RecipeConfig Logic

🡪This File receives all Cooking related data

🡪Initial Combinational Settings data, Steam Settings data, and Hot Air Settings Data are Received Here , these setting are important to do the cooking process.

🡪Finally, Receive the Recipe Configuration data and All steps (Step 1 to 6) data are received here.

**Flags**

combiRecipe\_Flag 🡪 This flag may be used to indicate whether combi recipe data has been received or not.

steamRecipe\_Flag 🡪 This flag may be used to indicate whether steam recipe data has been received or not.

bakingRecipe\_Flag 🡪 This flag may be used to indicate whether baking recipe data has been received or not.

recipeStepConfig\_ST recipeStepConfig 🡪 This likely represents the configuration settings for a specific recipe step.

recipeSteps\_ST recipeSteps[TOTAL\_NO\_RECIPE\_STEPS] 🡪 each element in the array represents the data for a specific recipe step. The array size is determined by the constant TOTAL\_NO\_RECIPE\_STEPS.

CS\_RecipeSettings\_ST CS\_RecipeSettings[2] 🡪 Each element in the array represents the settings for a specific recipe type. The array size is 2, indicating settings for two different recipes.

bakingRecipeSettings\_ST bakingRecipeSettings 🡪 This likely represents the settings specific to baking recipes.

**RecipeStepConfig RX Callback()**

🡪recipeStepConfig\_ST \*rSC: a pointer to a structure representing the configuration of a recipe step.

🡪DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

🡪 rSC->Byte = (UBYTE) \*(Data);: the byte of data received from HMI, and stores it in the Byte field of the recipeStepConfig structure.

**Validation:**

🡪 (rSC->Byte != 0): Checks if the extracted byte is not equal to zero. (Length == ComIf\_GetLength\_HMI\_recipeStepConfiguration()): Checks if the length of the received data matches the expected length for the recipe step configuration.

**RCF.recipeStepConfigFlag = TRUE;**

🡪If this check is true, set the recipeStep Config Flag to True. It is indicated by the initial of the recipes Config step, which is 'Received the MUC.'.

🡪This check is not true, it reports to HMI, And Set RCR Bit, This RCR Bit Requests the Current step of the data To HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.

**RecipeStep1RxCbk ()**

🡪 This function is Get the First step of the Configured data

recipeSteps\_ST \*rS1: a pointer to a structure representing the data for the first step of a recipe.

recipeStepConfig\_ST \*rSconfig: a pointer to a structure representing the configuration of a recipe step.

DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

**Validation:**

🡪 (RCF.recipeStepConfigFlag == TRUE): Checks if the extracted byte is not equal to zero. (Length == ComIf\_GetLength\_HMI\_recipeStepConfiguration()): Checks if the length of the received data matches the expected length for the recipe step configuration.

**🡪** If both validation conditions are met, a loop copies the received data into the Bytes array of the recipeSteps structure for the first step.

**🡪** **if((rS1->runTimeInMinutes == 0) && (ovenCookingState > OCS\_machineIdle))**, This Check is represented by the MultiTry Mode. It’s all values are Run time updates.

**🡪** **if(IsCookingProgress())**,This check is Represented by the Oven, it here the Cooking Progress, that time the Config Cooking Timing is converting to 100ms Updated to the runTimeIn100Millis variable.

**RCF.** **recipeStep1Flag= TRUE;**

🡪If this check is true, set the recipeStep Config Flag to True. It is indicated by the initial of the recipes Config step, which is 'Received the MUC.'.

🡪This check is not true, it reports to HMI, And Set RCR Bit, This RCR Bit Requests the Current step of the data To HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.

**RecipeStep2RxCbk ()**

🡪 This function is Get the second step of the Configured data

recipeSteps\_ST \*rS1: a pointer to a structure representing the data for the first step of a recipe.

DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

**Validation:**

🡪 (RCF.recipeStep1Flag==TRUE): Checks if the extracted byte is not equal to zero. (Length == ComIf\_GetLength\_HMI\_recipeStepConfiguration()): Checks if the length of the received data matches the expected length for the recipe step configuration.

**🡪** If both validation conditions are met, a loop copies the received data into the Bytes array of the recipeSteps structure for the first step.

**🡪** **if(IsCookingProgress())**, This check is Represented by the Oven, it here the Cooking Progress, that time the Config Cooking Timing is converting to 100ms Updated to the runTimeIn100Millis variable.

**RCF. recipeStep2Flag = TRUE;**

🡪If this check is true, set the recipeStep Config Flag to True. It is indicated by the initial of the recipes Config step, which is 'Received the MUC.'.

🡪This check is not true, it reports to HMI, And Set RCR Bit, This RCR Bit Requests the Current step of the data To HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.

Note:- Above the **RecipeStep2RxCbk()** method is Execute the balance steps.

**combiRecipeRxCbk()**

🡪This function, **combiRecipeRxCbk**, is a callback designed to handle the reception of data related to combi recipe settings.

🡪 CS\_RecipeSettings\_ST \*cRS: a pointer to a structure representing combi recipe settings.

🡪DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

🡪machineStatus\_ST \*MS: a pointer to a structure representing the status of the machine.

🡪 Flag Setting:

🡪combiRecipe\_Flag = TRUE;: Sets a global flag, likely indicating that combi recipe data has been received. This may be used for clearing the boot bit.

🡪 Checks if the length of the received data matches the expected length for combi recipe settings (Length == ComIf\_GetLength\_HMI\_combiRecipeSettings()).

If the validation is successful, a loop copies the received data into the Bytes array of the CS\_RecipeSettings structure for combi recipes.

🡪 If the validation condition is not met (indicating an error in received data), an error event is reported using Diag\_ReportEvent.

MS->BOOT is set to TRUE, likely indicating the received data is not proper or not received data, so it again requests the Combination Recipe setting to HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.

**steamRecipeSettingsRxCbk()**

🡪This function, **steamRecipeSettingsRxCbk**, is a callback designed to handle the reception of data related to Steam recipe settings.

🡪 CS\_RecipeSettings\_ST \*cRS: a pointer to a structure representing combi recipe settings.

🡪DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

🡪machineStatus\_ST \*MS: a pointer to a structure representing the status of the machine.

🡪 Flag Setting:

🡪 steamRecipe\_Flag = TRUE;: Sets a global flag, likely indicating that steam recipe data has been received. This may be used for clearing the boot bit.

🡪 Checks if the length of the received data matches the expected length for steam recipe settings (Length == ComIf\_GetLength\_HMI\_steamRecipeSettings ()).

If the validation is successful, a loop copies the received data into the Bytes array of the CS\_RecipeSettings structure for steam recipes.

🡪 If the validation condition is not met (indicating an error in received data), an error event is reported using Diag\_ReportEvent.

MS->BOOT is set to TRUE, likely indicating the received data is not proper or not received data, so it again requests the Combination Recipe setting to HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.

**bakingRecipeSettingsRxCbk()**

🡪This function, **bakingRecipeSettingsRxCbk**, is a callback designed to handle the reception of data related to bakingRecipe settings.

🡪 bakingRecipeSettings\_ST \*bRS : a pointer to a structure representing baking recipe settings.

🡪DiagnosticsData\_ST \*DD: a pointer to a structure for diagnostic data.

🡪machineStatus\_ST \*MS: a pointer to a structure representing the status of the machine.

🡪 Flag Setting:

🡪 bakingRecipe\_Flag = TRUE;: Sets a global flag, likely indicating that baking recipe data has been received. This may be used for clearing the boot bit.

🡪 Checks if the length of the received data matches the expected length for baking recipe settings (Length == ComIf\_GetLength\_HMI\_ bakingRecipeSettings ()).

If the validation is successful, a loop copies the received data into the Bytes array of the bakingRecipeSettings structure for baking recipes.

🡪 If the validation condition is not met (indicating an error in received data), an error event is reported using Diag\_ReportEvent.

MS->BOOT is set to TRUE, likely indicating the received data is not proper or not received data, so it again requests the Combination Recipe setting to HMI.

🡪 The Sync Flag Indicates the synchronization of uC and HMI.