

LCD1_4BIT

Blank Page

Index

LCD1_4BIT	1
Index	3
LCD1_4BIT.TDD	4
LCD Panel	4
Type List	6
Connect LCD Panel	8
User-Function-Codes (LCD)	10
Control characters of the LC-display	11
ESC-Commands LC-Display	12
Position cursor: ESC A	13
Activate special character set: ESC S	15
Load special character set: ESC L	16
Reset special character set: ESC R	18
Menu on the LCD Panel: ESC M	20
Define cursor: ESC c	22
LCD Panel - Special character sets	23
Pre-defined special character sets	25
80h 81h 82h 83h	25
Documentation History	33

LCD1_4BIT.TDD

LCD Panel

Further information on LCD1_4BIT.TDD, LCD panel:

- Type list
- Connect LCD Panel
- User-Function-Codes (LCD)
- Control characters of the LC-display
- ESC-Commands LC-display:
- Position cursor: ESC A
- Activate special character set: ESC S
- Load special character set: ESC L
- Reset special character set: ESC R
- Menu on the LCD Panel: ESC M
- Define cursor: ESC c
- LCD Panel - Special character sets
- Pre-defined special character sets

File name: LCD1_4BIT.TDD

INSTALL DEVICE #D, "LCD1_4BIT.TDD" [, LCD-Type, P2, ..., P10]

D is a constant, a variable or an expression of the data type BYTE, WORD, LONG in the range from 0 to 63 and stands for the device number of the driver.

LCD-Type specifies which type of LCD module is connected. In order to use the codes in the LCD type column of the following table, include the file UFUNCn.INC at the start of your program. The later the UFUNC-file version, the higher the n value in UFUNCn.INC.

P2...P10 are further parameters which modify the standard pin configuration of the LCD panel and the 'beep' audio output. These parameters are described at the end of LCD1_4BIT driver description section. Specification of a different LCD panel type alone, usually does not require the use of these parameters.

The LCD1_4BIT.TDD device drivers controls nearly the same LCD controllers as LCD1.TDD but only with a 4 bit data bus, so it is possible to use the free pins for other purposes. It is possible to use the lower or higher nibble of any Port as data bus. The 2 control lines can be placed everywhere. This driver contains no keyboard, because a keyboard needs the complete 8-bit data bus.

The device driver LCD1_4BIT.TDD assumes that the LCD panel is connected in the manner described in this chapter. You should always use the standard configuration unless special circuit requirements need otherwise. Please remember that changes to the connection configuration can easily lead to problems, which are often not immediately found.

All parameters are bytes and can remain unchanged by specifying 0 or 0EEH (=238) as a byte Values. For example, if you only wish to modify the logic address for sound (P8) enter the Value '0' for parameters P1 to P6 and '0EEH' as the Value for P7.

	Leave unchanged	Description of the parameter
P1	0	LCD-Display-Type (see extra table)
P2	0	Logic BUS address for LC-display and keyboard
P3	0	Bit-MASK for nibble for BUS for LC display 0FOH: higher nibble (Pin 4...7) (default) 00FH: lower nibble (Pin 0...3)
P4	0	Bit-MASK for LC-display signal 'E'
P5	0	Logic address LC-Display signal 'E'
P6	0	Bit-MASK for LC-Display signal 'RS'
P7	0	Logic address LC-Display signal 'RS'
P8	0EEH	Bit-Mask for sound (key click, 'beep')
P9	0	Logic address for sound (key click, 'beep')
P10	0EEH	0: Tone generated if Bit=0 0FFH: Tone generated if Bit=1

Note: if the parameter for bit mask is 0 then the pin is free, accessible for other device drivers or BASIC.

The specification of a different LCD panel type is normally sufficient for development work. When installing the LCD1_4BIT device driver, parameters can be used to configure the port to control external components.

Type List

Integrate the newest include file 'UFUNCTION.INC' to use the identifier for the LCD type. Parameter 'LCD-Type':

No	LCD-Type	Lines x Columns	Refresh-Mode
1	LCD1_1_8	1 x 8	1:8
2	LCD1_1_12	1 x 12	1:8
3	LCD1_1_16	1 x 16	1:8
4	LCD1_1_20	1 x 20	1:8
5	LCD1_1_40	1 x 40	1:8
6	LCD1_2_8	2 x 8	1:8
7	LCD1_2_12	2 x 12	1:8
8	LCD1_2_16	2 x 16	1:8
9	LCD1_2_20	2 x 20	1:8
10	LCD1_2_40	2 x 40	1:8
11	LCD1_4_20	4 x 20	1:8
12	LCD2_2_8	2 x 8	1:16
13	LCD2_2_12	2 x 12	1:16
14	LCD2_2_16	2 x 16 (default)	1:16
15	LCD2_2_20	2 x 20	1:16
16	LCD2_2_40	2 x 40	1:16
17	LCD2_4_20	4 x 20	1:16
18	LCD2_2_24	2 x 24	1:16
19	LCD2_4_16	4 x 16	1:16
20	LCDNJU_4_24	4 x 24	
21	LCDNJU_1_16	1 x 16	
22	LCDNJU_2_16	2 x 16	
23	LCDNJU_2_24	2 x 24	
24	LCDNJU_2_32	2 x 32	
25	LCDNJU_2_40	2 x 40	
26	LCDNJU_4_8	4 x 8	
27	LCDNJU_4_16	4 x 16	
28	LCDNJU_4_20	4 x 20	

LCD1_4BIT.TDD

If no LCD type is specified, Type 14 is the default type used.

Examples:

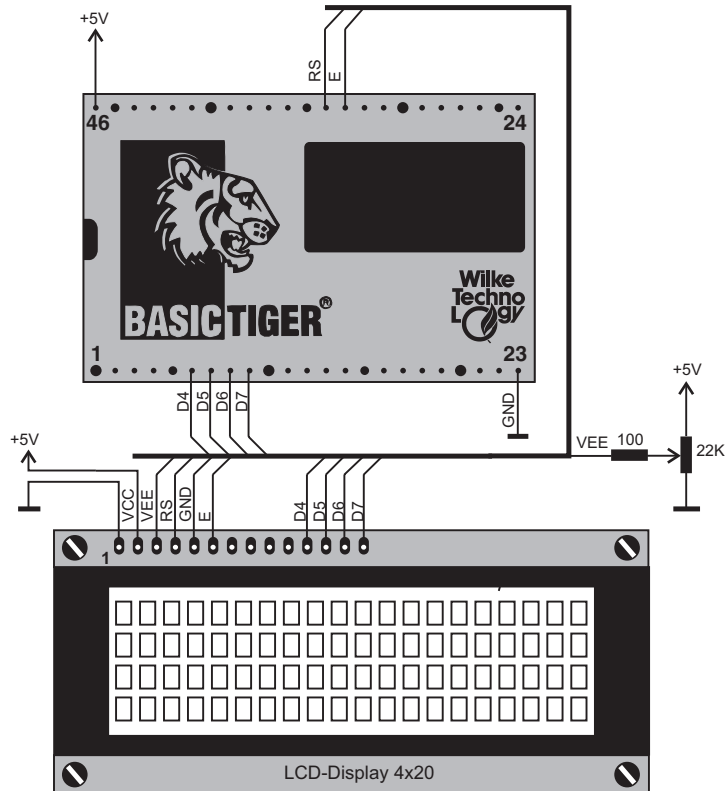
Change the LCD panel type to a model with 4 lines of 20 characters ('LCD1_2_40' is defined in the include file 'DEFINE_A.INC') used on the Plug'n'Play Lab:

```
INSTALL_DEVICE "LCD1_4BIT.TDD", LCD2_4_20
```

Connect LCD Panel

An LCD panel needs the data bus and two further I/O lines on the BASIC Tiger® module: 'Enable-LCD' (E) and 'Register select' (RS). Standard pin configuration:

LCD-Pin function	Pin code	Module A Pin No.	Tiny-Tiger Pin No.
D4→D7	L64 to L67	6 to 9	5 to 8
E (Enable)	L36	33	32
RS (Register Select)	L37	34	33



Since LCD panels have their own controller, they can crash or become unstable independently of their controlling environment. This can occur due to static discharges in the vicinity of the LCD panel or data cable crosstalk. The latter occurs particularly in very long cables. Experience has shown that an appropriate ground

LCD1_4BIT.TDD

connection is critical when using long data cables. If the display does crash, one solution is to reset the LCD panel from the controlling module sending the UFC command UFCO_LCD_RESET to the LCD1_4BIT device driver.

User-Function-Codes (LCD)

User Function Codes of LCD1_4BIT.TDD to request parameters (GET):

No	Symbol Prefix UFCI_	Description
33	UFCI_OBU_FILL	Number of bytes in output buffer (Byte)
34	UFCI_OBU_FREE	Free space in output buffer (Byte)
35	UFCI_OBU_VOL	Size of output buffer (Byte)
65	UFCI_LAST_ERRC	Last error code
99	UFCI_DEV_VERS	Driver version

User Function Codes of LCD1_4BIT.TDD to set parameters (PUT):

No	Symbol Prefix: UFCO_	Description
1	UFCO_IBU_ERASE	Delete keyboard input buffer
33	UFCO_OBU_ERASE	Delete LCD output buffer
176	UFCO_LCD_RESET	reset LCD

Control characters of the LC-display

Control characters are written directly to the LCD device with no Esc and Eos.

CLR	<01>	deletes the LCD screen
HOME	<02>	sets the cursor in the top left corner
FS	<05>	Cursor 1 position to the right
BS	<08>	Cursor 1 position to the left
LF, DO	<0Ah>	Cursor 1 position down
UP	<0Bh>	Cursor 1 position up
FF	<0Ch>	Form Feed
CR	<0Dh>	Carriage Return

```
PRINT #LCD, "<1>";
```

Deletes the LCD screen and moves the cursor to its 'home' position (X=0, Y=0). This can be directly followed by text command:

```
PRINT #LCD, "<1>Hello World"
```

ESC-Commands LC-Display

Overview:

ESC, A, x, y, EoS	Absolute cursor addressing
ESC, S, n, EoS	Activate special character set n n=0→15
ESC, L, n, Data, EoS	Load special character set n n=0→15 64 Bytes Data
ESC, R, n, EoS	Reset special character set n if n>15: reset all
ESC, M, n, EoS Tz STRING	Menu selection n n is the index in the menu Tz is the break character
ESC, c, n, EoS	Define cursor n=0: Cursor off n=1: Cursor on n>1: Cursor on + flash

ESC sequences must always be output in a PRINT or PUT instruction. If the line length allows, a number of ESC sequences can be output in one instruction.

Position cursor: ESC A

```
PRINT #D, "<1Bh>A"; CHR$(x); CHR$(y); "<F0h>";
```

This positions the display cursor using absolute reference values.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the driver.

x x-coordinate (column), at which the cursor is to be positioned.

y y-coordinate (line), at which the cursor is to be positioned.

Column and row numbers start at 0. The possible value range depends on the LCD type used. Entered values for x and y, which are too large are set 0.

A 4x20 LCD panel uses the following numbering system:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																				
1																				
2																				
3																				

Program example:

```

-----
'Name: GOTOXY.TIG
-----
'An asterisk ('*') moves over the LC-display from
'left to right and back.
-----
TASK MAIN                                'begin task MAIN
  BYTE X                                'var of type BYTE
  INSTALL DEVICE #1, "LCD1 4BIT.TDD"
  PRINT #1, "<1Bh>c<0><F0h>";           'disable cursor
  PRINT #1, "<1>";                       'clear screen
  LOOP 9999999                          'many loops
    FOR X = 0 TO 19                      'loop over all columns
      CALL STARXY (X,1)                  '"*" at line 1, column X
    NEXT                                'next column
    FOR X = 18 TO 1 STEP -1              'loop over all columns
      CALL STARXY (X,1)                  '"*" at line 1, column X
    NEXT                                'next column
  ENDOLOOP
END                                      'end task MAIN
-----
'Sub: cursor at position (x, y), print "*" and delete after 200ms
-----
SUB STARXY (BYTE X,Y)                   'begin subroutine STARXY
  PRINT #1, "<1Bh>A"; CHR$(X);&          'output "*" at position
      CHR$(Y); "<F0h>*";                 'line X, column Y
  WAIT DURATION 200                     'wait 200 ms
  PRINT #1, "<1Bh>A"; CHR$(X);&          'output "*" at position
      CHR$(Y); "<F0h> ";                 'line X, column Y
END                                      'end subroutine STARXY

```

Activate special character set: ESC S

```
PRINT #D, "<1BH>S"; CHR$(n); "<OF0H>";
```

Activates a special character set.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

n Number of the special character set (between 0 and 15).

There are 16 special character sets, of which only one can ever be active at one time. This determines the appearance of the 8 special characters available to be displayed. This means that all visible special characters must always be from the same special character set.

Characters from other special character sets, which are currently on display, are immediately converted to the appropriate new character set.

Program example:

```
'-----
'Name: SELECT_FONT.TIG
'-----
'repeats displaying all 16 special character sets one-by-one
'for 2 seconds each.
'-----
TASK MAIN                                'begin task MAIN
  BYTE X                                'var of type BYTE
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"
  PRINT #1, "<1BH>c<0><F0h>";              'set cursor off
  PRINT #1, "<1>";                          'clear screen
  FOR X=0 TO 7                            'loop over all chars
    PRINT #1, CHR$(80h + X); "-";          'output of char
  NEXT                                     'next char
  LOOP 9999999                            'many loops
  FOR X=0 TO 15                            'loop over all fonts
    PRINT #1, "<1BH>S";CHR$(X); "<F0h>"; 'activate font no. X
    WAIT DURATION 2000                    'wait 2 sec
  NEXT                                     'next font
ENDLOOP
END                                        'end task MAIN
```

Load special character set: ESC L

```
PRINT #D, "<1BH>L"; CHR$(n); Data record; "<0F0H>";
```

Defines or loads a special character set with configuration data.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

n Number of the special character set (between 0 and 15).

Data record 64 Bytes, which determines the font structure.

This ESC sequence loads your own special character set. Individual special character sets are required if a custom symbol has to be shown, or if the standard character set does not provide certain foreign language characters, i.e. ß or Σ. The special character set is transferred to the LCD and stored there. The character set has to be re-loaded after every power up.

Program example:

```

-----
'Name: LOAD_FNT.TIG
-----
'All characters of special character set 7 are shown on the
'display. In a loop this special character set is loaded with
'user defined characters and then set back to the standard
'special character set every 2 seconds. On the LC-display you
'can view these changes constantly.
-----

TASK MAIN                                'begin task MAIN
  STRING CSET$                            'var of type STRING
  BYTE X                                  'var of type BYTE
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"

  CSET$ = "&                                'data for USER-font
04 04 04 04 00 00 00 00&                'data char 0
01 02 02 04 00 00 00 00&                'data char 1
00 00 00 07 00 00 00 00&                'data char 2
00 00 00 04 02 02 01 00&                'data char 3
00 00 00 04 04 04 04 00&                'data char 4
00 00 00 04 08 08 10 00&                'data char 5
00 00 00 1C 00 00 00 00&                'data char 6
10 08 08 04 00 00 00 00"%                'data char 7

  PRINT #1, "<1Bh>c<0><F0h>";                'set cursor off
  PRINT #1, "<01h>";                        'clear screen
  FOR X=0 TO 7
    PRINT #1, CHR$(80h + X); " ";          'loops over all chars
  NEXT                                     'show chars of special font
  PRINT #1, "<1Bh>s<7><F0h>";                'next char
  LOOP 9999999                            'set special char set 7 on
  WAIT DURATION 2000                       'many loops
  PRINT #1, "<1Bh>L<7>";CSET$;"<F0h>";      'wait 2 sec.
  WAIT DURATION 2000                       'special font USER-defined
  PRINT #1, "<1Bh>R<7><F0h>";                'wait 2 sec.
  ENDLOOP                                  '"standard" special font
END                                        'end task MAIN

```

See also: Esc command S (activate character set) and Esc command R (reset character set).

Reset special character set: ESC R

```
PRINT #D, "1BH>R"; CHR$(n); "0F0H>";
```

Removes a previously self-defined special character set and resets this to the default special character set as printed on page.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

n Number of the special character set (between 0 and 15)

If a value greater than the maximum permitted font number, 15, is specified for n, all special character sets are reset to their defaults.

Program example:

```

-----
'Name: LOAD_FNT.TIG
-----
'All characters of special character set 7 are shown on the
'display. In a loop this special character set is loaded with
'user defined characters and then set back to the standard
'special character set every 2 seconds. On the LC-display you
'can view these changes constantly.
-----

TASK MAIN                                'begin task MAIN
  STRING CSET$                           'var of type STRING
  BYTE  X                                'var of type BYTE
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"
  CSET$ = "&                             'data for USER-font
04 04 04 04 00 00 00 00&               'data char 0
01 02 02 04 00 00 00 00&               'data char 1
00 00 00 07 00 00 00 00&               'data char 2
00 00 00 04 02 02 01 00&               'data char 3
00 00 00 04 04 04 04 00&               'data char 4
00 00 00 04 08 08 10 00&               'data char 5
00 00 00 1C 00 00 00 00&               'data char 6
10 08 08 04 00 00 00 00"%               'data char 7

  PRINT #1, "<1Bh>c<0><F0h>";           'set cursor off
  PRINT #1, "<01h>";                     'clear screen
  FOR X=0 TO 7                           'loops over all chars
    PRINT #1, CHR$(80h + X); " ";         'show chars of special font
  NEXT                                    'next char
  PRINT #1, "<1Bh>S<7><F0h>";           'set special char set 7 on
  LOOP 9999999                           'many loops
    WAIT DURATION 2000                     'wait 2 sec.
    PRINT #1, "<1Bh>L<7>";CSET$;"<F0h>"; 'special font USER-defined
    WAIT DURATION 2000                     'wait 2 sec.
    PRINT #1, "<1Bh>R<7><F0h>";         '"standard" special font
  ENDLLOOP
END                                        'end task MAIN

```

See also: Esc command S (activate character set) and Esc command L (load character set).

Menu on the LCD Panel: ESC M

PRINT #D, "<1BH>M"; CHR\$(n); "<0FOH>"; Menu\$;

The command 'M' allows a portion of a string to be displayed. The chosen portion (string element) is indexed by the byte after 'M'. This facility can be used to create a user menu selection.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

n Index of the current menu item to be shown

Menu\$ String containing the individual menu elements to be shown. This string initially contains the break character (in the example ":", followed by the menu elements. The individual elements are separated by the break characters. The first element's index value is 0, the second element is 1, etc. The end of the selection string is marked by a double break character.

WARNING: If the index is larger than the number of elements within the string, empty elements will be output! In the example program, there are 4 entries in the string (Index 0 to 3). With an index of n=5 an empty string would be displayed after "Your choice: ".

Program example:

```

'-----
'Name: MENU.TIG
'-----
'With keys <Up> and <Down> the index for the menu item to select
'is increased resp. decreased and the new menu item selected is
'shown on the LC-display, with <Return> the selected index is shown
'and the program is ended.
'-----
#include KEYB_PP.INC                                'English keyboard layout

TASK MAIN                                           'begin task MAIN
  STRING KEY$                                       'var of type STRING
  BYTE  M, N, FLAG                                 'vars of type BYTE
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"
  CALL INIT_KEYB (1)                               'set keyboard etc.

  M = 0                                             'menu index = 0
  FLAG = 1                                          'flag = 1 (true)
  WHILE FLAG = 1                                   'while flag = true
    CALL SHOWMENU (M)                              'show menu item M
    FOR N = 0 TO 0 STEP 0                          'endless loop until N=1(GET!)
      RELEASE_TASK                                'release rest of task time
      GET #1, #0, #1, 1, N                        'N=chars in keyboard buffer
    NEXT                                           'end of endless loop
    GET #1, 1, KEY$                                'read from keyboard buffer
    SWITCH KEY$                                    'conditional branch (content)
      CASE _CR:                                    'KEY$ = <Return>:
        FLAG = 0                                  'flag = false
      CASE _UP:                                    'KEY$ = <Up>:
        M = (M - 1) BITAND 3                      'decrease index (min. 0)
      CASE _DO:                                    'KEY$ = <Down>:
        M = (M + 1) BITAND 3                      'increase index (max. 3)
    ENSWITCH                                       'end of conditional branch
  ENDWHILE
  PRINT #1, _CLR;                                  'clear screen
  PRINT #1, "Menu index is: "; M                  'output to LC-display
END                                                'end task MAIN

'-----
'Subroutine: output currently selected menu item to LC-display
'-----
SUB SHOWMENU (BYTE I)                              'begin subroutine SHOWMENU
  PRINT #1, _CLR;                                  'clear screen
  PRINT #1, "your choice: ";                      'output of menu item no. I
  PRINT #1, "<1Bh>M"; CHR$(I); &
  "<F0h>:AUTO :MANUAL :ALARM :STOP :";
END                                                'end subroutine SHOWMENU

```

Define cursor: ESC c

PRINT #D, "<1Bh>c"; CHR\$(n); "<OF0h>";

Defines the appearance of the cursor on the display.

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

n determines the form of the cursor
n=0: Cursor is inactive
n=1: Cursor is active
n>1: Cursor is active and flashing

Program example:

```

'-----
'Name: CURSOR.TIG
'-----
TASK MAIN                                'begin task MAIN
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"      'many loops
  LOOP 9999999                            'loop from 0 to 2
    FOR X = 0 TO 2                        'set cursor-mode
      PRINT #1, "<1Bh>c"; CHR$(X); "<F0h>"; 'switch by index
      SWITCH X
        CASE 0:                          'if x=0:
          PRINT #1, "<1>Cursor off";      'output "cursor off"
        CASE 1:                          'if x=1:
          PRINT #1, "<1>Cursor on";       'output "cursor on"
        CASE 2:                          'if x=2:
          PRINT #1, "<1>Cursor blinks";  'output "cursor blinks"
        ENSWITCH                         'end of switch
      WAIT DURATION 3000                  'wait 3 sec
    NEXT                                'next value
  ENDLONG                                'end task MAIN
END

```

LCD Panel - Special character sets

A number of applications require special characters to be displayed. As an example, German "umlaut" characters are not included in the standard character set, but may be added in as special characters if required.

The LCD device driver supports the programming of the special character sets for the LCD panel with its ESC command sequences.

Each special character set consists of 8 characters of 8 bytes each. Within each of these bytes, the 5 lowest order bits are used for display purposes. Each character set thus consists of 64 bytes, which are declared during a programs definition.

There are 16 special character sets. Only one of these sets can ever be active. Remember, each special set contains 8 unique characters to be used on the display. This means that a display can only contain characters from the standard character set plus characters from the single specified special set, not a combination from the 16 special sets.

Bit	7	6	5	4	3	2	1	0
Byte 7 (0Eh)								
Byte 6 (1Fh)								
Byte 5 (1Bh)								
Byte 4 (11h)								
Byte 3 (1Fh)								
Byte 2 (1Fh)								
Byte 1 (11h)								
Byte 0 (11h)								

The codes for the special characters start at 80h for special character set 0.

Special character set	Codes	Special character set	Codes
0	80h→87h	8	C0h→C7h
1	88h→8Fh	9	C8h→CFh
2	90h→97h	10	D0h→D7h
3	98h→9Fh	11	D8h→DFh
4	A0h→A7h	12	E0h→E7h
5	A8h→AFh	13	E8h→EFh
6	B0h→B7h	14	F0h→F7h
7	B8h→BFh	15	F8h→FFh

If a special character is 'printed' to the panel, the character set from which it was derived is automatically set as the active special character set. Consequently, all characters from other special character sets currently displayed on the panel are automatically converted to the appropriate character from the now active character set.

In the following example, characters from the special character set 0 are to be displayed (äöüÄÖÜ..). After a brief pause, the display changes since characters from a different special character set are printed and only one special character set can be active at a time.

Program example:

```

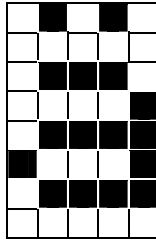
'-----
'  Name: SPECCHR1.TIG
'-----
TASK MAIN                                'begin task MAIN
  INSTALL DEVICE #1, "LCD1_4BIT.TDD"
  PRINT #1, "80818283"%                  'output of special font #0
  WAIT_DURATION 2000                     'wait 2 sec
  PRINT #1, "8C8D8E8F"%                  'output of special font #1
END                                       'end task MAIN

```

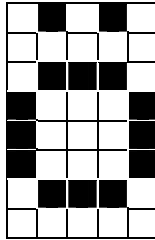

Pre-defined special character sets

All 16 special character sets are prefixed in the LCD1. The following pages show all 16 special character sets of the LCD1 driver.

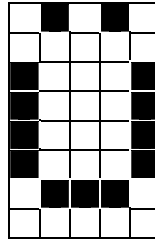
Special character set 0 (80h...87h)



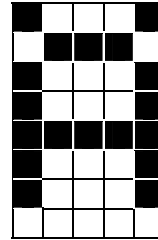
80h



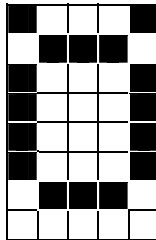
81h



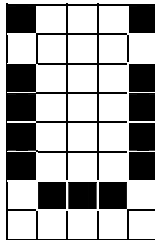
82h



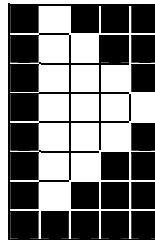
83h



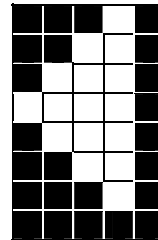
84h



85h

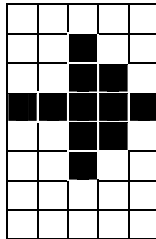


86h

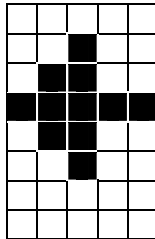


87h

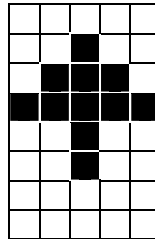
Special character set 1 (88h...8Fh)



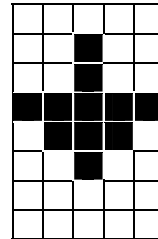
88h



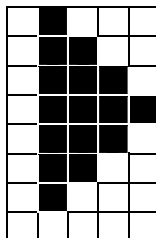
89h



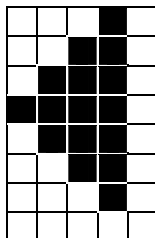
8Ah



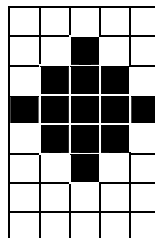
8Bh



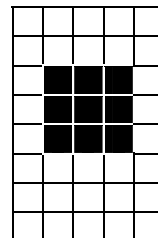
8Ch



8Dh

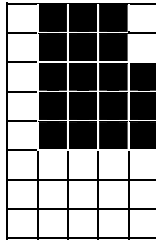


8Eh

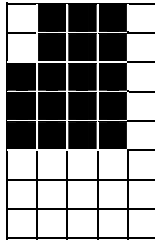


8Fh

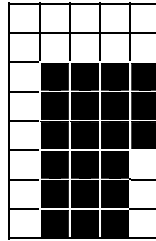
Special character set 2 (90h...97h)



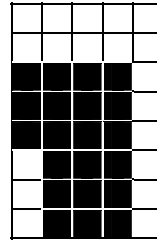
90h



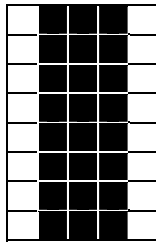
91h



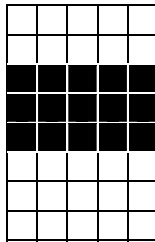
92h



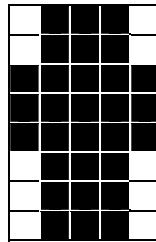
93h



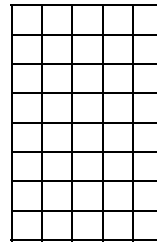
94h



95h

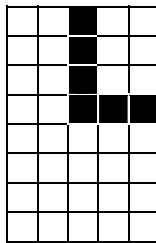


96h

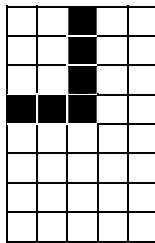


97h

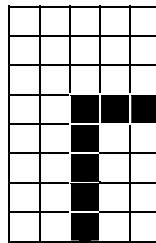
Special character set 3 (98h...9Fh)



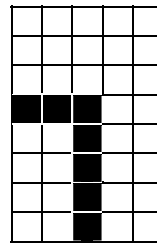
98h



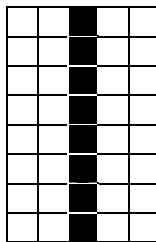
99h



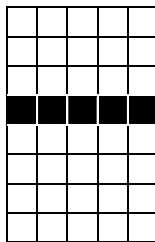
9Ah



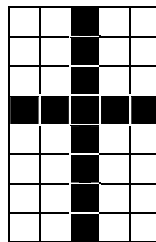
9Bh



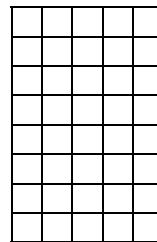
9Ch



9Dh

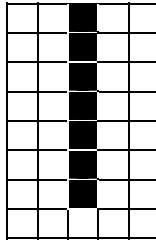


9Eh

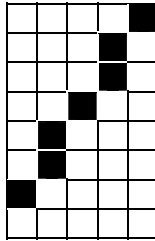


9Fh

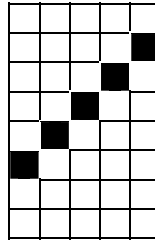
Special character set 4 (A0h...A7h)



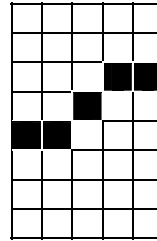
A0h



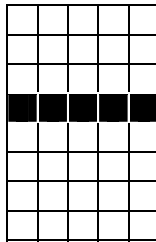
A1h



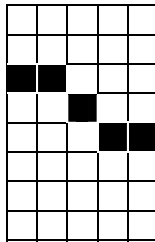
A2h



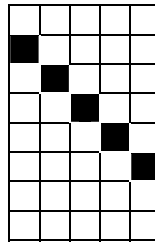
A3h



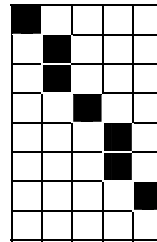
A4h



A5h

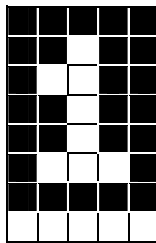


A6h

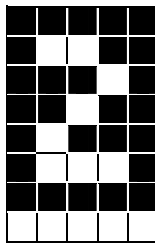


A7h

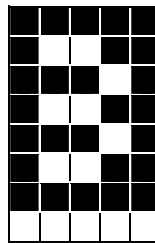
Special character set 5 (A8h...AFh)



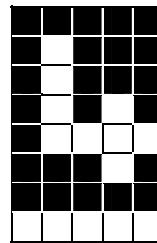
A8h



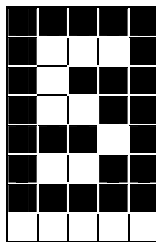
A9h



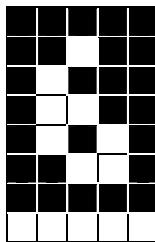
AAh



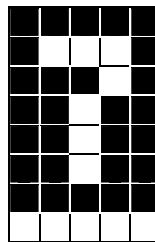
ABh



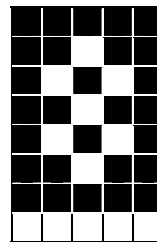
ACh



ADh

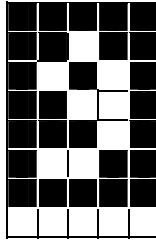


AEh

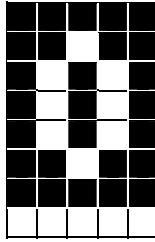


AFh

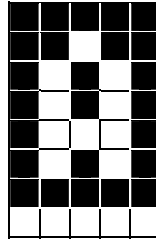
Special character set 6 (Boh...B7h)



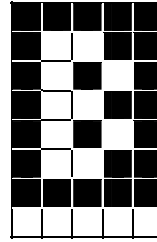
B0h



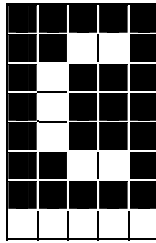
B1h



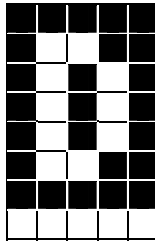
B2h



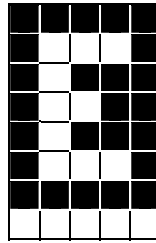
B3h



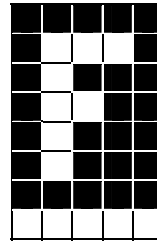
B4h



B5h

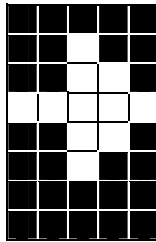


B6h

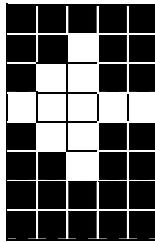


B7h

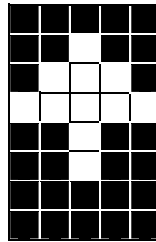
Special character set 7 (B8h...BFh)



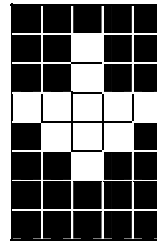
B8h



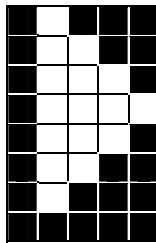
B9h



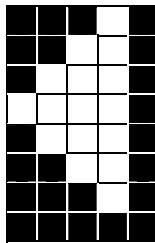
BAh



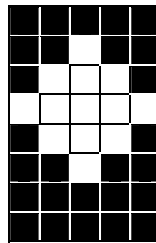
BBh



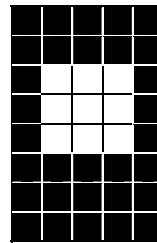
BCh



BDh

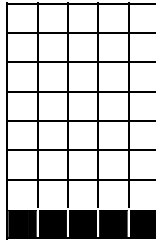


BEh

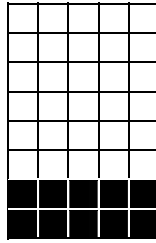


BFh

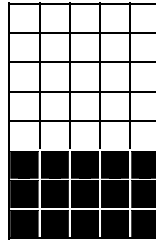
Special character set 8 (C0h...C7h)



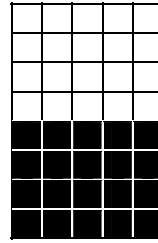
C0h



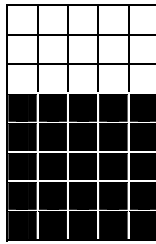
C1h



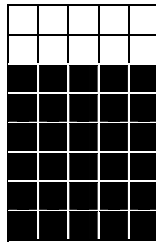
C2h



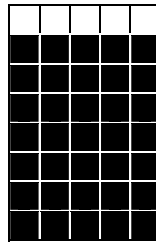
C3h



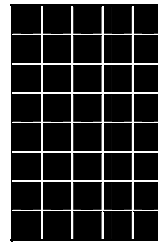
C4h



C5h

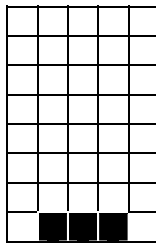


C6h

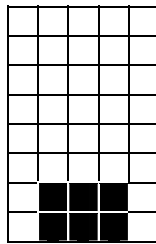


C7h

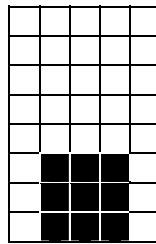
Special character set 9 (C8h...CFh)



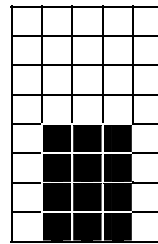
C8h



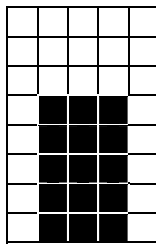
C9h



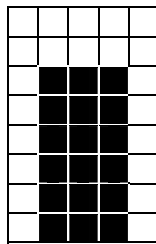
CAh



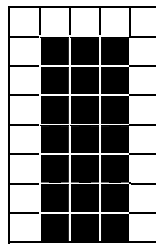
CBh



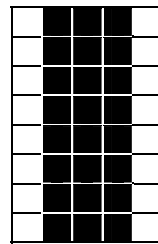
CCh



CDh

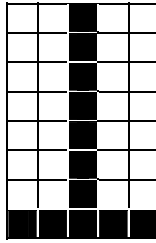


CEh

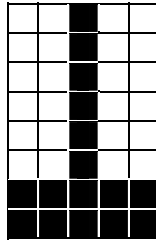


CFh

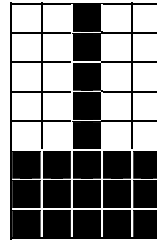
Special character set 10 (D0h...D8h)



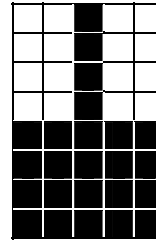
D0H



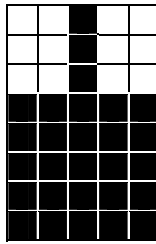
D1H



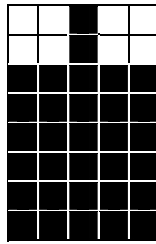
D2H



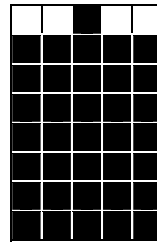
D3H



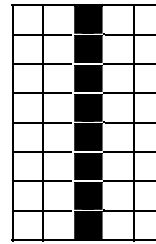
D4h



D5h

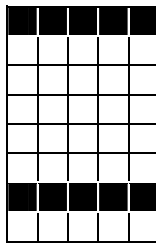


D6h

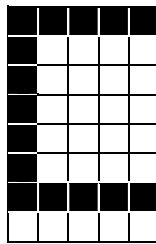


D7h

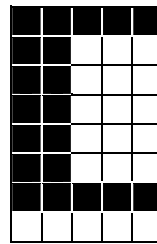
Special character set 11 (D8h...DFh)



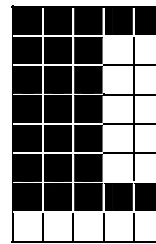
D8h



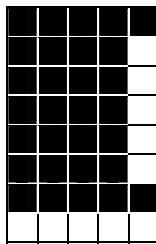
D9h



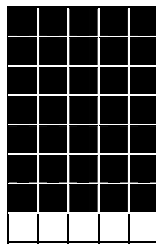
DAh



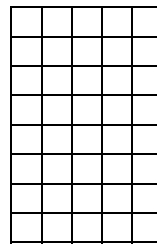
DBh



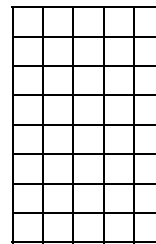
DCh



DDh

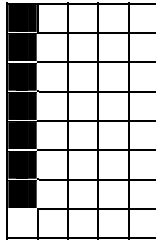


DEh

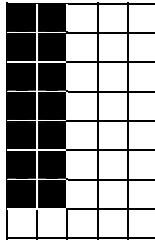


DFh

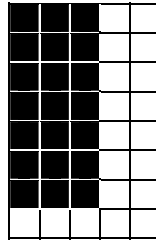
Special character set 12 (E0h...E7h)



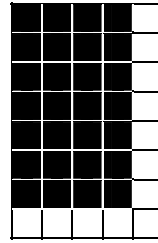
E0h



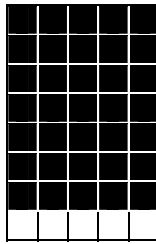
E1h



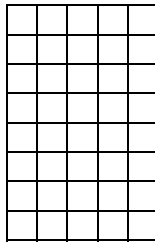
E2h



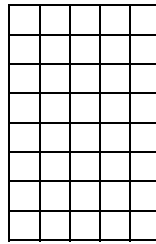
E3h



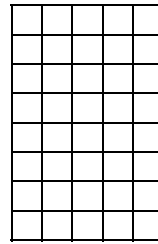
E4h



E5h

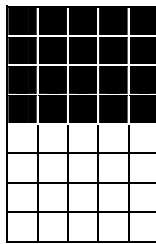


E6h

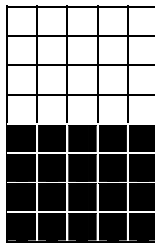


E7h

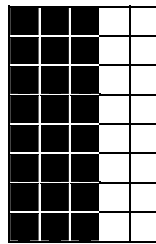
Special character set 13 (E8h...EFh)



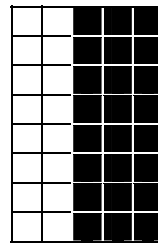
E8h



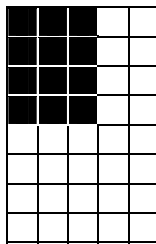
E9h



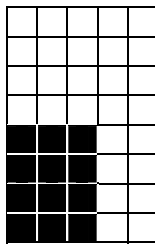
EAh



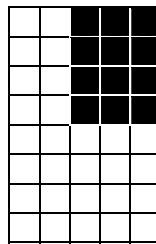
EBh



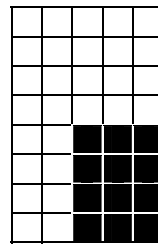
ECh



EDh

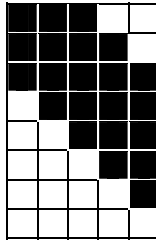


EEh

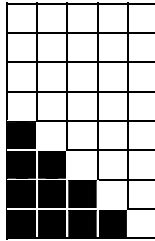


EFh

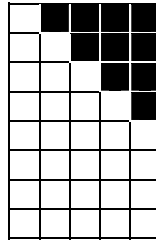
.....



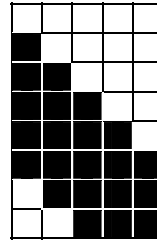
F0h



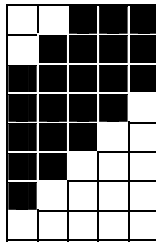
F1h



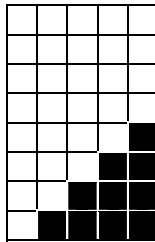
F2h



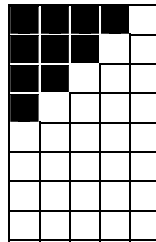
F3h



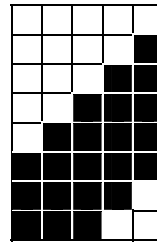
F4h



F5h

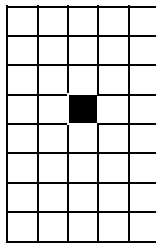


F6h

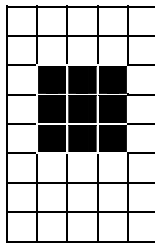


F7h

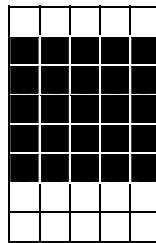
.....



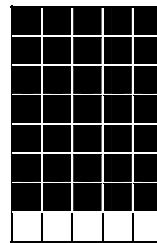
F8h



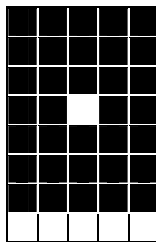
F9h



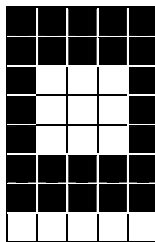
FAh



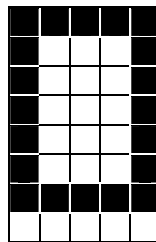
FBh



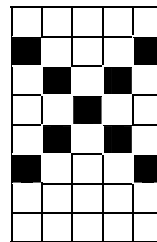
FCh



FDh



FEh



FFh

Documentation History

Version of Documentation	Version of SER1B	Description / Changes
001	1.00	- first version