SISTEMA LOGIKO PROGRAMAGARRIAK

PBL LANA

Ingeniaritza informatikoa



Egilea: Mikel Murua

Urtarrillak 2, 2021

Contenido

[BLOKE DIAGRAMA 3](#_Toc60442708)

[KONPONENTEAK 3](#_Toc60442709)

[FSTM 3](#_Toc60442710)

[F\_ZAT 3](#_Toc60442711)

[TEMP 3](#_Toc60442712)

[ADDER\_RESTA 4](#_Toc60442713)

[BLINK 4](#_Toc60442714)

[DECO\_BIN 4](#_Toc60442715)

[ZAZPI\_SEG4 4](#_Toc60442716)

[UART 6](#_Toc60442717)

[UART\_TX 6](#_Toc60442718)

[*baud\_rate\_clk\_generator* 6](#_Toc60442719)

[Uart\_RX 8](#_Toc60442720)

[Deco\_out 9](#_Toc60442721)

[Button\_controler 9](#_Toc60442722)

[Mux1 9](#_Toc60442723)

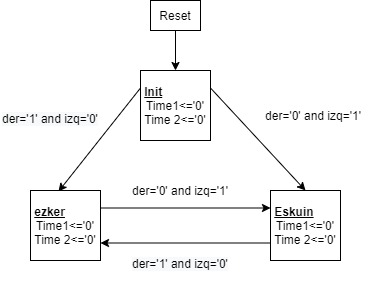
[Selector 9](#_Toc60442724)

[BIBLIOGRAFIA 10](#_Toc60442725)

# BLOKE DIAGRAMA

# KONPONENTEAK

## FSTM



Proiektu honetako egoera makina da , bertan erlojuaren 3 egoerak dauda, init esker eta eskuin, initen erlojua geldirik egongo da eskumako edo ezkerreko botoiak pultzatzen diren harte, gero botoiak behin sakatuta dagokion erlojua aktibatuko da, eta gero bestea sakatzean , aktibatuta zegoena gelditu eta bestea hasiko da kentzen.

## F\_ZAT

Frekuentzia zatitzaile normala, bertan plakaren osziladorearen 100.000.000 tik kontatzen dira 1s ren baliokidea den seinale constante bat bidaltzeko kanpora. Hau oso erabilgarria zaigu adibidez , kenketa egiterako orduan segunduro 1 kentzeko erlojuari, adder \_resta konponenentean, edo blink constante bat egiteko.

Kalkuluak:

## TEMP

Beste frekuentzia zatitzaile bat, hau bestearen antzela plakaren osziladorearen frekuentziaren tik ak kontatzen ditu, baina besteak ez bezala, honen seinale konstantea, frekuentzia jakin batera dago, zazpi segmentuko dekodetzaileko zenbakiaren 4 digituak batera ikus daitezen. Anodoaren aldaketa frekuentzia jakin honetara egitean 4 digitu oso ikusten baitira, inungoko distira arrarorik gabe.

Frekuentzia hori 6.250.000 ekoa da.

## ADDER\_RESTA

Konponentearen izenak dionez moduan, honek 2 erlojuen balioen kenketaz arduratzen den elementua da, val1 eta val2 balioen bitartez kontrolatzen dugu kenketa.

* Val1 -> eskerreko erlojua kontrolatzen du (1-aktibatuta 0- itzalita)
* Val2 -> eskumako erlojua kontrolatzen du (1-aktibatuta 0- itzalita)

Horretaz gain konponente honek ere beste 2 funtzio garrantzitsu ere baditu:

* Erloju bietatik batek 0 baliora iristen denean, erlojuen kenketa blokeatu egiten du. Reset eginez soilik desblokeatuz. Gero hasierako balioak hartzen ditu berriro.
* Beste funtzioa , ze erlojuk amaitu duen abisatzea da, horretarako fin seinalea erabiltzen dugu.
  + fin=”10” (ezkerrekoak amaitu)
  + fin=”01” (eskumakoak amaitu).

## BLINK

Izenak dioen moduan, irteerako led-ak dirdira egitarazten duen konponentea da, adder\_resta konponentearen fin seinaleak aktibatzen du, hórrela amaitu duen erlojuaren led-ak dir-dir egin dezan. Pasatzen zaion erlojua f\_zat konponenentearen 1s -ko seinalea da.

## DECO\_BIN

Konponente honen helburua, erlojuen balioa hartzea eta zazpi\_segmentuko dekodetzailearentzako seinale bihurtzea da. Hemen erlojuak eduki ditzakeen 100 egoerak biltzen dira. 0 tik 99 ko zenbakiak alegia. 2 deco bin erabiliko dira , 1 erloju bakoitzarentzat.

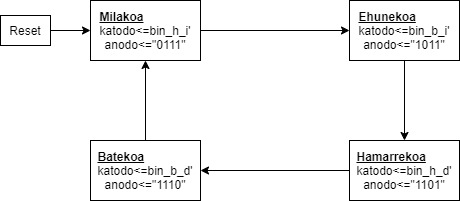
|  |  |  |
| --- | --- | --- |
| **Erloju balioa** | **Hamartarra** | **Batekoa** |
| 74 | 7 | 4 |

Kontuan hartuz, 7 segmentuko katodo bakoitzak digitu 1 erakutsi dezakeela bakarrik, deco bin honetan zenbakia, bere 2 digituetan banatzen da. Atal hamartarreko digituak eta bateko ataleko digituak.

Zenbaki hamartarraren balioa amartarra irteeratik kanporatzen da zazpi\_segmentuko dekodetzaileruntz, eta zenbaki bakoitia batekoa irteeratik kanporatzen da, leku berdinerantz. Horrela zenbaki guztia bidali dezakegu 7 segmenuko displaierantz.

## ZAZPI\_SEG4

Zazpi segmentuko dekodetzaile bat da, baina pixka bat moldatuta 4 anodoak batera ikusi daitezen. 4 zenbakiak batera ikusteko hurrengoko egoera makina erabiltzen da, egoera harteko aldaketa, aurretik azaldutako Temp konponentearen frekuentzia jakin horretan eginez, giza begiari 4 anodoak batera ikusten iruditzen zaio.



Horretaz gainera, 2 deco\_bin konponenentatik 4 balio jasotzen ditu, balio bakoitza katodo balio bat izanez.

* Bin\_h\_i-> ezkerreko erlojuaren balio hamartarra
* Bin\_b\_i-> ezkerreko erlojuaren bateko balioa
* Bin\_h\_d-> eskumako erlojuaren balio hamartarra
* Bin\_b\_d-> eskumakoko erlojuaren bateko balioa

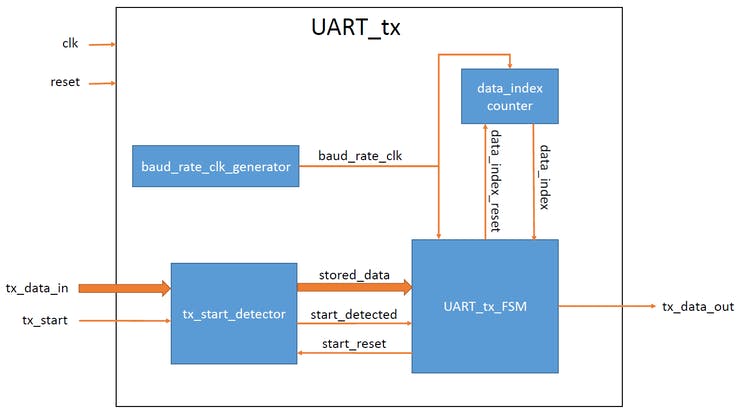
## UART

Konponente honen eginbearra barnean dituen Uart\_tx eta Uart\_rx konponenteen erabilpena erraztea da,bere irteera eta sarrera garrantzitsuenak hauek dira:

* Tx\_start: Linea seriean idazketa astea markatzen duen seinalea da.
  + 1- línea seriean idazten du
  + 0-Ez du línea seriean idazten
* Data\_in: Linea seriean behin tx\_start aktibaturik dagoenean, linea serietik bidaliko den edukia da.
* Tx : Linea seriearen idazketaren logika duen irteera da.
* Rx : Linea seriearen irakurtzearen logika duen sarrera da.
* Data out: Linea serietik irakurtzen den azken karakterea, irteera honetatik pasatzen da. 8 biteko formatuan.

### UART\_TX

Uart\_tx konponentea , UART-aren transmisorea da bere bloke diagrama hurrengokoa da:



#### baud\_rate\_clk\_generator

#### Plakak ordenagailuaren portu seriearekin komunikatzeko baudioak emen definitzen dira hurrengoko formularen bitartez.

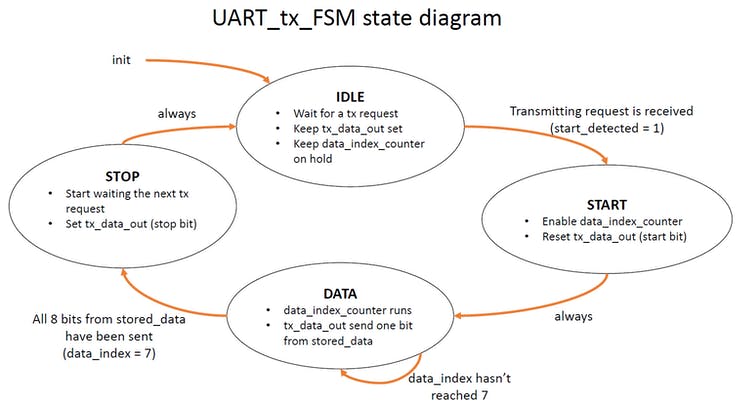
#### tx\_start seinalea iritsi dela konprobatzen duenean, iritsitako datuak gordetzen ditu stored\_data irteeran.

#### Data\_index\_counter

0 tik 7 ra doan counter simple bat da,hau baudioen frekuentzia Berean erabiltzen da store\_data , tx\_data\_out irteerara pasatzen joateko. Hau data\_index seinalea erabiliz egiten da, data\_index ak stored\_data bektorean zehar mugitzeko erabiliz edukiaren bitak banan banan bidaltzen joateko.

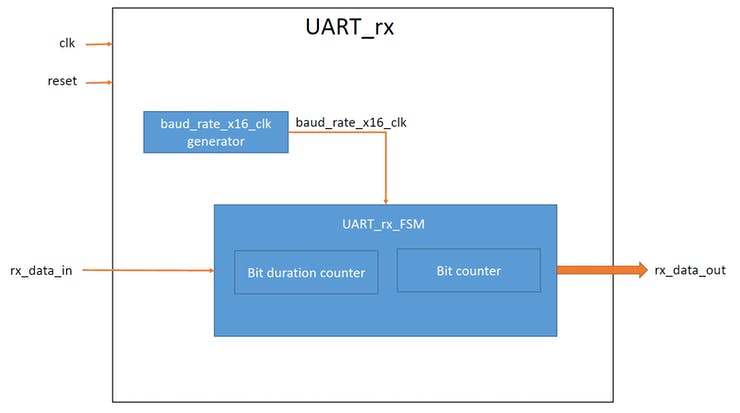
#### UART\_tx\_FSM (finite state machine)

#### UART\_tx aren egoera diagrama duen konponentea da, 4 egoera ditu IDLE, START, DATA eta STOP , egoera diagrama hurrengokoa da.



### Uart\_RX

UART\_rx linea serietik informazioa irakurtzen duen konponentea da, bere egoera diagrama hurrengokoa da:

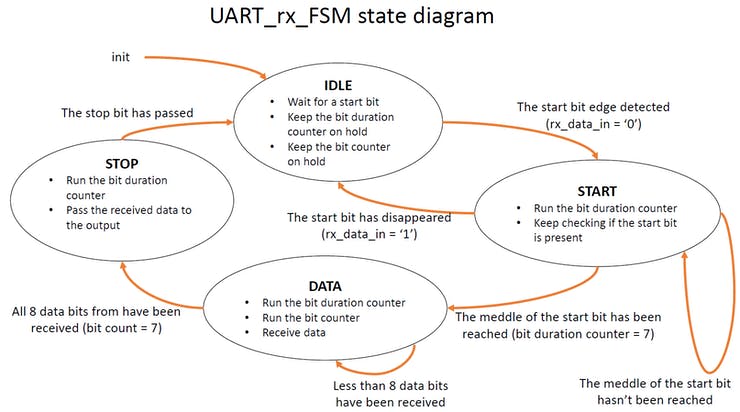


#### baud\_rate\_x16\_clk\_generator

#### Plakak ordenagailuaren portu seriearekin komunikatzeko baudioak emen definitzen dira hurrengoko formularen bitartez. Haurreko UART\_tx elementuan erabilitako formularen desberdina da.

#### UART\_rx\_FSM

#### UART\_rx aren egoera diagrama duen konponentea da, 4 egoera ditu IDLE, START, DATA eta STOP , egoera diagrama hurrengokoa da.



## Deco\_out

Konponente honek , Uart konponenteari, erlojuaren egoera bakoitzean línea serietik idatzi behar duen seinalea pasatzen dio, mandar irteeraren bitartez.

Horretaz gainera botoia sakatzean edo botoia sakatu gabe bidali behar duen ere zehazten du enviar irteeraren bidez. Adibidez amaierako seinaleak, botoiak sakatzean agertzen dira, horretaz gainera dagokion amaieraren karakterea gelditu gabe bidaliko du, reset ematen zaion harte.

|  |  |
| --- | --- |
| **Erlojuaren egoera** | **Uart-eko data\_in sarrerara bidaltzen dueña** |
| Eskerreko erlojua aktibatu | i |
| Eskumako erlojua aktibatu | d |
| Ezkerreko erlojua amaitu | g |
| Eskumako erlojua amaitu | h |

## Button\_controler

Botoien debouncerra da, hau da botoiak sakatzean,seinalea behin bakarrik errepikatzen dela zihurtatzen duen elementua da, horrelako 1 dago botoi bakoitzagatik.

## Mux1

Mux deitzen diot baina egia esanda,datuak bidaltzeko seinalea dagoen kontrolatzen duen konponentea da, or logiko bat eginez botoietatik datozen seinaleei, eta beste or logiko bat, deco\_out etik datorren seinaletik, idazketa aktibatzen duten egoeren seinaleak detektatzen ditugu, aldi berean aktibazioko seinalea bidaliz UART konponentearen tx\_start sarrerari, hórrela idazketa asteko.

## Selector

Emen aurrekoaren berdina egiten dugu, baina Uart konponenteari , línea seriean zer idatzi behar duen ordez. Linea seriean idatzi dugun karakterea jaso egiten du, eta balio hori kontuan hartuta, erlojuari hasierako balio bat edo beste bat pasatzen dio , valor irteraren bitartez. Balio hau reset egitean aplikatzen zaio erlojuari.

Definitutako balioak hurrengokoak dira:

|  |  |
| --- | --- |
| **Linea serieko balioa** | **Erlojuari pasatzen zaion hasiera balioa** |
| a | 33 |
| b | 99 |

# BIBLIOGRAFIA

* <https://reference.digilentinc.com/basys3/refmanual?_ga=2.145691956.1878583647.1608578818-1695917639.1608578818>
* <https://reference.digilentinc.com/reference/pmod/pmodrs232/reference-manual>
* <https://www.youtube.com/watch?v=sTHckUyxwp8>
* <https://vhdl.es/puerto-serie-en-vhdl/>
* <https://www.youtube.com/watch?v=LjVHW_GB4X0>
* <https://core-electronics.com.au/tutorials/serial-monitoring-with-tera-term.html>
* <https://github.com/Digilent/Basys-3-GPIO/blob/master/src/hdl/UART_TX_CTRL.vhd?_ga=2.187126056.1878583647.1608578818-1695917639.1608578818>
* <https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html>
* [https://www.hackster.io/alexey-sudbin/uart-interface-in-vhdl-for-basys3-board-eef170 - comments](https://www.hackster.io/alexey-sudbin/uart-interface-in-vhdl-for-basys3-board-eef170%20-%20comments)
* <https://github.com/AlexHDL/UART_controller>