CHAPTER 3

METHODOLOGY

3.1 Analysis Phase

The initial phase of this project includes background analysis or research. The primary purpose of research is to become familiar with the work to be done on related topics with a focus on both hardware and software to be used in the project and how to use a particular this software to keep the database part. Once the components are identified, then the data of each will collected for further studies and research.

3.2 Design Phase

There is three parts of it that needs to be assembled and work together as a system. Part 1 is WiDSTAC part (please refer to figure 1.1) Part 2 is about circuit for receiving or transmitting from/to PC part. (please refer to figure 1.2) Part 3 is about the created database (Microsoft Access), Visual Basic to act like interface to users and 8051 programming. Thus, I have broken down the design phase in to hardware design phase and software design phase.

3.2.1 Hardware Design

I need to have two different PCB board that contained circuit for WiDSTAC and Device for Receiving or Transmitting from/to PC.

3.2.1.1 WiDSTAC

WiDSTAC is combination of Microcontroller AT89S52, RF Module transmitter TX-F9912(315Mhz), RF Module receiver RX-PCR1A(315Mhx), LCD, keypad, transistor TIP31C, transistor BC547 and antenna.

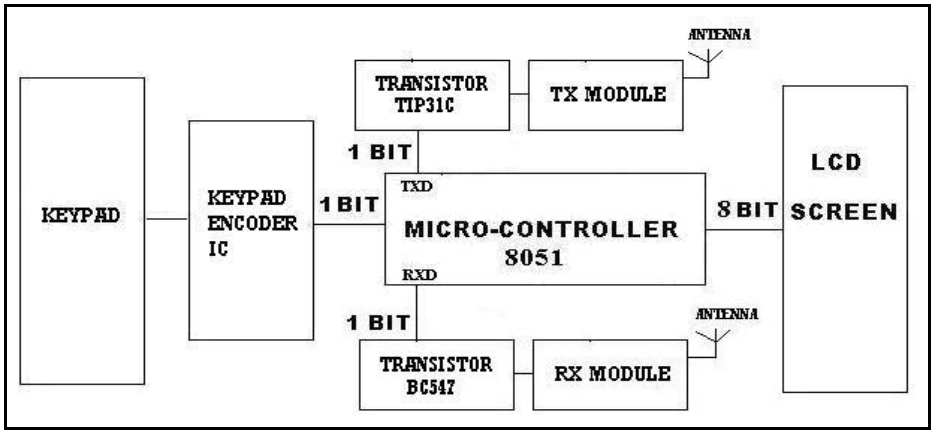

Figure 3.0: Early design of WiDSTAC

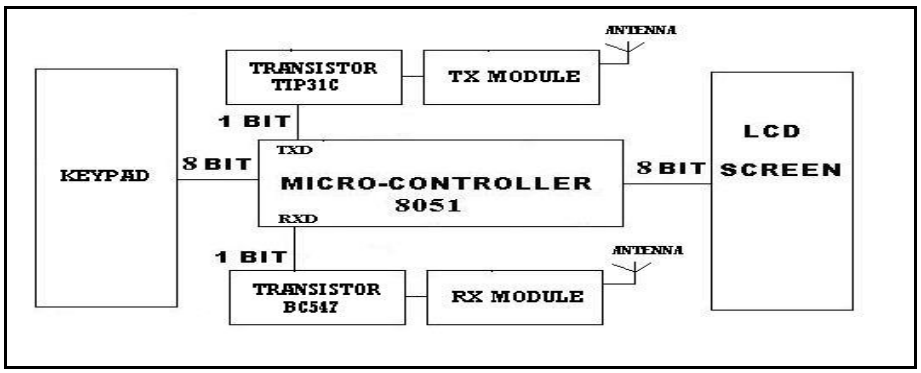

Figure 3.1: Final design of WiDSTAC

In the final design of WiDSTAC the keypad encoder IC is stripped off because there is method call keypad scanning in 8051 which can connect keypad directly to microcontroller.This method only use 7 I/O pin of microcontroller. Please refer figure 3.17 and figure 3.35 for more detail about keypad scanning method.

Table 3.0: Components List of WiDSTAC

No	Components		Quantity
1	Microcontroller AT89S52		1
2	RF Module transmitter - TX-F9912(315Mhz)		1
3	RF Module receiver - RX-PCR1A(315Mhx)		1
4	LCD 162A		1
5	Keypad - 3 x 4 buttons		1
6	Crystal 11.0592 MHz		1
7	Power regulator		1
8	Switch		2
9	Connector 6-pin		1
10	LED		1
11	Antenna		1
12	PCB board		1
13	Battery 9v		1
14	Transistor	TIP31C	1
		BC547	1
15	Resistor	330	1
		1k	10
		4.7k	2
		8.2k	1
16	capacitor	10uF	1
		33pF	2

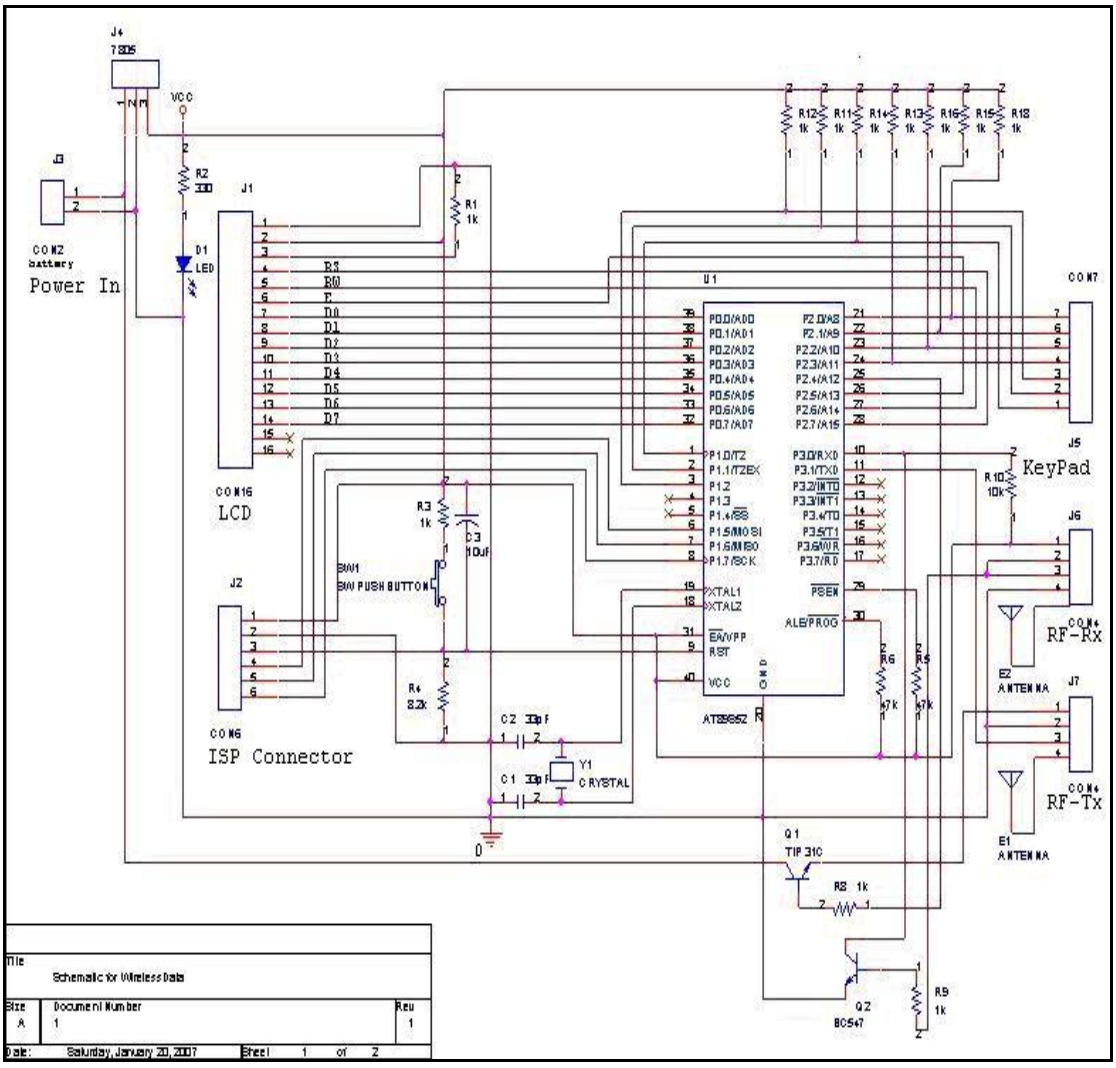

Figure 3.3: Schematic circuit of WiDSTAC

3.2.1.2 Device for Receiving or Transmitting from/to PC

Table 3.1: Components List of device for Receiving or Transmitting from/to PC

No	Components		Quantity
1	MAX232		1
2	RF Module transmitter - TX-F9912(315Mhz)		1
3	RF Module receiver - RX-PCR1A(315Mhx)		1
4	DE 9 female connector		1
5	Transistor BC547		1
6	Power regulator		1
7	Capacitor 10uF		4
8	LED		1
9	Antenna		1
10	PCB board		1
11	Battery 9v		1
12	Resistor	330	1
		1k	1
		10k	1

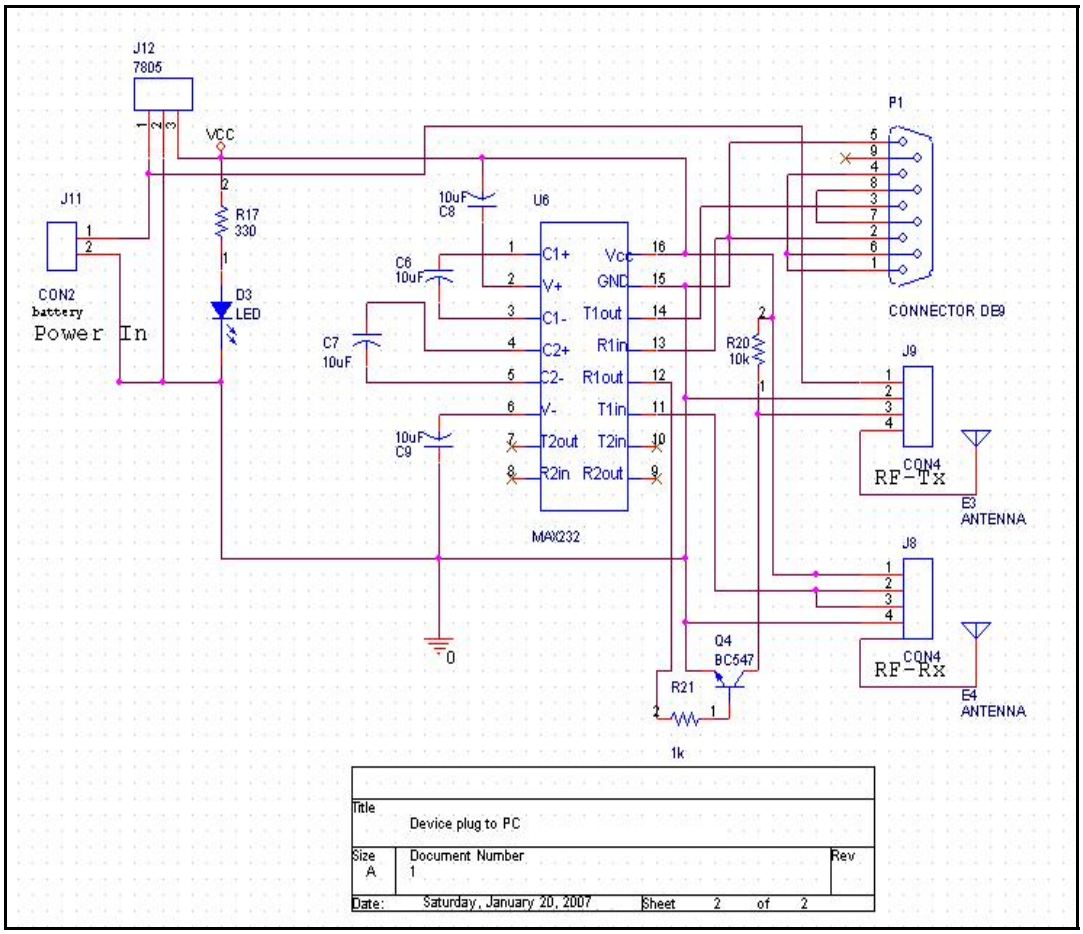

Figure 3.3: Schematic circuit of device for Receiving or Transmitting from/to PC

3.2.2 Software Design

The software design can divided to 3 parts as software design on 8051 program code, Visual Basic and Microsoft Access. 8051 program code as act like controller to handle the whole system key-in marks. Visual Basic act like middle people of 8051 program code and Microsoft Access when the data from WiDSTAC want save it to Microsoft Access as database or vice versa.

3.2.2.1 8051 Program Code (please refer Appendix A: 8051 program code)

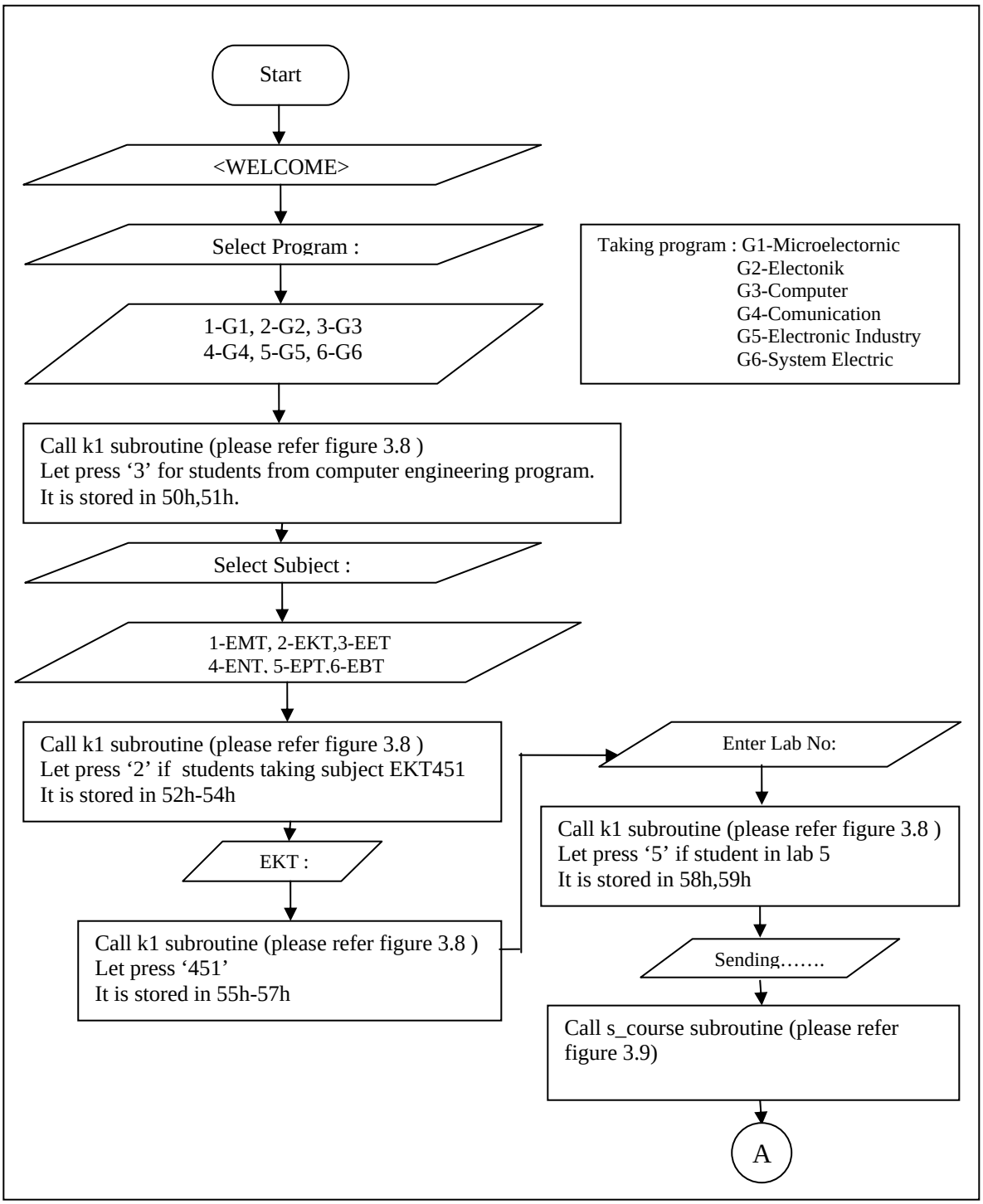

Figure 3.4: Part 1 of flowchart for 8051 program code

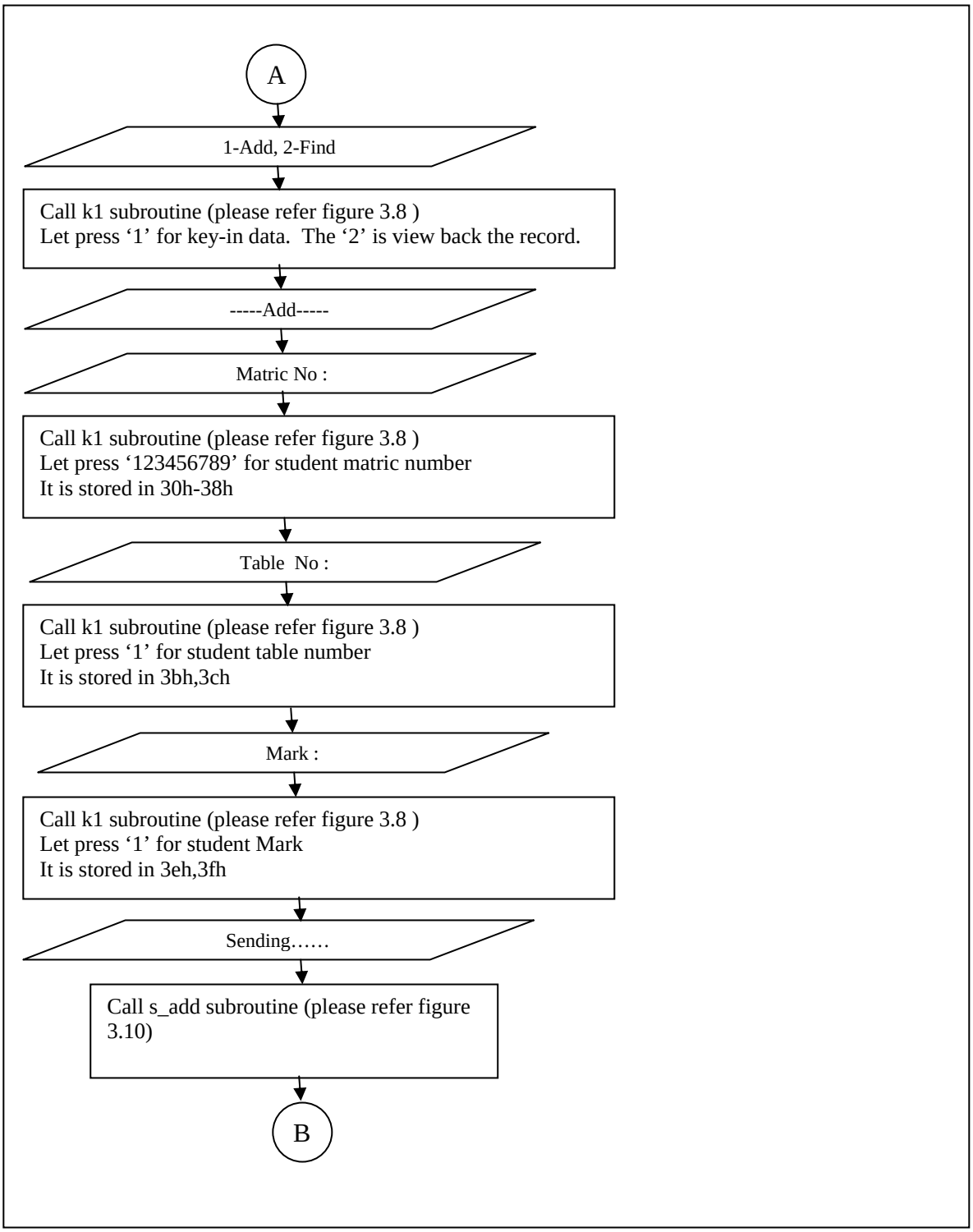

Figure 3.5: Part 2 of flowchart for 8051 program code

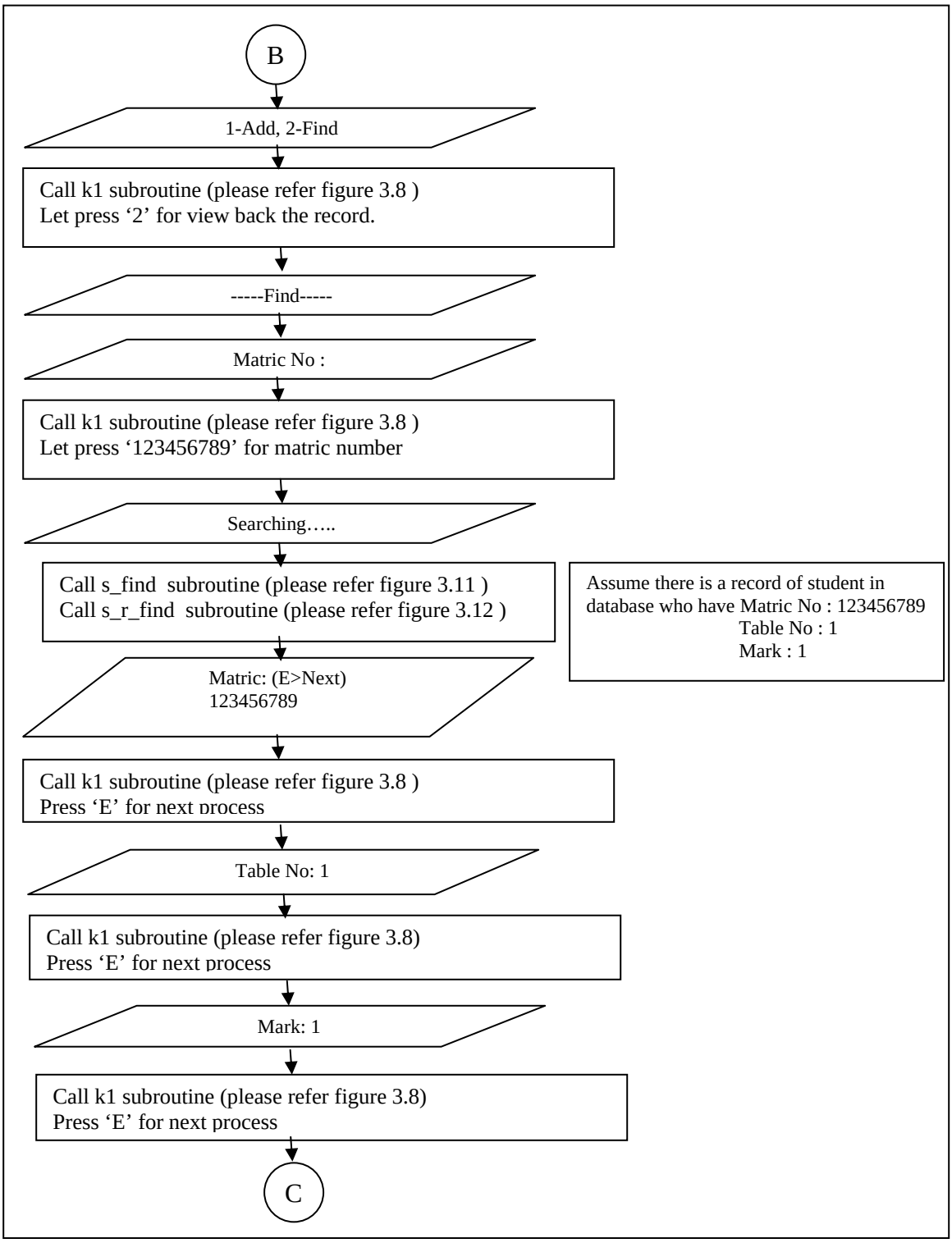

Figure 3.6: Part 3 of flowchart for 8051 program code

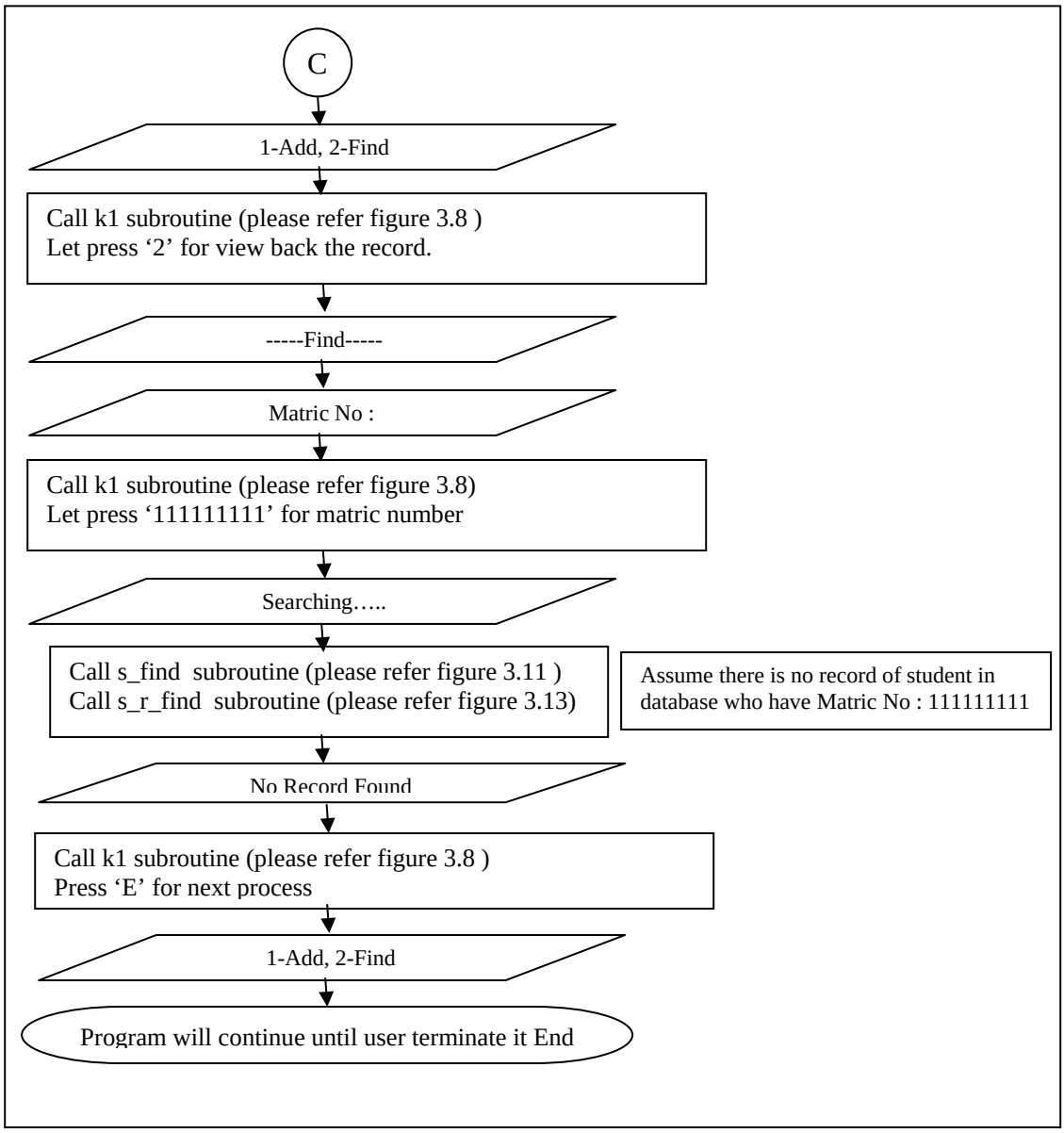

Figure 3.7: Part 4 of flowchart for 8051 program code

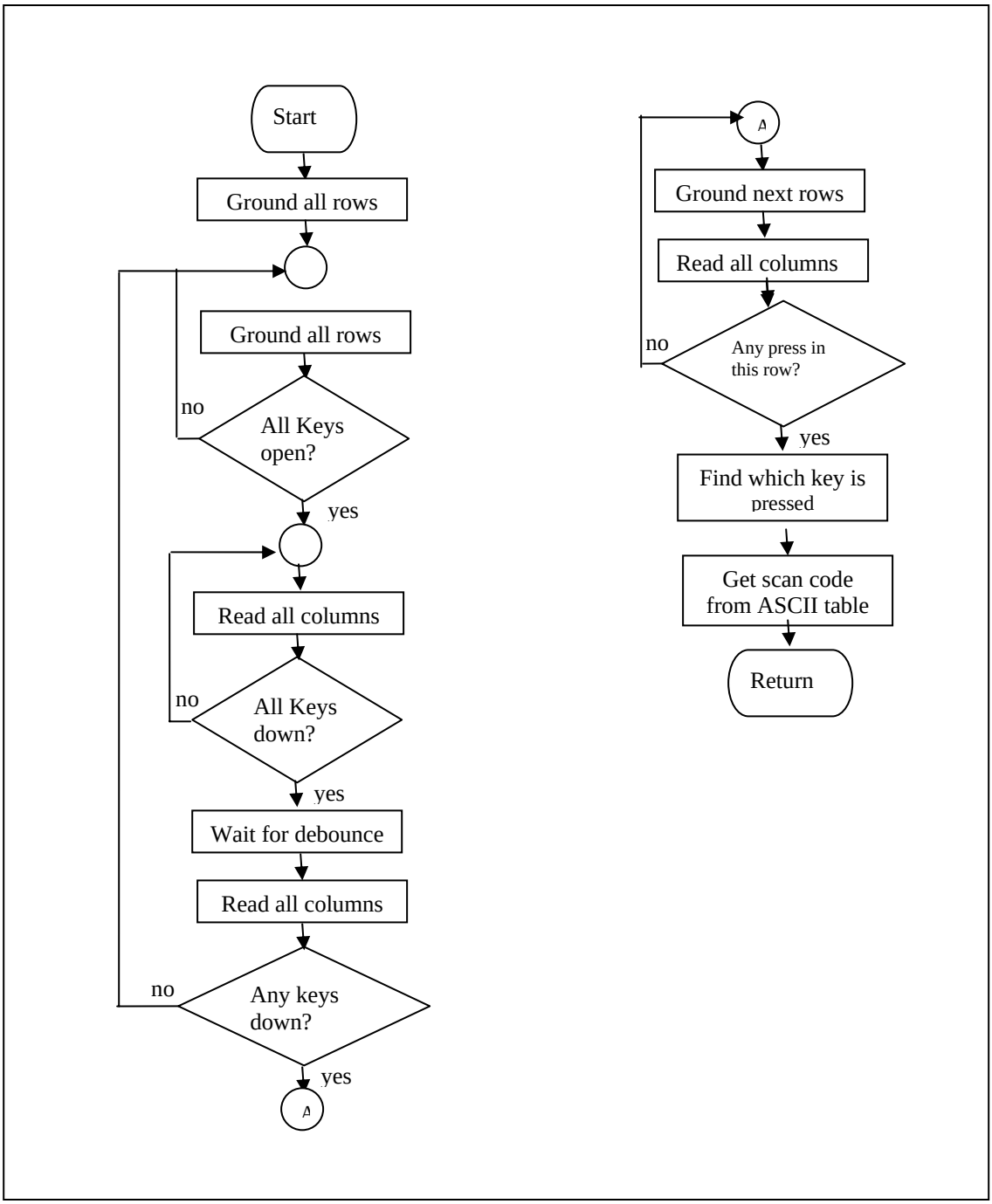

Figure 3.8: K1(keypad scanning) flowchart

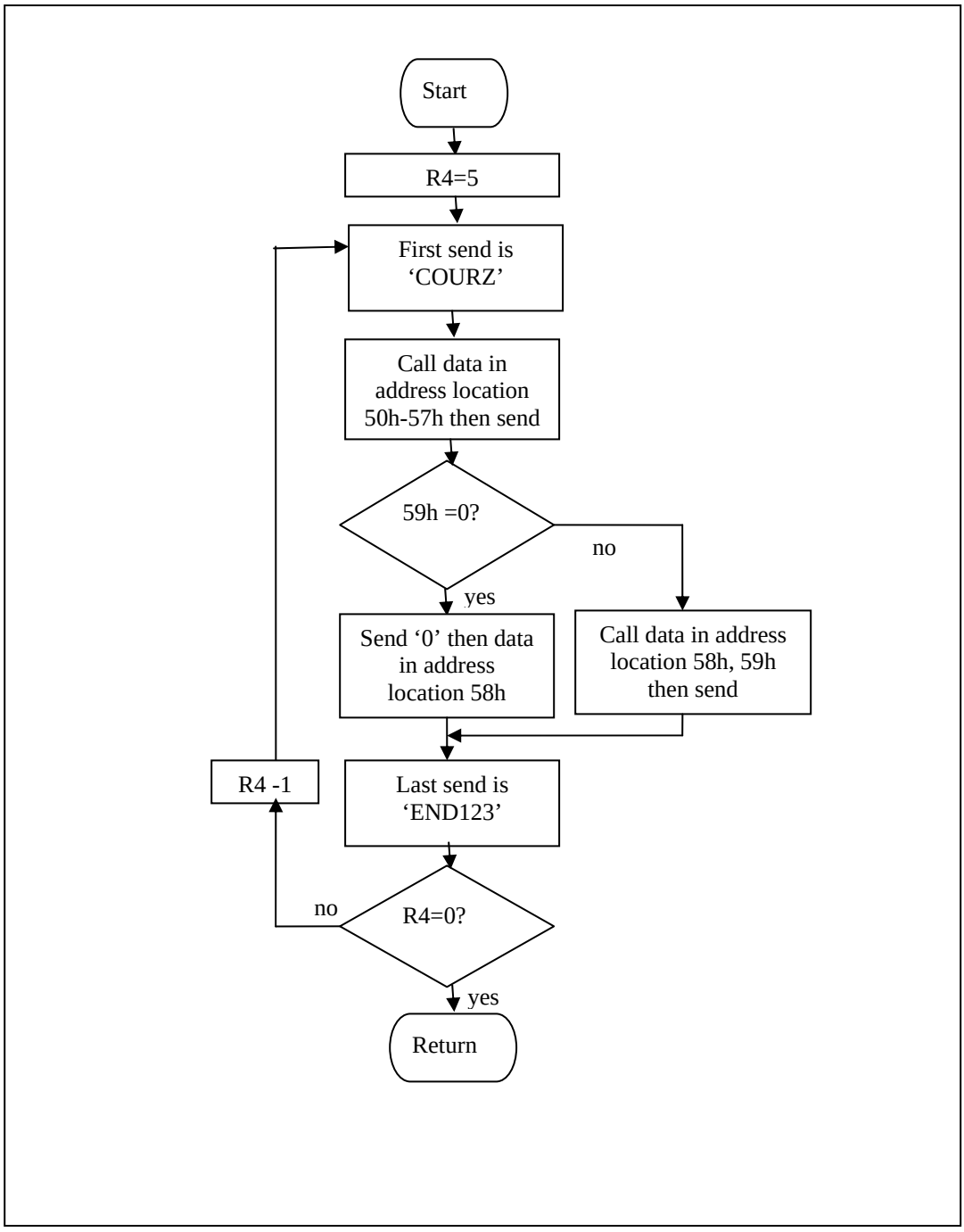

Figure 3.9: s_course flowchart

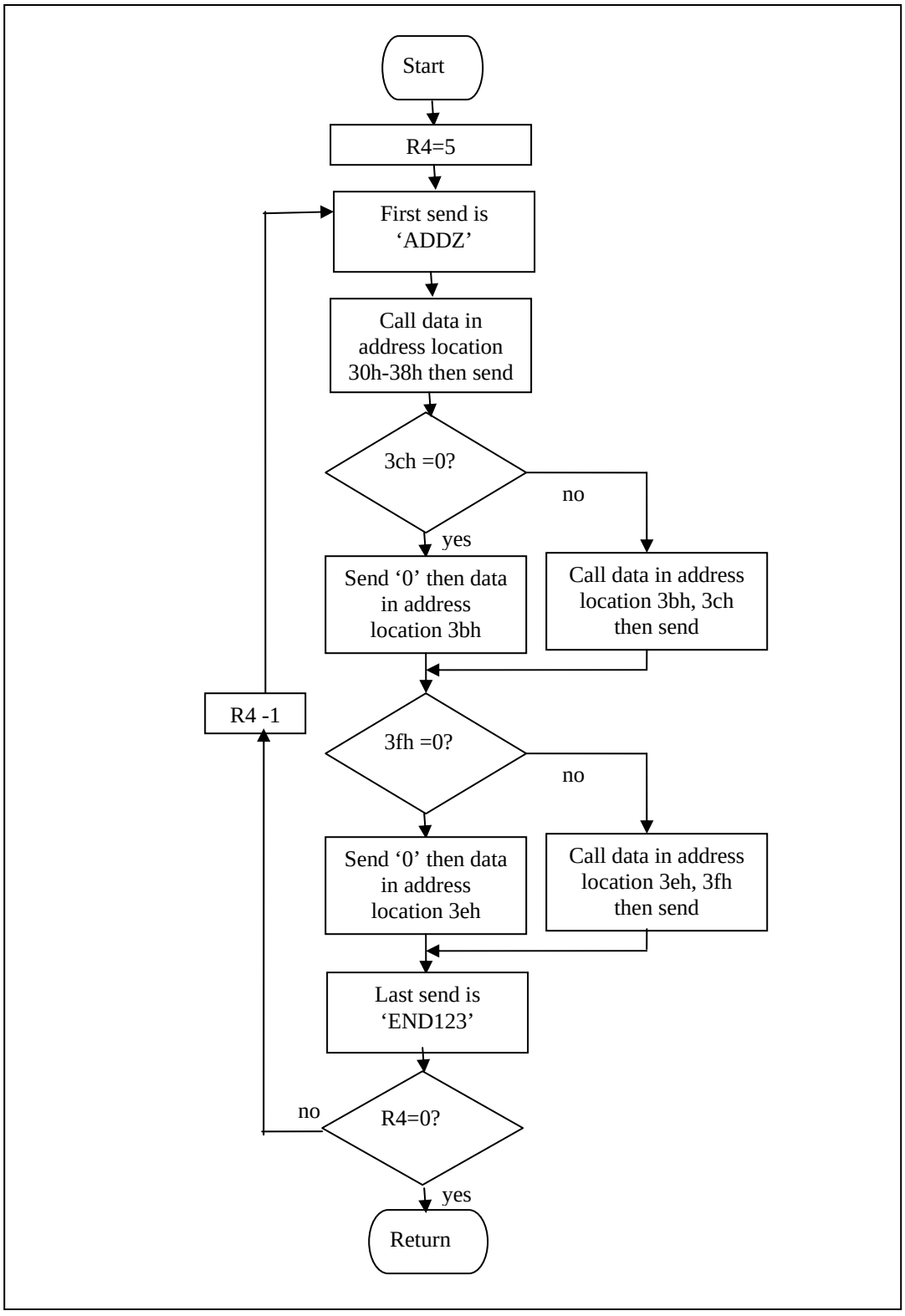

Figure 3.10: s_add flowchart

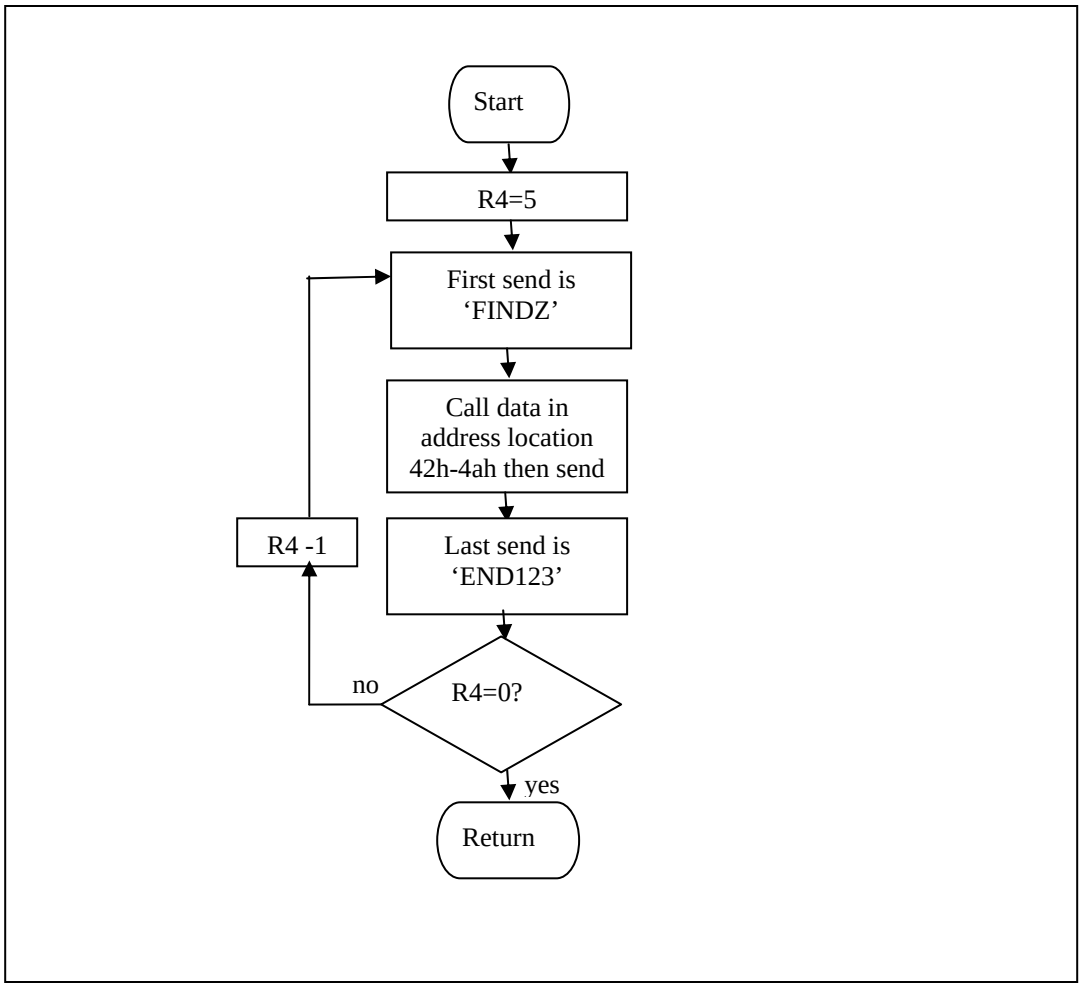

Figure 3.11: s_find flowchart

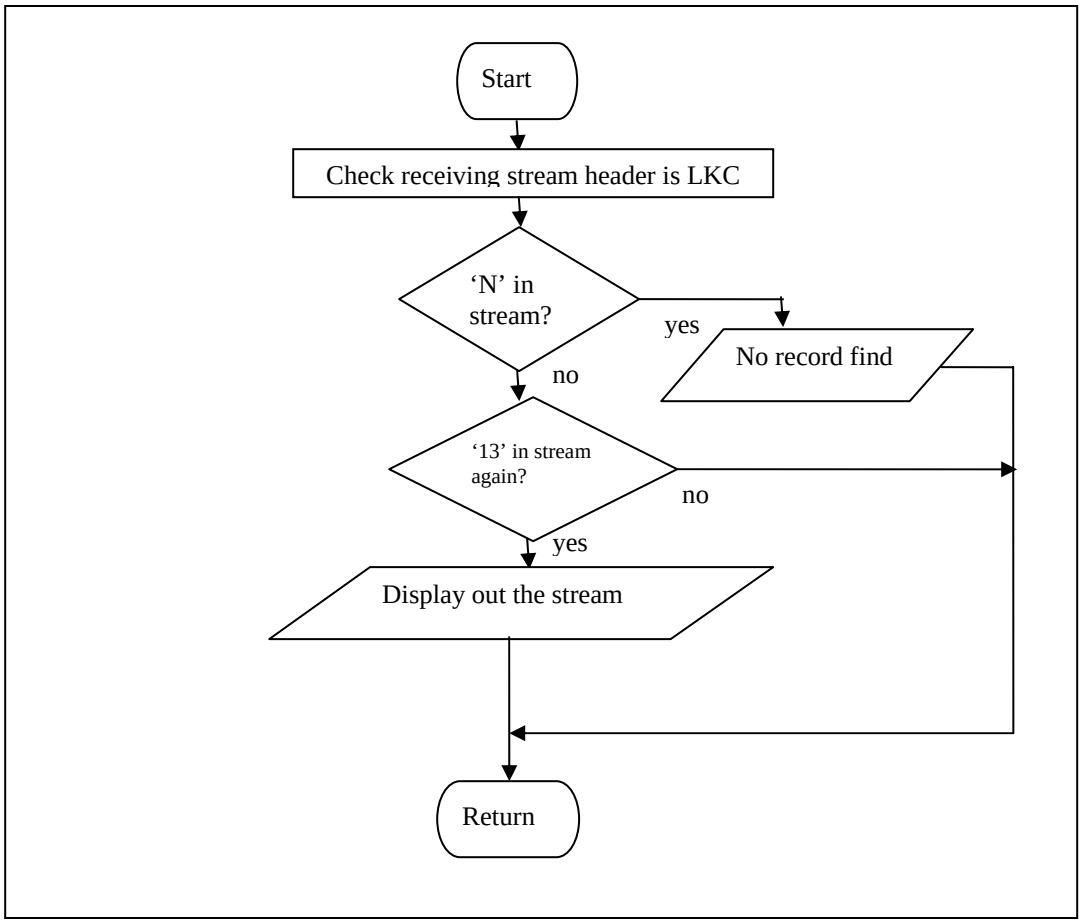

Figure 3.12: s_r_find flowchart

3.2.2.2 Visual Basic Program Code (please refer Appendix B: Visual Basic program code)

There is 6 forms created in Visual Basic application. There is copyright form, menu form, add and find Transmitting form, edit database form, marks statistics form, and graph form.

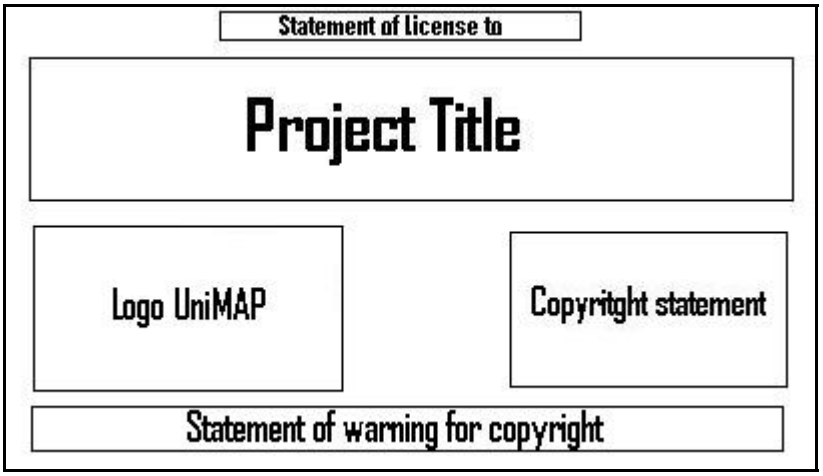

Figure 3.13: Copyright form design

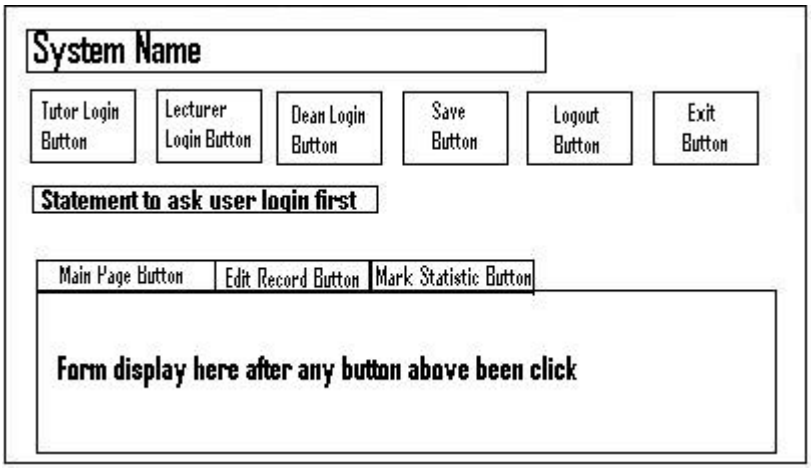

Figure 3.14: Menu form design

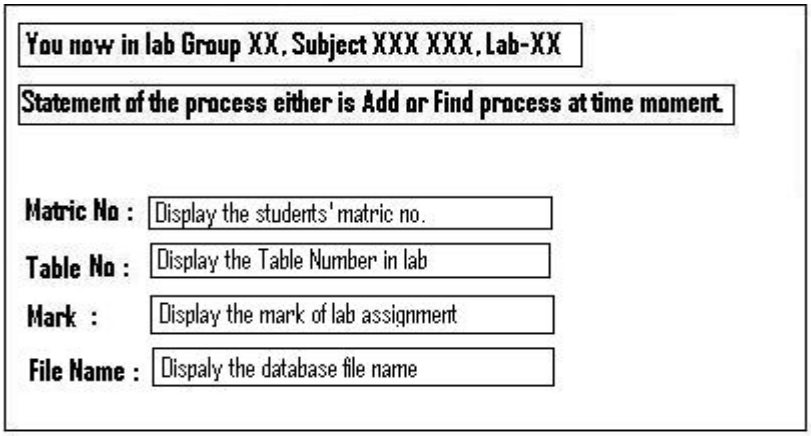

Figure 3.15: Add and Find Transmitting form design

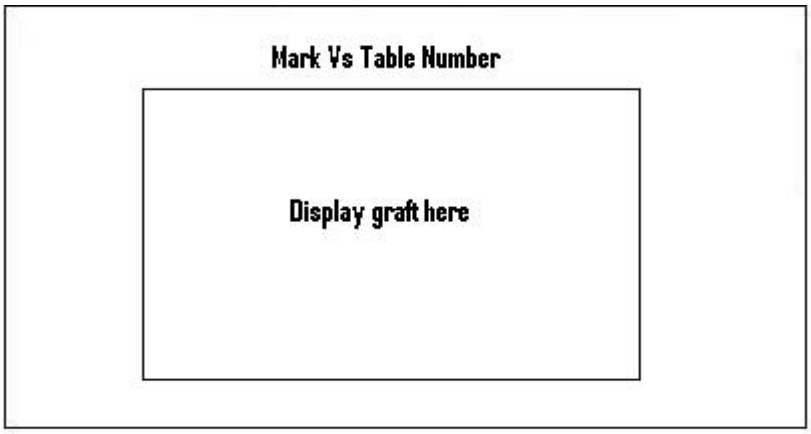

Figure 3.16: Edit database form design

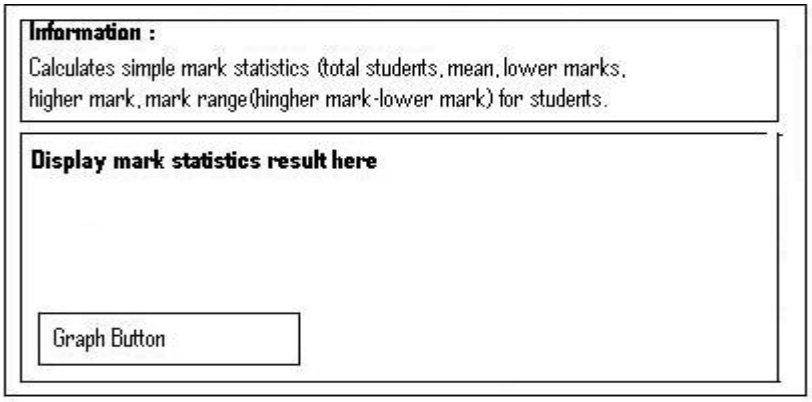

Figure 3.17: Marks Statistics form design

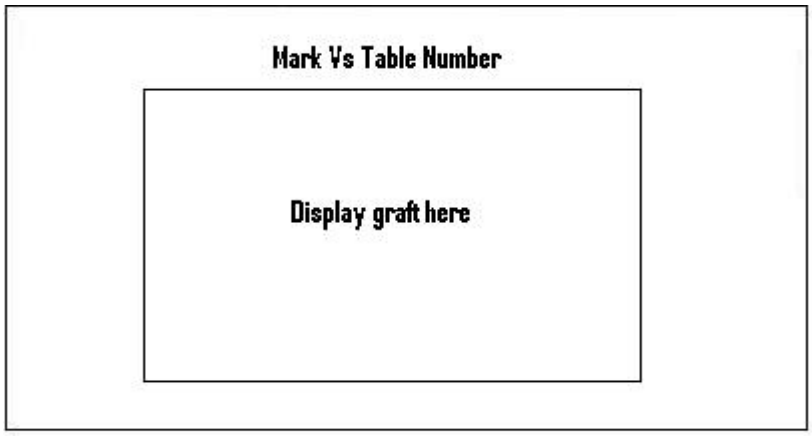

Figure 3.18: Graft form design

3.2.2.2 Microsoft Access

There is 2 different table that been created. Table User where got column 'Number', 'Username', 'Password', 'Identity' and 'Name'. Second table is Student where got column 'Matric Number', 'Table Number' and 'Mark'. There is 2 type of files will save after user use the WiDSTAC to key-in marks. Let say have one lab students from G6 taking subject EKT451, lab5. After the end of process two databases will keep. One of them will in path C:\VB\G6\EKT451\5.mdb, another will be C:\database\G6_VB_EKT451_ backUPfile.mdb as back-up file.

User : Table

No	Username	Password	Identity	Name
(AutoNumber)				

Figure 3.19: Table for User

Student : Table

Matric_No	Table_No	Mark

Figure 3.20: Table for Student

3.3 Implementation Phase

This phase we can divided to 2 part. Part 1 is hardware implementation phase and part 2 is software implementation phase.

3.3.1 Hardware Implementation Phase

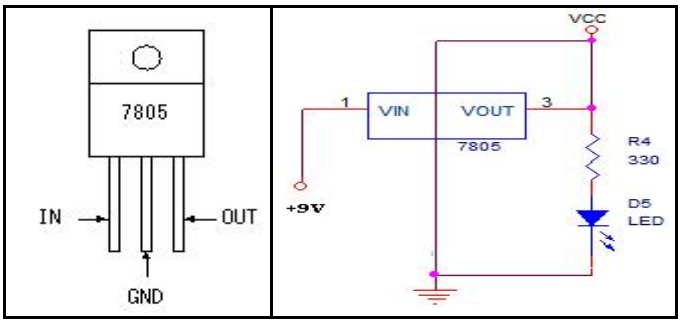

Figure 3.21: Power Regulator

The power regulator takes an input voltage of 9V and output a stable 5V, this is needed for the microcontroller as all component value have been selected based on a 5V supply.

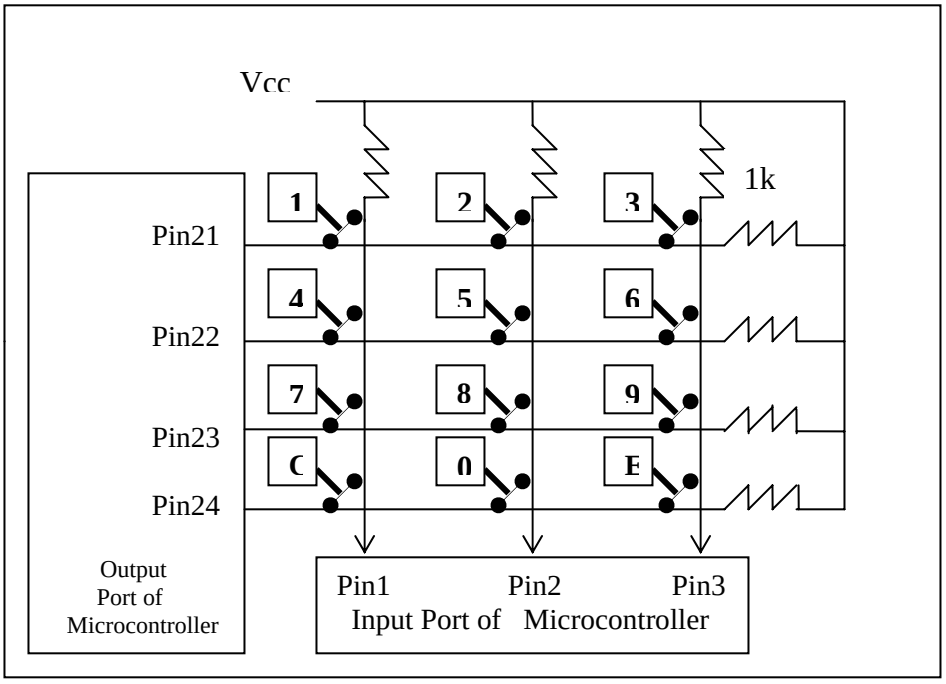

Figure 3.22: Keypad connection

The keypad button look like switch in circuit. The rows are connected to an output port and the columns are connected to an input port of the microcontroller. If no key has been pressed, reading the input port will give a 1 for all columns since they are connected to high (Vcc). If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground. It is the function of the microcontroller to scan the keypad continuously to detect the identify the key pressed. The resistor is act as a pull up resistor, to get 5V supply. It is used in the design of electronic logic circuits to ensure inputs to logic systems settle at expected logic levels if external devices are disconnected. This kind of resistor is also can used at the interface between two different types of logic devices.

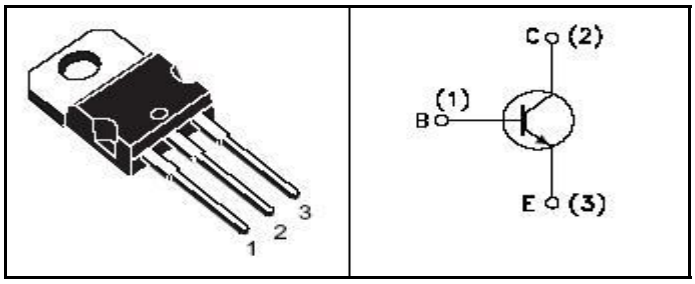

Figure 3.23: TIP31C [1]

TIP31C is NPN transistor use in medium power linear and switching applications. TIP31C is additional designations indicate increasing collector-base and collector emitter voltages. TIP31C is use as switch here. Collector (C) will connected to 9V, Base (B) will connected to pin 25 microcontroller, Emitter (E) will connect to pin 1 (Vcc) of RF Module transmitter TX-F9912. When pin 25 is provided 0 then the switch is open, no current flow. When pin 25 is provided 1 then the switch is close, current flow, RF Module transmitter TX-F9912 got 9V power.

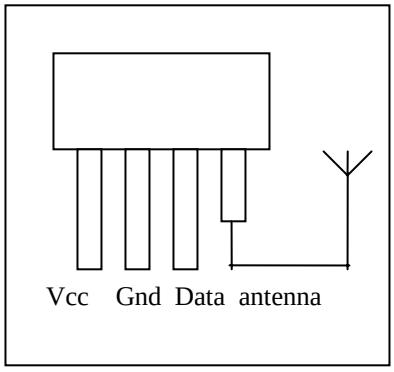

Figure 3.24: RF Module transmitter TX-F9912 connection pin

RF Module transmitter got 4 pin. Pin Vcc is connect to TIP31C to get 9V. Pin Gnd connected to Ground. Pin Data connect to pin 11 (P3.1/TXD) of microcontroller. (in

WiDSTAC). Pin Data connect to Collector of transistor BC547 (in device for receiving or transmitting from/to PC part).

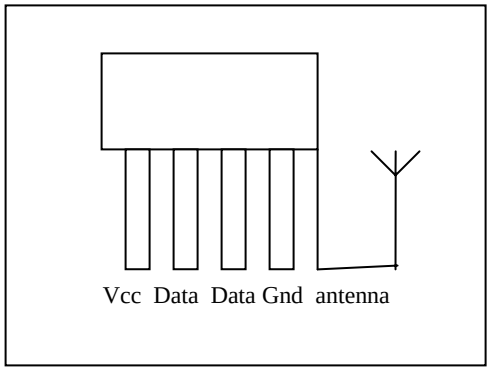

Figure 3.25: RF Module receiver RX-PCR1A connection pin

RF Module receiver have 4 pin with extra antenna. Pin Vcc connected to 5V. Two pin Data are connected together then connected to Base of transistor BC547 . (in WiDSTAC). The pin Data are connected to pin 11 of MAX 232 (in device for receiving or transmitting from/to PC part). Please refer to figure 3.23

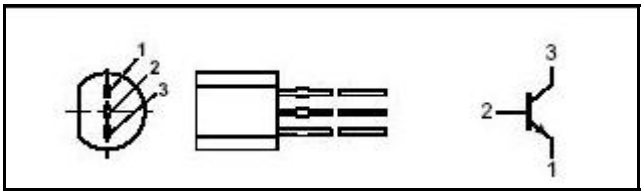

Figure 3.26: Transistor BC547

Transistor BC547 is used both in WiDSTAC and the device for receiving or transmitting from/to PC part. This transistor is used as inverter. In WiDSTAC, Collector(3) connected to 5V and Pin 10 (P3.0/RXD) of microcontroller. When Pin Data (RF Module receiver RX-PCR1A) is received 1 in Base(2) then the switch is open, Pin 10 will get value of 1 from voltage supply 5V and vice versa. Emitter (1) is connected to ground. In device for receiving or transmitting from/to PC part, Collector(3) is connected to 5V, Base(2) is connected to pin 12 (R1 out) from MAX 232 to receive data to transmit out. Emitter (1) is connected to ground.

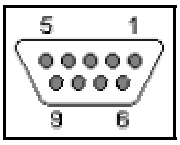

Figure 3.27: DB 9 female

DB 9 female is used to connect between PC and device for receiving or transmitting from/to PC part. Pin 1 (Carrier Detect), Pin 4(Data Terminal Ready) and Pin 6 (Data Set Ready) is connected together. Pin 7(Request to send) and Pin 8(Clear to send) is connected together. Pin 9 not used. Pin 5(ground) is connected to ground. Pin 2 (Received Data) is connected to Pin 13 (R1in) from MAX232. Pin 3 (transmitted data) is connected to Pin 14 (T1out) from MAX232.

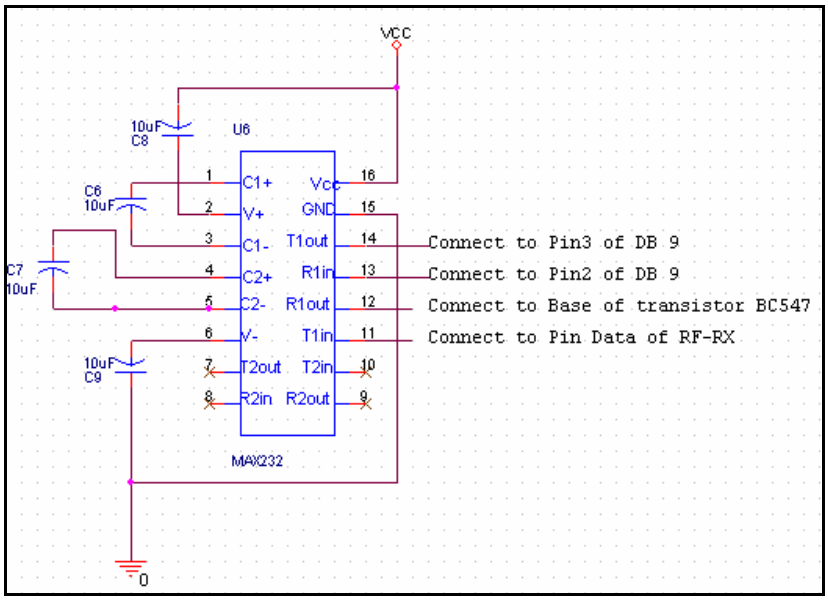

Figure 3.28: MAX 232 connection

MAX232 is made by Maxim Corporate. It can converts from RS232 voltage levels to TTL voltage levels and vice versa. MAX 232. Capacitor 10uF connected between Pin1 and Pin 3, Pin 4 and Pin 5, Pin 6 and Pin 15, Pin 2 and Pin16. Pin 14(T1out) is connected to Pin 3 of DB 9. Pin 13(R1in) is connected to Pin 2 of DB 9. Pin 12(R1out) is connected to Base of transistor BC547. Pin 11(T1in) is connected to Pin Data of RF-RX.

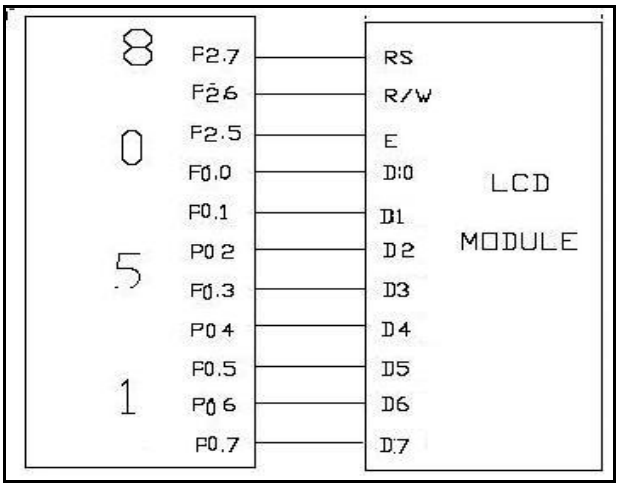

Figure 3.29: LCD connection

In LCD, Pin 2 is connected to 5V. Pin 1 and Pin 3 is connected with resistor 1k. At same time Pin 1 also grounded in the end of it. Pin data (DB0-DB7) is connected to Pin 32-Pin39 (P0.0-P0.7).

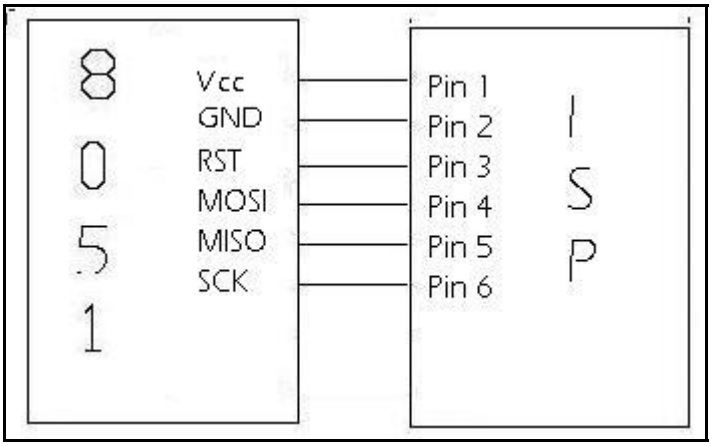

Figure 3.30: ISP connection

There is 6 pin in ISP to connect to microcontroller. MISO (Master In Slave Out) is the data input pin to the programmer. SCK(Serial Clock Output) is the clock output signal. MOSI (Master Out Slave In) is data output pin from the programmer. RST (Reset) is pin controls the target Device reset pin. It will driven High/Low according to device type.

In Microcontroller 8051 there is 8 pin that not been used, Pin 4 (P1.3), Pin 5(P1.4), Pin 12(P3.2), Pin13(P3.3), Pin 14(P3.4), Pin 15(P3.5), Pin 16(P3.6) and Pin 17(P3.7).

Table 3.2: Part 1 -Pin Connection of components in microcontroller

Components/pin		Microcontroller 8051 Pin
Keypad	Pin 1	Pin 1(P1.0)
	Pin 2	Pin 2(P1.1)
	Pin 3	Pin 3(P1.2)
	Pin 7	Pin 21(P2.0)
	Pin 6	Pin 22(P2.1)
	Pin 5	Pin23(P2.2)
	Pin 4	Pin 24(P2.3)
LCD	DO	Pin 32(P0.7)
	D1	Pin 33(P0.6)
	D2	Pin 34(P0.5)
	D3	Pin 35(P0.4)
	D4	Pin 36(P0.3)
	D5	Pin 37(P0.2)
	D6	Pin 38(P0.1)
	D7	Pin 39(P0.0)
	E	Pin 26(P2.5)
	R/W	Pin27(P2.6)
	RS	Pin 28(P2.7)
	Pin 2	Pin 40(5V)
	Pin 1	Pin 20(Ground)
ISP	Pin 1	Pin 40(5V)
	Pin 2	Pin 20(Ground)
	Pin 3	Pin 9(RST)
	Pin 4	Pin 6(P1.5/MOSI)
	Pin 5	Pin 7(P1.6/MISO)
	Pin 6	Pin 8(P1.7/SCK)
RF-RX	Vcc	Pin 40(5V)
	GND	Pin 20(Ground)
		Pin 29
RF-TX	DATA	Pin 11(TXD/P3.1)
	GND	Pin 20(Ground)
BC547	Pin C	Pin 10(RxD/P3.0)
	Pin E	Pin 20(Ground)

Table 3.3: Part 3 -Pin Connection of components in microcontroller

Components/pin		Microcontroller 8051 Pin
7805	GND Vcc	Pin 20(Ground) Pin 40(5V)
Crystal	Pin 1 Pin 2	Pin 18(XTAL1) Pin 19(XTAL2)
Resistor	4.7k	Pin 29(PSEN) Pin 30(ALE)
TIP31C	Pin B	Pin 25(P2.4)

Table 3.4: Pin Connection of components in MAX 232

Components/pin		MAX 232 Pin
7805	GND Vcc	Pin 15(Ground) Pin 16(5V)
DB 9	Pin 3 Pin 2	Pin 14(T1out) Pin 13(R1in)
RF-Tx	GND	Pin 15(Ground)
RF-Rx	Pin Data	Pin 11(T1in)
BC547	Pin B Pin E	Pin 12(R1out) Pin 15(Ground)
Capacitor 10uF		Pin 1 Pin 2 Pin 3 Pin 4 Pin 5 Pin 6

After testing the all components in Proto board until all is success.(please refer testing phase for detail). Then can do PCB board. Step 1 is do circuit lines using lettering paper on bare PCB board. Step 2 is take that PCB board do etch with Ferric Chloride to removes all non-masked copper. Step 3 is give the board a good wash under running water to remove all traces. Step 4 is cut the board to final size and drill holes in the board for components leads. These need very small holes (about 0.8mm). Step 5 is scrub the lettering paper and put the component inside and solder it.

Figure 3.31: Lettering paper

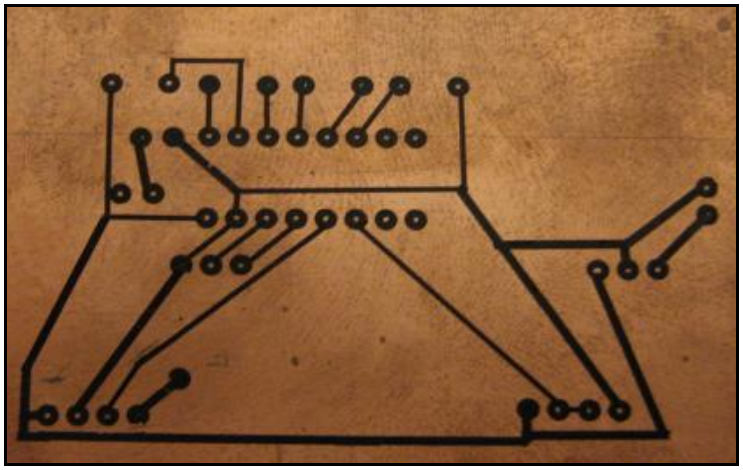

Figure 3.32: Do lettering in bare board (device for receiving or transmitting from/to PC part)

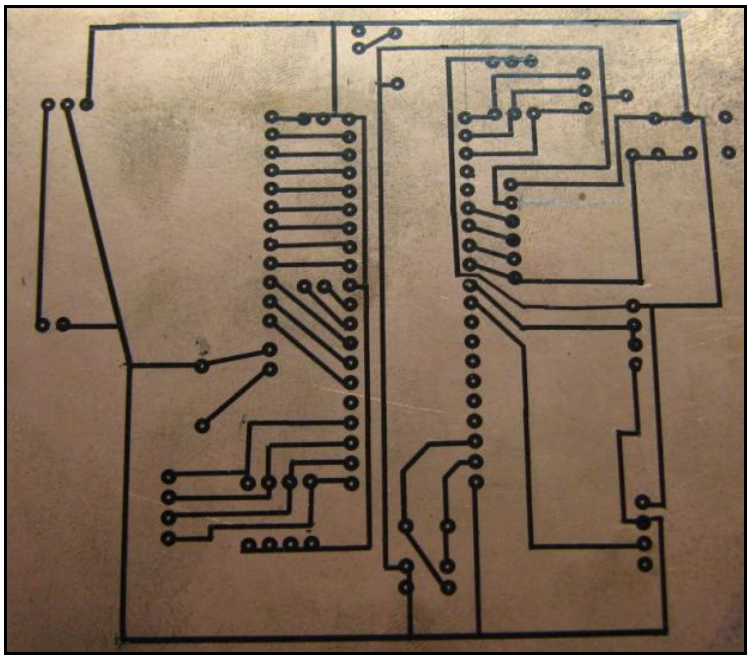

Figure 3.33: Do lettering in bare board (WiDSTAC part)

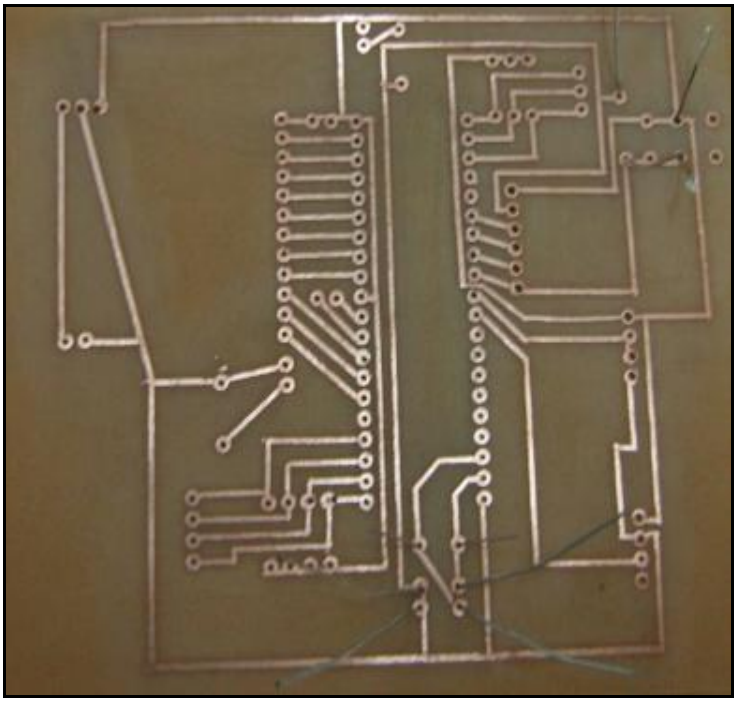

Figure 3.34: Put in components in PCB board

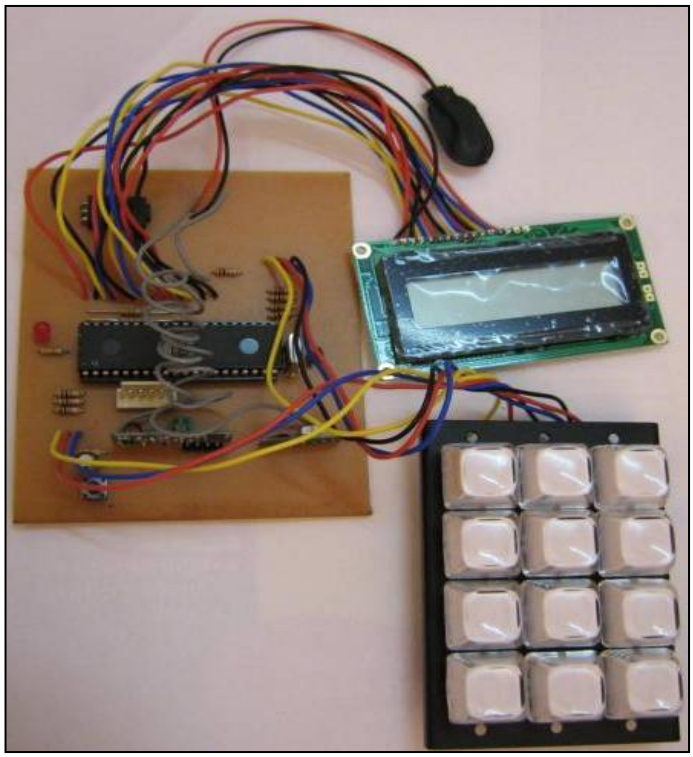

Figure 3.35: After solder (WiDSTAC)

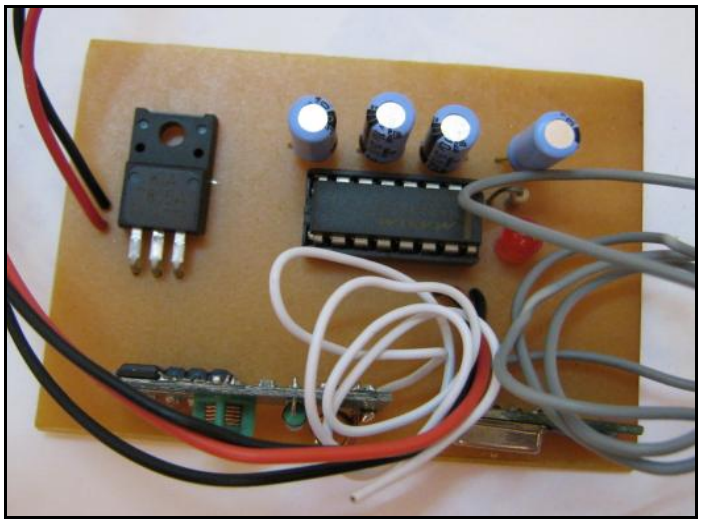

Figure 3.36: After solder (device for receiving or transmitting from/to PC part)

3.3.2 Software implementation Phase

```
;***********
;* 300Baud *
;***********
mov tmod,#20h   ;timer 1, mode 2 (auto reload)
mov th1,#0a0h           ;300 baud rate
mov scon,#50h   ;8 bit, 1 stop bit, REN enabled
setb tr1        ;start timer 1
```

Figure 3.37: Baud rate program

The baud rate for this program as 300. 8051 divides the crystal frequency by 12 to get machine cycle frequency. The value of crystal I used here is 11.0592 MHz, the machine cycle frequency is 921.6 kHz (11.0592 MHz /12 = 921.6 kHz). Before use by Timer 1, 921.6 kHz have to divided by 32 (8051 serial communication) and it gives 28,800 Hz. We use 28,800 Hz to find timer 1 value to set the baud rate. Since I want baud rate is 300, to find value for Timer 1 = 28,800/300 = 96 = 60hex = -160 = A0 hex

```
;*******************
;* Keypad Scanning *
;*******************
k1:     clr p2.0            ;ground all rows at once (4 pin output)
        clr p2.1
        clr p2.2
        clr p2.3
        mov a,p1                ;read all coloum (3 pin input)
        anl a,#00fh             ;mask unused bits
        cjne a,#00fh,k1         ;check till all keys released
k2:     call delay1             ;call delay1 subroutine
        mov a,p1                ;see if any key is pressed
        anl a,#00fh             ;mask unused bits
        cjne a,#00fh,over1      ;key pressed, wait closure
        sjmp k2                 ;chek till key pressed
over1:  call delay1             ;call delay1 subroutine
        mov a,p1                ;check key closure
        anl a,#00fh             ;mask unsed bits
        cjne a,#00fh,over2      ;key pressed,find row
        jmp k2                  ;if none, keep polling
```

Figure 3.38: Part 1:Keypad Scanning program

```
over2:  clr p2.0            ;ground row 0
        setb p2.1
        setb p2.2
        setb p2.3
        mov a,p1                    ;read all columns
        anl a,#00fh                 ;mask unused bits
        cjne a,#00fh,row_0          ;key row 0 find the column

        clr p2.1            ;ground row 1
        setb p2.0
        setb p2.2
        setb p2.3
        mov a,p1                    ;read all columns
        anl a,#00fh                 ;mask unused bits
        cjne a,#00fh,row_1          ;key row 1 find the column

        clr p2.2            ;ground row 2
        setb p2.0
        setb p2.1
        setb p2.3
        mov a,p1                    ;read all columns
        anl a,#00fh                 ;mask unused bits
        cjne a,#00fh,row_2          ;key row 2 find the column

        clr p2.3            ;ground row 3
        setb p2.0
        setb p2.1
        setb p2.2
        mov a,p1                    ;read all columns
        anl a,#00fh                 ;mask unused bits
        cjne a,#00fh,row_3          ;key row 3 find the column

        jmp k2                      ;if none,false input,repeat
row_0:  mov dptr,#kcode0            ;display "123"
        jmp find            ;jump find subroutine
row_1:  mov dptr,#kcode1            ;display "456"
        jmp find            ;jump find subroutine
row_2:  mov dptr,#kcode2            ;display "789"
        jmp find            ;jump find subroutine
row_3:  mov dptr,#kcode3            ;display "c0e"
find:   rrc a                       ;see if any CY bit low
        jnc match                   ;if zero,tet the ASCII code
        inc dptr            ;point to next column address
        jmp find            ;keep searching
```

Figure 3.39 : Part 2:Keypad Scanning program

```
match:   clr a                  ;set a=0 (match is found)
         movc a,@a+dptr         ;get ASCII code from table
         mov @r0,a              ;display pressed key
         cjne a,#"e",disp_1     ;if not "enter" press go disp_1
         jmp end_cov            ;jump end_cov
disp_1:  cjne a,#"c",disp_2     ;if not "cancel" press go
disp_2
         jmp end_cov            ;jump end_cov
disp_2:  inc r1                 ;increase r1
         acall data_in          ;call data_in subroutine
         inc r0                 ;increase r0
end_cov:
         ret                    ;return
```

Figure 3.40: Part 3:Keypad Scanning program

Keypad Scanning is the way to scanning and identifying the key that press. (please refer Figure 3.17: keypad connection). To detect a pressed key, the microcontroller will grounds all rows by providing 0 to the output latch then it reads the columns. If the data read from the columns is Pin 1- Pin 3 =111, no key has been pressed and the process continue until a key press is detected. Let say Pin 1-Pin 3=101, this means that a key in the column Pin 2 has been pressed. After a key press is detected, the microcontroller will go through the process of identifying the key. Staring with the top row, the microcontroller grounds it by providing a low to row Pin 21 only, then it reads the columns. If the data read is all 1s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns and checks for any zero. This process will continues until the row is identified. After identification of the row in which the key has been pressed, the next task is to find out which column the pressed key belongs to. This can done by used ASCII look-up table.

```
;********************
;* Delay1  Routine *
;********************   ;machine cycle
delay1:  mov r7,#05fh     ;1, 5fh=95
again1:  mov r6,#05fh     ;1, 5fh=95
again2:  djnz r6,again2   ;2
         djnz r7,again1   ;2
         ret              ;1
                          ;time delay=0.02s
;********************
;*Delay2  Routine *
;********************   ;machine cycle
delay2:  mov r7,#003h     ;1
again3:  mov r6,#0ffh     ;1, ffh=255
again4:  mov r5,#0ffh     ;1
again5:  djnz r5,again5   ;2
         djnz r6,again4   ;2
         djnz r7,again3   ;2
         ret              ;1
                          ;time delay=0.43s
```

Figure 3.41: Delay routine program

There is 2 delay routines in this program. The calculation of Delay1 Routine is : For the again2 loop, it have (2x95) x 1.085us = 206.15us. The again1 loop repeats the again2 loop 95 times, therefore, have 95 x 206.15us = 19584.25 us if not include overhead. However, 'mov r6,#05fh' and 'djnz r7,again1' at the beginning and end of the agian1 loop add (3 x 95 x 1.085us) = 309.225us to the time delay. As the result the total delay is 206.15us +19584.25 + 309.225us = 20099.625us = 0.02s. The calculation of Delay2 Routine is : For the again5 loop, it have (2x255) x 1.085us = 553.35us. The again4 loop repeats the again5 loop 95 times, therefore, have 255 x 553.35us = 141104.25 us if not include overhead. However, 'mov r5,#0ffh' and 'djnz r7,again3' at the beginning and end of the agian4 loop add (3 x 255 x 1.085us) = 830.025us to the time delay. And 'mov r6,#0ffh' and 'djnz r7,again3' at the beginning and end of the agian3 loop add (3 x 255 x 1.085us) = 830.025us to the time delay. As the result the total delay is 3 x (553.35us + 141104.25 us +830.025us +830.025us) = 0.43s

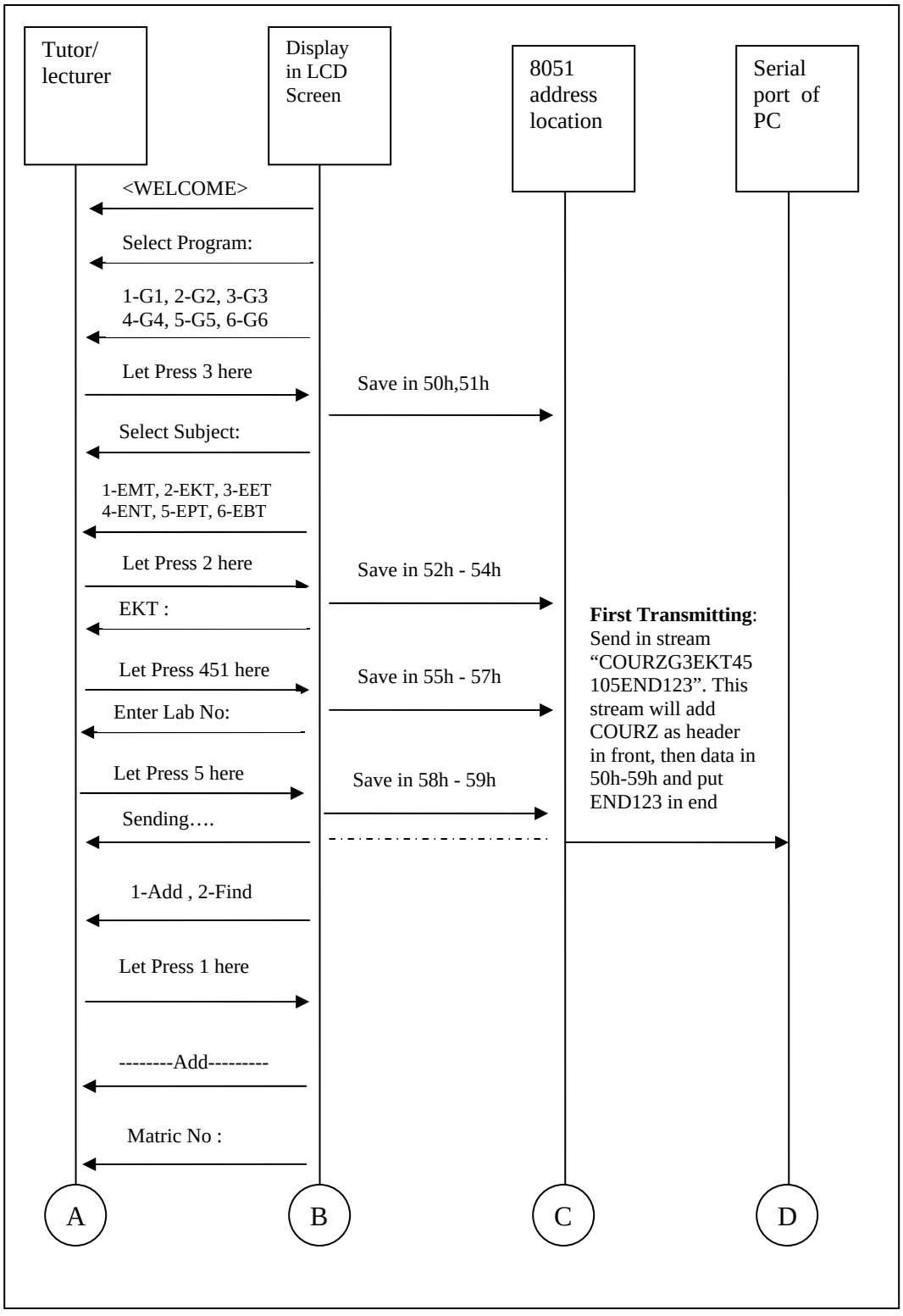

Figure 3.42: Part 1: Dataflow of 8051 program code

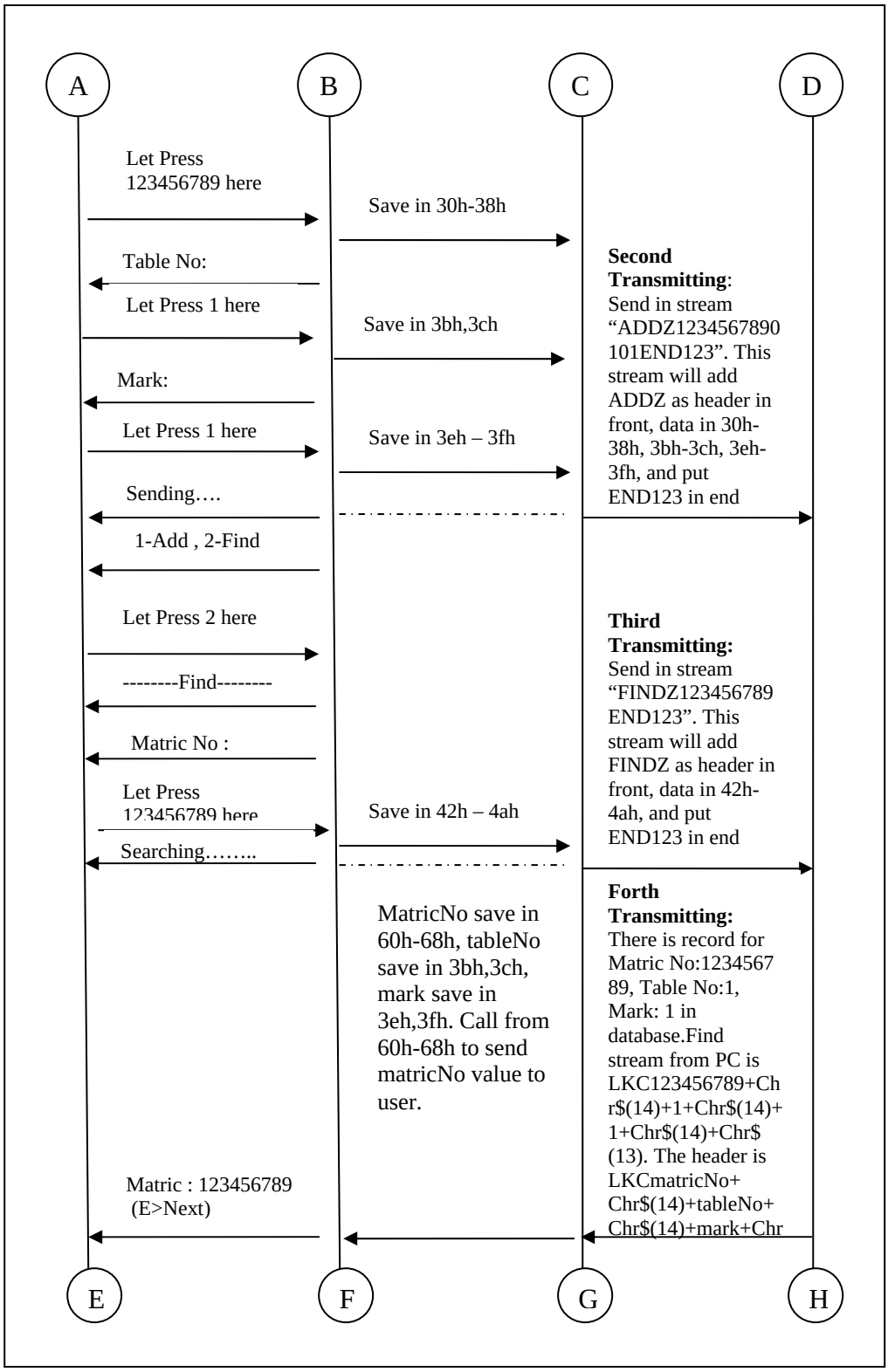

Figure 3.43: Part 2: Dataflow of 8051 program code

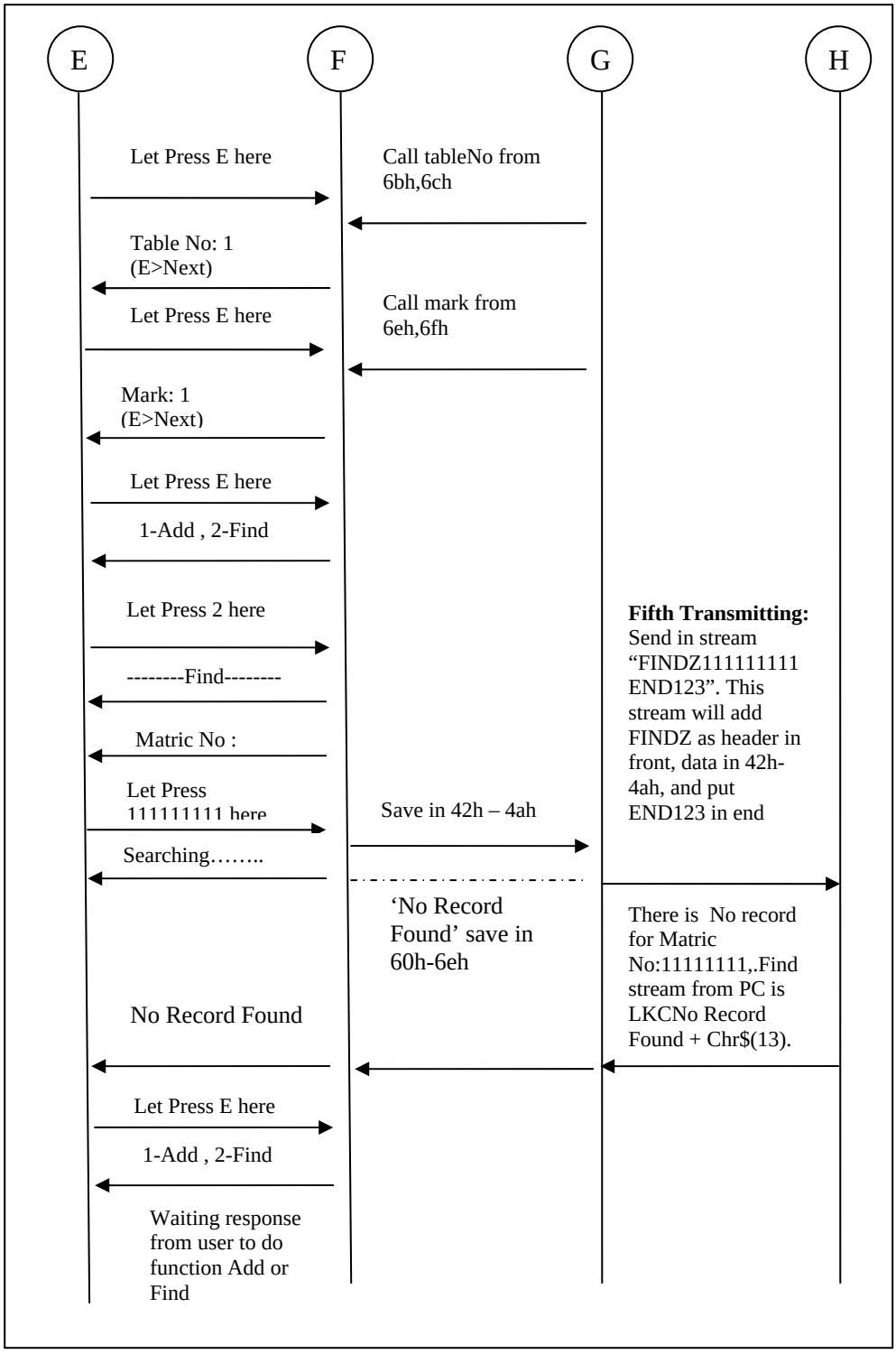

Figure 3.44: Part 3: Dataflow of 8051 program code

```
;************************************
;* Serial Sending for Find Process *
;************************************
s_find:
        setb p2.4           ;on TIP 31C
        mov r4,#5           ;r4=5, send stream 5 times
s_2:    mov a,#"F"          ;header "FINDZ"
        acall send
        mov a,#"I"
        acall send
        mov a,#"N"
        acall send
        mov a,#"D"
        acall send
        mov a,#"Z"
        acall send
        mov r0,#42h

        mov r3,#9           ;call 42-4Ah(maxtrix no.)
h_7:    clr a
        mov a,@r0
        acall send
        inc r0
        djnz r3,h_7

        mov a,#"E"          ;footer END123
        acall send
        mov a,#"N"
        acall send
        mov a,#"D"
        acall send
        mov a,#"1"
        acall send
        mov a,#"2"
        acall send
        mov a,#"3"
        acall send
        djnz r4,s_2
        clr p2.4
        ret
```

Figure 3.45: Serial Sending for find process

```
;*********************************
;* Receive Process *
;*********************************
s_r_find:
        mov a,#0c0h
        call command
        call recv
        cjne a,#"L",s_r_find        ;check header "LKC"
        call recv
        cjne a,#"K",s_r_find
        call recv
        cjne a,#"C",s_r_find
        call recv
        cjne a,#"N",r_3             ;if got N then papar
                                    ;"No Record Found"
        call data_in
r_1:    call recv
        cjne a,#13,r_2              ;take data until meet chr$(13)
r_6:    call k1
        cjne a,#"e",r_6
        jmp main
r_2:    call data_in
        jmp r_1



r_3:    mov r0,#60h                 ;take data and save in 60h
        mov @r0,a
        inc r0
r_4:    call recv
        cjne a,#13,r_5
        jmp recv_finish
r_5:    mov @r0,a
        inc r0
        jmp r_4
recv_finish:
        ret
```

Figure 3.46: Serial Receiving program code

Above two figures is part of find and receiving transmission data code. For more information about code please refer Appendix A: 8051 program code.

Now, let look Visual Basic program code. Please refer Appendix B: Visual Basic program code for more detail. There is 6 forms created in Visual Basic application. There is copyright form, menu form, add and find Transmitting form, edit database form, marks statistics form, and graph form.

Table 3.5: Forms' components

Form Name	**Component (quantity)**
Copyright form	Label (7) Image(1)-set Picture Property as the location of picture Timer(1)-set Interval Property the time in miliseconds
Menu form	Label(5) Textbox(5) CommandButton(3) – 'Main Page' to display Add and Find Transmitting form - 'Edit Record' to display Edit Database form - 'Mark Statistik' to display Mark Statistik form Toolbar(1)- create 6 buttons : 'tutor login', 'lecturer login', Dean Login', 'Save', 'Logout', 'Exit'
Add and Find Transmitting form	Label(9) CommandButton(1) Timer(1) – set time to take/read data MSComm1(1)- set MSComm1.CommPort = 1 MSComm1.Settings = "300,N,8,1" MSComm1.InputLen = 0 MSComm1.PortOpen = True MSComm1.RThreshold = 0
Edit Database form	CommandButton(1) Label(1) DataGrid(1)-set DataSource Property link to Adodc1 Adodc(1)-set ConnectionString Property as location of database, put table name as RecordSource Property
Marks statistics form	CommandButton(1) Label(7) Frame(2)
Graph form	Chart(1)

In Microsoft Access, there is 2 different table that been created. Table User where got column ‘Number’, ‘Username’, ‘Password’, ‘Identity’ and ’Name’. Second table is Student where got column ‘Matric Number’, ‘Table Number’ and ‘Mark’. Set ‘Matic Number’ as Primary key. There is 2 type of files will save after user use the WiDSTAC to key-in marks. Let say have one lab students from G6 taking subject EKT451, lab5. After the end of process two databases will keep. One of them will in path C:\VB\G6\EKT451\5.mdb, another will be C:\database\G6_VB_EKT451_backUPfile.mdb as back-up file. Set Database a password : >Tools>security>set database password and set the password but before this have to make sure the database is open in format Open Exclusive. It the same way if want to delete the password.

3.4 Testing Phase

For the hardware part, test the functional communication circuit with help of HyperTerminal software. It very useful for serial communication part. Try send ‘0123’ to display in window HyperTerminal. Plug Proto board that have all component to connect serial to PC’s serial port. Run HyperTerminal.exe, Put any name to name the New Connection like figure 3.42 just put 1 as the name of new connection.

Figure 3.47: New Connection of HyperTerminal

Figure 3.48: Set Connection as COM1

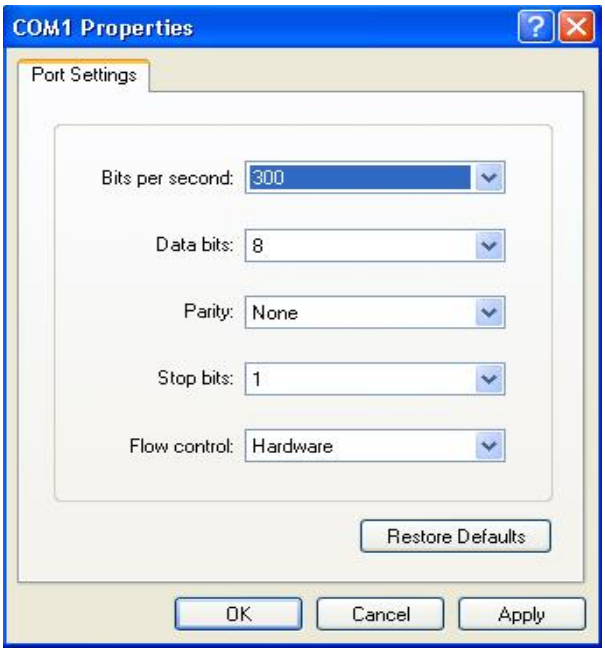

Figure 3.49: Set the COM1 Properties

The communication is using port COM1. It need to set to 300 baud rate because the 8051 program already set it as 300 baud rate. It have to be same baud rate to have transmission. The data bits is 8 bit. Did not use parity bit and use 1 stop bits.

Before test the 8051 program code, it need to compile it first to generate .Hex file by use cmd.exe. Type asm follow by file to compile it. If have error, edit the program until it success. Next it use ASP.exe to load to ISP connector to burn the program to microcontroller to test the program that been written. In ASP.exe, just click on button Open File to choice the .Hex file that been generated. After that, click on button Write to start the burning process. Have to make sure the connection in ISP connector in board already done before burning process started.

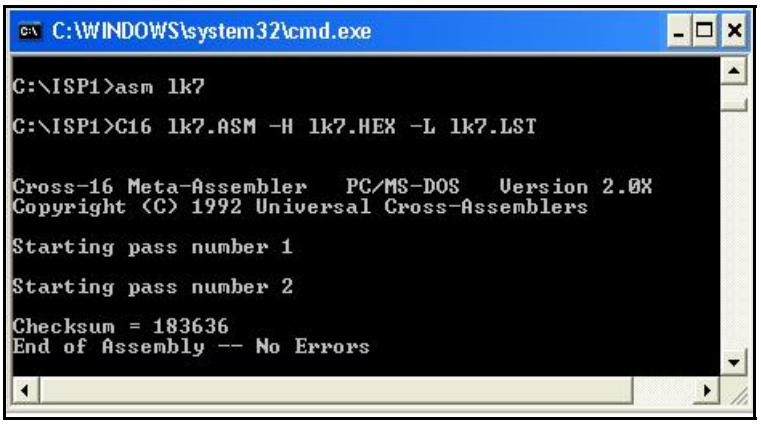

Figure 3.50: To compile 8051 program

Figure 3.51: To burn program 8051

In Visual Basic, after the form is made, to test it just click on button Run>Start. If got error edit it until it success. Have to make sure the link between serial communication and database is working well. All the software programming is test part by part to easy detected the problem. In this testing phase, it might change our previous software design or hardware design. Good design take less testing phase time. The testing phase is done if the system already function well.