Circuit Bodging
Audio Multiplexer

Audio amplifiers usually come with a single, glaring design flaw: Not enough auxiliary inputs. Not only that, but you’re usually required to press a button to switch between the amplifier’s limited number of inputs. This is unacceptable - we have better things to do than change input channels! In the spirit of encouraging laziness (usually dubbed ease-of-use), here’s a little gadget that will make your life easier.

Author: Ben Allen
Circuit Idea: Erik Roeling

The purpose of the circuit is to accept multiple inputs, and to switch the output to an active input automatically, with no user intervention. Also included is a user override button.

Switching the Audio

The first question to answer is “How do we present a single input signal to the output of the circuit?” The answer is fairly straightforward: We chose to use relays as shown in figure 1, as they are simple to implement. The reader can choose to use other methods such as triacs or FET switching if they feel inclined to do so. The relay in the schematic is shown in the ‘on’ position. Because the PIC cannot source enough current to drive the relay, a transistor is used for switching. The diode D1 provides a path for the coil in the relay to discharge.

Audio Logic

The next question is “How do we find out if the active input is presenting a signal?” The solution to this is presented in figure 2 on the next page. Since monaural signals are always transmitted on the left channel, only the left channel is monitored. This saves components, time spent soldering and eventually, money. The incoming signal must be amplified significantly to cause it to clip, and then cleaned up. This gives us a logical 0 when there is an audio signal present, and a logical 1 when there is not. The voltage gain was chosen to be about 100 - in figure 2 it is 101 (as determined by \((1 + (R1/R2))\), which is close enough. The resulting voltage is filtered by C1. This is then passed through a Schmitt trigger inverter to clean the signal up. Not that the schmitt trigger featured is a surface mount component and might be difficult to solder. Another note is that the opamp featured is a TL072, a two op-amp 8-pin package. This can be substituted for a single opamp IC, but the pinout will change. To prevent noise from playing too much of a part, R4 provides a channel to ground, but are chosen sufficiently large to not overload the audio too much.

Automatic Switching

Now we know if an audio signal is present or not. All that remains is to design a circuit that will automatically switch to the next input if the audio-to-logic circuit gives us a 1. To further save components and cut costs, only the output is monitored for an audio signal. This has the disadvantage that we have no idea what is going on with the other audio signals - we only know what is happening with the currently selected channel.

The Heart of it All

The selection circuit can be implemented in two ways. One option is to create combinatoric logic to select the correct output...
Correction

In the previous edition of Maxwell we featured an FM transmitter. After an email from an observant reader, it was determined that the allowed frequency for low power transmitters in Holland is 87.5MHz, and not the 90MHz featured in the article. The author apologises for the error.

Expansion

The circuit featured has 4 inputs, but this can easily be expanded by using a larger microcontroller with more IO pins. The code is easy to modify as only the next_output() function needs to be rewritten, and the modular relay design allows inputs to be added fairly easy.

The PIC Source Code:

```c
#include <12F675.h>
//FUSES INTRC_IO //Use the internal osc
//use delay(clock=400000000)

VOID next_output ()
{
    // This function sets the next output high.
}

//INT_EXT
VOID interrupt_routine ()
{
    //When the button is pressed, this function is called.
    next_output ();

    //To prevent the user from zipping through all
    //the channels faster than the speed of light,
    //the function waits 250ms after switching.
    delay_ms(250);

    // clear interrupt flag manually to be safe
    clear_interrupts(INT_EXT);
}

VOID main ()
{
    //Set out / inputs.
    SET_TRIS_A (0b00001100);
    enable_interrupts (INT_EXT);
    enable_interrupts(GLOBAL);

    WHILE (TRUE) // Loop forever
    {
        IF (INPUT (PIN_A3))
        {
            //Wait 750 ms, then check again.
            delay_ms (750);

            IF (INPUT (PIN_A3))
            {
                next_output ();
            }
        }
    }
}