

Adding Auxiliary Inputs to a Car Stereo Head Unit

**By
Ivan Baggett**

Overview

This document describes how to add auxiliary inputs to a 2-shaft style car stereo head unit.

Background

I was wanting to install a good audio system in my 1975 Cadillac Eldorado convertible. However, the factory radio location is the old 2-shaft style. Modern car audio head units use the DIN rectangular dash hole, which is about 7 x 2 inches. Personally, I don't like the way modern DIN units are made. I admit the DIN style is easier to install, but it also makes them easier to steal. The removable faceplate that modern head units have doesn't really solve the theft problem. Unless you remove the faceplate every time you exit the car, it will not be a theft deterrent. Removing and installing the faceplate each time, that's just too much work! So most people will leave the faceplate on, inviting theft. Especially with a convertible, where some mean person might decide to swipe only the faceplate, just to annoy me. The 2-shaft style head units can still be stolen, of course, but it is much more work for the thief. The 2-shaft head unit mounts from behind the dash, and their lack of a removable faceplate means they are a less tempting target for mischief.

For this reason, and also not wanting to cut a bigger hole in the dash of my classic car, I decided to refurbish an old Alpine 7164 AM/FM/cassette head unit that I hadn't used since the late 1980's. However, there was a problem: I haven't listened to cassettes ever since I started burning my own CD's years ago. So, how to play CD's through this old head unit? Two obvious answers: 1) FM modulator, or 2) cassette adapter. I was not thrilled with either of these choices. FM modulators are impractical unless you buy a really good one, with crystal-locked frequency PLL circuitry. The cheap FM modulators you can buy for \$20 to \$40 use variable capacitor tuning, and drift too much. This makes them worthless, in my opinion. Even if you buy a good FM modulator, the audio is still subjected to the limitations of FM modulation, such as frequency range of 50 – 15,000 Hz, and signal to noise ratio of about 50 – 60 dB. Not exactly up to CD standards! My Sony Walkman CD player came with a cassette adapter, which works surprisingly well. However, the audio is still subjected to the limitations of the tape head and preamp. Even though the Alpine 7164 has a good tape section (Dolby B and SCC head), it still falls short of the CD specification. Another drawback is that any dirt on the cassette adapter head can scratch the tape head where they touch. Finally, a cassette adapter looks goofy with it's cord sticking out of the tape slot.

I needed a third choice: an auxiliary input. But this Alpine head unit did not have aux inputs. Then I had the idea that perhaps I could modify the unit to add them.

Theory of Operation

A small circuit board was added to the inside of the Alpine head unit, with circuitry to add the aux input feature. The circuitry (see Fig. 2) consists of a tiny DPDT relay to switch between internal and aux inputs, and a PIC microcontroller to detect knob presses, and manage the switching of the audio sources and display modes. The PIC is inserted between the head unit circuitry and the knob switch contacts. When the PIC detects a knob press of more than 1 second duration, it switches the relay to the other audio source. If the duration of the knob press is less than 1 second, it passes the switch press on to the head unit circuitry (causing the default head unit behavior, i.e. display mode change in radio mode, or reversal of tape direction if in tape mode). Tape mode overrides radio or aux mode, so there is an input to the PIC from the tape switch to detect when the head unit is in tape mode.

To add aux inputs, it was necessary to find the location in the head unit circuitry just before the sound controls (volume, treble, bass, balance, fader). This was easy to do in the Alpine head unit, because this circuitry is located on PCB mounted directly to the left knob pots. Using a cheap battery powered audio amplifier/speaker from Radio Shack, I probed the circuitry on left knob (see Fig. 1) to find where the audio signals were present. Then, by making appropriate cuts of the PCB tracks, and adding some jumpers (see

Adding Auxiliary Inputs to a Car Stereo Head Unit

By
Ivan Baggett

Fig. 3), I was able to bring both audio channels off of and back on to the left knob board. Then, by using a scope and pushing/releasing the left knob, I was able to find the switch contacts that are used to switch display modes, reverse the tape direction, etc.

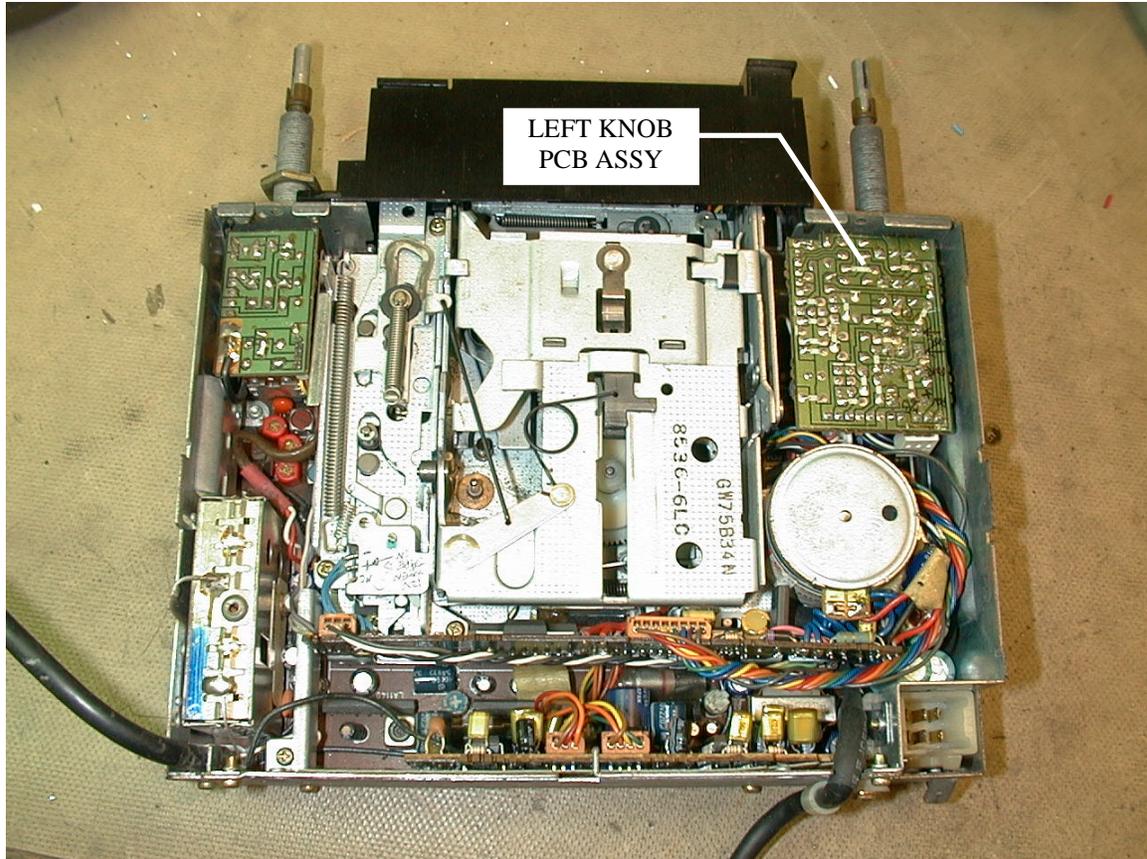


Figure 1

I also added line level inputs and outputs, which is known as a processor loop. This will make it possible to add an in-line equalizer or compander later. The entire modification was stuffed into the gap between the existing PCB's in the rear of the head unit (see Fig. 4).

See the schematic in Fig. 5. The PIC program was written in C using the CCS PIC-C compiler. The source code listing begins on the next page.

Results

Totally awesome! The aux input, when connected to my CD player, allows me to input the CD sound and have it pass through the volume/bass/treble/balance/fader just like radio and tape audio. The 1 second knob push to switch sources is easy to use, and does not interfere with standard head unit functions. Sometime in the future I may add a MP3/Ogg/FLAC jukebox to the aux input. If I need 2 aux inputs, I could add another relay and RCA pair, and use the spare PIC pin to control it.

Adding Auxiliary Inputs to a Car Stereo Head Unit

By
Ivan Baggett

PIC Source Code

```
#include "e:\audio\alpine_7164_mod\deck_mod.h"

/*
  Alpine 7164 AM/FM/Cassette deck modification.
  (C) 2003 Ivan L. Baggett
  Bagotronix Inc.

  Modified deck to add auxiliary input and processor loop.

  PIC Configuration notes:
  GP0, GP1, GP3 weak pull-ups DISABLED
  GP0:  input
  GP1:  input
  GP2:  output
  GP3:  used as -MCLR input
  GP4:  output
  GP5:  output (not used)
  Internal RC oscillator
*/

#byte INTCON = 0x0b    // INTCON register
#byte OSCCAL = 0x8f    // OSCCAL register, page 1 only
#byte STATUS = 0x03    // STATUS register

// #ROM 0x03ff = { 0x347C } // insert PIC OSC calibration value here on UV EPROM CHIPS
// ONLY!!!

#byte GPIO = 0x05
#byte TRIS = 0x85

#bit RELAYDRV = GPIO.2 // Selector Relay driver output (active high to energize
// relay)
#bit KNOBIN = GPIO.1   // Knob Switch input (active low when knob is pushed in)
#bit KNOBOUT = GPIO.4 // Knob Switch output (active high when knob is pushed in)
#bit TAPEIN = GPIO.0  // Tape In switch input (active high when tape is inserted)

// !!! NOTE: PICC compiler integer types are UNSIGNED by default !!!

// state constants
#define STATE_KNOB_PRESS      3 // raw Knob Switch state when pressed
#define STATE_KNOB_PRESS_DB   1 // debounced Knob Switch state when pressed
#define STATE_KNOB_RELEASE    2 // raw Knob Switch state when released
#define STATE_KNOB_RELEASE_DB 0 // debounced Knob Switch state when released

#define STATE_TAPE_IN         3 // raw Tape Switch state when tape is in
#define STATE_TAPE_IN_DB     1 // debounced Tape Switch state when tape is in
#define STATE_TAPE_OUT        2 // raw Tape Switch state when tape is out
#define STATE_TAPE_OUT_DB    0 // debounced Tape Switch state when tape is out

#define STATE_UNIT_RADIO      0 // unit is in Radio mode
#define STATE_UNIT_RADIO_KNOB_PRESS 1 // unit is in Radio mode
#define STATE_UNIT_RADIO_NORM  2 // unit is in Radio mode
#define STATE_UNIT_RADIO_KNOB_ASSERT 3 // unit is in Radio mode
#define STATE_UNIT_AUX         4 // unit is in Auxiliary mode
#define STATE_UNIT_AUX_KNOB_PRESS 5 // unit is in Auxiliary mode
#define STATE_UNIT_AUX_NORM     6 // unit is in Auxiliary mode
#define STATE_UNIT_AUX_TO_RADIO 7
#define STATE_UNIT_AUX_TO_RADIO_2 8
#define STATE_UNIT_TAPE        9 // unit is in Tape mode
#define STATE_UNIT_TAPE_KNOB_PRESS 10 // unit is in Tape mode
#define STATE_UNIT_TAPE_NORM    11 // unit is in Tape mode
#define STATE_UNIT_TAPE_KNOB_ASSERT 12 // unit is in Tape mode
```

Adding Auxiliary Inputs to a Car Stereo Head Unit

By

Ivan Baggett

```
#define KNOB_DB_INTERVAL      10    // # of ticks used for Knob Switch debouncing
interval
#define TAPE_DB_INTERVAL      10    // # of ticks used for Tape Switch debouncing
interval
#define MODE_CHANGE_INTERVAL  1000  // # of ticks used for sensing a mode change
condition
#define KNOB_TAP_INTERVAL     200   // # of ticks used for asserting Tape Switch
output during a Knob Switch normal function
#define DISPLAY_DELAY_INTERVAL 1000  // # of ticks used for delaying aux mode forced
time display when switching to aux mode

int8 tick = 0;                // the "tick" we are in right now
int8 tick_cnt = 0;           // 1 KHz tick counter
int8 tick_knobstate = 0;     // # of ticks knobstate has been in current state
int8 tick_tapestate = 0;     // # of ticks tapestate has been in current state
int8 tick_knobassert = 0;    // # of ticks Knob Switch output has been asserted

int16 tick_knobpress = 0;    // # of ticks Knob Switch has been held pressed
int16 tick_display;

// state variables
int8 knobstate = STATE_KNOB_RELEASE_DB;
int8 tapestate = STATE_TAPE_OUT_DB;
int8 unitstate = STATE_UNIT_RADIO;
int8 oldstate;

// Timer 0 interrupt service routine
#int_rtcc
void tick_handler (void) {
    ++tick_cnt;              // increment 1KHz ticker
}

// program starts here
void main (void) {

    // calibrate the RC oscillator
#asm
    movlw    0x20            // RP0 <- 1
    movwf   STATUS          // set to page 1
    call    0x03ff          // cal value is stored as a RETLW XX instruction, XX is the value
    movwf   OSCCAL          // store the cal value in the oscillator calibration register
#endasm

    setup_timer_0 (RTCC_INTERNAL | RTCC_DIV_4);
    enable_interrupts (GLOBAL);
    enable_interrupts (INT_RTCC);

    // init output states
    KNOBOUT = 0;            // knob is released
    RELAYDRV = 0;          // tuner/tape is selected

    // main program loop
    while (1) {

        while (tick == tick_cnt);    // synchronize with ISR

        tick = tick_cnt;

        // configure I/O pin directions
        TRIS = 0xcb;          // GP2, GP4, GP5 as outputs (0), all others are inputs (1)

        // debounce the inputs
        switch (knobstate) {
            case STATE_KNOB_PRESS:
                if (!KNOBIN) {
                    ++tick_knobstate;
                }
            }
        }
    }
}
```

Adding Auxiliary Inputs to a Car Stereo Head Unit

By

Ivan Baggett

```
    }
    else {
        knobstate = STATE_KNOB_RELEASE;
        tick_knobstate = 0;
    }
    if (tick_knobstate == KNOB_DB_INTERVAL)
        knobstate = STATE_KNOB_PRESS_DB;
    break;

case STATE_KNOB_PRESS_DB:
    if (!KNOBIN) {
        // nothing
    }
    else {
        knobstate = STATE_KNOB_RELEASE;
        tick_knobstate = 0;
    }
    break;

case STATE_KNOB_RELEASE:
    if (KNOBIN) {
        ++tick_knobstate;
    }
    else {
        knobstate = STATE_KNOB_PRESS;
        tick_knobstate = 0;
    }
    if (tick_knobstate == KNOB_DB_INTERVAL)
        knobstate = STATE_KNOB_RELEASE_DB;
    break;

case STATE_KNOB_RELEASE_DB:
    if (KNOBIN) {
        // nothing
    }
    else {
        knobstate = STATE_KNOB_PRESS;
        tick_knobstate = 0;
    }
    break;

default:
    break;
}

switch (tapestate) {
case STATE_TAPE_IN:
    if (TAPEIN) {
        ++tick_tapestate;
    }
    else {
        tapestate = STATE_TAPE_OUT;
        tick_tapestate = 0;
    }
    if (tick_tapestate == TAPE_DB_INTERVAL)
        tapestate = STATE_TAPE_IN_DB;
    break;

case STATE_TAPE_IN_DB:
    if (TAPEIN) {
        // nothing
    }
    else {
        tapestate = STATE_TAPE_OUT;
        tick_tapestate = 0;
    }
    break;
}
```

Adding Auxiliary Inputs to a Car Stereo Head Unit

By

Ivan Baggett

```
case STATE_TAPE_OUT:
    if (!TAPEIN) {
        ++tick_tapestate;
    }
    else {
        tapestate = STATE_TAPE_IN;
        tick_tapestate = 0;
    }
    if (tick_tapestate == TAPE_DB_INTERVAL)
        tapestate = STATE_TAPE_OUT_DB;
    break;

case STATE_TAPE_OUT_DB:
    if (!TAPEIN) {
        // nothing
    }
    else {
        tapestate = STATE_TAPE_IN;
        tick_tapestate = 0;
    }
    break;

default:
    break;
}

// using input states from above, control the output states of the Source Relay
driver and the Knob Switch output
switch (unitstate) {
    case STATE_UNIT_RADIO:
        if (tapestate == STATE_TAPE_IN_DB) {
            oldstate = STATE_UNIT_RADIO;
            unitstate = STATE_UNIT_TAPE;
            break;
        }
        if (knobstate == STATE_KNOB_RELEASE_DB) {
            unitstate = STATE_UNIT_RADIO_NORM;
        }
        break;

    case STATE_UNIT_RADIO_NORM:
        if (tapestate == STATE_TAPE_IN_DB) {
            oldstate = STATE_UNIT_RADIO;
            unitstate = STATE_UNIT_TAPE;
            break;
        }
        if (knobstate == STATE_KNOB_PRESS_DB) {
            tick_knobpress = 0;
            unitstate = STATE_UNIT_RADIO_KNOB_PRESS;
        }
        break;

    case STATE_UNIT_RADIO_KNOB_PRESS:
        if (knobstate == STATE_KNOB_PRESS_DB) {
            ++tick_knobpress;
            if (tick_knobpress == MODE_CHANGE_INTERVAL) {
                unitstate = STATE_UNIT_AUX;
            }
        }
        else {
            tick_knobassert = 0;
            unitstate = STATE_UNIT_RADIO_KNOB_ASSERT;
        }
        break;

    case STATE_UNIT_RADIO_KNOB_ASSERT:
```

Adding Auxiliary Inputs to a Car Stereo Head Unit

By

Ivan Baggett

```
// assert Knob Switch output for normal unit feature
KNOBOUT = 1;
++tick_knobassert;
if (tick_knobassert == KNOB_TAP_INTERVAL) {
    KNOBOUT = 0;
    unitstate = STATE_UNIT_RADIO;
}
break;

case STATE_UNIT_AUX:
    RELAYDRV = 1;           // select aux input
    if (tapestate == STATE_TAPE_IN_DB) {
        RELAYDRV = 0;     // select internal input
        oldstate = STATE_UNIT_AUX;
        unitstate = STATE_UNIT_TAPE;
        break;
    }
    tick_display = 0;
    if (knobstate == STATE_KNOB_RELEASE_DB) {
        unitstate = STATE_UNIT_AUX_NORM;
    }
    break;

case STATE_UNIT_AUX_NORM:
    RELAYDRV = 1;           // select aux input
    if (tick_display < DISPLAY_DELAY_INTERVAL) {
        ++tick_display;
    }
    if (tick_display == DISPLAY_DELAY_INTERVAL)
        KNOBOUT = 1;       // force display to show the clock when in aux
mode

    if (tapestate == STATE_TAPE_IN_DB) {
        RELAYDRV = 0;     // select internal input
        KNOBOUT = 0;     // stop forcing the display
        oldstate = STATE_UNIT_AUX;
        unitstate = STATE_UNIT_TAPE;
        break;
    }
    if (knobstate == STATE_KNOB_PRESS_DB) {
        tick_knobpress = 0;
        unitstate = STATE_UNIT_AUX_KNOB_PRESS;
    }
    break;

case STATE_UNIT_AUX_KNOB_PRESS:
    if (knobstate == STATE_KNOB_PRESS_DB) {
        ++tick_knobpress;
        if (tick_knobpress == MODE_CHANGE_INTERVAL) {
            RELAYDRV = 0;     // select internal input
            KNOBOUT = 0;     // stop forcing the display
            tick_display = 0;
            unitstate = STATE_UNIT_AUX_TO_RADIO;
        }
    }
    else {
        unitstate = STATE_UNIT_AUX;
    }
    break;

case STATE_UNIT_AUX_TO_RADIO:
    if (tick_display < KNOB_TAP_INTERVAL) {
        ++tick_display;
    }
    if (tick_display == KNOB_TAP_INTERVAL) {
        KNOBOUT = 1;       // tap the knob to force radio to display freq
```

Adding Auxiliary Inputs to a Car Stereo Head Unit

By

Ivan Baggett

```
        tick_display = 0;
        unitstate = STATE_UNIT_AUX_TO_RADIO_2;
    }
    break;

case STATE_UNIT_AUX_TO_RADIO_2:
    if (tick_display < KNOB_TAP_INTERVAL) {
        ++tick_display;
    }
    if (tick_display == KNOB_TAP_INTERVAL) {
        KNOBOUT = 0;           // release the knob
        unitstate = STATE_UNIT_RADIO;
    }
    break;

case STATE_UNIT_TAPE:
    if (tapestate == STATE_TAPE_OUT_DB) {
        unitstate = oldstate;
        break;
    }
    if (knobstate == STATE_KNOB_RELEASE_DB) {
        unitstate = STATE_UNIT_TAPE_NORM;
    }
    break;

case STATE_UNIT_TAPE_NORM:
    if (tapestate == STATE_TAPE_OUT_DB) {
        unitstate = oldstate;
        break;
    }
    if (knobstate == STATE_KNOB_PRESS_DB) {
        tick_knobassert = 0;
        unitstate = STATE_UNIT_TAPE_KNOB_ASSERT;
    }
    break;

case STATE_UNIT_TAPE_KNOB_ASSERT:
    KNOBOUT = 1;
    ++tick_knobassert;
    if (tick_knobassert == KNOB_TAP_INTERVAL) {
        KNOBOUT = 0;
        unitstate = STATE_UNIT_TAPE;
    }
    break;

default:
    break;
}

}

}
```

Adding Auxiliary Inputs to a Car Stereo Head Unit
By
Ivan Baggett

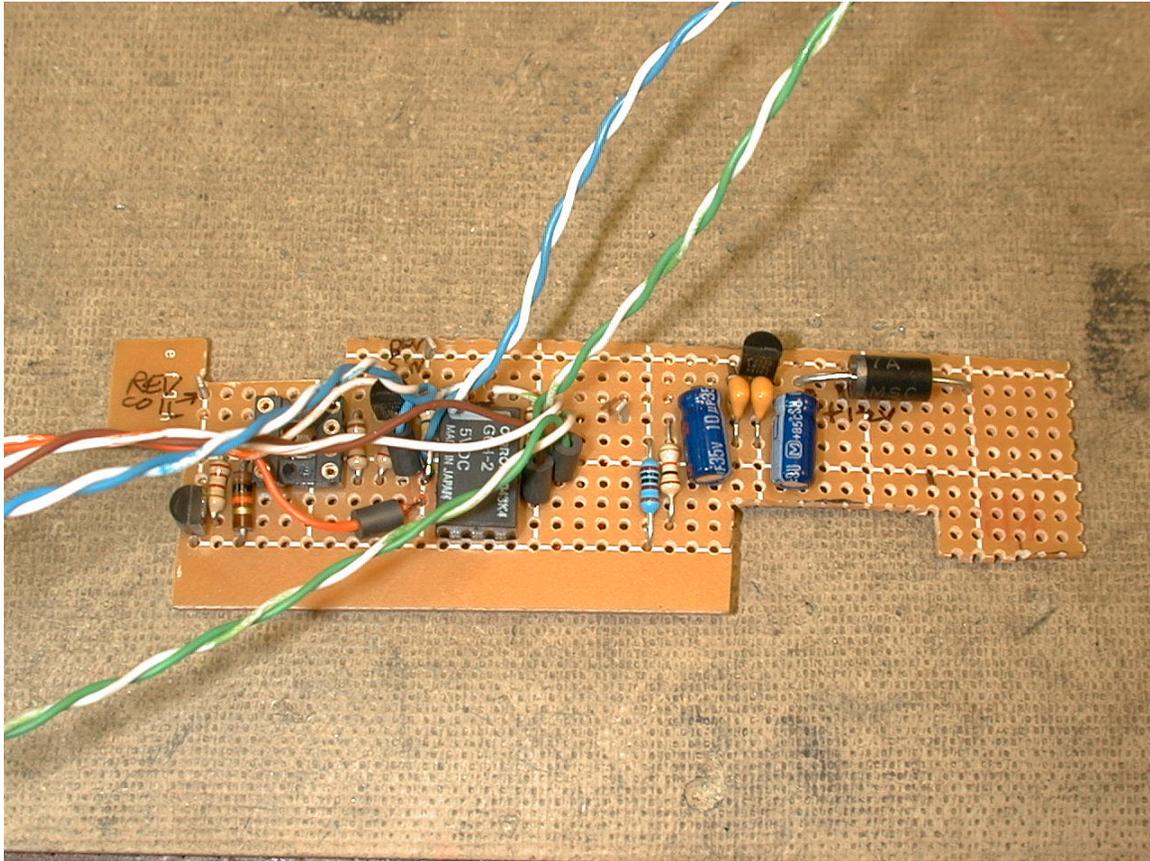


Figure 2

Adding Auxiliary Inputs to a Car Stereo Head Unit
By
Ivan Baggett

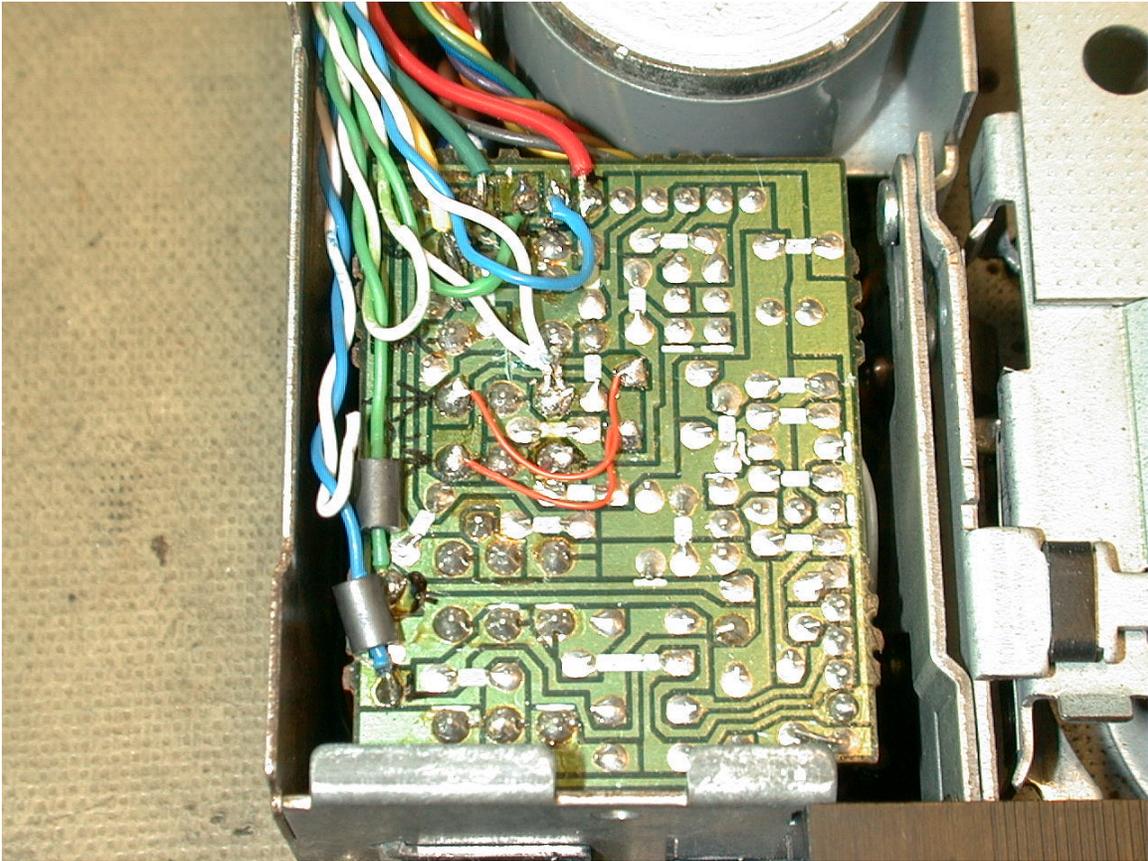


Figure 3

Adding Auxiliary Inputs to a Car Stereo Head Unit
By
Ivan Baggett

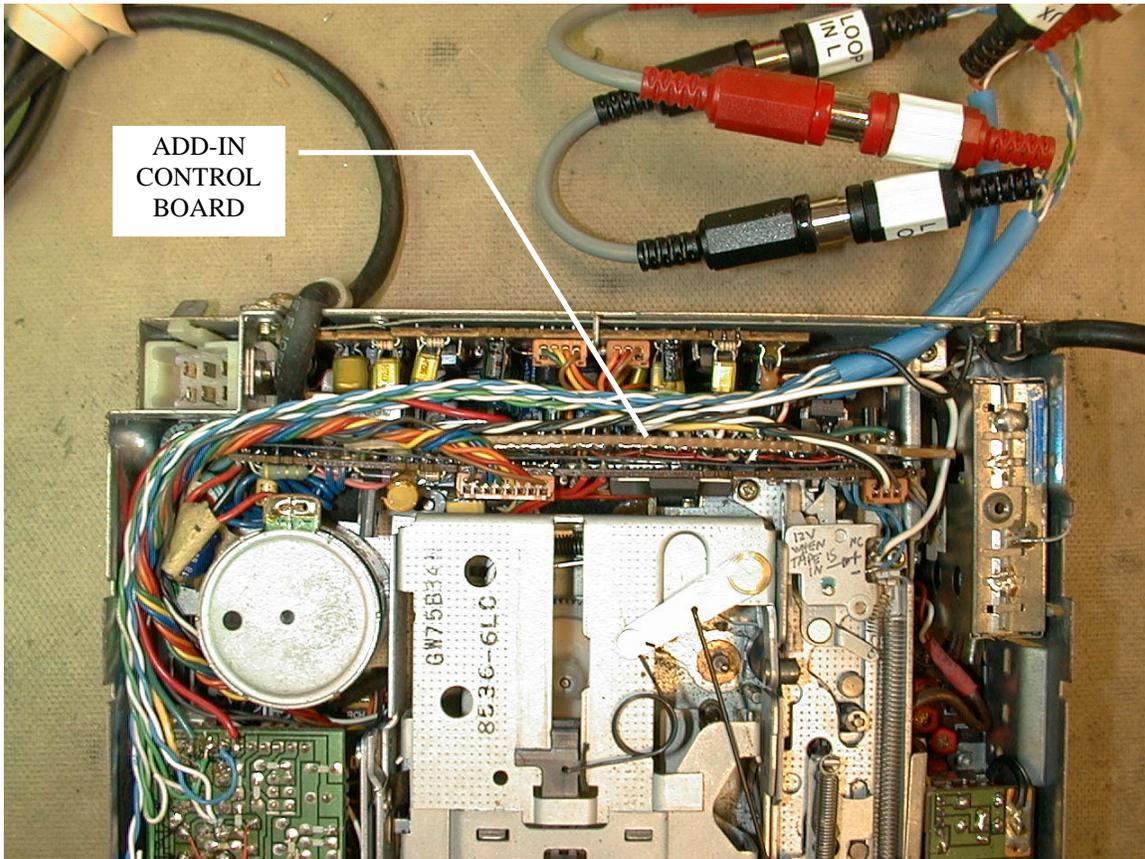


Figure 4

Adding Auxiliary Inputs to a Car Stereo Head Unit

By
Ivan Baggett

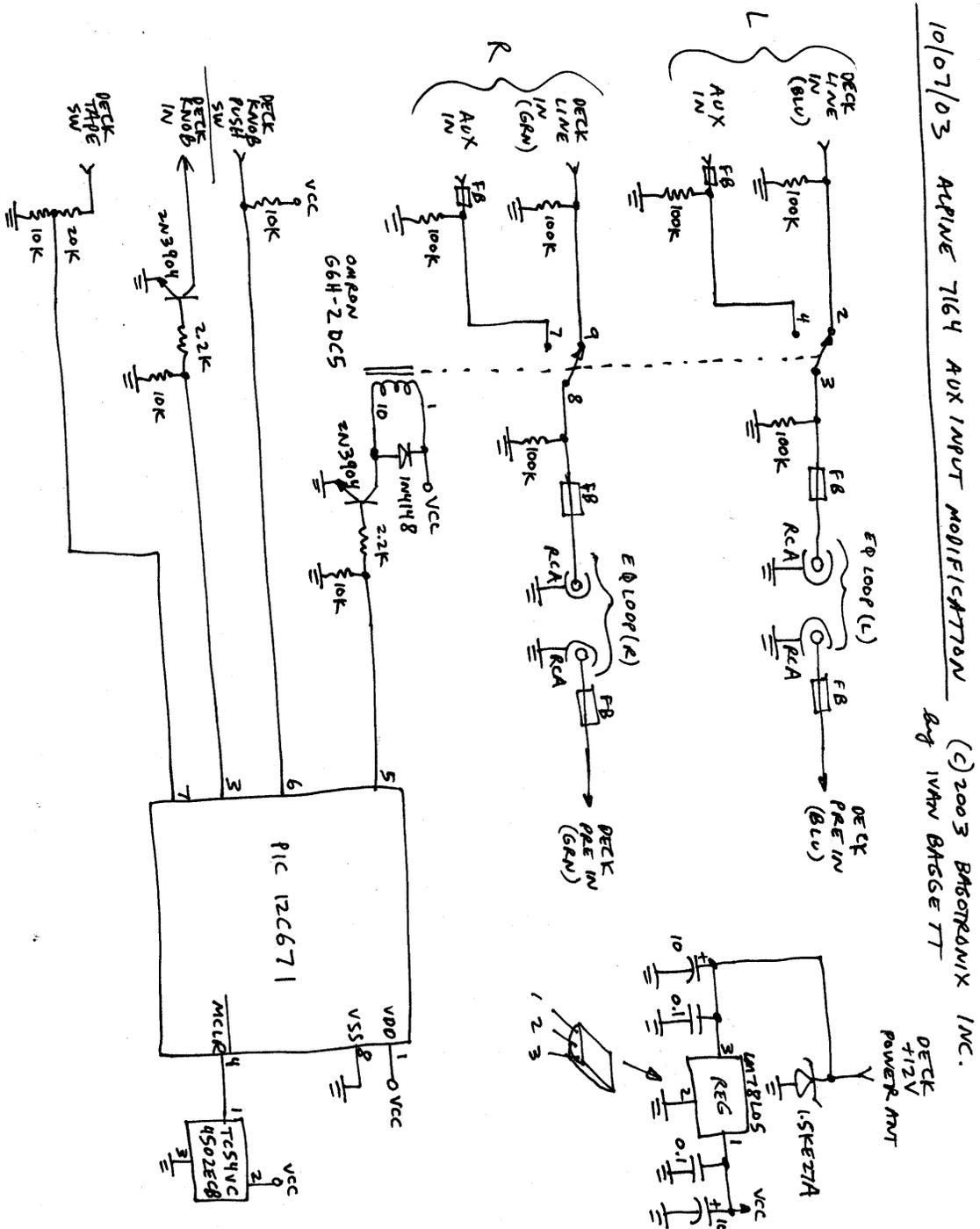


Figure 5