Journal of Engineering and Applied Sciences 2 (11): 1598-1606, 2007
ISSN: 1816-949X

PC-Based Centralized System for Controlling and Determining Status of Electrical Devices

Mohammad Rohul Amin
Department of Computer Science and Engineering,
Daffodil International University, Dhaka, Bangladesh

Abstract: At present a world that people prefer automated systems rather than manual systems because of efficient, reliable and timesaving. Since the automated centralized electrical system is very expensive, most of the offices, hotels and industries are still controlling their decentralized electrical systems manually. In case of large organization, these manual decentralized electrical systems are very time consuming for controlling and furthermore the administrator of existing system could not identify the present status (On/Off) of the electrical devices. The proposed system consists of an electronic circuit which interfaces with a computer through the parallel port for timely controlling the electrical devices centrally using the database software. The proposed database software records the transactions about each and every devices, can estimate the total used time for a specific device. It includes internal memory for PC-free operation; control over 4,000 devices at a time; feed-through outlet won't block from plugging in other devices; and provides easy trouble-free connection. This PC-based centralized system is very effective for the home, offices, hotels and industries in terms of cost and time for controlling the electrical devices.

Key words: Electrical circuit, interfacing, parallel port, IEEE 1284 Standard, DLL, VB, oracle

INTRODUCTION

Now a day all the manual systems are converted into the automated systems with the help of computers (Rafael, 2002; Wurtz *et al*., 1996; Anzioso *et al*., 2002) for smooth services. Basically if the electrical devices are controlling manually, a lot of problems which are referring like user needs to go at the field levels to control the electrical devices which take a lot of time; difficult for the administrator to know the present status (On/Off) of all the electrical devices; if the system depends on the employees, case of irresponsibility the devices can be miss-used and difficult to calculate the utility time for a specific electrical device.

The proposed system interfacing with the computer through parallel port for timely controlling lights, fans, air-coolers, motors, alarms and other electrical devices centrally using the database software (Microsoft Visual Basic (Wayne, 2005) at front-end and Oracle (Scott, 2005) at back-end). It has a strong central database which records the transactions (on-time, off-time, used-time, used date) about each and every electrical device. Hence, the system can estimate the total used-time for a specific device and automatically On/Off the devices based on the recorded on-time or off-time. Therefore, the system could save the electrical energy and ensure cost-effective development of the industry.

REQUIREMENT ANALYSIS

Parallel port (Dhananjay, 1998) sends 8 bits and receives 5 bits at a time. The proposed application used the parallel port to control the electrical devices. Parallel port is a simple and inexpensive tool for building computer controlled devices and projects. The simplicity and ease of programming makes parallel port popular in electronics hobbyist world. The parallel port is often used in computer controlled robots PIC programmers, home automation, etc.The IEEE 1284 Standard (Archived, 1994; Susan and Maj, 2006) which has been published in 1994 defines five modes of data transfer for parallel port. They are,

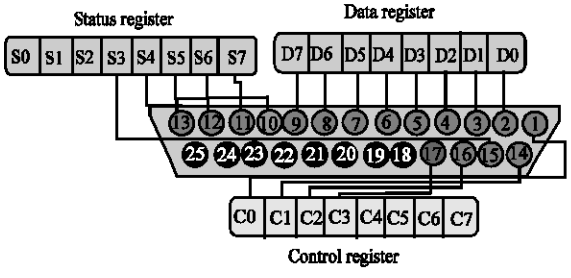


Fig.1: Parallel port pin status

- Compatibility Mode, Nibble Mode, Byte Mode
- EPP
- ECP

The pin outs of parallel port connector (Fig. 1) are shown below.

The lines in parallel port connector (Dhananjay, 1998) are divided into the three groups. They are data lines (data bus), control lines and status lines.

As the name refers, data is transferred over data lines, control lines are used to control the peripheral and of course, the peripheral returns status signals back computer through status lines. These lines are connected to data, control and status registers internally. The details of parallel port signal lines are shown in Table 1.

The data, control and status lines are connected to there corresponding registers inside the computer. So by manipulating these registers in program, one can easily read or write to parallel port with programming languages like VB (Jan, 1997).

The registers found in standard parallel port are data, status and control registers. As there names specify, data register is connected to data lines, control register is connected to control lines and status register is connected to status lines. For example, if write '1' to Data register, the line Data 0 will be driven to +5v. Just like this, it can programmatically turn on and off any of the data lines and control lines. The register addresses of LPT1 and LPT2 is shown in Table 2.

Table 1: Parallel port signal

pin no (DB25)	Signa name	Direction	Registerl -bit	Inverted
1	nStrobe	Out	Control-O	Yes
2	Data0	In/out	Data-O	No
3	Date1	In/out	Data-1	No
4	Data2	In/out	Data-2	No
5	Data3	In/out	Data-3	No
6	Data4	In/out	Data-4	No
7	Data5	In/out	Data-5	No
8	Data6	In/out	Data-3	No
9	Data7	In/out	Data-7	No
10	Nack	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper out	In	Status-5	No
13	Select	In	Status-4	No
14	Linefeed	Out	Control-1	Yes
15	nError	In	Status-3	No
16	nInitialize	Out	Control-2	No
17	nSelect-Printer	Out	Control-3	Yes
18-25	Ground	-	-	-

Table 2: Register addresses of LPT1 and LPT2

Register	LPT1	LPT2
Data register (baseaddress + 0)	0×378	0×278
Status register (baseaddress + 1)	0×379	0×279
Control register (baseaddress + 2)	0×37a	0×27a

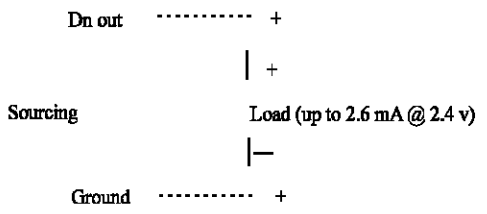


Fig. 2: Simple circuit

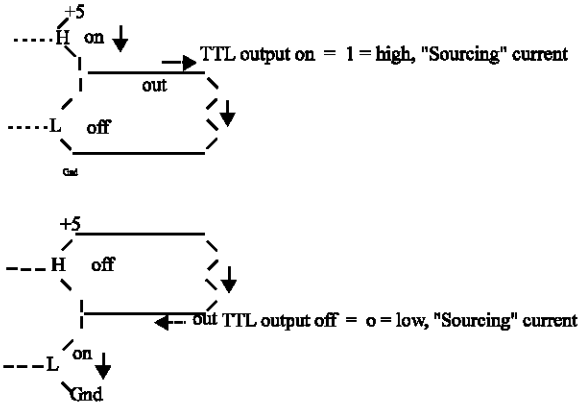


Fig. 3: Regular TTL circuit

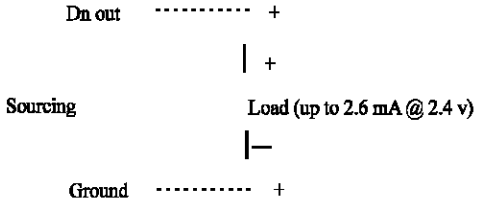


Fig. 4: Simple current sinking load connection

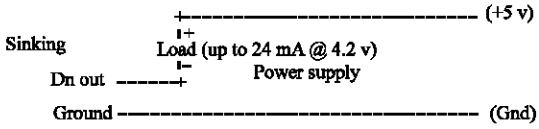


Fig. 5: Simple current sinking 5v connection

Those data pins are TTL level output pins. This means that they put out ideally 0V when they are in low logic level 0 and +5V when they are in high logic level 1. In real world the voltages can be something different from ideal when the circuit is loaded. The output current capacity of the parallel port is limited to only few milliamperes. A simple idea Fig. 2 shows how to connect load to a PC parallel port data pins (Dhananjay, 1998).

This is not the only way to connect things to a parallel port. The parallel port data pins are TTL outputs, that can both sink and source current. In ordinary parallel

port implementations the data outputs are 74LS374 IC totem-pole TTL outputs which can source 2.6 mA and sink 24 mA.

Regular TTL (Fig. 3) outputs basically consist of a two stacked transistor in series between +5 volts and ground, with the output coming from the connection between them. This is called a totem pole output. At any given time one of these transistors is conducting and the other is not. To pull the output high, the transistor from +5 to the output conducts (H), which sources positive current from the output to ground (that is, an external device between the output and ground will get power). To pull the output low, only the Lower transistor (L) conducts, sinking current to ground; an external device between +5 volts and the output can be energized.

The outputs are designed so that they give at least 2.4V at 2.6 mA load. This 2.6 mA figure is for ordinary LS-TLL circuits used, the LSI implementations used in many computers can give more or less.

Simple current sinking load connection (Fig. 4) is shown below.

When taking current from PC parallel port, keep the load low, only up to few milliamperes. Trying to toke too much current (for example shorting pins to ground) can fry the parallel port.

If an external +5 volt supply, then another option for connection: use the Data Out pins to sink up to 24 mA from +5 volt supply. This can be made with a circuit (Fig. 5) is shown below.

The idea of interface shown above can be expanded to control some external electronics by simply adding a buffer circuit to the parallel port (Dhananjay, 1998).

The following circuit (Fig. 6) is the simple interface can use to control relay from parallel port.

The circuit can handle relays which take currents up to 100 mA and operate at 24V or less. The circuit needs external power supply which has the output voltage which is right for controlling the relay.

One of the simples optoisolated output circuit for parallel port is the 4N33 (Fig. 7) based circuit.

The 4N33 optocouplet device has a Darlington output transistor that is capable of driving up to 30 mA of load safely. The maximum voltage on the output side is 30V. The input to output isolation can handle up to 1500V voltage. To connect the input side (+) to the parallel port, output pin can be used for the controlling. Then to connect the input (-) side to parallel port ground pin. The output side is connected to the circuit to be controlled at right polarity. This example circuit used 1 kohm resistor to limit the control current (circuit should also work well with 470 ohm resistor). Because the current fed to the optocoupler is very low (just few mA).

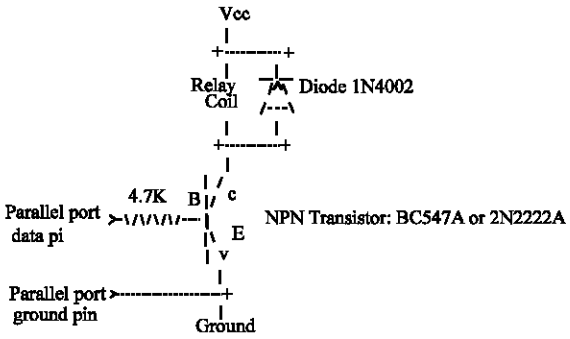


Fig. 6: Simple interface control relay from parallel port

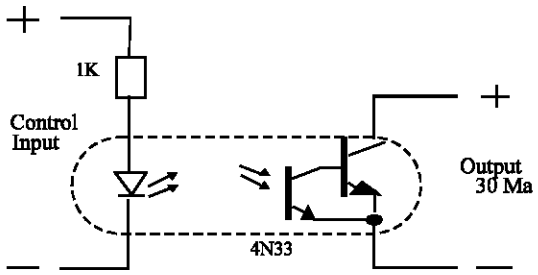


Fig. 7: Basic circuit with optoisolation

If a very good protection of the parallel port and more drive capacity consider optoisolation (Fig. 8) using the following type of circuit.

It is possible to control mains voltage through parallel port with an above circuit. When controlling mains voltage, need to be very carefully and know what to do it safely. Mains voltage can kill if get in touch with it and bad mains controlling circuit can burn down the house.

First idea for controlling mains power is to use one of the circuits above to control a relay that then controls the mains power. This suits for many applications as long as the relay is rated for the mains power switching applications and for the current rating of applications. The relay contact is used to switch the phase/live wire going to the equipment. The fuse here is used to protect the relay against overload. A relay will work on applications where the device is turned on and off quire rarely.

The outstanding feature of Inpout 32.dll (DDL, 2007) is it can work with all the windows versions without any modification in user code or the DLL itself. The Dll will check the operating system version when functions are called and if the operating system is WIN9X, the DLL will use _inp() and _outp functions for reading/writing the parallel port. On the other hand, if the operating system is WIN NT, 2000 or XP, it will install a kernel mode driver and talk to parallel port through that driver. The user code

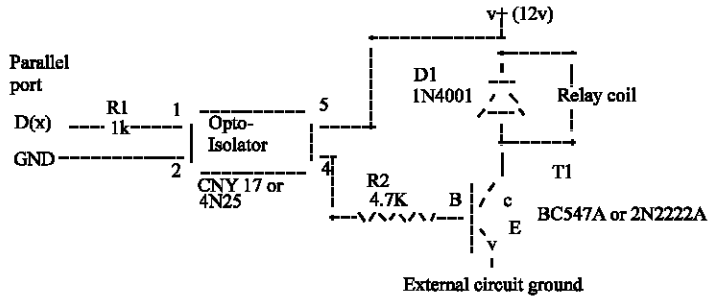


Fig. 8: Capacity consider optoisolation

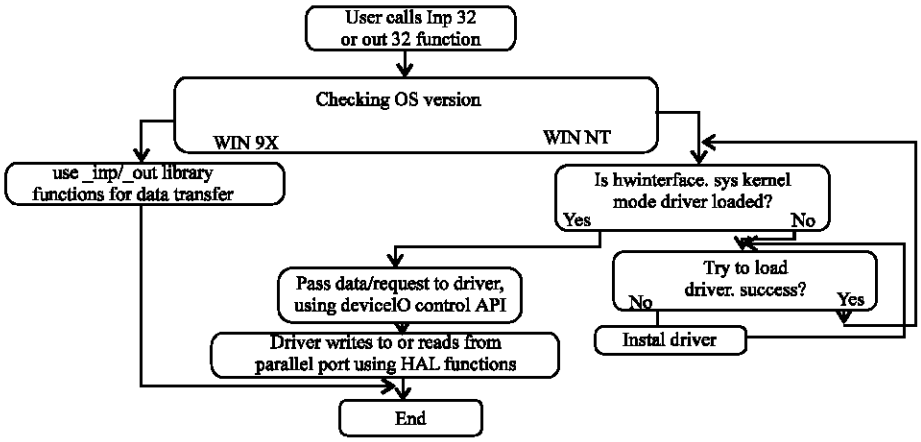


Fig. 9: Flow chart of DLL

will not be aware of the OS version on which it is running. This DLL can be used in WIN NT clone operating systems as if it is WIN 9X. The flow chart (Fig. 9) of the program is given below.

The two important building blocks of this program are a kernel mode device driver embedded in the DLL in binary form and the DLL itself.

The source code of Hwinterface.sys kernel mode driver is located in "kernel_mode_driver_source" directory. Where the hwinterfacedrv.c is the main application source file.

Three functions implemented in the driver are:

- 'DriverEntry' , Called when driver is loaded. Creates device object and symbolic links.
- 'hwinterfaceUnload', Called when driver is unloaded, performs clean up.
- 'hwinterfaceDeviceControl', handles calls made through DeviceIOControl API. Performs reading writing to the parallel port according to the control code passed.

The functions in the DLL are implemented in two source files, inpout32drv.cpp and osversion.cpp. osversion. cpp checks the version of operating system. inpout32drv.cpp does installing the kernel mode driver, loading it, writing/ reading parallel port etc.

The two functions exported from inpout32.dll are 'Inp32', reads data from and 'Out32', writes data to specified parallel port register.

The other functions implemented in Inpout32.dll are:

- 'DllMain', called when dll is loaded or unloaded. When the dll is loaded , it checks the OS version and loads hwinterface.sys if needed.
- 'Closedriver', close the opened driver handle. called before unloading the driver.
- 'Opendriver', open a handle to hwinterface driver.
- 'Inst' , Extract 'hwinterface.sys' from binary resource to 'systemroot\drivers' directory and creates a service. This function is called when 'Opendriver' function fails to open a valid handle to 'hwinterface' service.
- 'Start' , starts the hwinterface service using Service Control Manager APIs.
- 'SystemVersion' Checks the OS version and returns appropriate code.

PROPOSED SYSTEM DESIGN AND IMPLEMENTATION

E-R diagram (Fig. 10) of the proposed system shows the relationship between the entities.

The tabular representation (Fig. 11) of the proposed system contains the following fields:

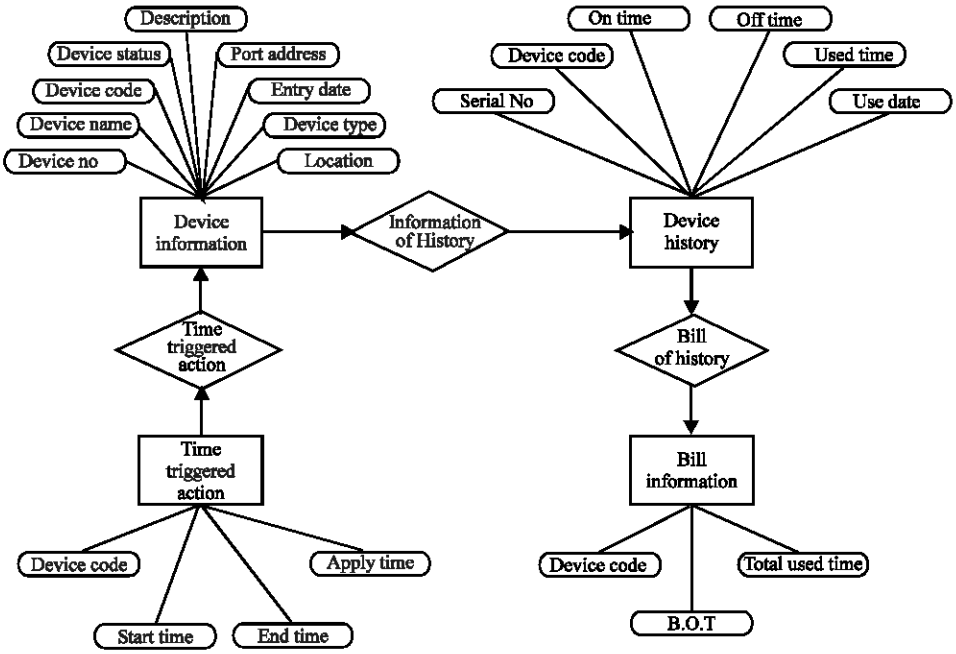


Fig. 10: E-R diagram of the proposed system

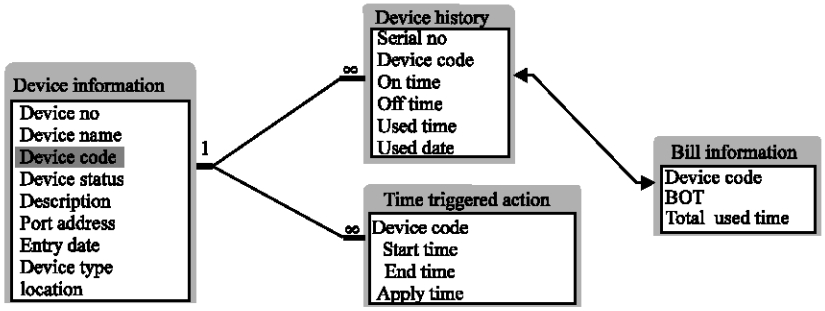


Fig. 11: Tabular representation of the proposed system

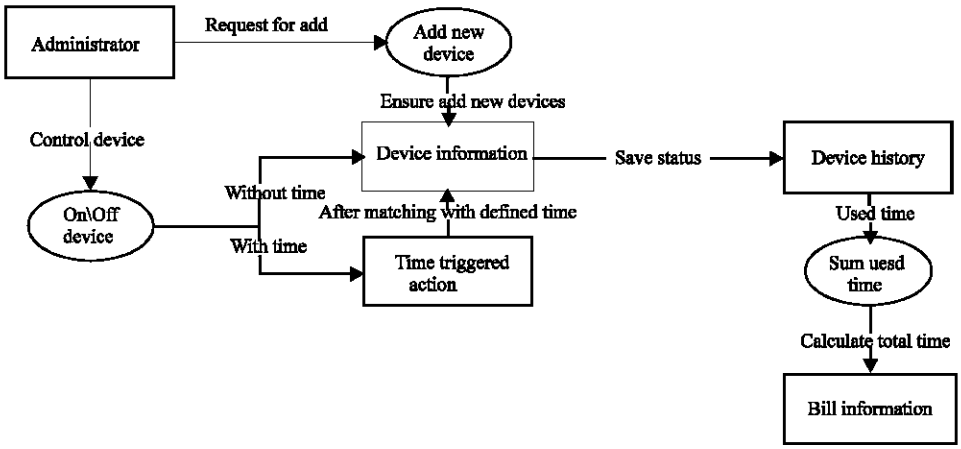


Fig. 12: DFD of the proposed system

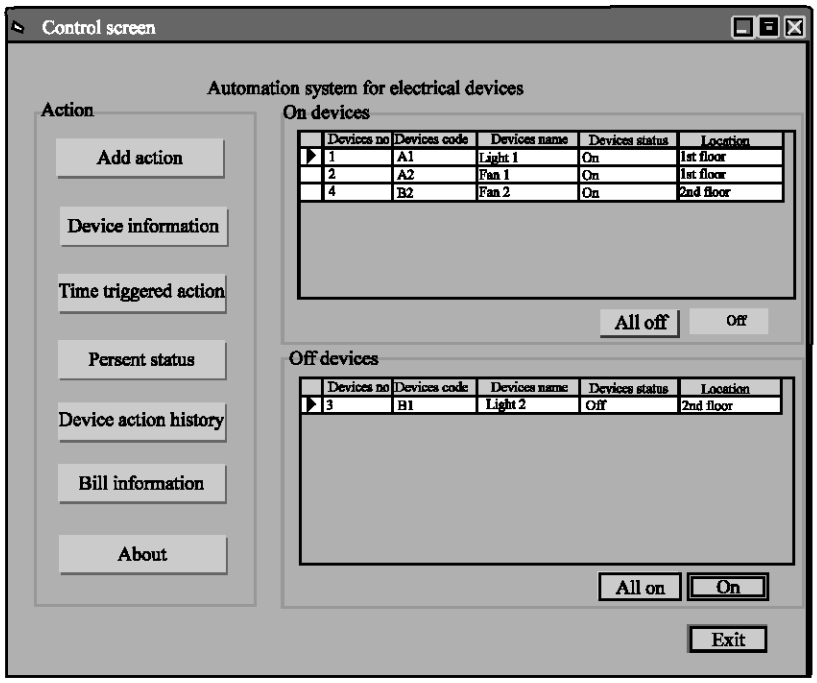


Fig. 13: Main control window

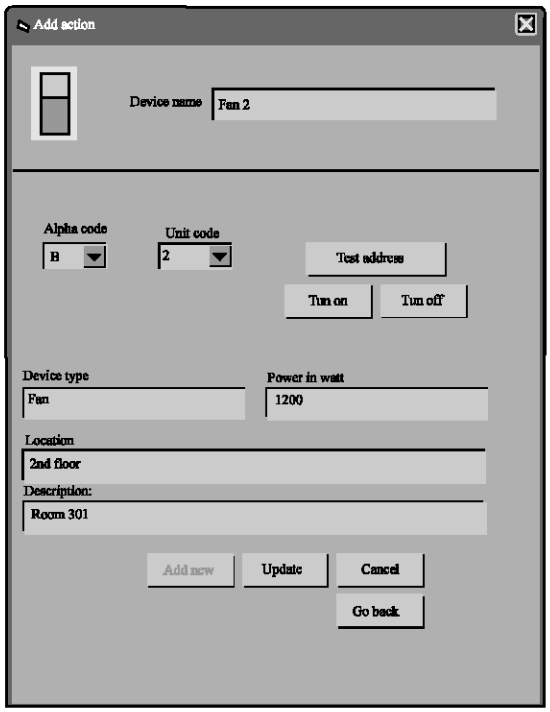


Fig. 14: Add new device window

- Device No→Distinct numbers of electrical devices.
- Device Name →Names of electrical devices.
- Device Code→Distinct addresses of electrical devices.
- Device Status→ Present status (On/Off) of electrical devices.
- Description→Special notes of electrical devices.
- Port Address →Addresses of USB ports are allocated for electrical devices.
- Entry Date → Dates when electrical devices are added to the system.
- Device Type →Categories of electrical devices.
- Location→Positions of electrical devices.
- Start Time→On times of electrical devices.
- End Time→Off times of electrical devices.
- Apply Time→Update times of electrical devices.
- Used Time→Time durations when the electrical devices are used.
- Use Date→Dates when the electrical devices are used
- Total Used Time → Total time used by the each electrical device.
- BOT→ Approximate consumes units of current.

Data Flow Diagram (DFD) (Fig. 12) shows the flow or direction of data.

From main control window (Fig. 13) of the proposed system, administrator can easily add new electrical devices, time triggered action apply, know the present status of each and every device and total devices action history.

In the add device window (Fig. 14) of the proposed system, administrator can easily add the new electrical device using device name, aloha code and unit code, device type, location and description of the electrical device. Administrator can check the addresses and status of the given electrical devices.

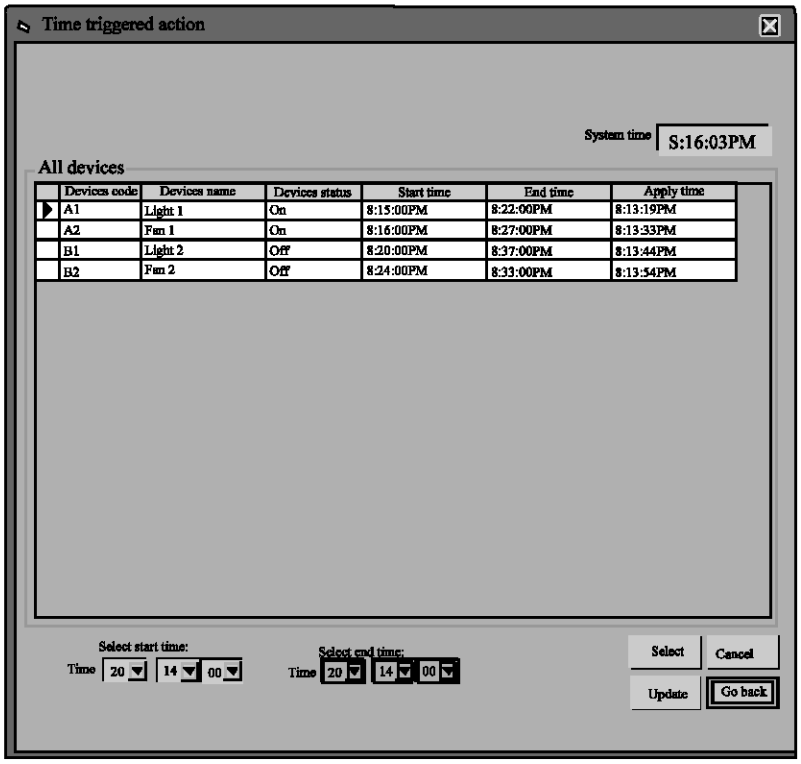


Fig. 15: Time triggered window

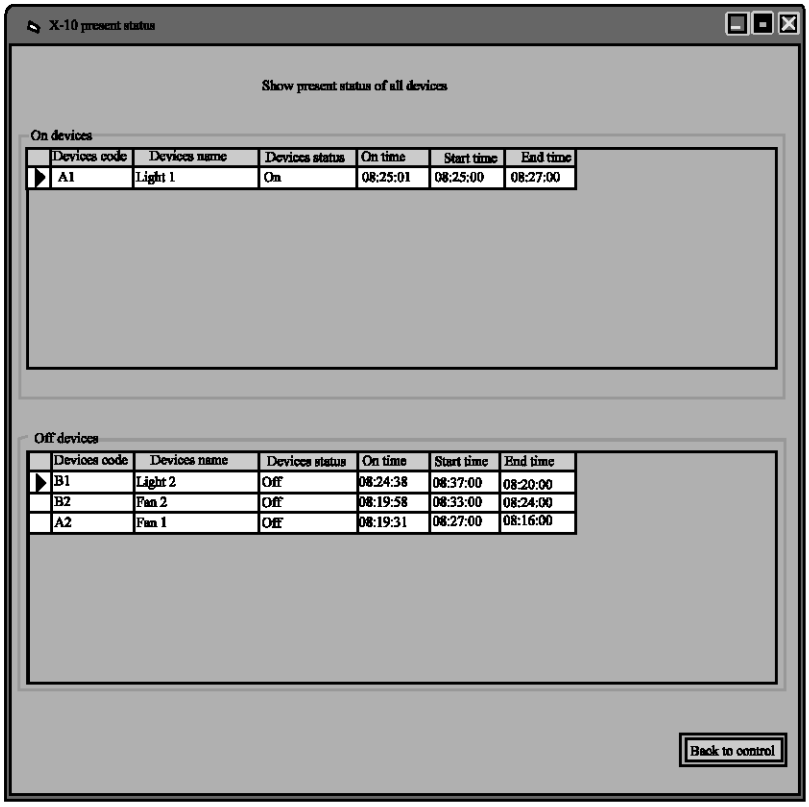
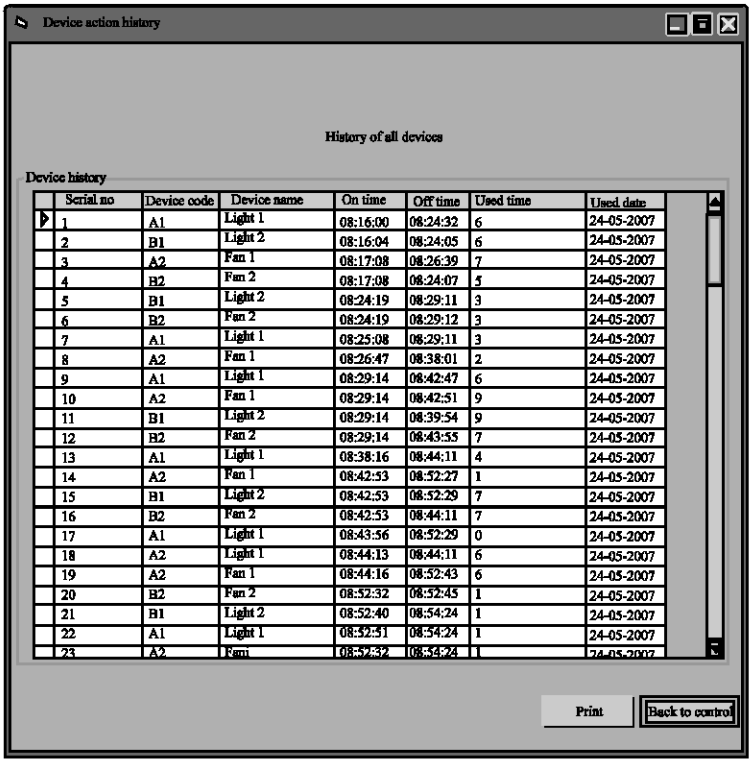
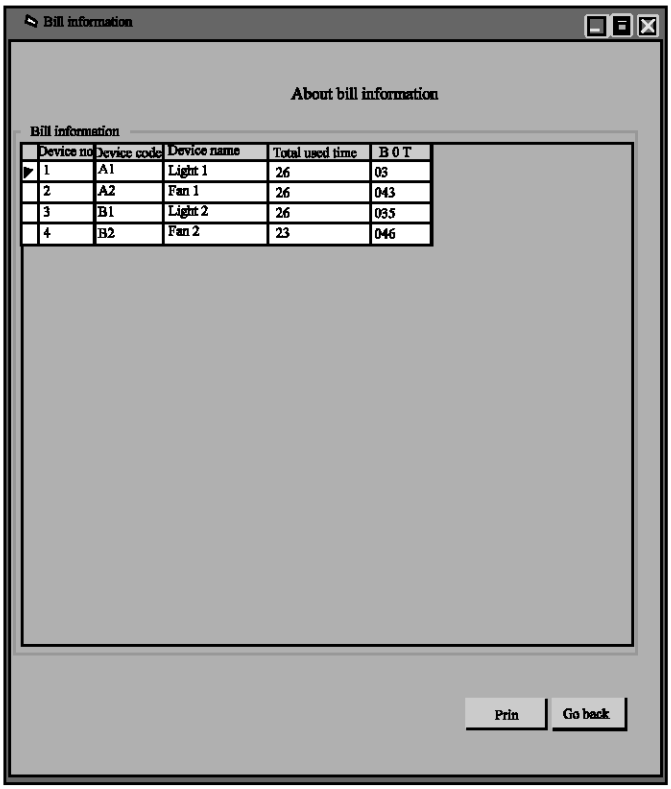


Fig. 16: Present status window



Serial no	Device code	Device name	On time	Off time	Used time	Used date
1	A1	Light 1	08:16:00	08:24:32	6	24-05-2007
2	B1	Light 2	08:16:04	08:24:05	6	24-05-2007
3	A2	Fan 1	08:17:08	08:26:39	7	24-05-2007
4	B2	Fan 2	08:17:08	08:24:07	5	24-05-2007
5	B1	Light 2	08:24:19	08:29:11	3	24-05-2007
6	B2	Fan 2	08:24:19	08:29:12	3	24-05-2007
7	A1	Light 1	08:25:08	08:29:11	3	24-05-2007
8	A2	Fan 1	08:26:47	08:38:01	2	24-05-2007
9	A1	Light 1	08:29:14	08:42:47	6	24-05-2007
10	A2	Fan 1	08:29:14	08:42:51	9	24-05-2007
11	B1	Light 2	08:29:14	08:39:54	9	24-05-2007
12	B2	Fan 2	08:29:14	08:43:55	7	24-05-2007
13	A1	Light 1	08:38:16	08:44:11	4	24-05-2007
14	A2	Fan 1	08:42:53	08:52:27	1	24-05-2007
15	B1	Light 2	08:42:53	08:52:29	7	24-05-2007
16	B2	Fan 2	08:42:53	08:44:11	7	24-05-2007
17	A1	Light 1	08:43:56	08:52:29	0	24-05-2007
18	A2	Light 1	08:44:13	08:44:11	6	24-05-2007
19	A2	Fan 1	08:44:16	08:52:43	6	24-05-2007
20	B2	Fan 2	08:52:32	08:52:45	1	24-05-2007
21	B1	Light 2	08:52:40	08:54:24	1	24-05-2007
22	A1	Light 1	08:52:51	08:54:24	1	24-05-2007
23	A2	Fan	08:52:32	08:54:24	1	24-05-2007

Fig.17: Device action history window



Device no	Device code	Device name	Total used time	B 0 T
1	A1	Light 1	26	03
2	A2	Fan 1	26	043
3	B1	Light 2	26	035
4	B2	Fan 2	23	046

Fig. 18: Bill information window

With the help of time triggered window (Fig. 15) of the system, administrator can timely control each and every electrical device with the start time and end time.

Present status window (Fig. 16), device action history window (Fig. 17) and bill information window (Fig. 18) show the different types of reports.

PERFORMANCE ANALYSIS

In order to overcome the limitations of manual system for controlling the electrical devices in terms of different aspects shown below, the proposed system requires a PC along with the database software and a simple logical electronic circuit.

- With the existing and proven technologies (e.g. lights, fans, etc.) for the large organizations, the manual system needs more man powers; whereas the proposed system needs a PC used as a server along with an administrator.
- For the large organizations, the users of manual system need to go at the field levels to control the electrical devices which take a lot of time; while an administrator of proposed system can centrally control over 4,000 electrical devices within a few seconds.
- For the manual system it is very difficult for an administrator to know the present status (On/Off) of all the electrical devices; whereas the proposed system provides a user-friendly monitoring system to identify the status and schedule control for the electrical devices.
- Manual system depends on the users and hence the case of irresponsibility the electrical devices can be miss-used and difficult to calculate the utility time for a specific electrical device; but the proposed system records the transactions (on-time, off-time and used-time) about each and every electrical device for saving electrical energy.
- The proposed system is very effective for the home, offices, hotels and industries in terms of low cost and high benefits for controlling the electrical devices.

CONCLUSION

The proposed system is designed as the automation for controlling and determining the status of electrical devices competently in terms of cost and time. The primary goal is to design an electronic circuit which is interfacing with the computer through the Parallel Port and to develop the database software (Microsoft Visual Basic 6.0 for front-end and Oracle 10g for back-end) for timely controlling the electrical devices centrally. The proposed system can integrate the whole electrical devices for home, office, hotel, or industries to schedule control for saving electrical energy and maintaining the electrical devices efficiently because it records the transactions (on-time, off-time and used-time) about each and every electrical device.

In future, the proposed system can be extended with the help of SMS (Sort Massage Service), Internet and sensor devices to control the electrical devices.

REFERENCES

Anzioso, F., A. Canova, P. J. Leonard, M. Ottella and D. Rodger, 2002. FE Analysis and Design of Electrical Devices for Automotive Applications. Int. Conf. Power Elec. Machines and Drives, 487: 159-164.

Archived, 1994. IEEE 1284 Standard: Standard Signaling Method for a Bi-Directional Parallel Peripheral Interface for Personal Computers. IEEE Computer Society.

Axelson, J.L., 1997. Parallel Port Complete: Programming, Interfacing and using the PC's Parallel Printer Port, Lakeview Research

Gadre, D., 1998. Programming the Parallel Port, Elsevier.

DLL, 2007, Inpout32.dll for WIN NT/2000/XP", http://www.logix4u.net/inpout32.htm.

Palacios, R., 2002. Remote Automatic Doorman via the Internet. Commun. ACM., 45: 23-25.

Urman, S., 2005. Oracle Database 10g PL/SQL Programming, Tata McGraw-Hill.

Susan, K. (Kathy) Land and Maj. Bret Wilson, 2006. Using IEEE Standards to Support America's Army Gaming Development, Computer, pp: 105-107.

Wayne Freeze, 2005. Visual Basic 6 Database Programming Bible, Wiley Dreamtech.

Wurtz, F., J. Bigeon and C. Poirson, 1996. A Methodology and a Tool for the Computer Aided Design with Constraints of Electrical Devices. IEEE. Trans. Magnetics, 32: 1429-1432.