

Microcontroller Basics Course (6)

part 6 (final): driving an LCD

By B. Kainka

Data output from the Flash Board usually takes place via the connected terminal, which means the PC. However, for stand-alone use without a terminal, a display is sometimes necessary.

Using an intelligent LC display with its own display controller in combination with a microcontroller board is very convenient. The display controller already contains its own character generator and can be commanded to output ASCII characters without an undue amount of effort.

Nowadays, almost all intelligent LCDs comply with the same standard. Here we will use a standard model with two lines of 16 characters. The connections to the display are listed in **Table 1**.

Table 1. LCD module connections

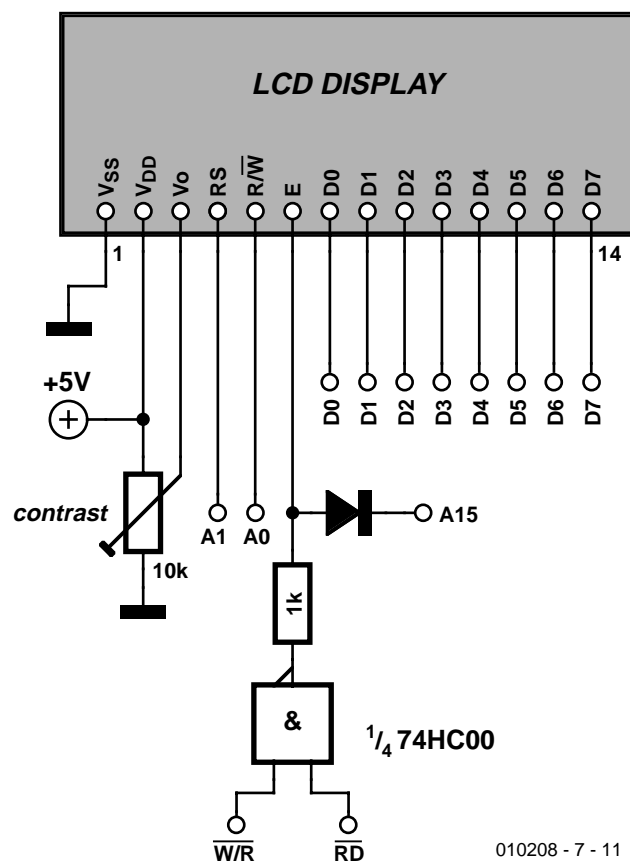
Pin	Signal
1	V_{SS} , 0V
2	V_{DD} , +5V
3	V_0 , contrast setting (0–2V)
4	RS, 1 = data, 0 = command
5	R/W, 0 = write, 1 = read
6	E, Enable signal, active High
7...14	Data bus, D0–D7

The data transfer follows the bus protocol of a 6800 processor. First the data direction must be set using the R/W line, following which the actual access takes place by means of a positive Enable pulse. With an 8051 processor, it is thus necessary to use a gate to combine the \overline{RD} and \overline{WR} signals. The data direction can be switched using an address line. An additional address line is needed to select the internal registers of the display controller via the RS line. This distinguishes

between data and commands.

The display is connected to the controller by means of a very simple

driver circuit without any address decoding, as shown in **Figure 1**. The display addresses appear above



010208 - 7 - 11

Figure 1. LCD hardware interface

address 8000h and are mirrored several times. Unfortunately, this means that it is not possible to use the board with an EEPROM in the same memory region.

Write accesses are only allowed with A0 = 0, since with A0 = 1 the display places its data on the bus. Since **WR** and **RD** are NANDed (which corresponds to WR OR RD due to the inversion), a write instruction to a read address would result in a bus conflict. Only the following addresses are thus available for driving the display:

The command register of the display can also be read back. It then supplies its Busy flag (BF) and the current cursor position (see **Table 3**). BF must without exception be tested before every write access to the LC display. A command or data may only be written to the display when BF is Low. Otherwise the display controller can be destroyed.

After power has been switched on, a few initialisation bytes must be written to the command register. An example is shown in **Table 4**.

The display has an internal data pointer assigned to the individual

program does without testing the Busy Flag, which does not cause any problems as long as driver execution from a pure BASIC-52 program takes place relatively slowly.

LCD driver for BASIC-52

Outputting individual characters is not particularly convenient in BASIC-52. However, the interpreter has a provision for an extension in the form of a specific driver that can be used to redirect PRINT commands. For this purpose, the user must generate an output routine that reads characters from register R5 (Bank 0) and directs them to the output, in this case the LCD. The user output command U0 1 can be used to redirect the output to a custom driver. When processing a PRINT command, the system jumps to address 4030h once for each character. A jump instruction to the output routine must be located at this address.

Another option for using a custom driver is provided by the PRINT@ command. This jumps to address 403Ch, where again a jump instruction to the output routine should be located. A custom driver routine will in any case only be active if bit 39 of the internal RAM (bit 7 of address 24h) is set. This can be done as part of the display initialisation routine, which is anyhow necessary.

The extension presented here includes the LCD driver along with the command extensions described below. BASIC-52 already provides numerous aids for its external extensions. Machine-language subroutines must obey the following rules:

- all registers except the eight registers in Bank 3 (18h-1Fh), along with the accumulator and the data pointer, must remain unchanged;
- the user may fully use Bank 3 for his own purposes.

Before the driver is used for the first time, the display must be prepared by calling the initialisation routine (CALL 4250h). For such user-generated functional extensions, BASIC-52 allows special calls using CALL 00, CALL 01, CALL 02 and so on up to CALL 127, which are automatically redirected to addresses 4100h, 4104h and so on up to 41FEh. Suitable jump instructions to the actual addresses allow the LCD initialisation routine (CALL 00) and the cursor routine to be called in a simplified manner. The cursor must be set using an a supplementary parameter:

CALL 427FH 0

or using the short form:

Function	7	6	5	4	3	2	1	0
Clear display	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	1	x
Shift	0	0	0	0	0	1	ID	S
	(ID= 1/0: right/left, S= 1/0: without/with text)							
Display & cursor	0	0	0	0	1	D	C	B
	(D,C,B= 1/0: Display, Cursor, Blink on/off)							
Shift	0	0	0	1	SC	RL	x	x
	(SC= 1/0: text/cursor to RL= 1/0: right/left)							
Initialisation	0	0	1	DL	N	x	x	x
	(DL= 1/0: 8/4-bit bus, N= 1/0: both lines/one line)							
Character generator	0	1	character			space		
Cursor position	1	memory address						

- 8000h: write command
- 8001h: read command
- 8002h: write data
- 8003h: read data

This memory range is mirrored as far as address FFFFh. This means that you can (for example) use the range F000h to F003h. The display has a large number of commands, with a distinction being made between different types of commands, each of which has a characteristic number of zeros in the more significant bit positions (see **Table 2**).

character positions. The following relationship applies to a 2 x 16 display:

- Line 1: addresses 00h through 0Fh
- Line 2: addresses 40h through 4Fh

A command to set the cursor position consists of a set bit 7 (80h = 128) together with the address (for example, 80h + 40h = C0h = 192 for the first character in the second line).

The small BASIC-52 program shown in **Listing 1** represents our first example of a simple driver. The

Ready?	BF	memory address (=cursor position)

Initialise with 8-bit data bus and two display lines	0011 1000 = 38h = 56
Enable display, disable cursor	0000 1100 = 0Ch = 12
Clear display	0000 0001 = 01h = 1

CALL 01 0

As can be seen in **Listing 2**, the cursor routine evaluates the expression following the CALL address, which may also be an arithmetic expression such as CALL 01 (64+4*N).

The display driver must be located in RAM starting at address 4030h. This is possible, since the entire RAM of the 89S8252 board also serves as program memory. However, the Basic RAM region must be reduced, for instance by using an MTOP=8191 statement, in order to avoid a conflict with the system's own memory management. The Basic program LCD2.BAS makes the LCD driver available in RAM (see **Listing 3**).

(010208-7)

Downloads

The download list for this issue (June 2002) can be found under 'Free Downloads' on the Elektor Electronics website. In this list, you can select the following for downloading:

- the program listings for this final instalment of the course (number **010208-17**)
- the complete set of all downloads for the course (number **010208-19**).

Conclusion & Competition

This instalment concludes the Microprocessor Basics Course. The essential elements of the hardware and software have been discussed. During the course, we received many comments and suggestions from readers, and these have frequently resulted in changes to the originally planned content of the course. We would like to cordially thank the readers for their responses. In spite of all of this, it was naturally not possible to fully cover this subject in the course. There are still many ideas and application areas left to be explored.

It is planned to have the 89S8252 Flash Board serve as the basis for new articles to appear in Elektor Electronics at irregular intervals. It has already proven itself as a platform for a wide variety of developments. If you have already implemented applications using this board or would like to do so in the near future, you can participate in an **Elektor Electronics readers' competition** to be announced in the coming issue (Summer Circuits issue, July/August 2002).

Listing 1. Driving the display in BASIC-52

```

1   REM LCD data output (LCD.BAS)
2   REM data 8002h, commands 8000h
10  STRING 80,16
20  GOSUB 1000
30  $(0)="Elektor 89S8252 "
40  GOSUB 1300
50  $(1)="LCD 2 * 16      "
60  GOSUB 1400
70  END
1000 REM LCD-Reset
1010 XBY(8000H)=56
1020 XBY(8000H)=12
1030 XBY(8000H)=1
1040 RETURN
1100 REM cursor position 0
1110 XBY(8000H)=128
1120 RETURN
1200 REM output
1210 XBY(8002H)=A
1220 RETURN
1300 REM line 1
1310 XBY(8000H)=128
1320 FOR I=1 TO 16 : XBY(8002H)=ASC$(0),I) : NEXT
1330 RETURN
1400 REM line 2
1410 XBY(8000H)=192
1420 FOR I=1 TO 16 : XBY(8002H)=ASC$(1),I) : NEXT
1430 RETURN

```

Listing 2. The display driver

```

;Basic 52 extension LCD driver (Basic52_LCD.asm)
4030 .org 4030H
4030 02 42 00 ljmp LCD ;UO 1
403C .org 403CH
403C 02 42 00 ljmp LCD ;PRINT@
4100 .org 4100H
4100 41 50 ajmp LCDInit ;CALL 0
4102 41 7F ajmp CURSOR ; CALL 1

4200 .org 4200H
4200 C0 83 LCD push DPH
4202 C0 82 push DPL
4204 C0 E0 push ACC
4206
4206 51 2E acall Busy
4208 E5 1C mov A,28 ;Cusor position
420A 44 80 orl A,#128
420C 90 80 00 mov DPTR,#8000H
420F F0 movx @DPTR,A ;set Cursor
4210 51 2E acall Busy
4212 E5 05 mov A,05 ;get char
4214 B4 0D 02 cjne A,#0DH,J1 ;= CR?
4217 80 1D sjmp CR
4219 B4 0A 02 cjne A,#0AH,J2 ;= LF?

```

```

421C 80 09          sjmp End
421E 90 80 02      J2      mov DPTR,#8002H
4221 F0           movx @DPTR,A      ;output
char
4222 E5 1C          mov A,28          ;bank 3,
R4, cursor pos.
4224 04           inc A
4225 F5 1C          mov 28,A          ;inc cur-
sor
4227 D0 E0          End      pop ACC
4229 D0 82          pop DPL
422B D0 83          pop DPH
422D 22           ret
422E
422E 90 80 01      Busy     mov DPTR,#8001H
4231 E0           movx A,@DPTR
4232 20 E7 F9          jb ACC.7,Busy
4235 22           ret
4236
4236 51 2E          CR      acall Busy
4238 74 20          mov a,#32
423A 90 80 02      mov DPTR,#8002h
423D F0           movx @DPTR,a
423E E5 1C          mov A,28
4240 04           inc A
4241 F5 1C          mov 28,a
4243 54 3F          anl A,#63
4245 B4 28 EE          cjne A,#40,CR
4248 E5 1C          J3      mov a,28
424A 54 C0          anl a,#192
424C F5 1C          mov 28,a
424E 80 D7          sjmp End
4250
4250
4250 C0 83          LCDInit push DPH
4252 C0 82          push DPL
4254 C0 E0          push ACC
4256
4256 51 2E          acall Busy
4258 90 80 00      mov DPTR,#8000H
425B 74 38          mov A,#38H
425D F0           movx @DPTR,A
425E 51 2E          acall Busy
4260 90 80 00      mov DPTR,#8000H
4263 74 06          mov A,#06H
4265 F0           movx @DPTR,A
4266 51 2E          acall Busy
4268 90 80 00      mov DPTR,#8000H
426B 74 0C          mov A,#0CH
426D F0           movx @DPTR,A
426E 51 2E          acall Busy
4270 90 80 00      mov DPTR,#8000H
4273 74 01          mov A,#01H
4275 F0           movx @DPTR,A
4276 D2 27          setb 027H ;PRINT@ enable
4278
4278 D0 E0          pop ACC
427A D0 82          pop DPL
427C D0 83          pop DPH
427E 22           ret
427F
427F C0 E0          CURSOR push ACC
4281 74 39          Mov A,#57
4283 12 00 30      lcall 030h
4286 74 01          Mov A,#1
4288 12 00 30      lcall 030h
428B E9           mov A,R1
428C F5 1C          mov 28,A
428E D0 E0          pop ACC
4290 22           ret

```

Listing 3. A loader program for the display driver

```

3999 REM LCD driver          (LCD2.bas)
4000 MTOP=8191
4029 REM output jumps
4030 DATA 002H,042H,000H,000H,000H,000H,000H,000H
4038 DATA 000H,000H,000H,000H,000H,000H,000H,000H
4039 FOR N=0 TO 15 : READ D : XBY(04030H+N)=D :
NEXT N
4099 REM CALL jumps
4100 DATA 041H,050H,041H,07FH,061H,000H,000H,000H
4101 FOR N=0 TO 7 : READ D : XBY(04100H+N)=D :
NEXT N
4199 REM LCD
4200 DATA 0C0H,083H,0C0H,082H,0C0H,0E0H,051H,02EH
4208 DATA 0E5H,01CH,044H,080H,090H,080H,000H,0F0H
4210 DATA 051H,02EH,0E5H,005H,0B4H,00DH,002H,080H
4218 DATA 01DH,0B4H,00AH,002H,080H,009H,090H,080H
4220 DATA 002H,0F0H,0E5H,01CH,004H,0F5H,01CH,0D0H
4228 DATA 0E0H,0D0H,082H,0D0H,083H,022H,090H,080H
4230 DATA 001H,0E0H,020H,0E7H,0F9H,022H,051H,02EH
4238 DATA 074H,020H,090H,080H,002H,0F0H,0E5H,01CH
4240 DATA 004H,0F5H,01CH,054H,03FH,0B4H,028H,0EEH
4248 DATA 0E5H,01CH,054H,0C0H,0F5H,01CH,080H,0D7H
4250 DATA 0C0H,083H,0C0H,082H,0C0H,0E0H,051H,02EH
4258 DATA 090H,080H,000H,074H,038H,0F0H,051H,02EH
4260 DATA 090H,080H,000H,074H,006H,0F0H,051H,02EH
4268 DATA 090H,080H,000H,074H,00CH,0F0H,051H,02EH
4270 DATA 090H,080H,000H,074H,001H,0F0H,0D2H,027H
4278 DATA 0D0H,0E0H,0D0H,082H,0D0H,083H,022H,0C0H
4280 DATA 0E0H,074H,039H,012H,000H,030H,074H,001H
4288 DATA 012H,000H,030H,0E9H,0F5H,01CH,0D0H,0E0H
4290 DATA 022H
4291 FOR N=0 TO 144 : READ D : XBY(04200H+N)=D :
NEXT N
4292 CALL 00

```

Listing 4. LCD driver test

```

100 REM driver use   (LCDDEMO.BAS)
110 UO 1 : REM User Output enable
120 CALL 01H 0 : REM Cursor to line 1
130 PRINT " PORT P1 = "
140 CALL 01H 64 : REM Cursor to line 2
150 PRINT PORT1
160 GOTO 150

```

Listing 5. The LCD.C module

```

// ----- READS51 generated header -----
// module : C:\Rigel\Reads51\Work\LCD\LCD.c
// created : 10:46:46, Tuesday, March 05, 2002
// -----

#define data_write 0x8002;
#define data_read 0x8003;
#define cmd_write 0x8000;
#define cmd_read 0x8001;

#include <Sio51.h>

void LCDinit(void){
//
//
#asm
    lcall Busy
    mov DPTR,#cmd_write
    mov A,#56
    movx @DPTR,A
    lcall Busy
    mov DPTR,#cmd_write
    mov A,#12
    movx @DPTR,A
    lcall Busy
    mov DPTR,#cmd_write
    mov A,#0x0C
    ;movx @DPTR,A
    lcall Busy
    mov DPTR,#cmd_write
    mov A,#1
    movx @DPTR,A
#endasm
}

#asm
    Busy:
    mov dptr, #cmd_read
    movx A, @dptr
    jb ACC.7, Busy
    ret
#endasm

void LCDwrite(unsigned char dat){
//
//
#asm
    lcall Busy

    mov a, BPL ;load dat to a
    add a, #0xFA
    mov dpl, a
    mov a, BPH
    addc a, #0xFF
    mov dph, a
    movx a, @dptr

    mov dptr, #data_write
    movx @dptr, a
#endasm

}

void LCDcursor (unsigned char pos)
{
#asm
    lcall Busy

    mov a, BPL ;load pos to a
    add a, #0xFA
    mov dpl, a
    mov a, BPH
    addc a, #0xFF
    mov dph, a
    movx a, @dptr

    mov dptr, #cmd_write
    orl a, #0x80
    movx @dptr, a
#endasm
}

main()
{
char dat;
unsigned char pos;
unsigned char n;

InitSerialPort0(DEF_SIO_MODE);
LCDinit();
while (1)
{
    pos=0;
    LCDcursor(pos);
    for(n=0; n<16; n++)
        LCDwrite (32);
    LCDcursor(pos);
    while (pos < 16)
    {
        dat=getc();
        if (dat == 13) pos=16;
        else LCDwrite(dat);
        pos=pos+1;
    }
    pos=64;
    LCDcursor(pos);
    for(n=0; n<16; n++)
        LCDwrite (32);
    LCDcursor(pos);
    while (pos < 80)
    {
        dat=getc();
        if (dat == 13) pos=80;
        else LCDwrite(dat);
        pos=pos+1;
    }
}
while (1);
}

```