AVR_GCC

Q.18 The representation of the decimal number (27.625)_10 in base-2 number system is

(A) 11011.110 (B) 11101.101
(C) 11011.101 (D) 10111.110

solution:

2 | 27		0.625 x 2 = 1.350 - 1
2 | 13 – 1		0.35 x 2 = 0.70 - 0
2 | 6 -- 1		0.7 x 2 = 1.4 - 1
2 | 3 -- 0		
2 | 1 --- 1		0.625 = 101

27.625 = 11011.101

27 = 11011

Code:

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
// TYPEDEFS
typedef uint8_t byte; // changed the name


// ------------------
//LCD DRIVER ROUTINES
//
// Routines:
// LCD_Init initializes the LCD controller
// LCD_Cmd sends LCD controller command
// LCD_Char sends single ascii character to display
// LCD_Clear clears the LCD display & homes cursor
// LCD_Integer displays an integer value
// LCD_Message displays a string
// PortB is used for data communications with the HD44780-controlled LCD.
// The following defines specify which port pins connect to the controller:
#define ClearBit(x,y) x &= ~_BV(y) // equivalent to cbi(x,y)
#define SetBit(x,y) x |= _BV(y) // equivalent to sbi(x,y)
#define LCD_RS 0 // pin for LCD R/S (eg PB0)
```

```
#define LCD_E 1 // pin for LCD enable
#define DAT4 2 // pin for d4
#define DAT5 3 // pin for d5
#define DAT6 4 // pin for d6
#define DAT7 5 // pin for d7
//// The following defines are controller commands
#define CLEARDISPLAY 0x01
#define INC 0x04

void PulseEnableLine ()
{
 SetBit(PORTB,LCD_E); // take LCD enable line high
 _delay_us(40); // wait 40 microseconds
 ClearBit(PORTB,LCD_E); // take LCD enable line low
}
void SendNibble(byte data)
{
 PORTB &= 0xC3; // 1100.0011 = clear 4 data lines
 if (data & _BV(4)) SetBit(PORTB,DAT4);
 if (data & _BV(5)) SetBit(PORTB,DAT5);
 if (data & _BV(6)) SetBit(PORTB,DAT6);
 if (data & _BV(7)) SetBit(PORTB,DAT7);
 PulseEnableLine(); // clock 4 bits into controller
}
void SendByte (byte data)
{
 SendNibble(data); // send upper 4 bits
 SendNibble(data<<4); // send lower 4 bits
 ClearBit(PORTB,5); // turn off boarduino LED
}
void LCD_Cmd (byte cmd)
{
 ClearBit(PORTB,LCD_RS); // R/S line 0 = command data
 SendByte(cmd); // send it
}
void LCD_Char (byte ch)
{
 SetBit(PORTB,LCD_RS); // R/S line 1 = character data
 SendByte(ch); // send it
}
void LCD_Init()
{
 LCD_Cmd(0x33); // initialize controller
 LCD_Cmd(0x32); // set to 4-bit input mode
 LCD_Cmd(0x28); // 2 line, 5x7 matrix
 LCD_Cmd(0x0C); // turn cursor off (0x0E to enable)
```

```
 LCD_Cmd(0x06); // cursor direction = right
 LCD_Cmd(0x01); // start with clear display
 _delay_ms(3); // wait for LCD to initialize
}
void LCD_Clear() // clear the LCD display
{
 LCD_Cmd(CLEARDISPLAY);
_delay_ms(3); // wait for LCD to process command
}

void LCD_Message(const char *text) // display string on LCD
{
 //while (*text) // do until /0 character
 LCD_Char(*text++); // send char & update char pointer
}

void LCD_Integer(int data)
// displays the integer value of DATA at current LCD cursor position
{
 char st[20] = ""; // save enough space for result
 itoa(data,st,10); //
 LCD_Message(st); // display in on LCD
}

void IntegerToBinary(int num,char *binary){
        for(int i = 7; i>= 0; i--){
                binary[i] = (num % 2) + '0';
                num /= 2;
        }
        binary[8] = '\0';
       }
// MAIN PROGRAM
int main(void)
{
  float number = 27.625;
  int a = (int)number;
  float b = number - a;
  char binary_whole_part[9]="";
  char binary_fractional_part[9]="";
  IntegerToBinary(a,binary_whole_part);
  for(int i = 0; i < 8; i++){
    b *= 2;
    int bit = (int)b;
    binary_fractional_part[i] = bit + '0';
    b -= bit;
  }
```

```
  binary_fractional_part[8] = '\0';
// use PortB for LCD interface
 DDRB = 0xFF; // 1111.1111; set PB0-PB7 as outputs
 LCD_Init(); // initialize LCD controller
 LCD_Cmd(0x80);
 LCD_Char('B');
 LCD_Char('i');
 LCD_Char('n');
 LCD_Char('a');
 LCD_Char('r');
 LCD_Char('y');
 LCD_Char(':');
 LCD_Cmd(0xC0);
 for(int i = 0;i<8;i++){
      LCD_Char(binary_whole_part[i]);
 }
 LCD_Char('.');
 for(int  i = 0;i<8;i++)
 {
      LCD_Char(binary_fractional_part[i]);
 }
 while(1)
 {
}
}
```

LCD to Arduino connections

LCD	ARDUINO
pin1(Vss)	*GND*
pin2(Vcc)	*Vcc*
pin3(VEE)	*220ohms(GND)*
pin4(RS)	*pin8*
pin5(RW)	*GND*
pin6(En)	*pin9*
pin11(D4)	*pin10*
pin12(D5)	*pin11*
pin13(D6)	*pin12*
pin14(D7)	*pin13*
pin15(LED)	*Vcc*
pin16(LED)	*GND*

Output: