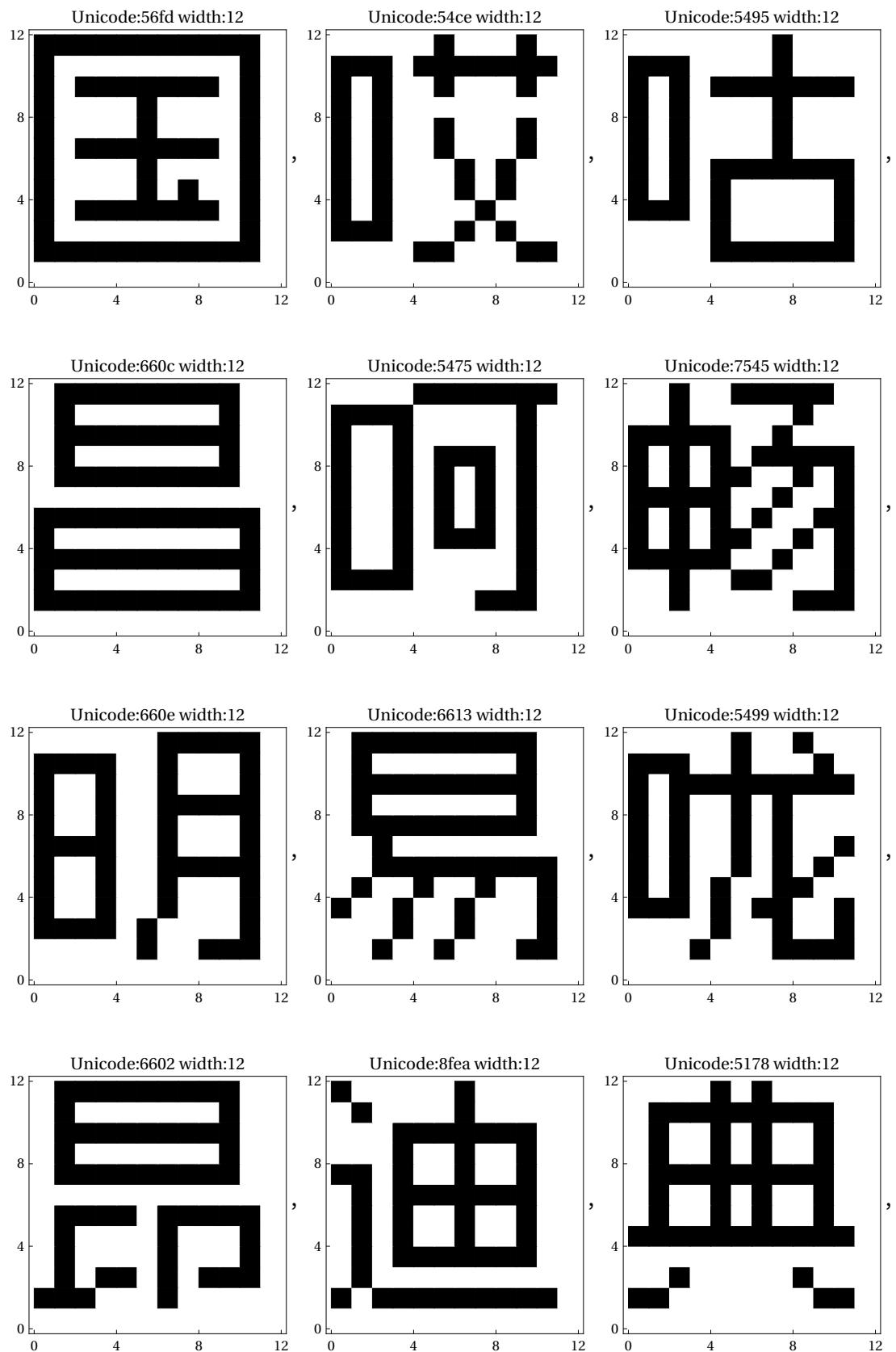


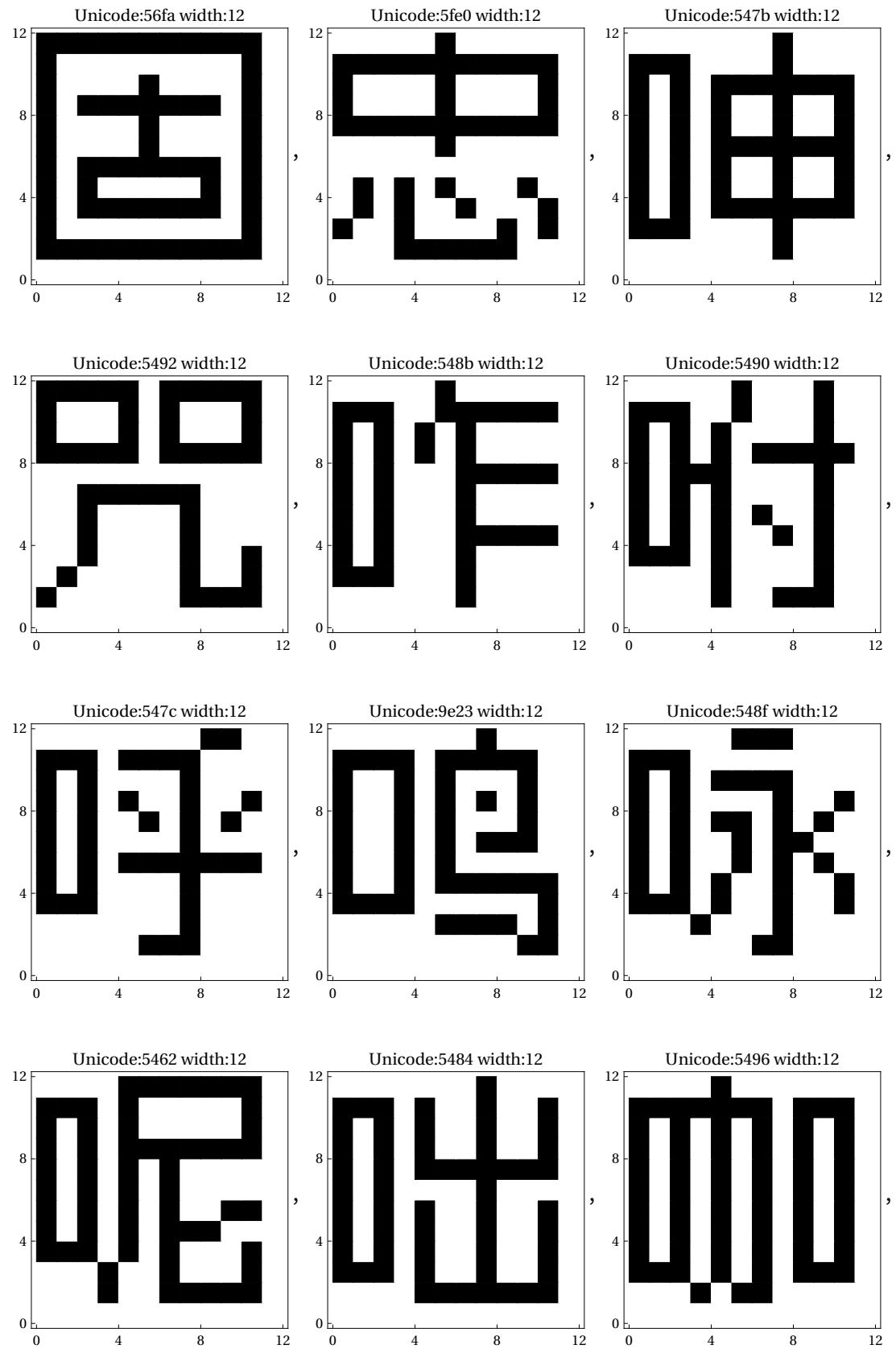


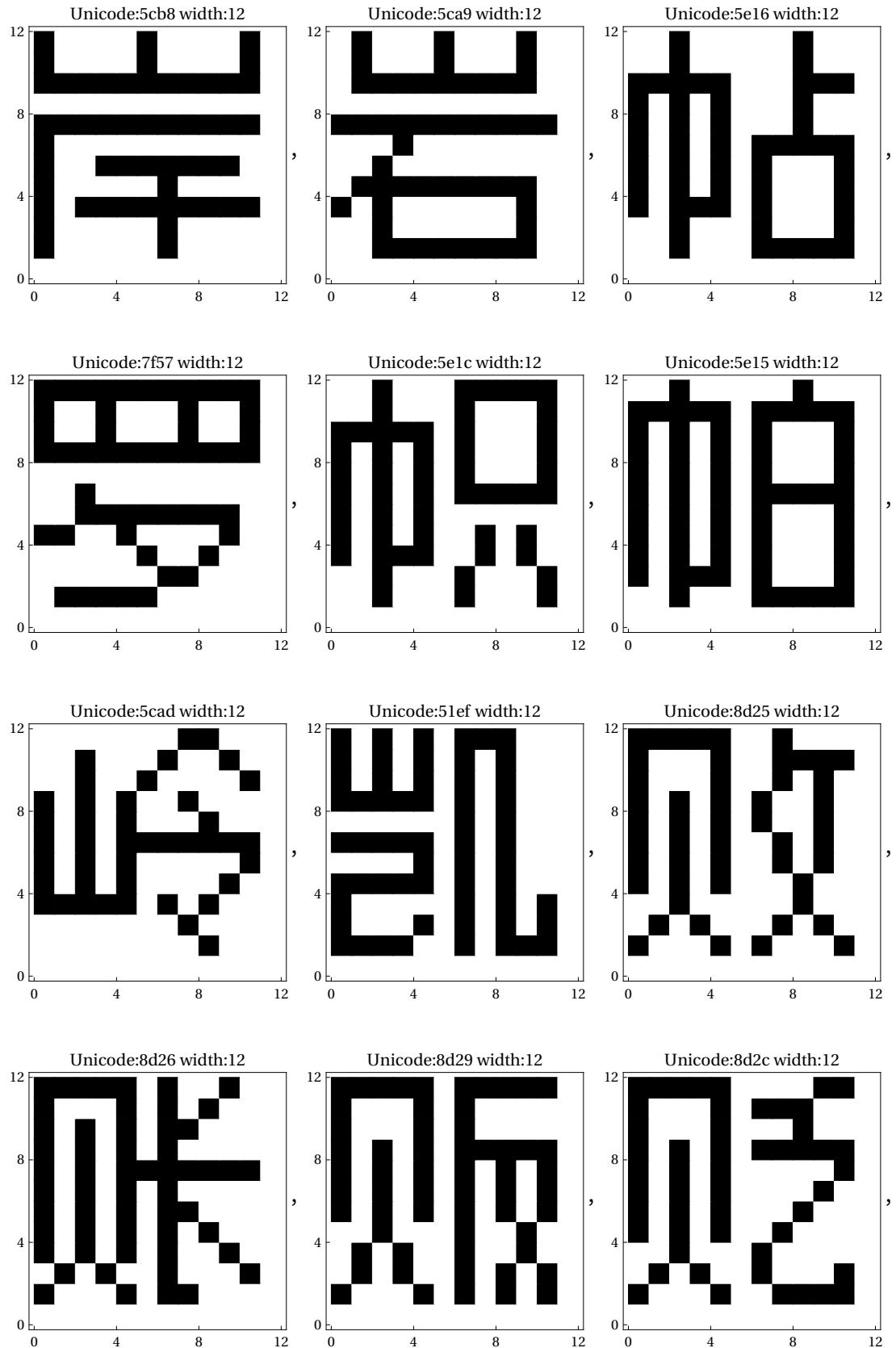


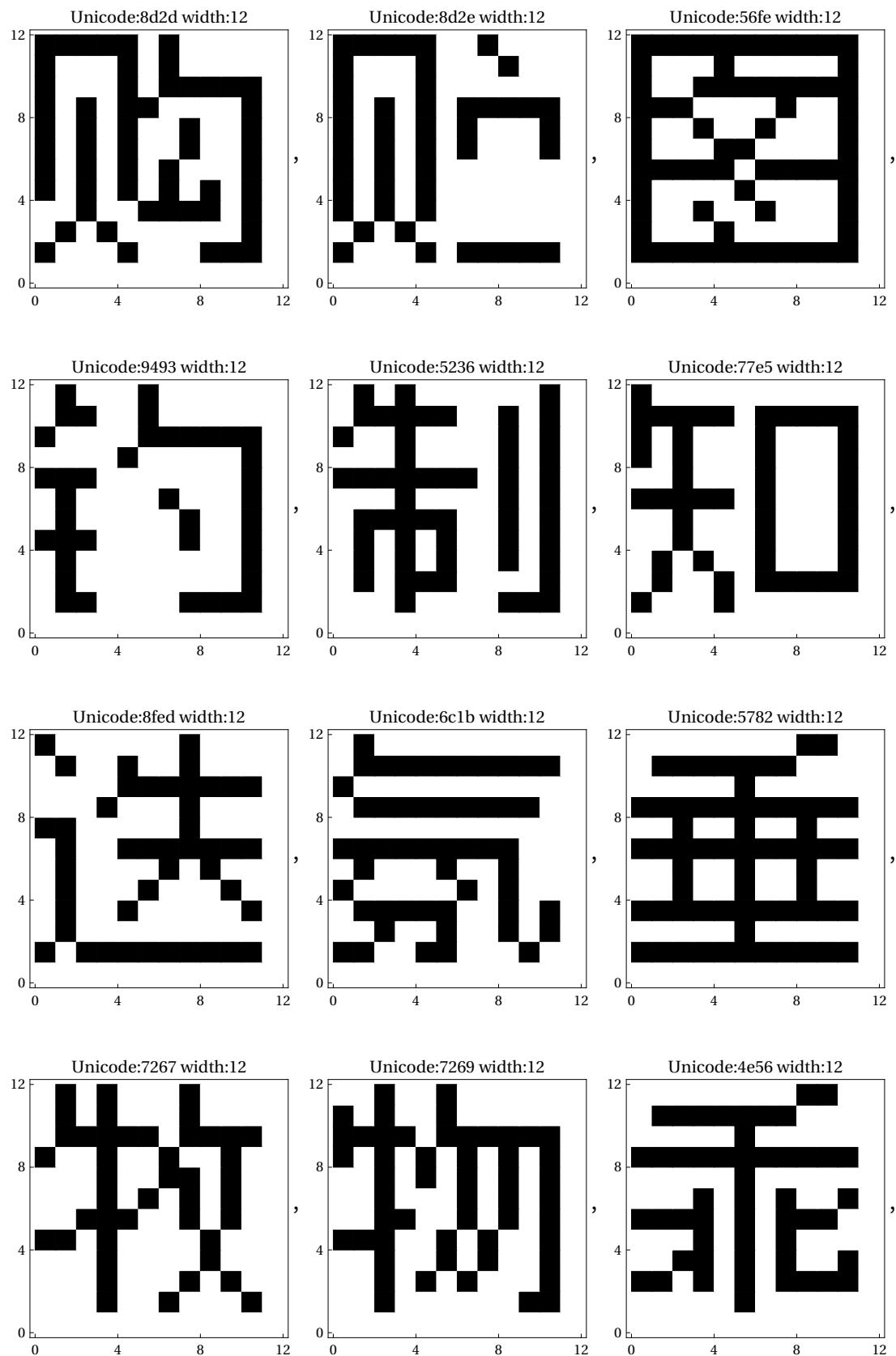


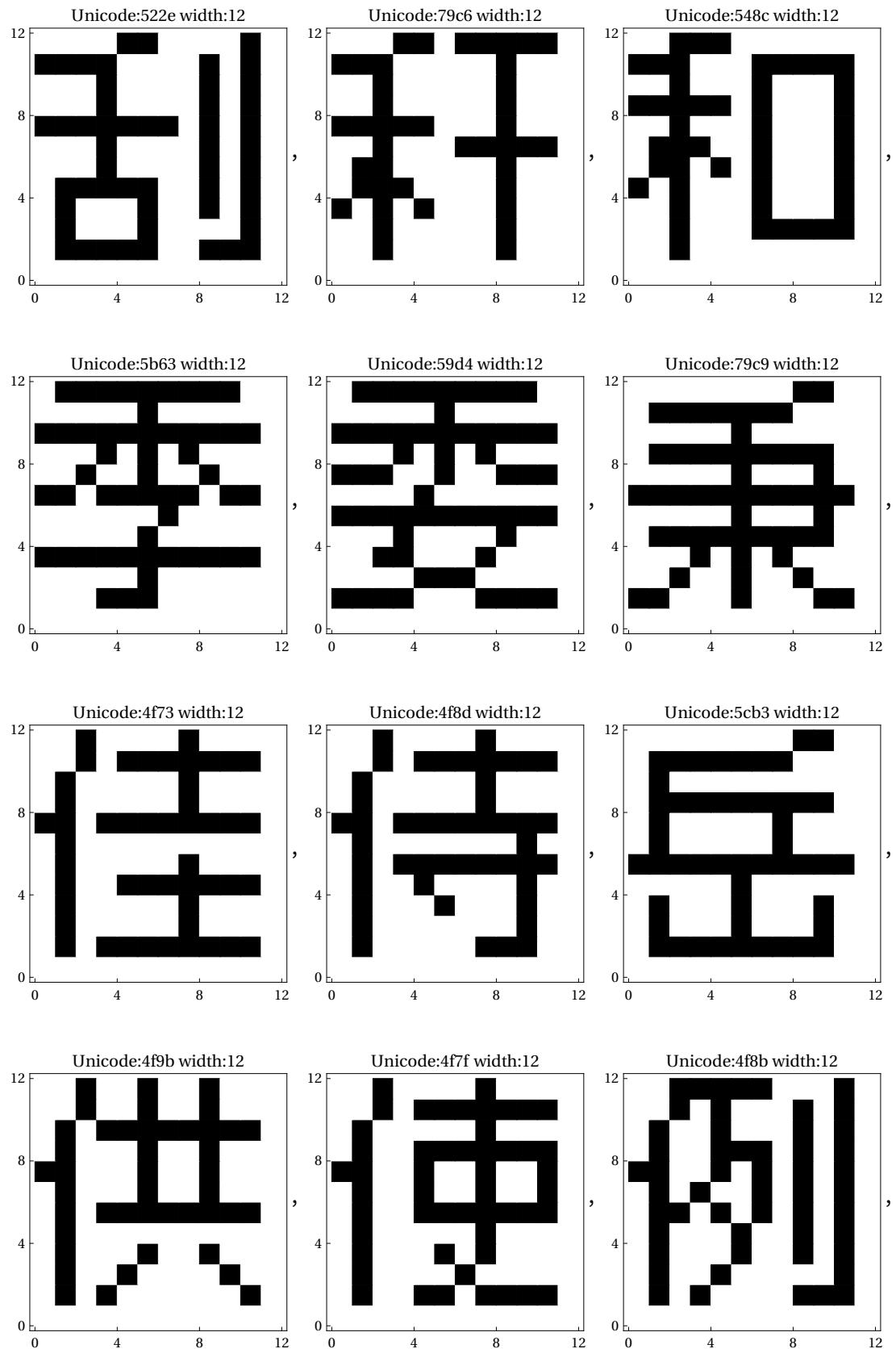


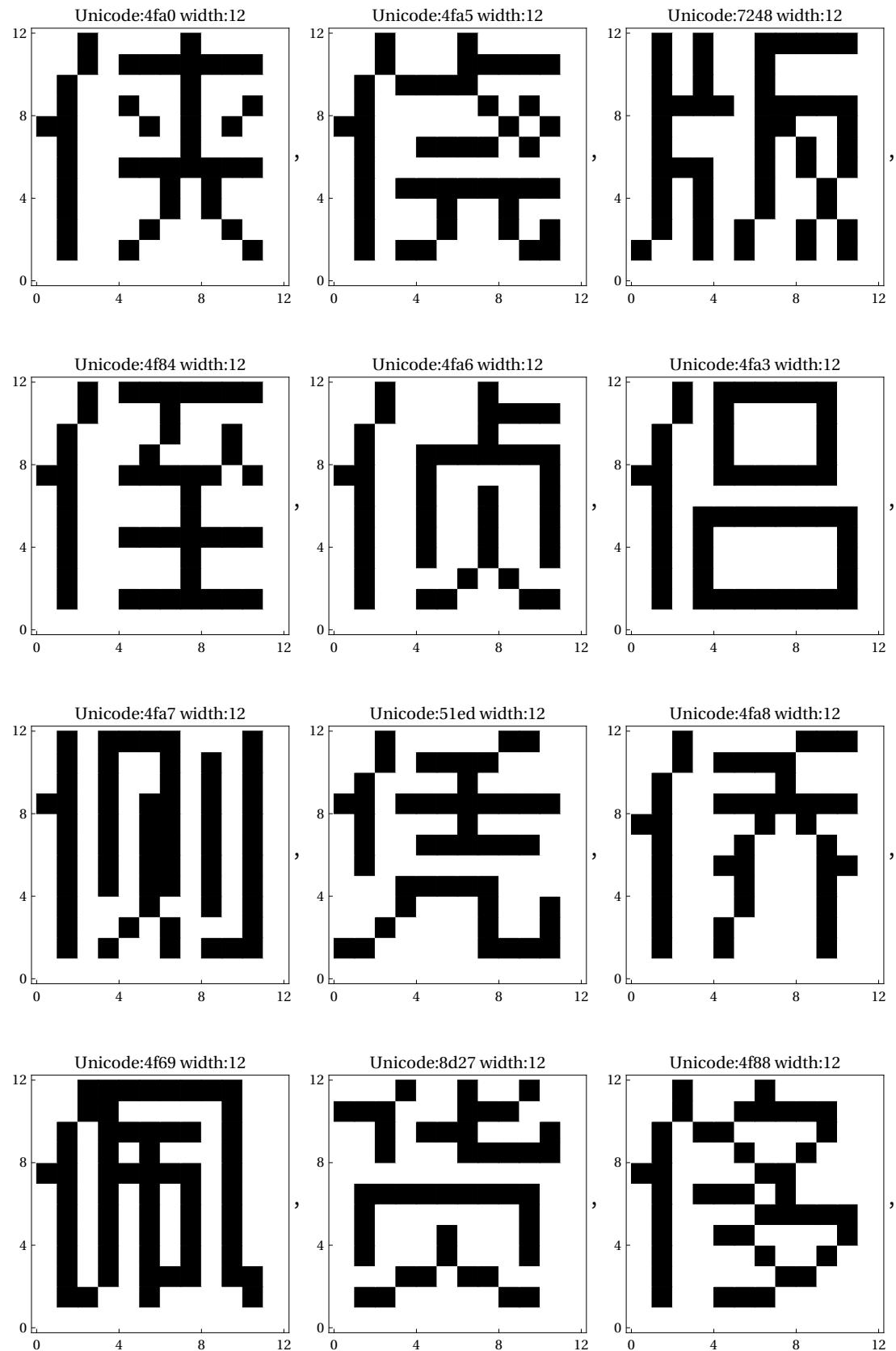


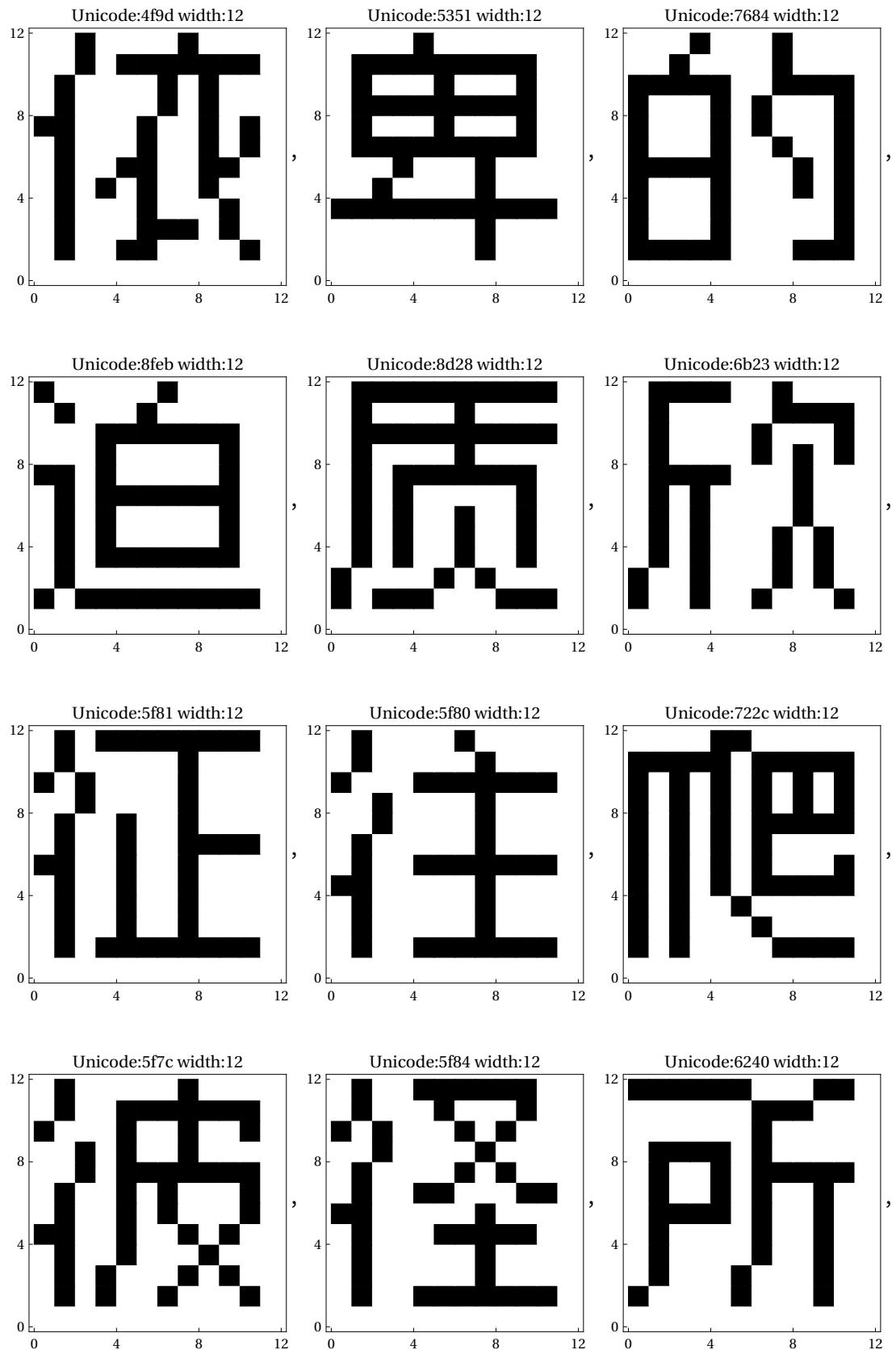


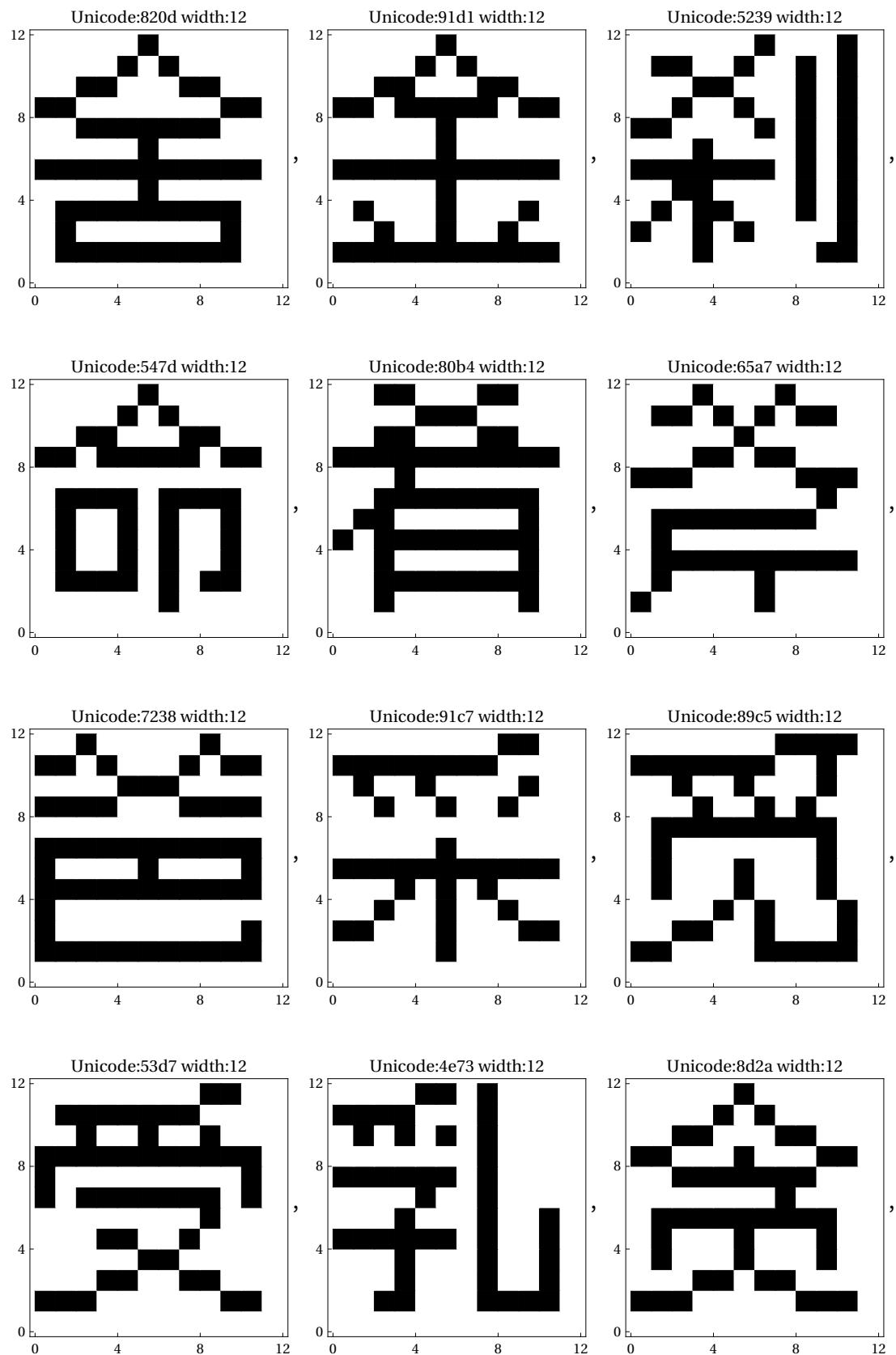


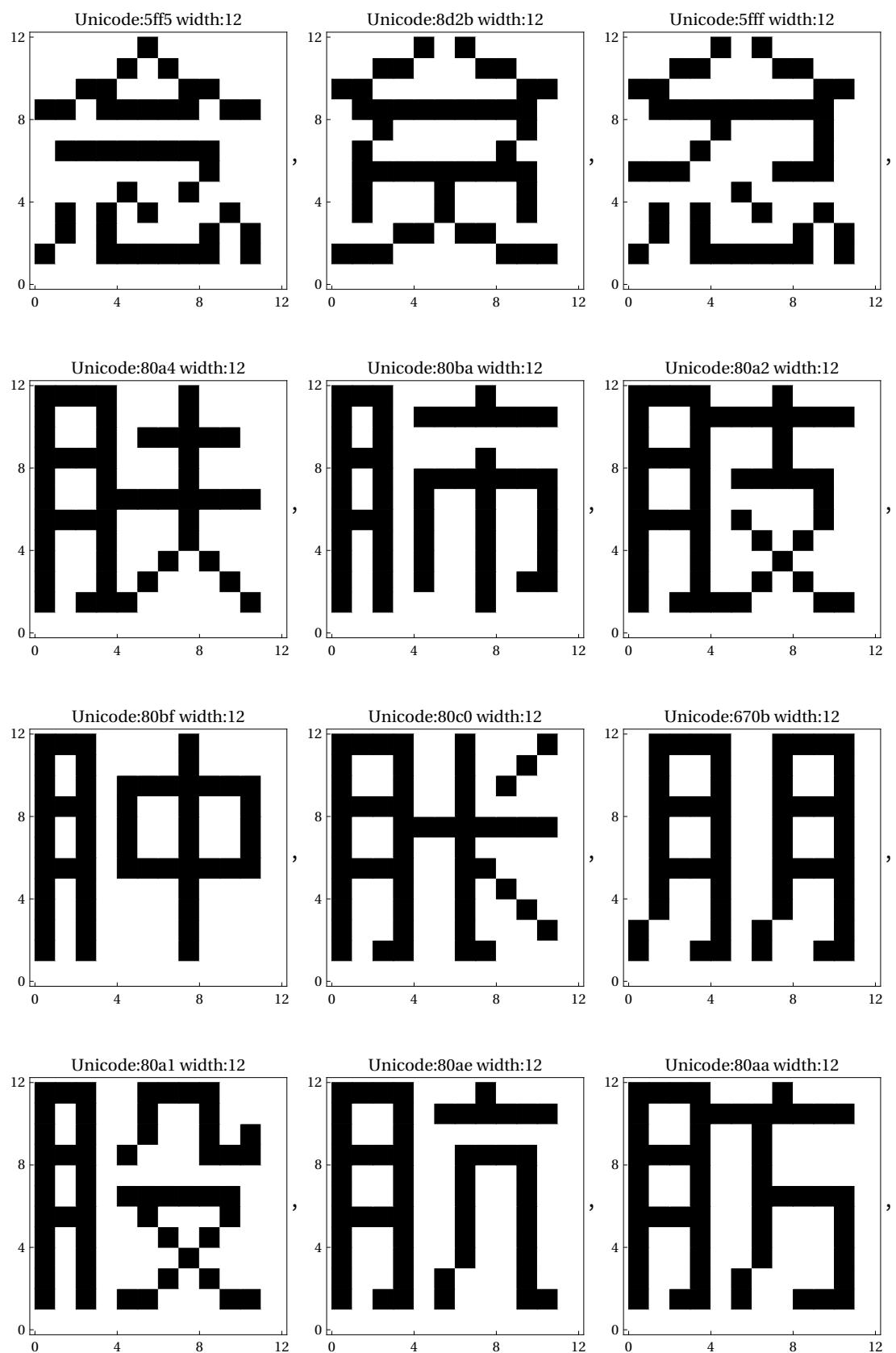


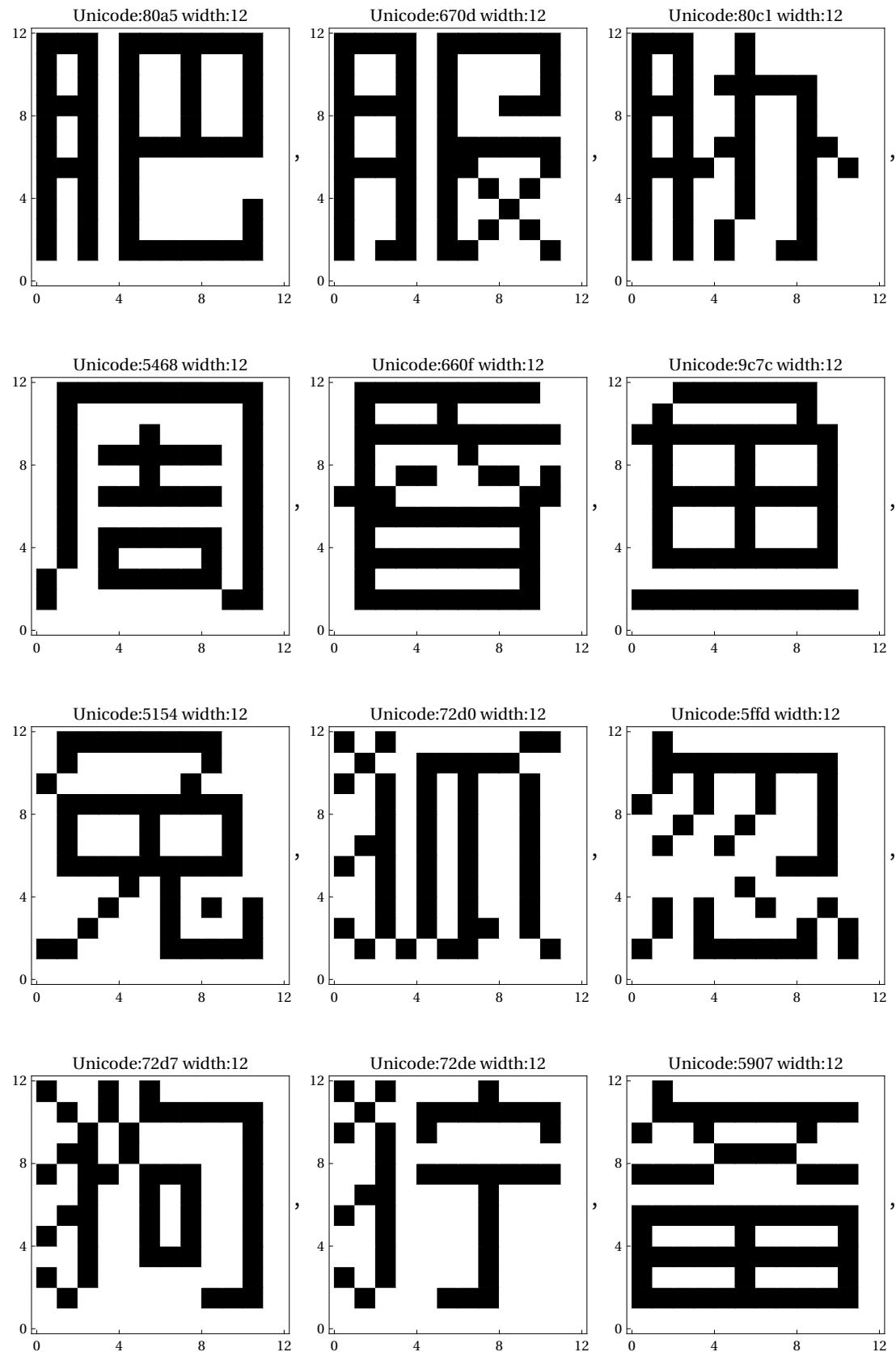


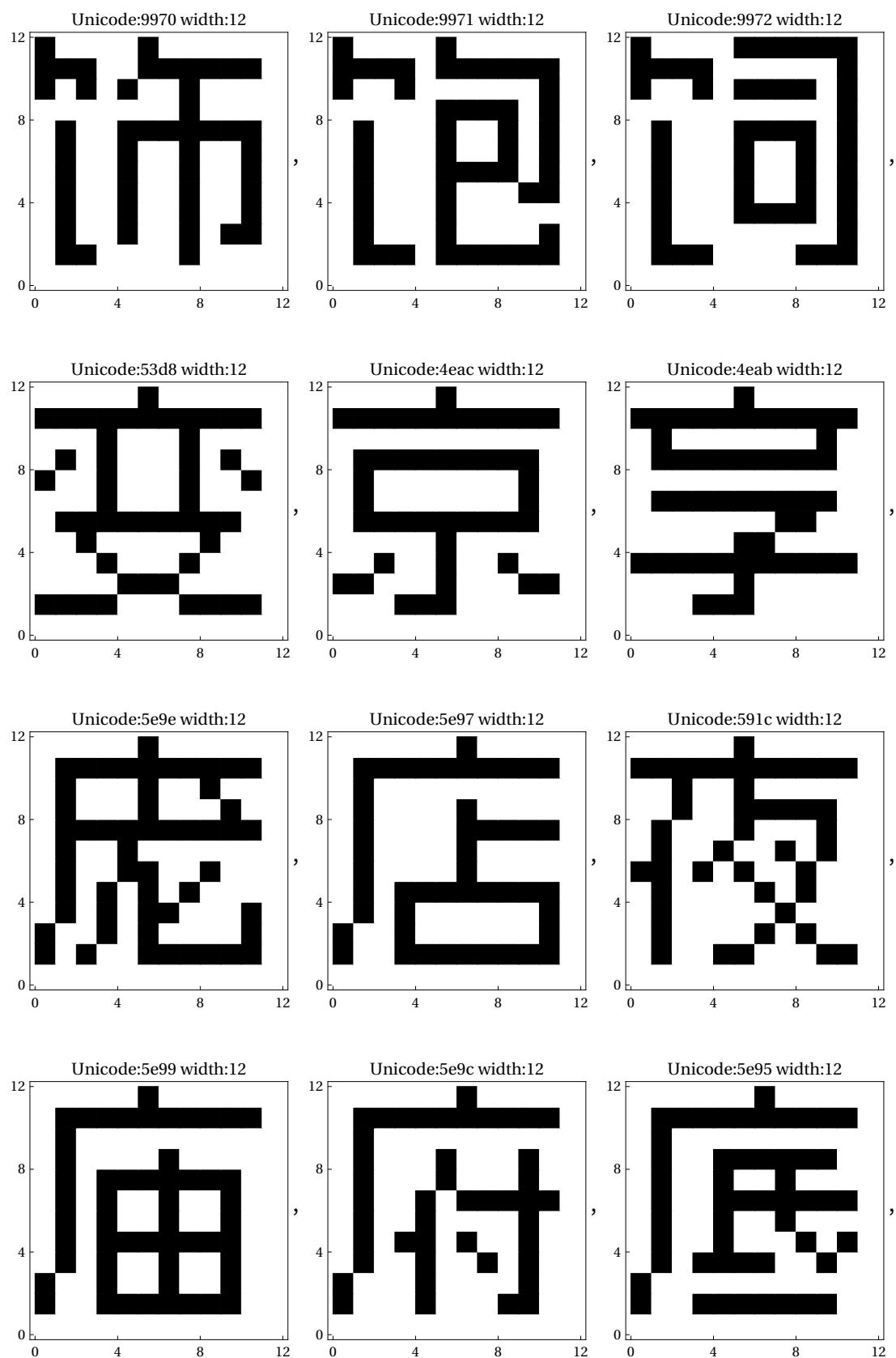


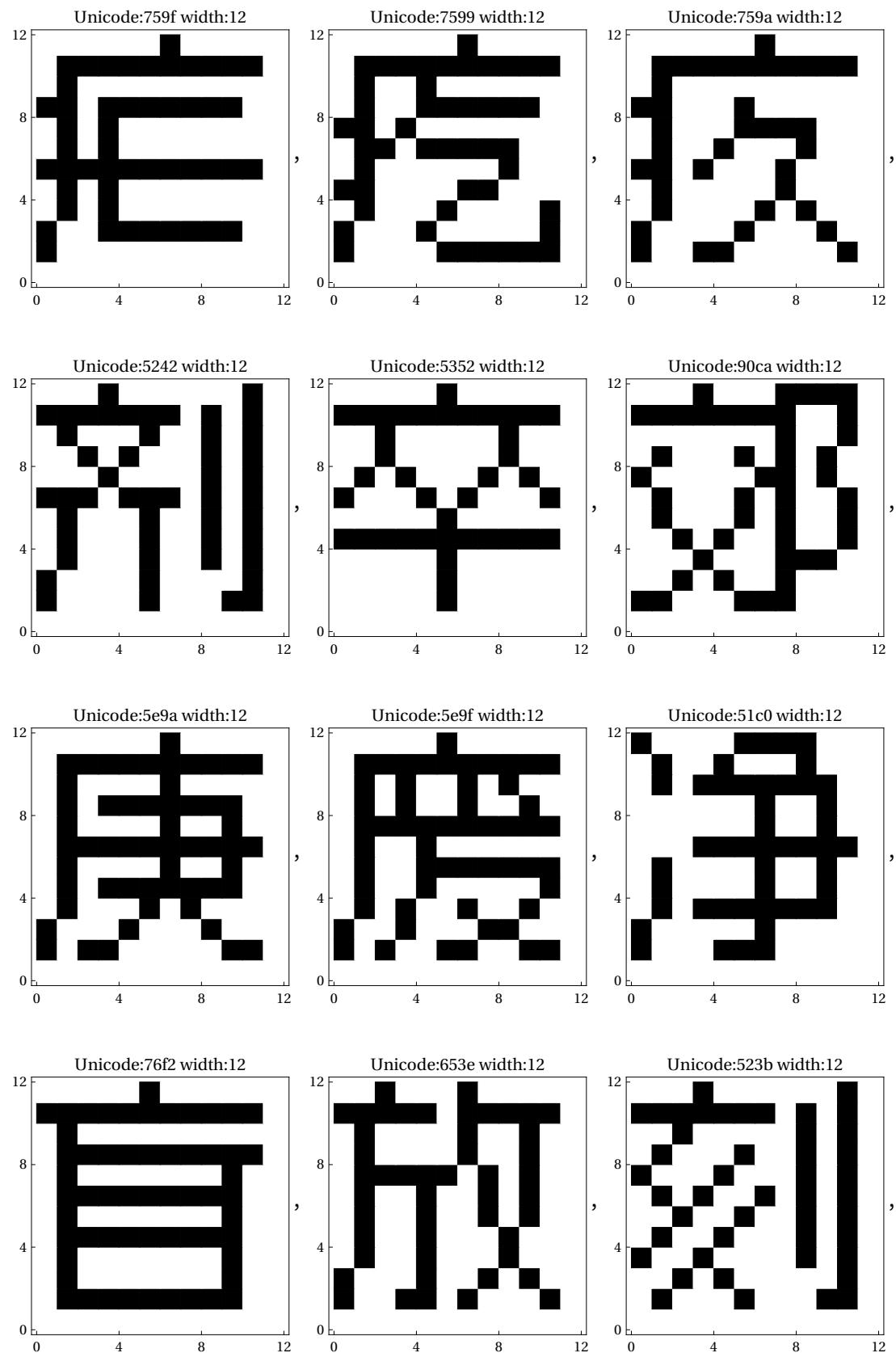


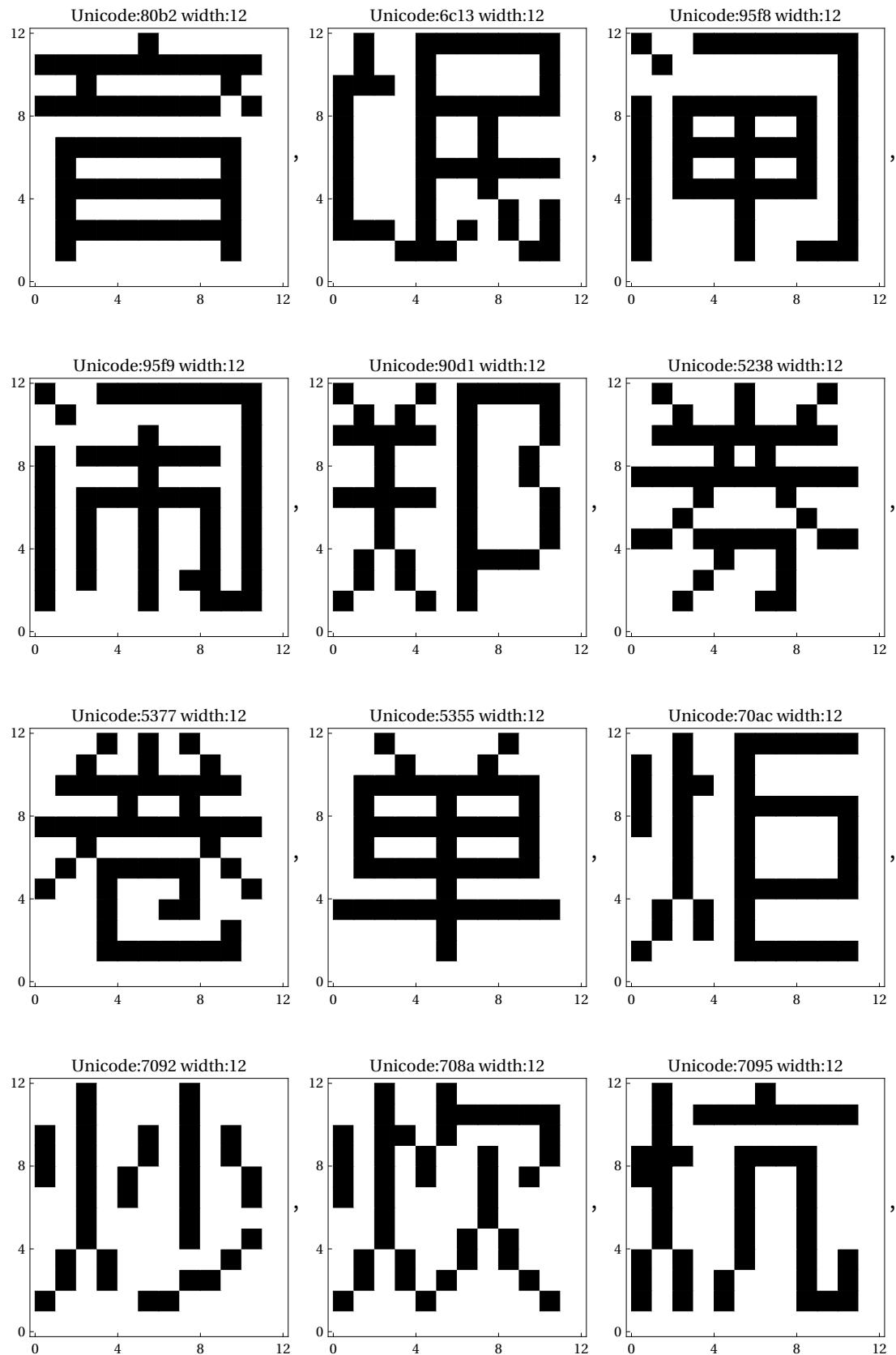


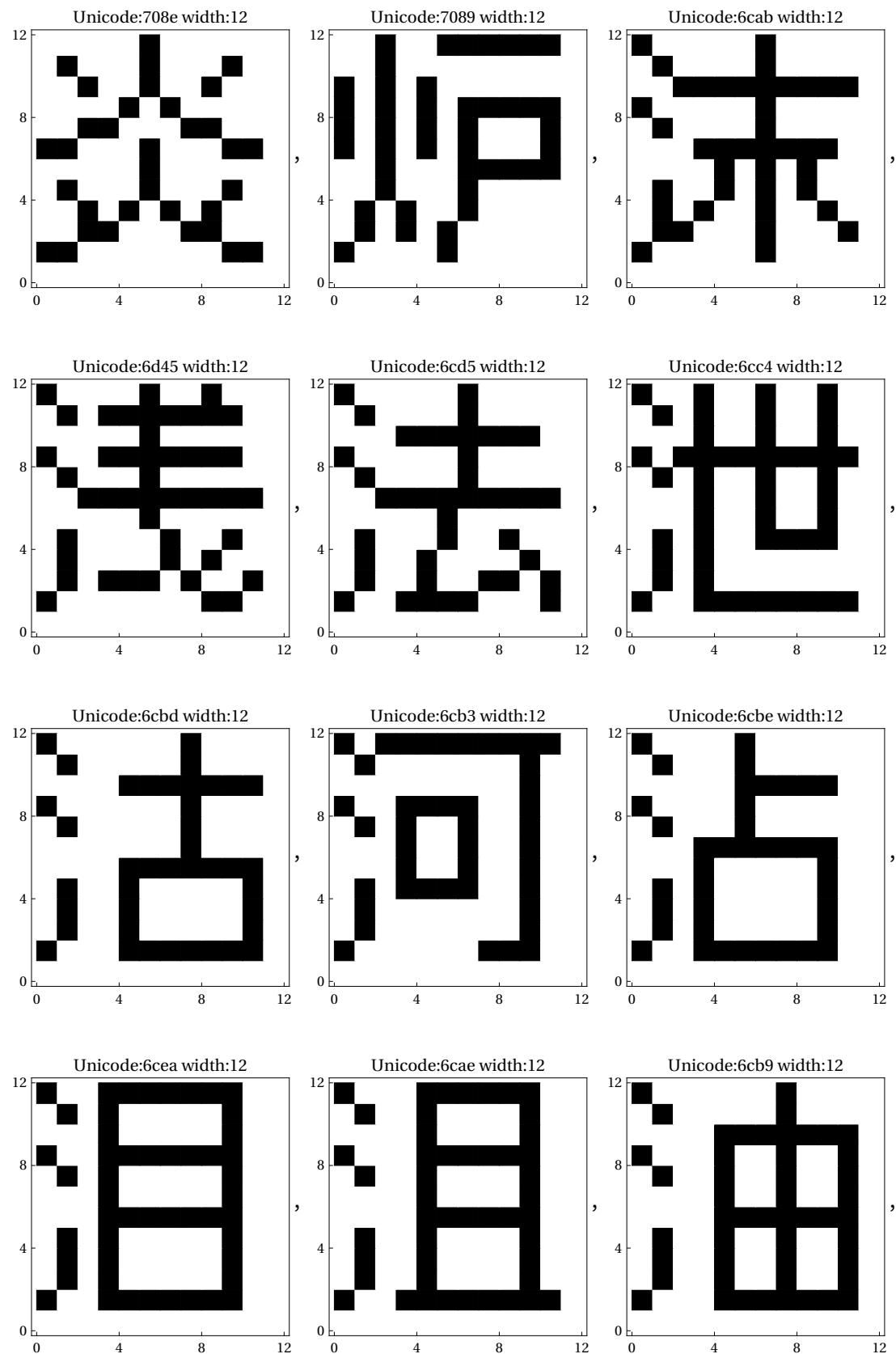


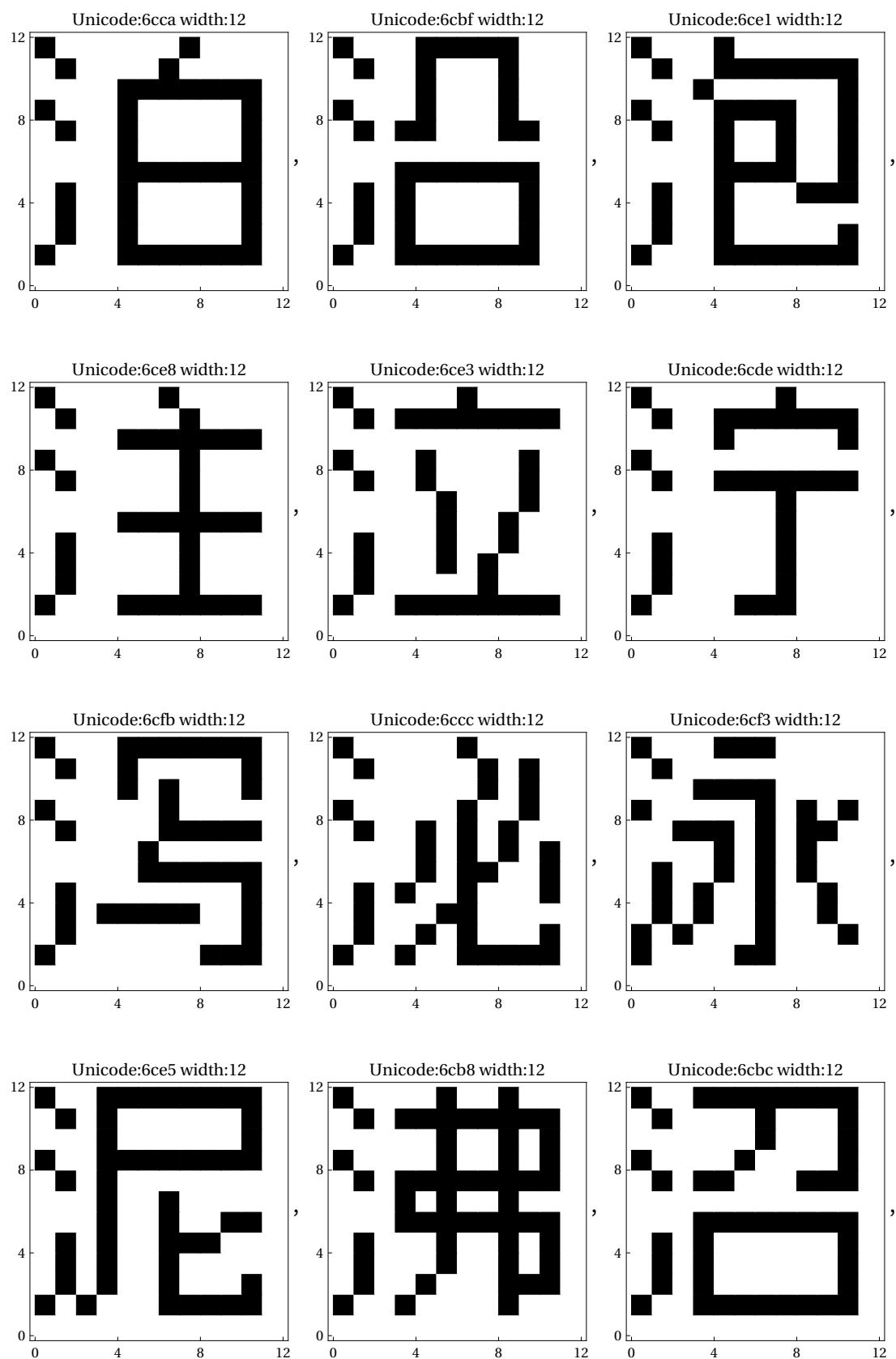


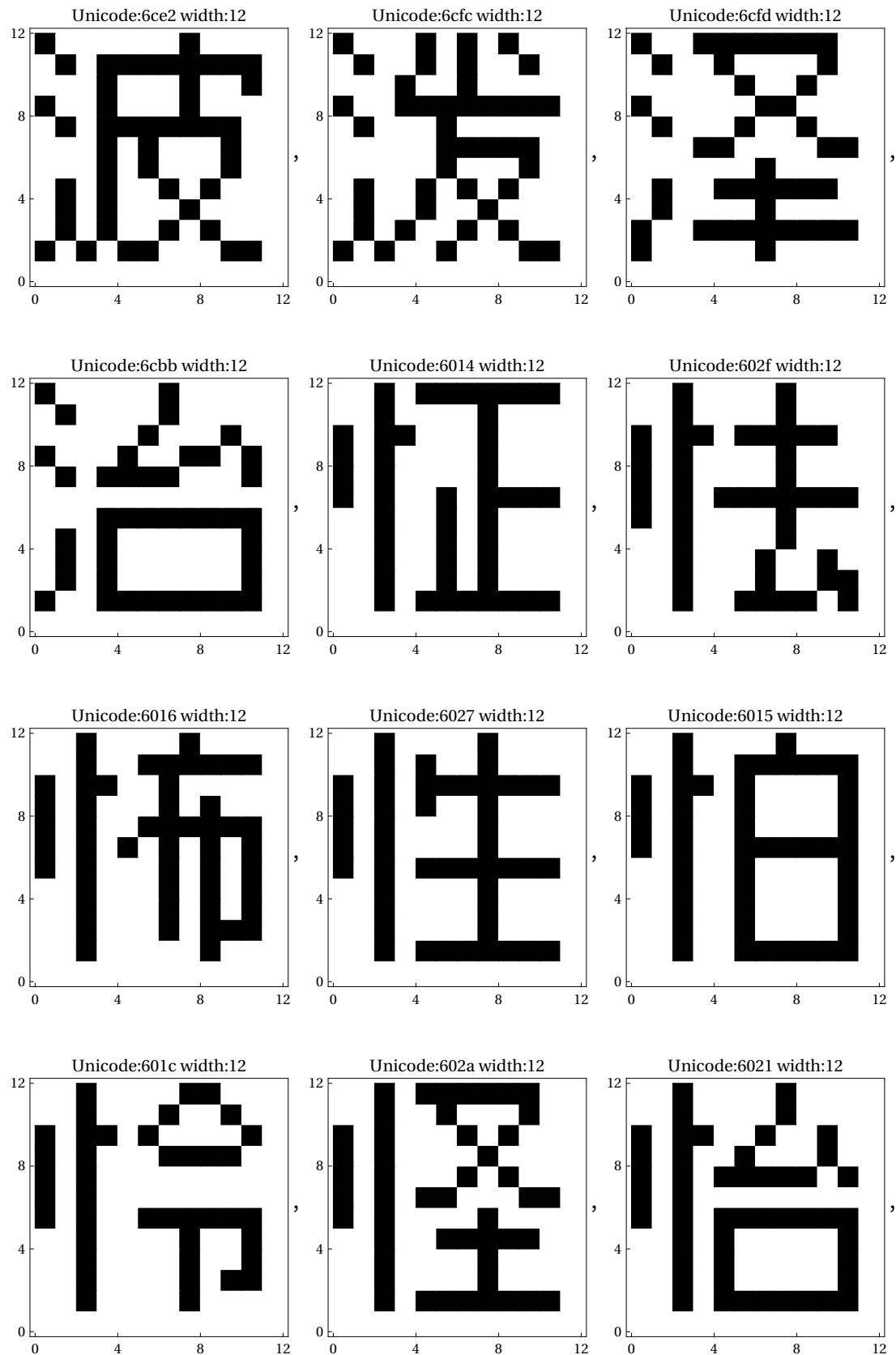


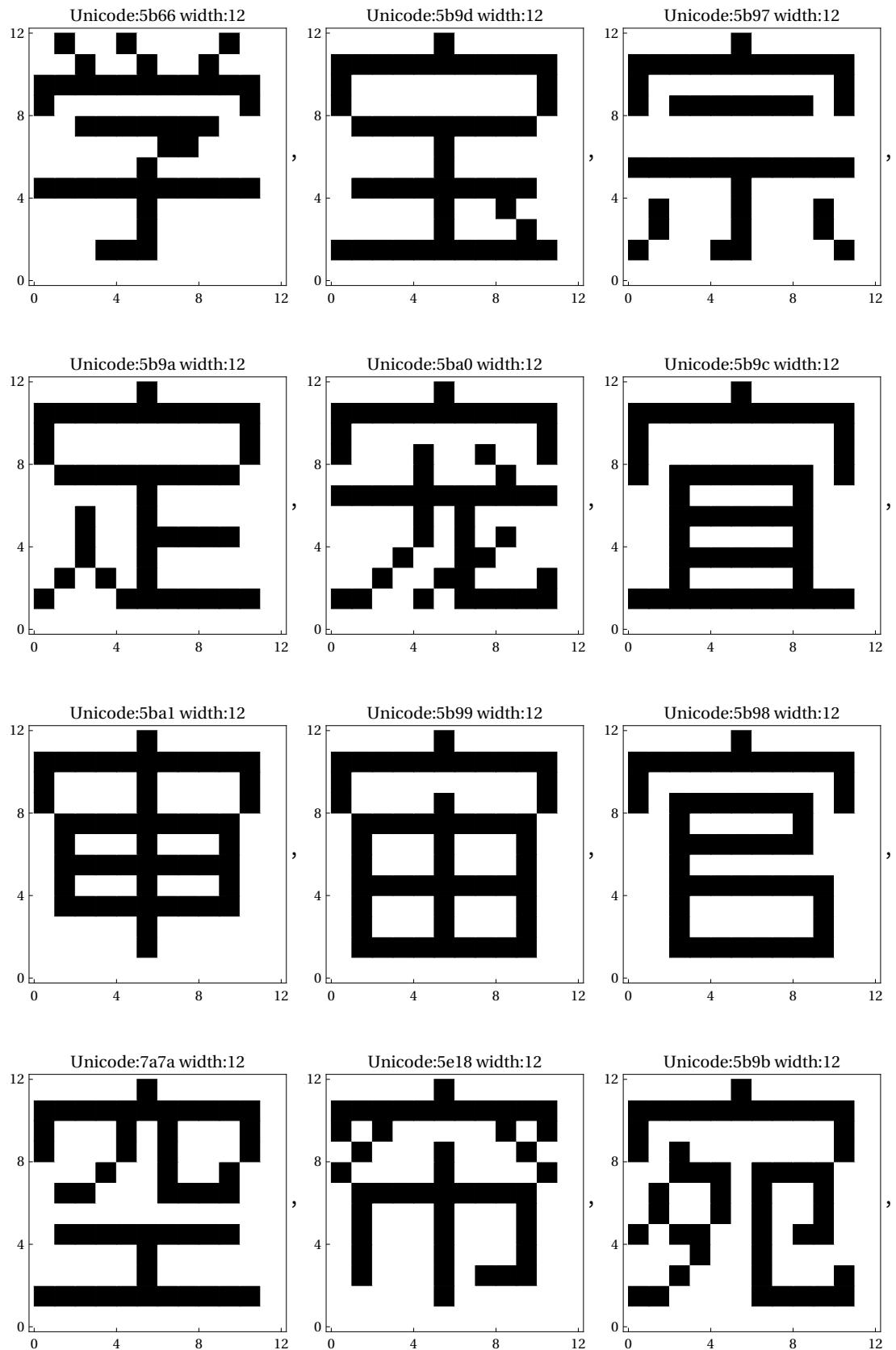


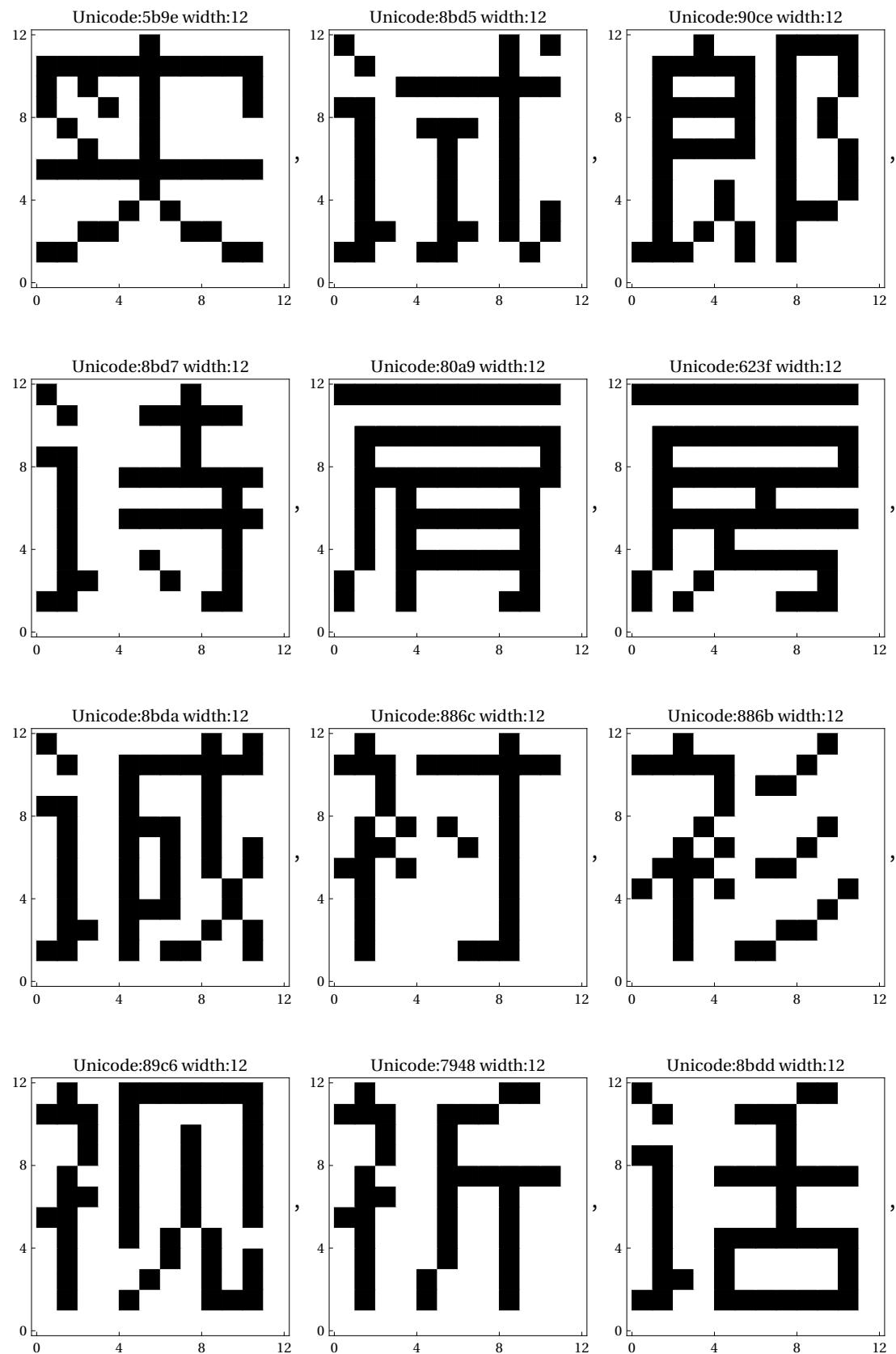


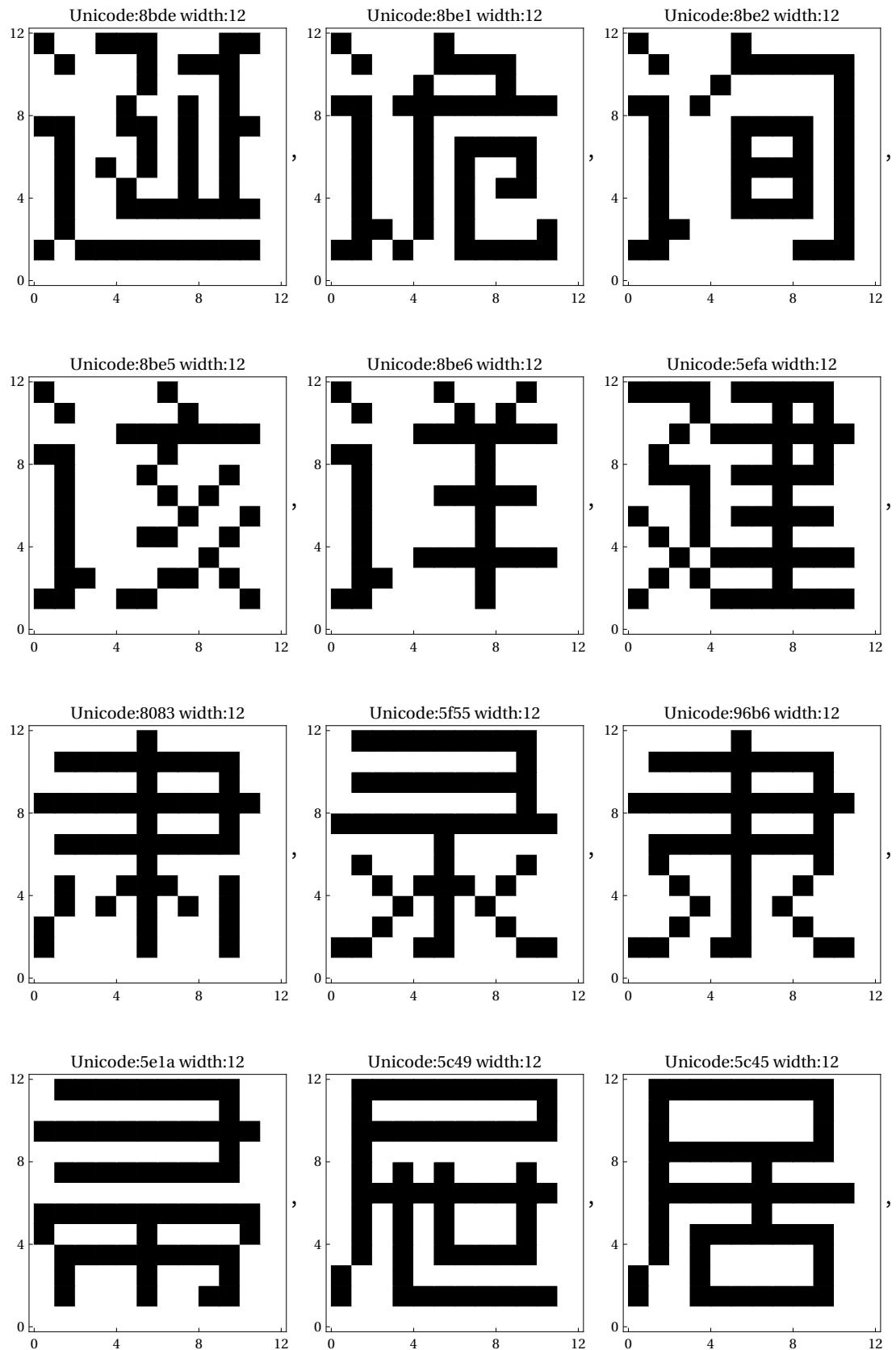


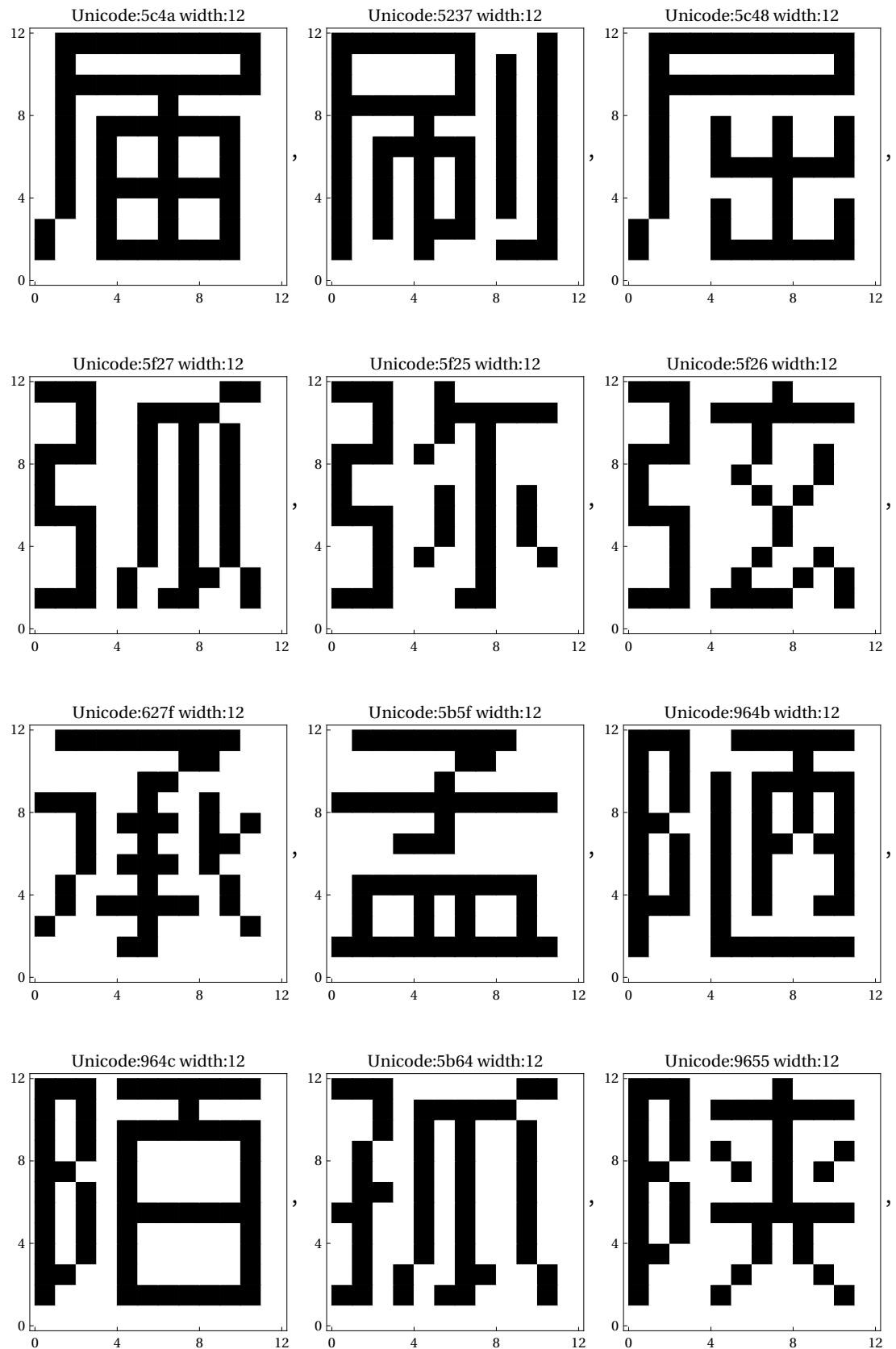


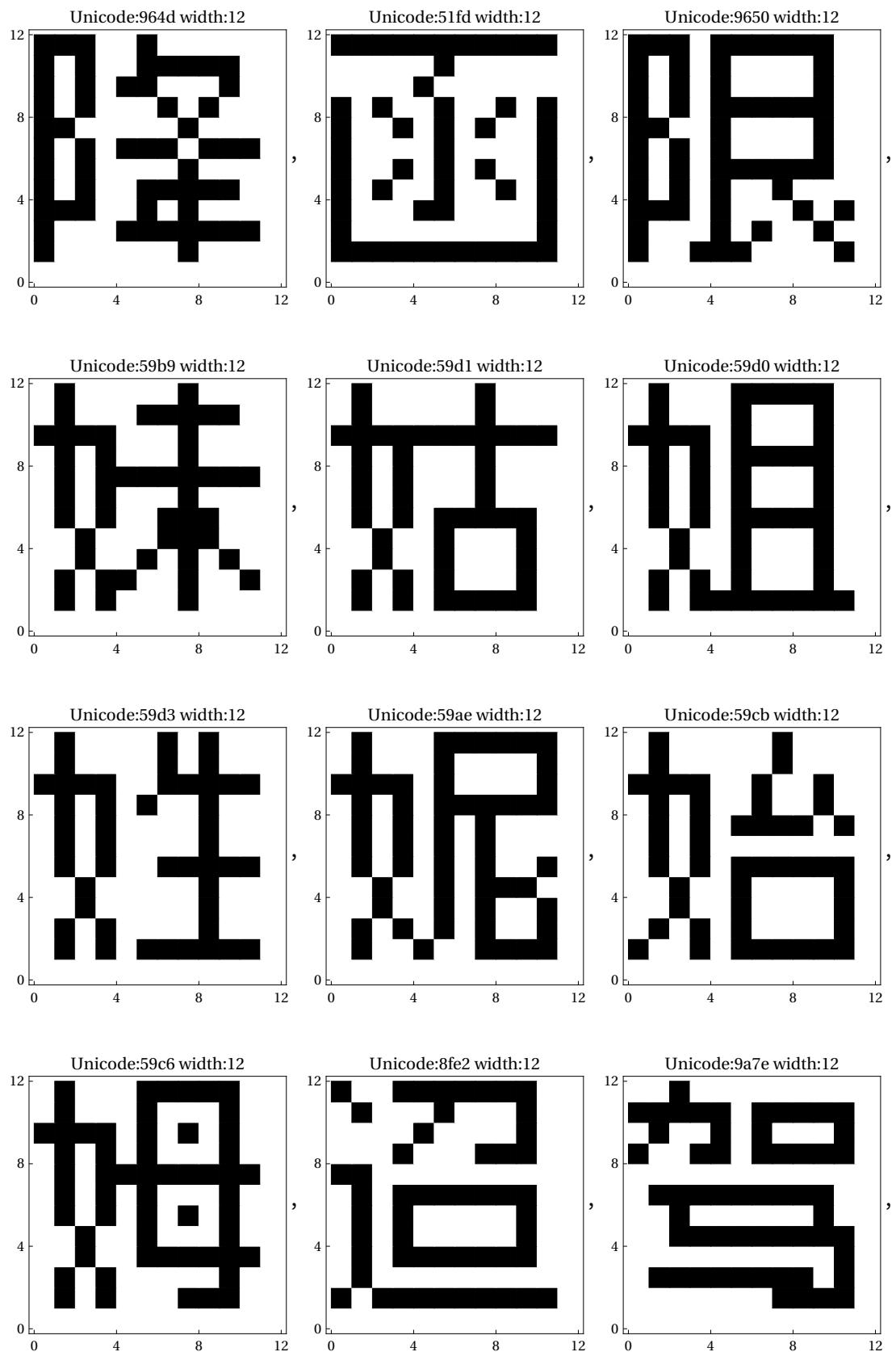


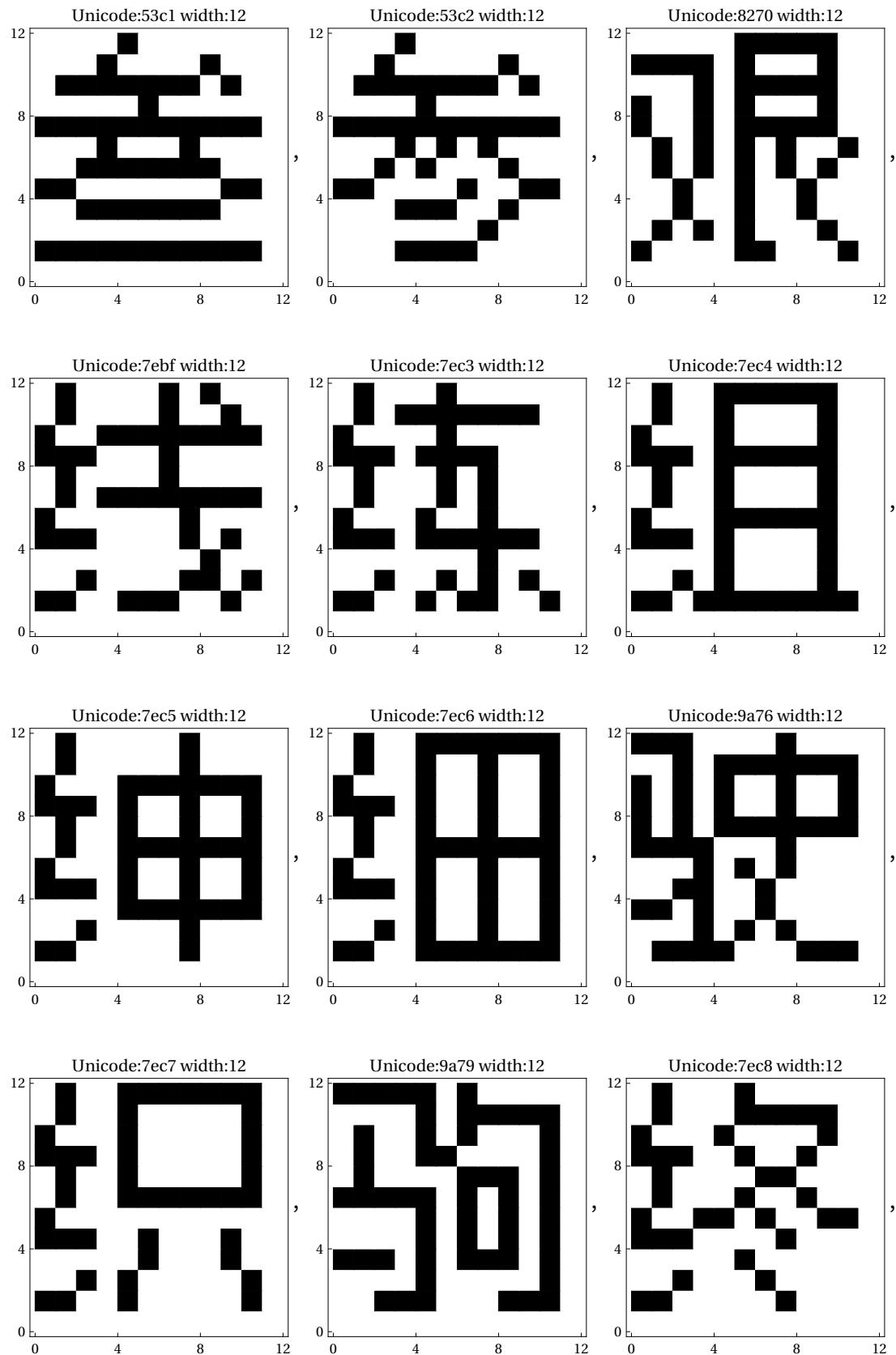


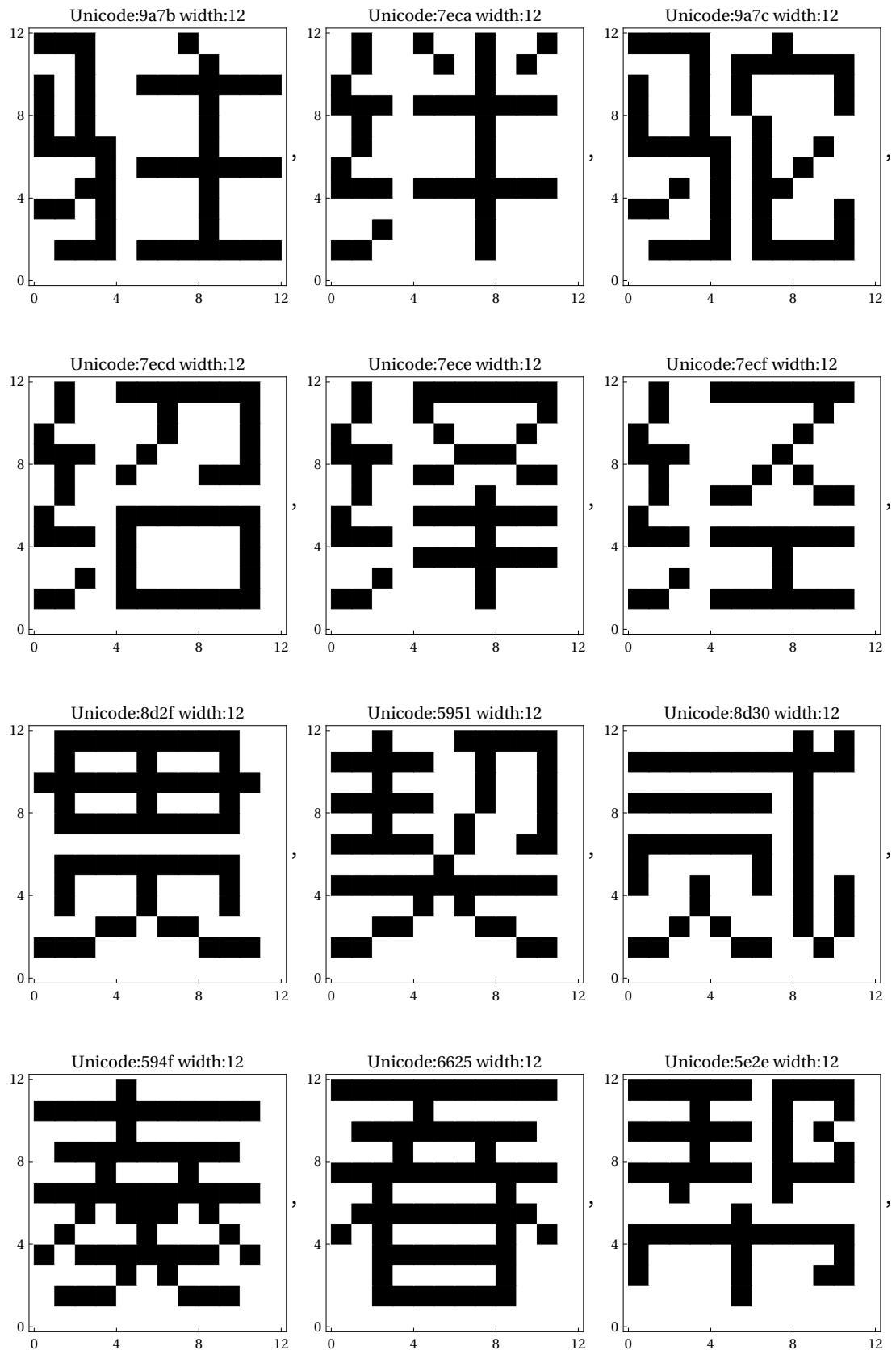


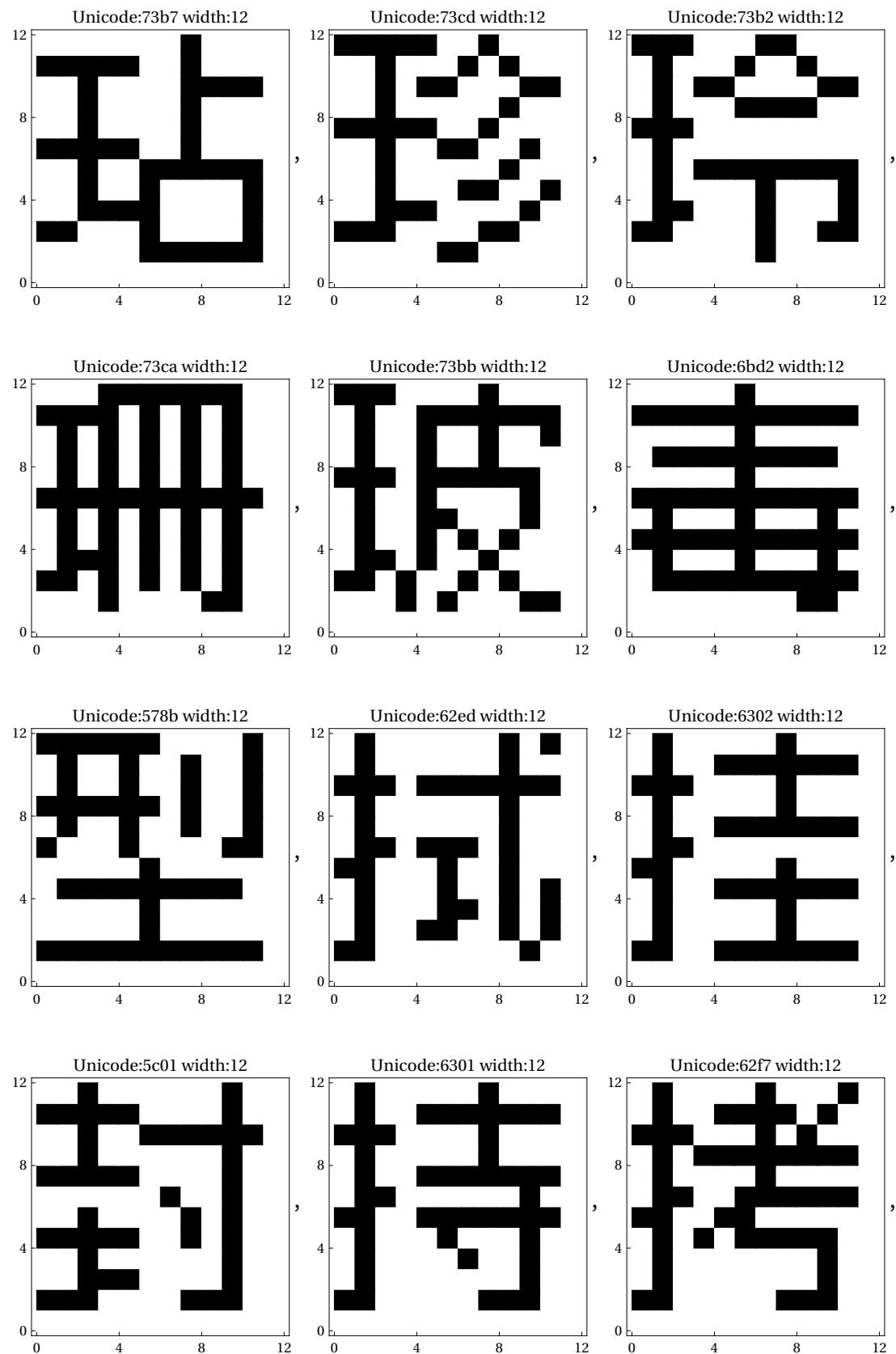


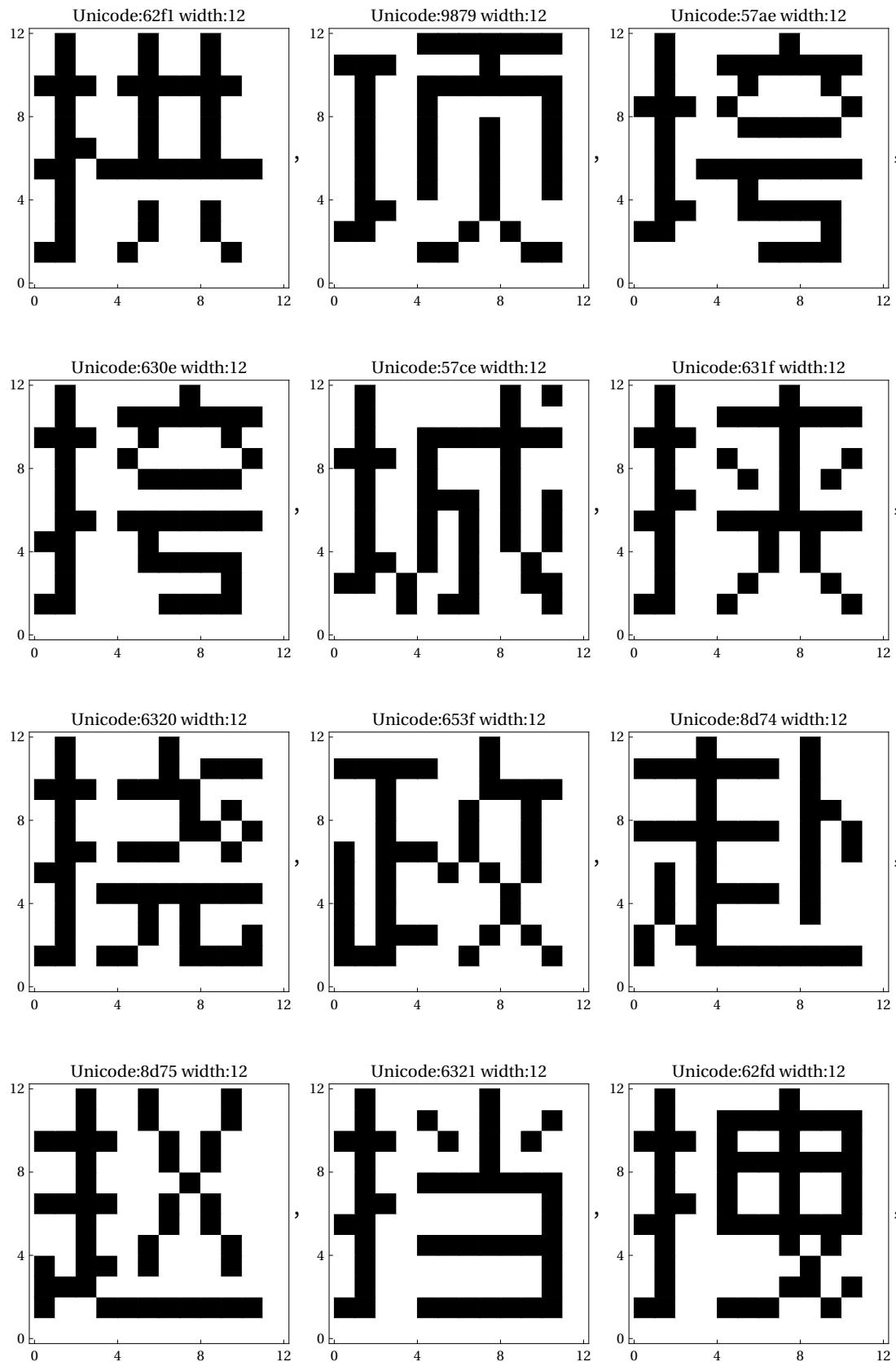


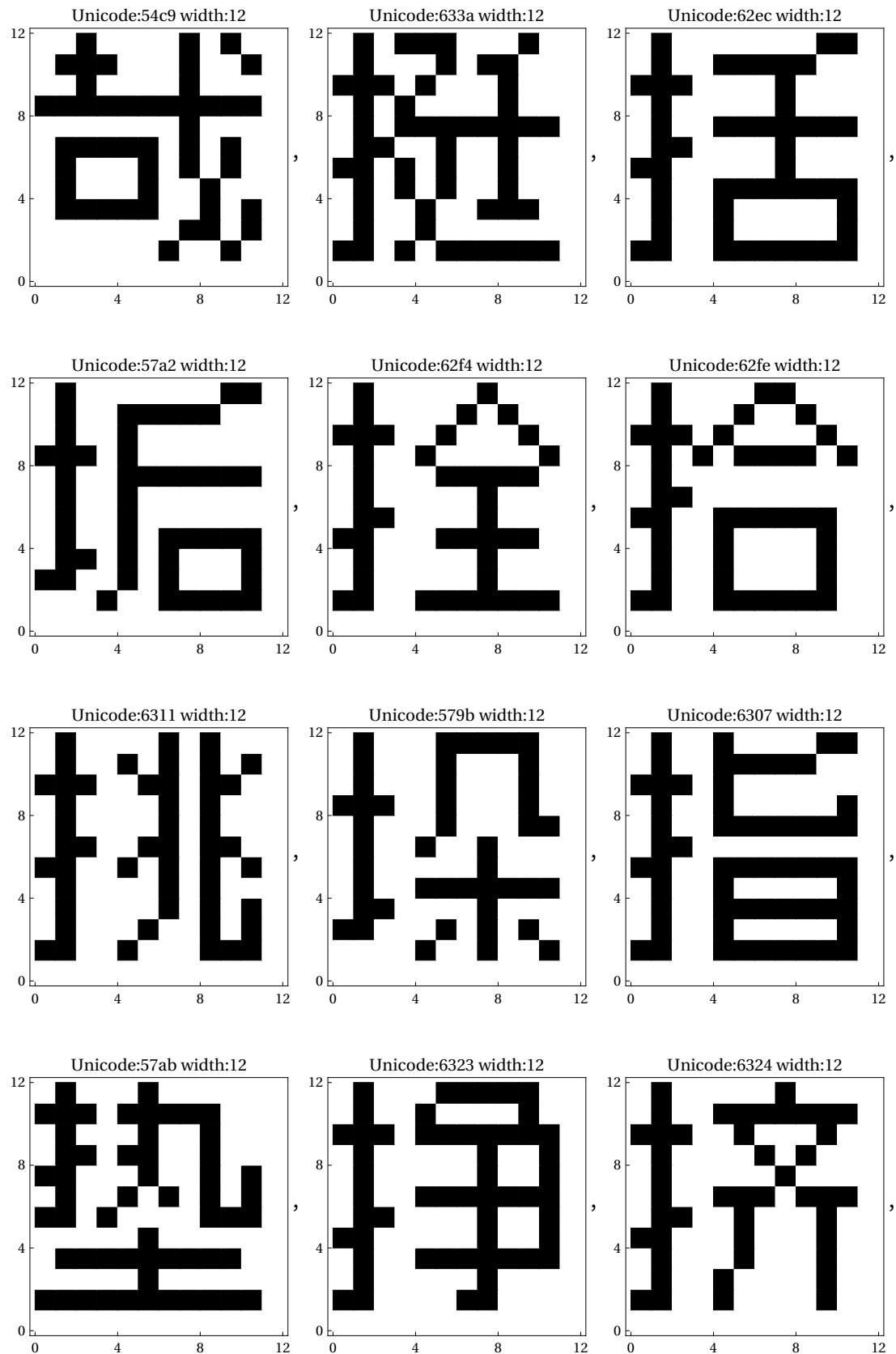


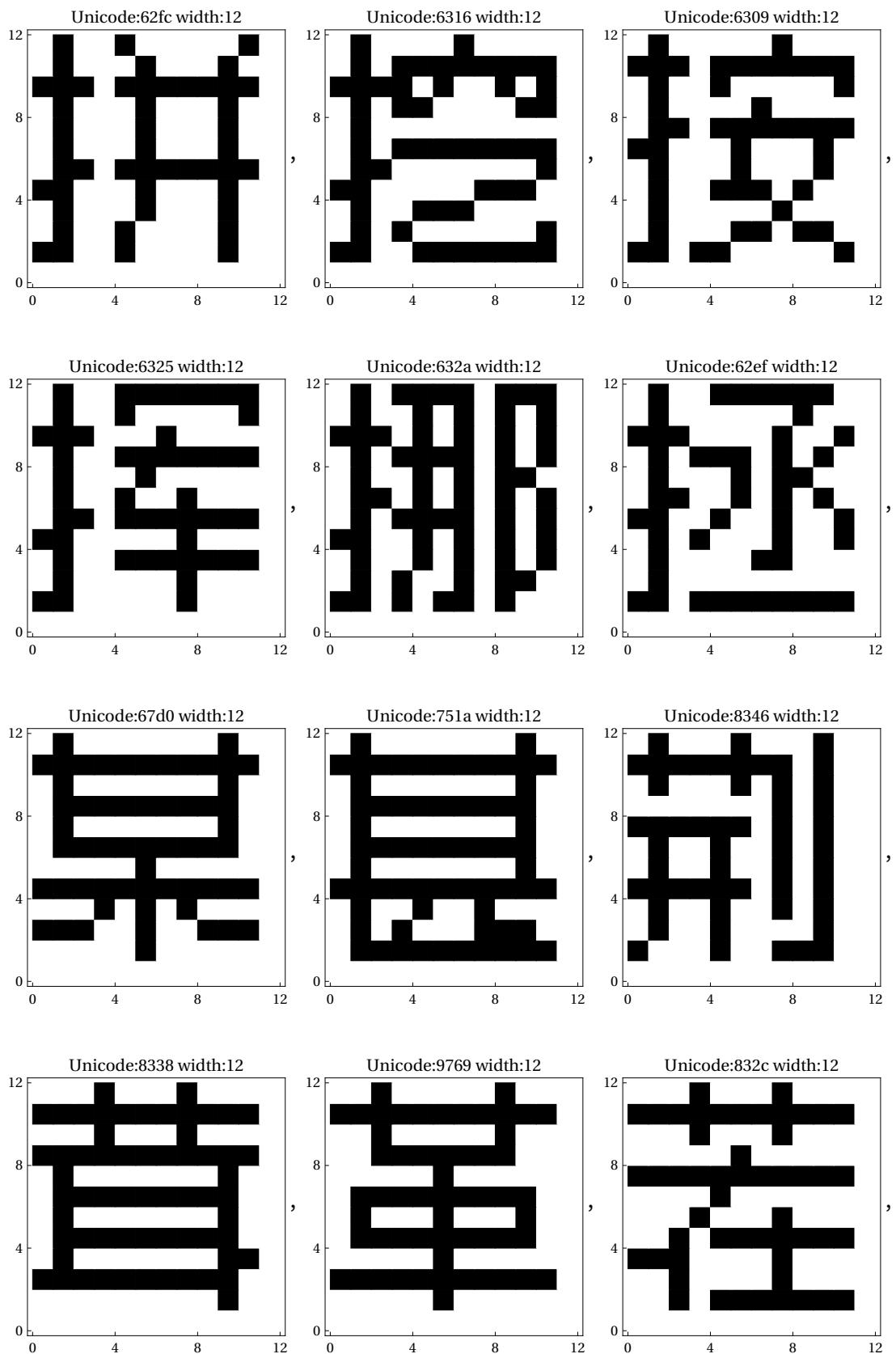


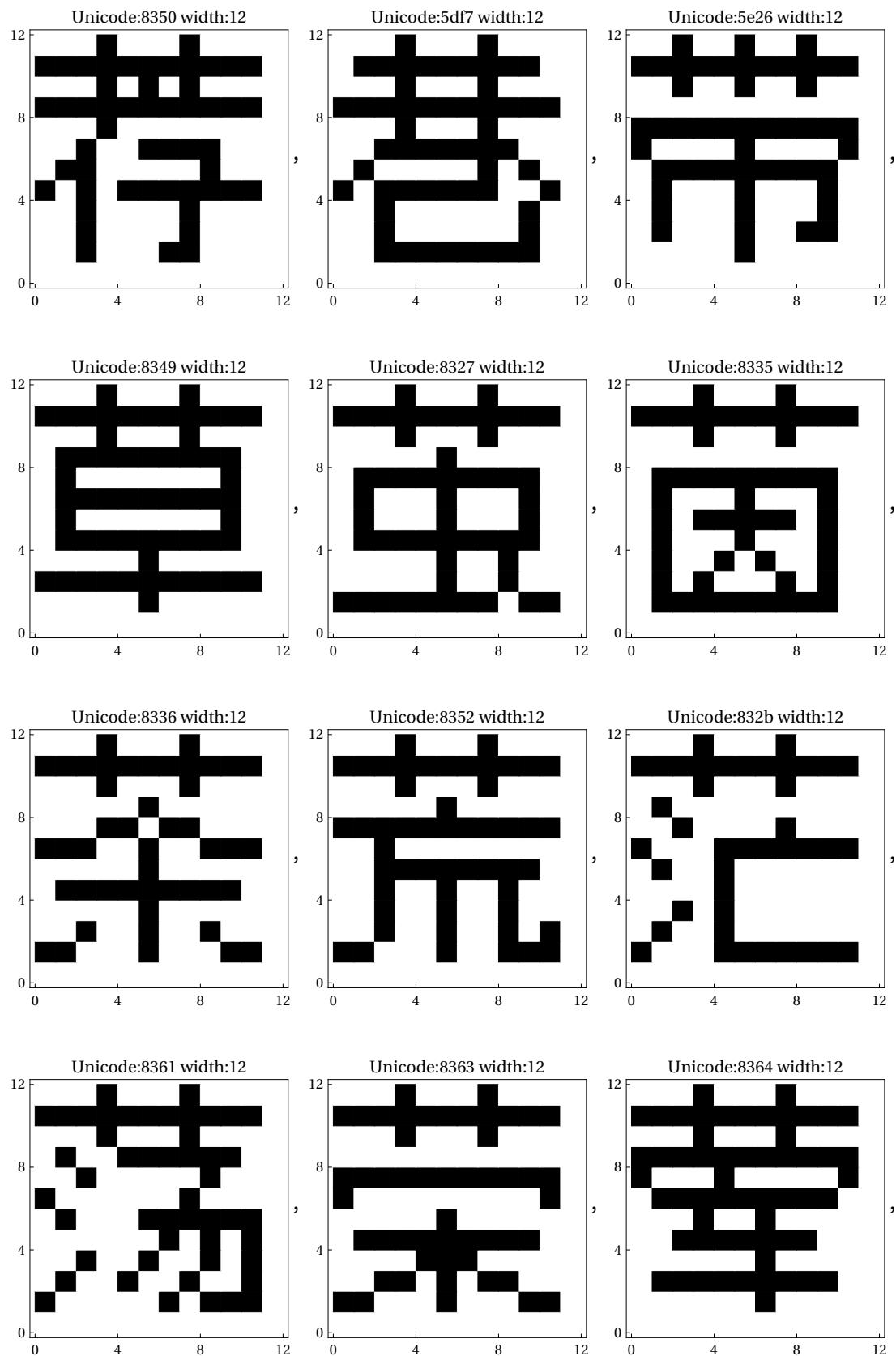


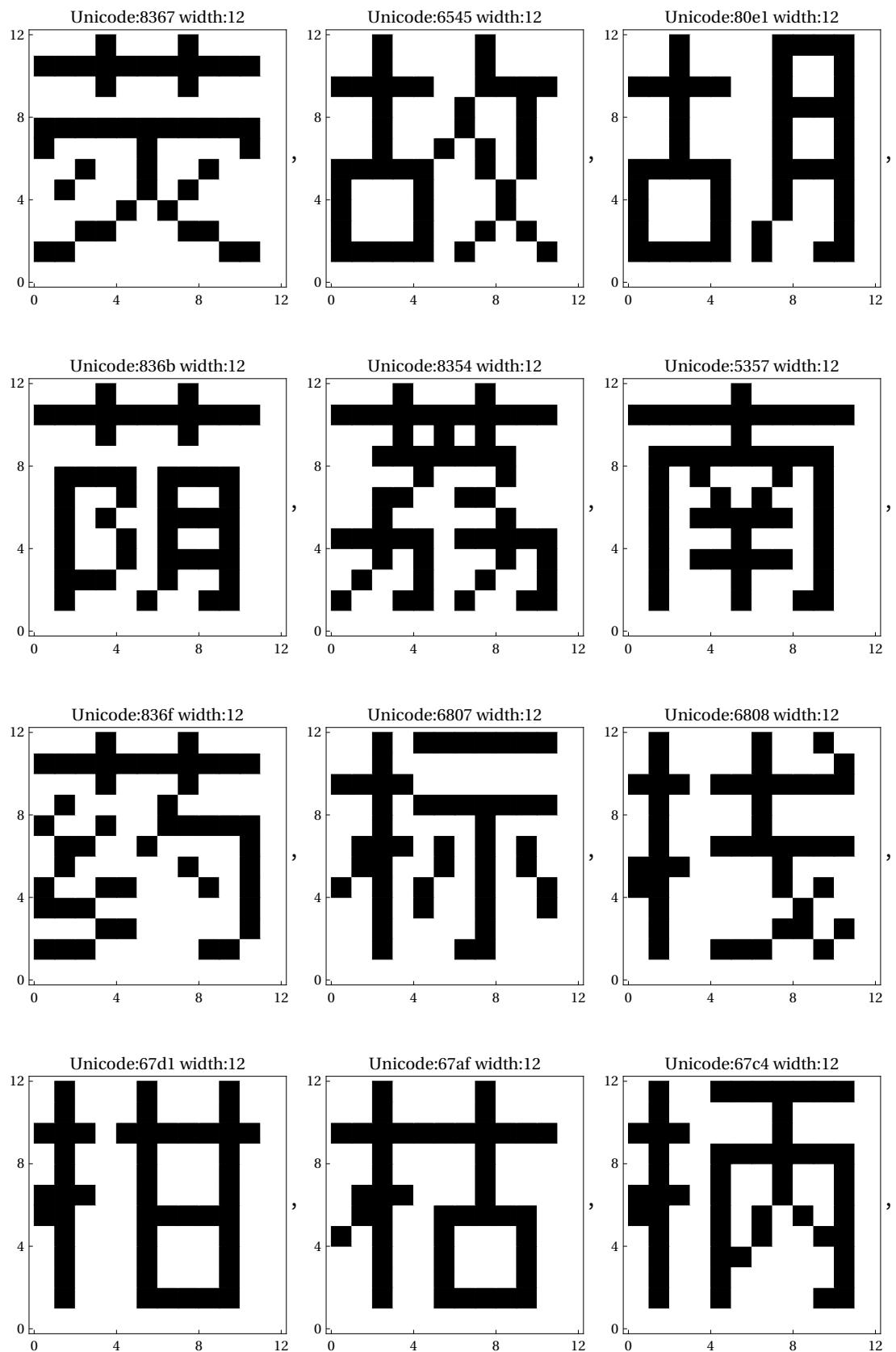


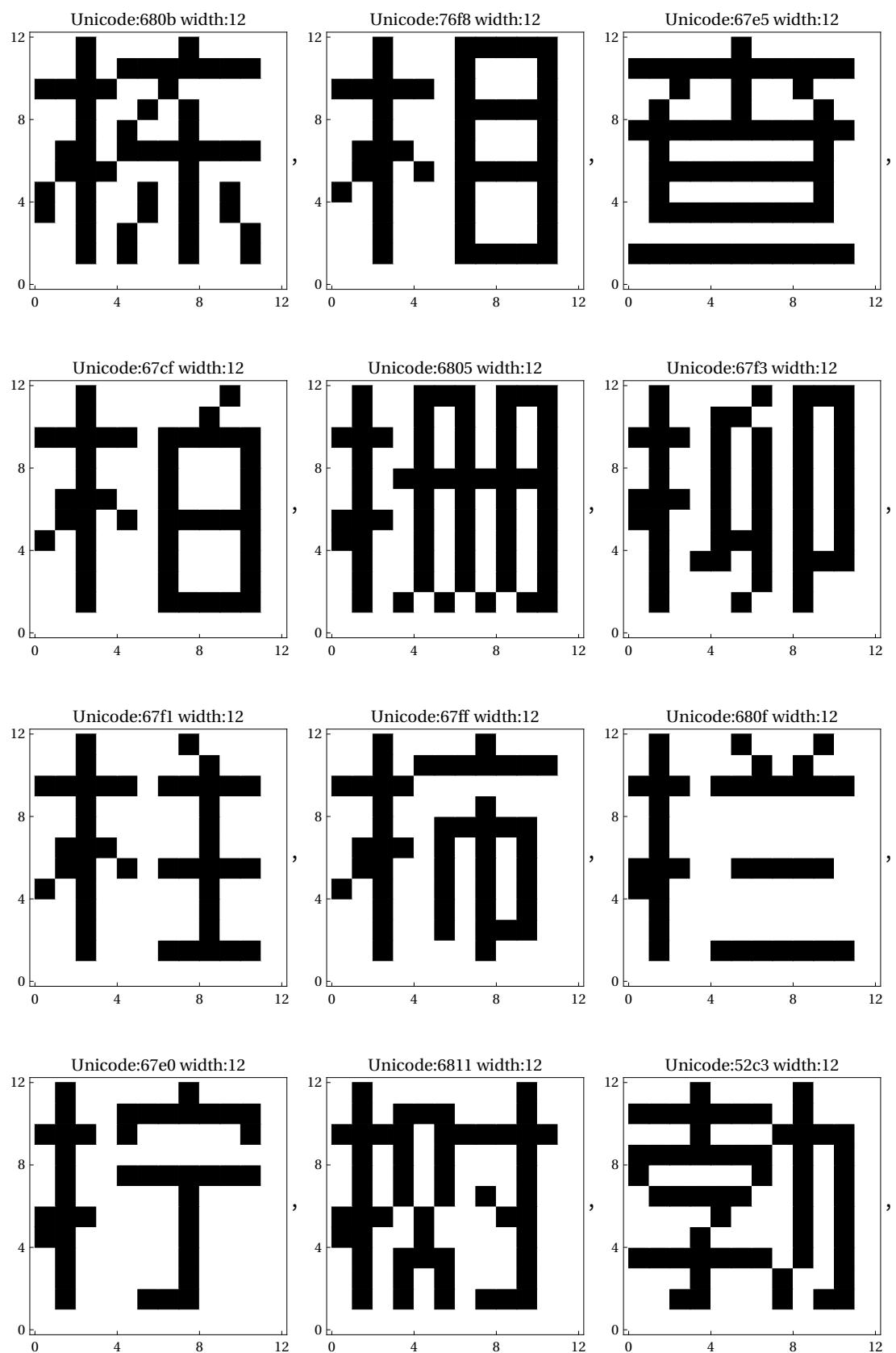


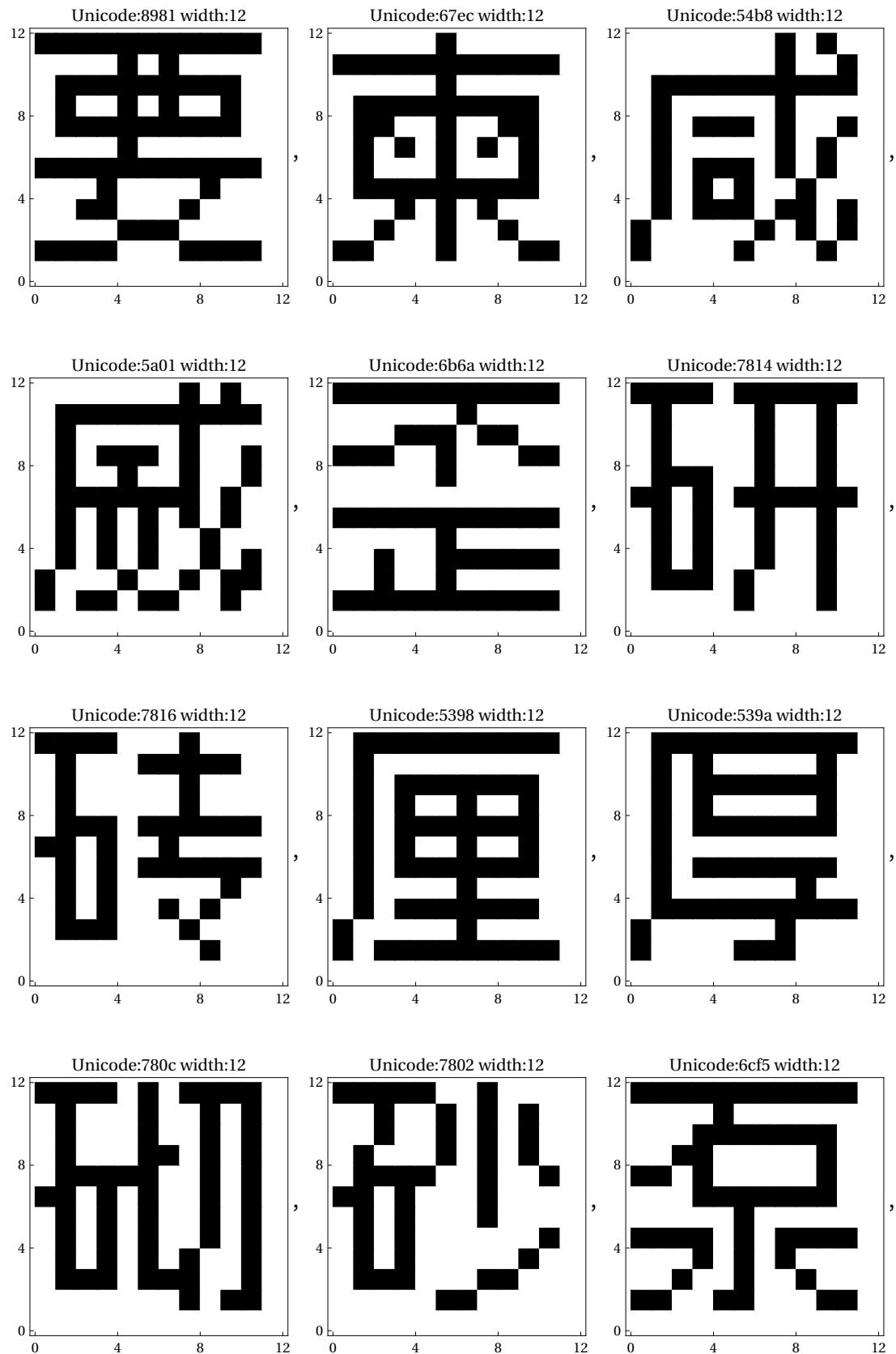


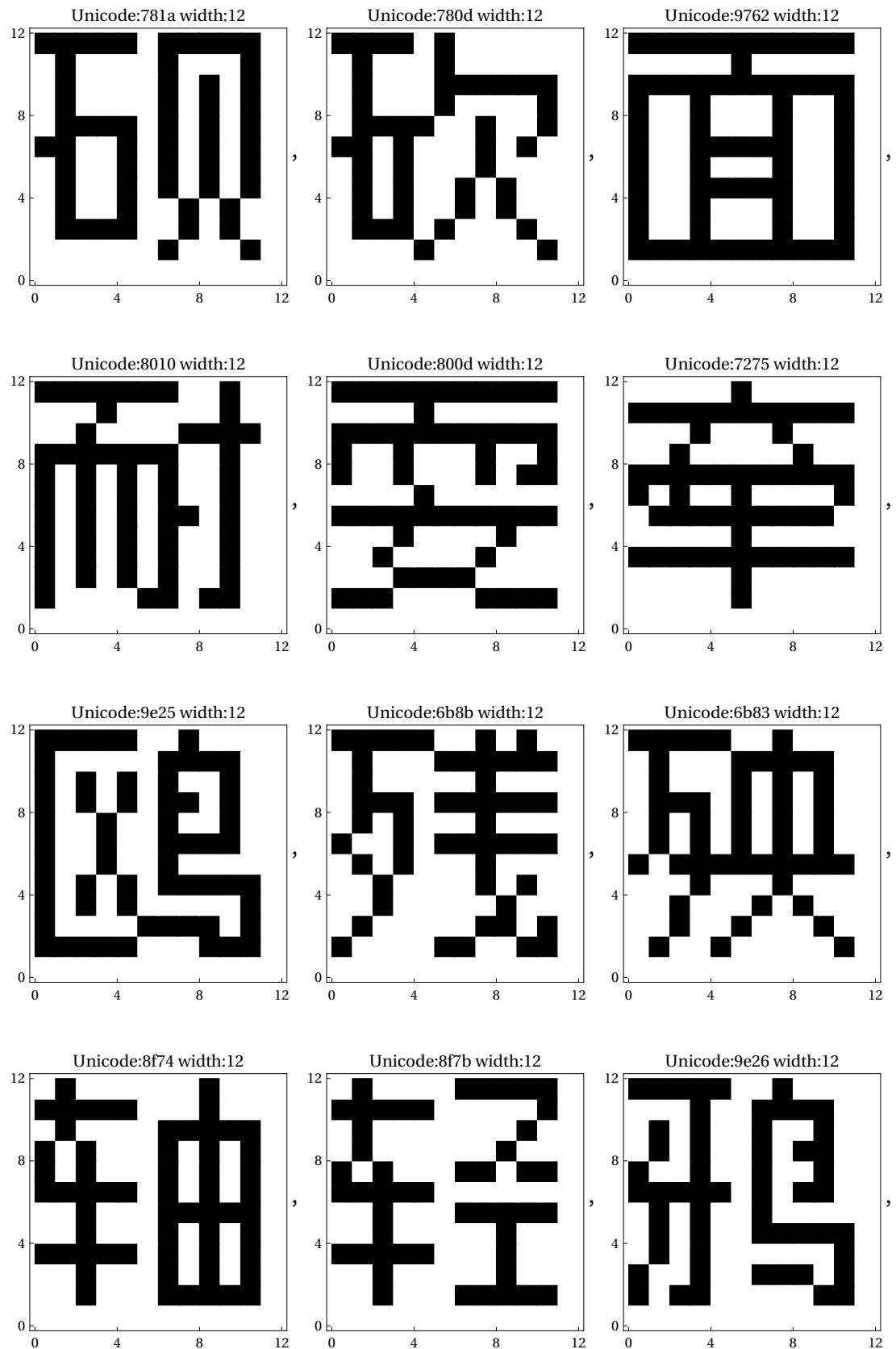


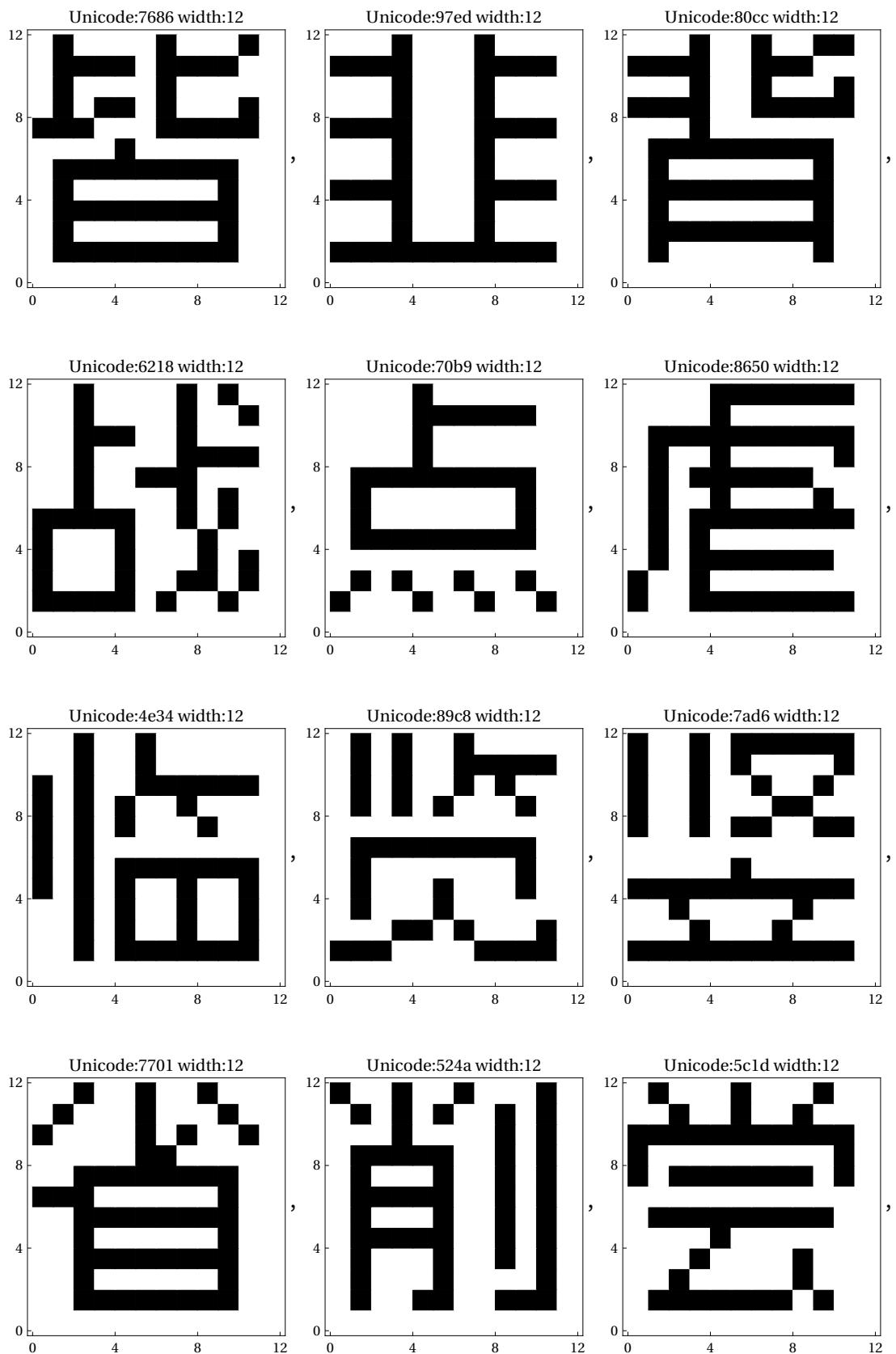


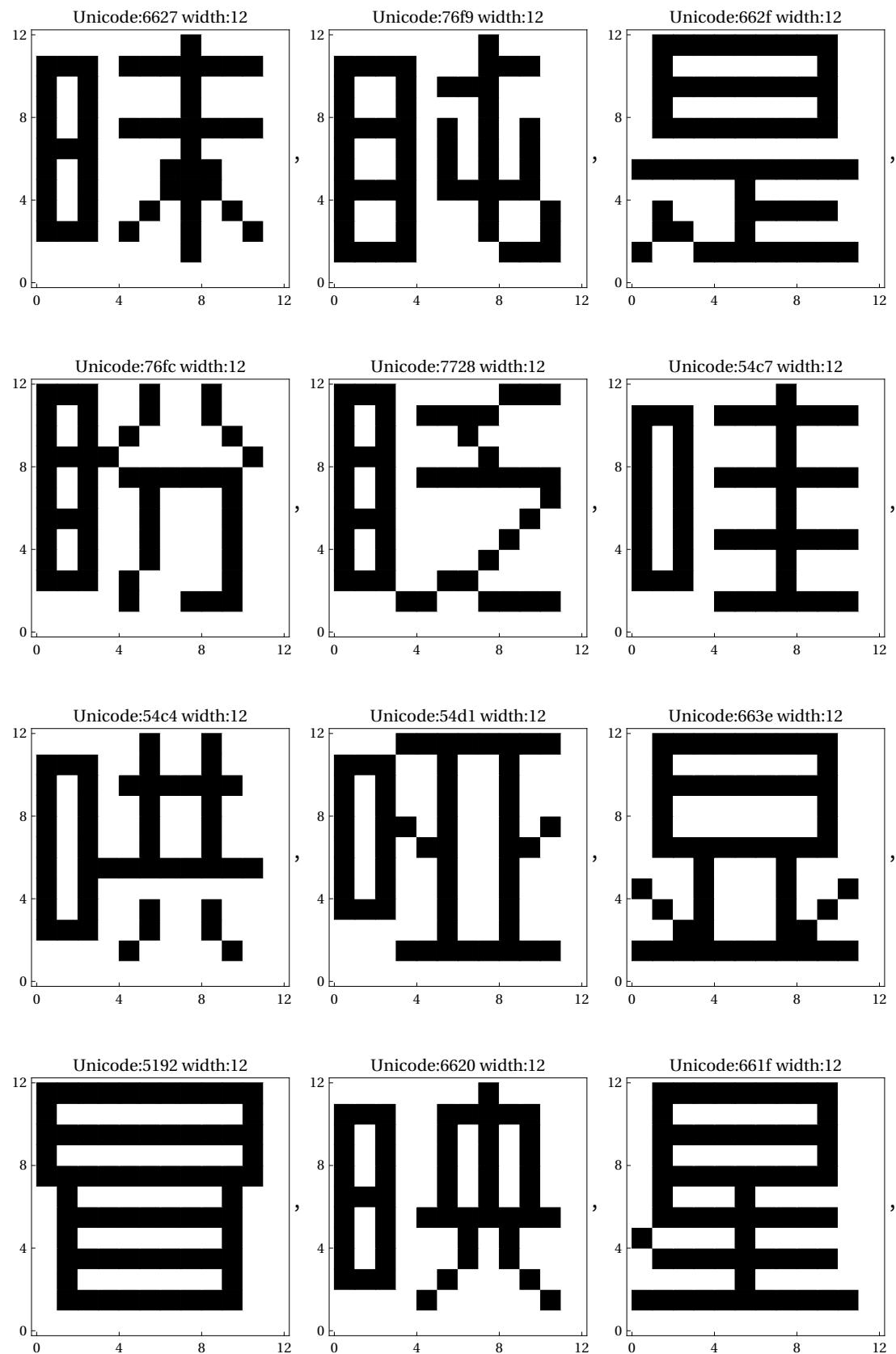


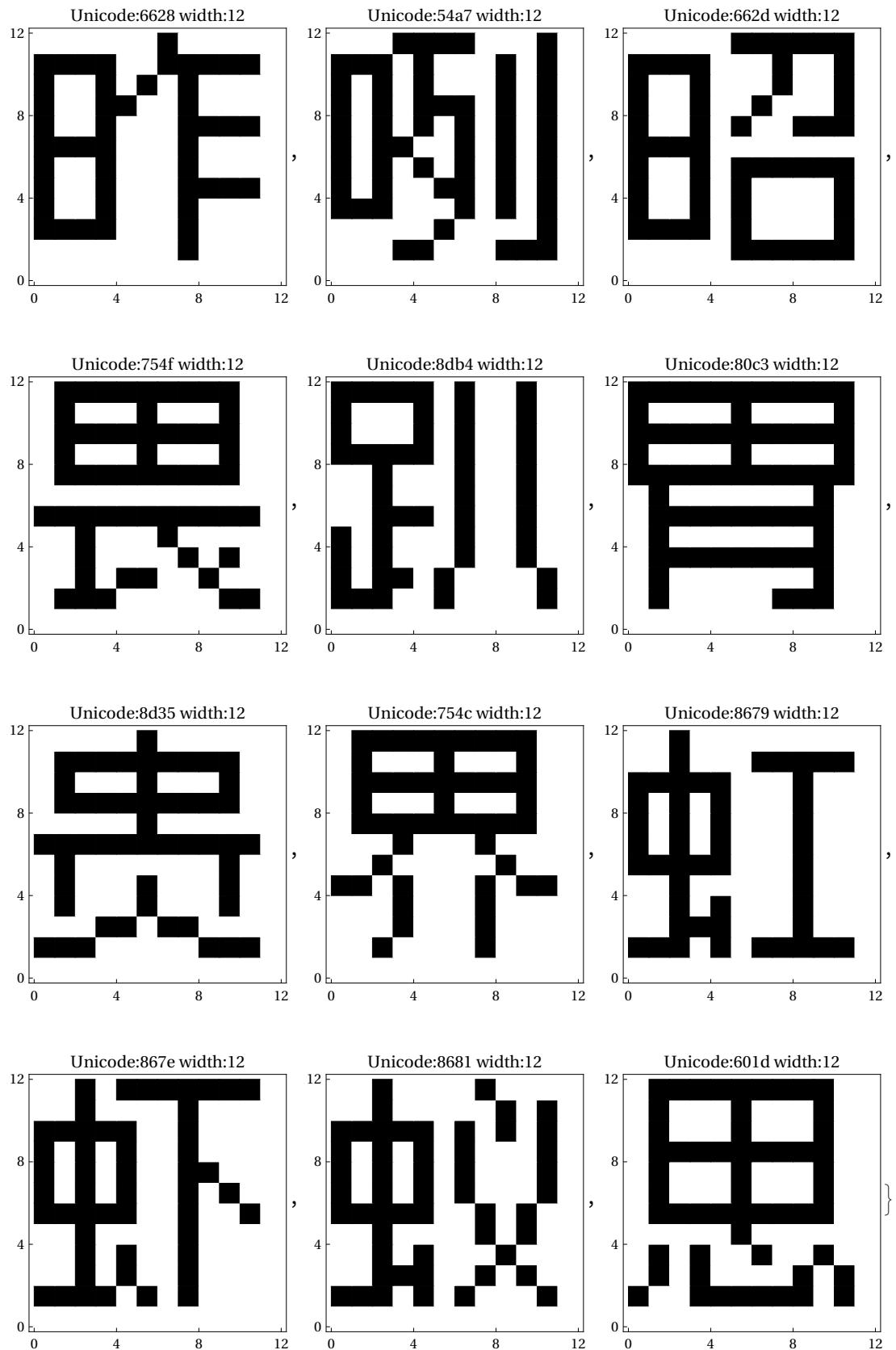













---

**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, 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
```