AVR Microcontroller

Microprocessor Course

Chapter 12

LCD and Keyboard Interfacing

Bahman 1397 (Version 1.2)

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

LCD operation

In recent years the LCD is finding widespread use replacing LEDs (seven segment LEDs or other multi segment LEDs). This is due to the following reasons:

1. The declining prices of LCDs.
2. The ability to display numbers, characters, and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.
3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data.
4. Ease of programming for characters and graphics.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

LCD pin descriptions

The LCD discussed in this section has 14 pins. The function of each pin is given in Table 12-1. Figure 12-1 shows the pin positions for various LCDs.

V_{CC}, V_{SS} and V_{EE}

While VCC and VSS provide +5 V and ground, respectively, VEE is used for controlling LCD contrast.

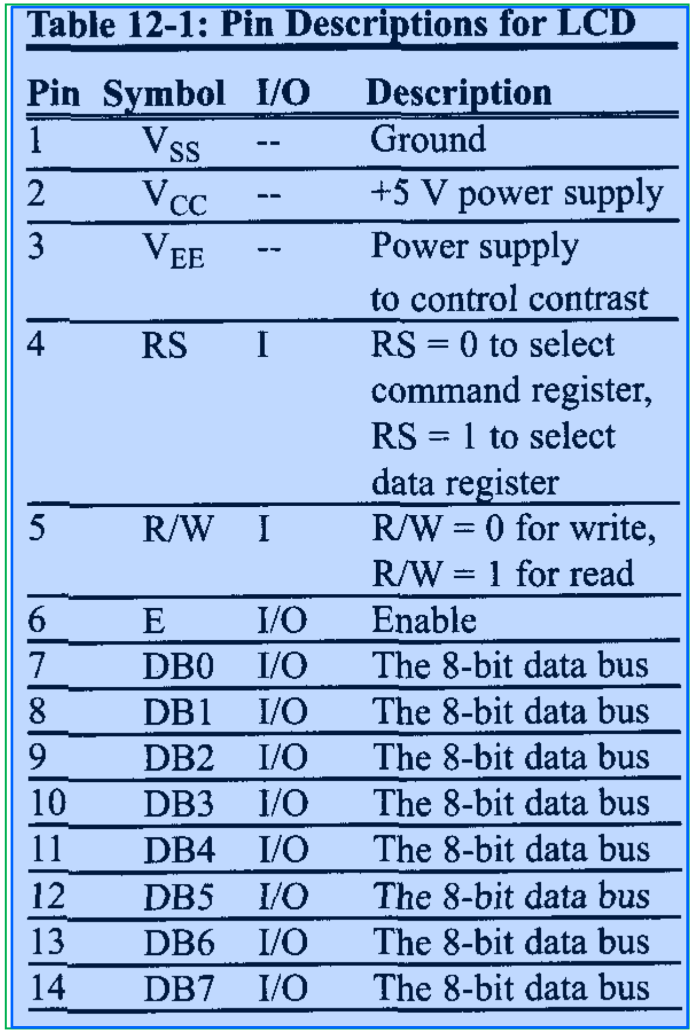


Table 12-1: Pin Descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5 V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

RS, register select

There are two very important registers inside the LCD. The RS pin is used for their selection as follows.

If RS = 0, the instruction command code register is selected, allowing the user to send commands such as clear display, cursor at home, and so on.

If RS = 1 the data register is selected, allowing the user to send data to be displayed on the LCD.

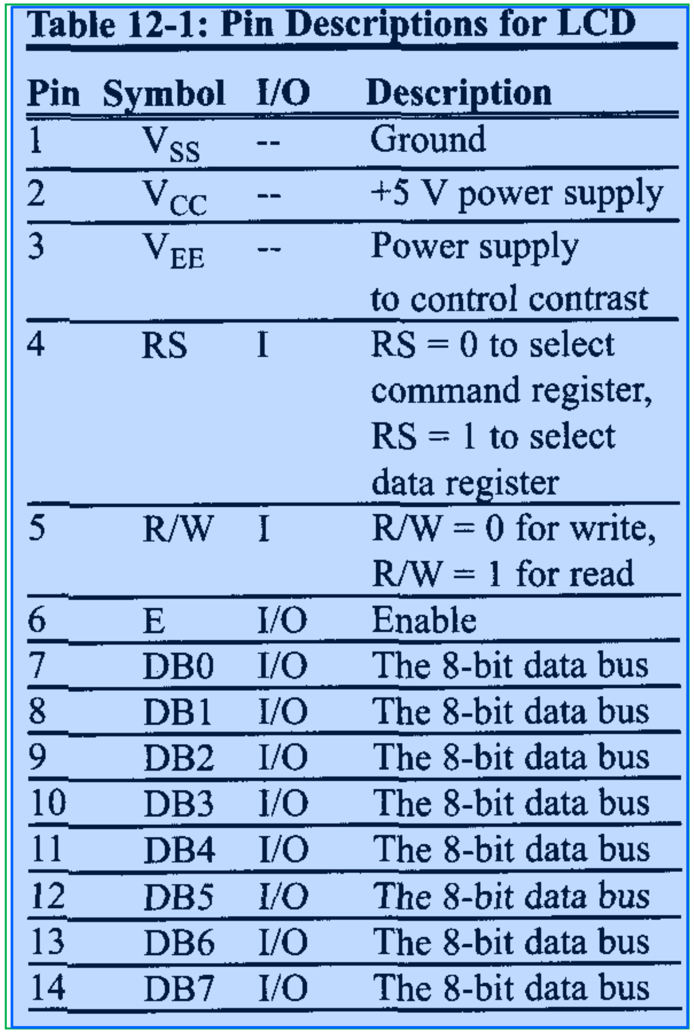


Table 12-1: Pin Descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5 V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

R/W, read/write

R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading; R/W = 0 when writing.

E, enable

The enable pin is used by the LCD to latch information presented to its data pins.

When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450 ns wide.

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

D0-D7

The 8-bit data pins, D0-D7, are used to send information to the LCD or read the content of the LCS's internal registers.

To display letters and numbers, we send ASCII codes for letters A-Z, a-z, and numbers 0-9 to these pins while making RS=1.

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

LCD Command Codes

These commands can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.

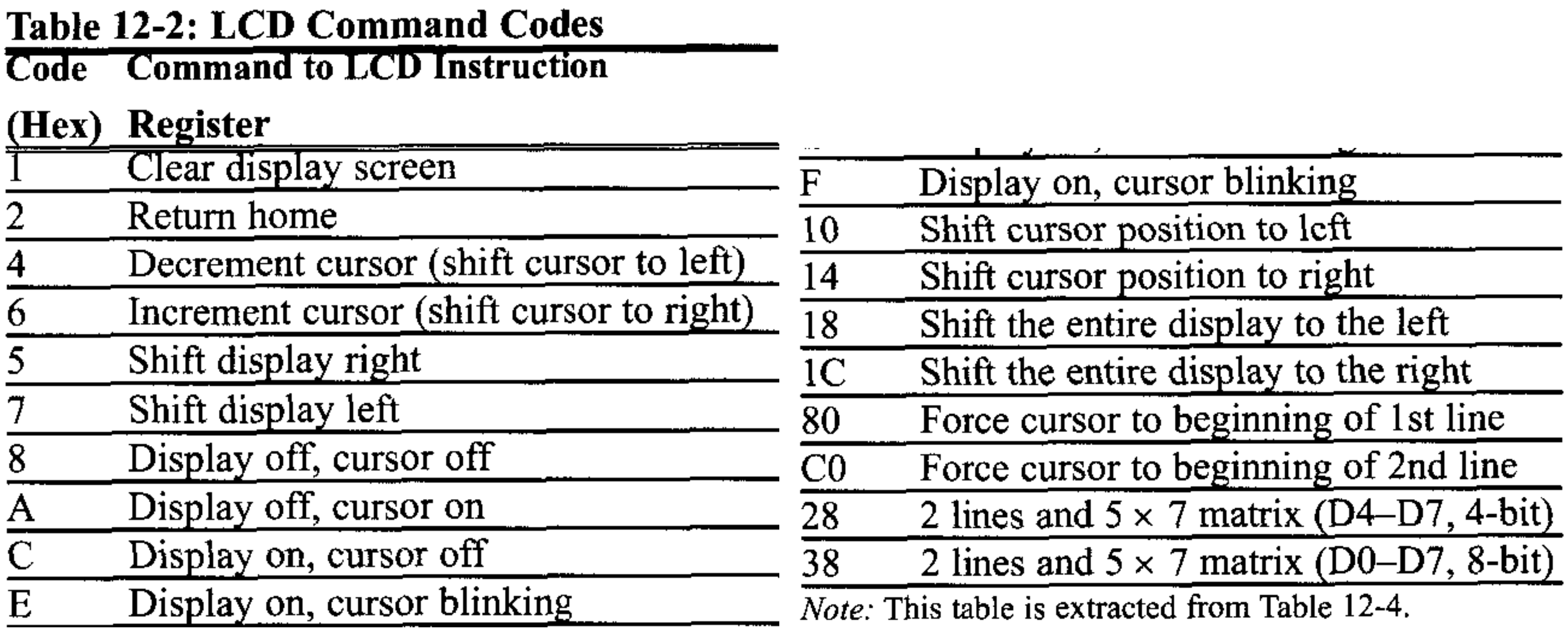


Table 12-2: LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
28	2 lines and 5 × 7 matrix (D4–D7, 4-bit)
38	2 lines and 5 × 7 matrix (D0–D7, 8-bit)

Note: This table is extracted from Table 12-4.

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

In this section you will see how to interface an LCD to the AVR in two different ways.

We can use 8-bit data or 4-bit data options.

The 8-bit data interfacing is easier to program but uses 4 more pins.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Dot matrix character LCDs are available in different packages. The Figure shows the position of each pin in different packages.

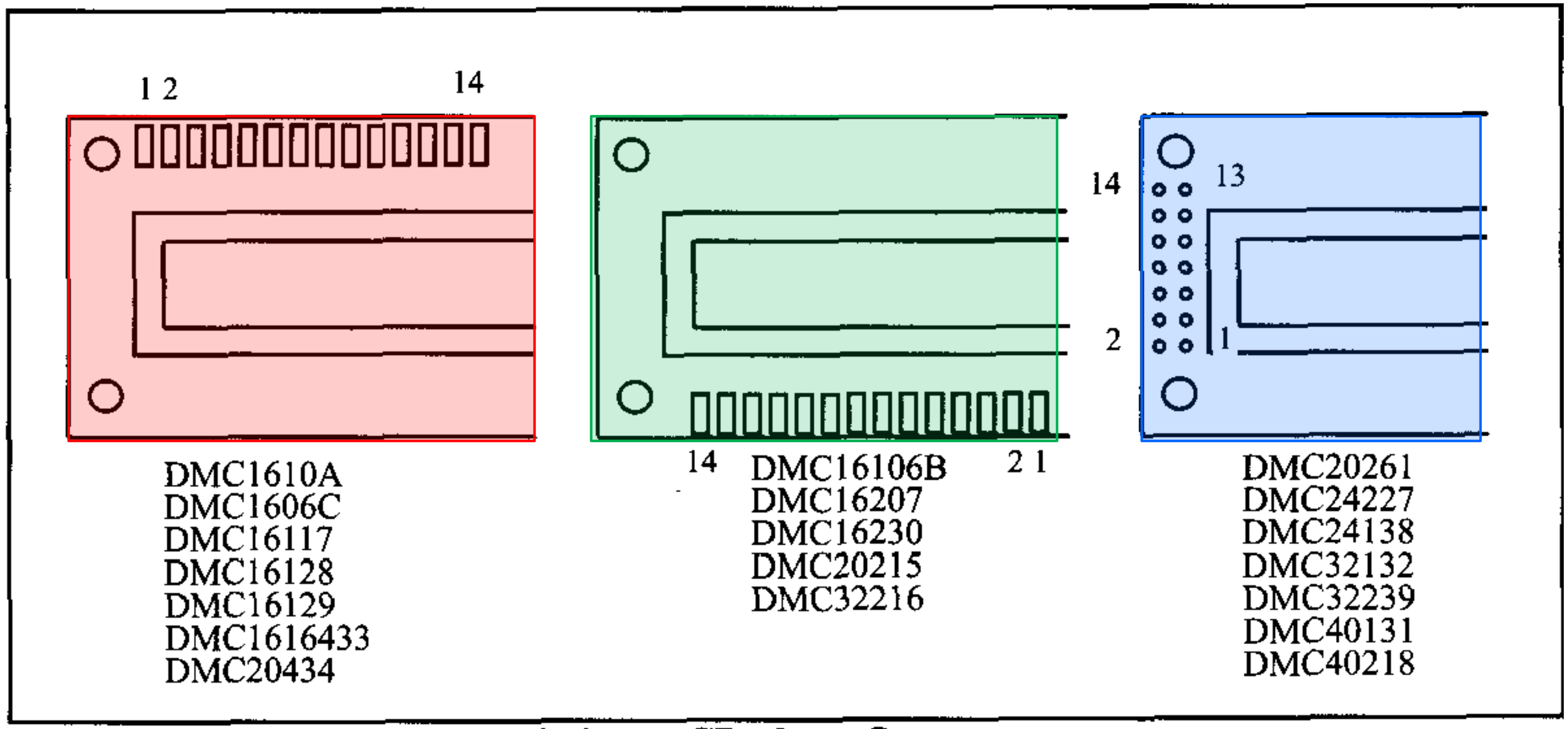


Figure 12-1. Pin Positions for Various LCDs from Optrex

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

Sending commands and data to LCDs

To send data and commands to LCDs you should do the following steps. Notice that steps 2 and 3 can be repeated many times:

1. Initialize the LCD.
2. Send any of the commands from Table 12-2 to the LCD.
3. Send the character to be shown on the LCD.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Initializing the LCD

To initialize the LCD for 5 x 7 matrix and 8-bit operation, the following sequence of commands should be sent to the LCD:

0x38, 0x0E, 0x01.

After power-up you should wait about 15ms before sending initializing commands to the LCD. If the LCD initializer function is not the first function in your code you can omit this delay.

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

Sending commands to the LCD

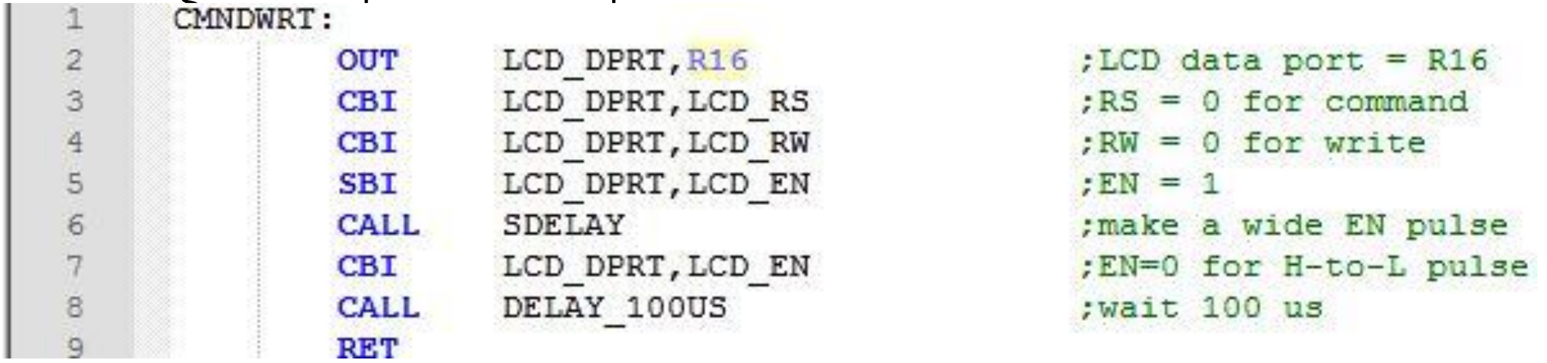
To send any of the commands from Table 12-2 to the LCD, make pins RS and R/W=0 and put the command number on the data pins (D0-D7). Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD.

RS = 0

R/W= 0

DATA on DATA Pins (D0..D7)

send a high-to-low pulse to the E pin



```
CMNDWRT:
        OUT     LCD_DPRT,R16            ;LCD data port = R16
        CBI     LCD_DPRT,LCD_RS         ;RS = 0 for command
        CBI     LCD_DPRT,LCD_RW         ;RW = 0 for write
        SBI     LCD_DPRT,LCD_EN         ;EN = 1
        CALL    SDELAY                  ;make a wide EN pulse
        CBI     LCD_DPRT,LCD_EN         ;EN=0 for H-to-L pulse
        CALL    DELAY_100US             ;wait 100 us
        RET
```

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Sending commands to the LCD

Notice that after each command you should wait about 100µs to let the LCD module run the command. Clear LCD and Return Home commands are exceptions to this rule. After the 0x01 and 0x02 commands you should wait for about 2 ms. Table 12-3 shows the details of commands and their execution times.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Sending data to the LCD

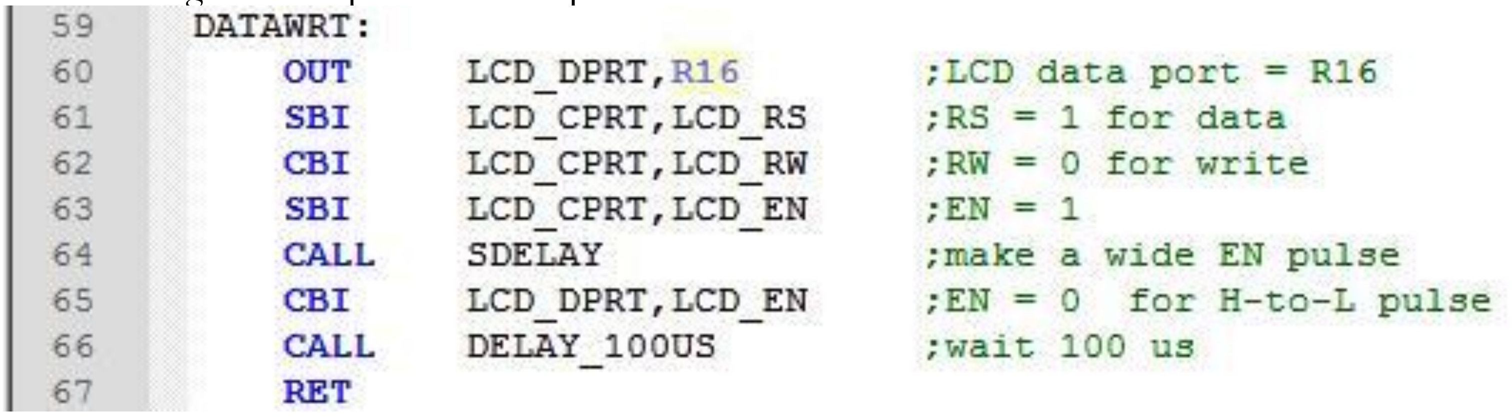
To send data to the LCD, make pins RS=1 and R/W=0. Then put the data on the data pins (D0-D7) and send a high-to-low pulse to the E pin to enable the internal latch of the LCD. Notice that after sending data you should wait about 100µs to let the LCD module write the data on the screen.

RS = 1

R/W= 0

DATA on DATA Pins (D0..D7)

send a high-to-low pulse to the E pin



```
DATAWRT:
        OUT     LCD_DPRT,R16        ;LCD data port = R16
        SBI     LCD_CPRT,LCD_RS     ;RS = 1 for data
        CBI     LCD_CPRT,LCD_RW     ;RW = 0 for write
        SBI     LCD_CPRT,LCD_EN     ;EN = 1
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_DPRT,LCD_EN     ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US         ;wait 100 us
        RET
```

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

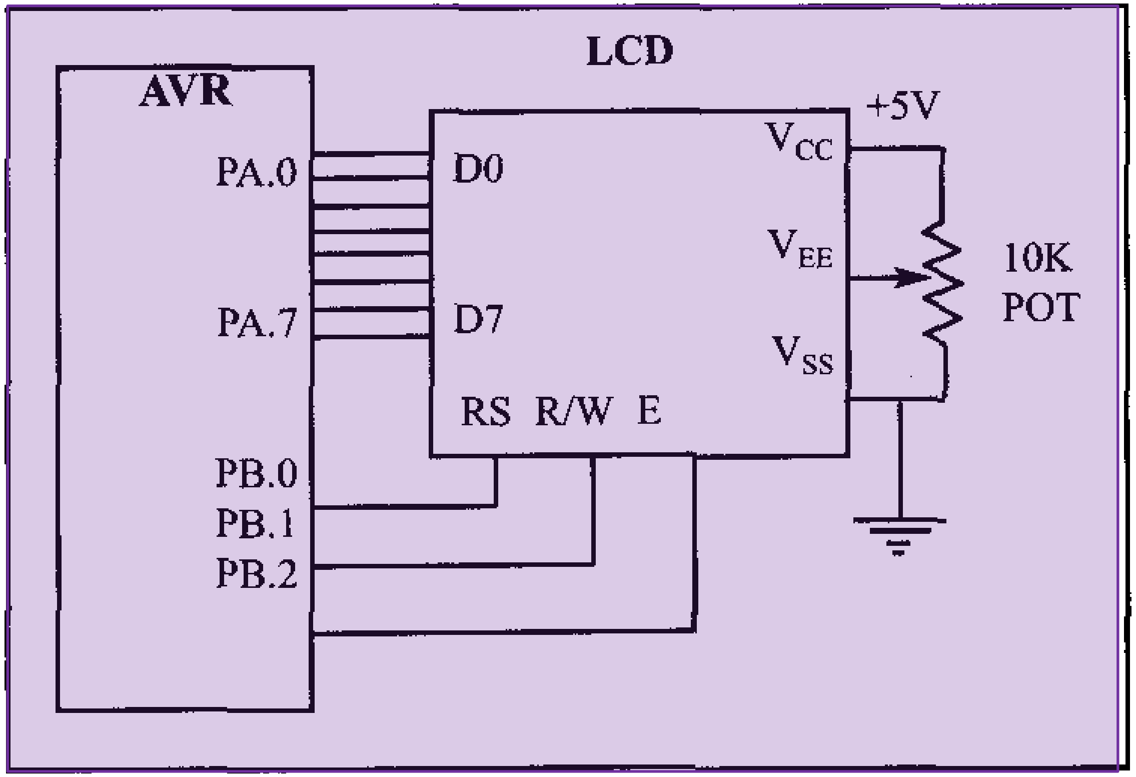
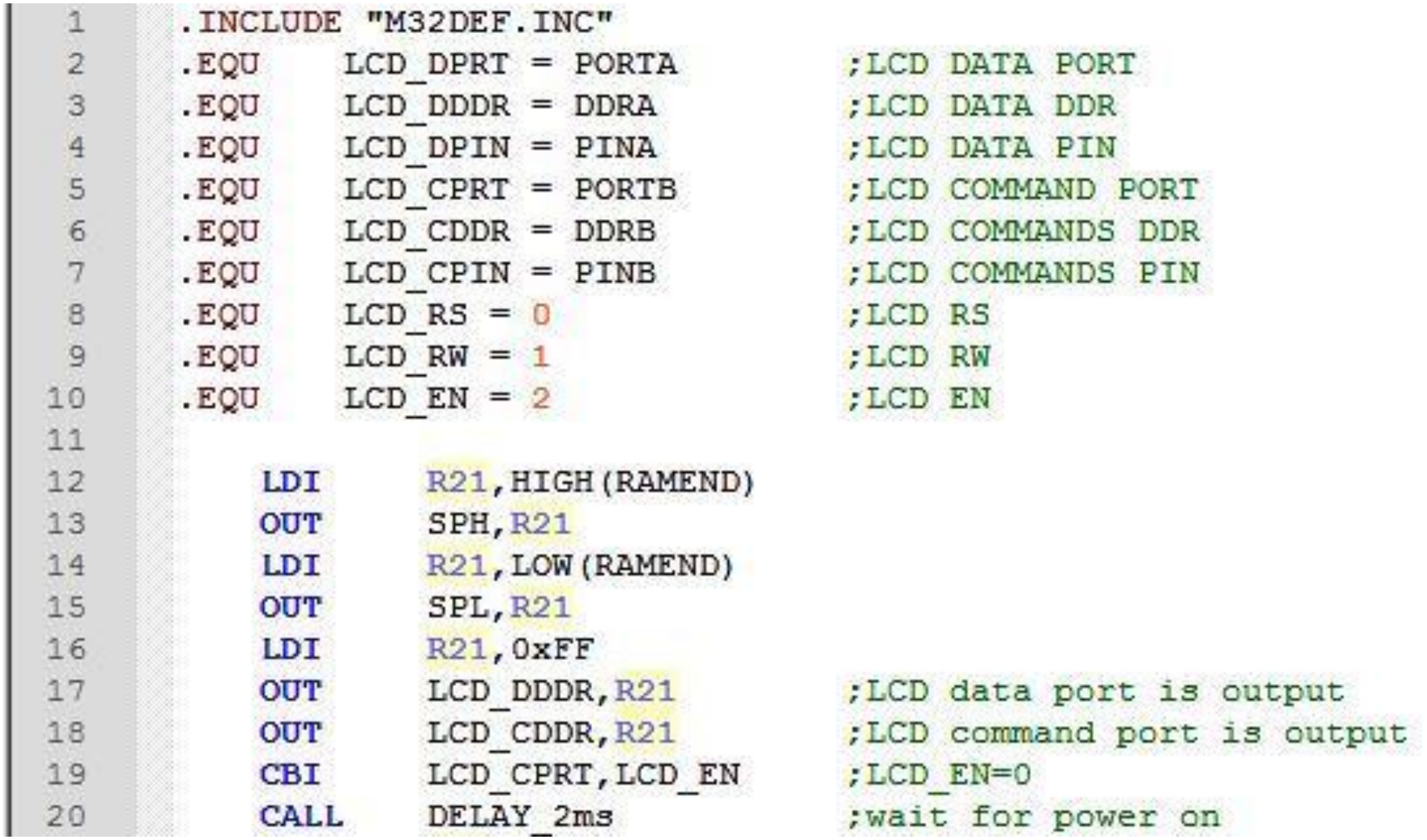


Figure 12-2. LCD Connections for 8-bit Data

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

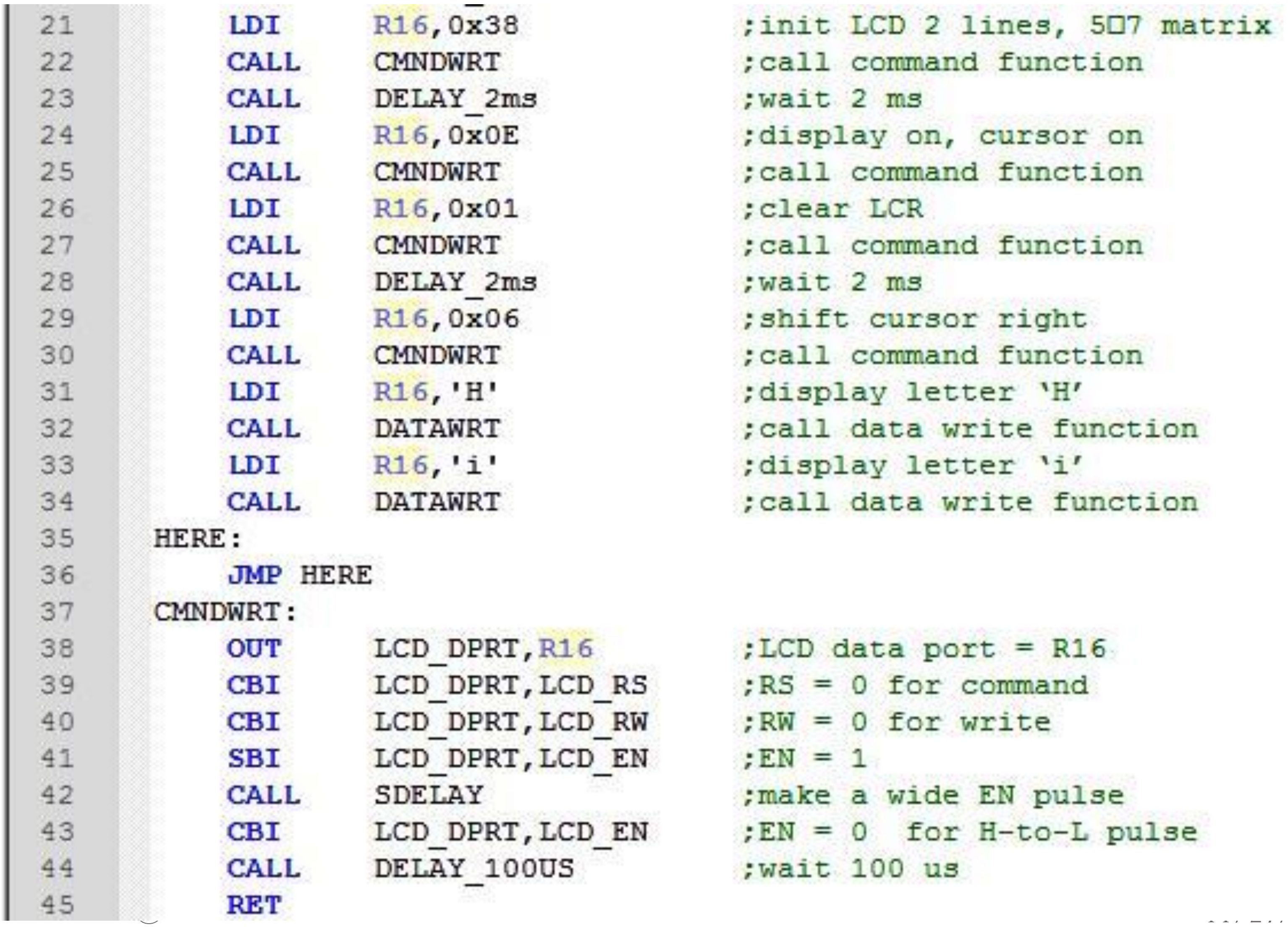
Program 12-1 shows how to write "Hi" on the LCD using 8-bit data. The AVR connection to the LCD for 8-bit data is shown in Figure of previous slide.



```
.INCLUDE "M32DEF.INC"
.EQU     LCD_DPRT = PORTA           ;LCD DATA PORT
.EQU     LCD_DDDR = DDRA            ;LCD DATA DDR
.EQU     LCD_DPIN = PINA            ;LCD DATA PIN
.EQU     LCD_CPRT = PORTB           ;LCD COMMAND PORT
.EQU     LCD_CDDR = DDRB            ;LCD COMMANDS DDR
.EQU     LCD_CPIN = PINB            ;LCD COMMANDS PIN
.EQU     LCD_RS = 0                 ;LCD RS
.EQU     LCD_RW = 1                 ;LCD RW
.EQU     LCD_EN = 2                 ;LCD EN

     LDI     R21,HIGH(RAMEND)
     OUT     SPH,R21
     LDI     R21,LOW(RAMEND)
     OUT     SPL,R21
     LDI     R21,0xFF
     OUT     LCD_DDDR,R21           ;LCD data port is output
     OUT     LCD_CDDR,R21           ;LCD command port is output
     CBI     LCD_CPRT,LCD_EN        ;LCD_EN=0
     CALL    DELAY_2ms              ;wait for power on
```

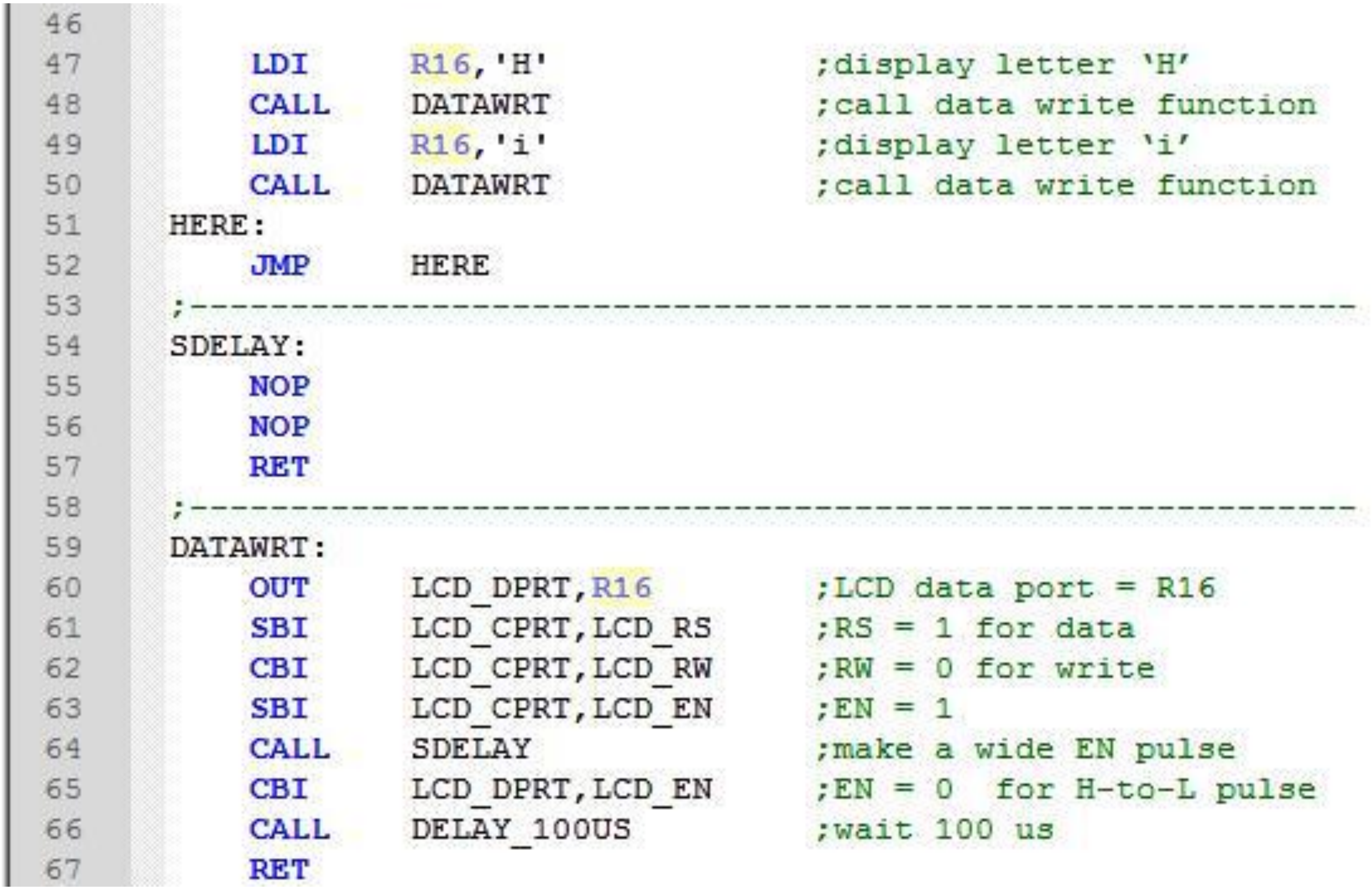
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
        LDI     R16,0x38            ;init LCD 2 lines, 5□7 matrix
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;wait 2 ms
        LDI     R16,0x0E            ;display on, cursor on
        CALL    CMNDWRT             ;call command function
        LDI     R16,0x01            ;clear LCR
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;wait 2 ms
        LDI     R16,0x06            ;shift cursor right
        CALL    CMNDWRT             ;call command function
        LDI     R16,'H'             ;display letter 'H'
        CALL    DATAWRT             ;call data write function
        LDI     R16,'i'             ;display letter 'i'
        CALL    DATAWRT             ;call data write function
HERE:
        JMP HERE
CMNDWRT:
        OUT     LCD_DPRT,R16        ;LCD data port = R16
        CBI     LCD_DPRT,LCD_RS     ;RS = 0 for command
        CBI     LCD_DPRT,LCD_RW     ;RW = 0 for write
        SBI     LCD_DPRT,LCD_EN     ;EN = 1
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_DPRT,LCD_EN     ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US         ;wait 100 us
        RET
```

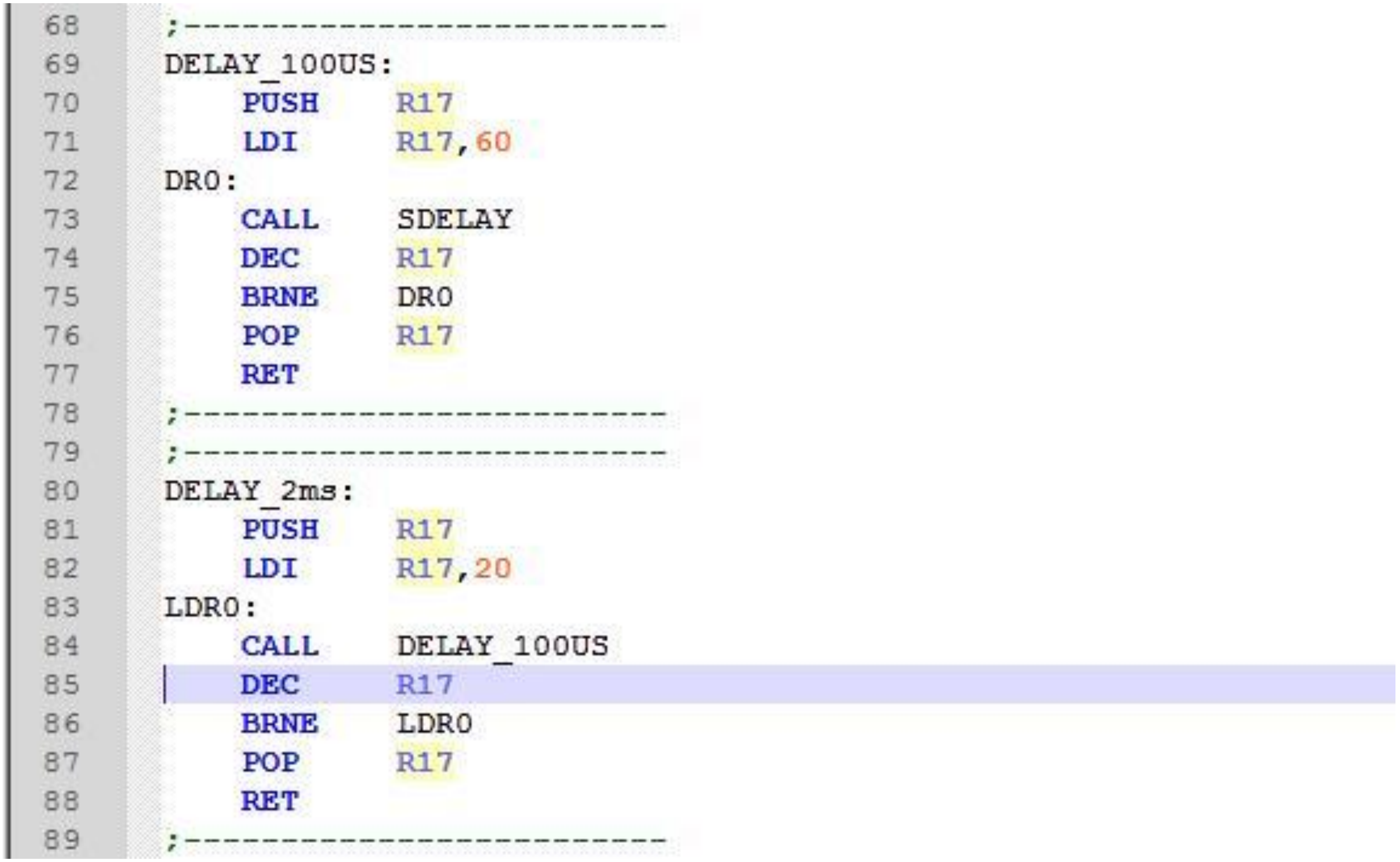
LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING



```
        LDI     R16,'H'             ;display letter 'H'
        CALL    DATAWRT             ;call data write function
        LDI     R16,'i'             ;display letter 'i'
        CALL    DATAWRT             ;call data write function
HERE:
        JMP     HERE
;---------------------------------------------------------
SDELAY:
        NOP
        NOP
        RET
;---------------------------------------------------------
DATAWRT:
        OUT     LCD_DPRT,R16        ;LCD data port = R16
        SBI     LCD_CPRT,LCD_RS     ;RS = 1 for data
        CBI     LCD_CPRT,LCD_RW     ;RW = 0 for write
        SBI     LCD_CPRT,LCD_EN     ;EN = 1
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_DPRT,LCD_EN     ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US         ;wait 100 us
        RET
```

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



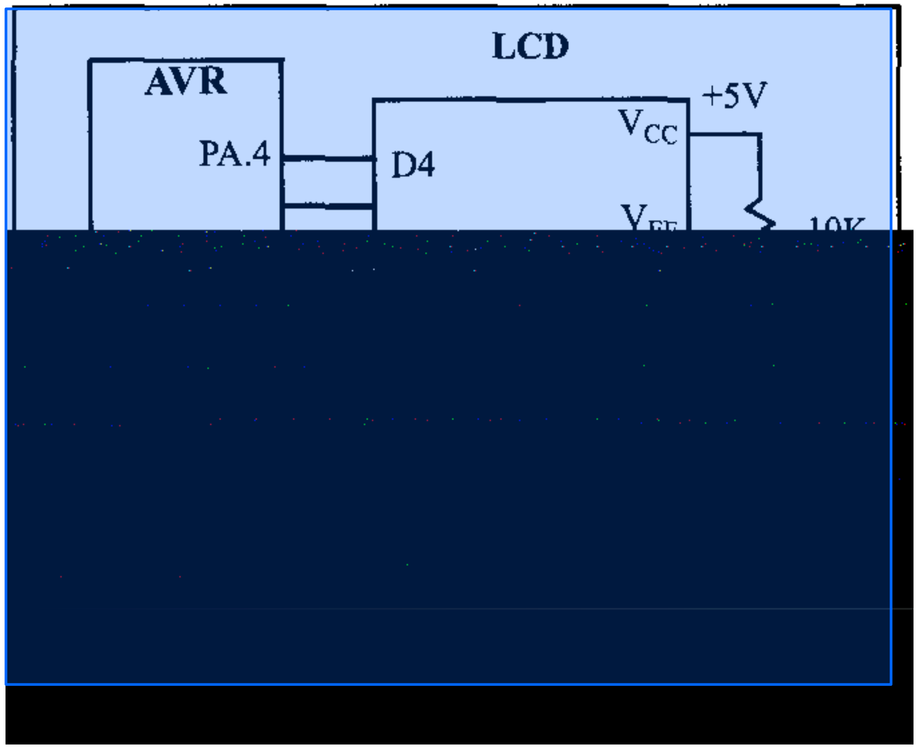
```
;-------------------------
DELAY_100US:
      PUSH    R17
      LDI     R17,60
DR0:
      CALL    SDELAY
      DEC     R17
      BRNE    DR0
      POP     R17
      RET
;-------------------------
;-------------------------
DELAY_2ms:
      PUSH    R17
      LDI     R17,20
LDR0:
      CALL    DELAY_100US
      DEC     R17
      BRNE    LDR0
      POP     R17
      RET
;-------------------------
```

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Sending code or data to the LCD 4 bits at a time

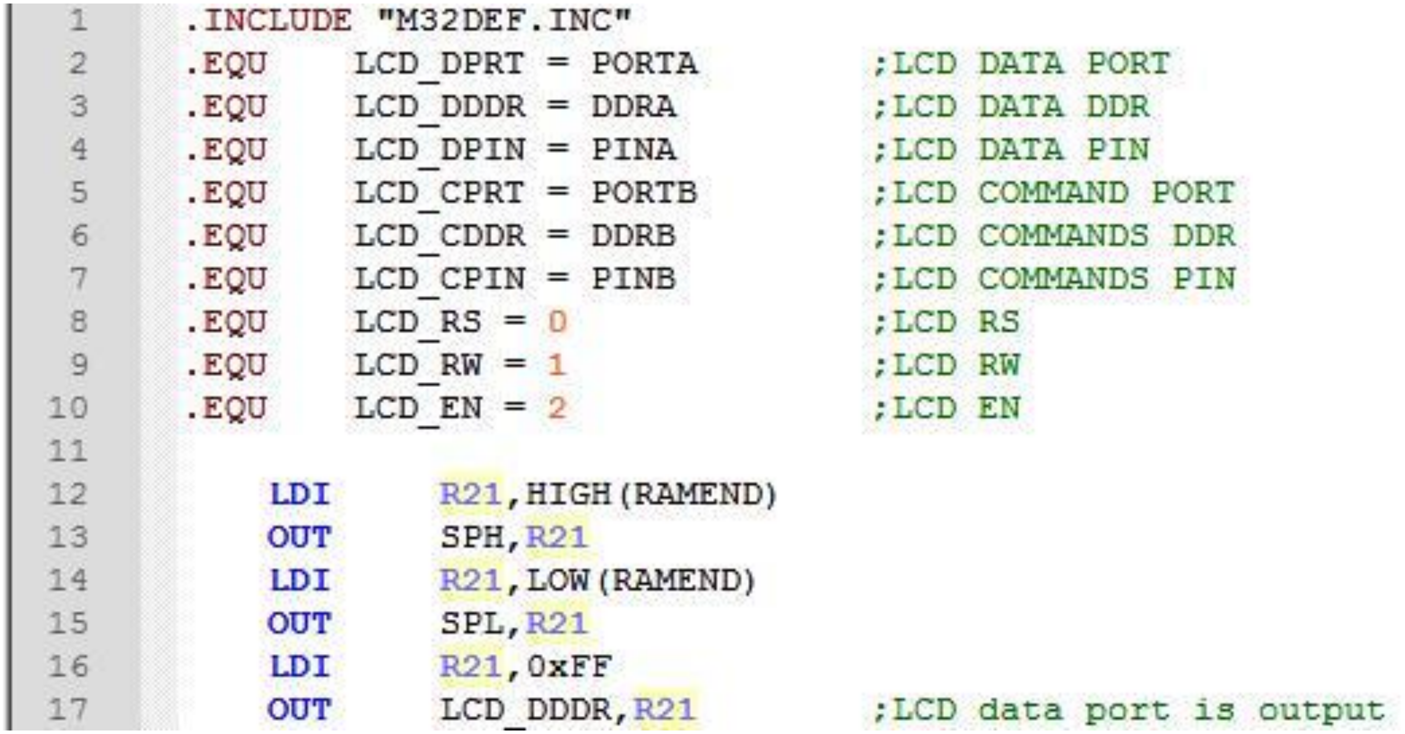
The LCD may be forced into the 4-bit mode as shown in Program 12-2. Notice that its initialization differs from that of the 8-bit mode and that data is sent out on the high nibble of Port A, high nibble first. In 4-bit mode, we initialize the LCD with the series 33,32, and 28 in hex. This represents nibbles 3, 3, 3, and 2, which tells the LCD to go into 4-bit mode.



LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

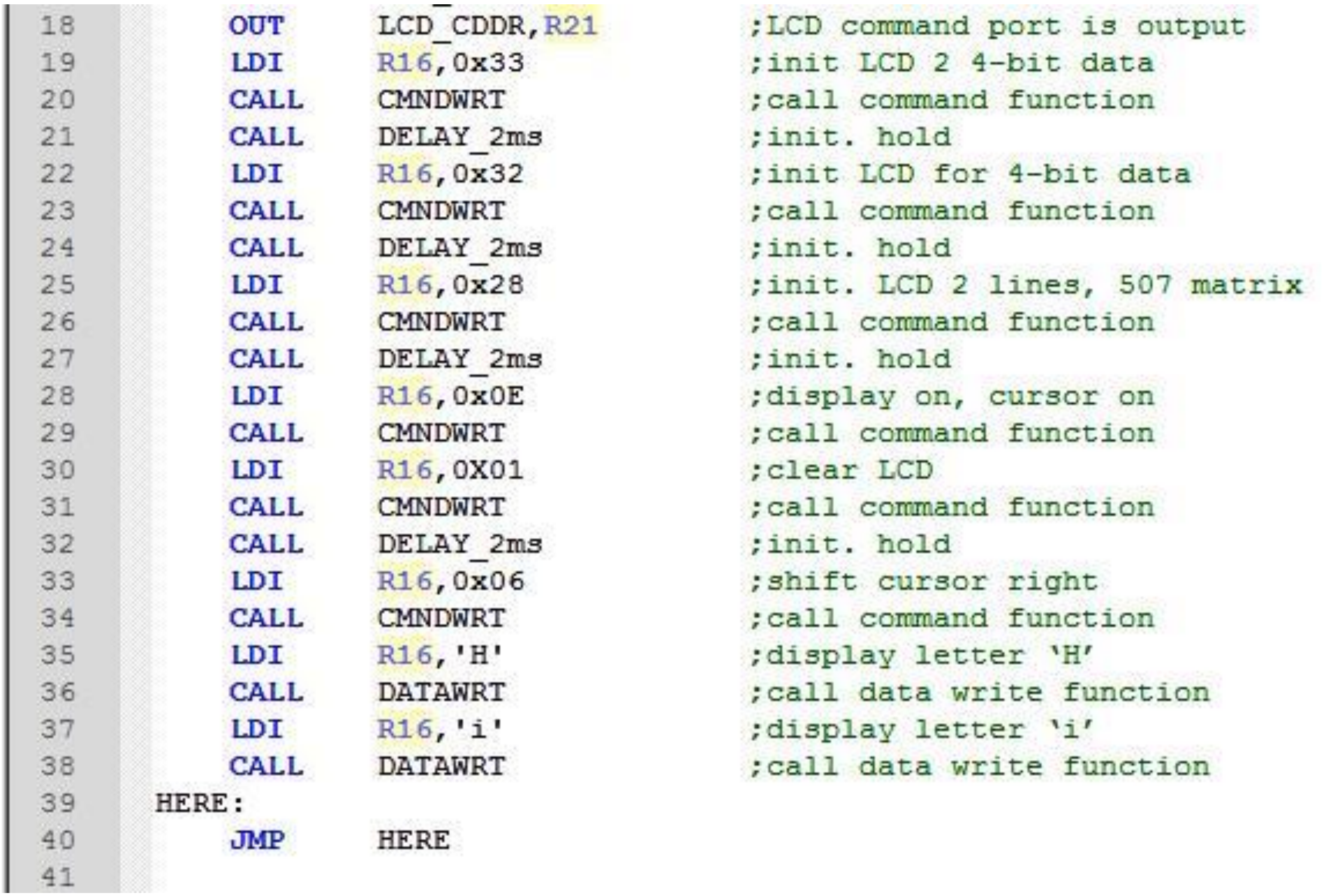
The value $28 initializes the display for 5x7 matrix and 4-bit operation as required by the LCD datasheet. The write routines (CMNDWRT and DATAWRT) send the high nibble first, then swap the low nibble with the high nibble before it is sent to data pins D4-D7.



```
.INCLUDE "M32DEF.INC"
.EQU     LCD_DPRT = PORTA           ;LCD DATA PORT
.EQU     LCD_DDDR = DDRA            ;LCD DATA DDR
.EQU     LCD_DPIN = PINA            ;LCD DATA PIN
.EQU     LCD_CPRT = PORTB           ;LCD COMMAND PORT
.EQU     LCD_CDDR = DDRB            ;LCD COMMANDS DDR
.EQU     LCD_CPIN = PINB            ;LCD COMMANDS PIN
.EQU     LCD_RS = 0                 ;LCD RS
.EQU     LCD_RW = 1                 ;LCD RW
.EQU     LCD_EN = 2                 ;LCD EN

    LDI      R21,HIGH(RAMEND)
    OUT      SPH,R21
    LDI      R21,LOW(RAMEND)
    OUT      SPL,R21
    LDI      R21,0xFF
    OUT      LCD_DDDR,R21           ;LCD data port is output
```

LCD AND KEYBOARD INTERFACING

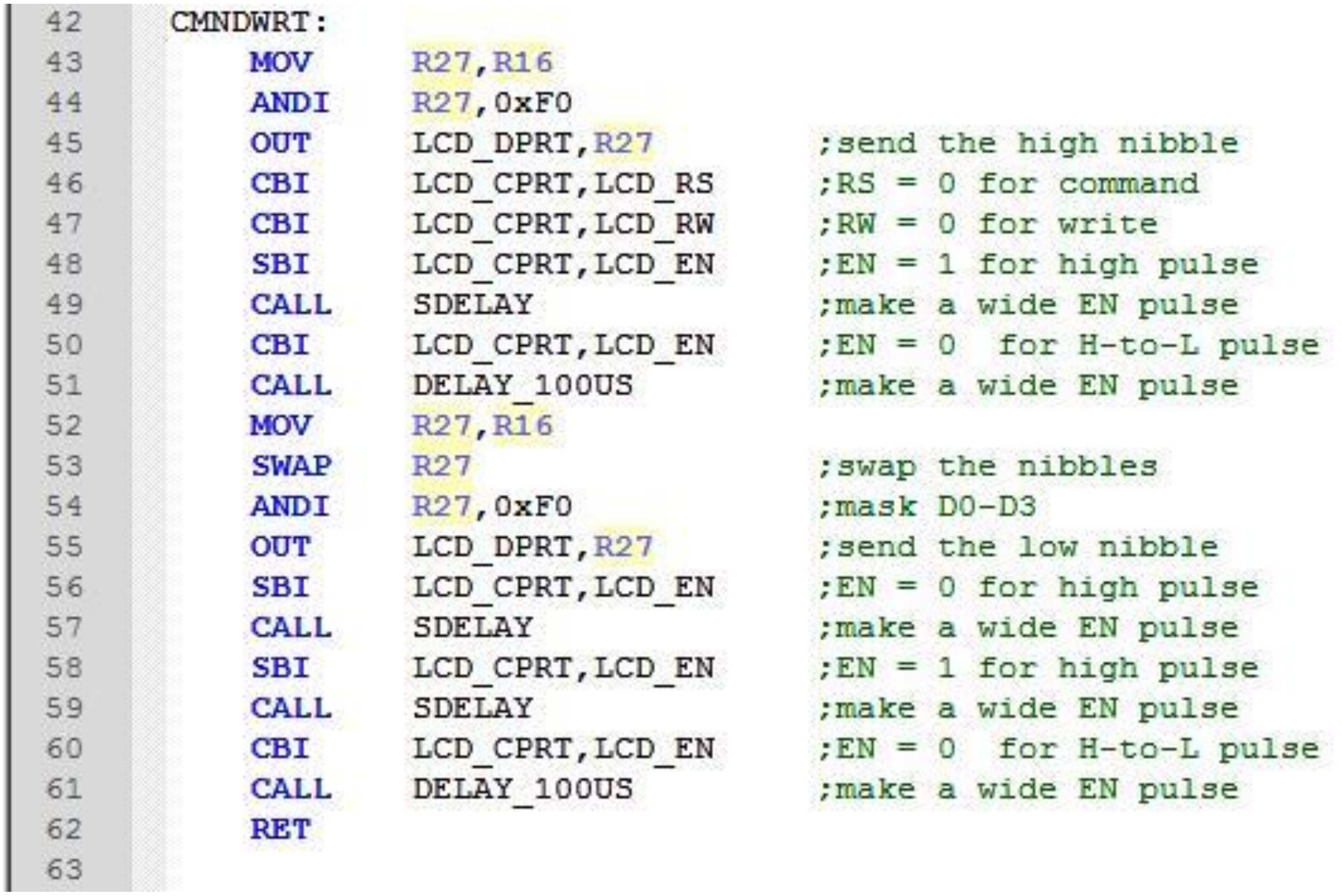
12.1 LCD INTERFACING



```
        OUT     LCD_CDDR,R21        ;LCD command port is output
        LDI     R16,0x33            ;init LCD 2 4-bit data
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;init. hold
        LDI     R16,0x32            ;init LCD for 4-bit data
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;init. hold
        LDI     R16,0x28            ;init. LCD 2 lines, 507 matrix
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;init. hold
        LDI     R16,0x0E            ;display on, cursor on
        CALL    CMNDWRT             ;call command function
        LDI     R16,0X01            ;clear LCD
        CALL    CMNDWRT             ;call command function
        CALL    DELAY_2ms           ;init. hold
        LDI     R16,0x06            ;shift cursor right
        CALL    CMNDWRT             ;call command function
        LDI     R16,'H'             ;display letter 'H'
        CALL    DATAWRT             ;call data write function
        LDI     R16,'i'             ;display letter 'i'
        CALL    DATAWRT             ;call data write function
HERE:
        JMP     HERE
```

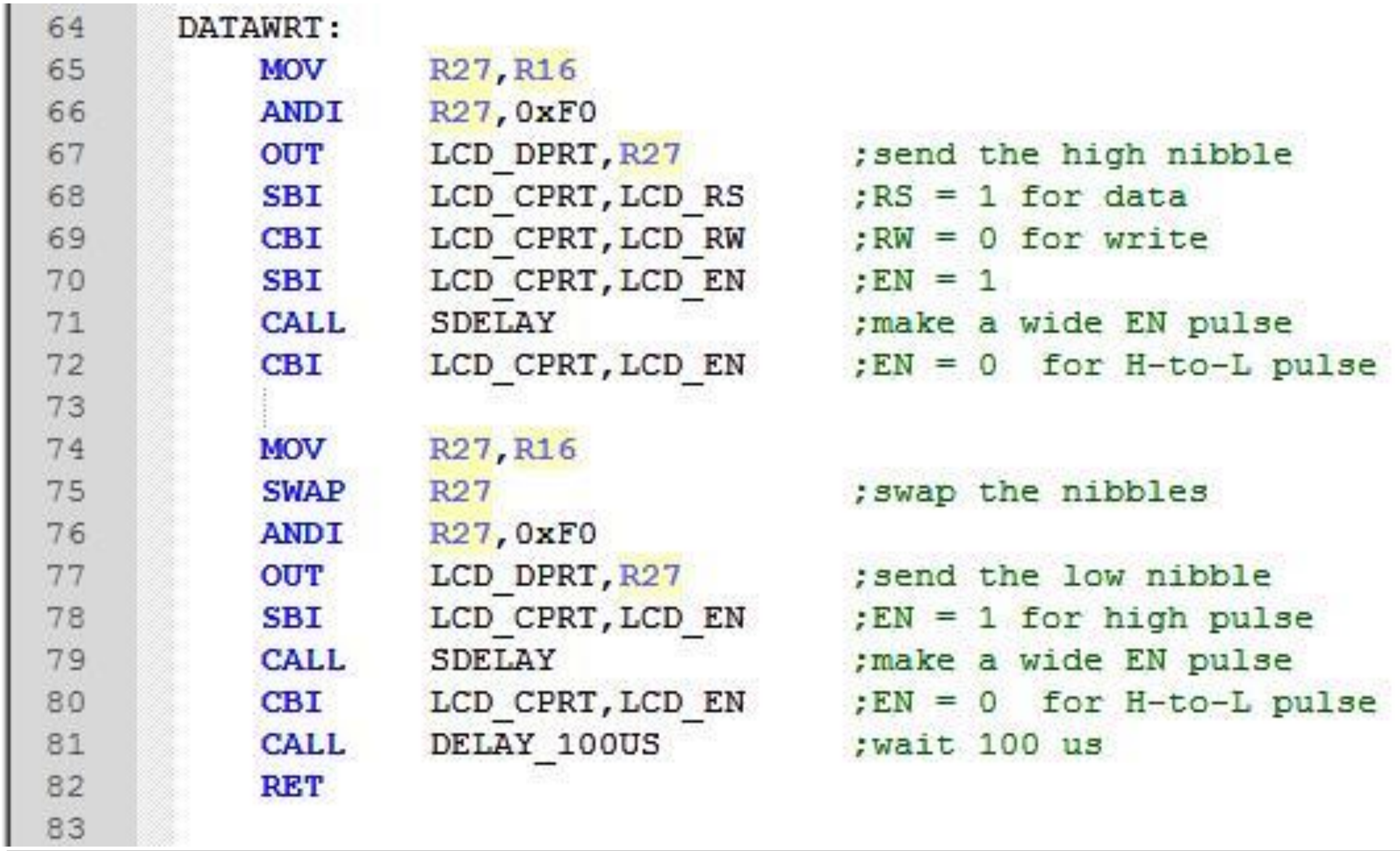
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
CMNDWRT:
        MOV     R27,R16
        ANDI    R27,0xF0
        OUT     LCD_DPRT,R27            ;send the high nibble
        CBI     LCD_CPRT,LCD_RS         ;RS = 0 for command
        CBI     LCD_CPRT,LCD_RW         ;RW = 0 for write
        SBI     LCD_CPRT,LCD_EN         ;EN = 1 for high pulse
        CALL    SDELAY                  ;make a wide EN pulse
        CBI     LCD_CPRT,LCD_EN         ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US             ;make a wide EN pulse
        MOV     R27,R16
        SWAP    R27                     ;swap the nibbles
        ANDI    R27,0xF0                ;mask D0-D3
        OUT     LCD_DPRT,R27            ;send the low nibble
        SBI     LCD_CPRT,LCD_EN         ;EN = 0 for high pulse
        CALL    SDELAY                  ;make a wide EN pulse
        SBI     LCD_CPRT,LCD_EN         ;EN = 1 for high pulse
        CALL    SDELAY                  ;make a wide EN pulse
        CBI     LCD_CPRT,LCD_EN         ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US             ;make a wide EN pulse
        RET
```

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING



```
DATAWRT:
        MOV     R27,R16
        ANDI    R27,0xF0
        OUT     LCD_DPRT,R27        ;send the high nibble
        SBI     LCD_CPRT,LCD_RS     ;RS = 1 for data
        CBI     LCD_CPRT,LCD_RW     ;RW = 0 for write
        SBI     LCD_CPRT,LCD_EN     ;EN = 1
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_CPRT,LCD_EN     ;EN = 0  for H-to-L pulse

        MOV     R27,R16
        SWAP    R27                 ;swap the nibbles
        ANDI    R27,0xF0
        OUT     LCD_DPRT,R27        ;send the low nibble
        SBI     LCD_CPRT,LCD_EN     ;EN = 1 for high pulse
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_CPRT,LCD_EN     ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US         ;wait 100 us
        RET
```

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

Sending code or data to the LCD using a single port

The above code showed how to send commands to the LCD with 4-bit data but we used two different ports for data and commands. In most cases it is preferred to use a single port. Program 12-3 shows Program 12-2 modified to use a single port for LCD interfacing.

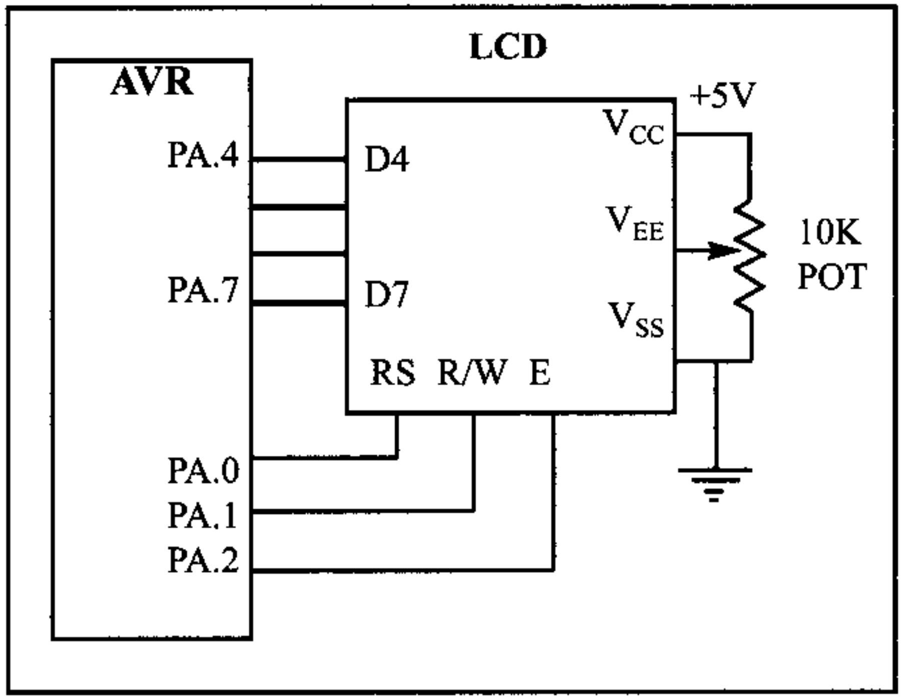


Figure 12-4. LCD Connections Using a Single Port

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

The picture can't be displayed.

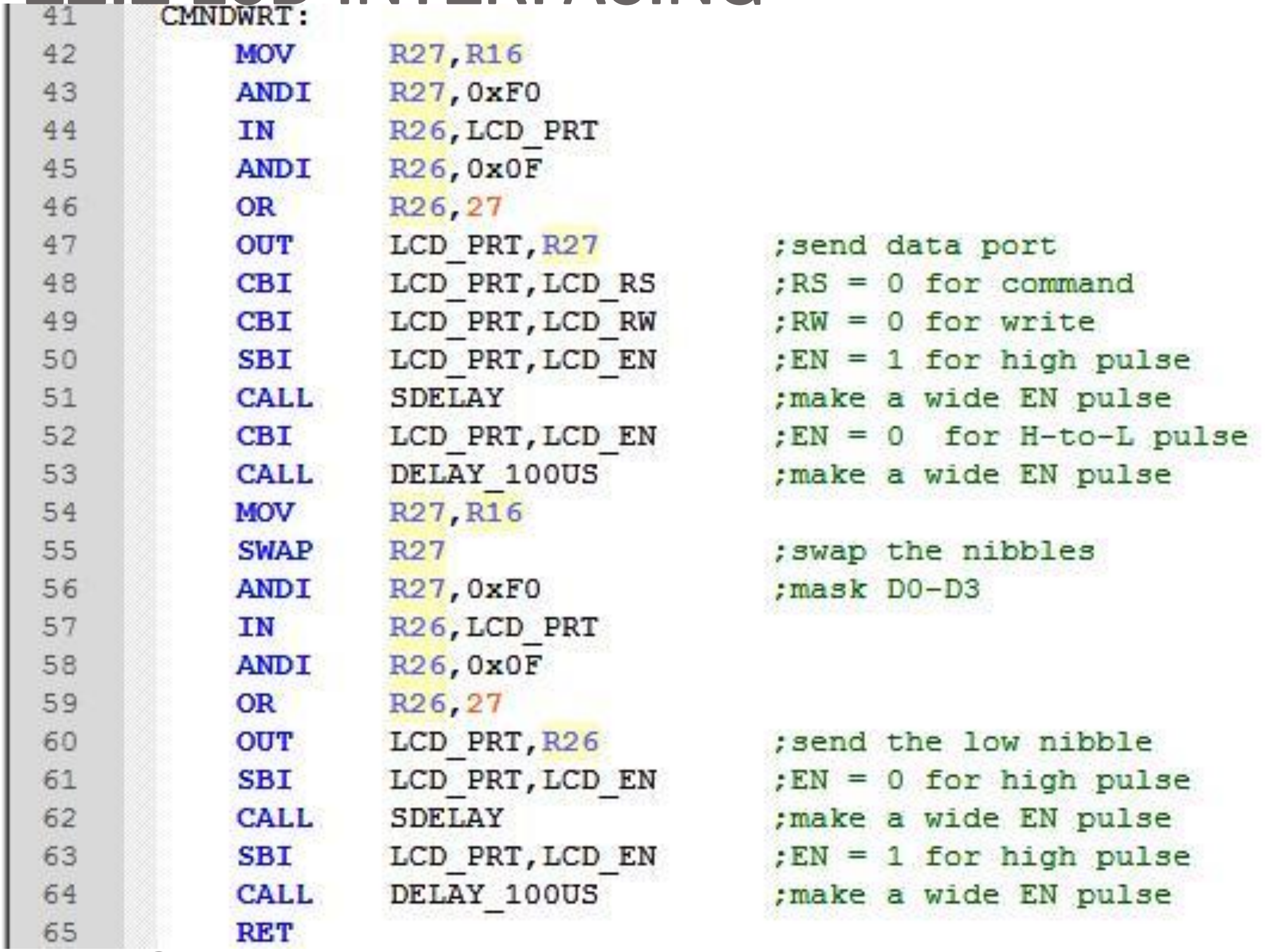
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

The picture can't be displayed.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
CMNDWRT:
        MOV     R27,R16
        ANDI    R27,0xF0
        IN      R26,LCD_PRT
        ANDI    R26,0x0F
        OR      R26,27
        OUT     LCD_PRT,R27         ;send data port
        CBI     LCD_PRT,LCD_RS      ;RS = 0 for command
        CBI     LCD_PRT,LCD_RW      ;RW = 0 for write
        SBI     LCD_PRT,LCD_EN      ;EN = 1 for high pulse
        CALL    SDELAY              ;make a wide EN pulse
        CBI     LCD_PRT,LCD_EN      ;EN = 0  for H-to-L pulse
        CALL    DELAY_100US         ;make a wide EN pulse
        MOV     R27,R16
        SWAP    R27                 ;swap the nibbles
        ANDI    R27,0xF0            ;mask D0-D3
        IN      R26,LCD_PRT
        ANDI    R26,0x0F
        OR      R26,27
        OUT     LCD_PRT,R26         ;send the low nibble
        SBI     LCD_PRT,LCD_EN      ;EN = 0 for high pulse
        CALL    SDELAY              ;make a wide EN pulse
        SBI     LCD_PRT,LCD_EN      ;EN = 1 for high pulse
        CALL    DELAY_100US         ;make a wide EN pulse
        RET
```

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

The picture can't be displayed.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

The picture can't be displayed.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

Sending information to LCD using the LPM instruction

Program 12-4 shows how to use the LPM instruction to send a long string of characters to an LCD. Program 12-4 shows only the main part of the code. The other functions do not change. If you want to use a single port you have to change the port definition in the beginning of the code according to Program 12-2.

The picture can't be displayed.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

The picture can't be displayed.

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

LCD data sheet

Here we deepen your understanding of LCDs by concentrating on two important concepts.

First we will show you the timing diagram of the LCD;
then we will discuss how to put data at any location.

LCD timing diagrams

In Figures 12-5 and 12-6 you can study and contrast the Write timing for the 8-bit and 4-bit modes. Notice that in the 4-bit operating mode, the high nibble is transmitted. Also notice that each nibble is followed by a high-to-low pulse to enable the internal latch of the LCD.

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

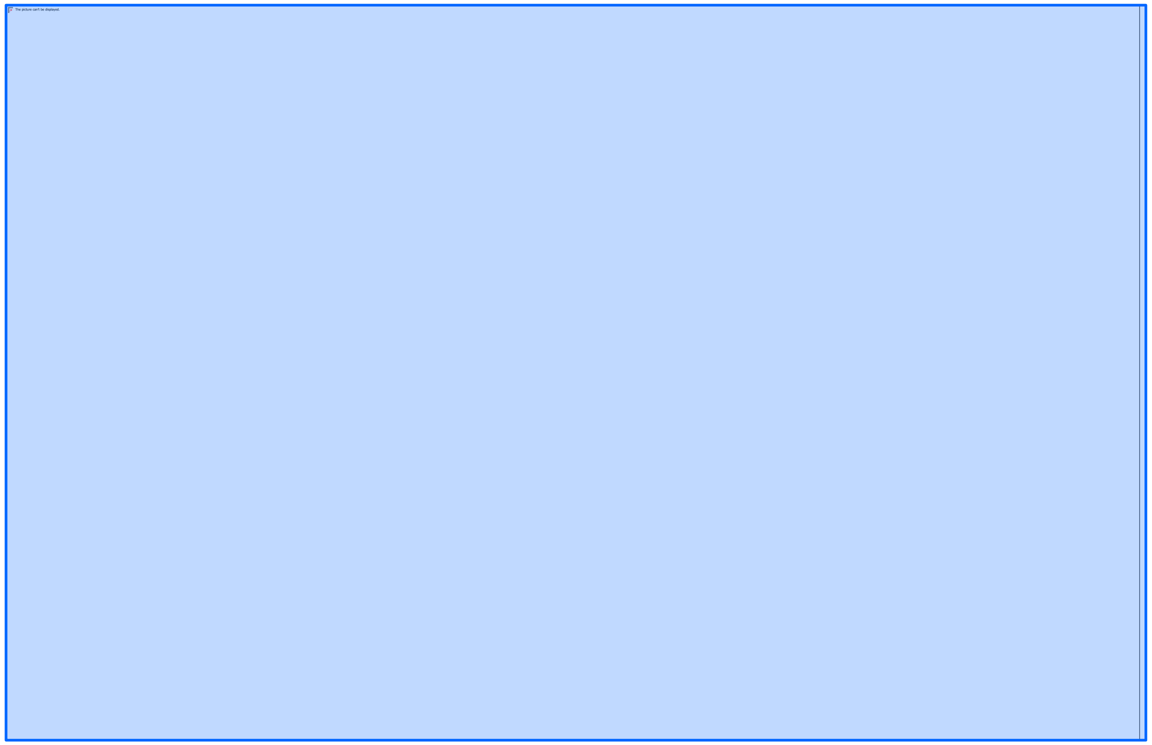
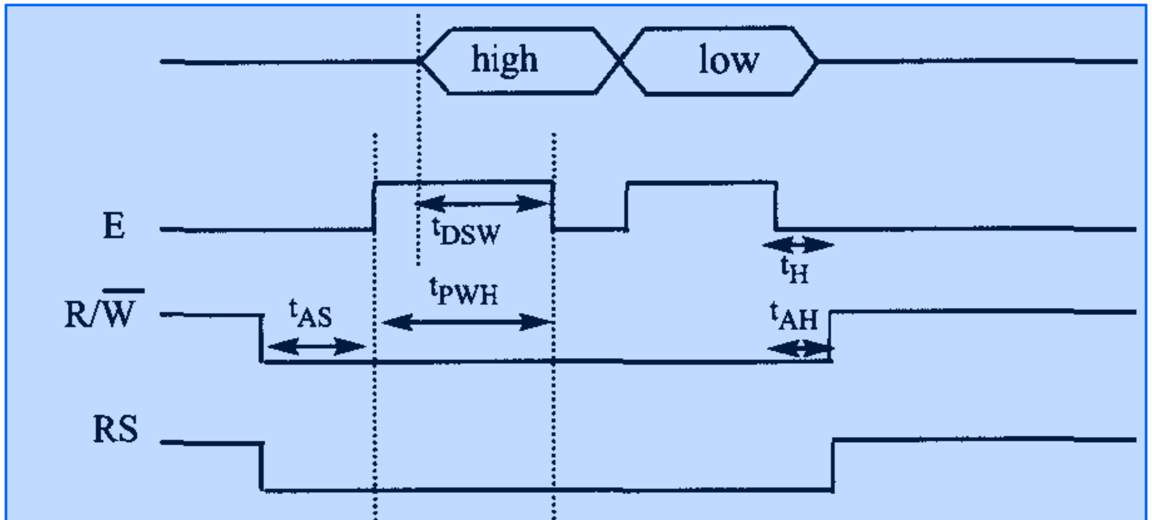


Figure 12-5. LCD Timing for Write (H-to-L for E line)

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

Notice that in the 4-bit operating mode, the high nibble is transmitted. Also notice that each nibble is followed by a high-to-low pulse to enable the internal latch of the LCD.



t_{PWH} = Enable pulse width = 450 ns (minimum)
t_{DSW} = Data setup time = 195 ns (minimum)
t_H = *Data hold time = 10 ns (minimum)*
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

Figure 12-6. LCD Timing for 4-bit Write

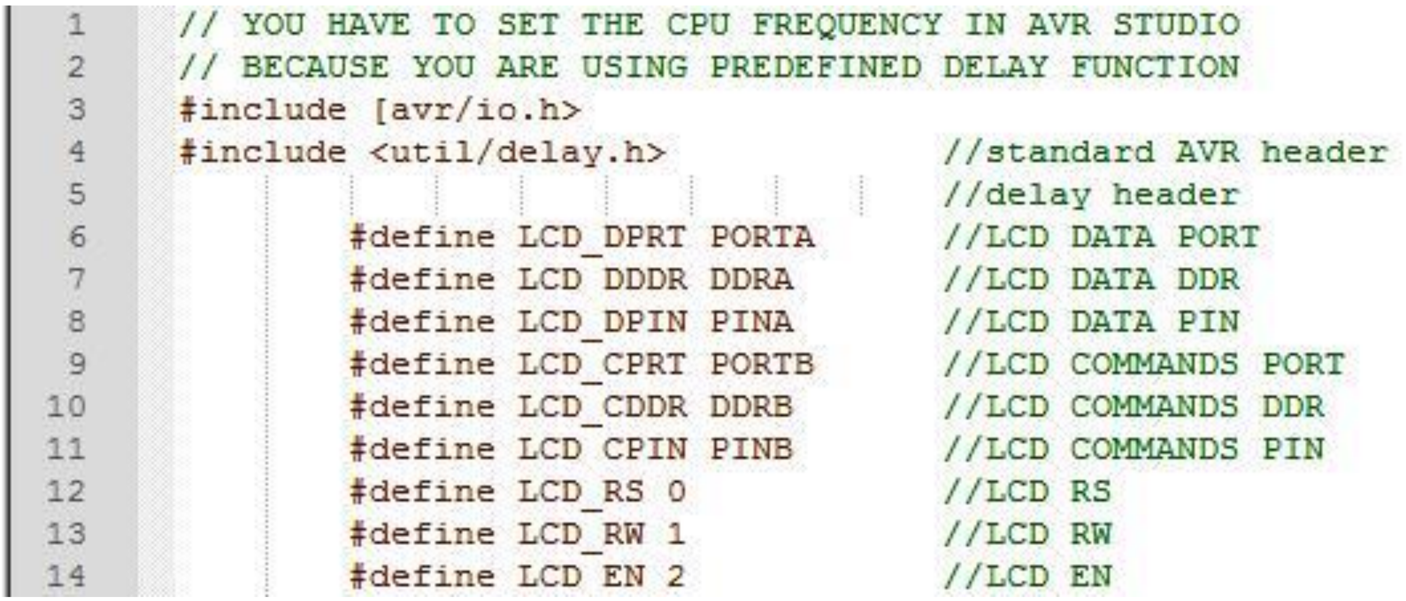
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING

LCD programming in C

Programs 12-5, 12-6, and 12-7 show how to interface an LCD to the AVR using C programming. The codes are modular to improve code clarity.

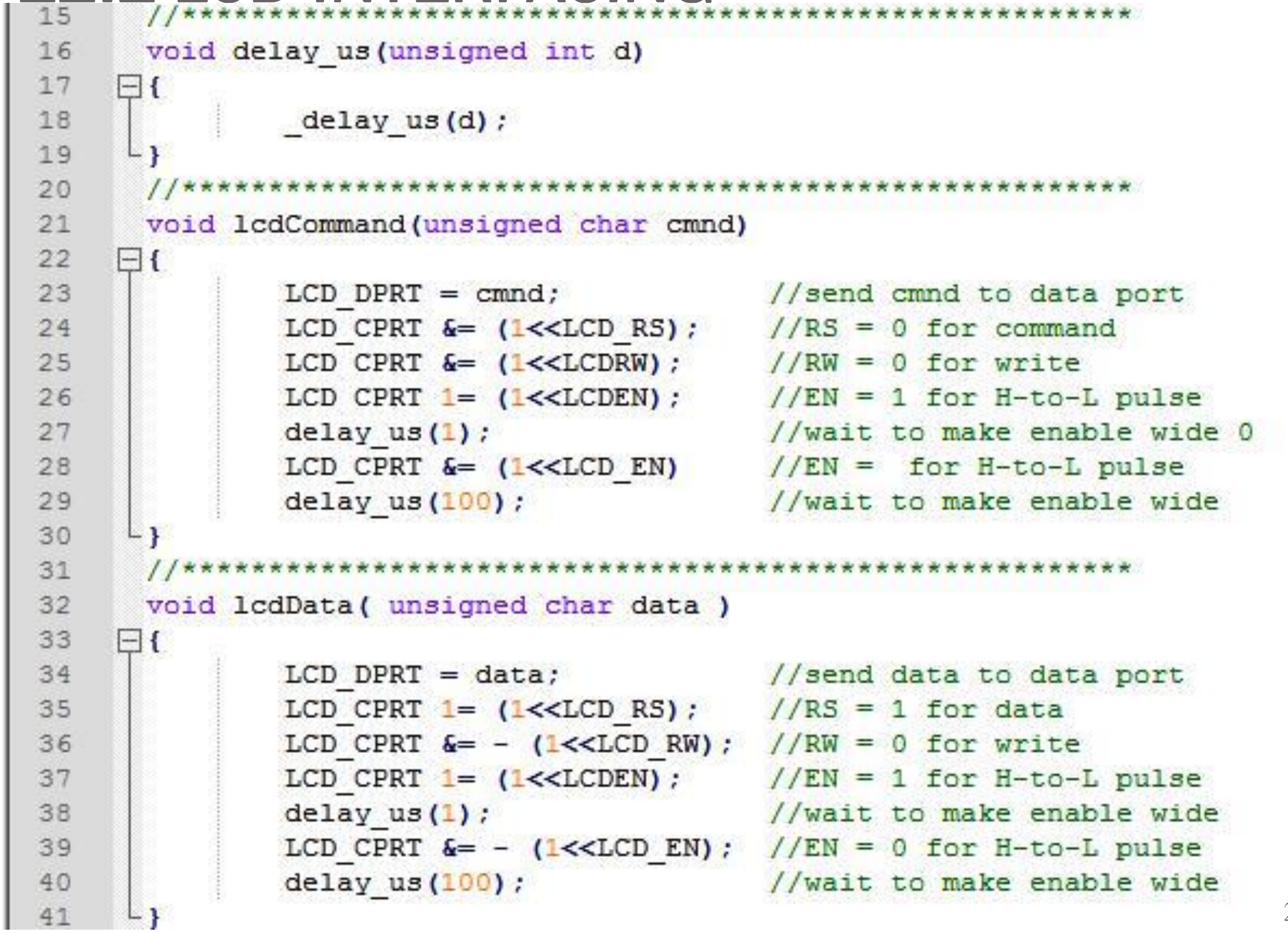
Program 12-5 shows how to use 8-bit data to interface an LCD to the AVR in C language.



```
// YOU HAVE TO SET THE CPU FREQUENCY IN AVR STUDIO
// BECAUSE YOU ARE USING PREDEFINED DELAY FUNCTION
#include [avr/io.h>
#include <util/delay.h>                //standard AVR header
                                       //delay header
        #define LCD_DPRT PORTA         //LCD DATA PORT
        #define LCD_DDDR DDRA          //LCD DATA DDR
        #define LCD_DPIN PINA          //LCD DATA PIN
        #define LCD_CPRT PORTB         //LCD COMMANDS PORT
        #define LCD_CDDR DDRB          //LCD COMMANDS DDR
        #define LCD_CPIN PINB          //LCD COMMANDS PIN
        #define LCD_RS 0               //LCD RS
        #define LCD_RW 1               //LCD RW
        #define LCD_EN 2               //LCD EN
```

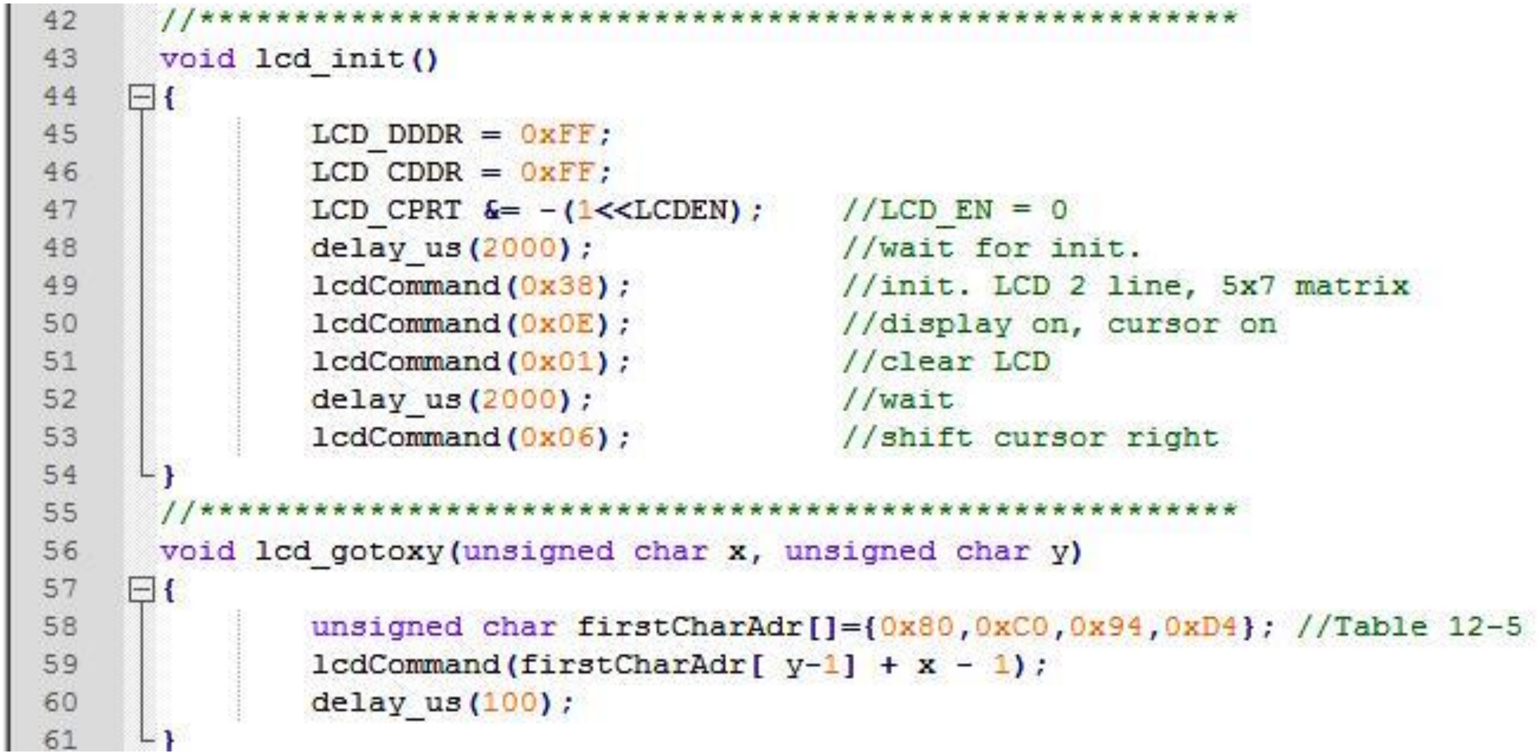
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
//****************************************************
void delay_us(unsigned int d)
{
        _delay_us(d);
}
//****************************************************
void lcdCommand(unsigned char cmnd)
{
        LCD_DPRT = cmnd;                //send cmnd to data port
        LCD_CPRT &= (1<<LCD_RS);        //RS = 0 for command
        LCD_CPRT &= (1<<LCDRW);         //RW = 0 for write
        LCD_CPRT 1= (1<<LCDEN);         //EN = 1 for H-to-L pulse
        delay_us(1);                    //wait to make enable wide 0
        LCD_CPRT &= (1<<LCD_EN)         //EN =  for H-to-L pulse
        delay_us(100);                  //wait to make enable wide
}
//****************************************************
void lcdData( unsigned char data )
{
        LCD_DPRT = data;                //send data to data port
        LCD_CPRT 1= (1<<LCD_RS);        //RS = 1 for data
        LCD_CPRT &= - (1<<LCD_RW);      //RW = 0 for write
        LCD_CPRT 1= (1<<LCDEN);         //EN = 1 for H-to-L pulse
        delay_us(1);                    //wait to make enable wide
        LCD_CPRT &= - (1<<LCD_EN);      //EN = 0 for H-to-L pulse
        delay_us(100);                  //wait to make enable wide
}
```

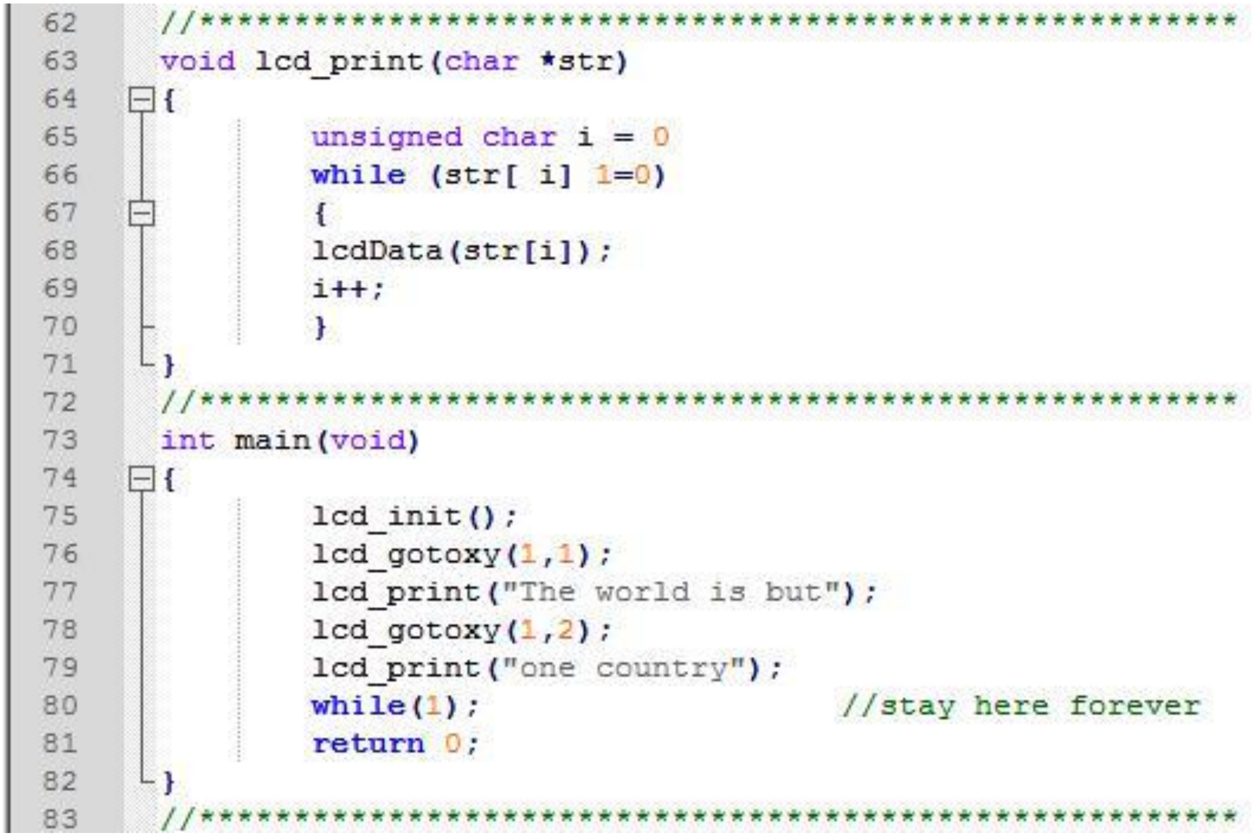
LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING



```c
//*******************************************************
void lcd_init()
{
        LCD_DDDR = 0xFF;
        LCD_CDDR = 0xFF;
        LCD_CPRT &= ~(1<<LCDEN);    //LCD_EN = 0
        delay_us(2000);             //wait for init.
        lcdCommand(0x38);           //init. LCD 2 line, 5x7 matrix
        lcdCommand(0x0E);           //display on, cursor on
        lcdCommand(0x01);           //clear LCD
        delay_us(2000);             //wait
        lcdCommand(0x06);           //shift cursor right
}
//*******************************************************
void lcd_gotoxy(unsigned char x, unsigned char y)
{
        unsigned char firstCharAdr[]={0x80,0xC0,0x94,0xD4}; //Table 12-5
        lcdCommand(firstCharAdr[ y-1] + x - 1);
        delay_us(100);
}
```

LCD AND KEYBOARD INTERFACING

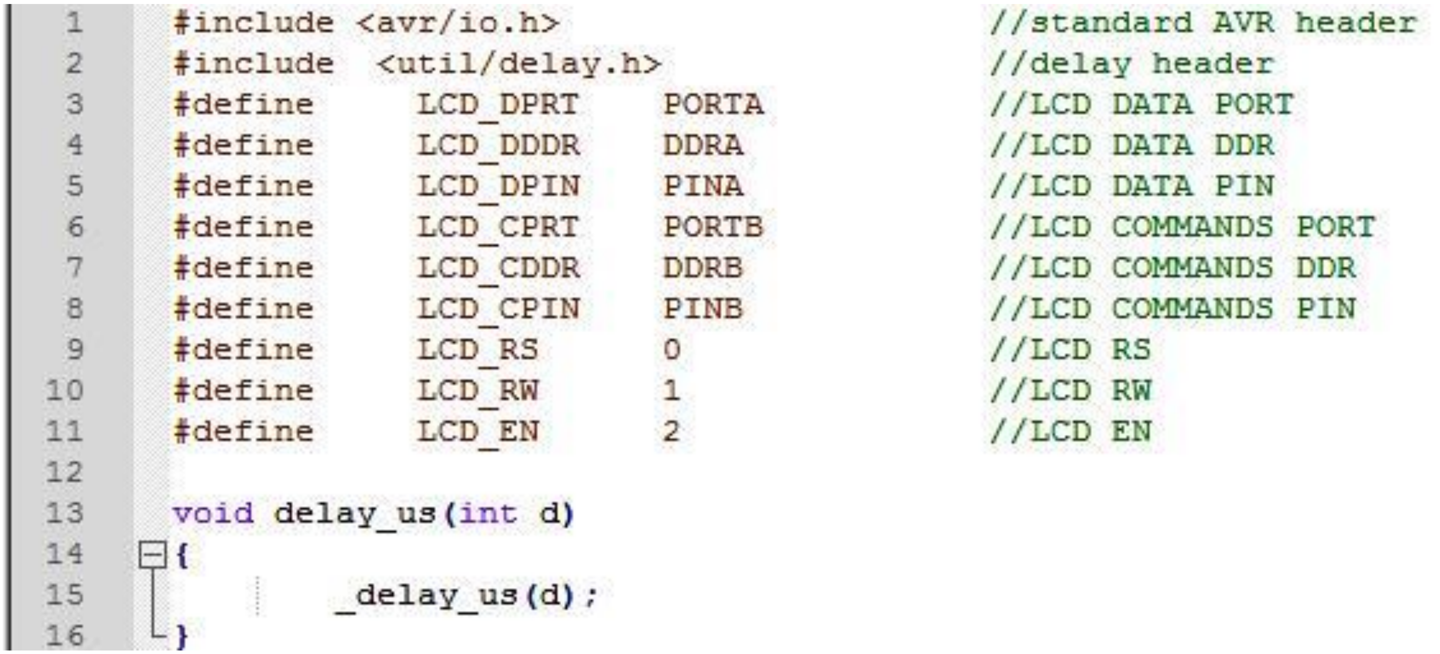
12.1 LCD INTERFACING



```
//*****************************************************
void lcd_print(char *str)
{
        unsigned char i = 0
        while (str[ i] 1=0)
        {
        lcdData(str[i]);
        i++;
        }
}
//*****************************************************
int main(void)
{
        lcd_init();
        lcd_gotoxy(1,1);
        lcd_print("The world is but");
        lcd_gotoxy(1,2);
        lcd_print("one country");
        while(1);                   //stay here forever
        return 0;
}
//*****************************************************
```

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

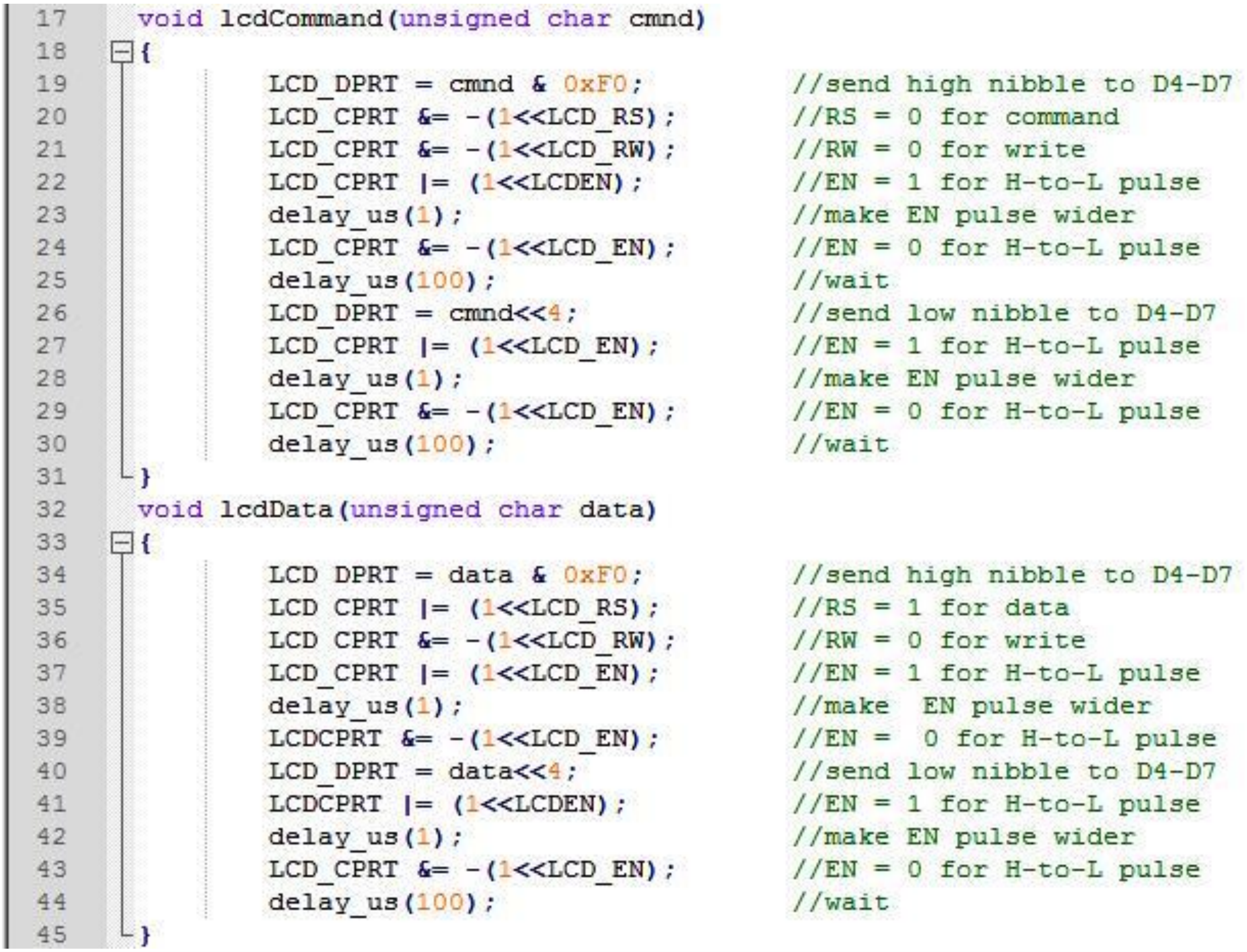
Program 12-6 shows how to use 4-bit data to interface an LCD to the AVR in C language.



```
#include <avr/io.h>                                       //standard AVR header
#include  <util/delay.h>                                  //delay header
#define       LCD_DPRT     PORTA                          //LCD DATA PORT
#define       LCD_DDDR     DDRA                           //LCD DATA DDR
#define       LCD_DPIN     PINA                           //LCD DATA PIN
#define       LCD_CPRT     PORTB                          //LCD COMMANDS PORT
#define       LCD_CDDR     DDRB                           //LCD COMMANDS DDR
#define       LCD_CPIN     PINB                           //LCD COMMANDS PIN
#define       LCD_RS       0                              //LCD RS
#define       LCD_RW       1                              //LCD RW
#define       LCD_EN       2                              //LCD EN

void delay_us(int d)
{
        _delay_us(d);
}
```

LCD AND KEYBOARD INTERFACING

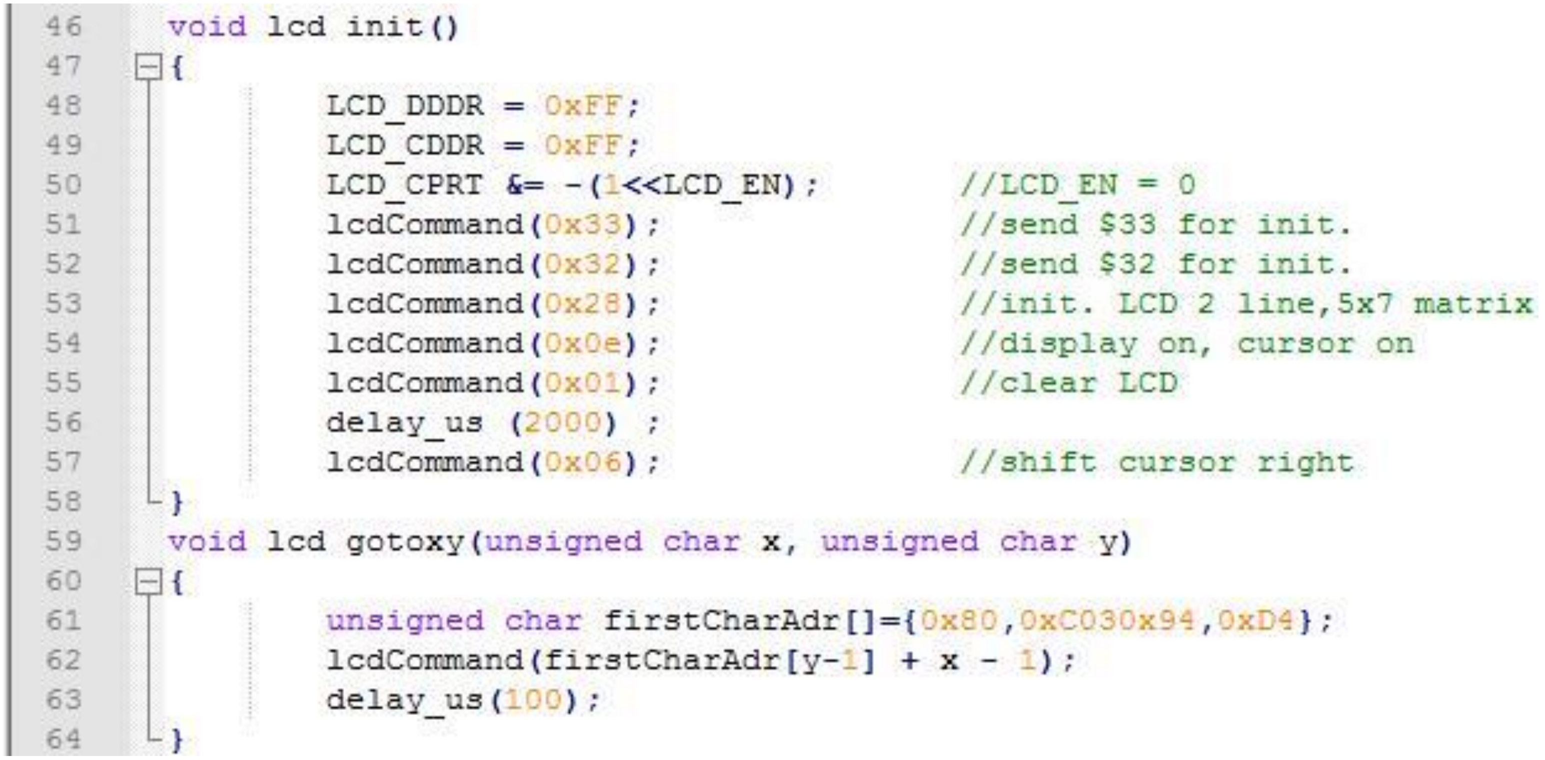
12.1 LCD INTERFACING



```c
void lcdCommand(unsigned char cmnd)
{
        LCD_DPRT = cmnd & 0xF0;         //send high nibble to D4-D7
        LCD_CPRT &= ~(1<<LCD_RS);       //RS = 0 for command
        LCD_CPRT &= ~(1<<LCD_RW);       //RW = 0 for write
        LCD_CPRT |= (1<<LCDEN);         //EN = 1 for H-to-L pulse
        delay_us(1);                    //make EN pulse wider
        LCD_CPRT &= ~(1<<LCD_EN);       //EN = 0 for H-to-L pulse
        delay_us(100);                  //wait
        LCD_DPRT = cmnd<<4;             //send low nibble to D4-D7
        LCD_CPRT |= (1<<LCD_EN);        //EN = 1 for H-to-L pulse
        delay_us(1);                    //make EN pulse wider
        LCD_CPRT &= ~(1<<LCD_EN);       //EN = 0 for H-to-L pulse
        delay_us(100);                  //wait
}
void lcdData(unsigned char data)
{
        LCD DPRT = data & 0xF0;         //send high nibble to D4-D7
        LCD CPRT |= (1<<LCD_RS);        //RS = 1 for data
        LCD CPRT &= ~(1<<LCD_RW);       //RW = 0 for write
        LCD_CPRT |= (1<<LCD_EN);        //EN = 1 for H-to-L pulse
        delay_us(1);                    //make  EN pulse wider
        LCDCPRT &= ~(1<<LCD_EN);        //EN =  0 for H-to-L pulse
        LCD_DPRT = data<<4;             //send low nibble to D4-D7
        LCDCPRT |= (1<<LCDEN);          //EN = 1 for H-to-L pulse
        delay_us(1);                    //make EN pulse wider
        LCD_CPRT &= ~(1<<LCD_EN);       //EN = 0 for H-to-L pulse
        delay_us(100);                  //wait
}
```

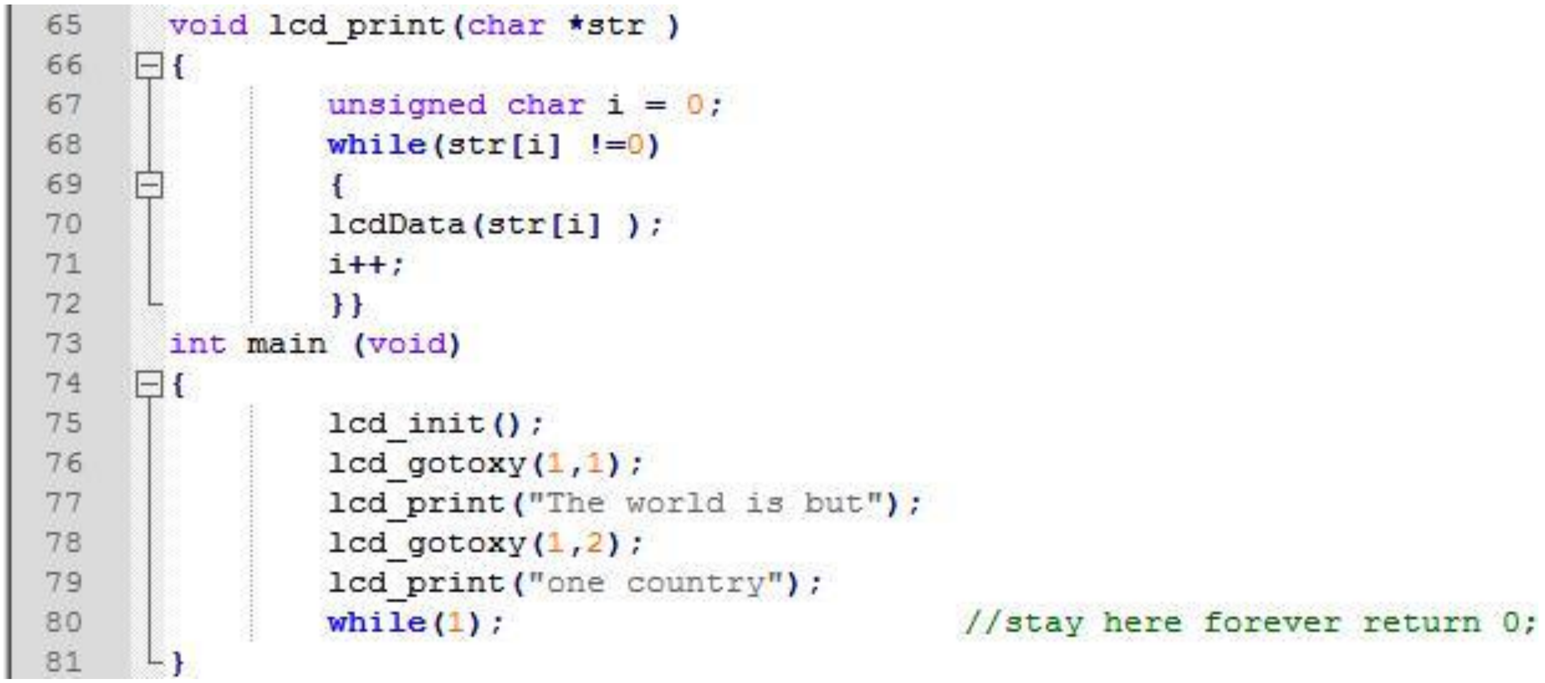
LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
void lcd_init()
{
        LCD_DDDR = 0xFF;
        LCD_CDDR = 0xFF;
        LCD_CPRT &= ~(1<<LCD_EN);         //LCD_EN = 0
        lcdCommand(0x33);                 //send $33 for init.
        lcdCommand(0x32);                 //send $32 for init.
        lcdCommand(0x28);                 //init. LCD 2 line,5x7 matrix
        lcdCommand(0x0e);                 //display on, cursor on
        lcdCommand(0x01);                 //clear LCD
        delay_us (2000) ;
        lcdCommand(0x06);                 //shift cursor right
}
void lcd_gotoxy(unsigned char x, unsigned char y)
{
        unsigned char firstCharAdr[]={0x80,0xC030x94,0xD4};
        lcdCommand(firstCharAdr[y-1] + x - 1);
        delay_us(100);
}
```

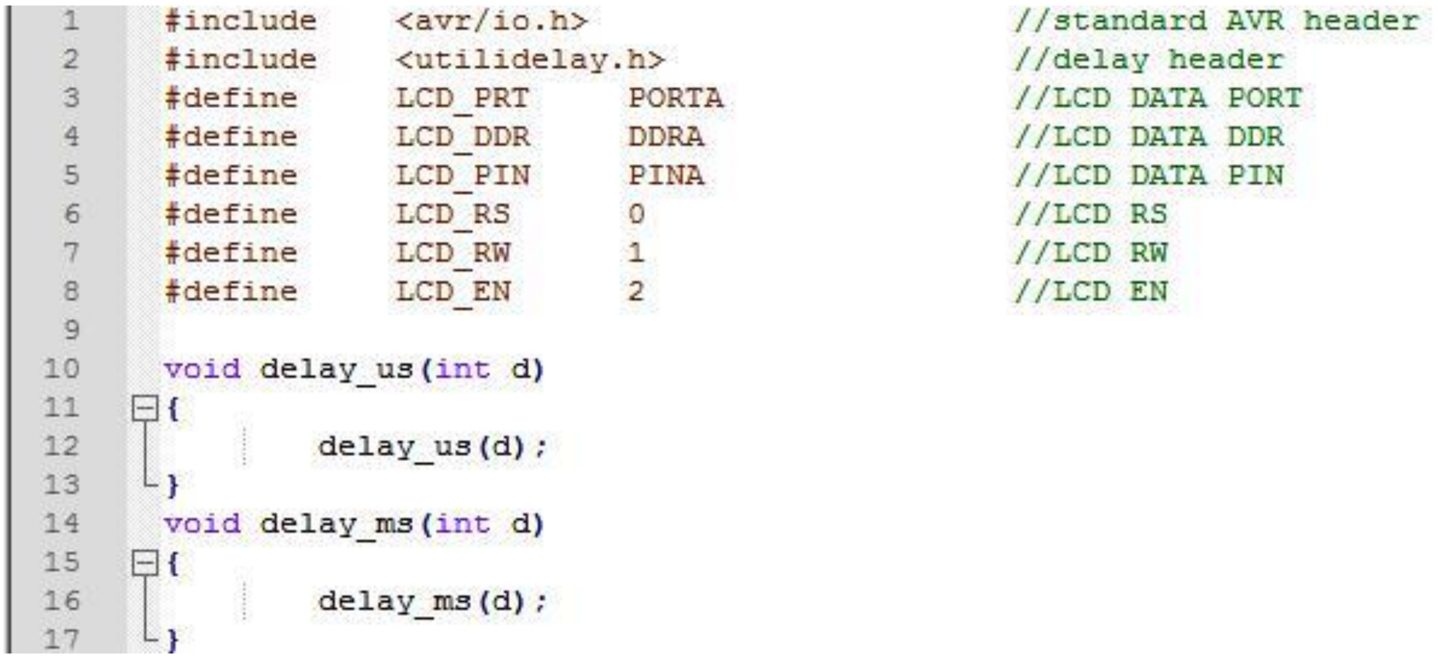
LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING



```c
void lcd_print(char *str )
{
        unsigned char i = 0;
        while(str[i] !=0)
        {
        lcdData(str[i] );
        i++;
        }}
int main (void)
{
        lcd_init();
        lcd_gotoxy(1,1);
        lcd_print("The world is but");
        lcd_gotoxy(1,2);
        lcd_print("one country");
        while(1);                         //stay here forever return 0;
}
```

LCD AND KEYBOARD INTERFACING
12.1 LCD INTERFACING

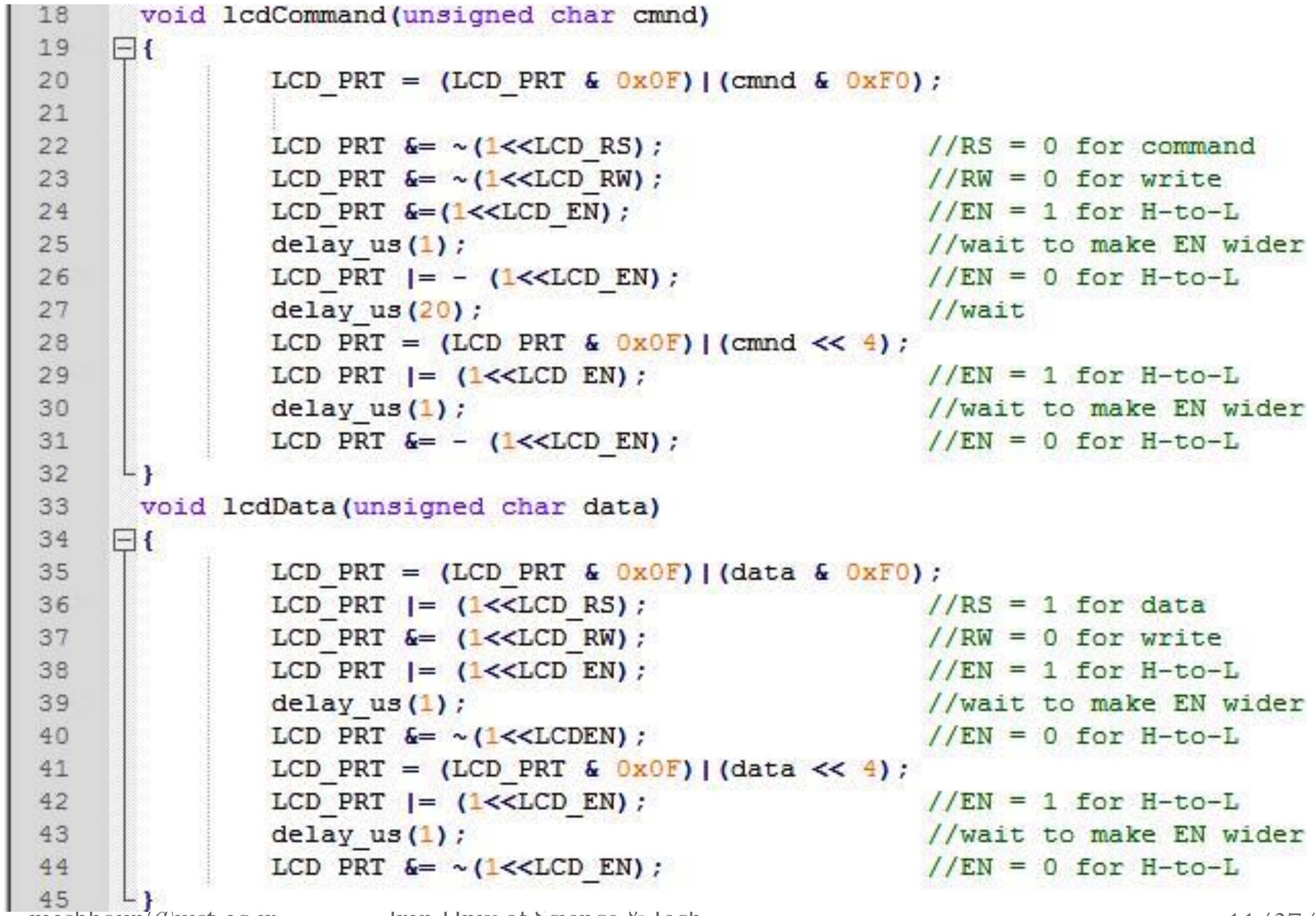
Program 12-7 shows how to use 4-bit data to interface an LCD to the AVR in C language. It uses only a single port. Also there are some useful functions to print a string (array of chars) or to move the cursor to a specific location.



```
#include    <avr/io.h>                          //standard AVR header
#include    <utilidelay.h>                      //delay header
#define     LCD_PRT     PORTA                   //LCD DATA PORT
#define     LCD_DDR     DDRA                    //LCD DATA DDR
#define     LCD_PIN     PINA                    //LCD DATA PIN
#define     LCD_RS      0                       //LCD RS
#define     LCD_RW      1                       //LCD RW
#define     LCD_EN      2                       //LCD EN

void delay_us(int d)
{
        delay_us(d);
}
void delay_ms(int d)
{
        delay_ms(d);
}
```

LCD AND KEYBOARD INTERFACING

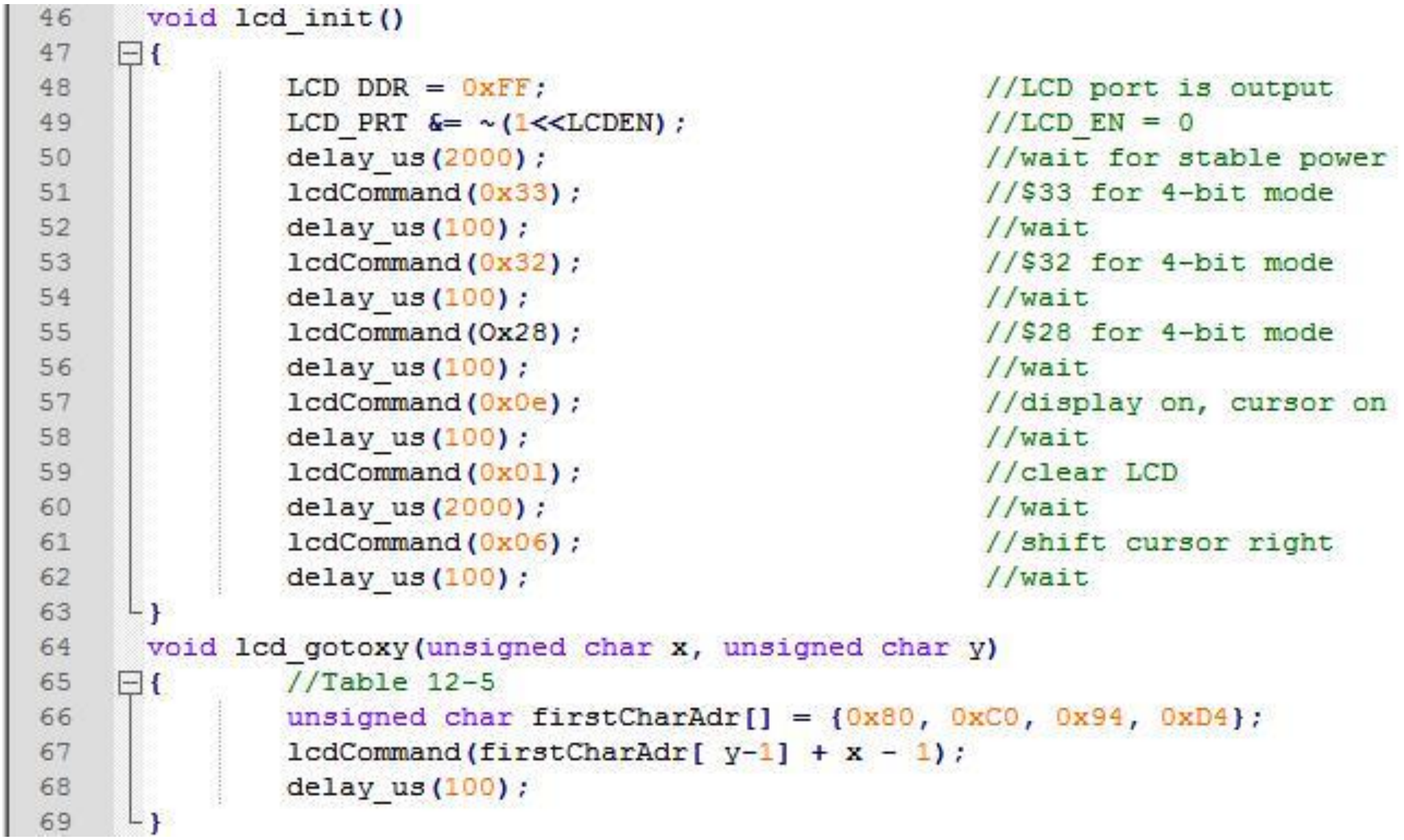
12.1 LCD INTERFACING



```
void lcdCommand(unsigned char cmnd)
{
        LCD_PRT = (LCD_PRT & 0x0F)|(cmnd & 0xF0);

        LCD_PRT &= ~(1<<LCD_RS);                        //RS = 0 for command
        LCD_PRT &= ~(1<<LCD_RW);                        //RW = 0 for write
        LCD_PRT &=(1<<LCD_EN);                          //EN = 1 for H-to-L
        delay_us(1);                                    //wait to make EN wider
        LCD_PRT |= - (1<<LCD_EN);                       //EN = 0 for H-to-L
        delay_us(20);                                   //wait
        LCD_PRT = (LCD_PRT & 0x0F)|(cmnd << 4);
        LCD_PRT |= (1<<LCD_EN);                         //EN = 1 for H-to-L
        delay_us(1);                                    //wait to make EN wider
        LCD_PRT &= - (1<<LCD_EN);                       //EN = 0 for H-to-L
}
void lcdData(unsigned char data)
{
        LCD_PRT = (LCD_PRT & 0x0F)|(data & 0xF0);
        LCD_PRT |= (1<<LCD_RS);                         //RS = 1 for data
        LCD_PRT &= (1<<LCD_RW);                         //RW = 0 for write
        LCD_PRT |= (1<<LCD_EN);                         //EN = 1 for H-to-L
        delay_us(1);                                    //wait to make EN wider
        LCD_PRT &= ~(1<<LCDEN);                         //EN = 0 for H-to-L
        LCD_PRT = (LCD_PRT & 0x0F)|(data << 4);
        LCD_PRT |= (1<<LCD_EN);                         //EN = 1 for H-to-L
        delay_us(1);                                    //wait to make EN wider
        LCD_PRT &= ~(1<<LCD_EN);                        //EN = 0 for H-to-L
}
```

LCD AND KEYBOARD INTERFACING

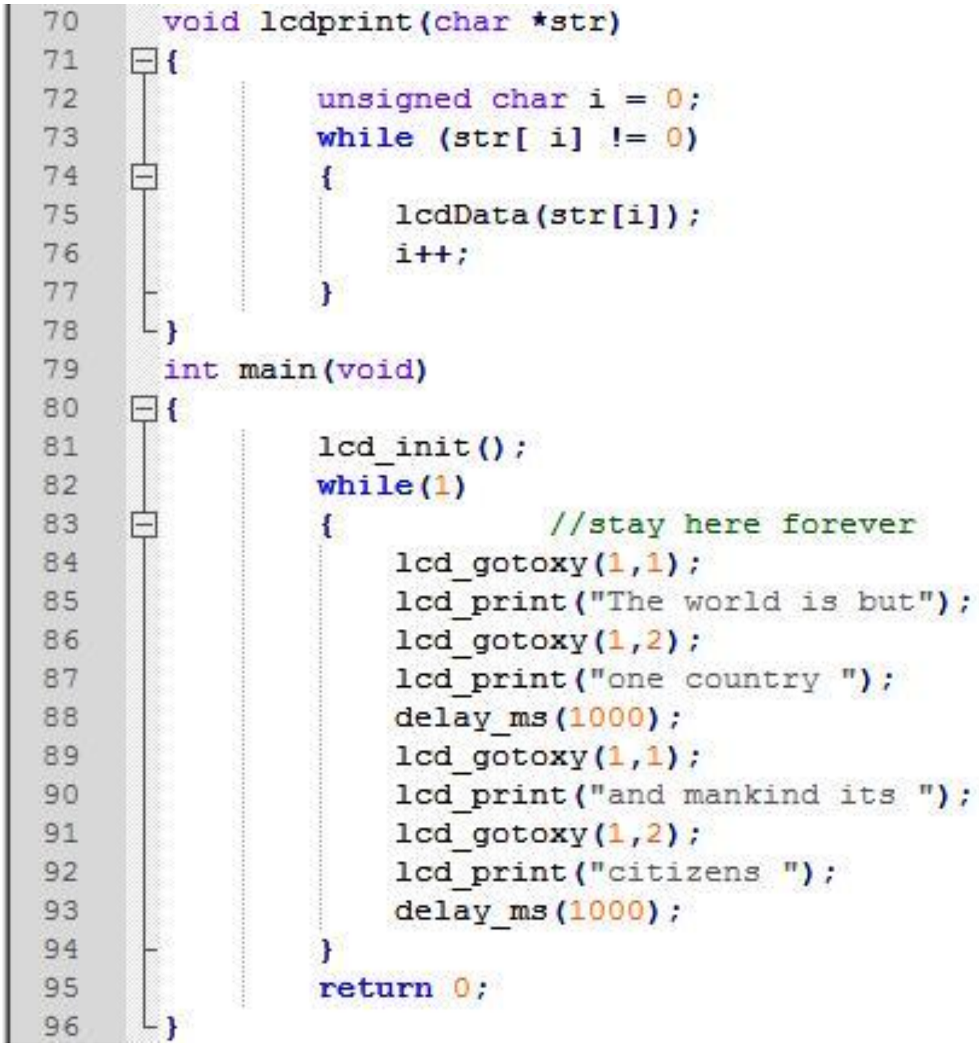
12.1 LCD INTERFACING



```
void lcd_init()
{
        LCD DDR = 0xFF;                                    //LCD port is output
        LCD_PRT &= ~(1<<LCDEN);                            //LCD_EN = 0
        delay_us(2000);                                    //wait for stable power
        lcdCommand(0x33);                                  //$33 for 4-bit mode
        delay_us(100);                                     //wait
        lcdCommand(0x32);                                  //$32 for 4-bit mode
        delay_us(100);                                     //wait
        lcdCommand(Ox28);                                  //$28 for 4-bit mode
        delay_us(100);                                     //wait
        lcdCommand(0x0e);                                  //display on, cursor on
        delay_us(100);                                     //wait
        lcdCommand(0x01);                                  //clear LCD
        delay_us(2000);                                    //wait
        lcdCommand(0x06);                                  //shift cursor right
        delay_us(100);                                     //wait
}
void lcd_gotoxy(unsigned char x, unsigned char y)
{       //Table 12-5
        unsigned char firstCharAdr[] = {0x80, 0xC0, 0x94, 0xD4};
        lcdCommand(firstCharAdr[ y-1] + x - 1);
        delay_us(100);
}
```

LCD AND KEYBOARD INTERFACING

12.1 LCD INTERFACING



```
void lcdprint(char *str)
{
        unsigned char i = 0;
        while (str[ i] != 0)
        {
            lcdData(str[i]);
            i++;
        }
}
int main(void)
{
        lcd_init();
        while(1)
        {           //stay here forever
            lcd_gotoxy(1,1);
            lcd_print("The world is but");
            lcd_gotoxy(1,2);
            lcd_print("one country ");
            delay_ms(1000);
            lcd_gotoxy(1,1);
            lcd_print("and mankind its ");
            lcd_gotoxy(1,2);
            lcd_print("citizens ");
            delay_ms(1000);
        }
        return 0;
}
```

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

Interfacing the keyboard to the AVR

At the lowest level, keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports; therefore, with two 8-bit ports, an 8x8 matrix of keys can be connected to a microcontroller.

When a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns.

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING

Scanning and identifying the key

The rows are connected to an output port and the columns are connected to an input port.

If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (VCC).

If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground.

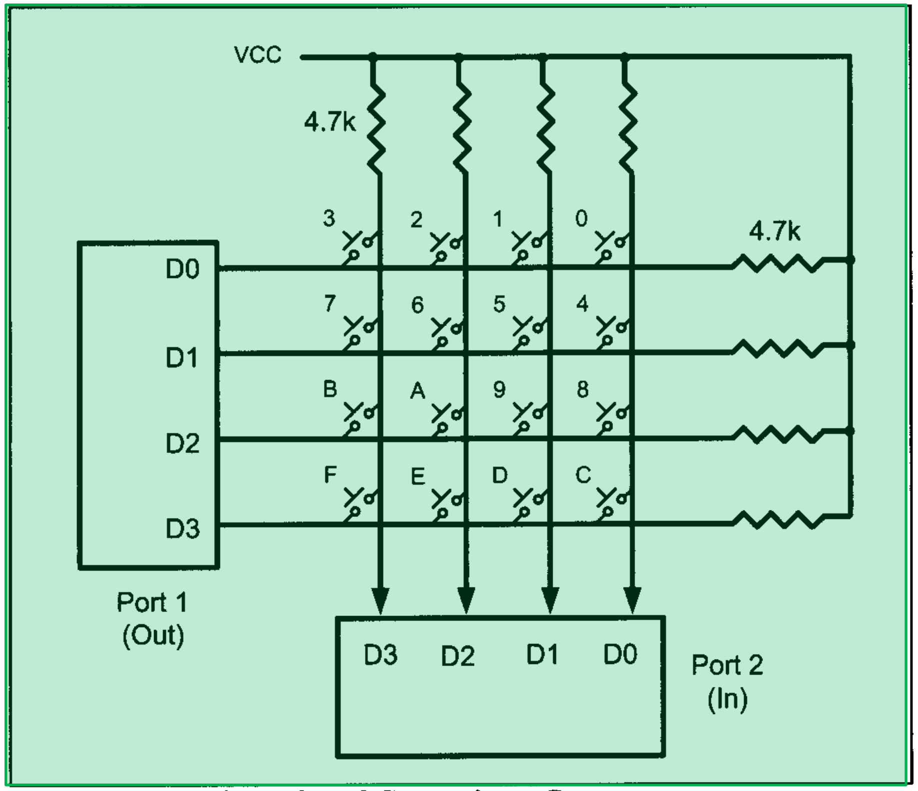


Figure 12-7. Matrix Keyboard Connection to Ports

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

Grounding rows and reading the columns

To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, and then it reads the columns.

If the data read from the columns is D3-D0=1111, no key has been pressed and the process continues until a key press is detected.

However, if one of the column bits has a zero, this means that a key press has occurred. For example, if D3-D0 = 1101, this means that a key in the D1 column has been pressed.

After a key press is detected, the microcontroller will go through the process of identifying the key.

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

Grounding rows and reading the columns

Starting with the top row, the microcontroller grounds it by providing a low to row D0 only; then it reads the columns.

If the data read is all 1s, no key in that row is activated and the process is moved to the next row.

It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified.

After identification of the row in which the key has been pressed, the next task is to find out which column the pressed key belongs to. This should be easy since the microcontroller knows at any time which row and column are being accessed. Look at Example 12-2.

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

Example 12-2

From Figure 12-7 identify the row and column of the pressed key for each of the following.

(a) 03-D0 = 1110 for the row, D3-D0 = 1011 for the column

(b) 03-D0 = 1101 for the row, D3-D0 = 0111 for the column

Solution:

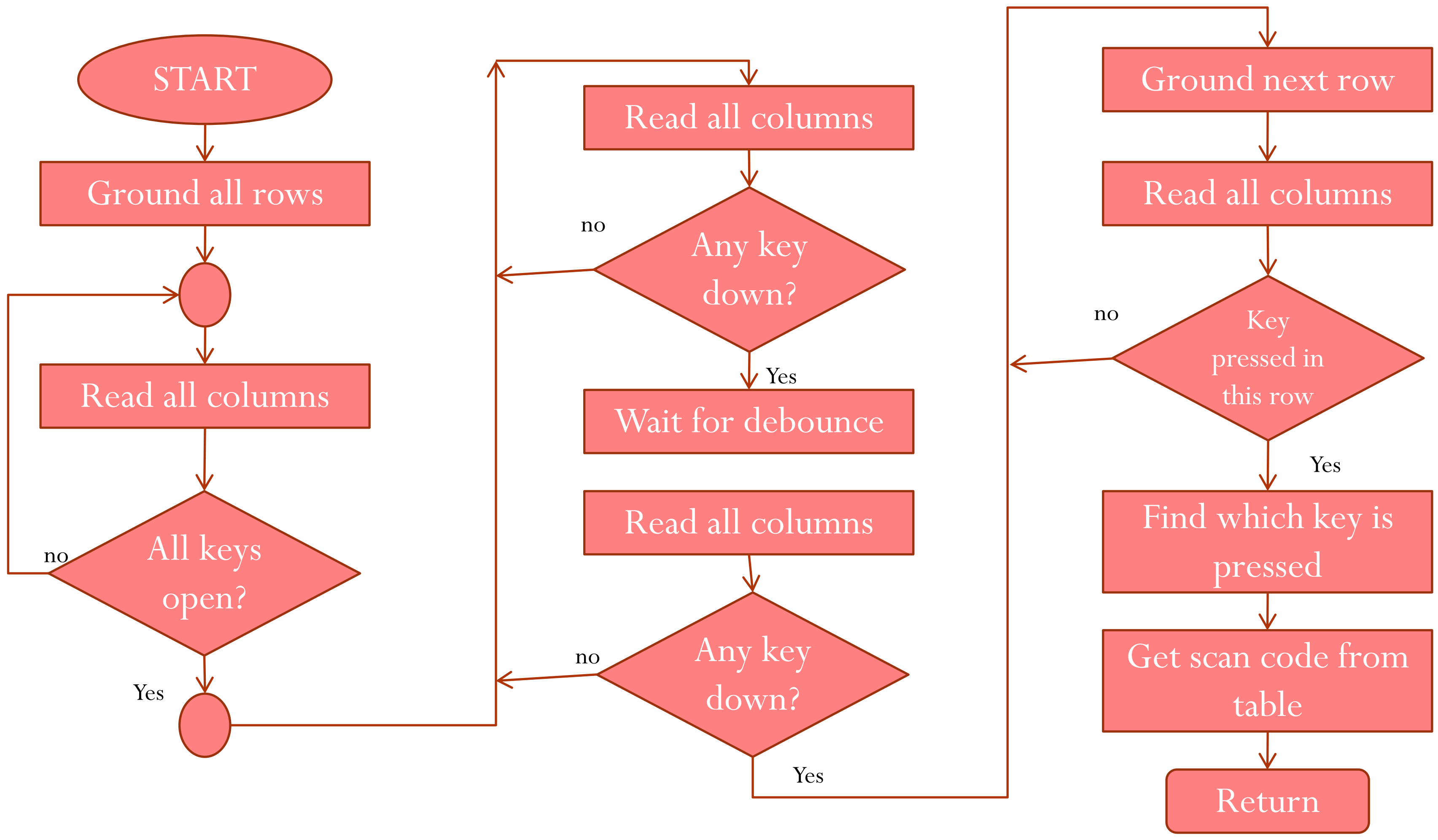
From Figure 12-7 the row and column can be used to identify the key.

(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.

(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING



LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

Program 12-8 goes through the following four major stages (Figure 12-8 flowcharts this process):

1. To make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high. When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed.

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

2. To see if any key is pressed, the columns are scanned over and over in an infinite loop until one of them has a 0 on it. Remember that the output latches connected to rows still have their initial zeros, making them grounded. After the key press detection, the microcontroller waits 20 ms for the bounce and then scans the columns again. This serves two functions: (a) it ensures that the first key press detection was not an erroneous one due to a spike noise, and (b) the 20-ms delay prevents the same key press from being interpreted as a multiple key press. Look at Figure 12-9. If after the 20-ms delay the key is still pressed, it goes to the next stage to detect which row it belongs to; otherwise, it goes back into the loop to detect a real key press.

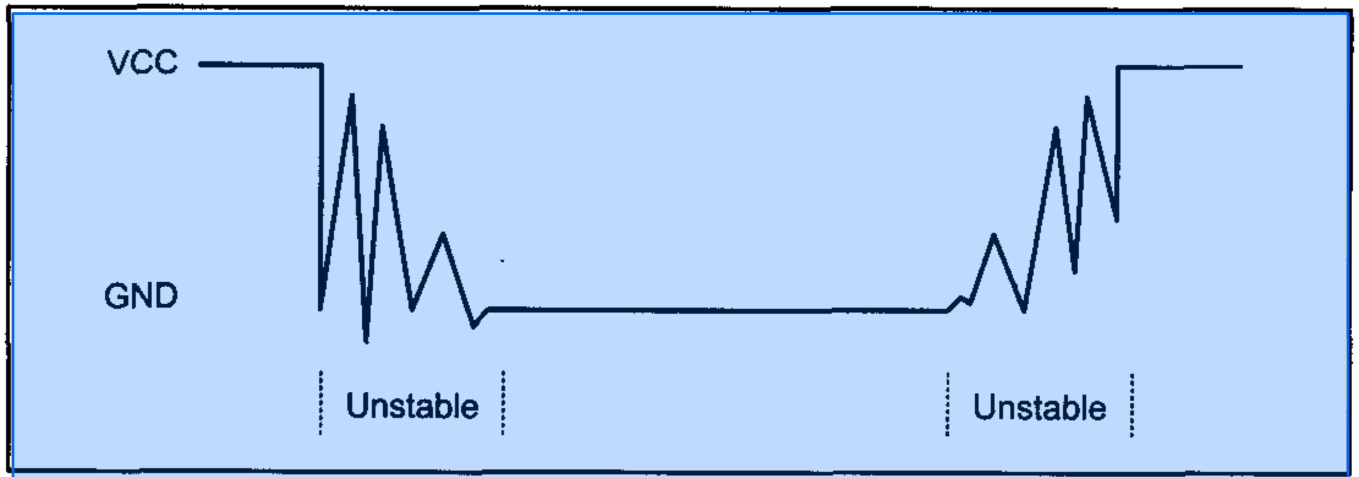


Figure 12-9. Keyboard Debounce

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

3. To detect which row the key press belongs to, the microcontroller grounds one row at a time, reading the columns each time. If it finds that all columns are high, this means that the key press cannot belong to that row; therefore, it grounds the next row and continues until it finds the row the key press belongs to. Upon finding the row that the key press belongs to, it sets up the starting address for the look-up table holding the scan codes (or the ASCII value) for that row and goes to the next stage to identify the key.

LCD AND KEYBOARD INTERFACING
12.2 KEYBOARD INTERFACING

4. To identify the key press, the microcontroller rotates the column bits, one bit at a time, into the carry flag and checks to see if it is low. Upon finding the zero, it pulls out the ASCII code for that key from the look-up table; otherwise, it increments the pointer to point to the next element of the look-up table.

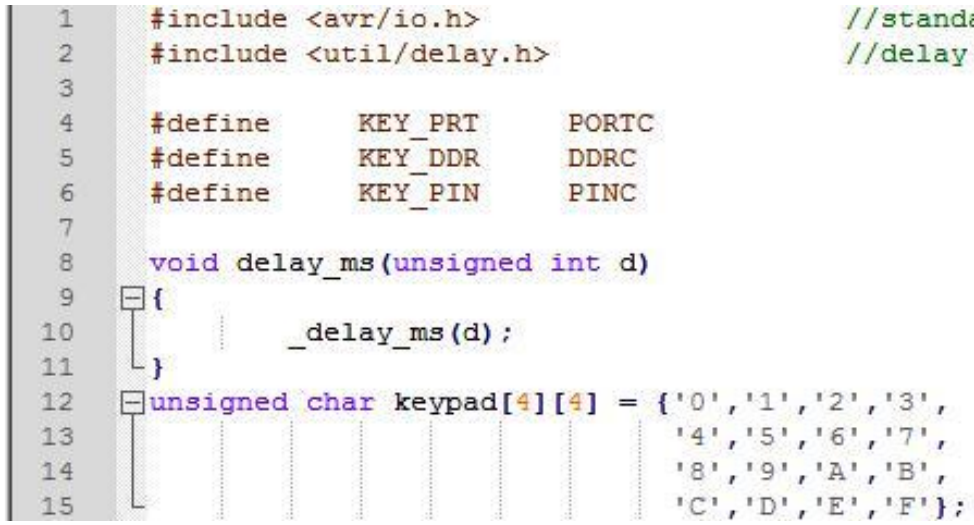
While the key press detection is standard for all keyboards, the process for determining which key is pressed varies. The look-up table method shown in Program 12-8 can be modified to work with any matrix up to 8 x 8. Example 12-3 shows keypad programming in C.

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING

Example 12-3

Write a C program to read the keypad and send the result to Port D.
PC0-PC3 connected to columns
PC4-PC7 connected to rows

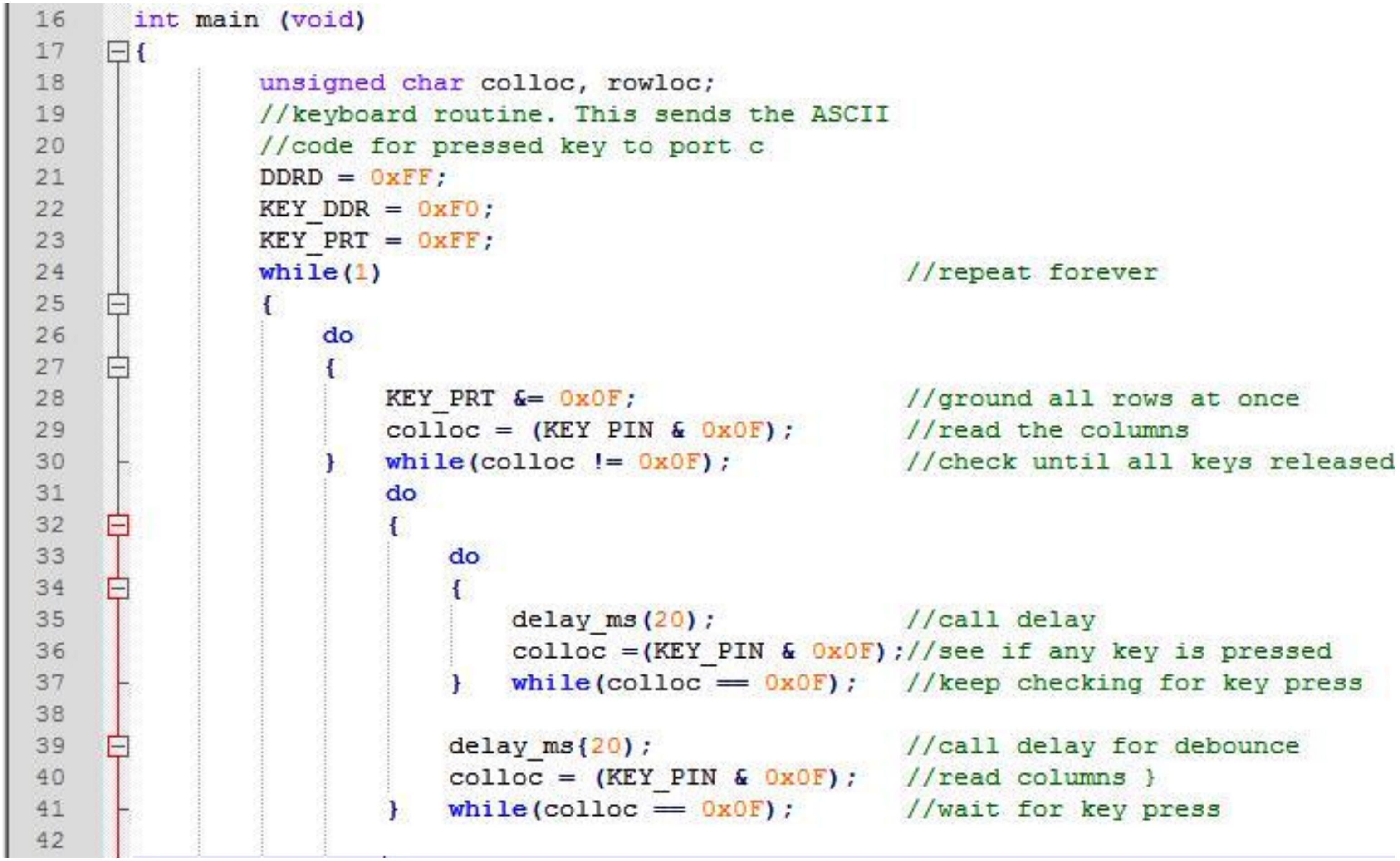


```
#include <avr/io.h>                         //standard AVR header
#include <util/delay.h>                     //delay header

#define     KEY_PRT     PORTC                        //keyboard PORT
#define     KEY_DDR     DDRC                         //keyboard DDR
#define     KEY_PIN     PINC                         //keyboard PIN

void delay_ms(unsigned int d)
{
        _delay_ms(d);
}
unsigned char keypad[4][4] = {'0','1','2','3',
                               '4','5','6','7',
                               '8','9','A','B',
                               'C','D','E','F'};
```

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING

Example 12-3

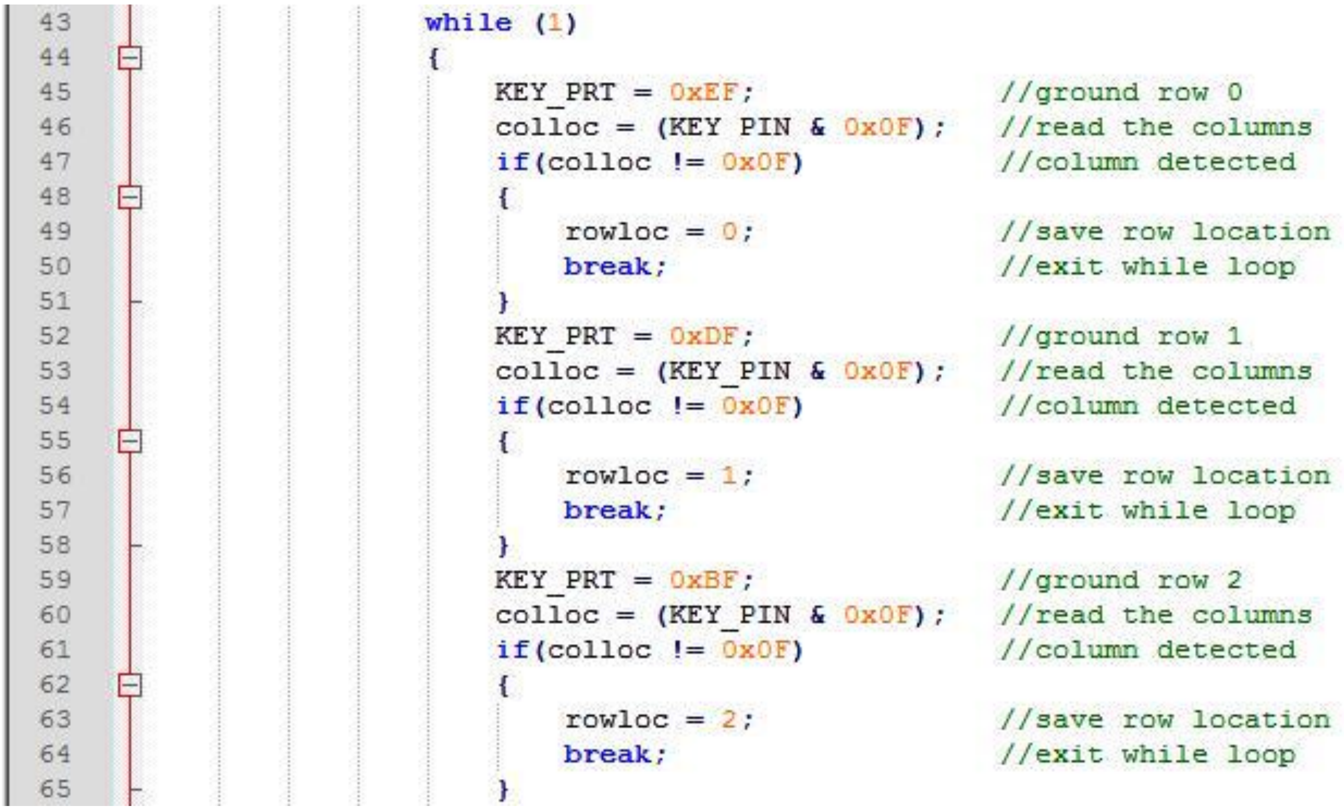


```
int main (void)
{
        unsigned char colloc, rowloc;
        //keyboard routine. This sends the ASCII
        //code for pressed key to port c
        DDRD = 0xFF;
        KEY_DDR = 0xF0;
        KEY_PRT = 0xFF;
        while(1)                                    //repeat forever
        {
            do
            {
                KEY_PRT &= 0x0F;                    //ground all rows at once
                colloc = (KEY PIN & 0x0F);          //read the columns
            }   while(colloc != 0x0F);              //check until all keys released
                do
                {
                    do
                    {
                        delay_ms(20);               //call delay
                        colloc =(KEY_PIN & 0x0F);//see if any key is pressed
                    }   while(colloc == 0x0F);      //keep checking for key press

                    delay_ms(20);                   //call delay for debounce
                    colloc = (KEY_PIN & 0x0F);      //read columns }
                }   while(colloc == 0x0F);          //wait for key press
```

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING

Example 12-3

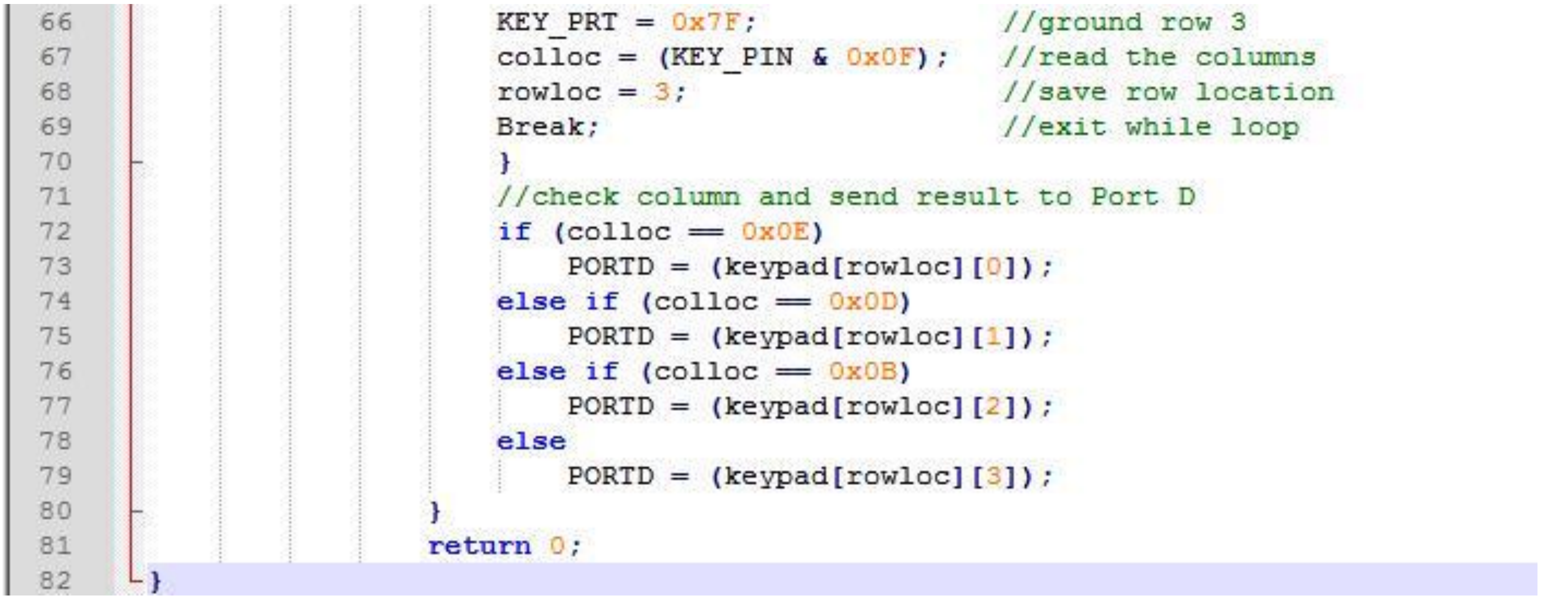


```
while (1)
{
    KEY_PRT = 0xEF;               //ground row 0
    colloc = (KEY_PIN & 0x0F);    //read the columns
    if(colloc != 0x0F)            //column detected
    {
        rowloc = 0;               //save row location
        break;                    //exit while loop
    }
    KEY_PRT = 0xDF;               //ground row 1
    colloc = (KEY_PIN & 0x0F);    //read the columns
    if(colloc != 0x0F)            //column detected
    {
        rowloc = 1;               //save row location
        break;                    //exit while loop
    }
    KEY_PRT = 0xBF;               //ground row 2
    colloc = (KEY_PIN & 0x0F);    //read the columns
    if(colloc != 0x0F)            //column detected
    {
        rowloc = 2;               //save row location
        break;                    //exit while loop
    }
```

LCD AND KEYBOARD INTERFACING

12.2 KEYBOARD INTERFACING

Example 12-3



```
                KEY_PRT = 0x7F;               //ground row 3
                colloc = (KEY_PIN & 0x0F);    //read the columns
                rowloc = 3;                   //save row location
                Break;                        //exit while loop
                }
                //check column and send result to Port D
                if (colloc == 0x0E)
                    PORTD = (keypad[rowloc][0]);
                else if (colloc == 0x0D)
                    PORTD = (keypad[rowloc][1]);
                else if (colloc == 0x0B)
                    PORTD = (keypad[rowloc][2]);
                else
                    PORTD = (keypad[rowloc][3]);
            }
            return 0;
}
```