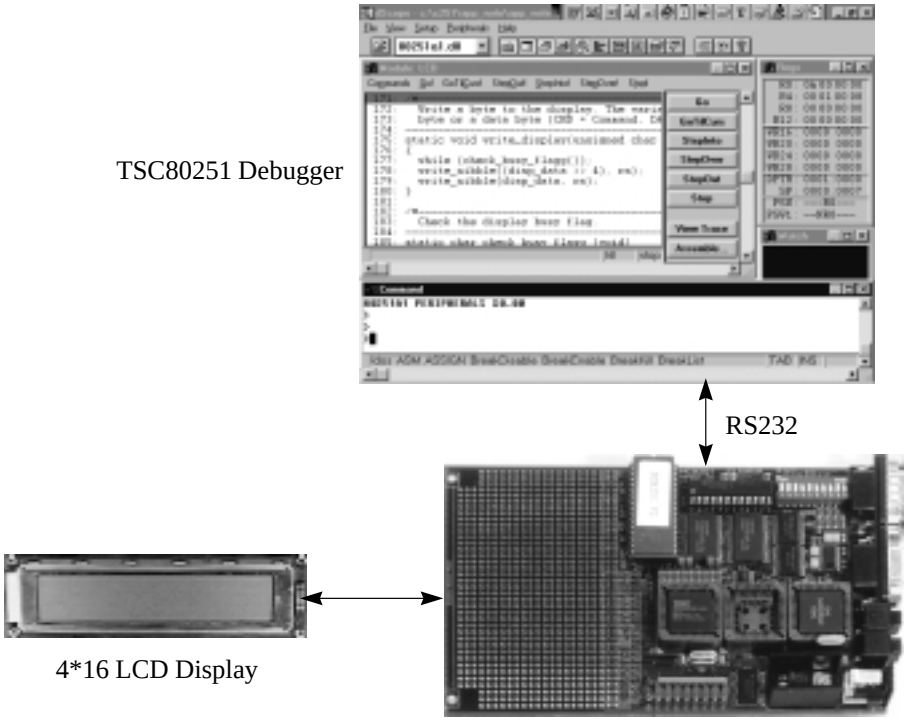
How to Interface a LCD Display to a TSC80251 Microcontroller

1. Description

This application note describes the software and hardware needed to interface an Hitachi HD44780 LCD controller & driver LSI to a TSC80251 microcontroller. The HD44780 is one of the most common controllers used for character displays from one up to four lines. It is also available from several different manufacturers. This application note demonstratres how to interface and program a four lines of 16 characters display from POWERTIP (PC 1604 –A), but can easily be used for lower resolution LCD display.

The hardware is based on the KEIL's evaluation board provided in the TSC80251 starter kit. This evaluation board is a single board computer with a small monitor program supported by Keil's software and BSO/TASKING software development tools. The board is connected to the PC via its own serial link and code can be downloaded from the development tool to the on–board RAM. The board is then used as a target system and simple emulator. A separate prototyping area can be used for user hardware.



2. Hardware

2.1. Hardware demonstration

Figure 1 shows the hardware schematic used for this application note. Sections hereafter explain how the controller is interfaced to the TSC80251.

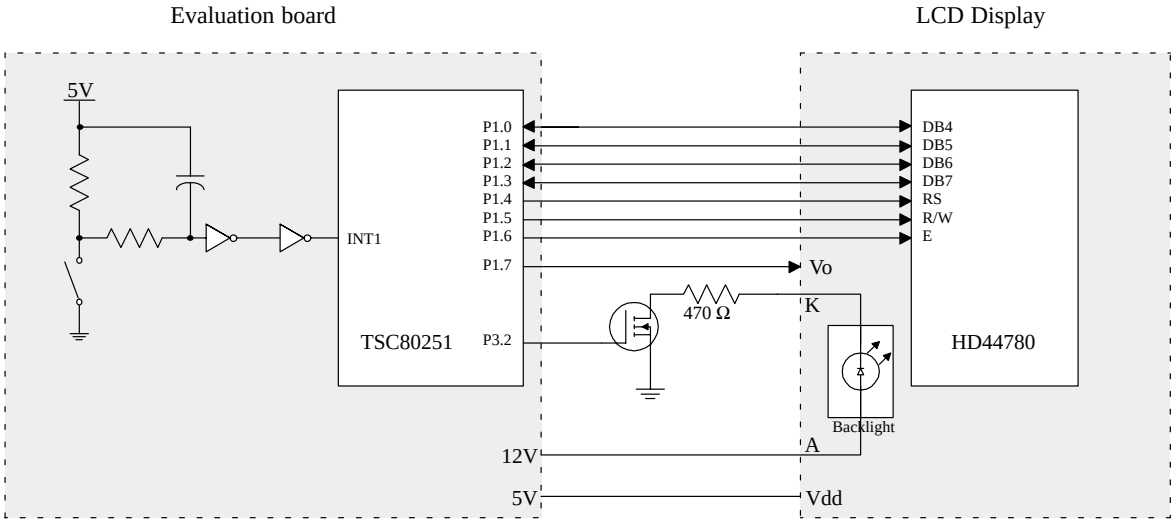


Figure 1. Hardware

2.2. Microcontroller interface

The HD44780 controller can be set up to communicate with 4–bit or 8–bit microcontrollers. The fastest way to communicate is to set up the interface for 8–bit communication and then interface the display as a memory mapped device. The drawbacks of this solution are located in the “glue” logic needed to interface the display and in the fact that wait states need to be programmed to avoid timing violations when interfaced to a TSC80251.

This application note shows how to interface the controller using the 4–bit interface mode.

When the controller is set up for 4–bit communication, the following signals are used:

- DB4, DB5, DB6, DB7 Multiplexed data bus signals (four bits are sent twice to form a byte)
- RS Register select
- R/W Read / Write signal
- E Enable

2.3. Contrast control

The recommended way to adjust the LCD contrast is to connect the Vo input of the display to a 10 kΩ variable resistor as shown in Figure 2. Adjustment in then done during factory setup. The idea proposed in this application note is to remove this variable resistor and to connect directly the Vo input to a PWM output of the Programmable Counter Array present in the TSC80251G1. This solution present the advantage to fully control the contrast by software and no external component are needed. Due to its reaction time LCD integrates the square wave signal to an analog one without adding any filter.

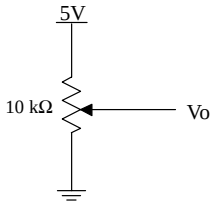


Figure 2. Contrast control

2.4. Backlight control

An output port bit is used for the backlight control through a Atmel Wireless & Microcontrollers LITTLE FOOT MOSFET.

On the evaluation board the push button connected to the external interrupt input 1 (INT1) is used by the demonstration software to toggle the LCD backlight on and off.

3. Timing

Execution times: Clear display 1.64 ms; Home cursor 40 µs; all others 40µs, except read busy flag which is complete in a single enable cycle (or two cycles in 4–bit mode), and character generator ram reads and writes which should be separated by 120µs delays. These execution times mean that after an operation, the CPU must perform Busy Flag checks until the BF (bit 7) is 0, or, when the connection to the module from the CPU is write–only, wait more than the execution time before the next operation. These times are usually strict, LCDs used in a write–only configuration should provide the specified delays.

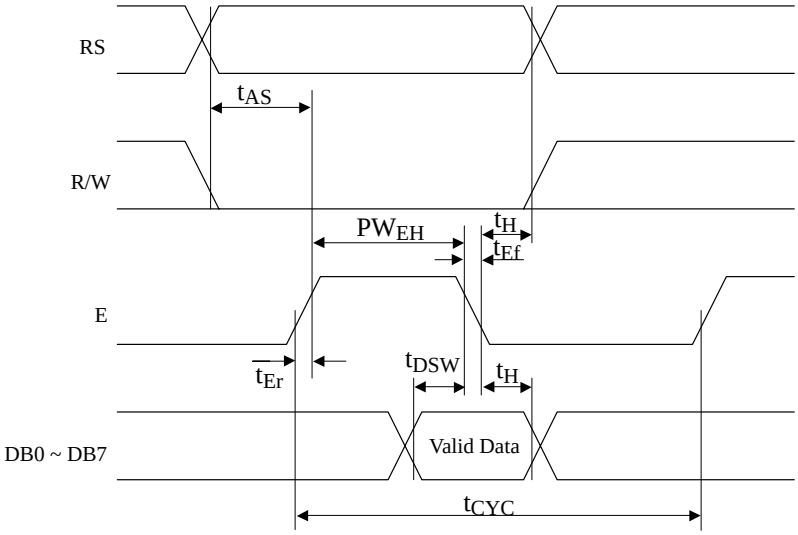


Figure 3. Write Operation

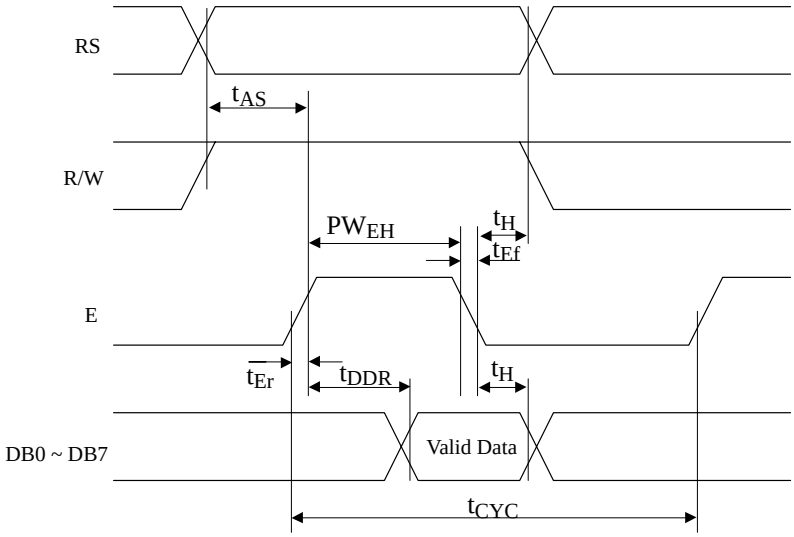


Figure 4. Read Operation

Table 1. Timing chart

Item	Symbol	Min	Max	Unit
Enable cycle time	t_{CYC}	500		ns
Enable pulse width	PW_{EH}	220		ns
Enable rise/fall time	t_{Er}, t_{Ef}		25	ns
RS, RW set ut time	t_{AS}	40		ns
Data delay time	t_{DDR}	60	120	ns
Data set up time	t_{DSW}	10		ns
Hold time	t_H	20		ns

Critisism timing is located in enable pulse width requirement. In the low level routines E pulse is created using the following ‘C’ instructions:

```
E_PORT = 1;            /* Set E high */
E_PORT = 0;            /* Set E low */
```

The compiler translates this to:

```
D296          setb   P1.6  – 4 states
C296          clr    P1.6  – 4 states
```

For a 24 MHz clock frequency one state takes 83.3ns. The E pulse will be at least 4*83.3 = 333 ns, so this value is higher than the min specification. Moreover, additional states have to be added if code is executed from external memory.

4. Software

The software written implements a ‘putchar’ function that replaces the default putchar. The ‘putchar’ function is the low–level character output routine used by the ANSI C stream I/O functions like printf, puts, vprintf etc. This way the standard ANSI C functions can be used for formatted output to the display. A number of display specific functions has also been implemented like ‘set_cursor’, ‘lcd_clr’ etc.

The following files are proposed in appendix:

- MAIN.C
  This file contains the main demonstration routine.

  void main (void); — main routine initializes LCD controller and external interrupt and write a string to the display.

- LCD.C
  This file contains the LCD display drivers.

void lcd_preter (char);	writes a character to the LCD display.
void lcd_init (void);	initializes the display after power up.
void lcd_clr (void);	clears the display and sets the cursor to home position.
void set_cursor (char, char);	sets the cursor to position y, x.
void cursor_on (void);	turns the cursor on.
void cursor_off (void);	turns the cursor off.
void backlight_on (void);	turns the backlight on.
void backlight_off (void);	turns the backlight off.

- PUTCHAR.C
  This file contains the putchar routine: the low level character output routine for the stream I/O routines.
- LCD.H
  This file is the LCD definition file and contains constants that can be ajusted by user depending on its application and LCD type (e.g. LCD port, number of row, medium contrast value...).
- PROTOS.H
  This file contains the declaration of the global functions in LCD.C. This file must be included in all files where the LCD functions are called.
- REG251G1.H
  This file is the register definition file and contains SFR and BIT addresses for the TSC80251G1.

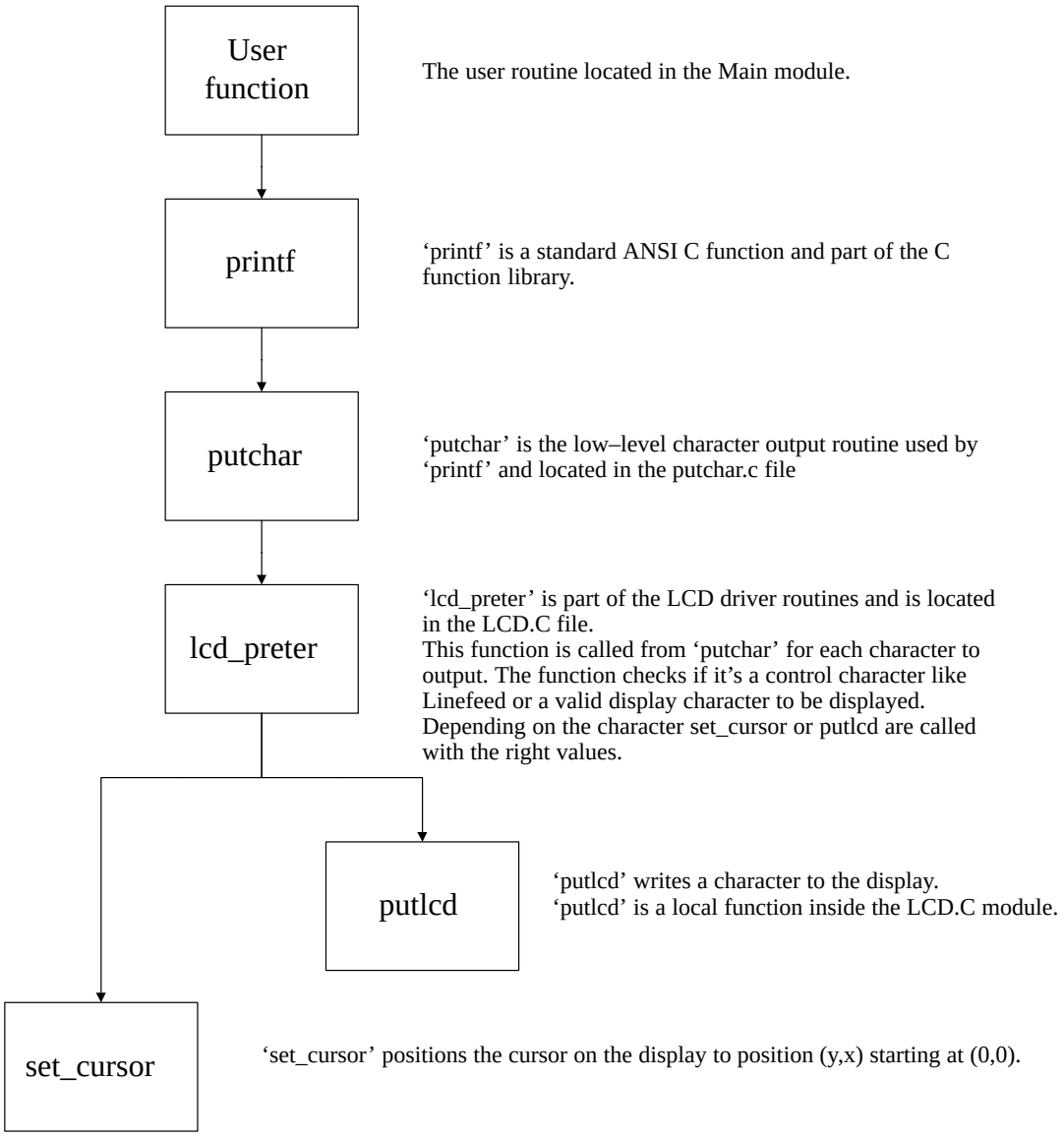
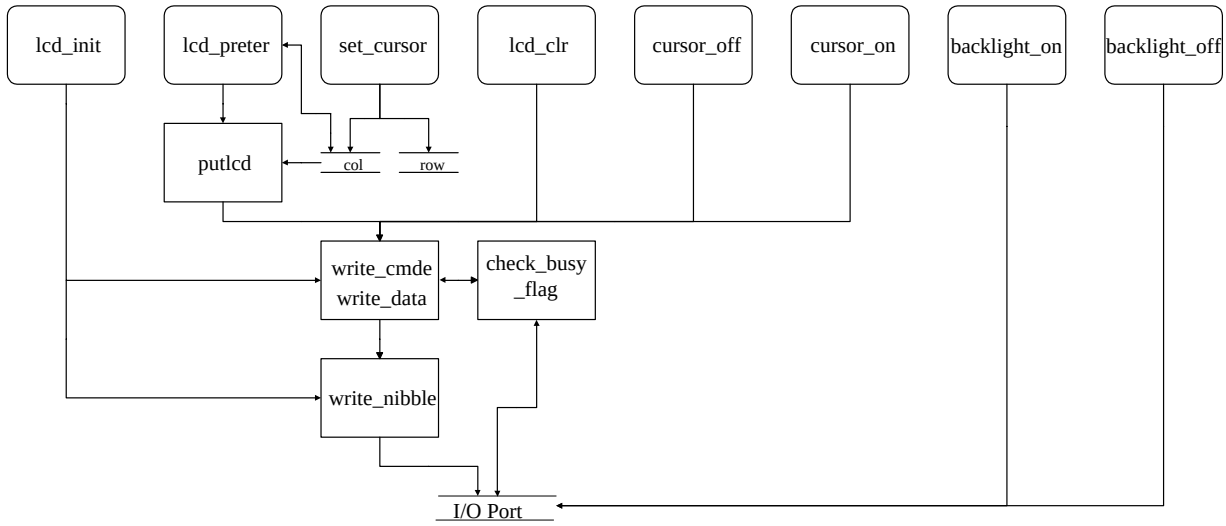


Figure 5. Print flow chart



The functions on the first line are global functions that may be called by the user.

Figure 6. LCD driver routines.

5. References

- TSC80251A1 Datasheet, Atmel Wireless & Microcontrollers
- TSC80251G1 Design Guide, Atmel Wireless & Microcontrollers
- TSC80251 Programmer's Guide, Atmel Wireless & Microcontrollers
- MCB251 Evaluation Board – User's Guide, KEIL Software
- Hitachi Liquid Crystal Character Display Module databook
- POWERTIP LCD modules databook, Powertip technology corp.

6. Sites to visit

- Atmel Wireless & Microcontrollers web pages: http://www.atmel–wm.com
- LCD Technical FAQ: http://www.paranoia.com/~filipg/HTML/LINK/F_LCD_tech.html
- HD44780–based Character–LCD http://www.iaehv.nl/users/pouweha/lcd.htm

7. Appendix A: Software

7.1. MAIN.C

```
/*
** ------------------------------------------------------------------------
** MAIN.C
** ------------------------------------------------------------------------
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** ------------------------------------------------------------------------
** Comments: This file contains the demonstration software for LCD routines
** ------------------------------------------------------------------------
*/

/*
** Includes
*/
#include <reg251G1.h>                   /* special function registers 251G1 */
#include <stdio.h>                      /* prototype for I/O functions */
#include ”protos.h”                     /* LCD driver prototypes */

/*
** Prototypes
*/
static void ex1_install (void);

/*
** ------------------------------------------------------------------------
** main – main user routine
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: demonstration routine
** ------------------------------------------------------------------------
*/
void main (void) {                      /* main entry for program */
  ex1_install();
  EA = 1;                               /* global interrupt enable */
  lcd_init();                           /* LCD display initialization */
  set_cursor(0,1);                      /* set the cursor position */
  printf(”Atmel TSC80251\n\r\n”);       /* write a text to the display */
  printf(” LCD CONTROLLER\r\n”);
  printf(” DEMO. SOFTWARE”);
  while(1);                             /* loop forever */
}


/*
** ------------------------------------------------------------------------
** ex1_install – external interrupt 1 installation
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: enable EX1 as edge interrupt
** ------------------------------------------------------------------------
*/
void ex1_install (void) {

  EX1 = 1;                              /* enable external interrupt 1 */
  IT1 = 1;                              /* make the interrupt edge triggered */
}
```

7.2. PUTCHAR.C

```
/*
** _____________________________________________________________________________
** PUTCHAR.C
** _____________________________________________________________________________
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** _____________________________________________________________________________
** Comments: This file contains the low level character output routine for
**           the stream I/O routines
** _____________________________________________________________________________
*/

/*
** Includes
*/
#include ”protos.h”                    /* LCD driver prototypes */


/*
** _____________________________________________________________________________
** putchar – send a character to the standard output
** _____________________________________________________________________________
** Inputs: char to output
**
** Outputs: char output
**
** _____________________________________________________________________________
** Comments: putchar is the low level character output routine for the stream
**           I/O routines. Default routine sends character to the serial
**           interface, new routine hereafter displays character on LCD
** _____________________________________________________________________________
*/
char putchar (char c) {
  lcd_preter(c);                       /* write the character to the display */
  return(c);
}
```

7.3. LCD.C

```
/*
** ------------------------------------------------------------------------
** LCD.C
** ------------------------------------------------------------------------
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** ------------------------------------------------------------------------
** Comments: This file contains the LCD display drivers
** ------------------------------------------------------------------------
*/

/*
** Includes
*/
#include <reg251G1.h>                   /* special function registers 251G1 */
#include <stdio.h>                      /* prototype for I/O functions */
#include ”lcd.h”                        /* define LCD constants */

/*
** Definitions
*/
static int row=0, col=0;                /* save the current cursor position */

/*
** Prototypes
*/
void lcd_preter (char);
void lcd_init (void);
void set_cursor (char,char);
void lcd_clr (void);
void cursor_on (void);
void cursor_off (void);
void backlight_on (void);
void backlight_off (void);

static void vo_install (void);
static char check_busy_flag (void);
static void putlcd (unsigned char);
static void write_cmde (unsigned char);
static void write_data (unsigned char);
static void write_nibble (unsigned char);

/*
** ------------------------------------------------------------------------
** lcd_init – LCD controller initialization and configuration routine
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: This function is called once to initialize the LCD display.
**           The HD44780 controller contains 8 RAM loactions where user
**           characters can be defined.
**           Four of those locations are used in the example to define
**           the local symbols å, Ö, Ä and Å used from the function lcd_preter
** ------------------------------------------------------------------------
*/
void lcd_init (void) {
char i;
static const char code init_tab[] = {0x33, 0x32, 0x28, 0x08, 0x01, 0x06, 0x0C};
   /* This data sequence inits the display for 4 bit interface,
      4 rows, 5*7 dots, +1 auto increment, display on, cursor off */
static const char code fonttab[] = {
  0x0E, 0x11, 0x0E, 0x01, 0x0F, 0x11, 0x0F, 0x00,  /* å – address 0 in CG RAM */
  0x0A, 0x0E, 0x11, 0x11, 0x11, 0x11, 0x0E, 0x00,  /* Ö – address 1 in CG RAM */
  0x0A, 0x00, 0x0E, 0x11, 0x11, 0x1F, 0x11, 0x00,  /* Ä – address 2 in CG RAM */
  0x0E, 0x11, 0x0E, 0x11, 0x11, 0x1F, 0x11, 0x00}; /* Å – address 3 in CG RAM */
```

```
  for (i = 0; i < sizeof init_tab; ++i) {
    write_cmde(initdata[i]);              /* write init data to the controller */
  }
  write_cmde(CG_RAM);                     /* select character graphic RAM */
  for (i = 0; i < sizeof font_tab; ++i) {
    write_data(fonttab[i]);               /* load Swedish characters */
  }
  write_cmde(DD_RAM);                     /* select display data RAM */
  vo_install();                           /* Vo driving */
}

/*
** ------------------------------------------------------------------------
** vo_install – contrast output installation
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: enable PWM output on PCA channel 4
** ------------------------------------------------------------------------
*/
static void vo_install (void) {
  VO_PORT = 1;                            /* enable alternate function */
  CMOD = 0x00;                            /* PCA clock = Fosc/12 */
  CCAPM4 = 0x42;                          /* set channel 4 as PWM output */
  CCAP4H = VO_MEDIUM                      /* set contrast to medium value */
  CR = 1;                                 /* start PCA timer */
}

/*
** ------------------------------------------------------------------------
** set_cursor – set the cursor position
** ------------------------------------------------------------------------
** Inputs: row and column position
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: Rows and columns start from 0 (zero–based)
** ------------------------------------------------------------------------
*/
void set_cursor (char rownr, char colnr) {
int cmd;
  rownr= rownr%NB_ROW;                    /* make sure position is in limits */
  colnr= colnr%NB_COL;                    /* make sure position is in limits */
  switch (rownr) {
    case 0:
      cmd = ROW0;
      break;
    case 1:
      cmd = ROW1;
      break;
    case 2:
      cmd = ROW2;
      break;
    case 3:
      cmd = ROW3;
      break;
  }
  write_cmde(cmd | colnr);                /* write the new cursor position */
  row = rownr;                            /* update the global row position */
  col = colnr;                            /* update the global column position */
}
```

```
/*
** ------------------------------------------------------------------------
** lcd_clr – clear display and set cusor to home
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments:
** ------------------------------------------------------------------------
*/
void lcd_clr(void) {
  write_cmde(CLR_LCD);
}

/*
** ------------------------------------------------------------------------
** cursor_on  – enable cursor
** cursor_off – disable cursor
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments:
** ------------------------------------------------------------------------
*/
void cursor_on (void) {
  write_cmde(CUR_ON);
}

void cursor_off (void) {
  write_cmde(CUR_OFF);
}

/*
** ------------------------------------------------------------------------
** backlight_on  – turn backlight LED on
** backlight_off – turn backlight LED off
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments:
** ------------------------------------------------------------------------
*/
void backlight_on (void) {
  BL_PORT= 1;
}

void backlight_off (void) {
  BL_PORT= 0;
}
```

```
/*
** ____________________________________________________________________________
** write_cmde – write a command byte to the display
** ____________________________________________________________________________
** Inputs: command byte to write
**
** Outputs:
**
** ____________________________________________________________________________
** Comments:
** ____________________________________________________________________________
*/
static void write_cmde (unsigned char cmde) {
  while (check_busy_flag());            /* wait if the display is busy */
  RS_PORT = 0;                          /* select command registers */
  write_nibble(cmde >> 4);              /* write the high nibble */
  write_nibble(cmde);                   /* write the low nibble */
}

/*
** ____________________________________________________________________________
** write_data – write a data byte to the display
** ____________________________________________________________________________
** Inputs: data byte to write
**
** Outputs:
**
** ____________________________________________________________________________
** Comments:
** ____________________________________________________________________________
*/
static void write_data (unsigned char byte) {
  while (check_busy_flag());            /* wait if the display is busy */
  RS_PORT = 1;                          /* select data registers */
  write_nibble(byte >> 4);              /* write the high nibble */
  write_nibble(byte);                   /* write the low nibble */
}

/*
** ____________________________________________________________________________
** check_busy_flag – check the display busy flag
** ____________________________________________________________________________
** Inputs:
**
** Outputs:
**
** ____________________________________________________________________________
** Comments: invoked from write_cmde and write_data
** ____________________________________________________________________________
*/
static char check_busy_flag (void) {
char bf;
  LCD_PORT |= LCD_DAT_MSK;              /* data bus high */
  RS_PORT = 0;                          /* select command registers */
  RW_PORT = 1;                          /* read access */
  E_PORT = 1;                           /* Set E high */
  bf = LCD_PORT & BF_BIT;               /* read busy flag */
  E_PORT = 0;                           /* set E low */
  E_PORT = 1;                           /* second read due to 4 bit interface */
  E_PORT = 0;
  return (bf);                          /* return busy flag */
}
```

```
/*
** ------------------------------------------------------------------------
** write_nibble – write a 4–bit nibble to the display
** ------------------------------------------------------------------------
** Inputs: nibble to write
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments:
** ------------------------------------------------------------------------
*/
static void write_nibble (unsigned char nibble) {
  LCD_PORT &= ~LCD_DAT_MSK;             /* data bus low */
  RW_PORT = 0;                          /* write access */
  nibble &= LCD_DAT_MSK;                /* clear high nibble */
  LCD_PORT |= nibble;                   /* write the nibble */
  E_PORT = 1;                           /* set E high */
  E_PORT = 0;                           /* set E low */
}

/*
** ------------------------------------------------------------------------
** lcd_preter – lcd display interpreter
** ------------------------------------------------------------------------
** Inputs:
**
** Outputs:
**
** ------------------------------------------------------------------------
** Comments: invoked from putchar
** ------------------------------------------------------------------------
*/
void lcd_preter (char c) {
  switch (c) {
    case C_R:                           /* Carriage Return */
      set_cursor(row, 0);
      break;
    case L_F:                           /* Line Feed */
      row = (row+1)%NB_ROW;
      set_cursor(row, 0);
      break;
    case B_S:                           /* Back Space */
      if (col != 0) {
        set_cursor(row, --col);
        putlcd(' ');
        set_cursor(row, --col);
      }
      break;
    case 'ö':
      putlcd(0xef);                     /* display special character */
      break;
    case 'ä':
      putlcd(0xe1);                     /* display special character */
      break;
    case 'å':
      putlcd(0);                        /* display special char from display RAM */
      break;
    case 'Ö':
      putlcd(1);                        /* display special char from display RAM */
      break;
    case 'Ä':
      putlcd(2);                        /* display special char from display RAM */
      break;
    case 'Å':
      putlcd(3);                        /* display special char from display RAM */
      break;
    default:
      if (c>0x1f && c<0x80) {           /* check if it's a valid character */
        putlcd(c);                      /* write the character to the display */
      }
      break;
  }
}
```

```
/*
** _____________________________________________________________________________
** putlcd – write one character to the display
** _____________________________________________________________________________
** Inputs: data to display
**
** Outputs:
**
** _____________________________________________________________________________
** Comments: invoked from lcd_preter
** _____________________________________________________________________________
*/
static void putlcd (unsigned char disp_data) {
  if (col<NB_COL && col>=0) {           /* check if we are inside the limits */
    write_data(disp_data);              /* write the character to the display */
    col++;                              /* update the column position */
  }
}

/*
** _____________________________________________________________________________
** interrupt_int1 – backlight toggling
** _____________________________________________________________________________
** Inputs:
**
** Outputs:
**
** _____________________________________________________________________________
** Comments: INT1 switch used for backlight on/off
** _____________________________________________________________________________
*/
void interrupt_int1 (void) interrupt 2 {
  BL_PORT= ~BL_PORT;
}
```

7.4. PROTOS.H

```
/*
** ____________________________________________________________________________
** PROTOS.H
** ____________________________________________________________________________
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** ____________________________________________________________________________
** Comments: This file contains the LCD prototypes definitions, it must be
**           included in all files where the LCD functions are called
** ____________________________________________________________________________
*/

#ifndef _PROTOS_H_
#define _PROTOS_H_

/*
** Prototypes
*/
extern void lcd_preter (char);
extern void lcd_init (void);
extern void lcd_clr (void);
extern void set_cursor (char, char);
extern void cursor_on (void);
extern void cursor_off (void);
extern void backlight_on (void);
extern void backlight_off (void);

#endif /* _PROTOS_H_ */
```

7.5. LCD.H

```
/*
** ____________________________________________________________________________
** LCD.H
** ____________________________________________________________________________
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** ____________________________________________________________________________
** Comments: This file contains the LCD controller definitions
** ____________________________________________________________________________
*/

#ifndef _LCD_H_
#define _LCD_H_

/*
** Defines
*/
/*----------------------------------------------------------------------------
   Definition for Hitachi HD44780 LCD controller 4x16 characters, 4 bit interface

   DB4       = LCD_PORT.0
   DB5       = LCD_PORT.1
   DB6       = LCD_PORT.2
   DB7       = LCD_PORT.3
   RS        = LCD_PORT.4
   R/W       = LCD_PORT.5
   E         = LCD_PORT.6
   Vo        = LCD_PORT.7
   Backlight = P3.2
----------------------------------------------------------------------------*/

#define LCD_PORT       P1              /* LCD port definition */
sbit RS_PORT = 0x94;                   /* P1.4 */
sbit RW_PORT = 0x95;                   /* P1.5 */
sbit E_PORT =  0x96;                   /* P1.6 */
sbit VO_PORT = 0x97;                   /* P1.7 */

sbit BL_PORT = 0xB2;                   /* P3.2 */

#define LCD_DAT_MSK    0x0F            /* data bus mask */
#define BF_BIT         0x08            /* busy flag mask */

#define NB_ROW         0x04            /* 4 display lines */
#define NB_COL         0x10            /* 16 characters per line */

/*
  Controller commands
*/
#define ROW0           0x80            /* row 0 command */
#define ROW1           0xC0            /* row 1 command */
#define ROW2           0x90            /* row 2 command */
#define ROW3           0xD0            /* row 3 command */
#define CG_RAM         0x40            /* character graphic RAM command */
#define DD_RAM         0x80            /* data display RAM command */
#define CLR_LCD        0x01            /* clear LCD command */
#define CUR_ON         0x0F            /* cusrsor ON command */
#define CUR_OFF        0x0C            /* cusrsor OFF command */

#define VO_MEDIUM      180             /* medium contrast value */

#define C_R            0x0D
#define L_F            0x0A
#define B_S            0x08

#endif /* _LCD_H_ */
```

7.6. REG251G1.H

```
/*
** ____________________________________________________________________________
** REG251G1.H
**____________________________________________________________________________
** Company: Atmel Wireless & Microcontrollers
** Revision: 1.0
** ____________________________________________________________________________
** Comments: This file contains the registers definition for
**           TSC8x251G1 microcontroller
** ____________________________________________________________________________
*/

#ifndef _REG251G1_H_
#define _REG251G1_H_

/*
** Defines
*/

/* BYTE REGISTERS
   ______________ */
/* ports */
sfr P0     = 0x80;
sfr P1     = 0x90;
sfr P2     = 0xA0;
sfr P3     = 0xB0;

/* system */
sfr ACC    = 0xE0;
sfr B      = 0xF0;

sfr PSW    = 0xD0;
sfr PSW1   = 0xD1;

sfr SP     = 0x81;
sfr SPH    = 0xBE;

sfr DPL    = 0x82;
sfr DPH    = 0x83;
sfr DPXL   = 0x84;

sfr IE0    = 0xA8;
sfr IPL0   = 0xB8;
sfr IPH0   = 0xB7;
sfr IE1    = 0xB1;
sfr IPL1   = 0xB2;
sfr IPH1   = 0xB3;

sfr WCON   = 0xA7;

/* timers */
sfr TCON   = 0x88;
sfr TMOD   = 0x89;
sfr TL0    = 0x8A;
sfr TL1    = 0x8B;
sfr TH0    = 0x8C;
sfr TH1    = 0x8D;

sfr T2CON  = 0xC8;
sfr T2MOD  = 0xC9;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;
sfr TL2    = 0xCC;
sfr TH2    = 0xCD;

/* uart */
sfr SCON   = 0x98;
sfr SBUF   = 0x99;
sfr SADDR  = 0xA9;
sfr SADEN  = 0xB9;
```

```
sfr BRL    = 0x9A;
sfr BRDCON = 0x9B;

/* pca */
sfr CCON   = 0xD8;
sfr CMOD   = 0xD9;
sfr CL     = 0xE9;
sfr CH     = 0xF9;
sfr CCAPM0 = 0xDA;
sfr CCAPM1 = 0xDB;
sfr CCAPM2 = 0xDC;
sfr CCAPM3 = 0xDD;
sfr CCAPM4 = 0xDE;
sfr CCAP0L = 0xEA;
sfr CCAP1L = 0xEB;
sfr CCAP2L = 0xEC;
sfr CCAP3L = 0xED;
sfr CCAP4L = 0xEE;
sfr CCAP0H = 0xFA;
sfr CCAP1H = 0xFB;
sfr CCAP2H = 0xFC;
sfr CCAP3H = 0xFD;
sfr CCAP4H = 0xFE;

/* power management */
sfr PCON   = 0x87;
sfr PFILT  = 0x86;
sfr CKRL   = 0x8E;
sfr POWM   = 0x8F;

/* sslc */
sfr SSCON  = 0x93;
sfr SSCS   = 0x94;
sfr SDAT   = 0x95;
sfr SSADR  = 0x96;
sfr SSBR   = 0x92;

/* keyboard */
sfr P1LS   = 0x9C;
sfr P1IE   = 0x9D;
sfr P1F    = 0x9E;

/* watchdog */
sfr WDTRST = 0xA6;


/* BIT REGISTERS
   ------------ */
/* PSW  */
sbit CY    = 0xD7;
sbit AC    = 0xD6;
sbit F0    = 0xD5;
sbit RS1   = 0xD4;
sbit RS0   = 0xD3;
sbit OV    = 0xD2;
sbit UD    = 0xD1;
sbit P     = 0xD0;

/* TCON */
sbit TF1   = 0x8F;
sbit TR1   = 0x8E;
sbit TF0   = 0x8D;
sbit TR0   = 0x8C;
sbit IE1_  = 0x8B;
sbit IT1   = 0x8A;
sbit IE0_  = 0x89;
sbit IT0   = 0x88;

/* T2CON */
sbit TF2   = 0xCF;
sbit EXF2  = 0xCE;
```

```
sbit RCLK  = 0xCD;
sbit TCLK  = 0xCC;
sbit EXEN2 = 0xCB;
sbit TR2   = 0xCA;
sbit C_T2_ = 0xC9;
sbit CP_RL2= 0xC8;

/* IE0 */
sbit EA    = 0xAF;
sbit EC    = 0xAE;
sbit EADC  = 0xAD;
sbit ES    = 0xAC;
sbit ET1   = 0xAB;
sbit EX1   = 0xAA;
sbit ET0   = 0xA9;
sbit EX0   = 0xA8;

/* IPL0 */
sbit IPLC  = 0xBE;
sbit IPLT2 = 0xBD;
sbit IPLS  = 0xBC;
sbit IPLT1 = 0xBB;
sbit IPLX1 = 0xBA;
sbit IPLT0 = 0xB9;
sbit IPLX0 = 0xB8;

/* P1 */
sbit SDA   = 0x97;
sbit SCL   = 0x96;
sbit CEX4  = 0x97;
sbit CEX3  = 0x96;
sbit CEX2  = 0x95;
sbit CEX1  = 0x94;
sbit CEX0  = 0x93;
sbit ECI   = 0x92;
sbit T2EX  = 0x91;
sbit T2    = 0x90;

/* P3 */
sbit RD    = 0xB7;
sbit WR    = 0xB6;
sbit T1    = 0xB5;
sbit T0    = 0xB4;
sbit INT1  = 0xB3;
sbit INT0  = 0xB2;
sbit TXD   = 0xB1;
sbit RXD   = 0xB0;

/* SCON */
sbit FE    = 0x9F;
sbit SM0   = 0x9F;
sbit SM1   = 0x9E;
sbit SM2   = 0x9D;
sbit REN   = 0x9C;
sbit TB8   = 0x9B;
sbit RB8   = 0x9A;
sbit TI    = 0x99;
sbit RI    = 0x98;

/* CCON */
sbit CF    = 0xDF;
sbit CR    = 0xDE;
sbit CCF4  = 0xDC;
sbit CCF3  = 0xDB;
sbit CCF2  = 0xDA;
sbit CCF1  = 0xD9;
sbit CCF0  = 0xD8;

/*  PCON  */
sbit SMOD1 = PCON ^ 7;
sbit SMOD0 = PCON ^ 6;
```

```
sbit RPD   = PCON ^ 5;
sbit POF   = PCON ^ 4;
sbit GF1   = PCON ^ 3;
sbit GF0   = PCON ^ 2;
sbit PD    = PCON ^ 1;
sbit IDL   = PCON ^ 0;

/* IE1 */
sbit SSIE  = IE1 ^ 5;
sbit KBIE  = IE1 ^ 0;

/* IPH0 */
sbit IPHC  = IPH0 ^ 6;
sbit IPHT2 = IPH0 ^ 5;
sbit IPHS  = IPH0 ^ 4;
sbit IPHT1 = IPH0 ^ 3;
sbit IPHX1 = IPH0 ^ 2;
sbit IPHT0 = IPH0 ^ 1;
sbit IPHX0 = IPH0 ^ 0;

/* IPH1 */
sbit IPHSS = IPH1 ^ 5;
sbit IPHKB = IPH1 ^ 0;

/* IPL1 */
sbit IPLSS = IPL1 ^ 5;
sbit IPLKB = IPL1 ^ 0;

#endif /* _REG251G1_H_ */
```