CONTENTS

1. INTRODUCTION

2. INTRODUCTION TOMICROCONTROLLER

3. FUNCTIONAL BLOCK DIAGRAM

4. FUNCTIONAL DESCRIPTION

5. HARDWARE

   * Power supply
   * ADC 0808
   * APR 9600
   * Amplifier for CT HTS 10P
   * LCD
   * Serial EEPROM 24C04

6. SOFTWARE

   * Flow Chart
   * Program
INTRODUCTION

Smart card enhanced in use today for several applications like mobile communication, banking etc., so in this project an attempt is made to have an smart card for electric power consumption. Here also the user has to purchase the smart card of particular units/amount of power by paying the amount initially. Then in order to consume the power the user has to insert the card, once the amount of power consumed by the user crosses the unit/amount as stored initially in the Corel, then automatically the system will disconnect the power line from the loads.
FUNCTIONAL DESCRIPTION
**Functional Descriptions of KPTCL**

**Smart Card (serial EEPROM 24C04)**
The serial EEPROM 24C04 is used here as a smart card to store the number of units. Micro controller communicates with serial EEPROM using I2C protocol. Once the system is switched ON, the micro controller first checks the content of serial EEPROM for units. If there is no EEPROM or there is no units. If there is no EEPROM or there is no units that is zero units in the EEPROM, controller will not proceed further.

**Relay Driver:**
This section is used to connect mains to the loads by driving the relay which is connected in series with the phase of the AC mains.
The micro controller drives the relay when there is some units in the inserted card. And releases the relay when the units are over.

**Current Processing section:**

Once the mains is connected the loads, then the loads start drawing current, this current taken by the loads is extracted with the help of current transducer HTS 10 P which is connected in series with the phase of the AC mains. The out of the transducer is a current signal of very low voltage (will be in terms of mile volts), which is amplified in two stages using lm 348 amplifier with the gain of 25. Then the amplified using diode OA79. and the rectified DC voltage of current signal is applied to the first channel of ADC 0809.
Analog to Digital Converter (ADC 0809)

ADC 0809 is used to convert the current input signal, and then sends start of conversion signal to the ADC, then ADC will convert the input analog voltage into 8-bit digital value, and then selects the other channel to get the voltage equivalent hex value. Once the micro controller gets the digital value of current and voltage, then it calculates the power that is $P=VI$. After one unit, the controller decrements the content of serial EEPROM and then compares with the 50% is matched it will give be continued, offer 75% is completed it will give another indication and playback circuit APR 9600. Once the unit become zero, the system informs by playing another message and another indication and disconnects the loads from the mains by releasing the Relay. Then after inserting the fresh card the above procedure will be continued.

Features:
* continuous display of unit and other information on 2 line LCD.
* If the card is removed in between, and after inserting again the meter reading starts from the previous balance.
* Percentage indication to know the balance.
* Voice message for the balance.

INTRODUCTION TO MICROCONTROLLER
8951/8051 MICROCONTROLLER:

Computer in its simplest form needs at least three basic blocks: the central processing unit (CPU). Input-output (I/O) and memory (RAM/ROM). The integrated form of CPU is the microprocessor. As the use of microprocessors in control applications increased, development of microcontroller unit or MCU took shape, wherein CPU, I/O and some limited memory on a single chip was fabricated. Intention was to reduce the chip count as much as possible. There are many types of microcontrollers currently in the market, out of which the 8031. Family from Intel, and second soured by many others, have gained immense popularity.

INTRODUCTION

Looking back into the history of microcomputers, one would at first come across the development of microprocessor, i.e., the processing element, and later on the peripheral devices. The three basic elements –the CPU, I/O devices and memory- have developed in distinct direction. While the CPU has been the proprietary item, the memory devices fall into general-purpose category and the I/O devices may be grouped somewhere in-between. The control applications of microprocessors have different requirements, both hardware-wise as well as software-wise. Whereas microprocessor has just sufficient number of on-chip devices to act as the CPU, a number of other auxiliary devices are needed to get a working microcontroller. Integration of I/O and memory with the CPU on a single chip ushered in the era of development of a new class of devices, i.e., the single chip microcontrollers. Now only one device was needed to run an independent control application. The 8048 from Intel, Z80 from Zilog, 6805 and 6811 from Motorola represent a few of the members of this large family. The
8048 from Intel became popular due to its use in the keyboard of the IBM PC.
The family of second generation microcontrollers from Intel, the 8051 and other related devices, has brought about a new revolution in this field. While the early microcontrollers had only limited memory and existent serial I/O capability, the 8051 provides for 4k PROM/ROM, 128 byte RAM and 32 I/O lines. It also includes a universal asynchronous receive-transmit (UART) device, two 16-bit timer/counter and elaborate Interrupt logic. Lack of multiply and divide instructions, has also been taken care of in the 8051. Thus the 8051 may be called nearly equivalent of the following devices on a single chip:

8085 + 8255 + 8251 + 8253 + 2764 + 2764 + 6116
(Microprocessor) (PPI) (USART) (TIMER) (EPROM) (RAM)

In short, the 8051 has the following on-chip facilities:

- 4k ROM (EPROM on 8751)
- 128 byte RAM
- USRT
- 32 Input-output port lines
- TWO 16-bit timer/counter
- Six interrupt sources and
- On-chip clock oscillator and power–on–reset circuitry.

The other members of this family, such as 8053, have one extra timer/counter, 8k ROM/EPROM and 256 byte RAM, while 8031 and 8032 are corresponding ROM-less versions of 8052, respectively. All these are also available in CMOS versions. The 8051 family includes a large numbers of members from many manufacturers, some of which are listed below.

8051 with 4k ROM and its
8031 ROM-less version.
8052 with 8k ROM, one additional counter/timer and 256-bit RAM; its other version (8052h) comes with built-in BASIC interpreter.
8032 & 8752 are ROM + I/O (80C31) Along with pulse width modulator.
80C592 With 8-channel A/D converter, by Philips.
AT 89C51 CMOS device with EPROM (100 times programmable), by ATMEL.

SALIENT FEATURES
The 8051 can be configured to bypass the internal 4k RAM and run solely with external program memory. For this its external access ( ) pin 31 has to be grounded, which makes it equivalent to 8031. The program store enable
(PSEN) signal acts as read pulse for program memory. The data memory is external only and a separate RD* signal is available for reading its contents. Use of external memory requires that three of its 8-bit ports (out of four) are configured to provide data/address multiplexed bus, Hi address bus and control signals related to external memory use. The RXD and TXD ports of UART also appear on pins 10 and 11 of 8051 and 8031, respectively. One 8-bit port, which is bit addressable and extremely useful for control applications, still remains free for use.

The UART utilizes one of the internal timers for generation of baud rate. The crystal used for generation of CPU clock has therefore to be chosen carefully. The 3.579MHz crystal; available abundantly, can provide a baud rate of 1200.

The 256-byte address space is utilized by the internal RAM and special function registers (SFRs) array which is separate from external RAM space of 64k. the 00-7f space is occupied by the RAM and the 80-FF space by the SFRs. The 128-byte internal RAM has been utilized in the following fashion:
- 00-1F: Used for four banks of eight registers of 8-bit each. The four banks may be selected by software any time during the program.
- 20-2F: The 16 bytes may be used as 128 bits oriented programs.
- 30-7F: This area is used for temporary storage, pointers and stack. On reset, the stack starts at 08 and gets incremented during use.

The list of special function registers along with their hex addresses is given in table 1.

HARDWARE DETAILS

The on-chip oscillator of 8031 can be used to generate system clock. Depending upon version of the device, crystals from 3.5 to 12 MHz may be used for this purpose. The system clock is internally divided by 6 and the resultant time period becomes one processor cycle. The instructions take mostly one or two processor cycles. The ALE (address latch enable) pulse rate is 1/6th of the system clock, except during access of internal program memory, and thus can be used for timing purposes.

The two internal timers are wired to the system clock and persecuting factor is decided by the software apart from the count stored in the two bytes of the timer control registers. One of the counters, as mentioned earlier, is used for generation of baud rate clock for the UART. It would be of interest to point out that the 8052 has a third timer which is usually used for generation of baud rate.

The reset input is normally low and taking it high reset the microcontroller. In the present hardware, a separate CMOS circuit has been used for
generation of reset signal so that it could be used to drive external devices as well.

PIN DETAILS OF 8951

VSS
Circuit ground potential.
VCC
5-volt power supply input for normal operation and program verification.

PORT 0
Port 0 is an 8-bit open drain bi-directional input output port. It is also the multiplexed low ordered address and data bus when using external memory. It is used for data output during program verification. Port 0 can sink (and in bus operations can source) 8 LSTTL loads.

PORT 1
Port 1 is an 8-bit quasi bi-directional I/O port. It is also used for low order address byte during program verification. Port 1 can sink / source 4 LSTTL loads.

PORT 2
Port 2 is an 8-bit quasi bi-directional I/O port. It also emits the high order address byte when addressing external memory. It is used for the high order address and the control signals during program verification. Port 2 can sink / source 4 LSTTL loads.

PORT 3
Port 3 is an 8-bit quasi bi-directional I/O port with internal pull ups. It also serves the function of various special features of the MCS-51th Family as listed below:
Port pin Alternate function

P3.0 RXD (serial input port)
P3.1 TXD (serial input port)
P3.2 INTO (external interrupt)
P3.3 INT1 (external interrupt)

P3.4 TO (timer/counter 1 external input)
P3.5 T1 (timer/counter 1 external input)

P3.6 WR (external data memory write strobe)
P3.7 RD (external data memory read strobe)

The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. Port 3 can sink/source 4 LSTTL loads.

RST
A high on this pin for two-machine cycle while the oscillator is running rests the devices. A small external pull down resistor (=8.2 kilo ohms) from RST to VSS permits power on reset when a capacitor (=10 microfarad) is also connected from this pin to VCC.

ALE
Address latch enable output for latching the low byte of the during access to external memory. ALE is activated at a constant rate of 1/6 the oscillator frequency except during an external data memory access at which time one ALE pulse is skipped. ALE can sink/source 8 LSTTL inputs.

PSEN
The program store enable output for latching the low byte of the during access to external memory six oscillator periods except during external data memory access PSEN remains high during internal program memory. Do not float EA during normal operation.

XTAL 1
Input to the inverting amplifier that forms the part of the oscillator and input to the internal clock generator. XTAL2 receives the oscillator signal when an external oscillator used.

XTAL 2
Output of the inverting amplifier that forms the part of the oscillator and input to the interval clock generator. XTAL2 receives the oscillator signal when an external oscillator used.
HARDWARE
POWER SUPPLY

Main building block of any electronic system is the power supply to provide required power for their operation. For the microcontroller, audio amplifier, keyboard, edge connector +5V, required. And for driving the motor +12V. Is required. The power supply provides regulated output voltage of +5V, and non regulated output voltage +12V.

Three terminal IC 7805 meets the requirement of +5V regulated. The secondary voltage from the main transformer is rectified by diodes D1-D4.
and are filtered by capacitor C1. This unregulated dc voltage is supplied to input pin of regulator IC. C2 is an input bypass capacitor and C3 is to improve ripple rejection. The IC used are fixed regulator with internal short circuit current limiting and thermal shut down capability.

**ADC 0808:**

The analog input voltage to be fed to the ADC is derived from the output of the LF-398, sample and hold. The ADC has a conversion time period of 2 micro sec for a 500kHz clock signal and the output of ADC is fed to the port A of 8255 to be read by the microprocessor. The clock signal required by the ADC (at pin no. 10) is derived from an oscillator circuit build with NAND gates using IC 7400 along with passive components as shown in the figure. The selected frequency is 500khz for the ADC operation.
Assume frequency required is 500kHz, R = 1k ohm Hence,

\[
C = \frac{1}{500kHz \times 1kohm} = 0.2\text{micro farad}
\]
SIGNLE CHIP VOICE RECORDING & PLAY BACK DEVICE ’60 SECOND DURATION’

In this project we used APR 9600 voice chip as a voice recording and play back device. It has 60-sec duration.

Features:

1. single chip, high quality voice recording and play back solution
2. No. external I.C.s required
4. Non volatile flash memory technology.
5. No battery back up required.
6. User selectable messaging options.
7. Sequential access of multiples variable duration message.
8. User friendly. Easy to use operation
9. Programming and development systems not required.
10. Level activated recording and edge-activated play back
11. Level activated recording and edge-activated play back switches
12. Low power consumption
13. Operating current 25 milli amps
14. Stand by current 1 micro amp
15. Automatic power down
16. Chip enable for simple message expansion

General Description:
APR 9600 device offers true single chip voice recording, non volatile storage and play back capability for 40 to 60 seconds, the device supports both random and sequential access of multiple messages. Sample rates are user selectable, allowing deginers to customize their design for unique quality and storage time needs. Integrated output amplifier, microphone amplifier and AGC ckts greatly simplify system design, the device is ideal for use in portable voice recorders, toys and many other consumer and industrial application. A plus integrated achieves this high levels of storage technology implemented in an advanced flash non volatile memory process, where each memory cell can store 256 voltage levels. This technology enables the natural from. It eliminates the need for encoding compression, which often introduce distortion.

Functional Description:

The block diagram (architecture) of APR 9600 is as At the hand side of the diagram are the analog inputs. A differential microphone amplifier including integrated AGC is included on chip for application requiring its use. The amplified microphone signal is fed into the device by connecting the Ana_Out pin to Ana_In pin though and external DC blocking capacitor. Recording can be fed directly into the Ana_In pin though a DC blocking capacitor, however the connection between between Ana_In and Ana_Out is still required for play block. The next block encountered by the input signal is the internal analyzing filter. The filter automatically adjusts its response according to the sampling frequency selected so Shannon’s sampling theorem is satisfied. After anti aliasing filtering is accomplished the signal is ready to be clocked into memory array. This storage is accomplished though combination of the sample is accomplished though combination of the sample and hold circuit and the analog read/ write circuits. When play back is desired the previously stored recording is retrieved from memory low pass filtered and amplified as shown on the right band side of the diagram. The signal can be heard by connecting a speaker to the sp+ & sp- pins. Chip wide management is accomplished though the device control block shown in the upper right hand corner. Message management is controlled though the message control block represented in the lower center of the block diagram. play back and record operations are managed by on chip circuitry. There are several available messaging modes depending upon desired operation these message mode determine message management style, message length and external parts count. Therefore designer must select the appropriate operating mode.
<table>
<thead>
<tr>
<th>MODE</th>
<th>MSEL1</th>
<th>MSEL2</th>
<th>M8-OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 random access 2 Fixed duration message</td>
<td>0</td>
<td>1</td>
<td>Pull this pin to Vcc though 100kreg</td>
</tr>
<tr>
<td>2 random access 4 fixed duration message</td>
<td>1</td>
<td>0</td>
<td>Pull this pin to Vcc though 100kg</td>
</tr>
<tr>
<td>3 random access 8 fixed</td>
<td>1</td>
<td>1</td>
<td>Becomes the m 8 message trigpin</td>
</tr>
<tr>
<td>Tape mode, normal operation</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5 tape mode, auto rewind operation</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In this project we have selected the 5th mode of operation that is ‘tape mode, auto rewind operation’ because we need the message to be automatically rewind to the start of message every time.

PIN DIAGRAM OF APR 9600
PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>Pin 1: M1_message</th>
<th>: A low edge on this pin plays or records the current message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 2: M2_Next</td>
<td>:this active low input pin forces a skip to the next message for either play back or recording</td>
</tr>
<tr>
<td>Pin 3,4,5,6: M3 to M8</td>
<td>: These pins should not be connected when the in auto rewind mode.</td>
</tr>
<tr>
<td>Pin 7 : OSCR</td>
<td>: Oscillator register : this input allow an</td>
</tr>
<tr>
<td>Pin 8 : M7_ End</td>
<td>external registor to be connected to the tank circuit of the internal oscillation</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Pin 10:Busy</td>
<td>During the play back a low level on this pin indicates that all recorded message has been played.</td>
</tr>
<tr>
<td>Pin 11: BE</td>
<td>This pin indicates that the device is currently busying internal functions &amp; can neither record nor play back at the current time.</td>
</tr>
<tr>
<td>Pin 12: VssD:</td>
<td>If this pin is pull high, bees is enabled if it is low, beef is disabled.</td>
</tr>
<tr>
<td>Pin 13 VssA</td>
<td>Digital GND connection</td>
</tr>
<tr>
<td>Pin 14: Spt</td>
<td>Analog Gnd connection</td>
</tr>
<tr>
<td>Pin 15 Sp-</td>
<td>Positive output for speaker connection</td>
</tr>
<tr>
<td>Pin 16 VCCA</td>
<td>Negative output for speaker connection</td>
</tr>
<tr>
<td>Pin 17: Mic in</td>
<td>Analog output for speaker connection</td>
</tr>
<tr>
<td>Pin 18: Mic ref</td>
<td>Microphone input : should be connected to the input as outlined in the reference schematic</td>
</tr>
<tr>
<td>Pin 19 AGC</td>
<td>Automatic gain control attack time the time constant of the RC n/w connected to this input determined AGC attack time</td>
</tr>
<tr>
<td>Pin 21 Ana_out</td>
<td>An analog out this pin must be connected to ana_in though 0.1 micro farad.</td>
</tr>
<tr>
<td>Pin 22 strobe</td>
<td>This pin indicates programming of each individual recording of each individual recording segment. The falling edge regins the beginning of sector and raising edge and sector is half full.</td>
</tr>
<tr>
<td>Pin 23: CE: Chip enable</td>
<td>A low level in this pin enable the device for operation.</td>
</tr>
<tr>
<td>Pin 24 MSEL1: Mode Select 1</td>
<td>This pin in conjunction with MSEC2 &amp; M8 option sets record and play back operation mode.</td>
</tr>
<tr>
<td>Pin 25 MSEC2</td>
<td>This pin in conjunction with MSEL2 &amp; M8_option sets record and play back operation mode.</td>
</tr>
<tr>
<td>Pin 26 External clock</td>
<td>This clock can be used instead of the internal</td>
</tr>
</tbody>
</table>
clk for greater programming control and accuracy. When using the internal clock, this pin should be grounded.

<table>
<thead>
<tr>
<th>Pin 27 RE: Recorded Enable</th>
<th>This pin controls whether the device is in write or read mode. Logic high is read.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 28 VccD</td>
<td></td>
</tr>
</tbody>
</table>

**TAPE MODE USING AUTO REWIND OPTION:**

Functional Description of play back mode on power-up, the device is ready to record or play back.
Before the beginning of play back the CE input must be set to low to enable the device and RE must be set to high to disable recording and enable play back.
The first high to low going pulse of the M1 message pin initiates play back from the beginning of the current message. When the M1 message pin pulses low the second time, the play back of current message stops immediately. When the M1 message pin pulses low a third time, Play back of the current message starts again from its beginning. If you hold the M1-message pin low continuously the same message will play continuously in looping fashion.
A 1,530ms period of silence or delay is inserted during looping as an indicator to the user of the transition between the beginning and end of the message.
Note that in auto rewind mode, the device always rewinds to the beginning of the current message. To listen to a subsequent message the device must be fast-forwarded past the current message to the next message. This function is accomplished by toggling the M2 next pin from high to low. The pulse must be low for least 400 cycle of the sample clock. A special case exists when the M2_next pin goes low during play back. Play back of the current message will stop the device will beeps, advance to the next message and initiates play back of next message.
If the CE pin goes low during play back, play back of the current message will stop, the device will beeps, rest to the beginning of the first message and wait for a subsequent play back command.
When you reach the end of memory array, any subsequent pulsing of M1 message or M2 next will only result in double beeps.
To proceed from this state the user must rewind the device to the beginning of the memory array. This can be accomplished by toggling CE pin low or cycling power.
RECORDING MODE:

On power up, the device is ready to record or play back, starting at the first address in the memory array. To record, ICE must be set low to enable the device and IRE must be set low to enable recording. A falling edge of the [M1-message pin initiates voice recording (indicated by one beep). A subsequent was rising edge of the recording (also indicated by one beep). If the (M1-message pin is held low beyond the end of the available memory, recording will stop automatically (indicated by two beeps). The device will then assert logic low on the sample clock regardless of the state of the M1-message pin. The device returns to standby mode when the M1-message pin goes high again.

After recording is finished, the device will automatically rewind to the beginning of the most recently recorded message and wait for the next user input. The auto rewind function is convenient because it allows the user to immediately playback and review the message without need to rewind.
LIQUID CRYSTAL DISPLAY (LCD) MODULE

Frequently a AT89C51 program must interact with the outside world using input and output devices that communicate directly with a human being. One of the most common output devices used is a LCD. Some common LCD’s are 16x2 and 20x2 displays, which mean 16 characters per 2 line and 20 characters per lines, respectively.

Fortunately, Standards exist which allow us to communicate with vast majority of LCD. The Standard is referred to as HD44780U, which refer to the controller chip, which Receivers data from microcontroller and communicates directly with LCD.

HD44780U

The 44780 standard requires 3 control lines as 4 or 8 1/0 lines for the data bus the user may Select whether the LCD is to operate with 4-bit data bus or 8-data bus. The 3 control lines are EN, RS and RW.

The EN line is called called Enable. This control line is used to tell LCD that we are sending it Data. To send data the, program should first send High in this line and then set the other two Control line and put data on the data bus. When other lines are ready, EN should be made LOW.
The RS line is Register selector line. When RS is LOW, the data is to be treated as a Command or special instruction (such as CLEAR SCREEN, ETC). When RS is HIGH, the data being sent is text data that should be displayed on the screen.

The RW line is read/write control line. When it is LOW, the information on data bus is being written to LCD. When RW is HIGH, the program is effectively querying the LCD with the instruction Get LCD status.

A more robust method is to use “GET LCD STATUS” command to determine if the LCD is the last really use the LCD, must initialize and configure it. This is accomplished by sending a number of instructions to the LCD. The first instruction will be to specify whether we are using 4 or 8-line data bus. Sending a 38h command to the LCD dose this. Before we send the Command the RS line should be made low. We then send the 0Eh command to turn the LCD On. Lastly we send the 06h command so that every time we send a character the cursor automatically moves right.

NOTE: the LCD can be cleared using the 01h command.

Cursor Positioning

The 44780 contain a certain amount of memory, which is assigned to display. All Text we write to 44780 is stored in this memory, and the 44780 subsequently reads this Memory to display the text on LCD itself. This memory maps is shown below.

DISPLAY
In the above memory map, area up to 0F and 4F is the visible display. As one can see, it measures 16 characters per 2 lines. The numbers in each box in memory address that corresponds to that on screen.

Thus the “Set Cursor Position” instruction \textbf{80h} tell the LCD to position the cursor. Adding the cursor position to 80h does these sets the cursor to the required position on the screen.

\textbf{PIN Assignment:}

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\textbf{Vss}</td>
</tr>
<tr>
<td>2</td>
<td>\textbf{VDD}</td>
</tr>
<tr>
<td>3</td>
<td>\textbf{V0}</td>
</tr>
<tr>
<td>4</td>
<td>\textbf{RS}</td>
</tr>
<tr>
<td>5</td>
<td>\textbf{R/W}</td>
</tr>
<tr>
<td>6</td>
<td>\textbf{E}</td>
</tr>
<tr>
<td>7</td>
<td>\textbf{DB0}</td>
</tr>
<tr>
<td>8</td>
<td>\textbf{DB1}</td>
</tr>
<tr>
<td>9</td>
<td>\textbf{DB2}</td>
</tr>
<tr>
<td>10</td>
<td>\textbf{DB3}</td>
</tr>
<tr>
<td>11</td>
<td>\textbf{DB4}</td>
</tr>
<tr>
<td>12</td>
<td>\textbf{DB5}</td>
</tr>
<tr>
<td>13</td>
<td>\textbf{DB6}</td>
</tr>
<tr>
<td>14</td>
<td>\textbf{DB7}</td>
</tr>
<tr>
<td>15</td>
<td>\textbf{LED-(K)}</td>
</tr>
<tr>
<td>16</td>
<td>\textbf{LED-(A)}</td>
</tr>
</tbody>
</table>
LCD INITIALIZING BY INSTRUCTION

8-bit interface mode

POWER ON

WAIT FOR MORE THAN 30MS
AFTER VDD RISES TO 4.5V

FUNCTIONAL SET

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>N</td>
<td>F</td>
<td>X</td>
</tr>
</tbody>
</table>

WAIT FOR MORE THAN 39US

DISPLAY ON/OFF CONTROL

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>

WAIT FOR MORE THAN 39US

DISPLAY CLEAR

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

WAIT FOR MORE THAN 1.53MS
<table>
<thead>
<tr>
<th>ENTRY MODE SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**INITIALIZATION END**

condition: fosc=270khz

<table>
<thead>
<tr>
<th>N</th>
<th>0</th>
<th>1-LINE MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2-LINE MODE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
<th>0</th>
<th>DISPLAY OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>DISPLAY ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>0</th>
<th>DISPLAY OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>DISPLAY ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>0</th>
<th>CURSOR OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>CURSOR ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>0</th>
<th>BLINK OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>BLINK ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I/D</th>
<th>0</th>
<th>DECREMENT MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>INCREMENT MODE</td>
</tr>
</tbody>
</table>
4-bit interface mode

POWER ON

WAIT FOR MORE THAN 30 MS AFTER VDD RISES TO 4.5V

FUNCTION SET

WAIT FOR MORE THAN 39US

DISPLAY ON/OFF CONTROL

WAIT FOR MORE THAN 39US

ENTRY MODE SET

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>N</td>
<td>F</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>F</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RS</th>
<th>RW</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

WAIT FOR MORE THAN 1.53MS
Condition: \( f_{0sc} = 270\text{khz} \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>1</th>
<th>0</th>
<th>N</th>
<th>0</th>
<th>1-LINE MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>2-LINE MODE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>F</th>
<th>0</th>
<th>DISPLAY OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>DISPLAY ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>D</th>
<th>0</th>
<th>DISPLAY OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>DISPLAY ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>C</th>
<th>0</th>
<th>CURSOR OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>CURSOR ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>B</th>
<th>0</th>
<th>BLINK OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>BLINK ON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>I/D</th>
<th>0</th>
<th>DECREMENT MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>INCREMENT MODE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
<th>SH</th>
<th>0</th>
<th>ENTIRE SHIFT OFF</th>
</tr>
</thead>
</table>
CT Amplifier HTS 10P Circuit
SERIAL EEPROMS:

FEATURES

- Low power CMOS technology
  - maximum write current 3 ma at 5.5v
  - maximum read current 400micro amps at 5.5v
  - Standby current 100na typical at 5.5v
- 2-write serial interface bus, I²C compatible
- Cascadable for up to eight devices
- Self-timed ERASE/WRITE cycle
- 64-byte page-write mode available
- 5ms max write protect for entire array
- Hardware write protect for entire array
- Schmitt trigger inputs for noise suppression
- 100,000 erase/write cycles guaranteed
- Electrostatic discharge protection>4000v
- Data relation> 200 years
- 8-pins PDIP and SOIC (208ml) packages
- Temperature ranges:
  - Industrial (I) -40° c to +85° c
  - Automotive (E) - 40° c to +125° c
DESCRIPTION

The Microchip Technology Inc. 24AA256/24LC256 (24x256*) is a 32k x8 (256k bit) Serial Electrically Erasable PROM, capable of operation across a board voltage range (1.8v to 5.5v). it has been developed for advanced, low power applications such as personal communications or data acqision. This device also has a page-write capability of up to 64 bytes of data. This device is capable of both random and sequential reads allow up to 256k boundary. Functional address lines allow up to eight devices on the same bus, for up to 2 Mbit address space. This device is available in the standard 8-pin plastic DIP, and 8-pin SOIC (208 mil) packages.
PIN DESCRIPTIONS

A0, A1, A2 Chip Address Inputs
The A0, A1, A2 inputs are used by the 24x256 for multiple device operation. The levels on these inputs are compared with the corresponding bits in the slave address. The chip is selected if the compare is true. Up to eight devices may be connected to the same bus by using different chip select bit combinations. If left unconnected, these inputs will be pulled down internally to Vss.

SDA Serial Data

This is a bi-directional pin used to transfer addresses and data into and out of the device. It is an open-drain terminal, therefore, the SDA bus requires a pull up resistor to Vcc (typical 10 k ohm for 100 khz, 2k ohm for 400 khz)

SCL SERIAL CLOCK

This input is used to synchronize the data transfer from and to the device.

WP

This pin can be connected to either Vss, Vcc or left floating. An internal pull-down on this pin will keep the device in the unprotected state if left floating. If tied to Vss or left floating, normal memory operation is enabled (read/write the memory 0000-7FFF). If tied to Vcc, WRITE operations are inhibited. Read operation as are not affected.

FUNCTIONAL DESCRIPTION

The 24xx256 support a bi-directional 2-write bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter, and a device receiving data as a receiver. The bus must be controlled by a master device which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions while the 24xx256 works as a slave. Both master and slave can operate as a transmitter or receiver, but the master device determines which mode is activated.
BUS CHARACTERISTICS

The following bus protocol has been defined:

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH; changes in the data line while the clock line is high will be interpreted as a START or STOP condition.

Accordingly, the following bus conditions have been defined (Figure 4-1).

**Bus not Busy (A)**

Both data and clock lines remain HIGH.

**Start Data Transfer (B)**

A HIGH to LOW, transmission of the SDA line while the clock (SCL) is high determines a START condition. All commands must be preceded by a START condition.

**Data Valid (D)**

The state of the data represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal.

The data on the clock signal. There is one bit of data per clock pulse.

Each data transfer is initiated with a START and STOP conditions is determined by the master device.

**Acknowledge**

Each receiving device, when addressed, is obliged to generate an acknowledge signal after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.
SOFTWARE
FLOW CHART:

START

READ CARD

IS UNIT = 0

CONNECT MAINS TO & DISPLAY UNITS ON LCD

CALCULATE POWER

IS 1 UNIT

DECREMENT UNIT BY 1

IS 50% OF UNIT OVER

IS 75% OF UNIT OVER

IS UNIT ZERO

ON GREEN

OFF GREEN

ON YELLOW

ON RED & PLAY MESSAGE
PROGRAM
#include <reg51g.h>       /* special function
   register declarations */
   /* for the intended 8051
   derivative */
#include <stdio.h>         /* prototype
   declarations for I/O functions */

#include <stdlib.h>        /* standard library .h-
   file */
#include <ctype.h>         /* character functions
   */
#include <math.h>          /* math functions
   */

void WRITE_EEPROM(unsigned char , unsigned char, unsigned char);
unsigned char data1 = 0x00;

void delay()
{
    unsigned char i;
    for(i=0;i<100;i++); /* 25 */
}
void delay1()
{
    int k;
    WRITE_EEPROM(0x09, 0x00, data1);
    for(k=0;k<2000;k++); /* 25 */
}

void delay3()
{
    int k;
    // WRITE_EEPROM(0x04, 0x00, data1);
    for(k=0;k<2000;k++); /* 25 */
}

void delay2()
{
    int i;
    for(i=0;i<50;i++)
    {
        delay1();
    }
}

void Controller(unsigned char LCD_Cntr)
{
    P1=0x40;
P2=LCD_Cntr;
P1=0x00;
delay();
}

void Controller1(unsigned char LCD_Cntr)
{
P1=0x41;
P2=LCD_Cntr;
P1=0x01;
delay();
}

void Data_Display(unsigned char Data)
{
P1=0x50;
P2=Data;
P1=0x10;
delay();
}

void Data_Display1(unsigned char Data)
{
P1=0x51;
P2=Data;
P1=0x11;
delay();

}

void lcd_init()
{
    Controller(0x38);
    /***********************************************************************/
    delay();
    Controller(0x0c);    /* LCD control signal
data's   */
    delay();
    Controller(0x06);    /* 38,0e and 06 hex***********/
    delay();
    Controller(0x01);    /* clear's LCD
display(01h)   */
}

void title()
{
    unsigned char i;
    char str1[] = "PRE-PAID METER";
    Controller(0x00);
delay3();

Controller(0x81);
delay3();

for(i=0;i<14;i++)
{
    Data_Display(str1[i]);
}

void cardd()
{
    unsigned char i;
    char str3[] = "INSERT CARD...";
    Controller(0xC1);
    delay3();
    delay3();
    for(i=0;i<14;i++)
    {
        Data_Display(str3[i]);
    }
}
void s_out(unsigned char cw) {

}
unsigned char i, carry=0;
for(i=0; i<8; i++)
{
    cw=cw<<1;
    ACH3=CY;     // bitwise sending data
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=0;
}
ACH3=1;
ACH3=1;
ACH3=1;               // ACH3 is data
ACH2=1;                // ACH2 is clock
ACH2=1;
ACH2=1;
ACH2=1;
ACH2=1;
CY = ACH3;

for(i=0; i<100; i++)    // Checking for ack
{
    CY=ACH3;
    if(CY==0)
{
carry=1;
break;
}
}

if(carry==0)
{
ACH0=0;       //OFF RELAY
while(1);
}

ACH2=0;
}

s_in()
{
unsigned char rd=0x00,i;
ACH3=1;
ACH2=0;
for(i=0;i<8;i++)
{
ACH2=0;
ACH2=0;
ACH2=0;
ACH2=1;
ACH2=1;
ACH2=1;
CY = ACH3;
rd=(rd | CY);
if (i!=7)                      //shifting 4 7 bits
    rd=rd<<1;
ACH2=0x00;
}
return (rd);
}

void stop()
{
    ACH3=0;
    ACH3=0;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH3=1;
}

void NAK()
{
    ACH3=1;
    ACH3=1;
    ACH3=1;
    ACH2=1;
    ACH2=1;
void WRITE_EEPROM(unsigned char al,unsigned char ah,unsigned char d)
{
  ACH3=1;
  ACH2=1;
  CY=start();
  while(CY!=0) //start will return 0 wen d bus is free
    {
    CY=start();
    }
  s_out(0xa0); //a0 write cmd
  s_out(ah);
  s_out(al);
  s_out(d);
  stop();
  delay();
}
READ_EEPROM(unsigned char al,unsigned char ah)
{
    unsigned char epm;
    delay();

    CY=start();
    while(CY!=0)
    {
        CY=start();
    }

    s_out(0xa0);
    s_out(ah);
    s_out(al);

    CY=start();
    while(CY!=0)
    {
        CY=start();
    }

    s_out(0xa1);                   //a1 read cmd
    epm=s_in();

    NAK();
unsigned char CT_MAX()
{
    unsigned char j,temp,temp3=0x00;

    for(j=0;j<25;j++)
    {
        ACH1 = 1;
        delay();
        delay();
        ACH1 = 0;
        delay3();
        delay2();
    }
}

CY=0;
return (epm);
}
temp = P0;

if(temp > temp3)
    temp3 = temp;
    delay3();
}
return(temp3);
}

void conv_disp(unsigned char d1)
{
    unsigned char a,temp;
    unsigned char mes[]=" UNITS";
    int i;

    delay1();
    Controller(0xc3);
    delay1();
    if(d1<=0x09)
    {
        Data_Display('0');
        Data_Display('0');
        Data_Display(d1+0x30);
    }
}
else

    if(d1 <= 0x63)
        {
            B=0x0a;
            a=d1/B;

            Data_Display('0');
            Data_Display(a+0x30);
            Data_Display(B+0x30);
        }
    else
        {
            B=0x0a;
            a=d1/B;
            temp=B;
            B=0x0a;
            a=a/B;
            Data_Display(a+0x30);
            Data_Display(B+0x30);
            Data_Display(temp+0x30);
        }
}
for(i=0;i<6;i++)
    Data_Display(mes[i]);

}

void conv_disp1(unsigned char d1)
{
    unsigned char a,temp,mes2[]=" UNITS";
    int i;
    delay1();
    Controler1(0xc4);
    delay1();

    if(d1<=0x09)
    {
        Data_Display1('0');
        Data_Display1('0');
        Data_Display1(d1+0x30);
    }

    else

if(d1<=0x63)
{
    B=0x0a;
    a=d1/B;

    Data_Display1('0');
    Data_Display1(a+0x30);
    Data_Display1(B+0x30);
}
else
{
    B=0x0a;
    a=d1/B;
    temp=B;
    B=0x0a;
    a=a/B;
    Data_Display1(a+0x30);
    Data_Display1(B+0x30);
    Data_Display1(temp+0x30);
}

for(i=0;i<6;i++)
{
    Data_Display1(mes2[i]);
}
void main()
{
  unsigned char
  x,temp0,temp4,ah=0,al=0,data50=0x00,data75=0x00;
  unsigned char
  a,r,s,flag=0,Ack=1,cw,i,d1,d2,d3,d4,temp1,temp2;
  unsigned char mes3[]=" NO BALANCE";
  unsigned char mes4[]="BALANCE OVER";
  int j;

  // P0 -------> ADC
  RXD=1;       // APR CE
  TXD=1;       // Trk 1
INT0=1;       // Trk 2
INT1=0;       // LED 1
T0=0;         // LED 2
T1=0;         // LED 3
ACH0=0;       // Relay
ACH1=0;       // SOC
delay3();
delay3();
lcd_init();
delay3();
Controler(0x01);
delay3();
// coll();

Controler(0x01);
delay3();
title();
for(j = 0; j < 50; j++)
delay3();
cardd();
ACH2=1;
ACH3=1;       // data t0
CY=1;

while(Ack)
{ }

CY=start();
while(CY!=0)
 {
    CY=start();
    CY=start();
 }

cw=0xa0;

for(i=0;i<8;i++)
 {
    cw=cw<<1;
    ACH3=CY;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=0;
    ACH2=1;
    ACH2=1;
    ACH2=0;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    ACH2=1;
    }
ACH2=1;
ACH2=1;
CY = ACH3;

for(i=0;i<100;i++)
{
CY=ACH3;
if(CY==0)
{
    Ack=0;
    break;
}
}
ACH2=0;

data1=READ_EEPROM(al,ah);    //READ UNIT

while(data1==0x00)
{
    data1=READ_EEPROM(al,ah);
    Controler(0xc1);
delay1();

for(i=0;i<11;i++)
{
    Data_Display(mes3[i]);
}

delay2();
}

Controler(0x01);
delay1();
title();
delay2();
delay2();

Controler(0xc0);
delay1();

conv_disp(data1);
data50=data1/2;
data75=data1/4;
ACH0=1; //ON RELAY
while(1)
{
    WRITE_EEPROM(al,ah,data1);
    delay();
    data1=READ_EEPROM(al,ah);
    if(flag)
    {
        Controler1(0xc0);
        delay();
        temp0=data1;
        conv_disp1(data1);
    }
    else
    {
        Controler1(0xc0);
        delay();
        temp0=data1;
        conv_disp1(data1);
    }
}

if(data1==0)
{
    ACH0=0;                //Off load
    T1=0;                  //Off yellow
INT1=1;                  //on red
    delay1();
RXD=0;                     // CE
    delay1();
    delay1();
    delay1();

INT0=0;                    //on track 2
    Controler(0xc2);
    delay1();
    for(i=0;i<12;i++)
    {
        Data_Display(mes4[i]);
    }
    delay1();
    delay1();
    delay1();
    delay1();

    for(j=0;j<25;j++)
    {
        delay1();
        delay();
    }
while(1)
{
    
}

} //IF 0

temp4 = CT_MAX();

if(temp4 > 0x0c)
{
    temp4 = 0xFF - temp4;

    for(j=0;j<temp4;j++)
    {
        delay1();
    }
}

data1=data1-1;
if(data1==data50)
{

T0=1; //on green
    delay1();
RXD=0; // CE
    delay1();
    delay1();
TXD=0; //on track1
    delay2();

    delay1();
    delay1();

    delay2();

for(i=0;i<=5;i++)
{
    delay2();
}

RXD=1;
delay2();
TXD=1;
RXD=0;
delay2();
delay2();
RXD=1;
}

else
if(data1==data75)
{
    flag=1;
    T0=0;     //off green
    T1=1;   //on orange
}

} //MAIN