7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

AN EMBEDDED REAL-TIME SYSTEM FOR DATA PROCESSING, BASED ON BACnet DATA COMMUNICATION PROTOCOL

Iurie GUZUN, Viaceslav GUZUN
Amann GmbH Regelungstechnik,Unteranger Str. 6, 82041 Muenchen, Oberhaching, Germany
SiemensVDO Automotive
Calea Martirilor Str. 1, 300724 Timisoara, Romania
i.guzun@amann-net.de

Abstract. *The purpose of this paper is to present a part of a project, which was developed at AMANN GmbH Muenchen, Oberhaching, Germany. This is an embedded software, which is included in the new device EWMS View BACnet Touch Screen. The software development was done in ANSI C++, and Borland C++ 5.02 compiler was used. This project is based on services provided by BACnet. Embedded system is controlled by SMX real-time multitasking operating system, and as a hardware platform a 16-bit AMD 80C186EM microcontroller is used. The Touch Screen is a black/white one with a resolution of 320 by 240 pixels .*
BACnet is a data communication protocol for building automation and control networks. Developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineer (ASHRAE), its purpose is to standardize communication between building automation devices and systems from different manufacturers. BACnet affords facility owners and managers maximum flexibility and cost-effectiveness by allowing control products made by different manufacturers to be integrated into a single, uniform system. BACnet is designed to allow lighting, life safety, access, security, power, vertical transportation and other building system control devices to interoperate. BACnet allows intelligent systems from various industries and manufacturers to exchange information, such as temperatures, setpoints, schedules, trend logs and alarms, and coordinate equipment operation to achieve optimum building performance. It saves facility owners/managers the costs of purchasing, engineering and maintaining custom interfaces for supervising equipment, as well as integrating diverse building systems such as lighting, fire, elevators, electrical services, etc.
BACnet is based on a four-layer collapsed architecture for Local Area Networks, that corresponds to the physical, data link, network, and application layers of the OSI model.
Keywords: *embedded systems, BACnet protocol, data acquisition, building automation control, multitasking.*

Introduction

We will begin the presentation of our paper by introducing some theoretical aspects of the BACnet protocol.
BACnet provides mechanisms by which computerized equipment of arbitrary function may exchange information, regardless of the particular building service it performs. Each network device is modeled as a collection of network-accessible, named entities called "objects". Each object is characterized by a set of attributes or "properties".
Communication is accomplished by reading and writing the properties of particular objects and by the mutually acceptable execution of the protocol "services". The sophistication of a specific device, in terms of its ability to carry out particular service requests or to understand the nature of particular object types, is reflected in the device's "conformance class". Each class specifies a minimum set of services, objects, and properties the device must support in order to claim membership in a particular class. Because of the Standard's adherence to the ISO concept of "layered" communication architecture, the same message may be exchanged using various network access methods and physical media. This standard was specifically tailored for heating, ventilating, air-conditioning, and refrigerating control equipment, but it is also intended to provide a basis for integrating other kinds of building control systems, such as lighting, security, and fire detection systems. [1]

Scope of the BACnet protocol

This protocol provides a comprehensive set of messages for conveying encoded binary, analogue, and alphanumeric data between devices including, but not limited to:
hardware binary input and output values, hardware analogue input and output values, software binary and analogue values, text string values, scheduler information, trends, alarm and event notification, and control logic.
This protocol models each building automation and control computer as a collection of data structures – "objects", the properties of which represent various aspects of the hardware, software, and operation of the device. These objects provide a means of identifying and accessing information without requiring knowledge of the details of the device's internal design and configuration. [1]

Modelling Control devices as a collection of objects.

The data structures used in a device to store information are a local matter. In order to exchange that information with another device using this protocol, there must be a "network-visible" representation. This clause defines a set of standard objects types. These object types define an abstract data structure that provides a framework for building the application layer services. All objects are referenced by their Object_Identifier property. Each object within a single BACnet Device shall have a unique value for the Object_Identifier property. When combined with the system-wide unique Object_Identifier of the BACnet Device, this provides a mechanism for referencing every object in the control system network. Our project uses the following collection of objects:
Analogue Input, Analogue Output, and Analogue Value.
Binary Input, Binary Output, and Binary Value
Multi-state Input, Multi-state Output, and Multi-state Value, Schedule, Device Object Type. [1]
The Analogue Input object type defines a standardized object whose properties represent the externally visible characteristics of an analogue input. Respectively, the Analogue Output or Analogue Value, is an object whose properties represent the externally visible characteristics of an analogue output. An "analogue value" is a control system parameter residing in the memory of the BACnet Device.
The Binary Input object type defines a standardized object whose properties represent the externally visible characteristics of a binary input. A "binary input" is a physical device or hardware input that can be in only one of two distinct states. Those states are referred to as ACTIVE or INACTIVE. A typical use of a binary input is to indicate whether a particular piece of mechanical equipment is on or running, such as a fan or pump, is running or idle. The state ACTIVE corresponds to the situation when the equipment is running, and INACTIVE corresponds to the situation when the equipment is off or idle. In some applications, electronic circuits may reverse the relationship between the application-level logical ACTIVE and INACTIVE and the physical state of the underlying hardware. For example, a normally open relay contact may result in an ACTIVE state when the relay is energized. The Binary Input object provides for this possibility by including a Polarity property. Respectively, a Binary Output is a binary output, which is a physical device or hardware output that can be in only one of two distinct states, mentioned above. In case of an Binary Value, this is a control system parameter residing in the memory of the BACnet Device. This parameter may assume also only one of two distinct states, referred to as ACTIVE and INACTIVE.
The Device object type defines a standardized object whose properties represent the externally visible characteristics of a BACnet Device. There shell be exactly one Device object in each BACnet Device. A Device object is referenced by its Object_Identifier property, which is not only unique to the BACnet Device that maintains this object but is also unique throughout the BACnet internetwork.
Multi-state Input object type defines a standardized object whose Present_Value represents the result of an algorithmic process within the BACnet Device in which the object

resides. For example, the Present_Value of the state Multi-state Input object may be the result of a logical combination of multiple binary inputs or the threshold of one or more analogue inputs or the result of a mathematical computation. The Present_Value property is an integer number representing the state. The State_Text property associates a description with each state. The Multi-state Output type is an object whose properties represent the desired state of one or more physical outputs or processes within the BACnet Device in which the object resides. For example, a particular state may represent the active/inactive condition of several physical outputs or perhaps the value of an analogue output. The Present_Value property is an unsigned integer number representing the state. The State_Text property associates a description with each state. And a Multi-state Value is a control system parameter residing in the memory of the BACnet Device. The Present_Value property is an unsigned integer number representing the state. The State_Text property associates a description with each state. The Schedule object type is an object used to describe a periodic schedule that may recur during a range of dates, with optional exceptions on arbitrary dates. The Schedule object also serves as a biding between these scheduled times and the writing of specified "values" to specific objects at those times. Schedules are divided in days, of which there are of two types: normal days within a week and exception days. Every of object types mentioned above has a list of properties which describes the object. [1].

Alarm and event services

This clause describes the conceptual approach and application services used in BACnet to manage communication related to events. In general, "events" are changes of value of certain properties of certain objects, or internal status changes, that meet predetermined criteria. There are three mechanisms provided in BACnet for managing events: change of value reporting, intrinsic reporting, and algorithmic change reporting. Here, we provide a short description only of first mechanism, because it is used in Alarm Control task.

Change of value (COV) reporting allows a COV-client to subscribe with a COV-server, on a permanent or temporary basis, to receive reports of some changes of value of some referenced property based on fixed criteria. If a standard object provides COV reporting, then changes of value of specific properties of the object, in some cases based on programmable increments, trigger COV notifications to be sent to one or more subscriber clients. Typically, COV notifications are sent to supervisory programs in COV-clients devices or to operators or logging devices. [1]
A COV notification is sent to a BACnet client every time when current value of a particular object was changed by a value of a so called COV-increment.

OUT_OF_RANGE Algorithm

To detect when Present_Value of any object is in alarm the OUT_OF_RANGE algorithm is used (see figure 1).

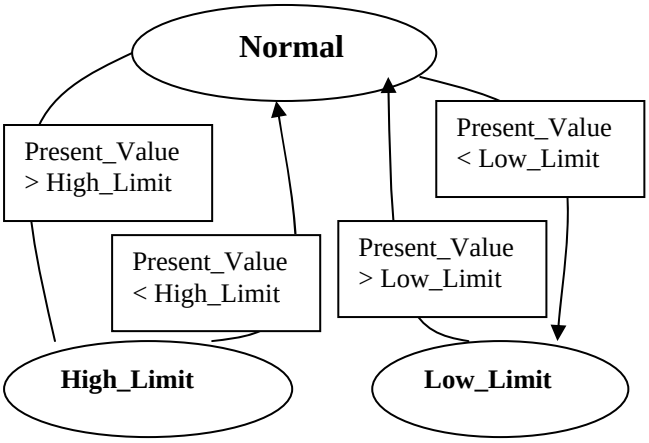

Figure 1. OUT_OF_RANGE Algorithm

This algorithm is used in case of all objects of any "Analogue" or "Binary" kind. OUT_OF_RANGE occurs if the referenced property leaves a range of values defined by the High_Limit and Low_Limit of the alarm data point and remains there for Time_Delay seconds. These two limits are specified doing the configuration of the Alarm data point. If the transition is to a value above the High_Limit or below the Low_Limit, a TO-OFFNORMAL transition is generated. In this case alarm beeper will start to beep.

The beeper will stop beep only in case when the User touched the screen. This will be an acknowledge to the raised Alarm event.

Object access services

This clause defines nine application services that collectively provide the means to access and manipulate the properties of BACnet objects. A BACnet object is any object whose properties are accessible through this protocol regardless of its particular function within the device in which it resides. We will describe only the services which are used by processes of the project.
ReadProperty Service – is used by a BACnet-client to request the value of the property of one BACnet Object. This service allows read access to any property of any object, whether a BACnet-defined object or not.
WriteProperty Service – is used by a BACnet-client to modify the value of a single specified property of a BACnet object. This service potentially allows write access to any object, whether a BACnet-defined object or not.

Remote device management services

The Who-Is service is used by sending BACnet-client to determine the device object identifier, the network address, or both, of other BACnet devices that share the same internetwork. The Who-Is service is an unconfirmed service. It means that the receiving device should not send an acknowledge (confirmation) to the sending device on receiving of this service. The Who-Is service may be used (1) to determine the device object identifier and network addresses of all devices on the network or (2) to determine the network address of the specific device whose device object identifier is known but whose address is not. [1] By address we understand the Media Access Address (MAC) of the device on the internetwork.

The statement of the problem

The main problem of this project was to implement the software, which shell run in an embedded system. It should also allow to support the BACnet data communication protocol. The touch-screen is a direct BACnet-user as a BACnet-Client. From this point of view, the software development was divided in a few steps. The services from the BACnet protocol side were already available so this is not the concern of this paper.
Every step had as a purpose the implementation of all abstract data structures which control separated parts of the functionality of the device. In first part were included methods needed for controlling the device at the lower hardware level, like processing screen touches, serial communication through the serial communication protocol, saving data in the flash memory, etc. As a next step was the development of methods which allow to extract users defined data points, in order to display them on-screen. All data points are implemented as good defined screen elements, as for example, static text or different kind of controls. Some of these controls have BACnet references. From other side, here were included all methods need for implementing interaction between the system and the user, by means of touching the screen. All these methods composing User Interface class. User Interface constructs and draws all current screen elements. Also it is necessary to say that all parts of the project which we will introduce step-by-step later are also organized as classes. These classes running in system as separated processes or tasks and controlled by the operating system mentioned above. Next part is concerned about updating all current screen graphical elements in order to have every time on-screen Present_Values of all these screen elements. These methods are included in Update Remote task. Update Remote reads from network and updates the currents screen data points. The next implemented part represents Alarming Control. This task checks data points, defined also by the user as alarms, to be inside of some defined by user boundaries. When the Present_Value of a data point is outside of these domains, an alarm event is generated. It will notify the user starting an Alarm Beeper task. The last parts represents the task which executes managing of all defined trend logs. This means data acquisition, filtering and drawing on screen of data for every trend once it is displayed on-

screen. One trend graphical view element manages in real-time up to four processes.

Concept of the touch-screen

The touch-screen is designed for handling 500 data points. A data point means one Property of an BACnet Object. The data points are organized in linked lists and are stored in the systems data memory flash. Later the contents of these pages will be read one bye one in systems RAM and put on-screen beginning with the default screen. The memory of the system is used dynamically, so the number of data points may differ in depends of how many pages(screens) user has described. In addition, user can describe alarm data points, data trend logs and schedules. One touch-screen can contain data points from more than one BACnet-Server. Here exists only one restriction, all devices must be connected to the same physical and logical network. [2]

The touch-screen is a simple BACnet-Client. It polls one or more BACnet Servers when it is active (the backlight of the screen is on) and the values are shown on-screen. In other words, as long as touch-screen in on, the Present_Value of all data points from current page (screen) are displayed. If there are trends configured (screen elements which allow to display in real-time the evolution of up to four processes per screen), the touch-screen is using COV mechanism, if it is available from the Server to update on-screen data. Otherwise it will poll in time-step of one second the Objects which have been set in the trends. The same mechanism of COV is used for Alarms monitoring in case if they were also configured. [2]

The hardware Configuration of the touch-screen is presented on figure 2. Used abbreviations standing for as following:

NV RAM – non volatile RAM;

RAM – is a SRAM with an 256Kx16 organisation;

FLASH- represents the program-memory 512K x16. Another Flash memory with the same organisation represents the Data Flash.

80C186 – AMD-Embedded processor 186EM;

RS232 Service – service port used for downloading to the device of the firmware, configuration files;

Ethernet 10 MBit – AMD AM79C961 chip;

LCD – LCD Display + Touch Screen of a resistive type;

Software Architecture

In the following we will introduce one by one the most important software components which were developed for this project. We will begin with the UserInterface task

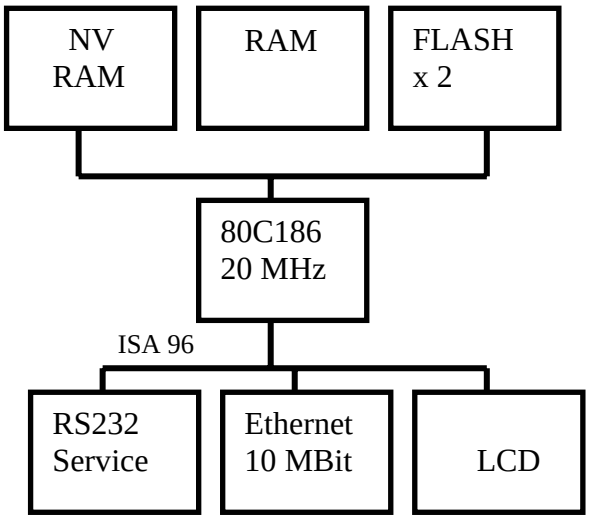

Figure 2. Hardware of the system

User Interface Task

The UserInterface is first task which is started and runs in system at device power-up. This task initializes and calibrates touch screen. At first run of the device the calibration constants are written in systems Data-Flash memory. This memory is divided in eight data banks as 64 Kbytes each. Because the access to the device is password protected, next is checked the user's Identification Number, up to five digits. Password protection is optional, if the user sets a 0 password, this means no password is used. Also, the user can change the password at every time. Every new choused password is saved in the flash memory. After this operation UserInterface task starts the rest of the tasks which are running in the system, like UpdateRemote, AlarmBeeper, UpdateAlarmList, and UpdateTraceList. Finally the configuration

of the first screen is read from Flash memory and is plotted on-screen. Inside of the endless loop part of the task, each thirty milliseconds UserInterface checks and processes screens touch buttons and controls touching. Once a button or a control is touched, and this screen element has a BACnet reference, a WritePropertyRequest is generated. According with this request a response from network server is generated, and the contents of the respected control is updated. Screen elements management like activation, deactivation, button press checking, and value updating is done by ScreenManager class methods. Each next screen configuration is read from Flash memory and is constructed in RAM as a list of screen elements. Configuration of each screen can be done by Hardware Configuration tool, implemented using Borland C++ Builder 4.0. This tool generates a binary file which is downloaded directly into the device Flash memory. Every configuration consists from a list of screens. Every screen has a link to the next one and to the previous screen, except the first which has no a link to the previous screen and the last which has no link to the next one. Every screen is built from a list of interconnected elements, exactly as it is done in case of the screens. This list will be read in the RAM each time when the current screen is changed by a press of ChangeScreen button. If the touch-screen is not activated, by a press of a button or a control for a period of time, then screen goes in "sleep" mode to assure power sawing. By default this interval of time is set to 60 seconds.

Update Remote Task

Next started task is UpdateRemote. The main goal of this task is continuous updating the contents of each current screen elements when the touch-screen is on. This is done every 25 milliseconds when a ReadPropertyRequest is generating in case of devices which are not supporting the COV mechanism. If was received an errorless response to this Request, then notifyValue method of UserInterface task updates current screen element. Periodically are updated only screen elements with BACnet reference. To reduce the network traffic, once the screen goes in "sleep" mode, this task is suspended until the touch-screen is not touched again. Also, UpdateScreenObjects method of UpdateRemote class updates each screen elements to unknown state before touch-screen goes to sleep. This is done because the state of each element starting from this moment of time is unknown. When the screen is touched the LCD wakes-up, and the task is resumed. Then via periodically generated ReadPropertyRequests elements from current screen are updated with new values, which were read from the local network. To assure a good work of the device, on power-up, this task generates Who-Is requests, as was mentioned above, to determine the device object identifier, the network address, or both, of other BACnet devices that share the same internetwork.

Update Alarm List Task

This is a complex task which provides alarming notifications when user specified data points go out of selected range. The Alarm notification is generated according to the OUT_OF_RANGE algorithm which was mentioned above. There are two kind of alarm data points. In case of "Analogue Values", like for example, outside temperature, an alarm point has two limits the Lower Limit (LL) and the High Limit (HL). The second type of alarm points are "Digital Values" or binary values with two states On and Off, for example a switch. For this kind of alarms the "Activation Condition" (AC) is defined for each alarm. In this case LL = HL = AC In addition, each alarm point has a BACnet Object Identifier of the data point on BACnet to which it belongs and a small description of the alarm. Configuration of each alarm data point can be done by user via the same Hardware Configuration tool. All selected alarm items are organized as a list which is stored also in Flash memory. On power-up of the device this data are read from Flash memory and are constructed as a sorted Alarm Data Points List in systems RAM memory. This task never gets suspended or stopped. Checking of each Alarm data point is done through Change of value (COV) reporting. For this purpose the task provides a covHandler method. To assure COV reporting

continuously the task generates periodically a SubscribeCOVRequest at the end of a 300 second interval. If the device does not support this kind of subscription, then Polling mechanism is used. Should be mentioned that updating of Alarm List is implemented as a state machine with four states. This state machine allows the correct updating of all data points from the Alarm List. It uses Data Polling as well as the COV mechanism to acquire new values for Alarm Data points. On the same state machine algorithm is based data acquisition for Data Trends which were mentioned above.

4.3.1. State Machine

In the following will be shortly described the state machine used for updating data points. See the figure 3.
At the power-up every data point is in *s_known* state. In this state a "Subscribe COV Request" and "Read Property Request" are generated. If SubscribeCOVRequest succeeded than a "Subscribe COV_Response" is generated and the state of the data point is changed to *s_coved*. The data point will stay in this state until Resubscribe Interval expires when a new "Subscribe COV Request" will be generated and so on. If "Subscribe COV Request" fails than it means that device does not support COV Reporting and Polling Data will be used instead. In this case after "Read Property Request" the state of the data point will be changed to *s_covpending* and after a Resubscribe interval in order to become "coved" a "Subscribe COV Request" will be sent and state of the data point will be changed to *s_poll*.
This means that is time for a new Poll cycle and a new data sample will be acquired. This scenario is endless used as long as the touch-screen is on-line and allows a correct updating of each data point. In addition, it allows to reduce network traffic, because in case of Polling Data every new sample is sampled when the period of time defined in state machine algorithm expires.
Correct timing is done by using timer-counters, provided by the operating system, which reside inside of every Alarm List element. The same procedure of updating is also used for each data point from inside the trends list.
When a new value is sampled its Present_Value is checked if it is inside of allowed limits, respectively if it is not equal to activation condition for binary values. If Present_Value is outside of the allowed limits the Alarm is raised. In this case the current screen is changed to the one were is placed the data point currently in alarm, and the Alarm Beeper is started. Also, the time, date and the description of the alarm is registered in a special list for keeping the alarm history. When is needed the user can visualize

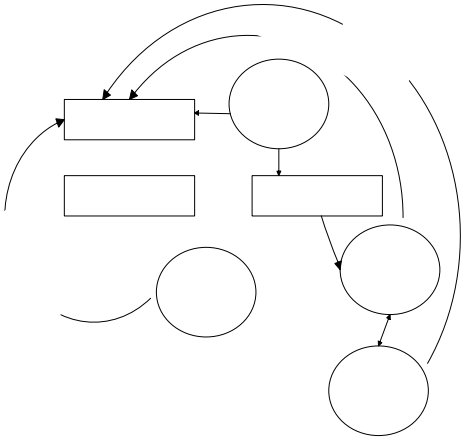

Figure 3. State Machine

this history. The beeper will beep until the user will not touch the screen which represent in this case that alarm was acknowledged. Also, the beeper is started even in case when the touch-screen is not activated (backlight is off).

Buttons and graphical elements

To draw on touch-screen all our graphical elements we used the C functions provided by GUI-Library [3]. The Simplify Technologies GUI-Library is a collection of programs and functions that can be used on drawing graphical elements on LCD's and Touch displays.
All graphical elements of the projects are implemented according to one pattern, ScreenRef, which is an abstract class. This class

provides methods for activating, deactivating, cleaning up, processing presses of the screen element, and notification in case when the current contents of screen element is changed. The element is updated from network. Screen elements can be divided in two groups: those which requires press from the user, and without press. Elements from first group and from the second as well can be divided further in those with BACnet reference and without. Controls without touch usually are different kind of indicators, like show the inside room temperature, the state of ventilation on or of and so on. As an example of a control which require a press is a button for switching on or of the light, etc. Also as graphical elements can be used bitmaps as backgrounds for each screen. As is well known that bitmap images are quite large, and require a large space to be stored, a compression/decompression algorithm is used. This algorithm is the same one as algorithm used in ZIP files. The type of bitmap images is black/white with a 640x480 pixels resolution.
Next screen element, DataTrace, displays in real-time the evolution of room temperature, for example. For this purpose is used real-time data acquisition. To achieve a good accuracy two buffers are used . A small one for over sampling which is about 10 samples, and a large one, about 240 samples. When the over sampling buffer is full, an average value of all samples is calculated and the result is stored in second buffer. Then this buffer is displayed on-screen. The screen can be scrolled horizontally, so it shows all the time only last screen-width number of samples.

Conclusions and future development

This project was successfully implemented at Amann GmbH and for one year runs in the touch-screen mentioned above.
In time it was enlarged because new futures were required.

One of this future, which is currently under development is to support Schedule Object type. One of the main problem of the project was the implementation of the decoding and then encoding of the data which coming from the network. In last case of the schedule this is quite a large amount of data, which is required from the network in real-time at the moment when the User begins to work with the schedule. Because the User can modify data after this it should be encoded and sent back to the network.
Also soon should be added the support for the IP stack. It means a possibility to use the IP transport protocol.

Acknowledgments

This paper is a part of EWMS View Touch Screen project, which was developed at AMMAN GmbH, Oberhaching. We would like to thank to Mr. Claus Färber, the leader of the Development Department, for his help and support on working at the project.

References

[1] ***, (2001), *BACnet A Data Communication Protocol for Building Automation and Control Networks*, ISSN 1041-2336, American Society of Heating, Refrigerating and Air-Conditioning Engineers, INC, pp. 1, 130, 135, 140, 148, 153, 159, 169, 203, 207, 211, 216, 224, 265, 296
[2] ***, (2002), *BACnet EWMS-View Technical Description*, AMANN GmbH, pp. 3-5, 7
[3] ***, (2001), *Simplify Technologies GUI-Bibliotek*, Programierung von Benutzeroberflächen für LDC- und Touch-Displays in ANSI-C, Simplify Technologies GmbH
[4] Ralph Moore, (1995), Simple Multitasking Executive, USER'S GUIDE, Micro Digital, Inc.