DIAL ALARM-2

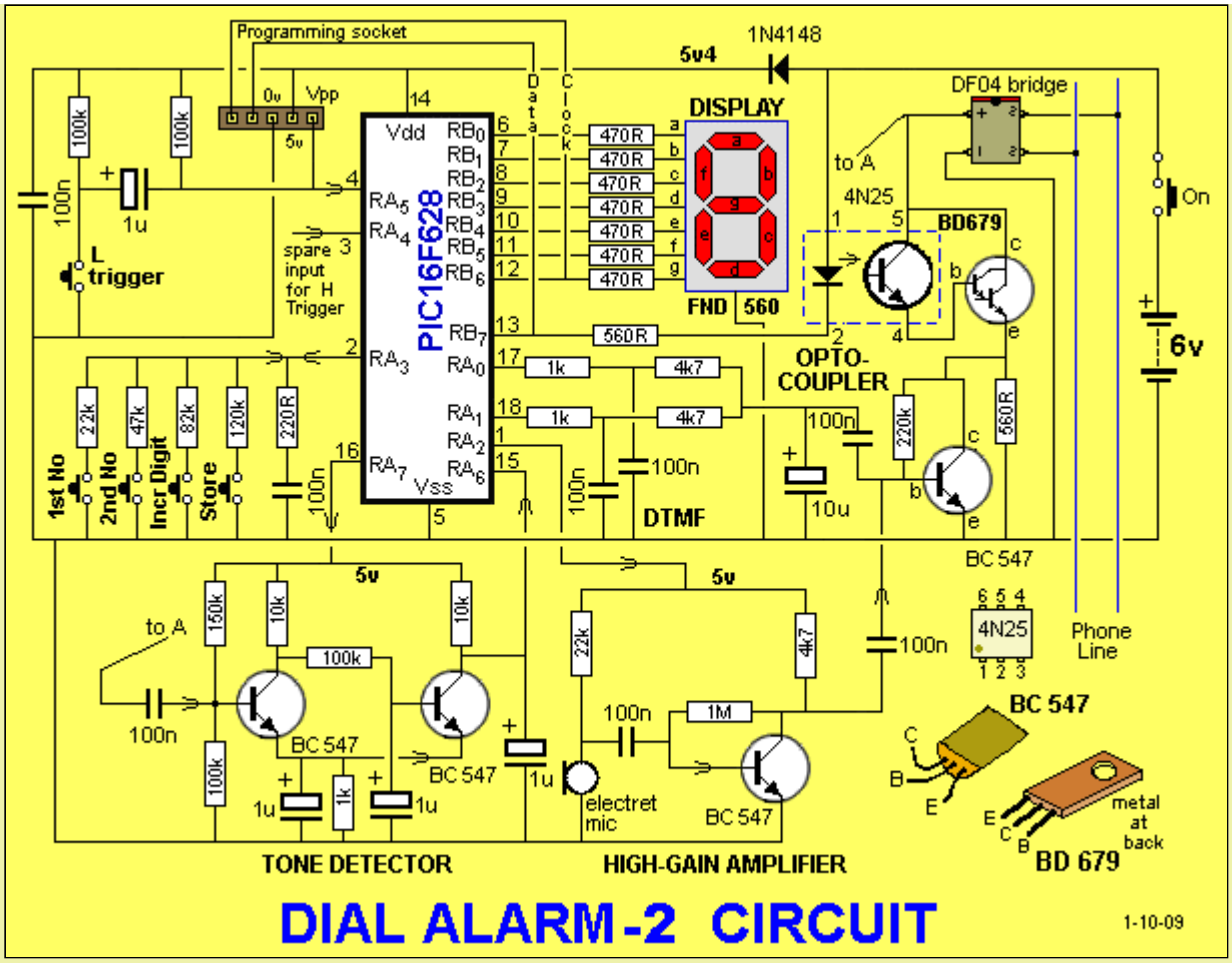
Home

Instruction Set for PIC16F628
PIC16F628 data (pdf)
BlankF628.asm
BlankF628.txt
If you have a laptop, buy:
Chip Programmer - PICkit2 from Modtronix
(MPASM and MPLAB come with PICkit2)
If you have a desktop, buy:
Multi Chip Programmer and
download MPASM and WinPIC.exe or WinPIC.zip
PIC16F628A.inc
Notepad2.zip Notepad2.exe
Library of Sub-routines "Cut and Paste"
Library of routines: **A-E** **E-P** **P-Z**
See more projects using micros:
Elektor,EPE,Silicon Chip

Buy a kit

A complete dialing alarm the size of a pack of cigarettes - with features that will amaze you

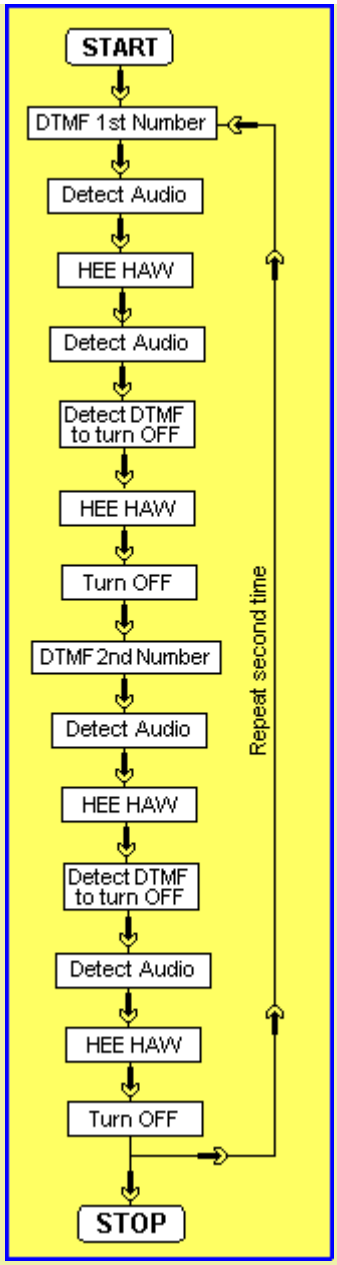
This is the lowest-cost dialing alarm on the market and shows what can be done with a PIC microcontroller. The complete circuit is shown below. You cannot see all the features of this project by looking at the circuit - most of them are contained in the program. So, read on and see what we have included. . .



The programming socket is not on the PC board - it has been added for the time when you want to modify the program and "burn" a new chip.
The arrows on the diagram show the direction of a signal. Outputs RA4 and RA6 produce rail voltage for the stages they are supplying.

Dial Alarm-2 has a single input (although a number of sensors can be placed in parallel on the same input line). The circuit requires a trigger pulse to turn on the Alarm. This is achieved by placing a 1u on the input line and keeping it discharged via two 100k resistors. When the input goes low, the 1u transfers the LOW to the micro and if the input remains LOW, the 1u charges via the second 100k resistor.

The micro executes the program and outputs a low on RB7 to turn on the LED in the opto-coupler and this causes the line to be "picked up" via a high-gain Darlington transistor. The micro then dials two phone numbers and produces a Hee Haw sound to alert the called party of an intrusion. The circuit also has a sensitive microphone with a high-gain amplifier. This is connected to the phone line when the alarm is triggered.
Amplified audio of the room is then passed down the line after the Hee Haw tone. This signal is clear enough to detect conversations and/or movement in the target area and the listener can determine the situation. If the sounds are determined to be family or staff etc, the alarm can be de-activated by pressing any of the buttons on the receiving phone. This will pass a tone down the line and is picked up by the alarm to shut it off. If the first number is not answered within a few seconds, a second number is called and the process is repeated. The two numbers are then called again and the alarm closes down. Simple but brilliant. The flow Diagram for the alarm is shown below:



Dial Alarm-2 Flow Diagram

Use **Dial Alarm-2** as a "Back-Up" Alarm

This alarm has been developed in response to a number of recent large robberies reported in the news. Robberies are a constantly increasing crime, but very few are reported, unless they have a "twist." Recently, the robbers navigated the conventional alarm system and broke into the night safe in the Manager's office. The haul was quite significant and it's surprising such a large amount of cash was kept on the premises. The weakest link in most alarm systems are the PIR detectors, used to detect movement. It's a known fact that they are very easy to foil. It's so easy we are forbidden to print details of how to do it. But many thieves must be aware of the trick and that's why a back-up system is essential.

The cheapest back-up system is the use of the phone line. I know what you are going to say. Cutting the telephone line is an easy matter and offers little security. But finding the line in a premises is not very easy and if there are two or more incoming lines, it's difficult to know which is connected to the dialler. Nothing is infallible, but for a lot less than $50 you can build this project and have a back-up to protect your property.

The other advantage of our design is the "set and forget feature." The alarm is designed to ring your mobile and if you keep your phone beside you 24 hours a day, you can have this peace of mind, whether you are in your office, factory, holiday house or quietly dining at your favourite restaurant.

You can protect any area where a telephone line can be installed. This includes houses-under- construction and outlying sheds.

Talking Electronics has been producing security devices for more than 15 years and this project is a culmination of those years of experience.

The high-sensitivity amplifier in the alarm is our development and comes from our highly successful **Infinity Bug**. This device connects to the phone line anywhere in the world and when the number is rung, the infinity

bug answers the call and lets you listen in to the activities in the room. It's just like being there. We have used the same circuit in this project. When it is activated, you can easily work out if it has been triggered by staff, a family member or an intruder. At least it prevents unnecessarily attending 90% of false alarms and offers enormous peace of mind.

The secret lies in the placement of the triggering device. We have provided only one input (trigger input). And there's a reason for this. The idea is to place the sensor near the target area or on an actual device, near the microphone.

For instance, it you are protecting a house, a thief always goes to the main bedroom and rummages through the drawers and cupboards. In this case a drawer that is never used should be wired with a magnetic switch (reed switch) or a movement detector such as a mercury switch. These switches can be housed in a plastic case for easy screwing to a wall or door and are very reliable in operation. When the drawer is pulled out or the door opened, the switch is activated. If you are protecting a wall safe, the switch is placed near the safe in a clipboard or picture so that when the board or picture is moved, the alarm is activated. If a room is to be monitored, the switch is placed on the door so that when it is opened, the alarm is activated. If other valuables are being protected (such as a VCR, scanner etc) a suggestion is to place a clipboard against the item. The idea is the clipboard has to be moved to get at the "valuables." The clipboard contains a magnet and the switch is nearby. The clipboard keeps the switch open (or closed) and when it is moved, the alarm is activated.

The ideal arrangement is to avoid touching the clipboard, drawer, door or other "prop" during normal activities and this keeps the alarm activated at all times.

Another suitable trigger device is a pressure mat. This is something that can be avoided by "those in the know" and you can monitor an area during your absence. The alarm can be used for other things too. You can determine when your business premises are opened up in the morning by placing a pressure mat or reed switch on a door. The same can apply to a particular room in your establishment.

The purpose of this article is not only to produce the worlds smallest dialling alarm but also show you how the program runs so you can modify any of the routines to suit your own particular requirements.

The program can be re-written to dial only one number for two rings then hang up, or three rings, then again after 2 minutes or any combination to suit your requirements. Many mobile phones identify the caller on the display and you can keep track of the exact time of arrival and departure of different personnel.

The alarm can be programmed to monitor machinery and dial your mobile when a breakdown occurs. It can monitor water level or even your mail box. The possibilities are unlimited and it's just a matter of modifying the program to suit your own needs.

But before you change any of the program you have to understand what the program does and be capable of changing the instructions without upsetting the operation of the alarm.

Remember: A little knowledge is a dangerous thing. Before doing any re-writing of the program you need to read our notes on programming and carry out one small modification at a time.

This is really a very advanced project. The fact that is looks simple is the power of the microcontroller. It's taking the place of at least 10 chips in a normal alarm.

Timing and DTMF tones have all been converted to instructions of a program. And the advantage of a program is the simplicity of alteration. A time-interval can be changed or a phone number altered with a few lines of code. Even new features can be added without the need for additional hardware. This project uses the PIC16F628A to its maximum and shows what can be done with a PIC microcontroller.

You can program a new number or change a number at any time by using the 4 buttons. The number is stored in EEPROM so it will not be lost when the power is removed.

Before we go any further we must state that this project cannot be connected to the public telephone system. Only approved devices can be connected to the Public Phone System and any experimental device must be approved for experimentation and connected via a "telephone Line Separating Device." These are available from Altronic Imports for approx $100.

This is unfortunately the case and when we discuss connecting the project "to the line," we are referring to an experimental telephone system such as the one we have put together at Talking Electronics, to test and develop projects such as these.

See the section "Testing The Project" for more details of the Test Circuit. It consists of 27v derived from 9v batteries, a 12v relay, a telephone and a socket, all in series. The 12v relay is included to limit the current.

Dial Alarm-2 is not isolated from the phone line nor does it have any spike protection. Normal phones has 5,000v isolation The maximum input rejection of **Dial Alarm-2** is 125v made up of 80v via the collector-emitter of the BD679 transistor and 45v via the collector-emitter of the BC547 transistor. The "ring-voltage" can be as high as 120v and the transistors are just at the point of zenering. They may clip the ring voltage if it exceeds 130v.

THE CIRCUIT

The circuit consists of 6 building blocks.

1. The trigger input.
2. The tone (whistle) detector.
3. The DTMF wave-shaping circuit.

4. The high-gain audio amplifier.
5. The microcontroller.
6. The programming buttons

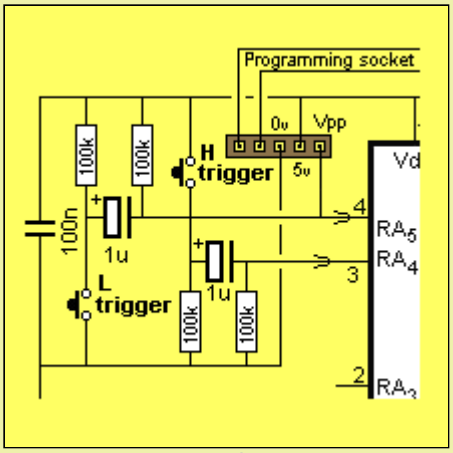
1. THE TRIGGER INPUTS

The project is connected to a 6v supply at all times and to extend the battery life, the micro turns off after use. The current drops to less than 1uA.
The trigger must be a pulse to prevent the circuit re-triggering. This is called a TRIGGER PULSE.
Two trigger inputs have been provided.
L trigger is a LOW trigger and this means the switch connects between the L Trigger input and 0v.
The H Trigger connects between the H Trigger line and 5v rail. See below for circuit.

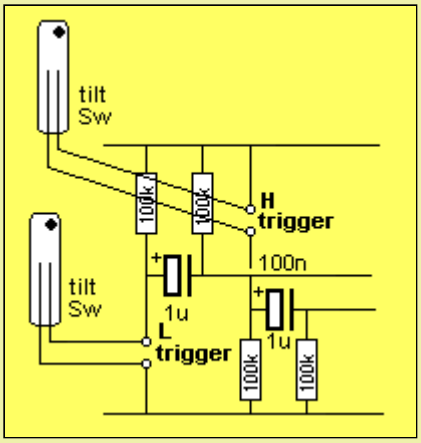
CONNECTING MORE INPUT DEVICES

Input devices are connected to the "L Trigger" (Low Trigger) and "H Trigger" (High Trigger) inputs.
For a sensor such as a mercury switch (tilt switch) or reed switch, it does not matter if they are placed on the L Trigger or H trigger. The alarm is "polling" both inputs.
But if the trigger device is a transistor or output from an alarm module, you need to know if the trigger will be LOW or HIGH when activated. The second diagram below shows how to connect these to the Alarm.

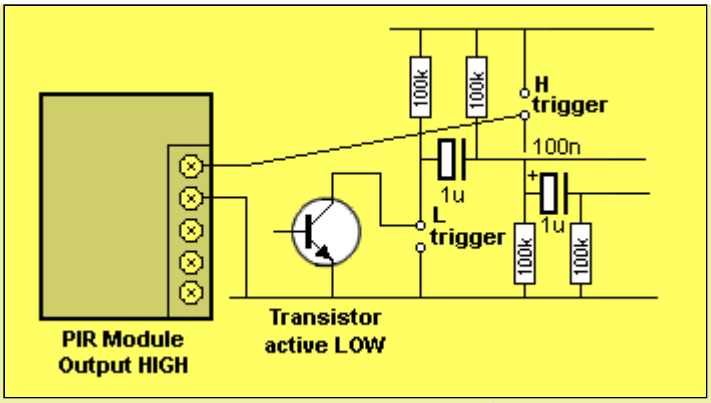
Only one input has been shown on the main circuit but if you want to add a High Trigger, the following circuit can be added. You will have to add the necessary code to the program to detect the H Trigger.



Adding a HIGH Trigger

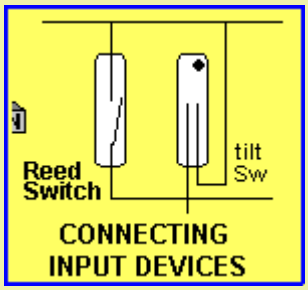


Connecting tilt switch "high" and "low"



Connecting transistor "low" and output of PIR Module "high"

More than one trigger device can be fitted to the alarm provided they are connected in parallel as shown in the diagram below.



2. THE TONE DETECTOR

The simplest building block in the project is the Tone Detector. It is designed to detect any tone of about 500Hz on the phone line such as a whistle or DTMF. When this tone is detected, the alarm will turn off. In this case the hardware does the detection.
The circuit amplifies the signal on the phone line and this turns on the second transistor. On the output is a 1u electrolytic. The stage sits with the collector at rail voltage, due to the biasing components keeping the first transistor on and the second transistor off. When a signal is delivered, the first transistor turns off and the collector of the second transistor goes low. This causes the electrolytic to discharge. This will be detected by the micro as a LOW.

3. THE DTMF WAVE-SHAPING CIRCUIT

Dialing a phone number is carried out by sending a tone down the line. So that whistling can not carry out a dialing operation, the telephone company decided to make the tone impossible to produce "by accident." Each dialing tone consists of two frequencies, sent at the same time. These frequencies must be in the shape of a sinewave as the detecting device "locks onto" each of the frequencies at the same time and produces a very-fast result. The only problem is a micro can only produce a square wave.
To convert a square wave into a sinewave, we need a wave shaping circuit. In essence this consists of charging and discharging a capacitor with a square wave and "picking off" the waveform.
The charging of a capacitor is exponential but if we take the beginning of the curve and compare it to a sinewave, the two match up fairly closely.
That's what we have done. We have charged a 100n capacitor very quickly via a 1k resistor so that it is nearly fully charged and then we begin to discharge it. The result is a fairly "peaky" sine wave. The waveform is picked off the capacitor via a 4k7 resistor and passed into an amplifier transistor (same transistor that amplifies the audio at the target zone). The two tones are produced at the same time by the micro and combined after the square waves have been shaped.
The component values have been especially chosen to produce the required sinusoidal waveform. The 10u on the output is very critic al as it determines the amplitude of the DTMF as well as the shape of the signal.

Getting the DTMF tone generator to work was one of the most difficult parts of this project as the tone detectors at the exchange are very "exacting" and critical.
Although we have generated ten tones in the micro, there are tone-generating chips that produce 16 tones, while only 12 tones are used on the telephone keypad. The additional 4 tones are shown on the diagram below as A, B, C and D. The two symbol keys are called "star" (*) and "hache" (hash) key # (also known as the pound key).
The extra tones can be generated by the program but are not needed in our situation. In the early days of

DTMF, the 4 extra tones were used by the telephone companies to route the calls and create call-charges. The basis of defeating these charges was through "blue boxes" held to the mouth-piece, while creating the extra tones. Things have been tightened up since then.

Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

4. THE HIGH GAIN AMPLIFIER

The high gain amplifier is the two-transistor amplifier at the bottom-right of the circuit. It is used to pick up sounds in the target area during an alarm activation. It is directly coupled to the phone line via a Darlington transistor and bridge. The bridge delivers the correct polarity to the circuit, irrespective of the polarity of the phone line and the change in impedance of any of the components connected to the phone line will result in a signal being sent down the line. The output stage of the high-gain amplifier is one of these components and it is biased ON via a 220k resistor. This turns it ON only very slightly, so that the audio signal will drive it correctly. The "load" for the transistor is all the other components connected in series with the transistor and this includes the "holding-in" relay and any isolating transformer at the exchange.
So, we have a two-transistor high-gain amplifier. A 20mV signal from the microphone will produce a 1,000mV signal on the collector of the first transistor and this will be passed to the output transistor.
The amplitude of the waveform across the output transistor is about 2-3v.
The unusual layout of the circuit may be confusing. The pre-amplifier section is powered from the micro while the output transistor is driven from the phone line and the AC signal through the 100n is amplified by the buffer (output) transistor.
The audio amplifier is turned off when the DTMF tone is sent down the line and when a "turn-off tone" is being detected from the receiving party.

5. THE MICROCONTROLLER

The heart of the project is the microcontroller. It is a 18-pin chip with 15 input/output lines and one input-only line (RA5 - pin 4) and one line that is "input and half-output" (RA4 - pin 3). The output lines change from low-to-high-to-low very quickly and each line can deliver a maximum of 25mA.
The program inside the micro determines what happens on each of the lines and the parts around the micro are merely interfacing components. In other words they adapt or modify or amplify a signal to suit the micro or phone line.
The micro never stops "running" and it executes instructions at the rate of one million per second (1 MIPS).
You need to understand PIC language to program the micro and Talking Electronics has produced PIC Programming pages on the web to help you develop a program.

6. THE PROGRAMMING BUTTONS

The 4 programming buttons are connected to a single line and a button-press is determined by the length of time it takes to discharge the 100n capacitor.
The capacitor is firstly charged by making the line a "high-output" and then turning it into an input and testing it at regular intervals to see when it is low.
We have already calculated how long it should take for the various buttons to discharge the 100n and we look at these intervals. But we don't know if a button has been pressed at exactly the beginning of the discharge cycle or part-way through. This will give a false reading. So we look initially to see if any of the buttons have been pressed and then repeat the cycle knowing the button has already been pressed.
The resistor values have been chosen to give different timing intervals for each button.

INSERTING A PHONE NUMBER

The micro is fully programmed but no phone numbers have been placed in EEPROM.
However we have placed a series of 1's to represent the first phone number and 2's to represent the second phone number.
The first thing you must do is install your own numbers.

ADDING A NEW PHONE NUMBER

1. Press first button (1st Phone No) for first phone number or 2nd button (2nd Phone No) for second number.
2. Keep first or second button pressed for 10 seconds. The first or second phone number will appear on the

7-segment display and then the bottom, middle, top segments will illuminate to show the number has been erased. You will now see the lower segment illuminated.
3. Use the **Incr Digit** button to scroll though the numbers.
4. Press **Store** when appropriate digit shows on 7-segment display.
5. When all the digits of the phone number have been inserted, turn the project OFF. Turn on again.

VIEWING NEW NUMBER

1. Press first button for half-second for (1st Phone No) for first phone number or press 2nd button for half-second for (2nd Phone No) for second number. Pressing longer than 5 seconds will erase the number.

CHANGING A NUMBER

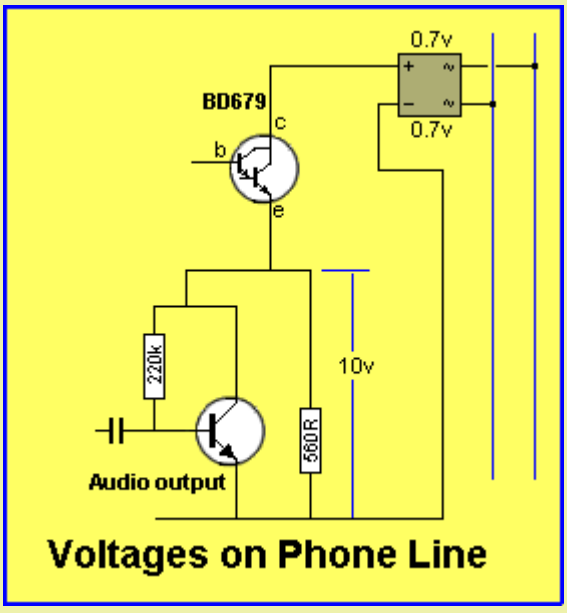
1. Press first button (1st Phone No) for first phone number to be changed or press 2nd button (2nd Phone No) for second number.
2. Keep first or second button pressed for 10 seconds. The first or second phone number will appear on the 7-segment display and then the bottom, middle, top segments will illuminate to show the number has been erased. You will now see the lower segment illuminated.
3. Use the Incr Digit button to scroll though the numbers.
4. Press Store when correct digit shows on 7-segment display.
5. When all the digits of the phone number have been inserted, turn the project OFF. Turn on again.

Note:

If the display shows "junk" or dashes when button 1 or 2 is pressed, press 1st or 2nd button for 10 seconds to clear the display. You will now be in "programming mode" and can insert new phone number. Turn project off when finished and turn it on. Dial Alarm-2 is now "armed."
Dial Alarm-2 does not have a delete or cancel button when in the programming mode.
If you make a mistake when adding a new phone number, turn the project OFF and ON. Push 1st or 2nd button for 10 seconds to get into programming mode and go to the instructions: Adding A New Phone Number, above.

THE PHONE VOLTAGE

Before designing any project for operation on the phone line, you have to understand how the 50v line operates. It's not like a normal 50v power supply. You cannot simply design something for 50v on your bench power-supply and connect it to the phone line.
The phone line is a 50v battery (actually slightly higher than 50v - about 52v - however some of the newer phone systems deliver a voltage as low as 35v - 40v) with a 1k relay in series with one line. When you short the two phone lines together, the relay pulls in to indicate the handset has been lifted. Under these circumstances the current flowing through the line will be 50/1,000 = 50mA. The relay will drop out at 15mA and so you can add devices to the phone line until the current falls to about 15mA without the line dropping out. It is best to keep the current high to prevent the line dropping out.

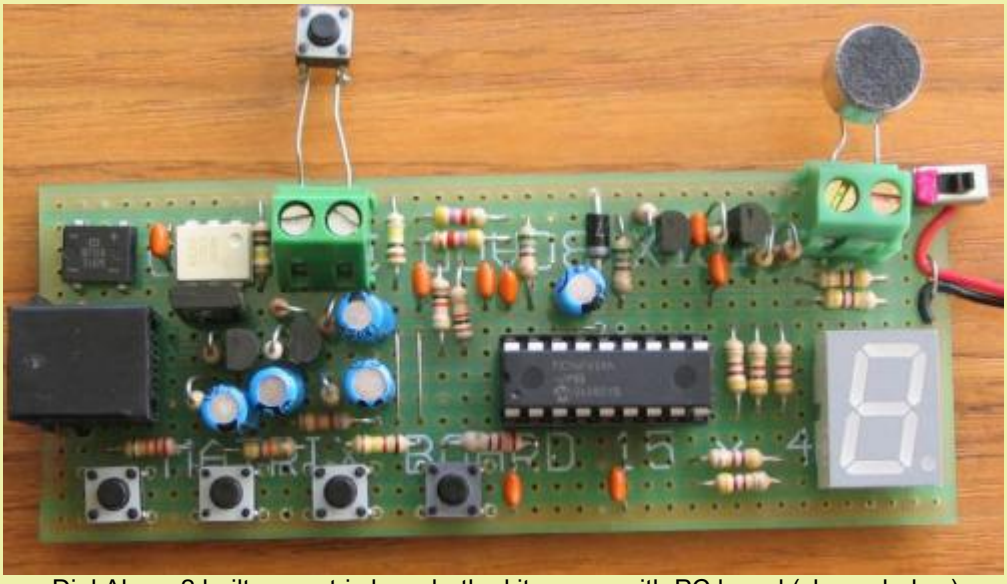


Most phones drop about 8 - 12v across them when they are working and this voltage can be used by the phone for the amplifying circuits, tone generators etc. Our design has a separate supply, however it could be

designed to use the phone voltage, if you wish. The 8v-10v across the audio output transistor gives it plenty of voltage for a good waveform. The audio is sensitive enough to hear a clock ticking in the target area.
The 10v is produced by the 560R resistor plus the effective resistance of the audio output transistor that has been turned on slightly via the 220k base-bias resistor.
The DTMF transistor is also turned on and this provides a load that has an effect on producing the 8-10v we need to keep the line "active."

BUILDING THE PROJECT

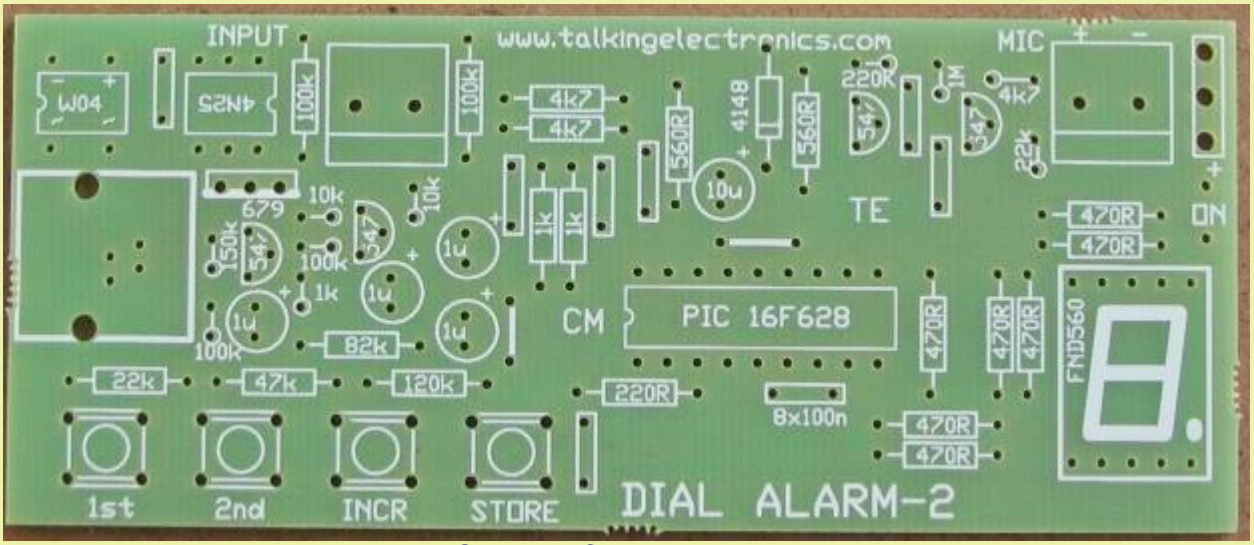
All the components fit onto a PC board labelled **Dial Alarm-2**. The placement of each component is clearly shown by the overlay on the board and the only component requiring careful attention is the bridge.
The bridge has positive (+) and negative (-) marked on the top of the device as well as AC inputs indicated by squiggle lines. Here is the original prototype. It has been designed with exactly the same layout as the PC board (shown below) to make it easy to design the board and prevent any mistakes. This is one of the secrets of "getting things right."



Dial Alarm-2 built on matrix board - the kit comes with PC board (shown below)

1st Phone No.	2nd Phone No.	Incr Digit	STORE
Push 10 sec to delete			Sw off when finished

Button details



PC board for Dial Alarm-2

The outside case of the electret microphone must go to the negative rail. The microphone can be fitted to a short length of twin lead or fine screened microphones lead (as supplied in the kit) so it can be positioned near the audio you wish to detect.
Solder the 8-pin IC socket for the microcontroller to the board so that the cut-out, covers the cut-out on the board. This way the chip will always be fitted around the correct way.
The 4-core telephone cable comes with 4-pin plugs crimped on each end. A 4-pin modular telephone socket is soldered to the board.

Dial Alarm-2

PARTS LIST

Cost $45.75 plus $6.50 postage
or the pre-programmed chip as a separate item is $15.00 plus $5.00 post

1 - 220R 1/4 watt
7 - 470R "
2 - 560R "
3 - 1k "
3 - 4k7 "
2 - 10k "
2 - 22k "
1 - 47k "
1 - 82k "
5 - 100k "
1 - 120k "
1 - 220k "
1 - 1M "

8 - 100n ceramics (monoblock)
4 - 1u 25vw electrolytics
1 - 10u 25vw electrolytics

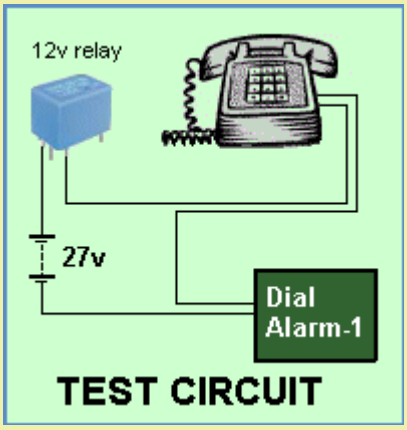
1 - 1N4148 signal diode
4 - BC 547 transistors or similar
1 - BD 679 transistor

1 - 7-segment display FND 500 or 560
1 - electret microphone insert
1 - DF 04 bridge
1 - -4N24 opto-coupler
1 - 4 pin US phone socket
4 - tactile switches
1 - 18 pin IC socket
1 - 4-cell AA battery holder
1 - on/off slide switch
4 - AA cells
2 - 2-way terminal blocks
3m - 4-core telephone cable with plugs crimped on the ends
1m - fine screened microphone lead
1m - fine tinned copper wire
1m - fine enamelled wire
2m - very fine solder
1 - PIC16F628A - Alarm2 (full version)
1 - Dial Alarm-2 PC board

Buy a kit

TESTING THE PROJECT

The project is tested either on a 50v line or the **Test Circuit** shown in the diagram below. The supply is three 9v batteries.



It does not matter which way around the phone or Dial Alarm-2is connected as both have a diode bridge to accept either polarity. When the mercury switch is activated, the alarm sends a Hee Haw tone down the line and this is detected by listening to the line via another telephone connected in series with the **Dial Alarm-2** project, as shown in the diagram above. The audio from the room is then sent down the line. After 15 seconds the Hee Haw is produced again over the audio and this is repeated at a further 15 seconds. The project then closes down, waits a few seconds then dials the second number and repeats the operation.
The two numbers are dialled again and the Alarm closes down.
You can repeat the sequence in the Test Circuit and during the listening period, push any of the buttons on the phone to send a DTMF tone down the line.
The project will close down.
If the input of the alarm is connected to a reed or mercury switch on a door, the door will have to be closed again to produce another trigger pulse.

IF THE PROJECT DOESN'T WORK

If the project doesn't work you will have to go to one of the following sections:

1. The turn-on circuit.

The project turns ON when a trigger pulse is detected on RA5 or RA7. RA5 detects a LOW and RA7 detects a HIGH. Make sure the project is in "Active" mode by pressing button 1 and viewing the 1st phone number. Make sure RA5 is HIGH when the project is in active-mode and RA7 is LOW. The trigger pulse will be very brief.

2. The tone detector amplifier

The operation of the Tone Detector circuit is very clever. The quiescent conditions are set by the two 100k biasing resistors. This puts the base of the first transistor at mid rail.
The emitter will be slightly less than this and the first transistor will be turned ON and the second transistor will be OFF.
When a signal is delivered to the first transistor, some part of the waveform will turn the transistor OFF and the emitter will drop just like an emitter-follower. The base of the second transistor will remain fixed by the 10u and when the first transistor turns off, the current though the lower 10k resistor will reduce and the voltage across this resistor will reduce. This means the base-emitter voltage of the second transistor will increase. This will turn on the second transistor and it will discharge the 1u. The 1u will discharge faster than it can charge via the 10k resistor and this will result in a LOW and be detected by the micro.
This means the signal actually passes through the stage via the emitters and not via the base of the second transistor. This circuit is a form of Schmitt Trigger. The voltage across the 1u is very easy to detect via a multimeter.

3. The DTMF Section.

The quickest way to determine if this section is working is to pick up the phone and activate the alarm, by tilting the mercury switch.
You will hear the DTMF tone being sent down the line if you have the project on a test-rig."
If these tones are not heard, you can produce a constant DTMF tone for say "0" by inserting the following instructions into the program:
Replace the last instruction for the sub-routine DTMF with: goto DTMF The first data-byte in EEPROM must be 3fh. Make sure they are removed after the testing is complete.

Placing a piezo between pin 17 and ground will allow you to hear one of the tones and between pin 18 and ground, the other tone.
The tones will be produced continuously and you can view them on a CRO and observe their wave-shape entering the phone line.

4. The High-Gain Audio Amplifier

The audio amplifier consists of two stages. The pre-amplifier (the low-signal stage) and the buffer stage (output stage).
The pre-amplifier section consists of a standard common-emitter amplifier with AC coupling (capacitor coupling) to the microphone.
You will need either a CRO or an audio tracer to listen to or observe the signal from the microphone through to the output transistor.
Our circuit had a gain of 50, with a 20mV signal (whistle) from the microphone producing 1,000mV (1v) signal into the base of the pre-amplifier stage. The output transistor amplifies this to produce a signal of about 3v on the phone line. You will need a CRO to view the waveforms if you think the audio stages are not operating correctly. A dual-trace CRO is best so you can observe the input and output of a particular stage at the same time.
This completes the coverage of all the individual building blocks in the circuit. If a fault still persists, the best way to tackle the problem is to get another electronics person to check the board. It may be a simple mistake such as swapping two components, a solder bridge or dry joint.
As a last resort, you can build another kit and with the second project working, compare the two.

THE PROGRAM

The program presented in this article has reduced features. The full version comes with the kit. This version only dials one number and does not detect a whistle to turn off. A PIC16F628A containing the full program is available for $15.00 plus $5.00 post - if you want to provide your own parts and PC board. A full kit is available by **emailing** Colin Mitchell.
However this program will provide a lot of features for experimenting.
The program does not follow the conventional line of looping Main and calling sub-routines. This program loops Main and goes to a sub-routine, depending on the result of a button press or trigger. From the sub-routine the micro goes to another sub-routine, depending on 1st number or 2nd number requirement and will go to a loop, requiring the project to be turned off. Refer to the following diagram to see how the micro advances through the various sub-routines. If a trigger pulse is detected, the micro will advance down Main and dial the two numbers (twice) and produce Hee Haw and listen for a tone to turn off. At the end, the project will return to "active state," waiting for another trigger pulse.

Here is the file for Dial Alarm-2, in 4 different forms. The program is a reduced version for experimenting. Full version is supplied in the Dial Alarm-2 kit.
DialAlarm2-1No.asm
DialAlarm2-1No.hex
DialAlarm2-1NoAsm.txt
DialAlarm2-1NoHex.txt

```
;*************************************************************
; Started 15/8/2009    Reduced version 6-10-2009
;  DIAL ALARM-2  with Hee Haw
;This is a reduced version of the program.
; It does not dial the second number and does not turn off
; via whistle or DTMF. It only detects a LOW trigger input on RA5.
;A PIC16F628A containing the full program is available for $15.00 plus $5.00 post, if you
; want to provide your own parts and PC board. A full kit is available by emailing Colin Mitchell.
;
;Port B drives 7 segment display
;
;Project comes on with a number for first and second phone number.
```

```
;To clear fist phone number, push "First Phone Number for 10 secs
; "_" will appear 16 times then "_" "-" "-" This indicates number erased
;and "incr digit" will work for first phone number. Store each number
;and switch project off when finished. New number will now be available
;for dialling. If fist number is "_" you must hold first sw for 10 secs to
;get into "Incr digit" to produce first number via incr Digit and Store.
;
;*****************************************************************

        list P = 16F628         ;microcontroller
        include         ;registers for F628

        __Config        _cp_off & _lvp_off & _pwrte_on
                & _wdt_off & _intRC_osc_noclkout & _mclre_off

        errorlevel -302         ;remove message about using proper bank


;code protection - off
;low-voltage programming - off
;power-up timer -  on
;watchdog timer - off
;use internal RC for 4MHz - all pins for in-out


;*****************************************************************
; variables - names and files
;*****************************************************************


                ;Files for F628 start at 20h


temp1           equ 20h ;for delay and Hee Haw
temp2           equ 21h ;for delay and Hee Haw
temp3           equ 22h ;for delay
Sw_Flag         equ     26h     ;
count           equ     27h     ;loops of discharge time for 100n

tempA           equ     29h     ;temporary storage for EEPROM read
tempB           equ     2Ah     ;temporary storage for EEPROM read
pointer         equ     2Bh     ;pointer for table

carrier         equ     2Ch     ;carrier file
lowtone         equ     2Dh     ;low tone
lowdec          equ     2Eh     ;decrementable low tone
hightone        equ     2Fh     ;high tone
highdec         equ     30h     ;decrementable high tone
epromstart      equ     31h     ;eeprom start address
loops           equ     32h     ;dtmf loops     and others
secondNo        equ     33h     ;to ring second number
secondtime      equ     34h     ;to ring second time
count2          equ     35h     ;counter to show "-" when alarm is working

;*****************************************************************
;Equates
;*****************************************************************
status  equ     0x03
cmcon   equ     0x1F
rp1     equ     0x06
rp0     equ     0x05
portA   equ     0x05
portB   equ     0x06
trisA   equ     0x85
trisB   equ     0x86
```

```
;*************************************************************
;Beginning of program
;*************************************************************
reset   org     00      ;reset vector address
        goto    SetUp


table1  addwf   PCL,F           ;02h,1  add W to program counter
        retlw   b'00001000'     ; "_"   -|-|-|D|-|-|- ready for number 08h
        retlw   b'00001000'     ; "_"   -|-|-|D|-|-|- ready for number 08h
        retlw   b'00111111'     ; "0"   -|F|E|D|C|B|A  3Fh
        retlw   b'00000110'     ; "1"   -|-|-|-|C|B|-  06h
        retlw   b'01011011'     ; "2"   G|-|E|D|-|B|A  5Bh
        retlw   b'01001111'     ; "3"   G|-|-|D|C|B|A  4Fh
        retlw   b'01100110'     ; "4"   G|F|-|-|C|B|-  66h
        retlw   b'01101101'     ; "5"   G|F|-|D|C|-|A  6Dh
        retlw   b'01111101'     ; "6"   G|F|E|D|C|-|A  7Dh
        retlw   b'00000111'     ; "7"   -|-|-|-|C|B|A  07h
        retlw   b'01111111'     ; "8"   G|F|E|D|C|B|A  7Fh
        retlw   b'01101111'     ; "9"   G|F|-|D|C|B|A  6Fh



        ;Table2 DTMF Low tones

Table2  ADDWF 02,1      ;
        NOP
        retlw 075h      ;1
        retlw 075h      ;2
        retlw 075h      ;3
        retlw 06Ah      ;4
        retlw 06Ah      ;5
        retlw 06Ah      ;6
        retlw 05Fh      ;7
        retlw 05Fh      ;8
        retlw 05Fh      ;9
        retlw 056h      ;0


        ;Table3 DTMF HIGH tones

Table3  ADDWF 02,1      ;
        NOP
        retlw 044h      ;1
        retlw 03Dh      ;2
        retlw 037h      ;3
        retlw 043h      ;4
        retlw 03Dh      ;5
        retlw 037h      ;6
        retlw 043h      ;7
        retlw 03Dh      ;8
        retlw 037h      ;9
        retlw 03Ch      ;0


;*************************************************************
;* port A and B initialisation
;Button Up  and Button Down recognised when project turned on.  *
;*************************************************************

SetUp   bsf     status,rp0
        clrf    trisA           ;Make all RA output
        clrf    trisB           ;Make all RB output
        bsf     trisA,5         ;Make RA5 input for LOW trigger
```

```
        bsf     trisA,6         ;make RA6 input for tone detect
        movlw   b'10000000'     ;Turn off T0CKI, prescale for TMR0 = 1:
        movwf   option_reg
        bcf     status,rp0      ;select programming area - bank0
        clrf    portA           ;Clear Port A of junk
        clrf    portB           ;Clear Port B of junk
        clrf    Sw_Flag
        clrf    pointer
        movlw   07h             ;turn comparators off and enable
        movwf   cmcon           ;    pins for I/O functions
        goto    Show            ;Show "dIAL-2"


                ;Delays

_1mS    nop
        decfsz  temp1,f
        goto    _1mS
        retlw   00

_4mS    movlw   04h
        movwf   temp2
_b      nop
        decfsz  temp1,f
        goto    _b
        decfsz  temp2,f
        goto    _b
        retlw   00

_10mS   movlw   0Ah
        movwf   temp2
_c      nop
        decfsz  temp1,f
        goto    _c
        decfsz  temp2,f
        goto    _c
        retlw   00



_50mS   movlw   40h
        movwf   temp2
_d      nop
        decfsz  temp1,f
        goto    _d
        decfsz  temp2,f
        goto    _c
        retlw   00


_100mS  movlw   64h
        movwf   temp2
_e      nop
        decfsz  temp1,f
        goto    _e
        decfsz  temp2,f
        goto    _e
        retlw   00

_250mS  movlw   0FFh
        movwf   temp2
_ee     nop
        decfsz  temp1,f
        goto    _ee
        decfsz  temp2,f
```

```
        goto    _ee
        retlw   00



_750mS  goto    $+1
        goto    $+1
        goto    $+1
        goto    $+1
        decfsz  temp1,1
        goto    _750mS
        decfsz  temp2,1
        goto    _750mS
        retlw   00

_2Sec   movlw   05h
        movwf   temp3
        movwf   temp2
_2      nop
        decfsz  temp1,f
        goto    _2
        decfsz  temp2,f
        goto    _2
        decfsz  temp3,f
        goto    _2
        retlw   00

_5Sec   movlw   0Ch
        movwf   temp3
        movwf   temp2
_5      nop
        nop
        nop
        nop
        decfsz  temp1,f
        goto    _5
        decfsz  temp2,f
        goto    _5
        decfsz  temp3,f
        goto    _5
        retlw   00


        ;clear1  clears the first phone number in EEPROM (from Read1) then
        ;   goes to incrA

clear1  movlw   08h             ;scans display "_" "-"  "-"
        movwf   portB
        call    _100mS
        movlw   40h
        movwf   portB
        call    _100mS
        movlw   01h
        movwf   portB
        call    _100mS
        movlw   00h
        movwf   tempA
        movlw   0Fh
        xorwf   tempA,0
        btfsc   status,2        ;zero flag in status file. Set if pointer is 0Fh
        goto    incrA           ;exits when 0Fh locations have "_" at each address
        movf    tempA,0
        bsf     status,rp0      ;select bank1
        movwf   eeadr           ;
        movlw   08h             ;put a "_" at each address
```

```
        movwf   eedata          ;
        bcf     status,rp0      ;select bank0
        call    write
        incf    tempA,1
        goto    $-.12


                ;clear2  clears the second phone number in EEPROM (from Read2) then
                ;   goes to incrB

clear2  movlw   08h
        movwf   portB
        call    _100mS
        movlw   40h
        movwf   portB
        call    _100mS
        movlw   01h
        movwf   portB
        call    _100mS
        movlw   20h
        movwf   tempA
        movlw   2Fh
        xorwf   tempA,0
        btfsc   status,2        ;zero flag in status file. Set if pointer is 2Fh
        goto    incrB           ;exits when 0Fh locations have "_" at each address
        movf    tempA,0
        bsf     status,rp0      ;select bank1
        movwf   eeadr           ;
        movlw   08h             ;put a "_" at each address
        movwf   eedata          ;
        bcf     status,rp0      ;select bank0
        call    write
        incf    tempA,1
        goto    $-.12

                ;dials the whole DTMF phone number and returns on finding "_" (08h)
                ;only start of EEPROM is needed  00h for 1st No and 20h for 2nd No

Dial    movf    epromstart,0    ;eeprom start address
        bsf     status,rp0
        movwf   EEADR
        bsf     EECON1,0        ;starts EEPROM read operation. Result in EEDATA
        movf    EEDATA,0        ;move read data into w
        bcf     status,rp0
        movwf   tempA           ;convert display value in tempA to 0-9 in "carrier"
        movlw   08h
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        retlw   00              ;end of number detected (08h) return
        movlw   01h
        movwf   carrier         ;create a value 0-9
        movlw   06h
        xorwf   tempA,0         ;compare tempA with 06h
        btfsc   status,2        ;zero flag in status file. Set if pointer is 06h
        goto    dddd
        incf    carrier,1
        movlw   5Bh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   4Fh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
```

```
        incf    carrier,1
        movlw   66h
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   6Dh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   7Dh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   07h
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   7Fh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   6Fh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
        movlw   3Fh
        xorwf   tempA,0         ;
        btfsc   status,2        ;
        goto    dddd
        incf    carrier,1
dddd    movf    carrier,0       ;carrier will be 1,2,3 ..0A
        call    table2          ;Get low-tone value
        movwf   lowtone         ;Put low-tone into low
        movwf   lowdec          ;Decrementable low-tone
        movf    carrier,0
        call    table3
        movwf   hightone        ;Put high-tone into high
        movwf   highdec         ;Decrementable high-tone
        call    DTMF
        call    _100mS          ;100mS delay between tones
        call    _100mS          ;100mS delay between tones
        incf    epromstart,1
        goto    dial



DTMF    movlw   0FFh            ;80 loops of tone to produce 1/10th sec
        movwf   loops           ;temp file for decrementing dtmf
        decfsz  highdec,1
        goto    $+5
        movlw   01h             ;to toggle RA0
        xorwf   portA,1
        movf    hightone,0
        movwf   highdec
        decfsz  lowdec,1
        goto    $-7
        movlw   02h             ;to toggle RA1
        xorwf   portA,1
        movf    lowtone,0
```

```
        movwf   lowdec
        decfsz  loops,1
        goto    $-.13
        bsf     portA,0         ;turn on output to keep Alarm on line
        bcf     portA,1
        retlw   00


                ;HeeHaw produces alarm-sound via opto coupler

Hee     movlw   0FFh
        movwf   temp1
        movlw   0C0h
        movwf   temp2
        bsf     portB,7
        nop
        decfsz  temp2,1
        goto    $-2
        movlw   0C0h
        movwf   temp2
        bcf     portB,7
        nop
        decfsz  temp2,1
        goto    $-2
        decfsz  temp1,1
        GOTO    $-.13

Haw     movlw   0C0h
        movwf   temp1
        movlw   0FFh
        movwf   temp2
        bsf     portB,7
        nop
        decfsz  temp2,1
        goto    $-2
        movlw   0FFh
        movwf   temp2
        bcf     portB,7
        nop
        decfsz  temp2,1
        goto    $-2
        decfsz  temp1,1
        goto    $-.13
        bcf     portB,7 ;Keep opto-coupler ON after Hee Haw
        retlw   00


                ;incr digits for first phone number (from Clear1)
                ;then detects "store."   Turn off project when finished new number.

incrA   movlw   08h
        movwf   portB           ;produce "_"
        movlw   01h
        movwf   pointer         ;pointer looks at table1 value
        call    Sw
        btfsc   Sw_Flag,3       ;has "Incr Digit" been pressed?
        call    Up
        call    Sw
        btfsc   Sw_Flag,7       ;has "Incr Digit" been released?
        goto    $-2
        call    Sw
        btfss   Sw_Flag,4       ;has "store" been pressed?
        goto    $-8
        movlw   00h
        movwf   tempA
```

```
        movf    tempA,0         ;find first blank location
        bsf     status,rp0
        movwf   EEADR           ;find first location 00  in EEPROM
        bsf     EECON1,0        ;starts EEPROM read operation. Result in EEDATA
        movf    EEDATA,0        ;move read data into w
        bcf     status,rp0
        movwf   tempB           ;see if location contains 08h
        movlw   08h
        xorwf   tempB,0         ;compare tempB with 08h
        btfsc   status,2
        goto    $+3
        incf    tempA,1
        goto    $-.12
        movf    pointer,0       ;copy pointer value into w
        call    table1          ;display value will return in w
        bsf     status,rp0      ;select bank1
        movwf   eedata          ;display values are stored in EEPROM
        bcf     status,rp0      ;select bank0
        movf    tempA,0         ;tempA will contain address
        bsf     status,rp0      ;select bank1
        movwf   eeadr           ;
        bcf     status,rp0      ;select bank0
        call    write
        goto    incrA

                ;incr digit for second phone number (from Clear2)
                ;then detects "store."   Turn off project when finished new number.

incrB   movlw   08h
        movwf   portB           ;produce "_"
        movlw   01h
        movwf   pointer         ;pointer looks at table1 value
        call    Sw
        btfsc   Sw_Flag,3       ;has "Incr Digit" been pressed?
        call    Up
        call    Sw
        btfsc   Sw_Flag,7       ;has "Incr Digit" been released?
        goto    $-2
        call    Sw
        btfss   Sw_Flag,4       ;has "store" been pressed?
        goto    $-8
        movlw   20h
        movwf   tempA
        movf    tempA,0         ;find first blank location
        bsf     status,rp0
        movwf   EEADR           ;find first location 20  in EEPROM
        bsf     EECON1,0        ;starts EEPROM read operation. Result in EEDATA
        movf    EEDATA,0        ;move read data into w
        bcf     status,rp0
        movwf   tempB           ;see if location contains 08h
        movlw   08h
        xorwf   tempB,0         ;compare tempB with 08h
        btfsc   status,2
        goto    $+3
        incf    tempA,1
        goto    $-.12
        movf    pointer,0       ;copy pointer value into w
        call    table1          ;display value will return in w
        bsf     status,rp0      ;select bank1
        movwf   eedata          ;display values are stored in EEPROM
        bcf     status,rp0      ;select bank0
        movf    tempA,0         ;tempA will contain address
        bsf     status,rp0      ;select bank1
        movwf   eeadr           ;
        bcf     status,rp0      ;select bank0
```

```
        call    write
        goto    incrB


                ;Read First Phone Number then to Clear1 if more than 10 secs

Read1   movlw   00h             ;first address in EEPROM for second number
        movwf   tempA
        movlw   0Fh
        xorwf   tempA,0
        btfsc   status,2        ;zero flag in status file. Set if pointer is 0Fh
        goto    $+.13
        movf    tempA,0
        bsf     status,rp0
        movwf   EEADR
        bsf     EECON1,0        ;starts EEPROM read operation. Result in EEDATA
        movf    EEDATA,0        ;move read data into w
        bcf     status,rp0
        movwf   portB           ;data stored as values for 7-seg display
        call    _750mS
        clrf    portB
        call    _100mS
        incf    tempA,1
        goto    $-.15

        call    Sw                      ;see if sw pressed for 10 secs
        btfss   Sw_Flag,1
        retlw   00
        call    _750mS
        call    Sw
        btfss   Sw_Flag,1
        retlw   00
        call    _750mS
        call    Sw
        btfss   Sw_Flag,1
        retlw   00
        call    _750mS
        call    Sw
        btfss   Sw_Flag,1
        retlw   00
        goto    clear1          ;clear first phone number


        ;Read second Phone Number then to Clear2 if more than 10 secs


Read2   movlw   20              ;first address in EEPROM for second number
        movwf   tempA
        movlw   2Fh
        xorwf   tempA,0
        btfsc   status,2        ;zero flag in status file. Set if pointer is 2Fh
        goto    $+.13
        movf    tempA,0
        bsf     status,rp0
        movwf   EEADR
        bsf     EECON1,0        ;starts EEPROM read operation. Result in EEDATA
        movf    EEDATA,0        ;move read data into w
        bcf     status,rp0
        movwf   portB           ;data stored as values for 7-seg display
        call    _750mS
        clrf    portB
        call    _100mS
        incf    tempA,1
        goto    $-.15
```

```
        call    Sw              ;see if sw pressed for 10 secs
        btfss   Sw_Flag,2
        retlw   00
        call    _750mS
        call    Sw
        btfss   Sw_Flag,2
        retlw   00
        call    _750mS
        call    Sw
        btfss   Sw_Flag,2
        retlw   00
        goto    clear2          ;clear second phone number


                ;Shows "dIAL-2" on start-up

Show    call    _750mS          ;delay to allow micro to start-up
        movlw   b'01011110'     ; "d"   G|-|E|D|C|B|-
        movwf   portB
        call    _750mS
        movlw   b'00000110'     ; "I"   -|-|-|-|C|B|-
        movwf   portB
        call    _750mS
        movlw   b'01110111'     ; "A"   G|F|E|-|C|B|A
        movwf   portB
        call    _750mS
        movlw   b'00111000'     ; "L"   -|F|E|D|-|-|-
        movwf   portB
        call    _750mS
        movlw   b'01000000'     ; "-"   G|-|-|-|-|-|-
        movwf   portB
        call    _750mS
        movlw   b'01011011'     ; "2"   G|-|E|D|-|B|A
        movwf   portB
        call    _750mS
        clrf    portB
        bsf     portB,7                 ;turn off optocoupler
        goto    Main


                ;Sw subroutine generates bit 1,2,3,4 in Sw_Flag file

Sw      bsf     status,rp0
        bcf     trisA,3                 ;Make bit 3 output
        bcf     status,rp0
        bsf     portA,3                 ;make bit 3 HIGH
        call    _1mS                    ;create delay to charge 100n
        bsf     status,rp0
        bsf     trisA,3                 ;Make bit 3 input
        bcf     status,rp0
        call    _10mS
        call    _10mS
        btfss   portA,3                 ;if set, no sw pushed
        goto    $+3                     ;sw pushed
        clrf    Sw_Flag                 ;no sw pressed
        retlw   00                              ;
        btfsc   Sw_Flag,7               ;test "first-pass" sw flag
        retlw   00
        clrf    count
        bsf     status,rp0
        bcf     trisA,3                 ;Make bit 3 output
        bcf     status,rp0
        bsf     portA,3                 ;make bit 3 HIGH
        call    _1mS                    ;create delay to charge 100n
```

```
        bsf     status,rp0
        bsf     trisA,3                 ;Make bit 3 input
        bcf     status,rp0
        call    _4mS
        incf    count,f
        btfsc   portA,3                 ;is input HIGH?
        goto    $-3                             ;count exits with 1-5
        bsf     Sw_Flag,0               ;set button-pushed flag

        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,1               ;set a flag-bit for first sw
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,2               ;set a flag-bit for second sw
        retlw   00
        decfsz  count,f
        goto    $+3
        bsf     Sw_Flag,3               ;set a flag-bit for third sw
        retlw   00
        bsf     Sw_Flag,4               ;set a flag-bit for fourth sw
        retlw   00


Up      incf    pointer,1
        movlw   0Ch                             ;put 13 into w
        xorwf   pointer,0       ;compare pointer with 12
        btfsc   status,2        ;zero flag in status file. Set if pointer is 12
        clrf    pointer
        movf    pointer,0       ;copy unit value into w
        call    table1          ;display value will return in w
        movwf   portB
        bsf     Sw_Flag,7       ;set "first-pass" sw flag
        retlw   00

write   bsf     status,rp0      ;select bank1
        bsf     eecon1,wren     ;enable write
        movlw   55h             ;unlock codes
        movwf   eecon2
        movlw   0aah
        movwf   eecon2
        bsf     eecon1,wr       ;write begins
        bcf     status,rp0      ;select bank0
writeA  btfss   pir1,eeif       ;wait for write to complete
        goto    writeA
        bcf     pir1,eeif
        bsf     status,rp0      ;select bank1
        bcf     eecon1,wren     ;disable other writes
        bcf     status,rp0      ;select bank0
        retlw   00


;*****************************************************************
;* Main                                                          *
;*****************************************************************

Main    call    Sw
        btfsc   Sw_Flag,1
        call    Read1
        btfsc   Sw_Flag,2
        call    Read2
        btfss   portA,5         ;trigger input for alarm - normally HIGH
        goto    $+8
        decfsz  count2,1        ;dec counter to show "-" alarm working
```

```
        goto    $+4
        bsf     portB,6         ;show "-"
        call    _4mS
        bcf     portB,6         ;clear "-"
        btfsc   portA,5         ;trigger input for alarm - normally HIGH
        goto    Main


                ;secondNo       to ring second number
                ;secondtime     to ring second time

        movlw   14h                     ;show "ii" when dialing numbers
        movwf   portB
Main1   movlw   02
        movwf   secondNo        ;to ring 2nd number
        movwf   secondtime      ;to ring numbers second time
        bcf     portB,7         ;to keep circuit ON
Main1a  call    _5Sec           ;delay before starting to dial
        movlw   00
Main2   movwf   epromstart      ;start of EEPROM for 1st Number
        call    Dial
        movlw   04h                     ;Put 4 loops into W
        movwf   loops
        call    _5Sec           ;5 second delay
        call    Hee                     ;Hee Haw outputs through opto coupler
        call    _750mS          ;for silence
        bsf     portA,2         ;turn on high-gain amplifier
        call    _5Sec           ;listen to target area for sounds
        call    _5Sec           ;listen to target area for sounds
        bcf     portA,2         ;turn off high-gain amplifier
        bsf     portA,7         ;turn on Tone Detector Cct
        call    _2Sec           ;listen for whistle     and let circuit settle
        btfss   portA,6         ;detect DTMF or whistle - will produce a LOW
        goto    Main3
        call    _2Sec           ;listen for whistle
        btfss   portA,6         ;detect DTMF or whistle - will produce a LOW
        goto    Main3
        call    _2Sec           ;listen for whistle
        btfss   portA,6         ;detect DTMF or whistle - will produce a LOW
        goto    Main3
        decfsz  loops,1
        goto    $-.17
        bsf     portB,7         ;hang up phone
        call    _5Sec           ;Delay before ringing
        bcf     portB,7         ;pick up phone line
        decfsz  secondNo,1      ;ring 2nd number
        goto    Main4
        decfsz  secondtime,1    ;Ring number the second time
        goto    Main5

Main3   clrf    portB           ;turn off display
        bcf     portA,7         ;turn off Tone Detector Cct
        bsf     portB,7         ;hang up phone
        GOTO    Main2           ;loop and call numbers again

Main4   movlw   00h             ;to ring phone number again
        goto    Main2

Main5   movlw   02
        movwf   secondNo
        goto    Main1a

;*****************************************************************
;*EEPROM     Values to burn into EEPROM                          *
;*****************************************************************
```

```
        org     2100h   ;16 locations 00 to 0F
        de      06h, 06h, 06h, 3Fh, 06h, 5Bh, 4Fh, 66h, 6Dh,
                                   7Dh, 07h, 7Fh, 6Fh, 08h, 08h, 08h

        org     2120h   ;16 locations 20 to 2F
        de      5Bh, 5Bh, 5Bh, 3Fh, 06h, 5Bh, 4Fh, 66h, 6Dh,
                                   7Dh, 07h, 7Fh, 6Fh, 08h, 08h, 08h

        org     2130h

        de      3Fh

        END
```

MODIFYING THE PROGRAM

To work on the program, you need to buy a PROGRAMMER (PICkit2 - see above or Multi Chip Programmer) and put the chip on a prototyping board with a 5-pin programming socket so it can be "burnt."
The connections for the Programming Socket are shown on the Dial Alarm-2 circuit above.
The next thing you will need is the .asm file and open it in NotePad2. Notepad2 is called a text editor and it will display the code in columns so that each line can be assembled in MPASM to create a .hex file. Call your program by a different name so that you can identify your changes.
MPASM will also produce a .lst file that shows any mistakes you have made. When your program is mistake-free, MPASM will produce a .hex file.
Download MPASM (v02.70), click: MPASM. The latest version of MPASM is very messy. Use the one we provide.

You will also need a text program such as TextPad or NotePad
You cannot use the .asm file above for TextPad as it has added spaces. These spaces will upset MPASM when it tries to compile the file to produce a .hex file. If you get an error on a line (from MPASM) that seems to be correct, try re-typing the line(s) as it may contain unseen spaces!
Then use Chip Programmer - PICkit2 or Multi Chip Programmer - to "burn" the program into the chip. The phone numbers are stored in EEPROM and are changed via software when the project is powered. You don't need to "burn" them into the chip.

If the program does not work, you may be told about and Emulator or Single Stepper, that will "solve all your problems." Let's see . . .

USING AN EMULATOR

An Emulator is a single-stepping program that goes though the code, one instruction at a time, so you can see what is happening.
A Single-Stepper program comes on the CD with PICkit-2 and you can use it if you wish. We have found it helpful in some ways as you can see the contents of each register before and after an instruction has been executed and this will let you know why an instruction may not be working. It is most helpful when executing a Boolean instruction as the result has to be worked out on paper and the single-stepper will let you check the result.
Our method of developing a program is slightly different. We suggest using the "cut and paste" method of creating a project.
Start by using one of the projects already available on our website by clicking HERE and removing the unwanted sub-routines. You will then have a layout for the processor, a few equate files, some sub-routines and Main.
Fir the micro to an experimental PC board containing 5 pins for In-Circuit Programming, add a LED and resistor and create a sub-routine that loops and blinks the LED.
You have now started.
Add a few more lines of code and test its operation.
This is exactly how we start every project.
It's wishful thinking to write a large program and expect it to "run." It possibly won't. Just add a few lines at a time and check the operation.
For the Dial Alarm-2 project, we tested each section separately and this consisted of more than 7 different blocks.
Not only do you have to get the electronics to work but you also need to get the program to interface to the block.
This is where an Emulator or Single-Stepper falls down. It cannot detect if the electronics section is working

and responding to the code.
For instance it cannot detect switch-bounce or if the 7-segment display is showing the correct segments.
Secondly, delays take a long while to execute and either the emulator skips over them or takes a long time to execute.
I have used a single-stepper and emulator for the PIC and these are some of the problems it did not solve.
By far the best method is MINE. It's simple and it works every time. It's back to basics.
If you are having trouble detecting if the micro is advancing though a sub-routine correctly, add an instruction that takes the micro to a small routine that outputs a tone to a piezo diaphragm or blinks a LED.
Put a GOTO instruction, say before a CALL instruction. If the LED blinks, the micro has reached the instruction. Then put the GOTO after the CALL. If the LED does not blink, the micro has not come out of the sub-routine. It may be stuck in the sub-routine or jumped to another address. Go to the sub-routine and work your way through each line with the GOTO concept.
It may be time-consuming but it is the only real way to follow the actual progress of the microcontroller. This approach was used to solve a problem with the original tone routine in the Alarm. The investigation solved the problem and also showed the sub-routine was not well-designed. A much simpler routine was put in its place. So, the hands-on approach solved two things at the same time.
A CRO was also used initially to check the quality of the DTMF waveform. It appeared to be perfect on the screen but was only being accepted by the exchange 80% of the time. By changing the wave-shaping components, the acceptance rose to 100%. The difference between the two waveforms could not be seen on the CRO, but a 8870 DTMF tone detector detected the difference. This is another case of going back to basics and using your knowledge of electronics to improve the quality of a waveform.
The point I am making is this . . .
All the tools of assistance for getting a project up-and-running have been provided in the articles on Talking Electronics website. The only test equipment you need is a multimeter (either analogue or digital) and a Logic Probe. Don't dream: "If only I had an emulator!" or "If only I had a CRO." You can do it all with basics and that is what the Talking Electronics PIC course is all about. Building this project and some of our other projects will show you how things go together, so you can design your own projects.

As I said above, one of the biggest problems is working out the correct order for testing a project. Things have to be done in the correct order and this quite often requires stripping the project down to the simplest circuit. In our case the first section to work on was the DTMF tones. Once these were 100% accepted by the exchange, the turn-on circuit and opto-coupler sections could be added. Then the audio amplifier had to be placed in parallel with the DTMF section without affecting the quality of the waveform of either the tones or the audio. This was quite a challenge and even though the final circuitry is simple, a lot of testing had to be done to make sure other designs were not better. The DTMF circuit was loaded with capacitors and resistors to see if the tone was still recognised by the exchange. This way you know you have a margin-of-error and any tolerances generated in the building of the project will not affect the outcome.
As each problem was solved, the project got nearer completion. By working with basics, the feeling is the project is advancing.
With the Dialling Alarm, there were more than 10 things to sort out.
Producing digits on the 7-segment display
Detecting 4 buttons
Reading and writing to EEPROM
The DTMF tone - duration, amplitude, clarity, getting 100% acceptance on the line,
The Darlington transistor
The audio amplifier, reducing hum, reducing motor-boating, improving output amplitude, gating.
The tone detecting circuit - detecting a tone but not detecting noise or talking
None of these would have been helped with an emulator or CRO. There is too much circuitry interdependence and the big problem with a CRO is the introduction of hum when the earth clip is connected to the project.
If there is any magic package or device that speeds up the process of development, I will let you know.

By adding some instructions to output to a display, you can see if the micro is going past the code you are having trouble with.
We also suggest creating your program by using sub-routines from other projects that are known to work correctly.

There is only one problem with an Emulator or Single-Stepper. They are nice, but relying on them is a **crutch**. You tend to think they will solve your problems.
This is a dangerous misconception because, in most cases the final solution will be to go back to basics.
They **can** be of assistance, but I am going to show the real way to problem-solving is using "tricks-of-the-trade."
The big problem with an emulator is INPUTS. If you have a push button in a circuit, the emulator does not carry out the operation of the push button.

The way to learn programming is to go through a project that has been written by some one else and study each line of code. You need to know two things. Firstly you need to know what each instruction is doing and then you need to know why the instruction has been used. In the Dial Alarm-2 program we have used a very simple format called linear programming.
The sub-routines are long and very few calls are made. In addition we have used very few Boolean commands and the simplest way to convert from a display value to a numerical equivalent.

24/8/09