Creating a MIDI Application

Published 2014-10-27 | (Compatible with SDK 4.5,5.0,5.1 and 2013,2014 models)

> This tutorial describes the usage of MIDI API of the Hardware API. With the MIDI API an application can interact with MIDI controllers connected to a TV over USB, play standard MIDI files, and use simulated MIDI controllers.

Contents

The application being developed in this tutorial demonstrates the development of an application that uses MIDI controller devices. The application accesses a MIDI controller device, plays a MIDI file, and has its own simulated MIDI keyboard.

Programming of applications using MIDI class is based on the MIDI API of the Hardware API.

Development Environment

Use Samsung Smart TV SDK(4.0 and above) to develop the application. The emulator provided along with the SDK can be used to debug and test the application before deploying it onto a TV. Refer Testing Your Application on a TV. Note that applications may perform better on the TV than on the emulator.

You can find general instructions for creating applications in Implementing Your Application Code.

Prerequisites

To develop TV applications, you need:

a Samsung TV connected to the Internet

SDK **(Samsung Smart TV SDK 4.0 and above is must)** or a text editor for creating HTML, JavaScript and CSS files

Source Files

> Note
>
> > The files needed for the sample application are here.
>
> The tutorial MIDI Application is located under Tutorial_MIDI. Source files in directory are explained in the table:

Directory	Description
CSS	Contains the StyleSheet file mycss.css
Midi	Contains a test midi file test.mid
JavaScript	Contains the JavaScript file main.js which does the following: initializes module registers events key handling displays the results
Resources	Contains the image folder which contains common images

Starting the MIDI Application

To start the MIDI application,

1. Select **Open App** on the menu and select MIDI_Tutorial.

   The following display on the emulator indicates that the application has been selected:

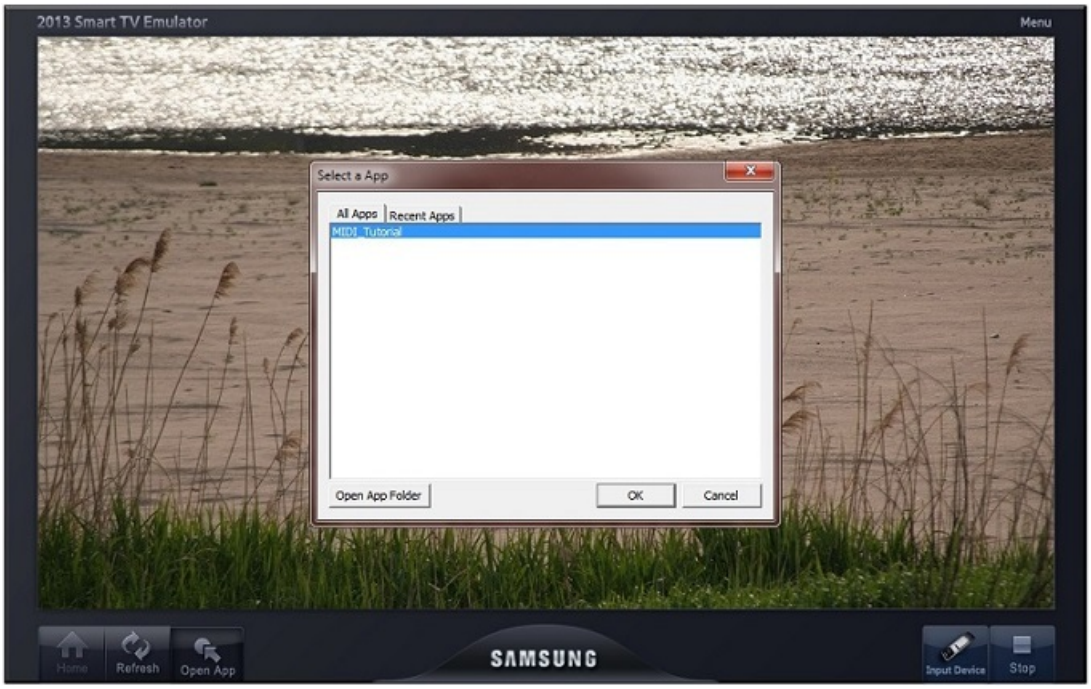

   Figure: Selecting the MIDI Tutorial application.

2. Select MIDI_Tutorial and press **OK**. If the following display appears on the emulator, it means that the application has started successfully:

Figure: MIDI application started

3. Connect a USB MIDI keyboard to the TV.

   Connected MIDI keyboard name is listed on the left side in MIDI Controller Devices list.

Figure: Connected MIDI

Using the MIDI

Following sections explain how to use the connected MIDI controller.

1. If you want to enable and play the connected MIDI keyboard, bring focus to the listed MIDI keyboard and press the **Enter** key on the remote control.

   After pressing the enter key, you can see the configure screen enabled as shown on the figure below.

Figure: Activate the MIDI device

Now pressing any key on the MIDI keyboard will play the corresponding note.

If you want to stop and disable the connected MIDI keyboard, just press the **Enter** key on the remote control again.

2. Notes can be directly heard by pressing keys on the software simulated keyboard as shown on the figure below.

Figure: Play the MIDI keyboard

3. Standard MIDI file can be played simply by pressing the **Play** (►) key of the remote control.

Pressing the **Stop** (■) key on the remote control will stop the MIDI file playback, as on the figure below.

Figure: Play MIDI file

4. You can set the channel, channel instruments, volume parameter and so on as shown on the figure below.

   Below figure shows the display change on screen:

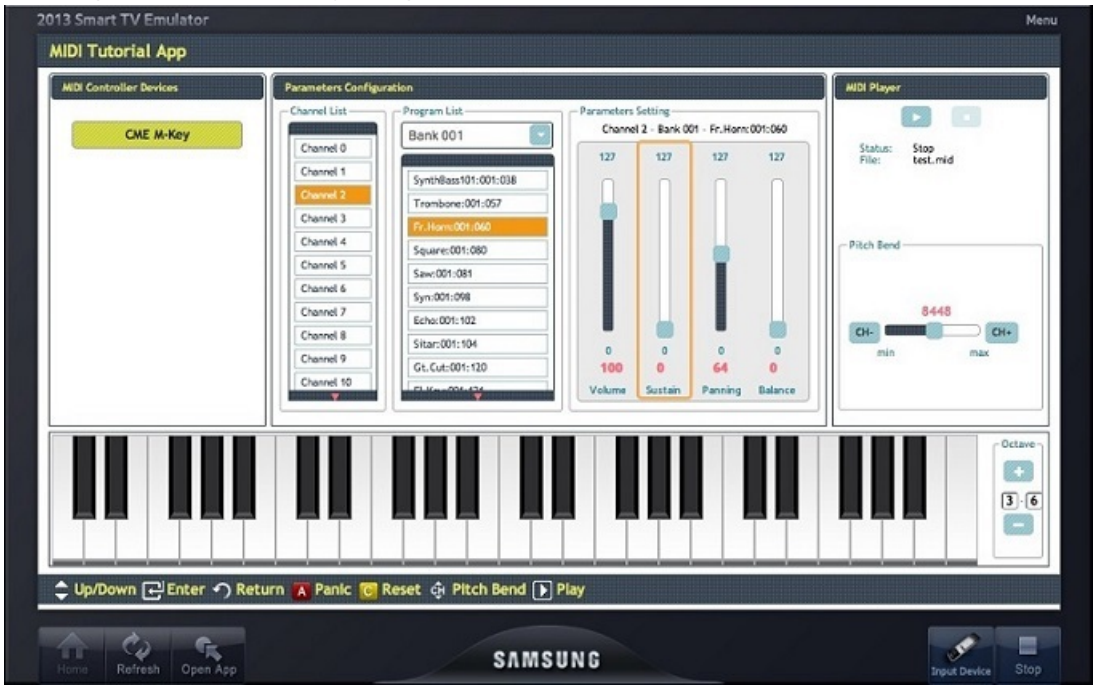

Figure: Set synthesizer parameters

You can navigate across the parameters by pressing **left / right key** key of the remote control on each item.

Pressing **up** and **down** keys will change values of respective parameters.

Getting connected MIDI devices

If MIDI keyboards are connected, the window.webapis.mididevice.getMIDIDevices() function will return each MIDI device instance related to each of the connected MIDI keyboards.

Function	Description
getMIDIDevices	Get MIDI keyboards device instances connected to TV. When it is called for the first time, it will initialize all the internal modules.

```
MidiWidget.onLoad = function () {
  ...
  midi.getMIDIDevices(MidiWidget.onCustomObtained);
};

MidiWidget.onCustomObtained = function (midis) {

};
```

Registering a Callback Function for a Device Connection Event

To receive a device connection event, register a callback function with window.webapis.mididevice.registerManagerCallback(). Once the callback registration is done, callback functions are called on connecting or disconnecting a MIDI controller devices such as MIDI keyboards. The callback function can get window.webapis.mididevice.ManagerEvent class object including event type and MIDI name as an input parameter.

Function	Description
registerManagerCallback	Register callback function to get the event for CONNECT/DISCONNECT.

```
var midi = window.webapis.mididevice || {};
MidiWidget.onLoad = function () {
  gWidgetAPI = new Common.API.Widget();
  gTVKey = new Common.API.TVKeyValue();
  gWidgetAPI.sendReadyEvent();
  ...
  midi.registerManagerCallback(MidiWidget.onDeviceStatusChange);
  midi.getMIDIDevices(MidiWidget.onCustomObtained);
};
MidiWidget.onCustomObtained = function (midis) {
  ...
  gInitSuccess = true;
};
MidiWidget.onDeviceStatusChange = function (sParam) {
  if (sParam.deviceType == midi.MIDI_DEVICE_SYNTHESIZER) {
    ...
  } else {
    switch(Number(sParam.eventType)){
      case midi.MGR_EVENT_DEV_CONNECT:
        ...
        break;

      case midi.MGR_EVENT_DEV_DISCONNECT:
        ...
        break;

      defaut:
        break;
    }
  }
}
```

play or stop playing MIDI

The MIDI module can play the MIDI keyboard or the standard MIDI file by calling the midi.startStream function. To stop playing the MIDI keyboard or the MIDI file, call the midi.stopStream function.

Function	Description

Function	Description
startStream	Acitvate the MIDI keyboard to play notes on key press or begin to play MIDI file.
stopStream	Deactivate the MIDI keyboard from playing the notes on key press or stop playing the MIDI file.

```
var sourceDeviceInfo = new midi.MIDIDeviceInfo();
sourceDeviceInfo.deviceName = o.device.getName();
sourceDeviceInfo.deviceID = o.device.getDeviceID();
sourceDeviceInfo.deviceType = o.device.getType();

var destDeviceInfo = new midi.MIDIDeviceInfo();
destDeviceInfo.deviceName = midisynthInstance.getName();
destDeviceInfo.deviceID = midisynthInstance.getDeviceID();
destDeviceInfo.deviceType = midi.MIDI_DEVICE_SYNTHESIZER;

midi.startStream(sourceDeviceInfo, destDeviceInfo);

var sourceDeviceInfo = new midi.MIDIDeviceInfo();
sourceDeviceInfo.deviceName = o.device.getName();
sourceDeviceInfo.deviceID = o.device.getDeviceID();
sourceDeviceInfo.deviceType = o.device.getType();

var destDeviceInfo = new midi.MIDIDeviceInfo();
destDeviceInfo.deviceName = midisynthInstance.getName();
destDeviceInfo.deviceID = midisynthInstance.getDeviceID();
destDeviceInfo.deviceType = midi.MIDI_DEVICE_SYNTHESIZER;

midi.stopStream(sourceDeviceInfo, destDeviceInfo);
```

Get file Play status

MIDI file playback status can be obtained by below API

Function	Description
getFilePlayStatus	Provides file playback status.

```
ret =  midi.getFilePlayStatus();
if (midi.MIDI_STREAM_STATUS_BUSY != ret) {
    ...
}
```

Set or get synthesizer parameter

MIDI supports getting or setting MIDI parameters for each of the channels. For example, you can set or get the volume of the MIDI channel

Function	Description
sendMessage	Sends a message from DTV widget to MIDI device. This is basically used to implement software MIDI controller.
getInformation	Used for getting MIDI parameters such as volume, pitch bend, program, velocity etc.. for a specified MIDI channel.

```
var evType = midi.MIDI_EVENT_CONTROL_CHANGE;
var evMsg = new midi.MIDIControlMessage();
evMsg.channel = MidiUI.channelList.activeItem.value;
evMsg.parameter = midi.MIDI_CC_BANK_SELECT_LSB;
evMsg.value = Number(i.value.bankNum);
var ret = midisynthInstance.sendMessage(evType, evMsg);

var evType = midi.MIDI_EVENT_PROGRAM_CHANGE;
var evMsg = new midi.MIDIControlMessage();
evMsg.channel = MidiUI.channelList.activeItem.value;
evMsg.parameter = 44;
evMsg.value = Number(i.value.programNum);
var ret = midisynthInstance.sendMessage(evType, evMsg);

var getChannelVolume = function(i){
   var evType = midi.MIDI_EVENT_CONTROL_CHANGE;
   var evMsg = new midi.MIDIControlMessage();
   evMsg.channel = i;
   evMsg.parameter = midi.MIDI_CC_CHANNEL_VOLUME_MSB;
   evMsg.value = 00;
   return midisynthInstance.getInformation(evType, evMsg);
};
```

> Warning
>
> This module is only supported on Samsung Smart TV models for 2013 and later. Samsung SDK 4.0 or higher could be used for development of MIDI applications.