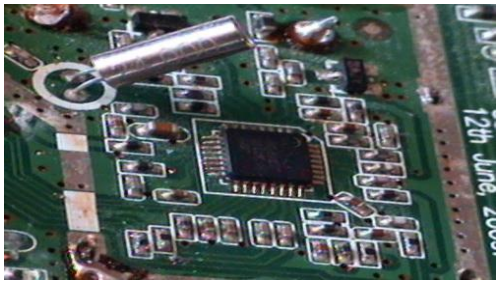
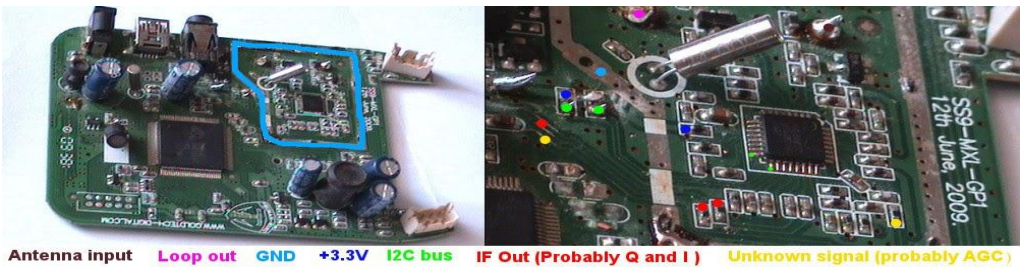
MXL5007T TUNER RADIO



MxL5007T is a tuner IC designed mostly for digital signals (DVB-T, ATSC), but it can be used for analog reception too. It has a programmable IF output and it can receive anything from 44 to 885 MHz. There is no datasheet for it, but there are Linux drivers.

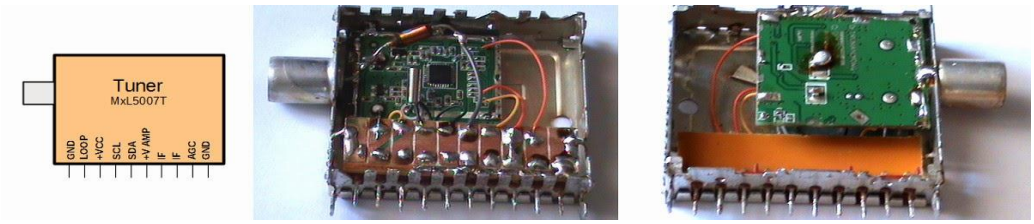
Hardware

I found this tuner in a SCART DVB-T receiver (Cobra Giraffa 180). Although the receiver specifications say it can receive 174-230 and 470-862 MHz these are only software limitations. Getting the tuner out of the receiver is no easy task because it is on the same PCB. First I had to trace some lines due to the lack of pinout information. Then I cut (really) the PCB and placed the tuner inside a metal box with easy to make connections.



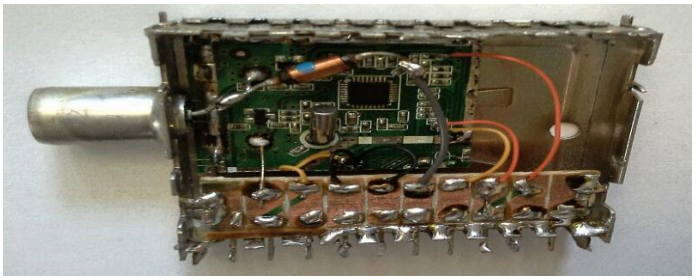
Left: the receiver PCB. The tuner is marked. Right: the IC connections and their meaning.

- **Antenna input** and **Loop out** - these were the easiest to identify as the original antenna connectors went straight into them.
- **3.3V** was guessed because it was the biggest width PCB track that entered the tuner. Further tracing showed it comes out of a 3.3V regulator.
- **I2C bus** was identified by the 4.7k pull-ups to 3.3V. I don't know which is SDA and which is SCL but that isn't too hard to find once you connect it to an I2C bus and read some identification registers. A wrong connection means that no START command will be correctly received, so the IC can't be damaged.
- **IF Out** - there are two lines that came out of the tuner area and once they get out of the IC, they are capacitor coupled.
- Another track is coming from the RF input (not marked in the picture) - this must be the supply for an amplifier mounted at the antenna.
- There is one last track going to the MxL5007, which I don't know what it is. Probably it is some AGC signal (maybe IF or RF).

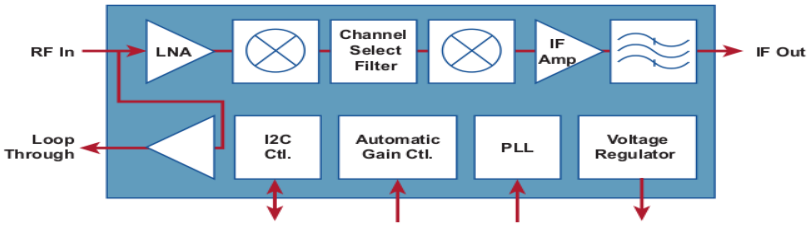


The tuner PCB inside a metal box from a broken tuner.

Looking at the block diagram of the IC, it can be seen that there is an AGC input. The voltage regulator powers the IC (probably the PLL?). It has no external function.

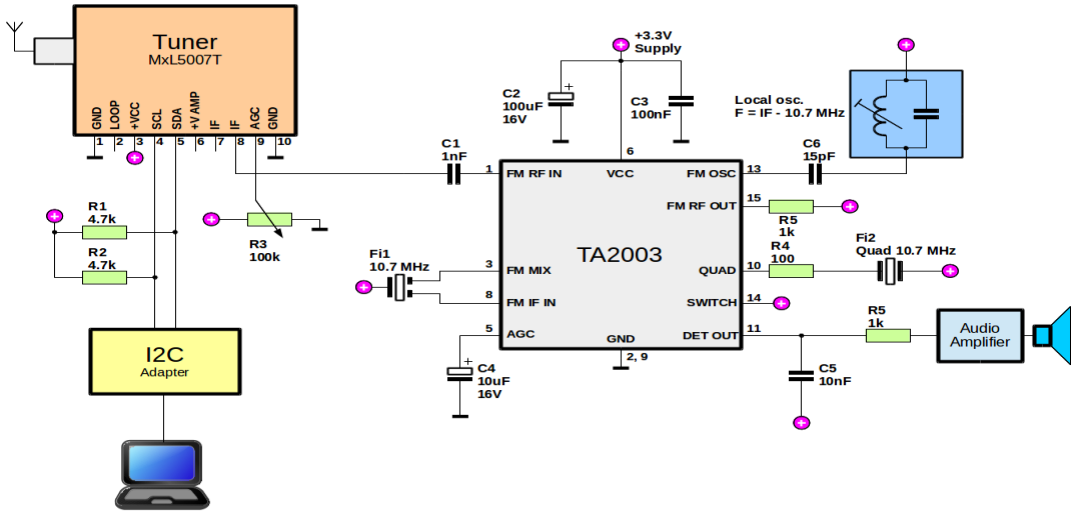


Larger image of the tuner



MxL5007T Block diagram (Source: Product Brief)

To test the reception and figure out how that AGC works I needed to build an FM detector. I choose a simple schematic using one dedicated IC: TA2003.



TA2003 FM detector for the tuner. Schematic based on datasheet and Uzzors2k webpage (CC-BY-NC-SA-3.0).

Local oscillator

The local oscillator's frequency depends on the intermediate frequency (IF) output from tuner. MxL5007T supports different IF values, which can be found from the Linux drivers mentioned in the **Software** section.

The IF values are as follows: 4 MHz, 4.5 MHz, 4.57 MHz, 5 MHz, 5.38 MHz, 6 MHz, 6.28 MHz, 9.1915 MHz, 35.25 MHz, 36.15 MHz and 44 MHz. For the detector I used, in low-side injection mode (local oscillator frequency is smaller than received frequency - the IF from tuner in this case) only the 35.25 MHz, 36.15 MHz and 44 MHz values can be used. Thus, the L.O. can have one of the following frequencies: 24.55 MHz, 25.45 MHz and 33.3 MHz. You can get both the coil and the capacitor from:

- a CB radio. They work on frequencies close to 27 MHz. Note that some have crystal oscillators.
- a R/C toy. Some use 27 MHz but others 49 MHz. The can use crystals too.
- the sound path of some analog TV sets. The picture IF is usually 38.9 MHz and the sound IF is 33.4 MHz.

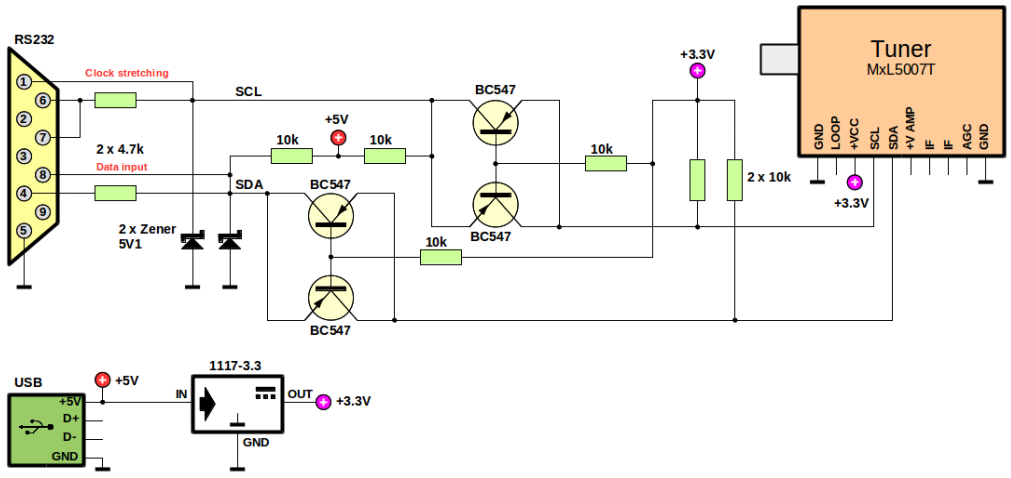
Can a crystal oscillator be used? Yes. The tuner has a bandwidth of 6 to 8 MHz. Thus, a crystal oscillator can only be used if:

$$|fcrystal+10.7-IF|<ftunerbw-frecbw2$$

- fcrystal is the crystal frequency in MHz
- IF is the intermediate frequency from tuner in MHz
- ftunerbw is the tuner bandwidth (6, 7 or 8 MHz)
- frecbw is the bandwidth of the received signal. For FM radio, according to Carson rule, the bandwidth is maximum 0.256 MHz (source: Wikipedia).

You can also build the coil. You need 10-15 turns on a slug tuned inductor. You will pair this with a 12-20 pF capacitor.

PC Adapter



The adapter I built to connect the tuner to PC. Based on Claudio Lanconelli's SIProg and I2C level shifter from Arduino Playground.

A microcontroller is needed for that. Arduino works too. However I used a simple serial port adapter and I bitbanged I2C over serial DTR and RTS. Unfortunately this works only on PCs with motherboard serial ports and not USB adapters.

I could have used 3.3V Zeners to limit RS232 signals to 3.3V without a level shifter but I'm not sure that a 3.3V level sent from tuner can be detected by the serial port as HIGH.

Software

The I2C control software is based on Michael Krufky's drivers for Linux available from linuxtv.org (mxl5007t.c andmxl5007t.h).

Identification

At this step, there is no need for a demodulator circuit. Just hook up the tuner to the I2C bus and power it (see the above schematic).

MxL5007T I2C address is 0xC0. Control variables are stored in registers. To read the ID register, the following commands are needed:

- Write: Start-C0-FB-D9-Stop
- Read: Start-C1-??-Stop

0xC0 is the write address, 0xC1 is the read address, 0xFB is probably a command that tells the IC to prepare the register for reading and 0xD9 is the actual register.

The return value can be:

- 0x11 for **v1 f1**
- 0x12 for **v1 f2**

- 0x21 for **v2.100 f1**
- 0x22 for **v2.100 f2**
- 0x23 for **v2.200 f1**
- 0x24 for **v2.200 f2**
- 0x14 for **v4**

Note that this register seems to be readable only after reset (at power-up) and in sleep mode if your I2C bus doesn't support clock stretching (in this case use a low speed bus). Once this has been read, it means that I2C pins were identified: SDA is connected to MxL5007T pin 16 and SCL to pin 17. My MxL5007T returned 0x14, so it's version 4.

The code listing

This is just an adaptation of Michael Krufky's driver. The code is AVR-GCC compatible. To use it, you'll need a implementation of these I2C functions in a header named i2c.h:

- int i2c_start(void); - returns 1 if successful.
- int i2c_stop(void); - returns 1 if successful.
- int i2c_write(char c); - returns 1 if write was acknowledged, otherwise 0; c is the variable to be written.

- char i2c_read(int ack); - returns the read 8-bit value; ack is 1 if the read should be acknowledged, otherwise 0.

```
#include "i2c.h"

enum mxl5007t_if_freq {
 MxL_IF_4_MHZ,   /* 4000000 */
 MxL_IF_4_5_MHZ,  /* 4500000 */
 MxL_IF_4_57_MHZ,  /* 4570000 */
 MxL_IF_5_MHZ,   /* 5000000 */
 MxL_IF_5_38_MHZ,  /* 5380000 */
 MxL_IF_6_MHZ,   /* 6000000 */
 MxL_IF_6_28_MHZ,  /* 6280000 */
 MxL_IF_9_1915_MHZ, /* 9191500 */
 MxL_IF_35_25_MHZ, /* 35250000 */
 MxL_IF_36_15_MHZ, /* 36150000 */
 MxL_IF_44_MHZ,   /* 44000000 */
};
enum mxl5007t_bw_mhz {
 MxL_BW_6MHz = 6,
 MxL_BW_7MHz = 7,
 MxL_BW_8MHz = 8,
};
enum mxl5007t_mode {
```

```
 MxL_MODE_ISDBT   =  0,

 MxL_MODE_DVBT   =  1,

 MxL_MODE_ATSC   =  2,

 MxL_MODE_CABLE   = 0x10,

};

// Public functions

char* mxl5007_get_id(void);

void mxl5007_sleep(void);

void mxl5007_init(enum mxl5007t_mode mode, enum mxl5007t_if_freq if_freq, int invert_if,
int loop_through);

void mxl5007_set_frequency(unsigned long int rf_freq_hz, enum mxl5007t_bw_mhz
bandwidth);

int mxl5007_is_locked_ref(void);

int mxl5007_is_locked_rf(void);
```

This is only the C header file, which shows the 6 public functions and their parameters. Download here the full code.

Setting the XTAL registers

In the C file, in the mxl5007_init function, there are two registers set for a 24 MHz crystal. For a different crystal replace those two set_reg_bits calls with one of the following:

```
// 16 MHz

set_reg_bits(init_tab, 0x03, 0xf0, 0x00);
```

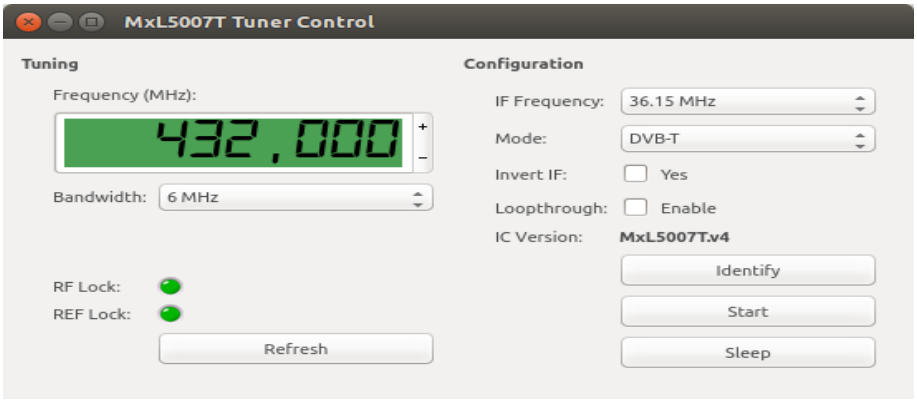
```
set_reg_bits(init_tab, 0x05, 0x0f, 0x00);
  // 20 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x10);
set_reg_bits(init_tab, 0x05, 0x0f, 0x01);
 // 20.25 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x20);
set_reg_bits(init_tab, 0x05, 0x0f, 0x02);
 // 20.48 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x30);
set_reg_bits(init_tab, 0x05, 0x0f, 0x03);
 // 24 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x40);
set_reg_bits(init_tab, 0x05, 0x0f, 0x04);
 // 25 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x50);
set_reg_bits(init_tab, 0x05, 0x0f, 0x05);
 // 25.14 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x60);
set_reg_bits(init_tab, 0x05, 0x0f, 0x06);
// 27 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x70);
set_reg_bits(init_tab, 0x05, 0x0f, 0x07);
```

```
 // 28.8 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x80);
set_reg_bits(init_tab, 0x05, 0x0f, 0x08);
 // 32 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0x90);
set_reg_bits(init_tab, 0x05, 0x0f, 0x09);
 // 40 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0xa0);
set_reg_bits(init_tab, 0x05, 0x0f, 0x0a);
// 44 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0xb0);
set_reg_bits(init_tab, 0x05, 0x0f, 0x0b);
// 48 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0xc0);
set_reg_bits(init_tab, 0x05, 0x0f, 0x0c);
// 49.3811 MHz
set_reg_bits(init_tab, 0x03, 0xf0, 0xd0);
set_reg_bits(init_tab, 0x05, 0x0f, 0x0d);
```

Tests and results

I've controlled the tuner with a small application written in C++ using Qt which works with the serial port adapter shown in the **Hardware** section. One thing that I noticed: MxL5007T supports very fast I2C.

First, I used no delay when bitbanging I2C and in standby mode it recognized the protocol. When turned on, the maximum bus speed slows down a bit and a clock stretching implementation of I2C is highly recommended.



Screenshot of tuner control application

To receive anything you need a strong input signal or a voltage on AGC pin of the tuner. To receive FM radio with an outdoor antenna, my tuner needed about 2 V via a voltage divider into the AGC pin. Higher than needed voltage (no more than VCC = 3.3 V) doesn't seem to affect the signal.

The reception quality is very good and although I could listen to analog FM sound on the correct frequencies, there is something odd. Tuning to the correct frequency of a radio station gives a distorted sound. To get a clear sound I have to tune to a frequency with 2.75 MHz higher, as if the detector would be set for 38.9 MHz reception (I used 36.15 MHz IF and 38.9 - 36.15 = 2.75 MHz). The same for analog TV. This made me think the detector local oscillator (L.O.) frequency is not correct. And, yes.

My detector couldn't even tune to 25.45 MHz (36.15 - 10.7 MHz). Setting the IF to 44 MHz fixed the issue. This is how you can tune your detector. Set the tuner to a frequency of a known station/signal and tune the slug inductor from detector's L.O. until you get the best reception.

IF inversion does not affect the signal unless your detector is not centered on the right IF frequency. ISDB, DVB and ATSC modes don't seem to change anything.

There is no difference between the two IF outputs, Just connect one to the detector. Do **not** short them. Do not connect the other to the ground.

Looking at the code, the frequency step seems to be 7.8125 kHz which is very small as common analog TV tuners have 31.25 up to 62.5 kHz frequency steps.

Source: http://onetransistor.blogspot.in/2014/08/mxl5007t-based-radio.html