Application Note

FUJITSU

Connecting standard LCD modules to the MB90670/5 series

History

01th Feb. 97	MM	V1.0	started
28th June 00	TKa	V1.1	New format

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer´s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

   NO LIABILITY FOR CONSEQUENTIAL DAMAGES

   To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

.

In many applications, standard LCD modules are used. Such modules feature a dedicated controller on board to drive a dot matrix with a high segment count. Furthermore this controller provides a parallel interface to establish communication with other devices. In the majority of cases this is a HD44780A00 (or a compatible one).

LCD standard display modules come in many sizes and shapes. Most of them are twisted nematic reflective displays which outputs one or more lines of alphanumerical characters. Each character is formed in a 5 by 8 matrix of dots. The intended applications are computer peripherals, word processors, facsimile machines, telecommunications systems, instruments, point-of-sale terminals, etc.

How LCD's work

As the name implies a liquid crystal is a compound that flows like a liquid but has a crystalline order in the arrangement of their molecules. In a display, these crystals keep their long axes aligned due to intermolecular forces. So by controlling the alignment, one effectively controls the optical properties as well as the ability of the crystals to affect the transmission of light.

In a display, the liquid crystal material is sealed between two glass plates, one bearing transistors to control the electrode (row electrodes) of each cell, the other bearing colour filters and an electrode (column electrodes) to complete the circuit. Polarizers in the front and the rear complete the array which is illuminated from behind.

Example application

Fig. 1 shows how to connect a standard LCD module to the 16-bit Starterkit (MB90678). The device is connected to Port 6 of the kit in a 4-bit mode, so only 7 portpins are used (4 Data-lines and 3 Control-lines). The port output type has to be CMOS push-pull. Every data byte is transmitted in 2 nibbles using a handshake protocol. Four different display types were used to test the application.

To activate the LCD-controller, an initial sequence has to be sent to the module. This procedure sets the device into the correct bus-mode and clears the internal RAM. Typical data transfer between the controller and the LCD then consists of commands (listed in table 2) and data (ASCII-compatible character set, see table 3). Example Code was written in C.

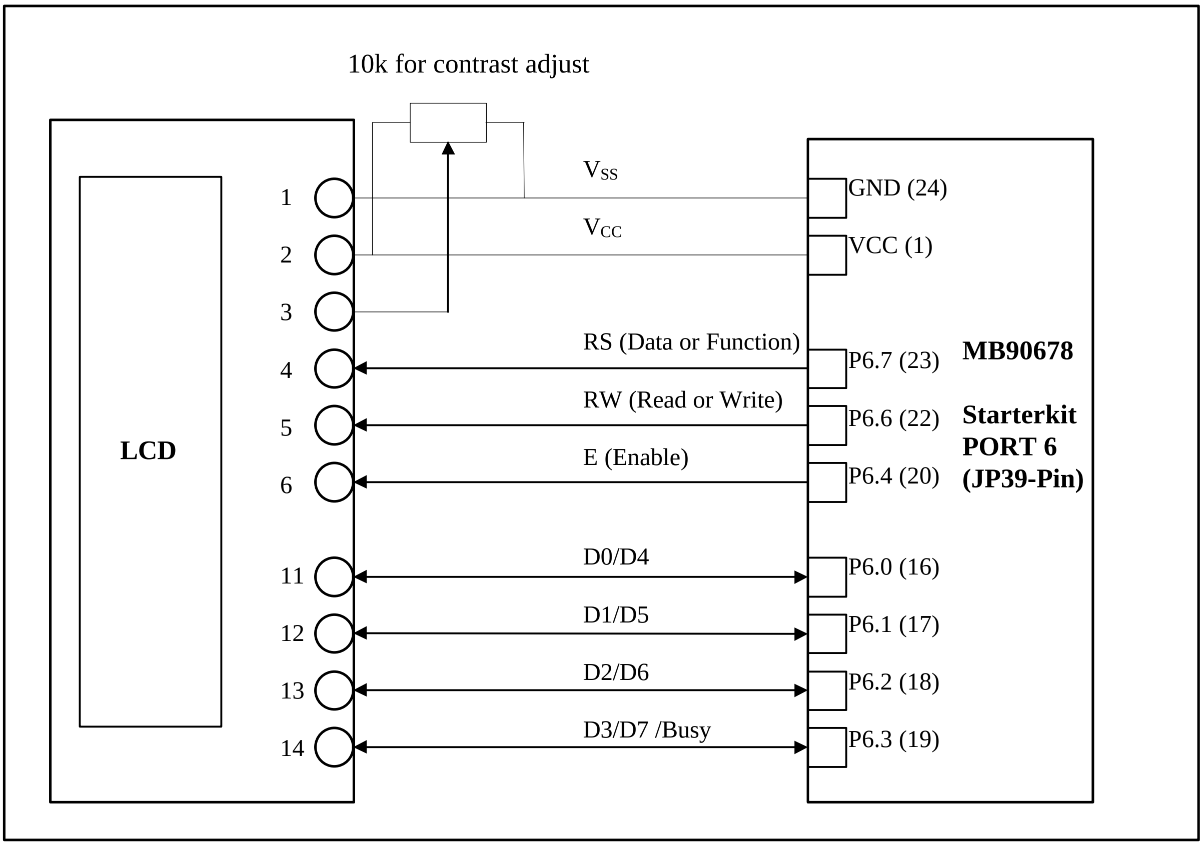


Fig. 1 : Connecting a standard LCD to the 16-bit-Starterkit

The pin assignment shown in Table 1 is the industry standard for character LCD-modules (except those with more than 80 characters). To be sure always check the manufacturers datasheet! The displays can be used in a 8-bit or a 4-bit-interface mode.

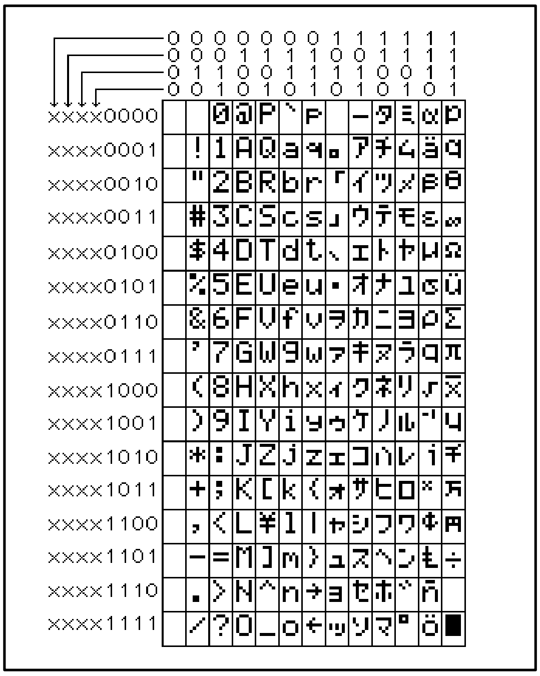
Pin	Symbol	Level	I/O-Function	Function (8-bit mode)	Function (4-bit mode)
1	Vss	-	-	Power supply (GND)	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)	Power supply (+5V)
3	Vee	-	-	Contrast adjust	Contrast adjust
4	RS	0/1	I	0 = Instruction 1 = data	0 = Instruction 1 = data
5	R/W	0/1	I	0 = Write to 1=Read from LCD	0 = Write to 1=Read from LCD
6	E	1 → 0	I	Enable signal	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)	not used
8	DB1	0/1	I/O	Data bus line 1	not used
9	DB2	0/1	I/O	Data bus line 2	not used
10	DB3	0/1	I/O	Data bus line 3	not used
11	DB4	0/1	I/O	Data bus line 4	Data bus line 0 (LSB) and 4
12	DB5	0/1	I/O	Data bus line 5	Data bus line 1 and 5
13	DB6	0/1	I/O	Data bus line 6	Data bus line 2 and 6
14	DB7	0/1	I/O	Data bus line 7 (MSB)	Data bus line 3 and 7 (MSB)

Table 1 : Pin assignment

Instruction	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution time**
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor home	1.64ms
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position	1.64ms
Entry mode set	0	0	0	0	0	0	0	1	ID	S	Sets cursor move direction (I/D),	40µs
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor	40µs
Cursor/ display shift	0	0	0	0	0	1	SL	RL	*	*	Sets cursor-move or display-shift	40µs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number	40µs
Set CGRAM address	0	0	0	1	a	a	a	a	a	a	Sets the CGRAM address.	40µs
Set DDRAM address	0	0	1	a	a	a	a	a	a	a	Sets the DDRAM address.	40µs
Read busy-flag	0	1	BF	a	a	a	a	a	a	a	Reads Busy-flag (BF) and address counter	0µs
Write to CG- or DD-RAM	1	0	d	d	d	d	d	d	d	d	Writes data to CGRAM or DDRAM.	40µs
Read from CG- or DD-RAM	1	1	d	d	d	d	d	d	d	d	Reads data from CGRAM or DDRAM.	40µs

Table 2 : HD44780 instruction set

Remarks:
DDRAM = Display Data RAM. CGRAM = Character Generator RAM. DDRAM address corresponds to cursor position.Address Counter used for both DDRAM and CGRAM. * = Don't care. ** = Based on Fosc = 250khz. a=address. d=data



	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000			0	@	P	`	p		ー	タ	ミ	α	p
xxxx0001		!	1	A	Q	a	q	。	ア	チ	ム	ä	q
xxxx0010		"	2	B	R	b	r	「	イ	ツ	メ	β	θ
xxxx0011		#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
xxxx0100		$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
xxxx0101		%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
xxxx0110		&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111		'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
xxxx1000		(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
xxxx1001		)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010		*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
xxxx1011		+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
xxxx1100		,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
xxxx1101		-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
xxxx1110		.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
xxxx1111		/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█

Table 3 **: Character Set**

```
/*
*********************************************************************
**                                                                 **
**                        DISPLAY-DEMO                             **
**                                                                 **
**      for MB90678 (16bit-Starterkit)                             **
**      can be run with Emulator                                   **
**                                                                 **
**      initializes an LCD-display and displays a message          **
**      LCD connected to Port 6 (default-LCD-Type : LTN111)        **
**                                                                 **
**      Author  : Markus Mierse                                    **
**      Version : 1                                                **
**      Date    : 1.2.97                                           **
**                                                                 **
**      (C) Fujitsu Mikroelektronik GmbH 1997                      **
**                                                                 **
*********************************************************************
*/


#include <sample\extn\mb90675.h>     /* for all Register Names */


/*----------------------- variables --------------------*/


typedef unsigned char BYTE;   /* BYTE definition */

int cursor;                   /* display position */

int i;                        /* general purpose integer variable */

/*---------------------  prototypes --------------------*/


void initdisp(void);
void outb(unsigned char);
void busy(void);
void print(char *Name2);
void printnum(int n);
void wait(int i);

/*---------------------------------------------------------------*/
/*                        MAIN PROGRAM                           */
/*---------------------------------------------------------------*/

void main(void)
{

  initdisp();                 /* initialize display on port 6 */
  print("Hello World !");     /* show a message on the LCD */
  wait(10000);                /* wait a short while */

  for (i=1;i<1000;i++)        /* count up to 1000 */
  {
   print("i=");
   printnum(i);               /* display the number */
   wait(5000);
  }
}

/*--------------------- function bodies ----------------------*/


void initdisp(void)           /* This Routine initializes the LCD on port6 */

{
       DDR6=0x0DF;            /* set Port to Output (P65 not used) */
       PDR6=0;                /* Port 6 Off*/
       PDR6=0x013;            /* Startup sequence */
       PDR6=0x03;
         wait(1000);
       PDR6=0x013;
       PDR6=0x03;
         wait(1000);
       PDR6=0x013;
       PDR6=0x03;
         wait(1000);
       PDR6=0x012;
       PDR6=0x02;
       outb(0x028);           /* Switch to 4-bit mode */
       outb(0x0C);            /* Cursor Off (on=0x0F) */
       outb(0x06);            /* No shift */
       outb(0x03);            /* Cursor home */
       outb(0x01);            /* Display clear */
}

/*--------------------------------------------------------------*/

void outb(unsigned char a)    /* send one byte to the display */
{
        BYTE b;

        cursor++;
        if (cursor == 9)
        {
         PDR6=0x01C;          /* correct position !LNT111-R only!*/
         PDR6=0x00C;
         PDR6=0x010;
         PDR6=0;
         busy();
        }
        b=(a & 0x0F0);        /* shift upper nibble */
        b = b >> 4;           /* to lower nibble */
        if (a & 0x080)  {b=(b | 0x080);};
                              /* but keep Bit 7 */
        b=(b & 135);          /* set other bits to zero  */
        b=(b | 16);           /* set E line  */
        PDR6=b;               /* send to LCD  */
        b=(b & 135);          /* clear E line  */
        PDR6=b;               /* send to LCD  */
        b=(a & 143);          /* take lower nibble  */
        b=(b | 16);           /* set E line */
        PDR6=b;               /* send to LCD  */
        b=(b & 143);          /* clear E line  */
        PDR6=b;               /* send to LCD  */
        busy();               /* wait for busy-line */

}


/*--------------------------------------------------------------*/


void busy(void)               /* This Routine polls the busy-line */
{
        BYTE b;
        PDR6=0;               /* Port 3 Off before reading ! */
        b=1;
        while (b)             /* wait for Busy-line */
        {
         DDR6=0x0D0;          /* set Bus as input to read Bit .3 (Busy) */
         PDR6=0x05F;          /* busy request */
         b=PDR6_PD63;         /* read Port  */
         PDR6_PD64 = 0;
         PDR6_PD64 = 1;       /* toggle E */
         PDR6_PD64 = 0;
         PDR6_PD66 = 0;
         DDR6=0x0DF;          /* reset Port to output */
        }
}


/*--------------------------------------------------------------*/


void print(char *Name2)       /* This Routine displays a String */
{
 unsigned char c;
 BYTE b;
 int i,l;
 outb(1);                     /* Display clear */
 l=strlen(Name2);
 cursor=0;
 for (i=0; i<l; i++)          /* go through string */
  {
   c=(Name2[i]);              /* pick char */
   b=(c | 128);
   outb(b);                   /* and display it */
  }
}
```

```
/*---------------------------------------------------------------*/

void printnum(int n)          /* show integer value on LCD display */
{
   float x;
   int l;
   if (n < 10)                /* only one digit value */
   {
    outb(0x0b0);
    outb(0x0b0);
    outb((n+48) | 128);
   }
   else if (n >= 10 && n<100)  /* two digit value */
   {
    outb(0x0b0);
    outb(((n/10)+48) | 128);
    x = n-(10*(n / 10));
    l = x;
    outb((l+48) | 128);
   }
   else if (n >= 100)         /* show three digits */
   {
    outb(((n/100)+48) | 128);
    x = n-(100*(n / 100));
    n = x;
    outb(((n/10)+48) | 128);
    x = n-(10*(n / 10));
    l = x;
    outb((l+48) | 128);
   }
}


/*---------------------------------------------------------------*/

void wait(int i)
{
  for (; i ; i--);      /* very simple delay loop */
}

/*---------------------------------------------------------------*/
```