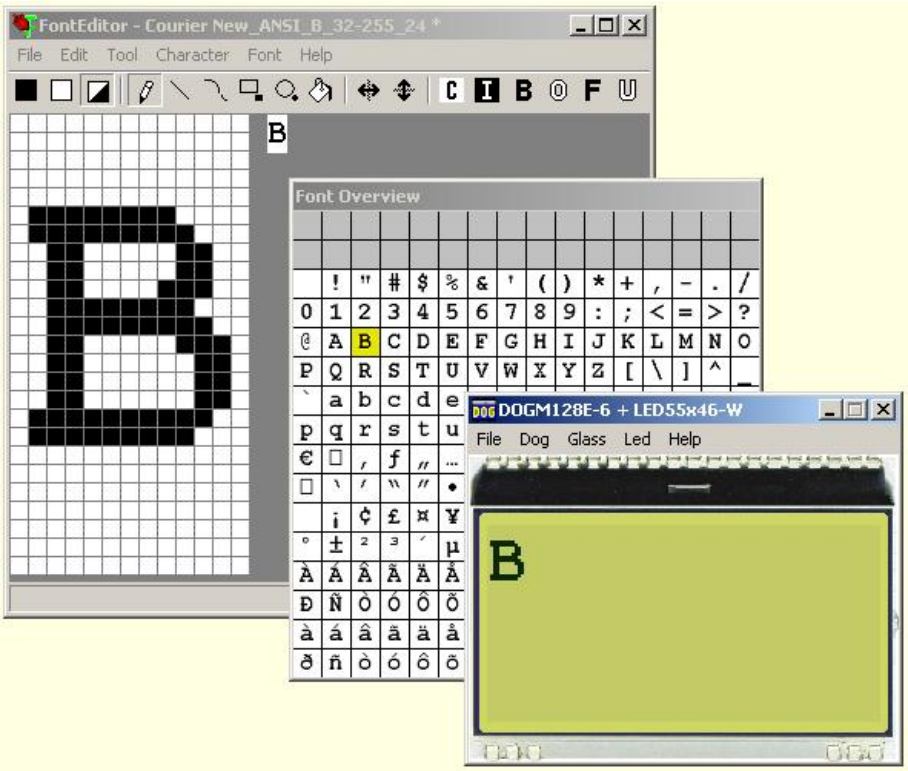


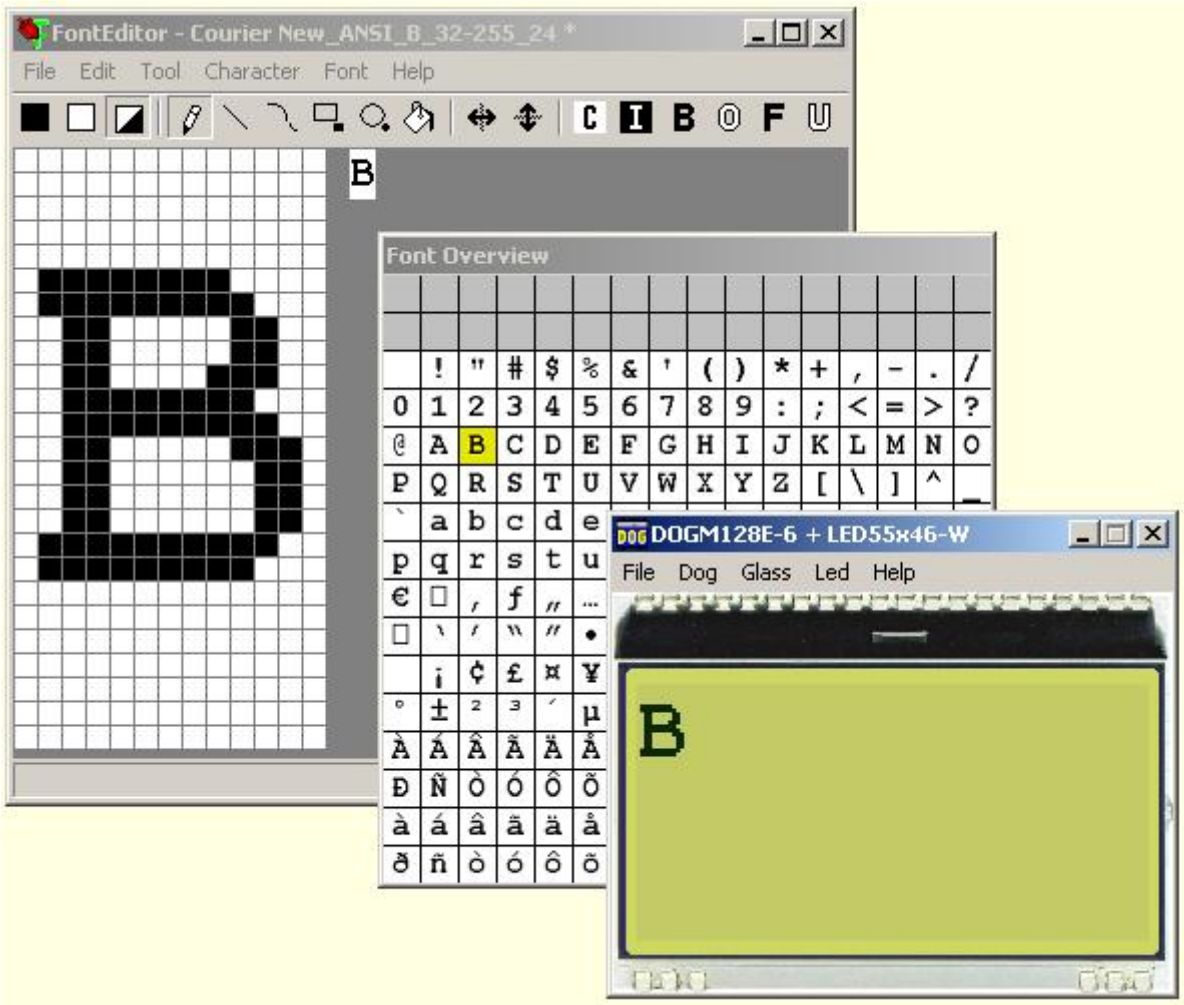
FontEditor V1.0 manual

Dezember 2009

Zeppelinstrasse 19, D-82205 Gilching
Phone +49-8105-778090, Fax +49-8105-778099
http://www.lcd-module.de

Table of Contents

1 General



All graphic displays from ELECTRONIC ASSEMBLY EA DOG series do provide a full graphic memory. To display some characters and text, a small bitmap needs to be written into this graphic memory. The FontEditor from ELECTRONIC ASSEMBLY generates those bitmap files. It will be included in µC sourcefiles then.

Use the ready made on-stick fonts *.fv
The FontEditor comes with some ready made fonts, they are optimized for good readability.

4x6.fv[20], 5x6.fv[21], 6x8.fv[22], 8x8.fv[23], 7x12.fv[24], 8x16.fv[25],
16x32_numbers.fv[26], 48x64_numbers.fv[27], 6x8_cyrillic.fv[28], 8x16_cyrillic.fv[29]

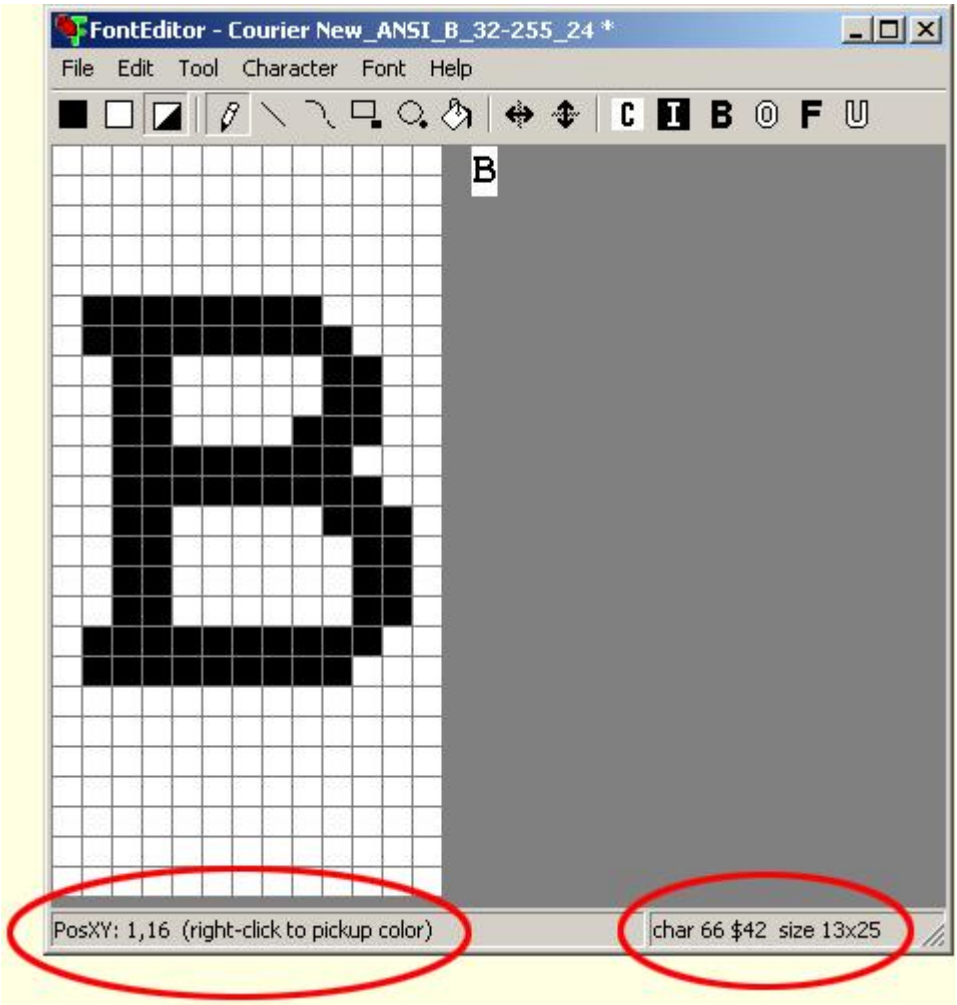
Thousand more fonts can be created easily with the help of Windows Font Import[7].

ELECTRONIC ASSEMBLY
making things easy
Zeppelinstrasse 19, D-82205 Gilching
Phone +49-8105-778090, Fax +49-8105-778099
http://www.lcd-module.de

2 Overview

2.1 Main window

Double click on the file FontEditor.exe opens the EA FontEditor.



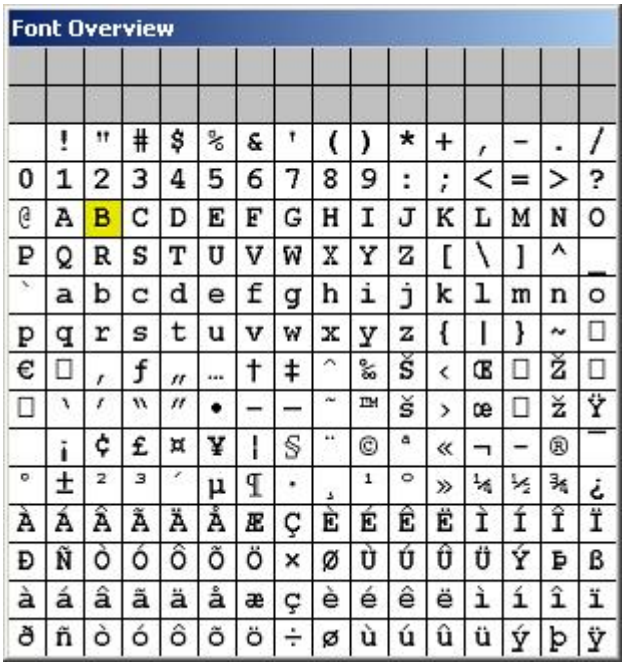
Note that the lower helpline does show useful additional information.

There are 2 possibilities to work with the FontEditor tool. You can click through the menu with the mouse or you can use the Shift, Alt Ctrl/Strg with some letters of the keyboard.

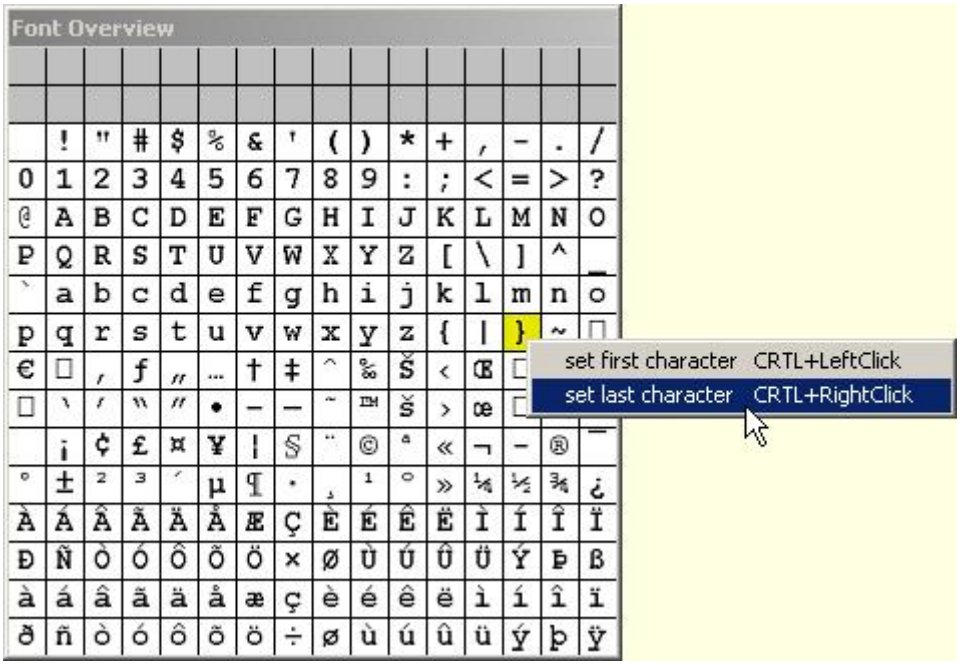
After pressing the Alt key in the Main Menu the letter with which you can open the menu item in combination with the Alt button is underlined.

2.2 Font Overview

The Window **Font Overview** is a preview function for the whole character set (codes 0x00 to 0xFF). The white fields do mark all included characters, while the gray fields represent non used codes. The yellow area marks the active character.



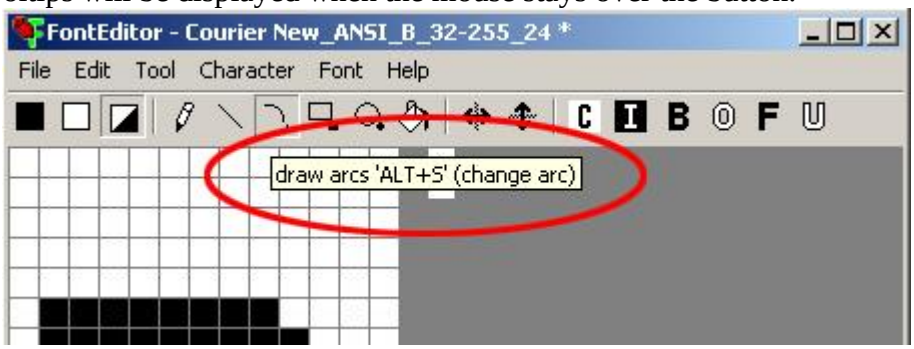
With a right-mouse-click the font can be cut. This action safes memory space.
Use left-mouse-click to enlarge the range of included character codes.



2.3 Toolbar

The most important functions of the Menu Tool 14 and Character 15 are also available in the toolbar for quick access.

Note that tooltips will be displayed when the mouse stays over the button.



Black Color
Draw color black. Every action you are doing will paint black color to the pixels you are touching.

White Color
Draw color white. Every action you are doing will paint white color to the pixels you are touching.

Inverse Color
Draw color inverse. Every action you are doing will paint the opposite color to the pixels they had been before. This means white pixels will get black and black will get white.

Draw Dots
Every dot you are clicking gets changed to the draw color.

Draw Lines
The software draws a straight line from one position to another as good as possible.

Draw Arc
The software draws an arc from one position to another as good as possible.
Alt+S changes the direction of bending the arc, also during execution.

Draw Rectangle
A rectangle is drawn from the starting point to the point leave the mouse button.
Alt+R changes the option to fill the rectangle, also during execution.

Draw Circles
The software draws a circle or ellipsoid from one point to another as good as possible.
Alt+C changes the option to fill the circle, also during execution.

Fill area
Click to an area and it will be filled with the draw color (black or white)

 **Horizontal Mirror**
will mirror the character in X-direction.

 **Vertical Mirror**
will mirror the character in Y-direction.

 **Clear**
Erases the character (white).

 **Invert**
Black dots turn to white and white dots to black.

 **Bold**
Makes the character bold, does mean adds 1 dot around.

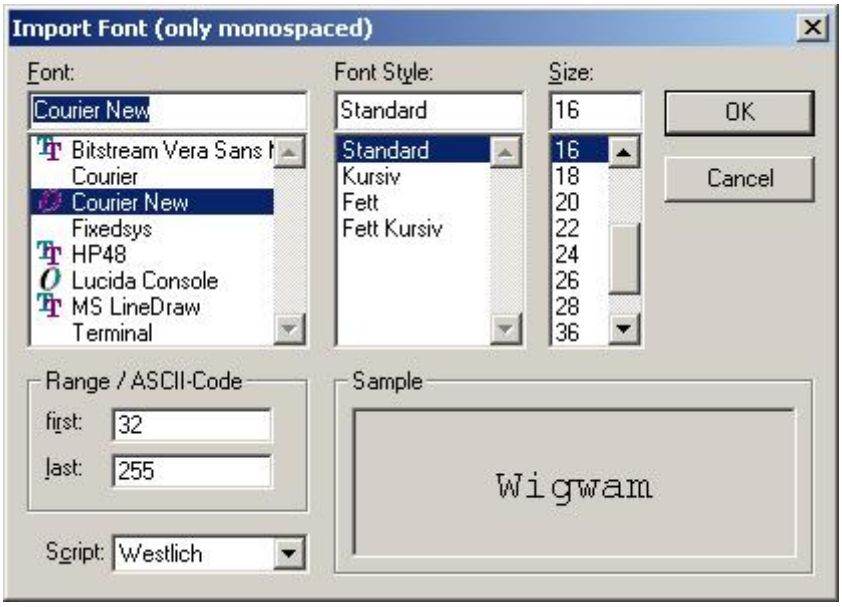
 **Outline**
Adds an outline around the character and deletes the inner area.

 **Fill**
Fills inner areas.

 **Unfill**
Unfills inner areas.

2.4 Import Fonts

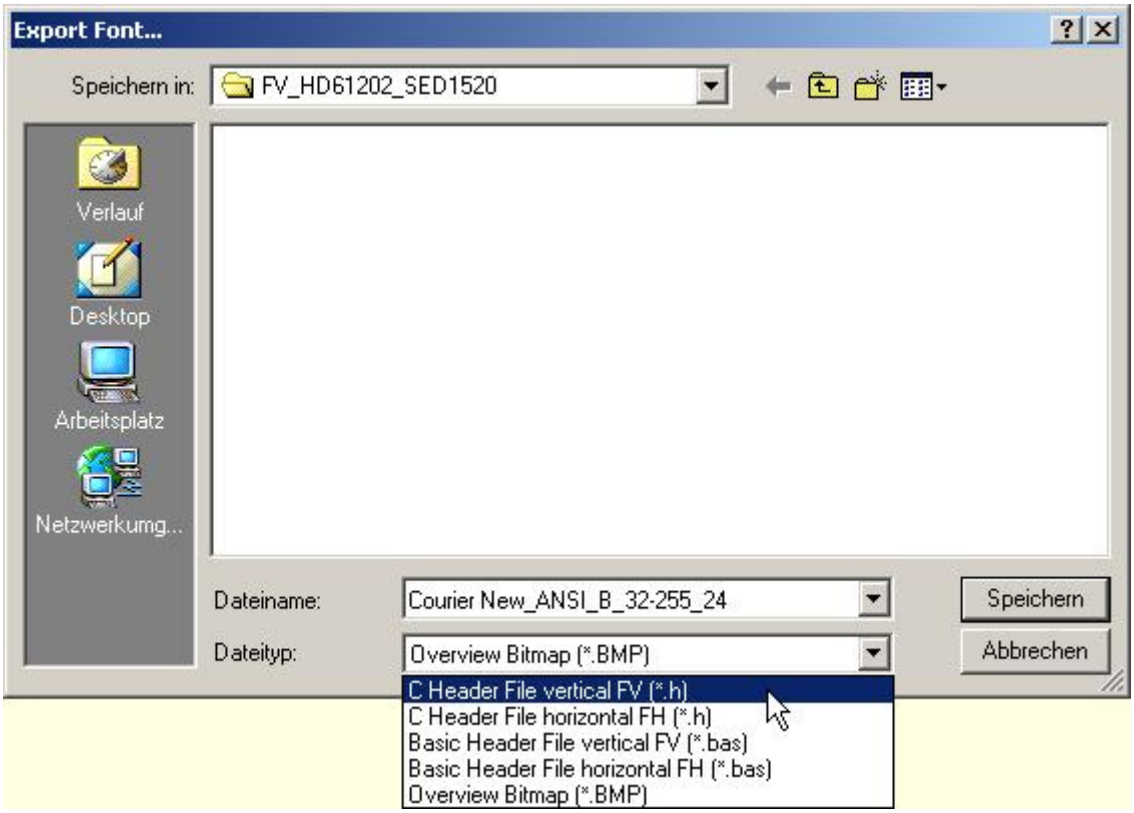
Import TrueType fonts
A smart way to generate larger fonts is to scan and convert Windows TrueTypeFonts. This is recommended for medium and large fonts with a pixel height of min. 16.



Note that the font styles "Kursiv"/"Italic" are not supported.

2.5 Export Fonts

Export font file
With the Export function the font is converted to a text file which may be directly included in a C and basic source code.
For other programming languages you can use the array data in *.h files by manually cut&paste into your source code.



Export a Overview Bitmap

File '16x32ZIF.fv' as '16x32ZIF.bmp'

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)											×	+	,	−	.	÷
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:					

Export a C-Header File

```
/* File 'AB.fv' as include

 the array starts with a 8 byte header:
  1st Byte: 'F' first 2 bytes are always FV
  2nd Byte: 'V' for FONT VERTICAL
  3rd Byte: First code to define
  4th Byte: Last  code to define
  5th Byte: Width of character in dots
  6th Byte: Height of character in dots
  7th Byte: Height of character in bytes
  8th Byte: Bytes needed for each character (1..255)
```

```
            or 0 for big fonts calculate WidthInDots * HeightInBytes
  After that font data will follow */

#define Font_AB_fh_LEN  40

unsigned char Font_AB_fh[Font_AB_fh_LEN] =
{
   70, 86, 65, 66,  8, 16,  2, 16,
    0,248, 12, 12, 12, 12,248,  0,  0, 63,  3,  3,  3,  3, 63,  0,
    0,252,140,140,140,140,120,  0,  0, 63, 49, 49, 49, 49, 30,  0
};
```

Export a Basic Source Code

```
'File 'AB.fv' as array

'the array starts with a 8 byte header:
' 1st Byte: 'F' first 2 bytes are always FV
' 2nd Byte: 'V' for FONT VERTICAL
' 3rd Byte: First code to define
' 4th Byte: Last  code to define
' 5th Byte: Width of character in dots
' 6th Byte: Height of character in dots
' 7th Byte: Height of character in bytes
' 8th Byte: Bytes needed for each character (1..255)
'           or 0 for big fonts calculate WidthInDots * HeightInBytes
' After that font data will follow

Dim Font_AB_fh(40)=_
{_
   70, 86, 65, 66,  8, 16,  2, 16,_
    0,248, 12, 12, 12, 12,248,  0,  0, 63,  3,  3,  3,  3, 63,  0,_
    0,252,140,140,140,140,120,  0,  0, 63, 49, 49, 49, 49, 30,  0_
};
```

2.6 DOG-Simulator

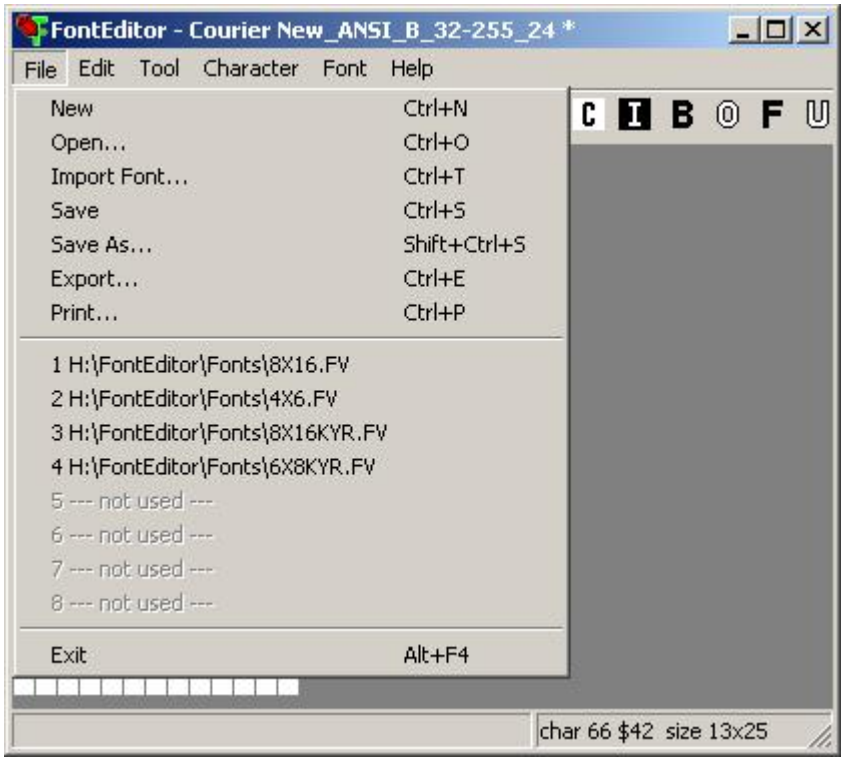
Simulation (StartDog.exe)
The simulator software for the EA DOG displays does perfectly cooperate with the FontEditor. When both programs are started, the simulated DOG display immediately shows the current character. With that it is easy to assess immediately size and design.
You will find a copy of StartDog.exe on the EA USBSTICK-FONT in \StartDog\StartDog.exe

LiveView on EA 9780-2USB
When both FontEditor.exe and StartDog.exe are running and the testboard EA 9780-2USB is connected to the USB, then the character is immeditaley visible live on the plugged-in DOG display.

Individual Text
With the menuentry Text for DOG-Simulator 13 you can change the displayed characters.

3 Menu

3.1 File



New
Creates a new Font, you are asked for character size in pixels in width and height.
After clicking the Ok button the new Font is created, ready to modify.

Open
Opens a window with several already existing fonts [2] in different sizes which you can modify and save under your own name.

Import Font
Gives the possibility to import fonts [7] (mono spaced only) from Windows System.
You can choose the character height you want to have and the ASCII Code range you want to import.
You can modify it letter by letter and store it as your own character set. Please be careful not to violate any copyrights.

Save
You can save previously created fonts under their original name. If it had not yet been saved before you will get the same Window as under "Save as" which asks for a name and extension.

Save as
You can change path, name and format of the font and store it there. The original font remains unchanged.

Export
Export the font [8] to different data formats:

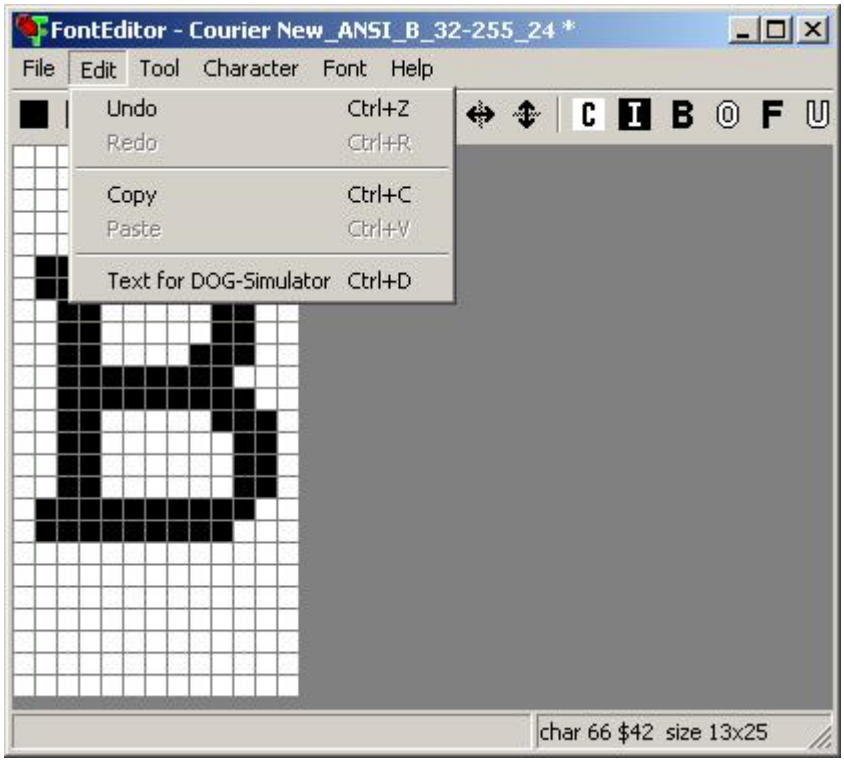
C Header File vertical FV(*.h)
C Header File horizontal FH(*.h)
Basic Header File vertical FV(*.bas)
Basic Header File horizontal FH(*.bas)
Overview Bitmap (*.bmp)

The files *.fv are for EA DOG display controller which use vertically oriented bytes (ST7565R, UC1610 and UC1701)
The *.fh files are for controller which have placed the bytes horizontally.
Overview Bitmap (*.bmp) creates a file which shows character font in a list which shows an overview of the characters and which character has which corresponding HEX-code. This *.bmp is good for documentation.

Print
Prints directly the Font Overview list with upper and lower hex codes.

Exit
This closes FontEditor.exe after asking whether the changes made should be saved.

3.2 Edit



Undo
This command gives you the possibility to make things undone step by step.

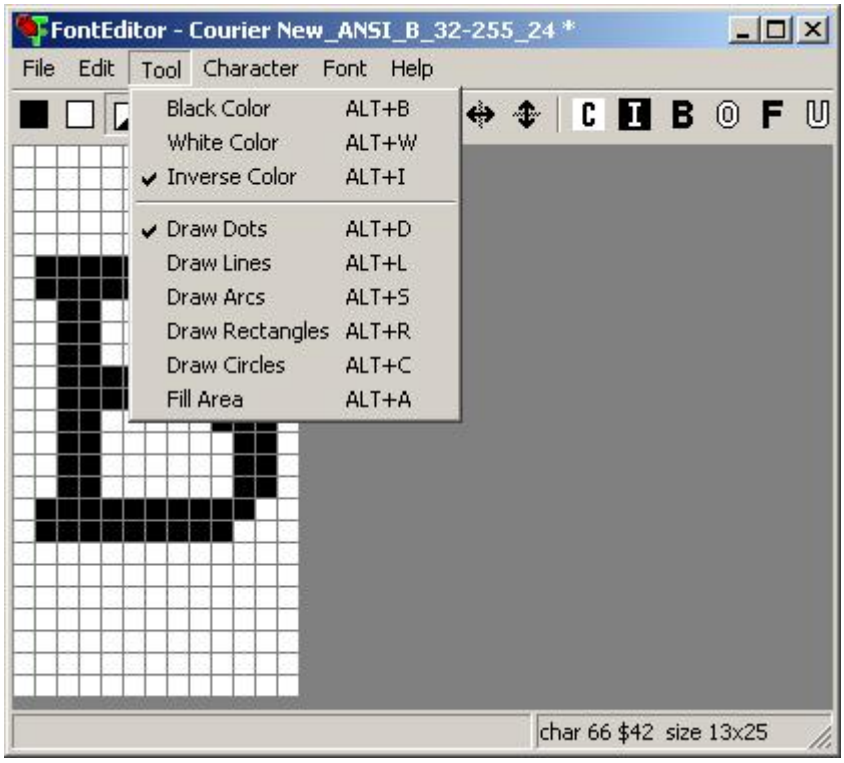
Redo
With the Redo function you can undo the steps you made with undo.

Copy
Copies the actual character into the clipboard.

Paste
Copies the clipboard content to the actual character.

Text for DOG-Simulator 10
Together with our simulation software StartDog.exe you can see immediately on the monitor how the character looks on the display. When using this simulator option you need to start "StartDog.exe". When you have in addition our new USB testboard EA 9780-2USB you can see the appearance life on the plugged in DOG module.
The sign "|" represents a line break, the "#" represents a kind of space holder when you are trying several letters out of the character font.

3.3 Tool



Black Color
Draw color black. Every action you are doing will paint black color to the pixels you are touching.

White Color
Draw color white. Every action you are doing will paint white color to the pixels you are touching.

Inverse Color
Draw color inverse. Every action you are doing will paint the opposite color.

Draw Dots
Every dot you are clicking gets changed to the draw color.

Draw Lines
The software draws a straight line from one position to another as good as possible.

Draw Arc
The software draws an arc from one position to another as good as possible.
Alt+S changes the direction of bending the arc, also during execution.

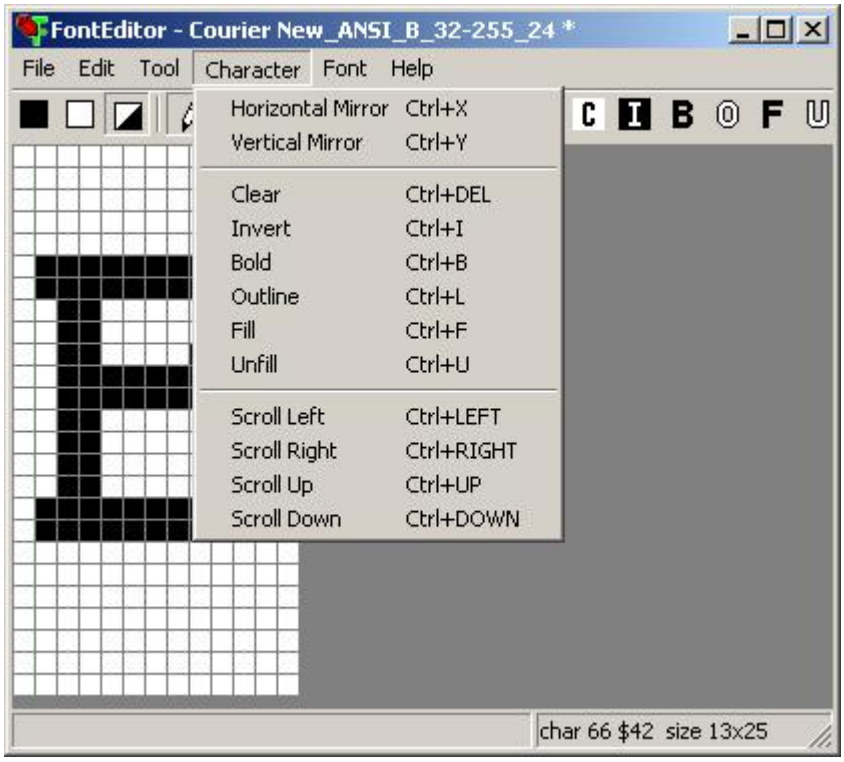
Draw Rectangle
A rectangle is drawn from the starting point to the point leave the mouse button.
Alt+R changes the option to fill the rectangle, also during execution.

Draw Circles
The software draws a circle or ellipsoid from one point to another as good as possible.
Alt+C changes the option to fill the circle, also during execution.

Fill area
Click to an area and it will be filled with the draw color (black or white)

3.4 Character

Preamble: all functions in the Menu Character do effect to 1 single character only.
The same functions can also be used to the whole font (all characters). Those functions are in the Menu Font 17.



Horizontal Mirror
will mirror the character in X-direction.

Vertical Mirror
will mirror the character in Y-direction.

Clear
Erases the character (white).

Invert
Black dots turn to white and white dots to black.

Bold
Makes the character bold, does mean adds 1 dot around.

Outline
Adds an outline around the character and deletes the inner area.

Fill
Fills inner areas.

Unfill
Unfills inner areas.

Scroll Left
Shift the whole character one bit to the left.

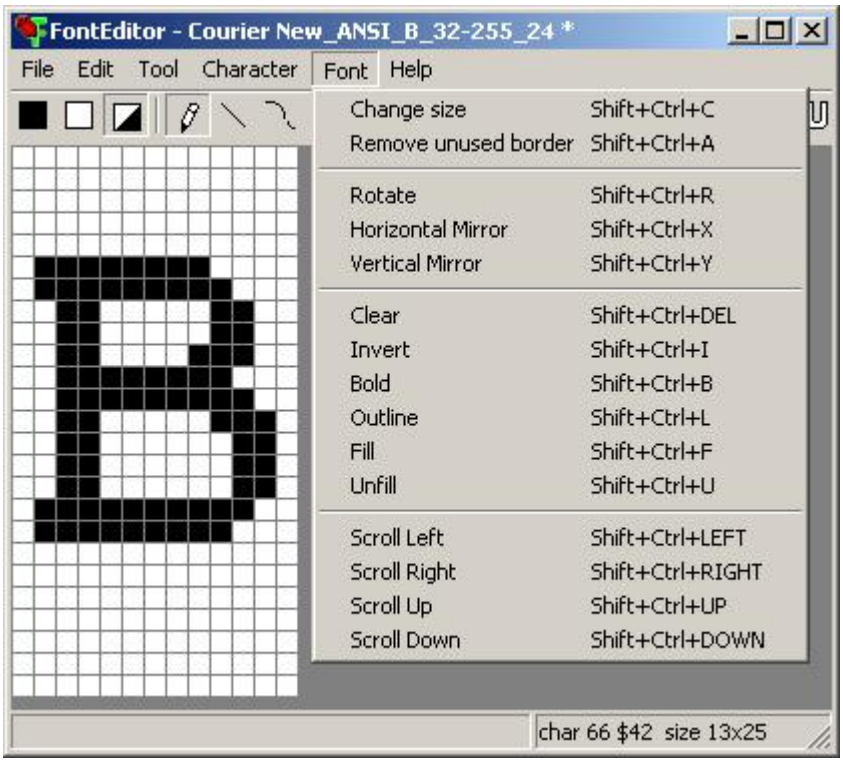
Scroll Right
Shift the whole character one bit to the right.

Scroll Up
Shift the whole character one bit upwards.

Scroll Down
Shift the whole character one bit downwards.

3.5 Font

Preamble: all functions do effect all characters simultanously. Compare with the Menu Character 15



Change size
A new draw area/size can be defined here.
Please note that this function does not scale the character !

Remove unused border
This function reduces the file size. The FontEditor scans all characters and if there are not used width and/or height, the whole character set will be cut out to minimum size.
To reduce to an optimum, sometimes it is necessary to scroll some single characters additionally and then to repeat the function "Remove unused border".

Rotate
When the EA DOG display should be used in portrait mode (e.g. 64x128 dots), then it can be assembled by 90° rotation. Of course all characters need to be rotated, too.
This function rotates the whole font by 90° clockwise.

Horizontal Mirror
will mirror the whole font in X-direction.

Vertical Mirror
will mirror the whole font in Y-direction.

Clear
Erases the whole font (white).

Invert
Black dots turn to white and white dots to black.

Bold
Makes the whole font bold, does mean adds 1 dot around.

Outline
Adds an outline around the whole font and deletes the inner area.

Fill
Fills inner areas.

Unfill
Unfills inner areas.

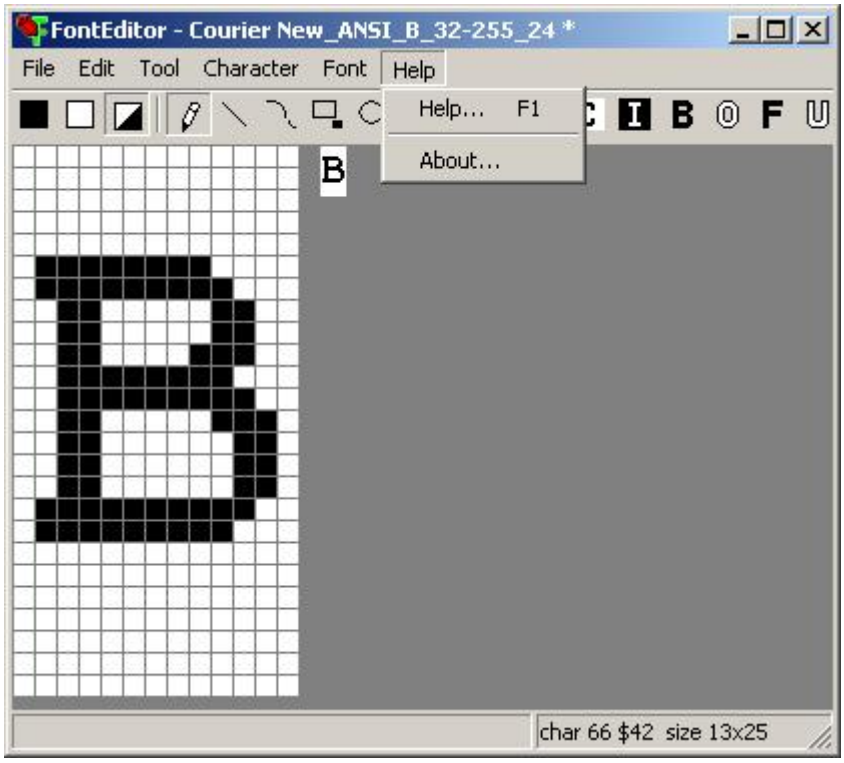
Scroll Left
Shift the whole font one bit to the left.

Scroll Right
Shift the whole font one bit to the right.

Scroll Up
Shift the whole font one bit upwards.

Scroll Down
Shift the whole font one bit downwards.

3.6 Help

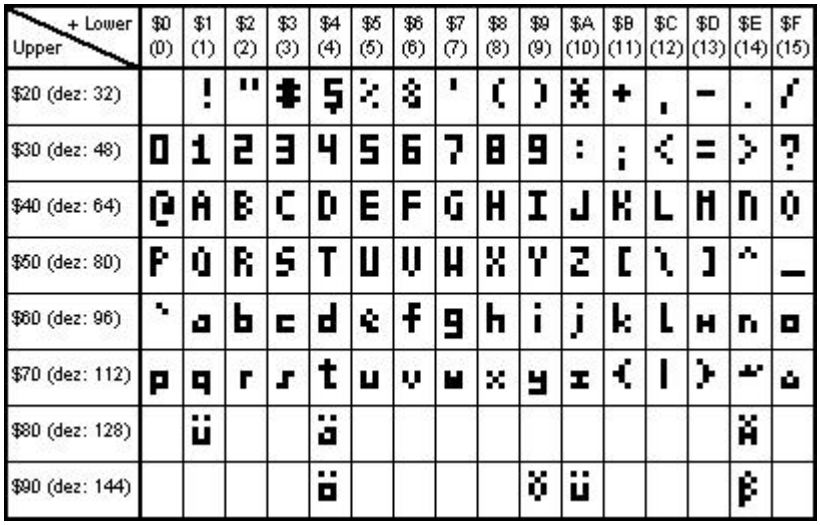


Help This click shows you the Help-File.

About This click shows you the software version of the FontEditor.

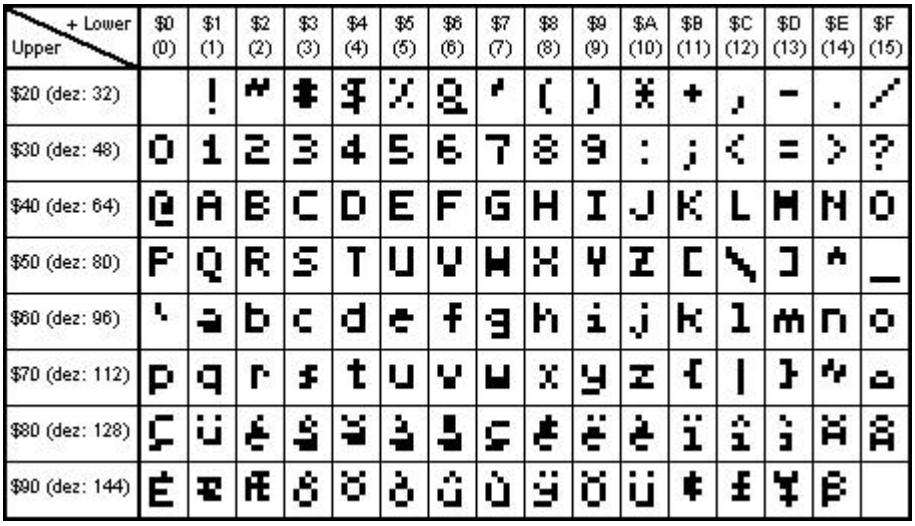
4 On-Stick Fonts

4.1 Font 4x6

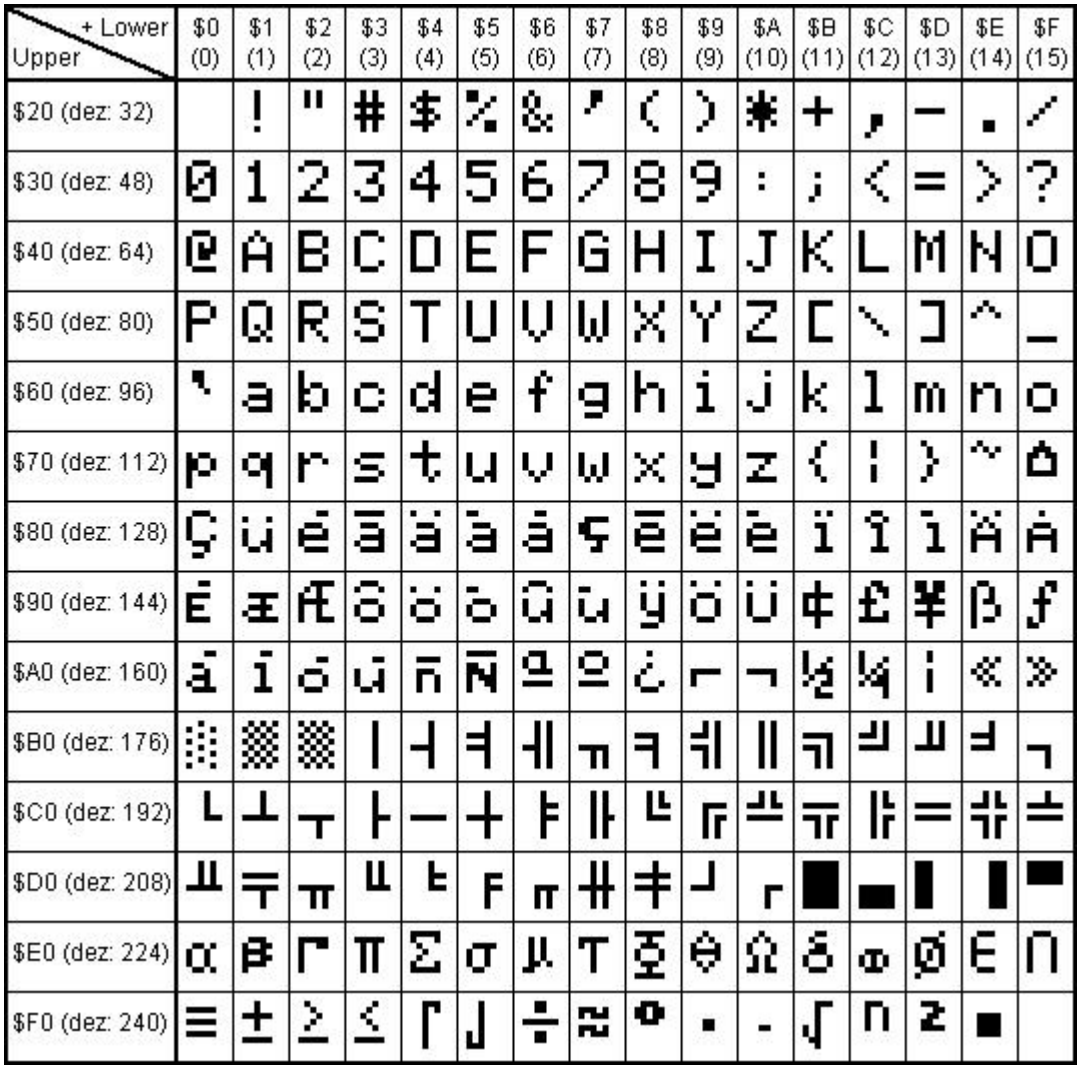


Upper \ + Lower	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)		ü			ä										Ä	
$90 (dez: 144)					ö					Ö	Ü				ß	

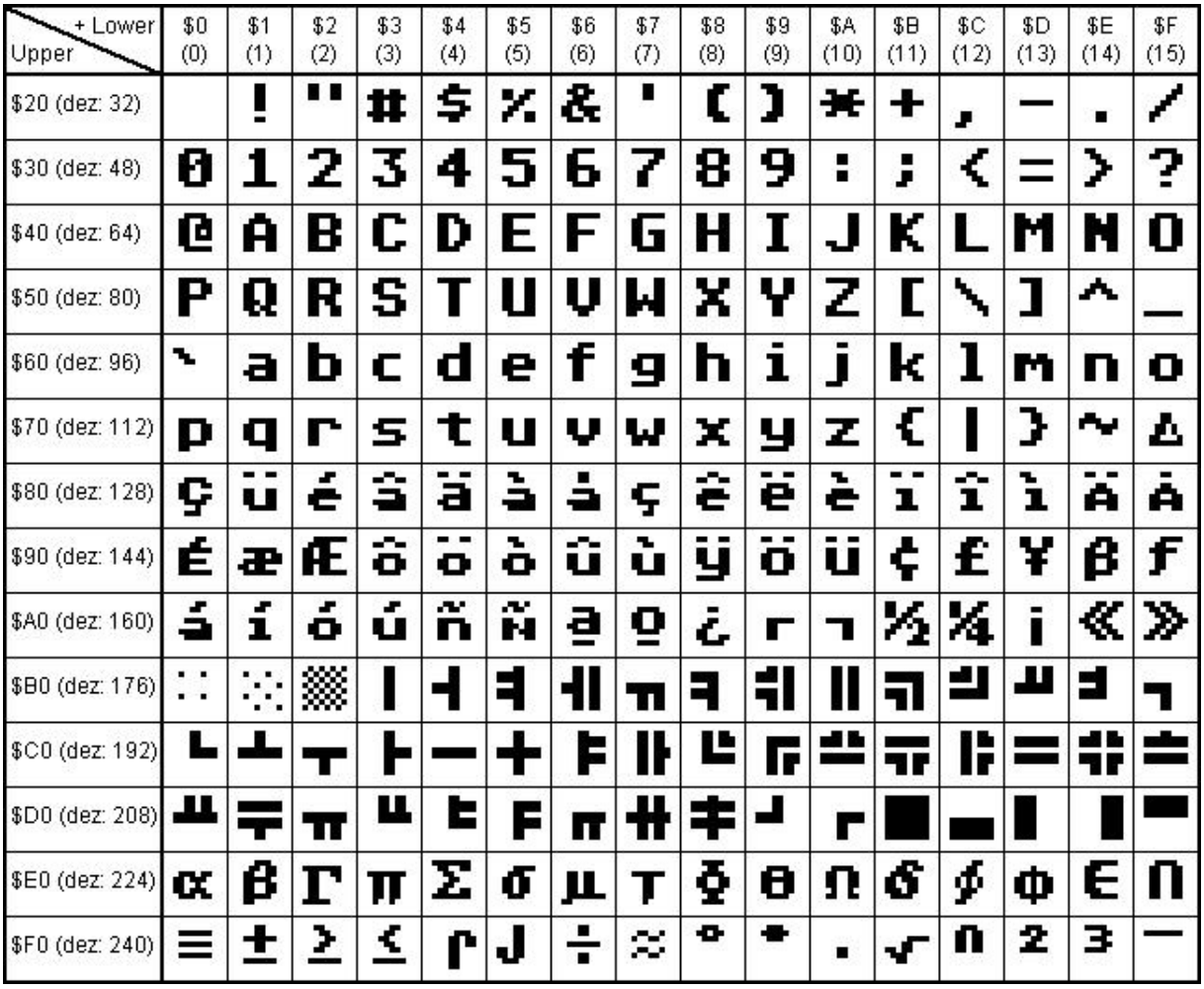
4.2 Font 5x6


+ Lower
Upper
$0 (0)
$1 (1)
$2 (2)
$3 (3)
$4 (4)
$5 (5)
$6 (6)
$7 (7)
$8 (8)
$9 (9)
$A (10)
$B (11)
$C (12)
$D (13)
$E (14)
$F (15)
$20 (dez: 32)
$30 (dez: 48)
$40 (dez: 64)
$50 (dez: 80)
$60 (dez: 96)
$70 (dez: 112)
$80 (dez: 128)
$90 (dez: 144)

4.3 Font 6x8


+ Lower
Upper
$0 (0)
$1 (1)
$2 (2)
$3 (3)
$4 (4)
$5 (5)
$6 (6)
$7 (7)
$8 (8)
$9 (9)
$A (10)
$B (11)
$C (12)
$D (13)
$E (14)
$F (15)
$20 (dez: 32)
$30 (dez: 48)
$40 (dez: 64)
$50 (dez: 80)
$60 (dez: 96)
$70 (dez: 112)
$80 (dez: 128)
$90 (dez: 144)
$A0 (dez: 160)
$B0 (dez: 176)
$C0 (dez: 192)
$D0 (dez: 208)
$E0 (dez: 224)
$F0 (dez: 240)

4.4 Font 8x8


+ Lower
Upper
$0 (0)
$1 (1)
$2 (2)
$3 (3)
$4 (4)
$5 (5)
$6 (6)
$7 (7)
$8 (8)
$9 (9)
$A (10)
$B (11)
$C (12)
$D (13)
$E (14)
$F (15)
$20 (dez: 32)
$30 (dez: 48)
$40 (dez: 64)
$50 (dez: 80)
$60 (dez: 96)
$70 (dez: 112)
$80 (dez: 128)
$90 (dez: 144)
$A0 (dez: 160)
$B0 (dez: 176)
$C0 (dez: 192)
$D0 (dez: 208)
$E0 (dez: 224)
$F0 (dez: 240)

4.5 Font 7x12

Upper \ + Lower	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	⌂
$80 (dez: 128)	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	ß	ƒ
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
$B0 (dez: 176)	▒	▒	▒	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
$C0 (dez: 192)	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
$D0 (dez: 208)	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▀	▄	▌	▐	■
$E0 (dez: 224)	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∅	φ	ε	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	¯

4.6 Font 8x16

Upper \ + Lower	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	△
$80 (dez: 128)	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	ö	Ü	¢	£	¥	ß	ƒ
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
$B0 (dez: 176)	⁝	⁝	▒	|	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
$C0 (dez: 192)	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
$D0 (dez: 208)	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▉	▄	▌	▐	▀
$E0 (dez: 224)	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	ϕ	φ	Є	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	•	∙	√	n	2	3	¯

4.7 Font 16x32 numbers

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)											×	+	,	−	.	÷
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:					

4.8 Font 48x64 numbers

$30 (48)	$31 (49)	$32 (50)	$33 (51)	$34 (52)	$35 (53)	$36 (54)	$37 (55)	$38 (56)	$39 (57)
0	1	2	3	4	5	6	7	8	9

4.9 Font 6x8 cyrillic

Upper \ + Lower	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$40 (dez: 64)	Ж	Ф	И	С	В	У	А	П	Р	Ш	О	Л	Д	Ь	Т	Щ
$50 (dez: 80)	З	Й	К	Ы	Е	Г	М	Ц	Ч	Н	Я	Х	Э	Ъ	Б	Ю
$60 (dez: 96)	ж	ф	и	с	в	у	а	п	р	ш	о	л	д	ь	т	щ
$70 (dez: 112)	з	й	к	ы	е	г	м	ц	ч	н	я	х	э	ъ	б	ю
$80 (dez: 128)	Ё															

4.10 Font 8x16 cyrillic

Upper \ + Lower	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$40 (dez: 64)	Ж	Ф	И	С	В	У	А	П	Р	Ш	О	Л	Д	Ь	Т	Щ
$50 (dez: 80)	З	Й	К	Ы	Е	Г	М	Ц	Ч	Н	Я	Х	Э	ъ	б	Ю
$60 (dez: 96)	ж	ф	и	с	в	у	а	п	р	ш	о	л	д	ь	т	щ
$70 (dez: 112)	з	й	к	ы	е	г	м	ц	ч	н	я	х	э	ъ	б	ю
$80 (dez: 128)	Ё															

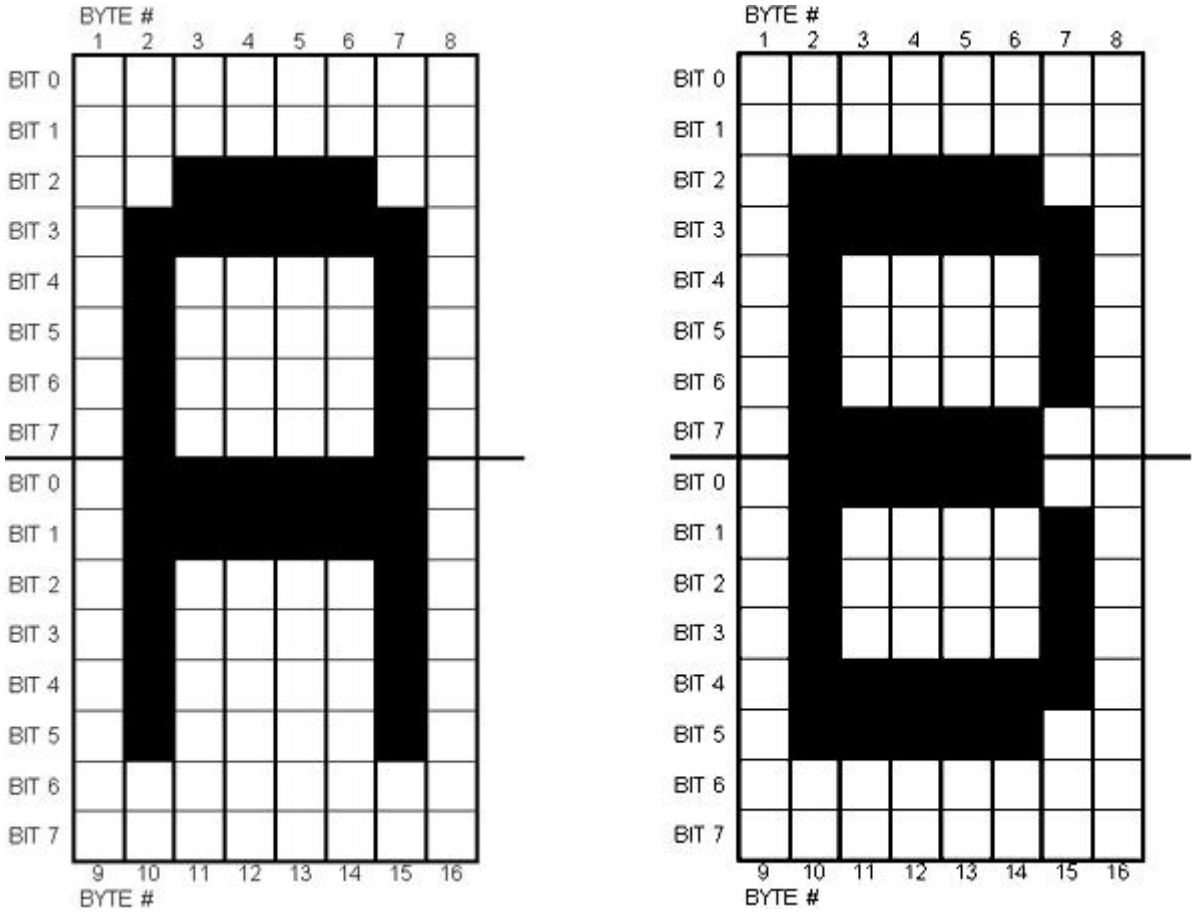
5 Font format *.FV

Font format *.FV for e.g. ST7565, UC1701, UC1610, SED1520 and KS0108 controller.

compatible displays from ELECTRONIC ASSEMBLY http://www.lcd-module.de:
EA DOGS102-6, EA DOGM132-5, EA DOGL128-6, EA DOGL128-6, EA DOGXL160-7,
EA DIP122-5, EA DIP128-6, EA DIP180-5

Each file starts with a 8 byte header, after that font data will follow.
One Byte represents 8 pixels in vertical orientation LSB=TOP and MSB=BOTTOM.

Example: 8x16 font, that defines character 'A' and 'B' only.



The result (40 Byte) in HEX:
Header (8 Byte):

```
  46   ; 'F' first 2 bytes are always FV
  56   ; 'V' for FONT VERTICAL
  41   ; first character code 'A'
  42   ; last  character code 'B'
  08   ; Width  in dots  8
  10   ; Height in dots 16
  02   ; Height in bytes 2
  10   ; Bytes needed for each character 16
         if 0 (for big fonts) calculate WidthInDots * HeightInBytes
```

Character 'A' (16 Byte):

```
  00 F8 0C 0C 0C 0C F8 00   00 3F 03 03 03 03 3F 00
```

Character 'B' (16 Byte):

```
  00 FC 8C 8C 8C 8C 78 00   00 3F 31 31 31 31 1E 00
```

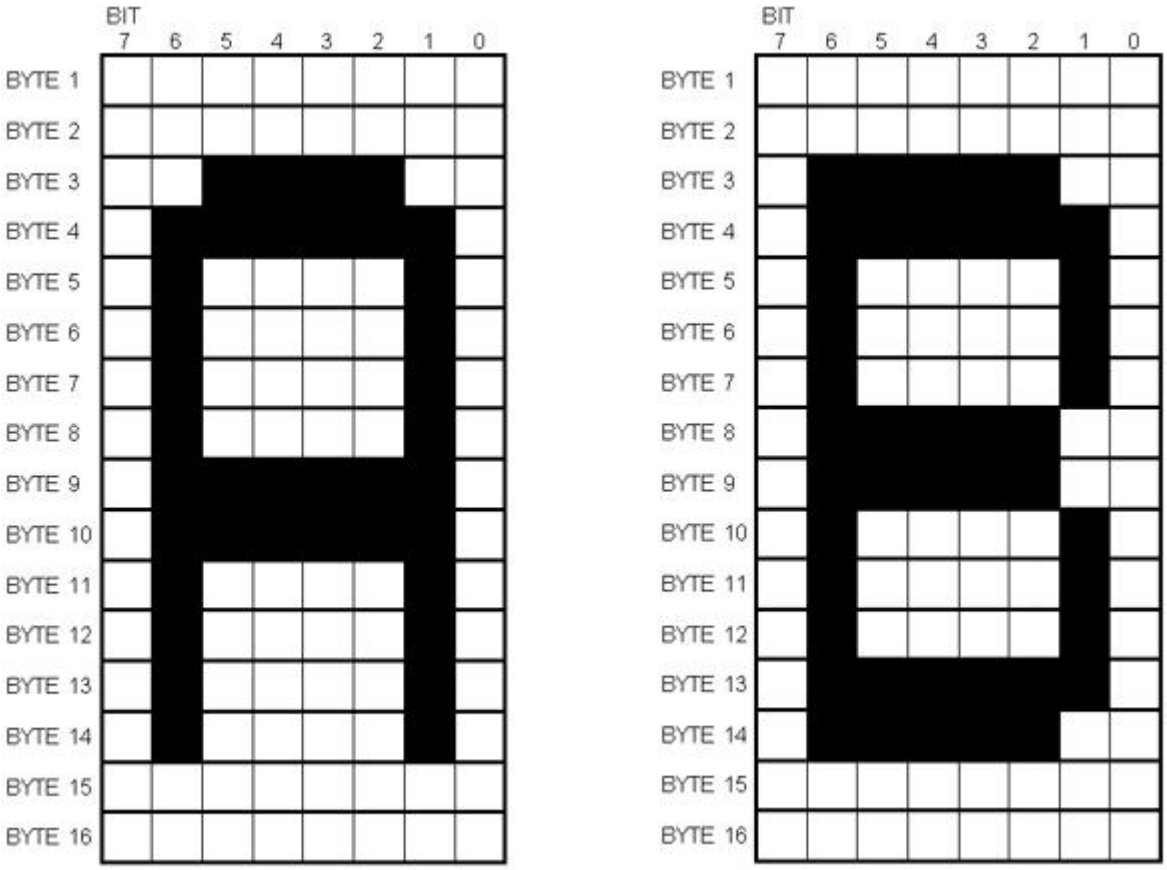
6 Font format *.FH

Font format *.FH for e.g. T6963 and S1D13700 controller.

compatible display from ELECTRONIC ASSEMBLY http://www.lcd-module.de:
EA DIP240-7, EA W240-7, EA W320-8

Each file starts with a 8 byte header, after that font data will follow.
One Byte represents 8 pixels in horizontal orientation MSB=LEFT and LSB=RIGHT.

Example: 8x16 font, that defines character 'A' and 'B' only.



The result (40 Byte) in HEX:
Header (8 Byte):

```
  46   ; 'F' first 2 bytes are always FH
  48   ; 'H' for FONT HORIZONTAL
  41   ; first character code 'A'
  42   ; last  character code 'B'
  08   ; Width  in dots  8
  10   ; Height in dots 16
  01   ; Width  in bytes 1
  10   ; Bytes needed for each character 16
         if 0 (for big fonts) calculate WidthInBytes * HeightInDots
```

Character 'A' (16 Byte):

```
  00 00 3C 7E 42 42 42 42 7E 7E 42 42 42 42 00 00
```

Character 'B' (16 Byte):

```
  00 00 7C 7E 42 42 42 7C 7C 42 42 42 7E 7C 00 00
```