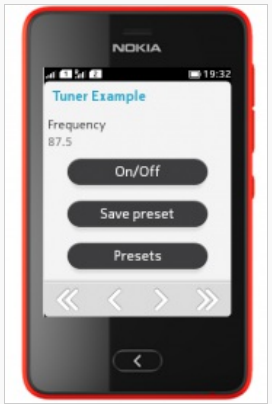
Implementing an AM/FM radio in Java ME

This article explains how to use the AMMS API to implement an AM/FM tuner on the Asha developer platform.

28 Jul 2013

Introduction

The Asha software platform introduces support for new features of the AMMS API, including the ability to access and use the device's AM/FM tuner, as reported on the Java Developer's Library pages.



Java app implementing an FM tuner on the Nokia Asha 501

This article shows how those capabilities can be used to implement a fully functional FM radio, with seek and tune functionality.

Implementing the FM radio

Accessing the device's radio tuner is performed through the TunerControl interface.

Initializing the TunerControl

The first step in the tuner initialization is the creation of the Player that will be responsible for the audio stream management. The creation is performed by using the `capture://radio` locator, and the `Player`'s creation is then finalized via its realize method.

```
Player tunerPlayer = Manager.createPlayer("capture://radio");
tunerPlayer.realize();
```

Once the `Player` instance is available, it is used to get access to the TunerControl via the standard getControl method, called with the `"javax.microedition.amms.control.tuner.TunerControl"` string argument.

```
TunerControl tunerControl = (TunerControl)tunerPlayer.getControl("javax.microedition.amms.control.tuner.TunerControl");
```

Starting and stopping the radio

Being controlled by a standard *MMAPI* `Player` object, the radio tuner can be controlled via its start and stop methods.

Setting modulation and frequency

The `TunerControl` interface defines two constants for the available tuner modulations:

- MODULATION_AM
- MODULATION_FM

Note: The Nokia 501 has a FM radio tuner, so the AM modulation is not available. Using the `MODULATION_AM` in any of the following methods causes an `IllegalArgumentException` to be raised.

For each of those modulations, it is possible to retrieve the minimum and maximum available frequencies via the `TunerControl`'s getMinFreq and getMaxFreq methods]:

```
int minFrequency = tunerControl.getMinFreq(TunerControl.MODULATION_FM);
int maxFrequency = tunerControl.getMaxFreq(TunerControl.MODULATION_FM);
```

Frequencies are handled by `TunerControl` in 100 Hertz, while FM frequencies are usually represented in MHz. Conversion from one format to the other can

(C) Copyright Nokia 2013. All rights reserved.

be performed through the following functions:

```
private double hundredHertzToMHz(int hundredHertz)
{
 return ((double)hundredHertz) / 10000;
}
private int mHzToHundredHertz(double mhz)
{
 return (int)(mhz * 10000);
}
```

By using the above information, it is possible to set a modulation and a frequency, picked in the allowed range, via the `TunerControl`'s setFrequency method, as shown in the following code snippet, that sets the FM modulation to its minimum available frequency:

```
tunerControl.setFrequency(tunerControl.getMinFreq(TunerControl.MODULATION_FM), TunerControl.MODULATION_FM);
```

Automatic and manual frequency tuning

Automatic radio tuning can be performed via the `TunerControl`'s seek method, that scans the frequency range (in upward on downward direction) for the specified modulation and starting from the frequency passed as argument, looking for a valid signal. The new frequency, or the zero value if no frequency is found, is then returned by the method.

Being a potentially time-consuming operation, the `seek` method must not be called in the main application thread. An example of how this operation can be performed in a separate `Thread` is shown in the following code snippet, that seeks upwards the next available signal starting from the frequency currently set on the `TunerControl`.

```
new Thread(new Runnable() {
 public void run()
 {
  try
  {
   tunerControl.seek(tunerControl.getFrequency(), tunerControl.getModulation(), true);
  }
  catch(Exception e)
  {
   // handle the raised Exception
  }
 }
}
}).start();
```

Manual frequency tuning can be perfomed via the `setFrequency` method introduced in the previous section. When manually tuning, the current tuner frequency is incremented or decremented by a constant value, typically 0.05/0.1MHz or a value that is equal or lower to the minimum frequency separation between two separate signals. More information about the topic iis available here: FM broadcast bands.

The following code snippet shows how frequency can be manually incremented/decremented, with a check to set frequencies in the allowed range:

```
private void moveFrequency(boolean upwards)
{
 int newFrequency = tunerControl.getFrequency() + mHzToHundredHertz(0.05) * (upwards ? 1 : -1);

 if(newFrequency >= minFrequency && newFrequency <= maxFrequency)
 {
  tunerControl.setFrequency(newFrequency, tunerControl.getModulation());
 }
}
```

Working with presets

The `TunerControl` interface provides support for managing the device radio presets, so that those can be retrieved and modified by a Java app.

Number of available preset slots can be retrieved via getNumberOfPresets, that on the Asha 501 returns nine (9).

Details of a preset can be retrieved by using its preset number (the presets list is 1-based, so the first preset is associated with number 1, the second with number 2, and so on) with the following methods:

- getPresetModulation returns the preset modulation
- getPresetFrequency returns the preset frequency
- getPresetStereoMode returns the preset stereo mode
- getPresetName returns the preset name

Using the above methods, a PopupList, from the Nokia UI API can be populated and used to allow the user to pick one of the available presets as follows:

```
PopupList popup = new PopupList();

for(int i = 1, num = tunerControl.getNumberOfPresets(); i <= num; i++)
{
 String modulation = tunerControl.getPresetModulation(i);
 int frequency = tunerControl.getPresetFrequency(i);

 popup.appendItem(new PopupListItem(i, "(" + i + ") " + modulation + " - " + frequency));
}
popup.setVisible(true);
```


(C) Copyright Nokia 2013. All rights reserved.

List of available radio presets

Similarly, a Java app can allow the user to modify a preset via one of the two available setPreset methods:

```
// sets the preset using the current modulation, frequency and stereo mode
tunerControl.setPreset(presetIndex);

// explicitly set the preset modulation, frequency and stereo mode
tunerControl.setPreset(presetIndex, frequency, modulation, stereoMode);
```

Note: Being a potentially time-consuming operation, the *setPreset* method must not be performed in the main application thread.

Using presets to configure the radio modulation and frequency can be easily performed by using the setPreset method:

```
tunerControl.usePreset(presetIndex);
```

Controlling the output device

The AudioOutputControl interface, provided via Nokia-specific MMAPI extensions, allows a Java app to control and modify the device used to reproduce audio streams.

A reference to the `AudioOutputControl` can be retrieved as follows:

```
AudioOutputControli audioControl = (AudioOutputControl) GlobalManager.getControl("com.nokia.mid.media.AudioOutputControl");
```

Setting the audio device can be performed via the setOutputMode method, by using one of the available audio mode constants defined by the `AudioOutputControl` interface, including the following:

- AudioOutputControl.PUBLIC: routes audio to loudspeaker, wireless(BT) speakers or TV output
- AudioOutputControl.PRIVATE: routes audio to earpiece or wired / wireless(BT) headset

```
audioControl.setOutputMode(AudioOutputControl.PUBLIC);
```

Note: Modifying the audio output device requires the Java app to be signed by the manufacturer. For this reason, tests performed with a sample unsigned Java app containing the above code fails on both the Nokia 501 emulator and device.

Further considerations

Simulation

FM radio cannot currently be simulated on the *Nokia Asha 1.0 simulator*, due to the lack of an integrated FM tuner: for this reason, testing must be performed on a real device, such as the *Nokia Asha 501*.

Supported modulations

Modulations supported by the device's radio tuner can be retrieved at runtime via the *System* property `tuner.modulations`. On the *Nokia Asha 501*, the returned value is `"fm"`.

RDS Control

The RDSControl is currently not supported by the Asha software platform, as reported here: AMMS API excluded features.

Usage scenarios

Controlling the FM radio from a Java app can be useful in various kind of apps and usage scenarios, including:

(C) Copyright Nokia 2013. All rights reserved.

- building an enhanced FM radio that retrieves and displays names and information about FM radio stations from a remote source
- allowing the user to manage more than the default nine presets, possibly creating different preset groups for different genres
- creating an FM radio alarm clock using PushRegistry
- building a photo viewer integrated with background audio taken from an FM radio station

Summary

This article explores the possibilities and capabilities offered by the javax.microedition.amms.control.tuner package of the AMMS API.

Source code of the Java app illustrated in this article is available here: Media:WikiTunerExample.zip.

Full JavaDocs is available here.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.