Create a motor interlock as a function block diagram that ensures that a motor is not started while equipment is still running.

Here's a function block diagram for a motor interlock that ensures that a motor is not started while equipment is still running:

sql Copy code

```
Start Switch    +---------+         +------------+
-------------->|         |         |            |
               | AND gate|-------->| Motor      |
Stop Switch    |         |         | Interlock  |
-------------->|         |         |            |
               +---------+         +------------+
                                   ^
Equipment Running                  |
Signal -----------------------------+
```

The diagram includes two switches, a start switch and a stop switch, that control the operation of the motor. The output of each switch is connected to an input of an AND gate. The other input of the AND gate is connected to an equipment running signal. This signal is generated by equipment sensors that detect when the equipment is still running.

If the equipment is still running, the equipment running signal will be high, and the output of the AND gate will be low regardless of the state of the start switch. This prevents the motor from starting until the equipment is stopped. If the equipment is not running, the equipment running signal will be low, and the output of the AND gate will depend on the state of the start switch. If the start switch is closed, the output of the AND gate will be high, and the motor will start. If the start switch is open, the output of the AND gate will be low, and the motor will not start.

Provide the content of the Motor Interlock function block as a 61131-3 structured text function block.

Sure, here's an example of a Motor Interlock function block written in 61131-3 structured text:

sql Copy code

```
FUNCTION_BLOCK MotorInterlock
VAR_INPUT
    StartSwitch: BOOL;           (* input: start switch signal *)
    StopSwitch: BOOL;            (* input: stop switch signal *)
    EquipmentRunning: BOOL;      (* input: equipment running signal *)
END_VAR
VAR_OUTPUT
    MotorInterlockOut: BOOL;     (* output: motor interlock signal *)
END_VAR
VAR
    Temp: BOOL;                  (* temporary variable *)
END_VAR

(* Check if equipment is running *)
IF EquipmentRunning THEN
    MotorInterlockOut := FALSE;  (* If equipment is running, prevent motor
ELSE
    (* If equipment is not running, check state of start switch *)
    IF StartSwitch THEN
        MotorInterlockOut := TRUE;   (* If start switch is closed, allow mo
    ELSE
        MotorInterlockOut := FALSE;  (* If start switch is open, prevent mo
    END_IF
END_IF

(* Check if stop switch is pressed *)
IF StopSwitch THEN
    MotorInterlockOut := FALSE;      (* If stop switch is pressed, stop mot
END_IF
```

This function block takes in three inputs: StartSwitch, StopSwitch, and EquipmentRunning, and outputs MotorInterlockOut. The function block checks the state of the inputs and sets the output accordingly. If the equipment is running, the function block prevents the motor from starting. If the start switch is closed and the equipment is not running, the function block allows the motor to start. If the stop switch is pressed, the function block stops the motor. The use of a temporary variable, Temp, is optional but can make the code more readable.