Synchronous Transmitter:

Table of contents:

```
1   --------------------title-------------------------------
2   -- Project Name:    Synchronous_Trans,itter
3   -- File Name:       Sync_Transmitter.vhd
4   -- Author:          Avishai Mostowicz & Reshef Finkelstein
5   -- Ver:             1
6   -- Created Date:    11/01/2018
7   ---------------------------------------------------------
8
9   library ieee;
10  use ieee.std_logic_1164.all;
11  use work.My_pack.all;
12
13  entity Sync_Transmitter is
14  port(
15        ser_data, Ser_Clk, available                : in std_logic;
16        To_7Seg_ones, To_7Seg_tens, To_7Seg_hunds : out std_logic_vector(6 downto 0);
17        To_7Seg_off                                 : out std_logic_vector(6 downto 0):=(
    others=>'1');
18        Reset, Sys_Clk                              : in std_logic;
19        SrartTr                                     : in std_logic;
20        SData                                       : out std_logic;
21        SClk                                        : Buffer std_logic :='1';
22        Not_found                                   : out std_logic);
23  end Sync_Transmitter;
24
25  architecture stractural of Sync_Transmitter is
26  ---------------Component Declaration:--------------------------
27  component CardRd_Ctrl
28  port(
29        rst, clk, data, crClk, available            : in std_logic;
30        sub_clk                                     : buffer std_logic;
31        To_7Seg_ones, To_7Seg_tens, To_7Seg_hunds : out std_logic_vector(6 downto 0);
32        To_7Seg_off                                 : out std_logic_vector(6 downto 0);
33        ready                                       : out std_logic:='0';
34        CardNum                                     : out std_logic_vector(11 downto 0)
35        );
36  end component;
37
38  component Tr_Ctrl
39  PORT(
40        Reset, Sys_Clk      : in std_logic;
41        Ready, SrartTr      : in std_logic;
42        CardNumber          : in std_logic_vector (11 DOWNTO 0);
43        SData               : out std_logic;
44        SClk                : Buffer std_logic :='1';
45        Not_found           : out std_logic
46     );
47  END component;
48
49
50  Signal S_CardNumber             :  std_logic_vector (11 downto 0);
51  signal S_Ready                  :  std_logic;
52
53  begin
54
55
56  ------------component instantiation:----------------------------
57
58  U1: CardRd_Ctrl
59  port map(
60           rst             =>    Reset,
61           clk             =>    Sys_Clk,
62           data            =>    ser_data,
63           crClk           =>    Ser_Clk,
64           available       =>    available,
65           To_7Seg_ones    =>    To_7Seg_ones,
```

```
66            To_7Seg_tens    =>    To_7Seg_tens,
67            To_7Seg_hunds   =>    To_7Seg_hunds,
68            ready           =>    S_Ready  ,
69            CardNum         =>    S_CardNumber
70            );
71
72    U2: Tr_Ctrl
73    port map(
74            Reset       =>    Reset,
75            Sys_Clk     =>    Sys_Clk,
76            Ready       =>    S_Ready,
77            SrartTr     =>    SrartTr,
78            CardNumber  =>    S_CardNumber,
79            SData       =>    SData,
80            SClk        =>    SClk,
81            Not_found   =>    Not_found
82            );
83
84    end stractural;
85
```

```
1  ----------Title--------------------------------
2  --Project Name: Synchronous_Transmitter
3  --File Name:    CardRd_Ctrl.vhd
4  --Author:       Avishai Mostowicz & Reshef Finkelstein
5  --Ver:          1
6  --Created Date: 06/01/2018
7  ----------------------------------------------
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.numeric_std.all;
12 use work.My_pack.all;
13
14 entity CardRd_Ctrl is
15 port(
16       rst, clk, data, crClk, available          : in std_logic;
17       sub_clk                                   : buffer std_logic;
18       To_7Seg_ones, To_7Seg_tens, To_7Seg_hunds : out std_logic_vector(6 downto 0);
19       To_7Seg_off                               : out std_logic_vector(6 downto 0);
20       ready                                     : out std_logic:='0';
21       CardNum                                   : out std_logic_vector(11 downto 0)
22       );
23 end CardRd_Ctrl;
24
25 architecture behave of CardRd_Ctrl is
26    type state_type is (Idle, GetData, Error, SendData);
27    signal cur_state : state_type;
28    signal invData                  : std_logic;
29    signal data_Buffer              : std_logic_vector(11 downto 0):= (others=>'0');
30    signal S_ones, S_tens, S_hunds : integer range 0 to 15;
31 begin
32 To_7Seg_off <= (others=>'1'); -- Setting the 4'th LED to Black so the 3 digit number
   will be more aesthetic
33 ------------------------Functions and procedures:------------------
34 Inv_Data(data,crClk,invData);
35 To_7Seg_ones  <= To_7Seg(S_ones);
36 To_7Seg_tens  <= To_7Seg(S_tens);
37 To_7Seg_hunds <= To_7Seg(S_hunds);
38 ---------------------------------------------------------------------
39 comb: process(available,crClk)
40 variable bitCnt                        : integer range 0 to 48; -- wait untill the data
   from the Card reader start to arive
41 variable TrashDatCnt                   : integer range 0 to 27; -- wait for the "trash"
   data from the card reader has finish
42 variable index                         : integer range 0 to 5;  -- index for the
   posotion of the inv data to be inserted to the temp buffer.
43 variable Tmpdata                       : std_logic_vector (3 downto 0); -- temp buffer -
   has the digit of the card number (ones, tens, hunds). after it gets the each digit. it
   insert it to the data buffer (the card number)
44 variable Parity                        : std_logic:='1'; -- check for parity and compare
   to the 4'th bit on each digit
45 variable DataBuff_Index                : integer  range 0 to 11; -- stores the temp
   cardnumber
46 variable finish                        : boolean:=true; -- flag set when data has been
   recived from the card reader
47 variable GotStartNibble, GotEndNibble  : boolean:=false; -- flag set when start note or
   end node has been detected
48 variable EndOfTrashData                : boolean:=false;
49 begin
50    if(available='1') then --when the card reader dosent operate. available (CLS) is '1'.
   in this setting we reset all the variables.
51       cur_state <= Idle;
52       index:=0;
53       bitCnt:=0;
54       DataBuff_Index:=0;
55       Parity:='1';
```

```
56          ready <= '0';
57          finish:=false;
58          GotEndNibble := false;
59          GotStartNibble := false;
60          TrashDatCnt:=0;
61          EndOfTrashData:=false;
62          Tmpdata:= (others=>'0');
63       elsif(crClk'event and crClk='0') then -- get the data on each falling edge of the
    card reader clock
64          case cur_state is
65    ---------------------------------------------------------------
66             when Idle =>
67             bitCnt:= bitCnt+1;
68                ready <= '0';
69                if(bitCnt>16 and finish=false) then
70                   if(index<4) then -- get the digit from the card reader
71                      Tmpdata(index):= invData;
72                      Parity:= Parity xor invData;
73                      index:= index+1;
74                   elsif(index=4 and Parity=invData) then
75                      if(Tmpdata="0010") then -- look for the start note. if it has been
    found. set flag to true and start reciving the cardnumber
76                         GotStartNibble:=true;
77                      end if;
78                      index:=0;
79                      Tmpdata:= (others=>'0');
80                      Parity:='1';
81                      if(GotStartNibble=true) then
82                         cur_state <= GetData;
83                         bitCnt:=0;
84                      else
85                         cur_state <= Idle; -- wait until the start note has arived.
86                      end if;
87                   end if;
88                else
89                   cur_state <= Idle;
90                end if;
91    ---------------------------------------------------------------
92
93             when GetData =>
94                if(GotStartNibble=true and EndOfTrashData=false) then --wait untill the
    "trash" data has passed before starting to save the card number digits.
95                   TrashDatCnt:=TrashDatCnt+1;
96                end if;
97                if(TrashDatCnt=22) then
98                   EndOfTrashData:=true;
99                end if;
100               if(Tmpdata="1111" and DataBuff_Index>8 and index=4 and Parity=invData) then
    -- after got all the card number. wait for the end note and then send the card number
    data and set the LED to the card number.
101                  GotEndNibble:=true;
102               end if;
103               if(index<4 and DataBuff_Index<=12 and EndOfTrashData=true) then -- start
    getting the digits from the card reader
104                  Tmpdata(index):= invData;
105                  Parity:= Parity xor invData;
106                  index:= index+1;
107               elsif(index=4 and Parity=invData and DataBuff_Index<=8 and EndOfTrashData=
    true) then -- if you got the digit and the Parity bit is equal to the Parity bit from
    the card reader => Save the digit.
108                  data_Buffer(11-DataBuff_Index downto 11-(DataBuff_Index+3)) <= Tmpdata;
109                  DataBuff_Index:= DataBuff_Index+4;
110                  index:=0;
111                  Tmpdata:= (others=>'0');
112                  Parity:='1';
113                  cur_state <= GetData;
```

```
114             elsif(DataBuff_Index>8 and GotEndNibble=true) then -- if you finished reading all the digit of the card number and you found the end note => start sending the card number data and light up the LED.
115                cur_state <= SendData;
116             elsif(Parity /= invData and DataBuff_Index<=8 and EndOfTrashData=true) then -- if the Parity bit is not equal to the Parity bit from the card reader then you got an error => display an error mess on the LED and wait for a new card to be read.
117                cur_state <= Error;
118             end if;
119   ------------------------------------------------------------------
120
121          when SendData => -- send the card number and light up the LED
122             S_ones  <= to_integer(unsigned(data_Buffer(3 downto 0)));
123             S_tens  <= to_integer(unsigned(data_Buffer(7 downto 4)));
124             S_hunds <= to_integer(unsigned(data_Buffer(11 downto 8)));
125             CardNum <= data_Buffer;
126             ready <= '1'; -- send a ready pulsh to the fifo with the card number
127             bitCnt:=0;
128             finish:=true;
129             cur_state <= Idle;
130   ------------------------------------------------------------------
131
132          when Error => -- if you got an error. write on the LED screen an error mess: 'Err' and reset the card number.
133             S_ones   <= 14;--number code for 'r' char
134             S_tens   <= 14;--number code for 'r' char
135             S_hunds  <= 13;--number code for 'E' char
136             CardNum  <= (others=>'0');
137             bitCnt:=0;
138             finish:=true;
139             cur_state <= Idle;
140   ------------------------------------------------------------------
141
142          when others => null;
143       end case;
144    end if;
145 end process;
146
147
148 ClkCreator: process(clk,rst) -- create a clock for the signal tap and for the card reader. it is necessary to use this clock to be able to read the data from the card reader beacuse the system clock (27MHz) is too fast.
149 variable cnt : integer range 0 to 1000;
150 begin
151    if(rst='1') then
152       cnt:=0;
153       sub_clk <= '1';
154    elsif(rising_edge(clk)) then
155       cnt:= cnt+1;
156       if(cnt=1000) then
157          sub_clk <= not sub_clk;
158          cnt:=0;
159       end if;
160    end if;
161 end process;
162 end behave;
```

```
1   --------------------title-------------------------------
2   -- Project Name:    Synchronous_Trans,itter
3   -- File Name:       Tr_Ctrl.vhd
4   -- Author:          Avishai Mostowicz & Reshef Finkelstein
5   -- Ver:             1
6   -- Created Date:    11/01/2018
7   ----------------------------------------------------------
8
9   library ieee;
10  use ieee.std_logic_1164.all;
11  use ieee.Numeric_std.all;
12  use work.My_pack.all;
13
14
15  entity Tr_Ctrl is
16  port(
17        Reset, Sys_Clk      : in std_logic;
18        Ready, SrartTr      : in std_logic;
19        CardNumber          : in std_logic_vector (11 DOWNTO 0);
20        SData               : out std_logic;
21        SClk                : Buffer std_logic :='1';
22        Not_found           : out std_logic
23     );
24  end Tr_Ctrl;
25
26
27
28
29  architecture behave of Tr_Ctrl is
30
31  component  ROM_MEM
32     PORT
33     (
34        address     : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
35        clock       : IN STD_LOGIC  := '1';
36        q           : OUT STD_LOGIC_VECTOR (11 DOWNTO 0)
37     );
38  end component;
39
40  type state_type is (Idle,Recive_MemCheck,DataPack,Change,SendPack_CheckSum);
41  signal cur_state                  : state_type;
42  signal SrartTr_Strobe             : std_logic:= '0';    --strobe for StartTr
43  signal SClk_Strobe                : std_logic:= '0';    --strobe for StartTr
44  signal MemAdd                     : std_logic_vector (6 DOWNTO 0):=(others=>'0');
    --signal that connects to the memory address
45  constant StartCondition           : std_logic_vector (11 DOWNTO 0):=(others=>'0');
46  constant EndCondition             : std_logic_vector (11 DOWNTO 0):=(others=>'1');
47  signal q_out                      : std_logic_vector (11 DOWNTO 0):=(others=>'0');
    --signal that connects to the memory out
48  Signal DataBuff                   : std_logic_vector (3*12-1 DOWNTO 0):=(others=>'0');
    --contain the 3 data packges (if N=2, in send mode send only 2 out 3)
49  signal Data1                      : std_logic_vector (11 DOWNTO 0):=(others=>'0');
50  signal Data2                      : std_logic_vector (11 DOWNTO 0):=(others=>'0');
51  signal Data3                      : std_logic_vector (11 DOWNTO 0):=(others=>'0');
52  signal Sample_in, not_sample_in   : std_logic;  -- for strobe
53  signal Sample_in1, not_sample_in1 : std_logic;  -- for strobe
54  signal SClk_On                    : Boolean := True;
55  signal CLK_counter                : integer:=0;
56
57
58  begin
59  -------------Functions Calls:-----------------------------------
60  Strobe(SrartTr,Reset,Sys_Clk,Sample_in,not_sample_in,SrartTr_Strobe);  -- For start
    transmmit
61  Strobe(SClk,Reset,Sys_Clk,Sample_in1,not_sample_in1,SClk_Strobe);      -- For SClk
62
```

```
63  ------------component instantiation:----------------------------
64
65  U1: ROM_MEM
66  port map(
67           address => MemAdd,
68           clock => Sys_Clk,
69           q      => q_out
70           );
71
72
73  State_Machine: process (Reset,Sys_Clk,Ready,SrartTr_Strobe,SClk_Strobe)
74        variable CardNum        : std_logic_vector (11 DOWNTO 0);     -- variable that
    recives the card number
75        variable Counter        : integer range 0 to 128;             -- counts the memory
    words
76        variable N              : integer;
77        variable Found          : Boolean := False;
78        Variable Data_F         : integer range 0 to 3;
79        variable Data_Finish    : Boolean :=False;
80        variable ADD_F          : integer range 0 to 128;
81        variable BIT_Send       : integer range 0 to 100;
82        variable START_CON_Send : integer range 0 to 12;
83        variable END_CON_Send   : integer range 0 to 12;
84        variable Check_CON_Send : integer range 0 to 12;
85        variable CheckSum       : integer range 0 to 100 :=0;
86        variable CheckSum_VEC   : std_logic_vector (11 DOWNTO 0);
87
88
89     begin
90
91
92           if(Reset='1') then
93              SData      <='1';
94              Not_found  <='0';
95              N          :=0;
96              counter    :=0;
97              CheckSum   :=0;
98              CardNum :=(others=>'0');
99              SClk_On <= FALSE;
100             cur_state<=Idle;
101
102          elsif Sys_Clk'event and Sys_Clk='1' then
103             case cur_state is
104
105                when Idle =>
106                   SData     <='1';
107                   Not_found <='0';
108                   CheckSum  :=0;
109                   N         :=0;
110                   counter   :=0;
111                   Data_F :=0;
112                   SClk_On <= FALSE;
113                   CardNum   :=(others=>'0');
114
115                   if(Ready='1') then             --recives Ready ='1' and moves to
    Recive_MemCheck;
116                      cur_state <=Recive_MemCheck;
117                   else
118                      cur_state<=Idle;
119                   end if;
120
121
122                when Recive_MemCheck =>
123                   CardNum := CardNumber;
124                   if(Counter>128) then          --Max Num of words in memory
125                      Not_found <='1';           --Turn on not found led
```

```
126                     cur_state<=Idle;
127                  elsif(Counter<=128) then
128                     MemAdd <= STD_LOGIC_VECTOR(to_unsigned(counter,7));     -- Card Number
129                     ADD_F := ADD_F+1;
130                     if(CardNum = q_out and ADD_F rem 2 = 0) then            --rem2=0 for
    delay, enter only when equal
131                        Not_found <='0';
132                        Found:=True;
133                        MemAdd <= STD_LOGIC_VECTOR(to_unsigned(counter+1,7));  --N, Num
    of data packges
134                        cur_state<=Change;
135                     elsif (CardNum /= q_out and ADD_F rem 2=0) then
136                        counter := counter+1;
137                        cur_state <=Recive_MemCheck;
138                     end if;
139                  end if;
140
141
142               when Change =>
143                  if(Found = TRUE) then                  --To save N to variable
144                     N := to_integer(unsigned(q_out));
145                     Found := False;
146                     cur_state <=DataPack;
147                  end if;
148
149                  if(Data_F =0) then                           --ping pong from DataPack
    to change address
150                     MemAdd <= STD_LOGIC_VECTOR(to_unsigned(counter+2,7));
151                  elsif(Data_F =1) then
152                     MemAdd <= STD_LOGIC_VECTOR(to_unsigned(counter+3,7));
153                  elsif(Data_F =2) then
154                     MemAdd <= STD_LOGIC_VECTOR(to_unsigned(counter+4,7));
155                  else
156                     Data_F :=0;
157                  end if;
158                  cur_state <=DataPack;
159
160
161
162
163
164
165               when DataPack =>
166
167                  if(N/=2 and N/=3) then                  --Fill data from memory
168                     cur_state<=Idle;
169                     elsif(Data_Finish = False) then
170                        cur_state<=Change;
171                        if(Data_F =0) then
172                           Data1 <= q_out;
173                           Data_F:=Data_F+1;
174                        elsif (Data_F =1) then
175                           Data2 <= q_out;
176                           Data_F:=Data_F+1;
177                        elsif (Data_F =2) then
178                           Data3 <= q_out;
179                           Data_F:=Data_F+1;
180                        elsif (Data_F =3) then
181                           Data_F :=0;
182                           Data_Finish := True;
183                           if(N=3) then
184                              DataBuff <= Data1&Data2&Data3;  --Normal send, 3 packages
    of data
185                           else
186                              DataBuff <= Data3&Data1&Data2;  --2 packeges of data, flip
    Beacuse MSB first
```

```
187                             end if;
188                          end if;
189                       end if;
190
191                       if(SrartTr_Strobe = '1') then
192                          Data_Finish := False;
193                          START_CON_Send:=12;
194                          END_CON_Send:=12;
195                          BIT_Send:=N*12;
196                          Check_CON_Send:=12;
197                          cur_state<=SendPack_CheckSum;
198                       elsif(Data_F =3) then
199                          cur_state<=DataPack;
200                       end if;
201
202
203                   when SendPack_CheckSum =>
204                       if(Ready='1') then
205                          counter := 0;
206                          cur_state <=Recive_MemCheck;
207                       end if;
208
209                       SClk_On <= TRUE;
210                       if(SCLk_Strobe='1') then
211                          if(START_CON_Send>0) then                          --Send Start
    Condition
212                             SData <= StartCondition(START_CON_Send-1);
213                             START_CON_Send:=START_CON_Send-1;
214                          end if;
215
216                          if(BIT_Send>0 and START_CON_Send=0) then           -- Send Data Pack
217                             SData <= DataBuff(BIT_Send-1);
218
219                             if(DataBuff(BIT_Send-1)='1') then              -- Calculate
    checkSum
220                                CheckSum :=CheckSum +1;
221                             end if;
222
223                             BIT_Send:=BIT_Send-1;
224                          end if;
225
226                          if(END_CON_Send>0 and BIT_Send=0) then             --Send End Condition
227                             SData <= ENDCondition(END_CON_Send-1);
228                             END_CON_Send:=END_CON_Send-1;
229                          end if;
230
231                          if(START_CON_Send=0 and BIT_Send=0 and END_CON_Send=0) then
    --Send CheckSum
232                             CheckSum_VEC := STD_LOGIC_VECTOR(to_unsigned(CheckSum,12));
233                             SData <= CheckSum_VEC(Check_CON_Send-1);
234                             Check_CON_Send:=Check_CON_Send-1;
235                          end if;
236
237                          if(START_CON_Send=0 and BIT_Send=0 and END_CON_Send=0 and
    Check_CON_Send=0) then
238                             cur_state<=IDLE;
239                          else
240                             cur_state<=SendPack_CheckSum;
241                       end if;
242                   else
243                       cur_state<=SendPack_CheckSum;
244                   end if;
245
246
247                   when others => null;
248
```

```
249                 end case;
250              end if;
251        end process;
252
253
254
255     SClk_PRC: process(Sys_Clk,SClk_On)              -- ~100MHz Freaquency
256     begin
257        if(SClk_On = FALSE) then
258           SClk <= '1';
259           CLK_counter <=0;
260        elsif(Sys_Clk'event and Sys_Clk='0' and SClk_On = TRUE) then
261              CLK_counter  <= CLK_counter +1;
262              if(CLK_counter = 7) then
263                 SClk <= NOT SClk;
264                 CLK_counter <=0;
265              end if;
266        end if;
267     end process;
268
269
270     end behave;
271
272
273
```

```
1   ----------------------title-------------------------------
2   -- Project Name:    Synchronous_Trans,itter
3   -- File Name:       My_pack.vhd
4   -- Author:          Avishai Mostowicz & Reshef Finkelstein
5   -- Ver:             1
6   -- Created Date:    11/01/2018
7   ----------------------------------------------------------
8
9   ------------------------ My_pack.vhd program ----------------------------------
10  LIBRARY IEEE;
11  USE ieee.std_logic_1164.ALL;
12  use ieee.Numeric_std.all;
13
14  PACKAGE My_pack IS
15        type stdlogic_to_char_t is array(std_logic) of character;
16           constant to_char : stdlogic_to_char_t := (
17                 'U' => 'U',
18                 'X' => 'X',
19                 '0' => '0',
20                 '1' => '1',
21                 'Z' => 'Z',
22                 'W' => 'W',
23                 'L' => 'L',
24                 'H' => 'H',
25                 '-' => '-');
26  ------------------------------------------------------------------------------------

27     FUNCTION To_String (data : std_logic_vector)  return string;
28  ------------------------------------------------------------------------------------
29     PROCEDURE generate_clock ( SIGNAL run: IN std_logic; SIGNAL clk: OUT std_logic;
30                                SIGNAL T_period,T_pulse,T_phase: IN time);
31  ------------------------------------------------------------------------------------

32     PROCEDURE Strobe (SrartTr: IN std_logic; Reset: IN std_logic;
33                       SIGNAL Sys_Clk: IN std_logic;
34                       SIGNAL Sample_in, not_sample_in : inout std_logic;
35                       SIGNAL SrartTr_Strobe: OUT std_logic);
36  ------------------------------------------------------------------------------------

37     FUNCTION To_7Seg ( data:integer range 0 to 14)RETURN std_logic_vector;
38  ------------------------------------------------------------------------------------
39     procedure Inv_Data (data_in : in std_logic;
40           signal cr_clk         : in std_logic;
41           signal data_out       : out std_logic
42                       );
43  ------------------------------------------------------------------------------------
44  END My_pack;
45  ------------------------------------------------------------------------------------
    ----------------------------------
46  PACKAGE BODY My_pack IS
47  ------------------ To_7Seg function -----------------------------------------------
48  FUNCTION To_7Seg ( data:integer range 0 to 14)RETURN std_logic_vector IS
49  VARIABLE temp:std_logic_vector (6 downto 0):=(others=>'1');
50  BEGIN
51     CASE   data  IS
52        WHEN  0  => temp :="1000000";  -- 40h
53        WHEN  1  => temp :="1111001";  -- 79h
54        WHEN  2  => temp :="0100100";  -- 24h
55        WHEN  3  => temp :="0110000";  -- 30h
56        WHEN  4  => temp :="0011001";  -- 19h
57        WHEN  5  => temp :="0010010";  -- 12h
58        WHEN  6  => temp :="0000010";  -- 02h
59        WHEN  7  => temp :="1111000";  -- 78h
60        WHEN  8  => temp :="0000000";  -- 00h
61        WHEN  9  => temp :="0010000";  -- 10h
62        WHEN  13 => temp :="0000110";  -- 'E' char
```

```
63          WHEN  14 => temp :="0101111";  -- 'r' char
64          WHEN OTHERS => NULL;
65       END CASE;
66       RETURN (temp);
67    END To_7Seg;
68    ---------------------------------------------------------------------------------
69
70    ------------------ Inv Data Procedure ------------------------------------------
71    procedure Inv_Data (data_in : in std_logic;
72             signal cr_clk      : in std_logic;
73             signal data_out    : out std_logic) is
74    begin
75       if(cr_clk'event and cr_clk='0') then
76          data_out <= not data_in;
77       end if;
78    end Inv_Data;
79    ---------------------------------------------------------------------------------
80
81    ------------------ Strobe Procedure --------------------------------------------
82    PROCEDURE Strobe (SrartTr: IN std_logic; Reset: IN std_logic;
83             SIGNAL Sys_Clk: IN std_logic;
84             SIGNAL Sample_in, not_sample_in : inout std_logic;
85             SIGNAL SrartTr_Strobe: OUT std_logic )  IS
86
87    BEGIN
88       if Reset='1' then
89          SrartTr_Strobe <='0';
90       elsif rising_edge(Sys_Clk) then
91          Sample_in<= NOT SrartTr;
92          not_sample_in<= not Sample_in;
93          SrartTr_Strobe <= not_sample_in and Sample_in;
94       end if;
95    END Strobe;
96    ---------------------------------------------------------------------------------
97
98    ----------- To_string Function Body : Convert a std_logic_vector to a string ------
99
100      FUNCTION to_string ( data:std_logic_vector)RETURN string IS
101         VARIABLE tmp : std_logic_vector(1 to data'length) :=(others=>'0');
102         VARIABLE result : string(tmp'range);
103      BEGIN
104         tmp:=data;
105         FOR i IN tmp'range LOOP
106               result(i) := to_char(tmp(i));
107         END LOOP;
108         RETURN result;
109      END to_string;
110   ---------------------------------------------------------------------------------
111
112   -------------------------------generate_clock procedure----------------------------
113      PROCEDURE generate_clock (SIGNAL run: IN std_logic; SIGNAL clk: OUT std_logic;
114                                SIGNAL  T_period,T_pulse,T_phase: IN time)  IS
115         BEGIN
116         WAIT UNTIL run='1';
117         WAIT FOR T_phase;
118                  WHILE run='1' LOOP
119            Clk<='1','0' AFTER T_pulse;
120            WAIT FOR T_period;
121                 END LOOP;
122         WAIT;
123           END generate_clock;
124   ---------------------------------------------------------------------------------
125   END My_pack;
126
127
```

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	000	000	000	254	002	456	235	262	.....
8	003	654	543	321	258	002	321	123	.....
16	000	000	000	000	000	000	000	000	.....
24	000	000	000	000	000	000	000	000	.....
32	000	000	000	000	000	000	000	000	.....
40	000	000	000	000	000	000	000	000	.....
48	000	000	000	000	000	000	000	000	.....
56	000	000	000	000	000	000	000	000	.....
64	000	000	000	000	000	000	000	000	.....
72	000	000	000	000	000	000	000	000	.....
80	000	000	000	000	000	000	000	000	.....
88	000	000	000	000	000	000	000	000	.....
96	000	000	000	000	000	000	000	000	.....
104	000	000	000	000	000	000	000	000	.....
112	000	000	000	000	000	000	000	000	.....
120	000	000	000	000	000	000	000	000	.....