

# Documentație PWMGEN

Brăduț-Mihai Iordache(Lawenda700)

Eduard Simion(eddy-24)

Mario Timoc(MarioTimoc21)

December 14, 2025

*Brădut-Mihai Iordache(Lawenda700), Eduard Simion(eddy-24), Mario Timoc(MarioTimoc21)2*

## Cuprins

<b>1 SPI BRIDGE</b>	<b>3</b>
<b>2 INSTRUCTION DECODER</b>	<b>4</b>
<b>3 REGISTERS</b>	<b>5</b>
<b>4 COUNTER</b>	<b>6</b>
<b>5 PWM GENERATOR</b>	<b>7</b>
<b>6 Modificări top</b>	<b>8</b>

## 1 SPI BRIDGE

Modulul SPI bridge folosește 2 blocuri `always`, unul pentru frontul crescător al ceasului și celălalt pentru frontul descrescător. Ambele verifică activarea funcției de reset și, în caz că aceasta este activată, resetează valorile output-urilor la valorile default + valoarea celor 2 countere interne. Pe frontul crescător (blocul `always posedge`) avem citirea în MOSI. Se citește de la MSB la LSB valoarea din `data_out` cu ajutorul unui counter care este decrementat de la 7 până la 0, iar odată ce se citește valoarea de la bitul 0 al `data_out`, counter-ul se resetează.

În mod similar, pe frontul descrescător (blocul `always negedge`) se scrie în registrul intermedian `data_int` de la MSB la LSB valorile primite din MISO, cu ajutorul unui counter intern care funcționează identic cu cel utilizat pentru citirea în MOSI pe posedge. Atunci când counter-ul pentru MISO atinge valoarea 0 și se resetează, se trimit și informația din `data_int` în `data_in` și se setează valoarea lui `byte_sync` la 1 (ea fiind default 0) pentru a semnala trimitera unui nou `data_in` către instruction decoder.

Citirea și scrierea se realizează doar atunci când chip select-ul `cs_n` este setat (are valoarea 0). Când nu este setat, citirea și scrierea nu se realizează, iar valoarea counterelor interne este în permanență ținută la 7 (poziția MSB).

## 2 INSTRUCTION DECODER

Instruction Decoder-ul funcționează asemenea unui AFD cu 3 stări. Trecerea dintr-o stare în alta se realizează pe frontul crescător al ceasului, iar în caz de reset se revine în starea S0. Stările automatului:

S0: Se verifică primirea semnalului `byte_sync` care semnalează primirea datelor de input. Se scrie în `data_write` pe poziția 0 valoarea bitului de pe poziția 6 din `data_write` (ne ajută la scrierea în MSB/LSB din registri), se preia valoarea bitului de pe poziția 7 și, în caz că este 1, `write` este setat ca 1 și se trece în starea S1 a automatului, iar în caz că este 0, `read` este setat ca 1 și se trece în starea S2 a automatului. Totodată, biții [5:0] din `data_in` sunt scriși în `addr`. În caz că `byte_sync` este 0, nu se întâmplă nimic.

S1: Se verifică primirea semnalului `byte_sync` care semnalează primirea datelor de input. Datele din `data_in` sunt transferate în totalitate în `data_write` și se trece în starea S0 a automatului.

În caz că `byte_sync` este 0, nu se întâmplă nimic.

S2: Se trec datele din `data_out` în `data_read` și se trece în starea S0 a automatului.

### 3 REGISTERS

Se stochează adresele regisztrelor pe 6 biți în parametrii locali ai modulului. Se declară un regisztr **sb** care va indica dacă scrierea/citirea se va face în MSB sau în LSB pentru regisztrele care sunt pe 16 biți. Funcționarea modulului va avea forma unui AFD cu 3 stări (S3, S4, S5). La activarea funcției de reset toți regisztrii sunt trecuți în valoarea default și se revine la starea S3. Trecerea dintr-o stare în alta se realizează pe frontul crescător al ceasului. Stările automatului:

S3: **sb** ia valoarea bitului 0 din **data\_write**. Dacă **write** este 1 se trece la starea S4, alternativ se trece la S5.

S4: **counter\_reset** revine la valoarea 0 (acesta trebuie, conform documentației, să fie 1 doar un singur ciclu de ceas după ce este setat). Se verifică, cu un bloc **case**, adresa la care trebuie scrisă informația din **data\_write** și implicit regisztrul. Dacă regisztrul este pe 16 biți, se verifică **sb** pentru a vedea dacă **data\_write** se scrie în MSB sau LSB. Se trece în starea S3.

S5: Se verifică, cu un bloc **case**, adresa la care se află regisztrul care trebuie citit în **data\_read** (dacă adresa nu este una în care se află unul dintre regiszre, se citește valoarea 0). Dacă regisztrul este pe 16 biți, se verifică **sb** pentru a vedea dacă **data\_read** se citește din MSB sau LSB. Se trece în starea S3.

## 4 COUNTER

Se creează un counter intern și un *period* intern (numit **ss**), care este un **wire** setat ca 1 pe 16 biți, shiftat la stânga cu **prescale** număr de biți. Se declară un bloc **always** activ pe frontul crescător al ceasului, care, la activarea funcției de reset, resetează **count\_val** la valoarea default (0 pentru **upnotdown** = 1 și **period** pentru **upnotdown** = 0) și valoarea counter-ului intern la 0. De menționat că registrul care dă valoarea lui **count\_val** este cu 1 mai mare decât valoarea lui **count\_val**, pentru a ține cont de faptul că numărătoarea începe de la 0. Numărătoarea începe atunci când **en** are valoarea 1.

Când **upnotdown** este 1, numărătoarea este crescătoare de la 0 la **period** - 1, iar când este 0, este descrescătoare de la **period** - 1 la 0.

Incrementarea/Decrementarea lui **count\_value** se realizează atunci când counter-ul intern ajunge la valoarea **ss** - 1 (deoarece începe de la 0). Când se incrementează/decrementează **count\_value**, counter-ul intern se resetează și el la 0. Când **count\_value** ajunge la valoarea maximă/minimă și trebuie incrementat/decrementat conform counter-ului intern, el de fapt se întoarce la valoarea minimă/maximă.

## 5 PWM GENERATOR

Se utilizează un bloc `always` activ pe frontul crescător al ceasului. La primirea semnalului de reset, valoarea lui `pwm_out` se resetează la 0.

Când `pwm_en` este 0, `pwm_out` este și el 0. Când `pwm_en` este 1, se verifică în `functions` tipul de aliniere al semnalului, valoarea `count_val` din momentul respectiv al ceasului și valoarea/valorile cu care aceasta din urmă trebuie comparată. În funcție de aceste condiții se determină semnalul `pwm_out` la momentul dat.

## 6 Modificări top

Fișierul top.v a fost modificat astfel încât la apelul modulului spi\_bridge.v să se utilizeze wire-rurile byte\_sync,data\_in și data\_out, iar în apelul modulului instr\_dcd intrarea byte\_sync primește valoarea wire-ului byte\_sync din top.