
# Software Design Specification

**for**

# MiniThermostat

**Document Version <1.0>**

**Prepared by**

**Group Name: DoePwr**


| | | |
|---|---|---|
| **John Doe** | **0000000** | **doe@mcmaster.ca** |
| **<name>** | **<student #>** | **<e-mail>** |
| **<name>** | **<student #>** | **<e-mail>** |
| **<name>** | **<student #>** | **<e-mail>** |
| **<name>** | **<student #>** | **<e-mail>** |


**Instructor:** Dr. K. Sartipi

**Course:** Software Engineering 3M04/3K04

**Lab Section:** X

**Teaching Assistant:** Jane Doe

**Date:** 01/01/01

# Contents



## List of Figures



## List of Tables

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 1.0 | John Doe | Complete first version of the MiniThermostat Software Design Specification document | 01/01/05 |

# 1 Introduction

This document provides a complete example of the first version of a Software Design Specification document for a small thermostat software system (namely MiniThermostat). This document is primarily based on the developed earlier MiniThermostat SRS document. In the following sections, we specify the purpose of this document, the overview of the MiniThermostat system and the sources used in the production of this document.

## 1.1 Document Purpose

The purpose of this document is to provide a high-level overview of the architecture for the MiniThermostat Software system. In addition, it outlines the different strategies and methods used to produce the most effective architectural structure for the software. This first version of the SDS document shall be used in the future for the development of the more detailed, low-level, design specification.

## 1.2 System Overview

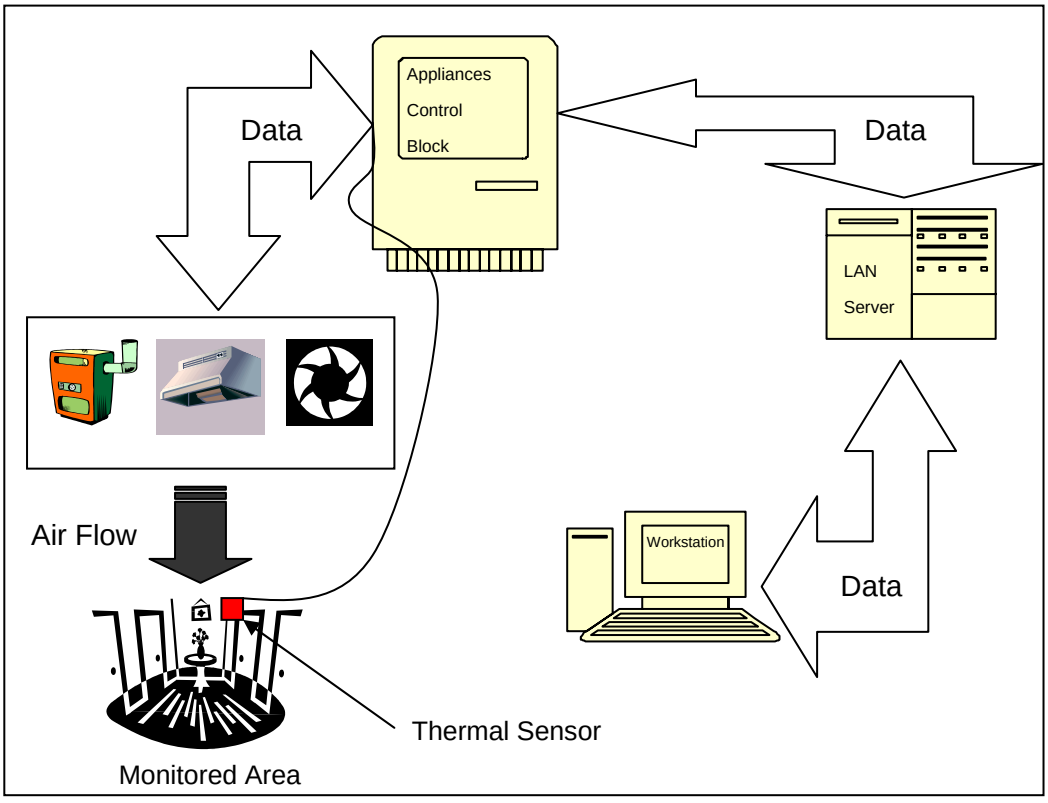


**Figure 1**: *MiniThermostat Environment*

MiniThermostat Software System runs on a single workstation computer. The minimal requirements related to the workstation, shall adhere to the ones specified in section 2.4 (Operating Environment) of the MiniThermostat SRS v. 1.0 document. The system will make a use of an appropriate LAN communication link, to communicate with the ACB. The ACB is programmed to

receive data from a thermal sensor, located in a predefined area, and operate the power relays of three appliances: Furnace, Air Conditioner and a Fan. The system, will communicate with the ACB to operate these appliances and sample the temperature in the monitored area using functions provided by the manufacturer of the ACB in controlblock.lib library. MiniThermostat main purpose is to provide the end-user with the ability to control climate settings in the monitored area, by either changing the different settings directly or through user defined programs. For more details, please refer to MiniThermostat Software Requirements Specification document v. 1.0.

## 1.3 Definitions, Acronyms and Abbreviations

ACB – Appliances Control Block
GUI – Graphical User Interface
LAN – Local Area Network
UI – User Interface

## 1.4 Supporting Materials

**The following standards apply**:

| | |
|---|---|
| J-STD-016-1995 | IEEE/EIA Standard for Information Technology, Software Lifecycle Processes, Software Development, Acquirer-Supplier Agreement |
| IEEE-STD-P1063 | IEEE Standard for Software User Documentation |

**The following texts and documents have been used in the process of developing this document**:

[1] J. Rumbaugh et al. *Object Oriented Modeling & Design*, Upper Saddle River, NJ: Prentice Hall, 1991.

[2] C. Ghezzi et al. *Fundamentals of Software Engineering*. Upper Saddle River, NJ: Prentice Hall, 2003.

[3] J. Doe, *MiniThermostat Software Requirements Specification*, 2005.

## 1.5 Document Overview

The next section of the MiniThermostat SDS v. 1.0 provides the architectural view of the system. It illustrates in detail how the system was decomposed into several main components and what were the reasons for this particular decomposition. Subsections of section two, describe in detail the components and their corresponding interfaces. Finally, section three of the document, provides a State Chart view of the MiniThermostat system and explanations regarding the different state transitions.

# 2 Architecture

This section provides the architecture design of the MiniThermostat software system. It includes the first version of the system Component Diagram which illustrates the different components, their interfaces and the dependencies between components. The main principles that led to the proposed design are:

- Anticipation of change
- Maintainability
- Separation of concerns
- Understandability

Section 2.2 discusses in detail the different components and how they adhere to the listed above principles. The next section provides the Component Diagram and the details about the proposed architecture.

## 2.1 Overview

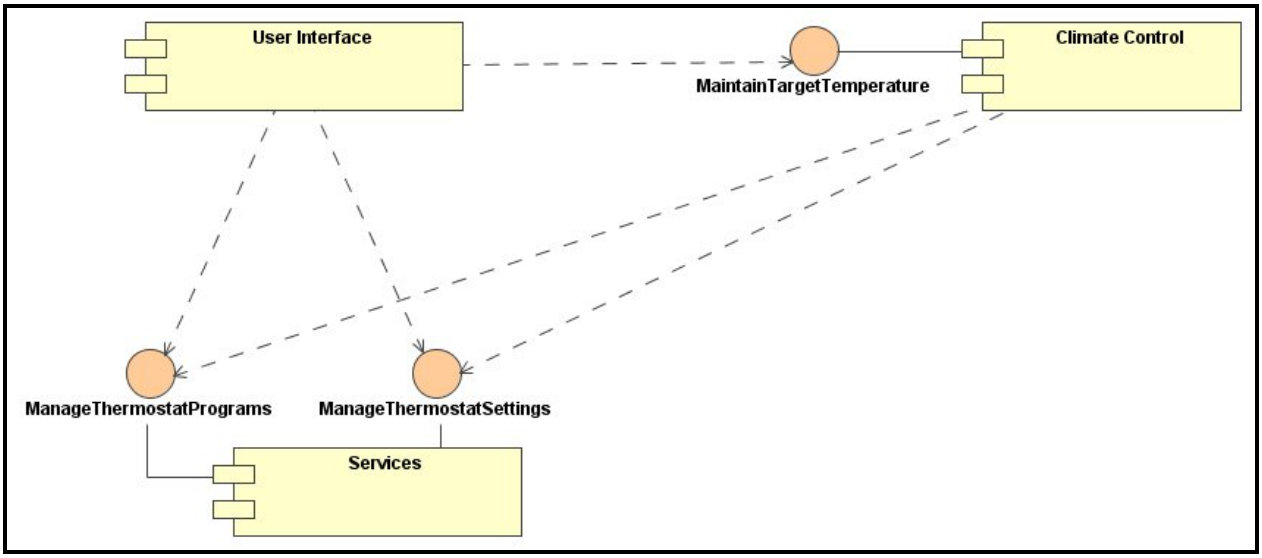


**Figure 2:** *MiniThermostat Component Diagram*

As illustrated in Figure 2, MiniThermostat architecture consists of three main components:

1. User Interface
2. Climate Control
3. Services

In order to make the architecture more understandable, maintainable and adoptable for changes it was decomposed according to the different functional areas the system covers. All three components cover specific functional areas, as specified in the MiniThermostat SRS document, and each can be easily modified without affecting the others. By decomposing the system, to only three main components, and providing clear and simple interfaces for each component, the architecture becomes more understandable and the system is easier to maintain.

## 2.2 User Interface

User Interface (UI) component provides the end-user with graphical interface, and sequences the different functions provided by the other components. UI does not provide any interfaces to other components and has three dependencies. UI depends on:

1. **ManageThermostatPrograms** interface provided by the *Services* component. Through this interface, UI provides the end-user with the ability to view and modify eight different programs. Every program includes the weekday and daytime when it is to be executed, and the target temperature the system should maintain.
2. **ManageThermostatSettings** interface provided by the *Services* component. Through this interface, UI provides the end-user with the ability to modify the different MiniThermostat settings.
3. **MaintainTargetTemperature** interface provided by the *Climate Control* component. Through this interface, the main function of monitoring and controlling a climate in a predefined area can be achieved for the MiniThermostat system.

By restricting the functions of sequencing and interacting with the end-user, to this specific component, the system is more maintainable in a sense that every problem related to graphical interface can be traced directly, and exclusively to this component. In addition this component completely covers the *User Interface* functional area specified in MiniThermostat SRS, thus contributing to architecture understandability.

## 2.3 Services

The services provided by the MiniThermostat system to the end-user are:

1. View, modify and save eight different programs.
2. Change selected, user defined, settings (for details refer to MiniThermostat SRS).

Services component handles data storage and data retrieval tasks corresponding to the different MiniThermostat services. Services component does not depend on any other component of the system and provides two interfaces for other components. These interfaces are:

1. **ManageThermostatPrograms** interface is being used by two components – *UI* and *Climate Control*. This interface allows for a two way (store/retrieve) data exchange. *UI* uses this interface to retrieve, change and store the different programs. *Climate Control* uses this interface to retrieve the data needed to perform its functions. The services provided by this interface are:
    - SetThermostatPrograms
    - SavePrograms
    - GetThermostatPrograms
2. **ManageThermostatSettings** interface is being used by the same two components mentioned above. Once again, *UI* uses this interface to exchange data both ways, whereas *Climate Control* will only use this interface to retrieve the current settings. The services provided by this interface are:
    - SetThermostatSettings
    - SaveSettings
    - GetThermostatSettings

As with *UI*, the restrictions of data handling tasks to one specific component aids in error tracing and maintenance. Any error related to data handling and storage, or to one of the services can be traced directly to this component. In addition, this restriction adds to the understandability of the system since the component responsibilities cover only one functional area.

## 2.4 Climate Control

Climate Control component covers the responsibilities of obtaining the current target temperature and maintaining it. This is the only component that interfaces with controlblock.lib library. The functions from this library are used to operate the different appliances (the complete list of the functions can be found in Appendix A). Climate Control depends on two interfaces provided by the *Services* component and provides only one interface. Climate Control depends on:

1. **ManageThermostatPrograms** interface provided by the *Services* component. Climate Control uses this interface to obtain information related to the user specified programs. The specific service provided by this interface is *GetThermostatPrograms.*
2. **ManageThermostatSettings** interface provided by the *Services* component. Climate Control uses this interface to obtain information related to the different MiniThermostat settings. The specific service provided by this interface is *GetThermostatSettings.*

The interface provided by the Climate Control component is **MaintainTargetTemperature**. UI depends on this interface to provide the system with the ability to monitor the current temperature in the area and maintain the target temperature.

# 3 High Level Design

The High-Level Design section describes in further detail the interactions between the system components and their corresponding interfaces. To illustrate the dynamic behaviour of the MiniThermostat system, State Chart view has been used. State Charts model the behaviour from the perspective of a single entity. The entity that is modelled in Figure 2 is the complete MiniThermostat system.

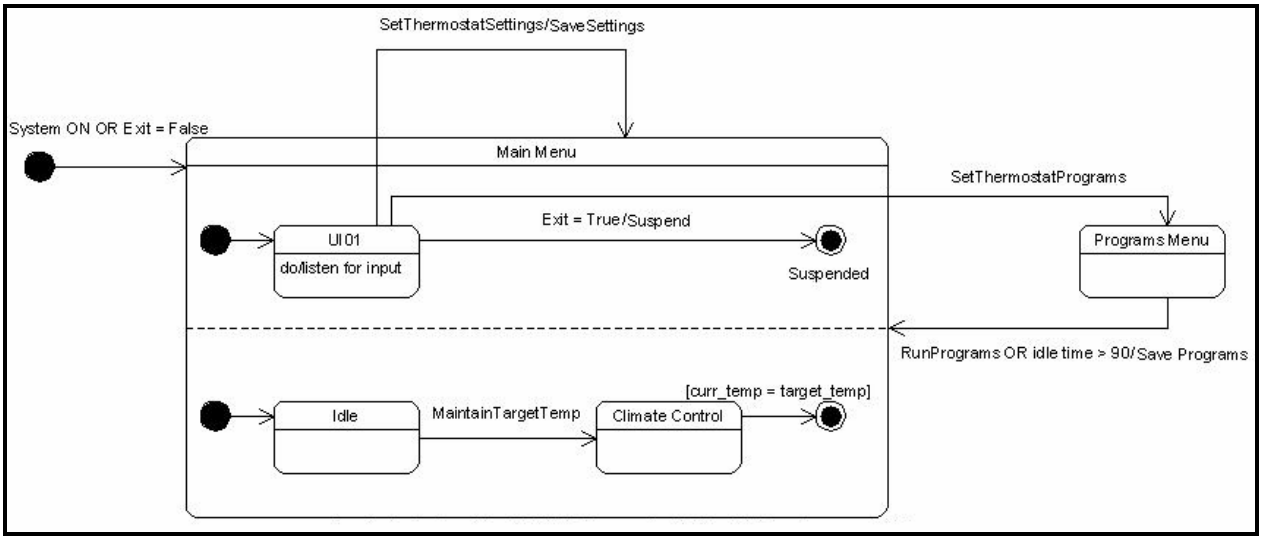


**Figure 3***: State Chart View of MiniThermostat System*

Two events will trigger the transition of the system from the initial state to the concurrent state *Main Menu*. The first event is the event of the system being turned ON (application started). The second event is for the case where the system is already ON, but is suspended. Since, even when suspended, the system responds to selected user inputs, the input that will generate the event *Exit = False*, will start-up the system.

The system starts by providing the end-user with the Main Menu interface and than waits for user inputs. The system performs the task of *Climate Control* concurrently. Initially the system is in the idle state, and the event *MaintainTargetTemp* triggers the transition to *Climate Control*. Every time, the end-user makes an input, this input is being checked for validity. Valid inputs will trigger events that will indicate whether a *Setting* or a *Program* related event occurred. For the case, when a setting is being changed *UI* uses the *ManageThermostatSetting* interface provided by the *Services* component. *SetThemrostatSettings* is the service that triggers the transition back to the composite state *Main Menu*. This transition is necessary, so that the changed settings will be in effect. Before returning to the concurrent state, the action of *Save Settings* is executed.

The same logic applies for the case where a programs related change occurred (i.e., *View Programs* button pressed in UI01). *UI* uses the *ManageThermostatPrograms* interface and the service that triggers the transition to *Programs Menu* state is *SetThermostatPrograms.* Transition back to the concurrent state *Main Menu* occurs when the end-user decides to run the modified programs. At this time, once again the transition to the concurrent state is necessary in order for the changes to be in effect. Before returning to the *Main Menu* state, the action of *Save Programs* is executed. This service is also provided through the *ManageThermostatPrograms* interface.

With the occurrence of event *Exit* = *True*, the system executes the action *Suspend* and enters the final state *Suspended*. Note that the system will still be in the concurrent state of *Climate Control.*

# Appendix A – controlblock.lib Functions

The following functions were provided in the library controlblock.lib by the manufacturer of the ACB:

**Table 1**: *ACB functions*

| **Function:** | **Description:** |
|---|---|
| int GetTemp(void) | Returns then current temperature in the monitored area. The temperature is returned in degrees Celsius and is rounded up to the closest integer. |
| void RunFurnace(void) | Powers ON the Furnace power relay |
| void RunAC(void) | Powers ON the Air Condictioner power relay |
| void RunFan(void) | Powers ON the Fan power relay |
| void StopFurnace(void) | Powers OFF the Furnace power relay |
| void StopAC(void) | Powers OFF the Air Conditioner power relay |
| void StopFan(void) | Powers OFF the Fan power relay |
| int GetAppliancesStatus(int appliance) | Given a valid input (1\|2\|3), where:<br>Furnace == 1<br>AC == 2<br>Fan == 3<br>This function will return a 1 if the appliance is ON and 0 if the appliance is OFF. For any other input the return value will be 2. |