Shizuku 2

(VRF System Emulator)

Reference Manual

2023/11/23

Table of contents

Section 1 Introduction

1.1 What is Shizuku2

Shizuku2 is a software that emulates the thermal environmental system of a building with a variable refrigerant flow (VRF) system installed (hereafter, referred to as an "emulator").

More buildings are installing VRF systems for air conditioning, and there is great value in correctly predicting their performance. However, VRF systems are more difficult to predict than central heat-source systems because of the greater interaction between the air-conditioning system and occupants. This is mainly because occupants can directly control the remote controller to alter the indoor environment. Another factor that makes it difficult to predict the performance is that the heat flow is difficult to measure accurately because of the direct heat exchange between the refrigerant and air.

Therefore, this emulator was developed to predict the effect of various VRF controls on energy consumption and thermal comfort. The building, VRF system, and occupants are each modeled precisely to simulate reality and correctly evaluate the tradeoffs between these two performances. Users of the emulator can attempt to control the VRF as if it exists in reality using BACnet—a general-purpose communication method that is also used in real buildings.

This document provides a reference manual on how to use an emulator. The subsequent sections of this chapter describe the building, VRFs, and occupants to be simulated. Section 2 discusses the installation of Shizuku2, its directory structure, and a simple execution example. Section 3 explains how to control the VRF system in the emulator using Microsoft Excel. Section 4 explains how to control the VRF using a different program. Section 5 lists points to consider when optimizing VRF operations.

1.2 Thermal environment system to be emulated

1) Building

The floor plan of the building to be simulated is shown in Fig. 1.1. Two offices face northwest and southwest. Each office is occupied by a different tenant. Both have floor areas of 273 m^2. There are no detailed partitions.

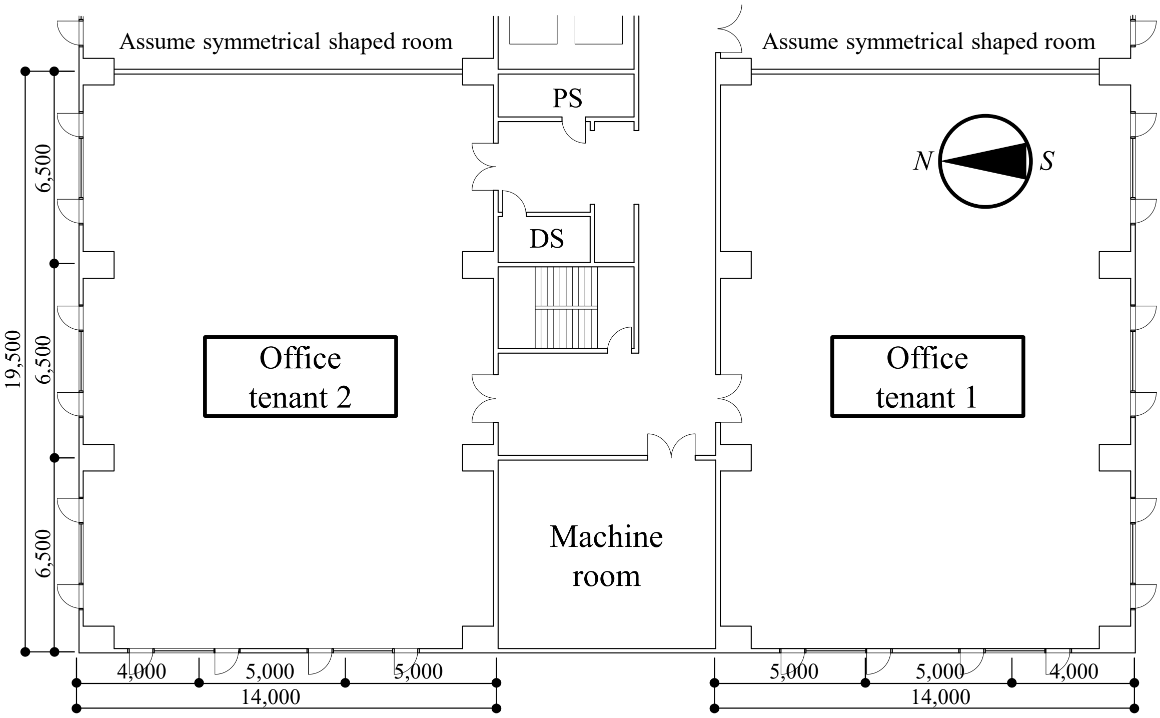


Fig. 1.1 Floor plan of the building

A cross-sectional view of the exterior wall is shown in Fig. 1.2. The total window area is 15.96 m^2 on the south and north sides and 10.64 m^2 on the west side.

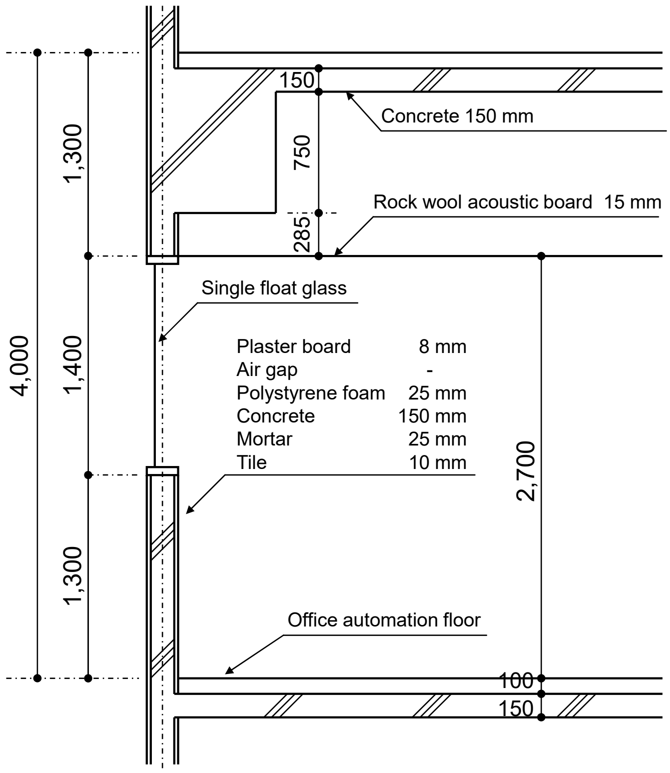


Fig. 1.2 Cross-sectional view of the exterior wall

We assume that the building would be constructed in Tokyo, Japan. The typical summer and winter weather data for Tokyo are shown in Fig. 1.3.

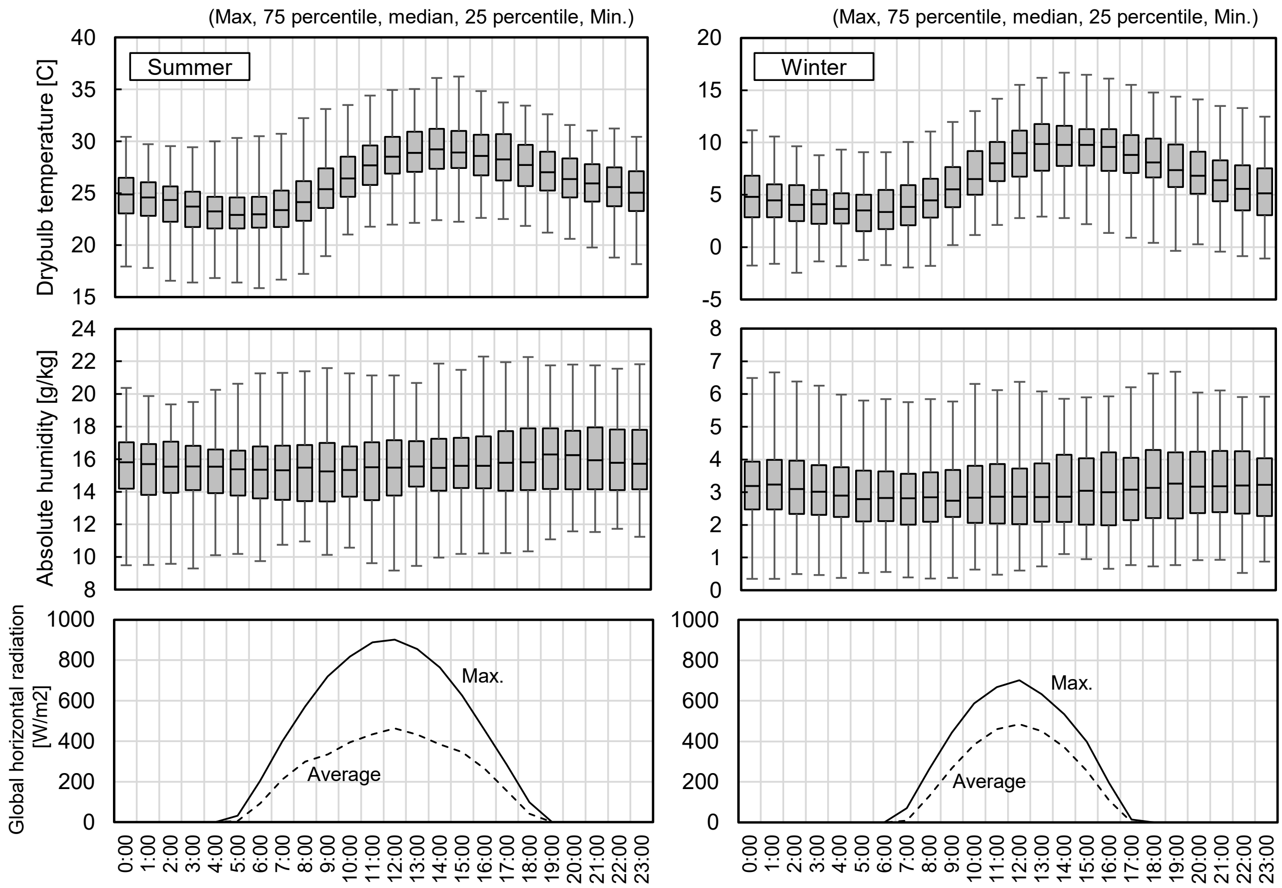


Fig. 1.3 Typical summer and winter weather data for Tokyo

The emulator simulates one week starting July 20 as the summer season and one week starting February 10 as the winter season. Fig. 1.3 shows the results of generating 100 random weather data points for July 20 and February 10, the first day of each simulation period, and obtaining their statistics.

2) VRF system

Four VRF systems exist: one for interior air conditioning and one for perimeter air conditioning in each of the north and south office rooms. Fig. 1.4 shows the zones for each indoor-unit air condition. Each zone has a small total heat exchanger for ventilation.

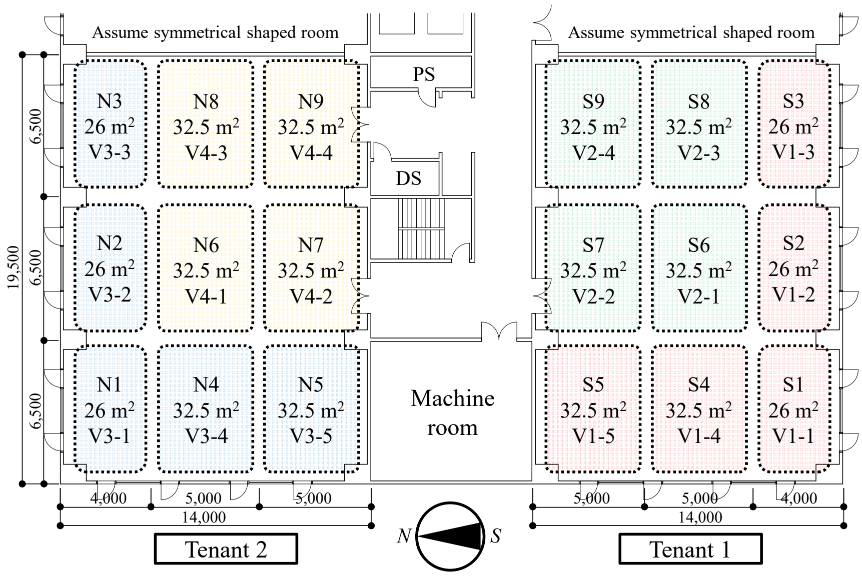


Fig. 1.4 Air-conditioning zone of the indoor unit

Table 1.1 shows the specifications of the outdoor units. All models are two-pipe systems without heat recovery. Table 1.2 and Table 1.3 show the specifications of the indoor units in each zone.

Table 1.1 Outdoor unit specifications

-	VRF1	VRF2	VRF3	VRF4
Cooling capacity [kW]	40.0	22.4	33.5	22.4
Cooling electricity [kW]	12.5	6.07	9.74	6.07
Heating capacity [kW]	45.0	25.0	37.5	25.0
Heating electricity [kW]	13.1	6.32	10.0	6.32
Air flow rate [m^3/min]	210	218	187	218
Electricity [kW]	0.58	0.52	0.42	0.52

Table 1.2 Indoor unit specifications

Indoor unit type	C56	C71
Nominal cooling capacity [kW]	5.6	7.1
Nominal heating capacity [kW]	6.3	8.0
Air flow rate [m^3/min]	15.5	22.0
Electricity [kW]	0.043	0.072

Table 1.3 Type of indoor units in each zone

Zone name	N1	N2	N3	N4	N5	N6	N7	N8	N9
I/U name	V3-1	V3-2	V3-3	V3-4	V3-5	V4-1	V4-2	V4-3	V4-4
I/U type	C71	C56	C56	C71	C71	C56	C56	C56	C56
Zone name	S1	S2	S3	S4	S5	S6	S7	S8	S9
I/U name	V1-1	V1-2	V1-3	V1-4	V1-5	V2-1	V2-2	V2-3	V2-4
I/U type	C71	C71	C71	C71	C71	C56	C56	C56	C56

3) Occupants

There are approximately 80 occupants in the office, although the number varies depending on the random seeding. Each occupant is modeled separately and has a different behavioral pattern and thermal preference. A list of the occupants is presented in Appendix 2.

Fig. 1.5 shows the number of office workers in a given week. The number of occupants in the office changes daily because the manner in which each occupant enters and leaves the office is determined stochastically. Some occupants work overtime and stay overnight, whereas others work on weekends.

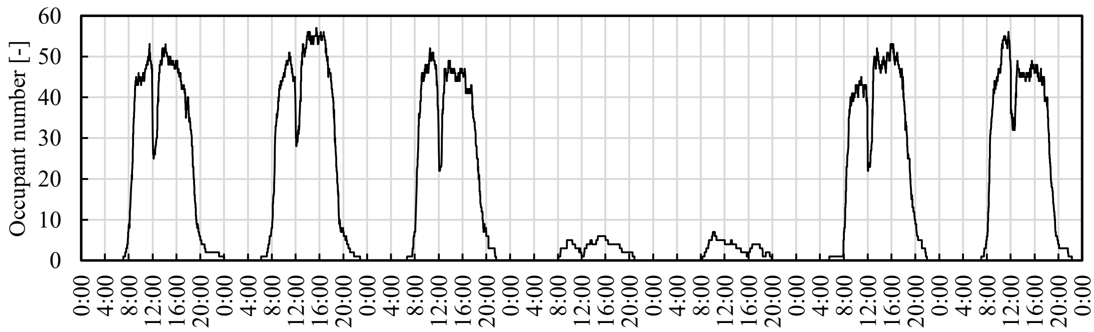


Fig. 1.5 Change in the number of weekly office workers

Office workers expressed probable dissatisfaction depending on the indoor environmental conditions. The four conditions are as follows:

1) Thermal environment is too hot or too cold.
2) Cold air directly contacts the body
3) Large temperature distribution in the vertical direction
4) Insufficient ventilation and dirty air.

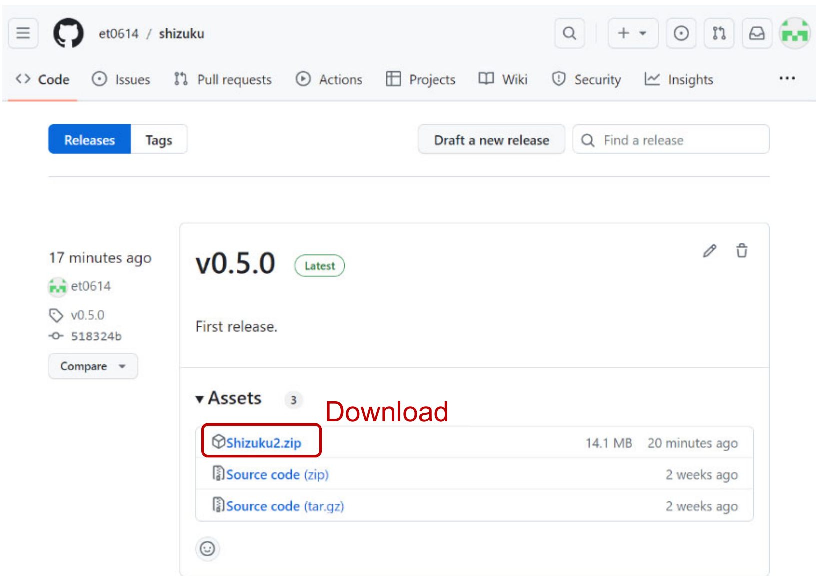
These environmental conditions vary depending on the operation of the VRF.

Section 2　　Installing and running the emulator

2.1　Installing the emulator

Download the latest software compressed file (Shizuku2.zip) from the following web site.

https://github.com/et0614/shizuku/releases



.NET 6.0, or higher, is required to execute the emulator. Download and install them from the following websites:

https://dotnet.microsoft.com/download

2.2 Contents of the directory

By unzipping the downloaded compressed file, you will see the directory shown in Fig. 2.1.

```
Shizuku2
    ├─ Shizuku2.exe                  (1)
    ├─ setting.ini                   (2)
    ├─ data (Directory)              (3)
    ├─ ExcelController.exe           (4)
    ├─ schedule.xlsx                 (4a)
    ├─ schedule_samples.xlsx         (4b)
    ├─ CaseStudyProcessor.exe        (4c)
    ├─ schedules (Directory)         (4d)
    ├─ DummyDeivceController.exe     (5)
    ├─ Libraries                     (6)
    └─ Other files
```

Fig. 2.1 Shizuku2 directory

“(1) Shizuku2.exe" is an executable emulator.

"(2) setting.ini" is the initial configuration file for changing the behavior of the emulator.

“(3) data" is the directory to which the results of the emulation are written.

The VRF in the emulator is controlled externally using BACnet communication. The easiest method is to use "(4) ExcelController.exe,” which reads the HVAC operation schedule entered in a Microsoft Excel file and controls the VRF while keeping it synchronized with the emulator. 4a–d show the related files and directories, respectively. The details are explained in Section 3.

“(5) DummyDeviceController.exe" is a sample program for testing BACnet communication using a dummy BACnet Device prepared in an emulator, which is described in the next section.

“(6) Libraries" is a directory containing program libraries used to communicate with the emulator in the python or C# languages.

2.3 Starting the emulator and testing BACnet communication

When Shizuku2.exe is executed, the startup screen shown in Fig. 2.2 appears.

The emulator contains models of the VRF and ventilation equipment, but they stop when the emulator starts up and will not move unless a startup signal is sent from the outside via BACnet communication. This equipment, which is controlled by BACnet communication, is called a BACnet controller.

To allow time for the BACnet controller outside the emulator to connect to the internal controller, the emulator enters an idle state once it starts and completes its preparatory calculations. Fig. 2.2 shows this state. Entering the "Enter" key on the keyboard brings the program out of its idle state and starts the calculation.

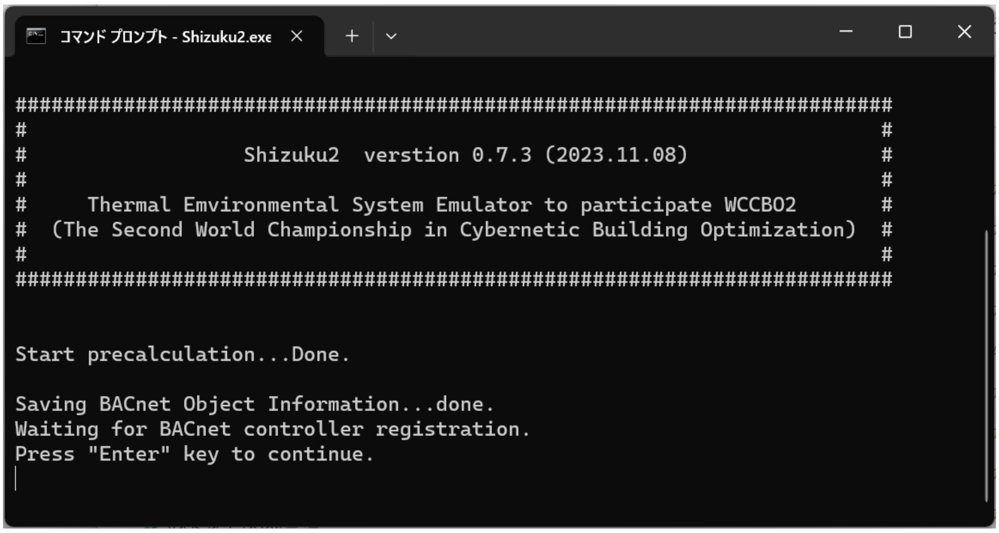


Fig. 2.2 Emulator startup screen

The emulator can respond to BACnet communication even in the idling state, as shown in Fig. 2.2. A dummy BACnet device is provided in the emulator to test whether BACnet communication can be performed normally. To communicate with this dummy device, start "DummyDeivceController.exe".

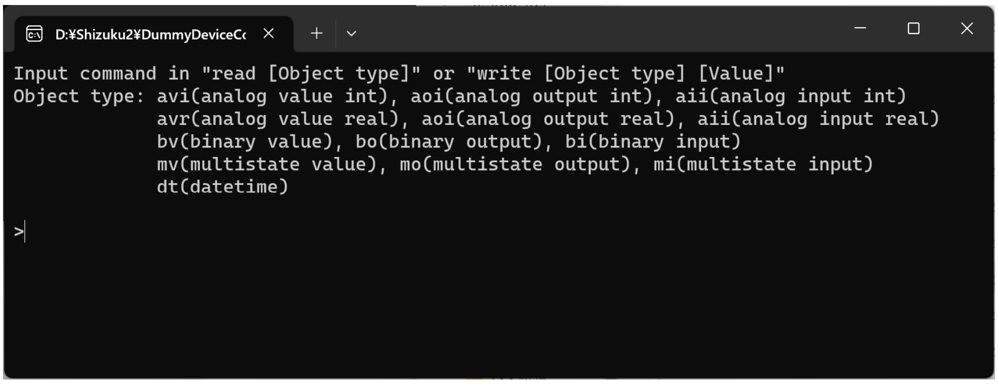


Fig. 2.3 Dummy device controller

Various state values inside the emulator are managed as BACnet objects. Appendix 1 provides a list of BACnet objects managed by the emulator.

The typical BACnet object types and their uses are listed in Table 2.1. The analog value, input, and output are objects for managing numeric values, such as integers and real numbers. The binary value, input, and output are the objects for managing Boolean values. Multistate values, inputs, and outputs are the objects for managing discrete integer values. The BACnet date time is an object for managing the date and time.

The value or output can be rewritten from outside the emulator and is primarily used to control the equipment, whereas the input is read only and primarily used to monitor the system status.

Table 2.1 Value and use example of object types

Object types	Value	Use example
Analog value, output	integer or real	Setting setpoint temperature of indoor unit
Analog input	integer or real	Monitor room temperature
Binary value, output	Boolean	Setting on/off status of VRF
Binary input	Boolean	Monitor on/off status of VRF
Multistate value, output	unsigned integer	Setting fan speed of indoor unit
Multistate input	unsigned integer	Monitor air flow direction of indoor unit
BACnet date time	date and time	Get current date and time in the emulator

One of these types of BACnet objects is provided in the dummy device of the emulator. For example, let us consider the value of an integer-type analog. If we type "read avi" in the console and hit the Enter key, we obtain the output shown in Fig. 2.4, which reads "1" as the current state value.

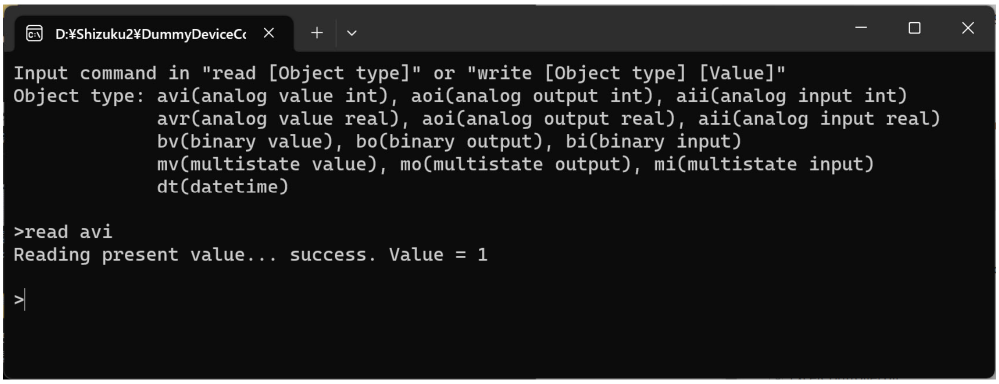


Fig. 2.4 Reading analog value (integer) from the emulator

The emulator screen displays a request to read the properties of the emulator as shown in Fig. 2.5. This status display is enabled only for dummy devices to test BACnet communication.

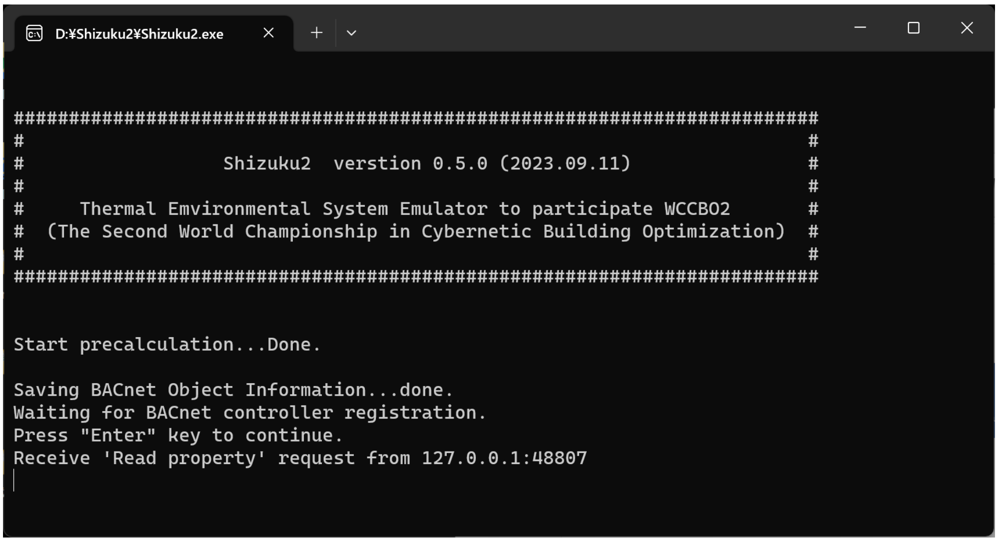


Fig. 2.5 Response of the emulator

The analog value can also be rewritten. Type "write avi 5" and press the Enter key to overwrite the value with “5”. If you input the "read avi" command again, the value will be overwritten with “5”, as shown in Fig. 2.6.

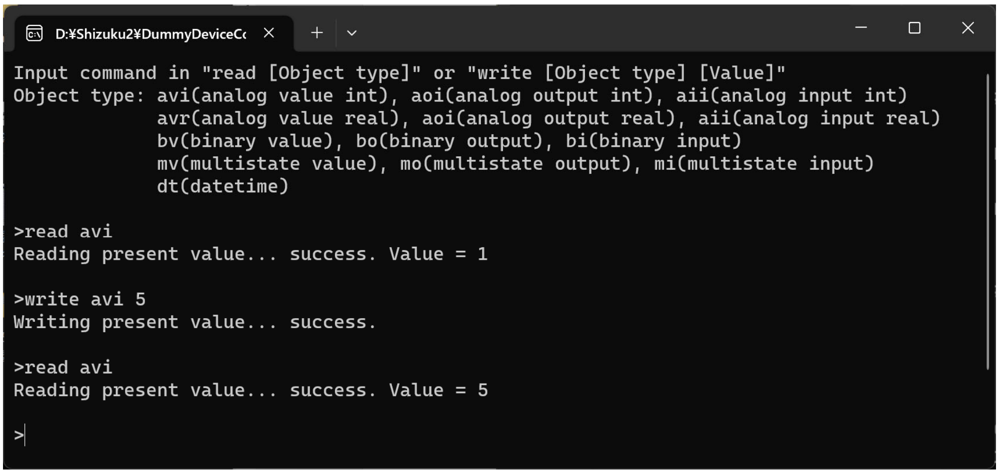


Fig. 2.6 Overwriting the analog value of the emulator

The *DummyDeivceController* is launched in a window separate from the emulator. This means that the emulator is operated via BACnet communication by an externally provided control system and not by the control system provided by the emulator.

Therefore, the user is free to decide which algorithm to use to control the VRF. One can even write a control program in one’s preferred language; it can even be distributed among several autonomous small control programs. These mechanisms are identical to those used in actual buildings.

2.4 Running the emulator

Entering the Enter key in the emulator window begins the simulation, as shown in Fig. 2.7.

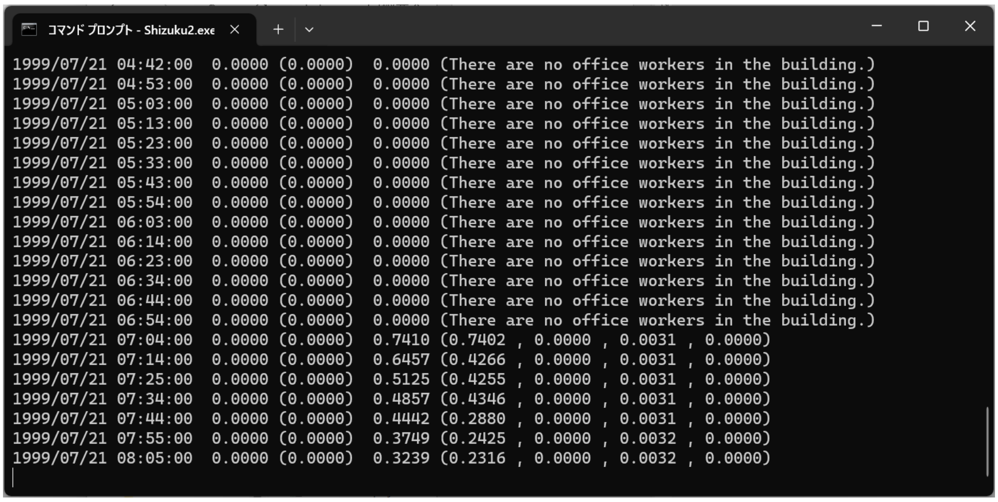


Fig. 2.7 Starting the simulation

By default, the acceleration is set at 600×. The emulator simulates a week in the summer or winter. Every second, the emulator advances 600 s; therefore, the calculation takes approximately 17 min.

During the calculation, the date and time are followed by seven numbers.

The two numbers on the left are energy related: the first is the total energy consumption [GJ], and the second, in parentheses, is the instantaneous energy consumption [GJ/h]. By default, the VRF and ventilation systems were stopped; therefore, zero continued to be displayed.

The five numbers on the right are comfort related: the first is the average dissatisfaction rate [-], and the four in parentheses are the instantaneous dissatisfaction rates from left to right: dissatisfaction rate due to thermal preference, cold air drafts, vertical temperature difference, and air contamination. The instantaneous dissatisfaction rate is displayed only when occupants are present in the building.

When the calculation is finished, the result is written under the "data" directory as shown in Fig. 2.8.

```
├─ data                  (3)
     ├ general.csv       (3a)
     ├ occupant.csv      (3b)
     ├ vent.csv          (3c)
     ├ vrf.csv           (3d)
     ├ zone.csv          (3e)
     ├ result.txt        (3f)
     └ result.szk        (3g)
```

Fig. 2.8 data directory

General information such as outdoor air conditions, energy consumption, and occupant dissatisfaction rate are written in "(3a) general.csv." "(3b) occupant.csv" contains information such as the temperature and thermal sensation reported by the occupants and the amount of clothing worn. "(3d) vent.csv" contains the CO2 level in the room and energy consumption of the ventilation system. “(3d) vrf.csv” contains the energy consumption of the VRF system and its operation status. “(3e) zone.csv” contains the temperature and humidity of the room. The calculation conditions and results are written in “(3f) result.txt”. The result is also written to the file "(3g) result.szk" in an encrypted format.

2.5 Setting emulation parameters

To change the calculation conditions, change the parameters of "setting.ini. The contents are shown in Fig. 2.9.

```
use_rso=1;              //Use random seed for determine occupants' behavior or not. (0:false, 1:true)
rseed_obhv=1;           //Random seed for determine occupants' behaviour randomly.
use_rsw=1;              //Use random seed for generating weather data or not. (0:false, 1:true)
rseed_w=1;              //Random seed for generating weather data.
rseed_oprm=1;           //Random seed for generating parameters of occupants' behaviour model.
timestep=60;            //Time step[sec] (0~3600)
scheduller=0;           //VRF scheduller enabled (0:disabled, 1:enabled)
controller=0;           //VRF controller type (0:Original, 1:Daikin, 2:Mitubishi Electric, 3:Toshiba, 4:Hitachi, 5:Panasonic)
weather=3;              //Weather data type (0:Load csv file, 1:Sapporo, 2:Sendai, 3:Tokyo, 4:Osaka, 5:Fukuoka, 6:Naha)
period=0;               //Simulation period (0:Summer, 1:Winter)
accelerationRate=600; //Default acceleration rate (1~)
userid=0;               //Unique ID to identify results data file
outputSpan=60;          //Time interval[sec] outputing results.
```

Fig. 2.9 Initialization file

The most important parameters are “period” and “accelerationRate”.

The “period” parameter changes the period of time for the simulation: 0 for one week in summer and 1 for one week in winter.

The " acceleration rate was the acceleration of the calculation. By default, it is set at 600, but it can be set to a larger value if the computer has a high capability. Conversely, if the computer is incapable of performing the calculation at the specified rate, "DELAYED" will be displayed, as shown in Fig. 2.10. If this display persists, the emulator will not be synchronized, and the acceleration must be reduced.

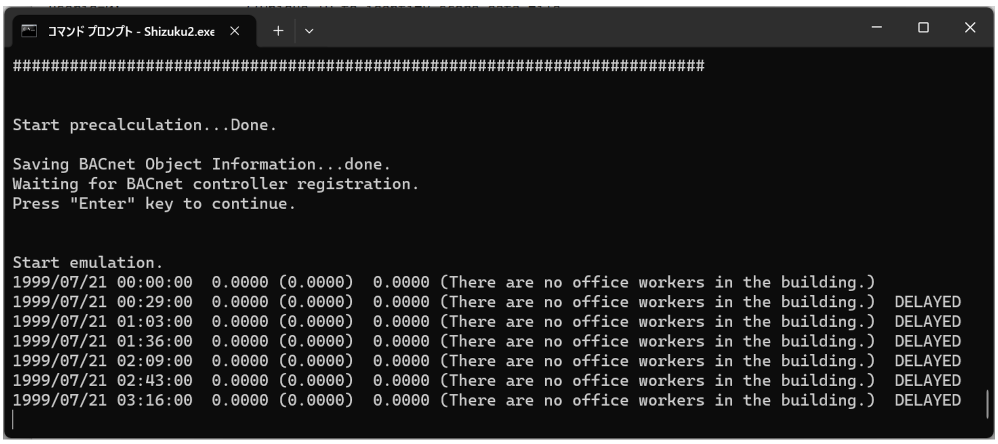


Fig. 2.10 Indication if calculation is not completed in time

Section 3 Controlling the emulator with a Microsoft Excel file

3.1 Software Description

The emulator is controlled using BACnet communication; however, many users have no experience in developing BACnet communication programs. Therefore, a method is provided to control the emulator in the same way as general periodic simulation software.

Fig. 3.1 shows the emulator directory.

```
Shizuku2
  ├─ Shizuku2.exe                  (1)
  ├─ setting.ini                   (2)
  ├─ data (Directory)              (3)
  ├─ ExcelController.exe           (4)
  ├─ schedule.xlsx                 (4a)
  ├─ schedule_samples.xlsx         (4b)
  ├─ CaseStudyProcessor.exe        (4c)
  ├─ schedules (Directory)         (4d)
  ├─ DummyDeivceController.exe     (5)
  ├─ Library                       (6)
  └─ Other files
```

Fig. 3.1 Shizuku2 directory

"(4) ExcelController.exe" enables the user to send control signals via BACnet, according to a schedule entered into a Microsoft Excel sheet. Fig. 3.2 shows the calculation process of *ExcelController*.

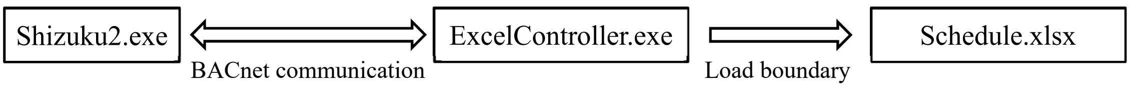
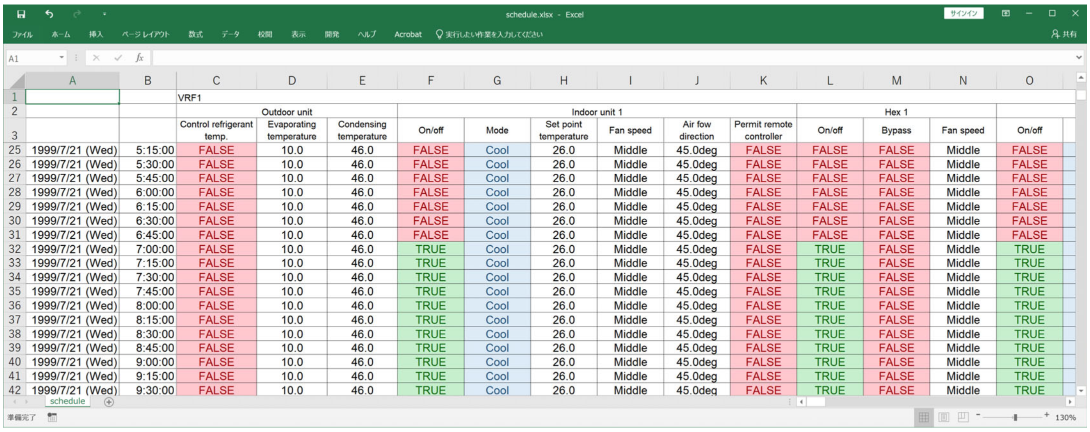


Fig. 3.2 Calculation process of ExcelController

When *ExcelController* is started, the schedule entered in "(4a) schedule.xlsx" is read as a boundary condition. When the time of the emulator (*Shizuku*2) begins to advance, *ExcelController* sends control signals according to the schedule loaded to the emulator in accordance with its speed.

The contents of schedule.xlsx are shown in Fig. 3.3.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			VRF1												
2			Outdoor unit			Indoor unit 1						Hex 1			
3			Control refrigerant temp.	Evaporating temperature	Condensing temperature	On/off	Mode	Set point temperature	Fan speed	Air fow direction	Permit remote controller	On/off	Bypass	Fan speed	On/off
25	1999/7/21 (Wed)	5:15:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
26	1999/7/21 (Wed)	5:30:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
27	1999/7/21 (Wed)	5:45:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
28	1999/7/21 (Wed)	6:00:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
29	1999/7/21 (Wed)	6:15:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
30	1999/7/21 (Wed)	6:30:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
31	1999/7/21 (Wed)	6:45:00	FALSE	10.0	46.0	FALSE	Cool	26.0	Middle	45.0deg	FALSE	FALSE	FALSE	Middle	FALSE
32	1999/7/21 (Wed)	7:00:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
33	1999/7/21 (Wed)	7:15:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
34	1999/7/21 (Wed)	7:30:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
35	1999/7/21 (Wed)	7:45:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
36	1999/7/21 (Wed)	8:00:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
37	1999/7/21 (Wed)	8:15:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
38	1999/7/21 (Wed)	8:30:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
39	1999/7/21 (Wed)	8:45:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
40	1999/7/21 (Wed)	9:00:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
41	1999/7/21 (Wed)	9:15:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE
42	1999/7/21 (Wed)	9:30:00	FALSE	10.0	46.0	TRUE	Cool	26.0	Middle	45.0deg	FALSE	TRUE	FALSE	Middle	TRUE

Fig. 3.3 Content of ExcelController

The controls are aligned vertically every 15 min. This 15-minute interval is a fixed value and cannot be changed. The controls for the outdoor and indoor units and ventilation systems are arranged horizontally. Table 3.1 shows a list of controllable items.

Table 3.1 Controllable items with ExcelController

	Name	Description	Value
Outdoor unit	Control refrigerant temp.	Whether or not the machine attempts to control the temperature of the refrigerant at a constant level.	True / False
	Evaporating temperature	The setpoint of the evaporating temperature when the temperature of the refrigerant is controlled to be constant.	Integer
	Condensing temperature	The setpoint of the condensing temperature when the temperature of the refrigerant is controlled to be constant.	Integer
Indoor unit	On/Off	On off status of the indoor unit.	True / False
	Mode	Operating mode of the indoor unit.	Cool / Heat / Fan
	Set point temperature	Room set point temperature of the indoor unit.	Real
	Fan speed	Fan speed of the indoor unit.	Low / Middle / High
	Air direction	Air direction of the indoor unit.	Horizontal ~ Vertical
	Permit remote controller	Whether or not to allow office workers to manipulate the room temperature setpoint	True / False
HEX	On/Off	On off status of the heat recovery ventilation.	True / False
	Bypass	Whether or not to supply outdoor air bypassing the heat exchanger.	True / False
	Fan speed	Fan speed of the heat recovery ventilation.	True / False

The "(4b) schedule_samples.xlsx" file contains several examples of the schedule.

Table 3.2 shows a list the examples prepared. There are 16 examples: H1‐H8 for the heating operation, and C1‐C8 for the cooling operation. They differ in terms of whether the condensation or evaporation temperature is fixed, the room temperature setpoint, fan speed, airflow direction, whether the occupant is allowed to use the remote controller, and whether the indoor unit in the interior zone is stopped.

Table 3.2 Conditions of simulation cases

Case	-	Condensing / Evaporating temperature [°C]	Setpoint temperature [°C]	Fan speed†	Airflow direction [degree]	Remote control permission	Stop VRF in the interior zone
H1	heating	46.0	22.0	Middle	45.0	false	false
H2		40.0	22.0	Middle	45.0	false	false
H3		46.0	26.0	Middle	45.0	false	false
H4		46.0	22.0	Low	45.0	false	false
H5		46.0	22.0	Middle	5.0	false	false
H6		46.0	22.0	Middle	90.0	false	false
H7		46.0	22.0	Middle	45.0	true	false
H8		46.0	22.0	Middle	45.0	false	true
C1	cooling	10.0	26.0	Middle	45.0	false	false
C2		15.0	26.0	Middle	45.0	false	false
C3		10.0	22.0	Middle	45.0	false	false
C4		10.0	26.0	Low	45.0	false	false
C5		10.0	26.0	Middle	5.0	false	false
C6		10.0	26.0	Middle	90.0	false	false
C7		10.0	26.0	Middle	45.0	true	false
C8		10.0	26.0	Middle	45.0	false	true

When calculations are performed for various cases with multiple schedules, it is difficult to manually replace the schedule files and repeat the calculations. In this case, "(4c) CaseStudyProcessor.exe" can be used to automatically perform calculations for multiple schedule files. As shown in Fig. 3.4, if one or more schedule files are placed in the "(4d) schedules" directory and run the "(4c) CaseStudyProcessor.exe," the calculation is executed continuously, replacing the schedule in the directory.

```
├── ExcelController.exe        (4)
├── schedule.xlsx              (4a)
├── schedule_samples.xlsx      (4b)
├── CaseStudyProcessor.exe     (4c)
├── schedules (Directory)      (4d)
│     ├ schedule1.xlsx
│     ├ schedule2.xlsx
│     ├ schedule3.xlsx
│     └ schedule4.xlsx
```

Fig. 3.4 Batch calculation method

3.2 Execution example

The emulator was started on a standby screen, as shown in Fig. 2.2. When *ExcelController* is started in this state, “Schedule.xlsx” is read and Fig. 3.5 is displayed.

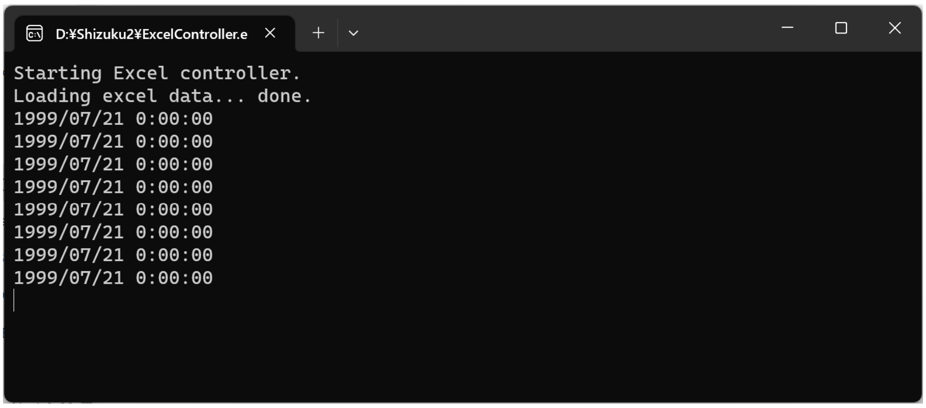


Fig. 3.5 Startup ExcelContoller

ExcelController displays the current date and time in the emulator every second. Because the emulator is still idling and not advancing time, the initial value “1999/07/21 0:00:00” is repeatedly displayed.

Entering the Enter key in the emulator window will display the output shown in Fig. 3.6 in the *ExcelContoller* window. Some controls have been sent to the emulator, and time has begun to move.

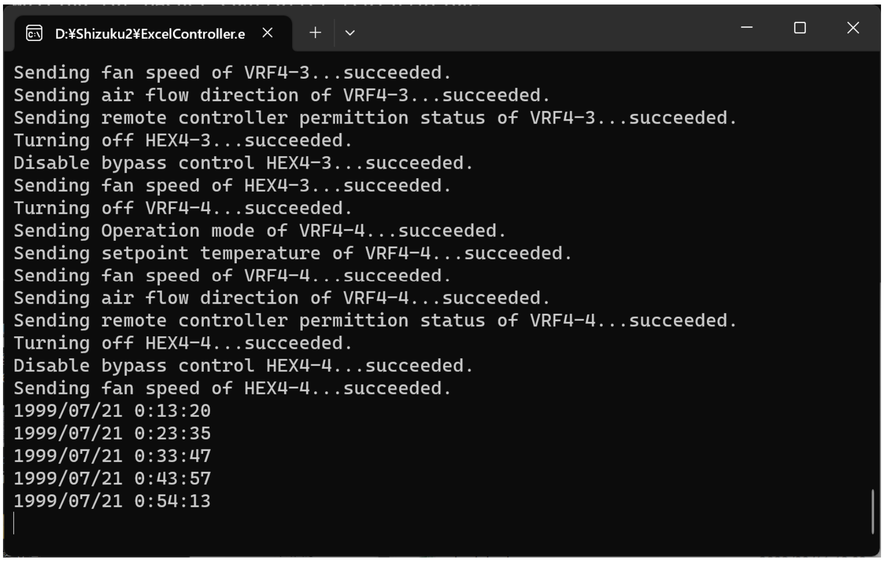


Fig. 3.6 Sending control signals according to schedule

Fig. 3.7 shows the emulator window after leaving it for a while and proceeding with the calculation until around 7:00 a.m. Unlike the case without *ExcelController*, energy is consumed around 7:00 a.m. because the VRF and ventilation systems are working. Because the temperature and humidity in the room are now controlled, and ventilation is enabled, the dissatisfaction due to thermal preference and air pollution is smaller than in the case of no control.

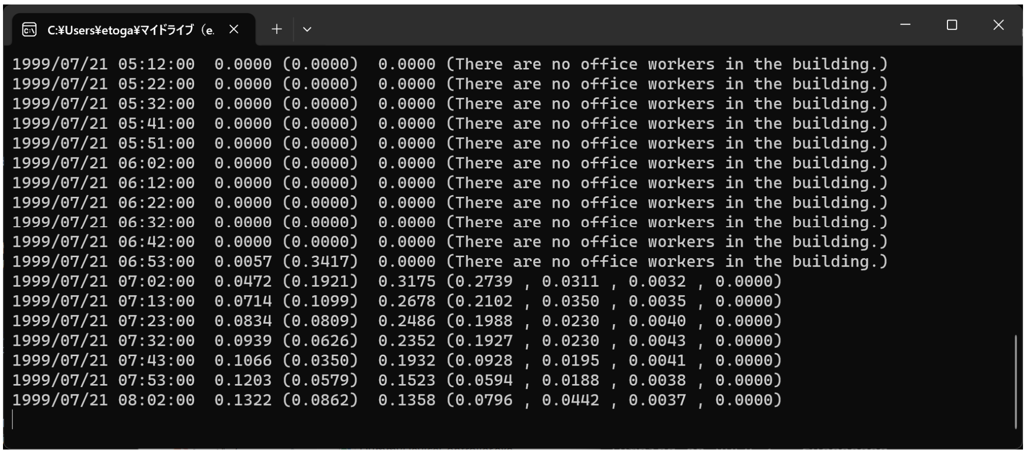


Fig. 3.7 Output of the emulator

Section 4 Controlling the emulator using programs

4.1 Common language-independent information

The specifications for BACnet communication are provided in ASHRAE Standard 135-2020. However, creating a program from scratch based on this specification is impractical. As listed below, libraries for BACnet communication have been developed in many languages, making this work easier.

C#:	BACsharp BACnet Stack	(https://bacsharp.sourceforge.net)
Java:	BACnet4J	(https://github.com/MangoAutomation/BACnet4J)
Python:	BACpypes	(https://bacpypes.readthedocs.io)
C:	BACnet Protocol Stack	(https://sourceforge.net/projects/bacnet)

Many BACnet devices are connected to the BACnet network, and various types of data are stored in these devices. This emulator provides the BACnet devices listed in Table 4.1.

Table 4.1 BACnet devices in the emulator

Name	ID	PORT	Description
DateTimeController	1	47809	Manage simulation date, time, and acceleration speed.
VRFController	2	47810	Operate VRF and manage current operating conditions.
VRFScheduller	3	47811	Manage VRF operations on a schedule. Whether or not to activate this device is optional.
EnvironmentMonitor	4	47812	Monitor outdoor weather conditions and indoor temperature and humidity.
OccupantMonitor	5	47813	Monitor information related to the occupants.
VentilationController	6	47814	Operate ventilation system and manage current operating conditions.
DummyDevice	9	47817	Dummy device to try BACnet communication.

Each BACnet device has an ID that identifies it. Each BACnet device has a different IP address; however, when multiple devices belong to the same IP address, as in this emulator, they are identified using different port numbers.

Several objects are found in a BACnet device, and information related to a device is stored in an object, e.g. in *VRFController*, the on/off status of the indoor unit, fan speed, power consumption, etc.. Each of these objects has its own instance number and type, and their combination is used as an ID with no duplicates. For example, information related to the power consumption of VRF1 is managed as instance number 1021 and as type “Analog Input.” A list of BACnet Devices in the emulator and the objects in each device are shown in Appendix 1.

DateTimeController manages the date and time of the simulation. Unlike real buildings, it contains information related to acceleration, and manipulating this value can change the speed at which the simulation moves forward.

VRFScheduller is a device that allows equipment to run on a standard schedule according to a prewritten program. This device can be enabled or disabled, and is disabled by default.

DummyDevice is used to check whether BACnet communication is possible, and does not affect the simulation results (comfort and energy consumption).

VRFController and *VentilationController* monitor the status and change the operation of the VRF and ventilation systems, respectively. *EnvironmentMonitor* and *OccupantMonitor* are used to monitor the outdoor/indoor air quality and thermal sensations of the occupants. The challenge is to use these four devices to monitor the thermal environment of a building and the response of the occupants while improving the operation of the HVAC system.

As mentioned, the instance number and type must be identified for communication via BACnet; however, writing such a program is complicated. Therefore, we have developed a BACnet communication library for this emulator. The languages available are Python and .NET (C# or Basic). These libraries are contained in the "Libraries" directory as shown in Fig. 4.1. In the following sections, we explain how to use thease libraries.

```
├─ Libraries              (6)
│    ├ python.zip         (6a)
│    └ dotnet.zip         (6b)
```

Fig. 4.1 Python and .NET libraries used to communicate with the emulator

4.2 Controller programs using Python

First, unzip “python.zip” and prepare some Python program files to communicate with the emulator.

Fig. 4.2 shows an UML diagram of the relationship between the classes defined in the library. BACpypes is a BACnet communication library written in Python, and the *PresentValueReadWriter* class uses it to implement the function of reading and writing the present value of any BACnet device. The *PresentValueReadWriter* class also implements processing to synchronize with the emulator.

By inheriting the *PresentValueReadWriter* class, four classes were defined to communicate with the concrete BACnet device in the emulator.

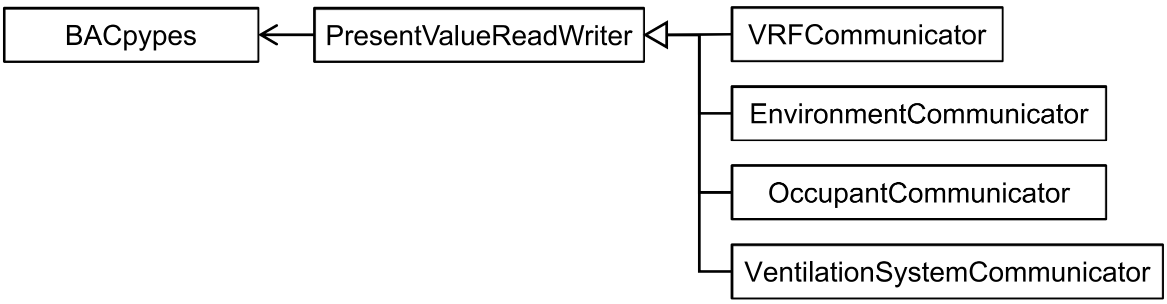


Fig. 4.2 UML of python classes for communicating with the emulator

An example of the development of a specific program for operating an emulator using these classes is presented below. The methods defined in these classes are documented on the following website:

http://www.wccbo.org/lib/python

As mentioned above, because we are using BACpypes, we must install them using the following commands: We will skip the explanation of general tasks, such as the installation of Python and PIP.

```
$ pip install bacpypes
```

1) Time synchronization

A program for synchronizing the emulator with the time is shown in Code 4.1.

Code 4.1 Synchronizing the time with the emulator (python)

sample1.py

```python
import time
import PresentValueReadWriter

pvrw = PresentValueReadWriter.PresentValueReadWriter(10)
print('Subscribe COV...',end='')
while not pvrw.subscribe_date_time_cov():
    time.sleep(0.1)
print('success')
```

```
 9
10 while True:
11     dt = pvrw.current_date_time()
12     print(dt.strftime('%Y/%m/%d %H:%M:%S'))
13     time.sleep(1.0)
```

In line 4, an instance of the *PresentValueReadWriter* class is created, which synchronizes the time with the emulator. The argument of the constructor is the ID of a BACnet device. Because information transmission in the BACnet network occurs between devices, another device is required to communicate with the device in the emulator. The ID of this device is given as an argument, which can be any value, but duplicate values are not allowed in the network; therefore, the value should be a number not used in Table 4.1. Because 1, 2, 3, 4, 5, 6, and 9 are already in use, 10 is used.

The "*subscribe_date_time_cov*" in line 6 is a method for synchronizing the time. It registers the device with the emulator such that it is notified when the emulator's acceleration changes. Because this registration process may fail owing to network conditions, it is looped in lines 6 and 7, and the registration process is repeated at 100-millisecond intervals until it succeeds.

After successful registration, the current date and time (datetime type) can be obtained using the "*current_date_time*" method shown in line 11. Here, in lines 12 and 13, the current date and time are written at one-second intervals.

The results of running Code 4.1 are shown below. First, the date/time display does not change because the emulator's time has stopped; however, when the emulator is moved, the time begins to advance.

```
Subscribe COV...success
1999/07/21 00:00:00
1999/07/21 00:00:00
1999/07/21 00:00:00
1999/07/21 00:00:00
1999/07/21 00:09:13
1999/07/21 00:19:17
1999/07/21 00:29:17
1999/07/21 00:39:19
...
```

As described above, the basis of schedule control is to keep checking the current date and time in a loop and start or stop the HVAC equipment at appropriate times.

Because all Fig. 4.2 classes are inherited from the *PresentValueReadWriter* class, time can be synchronized in the same manner, as explained above.

2) Monitoring of indoor and outdoor environments

The *EnvironmentCommunicator* class is used to monitor the indoor and outdoor environments. The program is shown in Code 4.2, where line 4 is a constructor, and the argument is the ID of the device used for communication.

The "*get_drybulb_temperature*" in line 8 is a method of obtaining the dry bulb temperature of the outdoor air. The return value is an array: the first is whether the communication was successful, and the second is the present value of the dry bulb temperature. Depending on whether the communication was successful, the results were presented in nine lines. Lines 12 and 16 represent the processes of monitoring the relative humidity of outdoor air and global horizontal radiation, respectively.

If you want to monitor the dry-bulb temperature of each zone in a room, assign the outdoor and indoor unit numbers of the VRF that is air-conditioning the zone concerned as arguments, and call the "*get_zone_drybulb_temperature*" method as shown in line 20. Here, the dry-bulb temperature of the zone in which VRF2-4 was air-conditioned was obtained. For relative humidity, do the same, using "*get_zone_relative_humidity*" as shown in line 24.

Code 4.2 Monitoring indoor and outdoor environments of the emulator (python)

sample2.py

```
1 import time
2 import EnvironmentCommunicator
3
4 eCom = EnvironmentCommunicator.EnvironmentCommunicator(14)
5
6 while True:
7       print('Reading outdoor air temperature... ',end='')
8       val = eCom.get_drybulb_temperature()
9       print('{:.1f}'.format(val[1]) + ' C' if val[0] else ' Failed')
10
11      print('Reading outdoor relative humidity... ',end='')
12      val = eCom.get_relative_humidity()
13      print('{:.1f}'.format(val[1]) + ' %' if val[0] else ' Failed')
14
15      print('Reading global horizontal radiation... ',end='')
16      val = eCom.get_global_horizontal_radiation()
17      print('{:.1f}'.format(val[1]) + ' W/m2' if val[0] else ' Failed')
18
19      print('Reading drybulb temperature of zone at VRF2-4... ',end='')
20      val = eCom.get_zone_drybulb_temperature(2,4)
21      print('{:.1f}'.format(val[1]) + ' C' if val[0] else ' Failed')
22
23      print('Reading relative humidity of zone at VRF2-4... ',end='')
24      val = eCom.get_zone_relative_humidity(2,4)
25      print('{:.1f}'.format(val[1]) + ' %' if val[0] else ' Failed')
26
27      print('')
28      time.sleep(1)
```

The results of the Code 4.2 run are shown below. the code shows how the temperature and humidity change as one advances through the time of the emulator.

```
Reading outdoor air temperature... 25.0 C
Reading outdoor relative humidity... 50.0 %
Reading global horizontal radiation... 0.0 W/m2
Reading drybulb temperature of zone at VRF2-4... 25.0 C
Reading relative humidity of zone at VRF2-4... 50.0 %

Reading outdoor air temperature... 25.0 C
Reading outdoor relative humidity... 50.0 %
...
```

3) Monitoring of occupants' information

The *OccupantCommunicator* class is used to obtain information on the office workers. The program is shown in Code 4.3, where the fourth line is the constructor and the argument is the ID of the device to be used for communication.

To obtain the number of occupants by tenant, use the "*get_occupant_number*" method as shown in line 8. The *OccupantCommunicator* class defines an enumerated type "*Tenant*" to distinguish between north and south tenants, which is given as an argument. Line 8 is an example of obtaining the number of north tenants. The return value is an array, the first being whether the communication was successful and the second being the number of occupants.

The number of occupants by zone can also be obtained using the "*get_zone_occupant_number*" method in line 12. In this case, the number of zones was provided as an argument. The zone numbers are shown in Fig. 1.4.

The average thermal sensation and average clo value by zone can be obtained using the "*get_averaged_thermal_sensation*" and "*get_averaged_clothing_index*" methods in lines 16 and 20, respectively. The return value is zero when there is no return to the office.

Line 24 shows an example of using the "*is_occupant_stay_in_office*" method, which determines whether an occupant stays in the office. To use this method, one must specify whether the tenant is north or south, and the index number of the occupant in that tenant. Line 24 monitors the occupancy status of the first occupant in the south office. The index numbers and seating zones for each occupant are presented in Appendix 2.

Using the same arguments, the thermal sensation and clo value for each occupant can be obtained using the "*get_thermal_sensation*" and "*get_clothing_index*" methods, as shown in lines 28 and 32, respectively.

Code 4.3 Monitoring the occupant state of the emulator (python)

sample3.py

```
1 import time
2 import OccupantCommunicator as occ
3
4 oCom = occ.OccupantCommunicator(15)
5
6 while True:
7     print('Reading occupant number in north tenant... ',end='')
8     val = oCom.get_occupant_number(occ.OccupantCommunicator.Tenant.North)
9     print(str(val[1]) if val[0] else ' Failed')
10
11     print('Reading occupant number in south tenant zone-1... ',end='')
12     val = oCom.get_zone_occupant_number(occ.OccupantCommunicator.Tenant.South,1)
13     print(str(val[1]) if val[0] else ' Failed')
14
15     print('Reading averaged thermal sensation (south tenant zone-1)... ',end='')
16     val = oCom.get_averaged_thermal_sensation(occ.OccupantCommunicator.Tenant.South,1)
17     print('{:.2f}'.format(val[1]) if val[0] else ' Failed')
18
19     print('Reading averaged clothing index (south tenant zone-1)... ',end='')
20     val = oCom.get_averaged_clothing_index(occ.OccupantCommunicator.Tenant.South,1)
21     print('{:.2f}'.format(val[1]) if val[0] else ' Failed')
22
23     print('Is occupant No.1 in south tenant stay in office? ... ',end='')
```

```
    val = oCom.is_occupant_stay_in_office(occ.OccupantCommunicator.Tenant.South, 1)
    print(str(val[1]) if val[0] else ' Failed')

    print('Reading thermal sensation of occupant No.2 in south tenant... ',end='')
    val = oCom.get_thermal_sensation(occ.OccupantCommunicator.Tenant.South, 2)
    print(str(val[1]) if val[0] else ' Failed')

    print('Reading clothing index of occupant No.3 in south tenant... ',end='')
    val = oCom.get_clothing_index(occ.OccupantCommunicator.Tenant.South, 3)
    print('{:.2f}'.format(val[1]) + ' Clo' if val[0] else ' Failed')

    print('')
    time.sleep(1)
```

The results of the Code 4.3 run are shown below. One can see how the number of occupants and the thermal sensation change as the time of the emulator advances.

```
Reading occupant number in south tenant... 0
Reading occupant number in north tenant... 0
Reading occupant number in south tenant zone-1... 0
Reading averaged thermal sensation (south tenant zone-1)... 0.0
Reading averaged clothing index (south tenant zone-1)... 0.0
Is occupant No.1 in south tenant stay in office? ... False
Reading thermal sensation of occupant No.2 in south tenant... 0
Reading clothing index of occupant No.3 in south tenant... 0.00 Clo

Reading occupant number in south tenant... 0
Reading occupant number in north tenant... 0
...
```

4) Changing the operation of ventilation system

The *VentilationSystemCommunicator* class is used to control the ventilation system. A sample program is shown in Code 4.4, where line 4 is the constructor and the argument is the ID of the device used for communication.

The CO2 level can be monitored for each tenant, and this information is obtained using the methods shown in lines 8 and 12. As with other classes, the return value is an array, the first indicating whether the communication succeeded and the second is the value of the CO2 level.

To run the total heat exchanger, use the "*start_ventilation*" method as shown in line 16. Because the location of the total heat exchanger is the same as that of the indoor unit of the VRF, the index numbers of the outdoor and indoor units of the VRF are given as arguments. Line 16 shows an example of starting the entire heat exchanger installed in the same zone as the indoor unit of VRF1-1. Line 20 describes how to stop the entire heat exchange.

The fan speed of the total heat exchanger can be controlled in high, medium, or low, and the current setting can be obtained using the "*get_fan_speed*" method, as shown in line 24. The arguments are the index numbers of the outdoor and indoor units of the VRF. The return value is an enumerated type named "*FanSpeed*" and takes three values: "*High*", "*Middle*", and "*Low*". If you want to change the setting, use the "*change_fan_speed*" method in line 28 and give "*FanSpeed*" as an argument in addition to the index number of outdoor and indoor units of the VRF. Line 28 is an example of setting to the "*Middle*" speed.

Code 4.4 Controlling the ventilation system of the emulator (python)

sample4.py

```
import time
import VentilationSystemCommunicator as vsc

vCom = vsc.VentilationSystemCommunicator(16)

while True:
    print('Reading CO2 level of south tenant... ',end='')
    val = vCom.get_south_tenant_CO2_level()
    print(str(val[1]) if val[0] else ' Failed')

    print('Reading CO2 level of north tenant... ',end='')
    val = vCom.get_north_tenant_CO2_level()
    print(str(val[1]) if val[0] else ' Failed')

    print('Turning on HEX1-1... ',end='')
    val = vCom.start_ventilation(1,1)
    print('success' if val[0] else ' Failed')

    print('Turning off HEX1-1... ',end='')
    val = vCom.stop_ventilation(1,1)
    print('success' if val[0] else ' Failed')

    print('Reading fan speed of HEX1-1... ',end='')
    val = vCom.get_fan_speed(1,1)
    print(str(val[1]) if val[0] else ' Failed')

    print('Changing fan speed of HEX1-1 to Middle...',end='')
    rslt = vCom.change_fan_speed(1,1,vsc.VentilationSystemCommunicator.FanSpeed.Middle)
    print('success' if rslt[0] else 'failed')

    print('')
    time.sleep(1)
```

The results of the Code 4.4 run are shown below. We can see how the CO2 level increases or decreases over time in the emulator.

```
Reading CO2 level of south tenant... 400
Reading CO2 level of north tenant... 400
Turning on HEX1-1... success
Turning off HEX1-1... success
Reading fan speed of HEX1-1... FanSpeed.High
Changing fan speed of HEX1-1 to Middle...success

Reading CO2 level of south tenant... 400
Reading CO2 level of north tenant... 400
...
```

5) Changing the operation of the VRF system

The *VRFSystemCommunicator* class is used to control the VRF system. The program is shown in Code 4.5, where line 4 is the constructor, and the argument is the ID of the device used for communication.

The indoor unit measures the dry-bulb temperature and relative humidity of the return air, and the values are obtained using the method shown in lines 8 and 12. The arguments are the index numbers of outdoor and indoor units. In this example, the return air status of VRF1-2 is obtained.

To start or stop the indoor unit, use the "*turn_on*" and "*turn_off*" methods as shown in lines 16 and 20. The outdoor unit starts if any of the connected indoor units start, and stops if all of them stop.

The operation mode is changed by the "*change_mode*" method shown in line 24. The argument is an enumerated type named "*Mode*" in addition to the outdoor and indoor unit index numbers. The operation mode can be selected from "*Cooling*," "*Heating*," or "*ThermoOff*". The VRF in this emulator does not recover heat and runs in either cooling or heating mode. When indoor units with different operation modes are connected to the same outdoor unit, the operation mode of the indoor unit with a smaller number of units is prioritized.

To change the fan speed, use the "*change_fan_speed*" method in line 32. Use the enumerated type named "*FanSpeed*" as the argument, and select from "*High*," "*Middle*," and "*Low*".

To change the air flow direction, use the "*change_direction*" method in line 36. It uses an enumerator named "*Direction*" as an argument and can be set in 22.5-degree increments. Five options exist: "*Horizontal*," "*Degree*_225," "*Degree*_450," "*Degree*_675," and "*Vertical*."

To enable the use of the indoor unit controller by an occupant, use the "*permit_local_control*" method in line 40. For prohibition, use the "*prohibit_local_control*" method in line 44. If allowed, the occupants will change the setpoint temperature according to their thermal preferences. While they feel satisfied with being able to operate the system themselves, there is the danger that a lot of energy will be expended to set it up as they wish.

Code 4.5 Controlling the VRF system of the emulator (python)

sample5.py

```python
1  import time
2  import VRFSystemCommunicator as vrc
3
4  vCom = vrc.VRFSystemCommunicator(12)
5
6  while True:
7      print('Reading return air temperature of VRF1-2...',end='')
8      rslt = vCom.get_return_air_temperature(1,2)
9      print(str(rslt[1]) + ' C' if rslt[0] else 'failed')
10
11     print('Reading return air relative humidity of VRF1-2...',end='')
12     rslt = vCom.get_return_air_relative_humidity(1,2)
13     print(str(rslt[1]) + ' %' if rslt[0] else 'failed')
14
15     print('Turning on VRF1-2...',end='')
16     rslt = vCom.turn_on(1,2)
17     print('success' if rslt[0] else 'failed')
18
19     print('Turning off VRF1-2...',end='')
20     rslt = vCom.turn_off(1,2)
21     print('success' if rslt[0] else 'failed')
22
23     print('Changing mode of VRF1-2 to cooling...',end='')
24     rslt = vCom.change_mode(1,2,vrc.VRFSystemCommunicator.Mode.Cooling)
```

```
    print('success' if rslt[0] else 'failed')

    print('Changing set point temperature of VRF1-2 to 26C...',end='')
    rslt = vCom.change_setpoint_temperature(1,2,26)
    print('success' if rslt[0] else 'failed')

    print('Changing fan speed of VRF1-2 to high...',end='')
    rslt = vCom.change_fan_speed(1,2,vrc.VRFSystemCommunicator.FanSpeed.High)
    print('success' if rslt[0] else 'failed')

    print('Changing direction of VRF1-2 to 45degree...',end='')
    rslt = vCom.change_direction(1,2,vrc.VRFSystemCommunicator.Direction.Degree_450)
    print('success' if rslt[0] else 'failed')

    print('Permitting local control of VRF1-2...',end='')
    rslt = vCom.permit_local_control(1,2)
    print('success' if rslt[0] else 'failed')

    print('Prohibiting local control of VRF1-2...',end='')
    rslt = vCom.prohibit_local_control(1,2)
    print('success' if rslt[0] else 'failed')

    print('')
    time.sleep(1)
```

The results of the Code 4.5 run are presented below. The return temperature and humidity fluctuate as the emulator advances.

```
Reading return air temperature of VRF1-2...24.0 C
Reading return air relative humidity of VRF1-2...50.0 %
Turning on VRF1-2...success
Turning off VRF1-2...success
Changing mode of VRF1-2 to cooling...success
Changing set point temperature of VRF1-2 to 26C...success
Changing fan speed of VRF1-2 to high...success
Changing direction of VRF1-2 to 45degree...success
Permitting local control of VRF1-2...success
Prohibiting local control of VRF1-2...success

Reading return air temperature of VRF1-2...24.0 C
Reading return air relative humidity of VRF1-2...50.0 %
...
```

6) Control according to schedule

An example of a simple scheduler is shown in Code 4.6.

Instances of communication with the VRF and ventilation system are created in Lines 6 and 7.

To control the air-conditioning units according to the current date and time, time synchronization is enabled in line 11; therefore, the synchronization of both the VRF and the ventilation system communication instance is not required.

The array in line 16 represents the number of indoor units in each VRF system.

The loop in lines 19–75 determines whether to control the air conditioning every 0.5 s; as shown in lines 18 and

74, the date and time of the previous loop are stored in "*last_dt*" to start the air conditioning when it changes from the time of day to stop to the time of day to run, and to stop it when the opposite is true. to the decision of whether to start air conditioning is determined by the day of the week and time and is calculated by the method defined in lines 77–83.

The current date and time are outputted to the console, as shown in lines 21 and 22.

The cooling and heating modes and setpoint temperature are switched according to the season, summer or winter, as shown in lines 25–28.

When starting up the air conditioning, not only do you start up the VRF and ventilation system, but also set the fan speed and air flow direction of the indoor unit, as shown in lines 31–58.

The process for stopping the system is shown in lines 61–72.

Code 4.6 Simple VRF and ventilation system scheduler for the emulator (python)

sample6.py

```
1  import time, datetime
2  import VRFSystemCommunicator as vrc
3  import VentilationSystemCommunicator as vsc
4
5  def main():
6      vrCom = vrc.VRFCommunicator(12)
7      vsCom = vsc.VentilationSystemCommunicator(16)
8
9      # Enable current_date_time method
10     print('Subscribe COV...')
11     while not vrCom.subscribe_date_time_cov():
12         time.sleep(0.1)
13     print('success')
14
15     # Number of indoor units in each VRF system
16     i_unit_num = [5,4,5,4]
17
18     last_dt = vrCom.current_date_time()
19     while True:
20         # Output current date and time
21         dt = vrCom.current_date_time()
22         print(dt.strftime('%Y/%m/%d %H:%M:%S'))
23
24         # Change mode, air flow direction, and set point temperature depends on season
25         is_s = 5 <= dt.month and dt.month <= 10
26         mode = vrc.VRFSystemCommunicator.Mode.Cooling if is_s else vrc.VRFSystemCommunicator.Mode.Heating
27         dir = vrc.VRFSystemCommunicator.Direction.Horizontal if is_s else vrc.VRFSystemCommunicator.Direction.Vertical
28         sp = 26 if is_s else 22
29
30         # When the HVAC changed to operating hours
31         if(not(is_hvac_time(last_dt)) and is_hvac_time(dt)):
32             for i in range(len(i_unit_num)):
33                 for j in range(i_unit_num[i]):
34                     v_name = 'VRF' + str(i + 1) + '-' + str(j+1)
35
36                     print('Turning on ' + v_name + '...',end='')
37                     rslt = vrCom.turn_on(i+1,j+1)
38                     print('success' if rslt[0] else 'failed: ' + rslt[1])
39
```

```
                print('Turning on ' + v_name + ' (Ventilation)...',end='')
                rslt = vsCom.start_ventilation(i+1,j+1)
                print('success' if rslt[0] else 'failed: ' + rslt[1])

                print('Changing mode of ' + v_name + ' to ' + str(mode) + '...',end='')
                rslt = vrCom.change_mode(i+1,j+1,mode)
                print('success' if rslt[0] else 'failed: ' + rslt[1])

                print('Changing set point temperature of ' + v_name + ' to ' + str(sp) + 'C...',end='')
                rslt = vrCom.change_setpoint_temperature(i+1,j+1,sp)
                print('success' if rslt[0] else 'failed: ' + rslt[1])

                print('Changing fanspeed of ' + v_name + ' to Middle...',end='')
                rslt = vrCom.change_fan_speed(i+1,j+1,vrc.VRFSystemCommunicator.FanSpeed.Middle)
                print('success' if rslt[0] else 'failed: ' + rslt[1])

                print('Changing direction of ' + v_name + ' to ' + str(dir) + '...',end='')
                rslt = vrCom.change_direction(i+1,j+1,dir)
                print('success' if rslt[0] else 'failed: ' + rslt[1])

        # When the HVAC changed to stop hours
        if(is_hvac_time(last_dt) and not(is_hvac_time(dt))):
            for i in range(len(i_unit_num)):
                for j in range(i_unit_num[i]):
                    v_name = 'VRF' + str(i + 1) + '-' + str(j+1)

                    print('Turning off ' + v_name + '...',end='')
                    rslt = vrCom.turn_off(i+1,j+1)
                    print('success' if rslt else 'failed')

                    print('Turning off ' + v_name + ' (Ventilation)...',end='')
                    rslt = vsCom.stop_ventilation(i+1,j+1)
                    print('success' if rslt else 'failed')

        last_dt = dt # Save last date and time
        time.sleep(0.5)

def is_hvac_time(dtime):
    start_time = datetime.time(7, 0)
    end_time = datetime.time(19, 0)
    now = dtime.time()
    is_business_hour = start_time <= now <= end_time
    is_weekday = (dtime.weekday() != 5 and dtime.weekday() != 6)
    return is_weekday and is_business_hour

if __name__ == "__main__":
    main()
```

7) CO2 level-based ventilation control

Code 4.7 shows a program that adjusts the ventilation volume according to the CO2 level.

The methods for synchronizing the time and determining the time of day for air conditioning are the same as those in Code 4.6.

Lines 21–40 show the processes for controlling the ventilation fan speed. The process is repeated at 1-second intervals during the day for air conditioning.

Lines 23–26 show the process of monitoring the CO2 levels for each tenant. The ventilation fan speed is changed according to the CO2 level using the "*get_fan_speed*" method as shown in lines 29 and 30. This method is defined in lines 43–49. The fan speed of each of the heat exchangers is changed from lines 37 to 40.

Code 4.7 Demand control ventilation with CO2 level (python)

sample7.py

```
1 import time, datetime
2 import VentilationSystemCommunicator as vsc
3
4 def main():
5     vsCom = vsc.VentilationSystemCommunicator(26)
6
7     # Enable current_date_time method
8     print('Subscribe COV...')
9     while not vsCom.subscribe_date_time_cov():
10        time.sleep(0.1)
11    print('success')
12
13    # Number of indoor units in each VRF system
14    i_unit_num = [5,4,5,4]
15
16    while True:
17        # Output current date and time
18        dt = vsCom.current_date_time()
19        print(dt.strftime('%Y/%m/%d %H:%M:%S'))
20
21        if(is_hvac_time(dt)):
22            # Get CO2 level
23            val = vsCom.get_south_tenant_CO2_level()
24            south_co2 = val[1] if val[0] else 1000
25            val = vsCom.get_north_tenant_CO2_level()
26            north_co2 = val[1] if val[0] else 1000
27
28            # Switch fan speed
29            south_fs = get_fan_speed(south_co2)
30            north_fs = get_fan_speed(north_co2)
31
32            # Output status
33            print('South tenant: ' + str(south_fs) + ' (' + str(south_co2) + ')')
34            print('North tenant: ' + str(north_fs) + ' (' + str(north_co2) + ')')
35
36            # Change fan speed
37            for i in range(len(i_unit_num)):
38                fs = south_fs if i == 0 or i==1 else north_fs
39                for j in range(i_unit_num[i]):
40                    val = vsCom.change_fan_speed(i+1,j+1,fs)
41        time.sleep(1.0)
42
43 def get_fan_speed(co2_level):
44    if co2_level < 600:
45        return vsc.VentilationSystemCommunicator.FanSpeed.Low
46    elif co2_level < 800:
47        return vsc.VentilationSystemCommunicator.FanSpeed.Middle
48    else:
49        return vsc.VentilationSystemCommunicator.FanSpeed.High
```

```python
50
51 def is_hvac_time(dtime):
52     start_time = datetime.time(7, 0)
53     end_time = datetime.time(19, 0)
54     now = dtime.time()
55     is_business_hour = start_time <= now <= end_time
56     is_weekday = (dtime.weekday() != 5 and dtime.weekday() != 6)
57     return is_weekday and is_business_hour
58
59 if __name__ == "__main__":
60     main()
61
```

This program only controls the fan speed of all heat exchangers; therefore, the on/off status must be controlled using other programs. You can run Code 4.6 you have already developed simultaneously. Because a BACnet device can communicate with multiple devices simultaneously, control functions can be distributed, as shown in Fig. 4.3. Avoid duplicating device IDs (line 7 of Code 4.6 and line 5 of Code 4.7).

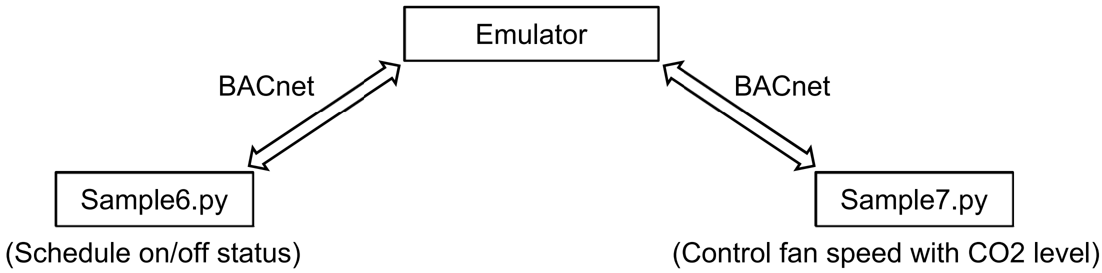


Fig. 4.3 Emulator control by multiple BACnet Devices

Fig. 4.4 shows how the CO2 level in the south office changes over a week when the ventilation is controlled only by Code 4.6 and when Code 4.7 is enabled. When ventilation is controlled by the CO2 level, the level remained at a slightly higher value. The primary energy consumption per week is reduced by more than 10%, from 8.73 GJ to 7.71 GJ.

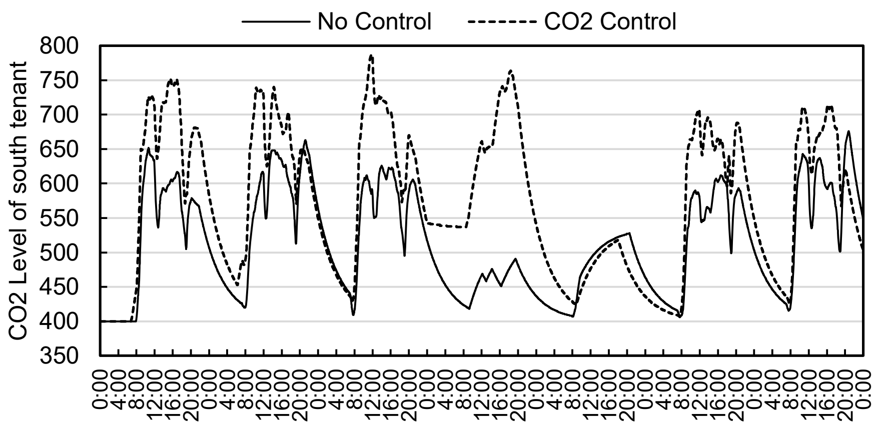


Fig. 4.4 CO2 level of the south office with and without CO2 control

4.3 Controller programs using C#

In this section, we use C# to develop a program with the same functionality as the program in Python developed in the previous section.

First, unzip “dotnet.zip” in the *Libraries* directory and prepare the Visual Studio solution files shown in Fig. 4.5.

```
SampleControllers
        ├ SampleContollers.sln
        ├ dll (directory)
        │      ├ Shizuku2Lib.dll
        │      └ Other files
        ├ Sample1
        ├ Sample2
        ├ Sample3
        ├ Sample4
        ├ Sample5
        ├ Sample6
        ├ Sample7
        └ publish
```

Fig. 4.5 Sample controller projects for Visual Studio

“*BACSharp*” is a BACnet communication library for. “NET”. “Shizuku2Lib.dll” in the “*dll*” directory communicates with emulators using *BACSharp*. By loading this DLL, one can easily communicate with the emulator in C# or basic language.

The structure of the prepared classes is the same as that of the Python library shown in Fig. 4.2, with the basic “*PresentValueReadWriter*” class and four concrete communication classes derived from it.

Below, we show concrete programs for the same functions as in the previous section; however, because the method names and functions defined in each class are almost the same as those in the Python library, we omit duplicate explanations.

Documentation for each class can be found at the following website.

http://www.wccbo.org/lib/dotnet

1) Time synchronization

As in the Python example, an instance of the *PresentValueReadWriter* class is created in line 9. This argument is the device ID. In C#, the “*StartService*” method must be called to initiate BACnet communication, as shown in line 10. This process is the same for the subsequent samples.

Time synchronization is initiated by registering with the COV in line 13. In C#, the current time can be referenced in the "*CurrentDateTime*" property, as shown in line 19.

Code 4.8 Time synchronization with the emulator (C#)

Sample1/Program.cs

```
1 using Shizuku2.BACnet;
2 
3 namespace Sample1
4 {
```

```
    internal class Program
    {
        static void Main(string[] args)
        {
            PresentValueReadWriter pvrw = new PresentValueReadWriter(10);
            pvrw.StartService();

            Console.Write("Subscribe COV...");
            while (!pvrw.SubscribeDateTimeCOV())
                Thread.Sleep(100);
            Console.WriteLine("success");

            while (true)
            {
                DateTime dt = pvrw.CurrentDateTime;
                Console.WriteLine(dt.ToString("yyyy/MM/dd HH:mm:ss"));
                Thread.Sleep(1000);
            }
        }
    }
}
```

2) Monitoring of indoor and outdoor environments

Instance creation is similar to Python.

In C#, the success or failure of communication is passed on with reference to the method argument. For example, in line 17, ”*succeeded*” is assigned the result of whether the communication was successful.

Code 4.9 Monitoring indoor and outdoor environments of the emulator (C#)

Sample2/Program.cs

```
1  using Shizuku2.BACnet;
2  
3  namespace Sample2
4  {
5      internal class Program
6      {
7          static void Main(string[] args)
8          {
9              EnvironmentCommunicator eCom = new EnvironmentCommunicator(14);
10             eCom.StartService();
11 
12             while (true)
13             {
14                 bool succeeded;
15 
16                 Console.Write("Reading outdoor air temperature...");
17                 double dbt = eCom.GetDrybulbTemperature(out succeeded);
18                 Console.WriteLine(succeeded ? dbt.ToString("F1") : "failed");
19 
20                 Console.Write("Reading outdoor air relative humidity...");
21                 double hmd = eCom.GetRelativeHumidity(out succeeded);
22                 Console.WriteLine(succeeded ? hmd.ToString("F1") : "failed");
23 
24                 Console.Write("Reading global horizontal radiation...");
```

```
                double rad = eCom.GetGlobalHorizontalRadiation(out succeeded);
                Console.WriteLine(succeeded ? rad.ToString("F1") : "failed");

                Console.Write("Reading drybulb temperature of zone at VRF2-4...");
                double dbtZn = eCom.GetZoneDrybulbTemperature(2, 4, out succeeded);
                Console.WriteLine(succeeded ? dbtZn.ToString("F1") : "failed");

                Console.Write("Reading relative humidity of zone at VRF2-4...");
                double hmdZn = eCom.GetZoneRelativeHumidity(2, 4, out succeeded);
                Console.WriteLine(succeeded ? hmdZn.ToString("F1") : "failed");

                Console.WriteLine();
                Thread.Sleep(1000);
            }
        }
    }
}
```

3) Monitoring of occupant information

The program flow is almost the same as a program in python.

Code 4.10 Monitoring the occupant state of the emulator (C#)

Sample3/Program.cs

```
using Shizuku2.BACnet;

namespace Sample3
{
    internal class Program
    {
        static void Main(string[] args)
        {
            OccupantCommunicator oCom = new OccupantCommunicator(15);
            oCom.StartService();

            while (true)
            {
                bool succeeded;

                Console.Write("Reading occupant number in north tenant......");
                int oNum = oCom.GetOccupantNumber(OccupantCommunicator.Tenant.North, out succeeded);
                Console.WriteLine(succeeded ? oNum.ToString() : "failed");

                Console.Write("Reading occupant number in south tenant zone-1...");
                int oNumZ = oCom.GetOccupantNumber(OccupantCommunicator.Tenant.North, 1, out succeeded);
                Console.WriteLine(succeeded ? oNumZ.ToString() : "failed");

                Console.Write("Reading averaged thermal sensation (south tenant zone-1)...");
                float aTS = oCom.GetAveragedThermalSensation(OccupantCommunicator.Tenant.North, 1, out succeeded);
                Console.WriteLine(succeeded ? aTS.ToString("F1") : "failed");

                Console.Write("Reading averaged clothing index (south tenant zone-1)...");
                float aCI = oCom.GetAveragedClothingIndex(OccupantCommunicator.Tenant.North, 1, out succeeded);
                Console.WriteLine(succeeded ? aCI.ToString("F1") : "failed");

```

```
            Console.Write("Is occupant No.1 in south tenant stay in office? ...");
            bool ocS = oCom.IsOccupantStayInOffice(OccupantCommunicator.Tenant.North, 1, out succeeded);
            Console.WriteLine(succeeded ? ocS.ToString() : "failed");

            Console.Write("Reading thermal sensation of occupant No.2 in south tenant...");
            OccupantCommunicator.ThermalSensation ts =
              oCom.GetThermalSensation(OccupantCommunicator.Tenant.South, 2, out succeeded);
            Console.WriteLine(succeeded ? ts.ToString() : "failed");

            Console.Write("Reading clothing index of occupant No.3 in south tenant...");
            float ci = oCom.GetClothingIndex(OccupantCommunicator.Tenant.North, 3, out succeeded);
            Console.WriteLine(succeeded ? ci.ToString("F2") : "failed");

            Console.WriteLine();
            Thread.Sleep(1000);
          }
      }
  }
}
```

4) Changing the operation of the ventilation system

The program flow is almost the same as a program in python.

Code 4.11 Control of the ventilation system of the emulator (C#)

Sample4/Program.cs

```
using Shizuku2.BACnet;

namespace Sample4
{
  internal class Program
  {
    static void Main(string[] args)
    {
      VentilationSystemCommunicator vCom = new VentilationSystemCommunicator(16);
      vCom.StartService();

      while (true)
      {
        bool succeeded;

        Console.Write("Reading CO2 level of south tenant...");
        double coS = vCom.GetSouthTenantCO2Level(out succeeded);
        Console.WriteLine(succeeded ? coS.ToString() : "failed");

        Console.Write("Reading CO2 level of north tenant...");
        double coN = vCom.GetNorthTenantCO2Level(out succeeded);
        Console.WriteLine(succeeded ? coN.ToString() : "failed");

        Console.Write("Turning on HEX1-1...");
        vCom.StartVentilation(1, 1, out succeeded);
        Console.WriteLine(succeeded ? "success" : "failed");

        Console.Write("Turning off HEX1-1...");
        vCom.StopVentilation(1, 1, out succeeded);
```

```
            Console.WriteLine(succeeded ? "success" : "failed");

            Console.Write("Reading fan speed of HEX1-1...");
            VentilationSystemCommunicator.FanSpeed fs = vCom.GetFanSpeed(1, 1, out succeeded);
            Console.WriteLine(succeeded ? fs.ToString() : "failed");

            Console.Write("Changing fan speed of HEX1-1 to Middle...");
            vCom.ChangeFanSpeed(1, 1, VentilationSystemCommunicator.FanSpeed.Middle, out succeeded);
            Console.WriteLine(succeeded ? "success" : "failed");

            Console.WriteLine();
            Thread.Sleep(1000);
        }
    }
  }
}
```

5) Changing the operation of the VRF system

The program flow is almost the same as the program in python.

Code 4.12 Control of the VRF system of the emulator (C#)

sample5.py

```
using Shizuku2.BACnet;

namespace Sample5
{
  internal class Program
  {
    static void Main(string[] args)
    {
      VRFSystemCommunicator vCom = new VRFSystemCommunicator(12);
      vCom.StartService();

      while (true)
      {
        bool succeeded;

        Console.Write("Reading return air temperature of VRF1-2...");
        double dbt = vCom.GetReturnAirTemperature(1, 2, out succeeded);
        Console.WriteLine(succeeded ? dbt.ToString("F1") : "failed");

        Console.Write("Reading return air relative humidity of VRF1-2...");
        double hmd = vCom.GetReturnAirRelativeHumidity(1, 2, out succeeded);
        Console.WriteLine(succeeded ? hmd.ToString("F1") : "failed");

        Console.Write("Turning on VRF1-2...");
        vCom.TurnOn(1, 2, out succeeded);
        Console.WriteLine(succeeded ? "success" : "failed");

        Console.Write("Turning off VRF1-2...");
        vCom.TurnOff(1, 2, out succeeded);
        Console.WriteLine(succeeded ? "success" : "failed");

        Console.Write("Changing mode of VRF1-2 to cooling...");
```

```
                vCom.ChangeMode(1, 2, VRFSystemCommunicator.Mode.Cooling, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing set point temperature of VRF1-2 to 26C...");
                vCom.ChangeSetpointTemperature(1, 2, 26, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing fan speed of VRF1-2 to high...");
                vCom.ChangeFanSpeed(1, 2, VRFSystemCommunicator.FanSpeed.High, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing direction of VRF1-2 to 45degree...");
                vCom.ChangeDirection(1, 2, VRFSystemCommunicator.Direction.Degree_450, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Permitting local control of VRF1-2...");
                vCom.PermitLocalControl(1,2,out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Prohibiting local control of VRF1-2...");
                vCom.ProhibitLocalControl(1,2,out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.WriteLine();
                Thread.Sleep(1000);
            }
        }
    }
}
```

6) Control according to schedule

The program flow is almost the same as a program in python.

Code 4.13 Simple VRF and ventilation system scheduler for the emulator (C#)

Sample6/Program.cs

```
using Shizuku2.BACnet;

namespace Sample6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            VRFSystemCommunicator vrCom = new VRFSystemCommunicator(12);
            VentilationSystemCommunicator vsCom = new VentilationSystemCommunicator(16);
            vrCom.StartService();
            vsCom.StartService();

            // Enable CurrentDateTime property
            Console.Write("Subscribe COV...");
            while (!vrCom.SubscribeDateTimeCOV())
                Thread.Sleep(100);
            Console.WriteLine("success");

```

```
// Number of indoor units in each VRF system
int[] iUnitNum = new int[] { 5, 4, 5, 4 };

DateTime lastDt = vrCom.CurrentDateTime;
while (true)
{
    DateTime dt = vrCom.CurrentDateTime;
    Console.WriteLine(dt.ToString("yyyy/MM/dd HH:mm:ss"));

    // Change mode, air flow direction, and set point temperature depends on season
    bool isSum = 5 <= dt.Month && dt.Month <= 10;
    VRFSystemCommunicator.Mode mode = VRFSystemCommunicator.Mode.Heating;
    VRFSystemCommunicator.Direction dir = VRFSystemCommunicator.Direction.Vertical;
    float sp = 22;
    if (isSum)
    {
        mode = VRFSystemCommunicator.Mode.Cooling;
        dir = VRFSystemCommunicator.Direction.Horizontal;
        sp = 26;
    }

    // When the HVAC changed to operating hours
    if (!isHVACTime(lastDt) && isHVACTime(dt))
    {
        for (int i = 0; i < iUnitNum.Length; i++)
        {
            for (int j = 0; j < iUnitNum[i]; j++)
            {
                bool succeeded;
                uint oIdx = (uint)(i + 1);
                uint iIdx = (uint)(j + 1);
                string vName = "VRF" + oIdx + "-" + iIdx;

                Console.Write("Turning on " + vName + "...");
                vrCom.TurnOn(oIdx, iIdx, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Turning on " + vName + "(Ventilation)...");
                vsCom.StartVentilation(oIdx, iIdx, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing mode of " + vName + " to " + mode + "...");
                vrCom.ChangeMode(oIdx, iIdx, mode, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing set point temperature of " + vName + " to " + sp + "C...");
                vrCom.ChangeSetpointTemperature(oIdx, iIdx, sp, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing fan speed of " + vName + " to Middle...");
                vrCom.ChangeFanSpeed(oIdx, iIdx, VRFSystemCommunicator.FanSpeed.Middle, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");

                Console.Write("Changing air flow direction of " + vName + " to " + dir + "...");
                vrCom.ChangeDirection(oIdx, iIdx, dir, out succeeded);
                Console.WriteLine(succeeded ? "success" : "failed");
            }
        }
```

```
            }
            // When the HVAC changed to stop hours
            else if (isHVACTime(lastDt) && !isHVACTime(dt))
            {
                for (int i = 0; i < iUnitNum.Length; i++)
                {
                    for (int j = 0; j < iUnitNum[i]; j++)
                    {
                        bool succeeded;
                        uint oIdx = (uint)(i + 1);
                        uint iIdx = (uint)(j + 1);
                        string vName = "VRF" + oIdx + "-" + iIdx;

                        Console.Write("Turning off " + vName + "...");
                        vrCom.TurnOff(oIdx, iIdx, out succeeded);
                        Console.WriteLine(succeeded ? "success" : "failed");

                        Console.Write("Turning off " + vName + "(Ventilation)...");
                        vsCom.StopVentilation(oIdx, iIdx, out succeeded);
                        Console.WriteLine(succeeded ? "success" : "failed");
                    }
                }
            }

            lastDt = dt;
            Thread.Sleep(500);
        }
    }

    static bool isHVACTime(DateTime dt)
    {
        bool isBusinessHour = 7 <= dt.Hour && dt.Hour <= 19;
        bool isWeekday = dt.DayOfWeek != DayOfWeek.Saturday && dt.DayOfWeek != DayOfWeek.Sunday;
        return isWeekday && isBusinessHour;
    }
  }
}
```

7) CO2 level-based ventilation control

The program flow is almost the same as the program in python.

Code 4.14 Demand control ventilation with CO2 level (C#)

Sample7/Program.cs

```
using Shizuku2.BACnet;

namespace Sample7
{
  internal class Program
  {
    static void Main(string[] args)
    {
      VentilationSystemCommunicator vsCom = new VentilationSystemCommunicator(26);
      vsCom.StartService();

```

```
            // Enable CurrentDateTime property
            Console.Write("Subscribe COV...");
            while (!vsCom.SubscribeDateTimeCOV())
                Thread.Sleep(100);
            Console.WriteLine("success");

            // Number of indoor units in each VRF system
            int[] iUnitNum = new int[] { 5, 4, 5, 4 };

            while (true)
            {
                DateTime dt = vsCom.CurrentDateTime;
                Console.WriteLine(dt.ToString("yyyy/MM/dd HH:mm:ss"));

                // When the HVAC changed to operating hours
                if (isHVACTime(dt))
                {
                    for (int i = 0; i < iUnitNum.Length; i++)
                    {
                        bool succeeded;
                        uint southCO2 = vsCom.GetSouthTenantCO2Level(out succeeded);
                        uint northCO2 = vsCom.GetNorthTenantCO2Level(out succeeded);

                        VentilationSystemCommunicator.FanSpeed southFS = getFanSpeed(southCO2);
                        VentilationSystemCommunicator.FanSpeed northFS = getFanSpeed(northCO2);

                        Console.WriteLine("South tenant: " + southFS.ToString() + "(" + southCO2.ToString() + ")");
                        Console.WriteLine("North tenant: " + northFS.ToString() + "(" + northCO2.ToString() + ")");

                        for (int j = 0; j < iUnitNum[i]; j++)
                        {
                            VentilationSystemCommunicator.FanSpeed fs = i == 0 ? southFS : northFS;
                            vsCom.ChangeFanSpeed((uint)(i + 1), (uint)(j + 1), fs, out _);
                        }
                    }
                }

                Thread.Sleep(1000);
            }
        }

        static VentilationSystemCommunicator.FanSpeed getFanSpeed(uint co2Level)
        {
            if (co2Level < 600) return VentilationSystemCommunicator.FanSpeed.Low;
            else if (co2Level < 800) return VentilationSystemCommunicator.FanSpeed.Middle;
            else return VentilationSystemCommunicator.FanSpeed.High;
        }

        static bool isHVACTime(DateTime dt)
        {
            bool isBusinessHour = 7 <= dt.Hour && dt.Hour <= 19;
            bool isWeekday = dt.DayOfWeek != DayOfWeek.Saturday && dt.DayOfWeek != DayOfWeek.Sunday;
            return isWeekday && isBusinessHour;
        }
    }
}
```

Section 5 Points to keep in mind when improving HVAC operations

This chapter discusses the mechanisms by which buildings, VRF, and occupant characteristics affect energy consumption and comfort. All of these are explicitly expressed inside the emulator using physical equations and statistics, and should be given attention to optimize the operation of the VRF.

5.1 Building-related notes

1) Owing to the influence of the outer envelope, the heat load trends differed between the perimeter and interior zones. Particularly in winter, heating and cooling may be required in perimeter and interior zones, respectively, and the heat supplied by the HVAC system may mix, resulting in losses.
2) Because of the changing position of the sun, the thermal environment varies with building orientation and time of day. The east side of the building has a greater influence on solar radiation in the morning, whereas the north side has a smaller influence throughout the day.
3) In winter, cooling and heating demands may switch during the day, with heating in the morning and cooling in the afternoon. This is particularly likely to occur in the interior zones, where the influence of the outer envelope is small.
4) Owing to the thermal capacity of the building, it takes time for the room temperature to stabilize after the air conditioning has started. This time is generally greater in winter than in summer because the temperature difference between the inside and outside of the building is greater.
5) Owing to the thermal capacity of the building, the indoor temperature does not immediately equal the outdoor temperature when air conditioning is turned off.
6) The perimeter zone has windows and exterior walls that are thermally influenced from the outdoors; therefore, the radiant thermal environment differs from that of the interior zone. Therefore, the perimeter zone feels warmer during the cooling season and colder during the heating season, compared to the interior zone, even when the air temperature and humidity are the same.
7) Indoor air mixes easily in the horizontal direction. Therefore, even if an indoor unit in one zone is stopped, the temperature and humidity do not change significantly because the air mixes with the adjacent zone.
8) As air has different densities depending on its temperature, a vertical temperature distribution is created, where the upper side is warmer and the lower side is cooler. Air is more difficult to mix vertically than horizontally; unless forced to do so by a fan, eliminating the vertical temperature distribution is difficult.
9) A total heat exchanger is a device that reduces energy by exchanging heat between the exhaust air from indoors and the supply air from outdoors. However, in some cases, such as cooling in the fall or winter, the heat load can be reduced by bypassing air and disabling heat recovery.

5.2 VRF system-related notes

1) Lowering the setpoint temperature during the cooling season increases the thermal load and energy consumption.
2) Increasing the evaporation temperature during the cooling operation reduces the energy consumption, even at the same cooling load. However, the maximum cooling capacity of the VRF will be reduced. In addition, the amount of dehumidification is reduced, which may affect comfort.
3) Increasing the setpoint temperature during the heating season increases the heat load and energy consumption.
4) Lowering the condensing temperature during the heating operation reduces energy consumption, even at the

same heating load. However, the maximum heating capacity of the VRF becomes smaller. In addition, the blowout temperature of the indoor unit will be lower, and the possibility of dissatisfaction owing to drafts will increase.

5) The energy efficiency of the VRF varies with the partial load rate, and is lower at lower load rates.
6) During cooling, the airflow blown out from the indoor unit does not go straight but curves downward and falls. The lower the blowing temperature, the greater is the curvature.
7) During heating, the airflow blown out from the indoor unit does not go straight but curves upward. The higher the blowing temperature, the greater is the curvature.
8) The higher the blowing air velocity of the indoor unit, the farther the airflow reaches. Therefore, if the air velocity is significantly reduced during heating, the airflow does not reach the lower space, thereby increasing the risk of a large vertical temperature distribution.
9) The higher the blowing-air velocity of the indoor unit, the greater its capacity. However, during cooling, the ratio of latent heat exchange (dehumidification) is reduced, which may affect comfort.
10) When the blowing angle of an indoor unit is made closer to the vertical direction, the ratio of airflow reaching the lower space increases, and the vertical temperature distribution decreases. However, the risk of a draft increases because the velocity of the airflow to the occupants increases.

5.3 Occupant related notes

1) Thermal sensations are primarily influenced by six factors: dry-bulb temperature, relative humidity, mean radiant temperature, relative air velocity, amount of clothing, and metabolic rate.
2) People have certain thermal preferences.
3) People may feel dissatisfied when there is a large vertical temperature distribution in a room.
4) When the airflow from the indoor unit directly hits the skin, occupants may complain of chills. However, when occupants feel that a space is warm, dissatisfaction is unlikely to occur.
5) Occupants are dissatisfied when the ventilation is low and the air is excessively polluted. (Note that in this emulator, occupants are programmed to complain when the CO2 level exceeds 1,000 ppm, but actual occupants are not as sensitive to CO2 concentration.)
6) Occupants decide the amount of clothing they will wear that day by referring to the thermal environment of the room on the previous day and the outside air conditions on the morning of the day. Even after arriving at work, occupants can adjust the amount of clothing they wear to some extent by wearing jackets or rolling their sleeves up.
7) Occupants are more likely to feel satisfied when they can operate air-conditioning units and adjust their thermal environments.
8) Occupants first try to adjust the thermal environment using their personal clothing, and when they do not resolve their dissatisfaction, they try to change their air conditioner settings.

[References]

1) ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) (2020): Standard 135-2020, BACnet - A Data Communication Protocol for Building Automation and Control Networks

Appendix 1

BACnet devices and objects

1) Objects in the “DateTimeController” device

Inst. No.	Type	Name	Description	Initial value
1	DATETIME_VALUE	Current date and time	Current date and time on the simulation. This value might been accelerated.	1999/7/21 0:00
2	ANALOG_OUTPUT	Acceleration rate	This object is used to set the acceleration rate to run the emulator.	0
3	DATETIME_VALUE	Base real date and time	Real world date and time starting to accelerate.	2023/9/25 18:42
4	DATETIME_VALUE	Base date and time in the simulation	Date and time on the simulation when the acceleration started	1999/7/21 0:00

2) Objects in the “VRFController” device

Instance number = 1000 × outdoor unit index + 100 × indoor unit index + member number.

For information related to the entire system, use zero for the indoor unit index.

Member numbers are as follows:

OnOff_Setting = 1, OnOff_Status = 2, OperationMode_Setting = 3, OperationMode_Status = 4, Setpoint_Setting = 5 and Setpoint_Status = 6

MeasuredRoomTemperature = 7, MeasuredRelativeHumidity = 8, FanSpeed_Setting = 9, FanSpeed_Status = 10, AirflowDirection_Setting = 11,

AirflowDirection_Status = 12, RemoteControllerPermittion_Setpoint_Setting = 13, RemoteControllerPermittion_Setpoint_Status = 14,

ForcedRefrigerantTemperature_Setting = 15, ForcedRefrigerantTemperature_Status = 16, EvaporatingTemperatureSetpoint_Setting = 17,

EvaporatingTemperatureSetpoint_Status = 18, CondensingTemperatureSetpoint_Setting = 19, CondensingTemperatureSetpoint_Status = 20,

Electricity = 21, HeatLoad = 22

Inst. No.	Type	Name	Description	Initial value
1015	BINARY_VALUE	RefrigerantTempCtrlSetting_VRF1	This object is used to change the forced evaporating/condensing control of VRF system.	0
1016	BINARY_INPUT	RefrigerantTempCtrlStatus_VRF1	This object is used to monitor the forced evaporating/condensing control of VRF system.	0
1017	ANALOG_VALUE	EvpTempSetting_VRF1	This object is used to set the evaporating temperature of VRF system.	10
1018	ANALOG_INPUT	EvpTempStatus_VRF1	This object is used to monitor the evaporating temperature of VRF system.	10
1019	ANALOG_VALUE	CndTempSetting_VRF1	This object is used to set the condensing temperature of VRF system.	45
1020	ANALOG_INPUT	CndTempStatus_VRF1	This object is used to monitor the condensing temperature of VRF system.	45
1021	ANALOG_INPUT	Electricity_VRF1	This object is used to monitor the outdoor unit's electric consumption (fans and compressors).	0
1022	ANALOG_INPUT	HeatLoad_VRF1	This object is used to monitor the heat load of VRF system.	0
1101	BINARY_OUTPUT	OnOffCommand_VRF1-1	This object is used to start (On)/stop (Off) the indoor unit.	0
1102	BINARY_INPUT	OnOffStatus_VRF1-1	This object is used to monitor the indoor unit's On/Off status.	0
1103	MULTI_STATE_OUTPUT	ModeCommand_VRF1-1	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1104	MULTI_STATE_INPUT	ModeStatus_VRF1-1	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1105	ANALOG_VALUE	TempSPSetting_VRF1-1	This object is used to set the indoor unit's setpoint.	24
1106	ANALOG_INPUT	TempSPStatus_VRF1-1	This object is used to monitor the indoor unit's setpoint.	24
1107	ANALOG_INPUT	RoomTemp_VRF1-1	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
1108	ANALOG_INPUT	RoomRHmid_VRF1-1	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
1109	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF1-1	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1110	MULTI_STATE_INPUT	AirFlowRateStatus_VRF1-1	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2

1111	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF1-1	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1112	MULTI_STATE_INPUT	AirDirectionStatus_VRF1-1	This object is used to monitor the indoor unit’s airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1113	BINARY_VALUE	RemoteControlStart_VRF1-1	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
1114	BINARY_INPUT	RemoteControlStart_VRF1-1	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
1121	ANALOG_INPUT	Electricity_VRF1-1	This object is used to monitor the indoor unit's electric consumption.	0
1122	ANALOG_INPUT	HeatLoad_VRF1-1	This object is used to monitor the heat load of indoor unit.	0
1201	BINARY_OUTPUT	OnOffCommand_VRF1-2	This object is used to start (On)/stop (Off) the indoor unit.	0
1202	BINARY_INPUT	OnOffStatus_VRF1-2	This object is used to monitor the indoor unit's On/Off status.	0
1203	MULTI_STATE_OUTPUT	ModeCommand_VRF1-2	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1204	MULTI_STATE_INPUT	ModeStatus_VRF1-2	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1205	ANALOG_VALUE	TempSPSetting_VRF1-2	This object is used to set the indoor unit's setpoint.	24
1206	ANALOG_INPUT	TempSPStatus_VRF1-2	This object is used to monitor the indoor unit's setpoint.	24
1207	ANALOG_INPUT	RoomTemp_VRF1-2	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
1208	ANALOG_INPUT	RoomRHmid_VRF1-2	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
1209	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF1-2	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1210	MULTI_STATE_INPUT	AirFlowRateStatus_VRF1-2	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1211	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF1-2	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1212	MULTI_STATE_INPUT	AirDirectionStatus_VRF1-2	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1213	BINARY_VALUE	RemoteControlStart_VRF1-2	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
1214	BINARY_INPUT	RemoteControlStart_VRF1-2	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
1221	ANALOG_INPUT	Electricity_VRF1-2	This object is used to monitor the indoor unit's electric consumption.	0
1222	ANALOG_INPUT	HeatLoad_VRF1-2	This object is used to monitor the heat load of indoor unit.	0
1301	BINARY_OUTPUT	OnOffCommand_VRF1-3	This object is used to start (On)/stop (Off) the indoor unit.	0
1302	BINARY_INPUT	OnOffStatus_VRF1-3	This object is used to monitor the indoor unit's On/Off status.	0
1303	MULTI_STATE_OUTPUT	ModeCommand_VRF1-3	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1304	MULTI_STATE_INPUT	ModeStatus_VRF1-3	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1305	ANALOG_VALUE	TempSPSetting_VRF1-3	This object is used to set the indoor unit's setpoint.	24
1306	ANALOG_INPUT	TempSPStatus_VRF1-3	This object is used to monitor the indoor unit's setpoint.	24
1307	ANALOG_INPUT	RoomTemp_VRF1-3	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
1308	ANALOG_INPUT	RoomRHmid_VRF1-3	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
1309	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF1-3	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1310	MULTI_STATE_INPUT	AirFlowRateStatus_VRF1-3	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1311	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF1-3	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1312	MULTI_STATE_INPUT	AirDirectionStatus_VRF1-3	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1313	BINARY_VALUE	RemoteControlStart_VRF1-3	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
1314	BINARY_INPUT	RemoteControlStart_VRF1-3	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
1321	ANALOG_INPUT	Electricity_VRF1-3	This object is used to monitor the indoor unit's electric consumption.	0
1322	ANALOG_INPUT	HeatLoad_VRF1-3	This object is used to monitor the heat load of indoor unit.	0
1401	BINARY_OUTPUT	OnOffCommand_VRF1-4	This object is used to start (On)/stop (Off) the indoor unit.	0
1402	BINARY_INPUT	OnOffStatus_VRF1-4	This object is used to monitor the indoor unit's On/Off status.	0
1403	MULTI_STATE_OUTPUT	ModeCommand_VRF1-4	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1404	MULTI_STATE_INPUT	ModeStatus_VRF1-4	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1405	ANALOG_VALUE	TempSPSetting_VRF1-4	This object is used to set the indoor unit's setpoint.	24
1406	ANALOG_INPUT	TempSPStatus_VRF1-4	This object is used to monitor the indoor unit's setpoint.	24
1407	ANALOG_INPUT	RoomTemp_VRF1-4	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
1408	ANALOG_INPUT	RoomRHmid_VRF1-4	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
1409	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF1-4	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1410	MULTI_STATE_INPUT	AirFlowRateStatus_VRF1-4	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2

1411	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF1-4	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1412	MULTI_STATE_INPUT	AirDirectionStatus_VRF1-4	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1413	BINARY_VALUE	RemoteControlStart_VRF1-4	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
1414	BINARY_INPUT	RemoteControlStart_VRF1-4	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
1421	ANALOG_INPUT	Electricity_VRF1-4	This object is used to monitor the indoor unit's electric consumption.	0
1422	ANALOG_INPUT	HeatLoad_VRF1-4	This object is used to monitor the heat load of indoor unit.	0
1501	BINARY_OUTPUT	OnOffCommand_VRF1-5	This object is used to start (On)/stop (Off) the indoor unit.	0
1502	BINARY_INPUT	OnOffStatus_VRF1-5	This object is used to monitor the indoor unit's On/Off status.	0
1503	MULTI_STATE_OUTPUT	ModeCommand_VRF1-5	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1504	MULTI_STATE_INPUT	ModeStatus_VRF1-5	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
1505	ANALOG_VALUE	TempSPSetting_VRF1-5	This object is used to set the indoor unit's setpoint.	24
1506	ANALOG_INPUT	TempSPStatus_VRF1-5	This object is used to monitor the indoor unit's setpoint.	24
1507	ANALOG_INPUT	RoomTemp_VRF1-5	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
1508	ANALOG_INPUT	RoomRHmid_VRF1-5	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
1509	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF1-5	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1510	MULTI_STATE_INPUT	AirFlowRateStatus_VRF1-5	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
1511	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF1-5	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1512	MULTI_STATE_INPUT	AirDirectionStatus_VRF1-5	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
1513	BINARY_VALUE	RemoteControlStart_VRF1-5	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
1514	BINARY_INPUT	RemoteControlStart_VRF1-5	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
1521	ANALOG_INPUT	Electricity_VRF1-5	This object is used to monitor the indoor unit's electric consumption.	0
1522	ANALOG_INPUT	HeatLoad_VRF1-5	This object is used to monitor the heat load of indoor unit.	0
2015	BINARY_VALUE	RefrigerantTempCtrlSetting_VRF2	This object is used to change the forced evaporating/condensing control of VRF system.	0
2016	BINARY_INPUT	RefrigerantTempCtrlStatus_VRF2	This object is used to monitor the forced evaporating/condensing control of VRF system.	0
2017	ANALOG_VALUE	EvpTempSetting_VRF2	This object is used to set the evaporating temperature of VRF system.	10
2018	ANALOG_INPUT	EvpTempStatus_VRF2	This object is used to monitor the evaporating temperature of VRF system.	10
2019	ANALOG_VALUE	CndTempSetting_VRF2	This object is used to set the condensing temperature of VRF system.	45
2020	ANALOG_INPUT	CndTempStatus_VRF2	This object is used to monitor the condensing temperature of VRF system.	45
2021	ANALOG_INPUT	Electricity_VRF2	This object is used to monitor the outdoor unit's electric consumption (fans and compressors).	0
2022	ANALOG_INPUT	HeatLoad_VRF2	This object is used to monitor the heat load of VRF system.	0
2101	BINARY_OUTPUT	OnOffCommand_VRF2-1	This object is used to start (On)/stop (Off) the indoor unit.	0
2102	BINARY_INPUT	OnOffStatus_VRF2-1	This object is used to monitor the indoor unit's On/Off status.	0
2103	MULTI_STATE_OUTPUT	ModeCommand_VRF2-1	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2104	MULTI_STATE_INPUT	ModeStatus_VRF2-1	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2105	ANALOG_VALUE	TempSPSetting_VRF2-1	This object is used to set the indoor unit's setpoint.	24
2106	ANALOG_INPUT	TempSPStatus_VRF2-1	This object is used to monitor the indoor unit's setpoint.	24
2107	ANALOG_INPUT	RoomTemp_VRF2-1	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
2108	ANALOG_INPUT	RoomRHmid_VRF2-1	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
2109	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF2-1	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2110	MULTI_STATE_INPUT	AirFlowRateStatus_VRF2-1	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2111	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF2-1	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2112	MULTI_STATE_INPUT	AirDirectionStatus_VRF2-1	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2113	BINARY_VALUE	RemoteControlStart_VRF2-1	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
2114	BINARY_INPUT	RemoteControlStart_VRF2-1	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
2121	ANALOG_INPUT	Electricity_VRF2-1	This object is used to monitor the indoor unit's electric consumption.	0
2122	ANALOG_INPUT	HeatLoad_VRF2-1	This object is used to monitor the heat load of indoor unit.	0
2201	BINARY_OUTPUT	OnOffCommand_VRF2-2	This object is used to start (On)/stop (Off) the indoor unit.	0
2202	BINARY_INPUT	OnOffStatus_VRF2-2	This object is used to monitor the indoor unit's On/Off status.	0

2203	MULTI_STATE_OUTPUT	ModeCommand_VRF2-2	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2204	MULTI_STATE_INPUT	ModeStatus_VRF2-2	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2205	ANALOG_VALUE	TempSPSetting_VRF2-2	This object is used to set the indoor unit's setpoint.	24
2206	ANALOG_INPUT	TempSPStatus_VRF2-2	This object is used to monitor the indoor unit's setpoint.	24
2207	ANALOG_INPUT	RoomTemp_VRF2-2	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
2208	ANALOG_INPUT	RoomRHmid_VRF2-2	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
2209	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF2-2	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2210	MULTI_STATE_INPUT	AirFlowRateStatus_VRF2-2	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2211	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF2-2	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2212	MULTI_STATE_INPUT	AirDirectionStatus_VRF2-2	This object is used to monitor the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2213	BINARY_VALUE	RemoteControlStart_VRF2-2	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
2214	BINARY_INPUT	RemoteControlStart_VRF2-2	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
2221	ANALOG_INPUT	Electricity_VRF2-2	This object is used to monitor the indoor unit's electric consumption.	0
2222	ANALOG_INPUT	HeatLoad_VRF2-2	This object is used to monitor the heat load of indoor unit.	0
2301	BINARY_OUTPUT	OnOffCommand_VRF2-3	This object is used to start (On)/stop (Off) the indoor unit.	0
2302	BINARY_INPUT	OnOffStatus_VRF2-3	This object is used to monitor the indoor unit's On/Off status.	0
2303	MULTI_STATE_OUTPUT	ModeCommand_VRF2-3	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2304	MULTI_STATE_INPUT	ModeStatus_VRF2-3	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2305	ANALOG_VALUE	TempSPSetting_VRF2-3	This object is used to set the indoor unit's setpoint.	24
2306	ANALOG_INPUT	TempSPStatus_VRF2-3	This object is used to monitor the indoor unit's setpoint.	24
2307	ANALOG_INPUT	RoomTemp_VRF2-3	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
2308	ANALOG_INPUT	RoomRHmid_VRF2-3	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
2309	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF2-3	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2310	MULTI_STATE_INPUT	AirFlowRateStatus_VRF2-3	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2311	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF2-3	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2312	MULTI_STATE_INPUT	AirDirectionStatus_VRF2-3	This object is used to monitor the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2313	BINARY_VALUE	RemoteControlStart_VRF2-3	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
2314	BINARY_INPUT	RemoteControlStart_VRF2-3	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
2321	ANALOG_INPUT	Electricity_VRF2-3	This object is used to monitor the indoor unit's electric consumption.	0
2322	ANALOG_INPUT	HeatLoad_VRF2-3	This object is used to monitor the heat load of indoor unit.	0
2401	BINARY_OUTPUT	OnOffCommand_VRF2-4	This object is used to start (On)/stop (Off) the indoor unit.	0
2402	BINARY_INPUT	OnOffStatus_VRF2-4	This object is used to monitor the indoor unit's On/Off status.	0
2403	MULTI_STATE_OUTPUT	ModeCommand_VRF2-4	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2404	MULTI_STATE_INPUT	ModeStatus_VRF2-4	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
2405	ANALOG_VALUE	TempSPSetting_VRF2-4	This object is used to set the indoor unit's setpoint.	24
2406	ANALOG_INPUT	TempSPStatus_VRF2-4	This object is used to monitor the indoor unit's setpoint.	24
2407	ANALOG_INPUT	RoomTemp_VRF2-4	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
2408	ANALOG_INPUT	RoomRHmid_VRF2-4	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
2409	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF2-4	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2410	MULTI_STATE_INPUT	AirFlowRateStatus_VRF2-4	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
2411	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF2-4	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2412	MULTI_STATE_INPUT	AirDirectionStatus_VRF2-4	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
2413	BINARY_VALUE	RemoteControlStart_VRF2-4	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
2414	BINARY_INPUT	RemoteControlStart_VRF2-4	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
2421	ANALOG_INPUT	Electricity_VRF2-4	This object is used to monitor the indoor unit's electric consumption.	0
2422	ANALOG_INPUT	HeatLoad_VRF2-4	This object is used to monitor the heat load of indoor unit.	0
3015	BINARY_VALUE	RefrigerantTempCtrlSetting_VRF3	This object is used to change the forced evaporating/condensing control of VRF system.	0
3016	BINARY_INPUT	RefrigerantTempCtrlStatus_VRF3	This object is used to monitor the forced evaporating/condensing control of VRF system.	0

3017	ANALOG_VALUE	EvpTempSetting_VRF3	This object is used to set the evaporating temperature of VRF system.	10
3018	ANALOG_INPUT	EvpTempStatus_VRF3	This object is used to monitor the evaporating temperature of VRF system.	10
3019	ANALOG_VALUE	CndTempSetting_VRF3	This object is used to set the condensing temperature of VRF system.	45
3020	ANALOG_INPUT	CndTempStatus_VRF3	This object is used to monitor the condensing temperature of VRF system.	45
3021	ANALOG_INPUT	Electricity_VRF3	This object is used to monitor the outdoor unit's electric consumption (fans and compressors).	0
3022	ANALOG_INPUT	HeatLoad_VRF3	This object is used to monitor the heat load of VRF system.	0
3101	BINARY_OUTPUT	OnOffCommand_VRF3-1	This object is used to start (On)/stop (Off) the indoor unit.	0
3102	BINARY_INPUT	OnOffStatus_VRF3-1	This object is used to monitor the indoor unit's On/Off status.	0
3103	MULTI_STATE_OUTPUT	ModeCommand_VRF3-1	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3104	MULTI_STATE_INPUT	ModeStatus_VRF3-1	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3105	ANALOG_VALUE	TempSPSetting_VRF3-1	This object is used to set the indoor unit's setpoint.	24
3106	ANALOG_INPUT	TempSPStatus_VRF3-1	This object is used to monitor the indoor unit's setpoint.	24
3107	ANALOG_INPUT	RoomTemp_VRF3-1	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
3108	ANALOG_INPUT	RoomRHmid_VRF3-1	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
3109	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF3-1	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3110	MULTI_STATE_INPUT	AirFlowRateStatus_VRF3-1	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3111	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF3-1	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3112	MULTI_STATE_INPUT	AirDirectionStatus_VRF3-1	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3113	BINARY_VALUE	RemoteControlStart_VRF3-1	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
3114	BINARY_INPUT	RemoteControlStart_VRF3-1	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
3121	ANALOG_INPUT	Electricity_VRF3-1	This object is used to monitor the indoor unit's electric consumption.	0
3122	ANALOG_INPUT	HeatLoad_VRF3-1	This object is used to monitor the heat load of indoor unit.	0
3201	BINARY_OUTPUT	OnOffCommand_VRF3-2	This object is used to start (On)/stop (Off) the indoor unit.	0
3202	BINARY_INPUT	OnOffStatus_VRF3-2	This object is used to monitor the indoor unit's On/Off status.	0
3203	MULTI_STATE_OUTPUT	ModeCommand_VRF3-2	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3204	MULTI_STATE_INPUT	ModeStatus_VRF3-2	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3205	ANALOG_VALUE	TempSPSetting_VRF3-2	This object is used to set the indoor unit's setpoint.	24
3206	ANALOG_INPUT	TempSPStatus_VRF3-2	This object is used to monitor the indoor unit's setpoint.	24
3207	ANALOG_INPUT	RoomTemp_VRF3-2	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
3208	ANALOG_INPUT	RoomRHmid_VRF3-2	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
3209	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF3-2	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3210	MULTI_STATE_INPUT	AirFlowRateStatus_VRF3-2	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3211	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF3-2	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3212	MULTI_STATE_INPUT	AirDirectionStatus_VRF3-2	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3213	BINARY_VALUE	RemoteControlStart_VRF3-2	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
3214	BINARY_INPUT	RemoteControlStart_VRF3-2	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
3221	ANALOG_INPUT	Electricity_VRF3-2	This object is used to monitor the indoor unit's electric consumption.	0
3222	ANALOG_INPUT	HeatLoad_VRF3-2	This object is used to monitor the heat load of indoor unit.	0
3301	BINARY_OUTPUT	OnOffCommand_VRF3-3	This object is used to start (On)/stop (Off) the indoor unit.	0
3302	BINARY_INPUT	OnOffStatus_VRF3-3	This object is used to monitor the indoor unit's On/Off status.	0
3303	MULTI_STATE_OUTPUT	ModeCommand_VRF3-3	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3304	MULTI_STATE_INPUT	ModeStatus_VRF3-3	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3305	ANALOG_VALUE	TempSPSetting_VRF3-3	This object is used to set the indoor unit's setpoint.	24
3306	ANALOG_INPUT	TempSPStatus_VRF3-3	This object is used to monitor the indoor unit's setpoint.	24
3307	ANALOG_INPUT	RoomTemp_VRF3-3	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
3308	ANALOG_INPUT	RoomRHmid_VRF3-3	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
3309	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF3-3	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3310	MULTI_STATE_INPUT	AirFlowRateStatus_VRF3-3	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2

3311	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF3-3	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3312	MULTI_STATE_INPUT	AirDirectionStatus_VRF3-3	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3313	BINARY_VALUE	RemoteControlStart_VRF3-3	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
3314	BINARY_INPUT	RemoteControlStart_VRF3-3	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
3321	ANALOG_INPUT	Electricity_VRF3-3	This object is used to monitor the indoor unit's electric consumption.	0
3322	ANALOG_INPUT	HeatLoad_VRF3-3	This object is used to monitor the heat load of indoor unit.	0
3401	BINARY_OUTPUT	OnOffCommand_VRF3-4	This object is used to start (On)/stop (Off) the indoor unit.	0
3402	BINARY_INPUT	OnOffStatus_VRF3-4	This object is used to monitor the indoor unit's On/Off status.	0
3403	MULTI_STATE_OUTPUT	ModeCommand_VRF3-4	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3404	MULTI_STATE_INPUT	ModeStatus_VRF3-4	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3405	ANALOG_VALUE	TempSPSetting_VRF3-4	This object is used to set the indoor unit's setpoint.	24
3406	ANALOG_INPUT	TempSPStatus_VRF3-4	This object is used to monitor the indoor unit's setpoint.	24
3407	ANALOG_INPUT	RoomTemp_VRF3-4	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
3408	ANALOG_INPUT	RoomRHmid_VRF3-4	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
3409	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF3-4	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3410	MULTI_STATE_INPUT	AirFlowRateStatus_VRF3-4	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3411	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF3-4	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3412	MULTI_STATE_INPUT	AirDirectionStatus_VRF3-4	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3413	BINARY_VALUE	RemoteControlStart_VRF3-4	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
3414	BINARY_INPUT	RemoteControlStart_VRF3-4	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
3421	ANALOG_INPUT	Electricity_VRF3-4	This object is used to monitor the indoor unit's electric consumption.	0
3422	ANALOG_INPUT	HeatLoad_VRF3-4	This object is used to monitor the heat load of indoor unit.	0
3501	BINARY_OUTPUT	OnOffCommand_VRF3-5	This object is used to start (On)/stop (Off) the indoor unit.	0
3502	BINARY_INPUT	OnOffStatus_VRF3-5	This object is used to monitor the indoor unit's On/Off status.	0
3503	MULTI_STATE_OUTPUT	ModeCommand_VRF3-5	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3504	MULTI_STATE_INPUT	ModeStatus_VRF3-5	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
3505	ANALOG_VALUE	TempSPSetting_VRF3-5	This object is used to set the indoor unit's setpoint.	24
3506	ANALOG_INPUT	TempSPStatus_VRF3-5	This object is used to monitor the indoor unit's setpoint.	24
3507	ANALOG_INPUT	RoomTemp_VRF3-5	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
3508	ANALOG_INPUT	RoomRHmid_VRF3-5	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
3509	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF3-5	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3510	MULTI_STATE_INPUT	AirFlowRateStatus_VRF3-5	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
3511	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF3-5	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3512	MULTI_STATE_INPUT	AirDirectionStatus_VRF3-5	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
3513	BINARY_VALUE	RemoteControlStart_VRF3-5	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
3514	BINARY_INPUT	RemoteControlStart_VRF3-5	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
3521	ANALOG_INPUT	Electricity_VRF3-5	This object is used to monitor the indoor unit's electric consumption.	0
3522	ANALOG_INPUT	HeatLoad_VRF3-5	This object is used to monitor the heat load of indoor unit.	0
4015	BINARY_VALUE	RefrigerantTempCtrlSetting_VRF4	This object is used to change the forced evaporating/condensing control of VRF system.	0
4016	BINARY_INPUT	RefrigerantTempCtrlStatus_VRF4	This object is used to monitor the forced evaporating/condensing control of VRF system.	0
4017	ANALOG_VALUE	EvpTempSetting_VRF4	This object is used to set the evaporating temperature of VRF system.	10
4018	ANALOG_INPUT	EvpTempStatus_VRF4	This object is used to monitor the evaporating temperature of VRF system.	10
4019	ANALOG_VALUE	CndTempSetting_VRF4	This object is used to set the condensing temperature of VRF system.	45
4020	ANALOG_INPUT	CndTempStatus_VRF4	This object is used to monitor the condensing temperature of VRF system.	45
4021	ANALOG_INPUT	Electricity_VRF4	This object is used to monitor the outdoor unit's electric consumption (fans and compressors).	0
4022	ANALOG_INPUT	HeatLoad_VRF4	This object is used to monitor the heat load of VRF system.	0
4101	BINARY_OUTPUT	OnOffCommand_VRF4-1	This object is used to start (On)/stop (Off) the indoor unit.	0
4102	BINARY_INPUT	OnOffStatus_VRF4-1	This object is used to monitor the indoor unit's On/Off status.	0

4103	MULTI_STATE_OUTPUT	ModeCommand_VRF4-1	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4104	MULTI_STATE_INPUT	ModeStatus_VRF4-1	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4105	ANALOG_VALUE	TempSPSetting_VRF4-1	This object is used to set the indoor unit's setpoint.	24
4106	ANALOG_INPUT	TempSPStatus_VRF4-1	This object is used to monitor the indoor unit's setpoint.	24
4107	ANALOG_INPUT	RoomTemp_VRF4-1	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
4108	ANALOG_INPUT	RoomRHmid_VRF4-1	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
4109	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF4-1	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4110	MULTI_STATE_INPUT	AirFlowRateStatus_VRF4-1	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4111	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF4-1	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4112	MULTI_STATE_INPUT	AirDirectionStatus_VRF4-1	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4113	BINARY_VALUE	RemoteControlStart_VRF4-1	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
4114	BINARY_INPUT	RemoteControlStart_VRF4-1	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
4121	ANALOG_INPUT	Electricity_VRF4-1	This object is used to monitor the indoor unit's electric consumption.	0
4122	ANALOG_INPUT	HeatLoad_VRF4-1	This object is used to monitor the heat load of indoor unit.	0
4201	BINARY_OUTPUT	OnOffCommand_VRF4-2	This object is used to start (On)/stop (Off) the indoor unit.	0
4202	BINARY_INPUT	OnOffStatus_VRF4-2	This object is used to monitor the indoor unit's On/Off status.	0
4203	MULTI_STATE_OUTPUT	ModeCommand_VRF4-2	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4204	MULTI_STATE_INPUT	ModeStatus_VRF4-2	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4205	ANALOG_VALUE	TempSPSetting_VRF4-2	This object is used to set the indoor unit's setpoint.	24
4206	ANALOG_INPUT	TempSPStatus_VRF4-2	This object is used to monitor the indoor unit's setpoint.	24
4207	ANALOG_INPUT	RoomTemp_VRF4-2	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
4208	ANALOG_INPUT	RoomRHmid_VRF4-2	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
4209	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF4-2	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4210	MULTI_STATE_INPUT	AirFlowRateStatus_VRF4-2	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4211	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF4-2	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4212	MULTI_STATE_INPUT	AirDirectionStatus_VRF4-2	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4213	BINARY_VALUE	RemoteControlStart_VRF4-2	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
4214	BINARY_INPUT	RemoteControlStart_VRF4-2	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
4221	ANALOG_INPUT	Electricity_VRF4-2	This object is used to monitor the indoor unit's electric consumption.	0
4222	ANALOG_INPUT	HeatLoad_VRF4-2	This object is used to monitor the heat load of indoor unit.	0
4301	BINARY_OUTPUT	OnOffCommand_VRF4-3	This object is used to start (On)/stop (Off) the indoor unit.	0
4302	BINARY_INPUT	OnOffStatus_VRF4-3	This object is used to monitor the indoor unit's On/Off status.	0
4303	MULTI_STATE_OUTPUT	ModeCommand_VRF4-3	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4304	MULTI_STATE_INPUT	ModeStatus_VRF4-3	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4305	ANALOG_VALUE	TempSPSetting_VRF4-3	This object is used to set the indoor unit's setpoint.	24
4306	ANALOG_INPUT	TempSPStatus_VRF4-3	This object is used to monitor the indoor unit's setpoint.	24
4307	ANALOG_INPUT	RoomTemp_VRF4-3	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
4308	ANALOG_INPUT	RoomRHmid_VRF4-3	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
4309	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF4-3	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4310	MULTI_STATE_INPUT	AirFlowRateStatus_VRF4-3	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4311	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF4-3	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4312	MULTI_STATE_INPUT	AirDirectionStatus_VRF4-3	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4313	BINARY_VALUE	RemoteControlStart_VRF4-3	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
4314	BINARY_INPUT	RemoteControlStart_VRF4-3	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
4321	ANALOG_INPUT	Electricity_VRF4-3	This object is used to monitor the indoor unit's electric consumption.	0
4322	ANALOG_INPUT	HeatLoad_VRF4-3	This object is used to monitor the heat load of indoor unit.	0
4401	BINARY_OUTPUT	OnOffCommand_VRF4-4	This object is used to start (On)/stop (Off) the indoor unit.	0
4402	BINARY_INPUT	OnOffStatus_VRF4-4	This object is used to monitor the indoor unit's On/Off status.	0

4403	MULTI_STATE_OUTPUT	ModeCommand_VRF4-4	This object is used to set an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4404	MULTI_STATE_INPUT	ModeStatus_VRF4-4	This object is used to monitor an indoor unit's operation mode. 1: cool; 2: heat; 3: fan	3
4405	ANALOG_VALUE	TempSPSetting_VRF4-4	This object is used to set the indoor unit's setpoint.	24
4406	ANALOG_INPUT	TempSPStatus_VRF4-4	This object is used to monitor the indoor unit's setpoint.	24
4407	ANALOG_INPUT	RoomTemp_VRF4-4	This object is used to monitor the room dry-bulb temperature detected by the indoor unit return air sensor.	24
4408	ANALOG_INPUT	RoomRHmid_VRF4-4	This object is used to monitor the room relative humidity detected by the indoor unit return air sensor.	50
4409	MULTI_STATE_OUTPUT	AirFlowRateCommand_VRF4-4	This object is used to set an indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4410	MULTI_STATE_INPUT	AirFlowRateStatus_VRF4-4	This object is used to monitor the indoor unit's fan speed. 1: Low; 2: Middle; 3: High	2
4411	MULTI_STATE_OUTPUT	AirDirectionCommand_VRF4-4	This object is used to change the indoor unit's airflow direction. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4412	MULTI_STATE_INPUT	AirDirectionStatus_VRF4-4	This object is used to monitor the indoor unit’s airflow direction.. 1: Horizontal; 2: 22.5deg; 3: 45deg; 4: 67.5deg; 5: Vertical	5
4413	BINARY_VALUE	RemoteControlStart_VRF4-4	This object is used to permit or prohibit the On/Off operation from the remote controller.	0
4414	BINARY_INPUT	RemoteControlStart_VRF4-4	This object is used to monitor status of permit or prohibit the On/Off operation from the remote controller.	0
4421	ANALOG_INPUT	Electricity_VRF4-4	This object is used to monitor the indoor unit's electric consumption.	0
4422	ANALOG_INPUT	HeatLoad_VRF4-4	This object is used to monitor the heat load of indoor unit.	0

3) Objects in the “EnvironmentMonitor” device

The formula for calculating the instance number is as follows

Dry-bulb temperature = 1000 × outdoor unit index + 100 × indoor unit index + 1

Relative humidity = 1000 × outdoor unit index + 100 × indoor unit index + 2

Inst. No.	Type	Name	Description	Initial value
1	ANALOG_INPUT	Outdoor_DBT	Outdoor dry-bulb temperature.	25
2	ANALOG_INPUT	Outdoor_RHMD	Outdoor relative humidity.	50
3	ANALOG_INPUT	G_Radiation	Global horizontal radiation.	0
4	ANALOG_INPUT	N_adiation	Nocturnal radiation.	0
1101	ANALOG_INPUT	DBT_VRF1-1	Dry-bulb temperature of zone at VRF1-1.	25
1102	ANALOG_INPUT	RHMD_VRF1-1	Relative humidity of zone at VRF1-1.	50
1201	ANALOG_INPUT	DBT_VRF1-2	Dry-bulb temperature of zone at VRF1-2.	25
1202	ANALOG_INPUT	RHMD_VRF1-2	Relative humidity of zone at VRF1-2.	50
1301	ANALOG_INPUT	DBT_VRF1-3	Dry-bulb temperature of zone at VRF1-3.	25
1302	ANALOG_INPUT	RHMD_VRF1-3	Relative humidity of zone at VRF1-3.	50
1401	ANALOG_INPUT	DBT_VRF1-4	Dry-bulb temperature of zone at VRF1-4.	25
1402	ANALOG_INPUT	RHMD_VRF1-4	Relative humidity of zone at VRF1-4.	50
1501	ANALOG_INPUT	DBT_VRF1-5	Dry-bulb temperature of zone at VRF1-5.	25
1502	ANALOG_INPUT	RHMD_VRF1-5	Relative humidity of zone at VRF1-5.	50
2101	ANALOG_INPUT	DBT_VRF2-1	Dry-bulb temperature of zone at VRF2-1.	25
2102	ANALOG_INPUT	RHMD_VRF2-1	Relative humidity of zone at VRF2-1.	50
2201	ANALOG_INPUT	DBT_VRF2-2	Dry-bulb temperature of zone at VRF2-2.	25
2202	ANALOG_INPUT	RHMD_VRF2-2	Relative humidity of zone at VRF2-2.	50
2301	ANALOG_INPUT	DBT_VRF2-3	Dry-bulb temperature of zone at VRF2-3.	25
2302	ANALOG_INPUT	RHMD_VRF2-3	Relative humidity of zone at VRF2-3.	50
2401	ANALOG_INPUT	DBT_VRF2-4	Dry-bulb temperature of zone at VRF2-4.	25
2402	ANALOG_INPUT	RHMD_VRF2-4	Relative humidity of zone at VRF2-4.	50

3101	ANALOG_INPUT	DBT_VRF3-1	Dry-bulb temperature of zone at VRF3-1.	25
3102	ANALOG_INPUT	RHMD_VRF3-1	Relative humidity of zone at VRF3-1.	50
3201	ANALOG_INPUT	DBT_VRF3-2	Dry-bulb temperature of zone at VRF3-2.	25
3202	ANALOG_INPUT	RHMD_VRF3-2	Relative humidity of zone at VRF3-2.	50
3301	ANALOG_INPUT	DBT_VRF3-3	Dry-bulb temperature of zone at VRF3-3.	25
3302	ANALOG_INPUT	RHMD_VRF3-3	Relative humidity of zone at VRF3-3.	50
3401	ANALOG_INPUT	DBT_VRF3-4	Dry-bulb temperature of zone at VRF3-4.	25
3402	ANALOG_INPUT	RHMD_VRF3-4	Relative humidity of zone at VRF3-4.	50
3501	ANALOG_INPUT	DBT_VRF3-5	Dry-bulb temperature of zone at VRF3-5.	25
3502	ANALOG_INPUT	RHMD_VRF3-5	Relative humidity of zone at VRF3-5.	50
4101	ANALOG_INPUT	DBT_VRF4-1	Dry-bulb temperature of zone at VRF4-1.	25
4102	ANALOG_INPUT	RHMD_VRF4-1	Relative humidity of zone at VRF4-1.	50
4201	ANALOG_INPUT	DBT_VRF4-2	Dry-bulb temperature of zone at VRF4-2.	25
4202	ANALOG_INPUT	RHMD_VRF4-2	Relative humidity of zone at VRF4-2.	50
4301	ANALOG_INPUT	DBT_VRF4-3	Dry-bulb temperature of zone at VRF4-3.	25
4302	ANALOG_INPUT	RHMD_VRF4-3	Relative humidity of zone at VRF4-3.	50
4401	ANALOG_INPUT	DBT_VRF4-4	Dry-bulb temperature of zone at VRF4-4.	25
4402	ANALOG_INPUT	RHMD_VRF4-4	Relative humidity of zone at VRF4-4.	50

4) Objects in the “OccupantMonitor” device

The formula for calculating the instance number is as follows

Occupant number of zone = 10000 x tenant index + 1000 x zone index + 1

Average thermal sensation of occupants in zone = 10000 x tenant index + 1000 × zone index + 3.

Average clo value of occupants staying in zone = 10000 x tenant index + 1000 × zone index + 4

Presence or absence of an occupant = 10000 x tenant index + 10 × occupant index + 2

Thermal sensation of occupant = 10000 x tenant index + 10 × occupant index + 3.

Clo value of occupant = 10000 x tenant index + 10 × occupant index + 4.

The following example shows the value when the random number seed (rseed_oprm) for the occupants is set to 1: Changing the random number seed changes the list of occupants.

Inst. No.	Type	Name	Description	Initial value
10001	ANALOG_INPUT	Occupant number	Number of occupants stay in office (tenant-1).	0
11001	ANALOG_INPUT	Occupant number_ZN1_TNT1	Number of occupants stay in zone-1 of tenant-1	0
11003	ANALOG_INPUT	Ave_T_Sensation_ZN1_TNT1	Averaged thermal sensation of zone-1 of tenant-1	0
11004	ANALOG_INPUT	Ave_Clo_ZN1_TNT1	Averaged clothing index of zone-1 of tenant-1	0
12001	ANALOG_INPUT	Occupant number_ZN2_TNT1	Number of occupants stay in zone-2 of tenant-1	0
12003	ANALOG_INPUT	Ave_T_Sensation_ZN2_TNT1	Averaged thermal sensation of zone-2 of tenant-1	0
12004	ANALOG_INPUT	Ave_Clo_ZN2_TNT1	Averaged clothing index of zone-2 of tenant-1	0

13001	ANALOG_INPUT	Occupant number_ZN3_TNT1	Number of occupants stay in zone-3 of tenant-1	0
13003	ANALOG_INPUT	Ave_T_Sensation_ZN3_TNT1	Averaged thermal sensation of zone-3 of tenant-1	0
13004	ANALOG_INPUT	Ave_Clo_ZN3_TNT1	Averaged clothing index of zone-3 of tenant-1	0
14001	ANALOG_INPUT	Occupant number_ZN4_TNT1	Number of occupants stay in zone-4 of tenant-1	0
14003	ANALOG_INPUT	Ave_T_Sensation_ZN4_TNT1	Averaged thermal sensation of zone-4 of tenant-1	0
14004	ANALOG_INPUT	Ave_Clo_ZN4_TNT1	Averaged clothing index of zone-4 of tenant-1	0
15001	ANALOG_INPUT	Occupant number_ZN5_TNT1	Number of occupants stay in zone-5 of tenant-1	0
15003	ANALOG_INPUT	Ave_T_Sensation_ZN5_TNT1	Averaged thermal sensation of zone-5 of tenant-1	0
15004	ANALOG_INPUT	Ave_Clo_ZN5_TNT1	Averaged clothing index of zone-5 of tenant-1	0
16001	ANALOG_INPUT	Occupant number_ZN6_TNT1	Number of occupants stay in zone-6 of tenant-1	0
16003	ANALOG_INPUT	Ave_T_Sensation_ZN6_TNT1	Averaged thermal sensation of zone-6 of tenant-1	0
16004	ANALOG_INPUT	Ave_Clo_ZN6_TNT1	Averaged clothing index of zone-6 of tenant-1	0
17001	ANALOG_INPUT	Occupant number_ZN7_TNT1	Number of occupants stay in zone-7 of tenant-1	0
17003	ANALOG_INPUT	Ave_T_Sensation_ZN7_TNT1	Averaged thermal sensation of zone-7 of tenant-1	0
17004	ANALOG_INPUT	Ave_Clo_ZN7_TNT1	Averaged clothing index of zone-7 of tenant-1	0
18001	ANALOG_INPUT	Occupant number_ZN8_TNT1	Number of occupants stay in zone-8 of tenant-1	0
18003	ANALOG_INPUT	Ave_T_Sensation_ZN8_TNT1	Averaged thermal sensation of zone-8 of tenant-1	0
18004	ANALOG_INPUT	Ave_Clo_ZN8_TNT1	Averaged clothing index of zone-8 of tenant-1	0
19001	ANALOG_INPUT	Occupant number_ZN9_TNT1	Number of occupants stay in zone-9 of tenant-1	0
19003	ANALOG_INPUT	Ave_T_Sensation_ZN9_TNT1	Averaged thermal sensation of zone-9 of tenant-1	0
19004	ANALOG_INPUT	Ave_Clo_ZN9_TNT1	Averaged clothing index of zone-9 of tenant-1	0
10012	BINARY_INPUT	Availability_OC_1	Availability of occupant-1 of tenant-1 (Dana Hattersley)	0
10013	ANALOG_INPUT	T_Sensation_OC_1	Thermal sensation of occupant-1 of tenant-1 (Dana Hattersley)	0
10014	ANALOG_INPUT	Clo_OC_1	Clothing index of occupant-1 of tenant-1 (Dana Hattersley)	0
10022	BINARY_INPUT	Availability_OC_2	Availability of occupant-2 of tenant-1 (Humphrey Lock)	0
10023	ANALOG_INPUT	T_Sensation_OC_2	Thermal sensation of occupant-2 of tenant-1 (Humphrey Lock)	0
10024	ANALOG_INPUT	Clo_OC_2	Clothing index of occupant-2 of tenant-1 (Humphrey Lock)	0
10032	BINARY_INPUT	Availability_OC_3	Availability of occupant-3 of tenant-1 (Cassie Harris)	0
10033	ANALOG_INPUT	T_Sensation_OC_3	Thermal sensation of occupant-3 of tenant-1 (Cassie Harris)	0
10034	ANALOG_INPUT	Clo_OC_3	Clothing index of occupant-3 of tenant-1 (Cassie Harris)	0
10042	BINARY_INPUT	Availability_OC_4	Availability of occupant-4 of tenant-1 (Cecil Topping)	0
10043	ANALOG_INPUT	T_Sensation_OC_4	Thermal sensation of occupant-4 of tenant-1 (Cecil Topping)	0
10044	ANALOG_INPUT	Clo_OC_4	Clothing index of occupant-4 of tenant-1 (Cecil Topping)	0
10052	BINARY_INPUT	Availability_OC_5	Availability of occupant-5 of tenant-1 (Laila Black)	0
10053	ANALOG_INPUT	T_Sensation_OC_5	Thermal sensation of occupant-5 of tenant-1 (Laila Black)	0
10054	ANALOG_INPUT	Clo_OC_5	Clothing index of occupant-5 of tenant-1 (Laila Black)	0
10062	BINARY_INPUT	Availability_OC_6	Availability of occupant-6 of tenant-1 (Clive Toolson)	0
10063	ANALOG_INPUT	T_Sensation_OC_6	Thermal sensation of occupant-6 of tenant-1 (Clive Toolson)	0
10064	ANALOG_INPUT	Clo_OC_6	Clothing index of occupant-6 of tenant-1 (Clive Toolson)	0
10072	BINARY_INPUT	Availability_OC_7	Availability of occupant-7 of tenant-1 (Monique Cartwright)	0
10073	ANALOG_INPUT	T_Sensation_OC_7	Thermal sensation of occupant-7 of tenant-1 (Monique Cartwright)	0
10074	ANALOG_INPUT	Clo_OC_7	Clothing index of occupant-7 of tenant-1 (Monique Cartwright)	0
10082	BINARY_INPUT	Availability_OC_8	Availability of occupant-8 of tenant-1 (Josiah Conder)	0
10083	ANALOG_INPUT	T_Sensation_OC_8	Thermal sensation of occupant-8 of tenant-1 (Josiah Conder)	0
10084	ANALOG_INPUT	Clo_OC_8	Clothing index of occupant-8 of tenant-1 (Josiah Conder)	0
10092	BINARY_INPUT	Availability_OC_9	Availability of occupant-9 of tenant-1 (Phil Barker)	0
10093	ANALOG_INPUT	T_Sensation_OC_9	Thermal sensation of occupant-9 of tenant-1 (Phil Barker)	0
10094	ANALOG_INPUT	Clo_OC_9	Clothing index of occupant-9 of tenant-1 (Phil Barker)	0

10102	BINARY_INPUT	Availability_OC_10	Availability of occupant-10 of tenant-1 (Meredith Baldridge)	0
10103	ANALOG_INPUT	T_Sensation_OC_10	Thermal sensation of occupant-10 of tenant-1 (Meredith Baldridge)	0
10104	ANALOG_INPUT	Clo_OC_10	Clothing index of occupant-10 of tenant-1 (Meredith Baldridge)	0
10112	BINARY_INPUT	Availability_OC_11	Availability of occupant-11 of tenant-1 (Angelica Roundell)	0
10113	ANALOG_INPUT	T_Sensation_OC_11	Thermal sensation of occupant-11 of tenant-1 (Angelica Roundell)	0
10114	ANALOG_INPUT	Clo_OC_11	Clothing index of occupant-11 of tenant-1 (Angelica Roundell)	0
10122	BINARY_INPUT	Availability_OC_12	Availability of occupant-12 of tenant-1 (Hermann Rietschel)	0
10123	ANALOG_INPUT	T_Sensation_OC_12	Thermal sensation of occupant-12 of tenant-1 (Hermann Rietschel)	0
10124	ANALOG_INPUT	Clo_OC_12	Clothing index of occupant-12 of tenant-1 (Hermann Rietschel)	0
10132	BINARY_INPUT	Availability_OC_13	Availability of occupant-13 of tenant-1 (Allyn Galbraith)	0
10133	ANALOG_INPUT	T_Sensation_OC_13	Thermal sensation of occupant-13 of tenant-1 (Allyn Galbraith)	0
10134	ANALOG_INPUT	Clo_OC_13	Clothing index of occupant-13 of tenant-1 (Allyn Galbraith)	0
10142	BINARY_INPUT	Availability_OC_14	Availability of occupant-14 of tenant-1 (Wallace Sabine)	0
10143	ANALOG_INPUT	T_Sensation_OC_14	Thermal sensation of occupant-14 of tenant-1 (Wallace Sabine)	0
10144	ANALOG_INPUT	Clo_OC_14	Clothing index of occupant-14 of tenant-1 (Wallace Sabine)	0
10152	BINARY_INPUT	Availability_OC_15	Availability of occupant-15 of tenant-1 (David Midwinter)	0
10153	ANALOG_INPUT	T_Sensation_OC_15	Thermal sensation of occupant-15 of tenant-1 (David Midwinter)	0
10154	ANALOG_INPUT	Clo_OC_15	Clothing index of occupant-15 of tenant-1 (David Midwinter)	0
10162	BINARY_INPUT	Availability_OC_16	Availability of occupant-16 of tenant-1 (Rowland Rouse)	0
10163	ANALOG_INPUT	T_Sensation_OC_16	Thermal sensation of occupant-16 of tenant-1 (Rowland Rouse)	0
10164	ANALOG_INPUT	Clo_OC_16	Clothing index of occupant-16 of tenant-1 (Rowland Rouse)	0
10172	BINARY_INPUT	Availability_OC_17	Availability of occupant-17 of tenant-1 (Yuichiro Iio)	0
10173	ANALOG_INPUT	T_Sensation_OC_17	Thermal sensation of occupant-17 of tenant-1 (Yuichiro Iio)	0
10174	ANALOG_INPUT	Clo_OC_17	Clothing index of occupant-17 of tenant-1 (Yuichiro Iio)	0
10182	BINARY_INPUT	Availability_OC_18	Availability of occupant-18 of tenant-1 (Zachariah Venables-Vernon-Harcourt)	0
10183	ANALOG_INPUT	T_Sensation_OC_18	Thermal sensation of occupant-18 of tenant-1 (Zachariah Venables-Vernon-Harcourt)	0
10184	ANALOG_INPUT	Clo_OC_18	Clothing index of occupant-18 of tenant-1 (Zachariah Venables-Vernon-Harcourt)	0
10192	BINARY_INPUT	Availability_OC_19	Availability of occupant-19 of tenant-1 (Allyn Lympany)	0
10193	ANALOG_INPUT	T_Sensation_OC_19	Thermal sensation of occupant-19 of tenant-1 (Allyn Lympany)	0
10194	ANALOG_INPUT	Clo_OC_19	Clothing index of occupant-19 of tenant-1 (Allyn Lympany)	0
10202	BINARY_INPUT	Availability_OC_20	Availability of occupant-20 of tenant-1 (Daiki Kobayashi)	0
10203	ANALOG_INPUT	T_Sensation_OC_20	Thermal sensation of occupant-20 of tenant-1 (Daiki Kobayashi)	0
10204	ANALOG_INPUT	Clo_OC_20	Clothing index of occupant-20 of tenant-1 (Daiki Kobayashi)	0
10212	BINARY_INPUT	Availability_OC_21	Availability of occupant-21 of tenant-1 (Yvonne Murrills)	0
10213	ANALOG_INPUT	T_Sensation_OC_21	Thermal sensation of occupant-21 of tenant-1 (Yvonne Murrills)	0
10214	ANALOG_INPUT	Clo_OC_21	Clothing index of occupant-21 of tenant-1 (Yvonne Murrills)	0
10222	BINARY_INPUT	Availability_OC_22	Availability of occupant-22 of tenant-1 (Vince Cok)	0
10223	ANALOG_INPUT	T_Sensation_OC_22	Thermal sensation of occupant-22 of tenant-1 (Vince Cok)	0
10224	ANALOG_INPUT	Clo_OC_22	Clothing index of occupant-22 of tenant-1 (Vince Cok)	0
10232	BINARY_INPUT	Availability_OC_23	Availability of occupant-23 of tenant-1 (Niccolo Giannetti)	0
10233	ANALOG_INPUT	T_Sensation_OC_23	Thermal sensation of occupant-23 of tenant-1 (Niccolo Giannetti)	0
10234	ANALOG_INPUT	Clo_OC_23	Clothing index of occupant-23 of tenant-1 (Niccolo Giannetti)	0
10242	BINARY_INPUT	Availability_OC_24	Availability of occupant-24 of tenant-1 (Elizabeth Roundell)	0
10243	ANALOG_INPUT	T_Sensation_OC_24	Thermal sensation of occupant-24 of tenant-1 (Elizabeth Roundell)	0
10244	ANALOG_INPUT	Clo_OC_24	Clothing index of occupant-24 of tenant-1 (Elizabeth Roundell)	0
10252	BINARY_INPUT	Availability_OC_25	Availability of occupant-25 of tenant-1 (Nicola Turnbull)	0
10253	ANALOG_INPUT	T_Sensation_OC_25	Thermal sensation of occupant-25 of tenant-1 (Nicola Turnbull)	0
10254	ANALOG_INPUT	Clo_OC_25	Clothing index of occupant-25 of tenant-1 (Nicola Turnbull)	0

10262	BINARY_INPUT	Availability_OC_26	Availability of occupant-26 of tenant-1 (Masahi Momota)	0
10263	ANALOG_INPUT	T_Sensation_OC_26	Thermal sensation of occupant-26 of tenant-1 (Masahi Momota)	0
10264	ANALOG_INPUT	Clo_OC_26	Clothing index of occupant-26 of tenant-1 (Masahi Momota)	0
10272	BINARY_INPUT	Availability_OC_27	Availability of occupant-27 of tenant-1 (Jade Mollison)	0
10273	ANALOG_INPUT	T_Sensation_OC_27	Thermal sensation of occupant-27 of tenant-1 (Jade Mollison)	0
10274	ANALOG_INPUT	Clo_OC_27	Clothing index of occupant-27 of tenant-1 (Jade Mollison)	0
10282	BINARY_INPUT	Availability_OC_28	Availability of occupant-28 of tenant-1 (Linus Hanley)	0
10283	ANALOG_INPUT	T_Sensation_OC_28	Thermal sensation of occupant-28 of tenant-1 (Linus Hanley)	0
10284	ANALOG_INPUT	Clo_OC_28	Clothing index of occupant-28 of tenant-1 (Linus Hanley)	0
10292	BINARY_INPUT	Availability_OC_29	Availability of occupant-29 of tenant-1 (Valentine Elliston)	0
10293	ANALOG_INPUT	T_Sensation_OC_29	Thermal sensation of occupant-29 of tenant-1 (Valentine Elliston)	0
10294	ANALOG_INPUT	Clo_OC_29	Clothing index of occupant-29 of tenant-1 (Valentine Elliston)	0
10302	BINARY_INPUT	Availability_OC_30	Availability of occupant-30 of tenant-1 (Roman Steele)	0
10303	ANALOG_INPUT	T_Sensation_OC_30	Thermal sensation of occupant-30 of tenant-1 (Roman Steele)	0
10304	ANALOG_INPUT	Clo_OC_30	Clothing index of occupant-30 of tenant-1 (Roman Steele)	0
10312	BINARY_INPUT	Availability_OC_31	Availability of occupant-31 of tenant-1 (Savannah Biggs)	0
10313	ANALOG_INPUT	T_Sensation_OC_31	Thermal sensation of occupant-31 of tenant-1 (Savannah Biggs)	0
10314	ANALOG_INPUT	Clo_OC_31	Clothing index of occupant-31 of tenant-1 (Savannah Biggs)	0
10322	BINARY_INPUT	Availability_OC_32	Availability of occupant-32 of tenant-1 (Howard Astley)	0
10323	ANALOG_INPUT	T_Sensation_OC_32	Thermal sensation of occupant-32 of tenant-1 (Howard Astley)	0
10324	ANALOG_INPUT	Clo_OC_32	Clothing index of occupant-32 of tenant-1 (Howard Astley)	0
10332	BINARY_INPUT	Availability_OC_33	Availability of occupant-33 of tenant-1 (Masato Miyata)	0
10333	ANALOG_INPUT	T_Sensation_OC_33	Thermal sensation of occupant-33 of tenant-1 (Masato Miyata)	0
10334	ANALOG_INPUT	Clo_OC_33	Clothing index of occupant-33 of tenant-1 (Masato Miyata)	0
10342	BINARY_INPUT	Availability_OC_34	Availability of occupant-34 of tenant-1 (Aileen Winder)	0
10343	ANALOG_INPUT	T_Sensation_OC_34	Thermal sensation of occupant-34 of tenant-1 (Aileen Winder)	0
10344	ANALOG_INPUT	Clo_OC_34	Clothing index of occupant-34 of tenant-1 (Aileen Winder)	0
10352	BINARY_INPUT	Availability_OC_35	Availability of occupant-35 of tenant-1 (Landon Ackroyd)	0
10353	ANALOG_INPUT	T_Sensation_OC_35	Thermal sensation of occupant-35 of tenant-1 (Landon Ackroyd)	0
10354	ANALOG_INPUT	Clo_OC_35	Clothing index of occupant-35 of tenant-1 (Landon Ackroyd)	0
10362	BINARY_INPUT	Availability_OC_36	Availability of occupant-36 of tenant-1 (Leo Quantrill)	0
10363	ANALOG_INPUT	T_Sensation_OC_36	Thermal sensation of occupant-36 of tenant-1 (Leo Quantrill)	0
10364	ANALOG_INPUT	Clo_OC_36	Clothing index of occupant-36 of tenant-1 (Leo Quantrill)	0
10372	BINARY_INPUT	Availability_OC_37	Availability of occupant-37 of tenant-1 (Eisuke Togashi)	0
10373	ANALOG_INPUT	T_Sensation_OC_37	Thermal sensation of occupant-37 of tenant-1 (Eisuke Togashi)	0
10374	ANALOG_INPUT	Clo_OC_37	Clothing index of occupant-37 of tenant-1 (Eisuke Togashi)	0
10382	BINARY_INPUT	Availability_OC_38	Availability of occupant-38 of tenant-1 (Wilhelmina Chalmers)	0
10383	ANALOG_INPUT	T_Sensation_OC_38	Thermal sensation of occupant-38 of tenant-1 (Wilhelmina Chalmers)	0
10384	ANALOG_INPUT	Clo_OC_38	Clothing index of occupant-38 of tenant-1 (Wilhelmina Chalmers)	0
20001	ANALOG_INPUT	Occupant number	Number of occupants stay in office (tenant-2).	0
21001	ANALOG_INPUT	Occupant number_ZN1_TNT2	Number of occupants stay in zone-1 of tenant-2	0
21003	ANALOG_INPUT	Ave_T_Sensation_ZN1_TNT2	Averaged thermal sensation of zone-1 of tenant-2	0
21004	ANALOG_INPUT	Ave_Clo_ZN1_TNT2	Averaged clothing index of zone-1 of tenant-2	0
22001	ANALOG_INPUT	Occupant number_ZN2_TNT2	Number of occupants stay in zone-2 of tenant-2	0
22003	ANALOG_INPUT	Ave_T_Sensation_ZN2_TNT2	Averaged thermal sensation of zone-2 of tenant-2	0
22004	ANALOG_INPUT	Ave_Clo_ZN2_TNT2	Averaged clothing index of zone-2 of tenant-2	0
23001	ANALOG_INPUT	Occupant number_ZN3_TNT2	Number of occupants stay in zone-3 of tenant-2	0
23003	ANALOG_INPUT	Ave_T_Sensation_ZN3_TNT2	Averaged thermal sensation of zone-3 of tenant-2	0

23004	ANALOG_INPUT	Ave_Clo_ZN3_TNT2	Averaged clothing index of zone-3 of tenant-2	0
24001	ANALOG_INPUT	Occupant number_ZN4_TNT2	Number of occupants stay in zone-4 of tenant-2	0
24003	ANALOG_INPUT	Ave_T_Sensation_ZN4_TNT2	Averaged thermal sensation of zone-4 of tenant-2	0
24004	ANALOG_INPUT	Ave_Clo_ZN4_TNT2	Averaged clothing index of zone-4 of tenant-2	0
25001	ANALOG_INPUT	Occupant number_ZN5_TNT2	Number of occupants stay in zone-5 of tenant-2	0
25003	ANALOG_INPUT	Ave_T_Sensation_ZN5_TNT2	Averaged thermal sensation of zone-5 of tenant-2	0
25004	ANALOG_INPUT	Ave_Clo_ZN5_TNT2	Averaged clothing index of zone-5 of tenant-2	0
26001	ANALOG_INPUT	Occupant number_ZN6_TNT2	Number of occupants stay in zone-6 of tenant-2	0
26003	ANALOG_INPUT	Ave_T_Sensation_ZN6_TNT2	Averaged thermal sensation of zone-6 of tenant-2	0
26004	ANALOG_INPUT	Ave_Clo_ZN6_TNT2	Averaged clothing index of zone-6 of tenant-2	0
27001	ANALOG_INPUT	Occupant number_ZN7_TNT2	Number of occupants stay in zone-7 of tenant-2	0
27003	ANALOG_INPUT	Ave_T_Sensation_ZN7_TNT2	Averaged thermal sensation of zone-7 of tenant-2	0
27004	ANALOG_INPUT	Ave_Clo_ZN7_TNT2	Averaged clothing index of zone-7 of tenant-2	0
28001	ANALOG_INPUT	Occupant number_ZN8_TNT2	Number of occupants stay in zone-8 of tenant-2	0
28003	ANALOG_INPUT	Ave_T_Sensation_ZN8_TNT2	Averaged thermal sensation of zone-8 of tenant-2	0
28004	ANALOG_INPUT	Ave_Clo_ZN8_TNT2	Averaged clothing index of zone-8 of tenant-2	0
29001	ANALOG_INPUT	Occupant number_ZN9_TNT2	Number of occupants stay in zone-9 of tenant-2	0
29003	ANALOG_INPUT	Ave_T_Sensation_ZN9_TNT2	Averaged thermal sensation of zone-9 of tenant-2	0
29004	ANALOG_INPUT	Ave_Clo_ZN9_TNT2	Averaged clothing index of zone-9 of tenant-2	0
20012	BINARY_INPUT	Availability_OC_1	Availability of occupant-1 of tenant-2 (Kim Collingwood)	0
20013	ANALOG_INPUT	T_Sensation_OC_1	Thermal sensation of occupant-1 of tenant-2 (Kim Collingwood)	0
20014	ANALOG_INPUT	Clo_OC_1	Clothing index of occupant-1 of tenant-2 (Kim Collingwood)	0
20022	BINARY_INPUT	Availability_OC_2	Availability of occupant-2 of tenant-2 (Takahiro Ueno)	0
20023	ANALOG_INPUT	T_Sensation_OC_2	Thermal sensation of occupant-2 of tenant-2 (Takahiro Ueno)	0
20024	ANALOG_INPUT	Clo_OC_2	Clothing index of occupant-2 of tenant-2 (Takahiro Ueno)	0
20032	BINARY_INPUT	Availability_OC_3	Availability of occupant-3 of tenant-2 (Kimberly Holder)	0
20033	ANALOG_INPUT	T_Sensation_OC_3	Thermal sensation of occupant-3 of tenant-2 (Kimberly Holder)	0
20034	ANALOG_INPUT	Clo_OC_3	Clothing index of occupant-3 of tenant-2 (Kimberly Holder)	0
20042	BINARY_INPUT	Availability_OC_4	Availability of occupant-4 of tenant-2 (Sophie Coffin)	0
20043	ANALOG_INPUT	T_Sensation_OC_4	Thermal sensation of occupant-4 of tenant-2 (Sophie Coffin)	0
20044	ANALOG_INPUT	Clo_OC_4	Clothing index of occupant-4 of tenant-2 (Sophie Coffin)	0
20052	BINARY_INPUT	Availability_OC_5	Availability of occupant-5 of tenant-2 (Rolla Carpenter)	0
20053	ANALOG_INPUT	T_Sensation_OC_5	Thermal sensation of occupant-5 of tenant-2 (Rolla Carpenter)	0
20054	ANALOG_INPUT	Clo_OC_5	Clothing index of occupant-5 of tenant-2 (Rolla Carpenter)	0
20062	BINARY_INPUT	Availability_OC_6	Availability of occupant-6 of tenant-2 (Pauline Gooding)	0
20063	ANALOG_INPUT	T_Sensation_OC_6	Thermal sensation of occupant-6 of tenant-2 (Pauline Gooding)	0
20064	ANALOG_INPUT	Clo_OC_6	Clothing index of occupant-6 of tenant-2 (Pauline Gooding)	0
20072	BINARY_INPUT	Availability_OC_7	Availability of occupant-7 of tenant-2 (Sei Nagashima)	0
20073	ANALOG_INPUT	T_Sensation_OC_7	Thermal sensation of occupant-7 of tenant-2 (Sei Nagashima)	0
20074	ANALOG_INPUT	Clo_OC_7	Clothing index of occupant-7 of tenant-2 (Sei Nagashima)	0
20082	BINARY_INPUT	Availability_OC_8	Availability of occupant-8 of tenant-2 (Louisa Street)	0
20083	ANALOG_INPUT	T_Sensation_OC_8	Thermal sensation of occupant-8 of tenant-2 (Louisa Street)	0
20084	ANALOG_INPUT	Clo_OC_8	Clothing index of occupant-8 of tenant-2 (Louisa Street)	0
20092	BINARY_INPUT	Availability_OC_9	Availability of occupant-9 of tenant-2 (Lindsay Buckler)	0
20093	ANALOG_INPUT	T_Sensation_OC_9	Thermal sensation of occupant-9 of tenant-2 (Lindsay Buckler)	0
20094	ANALOG_INPUT	Clo_OC_9	Clothing index of occupant-9 of tenant-2 (Lindsay Buckler)	0
20102	BINARY_INPUT	Availability_OC_10	Availability of occupant-10 of tenant-2 (Katsuyuki Edahiro)	0
20103	ANALOG_INPUT	T_Sensation_OC_10	Thermal sensation of occupant-10 of tenant-2 (Katsuyuki Edahiro)	0

20104	ANALOG_INPUT	Clo_OC_10	Clothing index of occupant-10 of tenant-2 (Katsuyuki Edahiro)	0
20112	BINARY_INPUT	Availability_OC_11	Availability of occupant-11 of tenant-2 (Carey Blanchfield)	0
20113	ANALOG_INPUT	T_Sensation_OC_11	Thermal sensation of occupant-11 of tenant-2 (Carey Blanchfield)	0
20114	ANALOG_INPUT	Clo_OC_11	Clothing index of occupant-11 of tenant-2 (Carey Blanchfield)	0
20122	BINARY_INPUT	Availability_OC_12	Availability of occupant-12 of tenant-2 (Cordelia Woodson)	0
20123	ANALOG_INPUT	T_Sensation_OC_12	Thermal sensation of occupant-12 of tenant-2 (Cordelia Woodson)	0
20124	ANALOG_INPUT	Clo_OC_12	Clothing index of occupant-12 of tenant-2 (Cordelia Woodson)	0
20132	BINARY_INPUT	Availability_OC_13	Availability of occupant-13 of tenant-2 (Theodore Place)	0
20133	ANALOG_INPUT	T_Sensation_OC_13	Thermal sensation of occupant-13 of tenant-2 (Theodore Place)	0
20134	ANALOG_INPUT	Clo_OC_13	Clothing index of occupant-13 of tenant-2 (Theodore Place)	0
20142	BINARY_INPUT	Availability_OC_14	Availability of occupant-14 of tenant-2 (Tomoya Katayama)	0
20143	ANALOG_INPUT	T_Sensation_OC_14	Thermal sensation of occupant-14 of tenant-2 (Tomoya Katayama)	0
20144	ANALOG_INPUT	Clo_OC_14	Clothing index of occupant-14 of tenant-2 (Tomoya Katayama)	0
20152	BINARY_INPUT	Availability_OC_15	Availability of occupant-15 of tenant-2 (Michaela Nutter)	0
20153	ANALOG_INPUT	T_Sensation_OC_15	Thermal sensation of occupant-15 of tenant-2 (Michaela Nutter)	0
20154	ANALOG_INPUT	Clo_OC_15	Clothing index of occupant-15 of tenant-2 (Michaela Nutter)	0
20162	BINARY_INPUT	Availability_OC_16	Availability of occupant-16 of tenant-2 (Hajime Ogata)	0
20163	ANALOG_INPUT	T_Sensation_OC_16	Thermal sensation of occupant-16 of tenant-2 (Hajime Ogata)	0
20164	ANALOG_INPUT	Clo_OC_16	Clothing index of occupant-16 of tenant-2 (Hajime Ogata)	0
20172	BINARY_INPUT	Availability_OC_17	Availability of occupant-17 of tenant-2 (Lewis Swaine)	0
20173	ANALOG_INPUT	T_Sensation_OC_17	Thermal sensation of occupant-17 of tenant-2 (Lewis Swaine)	0
20174	ANALOG_INPUT	Clo_OC_17	Clothing index of occupant-17 of tenant-2 (Lewis Swaine)	0
20182	BINARY_INPUT	Availability_OC_18	Availability of occupant-18 of tenant-2 (Valentine Wellington)	0
20183	ANALOG_INPUT	T_Sensation_OC_18	Thermal sensation of occupant-18 of tenant-2 (Valentine Wellington)	0
20184	ANALOG_INPUT	Clo_OC_18	Clothing index of occupant-18 of tenant-2 (Valentine Wellington)	0
20192	BINARY_INPUT	Availability_OC_19	Availability of occupant-19 of tenant-2 (Stephanie Hines)	0
20193	ANALOG_INPUT	T_Sensation_OC_19	Thermal sensation of occupant-19 of tenant-2 (Stephanie Hines)	0
20194	ANALOG_INPUT	Clo_OC_19	Clothing index of occupant-19 of tenant-2 (Stephanie Hines)	0
20202	BINARY_INPUT	Availability_OC_20	Availability of occupant-20 of tenant-2 (Leonard Hill)	0
20203	ANALOG_INPUT	T_Sensation_OC_20	Thermal sensation of occupant-20 of tenant-2 (Leonard Hill)	0
20204	ANALOG_INPUT	Clo_OC_20	Clothing index of occupant-20 of tenant-2 (Leonard Hill)	0
20212	BINARY_INPUT	Availability_OC_21	Availability of occupant-21 of tenant-2 (Hisao Ayame)	0
20213	ANALOG_INPUT	T_Sensation_OC_21	Thermal sensation of occupant-21 of tenant-2 (Hisao Ayame)	0
20214	ANALOG_INPUT	Clo_OC_21	Clothing index of occupant-21 of tenant-2 (Hisao Ayame)	0
20222	BINARY_INPUT	Availability_OC_22	Availability of occupant-22 of tenant-2 (Masanari Ukai)	0
20223	ANALOG_INPUT	T_Sensation_OC_22	Thermal sensation of occupant-22 of tenant-2 (Masanari Ukai)	0
20224	ANALOG_INPUT	Clo_OC_22	Clothing index of occupant-22 of tenant-2 (Masanari Ukai)	0
20232	BINARY_INPUT	Availability_OC_23	Availability of occupant-23 of tenant-2 (Pamela Stackhouse)	0
20233	ANALOG_INPUT	T_Sensation_OC_23	Thermal sensation of occupant-23 of tenant-2 (Pamela Stackhouse)	0
20234	ANALOG_INPUT	Clo_OC_23	Clothing index of occupant-23 of tenant-2 (Pamela Stackhouse)	0
20242	BINARY_INPUT	Availability_OC_24	Availability of occupant-24 of tenant-2 (William Trollope)	0
20243	ANALOG_INPUT	T_Sensation_OC_24	Thermal sensation of occupant-24 of tenant-2 (William Trollope)	0
20244	ANALOG_INPUT	Clo_OC_24	Clothing index of occupant-24 of tenant-2 (William Trollope)	0
20252	BINARY_INPUT	Availability_OC_25	Availability of occupant-25 of tenant-2 (Jasmine Flowers)	0
20253	ANALOG_INPUT	T_Sensation_OC_25	Thermal sensation of occupant-25 of tenant-2 (Jasmine Flowers)	0
20254	ANALOG_INPUT	Clo_OC_25	Clothing index of occupant-25 of tenant-2 (Jasmine Flowers)	0
20262	BINARY_INPUT	Availability_OC_26	Availability of occupant-26 of tenant-2 (Constantin Yaglou)	0
20263	ANALOG_INPUT	T_Sensation_OC_26	Thermal sensation of occupant-26 of tenant-2 (Constantin Yaglou)	0

20264	ANALOG_INPUT	Clo_OC_26	Clothing index of occupant-26 of tenant-2 (Constantin Yaglou)	0
20272	BINARY_INPUT	Availability_OC_27	Availability of occupant-27 of tenant-2 (Edwin Gwatkin)	0
20273	ANALOG_INPUT	T_Sensation_OC_27	Thermal sensation of occupant-27 of tenant-2 (Edwin Gwatkin)	0
20274	ANALOG_INPUT	Clo_OC_27	Clothing index of occupant-27 of tenant-2 (Edwin Gwatkin)	0
20282	BINARY_INPUT	Availability_OC_28	Availability of occupant-28 of tenant-2 (Jeff Northcutt)	0
20283	ANALOG_INPUT	T_Sensation_OC_28	Thermal sensation of occupant-28 of tenant-2 (Jeff Northcutt)	0
20284	ANALOG_INPUT	Clo_OC_28	Clothing index of occupant-28 of tenant-2 (Jeff Northcutt)	0
20292	BINARY_INPUT	Availability_OC_29	Availability of occupant-29 of tenant-2 (Pat Hightower)	0
20293	ANALOG_INPUT	T_Sensation_OC_29	Thermal sensation of occupant-29 of tenant-2 (Pat Hightower)	0
20294	ANALOG_INPUT	Clo_OC_29	Clothing index of occupant-29 of tenant-2 (Pat Hightower)	0
20302	BINARY_INPUT	Availability_OC_30	Availability of occupant-30 of tenant-2 (Brendon Byrd)	0
20303	ANALOG_INPUT	T_Sensation_OC_30	Thermal sensation of occupant-30 of tenant-2 (Brendon Byrd)	0
20304	ANALOG_INPUT	Clo_OC_30	Clothing index of occupant-30 of tenant-2 (Brendon Byrd)	0
20312	BINARY_INPUT	Availability_OC_31	Availability of occupant-31 of tenant-2 (Abel Cleverly)	0
20313	ANALOG_INPUT	T_Sensation_OC_31	Thermal sensation of occupant-31 of tenant-2 (Abel Cleverly)	0
20314	ANALOG_INPUT	Clo_OC_31	Clothing index of occupant-31 of tenant-2 (Abel Cleverly)	0
20322	BINARY_INPUT	Availability_OC_32	Availability of occupant-32 of tenant-2 (Daniel Calladine)	0
20323	ANALOG_INPUT	T_Sensation_OC_32	Thermal sensation of occupant-32 of tenant-2 (Daniel Calladine)	0
20324	ANALOG_INPUT	Clo_OC_32	Clothing index of occupant-32 of tenant-2 (Daniel Calladine)	0
20332	BINARY_INPUT	Availability_OC_33	Availability of occupant-33 of tenant-2 (Makoto Satoh)	0
20333	ANALOG_INPUT	T_Sensation_OC_33	Thermal sensation of occupant-33 of tenant-2 (Makoto Satoh)	0
20334	ANALOG_INPUT	Clo_OC_33	Clothing index of occupant-33 of tenant-2 (Makoto Satoh)	0
20342	BINARY_INPUT	Availability_OC_34	Availability of occupant-34 of tenant-2 (Walter Heston)	0
20343	ANALOG_INPUT	T_Sensation_OC_34	Thermal sensation of occupant-34 of tenant-2 (Walter Heston)	0
20344	ANALOG_INPUT	Clo_OC_34	Clothing index of occupant-34 of tenant-2 (Walter Heston)	0
20352	BINARY_INPUT	Availability_OC_35	Availability of occupant-35 of tenant-2 (Robin Hurst)	0
20353	ANALOG_INPUT	T_Sensation_OC_35	Thermal sensation of occupant-35 of tenant-2 (Robin Hurst)	0
20354	ANALOG_INPUT	Clo_OC_35	Clothing index of occupant-35 of tenant-2 (Robin Hurst)	0
20362	BINARY_INPUT	Availability_OC_36	Availability of occupant-36 of tenant-2 (Rick Dobbs)	0
20363	ANALOG_INPUT	T_Sensation_OC_36	Thermal sensation of occupant-36 of tenant-2 (Rick Dobbs)	0
20364	ANALOG_INPUT	Clo_OC_36	Clothing index of occupant-36 of tenant-2 (Rick Dobbs)	0
20372	BINARY_INPUT	Availability_OC_37	Availability of occupant-37 of tenant-2 (Oswald Coffin)	0
20373	ANALOG_INPUT	T_Sensation_OC_37	Thermal sensation of occupant-37 of tenant-2 (Oswald Coffin)	0
20374	ANALOG_INPUT	Clo_OC_37	Clothing index of occupant-37 of tenant-2 (Oswald Coffin)	0
20382	BINARY_INPUT	Availability_OC_38	Availability of occupant-38 of tenant-2 (Godfrey Doust)	0
20383	ANALOG_INPUT	T_Sensation_OC_38	Thermal sensation of occupant-38 of tenant-2 (Godfrey Doust)	0
20384	ANALOG_INPUT	Clo_OC_38	Clothing index of occupant-38 of tenant-2 (Godfrey Doust)	0
20392	BINARY_INPUT	Availability_OC_39	Availability of occupant-39 of tenant-2 (Hiroyuki Hatada)	0
20393	ANALOG_INPUT	T_Sensation_OC_39	Thermal sensation of occupant-39 of tenant-2 (Hiroyuki Hatada)	0
20394	ANALOG_INPUT	Clo_OC_39	Clothing index of occupant-39 of tenant-2 (Hiroyuki Hatada)	0
20402	BINARY_INPUT	Availability_OC_40	Availability of occupant-40 of tenant-2 (Lindsey Ottley)	0
20403	ANALOG_INPUT	T_Sensation_OC_40	Thermal sensation of occupant-40 of tenant-2 (Lindsey Ottley)	0
20404	ANALOG_INPUT	Clo_OC_40	Clothing index of occupant-40 of tenant-2 (Lindsey Ottley)	0
20412	BINARY_INPUT	Availability_OC_41	Availability of occupant-41 of tenant-2 (Malcolm Watt)	0
20413	ANALOG_INPUT	T_Sensation_OC_41	Thermal sensation of occupant-41 of tenant-2 (Malcolm Watt)	0
20414	ANALOG_INPUT	Clo_OC_41	Clothing index of occupant-41 of tenant-2 (Malcolm Watt)	0
20422	BINARY_INPUT	Availability_OC_42	Availability of occupant-42 of tenant-2 (Elton Vickers)	0
20423	ANALOG_INPUT	T_Sensation_OC_42	Thermal sensation of occupant-42 of tenant-2 (Elton Vickers)	0

20424	ANALOG_INPUT	Clo_OC_42	Clothing index of occupant-42 of tenant-2 (Elton Vickers)	0
20432	BINARY_INPUT	Availability_OC_43	Availability of occupant-43 of tenant-2 (Rodney Benge)	0
20433	ANALOG_INPUT	T_Sensation_OC_43	Thermal sensation of occupant-43 of tenant-2 (Rodney Benge)	0
20434	ANALOG_INPUT	Clo_OC_43	Clothing index of occupant-43 of tenant-2 (Rodney Benge)	0
20442	BINARY_INPUT	Availability_OC_44	Availability of occupant-44 of tenant-2 (Stanley Neilson)	0
20443	ANALOG_INPUT	T_Sensation_OC_44	Thermal sensation of occupant-44 of tenant-2 (Stanley Neilson)	0
20444	ANALOG_INPUT	Clo_OC_44	Clothing index of occupant-44 of tenant-2 (Stanley Neilson)	0
20452	BINARY_INPUT	Availability_OC_45	Availability of occupant-45 of tenant-2 (Willis Carrier)	0
20453	ANALOG_INPUT	T_Sensation_OC_45	Thermal sensation of occupant-45 of tenant-2 (Willis Carrier)	0
20454	ANALOG_INPUT	Clo_OC_45	Clothing index of occupant-45 of tenant-2 (Willis Carrier)	0
20462	BINARY_INPUT	Availability_OC_46	Availability of occupant-46 of tenant-2 (Emma Botting)	0
20463	ANALOG_INPUT	T_Sensation_OC_46	Thermal sensation of occupant-46 of tenant-2 (Emma Botting)	0
20464	ANALOG_INPUT	Clo_OC_46	Clothing index of occupant-46 of tenant-2 (Emma Botting)	0
20472	BINARY_INPUT	Availability_OC_47	Availability of occupant-47 of tenant-2 (Wanda Madgwick)	0
20473	ANALOG_INPUT	T_Sensation_OC_47	Thermal sensation of occupant-47 of tenant-2 (Wanda Madgwick)	0
20474	ANALOG_INPUT	Clo_OC_47	Clothing index of occupant-47 of tenant-2 (Wanda Madgwick)	0
20482	BINARY_INPUT	Availability_OC_48	Availability of occupant-48 of tenant-2 (Quincy Windsor-Clive)	0
20483	ANALOG_INPUT	T_Sensation_OC_48	Thermal sensation of occupant-48 of tenant-2 (Quincy Windsor-Clive)	0
20484	ANALOG_INPUT	Clo_OC_48	Clothing index of occupant-48 of tenant-2 (Quincy Windsor-Clive)	0

5) Objects in the “VentilationController” device

The formula for calculating the instance number is as follows:

On/off state = 1000 × outdoor unit index + 100 × indoor unit index + 3.

Enable bypass control = 1000 × outdoor unit index + 100 × indoor unit index + 4

Fan speed = 1000 × outdoor unit index + 100 × indoor unit index + 5.

Inst. No.	Type	Name	Description	Initial value
1	ANALOG_INPUT	CO2 level of south tenant	CO2 level of south tenant.	400
2	ANALOG_INPUT	CO2 level of north tenant	CO2 level of north tenant.	400
1103	BINARY_OUTPUT	On/Off setting/state (HEX1-1)	This object is used to control or monitor On/Off state of HEX1-1	0
1104	BINARY_OUTPUT	Bypass control setting/state (HEX1-1)	This object is used to control or monitor bypass control state of HEX1-1	0
1105	MULTI_STATE_OUTPUT	Fan speed (HEX1-1)	This object is used to control or monitor fan speed of HEX1-1. 1:Low; 2:Middle; 3:High	3
1203	BINARY_OUTPUT	On/Off setting/state (HEX1-2)	This object is used to control or monitor On/Off state of HEX1-2	0
1204	BINARY_OUTPUT	Bypass control setting/state (HEX1-2)	This object is used to control or monitor bypass control state of HEX1-2	0
1205	MULTI_STATE_OUTPUT	Fan speed (HEX1-2)	This object is used to control or monitor fan speed of HEX1-2. 1:Low; 2:Middle; 3:High	3
1303	BINARY_OUTPUT	On/Off setting/state (HEX1-3)	This object is used to control or monitor On/Off state of HEX1-3	0
1304	BINARY_OUTPUT	Bypass control setting/state (HEX1-3)	This object is used to control or monitor bypass control state of HEX1-3	0
1305	MULTI_STATE_OUTPUT	Fan speed (HEX1-3)	This object is used to control or monitor fan speed of HEX1-3. 1:Low; 2:Middle; 3:High	3
1403	BINARY_OUTPUT	On/Off setting/state (HEX1-4)	This object is used to control or monitor On/Off state of HEX1-4	0
1404	BINARY_OUTPUT	Bypass control setting/state (HEX1-4)	This object is used to control or monitor bypass control state of HEX1-4	0
1405	MULTI_STATE_OUTPUT	Fan speed (HEX1-4)	This object is used to control or monitor fan speed of HEX1-4. 1:Low; 2:Middle; 3:High	3
1503	BINARY_OUTPUT	On/Off setting/state (HEX1-5)	This object is used to control or monitor On/Off state of HEX1-5	0

1504	BINARY_OUTPUT	Bypass control setting/state (HEX1-5)	This object is used to control or monitor bypass control state of HEX1-5	0
1505	MULTI_STATE_OUTPUT	Fan speed (HEX1-5)	This object is used to control or monitor fan speed of HEX1-5. 1:Low; 2:Middle; 3:High	3
2103	BINARY_OUTPUT	On/Off setting/state (HEX2-1)	This object is used to control or monitor On/Off state of HEX2-1	0
2104	BINARY_OUTPUT	Bypass control setting/state (HEX2-1)	This object is used to control or monitor bypass control state of HEX2-1	0
2105	MULTI_STATE_OUTPUT	Fan speed (HEX2-1)	This object is used to control or monitor fan speed of HEX2-1. 1:Low; 2:Middle; 3:High	3
2203	BINARY_OUTPUT	On/Off setting/state (HEX2-2)	This object is used to control or monitor On/Off state of HEX2-2	0
2204	BINARY_OUTPUT	Bypass control setting/state (HEX2-2)	This object is used to control or monitor bypass control state of HEX2-2	0
2205	MULTI_STATE_OUTPUT	Fan speed (HEX2-2)	This object is used to control or monitor fan speed of HEX2-2. 1:Low; 2:Middle; 3:High	3
2303	BINARY_OUTPUT	On/Off setting/state (HEX2-3)	This object is used to control or monitor On/Off state of HEX2-3	0
2304	BINARY_OUTPUT	Bypass control setting/state (HEX2-3)	This object is used to control or monitor bypass control state of HEX2-3	0
2305	MULTI_STATE_OUTPUT	Fan speed (HEX2-3)	This object is used to control or monitor fan speed of HEX2-3. 1:Low; 2:Middle; 3:High	3
2403	BINARY_OUTPUT	On/Off setting/state (HEX2-4)	This object is used to control or monitor On/Off state of HEX2-4	0
2404	BINARY_OUTPUT	Bypass control setting/state (HEX2-4)	This object is used to control or monitor bypass control state of HEX2-4	0
2405	MULTI_STATE_OUTPUT	Fan speed (HEX2-4)	This object is used to control or monitor fan speed of HEX2-4. 1:Low; 2:Middle; 3:High	3
3103	BINARY_OUTPUT	On/Off setting/state (HEX3-1)	This object is used to control or monitor On/Off state of HEX3-1	0
3104	BINARY_OUTPUT	Bypass control setting/state (HEX3-1)	This object is used to control or monitor bypass control state of HEX3-1	0
3105	MULTI_STATE_OUTPUT	Fan speed (HEX3-1)	This object is used to control or monitor fan speed of HEX3-1. 1:Low; 2:Middle; 3:High	3
3203	BINARY_OUTPUT	On/Off setting/state (HEX3-2)	This object is used to control or monitor On/Off state of HEX3-2	0
3204	BINARY_OUTPUT	Bypass control setting/state (HEX3-2)	This object is used to control or monitor bypass control state of HEX3-2	0
3205	MULTI_STATE_OUTPUT	Fan speed (HEX3-2)	This object is used to control or monitor fan speed of HEX3-2. 1:Low; 2:Middle; 3:High	3
3303	BINARY_OUTPUT	On/Off setting/state (HEX3-3)	This object is used to control or monitor On/Off state of HEX3-3	0
3304	BINARY_OUTPUT	Bypass control setting/state (HEX3-3)	This object is used to control or monitor bypass control state of HEX3-3	0
3305	MULTI_STATE_OUTPUT	Fan speed (HEX3-3)	This object is used to control or monitor fan speed of HEX3-3. 1:Low; 2:Middle; 3:High	3
3403	BINARY_OUTPUT	On/Off setting/state (HEX3-4)	This object is used to control or monitor On/Off state of HEX3-4	0
3404	BINARY_OUTPUT	Bypass control setting/state (HEX3-4)	This object is used to control or monitor bypass control state of HEX3-4	0
3405	MULTI_STATE_OUTPUT	Fan speed (HEX3-4)	This object is used to control or monitor fan speed of HEX3-4. 1:Low; 2:Middle; 3:High	3
3503	BINARY_OUTPUT	On/Off setting/state (HEX3-5)	This object is used to control or monitor On/Off state of HEX3-5	0
3504	BINARY_OUTPUT	Bypass control setting/state (HEX3-5)	This object is used to control or monitor bypass control state of HEX3-5	0
3505	MULTI_STATE_OUTPUT	Fan speed (HEX3-5)	This object is used to control or monitor fan speed of HEX3-5. 1:Low; 2:Middle; 3:High	3
4103	BINARY_OUTPUT	On/Off setting/state (HEX4-1)	This object is used to control or monitor On/Off state of HEX4-1	0
4104	BINARY_OUTPUT	Bypass control setting/state (HEX4-1)	This object is used to control or monitor bypass control state of HEX4-1	0
4105	MULTI_STATE_OUTPUT	Fan speed (HEX4-1)	This object is used to control or monitor fan speed of HEX4-1. 1:Low; 2:Middle; 3:High	3
4203	BINARY_OUTPUT	On/Off setting/state (HEX4-2)	This object is used to control or monitor On/Off state of HEX4-2	0
4204	BINARY_OUTPUT	Bypass control setting/state (HEX4-2)	This object is used to control or monitor bypass control state of HEX4-2	0
4205	MULTI_STATE_OUTPUT	Fan speed (HEX4-2)	This object is used to control or monitor fan speed of HEX4-2. 1:Low; 2:Middle; 3:High	3
4303	BINARY_OUTPUT	On/Off setting/state (HEX4-3)	This object is used to control or monitor On/Off state of HEX4-3	0
4304	BINARY_OUTPUT	Bypass control setting/state (HEX4-3)	This object is used to control or monitor bypass control state of HEX4-3	0
4305	MULTI_STATE_OUTPUT	Fan speed (HEX4-3)	This object is used to control or monitor fan speed of HEX4-3. 1:Low; 2:Middle; 3:High	3
4403	BINARY_OUTPUT	On/Off setting/state (HEX4-4)	This object is used to control or monitor On/Off state of HEX4-4	0
4404	BINARY_OUTPUT	Bypass control setting/state (HEX4-4)	This object is used to control or monitor bypass control state of HEX4-4	0
4405	MULTI_STATE_OUTPUT	Fan speed (HEX4-4)	This object is used to control or monitor fan speed of HEX4-4. 1:Low; 2:Middle; 3:High	3

6) Objects in the "DummyDevice"

Inst. No.	Type	Name	Description	Initial value
1	ANALOG_VALUE	Analog value (int)	Dummy object to test communication of analog value (int).	1
2	ANALOG_OUTPUT	Analog output (int)	Dummy object to test communication of analog output (int).	2
3	ANALOG_INPUT	Analog input (int)	Dummy object to test communication of analog input (int).	3
4	ANALOG_VALUE	Analog value (float)	Dummy object to test communication of analog value (real).	4
5	ANALOG_OUTPUT	Analog output (float)	Dummy object to test communication of analog output (real).	5
6	ANALOG_INPUT	Analog input (float)	Dummy object to test communication of analog input (real).	6
7	BINARY_VALUE	Binary value	Dummy object to test communication of binary value.	0
8	BINARY_OUTPUT	Binary output	Dummy object to test communication of binary output.	0
9	BINARY_INPUT	Binary input	Dummy object to test communication of binary input.	0
10	MULTI_STATE_VALUE	Multistate value	Dummy object to test communication of multistate value.	1
11	MULTI_STATE_OUTPUT	Multistate output	Dummy object to test communication of multistate output.	2
12	MULTI_STATE_INPUT	Multistate input	Dummy object to test communication of multistate input.	3
13	DATETIME_VALUE	BACnet date time	Dummy object to test communication of bacnet date time.	1980/6/14 0:00

Appendix 2

Occupants

No	Tenant	Zone	First name	Last name	Age	Height	Weight	M/F
1	South	S1	Dana	Hattersley	45	160.9	69.4	F
2	South	S1	Humphrey	Lock	45	180.3	55.8	M
3	South	S1	Cassie	Harris	65	156.2	53.3	F
4	South	S2	Cecil	Topping	35	168.3	65.0	M
5	South	S2	Laila	Black	65	155.8	51.2	F
6	South	S2	Clive	Toolson	65	173.6	59.1	M
7	South	S3	Monique	Cartwright	25	159.3	50.2	F
8	South	S3	Josiah	Conder	55	170.0	72.0	M
9	South	S3	Phil	Barker	65	163.4	63.8	M
10	South	S4	Meredith	Baldridge	25	169.9	79.0	M
11	South	S4	Angelica	Roundell	35	164.0	51.3	F
12	South	S4	Hermann	Rietschel	35	172.3	66.2	M
13	South	S4	Allyn	Galbraith	45	172.0	66.2	M
14	South	S5	Wallace	Sabine	35	174.5	58.0	M
15	South	S5	David	Midwinter	45	165.2	77.2	M
16	South	S5	Rowland	Rouse	35	175.4	71.9	M
17	South	S5	Yuichiro	Iio	45	168.6	81.0	M
18	South	S6	Zachariah	Vernon	25	181.7	69.6	M
19	South	S6	Allyn	Lympany	65	149.0	58.2	F
20	South	S7	Daiki	Kobayashi	25	175.2	56.6	M
21	South	S7	Yvonne	Murrills	65	153.6	62.2	F
22	South	S7	Vince	Cok	55	162.8	66.6	M
23	South	S7	Niccolo	Giannetti	25	165.9	58.0	M
24	South	S7	Elizabeth	Roundell	65	153.0	62.4	F
25	South	S7	Nicola	Turnbull	45	160.3	61.6	F
26	South	S8	Masahi	Momota	55	156.5	67.3	M
27	South	S8	Jade	Mollison	65	153.7	64.0	F
28	South	S8	Linus	Hanley	45	160.9	77.5	M
29	South	S8	Valentine	Elliston	45	172.7	70.5	M
30	South	S8	Roman	Steele	45	173.4	68.8	M
31	South	S8	Savannah	Biggs	55	149.4	55.5	F
32	South	S8	Howard	Astley	25	173.5	72.0	M
33	South	S8	Masato	Miyata	35	179.6	64.4	M
34	South	S9	Aileen	Winder	45	153.8	60.6	F
35	South	S9	Landon	Ackroyd	25	173.2	49.9	M
36	South	S9	Leo	Quantrill	65	175.9	57.7	M
37	South	S9	Eisuke	Togashi	55	171.4	78.7	M
38	South	S9	Wilhelmina	Chalmers	35	160.5	57.0	F
39	North	N1	Kim	Collingwood	35	162.9	68.1	M
40	North	N1	Takahiro	Ueno	45	170.6	75.5	M
41	North	N1	Kimberly	Holder	25	157.5	44.1	F
42	North	N1	Sophie	Coffin	45	157.5	56.9	F
43	North	N1	Rolla	Carpenter	55	162.8	74.0	M
44	North	N2	Pauline	Gooding	35	164.5	48.0	F
45	North	N2	Sei	Nagashima	35	178.0	64.9	M
46	North	N2	Louisa	Street	45	156.7	40.3	F
47	North	N2	Lindsay	Buckler	25	157.1	46.9	F
48	North	N2	Katsuyuki	Edahiro	55	180.0	69.4	M
49	North	N3	Carey	Blanchfield	55	175.5	68.9	M
50	North	N3	Cordelia	Woodson	25	169.1	54.0	F
51	North	N3	Theodore	Place	35	172.7	70.7	M
52	North	N4	Tomoya	Katayama	45	171.3	67.1	M
53	North	N4	Michaela	Nutter	45	167.7	54.3	F
54	North	N4	Hajime	Ogata	65	158.1	69.7	M
55	North	N4	Lewis	Swaine	35	172.9	61.8	M
56	North	N4	Valentine	Wellington	45	170.7	73.6	M
57	North	N4	Stephanie	Hines	35	162.8	55.4	F
58	North	N4	Leonard	Hill	35	176.4	54.8	M
59	North	N5	Hisao	Ayame	35	180.8	67.0	M
60	North	N5	Masanari	Ukai	45	171.1	74.1	M
61	North	N5	Pamela	Stackhouse	45	164.6	53.4	F
62	North	N5	William	Trollope	35	179.1	35.1	M
63	North	N5	Jasmine	Flowers	65	160.4	44.1	F
64	North	N5	Constantin	Yaglou	35	164.2	73.1	M
65	North	N5	Edwin	Gwatkin	45	168.6	50.7	M
66	North	N6	Jeff	Northcutt	55	169.3	69.6	M
67	North	N6	Pat	Hightower	35	178.4	44.3	M
68	North	N6	Brendon	Byrd	25	170.5	71.0	M
69	North	N6	Abel	Cleverly	55	175.9	69.1	M
70	North	N6	Daniel	Calladine	35	167.9	66.2	M
71	North	N7	Makoto	Satoh	25	163.4	72.3	M
72	North	N7	Walter	Heston	35	176.1	80.4	M
73	North	N7	Robin	Hurst	25	177.7	60.6	M
74	North	N7	Rick	Dobbs	55	163.6	64.8	M
75	North	N8	Oswald	Coffin	45	168.7	59.0	M
76	North	N8	Godfrey	Doust	45	157.8	78.7	M
77	North	N8	Hiroyuki	Hatada	45	177.4	67.6	M
78	North	N8	Lindsey	Ottley	35	152.2	48.8	F
79	North	N8	Malcolm	Watt	35	167.6	74.0	M
70	North	N8	Elton	Vickers	45	179.8	64.3	M
81	North	N8	Rodney	Benge	35	169.4	69.9	M
82	North	N9	Stanley	Neilson	45	166.0	55.4	M
83	North	N9	Willis	Carrier	45	162.5	78.3	M
84	North	N9	Emma	Botting	45	165.5	51.0	F
85	North	N9	Wanda	Madgwick	35	150.4	48.6	F
86	North	N9	Quincy	Windsor-Clive	35	171.2	72.2	M

† Height, weight, and gender are just set for the fun of giving reality and do not affect the calculation results.