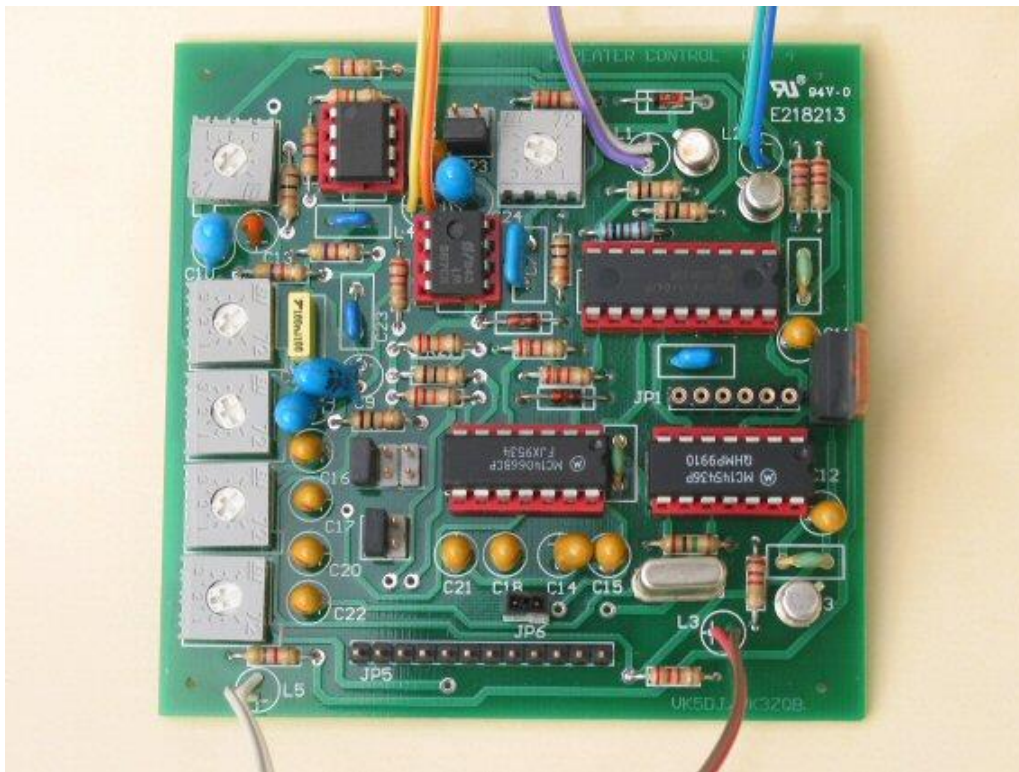# The VK5DJ Repeater Controller

March 2014
16F1827 version 7.06 for Voice ID

by John Drew, 34 Aitken Street, Millicent 5280
john@vk5dj.com

A design for a repeater controller for amateur or professional use. On a board 80mm by 82mm, the controller is a two port design intended to control a duplex repeater with or without a simplex gateway or link.

Based on the 16F1827 PIC it includes a range of remote controllable functions, and has the capacity to interface to either positive or negative active COS or PTT. From Version 7.01 in that a watchdog timer has been added so that in the event of a rare lockup the chip will reset itself after 256 seconds, secondly the chip is now working from the internal oscillator, no longer relying on the coupling from the MC145436. The 16F1827 is a superior chip to the replaced 16F628A and is also cheaper.

Check http://www.vk5dj.com for document and software upgrades

INTRODUCTION

The purpose of this document is to describe a controller for amateur repeaters. The controller and its associated software is capable of supporting a standard duplex repeater, and if required, a link transceiver to another system or a gateway on another band.

The **primary (or repeater) port** interfaces to a receiver supplying
- a mute output (COS) either negative or positive going, and
- received audio (approx 100mV).
- if CTCSS control is required, an additional audio output is required from the discriminator of the receiver.

As the controller manages audio, received signals need not be muted, though it should be noted that excessive audio drive can cause breakthrough across the 4066 CMOS switch.

The **primary (or repeater) port** interfaces to a transmitter
- controls the PTT (either low=operate or high=operate) and
- transmit audio input.

There is no audio amplifier on board so the TX audio input should not require high levels unless the receiver can provide it and it does not cause breakthrough.

The **secondary** (link or gateway) port interfaces to a receiver and transmitter in the same manner as the primary port. A single transceiver is used as either a link system or a gateway to another frequency.

The secondary port may be set to behave as a link (no tail except for a timeout when the morse 'TO' is switched through) or as a gateway where tails are provided, signalling beeps added, identification etc. The behaviour is set by a control byte in the PIC's EEDATA memory (see P10) and may be altered by remote control.

The direction of the mute may be set by an appropriate value in EEDATA (and correct placement of input resistors) at programming time. Similarly, the direction of the PTT lines may be set by an appropriate value in EEDATA inserted at programming time.

A truth table showing controller behaviour is included with this package.

GENERAL DESIGN

Hardware

This version uses the 16F1827 PIC because of its improved and larger memory (room for future improvements) and additional features. As for earlier versions a jumper should be installed under the board to access an additional port for DTMF purposes. See later comment and photo LINK.JPG.
The 16F1827 PIC Microcontroller, one of a family of computers on a chip, uses reprogrammable flash memory. The 'F' in the name stands for 'Flash'. A PIC with a 'C' is a once only programmable device. The 16F1827 has memory space for 4096 commands (14 bit words), it has 256 bytes of non-volatile, electrically erasable data

storage (EEDATA) used for callsign storage, timer delays etc, and 384 bytes of RAM for variable storage. It has two 8-bit bi-directional ports. In practice there are 15 I/O connections available to do clever things with. It all fits in an 18-pin DIL package. The chip will operate up to 32MHz but this design uses an internal 4MHz clock for the PIC16F1827 and a 3.579MHz crystal clock for the MC145436A.

The controller's chip line up is: 16F1827 for control, a MC145436 IC for DTMF detection, a combination of a CA3140 op amp operating as a second order low pass filter and a NE567 to detect CTCSS tones while a 4066 CMOS switch controls audio. A 78L05 provides regulated 5 volts to the ICs.

The choice of a NE567 to decode CTCSS is a practical one as the author had a number on hand, they are cheap and they work OK. There are a small number of specially designed CTCSS chips on the market. They are quite expensive and couldn't be located in Australia. If a reader is interested in using a proprietary CTCSS chip then a small outboard circuit could be developed. The controller board can interface to an external decoder (see jumper JP3 and pin 12 of the external connector).

Software
Rather than use the MPASM assembler directly, I chose to purchase and use the BASIC compiler, PROTON+ from Crownhill (see www.picbasic.org). The compiler produces the asm file for MPASM, which in turn creates the Hex file for the hardware programmer. To program your PIC you need the Hex file included in this package
- V706ctrl.hex

The programmer used in the VK5DJ shack is the PICkit3 available from Microchip Direct at reasonable cost. The free MPLAB suite is the programming software.

General operation
 A minimum of components and cost to achieve the following features:

- Two carrier-operated switch inputs (i.e. a receiver A and receiver B) either +ve or −ve going and programmable at 'burn' time. 'Mute' and 'COS' are used interchangeably in this documentation
- Two PTT outputs (any combination of low or high active programmed at 'burn' time.)
- Onboard DTMF decoder (jumpered to either receiver)
- Onboard CTCSS decoder (to meet licensing conditions or remove interference on main TX or link)
- Capacity to use 1750 tone access (set functions 01,03,05,0A). See Page 11.
- Timeout timer
- Callsign generator with timer (and choice of four identification modes)
- Tail with mode pips (high pitch and low pitch depending on receiver)
- Remote inhibit of timer using either DTMF or CTCSS
- Remote inhibit of primary repeater
- Remote inhibit of gateway (or link)
- Remote control of CTCSS requirement on primary and/or secondary ports
- Remote control of tail beeps (inhibit or allow)
- Remote callsign change and remote suppression of callsign

- Remote adjustment of callsign delay
- Remote control of morse code frequency
- Remote choice of some identification styles to cater for differing needs
- Remote adjustment of timeout period
- Remote adjustment to support either a link or a gateway
- Remote decision on link/gateway on or off with power on
- Remote adjustment of morse code speed
- Remote adjustment of COS trigger (delay opening up to 255msecs)
- Remote adjustment of tail lengths both pre and post beep
- Remote enable and timing of a regular CW message
- Remote change of the CW message
- Remote decision on whether to pass DTMF tones through
- Short tail on link if required
- DTMF can be supplied by external rcvr with/without rptr TX being keyed
- LEDs to indicate power on, DTMF decoded, CTCSS decoded, PTT main, PTT secondary
- Control of an external relay (V7.05 on) or control external voice ID card

The design is copyright and may not be used commercially without prior approval from VK5DJ. For amateur use there is no restriction on use. The author will accept no responsibility for any problems arising from use of this freely provided project.
The code may be changed from time to time to correct problems or add new features.
Updates will be placed on: http://www.vk5dj.com

The Controller
PCB board layouts including Protel and Gerber files, parts list etc is provided in the main zip file. Construction details are provided later in this document.

Connection to the real world is through a 12 pin plug mounted on the board.

Pin	Description
1	+12V in (approx 10mA)
2	DC common ground
3	Mute input from repeater receiver
4	Mute input from gateway or link transceiver
5	Repeater audio in (receiver approx 100mV)
6	Gateway/link audio in (receiver approx 100mV)
7	Discriminator of repeater receiver (for CTCSS)
8	Gateway/link audio out of board (microphone)
9	Repeater audio out of board (microphone)
10	Repeater PTT
11	Simplex PTT
12	External CTCSS detect input to board

Detailed description of main interconnector JP4

Pin 1 is the unregulated supply voltage to the board, nominally +12 volts but should lie between the limits of 8-14 volts for reliable operation.

Pin 2 is the common ground for power supply, logical inputs and audio lines.

Pin 3 the repeater receiver mute input is generally programmed for 0V when a signal is received (ie active) and +5 to 15V when the receiver quiet. The line is clamped on the board with a 5.6V zener. Note that the line is held to either +supply line or 0V via a 22k resistor mounted on the board to ensure that if a receiver is not connected the input line has a clear logic level. When constructing the board the <u>pull up resistor OR the pull down resistor</u> (R21) is installed depending on the mute state available from the receiver in use and the appropriate programming of the PIC.
If active high from mute then install a pull down resistor (reprogram PIC for active high). If active low from mute then install a pull up resistor.

Pin 4 the gateway (or link) transceiver mute input. See notes for pin 3 but R22.

Pin 5 is the repeater receiver audio in. A capacitor is included on the input in case this line has superimposed DC. Audio levels should be at least 100mV. Pot RV2 controls the level. The top of the volume control in the receiver is an ideal source although some transceivers have an output on a connector for this purpose. It doesn't matter if the audio is not muted as the board provides this function.

Pin 6 is the gateway (or link) receiver audio in. See notes for Pin 5 but use pot RV1 to control level.

Pin 7 is audio from the discriminator of the repeater receiver (or the receiver being used for CTCSS input). The audio must be taken at this early point to obtain sufficient CTCSS tone as these low frequencies are both low in deviation and strongly attenuated in the remainder of the receiver chain. Commercial CTCSS decoders use the discriminator or access a low pass filter (eg the Simoco PRF15-20 units). On this board I require separate audio in from the receivers and this comes in on pins 5 and 6. If CTCSS is required for BOTH ports then an external circuit will be needed to combine the audio from the two discriminators before connection to this point.

Pin 8 is the audio out of the board to the gateway or link microphone. The level is adjustable and depends on the settings of RV2 (received audio) and RV5 (tone outputs for beeps and call sign)
In standard repeater/gateway/link use both pins 8 and 9 are jumpered together on board (JP6). In the case of an on air cross-linking gateway (eg a uhf gateway into a remote 2m repeater via a simplex connection) JP6 is unjumpered.

Pin 9 is the audio out of the board to the main repeater microphone. Level depends on RV2 and RV5 when the repeater receiver is in use and RV1 and RV5 when the gateway/link is in use. See Pin 8 notes re JP6 use.

Pin 10 is the repeater PTT. If set for 0V as active the transistor could sink 20-50mA but a relay should not be switched as there is no protection from transients. The pin sits at +supply volts. Alternately this pin may be used for active high. See later for programming details. In active high mode the feed is effectively 2K2 in series with a LED to +12V and therefore has limited drive capabilities. It should be fine to drive a transistorised PTT. In this latter mode the PTT LEDs are <u>unlit</u> when PTT active.

Pin 11 is the simplex PTT. See notes for pin 10.

Pin 12 is the output of an external CTCSS board. The pin expects an active low. The inactive state of +5 to15V is zener protected on board. Jumper JP3 provides for either external CTCSS or internal CTCSS. If the internal CTCSS is not required do not install parts for IC3 (NE567 tone decoder) and IC6 (CA3140 op amp low pass filter).

INPUT
Mute (COS) inputs are available at pins 3 and 4 of the main connector. Repeater input overrides the gateway input. On mute active (received signal) the timer begins. On closure of the mutes the timer is reset. Note the speed at which the repeater COS causes the logic to perform is set in memory. (See remote control function "0"). The logic will ignore any COS opening shorter than the value set by function "0". Quiet locations will use a value of 0 while noisy locations may need a value as high as $FF (255msecs).
On timeout the board sends 'TO' and shuts down the PTT and audio of both transmitters. When both mutes are high the transmitter provides a tail and sends another 'TO'. The device is again ready for use.

The PIC Microcontroller
The following description assumes that mute outputs (COS) go low for active. This is for the sake of description only as the direction is programmable if the resistors R21, R22 are arranged appropriately and the EEDATA byte is changed at address 77 decimal. Depending on active low or active high COS one end of each resistor goes either to +5V or 0V. See the circuit diagram and the two sets of holes on the board.

The mutes from the primary and secondary receivers are applied to Ports B.4 and B.5 of the microcontroller. The voltage is clamped to no more than 5.1 volts by the zeners to protect the input of the 16F1827 from excessive voltage.

Assuming negative going mutes, a zero on a mute indicates that a signal is being received. The mutes are continually polled by the microcontroller at approximately 20msec intervals. Once a mute is open on Port B4 or B.5, the appropriate PTT pin is raised on Port B.3 or B.2 and at the same time the appropriate audio is switched on through Port B.7 or B.6. The timeout timer starts counting.

The duplex repeater always has priority.

For what I have termed 'Standard identification operation' when a mute goes high (receiver mute closes) the 16F1827 checks to see if a callsign is due, if so it sends a callsign, if not it sends a tail and if appropriate a beep (or two or three – see later). The timeout timer resets to zero. The callsign timer resets if a callsign was sent otherwise it keeps counting. The delays for the callsign and the timeout timer may be remotely changed. Other identification options are described later.

If a mute stays on for a longer period than allowed by the timeout value the audio is shut down, a 'TO' is sent in Morse code (Port A.4 goes high to turn on the audio switch), then the PTT is taken high (Port B.3 and Port B.2) and everything shuts down. When the mute finally goes high this frees the system and a further 'TO' is sent in Morse and the repeater/gateway is again available.

The DTMF controller system works in the following way.

A 145436-decoder chip has an oscillator running at 3.58MHz. This frequency is coupled to the PIC with the 16F628A version (but not with the PIC16F1827) and so provides clock signals to both the decoder and the chip. DTMF tones are applied to the input pin of the chip through JP2. The jumper on JP2 determines which receiver passes on the DTMF tones. On receiving a valid tone (the "*" character) the output pins are sensed by the PIC through input ports RA0,1,2,3. A three-digit address code is next sent by DTMF and required to access remote functions. For the sake of this document the address sequence will be *987. The address is followed by the control byte and possibly some data. Note that a "*" must be sent to initiate remote controls. Once a sequence is begun an incorrect password or a delay of greater than 5 secs between digits will cause the controller to drop out of remote control mode. A good control initiates an "OK" when the received signal disappears.

The password is set in addresses 74,75,76 Decimal (incorrect in earlier manuals) of the EEDATA at program time and cannot be changed by remote control.

FUNCTIONS: Note these are different from ALL versions prior to Version 6.00

In this description it is assumed the primary port is a duplexed repeater while people will set either the link or the gateway to manage the secondary port. The main difference between a 'link' and a 'gateway' is that the 'gateway' has a tail and callsign added while a link does not.

In the following table the values for Functions 01-03 are cumulative. For example in Function 2 to relay DTMF tones on main TX, inhibit 'roger beep' and have DTMF tones come in from external RX you need a command byte of 01+04+08=0D so you send *+password+02+0D. Where a total for a command exceeds 09 remember these are Hex digits so Hex 10 (which is decimal 16) + 8 +1 is Decimal 25 or Hex 19. The command must use the Hex number 19.

Most control sequences consist of * + password + 2 digit address + 2 digit data **except:**
* The callsign (0C) and message functions (0D) commands consist of
  * + password + 2 digit address + multiple 2 digit data + FF.

All remote control information is in Hex notation.

Example1: If the password is 987 then a control sequence of DTMF '*9870003' will inhibit the main timer. When the carrier is dropped an 'OK' is sent in Morse if the command was successful. The tail will now have an extra two beeps.
A DTMF sequence of *9870*00 will reset all in function 00 including the timer.

Example2: If the password is 987 (used in all examples here), then a control sequence of DTMF '*9870611' will set a callsign timer at 17 minutes (11 Hex = 17 Dec). When the carrier is dropped an 'OK' is sent in Morse if the command was successful.

Remote controls:

Function Hex	Value Hex	Description	Default
00	00	enable both TXs and timeout timer	00
	01	inhibit main TX and link TX	00
	02	inhibit link TX	00
	03	inhibit timeout	00
	04	pulse the secondary port for ½ sec	00
01	00	Second port behaves as Link	00
	01	Second port behaves as Gateway	00
	02	3sec delay callsign mode	00
	04	Swedish callsign mode	00
	06	Italian callsign mode	00
	08	Use 1750 beep	00
	10	Use extended tail	00
	20	Enable secondary port 0=off, 20=on	00
	40	No callsign	00
02	00	no DTMF relay, no beeps	00
	01	DTMF relayed on main TX	00
	02	DTMF relayed on link TX	00
	04	No Roger beeps on tails if set	00
	08	DTMF provided by external receiver – no mute	00
	10	DTMF in through secondary receiver – with mute	00
03	00	No CTCSS required for full operation	00
	01	Require CTCSS for gate/link to TX	00
	02	Require CTCSS on rptr rcvr for any to TX	00
	04	Require CTCSS on rptr rx to operate gateway TX	00
	08	Extended timeout to 1 hour with CTCSS	00
	10	CTCSS extended operation mode	00
	20	CTCSS or COS mute operation	00
04	00-4B	CTCSS delay in minutes after firing	0A
05	00-FF	Mute delay in ms	00
06	00-4B	Callsign delay in mins	0A
07	00-4B	Timeout in mins	03
08	00-96	Primary tail in 1/10 secs	0A
09	00-4B	Message delay in mins	1D
0A	00-13	Postbeep and mode timer (secs)	0F
0B	00-19	Morse speed see comments	04
0C	string + FF	Change callsign	
0D	string + FF	Change msg content	
0E	00	0E=0* to clear permissions set by function 0	
10	05-7F	Freq for morse – values from 05 to 7F	$76
11	00-FF	Length of tail on simplex port in 10mS increments	$00
12	00-01	External relay 00=OFF, 01=ON	$00
13	00-01	Establish voice ID mode (see appendix 7)	$00
14	00-01	no ACfail = 0, ACfail =1 long beep active	$00

NOTE: Functions 01-03 REQUIRE one entry of ALL needed options. For example in Function 01 you send 61 if you want Gateway mode On and no callsign (01+20+40).

The DTMF tone pads do not create the normal accepted notation and have to be translated in the PIC. The following table shows the conversion process and will be needed when the higher Hex counts are used. Note the star key sends a character decimal 14 or 'E' in Hex, while the cross hatch key sends a character decimal 15 or 'F' in Hex.

Some key pads may not be labelled in the letters A,B,C,D but usually these occupy the fourth column. My Yaesu FT50 has quite different labels than what is expected but the fourth column works in the way described with 'A' in the top right hand corner, then B,C,D below it in sequence.

LABEL ON DTMF PAD	NUMBER SENT	NUMBER received by software	
0	Binary 1010	Hex 0	Decimal 0
1	Binary 0001	Hex 1	Decimal 1
2	Binary 0010	Hex 2	Decimal 2
3	Binary 0011	Hex 3	Decimal 3
4	Binary 0100	Hex 4	Decimal 4
5	Binary 0101	Hex 5	Decimal 5
6	Binary 0110	Hex 6	Decimal 6
7	Binary 0111	Hex 7	Decimal 7
8	Binary 1000	Hex 8	Decimal 8
9	Binary 1001	Hex 9	Decimal 9
A	Binary 1101	Hex A	Decimal 10
B	Binary 1110	Hex B	Decimal 11
C	Binary 1111	Hex C	Decimal 12
D	Binary 0000	Hex D	Decimal 13
*	Binary 1011	Hex E	Decimal 14
#	Binary 1100	Hex F	Decimal 15

Fig 1

For complete control, you need a 16 digit DTMF keyboard.

Function 00 (on loss of power this is the only function that auto resets)
Restore both TXs and timeout   *+password+00+00

Inhibit the main and link transmitter *+password+00+01
The main TX **and** the link will be inhibited. To return the TX to normal operation send a * + password + 00+00.

Inhibit the link/gateway   *+password+00+02
The link or gateway only will be inhibited (the 'Roger beep' will be inhibited during this mode). To return the TX to normal operation send a * + password + 00 +00.

Cancel the main timer   *+password+00+03
Cancels the timeout timer – this enables the system to be used for extended club broadcasts. To return the TX to normal operation send a * + password + 00 + 00.

Pulse the secondary port   *+password+00+04
Pulses the secondary port for ½ second to reset a device on this port. Only practical if the secondary port is unused for linking or gateway and locked off in command 01.

Function 01 Secondary port and identification behaviours
Key in * + password + 01 + two digits

Function	Command	Description
01	00	Second port behaves as Link
01	01	Second port behaves as a Gateway
01	02	Three second delay callsign mode
01	04	Swedish callsign mode
01	06	Italian callsign mode
01	08	Use 1750 beep to trigger TX
01	10	Use extended tail
01	20	Enable secondary port
01	40	No callsign when set

Add the numbers to achieve the desired results (e.g. Hex 8 + Hex 4 = Hex 0C)
Examples:
- *+ password + 01 + 05 secondary port is a gateway (disabled), using the Swedish callsign mode, standard tail.
- *+ password + 01 + 18 secondary port is a link (disabled), extended tail, use 1750 beep.
- *+ password + 01 + 38 secondary port is a link (enabled), extended tail mode, use 1750 beep.
- *+ password + 01 + 26 secondary port is a link (enabled), Swedish callsign mode, 3sec delay callsign mode.
- *+ password + 01 + 68 secondary port is a link (enabled), no callsign, use 1750 beep, standard tail.

Secondary port behaviour
Port as a link: *+password +01+see table
Port as a gateway   *+password +01+see table

A gateway has callsign inserted and a tail, a link does not.

Callsign modes
Standard callsign mode: default mode if others not set
> The callsign is sent at the end of a transmission if the callsign timeout period has completed. If a callsign was not sent at the end of a QSO the callsign will be played once at the conclusion of the timeout and then the repeater will fall silent until the next mute opening.

3 Second Delay Callsign Mode:
> If the repeater is unused for period set in Function 0A (suggest 15 seconds) then a mute opening exceeding 3 seconds will initiate a callsign. Otherwise callsigns will appear on the end of minimum 3 second mute openings if the callsign timer has expired.

Swedish European Callsign Mode:

This mode idents at the start of transmissions if the call timer is expired, and during operation of the repeater whenever the timer expires. If extended tail timer is activated (Function 01 10) it idents when the extended tail timer shuts. The repeater then remains silent.

Italian European Callsign Mode:

This mode idents whenever the callsign timer expires but not if a signal is being received. If timer expires during use, it idents when the mute closes.

Operational mode

Beep operate:

Some repeater groups use a 1750 tone "beep" to open the repeater. Adding 08 Hex to the data requires a beep to open the repeater from sleep. The repeater then remains open until the extended tail timer expires. The code expects that the NE567 has been retuned to 1750Hz and the low pass filter retuned or omitted (or an external tone decoder accessed through pin 12 of JP5 and jumper JP3 set appropriately).

For this mode to work it is essential that the following are set:
- Function 01 with minimum value of 18 Hex (at least Beep operate and extended tail)
- Function 03 with 00 for no CTCSS conditions set.
- Function 05 with minimum value of 64 Hex (mute delay)
- Function 0A, extended tail timer, with minimum value of 05 to allow tail ending.
- Note: there is a 30 sec delay before function operates after setting up.

For example a code of *9870134 (4+8+16=28 or 1C in Hex) would set secondary port to a link, Swedish style identification, beep access, and use extended tail. *9870300 sets no requirement for CTCSS. *9870564 sets mute delay of 64msec. *9870A05 sets the extended tail to 5 secs.

Once opened by a 1750Hz beep the repeater is available until the repeater TX shuts down (period set by Function 0A expires). Tail ending by a station with no 1750 beep is acceptable until the extended tail timer expires. If there is a gap in transmission by the repeater transmitter a beep must be resent.

Use extended tail:

Activates the extended tail set in Function 0A. This tail occurs after the 'tail beep' and is used to allow the repeater to be re-keyed without hearing a noise burst as the repeater TX shuts down.
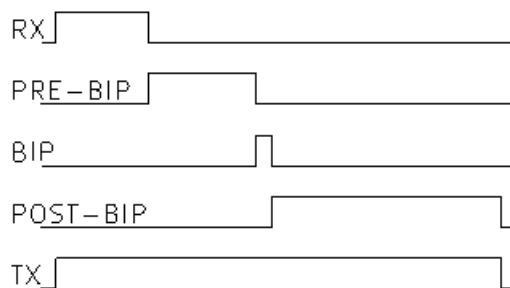
This diagram shows the relationship of the tail options. Pre-bip is the normal tail, while Post-bip is the extended tail. The timeout timer is reset when the RX COS drops out. The extended tail timer has multiple uses in the controller.

Initial state of the link:

        This function was requested by users as many people use the controller for a stand alone repeater with no link or gateway. As of Version 6.10 the link is normally off on first power up unless altered by this function.

        Including a 20h (remember these are Hex numbers) in the total for function '01' will cause the controller to boot up with the link/gateway available. The absence of 20h makes the link unavailable. Once the secondary port is enabled Function 00 may be used to regulate operation of the secondary port.

No Callsign:

In some countries no callsign is required. Setting a value of 40h will inhibit the callsign (or more correctly adding 40h to the other choices in this group).

Function 02

Relay DTMF codes and beep function   *+password+02+ two digit data

If your repeater needs to forward on DTMF tones you <u>may</u> need to activate this function if you want to send a *. Normally, the controller does not pass on legitimate command sequences so this function is needed if you wish to relay a sequence to control a remote system through a link.

For example:

*9870200 will inhibit DTMF tones beginning with a * from being relayed.

*9870201 will pass all tones through Repeater TX but not the simplex port TX

*9870202 will pass all tones through simplex TX but not the repeater port TX

*9870204 will inhibit 'roger' beeps on the tail.

*9870208 will enable DTMF to be supplied by external receiver with no COS

*9870210 will enable DTMF to be supplied by second receiver with COS

If the audio outputs are jumpered together on the board then clearly 01, 02, 03 all pass DTMF audio to both TXs.

Data Address+02+data	DTMF relayed (Gate/link)	DTMF relayed (Repeater)	Beeps (on tail)
00	No	No	Yes
01	No	Yes	Yes
02	Yes	No	Yes
03	Yes	Yes	Yes
04	No	No	No
05	No	Yes	No
06	Yes	No	No
07	Yes	Yes	No

Add 08h to above values to input DTMF audio from external receiver with no COS.

Add 10h to above values to enable DTMF through second port.

To use this command firstly put the JP2 jumper in the secondary position. Next send *9870210 with the **primary** receiver input COS active. Once function 0210 is set, the DTMF will respond to the COS of the secondary receiver. Once setup, any future "OK" is sent through the port that is being used for control.

Note if function 0210 is **not** set then "OK" is sent only to the primary TX.

If you turn off function 210 by sending *9870200 through the secondary port then you will no longer have full control through the secondary port. This can be regained by opening the primary receiver COS when sending DTMF *9870210 through the secondary receiver. You will then again have control through the secondary receiver.

Function 03
Require CTCSS to transmit   *+password+03+two digit data
To **fully** access these functions you will need a system of switching CTCSS tones in from both receivers. **The on-board jumper caters for CTCSS from <u>one</u> receiver**. I have provided the software functions, but you must provide your own external means of mixing the CTCSS signals and switching them via the appropriate mutes. Otherwise you must make a choice as to which receiver has the CTCSS function.

The CTCSS system also allows repeater operators to use a CTCSS tone to prevent timeout. This is achieved by sending a parameter of 8.

The selected choices will depend on local issues such as licensing or interference problems. In my network of repeaters I have CTCSS from the repeater receiver and therefore commands 00,02,04,06 will be used. CTCSS from the gateway/link receiver would access commands 00 and 01 if the CTCSS was available.

CTCSS **only** on the **gateway** receiver will allow you to use commands 00, 01, 08. CTCSS **only** on the **repeater** receiver will allow use of commands 00, 02, 04, 06, 08 (although 04 and 06 achieve the same outcome).
I have assumed in data 02, that CTCSS required on the repeater would of necessity require it on the gateway, otherwise people on the repeater system would hear only one half of the conversation and think the system is not in use.

Data Address+5+data	Mute open on	Repeater TX requires	Gateway TX requires
00	Rptr Rx	No CTCSS	No CTCSS
00	Gate RX	No CTCSS	Not applicable
01	Rptr Rx	No CTCSS	No CTCSS
01	Gate RX	CTCSS	Not applicable
02	Rptr Rx	CTCSS	CTCSS
02	Gate RX	No CTCSS	Not applicable
03	Rptr Rx	CTCSS	CTCSS
03	Gate RX	CTCSS	Not applicable
04	Rptr Rx	NO CTCSS	CTCSS
04	Gate RX	NO CTCSS	Not applicable
05	Rptr Rx	NO CTCSS	CTCSS
05	Gate RX	CTCSS	Not applicable
06	Rptr Rx	CTCSS	CTCSS
06	Gate RX	NO CTCSS	Not applicable
07	Rptr Rx	CTCSS	CTCSS
07	Gate RX	CTCSS	Not applicable

Use 32 (Hex 20) as an alternative to 02, 03, 06, or 07. **Enables TX by CTCSS or COS input from primary receiver.**

The next two commands should exist alone and not mixed with other parameters in this group.
   a) **Add 8 to any of the above to extend the timeout timer to 60 mins (convert total to Hex digits eg 08+04=0C).**

b) A value of hex 10 enables CTCSS extended operation. This value cannot be mixed with the other parameters.

Examples

(1) For example *9870301 requires CTCSS on the gateway receiver to access the repeater system, no CTCSS is required in the reverse direction, normal timeout works.

(2) For example *987030C requires CTCSS on the repeater receiver to access the gateway but not the repeater, and timeout is inhibited for special purposes.

(3) For example *9870310 enables one person with CTCSS to enable the system for a user without CTCSS. The system is enabled according to command 04. Note: do not use this command in association with others

Extended timeout timer in eedata: *+password+03+08

This is an alternative method of preventing the repeater from timing out for long broadcasts. If function 03 08 is set, a three second CTCSS tone will prevent timeout for 60 minutes after the tone disappears. Normal timeouts return 60 minutes later. Function 03 08 may be left operational until needed as it is only activated when the three second CTCSS tone is received. The CTCSS must be received for at least 3 seconds to activate this mode. When the extended timer is successfully set, two beeps are heard on the repeater audio. Continuous CTCSS transmissions are NOT allowed as the controller will loop within the extended timer routine.

CTCSS extended option: *+password+03+10

Used in conjunction with CTCSS access to ports, this function if activated, causes the repeater to be available for the length of time set in Function 04. Timing is constantly reset on receipt of CTCSS. Useful where non CTCSS enabled stations are talking to those that are so equipped. See function 04 for time setting.

CTCSS or COS operation of transmit: *+password+03+20

If set by adding 32 (Hex20) to the data sent, this function will cause EITHER CTCSS received or COS operation to activate the transmitter functions. The CTCSS acts as a pseudo COS. At this time only the primary receiver works with this option.
Do NOT use with values of 02, 03, 06, or 07.

Function 04:   *+password+04+two digit data

CTCSS delay in minutes, maximum of 75 minutes (4B Hex).
This is used in conjunction with CTCSS extended function see above.
Example: *987041E would set the timer for 30 minutes.

Function 05   *+password+05+two digit data

The Hex information number sent is the length of time the COS must be open before the logic accepts there is a legitimate signal. The values may be within $00 and $FF. The default condition is $00 (i.e. no delay). The maximum of $FF provides a delay of about 500msecs. Use this function in conjunction with Function 01 to achieve beep access.

A typical command might be : *9870532 where the $32 equates to 100 decimal msecs.

Function 06

Changing the callsign interval   *+password+06+ two digit data

The maximum period is 75 minutes (Hex 4B).
Each unit is 1 minute, so if you want 10 minutes:
Decimal 10 = Hex 0A. You will need to send a 0 then an A.
Your control sequence is *987060A
Another example, this time for 15 minutes:
Decimal 15=Hex F
Control sequence is *987060F

If you want a short delay and your calculation suggests a single digit, you MUST preface it with a 0. eg the control sequence might be *9870601

Function 07:

Changing the timeout interval   *+password+07+ two digit data
The same rules and strategy as 'changing the callsign interval', except the command is 07. Units are minutes. The maximum is 75 minutes =Hex 4B
Eg a 3 minute timeout is Hex 03.
Control sequence is *9870703
If you want a short delay and your calculation suggests a single digit, you MUST preface it with a 0. eg the control sequence might be *9870702

Function 08:

Setting the tail length   *+password+08+ two digit data
The full command is "* +password+08+data". Length of tail is determined by the Hex data number multiplied by 100msecs after the COS drops. This length is added to by beeps and callsigns.

To achieve a zero tail set the data to $00, then turn off the beep function (See control #). The acceptable values for data are between $00 and $4B (0-7.5 secs). These are in increments of 0.1 secs. I recommend a value of $0A for 1 sec delay. A long tail of $4B provides 7.5 seconds. The timeout resets when the COS drops so users may re-key the repeater during the tail if your operating protocols permit.

Function 09:

Period to repeat message in morse   *+password+09+ two digit data
A message may be sent at regular intervals eg "Club meeting cancelled tonight". A $00 in the timing location disables this function, otherwise a value from $01 to $4B will set repetitions from 1 minute to 75 minutes. A setting of Hex 3C (60 in decimal) will send the message every hour. The message is sent only in quiet times for the repeater. Timing is reset if the mute opens. A callsign is sent after the message.

Function 0A:

Extended tail timer: *+password+0A+ two digit data
This timer affects the period after the tail beep as an extended tail during which stations may call before the TX resets. It is used in conjunction with "Use extended tail" see function 01. Length of extended tail is determined by the Hex data number multiplied by 100msecs after the tail beep.
To achieve a zero extended tail either set this to 00 or turn off the extended tail flag in 01. Length of extended tail is determined by the Hex data number in seconds and is measured after the tail beep. To achieve a zero extended tail either set this to 00 or turn off the extended tail flag in 01.

Function 0B:

Changing the morse code speed   *+password+0B+ two digit data
The Hex information number sent is the length of a "dit" in 10msec increments. The default condition is $04 (i.e. 40msecs) and coincides with about 25wpm. Increasing this value slows the morse. A maximum value of $19 is allowed. In any case above $10 your listeners will be driven crazy by the slow morse. I suggest a range of $03-$07.
*9870B07 would set the speed at about 14wpm.

Function 0C:

Changing the callsign.   *+password+0C+ multiple 2 digit data
Normally the callsign will be set during the programming process by altering the data in EEDATA before burning. Alternatively it can be done on air. Firstly, as an interim measure set the callsign delay as 01. This will generate callsigns every minute to assist with checking. Return the delay to $0A (or whatever you choose) after successful programming of the callsign.

The PIC program works by reading the ASCII value of each letter of the callsign then reading a corresponding look up table item for the morse code construction.

Here is the ASCII table in **Hex** with its corresponding numeral or character:

ASCII	char	ASCII	char	ASCII	char	ASCII	char
2F	/	39	9	4A	J	54	T
30	0	41	A	4B	K	55	U
31	1	42	B	4C	L	56	V
32	2	43	C	4D	M	57	W
33	3	44	D	4E	N	58	X
34	4	45	E	4F	O	59	Y
35	5	46	F	50	P	5A	Z
36	6	47	G	51	Q		
37	7	48	H	52	R	00	space
38	8	49	I	53	S		

Here is an example of how you would use Function 0C by radio DTMF

To put VK5DJ into memory you would send from the DTMF pad:
*987 (the password) 0C (the command) 56 4B 35 44 4A FF (the call followed by FF)
I've shown spaces to make it easier to read but the numbers would be sent:
*9870C564B35444AFF

The callsign must finish with an FF. This is used by the software to determine EOM.
A space (00) can be inserted in a callsign or used to send a blank callsign. If using
Command 02 08 (DTMF from external rcvr) the code MUST receive a FF to abandon
the remote control loop as it has no other way of knowing the sequence is finished.

Function 0D:

Changing the morse message   *+password+0D+ multiple 2 digit data
The morse message shares the same memory area as the callsign. The software places
the callsign at memory bottom, its concluding FF is then used to find the start of the
message memory space. There is room for 68 characters for the two functions. If the
callsign takes 10 characters (including the FF) then there are 58 characters left for the
message. A 68 character callsign would leave no space for a message.
To change the message, send *+password+0D+message+FF
For example: *987 0D 48 45 4C 4C 4F FF will put HELLO in the message area.

A 00 puts in a word space, so to place "NO MEETING" in the message memory you
would send: *987 0D 4E 4F 00 4D 45 45 54 49 4E 47 FF

The spaces between the Hex characters are to help you read the pattern, in practice the
above would be sent: *9870D4E4F004D454554494E47FF

If you run out of memory the program will detect this and prevent you from
overwriting other memory. An emergency stop $FF exists in memory to avoid
overrun problems on playback. If you have to change the callsign and it is shorter or
longer than the original one, you will need to re-enter or change the message too. If
using Command 02 08 (DTMF from external rcvr) the code MUST receive a FF to
abandon the remote control loop as it has no other way of knowing the sequence is
finished.

Function 0E:

Reset TXs and timeout   *+password+0*+00

Send *+password + 0*+ 00 to reset defaults in the non- memorised functions.
Function 0 controls are reset to default - timer active, gateway mode on or off (subject to secondary port being activated by Function 01), main TX on.

Function 10:

Set frequency of the sound made by morse ident: *+password+10+two digit data
The default is 76h and most of the usable frequencies are around this mark. I have not allowed a value less than 5. This is a very low frequency. A maximum of 7F Hex is allowed, beyond this white noise is created.

Function 11:   *+password+11+two digit data

Set length of the tail on the simplex port in 10msec intervals. This is defaulted to zero but there may be circumstances when you need a short tail. For example to stop mute flutter on the repeater input translating as flutter on the link. Allowed range is 00-FF for a maximum of 2.55secs.

Function 12:   *+password+12+two digit data

Operate an external relay. Two digit data of 00 switches off the external relay, 01 turns it on. See Appendix 5 for hardware change required.

Function 13:   *+password+13+two digit data

Operates an external voice player. See Appendix 7 for wiring information.
00 = standard morse code ident
01 = voice ident using an ISD1820 voice sound recorder ex Ebay

Function 14:   *+password+14+two digit data

Activates ACfail. See Appendix 6 for wiring information.
00 = no ACfail used
01 = if ACfail wired to pin4 then low volts (power fail) activates a long beep

Further information

Tail operation

A tail is provided once both mutes are inactive. This is user settable and may be changed by remote control (see above). Audio is fed through during callsigns and tail. Transmissions arising from the link have a 'boop' tone while those from the repeater receiver have a 'beep' on the tail.
A 'Roger beep' is normally provided at the end of a tail.

- No beep on tail indicates Function 00 has gateway/link off

- 1 beep on tail means gateway/link on
- 2 beeps on tail indicate timer is inhibited and link/gateway off.
- 3 beeps on tail indicate gateway/link on and timer inhibited.

All beeps may be turned off using function 02 with a value of 04.

A tail of length set by function "08" is always generated on the repeater TX.
When programmed as a link, the simplex TX normally has a tail only if a timeout occurs. At this time a tail is provided with 'TO' sent in morse code.
It is possible to add a short tail to the link function secondary port to reduce weak signal flutter on the link port. This is done by adding (at program time only) a value at location 5A in the EEDATA. A hex value is programmed in to create a delay with multiples of ten millisecs. For example, a value of 10 Hex = 16 Dec = 10 * 16 mSec = 160 msec. So the maximum value of FF Hex equates to 255*10 = 2.55 secs.

Programmed as a gateway, tails and callsigns are provided on both TXs (providing link is not programmed off or CTCSS is not missing if required). Whether the controller supports a link or a gateway is set by the data in EEDATA memory location decimal 64 where 00=link and 01=gateway behaviours.

Call signs – ident protocol depends on the mode set in Function 01
Call signs are sent at periods determined by original programming of address 69 decimal in the EEDATA in the PIC or changed by remote control at the end of the over. In standard ident mode if a call is not due at the end of the last transmission of a contact one call sign will be sent at the end of the timing period. The repeater will then go silent until the next mute opening. A call will be generated at the end of this mute opening if the call delay period has expired since the last.

Message function –only when repeater is quiet
It is possible to send a message at a set interval providing that the mute does not open. Any opening of the mute automatically enforces a new count for the sending of the message.

The purpose of this facility is to provide a simple message facilty eg. "Club meeting cancelled tonight". The message can be sent at any interval from 1 minute to 75 minutes of repeater silence. A callsign will be sent at the end of the message providing that the callsign delay time is less than the message delay.

The length of the callsign and that of the message may be up to 62 characters. They share the same memory space. The message may be 62 minus (length of callsign) long. So if the callsign was VK5DJ followed by the FF there are 6 characters used. This leaves 56 characters for the message including the $FF.

CTCSS operation
The values on the circuit suit reception of a 118.8 Hz CTCSS signal. These low frequency signals are attenuated by tailoring of the receiver audio, so the board must take its audio from the discriminator circuit or early in the audio chain before tailoring.

Changing C7 or R8 changes the CTCSS frequency beyond the range achievable with control RV6. Lower values increase the frequency. LED L4 lights when a correct tone is received. L4 is a useful tune indicator when adjusting RV6.

Normal operation i.e. repeater and link TXs active, does not require a CTCSS signal. Through remote control functions 4 and 5 the link and repeater functions respectively will require CTCSS for access. These functions may be necessary to avoid interference or meet a licensing requirement to 'lock out' people without the required endorsements.

A requirement for CTCSS on the main repeater may be any combination (see command 03) or on the link may be mute only or mute+CTCSS. The CTCSS will need to come from only one receiver (unless there is an external mixer and some form of CAS control.) Generally CTCSS will not be a feature of a link. See function 03 in remote controls, but CTCSS may be required for gateway operation.

LEDs
See Jumper section for orientation of the board for this description.

The CTCSS detect LED is in the North East of the board and lights when there is a valid CTCSS signal received.

The repeater and simplex PTT LEDs are together in the South East corner. The repeater LED is the northernmost of the pair. The simplex PTT LED is the southernmost of the two. They light when PTT is low (i.e. active).

The DTMF valid LED is at the bottom (South West) corner of the board by itself and lights when a valid tone pair is received.
The power on LED is in the North West corner of the board.

Constructors may choose to mount the LEDs on the front panel of the box.

JP1 Test Function
Pin 1 is the output for DTMF decoder (data bit 0) – high if bin 0001 rcvd (pin 1 is nearest the 7805 voltage regulator)
Pin 2 is the output for DTMF decoder (data bit 1) – high if bin 0010 rcvd
Pin 3 is the output for DTMF decoder (data bit 2) – high if bin 0100 rcvd
Pin 4 is the output for DTMF decoder (data bit 3) – high if bin 1000 rcvd
Pin 5 is data valid and goes high for a valid tone detect (note mod to add link)
Pin 6 is the output of the Xtal oscillator and should be 3.579MHz

Jumpers
With the board held with components up and the main connector on the left (west) and the voltage regulator at the bottom (south), the trim pot for the CTCSS is to the east. From an imaginary point in the centre of the board the angles locate jumpers.

H means the jumper lies east/west, **V** means the jumper lies north/south.
Position number **1** is always either the northernmost or the easternmost position.
Position number **2** is the next to the south or west and so on.

JP1 are test points see above (190 degrees) with pin 1 nearest the 7805

JP2 controls the source of the audio for the DTMF tones. (300 degrees)
- H, 1 = repeater receiver audio
- H, 2 = simplex audio

JP3 internal or external CTCSS input to 16F1827 (75 degrees)
- V, 1 = external CTCSS decoder
- V, 2 = internal CTCSS decoder

JP4 audio source for CTCSS (320 degrees)
- H, 1 = repeater receiver audio
- H, 2 = simplex receiver audio
- H, 3 = discriminator audio from repeater receiver

JP5 main interconnector – see above (270 degrees)

JP6 join microphone points (270 degrees)
- V, 1 on = microphones are joined (gateway/link condition)
- V, 1 off = microphones separated - single repeater or range extender model.

Construction

Drill the mounting holes for your box, and trial mount the board.

Make the following decisions:
1. What polarities do the transceiver(s) require? See notes on page 28 regarding interface issues. The direction of the mutes and whether they are open collector or not affects the placement of R21, R22 to 0V or +V. The board provides for either connection. The EEDATA at address 78 will need to be modified. If the transmitters PTT require active low then everything is fine as is, otherwise the direction byte at decimal 84 in the PIC's EEDATA will need to be changed at program time.
2. Where do you want to mount the LEDs? If you want to put these on the front of the box then obviously you will install wires rather than the LEDs on the board.
3. Decide if you will be using CTCSS. If not you will not need the components associated with CA3140 and LM567. But you will need to include JP3, R4, and bridge L4. (You'll have one less LED so don't drill your front panel for it!!)

Mount the socket for the 16F1827 (strongly recommended, you might want an updated program one day). Note notch is to the regulator end.

Install the 7 jumpers (from top to bottom of board)

Install JP1-6

Install a wire jumper from pin 5 of JP1 (effectively pin 12 of the MC145436) and pin 15 of the 16F1827 on the boack of the board. Use a 1cm length of insulated wire. See LINK.JPG.
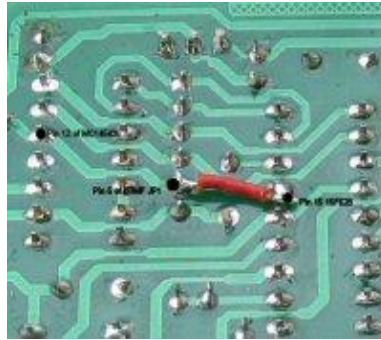
Install all the resistors but note positions of R21,R22. Solder to upper board and bottom trace where jumpers are required <u>if you are not using a plated through board</u>. I recommend mounting R21/R22 vertically with the resistor body against the board on the south side (see above for orientation). If you need to change COS polarity at a later date it is then easy to cut the resistor lead in its centre and bring a wire up from the alternate hole in the board.

Install all the capacitors, solder top and bottom where traces exist if you are not using a plated through board. **Note** that the circuit diagram shows capacitors C14,15,18,21 with negative to the outside world. Some radios have +ve voltage on their microphone inputs. This will result in distorted audio and possible failure of the capacitors. I strongly suggest these capacitors are wired with +ve to the outside world. The circuit is wrong as is the board overlay.

Install voltage regulator LM7805.

Either solder LEDs to board or attach wires for LEDs to be mounted on front panel at a later date. I suggest at least dangling the 'power on' LED temporarily on its wires for the next check.

Apply +12V to board and ensure that +5V is available in appropriate spots eg pin 4,14 of PIC.

Remove voltage and solder remaining solid-state devices – leave PIC out of socket for the moment.

Install in box and wire the board to the DB25 on the rear of the box. I used pins 1-12 to maintain the integrity of the numbering system.

Wire the LEDs located on the front panel and a power switch if you wish. I didn't use a switch, preferring the control logic to boot up as soon as the rigs have power to avoid unpredictable states in the rigs.

PROGRAMMING THE PIC
I use PICkit3 an excellent programmer from MPLAB that works on any PIC. I do recommend it.

Load the programmer memory with the file 'V703ctrl.hex' into your programmer of choice. Once loaded, your programmer should allow you to modify the callsign and the password within the data area although the callsign can be entered later. If you

want to add a short tail on the link this is also the time to enter a number at EEDATA
address 5A.

EEDATA memory map, all address numbers in decimal. Values must be in Hex.

```
EEDATA memory map, first number in decimal. Values must be in Hex.
0..61 ($00-$3D) callsign and message(Hex value of chars /,0..9,A..Z)
62    ($3E) emergency stop for callsign
63    ($3F) 0=link, 1=gateway, Callsign mode add 0, 2,4,8,16,32
             beep access add Hex 10 extend tail, inhibit link Hex 20
64    ($40) 0=inhib dtmf,1=OK rptr,2=OK simp,4=no beeps on tail
65    ($41) CTCSS address see manual
66    ($42) CTCSSdelay in minutes
67    ($43) Mutedelay in msec
68    ($44) Callsign delay in mins
69    ($45) Timeout delay in mins
70    ($46) Main tail prebeep in 1/10 secs
71    ($47) Interval for msgs in mins
72    ($48) 15 sec timer
73    ($49) Speed of CW see notes
74    ($4A) Password digit 1
75    ($4B) Password digit 2
76    ($4C) Password digit 3
77    ($4D) mute direction (0= both active high, 1=rpt low & simp
             high, 2=rpt high & simpl low, 3=both active low)
78    ($4E) ptt direction (0= both active high, 1=rpt low & simp
             high,2=rpt high & simpl low, 3=both active low)
79    ($4F) Call note value
80    ($50) Start of timeout message
83    ($53) Start of OK message
86    ($56) Beep tone
87    ($57) Boop tone
88    ($58) Start of CTCSSdelay anti jitter value default 2500
90    ($5A) Address for a short delay on link
91    ($5B) Start of morse translation table (begins with $A9)
```

If the defaults provided in my sample file are not what you want, now is a good time
to edit them. Remember that the "*" key sends Hex "E" and the "#" sends Hex "F"

<u>Default EEDATA settings</u>
Certain settings may be made only at PIC program time. Usually the defaults are fine.
The memory address numbers are given twice as different programmers label the
eedata differently.

Address in EEdata (1)	Address in EEdata (2)	Default value	Comments/options
4A Hex	214A Hex	9	*Password digit 1*
4B Hex	214B Hex	7	*Password digit 2*
4C Hex	214C Hex	3	*Password digit 3*
4D Hex	214D Hex	3	*mute direction* *0= both active high* *1=rpt low & simp high* *2=rpt high & simpl low* *3=both active low*
4E Hex	214E Hex	3	*ptt direction* *0= both active high* *1=rpt low & simp high*

			2=rpt high & simpl low *3=both active low*
4F Hex	214F Hex	76 Hex	*Call note value*
56 Hex	2156 Hex	76 Hex	*Beep tone*
57 Hex	2157 Hex	6C Hex	*Boop tone*
58 Hex	2158 Hex	C4 Hex	*This and the next Hex number provide the delay for CTCSS as an anti-jitter measure. 09C4 Hex equates to 2500 decimal.*
59 Hex	215A Hex	09 Hex	*Note that 2 byte Hex numbers are stored in reverse order.*
5A Hex	215B Hex	00 Hex	*Each hex unit equates to 10 mSec delay on link*

The callsign may easily be entered using the remote function, however if you really want to put it in the PIC memory at program time – I suggest you write out the numerals before hand. The CW message follows the FF at the end of the callsign. The message continues until location 63 (dec) or $3F (Hex)

Now put the required password numerals in locations 75,76,77. (Use 00 to 09,0A-0F). Remember a *=Hex 0E and #=Hex 0F. If you plan to relay DTMF tones for IRLP it may be wise to choose at least one digit of the three that is not base 10 eg A, B, C, D, * or # to avoid a false remote control.

Now put the appropriate number in the mute direction location at 78. The default is 03 that is for active low mutes. This must match the locations of resistors R21, R22.
(0=both active high, 1=rpt low & simp high, 2=rpt high & simpl low, 3=both active low)

Now put the appropriate number in the PTT direction location at 79. The default is 03 that is for active low PTT.
(0=both active high, 1=rpt low & simp high, 2=rpt high & simpl low, 3=both active low)

Now set the gateway/link selection byte at 64 decimal to 00 if you want the controller to support a link, to a 1 if you want the secondary port to be a gateway. If you are not using the secondary port use a zero. A zero will also set the callsign rules in 'standard mode'. See remote function '01' for other combinations. These values can be remotely changed once the controller is built. Other EEDATA memory values are probably OK as is.

Frequency of 'beep' for repeater receiver tail and 'boop' for link receiver tail are located at decimal 86 and 87 (see table above). If you don't like the frequencies I chose adjust these accordingly. The higher the number the higher the tone. I suggest adjustment by only one or two for starters. Maximum value is 127. Default 75 and 6C.

The CTCSS system has a 1 second hang delay built in (except when in mode 03 20). This delay is to stop jitter with noisy signals dropping the CTCSS. In some applications this may not be required. It is possible to change the antijitter value at memory locations Hex 58, 59. The default value is Hex 09C4 (2500 decimal), but because these are stored in reverse order they will appear in your programmer window as C409. Changing these two values to 00 00 will stop the CTCSS from hanging. Alternatively you may wish to put a smaller value such as Hex 0100 which

will be placed in memory as 00 at location $59 and 01 at $5A and will provide a delay of about 100msec.

If you require a short tail on the link you can add one by inserting a value at address $5A (00 default). Each unit delays 10 msec. e.g. a value of 20 Hex = 32 Decimal provides a 320 msec delay. This may be helpful in stopping chop on the link TX.

Before programming ensure that the programmer fuses are set as follows:

Config1 `FOSC_ECH,WDTE_ON,PWRTE_ON,MCLRE_OFF,CP_OFF,CPD_OFF,`
`        BOREN_ON,CLKOUTEN_OFF,IESO_ON,FCMEN_ON`
Config2 `WRT_ALL,PLLEN_OFF,STVREN_ON,BORV_19,LVP_OFF`

All this information is in the hex file and is usually automatically set by your programming software. There is a small chance that some software does not do this and the PIC fuses have to be hand activated. MPLAB reads the hex file automatically and gets the PIC fuses right for you, so I do recommend MPLAB.

Make sure the programmer is set for the 16F1827. Always recheck this before burning. Now program the chip.

TESTING
The following assumes active low situations. If not, you will no doubt work out your own testing procedure by inverting the logic. Pin numbers refer to either the DB25 or JP5 if you followed my advice on connections.
Temporarily connect the following:

- Switch R – SPST between pin 3 and earth pin 2
- Switch S – SPST between pin 4 and earth pin 2
- A SPDT switch and a headphone socket to listen to either pin 8 or pin 9 (earth is common)
- A SPDT switch and a socket to enable audio either from a receiver or an audio generator to direct audio to either pin 5 or pin 6.
- The capacity to measure the voltages at pins 10 and 11 (the PTT outputs)
- Connect jumpers – JP2, 1&2   JP4, 1&2   JP3, 1&2

On power up with default settings, the controller is set with link/gateway off.

Test logic functions:
- Headphone listening to repeater mike – pin 9
- RV5 half way
- Close switch R
- LED1 lights.
- Open switch R
- LED1 stays on for 1 sec (or possibly longer if it decides now is a good time for a callsign). Repeat opening and closing R to see if it matches your expectation.
- Close switch R and leave it on. After about 5 minutes you should hear 'TO' in Morse and LED1 should go out when the TO is finished.
- Open switch R and you should again hear 'TO' and LED1 lights as 'TO' is sent. The system will again respond to opening and closing of switch R.
- Wait a little longer and after a closure/opening of switch R a callsign should be generated. LED1 again indicates the PTT state for the repeater.

- Use a multimeter to see that pin 10 is going low when the LED1 lights. At other times it should be at about 12V.

Test DTMF capability:
Connect a DTMF generator (or a receiver) to the appropriate input (pin 5 with the test jumpers in place). Send a tone, LED 3 should light. If not adjust RV4.

With no input check the voltage on pins 1,2,3,4 of JP1. These should be at 0V. Send a DTMF C and check that all pins 1,2,3,4 are high.

If DTMF decode LED doesn't work with incoming tones, check level RV4 (nearest main socket or check with a CRO that there is a 3.58 MHz clock on pin 6 of JP1.

With switch R closed, send the address and a 1 (eg *9870120). Open switch R, hear an OK in the headphones, and then note that LED2 the link port now responds to switch R. This command turns on the secondary port.

Now close switch S and note that LED1 (primary repeater PTT) lights and on opening switch S hear a 'boop' on the tail.

Test CTCSS if installed
Feed a low level CTCSS tone into the appropriate pin (for the test setup use pin 5 and bridge JP4 bridge H1). I suggest 118.8hz. Adjust both RV6 (freq determination) and level RV3 for lighting of L4. Reduce level and retune RV6 for LED4 to find the most accurate setting of RV6. The level potentiometer may need to be reset when the receivers are finally connected. Note that I strongly recommend that you connect as close as possible to the receiver discriminator as receiver audio chains attenuate CTCSS tones. On 'switch on' the CTCSS decoder NE567 takes several minutes for its frequency to settle. Allow 'warm up time' before adjusting.

The 567 tend to drop out when there is a whistle or loud speech. The program allows for this and has a hang facility built in. As long as the LED flashes once every couple of seconds the system will work.

Audio switching:
With different audio provided to the two receiver output pins, switch R and switch S alternately note the different combinations of audio. Potentiometers RV1 (simplex TX audio) and RV2 (repeater TX audio) adjust levels.

FINAL TEST
Connect to your transmitters and receivers. All levels will need to be reset. Connect jumpers as appropriate. Generally JP6 is bridged EXCEPT when you use the board as a simplex range extender. Check general operation. It is at this time that some nice instruments help. Otherwise do what I do, get a group together to monitor your experiments as you adjust levels, flicking to and fro from the direct path to get an idea of similar levels – crude but it works.

Mounting

The production board will fit quite nicely in a small instrument case 110*140*35mm (Dick Smith H2512 - $9.75). A DB25 socket mounted on the rear is a simple way of achieving connectivity to the transceivers.

General notes

Note that in the parts overlay and circuit diagram LED L5 is the wrong polarity. It will need to be inverted. Also note that the pdf overlays have labels to JP2 and JP4 reversed for repeater and simplex audios. Experience suggests that a 22K pot at RV6 may be a better choice than 4K7 and provide better coverage of lower CTCSS frequencies.

The onboard CTCSS system is satisfactory for most tone access situations. Its failing is that the response of the NE567 is not especially sharp. Depending on audio level the NE567 may respond to nearby CTCSS frequencies.

If excellent discrimination between tones is required I recommend the use of a commercial external CTCSS decoder into pin 12 of the main connector. The pin expects active low. Set jumper JP3 to external.

I would like to thank Russell VK3ZQB for converting my hand drawn circuits into Protel so well, creating the parts lists and layout of the board. Without him the board would be hand drawn, twice the size and have many links. Dave Catlin VK2JDC chased up commercial manufacturers for boards. The support of Michael Carra in building a controller and discovering errors in the documentation is also appreciated. Arrio IW6BFE and others have provided valuable feedback, especially with regard to using a 8870 in place of the MC145436. Thanks Russell, Dave, Michael and Arrio. An Italian version of this manual has been written by Arrio and is on my website. It describes an older version but provides an introduction.

Boards

Other than LED L5 shown reversed on the overlay, there are no errors on the board Rev 4. To use this later version of the software intended for the 16F1827 it is necessary to put a jumper under the board – see elsewhere in this document.

I have a small number of commercial boards available for AUS $15 plus post and pack. The boards are double sided, plated through holes and a component mask. In Australia P/P is $2 while overseas orders need $7 for P/P. I have a similar number of MC145436 DTMF decoder chips for $7-50 plus P/P if not sold with board and I can provide programmed 16F1827 for $10 for those who cannot program or obtain a PIC.

Interface issues

The connection to the Carrier Operated Switch (COS or mute) of receivers can vary considerably from radio to radio. Here is some advice that may save some frustrating moments.

- To ease changing COS polarities:
  When wiring R21 and R22 stand the resistors on end with the resistor body in the R21, R22 resistor holes closest to the centre of the board. Bend the long top lead so that it may be soldered into the correct hole near C9. The hole closest to C9 connects the resistor to +5V whereas the inner hole connects it to

0V. If at a later date you need to change polarity it is easy to snip the long resistor tail and bend it out of the way. A new wire can then be soldered into the alternate hole and soldered to the remaining tail of the resistor. Board photos on this site do not YET show this suggestion. Alternatively leave R21 and R22 off the board entirely and wire them externally to either +5V or 0V.

- To cater for different COS situations use this table in association with the relevant section of the manual:

COS output circuit	Resistor (R21 or R22)	Software (address 78)
Open collector - neg active	to +5V	neg active
Open collector - pos active	to +5V	pos active
Off high, active low	to +5V	low active
Off low, active high	to 0V	high active

- **Note** that the circuit diagram shows capacitors C14,15,18,21 with negative to the outside world. Some radios have +ve voltage on their microphone inputs resulting in distorted audio and possible failure of the capacitors. I strongly suggest these capacitors are wired with +ve to the outside world. The circuit is wrong as is the board overlay.

- To cater for different PTT situations:

PTT feature	Note	Software (address 84)
0V off, +5V to +12V active	The LEDs indicate in the reverse sense. ie LED glows when the TX is not keyed.	pos active
+3V to +9V off, 0V active	To avoid the TX LED from glowing faintly in the inactive state you may need to add a small diode in the lead from the PTT of the radio to the controller. This is most easily done at JP5 with the cathode of the diode to the appropriate PTT pin of JP5 (pin 3 or 4) and the anode forming the new input connection to the board. This prevents the PTT circuitry from providing a current sink for the LED circuit, it does not affect operation of the PTT.	neg active
+9V or more off, 0V active		neg active

- Note that when using open collector COS equipment in the positive active mode it will be necessary to tie an unused simplex port COS to 0V. This is most easily done on the back of any input plug to the controller. If this is not done the software will think the COS is active. In this mode if the main receiver is turned off the controller will interpret this as a received signal and key the transmitter. In all other modes the controller will ignore loss of receiver power and not key the transmitter(s).

- To use the beep access mode (setting functions 0 and 9) you will need to either retune the NE567 for 1750Hz and bypass or retune the lowpass filter or use an external decoder accessed through pin 12 of J5 and change jumper JP3.

Have fun, I'm prepared to help fellow amateurs within reasonable limits

John Drew VK5DJ
Updated on 4 August 2012 for version 7.04 of software

Appendix 1:

Problems that may occur with some part or interface combinations
1. CTCSS decoder tends to false trigger when no CTCSS tone received.
   The solution is to increase value of C13 to 2uF
2. The PIC seems to have higher pitched tones and a shorter tail than expected
   even though the DTMF works (it is running off internal 4MHz oscillator).
   Solution is to increase C6, 33pF to 100pF to increase the coupling.
3. Distortion on TX audio caused by voltage on the transmitter inputs.
   Solution is to change direction of input/output coupling capacitors, see the
   note on P21 regarding capacitors C14,15,18,21.
4. If the remote control doesn't work on first putting power on the unit make sure
   you have put the jumper on the rear of the board. See page 21 of these notes.

Other issues that may show up:

1. If you are using **remote control** from a different receiver (using command 02
   08) and if you are changing the callsign or message, then the controller may
   seem to lock up if for some reason the 'FF' code was not received. Just send
   an FF or even several FF. This will finish the callsign or message loop and
   return normal control. In this circumstance you should check that the callsign
   or message was correctly updated and if not, repeat the whole process.

   There was a problem in versions prior to Version 6.43 where opening of the
   mute during control would lock the system until power was removed. It was
   caused by the filling of the 16F628 PIC's stack (too many Gosubs and
   arithmetic that used the stack!!). Exceeding the stack size causes unpredictable
   results, in this case a crash. I programmed around this issue, but is even less
   likely to occur with the PIC16F1827 as it has twice the stack size.

2. If the **DTMF decode** seems to be troublesome, especially with the A,B,C,D
   characters this is probably due to out of balance tones in the DTMF. Try
   adding a series 10K resistor on the audio input to the MC145436, with a 0.001
   (up to 0.02) capacitor to ground on the MC145436 input pin. This mod will
   reduce the level of higher frequency DTMF tones. This modification has
   worked in a couple of controllers that were showing intermittent operation.

Appendix 2:

Frequencies for values of the "beep" and "boop" tones with 3.579MHz clock
EEDATA Decimal 86 (hex 56) and 87 (hex 57) respectively

Value	Freq Hz	Value	Freq Hz	Value	Freq Hz	Value	Freq Hz
1	70	**33**	94	**65**	142	**97**	289
2	71	**34**	95	**66**	144	**98**	298
3	72	**35**	96	**67**	147	**99**	309
4	72	**36**	97	**68**	149	**100**	320
5	73	**37**	98	**69**	152	**101**	331
6	73	**38**	99	**70**	154	**102**	344
7	74	**39**	100	**71**	157	**103**	358
8	75	**40**	102	**72**	160	**104**	373
9	75	**41**	103	**73**	163	**105**	389
10	76	**42**	104	**74**	166	**106**	407
11	76	**43**	105	**75**	169	**107**	426
12	77	**44**	106	**76**	172	**108**	447
13	78	**45**	108	**77**	175	**109**	471
14	78	**46**	109	**78**	179	**110**	497
15	79	**47**	110	**79**	183	**111**	526
16	80	**48**	112	**80**	186	**112**	559
17	81	**49**	113	**81**	190	**113**	597
18	81	**50**	115	**82**	195	**114**	639
19	82	**51**	116	**83**	199	**115**	688
20	83	**52**	118	**84**	203	**116**	746
21	84	**53**	119	**85**	208	**117**	814
22	84	**54**	121	**86**	213	**118**	895
23	85	**55**	123	**87**	218	**119**	994
24	86	**56**	124	**88**	224	**120**	1119
25	87	**57**	126	**89**	229	**121**	1278
26	88	**58**	128	**90**	235	**122**	1491
27	89	**59**	130	**91**	242	**123**	1790
28	89	**60**	132	**92**	249	**124**	2237
29	90	**61**	134	**93**	256	**125**	2983
30	91	**62**	136	**94**	263	**126**	4474
31	92	**63**	138	**95**	271		
32	93	**64**	140	**96**	280		

All numbers in this case are decimal and you will need to convert to HEX for entry
into the EEDATA.

This table applies if a 3.579 xtal is running the PIC via the MC145436

Appendix 3:

Frequencies for values of the "beep" and "boop" tones with 4MHz clock
EEDATA Decimal 86 (hex 56) and 87 (hex 57) respectively

Value	Freq Hz	Value	Freq Hz	Value	Freq Hz	Value	Freq Hz
1	78	33	105	65	159	97	323
2	79	34	106	66	161	98	333
3	80	35	107	67	164	99	345
4	81	36	108	68	167	100	358
5	82	37	110	69	170	101	370
6	82	38	111	70	172	102	384
7	83	39	112	71	175	103	400
8	84	40	114	72	179	104	417
9	84	41	115	73	182	105	435
10	85	42	116	74	186	106	455
11	85	43	117	75	189	107	476
12	86	44	118	76	192	108	500
13	87	45	120	77	196	109	526
14	87	46	122	78	200	110	555
15	88	47	123	79	205	111	588
16	89	48	125	80	208	112	624
17	90	49	126	81	212	113	667
18	91	50	129	82	218	114	714
19	92	51	130	83	222	115	769
20	93	52	132	84	227	116	834
21	94	53	133	85	232	117	910
22	94	54	135	86	238	118	1000
23	95	55	137	87	244	119	1111
24	96	56	139	88	250	120	1251
25	97	57	141	89	256	121	1428
26	98	58	143	90	263	122	1666
27	99	59	145	91	270	123	2001
28	99	60	148	92	278	124	2500
29	101	61	150	93	286	125	3334
30	102	62	152	94	294	126	5000
31	103	63	154	95	303	127	10000
32	104	64	156	96	313		

All numbers in this case are decimal and you will need to convert to HEX for entry into the EEDATA.

Appendix 4:

Hex to decimal conversion table

HEX-DECIMAL TABLE		Hex-Deci 00---000	Hex-Deci 33---051	Hex-Deci 66---102	Hex-Deci 99---151	Hex-Deci CC---204	Hex-Deci FF---255
Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci
00---0	10---16	20---32	30---48	40---64	50---80	60---96	70---112
01---1	11---17	21---33	31---49	41---65	51---81	61---97	71---113
02---2	12---18	22---34	32---50	42---66	52---82	62---98	72---114
03---3	13---19	23---35	33---51	43---67	53---83	63---99	73---115
04---4	14---20	24---36	34---52	44---68	54---84	64---100	74---116
05---5	15---21	25---37	35---53	45---69	55---85	65---101	75---117
06---6	16---22	26---38	36---54	46---70	56---86	66---102	76---118
07---7	17---23	27---39	37---55	47---71	57---87	67---103	77---119
08---8	18---24	28---40	38---56	48---72	58---88	68---104	78---120
09---9	19---25	29---41	39---57	49---73	59---89	69---105	79---121
0A---10	1A---26	2A---42	3A---58	4A---74	5A---90	6A---106	7A---122
0B---11	1B---27	2B---43	3B---59	4B---75	5B---91	6B---107	7B---123
0C---12	1C---28	2C---44	3C---60	4C---76	5C---92	6C---108	7C---124
0D---13	1D---29	2D---45	3D---61	4D---77	5D---93	6D---109	7D---125
0E---14	1E---30	2E---46	3E---62	4E---78	5E---94	6E---110	7E---126
0F---15	1F---31	2F---47	3F---63	2F---79	5F---95	6F---111	7F---127
Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci	Hex-Deci
80---128	90---144	A0---160	B0---176	C0---192	D0---208	E0---224	F0---240
81---129	91---145	A1---161	B1---177	C1---193	D1---209	E1---225	F1---241
82---130	92---146	A2---162	B2---178	C2---194	D2---210	E2---226	F2---242
83---131	93---147	A3---163	B3---179	C3---195	D3---211	E3---227	F3---243
84---132	94---148	A4---164	B4---180	C4---196	D4---212	E4---228	F4---244
85---133	95---149	A5---165	B5---181	C5---197	D5---213	E5---229	F5---245
86---134	96---150	A6---166	B6---182	C6---198	D6---214	E6---230	F6---246
87---135	97---151	A7---167	B7---183	C7---199	D7---215	E7---231	F7---247
88---136	98---152	A8---168	B8---184	C8---200	D8---216	E8---232	F8---248
89---137	99---153	A9---169	B9---185	C9---201	D9---217	E9---233	F9---249
8A---138	9A---154	AA---170	BA---186	CA---202	DA---218	EA---234	FA---250
8B---139	9B---155	AB---171	BB---187	CB---203	DB---219	EB---235	FB---251
8C---140	9C---156	AC---172	BC---188	CC---204	DC---220	EC---236	FC---252
8D---141	9D---157	AD---173	BD---189	CD---205	DD---221	ED---237	FD---253
8E---142	9E---158	AE---174	BE---190	CE---206	DE---222	EE---238	FE---254
8F---143	9F---159	AF---175	BF---191	CF---207	DF---223	EF---239	FF---255

Acknowledgement of table source http://www.webelfin.com/webelfindesign/colorconvert.html

Appendix 5:

External relay control using command 12.
Important note: Command 13 must be a 00 as the same processor pin is used.

This facility was added in Version 7.05 as it was possible to dispense with the crystal oscillator input on pin 16 of the 16F1827 by using the internal oscillator. The pin has been redefined as an I/O pin.

Remove (or do not install if you are making a fresh start) capacitor C6 on pin 16 of the PIC.

Use the vacant hole near pin 16 to install a PCB pin for attachment of the external circuit.

It is necessary to protect the PIC by using an external transistor driver for the relay. If you are switching heavy loads then you may need to use a series resistor as low as 250 ohms to sufficiently drive the base of the transistor. Normally I would suggest a series resistor of 2K2 into a transistor with a gain of 100. That should comfortably drive relays that need less than 150mA at 12V.

The PIC outputs can supply up to 20 mA without stress although it is always best to keep dissipation as low as you can.

The diode across the relay reduces the risk of switching transients destroying the transistor.
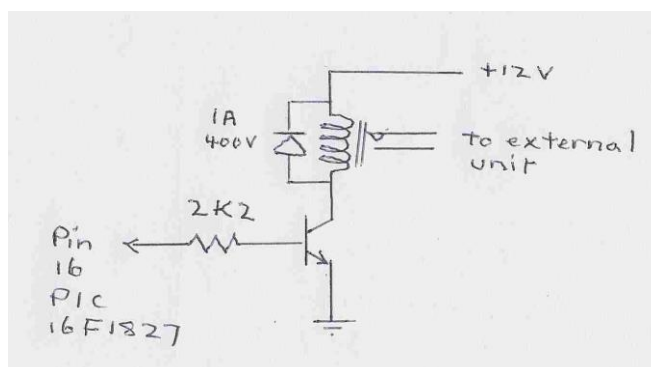
The choice of transistor is not especially critical providing it has a suitable current rating to drive the relay. Because the relay may be on for lengthy periods it may be necessary to heatsink the transistor.

Commands
*<pwd>1200 relay off

*<pwd>1201 relay on

If using voice ID you must have this function off. *<Pwd> 1200
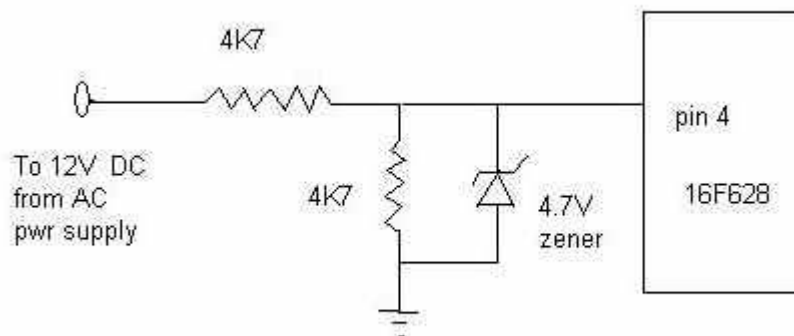
Appendix 6:

AC Fail indication

The controller supports AC Fail indication from Version 6.10

A hardware modification is required because I have had to press the MCLR pin (normally goes to +5V for normal operation) into operation as a port.

On the bottom of the board cut the trace between pin 4 (the MCLR pin) and pin 14 (Vdd - the +5V supply to the chip).

In the clear space near pins 1,2,3 drill a hole to take a pcb pin and glue it in place from the top of the board. Join pin 4 of the PIC to this pcb pin with a short piece of hookup wire.

It is ESSENTIAL that no more than +5V is ever applied to this pin, nor should the pin be left just floating. I suggest the following circuit:

Pin 4 of a 16F1827 (diagram shows a 16F628) may have three functions:
1. MCLR as a reset
2. Port A.5 as an input
3. Sense for end of message when using voice. See Appendix 7

For ACfail to work Command 14 must be set to 1. For example *<Pwd>1401 and Command 13 must be set to 0. For example *<Pwd> 1300

Appendix 7:

Control external voice player board with Command 13.
(Important note: If Command 13 is to be activated it is **critical** that Command 12 must be 00 as the same processor pin is used for both the "external relay" and to switch the audio card. In addition ACfail Command 14 must be set to 0 to turn off the ACfail message.

The play audio feature was added in Version 7.06. Use of the internal oscillator in the PIC1827 makes it possible to dispense with the crystal oscillator input on pin 16 of the 16F1827 by using the internal oscillator. The pin becomes available for another purpose.

Step 1
Do not install capacitor C6 on pin 16 of the PIC. Use the vacant hole near pin 16 to install a PCB pin for attachment of the external circuit. The pin has been redefined as an I/O pin. This pin becomes the 'Initiate play' output to the recorder/player board. Use a short lead to connect this to the PLAYE on the audio board.

Step 2
The next pin to be installed will be on pin 4 of the PIC. To make it available as an input pin it is necessary to cut the trace on the bottom of the board between pin 4 (the MCLR pin) and pin 14 (Vdd - the +5V supply to the chip). In a clear space near pins 1, 2, 3 drill a hole to take a PCB pin and glue it in place from the top of the board. Under the board join pin 4 of the PIC to this new PCB pin with a short piece of hookup wire. Connect the new pin with a short length of wire to the LED pin on the audio board. This LED flashes at the end of the message and the positive pulse is used to tell the controller to shut the transmitter down. It is NOT essential to connect the pullup resistor (see my web site http://www.vk5dj.com/page47.html) providing the audio board is always connected.

Step 3
Connect a common earth lead from the controller board to the GND of the audio board.

Step 4
Power the audio board from the controller board's 0 and 5V lines. Use inspection to find the correct spots. You could tack onto the voltage regulator pins but AVOID the +12V pin because you will destroy the audio board and probably the PIC too.

Commands
*<pwd>1300 CW ID

*<pwd>1301 Voice ID

The recording of your ID will need to be done prior to installing with the controller. The message can be of any length although the ISD1820 is limited to 12 secs in my board. When the playback is initiated the controller operates the PTT of the repeater transmitter. When the playback completes the LED flashes and turns off the transmitter.