# Disp\_x\_token module

|  |  |  |
| --- | --- | --- |
| Input name | Full name | Purpose |
| num\_token | Number of tokens to be dispensed | Input vector describing how many tokens are to be given can range from 1-15 |
| start | Start trigger | Start trigger for the FSM to start dispensing tokens |
| clock | Clock | Clock |
| reset | Reset signal | Reset |
| mtne\_mode | Maintenance mode enable | A toggle to override the FSM and set the servo directly |
| mtne\_pos | Maintenance position | The test position the servo should move to |

|  |  |  |
| --- | --- | --- |
| Parameter name | Full name | Purpose |
| wait\_cyles | Number of cycles to wait in each position | Adds a delay to the system so that the servo has time to dispense a token |
| servo\_in | Value for servo being in the in position | Allows the in position of the cycle to be set for each instantiation separately |
| servo\_out | Value for servo being in the out position | Allows the out position of the cycle to be set for each instantiation separately |

|  |  |  |
| --- | --- | --- |
| Output name | Full name | Purpose |
| pwm\_out | Pulse width modulation output | This is the signal that needs to be routed to the servo. |
| done | Done signal | Signal to state that the FSM has completed the dispense and is now waiting for the next command. |
| state\_out | Current FSM state output | This is a debug output that doesn’t need to be connected. |

## Modules instantiated.

* RCServo

## General Overview.

This module can control a servo to drop the required number of tokens. It initially sits in a “init” state. In this state the FSM waits for the start signal, once received the servo is moved to the out position as set by the parameter. The servo is held in this position for a number of PWM cycles equal to the wait\_cyles parameter before being pulled back in and paused for a second wait. Once it has done this a number of times equal to the number of tokens to dispense it returns to the “init” state and send the done signal.

## Notable points.

* Some registers are tagged as /\* synthesis keep \*/ this is to stop the system optimizing them away and effecting the final output.
* In maintenance mode the FSM still runs but its outputs are overridden and ignored, with instead the mtne\_pos vector being used to control the servo.
* There is no feedback in this module It always assumes it has been successful.
* In the pulse count register the flag is used to stop the register being incremented twice by mistake.

## FSM