FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

PREPAID SYSTEM FOR DOMESTIC ENERGY METER

By:

AATREYI BAL
(08148003059)

TANAYA BOSE
(08148003075)

ARINDAM BOSE
(08148003044)

SAIKAT MAJUMDER
(08148003008)

(4th year, ECE)

Under Guidance of:

ASST. PROF. SREYA CHAKRABARTI
Department of Electronics and Communication
Future Institute of Engineering and Management

ASST. PROF. MILAN MAZUMDAR
The Institution of Electronics and Telecommunication Engineers

ACKNOWLEDGEMENT

In doing this project, we have received help from our teachers and institution. We would like to thank our guide Mrs. Sreya Chakrabarti for her tireless help to us. We are also indebted to Mr. Milan Mazumdar, our external mentor, for his valuable guidance. We would thank our college, Future Institute of Engineering & Management, our HOD Mr. Debashish Chakrabarty and Mr. Sourav Ganguly for their cooperation.

ABSTRACT

In the present metering system there is scope for incorrect reading of meter due to human error. Moreover the billing amount for a particular month greatly depends on the date of meter reading. Again, if someone stays away from home for a long time, it is customary to inform the regional electric supply office, addressed to Commercial Executive, in order to avoid false meter reading. In spite of that, a bill is received every month which compulsorily payable. However, all this problems are resolved in the "Prepaid System for Domestic Energy Meter". This project deals with the designing of the prepaid system only and not the energy meter. The system is compatible with the domestic digital energy meters that are used in most of the houses at present.

TABLE OF CONTENTS

1. INTRODUCTION

This project, “PREPAID SYSTEM FOR DOMESTIC ENERGY METER”, aims at introducing a new model of electricity billing system. It will reduce problem associated with billing consumer living in isolated area and reduces deployment of manpower for taking meter readings. Moreover, by starting to understand your consumption, you are better empowered to make effective changes regarding your electricity consumption. Electricity coupons will be available at nearby shops. Maximum units to be used are programmed. This data is given to Microcontroller. Microcontroller is connected to digital energy meter. MCU is programmed to decrease the balance amount as a response to the information from the digital energy meter. Buzzer is used to warn the user. When maximum use is made, relay will cut off and controller has to be reset.

Before entering the project details we would like to mention that the project title emphasizes on “Domestic energy meter” because the basic difference between the domestic and industrial energy meter is that domestic sites receive electricity on one phase, whereas industrial sites receive it in three phases. Our prepaid system is equipped to measure single phase power, since this is what the vast majority of households will have in place.

2. WORKING PRINCIPLE

Every consumer can buy a memory card (is nothing but an EEPROM IC) with a password stored inside it using a MC program. The memory card is available at various ranges (i.e. Rs.50, Rs.100, Rs.200 etc.).In our project we have given the name for memory card as smart card.When the consumer inserts a smart card into the card reader which is connected kit. Then the card reader will read the stored information and delete the information from the EEPROM IC (smart card) using the MC program, so that the smart card cannot be reused by others. Suppose if a consumer buys a card for Rs.50/- he/she can insert this amount through the card reader so that prepaid energy meter with tariff indicator kit will be activated. According to the power consumption the amount will be reduced. When the amount is over, the relay will automatically shut down the whole system. In our project we also have a provision to give an alarm sound to consumer before the whole amount is reduced.

➢ Here's the procedure to create the cards.

- How to program a new card:

For making a unit price card for Rs.2.50

1. Insert the card into the Programmer
2. Dial 1*0250#

The format is
1 for unit price
* For start process
- Higher digit of the unit price
- Lower digit of the unit price
- Higher digit of the unit paisa
- Lower digit of the unit paisa

3. The red led will blink for every key press
4. If the programming done successfully then the Green led will long blink finally.
5. If it fails then the RED led will give a long blink

- For making a Recharge card for Rs.400

1. Insert the card into the Programmer
2. Dial 2*0400#
3. The red led will blink for every key press
4. If the programming done successfully then the Green led will long blink finally.
5. If it fails then the RED led will give a long blink.

3. BLOCK DIAGRAM

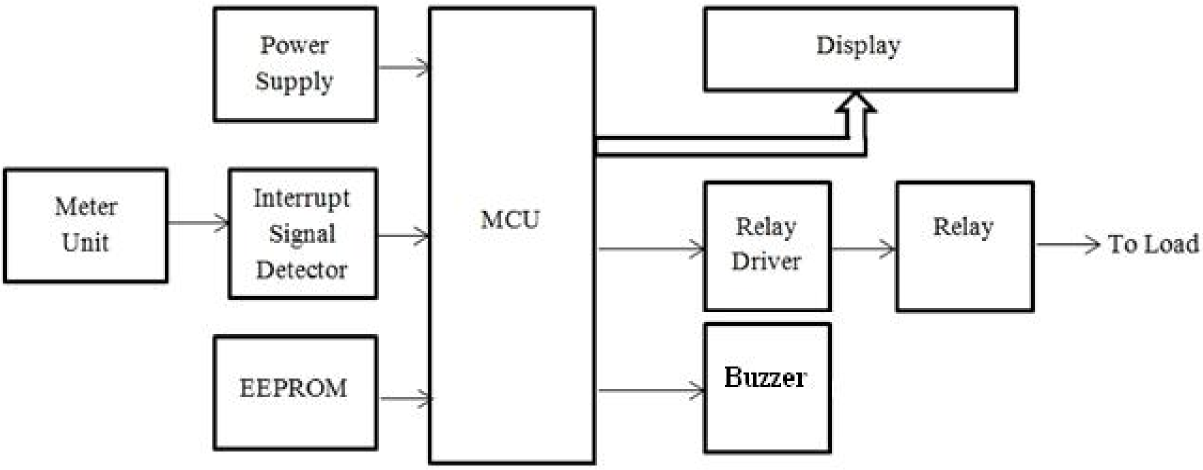


Fig 3.1: Block Diagram of Main Functional Circuit

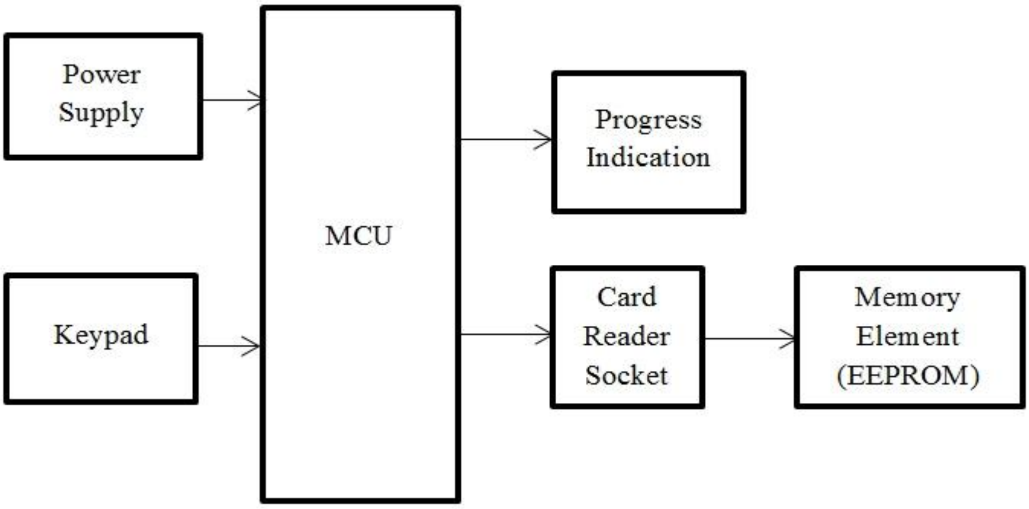


Fig 3.2: Block Diagram of Memory Element Programming Circuit

4. BLOCK DESCRIPTION

Main Functional Circuit

Meter Unit: For this project, we are considering the digital energy meter. The meter reading increases for every 3200 pulses. As soon as one unit of energy is spent, the meter unit sends an interrupt signal to the microcontroller via the interrupt signal detector.

Interrupt Signal Detector: This unit actually receives the signal from the meter unit, detects it and finally forwards it to the microcontroller.

Microcontroller: When the microcontroller unit receives the interrupt signal from the interrupt signal detector, it increases the meter reading count by one and resets the pulse count. The balance is also decreased as per tariff. Other than the computation activities, the microcontroller acts as the interface between the meter unit and the EEPROM.

EEPROM: The EEPROM plays a dual role in this circuit. It acts as the memory of the microcontroller and also as the rechargeable memory/smart card. The EEPROM can be separately programmed to store the tariff and the recharged balance. Once the EEPROM is read by the microcontroller, it becomes an invalid card and cannot be reused for that balance.

Display Unit: Usually, a LCD is used for the display unit. The display unit is used to indicate the recharged balance, the meter reading, the pulse count and the tariff. When each pulse of energy is spent, the pulse count is indicated. As soon the pulse count reaches 3200, the meter reading increases by one and the pulse reading is indicated to be reset. The balance is also decreased as per tariff.

Relay Driver: The relay driver interfaces the relay with the microcontroller. The microcontroller can provide only 5 volts whereas the relay requires 12 volts to function. Relay driver steps up the voltage and runs the relay. It also indicates the relay when to cut off the main supply.

Relay: The relay is the interface between the prepaid system and the main supply. When the balance amount decreases to a critical value, the relay is indicated by the driver to snap the main supply.

Buzzer: When the critical amount is reached, the microcontroller sends a signal to the buzzer which sends it ringing thereby making the customer aware.

Power Supply: This unit provides the necessary voltage (VCC and GND) to the circuit.

Memory Element Programming Circuit

Keypad Unit: This unit is used to enter the code to input the tariff and the recharge amount into the EEPROM. It is a 4x3 matrix of switches.

Microcontroller: The microcontroller interfaces the EEPROM to the keypad unit. It receives the input from the keypad and accordingly programs the EEPROM.

EEPROM: Here the EEPROM is the recharge card. When the tariff changes, it is used by the electric supply official to update the new tariff at every customer's system. Otherwise, when the balance exhausts the customer will get the EEPROM reprogrammed at the dealer's office.

Progress Indication Unit: This unit is actually a combination of two LEDs — one red and one green. With every key press the red LED blinks to indicate proper functioning of keys. When the microcontroller is successfully programmed, the green LED blinks twice to indicate success. If the red LED blinks after programming, it indicates unsuccessful programming of the EEPROM.

Power Supply: This unit provides the necessary voltage (VCC and GND) to the circuit.

5. CIRCUIT DIAGRAM

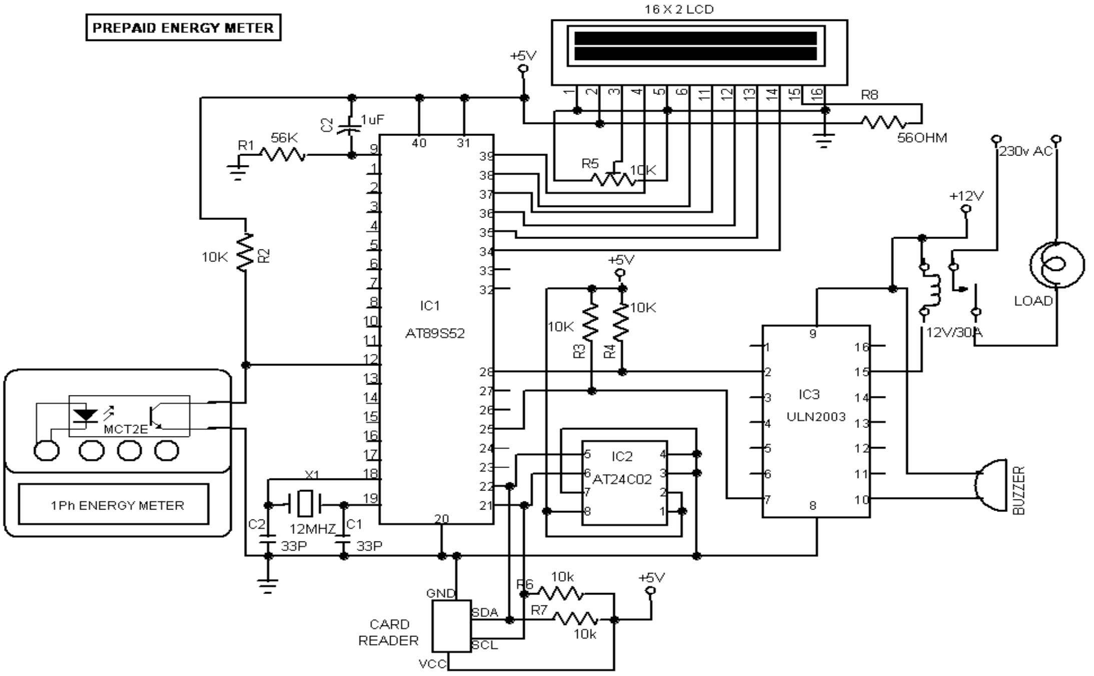


Fig 4.1: Circuit 1

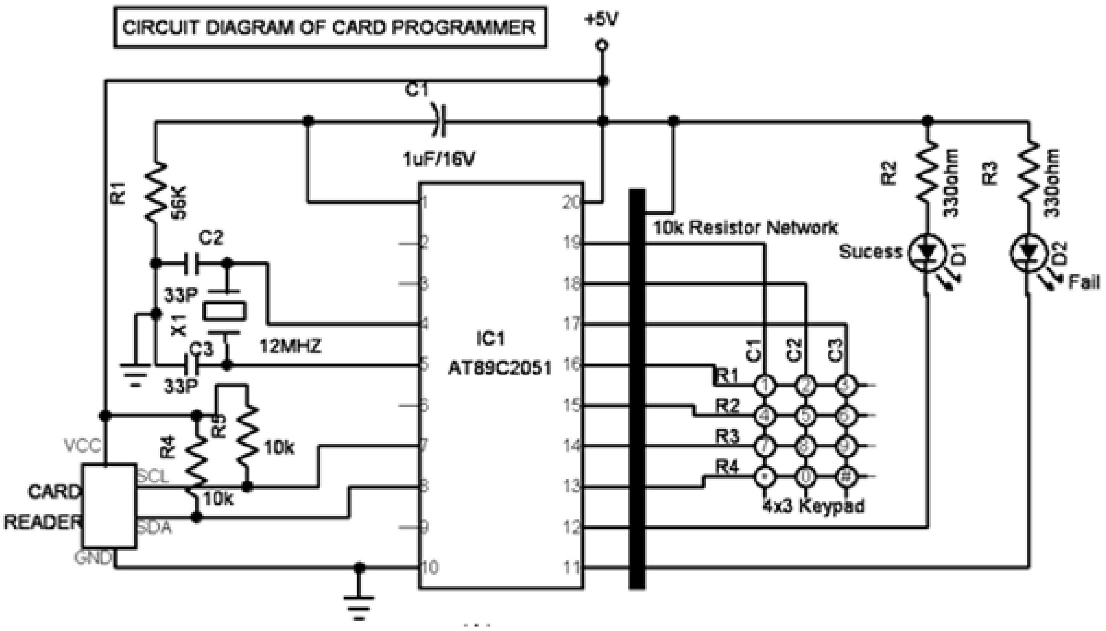
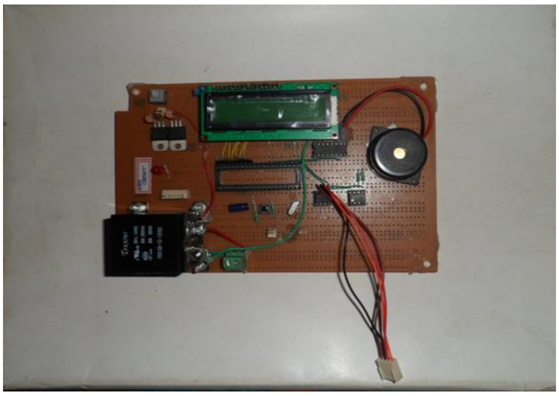
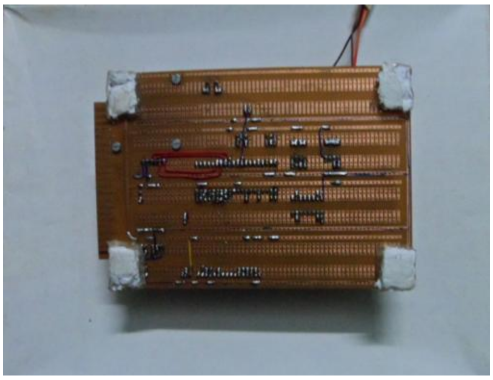


Fig 4.2: Circuit 2

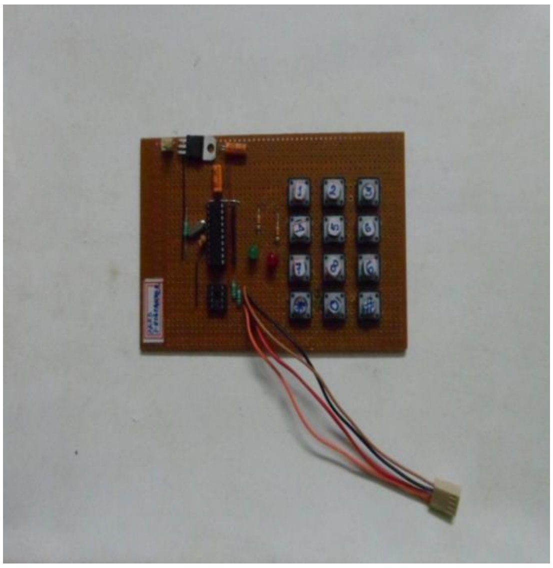
6. ORIGINAL CIRCUIT



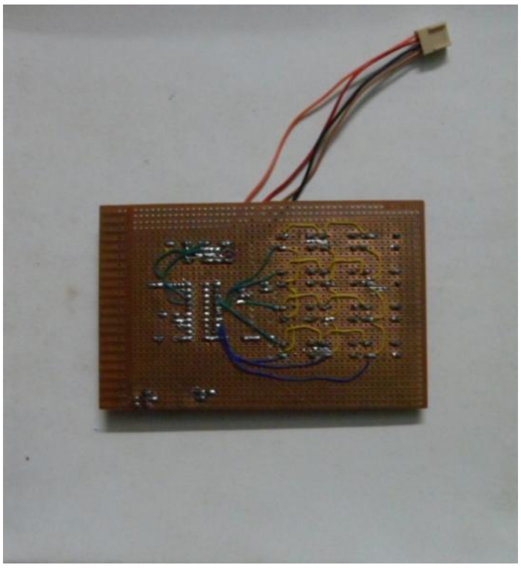
Main Circuit (Front View)



Main Circuit (Rear View)



Card Programmer (Front View)



Card Programmer (Rear View)

7. ADVANTAGES & DISADVANTAGES

This project has several advantages:

1. Conservation of energy.
2. Alert against unauthorized of the power supply.
3. Pay as per use.
4. Easy billing system.

However, like any other project this one has a few disadvantages:

1. Security issues.
2. Need of manual help in changing tariff.

8. CONCLUSION

The project has immense future prospect. For up-gradation, modem connection can be established between the power supply office and the individual meters in order to maintain the database of customers. The energy meter can also be up-graded with improved features. The project can be extended to serve industrial energy meters also.

9. REFERENCES

1. *Handbook for Electricity Metering*,TenthEditionby The Edison Electric Institute.
2. *The 8051 Microcontroller and Embedded Systems Using Assembly and C*, Second Edition by Mazidi, MazidiandMcKinlay,
3. *http://www.scribd.com*
4. *http://www.8051projects.info*
5. *http://www.8051projects.net*
6. *http://www.alldatasheets.com*
7. *Datasheets of AT89S52, AT24C02,ULN2003*

APPENDIX

A. Prepaid Energy Meter Program

```
RB0     EQU     000H    ; Select Register Bank 0
RB1     EQU     008H    ; Select Register Bank 1  ...poke to PSW to use

;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
;                       PORT DECLERATION
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
SDA1      EQU       P2.1       ;SDA=PIN5
SCL1      EQU       P2.0                    ;SCL=PIN6

WTCMD EQU   10100110B      ;WRITE DATA COMMAND Note 3
RDCMD EQU   10100111B      ;READ DATA COMMAND Note 3

WTCMD1 EQU   10100000B      ;WRITE DATA COMMAND Note 3
RDCMD1 EQU   10100001B      ;READ DATA COMMAND Note 3

RELAY                   EQU       P2.7
BUZZER  EQU       P2.4
; ***LCD CONTROL***
LCD_RS    EQU    P0.0     ;LCD REGISTER SELECT LINE
LCD_E     EQU    P0.1    ;LCD ENABLE LINE
LCD_DB4   EQU    P0.2     ;PORT 1 IS USED FOR DATA
LCD_DB5   EQU    P0.3     ;USED FOR DATA
LCD_DB6   EQU    P0.4    ;FOR DATA
LCD_DB7   EQU    P0.5     ;FOR DATA
; ***CURSOR CONTROL INSTRUCTIONS***

OFFCUR    EQU    0CH
BLINKCUR  EQU    0DH

; ***DISPLAY CONTROL INSTRUCTIONS***

CLRDSP    EQU    01H
ONDSP     EQU    0CH

; ***SYSTEM INSTRUCTIONS***

CONFIG    EQU    28H      ; 4-BIT DATA,2 LINES,5X7 MATRIX LCD
ENTRYMODE EQU    6        ; INCREMENT CURSOR DON'T SHIFT DISPLAY

DSEG          ; This is internal data memory
ORG     20H     ; Bit adressable memory

FLAGS1: DS                  1
        BCDCARRY            BIT        FLAGS1.0
        CARRY               BIT        FLAGS1.1
        TBIT                BIT        FLAGS1.2
```

```
           TBIT1              BIT   FLAGS1.3

READING:          DS                2
AMOUNT:           DS                3
COUNTER:          DS                2
TEMP:             DS                1
PRICE:    DS                2
BALANCE:          DS                1
BUZZ_COUNT:       DS        1
READ_BYTE:        DS        3
F1:               DS                1
F2:               DS                1
F3:               DS                1

STACK:   DS         1
CSEG         ; Code begins here

;  ---------==========----------=========---------=========---------
;  Main routine. Program execution starts here.
;  ---------==========----------=========---------=========---------
                  ORG     00H    ; Reset
                  AJMP MAIN

                  ORG 0003H
                  PUSH PSW
                  PUSH ACC
                  MOV PSW,#RB1               ; Select register bank 0
                  CALL INC_COUNTER
                  POP ACC
                  POP PSW
                  RETI
;  ---------==========----------=========---------=========---------
MAIN:
      MOV SP,#50H
      MOV PSW,#RB0         ; Select register bank 0
      MOV IE,#10000001B
          CALL RESETLCD4
          CALL TITLE1
          CLR BUZZER
                  CALL TITLE11
                  CALL DELAYY
                  CALL TITLE12
                  CALL DELAYY
                  CALL TITLE13
                  CALL DELAYY
          SETB RELAY
          CLR TBIT1
          MOV BUZZ_COUNT,#00H
          MOV READ_BYTE,#0FFH
```

```
        CALL READ_COUNTER
        MOV A,COUNTER
        CJNE A,#0FFH,BYPASS


                        CALL RESET_READING
                        CALL RESET_AMT
                        CALL RESET_COUNTER
                        CALL RESET_PRICE
                        CALL RESET_BALANCE                    ;RELAY ON/OFF BYTE
;                       CALL STORE_UNIT_PRICE
;                       CALL AMT_RECHARGE
                        CALL SYSTEM_RESET
                        CALL DELAYYS

BYPASS:
                        CALL READ_COUNTER
                        CALL READ_PRICE
                        CALL READ_BALANCE

MAINS:  CALL TITLE1
                        CALL DELAYY

                        MOV A,BALANCE
                        CJNE A,#00H,FG1
                        CLR RELAY
                        CALL RECHAGRE
                        CALL DELAYY
                        SETB BUZZER
                        AJMP MAINS

FG1:            SETB RELAY
                        MOV A,BUZZ_COUNT                      ;CHK TO SWITCH OFF THE BUZZER
                        CJNE A,#00H,AZX1
                        CLR BUZZER
                        AJMP AZX2
AZX1:           DEC BUZZ_COUNT
AZX2:
                        MOV R1,#READING                                       ;GET DATA IN
BYTES(RAM)
                        MOV R4,#05H                                           ;DATA ADDRESS IN
EEPROM
                        MOV R6,#2                                             ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        CALL DISP_READING
                        MOV TEMP,READING
                        CALL SEP_DISP
                        MOV TEMP,READING+1
                        CALL SEP_DISP

                        CALL DELAYY

                        MOV R1,#AMOUNT                                      ;GET DATA IN
BYTES(RAM)
                        MOV R4,#0AH                                         ;DATA ADDRESS IN
EEPROM
                        MOV R6,#3                                           ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        CALL AMT_READING
                        MOV TEMP,AMOUNT
                        CALL SEP_DISP
                        MOV TEMP,AMOUNT+1
                        CALL SEP_DISP
                        MOV R4,#'.'
    CALL WRLCDDATA
    CALL MDELAY
    MOV TEMP,AMOUNT+2
                        CALL SEP_DISP


                        CALL DELAYY
                        MOV R1,#COUNTER                                     ;GET DATA IN
BYTES(RAM)
                        MOV R4,#0EH                                         ;DATA ADDRESS IN
EEPROM
                        MOV R6,#2                                           ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        CALL COUNT_READING
                        MOV TEMP,COUNTER
                        CALL SEP_DISP
                        MOV TEMP,COUNTER+1
                        CALL SEP_DISP

                        CALL DELAYY
                        MOV R1,#PRICE                                   ;GET DATA IN BYTES(RAM)
                        MOV R4,#10H                                         ;DATA ADDRESS IN
EEPROM
                        MOV R6,#2                                           ;NUMBER OF BYTES
                        CALL READ_EEPROM


                        CALL READ_PRICE
                        CALL UNIT_PRICE
                        MOV A,PRICE
                        ADD A,#30h
                        MOV R4,A
                        CALL WRLCDDATA
    CALL MDELAY
                    MOV R4,#'.'
    CALL WRLCDDATA
    CALL MDELAY
                    MOV TEMP,PRICE+1
                    CALL SEP_DISP




                    CALL DELAYY
                    AJMP MAINS
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
;                   INCREMENT COUNTER BY 1
;                   IF COUNT=3200 THEN INCREMENT READING
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
INC_COUNTER:
                    MOV A,COUNTER+1
                    ADD A,#01
                    DA A
                    MOV COUNTER+1,A
                    JNC DCV2
                    MOV A,COUNTER
                    ADD A,#01
                    DA A
                    MOV COUNTER,A
                    CJNE A,#32h,DCV2
                    MOV COUNTER,#00H
              MOV COUNTER+1,#00H
              MOV R1,#COUNTER                   ;store COUNT
                    MOV R4,#0EH                         ;Starting Address IN EEPROM
                    MOV R6,#2                           ;STORE 2 BYTES
                    CALL STORE_EEPROM
                    CALL DELAY
                    AJMP DVB1

DCV2:         MOV R1,#COUNTER                   ;store COUNT
                    MOV R4,#0EH                         ;Starting Address IN EEPROM
                    MOV R6,#2                           ;STORE 2 BYTES
                    CALL STORE_EEPROM
                    CALL DELAY
                    RET

DVB1:   MOV A,READING+1                 ;INCREMENT READING BY 1
                    ADD A,#01
                    DA A
                    MOV READING+1,A
                    JNC DCS1
                    MOV A,READING
```

```
                        ADD A,#01
                        DA A
                        MOV READING,A
DCS1:           MOV R1,#READING                         ;store READING
                        MOV R4,#05H                             ;Starting Address IN EEPROM
                        MOV R6,#2                               ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY

                        MOV A,AMOUNT+2                                  ;SUBTRACT AMT0
FROM TOTAL0
                        CLR C
                        SUBB A,PRICE+1
                        CALL BCD_CONV
                        MOV AMOUNT+2,A
                        MOV A,AMOUNT+1                                  ;SUBTRACT AMT1
FROM TOTAL1
                        SUBB A,PRICE
                        CALL BCD_CONV
                        MOV AMOUNT+1,A
                        MOV A,AMOUNT                            ;SUBTRACT AMT2 FROM
TOTAL2
                        SUBB A,#00h
                        CALL BCD_CONV
                        MOV AMOUNT,A

                        MOV R1,#AMOUNT                          ;store AMOUNT
                        MOV R4,#0AH                             ;Starting Address IN EEPROM
                        MOV R6,#3                               ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY

                        MOV A,AMOUNT+1
                        CJNE A,#40H,FCX1
                        MOV BUZZ_COUNT,#02H
                        SETB BUZZER
FCX1:           CJNE A,#38H,FAX1
                        MOV BUZZ_COUNT,#02H
                        SETB BUZZER
FAX1:           CJNE A,#41H,FAAX1
                        MOV BUZZ_COUNT,#02H
                        SETB BUZZER
FAAX1:   CJNE A,#20H,FCX2
                        MOV BUZZ_COUNT,#03H
                        SETB BUZZER
FCX2:           CJNE A,#19H,FAX2
                        MOV BUZZ_COUNT,#03H
                        SETB BUZZER
FAX2:           CJNE A,#21H,FAAX2
```

```
                        MOV BUZZ_COUNT,#03H
                        SETB BUZZER
FAAX2:   CJNE A,#10H,FCX3
                        MOV BUZZ_COUNT,#04H
                        SETB BUZZER
FCX3:           CJNE A,#11H,FCX4
                        MOV BUZZ_COUNT,#04H
                        SETB BUZZER
FCX4:           CJNE A,#09H,FAX4
                        MOV BUZZ_COUNT,#04H
                        SETB BUZZER
FAX4:
                        MOV A,AMOUNT+2                                          ;SUBTRACT AMT0
FROM TOTAL0
                        CLR C
                        SUBB A,PRICE+1
                        CALL BCD_CONV
                        MOV A,AMOUNT+1                                          ;SUBTRACT AMT1
FROM TOTAL1
                        SUBB A,PRICE
                        MOV A,AMOUNT
                        CLR TBIT
                        JNC POP1

                        SETB TBIT
POP1:           CJNE A,#00H,BACK
                        JNB TBIT, BACK
                        MOV BALANCE,#00H
                        MOV R1,#BALANCE                             ;store COUNT
                        MOV R4,#15H                                 ;Starting Address IN EEPROM
                        MOV R6,#1                                   ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        CLR RELAY
                        SETB BUZZER
BACK:           RET
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
BCD_CONV:
                        CLR BCDCARRY
                        CLR CARRY
                        JNC LOP2
                        SETB CARRY
LOP2:           JNB AC,LOP1
                        SETB BCDCARRY
                        CLR C
                        SUBB A,#06H
LOP1:           JNB CARRY,LOP3
                        CLR C
                        SUBB A,#60H
```

```
LOP3:           CLR C
                        JNB CARRY,LOP4
                        SETB C
LOP4:           RET
;#################################################
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
;                       READ PULSE COUNTER FROM MEMORY
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

READ_BALANCE:
                        MOV R1,#BALANCE                                         ;GET DATA IN
BYTES(RAM)
                        MOV R4,#15H                                             ;DATA ADDRESS IN
EEPROM
                        MOV R6,#1                                               ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        RET
READ_COUNTER:
                        MOV R1,#COUNTER                                         ;GET DATA IN
BYTES(RAM)
                        MOV R4,#0EH                                             ;DATA ADDRESS IN
EEPROM
                        MOV R6,#2                                               ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        RET
READ_PRICE:
                        MOV R1,#PRICE                                   ;GET DATA IN BYTES(RAM)
                        MOV R4,#10H                                             ;DATA ADDRESS IN
EEPROM
                        MOV R6,#2                                               ;NUMBER OF BYTES
                        CALL READ_EEPROM
                        RET
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
SEP_DISP1:
                        MOV A,AMOUNT
                        ANL A,#0F0H
                        SWAP A
                        CJNE A,#00H,DAP1
                        MOV A,AMOUNT
                        ANL A,#0FH
                        AJMP DAP3

DAP1:           ADD A,#30H                      ;BOTH NOT EQUAL TO ZERO
                        MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
DAP2:           MOV A,AMOUNT
                        ANL A,#0FH
                        ADD A,#30H
```

```
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
DAP4:  MOV A,AMOUNT+1
                     ANL A,#0F0H
                     SWAP A
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
DAP5:  MOV A,AMOUNT+1
                     ANL A,#0FH
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
    MOV R4,#'.'
    CALL WRLCDDATA
    CALL MDELAY
    MOV A,AMOUNT+2
                     ANL A,#0F0H
                     SWAP A
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
    MOV A,AMOUNT+2
                     ANL A,#0FH
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
    RET

DAP3:         CJNE A,#00H,DAP2                    ;CHK 2 DIGIT
                     MOV A,AMOUNT+1
                     ANL A,#0F0H
                     SWAP A
                     CJNE A,#00H,DAP4                   ;CHK 3 DIGIT
                     AJMP DAP5


SEP_DISP:
                     MOV A,TEMP
                     ANL A,#0F0H
                     SWAP A
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
    MOV A,TEMP
                     ANL A,#0FH
                     ADD A,#30H
                     MOV R4,A
    CALL WRLCDDATA
    CALL MDELAY
    RET
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
AMT_RECHARGE:
                     MOV READ_BYTE,#01H
                     MOV READ_BYTE+1,#00H
                     MOV READ_BYTE+2,#10H
                     MOV R1,#READ_BYTE                         ;store COUNT
                     MOV R6,#3                                 ;STORE 2 BYTES
                     MOV A,#WTCMD1          ;LOAD WRITE COMMAND
                     CALL OUTS              ;SEND IT
                     MOV A,#20H             ;GET LOW BYTE ADDRESS
                     CALL OUT               ;SEND IT
BXLP:   MOV A,@R1                 ;GET DATA
                     CALL OUT               ;SEND IT
                     INC R1                 ;INCREMENT DATA POINTER
                     DJNZ R6,BXLP    ;LOOP TILL DONE
                     CALL STOP              ;SEND STOP CONDITION
                     CALL DELAY
                     RET

STORE_UNIT_PRICE:
                     MOV READ_BYTE,#00H
                     MOV READ_BYTE+1,#01H
                     MOV READ_BYTE+2,#00H
                     MOV R1,#READ_BYTE                         ;store COUNT
                     MOV R6,#3                                 ;STORE 2 BYTES
                     MOV A,#WTCMD1          ;LOAD WRITE COMMAND
                     CALL OUTS              ;SEND IT
                     MOV A,#20H             ;GET LOW BYTE ADDRESS
                     CALL OUT               ;SEND IT
BALP:   MOV A,@R1                 ;GET DATA
                     CALL OUT               ;SEND IT
                     INC R1                 ;INCREMENT DATA POINTER
                     DJNZ R6,BALP    ;LOOP TILL DONE
                     CALL STOP              ;SEND STOP CONDITION
                     CALL DELAY
                     RET

RESET_BALANCE:
                     MOV BALANCE,#0FFH
                     MOV R1,#BALANCE                           ;store COUNT
                     MOV R4,#15H                               ;Starting Address IN EEPROM
```

```
                        MOV R6,#1                                     ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        RET
RESET_PRICE:
                 MOV PRICE,#02H
                 MOV PRICE+1,#00H
                 MOV R1,#PRICE                        ;store COUNT
                        MOV R4,#10H                                   ;Starting Address IN EEPROM
                        MOV R6,#2                                     ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        RET

RESET_COUNTER:
                 MOV COUNTER,#00H
                 MOV COUNTER+1,#10H
                 MOV R1,#COUNTER                              ;store COUNT
                        MOV R4,#0EH                                   ;Starting Address IN EEPROM
                        MOV R6,#2                                     ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        RET

RESET_AMT:
                        MOV AMOUNT,#00H  ;
                        MOV AMOUNT+1,#05H
                        MOV AMOUNT+2,#00H
                        MOV R1,#AMOUNT                                ;store READING
                        MOV R4,#0AH                                   ;Starting Address IN EEPROM
                        MOV R6,#3                                     ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        RET
RESET_READING:
                        MOV READING,#00H
                        MOV READING+1,#05H
                        MOV R1,#READING                       ;store READING
                        MOV R4,#05H                                   ;Starting Address IN EEPROM
                        MOV R6,#2                                     ;STORE 2 BYTES
                        CALL STORE_EEPROM
                        CALL DELAY
                        RET
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DELAYY:
                MOV F1,#0FH
SEP3:   MOV F2,#0fFH
SEP2:   MOV F3,#0FFH
SEP1:   DJNZ F3,SEP1
```

```
                    DJNZ F2,SEP2
                    CALL CARD_READ
                    MOV A,READ_BYTE
                    CJNE A,#0FFH,DSP1
                    CLR TBIT1
DSP3A:DJNZ F1,SEP3
                    RET


DELAYYS:
                    MOV F1,#0FH
S5P3:     MOV F2,#0fFH
S5P2:     MOV F3,#0FFH
S5P1:     DJNZ F3,S5P1
                    DJNZ F2,S5P2
                    DJNZ F1,S5P3
                    RET

DSP1:     JB TBIT1,DSP3A
                    CALL TITLE3
                    CALL DELAYS
                    CALL DELAYS
                    CALL CARD_READ
                    MOV A,READ_BYTE
                    CJNE A,#00H,DSP2
                    CALL TITLE4              ;              NEW UNIT PRICE
                    MOV PRICE,READ_BYTE+1
                    MOV PRICE+1,READ_BYTE+2
                    MOV R1,#PRICE                           ;store COUNT
                    MOV R4,#10H                                       ;Starting Address IN EEPROM
                    MOV R6,#2                                         ;STORE 2 BYTES
                    CALL STORE_EEPROM
                    CALL DELAYS
                    SETB TBIT1
                    AJMP RESETX_CHIP

DSP2: CJNE A,#01H,DSP3
                    CALL TITLE5              ;              NEW RECHARGE

                    MOV A,AMOUNT+1
                    ADDC A,READ_BYTE+2
                    DA A
                    MOV AMOUNT+1,A
                    MOV A,AMOUNT
                    ADD A,READ_BYTE+1
                    DA A
                    MOV AMOUNT,A

                    MOV R1,#AMOUNT                                    ;store READING
```

```
                MOV R4,#0AH                                     ;Starting Address IN EEPROM
                MOV R6,#03h                                     ;STORE 2 BYTES
                CALL STORE_EEPROM
                CALL DELAYS
                SETB TBIT1
                CALL RESET_BALANCE

RESETX_CHIP:
                        MOV READ_BYTE,#0AAH                     ;ERASE AMOUNT
                        MOV READ_BYTE+1,#0FFH
                        MOV READ_BYTE+2,#0FFH
                        MOV R1,#READ_BYTE                               ;store COUNT
                        MOV R6,#3                                       ;STORE 2 BYTES
                        MOV A,#WTCMD1                   ;LOAD WRITE COMMAND
                        CALL OUTS                       ;SEND IT
                        MOV A,#20H                      ;GET LOW BYTE ADDRESS
                        CALL OUT                        ;SEND IT
BBLP:   MOV A,@R1                       ;GET DATA
                        CALL OUT                        ;SEND IT
                        INC R1                          ;INCREMENT DATA POINTER
                        DJNZ R6,BBLP            ;LOOP TILL DONE
                        CALL STOP                       ;SEND STOP CONDITION
                        CALL DELAY
                        RET

DSP3:   CJNE A,#0AAH,DSP4
                        CALL TITLE6             ;               NEW RECHARGE
                        CALL DELAYS
                        SETB TBIT1
DSP4:           RET
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DELAY:
                MOV R6,#0FFH
RE1:    MOV R7,#0FFH
RE:     NOP
                DJNZ R7,RE
                DJNZ R6,RE1
                RET
;***********************************************************
CARD_READ:
                        MOV R1,#READ_BYTE                                       ;GET DATA IN
BYTES(RAM)
                        MOV R6,#3                                               ;NUMBER OF BYTES
                        MOV A,#WTCMD1                   ;LOAD WRITE COMMAND TO SEND ADDRESS
                        CALL OUTS                       ;SEND IT
                        MOV A,#20H                      ;GET LOW BYTE ADDRESS
                        CALL OUT                        ;SEND IT
                        MOV A,#RDCMD1                   ;LOAD READ COMMAND
                        CALL OUTS                       ;SEND IT
```

```
BXDLP:  CALL IN                         ;READ DATA
                    MOV @R1,a                   ;STORE DATA
                    INC R1                      ;INCREMENT DATA POINTER
                    DJNZ R6,AXLP          ;DECREMENT LOOP COUNTER
                    CALL STOP                   ;IF DONE, ISSUE STOP CONDITION
                    RET                               ;DONE, EXIT ROUTINE
AXLP:   CLR SDA1 ;NOT DONE, ISSUE ACK
                    SETB SCL1
                    NOP ;NOTE 1
                    NOP
                    NOP
                    NOP ;NOTE 2
                    NOP
                    CLR SCL1
                    JMP BXDLP ;CONTINUE WITH READS
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
;                   READ DATA FROM EEPROM
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
READ_EEPROM:
                    MOV A,#WTCMD                ;LOAD WRITE COMMAND TO SEND ADDRESS
                    CALL OUTS                   ;SEND IT
                    MOV A,R4                    ;GET LOW BYTE ADDRESS
                    CALL OUT                    ;SEND IT
                    MOV A,#RDCMD          ;LOAD READ COMMAND
                    CALL OUTS                   ;SEND IT
BRDLP:  CALL IN                         ;READ DATA
                    MOV @R1,a                   ;STORE DATA
                    INC R1                      ;INCREMENT DATA POINTER
                    DJNZ R6,AKLP          ;DECREMENT LOOP COUNTER
                    CALL STOP                   ;IF DONE, ISSUE STOP CONDITION
                    RET                               ;DONE, EXIT ROUTINE
AKLP:   CLR SDA1 ;NOT DONE, ISSUE ACK
                    SETB SCL1
                    NOP ;NOTE 1
                    NOP
                    NOP
                    NOP ;NOTE 2
                    NOP
                    CLR SCL1
                    JMP BRDLP ;CONTINUE WITH READS
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
;                         STORE DATA IN EEPROM
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
STORE_EEPROM:
                    MOV A,#WTCMD                ;LOAD WRITE COMMAND
```

```
                        CALL OUTS                       ;SEND IT
                        MOV A,R4                        ;GET LOW BYTE ADDRESS
                        CALL OUT                        ;SEND IT
BTLP:   MOV A,@R1                       ;GET DATA
                        CALL OUT                        ;SEND IT
                        INC R1                          ;INCREMENT DATA POINTER
                        DJNZ R6,BTLP            ;LOOP TILL DONE
                        CALL STOP                       ;SEND STOP CONDITION
                        RET
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%


;############################################################
;                       DISPLAY ROUTINES
;############################################################
TITLE1:
        MOV DPTR,#MSAG1
        CALL LCD_MSG
        RET
MSAG1:
        DB 1H,81H,'PREPAID SYSTEM',0C0H,'FOR ENERGY METER',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE11:
        MOV DPTR,#MSAG11
        CALL LCD_MSG
        RET
MSAG11:
        DB 1H,81H,'A PROJECT FOR',0C6H,'FIEM',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE12:
        MOV DPTR,#MSAG22
        CALL LCD_MSG
        RET
MSAG22:
        DB 1H,80H,'BY AATREYI BAL,',0C0H,'SAIKAT MAJUMDAR,',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE13:
        MOV DPTR,#MSAG33
        CALL LCD_MSG
        RET
MSAG33:
        DB 1H,82H,'ARINDAM BOSE',0C0H,'AND TANAYA BOSE',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DISP_READING:
        MOV DPTR,#MSAG2
        CALL LCD_MSG
        RET
MSAG2:
```

```
        DB 1H,82H,'METER READING',0C6H,00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
AMT_READING:
        MOV DPTR,#MSAG3
        CALL LCD_MSG
        RET
MSAG3:
        DB 1H,81H,'BALANCE AMOUNT',0C3H,'Rs.',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
COUNT_READING:
        MOV DPTR,#MSAG4
        CALL LCD_MSG
        RET
MSAG4:
        DB 1H,82H,'PULSE COUNT',0C6H,00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
UNIT_PRICE:
        MOV DPTR,#MSAG14
        CALL LCD_MSG
        RET
MSAG14:
        DB 1H,83H,'UNIT PRICE',0C4H,'Rs ',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RECHAGRE:
        MOV DPTR,#MSAG5
        CALL LCD_MSG
        RET
MSAG5:
        DB 1H,80H,'Please Recharge',0C2H,'your Account',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE3:
        MOV DPTR,#MSAG6
        CALL LCD_MSG
        RET
MSAG6:
        DB 1H,84H,'New Card',0C1H,'** DETECTED **',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE4:
        MOV DPTR,#MSAG7
        CALL LCD_MSG
        RET
MSAG7:
        DB 1H,81H,'NEW UNIT PRICE',0C1H,'** STORED **',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE5:
        MOV DPTR,#MSAG8
        CALL LCD_MSG
        RET
MSAG8:
```

```
            DB 1H,83H,'NEW AMOUNT',0C1H,'** RECHARGED **',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TITLE6:
            MOV DPTR,#MSAG9
            CALL LCD_MSG
            RET
MSAG9:
            DB 1H,82H,'INVALID CARD',0C0H,'****************',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SYSTEM_RESET:
            MOV DPTR,#MSAG91
            CALL LCD_MSG
            RET
MSAG91:
            DB 1H,80H,'System Restored',0C0H,'****************',00H
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;***********************************************************
; INITIALIZE THE LCD 4-BIT MODE
;***********************************************************
INITLCD4:
       CLR        LCD_RS     ; LCD REGISTER SELECT LINE
       CLR        LCD_E      ; ENABLE LINE
       MOV        R4, #CONFIG; FUNCTION SET - DATA BITS,
                           ; LINES, FONTS
       CALL       WRLCDCOM4
       MOV        R4, #ONDSP ; DISPLAY ON
       CALL       WRLCDCOM4
       MOV        R4, #ENTRYMODE ; SET ENTRY MODE
       CALL       WRLCDCOM4  ; INCREMENT CURSOR RIGHT, NO SHIFT
       MOV        R4, #CLRDSP; CLEAR DISPLAY, HOME CURSOR
       CALL       WRLCDCOM4
       RET
; **********************************************************
; SOFTWARE VERSION OF THE POWER ON RESET
; **********************************************************
RESETLCD4:
       CLR        LCD_RS     ; LCD REGISTER SELECT LINE
       CLR        LCD_E      ; ENABLE LINE
       CLR        LCD_DB7    ; SET BIT PATTERN FOR...
       CLR        LCD_DB6    ; ... POWER-ON-RESET
       SETB       LCD_DB5
       SETB       LCD_DB4
       SETB       LCD_E      ; START ENABLE PULSE
       CLR        LCD_E      ; END ENABLE PULSE
       MOV        A, #4      ; DELAY 4 MILLISECONDS
       CALL       MDELAY
       SETB       LCD_E      ; START ENABLE PULSE
       CLR        LCD_E      ; END ENABLE PULSE
       MOV        A, #1      ; DELAY 1 MILLISECOND
```

```
        CALL      MDELAY
        SETB       LCD_E      ; START ENABLE PULSE
        CLR        LCD_E      ; END ENABLE PULSE
        MOV        A, #1      ; DELAY 1 MILLISECOND
        CALL      MDELAY
        CLR        LCD_DB4    ; SPECIFY 4-BIT OPERATION
        SETB       LCD_E      ; START ENABLE PULSE
        CLR        LCD_E      ; END ENABLE PULSE
        MOV        A, #1      ; DELAY 1 MILLISECOND
        CALL      MDELAY
        MOV        R4, #CONFIG; FUNCTION SET
        CALL      WRLCDCOM4
        MOV        R4, #08H   ; DISPLAY OFF
        CALL      WRLCDCOM4
        MOV        R4, #1     ; CLEAR DISPLAY, HOME CURSOR
        CALL      WRLCDCOM4
        MOV        R4,#ENTRYMODE  ; SET ENTRY MODE
        ACALL      WRLCDCOM4
                JMP INITLCD4

; *********************************************************
; SUB RECEIVES A COMMAND WORD TO THE LCD
; COMMAND MUST BE PLACED IN R4 BY CALLING PROGRAM
; *********************************************************
WRLCDCOM4:
        CLR        LCD_E
        CLR        LCD_RS     ; SELECT READ COMMAND
        PUSH       ACC        ; SAVE ACCUMULATOR
        MOV        A, R4      ; PUT DATA BYTE IN ACC
        MOV        C, ACC.4   ; LOAD HIGH NIBBLE ON DATA BUS
        MOV        LCD_DB4, C ; ONE BIT AT A TIME USING...
        MOV        C, ACC.5   ; BIT MOVE OPERATOINS
        MOV        LCD_DB5, C
        MOV        C, ACC.6
        MOV        LCD_DB6, C
        MOV        C, ACC.7
        MOV        LCD_DB7, C
        SETB       LCD_E      ; PULSE THE ENABLE LINE
        CLR        LCD_E
        MOV        C, ACC.0   ; SIMILARLY, LOAD LOW NIBBLE
        MOV        LCD_DB4, C
        MOV        C, ACC.1
        MOV        LCD_DB5, C
        MOV        C, ACC.2
        MOV        LCD_DB6, C
        MOV        C, ACC.3
        MOV        LCD_DB7, C
        CLR        LCD_E
        SETB       LCD_E      ; PULSE THE ENABLE LINE
        CLR        LCD_E
        CALL MADELAY
        POP        ACC
        RET
; **********************************************************
; SUB TO RECEIVE A DATA WORD TO THE LCD
; DATA MUST BE PLACED IN R4 BY CALLING PROGRAM
; **********************************************************
WRLCDDATA:
        CLR        LCD_E
        SETB       LCD_RS     ; SELECT READ DATA
           PUSH       ACC       ; SAVE ACCUMULATOR
        MOV        A, R4      ; PUT DATA BYTE IN ACC
        MOV        C, ACC.4   ; LOAD HIGH NIBBLE ON DATA BUS
        MOV        LCD_DB4, C ; ONE BIT AT A TIME USING...
        MOV        C, ACC.5   ; BIT MOVE OPERATOINS
        MOV        LCD_DB5, C
        MOV        C, ACC.6
        MOV        LCD_DB6, C
        MOV        C, ACC.7
        MOV        LCD_DB7, C
        SETB       LCD_E      ; PULSE THE ENABLE LINE
        CLR        LCD_E
        MOV        C, ACC.0   ; SIMILARLY, LOAD LOW NIBBLE
        MOV        LCD_DB4, C
        MOV        C, ACC.1
        MOV        LCD_DB5, C
        MOV        C, ACC.2
        MOV        LCD_DB6, C
        MOV        C, ACC.3
        MOV        LCD_DB7, C
        CLR        LCD_E
        SETB       LCD_E      ; PULSE THE ENABLE LINE
        CLR        LCD_E
        NOP
        NOP
        POP        ACC
        RET

; **********************************************************
; SUB TAKES THE STRING IMMEDIATELY FOLLOWING THE CALL AND
; DISPLAYS ON THE LCD. STRING MUST BE TERMINATED WITH A
; NULL (0).
; **********************************************************
LCD_MSG:
              CLR A                        ; Clear Index
              MOVC A,@A+DPTR                      ; Get byte pointed by Dptr
              INC DPTR                            ; Point to the next byte
              JZ LCD_Msg9                  ; Return if found the zero (end of stringz)
```

```
      CJNE A,#01H,Lcd_Msg1           ; Check if is a Clear Command
            MOV R4,A
            CALL WRLCDCOM4           ;If yes, RECEIVE it as command to LCD
            JMP   LCD_MSG            ;Go get next byte from stringz

Lcd_Msg1: CJNE A,#0FFH,FLL           ;Check for displaying full character
            MOV R4,A
            CALL WRLCDDATA
            JMP LCD_MSG
 FLL:       CJNE  A,#080h,$+3        ; Data or Address?  If => 80h then is address.
            JC    Lcd_Msg_Data       ; Carry will be set if A < 80h (Data)
            MOV R4,A
            CALL  WRLCDCOM4        ; Carry not set if A=>80, it is address
            JMP Lcd_Msg              ; Go get next byte from stringz

Lcd_Msg_Data:            ;
            MOV R4,A
            CALL WRLCDDATA           ; It was data, RECEIVE it to Lcd
            JMP  Lcd_Msg                         ; Go get next byte from stringz
Lcd_Msg9:

            RET                  ; Return to Caller

; ***********************************************************
; 1 MILLISECOND DELAY ROUTINE
; ***********************************************************

MDELAY:
       PUSH       ACC
       MOV        A,#0A6H
MD_OLP:
       INC        A
       NOP
       NOP
       NOP
       NOP
       NOP
       NOP
       NOP
       NOP
       JNZ        MD_OLP
       NOP
       POP        ACC
       RET
MADELAY:
       PUSH       ACC
       MOV        A,#036H
MAD_OLP:
       INC        A
```

```
     NOP
     NOP
     NOP
     NOP
     NOP
     NOP
     NOP
     NOP
     JNZ      MAD_OLP
     NOP
     POP       ACC
     RET
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DELAYS:                 ;One second delay routine
 MOV R6, #00H             ;put 0 in register R6 (R6 = 0)
  MOV R5, #04H            ;put 5 in register R5 (R5 = 4)
LOOPB:
  INC R6                 ;increase R6 by one (R6 = R6 +1)
  ACALL DELAYMS            ;call the routine above. It will run and return to here.
  MOV A, R6               ;move value in R6 to A
  JNZ LOOPB               ;if A is not 0, go to LOOPB
 DEC R5                  ;decrease R5 by one. (R5 = R5 -1)
 MOV A, R5               ;move value in R5 to A
  JNZ LOOPB               ;if A is not 0 then go to LOOPB.
 RET
;**************************************************************************
DELAYMS:                 ;millisecond delay routine
;                 ;
  MOV R7,#00H              ;put value of 0 in register R7
LOOPA:
  INC R7                 ;increase R7 by one (R7 = R7 +1)
  MOV A,R7                ;move value in R7 to Accumlator (also known as A)
  CJNE A,#0FFH,LOOPA        ;compare A to FF hex (256). If not equal go to LOOPA
  RET                   ;return to the point that this routine was called from
;**************************************************************************

;***********************************************************************
; THIS ROUTINE SENDS OUT CONTENTS OF THE ACCUMULATOR
; to the EEPROM and includes START condition.  Refer to the data sheets
; for discussion of START and STOP conditions.
;***********************************************************************

OUTS:   MOV     R2,#8          ;LOOP COUNT -- EQUAL TO BIT COUNT
    SETB   SDA1          ;INSURE DATA IS HI
    SETB    SCL1          ;INSURE CLOCK IS HI
    NOP                  ;NOTE 1
    NOP
    NOP
    CLR     SDA1          ;START CONDITION -- DATA = 0
```

```
      NOP                ;NOTE 1
      NOP
      NOP
      CLR    SCL1          ;CLOCK = 0
OTSLP:  RLC    A            ;SHIFT BIT
      JNC    BITLS
      SETB   SDA1          ;DATA = 1
      JMP    OTSL1         ;CONTINUE
BITLS:  CLR    SDA1          ;DATA = 0
OTSL1:  SETB   SCL1          ;CLOCK HI
      NOP                ;NOTE 1
      NOP
      NOP

      CLR    SCL1          ;CLOCK LOW
      DJNZ   R2,OTSLP      ;DECREMENT COUNTER
      SETB   SDA1          ;TURN PIN INTO INPUT
      NOP                ;NOTE 1

      SETB   SCL1          ;CLOCK ACK
      NOP                ;NOTE 1
      NOP
      NOP

      CLR    SCL1
      RET

;**********************************************************************
; THIS ROUTINE SENDS OUT CONTENTS OF ACCUMLATOR TO EEPROM
; without sending a START condition.
;**********************************************************************

OUT:   MOV    R2,#8         ;LOOP COUNT -- EQUAL TO BIT COUNT
OTLP:  RLC    A            ;SHIFT BIT
      JNC    BITL
      SETB   SDA1          ;DATA = 1
      JMP    OTL1          ;CONTINUE
BITL:  CLR    SDA1          ;DATA = 0
OTL1:  SETB   SCL1          ;CLOCK HI
      NOP                ;NOTE 1
      NOP
      NOP

      CLR    SCL1          ;CLOCK LOW
      DJNZ   R2,OTLP       ;DECREMENT COUNTER
      SETB   SDA1          ;TURN PIN INTO INPUT
      NOP                ;NOTE 1

      SETB   SCL1          ;CLOCK ACK
```

```
       NOP                  ;NOTE 1
       NOP
       NOP

       CLR     SCL1
       RET


STOP:   CLR     SDA1           ;STOP CONDITION SET DATA LOW
       NOP                  ;NOTE 1
       NOP
       NOP

       SETB    SCL1           ;SET CLOCK HI
       NOP                  ;NOTE 1
       NOP
       NOP

       SETB    SDA1           ;SET DATA HIGH
       RET
;******************************************************************
; THIS ROUTINE READS A BYTE OF DATA FROM EEPROM
; From EEPROM current address pointer.
; Returns the data byte in R1
;******************************************************************
CREAD:  MOV     A,#RDCMD        ;LOAD READ COMMAND
       CALL    OUTS           ;SEND IT
       CALL    IN             ;READ DATA
       MOV     R1,A           ;STORE DATA
       CALL    STOP           ;SEND STOP CONDITION
       RET

;*********************************************************************
; THIS ROUTINE READS IN A BYTE FROM THE EEPROM
; and stores it in the accumulator
;*********************************************************************

IN:     MOV     R2,#8          ;LOOP COUNT
       SETB    SDA1           ;SET DATA BIT HIGH FOR INPUT
INLP:   CLR     SCL1           ;CLOCK LOW
       NOP                  ;NOTE 1
       NOP
       NOP
       NOP

       SETB    SCL1           ;CLOCK HIGH
       CLR     C              ;CLEAR CARRY
       JNB     SDA1,INL1      ;JUMP IF DATA = 0
       CPL     C              ;SET CARRY IF DATA = 1
```

```
INL1:   RLC     A              ;ROTATE DATA INTO ACCUMULATOR
       DJNZ    R2,INLP         ;DECREMENT COUNTER
       CLR     SCL1           ;CLOCK LOW
       RET

;***********************************************************************
; This routine test for WRITE DONE condition
; by testing for an ACK.
; This routine can be run as soon as a STOP condition
; has been generated after the last data byte has been sent
; to the EEPROM. The routine loops until an ACK is received from
; the EEPROM. No ACK will be received until the EEPROM is done with
; the write operation.
;***********************************************************************
ACKTST: MOV     A,#WTCMD        ;LOAD WRITE COMMAND TO SEND ADDRESS
       MOV     R2,#8          ;LOOP COUNT -- EQUAL TO BIT COUNT
       CLR     SDA1           ;START CONDITION -- DATA = 0
       NOP                  ;NOTE 1
       NOP
       NOP

       CLR     SCL1           ;CLOCK = 0
AKTLP:  RLC     A              ;SHIFT BIT
       JNC     AKTLS
       SETB    SDA1           ;DATA = 1
       JMP     AKTL1          ;CONTINUE
AKTLS:  CLR     SDA1           ;DATA = 0
AKTL1:  SETB    SCL1           ;CLOCK HI
       NOP                  ;NOTE 1
       NOP
       NOP

       CLR     SCL1           ;CLOCK LOW
       DJNZ    R2,AKTLP        ;DECREMENT COUNTER
       SETB    SDA1           ;TURN PIN INTO INPUT
       NOP                  ;NOTE 1

       SETB    SCL1           ;CLOCK ACK
       NOP                  ;NOTE 1
       NOP
       NOP

       JNB     SDA1,EXIT       ;EXIT IF ACK (WRITE DONE)
       JMP     ACKTST         ;START OVER
EXIT:   CLR     SCL1           ;CLOCK LOW
       CLR     SDA1           ;DATA LOW
       NOP                  ;NOTE 1
       NOP
       NOP
```

```
      SETB    SCL1           ;CLOCK HIGH
      NOP
      NOP
      SETB    SDA1           ;STOP CONDITION
      RET
;********************************************************************
 END
```

B. Card Programmer Program

```
SDA1 EQU P3.4    ;SDA=PIN5
SCL1 EQU P3.3    ;SCL=PIN6
WTCMD   EQU     10100000B       ;WRITE DATA COMMAND Note 3
RDCMD   EQU     10100001B       ;READ DATA COMMAND Note 3


RED       EQU       P3.7
GREEN     EQU       P1.0

KEYS      EQU       P1

ROW1      EQU       P1.1
ROW2      EQU       P1.2
ROW3      EQU       P1.3
ROW4      EQU       P1.4
COL1      EQU       P1.7
COL2      EQU       P1.6
COL3      EQU       P1.5


DSEG          ; This is internal data memory
ORG     20H     ; Bit adressable memory
KEY:      DS        1
N0:       DS        1
N1:       DS        1
N2:       DS        1
N3:       DS        1
N4:       DS        1
N5:       DS        1


COUNT:  DS        1
PASS0:  DS        1
PASS1:  DS        1
PASS2:  DS        1
CHANGE:             DS        1
CSEG          ; Code begins here

;  ---------==========----------=========---------========---------
;  Main routine. Program execution starts here. 8889
;  ---------==========----------=========---------========---------
                    ORG     00H    ; Reset

                    MOV SP,#60H


                    CLR RED
                    CLR GREEN
```

```
                        CALL DELAY
                        CALL DELAY
                        SETB RED
                        SETB GREEN

                        MOV N1,#01H
                        MOV N2,#0FFH
                        MOV N3,#0FFH
                        MOV N4,#0FFH
                        MOV N5,#0FFH


                        MOV R3,#01H
;                       MOV N2,#23H
;                       MOV N4,#45H
;                       CALL SAX

KEYBOARD:
            MOV KEY,#00H
            SETB COL1
            SETB COL2
            SETB COL3
K11:        CLR ROW1
            CLR ROW2
            CLR ROW3
            CLR ROW4
            MOV A,KEYS
            ANL A,#11100000B
            CJNE A,#11100000B,K11               ;check till all keys released
K2:         ACALL DEALAY            ;call 20 msec delay
            MOV A,KEYS                          ;see if any key is pressed
            ANL A,#11100000B                    ;mask unused bits
            CJNE A,#11100000B,OVER              ;key pressed, await closure
            SJMP K2
OVER:       ACALL DEALAY
            MOV A,KEYS
            ANL A,#11100000B
            CJNE A,#11100000B,OVER1
            SJMP K2
OVER1:      MOV A,KEYS
            ORL A,#11111110B
            MOV KEYS,A
            CLR ROW1
            MOV A,KEYS
            ANL A,#11100000B
            CJNE A,#11100000B,ROW_1
            MOV A,KEYS
            ORL A,#11111110B
            MOV KEYS,A
```

```
        CLR ROW2
        MOV A,KEYS
        ANL A,#11100000B
        CJNE A,#11100000B,ROW_2
        MOV A,KEYS
        ORL A,#11111110B
        MOV KEYS,A
        CLR ROW3
        MOV A,KEYS
        ANL A,#11100000B
        CJNE A,#11100000B,ROW_3
        MOV A,KEYS
        ORL A,#11111110B
        MOV KEYS,A
        CLR ROW4
        MOV A,KEYS
        ANL A,#11100000B
        CJNE A,#11100000B,ROW_4
        LJMP K2


ROW_1:  RLC A
        JC MAT1
        MOV KEY,#01H
        AJMP K1
MAT1:   RLC A
        JC MAT2
        MOV KEY,#02H
        AJMP K1
MAT2:   RLC A
        JC K1
        MOV KEY,#03H
        AJMP K1

ROW_2:  RLC A
        JC MAT3
        MOV KEY,#04H
        AJMP K1
MAT3:   RLC A
        JC MAT4
        MOV KEY,#05H
        AJMP K1
MAT4:   RLC A
        JC K1
        MOV KEY,#06H
        AJMP K1

ROW_3:  RLC A
        JC MAT5
        MOV KEY,#07H
        AJMP K1
MAT5:   RLC A
        JC MAT6
        MOV KEY,#08H
        AJMP K1
MAT6:   RLC A
        JC K1
        MOV KEY,#09H
        AJMP K1

ROW_4:  RLC A
        JC MAT7
        MOV KEY,#0AH
        AJMP K1
MAT7:   RLC A
        JC MAT8
        MOV KEY,#00H                        ;for 0
        AJMP K1
MAT8:   RLC A
        JC K1
        MOV KEY,#0FH

K1:
                CLR RED
                CALL DELAY
                CALL DELAY
                SETB RED


                MOV A,KEY
                CJNE A,#0FH,G0




                CJNE R3,#07H,G0
                AJMP G8
G0:     CJNE R3,#01H,G11
                INC R3
                MOV N0,KEY
                AJMP KEYBOARD
G11:    CJNE R3,#02H,G1
                INC R3
                MOV N1,KEY
                AJMP KEYBOARD
G1:     CJNE R3,#03H,G2
                INC R3
                MOV N2,KEY
                AJMP KEYBOARD
```

```
G2:     CJNE R3,#04H,G3
                INC R3
                MOV N3,KEY
                AJMP KEYBOARD
G3:     CJNE R3,#05H,G4
                INC R3
                MOV N4,KEY
                AJMP KEYBOARD
G4:     CJNE R3,#06H,G5
                INC R3
                MOV N5,KEY
G5:     AJMP KEYBOARD

G8:

                MOV A,N2
                SWAP A
                ORL A,N3
                MOV N2,A                        ;HIGHER DIGITSS IN N2
                MOV A,N4
                SWAP A
                ORL A,N5
                MOV N3,A                        ;LOWER DISITS IN N3

                MOV A,N0
                CJNE A,#01H,STR_AMT

                MOV N1,#00H
                MOV R1,#N1                      ;store COUNT
                MOV R4,#20H                             ;Starting Address IN EEPROM
                MOV R6,#3                               ;STORE 2 BYTES
                CALL STORE_EEPROM

        CALL DELAY
        CALL DELAY
                AJMP CHK_DATA

BV1S:   AJMP BV1
STR_AMT:
                CJNE A,#02H,BV1S

                MOV N1,#01H
                MOV R1,#N1                      ;store COUNT
                MOV R4,#20H                             ;Starting Address IN EEPROM
                MOV R6,#3                               ;STORE 2 BYTES
                CALL STORE_EEPROM
        CALL DELAY
        CALL DELAY
```

```
;  ---------==========----------=========---------========---------
;CHECK WITH DATA STORED IN MEMORY
;  ---------==========----------=========---------========---------
CHK_DATA:

                MOV R1,#PASS0                                   ;GET DATA IN BYTES(RAM)
                MOV R4,#20H                                             ;DATA ADDRESS IN EEPROM
                MOV R6,#3                                               ;NUMBER OF BYTES
                CALL READ_EEPROM

                MOV A,N1
                CJNE A,PASS0, BV1
                MOV A,N2
                CJNE A,PASS1,BV1
                MOV A,N3
                CJNE A,PASS2,BV1
        CLR GREEN
        CALL DELAY1
        CALL DELAY1
        SETB GREEN
        CALL DELAY1
        CALL DELAY1
        CLR GREEN
        CALL DELAY1
        CALL DELAY1
        SETB GREEN
                MOV R3,#01H
                MOV N0,#0FFH
                MOV N1,#0FFH
                MOV N2,#0FFH
                MOV N3,#0FFH
                MOV N4,#0FFH
                MOV N5,#0FFH
        AJMP KEYBOARD

BV1:    CLR RED
        CALL DELAY1
        CALL DELAY1
        SETB RED
        CALL DELAY1
        CALL DELAY1
        CLR RED
        CALL DELAY1
        CALL DELAY1
        SETB RED
        MOV R3,#01H
        MOV N0,#0FFH
        MOV N1,#0FFH
                MOV N2,#0FFH
                MOV N3,#0FFH
                MOV N4,#0FFH
                MOV N5,#0FFH
        AJMP KEYBOARD

;(((((((((((((((((((((((((((((((
DEALAY:
                        MOV R1,#50
REPP2:  NOP
                        DJNZ R1,REPP2
                        RET
;(((((((((((((((((((((((((((((((
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
;                       READ DATA FROM EEPROM
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
READ_EEPROM:
                        MOV A,#WTCMD                    ;LOAD WRITE COMMAND TO SEND ADDRESS
                        CALL OUTS                       ;SEND IT
                        MOV A,R4                        ;GET LOW BYTE ADDRESS
                        CALL OUT                        ;SEND IT
                        MOV A,#RDCMD            ;LOAD READ COMMAND
                        CALL OUTS                       ;SEND IT
BRDLP:  CALL IN                         ;READ DATA
                        MOV @R1,a                       ;STORE DATA
                        INC R1                          ;INCREMENT DATA POINTER
                        DJNZ R6,AKLP            ;DECREMENT LOOP COUNTER
                        CALL STOP                       ;IF DONE, ISSUE STOP CONDITION
                        RET                                     ;DONE, EXIT ROUTINE
AKLP:   CLR SDA1 ;NOT DONE, ISSUE ACK
                        SETB SCL1
                        NOP ;NOTE 1
                        NOP
                        NOP
                        NOP ;NOTE 2
                        NOP
                        CLR SCL1
                        JMP BRDLP ;CONTINUE WITH READS
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
;                               STORE DATA IN EEPROM
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
STORE_EEPROM:
                        MOV A,#WTCMD                    ;LOAD WRITE COMMAND
                        CALL OUTS                       ;SEND IT
                        MOV A,R4                        ;GET LOW BYTE ADDRESS
                        CALL OUT                        ;SEND IT
```

```
BTLP:      MOV A,@R1                              ;GET DATA
                                 CALL OUT                               ;SEND IT
                                 INC R1                                 ;INCREMENT DATA POINTER
                                 DJNZ R6,BTLP              ;LOOP TILL DONE
                                 CALL STOP                              ;SEND STOP CONDITION
                                 RET
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
;**********************************************************************
; THIS ROUTINE SENDS OUT CONTENTS OF THE ACCUMULATOR
; to the EEPROM and includes START condition.  Refer to the data sheets
; for discussion of START and STOP conditions.
;**********************************************************************

OUTS:   MOV     R2,#8          ;LOOP COUNT -- EQUAL TO BIT COUNT
      SETB   SDA1          ;INSURE DATA IS HI
      SETB    SCL1          ;INSURE CLOCK IS HI
      NOP                 ;NOTE 1
      NOP
      NOP
      CLR     SDA1          ;START CONDITION -- DATA = 0
      NOP                 ;NOTE 1
      NOP
      NOP
      CLR     SCL1          ;CLOCK = 0
OTSLP:  RLC     A             ;SHIFT BIT
      JNC     BITLS
      SETB    SDA1          ;DATA = 1
      JMP     OTSL1          ;CONTINUE
BITLS:  CLR     SDA1          ;DATA = 0
OTSL1:  SETB    SCL1          ;CLOCK HI
      NOP                 ;NOTE 1
      NOP
      NOP

      CLR     SCL1          ;CLOCK LOW
      DJNZ    R2,OTSLP       ;DECREMENT COUNTER
      SETB    SDA1          ;TURN PIN INTO INPUT
      NOP                 ;NOTE 1

      SETB    SCL1          ;CLOCK ACK
      NOP                 ;NOTE 1
      NOP
      NOP

      CLR     SCL1
      RET

;**********************************************************************
```

```
; THIS ROUTINE SENDS OUT CONTENTS OF ACCUMLATOR TO EEPROM
; without sending a START condition.
;*********************************************************************

OUT:    MOV     R2,#8           ;LOOP COUNT -- EQUAL TO BIT COUNT
OTLP:   RLC     A               ;SHIFT BIT
       JNC     BITL
       SETB    SDA1            ;DATA = 1
       JMP     OTL1            ;CONTINUE
BITL:   CLR     SDA1            ;DATA = 0
OTL1:   SETB    SCL1            ;CLOCK HI
       NOP                     ;NOTE 1
       NOP
       NOP

       CLR     SCL1            ;CLOCK LOW
       DJNZ    R2,OTLP         ;DECREMENT COUNTER
       SETB    SDA1            ;TURN PIN INTO INPUT
       NOP                     ;NOTE 1

       SETB    SCL1            ;CLOCK ACK
       NOP                     ;NOTE 1
       NOP
       NOP

       CLR     SCL1
       RET


STOP:   CLR     SDA1            ;STOP CONDITION SET DATA LOW
       NOP                     ;NOTE 1
       NOP
       NOP

       SETB    SCL1            ;SET CLOCK HI
       NOP                     ;NOTE 1
       NOP
       NOP

       SETB    SDA1            ;SET DATA HIGH
       RET
;******************************************************************
; THIS ROUTINE READS A BYTE OF DATA FROM EEPROM
; From EEPROM current address pointer.
; Returns the data byte in R1
;******************************************************************
CREAD:  MOV     A,#RDCMD        ;LOAD READ COMMAND
       CALL    OUTS            ;SEND IT
       CALL    IN              ;READ DATA
       MOV     R1,A           ;STORE DATA
       CALL    STOP           ;SEND STOP CONDITION
       RET

;********************************************************************
; THIS ROUTINE READS IN A BYTE FROM THE EEPROM
; and stores it in the accumulator
;********************************************************************

IN:     MOV     R2,#8          ;LOOP COUNT
       SETB    SDA1           ;SET DATA BIT HIGH FOR INPUT
INLP:   CLR     SCL1           ;CLOCK LOW
       NOP                    ;NOTE 1
       NOP
       NOP
       NOP

       SETB    SCL1           ;CLOCK HIGH
       CLR     C              ;CLEAR CARRY
       JNB     SDA1,INL1      ;JUMP IF DATA = 0
       CPL     C              ;SET CARRY IF DATA = 1
INL1:   RLC     A              ;ROTATE DATA INTO ACCUMULATOR
       DJNZ    R2,INLP        ;DECREMENT COUNTER
       CLR     SCL1           ;CLOCK LOW
       RET

;*******************************************************************
; This routine test for WRITE DONE condition
; by testing for an ACK.
; This routine can be run as soon as a STOP condition
; has been generated after the last data byte has been sent
; to the EEPROM. The routine loops until an ACK is received from
; the EEPROM. No ACK will be received until the EEPROM is done with
; the write operation.
;*******************************************************************
ACKTST: MOV     A,#WTCMD       ;LOAD WRITE COMMAND TO SEND ADDRESS
       MOV     R2,#8          ;LOOP COUNT -- EQUAL TO BIT COUNT
       CLR     SDA1           ;START CONDITION -- DATA = 0
       NOP                    ;NOTE 1
       NOP
       NOP

       CLR     SCL1           ;CLOCK = 0
AKTLP:  RLC     A              ;SHIFT BIT
       JNC     AKTLS
       SETB    SDA1           ;DATA = 1
       JMP     AKTL1          ;CONTINUE
AKTLS:  CLR     SDA1           ;DATA = 0
AKTL1:  SETB    SCL1           ;CLOCK HI
```

```
       NOP                   ;NOTE 1
       NOP
       NOP

       CLR     SCL1           ;CLOCK LOW
       DJNZ    R2,AKTLP        ;DECREMENT COUNTER
       SETB    SDA1            ;TURN PIN INTO INPUT
       NOP                   ;NOTE 1

       SETB    SCL1           ;CLOCK ACK
       NOP                   ;NOTE 1
       NOP
       NOP

       JNB     SDA1,EXIT       ;EXIT IF ACK (WRITE DONE)
       JMP     ACKTST          ;START OVER
EXIT:   CLR     SCL1           ;CLOCK LOW
       CLR     SDA1           ;DATA LOW
       NOP                   ;NOTE 1
       NOP
       NOP

       SETB    SCL1           ;CLOCK HIGH
       NOP
       NOP
       SETB    SDA1            ;STOP CONDITION
       RET
;********************************************************************

DELAY:   MOV R0,#0FFH
INLOP:   MOV R1,#0FFH
                         DJNZ R1,$
                         DJNZ R0,INLOP
                         RET

DELAY1: MOV R0,#0FFH
INLOP1: MOV R1,#0FFH
                         DJNZ R1,$
                         DJNZ R0,INLOP1
                         RET


END
```