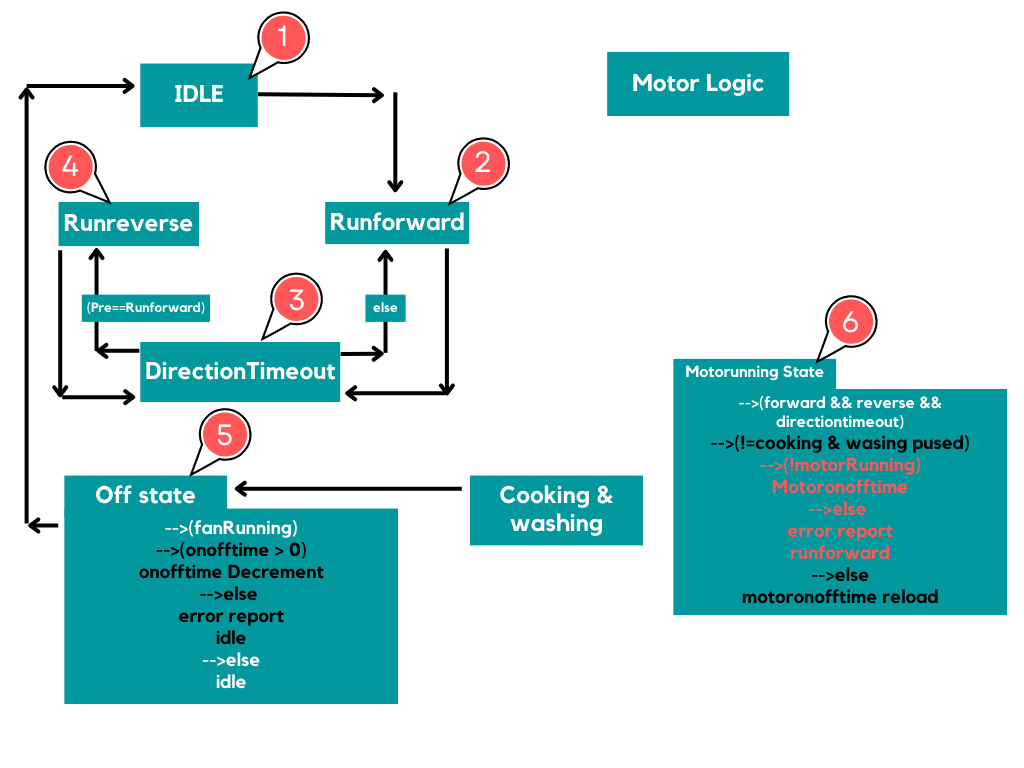
**Fan Motor logic**

**Using Macros**

* **MOTOR\_DIRECTION\_SWITCH\_TIMEOUT**

**🡪** This macro is used for changing the direction of the motor.

* **MOTOR\_ON\_OFF\_TIMEOUT**

**🡪**This macro is used for find the motor status.

* **MOTOR\_SWITCHED\_TIMEOUT**

**🡪** This macro is used to wait to take some seconds, after that Turn ON Forward / Reverse.

**Using Variables**

**fanMotorControl** 🡪 This variable used for controlling the machine states.

**prevMotorControlState**  🡪 This variable used for saving the current sate.

**motorDirectionSwitch , motorOnOffTimeout, motorswitchedtimeout**

🡪 These variables are used for the timeouts

**Flags**

* ovenPauseControl
* fanMotorRunningStatus

**Flow of states**

**🡪Motor \_ Idle**

🡪**Motor\_Forward**

🡪**Motor\_DirectionTimeout**

🡪**Motor\_Reverse**

**🡪Motor\_DirectionTimeout**

🡪 **Motor\_Off**

1. **Motor\_Idle**

🡪 In this state, the Motor works nothing.

1. **Motor\_RunForward**

**🡪** This condition allows the fan motor to turn in the forward direction. In the case of a Variable Frequency Drive (VFD), it makes sure that the previous motor direction (reverse) is turned off before turning on the forward motor to avoid problems.

**🡪** **(motorswitchedtimeout > 0)**

**🡪** This code verifies the motorswitchedtimeout variable's value. It indicates that the previous motor switching activity is still in progress if it is greater than 0. In this instance, it turns off the forward and reverse motor switches and decreases the motorswitchedtimeout.

🡪 **Timeout Handling**

🡪If motorswitchedtimeout is greater than 0, it means that some motor switching operation is currently in progress. The code decrements motorswitchedtimeout to wait for that operation to complete.

🡪During this waiting period, both the reverse and forward motor switches are turned off using the TURN\_REVERSE\_MOTOR\_OFF() and TURN\_FORWARD\_MOTOR\_OFF() functions.

🡪**Change Forward**

🡪 Motor switching is assumed to be complete when motorswitchedtimeout equals 0. The code then moves on to the else block in this situation.

🡪The **TURN\_FORWARD\_MOTOR\_ON()** function is used to activate the forward motor in the else block.

🡪 **Variable Updates**

🡪**prevMotorControlState** is updated to **E\_fanMotorRunForward** to keep track of the previous motor control state.

**🡪motorswitchedtimeout** is reset to a value defined as **MOTOR\_SWITCHED\_TIMEOUT**, presumably for some timeout mechanism.

**🡪motorDirectionSwitch** is also reset, possibly for a direction switch timeout mechanism.

🡪 The state is changed to **E\_fanMotorDirectionTimout**, a timeout condition.

1. **Motor\_DirectionTimeout**

🡪 This condition is responsible for controlling the fan motor's direction. It switches between forward and reverse shifting using a timeout mechanism.

🡪 **(motorDirectionSwitch > 0)**

**🡪** It checks the motorDirectionSwitch variable's value. This indicates that the direction switching timeout is still running if the value is greater than 0.

**🡪** **Direction Switch**

🡪When the **motorDirectionSwitch** timer reaches 0, it means it's time to switch the direction of the fan motor.

🡪The code then checks the **prevMotorControlState** variable, which should store the previous state of motor control (whether it was running in the forward direction or not).

🡪 If the previous state stored in **prevMotorControlState** **is E\_fanMotorRunForward**, it indicates that the fan motor was running in the forward direction before. In this case, the code sets the **fanMotorControl** variable to **E\_fanMotorRunReverse**, which is a signal to switch the motor to reverse direction.

🡪If the previous state is anything other than forward, it sets the fanMotorControl variable to

E\_fanMotorRunForward, indicating that the fan motor should run in the forward direction.

1. **Motor\_RunReverse**

🡪 This is controls the direction of a fan motor. In particular, this state is responsible for running the fan motor in the reverse direction. It ensures that before turning on the reverse motor, any previous motor direction (forward) is switched off to prevent conflicts, especially in the context of a Variable Frequency Drive (VFD).

**🡪** **(motorswitchedtimeout > 0)**

**🡪** This code verifies the motorswitchedtimeout variable's value. It indicates that the previous motor switching activity is still in progress if it is greater than 0. In this instance, it turns off the forward and reverse motor switches and decreases the motorswitchedtimeout.

🡪 **Timeout Handling**

🡪If motorswitchedtimeout is greater than 0, it means that some motor switching operation is currently in progress. The code decrements motorswitchedtimeout to wait for that operation to complete.

🡪During this waiting period, both the reverse and forward motor switches are turned off using the TURN\_REVERSE\_MOTOR\_OFF() and TURN\_FORWARD\_MOTOR\_OFF() functions.

🡪**Change Reverse**

🡪 Motor switching is assumed to be complete when motorswitchedtimeout equals 0. The code then moves on to the else block in this situation.

🡪The **TURN\_REVERSE\_MOTOR\_ON()** function is used to activate the Reverse motor in the else block.

🡪 **Variable Updates**

🡪**prevMotorControlState** is updated to **E\_fanMotorRunReverse** to keep track of the previous motor control state.

**🡪motorswitchedtimeout** is reset to a value defined as **MOTOR\_SWITCHED\_TIMEOUT**, presumably for some timeout mechanism.

**🡪motorDirectionSwitch** is also reset, possibly for a direction switch timeout mechanism.

🡪 The state is changed to **E\_fanMotorDirectionTimout**, a timeout condition.

1. **Motor\_OFF**

🡪This state is responsible for ensuring that the fan motor is turned off.

🡪**(IS\_FAN\_MOTOR\_RUNNING())**

🡪Checking If Motor Is Still Running

🡪If the motor is still running despite the switches being turned off, it indicates an error. In this case, it proceeds to handle the error.

🡪Handling Motor Successfully Turning Off:

🡪If the motor is not running, meaning it has successfully turned off, it simply transitions the **fanMotorControl** state to **E\_fanMotorIdle** to indicate that the motor is in an idle state.

🡪It also resets the **motorOnOffTimeout** to **MOTOR\_ON\_OFF\_TIMEOUT**.

1. Motor\_running\_state

**🡪((fanMotorControl == E\_fanMotorRunForward) || (fanMotorControl == E\_fanMotorRunReverse) || (fanMotorControl == E\_fanMotorDirectionTimout))**

🡪That checks if the fanMotorControl variable is set to either E\_fanMotorRunForward, E\_fanMotorRunReverse, or E\_fanMotorDirectionTimout. These states suggest that the fan motor should be running.

**🡪((MS->MachineStaus != MS\_cookingPaused) && (MS->MachineStaus != MS\_washingPaused))**

**🡪,**It checks if the machine status (MS->MachineStaus) is not in a paused state.

🡪**(!IS\_FAN\_MOTOR\_RUNNING())**

🡪If the motor is not running, it proceeds to reports an error using Diag\_ReportEvent().