Lawrence Berkeley National Laboratory
LBL Publications

Title

Fault-Free Optimal Control Sequence Specifications

Permalink

https://escholarship.org/uc/item/0dp15101

Authors

Granderson, Jessica
Casillas, Armando
House, John

Publication Date

2026-10-06

DOI

10.20357/B7B897

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial License, available at https://creativecommons.org/licenses/by-nc/4.0/

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California



Buildings & Industrial Energy Systems
Energy Technologies Area
Lawrence Berkeley National Laboratory

Fault-Free Optimal Control Sequence Specifications

Jessica Granderson, Armando Casillas, John House

Lawrence Berkeley National Laboratory

Energy Technologies Area
October 2026

Granderson, J, Casillas, A, House, J. Lawrence Berkeley National Laboratory. Fault-Free Optimal Control Sequence Specifications. 2026. DOI: 10.20357/B7B897



This work was supported by the Assistant Secretary Office of Critical Minerals and Energy Innovation (CMEI) of the US Department of Energy under Contract No. DE-AC02-05CH11231.

Disclaimer:

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

Fault-Free Optimal Control Sequence Specifications

Automated analytics and control solutions for fault correction, functional performance testing, demand flexibility, and energy optimization



Jessica Granderson, Armando Casillas, John House

Building and Industrial Energy Systems Division

Lawrence Berkeley National Laboratory

Suggested attribution: Granderson, J, Casillas, A, House, J. Lawrence Berkeley National Laboratory. Fault-Free Optimal Control Sequence Specifications. 2026. DOI:#####."

This work was supported by the Assistant Secretary for Critical Minerals and Energy Innovation, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Disclaimer: This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

1. Introduction **7**
2. Fault Correction Algorithms **8**
2.1 Correction of Economizer High Limit Temperature Setpoint 8
2.2 Correction of Zone Temperature Setpoints 11
2.3 Release of unnecessary control overrides 15
2.4 Correction of Incorrectly Programmed HVAC Schedules 18
2.5 Automated Loop Tuning 21
3. Continuous Optimization Algorithms **23**
3.1 Rogue Zone Suppression 24
3.2 AHU Supply Air Temperature Setpoint Trim and Respond 27
3.3 AHU Supply Air Static Pressure Setpoint Trim and Respond 31
4. Demand Flexibility Algorithms **37**
4.1 Zone-level Strategy: Zone Temperature Adjustment 37
4.1.1 Required data points 40
4.1.2 Zone prioritization and qualification check 41
4.1.3 Shed strategy 43
4.1.4 Ratcheting Sequence: Incremental Shed Preserving Temperature Ratcheting 44
4.1.5 Ratcheting Sequence: Automated/Manual Performance Target Based Ratcheting 46
4.1.6 Post-event strategy (Rebound management) 48
4.2 Zone-level Strategy (AHU, MZ-VAV): Zone Precooling 48
4.2.1 Simple Approach 49
4.2.1.1 Required data points 49
4.2.1.2 Parameter definition 51
4.2.1.3 Zone prioritization and qualification check 51
4.2.1.4 Precool Strategy 52
4.2.2 Complex Approach 52
4.2.2.1 Required data points 52
4.2.2.2 Parameter definition 54
4.2.2.3 Zone prioritization and qualification check 55
4.2.2.4 Precool strategy 55
4.2.2.5 Demand modulation via max VAV limit 56
4.3 Zone-level Strategy (SZ-RTU): Zone Precooling 56
4.3.1 Simple approach 57
4.3.1.1 Required data points 57
4.3.1.2 Parameter definition 58
4.3.1.3 Zone prioritization and qualification check 59
4.3.1.4 Precool strategy 59
4.3.2 Intermediate approach 60
4.3.2.1 Required data points 60
4.3.2.2 Parameter definition 61

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.3.2.3 Zone prioritization and qualification check....61
4.3.2.4 Precool strategy....62
4.3.3 Complex Approach....63
4.3.3.1 Required data points....63
4.3.3.2 Parameter definition....65
4.3.3.3 Zone prioritization and qualification check....65
4.3.3.4 Precool strategy....66
4.3.3.5 Demand modulation via max RTU limit....66
4.4 Plant-level Strategy: Chilled water temperature setpoint reset....67
4.4.1 Required data points....67
4.4.2 Strategy description....69
4.4.2.1 Chiller plant load check....72
4.4.2.2 Initial shed....73
4.4.2.3 Ratcheting strategy....73
4.4.2.4 Rebound management....74
4.5 AHU Level Strategy: Fan/VFD Speed Limiting....75
4.5.1 Required data points....76
4.5.2 Prioritization and qualification check....78
4.5.3 Shed strategy....79
4.5.4 Ratcheting sequence....79
4.5.5 Post-event strategy (Rebound management)....79
4.6 AHU Level Strategy: Cooling Valve Limiting....81
4.6.1 Required data points....82
4.6.2 Prioritization and qualification check....86
4.6.3 Shed strategy....86
4.6.4 Ratcheting sequence....86
4.6.5 Post-event strategy (Rebound management)....87
5. Auto-Commissioning Algorithms....87
5.1 Open-loop Functional Tests for VAV Terminal Units....87
5.1.1 System Description....87
5.1.2 Process Description....88
5.1.3 Required Data Points....90
5.1.4 Description of the Open-loop Functional Test and Analysis....95
5.1.4.1 Proactive Functional Test....95
5.1.4.1.1 Verify Test Conditions....95
5.1.4.1.2 Perform Open-loop Tests....95
5.1.4.1.2.1 Damper Tests....96
5.1.4.1.2.2 Reheat Valve Tests....98
5.1.4.2 Functional Test Analysis....100
5.1.4.2.1 Rule Set....100
5.1.4.2.1.1 Rules for Damper Tests....100

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Test Segment D1....100
Test Segment D2....100
Test Segment D3....101
Test Segment D4....101
Test Segment D5....102
5.1.4.2.1.2 Rules for Reheat Valve Tests....104
Test Segment V1....105
Test Segment V2....105
Test Segment V3....105
Test Segment V4....106
5.1.4.2.2 Root Cause Diagnostics....108
5.1.4.2.2.1 Multi-Mode Analysis....109
5.1.4.2.2.2 Inverse Multi-Mode Analysis....114
Leaking Damper....114
Stuck Open Damper....115
Stuck Closed Damper....117
Leaking Reheat Valve....118
Stuck Open Reheat Valve....119
Stuck Closed Reheat Valve....120
Reheat Valve Actuator Reversal....122
Discharge Air Temperature or AHU Supply Air Temperature Sensor Fault....123
5.1.5 Appendix....124
5.2 Open-loop Functional Tests for AHUs....127
5.2.1 System Description....127
5.2.2 Process Description....128
5.2.3 Required Data....130
5.2.4 Description of the Open-loop Functional Test and Analysis....133
5.2.4.1 Pretest Condition Check - Step 1....133
5.2.4.2 Proactive Functional Test - Step 2....133
5.2.4.2.1 Description of Tests....134
5.2.4.2.2 Pseudocode for Test Implementation....148
5.2.4.3 Functional Test Analysis - Steps 3 - 7....150
5.3 Closed-loop Functional Tests for VAV Terminal Units....159
5.3.1 Overview....159
5.3.2 System Description....159
5.3.3 Process Description....161
5.3.4 Required Data Points....164
5.3.5 Description of the Closed-loop Functional Test and Analysis....168
5.3.5.1 Proactive Functional Test....168
5.3.5.1.1 Verify Test Conditions....168
5.3.5.1.2 Perform Closed-loop Tests....170

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.3.5.2 Functional Test Analysis..... 172
5.3.5.2.1 Rule Set..... 172
5.3.5.2.1.1 Cooling Mode Rules..... 172
5.3.5.2.1.2 Deadband Mode Rules..... 173
5.3.5.2.1.3 Heating Mode Rules..... 174
5.3.5.2.2 Root Cause Diagnostics..... 177
5.3.5.2.2.1 Multi-Mode Analysis..... 178
5.3.5.2.2.2 Inverse Multi-Mode Analysis..... 183
Stuck Damper..... 183
Leaking Reheat Valve..... 185
Stuck Open Reheat Valve..... 186
Stuck Closed Reheat Valve..... 187
Reheat Valve Actuator Reversal..... 187
Discharge Air Temperature or AHU Supply Air Temperature Sensor Bias Fault 188
5.4 Closed-loop Functional Tests for AHU Supply Air Temperature and Outdoor Air Control... 189
5.4.1 Overview..... 189
5.4.2 System Description..... 190
5.4.3 Process description..... 192
5.4.4 Required Data Points..... 194
5.4.5 Part 1_Proactive Functional Test..... 198
5.4.5.1 Check test conditions..... 198
5.4.5.2 Force AHU to operate in four operating modes and record test data..... 199
5.4.6 Part I_Functional Test Analysis..... 200
5.4.6.1 Procedure..... 200
5.4.6.1.1 Group test data into different operating modes..... 200
5.4.6.1.2 Apply APAR and auxiliary rules for assessing AHU performance..... 201
5.4.6.1.3 Identify shared faults from triggered APAR rules..... 203
5.4.6.1.4 Identify excluded faults from non-triggered APAR rules..... 204
5.4.6.1.5 Identify excluded faults from auxiliary rules..... 204
5.4.6.1.6 Remove excluded faults from shared faults..... 204
5.4.6.1.7 Determine if a single root-cause fault identified..... 205
5.4.6.2 Implementation and pseudocode..... 205
5.4.6.2.1 Group test data into different operating modes..... 205
5.4.6.2.2 Apply APAR and auxiliary rules for assessing AHU performance..... 205
5.4.6.2.3 Identify shared faults from triggered APAR rules..... 208
5.4.6.2.4 Identify excluded faults from non-triggered APAR rules..... 211
5.4.6.2.5 Identify excluded faults from auxiliary rules..... 213
5.4.6.2.6 Remove excluded faults from shared faults..... 215
5.4.6.2.7 Determine if a single root-cause fault identified..... 216

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.4.7 Part II_Proactive Functional Test And Analysis..... 217
5.4.7.1 Stuck/Leaking Heating Valve Test.....218
5.4.7.1.1 Procedure.....218
5.4.7.1.2 Implementation.....220
5.4.7.2 Stuck/Leaking Cooling Coil Valve Test.....220
5.4.7.2.1 Procedure.....220
5.4.7.2.2 Implementation.....221
5.4.8 Appendix.....222
5.5 Closed-loop Functional Tests of Static Pressure Reset.....225
1. 5.5.1 Overview.....225
2. 5.5.2 System Description.....225
3. 5.5.3 Process description.....226
5.5.3.1 Required Data Points.....227
5.5.3.2 Proactive Functional Test.....230
5.5.3.3 Functional Test Analysis..... 234
5.5.3.3.1 Test Segment 1..... 235
5.5.3.3.2 Test Segment 2..... 236
5.5.3.3.3 Test Segment 3..... 238
5.5.3.3.4 Test Segment 4..... 239
5.5.3.4 Functional Test Outcomes..... 240
5.5.3.4.1 Static Pressure Reset Performance.....240
5.5.3.4.2 Setpoint Tracking Performance.....241
5.5.3.4.3 Capacity and Turndown Performance.....242
5.5.4 Other Considerations.....242

1. Introduction

Fault-free optimal control refers to a suite of capabilities to streamline and automate the correction of common HVAC faults, and the implementation of advanced HVAC control strategies. These capabilities are implemented using energy management and information system (EMIS) software integrated with the building automation system (BAS). They are being delivered by the U.S. Department of Energy and Berkeley Lab in partnership with leading providers of smart building analytics. The spectrum of fault-free optimal control strategies ranges from one-time fixes - where common control faults are automatically identified and upon user approval, automatically corrected - to continuous supervisory updates of control parameters. These supervisory updates can drive an afternoon of demand response or even resets that provide year-round energy-efficient operation.

The fault-free optimal control strategies reside in the EMIS and are enabled by two-way connectivity between the EMIS and the BAS. Operational data are read from the BAS (and other sources, e.g., energy meters, or weather data) and the analytics in the EMIS determine the presence of faults, corrective actions, and supervisory control updates. The user can then

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

authorize the push of fault corrections and advanced control sequences from the EMIS to the BAS. This enables rapid resolution of problems and control improvements through user-friendly analytics software, and removes the need for time-consuming manual changes to the BAS – improving today's common practice. The set of corrective control strategies developed to date consists of:

- Correction of zone temperature setback setpoints
- Correction of economizer high limit temperature setpoint
- Correction of incorrectly programmed HVAC schedules
- Correction of biased temperature sensors
- Release of unnecessary control overrides
- Automated loop tuning

The optimization strategies include best practice supervisory reset strategies from ASHRAE Guideline 36 that promote energy efficient operation, as well as load shifting and shedding sequences for demand flexibility. In each case, control parameters calculated within the EMIS are pushed continuously to the BAS. The full set of optimal control strategies developed to date consists of:

Zonal reset strategies -

- Supply air temperature reset
- Supply air static pressure reset
- Rogue zone suppression

Demand flexibility via -

- Zone temperature adjustment
- Zone temperature adjustment for precooling
- Chilled water temperature setpoint reset
- Supply fan and cooling coil valve limiting

Apart from the corrective control and optimization strategies outlined above, FFOC also includes a set of closed-loop and open-loop automated functional tests that can be deployed from the EMIS tool to help with the commissioning process.

The next section presents detailed descriptions of each FFOC strategy, including algorithm logic and specifications, along with a summary of key variables involved in implementation. This includes: required data points that must be read from the BAS and written to the BAS, data sampling requirements and how frequently the algorithms are executed.

2. Fault Correction Algorithms

These algorithms are used to automatically correct a fault that has been detected and diagnosed by an EMIS tool and generally involve overwriting setpoints, schedules, or other variables and properties in the BAS with corrected values from the EMIS tool. Most corrective actions are typically implemented as a one-time event in the sense that only a single communication from the FDD tool to the BAS is needed to resolve the issue. But they can also be applied continuously, functioning as supervisory control updates that maintain optimal performance over time.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

2.1 Correction of Economizer High Limit Temperature Setpoint

Overview: Airside economizers promote energy efficient AHU operation by enabling “free cooling” with outdoor air when conditions are suitable. Economizer logic controls dampers to allow mild outdoor air in excess of ventilation requirements to enter the AHU, thereby reducing the mechanical cooling load. A high limit setpoint determines the condition above which economizer operation and free cooling are disabled. Design engineers may specify an unnecessarily low setpoint out of concern that humidity control could be jeopardized at warmer outdoor conditions, and facilities staff may lower the high limit setpoint in response to operational problems unrelated to the economizer. Consequently, the range of conditions over which free cooling is available will be reduced. The one-time corrective strategy outlined below applies to dry-bulb economizers with fixed high limit temperature setpoints. A strategy for continuous correction/optimization of differential dry-bulb economizers is also described.

Algorithm Description: The algorithm for correcting fixed dry-bulb economizers compares the actual high limit setpoint from the BAS to the high limit setpoint from ASHRAE 90.1 - 2016 and the lowest cooling setpoint for all zones served by the AHU. If the high limit setpoint from the BAS is less than the minimum of these values, an opportunity to expand the range of free cooling exists and a fault is identified. Once the fault is diagnosed, the corrective strategy pushes the corrected high limit temperature setpoint (the minimum value of the high limit setpoint defined in ASHRAE 90.1-2016 and all cooling setpoints for zones served by an AHU) to the BAS using the two-way communication capabilities between the EMIS tool and BAS.

Averaged return air temperature can also be used in lieu of the lowest cooling setpoint of all zones served by AHU. If the return air temperature is used, it needs to be averaged over an appropriate time interval, selected based on operational goals and the fault correction process. For example, if it's done on a monthly basis, the algorithm should ensure that the corrected high limit setpoint does not exceed the time-averaged RAT during that period.

This algorithm can also operate in a continuous optimization mode by periodically checking whether the economizer is enabled under appropriate conditions. In this case, instead of modifying the high limit setpoint, the economizer enable point is overwritten to maintain optimal control - effectively implementing a differential dry-bulb economizer sequence as described in Figure 1.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

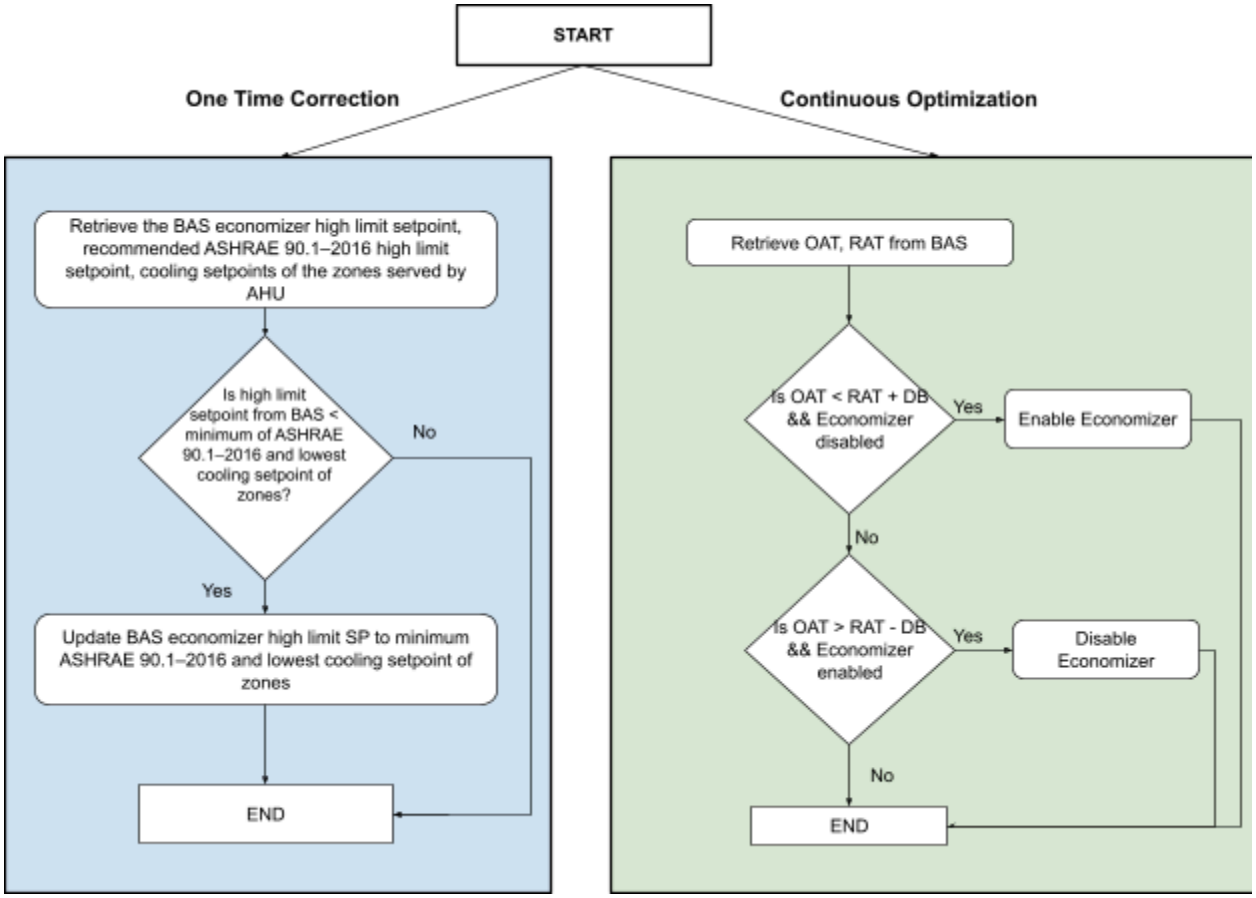


Figure 1: Flow chart of the economizer setpoint correction process

Pseudocode:

One-time correction

1. Get current economizer high limit setpoint from BAS.
2. Get ASHRAE 90.1-2016 high limit setpoint.
3. Get the return air temperature.
4. Get the lowest zone cooling setpoint of all the zones served by the AHU.
5. Based on user preference, select one:
    a. Averaged value of return air temperature (over weekly or monthly)
    b. lowest zone cooling setpoint

   Call this the ‘user-defined reference’.
6. Calculate the target high limit setpoint:
    a. target_setpoint = min (ASHRAE 90.1 high limit, user-defined reference)
7. If BAS high limit setpoint != target_setpoint
    a. Fault/opportunity identified.
    b. Push target_setpoint to BAS.

Continuous Optimization

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

At each scheduled evaluation (e.g., every 15 minutes):

1. Get the current return air temperature.
2. Get the current outdoor air temperature.
3. Get the current economizer status (ON or OFF).
4. If economizer is ON and outdoor air > return air + deadband, disable economizer.
5. Else if economizer is OFF and outdoor air < return air - deadband, enable economizer.

Summary of key variables:

Data points read from BAS:

- AHU economizer high limit lockout temperature Setpoint
- Zone cooling setpoint for all zones served by the AHU or the AHU return air temperature (AHU return air temperature can be used in lieu of the minimum zone cooling setpoint.)

Other required information:

- Economizer high limit lockout temperature setpoint based on the energy code for the climate zone and economizer control strategy being used.
- This algorithm applies only to AHUs with fixed dry-bulb economizer logic.

Data sampling/algorithm execution frequency:

- FDD algorithms for this fault are typically executed using data obtained at a sampling rate of 5–15 minutes. For one-time fault correction or diagnostic assessment, less frequent data sampling (e.g., weekly) may be acceptable. However, if continuous correction or ongoing monitoring is required, a higher sampling frequency (sampling time of 15 mins or less) is necessary to ensure timely detection and response.

Data points written to BAS/controller:

- Corrected AHU economizer high limit lockout temperature setpoint
- For one-time correction, the corrected economizer high limit lockout temperature setpoint must be written back to the BAS at a higher priority than the existing setpoint.
- For continuous correction, the system may need to actively enable or disable the economizer based on real-time conditions rather than a one-time setpoint override.

2.2 Correction of Zone Temperature Setpoints

Overview: Zone temperature setpoints are significant drivers of energy and comfort in buildings and consequently many organizations have policies defining acceptable values or ranges for these setpoints. Occupant complaints sometimes prompt site operational staff to change these values as a short-term remedy, but the modified setpoints can persist indefinitely. Portfolio energy and facility managers benefit from analytics that identify when these changes occur, and from the ability to push corrective action directly from the analytics platform to the BAS.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Algorithm Description: To detect and diagnose the presence of zone temperature setpoint faults, for each zone the actual setpoints from the BAS are read and analyzed by the EMIS tool. The EMIS is also configured with the value of each setpoint as dictated by site operational policy (best practice reference values). By comparing the actual setpoints from the BAS with operational policy setpoints configured in the EMIS tool, setpoints that do not conform to policy can be flagged as faults. Upon user authorization, setpoints that align with operational policy are pushed to the BAS using the two-way communication capabilities between the EMIS tool and BAS.

Algorithm variants

Due to the diverse nature of BAS implementations, the algorithm can be adapted based on controller configuration and point availability.

- **Direct access to control inputs:** When the control inputs used by the BAS to calculate effective heating and cooling setpoints are accessible (as shown in blue in Figure 2), the fault correction algorithm directly adjusts them by comparing them against best practice reference values (see Figure 3). These control inputs typically include:
    - Occupied zone heating setpoint
    - Occupied zone cooling setpoint
    - Unoccupied zone heating setpoint
    - Unoccupied zone cooling setpoint
- If these inputs are not explicitly available, they can be calculated by reverse-engineering the formula used for effective setpoint calculation.
- **Direct modification of effective setpoints:** Alternatively, the algorithm can directly modify the effective setpoints and/or their associated deadband values. However, caution is advised when overriding effective setpoints directly, as this approach may bypass critical BAS functionalities, such as scheduling. In this scenario, the effective setpoints are directly compared against expected user-defined values.

Setpoint assignment based on continuous optimization: While the process described above describes one-time fault correction, the same algorithm can also be implemented in a continuous optimization mode. In this mode, the EMIS tool continuously monitors zone setpoints and compares them to user-defined expectations. Unlike the static, policy-based approach above, the expected setpoint in this case can be determined dynamically based on external variables such as Outdoor Air Temperature.

Pseudocode:

One-time correction

1. Read the actual zone heating and cooling setpoints from the BAS.
2. Get the expected setpoints which are dictated by site operational policy or best practice reference values

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

3. Compare the actual setpoints to the expected setpoints:
    a. If the actual setpoints differ from the policy-based ones, flag a fault or an opportunity.
4. If the user approves, push the corrected setpoints to the BAS

Continuous Optimization

At each scheduled evaluation (e.g., every 15 minutes):

1. Read the actual zone setpoint from the BAS
2. Calculate the expected setpoint dynamically based on a reset schedule or a formula
3. Compare the actual setpoint to the expected value:
    a. If the actual setpoints differ from the expected value, flag a fault or an opportunity.
4. Push the corrected setpoints to the BAS automatically.

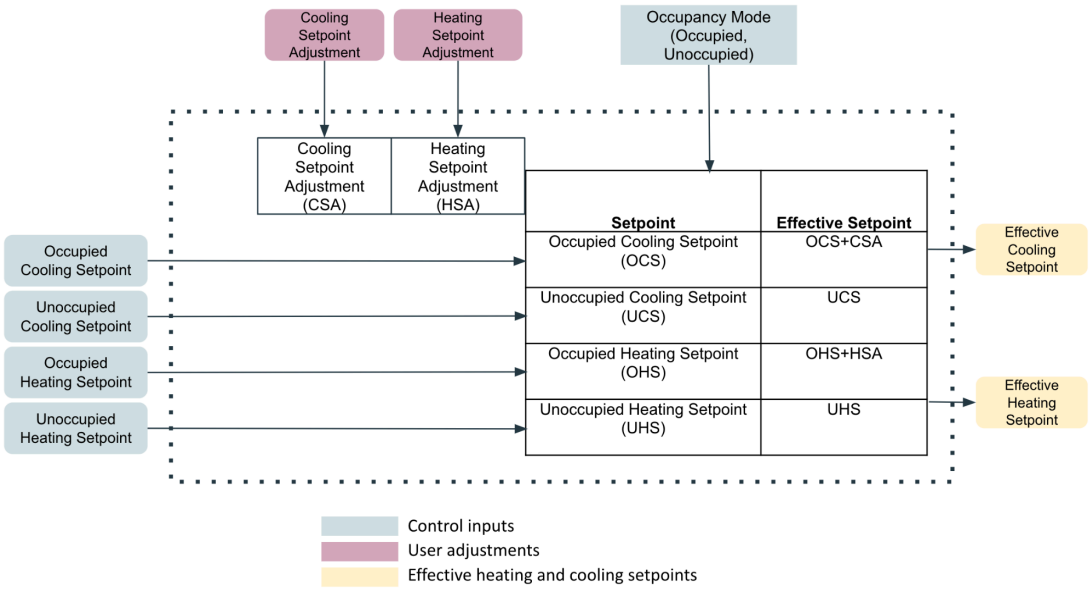


Figure 2: Figure depicting the algorithm used to determine effective setpoints for an example controller (ALC Eikon controller)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

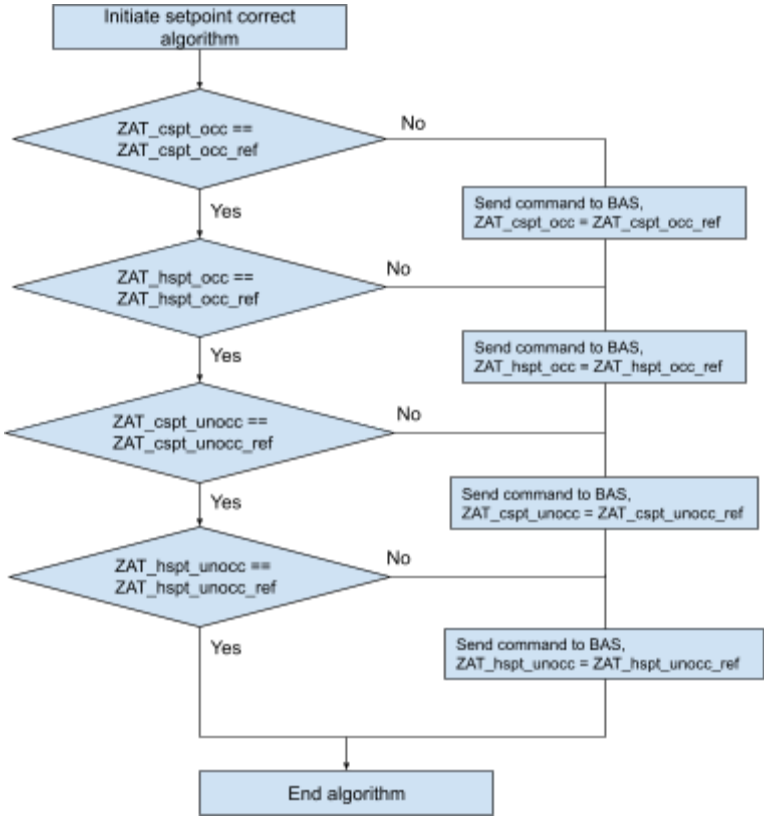


Figure 3: Flowchart describing typical implementation of zone setpoint correction algorithm - using one-time fix as an example.

Summary of key variables:

Data points read from BAS:

- Occupied Zone Heating Setpoint (ZAT_hspt_occ)
- Occupied Zone Cooling Setpoint (ZAT_cspt_occ)
- Unoccupied Zone Heating Setpoint (ZAT_hspt_unocc)
- Unoccupied Zone Cooling Setpoint (ZAT_cspt_unocc)
- Effective setpoint and deadband for cases where only calculated effective setpoints are available.

Other required information:

- User-defined (or best practice ‘reference’) values of the occupied and unoccupied zone heating and cooling setpoints.
    - Reference occupied Zone Heating Setpoint (ZAT_hspt_occ_ref)
    - Reference occupied Zone Cooling Setpoint (ZAT_cspt_occ_ref)
    - Reference unoccupied Zone Heating Setpoint (ZAT_hspt_unocc_ref)
    - Reference unoccupied Zone Cooling Setpoint (ZAT_cspt_unocc_ref)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Data sampling/algorithm execution frequency:

- FDD algorithms for this fault are typically executed using data obtained at a sampling rate of 5-15 minutes
- A less frequent data sampling (e.g., weekly) is also acceptable for one-time correction but continuous optimization would require a minimum of 15 minutes.

Data points written to BAS/controller:

- Corrected values of the following points:
    - Occupied Zone Heating Setpoint
    - Occupied Zone Cooling Setpoint
    - Unoccupied Zone Heating Setpoint
    - Unoccupied Zone Cooling Setpoint
- Corrected values written back to BAS must have higher priority than existing setpoints**.**

2.3 Release of unnecessary control overrides

Overview: Manual overrides of control signals are commonly used for troubleshooting and as a temporary remedy for operational issues. When a control signal is in “manual”, the strategy that normally determines the output to a damper, valve, or variable frequency drive is disabled and a constant value is output instead. It is not uncommon for a control system to have numerous control signals overridden at any given time, some intentional and some unintentional. The result can be increased energy use, poor indoor air quality, and other negative consequences. To correct this fault, the manual override must be identified and, upon authorization, released so that the control signal can operate in “automatic” mode.

Algorithm Description: The FDD algorithm scans the appropriate property or attribute for all control signals to identify manual overrides. Analytical logic is used to determine if a manual override has exceeded the allowable override period (e.g., 1 hr, 1 day) set by operational policy. Only those exceeding this threshold are flagged, preventing nuisance messages identifying recently enabled overrides. Once the fault is diagnosed and correction authorized, the strategy releases the override by pushing a corrected property corresponding to the manual/automatic status of the control signal to the BAS.

Depending upon the BAS, the overrides can be identified and released through two common methods:

- **Explicit override flags:** Some BAS systems include a specific point (e.g., Manual_Override) that indicates the override status of a control signal. A value of 1 typically indicates manual mode, and 0 indicates automatic mode. To release the override, the EMIS tool writes a 0 to the Manual_Override point.
- **BACnet priority array:** The overrides can be identified by monitoring the priority level of the BACnet points. If the point’s BACnet priority array at index 8 (the typical index used for operator override in BACnet based systems) has a non-null value, it indicates an overridden state. To release the override, the EMIS tool issues a BACnet write-property

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

request with a null value at priority 8 to relinquish the override, allowing the automatic control logic to resume. This method is illustrated with examples in table 1 for different HVAC components.

Table 1: Examples of override faults and resolution options.

Component type	Override point	BACnet object type and property	Indicator of override (BACnet)	Release of override (BACnet)	Examples
Fan	Fan control signal	Analog/Binary Output - Priority_Array property corresponding to fan control signal	Slot #8 of Priority_Array should have "non-null" value	"null" should be written to Slot #8 of Priority_Array	The fan on a fan-powered terminal unit might be overridden to run continuously to dilute fumes or odour or other specific needs. Consequently, the fan ran continuously for a significant period of time before it was discovered and returned to "automatic" control.
Damper	Damper control signal	Analog/Binary Output - Priority_Array property corresponding to damper control signal	Slot #8 of Priority_Array should have "non-null" value	"null" should be written to Slot #8 of Priority_Array	1. An AHU outdoor air damper may be manually overridden to a more open position to increase ventilation. 2. AHU OA damper may be overridden to the closed position due to poor air quality (e.g., due to wildfires)..
Valve	Valve control signal	Analog/Binary Output - Priority_Array property corresponding to valve control signal	Slot #8 of Priority_Array should have "non-null" value	"null" should be written to Slot #8 of Priority_Array	Operators override the VAV box reheat valve position to provide more heating.
Pump	Pump speed control signal	Analog/Binary Output - Priority_Array property corresponding to pump control signal	Slot #8 of Priority_Array should have "non-null" value	"null" should be written to Slot #8 of Priority_Array	Chilled water pump VFD control may be disabled, with pump speed fixed at maximum, due to system capacity changes following a building renovation.
Sensor	Sensor Present_value	Analog Input - Out_Of_Service to enable the overridden action; Present_Value property to assign the overridden value;	Status_Flags property has boolean flag for "Overridden" set to "T" or "1"; Out_Of_Service property (True/False)	"F" or "0" should be written to "Out_Of_Service" to release override; Default/Preferred value should be used	Operators may perceive the CO2 reading to be unimportant and override it to a nominal value to avoid replacing the sensor when it fails.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

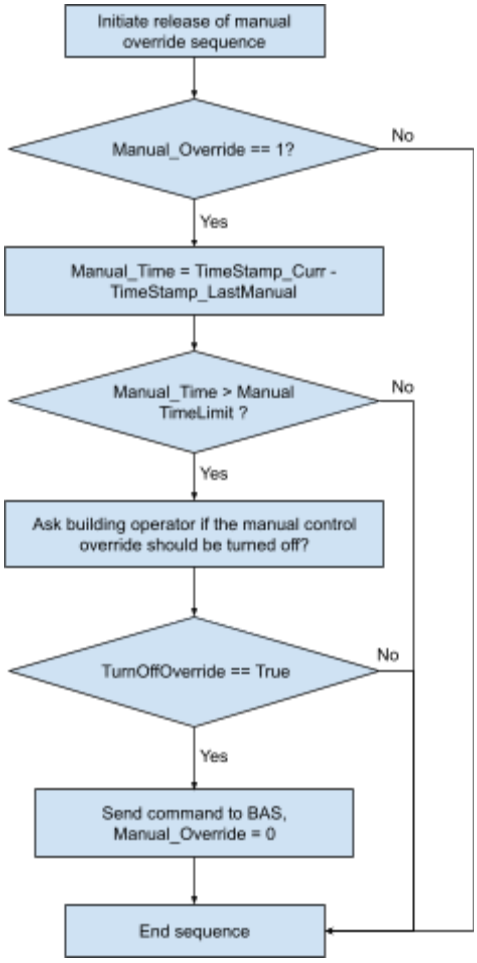


Figure 4: Flowchart describing the release of unnecessary control overrides. Manual overrides can be identified either by monitoring the BACnet object's priority array (e.g., a non-null value at priority 8) or through an explicit point in the BAS (e.g., a Manual_Override flag) indicating that the point is in manual mode (illustrated in the flow chart).

Pseudocode:

1. Scan all control signals in the building automation system (e.g., damper and valve commands).
2. Check how overrides are reported for that control signal:
    a. Look for a specific variable (like "Manual_Override") indicating the control is in manual mode.
    b. Look at the BACnet priority array, where a value at priority level 8 means someone has manually overridden the control.
3. If an override is found:
    a. Check how long it's been overridden.
    b. If it has been in manual mode longer than a defined threshold (for example, more than 8 hours), then flag as a fault.
4. If the override fault is approved for correction:
    a. For systems with a Manual_Override point: set it to 0 (meaning automatic mode).

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

b. Otherwise, send a command to clear the value at priority level 8, which releases the manual override and lets automatic control resume.

Summary of key variables:

Data points read from BAS:

- All system setpoints (AHU setpoints, VAV box and zone setpoints, plant setpoints, etc.)
- Control signals (i.e., commanded values) to valve and damper actuators, pumps, fans, etc.

Other required information:

- Data points read from BAS should have a property or attribute indicating that the setpoint is overridden or that the control signal is in “manual” or “automatic” control mode.

Data sampling/algorithm execution frequency:

- FDD algorithms for this fault are typically executed using data obtained at a sampling rate of 5-15 minutes
- Less frequent data can be used; however, unintended overrides will likely persist longer

Data points written to BAS/controller:

- The property or equivalent flag for a control signal that releases the control signal from manual control to automatic control.

2.4 Correction of Incorrectly Programmed HVAC Schedules

Overview: Schedule errors are a common fault type that cause excessive energy use due to equipment running when it isn’t needed, and that cause comfort complaints and/or indoor air quality concerns when equipment is prevented from running when it is needed. Scheduling errors may arise, for instance, when temporary schedule changes are made to accommodate a special event and are subsequently forgotten. Rooms with custom schedules, such as conference rooms and classrooms, are particularly susceptible to this type of error. Organizations can benefit from FDD and fault correction capabilities that ensure actual operating schedules conform with schedules defined by site operational policy.

Algorithm Description: Schedules defined by operational policy provide the intended times when equipment will turn on and off. By comparing the effective schedule of an equipment from the BAS with operational policy schedules configured in the FDD tool, scheduling faults can be identified. Once the fault is diagnosed and upon user authorization, a corrected schedule object (or objects) is pushed to the BAS. The simplified version of the algorithm is described in Figure 5.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

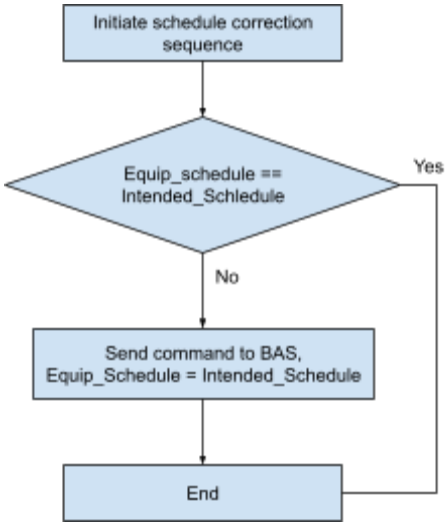


Figure 5: Flowchart describing correction of incorrectly programmed HVAC schedules

There are two methods of inferring the effective operating schedule of a piece of equipment, 1) by using recorded equipment states, and 2) by reading a BACnet schedule object. The inferred schedule of a piece of equipment is then compared to an intended schedule (the schedule defined by operational policy) and time periods where the inferred schedule and the intended schedule disagree are flagged.

Method 1 compares the effective operating schedule of an Air Handling Unit from recorded equipment states with the intended schedule defined in the FDD tool, and flags periods of disagreement. To accommodate a broad range of potential status indicators and to allow for missing data, Method 1 looks for both fan command and status feedback, as well as any other run commands and status feedbacks under the given AHU. A “union” of available inputs is used to represent the operating status of the AHU, which is then compared to the intended schedule that was configured in the EMIS tool based on operational policy. After allowing for a warm-up period (adjustable, typically 3 hours prior to scheduled occupancy), the mismatched periods are summed over a selectable analysis period, typically 2 days. If the cumulative time which the AHU is ON but should be OFF or is OFF but should be ON exceeds a user-defined time threshold, a fault is identified.

Method 2 involves directly reading and writing BACnet schedule objects. The EMIS tool reads the current BACnet schedule of equipment from the BAS. The intended schedule based on operational policy is configured in the EMIS tool. If a mismatch is found, and correction is authorized, the EMIS tool pushes the intended schedule back to the BAS by writing to the BACnet schedule object.

Pseudocode:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Determine how to infer the actual operating schedule for each piece of equipment (e.g., for each AHU) using either:

1. Method 1: Use recorded equipment status (e.g., fan status, run commands).
2. Method 2: Read the BACnet schedule object directly from the BAS

Method 1: Using Equipment Status

1. Check when the equipment was ON or OFF by combining all available signals.
2. Compare it to the intended schedule in the EMIS.
3. Calculate the cumulative time when actual and intended schedules do not match.
4. If the mismatch is more than a set limit (e.g., 5 hours in 2 days), flag a fault.
5. With user approval, push the correct schedule back to the BAS.

Method 2: Using BACnet schedule objects

1. Read the current schedule from the BAS.
2. Compare it to the intended schedule stored in the EMIS tool.
3. Calculate the cumulative time when actual and intended schedules do not match.
4. If the mismatch is more than a set limit, flag a fault.
5. With user approval, push the correct schedule back to the BAS.

Summary of key variables:

Data points read from BAS:

- Operational schedules (schedule objects) for AHUs, VAV boxes, and other equipment for weekdays, weekends, and holidays
- Fan command, fan status or any equivalent point to identify the operating status of the equipment.

Other required information:

- User-defined schedules or schedules defined by operational policy providing intended times when equipment will turn ON and OFF.

Data sampling/algorithm execution frequency:

- FDD algorithms for this fault are typically executed using data obtained at a sampling rate of 5-15 minutes
- Due to the nature of this fault, less frequent data sampling (e.g., weekly) is also acceptable

Data points written to BAS/controller:

- Corrected equipment schedule objects which are the HVAC equipment on/off times that are programmed in the BAS for weekends, weekdays and holidays.
- Corrected schedules written back to BAS must have higher priority than existing schedules

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

2.5 Automated Loop Tuning

Overview: One of the most common problems encountered with building controls is poorly tuned proportional-integral (PI) control loops. PI control loops are used to control valves, dampers, variable frequency drives, and sometimes even setpoints. If not properly tuned, the components they control will respond too aggressively or too sluggishly and, as a consequence, the process variable they control (e.g., supply air temperature) will not track its setpoint. Overly aggressive tuning is particularly problematic because it leads to oscillating or "hunting" control loops which can result in premature wear and failure of controlled components. To correct this problem, a test is performed to determine new controller parameters (i.e., proportional gain and integral time) that result in stable operation.

Algorithm Description: To detect and diagnose the presence of hunting control loops, FDD algorithms typically monitor control signals such as the commanded signal to a valve, process variables such as the supply air temperature of an AHU, or the difference between the process variable and its setpoint. The FDD algorithm analyzes time series data associated with these data points to identify oscillations in the control loop. Once the fault is diagnosed, a series of open-loop step tests is performed to determine the dynamic characteristics of the process being controlled. From these process parameters, new controller parameters are calculated which, upon authorization, are pushed to the BAS.

This fault requires a one-time correction. The corrective strategy applies to single-input single-output control loops and the form of the PI control algorithm implemented in the controller must be known to ensure the corrected control parameters are appropriate for the controller. Although data collected at 5 minute intervals are frequently used to detect and diagnose hunting control loops, sampling at 1 minute intervals is necessary for the corrective strategy to determine the dynamic characteristics of the process being tuned.

The auto-correction algorithm was developed as four separate software modules, in the FDD tool (Figure 6):

- Detection of hunting faults (3.1)
- Storage of PID parameters and test results into a database provided via a user interface (3.2)
- Management of active tests to generate data (3.3)
- Calculation of improved PID parameters (3.4) (Figure 7).

Each of the software modules is explained in more details in the following paper:
https://www.sciencedirect.com/science/article/pii/S0378778823000269

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

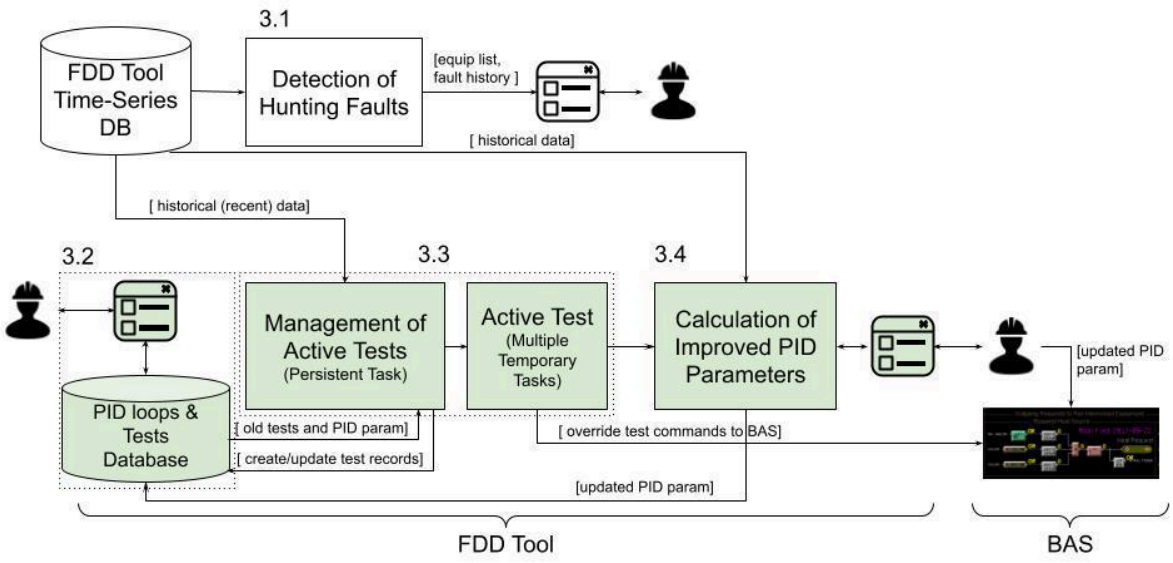


Figure 6: Software modules created (in green) or updated (in white) in the FDD tool to implement the auto-correction algorithm

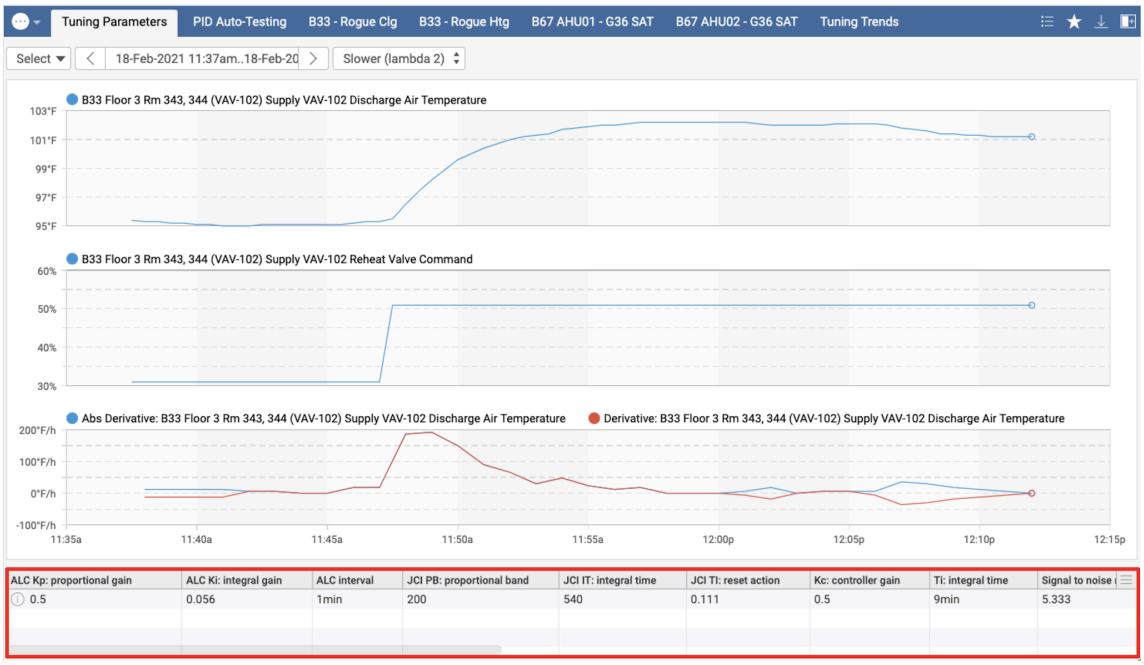


Figure 7: Snapshot of the interface depicting control and process variables (top panels) and PID parameters (in red)

Summary of key variables:

Data points read from BAS:

- AHU supply air temperature control

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

    - Control signal to the AHU cooling coil valve
    - Control signal to the AHU heating coil valve
    - Control signal to the AHU mixing box dampers
    - Supply air temperature
    - Supply air temperature setpoint
- VAV box discharge air temperature control
    - Control signal to the VAV box reheat coil valve
    - VAV box discharge air temperature
    - VAV box discharge air temperature setpoint

Other required information:

- The form of the PID control algorithm implemented in the controller must be known to ensure the corrected control parameters are appropriate for the controller.
- The self-correction algorithm applies to single input - single output control loops.

Data sampling/algorithm execution frequency:

- The data sampling rate for this fault is 1 minute. This is necessary to detect and diagnose the hunting fault and to characterize the response of the process variable during self- correction testing.
- Corrections occur after a hunting fault is detected and successful open-loop tests are performed

Data points written to BAS/controller:

- Corrected controller parameters:
    - Proportional gain
    - Integral time
- These parameters must be exposed as objects within each BACnet-certified controller so they can be overwritten.
- The setpoint for the loop being tuned must also be written back to the BAS or controller to create open-loop step changes used in the open- loop tests to determine the new control parameters.

3. Continuous Optimization Algorithms

Implementing best practice zonal reset strategies is critical to year-round operational efficiency. Nonetheless, industry experts indicate these strategies are not common in the field, and the cost and complexity of updating sequences of operation across a diversity of BAS manufacturers and vintages in a portfolio can be prohibitive. Some organizations have found value in leveraging their EMIS platform with two-way communication capabilities to push reset strategies to the BAS, enabling more efficient control sequences to be applied with far less effort and cost than would be required if they were implemented exclusively within the BAS. We have implemented AHU supply air temperature (SAT) reset and AHU static pressure reset, two zonal reset strategies, using the well-known trim and respond reset logic and rogue zone suppression. These reset strategies are driven by cooling (SAT reset) and pressure (static pressure reset) requests from the zones served by that AHU, and the rogue zone suppression

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

strategy identifies situations when persistent requests from individual zones (so-called rogue zones) should be ignored to enable the reset strategies to work as intended.

3.1 Rogue Zone Suppression

Overview: The rogue zone suppression strategy identifies zones that are adversely affecting SAT and supply air pressure reset strategies by persistently issuing cooling or pressure requests due to design or operational problems.

Algorithm Description: ASHRAE Guideline 36 defines high-performance control sequences for AHU–VAV systems. The “Trim and Respond” logic (see Section 3.2 and Section 3.3) is adopted to reset the supply air temperature and static pressure setpoints at an AHU. The adjustment of these setpoints depends on the number of “requests” generated by downstream zones that are served by the same AHU. For each time step, the change of value of the setpoint (SPchange) is determined by Equations (1) and (2) below:

$$SPchange = SPres * (R - I) \quad (1)$$

$$R = \sum IM_i \, Request_i \quad (2)$$

where SPres is a unit respond amount (e.g., 0.06 inH_20 for static pressure setpoint), R is the total number of requests from the downstream zones, I is the defined number of ignored requests,, IM is the importance multiplier that is used in the control sequence to decide if the cooling requests from the zone level should be used to control the upstream AHU, Request is the number of cooling requests from the zone, and subscript i is the indicator of the downstream zone. Therefore, if there is a rogue zone that continuously sends cooling requests whenever its schedule is on due to, for instance, zone-level equipment problems, the parameter R will always include this request, and this can force the setpoint in the control loop to its maximum cooling or maximum pressure value. Excluding rogue zones from the setpoint calculation improves operation and increases energy savings.

Two correction strategies were developed to eliminate the rogue zone impacts (i.e., to enable rogue zones to be ignored when calculating setpoint updates). The first is to overwrite I in Equation (1). The auto-correction algorithm increases I by n for each currently identified rogue zone, where n is the number of requests determined in the control sequence of that rogue zone. The second correction strategy is to overwrite the value of IM for the rogue zone in Equation (2). When the FDD tool flags the rogue zone fault, the value of IM of the rogue zone is overwritten to be zero. Therefore, the cooling requests from the rogue zone can be removed.

There are many ways to identify a rogue zone depending on the characteristics of the building. The following are two possible structured approaches to its identification:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- **Using request hours calculations:** As suggested in ASHRAE guideline 36, the Cumulative percentage of request hours is the share of the total system run hours that a single zone accounts for. If this percentage for any zone exceeds 70% and the total system run hours with Trim and Respond active exceeds 40 hours, the zone is designated as a rogue zone as described in the flowchart in Figure 8. For each rogue zone identified, either the Importance Multiplier for that zone can be set to 0, or the number of ignored requests can be increased by the number of requests the rogue zone is currently issuing.

- **Using Fault Detection and Diagnostics (FDD):** A zone is identified as a rogue zone when one or more disqualifying conditions are detected for that zone AND the zone is sending a request to the AHU. The cooling requests from the rogue zone are ignored in the calculation of total requests of all zones (i.e., the variable I is increased to account for the number of requests coming from this newly identified rogue zone). When the fault is removed, I is reset to its previous value

  Some examples of disqualifying conditions (with adjustable fault thresholds) for cooling requests include:
  - Leaky reheat valve (VAV box DAT > AHU SAT + 5°F and reheat valve is closed)
  - Supply airflow setpoint not met (<90% or >110% of setpoint and delta > 50 CFM)
  - Zone cooling setpoint too low (lower than 72°F unless exempt).

Pseudocode:

1. For each zone served by AHU, retrieve requests
2. Identify rogue zones using one of the following methods:
   Method 1: Request Hours Calculation (ASHRAE Guideline 36 approach)
   a. Keep track of how often each zone requests cooling over time.
   b. If a zone is requesting cooling more than 70% of the time, and the system has been running in trim-and-respond mode for at least 40 hours, mark that zone as a rogue zone.

   Method 2: FDD-based approach
   c. For zones currently issuing a request, check for known issues (examples below):
      i. The zone discharge air is significantly warmer than the AHU supply air , even though the reheat valve is off.
      ii. The airflow to the zone is not maintained at its setpoint.
      iii. The zone's cooling setpoint is set too low (e.g., below 72°F).
   d. If one or more of these problems are happening, label the zone as a rogue zone.
3. For each rogue zone, choose a way to ignore its cooling request:
   a. Option 1: Increase the number of ignored requests by the number of requests being issued by the rogue zone

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

b. Option 2: Set that zone's IM to zero

4. Calculate the net cooling requests based on total requests and total ignored requests (i.e. R - I)
5. Use the net cooling requests to calculate how much to change the supply air temperature (or pressure) setpoint
6. Adjust the setpoint and push it to the BAS.

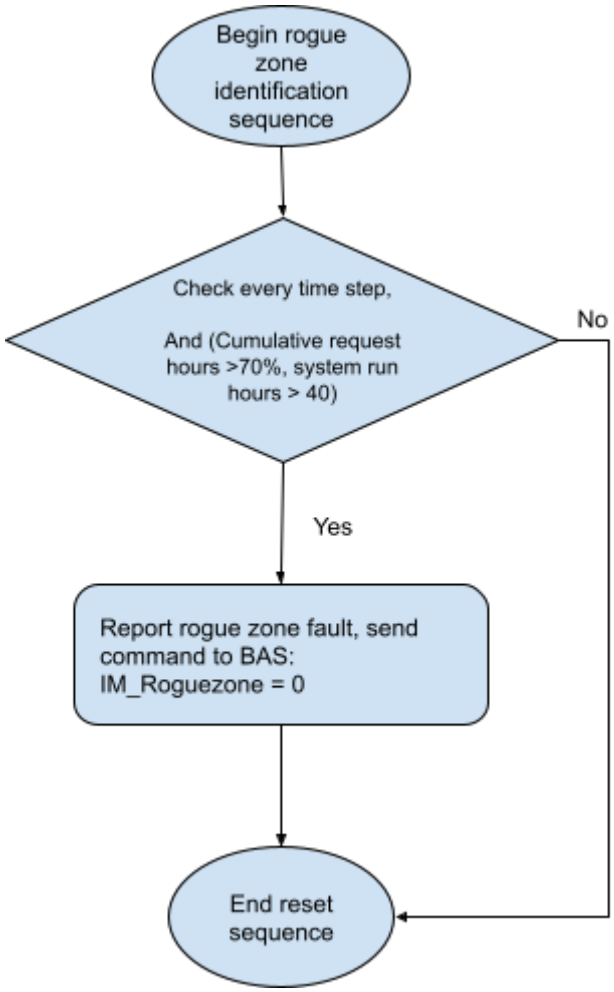


Figure 8: Flowchart describing rogue zone identification using request hours method as an example

Summary of key variables:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Data points read from BAS:

- VAV box damper control signal (and/or cooling loop output) for all zones served by the AHU
- Operational mode (Occupied / Unoccupied)
- If using request hours method:
    - IM - Importance Multiplier for each zone
    - Cumulative percentage of Request Hours for each zone.
    - Total system run hours with trim and respond active.
    - I - Number of ignored requests
- If using FDD, additional points may need to be read from BAS based on the fault rules. For example, with leaky reheat valves, VAV box DAT and AHU SAT are required.

Other required information:

- A heartbeat signal should be sent at a regular interval from the EMIS tool to the BAS to verify that two-way communication is live. If the heartbeat signal is not received at the BAS, control should revert to the original BAS strategies and setpoints.
- Cumulative Percentage of Request Hours and Total System Run Hours can be read from the BAS or calculated. If calculated, additional data from the BAS will be needed (e.g., VAV damper control signal and/or cooling loop output for each zone is commonly used to identify zones issuing requests).

Data sampling/algorithm execution frequency:

- The algorithms run continuously in the EMIS and new setpoints are pushed to the BAS using the two-way communication capabilities between the two.
- Ideally, data are read from and written back to the BAS at fixed time intervals of 2 to 5 minutes. Longer intervals can be used; however, prolonged periods with sustained cooling requests are more likely when the sampling interval exceeds 5 minutes.

Data points written to BAS/controller:

- IM - Importance multiplier (or) I - Number of ignored requests
- These values are written back only if the Trim and Respond algorithm is implemented in the BAS.

3.2 AHU Supply Air Temperature Setpoint Trim and Respond

Overview: SAT reset strategies respond to changing building loads by continually adjusting the SAT setpoint of an AHU between its minimum and maximum allowable values to ensure all zones receive adequate cooling while avoiding unnecessarily low setpoints that cause excessive chiller energy use. When the number of cooling requests exceeds an operational policy threshold, the SAT setpoint is lowered. Otherwise the setpoint is raised. When identified, rogue zones can be ignored by effectively increasing the threshold on the number of cooling requests needed to lower the setpoint.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Algorithm Description: The reset algorithm uses the ASHRAE Guideline 36 “Trim and Respond” logic for the supply air temperature setpoint. To optimize the operation of the AHU and minimize discomfort, the supply air temperature setpoint (SAT_SP) is continually reset using the Trim and Respond logic between a minimum and maximum setpoint (SPmin and SPmax). At system startup, the setpoint is the initial setpoint (SP0). The reset logic is active while the supply air fan is proven on, starting a delay timer (Td) after the initial device start command. When active, for every time step T, when the cooling request from the downstream zones (R) is less than or equal to a defined number of ignored requests (I), the setpoint is trimmed by a trim amount (SPtrim). If R is greater than I, the setpoint changes by a respond amount, (i.e., SPres * (R − I)). A maximum response per time interval is specified as SPres-max. The reset strategy is typically applied only when outdoor air temperatures are below 70 F.

The cooling requests can be calculated in different ways, of which two common methods are described below.

- Method 1: Using zone cooling demand output
    - The zone cooling demand is the output of a proportional integral (PI) control loop based on the difference between on the zone cooling setpoint ZAT_CLG_SP and the zone temperature ZAT
    - If the zone cooling demand output > 95% (adjustable), then a request is generated
- Method 2: Using only ZAT and ZAT_CLG_SP
    - One request is generated if ZAT is 0.1ºF above ZAT_CLG_SP. This request is released if ZAT falls 0.1ºF below ZAT_CLG_SP.
    - One additional request is generated if ZAT is 3ºF or more above ZAT_CLG_SP. A second additional request is generated if ZAT is 5ºF or more above ZAT_CLG_SP. If ZAT falls 0.1ºF below ZAT_CLG_SP, all the requests are removed.
    - A multiplier can be applied to all requests. The default multiplier value is default 1.

Variable	**Definition**
SP0	Supply air temperature initial setpoint
SPmin	Supply air temperature minimum setpoint
SPmax	Supply air temperature maximum setpoint
Td	Delay timer
T	Time interval at which algorithm is executed
I	Number of ignored requests

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

R	Zone cooling SAT requests
SPtrim	(+) Supply air temperature setpoint trim amount
SPres	(-) Supply air temperature respond amount
SPres-max	Maximum supply air temperature respond amount
ZAT	Zone temperature
ZAT_CLG_SP	Zone cooling setpoint

Table 2: Trim & Respond variables for supply air temperature reset

Pseudocode:

- If the AHU is enabled:
  - For each zone, calculate the number of requests
  - Sum the requests R for all the zones
  - Subtract the number of ignored requests I
  - Calculate the net number of requests R' = R - I
  - If R' <= 0
    - SAT_SP is trimmed by a fixed amount (SPtrim) provided the updated value of SAT_SP is greater than or equal to SPmin
  - If R' > 0
    - SAT_SP is increased (respond) by an amount proportional to R' (SPres with a maximum response of SPres-max) provided the updated value of SAT_SP is less than or equal to SPmax

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

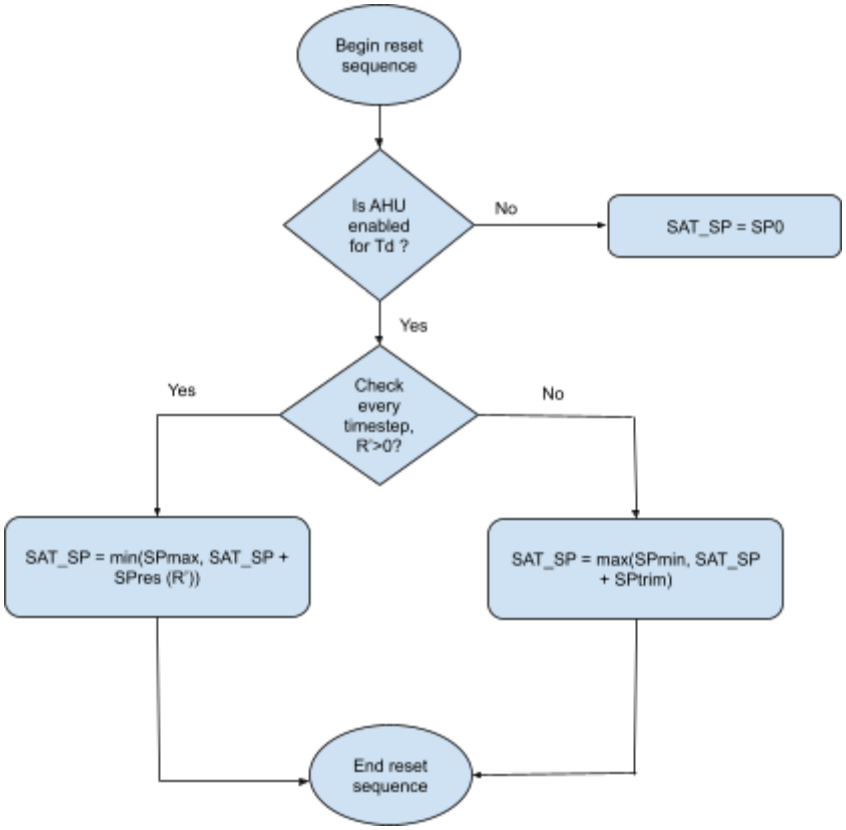


Figure 9: Flow chart describing the supply temperature reset algorithm

Summary of key variables:

Data points read from BAS:

- AHU supply air temperature setpoint
- VAV box cooling loop output for all zones served by the AHU
- Operational mode (Occupied / Unoccupied)
- The outdoor air temperature range over which the setpoint is reset
- Trim and respond strategy parameters described in Table 2

Other required information:

- A heartbeat signal should be sent at a regular interval (e.g., at the sampling interval of the reset strategy) from the EMIS tool to the BAS to verify that two-way communication is live. If the heartbeat signal is not received at the BAS, control should revert to the original BAS strategies and setpoints.

Data sampling/algorithm execution frequency:

- The algorithms run continuously in the EMIS and new setpoints are pushed to the BAS using the two-way communication capabilities between the two.
- Ideally, data are read from and written back to the BAS at fixed time intervals of 2 to 5 minutes. Longer intervals can be used; however, prolonged periods with sustained cooling requests are more likely when the sampling interval exceeds 5 minutes.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Data points written to BAS/controller:

- AHU supply air temperature setpoint
- The setpoint must be written to the BAS at a higher priority level than the setpoint determined by the BAS logic.

3.3 AHU Supply Air Static Pressure Setpoint Trim and Respond

Overview: Static pressure reset strategies respond to changing building loads by continually adjusting the static pressure setpoint of an AHU between its minimum and maximum allowable values to ensure all zones receive adequate airflow while avoiding unnecessarily high setpoints that result in excessive fan energy use. Static pressure reset strategies using trim and respond (T&R) logic make setpoint adjustments based on zone pressure requests, which are typically issued when the control signal to the VAV box damper is nearly fully open, indicating that the static pressure should be increased to ensure the current zone load can be met. When the number of pressure requests issued by the individual zones exceeds an operational policy threshold, the static pressure setpoint is increased. Otherwise the setpoint is decreased. The algorithm enables problematic or rogue zones to be ignored by effectively increasing the threshold on the number of pressure requests needed to raise the setpoint.

The specification begins with a description of the baseline ASHRAE Guideline 36 T&R (G36 T&R) sequences. Alternatives to the G36 T&R sequences, reflecting design choices that can be made through modifications of baseline T&R logic and parameters, as well as additional optimization capabilities that can augment the T&R logic, are also described.

3.3.1 Guideline 36 Trim and Respond

Algorithm Description: The G36 T&R algorithm is intended to optimize the operation of the AHU and minimize discomfort by continually resetting the static pressure setpoint (SSP_SP) between a minimum and maximum setpoint (SPmin and SPmax). At startup of the AHU, the setpoint is the initial setpoint (SP0). The reset logic is active while the supply air fan is proven on, starting a delay timer (Td) after the initial device start command. When active, for every time step T, when the number of pressure requests from downstream zones (R) is less than or equal to a defined number of ignored requests (I), the setpoint is trimmed by an amount defined by the parameter SPtrim. If R is greater than I, the setpoint responds or increases by an amount proportional to the difference between the number of requests and the number of ignored requests (i.e., SPres * (R − I)). A maximum response at each time step is specified as SPres-max.

Table 3: G36 T&R variables for supply air static pressure reset

Variable	Definition
SP0	Static pressure initial setpoint
SPmin	Static pressure minimum setpoint

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

SPmax	Static pressure maximum setpoint
Td	Delay timer
T	Time interval at which algorithm is executed
I	Number of ignored requests
R	Zone pressure requests
SPtrim	(-) Static pressure setpoint trim amount
SPres	(+) Static pressure setpoint respond amount
SPres-max	Maximum static pressure setpoint respond amount

Pseudocode:

- If the AHU is enabled:
    - For each zone, calculate the number of requests based on the VAV damper position:
        - (trigger condition) if DMP_i > 95% generate a request
        - (release condition) if DMP_i < 85% remove the request
    - Sum the requests R for all the zones
    - Subtract the number of ignored requests I
    - Calculate the net number of requests R' = R - I
    - If R' <=0
        - SSP_SP is trimmed by a fixed amount (SPtrim) provided the updated value of SSP_SP is greater than or equal to SPmin
    - If R' >0
        - SSP_SP is increased (respond) by an amount proportional to R' (SPres*R' with a maximum response of SPres-max) provided the updated value of SSP_SP is less than or equal to SPmax

A flowchart illustrating the G36 T&R logic is shown in Figure 10.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

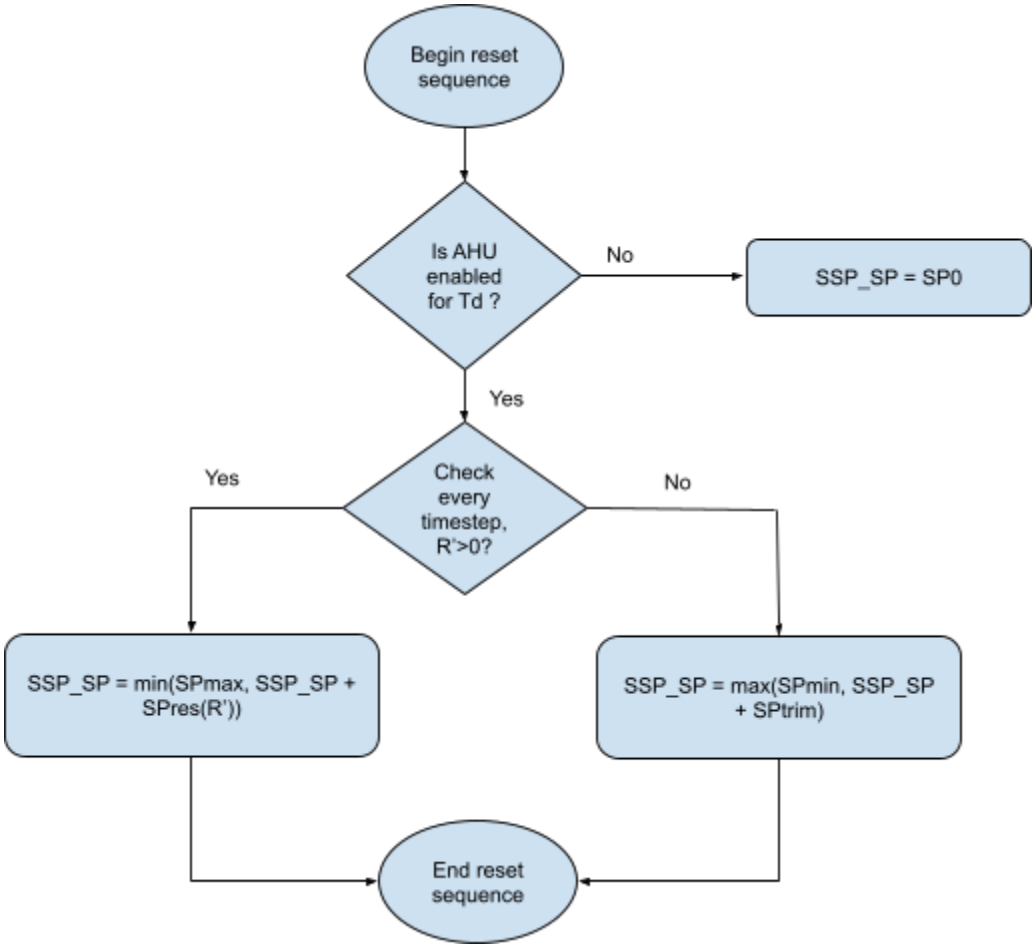


Figure 10: Flow chart describing the G36 T&R algorithm for static pressure reset.

Summary of key variables:

Data points read from BAS:

- AHU supply air static pressure setpoint
- VAV box damper control signal for all zones served by the AHU
- Operational mode (Occupied / Unoccupied)
- Trim and respond strategy parameters described in Table 3

Other required information:

- A heartbeat signal should be sent at a regular interval (e.g., at the sampling interval of the reset strategy) from the EMIS tool to the BAS to verify that two-way communication is live. If the heartbeat signal is not received at the BAS, control should revert to the original BAS strategies and setpoints.

Data sampling/algorithm execution frequency:

- The algorithms run continuously in the EMIS and new setpoints are pushed to the BAS using the two-way communication capabilities between the two.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Ideally, data are read from and written back to the BAS at fixed time intervals of 2 to 5 minutes. Longer intervals can be used; however, prolonged periods with sustained pressure requests are more likely when the sampling interval exceeds 5 minutes.

Data points written to BAS/controller:

- AHU supply air static pressure setpoint
- The setpoint must be written to the BAS at a higher priority level than the setpoint determined by the BAS logic.

3.3.2 Guideline 36 Trim and Respond Variant

Algorithm Description: The G36 T&R variant logic modifies the baseline G36 T&R logic by introducing a request deadband within which no changes are made to the static pressure setpoint. The intent is to reduce the frequency of changes to the setpoint to create a more stable setpoint profile. For this variant, the static pressure setpoint responds (is increased) when R' = R - I > R'res ≥ 0. The setpoint is trimmed (decreased) when R' < R'trim ≤ 0. If R'trim ≤ R' ≤ R'res, the setpoint is unchanged. R'res and R'trim are adjustable parameters that can be used to expand or shrink the deadband. The remainder of the algorithm is consistent with the baseline G36 T&R logic.

Table 4: Additional variables for implementing G36 T&R variant for supply air static pressure reset

Variable	Definition
R'trim	Threshold number of net requests (R - I) below which the setpoint is trimmed (decreased)
R'res	Threshold number of net requests (R - I) above which the setpoint responds (increases)

Pseudocode:

- **If the AHU is enabled:**
  - **For each zone, calculate the number of requests based on the VAV damper position:**
    - **(trigger condition) if DMP_i > 95% generate a request**
    - **(release condition) if DMP_i < 85% remove the request**
  - **Sum the requests R for all the zones**
  - **Subtract the number of ignored requests I**
  - **Calculate the net number of requests R' = R - I**
  - **If R' > R'res**
    - **SSP_SP is increased (respond) by an amount proportional to R' (SPres*R' with a maximum response of SPres-max) provided the**

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

updated value of SSP_SP is less than or equal to SPmax

- **If R' < R'trim**
  - **SSP_SP is trimmed by a fixed amount (SPtrim) provided the updated value of SSP_SP is greater than or equal to SPmin**
- **If R'trim ≤ R' ≤ R'res**
  - **SSP_SP is unchanged from the previous time step**

A flowchart illustrating the G36 T&R Variant logic is shown in Figure 11.

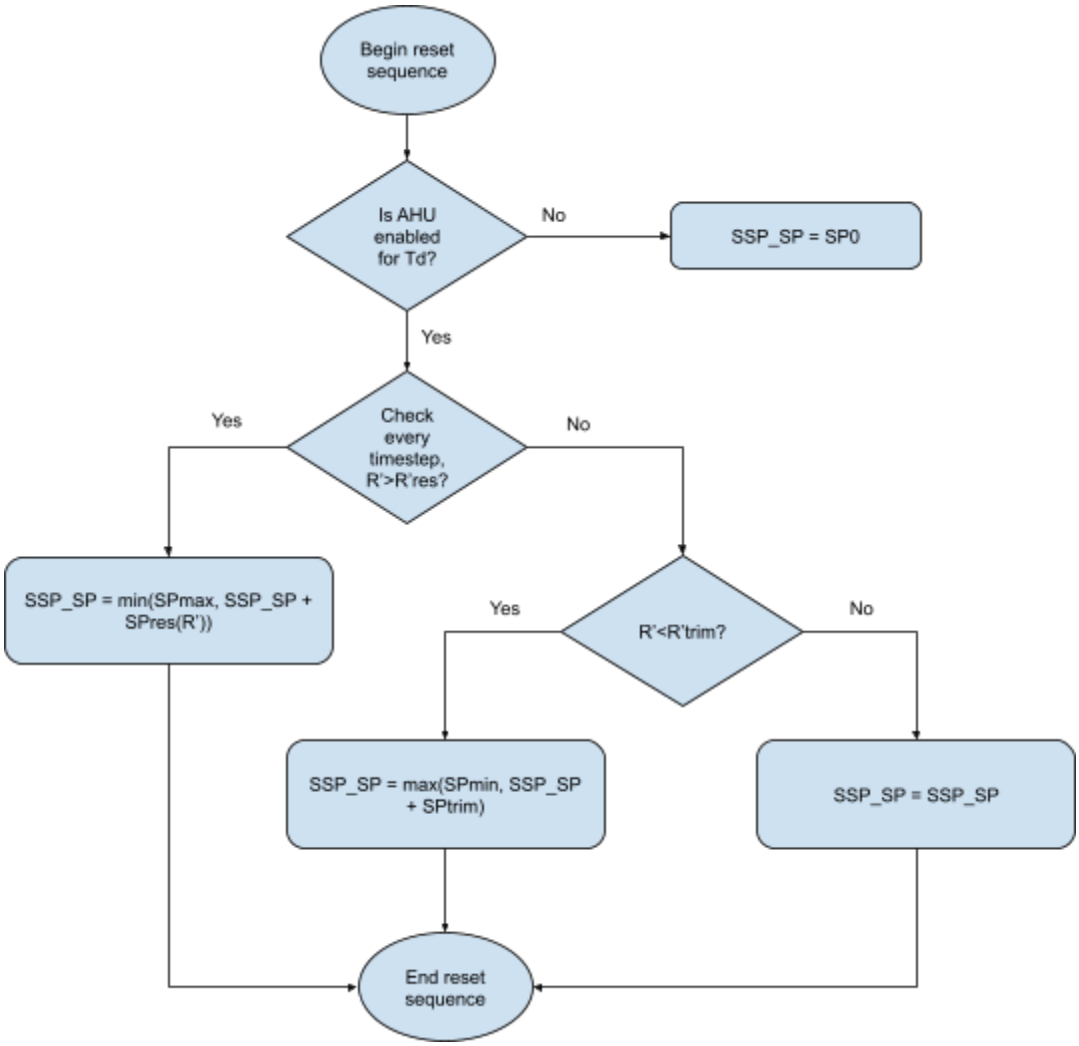


Figure 11: Flow chart describing the Guideline 36 Variant T&R algorithm for static pressure reset.

Summary of key variables:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Variables in Table 4 and all data and information list in “Summary of key variables” section of 3.3.1.

3.3.3 Hybrid Trim and Respond

Algorithm Description: The Hybrid T&R logic implements an alternative strategy that runs in parallel to the baseline G36 T&R logic. Here we assume the alternative strategy determines a predictive optimization profile for the static pressure setpoint subject to constraints. A predictive algorithm could allow you to identify optimal trajectories to move the AHU towards the most efficient operating point as opposed to the reactive, incremental nature of G36’s T&R logic. At each time step, both the predictive optimization strategy and the G36 T&R strategy calculate an update to the static pressure setpoint. If a feasible solution to the optimization problem is determined, that setpoint is used. Otherwise the G36 T&R update of the setpoint is used. The constraints on the predictive optimization strategy include those implemented by G36 T&R, such as limiting the setpoint to operate between SPmin and SPmax, and limiting the maximum rate of response of the setpoint. Predictive optimization represents one alternative strategy that could be implemented in the Hybrid T&R logic. A flowchart illustrating the Hybrid T&R logic is shown in Figure 12. Note that the G36 T&R logic from Figure 10 is depicted as a single component on the right side of this flowchart. Further details of the predictive optimization strategy are not presented here.

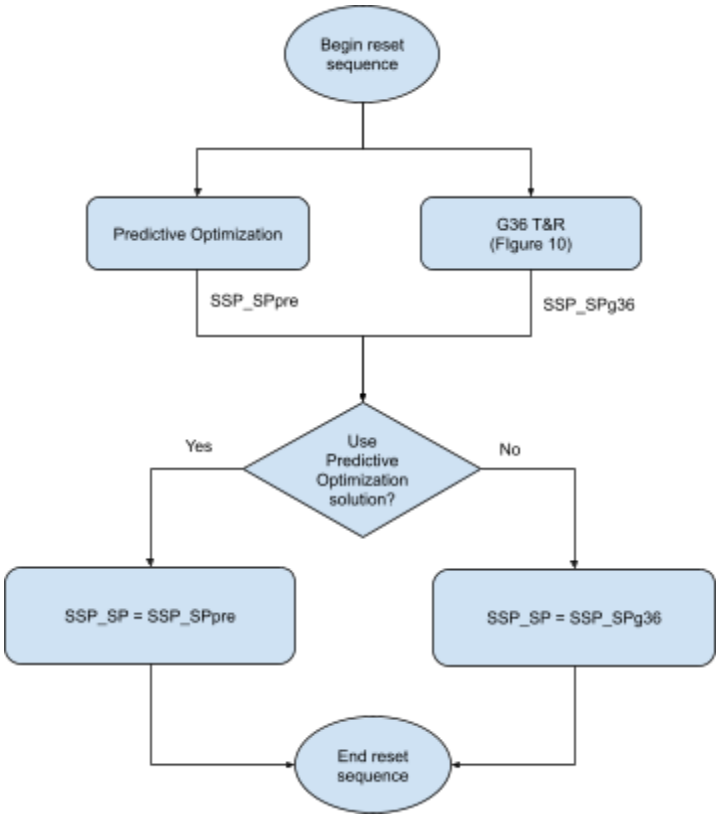


Figure 12: Flow chart describing the Hybrid T&R algorithm for static pressure reset.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4. Demand Flexibility Algorithms

These algorithms implement demand flexibility strategies in the EMIS tool that shed or shift electricity demand. The algorithms require continuous real-time communication between the FDD tool and BAS to facilitate demand reduction and prevent occupant discomfort. The following algorithms have been developed and are described in Demand Flexibility Strategies: Fan and Valve Limiting and Demand Flexibility Strategies

- Zone temperature adjustment
- Zone temperature adjustment for precooling with VAV systems
- Zone temperature adjustment for precooling with single-zone rooftop units
- Chilled water supply temperature setpoint reset
- Supply fan and cooling coil valve limiting

4.1 Zone-level Strategy: Zone Temperature Adjustment

Figure 13 shows a process flow diagram of zone temperature adjustment strategy. It should have four components:

For each zone in white list for every timestep:

(1) ZONE QUALIFICATION CHECK
   - Verify runaway conditions (too hot)
   - Verify starved conditions (no air)
   - Verify classification of rogue zone
   - Is this zone classified as “hands-off”
   - Release setpoints for zones not in the list

(2) SHED STRATEGY
   - Apply zone temperature adjustment

(3) RATCHET STRATEGY
   - Sort all zones in white list by deviation to Zone Temperature Adjustment (ZTA) setpoint
   - Zones closer to new setpoint will ratchet up to extend shed or gradually add zones based on a demand target

(4) POST-EVENT STRATEGY (rebound management)
   - Prioritization by sorting all zones in white list by deviation to ZTA setpoint
   - Staggered release of zones

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

This specification includes four variants of zone temperature-based strategies, depicted in Figure 13 and requires a set of variables and user defined-parameters as summarized in Table 5.

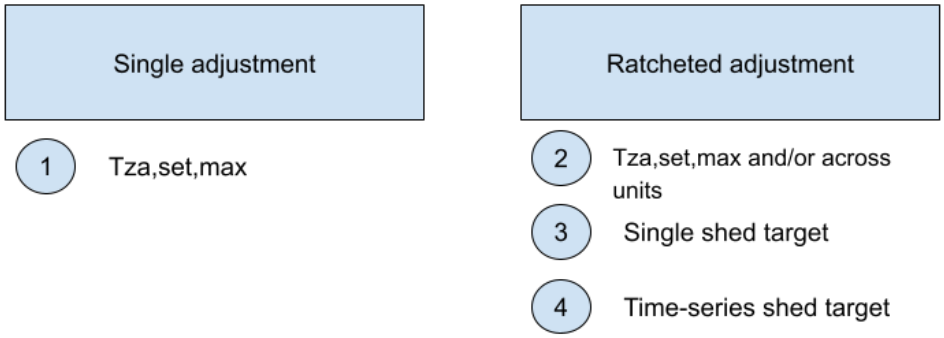


Figure 13: Illustration of the four variants of zone temperature adjustment strategies that are included in the demand flexibility specification.

In all variants a zone temperature setpoint limit (maximum and/or minimum) is a parameter defined by the user. In Variant-1, the zone temperature setpoints are adjusted in a single step. Variant-2 uses a ratchet across HVAC units, or temperature setpoints. That is, either all qualified zone temperature setpoints are adjusted over incremental steps (up to but not exceeding the maximum temperature limit), or HVAC units themselves are incrementally adjusted to the setpoint limit (up to but not exceeding the total number of qualified units). Variants 3 and 4 introduce a performance-based load target, in which temperature setpoints or number of zones/units are ratcheted until the target change in load is met. For Variant-3, the target may be a single absolute load (kW) for the duration of the flexibility period, or in Variant-4, the target may be a time series of different load values (for example, if the site is tracking performance relative to a historic baseline).

These variants accommodate a wide set of use cases that span the most common HVAC system configurations, diverse grid service program designs, site demand flexibility goals, and operational practices and preferences. For example, a setpoint-based ratchet on a collection of staged-RTU systems will eventually result in a plateaued load reduction, while a ratchet based on the number of adjusted units is more appropriate for continuous load reduction. Considering operational practices and preferences, a single-step adjustment may be useful for an owner who prioritizes simplicity. For sites that seek to participate in demand flexibility programs that involve bidding or penalties, or that prioritize achieved load reductions for other reasons, a target-based ratchet may be ideal - it allows the site to deepen and extend a load shed across longer time durations. In general, zone temperature setpoint adjustment can be implemented to pre-cool zones prior to the response period, in order to deepen the achieved load reduction, or it can be used exclusively during the response period.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

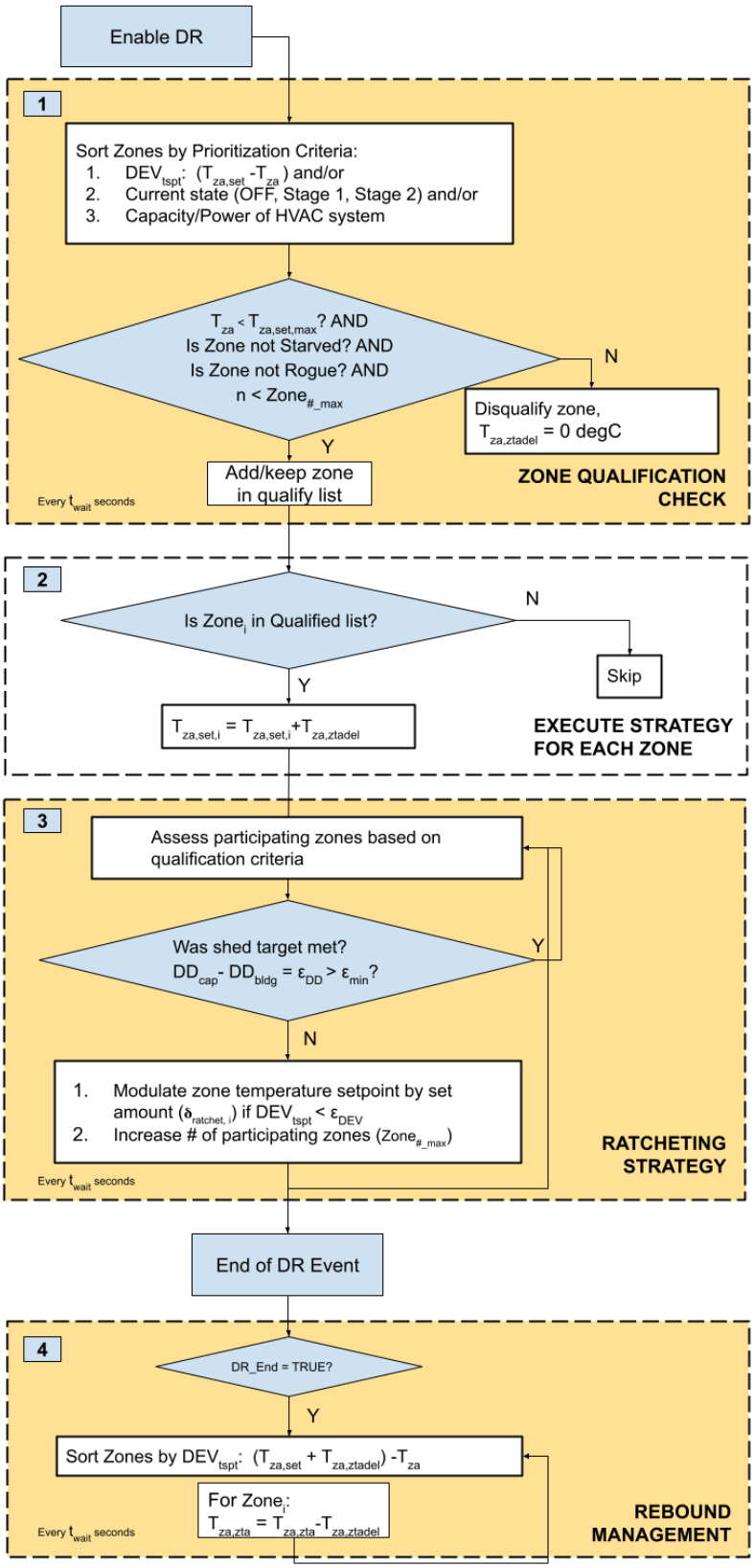


Figure 14: Zone temperature adjustment strategy

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.1.1 Required data points

Table 5: Inventory of parameters and variables

Abbreviation	Data point	Data type	Unit	Requirements (Defined/Calculated/ Read/Write)
$T_{za,set,min}$	Minimum allowable zone air temperature setpoint	User-defined parameter	°C	Defined
$T_{za,set,max}$	Maximum allowable zone air temperature setpoint	User-defined parameter	°C	Defined
n	# of zones on qualified list	Algorithm variable	-	Defined
$Zone_{\#_max}$	Maximum # of zones allowed on qualified list	User-defined parameter, Algorithm variable	-	Defined
DD_{cap}	objective electrical demand decrease	User-defined parameter	kW	Defined
ε_{min}	Minimum allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter	kW	Defined
t_{wait}	Wait time to determine conditions for ratcheting	User-defined parameter	seconds	Defined
ε_{DEV}	Threshold to trigger temperature ratcheting to maintain shed	User-defined parameter	°C	Defined
$T_{za,set,\ def}$	Default zone air temperature setpoint for all zones	User-defined parameter	°C	Defined
$T_{za,ztadel}$	Adjustment factor for ZTA strategy	User-defined parameter	°C	Defined/Write
$\delta_{ratchet,\ i}$	Zone-specific incremental delta for ratcheting zone temperature setpoint	User-defined parameter	°C	Defined/Write
DD_{bldg}	Building demand decrease	Time-series sensor reading from BAS/meter	kW	Calculated

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

ε_{DD}	Calculated difference between DD_{cap} and DD_{bldg}	Algorithm variable	kW	Calculated
DEV_{tspt}	Zone temperature deviation from effective setpoint	Algorithm variable	°C	Calculated
T_{za}	Zone air temperature	Time-series sensor reading from BAS	°C	Read
P_{bldg}	Electric demand from building meter	Time-series sensor reading from BAS	kW	Read
$T_{za,set}$	Effective zone air temperature setpoint for all zones	Time-series sensor reading from BAS	°C	Read/Write
$T_{za,set,i}$	Effective zone air temperature setpoint for each zone ($Zone_i$), which is equal to $T_{za,set}$ during regular hours, and is equal to a different value during shed and precool periods	Time-series sensor reading from BAS	°C	Read/Write

4.1.2 Zone prioritization and qualification check

All participating zones or units should be prioritized by one of or a combination of relevant characteristics such as cooling capacity, current deviation from setpoint (DEV_{tspt}) or current state (off, stage 1 or stage 2) in order to ensure the most comfortable zones or largest capacity units are called upon first in the event. The units that are prioritized are then sequentially qualified to make sure that the units or zones are not above the $T_{za,set,max,}$ threshold or faulted in any way. The qualified units will only be added to the active list if the max number of zones called upon has not been reached. Once $Zone_i$ is in the active list, the setpoint is adjusted to shut the compressor off in Function 2 as shown in Figure 16.

At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as thermal runaway and VAV starving conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. The thermal runaway check will examine values for T_{za} for instances of thermal runaway: determining whether or not T_{za} is above the comfort threshold $T_{za,set,max}$. The sequence may also check VAV damper command values and VAV airflow for starving zones. Finally, other disqualifying conditions may be factored in such as rogue zone classification. The qualified units will only be added to the active list if the max number of zones called upon has not been

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

reached. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle as illustrated in Figure 15.
At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as thermal runaway and VAV starving conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. The thermal runaway check will examine values for T_{za} for instances of thermal runaway: determining whether or not T_{za} is above the comfort threshold $T_{za,set,max}$. The sequence may also check VAV damper command values and VAV airflow for starving zones. Finally, other disqualifying conditions may be factored in such as rogue zone classification. The qualified units will only be added to the active list if the max number of zones called upon has not been reached. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle:

```
#Read all T_za,i from zone_i in zone_list:
      # If T_za,i < T_za,set,max &
          #zone_i not {Starved} &
         # zone_i not {Rogue Zone}&
         #n < Zone_#max:
            # qualify_list.append(zone_i)
      #else Continue
```

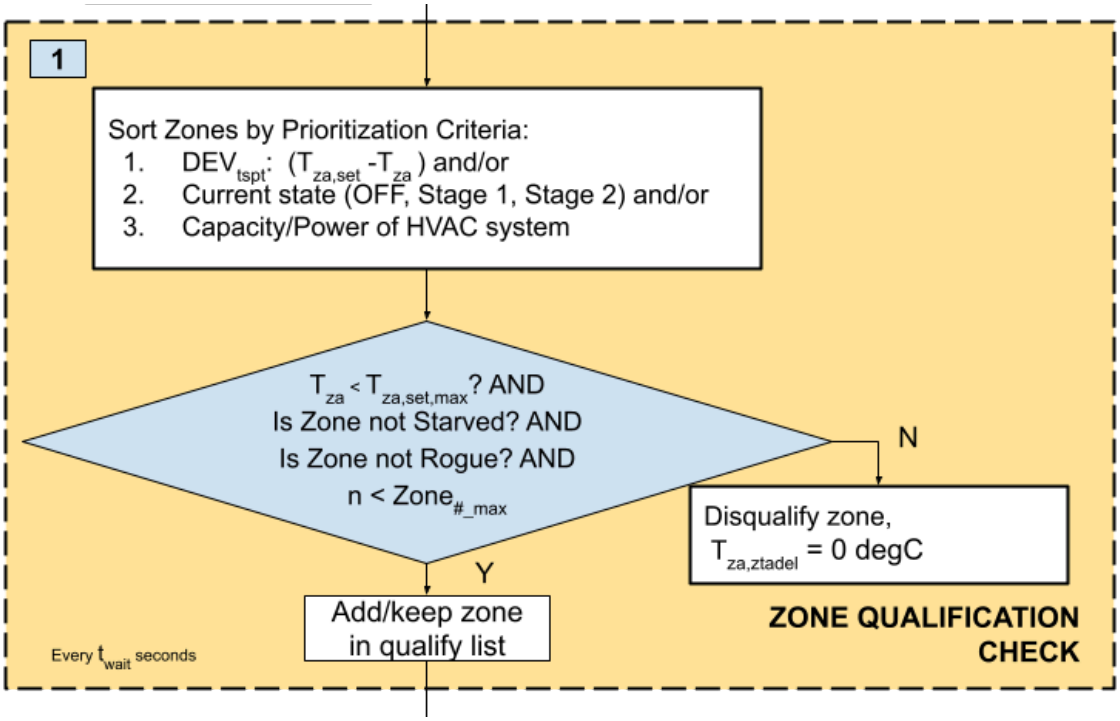


Figure 15: Logic for zone prioritization and qualification checks

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.1.3 Shed strategy

The qualified units will only be added to the active list if the max number of zones called upon has not been reached. Once $Zone_i$ is in the active list, the setpoint is adjusted to shut the compressor off or drive the VAV to minimum in Function 2 as shown in Figure 16 by adjusting setpoints $T_{za,set}$ by a predetermined initial offset value, or a redefined value for $T_{za,ztadel,i}$ (e.g. 2F-4F) or $T_{za,ztadel,i}$ + $\boldsymbol{\delta_{ratchet,\ i}}$, for each ratcheted zone. The event strategy will then monitor the zone air temperature T_{za} at regular intervals in order to determine whether or not the zone can be further adjusted to maintain shed or if conditions in each zone have reached unmanageable conditions that could lead to thermal comfort issues. At each logical cycle, the ratcheting and occupant comfort checks are explained in more detail in the following sections.

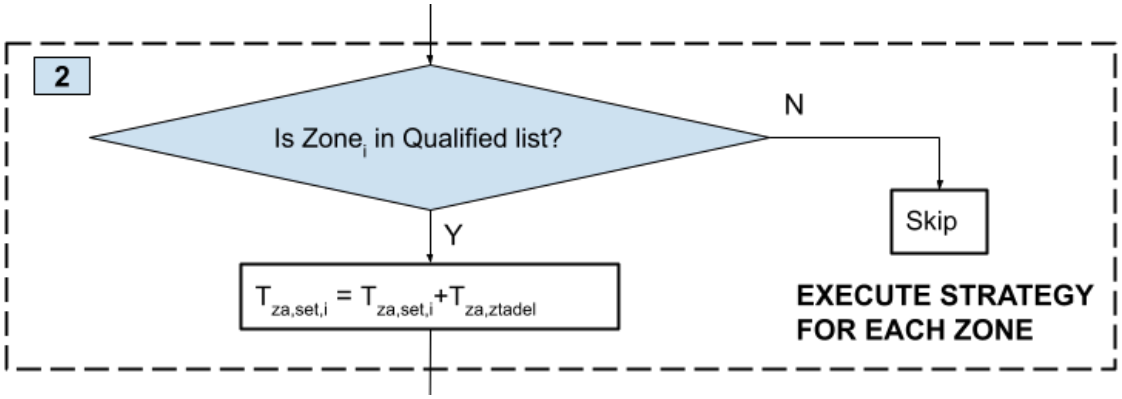


Figure 16: Logic for ZTA strategy once pre-event check is completed

```
# For zones in qualify_list:
      # T_za,set_i= T_za,set+T_za,ztadel,i
# For zones in zone_list & not in qualify_list:
      # T_za,set_i= T_za,set
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.1.4 Ratcheting Sequence: Incremental Shed Preserving Temperature Ratcheting

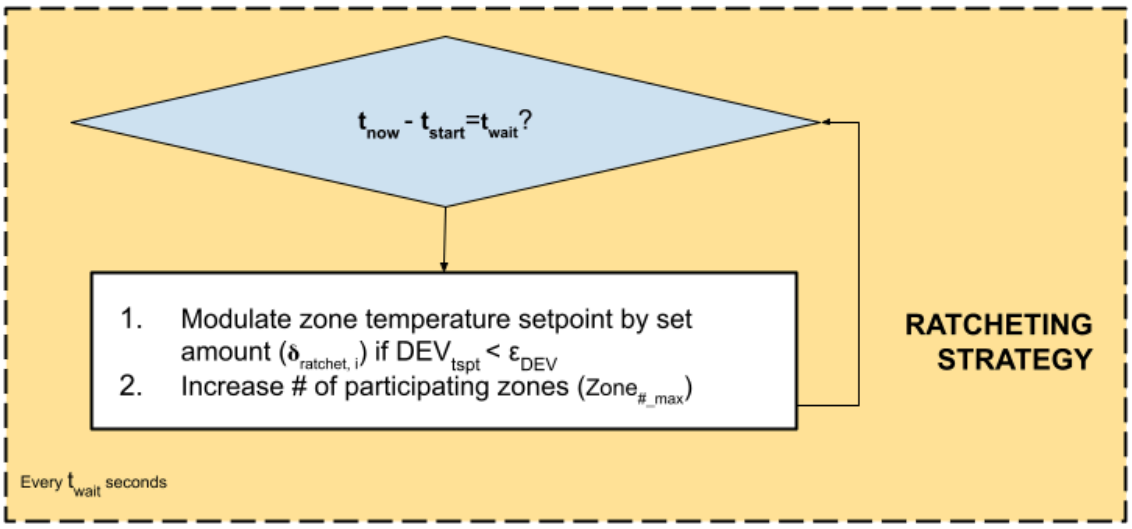


Figure 17: Incremental temperature ratcheting logic

This ratcheting sequence represents Variant-2 in which there is room for sustained shed by incrementally adjusting the zone temperature setpoint to preserve shed or elongate the “float” time and maintain lower cooling demand. This use case may address building operator concerns of thermal inertia associated with driving the new setpoint close to a comfort limit and allows for safer, incremental changes to the building operation. This assumes there is an acceptable temperature range between the original zone temperature adjustment and a customer determined maximum comfort-based temperature threshold. Once the event starts and the zone temperature adjustment is after a defined wait period, the ratcheting sequence determines zone temperature setpoints that may be further adjusted in order to maintain or preserve demand shed based on comfort limits and current zone conditions. These zones would be displayed to an operator to approve ratcheting. This is achieved by sorting the current zone temperature readings relative to the adjusted zone temperature adjusted setpoint to determine setpoint deviation, DEV_{tspt} as defined by Equation #1:

Eq#1: $DEV_{tspt,i} = (T_{za,set} + T_{za,ztadel,i}) - T_{za,i}$

All zone temperatures that are below a certain threshold of the original adjusted setpoint will be further adjusted by an increment manually determined by the operator or automatically using a dynamic algorithm. This ratcheting strategy may run continuously at a specified frequency, with the expectation that a new set of zones will cross the deviation threshold and have their

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

setpoints adjusted. The relationship between these setpoints to the increasing zone air temperature is displayed in Figure 18. The ratcheting sequence is superseded by the zone qualification check, which releases zones that are approaching uncomfortable conditions for occupants defined as $T_{za,set,max}$. The following pseudocode along with Figure 17 summarizes the above approach:

```
#Wait(t_wait seconds)
#Read all T_za,i:
#DEV_stpt_i= (T_za,set + T_za,ztadel,i) -T_za,i
#Sort all T_za,i by Dev_stpt_i
For all T_za,i :
#If DEV_stpt_i < error_dev
      #T_za,ztadel,i= T_za,ztadel,i+δ_ratchet,i
#else Continue
```

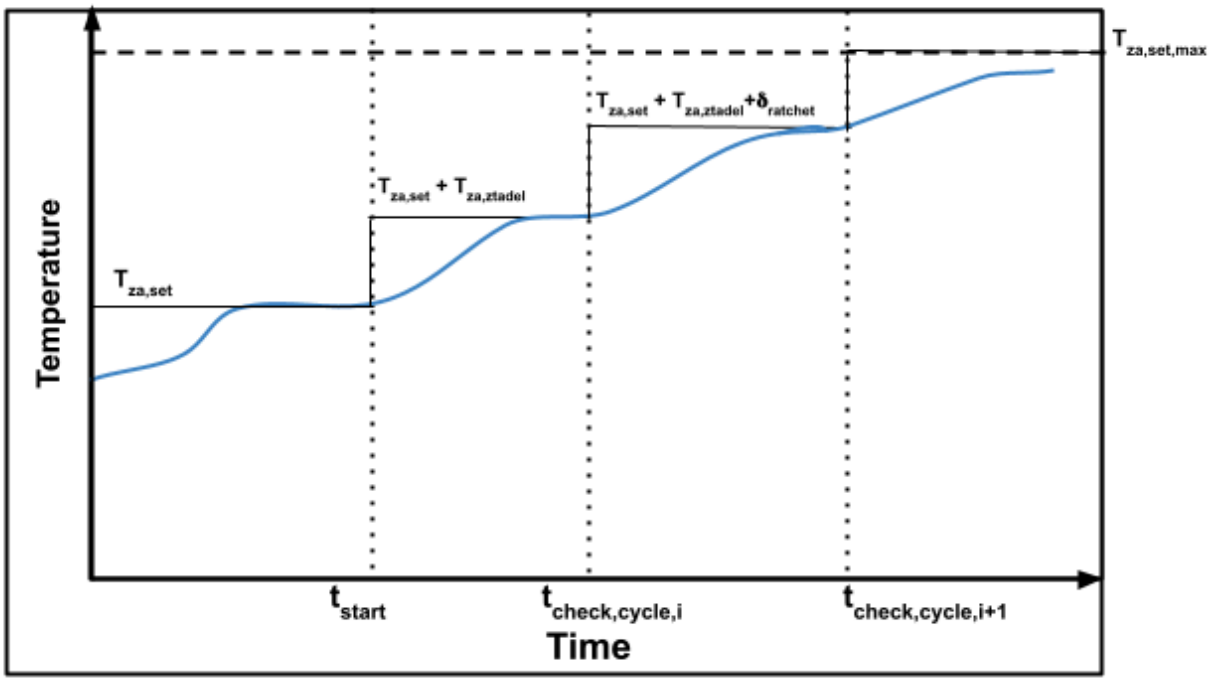


Figure 18: Illustration of how and when the zone temperature setpoint would be increased

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.1.5 Ratcheting Sequence: Automated/Manual Performance Target Based Ratcheting

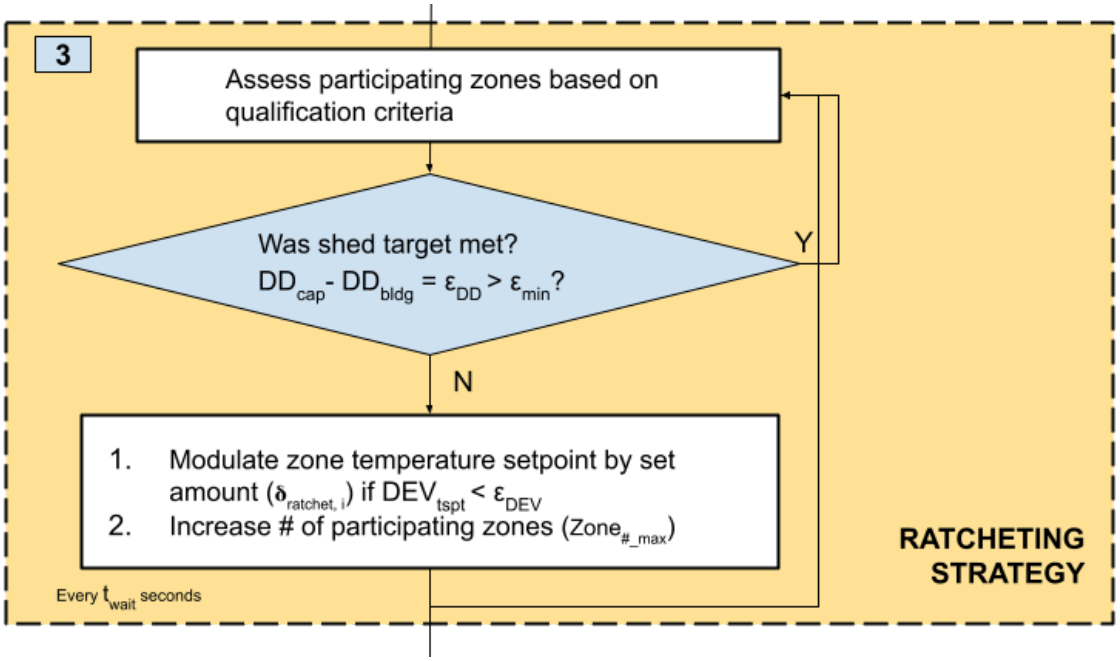


Figure 19: Logic for performance based ratcheting strategy

As established for Variant 3 and 4, it is possible to implement load shed in a 'closed' loop manner, in order to meet a specific shed target, for example a % demand reduction, an absolute kW reduction. (Provided that the target must not exceed the capacity of the building and its systems). Under this paradigm, ratcheting is employed as a quality assurance mechanism. The building demand reduction may be quantified using a simple reference, or by using a DR-program aligned baseline as the reference :

1. Single load shed target (Variant 3): Measure the change in electric load after the shed strategy has been enabled compared to the demand prior to initiating the shed
2. Time-series shed target (Variant 4): Measure the load reduction of the building at each time step compared to a baseline model constructed from historic building load data.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

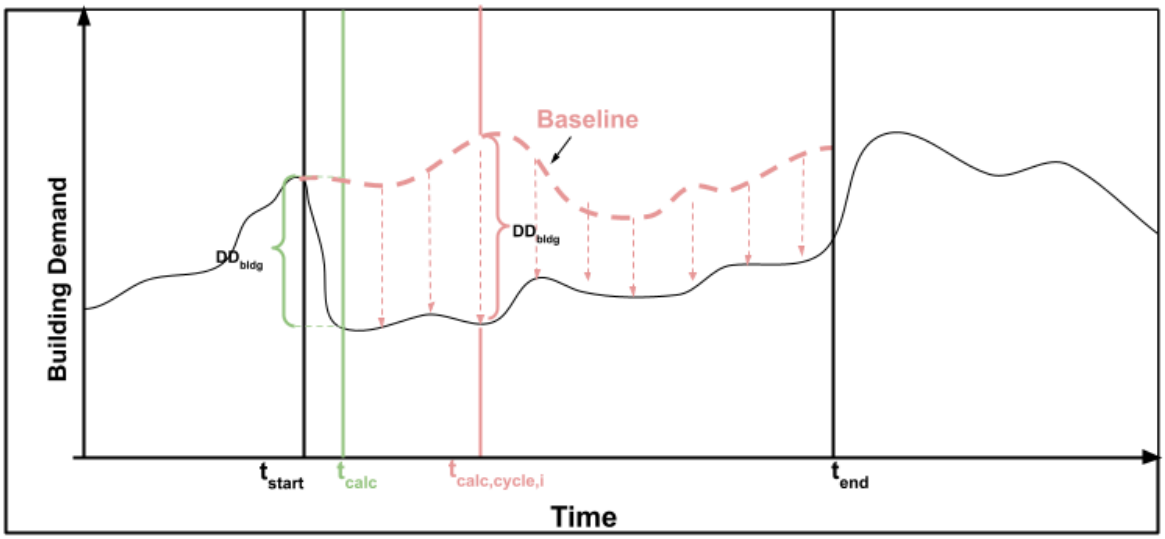


Figure 20: Illustration of two different calculation methods for DD_{bldg}

Assuming the demand decrease of the building can be measured applying the two above methods, the following equation governs the degree of error and need for ratcheting:

DD_{cap}-DD_{bldg} = ε_{DD}

The calculated error in demand decrease will reference a set of additional measures that can be taken by the building to minimize the error. In the case of a zone temperature adjustment, the damper positions of white listed zoned or the calculation of $DEV_{tspt,i}$ may be leveraged to determine which zones have the capacity to have their temperature setpoints further offset.

```
#Wait(t_wait seconds)
#ε_DD=DD_cap-DD_bldg
#if   ε_DD > ε_min
      #Read all T_za,i in qualify_list:
      #DEV_stpt_i= (T_za,set + T_za,ztadel,i) -T_za,i
      #Sort all T_za,i by DEV_stpt_i
      #OR
      #Sort all zones by u_vdm
      For all T_za,i :
            #If DEV_stpt_i < 0.1
                  # T_za,ztadel,i= T_za,ztadel,i+δ_ratchet,i
#else Continue
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.1.6 Post-event strategy (Rebound management)

These logical processes cycle at a frequency defined by t_{wait} until the DF event has concluded. After the DF event has concluded, a rebound management function, Function 4 in Figure 21 is triggered. In order to avoid a rebound effect in which a new peak is created in building demand, the zone temperature adjustments are released to pre-event values sequentially by sorting all participating zones by degree of deviation from new setpoint $DEV_{tspt,i}$. As zones are released, the setpoints are gradually returned back to their default value, $T_{za,set,\ def}$ by an interval defined by $\boldsymbol{\delta}_{ratchet,\ i}$ at the logical cycle frequency, $t_{wait,}$.

```
#if DR_Enable = FALSE
      #Sort all T_za,i by DEV_stpt_i
            #For all T_za,i :
                  #Wait (t_wait)
                  #T_za,ztadel,-1= 0
```

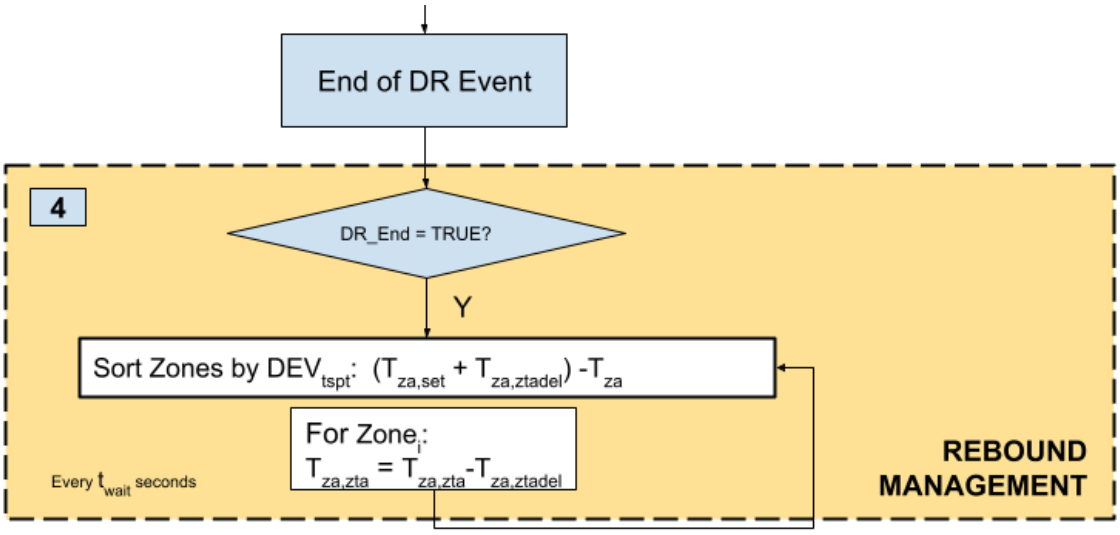


Figure 21: Rebound management control sequence

4.2 Zone-level Strategy (AHU, MZ-VAV): Zone Precooling

PARAMETER DEFINITION

- Calculate length and depth of Precool before event:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Linear relationship between time delta and depth of precool (shorter precool means deeper precool)
- OAT could be an input to precool setpoint calculation

ZONE QUALIFICATION
- Qualify zones
  - Verify setpoint > min setpoint *(precool)*
  - Verify starved conditions (no air)
  - Verify zone is not rogue
  - Verify if zone has been precooled
  - Add to list of VAVs that can participate
  - Release setpoints for VAVs not in the list

PRECOOL STRATEGY (initialization + strategy at each cycle)
- For each zone in white list for every timestep:
  - Sort all zones in white list by deviation to precool setpoint
  - Apply setpoints for pre-cooled zones
  - Release setpoints for zones not in the list

MODULATION OF DEMAND BASED VIA MAX VAVS (demand peak mitigation)
- Compare current demand and monthly peak demand
  - If difference is greater than min threshold, add 1 to max VAV #
  - If difference is too close to min threshold, subtract 1 to max VAV #

4.2.1 Simple Approach

4.2.1.1 Required data points

Table 6: Required Data Points from BAS (controllers)

Abbreviation	Data point	Data type	Requirements (Read/Write)
T_{za}	Zone air temperature	Time-series sensor reading from BAS	Read
$\dot{v}_{vav,\,i}$	VAV box volumetric flow	Time-series sensor reading from BAS	Read
u_{vdm}	VAV box damper control signal	Time-series sensor reading from BAS	Read
$T_{za,set}$	Effective zone air temperature setpoint for all zones	Time-series sensor reading from BAS	Read/Write

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$T_{za,set,i}$	Effective zone air temperature setpoint for each zone ($zone_i$), which is equal to $T_{za,set}$ during regular hours, and is equal to a different value during shed and precool periods	Time-series sensor reading from BAS	Read/Write

Table 7: New parameters and defined threshold in the DF algorithm

Abbreviation	**Data point**	**Data type**
$T_{za,pcdel}$	adjustment factor for precool strategy	Algorithm variable
DEV_{tspt}	Zone temperature deviation from effective setpoint	Algorithm variable
$T_{za,set,min}$	Comfort-based minimum zone air temperature setpoint	User-defined parameter
$\dot{v}_{vav,max,i}$	zone-specific VAV box volumetric flow max setpoint	User-defined parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

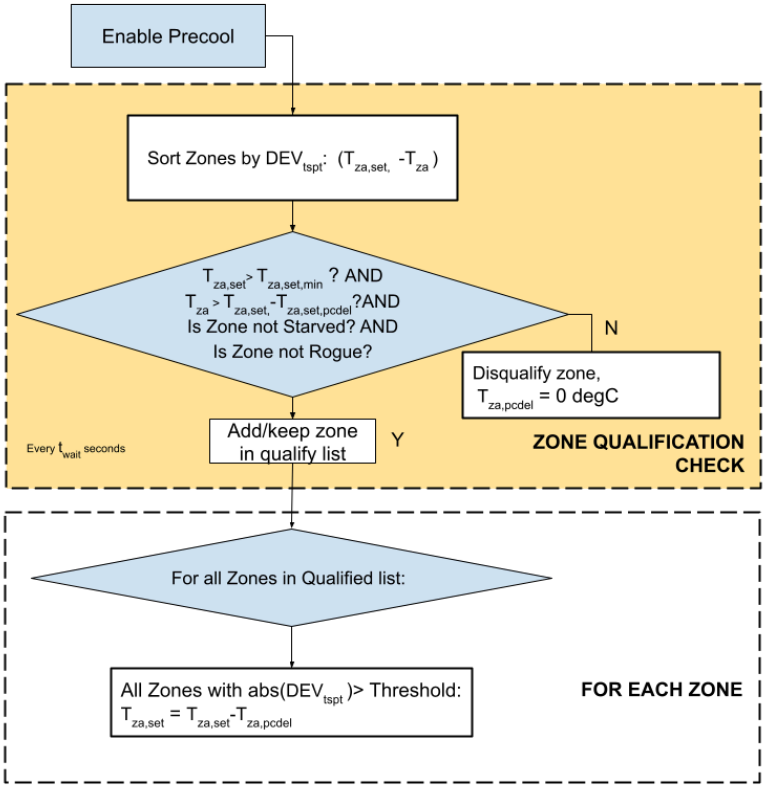


Figure 22: Zone qualification check for precool strategy

4.2.1.2 Parameter definition

The depth and timing of precooling strategies may be constant values in simple cases, but in order to accommodate dynamic goals in certain DR programs, a control strategy might be in place. Assuming a day ahead signal and notice of duration of the shed event, the strategy could take into account these inputs along with other measurements such as predicted outdoor air temperatures to set the time at which precooling starts ($\mathbf{t_{start}}$) as well as how low the precool temperature is set ($\mathbf{T_{za,pcdel}}$).

4.2.1.3 Zone prioritization and qualification check

Zones being considered for precooling, should be prioritized based on how warm each zone is relative to current setpoint (driving demand), in order to create more potential during shed strategy. The logic then examines all zones and focuses on those that will bring most benefit during the shed event, by removing problematic zones, zones that are already too cold and prioritizing warmer zones. At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as rogue zone status, zone conditions that are at or below the precool threshold or VAV starving conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

The thermal comfort check will examine values for T_{za} values, determining whether or not T_{za} is below the comfort threshold. The sequence will also check VAV damper command values and VAV airflow for starving zones. Finally, other disqualifying conditions may be factored in such as rogue zone classification. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle:

```
#Read all T_za,i in zone_list:
      # If T_za,i > T_za,set,min &
    # Zone_i not {Starved}&
      # Zone_i not {Rogue Zone}:
           # qualify_list.append(Ti)
      #else Continue
```

4.2.1.4 Precool Strategy

Next comes the execution of the precool sequence, in this case it would start with prioritization of zones by sorting all qualifying zones by calculating the deviation between the current zone temperature and the current zone temperature setpoint and adjusting qualifying zone setpoints $\mathbf{T_{za,set}}$ by a predetermined initial offset value, or a redefined value for $\mathbf{T_{za,pcdel}}$ (e.g. 2F to 4F).

```
#Read all T_za,i in qualify_list:
#DEV_stpt_i= (Tza,set-Tza,pcdel,i) - Tza,i
#Sort all T_za,i by DEV_stpt_i
For all T_za,i :
      #If DEV_stpt_i > 3 degC
           # Tza,set,i=Tza,set,i-Tza,pcdel
#else Continue
```

4.2.2 Complex Approach

4.2.2.1 Required data points

Table 8: Required Data Points from BAS (controllers)

Abbreviation	Data point	Data type	Requirements (Read/Write)
T_{za}	Zone air temperature	Time-series sensor reading from BAS	Read
$\dot{v}_{vav,\,i}$	VAV box volumetric flow	Time-series sensor reading	Read

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		from BAS	
u_{vdm}	VAV box damper control signal	Time-series sensor reading from BAS	Read
$T_{za,set}$	Effective zone air temperature setpoint	Time-series sensor reading from BAS	Read/Write

Table 9: New parameters and defined threshold in the DF algorithm

Abbreviation	Data point	Data type
$T_{za,ztadel}$	adjustment factor for ZTA strategy	Algorithm variable
DEV_{tspt}	Zone temperature deviation from effective setpoint	Algorithm variable
$T_{za,set,min}$	Comfort-based minimum zone air temperature setpoint	User-defined parameter
$\dot{v}_{vav,max,i}$	zone-specific VAV box volumetric flow max setpoint	User-defined parameter
P_{bldg}	building power consumption	Time-series sensor reading from BAS
$P_{peak_monthly}$	Monthly peal building power	Algorithm variable
ε_0	Calculated difference between P_{bldg} and $P_{peak_monthly}$	Algorithm variable
ε_{min}	Minimum allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter
db_{ε}	Deadband value for ε_{min}	User-defined parameter
$VAV_{\#_max}$	Maximum # of VAVs allowed on qualified list	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

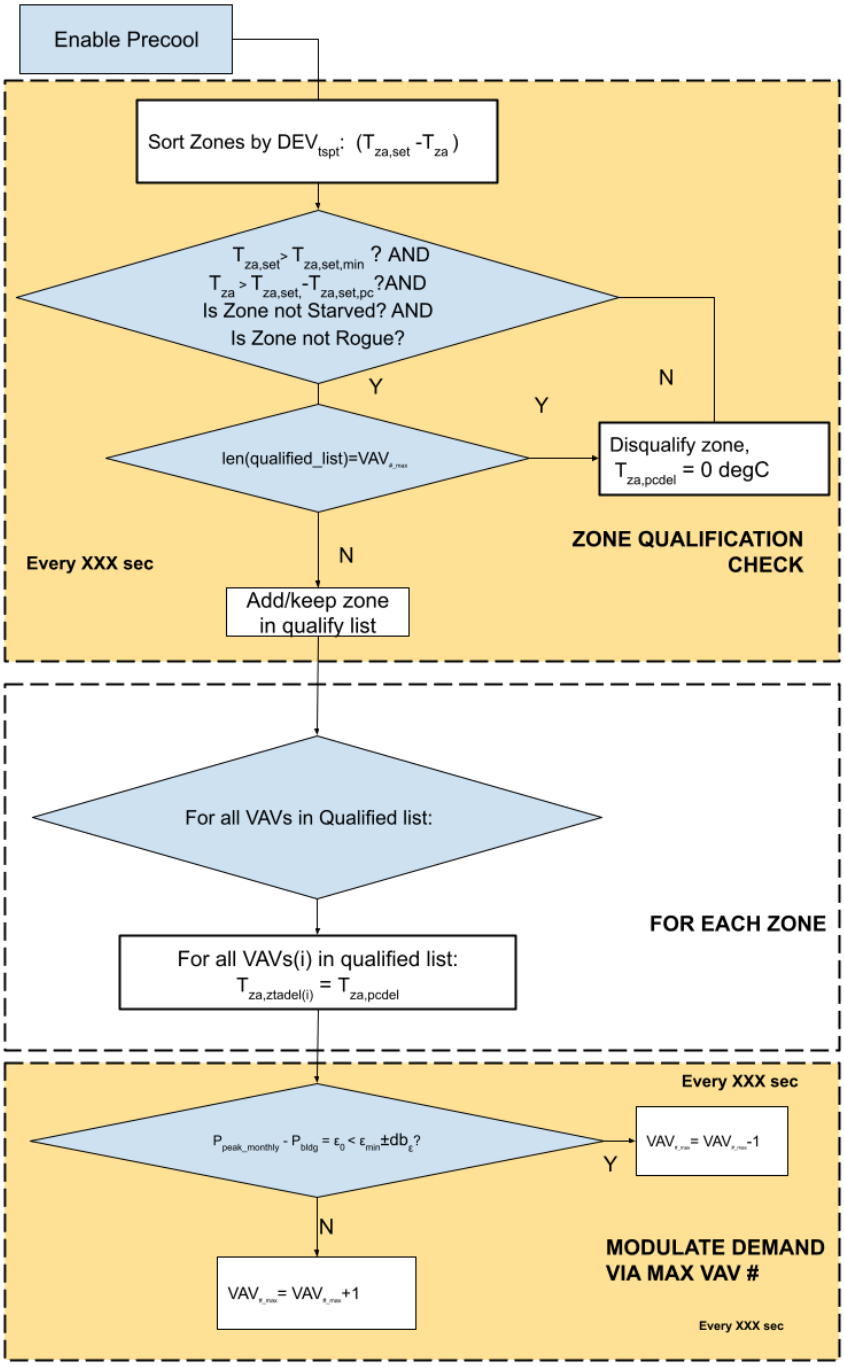


Figure 23: Zone level pre-cool strategy - complex approach

4.2.2.2 Parameter definition

The depth and timing of pre cooling strategies may be constant values in simple cases, but in order to accommodate dynamic goals in certain DR programs, a control strategy might be in place. Assuming a day ahead signal and notice of duration of the shed event, the strategy could

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

take into account these inputs along with predicted outdoor air temperatures to set the time at which precooling starts ($\mathbf{t_{start}}$) as well as how low the precool temperature is set ($\mathbf{T_{za,pcdel}}$).

4.2.2.3 Zone prioritization and qualification check

For precooling, the logic first examines all VAVs and focuses on those that will bring most benefit during the shed event, by removing problematic equipment, zones that are already too cold or zones that have recently been precooled. At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as faulted status, zone conditions that are at or below the precool threshold or other comfort conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. Existing faults in the VAV that could be classified as disqualifying may include any that would negatively affect occupant comfort or the ability for the VAV to meet setpoint, this could include undercharge or fouling faults, staging faults, or programmatic faults to the thermostat. The thermal comfort check will examine values for $\mathbf{T_{za}}$ values, determining whether or not $\mathbf{T_{za}}$ is below the comfort threshold. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle. In addition to these initial checks, there is another demand peak prevention condition that determines whether or not the qualified list has reached its maximum allowable number of running units. In this case, the maximum number of VAVs is determined by calculating the difference between the current monthly peak and the current demand of the building. If the number of activated VAVs is equal to the max, a qualifying VAV is skipped until the number falls below maximum:

```
#Read all VAVs in VAV_list:

#Sort all Tza,i by DEV_stpt

#If:
#    VAVi not {precooled} in last X cycles AND
#     Tza,seti > Tza,set,min  AND
#     Tzai > Tzai,set,min AND
#     Zone/VAVi not {faulted}
# If len(qualify_list)= VAV#_max:
      # Continue
# else:
      # qualify_list.append(Tzai)
```

4.2.2.4 Precool strategy

The qualification check is followed by the execution of the precool sequence, in this case it would start with prioritization of zones by sorting all qualifying zones by an individual or combined list of prioritization criteria. This includes the following criteria:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Deviation of current zone temperature from current set point: (DEVtspt =Tza,set -Tza)

As an example, the zones could be prioritized by calculating the deviation between the current zone temperature and the current zone temperature setpoint and adjusting qualifying zone setpoints $T_{za,set}$ by a predetermined initial offset value, or a redefined value for $T_{za,pcdel}$ (e.g. 2F-4F).

```
#Read all T_za,i in qualify_list:
#DEV_stpt= (Tza,set-Tza,pcdel) - Tza,i
For T_za,i :
# Tza,set,i=Tza,set,i-Tza,pcdel
```

4.2.2.5 Demand modulation via max VAV limit

In cases where the building is avoiding a spike in demand from simultaneous activation of VAVs, the complex form of this strategy actively monitors the current demand (assuming this measurement is available at frequent intervals) of the building and compares it to the established monthly peak. As long as the demand does not exceed the minimum error, ε_{min} the maximum number of activated VAVs incrementally increases by 1. In the case that the demand gets too close to the new peak, less than ε_{min} , the max number of VAVs decreases.

```
# If P_peak_monthly - P_bldg = ε_0 > ε_min ± db_ε
# VAV#_max= VAV#_max-1
# else
# VAV#_max= VAV#_max+1
```

4.3 Zone-level Strategy (SZ-RTU): Zone Precooling

PARAMETER DEFINITION

- Calculate length and depth of Precool before event:
  - Linear relationship between time delta and depth of precool (shorter precool means deeper precool)
  - OAT could be input to precool setpoint calculation

ZONE QUALIFICATION

- Qualify zones
  - Verify setpoint > min setpoint *(precool)*
  - Verify zones are not violating comfort conditions (temp, CO2)
  - Verify zone /RTUs not faulted

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Verify if max number of RTUs in qualified list has been reached (demand limiting strategy)
- Verify if zone has been precooled
- Add to list of RTUs that can participate
- Release setpoints for RTUs not in the list

PRECOOL STRATEGY (initialization + strategy at each cycle)

- For each RTU in white list for every timestep:
  - Sort all RTUS in qualified list by one or multiple factors including:
    - Zone type/priority
    - DEVtspt: (Tza,set -Tza)
    - Current state (OFF, Stage 1, Stage 2)
    - Capacity/Power of RTU
    - Estimated rate of change of cooling load
  - Apply pre cool setpoints sequentially to each RTU
  - Return precooled zones to original setpoints to avoid frequent activations (disqualify by attaching “recently activated” tag that may expire after XXX cycles or seconds)

MODULATION OF DEMAND BASED VIA MAX RTUS (demand peak mitigation)

- Compare current demand and monthly peak demand
  - If difference is greater than min threshold, add 1 to max RTU #
  - If difference is too close to min threshold, subtract 1 to max RTU #

4.3.1 Simple approach

4.3.1.1 Required data points

Table 10: Required Data Points from BAS (controllers)

Point	Description	Type	**Requirements (Read/Write)**
T_{za}	Zone air temperature	Time-series sensor reading from BAS	Read
$T_{za,set}$	Effective zone air temperature setpoint	Time-series sensor reading from BAS	Read/Write

Table 11: New parameters and defined threshold in the DF algorithm

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Abbreviation	Data point	Data type
$T_{za,pcdel}$	adjustment factor for ZTA strategy	Algorithm variable
$T_{za,set,min}$	Comfort-based minimum zone air temperature setpoint	User-defined parameter

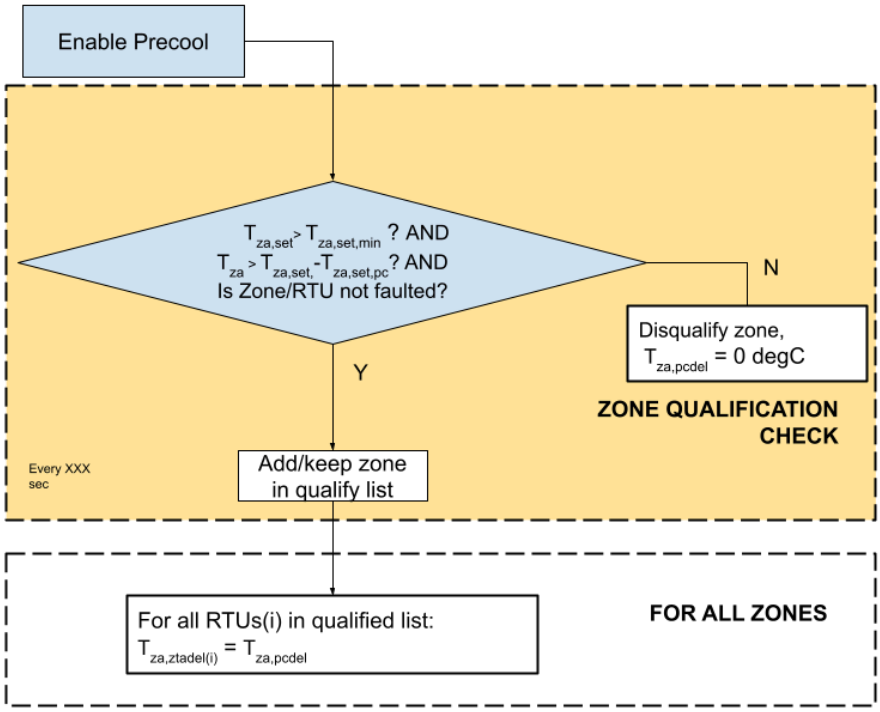


Figure 24: Zone-level pre-cooling for rooftop units (RTU) - simple approach

In this use case, all qualifying RTUs are precooled at once and activated without any regard to new demand spikes as a result of simultaneous operation.

4.3.1.2 Parameter definition

The depth and timing of precooling strategies may be constant values in simple cases, but in order to accommodate dynamic goals in certain DR programs, a control strategy might be in place. Assuming a day ahead signal and notice of duration of the shed event, the strategy could take into account these inputs along with predicted outdoor air temperatures to set the time at which precooling starts ($\mathbf{t_{start}}$) as well as how low the precool temperature is set ($\mathbf{T_{za,pcdel}}$).

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.3.1.3 Zone prioritization and qualification check

For precooling, the logic first examines all RTUs and focuses on those that will bring most benefit during the shed event, by removing problematic equipment, zones that are already too cold or zones that have recently been precooled. At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as faulted status, zone conditions that are at or below the precool threshold or other comfort conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. Existing faults in the RTu that could be classified as disqualifying may include any that would negatively affect occupant comfort or the ability for the RTU to meet setpoint, this could include undercharge or fouling faults, staging faults, or programmatic faults to the thermostat. The thermal comfort check will examine values for $\mathbf{T_{za}}$ values, determining whether or not $\mathbf{T_{za}}$ is below the comfort threshold. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle:

```
#Read all RTUs in RTU_list:
#If:
#    RTU_i not {precooled} in last X cycles AND
#     Tza,set_i > Tza,set,min  AND
#     Tza_i > Tza_i,set,min AND
#     Zone/RTU_i not {faulted}
#qualify_list.append(Ti)
#else: Continue
```

4.3.1.4 Precool strategy

The qualification check is followed by the execution of the precool sequence, in this case it would activate all qualifying RTUs at once. The strategy adjusts qualifying zone setpoints $T_{za,set}$ by a predetermined initial offset value, or a redefined value for $T_{za,pcdel}$ (e.g. 2F to 4F). This new setpoint will persist until the precool strategy is disabled.

```
#Read all T_i in qualify_list:
For T_i :
# Tza,set,i=Tza,set,i-Tza,pcdel
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.3.2 Intermediate approach

4.3.2.1 Required data points

Table 12: Required Data Points from BAS (controllers)

Point	Description	Type	**Requirements (Read/Write)**
T_{za}	Zone air temperature	Time-series sensor reading from BAS	Read
$T_{za,set}$	Effective zone air temperature setpoint	Time-series sensor reading from BAS	Read/Write

Table 13: New parameters and defined threshold in the DF algorithm

Abbreviation	**Data point**	**Data type**
$T_{za,pcdel}$	adjustment factor for ZTA strategy	Algorithm variable
DEV_{tspt}	Zone temperature deviation from effective setpoint	Algorithm variable
$T_{za,set,min}$	Comfort-based minimum zone air temperature setpoint	User-defined parameter
$RTU_{\#_max}$	Maximum # of VAVs allowed on qualified list	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

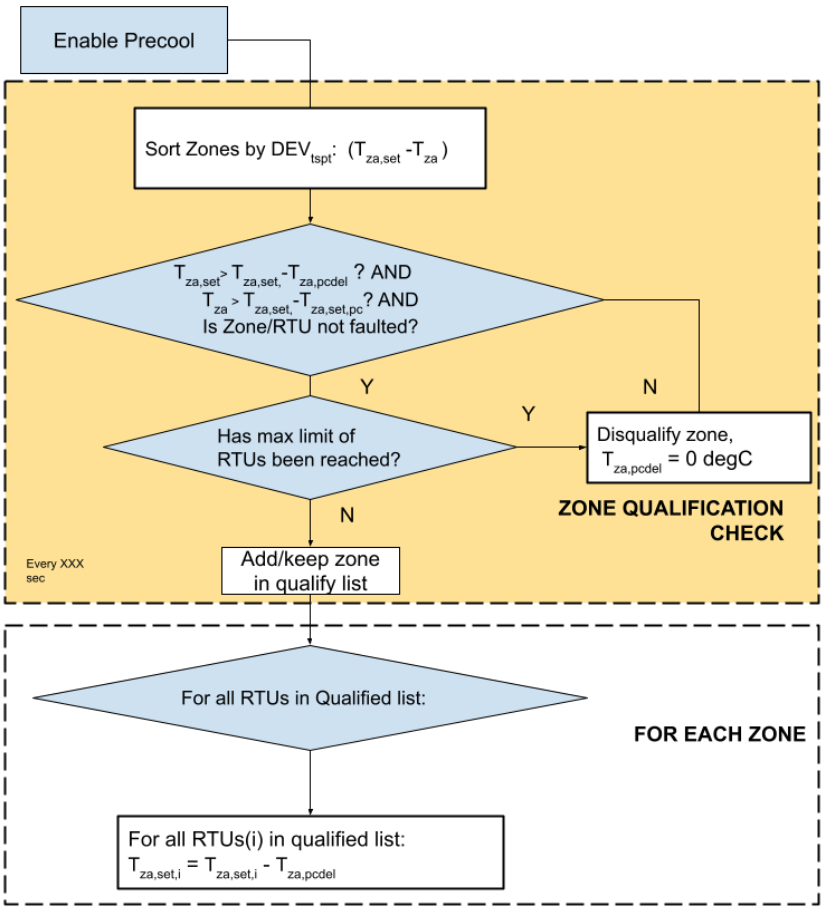


Figure 25: Zone-level pre-cooling for rooftop units (RTU) - intermediate approach

In this use case, multiple RTUs are given sorting criteria based on current zone temperature relative to the precool setpoint. Zones that are warmer are prioritized and precooled sequentially. The logic may incorporate a simple demand limiting strategy by setting a maximum number of RTUs that can be qualified/activated at once.

4.3.2.2 Parameter definition

The depth and timing of precooling strategies may be constant values in simple cases, but in order to accommodate dynamic goals in certain DR programs, a control strategy might be in place. Assuming a day ahead signal and notice of duration of the shed event, the strategy could take into account these inputs along with predicted outdoor air temperatures to set the time at which precooling starts ($\mathbf{t_{start}}$) as well as how low the precool temperature is set ($\mathbf{T_{za,pcdel}}$).

4.3.2.3 Zone prioritization and qualification check

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

For precooling, the logic first examines all RTUs and focuses on those that will bring most benefit during the shed event, by removing problematic equipment, zones that are already too cold or zones that have recently been precooled. At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as faulted status, zone conditions that are at or below the precool threshold or other comfort conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. Existing faults in the RTu that could be classified as disqualifying may include any that would negatively affect occupant comfort or the ability for the RTU to meet setpoint, this could include undercharge or fouling faults, staging faults, or programmatic faults to the thermostat. The thermal comfort check will examine values for $\mathbf{T_{za}}$ values, determining whether or not $\mathbf{T_{za}}$ is below the comfort threshold. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle. In addition to these initial checks, there is another demand spike prevention condition that also determines whether or not the qualified list has reached its maximum allowable number of running units. If this is the case an RTU is skipped until the number falls below maximum:

```
#Read all RTUs in RTU_list:
#DEV_stpt= (Tza,set-Tza,pcdel) - Tza,i
#Sort all Zones_i by DEV_stpt
#If:
#    RTU_i not {precooled} in last X cycles AND
#     Tza,set_i > Tza,set,min  AND
#     Tza_i > Tza_i,set,min AND
#     Zone/RTU_i not {faulted}
#     len(qualify_list)< max_RTU

#qualify_list.append(Ti)
#else: Continue
```

4.3.2.4 Precool strategy

The qualification check is followed by the execution of the precool sequence, in this case it would start with prioritization of zones by sorting all qualifying zones by **deviation of current zone temperature from current set point:** (DEVtspt =Tza,set -Tza). The strategy adjusts qualifying zone setpoints $T_{za,set}$ by a predetermined initial offset value, or a redefined value for $T_{za,pcdel}$ (e.g. 2F to 4F). At each cycle, the next RTU kicks on based on the prioritization, until the RTU max limit has been reached.

```
#Read all Zones_i in qualify_list:

For Zones_i :
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# Tza,set,i=Tza,set,i-Tza,pcdel
```

4.3.3 Complex Approach

4.3.3.1 Required data points

Table 14: Required Data Points from BAS (controllers)

Point	Description	Type	**Requirements (Read/Write)**
T_{za}	Zone air temperature	Time-series sensor reading from BAS	Read
$T_{za,set}$	Effective zone air temperature setpoint	Time-series sensor reading from BAS	Write
P_{bldg}	building power consumption	Time-series sensor reading from BAS	Read/Write

Table 15: New parameters and defined threshold in the DF algorithm

Abbreviation	**Data point**	**Data type**
$T_{za,ztadel}$	adjustment factor for ZTA strategy	Algorithm variable
DEV_{tspt}	Zone temperature deviation from effective setpoint	Algorithm variable
$T_{za,set,min}$	Comfort-based minimum zone air temperature setpoint	User-defined parameter
$P_{peak_monthly}$	Monthly peal building power	Algorithm variable
ε_0	Calculated difference between P_{bldg} and $P_{peak_monthly}$	Algorithm variable
ε_{min}	Minimum allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

db_{ε}	Deadband value for ε_{min}	User-defined parameter
$RTU_{\#_max}$	Maximum # of VAVs allowed on qualified list	Algorithm variable

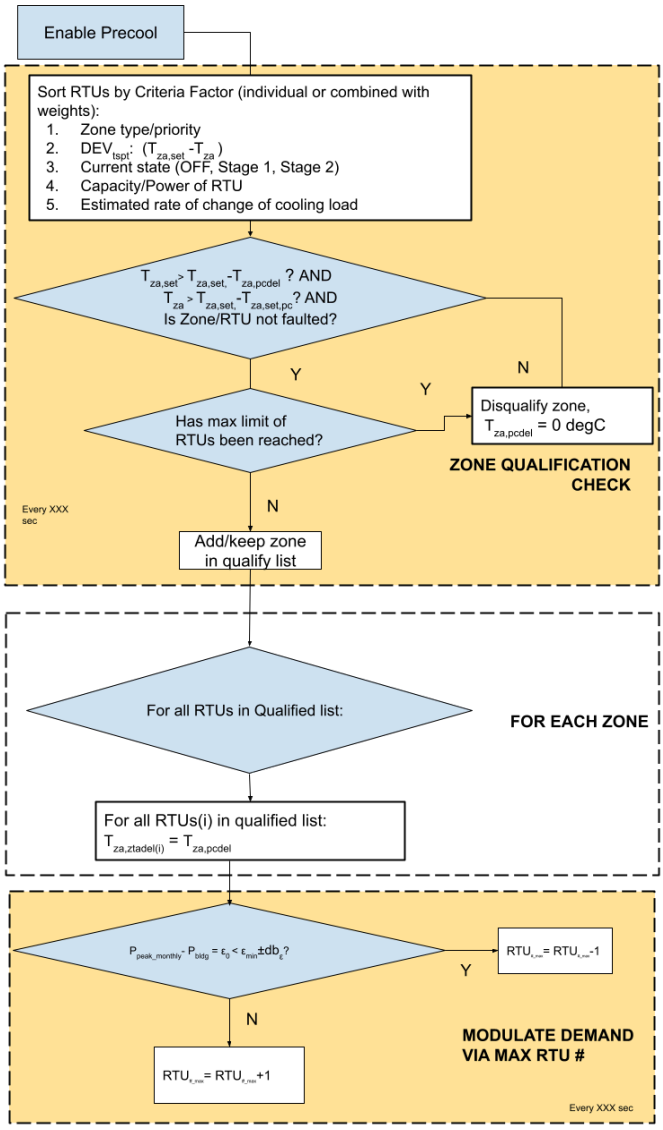


Figure 26: Zone-level pre-cooling for rooftop units (RTU) - complex approach

In this use case, multiple RTUs are given sorting criteria based on a weighted product of current zone temperature relative to the precool setpoint and another characteristic. Zones that are warmer are prioritized and precooled sequentially. The logic may incorporate a modulating form of the demand limiting strategy by dynamically setting a maximum number of RTUs that can be qualified/activated at once based on how close the strategy gets to creating a new demand peak.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.3.3.2 Parameter definition

The depth and timing of pre-cooling strategies may be constant values in simple cases, but in order to accommodate dynamic goals in certain DR programs, a control strategy might be in place. Assuming a day ahead signal and notice of duration of the shed event, the strategy could take into account these inputs along with predicted outdoor air temperatures to set the time at which precooling starts ($\mathbf{t_{start}}$) as well as how low the precool temperature is set ($\mathbf{T_{za,pcdel}}$).

4.3.3.3 Zone prioritization and qualification check

For precooling, the logic first examines all RTUs and focuses on those that will bring most benefit during the shed event, by removing problematic equipment, zones that are already too cold or zones that have recently been precooled. At each logical cycle, the zone qualification check examines each zone for current disqualifying zone conditions such as faulted status, zone conditions that are at or below the precool threshold or other comfort conditions, and releases the zone temperature adjustment for zones that are approaching uncomfortable conditions for occupants. Existing faults in the RTu that could be classified as disqualifying may include any that would negatively affect occupant comfort or the ability for the RTU to meet setpoint, this could include undercharge or fouling faults, staging faults, or programmatic faults to the thermostat. The thermal comfort check will examine values for $\mathbf{T_{za}}$ values, determining whether or not $\mathbf{T_{za}}$ is below the comfort threshold. Zones that are in the occupied list and pass the qualification checks are appended to the qualified list of zones for the current iteration of the logical cycle. In addition to these initial checks, there is another demand peak prevention condition that determines whether or not the qualified list has reached its maximum allowable number of running units. In this case, the maximum number of RTUs is determined by calculating the difference between the current monthly peak and the current demand of the building. If the number of activated RTUs is equal to the max, a qualifying RTU is skipped until the number falls below maximum. In this case the strategy would start with prioritization of zones by sorting all qualifying zones by an individual or combined list of prioritization criteria. This includes the following criteria:

1. **Zone type/priority:** Zone use type and priority may vary across RTUs, zones that serve high density of occupants or require more regulated temperature or may be prioritized higher than less frequented areas.
2. **Deviation of current zone temperature from current set point:** (DEVtspt =Tza,set -Tza)
3. **Current state:** (OFF, Stage 1, Stage 2)
4. **Capacity/Power of RTU:** Rated cooling capacity, current power draw from RTU
5. **Estimated rate of change of cooling load:** Interaction between rated cooling capacity and current thermal load of space

As an example, the zones could be prioritized by calculating the deviation between the current

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

zone temperature and the current zone temperature setpoint and adjusting qualifying zone setpoints $T_{za,set}$ by a predetermined initial offset value, or a redefined value for $T_{za,pcdel}$ (e.g. 2F-4F).

```
#Read all Tza_i in qualify_list:
#DEV_stpt= (Tza,set-Tza,pcdel) - Tza,i
#Sort all Zones_i by DEV_stpt

#Read all RTUs in RTU_list:

#If:
#    RTU_i not {precooled} in last X cycles AND
#     Tza,set_i > Tza,set,min  AND
#     Tza_i > Tza_i,set,min AND
#     Zone/RTU_i not {faulted}
# If len(qualify_list)= RTU#_max:
# Continue
# else:
# qualify_list.append(Ti)
```

4.3.3.4 Precool strategy

The qualification check is followed by the execution of the precool sequence:

```
For Tza_i :
# Tza,set,i=Tza,set,i-Tza,pcdel
```

4.3.3.5 Demand modulation via max RTU limit

In cases where the building is avoiding a spike in demand from simultaneous activation of RTUs, the complex form of this strategy actively monitors the current demand of the building (assuming this measurement is available at frequent intervals) and compares it to the established monthly peak. As long as the demand does not exceed the minimum error, ε_{min} the maximum number of activated RTUs incrementally increases by 1. In the case that the demand gets too close to the new peak, less than ε_{min} , the max number of RTUs decreases. Given the possibility of multiple RTUs with different capacities, it is important users set ε_{min} values to be no smaller than the estimated power draw from the largest existing unit.

```
# If P_peak_monthly - P_bldg = ε_0 < ε_min ± db_ε
# RTU#_max= RTU#_max-1
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# else
# RTU#_max= RTU#_max+1
```

4.4 Plant-level Strategy: Chilled water temperature setpoint reset

CHILLER PLANT LOAD CHECK

- Qualify a chiller plant by load
  - Verify that there are no AHUs served by the chiller plant for which the control signal to the cooling coil valve exceeds a control signal threshold and the supply air temperature deviation exceeds its threshold;
  -
  - And verify the chilled water supply temperature does not exceed a maximum value.

INITIAL SHED

- For all chiller plants in qualified list, for the initial timestep:
  -
  - Apply new chilled water setpoints for chillers

RATCHET STRATEGY

- If target load shed is not met and/or the plant qualification limits are not exceeded, add incremental delta to new setpoints for all qualified chillers

4.4.1 Required data points

Table 16: Required Data Points from BAS (controllers)

Abbreviation	**Description**	**Type**	**Requirements (Defined/Calculated/Read/Write)**
$CS_{AHU,CC,i}$	AHU-specific cooling coil valve control signal	Time-series reading from BAS	Read
$CS_{AHU,CC,max}$	Maximum allowable AHU cooling coil valve control signal for qualifying chiller plants	User-defined parameter	Read

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

DD_{bldg}	Building demand decrease	Algorithm variable	Calculated
DD_{cap}	Target electrical demand decrease	User-defined parameter	Defined
$DEV_{SAT,AHU,i}$	AHU-specific supply air temperature deviation from setpoint	Algorithm variable	Calculated
P_{bldg}	Building power consumption	Time-series sensor reading from BAS	Read
SAT_i	Supply air temperature of AHU_i	Time-series sensor reading from BAS	Read
$SAT_{set,i}$	Supply air temperature setpoint of AHU_i	User-defined or algorithm-determined parameter	Read
T_{chw}	Chilled water supply temperature	Time-series sensor reading from BAS	Read
T_{chwset}	Chilled water supply temperature setpoint	User-defined or algorithm-determined parameter	Read/Write
$T_{chwset,del}$	Adjustment factor for CHW reset strategy	Algorithm variable	Write
$T_{chwset,max}$	Comfort-based maximum chilled water supply temperature setpoint	User-defined parameter	Defined
t_{wait}	Wait time for ratcheting and rebound management	User-defined parameter	Defined
$Threshold_{SAT,AHU}$	Maximum allowable deviation of AHU supply air temperature above its setpoint	User-defined parameter	Defined
$\delta_{ratchet}$	Incremental delta for ratcheting chilled water supply temperature setpoint	Algorithm variable	Defined
ε_{DD}	Calculated difference between DD_{cap} and DD_{bldg}	Algorithm variable	Calculated

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

ε_{min}	Threshold for allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter	Defined

4.4.2 Strategy description

This strategy is intended for a portfolio of buildings, with each building containing its own chilled water plant consisting of one or more chillers serving multiple AHUs. A schematic diagram of a representative chilled water plant and distribution system is shown in Figure 27. Staging of chillers is assumed to be determined by the cooling load at the secondary loop, which is driven by the AHU cooling coil demand. Chilled water (CHW) temperature reset allows for direct control of electrical demand by enabling the reduction of the speed of the chiller compressor. Some estimates are that the COP of centrifugal chillers increases 2% for every 1°F increase to the CHW temperature setpoint[1]. Reset strategies for chillers have traditionally been implemented as a continuous load-based strategy, related to outdoor air conditions or the current demand of the downstream AHU equipment. The thermal inertia in larger chilled water distribution systems will delay the impact of reset strategies at the zone level. Precooling the chilled water prior to the shed event could further delay or reduce the effect.

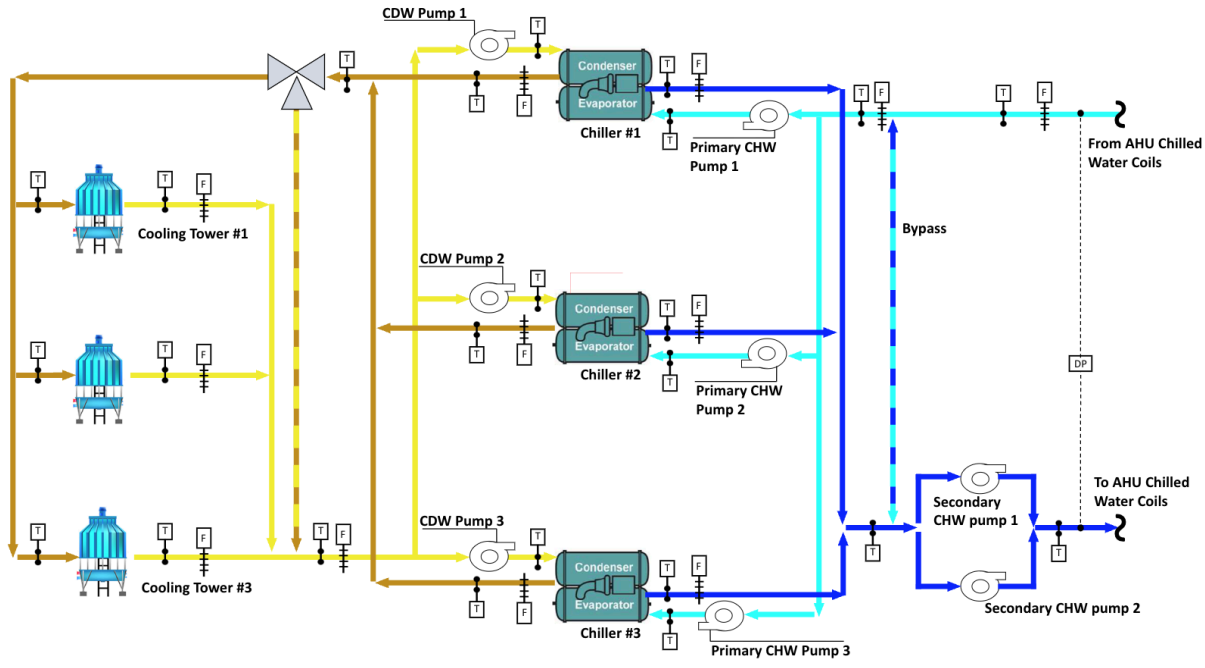


Figure 27: Diagram of assumed chiller plant configuration

[1] https://www.gsa.gov/cdnstatic/Control_Optimization_System_for_Chiller_Plants.pdf

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Although this strategy will provide immediate demand reductions, a tradeoff between the reduced demand from the chiller and an increase in demand from the downstream pump and fan equipment exists. Assuming the supply air temperatures (SAT) of the downstream equipment remain static, the CHW temperature increase will cause all downstream cooling valves to open in order to maintain a constant SAT setpoint. For chiller plants such as the one in Figure 27, this will cause the variable speed pump systems to increase their flow to maintain the differential pressure in the chilled water distribution system at setpoint. Once the supply air temperature can no longer be met by the AHU, the zone air temperature will slowly start to increase until downstream VAV's start to request additional airflow, thus increasing AHU fan speed.

Four variants for implementing chilled water reset are described in this specification: 1) a singular shed command to all qualifying chiller plants at the start of an event; 2) an incremental shed command to all qualifying plants; 3) an incremental shed command to all qualifying plants based on real time demand readings and a single load shed target; and 4) an incremental shed command to all qualifying plants based on realtime readings and a time series load shed target. To optimize building level demand reductions with controlled impact on zone conditions, continuous load checking is needed at the plant and AHU equipment levels.This may be best achieved by implementing a **conservative, incremental reset approach** (i.e., Variant 2, 3, or 4) while monitoring the load conditions on downstream AHUs.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

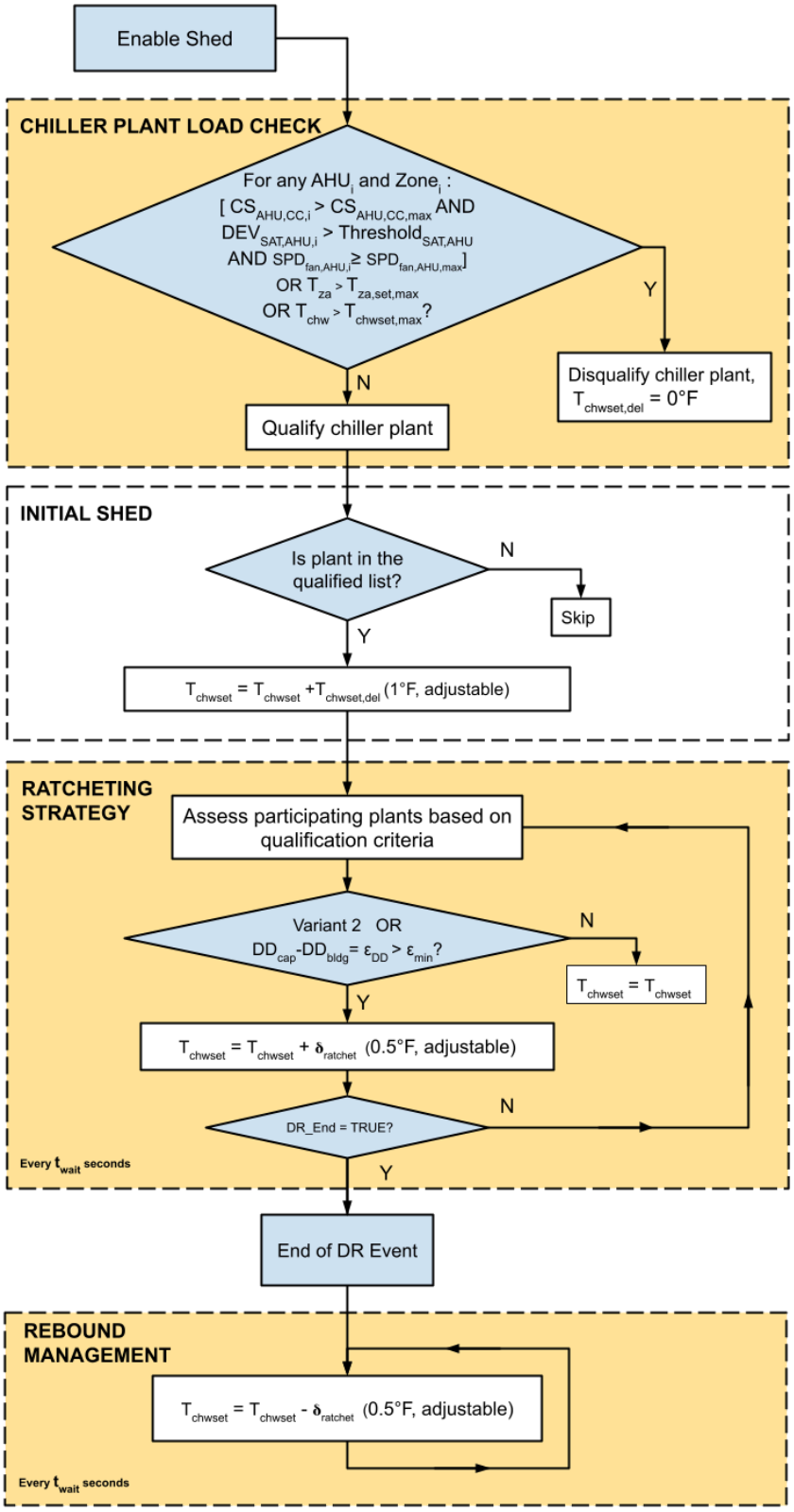


Figure 28: A complete algorithm for chilled water temperature reset

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

A flow diagram illustrating the implementation of the chilled water reset algorithm is shown in Figure 28. The algorithm is divided into four components, namely: 1) chiller plant load check; 2) initial shed; 3) ratcheting strategy; and 4) rebound management. The four components are described in the ensuing sections.

4.4.2.1 Chiller plant load check

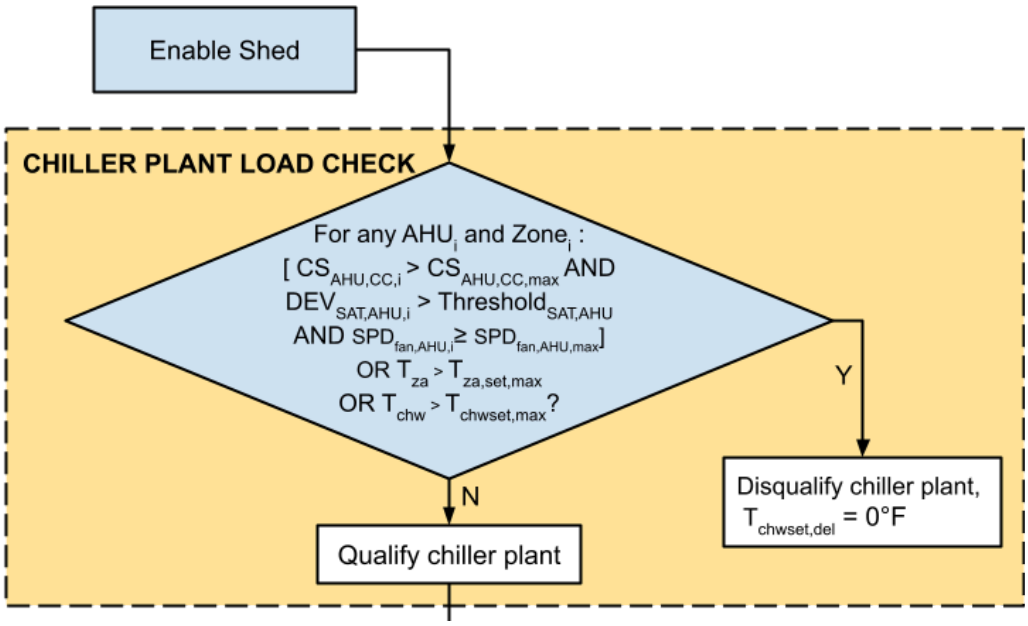


Figure 29: Algorithm for checking chiller plant load

Figure 29 shows the first component of the chilled water reset strategy, the Chiller Plant Load Check. This component determines whether a chiller plant meets operational criteria and is qualified for inclusion in the strategy.

While the load shed signal is enabled, the Chiller Plant Load Check sequence will monitor the load conditions on downstream AHUs as well as the chilled water temperature. A chiller plant is disqualified if: 1) the control signal to the cooling coil valve of any AHU_i it serves ($CS_{AHU,CC,i}$) exceeds a maximum threshold ($CS_{AHU,CC,max}$) and simultaneously the deviation of that AHU's supply air temperature (SAT) from its setpoint ($DEV_{SAT,AHU,i} = SAT_i - SAT_{set,i}$) exceeds a threshold common to all AHUs ($Threshold_{SAT,AHU}$); or 2) the chilled water temperature (T_{chw}) exceeds a defined maximum chilled water setpoint ($T_{chwset,max}$). If there are no disqualifying conditions, the chiller plant is qualified to execute the DF strategy. Pseudocode outlining the steps implemented for chiller plant qualification is presented below.

```
#Read all plant_i in plant_list. For each plant_i:

      #Read all AHU_i fed by plant_i. For each AHU_i:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
            # [If CS_AHU,CC,i > CS_AHU,CC,max
            # AND
            # If DEV_SAT,AHU,i > Threshold_SAT,AHU]
            # OR
#Read T_chw for each plant_i:
            # If T_chw > T_chwset,max
disqualify_list.append(plant_i)
else qualify_list.append(plant_i)
```

4.4.2.2 Initial shed

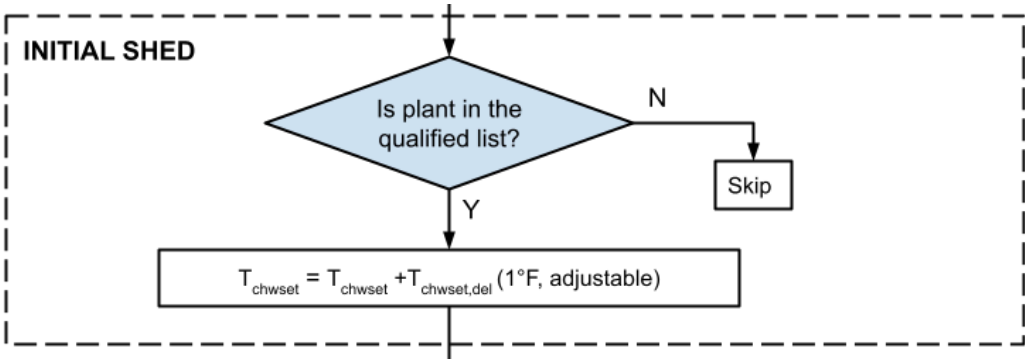


Figure 30: Algorithm for initial shed

The Initial Shed component of the strategy is shown in Figure 30. For the first iteration of the DF strategy, all chiller plants in the qualified list will have their current chilled water temperature setpoint increased by an amount $T_{chwset,del}$ (suggested value of 1°F, adjustable). For Variant 1, a larger setpoint increase of 2°F or more may be appropriate.

4.4.2.3 Ratcheting strategy

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

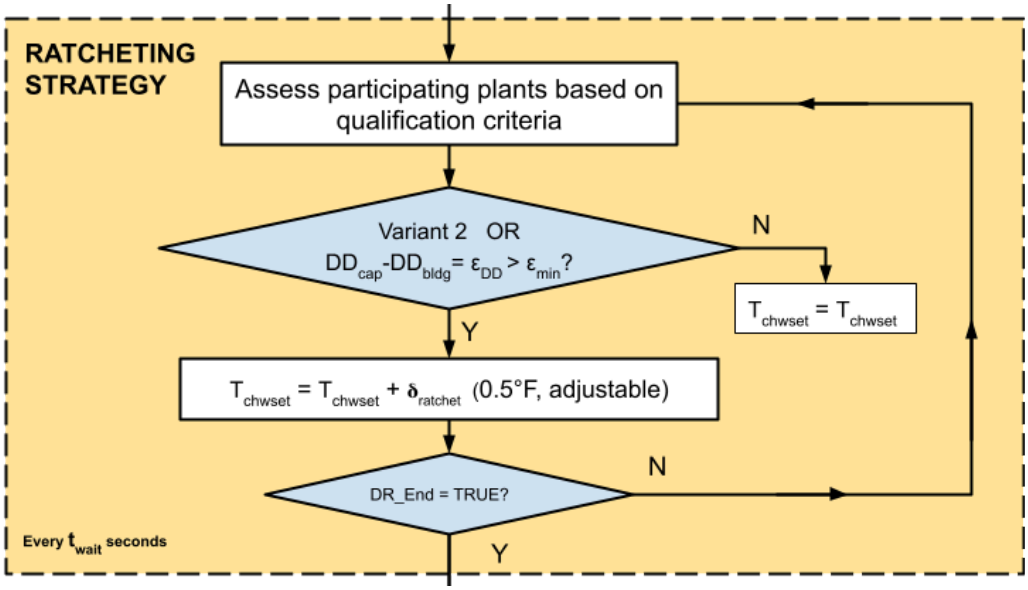


Figure 31: Algorithm for ratcheting of chiller supply temperature setpoint reset

Figure 31 shows the third component of the reset strategy, the Ratcheting Strategy. The conservative, incremental approach to demand reduction is accomplished by gradually increasing the chilled water setpoint until either a limiting threshold is reached (i.e., Variant 2) or until a demand reduction target is achieved (i.e., Variants 3 and 4).The target-based approach of Variants 3 and 4 controls the load shed achieved by implementing an initial reset of the CHW temperature for all qualified plants (shown in Initial Shed in Figure 30), then adds an incremental reset in order to reduce the error between the current load shed (DD_{bldg}) and the target load shed (DD_{cap}). DD_{bldg} is determined by subtracting the current measured power, P_{bldg}, from a baseline model. Ratcheting continues at regular intervals (e.g., every 15 minutes) until the target load shed is achieved or until the chiller plant is no longer qualified. Ratcheting is not implemented for Variant 1.

4.4.2.4 Rebound management

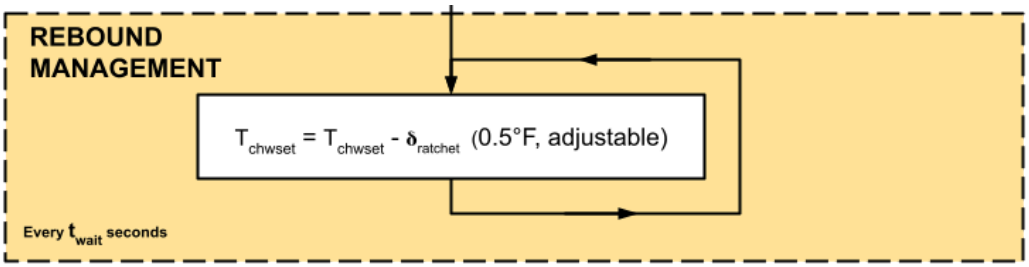


Figure 32: Algorithm for rebound management

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Releasing the CHW temperature setpoint at the end of the demand response event may result in a significant rebound effect, especially as the chiller plant works to accommodate the chilled water demand from downstream AHUs. To mitigate this effect, the specified strategy recommends an incremental decrease of each chiller plant's supply water temperature setpoint executed at regular intervals to allow for a slow return to pre-event conditions. The rebound management strategy is illustrated in Figure 32.

4.5 AHU Level Strategy: Fan/VFD Speed Limiting

The following strategy assumes the fan motor speed and chilled water valve positions are accessible outputs of the control algorithm. There is also an assumption that the existing native control sequence incorporates supply air temperature and static pressure resets. **This specification specifically avoids static pressure and supply air temperature resets as the load reduction mechanism, given we are envisioning these strategies as already in place via ASHRAE G36 trim and respond sequences.**

AHU fan/VFD speed limiting (hereafter referred to as fan speed limiting) as a shed strategy has been mentioned in several studies performed over the past 20 years including:

- Piette et al. (2005)[2], who described its use as a strategy at a number of field sites participating in demand response (DR) programs, such as the Bank of America Concord Technology Center, where they rotated the fan participating in DR to share reduction of service throughout the site, and UC Santa Barbara, where the VFD limit was reduced based on price thresholds;
- Motegi et al. (2007)[3], who tested two different load shed variants based on VFD limiting, one in which the VFD limit is an absolute value and another in which there is active VFD reduction relative to a pre-DR mode;
- Kim et al (2014)[4], who deployed fan limiting and reduction as part of a tiered DR shed strategy in four demonstration sites in New York; and
- Blum and Norford (2014)[5], who demonstrated successful shed via supply fan static pressure limiting in a Modelica simulation study.

Similar to the zone level shed strategies described by Granderson et al. (2025)[6], the fan/VFD limiting strategy comes in four variants, which consist of: 1) a singular shed command to all qualifying systems at the start of an event; 2) an incremental shed command to all qualifying systems; 3) an incremental shed command to all qualifying systems based on real time demand readings and a single load shed target; and 4) an incremental shed command to all qualifying

[2] https://eta-publications.lbl.gov/sites/default/files/58178.pdf
[3] https://eta-publications.lbl.gov/sites/default/files/59975.pdf
[4] https://eta-publications.lbl.gov/sites/default/files/lbnl-186969_nyserda_20723.pdf
[5] https://www.tandfonline.com/doi/full/10.1080/10789669.2014.958975
[6] https://eta-publications.lbl.gov/sites/default/files/2026-01/df_open_spec_paper_1.pdf

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

systems based on realtime readings and a time series load shed target. The four variants are enumerated in Figure 33.

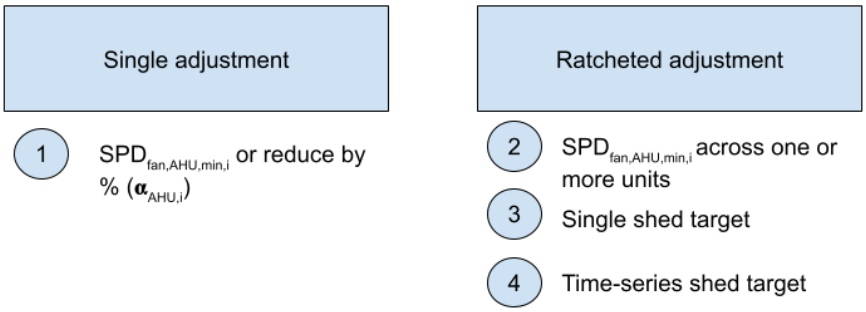


Figure 33: Four load shed variants applicable to AHU fan/VFD speed limiting.

This strategy assumes the AHU has one fan and is generalized to include two different cases in which the control algorithm implements speed control during a demand flexibility (DF) event by either directly controlling a commandable point that limits the maximum speed of the fan, or directly adjusts the fan speed of each AHU. Controlling a parameter that limits the maximum speed enables the system to continue to operate in closed-loop control, whereas controlling the speed directly constitutes open-loop control. The DF strategy employed at a particular site with multiple AHUs can be a combination of the different variants. Alternatively, a reduction such as the one described in Variant 1 may be applied across all AHUs to simplify implementation.

There are a number of considerations to keep in mind when initiating a fan limiting strategy. For instance, a fan speed limit or reduction is expected to result in a reduction in static pressure in the supply duct, and lower airflow rates being supplied to the downstream VAV boxes. This will cause the VAV dampers to open further as the VAV airflow control works to maintain the current airflow setpoint. Although a reduction of fan speed can reduce fan power effectively, it should be used with caution to ensure minimum ventilation requirements are met. VAV minimum airflows should correspond to minimum ventilation requirements and should be used as a qualification check for this strategy.

Capping the fan speed will also affect the waterside of the AHU operation. For a fixed supply air temperature setpoint, as the airflow decreases in the supply duct, the SAT control loop will decrease the AHU cooling valve command in a proportional manner. On the other hand, if the supply air temperature setpoint is resetting, the reduction in airflow rate will tend to cause the reset strategy to decrease the setpoint if zones are not receiving adequate cooling, and the AHU cooling valve command will increase to meet this lower setpoint. In order to ensure that reductions in fan energy use are not offset by increases in chiller energy use, it is recommended to set the supply air temperature setpoint to a constant value during the DF event.

4.5.1 Required data points

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

The required points and specified user defined parameters are found in Table 17 below.

Table 17: Inventory of parameters and variables for fan speed limiting.

Abbreviation	**Data point**	**Data type**	**Units**	**Requirements (Defined/Calculated/ Read/Write)**
DD_{bldg}	Building demand decrease from reading relative to baseline	Time-series sensor reading from BAS/meter	kW	Calculated
DD_{cap}	Target electrical demand decrease	User-defined parameter	kW	Defined
$Fan_lim_{AHU,i}$	Fan speed limit at each AHU	Time-series reading from BAS	%	Read/Write
$Fan_lim_{AHU,pre,i}$	Fan speed limit value prior to DF event	User-defined parameter	%	Defined
P_{bldg}	Electric demand from building meter	Time-series sensor reading from BAS	kW	Read
$SPD_{fan,AHU,i}$	Fan speed at each AHU	Time-series sensor reading from BAS	%	Read/Write
$SPD_{fan,AHU,min,i}$	Minimum allowable fan speed at each AHU	User-defined parameter	%	Defined
$T_{AHU,\ ra,i}$	Return air temperature at each AHU	Time-series sensor reading from BAS	°C	Read/Write
$T_{AHU,sa_spt,i}$	AHU specific supply air temperature setpoint	Time-series sensor reading from BAS	°C	Read
$T_{zone,i}$	Zone-specific air temperature	Time-series sensor reading from BAS	°C	Read
$T_{zone,max,i}$	Maximum allowable zone air temperature	User-defined parameter	°C	Defined
t_{wait}	Wait time to determine conditions for ratcheting	User-defined	seconds	Defined

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		parameter		
$V_{flow,VAV,i}$	Measured volumetric flow at each VAV	Time-series sensor reading from BAS	CFM	Read
$V_{flow,VAV,min,i}$	Minimum allowable VAV flow at each zone	User-defined parameter	CFM	Defined
$\alpha_{AHU,i}$	AHU-specific fan upper limit relative to max fan speed	User-defined parameter	%	Defined/Write
$\delta_{ratchet,\ i}$	AHU-specific incremental delta for ratcheting fan speed both down and up	User-defined parameter	%	Defined/Write
ε_{DD}	Calculated difference between DD_{cap} and DD_{bldg}	Algorithm variable	kW	Calculated
ε_{min}	Threshold for allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter	kW	Defined

The fan speed limiting strategy is shown in Figure 34 and is implemented using four components, namely: 1) Initial Qualification; 2) Initial Shed; 3) Ratcheting Strategy; and 4) Rebound Management.

4.5.2 Prioritization and qualification check

Upon initiating the DF event, the Initial Qualification component will sort all AHUs by either fan size (associated with kW reduction potential) or current speed of each fan. Next, a qualification check is performed to verify:

- the AHU associated with the fan is not faulted in such a manner that reducing the speed may result in adverse effects downstream;
- the current speed of each fan $SPD_{fan,AHU,i}$ is above its minimum allowable value $SPD_{fan,AHU,min,i}$, where “i” is the index for the fan;
- the airflow rate of all downstream VAVs associated with each AHU meets or exceeds the VAV minimum associated with ventilation standards; and
- all downstream zone temperatures are maintained below their maximum allowable value.

If the AHU has met all of these conditions, it is added to the qualified list of participating systems for the duration of the event or until one or more of the conditions is no longer met.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.5.3 Shed strategy

Next, the Initial Shed component reduces the speed of all AHU fans in the qualified list via the fan speed limit variable, $Fan_lim_{AHU,i}$. For open-loop control (direct speed control), $Fan_lim_{AHU,i}$ will represent the fan speed command, and for closed-loop control it will represent the fan maximum speed parameter. For either open-loop or closed-loop control and for all variants, $Fan_lim_{AHU,i}$ is reduced by a relative amount $\alpha_{AHU,i}$ (e.g., 80%, adjustable). Note that the index "i" on the parameter $\alpha_{AHU,i}$ indicates that a unique value of the parameter can be defined for each individual fan. Note also that it is recommended to disable any existing supply air temperature reset strategy and overwrite each AHU supply air temperature setpoint to a constant value with the user-defined parameter $T_{AHU,sa_spt,i}$.

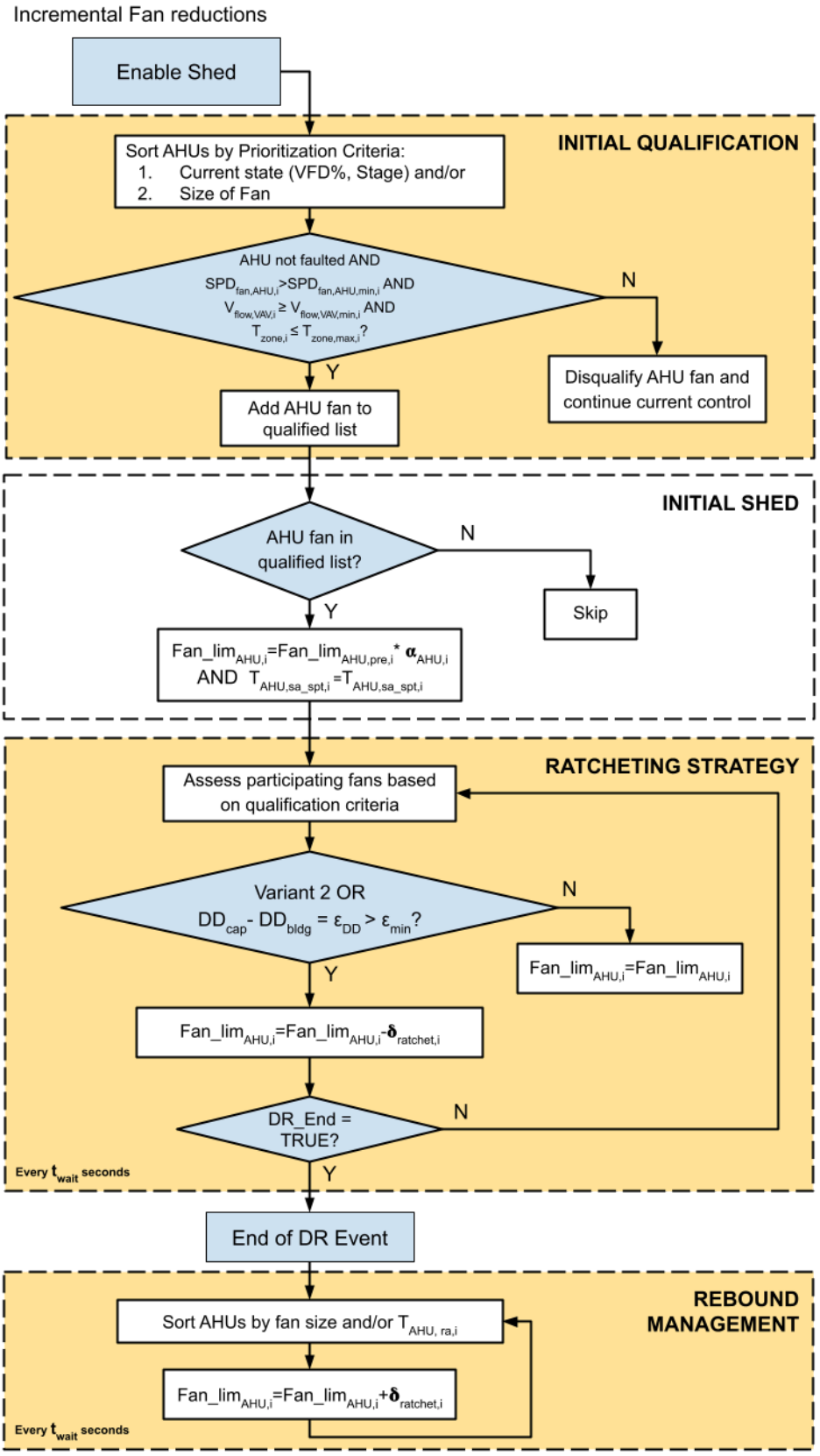
4.5.4 Ratcheting sequence

The Ratcheting Strategy component implements an incremental reduction of $\delta_{ratchet,i}$ to the fan speed limit variable. For Variant 2, incremental reductions are implemented every t_{wait} seconds until one or more of the qualification criteria are no longer satisfied. For Variants 3 and 4, the Ratcheting Strategy compares the current reduction in power DD_{bldg} (determined by subtracting the current measured power, P_{bldg}, from a baseline model) to a target value DD_{cap}. If the target reduction is not met and the fan remains qualified, the strategy continues to implement additional reductions of the fan speed limit variable until the error between the shed target and measured shed ε_{DD} falls below the minimum error ε_{min}.

4.5.5 Post-event strategy (Rebound management)

Early publications of this strategy reported that a significant rebound effect may occur after releasing the fan speed limits, as the fans work to make up the airflow demand from downstream VAVs. To mitigate this effect, the Demand Management component recommends a staggered, incremental return to normal operation to allow for a slow return to pre-event conditions. For the closed-loop implementation of this strategy, the fan speed limit parameter should be gradually increased back to its pre-event value. For the open-loop implementation, the fan speed should be gradually increased until the supply air static pressure meets or exceeds the static pressure setpoint, at which point operation of the fan should revert to closed-loop control. Increases to the speed (or maximum speed parameter) occur at regular intervals of t_{wait} seconds. As soon as fan operation has returned to normal closed-loop control, supply air temperature reset (if disabled) should be re-enabled. The incremental return to normal conditions applies to all Variants and should be prioritized by either fan size or the highest current return air temperature ($T_{AHU,ra,i}$) of each AHU (a proxy for current downstream VAV thermal load).

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/



Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Figure 34: Fan speed limiting strategy.

4.6 AHU Level Strategy: Cooling Valve Limiting

Capping the maximum open position of AHU cooling valves reduces chilled water flow through cooling coils, thus resulting in a reduction in the cooling capacity delivered by the AHU, and ultimately reducing chiller load. This strategy has been implemented in the past in both simulation and field studies, including:

- Piette et al. (2004)[7], who described how the strategy was used in a number of field sites participating in demand response (DR) programs, including at UC Santa Barbara where they reduced the maximum open position of a chilled water valve to 0% based on price thresholds.
- Motegi et al (2007)[8], who described one load shedding variant where the limit is set to an absolute %-open value, and another in which the valve position is actively reduced relative to the pre-DR conditions. The authors of the study recommended limiting the supply fan VFD speed concurrently to avoid unwanted increases in fan energy use during a DR event.
- ASHRAE RP-1390 (2011)[9], which also recommended locking the fan speed to its pre-event value to avoid an increase in fan energy.
- Dai, Li and Wang (2023)[10], who used simulation to develop a chilled water valve limiting scheme which monitors zone temperatures to ensure occupant comfort.

Similar to the zone-level shed strategies described by Granderson et al. (2025)[5], the cooling valve limiting strategy comes in four variants, which consist of: 1) a singular shed command to all qualifying systems at the start of an event; 2) an incremental shed command to all qualifying systems; 3) an incremental shed command to all qualifying systems based on real time demand readings and a single load shed target; and 4) an incremental shed command to all qualifying systems based on real time readings and a time series load shed target. The four variants are enumerated in Figure 35.

[7] https://eta-publications.lbl.gov/sites/default/files/58178.pdf
[8] https://eta-publications.lbl.gov/sites/default/files/59975.pdf
[9]https://store.accuristech.com/tia/standards/rp-1390-short-term-curtailment-of-hvac-loads-in-buildings?product_id=1810109&srsltid=AfmBOoq79KcrqsW_DghJ5QnTvlXNTS30AB8YjqJV2UPYacj7MXsOZ-Z8
[10]https://research.polyu.edu.hk/en/publications/a-smart-feedback-control-strategy-for-building-fast-demand-respon

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

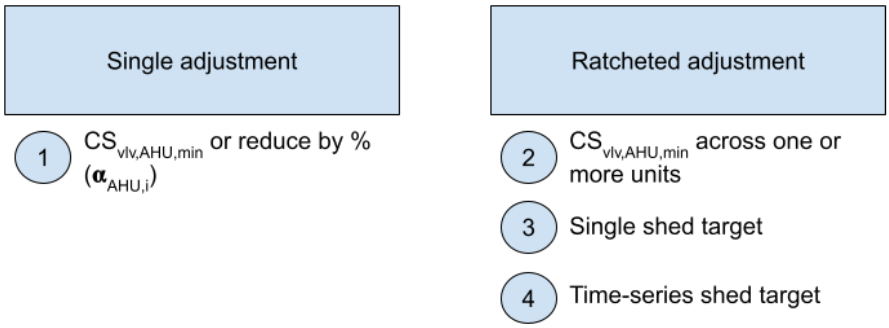


Figure 35: Four load shed variants applicable to AHU cooling valve limiting.

Consideration should be given to how this strategy will impact other aspects of equipment operation and comfort control. Throttling the cooling coil valve will cause the supply air temperature to drift above setpoint, which could have two effects. First, zone air temperature and relative humidity may increase due to reduced cooling capacity and dehumidification at the coil. In response to rising zone air temperatures, the VAV boxes will open to deliver more cooling to the zones and the fan speed will increase to maintain the duct static pressure at its setpoint. To avoid an unwanted increase in fan power, it is recommended to limit the AHU fan speed to its pre-event value.

4.6.1 Required data points

The required points and specified user defined parameters are listed in Table 18.

Table 18: Inventory of parameters and variables for valve limiting.

Abbreviation	Data point	Data type	Unit	Requirements (Defined/Calculated/ Read/Write)
$CS_{vlv,AHU,i}$	Chilled water valve command signal at each AHU	Time-series sensor reading from BAS	%	Read/Write
$CS_{vlv,AHU,min}$	Minimum allowable valve control signal	User-defined parameter	%	Defined
DD_{bldg}	Building demand decrease	Time-series sensor reading from BAS/meter	kW	Calculated

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

DD_{cap}	objective electrical demand decrease	User-defined parameter	kW	Defined
DEV_{satspt}	Deviation between supply air temperature and setpoint	Algorithm variable	°C	Calculated
P_{bldg}	Electric demand from building meter	Time-series sensor reading from BAS	kW	Read
$RH_{zone,i}$	Zone-specific relative humidity	Time-series sensor reading from BAS	%	Read
$RH_{zone,max,i}$	Maximum allowable relative humidity setpoint	User-defined parameter	%	Defined
$SPD_{fan,AHU,limit,i}$	Fan speed limit used to prevent the fan from compensating for the valve reduction, equal to speed of the fan prior to the DF event	User-defined parameter	%	Calculated/Write
$SPD_{fan,AHU,pre,i}$	Fan speed at each AHU prior to event	Time-series sensor reading from BAS	%	Read
$T_{AHU,sa,\ i}$	AHU supply air temperature	Time-series sensor reading from BAS	°C	Read
$T_{AHU,sa_spt,\ i}$	AHU supply air temperature setpoint	Time-series sensor reading from BAS	°C	Read
$T_{zone,i}$	Zone-specific air temperature	Time-series sensor reading from BAS	°C	Read
$T_{zone,max,i}$	Maximum allowable zone air temperature setpoint	User-defined parameter	°C	Defined

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

t_{wait}	Wait time to determine conditions for ratcheting	User-defined parameter	seconds	Defined
$\alpha_{AHU,i}$	AHU-specific limit relative to max chilled water valve command signal	User-defined parameter	%	Defined/Write
$\delta_{ratchet,\ i}$	AHU-specific incremental delta for ratcheting chilled water valve command signal	User-defined parameter	%	Defined/Write
ε_{DD}	Calculated difference between DD_{cap} and DD_{bldg}	Algorithm variable	kW	Calculated
ε_{min}	Minimum allowable difference between DD_{cap} and DD_{bldg}	User-defined parameter	kW	Defined

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

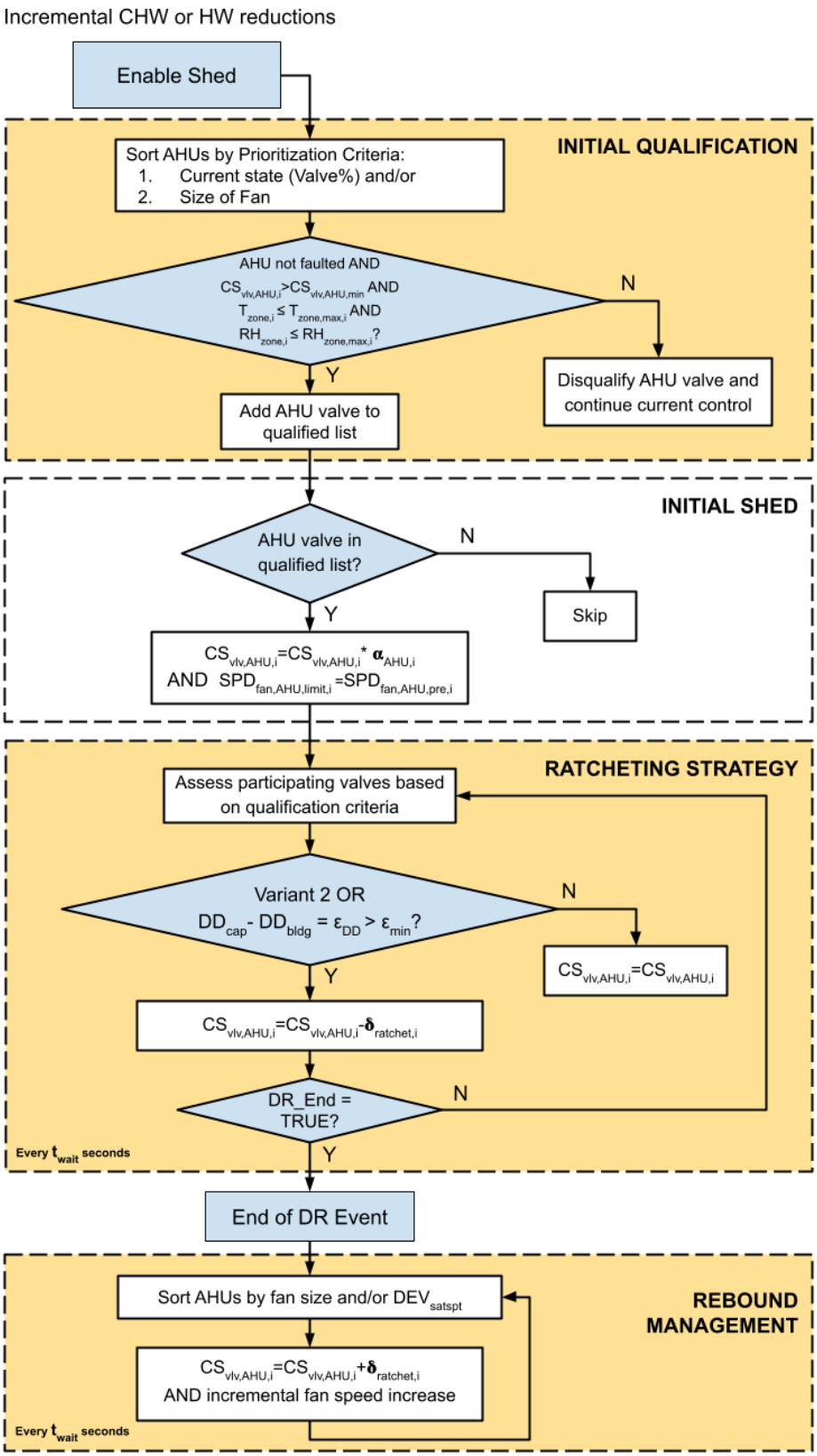


Figure 36: AHU cooling valve limiting strategy.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

4.6.2 Prioritization and qualification check

The cooling valve limiting strategy is implemented using four functions as shown in Figure 36. Upon initiating the demand flexibility event, the strategy will sort all AHUs by either fan size or current state of valve positions. Priority will be given to AHUs with the largest fans and/or whose valves are the furthest open, as they have the most potential for significant reductions in electricity demand For each AHU, the qualification check found in the Initial Qualification function verifies that:

- The associated AHU is not faulted in such a manner that reducing the cooling valve position may result in adverse effects downstream,
- The control signal to the AHU valve $CS_{vlv,AHU,i}$ is greater than the minimum allowable value defined for all valves $CS_{vlv,AHU,min}$,
- All downstream zones associated the AHU are within comfort bounds set by $T_{zone,max,i}$ and $RH_{zone,max,i}$

If the AHU has met all of these checks, it is added to the list of participating systems for the duration of the event or until one or more of the conditions in the qualification check is no longer met.

4.6.3 Shed strategy

The Initial Shed function in Figure 36 iterates through all of the qualifying AHUs and reduces the control signal to theAHU cooling valve by a relative amount $\alpha_{AHU,i}$ (e.g., 80%, adjustable). Alternatively, foror Variant 1, the cooling valve control signal could be reduced to its minimum value $CS_{vlv,AHU,min}$. For all variants, the AHU supply fan speed is limited to its value prior to the demand flexibility (DF) event, $SPD_{fan,AHU,pre,i}$. The DF strategy employed at a site with multiple AHUs can be a combination of the different variants. In addition, a reduction such as the one described in Variant 1 may be applied across all AHUs to simplify implementation.

4.6.4 Ratcheting sequence

The Ratcheting Strategy function implements an incremental reduction of $\delta_{ratchet,i}$ to the cooling valve control signal. For Variant 2, incremental reductions are implemented every t_{wait} seconds until one or more of the qualification criteria are no longer satisfied. For Variants 3 and 4, the Ratcheting Strategy function compares the current reduction in power DD_{bldg} (determined by subtracting the current measured power, P_{bldg}, from a baseline model) to a target value DD_{cap}. If the target reduction is not yet met and the AHU remains qualified, the strategy continues to

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

impose additional reductions to the valve control signal until the error between the shed target and measured shed ε_{DD} falls below the minimum allowable error ε_{min}.

4.6.5 Post-event strategy (Rebound management)

Releasing the cooling valve limits and associated fan speed limits following the DF event may result in a significant rebound effect as the AHUs work to make up the cooling demand from downstream VAVs. To mitigate this effect and allow for a slow return to pre-event conditions, the specified strategy recommends a staggered, incremental return to the valve position where the supply air temperature is at or below its setpoint followed by a switch to closed-loop control of the cooling valve. A similar incremental release of the fan speed (initially open-loop, and eventually switching to closed-loop) is recommended once the supply air temperature is under closed-loop control. This incremental return to normal conditions should be prioritized by either fan size or DEV_{satspt}, the deviation between current supply air temperature $T_{AHU,sa,i}$ and the supply air temperature setpoint $T_{AHU,sa_spt,i}$, with priority given to the largest fans and the AHUs whose supply air temperatures exceed their setpoints by the greatest amount.

5. Auto-Commissioning Algorithms

These algorithms implement open-loop and closed-loop functional tests through the EMIS platform. The algorithms require continuous real-time communication between the EMIS tool and BAS. The following algorithms have been developed and are described in respective links:

5.1 Open-loop Functional Tests for VAV Terminal Units

5.1.1 System Description

The system undergoing testing is a single duct pressure-independent VAV terminal unit with hydronic reheat. A schematic diagram of the system is shown in Figure 37.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

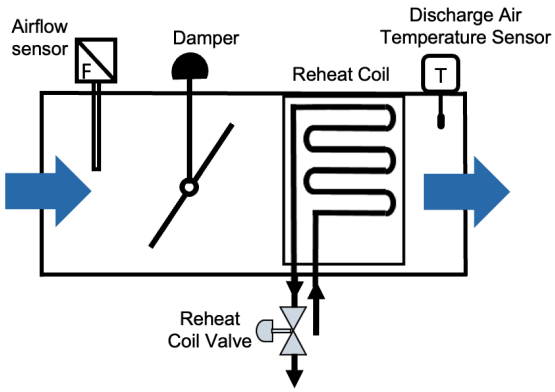


Figure 37: Schematic diagram of a throttling VAV terminal unit with hydronic reheat.

This type of terminal unit consists of a damper and airflow sensor for controlling the volume of air delivered to a zone, and a reheat coil, reheat coil valve, and discharge air temperature sensor that are used to control the temperature of the air delivered to a zone. The open-loop functional tests described in this document are designed to assess the components of the terminal unit by overriding control signals to the damper and reheat valve to force these components to open and close. In doing so the tests help determine whether the components are properly installed and operational. The tests also use changes to the AHU fan operation to obtain additional operational insight for detecting and diagnosing component faults. The tests do not assess feedback control (i.e., the performance of proportional-integral control loops) or control sequences (i.e., transitions between operating modes in response to load or setpoint changes).

5.1.2 Process Description

Figure 38 shows a process flow diagram for the testing and analysis. The process consists of two modules outlined with dashed lines. The *Proactive Functional Test* module involves verifying that the required conditions for the test are satisfied, and executing the test using overrides of the damper, reheat valve, and AHU fan operation. The *Functional Test Analysis* module uses the test data to evaluate rules describing the expected performance in individual test segments. When a functional test fails because a rule defining expected performance is not satisfied, additional analysis is performed in an effort to determine the root cause fault. Two methods for performing this additional analysis are identified, namely, the multi-mode analysis (Lin et al., 2025) and the inverse multi-mode analysis. Further explanation of each of these processes for analyzing a failed test is provided in Section 5.1.4.2.2.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

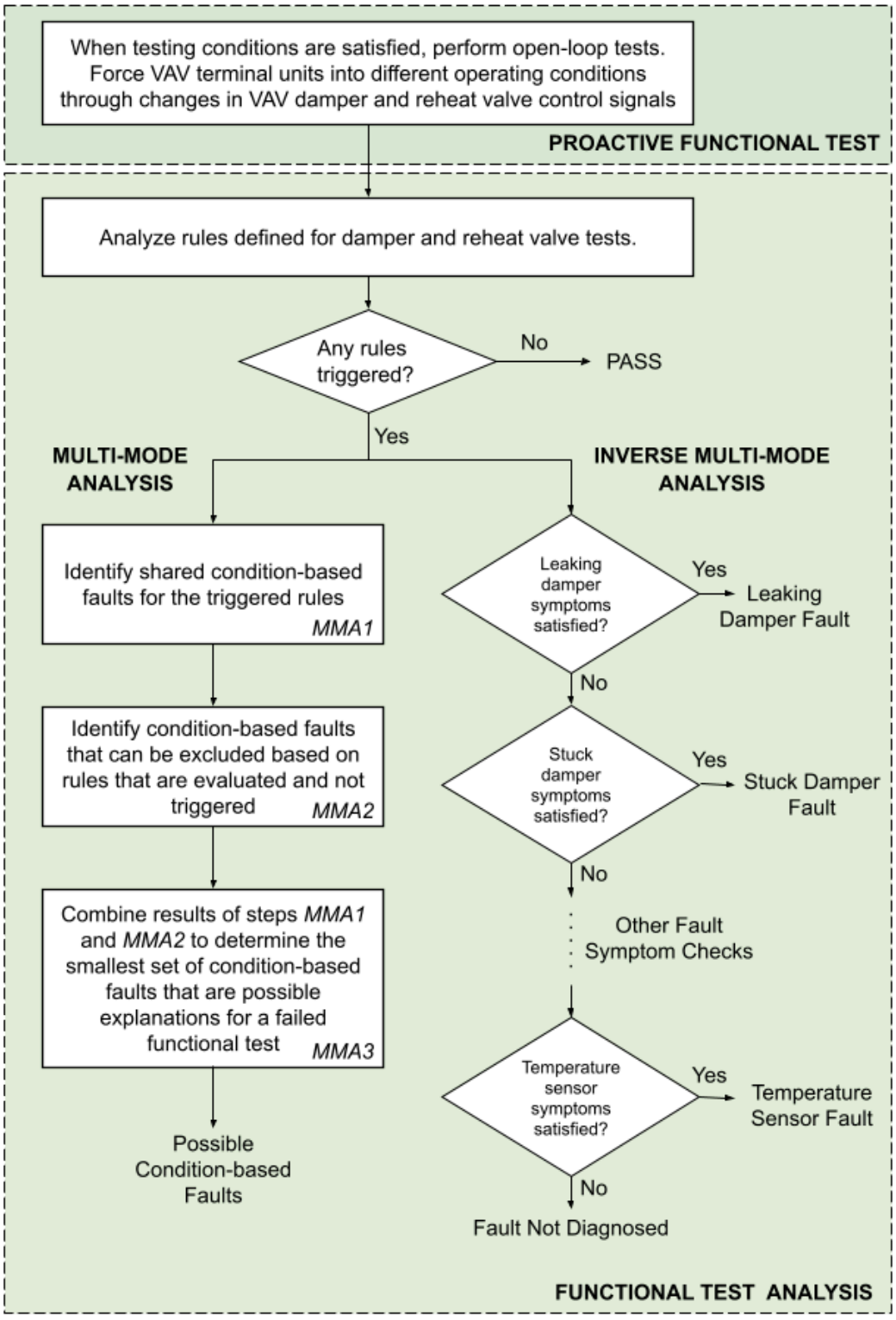


Figure 38: Process flow diagram for the open-loop functional test and analysis for terminal units.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.1.3 Required Data Points

Measurement and control points from the building automation system (BAS) that are used in this test are listed in Table 19. All other variables and parameters used to implement the functional test and analyze its results are listed in Table 20.

Table 19: Data points needed from the building automation system.

Abbreviation	Data point	Data type	Requirements (Read/Write)
AHU_{occ}	AHU occupancy status [0 = unoccupied; 1 = occupied]	Control setting	R
F_{vav}	VAV terminal unit air flow rate [CFM]	Sensor reading	R
$F_{vav,max}$	VAV terminal unit design maximum airflow rate [CFM]	Control parameter	R
$F_{vav,min}$	VAV terminal unit design minimum airflow rate [CFM]	Control parameter	R
P_{sa}	AHU supply air static pressure [in H_2O]	Sensor reading	R
$P_{ss,set}$	AHU supply air static pressure setpoint [in H_2O]	Control parameter	R
T_{da}	VAV discharge air temperature [°F]	Sensor reading	R
T_{hws}	Heating water supply temperature [°F]	Sensor reading	R
T_{sa}	AHU supply air temperature [°F]	Sensor reading	R
$u_{hwp,stat}$	heating water circulating pump status [ON/OFF]	Status feedback signal	R
u_{rf}	AHU return fan control signal [°F]	Control signal	R/W
u_{sf}	AHU supply fan control signal [°F]	Control signal	R/W
$u_{vav,dm}$	VAV damper control signal [0 = closed; 1 = open]	Control signal	R/W
$u_{vav,rh}$	VAV reheat valve control signal [0 = closed; 1 = open]	Control signal	R/W
ZN_{occ}	Zone occupancy status [0 = unoccupied; 1 = occupied]	Control setting	R

Table 20: Variables and parameters used to implement and evaluate the functional test.

Abbreviation	Data point	Data type

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

cause_RDi	[1 x 15] binary array defining possible condition-based faults for the ith (i = 1 to 6) damper rule	Algorithm parameter
cause_RVi	[1 x 15] binary array defining possible condition-based faults for the ith (i = 1 to 5) valve rule	Algorithm parameter
Di	Denotes values associated with the ith damper segment (i = 1-5)	Algorithm variable
e_{Psa}	supply air pressure error ($P_{sa,set}$ - P_{sa}) [in H_2O]	Algorithm variable
excludeCause	[1 x 15] binary array of excluded causes based on rules that are evaluated but not triggered	Algorithm variable
exclude_RDi	[1 x 15] binary array of condition-based faults that can be excluded based on rule RDi (i = 4, 5, 6) in Table 22	Algorithm parameter
exclude_RVi	[1 x 15] binary array of condition-based faults that can be excluded based on rule RVi (i = 1-5) in Table 22	Algorithm parameter
f	fraction of samples in a test segment meeting the static pressure control criterion	Algorithm variable
f_{min}	minimum fraction of samples in a test segment that must meet the static pressure control criterion to conclude that the static pressure is under control	User-defined parameter
$F_{i,j}$	airflow rate for terminal unit damper commanded position i (i = 0, 25, 100% open) and fan speed j (j = Off, Lo, Hi) [CFM]	Variable used to describe fault symptoms
$F_{vav,Di}$	average airflow rate for test segment Di [CFM]	Algorithm variable
$F_{vav,Vi}$	average airflow rate for test segment Vi [CFM]	Algorithm variable
functTestResult	binary variable indicating the outcome of the functional test [Pass or Fail]	Algorithm variable
Ncause	array of possible causes for a failed functional test based on triggered rules and excluded causes from the cases in Table 22	Algorithm variable
NcauseFinal	array of the largest element(s) of array “Ncause” which contains the condition-based fault(s) most likely to be the root cause of the failed test	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

NcauseTrig	[1 x 11] array for counting the number of times individual condition-based faults are identified as possible causes for a triggered rule	Algorithm variable
notTrigRules	list of rules where APAR rules are evaluated and not triggered	Algorithm variable
numRulesDmp	integer number of damper rules	Algorithm parameter
numRulesVlv	Integer number of valve rules	Algorithm parameter
$P_{sa,set,max}$	maximum allowable supply air pressure setpoint during operation [in H_2O]	User-defined parameter
$P_{sa,set,min}$	minimum allowable supply air pressure setpoint during operation [in H_2O]	User-defined parameter
RDi	string variable for damper rule i (i = 1-6)	Algorithm variable
RVi	string variable for reheat valve rule i (i = 1-5)	Algorithm variable
Rule_RDi	Binary variable indicating if the ith damper rule is satisfied (True) or triggered/violated (False)	Algorithm variable
Rule_RVi	Binary variable indicating if the ith valve rule is satisfied (True) or triggered/violated (False)	Algorithm variable
runAnalysisDmp	binary variable indicating if the damper analysis should be executed [True / False]	Algorithm variable
runAnalysisVlv	binary variable indicating if the valve analysis should be executed [True / False]	Algorithm variable
runTestDmp	binary variable indicating if damper test should be executed [True / False]	Algorithm variable
runTestVlv	binary variable indicating if reheat valve test should be executed [True / False]	Algorithm variable
rule	iterator variable used in loop to identify triggered rules in the list “trigRules” and non-triggered rules in the list “notTrigRules”	Algorithm variable
SF_{off}	binary variable indicating if the supply fan satisfies the “off” conditions [True / False]	Algorithm variable
SF_{max}	binary variable indicating if the supply fan satisfies the maximum flow conditions [True / False]	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

SF_{min}	binary variable indicating if the supply fan satisfies the minimum flow conditions [True / False]	Algorithm variable
testSeg	integer variable for identifying individual test segments [1 to 4]	Algorithm variable
testSegCount	integer variable for counting samples in a test segment	Algorithm variable
testSegEndDmp	real array with timestamp corresponding to the end of each damper test segment [1 x 5 array]	Algorithm variable
testSegEndVlv	real array with timestamp corresponding to the end of each reheat valve test segment [1 x 4 array]	Algorithm variable
testSegLabelDmp	integer array with label (“testSeg”) for each damper test segment [1 x 5 array]	Algorithm variable
testSegLabelVlv	integer array with label (“testSeg”) for each reheat valve test segment [1 x 4 array]	Algorithm variable
testSegStartDmp	real array with timestamp corresponding to the start of each damper test segment [1 x 5 array]	Algorithm variable
testSegStartVlv	real array with timestamp corresponding to the start of each reheat valve test segment [1 x 4 array]	Algorithm variable
trigRules	String array of triggered rules [RDi, …, RVi]	Algorithm variable
$T_{hws,min}$	minimum heating water supply temperature for executing the functional tests [°F]	User-defined parameter
ΔT_{des}	design temperature rise across the terminal unit reheat coil [°F]	User-defined parameter
$\Delta T_{i,j}$	temperature rise across the terminal unit reheat coil for valve commanded position i (i = 0, 25, 100% open) and fan speed j (j = Lo, Hi) [°F]	Variable used to describe fault symptoms
ΔT_{Vi}	average temperature rise across the terminal unit reheat coil for test segment Vi [°F]	Algorithm variable
Vi	denotes values associated with the ith valve segment (i = 1-4)	Algorithm variable
Inverse Multi-Mode Diagnoses		
LkgVlv	leaking reheat valve fault	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

LkgDmp	leaking damper fault	Algorithm variable
StkClosedDm	stuck closed damper fault	Algorithm variable
StkClosedVlv	stuck closed reheat valve fault	Algorithm variable
StkOpenDmp	stuck open damper fault	Algorithm variable
StkOpenVlv	stuck open reheat valve fault	Algorithm variable
TempSenFaul	temperature sensor fault	Algorithm variable
VlvActRev	valve actuator reversal fault	Algorithm variable
Thresholds		
ε_{Di}	threshold value/coefficient for damper rule i	User-defined parameter
ε_{LDi}	threshold value/coefficient for symptom i of a leaking damper	User-defined parameter
ε_{LVi}	threshold value/coefficient for symptom i of a leaking valve	User-defined parameter
ε_{SCDi}	threshold value/coefficient for symptom i of a stuck closed damper	User-defined parameter
ε_{SCVi}	threshold value/coefficient for symptom i of a stuck closed valve	User-defined parameter
$\varepsilon_{SF,max}$	threshold value/coefficient for assessing the ability of the supply fan to maintain the static pressure at its maximum setpoint	User-defined parameter
$\varepsilon_{SF,min}$	threshold value/coefficient for assessing the ability of the supply fan to maintain the static pressure at its minimum setpoint	User-defined parameter
$\varepsilon_{SF,off}$	threshold value for assessing if the supply fan is off	User-defined parameter
ε_{SODi}	threshold value/coefficient for symptom i of a stuck open damper	User-defined parameter
ε_{SOVi}	threshold value/coefficient for symptom i of a stuck open valve	User-defined parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

ε_{TSi}	threshold value/coefficient for symptom i of a temperature sensor fault	User-defined parameter
ε_{Vi}	threshold value/coefficient for reheat valve rule i	User-defined parameter
ε_{VARi}	threshold value/coefficient for symptom i of a valve actuator reversal fault	User-defined parameter

5.1.4 Description of the Open-loop Functional Test and Analysis

5.1.4.1 Proactive Functional Test

5.1.4.1.1 Verify Test Conditions

Prior to executing the test, the following conditions are to be verified:

- The occupancy status of the AHU (AHU_{occ}) and zones (ZN_{occ}) used in the test are “True” or “occupied”
- If implemented, static pressure reset algorithm is disabled to avoid interfering with the current test
- Heating water circulation pump is operational as indicated by pump status ($u_{hwp,stat}$) or a similar indicator of pump operation
- Heating water supply temperature (T_{hws}) meets or exceeds its minimum value ($T_{hws,min}$)

Pseudocode verifying that test conditions are satisfied is provided below.

```
# Assign value to T_hws,min (e.g., 120°F)
# Read u_hwp,stat, T_hws, ZN_occ, and AHU_occ and verify test conditions are
#   satisfied
      # if (AHU_occ == “True”
      #   & ZN_occ == “True”
      #   & T_hws >= T_hws,min
      #   & u_hwp,stat == “True”):
            # runTestDmp = “True”
            # runAnalysisDmp = “False”
      # else:
            # runTestDmp = “False”
```

5.1.4.1.2 Perform Open-loop Tests

If *runTestDmp* is “True”, the *Proactive Functional Test* outlined in the process flow diagram in Figure 38 is executed. The test involves overriding the VAV terminal unit damper, the VAV terminal unit reheat valve, and the AHU fan operation using the profiles in the next subsection to assess the performance of individual terminal unit components. Control sequences and logic

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

are not assessed. Note that a group of terminal units can be tested simultaneously using this procedure; however, the description here is for a single terminal unit.

For clarity, the description of the tests is divided into two parts, namely, damper tests and reheat valve tests.

5.1.4.1.2.1 Damper Tests

The damper test overrides the AHU fan operation and VAV terminal unit damper control signal every 15 minutes to provide test data over five test segments. The test segments are as follows:

- Test Segment D1: $u_{vav,dm} = 0$, $u_{sf} = 0$, $u_{rf} = 0$
- Test Segment D2: $u_{vav,dm} = 0$, $P_{sa} = P_{sa,set} = P_{sa,set,min}$
- Test Segment D3: $u_{vav,dm} = 0$, $P_{sa} = P_{sa,set} = P_{sa,set,max}$
- Test Segment D4: $u_{vav,dm} = 1$, $P_{sa} = P_{sa,set} = P_{sa,set,max}$
- Test Segment D5: $u_{vav,dm} = 0.25$, $P_{sa} = P_{sa,set} = P_{sa,set,max}$

where $u_{vav,dm}$ is the control signal to the VAV terminal unit damper (0 = closed, 1 = fully open), u_{sf} is the control signal to the supply fan, u_{rf} is the control signal to the return fan, P_{sa} is the AHU static pressure measurement, $P_{sa,set}$ is the static pressure setpoint, and $P_{sa,set,min}$ and $P_{sa,set,max}$ are minimum and maximum values of the static pressure setpoint used, for instance, in a static pressure reset strategy. Testing profiles of the supply fan static pressure setpoint, VAV terminal unit damper control signal, and VAV terminal unit reheat valve control signal are shown in Figure 39. The control signal and setpoint profiles used in test segments D1-D5 correspond to the values between time = 0-75 min in Figure 39. Note that the test setup in segment D5 will also be used in the reheat valve tests and therefore has a dual designation as test segment V1.

In segment D1, the damper is closed and the AHU fans are off. In segment D2, the damper remains closed and the AHU supply fan is operated to maintain the static pressure at its minimum setpoint. In segment D2 and throughout the remainder of the functional test, the AHU return fan is operated according to its normal closed-loop control strategy (i.e., normally to maintain a small positive building pressure). In segment D3, the damper remains closed and the static pressure setpoint is increased to its maximum value, thereby causing the supply fan speed to increase. In segment D4, the damper is commanded to the fully open position ($u_{vav,dm} = 1$) and the fan control is the same as in segment D3. Finally, in segment D5, the damper is commanded to a partially open position (e.g., 25% open) and fan control is again unchanged.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

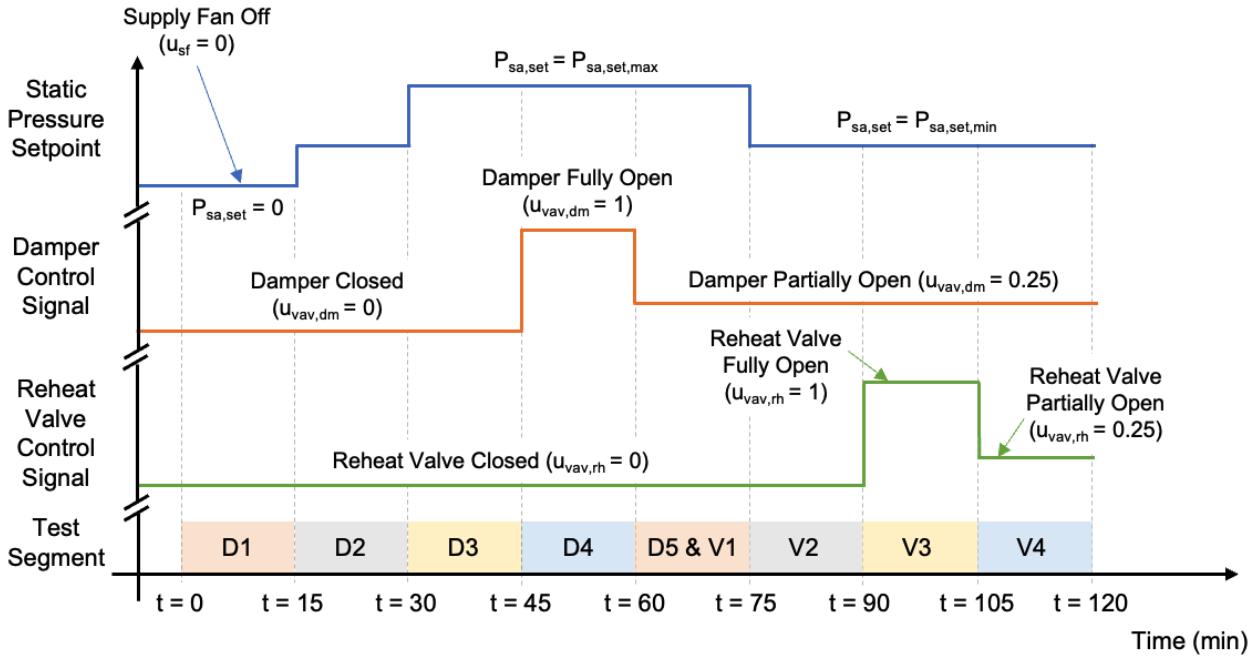


Figure 39: Control signal and setpoint profiles for implementing open-loop functional tests for a VAV terminal unit.

As the test is performed, data are collected and stored at a fixed sampling rate. A sampling rate of 1 minute is preferred and assumed here. The data to be read and stored are AHU_{occ}, F_{vav}, P_{sa}, $P_{sa,set}$, T_{da}, T_{hws}, T_{sa}, u_{rf}, u_{sf}, $u_{vav,dm}$, $u_{vav,rh}$, and ZN_{occ}. Descriptions of all data points are provided in Table 19.

Pseudocode for implementing test segments D1 - D5 follows:

```
# If test conditions are satisfied, run the test and store the
#   following data points with timestamp at each sampling time:
AHUocc,
#   Fvav, Psa, Psa,set, Tda, Thws, Tsa, urf, usf, uvav,dm, uvav,rh, and ZNocc
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
     # if runTestDmp == "True":
          # testSeg = 1
          # while testSeg < 6:
               # testSegStartDmp[testSeg] = currentTime
               # testSegLabelDmp[testSeg] = testSeg
               # testSegCount = 0
               # while testSegCount < 15:  # each segment runs 15
min
                    # if testSeg == 1:
                         # usf = 0   # AHU supply fan off
                         # urf = 0   # AHU return fan off
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                        # u_vav,dm = 0
                        # u_vav,rh = 0
                    # elif testSeg == 2:
                        # P_sa = P_sa,set = P_sa,set,min  # AHU supply fan
                        # u_rf from feedback control
                        # u_vav,dm = 0
                        # u_vav,rh = 0
                    # elif testSeg == 3:
                        # P_sa = P_sa,set = P_sa,set,max  # AHU supply fan
                        # u_rf from feedback control
                        # u_vav,dm = 0
                        # u_vav,rh = 0
                    # elif testSeg == 4:
                        # P_sa = P_sa,set = P_sa,set,max  # AHU supply fan
                        # u_rf from AHU feedback control
                        # u_vav,dm = 1
                        # u_vav,rh = 0
                    # elif testSeg == 5:
                        # P_sa = P_sa,set = P_sa,set,max  # AHU supply fan
                        # u_rf from AHU feedback control
                        # u_vav,dm = 0.25  # adjustable
                        # u_vav,rh = 0
                    # testSegCount += 1
                # testSegEndDmp[testSeg] = currentTime
                # testSeg += 1
        # runTestVlv = "True"  # run reheat valve test
        # runAnalysisVlv = "False"
# else:
        # runTestVlv = "False"
```

5.1.4.1.2.2 Reheat Valve Tests

The reheat valve tests involve overriding the AHU fan operation and VAV terminal unit reheat valve control signal every 15 minutes to provide test data over four test segments. Throughout the four test segments the terminal unit damper is maintained at a fixed position and the AHU return fan is operated according to its normal feedback control strategy (i.e., normally to maintain a small positive building pressure). The test segments are as follows:

- Test Segment V1: $u_{vav,dm} = 0.25$, $u_{vav,rh} = 0$, $P_s = P_{s,set} = P_{s,set,max}$
- Test Segment V2: $u_{vav,dm} = 0.25$, $u_{vav,rh} = 0$, $P_s = P_{s,set} = P_{s,set,min}$
- Test Segment V3: $u_{vav,dm} = 0.25$, $u_{vav,rh} = 1$, $P_s = P_{s,set} = P_{s,set,min}$
- Test Segment V4: $u_{vav,dm} = 0.25$, $u_{vav,rh} = 0.25$, $P_s = P_{s,set} = P_{s,set,min}$

where $u_{vav,rh}$ is the control signal to the VAV terminal unit reheat valve (0 = closed, 1 = fully open). The control signal and setpoint profiles used in test segments V1-V4 correspond to the values between time = 60-120 min in Figure 39.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

In segment V1 (labeled segment D5 in the damper tests), the reheat valve is closed and the supply fan is controlled to maintain the static pressure at its maximum setpoint. In segment V2, the reheat valve remains closed and the static pressure setpoint is decreased to its minimum value in order to decrease the airflow through the terminal unit. In segment V3, the reheat valve is commanded to the fully open position ($u_{vav,rh}$ = 1) and the supply fan continues to operate to maintain the static pressure at its minimum setpoint as in segment V2. Finally, in segment V4, the reheat valve is commanded to a partially open position (e.g., 25% open) and fan control remains as in segment V3.

Pseudocode for implementing test segments V1 - V4 follows. Note that although segments D5 and V1 are identical, the implementation of segment V1 is included in the pseudocode to aid in the understanding of the tests and analysis. This may be particularly useful if the reheat valve tests are used at a test site but the damper tests are not.

```
# If test conditions are satisfied, run the reheat valve test and
#   store the following data points with timestamp at each sampling
#   time: AHU_occ, F_vav, P_sa, P_sa,set, T_da, T_hws, T_sa, u_rf, u_sf, u_vav,dm, u_vav,rh,
and
#   ZN_occ
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
     # if runTestVlv == "True":
          # testSeg = 1
          # while testSeg < 5:
               # testSegStartVlv[testSeg] = currentTime
               # testSegLabelVlv[testSeg] = testSeg
               # testSegCount = 0
               # while testSegCount < 15:  # each segment runs 15
min
                    # if testSeg == 1:
                         # P_sa = P_sa,set = P_sa,set,max  # AHU supply fan
                         # u_rf from AHU feedback control
                         # u_vav,dm = 0.25
                         # u_vav,rh = 0
                    # elif testSeg == 2:
                         # P_sa = P_sa,set = P_sa,set,min  # AHU supply fan
                         # u_rf from AHU feedback control
                         # u_vav,dm = 0.25
                         # u_vav,rh = 0
                    # elif testSeg == 3:
                         # P_sa = P_sa,set = P_sa,set,min  # AHU supply fan
                         # u_rf from AHU feedback control
                         # u_vav,dm = 0.25
                         # u_vav,rh = 1
                    # elif testSeg == 4:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                    # P_sa = P_sa,set = P_sa,set,min  # AHU supply fan
                    # u_rf from AHU feedback control
                    # u_vav,dm = 0.25
                    # u_vav,rh = 0.25
                # testSegCount += 1
            # testSegEndVlv[testSeg] = currentTime
            # testSeg += 1
    # runAnalysisDmp = "True"  # run damper analysis
```

5.1.4.2 Functional Test Analysis

This section describes a set of rules that can be used to analyze data obtained from the functional tests. For clarity, the description of the tests and the accompanying pseudocode is limited to the main steps needed to implement rules pertaining to the damper tests and reheat valve tests. Checks of the AHU supply fan operation to confirm that the appropriate conditions for each test segment are satisfied are described in the Appendix (Section 5.1.5).

5.1.4.2.1 Rule Set

5.1.4.2.1.1 Rules for Damper Tests

Rules that are applicable to the damper tests are described below. For each rule, a list of condition-based faults is provided that are considered possible explanations for a triggered rule.

Test Segment D1

In segment D1, the terminal unit damper is closed and the AHU fans are off. There should be no airflow through the terminal unit and the flow measurement is expected to be near zero. There is one test criterion in segment D1 given by rule RD1:

$$\text{RD1: } |F_{vav,D1}| \leq \varepsilon_{D1}$$

where $F_{vav,D1}$ is the measured airflow rate in segment D1 and ε_{D1} is the threshold. Faults that are possible explanations for rule RD1 being triggered are:

- Airflow sensor fault

Test Segment D2

In segment D2, the terminal unit damper is closed and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its minimum setpoint. With the damper closed, the measured airflow rate is expected to be near zero. There is one test criterion in segment D2 given by rule RD2:

$$\text{RD2: } |F_{vav,D2}| \leq \varepsilon_{D2}\, F_{vav,min}$$

where $F_{vav,D2}$ is the measured airflow rate in segment D2, $F_{vav,min}$ is the design minimum airflow setpoint for the terminal unit, and ε_{D2} is a fractional threshold with a suggested value between 0 and 0.5. If the absolute value of the measured airflow rate exceeds some fraction of $F_{vav,min}$,

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

where the fraction is defined by ε_{D2}, the rule is not satisfied. Faults that are possible explanations for rule RD2 being triggered are:

- Airflow sensor fault
- Leaking damper
- Stuck damper
- Damper shaft incorrectly aligned with actuator (e.g., damper not closed even when actuator is in fully closed position)
- Damper range limited (i.e., damper restricted from opening and closing fully)
- Actuator direction reversal (i.e., damper is fully open rather than fully closed for $u_{vav,dm}$ = 0)

Note that rules RD1 and RD2 are identical in form; however, it is anticipated that ε_{D2} will be larger than ε_{D1} to allow for a small amount of leakage past the closed damper when the AHU fans are operating.

Test Segment D3

Operation in segment D3 is identical to that in segment D2 except that the AHU supply fan is now controlled to maintain the static pressure to its maximum setpoint. As before, with the damper closed, the measured airflow rate is expected to be near zero. If there is leakage past the closed damper, data from this segment combined with data from segment D2 are expected to be useful for diagnosing the fault. There is one test criterion in segment D3 given by rule RD3:

$$\text{RD3:} \quad | F_{vav,D3} | \leq \varepsilon_{D3} F_{vav,min}$$

where $F_{vav,D3}$ is the measured airflow rate in segment D3 and ε_{D3} is a fractional threshold with a suggested value between 0 and 0.5. Because it is expected that there will be some leakage past the closed damper and that leakage will be greater at higher inlet pressures, ε_{D3} should be set to a value that is greater than ε_{D2}. Faults that are possible explanations for rule RD3 being triggered are:

- Airflow sensor fault
- Leaking damper
- Stuck damper
- Damper shaft incorrectly aligned with actuator (e.g., damper not closed even when actuator is in fully closed position)
- Damper range limited (i.e., damper restricted from opening and closing fully)
- Actuator direction reversal (i.e., damper is fully open rather than fully closed for $u_{vav,dm}$ = 0)

Test Segment D4

In segment D4, the terminal unit damper is fully open and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its maximum setpoint. The measured airflow rate through the terminal unit is expected to be close

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

to, or above, the design maximum airflow setpoint. There is one test criterion in segment D4 given by rule RD4:

$$\text{RD4:}\quad F_{vav,max} - F_{vav,D4} \leq \varepsilon_{D4}\, F_{vav,max}$$

where $F_{vav,D4}$ is the measured airflow rate in segment D4, $F_{vav,max}$ is the design maximum airflow setpoint for the terminal unit, and ε_{D4} is a threshold value between 0 and 0.1. Note that any airflow rate $F_{vav,D4}$ that exceeds $F_{vav,max}$ will satisfy rule RD4. If rule RD4 is not satisfied, the measured airflow rate is lower than expected with the damper fully open. Faults that are possible explanations for rule RD4 being triggered are:

- Airflow sensor fault
- Stuck damper
- Blockage in the ductwork or terminal unit
- Unbalanced airflow distribution network
- Damper shaft incorrectly aligned with actuator (e.g., damper not fully open even when actuator is in fully open position)
- Damper range limited (i.e., damper restricted from opening and closing fully)
- Actuator direction reversal (i.e., damper is fully closed rather than fully open for $u_{vav,dm}$ = 1)
- Design fault (e.g., terminal unit is undersized)

Test Segment D5

In segment D5, the terminal unit damper is partially open and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its maximum setpoint. The measured airflow rate is expected to be greater than the airflow rate in segment D3 when the damper is closed, but less than the airflow rate in segment D4 when the damper is fully open. Thus there are two test criterion in segment D5 given by rules RD5 and RD6:

$$\text{RD5:}\quad F_{vav,D5} - F_{vav,D3} \geq \varepsilon_{D5}\,(F_{vav,max} - F_{vav,min})$$
$$\text{RD6:}\quad F_{vav,D4} - F_{vav,D5} \geq \varepsilon_{D6}\,(F_{vav,max} - F_{vav,min})$$

where $F_{vav,D5}$ is the measured airflow rate in segment D5, $F_{vav,min}$ is the design minimum airflow setpoint for the terminal unit, and ε_{D5} and ε_{D6} are threshold coefficients with suggested values between 0 and 0.1. If rule RD5 is not satisfied, the measured airflow rate in segment D5 when the damper is partially open is lower than expected in comparison to segment D3 when the damper is closed. Faults that are possible explanations for rule RD5 being triggered are:

- Airflow sensor fault
- Leaking damper (i.e., flow through the closed damper in segment D3 is comparable to flow when the damper is partially open in segment D5)
- Stuck damper
- Blockage in the ductwork or terminal unit
- Unbalanced airflow distribution network

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Damper shaft incorrectly aligned with actuator (i.e., damper is actually less open in segment D5 than in segment D3)
- Damper range limited (i.e., damper restricted from opening and closing fully)
- Actuator direction reversal (i.e., damper is actually less open in segment D5 than in segment D3)

If rule RD6 is not satisfied, the measured airflow rate in segment D5 is higher than expected in comparison to segment D4 when the damper is fully open. Faults that are possible explanations for rule RD6 being triggered are:

- Airflow sensor fault
- Stuck damper
- Blockage in the ductwork or terminal unit
- Unbalanced airflow distribution network
- Damper shaft incorrectly aligned with actuator (i.e., damper is actually less open in segment D4 than in segment D5)
- Damper range limited (i.e., damper restricted from opening and closing fully)
- Actuator direction reversal (i.e., damper is actually less open in segment D4 than in segment D5)

Pseudocode for implementing the damper test analysis follows. Note that the analysis in a test segment is only performed if the supply fan is operating as intended. A description of criterion for evaluating supply fan operation in each test segment is provided in the Appendix (Section 5.1.5).

```
# For each test segment D1-D5, evaluate the segment rule(s) using
#   measurement averages for the last 10 minutes of the segment
# If runAnalysisDmp == "True":
# Set user-defined thresholds and configure design parameters
      # ε_D1 = 20 CFM  # threshold for rule RD1 [0 - 50 CFM]
      # ε_D2 = 0.2     # threshold for rule RD2 [0 - 0.5]
      # ε_D3 = 0.3     # threshold for rule RD3 [0 - 0.5], ε_D3 > ε_D2
      # ε_D4 = 0.1     # threshold for rule RD3 [0 - 0.1]
      # ε_D5 = 0.1     # threshold for rule RD4 [0 - 0.1]
      # ε_D6 = 0.1     # threshold for rule RD4 [0 - 0.1]
      # F_vav,min = design minimum airflow rate from design documents
      # F_vav,max = design maximum airflow rate from design documents
# Initialize all rules to be satisfied (i.e., not triggered)
      # Rule_RDi = "True" for i = 1-6
# Process test segment data
      # testSeg = 1
      # trigRules = []    # Initialize array of triggered rules
      # notTrigRules = [] # Initialize array of non-triggered rules
      # while testSeg < 5:
            # Calculate and store segment average for F_vav,Di
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#   (i = testSeg) based on final 10 segment samples
# Evaluate segment rules
# if testSeg == 1 & SF_off == "True": # SF_off indicates supply
                                      # fan is off
     # if |F_vav,D1| <= ε_D1:
          # notTrigRules = notTrigRules + RD1
     # else:
          # Rule_RD1 = "False"
          # trigRules = trigRules + RD1
     # testSeg += 1
# elif testSeg == 2 & SF_min == "True": # SF_min indicates P_s
                                        #  is controlled
     # if  |F_vav,D2| <= ε_D2 F_vav,min:
          # notTrigRules = notTrigRules + RD2
     # else:
          # Rule_RD2 = "False"
          # trigRules = trigRules + RD2
     # testSeg += 1
# elif testSeg == 3 & SF_max == "True": # SF_max indicates P_s
                                        #  is controlled
     # if  |F_vav,D3| <= ε_D3 F_vav,min:
          # notTrigRules = notTrigRules + RD3
     # else:
          # Rule_RD3 = "False"
          # trigRules = trigRules + RD3
     # testSeg += 1
# elif testSeg == 4 & SF_max == "True":
     # if F_vav,max - F_vav,D4 ≤ ε_D4 F_vav,max:
          # notTrigRules = notTrigRules + RD4
     # else:
          # Rule_RD4 = "False"
          # trigRules = trigRules + RD4
     # testSeg += 1
# elif testSeg == 5 & SF_max == "True":
     # if F_vav,D5 - F_vav,D3 ≥ ε_D5 (F_vav,max - F_vav,min):
          # notTrigRules = notTrigRules + RD5
     # else:
          # Rule_RD5 = "False"
          # trigRules = trigRules + RD5
     # if F_vav,D4 - F_vav,D5 ≥ ε_D6 (F_vav,max - F_vav,min):
          # notTrigRules = notTrigRules + RD5
     # else:
          # Rule_RD6 = "False"
          # trigRules = trigRules + RD6
     # testSeg += 1
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# runAnalysisVlv = "True"  # run damper analysis
```

5.1.4.2.1.2 Rules for Reheat Valve Tests

Rules that are applicable to the reheat valve tests are described below. Recall that for all reheat valve tests, the terminal unit damper is maintained at a fixed, partially open position.

Test Segment V1

In segment V1, the reheat valve is commanded to the closed position and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its maximum setpoint. There should be no hot water flowing through the reheat coil. Thus the temperature rise across the reheat coil should be nearly zero. There is one test criterion in segment V1 given by rule RV1:

$$\text{RV1: } |\Delta T_{V1}| \leq \varepsilon_{V1}$$

where $\Delta T_{Vi} = T_{da,Vi} - T_{sa,Vi}$ is the temperature rise across the reheat coil in segment Vi (i = 1 to 4) found by subtracting the AHU supply air temperature T_{sa} measured in segment Vi from the terminal unit discharge air temperature T_{da} measured in segment Vi. If the absolute value of the temperature rise exceeds the threshold ε_{V1}, the rule is violated. Faults that are possible explanations for rule RV1 being triggered are:

- Leaking reheat valve
- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Actuator direction reversal (i.e., valve is actually fully open rather than fully closed)

Test Segment V2

In segment V2, the reheat valve remains in the closed position and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its minimum setpoint. There should be no hot water flowing through the reheat coil and the discharge air temperature from the terminal unit T_{da} should be nearly the same as the AHU supply air temperature T_{sa}. The same test criterion used in segment V1 applies in segment V2 and is given by rule RV2:

$$\text{RV2: } |\Delta T_{V2}| \leq \varepsilon_{V2}$$

where ΔT_{V2} is the temperature rise across the reheat coil in segment V2 and the threshold is ε_{V2}. This rule is intended to help diagnose leaking valve faults, as ΔT_{V2} is expected to be larger than ΔT_{V1} due to the lower airflow rate in segment V2 compared to segment V1. Faults that are possible explanations for rule RV2 being triggered are:

- Leaking reheat valve
- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Actuator direction reversal (i.e., valve is actually fully open rather than fully closed)

Test Segment V3

In segment V3, the reheat valve is fully open and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its minimum setpoint. This is done because this mimics a common sequencing strategy for VAV terminal units wherein the airflow rate to the zone is controlled to its design minimum value in the heating mode. There is one test criterion in segment V3 given by rule RV3:

$$\text{RV3:}\ \ \Delta T_{V3} \geq \varepsilon_{V3}\, \Delta T_{des}$$

where ΔT_{V3} is the temperature rise across the reheat coil in segment V3, ΔT_{des} is the design temperature rise across the coil, and a threshold ε_{V3} between 0.7 and 0.9 is suggested. Thus, for fault free operation it is expected that the temperature rise across the coil will exceed some fraction of T_{des} with the fraction given by ε_{V3}. Faults that are possible explanations for rule RV3 being triggered are:

- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Actuator direction reversal (i.e., valve is actually fully closed rather than fully open)
- Inadequate heating water flow rate (e.g., due to blockage/obstruction in the heating water piping or reheat coil, poor balancing, design error, etc.)

Test Segment V4

In segment V4, the reheat valve is partially open and the AHU fans operate according to their normal feedback control logic with the supply fan controlling the static pressure to its minimum setpoint as in segment V3. There are two test criterion in segment V4 given by rules RV4 and RV5:

$$\text{RV4:}\ \ \Delta T_{V4} - \Delta T_{V2} \geq \varepsilon_{V4}\, \Delta T_{des}$$
$$\text{RV5:}\ \ \Delta T_{V3} - \Delta T_{V4} \geq \varepsilon_{V5}\, \Delta T_{des}$$

where ΔT_{V4} is the temperature rise across the reheat coil in segment V4, and ε_{D4} and ε_{D5} are threshold coefficients with suggested values between 0.05 and 0.2. If rule RV4 is not satisfied, the temperature rise in segment V4 when the valve is partially open is lower than expected in comparison to segment V2 when the valve is closed. Faults that are possible explanations for rule RV4 being triggered are:

- Leaking reheat valve (i.e., flow through the closed valve in segment V2 is comparable to flow when the valve is partially open in segment V4)
- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Actuator direction reversal (i.e., valve is actually less open in segment V4 than in segment V2)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Inadequate heating water flow rate (e.g., due to blockage/obstruction in the heating water piping or reheat coil, poor balancing, design error, etc.)

If rule RV5 is not satisfied, the temperature rise in segment V4 is higher than expected in comparison to segment V3 when the valve is fully open. Faults that are possible explanations for rule RV5 being triggered are:

- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Actuator direction reversal (i.e., valve is actually more open in segment V4 than in segment V3)
- Inadequate heating water flow rate (e.g., due to blockage/obstruction in the heating water piping or reheat coil, poor balancing, design error, etc.)

Pseudocode for implementing the reheat valve test analysis follows. Note that the analysis in a test segment is only performed if the supply fan is operating as intended. A description of criterion for evaluating supply fan operation in each test segment is provided in the Appendix (Section 5.1.5).

```
# For each test segment V1-V4, evaluate the segment rule(s) using
#   measurement averages for the last 10 minutes of the segment
# If runAnalysisVlv == "True":
# Set user-defined thresholds and configure design parameters
     # εV1 = 5°F  # threshold for rule RV1; [3-10°F] recommended
     # εV2 = 5°F  # threshold for rule RV2; [3-10°F] recommended
     # εV3 = 0.8  # threshold for rule RV3; [0.7 - 0.9] recommended
     # εV4 = 0.1  # threshold for rule RV4; [0.05-0.2] recommended
     # εV5 = 0.1  # threshold for rule RV5; [0.05-0.2] recommended
     # ΔTdes = design temperature rise across the reheat coil
# Initialize all rules to be satisfied (i.e., not triggered)
     # Rule_RVi = "True" for i = 1-5
# Process test segment data
     # testSeg = 1
     # while testSeg < 5:
          # Calculate and store segment averages for Tda,Vi and Tsa,Vi
          #   (i = testSeg) based on final 10 segment samples
          # Evaluate segment rules storing triggered rules in the
          #   array trigRules and non-triggered rules in the array
          #   notTrigRules which are initialized for segments D1-D5
          # ΔTVi = Tda,Vi - Tsa,Vi  # Store calculated ΔTVi
          # if testSeg == 1 & SFmax == "True" # SFmax indicates Ps
                                              #  is controlled
               # if |ΔTV1| ≤ εV1:
                    # notTrigRules = notTrigRules + RV1
               # else:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                    # Rule_RV1 = "False"
                    # trigRules = trigRules + RV1
               # testSeg += 1
          # elif testSeg == 2 & SF_min == "True": # SF_min indicates P_s
                                                  #  is controlled
               # if |ΔT_V2| ≤ ε_V2 :
                    # notTrigRules = notTrigRules + RV2
               # else:
                    # Rule_RV2 = "False"
                    # trigRules = trigRules + RV2
               # testSeg += 1
          # elif testSeg == 3 & SF_min == "True":
               # if ΔT_V3 ≥ ε_V3 ΔT_des:
                    # notTrigRules = notTrigRules + RV3
               # else:
                    # Rule_RV3 = "False"
                    # trigRules = trigRules + RV3
               # testSeg += 1
          # elif testSeg == 4 & SF_min == "True":
               # if ΔT_V4 - ΔT_V1 ≥ ε_V4 ΔT_des:
                    # notTrigRules = notTrigRules + RV4
               # else:
                    # Rule_RV4 = "False"
                    # trigRules = trigRules + RV4
               # if ΔT_V3 - ΔT_V4 ≥ ε_V4 ΔT_des:
                    # notTrigRules = notTrigRules + RV5
               # else:
                    # Rule_RV5 = "False"
                    # trigRules = trigRules + RV5
               # testSeg += 1
     # if trigRules = []:
     #    functTestResult = "Pass"
     # else:
     #    functTestResult = "Fail"
     # Perform multi-mode analysis or inverse multi-mode analysis
for
     #   root cause fault diagnosis
```

5.1.4.2.2 Root Cause Diagnostics

Two processes for analyzing the results of failed tests and performing root cause diagnostics have been developed and are shown in the process flow diagram in Figure 38. The first is referred to as the multi-mode analysis and involves combining fault symptoms revealed through triggered rules in different test segments with inferences that can be made based on rules that

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

are not triggered in order to find the root cause fault. Thus, symptoms are used to identify the fault. This multi-mode analysis is described in Lin et al. 2025[11].

The second process is referred to as the inverse multi-mode analysis and involves identifying common faults for the system of interest and listing the symptoms that are expected for each fault based on the test performed. Thus, in this case the fault is used to identify the expected symptoms. This approach is proposed for VAV terminal units because of the relative simplicity of the equipment and controls.

5.1.4.2.2.1 Multi-Mode Analysis

The description of the test segments in Section 5.1.4.2.1 includes one or more operational rules that apply to each segment as well as one or more condition-based faults that are possible explanations why each rule might be triggered. In all, 11 rules were identified. The possible faults that could explain why each rule might be triggered are listed in Table 21. Fifteen (15) possible faults are identified, with nearly all applying to three or more rules.

Table 21: Condition-based fault explanations for triggered rules.

[11] Lin, G., House, J., Chen, Y., Granderson, J., and Zhang, W. 2025. Active Multi-mode Data Analysis to Improve Fault Diagnosis in AHUs. Accepted for Energy and Buildings.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rules	Explanations for Rule Violations														
	1 - Airflow Sensor Error	2 - Leaking Damper	3 - Stuck Damper	4 - Damper Shaft/Actuator Misaligned	5 - Damper Range Limited	6 - Damper Actuator Direction Reversal	7 - Airflow Obstruction	8 - Unbalanced Airflow Distribution Network	9 - Undersized Terminal Unit	10 - Leaking Valve	11 - Stuck Valve	12 - Discharge Air Temperature Sensor Fault	13 - AHU Supply Air Temperature Sensor Fault	14 - Valve Actuator Direction Reversal	15 - Inadequate Heating Water Flow Rate
RD1: $\lvert F_{vav,D1} \rvert \le \varepsilon_{D1}$	X														
RD2: $\lvert F_{vav,D2} \rvert \le \varepsilon_{D2} F_{vav,min}$	X	X	X	X	X	X									
RD3: $\lvert F_{vav,D3} \rvert \le \varepsilon_{D3} F_{vav,min}$	X	X	X	X	X	X									
RD4: $F_{vav,max} - F_{vav,D4} \le \varepsilon_{D4} F_{vav,max}$	X		X	X	X	X	X	X	X						
RD5: $F_{vav,D5} - F_{vav,D3} \ge \varepsilon_{D5} (F_{vav,max} - F_{vav,min})$	X	X	X	X	X	X	X	X							
RD6: $F_{vav,D4} - F_{vav,D5} \ge \varepsilon_{D6} (F_{vav,max} - F_{vav,min})$	X		X	X	X	X	X	X							
RV1: $\lvert \Delta T_{V1} \rvert \le \varepsilon_{V1}$										X	X	X	X	X	
RV2: $\lvert \Delta T_{V2} \rvert \le \varepsilon_{V2}$										X	X	X	X	X	
RV3: $\Delta T_{V3} \ge \varepsilon_{V3} \Delta T_{des}$											X	X	X	X	X
RV4: $\Delta T_{V4} - \Delta T_{V2} \ge \varepsilon_{V4} \Delta T_{des}$										X	X	X	X	X	X
RV5: $\Delta T_{V3} - \Delta T_{V4} \ge \varepsilon_{V5} \Delta T_{des}$											X	X	X	X	X

The multi-mode analysis provides a process for combining the results of the various test segments to determine the smallest possible set of condition-based faults that are possible explanations for a failed functional test. If it is assumed that no more than one fault is present during a functional test and if multiple rules are triggered during the test, then the triggered rules must share at least one condition-based fault among their possible fault explanations. Stated another way, each triggered rule has a set of condition-based faults as possible explanations and the intersection of these sets should contain the actual condition-based fault. The first step in the multi-mode analysis is to identify the shared condition-based faults for the triggered rules. This step is labeled MMA1 in Figure 38.

Table 21 lists 15 condition-based faults that are possible explanations for one or more triggered rules. To find the intersection of the set of possible explanations for each triggered rule, a 1x15 array *NcauseTrig* is defined and initialized with all elements equal to 0. The second dimension of this array corresponds to the total number of condition-based faults identified in Table 21. This array will be used to count the number of times each condition-based fault is identified as a possible explanation in the triggered rules. Next, a 1x15 array *cause_rule* (where *rule* =

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

RD1-RD6 and RV1-RV6) is defined for each rule, where an element of the array is assigned the value 1 if the condition-based fault corresponding to that element is a possible cause for a particular rule being triggered, and the element is assigned the value 0 otherwise. As an example, referring to Table 21, Rule RD2 has six condition-based faults as possible explanations for why this rule might be triggered. The six condition-based faults are:

1 - Airflow sensor error
2 - Leaking damper
3 - Stuck damper
4 - Damper shaft/actuator misaligned
5 - Damper range limited
6 - Damper actuator direction reversal

Thus, the array *cause_rule* where *rule* = RD1 (i.e., *cause_RD1*) is defined as:

```
cause_RD1 = [1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
```

where we note that elements 1-6 are set to 1. Arrays *cause_RD2* through *cause_RD6* and *cause_RV1* through *cause_RV5* are defined in the same manner.

Next, for each rule in the arrays *trigRules*, the array *cause_rule* is added to the array *NcauseTrig*. This step counts the number of times each condition-based fault is identified as a possible explanation for a triggered rule. When this step has been completed, there should be one or more elements in *NcauseTrig* that are equal to the number of rules in the array *trigRules*. These elements correspond to the condition-based faults that are the possible causes for the failed functional test.

Pseudocode implementing step MMA1 follows:

```
# Initialize 1x15 array for counting possible causes for triggered
#   rules
      #     NcauseTrig = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x15 array of possible causes for each of the 11 rules
      #     cause_RD1 = [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #     cause_RD2 = [1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
      #     cause_RD3 = [1,1,1,1,1,1,0,0,0,0,0,0,0,0,0]
      #     cause_RD4 = [1,0,1,1,1,1,1,1,1,0,0,0,0,0,0]
      #     cause_RD5 = [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0]
      #     cause_RD6 = [1,0,1,1,1,1,1,1,1,0,0,0,0,0,0]
      #     cause_RV1 = [0,0,0,0,0,0,0,0,0,1,1,1,1,1,0]
      #     cause_RV2 = [0,0,0,0,0,0,0,0,0,1,1,1,1,1,0]
      #     cause_RV3 = [0,0,0,0,0,0,0,0,0,0,1,1,1,1,1]
      #     cause_RV4 = [0,0,0,0,0,0,0,0,0,1,1,1,1,1,1]
      #     cause_RV5 = [0,0,0,0,0,0,0,0,0,0,1,1,1,1,1]
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      #     i = 1
      #     numRulesDmp = 6
# Count the possible causes for triggered damper rules in trigRules
      #     While i <= numRulesDmp:
            #     if Rule_RDi in trigRules:
                  #     NcauseTrig = NcauseTrig + cause_RDi
            #     i += 1
      #     i = 1
      #     numRulesVlv = 5
# Count the possible causes for triggered valve rules in trigRules
      #     While i <= numRulesVlv:
            #     if Rule_RVi in trigRules:
                  #     NcauseTrig = NcauseTrig + cause_RVi
            #     i += 1
      #     goto "MMA2"
```

Next, rules that are evaluated and not triggered are used to identify faults that can be excluded as possible causes for a failed functional test. For example, if either of the rules in segment D5 are satisfied (not triggered), this is sufficient evidence to infer that the damper is not stuck and that the damper is rotating in the appropriate direction as its control signal is changed. The rules that can be used in this way are listed in Table 22 along with the condition-based fault or faults that can be removed in each case. This step is labeled MMA2 in Figure 38.

Table 22: Condition-based faults that can be removed as possible causes of a failed functional test based on damper and reheat valve rules that are evaluated in the test and not triggered.

Test Segment	Rule Evaluated and not Triggered	Condition-based Fault(s) Removed *
D4	RD4: $F_{vav,max} - F_{vav,D4} \leq \varepsilon_{D4} F_{vav,max}$	Airflow obstruction [7] Undersized terminal unit [9]
D5	RD5: $F_{vav,D5} - F_{vav,D3} \geq \varepsilon_{D5} (F_{vav,max} - F_{vav,min})$	Stuck damper [3] Damper actuator direction reversal [6]
D5	RD6: $F_{vav,D4} - F_{vav,D5} \geq \varepsilon_{D6} (F_{vav,max} - F_{vav,min})$	Stuck damper [3] Damper actuator direction reversal [6]
V1	RV1: $\lvert \Delta T_{V1} \rvert \leq \varepsilon_{V1}$	Discharge air temperature sensor fault [12] AHU supply air temperature sensor fault [13]
V2	RV2: $\lvert \Delta T_{V2} \rvert \leq \varepsilon_{V2}$	Discharge air temperature sensor fault [12] AHU supply air temperature sensor fault [13]
V3	RV3: $\Delta T_{V3} \geq \varepsilon_{V3} \Delta T_{des}$	Inadequate heating water flow rate [15]
V4	RV4: $\Delta T_{V4} - \Delta T_{V2} \geq \varepsilon_{V4} \Delta T_{des}$	Stuck valve [11] Valve actuator direction reversal [14]
V4	RV5: $\Delta T_{V3} - \Delta T_{V4} \geq \varepsilon_{V5} \Delta T_{des}$	Stuck valve [11]

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		Valve actuator direction reversal [14]

* The number in brackets in the last column corresponds to the explanation number for the condition-based fault from Table 21.

To implement this step, a 1x15 array *excludeCause* is defined and initialized with all elements equal to 0. Next, a 1x15 array *exclude_rule* is defined for each fault exclusion rule in Table 22. An element of the array is assigned the value 1 if the condition-based fault corresponding to that element can be removed as a possible cause for a failed functional test, and the element is assigned the value 0 otherwise. As an example, the array *exclude_RD4* (i.e., *rule* is set equal to RD4 corresponding to Rule RD4 described in Segment D4) is defined as

```
exclude_RD4 = [0,0,0,0,0,0,1,0,1,0,0,0,0,0,0]
```

because, as shown in Table 22, when Rule RD4 is evaluated and not triggered, the airflow obstruction fault (explanation #7 in Table 21) and undersized terminal unit fault (explanation #9 in Table 21) can be removed as possible causes for the failed functional test. Thus, elements 7 and 9 of the array exclude_RD4 are set to 1. Similar arrays are defined for the remaining fault exclusions in Table 22. Next, for each rule in the array *notTrigRules*, the array *exclude_rule* is added to the array *excludeCause*.

When step MMA2 has been completed, the elements of array *excludeCause* that are greater than or equal to 1 will indicate the condition-based faults that can be removed as a possible cause for the failed functional test.

Pseudocode implementing step MMA2 follows:

```
# Initialize 1x15 array for identifying excluded causes based on
rules
#   that are evaluated and not triggered
     # excludeCause = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Initialize a 1x15 array of excluded causes for Rules RD1-RD6 and
#   RV1-RV5
     # for i = 1:6:
          # exclude_RDi = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
     # for i = 1:5:
          # exclude_RVi = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x15 array of excluded causes for rules in Table 22
     # Rule RD4 evaluated and not triggered
     #    exclude_RD4 = [0,0,0,0,0,0,1,0,1,0,0,0,0,0,0]
     # Rule RD5 evaluated and not triggered
     #    exclude_RD5 = [0,0,1,0,0,1,0,0,0,0,0,0,0,0,0]
     # Rule RD6 evaluated and not triggered
     #    exclude_RD6 = [0,0,1,0,0,1,0,0,0,0,0,0,0,0,0]
     # Rule RV1 evaluated and not triggered
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      #     exclude_RV1 = [0,0,0,0,0,0,0,0,0,0,0,1,1,0,0]
      # Rule RV2 evaluated and not triggered
      #     exclude_RV2 = [0,0,0,0,0,0,0,0,0,0,0,1,1,0,0]
      # Rule RV3 evaluated and not triggered
      #     exclude_RV3 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      # Rule RV4 evaluated and not triggered
      #     exclude_RV4 = [0,0,0,0,0,0,0,0,0,0,1,0,0,1,0]
      # Rule RV5 evaluated and not triggered
      #     exclude_RV5 = [0,0,0,0,0,0,0,0,0,0,1,0,0,1,0]
# Define loop variables
      #     i = 1
      #     numRulesDmp = 6
# Update array excludeCause using damper rules in array notTrigRules
      #     While i <= numRulesDmp:
            #     if Rule_RDi in notTrigRules:
                  #     excludeCause = excludeCause + exclude_RDi
            #     i += 1
      #     i = 1
      #     numRulesVlv = 5
# Update array excludeCause using valve rules in array notTrigRules
      #     While i <= numRulesVlv:
            #     if Rule_RVi in notTrigRules:
                  #     excludeCause = excludeCause + exclude_RVi
            #     i += 1
      #     goto "MMA3"
```

Next, in step MMA3, results from the previous two steps are combined to determine the smallest set of condition-based faults that are possible explanations for the failed functional test. More specifically, this procedure removes any faults identified in step MMA2 from the shared condition-based faults identified in step MMA1.

A 1x15 array *Ncause* is determined by subtracting the array *excludeCause* (step MMA2) from the array *NcauseTrig* (step MMA1). The element(s) of *Ncause* with the largest magnitude identifies the condition-based fault(s) that is the cause of the failed functional test. Finally, an array *NcauseFinal* is initialized and used to store the largest element(s) of array *Ncause*. *NcauseFinal* will contain the smallest set of condition-based faults that are possible explanations for a failed functional test.

Pseudocode implementing step MMA3 follows:

```
# Determine the array of possible causes for the failed
#   functional test based on triggered rules and excluded causes
      #     Ncause = NcauseTrig - excludeCause
# Initialize array NcauseFinal, then store largest element(s) of
#   Ncause in this array
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# NcauseFinal = max(Ncause)
```

5.1.4.2.2.2 Inverse Multi-Mode Analysis

The inverse multi-mode analysis starts by identifying common faults for the system of interest, and then identifies symptoms and/or rule outcomes (a rule that is satisfied can be used as an indicator of a particular fault if other symptoms are present) that are anticipated to be present based on the tests performed. The fault list consists of the following:

- Leaking damper
- Stuck open damper
- Stuck closed damper
- Leaking reheat coil valve
- Stuck open reheat coil valve
- Stuck closed reheat coil valve
- Reheat valve actuator reversal fault
- Discharge air temperature or AHU supply air temperature sensor fault

The symptoms and rule outcomes for each of these faults are listed and explained below. In each case it is assumed that only a single fault is present at any time. The symptoms of damper faults are based on airflow measurements in different test segments and those for reheat valve and temperature sensor faults are based on temperature measurements. To aid the understanding of this analysis, the symptoms and rule outcomes are first described using these measurements and generic descriptions of damper, valve, and fan operation associated with the measurements. That is, symptoms and rule outcomes of damper faults are defined for operation with damper closed (0%), partially open (25%), and fully open (100%) and the supply fan off (Off), at low speed (Lo), and high speed (Hi). Similarly, reheat valve and temperature sensor faults are defined for operation with the reheat valve closed (0%), partially open (25%), and fully open (100%) and with the supply fan at low speed (Lo) and high speed (Hi). These generic conditions are then mapped onto the test segments in Figure 39. Note the low speed and high speed are relative terms and are assumed to be achieved by operating the supply fan to maintain the static pressure at its minimum and maximum setpoints, respectively.

Leaking Damper

The symptoms and rule outcomes of a leaking damper fault are:

$F_{0,Off} \approx 0$	The airflow rate with the fan off is negligible.
$F_{0,Lo} >> 0$	There is excessive airflow with the damper commanded closed and the fan on.
$F_{0,Hi} >> F_{0,Lo}$	The airflow rate is significantly higher with the damper commanded closed and the fan operating at high speed compared to low speed.

where the first subscript on F is the damper position and the second subscript is the fan operation. The symptoms and rule outcomes are mapped to the test profile in Table 23 below:

Table 23: Symptoms and rule outcomes mapped to test profile for leaking damper.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Symptom or Rule Outcome	Description
RD1 = “True”	Airflow rate is negligible with the supply fan off
RD2 = “False”	Airflow rate exceeds the threshold in RD2 with the supply fan maintaining the static pressure at its minimum setpoint and the damper closed
$F_{vav,D3} - F_{vav,D2} > \varepsilon_{LD3} F_{vav,max}$	Airflow rate with the supply fan maintaining the maximum static pressure setpoint and the damper closed (segment D3) is significantly larger than when the supply fan is maintaining the minimum static pressure setpoint and the damper is closed (segment D2)

where RDi is the criterion defined in Section 5.1.4.2.1.1 for operation in segment Di, $F_{vav,Di}$ is the average airflow rate in segment Di, $F_{vav,max}$ is the design maximum airflow rate, and ε_{LD3} is a decimal coefficient used to determine the threshold for the third symptom. A value of 0.05 is suggested for ε_{LD3}.

Pseudocode for implementing the leaking damper analysis follows.

```
# Set user-defined thresholds for leaking damper symptoms and set
#   design parameters
      # ε_LD3 = 0.05   # threshold for symptom 3
      # F_vav,max = design maximum airflow rate from design documents
# If RD1 == “True” & RD2 == “False” & …
#     F_vav,D3 - F_vav,D2 > ε_LD3 F_vav,max:
      # LkgDmp = ”True”
# Else:
      # LkgDmp = ”False”
```

Stuck Open Damper

The symptoms and rule outcomes of a stuck open damper fault are:

$F_{0,Off} \approx 0$	The airflow rate with the fan off is negligible.
$F_{0,Lo} >> 0$	There is excessive airflow with the damper commanded closed and the fan on.
$F_{0,Hi} \approx F_{100,Hi}$	The airflow rate does not change appreciably with the damper commanded closed and then fully open with the fan operating at high speed.
$F_{0,Hi} \approx F_{25,Hi}$	The airflow rate does not change appreciably with the damper commanded closed and then partially open with the fan operating at high speed.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$F_{25,Hi} >> F_{25,Lo}$	The airflow rate is significantly higher with the damper commanded partially open and the fan operating at high speed compared to low speed.

The symptoms and rule outcomes are mapped to the test profile in Table 24 below:

Table 24: Symptoms and rule outcomes mapped to test profile for stuck open damper.

Symptom or Rule Outcome	Description
RD1 = "True"	Airflow rate is negligible with the supply fan off
RD2 = "False"	Airflow rate exceeds the threshold in RD2 with the supply fan maintaining the static pressure at its minimum setpoint and the damper commanded closed
$\lvert F_{vav,D3} - F_{vav,D4} \rvert < \varepsilon_{SOD1} F_{vav,max}$	The difference in the airflow rate with the fan maintaining the maximum static pressure setpoint is negligible when the damper is commanded fully closed (segment D3) and fully open (segment D4)
$\lvert F_{vav,D3} - F_{vav,D5} \rvert < \varepsilon_{SOD2} F_{vav,max}$	The difference in the airflow rate with the fan maintaining the maximum static pressure setpoint is negligible when the damper is commanded fully closed (segment D3) and partially open (segment D5)
$F_{vav,D5} - F_{vav,V2} > \varepsilon_{SOD3} F_{vav,max}$	Airflow rate with the damper commanded partially open should be higher when the supply fan maintains the static pressure at its maximum setpoint (segment D5) compared to when it maintains its minimum setpoint (segment V2)

where RDi is the criterion defined in Section 5.1.4.2.1.1 for operation in segment Di, $F_{vav,Di}$ and $F_{vav,Vi}$ are the average airflow rates in segments Di, and Vi, $F_{vav,max}$ is the design maximum airflow rate, and ε_{SODi} are decimal coefficients used to determine the thresholds for the three symptoms. A value of 0.1 is suggested for ε_{SOD1} and ε_{SOD2}, and a value of 0.05 is suggested for ε_{SOD3}.

Pseudocode for implementing the stuck open damper analysis follows.

```
# Set user-defined thresholds for stuck open damper symptoms and set
#   design parameters
      # ε_SOD1 = 0.1    # threshold for symptom 1
      # ε_SOD2 = 0.1    # threshold for symptom 2
      # ε_SOD3 = 0.05    # threshold for symptom 3
      # F_vav,max = design maximum airflow rate from design documents
# If RD1 == "True" & RD2 == "False" & …
#     |F_vav,D3 - F_vav,D4| < ε_SOD1 F_vav,max & …
#     |F_vav,D3 - F_vav,D5| < ε_SOD2 F_vav,max & …
#     F_vav,D5 - F_vav,V2 > ε_SOD3 F_vav,max:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      # StkOpenDmp = "True"
# Else:
      # StkOpenDmp = "False"
```

Stuck Closed Damper

The symptoms and rule outcomes of a stuck closed damper fault are:

$F_{0,Off} \approx 0$	The airflow rate with the fan off is negligible.
$F_{0,Lo} \approx 0$	The airflow rate with the damper commanded closed and the fan on is negligible.
$F_{0,Hi} \approx F_{100,Hi}$	The airflow rate does not change appreciably with the damper commanded closed and then fully open with the fan operating at high speed.
$F_{0,Hi} \approx F_{25,Hi}$	The airflow rate does not change appreciably with the damper commanded closed and then partially open with the fan operating at high speed.
$F_{25,Hi} \approx F_{25,Lo}$	The airflow rate does not change appreciably with the damper commanded partially open and the fan operating at high speed compared to low speed.

The symptoms and rule outcomes are mapped to the test profile in Table 25 below:

Table 25: Symptoms and rule outcomes mapped to test profile for stuck closed damper.

Symptom or Rule Outcome	Description
RD1 = "True"	Airflow rate is negligible with the supply fan off
RD2 = "True"	Airflow rate does not exceed the threshold in RD2 with the supply fan maintaining the static pressure at its minimum setpoint and the damper commanded closed
$\lvert F_{vav,D3} - F_{vav,D4} \rvert < \varepsilon_{SCD1} F_{vav,max}$	The difference in the airflow rate with the fan maintaining the maximum static pressure setpoint is negligible when the damper is commanded fully closed (segment D3) and fully open (segment D4)
$\lvert F_{vav,D3} - F_{vav,D5} \rvert < \varepsilon_{SCD2} F_{vav,max}$	The difference in the airflow rate with the fan maintaining the maximum static pressure setpoint is negligible when the damper is commanded fully closed (segment D3) and partially open (segment D5)
$\lvert F_{vav,D5} - F_{vav,V2} \rvert < \varepsilon_{SCD3} F_{vav,max}$	The difference in the airflow rate with the damper commanded partially open is negligible when the supply fan maintains the static pressure at its maximum setpoint (segment D5) compared to when it maintains its minimum setpoint (segment V2)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

where RDi is the criterion defined in Section 5.1.4.2.1.1 for operation in segment Di, $F_{vav,Di}$ and $F_{vav,Vi}$ are the average airflow rates in segments Di, and Vi, $F_{vav,max}$ is the design maximum airflow rate, and ε_{SCDi} are decimal coefficients used to determine the thresholds for the three symptoms. A value of 0.1 is suggested for all the decimal coefficients.

Pseudocode for implementing the stuck closed damper analysis follows.

```
# Set user-defined thresholds for stuck closed damper symptoms and
set
#   design parameters
      # ε_SCD1 = 0.1    # threshold for symptom 1
      # ε_SCD2 = 0.1    # threshold for symptom 2
      # ε_SCD3 = 0.1    # threshold for symptom 3
      # F_vav,max = design maximum airflow rate from design documents
# If RD1 == "True" & RD2 == "True" & …
#     |F_vav,D3 - F_vav,D4| < ε_SCD1 F_vav,max & …
#     |F_vav,D3 - F_vav,D5| < ε_SCD2 F_vav,max & …
#     |F_vav,D5 - F_vav,V2| < ε_SCD3 F_vav,max:
      # StkClosedDmp = "True"
# Else:
      # StkClosedDmp = "False"
```

Leaking Reheat Valve

The symptoms and rule outcomes of a leaking reheat valve fault are:

$\Delta T_{0,Lo} >> 0$	The temperature rise across the reheat coil with the valve commanded closed is excessive.
$\Delta T_{0,Lo} > \Delta T_{0,Hi}$	The temperature rise across the reheat coil with the valve commanded closed is larger at low speeds than at high speeds.
$\Delta T_{100,Lo} >> \Delta T_{0,Lo}$	The temperature rise across the reheat coil with the valve commanded fully open is significantly greater than with the valve commanded closed.

where the first subscript on ΔT is the reheat valve commanded position and the second subscript is the fan operation. The damper is commanded to a constant partially open position for this analysis. The symptoms and rule outcomes are mapped to the test profile in Table 26 below:

Table 26: Symptoms and rule outcomes mapped to test profile for leaking reheat valve.

Symptom or Rule Outcome	Description
RV2 = "False"	The temperature rise across the reheat coil with the valve commanded closed exceeds the threshold in rule RV2

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$\Delta T_{V2} - \Delta T_{V1} > \varepsilon_{LV2}$	The temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) should be larger than with the fan is maintaining the maximum static pressure setpoint (segment V1)
$\Delta T_{V3} - \Delta T_{V2} > \varepsilon_{LV3}$	The temperature rise across the reheat coil with the valve commanded fully open and the fan maintaining the minimum static pressure setpoint (segment V3) should be significantly greater than when the valve is commanded closed (segment V2)

where RV2 is the criterion defined in Section 5.1.4.2.1.2 for operation in test segment V2, ΔT_{Vi} is the average temperature rise across the reheat coil in segment Vi, and ε_{LVi} are thresholds for the symptoms. A value of 2°F is suggested for ε_{LV2} and a value of 5°F is suggested for ε_{LV3}.

Pseudocode for implementing the leaking valve analysis follows.

```
# Set user-defined thresholds for leaking valve symptoms
      # ε_LV2 = 2°F  # threshold for symptom 2
      # ε_LV3 = 5°F  # threshold for symptom 3
# If RV2 == "False" & …
#     ΔT_V2 - ΔT_V1 > ε_LV2 & …
#     ΔT_V3 - ΔT_V2 > ε_LV3:
      # LkgVlv = "True"
# Else:
      # LkgVlv = "False"
```

Stuck Open Reheat Valve

The symptoms and rule outcomes of a stuck open reheat valve fault are:

$\Delta T_{0,Lo} >> 0$	The temperature rise across the reheat coil with the valve commanded closed is excessive.
$\Delta T_{0,Lo} > \Delta T_{0,Hi}$	The temperature rise across the reheat coil with the valve commanded closed is larger at low speeds than at high speeds.
$\Delta T_{100,Lo} \approx \Delta T_{0,Lo}$	The temperature rise across the reheat coil with the valve commanded fully open is nearly the same as with the valve commanded closed.

The damper is commanded to a constant partially open position for this analysis. The symptoms and rule outcomes are mapped to the test profile in Table 27 below:

Table 27: Symptoms and rule outcomes mapped to test profile for stuck open reheat valve.

Symptom or Rule Outcome	Description

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

RV2 = “False”	The temperature rise across the reheat coil with the valve commanded closed exceeds the threshold in rule RV2
$\Delta T_{V2} - \Delta T_{V1} > \varepsilon_{SOV2}$	The temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) should be larger than with the fan is maintaining the maximum static pressure setpoint (segment V1)
$\lvert \Delta T_{V3} - \Delta T_{V2} \rvert < \varepsilon_{SOV3}$	The difference in the temperature rise across the reheat coil with the valve commanded fully open (segment V3) and fully closed (segment V2) is negligible with the fan maintaining the minimum static pressure setpoint in each segment

where RV2 is the criterion defined in Section 5.1.4.2.1.2 for operation in test segment V2, ΔT_{Vi} is the average temperature rise across the reheat coil in segment Vi, and ε_{SOVi} are thresholds for the symptoms. A value of 2°F is suggested for ε_{SOV2} and ε_{SOV3}.

Pseudocode for implementing the stuck open reheat valve analysis follows.

```
# Set user-defined thresholds for stuck open reheat valve symptoms
      # ε_SOV2 = 2°F  # threshold for symptom 2
      # ε_SOV3 = 2°F  # threshold for symptom 3
# If RV2 == “False” & …
#     ΔT_V2 - ΔT_V1 > ε_SOV2 & …
#     |ΔT_V3 - ΔT_V2| < ε_SOV3:
      # StkOpenVlv = ”True”
# Else:
      # StkOpenVlv = ”False”
```

Stuck Closed Reheat Valve

The symptoms and rule outcomes of a stuck closed reheat valve fault are:

$\Delta T_{0,Lo} \approx 0$	The temperature rise across the reheat coil with the valve commanded closed is negligible.
$\Delta T_{0,Lo} \approx \Delta T_{0,Hi}$	The temperature rise across the reheat coil with the valve commanded closed is nearly the same at low and high speeds.
$\Delta T_{100,Lo} \approx \Delta T_{0,Lo}$	The temperature rise across the reheat coil with the valve commanded fully open is nearly the same as with the valve commanded closed.

The damper is commanded to a constant partially open position for this analysis. The symptoms and rule outcomes are mapped to the test profile in Table 28 below:

Table 28: Symptoms and rule outcomes mapped to test profile for stuck closed reheat valve.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Symptom or Rule Outcome	Description
RV2 = “True”	The temperature rise across the reheat coil with the valve commanded closed does not exceed the threshold in rule RV2
$\lvert \Delta T_{V2} - \Delta T_{V1} \rvert < \varepsilon_{SCV2}$	The difference in the temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) is negligible from the value with the fan maintaining the maximum static pressure setpoint (segment V1)
$\lvert \Delta T_{V3} - \Delta T_{V2} \rvert < \varepsilon_{SCV3}$	The difference in the temperature rise across the reheat coil with the valve commanded fully open (segment V3) and fully closed (segment V2) is negligible with the fan maintaining the minimum static pressure setpoint in each segment*
$T_{da,V3} < 80°F$	The discharge air temperature indicates that the space served by the terminal unit is not being heated.

where RV2 is the criterion defined in Section 5.1.4.2.1.2 for operation in test segment V2, ΔT_{Vi} is the average temperature rise across the reheat coil in segment Vi, $T_{da,V3}$ is the average discharge air temperature in segment V3, and ε_{SCVi} are thresholds for the symptoms. A value of 2°F is suggested for ε_{SCV2} and ε_{SCV3}.

Pseudocode for implementing the stuck closed reheat valve analysis follows.

```
# Set user-defined thresholds for stuck closed reheat valve symptoms
      # ε_SCV2 = 2°F  # threshold for symptom 2
      # ε_SCV3 = 2°F  # threshold for symptom 3
# If RV2 == “True” & …
#     |ΔT_V2 - ΔT_V1| < ε_SCV2 & …
#     |ΔT_V3 - ΔT_V2| < ε_SCV3 & …
#     T_da,V3 < 80°F:
      # StkClosedVlv = ”True”
# Else:
      # StkClosedVlv = ”False”
```

Reheat Valve Actuator Reversal

The symptoms and rule outcomes of a reheat valve actuator reversal fault are:

$\Delta T_{0,Lo} >> 0$	The temperature rise across the reheat coil with the valve commanded closed is excessive.
$\Delta T_{0,Lo} > \Delta T_{0,Hi}$	The temperature rise across the reheat coil with the valve commanded closed is larger at low speeds than at high speeds.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$\Delta T_{25,Lo} << \Delta T_{0,Lo}$ The temperature rise across the reheat coil with the valve commanded partially open is significantly less than with the valve commanded closed.

$\Delta T_{100,Lo} << \Delta T_{25,Lo}$ The temperature rise across the reheat coil with the valve commanded fully open is significantly less than with the valve commanded partially open.

The damper is commanded to a constant partially open position for this analysis. The symptoms and rule outcomes are mapped to the test profile in Table 29 below:

Table 29: Symptoms and rule outcomes mapped to test profile for reheat valve actuator reversal.

Symptom or Rule Outcome	Description
RV2 = "False"	The temperature rise across the reheat coil with the valve commanded closed exceeds the threshold in rule RV2
$\Delta T_{V2} - \Delta T_{V1} > \varepsilon_{VAR2}$	The temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) is larger than with the fan is maintaining the maximum static pressure setpoint (segment V1)
$\Delta T_{V2} - \Delta T_{V4} > \varepsilon_{VAR3}$	The temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) is significantly greater than when the valve is partially open (segment V4)
$\Delta T_{V4} - \Delta T_{V3} > \varepsilon_{VAR4}$	The temperature rise across the reheat coil with the valve commanded partially open and the fan maintaining the minimum static pressure setpoint (segment V4) is significantly greater than when the valve is fully open (segment V3)

where RV2 is the criterion defined in Section 5.1.4.2.1.2 for operation in test segment V2, ΔT_{Vi} is the average temperature rise across the reheat coil in segment Vi, and ε_{VARi} are thresholds for the symptoms. A value of 2°F is suggested for ε_{VAR2} and a value of 5°F is suggested for ε_{VAR3} and ε_{VAR4}.

Pseudocode for implementing the reheat valve actuator reversal analysis follows.

```
# Set user-defined thresholds for reheat valve actuator reversal
#   symptoms
      # ε_VAR2 = 2°F  # threshold for symptom 2
      # ε_VAR3 = 5°F  # threshold for symptom 3
      # ε_VAR4 = 5°F  # threshold for symptom 4
# If RV2 == "False" & …
#     ΔT_V2 - ΔT_V1 > ε_VAR2 & …
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#     ΔT_V2 - ΔT_V4 > ε_VAR3 & …
#     ΔT_V4 - ΔT_V3 > ε_VAR4:
      # VlvActRev = ”True”
# Else:
      # VlvActRev = ”False”
```

Discharge Air Temperature or AHU Supply Air Temperature Sensor Fault

The symptoms and rule outcomes of a discharge air temperature or AHU supply air temperature sensor fault are:

$\Delta T_{0,Lo} >> 0$	The temperature rise across the reheat coil with the valve commanded closed is excessive.
$\Delta T_{0,Lo} \approx \Delta T_{0,Hi}$	The temperature rise across the reheat coil with the valve commanded closed is nearly the same at low and high speeds.
$\Delta T_{100,Lo} > \Delta T_{0,Lo}$	The temperature rise across the reheat coil with the valve commanded fully open is significantly larger than with the valve commanded closed.

The damper is commanded to a constant partially open position for this analysis. The symptoms and rule outcomes are mapped to the test profile in Table 30 below:

Table 30: Symptoms and rule outcomes mapped to test profile for discharge air or AHU supply air temperature sensor fault.

Symptom or Rule Outcome	Description
RV2 = “False”	The temperature rise across the reheat coil with the valve commanded closed exceeds the threshold in rule RV2
$\lvert \Delta T_{V2} - \Delta T_{V1} \rvert < \varepsilon_{TS2}$	The difference in the temperature rise across the reheat coil with the valve commanded closed and the fan maintaining the minimum static pressure setpoint (segment V2) is negligible from the value with the fan maintaining the maximum static pressure setpoint (segment V1)
$\Delta T_{V3} - \Delta T_{V2} > \varepsilon_{TS3}$	The temperature rise across the reheat coil with the valve commanded fully open and the fan maintaining the minimum static pressure setpoint (segment V3) should be significantly larger than when the valve is commanded closed (segment V2)

where RV2 is the criterion defined in Section 5.1.4.2.1.2 for operation in test segment V2, ΔT_{Vi} is the average temperature rise across the reheat coil in segment Vi, and ε_{TSi} are thresholds for the temperature sensor symptoms. A value of 2°F is suggested for ε_{TS2} and a value of 5°F is suggested for ε_{TS3}.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Pseudocode for implementing the discharge air temperature or AHU supply air temperature sensor fault analysis follows.

```
# Set user-defined thresholds for temperature sensor fault symptoms
      # ε_TS2 = 2°F  # threshold for symptom 2
      # ε_TS3 = 5°F  # threshold for symptom 3
# If RV2 == "False" & …
#     |ΔT_V2 - ΔT_V1| < ε_TS2 & …
#     ΔT_V3 - ΔT_V2 > ε_TS3:
      # TempSensFault = "True"
# Else:
      # TempSensFault = "False"
```

5.1.5 Appendix

The AHU supply fan operation is important to the interpretation of the open-loop functional tests described in this document. The supply fan operation will affect the volume of air delivered to a terminal unit which is important, in particular, to the evaluation of the ability of a terminal unit to deliver its design airflow rate. The airflow rate will also impact the temperature rise of air flowing across the reheat coil when heating water flows through the coil, and measuring the temperature rise at different airflow rates may enable leaking valve faults to be diagnosed. Thus, criteria are needed to ensure that the desired operation of the supply fan is achieved during testing.

In test segment D1 (see Figure 39), the supply fan is off (i.e., $u_{sf} = 0$). To verify this, the following criterion can be used:

$$SF_{off}: | P_{sa,i} | < \varepsilon_{SFoff} \text{ for } f > f_{min} \text{ of the final 10 test segment samples}$$

where $P_{sa,i}$ is the AHU static pressure at sampling instant i, ε_{SFoff} is a threshold with a suggested value of 0.05 inH_20, f is the fraction of the final 10 test segment samples (assuming 15 one-minute samples per test segment) that satisfy the criterion, and f_{min} is the minimum decimal fraction of samples that must satisfy the criterion (e.g., 0.75) in order for the test segment fan conditions to be met.

In test segments D2 and V2-V4, the AHU supply fan is controlled to maintain the static pressure at its minimum setpoint. To verify the static pressure control, at each sampling time instance i, the supply air pressure error, $e_{Psa,i}$, is calculated as

$$e_{Psa,i} = P_{sa,set,i} - P_{sa,i}$$

The supply fan criterion at the minimum static pressure setpoint is then defined as follows:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

SF_{min}: $|e_{Psa,i}| \leq \varepsilon_{SFmin} (P_{sa,set,max} - P_{sa,set,min})$ for $f > f_{min}$ of the final 10 test segment samples

where ε_{SFmin} is a user-defined decimal coefficient used to determine the threshold on individual sample errors as a fraction of the setpoint range (i.e., ε_{SFmin} = 0.1 indicates the absolute value of a sample error must be less than or equal to 10% of the allowable setpoint range to be considered "in control"), f is the fraction of the final 10 test segment samples that satisfy the criterion (assuming 15 one-minute samples per test segment), and f_{min} is the minimum decimal fraction of samples that must be in control to satisfy the criterion (e.g., 0.75). The thresholds ε_{SFmin} and f_{min} can be adjusted based on field experience, owner requirements, number of data samples, etc.

A similar criterion is defined for test segments D3-D5 and V1 for which the AHU supply fan is controlled to maintain the static pressure at its maximum setpoint. The criterion is defined as follows:

SF_{max}: $|e_{Psa,i}| \leq \varepsilon_{SFmax} (P_{sa,set,max} - P_{sa,set,min})$ for $f > f_{min}$ of the final 10 test segment samples

where ε_{SFmax} is a user-defined decimal coefficient used to determine the threshold on individual sample errors as a fraction of the setpoint range when operating at the maximum static pressure setpoint.

Pseudocode for implementing the analysis of the supply fan control follows. Note that SF_{max} and SF_{min} must be calculated based on the data for each test segment in which they are applicable.

```
# Read P_sa and P_sa,set for all data samples in test segments D1-D5 and
#   V1-V4
# Set user-defined parameters and threshold(s)
      # P_sa,set,max   # defined in design drawings
      # P_sa,set,min   # defined in design drawings
      # ε_SFoff  = 0.05  # threshold parameter for SF_off; [0 - 0.1]
      # ε_SFmax  = 0.1   # threshold parameter for SF_max; [0 - 0.2]
      # ε_SFmin  = 0.1   # threshold parameter for SF_max; [0 - 0.2]
      # f_min = 0.75     # minimum fraction of "in control" samples in a
                         #   test segment required for the segment to be
                         #   considered under control
# For each data sample i, compute the supply air static pressure
error
      # e_Psa,i = P_sa,set,i - P_sa,i for all i in a test segment
#
# ****** Supply Fan Off ******
# Initialize counters
      # count = 0
      # i = 0
# Test segment D1
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      # if testSeg == D1:
            # while i <= 15:
                  # if i > 5:  # check last 10 test segment samples
                        # If |P_sa,i| < ε_SFoff:
                              # count += 1
                  # i += 1
            # if (count/10) > f_min:
                  # SF_off = "True"
            # else:
                  # SF_off = "False"
#
# ** Supply Fan Controlled to Maintain Min Static Pressure Setpoint
**
# Initialize counters
      # count = 0
      # i = 0
# Test segment D2, V2, V3 or V4
      # if testSeg == D2 (or V2, V3 or V4):
            # while i <= 15:
                  # if i > 5:  # check last 10 test segment samples
                        # If | e_Psa,i | ≤ ε_SFmin (P_sa,set,max - P_sa,set,min):
                              # count += 1
                  # i += 1
            # if (count/10) > f_min:
                  # SF_min = "True" # Note that SF_min must be evaluated
                                    # separately for each test segment
            # else:
                  # SF_min = "False"
#
# ** Supply Fan Controlled to Maintain Max Static Pressure Setpoint
**
# Initialize counters
      # count = 0
      # i = 0
# Test segment D3, D4, D5 or V1
      # if testSeg == D3 (or D4, D5 or V1):
            # while i <= 15:
                  # if i > 5:  # check last 10 test segment samples
                        # If | e_Psa,i | ≤ ε_SFmax (P_sa,set,max - P_sa,set,min):
                              # count += 1
                  # i += 1
            # if (count/10) > f_min:
                  # SF_max = "True" # Note that SF_max must be evaluated
                                    # separately for each test segment
            # else:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

SF_{max} = "False"

5.2 Open-loop Functional Tests for AHUs

5.2.1 System Description

The open-loop functional performance tests described in this document have been developed to test components commonly found in single duct variable-air-volume (VAV) AHUs. A schematic diagram showing the main components of the AHU as well as common measurement and control points is shown in Figure 40.

Outdoor air enters the AHU and is mixed with air recirculated from the building. The mixed air is conditioned by the heating coil or cooling coil as needed to achieve the supply air temperature setpoint. The conditioned air is then distributed to the building. The supply fan is commonly operated to maintain the static pressure in the supply ductwork at a sufficiently high pressure to ensure the necessary quantity of air reaches each zone served by the AHU. The return fan draws air from the zones and is commonly controlled to maintain the building at a fixed positive pressure to prevent infiltration. Circulating pumps are used to deliver chilled water and heating water to the AHU cooling and heating coils, respectively.

In the commissioning process, open-loop functional tests are commonly used to verify components are working as intended. Open-loop functional tests may include such tests as checking that the position of a damper changes from fully closed to fully open when its control signal is changed from 0 to 100%, or verifying that the temperature downstream from a heating coil increases when the heating valve is opened. In open-loop tests such as these the normal feedback control and built-in protections associated with the overridden control signals are disabled. Thus, care must be exercised to avoid damaging equipment or ductwork.

The open-loop functional tests described in this specification involve overriding control signals to the AHU mixing box dampers, heating coil valve, cooling coil valve, supply fan, return fan, chilled water circulating pump and heating water circulating pump in eight different test segments in order to assess the performance of individual AHU components. The tests are designed to identify temperature sensor faults, stuck and leaking valves, stuck and leaking dampers, and circulating pump faults.

The next section provides an overview of the overall process for testing and analyzing AHU component performance through open-loop functional tests. Data requirements for the test and the variables and parameters used to implement the functional test and analyze its results are

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

then presented. Finally, individual steps in the overall process are described and pseudocode for implementing the test and analysis is presented.

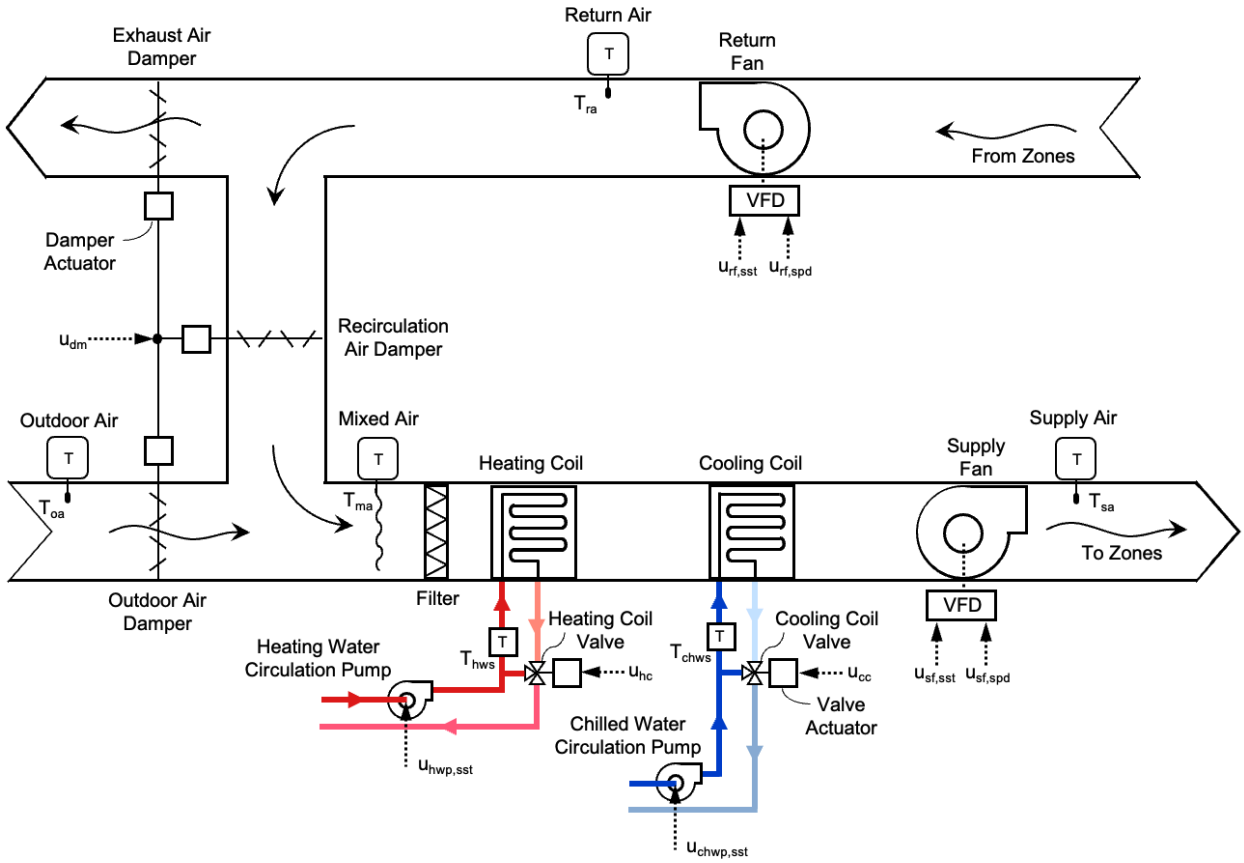


Figure 40: Schematic diagram of a single-zone VAV AHU.

5.2.2 Process Description

Figure 41 shows a process flow diagram for the testing and analysis. The process consists of three modules outlined with dashed lines. The *Pretest Condition Check* module evaluates a number of operational conditions to verify that they are appropriate for conducting the test. The *Proactive Functional Test* module executes the test using overrides of the dampers, valves, fans, and pumps to achieve testing conditions designed to identify common condition-based faults. The *Functional Test Analysis* module uses the test data to evaluate rules describing the expected performance in individual test segments. When a functional test fails because a rule defining expected performance is not satisfied, a multi-test analysis is performed in an effort to

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

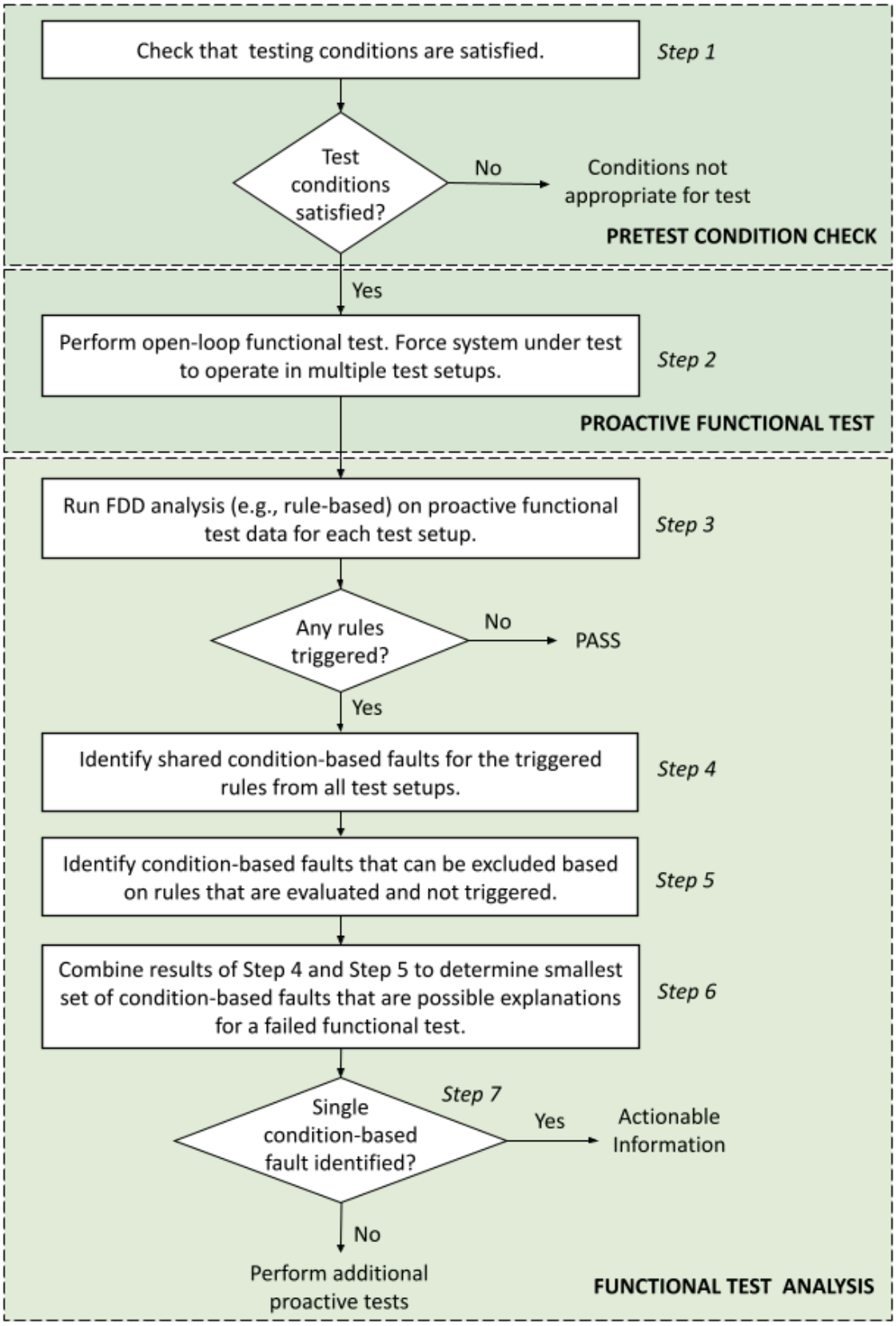


Figure 41: Process flow diagram for the AHU open-loop functional test and analysis.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

determine the root cause fault. The multi-test analysis uses the same process for these open-loop tests as the multi-mode analysis described by Lin et al. (2025)[12] uses for the analysis of closed-loop functional tests. Further explanation of the multi-test analysis is provided in Section 5.2.4.3.

5.2.3 Required Data

Measurement and control points from the building automation system (BAS) that are used in this test are listed in Table 31. All other variables and parameters used to implement the functional test and analyze its results are listed in Table 32.

Table 31: Required Data Points from BAS (controllers).

Abbreviation	Data point	Data type	Requirements (Read/Write)
T_{chws}	chilled water supply temperature [°F]	Time-series sensor reading from BAS	Read
T_{hws}	heating water supply temperature [°F]	Time-series sensor reading from BAS	Read
T_{ma}	mixed air temperature [°F]	Time-series sensor reading from BAS	Read
T_{oa}	outdoor air temperature [°F]	Time-series sensor reading from BAS	Read
T_{ra}	return air temperature [°F]	Time-series sensor reading from BAS	Read
T_{sa}	supply air temperature [°F]	Time-series sensor reading from BAS	Read
u_{cc}	cooling coil valve control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read/Write
$u_{chwp,sst}$	chilled water circulating pump start/stop signal [ON/OFF]	Time-series status signal from BAS	Read/Write
u_{dm}	mixing box damper control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read/Write

[12] Lin, G., House, J., Chen, Y., Granderson, J., and Zhang, W. 2025. Active Multi-mode Data Analysis to Improve Fault Diagnosis in AHUs. Accepted for Energy and Buildings.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

u_{hc}	heating coil valve control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read/Write
$u_{hwp,sst}$	heating water circulating pump start/stop signal [ON/OFF]	Time-series status signal from BAS	Read/Write
$u_{rf,spd}$	return fan speed [%]	Time-series control signal from BAS	Read/Write
$u_{rf,sst}$	return fan start/stop signal [ON/OFF]	Time-series control signal from BAS	Read/Write
$u_{sf,spd}$	supply fan speed [%]	Time-series control signal from BAS	Read/Write
$u_{sf,sst}$	supply fan start/stop signal [ON/OFF]	Time-series control signal from BAS	Read/Write
ZN_{occ}	Zone occupancy status [0 = unoccupied; 1 = occupied]	Control setting	Read

Table 32: Variables and parameters used to implement and evaluate the functional test.

Abbreviation	Data point	Data type
cause_i	[1 x 13] binary array defining possible condition-based faults for the ith rule (i = 1 to 16)	Algorithm parameter
error	binary variable indicating the array “NcauseFinal” is empty (True) or not empty (False)	Algorithm variable
excludeCause	[1 x 13] binary array of excluded causes based on rules that are evaluated but not triggered	Algorithm variable
exclude_i	[1 x 13] binary array of condition-based faults that can be excluded based on the ith rule in Table 35	Algorithm parameter
exclude_i_j	[1 x 13] binary array of condition-based faults that can be excluded based on the ith and jth rule in Table 35	Algorithm parameter
functTestResult	binary variable indicating the outcome of the functional test [Pass or Fail]	Algorithm variable
Ncause	array of possible causes for a failed functional test based on triggered rules and excluded causes from the rules in Table 35	Algorithm variable
NcauseFinal	array of the largest element(s) of array “Ncause”	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

	which contains the condition-based fault(s) most likely to be the root cause of the failed test	
NcauseTrig	[1 x 13] array for counting the number of times individual condition-based faults are identified as possible causes for triggered rules	Algorithm variable
notTrigRules	array of rules that are evaluated and not triggered	Algorithm variable
numRules	integer number of rules (maximum value of “rule”)	Algorithm parameter
rule	iterator variable used in loop to identify triggered rules in the list “trigRules” and non-triggered rules in the list “notTrigRules”	Algorithm variable
Rule_i	Binary variable indicating if a rule is satisfied (True) or triggered/violated (False)	Algorithm variable
runAnalysis	binary variable indicating whether or not analysis in Step 3 should be executed (False if runTest = False)	Algorithm variable
runTest	Binary variable indicating whether or not the conditions are suitable to execute the functional test in Step 2	Algorithm variable
singleCause	binary variable indicating whether or not a single cause for the failed test has been identified	Algorithm variable
$T_{chws,max}$	maximum allowable chilled water supply temperature for testing [°F]	User-defined parameter
$T_{hws,min}$	minimum allowable heating water supply temperature for testing [°F]	User-defined parameter
$T_{oa,min}$	minimum allowable outdoor air temperature for testing [°F]	User-defined parameter
$\Delta T_{ra,oa,min}$	minimum allowable temperature difference between the return and outdoor air for the functional test to be performed [°F]	User-defined parameter
ΔT_{sf}	temperature rise across the supply fan [°F]	User-defined parameter
testSeg	integer variable for labeling test segments [1 to8]	Algorithm variable
testSegCount	integer variable for counting samples in a test segment	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

testSegEnd	real array with timestamp corresponding to the end of each test segment [1 x 8 array]	Algorithm variable
testSegLabel	integer array with label (“testSeg”) for each test segment [1 x 8 array]	Algorithm variable
testSegStart	real array with timestamp corresponding to the start of each test segment [1 x 8 array]	Algorithm variable
trigRules	array of triggered rules	Algorithm variable
$\varepsilon_{OL,i}$	threshold for rule i (i = 1-16) of the open-loop functional test	User-defined parameter

5.2.4 Description of the Open-loop Functional Test and Analysis

This section provides a detailed description of the individual steps in the open-loop functional test and analysis, beginning with the pretest condition check.

5.2.4.1 Pretest Condition Check - Step 1

Prior to executing the test, the following conditions are to be verified:

- Outdoor air temperature: $40°F \leq T_{oa} \leq T_{ra} - \Delta T_{ra,oa,min}$
- Chilled water supply temperature: $T_{chws} \leq 45°F$
- Heating water supply temperature: $T_{hws} \geq 120°F$
- The occupancy status of the zones (ZN_{occ}) served by the AHU under test are “True” or “occupied”

Pseudocode for verifying the test conditions are satisfied follows:

```
# Assign values to T_oa,min=40°F, ΔT_ra,oa,min=10°F, T_chws,max=45°F, T_hws,min=120°F
#
# Read T_oa, T_ra, T_chws, T_hws, and ZN_occ and check if test conditions are
#   satisfied
      # if (T_oa >= T_oa,min & T_oa <= T_ra – ΔT_ra,oa,min
      #   & T_chws <= T_chws,max
      #   & T_hws >= T_hws,min
      #   & ZN_occ == “True”):
            # runTest = “True”
      # else:
            # runTest = “False”
```

5.2.4.2 Proactive Functional Test - Step 2

If *runTest* is “True”, the *Proactive Functional Test* outlined in the process flow diagram in Figure 41 is executed. The test involves overriding control signals to the AHU mixing box dampers, heating coil valve, cooling coil valve, supply fan, return fan, chilled water circulating pump and

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

heating water circulating pump in eight different test segments in order to assess the performance of individual AHU components. Each test segment is 15 minutes long. The test segments and names are listed below:

- Test Segment 1 - Temperature Sensor Comparison Test with 100% Outdoor Air
- Test Segment 2 - Recirculation Air Damper Leakage Test
- Test Segment 3 - Temperature Sensor Comparison Test with 100% Recirculation Air
- Test Segment 4 - Outdoor Air Damper Leakage Test
- Test Segment 5 - Heating Valve Leakage Test
- Test Segment 6 - Heating Valve Actuator Test
- Test Segment 7 - Cooling Valve Leakage Test
- Test Segment 8 - Cooling Valve Actuator Test

Note that control sequences and logic are not assessed in these open-loop functional tests.

As the tests are performed, data are collected and stored at a fixed sampling rate. Faster sampling rates such as 1-2 minutes are preferred. The start and end of each test segment should be recorded, and the test segments should be labeled 1-8 to designate the intended test. The data to be read and stored are T_{sa}, T_{ra}, T_{oa}, T_{ma}, u_{cc}, u_{dm}, u_{hc}, $u_{sf,spd}$, $u_{sf,sst}$, $u_{rf,spd}$, $u_{rf,sst}$, $u_{hwp,sst}$, and $u_{chwp,sst}$. Descriptions of all data points are provided in Table 31.

Individual test segments are described in the following section. There is considerable redundancy among the rules in the eight test segments. This was done to enable individual test segments to be performed in isolation without reliance on the other test segments. After describing all of the test segments in Section 5.2.4.2.1, pseudocode for implementing them is provided in Section 5.2.4.2.2.

5.2.4.2.1 Description of Tests

The complete open-loop functional test for AHUs consists of eight test segments achieved by overriding control signals to dampers, valves, fans, and pumps. A detailed description of each segment of the overall test follows and includes the test setup as well as rules that are applicable to that setup and a list of possible condition-based faults that could cause those rules to be triggered (i.e., violated). Table 33 provides a listing of all rules and the condition-based faults that could cause the rules to be triggered (i.e., violated). Table 34 provides the values of all thresholds used in the rules.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Table 33: Open-loop functional test rule set and possible explanations for triggered rules.

	Explanations for Rule Violations												
Rules	1 - Supply Air Temperature Sensor Fault	2 - Return Air Temperature Sensor Fault	3 - Mixed Air Temperature Sensor Fault	4 - Outdoor Air Temperature Sensor Fault	5 - Leaking Cooling Coil Valve	6 - Stuck Cooling Coil Valve	7 - Chilled Water Circulation Pump Fault	8 - Leaking Heating Coil Valve	9 - Stuck Heating Coil Valve	10 - Heating Water Circulation Pump Fault	11 - Leaking Recirculation Air Damper	12 - Leaking Outdoor Air Damper	13 - Stuck Mixing Box Damper(s)
Rule 1: $\lvert T_{sa} - T_{ma} - \Delta T_{sf} \rvert \leq \varepsilon_{OL,1}$	X		X		X	X		X	X				
Rule 2: $\lvert T_{oa} - T_{ma} \rvert \leq \varepsilon_{OL,2}$			X	X							X		X
Rule 3: $\lvert T_{sa} - T_{ma} - \Delta T_{sf} \rvert \leq \varepsilon_{OL,3}$	X		X		X	X		X	X				
Rule 4: $\lvert T_{oa} - T_{ma} \rvert \leq \varepsilon_{OL,4}$			X	X							X		X
Rule 5: $\lvert T_{sa} - T_{ma} \rvert \leq \varepsilon_{OL,5}$	X		X		X	X		X	X				
Rule 6: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,6}$		X	X									X	X
Rule 7: $\lvert T_{sa} - T_{ma} - \Delta T_{sf} \rvert \leq \varepsilon_{OL,7}$	X		X		X	X		X	X				
Rule 8: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,8}$		X	X									X	X
Rule 9: $\lvert T_{sa} - T_{ma} \rvert \leq \varepsilon_{OL,9}$	X		X		X	X		X	X				
Rule 10: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,10}$		X	X									X	X
Rule 11: $T_{sa} - T_{ma} \geq \varepsilon_{OL,11}$	X		X		X	X			X	X			
Rule 12: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,12}$		X	X									X	X
Rule 13: $\lvert T_{sa} - T_{ma} \rvert \leq \varepsilon_{OL,13}$	X		X		X	X			X	X			
Rule 14: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,14}$		X	X									X	X
Rule 15: $T_{ma} - T_{sa} \geq \varepsilon_{OL,15}$	X		X			X	X		X	X			
Rule 16: $\lvert T_{ra} - T_{ma} \rvert \leq \varepsilon_{OL,16}$		X	X									X	X

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Table 34: Threshold values for open-loop functional test rule set.

Threshold	Value
$\varepsilon_{OL,1}, \varepsilon_{OL,3}, \varepsilon_{OL,7}$	3°F
$\varepsilon_{OL,2}, \varepsilon_{OL,4}, \varepsilon_{OL,5}, \varepsilon_{OL,6}, \varepsilon_{OL,8}, \varepsilon_{OL,9}, \varepsilon_{OL,10}, \varepsilon_{OL,6}, \varepsilon_{OL,12}, \varepsilon_{OL,13}, \varepsilon_{OL,14}, \varepsilon_{OL,16}$	2°F
$\varepsilon_{OL,11}, \varepsilon_{OL,15}$	10°F

Test Segment 1 - Temperature Sensor Comparison Test with 100% Outdoor Air
This test compares the supply air, mixed air, and outdoor air temperatures under conditions designed to mitigate the possibility of leakage through the heating coil and cooling coil valves and the recirculation damper. The test setup is illustrated in Figure 42 and the open-loop control signals used to implement the test are described below.

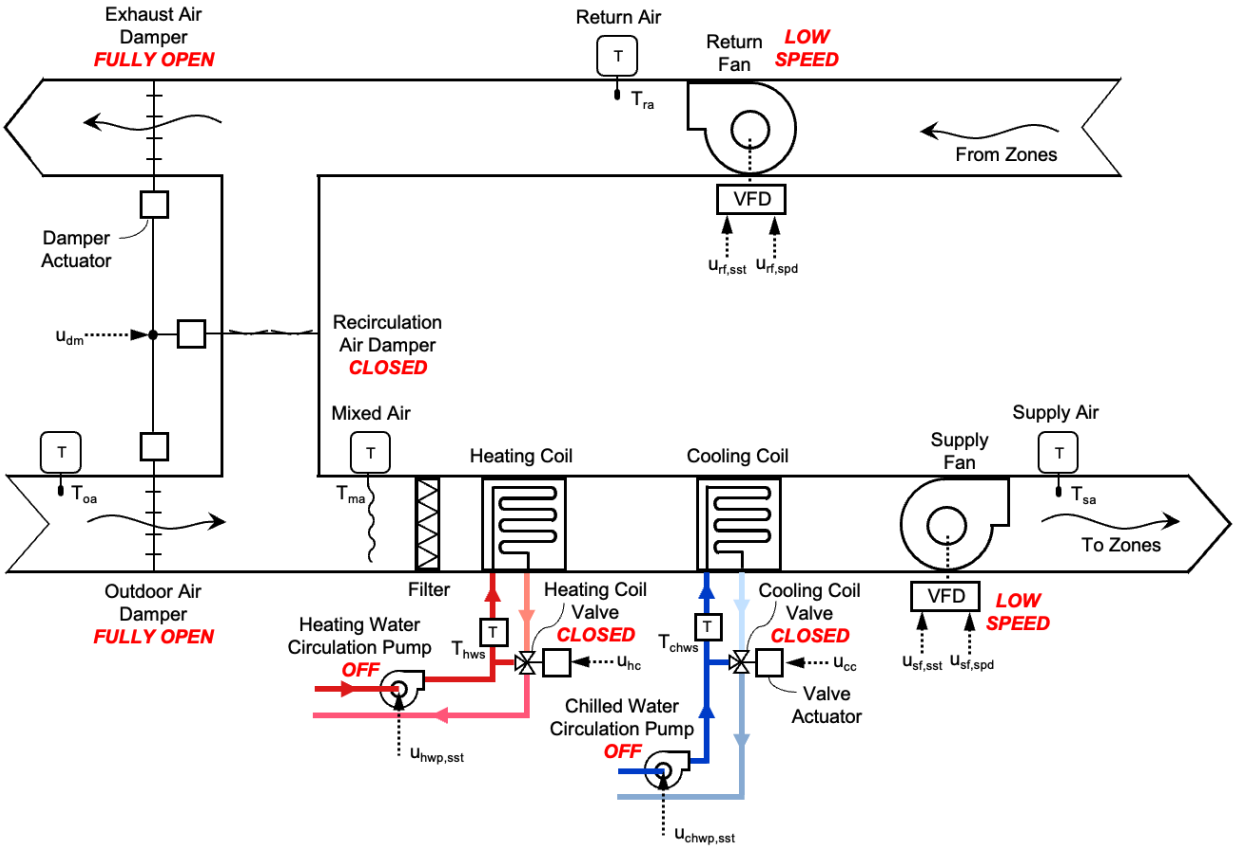


Figure 42: Setup for test segment 1.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 100% outdoor air position (recirculation air damper is fully closed): u_{dm} = 1 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded ON: $u_{sf,sst}$ = On / True

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- AHU supply fan is commanded to operate at low speed: $u_{sf,spd}$ = 30% (adjustable)
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

Note that for each fan there are two points that must be controlled. One point with subscript “sst” is used to start and stop the fan. The second point, with subscript “spd” is used to command the speed of the fan. The circulating pumps are assumed to be constant speed and therefore have only a control signal with subscript “sst” to signify that the point starts and stops the pump.

For the test conditions in Segment 1 the following rules apply:

Rule 1: $| T_{sa} - T_{ma} - \Delta T_{sf} | \leq \varepsilon_{OL,1}$
Rule 2: $| T_{oa} - T_{ma} | \leq \varepsilon_{OL,2}$

Each rule has a threshold or tolerance denoted by $\varepsilon_{OL,i}$ where the subscript “OL” refers to open-loop and the index “i” is the rule number. Recommended values of all thresholds are provided in Table 33. If Rule 1 is triggered (i.e., if the inequality is not satisfied) there is an inconsistency between the supply air and mixed air temperatures because with the heating and cooling valves commanded to the closed position and the circulating pumps off, these values should differ only by the temperature rise across the supply fan (ΔT_{sf}). If Rule 2 is triggered, there is an inconsistency between the outdoor air and mixed air temperatures because these values should be approximately equal with dampers commanded to the 100% outdoor air position.

There are a number of condition-based faults that could cause Rule 1 to be triggered. These faults will be used in the analysis described later to determine the smallest set of condition-based faults that are possible explanations for a failed functional test. Condition-based faults that are possible explanations for Rule 1 being triggered are:

- Mixed air temperature sensor error
- Outdoor air temperature sensor error
- Leaking cooling coil valve
- Stuck cooling coil valve
- Leaking heating coil valve
- Stuck heating coil valve

The condition-based faults that are possible explanations for Rule 2 being triggered consist of the following:

- Mixed air temperature sensor error
- outdoor air temperature sensor error
- Stuck mixing box damper(s)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Leaking recirculation air damper

Test Segment 2 - Recirculation Air Damper Leakage Test
This test compares the outdoor air, mixed air, and supply air temperatures under conditions designed to help reveal leakage through the recirculation air damper while mitigating the possibility of leakage through the heating coil and cooling coil valves. The test setup is illustrated in Figure 43 and the open-loop control signals used to implement the test are described below.

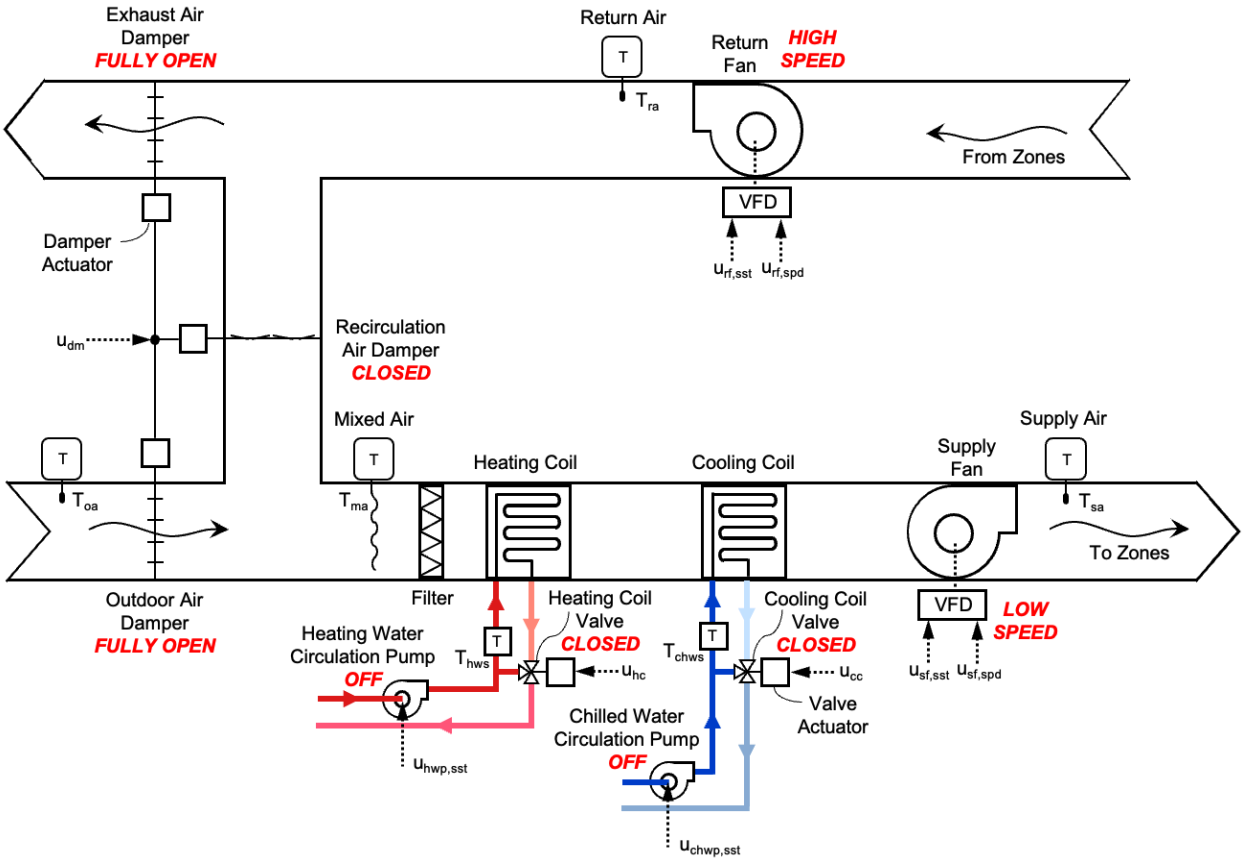


Figure 43: Setup for test segment 2.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 100% outdoor air position (recirculation air damper is fully closed): u_{dm} = 1 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded ON: $u_{sf,sst}$ = On / True
- AHU supply fan is commanded to operate at low speed: $u_{sf,spd}$ = 30% (adjustable)
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at high speed: $u_{rf,spd}$ = 80% (adjustable)
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded OFF: $u_{chwp,sst}$ = Off / False
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

The higher return fan speed used in this test segment (compared to test segment 1) will increase the pressure on the upstream side of the recirculation damper, thereby making leakage through the damper more evident. A change in T_{ma} is expected if there is significant leakage (i.e., because $T_{ra} > T_{oa}$, T_{ma} is expected to increase).

For the test conditions in Segment 2 the following rules apply:

Rule 3: $| T_{sa} - T_{ma} - \Delta T_{sf} | \leq \varepsilon_{OL,3}$
Rule 4: $| T_{oa} - T_{ma} | \leq \varepsilon_{OL,4}$

No additional explanation of the rules is provided since they are identical to Rules 1 and 2.

Test Segment 3 - Temperature Sensor Comparison Test with 100% Recirculation Air
This test compares the supply air, mixed air, and return air temperatures under conditions designed to mitigate the possibility of leakage through the outdoor air damper and the heating coil and cooling coil valves. The test setup is illustrated in Figure 44 and the open-loop control signals used to implement the test are described below.

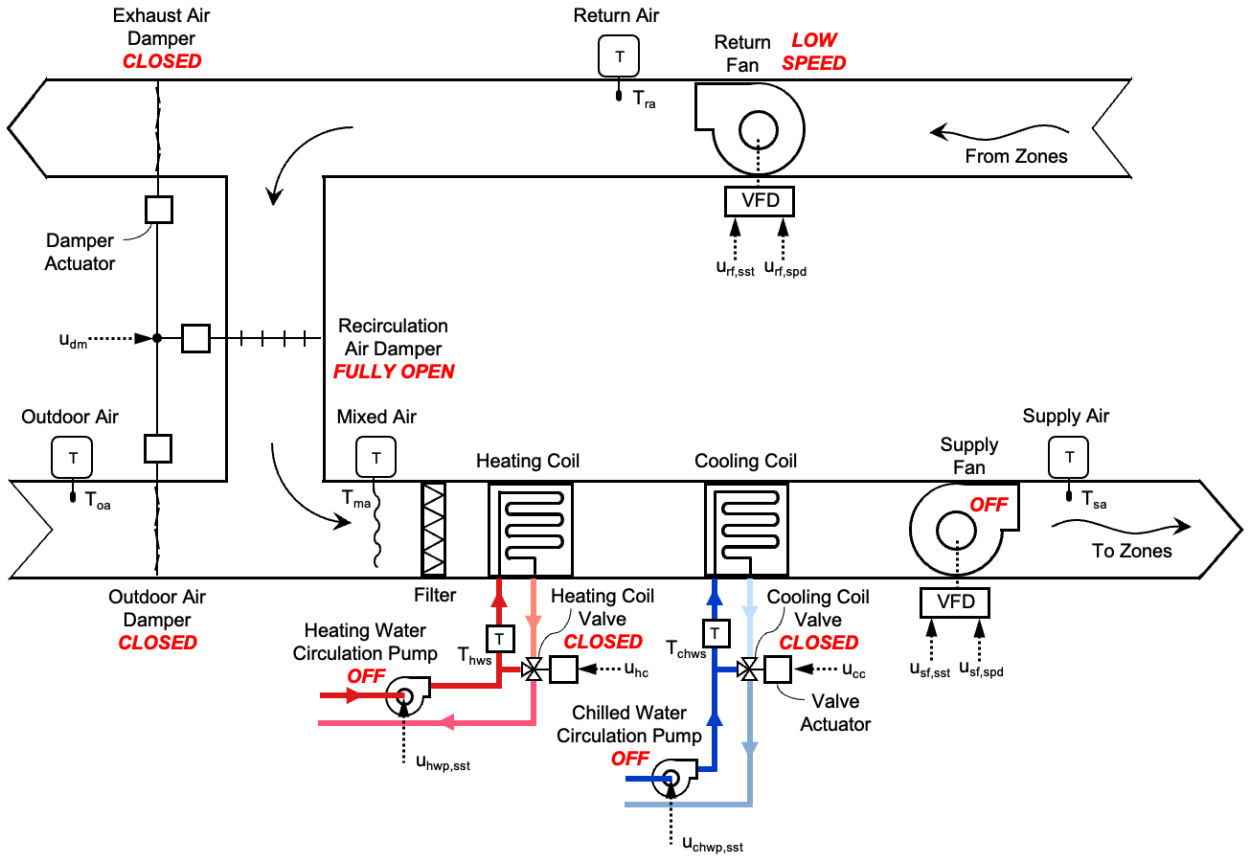


Figure 44: Setup for test segment 3.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is 100% open): u_{dm} = 0 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded OFF: $u_{sf,sst}$ = Off / False
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded OFF: $u_{chwp,sst}$ = Off / False
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

Note that the pressure in the mixing plenum is expected to be positive during this test because the supply fan is off. Thus, any leakage through the outdoor air damper should be from the AHU to the exterior of the building.

For the test conditions in Segment 3 the following rules apply:

Rule 5: $| T_{sa} - T_{ma} | \leq \varepsilon_{OL,5}$
Rule 6: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,6}$

Rule 5 is similar to Rule 1, except that with the supply fan off for Segment 3, it is not necessary to account for the fan temperature rise. If Rule 6 is triggered there is an inconsistency between the return air and mixed air temperature sensors. Condition-based faults that are possible explanations for this inconsistency include the following:

- Mixed air temperature sensor error
- Return air temperature sensor error
- Stuck mixing box damper(s)
- Leaking outdoor air damper

Depending on the location of the return air temperature sensor, Rule 6 may need to be modified to account for temperature rise across the return fan. In addition, this test will need to use the supply fan for air circulation if there is no return fan. In this case, the criteria for comparing the supply and mixed air temperatures will need to account for the temperature rise across the supply fan.

Test Segment 4 - Outdoor Air Damper Leakage Test

This test compares the supply air, mixed air, and return air temperatures under conditions designed to help reveal leakage through the outdoor air damper while mitigating the possibility of leakage through the heating coil and cooling coil valves. The test setup is illustrated in Figure 45 and the open-loop control signals used to implement the test are described below.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

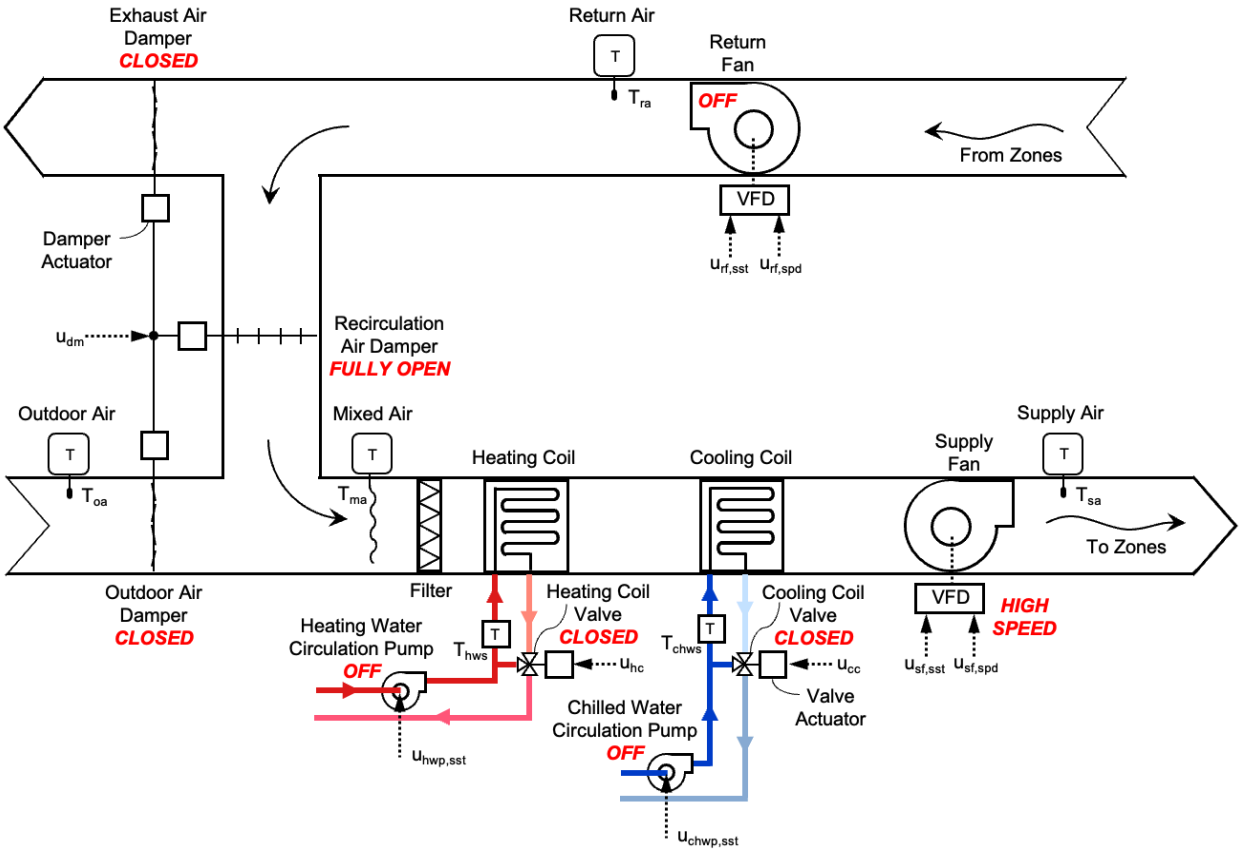


Figure 45: Setup for test segment 4.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is fully open): u_{dm} = 0 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded ON: $u_{sf,sst}$ = On / True
- AHU supply fan is commanded to operate at high speed: $u_{sf,spd}$ = 80% (adjustable)
- AHU return fan is commanded OFF: $u_{rf,sst}$ = Off / False
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded OFF: $u_{chwp,sst}$ = Off / False
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

Note that with the supply fan running at high speed, the pressure in the mixing plenum should be negative relative to outdoor conditions. Thus, if there is significant leakage through the outdoor air damper, it should be evident from the mixed air temperature (i.e., if $T_{oa} < T_{ra}$, the mixed air temperature should decrease relative to its value in Segment 3 due to the leakage).

For the test conditions in Segment 4 the following rules apply:

Rule 7: $| T_{sa} - T_{ma} - \Delta T_{sf} | \leq \varepsilon_{OL,7}$

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rule 8: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,8}$

Rule 7 is identical to Rules 1 and 3, while Rule 8 is identical to Rule 6. Thus, no additional explanation of the rules is provided.

Note that depending on the location of the return air temperature sensor, it may be necessary to account for temperature rise across the return fan in Rule 8.

Test Segment 5 - Heating Valve Leakage Test

This test compares the supply air, mixed air, and return air temperatures under conditions designed to reveal leakage through the heating coil valve while mitigating the possibility of leakage through the outdoor air damper and cooling coil valve. The test setup is illustrated in Figure 46 and the open-loop control signals used to implement the test are described below.

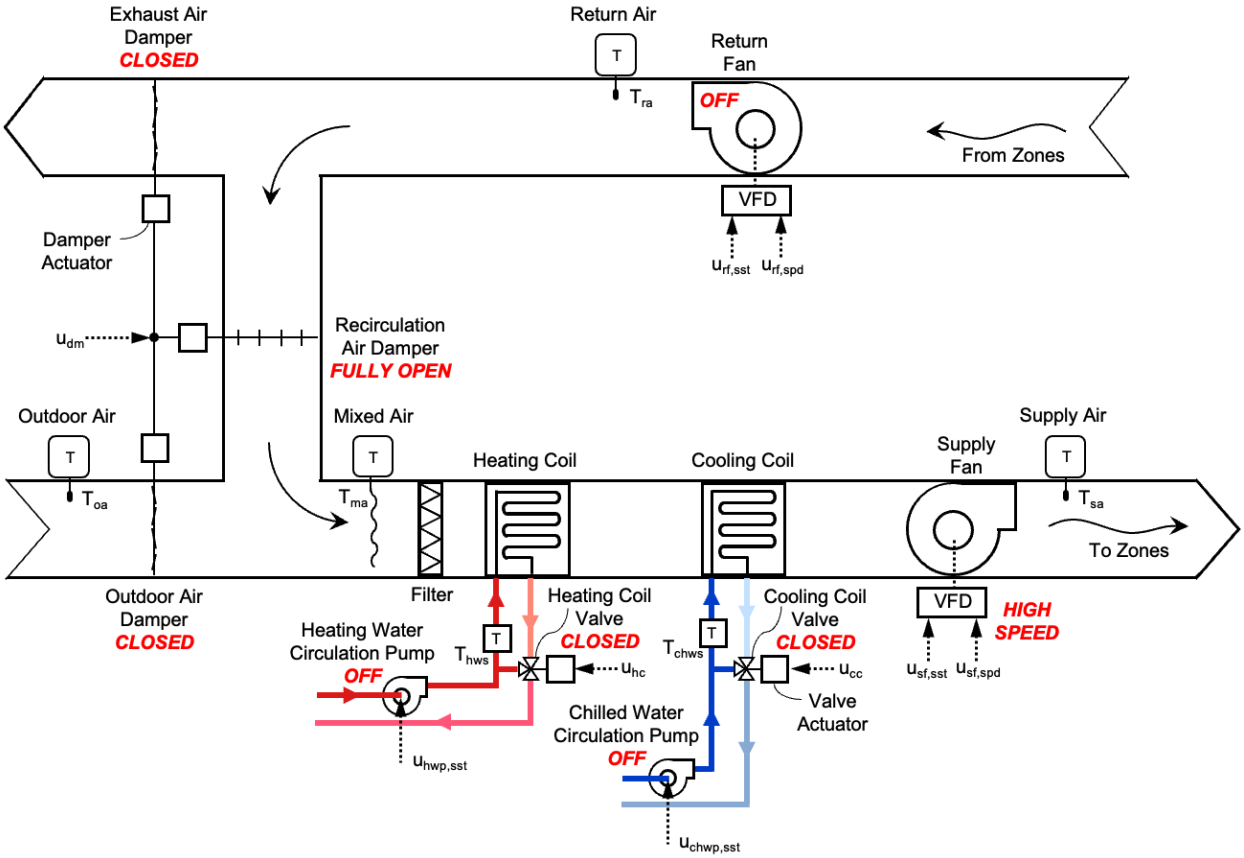


Figure 46: Setup for test segment 5.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is fully open): $u_{dm} = 0$ (0 = closed, 1 = 100% open)
- AHU supply fan is commanded OFF: $u_{sf,sst}$ = Off / False
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded ON: $u_{hwp,sst}$ = On / True
- Chilled water circulation pump is commanded OFF: $u_{chwp,sst}$ = Off / False
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

For the test conditions in Segment 5 the following rules apply:

Rule 9: $| T_{sa} - T_{ma} | \leq \varepsilon_{OL,9}$
Rule 10: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,10}$

Rule 9 is identical to Rule 5, while Rule 10 is identical to Rules 6 and 8. Thus, no additional explanation of the rules is provided.

Note that depending on the location of the return air temperature sensor, it may be necessary to account for temperature rise across the return fan in Rule 10.

Test Segment 6 - Heating Valve Actuator Test
This test compares the supply air, mixed air, and return air temperatures under conditions designed to reveal a heating coil valve that is stuck or has reverse actuation (opening when it should be closing, and vice versa) while mitigating the possibility of leakage through the outdoor air damper and cooling coil valve. The test setup is illustrated in Figure 47 and the open-loop control signals used to implement the test are described below.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

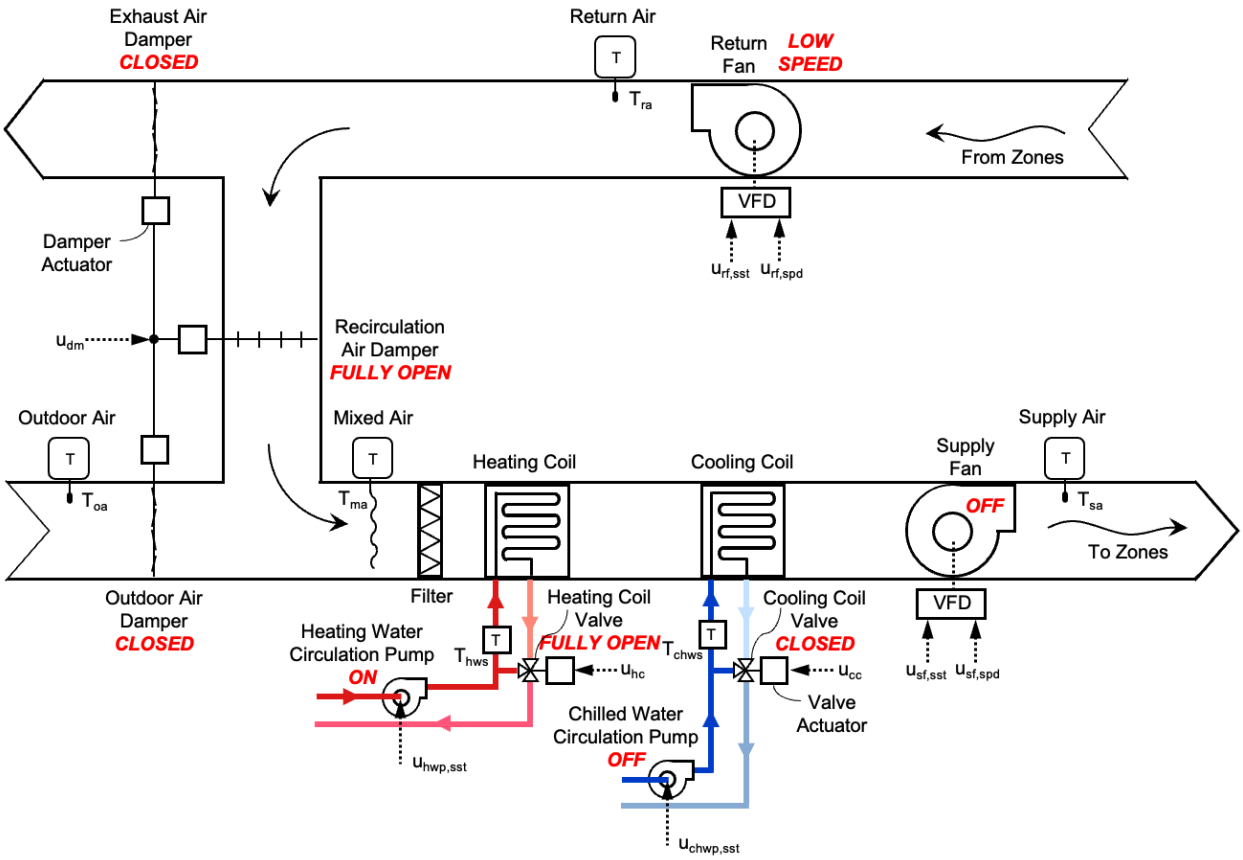


Figure 47: Setup for test segment 6.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is fully open): u_{dm} = 0 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded OFF: $u_{sf,sst}$ = Off / False
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded ON: $u_{hwp,sst}$ = On / True
- Chilled water circulation pump is commanded OFF: $u_{chwp,sst}$ = Off / False
- Heating coil valve is commanded to fully open position: u_{hc} = 1 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

For the test conditions in Segment 6 the following rules apply:

Rule 11: $T_{sa} - T_{ma} \geq \varepsilon_{OL,11}$
Rule 12: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,12}$

If the heating valve is operating as expected, the supply air temperature should be significantly higher than the mixed air temperature and the threshold value in Rule 11 ($\varepsilon_{OL,11}$) should be 10°F

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

or higher. Condition-based faults that are possible explanations for why Rule 11 might be triggered include the following:

- Supply air temperature sensor error
- Mixed air temperature sensor error
- Stuck heating coil valve
- Leaking cooling coil valve
- Stuck cooling coil valve
- Heating water circulation pump fault

Rule 12 is identical to Rules 6, 8, and 10. Depending on the location of the return air temperature sensor, it may be necessary to account for temperature rise across the return fan in this rule.

Test Segment 7 - Cooling Valve Leakage Test

This test compares the supply air, mixed air, and return air temperatures under conditions designed to reveal leakage through the cooling coil valve while mitigating the possibility of leakage through the outdoor air damper and heating coil valve. The test setup is illustrated in Figure 48 and the open-loop control signals used to implement the test are described below.

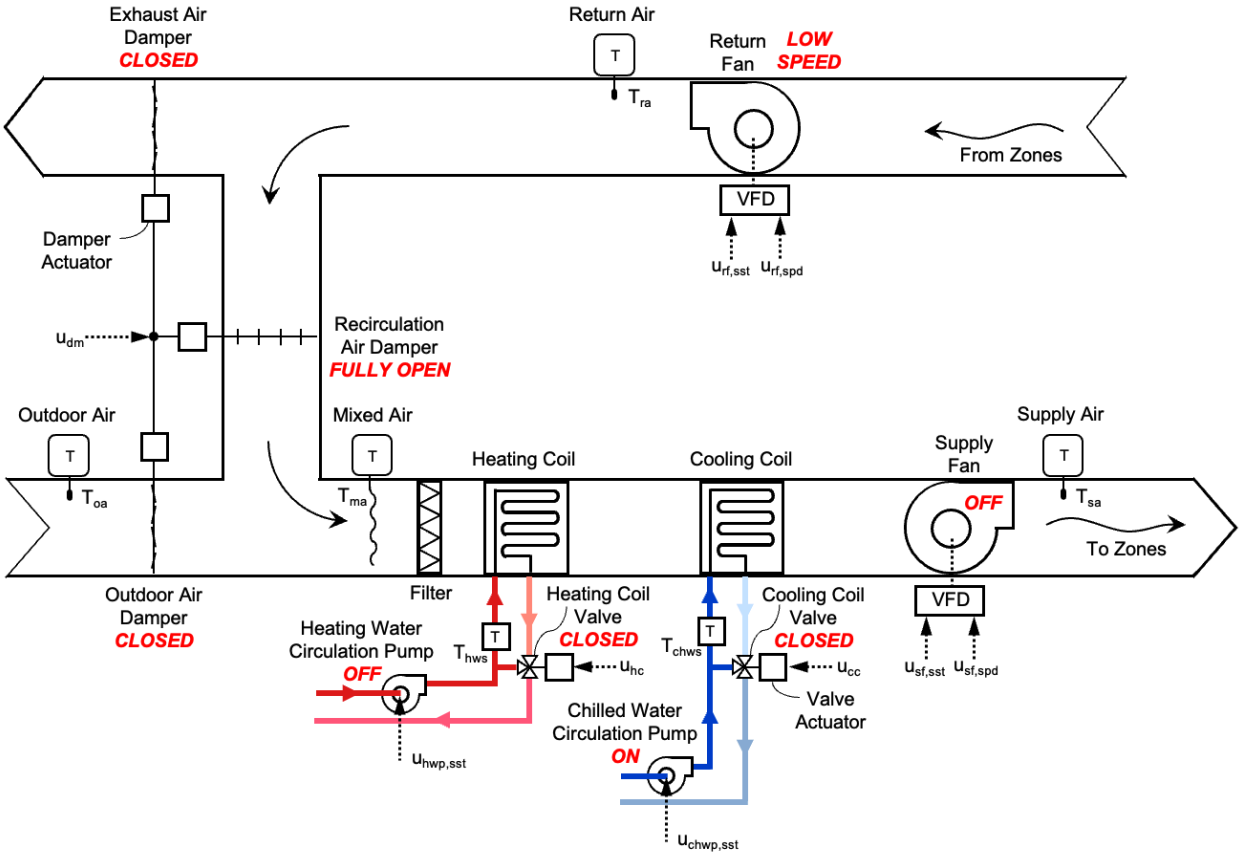


Figure 48: Setup for test segment 7.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is fully open): u_{dm} = 0 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded OFF: $u_{sf,sst}$ = Off / False
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded ON: $u_{chwp,sst}$ = On / True
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to closed position: u_{cc} = 0 (0 = closed, 1 = 100% open)

For the test conditions in Segment 7 the following rules apply:

Rule 13: $| T_{sa} - T_{ma} | \leq \varepsilon_{OL,13}$
Rule 14: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,14}$

Rule 13 is identical to Rules 5 and 9, while Rule 14 is identical to Rules 6, 8, 10, and 12. Depending on the location of the return air temperature sensor, it may be necessary to account for temperature rise across the return fan in Rule 14.

Test Segment 8 - Cooling Valve Actuator Test

The test compares the supply air, mixed air, and return air temperatures under conditions designed to reveal a cooling coil valve that is stuck or has reverse actuation (opening when it should be closing, and vice versa) while mitigating the possibility of leakage through the outdoor air damper and heating coil valve. The test setup is illustrated in Figure 49 and the open-loop control signals used to implement the test are described below.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

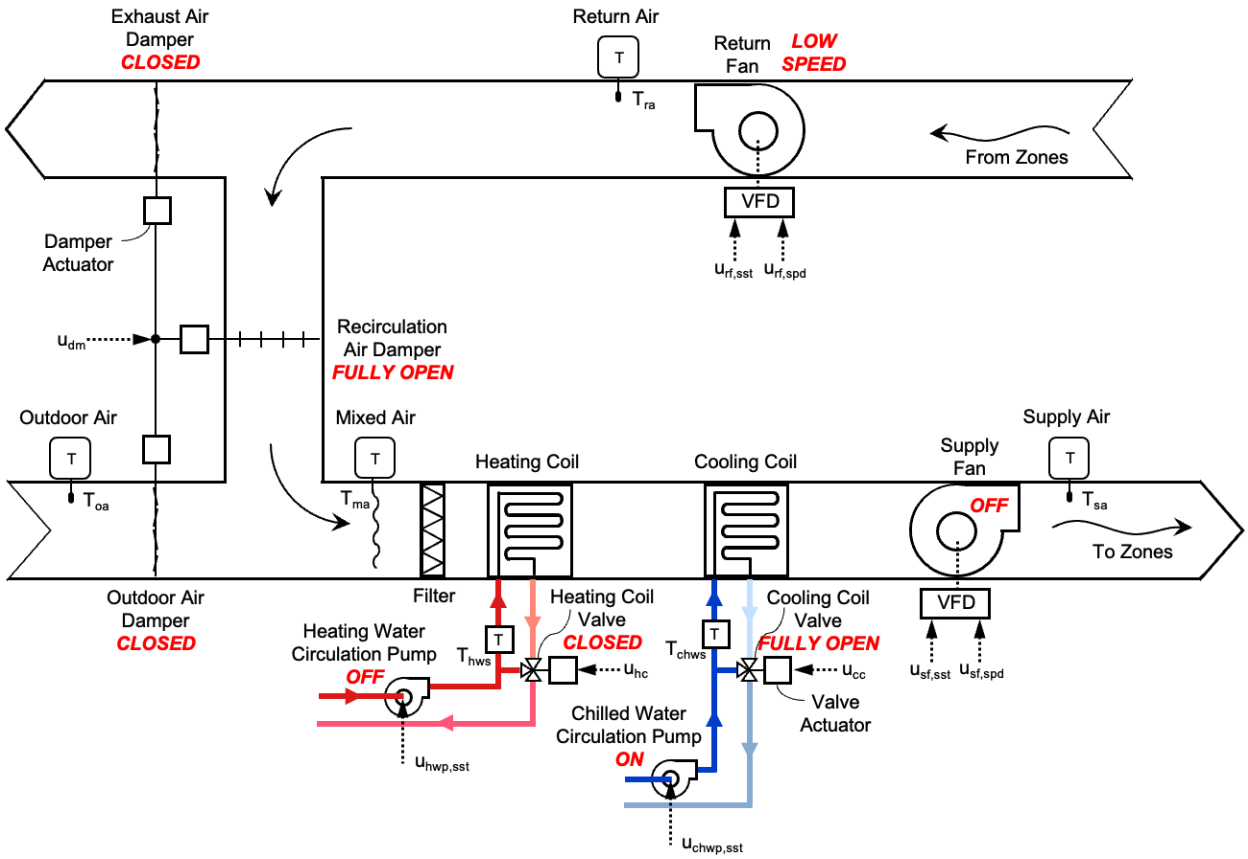


Figure 49: Setup for test segment 8.

Open-loop Control Signals:

- Mixing box dampers are commanded to the 0% outdoor air position (recirculation air damper is fully open): u_{dm} = 0 (0 = closed, 1 = 100% open)
- AHU supply fan is commanded OFF: $u_{sf,sst}$ = Off / False
- AHU return fan is commanded ON: $u_{rf,sst}$ = On / True
- AHU return fan is commanded to operate at low speed: $u_{rf,spd}$ = 30% (adjustable)
- Heating water circulation pump is commanded OFF: $u_{hwp,sst}$ = Off / False
- Chilled water circulation pump is commanded ON: $u_{chwp,sst}$ = On / True
- Heating coil valve is commanded to closed position: u_{hc} = 0 (0 = closed, 1 = 100% open)
- Cooling coil valve is commanded to fully open position: u_{cc} = 1 (0 = closed, 1 = 100% open)

For the test conditions in Segment 8 the following rules apply:

Rule 15: $T_{ma} - T_{sa} \geq \varepsilon_{OL,15}$
Rule 16: $| T_{ra} - T_{ma} | \leq \varepsilon_{OL,16}$

If the cooling valve is operating as expected, the supply air temperature should be significantly lower than the mixed air temperature and the threshold value in Rule 15 ($\varepsilon_{OL,15}$) should be 10°F

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

or higher. Condition-based faults that are possible explanations for why Rule 15 might be triggered include the following:

- Supply air temperature sensor error
- Mixed air temperature sensor error
- Leaking heating coil valve
- Stuck heating coil valve
- Stuck cooling coil valve
- Chilled water circulation pump fault

Rule 16 is identical to Rules 6, 8, 10, 12, and 14. Depending on the location of the return air temperature sensor, it may be necessary to account for temperature rise across the return fan in Rule 16.

5.2.4.2.2 Pseudocode for Test Implementation

The profiles of all commands necessary to implement all test segments of the functional test are shown in Figure 50.

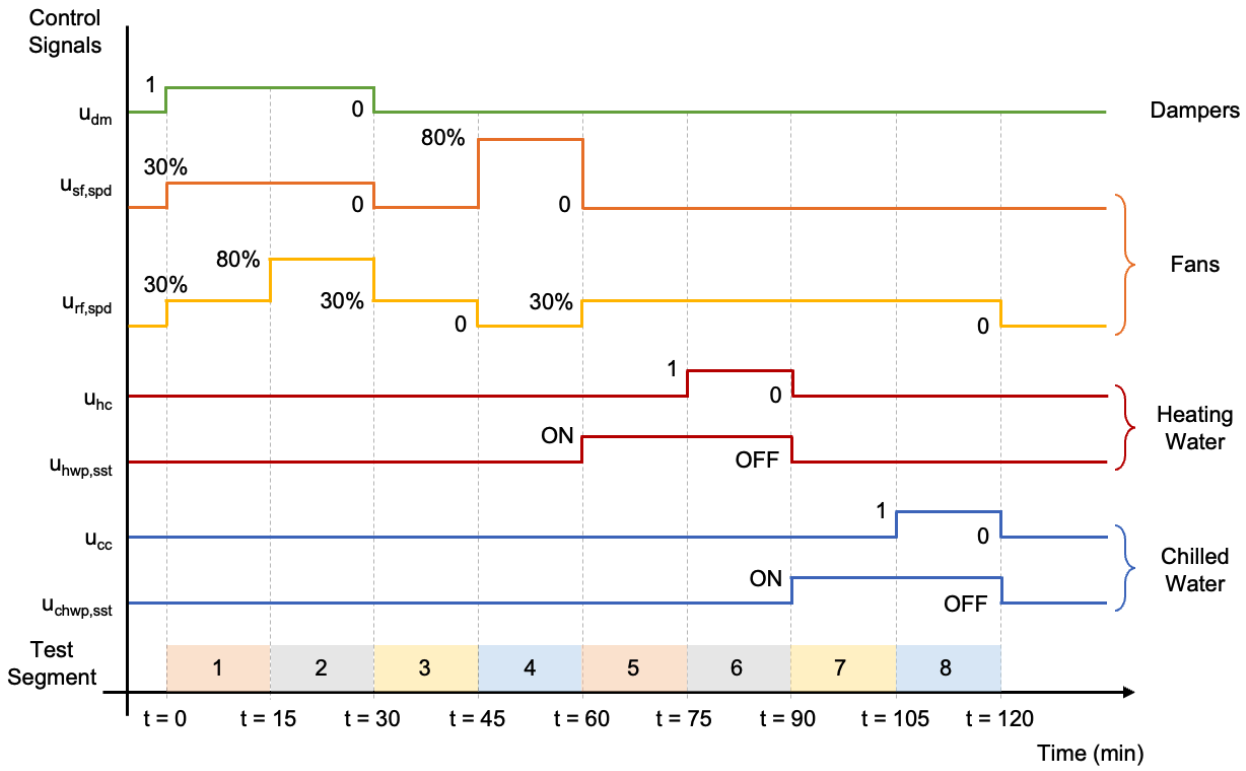


Figure 50: Profiles of control signal for implementing the open-loop functional test.

Pseudocode for implementing *Proactive Functional Test* follows:

```
# If test conditions are satisfied, run the test and store the
#   following data points with timestamp: T_sa, T_ma, T_oa, T_ra,
#   u_cc, u_dm, u_hc, u_sf,sst, u_sf,spd, u_rf,sst, u_rf,spd, u_chwp,sst, and u_hwp,sst
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
     # if runTest == "True":
          # testSeg = 1
          # Initialize control signal values
          # u_dm = 0
          # u_sf,sst = OFF
          # u_sf,spd = 0%
          # u_rf,sst = OFF
          # u_rf,spd = 0%
          # u_hwp,sst = OFF
          # u_chwp,sst = OFF
          # u_hc = 0
          # u_cc = 0
          # while testSeg < 9:
               # testSegCount = 0
               # testSegStart[testSeg] = currentTime
               # testSegLabel[testSeg] = testSeg      # intended
test
               # while testSegCount < 15:  # each segment runs 15
min
                    # if testSeg == 1: # update from initialization
                         # u_dm = 1
                         # u_sf,sst = ON
                         # u_sf,spd = 30%
                         # u_rf,sst = ON
                         # u_rf,spd = 30%
                    # elif testSeg == 2: # update from testSeg = 1
                         # u_rf,spd = 80%
                    # elif testSeg == 3:  # update from testSeg = 2
                         # u_dm = 0
                         # u_sf,sst = OFF
                         # u_sf,spd = 0%
                         # u_rf,spd = 30%
                    # elif testSeg == 4: # update from testSeg = 3
                         # u_sf,sst = ON
                         # u_sf,spd = 80%
                         # u_rf,sst = OFF
                         # u_rf,spd = 0%
                    # elif testSeg == 5: # update from testSeg = 4
                         # u_sf,sst = OFF
                         # u_sf,spd = 0%
                         # u_rf,sst = ON
                         # u_rf,spd = 30%
                         # u_hwp,sst = ON
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                    # elif testSeg == 6: # update from testSeg = 5
                          # u_hc = 1
                    # elif testSeg == 7: # update from testSeg = 6
                          # u_hwp,sst = OFF
                          # u_chwp,sst = ON
                          # u_hc = 0
                    # elif testSeg == 8: # update from testSeg = 7
                          # u_cc = 1
                    # testSegCount += 1
               # testSegEnd[testSeg] = currentTime
               # testSeg += 1
         # Reset control signals to initial values
         # u_dm = 0
         # u_sf,sst = OFF
         # u_sf,spd = 0%
         # u_rf,sst = OFF
         # u_rf,spd = 0%
         # u_hwp,sst = OFF
         # u_chwp,sst = OFF
         # u_hc = 0
         # u_cc = 0
         #
         # runAnalysis = "True"
   else:
         # runAnalysis = "False"
         # goto - "Functional Test Analysis - Step 3"
```

5.2.4.3 Functional Test Analysis - Steps 3 - 7

This section describes the analysis of the test data from 4.2 *Proactive Functional Test*. The analysis is comprised of the following steps:

- Step 3: Apply the test segment rules from Section 5.2.4.2 to the corresponding data collected in each segment;
- Step 4: Identify shared faults from triggered rules;
- Step 5: Identify excluded faults from non-triggered rules;
- Step 6: Remove excluded faults (Step 5) from shared faults (Step 4); and
- Step 7: Determine if a single root-cause fault has been identified.

Note that if there are no rules triggered in Step 3, the AHU has passed the functional test and all the components are considered to be functioning as expected. Because the individual test setups are implemented by control signal overrides, the test does not assess control sequences and logic.

The following sections provide details of the individual steps in the analysis and pseudocode for their implementation.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Step 3 - Apply test segment rules

The rules defined for a given test segment are applied to averages of temperature sensor measurements for T_{sa}, T_{ra}, T_{ma}, and T_{oa} collected during the segment. We assume data are collected at 1 minute intervals and each test segment is 15 minutes in length. Measurements over the last 10 minutes in each test segment are used to determine the average values used when evaluating the rules. Once the data are divided into segments and averaged, the segment rules are applied. Rules are identified as either triggered or not triggered.

Two arrays are initialized in this step and maintained throughout the test. The array (*trigRules*) will contain triggered rules. A second array (*notTrigRules*) will contain rules that are evaluated and not triggered. After processing the data, each of the two arrays should contain no more than one occurrence of a rule. If the array *trigRules* is empty at the conclusion of the test, the AHU has passed the test. If one or more rules have been triggered, the AHU is deemed to have failed the test and further analysis will be conducted to determine the cause of the failure.

Pseudocode implementing Step 3 follows:

```
# For each test segment 1-8, evaluate the segment rule(s) using
#   measurement averages for the last 10 minutes of the segment
# If runAnalysis == "True":
# Set user-defined thresholds and configure design parameters
      # ε_OL1 = 3°F   # threshold for rule 1
      # ε_OL2 = 2°F   # threshold for rule 2
      # ε_OL3 = 3°F   # threshold for rule 3
      # ε_OL4 = 2°F   # threshold for rule 4
      # ε_OL5 = 2°F   # threshold for rule 5
      # ε_OL6 = 2°F   # threshold for rule 6
      # ε_OL7 = 3°F   # threshold for rule 7
      # ε_OL8 = 2°F   # threshold for rule 8
      # ε_OL9 = 2°F   # threshold for rule 9
      # ε_OL10 = 2°F   # threshold for rule 10
      # ε_OL11 = 10°F  # threshold for rule 11
      # ε_OL12 = 2°F   # threshold for rule 12
      # ε_OL13 = 2°F   # threshold for rule 13
      # ε_OL14 = 2°F   # threshold for rule 14
      # ε_OL15 = 10°F  # threshold for rule 15
      # ε_OL16 = 2°F   # threshold for rule 16
      # ΔT_sf = 1°F  # temperature rise across the supply fan
# Initialize all rules to be satisfied (i.e., not triggered)
      # Rule_i = "True" for i = 1-16
# Process test segment data
      # testSeg = 1
      # trigRules = []    # Initialize array of triggered rules
      # notTrigRules = [] # Initialize array of non-triggered rules
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# while testSeg < 9:
      # Calculate segment averages for Tsa, Tra, Tma, and Toa based
      #   on final 10 segment samples
      # Evaluate segment rules
      # if testSeg == 1:
            # if |Tsa - Tma - ΔTsf| ≤ εOL,1:    # Rule 1
                  # notTrigRules = notTrigRules + Rule_1
            # else:
                  # Rule_1 = "False"
                  # trigRules = trigRules + Rule_1
            # if |Toa - Tma| ≤ εOL,2:    # Rule 2
                  # notTrigRules = notTrigRules + Rule_2
            # else:
                  # Rule_2 = "False"
                  # trigRules = trigRules + Rule_2
      # elif testSeg == 2:
            # if |Tsa - Tma - ΔTsf| ≤ εOL,3:    # Rule 3
                  # notTrigRules = notTrigRules + Rule_3
            # else:
                  # Rule_3 = "False"
                  # trigRules = trigRules + Rule_3
            # if |Toa - Tma| ≤ εOL,4:    # Rule 4
                  # notTrigRules = notTrigRules + Rule_4
            # else:
                  # Rule_4 = "False"
                  # trigRules = trigRules + Rule_4
                  .
                  .
                  .
      # elif testSeg == 8:
            # if Tma - Tsa ≥ εOL,15:
                  # notTrigRules = notTrigRules + Rule_15
            # else:
                  # Rule_15 = "False"
                  # trigRules = trigRules + Rule_15
            # if |Tra - Tma| ≤ εOL,16:
                  # notTrigRules = notTrigRules + Rule_16
            # else:
                  # Rule_16 = "False"
                  # trigRules = trigRules + Rule_16
#
# if trigRules = []:
#     functTestResult = "Pass"
# else:
#     functTestResult = "Fail"
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#     goto "Functional Test Analysis - Step 4"
```

Step 4 - Identify shared faults from triggered rules

The multi-mode analysis provides a process for combining the results of the various test segments to determine the smallest possible set of condition-based faults that are possible explanations for a failed functional test. If it is assumed that no more than one fault is present during a functional test and if multiple rules are triggered during the test, then the triggered rules must share at least one condition-based fault among their possible fault explanations. Stated another way, each triggered rule has a set of condition-based faults as possible explanations and the intersection of these sets should contain the actual condition-based fault. Thus, this step in the multi-mode analysis identifies the shared condition-based faults for the triggered rules.

Table 33 lists 13 condition-based faults that are possible explanations for one or more triggered rules. To find the intersection of the set of possible explanations for each triggered rule, a 1x13 array *NcauseTrig* is defined and initialized with all elements equal to 0. The second dimension of this array corresponds to the total number of condition-based faults identified in Table 33. This array will be used to count the number of times each condition-based fault is identified as a possible explanation in the triggered rules. Next, a 1x13 array *cause_rule* (where *rule* = 1 to 16) is defined for each rule, where an element of the array is assigned the value 1 if the condition-based fault corresponding to that element is a possible cause for a particular rule being triggered, and the element is assigned the value 0 otherwise. As an example, referring to Table 33, Rule 1 has six condition-based faults as possible explanations for why this rule might be triggered. The six condition-based faults are:

1 - Supply air temperature sensor error
3 - Mixed air temperature sensor error
5 - Leaking cooling coil valve
6 - Stuck cooling coil valve
8 - Leaking heating coil valve
9 - Stuck heating coil valve

Thus, the array *cause_rule* where *rule* = 1 (i.e., *cause_1*) is defined as:

```
cause_1 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
```

where we note that elements 1, 3, 5, 6, 8, and 9 are set to 1. Arrays *cause_2* through *cause_16* are defined in the same manner.

Next, for each rule in the array *trigRules*, the array *cause_rule* is added to the array *NcauseTrig*. This step counts the number of times each condition-based fault is identified as a possible explanation for a triggered rule. When this step has been completed, there should be one or more elements in *NcauseTrig* that are equal to the number of rules in the array *trigRules*. These elements correspond to the condition-based faults that are the possible causes for the failed

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

functional test.

Pseudocode implementing Step 4 follows:

```
# Initialize 1x13 array for counting possible causes for triggered
#   rules
      #     NcauseTrig = [0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x13 array of possible causes for each of the 16 rules
      #     cause_1 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_2 = [0,0,1,1,0,0,0,0,0,0,1,0,1]
      #     cause_3 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_4 = [0,0,1,1,0,0,0,0,0,0,1,0,1]
      #     cause_5 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_6 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     cause_7 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_8 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     cause_9 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_10 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     cause_11 = [1,0,1,0,1,1,0,0,1,1,0,0,0]
      #     cause_12 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     cause_13 = [1,0,1,0,1,1,0,1,1,0,0,0,0]
      #     cause_14 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     cause_15 = [1,0,1,0,0,1,1,0,1,1,0,0,0]
      #     cause_16 = [0,1,1,0,0,0,0,0,0,0,0,1,1]
      #     rule = 1
      #     numRules = 16
# Count the possible causes for all triggered rules in array
trigRules
      #     While rule <= numRules:
            #     if rule in trigRules:
                  #     NcauseTrig = NcauseTrig + cause_rule
            #     rule += 1
      #     goto “Functional Test Analysis - Step 5”
```

Step 5 - Identify excluded faults from non-triggered rules

In a similar manner, if it is assumed that at most there is one fault present during an open-loop test, specific rules that are evaluated and not triggered can be used to remove one or more condition-based faults as possible explanations for a failed functional test. As an example, if the AHU operates with the outdoor air damper closed and the recirculation damper fully open (100% recirculation air) and the return and mixed air temperatures are equal within the tolerances defined, it can be concluded that the return and mixed air temperature sensors are not faulted. If a mixed air temperature sensor fault is a possible explanation for one or more rules that are triggered in the functional test, that explanation can be removed. Therefore, the knowledge of which rules are not triggered is used in Step 5 to identify excluded faults. Table 35

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

lists rules that can be used in this way. Note that two rules must be evaluated and not triggered to exclude certain condition-based faults.

To implement this step, a 1x13 array *excludeCause* is defined and initialized with all elements equal to 0. Next, a 1x13 array *exclude_rule* is defined for each fault exclusion that is based on a single rule in Table 35. An element of the array is assigned the value 1 if the condition-based fault corresponding to that element can be removed as a possible cause for a failed functional test, and the element is assigned the value 0 otherwise. As an example, the array *exclude_3* (i.e., *rule* is set equal to 3 corresponding to Rule 3 described in Segment 2) is defined as

```
exclude_3 = [1,0,1,0,0,0,0,0,0,0,0,0,0]
```

because, as shown in Table 35, when Rule 3 is evaluated and not triggered, the supply air temperature sensor fault (explanation #1 in Table 33) and mixed air temperature sensor fault (explanation #3 in Table 33) can be removed as possible causes for the failed functional test. Thus, elements 1 and 3 of the array exclude_3 are set to 1. Similar arrays are defined for the remaining fault exclusions in Table 35 based on single rules. Next, for each rule in the array *notTrigRules*, the array *exclude_rule* is added to the array *excludeCause*.

For those cases in Table 35 where two rules must be evaluated and not triggered to exclude a particular condition-based fault, a 1x13 array *exclude_rule_rule* is defined and used as *exclude_rule* is used for the single rule cases.

Table 35: Condition-based faults that can be removed as possible causes of a failed functional test based on rules that are evaluated and not triggered.

Test Segment	Rule(s) Evaluated and not Triggered	Condition-based Fault(s) Removed *
1	Rule 1: $\lvert T_{sa} - T_{ma} - \Delta T_{sf} \rvert \le \varepsilon_{OL,1}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3]
1	Rule 2: $\lvert T_{oa} - T_{ma} \rvert \le \varepsilon_{OL,2}$	Mixed air temperature sensor fault [3] Outdoor air temperature sensor fault [4]
2	Rule 3: $\lvert T_{sa} - T_{ma} - \Delta T_{sf} \rvert \le \varepsilon_{OL,3}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3]
2	Rule 4: $\lvert T_{oa} - T_{ma} \rvert \le \varepsilon_{OL,4}$	Mixed air temperature sensor fault [3] Outdoor air temperature sensor fault [4] Leaking recirculation air damper [11]
3	Rule 5: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,5}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3]
3	Rule 6: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,6}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3]
1 & 3	Rule 2: $\lvert T_{oa} - T_{ma} \rvert \le \varepsilon_{OL,2}$	Stuck mixing box damper(s) [13]

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

	& Rule 6: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,6}$	
4	Rule 7: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,7}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3]
4	Rule 8: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,6}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3] Leaking outdoor air damper [12]
5	Rule 9: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,9}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3] Leaking heating coil valve [8]
5	Rule 10: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,10}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3]
6	Rule 12: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,12}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3]
5 & 6	Rule 9: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,9}$ & Rule 11: $T_{sa} - T_{ma} \ge \varepsilon_{OL,11}$	Stuck heating coil valve [9]
7	Rule 13: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,13}$	Supply air temperature sensor fault [1] Mixed air temperature sensor fault [3] Leaking cooling coil valve [5]
7	Rule 14: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,14}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3]
8	Rule 16: $\lvert T_{ra} - T_{ma} \rvert \le \varepsilon_{OL,16}$	Return air temperature sensor fault [2] Mixed air temperature sensor fault [3]
7 & 8	Rule 13: $\lvert T_{sa} - T_{ma} \rvert \le \varepsilon_{OL,13}$ & Rule 15: $T_{ma} - T_{sa} \ge \varepsilon_{OL,15}$	Stuck cooling coil valve [6]

* The number in brackets in the last column corresponds to the explanation number for the condition-based fault from Table 33.

When Step 5 has been completed, the elements of array *excludeCause* that are greater than or equal to 1 will indicate the condition-based faults that can be removed as a possible cause for the failed functional test.

Pseudo code implementing Step 5 follows:

```
# Initialize 1x13 array for identifying excluded causes based on
rules
#   that are evaluated and not triggered
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
     # excludeCause = [0,0,0,0,0,0,0,0,0,0,0,0,0]
# Initialize 1x13 array of excluded causes for Rules 1-16
     # for i = 1:16:
          # exclude_i = [0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x13 array of excluded causes for each single rule in
#   Table 35
     # Rule 1 evaluated and not triggered
     #    exclude_1 = [1,0,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 2 evaluated and not triggered
     #    exclude_2 = [0,0,1,1,0,0,0,0,0,0,0,0,0]
     # Rule 3 evaluated and not triggered
     #    exclude_3 = [1,0,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 4 evaluated and not triggered
     #    exclude_4 = [0,0,1,1,0,0,0,0,0,0,1,0,0]
     # Rule 5 evaluated and not triggered
     #    exclude_5 = [1,0,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 6 evaluated and not triggered
     #    exclude_6 = [0,1,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 7 evaluated and not triggered
     #    exclude_7 = [1,0,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 8 evaluated and not triggered
     #    exclude_8 = [0,1,1,0,0,0,0,0,0,0,0,1,0]
     # Rule 9 evaluated and not triggered
     #    exclude_9 = [1,0,1,0,0,0,0,1,0,0,0,0,0]
     # Rule 10 evaluated and not triggered
     #    exclude_10 = [0,1,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 12 evaluated and not triggered
     #    exclude_12 = [0,1,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 13 evaluated and not triggered
     #    exclude_13 = [1,0,1,0,1,0,0,0,0,0,0,0,0]
     # Rule 14 evaluated and not triggered
     #    exclude_14 = [0,1,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 16 evaluated and not triggered
     #    exclude_16 = [0,1,1,0,0,0,0,0,0,0,0,0,0]
# Assign 1x13 array of excluded causes for each double rule in
#   Table 35
     # Rule 2 and Rule 6 evaluated and not triggered
     #    exclude_2_6 = [0,0,0,0,0,0,0,0,0,0,0,0,1]
     # Rule 9 and Rule 11 evaluated and not triggered
     #    exclude_9_11 = [0,0,0,0,0,0,0,0,1,0,0,0,0]
     # Rule 13 and Rule 15 evaluated and not triggered
     #    exclude_13_15 = [0,0,0,0,0,1,0,0,0,0,0,0,0]
# Define loop variables
     #    rule = 1
     #    numRules = 16
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# Update array excludeCause using rules in array notTrigRules
      #     for rule = 1:16:
            #     if rule in notTrigRules:
                  #     excludeCause = excludeCause + exclude_rule
      #     if (rule = 2 & rule = 6) in notTrigRules:
            #     excludeCause = excludeCause + exclude_2_6
      #     if (rule = 9 & rule = 11) in notTrigRules:
            #     excludeCause = excludeCause + exclude_9_11
      #     if (rule =13 & rule = 15) in notTrigRules:
            #     excludeCause = excludeCause + exclude_13_15
      #     goto "Functional Test Analysis - Step 6"
```

Step 6 - Remove excluded faults from shared faults

In Step 6, results from the previous two steps are combined to determine the smallest set of condition-based faults that are possible explanations for the failed functional test. More specifically, this procedure removes any faults identified in Step 5 from the shared condition-based faults identified in Step 4.

A 1x13 array *Ncause* is determined by subtracting the array *excludeCause* (Step 5) from the array *NcauseTrig* (Step 4). The element(s) of *Ncause* with the largest magnitude identifies the condition-based fault(s) that is the cause of the failed functional test. Finally, an array *NcauseFinal* is initialized and used to store the largest element(s) of array *Ncause*. *NcauseFinal* will contain the smallest set of condition-based faults that are possible explanations for a failed functional test.

Pseudo code implementing Step 6 follows:

```
# Determine the array of possible causes for the failed
#   functional test based on triggered rules and excluded causes
      #     Ncause = NcauseTrig - excludeCause
# Initialize array NcauseFinal, then store largest element(s) of
#   Ncause in this array
      # NcauseFinal = max(Ncause)
      # goto "Functional Test Analysis - Step 7"
```

Step 7 - Determine if a single root-cause fault identified

The final step of the analysis assesses whether a single condition-based fault has been determined as the cause for a failed functional test. Pseudocode implementing Step 7 of the analysis procedure follows:

```
# Determine if a single possible cause (condition-based fault) has
#   been identified for a failed functional test
      #     If len(NcauseFinal) == 0:
            #     error = "True"   # ERROR -> No causes found
            #     singleCause = "False"
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#     ElseIf len(NcauseFinal) == 1:
      #     error = "False"
      #     singleCause = "True"
#     Else:
      #     error = "False"
      #     singleCause = "False"
# If singleCause == "True":
      #     End of Analysis
# ElseIf error == "False" AND singleCause == "False":
      #     Additional tests or manual analysis needed to
      #       determine the root cause of the failed functional
      #       test
```

5.3 Closed-loop Functional Tests for VAV Terminal Units

5.3.1 Overview

The closed-loop functional tests described in this document are designed to assess the performance of variable-air-volume (VAV) terminal units. The tests involve systematically overriding the zone temperature setpoints of the terminal unit under test in order to cause the unit to transition into different operating modes. These operating modes are experienced during normal operation as building loads vary; however, by overriding the setpoints, mode transitions are accelerated, enabling data for all operating modes to be collected in a 1-2 hour test period. The test data can then be analyzed to determine if the terminal unit is operating as expected and, if not, what fault(s) underlie current operation.

5.3.2 System Description

The system undergoing testing is a single duct pressure-independent VAV terminal unit with hydronic reheat. A schematic diagram of the system is shown in Figure 51. This type of terminal unit consists of a damper and airflow sensor for controlling the volume of air delivered to a zone, and a reheat coil, reheat coil valve, and discharge air temperature sensor that are used to control and/or monitor the temperature of the air delivered to a zone.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

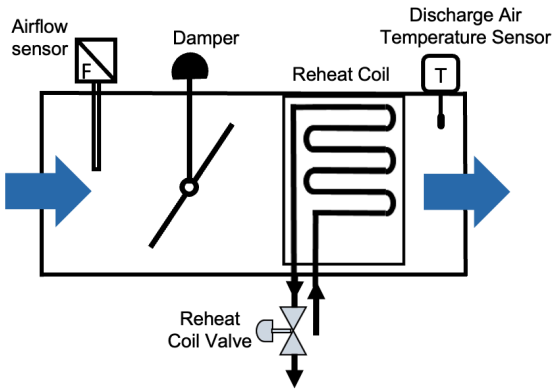


Figure 51: Schematic diagram of a single duct VAV terminal unit with hydronic reheat.

Figure 52 shows a conventional sequence of operation for a single duct VAV terminal unit with hydronic reheat. There are three modes of operation, namely, cooling mode, deadband mode, and heating mode. The mode that is active at any given time is dictated by the zone air temperature in relation to the zone heating and cooling setpoints, and the capacity of heating or cooling delivered at any given time is dictated by the zone temperature error (i.e., the difference between the zone temperature and the setpoint corresponding to the active mode).

The cooling mode is active when the zone temperature T_{za} is greater than the zone cooling setpoint $T_{za,clg,set}$. In the cooling mode the airflow setpoint increases as the zone temperature error increases. The damper is modulated to maintain the actual airflow rate at the time-varying setpoint and the reheat coil valve is commanded closed. The maximum airflow setpoint for each terminal unit is based on design conditions for the zone it serves.

The heating mode is active when the zone temperature T_{za} is less than the zone heating setpoint $T_{za,htg,set}$. In the heating mode, the reheat coil valve is controlled to raise the zone temperature and maintain it at the heating setpoint, and the damper is controlled to maintain the airflow rate at its minimum setpoint.

The deadband mode is active when the zone temperature T_{za} is between the zone heating and cooling setpoints ($T_{za,htg,set}$ and $T_{za,clg,set}$). In the deadband mode the damper is modulated to maintain the airflow rate at its minimum setpoint and the reheat coil valve is commanded closed.

Figure 52 is just one possible control sequence for VAV terminal units. The dual-maximum control sequence (Taylor et al., 2012) is also common and is shown in Figure 53. The two control sequences differ only in the heating mode. As illustrated in Figure 53, in the heating mode the dual-maximum sequence first controls the heating valve to maintain the discharge air temperature at a setpoint that is reset based on the zone temperature error, and then controls the terminal unit damper to maintain the airflow rate at a setpoint that is reset based on the zone temperature error. For purposes of describing the closed-loop functional test procedure and analysis, we base the discussion on the conventional sequence in Figure 52. Changes that are

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

necessary when the dual-maximum control sequence is implemented are described in footnotes.

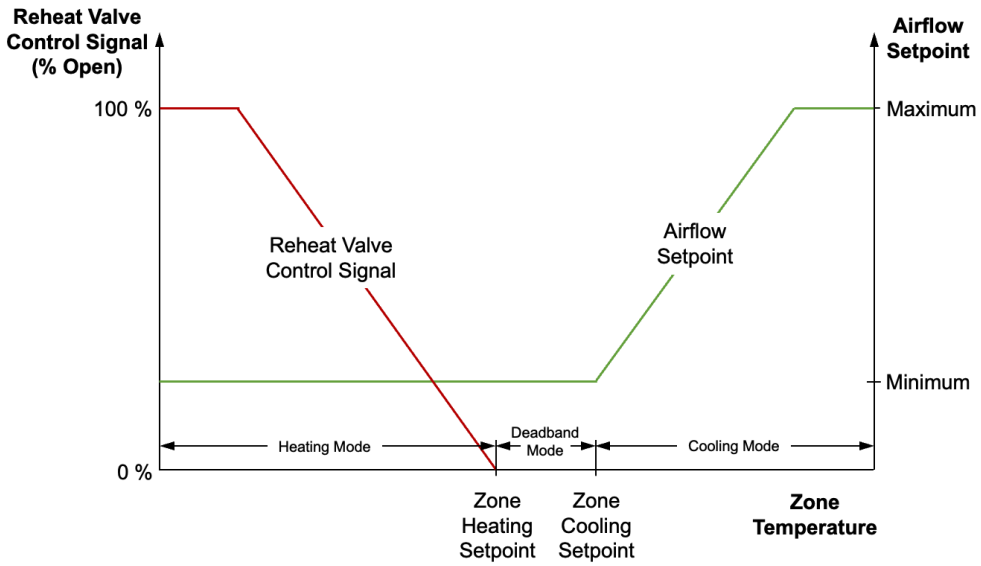


Figure 52: Illustration of a conventional control sequence for a single duct VAV terminal unit with hydronic reheat.

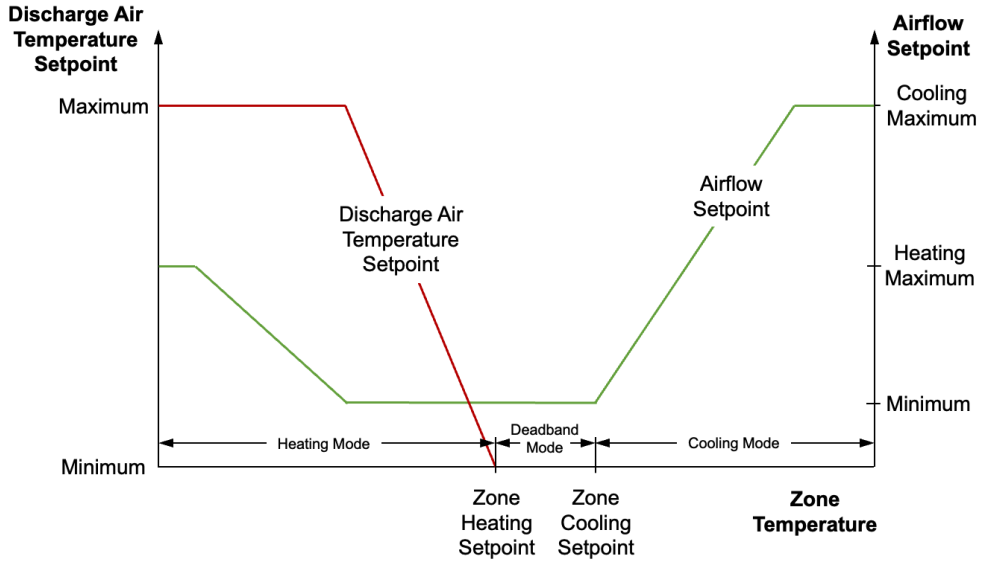


Figure 53: Illustration of the dual-maximum control sequence for a single duct VAV terminal unit with hydronic reheat.

5.3.3 Process Description

Figure 54 shows a process flow diagram for the closed-loop testing and analysis of terminal units. The process consists of two modules outlined with dashed lines. The *Proactive Functional Test* module involves verifying that the required conditions for the test are satisfied, and executing the test through adjustments of the zone heating and cooling setpoints that force the

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

terminal unit into its different modes of operation. The *Functional Test Analysis* module uses the test data to evaluate rules describing the expected performance in each mode of operation. When a functional test fails because a rule defining expected performance is not satisfied, additional analysis is performed in an effort to determine the root cause fault. Two methods for performing this additional analysis are identified, namely, the multi-mode analysis (Lin et al., 2025) and the inverse multi-mode analysis. Further explanation of each of these processes for analyzing a failed test is provided in Section 5.3.5.2.2.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

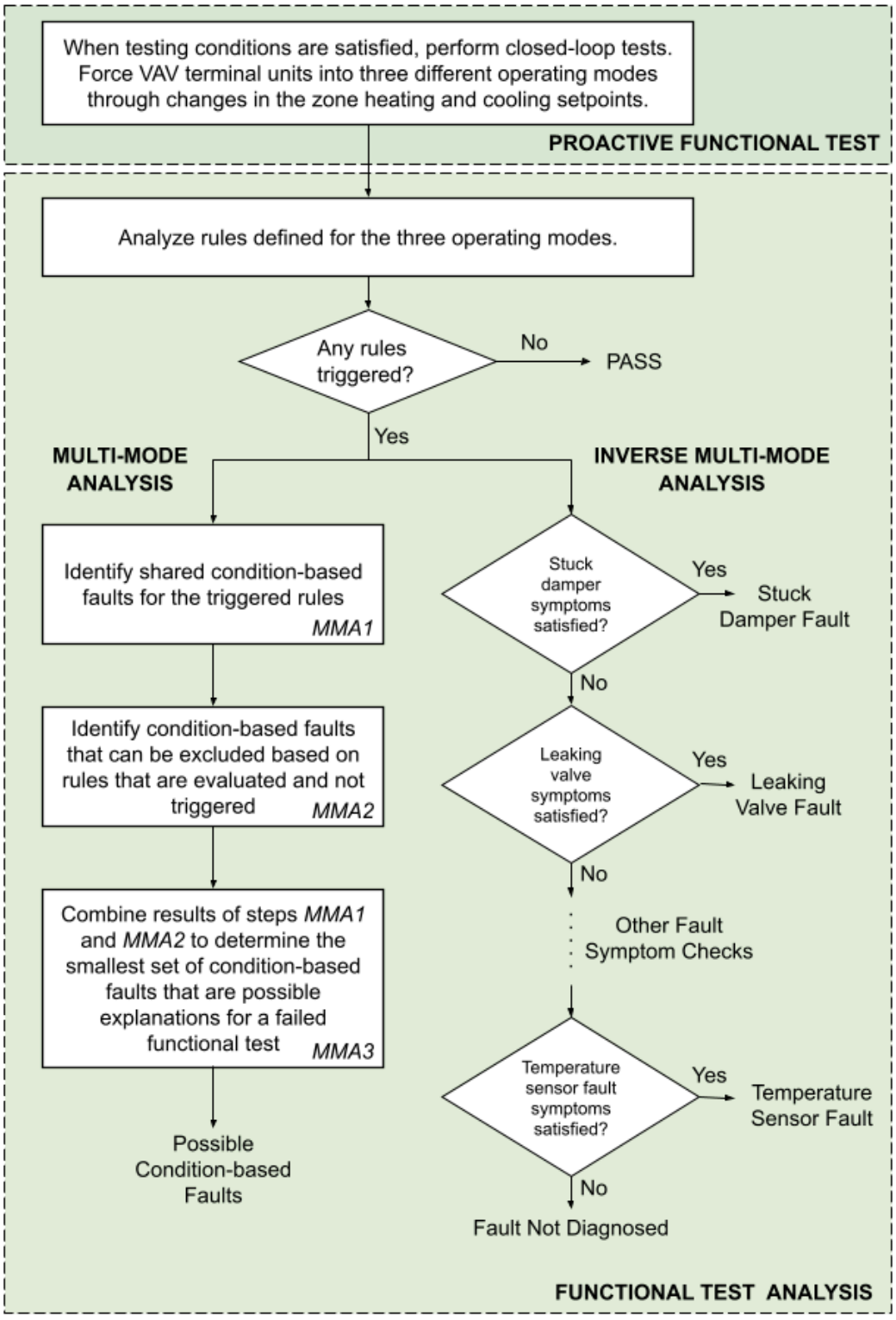


Figure 54: Process flow diagram for the closed-loop functional test and analysis for terminal units.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.3.4 Required Data Points

Measurement and control points from the building automation system (BAS) that are used in this test are listed in Table 36. All other variables and parameters used to implement the functional test and analyze its results are listed in Table 37.

Table 36: Data points needed from the building automation system.

Abbreviation	Data point	Data type	Requirements (Read/Write)
AHU_{occ}	AHU occupancy status [0 = unoccupied; 1 = occupied]	Control setting	R
F_{vav}	VAV terminal unit airflow rate [CFM]	Sensor reading	R
$F_{vav,clg,max}$	VAV terminal unit design maximum airflow rate in the cooling mode [CFM]	Control parameter	R
$F_{vav,htg,max}$	VAV terminal unit design maximum airflow rate in the heating mode [CFM]	Control parameter	R
$F_{vav,min}$	VAV terminal unit design minimum airflow rate [CFM]	Control parameter	R
$F_{vav,set}$	VAV terminal unit airflow setpoint [CFM]	Control parameter	R
P_{sa}	AHU supply air static pressure [in H_2O]	Sensor reading	R
$P_{ss,set}$	AHU supply air static pressure setpoint [in H_2O]	Control parameter	R
$T_{vav,da}$	VAV discharge air temperature [°F]	Sensor reading	R
T_{hws}	Heating water supply temperature [°F]	Sensor reading	R
T_{sa}	AHU supply air temperature [°F]	Sensor reading	R
$T_{sa,set}$	AHU supply air temperature setpoint [°F]	Control parameter	R
T_{za}	Zone air temperature [°F]	Sensor reading	R
$T_{za,clg,set}$	zone cooling setpoint [°F]	Control parameter	R/W
$T_{za,htg,set}$	zone heating setponit [°F]	Control parameter	R/W
$u_{hwp,stat}$	heating water circulating pump status [ON/OFF]	Status feedback signal	R
$u_{vav,dm}$	VAV damper control signal [0 = closed; 1 = open]	Control signal	R

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$u_{vav,rh}$	VAV reheat valve control signal [0 = closed; 1 = open]	Control signal	R
ZN_{occ}	Zone occupancy status [0 = unoccupied; 1 = occupied]	Control setting	R

Table 37: Variables and parameters used to implement and evaluate the functional test.

Abbreviation	Data point	Data type
cause_rule	[1 x 14] binary array defining possible condition-based faults for rule i (i = 1 to 11)	Algorithm parameter
count	variable for counting the number of times the airflow control criterion in Rules 1, 5, and 9 is satisfied	Algorithm variable
countPsa	counter for number of times the AHU static pressure control criterion is satisfied	Algorithm variable
countTsa	counter for number of times the AHU supply air temperature control criterion is satisfied	Algorithm variable
countS1	counter for number of times the saturation criteria in symptom S1 is satisfied	Algorithm variable
countS2	counter for number of times the saturation criteria in symptom S2 is satisfied	Algorithm variable
e_{Psa}	supply air pressure error ($P_{sa,set}$ - P_{sa}) [in H_2O]	Algorithm variable
e_{Tsa}	AHU supply air temperature error ($T_{sa,set}$ - T_{sa}) [°F]	Algorithm variable
excludeCause	[1 x 14] binary array of excluded causes based on rules that are evaluated but not triggered	Algorithm variable
exclude_rule	[1 x 14] binary array of condition-based faults that can be excluded based on rule i (i = 1, 3, 5, 7, 9, and 11) in Table 39	Algorithm variable
functTestResult	binary variable indicating the outcome of the functional test [Pass or Fail]	Algorithm variable
Ncause	array of possible causes for a failed functional test based on triggered rules and excluded causes from the cases in Table 39	Algorithm variable
NcauseFinal	array of the largest element(s) of array “Ncause” which contains the condition-based fault(s) most likely to be the root cause of the failed test	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

NcauseTrig	[1 x 14] array for counting the number of times individual condition-based faults are identified as possible causes for a triggered rule	Algorithm variable
notTrigRules	list of rules where APAR rules are evaluated and not triggered	Algorithm variable
numRules	integer number of rules	Algorithm parameter
$P_{sa,set,max}$	maximum allowable supply air pressure setpoint during operation [in H_2O]	User-defined parameter
Rule_i	Binary variable indicating if the ith rule is satisfied (True) or triggered/violated (False)	Algorithm variable
runTest	binary variable indicating if the test should be executed [True / False]	Algorithm variable
testSeg	integer variable for identifying individual test segments (modes) [1 to 3]	Algorithm variable
testSegCount	integer variable for counting samples in a test segment	Algorithm variable
testSegEnd	real array with timestamp corresponding to the end of each test segment [1 x 3 array]	Algorithm variable
testSegLabel	integer array with label (“testSeg”) for each test segment [1 x 3 array]	Algorithm variable
testSegStart	real array with timestamp corresponding to the start of each test segment [1 x 3 array]	Algorithm variable
trigRules	String array of triggered rules	Algorithm variable
$T_{hws,min}$	minimum heating water supply temperature for executing the functional tests [°F]	User-defined parameter
$T_{vav,da,max}$	maximum allowable discharge air temperature from the terminal unit [°F]	User-defined parameter
ΔT_i	temperature rise across the terminal unit reheat coil in mode i (i = 1 to 3) [°F]	Variable used to describe fault symptoms
Inverse Multi-Mode Diagnoses		
LkgVlv	leaking reheat valve fault	Algorithm variable
StkClosedVlv	stuck closed reheat valve fault	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

StkDamper	stuck damper fault	Algorithm variable
StkOpenVlv	stuck open reheat valve fault	Algorithm variable
TempSenFault	temperature sensor fault	Algorithm variable
VlvActRev	valve actuator reversal fault	Algorithm variable
Thresholds		
Pct_{eTsa}	threshold for supply air temperature control - percentage of samples that must be under control [%]	User-defined parameter
Pct_{ePsa}	threshold for static pressure control - percentage of samples that must be under control [%]	User-defined parameter
Pct_{Fvav}	threshold for VAV terminal unit airflow control - percentage of samples that must be under control [%]	User-defined parameter
Pct_{S1}	threshold for symptom S1 - percentage of samples that satisfy S1 saturation criteria [%]	User-defined parameter
Pct_{S2}	threshold for symptom S2 - percentage of samples that satisfy S2 saturation criteria [%]	User-defined parameter
ε_{dm}	threshold for VAV terminal unit damper saturation [-]	User-defined parameter
ε_{Fset}	threshold for VAV terminal unit airflow setpoint [-]	User-defined parameter
ε_{Fvav}	threshold for VAV terminal unit airflow control [-]	User-defined parameter
$\varepsilon_{\Delta F}$	threshold for airflow comparisons [-]	User-defined parameter
ε_{LVi}	threshold value/coefficient for symptom i of a leaking valve [°F]	User-defined parameter
ε_{Psa}	threshold for static pressure control [in H_2O]	User-defined parameter
$\varepsilon_{rh,max}$	threshold for maximum control signal to reheat coil valve [-]	User-defined parameter
$\varepsilon_{rh,min}$	threshold for minimum control signal to reheat coil valve [-]	User-defined parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

ε_{SCVi}	threshold value/coefficient for symptom i of a stuck closed valve [°F]	User-defined parameter
ε_{SOVi}	threshold value/coefficient for symptom i of a stuck open valve [°F]	User-defined parameter
ε_{TSi}	threshold value/coefficient for symptom i of a temperature sensor fault [°F]	User-defined parameter
ε_{Tsa}	threshold for supply air temperature control [°F]	User-defined parameter
$\varepsilon_{\Delta T,max}$	threshold for maximum temperature rise across the reheat coil ($T_{vav,da}$ - T_{sa}) [°F]	User-defined parameter
$\varepsilon_{\Delta T,min}$	threshold for minimum temperature rise across the reheat coil ($T_{vav,da}$ - T_{sa}) [°F]	User-defined parameter
ε_{VARi}	threshold value/coefficient for symptom i of a valve actuator reversal fault [°F]	User-defined parameter

5.3.5 Description of the Closed-loop Functional Test and Analysis

5.3.5.1 Proactive Functional Test

This section describes the components of the first module of the closed-loop functional test and analysis - the proactive functional test.

5.3.5.1.1 Verify Test Conditions

Prior to executing the test, the following conditions are to be verified:

- The occupancy status of the AHU (AHU_{occ}) and zones (ZN_{occ}) used in the test are “True” or “occupied”
- If implemented, static pressure reset and supply air temperature reset strategies are disabled to avoid interfering with the current test
- AHU supply air temperature (T_{sa}) is maintained at setpoint ($T_{sa,set}$)
- AHU static pressure (P_{sa}) is maintained at setpoint ($P_{sa,set}$)
- Heating water circulation pump is operational as indicated by pump status ($u_{hwp,stat}$) or a similar indicator of pump operation
- Heating water supply temperature (T_{hws}) meets or exceeds its minimum value ($T_{hws,min}$)

AHU supply air conditions are verified to ensure the inlet conditions to the VAV terminal units are appropriate for testing. This aspect of the verification should be performed over a 10-30 minute interval to ensure stable operation of the AHU. Other conditions listed above can be verified with a single measurement just prior to initiating testing. Note that because multiple VAV terminal units may be tested simultaneously, it is recommended that the static pressure setpoint ($P_{sa,set}$) be set to its maximum allowable value ($P_{sa,set,max}$) for testing.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

In the pseudocode that follows, the pretest verification of testing conditions is conducted over a 15-minute period and the control of the AHU supply air temperature and static pressure are assessed over the final 10 minutes of the pretest period using 1-minute data samples. To verify the AHU conditions, the supply air temperature error, e_{Tsa}, and the static pressure error, e_{Psa}, are defined as

$$e_{Tsa} = T_{sa,set} - T_{sa} \quad (1)$$

and

$$e_{Psa} = P_{sa,set} - P_{sa} \quad (2)$$

The AHU supply air temperature is considered under control if the absolute value of e_{Tsa} is less than or equal to a user-defined threshold for a sufficient percentage of data samples in the last 10 minutes of the pretest period. In equation form, the supply air temperature error criterion is given by

$$| e_{Tsa,i} | \leq \varepsilon_{Tsa} \text{ for at least } Pct_{eTsa} \text{ of all samples i} \quad (3)$$

where $e_{Tsa,i}$ is the supply air temperature error for sample i, ε_{Tsa} is a user-defined error threshold (e.g., 1°F, adjustable), and Pct_{eTsa} is a user-defined percentage threshold (e.g., 80%, adjustable).

Similarly, the static pressure is considered under control if the absolute value of e_{Psa} is less than or equal to a user-defined threshold for a sufficient percentage of data samples in the last 10 minutes of the pretest period. In equation form, the static pressure error criterion is given by

$$| e_{Psa,i} | \leq \varepsilon_{Psa} \text{ for at least } Pct_{ePsa} \text{ of all samples i} \quad (4)$$

where $e_{Psa,i}$ is the static pressure error for sample i, ε_{Psa} is a user-defined error threshold (e.g., 0.1 inH_20, adjustable), and Pct_{ePsa} is a user-defined percentage threshold (e.g., 80%, adjustable). The thresholds in Equations (3) and (4) can be adjusted based on field experience, owner requirements, number of data samples, etc.

Pseudocode verifying that test conditions are satisfied is provided below.

```
# Verify test conditions are satisfied
# Read and store the following measurement and control points (with
#   timestamp) at each sampling time during the pretest period:
#   T_sa, T_sa,set, P_sa, P_sa,set, u_hwp,stat, T_hws, ZN_occ, and AHU_occ
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
# Define setpoints, parameters, and threshold values
#   T_sa,set = 55°F
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#    Psa,set = Psa,set,max (user-defined maximum static pressure setpoint)
#    Thws,min = 120°F
#    εTsa = 1°F
#    εPsa = 0.1 inH20
#    PCTeTsa = 80%
#    PCTePsa = 80%
# For each data sample i, compute the supply air temperature and
#    static pressure errors
      # eTsa,i = Tsa,set,i - Tsa,i for all i in pretest period
      # ePsa,i = Psa,set,i - Psa,i for all i in pretest period
# Initialize counters
      # countTsa = 0
      # countPsa = 0
      # i = 1       # sample iterator
      # imax = 15   # example with 15 minute pretest period
      # while i < (imax + 1):
            # if i > 5 & abs(eTsa,i) ≤ εTsa:  # consider last 10 samples
                  # countTsa += 1
            # if i > 5 & abs(ePsa,i) ≤ εPsa:  # consider last 10 samples
                  # countPsa += 1
            # i += 1
      # if (AHUocc == "True"
      #  & ZNocc == "True"
      #  & countTsa/10*100 ≥ PCTeTsa  # divide by 10 1-min. samples
      #  & countPsa/10*100 ≥ PCTePsa
      #  & Thws ≥ Thws,min
      #  & uhwp,stat == "True"):
            # runTest = "True"
      # else:
            # runTest = "False"
```

5.3.5.1.2 Perform Closed-loop Tests

If *runTest* is "True", the *Proactive Functional Test* outlined in the process flow diagram in Figure 54 is executed. The test involves overriding the occupied heating and cooling setpoints ($T_{za,htg,set}$ and $T_{za,clg,set}$) for an individual zone every 30 minutes to cause the terminal unit's sequencing logic to force the unit into each of its three modes of operation, beginning with the cooling mode in the first 30-minute test segment, then the deadband mode, and finally the heating mode. The setpoints used in the three operating modes are determined from the value of the zone air temperature (T_{za}) at the beginning of each 30-minute test segment as follows:

- Cooling Mode: $T_{za,clg,set} = T_{za}$ - 10°F; $T_{za,htg,set} = T_{za}$ - 20°F
- Deadband Mode: $T_{za,clg,set} = T_{za}$ + 10°F; $T_{za,htg,set}$ = same value as used in cooling mode
- Heating Mode: $T_{za,clg,set} = T_{za}$ + 20°F; $T_{za,htg,set} = T_{za}$ + 10°F

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Note that setpoints selected for the heating and cooling modes are intended to be sufficiently above or below the current zone air temperature such that the VAV terminal unit will operate to provide maximum heating or maximum cooling throughout the test of that mode. Note further that when transitioning from the cooling mode to the deadband mode, it is only necessary to adjust the zone cooling setpoint. Finally we note that the order in which the setpoints are adjusted may be important. When lowering the setpoints to force operation in the cooling mode, it is recommended to adjust the zone heating setpoint first so that $T_{za,htg,set}$ is always lower than $T_{za,clg,set}$. For the same reason, when raising the setpoints, the zone cooling setpoint should be adjusted first. When the test has been completed, the overrides of $T_{za,clg,set}$ and $T_{za,htg,set}$ are released and the setpoints return to their normal values.

As the test is performed, data are collected and stored at a fixed sampling rate. A sampling rate of 1 minute is preferred and assumed here. The start and end time of each test segment should be recorded, and each test segment should be labeled 1-3 (e.g., cooling = 1, deadband = 2, heating = 3) to designate the mode of operation that was intended for that segment. The data to be read and stored are AHU_{occ}, T_{sa}, $T_{sa,set}$, P_{sa}, $P_{sa,set}$, T_{hws}, $T_{vav,da}$, T_{za}, $T_{za,clg,set}$, $T_{za,htg,set}$, F_{vav}, $u_{vav,dm}$, $u_{vav,rh}$, and ZN_{occ}. Descriptions of all data points are provided in Table 36.

Pseudocode for implementing the functional test follows:

```
# If test conditions are satisfied, run the test and store the
#   following data points with timestamp: AHU_occ, T_sa, T_sa,set, P_sa,
P_sa,set,
#   T_hws, T_vav,da, T_za, T_za,clg,set, T_za,htg,set, F_vav, u_vav,dm, u_vav,rh, and ZN_occ
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
     # if runTest == True:
          # testSeg = 1
          # while testSeg < 4:
               # Read current T_za   # update each new test segment
               # testSegCount = 0
               # testSegStart[testSeg] = currentTime
               # testSegLabel[testSeg] = testSeg      # intended
mode
               # while testSegCount < 30:  # each segment runs 30
min
                    # if testSeg == 1:    # Cooling mode
                         # T_za,htg,set = T_za - 20°F
                         # T_za,clg,set = T_za - 10°F
                    # elif testSeg == 2:  # Deadband mode
                         # T_za,clg,set = T_za + 10°F
                    # elif testSeg == 3:  # Heating mode
                         # T_za,clg,set = T_za + 20°F
                         # T_za,htg,set = T_za + 10°F
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                # testSegCount += 1
            # testSegEnd[testSeg] = currentTime
            # testSeg += 1
    # Release overrides of the cooling and heating setpoints
    # T_za,clg,set = null
    # T_za,htg,set = null
```

5.3.5.2 Functional Test Analysis

This section describes the components of the second module of the closed-loop functional test and analysis - the functional test analysis.

5.3.5.2.1 Rule Set

Steady-state rules applicable to each of the modes of operation for a VAV terminal unit are described in this section. Recall that each mode is active for a 30-minute test segment. To minimize the effects of transients associated with mode changes, it is recommended that only the final 15 minutes of data in a given mode be used to assess the validity of the rules in a given mode.

5.3.5.2.1.1 Cooling Mode Rules

Rules that are applicable to operation in the cooling mode are described below. For each rule, a list of condition-based faults is provided that are considered possible explanations for a triggered rule.

$$\text{Rule 1: } |F_{vav} - F_{vav,set}| \leq \varepsilon_{Fvav} F_{vav,set} \quad \text{for at least } Pct_{Fvav} \text{ of samples} \tag{5}$$

$$\text{Rule 2: } u_{vav,rh} \leq \varepsilon_{rh,max} \tag{6}$$

$$\text{Rule 3: } |T_{sa} - T_{vav,da}| \leq \varepsilon_{\Delta T,max} \tag{7}$$

Rule 1 states that the airflow rate F_{vav} through the terminal unit should be maintained at its setpoint $F_{vav,set}$ within the tolerance $\varepsilon_{Fvav} F_{vav,set}$ for a percentage of data samples given by the user-defined threshold Pct_{Fvav}. Default values recommended for the two thresholds are ε_{Fvav} = 0.1 and Pct_{Fvav} = 90%, and only samples in the last 15 minutes of the test segment are considered. Faults that are possible explanations for Rule 1 being triggered are:

- Control parameter configuration fault (e.g., incorrect tuning parameters causing unstable control)
- Stuck damper
- Airflow sensor error
- Airflow obstruction (e.g., blockage in the ductwork or terminal unit)
- Unbalanced airflow distribution network
- Damper range limited (i.e., damper restricted from opening and/or closing fully)
- Damper actuator direction reversal (i.e., damper rotates opposite the intended direction resulting in the damper being driven closed rather than open)
- Undersized terminal unit

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rule 2 states that the reheat valve should be commanded closed in the cooling mode. The default threshold for Rule 2 is $\varepsilon_{rh,max}$ = 0.02. Faults that are possible explanations for Rule 2 being triggered are:

- Control parameter configuration fault
- Control logic fault

Rule 3 states that the AHU supply air temperature T_{sa} and the terminal unit discharge air temperature $T_{vav,da}$ should be nearly the same in the cooling mode. The default threshold for Rule 3 is $\varepsilon_{\Delta T,max}$ = 5°F. Faults that are possible explanations for Rule 3 being triggered are:

- Control parameter configuration fault
- Control logic fault
- Leaking reheat valve
- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Valve actuator direction reversal (i.e., valve rotates opposite the intended direction resulting in the valve being driven fully open rather than fully closed)

Note that because the zone temperature setpoints for the cooling mode are set significantly below the actual zone temperature, it is anticipated that the terminal unit will operate at maximum cooling capacity throughout the test. In this case, another rule can be added to the analysis to check that the airflow setpoint $F_{vav,set}$ is equal to the maximum airflow setpoint in the cooling mode $F_{vav,clg,max}$. This rule could be triggered if: 1) a control parameter configuration fault is present (e.g., $F_{vav,clg,max}$ is configured with the wrong value); 2) a control logic fault is present (e.g., $F_{vav,set}$ is set to a parameter other than $F_{vav,clg,max}$); or 3) the room has cooled sufficiently during the test to cause the airflow setpoint to be reset away from its maximum value. Thus, if this additional rule is included in the analysis, care must be exercised to ensure the terminal unit operates with maximum capacity throughout the test. To avoid false alarms that will arise if this condition is not satisfied, the rule is not included here.

5.3.5.2.1.2 Deadband Mode Rules

Rules that are applicable to operation in the deadband mode are described below.

$$\text{Rule 4: } |F_{vav,set} - F_{vav,min}| \le \varepsilon_{Fset} F_{vav,min} \quad (8)$$

$$\text{Rule 5: } |F_{vav} - F_{vav,set}| \le \varepsilon_{Fvav} F_{vav,set} \quad \text{for at least } Pct_{Fvav} \text{ of samples} \quad (9)$$

$$\text{Rule 6: } u_{vav,rh} \le \varepsilon_{rh,max} \quad (10)$$

$$\text{Rule 7: } |T_{sa} - T_{vav,da}| \le \varepsilon_{\Delta T,max} \quad (11)$$

Rule 4 states that in the deadband mode, the airflow setpoint for the terminal unit $F_{vav,set}$ should equal the minimum setpoint value $F_{vav,min}$ within the tolerance defined by $\varepsilon_{Fset} F_{vav,min}$. The default value of ε_{Fset} is 0.005. Faults that are possible explanations for Rule 4 being triggered are:

- Control parameter configuration fault (e.g., $F_{vav,min}$ is configured with the wrong value)
- Control logic fault (e.g., $F_{vav,set}$ is set to a parameter other than $F_{vav,min}$)

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rule 5 is identical to Rule 1 described for the cooling mode; however, because the airflow setpoint is equal to its minimum value in the deadband mode and its maximum value in the cooling mode, the possible explanations for Rule 5 being triggered are different from those for Rule 1. Specifically, the undersized terminal unit listed as a possible explanation for Rule 1 being triggered is removed for Rule 5. Thus, the possible explanations for Rule 5 being triggered are:

- Control parameter configuration fault (e.g., incorrect tuning parameters causing unstable control)
- Stuck damper
- Airflow sensor error
- Airflow obstruction (e.g., blockage in the ductwork or terminal unit)
- Unbalanced airflow distribution network
- Damper range limited (i.e., damper restricted from opening and/or closing fully)
- Damper actuator direction reversal (i.e., damper rotates opposite the intended direction)

Rules 6 and 7 are identical to Rules 2 and 3 described for the cooling mode and, therefore, a description of the rules is not repeated here. Possible explanations for Rules 6 and 7 being triggered are also the same as those for Rules 2 and 3.

5.3.5.2.1.3 Heating Mode Rules

Rules that are applicable to operation in the heating mode are described below.

$$\text{Rule 8: } |F_{vav,set} - F_{vav,min}| \leq \varepsilon_{Fset} F_{vav,min} \quad (12)$$

$$\text{Rule 9: } |F_{vav} - F_{vav,set}| \leq \varepsilon_{Fvav} F_{vav,set} \quad \text{for at least } Pct_{Fvav} \text{ of samples} \quad (13)$$

$$\text{Rule 10: } u_{vav,rh} > \varepsilon_{rh,min} \quad (14)$$

$$\text{Rule 11: } T_{vav,da} - T_{sa} > \varepsilon_{\Delta T,min} \quad (15)$$

$$\text{Rule 12: } T_{vav,da} \leq T_{vav,da,max} \quad (16)$$

Rule 8 is identical to Rule 4. Thus, no further explanation of Rule 8 is provided.[13] Rule 9 is identical to Rules 1 and 5, and the possible explanations for Rule 9 being triggered are identical to those for Rule 5.

Rule 10 states that in the heating mode, the control signal to the reheat valve should be greater than the threshold $\varepsilon_{rh,min}$ = 0.10. Faults that are possible explanations for Rule 10 being triggered are:

- Control parameter configuration fault (e.g., sign of the proportional gain of the proportional-integral controller is incorrect)

[13] When the dual-maximum control sequence is implemented, $F_{vav,min}$ should be replaced by $F_{vav,htg,max}$. Thus, the rule becomes: $|F_{vav,set} - F_{vav,htg,max}| \leq \varepsilon_{Fset} F_{vav,htg,max}$

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rule 11 states that the temperature rise across the reheat coil ($T_{vav,da}$ - T_{sa}) in the heating mode should exceed the minimum threshold defined by $\varepsilon_{\Delta T,min}$. The default value of the threshold is $\varepsilon_{\Delta T,min}$ = 15°F. Faults that are possible explanations for Rule 11 being triggered are:

- Control logic fault
- Stuck reheat valve
- Discharge air temperature sensor fault
- AHU supply air temperature sensor fault
- Valve actuator direction reversal (i.e., valve is actually driven closed rather than open)

Rule 12 states that the discharge air temperature must not exceed a maximum threshold given by $T_{vav,da,max}$. The purpose of this rule is to avoid excessive discharge air temperatures that result in stratification and short circuiting of warm supply air directly into the return grill. Faults that are possible explanations for Rule 12 being triggered are:

- Stuck damper
- Airflow sensor error
- Airflow obstruction (e.g., blockage in the ductwork or terminal unit)
- Stuck reheat valve
- Discharge air temperature sensor fault

Pseudocode implementing the rule set for the cooling, deadband, and heating modes follows.

```
# For each test segment, evaluate the rules associated with the mode
#   for that segment.
# 1-minute sampling is assumed.
# Rules 1, 5, and 9, are evaluated using the data from each of the
#   last 15 sample times in a segment. All other rules are evaluated
#   using measurement averages for the last 15 samples of the
segment.
# Set user-defined thresholds
      # εFvav = 0.1    # flow threshold for Rules 1, 5, and 9
      # PctFvav = 90%  # sample threshold for Rules 1, 5, and 9
      # εrh,max = 0.02  # valve control signal threshold for Rules 2 and
6
      # εΔT,max = 5°F   # temperature rise threshold for Rules 3 and 7
      # εFset = 0.005  # airflow setpoint threshold for Rules 4 and 8
      # εrh,min = 0.1   # valve control signal threshold for Rule 10
      # εΔT,min = 15°F  # temperature rise threshold for Rule 11
      # Tvav,da,max = 100°F  # maximum allowable discharge air temperature
# Initialize all rules to be satisfied (i.e., not triggered)
      # Rule_i = "True" for i = 1-12
# Process test segment data
      # testSeg = 1
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# trigRules = []    # Initialize array of triggered rules
# notTrigRules = [] # Initialize array of non-triggered rules
# while testSeg < 4:
      # Initialize counter
      # count = 0
      # i = 1      # sample iterator
      # imax = 30  # example with 30 minute test segment
      # while i < (imax + 1):
            # if i > 15 & |F_vav - F_vav,set| ≤ ε_Fvav F_vav,set: # use last
                                                          # 15 samples
                  # count += 1
            # i += 1
      # Calculate and store segment average for u_vav,rh, T_sa, T_vav,da,
      #   F_vav, and F_vav,set based on final 15 segment samples
      #
      # Evaluate segment rules
      # **** Cooling Mode ****
      if testSeg == 1 & T_za > T_za,clg,set > T_za,htg,set
            # if count/15*100 ≥ Pct_Fvav:
                  # notTrigRules = notTrigRules + Rule_1
            # else:
                  # Rule_1 = "False"
                  # trigRules = trigRules + Rule_1
            # if u_vav,rh ≤ ε_rh,max:
                  # notTrigRules = notTrigRules + Rule_2
            # else:
                  # Rule_2 = "False"
                  # trigRules = trigRules + Rule_2
            # if |T_sa - T_vav,da| ≤ ε_ΔT,max:
                  # notTrigRules = notTrigRules + Rule_3
            # else:
                  # Rule_3 = "False"
                  # trigRules = trigRules + Rule_3
      # **** Deadband Mode ****
      # elif testSeg == 2 & T_za,clg,set > T_za > T_za,htg,set
            # if |F_vav,set - F_vav,min| ≤ ε_Fset F_vav,min:
                  # notTrigRules = notTrigRules + Rule_4
            # else:
                  # Rule_4 = "False"
                  # trigRules = trigRules + Rule_4
            # if count/15*100 ≥ Pct_Fvav:
                  # notTrigRules = notTrigRules + Rule_5
            # else:
                  # Rule_5 = "False"
                  # trigRules = trigRules + Rule_5
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                # if u_vav,rh ≤ ε_rh,max:
                    # notTrigRules = notTrigRules + Rule_6
                # else:
                    # Rule_6 = "False"
                    # trigRules = trigRules + Rule_6
                # if |T_sa - T_vav,da| ≤ ε_ΔT,max:
                    # notTrigRules = notTrigRules + Rule_7
                # else:
                    # Rule_7 = "False"
                    # trigRules = trigRules + Rule_7
            # **** Heating Mode ****
            # elif testSeg == 3 & T_za,clg,set > T_za,htg,set > T_za
                # if |F_vav,set - F_vav,min| ≤ ε_Fset F_vav,min:
                    # notTrigRules = notTrigRules + Rule_8
                # else:
                    # Rule_8 = "False"
                    # trigRules = trigRules + Rule_8
                # if count/15*100 ≥ Pct_Fvav:
                    # notTrigRules = notTrigRules + Rule_9
                # else:
                    # Rule_9 = "False"
                    # trigRules = trigRules + Rule_9
                # if u_vav,rh > ε_rh,min:
                    # notTrigRules = notTrigRules + Rule_10
                # else:
                    # Rule_10 = "False"
                    # trigRules = trigRules + Rule_10
                # if T_vav,da - T_sa > ε_ΔT,min:
                    # notTrigRules = notTrigRules + Rule_11
                # else:
                    # Rule_11 = "False"
                    # trigRules = trigRules + Rule_11
                # if T_vav,da ≤ T_vav,da,max:
                    # notTrigRules = notTrigRules + Rule_12
                # else:
                    # Rule_12 = "False"
                    # trigRules = trigRules + Rule_12
            # testSeg += 1
        # if trigRules = []:
        #     functTestResult = "Pass"
        # else:
        #     functTestResult = "Fail" # Perform multi-mode analysis or
                                       #   inverse multi-mode analysis
                                       #   for root cause fault
diagnosis
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.3.5.2.2 Root Cause Diagnostics

Two processes for analyzing the results of failed tests and performing root cause diagnostics have been developed and are shown in the process flow diagram in Figure 54. The first is referred to as the multi-mode analysis and involves combining fault symptoms revealed through triggered rules in different modes with inferences that can be made based on rules that are not triggered in order to find the root cause fault. Thus, symptoms are used to identify the fault. This multi-mode analysis is described in Lin et al. 2025.

The second process is referred to as the inverse multi-mode analysis and involves identifying common faults for the system of interest and listing the symptoms that are expected for each fault for the different modes of operation. Thus, in this case the fault is used to identify the expected symptoms. This approach is proposed for VAV terminal units because of the relative simplicity of the equipment and controls.

5.3.5.2.2.1 Multi-Mode Analysis

The description of the rule set in Section 5.3.5.2.1 includes rules that apply to one or more modes of operation as well as one or more condition-based faults that are possible explanations why each rule might be triggered. In all, 12 rules were established. The possible faults that could explain why each rule might be triggered are listed in Table 38. Fourteen (14) possible faults are identified.

Table 38: Condition-based fault explanations for triggered rules.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		Explanations for Rule Violations													
Mode	**Rule**	1 - Airflow Sensor Error	2 - Stuck Damper	3 - Damper Range Limited	4 - Damper Actuator Direction Reversal	5 - Airflow Obstruction	6 - Unbalanced Airflow Distribution Network	7 - Undersized Terminal Unit	8 - Leaking Valve	9 - Stuck Valve	10 - Discharge Air Temperature Sensor Fault	11 - AHU Supply Air Temperature Sensor Fault	12 - Valve Actuator Direction Reversal	13 - Control Parameter Configuration Error	14 - Control Logic Error
Cooling	Rule 1: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	X	X	X	X	X	X	X						X	
	Rule 2: $u_{vav,rh} \le \varepsilon_{rh,max}$													X	X
	Rule 3: $\lvert T_{sa} - T_{vav,da} \rvert \le \varepsilon_{\Delta T,max}$								X	X	X	X	X	X	X
Deadband	Rule 4: $\lvert F_{vav,set} - F_{vav,min} \rvert \le \varepsilon_{Fset} F_{vav,min}$													X	X
	Rule 5: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	X	X	X	X	X	X							X	
	Rule 6: $u_{vav,rh} \le \varepsilon_{rh,max}$													X	X
	Rule 7: $\lvert T_{sa} - T_{vav,da} \rvert \le \varepsilon_{\Delta T,max}$								X	X	X	X	X	X	X
Heating	Rule 8: $\lvert F_{vav,set} - F_{vav,min} \rvert \le \varepsilon_{Fset} F_{vav,min}$													X	X
	Rule 9: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	X	X	X	X	X	X							X	
	Rule 10: $u_{vav,rh} > \varepsilon_{rh,min}$													X	
	Rule 11: $T_{vav,da} - T_{sa} > \varepsilon_{\Delta T,min}$									X	X	X	X		X
	Rule 12: $T_{vav,da} \le T_{vav,da,max}$	X	X			X				X	X				

The multi-mode analysis provides a process for combining the results of the various test segments to determine the smallest possible set of condition-based faults that are possible explanations for a failed functional test. If it is assumed that no more than one fault is present during a functional test and if multiple rules are triggered during the test, then the triggered rules must share at least one condition-based fault among their possible fault explanations. Stated another way, each triggered rule has a set of condition-based faults as possible explanations and the intersection of these sets should contain the actual condition-based fault. The first step in the multi-mode analysis is to identify the shared condition-based faults for the triggered rules. This step is labeled MMA1 in Figure 54.

Table 38 lists 14 condition-based faults that are possible explanations for one or more triggered rules. To find the intersection of the set of possible explanations for each triggered rule, a 1x14 array *NcauseTrig* is defined and initialized with all elements equal to 0. The second dimension of this array corresponds to the total number of condition-based faults identified in Table 38. This array will be used to count the number of times each condition-based fault is identified as a possible explanation in the triggered rules. Next, a 1x14 array *cause_rule* (where *rule* = 1 to 12) is defined for each rule, where an element of the array is assigned the value 1 if the

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

condition-based fault corresponding to that element is a possible cause for a particular rule being triggered, and the element is assigned the value 0 otherwise. As an example, referring to Table 38, Rule 1 has eight condition-based faults as possible explanations for why this rule might be triggered. The eight condition-based faults are:

1 - Airflow sensor error
2 - Stuck damper
3 - Damper range limited
4 - Damper actuator direction reversal
5 - Airflow obstruction
6 - Unbalanced airflow distribution network
7 - Undersized terminal unit
13 - Control parameter configuration error

Thus, the array *cause_rule* where *rule* = 1 (i.e., *cause_1*) is defined as:

```
cause_1 = [1,1,1,1,1,1,1,0,0,0,0,0,1,0]
```

where we note that elements 1-7 and element 13 are set to 1. Arrays *cause_2* to *cause_12* are defined in the same manner.

Next, for each rule in the array *trigRules*, the array *cause_rule* is added to the array *NcauseTrig*. This step counts the number of times each condition-based fault is identified as a possible explanation for a triggered rule. When this step has been completed, there should be one or more elements in *NcauseTrig* that are equal to the number of rules in the array *trigRules*. These elements correspond to the condition-based faults that are the possible causes for the failed functional test.

Pseudocode implementing step MMA1 follows:

```
# Initialize 1x14 array for counting possible causes for triggered
#   rules
      #     NcauseTrig = [0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x14 array of possible causes for each of the 12 rules
      #     cause_1 = [1,1,1,1,1,1,1,0,0,0,0,0,1,0]
      #     cause_2 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1]
      #     cause_3 = [0,0,0,0,0,0,0,1,1,1,1,1,1,1]
      #     cause_4 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1]
      #     cause_5 = [1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_6 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1]
      #     cause_7 = [0,0,0,0,0,0,0,1,1,1,1,1,1,1]
      #     cause_8 = [0,0,0,0,0,0,0,0,0,0,0,0,1,1]
      #     cause_9 = [1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_10 = [0,0,0,0,0,0,0,0,0,0,0,0,1,0]
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      #     cause_11 = [0,0,0,0,0,0,0,0,1,1,1,1,0,1]
      #     cause_12 = [1,1,0,0,1,0,0,0,1,1,0,0,0,0]
      #     i = 1
      #     numRules = 12
# Count the possible causes for triggered rules in trigRules
      #     While i <= numRules:
            #     if Rule_i in trigRules:
                  #     NcauseTrig = NcauseTrig + cause_i
            #     i += 1
      #     goto "MMA2"
```

Next, rules that are evaluated and not triggered are used to identify faults that can be excluded as possible causes for a failed functional test. This step is labeled MMA2 in Figure 54. If, for example, Rule 1, Rule 5, or Rule 9 are satisfied (not triggered), this is sufficient evidence to infer that the damper is not stuck and that the damper is rotating in the appropriate direction when its control signal is changed. The rules that can be used in this way to exclude possible causes of a failed test are listed in Table 39 along with the condition-based fault or faults that can be removed in each case. Note that while Rules 3 and 7 are the same (only applied in different modes), if Rule 3 is evaluated and not triggered, the leaking valve fault is not removed; however, if Rule 7 is evaluated and not triggered, the leaking valve fault is removed. The reason for this difference is that we expect the airflow rate in the cooling mode to be higher and the effect of a leaking valve fault to be smaller in this mode than it would be in the deadband mode. Thus, satisfying Rule 3 does not necessarily mean that the reheat valve is not leaking.

Table 39: Condition-based faults that can be removed as possible causes of a failed functional test based on rules that are evaluated in the test and not triggered.

Rule Evaluated and not Triggered	Condition-based Fault(s) Removed *
Rule 1: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	Stuck damper [2] Damper actuator direction reversal [4] Airflow obstruction [5] Unbalanced airflow distribution network [6] Undersized terminal unit [7]
Rule 3: $\lvert T_{sa} - T_{vav,da} \rvert \le \varepsilon_{\Delta T,max}$	Discharge air temperature sensor fault [10] AHU supply air temperature sensor fault [11] Valve actuator direction reversal [12]
Rule 5: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	Stuck damper [2] Damper actuator direction reversal [4]
Rule 7: $\lvert T_{sa} - T_{vav,da} \rvert \le \varepsilon_{\Delta T,max}$	Leaking valve [8] Discharge air temperature sensor fault [10] AHU supply air temperature sensor fault [11] Valve actuator direction reversal [12]
Rule 9: $\lvert F_{vav} - F_{vav,set} \rvert \le \varepsilon_{Fvav} F_{vav,set}$	Stuck damper [2] Damper actuator direction reversal [4]

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Rule 11: $T_{vav,da} - T_{sa} > \varepsilon_{\Delta T,min}$	Valve actuator direction reversal [12]

* The number in brackets in the last column corresponds to the explanation number for the condition-based fault from Table 38.

To implement this step, a 1x14 array *excludeCause* is defined and initialized with all elements equal to 0. Next, a 1x14 array *exclude_rule* is defined for each fault exclusion rule in Table 39. An element of the array is assigned the value 1 if the condition-based fault corresponding to that element can be removed as a possible cause for a failed functional test, and the element is assigned the value 0 otherwise. As an example, the array *exclude_1* (i.e., *rule* is set equal to 1 corresponding to Rule 1) is defined as

```
exclude_1 = [0,1,0,1,1,1,1,0,0,0,0,0,0,0]
```

because, as shown in Table 39, when Rule 1 is evaluated and not triggered, the stuck damper fault (explanation #2 in Table 38), damper actuator direction reversal fault (explanation #4 in Table 38), airflow obstruction fault (explanation #5 in Table 38), unbalanced airflow distribution network fault (explanation #6 in Table 38), and undersized terminal unit fault (explanation #7 in Table 38) can be removed as possible causes for the failed functional test. Thus, elements 2, 4, 5, 6, and 7 of the array exclude_1 are set to 1. Similar arrays are defined for the remaining fault exclusions in Table 39. Next, for each rule in the array *notTrigRules*, the array *exclude_rule* is added to the array *excludeCause*.

When step MMA2 has been completed, the elements of array *excludeCause* that are greater than or equal to 1 will indicate the condition-based faults that can be removed as a possible cause for the failed functional test.

Pseudocode implementing step MMA2 follows:

```
# Initialize 1x14 array for identifying excluded causes based on
rules
#   that are evaluated and not triggered
     # excludeCause = [0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Initialize a 1x14 array of excluded causes for Rules 1-12
     # for i = 1:12:
          # exclude_i = [0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x14 array of excluded causes for rules in Table 39
     # Rule 1 evaluated and not triggered
     #    exclude_1 = [0,1,0,1,1,1,1,0,0,0,0,0,0,0]
     # Rule 3 evaluated and not triggered
     #    exclude_3 = [0,0,0,0,0,0,0,0,0,1,1,1,0,0]
     # Rule 5 evaluated and not triggered
     #    exclude_5 = [0,1,0,1,0,0,0,0,0,0,0,0,0,0]
     # Rule 7 evaluated and not triggered
     #    exclude_7 = [0,0,0,0,0,0,0,1,0,1,1,1,0,0]
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      # Rule 9 evaluated and not triggered
      #     exclude_9 = [0,1,0,1,0,0,0,0,0,0,0,0,0,0]
      # Rule 11 evaluated and not triggered
      #     exclude_11 = [0,0,0,0,0,0,0,0,0,0,0,1,0,0]
# Define loop variables
      #     i = 1
      #     numRules = 12
# Update array excludeCause using rules in array notTrigRules
      #     While i <= numRules:
            #     if Rule_i in notTrigRules:
                  #     excludeCause = excludeCause + exclude_i
            #     i += 1
      #     goto "MMA3"
```

Next, in step MMA3, results from the previous two steps are combined to determine the smallest set of condition-based faults that are possible explanations for the failed functional test. More specifically, this procedure removes any faults identified in step MMA2 from the shared condition-based faults identified in step MMA1.

A 1x14 array *Ncause* is determined by subtracting the array *excludeCause* (step MMA2) from the array *NcauseTrig* (step MMA1). The element(s) of *Ncause* with the largest magnitude identifies the condition-based fault(s) that is the cause of the failed functional test. Finally, an array *NcauseFinal* is initialized and used to store the largest element(s) of array *Ncause*. *NcauseFinal* will contain the smallest set of condition-based faults that are possible explanations for a failed functional test.

Pseudocode implementing step MMA3 follows:

```
# Determine the array of possible causes for the failed
#   functional test based on triggered rules and excluded causes
      #     Ncause = NcauseTrig - excludeCause
# Initialize array NcauseFinal, then store largest element(s) of
#   Ncause in this array
      # NcauseFinal = max(Ncause)
```

5.3.5.2.2.2 Inverse Multi-Mode Analysis

The inverse multi-mode analysis starts by identifying common faults for the system of interest, and then identifies symptoms that are anticipated to be present based on the tests performed. The fault list for closed-loop testing consists of the following:

- Stuck damper
- Leaking reheat coil valve
- Stuck open reheat coil valve
- Stuck closed reheat coil valve
- Reheat valve actuator reversal fault

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- Discharge air temperature or AHU supply air temperature sensor bias fault

The symptoms for each of these faults are listed and explained below. In each case it is assumed that only a single fault is present at any time. The symptoms of damper faults are based on airflow measurements in different operating modes and those for reheat valve and temperature sensor faults are based on temperature measurements.

Stuck Damper

The symptoms of a stuck damper fault are that the control signal to the damper will be saturated in either the fully open or fully closed position and the airflow setpoint will not be satisfied. Both of these symptoms will be observed in all three modes of operation. Furthermore, the airflow rate in the different modes will be approximately the same provided the inlet pressure to the terminal unit is constant. This latter requirement should be satisfied if a fixed static pressure setpoint is used during testing. The symptoms are written in equation form below.

Symptoms that apply to all modes:

$$\text{S1i: } 1 - u_{vav,dm} \le \varepsilon_{dm} \text{ and } F_{vav,set} - F_{vav} > \varepsilon_{Fvav} F_{vav,set} \text{ for at least } Pct_{S1,S2} \text{ of samples} \quad (16)$$

or

$$\text{S2i: } u_{vav,dm} \le \varepsilon_{dm} \text{ and } F_{vav} - F_{vav,set} > \varepsilon_{Fvav} F_{vav,set} \text{ for at least } Pct_{S1,S2} \text{ of samples} \quad (17)$$

where i (i = 1 to 3) corresponds to the operating mode, with i = 1 for the cooling mode, i = 2 for the deadband mode, and i = 3 for the heating mode. Symptom S1i indicates that the damper is saturated at the fully open position ($u_{vav,dm,i} \approx 1$) within the threshold ε_{dm} ($\varepsilon_{dm} = 0.02$ is recommended as the default value), and the airflow rate $F_{vav,i}$ is significantly less than the setpoint $F_{vav,set}$. S2i is similar and indicates the damper is saturated at the fully closed position ($u_{vav,dm,i} \approx 0$) and the airflow rate is significantly higher than the setpoint. When evaluating S1i and S2i, it is recommended that data for the final 15 minutes of each operating mode be used and that each condition for a given symptom be required to be satisfied a minimum of $PCT_{S1,S2}$ percent (e.g., 100%, adjustable) of those final 15 samples.

Symptoms that compare operation between modes:

$$\text{S3: } | F_{vav,1} - F_{vav,2} | \le \varepsilon_{\Delta F} F_{vav,clg,max} \quad (18)$$

$$\text{S4: } | F_{vav,1} - F_{vav,3} | \le \varepsilon_{\Delta F} F_{vav,clg,max} \quad (19)$$

Symptom S3 compares the flow rates between the cooling and deadband modes, and S4 compares the flow rates between the cooling and heating modes. In both S3 and S4, a threshold defined by $\varepsilon_{\Delta F} F_{vav,clg,max}$ is used and a default value of $\varepsilon_{\Delta F} = 0.1$ is recommended. S3 and S4 can be evaluated using the average values of $F_{vav,i}$ (i = 1 to 3) over the final 15 minutes of each operating mode.

Pseudocode for implementing the stuck open damper analysis follows.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# 1-minute sampling is assumed
# Set user-defined thresholds for stuck damper symptoms and set
#   design parameters
     # ε_dm = 0.02    # damper control signal threshold parameter
     # ε_Fvav = 0.1    # airflow threshold parameter
     # ε_ΔF = 0.1     # airflow difference threshold parameter
     # Pct_S1,S2 = 100% # sample threshold for symptoms S1i and S2i
     # F_vav,clg,max = design maximum airflow rate in the cooling mode
# Initialize all symptoms to be false (i.e., not triggered/satisfied)
     # S1i = "False" for i = 1-3
     # S2i = "False" for i = 1-3
     # S3 = "False"
     # S4 = "False"
# Process test segment data
     # testSeg = 1
     # while testSeg < 4: # symptoms evaluated using data for
                          #   test segment "testSeg"
          # Initialize counters
          # countS1 = 0
          # countS2 = 0
          # j = 1      # sample iterator
          # jmax = 30  # example with 30 minute test segment
          # while j < (jmax + 1):
               # if j > 15 & 1 - u_vav,dm,j ≤ ε_dm
               #           & F_vav,set - F_vav,j > ε_Fvav F_vav,set:  # S1i
                    # countS1 += 1
               # if j > 15 & u_vav,dm,j ≤ ε_dm
               #           & F_vav,j - F_vav,set > ε_Fvav F_vav,set:  # S2i
                    # countS2 += 1
               # j += 1
          # If countS1/15*100 ≥ Pct_S1:
               # S1i = "True" (i = testSeg)
          # If countS2/15*100 ≥ Pct_S2:
               # S2i = "True" (i = testSeg)
          # testSeg += 1
     # Evaluate S3 and S4 using averages based on final 15 samples
     #   in each operating mode (test segment)
     # if | F_vav,1 - F_vav,2 | ≤ ε_ΔF F_vav,clg,max:
          # S3 = "True"
     # if | F_vav,1 - F_vav,3 | ≤ ε_ΔF F_vav,clg,max:
          # S4 = "True"
     # Check all symptoms
     # If (S11 == "True" or S21 == "True") &
     #    (S12 == "True" or S22 == "True") &
     #    (S13 == "True" or S23 == "True") &
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#     S3 == "True" & S4 == "True":
          # StkDamper = "True"
# Else:
          # StkDamper = "False"
```

Leaking Reheat Valve

Three symptoms are used to identify a leaking reheat valve fault based on the three modes of operation of the terminal unit. First, this fault will result in an excessive temperature increase from the AHU supply air temperature to the discharge air temperature leaving the terminal unit (referred to as the temperature rise across the coil) in the deadband mode. Second, the temperature rise across the coil will be larger in the deadband mode than in the cooling mode because of the higher airflow rate in the cooling mode. Finally, the temperature rise in the heating mode will be greater than in the deadband mode because the valve will likely be fully open in the heating mode. In equation form, the symptoms are the following:

$$S5: \Delta T_2 > \varepsilon_{LV1} \quad (20)$$

$$S6: \Delta T_2 - \Delta T_1 > \varepsilon_{LV2} \quad (21)$$

$$S7: \Delta T_3 - \Delta T_2 > \varepsilon_{LV3} \quad (22)$$

where $\Delta T_i = T_{vav,da} - T_{sa}$ and the subscript i = 1 corresponds to the cooling mode, i = 2 the deadband mode, and i = 3 the heating mode. Thresholds ε_{LV1} and ε_{LV3} have default values of 5°F and threshold ε_{LV2} has a default value of 2°F. S5 to S7 can be evaluated using average values of $T_{vav,da}$ and T_{sa} calculated over the final 15 minutes of each operating mode.

Pseudocode for implementing the leaking valve analysis follows.

```
# Set user-defined thresholds for leaking valve symptoms
      # ε_LV1 = 5°F
      # ε_LV2 = 2°F
      # ε_LV3 = 5°F
# Calculate ΔT_i using segment (mode) averages for T_sa and T_vav,da
#   based on final 15 segment samples
      # ΔT_i = T_vav,da,i - T_sa,i  for i = 1 to 3 (i = segment or mode)
# Check all symptoms
      # If ΔT_2 > ε_LV1 & ΔT_2 - ΔT_1 > ε_LV2 and ΔT_3 - ΔT_2 > ε_LV3:
            # LkgVlv = "True"
      # Else:
            # LkgVlv = "False"
```

Stuck Open Reheat Valve

The symptoms of a stuck open reheat valve fault are similar to those of a leaking reheat valve fault. There will be an excessive temperature rise across the coil in the deadband mode. In addition, the temperature rise across the coil will be larger in the deadband mode than in the cooling mode because of the higher airflow rate in the cooling mode. Finally, and here the

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

symptom is different than for a leaking valve fault, the temperature rise across the reheat coil in the heating mode is expected to be approximately the same as the temperature rise in the deadband mode. In equation form, the symptoms of the fault are the following:

$$\text{S8: } \Delta T_2 > \varepsilon_{SOV1} \quad (23)$$
$$\text{S9: } \Delta T_2 - \Delta T_1 > \varepsilon_{SOV2} \quad (24)$$
$$\text{S10: } | \Delta T_3 - \Delta T_2 | < \varepsilon_{SOV3} \quad (25)$$

where threshold ε_{SOV1} has a default value of 5°F, and thresholds ε_{SOV2} and ε_{SOV3} have a default value of 2°F. S8 to S10 can be evaluated using average values of $T_{vav,da}$ and T_{sa} calculated over the final 15 minutes of each operating mode.

Pseudocode for implementing the stuck open reheat valve analysis follows.

```
# Set user-defined thresholds for leaking valve symptoms
      # ε_SOV1 = 5°F
      # ε_SOV2 = 2°F
      # ε_SOV3 = 2°F
# Calculate ΔT_i using segment (mode) averages for T_sa and T_vav,da
#   based on final 15 segment samples
      # ΔT_i = T_vav,da,i - T_sa,i  for i = 1 to 3 (i = segment or mode)
# Check all symptoms
      # If ΔT_2 > ε_SOV1 & ΔT_2 - ΔT_1 > ε_SOV2 and |ΔT_3 - ΔT_2| < ε_SOV3:
            # StkOpenVlv = "True"
      # Else:
            # StkOpenVlv = "False"
```

Stuck Closed Reheat Valve

A stuck closed reheat valve fault is expected to have a temperature rise across the reheat coil that is approximately equal to zero in all modes. In equation form, the symptoms of the fault can be expressed as:

$$\text{S11: } \Delta T_2 < \varepsilon_{SCV1} \quad (23)$$
$$\text{S12: } | \Delta T_2 - \Delta T_1 | < \varepsilon_{SCV2} \quad (24)$$
$$\text{S13: } | \Delta T_3 - \Delta T_2 | < \varepsilon_{SCV3} \quad (25)$$

where symptom S11 states that the temperature rise across the reheat coil in the deadband mode should be smaller than the threshold ε_{SCV1}, symptom S12 states that the difference in the temperature rise across the reheat coil in the deadband and cooling modes should be less than the threshold ε_{SCV2}, and symptom S13 states that the difference in the temperature rise across the reheat coil in the heating and deadband modes should be less than the threshold ε_{SCV3}. Threshold ε_{SCV1} has a default of 5°F and thresholds ε_{SCV2} and ε_{SCV3} have default values of 2°F.

Pseudocode for implementing the stuck closed reheat valve analysis follows.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# Set user-defined thresholds for leaking valve symptoms
      # ε_SCV1 = 5°F
      # ε_SCV2 = 2°F
      # ε_SCV3 = 2°F
# Calculate ΔT_i using segment (mode) averages for T_sa and T_vav,da
#   based on final 15 segment samples
      # ΔT_i = T_vav,da,i - T_sa,i  for i = 1 to 3 (i = segment or mode)
# Check all symptoms
      # If ΔT_2 < ε_SCV1 & |ΔT_2 - ΔT_1| < ε_SCV2 and |ΔT_3 - ΔT_2| < ε_SCV3:
            # StkClosedVlv = "True"
      # Else:
            # StkClosedVlv = "False"
```

Reheat Valve Actuator Reversal

If the reheat valve actuator direction is reversed, the valve will be closed when it is supposed to be open, and open when it is supposed to be closed. Thus, in the deadband mode, there will be a large temperature rise across the reheat coil because the valve will be open when it is supposed to be closed. This same behavior will be seen in the cooling mode, but the temperature rise will be smaller than in the deadband mode because the airflow rate is higher in the cooling mode. Finally, in the heating mode, the valve will be driven closed, so the temperature rise in the deadband mode should be much larger than in the heating mode. In equation form, the symptoms of a reheat valve actuator reversal fault are:

$$\text{S14: } \Delta T_2 > \varepsilon_{VAR1} \tag{26}$$
$$\text{S15: } \Delta T_2 - \Delta T_1 > \varepsilon_{VAR2} \tag{27}$$
$$\text{S16: } \Delta T_2 - \Delta T_3 > \varepsilon_{VAR3} \tag{28}$$

where the thresholds ε_{VAR1} and ε_{VAR3} have default values of 15°F, and ε_{VAR2} has a default value of 2°F.

Pseudocode for implementing the reheat valve actuator reversal analysis follows.

```
# Set user-defined thresholds for reheat valve actuator reversal
#   symptoms
      # ε_VAR1 = 15°F
      # ε_VAR3 = 2°F
      # ε_VAR3 = 15°F
# Calculate ΔT_i using segment (mode) averages for T_sa and T_vav,da
#   based on final 15 segment samples
      # ΔT_i = T_vav,da,i - T_sa,i  for i = 1 to 3 (i = segment or mode)
# Check all symptoms
      # If ΔT_2 > ε_VAR1 & ΔT_2 - ΔT_1 > ε_VAR2 and ΔT_2 - ΔT_3 > ε_VAR3:
            # VlvActRev = "True"
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
    # Else:
        # VlvActRev = ”False”
```

Discharge Air Temperature or AHU Supply Air Temperature Sensor Bias Fault

Sensor bias faults of the discharge air temperature sensor $T_{vav,da}$ and the AHU supply air temperature sensor T_{sa} are expected to have the same symptoms. In the deadband mode when the heating valve is closed, the values of $T_{vav,da}$ and T_{sa} are expected to be significantly different due to the bias fault. It is also expected that the temperature difference across the coil ($T_{vav,da}$ - T_{sa}) will be approximately the same in the cooling mode as in the deadband mode. Finally, in the heating mode, the temperature rise across the coil is expected to be very large compared to the deadband mode. In equation form, the symptoms of a discharge air temperature or AHU supply air temperature sensor fault are:

$$S17: |\Delta T_2| > \varepsilon_{TS1} \quad (26)$$

$$S18: |\Delta T_2 - \Delta T_1| < \varepsilon_{TS2} \quad (27)$$

$$S19: \Delta T_3 - \Delta T_2 > \varepsilon_{TS3} \quad (28)$$

where the threshold ε_{TS1} has a default value of 5°F, ε_{TS2} has a default of 2°F, and ε_{TS3} has a default value of 15°F.

Pseudocode for implementing the discharge air temperature or AHU supply air temperature sensor fault analysis follows.

```
# Set user-defined thresholds for temperature sensor fault symptoms
    # ε_TS1 = 5°F
    # ε_TS2 = 2°F
    # ε_TS3 = 15°F
# Calculate ΔT_i using segment (mode) averages for T_sa and T_vav,da
#   based on final 15 segment samples
    # ΔT_i = T_vav,da,i - T_sa,i  for i = 1 to 3 (i = segment or mode)
# Check all symptoms
    # If |ΔT_2| > ε_TS1 & |ΔT_2 - ΔT_1| < ε_TS2 and ΔT_3 - ΔT_2 > ε_TS3:
        # TempSensFault = ”True”
    # Else:
        # TempSensFault = ”False”
```

5.4 Closed-loop Functional Tests for AHU Supply Air Temperature and Outdoor Air Control

5.4.1 Overview

Closed-loop functional tests are used in commissioning (Cx) to determine if systems are performing as intended. In closed-loop tests, the output to all final control elements (e.g.,

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

actuators, variable frequency drives, etc.) is determined by the control algorithm running in the building automation system (BAS) using a measured process variable as input. That is, normal feedback control of the system or process is unchanged and any system protections implemented in the BAS remain in effect. Consequently, closed-loop tests are well suited to automated testing because they are unlikely to create operating conditions that can damage equipment.

The closed-loop tests described here involve a series of setpoint adjustments designed to force the system under test into different operating conditions. These are normal modes or conditions that the systems will operate under as building loads naturally change; however, by proactively modifying the setpoints, the system can be forced to transition between modes on time scales that are much shorter than could be achieved with natural load changes. This is important because faults can be masked during certain operating conditions and apparent in others. Thus, proactive closed-loop testing compresses the time needed to collect a rich data set representative of a broad range of normal operational conditions, improving the likelihood that a root cause diagnosis can be obtained.

This specification describes testing procedures and analyses that can be automated and used to verify that components and control strategies used for control and/or monitoring of an air-handling unit's (AHU's) supply air temperature and outdoor air intake are operating as expected. The functional tests are intended to be generally applicable with the ability to assess the performance of equipment, systems, and controls without the need for detailed knowledge of the control sequences. Implemented through setpoint changes, a functional test consists of a sequence of steps or segments in which different processes are used to control the supply air temperature using the existing closed-loop control strategies implemented in the AHU controller. For each test segment, relationships utilizing measured variables and control parameters are used to verify that operation is as expected.

5.4.2 System Description

The closed-loop test described in this section was developed for single duct variable-air-volume (VAV) AHUs. A schematic diagram showing the primary components of the AHU as well as common measurement and control points is shown in Figure 55.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

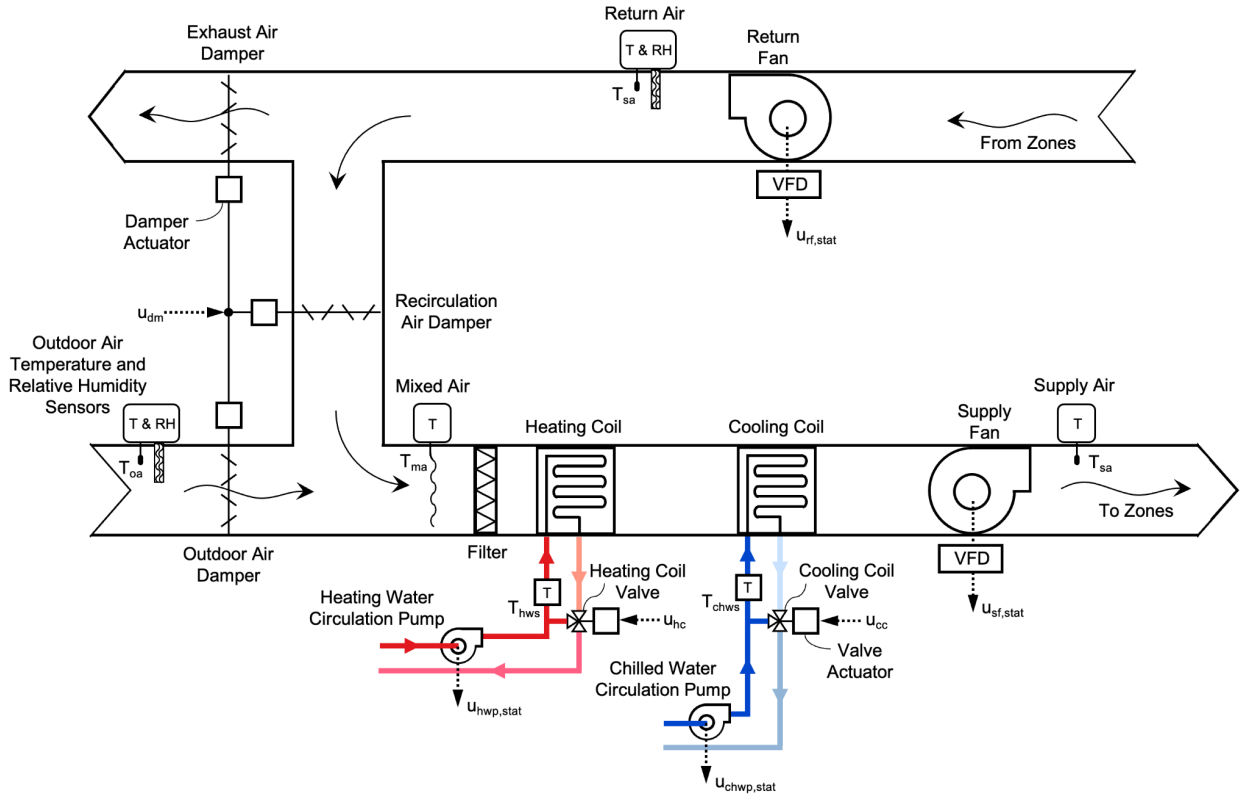


Figure 55: Schematic diagram of a single-zone VAV AHU.

Outdoor air enters the AHU and is mixed with air recirculated from the building. The mixed air is conditioned by the heating coil or cooling coil as needed to achieve the supply air temperature setpoint. The conditioned air is then distributed to the building. The supply fan is commonly operated to maintain the static pressure in the supply ductwork at a sufficiently high pressure to ensure the necessary quantity of air reaches each zone served by the AHU. The return fan draws air from the zones and is commonly controlled to maintain the building at a fixed positive pressure to prevent infiltration.

AHUs commonly have four processes that are used for supply air temperature control. The four processes are: 1) heating; 2) cooling with outdoor air; 3) mechanical cooling with maximum outdoor air; and 4) mechanical cooling with minimum outdoor air. The AHU sequencing strategy determines which of these processes is used to control the supply air temperature at any given time. A representative sequencing strategy is shown in Figure 56.

In operating Mode 1 (heating), the heating coil valve is controlled to maintain the supply air temperature at setpoint. This mode is normally used in cold outdoor conditions and the mixing box dampers are positioned to enable the minimum amount of outdoor air that will satisfy ventilation requirements. As the outdoor air temperature warms, the heating coil valve will

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

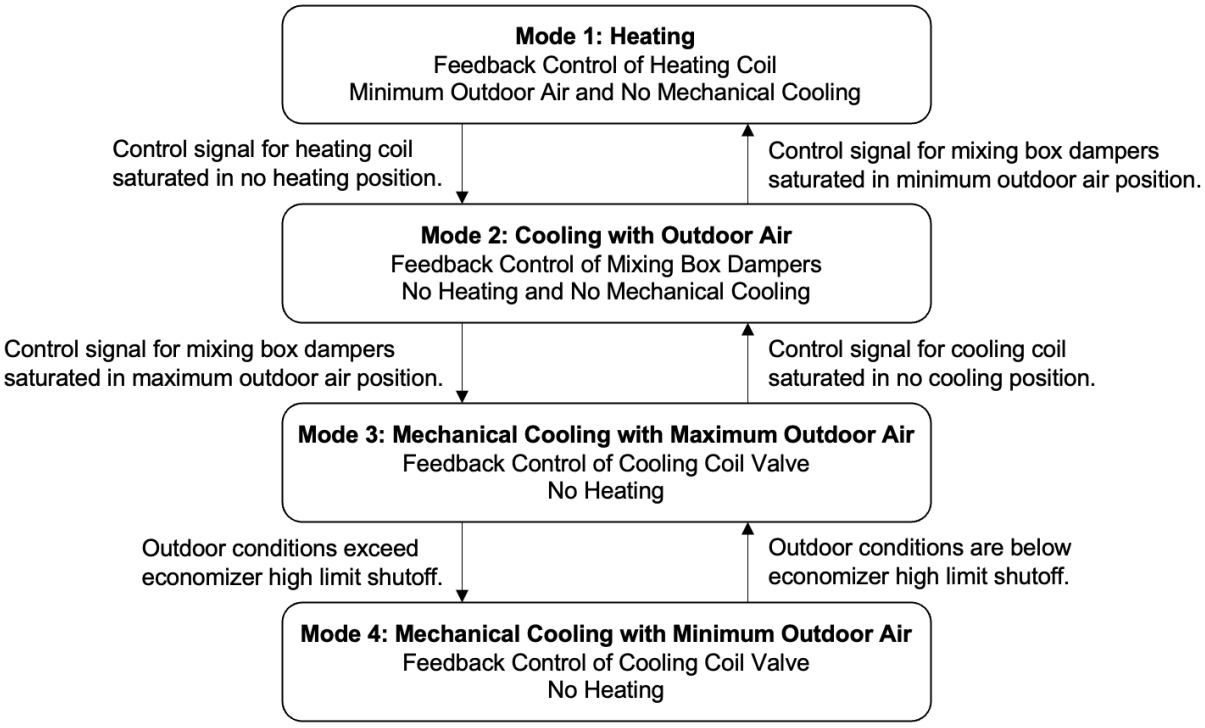


Figure 56: Representative sequencing logic for AHU supply air temperature control.

modulate toward the closed position. When the valve is completely closed, the AHU transitions to Mode 2.

In operating Mode 2 (cooling with outdoor air), the mixing box dampers are controlled to maintain the supply air temperature. The dampers modulate and adjust the fraction of outdoor air and return air in the mixed air stream. As the outdoor air temperature warms, the dampers modulate to allow a higher fraction of outdoor air in the mixed air stream. When the outdoor air damper is fully open, the AHU transitions to Mode 3. If the outdoor air temperature cools, the dampers will modulate to provide a smaller outdoor air fraction. When the outdoor air damper is at the minimum position, the AHU transitions to Mode 1.

In operating Mode 3 (mechanical cooling with maximum outdoor air), the cooling coil valve is controlled to maintain the supply air temperature at setpoint and the outdoor air dampers are positioned for maximum (or 100%) outdoor air. If the outdoor conditions exceed the economizer high limit shutoff, the AHU transitions to Mode 4. There are several common economizer high limit shutoff conditions. In Figure 56, a fixed dry-bulb temperature high limit shutoff is used. When the outdoor air dry-bulb temperature exceeds this condition, the economizer is disabled. Throughout the description of the closed-loop functional test that follows, economizer control is based on this dry-bulb temperature high limit shutoff criterion. As the outdoor air temperature cools, the cooling coil valve will modulate toward the closed position. When the valve is fully closed, the AHU transitions to Mode 2.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

In operating Mode 4 (mechanical cooling with minimum outdoor air), the cooling coil valve is controlled to maintain the supply air temperature at setpoint and the outdoor air dampers are positioned for minimum outdoor air. If the outdoor conditions drop below the economizer high limit shutoff, the AHU transitions to Mode 3.

5.4.3 Process description

Figure 57 shows a process flow diagram for the general testing and analysis procedure. *Part I* of the test is shown in green and represents closed-loop testing and the associated analysis. *Part II* is shown in yellow and represents additional closed-loop, open-loop or hybrid testing and the associated analysis. The *Part II* testing will only be carried out if the analysis of the closed-loop functional test in *Part I* results in more than one condition-based fault as the possible cause for the failed functional test. Condition-based faults are faults that can be characterized as having an improper or undesirable physical condition in a system or piece of equipment (Frank et al., 2019[14]), examples being stuck valves and excessive sensor bias. The five sections of the process flow diagram outlined with dashed lines are described next.

[14] Frank, S., Lin, G., Jin, X., Singla, R., Farthing, A., and Granderson, J. 2019. “A performance evaluation framework for building fault detection and diagnosis algorithms.” *Energy and Buildings*. 192:84-92. DOI: https://doi.org/10.1016/j.enbuild.2019.03.024

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

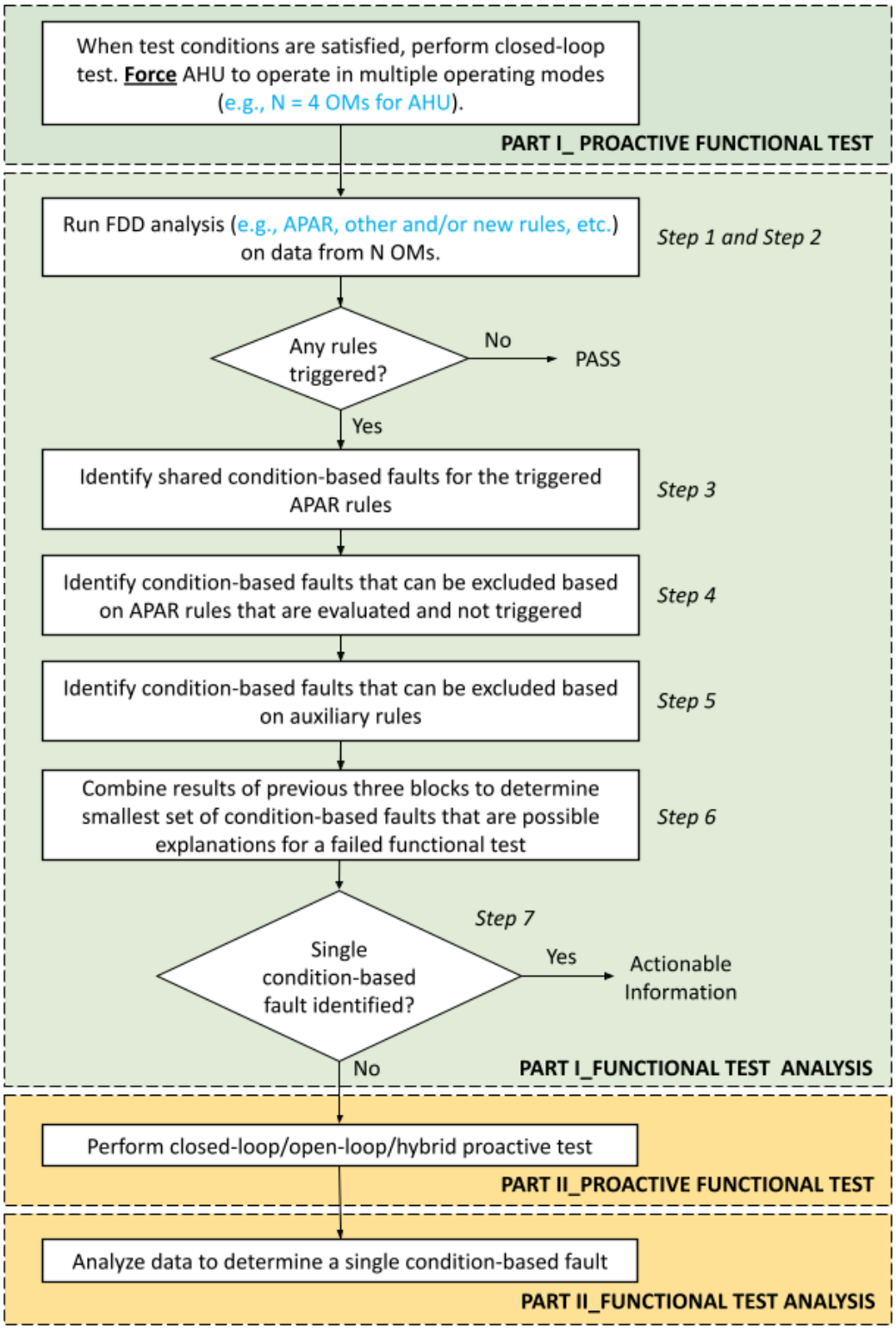


Figure 57: Process flow diagram for the AHU supply air temperature and outdoor air control functional test and analysis.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.4.4 Required Data Points

Measurement and control points from the building automation system (BAS) that are used in this test are listed in Table 40. All other variables and parameters used to implement the functional test and analyze its results are listed in Table 41.

Table 40: Required Data Points from BAS (controllers).

Abbreviation	Data point	Data type	Requirements (Read/Write)
T_{chws}	chilled water supply temperature [°F]	Time-series sensor reading from BAS	Read
$T_{econ,set}$	dry bulb economizer setpoint [°F]	Time-series control setting from BAS	Read/Write
T_{hws}	heating water supply temperature [°F]	Time-series sensor reading from BAS	Read
T_{ma}	mixed air temperature [°F]	Time-series sensor reading from BAS	Read
T_{oa}	outdoor air temperature [°F]	Time-series sensor reading from BAS	Read
$T_{oa,ws}$	auxiliary outdoor air temperature [°F]	Time-series sensor reading from local weather station	Read
T_{ra}	return air temperature [°F]	Time-series sensor reading from BAS	Read
T_{sa}	supply air temperature [°F]	Time-series sensor reading from BAS	Read
$T_{sa,set}$	supply air temperature setpoint [°F]	Time-series control setting from BAS	Read/Write
$T_{zda,i}$	discharge air temperature of the ith zone or VAV box [°F]	Time-series sensor reading from BAS	Read
$T_{za,i}$	zone air temperature of the ith zone [°F]	Time-series sensor reading from BAS	Read
u_{cc}	cooling coil valve control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read
$u_{chwp,stat}$	chilled water circulating pump status [ON/OFF]	Time-series status signal from BAS	Read

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

u_{dm}	mixing box damper control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read
u_{hc}	heating coil valve control signal where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read
$u_{hwp,stat}$	heating water circulating pump status [ON/OFF]	Time-series status signal from BAS	Read
$u_{rf,stat}$	return fan status [ON/OFF]	Time-series status signal from BAS	Read
$u_{sf,stat}$	supply fan status [ON/OFF]	Time-series status signal from BAS	Read
$u_{zhc,i}$	heating coil valve control signal for the ith VAV box where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read

Table 41: Variables and parameters used to implement and evaluate the functional test.

Abbreviation	Data point	Data type
auxRule	iterator variable used in loop to identify triggered auxiliary rules in the list “trigAuxRules”	Algorithm variable
cause_i	binary array defining possible condition-based faults for the ith (i = 1 to 28) APAR rule [1 x 21 array]	Algorithm parameter
error	binary variable indicating the array “NcauseFinal” is empty (True) or not empty (False)	Algorithm variable
excludeAuxCause	binary array of excluded causes based on auxiliary rules that are triggered[1 x 21 array]	Algorithm variable
excludeCause	binary array of excluded causes based on APAR rules that are evaluated but not triggered [1 x 21 array]	Algorithm variable
exclude_auxi	binary array of condition-based faults that can be excluded based on the ith (i = 1 to 5) auxiliary rule [1 x 21 array]	Algorithm parameter
exclude_i	binary array of condition-based faults that can be excluded based on the ith (i = 3, 7, 10, 13, and 19) rule in Table 42 [1 x 21 array]	Algorithm parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

functTestResult	binary variable indicating the outcome of the functional test [Pass or Fail]	Algorithm variable
modeCount_i	counter for number of data samples in Mode “i” (i = 1 to 4)	Algorithm variable
MT_{max}	threshold for maximum allowable number of mode transitions per hour	User-defined parameter
N_z	Number of zones used to evaluate the median zone temperature in auxiliary rule #4	User-defined parameter
Ncause	array of possible causes for a failed functional test based on triggered APAR rules and excluded causes from the cases in Table 42 and auxiliary rules	Algorithm variable
NcauseFinal	array of the largest element(s) of array “Ncause” which contains the condition-based fault(s) most likely to be the root cause of the failed test	Algorithm variable
NcauseTrig	array for counting the number of times individual condition-based faults are identified as possible causes for a triggered APAR rule [1 x 21 array]	Algorithm variable
notTrigRules	list of rules where APAR rules are evaluated and not triggered	Algorithm variable
numAuxRules	integer number of auxiliary rules (maximum value of “auxRule”)	Algorithm parameter
numRules	integer number of APAR rules (maximum value of “rule”)	Algorithm parameter
Q_{oa}/Q_{sa}	outdoor airflow fraction = $(T_{ma} - T_{ra}) / (T_{oa} - T_{ra})$ used in APAR	User-defined parameter
$(Q_{oa}/Q_{sa})_{min}$	design minimum outdoor air fraction used in APAR	User-defined parameter
rule	iterator variable used in loop to identify triggered APAR rules in the list “trigRules” and non-triggered APAR rules in the list “notTrigRules”	Algorithm variable
sampleCount	counter for number of data samples in a test segment	Algorithm variable
singleCause	binary variable indicating whether or not a single cause for the failed test has been identified	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

$T_{chws,max}$	maximum allowable chilled water supply temperature for testing [°F]	User-defined parameter
$T_{db,hiLim}$	dry bulb economizer high limit shutoff [°F]	User-defined parameter
$T_{hws,min}$	minimum allowable heating water supply temperature for testing [°F]	User-defined parameter
$T_{oa,min}$	minimum allowable outdoor air temperature for testing [°F]	User-defined parameter
$T_{ra,max}$	maximum allowable return air temperature for testing [°F]	User-defined parameter
$T_{sa,set,min}$	minimum allowable supply air temperature setpoint [°F]	User-defined parameter
$\Delta T_{ra,oa,min}$	minimum temperature difference between the return and outdoor air streams used in APAR [°F]	User-defined parameter
ΔT_{rf}	temperature rise across the return fan used in APAR and auxiliary rules [°F]	User-defined parameter
ΔT_{sf}	temperature rise across the supply fan used in APAR [°F]	User-defined parameter
testSeg	integer variable for labeling test segments [1 to 4]	Algorithm variable
testSegCount	integer variable for counting samples in a test segment	Algorithm variable
testSegEnd	real array with timestamp corresponding to the end of each test segment [1 x 4 array]	Algorithm variable
testSegLabel	integer array with label (“testSeg”) for each test segment [1 x 4 array]	Algorithm variable
testSegStart	real array with timestamp corresponding to the start of each test segment [1 x 4 array]	Algorithm variable
trigAuxRules	list of triggered auxiliary rules	Algorithm variable
trigRules	list of triggered APAR rules	Algorithm variable
ε_{cc}	cooling coil valve control signal threshold used in APAR	User-defined parameter
ε_{dm}	mixing box damper control signal threshold used in APAR	User-defined parameter
ε_{f}	outdoor airflow fraction threshold used in APAR	User-defined

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		parameter
ε_{hc}	heating coil valve control signal threshold used in APAR	User-defined parameter
ε_{oat}	outdoor air temperature threshold used in auxiliary rules	User-defined parameter
ε_{t}	temperature threshold used in APAR and auxiliary rules	User-defined parameter
ε_{zat}	zone air temperature threshold used in auxiliary rules	User-defined parameter
ε_{zdat}	zone discharge air temperature threshold used in auxiliary rules	User-defined parameter

5.4.5 Part 1_Proactive Functional Test

This section provides the steps for checking the test conditions and forcing the AHU to operate in four different operating modes.

5.4.5.1 Check test conditions

Prior to executing the test, the following conditions must be verified:

- Outdoor air temperature: $50°F \leq T_{oa} \leq T_{ra} - \Delta T_{ra,oa,min}$
- Return air temperature: $T_{oa} + \Delta T_{ra,oa,min} \leq T_{ra} \leq 80°F$
- Chilled water supply temperature: $T_{chws} \leq 45°F$
- Heating water supply temperature: $T_{hws} \geq 120°F$
- Supply fan and return fan are operating (as indicated by fan status points or other proof the fans are operating)
- Heating water and chilled water circulation pumps are operating (as indicated by pump status points or other proof the pumps are operating)
- AHU and VAV boxes are operating in occupied mode

Pseudocode verifying the test conditions are satisfied follows:

```
# Assign values to T_oa,min=50°F, T_ra,max=80°F, ΔT_ra,oa,min=10°F,
#   T_chws,max=45°F, T_hws,min=120°F
# Read T_oa, T_ra, T_chws, T_hws, u_rf,stat, u_sf,stat, u_chwp,stat, u_hwp,stat and check if
#   test conditions are satisfied
      # if (T_oa >= T_oa,min & T_oa <= T_ra - ΔT_ra,oa,min
      #   & T_ra >= T_oa + ΔT_ra,oa,min & T_ra <= T_ra,max
      #   & T_chws <= T_chws,max
      #   & T_hws >= T_hws,min
      #   & u_rf,stat == True
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
#   & u_sf,stat == True
#   & u_chwp,stat == True
#   & u_hwp,stat == True):
      # runTest = True
# else:
      # runTest = False
```

5.4.5.2 Force AHU to operate in four operating modes and record test data

If *runTest* is “True” after checking the test conditions the functional test can be executed. The test involves modifying the supply air temperature setpoint ($T_{sa,set}$) and the economizer dry-bulb high limit shutoff setpoint ($T_{econ,set}$) every 30 minutes to cause the AHU sequencing logic to force the unit into each of its four normal operating modes, beginning with Mode 1 (heating) in the first 30-minute test segment, then Mode 2 (cooling with outdoor air), Mode 3 (mechanical cooling with 100% outdoor air), and finally Mode 4 (mechanical cooling with minimum outdoor air), if the AHU is operating correctly. The setpoints used in the four operating modes are determined from the values of T_{oa} and T_{ra} at the beginning of each 30 minute test segment as follows:

- Mode1: $T_{sa,set} = T_{ra} + 5°F$, $T_{econ,set} = 75°F$
- Mode2: $T_{sa,set} = (T_{oa} + T_{ra})/2$, $T_{econ,set} = 75°F$
- Mode3: $T_{sa,set} = \max(T_{oa} - 5°F, 50°F)$, $T_{econ,set} = 75°F$
- Mode4: $T_{sa,set} = \max(T_{oa} - 5°F, 50°F)$, $T_{econ,set} = T_{oa} - 10°F$

As the test is performed, data are collected and stored at a fixed sampling rate. Faster sampling rates such as 1-2 minutes are preferred, however, the closed-loop test described here can be performed at slower sampling rates of 5-15 minutes. The start and end time of each test segment should be recorded, and each test segment should be labeled 1-4 to designate the mode of operation that was intended for that segment. The data to be read and stored are T_{sa}, T_{ra}, T_{oa}, $T_{oa,ws}$, T_{ma}, $T_{sa,set}$, u_{cc}, u_{dm}, u_{hc}, $u_{zhc,i}$, $T_{zda,i}$, and $T_{za,i}$, for all i, where i is the zone or VAV box index. Descriptions of all data points are provided in Table 40.

Pseudocode for implementing the functional test follows:

```
# If test conditions are satisfied, run the test and store the
#   following data points with timestamp: T_econ,set, T_ma, T_oa, T_oa,ws, T_ra,
# T_sa, T_sa,set, u_cc,  u_dm, u_hc, u_zhc,i, T_zda,i, and T_za,i
# Data are sampled and stored at 1 minute intervals (illustrative
#   example - other sampling intervals can be used)
     # if runTest == True:
           # Assign values to parameters to determine T_sa,set and T_econ,set
           # T_db,hiLim = 75°F
           # T_sa,set,min = 50°F
           # testSeg = 1
           # while testSeg < 5:
                # Read current T_ra, T_oa   # update each new test
     segment
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                # testSegCount = 0
                # testSegStart[testSeg] = currentTime
                # testSegLabel[testSeg] = testSeg      # intended
mode
                # while testSegCount < 30:  # each segment runs 30
min
                    # if testSeg == 1:
                        # T_sa,set = T_ra + 5°F
                        # T_econ,set = T_db,hiLim
                    # elif testSeg == 2:
                        # T_sa,set = (T_oa + T_ra)/2
                        # T_econ,set = T_db,hiLim
                    # elif testSeg == 3:
                        # T_sa,set = max(T_oa-5°F,T_sa,set,min)
                        # T_econ,set = T_db,hiLim
                    # elif testSeg == 4:
                        # T_sa,set = max(T_oa-5°F,T_sa,set,min)
                        # T_econ,set = T_oa - 10°F
                    # testSegCount += 1
                # testSegEnd[testSeg] = currentTime
                # testSeg += 1
```

5.4.6 Part I_Functional Test Analysis

This section describes the analysis of the test data from *5. Part I_Proactive Functional Test*. First we describe the overall procedure for the analysis, and then its implementation with pseudocode. The rule-based analysis is built upon the APAR rule set (House et al., 2001)[15] and extends APAR through the introduction of new rules and the analysis of data from multiple modes of operation.

5.4.6.1 Procedure

The procedure for analyzing the functional test data set is described next and is comprised of the following steps: 1) Group test data into different operating modes; 2) Apply APAR and auxiliary rules for assessing AHU performance; 3) Identify shared faults from triggered APAR rules; 4) Identify excluded faults from non-triggered APAR rules; 5) Identify excluded faults from auxiliary rules; 6) Remove excluded faults from shared faults; and 7) Determine if a single root-cause fault identified. Note that if there are no APAR rules triggered in Step 2, the AHU has passed the functional test and is considered to be operating normally. In this case, Steps 3-7 are not implemented.

[15] House, J. M., Vaezi-Nejad, H., and Whitcomb, J. M. 2001. “An expert rule set for fault detection in air-handling units.” *ASHRAE Transactions*. 107(1):858-871.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.4.6.1.1 Group test data into different operating modes

APAR is designed to assess the performance of the components and control logic used for supply air temperature and outdoor air control and consists of 28 rules that identify conditions or symptoms that are indicative of a fault. The APAR rules are provided as Table 44 in the Appendix (Section 5.4.8). User-defined parameter values from House et al. (2001) needed to implement APAR (e.g., thresholds) are listed in Table 45. Additional rules, known as auxiliary rules, that extend APAR are described in Step 2 of this procedure. Using APAR and the auxiliary rules requires the data set to be divided according to the operating mode of the AHU.

Each of the four AHU operating modes for supply air temperature and outdoor air control has a subset of rules that apply only in that mode. Other rules apply across all four operating modes, while still others identify conditions that are inconsistent with any of the four operating modes (e.g., simultaneous heating and cooling). Control signals for the heating coil valve, cooling coil valve, and mixing box dampers are used at each sample time to assign data to an operating mode as follows:

- $u_{hc} > \varepsilon_{hc}$ AND $u_{cc} \leq \varepsilon_{cc}$ AND $u_{dm} \leq \varepsilon_{dm}$ → Mode 1 (1)
- $u_{hc} \leq \varepsilon_{hc}$ AND $u_{cc} \leq \varepsilon_{cc}$ AND $\varepsilon_{dm} \leq u_{dm} \leq 1 - \varepsilon_{dm}$ → Mode 2 (2)
- $u_{hc} \leq \varepsilon_{hc}$ AND $u_{cc} > \varepsilon_{cc}$ AND $u_{dm} > 1 - \varepsilon_{dm}$ → Mode 3 (3)
- $u_{hc} \leq \varepsilon_{hc}$ AND $u_{cc} > \varepsilon_{cc}$ AND $u_{dm} \leq \varepsilon_{dm}$ → Mode 4 (4)
- u_{hc}, u_{cc}, and u_{dm} are inconsistent with Modes 1-4 → Mode 5

where tolerances ε_{hc}, ε_{cc} and ε_{dm} are user defined and Mode 5 is an unknown mode of operation for which the control signals are inconsistent with operation in Modes 1-4. Data assigned to Mode 5 may indicate the presence of a sequencing logic error. The three control signals and supply air, return air, mixed air, and outdoor air temperature sensors are used in the rule-based analysis to assess the AHU performance.

5.4.6.1.2 Apply APAR and auxiliary rules for assessing AHU performance

The main components to the rule-based analysis of the test data are described below:

- Evaluate and identify APAR rules that are triggered and those that are not triggered; and
- Evaluate and identify auxiliary rules that are triggered.

If none of the APAR rules are triggered, the AHU has passed the functional test and no further analysis is performed for this test (i.e., Steps 3-7 are not executed). If one or more of the APAR rules are triggered, the AHU has failed the functional test and analysis of the test data continues with Step 3.

<u>Evaluate and identify APAR rules that are triggered and those that are not triggered</u>
Once the data are divided according to operating mode, the APAR rules (Table 44) can be applied and triggered rules identified. The knowledge of which APAR rules are triggered is used in Step 3 to find the shared faults.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

In a similar manner, if it is assumed that at most there is one fault present during a closed-loop test, specific APAR rules that are evaluated and not triggered can be used to remove one or more condition-based faults as possible explanations for a failed functional test. As an example, if the AHU maintains the supply air temperature at setpoint during the heating mode (Mode 1 - APAR Rule #3 is not triggered), it can be concluded that the heating coil valve cannot be stuck. If a stuck heating coil valve is a possible explanation for one or more APAR rules that are triggered in the functional test, that explanation can be removed. Therefore, the knowledge of which APAR rules are not triggered is used in Step 4 to identify excluded faults. Importantly, removal of a possible explanation for a failed functional test can only be done if the appropriate APAR rule is *evaluated and not triggered*.

Evaluate and identify auxiliary rules that are triggered

The idea of removing possible causes or explanations for a failed functional test can be extended by adding rules to the existing 28 APAR rules. These additional rules are the auxiliary rules that were first mentioned in Step 1 of this analysis procedure. The knowledge of which auxiliary rules are triggered is used in Step 5 to identify excluded faults.

The first two auxiliary rules apply in Mode 2 and are derived from APAR Rule #7 (see Table 44). In Mode 2, the heating coil valve and cooling coil valve are commanded closed. Under these conditions, T_{sa} and T_{ma} should differ only by the temperature rise across the fan and any tolerances associated with normal sensor error. House et al. (2001) identify six possible explanations for why Rule #7 might be triggered, including a leaking cooling coil valve and a leaking heating coil valve. Further scrutiny of Rule #7 can eliminate one of these two possible faults when it is assumed that no more than one fault is present during the functional test. The first two auxiliary rules *are thus applied when Rule #7 is triggered* and are given by:

Auxiliary Rule #1: If APAR Rule #7 == True AND $T_{ma} < T_{sa} - \Delta T_{sf} - \varepsilon_t$ in Mode 2 (5)
→ Remove leaking cooling coil valve [5] as a possible fault

Auxiliary Rule #2: If APAR Rule #7 == True AND $T_{ma} > T_{sa} - \Delta T_{sf} + \varepsilon_t$ in Mode 2 (6)
→ Remove leaking heating coil valve [12] as a possible fault

where ε_t is a user selected threshold and the number in square brackets corresponds to the condition-based fault number from Table 46. Auxiliary Rule #1 indicates that T_{ma} is sufficiently low compared to T_{sa} that there could not be a leaking cooling coil valve fault present. Similarly, Auxiliary Rule #2 indicates T_{ma} is sufficiently high compared to T_{sa} that there could not be a leaking heating coil valve fault present.

Auxiliary Rule #3 compares T_{sa} to the discharge air temperature (T_{zda}) from several VAV boxes to determine if there is a supply air temperature sensor error. The rule applies in any of the four AHU operating modes, provided the VAV boxes used in the comparison are operating in the cooling or deadband mode (i.e., the valve for the reheat coil must be commanded closed), and is expressed as:

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Auxiliary Rule #3: If $| T_{sa} - T_{zda,i} | \leq \varepsilon_{zdat}$ for any i in i = 1 to 3 in any Mode (7)
→ Remove T_{sa} sensor error [1] as a possible fault

where ε_{zdat} is a user-selected threshold for the zone discharge air temperature. Auxiliary Rule #3 indicates that if there is agreement between T_{sa} and $T_{zda,i}$ for any of the three VAV boxes used in the comparison, it can be concluded that there is no supply air temperature sensor error. This conclusion holds if Auxiliary Rule #3 is triggered (i.e., satisfied) under any of the four AHU operating modes. Ideally the VAV boxes selected for the comparison should be in close proximity to the AHU. This will help minimize any temperature change due to heat transfer through the supply air ductwork.

Auxiliary Rule #4 compares T_{ra} to the median zone air temperature from a sample of zones served by the AHU. The rule applies in any of the four AHU operating modes and is expressed as:

Auxiliary Rule #4: If $| T_{ra} - \text{median}(T_{za,i}) | \leq \varepsilon_{zat}$ for i = 1 to N_z in any Mode (8)
→ Remove T_{ra} sensor error [2] as a possible fault

where $T_{za,i}$ is the air temperature of zone i, ε_{zat} is a user-selected threshold for the zone air temperature, and N_z is the number of zones used to determine the median. It is recommended that the median be calculated from the temperatures of all zones served by the AHU. If there is agreement between T_{ra} and the median value of the sampled zone temperatures, it can be concluded that there is no return air temperature sensor error. This conclusion holds if Auxiliary Rule #4 is triggered under any of the four AHU operating modes. It should be noted that the threshold ε_{zat} may need to account for a temperature rise across the return fan as well as normal sensor tolerances.

Auxiliary Rule #5 compares T_{oa} to the ambient air temperature measured by a local weather station ($T_{oa,ws}$). Alternatively, for a campus setting, the outdoor air temperature measured at another building could be used. The rule applies in any of the four AHU operating modes and is expressed as:

Auxiliary Rule #5: If $| T_{oa} - T_{oa,ws} | \leq \varepsilon_{oat}$ in any Mode (9)
→ Remove T_{oa} sensor error [4] as a possible fault

where ε_{oat} is a user selected threshold for the outdoor air temperature. Auxiliary Rule #5 indicates that if there is agreement between T_{oa} and $T_{oa,ws}$, it can be concluded that there is no outdoor air temperature sensor error.

5.4.6.1.3 Identify shared faults from triggered APAR rules

House et al. (2001) list one or more possible explanations for why each of the 28 rules in APAR might be triggered. These explanations are condition-based faults, such as a stuck cooling coil valve or a mixed air temperature sensor error, that can lead to the fault symptom identified by a particular APAR rule. In all, House et al. (2001) identified and tabulated 21 different

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

condition-based faults for the APAR rules. This tabulation is provided in Table 46 of the Appendix (Section 5.4.8).

If it is assumed that no more than one fault is present during a closed-loop functional test and multiple APAR rules are triggered during the test, then the triggered rules must share at least one condition-based fault among their possible fault explanations. Stated another way, each triggered rule has a set of condition-based faults as possible explanations and the intersection of these sets should contain the actual condition-based fault. In this step of the procedure, the shared condition-based faults for the triggered APAR rules are identified.

5.4.6.1.4 Identify excluded faults from non-triggered APAR rules

In this step, APAR rules that were evaluated and not triggered are used to identify faults that can be excluded as possible causes for a failed functional test. Table 42 lists five APAR rules that can be used in this way. Also listed for each rule is the condition-based fault that can be removed if the rule is evaluated and not satisfied.

Table 42: Condition-based faults that can be removed as possible causes of a failed functional test based on APAR rules that are evaluated in the test and not triggered.

Mode	Rule Evaluated and not Triggered	Condition-based Fault(s) Removed *
1	#3: $\lvert u_{hc} - 1\rvert \le \varepsilon_{hc}$ & $T_{sa,set} - T_{sa} \ge \varepsilon_t$	Stuck heating coil valve [13]
2	#7: $\lvert T_{sa} - \Delta T_{sf} - T_{ma}\rvert > \varepsilon_t$	Supply air temperature sensor error [1] Mixed air temperature sensor error [3] Leaking cooling coil valve [5] Leaking heating coil valve [12]
3	#10: $\lvert T_{oa} - T_{ma}\rvert > \varepsilon_t$	Mixed air temperature sensor error [3] Outdoor air temperature sensor error [4]
3	#13: $\lvert u_{cc} - 1\rvert \le \varepsilon_{cc}$ & $T_{sa} - T_{sa,set} \ge \varepsilon_t$	Stuck cooling coil valve [6]
4	#19: $\lvert u_{cc} - 1\rvert \le \varepsilon_{cc}$ & $T_{sa} - T_{sa,set} \ge \varepsilon_t$	Stuck cooling coil valve [6]

* The number in brackets in the last column corresponds to the explanation number for the condition-based fault from Table 46.

5.4.6.1.5 Identify excluded faults from auxiliary rules

In this step, individual auxiliary rules that are triggered are combined to identify all the faults that can be excluded on this basis following the instructions in Step 2.

5.4.6.1.6 Remove excluded faults from shared faults

In this step, results from Steps 3-5 are combined to determine the smallest set of condition-based faults that are possible explanations for the failed functional test. More specifically, this procedure removes any faults identified in Step 4 (Identify excluded faults from non-triggered APAR rules) and Step 5 (Identify excluded faults from auxiliary rules) from the

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

shared condition-based faults identified in Step 3 (Identify shared faults from triggered APAR rules).

5.4.6.1.7 Determine if a single root-cause fault identified

The final step of this analysis is to determine if a single condition-based fault has been identified as the root cause of the failed functional test. If a single fault has been identified, the test analysis is complete. If multiple condition-based faults have been identified, further testing is needed and two such tests are described in *7. Part II_Proactive Functional Test and Analysis*. If there are no condition-based faults identified, an error has occurred in the analysis (likely an error in the software implementation of the functional test analysis) because at least one fault should be listed as a possible explanation for any triggered APAR rules from Step 2.

5.4.6.2 Implementation and pseudocode

This section describes the implementation and pseudocode of the procedure just described for analyzing the functional test data set, following the same steps: 1) Group test data into different operating modes; 2) Apply APAR and auxiliary rules for assessing AHU performance; 3) Identify shared faults from triggered APAR rules; 4) Identify excluded faults from non-triggered APAR rules; 5) Identify excluded faults from auxiliary rules; 6) Remove excluded faults from shared faults; and 7) Determine if a single root-cause fault identified.

5.4.6.2.1 Group test data into different operating modes

Recall from the description of *Part I_Proactive Functional Test* that each of the four test segments of the functional test consist of 30 minutes of test data. In this implementation step, the data for each test segment are assigned to operating modes as described by Eq. (1)-(4). Averages for T_{ma}, T_{oa}, $T_{oa,ws}$, T_{ra}, T_{sa}, $T_{sa,set}$, u_{cc}, u_{dm}, u_{hc}, $u_{zhc,i}$, $T_{zda,i}$, and $T_{za,i}$ are calculated as the data are processed. In the first 15 minutes of the functional test, the averages are based on the arithmetic mean of all samples recorded to that point. The averages are moving averages based on a 15-minute sliding window after the initial 15 minutes.

5.4.6.2.2 Apply APAR and auxiliary rules for assessing AHU performance

Three arrays are initialized in this step and maintained throughout the test. The array (*trigRules*) will contain triggered APAR rules. A second array (*notTrigRules*) will contain APAR rules that are evaluated and not triggered. The third array (*trigAuxRules*) will contain triggered auxiliary rules.

If all data samples from the final 15 minutes of a given test segment come from a single mode of operation, averaged data values at the end of the test segment are processed through the appropriate APAR and auxiliary rules based on the mode of operation determined for that segment. Otherwise the data from that test segment are not processed through any rules. Default parameter values needed to implement the auxiliary rules are listed in Table 43.

Table 43: Default parameter values for implementing the auxiliary rules.

Parameter	Value

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

N_z	All zones served by AHU
ε_{oat}	3°F
ε_{zat}	3°F
ε_{zdat}	3°F

After processing the data, each of the three arrays should contain no more than one occurrence of a rule. If the array *trigRules* is empty at the conclusion of the test, the AHU is deemed to have passed the test. If one or more APAR rules have been triggered, the AHU is deemed to have failed the test and further analysis will be conducted to determine the cause of the failure.

Pseudocode implementing Step 1 and Step 2 of the analysis procedure assuming a 1-minute sampling interval follows:

```
# For each test segment, determine mode and evaluate APAR and
#   auxiliary rules
     # testSeg = 1
     # Initialize arrays for triggered and non-triggered rules
     # trigRules = []    # triggered APAR rules
     # notTrigRuless = [] #  APAR rules that are
     #                       evaluated and not triggered
     # trigAuxRules = [] # triggered auxiliary rules
     # while testSeg < 5:
          # modeCount_i = 0 for i = 1:4
          # testSegCount = 1
          # if testSegCount <= 15 && testSeg == 1:
               # Calculate averages for Tma, Toa, Toa,ws, Tra, Tsa,
          Tsa,set,
               #   ucc, udm, uhc, uzhc,i, Tzda,i, and Tza,i based on
               #   arithmetic mean of first testSegCount samples
               # testSegCount += 1
          # elif testSegCount <= 15:
               # Calculate moving averages based on 15 sample
          sliding
               #   window for Tma, Toa, Toa,ws, Tra, Tsa, Tsa,set, ucc, udm,
               #   uhc, uzhc,i, Tzda,i, and Tza,i
               # testSegCount += 1
          # else 15 < testSegCount <= 30: # 1-minute data
               # if uhc > εhc & ucc ≤ εcc & udm ≤ εdm:
               #    modeCount_1 += 1  # Mode 1
               # elif uhc ≤ εhc & ucc ≤ εcc & udm > εdm & udm < 1 - εdm:
               #    modeCount_2 += 1  # Mode 2
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
                # elif u_hc ≤ ε_hc & u_cc > ε_cc & u_dm ≥ 1 - ε_dm:
                #     modeCount_3 += 1  # Mode 3
                # elif u_hc ≤ ε_hc & u_cc > ε_cc & u_dm ≤ ε_dm:
                #     modeCount_4 += 1  # Mode 4
                # else:
                #     modeCount_5 += 1  # Mode 5 (unknown mode)
                # Calculate moving averages based on 15 sample
          sliding
                #   window for T_ma, T_oa, T_oa,ws, T_ra, T_sa, T_sa,set, u_cc, u_dm,
                #   u_hc, u_zhc,i, T_zda,i, and T_za,i
                # testSegCount += 1

          # if testSegCount == 31:
                # testSeg += 1
                # if max(modeCount_i) == 15:
                      # Evaluate APAR and auxiliary rules for the
                mode
                      #   corresponding to max(modeCount_i) using
                      #   current values of averaged data
                           # Update trigRules with new triggered APAR
                           #   rules
                           # Update notTrigRules with APAR rules
                           #   that are evaluated and not triggered
                           # Update trigAuxRules with new triggered
                           #   auxiliary rules

    # if trigRules = []:
    #     functTestResult = "Pass"
    # else:
    #     functTestResult = "Fail"
    #     goto "Part I_Functional Test Analysis - Step 3"
```

A flow chart showing the main steps outlined in this chunk of pseudocode is shown in Figure 58.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

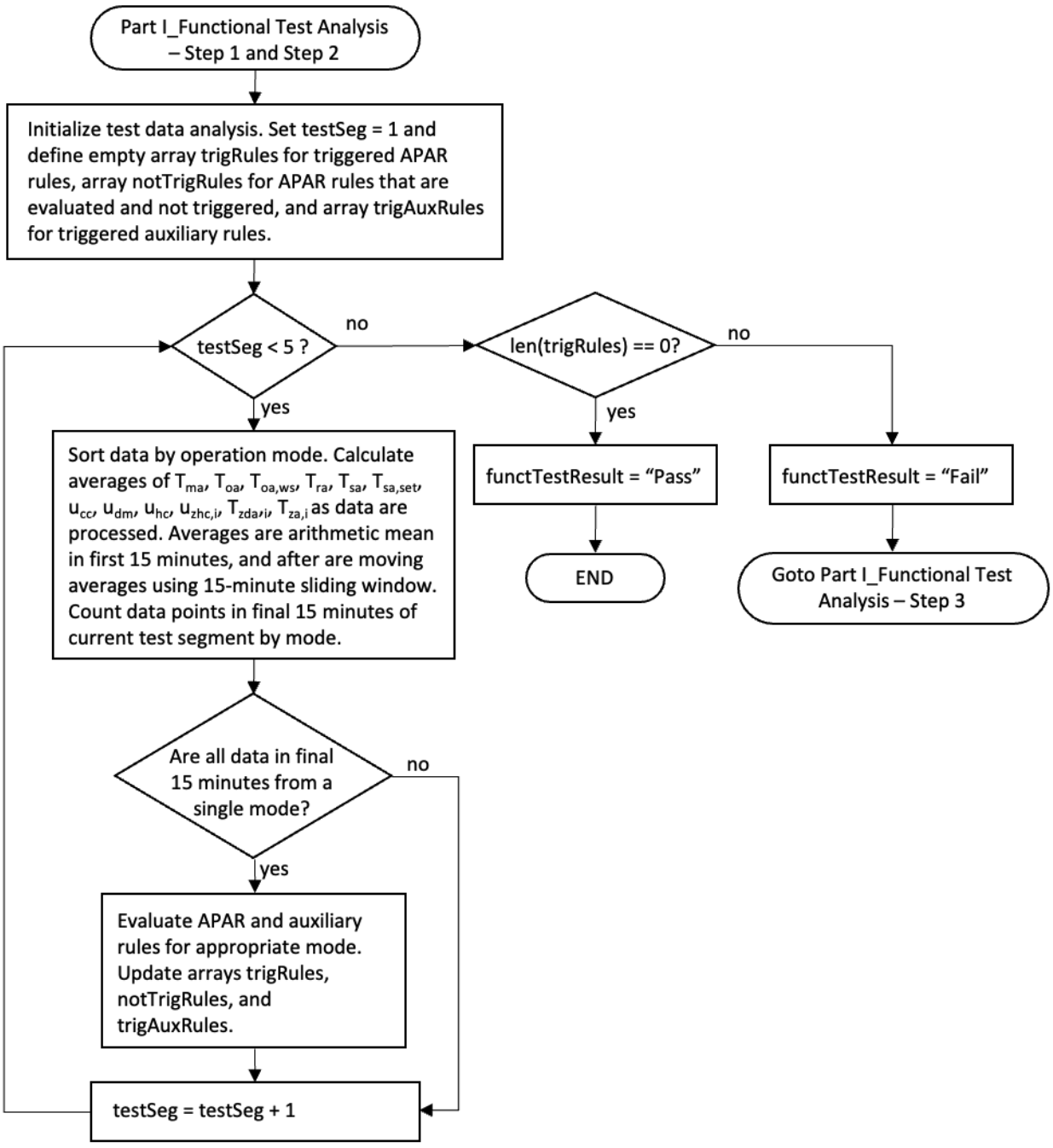


Figure 58: Flowchart implementing Step 1 and Step 2 of the analysis procedure.

5.4.6.2.3 Identify shared faults from triggered APAR rules

House et al. (2001) identified and tabulated 21 different condition-based faults for the APAR rules (Table 46). To find the intersection of the set of possible explanations for each triggered APAR rule, a 1x21 array *NcauseTrig* is defined and initialized with all elements equal to 0. The second dimension of this array corresponds to the total number of condition-based faults

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

identified by House et al. (2001). This array will be used to count the number of times each condition-based fault is identified as a possible explanation in the triggered APAR rules. Next, a 1x21 array *cause_rule* (where *rule* = 1 to 28) is defined for each APAR rule, where an element of the array is assigned the value 1 if the condition-based fault corresponding to that element is a possible cause for a particular APAR rule being triggered, and the element is assigned the value 0 otherwise. As an example, referring to Table 46, APAR Rule #1 has eight condition-based faults as possible explanations for why this rule might be triggered. The eight condition-based faults are:

1 - Supply air temperature sensor error
3 - Mixed air temperature sensor error
5 - Leaking cooling coil valve
6 - Stuck cooling coil valve
13 - Stuck heating coil valve
14 - Undersized heating coil
16 - Heating water supply temperature too low
17 - Heating water circulating pump fault

Thus, the array *cause_rule* where *rule* = 1 (i.e., *cause_1*) is defined as:

```
cause_1 = [1,0,1,0,1,1,0,0,0,0,0,0,1,1,0,1,1,0,0,0,0]
```

where we note that elements 1, 3, 5, 6, 13, 14, 16, and 17 are set to 1. Arrays *cause_2* through *cause_28* are defined in the same manner.

Next, for each rule in the array *trigRules*, the array *cause_rule* is added to the array *NcauseTrig*. This step counts the number of times each condition-based fault is identified as a possible explanation for a triggered APAR rule. When this step has been completed, there should be one or more elements in *NcauseTrig* that are equal to the number of rules in the array *trigRules*. These elements correspond to the condition-based faults that are the possible causes for the failed functional test.

Pseudocode implementing Step 3 of the analysis procedure follows:

```
# Initialize 1x21 array for counting possible causes for triggered
#   APAR rules
      #    NcauseTrig = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x21 array of possible causes for each APAR rule
      #    cause_1 = [1,0,1,0,1,1,0,0,0,0,0,0,1,1,0,1,1,0,0,0,0]
      #    cause_2 = [0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
      #    cause_3 = [1,0,0,0,1,1,0,0,0,0,0,0,1,1,1,1,1,0,0,1,0]
      #    cause_4 = [1,0,0,0,1,1,0,0,0,0,0,0,1,1,1,1,1,0,0,1,0]
      #    cause_5 = [1,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #    cause_6 = [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      #     cause_7 = [1,0,1,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0]
      #     cause_8 = [1,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0]
      #     cause_9 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_10 = [0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0]
      #     cause_11 = [1,0,1,0,0,1,1,0,1,1,0,1,1,0,0,0,0,0,0,0,0]
      #     cause_12 = [1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_13 = [1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_14 = [1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_15 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_16 = [1,0,1,0,0,1,1,0,1,1,0,1,1,0,0,0,0,0,0,0,0]
      #     cause_17 = [1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_18 = [0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
      #     cause_19 = [1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_20 = [1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0]
      #     cause_21 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_22 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_23 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_24 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
      #     cause_25 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
      #     cause_26 = [0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #     cause_27 = [0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #     cause_28 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
      #     rule = 1
      #     numRules = 28
# Count the possible causes for all triggered rules in array
trigRules
      #     While rule <= numRules:
            #     if rule in trigRules:
                  #     NcauseTrig = NcauseTrig + cause_rule
            #     rule += 1
      #     goto "Part I_Functional Test Analysis - Step 4"
```

A flowchart outlining the main steps in this chunk of pseudocode is shown in Figure 59.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

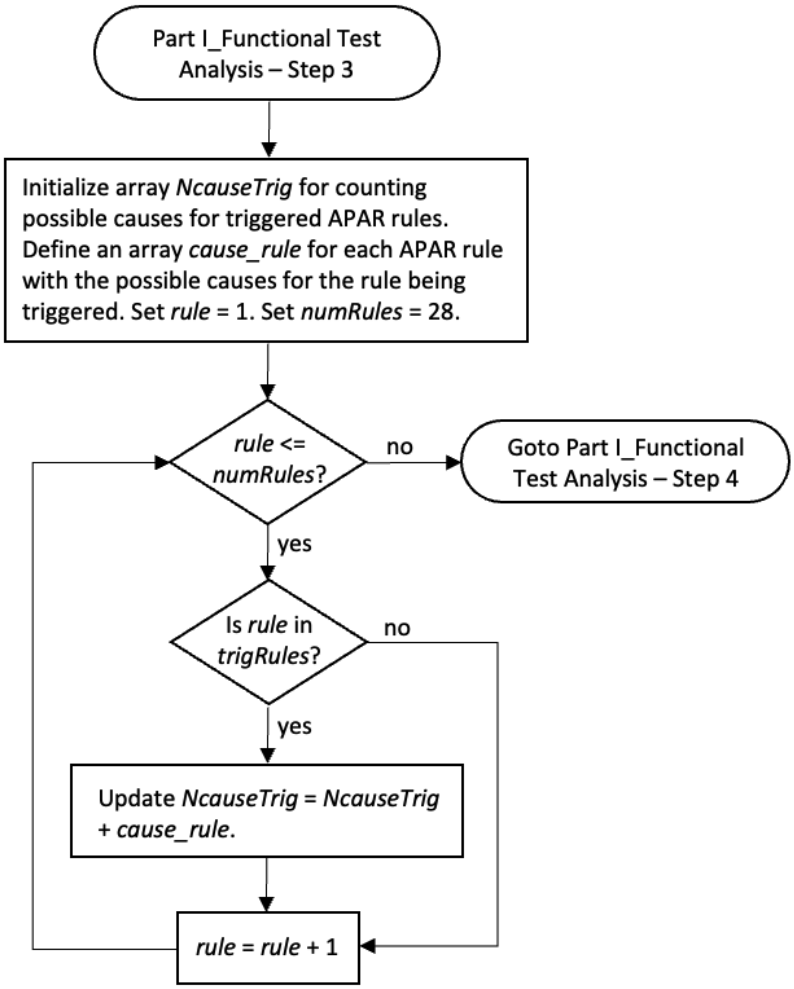


Figure 59: Flowchart implementing Step 3 of the analysis procedure.

5.4.6.2.4 Identify excluded faults from non-triggered APAR rules

This step implements the logic needed to remove one or more condition-based faults as a possible cause of a failed functional test based on APAR rules that were evaluated and not triggered. A 1x21 array *excludeCause* is defined and initialized with all elements equal to 0. Next, a 1x21 array *exclude_rule* is defined for each rule in Table 42, where an element of the array is assigned the value 1 if the condition-based fault corresponding to that element can be removed as a possible cause for a failed functional test, and the element is assigned the value 0 otherwise. As an example, the array *exclude_3* (i.e., *rule* is set equal to 3 corresponding to APAR Rule #3) is defined as

```
exclude_3 = [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
```

because, as shown in Table 42, when APAR Rule #3 is evaluated and not triggered, the stuck heating coil valve fault (explanation #13 in Table 46) can be removed as a possible cause for the failed functional test. Thus, element 13 of the array exclude_3 is set to 1. Similar arrays are

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

defined for the remaining rules in Table 42.

Next, for each rule in the array *notTrigRules*, the array *exclude_rule* is added to the array *excludeCause*. When this step has been completed, the elements of array *excludeCause* that are greater than or equal to 1 will indicate the condition-based faults that can be removed as a possible cause for the failed functional test.

Pseudocode implementing Step 4 of the analysis procedure follows:

```
# Initialize 1x21 array for identifying excluded causes based on APAR
#   rules that are evaluated and not triggered
      #     excludeCause = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x21 array of excluded causes for each rule in Table 42
      # Rule 3 evaluated and not triggered
      #     exclude_3 = [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
      # Rule 7 evaluated and not triggered
      #     exclude_7 = [1,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
      # Rule 10 evaluated and not triggered
      #     exclude_10 = [0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      # Rule  13 evaluated and not triggered
      #     exclude_13 = [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      # Rule 19 evaluated and not triggered
      #     exclude_19 = [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Define loop variables
      #     rule = 1
      #     numRules = 28
# Update array excludeCause using rules in array notTrigRules
      #     While rule <= numRules:
            #     if rule in notTrigRules:
                  #     excludeCause = excludeCause + exclude_rule
            #     rule += 1
      #     goto "Part I_Functional Test Analysis - Step 5"
```

A flowchart outlining the main steps in this chunk of pseudocode is shown in Figure 60.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

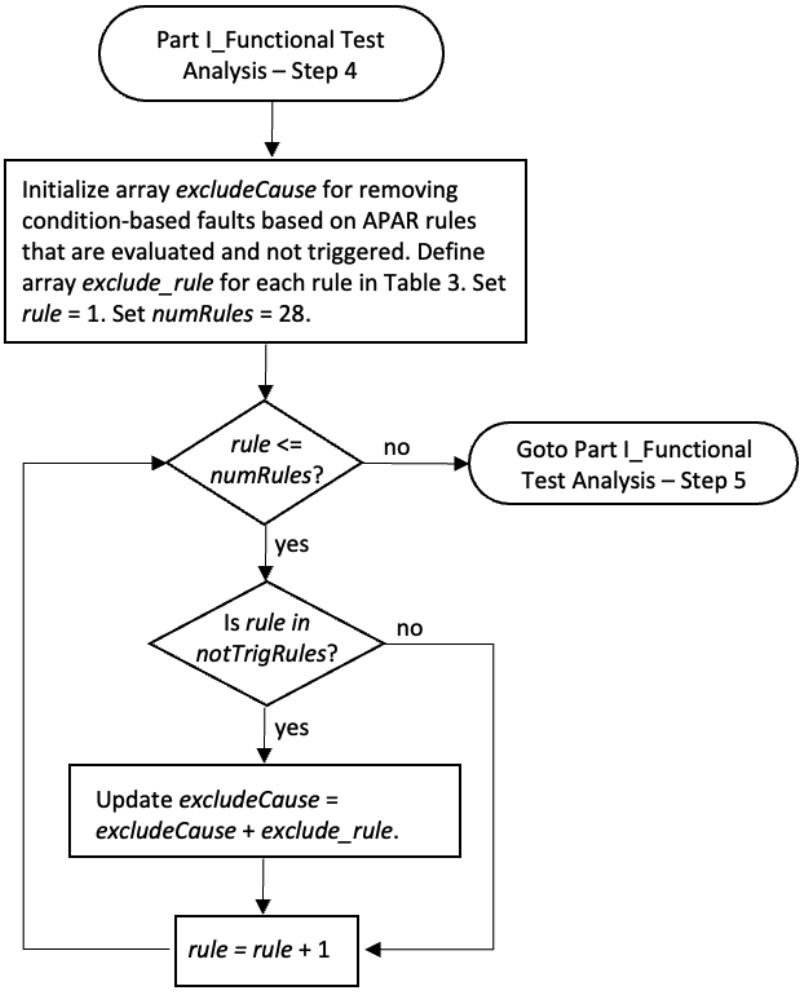


Figure 60: Flowchart implementing Step 4 of the analysis procedure.

5.4.6.2.5 Identify excluded faults from auxiliary rules

This step identifies condition-based faults that can be removed as possible causes of a failed functional test based on the auxiliary rules. To implement the logic, a 1x21 array *excludeAuxCause* is defined and initialized with all elements equal to 0. Next, a 1x21 array *exclude_auxRule* is defined for each auxiliary rule, where an element of the array is assigned the value 1 if the condition-based fault corresponding to that element can be removed as a possible cause for a failed functional test based on that auxiliary rule, and the element is assigned the value 0 otherwise. As an example, the array *exclude_aux2* is defined as

```
exclude_aux2 = [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
```

where “auxRule” is set equal to “2” and signifies that this array corresponds to Auxiliary Rule #2. If Auxiliary Rule #2 is triggered, a leaking heating coil valve can be excluded as a possible fault. Since this is the 12th condition-based fault in Table 46, a 1 is inserted in element 12 of *exclude_aux2*. Similar arrays are defined for each of the remaining auxiliary rules.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Then, for each rule in the array *trigAuxRules*, the array *exclude_auxRule* is added to the array *excludeAuxCause*. When this step has been completed, the elements of array *excludeAuxCause* that are greater than or equal to 1 will indicate the condition-based faults that can be removed as a possible cause for the failed functional test based on the auxiliary rules.

Pseudocode implementing Step 5 of the analysis procedure follows:

```
# Initialize 1x21 array for identifying excluded causes based on
#   triggered auxiliary rules
      #     excludeAuxCause =
      #          [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Assign 1x21 array of excluded causes for each auxiliary rule
      # In Mode 2; If Auxiliary Rule #1 is triggered:
            # Exclude leaking cooling coil valve fault (explanation #5
            #   from Table 46)
            #     exclude_aux1 =
            #          [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #
      # In Mode 2; If Auxiliary Rule #2 is triggered:
            # Exclude leaking heating coil valve fault (explanation
#12
            #   from Table 46)
            #     exclude_aux2 =
            #          [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]
      #
      # For any Mode; If Auxiliary Rule #3 is triggered:
            # Exclude T_sa fault (explanation #1 from Table 46)
            #     exclude_aux3 =
            #          [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #
      # For any Mode; If Auxiliary Rule #4 is triggered:
            # Exclude T_ra fault (explanation #2 from Table 46)
            #     exclude_aux4 =
            #          [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      #
      # For any Mode; If Auxiliary Rule #5 is triggered:
            # Exclude T_oa fault (explanation #4 from Table 46)
            #     exclude_aux5 =
            #          [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
# Define loop variables
      #     auxRule = 1
      #     numAuxRules = 5
# Update array excludeAuxCause using cases in array trigAuxRules
      #     While auxRule <= numAuxRules:
            #     if auxRule in trigAuxCases:
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
            #     excludeAuxCause = excludeAuxCause
            #           + exclude_auxRule
        #     auxRule += 1
    #     goto "Part I_Functional Test Analysis - Step 6"
```

A flowchart outlining the main steps in this chunk of pseudocode is shown in Figure 61.

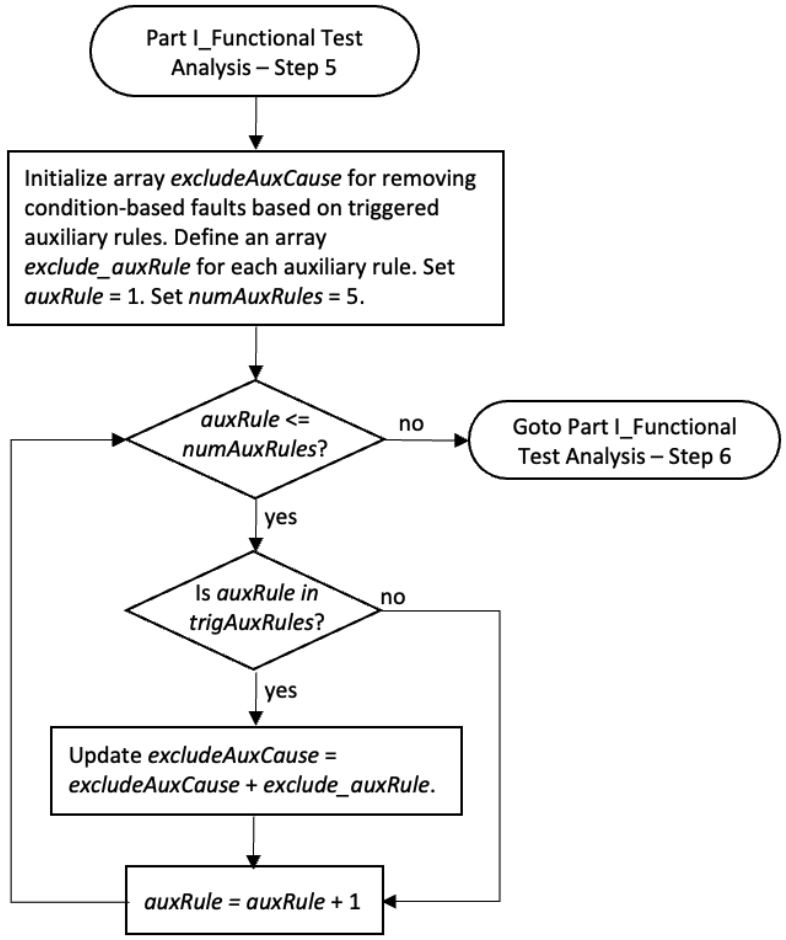


Figure 61: Flowchart implementing Step 5 of the analysis procedure.

5.4.6.2.6 Remove excluded faults from shared faults

In this step, any faults identified in Step 4 (Identify excluded faults from APAR rules) and Step 5 (Identify excluded faults from auxiliary rules) are removed from the shared condition-based faults identified in Step 3 (Identify shared faults from APAR rules). A 1x21 array *Ncause* is determined by subtracting the array *excludeCause* (Step 4) and the array *excludeAuxCause* (Step 5) from the array *NcauseTrig* (Step 3). The element(s) of *Ncause* with the largest magnitude identifies the condition-based fault(s) that is the cause of the failed functional test. Finally, an array *NcauseFinal* is initialized and used to store the largest element(s) of array *Ncause*. This array will contain the smallest set of condition-based faults that are possible

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

explanations for a failed functional test.

Pseudocode implementing Step 6 of the analysis procedure follows:

```
# Determine the array of possible causes for the failed
#   functional test based on triggered APAR rules and excluded causes
#   from the cases in Table 46 and auxiliary rules
     #    Ncause = NcauseTrig - excludeCause - excludeAuxCause
# Initialize array NcauseFinal, then store largest element(s) of
#   Ncause in this array
     # NcauseFinal = max(Ncause)
     # goto "Part I_Functional Test Analysis - Step 7"
```

5.4.6.2.7 Determine if a single root-cause fault identified

The final step (Step 7) of the analysis assesses whether a single condition-based fault has been determined as the cause for a failed functional test. Pseudocode implementing Step 7 of the analysis procedure follows:

```
# Determine if a single possible cause (condition-based fault) has
#   been identified for a failed functional test
     #    If len(NcauseFinal) == 0:
          #    error = True   # ERROR —> No causes found
          #    singleCause = False
     #    ElseIf len(NcauseFinal) == 1:
          #    error = False
          #    singleCause = True
     #    Else:
          #    error = False
          #    singleCause = False
     # If singleCause == True:
          #    End of Analysis
     # ElseIf error == False AND singleCause == False:
          #    Additional closed-loop, open-loop or hybrid tests are
          #      needed to determine root cause of failed functional
          #      test
          #    goto "Part II_Proactive Functional Test" to determine
          #      if there are further tests to be executed
```

A flowchart outlining the main steps in the pseudocode for Step 6 and Step 7 is shown in Figure 62.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

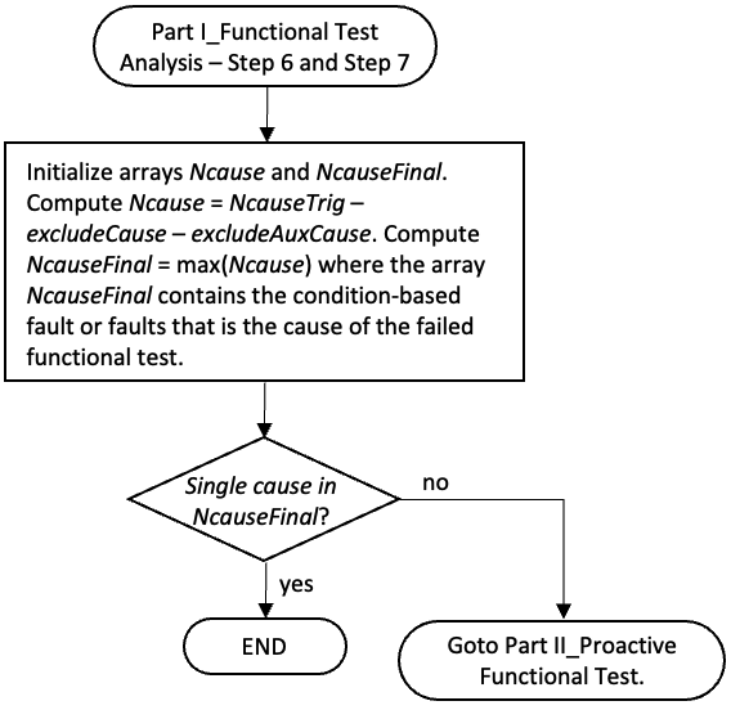


Figure 62: Flowchart implementing Step 6 and Step 7 of the analysis procedure.

5.4.7 Part II_Proactive Functional Test And Analysis

In the event of a failed functional test, the *Part I_Functional Test Analysis* described in Section 5.4.6 should be able to narrow the list of possible causes of the failed test to one to three condition-based faults. For failed tests with more than one possible condition-based fault, additional tests are needed to identify the root cause of the failure. Two closed-loop tests are described here that are designed to diagnose faults associated with the heating and cooling coil valves when they are among the possible faults identified in *6. Part I_Functional Test Analysis*.

The *Stuck/Leaking Heating Valve Test* is a three-step test implemented through changes in the supply air temperature setpoint. The first two steps are used to determine if the heating valve is stuck by adjusting $T_{sa,set}$ to values that require operation in the heating mode (Mode 1) and observing if the valve is able to modulate to satisfy the new setpoints.[16] The third step is used to determine if the heating valve is leaking by forcing operation in the cooling with outdoor air mode (Mode 2) and analyzing the data for evidence of leakage when the heating coil valve is closed. This test should only be performed if a stuck and/or leaking heating valve fault is determined to be one of the possible causes for the failed functional test and if the supply air

[16] No such requirement is imposed in the *Part I_Proactive Functional Test*, where the supply air temperature setpoint that is intended to force operation in the heating mode (Mode 1) could actually result in operation in the cooling with outdoor air mode (Mode 2) if a stuck or leaking heating coil valve is present.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

temperature sensor and mixed air temperature sensor are not identified as possible causes for the failed test.

The *Stuck/Leaking Cooling Valve Test* is very similar to the *Stuck/Leaking Heating Valve Test*. It is a three-step test implemented through changes in $T_{sa,set}$ and the economizer setpoint $T_{econ,set}$. The first two steps are used to determine if the cooling valve is stuck by adjusting $T_{sa,set}$ and $T_{econ,set}$ to values that should require operation in the cooling with minimum outdoor air mode (Mode 4) and observing if the valve is able to modulate to satisfy the new setpoints. The third step is used to determine if the cooling valve is leaking by forcing operation in the cooling with outdoor air mode (Mode 2) and analyzing the data for evidence of leakage when the cooling coil valve is closed. This test should only be performed if a stuck and/or leaking cooling valve fault is determined to be one of the possible causes for the failed functional test and if the supply air temperature sensor and mixed air temperature sensor are not identified as possible causes for the failed test.

Prior to executing either test, the following conditions must be verified:

- Outdoor air temperature: $50°F \leq T_{oa} \leq T_{ra} - \Delta T_{ra,oa,min}$
- Return air temperature: $T_{oa} + \Delta T_{ra,oa,min} \leq T_{ra} \leq 80°F$
- Chilled water supply temperature: $T_{chws} \leq 45°F$
- Heating water supply temperature: $T_{hws} \geq 120°F$
- Supply fan and return fan are operating (as indicated by fan status points or other proof the fans are operating)
- Heating water and chilled water circulation pumps are operating (as indicated by pump status points or other proof the pumps are operating)
- AHU and VAV boxes are operating in occupied mode

5.4.7.1 Stuck/Leaking Heating Valve Test

This test should be run only if a stuck and/or leaking heating coil valve (or both) are among the multiple condition-based faults that are possible causes for a failed functional test, and if the supply air temperature sensor and mixed air temperature sensor are not among the possible causes for the failed test.

5.4.7.1.1 Procedure

The steps involved in the test are: Step 1 - Maximum Heating; Step 2 - Intermediate Heating; and Step 3 - Cooling with Outdoor Air. Step 1 and Step 2 are used to check for a stuck valve, and Step 3 is used to check for a leaking valve.

<u>Step 1 - Maximum Heating</u>
The first step in this test is to set the supply air temperature setpoint to $T_{sa,set,1} = 100°F$ where the subscript “1” refers to Step 1 of this test. This subscripting convention is used throughout the description of this test to distinguish values used in different steps of the procedure. This should force the AHU to operate in the heating mode. After steady-state conditions are achieved

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

(approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,1}$), mixed air temperature ($T_{ma,1}$) and the control signal to the heating coil valve (u_{hc}).

- If $| T_{sa,1} - T_{sa,set,1} | \leq \varepsilon_t$, and $\varepsilon_{hc} < u_{hc} < 1 - \varepsilon_{hc}$, it can be concluded that the valve is not stuck.
- If $u_{hc} \leq \varepsilon_{hc}$ (valve commanded fully closed and AHU has transitioned to Mode 2), repeat Step 1 with a higher value of $T_{sa,set,1}$.
- If $| T_{sa,1} - \Delta T_{sf} - T_{ma,1} | \leq \varepsilon_t$ and $| u_{hc} - 1 | \leq \varepsilon_{hc}$ (valve commanded fully open), it can be concluded that the valve is stuck closed. Note, however, that this could be due to the actuator driving the valve in the wrong direction (i.e., closing the valve when it should be opening the valve).
- If $T_{sa,set,1} - T_{sa,1} > \varepsilon_t$ and $| u_{hc} - 1 | \leq \varepsilon_{hc}$, the valve could be stuck or the heating coil might not have adequate capacity to meet the setpoint. Proceed to Step 2.

Step 2 - Intermediate Heating

Change the value of the supply air temperature setpoint to $T_{sa,set,2} = T_{sa,1} - 5°F$ where $T_{sa,1}$ is the supply air temperature at the end of Step 1. If the AHU was unable to satisfy the setpoint in Step 1 due to a lack of capacity, the heating coil should now have adequate capacity to satisfy the reduced setpoint. After steady-state conditions are achieved (approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,2}$) and the control signal to the heating coil valve (u_{hc}).

- If $| T_{sa,2} - T_{sa,set,2} | \leq \varepsilon_t$, $u_{hc} \leq \varepsilon_{hc}$ and $u_{cc} \leq \varepsilon_{cc}$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$, it can be concluded that the valve is stuck and the AHU controls are compensating for the fault by operating in Mode 2.
- If $| T_{sa,2} - T_{sa,set,2} | \leq \varepsilon_t$ and $\varepsilon_{hc} < u_{hc} < 1 - \varepsilon_{hc}$ (i.e., the valve is modulating to maintain the setpoint), it can be concluded that the valve is not stuck. Proceed to Step 3 to determine if the valve is leaking.

Step 3 - Cooling with Outdoor Air

Change the value of the supply air temperature setpoint to $T_{sa,set,3} = \max(55°F, T_{oa} + 5°F)$. This should force the AHU to operate in Mode 2 (cooling with outdoor air) or Mode 3 (mechanical cooling with 100% outdoor air). After steady-state conditions are achieved (approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,3}$) and the control signal to the mixing box dampers (u_{dm}) and cooling coil valve (u_{cc}).

- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$, $T_{ma} \geq T_{sa} - \Delta T_{sf} - \varepsilon_t$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$ (i.e., operation in Mode 2 with the mixing box dampers modulating to maintain the setpoint), it can be concluded that the heating valve is not leaking.
- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$, $T_{ma} < T_{sa} - \Delta T_{sf} - \varepsilon_t$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$ (i.e., operation in Mode 2 with the mixing box dampers modulating to maintain the setpoint), it can be concluded that the heating valve is leaking.
- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$ and $\varepsilon_{cc} < u_{cc} < 1 - \varepsilon_{cc}$ (i.e., operation in Mode 3 with the cooling coil valve modulating to maintain the setpoint), it can be concluded that the heating valve is leaking.
- If none of these sets of conditions are satisfied, the test is inconclusive regarding whether or not the heating valve is leaking.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

If this test is performed immediately following the *Part I_Proactive Functional Test*, the data collected in that test for operation in Mode 2 (cooling with outdoor air) can be used instead of executing this step.

5.4.7.1.2 Implementation

The implementation of this test will parallel the implementation and pseudocode in 2.4.1 *Part I_Proactive Functional Test*. Specifically, after verifying the testing conditions are satisfied, setpoint changes will be implemented every 30 minutes to initiate a new step in the test. The resultant data will be labeled and stored. The analysis of the test data will parallel that described for Step 1 and Step 2 in *6. Part I_Functional Test Analysis*. Specifically, the sampled BAS data will be assigned to operating modes as defined by Eq. (1)-(4), and as the data are processed, moving averages of the data will be calculated. If all data samples from the final 15 minutes of a given test step are from a single mode of operation, averaged values of the data at the end of the step will be processed through the set of conditions defined for each step.

5.4.7.2 Stuck/Leaking Cooling Coil Valve Test

This test should be run only if a stuck or leaking cooling coil valve (or both) are among the multiple condition-based faults that are possible causes for a failed functional test, and if the supply air temperature sensor and mixed air temperature sensor are not among the possible causes for the failed test.

5.4.7.2.1 Procedure

The steps involved in the test are: Step 1 - Maximum Cooling; Step 2 - Intermediate Cooling; and Step 3 - Cooling with Outdoor Air. (Step 1 and Step 2 are used to check for a stuck valve, and Step 3 is used to check for a leaking valve.

<u>Step 1 - Maximum Cooling</u>

The first step in this test is to set the supply air temperature setpoint to $T_{sa,set,1}$ = min(50°F, T_{oa} - 5°F) and the economizer high limit shutoff to $T_{econ,set}$ = T_{oa} - 10°F. The subscript “1” in $T_{sa,set,1}$ refers to Step 1 of this test. This subscripting convention is used throughout the description of this test to distinguish values used in different steps of the procedure. The defined setpoints should force the AHU to operate in Mode 4 (mechanical cooling with minimum outdoor air). After steady-state conditions are achieved (approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,1}$), mixed air temperature ($T_{ma,1}$) and the control signal to the cooling coil valve (u_{cc}).

- If $| T_{sa,1} - T_{sa,set,1} | \leq \varepsilon_t$, and $\varepsilon_{cc} < u_{cc} < 1 - \varepsilon_{cc}$, it can be concluded that the valve is not stuck.
- If $u_{cc} \leq \varepsilon_{cc}$ (cooling valve commanded fully closed), repeat Step 1 with a lower value of $T_{sa,set,1}$.
- If $| T_{sa,1} - \Delta T_{sf} - T_{ma,1} | \leq \varepsilon_t$ and $| u_{cc} - 1 | \leq \varepsilon_{cc}$ (valve commanded fully open), it can be concluded that the valve is stuck closed. Note, however, that this could be due to the actuator driving the valve in the wrong direction (i.e., closing the valve when it should be opening the valve).

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

- If $T_{sa,1} - T_{sa,set,1} > \varepsilon_t$ and $| u_{cc} - 1 | \leq \varepsilon_{cc}$, the valve could be stuck or the cooling coil might not have adequate capacity to meet the setpoint. Proceed to Step 2.

Step 2 - Intermediate Cooling

Change the value of the supply air temperature setpoint to $T_{sa,set,2} = T_{sa,1} + 2°F$. If the AHU was unable to satisfy the setpoint in Step 1 due to a lack of capacity, the cooling coil should now have adequate capacity to satisfy the higher setpoint. After steady-state conditions are achieved (approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,2}$) and the control signal to the cooling coil valve (u_{cc}).

- If $| T_{sa,2} - T_{sa,set,2} | \leq \varepsilon_t$, $u_{hc} > \varepsilon_{hc}$ and $u_{cc} \leq \varepsilon_{cc}$ and $u_{dm} \leq \varepsilon_{dm}$, it can be concluded that the valve is stuck and the AHU controls are compensating for the fault by operating in the heating mode (Mode 1).
- If $| T_{sa,2} - T_{sa,set,2} | \leq \varepsilon_t$ and $u_{cc} > \varepsilon_{cc}$ (i.e., the cooling valve is modulating to maintain the setpoint), it can be concluded that the valve is not stuck. Proceed to Step 3 to determine if the valve is leaking.

Step 3 - Cooling with Outdoor Air

Change the value of the supply air temperature setpoint to $T_{sa,set,3} = T_{oa} + 5°F$ and change the economizer high limit shutoff to $T_{econ,set} = 75°F$. This should force the AHU to operate in Mode 2 (cooling with outdoor air) or Mode 1 (heating). After steady-state conditions are achieved (approximately 15-30 minutes), observe the values of the supply air temperature ($T_{sa,3}$) and the control signal to the mixing box dampers (u_{dm}) and heating coil valve (u_{hc}).

- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$, $T_{ma} \leq T_{sa} - \Delta T_{sf} + \varepsilon_t$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$ (i.e., operation in Mode 2 with the mixing box dampers modulating to maintain the setpoint), it can be concluded that the cooling valve is not leaking.
- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$, $T_{ma} > T_{sa} - \Delta T_{sf} + \varepsilon_t$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$ (i.e., operation in Mode 2 with the mixing box dampers modulating to maintain the setpoint), it can be concluded that the cooling valve is leaking.
- If $| T_{sa,3} - T_{sa,set,3} | \leq \varepsilon_t$ and $u_{hc} > \varepsilon_{hc}$ (i.e., operation in Mode 1 with the heating coil valve modulating to maintain the setpoint), it can be concluded that the cooling valve is leaking.
- If none of these sets of conditions are satisfied, the test is inconclusive regarding whether or not the cooling valve is leaking.

5.4.7.2.2 Implementation

The implementation of this test will parallel the implementation described in *7.1 Stuck/Leaking Heating Coil Valve Fault*.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.4.8 Appendix

Table 44: APAR rules from Schein et al. (2006)[17].

Mode	Rule #	**Rule Expression** (faults are signaled by true expressions)
Heating (Mode 1)	1	$T_{sa} < T_{ma} + \Delta T_{sf} - \varepsilon_t$
	2	For $\lvert T_{ra} - T_{oa} \rvert \geq \Delta T_{ra,oa,min}$: $\lvert Q_{oa}/Q_{sa} - (Q_{oa}/Q_{sa})_{min} \rvert > \varepsilon_f$
	3	$\lvert u_{hc} - 1 \rvert \leq \varepsilon_{hc}$ and $T_{sa,set} - T_{sa} \geq \varepsilon_t$
	4	$\lvert u_{hc} - 1 \rvert \leq \varepsilon_{hc}$
Cooling with Outdoor Air (Mode 2)	5	$T_{oa} > T_{sa,set} - \Delta T_{sf} + \varepsilon_t$
	6	$T_{sa} > T_{ra} - \Delta T_{rf} + \varepsilon_t$
	7	$\lvert T_{sa} - \Delta T_{sf} - T_{ma} \rvert > \varepsilon_t$
Mechanical Cooling with 100% Outdoor Air (Mode 3)	8	$T_{oa} < T_{sa,set} - \Delta T_{sf} - \varepsilon_t$
	9	$T_{oa} > T_{econ,set} + \varepsilon_t$
	10	$\lvert T_{oa} - T_{ma} \rvert > \varepsilon_t$
	11	$T_{sa} > T_{ma} + \Delta T_{sf} + \varepsilon_t$
	12	$T_{sa} > T_{ra} - \Delta T_{rf} + \varepsilon_t$
	13	$\lvert u_{cc} - 1 \rvert \leq \varepsilon_{cc}$ and $T_{sa} - T_{sa,set} \geq \varepsilon_t$
	14	$\lvert u_{cc} - 1 \rvert \leq \varepsilon_{cc}$
Mechanical Cooling with Minimum Outdoor Air (Mode 4)	15	$T_{oa} < T_{econ,set} - \varepsilon_t$
	16	$T_{sa} > T_{ma} + \Delta T_{sf} + \varepsilon_t$
	17	$T_{sa} > T_{ra} - \Delta T_{rf} + \varepsilon_t$
	18	For $\lvert T_{ra} - T_{oa} \rvert \geq \Delta T_{ra,oa,min}$: $\lvert Q_{oa}/Q_{sa} - (Q_{oa}/Q_{sa})_{min} \rvert > \varepsilon_f$
	19	$\lvert u_{cc} - 1 \rvert \leq \varepsilon_{cc}$ and $T_{sa} - T_{sa,set} \geq \varepsilon_t$
	20	$\lvert u_{cc} - 1 \rvert \leq \varepsilon_{cc}$
Unknown Occupied Mode (Mode 5)	21	$u_{cc} > \varepsilon_{cc}$ and $u_{hc} > \varepsilon_{hc}$ and $\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$
	22	$u_{hc} > \varepsilon_{hc}$ and $u_{cc} > \varepsilon_{cc}$
	23	$u_{hc} > \varepsilon_{hc}$ and $u_{dm} > \varepsilon_{dm}$

[17] Schein, J., Bushby, S. T., Castro, N. S., and House, J. M. 2006. “A rule-based fault detection method for air handling units.” *Energy and Buildings*. 38:1485-1492.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

	24	$\varepsilon_{dm} < u_{dm} < 1 - \varepsilon_{dm}$ and $u_{cc} > \varepsilon_{cc}$
All Occupied Modes (Modes 1, 2, 3, 4, and 5)	25	$\lvert T_{sa} - T_{sa,set} \rvert > \varepsilon_t$
	26	$T_{ma} < \min(T_{ra}, T_{oa}) - \varepsilon_t$
	27	$T_{ma} > \max(T_{ra}, T_{oa}) + \varepsilon_t$
	28	Number of mode transitions per hour > MT_{max}

Table 45: User-defined parameter values used in APAR rules from House et al. (2001).

Parameter	**Value** *
MT_{max}	6 transitions
$(Q_{oa}/Q_{sa})_{min}$	0.2 [-]
$\Delta T_{ra,oa,min}$	10°F
ΔT_{rf}	0°F
ΔT_{sf}	2°F
ε_{cc}	0.05 [-]
ε_{dm}	0.05 [-]
ε_f	0.3 [-]
ε_{hc}	0.05 [-]
ε_t	3°F

* Parameter values may need to be adapted based on the AHU configuration, field experience implementing FDD rules, etc.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

Table 46: Condition-based fault explanations for triggered APAR rules from House et al. (2001).

Rule	Condition-based Fault																			
	1 - Supply air temperature sensor error	2 - Return air temperature sensor error	3 - Mixed air temperature sensor error	4 - Outdoor air temperature sensor error	5 - Leaking cooling coil valve	6 - Stuck cooling coil valve	7 - Undersized cooling coil	8 - Fouled cooling coil	9 - Chilled water supply temperature too high	10 - Chilled water circulating pump fault	11 - Chilled water not available	12 - Leaking heating coil valve	13 - Stuck heating coil valve	14 - Undersized heating coil	15 - Fouled heating coil	16 - Heating water supply temperature too low	17 - Heating water circulating pump fault	18 - Leaking mixing box damper(s)	19 - Stuck mixing box damper(s)	20 - Controller tuning error	21 - Controller logic error
1	x		x		x	x							x	x		x	x				
2		x	x	x															x		
3	x				x	x							x	x	x	x	x			x	
4	x				x	x							x	x	x	x	x			x	
5	x			x	x	x															
6	x	x																			
7	x		x		x	x						x	x								
8	x			x								x	x					x	x		
9																					x
10			x	x														x	x		
11	x		x			x	x		x	x		x	x								
12	x	x				x	x	x	x	x	x	x	x							x	
13	x					x	x	x	x	x	x	x	x							x	
14	x					x	x	x	x	x	x	x	x							x	
15																					x
16	x		x			x	x		x	x		x	x								
17	x	x				x	x	x	x	x	x	x	x							x	
18		x	x	x															x		
19	x					x	x	x	x	x	x	x	x							x	
20	x					x	x	x	x	x	x	x	x							x	
21																					x
22																					x
23																					x
24																					x
25																				x	
26		x	x	x																	
27		x	x	x																	
28																				x	

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.5 Closed-loop Functional Tests of Static Pressure Reset

1. 5.5.1 Overview

Static pressure reset algorithms are energy-efficient control strategies aimed at reducing fan energy use. These strategies respond to changing zone loads by continually adapting the static pressure setpoint of an AHU to ensure all zones receive adequate airflow while avoiding unnecessarily high setpoints that result in fan energy waste.

This section describes testing procedures and analyses that can be automated and used to verify that a static pressure reset algorithm is implemented and is functioning to reset the static pressure setpoint in response to the test conditions imposed. The test does not, however, assess whether the implemented static pressure algorithm can be improved. The test involves overriding the static pressure setpoint from its current value to its maximum value, holding it at the maximum/minimum value for a period of time, and then releasing the setpoint and allowing the static pressure reset algorithm to adjust it in response to the current zone loads. Performance is evaluated based on evidence that the static pressure reset algorithm is working as intended by adjusting the setpoint away from the maximum and/or minimum setpoints, as well as by the ability of the AHU to satisfy the static pressure setpoint throughout the test.

The test is expected to be generally applicable. It does not rely on a detailed understanding of the implementation of the static pressure reset algorithm. Furthermore, it is conducted in closed-loop through setpoint changes and does not require control signals to be overridden. With the exception of setpoint overrides, all control is provided by the algorithms running in the building automation system (BAS).

2. 5.5.2 System Description

This test was developed for variable-air-volume (VAV) AHUs with static pressure reset. A schematic diagram of the supply fan section of an AHU is shown in Figure 63. VAV AHUs typically use a variable frequency drive (VFD) to modulate the supply fan speed and maintain the supply air static pressure P_{sa} at its setpoint $P_{sa,set}$. A PI or PID controller in the AHU controller uses the static pressure error e_{Psa} to calculate the control signal u_{sf} for the VFD speed.

Figure 63 also illustrates an implementation of a static pressure reset algorithm. There are different reset algorithms used in practice. Here we describe an implementation based on the *Trim and Respond* algorithm because it is well known in the building control industry. As previously stated, the functional test described in a later section is not tied to the reset algorithm that is implemented.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

The purpose of a static pressure reset algorithm is to adjust the static pressure setpoint upward and downward between minimum and maximum allowable values such that there is always one or more VAV box dampers that are nearly fully open. As zone loads change throughout the day, the VAV boxes whose dampers are closest to being fully open are expected to change. If they do not (i.e., if the same VAV box dampers are always the most fully open), this may indicate there is a design or operational fault that could be preventing the reset algorithm from working as intended.

In the implementation illustrated in Figure 63, the control (commanded) signal to each VAV box damper $u_{zdm,i}$ is used as a proxy for position by the reset algorithm to determine which dampers are nearly fully open. Dampers whose control signals indicate they are nearly fully open generate a request for cooling. If there are a sufficient number of requests, the static pressure setpoint is increased (unless it is already at its maximum allowable value) and all dampers are expected to close by a small amount in response. If there is not a sufficient number of requests, the static pressure setpoint is decreased (unless it is already at its minimum allowable value) and all dampers are expected to open by a small amount in response.

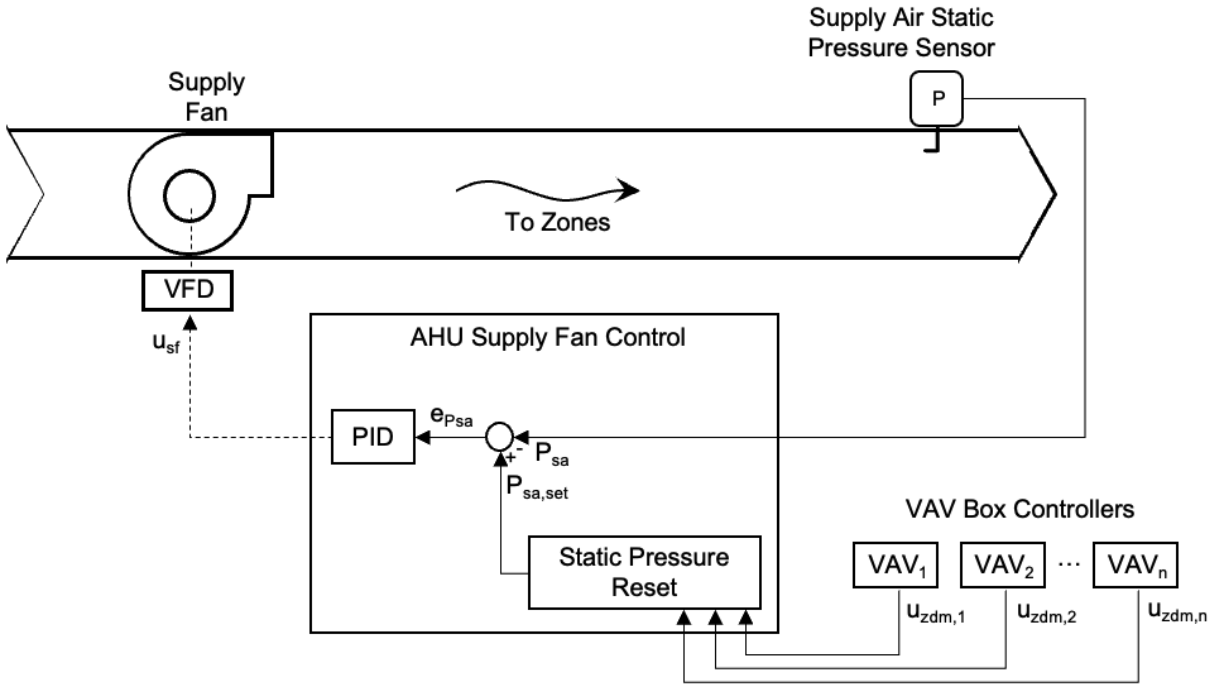


Figure 63: Schematic diagram illustrating VAV AHU supply fan control with static pressure reset.

3. 5.5.3 Process description

Figure 64 shows a process flow diagram for the testing and analysis. The process consists of three modules outlined with dashed lines. The *Proactive Functional Test* module involves

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

verifying that the required conditions for the test are satisfied, and executing the test using overrides of the static pressure setpoint. The *Functional Test Analysis* module uses the test data to evaluate pass-fail criteria associated with different test segments. Finally, the *Functional Test Outcomes* module combines the results from criteria in different test segments to draw conclusions about the success or failure of the test.

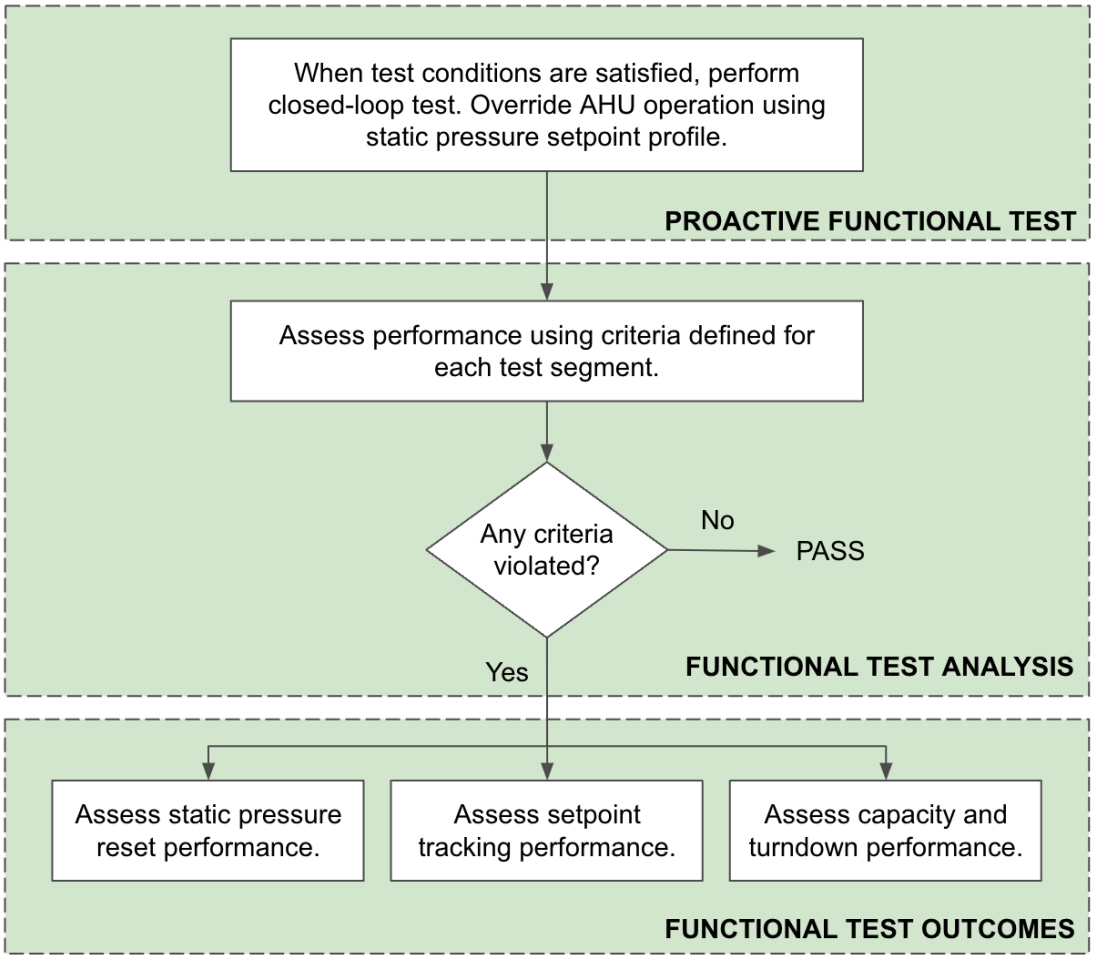


Figure 64: Process flow diagram for the static pressure reset functional test and analysis.

5.5.3.1 Required Data Points

Measurement and control points from the building automation system (BAS) that are used in this test are listed in Table 47. All other variables and parameters used to implement the functional test and analyze its results are listed in Table 48.

Table 47: Required data points from BAS (controllers).

Abbreviation	Data point	Data type	Requirements (Read/Write)
AHU_mode	AHU mode of operation [Occupied or Unoccupied]	Time-series control setting from BAS	Read

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

currentTime	Timestamp for current time	Timestamp from BAS	Read
$f_{za,i}$	airflow rate for the the ith zone or VAV box [CFM]	Time-series sensor reading from BAS	Read
P_{sa}	AHU supply air static pressure [in H_2O]	Time-series sensor reading from BAS	Read
$P_{sa,set}$	AHU supply air static pressure setpoint [in H_2O]	Time-series control setting from BAS and functional test output written to BAS	Read/Write
T_{oa}	outdoor air temperature [°F]	Time-series sensor reading from BAS	Read
$T_{sa,set}$	supply air temperature setpoint [°F]	Time-series control setting from BAS	Read/Write
u_{sf}	supply fan control signal [0-100%]	Time-series control signal from BAS	Read
$u_{zdm,i}$	damper control signal for the ith VAV box where 0 = closed and 1 = open [0 - 1]	Time-series control signal from BAS	Read

Table 48: Variables and parameters used to implement and evaluate the functional test.

Abbreviation	Data point	Data type
C_i	ith Criterion used to evaluate test	Algorithm variable
Ci_count	counter variable for ith Criterion (i = 1, 2, 3, 5, 6, and 7)	Algorithm variable
$C_{i,TH}$	threshold for functional test Criterion i expressed as a percentage of all data points in a test segment [default = 75%]	User-defined parameter
e_{Psa}	supply air pressure error ($P_{sa,set}$ - P_{sa}) [in H_2O]	Algorithm variable
F_{err}	decimal fraction that determines how large the absolute value of of the sample error $e_{Psa,i}$ can be and still be considered “in control” [default = 0.1]	User-defined parameter
$F_{i,TH}$	decimal coefficient that determines the threshold for Criterion i as a fraction of	User-defined parameter

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

	the difference between the setpoint at t = 10 min ($P_{sa,set,pre}$) and either $P_{sa,set,max}$ ($F_{4,TH}$) or $P_{sa,set,min}$ ($F_{8,TH}$) [default = 0.05]	
F_{st}	decimal coefficient that determines the range of allowable values of the static pressure setpoint at the start of the functional test [default = 0.25]	User-defined parameter
i	Iterator variable for data samples in a test segment	Algorithm variable
imax	maximum number of samples in a test segment	Algorithm parameter
$P_{sa,set,PL}$	supply air pressure setpoint priority level	Algorithm variable
$P_{sa,set,fin2}$	final value of the supply air pressure setpoint in test segment 2 [in H_2O]	Algorithm parameter
$P_{sa,set,fin4}$	final value of the supply air pressure setpoint in test segment 4 [in H_2O]	Algorithm parameter
$P_{sa,set,max}$	maximum allowable supply air pressure setpoint during operation [in H_2O]	User-defined parameter
$P_{sa,set,min}$	minimum allowable supply air pressure setpoint during operation [in H_2O]	User-defined parameter
$P_{sa,set,max,st}$	calculated maximum allowable supply air pressure setpoint at the start of the test [in H_2O]	Algorithm parameter
$P_{sa,set,min,st}$	calculated minimum allowable supply air pressure setpoint at the start of the test [in H_2O]	Algorithm parameter
$P_{sa,set,pre}$	supply air pressure setpoint at end of pretest period [in H_2O]	Algorithm parameter
PL_{reset}	priority level of static pressure setpoint from reset algorithm	Algorithm variable
PL_{funct}	priority level of static pressure setpoint from proactive functional test	Algorithm variable
pretestCount	counter for pretest test segment	Algorithm variable

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

pretestEnd	scalar timestamp corresponding to end of the pretest test segment	Algorithm variable
pretestStart	scalar timestamp corresponding to the start of the pretest test segment	Algorithm variable
testSeg	integer variable for labeling test segments [1 to 4]	Algorithm variable
testSegCount	integer variable for counting samples in a test segment	Algorithm variable
testSegEnd	real array with timestamp corresponding to the end of each test segment [1 x 4 array]	Algorithm variable
testSegLabel	integer array with label (“testSeg”) for each test segment [1 x 4 array]	Algorithm variable
testSegStart	real array with timestamp corresponding to the start of each test segment [1 x 4 array]	Algorithm variable
$T_{oa,max}$	maximum allowable outdoor air temperature during testing [°F]	User-defined parameter
$T_{oa,min}$	minimum allowable outdoor air temperature during testing [°F]	User-defined parameter
$T_{sa,const}$	constant value of supply air temperature setpoint during testing [°F]	User-defined parameter

5.5.3.2 Proactive Functional Test

This section provides the steps for checking the test conditions and invoking the test through changes to the static pressure setpoint. This is a closed-loop test. With the exception of overrides of setpoints, all operation is controlled by the closed-loop control logic used for normal operation of the AHU.

1. Check test conditions

Prior to executing the test, the following conditions must be verified:

- It is preferable to run the test during an unoccupied period to reduce the variability of the building load. In this case, it will likely be necessary to override the AHU mode of operation to make it operate in the occupied mode for the period of the test.
- If implemented, the supply air temperature (SAT) reset algorithm must be disabled to avoid interfering with the current test. This can be accomplished by setting the supply air temperature setpoint $T_{sa,set}$ to constant value ($T_{sa,const}$) and assigning a higher priority to

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

this setpoint assignment than that of the SAT reset algorithm.

- It is preferable, but not required, to run the test when the outdoor air temperature T_{oa} is between 50°F and 75°F. Under these conditions it is anticipated that the static pressure setpoint determined by the reset algorithm will be near the middle of the range defined by user-defined minimum ($P_{sa,set,min}$) and maximum ($P_{sa,set,max}$) allowable values.
- The static pressure setpoint at the start of the test must be in a range defined by $P_{sa,set,min,st} \leq P_{sa,set} \leq P_{sa,set,max,st}$ where

$$P_{sa,set,min,st} = P_{sa,set,min} + F_{st} (P_{sa,set,max} - P_{sa,set,min}) \quad (1)$$

and

$$P_{sa,set,max,st} = P_{sa,set,min} + (1 - F_{st}) (P_{sa,set,max} - P_{sa,set,min}) \quad (2)$$

The subscript “st” refers to start and F_{st} is a user-defined decimal coefficient [0.1,0.33] that determines how large the range of startup conditions is for the test. Values of F_{st} up to 0.5 are physically meaningful; however a value greater than 0.33 will create overly restrictive conditions for testing. Values of F_{st} below 0.1 will enable testing even when the static pressure setpoint is near its minimum or maximum value, and this will limit the effectiveness of the test. The recommended default value for F_{st} is 0.25.

Pseudo code verifying that test conditions are satisfied is provided below. Note that the pseudo code requires the outdoor air temperature T_{oa} to be in the range [50°F, 75°F]. This is a recommendation that can be removed.

```
# Assign values to user-defined parameters and thresholds
#   T_sa,const=55°F, T_oa,min=50°F, T_oa,max=75°F, P_sa,set,min=0.1 inH2O,
#   P_sa,set,max=1.5 inH2O, F_st=0.25
# Determine allowable range for P_sa,set at start of test
#   P_sa,set,min,st = P_sa,set,min + F_st (P_sa,set,max - P_sa,set,min)
#   P_sa,set,max,st = P_sa,set,min + (1-F_st)(P_sa,set,max - P_sa,set,min)
# Read T_oa,AHU_mode, and P_sa,set and check if test conditions are
#   satisfied
     # if (AHU_mode == Unoccupied
     #  & T_oa >= T_oa,min & T_oa <= T_oa,max
     #  & P_sa,set >= P_sa,set,min,st & P_sa,set <= P_sa,set,max,st):
           # AHU_mode = Occupied # override AHU operation mode
           # T_sa,set = T_sa,const  # set at highest priority
           # runTest = True
     # else:
           # runTest = False
```

2. Modify supply air static pressure setpoint over four test segments and record test data

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

If *runTest* is “True”, the *Proactive Functional Test* can be executed. The test involves modifying the supply air static pressure setpoint ($P_{sa,set}$) over four 20-minute test segments using the profile shown in Figure 65. Throughout the test, measurement and control points will be collected at uniform time intervals. The test is described assuming a sampling rate of *5 minutes*. Sampling rates of 1-minute or less are preferred where possible because higher sampling rates yield additional insight into the control response. The data to be collected and stored are P_{sa}, $P_{sa,set}$, u_{sf}, T_{oa}, $T_{sa,set}$, $u_{zdm,i}$, and $f_{za,i}$ for all i, where i is the zone index. Other points may also be collected and stored according to the standard monitoring practices at a particular site.

A pretest period begins at time t = 0. Data collected during this period are used to establish the initial operating condition for the AHU supply pressure control loop and the flow control loops for all VAV boxes served by the AHU. The value of the static pressure setpoint at the end of the pretest period is recorded as $P_{sa,set,pre}$.

At t = 10 min, the first test segment begins by overriding the static pressure setpoint to its maximum allowable value, $P_{sa,set,max}$. The priority level for the static pressure setpoint ($P_{sa,set,PL}$) assigned by the functional test (PL_{funct}) must be higher than that of the static pressure reset algorithm (PL_{reset}) in this test segment to assure that the test is executed as intended. For BACnet systems, this means $PL_{funct} < PL_{reset}$ (i.e., lower numbers correspond to a priority level). The setpoint is held constant at $P_{sa,set,max}$ during this first test segment.

At t = 30 min, the setpoint override is released (by setting PL_{funct} = null in BACnet systems) and the reset algorithm is allowed to adjust the setpoint over the second test segment. It is anticipated that $P_{sa,set}$ will be reduced from $P_{sa,set,max}$ over this segment of the test. At t = 50 min, the setpoint is overridden to its minimum allowable value, $P_{sa,set,min}$, with $PL_{funct} < PL_{reset}$. The setpoint is held constant at $P_{sa,set,min}$ during the third test segment, and then at t = 70 min, it is released (by setting PL_{funct} < null) and the reset algorithm is again allowed to adjust its value. In this case it is anticipated that $P_{sa,set}$ will increase from $P_{sa,set,min}$ over the final 20 min segment of the test. The test ends at t = 90 min.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

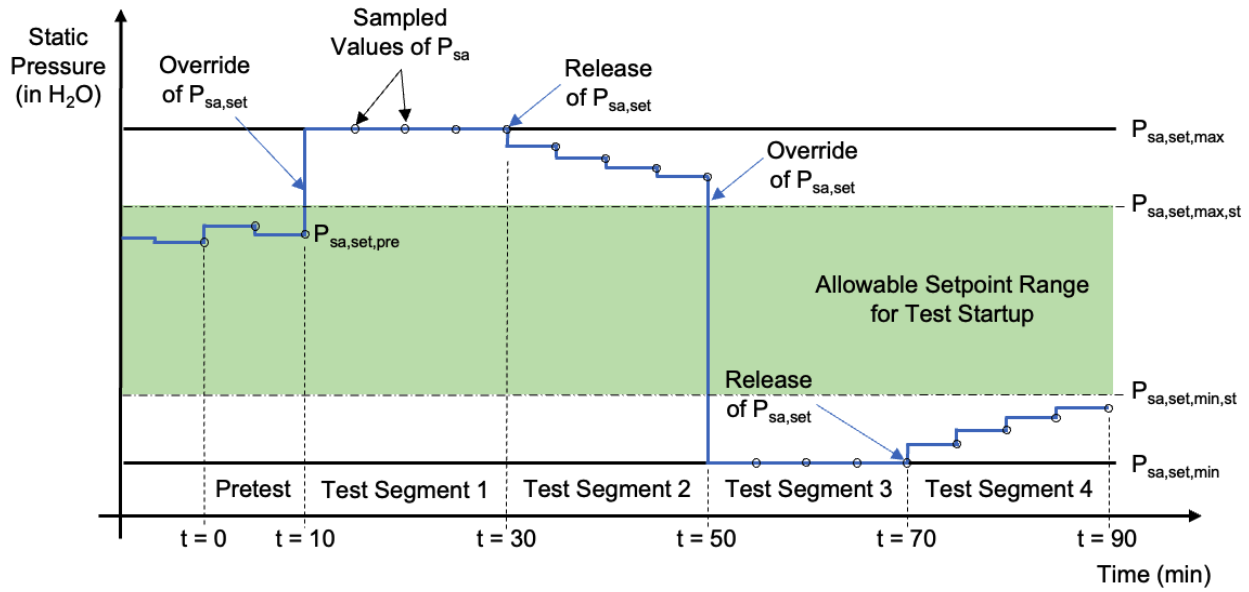


Figure 65: Static pressure setpoint profile (shown in blue) for the *Proactive Functional Test*.

Pseudo code for implementing the *Proactive Functional Test* follows:

```
# If test conditions are satisfied, run the test at 5 minute
intervals #   (illustrative example - other sampling intervals can be
used)
      # if runTest == True:
            # Assign values to user-defined parameters
            #   P_sa,set,min=0.1 inH2O, P_sa,set,max=1.5 inH2O
            # Read PL_reset  # priority level of reset algorithm static
            #                  pressure setpoint assignment
            # pretestCount = 0  # counter for pretest samples
      # while runTest == True: # execute commands at 5 minute
intervals
            # pretestStart = currentTime
            # while pretestCount < 2:
                  # Read and record  P_sa, P_sa,set, u_sf, T_oa, T_sa,set, u_zdm,i,
            and f_za,i for all i
                  #   (first sample at t = 0)
                  # pretestCount += 1
            # Read and record P_sa, P_sa,set, u_sf, T_oa, T_sa,set, u_zdm,i, and f_za,i
            #   for all i (last pretest sample at t = 10 min)
            # pretestEnd = currentTime
            # P_sa,set,pre = P_sa,set  # store setpoint at end of pretest
      period
            # testSeg = 1
            # testSegStart[testSeg] = currentTime
            # testSegLabel[testSeg] = testSeg
            # testSegCount = 0
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
            # while testSegCount < 5:  # each segment runs 20 min
                  # if testSegCount > 0:
                        # Read and record Psa, Psa,set, usf, Toa, Tsa,set,
                        uzdm,i,
                        and fza,i for all i (first
                        #   sample at t = 5 min after test segment
                        #   begins)
                  # if testSegCount == 4:
                        # testSegEnd[testSeg] = currentTime
                        # testSeg +=1
                        # testSegStart[testSeg] = currentTime
                        # testSegCount = 0
                  # if testSeg == 1:
                        # Psa,set = Psa,set,max
                        # PLfunct = PLreset - 1
                  # elif testSeg == 2:
                        # Psa,set = Psa,set,max
                        # PLfunct = null
                        # Psa,set,fin2 = Psa,set
                  # elif testSeg == 3:
                        # Psa,set = Psa,set,min
                        # PLfunct = PLreset - 1
                  # elif testSeg == 4:
                        # Psa,set = Psa,set,min
                        # PLfunct = null
                        # Psa,set,fin4 = Psa,set
                  # Psa,set,PL = PLfunct  # priority level assigned to
Psa,set
                  #                       by the functional test
                  # testSegCount += 1
```

5.5.3.3 Functional Test Analysis

This section presents the analysis of the test results which entails assessing performance using eight criteria (C_1 - C_8) defined in this section. The criteria are summarized in Table 49 and used to: 1) assess if the static pressure reset algorithm is working correctly using criterion C_4 (Test Segment 2) and C_8 (Test Segment 4); 2) assess the performance of the supply air pressure control using criterion C_1, C_3, C_5, and C_7 (Test Segments 1, 3, 5, and 7, respectively); and 3) assess the ability of the supply fan to achieve the maximum and minimum allowable static pressure setpoints using criterion C_2 (Test Segment 1) and C_6 (Test Segment 3).

Table 49: Summary of eight criteria used in the static pressure reset test analysis.

Criterion	Related Test Segment

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

		1	2	3	4
C_1	Assess the ability to maintain P_{sa} at the maximum setpoint	X			
C_2	Assess the ability to reach the maximum setpoint	X			
C_3	Assess the ability to track the changing setpoint		X		
C_4	Assess the reset performance (if the setpoint at the end of the test segment is less than the maximum setpoint)		X		
C_5	Assess the ability to maintain P_{sa} at its minimum setpoint			X	
C_6	Assess the ability to reach the minimum setpoint			X	
C_7	Assess the ability to track the changing setpoint				X
C_8	Assess the reset performance (if the setpoint at the end of the test segment is greater than the minimum setpoint)				X

At each sampling time instance i, the supply air pressure error, $e_{Psa,i}$, is calculated as

$$e_{Psa,i} = P_{sa,set,i} - P_{sa,i} \quad (3)$$

Using 5-minute data sampling, there will be four data points collected during each test segment, the first (i = 1) at t = 5 min after the test segment begins and the last (i = 4) at t = 20 min after the test segment begins. Performance is assessed in each test segment using two criteria based on the supply air pressure error. The criteria for each test segment are defined next.

5.5.3.3.1 Test Segment 1

In Test Segment 1 (t = 10-30 min), the static pressure setpoint is held constant at its maximum allowable value. The first performance criterion C_1 assesses the ability of the AHU to maintain P_{sa} at the maximum setpoint. The criterion is satisfied if the fraction of data samples in the test segment for which the absolute value of $e_{Psa,i}$ is less than or equal to a user-defined fraction of the allowable static pressure setpoint range, meets or exceeds a user-defined threshold. In equation form, criterion C_1 is given by

$$C_1: \ | e_{Psa,i} | \leq F_{err} (P_{sa,set,max} - P_{sa,,set,min}) \text{ for at least } C_{1,TH} \text{ of all samples i (i = 1-4)} \quad (4)$$

where F_{err} is a user-defined decimal coefficient that determines the threshold on individual sample errors as a fraction of the setpoint range (i.e., F_{err} = 0.1 indicates the absolute value of a sample error must be less than or equal to 10% of the allowable setpoint range to be considered “in control”), and $C_{1,TH}$ is the threshold for this criterion expressed as a percentage of all samples (e.g., $C_{1,TH}$ = 75%). This threshold can be adjusted based on field experience, owner requirements, number of data samples, etc.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

The second criterion, C_2, assesses whether the supply fan is able to reach the maximum static pressure setpoint. Criterion C_2 is expressed in equation form as

$$C_2: \ e_{Psa,i} \leq 0 \text{ for at least one i (i = 1-4)} \quad (5)$$

Failure to reach the maximum setpoint may be due to a design error (e.g., fan has insufficient capacity) or an operational fault; however, it may also be that the sampling rate is not sufficient to reproduce the true pressure response and periods of operation above the setpoint may have been missed. Hence, violation of this criterion should be treated as a warning and further investigation of the pressure response should be performed.

Pseudo code for implementing the analysis for Test Segment 1 follows:

```
# Read P_sa,set, P_sa,set,max, P_sa,set,min, and P_sa for all data samples in
#   Test Segment 1
# Set user-defined threshold(s)
     # C_1,TH = 75 %
     # F_err = 0.1
# For each data sample i, compute the supply air static pressure
error
     # e_Psa,i = P_sa,set,i - P_sa,i for all i in Test Segment 1
# Initialize counters
     # C1_count = 0
     # C2_count = 0
     # i = 1
     # imax = 4
     # while i < (imax + 1):
          # if abs(e_Psa,i) < F_err*(P_sa,set,max - P_sa,set,min):
               # C1_count += 1
          # if e_Psa,i <= 0:
               # C2_count += 1
          # i += 1
# Determine if criteria for Test Segment 1 are satisfied
     # if C1_count/imax * 100 ≥ C_1,TH:
          # C_1 = "True"
     # else:
          # C_1 = "False"
     # if C2_count > 0:
          # C_2 = "True"
     # else:
          # C_2 = "False"
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.5.3.3.2 Test Segment 2

In Test Segment 2 (t = 30-50 min), the static pressure setpoint is controlled by the reset algorithm in the BAS. It is anticipated that the setpoint will decrease from the maximum value $P_{sa,set,max}$ enforced in Test Segment 1. Similar to Test Segment 1, the first performance criterion in this test segment assesses the ability of the AHU to track the supply air pressure setpoint. The only difference is that now it is anticipated that the static pressure setpoint will be changing rather than constant. The criterion, C_3, is satisfied if the fraction of data samples in the test segment for which the absolute value of $e_{Psa,i}$ is less than or equal to a user-defined fraction of the allowable static pressure setpoint range, meets or exceeds a user-defined threshold. In equation form, criterion C_3 is given by

$$C_3: \ | e_{Psa,i} | \leq F_{err} (P_{sa,set,max} - P_{sa,set,min}) \text{ for at least } C_{3,TH} \text{ of all samples i (i = 1-4)} \quad (6)$$

The second criterion for this test segment, C_4, assesses whether the static pressure setpoint at the end of the test segment is less than the maximum static pressure setpoint. Criterion C_4 is expressed in equation form as

$$C_4: \ P_{sa,set,max} - P_{sa,set,fin2} > F_{4,TH} (P_{sa,set,max} - P_{sa,set,pre}) \quad (7)$$

where $F_{4,TH}$ is a user-defined decimal coefficient that determines the criterion threshold as a fraction of the difference between the setpoint at t = 10 min ($P_{sa,set,pre}$) and the maximum allowable setpoint ($P_{sa,set,max}$), and the subscript “fin2” indicates that the evaluation of Eq. (7) should be based on the final setpoint sample from Test Segment 2.

Pseudo code for implementing the analysis for Test Segment 2 follows:

```
# Read P_sa,set, P_sa,set,max, P_sa,set,min, and P_sa for all data samples in
#   Test Segment 2
# Assign values to user-defined threshold(s)
      # C_3,TH = 75 %
      # F_err = 0.1
      # F_4,TH = 0.05
# For each data sample i, compute the supply air static pressure
error
      # e_Psa,i = P_sa,set,i - P_sa,i for all i in Test Segment 2
# Initialize counters
      # C3_count = 0
      # i = 1
      # imax = 4
      # while i < (imax + 1):
            # if abs(e_Psa,i) < F_err*(P_sa,set,max - P_sa,set,min):
                  # C3_count += 1
            # i += 1
# Determine if criteria for Test Segment 2 are satisfied
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
# if C3_count/imax * 100 ≥ C3,TH:
      # C3 = "True"
# else:
      # C3 = "False"
# if (Psa,max - Psa,set,fin2) > F4,TH*(Psa,max - Psa,set,pre):
      # C4 = "True"
# else:
      # C4 = "False"
```

5.5.3.3.3 Test Segment 3

In Test Segment 3 (t = 50-70 min), the static pressure setpoint is held constant at its minimum allowable value. The first performance criterion C_5, for this test segment assesses the ability of the AHU to maintain P_{sa} at its minimum setpoint. The criterion is satisfied if the fraction of data samples in the test segment for which the absolute value of $e_{Psa,i}$ is less than or equal to a user-defined fraction of the allowable static pressure setpoint range, meets or exceeds a user-defined threshold. In equation form, criterion C_5 is given by

$$C_5: \ | e_{Psa,i} | \leq F_{err} (P_{sa,set,max} - P_{sa,set,min}) \text{ for at least } C_{5,TH} \text{ of all samples i (i = 1-4)} \quad (8)$$

The second criterion of this test segment, C_6, assesses whether the supply fan is able to reach the minimum static pressure setpoint. Criterion C_6 is expressed in equation form as

$$C_6: \ e_{Psa,i} \geq 0 \text{ for at least one i (i = 1-4)} \quad (9)$$

Failure to reach the minimum setpoint may signal that the supply fan turndown ratio is not sufficient and the minimum fan speed is too high to achieve this setpoint; however, it may also be that the sampling rate is not sufficient to reproduce the true pressure response and periods of operation below the setpoint may have been missed.

Pseudo code for implementing the analysis for Test Segment 3 follows:

```
# Read Psa,set, Psa,set,max, Psa,set,min, and Psa for all data samples in
#   Test Segment 3
# Set user-defined threshold(s)
      # C3,TH = 75 %
      # Ferr = 0.1
# For each data sample i, compute the supply air static pressure
error
      # ePsa,i = Psa,set,i - Psa,i for all i in Test Segment 3
# Initialize counters
      # C5_count = 0
      # C6_count = 0
      # i = 1
      # imax = 4
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      # while i < (imax + 1):
            # if abs(e_Psa,i) < F_err*(P_sa,set,max - P_sa,set,min):
                  # C5_count += 1
            # if e_Psa,i > 0:
                  # C6_count += 1
            # i += 1
# Determine if criteria for Test Segment 3 are satisfied
      # if C5_count/imax * 100 ≥ C_5,TH:
            # C_5 = "True"
      # else:
            # C_5 = "False"
      # if C6_count > 0:
            # C_6 = "True"
      # else:
            # C_6 = "False"
```

5.5.3.3.4 Test Segment 4

In Test Segment 4 (t = 70-90 min), the static pressure setpoint is controlled by the reset algorithm in the BAS. It is anticipated that the setpoint will increase from the minimum value $P_{sa,min}$ enforced in Test Segment 3. As is the case with Test Segment 2, the first performance criterion in Test Segment 4 assesses the ability of the AHU to track a supply air pressure setpoint that is changing. The criterion, C_7, is satisfied if the fraction of data samples in the test segment for which the absolute value of $e_{Psa,i}$ is less than or equal to a user-defined fraction of the allowable static pressure setpoint range, meets or exceeds a user-defined threshold. In equation form, criterion C_7 is given by

$$C_7: \ | e_{Psa,i} | \leq F_{err} (P_{sa,set,max} - P_{sa,set,min}) \text{ for at least } C_{7,TH} \text{ of all samples i (i = 1-4)} \quad (10)$$

The second criterion for this test segment, C_8, assesses whether the static pressure setpoint at the end of the test segment is greater than the minimum static pressure setpoint. Criterion C_8 is expressed in equation form as

$$C_8: \ P_{sa,set,fin4} - P_{sa,set,min} > F_{8,TH} (P_{sa,set,pre} - P_{sa,set,min}) \quad (11)$$

where $F_{8,TH}$ is a user-defined decimal coefficient that determines the criterion threshold as a fraction of the difference between the setpoint at t = 10 min ($P_{sa,set,pre}$) and the minimum allowable setpoint ($P_{sa,set,min}$), and the subscript "fin4" indicates that the evaluation of Eq. (11) should be based on the final setpoint sample from Test Segment 4.

Pseudo code for implementing the analysis for Test Segment 4 follows:

```
# Read P_sa,set, P_sa,set,max, P_sa,set,min, and P_sa for all data samples in
#   Test Segment 4
# Set user-defined threshold(s)
```

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

```
      # C7,TH = 75 %
      # Ferr = 0.1
      # F8,TH = 0.05
# For each data sample i, compute the supply air static pressure
error
      # ePsa,i = Psa,set,i - Psa,i for all i in Test Segment 4
# Initialize counters
      # C7_count = 0
      # i = 1
      # imax = 4
      # while i < (imax + 1):
            # if abs(ePsa,i) < Ferr*(Psa,set,max - Psa,set,min):
                  # C7_count += 1
            # i += 1
# Determine if criteria for Test Segment 4 are satisfied
      # if C7_count/imax * 100 ≥ C7,TH:
            # C7 = "True"
      # else:
            # C7 = "False"
      # if (Psa,set,fin4 - Psa,set,min) > F8,TH*(Psa,set,pre - Psa,set,min):
            # C8 = "True"
      # else:
            # C8 = "False"
```

5.5.3.4 Functional Test Outcomes

Each of the eight criteria (C_1 - C_8) defined for the four test segments can be either "True" or "False". The eight criteria are considered together to assess the overall performance of the AHU static pressure reset and static pressure control algorithms. The criteria are divided according to function as follows:

- Static Pressure Reset Performance - C_4 and C_8 are used to assess whether the static pressure setpoint is being reset from its maximum and minimum values, respectively.
- Setpoint Tracking Performance - C_1, C_3, C_5, and C_7 are used to assess how well the static pressure tracks its setpoint during the test.
- Capacity and Turndown Performance - C_2 and C_6 assess the ability of the supply fan to provide the maximum and minimum allowable static pressure setpoints, respectively.

5.5.3.4.1 Static Pressure Reset Performance

Static pressure reset performance is assessed using criterion C_4 in Test Segment 2 and criterion C_8 in Test Segment 4. Figure 66 contains a flowchart for this assessment. If both C_4 and C_8 are "True", the test indicates the static pressure reset algorithm is functioning as expected. If both C_4 and C_8 are "False", the static pressure reset algorithm is not functioning as expected. This outcome is most likely due to a software implementation error in the controller rather than a physical fault with a component of the fan control system.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

The two other outcomes associated with C_4 and C_8 are more nuanced. When C_4 is “True” and C_8 is “False”, the reset algorithm is resetting from its maximum allowable setpoint, but not from its minimum. This could indicate correct operation if the test is performed when the zone loads are minimal; however, this outcome is not expected if $F_{st} > 0.1$ because restrictions on the static pressure setpoint in relation to its maximum and minimum allowable values would prevent the test from being run under extremely low-load conditions. In general, if the setpoint is frequently near its minimum value, it could indicate there is opportunity to further reduce $P_{sa,set,min}$.

When C_4 is “False” and C_8 is “True”, the reset algorithm is resetting from its minimum allowable setpoint, but not from its maximum. This could indicate correct operation if the test is run during a period of high loads, however, this explanation would be unlikely if $F_{st} > 0.1$ for the reasons explained in the previous paragraph. This outcome could also indicate the presence of a problem. Failure to reset from the maximum setpoint is a common issue in the field and is frequently linked to a “rogue” zone that is persistently requesting more cooling. This should be investigated to determine if there is an operational fault that can be corrected to address this behavior, or if there is an issue with insufficient capacity due to a design error (or a change of use of the zone). In the latter case, the static pressure reset algorithm may have provisions for removing the rogue zone from consideration. The inability to reset from the maximum setpoint could also be caused by a software implementation error in the controller.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

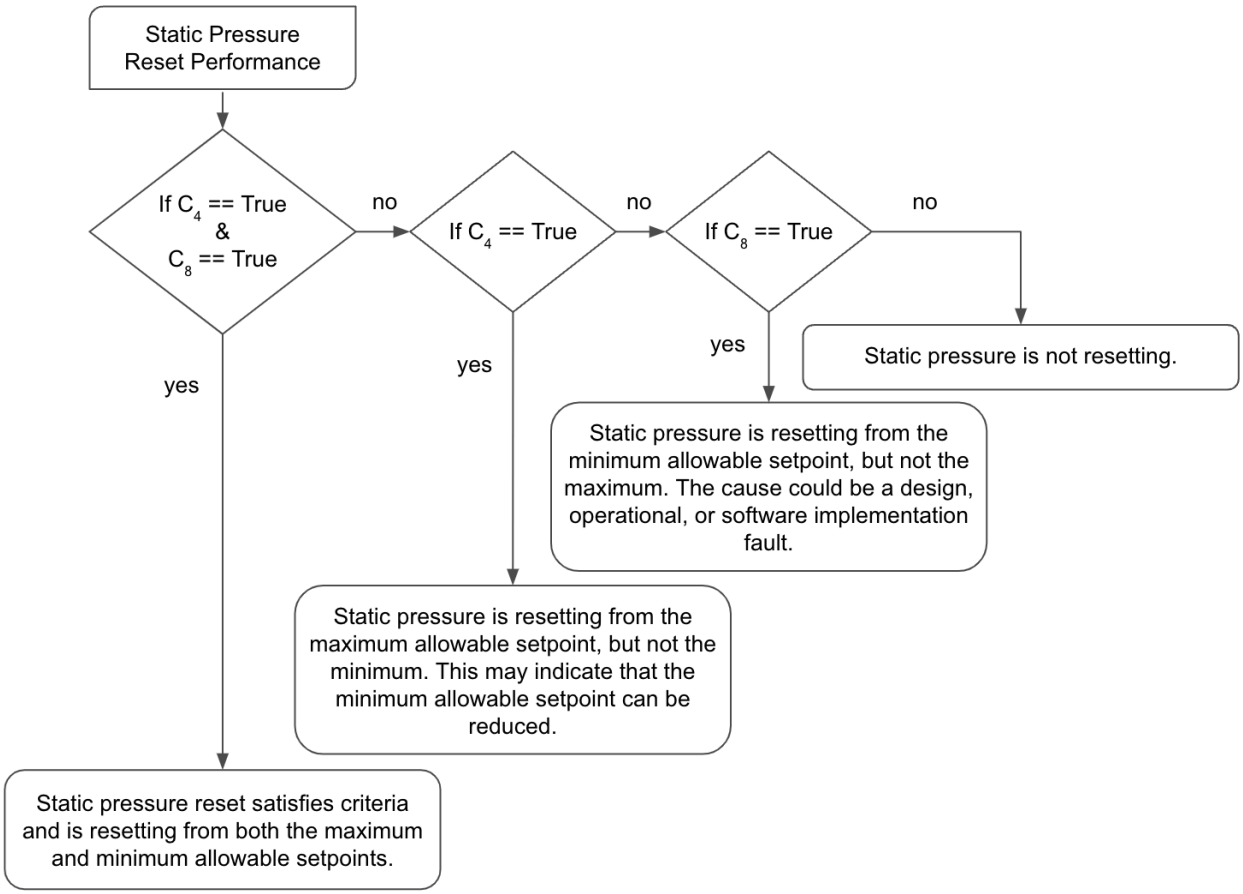


Figure 66: Assessment of static pressure reset performance using criteria C_4 and C_8.

5.5.3.4.2 Setpoint Tracking Performance

Setpoint tracking performance is characterized by criteria C_1, C_3, C_5, and C_7. If two or more of the criteria are violated (this guidance can be adjusted if two violations are considered either too lax or too stringent), the supply air pressure control loop should be further assessed. A first step could be to rerun the analysis after relaxing the thresholds for these criteria. Ultimately it may be necessary to retune this loop to achieve the desired setpoint tracking performance.

5.5.3.4.3 Capacity and Turndown Performance

C_2 and C_6 are used to assess the ability of the supply fan to satisfy the maximum static pressure setpoint and the minimum static pressure setpoint, respectively. The supply fan should have adequate capacity to reach the maximum static pressure setpoint. Similarly, the turndown ratio of the supply fan should be adequate to reach the minimum static pressure setpoint. It is important to understand that periods when the criteria were satisfied may have been missed due to the sampling rate used for data collection. Thus, these criteria are useful performance indicators, but their violation should be interpreted as a warning rather than as a critical issue meriting immediate attention.

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/

5.5.4 Other Considerations

The functional test analysis can be made more granular to yield a more complete understanding of the AHU static pressure reset and static pressure setpoint tracking performance. For instance, multiple thresholds can be used with criteria C_4 and C_8 to pinpoint how far the reset algorithm moves the setpoint from its maximum and minimum allowable values, respectively. Likewise, multiple thresholds can be used with criteria C_1, C_3, C_5, and C_7 to better quantify setpoint tracking. In each case, the performance measured against these different thresholds could be displayed as heat maps, and multiple AHUs could be displayed together to quickly gain an understanding of which systems are working well and which ones are not.

This functional test also provides an opportunity to assess the response of the VAV boxes served by the AHU. For instance, when the static pressure setpoint is overridden to its maximum value, all VAV boxes served by the AHU should respond by closing their dampers to some degree (unless the static pressure setpoint was already at it maximum value, or a VAV box damper was already at its minimum open position). Likewise, when the static pressure setpoint is overridden to its minimum value, all VAV boxes served by the AHU should respond by opening their dampers to some degree (unless the static pressure setpoint was already at its minimum value, or a VAV box damper was already at its maximum open position).

Open Specifications of FFOC Strategies © 2026 by The Regents of the University of California, through Lawrence Berkeley National Laboratory, is licensed under Creative Commons Attribution 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/