# Getting Started with FM Radio on Maemo

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The article is believed to be still valid for the original topic scope.

## Introduction

Some Maemo based devices, such as N800, have embedded FM radio support. Therefore, this article explains how to develop an application based on the FM Radio support.

It is possible to develop an FM Radio application using two different approaches, namely:

1. Direct FM Radio access using the V4L2 ( Video4linux 2) API;
2. 3rd party APIs which wrapper the previous approach, such as the Gstreamer Framework and the Easy API.

More specifically, in this article, it is shown how to develop an FM Radio applications using the V4L2 API.

## FM Radio using V4L2

V4L2 is a Linux API which make possible to access web cams, TV tuners cards, and Radio devices. As Maemo is a Linux based distribution, V4L2 support is already integrated in the distribution.

To listen to your favorite FM radio station on Maemo, you have to follow these steps:

1. Open the Radio Device
2. Open the Mixer Device to controls the volume
3. Set the FM radio frequency
4. Set the volume

### Open and prepare the Radio Device

V4L2 uses a set of defined structures which are used to set devices capabilities, tuner characteristics, tuner frequencies, and other features. Four specific structures are used in this example, which will be held in one application specific structure:

```
struct RadioDescriptor
{
  /* V4l2 structures */
  struct v4l2_capability vc;
  struct v4l2_tuner vt;
  struct v4l2_control vctrl;
  struct v4l2_frequency vf;

  int radio_fd;
  int mixer_fd;

  unsigned int freq;
  unsigned int  vol;

  boolean silent;
};
```

Also, in this structure one can store the volume level, the tuned frequency and file descriptors for both, the radio and the mixer device.

The radio and mixer devices are in the following location:


(C) Copyright Nokia 2014. All rights reserved.

```
#define DEFAULT_DEVICE "/dev/radio0"
#define DEFAULT_MIXER  "/dev/mixer"
```

Also, it is necessary to include the following headers:

```
#include <linux/videodev.h>
#include <linux/soundcard.h>
#include <sys/ioctl.h>
```

Now to prepare the radio device, we should open the radio device, query its capabilities, set the tuner index, and open the mixer device. This sequence is shown in the follow code example:

## Open the radio device

```
int radio_open(RadioDescriptor *radioDesc)

{
 // Open the Radio device.
          if ((radioDesc->radio_fd = open(DEFAULT_DEVICE, O_RDONLY))< 0)
          {
                  goto error;
          }

      // Query Radio device capabilities.
          if (ioctl(radioDesc->radio_fd, VIDIOC_QUERYCAP, &(radioDesc->vc))<0)
          {
              goto error;
          }

      // Set tuner index. Must be 0.
          (radioDesc->vt).index = 0;
          ioctl(radioDesc->radio_fd, VIDIOC_S_TUNER, &(radioDesc->vt));

          if (!((radioDesc->vc).capabilities & V4L2_CAP_TUNER))
          {
              goto error;
          }

      // Open the Mixer device.
          radioDesc->mixer_fd = open(DEFAULT_MIXER, O_RDWR);
          if (radioDesc->mixer_fd < 0)
          {
                  goto error;
          }

          return 1;

error:
         if (radioDesc->radio_fd >= 0)
         {
                 close(radioDesc->radio_fd);
                 radioDesc->radio_fd = -1;
         }
         return 0;
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

## Listen your favorite FM Radio Station

After initializing the radio device, we must set the frequency to tune, and set the volume.

First, let set the radio device as not muted. To do this, let us use the control V4L2 structure as follow:

```
void radio_unmute(RadioDescriptor* radioDesc)
{
 int res;

//Set the feature ID to controls
 (radioDesc->vctrl).id = V4L2_CID_AUDIO_MUTE;

//Set the value to set to the previous ID
 (radioDesc->vctrl).value = 0;

//Send the control to the opened device
 res = ioctl(radioDesc->radio_fd, VIDIOC_S_CTRL, &(radioDesc->vctrl));
 if( res > 0 )
 {
  radioDesc->silent = 0;
 }
}
```

Then, lets set the volume in mixer device as follow:

```
void radio_setvolume(RadioDescriptor* radioDesc, unsigned int vol)
{
 int res;

 vol |= (vol << 8);
//Sending the volume to the mixer device
 res = ioctl(radioDesc->mixer_fd, SOUND_MIXER_WRITE_LINE, &vol);
 if(res > 0)
 {
  radioDesc->vol = vol;
 }

}
```

After setting the volume to listen, lets set the station frequency as follows:

```
int radio_setfreq(RadioDescriptor* radioDesc, unsigned int freq)
{
 int res;

//Check if the radio device is opened
 if (radioDesc->radio_fd < 0)
  return -1;

//Setting the frequency
 (radioDesc->vf).tuner = 0;
```


(C) Copyright Nokia 2014. All rights reserved.

```
 (radioDesc->vf).frequency = (freq*FREQ_FRAC);

//Send the frequency to the radio device
 res = ioctl(radioDesc->radio_fd, VIDIOC_S_FREQUENCY, &(radioDesc->vf));
 if(res > 0)
  radioDesc->freq = freq;

 return res;
}
```

After that, your radio device should be running. The following main() entry shows this procedure.

```
int main()
{
//Creating our handler struct
 RadioDescriptor* desc  = (RadioDescriptor *) malloc(sizeof(RadioDescriptor));
//Open the device
 radio_open(desc);
 radio_unmute(desc);
//Setting the radio frequency in Hz (104.1 MHz)
 radio_setfreq(desc, 104100000);
 radio_setvolume(desc, 50);
//Sleeping the application for while just to play the radio for 10 seconds
 sleep(10);
//Mute the mixer device
 radio_mute(desc);
//Close the radio device
 radio_close(desc);

 return 0;
}
```

**IMPORTANT:** It is necessary to hold your application for while (10 seconds for example), that way before leaving the application the radio device will be closed. If you do not make this control in your application, after open the device it will remain opened (and working) and the file descriptor will be lost, then it will be not possible to stop it.

## Example source code

- The modified application can be found here: radioAPI.zip

## External links

- Video4Linux wiki
- Video4Linux resources


(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.