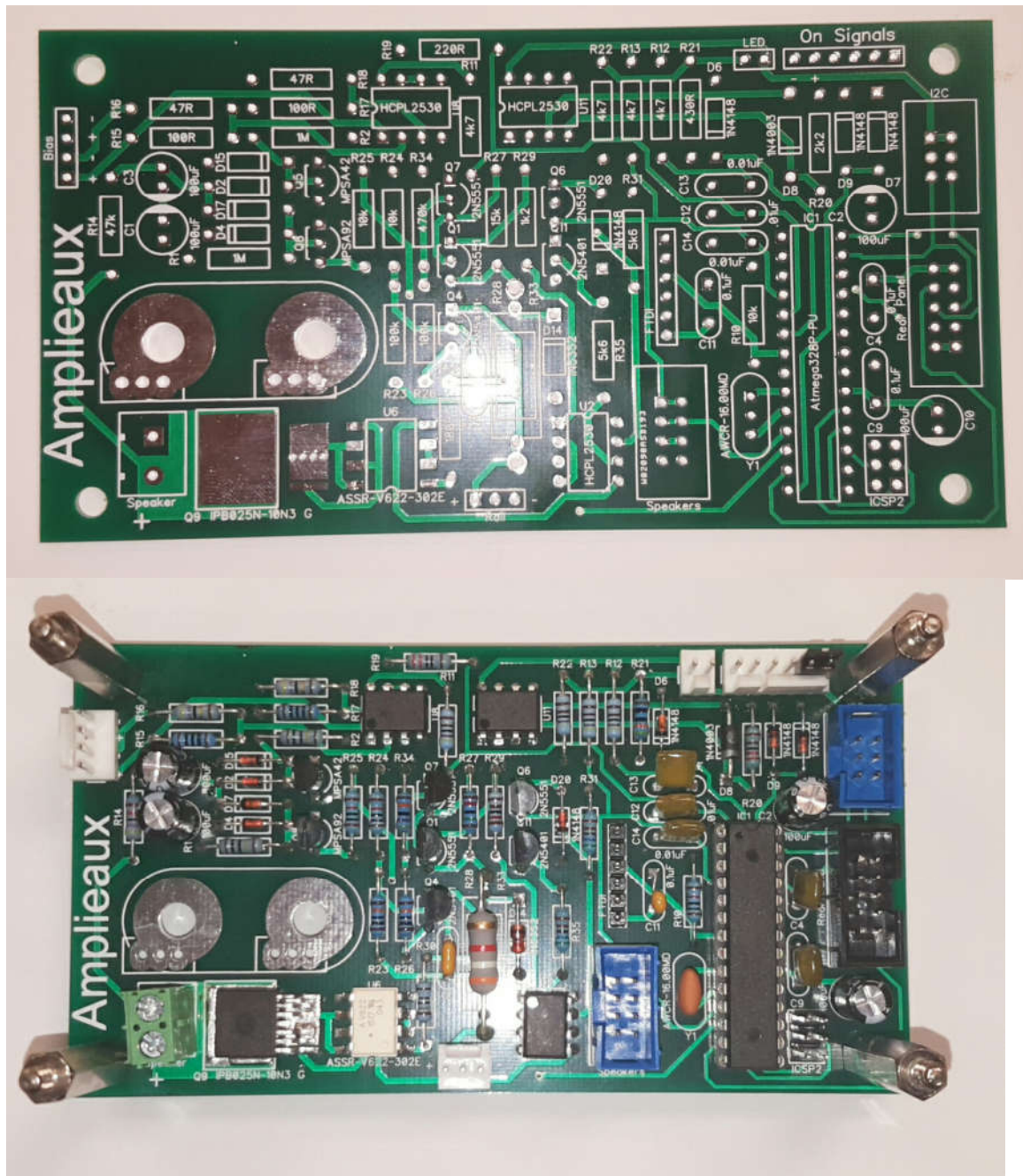


Through Hole Control Board



The Through Hole Control Board is a microprocessor controlled monitoring board that will shut down and send a signal to disconnect speakers on any sign of problem from the amplifiers it is connected to.

It has on board current sensing connections that monitor the voltage drop through a pair of emitter resistors of up to two amplifier channels. It monitors temperature of up to 8 locations with remote temperature sensors connected over an I2C communication bus. It has an on board DC detection circuit to detect DC in the amplifier output signals and discrete a solid state speaker relay to switch speaker output lines off and on. It has connections for a front panel power button, a front panel indicator LED, and a 12V or closed contact style remote trigger system too.

Operation

The control board along with it's supply/relay board take care of all mains switching, fusing and soft-starting of the main supply transformer. If the Amp Control board receives an on signal from either the front panel switch or a remote trigger signal, it then begins a softstart sequence. It first engages an on-board inrush relay that powers the main supply transformer through an on board NTC resistor. After the inrush delay times out a main power relay is engaged. After a programmed delay, DC detection monitoring begins, after which the speaker relays are engaged. Current, temperature and mains voltage loss monitoring immediately begin. If the front panel switch or remote trigger signal is switched off, speaker relays are immediately signaled off, followed by supply rail shut downs if present and the power relay.

If a shutdown event occurs, speaker relays are immediately switched off, followed again by supply rail shut downs if present, and the power relay is switched off. The front panel LED will begin flashing a trouble code indicating the reason for the shutdown.

- Intensive blinking (every 0.2 sec) - DC offset;
- 1 blink every 2 seconds - AC failure;
- 2 blinks every 2 seconds - overheat;
- 3 blinks every 2 seconds - OPS over-current.

BOM

Qty	RefDes	Name	Value	Mouser	Digikey	Newark
	C1,C2, C3,			647-		
4	C10	6.3x11x2	100uF	UVZ1V101MED1TD		65R3213
3	C4, C9, C11	CAP200	0.1uF	80-C322C104K1R	399-9875-1-ND	18K5742
1	C5	CAP200	1uF	80-C340C105K1R	399-4426-ND	18K5927
3	C12, C13, C14	CAP200	0.01uF	80-C315C103K1R	399-9857-1-ND	30X0625
	D2, D4, D6, D7, D9 D15,					
8	D17, D20	DIODE_1N4148	1N4148	512-1N4148	1N4148FSTR-ND	05R0353
1	D8	1N4003	1N4003	863-1N4003G	1N4003GOS-ND	87X9764
1	D14	1N4744	1N4744	78-1N4744A	1N4744AVSCT-ND	33C2792
		Atmega328P-			ATMEGA328P-PU-	
1	IC1	Arduino	PU	556-ATMEGA328P-PU	ND	68T2944
2	J1, J6	Faston		571-624091	A24742-ND	01J0288

1	J2	Rear Panel	52601-S10-4LF	649-52601-S10-4LF	609-3592-ND	84K5868 282836-2
1	J3	282836-2	Speaker	571-2828362	A98076-ND	2
1	J4	HDR-1x4	Bias	538-22-23-2041	WM4202-ND	38C0355
1	J5	HDR-1x2	LED	538-22-23-2021	WM4200-ND	25C3832
1	J8	HDR-1x6	Start	538-22-23-2061	WM4204-ND	38C9178
1	J9	HDR-1x6	FTDI			
1	J10	90131-0123	ICSP2	538-90131-0123	WM8121-ND	60H4442
1	J11	I2C	75869-131LF	649-75869-131LF	609-2845-ND	62K8134
1	J12	Speakers	75869-132LF	649-75869-132LF	609-3530-ND	62K8136
1	J16	HDR-1x3	Rail	538-22-23-2031		
3	Q1, Q6, Q7	2N5551	2N5551	512-2N5551TA	2N5551TAFSCT-ND	31Y5847
2	Q4, Q11	2N5401	2N5401	610-2N5401	2N5401CS-ND	
1	Q5	MPSA42	MPSA42	512-MPSA42	MPSA42FS-ND	97K5369
1	Q8	MPSA92	MPSA92	610-MPSA92	MPSA92CS-ND	
			IPB025N-10N3		IPB025N10N3 GCT-ND47W3462	
2	Q9, Q10	MOSFETN_D	G	726-IPB025N10N3G		
2	R1, R2	RES500	1M			58K3799
3	R10, R24, R25 R11, R12,	RES500	10k			58K3797
4	R13, R22	RES500	4k7			58K3858
1	R14	RES500	47k			58K3860
2	R15, R17	RES500	100R			58K3795
2	R16, R18	RES500	47R			58K3856
1	R19	RES500	220R			58K3827
1	R20	RES500	2k2			58K3828
1	R21	RES500	430R			59K8675
2	R23, R26	RES500	100k			58K3798
1	R27	RES500	15k			58K3813
1	R28	RES700	4k2			58K9229
1	R29	RES500	1k2			58K3805
1	R30	RES500	180R			58K3818
2	R31, R35	RES500	5k6			59K8680
1	R33	RES900	4k2	71-CPF24.221%T1		
1	R34	RES500	470k			58K3861
3	U2, U8, U11	HCPL2530S	HCPL2530 ASSR-V622-	512-HCPL-2530	HCPL2530-ND	67K0214
1	U6	Mosfet Driver AWCR-	302E AWCR-	630-ASSR-V622-302E	516-2689-5-ND	
1	Y1	16.00MD	16.00MD	815-AWCR-16.00MD	535-9355-ND	13J1998

Note – Values of R28 & R33 may need to altered to match rail voltages used in the amplifier. See testing instructions below.

Tools Required

These instructions are written assuming you are using a conventional soldering iron. Basic tools and supplies are required. You will need a fine tip for small parts, as well as a large tip. A good temperature controlled 40W station set to 700F makes soldering very easy. Aside from this you will need the following:

- Isopropyl alcohol
- Liquid or gel flux
- Flux remover
- Good quality flux core solder
- Solder wick or a solder sucker
- Good quality fine tip tweezers
- A good light. A Luxo type with a fluorescent ring tube bulb and magnifier works great
- Multimeter
- Adjustable power supply
- USBtinyISP or similar in circuit programmer
- USB to serial UART. Our rear panel mount USB adapter will work great here.
- A clean work surface, preferably over a smooth hard freshly swept floor.

Assembly

To begin, first wash the circuit board thoroughly with isopropyl alcohol. Wipe it dry with clean paper towel and immediately coat all solder pads with liquid or gel flux to prevent oxidization.

Proceed with assembly by installing all the lower height parts first, and work your way up to the tallest parts.

Before installing the speaker relay Mosfets, insert some cut off resistor leads or single strand wires through the via holes in the pads and solder in place. This is done easily with the board standing vertically while you solder the wires in place. These wires will be carrying the amplifier output signal from one side of the board to the other. The vias alone wouldn't handle the current from the amplifier. If you happen to have some low temperature solder paste apply this to the large Mosfet pad now. Next set a Mosfet in place and reflow the solder. If the Mosfet is straight and sitting flat, solder all the pins down. Next begin heating the tab of the Mosfet and the large pad on the board with an iron with a large tip. This will take a lot of heat. Once solder is flowing go around the whole perimeter with heat and solder. Press down lightly on the Mosfet with your tweezers to squeeze out excess solder. Hold it in place while the solder solidifies. Immediately flip the board over, trim the leads on the opposite side and

install the next Mosfet while the board is still hot. It solders much easier this way. Repeat this with the other pair of Mosfets.

Follow soldering by washing all the excess flux off the board. Flux remover or Isopropyl alcohol will remove most of the goo. This can be followed by a mild soap and water wash and distilled water rinse if you used the suggested wash approved relays. Allow some time for complete air drying before applying power.

Installing the bootloader

To flash the bootloader you will need a USBtinyISP programmer or similar tool. The bootloader is flashed with the Arduino IDE available for download from the Arduino website.

<https://www.arduino.cc/en/Main/Software>

In the Arduino IDE, under tools, select Arduino Uno for board, and USBtinyISP for programmer. Connect the USBtinyISP with the 6-wire ribbon cable to the ISP2 header, paying attention to the location of pin 1. Under tools in the Arduino IDE, select burn bootloader. The bootloader may load in a second, or may take a minute to load. Once it is successfully loaded, you are ready to proceed with loading the software.

Loading the Software

Connect the control board to a supply/relay board. The first software to load is a sketch called I2C scanner. Connect the control board to a computer with a USB to Serial UART. With the I2C scanner sketch open, in the Arduino IDE under tools, you need to select Arduino Uno board again and the port that the serial adapter cable is connected on (usually the last on the list). Hit the upload button, and let the software load. This should take less than 30 seconds. Once upload is complete, open the serial monitor in the Arduino IDE. You should see

Scanning...

I2C device found at address 0x?? (There will be a 2-digit Hexadecimal value in place of the two question marks)

0x?? (Whatever digits are in place of the question marks) is the I2C address of the port expander on the supply/relay board. Record this address for later. Next plug in each other device you are going to be connecting with I2C communication (temp sensors or supply boards) one at a time and record the new I2C address that appears in the serial monitor window. Once you have all your I2C addresses recorded, you are ready to proceed to prepare the Amp Control software for loading.

```

linear gain lab 2014.12.20
Modified for I2C controls 2012.3.14
*/

#include <Wire.h>

const int TubesAreHere =      0;    // 1 = Tubes, 0 = NO Tube
const int Supply1IsHere =     0;    // 1 = I2C supply is present
const int Supply2IsHere =     0;    // 1 = 2nd I2C supply is present
const int temp1IsHere =       1;    // 1 = heat sink temp sensor present
const int temp2IsHere =       1;    // 1 = 2nd heat sink temp sensor present
const int debugEnabled =      1;    // 1 = enable serial data to console

long heatingDelay =           25000; // wait for tubes pre-heating (mS)
long inrushDelay1 =           1000;  // wait for soft start (mS)
long inrushDelay2 =           3000;  // wait for second soft start (mS)
long overlapDelay =           500;   // overlap before inrush goes off, if "0" - stays on (ms)
long speakersDelay =          5000;  // wait before connecting speakers (mS)
const int supply1Address =     0x23;  // address of supply1 I2C expander
const int supply2Address =     0x3F;  // address of supply2 I2C expander
const int relayAddress =       0x39;  // address of Power Relays expander
const int temp1Address =       0x48;  // address of temp sensor 1
const int temp2Address =       0x4D;  // address of temp sensor 2
const int tempLimit =          80;    // shut down temperature in Celcius

```

Preparing the Software

“0” disables an option, “1” enables it.

`const int TubesAreHere = 0;` This is to select tube preheating delay for a tube hybrid design amplifier.

`const int Supply1IsHere = 0;` If you are using a supply with quick shut down rails, enter 1, otherwise enter 0.

`const int Supply2IsHere = 0;` If you are using a second supply with quick shut down rails, enter 1, otherwise enter 0.

`const int temp1IsHere = 1;` If you are installing a I2C temperature sensor, enter 1

`const int temp2IsHere = 1;` If you are installing a second I2C temperature sensor, enter 1

`const int debugEnabled = 1;` If this is enabled, you can watch operation of the power up sequence and see temperature readings in the serial monitor window. This should be disabled when the amplifier is put into service.

`long heatingDelay = 25000;` This is tube pre-heating delay in milliseconds. This is disabled if TubesAreHere = 0; was selected in the first line.

`long inrushDelay1 = 1000;` This is the length of delay in milliseconds between the first inrush relay engagement and the second.

`long inrushDelay2 = 3000;` This is the second inrush delay. This value can be adjusted shorter. Minimum time length should be the length of time it takes for the supply to reach 90% of full charge.

`Long overlapDelay =` 500; This is the amount of time between when the power relay engages and the inrush relay disengages

`Long speakersDelay =` 5000; This is the delay between power on and speaker relay engagement. This can be shorter, but it's necessary to wait until the amplifier has settled and DC offset has dropped to a minimum, otherwise the DC offset may be triggered, or you may hear a turn on pop through the speaker.

`const int supply1Address =` 0x23; If you are using a supply with quick shut down rails, enter the address here, otherwise ignore this.

`const int supply2Address =` 0x23; If you are using a second supply with quick shut down rails, enter the address here, otherwise ignore this.

`const int relayAddress =` 0x39; Enter your relay expander address (the first address you saw in the serial monitor window)

`const int temp1Address =` 0x48; Enter the address of your temperature sensor here.

`const int temp2Address =` 0x48; If you are using two temperature sensors enter the address of your second temperature sensor here.

`const int tempLimit =` 80; Enter the shutdown temperature of the amplifier in Celsius.

Software is ready to run, load it on as you did the I2C Scanner sketch.

Testing

The Amp Control board should now be operational. Connect a LED to the LED header and install a jumper on pin 1-2 of J8 header. Relays should be heard engaging. After the power relays are engaged, the DC detection circuits should quickly activate and cause the board to go into shut down, with the LED blinking rapidly. This is because there are no rail feeds from the amplifier supplies connected to the control board. You will need to connect rail voltages to continue testing. Power will need to be removed and reconnected to the control board to reset the fault.

Overcurrent protection can be tested by applying voltage to the bias connector. Pin 1 is the positive input for 1 channel, pin 2 negative. Pin 3 is positive for the other channel, pin 4 negative. Overcurrent protection should be activated around 2.4V, at which time the relay should click off, and the LED will begin flashing. The control board will now stay locked off until power is disconnected, and reconnected.

Temperature sensor operation can be verified by watching the serial monitor window. Apply heat to the temperature sensor(s) and watch the reading climb until it reaches its shut down point, at which time the relays should click off, the serial monitor window will stop scrolling, and the LED should be flashing a trouble code.

Power loss detection is hard to physically test. You would need to interrupt mains power and reconnect it before the supply cap for the microcontroller drains low enough to cause it to reboot. Measure voltage across C7. It should be above 4V. Shorting pin 2 to pin 3 on U7 on the supply board

should trigger a voltage loss shutdown. This circuit is current limited, so shorting it won't damage anything.

The DC detection circuit is designed to operate at 15VDC from the rail feeds, and is regulated by R28, R33 and D14. R28 and R33 need to be selected to deliver approximately 16mA (8mA each) to the circuit. Their value is calculated by subtracting 15(V) from the rail voltage and dividing the remainder by 0.008(A).

With no DC present on the speaker input connector, current flow through R29 should be approximately 8mA.

If the emitter of Q11 is receiving the speaker on signal, current flow through R30 should measure approximately 8mA. Drain to source resistance of the output Mosfets should be very close to zero ohms when on, and very high when off.

To test operation, apply low voltage to the speaker power input connection with an adjustable power supply. Connect the supply ground to the rail feed ground. Speaker ground on the control board is not connected to the sensing circuit. Slowly increase this until the protection system is activated. This should happen at less than 2VDC. Power cycle the protection board and repeat the test with the voltage source reversed. Again, it should trigger a shutdown at less than 2VDC.

If all is operating properly, the Amp Control board is ready for final software install (disable debugging) and is ready for service.

Circuit Explanations

Microprocessor

The ATMEGA328 (IC1) is a fairly simple microcontroller to operate. It requires 5V supplies to operate and an external 16MHZ crystal (Y1). The supplies are decoupled through C9 and C10.

Digital circuits require a high or a low state for their inputs. If an input is left floating and is allowed to drift to a voltage level somewhere close to the middle of high or low (+5 or 0 in this case) The device reading the signal won't know how to interpret the signal, so it's impossible to tell what it will do in this situation. With all digital circuits, we can't allow this to happen. To stop this, we use pull up or pull down resistors. A pull up resistor is a resistor usually between 1K and 10 k that will hold the signal line at a high state (5V in this case) until the output from a device or circuit pulls it low. A pull-down resistor does the same thing, but pulls the line to 0V instead, allowing a circuit to pull the line high.

There is a reset pin on IC1 that requires 5V for normal operation. We connect this through a "pull up" resistor (R10) to allow us to momentarily ground the reset pin to reset the IC. We need to momentarily ground this for programming.

The digital and analog input/output pins in the Atmega328 can be configured to be either inputs or outputs. We do this by assigning a name to the pin at the beginning of the software, then declaring it an input or output in the setup loop of the software. The setup loop follows the line `"void setup() { "`

`const int PowerSwitch = 8; // const int = constant integer which means the value is an integer, we make it constant so routines in the software aren't allowed to change it's value. PowerSwitch is what we named the power switch input. We assigned it to digital I/O pin 8.`

`pinMode (PowerSwitch, INPUT); //We set the pin mode of the PowerSwitch pin (digital I/O pin 8) as an input.`

When you see a line of code with `“//”` any code before the `“//”` is active code used by the software, anything following the `“//”` is a comment used to explain the line of code to make it easier for anyone reading the code later to understand it.

When I refer to a digital input or digital output in the following text, this is referring to the associated I/O pin with it's assigned duty.

To access different sections of the memory in the microcontroller we add a couple programming ports. We use the ICSP2 (In Circuit Serial Programming) port to flash a bootloader (operating system) to IC1. Our actual software we want to operate and any serial communication we would like to do is done through FTDI (USB to Serial adapter port) connector. We connect the reset line between this port and the reset pin on IC1 to allow us to give a sharp voltage bounce to the reset pin when required.

We have an I2C port to allow us to connect external devices to the system over an I2C bus ribbon cable. This is a digital communication protocol designed by Phillips (now NPX) in the early 1980s. You can read more about it here. <http://www.ti.com/lit/an/slva704/slva704.pdf> We also communicate with the supply/relay board with this.

Start up inputs

We have three options for power on command inputs. We can connect a front panel power button/switch, a rear panel 12VDC trigger input and a rear panel shorted contacts trigger (Audiolab).

The front panel button and the rear panel shorted contact trigger both work the same way. We hold the digital line (Digital Input 9) to the microcontroller at a “high state” (5V) by adding a pull up resistor (R12). We add a little bit of debounce with C13 to filter out any contact arcing in the switches, etc..

The 12V trigger is done through an optoisolator (U11) to protect the microcontroller from accidental damage from the external power supply. An optoisolator is basically an LED for an input light source, and a photo eye light detector for an output. We limit current to the input LED through R20 so we don't burn up the LED, and we add a reverse polarity protection diode (D8) to short the voltage away from the input LED in case power is applied incorrectly. The output from U11 is connected the same digital input line as the other two triggers.

Over-current detection

We use the voltage drop across a pair of emitter resistors to measure current flow in the output stage of the amplifier. We use an optoisolator to do this to protect the microcontroller from the high voltage of the output of the amplifier. There are two “Bias” inputs in the overcurrent connector. Each has it's own voltage divider circuit (R15/R16, R17/R18) to allow us to adjust the voltage required to activate the optoisolator. When the voltage difference between the two emitter resistors becomes high enough, the

optoisolator output circuit activates, and pulls the digital input line to the microcontroller (digital input 3) low. We add a pull up resistor (R11) and debouncing capacitor (C12) to the digital line to hold it normally high and to reduce false triggering.

DC Detection

The control board has a DC detection circuit for one amplifier output, an output Mosfet relay, and a connector to add more DC detection boards to the circuit.

A small sample from the output of the amplifier is taken through R14. Any AC in this sample is shorted to **analog ground** (there are two separate “grounds” in the control board, analog and digital. They are not connected to each other) through a pair of opposite orientation series capacitors (C1 and C3). In normal operation, no voltage will remain after the capacitors. If there is enough DC present, the capacitors will block it from ground and pass it to D2 and D17. If there is positive DC present, current will flow through D2, into the base of Q5. The emitter of Q5 is connected to the emitter of Q8. The base of Q8 is held at near analog ground potential through D4. If the DC voltage becomes high enough to overcome the forward voltage of the two diodes and the two transistor bases, current will flow through R24 and R25. If negative DC is present on the output, it is routed through D17 to Q8 and causes the same reaction. We add R1 and R12 to drain any stray voltage from the bases of Q5 and Q8.

The emitter of Q4 is connected to positive rail voltage from the amplifier. It's base is connected to rail voltage through a 100k resistor (R23) to create a simple constant current source. The current that flows through the collector of Q4 flows to the base of Q7 through a 100k resistor(R26). When there is no current flow through Q5/Q8, Base and emitter voltage of Q4 will be equal, so no current will flow through it's collector. We add C5 to stabilize the voltage to the base of Q4. This is required if there is a large amount of ripple in the rail voltage feeds. We add the 470k resistor (R34) to the base of Q7 to ensure it turns off.

We pass rail voltage through a pair of “ballast resistors” (R28 and R33) and regulate it to around 15V with D14. We pass this voltage to the collector of Q1. We also pass this voltage to the base of Q1 through a 15k resistor (R27) to form a constant current source. Current flows through the emitter of Q1, through the inputs of an optoisolator (U2). We need to put R29 in series with the inputs (LEDs) of U2 to stop them from burning up. In normal operation with no DC present on the output of the amplifier, there should always be approximately 8mA of current flowing through the inputs of U2. This is called operating in failsafe configuration. If there was an open circuit failure in the DC detection circuit, or in the rail feeds to it, U2 would switch off signaling an alarm and turning off the speaker output.

If there is DC present on the output of the amplifier, current will flow through the bases of Q5 and Q8, causing current to flow through the collectors of Q5 and Q8 to rail negative. This will cause voltage to drop at the base of Q4, causing current to flow through it's collector, through the base of Q7. When current flows through the base of Q7, it causes all the available current provided to the base of Q1 to flow through the collector of Q7 to ground. This causes Q1 to switch off, which in turn switches off U2.

DC Offset Alarm

The emitter of Q6 is connected to digital input 2 on the microcontroller. We add a pull up resistor (R13) and a capacitor (C14) to stabilize the voltage here. We feed current to the base of Q6 through a resistor (R31) from the 12V supply. This will cause current to flow through the emitter of Q6, through D14 to ground. This will signal an alarm condition to the microcontroller, telling it there is DC in the output signal of the amplifier, causing a shut down.

In normal operation, the inputs of U2 are active (on). The base of Q6 is connected to output 1 of U2, which causes all the current supplied by R31 to flow through U2 to **digital ground**. This prevents current flow to through the base of Q6, so no current will flow through the collector of Q6, so no alarm signal is detected by the microcontroller.

If the inputs of U2 stop receiving current from the DC detection circuit, current flow stops through the output of U2, starting current flow through the base of Q6 again. We add the diode (D20) in series with the emitter of Q6 to ground to raise the voltage required to cause current flow in the base of Q6 because the on state (low) output of U2 is around 1V.

Mosfet Relay

We use a Mosfet relay driver (U6) to drive a pair of output Mosfets connected in series with opposed orientation. The microcontroller gives an on signal through digital output 6. We pass this signal through the emitter of Q11. The base of Q11 is connected to output 2 of U2 through a resistor (R35). In normal operation the output of U2 will be at a low state, which will cause current to flow through the base and collector of Q11. Current will flow through R30, through the inputs of U6. If DC is present on the output of the amplifier, U2 will switch off, causing current to stop flowing through the base of Q11, turning off the inputs to U6. At the same time, the microcontroller will have received an alarm signal and will have gone into protection shut down, which will shut off the signal from digital output 6 adding a redundant shut down of the Mosfet relay driver.

The Mosfet driver (U6) output approximately 7V when there is current flowing through it's inputs. We parallel the outputs to add some extra current for faster charging and discharging of the output Mosfet gates. U6 also contains quick shut down circuits that quickly turn the output Mosfets off when needed. The output Mosfets contain safety diodes from drain to source to protect themselves so we need to use two in series with opposite orientation to be able to switch off an AC signal.

External speaker relay connector

The external speaker relay connector contains 12V, digital ground, a DC detection alarm connection to digital input 2 from the microcontroller (parallels the onboard DC alarm circuit) and a turn on signal line from digital output 5 of the microcontroller.

