|  |
| --- |
| 1.1 Definition Caller ID (caller identification or CID, and more properly calling number identification - CNID) is a telephony intelligent network service that transmits the caller's telephone number and in some places the caller's name to the called party's telephone equipment during the ringing signal or when the call is being set up but before the call is answered. In the USA, CNID is typically transmitted digitally using Bell 202 modulation between the first and second rings, but in other countries other methods are also used. Instead of sending the caller ID in between the first and second ring, some systems use a "line reversal" to announce the caller ID, or caller ID signals are simply sent without any announcement. Instead of Bell 202 the European alternative V.23 is sometimes used, (without the 75 baud reverse channel) or the data is sent using DTMF signaling.  Caller ID system is a feature of digital telephone network, which allows the telephone number of the caller to be transferred to the called person before he/she answers the phone. This feature can be useful for example for storing numbers of the people who have tried to call you when you were not home. This feature is nowadays activated so that the caller number is transferred always to the telephone central near called person. If the called person wants to get that information, he/she must order this service from telephone company.  **1.2 Features and Benefits:**   * Instantly know who's calling: caller id displays the caller's name and number after the first ring, so you can decide instantly whether or not to answer. * Saves call information: caller id saves names, numbers, dates and times of incoming calls .It makes returning calls simple. * Time saving: your time is valuable .If you know who's trying to reach you ,you can determine if the call is important or not .you will avoid the hassle of unwanted callers and spend your time doing what you want to do. * Blocked Calls: At your option, unidentified or anonymous, out of area, blocked and private calls are not allowed to ring your phone. Then select between allowing a message or ignore with no answer. |

**1.3 Data Message Format**

The following 2 formats are used to send the CID information:

* SDMF - Single Data Message Format .
* MDMF - Multiple Data Message Format.

SDMF supports a single data type and is used to send the phone number for number only service.

MDMF supports multiple data types and is used to send name and number information. It is structured so that new types of data can be added easily (address?, city?, state?, etc).

Both formats include a value to determine the type of data to follow (data type word), a value to indicate the length of the data (data length word), and a checksum.

Single Data Message Format (SDMF)

The Single Data Message Format (SDMF) is as follows :

(1) Header, consisting of:-

(a) The 'Message Type', which is an 8-bit word identifying the feature

generating the message.

(b) The 'Message Length', which is an 8-bit word indicating the number of message words following in the message body (1 - 255).

(2) Message Body

The message body contains up to 255 8-bit words.



The Single Data Message Format includes the date, time, and phone number of the calling party. Figure 1 shows a timing diagram for the single data message format. The message type is transmitted as the first part of the data. The second part contains the length of the message, followed by the month, day, military hour, minute, and the 10-digit phone number. If the caller is out our your area, or blocking his number, then an O or P will be received instead of the actual phone number. The last part of the message is a checksum, which is equivalent to the 2's compliment of the sum of all the bits in the word.

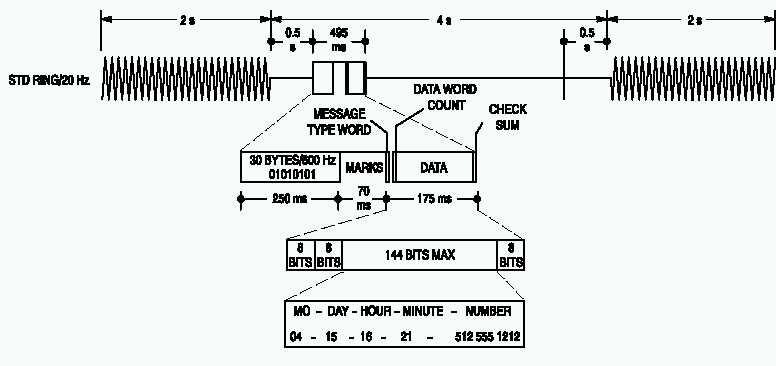


Figure 1: Simple Data Message Format

Multiple Data Message Format (MDMF)

The Multiple Data Message Format (MDMF) is as follows:-

(1) Header, consisting of:-

(a) The 'Message Type', which is an 8-bit word identifying the feature generating the message.

(b) The 'Message Length', which is an 8-bit word indicating the number of message words following (1 - 255). This includes the parameter message headers as well as the parameter message bodies.

(2) Message body:-

(a) Parameter Message Header, consisting of:-

(i) The 'Parameter Message Type', which is an 8-bit word identifying the feature generating the parameter message.

(ii) The 'Parameter Message Length', which is an 8-bit word indicating the length of the parameter message.

(b) Parameter Message Body containing a series of 8-bit words.

****

The Multiple Data Message Format has one simple improvement over the Single Data Message Format. It allows for the transmission of the name of the calling party in addition to the number, date, and time provided by SDMF. Figure 2 shows a timing diagram of the Multiple Data Message Format.

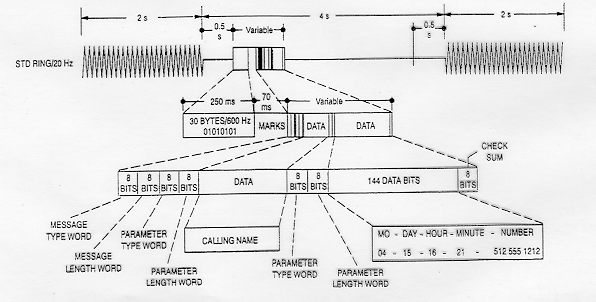


 Figure 2: Multiple Data Message Format

The messages delivered by the multiple data message format are broken into different parameters. For example, the date and time of the incoming call are delivered separately from the calling number. The arrangement of these parameters within the message depends upon the local Caller ID provider.

## SDMF Example

Here is an example of a SDMF message. Each byte is in HEX.

04 12 30 39 33 30 31 32 32 34 36 30 39 35 35 35 31 32 31 32 51

|  |  |  |
| --- | --- | --- |
| Hexadecimal Value | Description | ASCII Value |
| 04 | Message type word –  4 indicates SDMF |  |
| 12 | Message length  18 decimal, number of bytes in date, time and phone number |  |
| 30,39 | Month (September) | 0  9 |
| 33,30 | Day (30) | 3  0 |
| 31,32 | Hour (12) | 1  2 |
| 32,34 | Minutes (24) | 2  4 |
| 36,30,39 | Area code (609) | 6  0  9 |
| 35,35,35 | Prefix (555) | 5  5  5 |
| 31,32,31,32 | Suffix (1212) | 1  2  1  2 |
| 51 | Checksum Word |  |

Thus, the CID string can be summarized as follows:

The message is in SDMF format, consisting of 18 bytes of information, not including the checksum. The call was made on September 30 at 12:24pm. The calling party's phone number was (609)555-1212.

If the calling party's directory number is not available to the terminating central office, the phone number field contains an ASCII 'O'. If the calling party invokes the privacy capability (blocking), the phone number field contains an ASCII 'P'.

The following SDMF string is an example of a call that was blocked. The 50 just before the checksum is the ASCII code for a 'P'.

04 09 30 39 33 30 31 32 32 34 50 12

## MDMF Example

Here is an example of a MDMF message. Each byte is in HEX.

80 20 01 08 30 33 32 34 30 39 30 32 07 08 4A 4F 48 4E 20 44 4F 45 02 0A 38 30 30 35 35 35 31 32 31 32 7D

|  |  |
| --- | --- |
| Hexadecimal Value | Description |
| 80 | Message type word, 80 indicates MDMF |
| 20 | Message length  Length of data, 32 bytes in date,time,name and number |
| 01 | data type, 1 = date & time |
| 08 | length of date and time, 8 bytes |
| 30,33 | 03, March |
| 32,34 | 24, 24th day |
| 30,39 | 09, hour |
| 30,32 | 02, minutes (9:02am) |
| 07 | data type, 7 = name |
| 08 | length of name, 8 bytes |
| 4A | 'J' |
| 4F | 'O' |
| 48 | 'H' |
| 4E | 'N' |
| 20 | ' ' (space) |
| 44 | 'D' |
| 4F | 'O' |
| 45 | 'E' |
| 02 | data type, 2 = phone number |
| 0A | length of phone number, 10 bytes |
| 38 | 8 |
| 30 | 0 |
| 30 | 0 |
| 35 | 5 |
| 35 | 5 |
| 35 | 5 |
| 31 | 1 |
| 32 | 2 |
| 31 | 1 |
| 32 | 2 |
| 7D | Checksum |

The following is a list of possible data types:

* 1 - date & time
* 2 - phone number
* 4 - number not present
* 7 - name
* 8 - name not present

If the calling party's directory number is not available to the terminating central office, the phone number and name fields contain an ASCII 'O'. If the calling party invokes the privacy capability (blocking), the phone number and name fields contain an ASCII 'P'.

From what I have seen, the types 4 and 8 are always followed by a length value of 1 and then either a 'P' or an 'O' to indicate unavailable or blocked, respectively.

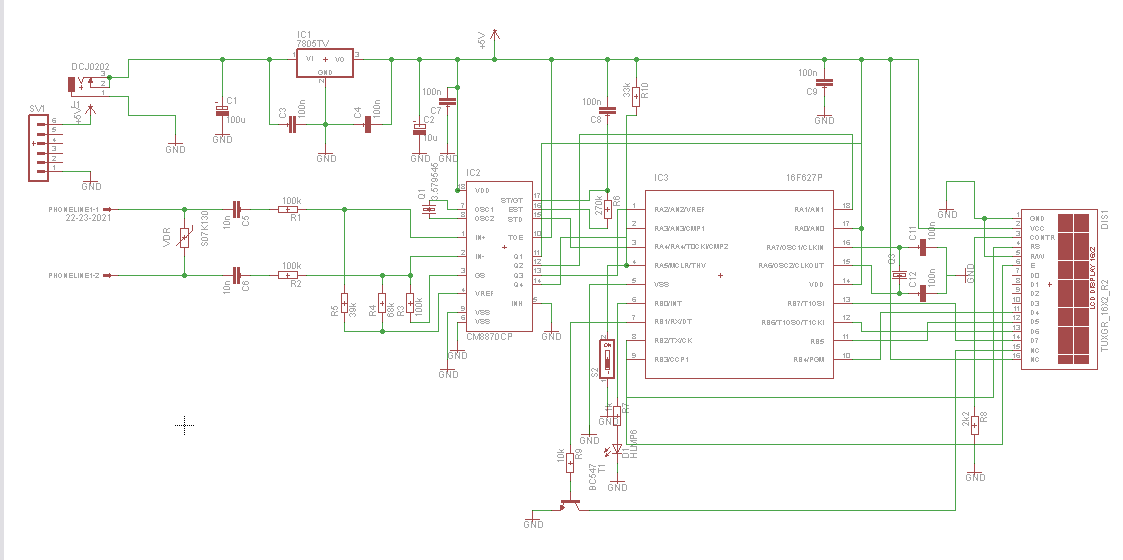
## Checksum

The checksum value contains the two's complement of the modulo 256 sum of the other words in the data message (i.e., message type, message length, and data words). The receiving equipment may calculate the modulo 256 sum of the received words and add this sum to the received checksumword. A result of zero generally indicates that the message was correctly received. Message retransmission is not supported.

* 1. **FLOW CHART Of ID CALLER**



2.1 **Basic ID Caller**



**The Basic ID Caller contains of**:

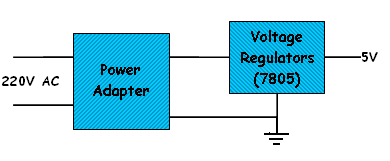
PIC 16f628A (microcontroller)

DTMF decoder (CM8870)

In this circuit the caller ID signal passes through from telephone line then decoding the ID signal and displaying the number on LCD.

**Power Supply:**

The Power supply circuitry shown in figure below, is used for feeding the boards with +5VDC power supply. It consist from the AC power supply block, power adapter, and 5V regulator.

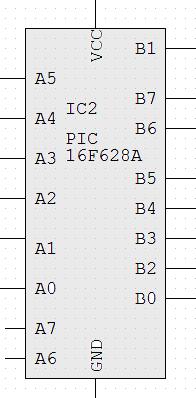


The Power adapter converts the 230V-AC to 12V-DC.While the voltage regulators (7805), are used for fixing the output voltages regardless of the changes in the input voltages.

**2.2 Microcontroller PIC 16F628A:**

18 pin, Flash Based, 8\_bit CMOS

Microcontroller with nano watt technology.



* **Features of PIC 16f628a:**

1. 8 level deep hardware stack

2. Maximize the system reliability, high performance.

3. Minimize cost through elimination of external components

4. Provide power \_saving operating mode.

5. Offer code protection

6. Oscillator selection, it has internal oscillator from 4-48 MHz

7. OSU timer (oscillator starting up timer), intended to keep the chip in reset until the crystal oscillator is stable.

* + 8. PWRT (power \_up timer), provides fixed delay of 72 ms (nominal) on power up only, and designed to keep the part in reset while the power supply stabilized.

**Table 2.1 pic16f628A**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Timers  8/16-bit | Comparators | USART | CCP  (PWM) | I/O | Data Memory | | Program memory | Device |
| EEPROM  (bytes) | SRAM  (bytes) | Flash  (words) |  |
| 2/1 | 2 | Y | 1 | 16 | 128 | 224 | 2048 | PIC 16f628A |

**Table 2.2 Device Memory List**

|  |  |  |  |
| --- | --- | --- | --- |
| Memory | | | Device |
| EEPROM  Data | RAM  Data | Flash Program | PIC 16f628A |
| 128\*8 | 224\*8 | 2048\*14 |

# Dual-tone multi-frequency signaling(DTMF)

Dual-tone multi-frequency signaling (DTMF) is used for  [telecommunication signaling](http://en.wikipedia.org/wiki/Signalling_(telecommunications)) over analog telephone lines in the voice-frequency band between [telephone](http://en.wikipedia.org/wiki/Telephone) handsets and other communications devices and the [switching center](http://en.wikipedia.org/wiki/Automatic_telephone_exchange). The version of DTMF that is used in [push-button telephones](http://en.wikipedia.org/wiki/Push-button_telephone) for tone dialing is known as Touch-Tone.

DTMF Decoder is a very easy to use program to decode DTMF dial tones found on telephone lines with touch tone phones.   
DTMF Decoder is also used for receiving data transmissions over the air in amateur radio frequency bands.

The following are the frequencies used for the DTMF (dual-tone, multi-frequency) system, which is also referred to as tone dialing. The signal is encoded as a pair of sinusoidal (sine wave) tones from the table below which are mixed with each other. DTMF is used by most PSTN (public switched telephone networks) systems for number dialing, and is also used for voice-response systems such as telephone banking and sometimes over private radio networks to provide signaling and transferring of small amounts of data.

Table of DTMF frequencies

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol | | Tone B [Hz] | | | |
| 1209 | 1336 | 1477 | 1633 |
| Tone  A  [Hz] | 697 | 1 | 2 | 3 | A |
| 770 | 4 | 5 | 6 | B |
| 852 | 7 | 8 | 9 | C |
| 941 | \* | 0 | # | D |

For example, in order to generate the DTMF tone for "1", you mix a pure 697 Hz signal with a pure 1209 Hz signal, like so:

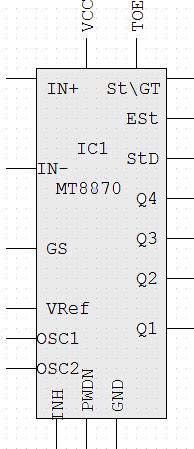
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Waveform for a pure 697 Hz Sine Wave | + | Waveform for a pure 1209 Hz Sine Wave | = | Waveform for DTMF Tone 1 | | | | | | |
| 697 Hz Sine Wave | | + | 1209 Hz Sine Wave | = | DTMF Tone "1" |

Two Pure Sine Waves combine for form the DTMF Tone for "1"

* What about the A, B, C, and D keys?

The standard defines the DTMF tones for 16 keys, but telephones only use 12 of these 16 keys. The remaining 4 tones are sometimes used within the telephone networks.

In our project we use DTMF Decoder/ MT8870 ,figure below shows the pin out of it, and we use an oscillator 3,579545 Hz.

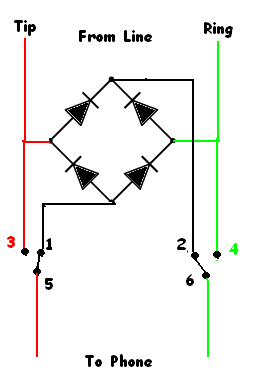
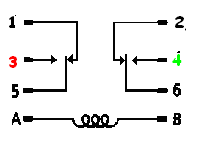


**2.4 Silencer**

This device, wired in series to phone line, will block the ring signal to all devices down-line until I decide to pass the signal or not according to control bit of the microcontroller. But you will still be able to make outgoing calls.

The silencer consists of a DPDT relay and 4 diodes wired as shown below The circuit will disable the ringer, and it will triggered by an control bit of the microcontroller.

If the control bit of the microcontroller is that for known ID calls then the relay will inverse the contactors and connect the phone with the incoming AC ring signal(3,4).else (private calls)the phone still connected to the output of diode bridge (1,2) and the incoming AC ring signal will not go through and phone will not ring(in this case the speech chip will play the recorded message and stop after certain time then phone is reconnect to the line ,using the relay ( as shown in the block diagram of the system in the second page)

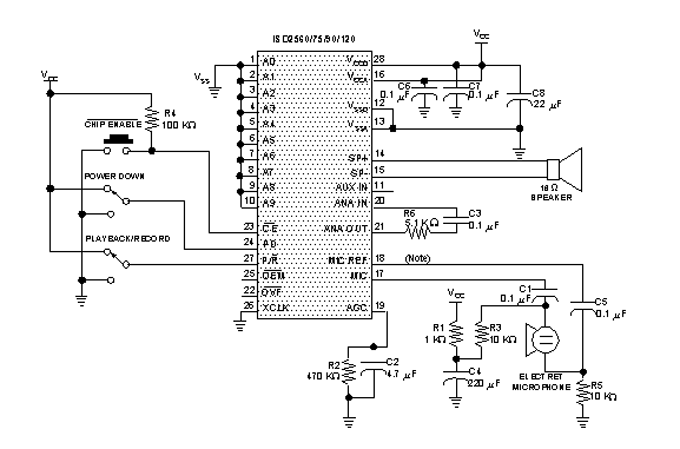


**2.5 Voice Recording ISD-25120:**

For private number we use an ISD-25120 Voice recording IC. This chip will allow us to record up to 120 seconds worth of sound data at specified addresses.

The ISD25120 is a moderately sophisticated component and has several modes of operation. Our application will use the simplest operational mode, M6, also referred to as Push-Button mode. In this mode, the device has the ability to record and playback a single message.

Note that the following circuit is compatible with the ISD2560, ISD2575, ISD2590 and ISD25120.These parts differ in the length of the recorded message space (60, 75, 90 and 120 seconds). The increase is storage space is accomplished by lowering the audio sampling rate. This does give you more storage, but the price you pay is in sound quality.



Using the ISD25120 in Push-Button mode is very easy. Here are the steps:

• Bring PD high, then low to stop the current cycle and reset the device

• Set PR high to play, low to record

• The CE\ line (high-low-high) to start the cycle

• Bring PD high to stop record cycle and reset the device.

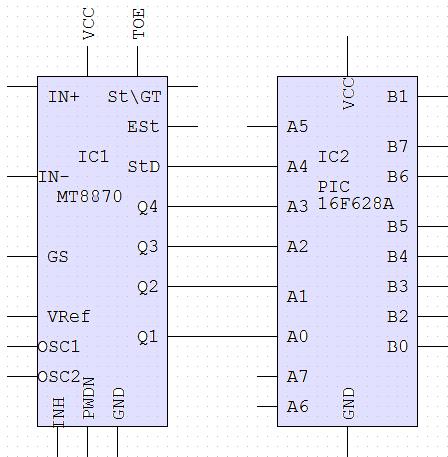
ISD25120 has a CMOS Single Chip Voice Record/Playback device as part of their product line. The CMOS device includes an on chip oscillator, microphone preamplifier, automatic gain control, anti-aliasing filter, smoothing filter. Speaker amplifier, and high density multilevel storage array. In addition, the ISD2500 is micro-controller compatible. Recordings are stored in on-chip nonvolatile memory cells, providing zero-power message storage.

We first determine the starting address of the record, we start with address 0 by making the Address\Mode input all zero's(pin1-pin10)the address are then latched. Then we use Push Button Mode (m6).In this mode pin7(m6),pin9(m8) and pin10(m9) should be high. And then start the recording with the PD (power down)pin be low and P/R pin is taken low then CE is pulsed low to start recording ,and when then CE is pulsed low again the recording is pauses. For playback the PD(power down)pin be low and P/R pin is taken high(playback)then CE is pulsed low to start playback. Until the end of the message.

**3.1 Input connection:**

**\*DTMF Decoder interfaced with microcontroller**

As we can see from the following circuit ,the output pins from MT8870( Q1,Q2,Q3,Q4, and StD(delayed steering output)) are connected to port A of the PIC16f628a( pin A0 … pin A5).



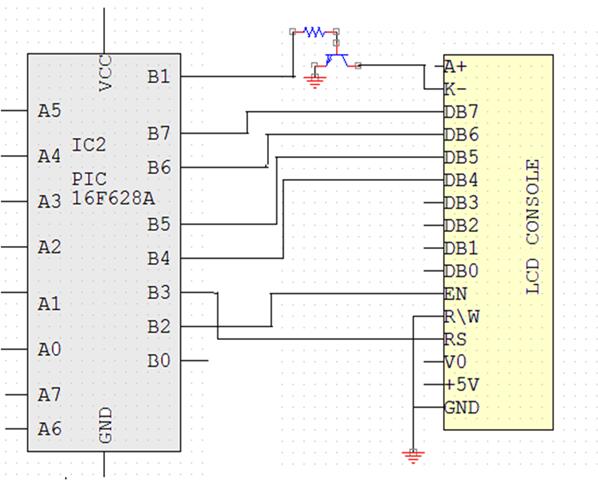
**3.2 Output connection:**

**\*LCD Interfaced with microcontroller**

We use Liquid Crystal Display (LCD) to display the number , and use port B of the PIC 16f628a.

The enable line(EN) allows access to the display through R/W and R/S lines-which connected to pin B2.The Read Write line (R/W) deermines the direction of data between the LCD and microcontroller. The Regiser select(RS) line,the LCD interprets the tybe of data on data lines -which connected to pin B3.

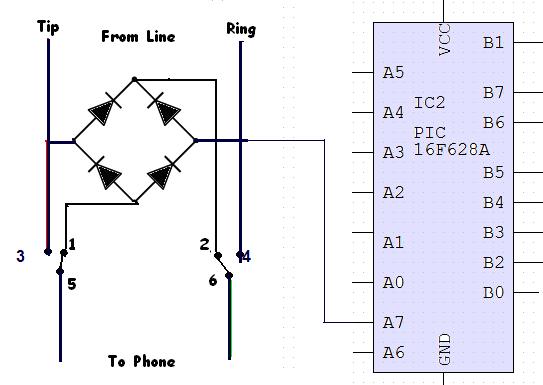
The circuit diagram for interfacing PIC to LCD in the following figure:



**\*microcontroller connected to relay**

The output of the pic is connected to relay which connected to silencer to prevent the phone from ring if private number is call or to enable the ring to go on.

The interfaced with the microcontroller through port A ,we use pin A7. As shown in figure below:



**\*microcontroller connected to speech chip:**

If we want to send message to those who call us ,especially if the caller number is private we use the speech chip , and we control it's power and the playback of the message using the microcontroller, then the power down will return high after the playback to disable the chip.

**The power supply**



**The basic id caller**



**Silencer circuit**



**Speech recording circuit**



**4.1 Improvement of ID Caller**

The problem we have faced when we built the basic id caller is the standard where the Palestine Telecommunications Company is used to send the Caller ID data down the line- which is FSK(Frequency Shift Keying)- modulation.

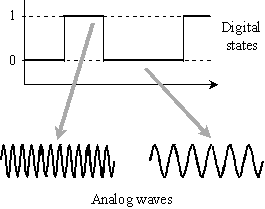
So to solve this problem we can use a convertor to convert from DTMF to FSK(Frequency Shift Keying).

The chip used to demodulate FSK is a decoder chip (MC14LS5447). This chip is manufactured by Motorola Company and designed to demodulate bell core 202,1200 baud FSK (Frequency Shift Keying).

Decoder chip primary application is in products that what will be receive and display the calling number or the message waiting indicator sent to subscribers from participating central office facilities of the public switched telephone network. The device also contains carrier detect circuits and telephone ring detector which may be used to power up the device.

FSK decoder converts standard Bellcore 200 ID signal into a serial ASCII data .

**Frequency-shift keying (FSK)** is a method of transmitting [digital](http://searchCIO-Midmarket.techtarget.com/sDefinition/0,,sid183_gci211948,00.html) signals. The two [binary](http://searchCIO-Midmarket.techtarget.com/sDefinition/0,,sid183_gci211661,00.html) states, logic 0 (low) and 1 (high), are each represented by an [analog](http://searchCIO-Midmarket.techtarget.com/sDefinition/0,,sid183_gci211561,00.html) [waveform](http://searchCIO-Midmarket.techtarget.com/sDefinition/0,,sid183_gci213731,00.html). Logic 0 is represented by a wave at a specific [frequency](http://searchCIO-Midmarket.techtarget.com/sDefinition/0,,sid183_gci212160,00.html), and logic 1 is represented by a wave at a different frequency. A [modem](http://searchMobileComputing.techtarget.com/sDefinition/0,,sid40_gci212583,00.html) converts the binary data from a computer to FSK for transmission over telephone lines, cables, [optical fiber](http://searchTelecom.techtarget.com/sDefinition/0,,sid103_gci212716,00.html), or wireless media. The modem also converts incoming FSK signals to digital low and high states, which the computer can "understand."



**Another improvement**

Blocking incoming calls-you can block telephone numbers of people which you don't want to talk with him/her at any time and any place, without giving this person any feeling that you reject his/her call.

By the microcontroller we can program it to decide who can ring your phone and only lets them through when you want to answer .

You can appoint the time of receive or not any calls, by program for example A-list to only ring at certain times, the time you allow them to call .if any one wants to reject any call after 10 pm ,so you can program your id caller using this time to reject any number after 10 pm.

Also, you can use B list to allocate the emergency calls by set the Ring Controller to only allow calls from this list.

Or you want to avoid having to deal with nuisance calls from someone who you don't ever to talk to them, by using C list.

#include "C:\Documents and Settings\feras\My Documents\heba.h"

#include <2401.C>

#include <stdlib.h>

#include <string.h>

#include <LCD.C>

#int\_TIMER0

TIMER0\_isr()

{

}

#int\_EEPROM

EEPROM\_isr()

{

}

#define WELLECOM\_MESSAGE"CALLER ID"

#define PRIVATE\_CHAR1'P'

#define PRIVATE\_CHAR2'p'

#define OUTAREA\_CHAR1'O'

#define OUTAREA\_CHAR2'o'

#define RELAY PIN\_A6

#define SPEECH\_CE PIN\_A7

#define TCE 100\\ns

#define TPUD 50\\ms

#define SPEECH\_PERIOD 22//sec

#define LCD\_ALL 32

#define EEPROM\_BUF\_SIZE 19

#define EE\_CALL\_NO 0

#define EE\_CALL\_COUNT 1

#define EE\_NO\_START 2

#define EE\_ALL\_NOS 5

#define SEC4 80

#define SEC10 150

#define BUFFER\_SIZE 32

BYTE buffer[BUFFER\_SIZE];

BYTE next\_in=0;

BYTE next\_out=0;

#define bkbhit (next\_in!=next\_out)

char eeprom\_buf[EEPROM\_BUF\_SIZE];

int eeIndex;

char str[32];

int i;

short timeout\_error;

int ringCount;

int timerTicks;

int1 finishedRings;

int1 isChar There;

int1 lcd\_cleared;

int displayTicks;

int displayIndex;

int1 finished Display;

int1 finished;

char intc;

int k;

int j;

char ctmp;

void Init\_ring\_detector();

void RDO\_fn();

void lcd\_print();

int1 is Char\_mode(char c);

char timed\_getc();

BYTE bgetc();

void speechChip\_on();

void speechChip\_off();

#int\_TIMER0

TIMER0 \_isr()

{

if(!finishedRings)

{

timerTicks++;

if (timerTicks==SEC4)

{

timerTicks++;

finishedRings = 1;

RingCount = 0;

finished = 0;

}

}

if (displayTicks < SEC10)

{

displayTicks++;

}

else if (displayTicks==sec10)

{

displayTicks++;

finishedDisplay=1;

displayIndex=7;

}

}

#int\_RDA

RDA\_isr()

{

int t;

if (ringCount==1)

{

buffer[next\_in]=getc();

t=next\_in;

next\_in=(next\_in+1)%BUFFER\_SIZE;

if(next\_in==next\_out)

{

next\_in=t;

}

}

else

{

ctmp=getc();

}

}

void main()

{

chat c;

int1 is\_line\_2;

int32 timeout;

setup\_adc\_ports(NO\_ANALOGS);

setup\_adc(ADC\_OFF);

setup\_psp(PSP\_DISABLED);

setup\_spi(FALSE);

setup\_timer\_1(T1\_DISABLED);

setup\_timer\_2(T2\_DISABLED,0,1);

restart\_wdt();

restart\_wdt();

lcd\_init();

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_gotoxy(1,2);

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_gotoxy(1,1);

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_gotoxy(1,2);

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_gotoxy(1,1);

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_gotoxy(1,2);

sprintf(str,WELLCOME,MESSAGE);

lcd\_print();

delay\_ms(400);

lcd\_putc('\f');

delay\_ms(400);

lcd\_print();

lcd\_gotoxy(1,1);

Init\_ring\_detector();

lcd\_cleared=1;

while(1)

{

RDO\_fn();

if (ringCount==1)

{

lcd\_cleared=0;

finshed=0;

restart\_wdt();

if(bkbhit)

{

c=bgetc();

if(isChar\_mod(c))

{

eeprom\_buf[eeIndex]=c;

eeIndex++;

restart\_wdt();

lcd\_putc(c);

restart\_wdt();

K++;

if(j==2)

{

lcd\_putc('/');

}

else if(j==4)

{

lcd\_putc('');

}

else if(j==6)

{

lcd\_putc(':');

}

if(j==8)

lcd\_gotoxy(1,2);

}

if((c==PRIVATE\_CHAR1)||(c==PRIVATE\_CHAR2))

{

speechChip\_on();

ringCount=3;

finished=0;

lcd\_cleared=0;

}

else

{ output\_low(RELAY); }

}

restart\_wdt();

}

if(ringCount>=2)

{

int ewriteTmp,no\_calls,call\_count\_t;

char c\_no\_calls,c\_call\_count;

ringCount=0;

if(eeIndex>0)

{

restart\_wdt();

c\_no\_calls=read\_eeprom(EE\_CALL\_NO);

c\_call\_count=read\_eeprom(EE\_CALL\_COUNT);

restart\_wdt();

no\_calls=c\_no\_calls-'0';

call\_calls\_t=c\_call\_count\_'0';

restart\_wdt();

if((no\_calls>=1)&&(no\_calls<=5))

{

restart\_wdt();

if(no\_calls==5)

{

no\_calls=5;

} els {

no\_calls++;

} }

else

{

no\_calls=1;

}

restart\_wdt();

write\_eeprom(EE\_CALL\_NO,(no\_calls+'0'));

restart\_wdt();

if((call\_count\_t>=0)&&(call\_count\_t<=4))

{

call\_count\_t++;

call\_count\_t=call\_count\_t%(EE\_ALL\_NOS);

}

else

{

call\_count\_t=0;

restatrt\_wdt();

}

restart\_wdt();

write\_eeprom(EE\_CALL\_COUNT,(call\_count\_t+'0'));

for(ewriteTmp=0;ewriteTmp<eeIndex;ewriteTmp++)

{ restart\_wdt();

write\_eeprom(((EE\_NO\_START+(call\_count\_t\*EEPROM\_BUF\_SIZE))+ewrite Tmp);

eeprom\_buf[ewriteTmp]);

restart\_wdt();

}

restart\_wdt();

write\_eeprom(((EE\_NO\_START+(call\_count\_t\*EEPROM\_BUF\_SIZE))+ewriteTmp),'#');

restart\_wdt();

eeIndex=0;

}

}

if((!finished)&&((finishedRings&&!lcd\_cleared)||(finishedRings&&finishedDisplay)))//(

finishedRings)//(ringCount==2)

{

j=0;

k=0;

is\_line\_2=0;

ringCount=0;

lcd\_cleared=1 ;

finished=1;

lcd\_putc('\f');

sprintf(str,WELLCOME\_MESSAGE);

lcd\_print();

lcd\_gotoxy(1,1);

restart\_wdt();

finishedDisplay=0;

output\_high(RELAY);

}

restart\_wdt();

}

}

void Init\_ring\_detector()

{

restart\_wdt();

ringCount=0;

timerTicks=0;

finishedRings=1;

isChar There=0;

finished=1;

displayTicks=SEC10+10;

displayIndex=7;

finishedDisplay=0;

output\_high(RELAY);

output\_high(SPEECH\_CE);

output\_high(SPEECH\_PD);

j=0;

k=0;

eeIndex=0;

setup\_counters(RTCC\_INTERNAL,RTCC\_DIV\_256);

enable\_interrupts(INT\_RDA);

enable\_interrupts(INT\_TIMER0);

enable\_interrupts(global);

restart\_wdt();

}

void RDO\_fn()

{

if(!input(RDO))//=0

{

delay\_ms(10);

if(input(RDO))

{

return;

}

while(!input(RDO))

{

restart\_wdt();

}

timer Ticks=0;

ringCount++;

finished Rings=0;

finished=1;

return;

}

}

void lcd\_print()

{

for(i=0 ; i<LCD\_ALL;i++)

{

restart\_wdt();

if(str[i]=='\0')

{

break;

}

lcd\_putc(str[i]);

restart\_wdt()

}

}

int1 isChar\_mod(char c)

{

if((c=='p')||(c=='P'))

{

return(1);

}

else if((c>='o')||(c<=='O'))

{

return(1);

}

else if((c>='0')&&(c<='9'))

{

return(1);

}

else

{

return(0);

}

}

char timed\_getc()

{

long timeout;

timeout\_error=FALSE;

timeout=0;

while(!kbhit()&&(++timeout<50000))//1\2 second

{

restart\_wdt();

delay\_ms(10);

restart\_wdt();

}

if(kbhit())

{

restart\_wdt();

return(getc());

}

else

{

restart\_wdt();

timeout\_error=TRUE;

return(0);

}

}

BYTE bgetc()

{

BYTE c;

restart\_wdt();

while(!bkhit)

{

restart\_wdt();

}

c=buffer[next\_out];

next\_out=(next\_out+1)%BUFFER\_SIZE;

return(c);

}

displayTicks=0;

finished=0:

if(displayIndex>EE\_ALL\_NOS)

{

restart\_wdt();

c=read\_eeprom(EE\_CALL\_COUNT);

displayIndex=c-'0';

}

else

{

displayIndex++;

displayIndex=displayIndex%(EE\_ALL\_NOS);

displayTicks=0;

}

tmp=0:

finishedDisplay=0;

j=0;

k=0;

lcd\_init();

lcd\_gotoxy(1,1);

for(ii=0;ii<EEPROM\_BUF\_SIZE;ii++)

{

restart\_wdt():

c=read\_eeprom(((EE\_NO\_START+(displayIndex\*EEPROM\_BUF\_SIZE)+tmp));

tmp++;

if(c=='#')

{

if(c==0)

{

lcd\_putc("deleted");

}

return;

}

if(isChar\_mod(c))

{

restart\_wdt();

lcd\_putc(c);

restart\_wdt();

restart\_wdt();

J++;

k++;

if(j==2)

{

lcd\_putc('/');

}

else if(j==4)

{

lcd\_putc('');

}

else if(j==6)

{

lcd\_putc(':');

}

if(j==8)

{

lcd\_gotoxy(1,2);

}

restart\_wdt();

}

}

restart\_wdt();

return;

}

}

displayTicks=0;

finished=0;

if(displayIndex>=(EE\_ALL\_NOS);

{

restart\_wdt();

c=read\_eeprom(EE\_CALL\_COUNT);

displayIndex=c-'0';

}

else

{

if(display Index>0)

{

displayIndex=dislpayIndex-1;

}

else

{

displayIndex=EE\_ALL\_NOS-1;

}

displayIndex=displayIndex%(EE\_ALL\_NOS);

displayTicks=0;

}

tmp=0;

finishedDisplay=0;

lcd\_init();

lcd\_gotoxy(1,1);

j=0;

k=0;

for(ii=0; ii<EEPROM\_BUF\_SIZE;ii++)

{ c=read\_eeprom(((EE\_NO\_START+(displayIndex\*EEPROM\_BUF\_SIZE\_)+tmp));

tmp++:

if(c=='#')

{

if(ii==0)

{

lcd\_putc("deleted");

}

return;

}

if((isChar\_mod(c))

{

restart\_wdt();

lcd\_put(c);

restart\_wdt();

restart\_wdt();

j++;

k++;

if(j==2)

{

lcd\_putc('/');

}

else if((j==4)

{

lcd\_putc('');

}

else if(j==6)

{

lcd\_putc(':');

}

if(j==8)

{

lcd\_gotoxy(1,2);

}

restart\_wdt();

return;

}

}

if(display Index<EE\_ALL\_NOS)

{

lcd\_putc("Deleting Number");

write\_eeprom(((EE\_NO\_START+(displeyIndex\*EEPROM\_BUF\_SIZE))),'#');

} return; }

VOID speechchip\_on()

{

int16 i;

finished=0;

output\_high(RELAY);

output\_low(SPEECH\_PD);

delay\_ms(TPUD);

outpu\_low(SPEECH\_CE);

delay\_ms(10);

for(i=0;i<SPEECH\_PERIOD;i++)

{

delay\_ms(1000);

}

output\_high(SPEECH\_PD);

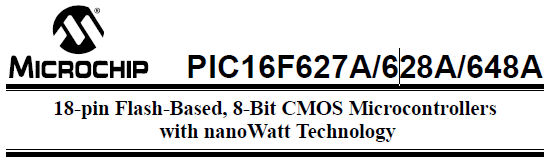
}

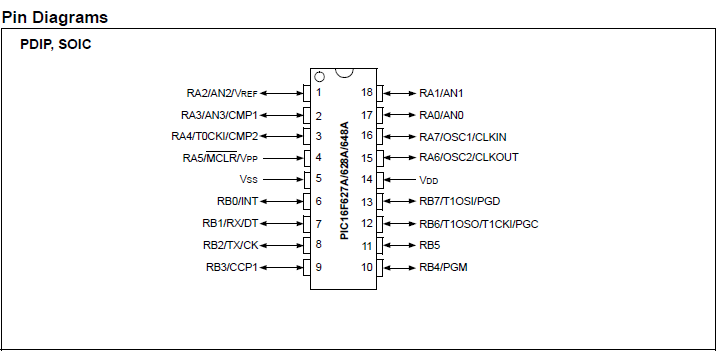
void speechChip\_off()

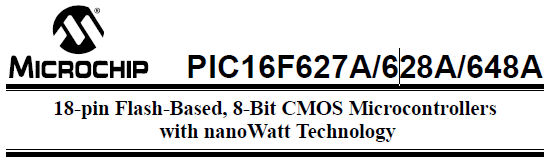
{

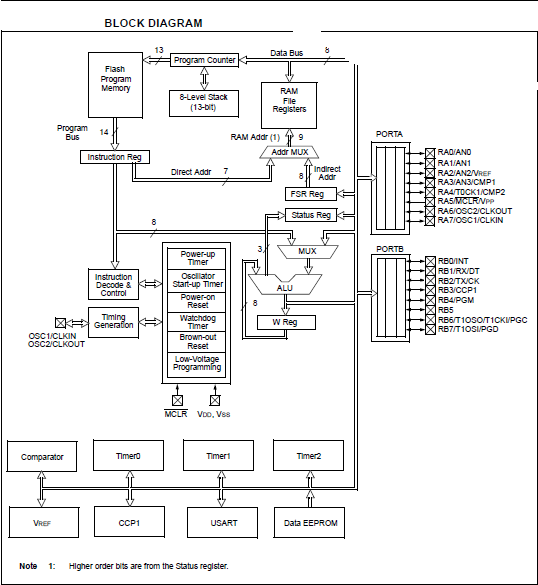
}

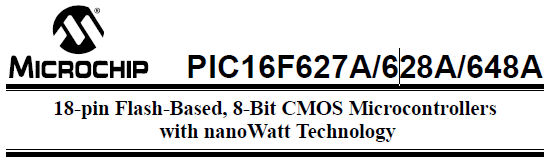
**Appendix 1:**

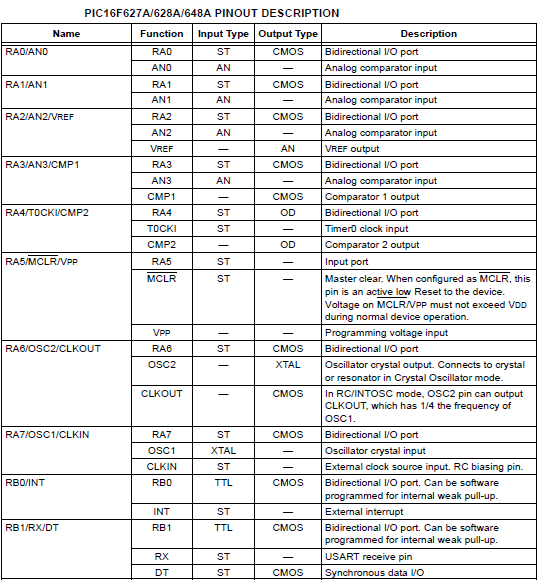


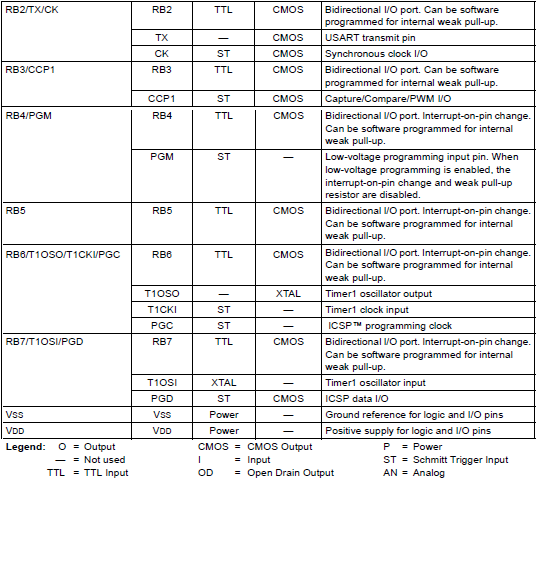




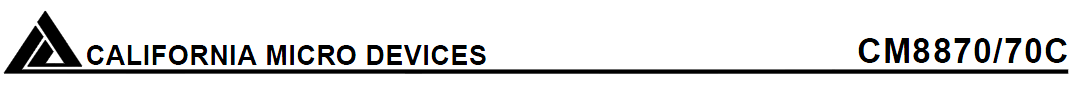


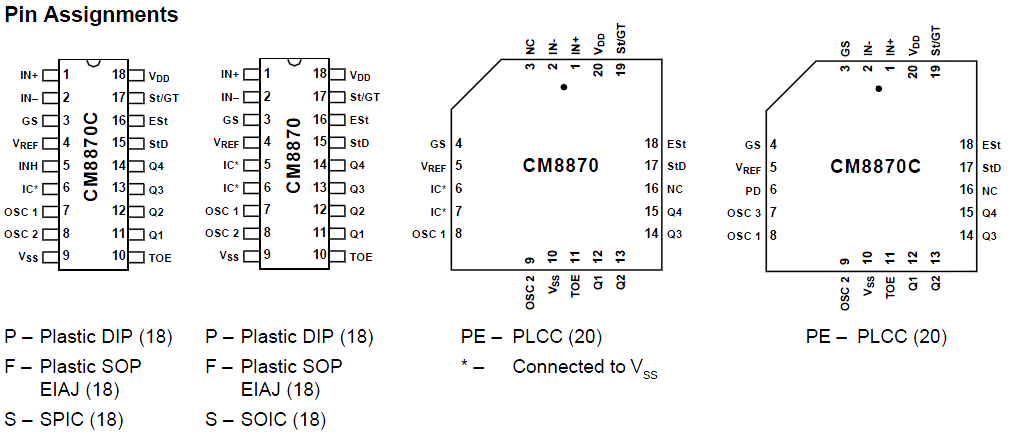


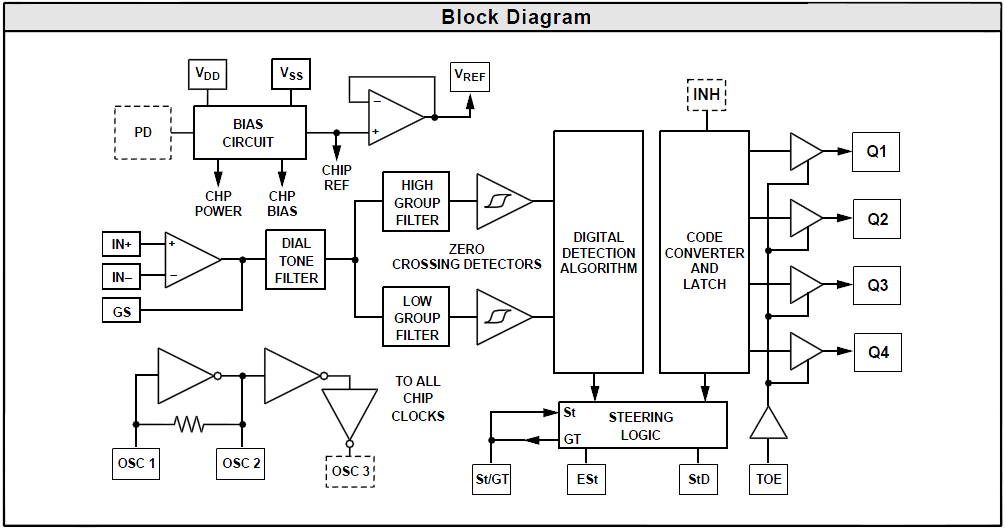


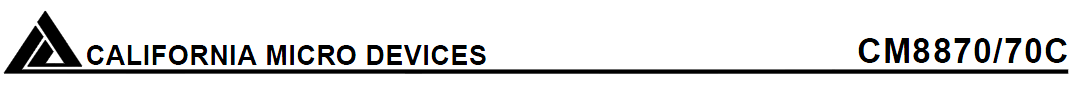


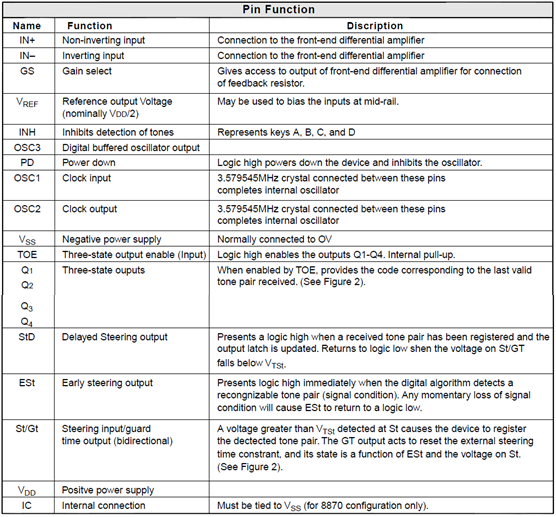
**Appendix 2:**



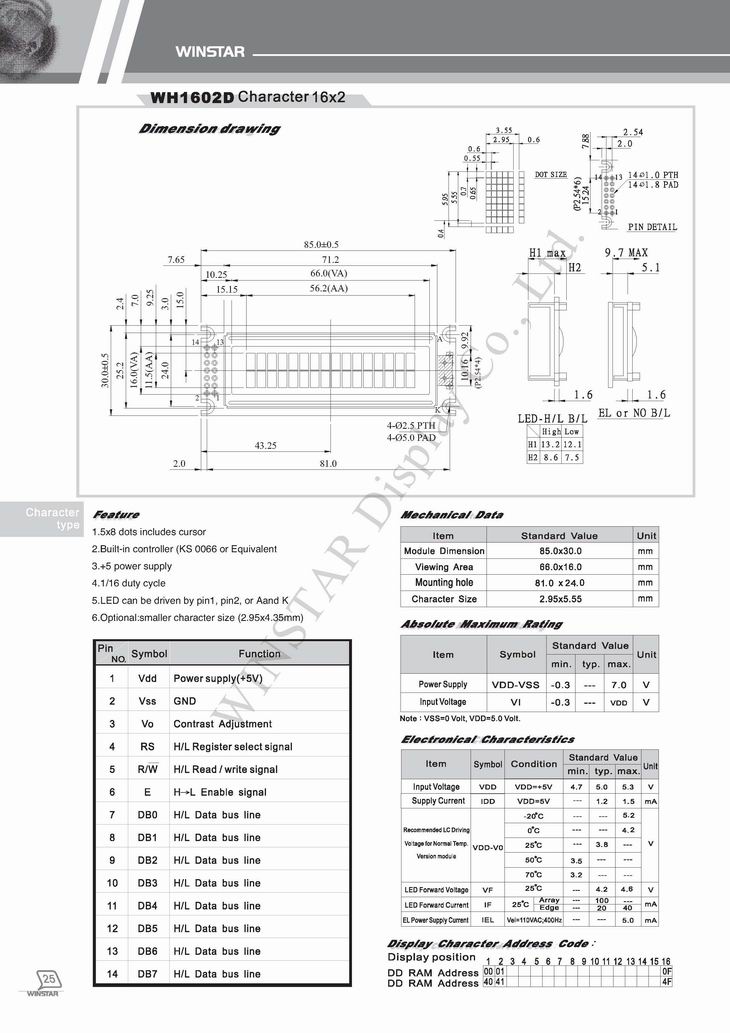






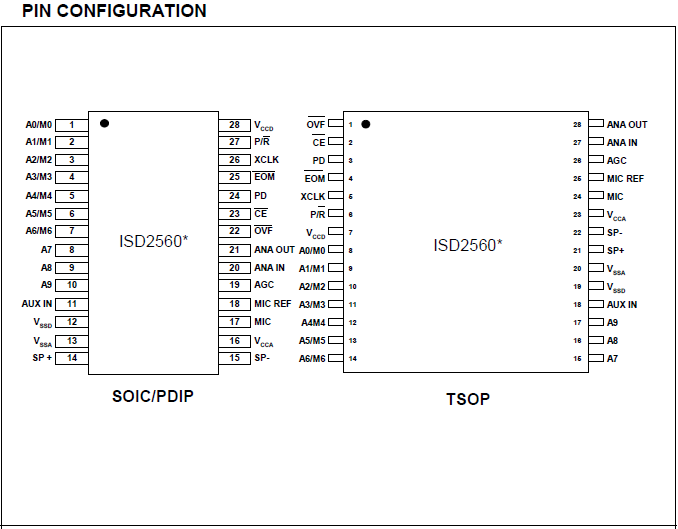


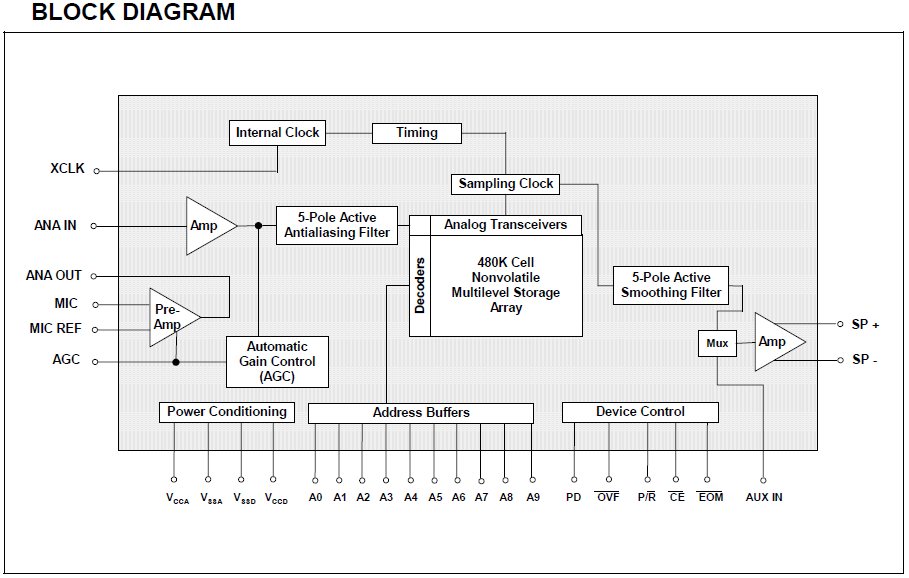
**Appendex 3**

**Liquid Crystal Display (LCD)**  

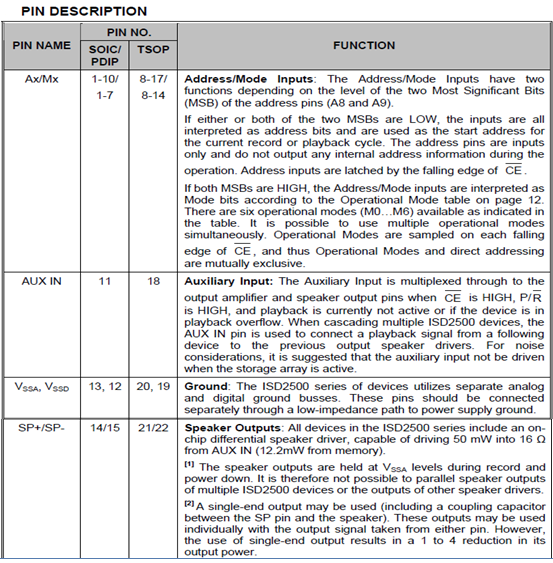
**Appendix 4:**

****

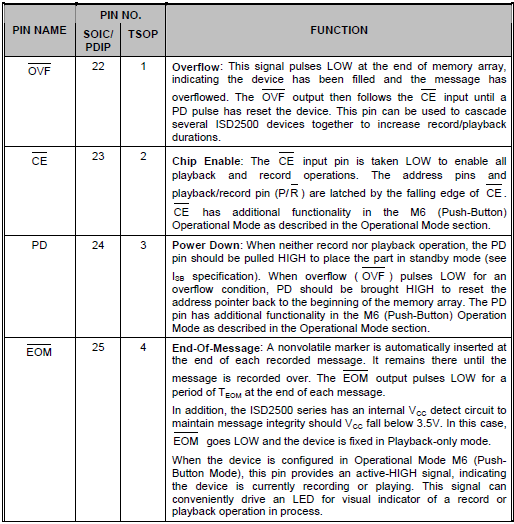
****

****

****

****

****

****