北京航空航天大學
BEIHANG UNIVERSITY

Curriculum Design of Practice on Electrical Technology

Pricing System Based on IR Sensor and FPGA

Beihang University

School of Automation Science and Electrical Engineering

Heng Cao(16031192) *Xing-Yuan Xu*(16711100)
Han-Tao Li(16711094) *Mu-Ze Li*(16101050)

Index

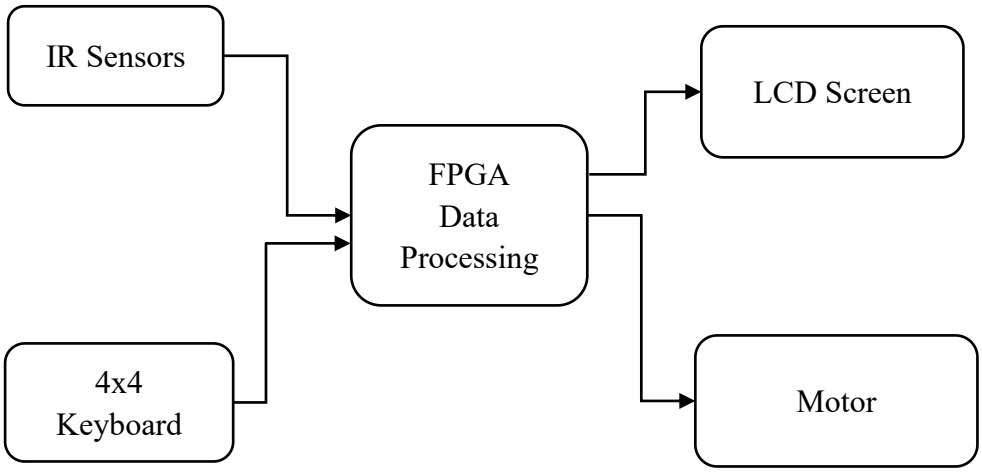
I. Background of the Project

Nowadays,shopping is very convenient.The main ways to shop are online shopping and offline shopping. Although online shopping is prevalent,it also has its shortcomings, so offline shopping can not be replaced. When shopping in supermarkets, it is inevitable to use the cash register. Our group wonders the structure of the cash register.So we try to create a simulated cash register through efforts. Under the condition of using only FPGA as the central control, not only the cost can be reduced, but also the calculation accuracy is high.

I. System design ideas

i) General design

The purpose of this project is to design a simplified and yet well-functional cash register. So after referencing real-life cash register designs, we figured the direct interface for our cash register would be based around a LCD screen, for displaying the item, its price, as well as the total price to be payed when registering a customer and the item, its current price, the price to be changed to when changing the price of an item. The processing of the data needed to be displayed would be done by an FPGA with sensors providing the needed signals in. We considered IR sensors would be a viable and simple solution to scan in the pattern code (black and white) on the items to identify them. Other signals needed to be entered manually, such as the new price of the item, changing through different modes of the cash register would be provided by a 4x4 keyboard. Output signals, aside from to the LCD, would be to a motor driver board to drive a motor and imitate the opening of the drawer on the cash register when accepting payment.

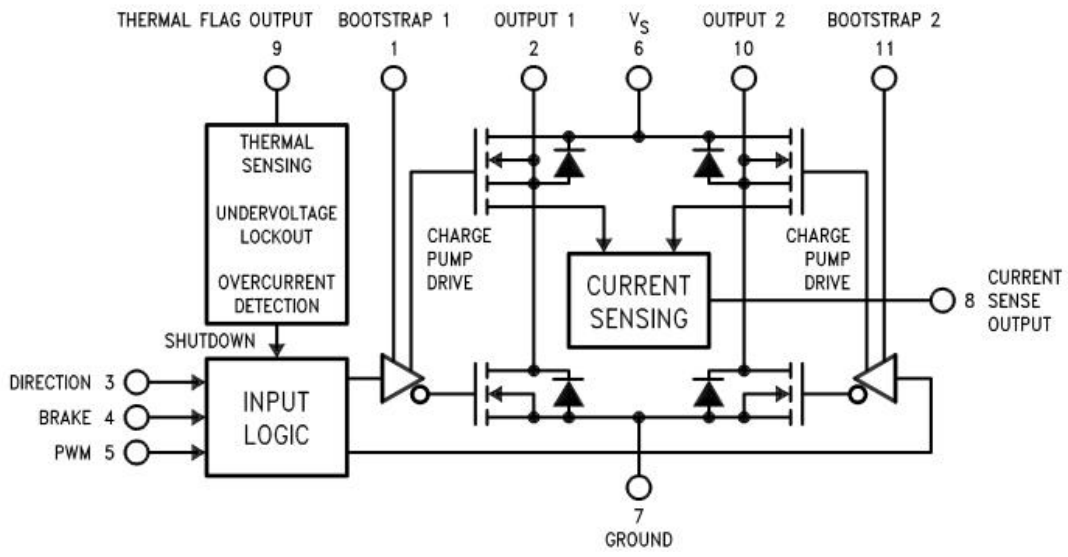


ii) Drawer circuit

In this part,we have to drive a motor to work for several seconds in both directions to simulate the condition after checking out.

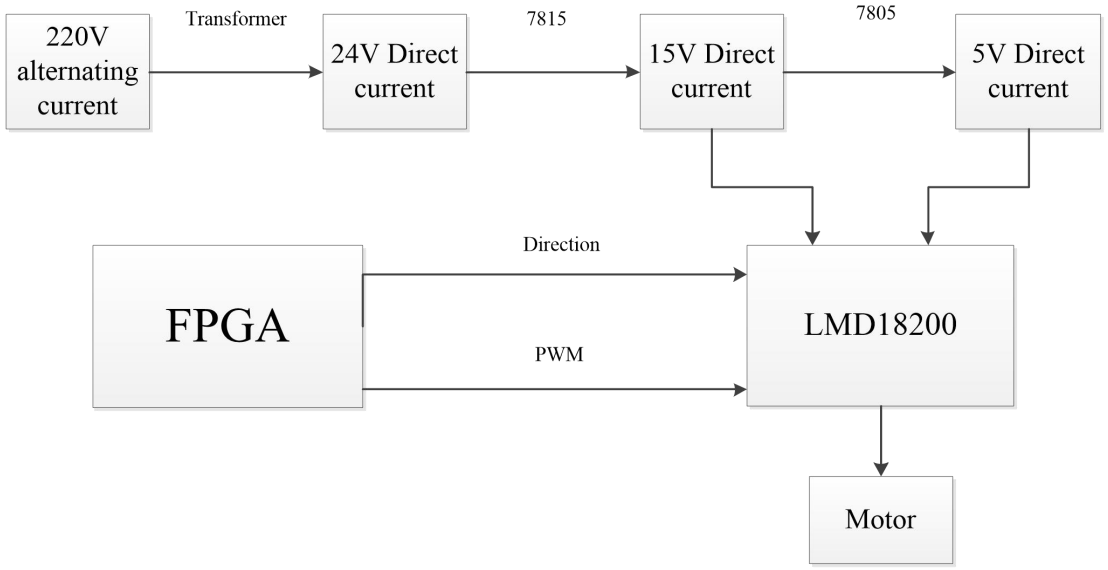
A.Motor drive chip

The LMD18200 is a 3A H-bridge designed for motion control applications.The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure.Ideal for driving DC and stepper motors;the LMD18200 accommodates peak output currents up to 6A. An innovative circuit which facilitates low-loss sensing of the output current has been implemented.



B.Motor drive circuit

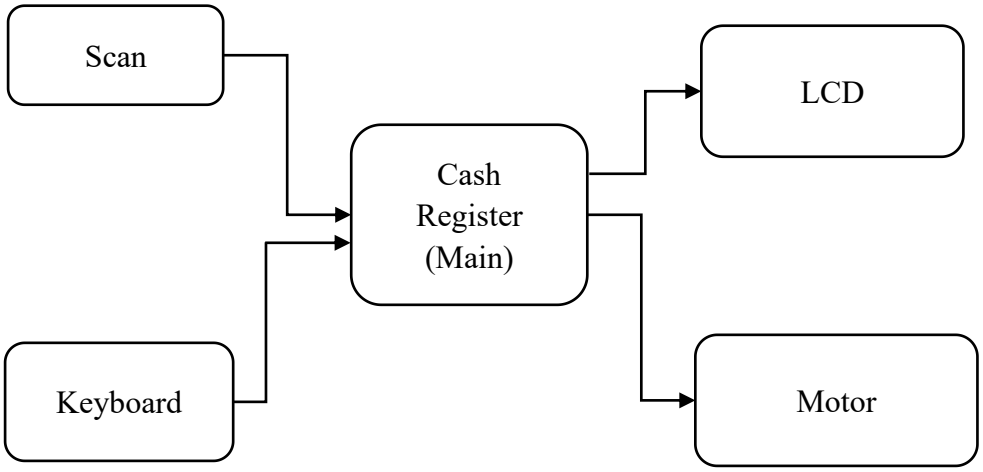
In order to drive the chip, we need to give LMD18200 proper voltage. The chip both require 5V and 15V.As a result, We have to provide the chip with them. FPGA gives the signal of the speed and orientation of the motor. The following chart shows how we can complete the task.



iii) The logic control and human-computer interaction interface based on FPGA

The FPGA segment of the cash register consists of 5 parts: cash register,

keyboard, LCD, scan, motor, corresponding to the design of the entire system.



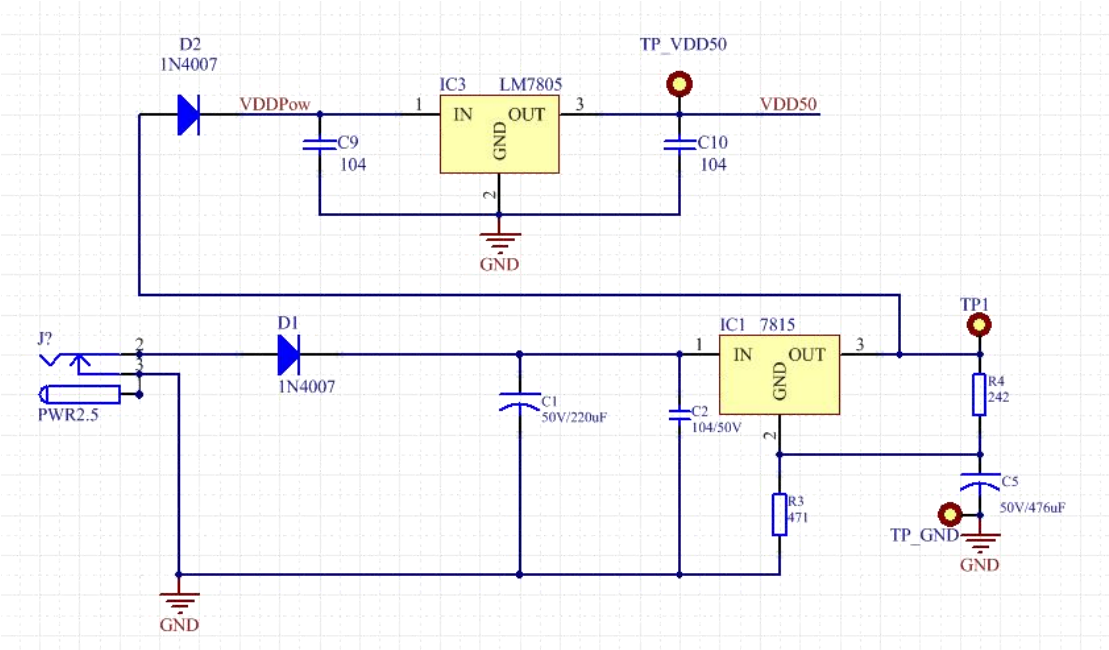
The cash register part is the main part of the FPGA segment. Its job is to switch through the various stages the cash register is working in according to user input as well as process the data scanned in through the sensors to calculate a total price for the LCD part. The total price is calculated by multiplying teach item amount to its corresponding price and adding them all up.

II. Principles of the project

i) Principles of the motor part

1.Power supply part

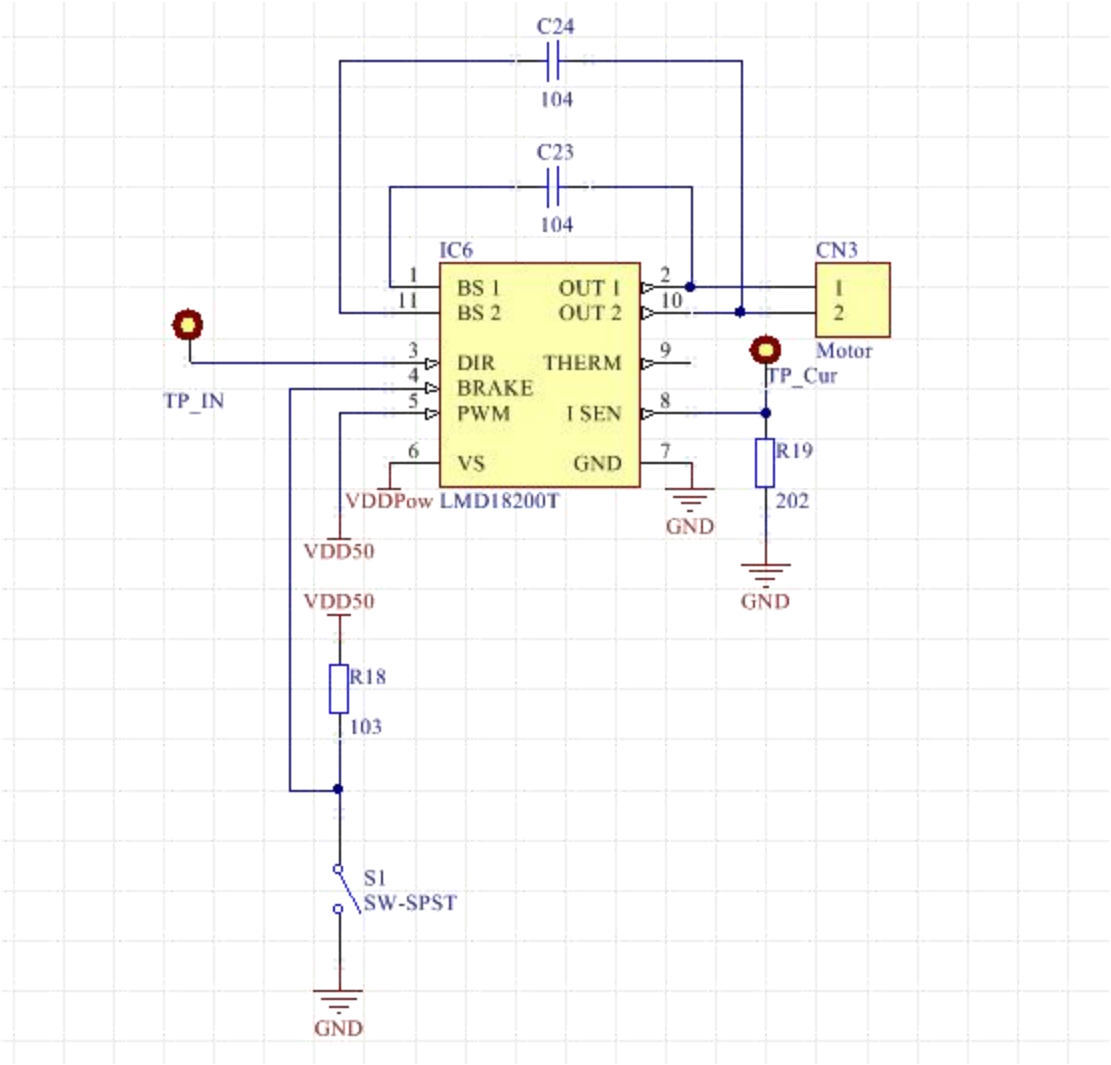
In this part ,we need to create two kinds of voltage.The schematic diagram is shown followed:



Because we use a external transformer,the voltage input is 24V direct voltage. After going through 7815, VDDPow becomes 15V, which is the input voltage of 7805. After going through 7805, the output voltage is 5V.15V and 5V are both indispensable for the operating of LMD18200. All the Resistors and capacitances are essential to purify the voltage.

2.Motor partial circuit

In order to drive a motor ,we choose the LMD18200,which is particularly designed for the motor.We design the circuit as shown below:



We give signal to pin three and pin five.In the diagram above,we connect pin 5 to 5V,but actually we can give it PWM to design the speed of the motor.The closer the voltage is to numerical 1,the faster the motor will operate.Pin three is used to control the orientation of the motor's output.If we change the input voltage from 0 to numerical 1,the direction of the motor will reverse.

ii) Principles of the LCD screen part

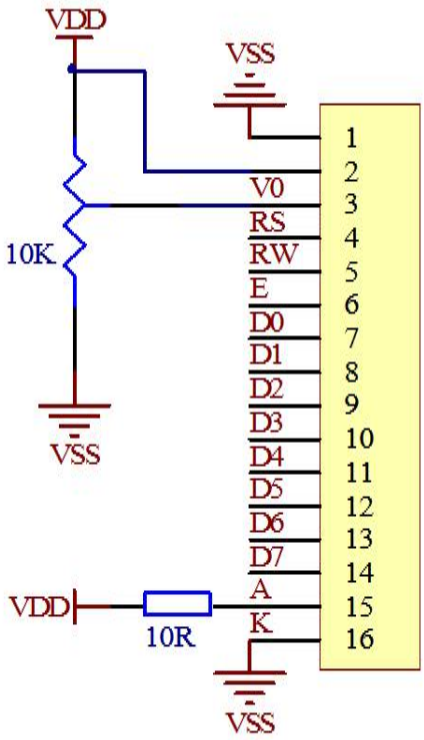
1.Basic structure

1602 liquid crystal is also called 1602 character liquid crystal. It is a dot matrix liquid crystal module used to display letters, numbers, symbols and so on. It consists

of several 5X7 or 5X11 dot-matrix character bits. Each dot-matrix character bit can display a character. There is a dot-distance interval between each bit, and there is a space between each line. It plays the role of character spacing and line spacing.

1602 LCD refers to the content of the display is 16X2, that is to say, it can display two lines, each line of 16 characters LCD module (display characters and numbers).

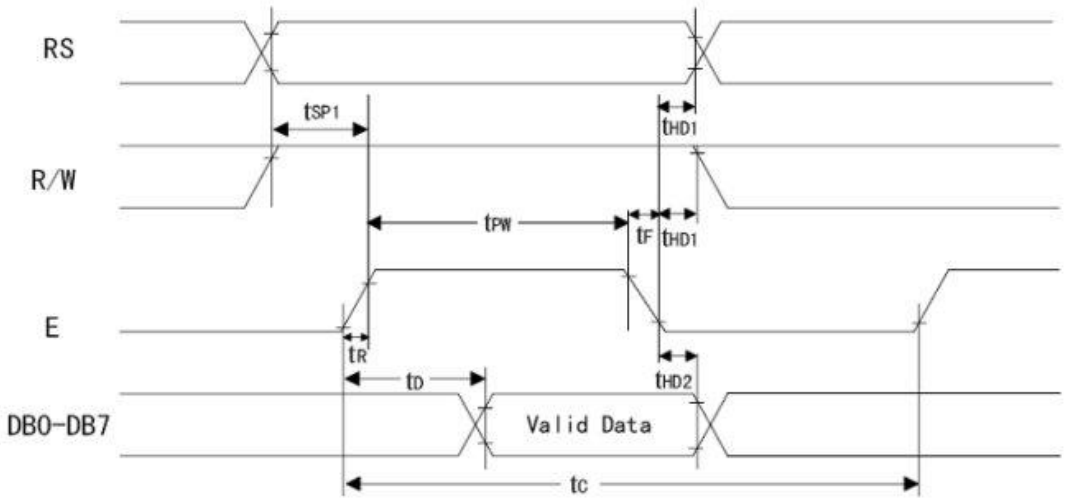
The schematic diagram is as follows:

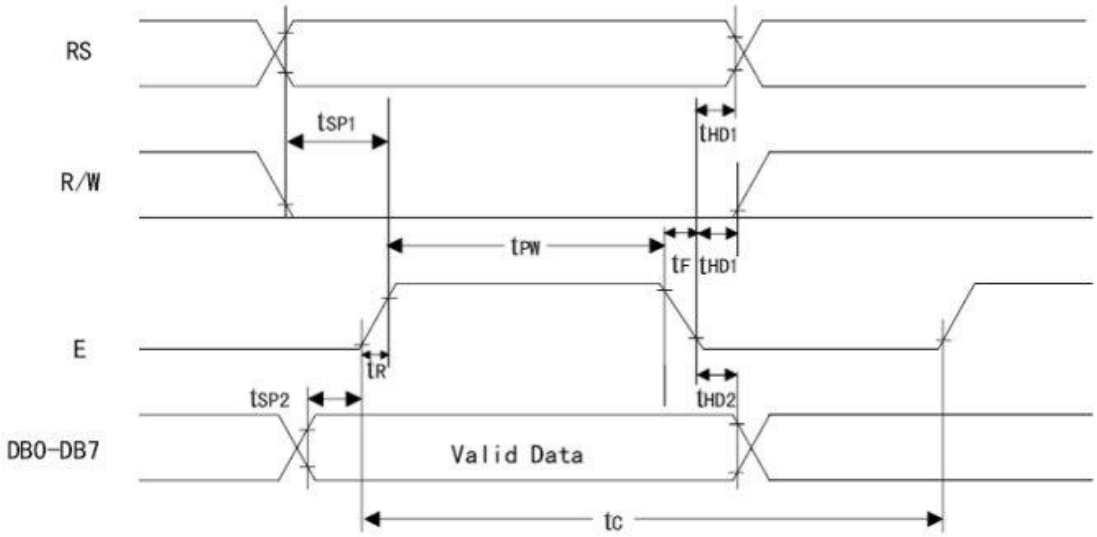


2.Basic operation

First, the selection of registers and the selection of read and write operations should be configured. RS is register selection, RS = 1 means operating on data; RS = 0 means operating on instructions. Next, read and write operation selection should be configured, RW = 0 means writing operation. Open the enable port and input the enable signal E = 1, then data bus assign data to DB0~DB7 and transmit data.

Below is the reading operation sequence diagram and writing operation sequence diagram of 1602LCD.



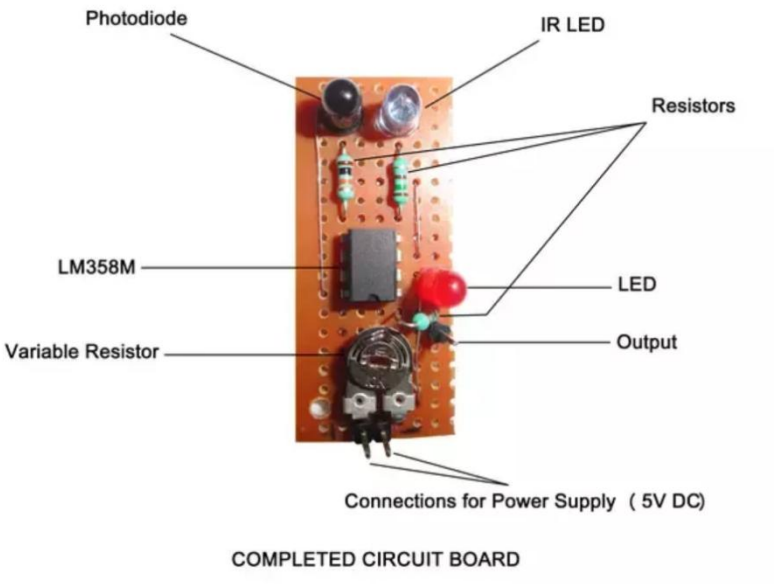


iii) Principles of the IR sensor part

1.Basic function

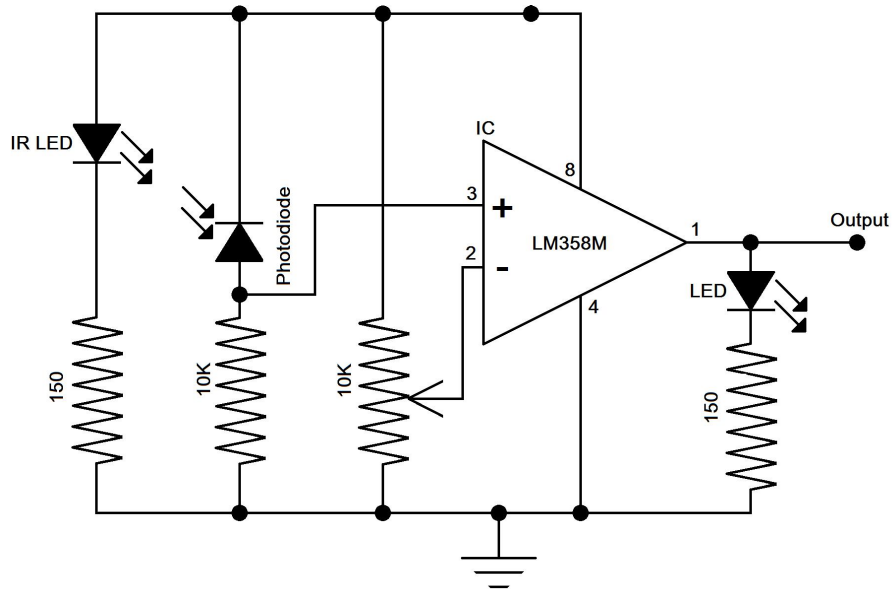
In the merchandise scanning process, each item will correspond to a fixed black bar code. Therefore, it is only necessary to apply an infrared sensor to the FPGA to implement commodity scanning. The product plans to use four infrared sensors to implement the scanning function, which can define the price of up to 16 items.

The diagram of IR sensor is as follows:



COMPLETED CIRCUIT BOARD

An IR sensor is a device which detects IR radiation falling on it. It is basically a device which consists of a pair of an IR LED and a photodiode which are collectively called a photo-coupler or an opto-coupler. The IR LED emits IR radiation, reception or intensity of reception of which by the photodiode dictates the output of the sensor.

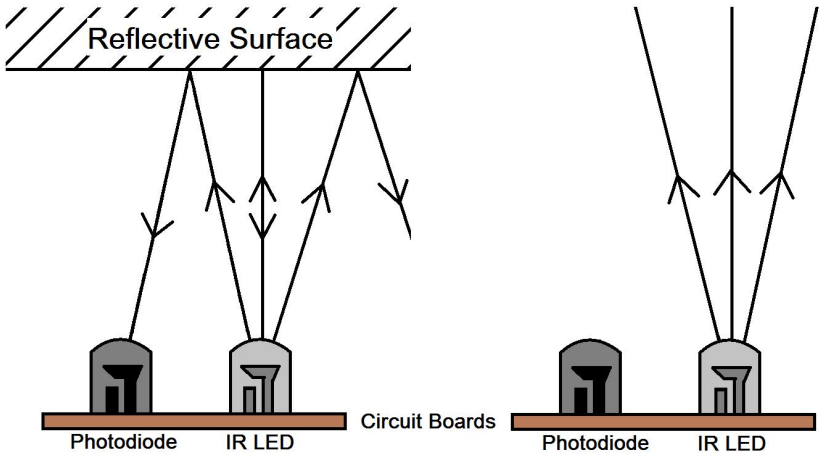


2.Basic operation

We may hold the IR LED directly in front of the photodiode, such that almost all the radiation emitted, reaches the photodiode. This creates an invisible line of IR radiation between the IR LED and the photodiode. Now, if an opaque object is placed obstructing this line, the radiation will not reach the photodiode and will get either reflected or absorbed by the obstructing object. This mechanism is used in object counters and burglar alarms. If we place an opaque object in front the two, two cases occur:

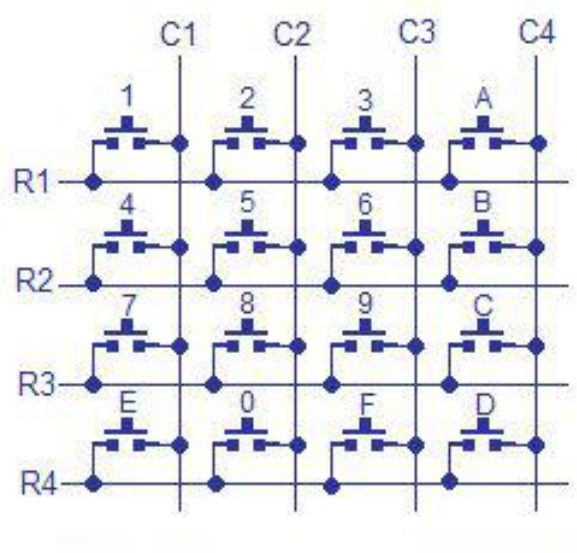
If the object is reflective, (White or some other light color), then most of the radiation will get reflected by it and will get incident on the photodiode. For further understanding, please refer to the left part of the illustration below.

If the object is non-reflective, (Black or some other dark color), then most of the radiation will get absorbed by it and will not become incident on the photodiode. It is like there being no surface (object) at all, for the sensor, as in both the cases, it does not receive any radiation.

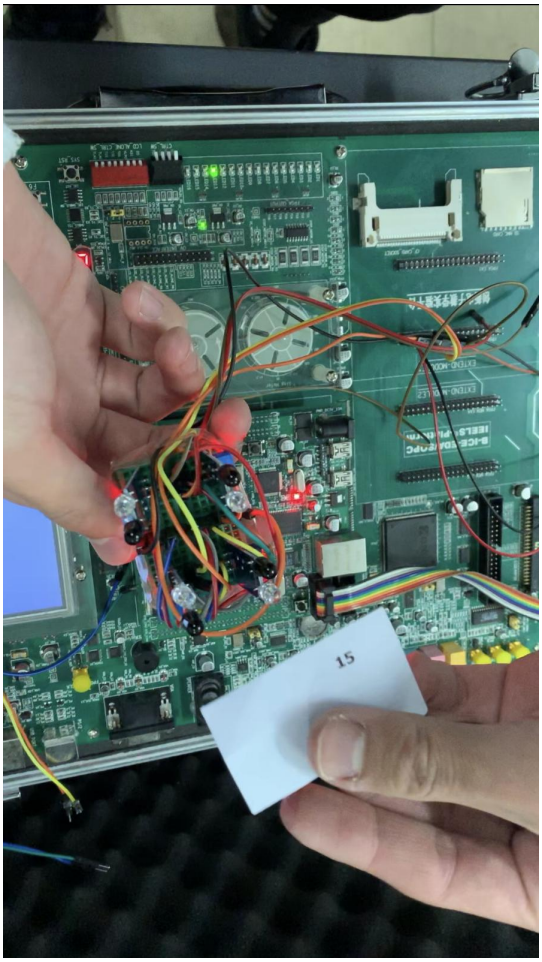


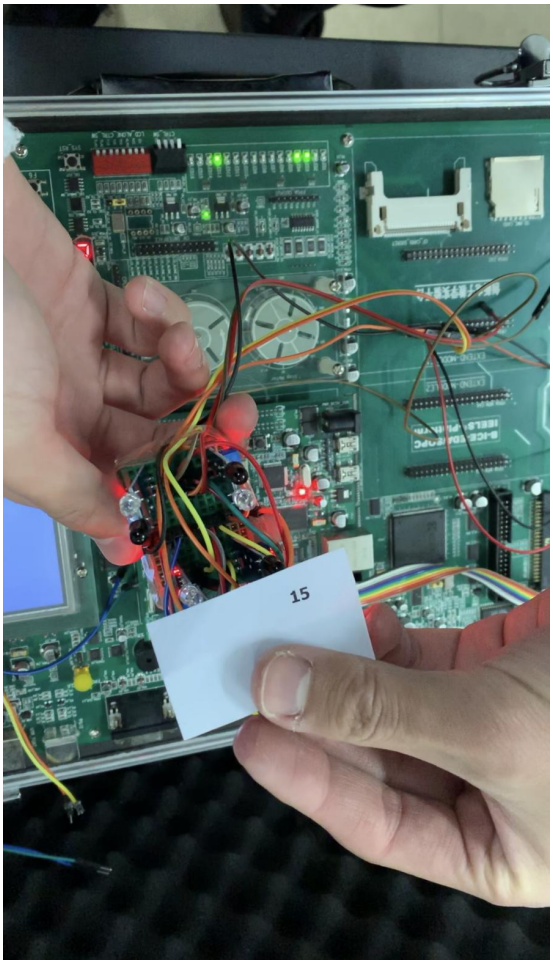
iv) Principles of the 4x4 Keyboard part

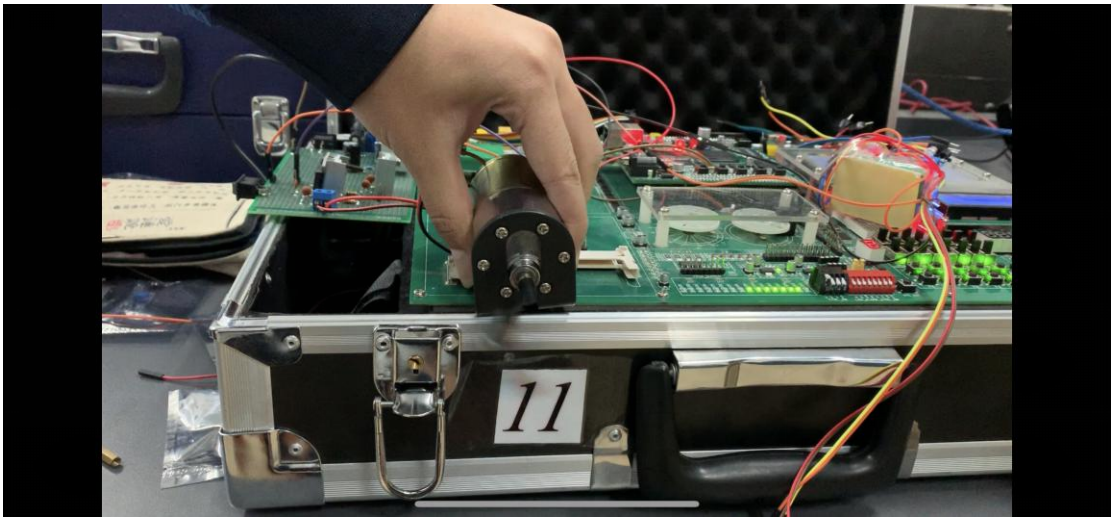
The user's data input is realized by the 4x4 keyboard. As shown in the following figure, the 4X4 keyboard monitors input by scanning mode, four lines as input lines and four lines as output lines, and monitors the output status of the lines according to the flow of clocks to one of the four lines. If one line outputs a low level, it proves that the 4X4 keyboard monitors input by scanning mode, four lines as input lines and four lines as output lines. The key on the corresponding intersection point is pressed.

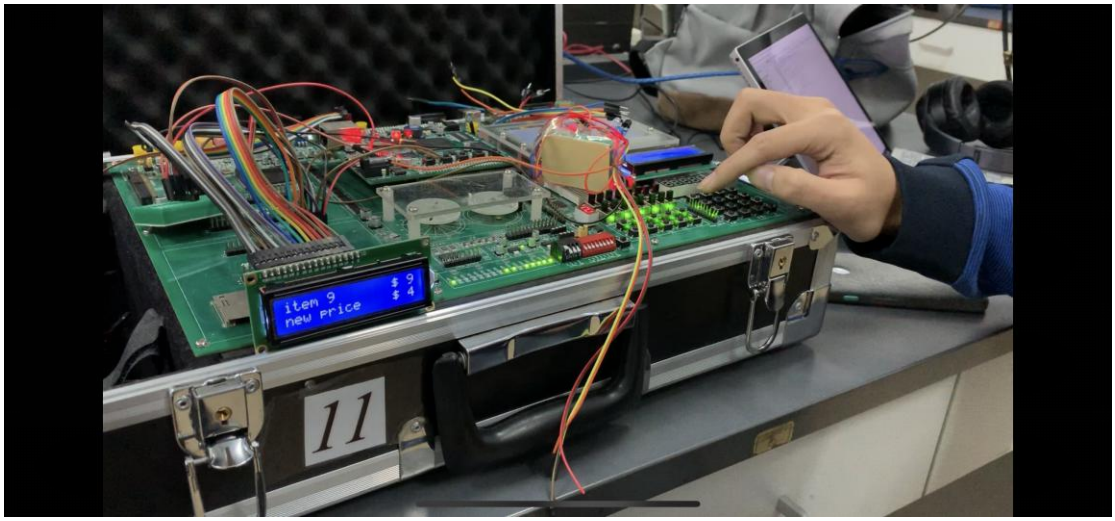


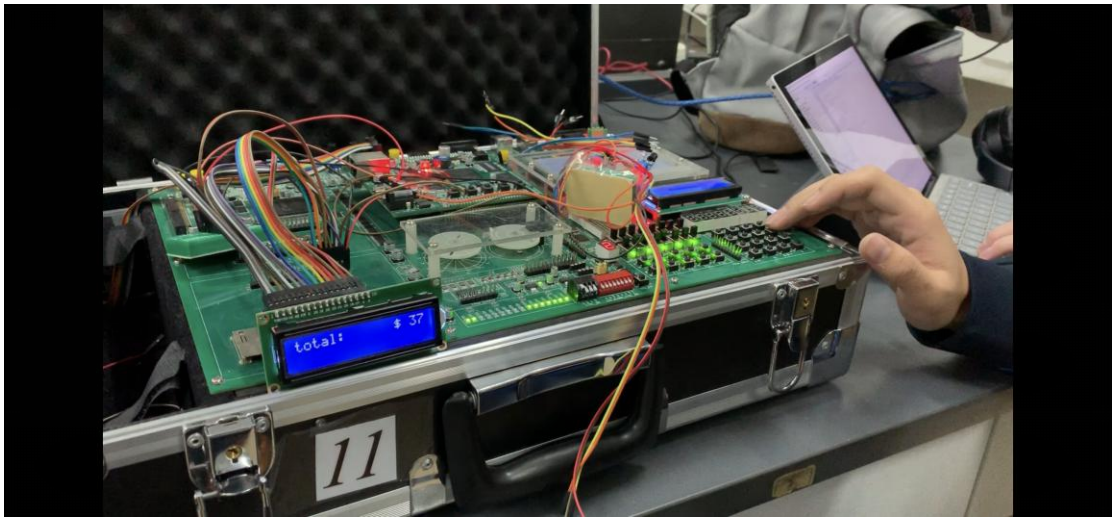
III. Display of the system

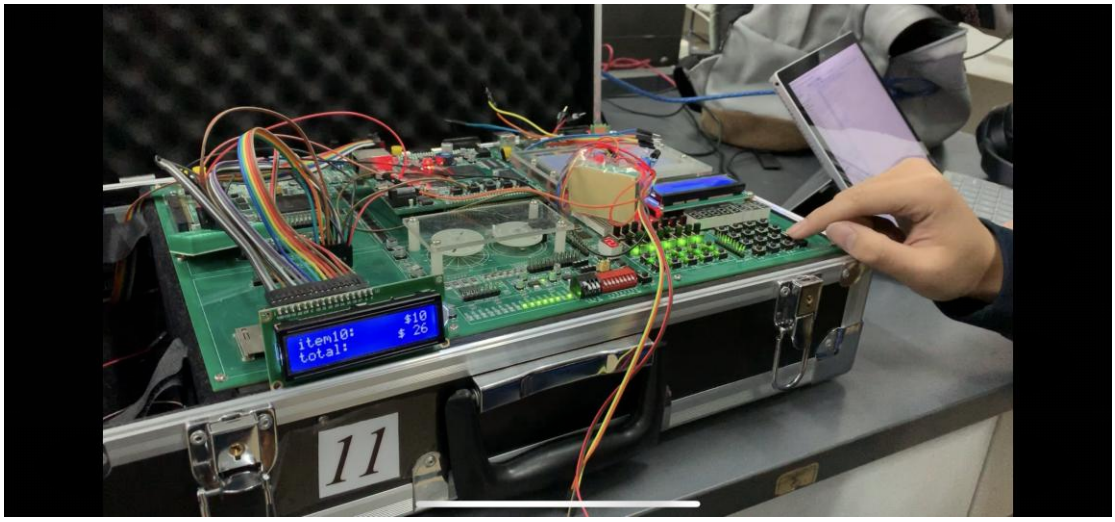







item 9 $ 9
new price $ 4


total: $ 37


item10: $10
total: $ 26

IV. Summary and outlook

In this comprehensive design, the division of tasks of members is as follows:

Heng Cao is responsible for the programming and debugging of programs

Xing-Yuan Xu is responsible for the motor part

Han-Tao Li is responsible for the LCD screen part

Mu-Ze Li is responsible for the IR sensor part

Through the practices on electrical technology in two semesters, we have mastered the basic knowledge, basic methods and techniques required for the experiment. Our ability to connect theory with practice, to analyze problems and to solve problems has also been improved.

In the design of such an integrated system, we have used the circuit knowledge we have learned in the past year, carefully drafted the plan and actively carried out the simulation. In the final circuit construction and code debugging, we were able to calm down to find out the problems in the original design hardware and software, and try to solve the problem. So that we can ultimately achieve our desired results.

Appendix. FPGA source code

1.cash_register.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;


entity cash_register is
    port(   clk             :   in      std_logic;
            KBCol           :   in  std_logic_vector(3 downto 0);
            KBROw           :   out  std_logic_vector(3 downto 0);
            scanin          :   in      std_logic_vector(3 downto 0);
            motor_out       :   out  std_logic;
            motor_pwm       :   out  std_logic;
            reset_lcd       :   in      std_logic;
            oe_l            :   out  std_logic;
            rs_l            :   out  std_logic;
            rw_l            :   out  std_logic;
            data_l          :   out  std_logic_vector(7 downto 0);
            state_o         :   out  std_logic_vector(7 downto 0));
end;


architecture behavioral of cash_register is

    component keyboard is
        port(   clk_k           :   in      std_logic;
                KBCol_k         :   in  std_logic_vector(3 downto 0);
                KBROw_k         :   out  std_logic_vector(3 downto 0);
                output_k        :   out  std_logic_vector(15 downto 0));
    end component;

    component scan is
        port( start_s :     in      std_logic;
                scanin_s:   in      std_logic_vector(3 downto 0);
                output_s:   out  std_logic_vector(3 downto 0));
    end component;

    component motor is
        port(   clk_m           :   in      std_logic;
```

```
            input_m     :   in      std_logic;
            out_m       :   out     std_logic;
            pwmo_m      :   out     std_logic);

    end component;

    component lcd is
        port(   clk_1               :   in      std_logic;
                reset_1             :   in      std_logic;
                oe                  :   out     std_logic;
                rs                  :   out     std_logic;
                rw                  :   out     std_logic;
                data                :   out     std_logic_vector(7 downto 0);
                state_input_1   :   in      std_logic_vector(7 downto 0);
                scan_item_1         :   in      integer;
                scan_item_price :   in      integer;
                total_price         :   in      integer;
                input_flag_1    :   in      std_logic;
                set_scan_item       :   in      integer;
                price2_1            :   in      integer;
                price1_1            :   in      integer);
    end component;

    type key_state is (ka,kb,kc,kd,ke,kf,k1,k2,k3,k4,k5,k6,k7,k8,k9,k0,nul);
    type state is (scan_item_initial, scan_item_scan,
set_price_initial,set_price_scan,set1,set2,setw, pay);
    type item is array(0 to 15) of integer;
    type prices is array(0 to 15) of integer;



    signal key                          :   std_logic_vector(15 downto 0);
    signal pkey                         :   key_state;
    signal scansig                      :   std_logic_vector(3 downto 0);
    signal current_state            :   state:=scan_item_initial;
    signal item_num                 :   item:=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    signal item_price               :
    prices:=(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);
    signal total                        :   integer:=0;
    signal current_item             :   integer;
    signal price,price1,price2      :   integer:=0;
    signal knum                         :   integer;
    signal disp_state               :   std_logic_vector(7 downto 0);
    signal sum1,sum2,sum3           :   integer;
```

```
    signal input_flag                   :    std_logic:='0';
    signal current_scan_item      :    integer;
    signal current_scan_item_price    :    integer;



begin
    key_board:keyboard                                                    port
map(clk_k=>clk,KBCol_k=>KBCol,KBRow_k=>KBRow,output_k=>key);
    SC:scan port map(start_s=>key(0),scanin_s=>scanin,output_s=>scansig);
    MT:motor                                                              port
map(clk_m=>clk,input_m=>key(8),out_m=>motor_out,pwmo_m=>motor_pwm);
    LCD_dis:lcd                                                           port
map(clk_l=>clk,reset_l=>reset_lcd,oe=>oe_l,rs=>rs_l,rw=>rw_l,data=>data_l,state_i
nput_l=>disp_state,scan_item_l=>current_scan_item,

    scan_item_price=>current_scan_item_price,total_price=>total,input_flag_l=>inp
ut_flag,set_scan_item=>current_item,price2_l=>price2,
                                price1_l=>price1);



    keybind:process(key)
    begin
        case key is
            when "1000000000000000"=>pkey<=k1;
                                        knum<=1;
            when "0100000000000000"=>pkey<=k2;
                                        knum<=2;
            when "0010000000000000"=>pkey<=k3;
                                        knum<=3;
            when "0001000000000000"=>pkey<=kc;--set price
            when "0000100000000000"=>pkey<=k4;
                                        knum<=4;
            when "0000010000000000"=>pkey<=k5;
                                        knum<=5;
            when "0000001000000000"=>pkey<=k6;
                                        knum<=6;
            when "0000000100000000"=>pkey<=kd;--pay,motor
            when "0000000010000000"=>pkey<=k7;
                                        knum<=7;
            when "0000000001000000"=>pkey<=k8;
                                        knum<=8;
```

```
            when "0000000000100000"=>pkey<=k9;
                                                knum<=9;
            when "0000000000010000"=>pkey<=ke;--sure
            when "0000000000001000"=>pkey<=ka; --del
            when "0000000000000100"=>pkey<=k0;
                                                knum<=0;
            when "0000000000000010"=>pkey<=kb; --return
            when "0000000000000001"=>pkey<=kf;--start scan
            when "0000000000000000"=>pkey<=nul;
            when others=>null;
        end case;
    end process keybind;



    main:process(clk)
    begin
        if rising_edge(clk) then
            if pkey= nul then
                input_flag<='0';
            end if;
            if input_flag='0' then
                case current_state is
                    when scan_item_initial=>
                        if pkey=kf then
                            case scanin is
                                when "0000"=>item_num(0)<=    item_num(0)+1;

    current_scan_item<=0;
                                when "0001"=>item_num(1)<=    item_num(1)+1;

    current_scan_item<=1;
                                when "0010"=>item_num(2)<=    item_num(2)+1;

    current_scan_item<=2;
                                when "0011"=>item_num(3)<=    item_num(3)+1;

    current_scan_item<=3;
                                when "0100"=>item_num(4)<=    item_num(4)+1;

    current_scan_item<=4;
                                when "0101"=>item_num(5)<=    item_num(5)+1;
```

```
current_scan_item<=5;
                        when "0110"=>item_num(6)<=    item_num(6)+1;
current_scan_item<=6;
                        when "0111"=>item_num(7)<=    item_num(7)+1;
current_scan_item<=7;
                        when "1000"=>item_num(8)<=    item_num(8)+1;
current_scan_item<=8;
                        when "1001"=>item_num(9)<=    item_num(9)+1;
current_scan_item<=9;
                        when "1010"=>item_num(10)<=item_num(10)+1;
current_scan_item<=10;
                        when "1011"=>item_num(11)<=item_num(11)+1;
current_scan_item<=11;
                        when "1100"=>item_num(12)<=item_num(12)+1;
current_scan_item<=12;
                        when "1101"=>item_num(13)<=item_num(13)+1;
current_scan_item<=13;
                        when "1110"=>item_num(14)<=item_num(14)+1;
current_scan_item<=14;
                        when "1111"=>item_num(15)<=item_num(15)+1;
current_scan_item<=15;
                        when others=>null;
                    end case;
                    current_state<=scan_item_scan;
                    input_flag<='1';
                end if;
                if pkey=kc then
                    current_state<=set_price_initial;
                    item_num<=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
                    input_flag<='1';
                end if;
            when scan_item_scan=>
                if pkey=kf then
                    case scanin is
```

```
                    when "0000"=>item_num(0)<=  item_num(0)+1;
current_scan_item<=0;
                    when "0001"=>item_num(1)<=  item_num(1)+1;
current_scan_item<=1;
                    when "0010"=>item_num(2)<=  item_num(2)+1;
current_scan_item<=2;
                    when "0011"=>item_num(3)<=  item_num(3)+1;
current_scan_item<=3;
                    when "0100"=>item_num(4)<=  item_num(4)+1;
current_scan_item<=4;
                    when "0101"=>item_num(5)<=  item_num(5)+1;
current_scan_item<=5;
                    when "0110"=>item_num(6)<=  item_num(6)+1;
current_scan_item<=6;
                    when "0111"=>item_num(7)<=  item_num(7)+1;
current_scan_item<=7;
                    when "1000"=>item_num(8)<=  item_num(8)+1;
current_scan_item<=8;
                    when "1001"=>item_num(9)<=  item_num(9)+1;
current_scan_item<=9;
                    when "1010"=>item_num(10)<=item_num(10)+1;
current_scan_item<=10;
                    when "1011"=>item_num(11)<=item_num(11)+1;
current_scan_item<=11;
                    when "1100"=>item_num(12)<=item_num(12)+1;
current_scan_item<=12;
                    when "1101"=>item_num(13)<=item_num(13)+1;
current_scan_item<=13;
                    when "1110"=>item_num(14)<=item_num(14)+1;
```

```
current_scan_item<=14;
                        when "1111"=>item_num(15)<=item_num(15)+1;

current_scan_item<=15;
                        when others=>null;
                    end case;
                    current_state<=scan_item_scan;
                    input_flag<='1';
                end if;
                if pkey=kd then
                    current_state<=pay;
                    input_flag<='1';
                end if;
                if pkey=kc then
                    current_state<=set_price_initial;
                    item_num<=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
                    input_flag<='1';
                end if;
            when pay=>
                if pkey=kd then
                    current_state<=scan_item_initial;
                    item_num<=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
                    input_flag<='1';
            end if;
            when set_price_initial=>
                if pkey=kf then
                    case scanin is
                        when x"0"=>current_item<=0;
                        when x"1"=>current_item<=1;
                        when x"2"=>current_item<=2;
                        when x"3"=>current_item<=3;
                        when x"4"=>current_item<=4;
                        when x"5"=>current_item<=5;
                        when x"6"=>current_item<=6;
                        when x"7"=>current_item<=7;
                        when x"8"=>current_item<=8;
                        when x"9"=>current_item<=9;
                        when x"a"=>current_item<=10;
                        when x"b"=>current_item<=11;
                        when x"c"=>current_item<=12;
                        when x"d"=>current_item<=13;
                        when x"e"=>current_item<=14;
                        when x"f"=>current_item<=15;
                    end case;
```

```
                    current_state<=set_price_scan;
                    input_flag<='1';
                end if;
                if pkey=kc then
                    current_state<=scan_item_initial;
                    input_flag<='1';
                end if;
            when set_price_scan=>
                if pkey=kf then
                    case scanin is
                        when x"0"=>current_item<=0;
                        when x"1"=>current_item<=1;
                        when x"2"=>current_item<=2;
                        when x"3"=>current_item<=3;
                        when x"4"=>current_item<=4;
                        when x"5"=>current_item<=5;
                        when x"6"=>current_item<=6;
                        when x"7"=>current_item<=7;
                        when x"8"=>current_item<=8;
                        when x"9"=>current_item<=9;
                        when x"a"=>current_item<=10;
                        when x"b"=>current_item<=11;
                        when x"c"=>current_item<=12;
                        when x"d"=>current_item<=13;
                        when x"e"=>current_item<=14;
                        when x"f"=>current_item<=15;
                    end case;
                    current_state<=set_price_scan;
                    input_flag<='1';
                end if;
                if pkey=ke then
                    current_state<=set1;
                    input_flag<='1';
                end if;
                if pkey=kc then
                    current_state<=scan_item_initial;
                    input_flag<='1';
                end if;
            when set1=>
                if pkey=k1 or pkey=k2 or pkey=k3 or pkey=k4 or pkey=k5
or pkey=k6 or pkey=k7 or pkey=k8 or pkey=k9 then
                    price<=knum;
                    price1<=knum;
                    current_state<=set2;
```

```
                input_flag<='1';
            end if;
            if pkey=ke then
                item_price(current_item)<=price;
                price<=0;
                price1<=0;
                price2<=0;
                current_state<=set1;
                input_flag<='1';
            end if;
            if pkey=kb then
                price<=0;
                price1<=0;
                price2<=0;
                current_state<=set_price_initial;
                input_flag<='1';
            end if;
            if pkey=kc then
                current_state<=scan_item_initial;
                input_flag<='1';
            end if;
        when set2=>
            if pkey=k1 or pkey=k2 or pkey=k3 or pkey=k4 or pkey=k5
or pkey=k6 or pkey=k7 or pkey=k8 or pkey=k9 or pkey=k0 then
                price<=price1*10+knum;
                price2<=price1;
                price1<=knum;
                current_state<=setw;
                input_flag<='1';
            end if;
            if pkey=ke then
                item_price(current_item)<=price;
                price<=0;
                price1<=0;
                price2<=0;
                current_state<=set1;
                input_flag<='1';
            end if;
            if pkey=kb then
                price<=0;
                price1<=0;
                price2<=0;
                current_state<=set_price_initial;
                input_flag<='1';
```

```
                    end if;
                    if pkey=ka then
                        price1<=0;
                        price<=0;
                        price2<=0;
                        current_state<=set1;
                        input_flag<='1';
                    end if;
                    if pkey=kc then
                        current_state<=scan_item_initial;
                        input_flag<='1';
                    end if;
                when setw=>
                    if pkey=ke then
                        item_price(current_item)<=price;
                        price<=0;
                        price1<=0;
                        price2<=0;
                        current_state<=set1;
                        input_flag<='1';
                    end if;
                    if pkey=kb then
                        price<=0;
                        price1<=0;
                        price2<=0;
                        current_state<=set_price_initial;
                        input_flag<='1';
                    end if;
                    if pkey=ka then
                        price<=price2;
                        price1<=price2;
                        price2<=0;
                        current_state<=set2;
                        input_flag<='1';
                    end if;
                    if pkey=kc then
                        current_state<=scan_item_initial;
                        input_flag<='1';
                    end if;
            end case;
        end if;
    end if;
end process main;
```

```
    state_transfer:process(current_state)
    begin
        case current_state is
                    when scan_item_initial=>disp_state<="00000001";
                    when scan_item_scan=>disp_state<="00000010";
                    when set_price_initial=>disp_state<="00000100";
                    when set_price_scan=>disp_state<="00001000";
                    when set1=>disp_state<="00010000";
                    when set2=>disp_state<="00100000";
                    when setw=>disp_state<="01000000";
                when pay=>disp_state<="10000000";
                when others=>null;
        end case;
        state_o<=disp_state;
    end process state_transfer;




    item_transfer:process(current_scan_item,current_state)
    begin
        if current_state=scan_item_initial or current_state=scan_item_scan then
            current_scan_item_price<=item_price(current_scan_item);
        else
            if current_state=set_price_initial or current_state=set_price_scan or
current_state=set1 or current_state=set2 or current_state=setw then
                current_scan_item_price<=item_price(current_item);
            end if;
        end if;
    end process item_transfer;

    money_count:process(item_num)
    begin

    total<=item_num(0)*item_price(0)+item_num(1)*item_price(1)+item_num(2)*it
em_price(2)

    +item_num(3)*item_price(3)+item_num(4)*item_price(4)+item_num(5)*item_pr
ice(5)

    +item_num(6)*item_price(6)+item_num(7)*item_price(7)+item_num(8)*item_pr
ice(8)

    +item_num(9)*item_price(9)+item_num(10)*item_price(10)+item_num(11)*ite
```

```
m_price(11)

    +item_num(12)*item_price(12)+item_num(13)*item_price(13)+item_num(14)*it
em_price(14)+item_num(15)*item_price(15);
    end process money_count;



end behavioral;
```

2. lcd.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;


entity lcd is
    generic( N      :   integer:=200;
             delay  :   integer:=100);

    port(   clk_1                   :   in      std_logic;
            reset_1             :   in      std_logic;
            oe                      :   out  std_logic;
            rs                      :   out  std_logic;
            rw                      :   out  std_logic;
            data                :   out  std_logic_vector(7 downto 0);
            state_input_1   :   in      std_logic_vector(7 downto 0);
            scan_item_1         :   in      integer;
            scan_item_price :   in      integer;
            total_price         :   in      integer;
            input_flag_1    :   in      std_logic;
            set_scan_item       :   in      integer;
            price2_1            :   in      integer;
            price1_1            :   in      integer);

end;

architecture behavioral of lcd is
    type                                                                    state
is(clear_lcd,entry_set,display_set,function_set,position_set1,write_data1,position_set
2, write_data2,stop);
    type text_short is array(0 to 23) of std_logic_vector(7 downto 0);
    type text_long is array(0 to 31) of std_logic_vector(7 downto 0);
    type dis_state is(initial,scanning,set_scan_initial,set_scanning,set1,set2,setw,pay);




    constant scan_item_initial:     text_short:=(
    x"57",x"65",x"6c",x"63",x"6f",x"6d",x"65",x"21",
```

```
    x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"20",x"20",x"20");
    constant set_initial_text   :    text_long:=(
    x"20",x"20",x"20",x"20",x"73",x"63",x"61",x"6e",x"20",x"69",x"74",x"65",x"6d
",x"20",x"20",x"20",

    x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"20",x"20",x"20");
    signal set_scanning_text   :    text_long:=(
    x"69",x"74",x"65",x"6d",x"30",x"30",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"24",x"30",x"30",

    x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"20",x"20",x"20");
    signal scanning_text        :    text_long:=(
    x"69",x"74",x"65",x"6d",x"30",x"30",x"3a",x"20",x"20",x"20",x"20",x"20",x"20
",x"24",x"30",x"30",

    x"74",x"6f",x"74",x"61",x"6c",x"3a",x"20",x"20",x"20",x"20",x"20",x"20",x"24",
x"30",x"30",x"30");
    signal set1_text            :    text_long:=(
    x"69",x"74",x"65",x"6d",x"30",x"30",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"24",x"30",x"30",

    x"6e",x"65",x"77",x"20",x"70",x"72",x"69",x"63",x"65",x"20",x"20",x"20",x"20
",x"24",x"20",x"30");
    signal set2_text            :    text_long:=(
    x"69",x"74",x"65",x"6d",x"30",x"30",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"24",x"30",x"30",

    x"6e",x"65",x"77",x"20",x"70",x"72",x"69",x"63",x"65",x"20",x"20",x"20",x"20
",x"24",x"20",x"30");
    signal setw_text            :    text_long:=(
    x"69",x"74",x"65",x"6d",x"30",x"30",x"20",x"20",x"20",x"20",x"20",x"20",x"20
",x"24",x"30",x"30",

    x"6e",x"65",x"77",x"20",x"70",x"72",x"69",x"63",x"65",x"20",x"20",x"20",x"20
",x"24",x"30",x"30");

    signal pay_text             :    text_long:=(
    x"74",x"6f",x"74",x"61",x"6c",x"3a",x"20",x"20",x"20",x"20",x"20",x"20",x"24",
x"30",x"30",x"30",

    x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20",x"20
```

```
",x"20",x"20",x"20");
    signal clk_250Khz,clk_1Hz   :   std_logic;
    signal cnt1,cnt2            :   integer range 0 to 200000;
    signal current_state        :   state:=clear_lcd;
    signal current_dis_state    :   dis_state:=initial;
    signal state_change         :   std_logic;
    signal scan_item            :   integer;

begin
    lcd_clk:process(clk_l,reset_l,total_price)
        variable c1:integer range 0 to 100;
        variable c2:integer range 0 to 50000000;
        variable clk0,clk1:std_logic;
    begin
        if(reset_l='0')then
            c1:=0;
            c2:=0;
        else
            if(clk_l'event and clk_l='1')then
                if(c1=N/2-1)then
                    c1:=0;
                    clk0:=not clk0;
                else
                    c1:=c1+1;
                end if;
                if(c2=50000000/2-1)then
                    c2:=0;
                    clk1:=not clk1;
                else
                    c2:=c2+1;
                end if;
            end if;
        end if;
        clk_250Khz<=clk0;
        clk_1hz<=clk1;
    end process lcd_clk;


    display_state:process(state_input_l)
    begin
        case state_input_l is
            when "00000001"=>current_dis_state<=initial;
            when "00000010"=>current_dis_state<=scanning;
            when "00000100"=>current_dis_state<=set_scan_initial;
```

```
            when "00001000"=>current_dis_state<=set_scanning;
            when "00010000"=>current_dis_state<=set1;
            when "00100000"=>current_dis_state<=set2;
            when "01000000"=>current_dis_state<=setw;
            when "10000000"=>current_dis_state<=pay;
            when others=>null;
        end case;
    end process display_state;


    control:process(clk_250Khz,reset_l)
    begin
        if(reset_l='0')then
            current_state<=clear_lcd;
            cnt1<=0;
            cnt2<=0;
        else
            if rising_edge(clk_250Khz)then
                if input_flag_l='1' then
                    current_state<=clear_lcd;
                end if;
                case current_dis_state is
                    when initial=>
                        case current_state is
                            when clear_lcd=>
                                oe<='1';
                                rs<='0';
                                rw<='0';
                                data<=x"01";
                                cnt1<=cnt1+1;
                                if(cnt1>delay*1 and cnt1<=delay*6)then
                                    oe<='0';
                                else
                                    oe<='1';
                                end if;
                                if(cnt1=delay*7)then
                                    current_state<=entry_set;
                                    cnt1<=0;
                                end if;
                            when entry_set=>
                                oe<='1';
                                rs<='0';
                                rw<='0';
                                data<=x"06";
```

```
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=display_set;
                cnt1<=0;
            end if;
        when display_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"0C";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=function_set;
                cnt1<=0;
            end if;
        when function_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"38";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=position_set1;
                cnt1<=0;
            end if;
        when position_set1=>
            oe<='1';
            rs<='0';
            rw<='0';
```

```
            data<=x"84";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data1;
                cnt1<=0;
            end if;
        when write_data1=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=7)then
                data<=scan_item_initial(cnt2);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
            oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data1;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
            else
                cnt2<=0;
                current_state<=position_set2;
            end if;
        when position_set2=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"C0";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
```

```
                    current_state<=write_data2;
                    cnt1<=0;
                end if;
            when write_data2=>
                oe<='1';
                rs<='1';
                rw<='0';
                if(cnt2<=15)then
                    data<=scan_item_initial(cnt2+8);
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=write_data2;
                        cnt1<=0;
                        cnt2<=cnt2+1;
                    end if;
                else
                    cnt2<=0;
                    current_state<=position_set1;
                end if;
            when stop=>null;
        end case;
    when scanning=>
        case current_state is
            when clear_lcd=>
                oe<='1';
                rs<='0';
                rw<='0';
                data<=x"01";
                cnt1<=cnt1+1;
                if(cnt1>delay*1 and cnt1<=delay*6)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*7)then
                    current_state<=entry_set;
                    cnt1<=0;
                end if;
            when entry_set=>
```

```
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"06";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=display_set;
                cnt1<=0;
            end if;
        when display_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"0C";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=function_set;
                cnt1<=0;
            end if;
        when function_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"38";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=position_set1;
                cnt1<=0;
            end if;
```

```
when position_set1=>
    oe<='1';
    rs<='0';
    rw<='0';
    data<=x"80";
    cnt1<=cnt1+1;
    if(cnt1>delay and cnt1<=delay*2)then
        oe<='0';
    else
        oe<='1';
    end if;
    if(cnt1=delay*3)then
        current_state<=write_data1;
        cnt1<=0;
    end if;
when write_data1=>
    oe<='1';
    rs<='1';
    rw<='0';
    if(cnt2<=15)then
        data<=scanning_text(cnt2);
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
    oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=write_data1;
            cnt1<=0;
            cnt2<=cnt2+1;
        end if;
    else
        cnt2<=0;
        current_state<=position_set2;
    end if;
when position_set2=>
    oe<='1';
    rs<='0';
    rw<='0';
    data<=x"C0";
    cnt1<=cnt1+1;
    if(cnt1>delay and cnt1<=delay*2)then
        oe<='0';
```

```
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=write_data2;
                        cnt1<=0;
                    end if;
                when write_data2=>
                    oe<='1';
                    rs<='1';
                    rw<='0';
                    if(cnt2<=15)then
                        data<=scanning_text(cnt2+16);
                        cnt1<=cnt1+1;
                        if(cnt1>delay and cnt1<=delay*2)then
                            oe<='0';
                        else
                            oe<='1';
                        end if;
                        if(cnt1=delay*3)then
                            current_state<=write_data2;
                            cnt1<=0;
                            cnt2<=cnt2+1;
                        end if;
                    else
                        cnt2<=0;
                        current_state<=position_set1;
                    end if;
                when stop=>null;
            end case;
        when set_scan_initial=>
            case current_state is
                when clear_lcd=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"01";
                    cnt1<=cnt1+1;
                    if(cnt1>delay*1 and cnt1<=delay*6)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*7)then
```

```
                current_state<=entry_set;
                cnt1<=0;
            end if;
        when entry_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"06";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=display_set;
                cnt1<=0;
            end if;
        when display_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"0C";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=function_set;
                cnt1<=0;
            end if;
        when function_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"38";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
```

```
        if(cnt1=delay*3)then
            current_state<=position_set1;
            cnt1<=0;
        end if;
    when position_set1=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"80";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=write_data1;
            cnt1<=0;
        end if;
    when write_data1=>
        oe<='1';
        rs<='1';
        rw<='0';
        if(cnt2<=15)then
            data<=set_initial_text(cnt2);
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
        oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data1;
                cnt1<=0;
                cnt2<=cnt2+1;
            end if;
        else
            cnt2<=0;
            current_state<=position_set2;
        end if;
    when position_set2=>
        oe<='1';
        rs<='0';
        rw<='0';
```

```
                    data<=x"C0";
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=write_data2;
                        cnt1<=0;
                    end if;
                when write_data2=>
                    oe<='1';
                    rs<='1';
                    rw<='0';
                    if(cnt2<=15)then
                        data<=set_initial_text(cnt2+16);
                        cnt1<=cnt1+1;
                        if(cnt1>delay and cnt1<=delay*2)then
                            oe<='0';
                        else
                            oe<='1';
                        end if;
                        if(cnt1=delay*3)then
                            current_state<=write_data2;
                            cnt1<=0;
                            cnt2<=cnt2+1;
                        end if;
                    else
                        cnt2<=0;
                        current_state<=position_set1;
                    end if;
                when stop=>null;
            end case;
        when set_scanning=>
            case current_state is
                when clear_lcd=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"01";
                    cnt1<=cnt1+1;
                    if(cnt1>delay*1 and cnt1<=delay*6)then
                        oe<='0';
```

```
        else
            oe<='1';
        end if;
        if(cnt1=delay*7)then
            current_state<=entry_set;
            cnt1<=0;
        end if;
    when entry_set=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"06";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=display_set;
            cnt1<=0;
        end if;
    when display_set=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"0C";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=function_set;
            cnt1<=0;
        end if;
    when function_set=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"38";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
```

```
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=position_set1;
            cnt1<=0;
        end if;
    when position_set1=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"80";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=write_data1;
            cnt1<=0;
        end if;
    when write_data1=>
        oe<='1';
        rs<='1';
        rw<='0';
        if(cnt2<=15)then
            data<=set_scanning_text(cnt2);
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
        oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data1;
                cnt1<=0;
                cnt2<=cnt2+1;
            end if;
        else
            cnt2<=0;
            current_state<=position_set2;
        end if;
```

```
                when position_set2=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"C0";
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=write_data2;
                        cnt1<=0;
                    end if;
                when write_data2=>
                    oe<='1';
                    rs<='1';
                    rw<='0';
                    if(cnt2<=15)then
                        data<=set_scanning_text(cnt2+16);
                        cnt1<=cnt1+1;
                        if(cnt1>delay and cnt1<=delay*2)then
                            oe<='0';
                        else
                            oe<='1';
                        end if;
                        if(cnt1=delay*3)then
                            current_state<=write_data2;
                            cnt1<=0;
                            cnt2<=cnt2+1;
                        end if;
                    else
                        cnt2<=0;
                        current_state<=position_set1;
                    end if;
                when stop=>null;
            end case;
        when set1=>
            case current_state is
                when clear_lcd=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
```

```
            data<=x"01";
            cnt1<=cnt1+1;
            if(cnt1>delay*1 and cnt1<=delay*6)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*7)then
                current_state<=entry_set;
                cnt1<=0;
            end if;
        when entry_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"06";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=display_set;
                cnt1<=0;
            end if;
        when display_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"0C";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=function_set;
                cnt1<=0;
            end if;
        when function_set=>
            oe<='1';
            rs<='0';
```

```
            rw<='0';
            data<=x"38";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=position_set1;
                cnt1<=0;
            end if;
        when position_set1=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"80";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data1;
                cnt1<=0;
            end if;
        when write_data1=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=15)then
                data<=set1_text(cnt2);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
            oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data1;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
```

```
                else
                    cnt2<=0;
                    current_state<=position_set2;
                end if;
            when position_set2=>
                oe<='1';
                rs<='0';
                rw<='0';
                data<=x"C0";
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data2;
                    cnt1<=0;
                end if;
            when write_data2=>
                oe<='1';
                rs<='1';
                rw<='0';
                if(cnt2<=15)then
                    data<=set1_text(cnt2+16);
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=write_data2;
                        cnt1<=0;
                        cnt2<=cnt2+1;
                    end if;
                else
                    cnt2<=0;
                    current_state<=position_set1;
                end if;
            when stop=>null;
        end case;
    when set2=>
        case current_state is
```

```
when clear_lcd=>
    oe<='1';
    rs<='0';
    rw<='0';
    data<=x"01";
    cnt1<=cnt1+1;
    if(cnt1>delay*1 and cnt1<=delay*6)then
        oe<='0';
    else
        oe<='1';
    end if;
    if(cnt1=delay*7)then
        current_state<=entry_set;
        cnt1<=0;
    end if;
when entry_set=>
    oe<='1';
    rs<='0';
    rw<='0';
    data<=x"06";
    cnt1<=cnt1+1;
    if(cnt1>delay and cnt1<=delay*2)then
        oe<='0';
    else
        oe<='1';
    end if;
    if(cnt1=delay*3)then
        current_state<=display_set;
        cnt1<=0;
    end if;
when display_set=>
    oe<='1';
    rs<='0';
    rw<='0';
    data<=x"0C";
    cnt1<=cnt1+1;
    if(cnt1>delay and cnt1<=delay*2)then
        oe<='0';
    else
        oe<='1';
    end if;
    if(cnt1=delay*3)then
        current_state<=function_set;
        cnt1<=0;
```

```
        end if;
    when function_set=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"38";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=position_set1;
            cnt1<=0;
        end if;
    when position_set1=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"80";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=write_data1;
            cnt1<=0;
        end if;
    when write_data1=>
        oe<='1';
        rs<='1';
        rw<='0';
        if(cnt2<=15)then
            data<=set2_text(cnt2);
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
        oe<='1';
            end if;
            if(cnt1=delay*3)then
```

```
                    current_state<=write_data1;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
            else
                cnt2<=0;
                current_state<=position_set2;
            end if;
        when position_set2=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"C0";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data2;
                cnt1<=0;
            end if;
        when write_data2=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=15)then
                data<=set2_text(cnt2+16);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data2;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
            else
                cnt2<=0;
                current_state<=position_set1;
            end if;
```

```
                when stop=>null;
            end case;
        when setw=>
            case current_state is
                when clear_lcd=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"01";
                    cnt1<=cnt1+1;
                    if(cnt1>delay*1 and cnt1<=delay*6)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*7)then
                        current_state<=entry_set;
                        cnt1<=0;
                    end if;
                when entry_set=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"06";
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
                    end if;
                    if(cnt1=delay*3)then
                        current_state<=display_set;
                        cnt1<=0;
                    end if;
                when display_set=>
                    oe<='1';
                    rs<='0';
                    rw<='0';
                    data<=x"0C";
                    cnt1<=cnt1+1;
                    if(cnt1>delay and cnt1<=delay*2)then
                        oe<='0';
                    else
                        oe<='1';
```

```
            end if;
            if(cnt1=delay*3)then
                current_state<=function_set;
                cnt1<=0;
            end if;
        when function_set=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"38";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=position_set1;
                cnt1<=0;
            end if;
        when position_set1=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"80";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data1;
                cnt1<=0;
            end if;
        when write_data1=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=15)then
                data<=setw_text(cnt2);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
```

```
                else
            oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data1;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
            else
                cnt2<=0;
                current_state<=position_set2;
            end if;
        when position_set2=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"C0";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data2;
                cnt1<=0;
            end if;
        when write_data2=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=15)then
                data<=setw_text(cnt2+16);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data2;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
```

```
                else
                    cnt2<=0;
                    current_state<=position_set1;
                end if;
            when stop=>null;
        end case;
    when pay=>
        case current_state is
            when clear_lcd=>
                oe<='1';
                rs<='0';
                rw<='0';
                data<=x"01";
                cnt1<=cnt1+1;
                if(cnt1>delay*1 and cnt1<=delay*6)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*7)then
                    current_state<=entry_set;
                    cnt1<=0;
                end if;
            when entry_set=>
                oe<='1';
                rs<='0';
                rw<='0';
                data<=x"06";
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=display_set;
                    cnt1<=0;
                end if;
            when display_set=>
                oe<='1';
                rs<='0';
                rw<='0';
                data<=x"0C";
                cnt1<=cnt1+1;
```

```
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=function_set;
            cnt1<=0;
        end if;
    when function_set=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"38";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=position_set1;
            cnt1<=0;
        end if;
    when position_set1=>
        oe<='1';
        rs<='0';
        rw<='0';
        data<=x"80";
        cnt1<=cnt1+1;
        if(cnt1>delay and cnt1<=delay*2)then
            oe<='0';
        else
            oe<='1';
        end if;
        if(cnt1=delay*3)then
            current_state<=write_data1;
            cnt1<=0;
        end if;
    when write_data1=>
        oe<='1';
        rs<='1';
        rw<='0';
        if(cnt2<=15)then
```

```
                data<=pay_text(cnt2);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
            oe<='1';
                end if;
                if(cnt1=delay*3)then
                    current_state<=write_data1;
                    cnt1<=0;
                    cnt2<=cnt2+1;
                end if;
            else
                cnt2<=0;
                current_state<=position_set2;
            end if;
        when position_set2=>
            oe<='1';
            rs<='0';
            rw<='0';
            data<=x"C0";
            cnt1<=cnt1+1;
            if(cnt1>delay and cnt1<=delay*2)then
                oe<='0';
            else
                oe<='1';
            end if;
            if(cnt1=delay*3)then
                current_state<=write_data2;
                cnt1<=0;
            end if;
        when write_data2=>
            oe<='1';
            rs<='1';
            rw<='0';
            if(cnt2<=15)then
                data<=pay_text(cnt2+16);
                cnt1<=cnt1+1;
                if(cnt1>delay and cnt1<=delay*2)then
                    oe<='0';
                else
                    oe<='1';
                end if;
                if(cnt1=delay*3)then
```

```
                                        current_state<=write_data2;
                                        cnt1<=0;
                                        cnt2<=cnt2+1;
                                    end if;
                                else
                                    cnt2<=0;
                                    current_state<=position_set1;
                                end if;
                            when stop=>null;
                        end case;
                    end case;
                end if;
            end if;
        end process control;


        num_change:process(set_scan_item,scan_item_price,total_price,scan_item_l,curr
ent_dis_state,price1_l,price2_l)
            variable n1             :   integer;
            variable m1,m12 :   integer;
        begin
            if current_dis_state=scanning then
                scan_item<=scan_item_l;
            else
                if current_dis_state=set_scanning then
                    scan_item<=set_scan_item;
                end if;
            end if;
            case scan_item is
                when 0=>
                    scanning_text(4)<=x"20";
                    scanning_text(5)<=x"30";
                    set_scanning_text(4)<=x"20";
                    set_scanning_text(5)<=x"30";
                when 1=>
                    scanning_text(4)<=x"20";
                    scanning_text(5)<=x"31";
                    set_scanning_text(4)<=x"20";
                    set_scanning_text(5)<=x"31";
                when 2=>
                    scanning_text(4)<=x"20";
                    scanning_text(5)<=x"32";
                    set_scanning_text(4)<=x"20";
                    set_scanning_text(5)<=x"32";
```

```
when 3=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"33";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"33";
when 4=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"34";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"34";
when 5=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"35";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"35";
when 6=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"36";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"36";
when 7=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"37";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"37";
when 8=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"38";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"38";
when 9=>
    scanning_text(4)<=x"20";
    scanning_text(5)<=x"39";
    set_scanning_text(4)<=x"20";
    set_scanning_text(5)<=x"39";
when 10=>
    scanning_text(4)<=x"31";
    scanning_text(5)<=x"30";
    set_scanning_text(4)<=x"31";
    set_scanning_text(5)<=x"30";
when 11=>
    scanning_text(4)<=x"31";
    scanning_text(5)<=x"31";
    set_scanning_text(4)<=x"31";
```

```
                set_scanning_text(5)<=x"31";
            when 12=>
                scanning_text(4)<=x"31";
                scanning_text(5)<=x"32";
                set_scanning_text(4)<=x"31";
                set_scanning_text(5)<=x"32";
            when 13=>
                scanning_text(4)<=x"31";
                scanning_text(5)<=x"33";
                set_scanning_text(4)<=x"31";
                set_scanning_text(5)<=x"33";
            when 14=>
                scanning_text(4)<=x"31";
                scanning_text(5)<=x"34";
                set_scanning_text(4)<=x"31";
                set_scanning_text(5)<=x"34";
            when 15=>
                scanning_text(4)<=x"31";
                scanning_text(5)<=x"35";
                set_scanning_text(4)<=x"31";
                set_scanning_text(5)<=x"35";
            when others=>null;
        end case;
        if scan_item_price<=9 then
            scanning_text(14)<=x"20";
            scanning_text(15)<=conv_std_logic_vector(48+scan_item_price,8);
            set_scanning_text(14)<=scanning_text(14);
            set_scanning_text(15)<=scanning_text(15);
        else
            n1:=scan_item_price mod 10;

    scanning_text(14)<=conv_std_logic_vector(48+(scan_item_price-n1)/10,8);
            scanning_text(15)<=conv_std_logic_vector(48+n1,8);
            set_scanning_text(14)<=scanning_text(14);
            set_scanning_text(15)<=scanning_text(15);
        end if;
        if total_price>999 then
            scanning_text(29)<=x"39";
            scanning_text(30)<=x"39";
            scanning_text(31)<=x"39";
        else
            if total_price>99 then
                m12:=total_price mod 100;
                m1:=m12 mod 10;
```

```
    scanning_text(29)<=conv_std_logic_vector(48+(total_price-m12)/100,8);
                scanning_text(30)<=conv_std_logic_vector(48+(m12-m1)/10,8);
                scanning_text(31)<=conv_std_logic_vector(48+m1,8);
            else
                if total_price>9 then
                    scanning_text(29)<=x"20";
                    m1:=total_price mod 10;

    scanning_text(30)<=conv_std_logic_vector(48+(total_price-m1)/10,8);
                    scanning_text(31)<=conv_std_logic_vector(48+m1,8);
                else
                    scanning_text(29)<=x"20";
                    scanning_text(30)<=x"20";
                    scanning_text(31)<=conv_std_logic_vector(48+total_price,8);
                end if;
            end if;
        end if;
        set1_text(4)<=scanning_text(4);
        set1_text(5)<=scanning_text(5);
        set1_text(14)<=scanning_text(14);
        set1_text(15)<=scanning_text(15);
        set2_text(4)<=scanning_text(4);
        set2_text(5)<=scanning_text(5);
        set2_text(14)<=scanning_text(14);
        set2_text(15)<=scanning_text(15);
        set2_text(31)<=conv_std_logic_vector(48+price1_l,8);
        setw_text(4)<=scanning_text(4);
        setw_text(5)<=scanning_text(5);
        setw_text(14)<=scanning_text(14);
        setw_text(15)<=scanning_text(15);
        setw_text(31)<=conv_std_logic_vector(48+price1_l,8);
        setw_text(30)<=conv_std_logic_vector(48+price2_l,8);
        pay_text(13)<=scanning_text(29);
        pay_text(14)<=scanning_text(30);
        pay_text(15)<=scanning_text(31);
    end process num_change;




end behavioral;
```

3. keyboard.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity keyboard is
    port(   clk_k          :   in      std_logic;
            KBCol_k        :   in  std_logic_vector(3 downto 0);
            KBROw_k        :   out std_logic_vector(3 downto 0);
            output_k       :   out std_logic_vector(15 downto 0));
end;


architecture behavioral of keyboard is
    type key is (ka,kb,kc,kd,ke,kf,k1,k2,k3,k4,k5,k6,k7,k8,k9,k0,nul);

    signal count :   std_logic_vector(1 downto 0);
    signal pkey      :   key :=nul;
begin

    cont:process(clk_k)
    begin
        if rising_edge(clk_k) then
            count<=count+1;
        end if;
    end process cont;

    row:process(clk_k)
    begin
        if rising_edge(clk_k) then
            case count is
                when "00"=>KBRow_k<="0111";
                when "01"=>KBRow_k<="1011";
                when "10"=>KBRow_k<="1101";
                when "11"=>KBRow_k<="1110";
                when others=>KBRow_k<="1111";
            end case;
        end if;
    end process row;

column:process(clk_k)
```

```
    variable key_tmp:    std_logic;
    variable sta     :   std_logic_vector(1 downto 0);
begin
    if rising_edge(clk_k) then
        if count=sta then
            if key_tmp='0' then
                pkey<=nul;
            end if;
        end if;
        case count is
            when "00"=>
                case KBCol_k is
                    when "1110"=>  pkey<=k0;--
                                   key_tmp:='1';
                                   sta:="00";
                    when "1101"=>  pkey<=k1;--
                                   key_tmp:='1';
                                   sta:="00";
                    when "1011"=>  pkey<=k2;--
                                   key_tmp:='1';
                                   sta:="00";
                    when "0111"=>  pkey<=k3;--
                                   key_tmp:='1';
                                   sta:="00";
                    when others=>  key_tmp:='0';
                end case;
                when "01"=>
                case KBCol_k is
                    when "1110"=>  pkey<=kc;--
                                   key_tmp:='1';
                                   sta:="01";
                    when "1101"=>  pkey<=kd;--
                                   key_tmp:='1';
                                   sta:="01";
                    when "1011"=>  pkey<=ke;--
                                   key_tmp:='1';
                                   sta:="01";
                    when "0111"=>  pkey<=kf;--
                                   key_tmp:='1';
                                   sta:="01";
                    when others=>  key_tmp:='0';
                end case;
            when "10"=>
                case KBCol_k is
```

```
                    when "1110"=>  pkey<=k8;--
                                   key_tmp:='1';
                                   sta:="10";
                    when "1101"=>  pkey<=k9;--
                                   key_tmp:='1';
                                   sta:="10";
                    when "1011"=>  pkey<=ka;--
                                   key_tmp:='1';
                                   sta:="10";
                    when "0111"=>  pkey<=kb;--
                                   key_tmp:='1';
                                   sta:="10";
                    when others=>  key_tmp:='0';
                end case;
            when "11"=>
                case KBCol_k is
                    when "1110"=>  pkey<=k4;--
                                   key_tmp:='1';
                                   sta:="11";
                    when "1101"=>  pkey<=k5;--
                                   key_tmp:='1';
                                   sta:="11";
                    when "1011"=>  pkey<=k6;--
                                   key_tmp:='1';
                                   sta:="11";
                    when "0111"=>  pkey<=k7;--
                                   key_tmp:='1';
                                   sta:="11";
                    when others=>  key_tmp:='0';
                end case;
            when others=>key_tmp:='0';
        end case;
    end if;
end process column;


key_out:process(pkey)
begin
    case pkey is
        when k1=>output_k<="1000000000000000";
        when k2=>output_k<="0100000000000000";
        when k3=>output_k<="0010000000000000";
        when kc=>output_k<="0001000000000000";
        when k4=>output_k<="0000100000000000";
```

```
            when k5=>output_k<="0000010000000000";
            when k6=>output_k<="0000001000000000";
            when kd=>output_k<="0000000100000000";
            when k7=>output_k<="0000000010000000";
            when k8=>output_k<="0000000001000000";
            when k9=>output_k<="0000000000100000";
            when ke=>output_k<="0000000000010000";
            when ka=>output_k<="0000000000001000";
            when k0=>output_k<="0000000000000100";
            when kb=>output_k<="0000000000000010";
            when kf=>output_k<="0000000000000001";
            when nul=>output_k<="0000000000000000";
        end case;
    end process key_out;
end behavioral;
```

4. motor.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;


entity motor is
    port(   clk_m          :   in      std_logic;
            input_m    :   in      std_logic;
            out_m          :   out  std_logic;
            pwmo_m         :   out  std_logic;
            motor_start  :   in   std_logic);
end;

architecture behavioral of motor is
    signal pwm  :   std_logic;
    signal m    :   integer:=0;
    signal o    :   std_logic:='1';
    signal pwm_o    :   std_logic:='0';
    signal direction:  std_logic:='1';
begin
    set_pwm:process(clk_m)
    variable n  :   integer:=0;
    begin
        if rising_edge(clk_m) then
            if n=50 then
                pwm<=not pwm;
            end if;
            if n=100 then
                pwm<=not pwm;
                n:=0;
            end if;
            n:=n+1;
        end if;
    end process set_pwm;


    output:process(clk_m)
    begin
        if rising_edge(clk_m) then
            if pwm_o='0' then
```

```
                if input_m='1' then
                    if direction='1' then
                        out_m<='1';
                        pwm_o<='1';
                        pwmo_m<=pwm;
                    else
                        out_m<='0';
                        pwm_o<='1';
                        pwmo_m<=pwm;
                    end if;
                end if;
            else
                m<=m+1;
                if m>1*24000000 then
                    pwm_o<='0';
                    m<=0;
                    pwmo_m<='0';
                    direction<=not direction;
                end if;
            end if;
        end if;
    end process output;

end behavioral;
```

5. scan.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity scan is
    port( start_s :    in        std_logic;
          scanin_s:    in        std_logic_vector(3 downto 0);
          output_s:    out  std_logic_vector(3 downto 0));
end;


architecture behavioral of scan is
begin

    process(start_s)
    begin
        if rising_edge(start_s) then
            output_s<=scanin_s;
        end if;
    end process;
end behavioral;
```