

For AGNPRO miniBox SD250-N and SD150 Media Players

Touch Screen Content Authoring Guide

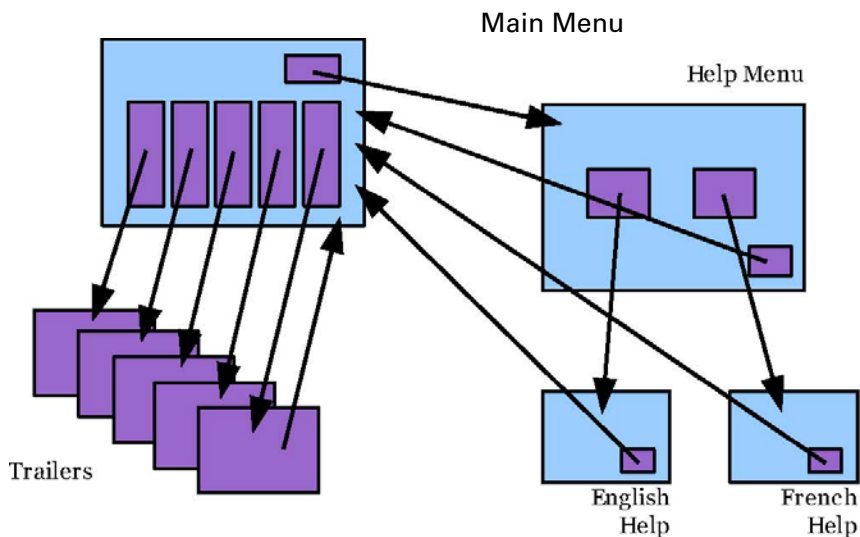
Introduction

The SD250-N and SD150 Media Players are equipped with an RS-232 port as well as drivers to support touch panels. The media player can be connected to touch screens to drive interactive kiosk applications.

Sample Application

Throughout this document, we will use a “Movie Preview” kiosk application for illustration. The preview consists of an animated menu which shows five “movie buttons” and one “help button.” Each movie button plays a movie trailer when touched. The help button shows a language selection page, offering instructions in the “English” and “French” languages. Each selection offers one page of help information and allows an option of jumping back to the “Menu.”

To summarize, the sample includes a number of touch screen layouts.



Purple areas mark touch-sensitive regions of the screen.

Media Files

The first step towards creating interactive contents to play on the media player is to create the media files. The SD250-N and AD150 are capable of playing back MPEG-1, MPEG-2, MPEG-4 (DivX) and JPEG media files. All interactive content on the media player consists of

motion video and still images. All dynamic responses need be emulated using video.

In the Sample Application, we will use the following MPEG-4 DivX (Windows AVI file format) to carry out the desired interactivity:

- 1 Main menu (MENU.AVI)
- 2 Trailers (MOVIE01.AVI, MOVIE02.AVI, MOVIE03.AVI, MOVIE04.AVI, and MOVIE05.AVI)
- 3 Help language selector (HELPLANG.AVI)
- 4 Help in English (HELPENG.AVI)
- 5 Help in French (HELPFRC.AVI)

We are assuming all pages have dynamic contents. In case some pages are static, JPEG files can be used in place of the MPEG-4 AVIs.

Also note that file names do not contain spaces. This is a requirement of the current system.

Media Source Formats

Any MPEG-1, MPEG-2, MPEG-4 (DivX) video and JPEG image files can be used to create the interactive application. However, there are times when some of the contents are created in other animation file formats, such as the Macromedia Flash SWF files.

While there are a number of tools available to convert files to a compatible format, we provide a free Windows XP transcoding utility called Adfotain Transcoder LE for the purpose of transcoding video files into MPEG-4 (generated as .DSA files, which are also compatible).

Basic Touch Programming

Once the media files are prepared, they must be linked using a playlist file. The media player supports this using a file named "PLAYLIST" located in the root directory of the onboard storage.

The PLAYLIST file is an ASCII text file defining multiple "play lists." Each play list can be defined to start at a specified time. While a play list is in action, a number of "events" may be defined to interrupt the playback and skip to another play list. Note that the file must be named PLAYLIST in all caps with no file extension.

Touch regions are defined by rectangle coordinates with the upper left hand corner defined as (0,0) and the lower right hand corner as (640,480). When a touch event is detected within the rectangles, the system triggers a new play list to start.

Also note that touch events are independent of the media file playing. That is, while one uses the PLAYLIST file to define touchable regions of the screen, the actual "button" or animated hint that guides the user to touch that part of the screen must be rendered in the media file. One cannot define button images or animations using PLAYLIST files.

Sample Scripts

Our sample application requires defining the following rectangles:

Main menu (MENU.AVI)

- a) Movie 1 button: (30,150)-(130,450)
- b) Movie 2 button: (150,150)-(250,450)
- c) Movie 3 button: (270,150)-(370,450)
- d) Movie 4 button: (390,150)-(490,450)
- e) Movie 5 button: (510,150)-(610,450)
- f) Help button: (450,60)-(610,120)

Trailers (MOVIE01.AVI, MOVIE02.AVI, MOVIE03.AVI, MOVIE04.AVI, and MOVIE05.AVI)

- a) Full screen return home: (0,0)-(640,480)

We define the rest of the system in the next section.

The first portion of the sample PLAYLIST file looks like the following:

```
[PL1:start]
@(RECT:30,150,130,450)^[PL2]
@(RECT:150,150,250,450)^[PL3]
@(RECT:270,150,370,450)^[PL4]
@(RECT:390,150,490,450)^[PL5]
@(RECT:510,150,610,450)^[PL6]
@(RECT:450,60,610,120)^[PL7]
```

```
<MENU.AVI>
^[PL1]
```

The above script defines PL1 as a playlist that starts as soon as the media player is powered up. Then there are 6 lines each defining a rectangle corresponding to an area of the screen.

When touched, PL1 will be interrupted and PL2 to PL7, respectively, will kick in to play. Then we specify that PL1 consists of just video file MENU.AVI. The last line specifies that at the end of PL1, we loop back to the start of PL1 and play MENU.AVI again.

As a note, the @... lines specify events as play list flow controls and do not define media files to play, whereas the <...> line specifies the actual files to play. Spacing is optional and purely for enhancing readability.

Following PL1, we define PL2 to PL6 which are the movie trailers:

```
[PL2]

@(RECT:0,0,640,480)^[PL1]
    <MOVIE01.AVI>
    ^[PL1]
[PL3]

@(RECT:0,0,640,480)^[PL1]
```

```

        <MOVIE02.AVI>
        ^[PL1]
    [PL4]

    @(RECT:0,0,640,480)^[PL1]
        <MOVIE03.AVI>
        ^[PL1]
    [PL5]

    @(RECT:0,0,640,480)^[PL1]
        <MOVIE04.AVI>
        ^[PL1]

    [PL6]

    @(RECT:0,0,640,480)^[PL1]
        <MOVIE05.AVI>
        ^[PL1]

```

The main feature to note is that at the end of each PL2 to PL6, PL1 is resumed. Also when any part of the screen is touched while the video is playing, PL1 is also played to instead of the currently playing file.

With the exception of the help menus, the play list above creates a working movie preview kiosk application (with PL7 not yet defined). In the next section, we will define the help menus as a hierarchical structure.

Hierarchical Menu System

The PLAYLIST scripting system in the media player allows complex menu graphs, not just lists and trees, to be defined. Our sample application includes a help system that first asks the user for language selection before displaying the help content. Continuing the example in the previous section, the remainder of the menu system looks like the following:

Help language selector (HELPLANG.AVI)

- a) English selector button: (200,40)-(280,300)
- b) French selector button: (200,340)-(280,600)
- c) Return home button: (450,360)-(610,420)

Help in English (HELPENG.AVI)

- a) Return home button: (450,360)-(610,420)

Help in French (HELPFRC.AVI)

- a) Return home button: (450,360)-(610,420)

Continuing our previous PLAYLIST file, we write the following scripts:

```
[PL7]
```

```

@(RECT:200,40,280,300)^[PL8]

@(RECT:200,340,280,600)^[PL9]

@(RECT:450,360,610,420)^[PL1]
  <HELPLANG.AVI>

  ^[PL7]

```

PL7 has three defined rectangular regions, used for linking English and French help, and linking back to the main menu, respectively. When the HELPLANG.AVI finished playing, PL7 is looped again. In other words, PL7 loops indefinitely until someone touches one of the rectangular regions.

```

[PL8]

@(RECT:450,360,610,420)^[PL1]
  <HELPENG.AVI>
  ^[PL8]

```

```

[PL9]

@(RECT:450,360,610,420)^[PL1]
  <HELPFRC.AVI>
  <HELPFRC.AVI>
  <HELPFRC.AVI>
  ^[PL1]

```

PL8 and PL9 each shows help instructions in English and French, respectively. Here we show different implementation options between the two languages. PL8 loops indefinitely until someone touches the "Return home" button, at which point it goes back to the menu page. PL9, however, plays the French help video 3 times, then it goes back to the main menu. This feature allows creating kiosk applications that always go back to the intro screen when the user walks away in the middle of a selection.

Full Sample PLAYLIST Code

The complete PLAYLIST code for the sample application is included below for reference.

```

[PL1:start]
@(RECT:30,150,130,450)^[PL2]
@(RECT:150,150,250,450)^[PL3]
@(RECT:270,150,370,450)^[PL4]
@(RECT:390,150,490,450)^[PL5]
@(RECT:510,150,610,450)^[PL6]
@(RECT:450,60,610,120)^[PL7]
  <MENU.AVI>

```

^[PL1]
[PL2]

@(RECT:0,0,640,480)^[PL1]
<MOVIE01.AVI>
^[PL1]

[PL3]

@(RECT:0,0,640,480)^[PL1]
<MOVIE02.AVI>
^[PL1]

[PL4]

@(RECT:0,0,640,480)^[PL1]
<MOVIE03.AVI>
^[PL1]

[PL5]

@(RECT:0,0,640,480)^[PL1]
<MOVIE04.AVI>
^[PL1]

[PL6]

@(RECT:0,0,640,480)^[PL1]
<MOVIE05.AVI>
^[PL1]

[PL7]

@(RECT:200,40,280,300)^[PL8]
@(RECT:200,340,280,600)^[PL9]
@(RECT:450,360,610,420)^[PL1]

<HELPLANG.AVI>
^[PL7]

[PL8]

@(RECT:450,360,610,420)^[PL1]
<HELPENG.AVI>
^[PL8]

[PL9]

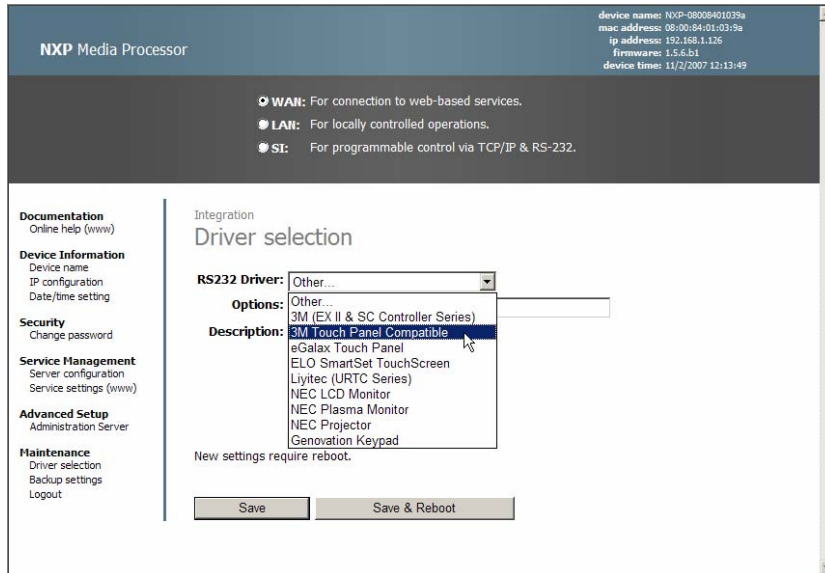
@(RECT:450,360,610,420)^[PL1]
<HELPFRC.AVI>
<HELPFRC.AVI>
<HELPFRC.AVI>
^[PL1]

Setting the Touchscreen Driver

miniBox SD250-N and SD150 models interface with 3M Microtouch, ELO and eGalax touch screen controllers via the built-in serial port (touch is not supported via USB).

Model SD250-N

To set the touch screen driver, log into the unit with a web browser (see N-series User Manual for additional information on accessing browser-based configuration tools). Next, choose “Driver Selection” from the left menu, and select the appropriate driver from the list, then click “Save and Reboot”.



Model SD150

With the SD150 model media player, touch screen driver selection is accomplished via the playlist, since the SD150 lacks a browser-based interface for driver selection.

To select the correct touch screen driver within a playlist, insert a tag on the **first line** of the playlist, [OPTIONS] then insert a <<driver>> line indicating the driver that is needed for your touchscreen model.

For 3M Touch Screens:

```
[OPTIONS]  
<<driver:touchpanel=3M,640x480r, ttyS0,d>>
```

For 3M Compatible Screens:

```
[OPTIONS]  
<<driver:touchpanel=3M,640x480r, ttyS0,ds>>
```

For eGalax Touch Screens:

```
[OPTIONS]  
<<driver:touchpanel=eGalax,640x480r, ttyS0,d>>
```

For ELO SmartSet Touch Screens:

```
[OPTIONS]  
<<driver:touchpanel=ELOSmartSet,640x480r, ttyS0,d>>
```

Remote Media Updating

Our Central Management Server allows you to remotely update media and playlists via the “manual programming” feature.

Technical Specifications

Maximum number of play lists (PLnnn): 256

Maximum number of play lists with scheduled start time: 64

Maximum number of play list instructions: 2048 (in all)

Maximum number of files on storage: 512 (in all)

Maximum number of triggering events: 512 (in all)

Appendix: Transcoding Files

The Adfotain Transcoder LE takes a number of file formats as source and produces DSA files which are MPEG-4 compatible. The accepted source files include:

- SWF (Macromedia Flash) files
- PPT (Microsoft PowerPoint) files
- AVI/WMV (Microsoft Windows Media) files 3

Adfotain Transcoder XE, available for \$199, adds these file formats

- MOV (Apple QuickTime) files
- VOB (DVD video) files

Of the file formats mentioned, Adfotain Transcoder performs best when the source file is an AVI/WMV file. Transcoding from AVI/WMV keeps the maximum frame rate possible and the best sound quality available from the source media file. If your authoring tool is able to output multiple file formats from the list above, the most preferred format is AVI or WMV files.

Quality Adjustments

Adjusting video quality during media transcoding is a subject that requires some experience and extensive trial and error. This section does not intend to cover all issues related to the subject area but tries to provide basic information to help getting started.

A number of key parameters control the outcome of each file conversion.

- 1 Source (capture) quality
- 2 Target bit rate
- 3 Target frame rate
- 4 I-frame frequency

Source quality is the quality of the AVI/WMV file to be used for transcoding. When producing the file using your authoring tool, select the highest quality possible.

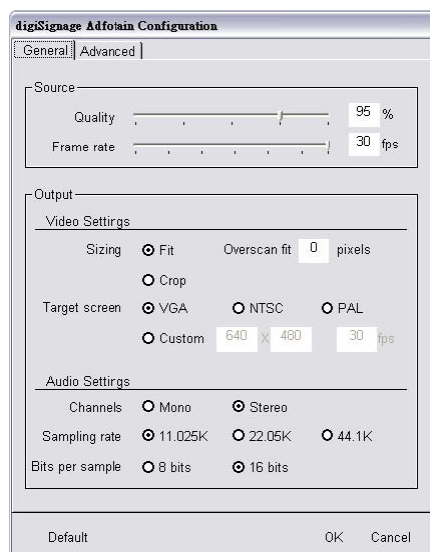
Target bit rate decides the size of the transcoded file. A typical good quality standard-definition MPEG-4 video runs at 4 Mbps (mega-bits per second), which translates to 30 MB (megabytes) per minute of video. It is common to apply target bit rate in the range of 1 to 8 Mbps, which translates to 7.5 to 60 MB per minute of video. Higher bit rates allow more video information to be encoded in the target media file.

Target frame rate dictates the number of distinct video frames shown per second of video. The more frames allowed, the smoother the video appears on screen. Typical DVD content is encoded at 24 fps (frames per second).

MPEG-4 frames are encoded in one of three formats: the I-, P-, and B-frame. I-frames encode full video details, whereas P-frames and B-frames encode only portions that changed between frames. As a result I-frames are usually much larger than P- and B-frames. I-frames are used to stop error propagation which accumulate when only P- and B-frames are used.

Therefore it is desirable to reduce the number of I-frames as long as accumulated error is visually acceptable.

Adfotain Transcoder XE Configuration Screen



Adjusting these parameters usually require multiple iterations. How long it takes to find the optimal combination depends on media contents and user's expertise.