**Steam Control Logic – Block diagram**

Flow of states

🡪Steam\_Idle

🡪Steam\_stepOn/Off / SteamOn

🡪SteamTimeout

🡪SteamOff

🡪SteamTimeout

🡪SteamOn

🡪SteamDeactive.

Using macros

Steam level factor – it is used for receipe setting (steam level on ,off time that is multiplied with steam level factor.

Using variables

* **Steam control ,Prevsteamcontrol**

🡺 These variables are of the enumerated type steamControlType representing the current and previous states of the steam control.

* **UBYTE cycleCount = 0;** (5sec)(50->100s)used to cyclically run 5 sec =>steponoffcount

🡺 it is used to cyclically turn on the steam after 5 sec.

* **UBYTE onOffStepCount = 0;**

**🡺**it is used to turn on the waterinlet valve for 1 sec.

🡺used to turn off the waterinletvalve after 3 sec .

* **UBYTE NoWaterReportCount = 0;**
* **UBYTE TurnOffIndicationCounter = 0;**

**🡺** These variables are used for counting the reports, and indications related to the steam control process.

* **SteamLevelTimout**

**🡺**It decrements the SteamLevelTimout variable as long as it's greater than zero.

🡺variable serves as a timer to control how long a specific steam level should be active. It helps manage the timing of turning the steam on and off based on the configured durations for different steam levels. If the timeout reaches zero, it triggers the deactivation of steam, contributing to the overall control logic of the steam system.

🡺During the **E\_steamTimeOut** state, the **SteamLevelTimeout** is decremented until it reaches zero.

* **Steam level**

🡺The function **updateSteamLevel** takes a parameter **SteamLevel** (representing low, medium, or high steam level).

* **CS\_RecipeSettings\_ST**

🡺a pointer to a structure **CS\_RecipeSettings\_ST** containing settings for different steam levels.

* **BOOL IsWaterInletOpenFlag = FALSE;**

🡺A boolean flag indicating whether the water inlet is open or closed.

* **volatile BOOL SteamTurnOff = FALSE;**

🡺A boolean flag indicating whether the steam should be turned off. The volatile keyword suggests that the variable can be modified asynchronously

Explaination

1. **Idle**
   * In this state, the steam remains Idle.
2. **SteamOn**

* In this state, the water inlet valve is turned on. The SteamLevelTimout is set to steamLevelOnTime, and the control\_state is switched to E\_steamTimeOut. The prevSteamControl is updated to E\_steamOn.

1. **SteamTimeOut**

* This case handles a timeout mechanism. It decrements the SteamLevelTimout variable as long as it's greater than zero. When it reaches zero, it toggles the steamControl state between E\_steamOn and E\_steamOff based on the previous state stored in prevSteamControl.

1. **SteamOff**

* In this State, the water inlet valve is turned off, and the SteamLevelTimout is set to a value called steamLevelOffTime. The control\_state is switched to E\_steamTimeOut, and the prevSteamControl is updated to E\_steamOff.

**SteamStepOnOff**

* In this state, First TurnOn steam that time water inlet valve is 30s(1 sec ON – 4 sec OFF)- to Switched steamOn/steamOff. because its used to overcome the rampUP.
* In this state, there is a sequence of steps to control a water inlet valve based on the values of onOffStepCount and cycleCount. Depending on the values of these variables, the water inlet valve is turned on or off, and these variables are updated accordingly. If cycleCount exceeds 50, the control\_state is switched to E\_steamOn, and a timeout value (SteamLevelTimout) is set to steamLevelOnTime.

**SteamDeactivate**

* In this case, the water inlet valve is turned off, and control states are reset. prevSteamControl and steamControl are set to E\_steamIdle, and NoWaterReportCount is reset to zero.This state is switched to idle state.