SYSTEM DESIGN REPORT

WARDEN

Troy

1. INTRODUCTION

1.1 Goals and Objectives

WARDEN is a card based wireless door control system for companies, laboratories, hospitals especially areas where limited people are allowed to enter.

The users will be given cards that are installed to the system. The user will be able to enter a door if his card is given the entrance permission. Allowed users will be able to enter whereas the rest will not. System will give an alarm if door is opened for a long time or tried to be opened by an illegal way.

Assigning permissions will be done by the admin of the system using a master computer. Tracing the events going on the doors will also be done via the master computer. This will be provided by keeping the logs of the events of the door on the master computer.

Adding/removing/editing users or doors will also be provided to the system admin.

1.2 Statement of Scope

WARDEN is a door security system. Users will have cards and they will get them read to the card readers on the doors for entrance. The permission of the user will be checked and the entrance will be granted or denied.

There will be doors and users that are added to the system. Users and doors can be added to, deleted from the system or edited via the master computer user interface by the system admin. The user information and permission will be kept in the master computer. A user may have different permissions for different doors. There will also be user grouping. The members of the group will automatically get the group permissions. To reduce the waiting time of the users, the permissions of the users will also be stored locally at the doors; that is every door will store its own permission file. The update of permissions will be done from the master computer and the updated files will be sent to the doors. The master will be able to make an update whenever there is such a request from the user admin.

Another important task of the system is keeping the logs of the events on the doors like 'confirmed accesses, 'not defined user' and 'force attempt'. When a user gets its card read, the card id is sent to the door. It is checked and if the user has permission to enter, it will be reported to the master as a ‘confirmed access’. If the card id is invalid, then the report will be ‘user not defined’. If the door is open by brutal force; that is if the door opens without id authentication, this will be reported as a ‘forced attempt’. There will be a handle on one side of the door in case of an emergency. When this handle is used, it will also be reported as a ‘forced attempt’. The logs will be reported to the master immediately after an event. In case of a system crash, certain amount of these logs will also be stored locally at the doors.

The master will also provide a holiday and an emergency mode. When the holiday mode is enabled, the system will be shut down. The doors will not be opened even if an identified user gets his card read until this mode is disabled. When the emergency mode is enabled, the system will permit everyone. The doors will be open until this mode is disabled.

When the card is read to the card reader on the door, the card id will be checked from the local database. The card reader will send the id in Wiegand protocol, whereas the local board will get this data in RS232 protocol. Another task of our system is to establish the communication between the card reader and the board, which is a converter between Wiegand and RS232 protocols.

The communication between the doors and the master will be established through a wireless network. The risk of a crash in the communication due to external physical factors will be reduced. This will also make introducing new doors to the system very easier.

1.3 Design Constraints

The response speed of the system is very important. After getting read his card, the user shouldn’t wait for a long period of time. The authentication process should be as fast as possible.

The data transfer between the master computer and the door systems should also be fast. The amount of data traveling should be as small as possible. Also an encryption algorithm should be applied as we are using wireless network. Although the wireless protocols encrypt the data before sending, any wireless receiver using the same protocol can decrypt the data.

The system that will be established at the doors will have a totally 128MB disk. The operating system and the locally stored data should be as small as possible due to this disk space constraint.

The design of Warden in component and interface base is below.

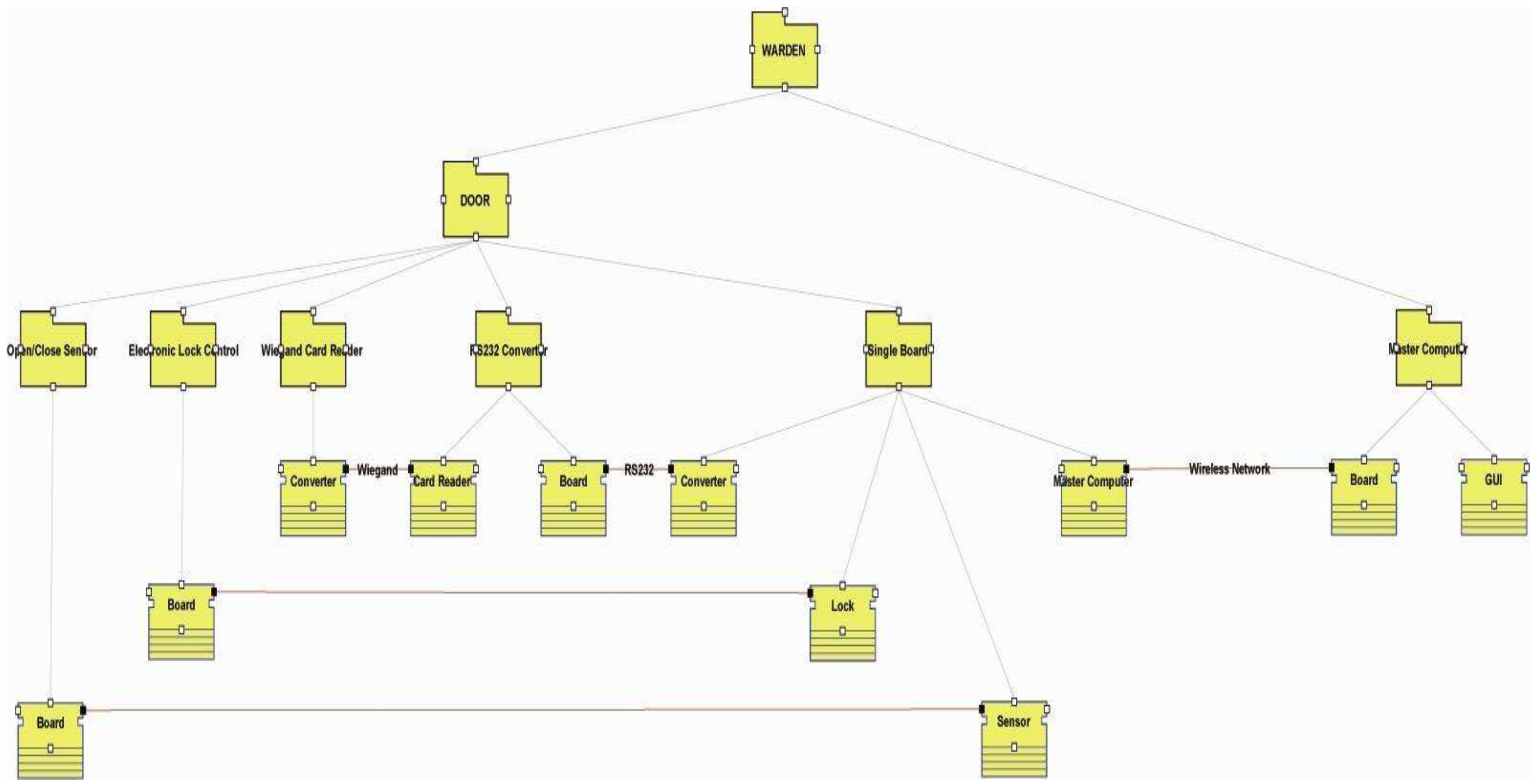


Figure 1 - Interfaces of components

2. COMPONENTS and INTERFACES

We can mainly decompose WARDEN into two main components: DOOR and Master Computer.

The DOOR component is also composed of 5 components: Open/Close Sensor, Electronic Lock Control, Wiegand Card Reader, RS232 Converter and Single Board. For all DOOR additions to system, these entire 5 components will be added.

In diagram the names of interfaces are put according to the component that is communicated with.

2.1. Open/Close Sensor – Single Board

The Open/Close Sensor component has an interface to interact with the Single Board component. The sensor will inform the single board about the open/closed situation of the door by one bit data. This will be achieved via the parallel port and the single board takes one bit data by pooling. We are planning to use a simulator in design process instead of this component.

2.2. Electronic Control Lock – Single Board

Similarly, the single board will open or close the door lock via parallel port. The output data is again one bit. This output signal and the electric supply for the lock will be given as input to an AND gate and the output will be connected to the lock. One pin of the port will be used to input from the sensor and one to output to the AND gate. We are planning to use a led in design process instead of this component.

2.3. Card Reader – Single Board (Converter)

2.3.1. Software of Converter

In the programmable integrated circuit, there will be a program which will be implemented in assembly language. Source code will be implemented in AVR Studio 3.53 environment. The duty of this code is converting Wiegand protocol to RS232 protocol to make computer environment understand Wiegand protocol. Also it must convert RS232 protocol to Wiegand to send the orders of computer environment to the card reader.

RS232 Protocol

RS232 is an asynchronous serial communications protocol, widely used on computers. Asynchronous means it doesn't have any separate synchronizing clock signal, so it has to synchronize itself to the incoming data; it does this by the use of 'START' and 'STOP' signals. The signal itself is slightly unusual for computers, as rather than the normal 0V to 5V range, it uses +12V to -12V.

Current usage of RS232 states that, data is transmitted in groups or characters of 7 or 8 bits. Each character is preceded by a start bit that must be 0 and is followed by at least one stop bit that must be a one. And also we have parity bit and it is optional. The parity may be odd, even or may not be present.

Examples:

- Let's send "troy" string by rs232. We choose 8 bit data and even parity.

  000101010110010011100101111011001010011110 11

  So, these bits are sent by left to right.

- There is another example with diagram for sending 0100111. As you can see in this example we don't use parity bit and we choose 7 bits data.

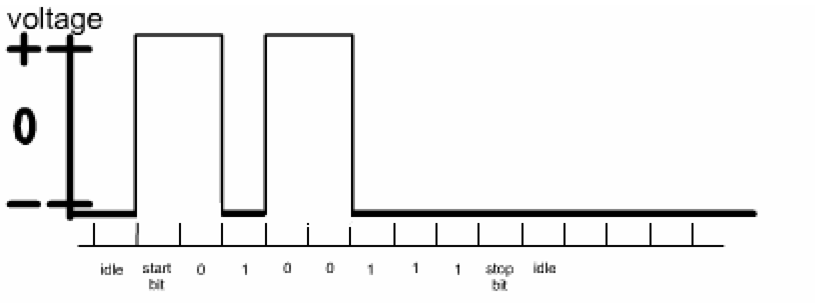


Figure 2 – Bit Sequence of 0100111

Wiegand Output Protocol

The Wiegand protocol (26 bit mode) itself is made up of a leading even parity bit (for b0 - b11), 24 bits of data (from transponder data) and a trailing odd parity bit (for b12 - b23). The 36 bit mode has the same format except 34 bits are used to form the data sequence.

For example:

```
H4001 tag data (Hex):        04 60 22 12 75
Wiegand 26 bit sequence:   E (b0 --------- b11) (b12 -------- b23) O
                           E (0       4       6      0       2      2) O
                           1 0000  0100  0110  0000  0010 00101
```

Where E is EVEN parity bit for bit 0 to 11 and O is ODD parity bit for bits 12 to 23.The physical Wiegand protocol is asynchronously transmitted as low going 50 mS pulses on the appropriate DATA low or DATA high pins. These pulses are separated by 2mS periods.

Wiegand Protocol Timing Diagram

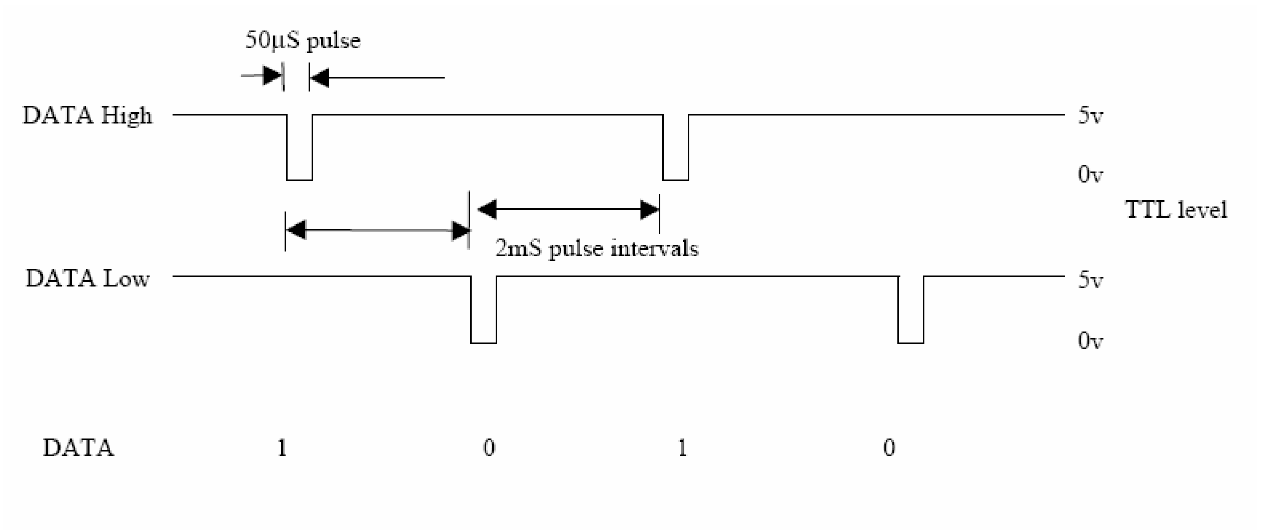


Figure 3 - Timing Diagram

This program will support two card readers. They can be Wiegand 26 or Wiegand 37 interface. Transmitted by the card reader data will be checked for errors and retransmitted via RS232 interface to single board. Both interfaces, Wiegand and RS232 will be implemented in software as interrupt service routines. Due to lack of RAM, transmission via RS232 and receiving from RS232 will be done separately.

There will be three interrupts in software, two for Wiegand interface and one for RS232 interface. There will be two interrupts for Wiegand interface since there will be two card readers at the doors, one inside one outside. Wiegand interfaces will use external interrupt INT0 and INT1 while RS232 interface will use timer T_0 overflow interrupt.

Timer T_0 overflow interrupt is used for both: running timeout timers of access control reader and RS232 transmissions. T_0 overflow interrupt generates interrupts during every RS232 bit. RS232 reception will be done by polling and transmission is done by T_0 overflow interrupt control. T_0 interrupt will count until RS232 transmission is completed since RS232 transmission time and one cycle of T_0 interrupt is not same. When count of T_0 interrupt is satisfied a new transmission is allowed. When transmission via RS232 is started, other two interrupts will be disabled to stop data receiving from readers. When transmission is completed or an error occurs interrupts will be enabled again.

Other two interrupts INT0 and INT1 will work almost same. There will be three main differences between these two interrupts are:

1- External Interrupt that they will use (one will use INT0 and other will use INT1)
2- Data pins data they will get data from reader.
3- Reader that they will get data from

Interrupts will start when a data comes from card reader. If a bit is received and an interrupt is invoked other interrupt will be disabled. If data transmission from card reader will be cut or all data is read then two interrupts will be enabled again. In both interrupts, during each period one bit Wiegand data will be read and stored in buffer. In Wiegand interface, there will a time delay between two bits. This time delay will be controlled by a timer. Number of data read from card reader will represent data format.

Since Wiegand interface has parity bits, in the software, parity check will be done. If an error has occurred, this data will not be send via RS232. If parity bits are correct then correct data will be send via RS232.

According to data received from RS232, LEDs and beeper will be set to invoke user about the result of process. Most of the routines in the program will be controlled by flags and these flags are set in T0 interrupt.

Technical Data:

- Wiegand data format	: Wiegand26 or Wiegand 37
- RS232	: full duplex
- RS232 speed	: 1200 bit/s
- RS232 data format	: 8N1

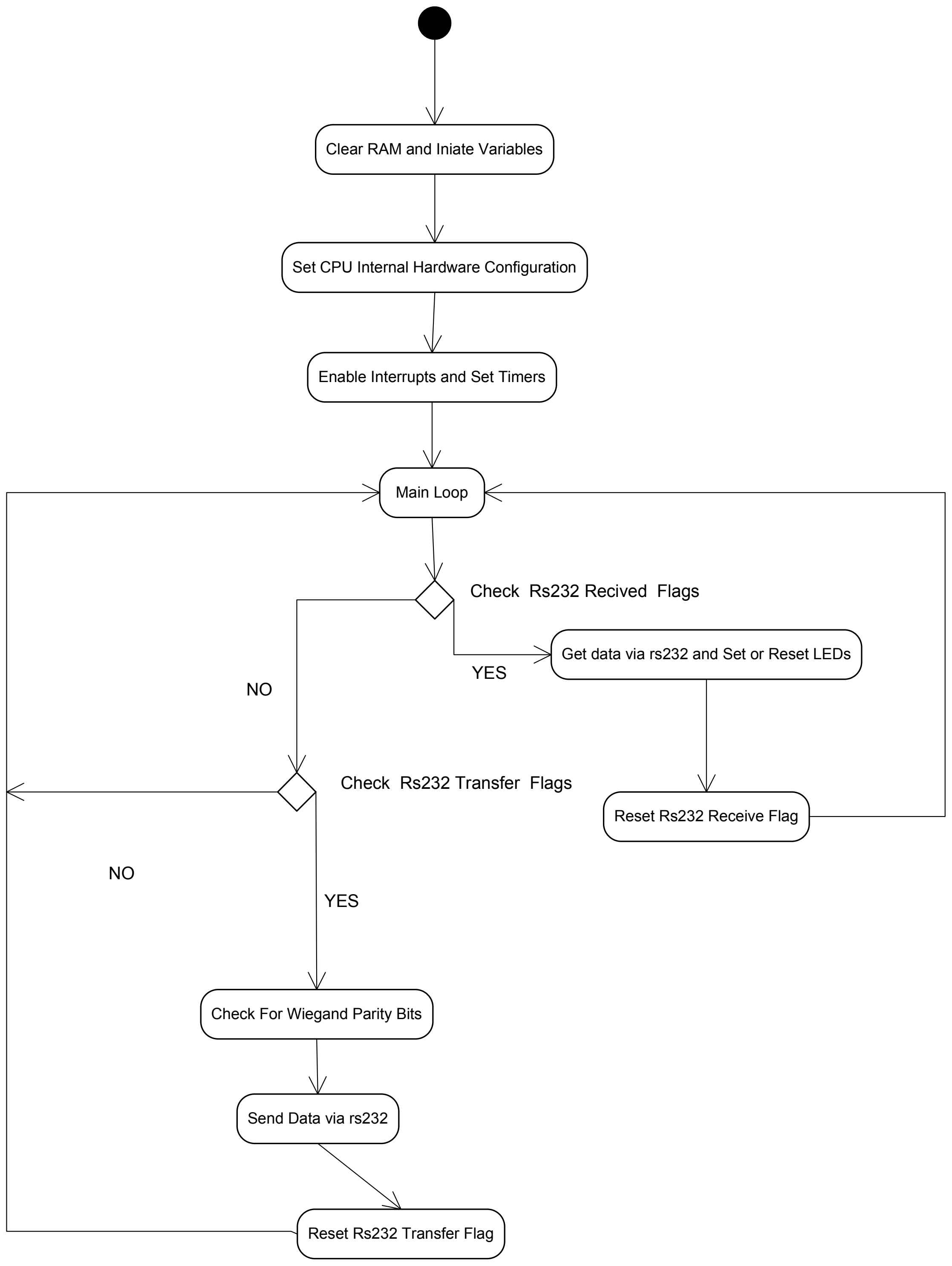

Clear RAM and Iniate Variables
Set CPU Internal Hardware Configuration
Enable Interrupts and Set Timers
Main Loop
Check Rs232 Recived Flags
Get data via rs232 and Set or Reset LEDs
YES
NO
Check Rs232 Transfer Flags
Reset Rs232 Receive Flag
NO
YES
Check For Wiegand Parity Bits
Send Data via rs232
Reset Rs232 Transfer Flag

Figure 4 - State Chart Diagram of Converter Application

This state chart diagram is the state chart of main program in programmable integrated circuit.

Clear RAM and Initiate Variables State

This state initiates all variables and clear RAM. When the program first started, this state is executed.

Set CPU Internal Hardware Configuration State

This state initiates status of internal structure of hardware and specifies input and output pins of programmable integrated circuit.

Enable Interrupts and Set Timers State

In code there are some interrupts and time counters. This state enables interrupts and initiates timer registers and start T0 interrupt.

Main Loop State

This is the main part of program. It sets and checks status of registers.

Get data via RS232 and Set or Reset LEDs State

This state receives data via RS232 and according to this data set or reset LEDs to invoke user.

Reset RS232 Receive Flag State

This is the end of receiving data via RS232. When data receiving finishes, status flags of RS232 are reset for another receive.

Check for Wiegand Parity Bits State

Wiegand protocol has parity bits to check correctness of data. Before sending Wiegand Data via RS232 these parity bits are checked.

Send Data via RS232 State

After receiving data from Wiegand protocol and checking its parity bits this data is send via RS232.

Reset RS232 Transfer Flag State

This step resets transfer flag for other transfers after transferring data via RS232 is completed.

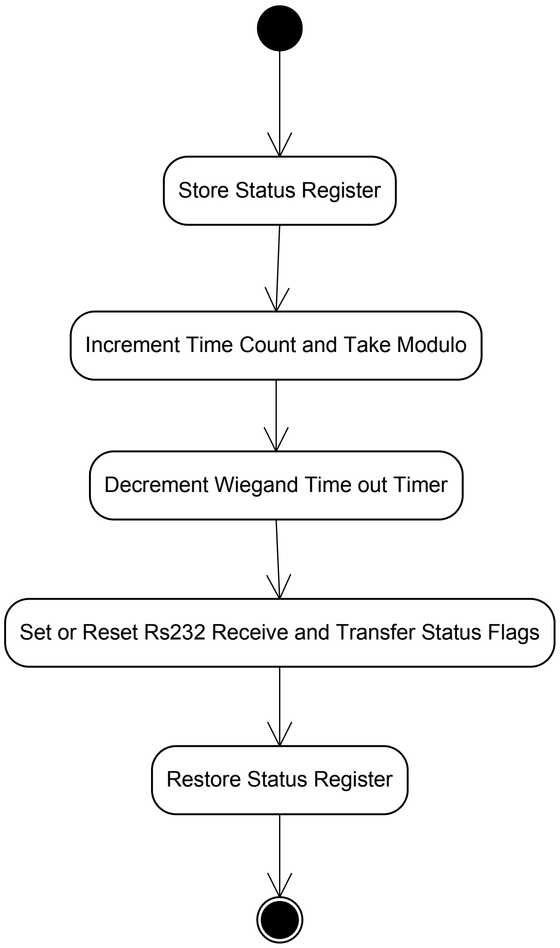

Store Status Register
Increment Time Count and Take Modulo
Decrement Wiegand Time out Timer
Set or Reset Rs232 Receive and Transfer Status Flags
Restore Status Register

Figure 5 - T_0 Interrupt State Chart Diagram

This interrupt is the main interrupt of the program in programmable integrated circuit. T0 interrupt is always executed and it specifies states according to timers of program. It decides sending, receiving data via RS232 and getting data from Wiegand protocol.

Store Status Register State

In the programmable integrated circuit, there are registers that control the status of executable program. In the execution of T_0, interrupt stores these registers not to modify these registers.

Decrement Wiegand Time out Timer State

Wiegand time out timer is used to check whether time to wait for other bit is exceed some Wiegand send data time or not. In every interrupt, this count is decremented and other interrupts are executed according to this timer.

Set or Reset Rs232 Receive and Transfer Status Flags State

Transferring and sending data via RS232 is done according to status of RS232. While sending data via RS232, status of program does not allow another send attempt till sending is finished. While receiving data from RS232, status of program does not allow another receive attempt till sending is finished.

Restore Status Register State

When T_0 interrupt is called program has some registers to execute program. After finishing execution of T_0 interrupt these registers are set back to their old values.

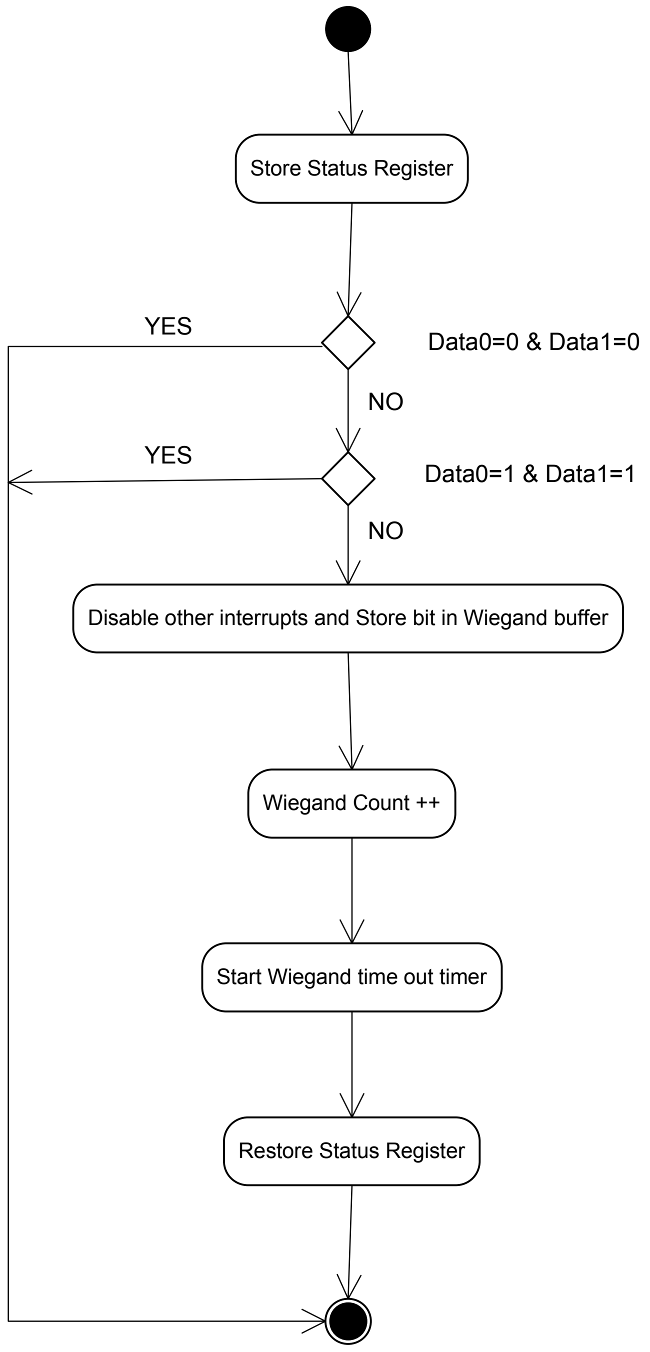


Figure 6 - Interrupt State Chart Diagram

Our program has two interrupts Int0 and Int1. These two interrupts work similar. Int0 controls card reader that is inside and Int1 controls card reader that is outside. These two interrupts is started from different pins and get data from different pins. When one interrupt starts other interrupt is disabled. Both interrupt send data via

RS232 and use some data pins. These interrupts starts when data flow starts from their data pins.

Store Status Register State

In the programmable integrated circuit, there are registers that control the status of executable program. In the execution of Int0 or Int1, interrupt stores these registers not to modify these registers.

Disable Other Interrupts and Store Bit in Wiegand Buffer State

When data flow starts from one of the data pins that interrupts follow, other interrupt is disabled to enable reading from one reader and data is received from reader. This data is stored in a register that is reserved for Wiegand data as buffer.

Wiegand Count ++ State

Reader sends data serial. Two decide whether transferring data is completed or not received character count is incremented.

Start Wiegand Time-out Timer State

While receiving data from reader, there is a delay between two data bits. To check this time delay interrupt set Wiegand Time-out counter and start to decrement.

Restore Status Register State

When Int0 or Int1 interrupt is called program has some registers to execute program. After finishing execution of Int0 or Int1 interrupt these registers are set back to their old values.

2.3.2. Hardware of Converter

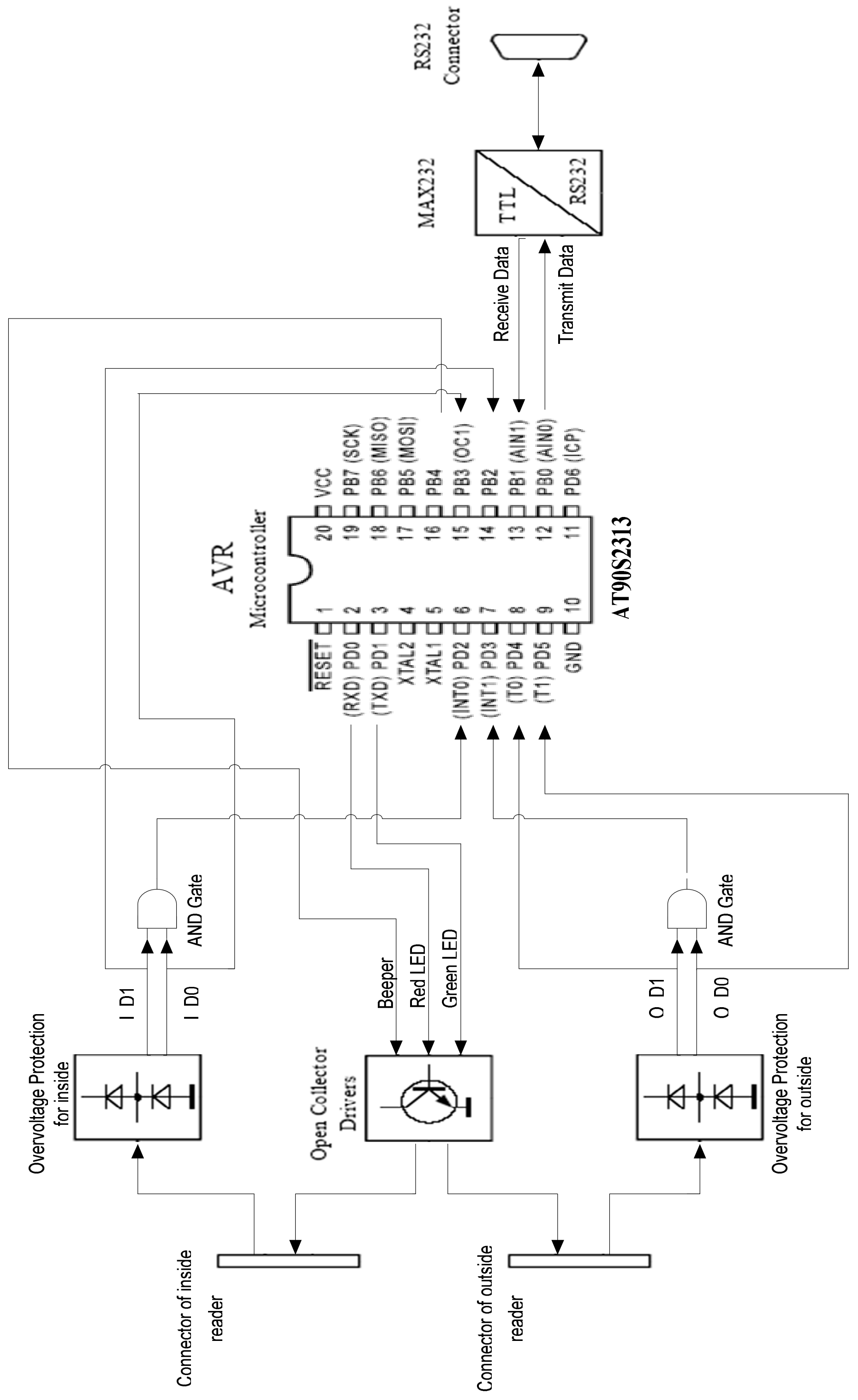
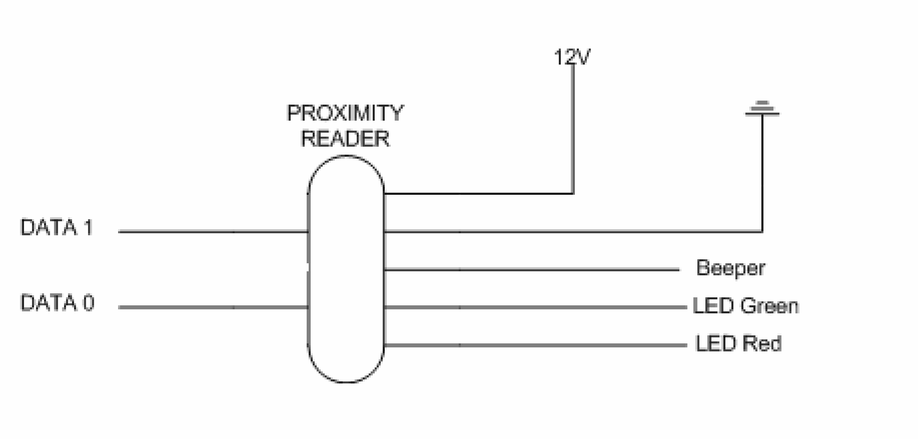


Figure 7 – General View of Hardware

2.3.3. Parts of Converter:

2.3.3.1. 2 Proximity Readers:

In our system there are two readers for each door. Theses reader is used for taking card-id from user. Then these readers send card-id by Wiegand protocol which is 26 or 37 bit serial protocol. Our converter supports both of them. Proximity reader has 2 outputs and 5 inputs as shown in below diagram.



We have Data 1 and Data 0 lines as output. And also we have 12V, ground, Beeper, LED Green and LED Red as input.

Beeper, LED Green and LED Red inputs are used for returning the validity of card-id. This information come form single board computer via rs232.

The reader transmits data by pulling low “Data 0” line when sending logic zero and “Data 1” line when sending logic one. The bit pulse lasts for about 50ms while interval between two bits equals approximately 2ms. Absence of pulses for about 200ms signals the new data block.

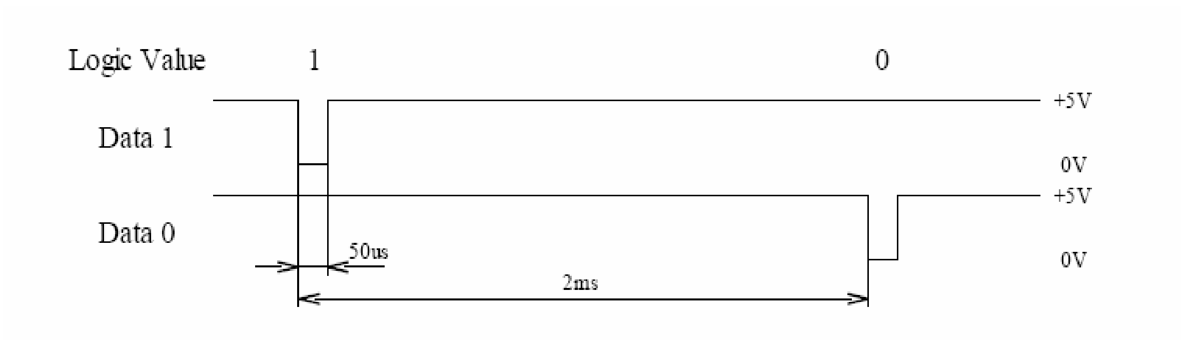


Figure 8 - Wiegand interface waveforms diagram

Wiegand 26 data block consists of 26 bits. The first bit and the last bit are even parity bit of the first half of the data block and odd parity bit of the second half of the data block respectively, while bits b2...b9 and bits b10...b25 represent 8-bit facility code and 16-bit card number respectively. The structure of Wiegand 26 data block is shown in below.

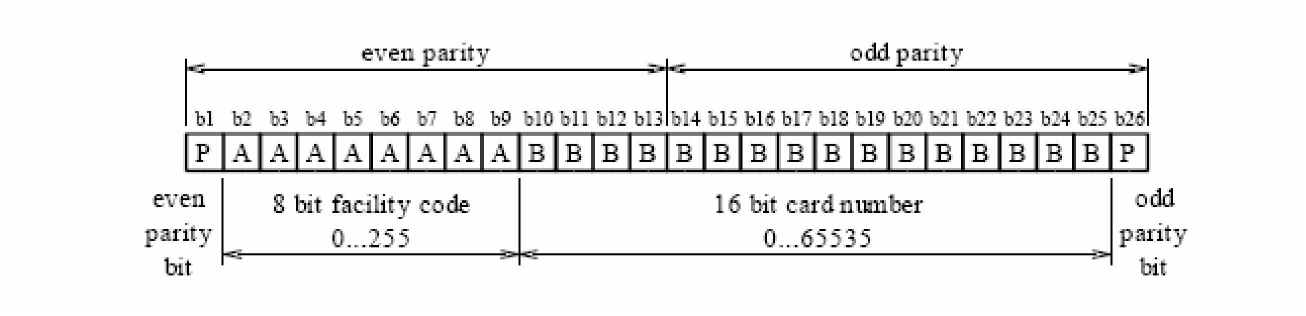

even parity
odd parity
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16 b17 b18 b19 b20 b21 b22 b23 b24 b25 b26
P A A A A A A A A B B B B B B B B B B B B B B B B P
even parity bit
8 bit facility code
0...255
16 bit card number
0...65535
odd parity bit

Figure 9 - Structure of Wiegand 26 data block

Wiegand 37 data block consists of 37 bits. As in Wiegand 26, the first bit and the last bit are even parity bit of the first half of the data block and odd parity bit of the second half of the data block respectively. Owing to the odd number of bits in the data block, the bit b19 is counted for both even and odd parity. Bits b2...b36 represent 35-bit card number. The structure of Wiegand 37 data block is shown in below.

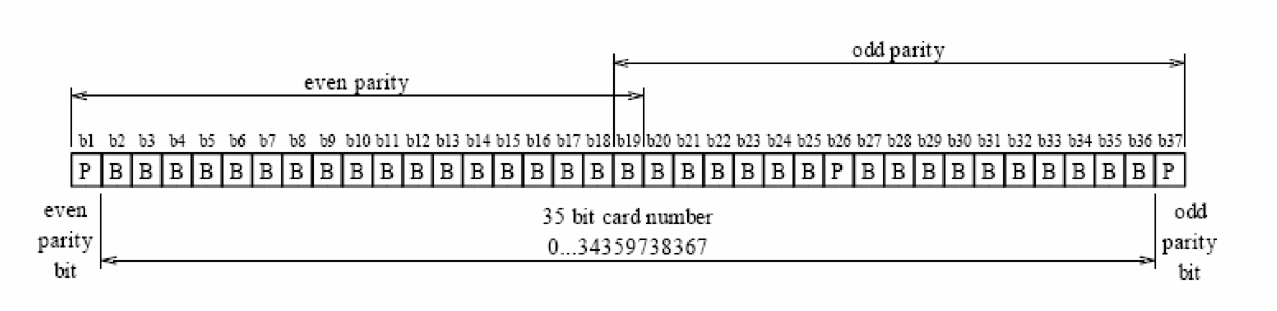

odd parity
even parity
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16 b17 b18 b19 b20 b21 b22 b23 b24 b25 b26 b27 b28 b29 b30 b31 b32 b33 b34 b35 b36 b37
P B B B B B B B B B B B B B B B B B B B B B B B B P B B B B B B B B B B P
even parity bit
35 bit card number
0...34359738367
odd parity bit

Figure 10 - Structure of Wiegand 37 data block

2.3.3.2. Over voltage Protection unit

In our system we use AVR microcontroller. We have to prevent our microcontroller against the over voltage. So, we include this unit in our system. This unit takes two input from Wiegand which are DATA 0 and DATA 1 line and it has two outputs which are OD_0 and OD_1.

2.3.3.3. AND Gate

As we explained, Wiegand protocol sends data when one of the lines comes to zero. In software of our system we take the Wiegand input by interrupt. We have two reader so we have two and gate. First AND gate generates interrupt (Int1) whenever negative pulse on 0_D1 or 0_D0 line appears. Second AND gate generates interrupt (Int0) whenever negative pulse on I_D1 or I_D0 line appears.

2.3.3.4. AVR Microcontroller

We choose AT90S2313 model which is production of ATMEL Corporation. The AT90S2313 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. We add instruction set to the Appendix. By executing instructions in a single clock cycle, the AT90S2313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed. The AVR core combines a instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle.

There are two main reasons why we choose AT90S2313 microcontroller. Firstly, it has two interrupts which are INT 0 and INT 1 in PD2 and PD3 pin. Secondly, it has timer, and we need timer because it we will check rs232 input in every particular time interval. Thirdly, this microcontroller provides us a developer studio (AVR Studio) and simulator of microcontroller.

This microcontroller has 20 pin. 2 pin for VCC and Ground, 1 pin for reset, 7 pin for PD0-PD6, 8 pin for PB0-PB7 and 2 pin for Oscillator which are XTAL1 and XTAL2.

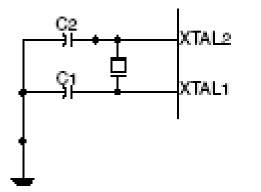


Figure 11 - Oscillator Connections

By using this oscillator we will create 7.3728 MHz.

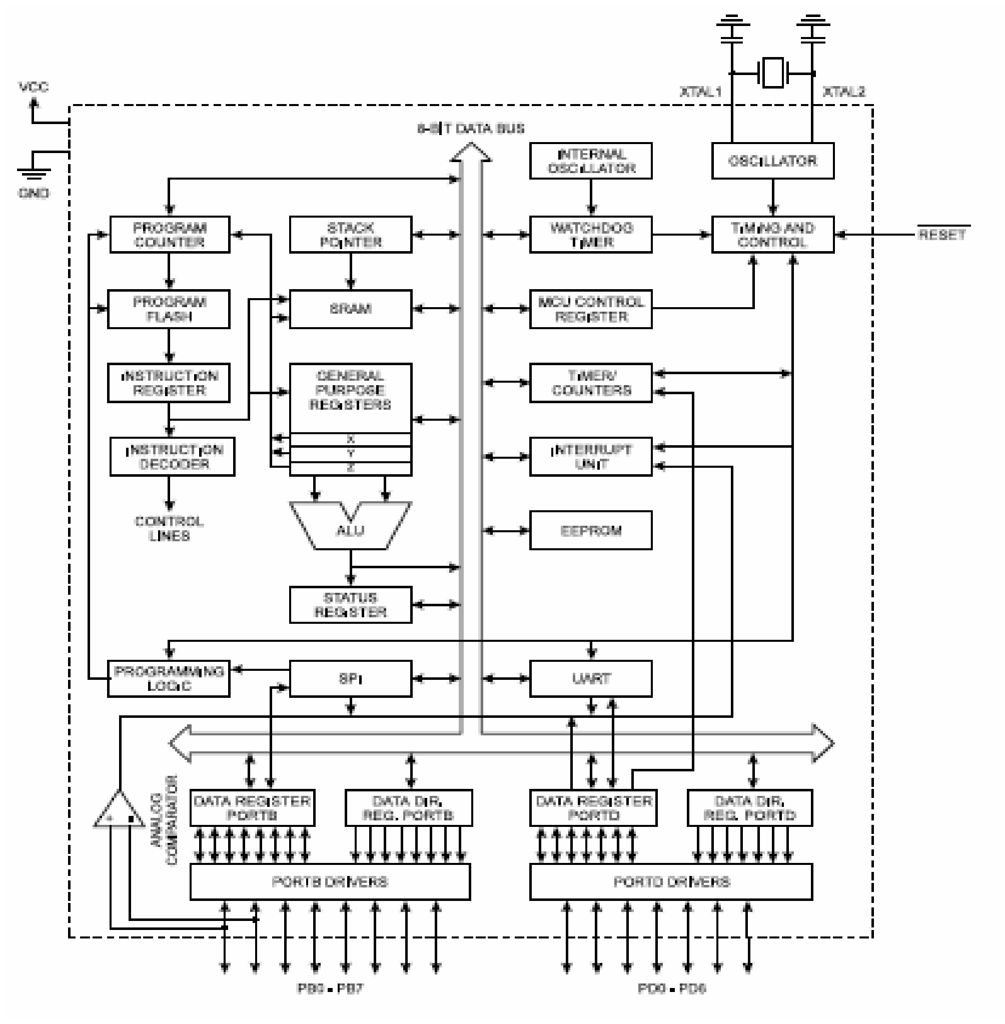


Figure 12 - The AT90S2313 Block Diagram

2.3.3.5. MAX232

This chip is an Rs232 voltage level connector. As you can see below diagrams, our microcontroller use “TTL/CMOS Serial Logic Waveform” and our single board computer use “RS-232 Logic Waveform”. For transferring “TTL/CMOS Serial Logic” to “RS-232 Logic” and also for “RS-232 Logic” to “TTL/CMOS Serial Logic” , we use MAX232 circuit.

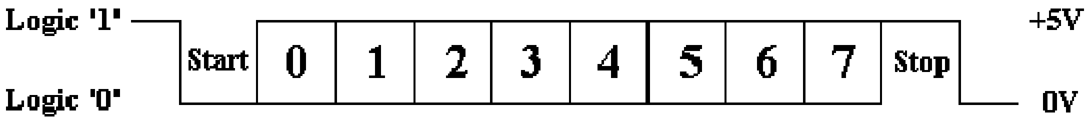


Figure 13 - TTL/CMOS Serial Logic Waveform

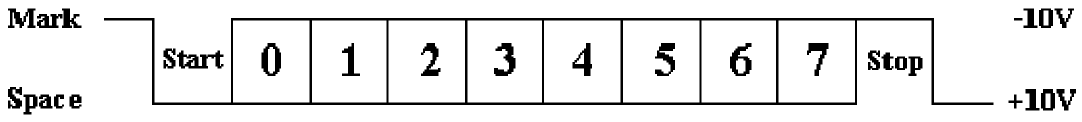


Figure 14 - RS-232 Logic Waveform

We connect PB1 and PB0 pin of AVR Microcontroller to the TR2IN and R2OUT and we connect ground to TR1IN respectively. And take the output from TR2OUT, R2IN and TR1OUT and connect these pin to the rs232 connector.

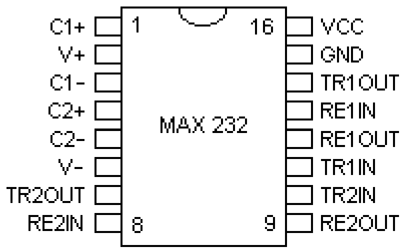


Figure 15 - Pin outs for the MAX232

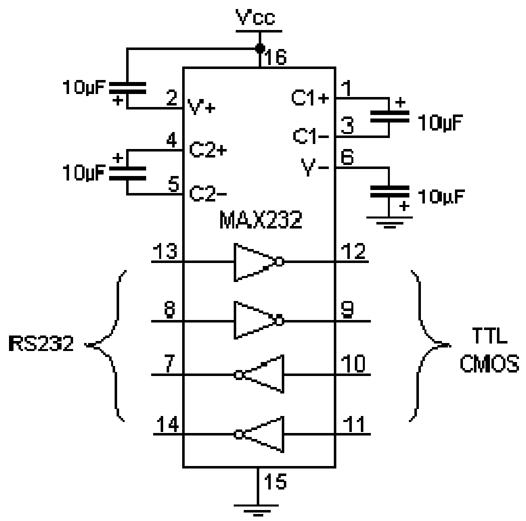


Figure 16 -Typical Max232 Circuit

2.3.3.6. RS232 Connector

We take TR2OUT, R2IN and TR1OUT pin, then we connect these pin to RS232 connector as shown in below diagram.

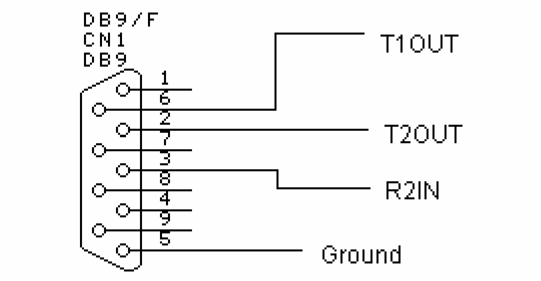


Figure 17 - RS232 Pin Connection

Figure 18 - RS232 Connector End

2.3.3.7. Power Supply

We use voltage regulator LM78L05 unit. Above diagram takes power between +9...15Vdc and generates always constant voltage which are +12V and +5V.

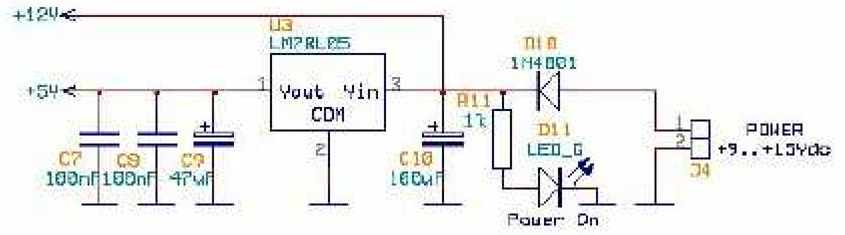
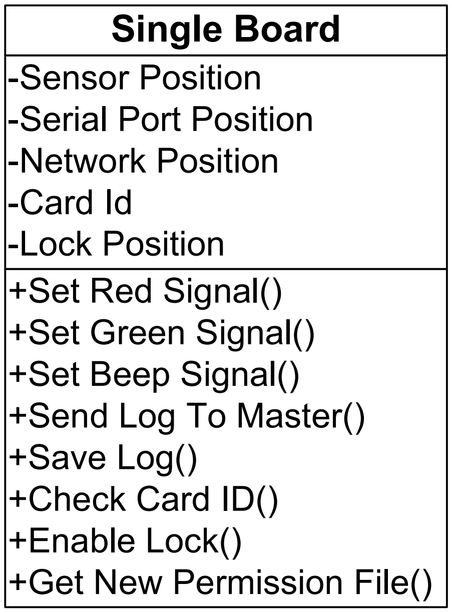


Figure 19 -Diagram of power supply of the converter

2.4. Single Board

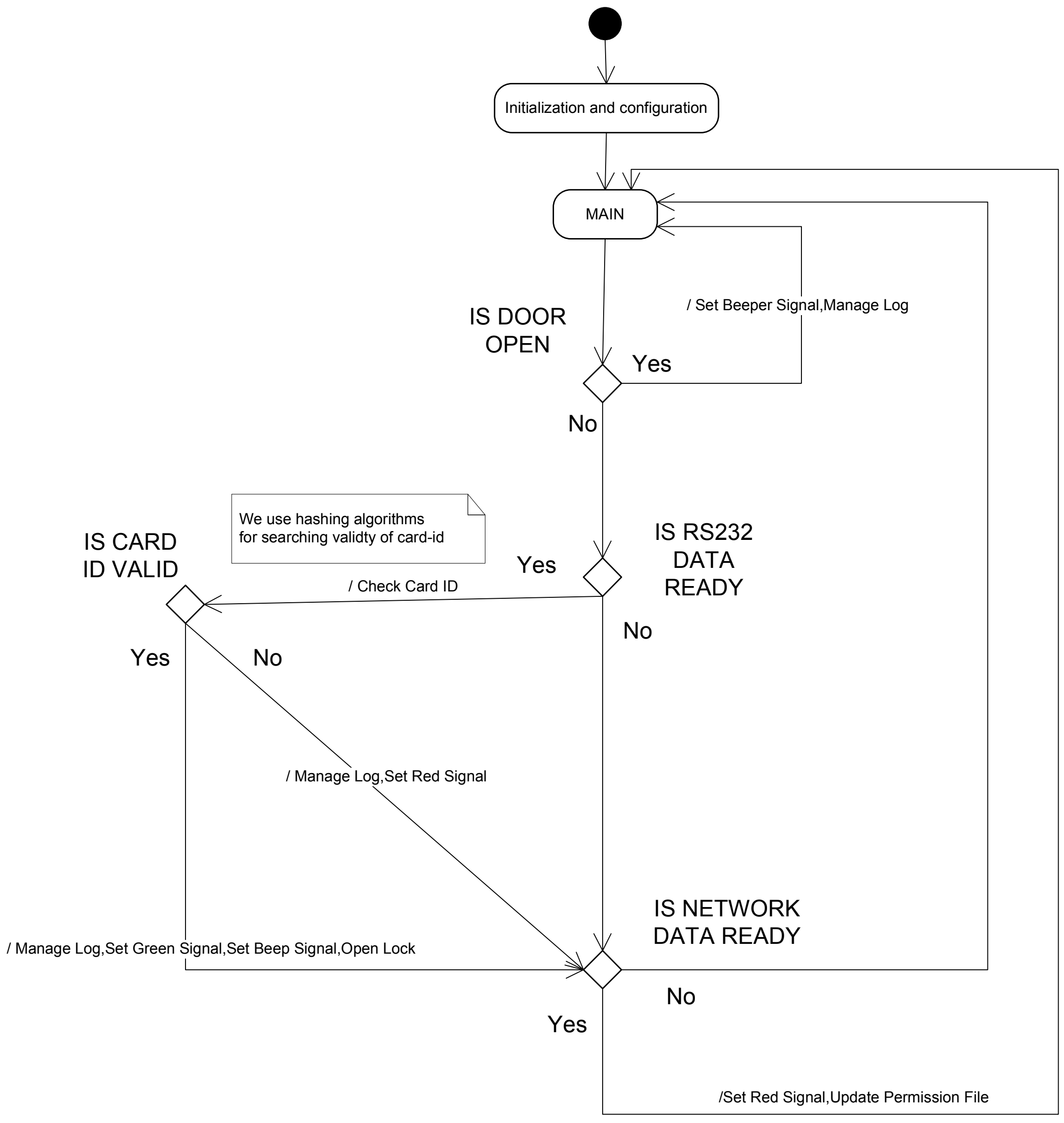
The application that will be run on single board is just an interface between master computer and doors. This program makes operations faster to evaluate. It doesn't be a complicated one, but should run as faster as it can be. We design it as one class structure.

2.4.1. Single Board Class



The program will be implemented as a loop that waits some guards to change. The main structure is pooling.

2.4.2. State Chart of Single Board Class



Initialization and configuration
MAIN
IS DOOR OPEN
Yes
/ Set Beeper Signal,Manage Log
No
We use hashing algorithms for searching validty of card-id
IS CARD ID VALID
Yes
IS RS232 DATA READY
/ Check Card ID
No
Yes
No
/ Manage Log,Set Red Signal
IS NETWORK DATA READY
/ Manage Log,Set Green Signal,Set Beep Signal,Open Lock
No
Yes
/Set Red Signal,Update Permission File

Figure 20 - State Chart Diagram of Single Board

2.4.3. Definition of State Chart Diagram

Set Red Signal:

Send a signal to the converter that turns on the red LED of card reader.

Set Green Signal:

Send a signal to the converter that turns on the green LED of card reader.

Set Beep Signal:

Send a signal to the converter that activates beeper of card reader.

These three set signals are sending by serial port. Writing to the port is explained below.

Pseudo Code

```
PROCEDURE Set Port

        GET (Port)

        WHILE Port NOT EQUAL Port Array Element
                NEXT Port Array Element
    END WHILE

        IF END of Port Array REACHED
                THEN Port NOT FOUND
                ELSE PORT <- Port Array Element
END Set Port


PROCEDURE Write Port

        Set Port

        ADD LISTENER TO USER

        WHEN EVENT
                IF EVENT EQUAL DATA TO SEND
                THEN WRITE DATA
        END WHEN

END Write Port
```

Manage Log (Save Log + Send Log to Master) :

Single board will send every log entry to master computer immediately by encoding. But because of some electricity cut or other problems this log entry may not be delivered immediately. For this case, single board will store log file internally. This log data also will be store encoded. In single board, there is no much memory because of this log file will be stored circular. It will take last 5000 entry. Also there will be another file to store current line of log file. If an error occurs current line of log file will be taken and then storing logs to file will continue from this position without any loose of data. To ensure that log and current line is stored in file these two files will be closed and opened.

Log data will contain:

- User card id(16 bit)
- Operation type(entry, exist, force attempt)(2 bits)
- Time (11 bits = 5 bit hour field and 6 bits minute field)
- Date(16 bits = 5 bits day 4 bits mount and 7 bits year)

One log data will be 45 bits. For 5000 log data will be approximately 30Kbytes.

Pseudo Code

At the beginning log file is full of with a special character. We only change these characters to log data. Also current file position is 0.

```
Int iLoopCouter = 0;
Open(logFile);
Open(currentPosFile)
While(true)
{
        if(a card is read)
        {
                decide on operation type;
                produce log data;
                encode data;
                send log data;
                if(current position > end of logFile)
                {
                    set current pos to 0;
                }
                go to current position in logFile
                write log data to logFile;
                increment current position;
                write current pos to currentPosFile;
                increment iLoopcount;
                if(iLoopCount == 100)
                {
                    iLoopCounter = 0;
                    fclose(logFile);
                    fclose(currentPosFile);
                    fopen(logFile);
                    fopen(currentPosFile);
                }
        }
}
```

Get New Permission File:

Master computer will send permission files to doors in case of request from master computer user. When a send request come from master computer to single board, single board will start to listen network. While transferring, single board will decode permission file and store it. Since we will need all file in the single board, file will not be closed and opened while transferring permissions.

Pseudo Code

```
If(a send request from master exists)
{
    fopen(permissionFile);
    clear permissionFile;
    while(transmittion)
    {
        get file data;
        decode data;
        write to file according to hash order;
    }
    fclose(permissionFile);
```

}

Check Card ID:

After the card ID is taken from serial port by the code below, it is searched in the permission file that is stored.

Pseudo Code

```
PROCEDURE Set Port

        GET (Port)

        WHILE Port NOT EQUAL Port Array Element
                NEXT Port Array Element
    END WHILE

        IF END of Port Array REACHED
                THEN Port NOT FOUND
                ELSE PORT <- Port Array Element
END Set Port

PROCEDURE Listen Port

        Set Port

        ADD LISTENER TO PORT

        WHEN EVENT
                IF EVENT EQUAL DATA AVAILABLE
                        THEN READ DATA
        END WHEN

END Listen Port
```

2.4.4. Hardware of Single Board

The hardware supports that main board VIA ATX Main Board has are below.

- Processor
- RAM (128 MB)
- Portable Hard Disk (Flash Disk, etc)
- WiFi Ethernet Card

The operating system will be embedded Linux. Our program will run on this operating system. Both will be in flash disk.

2.5. Master Computer

Most bubbles in use case diagram belong to master computer.

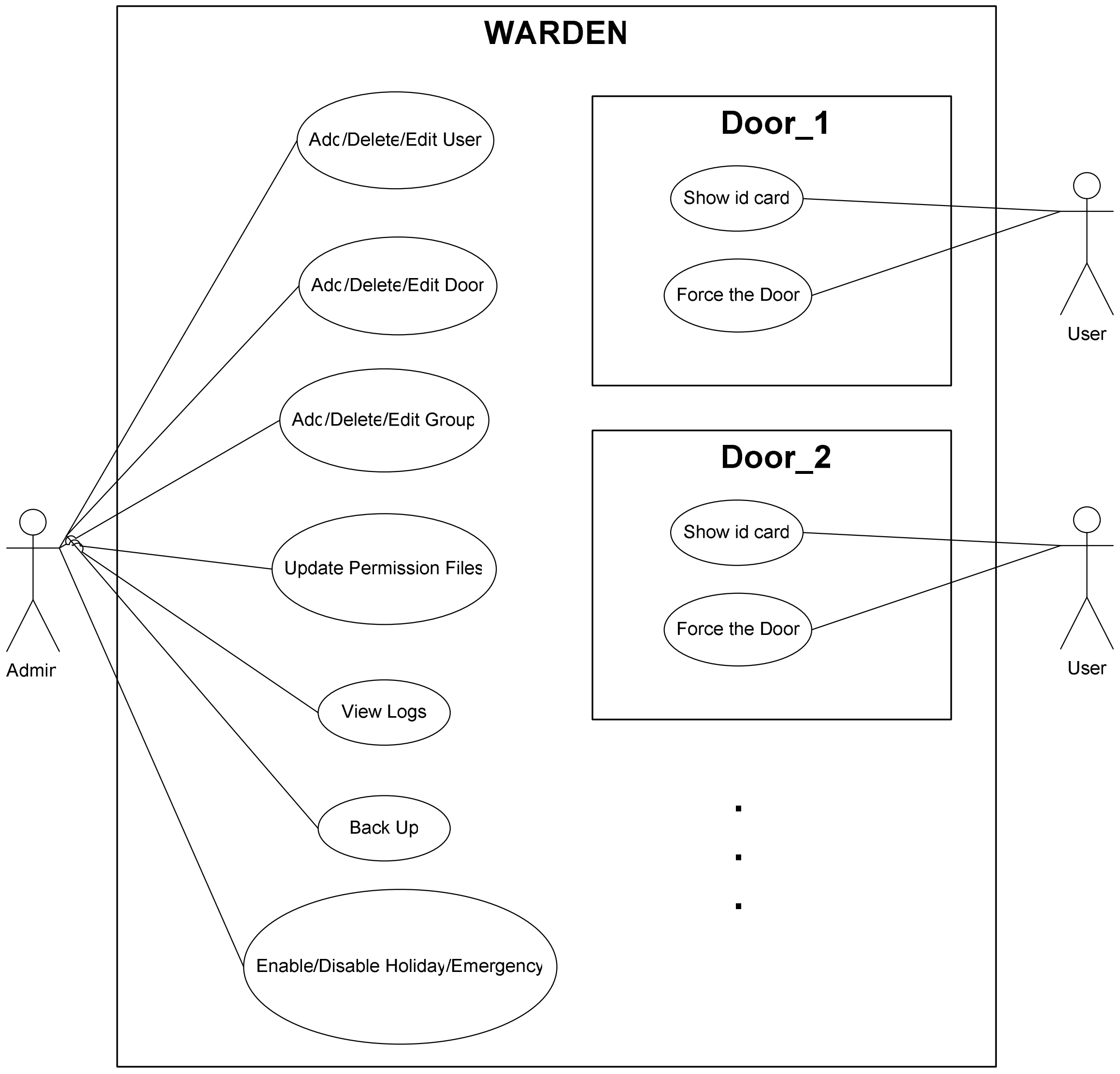


Figure 21 – Use Case Diagram of Warden

The user interfaces of warden are only the graphical application in master computer, card readers and the doors. The ones other than graphical user interface are not needed to design. The design of GUI is in this part.

2.5.1. Class Diagrams

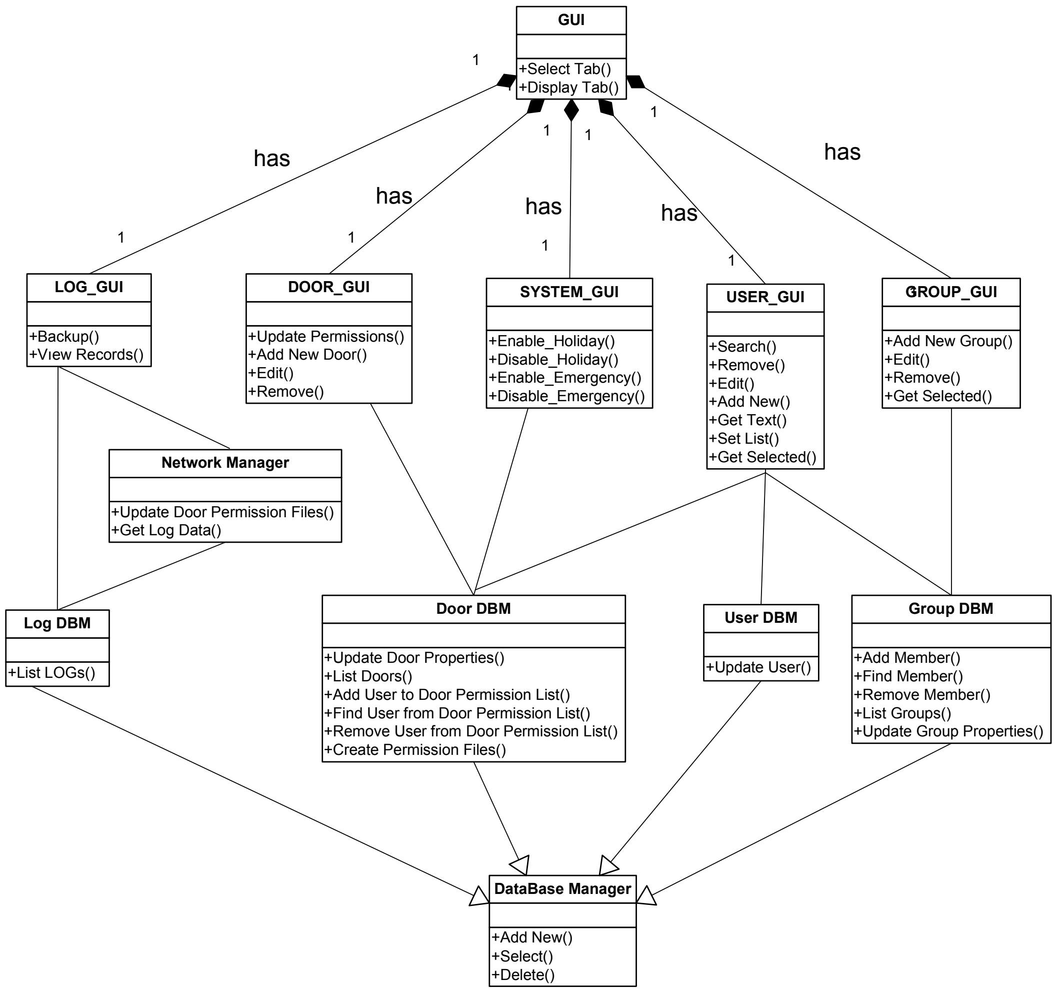


Figure 22 - Class Diagram of Master Computer Application

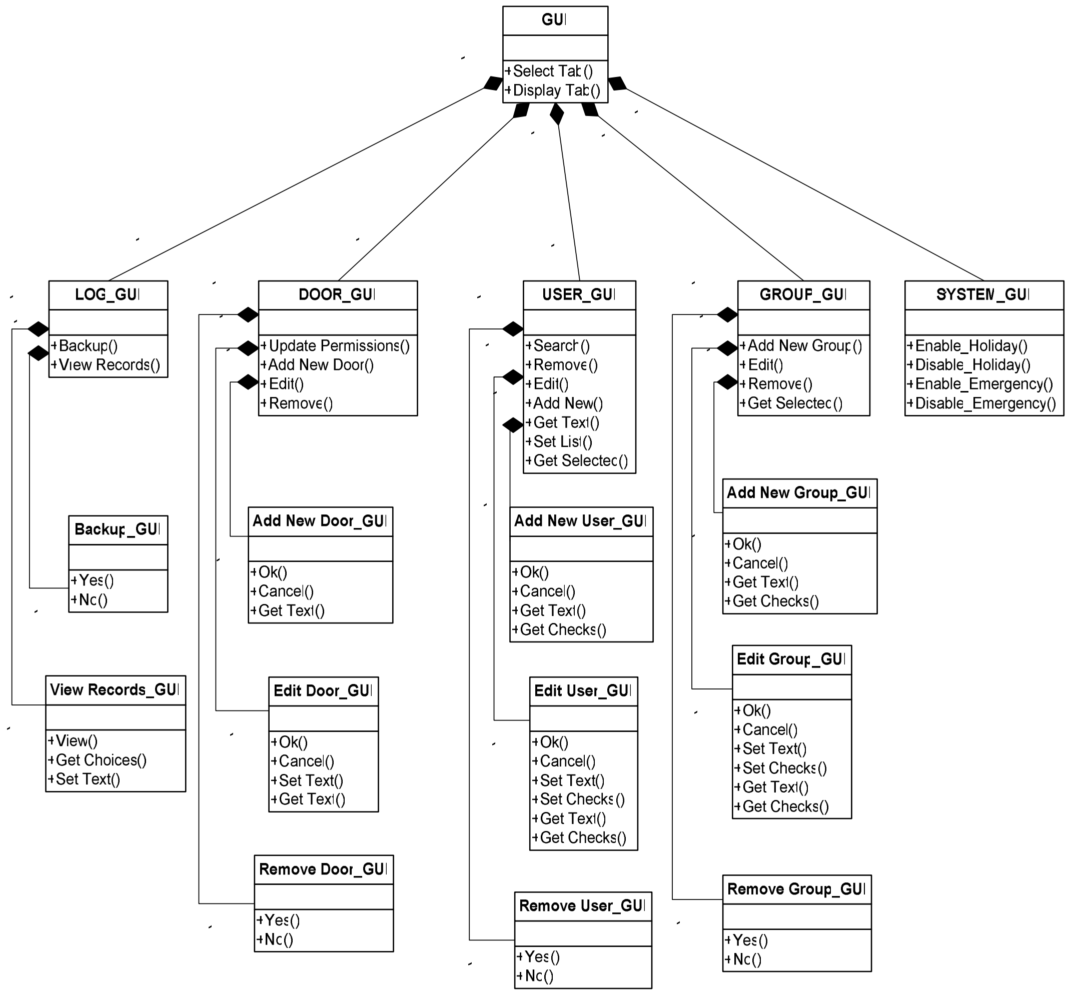
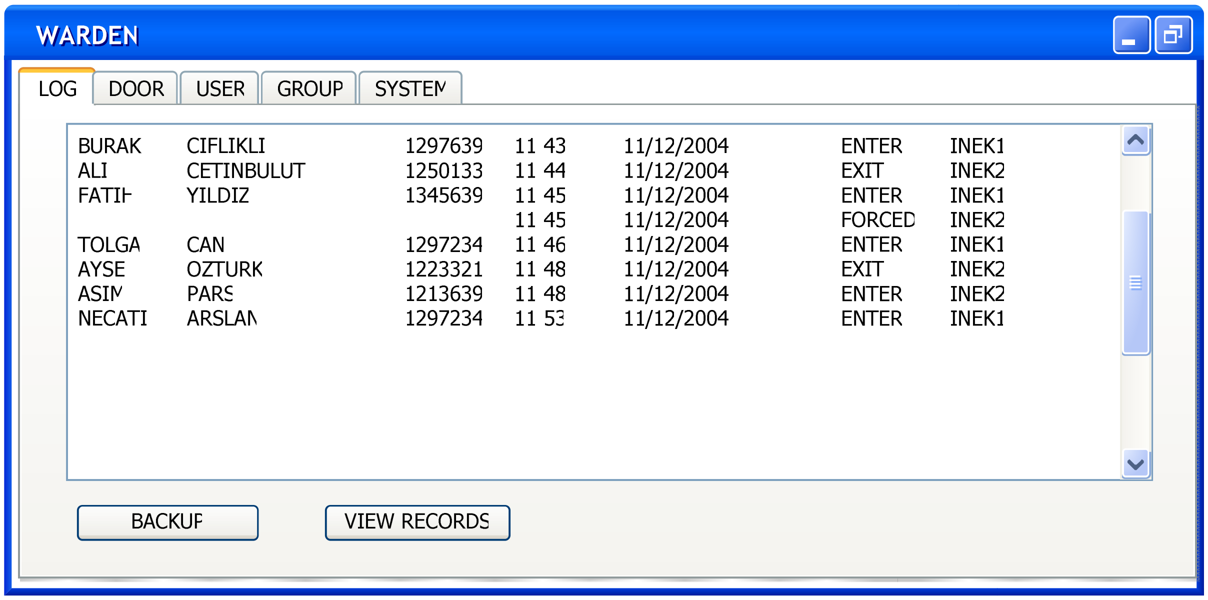


Figure 23 – Extended Class Diagram of GUI Classes

2.5.2. Graphical User Interface Design

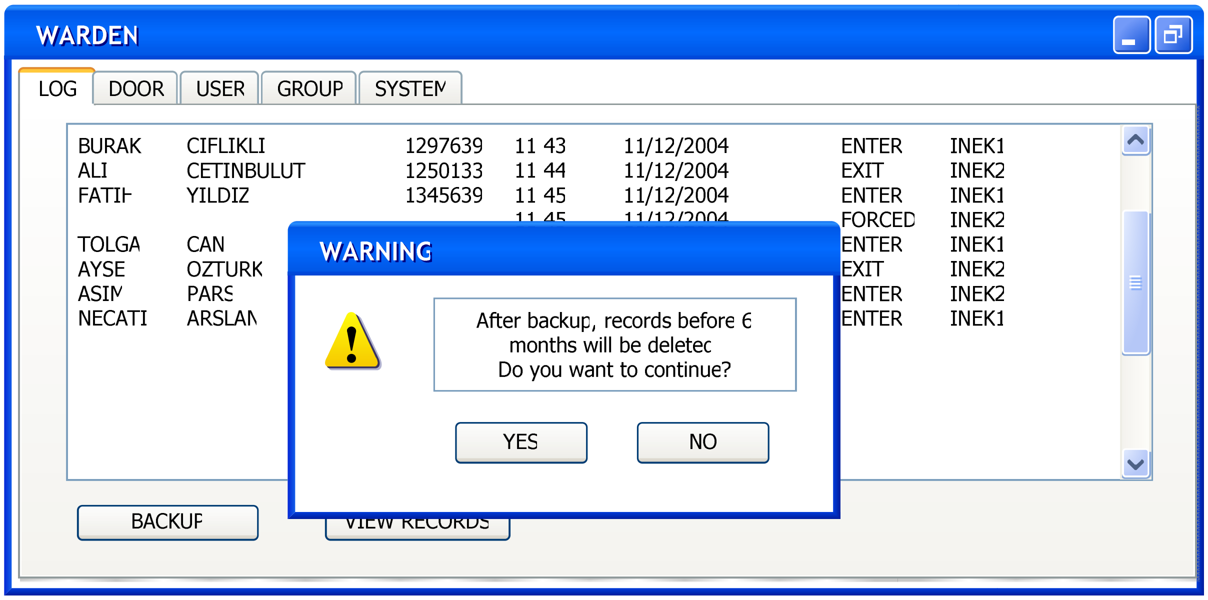
2.5.2.1. LOG



BURAK	CIFLIKLI	1297639	11 43	11/12/2004	ENTER	INEK1
ALI	CETINBULUT	1250133	11 44	11/12/2004	EXIT	INEK2
FATIH	YILDIZ	1345639	11 45	11/12/2004	ENTER	INEK1
			11 45	11/12/2004	FORCED	INEK2
TOLGA	CAN	1297234	11 46	11/12/2004	ENTER	INEK1
AYSE	OZTURK	1223321	11 48	11/12/2004	EXIT	INEK2
ASIM	PARS	1213639	11 48	11/12/2004	ENTER	INEK2
NECATI	ARSLAN	1297234	11 53	11/12/2004	ENTER	INEK1

This is the main window of the LOG tab of the GUI of the master computer. From this window, the events that are formed at that moment will be traced. The user name, surname, user ID, event date, event time, event name and event location will be displayed. The "BACKUP" button will be used to take a backup of the records and the "VIEW RECORDS" button will be used to display previous records.

BACKUP

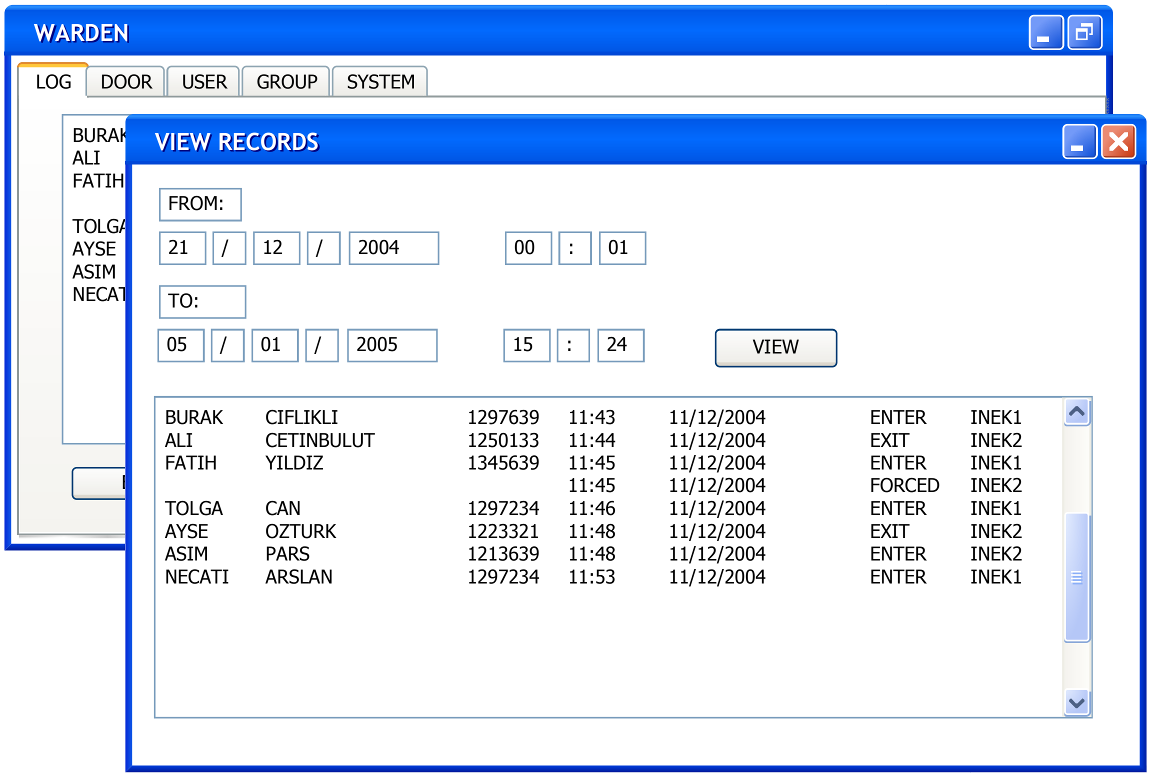


BURAK	CIFLIKLI	1297639	11 43	11/12/2004	ENTER	INEK1
ALI	CETINBULUT	1250133	11 44	11/12/2004	EXIT	INEK2
FATIH	YILDIZ	1345639	11 45	11/12/2004	ENTER	INEK1
			11 45	11/12/2004	FORCED	INEK2
TOLGA	CAN				ENTER	INEK1
AYSE	OZTURK				EXIT	INEK2
ASIM	PARS				ENTER	INEK2
NECATI	ARSLAN				ENTER	INEK1

After backup, records before 6 months will be deleted
Do you want to continue?

This window will be opened when the "BACKUP" button of the main window of the LOG tab is pressed. A backup is formed in case there is not enough disk space in the master computer. So after the backup, the logs of the system will be copied to a backup folder, then the logs before 6 month will be deleted. The user will be informed about the situation by this warning window. If the "YES" button is pressed, the backup will be taken, and if the "NO" button is pressed, the operation will be cancelled and the warning window will be closed.

VIEW RECORDS



FROM:

21 / 12 / 2004   00 : 01

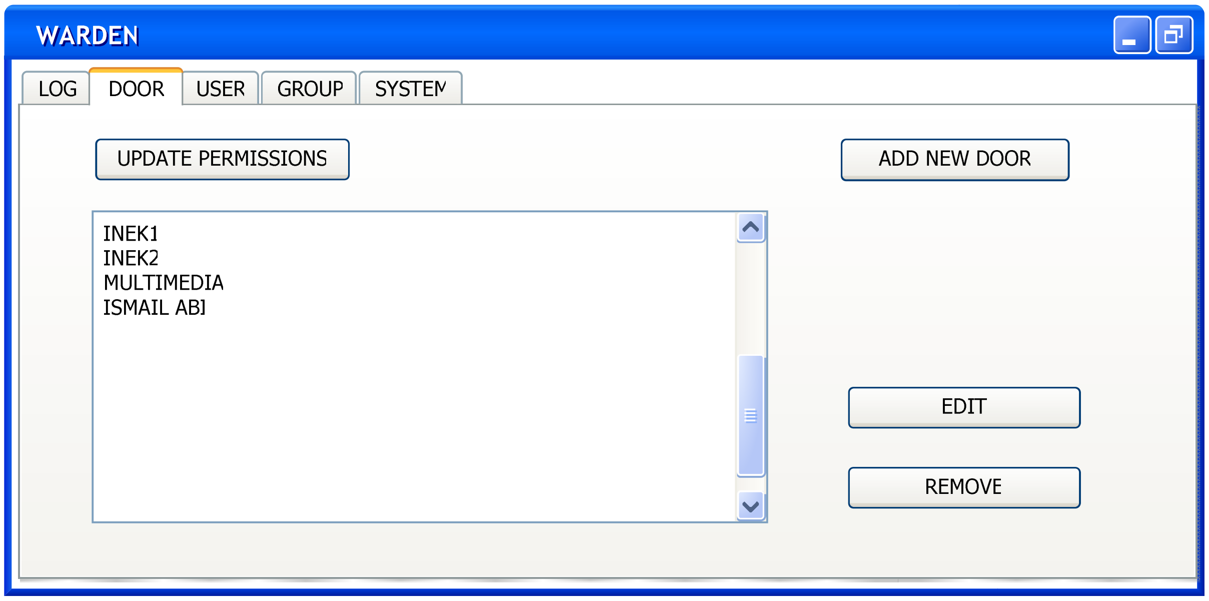
TO:

05 / 01 / 2005   15 : 24

```
BURAK    CIFLIKLI      1297639   11:43   11/12/2004   ENTER    INEK1
ALI      CETINBULUT    1250133   11:44   11/12/2004   EXIT     INEK2
FATIH    YILDIZ        1345639   11:45   11/12/2004   ENTER    INEK1
                                 11:45   11/12/2004   FORCED   INEK2
TOLGA    CAN           1297234   11:46   11/12/2004   ENTER    INEK1
AYSE     OZTURK        1223321   11:48   11/12/2004   EXIT     INEK2
ASIM     PARS          1213639   11:48   11/12/2004   ENTER    INEK2
NECATI   ARSLAN        1297234   11:53   11/12/2004   ENTER    INEK1
```

This window will be opened when the "VIEW RECORDS" button of the main window of the LOG tab is pressed. The user will choose the time interval of the events of whose records are wanted to be displayed. This interval will be squeezed into the record interval of the system; user will not be able to request for the previously deleted records. When the "VIEW" button is pressed, the logs within the chosen interval will be displayed.

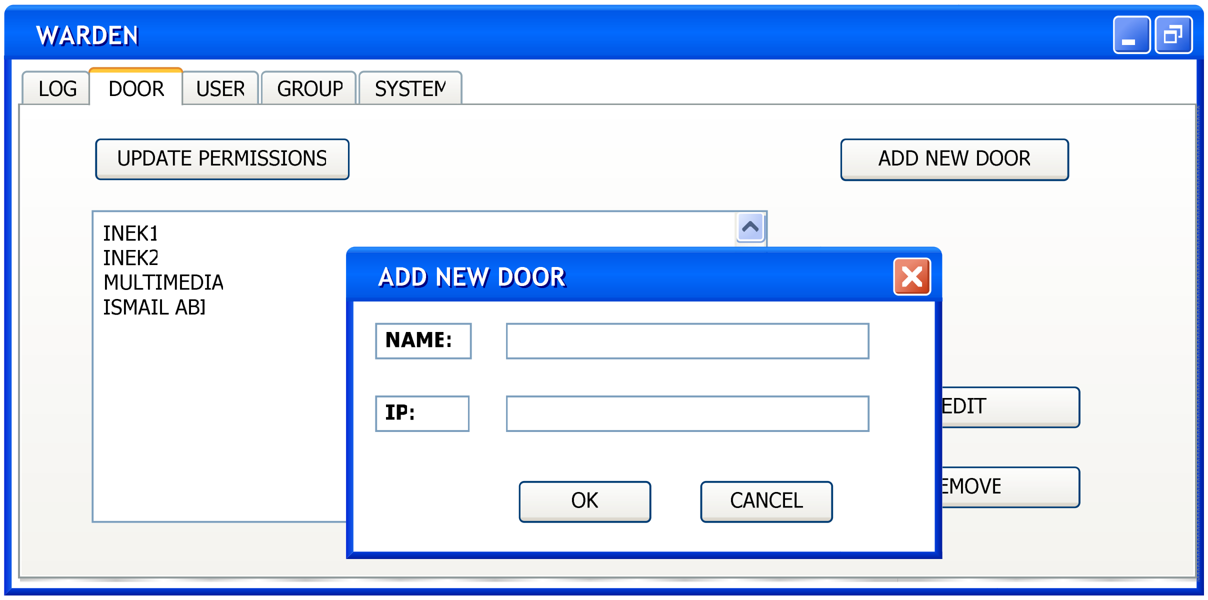
2.5.2.2. DOOR



INEK1
INEK2
MULTIMEDIA
ISMAIL ABI

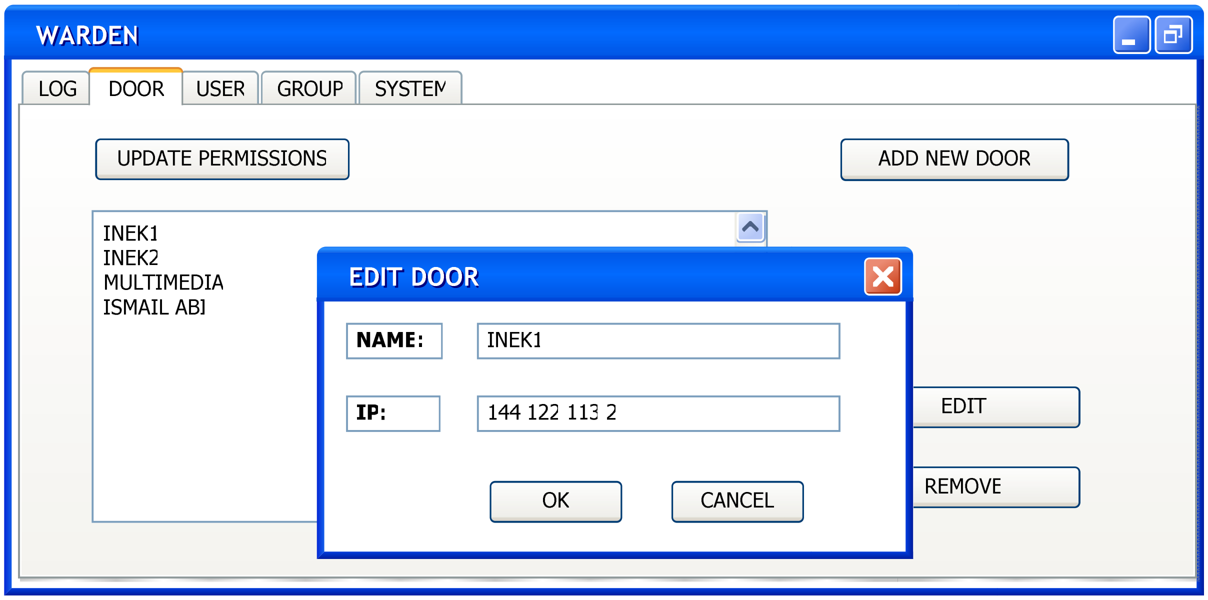
This is the main window of the DOOR tab of the GUI of the master computer. From this window, the doors will be added, removed and edited. The "UPDATE PERMISSIONS" button will be used to send the permission files from the master to the doors. "ADD NEW DOOR" button will be used to install a new door to the system. To edit a previously installed door, the "EDIT" button will be used. Firstly a door from the list will be selected and this button will be pressed. Similarly, to remove a selected door, the "REMOVE" button will be pressed.

ADD NEW DOOR



This window will be opened when the "ADD NEW DOOR" button of the main window of the DOOR tab is pressed. The user will enter the name and the network IP of the door. If the "OK" button is pressed, the door will be installed. If a problem occurs in installation user will be informed. If the "CANCEL" button is pressed, the operation will be cancelled and the window will be closed.

EDIT DOOR

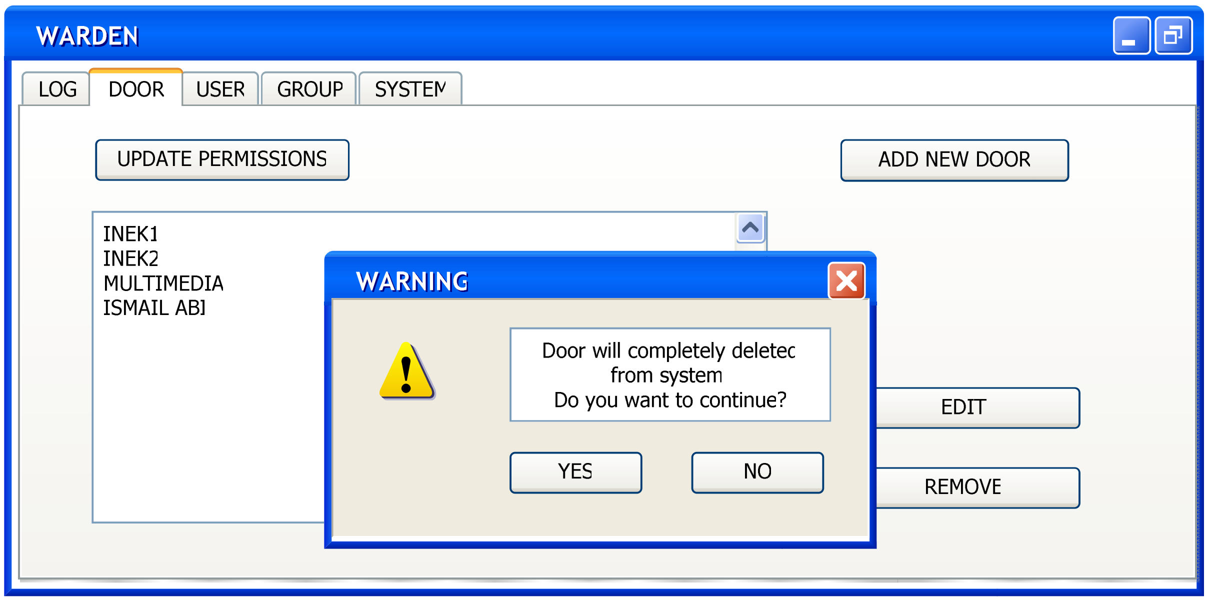


NAME: INEK1

IP: 144 122 113 2

This window will be opened when the "EDIT" button of the main window of the DOOR tab is pressed after a door is selected. The name and the IP of the selected door will be displayed. The user will be able to change these properties. IP changes are needed in case of an Ethernet card failure. If the "OK" button is pressed, the door will be updated, and if the "CANCEL" button is pressed, the operation will be cancelled and the window will be closed.

REMOVE DOOR

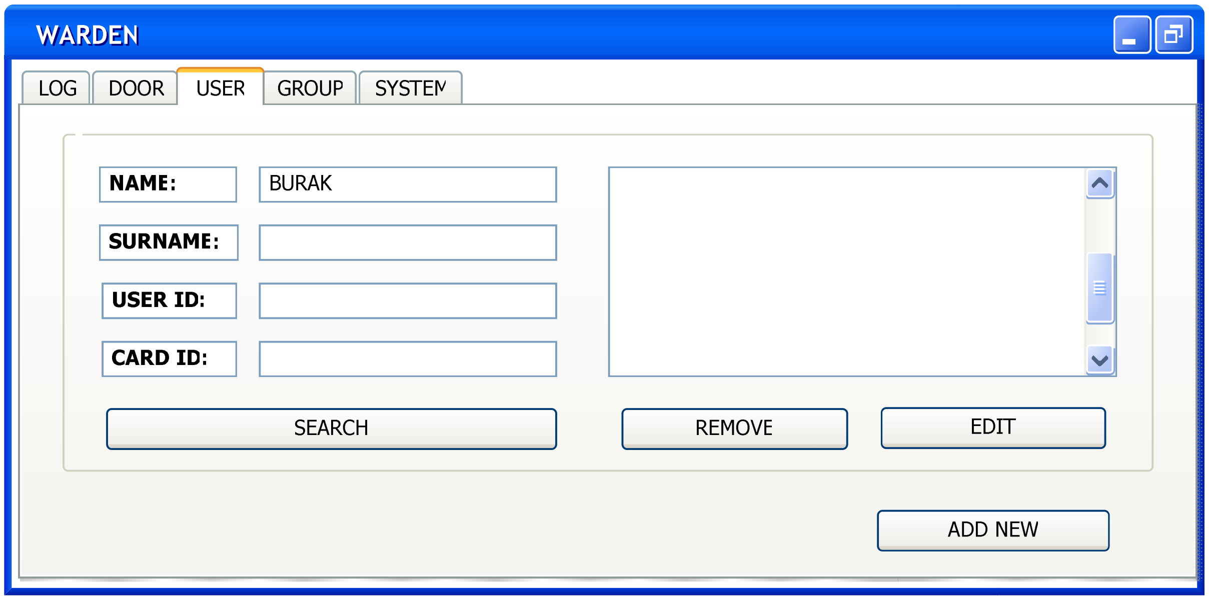


INEK1
INEK2
MULTIMEDIA
ISMAIL ABI

Door will completely deleted
from system
Do you want to continue?

This window will be opened when the "REMOVE" button of the main window of the DOOR tab is pressed after a door is selected. If the "YES" button is pressed, the door will be removed from the system, and if the "NO" button is pressed, the operation will be cancelled and the window will be closed.

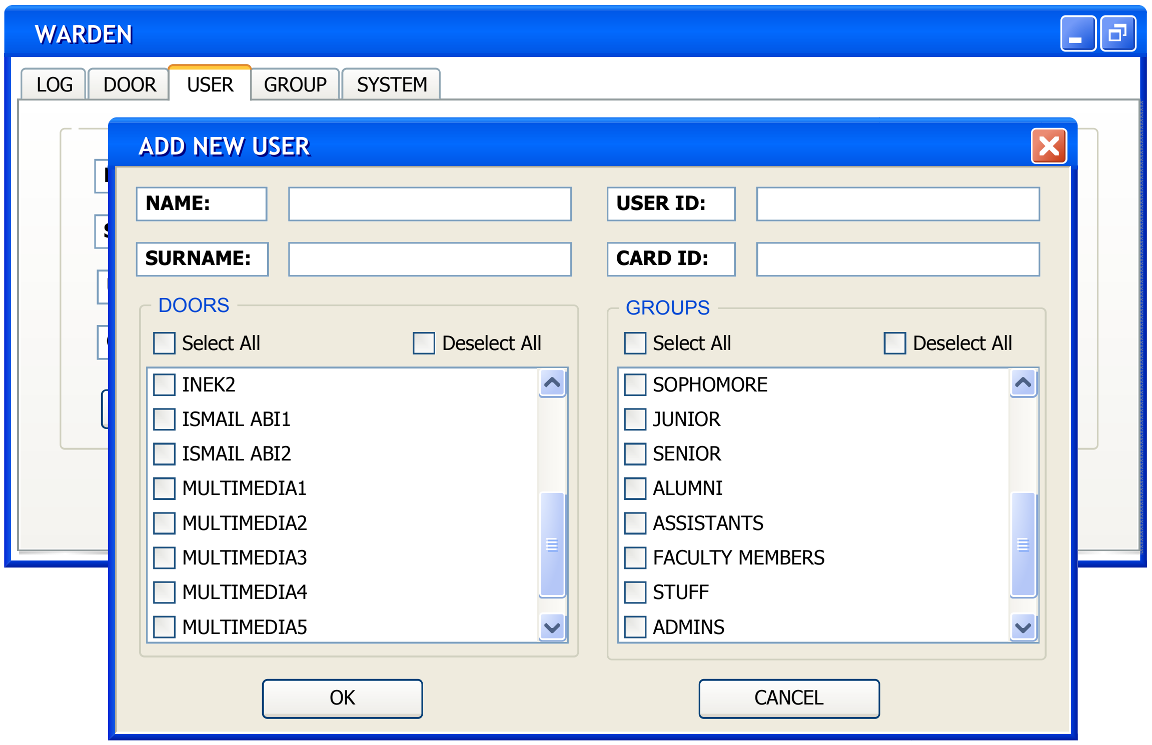
2.5.2.3. USER



NAME:	BURAK
SURNAME:	
USER ID:	
CARD ID:	

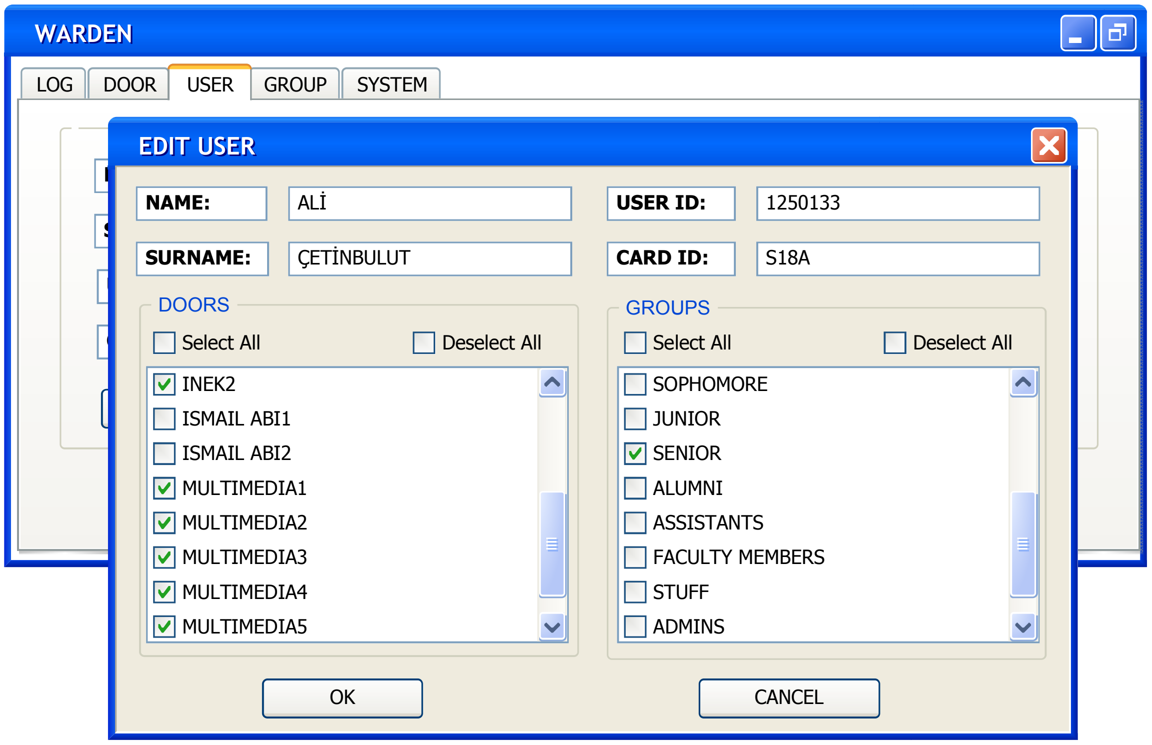
This is the main window of the USER tab of the GUI of the master computer. From this window, the users will be added, removed and edited. This window contains a small search engine. The users that have all the properties that are entered will be displayed when the "SEARCH" button is pressed. To edit a user, the "EDIT" button will be used. Firstly the user will be selected from the list and this button will be pressed. Similarly, to remove a user, the "REMOVE" button will be pressed. "ADD NEW" button will be used to add a new user to the system.

ADD NEW USER



This window will be opened when the "ADD NEW" button of the main window of the USER tab is pressed. Firstly the name, surname, user ID and the card ID of the new user will be entered. Secondly, the doors that the user will have permission to enter will be selected. Lastly, the groups of which the user is a member will be selected. If the "OK" button is pressed, the user will be added, and if the "CANCEL" button is pressed, the operation will be cancelled and the window will be closed.

EDIT USER



NAME:	ALİ	USER ID:	1250133
SURNAME:	ÇETİNBULUT	CARD ID:	S18A

DOORS

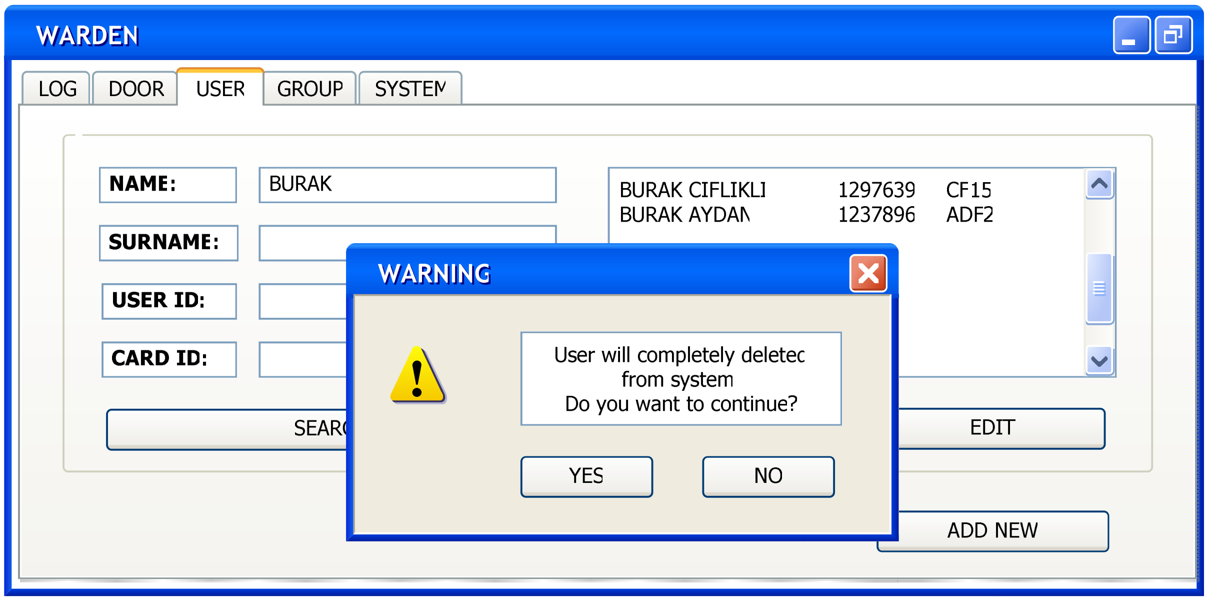
- [ ] Select All
- [ ] Deselect All
- [✓] INEK2
- [ ] ISMAIL ABI1
- [ ] ISMAIL ABI2
- [✓] MULTIMEDIA1
- [✓] MULTIMEDIA2
- [✓] MULTIMEDIA3
- [✓] MULTIMEDIA4
- [✓] MULTIMEDIA5

GROUPS

- [ ] Select All
- [ ] Deselect All
- [ ] SOPHOMORE
- [ ] JUNIOR
- [✓] SENIOR
- [ ] ALUMNI
- [ ] ASSISTANTS
- [ ] FACULTY MEMBERS
- [ ] STUFF
- [ ] ADMINS

This window will be opened when the "EDIT" button of the main window of the USER tab is pressed after a door is selected. The properties of the user will be displayed. These properties can be changed. If the "OK" button is pressed, the user properties will be updated, and if the "CANCEL" button is pressed, the operation will be cancelled and the window will be closed.

REMOVE USER



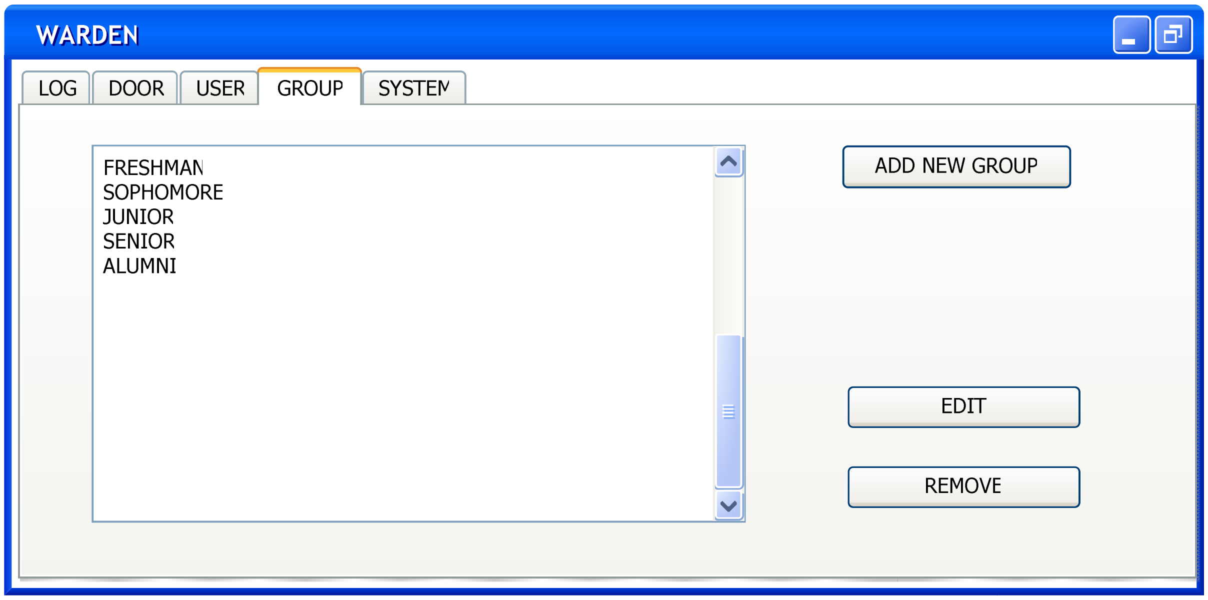
BURAK

BURAK CIFLIKLI	1297639	CF15
BURAK AYDAN	1237896	ADF2

User will completely deleted
from system
Do you want to continue?

This window will be opened when the "REMOVE" button of the main window of the USER tab is pressed after a user is selected. If the "YES" button is pressed, the user will be removed from the system, and if the "NO" button is pressed, the operation will be cancelled and the window will be closed.

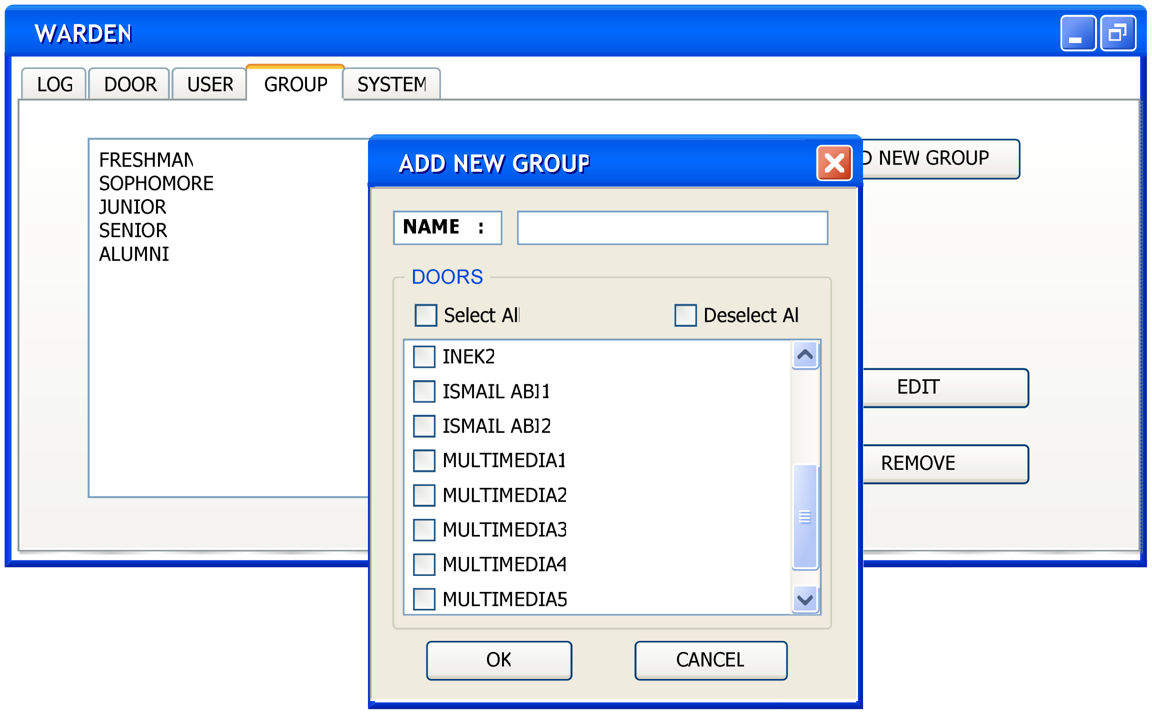
2.5.2.4. GROUP



FRESHMAN
SOPHOMORE
JUNIOR
SENIOR
ALUMNI

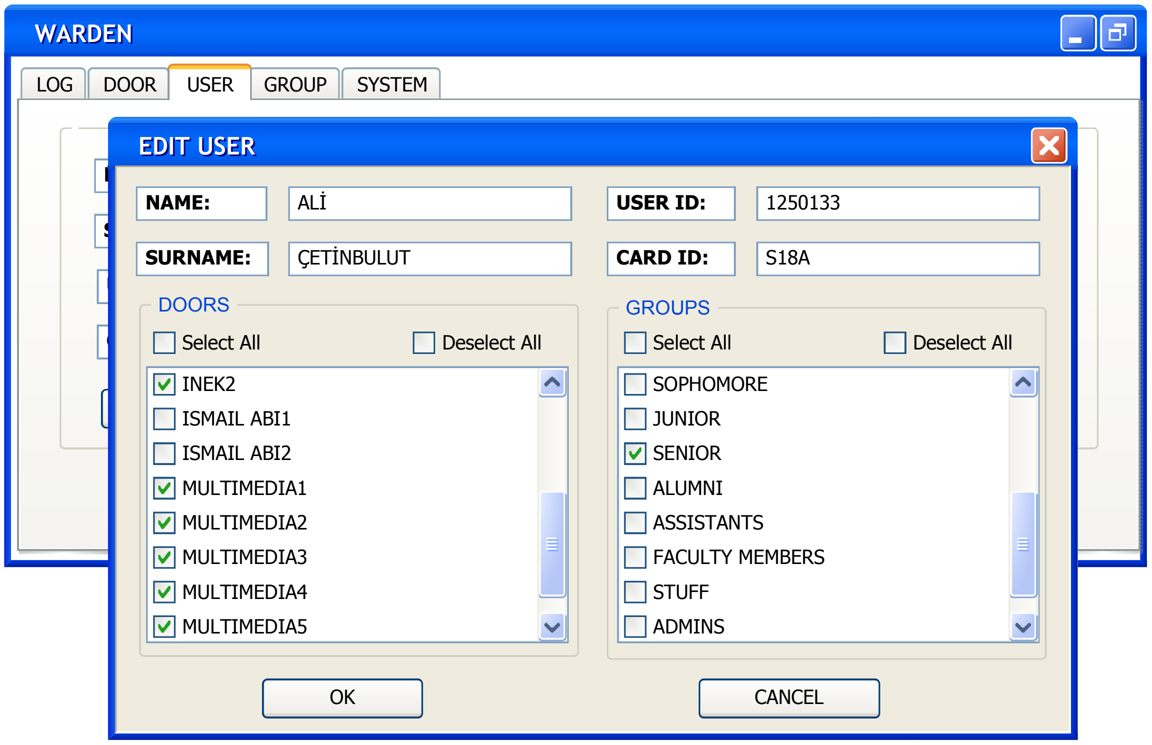
This is the main window of the GROUP tab of the GUI of the master computer. From this window, the user groups will be created, removed and edited. The “ADD NEW GROUP” button will be used to create a new user group. To edit a previously created group, the “EDIT” button will be used. Firstly a group from the list will be selected and this button will be pressed. Similarly, to remove a selected group, the “REMOVE” button will be pressed.

ADD NEW GROUP



This window will be opened when the “ADD NEW GROUP” button of the main window of the GROUP tab is pressed. The user will enter the name and the entrance permissions for each door of the group. If the “OK” button is pressed, the group will be created, and if the “CANCEL” button is pressed, the operation will be cancelled and the window will be closed.

EDIT GROUP



NAME:	ALİ	**USER ID:**	1250133
SURNAME:	ÇETİNBULUT	**CARD ID:**	S18A

DOORS

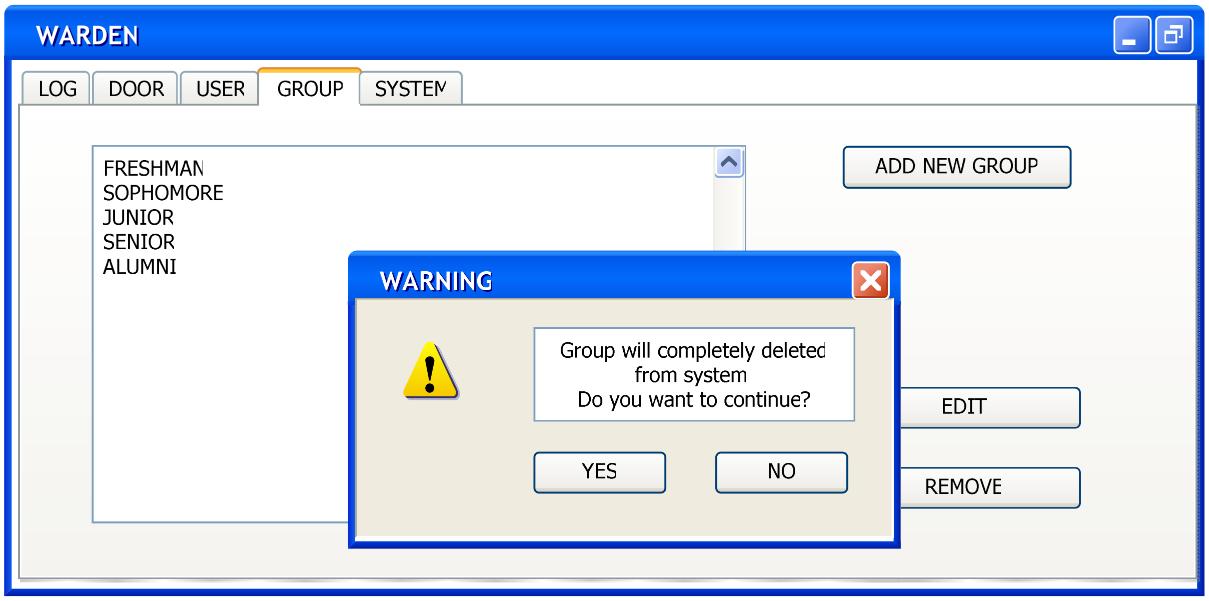
- [ ] Select All
- [ ] Deselect All
- [✓] INEK2
- [ ] ISMAIL ABI1
- [ ] ISMAIL ABI2
- [✓] MULTIMEDIA1
- [✓] MULTIMEDIA2
- [✓] MULTIMEDIA3
- [✓] MULTIMEDIA4
- [✓] MULTIMEDIA5

GROUPS

- [ ] Select All
- [ ] Deselect All
- [ ] SOPHOMORE
- [ ] JUNIOR
- [✓] SENIOR
- [ ] ALUMNI
- [ ] ASSISTANTS
- [ ] FACULTY MEMBERS
- [ ] STUFF
- [ ] ADMINS

This window will be opened when the "EDIT" button of the main window of the GROUP tab is pressed after a user group is selected. The name and the entrance permissions for each door of the selected group will be displayed. The user will be able to change these properties. If the "OK" button is pressed, the group will be updated, and if the "CANCEL" button is pressed, the operation will be cancelled and the window will be closed.

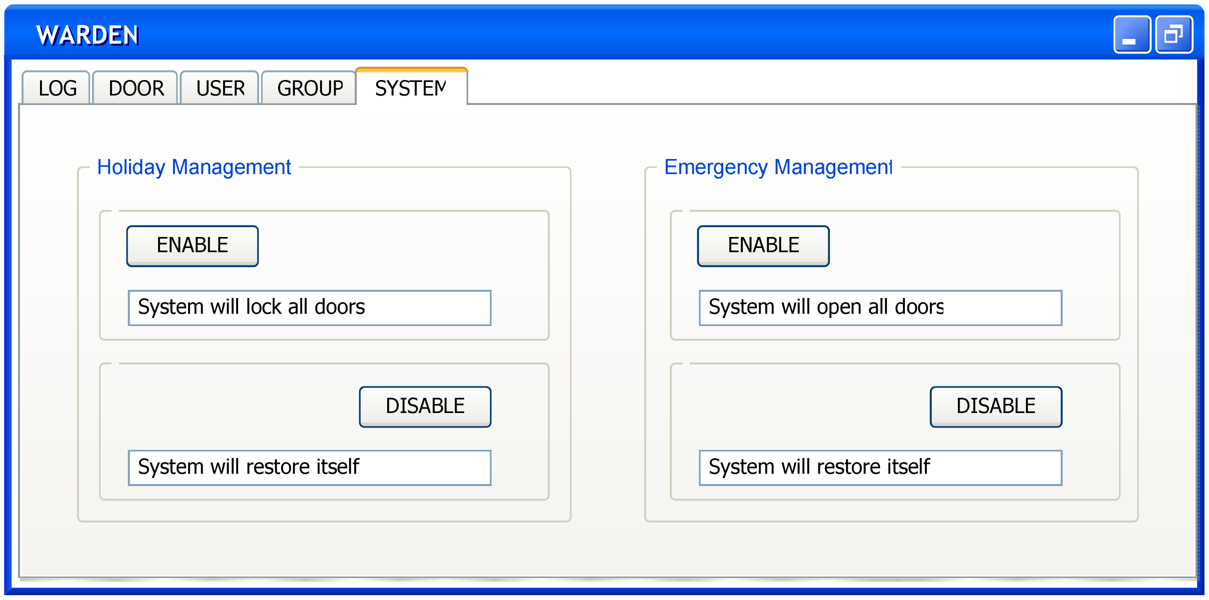
REMOVE GROUP



Group will completely deleted from system
Do you want to continue?

This window will be opened when the "REMOVE" button of the main window of the GROUP tab is pressed after a user group is selected. If the "YES" button is pressed, the group will be removed from the system, and if the "NO" button is pressed, the operation will be cancelled and the window will be closed.

2.5.2.5. SYSTEM



System will lock all doors

System will restore itself

System will open all doors

System will restore itself

We have two different controls for two different situations. One is for holidays. In these days nobody can use doors. We make this by sending empty card Id files to doors. The other situation is emergency, like fire or something similar. When you enable emergency management, all doors can be used by every user in system. We make this by sending all card Ids to all doors.

2.5.3. Network Class Design

Update Door Permission Files:

We will send permission files from master computer to single board. File data will be encoded for security. When “UPDATE PERMISSION” button is pressed, every permission file is decoded and sent to doors via WiFi.

Pseudo Code

```
fopen(doorNamesFile);
while(doorNamesFile != EOF)
{
        get door name and IP;
        concatenate required file name for permission file;
        fopen(doorPermisionFile);
        read all permission data;
        encode data;
        send encoded permission data to door IP;
        fclose(doorPermisionFile);
}
fclose(doorNamesFile);
```

Let All:

All card ID’s of the system will be send to doors so every user can use doors. After these files is created, they will be send by the pseudo code be above.

Deny All:

Empty files will be send to doors again by the pseudo code above.

Get Log Data:

Single board only sends log data to master computer. This transfer will be done when a card is read from card reader. Coming data will include card id, operation type, date, time and location. Location data will get from header. Since we store logs day by day, we will not write day to the log. All of our transmission via WiFi is encoded so when a log data is received first it is decoded and added to that day's log file. Not to loose data because of electricity cut or another problem, after writing some log data to file, file will be closed and opened.

Pseudo Code

```
Int iLoopCount;
fopen(todayLogFile);
While(true)
{
        if(a new log data comes)
        {
                decode data;
                generate log data;
                add to log file;
                show log to user;
                increment iLoopCount;
                if(time == 23:59)
                {
                        fclose(todayLogFile);
                        fopen(tomorrowLogFile);
                }
                if(iLoopCount == 100)
                {
                        fclose(todayLogFile);
                        fopen(todayLogFile);
                        iLoopCount = 0;
                }
        }
}
```

2.5.4. Database Design

There will be many data stored in WARDEN. When we start with the master computer; user information, user groups, doors, permissions and logs will be stored. In the previous phases of the project, usage of a file structure was stated; but after some more tests, a faster data design is formed, especially after the addition of a user search engine to the master computer.

2.5.4.1. ER-Diagram of Master Computer

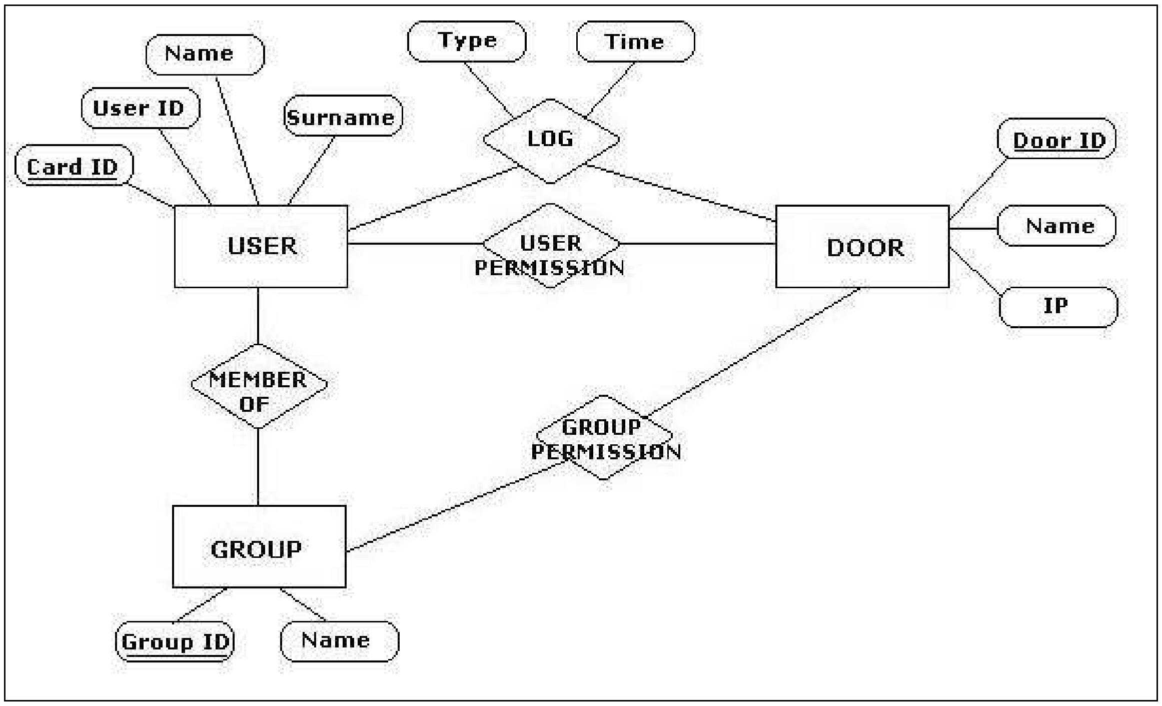


Figure 24 - ER Diagram

2.5.4.2. Entities

(1) USER: This entity will store the information of the system users. Its attributes are:

- Card ID
- User ID
- Name
- Surname

Card ID is the 16-digit number which is embedded into the card of the user and unique for every one. User ID is the identification number that will be given by the system admin. Student ID will be used in a university for instance. Name and surname of the user will also be stored.

(2) DOOR: This entity will store the information of the doors that are installed to the system. Its attributes are:

- Door ID
- Name
- IP

Door ID is a number which will be unique for every door. IP will be the IP address or the Ethernet card attached to the single board of the door. Name of the door will also be stored.

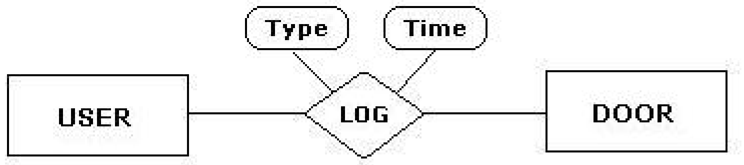
(3) GROUP: This entity will store the information of pre-defined user groups. Its attributes are:

- Group ID
- Name

Group ID is a number which will be unique for every group. Name of each group will also be stored.

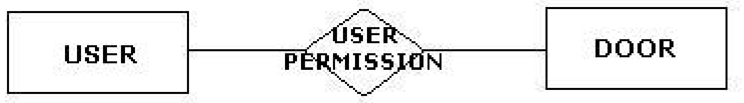
2.5.4.3. Relationships

(1) <u>LOG</u>



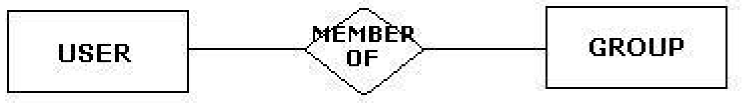
This relationship stores the logs. The actor of the event (user) is identified by his card and user IDs. The event location (door) is identified by the door ID. This relationship will have two additional attributes; type and time. Time will be the event occurrence time and type will be the event type like enter, exit or forced.

(2) <u>USER PERMISSION</u>



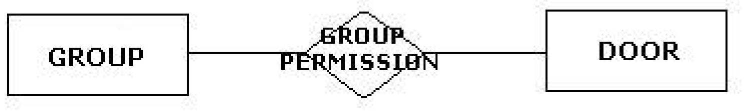
This relationship shows individual user entrance permissions by storing door and permitted user pairs. A user and a door as a pair mean that user has entrance permission for that door.

(3) <u>MEMBER OF</u>



WARDEN provides user-grouping and this relationship shows the members of the groups by storing the user and group pairs. A user and a group pair mean that user is a member of that group.

(4) GROUP PERMISSION



Groups will have entrance permissions which the members will directly get. The group permissions are stored within this relation by storing the group and door pairs. A group and a door as a pair mean that group has entrance permission for that door.

2.5.4.4. Database Tables of Master Computer

(1) USER

USER	
PK	**Card ID**
	User ID **Name** **Surname**

USER table will store the user information. Card ID will be primary key but the other columns; User ID, Name and Surname will also be required.

(2) GROUPS

GROUPS	
PK	**Group ID**
	Group Name

GROUPS table will store the group information. Group ID will be primary key but the other column; Group Name will also be required.

(3) GROUP

GROUP	
PK **PK**	**Group ID** **Card ID**

GROUP table will store the group members. Group ID and Card ID pair will be primary key. Group ID will be foreign key from GROUPS table and Card ID will be foreign key from USER table.

(4) DOOR

DOOR	
PK	**Door ID**
	Door Name **Door IP**

DOOR table will store the door information. Door ID will be primary key but the other columns; Door Name and Door IP will also be required.

(5) USER_PERMISSION

USER_PERMISSION	
PK **PK**	**Card ID** **Door ID**

USER_PERMISSION table will store the individual user entrance permissions. Card ID and Door ID pair will be primary key. Card ID will be foreign key from USER table and Door ID will be foreign key from DOOR table.

(6) GROUP_PERMISSION

GROUP PERMISSION	
PK **PK**	**Group ID** **Door ID**

GROUP_PERMISSION table will store the group entrance permissions. Group ID and Door ID pair will be primary key. Group ID will be foreign key from GROUPS table and Door ID will be foreign key from DOOR table.

(7) LOG

LOG	
PK	**LOG ID**
FK1 **FK2**	Card ID **Type** **Time** **Door ID**

LOG table will store the logs. LOG ID will be the primary key but the columns; User ID, Card ID, Type, Time and Door ID will also be required. Card ID will be foreign key from USER table and Door ID will be foreign key from DOOR table.

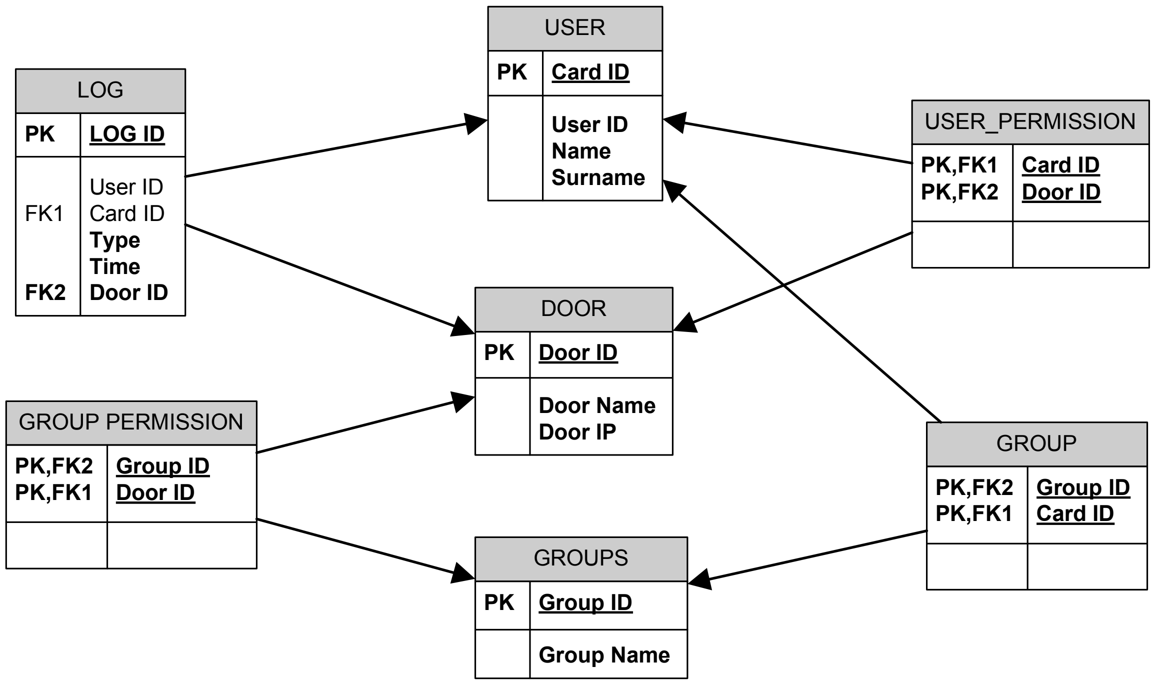


Figure 25 – Table Relations Diagram

2.5.4.5. SQL Statements of System

Remove User from Group

```
DELETE FROM GROUP
WHERE Card ID = GIVEN CARD ID
```

Update User

```
UPDATE USER
        SET Card ID = GIVEN CARD ID, User ID = GIVEN USER ID,
Name = GIVEN NAME, Surname = GIVEN SURNAME
WHERE Card ID = GIVEN CARD ID
```

Display Group List

```
SELECT Group Name
FROM GROUPS
WHERE
```

Add New Group to Group List

```
INSERT INTO GROUPS
VALUES (GIVEN GROUP ID, GIVEN GROUP NAME)
```

Add Group to Group Permissions List

```
INSERT INTO GROUP PERMISSION
VALUES (GIVEN GROUP ID, GIVEN DOOR ID)
```

Update Group

```
UPDATE GROUPS
        SET Group ID = GIVEN GROUP ID, Group Name = GIVEN GROUP
NAME
WHERE Group ID = GIVEN GROUP ID
```

Remove Group from Group List

```
DELETE FROM GROUPS
WHERE Group ID = GIVEN GROUP ID
```

Remove Group from Group Permission List

```
DELETE FROM GROUP PERMISSION
WHERE Group ID = GIVEN GROUP ID
```

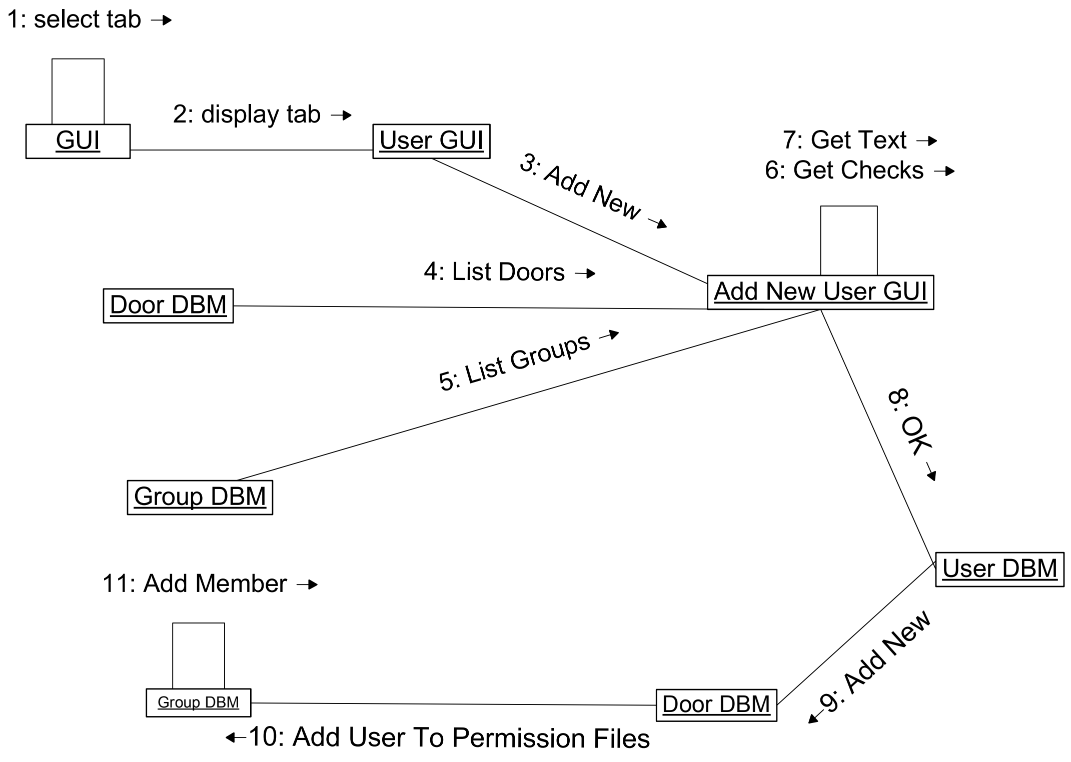
2.5.4.6. Upload files and Single Board

The entrance of a user will be authorized according to his permission. Each door will keep a local permission file. This will be faster than a permission check from the master computer. This local file will consist of the card IDs of the users that have entrance permission. The card IDs will be hashed.

These local files will be installed and uploaded from the master computer. The master computer will keep a permission file for each of the doors. These files will be exactly the same as the local ones. These files will be prepared just before the upload process. The USER_PERMISSION, GROUP and GROUP_PERMISSION tables will be joined and the card IDs of the users that have entrance permission will be saved.

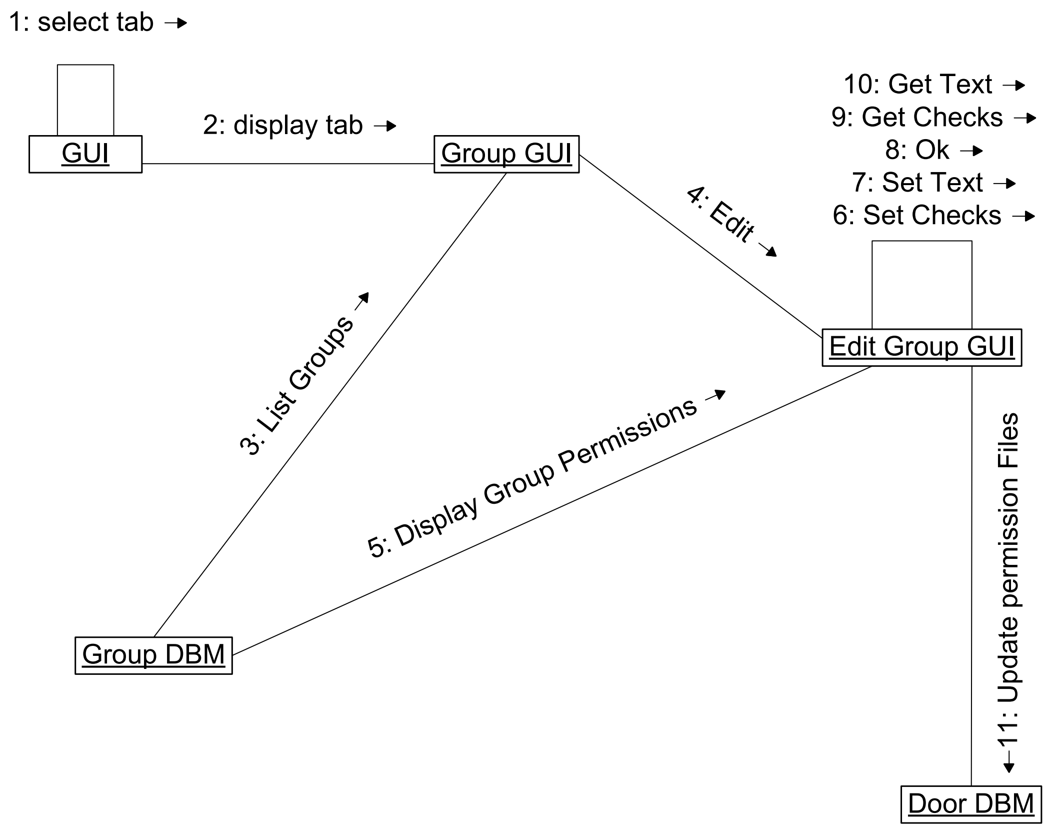
2.5.6. Collaboration Diagrams

2.5.6.1. Add User



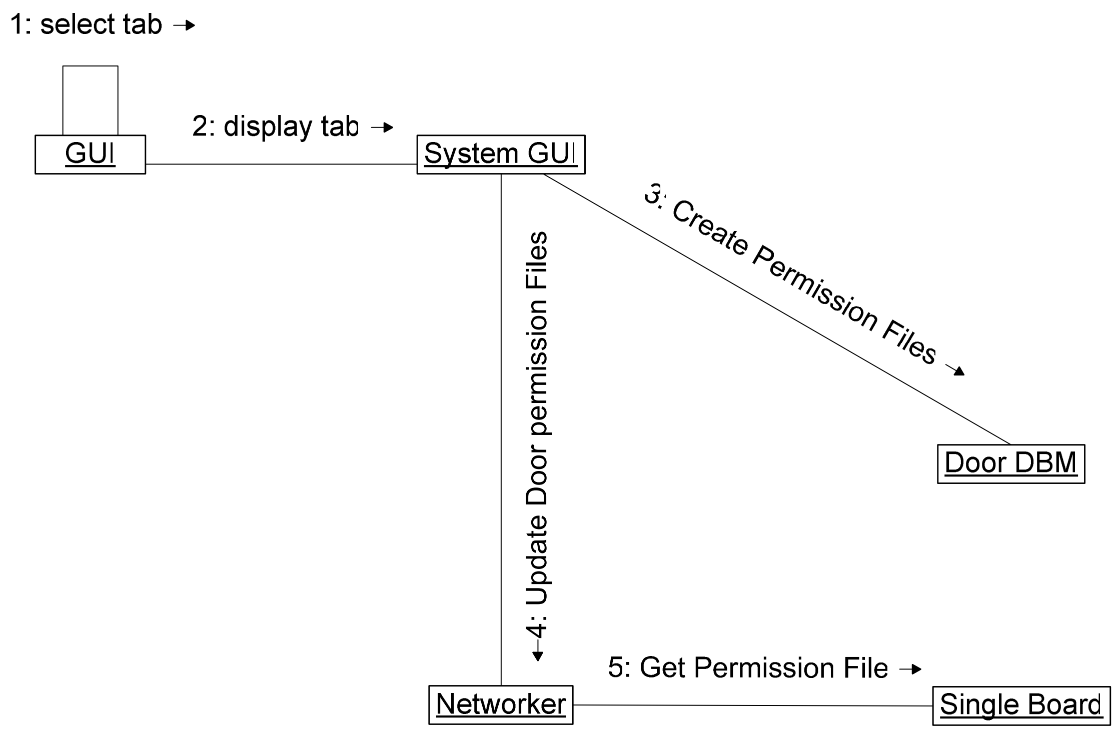
When the user selects the 'USER' tab of the master computer user interface, firstly the 'select tab' operation of the 'GUI' class is called. Then the 'display tab' operation is called to display the selected tab. If the user presses the 'Add New' button, the 'Add New' operation of 'User GUI' class is called and 'Add New User' GUI is displayed. Then the 'List Doors' and the 'List Groups' operations of the 'Door DBM' and 'Group DBM' classes are called to display the user the list of previously installed doors and previously formed groups. When the user presses the 'OK' button, 'OK', 'Get Text' and 'Get Checks' operations are called to get the user properties. Lastly, 'Add New' operation of 'User DBM', 'Add User to Door Permission List' operation of 'Door DBM' and 'Add Member' operation of 'Group DBM' are called to add the user, assign the user permissions and group memberships.

2.5.6.2. Edit Group



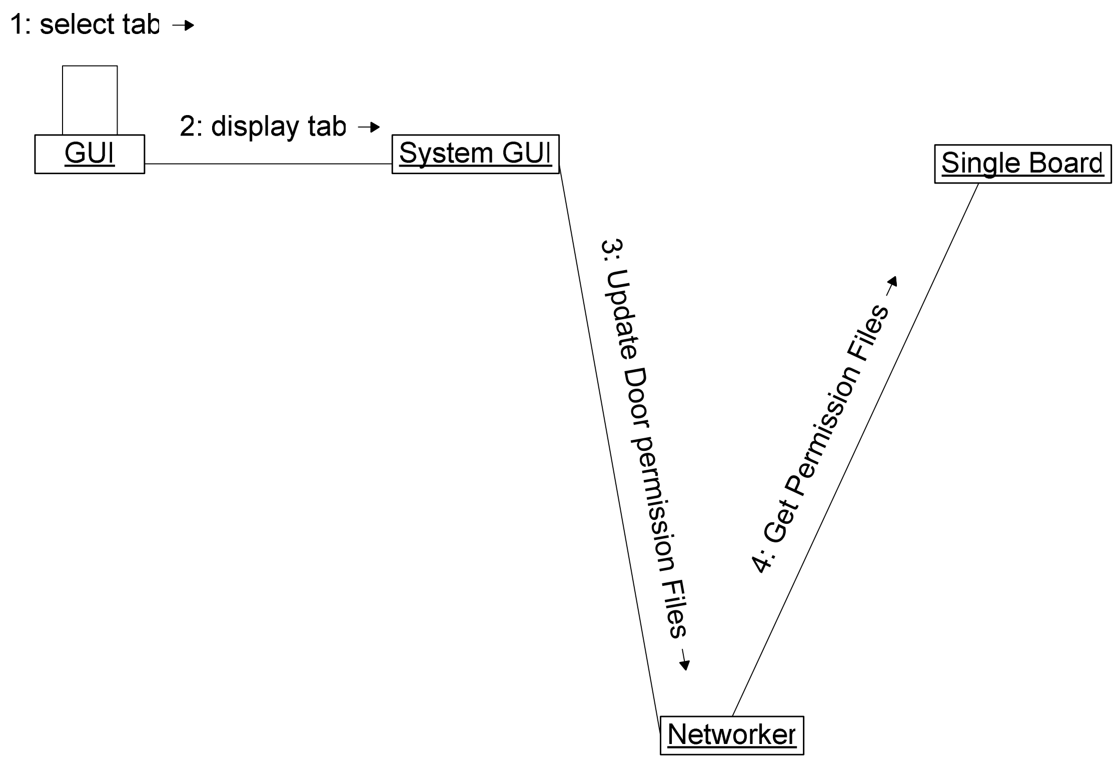
When the user selects the 'GROUP' tab of the master computer user interface, firstly the 'select tab' operation of the 'GUI' class is called. Then the 'display tab' operation is called to display the selected tab. 'List Groups' operation of the 'Data Manager' class is called to display the list of previously formed groups. If the user presses the 'Edit' button after selecting a group, 'Edit Group' GUI is displayed. 'Display Group Permissions' operation of the 'Data Manager' class is called to get the group permissions over the doors from file structure. Then 'Set Checks' and 'Set Text' operations of the 'Edit Group GUI' class are called to display group name and permissions. When the user presses the 'OK' button, 'OK', 'Get Text' and 'Get Checks' operations are called to get the group properties. Lastly, 'Update Door Permission Files' operation of the 'Data Manager' class is called to update group and user permissions.

2.5.6.3. Enable Holiday



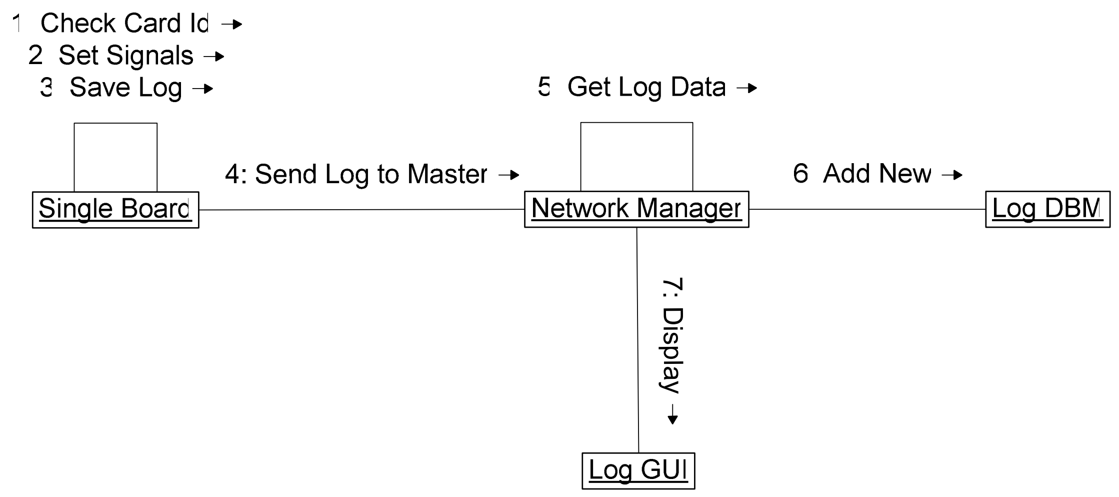
When the user selects the 'SYSTEM' tab of the master computer user interface, firstly the 'select tab' operation of the 'GUI' class is called. Then the 'display tab' operation is called to display the selected tab. If the user presses the 'Enable' button of 'Holiday Management', 'Enable Holiday' operation of the 'System GUI' class and then this operation calls the 'Deny All' operation of the 'Data Manager' class is called to update user permissions to deny every entrance.

2.5.6.4. Update Permission Files



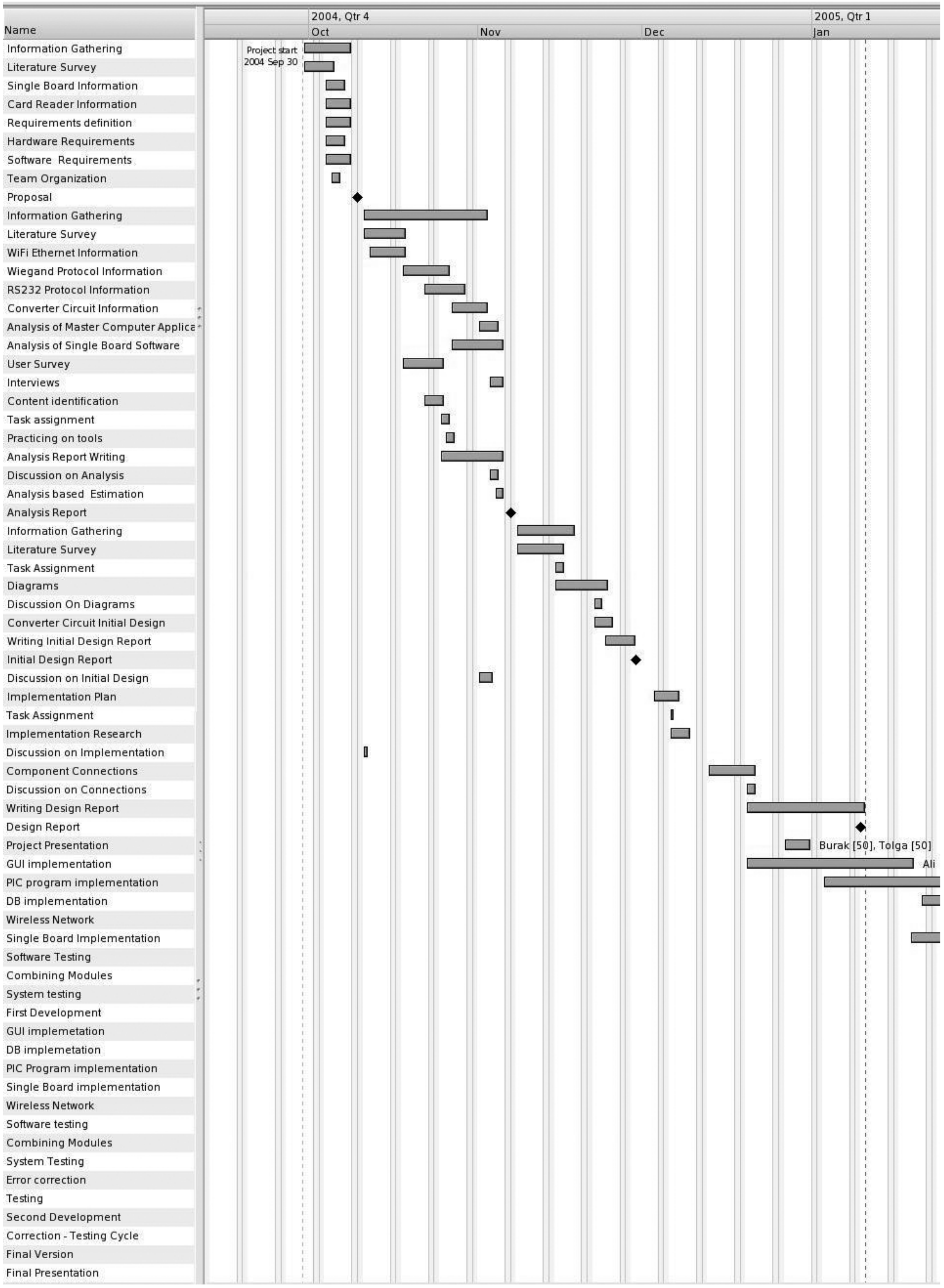
When the user selects the 'DOOR' tab of the master computer user interface, firstly the 'select tab' operation of the 'GUI' class is called. Then the 'display tab' operation is called to display the selected tab. If the user presses the 'update permissions' button, the files that will be uploaded to the doors are updated by the 'Update Door permission Files' operation of the 'Network Manager' class. Finally, 'Get Permission Files' operation of the single board is called to finalize the update process.

2.5.6.5. Show ID Card



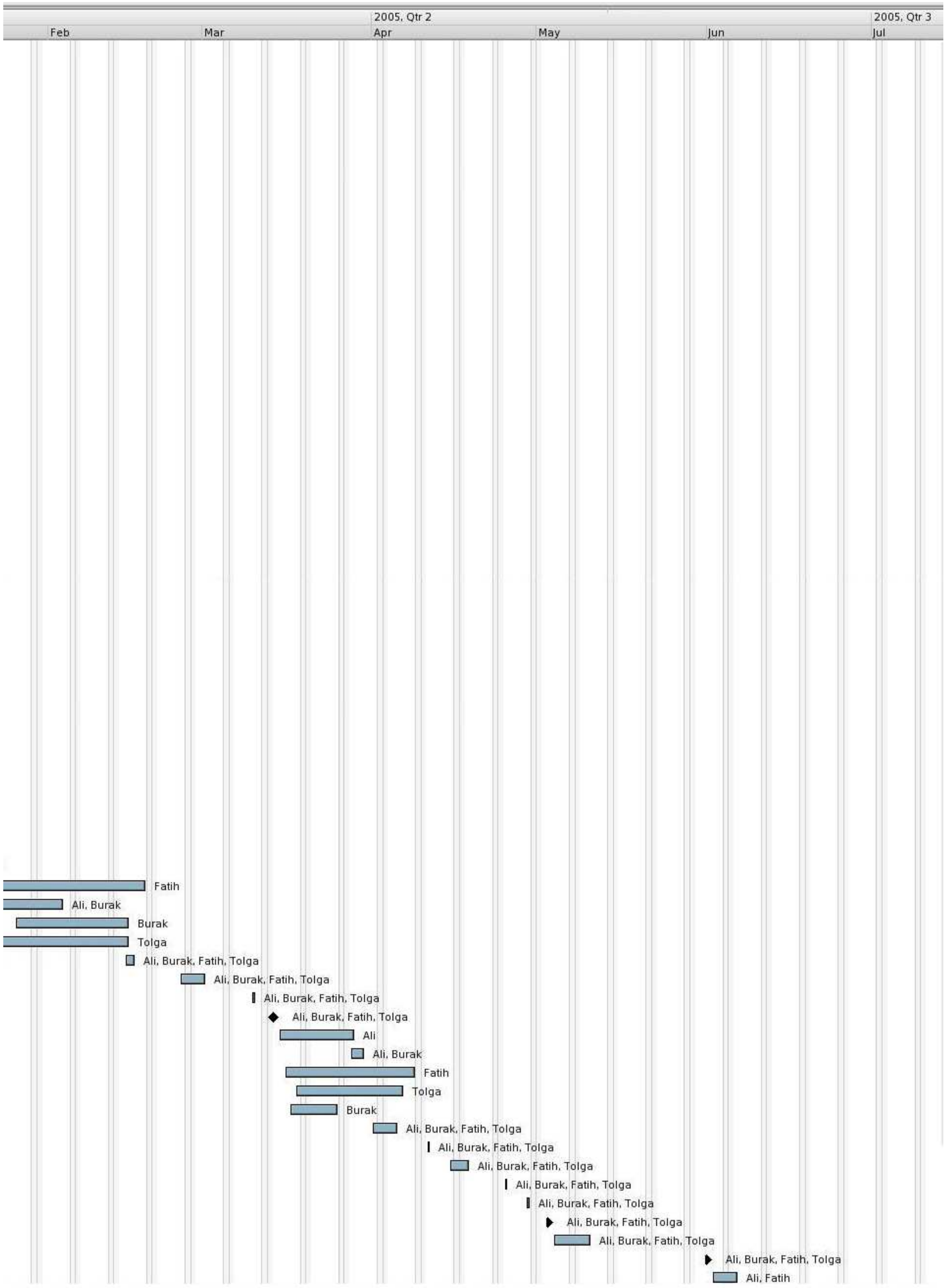
When the user gets his card read to the reader, firstly 'Check Card id' operation of the 'Single Board' class is called. Then the 'Set Signals' operation is called to set the proper signals according to the result of the check. Then the 'Save log' operation is called to locally save the log of the event. Finally, 'Send Log to Master' operation is called to finalize the single board part of the process. The 'Get Log Data' operation of the 'Network Manager' class is called to load the LOG to the master computer. New LOG is added to the database by calling the 'Add new' operation of the 'Log DBM' class and finally the 'Display' operation of the 'Log GUI' class is called to display the log.

3. GANNT CHART



Name

- Information Gathering
- Literature Survey
- Single Board Information
- Card Reader Information
- Requirements definition
- Hardware Requirements
- Software Requirements
- Team Organization
- Proposal
- Information Gathering
- Literature Survey
- WiFi Ethernet Information
- Wiegand Protocol Information
- RS232 Protocol Information
- Converter Circuit Information
- Analysis of Master Computer Applica
- Analysis of Single Board Software
- User Survey
- Interviews
- Content identification
- Task assignment
- Practicing on tools
- Analysis Report Writing
- Discussion on Analysis
- Analysis based Estimation
- Analysis Report
- Information Gathering
- Literature Survey
- Task Assignment
- Diagrams
- Discussion On Diagrams
- Converter Circuit Initial Design
- Writing Initial Design Report
- Initial Design Report
- Discussion on Initial Design
- Implementation Plan
- Task Assignment
- Implementation Research
- Discussion on Implementation
- Component Connections
- Discussion on Connections
- Writing Design Report
- Design Report
- Project Presentation
- GUI implementation
- PIC program implementation
- DB implementation
- Wireless Network
- Single Board Implementation
- Software Testing
- Combining Modules
- System testing
- First Development
- GUI implemetation
- DB implemetation
- PIC Program implementation
- Single Board implementation
- Wireless Network
- Software testing
- Combining Modules
- System Testing
- Error correction
- Testing
- Second Development
- Correction - Testing Cycle
- Final Version
- Final Presentation


2005, Qtr 2
2005, Qtr 3
Feb
Mar
Apr
May
Jun
Jul
Fatih
Ali, Burak
Burak
Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali
Ali, Burak
Fatih
Tolga
Burak
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Burak, Fatih, Tolga
Ali, Fatih