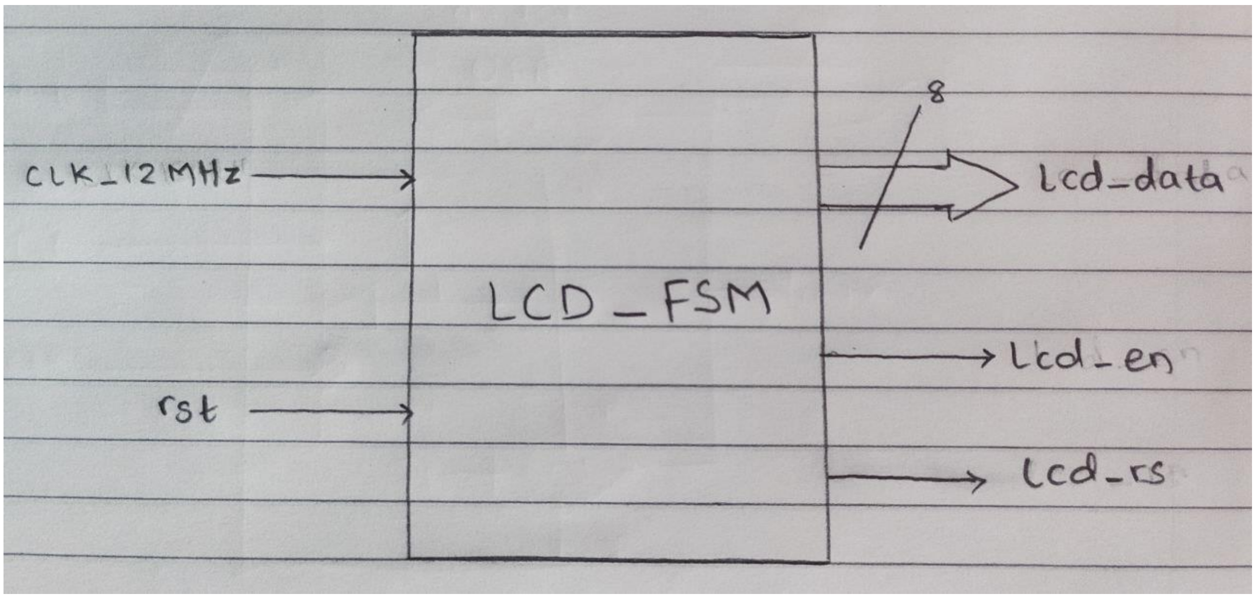
BLOCK DIAGRAM



FUNCTION TABLE

rst	clk_12MHz / 65536	lcd_data	lcd_rs	lcd_en
1	x	38h	0	x
0	↑	06h	0	↑
0	↑	0Ch	0	↑
0	↑	01h	0	↑
0	↑	50h (P)	1	↑
0	↑	49h (I)	1	↑
0	↑	43h (C)	1	↑
0	↑	54h (T)	1	↑
0	↑	20h ( )	1	↑

MAIN VHDL MODEL (MVM)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity LCD_FSM is
Port (      rst : in std_logic;                    -- reset
        clk_12Mhz : in std_logic;                  -- high freq. clock
        lcd_rs : out std_logic;                    -- LCD RS control
        lcd_en : out std_logic;                    -- LCD Enable
        lcd_data : out std_logic_vector(7 downto 0));       -- LCD Data port
end LCD_FSM;

architecture Behavioral of LCD_FSM is

signal div : std_logic_vector(15 downto 0); --- delay timer 1
signal clk_fsm,lcd_rs_s: std_logic;
-- LCD controller FSM states
type state is (reset,func,mode,cur,clear,d0,d1,d2,d3,d4,hold);
signal ps1,nx      : state;
signal dataout_s  : std_logic_vector(7 downto 0); --- internal data command multiplexer

begin

----- clk divider ---------------------------------
process(rst,clk_12Mhz)
begin
if(rst = '1')then
            div <= (others=>'0');
elsif( clk_12Mhz'event and clk_12Mhz ='1')then

            div <= div + 1;
            end if;

end process;
----------------------------------------------------
clk_fsm <= div(15);

----- Presetn state Register -----------------------
process(rst,clk_fsm)
begin
if(rst = '1')then
            ps1       <= reset;
elsif (rising_edge(clk_fsm)) then
            ps1       <= nx;

end if;
end process;
```

```
----- state and output  decoding process
process(ps1)
begin
case(ps1) is

        when reset =>
                        nx      <= func;
                        lcd_rs_s        <= '0';
                        dataout_s       <= "00111000";          -- 38h

        when func       =>
                        nx      <= mode;
                        lcd_rs_s        <= '0';
                        dataout_s       <= "00111000";          -- 38h

        when mode       =>
                        nx      <= cur;
                        lcd_rs_s        <= '0';
                        dataout_s       <= "00000110";          -- 06h

        when cur        =>
                        nx      <= clear;
                        lcd_rs_s        <= '0';
                        dataout_s       <= "00001100";          -- 0Ch  curser at starting point of
line1

        when clear=>
                        nx      <= d0;
                        lcd_rs_s        <= '0';
                        dataout_s       <= "00000001";          -- 01h

        when d0         =>
                        lcd_rs_s        <= '1';
                        dataout_s       <= "01010000";          -- P ( Decimal = 80 , HEX = 50 )
                        nx      <= d1;

        when d1         =>
                        lcd_rs_s        <= '1';
                        dataout_s       <= "01001001";          -- I ( Decimal = 73 , HEX = 49 )
                        nx      <= d2;

        when d2         =>
                        lcd_rs_s        <= '1';
                        dataout_s       <= "01000011";          -- C ( Decimal = 67 , HEX = 43 )
                        nx      <= d3;

        when d3         =>
                        lcd_rs_s        <= '1';
                        dataout_s       <= "01010100";          -- T ( Decimal = 84 , HEX = 54 )
                        nx      <= d4;
```

```
            when d4         =>
                            lcd_rs_s        <= '1';
                            dataout_s       <= "00100000";          -- space ( Decimal = 32 , HEX = 20 )
                            nx      <= hold;

            when hold       =>
                            lcd_rs_s        <= '0';
                            dataout_s       <= "00000000";          -- hold  ( Decimal = 32 , HEX = 00 ) ,
NULL
                            nx      <= hold;

when others=>
                            nx      <= reset;
                            lcd_rs_s        <= '0';
                            dataout_s       <= "00000001";          -- CLEAR ( Decimal = 1 , HEX = 01 )
end case;
end process;


lcd_en <= clk_fsm;
lcd_rs <= lcd_rs_s;
lcd_data <= dataout_s;

end Behavioral;
```

TESTBENCH VHDL MODEL (TVM)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY LCD_Test IS
END LCD_Test;

ARCHITECTURE behavior OF LCD_Test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT LCD_FSM
    PORT(
         rst : IN  std_logic;
         clk_12Mhz : IN  std_logic;
         lcd_rs : OUT  std_logic;
         lcd_en : OUT  std_logic;
         lcd_data : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal rst : std_logic := '0';
   signal clk_12Mhz : std_logic := '0';

 	--Outputs
   signal lcd_rs : std_logic;
   signal lcd_en : std_logic;
   signal lcd_data : std_logic_vector(7 downto 0);

   -- Clock period definitions
   constant clk_12Mhz_period : time := 10 ns;

BEGIN

	-- Instantiate the Unit Under Test (UUT)
   uut: LCD_FSM PORT MAP (
          rst => rst,
          clk_12Mhz => clk_12Mhz,
          lcd_rs => lcd_rs,
          lcd_en => lcd_en,
          lcd_data => lcd_data
        );

   -- Clock process definitions
   clk_12Mhz_process :process
   begin
```

```
                        clk_12Mhz <= '0';
                        wait for clk_12Mhz_period/2;
                        clk_12Mhz <= '1';
                        wait for clk_12Mhz_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      rst <= '1';
      wait for 20 ns;

     rst <= '0';
      -- insert stimulus here

      wait;
   end process;

END;
```