Bilkent University

EE102 – 04
Term Project

Device for Calculating the Area of a Rectangular Room

Mehmet Emin Yakşi – 22203637

19.05.2024

Purpose:

The aim of this project is to design a device that can calculate the area of a rectangular room after it is placed in the corner of the room with a Basys3 FPGA board and VHDL.

Methodology:

First of all I decided on my components. In this project I need to take two measurements for two sides of the room. Therefore, I need a 90 degrees movement between two measurements. I used an SG90 servo motor to do this turn. Another component that I used is HC-SR04 ultrasonic distance sensor. With this ultrasonic sensor I can measure the two sides of the room. I combined these two components by placing an ultrasonic sensor on the top of the servo motor. In this way I completed the mechanical part of my project. In the next step I need to show corresponding measurements to the user. I used a VGA monitor to display measurements. I also used 4 seven-segment displays to show the area. Then I wrote my VHDL codes according to ”design specifications” part. Finally I made pin connections and generated bitstream file then uploaded it on Basys3 FPGA board.

Design Specifications:

In this project I have following modules:

1. Servo:

   clk input std_logic;
   sw: input std_logic;
   pwm: output std_logic;

   This module takes inputs as “sw”. “1” indicates the initial direction of servo and “2” indicates the direction of servo after it turned 90 degrees. The working

principle of servo depends on the “pwm” signal that we applied to its signal pin. According to the datasheet, I need to generate waveforms which have a 20ms period. The direction of servo depends on when our signal is HIGH at the beginning between 1-2 ms. SG90 can rotate 180 degrees and we divide 1 ms to 180 degrees. If we apply 1 ms high it becomes 0 degrees. Similarly if we apply 2 ms HIGH it becomes 180 degrees. I divided the 100Mhz clock by 50 to generate a 20 ms period which corresponds to 50hz. In each wave I decided the waveforms by calculating the necessary timing to turn the servo 90 degrees.

2. HCSR04:

   clk: in std_logic;
   echo: in std_logic;
   trig: out std_logic;
   distance: out std_logic_vector(7 downto 0);

This module makes distance measurement and gives a distance output. The working principle of an ultrasonic sensor depends upon trigger output and echo input. It works with pwm signals. Datasheet recommends a 60 ms measurement cycle which is the period of pwm signals applied to the trigger pin. When we apply 10 us high to the trigger pin it sends ultrasonic waves and makes the echo pin high. When it reaches the sensor again it makes the echo pin low. This signal gives us the time interval of the wave traveled. We can use this time and speed of sound in the air and calculate the distance. I used the formula uS/58 = centimeters.

3. Binary_bcd:

   clk, reset: in std_logic;
   binary_in: in std_logic_vector(15 downto 0);
   bcd0, bcd1, bcd2, bcd3, bcd4: out std_logic_vector(3 downto 0);

This module takes 16 bits binary input and converts it to binary coded decimal form with shift left add 3 algorithm. In this form each decimal digit is coded by 4 bits of binary number. I used this bcd output to convert my area std_logic_vector and show it on a seven-segment display. This module is combined with BCD_to sevenseg module and I can decide the seven segment cathode outputs.

4. BCD_to_sevenseg:

   bnum : in STD_LOGIC_VECTOR (3 downto 0);
   sevenseg : out STD_LOGIC_VECTOR (7 downto 0);

   This module takes BCD input and matches it with seven segment cathode output.

5. Top:

   clk : in STD_LOGİC;

```
echo: in std_logic;
reset : in  STD_LOGIC;
 trig: out std_logic;
sevenseg : out std_logic_vector (7 downto 0);
anodes_o : out std_logic_vector (3 downto 0);
pwm : out std_logic;
side1_o : out std_logic_vector (7 downto 0);
side2_o : out std_logic_vector (7 downto 0);
area_o : out std_logic_vector (15 downto 0)
```

This module is responsible for calculating the area and displaying it on a seven-segment display. This module also takes reset input and starts the measurement process sequentially by the next event counter. I instantiated Servo, HCSR04, Binary_BCD and BCD_to_sevenseg modules in the top module. It takes the distance output of HCSR04 and uses it for measurements. Servo is used for rotating ultrasonic sensors after the first measurement. For the seven segment display I wrote an anode process. I swapped 4 displays in high frequency and reached the persistence of vision that we learned in previous labs. I also used the Binary_BCD module to convert the area signal to BCD. After that I instantiated my Bcd_to_sevenseg module 4 times for each display. With seven segment cathode outputs I wrote my cathode process to show each digit.

6. Sync:

```
clk100: IN STD_LOGIC;
echo: in std_logic;
trig: out std_logic;
HSYNC: OUT STD_LOGIC;
VSYNC: OUT STD_LOGIC;
R: OUT STD_LOGIC_VECTOR(3 downto 0);
G: OUT STD_LOGIC_VECTOR(3 downto 0);
B: OUT STD_LOGIC_VECTOR(3 downto 0);
reset : in  STD_LOGIC;
sevenseg : out std_logic_vector (7 downto 0);
anodes_o : out std_logic_vector (3 downto 0);
pwm : out std_logic
```

This module is used for drawing measurements on a VGA monitor. I instantiated my top module in it. I used 1024x1280 60Hz resolution. For this resolution I need a 108 Mhz clock. I used the Vivado IP generator of the Clocking wizard to do this. I generated Hsync and Vsync waveforms to use a VGA monitor. Between the front porch and back porch I Applied low and otherwise high for each signal. I counted SYNC signals according to timing values specified for my resolution. We decide which pixels to turn on by checking current hcount and vcount positions. If the current pixel is in our drawing interval, we define RGB values according to the color we want to show. Each pixel includes RGB colors. If we want white we give all three colors to one. Otherwise if we want to have black pixel we give 0 for all three colors. I wanted white background color and used black for drawing.

7. Text drawing modules from https://github.com/Derek-X-Wang/VGA-Text-Generator:

   I used three modules from this source:

1. commonPak.vhd
2. pixel_on_text.vhd

   clk: in std_logic;
   displayText: in string (1 to textLength) := (others => NUL);
   position: in point_2d := (0, 0);
   horzCoord: in integer;
   vertCoord: in integer;
   pixel: out std_logic := '0'

3. Font_Rom.vhd

I used these modules to draw text on vga. I instantiated the pixel_on_text module in the sync module. It takes string input of measurements. It takes pixel coordinates as position input for where to draw text. It also takes horizontal and vertical current coordinates to give output. The output pixel indicates where we have a pixel included in text to show.

Results:

As a result I have a working device for calculating the area of a rectangular room automatically. After setting up the project environment and making cable connections I uploaded my bitstream file to fpga. When I pressed the reset button it resetted and started to area measurement. First it measured side1 and waited for a while. After that it rotated 90 degrees. When it turned to the other side, it took another measurement for side2. Then it calculated the area and displayed it on a seven segment display. It also showed both side and area measurements on a vga monitor around a rectangle. The results are shown below:

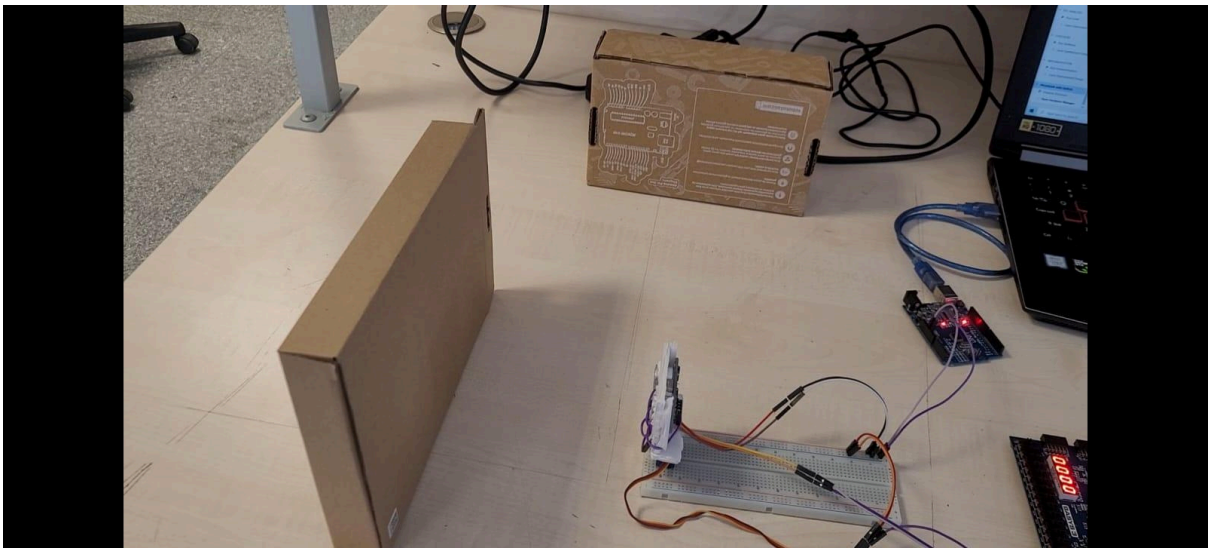

Figure 1.1 Measuring side 1.

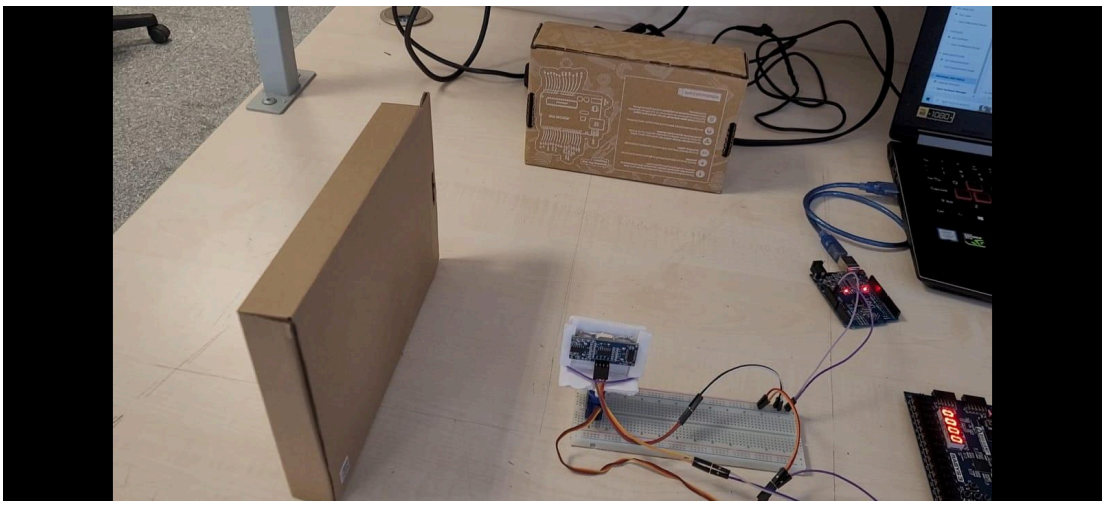

Figure 1.2 Measuring side 2.

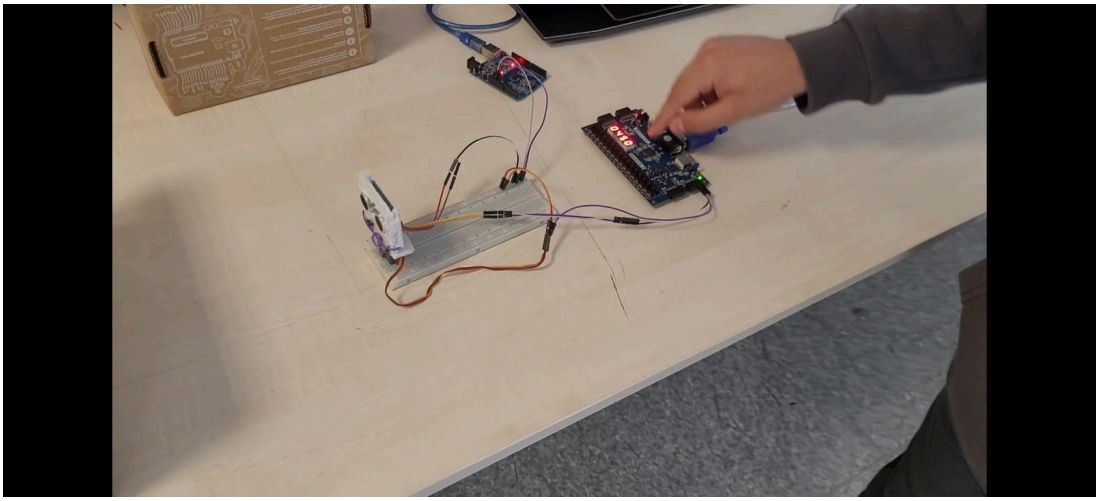

Figure 1.3 Area on seven segment display.

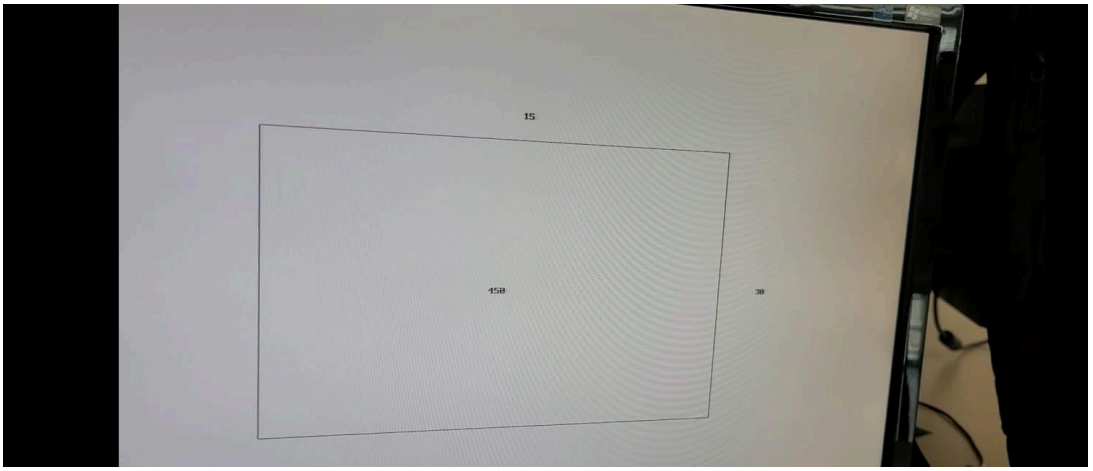

Figure 1.4 Measurements on a VGA monitor.

Conclusion:

In conclusion I designed and implemented a working device that can calculate the area of a rectangular room after making necessary measurements automatically. I used an ultrasonic sensor, servo motor, seven segment display and a vga monitor as external components. I used basys3 FPGA board and VHDL for all the processes. In my opinion it is a good idea for a product. It is simple to use. It also has a low cost due to cheap sensors. With necessary changes it can be used in industry. In my proposal I wrote that I will show the area just on a seven segment display. After my project worked well until this point I thought adding a VGA monitor could be a good idea. Then I started to learn VGA and combined it with my project. I preferred to use written codes for text drawing because it requires lots of data on ROM.

Sources:

http://www.tinyvga.com/vga-timing
https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
https://github.com/Derek-X-Wang/VGA-Text-Generator
http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
https://allaboutfpga.com/vhdl-code-for-binary-to-bcd-converter/
https://youtu.be/WK5FT5RD1sU?si=22Wy8HZtse1AfOAe

Youtube Link:
https://www.youtube.com/watch?v=4fVQo5mlX5c

VHDL Code:

Servo:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity servo is
    Port (
          clk : in STD_LOGIC;
          sw : in STD_LOGIC;
          pwm : out STD_LOGIC);
end servo;

architecture Behavioral of servo is

constant limit : integer := 100000000/50;
signal timer : integer range 0 to limit;
signal angle :  integer range 50000 to 250000;
```

```
begin

process (clk) begin
if rising_edge(clk) then
    if timer = limit - 1 then
        timer <= 0;
    else
        timer <= timer + 1;
    end if;

    if angle > timer then
        pwm <= '1';
    else
        pwm <= '0';
    end if;
end if;
end process;

process(sw) begin
if sw = '0' then
    angle <= 52000;
elsif sw = '1' then
    angle <= 145000;

end if;
end process;



end Behavioral;

----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------

HCSR04:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;



entity hcsr04 is
            port(
            clk: in std_logic;
    echo: in std_logic;
```

```
        trig: out std_logic;
    distance: out std_logic_vector(7 downto 0)
        );
end hcsr04;

architecture Behavioral of hcsr04 is

constant limit : integer := 6000000;
signal timer : integer range 0 to limit;
signal counter : integer range 0 to 100000000;
signal echo_p : std_logic := '0';
signal echo_fe : std_logic;



begin

process (clk) begin
if rising_edge(clk) then

    if timer = limit - 1 then
        timer <= 0;
    else
        timer <= timer + 1;
    end if;

    if timer < 1000 then
        counter <= 0;
        trig <= '1';
    else
        trig <= '0';
    end if;

    if echo = '1' then
        counter <= counter + 1;
    end if;

    if echo_fe = '1' then
        distance <= std_logic_vector(TO_UNSIGNED(integer(counter/5800), distance'length));
    end if;

end if;
end process;

echo_p <= echo when rising_edge(clk);
echo_fe <= not echo and echo_p;
```

```
end Behavioral;

----------------------------------------------------------
------------------------------------------------------------
-----------------------------------------------------

Binary_BCD:

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

entity binary_bcd is
    port(
        clk, reset: in std_logic;
        binary_in: in std_logic_vector(15 downto 0);
        bcd0, bcd1, bcd2, bcd3, bcd4: out std_logic_vector(3 downto 0)
    );
end binary_bcd ;

architecture behaviour of binary_bcd is
    type states is (start, shift, done);
    signal state, state_next: states;

    signal binary, binary_next: std_logic_vector(15 downto 0);
    signal s_bcd, s_bcd_reg, s_bcd_next: std_logic_vector(19 downto 0);
    signal s_bcd_out_reg, s_bcd_out_reg_next: std_logic_vector(19 downto 0);
    signal shift_counter, shift_counter_next: integer range 0 to 16;
begin

    process(clk, reset)
    begin
        if reset = '1' then
            binary <= (others => '0');
            s_bcd <= (others => '0');
            state <= start;
            s_bcd_out_reg <= (others => '0');
            shift_counter <= 0;
        elsif falling_edge(clk) then
            binary <= binary_next;
            s_bcd <= s_bcd_next;
            state <= state_next;
            s_bcd_out_reg <= s_bcd_out_reg_next;
            shift_counter <= shift_counter_next;
        end if;
    end process;
```

```
   convert:
   process(state, binary, binary_in, s_bcd, s_bcd_reg, shift_counter)
   begin
      state_next <= state;
      s_bcd_next <= s_bcd;
      binary_next <= binary;
      shift_counter_next <= shift_counter;

      case state is
         when start =>
            state_next <= shift;
            binary_next <= binary_in;
            s_bcd_next <= (others => '0');
            shift_counter_next <= 0;
         when shift =>
            if shift_counter = 16 then
               state_next <= done;
            else
               binary_next <= binary(14 downto 0) & 'L';
               s_bcd_next <= s_bcd_reg(18 downto 0) & binary(15);
               shift_counter_next <= shift_counter + 1;
            end if;
         when done =>
            state_next <= start;
      end case;
   end process;

   s_bcd_reg(19 downto 16) <= s_bcd(19 downto 16) + 3 when s_bcd(19 downto 16) > 4
else
                           s_bcd(19 downto 16);
   s_bcd_reg(15 downto 12) <= s_bcd(15 downto 12) + 3 when s_bcd(15 downto 12) > 4
else
                           s_bcd(15 downto 12);
   s_bcd_reg(11 downto 8) <= s_bcd(11 downto 8) + 3 when s_bcd(11 downto 8) > 4 else
                           s_bcd(11 downto 8);
   s_bcd_reg(7 downto 4) <= s_bcd(7 downto 4) + 3 when s_bcd(7 downto 4) > 4 else
                           s_bcd(7 downto 4);
   s_bcd_reg(3 downto 0) <= s_bcd(3 downto 0) + 3 when s_bcd(3 downto 0) > 4 else
                           s_bcd(3 downto 0);

   s_bcd_out_reg_next <= s_bcd when state = done else
                         s_bcd_out_reg;

   bcd4 <= s_bcd_out_reg(19 downto 16);
   bcd3 <= s_bcd_out_reg(15 downto 12);
   bcd2 <= s_bcd_out_reg(11 downto 8);
   bcd1 <= s_bcd_out_reg(7 downto 4);
   bcd0 <= s_bcd_out_reg(3 downto 0);
```

```
end behaviour;

----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------
BCD_to_sevenseg:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity bcd_to_sevenseg is
    Port ( bnum : in STD_LOGIC_VECTOR (3 downto 0);
           sevenseg : out STD_LOGIC_VECTOR (7 downto 0));
end bcd_to_sevenseg;

architecture Behavioral of bcd_to_sevenseg is

begin

process (bnum) begin

case bnum is
    when "0000" => sevenseg <= "11000000";
    when "0001" => sevenseg <= "11111001";
    when "0010" => sevenseg <= "10100100";
    when "0011" => sevenseg <= "10110000";
    when "0100" => sevenseg <= "10011001";
    when "0101" => sevenseg <= "10010010";
    when "0110" => sevenseg <= "10000010";
    when "0111" => sevenseg <= "11111000";
    when "1000" => sevenseg <= "10000000";
    when "1001" => sevenseg <= "10010000";
    when others => sevenseg <= "11111111";

end case;
end process;



end Behavioral;


----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------
```

Top:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;


entity top is
generic(clk_freq : integer := 100_000_000);

    Port (
          clk : in STD_LOGIC;
          echo: in std_logic;
          reset : in  STD_LOGIC;
                trig: out std_logic;
          sevenseg : out std_logic_vector (7 downto 0);
          anodes_o : out std_logic_vector (3 downto 0);
          pwm : out std_logic;
          side1_o : out std_logic_vector (7 downto 0);
          side2_o : out std_logic_vector (7 downto 0);
          area_o : out std_logic_vector (15 downto 0)
          );
end top;

architecture Behavioral of top is

component servo is
    Port (
          clk : in STD_LOGIC;
          sw : in STD_LOGIC;
          pwm : out STD_LOGIC);
end component;


component bcd_to_sevenseg is
    Port ( bnum : in STD_LOGIC_VECTOR (3 downto 0);
          sevenseg : out STD_LOGIC_VECTOR (7 downto 0));
end component;


component binary_bcd is
    generic(N: positive := 16);
    port(
        clk, reset: in std_logic;
        binary_in: in std_logic_vector(N-1 downto 0);
        bcd0, bcd1, bcd2, bcd3, bcd4: out std_logic_vector(3 downto 0)
```

```
    );
end component ;

component hcsr04 is
          port(
          clk: in std_logic;
    echo: in std_logic;
          trig: out std_logic;
    distance: out std_logic_vector(7 downto 0)
          );
end component;

constant clk_lim : integer := clk_freq/1000;
constant event_timer_lim : std_logic_vector := "11111111111111111111111111";
signal timer : integer range 0 to clk_lim;
signal anodes : std_logic_vector(3 downto 0) := "1110";
signal first : std_logic_vector(3 downto 0);
signal second : std_logic_vector(3 downto 0);
signal third : std_logic_vector(3 downto 0);
signal fourth : std_logic_vector(3 downto 0);
signal fsevenseg : std_logic_vector(7 downto 0);
signal ssevenseg : std_logic_vector(7 downto 0);
signal tsevenseg : std_logic_vector(7 downto 0);
signal fosevenseg : std_logic_vector(7 downto 0);
signal distance_s : std_logic_vector(7 downto 0);
signal distance_a : std_logic_vector(7 downto 0) := "00000000";
signal distance_b : std_logic_vector(7 downto 0) := "00000000";
signal area : std_logic_vector(15 downto 0) := "0000000000000000";
signal event_timer : std_logic_vector(26 downto 0) := (others => '0');
signal next_event : integer range 0 to 8;
signal turn : std_logic := '0';

begin

angle: servo
port map(clk => clk,
        sw => turn,
        pwm => pwm);

dist: hcsr04
port map (clk => clk,
         echo => echo,
         trig => trig,
         distance => distance_s);

bin: binary_bcd
generic map(N => 16)
port map(clk => clk,
```

```
        reset => reset,
        binary_in => area,
        bcd0 => fourth,
        bcd1 => third,
        bcd2 => second,
        bcd3 => first);

fd: bcd_to_sevenseg
port map (
bnum => first,
sevenseg => fsevenseg
);
sd: bcd_to_sevenseg
port map (
bnum => second,
sevenseg => ssevenseg
);
td: bcd_to_sevenseg
port map (
bnum => third,
sevenseg => tsevenseg
);
fod: bcd_to_sevenseg
port map (
bnum => fourth,
sevenseg => fosevenseg
);



main_process: process(clk) begin
if (rising_edge(clk)) then

    if next_event < 6 then
        if (event_timer = event_timer_lim-1) then
            event_timer <= (others => '0');
            next_event <= next_event + 1;
        else
            event_timer <= event_timer+1;
    end if;
    end if;



    if (timer = clk_lim-1) then
        timer <= 0;
        anodes(3 downto 1) <= anodes(2 downto 0);
        anodes(0) <= anodes(3);
```

```
    else
        timer <= timer+1;
    end if;

    if (anodes(0) = '0') then
        sevenseg <= fosevenseg;
    elsif (anodes(1) = '0') then
        sevenseg <= tsevenseg;
    elsif (anodes(2) = '0') then
        sevenseg <= ssevenseg;
    elsif (anodes(3) = '0') then
        sevenseg <= fsevenseg;
    end if;

    if reset = '1' then
        distance_a <= (others => '0');
        distance_b <= (others => '0');
        next_event <= 0;
        turn <= '0';
    end if;

    if next_event = 2 then
        distance_a <= distance_s;
    elsif next_event = 3 then
        turn <= '1';
    elsif next_event = 5 then
    distance_b <= distance_s;
    end if;

end if;

end process;


area_o <= area;
side2_o <= distance_b;
side1_o <= distance_a;
area <= distance_a * distance_b;
anodes_o <= anodes;



end Behavioral;

----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------
```

SYNC:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.commonPak.all;

ENTITY SYNC IS
PORT(
CLK100: IN STD_LOGIC;
echo: in std_logic;
trig: out std_logic;
HSYNC: OUT STD_LOGIC;
VSYNC: OUT STD_LOGIC;
R: OUT STD_LOGIC_VECTOR(3 downto 0);
G: OUT STD_LOGIC_VECTOR(3 downto 0);
B: OUT STD_LOGIC_VECTOR(3 downto 0);
reset : in  STD_LOGIC;
sevenseg : out std_logic_vector (7 downto 0);
anodes_o : out std_logic_vector (3 downto 0);
pwm : out std_logic
);
END SYNC;

ARCHITECTURE MAIN OF SYNC IS

component Pixel_On_Text is
        generic(
    textLength: integer := 3
        );
        port (
                clk: in std_logic;
                displayText: in string (1 to textLength) := (others => NUL);
                -- top left corner of the text
                position: in point_2d := (0, 0);
                -- current pixel postion
                horzCoord: in integer;
                vertCoord: in integer;

                pixel: out std_logic := '0'
        );

end component;


component clk_wiz_0
port
```

```
 (-- Clock in ports
  -- Clock out ports
  clk_108          : out    std_logic;
  clk_100          : in     std_logic
 );
end component;

component top is
generic(clk_freq : integer := 100_000_000);

    Port (
          clk : in STD_LOGIC;
          echo: in std_logic;
          reset : in  STD_LOGIC;
                trig: out std_logic;
          sevenseg : out std_logic_vector (7 downto 0);
          anodes_o : out std_logic_vector (3 downto 0);
          pwm : out std_logic;
          side1_o : out std_logic_vector (7 downto 0);
          side2_o : out std_logic_vector (7 downto 0);
          area_o : out std_logic_vector (15 downto 0)
          );
end component;

signal clk108: std_logic;
signal t1: std_logic;
signal t2: std_logic;
signal t3: std_logic;
signal side1 : std_logic_vector(7 downto 0);
signal side2 : std_logic_vector(7 downto 0);
signal area : std_logic_vector(15 downto 0);

signal dist1 : integer range 0 to 256;
signal dist2 : integer range 0 to 256;
signal areaint : integer range 0 to 32768;

signal str1: string (1 to 3);
signal str2: string (1 to 3);
signal str3: string (1 to 3);




SIGNAL HPOS: INTEGER RANGE 0 TO 1688:=0;
SIGNAL VPOS: INTEGER RANGE 0 TO 1066:=0;
```

```
BEGIN

    topmod: top
    port map(clk => clk100,
          echo => echo,
          reset =>reset,
                trig =>trig,
          sevenseg =>sevenseg,
          anodes_o =>anodes_o,
          pwm => pwm,
          side1_o => side1,
          side2_o => side2,
          area_o => area);



    textElement1: entity work.Pixel_On_Text
    generic map (
       textLength => 3
    )
    port map(
       clk => clk108,
       displayText => str1,
       position => (1420, 554), -- text position (top left)
       horzCoord => HPOS,
       vertCoord => VPOS,
       pixel => t1 -- result
    );

    textElement2: entity work.Pixel_On_Text
    generic map (
       textLength => 3
    )
    port map(
       clk => clk108,
       displayText => str2,
       position => (1048, 250), -- text position (top left)
       horzCoord => HPOS,
       vertCoord => VPOS,
       pixel => t2 -- result
       );

    textElement3: entity work.Pixel_On_Text
    generic map (
       textLength => 3
    )
    port map(
       clk => clk108,
```

```
      displayText => str3,
      position => (1018, 554), -- text position (top left)
      horzCoord => HPOS,
      vertCoord => VPOS,
      pixel => t3 -- result
    );









  clocking_wizard : clk_wiz_0
  port map (
 -- Clock out ports
  clk_108 => clk108,
  -- Clock in ports
  clk_100 => clk100

 );
 PROCESS(clk108)
 BEGIN
IF(clk108'EVENT AND clk108='1')THEN


    IF(HPOS<1688)THEN
  HPOS<=HPOS+1;
  ELSE
  HPOS<=0;
     IF(VPOS<1066)THEN
       VPOS<=VPOS+1;
     ELSE
       VPOS<=0;
     END IF;
  END IF;
  IF((HPOS>0 AND HPOS<408) OR (VPOS>0 AND VPOS<42))THEN
  R<=(others=>'0');
  G<=(others=>'0');
  B<=(others=>'0');
  END IF;

  IF(HPOS>48 AND HPOS<160)THEN----HSYNC
    HSYNC<='0';
  ELSE
    HSYNC<='1';
```

```
    END IF;

    IF(VPOS>0 AND VPOS<4)THEN----------vsync
      VSYNC<='0';
    ELSE
      VSYNC<='1';
    END IF;


     if (t1 = '1' or t2 = '1' or t3 = '1') then
    R <= "0000";
    G <= "0000";
    B <= "0000";

    elsif HPOS < 1368 and HPOS > 728 and (VPOS = 298 or VPOS = 810) then
    R <= "0000";
    G <= "0000";
    B <= "0000";

    elsif VPOS > 298 and VPOS < 810 and (HPOS = 728 or HPOS = 1368) then
    R <= "0000";
    G <= "0000";
    B <= "0000";

    else
    R <= "1111";
    G <= "1111";
    B <= "1111";
    end if;


    dist1 <= TO_INTEGER(UNSIGNED(side1));
    dist2 <= TO_INTEGER(UNSIGNED(side2));
    areaint <= TO_INTEGER(UNSIGNED(area));

    case dist1 is
when  0 => str1 <=  "  0";
when  1 => str1 <=  "  1";
when  2 => str1 <=  "  2";
when  3 => str1 <=  "  3";
when  4 => str1 <=  "  4";
when  5 => str1 <=  "  5";
when  6 => str1 <=  "  6";
when  7 => str1 <=  "  7";
when  8 => str1 <=  "  8";
when  9 => str1 <=  "  9";
when  10 => str1 <=  " 10";
when  11 => str1 <=  " 11";
```

```
when  12 => str1 <=  " 12";
when  13 => str1 <=  " 13";
when  14 => str1 <=  " 14";
when  15 => str1 <=  " 15";
when  16 => str1 <=  " 16";
when  17 => str1 <=  " 17";
when  18 => str1 <=  " 18";
when  19 => str1 <=  " 19";
when  20 => str1 <=  " 20";
when  21 => str1 <=  " 21";
when  22 => str1 <=  " 22";
when  23 => str1 <=  " 23";
when  24 => str1 <=  " 24";
when  25 => str1 <=  " 25";
when  26 => str1 <=  " 26";
when  27 => str1 <=  " 27";
when  28 => str1 <=  " 28";
when  29 => str1 <=  " 29";
when  30 => str1 <=  " 30";
when  31 => str1 <=  " 31";
when  32 => str1 <=  " 32";
when  33 => str1 <=  " 33";
when  34 => str1 <=  " 34";
when  35 => str1 <=  " 35";
when  36 => str1 <=  " 36";
when  37 => str1 <=  " 37";
when  38 => str1 <=  " 38";
when  39 => str1 <=  " 39";
when  40 => str1 <=  " 40";
when  41 => str1 <=  " 41";
when  42 => str1 <=  " 42";
when  43 => str1 <=  " 43";
when  44 => str1 <=  " 44";
when  45 => str1 <=  " 45";
when  46 => str1 <=  " 46";
when  47 => str1 <=  " 47";
when  48 => str1 <=  " 48";
when  49 => str1 <=  " 49";
when  50 => str1 <=  " 50";
when  51 => str1 <=  " 51";
when  52 => str1 <=  " 52";
when  53 => str1 <=  " 53";
when  54 => str1 <=  " 54";
when  55 => str1 <=  " 55";
when  56 => str1 <=  " 56";
when  57 => str1 <=  " 57";
when  58 => str1 <=  " 58";
when  59 => str1 <=  " 59";
```

```
when  60 => str1 <=  " 60";
when  61 => str1 <=  " 61";
when  62 => str1 <=  " 62";
when  63 => str1 <=  " 63";
when  64 => str1 <=  " 64";
when  65 => str1 <=  " 65";
when  66 => str1 <=  " 66";
when  67 => str1 <=  " 67";
when  68 => str1 <=  " 68";
when  69 => str1 <=  " 69";
when  70 => str1 <=  " 70";
when  71 => str1 <=  " 71";
when  72 => str1 <=  " 72";
when  73 => str1 <=  " 73";
when  74 => str1 <=  " 74";
when  75 => str1 <=  " 75";
when  76 => str1 <=  " 76";
when  77 => str1 <=  " 77";
when  78 => str1 <=  " 78";
when  79 => str1 <=  " 79";
when  80 => str1 <=  " 80";
when  81 => str1 <=  " 81";
when  82 => str1 <=  " 82";
when  83 => str1 <=  " 83";
when  84 => str1 <=  " 84";
when  85 => str1 <=  " 85";
when  86 => str1 <=  " 86";
when  87 => str1 <=  " 87";
when  88 => str1 <=  " 88";
when  89 => str1 <=  " 89";
when  90 => str1 <=  " 90";
when  91 => str1 <=  " 91";
when  92 => str1 <=  " 92";
when  93 => str1 <=  " 93";
when  94 => str1 <=  " 94";
when  95 => str1 <=  " 95";
when  96 => str1 <=  " 96";
when  97 => str1 <=  " 97";
when  98 => str1 <=  " 98";
when  99 => str1 <=  " 99";
when  100 => str1 <=  "100";
when  101 => str1 <=  "101";
when  102 => str1 <=  "102";
when  103 => str1 <=  "103";
when  104 => str1 <=  "104";
when  105 => str1 <=  "105";
when  106 => str1 <=  "106";
when  107 => str1 <=  "107";
```

```
when  108 => str1 <=  "108";
when  109 => str1 <=  "109";
when  110 => str1 <=  "110";
when  111 => str1 <=  "111";
when  112 => str1 <=  "112";
when  113 => str1 <=  "113";
when  114 => str1 <=  "114";
when  115 => str1 <=  "115";
when  116 => str1 <=  "116";
when  117 => str1 <=  "117";
when  118 => str1 <=  "118";
when  119 => str1 <=  "119";
when  120 => str1 <=  "120";
when  121 => str1 <=  "121";
when  122 => str1 <=  "122";
when  123 => str1 <=  "123";
when  124 => str1 <=  "124";
when  125 => str1 <=  "125";
when  126 => str1 <=  "126";
when  127 => str1 <=  "127";
when  128 => str1 <=  "128";
when  129 => str1 <=  "129";
when  130 => str1 <=  "130";
when  131 => str1 <=  "131";
when  132 => str1 <=  "132";
when  133 => str1 <=  "133";
when  134 => str1 <=  "134";
when  135 => str1 <=  "135";
when  136 => str1 <=  "136";
when  137 => str1 <=  "137";
when  138 => str1 <=  "138";
when  139 => str1 <=  "139";
when  140 => str1 <=  "140";
when  141 => str1 <=  "141";
when  142 => str1 <=  "142";
when  143 => str1 <=  "143";
when  144 => str1 <=  "144";
when  145 => str1 <=  "145";
when  146 => str1 <=  "146";
when  147 => str1 <=  "147";
when  148 => str1 <=  "148";
when  149 => str1 <=  "149";
when  150 => str1 <=  "150";
when  151 => str1 <=  "151";
when  152 => str1 <=  "152";
when  153 => str1 <=  "153";
when  154 => str1 <=  "154";
when  155 => str1 <=  "155";
```

```
when  156 => str1 <=  "156";
when  157 => str1 <=  "157";
when  158 => str1 <=  "158";
when  159 => str1 <=  "159";
when  160 => str1 <=  "160";
when  161 => str1 <=  "161";
when  162 => str1 <=  "162";
when  163 => str1 <=  "163";
when  164 => str1 <=  "164";
when  165 => str1 <=  "165";
when  166 => str1 <=  "166";
when  167 => str1 <=  "167";
when  168 => str1 <=  "168";
when  169 => str1 <=  "169";
when  170 => str1 <=  "170";
when  171 => str1 <=  "171";
when  172 => str1 <=  "172";
when  173 => str1 <=  "173";
when  174 => str1 <=  "174";
when  175 => str1 <=  "175";
when  176 => str1 <=  "176";
when  177 => str1 <=  "177";
when  178 => str1 <=  "178";
when  179 => str1 <=  "179";
when  180 => str1 <=  "180";
when  181 => str1 <=  "181";
when  182 => str1 <=  "182";
when  183 => str1 <=  "183";
when  184 => str1 <=  "184";
when  185 => str1 <=  "185";
when  186 => str1 <=  "186";
when  187 => str1 <=  "187";
when  188 => str1 <=  "188";
when  189 => str1 <=  "189";
when  190 => str1 <=  "190";
when  191 => str1 <=  "191";
when  192 => str1 <=  "192";
when  193 => str1 <=  "193";
when  194 => str1 <=  "194";
when  195 => str1 <=  "195";
when  196 => str1 <=  "196";
when  197 => str1 <=  "197";
when  198 => str1 <=  "198";
when  199 => str1 <=  "199";
when  200 => str1 <=  "200";
when  201 => str1 <=  "201";
when  202 => str1 <=  "202";
when  203 => str1 <=  "203";
```

```
when  204 => str1 <=  "204";
when  205 => str1 <=  "205";
when  206 => str1 <=  "206";
when  207 => str1 <=  "207";
when  208 => str1 <=  "208";
when  209 => str1 <=  "209";
when  210 => str1 <=  "210";
when  211 => str1 <=  "211";
when  212 => str1 <=  "212";
when  213 => str1 <=  "213";
when  214 => str1 <=  "214";
when  215 => str1 <=  "215";
when  216 => str1 <=  "216";
when  217 => str1 <=  "217";
when  218 => str1 <=  "218";
when  219 => str1 <=  "219";
when  220 => str1 <=  "220";
when  221 => str1 <=  "221";
when  222 => str1 <=  "222";
when  223 => str1 <=  "223";
when  224 => str1 <=  "224";
when  225 => str1 <=  "225";
when  226 => str1 <=  "226";
when  227 => str1 <=  "227";
when  228 => str1 <=  "228";
when  229 => str1 <=  "229";
when  230 => str1 <=  "230";
when  231 => str1 <=  "231";
when  232 => str1 <=  "232";
when  233 => str1 <=  "233";
when  234 => str1 <=  "234";
when  235 => str1 <=  "235";
when  236 => str1 <=  "236";
when  237 => str1 <=  "237";
when  238 => str1 <=  "238";
when  239 => str1 <=  "239";
when  240 => str1 <=  "240";
when  241 => str1 <=  "241";
when  242 => str1 <=  "242";
when  243 => str1 <=  "243";
when  244 => str1 <=  "244";
when  245 => str1 <=  "245";
when  246 => str1 <=  "246";
when  247 => str1 <=  "247";
when  248 => str1 <=  "248";
when  249 => str1 <=  "249";
when  250 => str1 <=  "250";
when  251 => str1 <=  "251";
```

```
when  252 => str1 <=  "252";
when  253 => str1 <=  "253";
when  254 => str1 <=  "254";
when  255 => str1 <=  "255";
when  256 => str1 <=  "256";
when others => str1 <= "   ";
   end case;

   case dist2 is
when  0 => str2 <=  "  0";
when  1 => str2 <=  "  1";
when  2 => str2 <=  "  2";
when  3 => str2 <=  "  3";
when  4 => str2 <=  "  4";
when  5 => str2 <=  "  5";
when  6 => str2 <=  "  6";
when  7 => str2 <=  "  7";
when  8 => str2 <=  "  8";
when  9 => str2 <=  "  9";
when  10 => str2 <=  " 10";
when  11 => str2 <=  " 11";
when  12 => str2 <=  " 12";
when  13 => str2 <=  " 13";
when  14 => str2 <=  " 14";
when  15 => str2 <=  " 15";
when  16 => str2 <=  " 16";
when  17 => str2 <=  " 17";
when  18 => str2 <=  " 18";
when  19 => str2 <=  " 19";
when  20 => str2 <=  " 20";
when  21 => str2 <=  " 21";
when  22 => str2 <=  " 22";
when  23 => str2 <=  " 23";
when  24 => str2 <=  " 24";
when  25 => str2 <=  " 25";
when  26 => str2 <=  " 26";
when  27 => str2 <=  " 27";
when  28 => str2 <=  " 28";
when  29 => str2 <=  " 29";
when  30 => str2 <=  " 30";
when  31 => str2 <=  " 31";
when  32 => str2 <=  " 32";
when  33 => str2 <=  " 33";
when  34 => str2 <=  " 34";
when  35 => str2 <=  " 35";
when  36 => str2 <=  " 36";
when  37 => str2 <=  " 37";
when  38 => str2 <=  " 38";
```

```
when  39 => str2 <=  " 39";
when  40 => str2 <=  " 40";
when  41 => str2 <=  " 41";
when  42 => str2 <=  " 42";
when  43 => str2 <=  " 43";
when  44 => str2 <=  " 44";
when  45 => str2 <=  " 45";
when  46 => str2 <=  " 46";
when  47 => str2 <=  " 47";
when  48 => str2 <=  " 48";
when  49 => str2 <=  " 49";
when  50 => str2 <=  " 50";
when  51 => str2 <=  " 51";
when  52 => str2 <=  " 52";
when  53 => str2 <=  " 53";
when  54 => str2 <=  " 54";
when  55 => str2 <=  " 55";
when  56 => str2 <=  " 56";
when  57 => str2 <=  " 57";
when  58 => str2 <=  " 58";
when  59 => str2 <=  " 59";
when  60 => str2 <=  " 60";
when  61 => str2 <=  " 61";
when  62 => str2 <=  " 62";
when  63 => str2 <=  " 63";
when  64 => str2 <=  " 64";
when  65 => str2 <=  " 65";
when  66 => str2 <=  " 66";
when  67 => str2 <=  " 67";
when  68 => str2 <=  " 68";
when  69 => str2 <=  " 69";
when  70 => str2 <=  " 70";
when  71 => str2 <=  " 71";
when  72 => str2 <=  " 72";
when  73 => str2 <=  " 73";
when  74 => str2 <=  " 74";
when  75 => str2 <=  " 75";
when  76 => str2 <=  " 76";
when  77 => str2 <=  " 77";
when  78 => str2 <=  " 78";
when  79 => str2 <=  " 79";
when  80 => str2 <=  " 80";
when  81 => str2 <=  " 81";
when  82 => str2 <=  " 82";
when  83 => str2 <=  " 83";
when  84 => str2 <=  " 84";
when  85 => str2 <=  " 85";
when  86 => str2 <=  " 86";
```

```
when  87 => str2 <=  " 87";
when  88 => str2 <=  " 88";
when  89 => str2 <=  " 89";
when  90 => str2 <=  " 90";
when  91 => str2 <=  " 91";
when  92 => str2 <=  " 92";
when  93 => str2 <=  " 93";
when  94 => str2 <=  " 94";
when  95 => str2 <=  " 95";
when  96 => str2 <=  " 96";
when  97 => str2 <=  " 97";
when  98 => str2 <=  " 98";
when  99 => str2 <=  " 99";
when  100 => str2 <=  "100";
when  101 => str2 <=  "101";
when  102 => str2 <=  "102";
when  103 => str2 <=  "103";
when  104 => str2 <=  "104";
when  105 => str2 <=  "105";
when  106 => str2 <=  "106";
when  107 => str2 <=  "107";
when  108 => str2 <=  "108";
when  109 => str2 <=  "109";
when  110 => str2 <=  "110";
when  111 => str2 <=  "111";
when  112 => str2 <=  "112";
when  113 => str2 <=  "113";
when  114 => str2 <=  "114";
when  115 => str2 <=  "115";
when  116 => str2 <=  "116";
when  117 => str2 <=  "117";
when  118 => str2 <=  "118";
when  119 => str2 <=  "119";
when  120 => str2 <=  "120";
when  121 => str2 <=  "121";
when  122 => str2 <=  "122";
when  123 => str2 <=  "123";
when  124 => str2 <=  "124";
when  125 => str2 <=  "125";
when  126 => str2 <=  "126";
when  127 => str2 <=  "127";
when  128 => str2 <=  "128";
when  129 => str2 <=  "129";
when  130 => str2 <=  "130";
when  131 => str2 <=  "131";
when  132 => str2 <=  "132";
when  133 => str2 <=  "133";
when  134 => str2 <=  "134";
```

```
when  135 => str2 <=  "135";
when  136 => str2 <=  "136";
when  137 => str2 <=  "137";
when  138 => str2 <=  "138";
when  139 => str2 <=  "139";
when  140 => str2 <=  "140";
when  141 => str2 <=  "141";
when  142 => str2 <=  "142";
when  143 => str2 <=  "143";
when  144 => str2 <=  "144";
when  145 => str2 <=  "145";
when  146 => str2 <=  "146";
when  147 => str2 <=  "147";
when  148 => str2 <=  "148";
when  149 => str2 <=  "149";
when  150 => str2 <=  "150";
when  151 => str2 <=  "151";
when  152 => str2 <=  "152";
when  153 => str2 <=  "153";
when  154 => str2 <=  "154";
when  155 => str2 <=  "155";
when  156 => str2 <=  "156";
when  157 => str2 <=  "157";
when  158 => str2 <=  "158";
when  159 => str2 <=  "159";
when  160 => str2 <=  "160";
when  161 => str2 <=  "161";
when  162 => str2 <=  "162";
when  163 => str2 <=  "163";
when  164 => str2 <=  "164";
when  165 => str2 <=  "165";
when  166 => str2 <=  "166";
when  167 => str2 <=  "167";
when  168 => str2 <=  "168";
when  169 => str2 <=  "169";
when  170 => str2 <=  "170";
when  171 => str2 <=  "171";
when  172 => str2 <=  "172";
when  173 => str2 <=  "173";
when  174 => str2 <=  "174";
when  175 => str2 <=  "175";
when  176 => str2 <=  "176";
when  177 => str2 <=  "177";
when  178 => str2 <=  "178";
when  179 => str2 <=  "179";
when  180 => str2 <=  "180";
when  181 => str2 <=  "181";
when  182 => str2 <=  "182";
```

```
when  183 => str2 <=  "183";
when  184 => str2 <=  "184";
when  185 => str2 <=  "185";
when  186 => str2 <=  "186";
when  187 => str2 <=  "187";
when  188 => str2 <=  "188";
when  189 => str2 <=  "189";
when  190 => str2 <=  "190";
when  191 => str2 <=  "191";
when  192 => str2 <=  "192";
when  193 => str2 <=  "193";
when  194 => str2 <=  "194";
when  195 => str2 <=  "195";
when  196 => str2 <=  "196";
when  197 => str2 <=  "197";
when  198 => str2 <=  "198";
when  199 => str2 <=  "199";
when  200 => str2 <=  "200";
when  201 => str2 <=  "201";
when  202 => str2 <=  "202";
when  203 => str2 <=  "203";
when  204 => str2 <=  "204";
when  205 => str2 <=  "205";
when  206 => str2 <=  "206";
when  207 => str2 <=  "207";
when  208 => str2 <=  "208";
when  209 => str2 <=  "209";
when  210 => str2 <=  "210";
when  211 => str2 <=  "211";
when  212 => str2 <=  "212";
when  213 => str2 <=  "213";
when  214 => str2 <=  "214";
when  215 => str2 <=  "215";
when  216 => str2 <=  "216";
when  217 => str2 <=  "217";
when  218 => str2 <=  "218";
when  219 => str2 <=  "219";
when  220 => str2 <=  "220";
when  221 => str2 <=  "221";
when  222 => str2 <=  "222";
when  223 => str2 <=  "223";
when  224 => str2 <=  "224";
when  225 => str2 <=  "225";
when  226 => str2 <=  "226";
when  227 => str2 <=  "227";
when  228 => str2 <=  "228";
when  229 => str2 <=  "229";
when  230 => str2 <=  "230";
```

```
when  231 => str2 <=  "231";
when  232 => str2 <=  "232";
when  233 => str2 <=  "233";
when  234 => str2 <=  "234";
when  235 => str2 <=  "235";
when  236 => str2 <=  "236";
when  237 => str2 <=  "237";
when  238 => str2 <=  "238";
when  239 => str2 <=  "239";
when  240 => str2 <=  "240";
when  241 => str2 <=  "241";
when  242 => str2 <=  "242";
when  243 => str2 <=  "243";
when  244 => str2 <=  "244";
when  245 => str2 <=  "245";
when  246 => str2 <=  "246";
when  247 => str2 <=  "247";
when  248 => str2 <=  "248";
when  249 => str2 <=  "249";
when  250 => str2 <=  "250";
when  251 => str2 <=  "251";
when  252 => str2 <=  "252";
when  253 => str2 <=  "253";
when  254 => str2 <=  "254";
when  255 => str2 <=  "255";
when  256 => str2 <=  "256";
when others => str2 <= "   ";
   end case;

   case areaint is
   when  0 => str3 <=  "  0";
   when  1 => str3 <=  "  1";
   when  2 => str3 <=  "  2";
   when  3 => str3 <=  "  3";
   when  4 => str3 <=  "  4";
   when  5 => str3 <=  "  5";
   when  6 => str3 <=  "  6";
   when  7 => str3 <=  "  7";
   when  8 => str3 <=  "  8";
   when  9 => str3 <=  "  9";
   when  10 => str3 <=  " 10";
   when  11 => str3 <=  " 11";
   when  12 => str3 <=  " 12";
   when  13 => str3 <=  " 13";
   when  14 => str3 <=  " 14";
   when  15 => str3 <=  " 15";
   when  16 => str3 <=  " 16";
   when  17 => str3 <=  " 17";
```

```
when  18 => str3 <=  " 18";
when  19 => str3 <=  " 19";
when  20 => str3 <=  " 20";
when  21 => str3 <=  " 21";
when  22 => str3 <=  " 22";
when  23 => str3 <=  " 23";
when  24 => str3 <=  " 24";
when  25 => str3 <=  " 25";
when  26 => str3 <=  " 26";
when  27 => str3 <=  " 27";
when  28 => str3 <=  " 28";
when  29 => str3 <=  " 29";
when  30 => str3 <=  " 30";
when  31 => str3 <=  " 31";
when  32 => str3 <=  " 32";
when  33 => str3 <=  " 33";
when  34 => str3 <=  " 34";
when  35 => str3 <=  " 35";
when  36 => str3 <=  " 36";
when  37 => str3 <=  " 37";
when  38 => str3 <=  " 38";
when  39 => str3 <=  " 39";
when  40 => str3 <=  " 40";
when  41 => str3 <=  " 41";
when  42 => str3 <=  " 42";
when  43 => str3 <=  " 43";
when  44 => str3 <=  " 44";
when  45 => str3 <=  " 45";
when  46 => str3 <=  " 46";
when  47 => str3 <=  " 47";
when  48 => str3 <=  " 48";
when  49 => str3 <=  " 49";
when  50 => str3 <=  " 50";
when  51 => str3 <=  " 51";
when  52 => str3 <=  " 52";
when  53 => str3 <=  " 53";
when  54 => str3 <=  " 54";
when  55 => str3 <=  " 55";
when  56 => str3 <=  " 56";
when  57 => str3 <=  " 57";
when  58 => str3 <=  " 58";
when  59 => str3 <=  " 59";
when  60 => str3 <=  " 60";
when  61 => str3 <=  " 61";
when  62 => str3 <=  " 62";
when  63 => str3 <=  " 63";
when  64 => str3 <=  " 64";
when  65 => str3 <=  " 65";
```

```
when  66 => str3 <=  " 66";
when  67 => str3 <=  " 67";
when  68 => str3 <=  " 68";
when  69 => str3 <=  " 69";
when  70 => str3 <=  " 70";
when  71 => str3 <=  " 71";
when  72 => str3 <=  " 72";
when  73 => str3 <=  " 73";
when  74 => str3 <=  " 74";
when  75 => str3 <=  " 75";
when  76 => str3 <=  " 76";
when  77 => str3 <=  " 77";
when  78 => str3 <=  " 78";
when  79 => str3 <=  " 79";
when  80 => str3 <=  " 80";
when  81 => str3 <=  " 81";
when  82 => str3 <=  " 82";
when  83 => str3 <=  " 83";
when  84 => str3 <=  " 84";
when  85 => str3 <=  " 85";
when  86 => str3 <=  " 86";
when  87 => str3 <=  " 87";
when  88 => str3 <=  " 88";
when  89 => str3 <=  " 89";
when  90 => str3 <=  " 90";
when  91 => str3 <=  " 91";
when  92 => str3 <=  " 92";
when  93 => str3 <=  " 93";
when  94 => str3 <=  " 94";
when  95 => str3 <=  " 95";
when  96 => str3 <=  " 96";
when  97 => str3 <=  " 97";
when  98 => str3 <=  " 98";
when  99 => str3 <=  " 99";
when  100 => str3 <=  "100";
when  101 => str3 <=  "101";
when  102 => str3 <=  "102";
when  103 => str3 <=  "103";
when  104 => str3 <=  "104";
when  105 => str3 <=  "105";
when  106 => str3 <=  "106";
when  107 => str3 <=  "107";
when  108 => str3 <=  "108";
when  109 => str3 <=  "109";
when  110 => str3 <=  "110";
when  111 => str3 <=  "111";
when  112 => str3 <=  "112";
when  113 => str3 <=  "113";
```

```
when  114 => str3 <=  "114";
when  115 => str3 <=  "115";
when  116 => str3 <=  "116";
when  117 => str3 <=  "117";
when  118 => str3 <=  "118";
when  119 => str3 <=  "119";
when  120 => str3 <=  "120";
when  121 => str3 <=  "121";
when  122 => str3 <=  "122";
when  123 => str3 <=  "123";
when  124 => str3 <=  "124";
when  125 => str3 <=  "125";
when  126 => str3 <=  "126";
when  127 => str3 <=  "127";
when  128 => str3 <=  "128";
when  129 => str3 <=  "129";
when  130 => str3 <=  "130";
when  131 => str3 <=  "131";
when  132 => str3 <=  "132";
when  133 => str3 <=  "133";
when  134 => str3 <=  "134";
when  135 => str3 <=  "135";
when  136 => str3 <=  "136";
when  137 => str3 <=  "137";
when  138 => str3 <=  "138";
when  139 => str3 <=  "139";
when  140 => str3 <=  "140";
when  141 => str3 <=  "141";
when  142 => str3 <=  "142";
when  143 => str3 <=  "143";
when  144 => str3 <=  "144";
when  145 => str3 <=  "145";
when  146 => str3 <=  "146";
when  147 => str3 <=  "147";
when  148 => str3 <=  "148";
when  149 => str3 <=  "149";
when  150 => str3 <=  "150";
when  151 => str3 <=  "151";
when  152 => str3 <=  "152";
when  153 => str3 <=  "153";
when  154 => str3 <=  "154";
when  155 => str3 <=  "155";
when  156 => str3 <=  "156";
when  157 => str3 <=  "157";
when  158 => str3 <=  "158";
when  159 => str3 <=  "159";
when  160 => str3 <=  "160";
when  161 => str3 <=  "161";
```

```
when  162 => str3 <=  "162";
when  163 => str3 <=  "163";
when  164 => str3 <=  "164";
when  165 => str3 <=  "165";
when  166 => str3 <=  "166";
when  167 => str3 <=  "167";
when  168 => str3 <=  "168";
when  169 => str3 <=  "169";
when  170 => str3 <=  "170";
when  171 => str3 <=  "171";
when  172 => str3 <=  "172";
when  173 => str3 <=  "173";
when  174 => str3 <=  "174";
when  175 => str3 <=  "175";
when  176 => str3 <=  "176";
when  177 => str3 <=  "177";
when  178 => str3 <=  "178";
when  179 => str3 <=  "179";
when  180 => str3 <=  "180";
when  181 => str3 <=  "181";
when  182 => str3 <=  "182";
when  183 => str3 <=  "183";
when  184 => str3 <=  "184";
when  185 => str3 <=  "185";
when  186 => str3 <=  "186";
when  187 => str3 <=  "187";
when  188 => str3 <=  "188";
when  189 => str3 <=  "189";
when  190 => str3 <=  "190";
when  191 => str3 <=  "191";
when  192 => str3 <=  "192";
when  193 => str3 <=  "193";
when  194 => str3 <=  "194";
when  195 => str3 <=  "195";
when  196 => str3 <=  "196";
when  197 => str3 <=  "197";
when  198 => str3 <=  "198";
when  199 => str3 <=  "199";
when  200 => str3 <=  "200";
when  201 => str3 <=  "201";
when  202 => str3 <=  "202";
when  203 => str3 <=  "203";
when  204 => str3 <=  "204";
when  205 => str3 <=  "205";
when  206 => str3 <=  "206";
when  207 => str3 <=  "207";
when  208 => str3 <=  "208";
when  209 => str3 <=  "209";
```

```
when  210 => str3 <=  "210";
when  211 => str3 <=  "211";
when  212 => str3 <=  "212";
when  213 => str3 <=  "213";
when  214 => str3 <=  "214";
when  215 => str3 <=  "215";
when  216 => str3 <=  "216";
when  217 => str3 <=  "217";
when  218 => str3 <=  "218";
when  219 => str3 <=  "219";
when  220 => str3 <=  "220";
when  221 => str3 <=  "221";
when  222 => str3 <=  "222";
when  223 => str3 <=  "223";
when  224 => str3 <=  "224";
when  225 => str3 <=  "225";
when  226 => str3 <=  "226";
when  227 => str3 <=  "227";
when  228 => str3 <=  "228";
when  229 => str3 <=  "229";
when  230 => str3 <=  "230";
when  231 => str3 <=  "231";
when  232 => str3 <=  "232";
when  233 => str3 <=  "233";
when  234 => str3 <=  "234";
when  235 => str3 <=  "235";
when  236 => str3 <=  "236";
when  237 => str3 <=  "237";
when  238 => str3 <=  "238";
when  239 => str3 <=  "239";
when  240 => str3 <=  "240";
when  241 => str3 <=  "241";
when  242 => str3 <=  "242";
when  243 => str3 <=  "243";
when  244 => str3 <=  "244";
when  245 => str3 <=  "245";
when  246 => str3 <=  "246";
when  247 => str3 <=  "247";
when  248 => str3 <=  "248";
when  249 => str3 <=  "249";
when  250 => str3 <=  "250";
when  251 => str3 <=  "251";
when  252 => str3 <=  "252";
when  253 => str3 <=  "253";
when  254 => str3 <=  "254";
when  255 => str3 <=  "255";
when  256 => str3 <=  "256";
when  257  => str3 <=  "257";
```

```
when  258  => str3 <=  "258";
when  259  => str3 <=  "259";
when  260  => str3 <=  "260";
when  261  => str3 <=  "261";
when  262  => str3 <=  "262";
when  263  => str3 <=  "263";
when  264  => str3 <=  "264";
when  265  => str3 <=  "265";
when  266  => str3 <=  "266";
when  267  => str3 <=  "267";
when  268  => str3 <=  "268";
when  269  => str3 <=  "269";
when  270  => str3 <=  "270";
when  271  => str3 <=  "271";
when  272  => str3 <=  "272";
when  273  => str3 <=  "273";
when  274  => str3 <=  "274";
when  275  => str3 <=  "275";
when  276  => str3 <=  "276";
when  277  => str3 <=  "277";
when  278  => str3 <=  "278";
when  279  => str3 <=  "279";
when  280  => str3 <=  "280";
when  281  => str3 <=  "281";
when  282  => str3 <=  "282";
when  283  => str3 <=  "283";
when  284  => str3 <=  "284";
when  285  => str3 <=  "285";
when  286  => str3 <=  "286";
when  287  => str3 <=  "287";
when  288  => str3 <=  "288";
when  289  => str3 <=  "289";
when  290  => str3 <=  "290";
when  291  => str3 <=  "291";
when  292  => str3 <=  "292";
when  293  => str3 <=  "293";
when  294  => str3 <=  "294";
when  295  => str3 <=  "295";
when  296  => str3 <=  "296";
when  297  => str3 <=  "297";
when  298  => str3 <=  "298";
when  299  => str3 <=  "299";
when  300  => str3 <=  "300";
when  301  => str3 <=  "301";
when  302  => str3 <=  "302";
when  303  => str3 <=  "303";
when  304  => str3 <=  "304";
when  305  => str3 <=  "305";
```

```
when  306  => str3 <=  "306";
when  307  => str3 <=  "307";
when  308  => str3 <=  "308";
when  309  => str3 <=  "309";
when  310  => str3 <=  "310";
when  311  => str3 <=  "311";
when  312  => str3 <=  "312";
when  313  => str3 <=  "313";
when  314  => str3 <=  "314";
when  315  => str3 <=  "315";
when  316  => str3 <=  "316";
when  317  => str3 <=  "317";
when  318  => str3 <=  "318";
when  319  => str3 <=  "319";
when  320  => str3 <=  "320";
when  321  => str3 <=  "321";
when  322  => str3 <=  "322";
when  323  => str3 <=  "323";
when  324  => str3 <=  "324";
when  325  => str3 <=  "325";
when  326  => str3 <=  "326";
when  327  => str3 <=  "327";
when  328  => str3 <=  "328";
when  329  => str3 <=  "329";
when  330  => str3 <=  "330";
when  331  => str3 <=  "331";
when  332  => str3 <=  "332";
when  333  => str3 <=  "333";
when  334  => str3 <=  "334";
when  335  => str3 <=  "335";
when  336  => str3 <=  "336";
when  337  => str3 <=  "337";
when  338  => str3 <=  "338";
when  339  => str3 <=  "339";
when  340  => str3 <=  "340";
when  341  => str3 <=  "341";
when  342  => str3 <=  "342";
when  343  => str3 <=  "343";
when  344  => str3 <=  "344";
when  345  => str3 <=  "345";
when  346  => str3 <=  "346";
when  347  => str3 <=  "347";
when  348  => str3 <=  "348";
when  349  => str3 <=  "349";
when  350  => str3 <=  "350";
when  351  => str3 <=  "351";
when  352  => str3 <=  "352";
when  353  => str3 <=  "353";
```

```
when  354  => str3 <=  "354";
when  355  => str3 <=  "355";
when  356  => str3 <=  "356";
when  357  => str3 <=  "357";
when  358  => str3 <=  "358";
when  359  => str3 <=  "359";
when  360  => str3 <=  "360";
when  361  => str3 <=  "361";
when  362  => str3 <=  "362";
when  363  => str3 <=  "363";
when  364  => str3 <=  "364";
when  365  => str3 <=  "365";
when  366  => str3 <=  "366";
when  367  => str3 <=  "367";
when  368  => str3 <=  "368";
when  369  => str3 <=  "369";
when  370  => str3 <=  "370";
when  371  => str3 <=  "371";
when  372  => str3 <=  "372";
when  373  => str3 <=  "373";
when  374  => str3 <=  "374";
when  375  => str3 <=  "375";
when  376  => str3 <=  "376";
when  377  => str3 <=  "377";
when  378  => str3 <=  "378";
when  379  => str3 <=  "379";
when  380  => str3 <=  "380";
when  381  => str3 <=  "381";
when  382  => str3 <=  "382";
when  383  => str3 <=  "383";
when  384  => str3 <=  "384";
when  385  => str3 <=  "385";
when  386  => str3 <=  "386";
when  387  => str3 <=  "387";
when  388  => str3 <=  "388";
when  389  => str3 <=  "389";
when  390  => str3 <=  "390";
when  391  => str3 <=  "391";
when  392  => str3 <=  "392";
when  393  => str3 <=  "393";
when  394  => str3 <=  "394";
when  395  => str3 <=  "395";
when  396  => str3 <=  "396";
when  397  => str3 <=  "397";
when  398  => str3 <=  "398";
when  399  => str3 <=  "399";
when  400  => str3 <=  "400";
when  401  => str3 <=  "401";
```

```
when  402  => str3 <=  "402";
when  403  => str3 <=  "403";
when  404  => str3 <=  "404";
when  405  => str3 <=  "405";
when  406  => str3 <=  "406";
when  407  => str3 <=  "407";
when  408  => str3 <=  "408";
when  409  => str3 <=  "409";
when  410  => str3 <=  "410";
when  411  => str3 <=  "411";
when  412  => str3 <=  "412";
when  413  => str3 <=  "413";
when  414  => str3 <=  "414";
when  415  => str3 <=  "415";
when  416  => str3 <=  "416";
when  417  => str3 <=  "417";
when  418  => str3 <=  "418";
when  419  => str3 <=  "419";
when  420  => str3 <=  "420";
when  421  => str3 <=  "421";
when  422  => str3 <=  "422";
when  423  => str3 <=  "423";
when  424  => str3 <=  "424";
when  425  => str3 <=  "425";
when  426  => str3 <=  "426";
when  427  => str3 <=  "427";
when  428  => str3 <=  "428";
when  429  => str3 <=  "429";
when  430  => str3 <=  "430";
when  431  => str3 <=  "431";
when  432  => str3 <=  "432";
when  433  => str3 <=  "433";
when  434  => str3 <=  "434";
when  435  => str3 <=  "435";
when  436  => str3 <=  "436";
when  437  => str3 <=  "437";
when  438  => str3 <=  "438";
when  439  => str3 <=  "439";
when  440  => str3 <=  "440";
when  441  => str3 <=  "441";
when  442  => str3 <=  "442";
when  443  => str3 <=  "443";
when  444  => str3 <=  "444";
when  445  => str3 <=  "445";
when  446  => str3 <=  "446";
when  447  => str3 <=  "447";
when  448  => str3 <=  "448";
when  449  => str3 <=  "449";
```

```
when  450  => str3 <=  "450";
when  451  => str3 <=  "451";
when  452  => str3 <=  "452";
when  453  => str3 <=  "453";
when  454  => str3 <=  "454";
when  455  => str3 <=  "455";
when  456  => str3 <=  "456";
when  457  => str3 <=  "457";
when  458  => str3 <=  "458";
when  459  => str3 <=  "459";
when  460  => str3 <=  "460";
when  461  => str3 <=  "461";
when  462  => str3 <=  "462";
when  463  => str3 <=  "463";
when  464  => str3 <=  "464";
when  465  => str3 <=  "465";
when  466  => str3 <=  "466";
when  467  => str3 <=  "467";
when  468  => str3 <=  "468";
when  469  => str3 <=  "469";
when  470  => str3 <=  "470";
when  471  => str3 <=  "471";
when  472  => str3 <=  "472";
when  473  => str3 <=  "473";
when  474  => str3 <=  "474";
when  475  => str3 <=  "475";
when  476  => str3 <=  "476";
when  477  => str3 <=  "477";
when  478  => str3 <=  "478";
when  479  => str3 <=  "479";
when  480  => str3 <=  "480";
when  481  => str3 <=  "481";
when  482  => str3 <=  "482";
when  483  => str3 <=  "483";
when  484  => str3 <=  "484";
when  485  => str3 <=  "485";
when  486  => str3 <=  "486";
when  487  => str3 <=  "487";
when  488  => str3 <=  "488";
when  489  => str3 <=  "489";
when  490  => str3 <=  "490";
when  491  => str3 <=  "491";
when  492  => str3 <=  "492";
when  493  => str3 <=  "493";
when  494  => str3 <=  "494";
when  495  => str3 <=  "495";
when  496  => str3 <=  "496";
when  497  => str3 <=  "497";
```

```
when  498  => str3 <=  "498";
when  499  => str3 <=  "499";
when  500  => str3 <=  "500";
when  501  => str3 <=  "501";
when  502  => str3 <=  "502";
when  503  => str3 <=  "503";
when  504  => str3 <=  "504";
when  505  => str3 <=  "505";
when  506  => str3 <=  "506";
when  507  => str3 <=  "507";
when  508  => str3 <=  "508";
when  509  => str3 <=  "509";
when  510  => str3 <=  "510";
when  511  => str3 <=  "511";
when  512  => str3 <=  "512";
when  513  => str3 <=  "513";
when  514  => str3 <=  "514";
when  515  => str3 <=  "515";
when  516  => str3 <=  "516";
when  517  => str3 <=  "517";
when  518  => str3 <=  "518";
when  519  => str3 <=  "519";
when  520  => str3 <=  "520";
when  521  => str3 <=  "521";
when  522  => str3 <=  "522";
when  523  => str3 <=  "523";
when  524  => str3 <=  "524";
when  525  => str3 <=  "525";
when  526  => str3 <=  "526";
when  527  => str3 <=  "527";
when  528  => str3 <=  "528";
when  529  => str3 <=  "529";
when  530  => str3 <=  "530";
when  531  => str3 <=  "531";
when  532  => str3 <=  "532";
when  533  => str3 <=  "533";
when  534  => str3 <=  "534";
when  535  => str3 <=  "535";
when  536  => str3 <=  "536";
when  537  => str3 <=  "537";
when  538  => str3 <=  "538";
when  539  => str3 <=  "539";
when  540  => str3 <=  "540";
when  541  => str3 <=  "541";
when  542  => str3 <=  "542";
when  543  => str3 <=  "543";
when  544  => str3 <=  "544";
when  545  => str3 <=  "545";
```

```
when  546  => str3 <=  "546";
when  547  => str3 <=  "547";
when  548  => str3 <=  "548";
when  549  => str3 <=  "549";
when  550  => str3 <=  "550";
when  551  => str3 <=  "551";
when  552  => str3 <=  "552";
when  553  => str3 <=  "553";
when  554  => str3 <=  "554";
when  555  => str3 <=  "555";
when  556  => str3 <=  "556";
when  557  => str3 <=  "557";
when  558  => str3 <=  "558";
when  559  => str3 <=  "559";
when  560  => str3 <=  "560";
when  561  => str3 <=  "561";
when  562  => str3 <=  "562";
when  563  => str3 <=  "563";
when  564  => str3 <=  "564";
when  565  => str3 <=  "565";
when  566  => str3 <=  "566";
when  567  => str3 <=  "567";
when  568  => str3 <=  "568";
when  569  => str3 <=  "569";
when  570  => str3 <=  "570";
when  571  => str3 <=  "571";
when  572  => str3 <=  "572";
when  573  => str3 <=  "573";
when  574  => str3 <=  "574";
when  575  => str3 <=  "575";
when  576  => str3 <=  "576";
when  577  => str3 <=  "577";
when  578  => str3 <=  "578";
when  579  => str3 <=  "579";
when  580  => str3 <=  "580";
when  581  => str3 <=  "581";
when  582  => str3 <=  "582";
when  583  => str3 <=  "583";
when  584  => str3 <=  "584";
when  585  => str3 <=  "585";
when  586  => str3 <=  "586";
when  587  => str3 <=  "587";
when  588  => str3 <=  "588";
when  589  => str3 <=  "589";
when  590  => str3 <=  "590";
when  591  => str3 <=  "591";
when  592  => str3 <=  "592";
when  593  => str3 <=  "593";
```

```
when  594  => str3 <=  "594";
when  595  => str3 <=  "595";
when  596  => str3 <=  "596";
when  597  => str3 <=  "597";
when  598  => str3 <=  "598";
when  599  => str3 <=  "599";
when  600  => str3 <=  "600";
when  601  => str3 <=  "601";
when  602  => str3 <=  "602";
when  603  => str3 <=  "603";
when  604  => str3 <=  "604";
when  605  => str3 <=  "605";
when  606  => str3 <=  "606";
when  607  => str3 <=  "607";
when  608  => str3 <=  "608";
when  609  => str3 <=  "609";
when  610  => str3 <=  "610";
when  611  => str3 <=  "611";
when  612  => str3 <=  "612";
when  613  => str3 <=  "613";
when  614  => str3 <=  "614";
when  615  => str3 <=  "615";
when  616  => str3 <=  "616";
when  617  => str3 <=  "617";
when  618  => str3 <=  "618";
when  619  => str3 <=  "619";
when  620  => str3 <=  "620";
when  621  => str3 <=  "621";
when  622  => str3 <=  "622";
when  623  => str3 <=  "623";
when  624  => str3 <=  "624";
when  625  => str3 <=  "625";
when  626  => str3 <=  "626";
when  627  => str3 <=  "627";
when  628  => str3 <=  "628";
when  629  => str3 <=  "629";
when  630  => str3 <=  "630";
when  631  => str3 <=  "631";
when  632  => str3 <=  "632";
when  633  => str3 <=  "633";
when  634  => str3 <=  "634";
when  635  => str3 <=  "635";
when  636  => str3 <=  "636";
when  637  => str3 <=  "637";
when  638  => str3 <=  "638";
when  639  => str3 <=  "639";
when  640  => str3 <=  "640";
when  641  => str3 <=  "641";
```

```
when  642  => str3 <=  "642";
when  643  => str3 <=  "643";
when  644  => str3 <=  "644";
when  645  => str3 <=  "645";
when  646  => str3 <=  "646";
when  647  => str3 <=  "647";
when  648  => str3 <=  "648";
when  649  => str3 <=  "649";
when  650  => str3 <=  "650";
when  651  => str3 <=  "651";
when  652  => str3 <=  "652";
when  653  => str3 <=  "653";
when  654  => str3 <=  "654";
when  655  => str3 <=  "655";
when  656  => str3 <=  "656";
when  657  => str3 <=  "657";
when  658  => str3 <=  "658";
when  659  => str3 <=  "659";
when  660  => str3 <=  "660";
when  661  => str3 <=  "661";
when  662  => str3 <=  "662";
when  663  => str3 <=  "663";
when  664  => str3 <=  "664";
when  665  => str3 <=  "665";
when  666  => str3 <=  "666";
when  667  => str3 <=  "667";
when  668  => str3 <=  "668";
when  669  => str3 <=  "669";
when  670  => str3 <=  "670";
when  671  => str3 <=  "671";
when  672  => str3 <=  "672";
when  673  => str3 <=  "673";
when  674  => str3 <=  "674";
when  675  => str3 <=  "675";
when  676  => str3 <=  "676";
when  677  => str3 <=  "677";
when  678  => str3 <=  "678";
when  679  => str3 <=  "679";
when  680  => str3 <=  "680";
when  681  => str3 <=  "681";
when  682  => str3 <=  "682";
when  683  => str3 <=  "683";
when  684  => str3 <=  "684";
when  685  => str3 <=  "685";
when  686  => str3 <=  "686";
when  687  => str3 <=  "687";
when  688  => str3 <=  "688";
when  689  => str3 <=  "689";
```

```
when  690  => str3 <=  "690";
when  691  => str3 <=  "691";
when  692  => str3 <=  "692";
when  693  => str3 <=  "693";
when  694  => str3 <=  "694";
when  695  => str3 <=  "695";
when  696  => str3 <=  "696";
when  697  => str3 <=  "697";
when  698  => str3 <=  "698";
when  699  => str3 <=  "699";
when  700  => str3 <=  "700";
when  701  => str3 <=  "701";
when  702  => str3 <=  "702";
when  703  => str3 <=  "703";
when  704  => str3 <=  "704";
when  705  => str3 <=  "705";
when  706  => str3 <=  "706";
when  707  => str3 <=  "707";
when  708  => str3 <=  "708";
when  709  => str3 <=  "709";
when  710  => str3 <=  "710";
when  711  => str3 <=  "711";
when  712  => str3 <=  "712";
when  713  => str3 <=  "713";
when  714  => str3 <=  "714";
when  715  => str3 <=  "715";
when  716  => str3 <=  "716";
when  717  => str3 <=  "717";
when  718  => str3 <=  "718";
when  719  => str3 <=  "719";
when  720  => str3 <=  "720";
when  721  => str3 <=  "721";
when  722  => str3 <=  "722";
when  723  => str3 <=  "723";
when  724  => str3 <=  "724";
when  725  => str3 <=  "725";
when  726  => str3 <=  "726";
when  727  => str3 <=  "727";
when  728  => str3 <=  "728";
when  729  => str3 <=  "729";
when  730  => str3 <=  "730";
when  731  => str3 <=  "731";
when  732  => str3 <=  "732";
when  733  => str3 <=  "733";
when  734  => str3 <=  "734";
when  735  => str3 <=  "735";
when  736  => str3 <=  "736";
when  737  => str3 <=  "737";
```

```
when  738  => str3 <=  "738";
when  739  => str3 <=  "739";
when  740  => str3 <=  "740";
when  741  => str3 <=  "741";
when  742  => str3 <=  "742";
when  743  => str3 <=  "743";
when  744  => str3 <=  "744";
when  745  => str3 <=  "745";
when  746  => str3 <=  "746";
when  747  => str3 <=  "747";
when  748  => str3 <=  "748";
when  749  => str3 <=  "749";
when  750  => str3 <=  "750";
when  751  => str3 <=  "751";
when  752  => str3 <=  "752";
when  753  => str3 <=  "753";
when  754  => str3 <=  "754";
when  755  => str3 <=  "755";
when  756  => str3 <=  "756";
when  757  => str3 <=  "757";
when  758  => str3 <=  "758";
when  759  => str3 <=  "759";
when  760  => str3 <=  "760";
when  761  => str3 <=  "761";
when  762  => str3 <=  "762";
when  763  => str3 <=  "763";
when  764  => str3 <=  "764";
when  765  => str3 <=  "765";
when  766  => str3 <=  "766";
when  767  => str3 <=  "767";
when  768  => str3 <=  "768";
when  769  => str3 <=  "769";
when  770  => str3 <=  "770";
when  771  => str3 <=  "771";
when  772  => str3 <=  "772";
when  773  => str3 <=  "773";
when  774  => str3 <=  "774";
when  775  => str3 <=  "775";
when  776  => str3 <=  "776";
when  777  => str3 <=  "777";
when  778  => str3 <=  "778";
when  779  => str3 <=  "779";
when  780  => str3 <=  "780";
when  781  => str3 <=  "781";
when  782  => str3 <=  "782";
when  783  => str3 <=  "783";
when  784  => str3 <=  "784";
when  785  => str3 <=  "785";
```

```
when  786  => str3 <=  "786";
when  787  => str3 <=  "787";
when  788  => str3 <=  "788";
when  789  => str3 <=  "789";
when  790  => str3 <=  "790";
when  791  => str3 <=  "791";
when  792  => str3 <=  "792";
when  793  => str3 <=  "793";
when  794  => str3 <=  "794";
when  795  => str3 <=  "795";
when  796  => str3 <=  "796";
when  797  => str3 <=  "797";
when  798  => str3 <=  "798";
when  799  => str3 <=  "799";
when  800  => str3 <=  "800";
when  801  => str3 <=  "801";
when  802  => str3 <=  "802";
when  803  => str3 <=  "803";
when  804  => str3 <=  "804";
when  805  => str3 <=  "805";
when  806  => str3 <=  "806";
when  807  => str3 <=  "807";
when  808  => str3 <=  "808";
when  809  => str3 <=  "809";
when  810  => str3 <=  "810";
when  811  => str3 <=  "811";
when  812  => str3 <=  "812";
when  813  => str3 <=  "813";
when  814  => str3 <=  "814";
when  815  => str3 <=  "815";
when  816  => str3 <=  "816";
when  817  => str3 <=  "817";
when  818  => str3 <=  "818";
when  819  => str3 <=  "819";
when  820  => str3 <=  "820";
when  821  => str3 <=  "821";
when  822  => str3 <=  "822";
when  823  => str3 <=  "823";
when  824  => str3 <=  "824";
when  825  => str3 <=  "825";
when  826  => str3 <=  "826";
when  827  => str3 <=  "827";
when  828  => str3 <=  "828";
when  829  => str3 <=  "829";
when  830  => str3 <=  "830";
when  831  => str3 <=  "831";
when  832  => str3 <=  "832";
when  833  => str3 <=  "833";
```

```
when  834  => str3 <=  "834";
when  835  => str3 <=  "835";
when  836  => str3 <=  "836";
when  837  => str3 <=  "837";
when  838  => str3 <=  "838";
when  839  => str3 <=  "839";
when  840  => str3 <=  "840";
when  841  => str3 <=  "841";
when  842  => str3 <=  "842";
when  843  => str3 <=  "843";
when  844  => str3 <=  "844";
when  845  => str3 <=  "845";
when  846  => str3 <=  "846";
when  847  => str3 <=  "847";
when  848  => str3 <=  "848";
when  849  => str3 <=  "849";
when  850  => str3 <=  "850";
when  851  => str3 <=  "851";
when  852  => str3 <=  "852";
when  853  => str3 <=  "853";
when  854  => str3 <=  "854";
when  855  => str3 <=  "855";
when  856  => str3 <=  "856";
when  857  => str3 <=  "857";
when  858  => str3 <=  "858";
when  859  => str3 <=  "859";
when  860  => str3 <=  "860";
when  861  => str3 <=  "861";
when  862  => str3 <=  "862";
when  863  => str3 <=  "863";
when  864  => str3 <=  "864";
when  865  => str3 <=  "865";
when  866  => str3 <=  "866";
when  867  => str3 <=  "867";
when  868  => str3 <=  "868";
when  869  => str3 <=  "869";
when  870  => str3 <=  "870";
when  871  => str3 <=  "871";
when  872  => str3 <=  "872";
when  873  => str3 <=  "873";
when  874  => str3 <=  "874";
when  875  => str3 <=  "875";
when  876  => str3 <=  "876";
when  877  => str3 <=  "877";
when  878  => str3 <=  "878";
when  879  => str3 <=  "879";
when  880  => str3 <=  "880";
when  881  => str3 <=  "881";
```

```
when  882  => str3 <=  "882";
when  883  => str3 <=  "883";
when  884  => str3 <=  "884";
when  885  => str3 <=  "885";
when  886  => str3 <=  "886";
when  887  => str3 <=  "887";
when  888  => str3 <=  "888";
when  889  => str3 <=  "889";
when  890  => str3 <=  "890";
when  891  => str3 <=  "891";
when  892  => str3 <=  "892";
when  893  => str3 <=  "893";
when  894  => str3 <=  "894";
when  895  => str3 <=  "895";
when  896  => str3 <=  "896";
when  897  => str3 <=  "897";
when  898  => str3 <=  "898";
when  899  => str3 <=  "899";
when  900  => str3 <=  "900";
when  901  => str3 <=  "901";
when  902  => str3 <=  "902";
when  903  => str3 <=  "903";
when  904  => str3 <=  "904";
when  905  => str3 <=  "905";
when  906  => str3 <=  "906";
when  907  => str3 <=  "907";
when  908  => str3 <=  "908";
when  909  => str3 <=  "909";
when  910  => str3 <=  "910";
when  911  => str3 <=  "911";
when  912  => str3 <=  "912";
when  913  => str3 <=  "913";
when  914  => str3 <=  "914";
when  915  => str3 <=  "915";
when  916  => str3 <=  "916";
when  917  => str3 <=  "917";
when  918  => str3 <=  "918";
when  919  => str3 <=  "919";
when  920  => str3 <=  "920";
when  921  => str3 <=  "921";
when  922  => str3 <=  "922";
when  923  => str3 <=  "923";
when  924  => str3 <=  "924";
when  925  => str3 <=  "925";
when  926  => str3 <=  "926";
when  927  => str3 <=  "927";
when  928  => str3 <=  "928";
when  929  => str3 <=  "929";
```

```
when  930  => str3 <=  "930";
when  931  => str3 <=  "931";
when  932  => str3 <=  "932";
when  933  => str3 <=  "933";
when  934  => str3 <=  "934";
when  935  => str3 <=  "935";
when  936  => str3 <=  "936";
when  937  => str3 <=  "937";
when  938  => str3 <=  "938";
when  939  => str3 <=  "939";
when  940  => str3 <=  "940";
when  941  => str3 <=  "941";
when  942  => str3 <=  "942";
when  943  => str3 <=  "943";
when  944  => str3 <=  "944";
when  945  => str3 <=  "945";
when  946  => str3 <=  "946";
when  947  => str3 <=  "947";
when  948  => str3 <=  "948";
when  949  => str3 <=  "949";
when  950  => str3 <=  "950";
when  951  => str3 <=  "951";
when  952  => str3 <=  "952";
when  953  => str3 <=  "953";
when  954  => str3 <=  "954";
when  955  => str3 <=  "955";
when  956  => str3 <=  "956";
when  957  => str3 <=  "957";
when  958  => str3 <=  "958";
when  959  => str3 <=  "959";
when  960  => str3 <=  "960";
when  961  => str3 <=  "961";
when  962  => str3 <=  "962";
when  963  => str3 <=  "963";
when  964  => str3 <=  "964";
when  965  => str3 <=  "965";
when  966  => str3 <=  "966";
when  967  => str3 <=  "967";
when  968  => str3 <=  "968";
when  969  => str3 <=  "969";
when  970  => str3 <=  "970";
when  971  => str3 <=  "971";
when  972  => str3 <=  "972";
when  973  => str3 <=  "973";
when  974  => str3 <=  "974";
when  975  => str3 <=  "975";
when  976  => str3 <=  "976";
when  977  => str3 <=  "977";
```

```
   when  978  => str3 <=  "978";
   when  979  => str3 <=  "979";
   when  980  => str3 <=  "980";
   when  981  => str3 <=  "981";
   when  982  => str3 <=  "982";
   when  983  => str3 <=  "983";
   when  984  => str3 <=  "984";
   when  985  => str3 <=  "985";
   when  986  => str3 <=  "986";
   when  987  => str3 <=  "987";
   when  988  => str3 <=  "988";
   when  989  => str3 <=  "989";
   when  990  => str3 <=  "990";
   when  991  => str3 <=  "991";
   when  992  => str3 <=  "992";
   when  993  => str3 <=  "993";
   when  994  => str3 <=  "994";
   when  995  => str3 <=  "995";
   when  996  => str3 <=  "996";
   when  997  => str3 <=  "997";
   when  998  => str3 <=  "998";
   when  999  => str3 <=  "999";
   when others => str3 <= "   ";
   end case;




 END IF;
 END PROCESS;
 END MAIN;
```


Text generation codes: from https://github.com/Derek-X-Wang/VGA-Text-Generator

Pixel_on_text:

```
-- Pixel_On_Text determines if the current pixel is on text
-- param:
--   textlength, use to init the string
-- input:
```

```
--   VGA clock(the clk you used to update VGA)
--   display text
--   top left corner of the text box
--   current X and Y position
-- output:
--   a bit that represent whether is the pixel in text

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

-- note this line.The package is compiled to this directory by default.
-- so don't forget to include this directory.
library work;
-- this line also is must.This includes the particular package into your program.
use work.commonPak.all;

entity Pixel_On_Text is
        generic(
          -- needed for init displayText, the default value 11 is just a random number
      textLength: integer := 11
        );
        port (
                clk: in std_logic;
                displayText: in string (1 to textLength) := (others => NUL);
                -- top left corner of the text
                position: in point_2d := (0, 0);
                -- current pixel postion
                horzCoord: in integer;
                vertCoord: in integer;

                pixel: out std_logic := '0'
        );

end Pixel_On_Text;

architecture Behavioral of Pixel_On_Text is

        signal fontAddress: integer;
        -- A row of bit in a charactor, we check if our current (x,y) is 1 in char row
```

```
        signal charBitInRow: std_logic_vector(FONT_WIDTH-1 downto 0) := (others => '0');
        -- char in ASCII code
        signal charCode:integer := 0;
        -- the position(column) of a charactor in the given text
        signal charPosition:integer := 0;
        -- the bit position(column) in a charactor
        signal bitPosition:integer := 0;
begin
   -- (horzCoord - position.x): x positionin the top left of the whole text
   charPosition <= (horzCoord - position.x)/FONT_WIDTH + 1;
   bitPosition <= (horzCoord - position.x) mod FONT_WIDTH;
   charCode <= character'pos(displayText(charPosition));
   -- charCode*16: first row of the char
   fontAddress <= charCode*16+(vertCoord - position.y);


        fontRom: entity work.Font_Rom
        port map(
                clk => clk,
                addr => fontAddress,
                fontRow => charBitInRow
        );

        pixelOn: process(clk)
                variable inXRange: boolean := false;
                variable inYRange: boolean := false;
        begin
      if rising_edge(clk) then

        -- reset
        inXRange := false;
        inYRange := false;
        pixel <= '0';
        -- If current pixel is in the horizontal range of text
        if horzCoord >= position.x and horzCoord < position.x + (FONT_WIDTH * textlength)
then
            inXRange := true;
        end if;

        -- If current pixel is in the vertical range of text
        if vertCoord >= position.y and vertCoord < position.y + FONT_HEIGHT then
            inYRange := true;
        end if;

        -- need to check if the pixel is on for text
        if inXRange and inYRange then
            -- FONT_WIDTH-bitPosition: we are reverting the charactor
            if charBitInRow(FONT_WIDTH-bitPosition) = '1' then
```

```
                pixel <= '1';
            end if;
        end if;

                end if;
        end process;

end Behavioral;
```


---

commonPak:

```
--  Original Source from FP-V-GA-Text: https://github.com/MadLittleMods/FP-V-GA-Text
--
--      Package File Template
--
--      Purpose: This package defines supplemental types, subtypes,
--              constants, and functions
--
--   To use any of the example code shown below, uncomment the lines and modify as
necessary
--

library IEEE;
use IEEE.STD_LOGIC_1164.all;

use ieee.math_real.all;

package commonPak is



    constant ADDR_WIDTH : integer := 11;
    constant DATA_WIDTH : integer := 8;

    constant FONT_WIDTH : integer := 8;
    constant FONT_HEIGHT : integer := 16;


----------------------------------------
```

```
	type point_2d is
	record
		x : integer;
		y : integer;
	end record;

	type type_textColorMap is array(natural range <>) of std_logic_vector(7 downto 0);


----------------------------------------


	type type_drawElement is
	record
		pixelOn: boolean;
		rgb: std_logic_vector(7 downto 0);
	end record;
	constant init_type_drawElement: type_drawElement := (pixelOn => false, rgb =>
(others => '0'));
	type type_drawElementArray is array(natural range <>) of type_drawElement;



----------------------------------------

	type type_inArbiterPort is
	record
		dataRequest: boolean;
		addr: std_logic_vector(ADDR_WIDTH-1 downto 0);
		writeRequest: boolean;
		writeData: std_logic_vector(DATA_WIDTH-1 downto 0);
	end record;
	constant init_type_inArbiterPort: type_inArbiterPort := (dataRequest => false, addr =>
(others => '0'), writeRequest => false, writeData  => (others => '0'));
	type type_inArbiterPortArray is array(natural range <>) of type_inArbiterPort;


	type type_outArbiterPort is
	record
		dataWaiting: boolean;
		data: std_logic_vector(DATA_WIDTH-1 downto 0);
		dataWritten: boolean;
	end record;
	constant init_type_outArbiterPort: type_outArbiterPort := (dataWaiting => false, data
=> (others => '0'), dataWritten => false);
	type type_outArbiterPortArray is array(natural range <>) of type_outArbiterPort;
```

```
----------------------

        function log2_float(val : positive) return natural;

end commonPak;

package body commonPak is
        function log2_float(val : positive) return natural is
        begin
                return integer(ceil(log2(real(val))));
        end function;
end commonPak;




------------------------------------------------------------
------------------------------------------------------------
----------------------------------------------------------

Font_Rom:
-- character ROM
--   - 8-by-16 (8-by-2^4) font
--   - 128 (2^7) characters
--   - ROM size: 512-by-8 (2^11-by-8) bits
--              16K bits: 1 BRAM

-- I got this copy from FP-V-GA-Text: https://github.com/MadLittleMods/FP-V-GA-Text
-- and I remove some functionalities to keep it as simply as possiable
-- Original Source: https://github.com/thelonious/vga_generator/tree/master/vga_text

-- VHDL'93 supports the full table of ISO-8859-1 characters (0x00 through 0xFF(255))
-- ISO-8859-1 Table: http://kireji.com/reference/iso88591.html

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

-- note this line.The package is compiled to this directory by default.
```

```
-- so don't forget to include this directory.
library work;
-- this line also is must.This includes the particular package into your program.
use work.commonPak.all;

entity Font_Rom is
	port(
		clk: in std_logic;
		addr: in integer;
		fontRow: out std_logic_vector(FONT_WIDTH-1 downto 0)
	);
end Font_Rom;

architecture Behavioral of Font_Rom is

   -- 2^7 charactors
   -- + 2^4 row per charactor
   -- therefore the total array size is 2^11 = 2048
	type rom_type is array (0 to 2**11-1) of std_logic_vector(FONT_WIDTH-1 downto 0);

	-- ROM definition
	signal ROM: rom_type := (   -- 2^11-by-8
		-- NUL: code x00
		"00000000", -- 0
		"00000000", -- 1
		"00000000", -- 2
		"00000000", -- 3
		"00000000", -- 4
		"00000000", -- 5
		"00000000", -- 6
		"00000000", -- 7
		"00000000", -- 8
		"00000000", -- 9
		"00000000", -- a
		"00000000", -- b
		"00000000", -- c
		"00000000", -- d
		"00000000", -- e
		"00000000", -- f
		-- SOH: code x01
		"00000000", -- 0
		"00000000", -- 1
		"01111110", -- 2  ******
		"10000001", -- 3 *      *
		"10100101", -- 4 * *  * *
		"10000001", -- 5 *      *
		"10000001", -- 6 *      *
		"10111101", -- 7 * **** *
```

```
"10011001", -- 8 *  **  *
"10000001", -- 9 *      *
"10000001", -- a *      *
"01111110", -- b  ******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- STX: code x02
"00000000", -- 0
"00000000", -- 1
"01111110", -- 2  ******
"11111111", -- 3 ********
"11011011", -- 4 ** ** **
"11111111", -- 5 ********
"11111111", -- 6 ********
"11000011", -- 7 **    **
"11100111", -- 8 ***  ***
"11111111", -- 9 ********
"11111111", -- a ********
"01111110", -- b  ******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- ETX: code x03
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"01101100", -- 4  ** **
"11111110", -- 5 *******
"11111110", -- 6 *******
"11111110", -- 7 *******
"11111110", -- 8 *******
"01111100", -- 9  *****
"00111000", -- a   ***
"00010000", -- b    *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- EOT: code x04
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00010000", -- 4    *
```

```
"00111000", -- 5   ***
"01111100", -- 6  *****
"11111110", -- 7 *******
"01111100", -- 8  *****
"00111000", -- 9   ***
"00010000", -- a    *
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- ENQ: code x05
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00011000", -- 3    **
"00111100", -- 4   ****
"00111100", -- 5   ****
"11100111", -- 6 ***  ***
"11100111", -- 7 ***  ***
"11100111", -- 8 ***  ***
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- ACK: code x06
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00011000", -- 3    **
"00111100", -- 4   ****
"01111110", -- 5  ******
"11111111", -- 6 ********
"11111111", -- 7 ********
"01111110", -- 8  ******
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- BEL: code x07
"00000000", -- 0
"00000000", -- 1
```

```
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00011000", -- 6    **
"00111100", -- 7   ****
"00111100", -- 8   ****
"00011000", -- 9    **
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- BS: code x08
"11111111", -- 0 ********
"11111111", -- 1 ********
"11111111", -- 2 ********
"11111111", -- 3 ********
"11111111", -- 4 ********
"11111111", -- 5 ********
"11100111", -- 6 ***  ***
"11000011", -- 7 **    **
"11000011", -- 8 **    **
"11100111", -- 9 ***  ***
"11111111", -- a ********
"11111111", -- b ********
"11111111", -- c ********
"11111111", -- d ********
"11111111", -- e ********
"11111111", -- f ********
-- HT: code x09
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00111100", -- 5   ****
"01100110", -- 6  **  **
"01000010", -- 7  *    *
"01000010", -- 8  *    *
"01100110", -- 9  **  **
"00111100", -- a   ****
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
```

```
-- LF: code x0a
"11111111", -- 0 ********
"11111111", -- 1 ********
"11111111", -- 2 ********
"11111111", -- 3 ********
"11111111", -- 4 ********
"11000011", -- 5 **    **
"10011001", -- 6 *  **  *
"10111101", -- 7 * **** *
"10111101", -- 8 * **** *
"10011001", -- 9 *  **  *
"11000011", -- a **    **
"11111111", -- b ********
"11111111", -- c ********
"11111111", -- d ********
"11111111", -- e ********
"11111111", -- f ********
-- code x0b
"00000000", -- 0
"00000000", -- 1
"00011110", -- 2    ****
"00001110", -- 3     ***
"00011010", -- 4    ** *
"00110010", -- 5   **  *
"01111000", -- 6  ****
"11001100", -- 7 **  **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01111000", -- b  ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x0c
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01100110", -- 6  **  **
"00111100", -- 7   ****
"00011000", -- 8    **
"01111110", -- 9  ******
"00011000", -- a    **
"00011000", -- b    **
"00000000", -- c
```

```
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x0d
"00000000", -- 0
"00000000", -- 1
"00111111", -- 2   ******
"00110011", -- 3   **  **
"00111111", -- 4   ******
"00110000", -- 5   **
"00110000", -- 6   **
"00110000", -- 7   **
"00110000", -- 8   **
"01110000", -- 9  ***
"11110000", -- a ****
"11100000", -- b ***
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x0e
"00000000", -- 0
"00000000", -- 1
"01111111", -- 2  *******
"01100011", -- 3  **   **
"01111111", -- 4  *******
"01100011", -- 5  **   **
"01100011", -- 6  **   **
"01100011", -- 7  **   **
"01100011", -- 8  **   **
"01100111", -- 9  **  ***
"11100111", -- a ***  ***
"11100110", -- b ***  **
"11000000", -- c **
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x0f
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00011000", -- 3    **
"00011000", -- 4    **
"11011011", -- 5 ** ** **
"00111100", -- 6   ****
"11100111", -- 7 ***  ***
"00111100", -- 8   ****
"11011011", -- 9 ** ** **
```

```
"00011000", -- a    **
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x10
"00000000", -- 0
"10000000", -- 1 *
"11000000", -- 2 **
"11100000", -- 3 ***
"11110000", -- 4 ****
"11111000", -- 5 *****
"11111110", -- 6 *******
"11111000", -- 7 *****
"11110000", -- 8 ****
"11100000", -- 9 ***
"11000000", -- a **
"10000000", -- b *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x11
"00000000", -- 0
"00000010", -- 1       *
"00000110", -- 2      **
"00001110", -- 3     ***
"00011110", -- 4    ****
"00111110", -- 5   *****
"11111110", -- 6 *******
"00111110", -- 7   *****
"00011110", -- 8    ****
"00001110", -- 9     ***
"00000110", -- a      **
"00000010", -- b       *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x12
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2    **
"00111100", -- 3   ****
"01111110", -- 4  ******
"00011000", -- 5    **
"00011000", -- 6    **
```

```
"00011000", -- 7    **
"01111110", -- 8  ******
"00111100", -- 9   ****
"00011000", -- a    **
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x13
"00000000", -- 0
"00000000", -- 1
"01100110", -- 2  **  **
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01100110", -- 6  **  **
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"00000000", -- 9
"01100110", -- a  **  **
"01100110", -- b  **  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x14
"00000000", -- 0
"00000000", -- 1
"01111111", -- 2  *******
"11011011", -- 3 ** ** **
"11011011", -- 4 ** ** **
"11011011", -- 5 ** ** **
"01111011", -- 6  **** **
"00011011", -- 7    ** **
"00011011", -- 8    ** **
"00011011", -- 9    ** **
"00011011", -- a    ** **
"00011011", -- b    ** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x15
"00000000", -- 0
"01111100", -- 1  *****
"11000110", -- 2 **   **
"01100000", -- 3  **
```

```
"00111000", -- 4   ***
"01101100", -- 5  ** **
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"01101100", -- 8  ** **
"00111000", -- 9   ***
"00001100", -- a     **
"11000110", -- b **   **
"01111100", -- c  *****
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x16
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"11111110", -- 8 *******
"11111110", -- 9 *******
"11111110", -- a *******
"11111110", -- b *******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x17
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2    **
"00111100", -- 3   ****
"01111110", -- 4  ******
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"01111110", -- 8  ******
"00111100", -- 9   ****
"00011000", -- a    **
"01111110", -- b  ******
"00110000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x18
"00000000", -- 0
```

```
"00000000", -- 1
"00011000", -- 2    **
"00111100", -- 3   ****
"01111110", -- 4  ******
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x19
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2    **
"00011000", -- 3    **
"00011000", -- 4    **
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"01111110", -- 9  ******
"00111100", -- a   ****
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x1a
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00011000", -- 5    **
"00001100", -- 6     **
"11111110", -- 7 *******
"00001100", -- 8     **
"00011000", -- 9    **
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
```

```
"00000000", -- f
-- code x1b
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00110000", -- 5   **
"01100000", -- 6  **
"11111110", -- 7 *******
"01100000", -- 8  **
"00110000", -- 9   **
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x1c
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"11000000", -- 6 **
"11000000", -- 7 **
"11000000", -- 8 **
"11111110", -- 9 *******
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x1d
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00100100", -- 5   *  *
"01100110", -- 6  **  **
"11111111", -- 7 ********
"01100110", -- 8  **  **
"00100100", -- 9   *  *
"00000000", -- a
"00000000", -- b
```

```
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x1e
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00010000", -- 4    *
"00111000", -- 5   ***
"00111000", -- 6   ***
"01111100", -- 7  *****
"01111100", -- 8  *****
"11111110", -- 9 *******
"11111110", -- a *******
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x1f
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"11111110", -- 4 *******
"11111110", -- 5 *******
"01111100", -- 6  *****
"01111100", -- 7  *****
"00111000", -- 8   ***
"00111000", -- 9   ***
"00010000", -- a    *
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x20
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
```

```
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x21
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2    **
"00111100", -- 3   ****
"00111100", -- 4   ****
"00111100", -- 5   ****
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00000000", -- 9
"00011000", -- a    **
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x22
"00000000", -- 0
"01100110", -- 1  **  **
"01100110", -- 2  **  **
"01100110", -- 3  **  **
"00100100", -- 4   *  *
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x23
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"01101100", -- 3  ** **
"01101100", -- 4  ** **
"11111110", -- 5 *******
```

```
"01101100", -- 6  ** **
"01101100", -- 7  ** **
"01101100", -- 8  ** **
"11111110", -- 9 *******
"01101100", -- a  ** **
"01101100", -- b  ** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x24
"00011000", -- 0     **
"00011000", -- 1     **
"01111100", -- 2   *****
"11000110", -- 3  **   **
"11000010", -- 4  **    *
"11000000", -- 5  **
"01111100", -- 6   *****
"00000110", -- 7       **
"00000110", -- 8       **
"10000110", -- 9  *    **
"11000110", -- a  **   **
"01111100", -- b   *****
"00011000", -- c     **
"00011000", -- d     **
"00000000", -- e
"00000000", -- f
-- code x25
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"11000010", -- 4 **    *
"11000110", -- 5 **   **
"00001100", -- 6     **
"00011000", -- 7    **
"00110000", -- 8   **
"01100000", -- 9  **
"11000110", -- a **   **
"10000110", -- b *    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x26
"00000000", -- 0
"00000000", -- 1
"00111000", -- 2   ***
```

```
"01101100", -- 3  ** **
"01101100", -- 4  ** **
"00111000", -- 5   ***
"01110110", -- 6  *** **
"11011100", -- 7 ** ***
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01110110", -- b  *** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x27
"00000000", -- 0
"00110000", -- 1   **
"00110000", -- 2   **
"00110000", -- 3   **
"01100000", -- 4  **
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x28
"00000000", -- 0
"00000000", -- 1
"00001100", -- 2    **
"00011000", -- 3   **
"00110000", -- 4   **
"00110000", -- 5   **
"00110000", -- 6   **
"00110000", -- 7   **
"00110000", -- 8   **
"00110000", -- 9   **
"00011000", -- a   **
"00001100", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x29
```

```
"00000000", -- 0
"00000000", -- 1
"00110000", -- 2   **
"00011000", -- 3    **
"00001100", -- 4     **
"00001100", -- 5     **
"00001100", -- 6     **
"00001100", -- 7     **
"00001100", -- 8     **
"00001100", -- 9     **
"00011000", -- a    **
"00110000", -- b   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x2a
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01100110", -- 5  **  **
"00111100", -- 6   ****
"11111111", -- 7 ********
"00111100", -- 8   ****
"01100110", -- 9  **  **
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x2b
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00011000", -- 5    **
"00011000", -- 6    **
"01111110", -- 7  ******
"00011000", -- 8    **
"00011000", -- 9    **
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
```

```
"00000000", -- e
"00000000", -- f
-- code x2c
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00011000", -- 9    **
"00011000", -- a    **
"00011000", -- b    **
"00110000", -- c   **
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x2d
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"01111110", -- 7  ******
"00000000", -- 8
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x2e
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
"00011000", -- a    **
```

```
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x2f
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000010", -- 4       *
"00000110", -- 5      **
"00001100", -- 6     **
"00011000", -- 7    **
"00110000", -- 8   **
"01100000", -- 9  **
"11000000", -- a **
"10000000", -- b *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- 0: code x30
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11001110", -- 5 **  ***
"11011110", -- 6 ** ****
"11110110", -- 7 **** **
"11100110", -- 8 ***  **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- 1: code x31
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2
"00111000", -- 3
"01111000", -- 4    **
"00011000", -- 5   ***
"00011000", -- 6  ****
"00011000", -- 7    **
```

```
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"01111110", -- b    **
"00000000", -- c    **
"00000000", -- d  ******
"00000000", -- e
"00000000", -- f
-- 2: code x32
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"00000110", -- 4      **
"00001100", -- 5     **
"00011000", -- 6    **
"00110000", -- 7   **
"01100000", -- 8  **
"11000000", -- 9 **
"11000110", -- a **   **
"11111110", -- b *******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- 3: code x33
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"00000110", -- 4      **
"00000110", -- 5      **
"00111100", -- 6   ****
"00000110", -- 7      **
"00000110", -- 8      **
"00000110", -- 9      **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- 4: code x34
"00000000", -- 0
"00000000", -- 1
"00001100", -- 2     **
"00011100", -- 3    ***
"00111100", -- 4   ****
```

```
"01101100", -- 5  ** **
"11001100", -- 6 **  **
"11111110", -- 7 *******
"00001100", -- 8     **
"00001100", -- 9     **
"00001100", -- a     **
"00011110", -- b    ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x35
"00000000", -- 0
"00000000", -- 1
"11111110", -- 2 *******
"11000000", -- 3 **
"11000000", -- 4 **
"11000000", -- 5 **
"11111100", -- 6 ******
"00000110", -- 7      **
"00000110", -- 8      **
"00000110", -- 9      **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x36
"00000000", -- 0
"00000000", -- 1
"00111000", -- 2   ***
"01100000", -- 3  **
"11000000", -- 4 **
"11000000", -- 5 **
"11111100", -- 6 ******
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x37
"00000000", -- 0
"00000000", -- 1
```

```
"11111110", -- 2 *******
"11000110", -- 3 **   **
"00000110", -- 4      **
"00000110", -- 5      **
"00001100", -- 6     **
"00011000", -- 7    **
"00110000", -- 8   **
"00110000", -- 9   **
"00110000", -- a   **
"00110000", -- b   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x38
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"01111100", -- 6  *****
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x39
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"01111110", -- 6  ******
"00000110", -- 7      **
"00000110", -- 8      **
"00000110", -- 9      **
"00001100", -- a     **
"01111000", -- b  ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
```

```
-- code x3a
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00011000", -- 4    **
"00011000", -- 5    **
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00011000", -- 9    **
"00011000", -- a    **
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x3b
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00011000", -- 4    **
"00011000", -- 5    **
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00011000", -- 9    **
"00011000", -- a    **
"00110000", -- b   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x3c
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000110", -- 3      **
"00001100", -- 4     **
"00011000", -- 5    **
"00110000", -- 6   **
"01100000", -- 7  **
"00110000", -- 8   **
"00011000", -- 9    **
"00001100", -- a     **
"00000110", -- b      **
"00000000", -- c
```

```
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x3d
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111110", -- 5  ******
"00000000", -- 6
"00000000", -- 7
"01111110", -- 8  ******
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x3e
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"01100000", -- 3  **
"00110000", -- 4   **
"00011000", -- 5    **
"00001100", -- 6     **
"00000110", -- 7      **
"00001100", -- 8     **
"00011000", -- 9    **
"00110000", -- a   **
"01100000", -- b  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x3f
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"00001100", -- 5     **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00000000", -- 9
```

```
"00011000", -- a    **
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x40
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"11011110", -- 6 ** ****
"11011110", -- 7 ** ****
"11011110", -- 8 ** ****
"11011100", -- 9 ** ***
"11000000", -- a **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- A: code x41
"00000000", -- 0
"00000000", -- 1
"00010000", -- 2    *
"00111000", -- 3   ***
"01101100", -- 4  ** **
"11000110", -- 5 **   **
"11000110", -- 6 **   **
"11111110", -- 7 *******
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"11000110", -- b **   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- B: code x42
"00000000", -- 0
"00000000", -- 1
"11111100", -- 2 ******
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01111100", -- 6  *****
```

```
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"11111100", -- b ******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- C: code x43
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"01100110", -- 3  **  **
"11000010", -- 4 **    *
"11000000", -- 5 **
"11000000", -- 6 **
"11000000", -- 7 **
"11000000", -- 8 **
"11000010", -- 9 **    *
"01100110", -- a  **  **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- D: code x44
"00000000", -- 0
"00000000", -- 1
"11111000", -- 2 *****
"01101100", -- 3  ** **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01100110", -- 6  **  **
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01101100", -- a  ** **
"11111000", -- b *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x45
"00000000", -- 0
"00000000", -- 1
"11111110", -- 2 *******
"01100110", -- 3  **  **
```

```
"01100010", -- 4  **   *
"01101000", -- 5  ** *
"01111000", -- 6  ****
"01101000", -- 7  ** *
"01100000", -- 8  **
"01100010", -- 9  **   *
"01100110", -- a  **  **
"11111110", -- b *******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x46
"00000000", -- 0
"00000000", -- 1
"11111110", -- 2 *******
"01100110", -- 3  **  **
"01100010", -- 4  **   *
"01101000", -- 5  ** *
"01111000", -- 6  ****
"01101000", -- 7  ** *
"01100000", -- 8  **
"01100000", -- 9  **
"01100000", -- a  **
"11110000", -- b ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x47
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"01100110", -- 3  **  **
"11000010", -- 4 **    *
"11000000", -- 5 **
"11000000", -- 6 **
"11011110", -- 7 ** ****
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"01100110", -- a  **  **
"00111010", -- b   *** *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- H: code x48
"00000000", -- 0
```

```
"00000000", -- 1
"11000110", -- 2 **   **
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"11111110", -- 6 *******
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"11000110", -- b **   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- I: code x49
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"00011000", -- 3    **
"00011000", -- 4    **
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- J: code x4a
"00000000", -- 0
"00000000", -- 1
"00011110", -- 2    ****
"00001100", -- 3     **
"00001100", -- 4     **
"00001100", -- 5     **
"00001100", -- 6     **
"00001100", -- 7     **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01111000", -- b  ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
```

```
"00000000", -- f
-- K: code x4b
"00000000", -- 0
"00000000", -- 1
"11100110", -- 2 ***  **
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01101100", -- 5  ** **
"01111000", -- 6  ****
"01111000", -- 7  ****
"01101100", -- 8  ** **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"11100110", -- b ***  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- L: code x4c
"00000000", -- 0
"00000000", -- 1
"11110000", -- 2 ****
"01100000", -- 3  **
"01100000", -- 4  **
"01100000", -- 5  **
"01100000", -- 6  **
"01100000", -- 7  **
"01100000", -- 8  **
"01100010", -- 9  **   *
"01100110", -- a  **  **
"11111110", -- b *******
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- M: code x4d
"00000000", -- 0
"00000000", -- 1
"11000011", -- 2 **    **
"11100111", -- 3 ***  ***
"11111111", -- 4 ********
"11111111", -- 5 ********
"11011011", -- 6 ** ** **
"11000011", -- 7 **    **
"11000011", -- 8 **    **
"11000011", -- 9 **    **
"11000011", -- a **    **
"11000011", -- b **    **
```

```
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- N: code x4e
"00000000", -- 0
"00000000", -- 1
"11000110", -- 2 **   **
"11100110", -- 3 ***  **
"11110110", -- 4 **** **
"11111110", -- 5 *******
"11011110", -- 6 ** ****
"11001110", -- 7 **  ***
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"11000110", -- b **   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- O: code x4f
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- P: code x50
"00000000", -- 0
"00000000", -- 1
"11111100", -- 2 ******
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01111100", -- 6  *****
"01100000", -- 7  **
"01100000", -- 8  **
```

```
"01100000", -- 9  **
"01100000", -- a  **
"11110000", -- b ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- Q: code x510
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11010110", -- 9 ** * **
"11011110", -- a ** ****
"01111100", -- b  *****
"00001100", -- c     **
"00001110", -- d     ***
"00000000", -- e
"00000000", -- f
-- code x52
"00000000", -- 0
"00000000", -- 1
"11111100", -- 2 ******
"01100110", -- 3  **  **
"01100110", -- 4  **  **
"01100110", -- 5  **  **
"01111100", -- 6  *****
"01101100", -- 7  ** **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"11100110", -- b ***  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x53
"00000000", -- 0
"00000000", -- 1
"01111100", -- 2  *****
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"01100000", -- 5  **
```

```
"00111000", -- 6   ***
"00001100", -- 7     **
"00000110", -- 8      **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x54
"00000000", -- 0
"00000000", -- 1
"11111111", -- 2 ********
"11011011", -- 3 ** ** **
"10011001", -- 4 *  **  *
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x55
"00000000", -- 0
"00000000", -- 1
"11000110", -- 2 **   **
"11000110", -- 3 **   **
"11000110", -- 4 **   **
"11000110", -- 5 **   **
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x56
"00000000", -- 0
"00000000", -- 1
"11000011", -- 2 **    **
```

```
"11000011", -- 3 **    **
"11000011", -- 4 **    **
"11000011", -- 5 **    **
"11000011", -- 6 **    **
"11000011", -- 7 **    **
"11000011", -- 8 **    **
"01100110", -- 9  **  **
"00111100", -- a   ****
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x57
"00000000", -- 0
"00000000", -- 1
"11000011", -- 2 **    **
"11000011", -- 3 **    **
"11000011", -- 4 **    **
"11000011", -- 5 **    **
"11000011", -- 6 **    **
"11011011", -- 7 ** ** **
"11011011", -- 8 ** ** **
"11111111", -- 9 ********
"01100110", -- a  **  **
"01100110", -- b  **  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f

-- code x58
"00000000", -- 0
"00000000", -- 1
"11000011", -- 2 **    **
"11000011", -- 3 **    **
"01100110", -- 4  **  **
"00111100", -- 5   ****
"00011000", -- 6    **
"00011000", -- 7    **
"00111100", -- 8   ****
"01100110", -- 9  **  **
"11000011", -- a **    **
"11000011", -- b **    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
```

```
-- code x59
"00000000", -- 0
"00000000", -- 1
"11000011", -- 2 **    **
"11000011", -- 3 **    **
"11000011", -- 4 **    **
"01100110", -- 5  **  **
"00111100", -- 6   ****
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5a
"00000000", -- 0
"00000000", -- 1
"11111111", -- 2 ********
"11000011", -- 3 **    **
"10000110", -- 4 *    **
"00001100", -- 5     **
"00011000", -- 6    **
"00110000", -- 7   **
"01100000", -- 8  **
"11000001", -- 9 **     *
"11000011", -- a **    **
"11111111", -- b ********
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5b
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"00110000", -- 3   **
"00110000", -- 4   **
"00110000", -- 5   **
"00110000", -- 6   **
"00110000", -- 7   **
"00110000", -- 8   **
"00110000", -- 9   **
"00110000", -- a   **
"00111100", -- b   ****
"00000000", -- c
```

```
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5c
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"10000000", -- 3 *
"11000000", -- 4 **
"11100000", -- 5 ***
"01110000", -- 6  ***
"00111000", -- 7   ***
"00011100", -- 8    ***
"00001110", -- 9     ***
"00000110", -- a      **
"00000010", -- b       *
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5d
"00000000", -- 0
"00000000", -- 1
"00111100", -- 2   ****
"00001100", -- 3     **
"00001100", -- 4     **
"00001100", -- 5     **
"00001100", -- 6     **
"00001100", -- 7     **
"00001100", -- 8     **
"00001100", -- 9     **
"00001100", -- a     **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5e
"00010000", -- 0    *
"00111000", -- 1   ***
"01101100", -- 2  ** **
"11000110", -- 3 **   **
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
```

```
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x5f
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"11111111", -- d ********
"00000000", -- e
"00000000", -- f
-- code x60
"00110000", -- 0   **
"00110000", -- 1   **
"00011000", -- 2    **
"00000000", -- 3
"00000000", -- 4
"00000000", -- 5
"00000000", -- 6
"00000000", -- 7
"00000000", -- 8
"00000000", -- 9
"00000000", -- a
"00000000", -- b
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- a: code x61
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111000", -- 5  ****
"00001100", -- 6     **
```

```
"01111100", -- 7  *****
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01110110", -- b  *** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- b: code x62
"00000000", -- 0
"00000000", -- 1
"11100000", -- 2  ***
"01100000", -- 3   **
"01100000", -- 4   **
"01111000", -- 5   ****
"01101100", -- 6   ** **
"01100110", -- 7   **  **
"01100110", -- 8   **  **
"01100110", -- 9   **  **
"01100110", -- a   **  **
"01111100", -- b   *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- c: code x63
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111100", -- 5  *****
"11000110", -- 6 **   **
"11000000", -- 7 **
"11000000", -- 8 **
"11000000", -- 9 **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- d: code x64
"00000000", -- 0
"00000000", -- 1
"00011100", -- 2    ***
"00001100", -- 3     **
```

```
"00001100", -- 4     **
"00111100", -- 5   ****
"01101100", -- 6  ** **
"11001100", -- 7 **  **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01110110", -- b  *** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- e: code x65
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111100", -- 5  *****
"11000110", -- 6 **   **
"11111110", -- 7 *******
"11000000", -- 8 **
"11000000", -- 9 **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- f: code x66
"00000000", -- 0
"00000000", -- 1
"00111000", -- 2   ***
"01101100", -- 3  ** **
"01100100", -- 4  **  *
"01100000", -- 5  **
"11110000", -- 6 ****
"01100000", -- 7  **
"01100000", -- 8  **
"01100000", -- 9  **
"01100000", -- a  **
"11110000", -- b ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- g: code x67
"00000000", -- 0
```

```
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01110110", -- 5  *** **
"11001100", -- 6 **  **
"11001100", -- 7 **  **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01111100", -- b  *****
"00001100", -- c     **
"11001100", -- d **  **
"01111000", -- e  ****
"00000000", -- f
-- h: code x68
"00000000", -- 0
"00000000", -- 1
"11100000", -- 2 ***
"01100000", -- 3  **
"01100000", -- 4  **
"01101100", -- 5  ** **
"01110110", -- 6  *** **
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"11100110", -- b ***  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- i: code x69
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2    **
"00011000", -- 3    **
"00000000", -- 4
"00111000", -- 5   ***
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
```

```
"00000000", -- f
-- j: code x6a
"00000000", -- 0
"00000000", -- 1
"00000110", -- 2     **
"00000110", -- 3     **
"00000000", -- 4
"00001110", -- 5    ***
"00000110", -- 6     **
"00000110", -- 7     **
"00000110", -- 8     **
"00000110", -- 9     **
"00000110", -- a     **
"00000110", -- b     **
"01100110", -- c  **  **
"01100110", -- d  **  **
"00111100", -- e   ****
"00000000", -- f
-- k: code x6b
"00000000", -- 0
"00000000", -- 1
"11100000", -- 2 ***
"01100000", -- 3  **
"01100000", -- 4  **
"01100110", -- 5  **  **
"01101100", -- 6  ** **
"01111000", -- 7  ****
"01111000", -- 8  ****
"01101100", -- 9  ** **
"01100110", -- a  **  **
"11100110", -- b ***  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- l: code x6c
"00000000", -- 0
"00000000", -- 1
"00111000", -- 2   ***
"00011000", -- 3    **
"00011000", -- 4    **
"00011000", -- 5    **
"00011000", -- 6    **
"00011000", -- 7    **
"00011000", -- 8    **
"00011000", -- 9    **
"00011000", -- a    **
"00111100", -- b   ****
```

```
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- m: code x6d
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11100110", -- 5 ***  **
"11111111", -- 6 ********
"11011011", -- 7 ** ** **
"11011011", -- 8 ** ** **
"11011011", -- 9 ** ** **
"11011011", -- a ** ** **
"11011011", -- b ** ** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- n: code x6e
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11011100", -- 5 ** ***
"01100110", -- 6  **  **
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"01100110", -- b  **  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- o: code x6f
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111100", -- 5  *****
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"11000110", -- 8 **   **
```

```
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x70
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11011100", -- 5 ** ***
"01100110", -- 6  **  **
"01100110", -- 7  **  **
"01100110", -- 8  **  **
"01100110", -- 9  **  **
"01100110", -- a  **  **
"01111100", -- b  *****
"01100000", -- c  **
"01100000", -- d  **
"11110000", -- e ****
"00000000", -- f
-- code x71
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01110110", -- 5  *** **
"11001100", -- 6 **  **
"11001100", -- 7 **  **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01111100", -- b  *****
"00001100", -- c     **
"00001100", -- d     **
"00011110", -- e    ****
"00000000", -- f
-- code x72
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11011100", -- 5 ** ***
```

```
"01110110", -- 6  *** **
"01100110", -- 7  **  **
"01100000", -- 8  **
"01100000", -- 9  **
"01100000", -- a  **
"11110000", -- b ****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x73
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"01111100", -- 5  *****
"11000110", -- 6 **   **
"01100000", -- 7  **
"00111000", -- 8   ***
"00001100", -- 9     **
"11000110", -- a **   **
"01111100", -- b  *****
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x74
"00000000", -- 0
"00000000", -- 1
"00010000", -- 2    *
"00110000", -- 3   **
"00110000", -- 4   **
"11111100", -- 5 ******
"00110000", -- 6   **
"00110000", -- 7   **
"00110000", -- 8   **
"00110000", -- 9   **
"00110110", -- a   ** **
"00011100", -- b    ***
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x75
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
```

```
"00000000", -- 3
"00000000", -- 4
"11001100", -- 5 **  **
"11001100", -- 6 **  **
"11001100", -- 7 **  **
"11001100", -- 8 **  **
"11001100", -- 9 **  **
"11001100", -- a **  **
"01110110", -- b  *** **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x76
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11000011", -- 5 **    **
"11000011", -- 6 **    **
"11000011", -- 7 **    **
"11000011", -- 8 **    **
"01100110", -- 9  **  **
"00111100", -- a   ****
"00011000", -- b    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x77
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11000011", -- 5 **    **
"11000011", -- 6 **    **
"11000011", -- 7 **    **
"11011011", -- 8 ** ** **
"11011011", -- 9 ** ** **
"11111111", -- a ********
"01100110", -- b  **  **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x78
```

```
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11000011", -- 5 **    **
"01100110", -- 6  **  **
"00111100", -- 7   ****
"00011000", -- 8    **
"00111100", -- 9   ****
"01100110", -- a  **  **
"11000011", -- b **    **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x79
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11000110", -- 5 **   **
"11000110", -- 6 **   **
"11000110", -- 7 **   **
"11000110", -- 8 **   **
"11000110", -- 9 **   **
"11000110", -- a **   **
"01111110", -- b  ******
"00000110", -- c      **
"00001100", -- d     **
"11111000", -- e *****
"00000000", -- f
-- code x7a
"00000000", -- 0
"00000000", -- 1
"00000000", -- 2
"00000000", -- 3
"00000000", -- 4
"11111110", -- 5 *******
"11001100", -- 6 **  **
"00011000", -- 7    **
"00110000", -- 8   **
"01100000", -- 9  **
"11000110", -- a **   **
"11111110", -- b *******
"00000000", -- c
"00000000", -- d
```

```
"00000000", -- e
"00000000", -- f
-- code x7b
"00000000", -- 0
"00000000", -- 1
"00001110", -- 2    ***
"00011000", -- 3   **
"00011000", -- 4   **
"00011000", -- 5   **
"01110000", -- 6  ***
"00011000", -- 7   **
"00011000", -- 8   **
"00011000", -- 9   **
"00011000", -- a   **
"00001110", -- b    ***
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x7c
"00000000", -- 0
"00000000", -- 1
"00011000", -- 2   **
"00011000", -- 3   **
"00011000", -- 4   **
"00011000", -- 5   **
"00000000", -- 6
"00011000", -- 7   **
"00011000", -- 8   **
"00011000", -- 9   **
"00011000", -- a   **
"00011000", -- b   **
"00000000", -- c
"00000000", -- d
"00000000", -- e
"00000000", -- f
-- code x7d
"00000000", -- 0
"00000000", -- 1
"01110000", -- 2  ***
"00011000", -- 3   **
"00011000", -- 4   **
"00011000", -- 5   **
"00001110", -- 6    ***
"00011000", -- 7   **
"00011000", -- 8   **
"00011000", -- 9   **
"00011000", -- a   **
```

```
		"01110000", -- b  ***
		"00000000", -- c
		"00000000", -- d
		"00000000", -- e
		"00000000", -- f
		-- code x7e
		"00000000", -- 0
		"00000000", -- 1
		"01110110", -- 2  *** **
		"11011100", -- 3 ** ***
		"00000000", -- 4
		"00000000", -- 5
		"00000000", -- 6
		"00000000", -- 7
		"00000000", -- 8
		"00000000", -- 9
		"00000000", -- a
		"00000000", -- b
		"00000000", -- c
		"00000000", -- d
		"00000000", -- e
		"00000000", -- f
		-- code x7f
		"00000000", -- 0
		"00000000", -- 1
		"00000000", -- 2
		"00000000", -- 3
		"00010000", -- 4    *
		"00111000", -- 5   ***
		"01101100", -- 6  ** **
		"11000110", -- 7 **   **
		"11000110", -- 8 **   **
		"11000110", -- 9 **   **
		"11111110", -- a *******
		"00000000", -- b
		"00000000", -- c
		"00000000", -- d
		"00000000", -- e
		"00000000"  -- f
	);
begin

	pixelOn: process (clk)
	begin
		if rising_edge(clk) then
			-- Read from Rom
			fontRow <= ROM(addr);
		end if;
```

```
        end process;

end Behavioral;
```