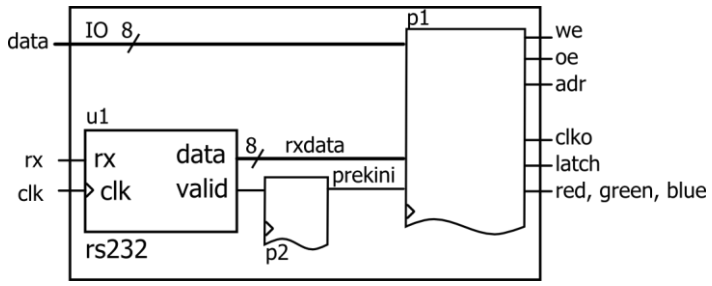
7. vaja: krmiljenje kocke LED

Končno vezje za krmiljenje kocke bo sestavljeno iz krmilnika LED in sprejemnika RS232 signalov v programirljivem vezju in zunanjega pomnilnika RAM. Sprejemnik bomo dodali v vezje kot komponento, ki jo deklariramo v arhitekturnem delu krmilnika.

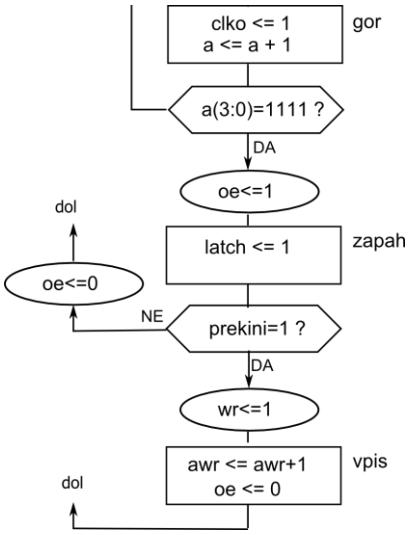
```
component RS232 is  -- deklaracija komponente
  Port ( clk : in  STD_LOGIC;
         rx : in  STD_LOGIC;
         data : out  STD_LOGIC_VECTOR(7 downto 0);
         valid: out STD_LOGIC);
end component;
begin
-- dodaj RS232 komponento
u1: RS232 port map (clk=>clk, rx=>rx, data=>rxdata, valid=>valid);
```



V vezju dodaj sinhroni proces za nastavljanje prekinitvenega signala. Signal prekini naj se postavi na '1' ob impulzu signala valid, in na '0' ko bo sekvenčni avtomat v stanju *vpis*. To naj bo novo stanje v obstoječem procesu, kjer bomo dodali logiko za vpisovanje sprejetega podatka v pomnilnik.

7.1 Vpis podatkov v pomnilnik

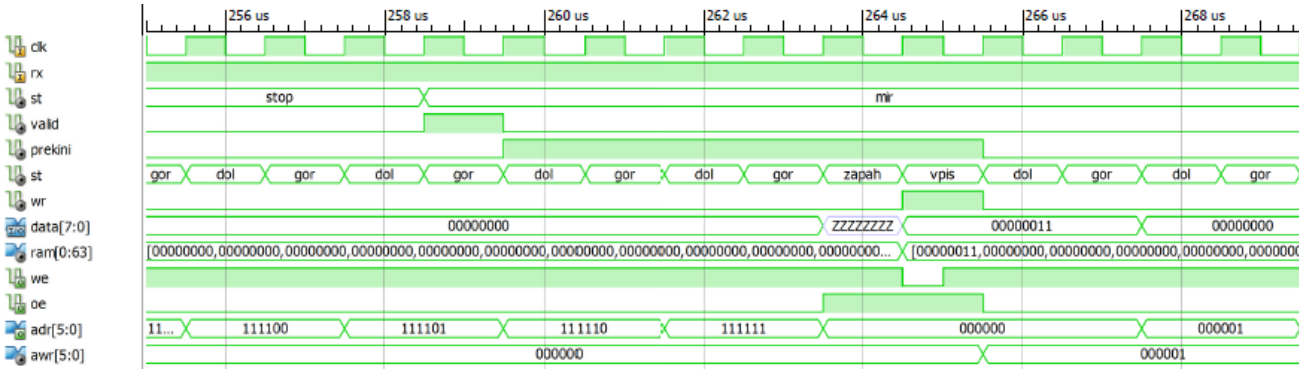
V sekvenčno vezje bomo dodali stanje *vpis* v katerega preidemo iz stanja *zapah* v primeru prekinitve. Ob prehodu v stanje zapah onemogočimo branje iz pomnilnika (oe<=1). Če ni prekinitve branje ponovno omogočimo, sicer pa aktiviramo signal za vpis (wr<=1), ki je v vseh ostalih primerih neaktiven.



V stanju vpis povečamo naslovni števec za pisanje in se vrnemo v stanje *dol.*

Na dvosmerno podatkovno vodilo pomnilnika prenesemo podatek iz RS232 sprejemnika, ko smo v stanu *vpis*, ob aktivnem signalu wr='1' pa sprožimo signal za vpis (we) in preklopimo pomnilniški naslov na naslovni števec za pisanje:

```
data <= rxdata when st=vpis else "ZZZZZZZZ";
we <= '0' when wr='1' and clk='1' else '1';
adr <= std_logic_vector(awr) when wr='1' else std_logic_vector(a);
```



7.1 Testna struktura

Simulacijo vezja bomo izvedli s testno strukturo, ki vsebuje model asinhronega pomnilnika in stavke za opis RS232 podatkovnega prenosa. Med deklaracijami signalov določimo periodo ure in bitno hitrost prenosa (br):

```
-- Clock period definitions
constant clk_period : time := 1 us;
constant br : time := 26 us;
```

Deklarirajmo še nov podatkovni tip in signal za model pomnilnika ter dodajmo opis pomnilnika:

```
type pomnilnik is array (0 to 63) of std_logic_vector(7 downto 0);
signal ram: pomnilnik := ( others => "00000000" );

BEGIN
        --  preprost model asinhronega pomnilnika
        data <= ram(to_integer(unsigned(adr))) when oe='0' else "ZZZZZZZZ";
        ram(to_integer(unsigned(adr))) <= data when we='0';
```

V procesu s stimulatorji opišimo zaporedje prenosa vrednosti "00000011" preko RS232:

```
-- Stimulus process
stim_proc: process
begin
  rx <= '1';
  wait for 10 us;
  rx <= '0'; wait for br; -- start bit
  rx <= '1'; wait for br; -- LSB
  rx <= '1'; wait for br;
  rx <= '0'; wait for br;
  rx <= '0'; wait for br;
  rx <= '0'; wait for br;
  rx <= '0'; wait for br;
  rx <= '0'; wait for br;
  rx <= '0'; wait for br; -- MSB
  rx <= '1'; wait for br; -- stop bit
 …
 wait;
```