

# **Product Technology Partners**

## **MSSR300 ActiveX Control: Programming Guide**

**Project P264**

**Document no. M02090301**

**Issue 3.00.6505.2**

Product  
Technology  
Partners

# Document Control

Client:	Product Technology Partners		
Title:	MSSR300 ActiveX Control: Programming Guide		
Project:	P264		
Document no.:	M02090301	Issue:	3.00.6505.2
File:	mssr300.doc		
Date:	5 May 2005	Revision:	54
Controlled by:	M J Saxon		
Reviewed by:	M J Saxon	Signed:	
Authorised by:	M J Saxon	Signed:	

## Modification Record

New issue no.	Date	Change note/Reason	Affected content
2.10	25 Nov 2003	First Release for 2.10	
2.10	23 Dec 2003	Typo correction	
2.10.5120	20 Mar 2004	Minor update	
2.10.5817.1	17 Oct 2004	Modified AcmFormat property	
3.00.6503.1	03 May 2005	New features (UNC, speed change)	
3.00.6505.2	05 May 2005	New data copy/move methods	

<b>Product  Technology  Partners</b>	Newton House • 147 St Neots Road • Hardwick • Cambridge • CB3 7QJ
	Tel: +44 (0)1954 212806 • Fax: +44 (0)1954 212808
	E-mail: support@ptpart.co.uk

# Contents

<b>1. Description .....</b>	<b>1</b>
<b>1.1 About the MSSR300 Control .....</b>	<b>1</b>
<b>1.2 Configurable display .....</b>	<b>2</b>
<b>1.3 Simple use of the Control.....</b>	<b>2</b>
<b>2. Evaluation version restrictions.....</b>	<b>4</b>
<b>3. Data types and constants.....</b>	<b>5</b>
<b>3.1 Data types.....</b>	<b>5</b>
<b>3.2 Constants .....</b>	<b>5</b>
<b>4. Properties .....</b>	<b>6</b>
<b>5. Methods .....</b>	<b>13</b>
<b>5.1 ChooseAcmFormat.....</b>	<b>15</b>
<b>5.2 ChooseSpecificAcmFormat .....</b>	<b>15</b>
<b>5.3 CopyOrMoveSamples .....</b>	<b>15</b>
<b>5.4 CopyOrMoveTime .....</b>	<b>16</b>
<b>5.5 DeleteSamples .....</b>	<b>16</b>
<b>5.6 DeleteTime.....</b>	<b>16</b>
<b>5.7 GetAcmFormatAsString .....</b>	<b>16</b>
<b>5.8 GetCurrentFormatAsString .....</b>	<b>17</b>
<b>5.9 GetLevels.....</b>	<b>17</b>
<b>5.10 OpenExistingOrNewFile .....</b>	<b>17</b>
<b>5.11 OpenNewFile .....</b>	<b>18</b>
<b>5.12 OpenTemporaryFile .....</b>	<b>19</b>
<b>5.13 Register .....</b>	<b>19</b>
<b>5.14 RePack.....</b>	<b>19</b>
<b>5.15 Reset .....</b>	<b>20</b>
<b>5.16 ShowMixer .....</b>	<b>20</b>

<b>5.17 StartPlayback .....</b>	<b>20</b>
<b>5.18 StartRecording .....</b>	<b>20</b>
<b>5.19 Stop.....</b>	<b>21</b>
<b>6. Events/Callbacks.....</b>	<b>22</b>
<b>6.1 NewLevelValues.....</b>	<b>22</b>
<b>6.2 NewVoxState .....</b>	<b>22</b>
<b>6.3 PlaybackStopped .....</b>	<b>22</b>
<b>6.4 RecordingStopped.....</b>	<b>23</b>
<b>7. Error codes .....</b>	<b>24</b>
<b>8. Acn Format Strings.....</b>	<b>26</b>
<b>9. Files specified by UNC path.....</b>	<b>27</b>

# 1. Description

## 1.1 About the MSSR300 Control

The MSSR300 control is an ActiveX control for sound recording and playback. Recording and playback is to and from disk files, so recording is limited only by the amount of available disk space.

Features of the control include:

- A configurable peak level meter display. This is a simple bar graph display; but the control provides the necessary peak level data for you to draw your own display if you so wish.
- Support for recording and playback of PCM WAV files, mono or stereo, with 8 or 16 bit samples, and with sampling rates from 192000 down to 6000 samples per second. Recording and playback of 8-bit A-law and  $\mu$ -law encoded files is also directly supported. MSSR300 will also support any format for which an ACM codec<sup>1</sup> is installed and which can be recorded and replayed in real time.
- Support for recording and playback of MP3 files.
- Support for recording and playback of other raw binary data files of known format.
- The ability to insert tone markers each time recording is started and/or finished.
- The ability to set a maximum recording duration.
- The ability to start recording or playback at any point within the current file.
- Recording to temporary files that are deleted when closed, or working with new or existing permanent WAVE files.
- The ability to pause recording or playback at any time
- Event callbacks and state monitoring properties to allow recording or playback progress to be monitored.
- The ability to select specific devices for recording and playback
- Level-controlled recording (VOX, or Voice-Operated Recording)
- The ability to playback at varying speeds
- The ability to access files specified by UNC paths

---

<sup>1</sup> See the Windows help system or MSDN for details of the Windows Audio Compression Manager (ACM).

- 30
- Simple editing functions with the ability to copy, move and delete data within a file.

## 1.2 Configurable display

35 The recorder control provides a configurable display in the form of a bar graph (or two bar graphs, for stereo recording and playback). Each bar graph consists of a number of segments, as follows:

- A high-level or overload segment, which indicates when the recording or playback level is overloading the sound card hardware (peak level greater than 90% of the maximum possible level)
- 40 • Three mid-level or ‘warning’ segments, indicating peak levels greater than  $-9\text{dB}$ ,  $-6\text{dB}$ , and  $-3\text{dB}$  relative to the maximum possible level
- Up to 16 low-level segments, indicating peak level increases in steps of  $6\text{dB}$ .

45 The control property **NumSegments** (see section 4) is used to set the total number of segments. The **BackColor**, **LowColor**, **MidColor** and **HighColor** properties are used to set the colors of the control background, and low-level, mid-level and high-level segments respectively. The **Orientation** property is used to set the orientation of the bar graph display. The **Style** property allows you to set rectangular or elliptical bar graph segments, with optional 3D effect and ‘shadow’.

## 1.3 Simple use of the Control

At application design time:

- 50
- Use the design-time properties listed in section 4 to set the control properties the way you want them.

At application run time:

- 55 • First, if you have purchased a licence for the MSSR300 control (a licence for QuickRecord will also work), use the **Register** method to fully enable the control.
- Use the **Reset** method to reset the control, preferably before the first time it is displayed.
- 60 • Use the **OpenTemporaryFile**, **OpenNewFile**, or **OpenExistingOrNewFile** methods to open an appropriate file for recording or playback.
- You can use the **CurrentTimePosition** or **CurrentSamplePosition** properties in conjunction with the **TotalTime** or **NumSamples** properties to set the point at which you wish to start recording or playback.
- 65 • Use the **StartRecording** or **StartPlayback** methods to start recording or playback, respectively.

- You can use the **Paused** property at any time to pause or resume recording or playback.
- Use the **Stop** method to stop recording or playback, if required.
- 70 • You can change the **TemporaryFile** property at any time a file is open, if (for example) you are working with a temporary file and you decide you want to keep it.
- Use the **Reset** method to close the current file and return the control to idle mode.

75 The QuickRecord application, supplied with the MSSR300 control, is an example of the use of the control and demonstrates many of the control's features in a transparent way.

## 2. Evaluation version restrictions

The MSSR300 control is supplied with the QuickRecord application as a 30-day evaluation package.

80 For the first 20 days of the evaluation period, use of the control is unencumbered. For the last 10 days of the evaluation period, a dialog is displayed the first time that the **Reset** method is called that will advise you of the evaluation time remaining. After the end of the evaluation period, most method calls on the control will result in an error code 99 being returned, meaning that the evaluation period has expired and the control cannot be used.

85 If you wish to continue using the control after the end of the 30-day evaluation period, you must purchase a key for the use of both QuickRecord and the MSSR300 control. As described in section 1.3 above, you must then use the **Register** method in your application to provide the control with the key each time an instance of the control is created.

90 For more information regarding purchase of a key, point your browser at <http://www.ptpart.co.uk/quickrecord/>.

## 3. Data types and constants

### 3.1 Data types

95 Different keywords are used in C/C++, Delphi and Visual Basic to indicate data types. In this document, the following keywords are used:

SHORT	16-bit integer (C/C++ type <b>short</b> , Delphi type <b>Smallint</b> , VB type <b>Integer</b> )
LONG	32-bit integer (C/C++ type <b>long</b> , Delphi type <b>Longint</b> , VB type <b>Long</b> )
100 SINGLE	32-bit floating-point number (C/C++ type <b>float</b> , Delphi type <b>Single</b> , VB type <b>Single</b> )
DOUBLE	64-bit floating-point number (C/C++ type <b>double</b> , Delphi type <b>Double</b> , VB type <b>Double</b> )
COLOR	32-bit unsigned integer containing a color value (C/C++ type <b>OLECOLOR</b> , Delphi type <b>OLECOLOR</b> , VB type <b>Long</b> )
105 STRING	A text string (C/C++ type typically <b>BSTR</b> , Delphi type <b>String</b> , VB type <b>String</b> )
VARIANT	Variant type (C/C++ type <b>OLEVARIANT</b> , Delphi type <b>OleVariant</b> , VB type <b>Variant</b> (default type))

### 3.2 Constants

110 A number of the control properties take integer values that have specific meanings. For your convenience, we supply three files that define mnemonic symbols for these constants:

- **MSSR300Const.h** for C/C++ programmers
- **MSSR300Const.bas** for Microsoft Visual Basic programmers
- 115 • **MSSR300Const.pas** for Borland Delphi programmers.

# 4. Properties

The following table lists the properties of the MSSR300 Control. All properties may be read at run time. The properties are divided into the following categories:

- Appearance
- 120 • Status Information
- File Format Information
- File Information
- Recording and Playback Device Information
- Recording Options
- 125 • Recording and Playback Options
- Tone Generation
- Voice-Operated Recording (VOX)

Name	Type	Design time R/W	Run time Write	Default	Description
<b>APPEARANCE</b>					
BackColor	COLOR	✓	✓	Button face color	Background color for the control
LowColor	COLOR	✓	✓	Green	Color used for the low-level segments (peak level less than –10dB) in the bar graph displays
MidColor	COLOR	✓	✓	Yellow	Color used for the mid-level segments (peak level between 0 and –10dB) in the bar graph displays
HighColor	COLOR	✓	✓	Red	Color used for the ‘overload’ segment in the bar graph displays
NumSegments	SHORT	✓	✓	8	Total number of segments in each bar of the display. Range 1 (overload segment only displayed) to 20 (overload + 3 mid-level segments + 16 low-level segments)
Orientation	SHORT	✓	✓	0	Orientation of the bar graph displays. Allowed values are: 0 (MSSR_LEFTRIGHT) = left to right, left channel on top bar graph 1 (MSSR_BOTTOMTOP) = bottom to top, left channel on left bar graph 2 (MSSR_RIGHTLEFT) = right to left, left channel on bottom bar graph 3 (MSSR_TOPBOTTOM) = top to bottom, left channel on

Name	Type	Design time R/W	Run time Write	Default	Description
					right bar graph
Style	SHORT	✓	✓	0	<p>Style of bar graph display. Basic values are:</p> <p>0 (MSSR_RECTANGLE) = rectangular segments</p> <p>1 (MSSR_ELLIPSE) = elliptical (round) segments</p> <p>These can be combined with two optional modifier flags as follows:</p> <p>2 (MSSR_3D_EFFECT): The segments are given a 3D effect</p> <p>4 (MSSR_SHADOW): A dark segment is shown for each unlit segment</p>
<b>STATUS INFORMATION</b>					
Status	SHORT				<p>Current status of the control. Possible values are:</p> <p>0 (MSSR_UNINITIALIZED) = control not yet initialised</p> <p>1 (MSSR_IDLE) = control is idle (no file open)</p> <p>2 (MSSR_READY) = control has a file open, is ready for recording or playback</p> <p>3 (MSSR_RECORDING) = control is currently recording</p> <p>4 (MSSR_PLAYBACK) = control is currently in playback</p>
CurrentSamplePosition	LONG		✓		<p>Current recording/playback position, in samples, when Status &gt;= 2. The first sample is sample 0.</p> <p>The sample position when read may not equal the last value written, if an ACM file format is in use.</p>
CurrentTimePosition	DOUBLE		✓		<p>Current recording/playback position, in seconds, when Status &gt;= 2</p> <p>The time position when read may not equal the last value written, if an ACM file format is in use.</p>
LastError	SHORT			0	<p>This property contains the last operating error returned by the control. It is reset to MSSRERROR_NOERROR after it is read. See section 7 for a list of error codes.</p>

Name	Type	Design time R/W	Run time Write	Default	Description
PlayOnly	SHORT		? <sup>2</sup>	0	Controls whether the file to be opened next will be opened for playback only (i.e. read only)
<b>FILE FORMAT INFORMATION</b>					
SampleRate	LONG	✓	?	44100	Sampling rate to be used when creating new wave files, and current sampling rate in use. Allowed values are 192000, 96000, 88200, 64000, 48000, 44100, 32000, 22050, 16000, 11025, 8000 and 6000 samples per second.
BitsPerSample	SHORT	✓	?	16	Format of WAVE file to be used when creating new wave files, and current wave format in use. Allowed values are: 8 (MSSR_PCM8) = 8-bit PCM (low quality) 16 (MSSR_PCM16) = 16-bit PCM (high quality) 106 (MSSR_ALAW) = 8-bit A-law 107 (MSSR_MULAW) = 8-bit $\mu$ -law 100 (MSSR_ACM) = Use the ACM format specified in the AcmFormat property
NumChannels	SHORT	✓	?	2	Number of channels to record. Allowed values are: 1 (MSSR_MONO) = monophonic recording 2 (MSSR_STEREO) = stereophonic recording
AcmFormat	VARIANT		✓	Unset	ACM wave file format. This property is an array of byte of variable size. Actually contains the WAVEFORMATEX-based structure specifying the wave format.  NOTE that this property may be SET (written) by assigning a special text string (as well as by assigning an array of byte). For more information on this text string, see section 8. However, when read, the property is always an array of byte.

<sup>2</sup> Properties with a run-time write indicator of '?' can only be written if the Status is < 2 (i.e. it is MSSRSTATUS\_UNINITIALIZED or MSSRSTATUS\_IDLE).

Name	Type	Design time R/W	Run time Write	Default	Description
DataOnly	SHORT		?	0	<p>Determines type of file when opening a new or existing file.</p> <p>0 (MSSR_FILE_WAV): New files will be WAV files, existing files are assumed to be WAV files.</p> <p>1 (MSSR_FILE_BINARY): New files will be, and existing files will be assumed to be, raw binary data of the type specified by SampleRate, BitsPerSample, NumChannels and AcFormat.</p> <p>2 (MSSR_FILE_CHECK_MP3): New files will be raw binary data of the type specified by SampleRate, BitsPerSample, NumChannels and AcFormat. Existing files are assumed to be MP3 files and will be checked for valid MP3 header information: If found, SampleRate, BitsPerSample, NumChannels and AcFormat are set appropriately.</p> <p>NEGATIVE NUMBER: If DataOnly is negative, then the action is the same as MSSR_FILE_BINARY except that for existing files up to -DataOnly bytes at the start of the file will be ignored (useful for files with fixed-size headers, e.g. NIST format)</p>
<b>FILE INFORMATION</b>					
Filename	STRING				Contains the name of the wave file currently open, when Status >= 2.
TemporaryFile	SHORT		✓		<p>Flag to indicate whether wave file is temporary or permanent. If temporary, the file will be deleted when the Reset method is called or the control is closed. Allowed values are:</p> <p>(MSSR_PERMANENT) = the file should not be deleted</p> <p>(MSSR_TEMPORARY) = the file should be deleted</p>
NumSamples	LONG				Contains the current length of the opened file, in samples, when Status >= 2.
TotalTime	DOUBLE				Contains the current length of the opened file, in seconds, when Status >= 2
UNCUser	STRING		✓	Empty	Contains the user name to use for connections when specifying a file

Name	Type	Design time R/W	Run time Write	Default	Description
				string	by UNC pathname. See section 9 for information on this property.
UNCPassword	STRING		✓	Empty string	Contains the password to use for connections when specifying a file by UNC pathname. See section 9 for information on this property.
<b>RECORDING AND PLAYBACK DEVICE INFORMATION</b>					
NRecordDevices	LONG				Number of recording devices available in the system
RecordDevice	LONG		✓	-1	The recording device to use. Range is -1..NRecordDevices-1. A value of -1 indicates that the default (Windows-allocated) device should be used.
RecordDeviceName(deviceNumber)	STRING				The system name of a specific recording device. The 'deviceNumber' argument can take the values 0..NRecordDevices-1
NPlaybackDevices	LONG				Number of playback devices available in the system
PlaybackDevice	LONG		✓	-1	The playback device to use. Range is -1..NPlaybackDevices-1. A value of -1 indicates that the default (Windows-allocated) device should be used.
PlaybackDeviceName(deviceNumber)	STRING				The system name of a specific playback device. The 'deviceNumber' argument can take the values 0..NPlaybackDevices-1
<b>RECORDING OPTIONS</b>					
RecordLimit	LONG	✓	✓	0	If set to a non-zero value, will limit total recording time on a file to this number of samples (note RecordLimit and RecordTimeLimit are two views on the same data)
RecordTimeLimit	DOUBLE	✓	✓	0.0	If set to a non-zero value, will limit total recording time on a file to this number of seconds (note RecordLimit and RecordTimeLimit are two views on the same data)
Insert	SHORT		✓	0	Set to 1 for recording within an existing file to be inserted within the file rather than overwriting existing data

Name	Type	Design time R/W	Run time Write	Default	Description
<b>RECORDING AND PLAYBACK OPTIONS</b>					
Paused	SHORT		✓	0	Flag to control pausing in both recording and playback. Allowed values are:  0 (MSSR_RUNNING) = recording/playback is not paused  1 (MSSR_PAUSED) = recording/playback is paused
PlaySpeed	SHORT		✓	0	This property controls playback speed. A value of 0 disables speed control (playback is at normal speed). A positive value 1-32767 causes playback to be at 1%-32767% of normal speed. So a value of 100 will cause playback at normal speed, a value of 200 playback at twice speed, a value of 50 playback at half speed. This property may be adjusted during playback, except that it cannot (during playback) be changed from zero to non-zero or vice versa.
PlayQuality	SHORT		✓	1000	This property controls playback sound quality when PlaySpeed is non-zero. PlayQuality cannot be changed whilst playback is in progress. Adjust this property as required to give best sound quality on playback at slow or fast speeds.  A PlayQuality between 100 and 32767 preserves pitch; a PlayQuality of 0 does not preserve pitch
<b>STONE GENERATION</b>					
PreToneDuration	SINGLE	✓	✓	0.0	If this property is set to a non-zero value (in seconds), an 880Hz tone of this duration will be placed in the file during recording, at the start of recording (if not paused), and each time the pause flag is reset.
PostToneDuration	SINGLE	✓	✓	0.0	If this property is set to a non-zero value (in seconds), an 880Hz tone of this duration will be placed in the file during recording, each time the pause flag is set and at the end of recording (if not paused).

Name	Type	Design time R/W	Run time Write	Default	Description
<b>VOICE-OPERATED RECORDING (VOX)</b>					
VoxChannel	SHORT		? <sup>3</sup>	0	The channel used for VOX (Voice-operated recording). Use VOX to skip recording during quiet periods. Allowed values are: 0 (MSSR_VOXOFF) = VOX is disabled 1 (MSSR_VOXLEFT) = Use the left channel for VOX control (or only channel, when NumChannels=1) 2 (MSSR_VOXRIGHT) = Use the right channel for VOX control (or only channel, when NumChannels=1) 3 (MSSR_VOXBOTH) = Use both left and right channels for VOX control (or only channel, when NumChannels=1)
VoxStopLevel	SINGLE		✓	0	Peak sound level for VOX to pause recording (when VOX is activated). Normally negative (maximum level is 0.0dB). Pause in recording occurs when peak sound level is less than <i>VoxStopLevel</i> for a duration more than <i>VoxTime</i>
VoxStartLevel	SINGLE		✓	0	Peak sound level for VOX to restart recording (when VOX is activated). Recording restarts when the peak sound level exceeds <i>VoxStartLevel</i>
VoxTime	SHORT		✓	0	Minimum time (specified in units of 0.1sec) for which peak sound level must not exceed <i>VoxStopLevel</i> in order for a VOX-operated pause to start
VoxPaused	SHORT				Flag indicating whether a VOX-controlled pause in recording is in effect. Zero=not paused, non-zero=paused

<sup>3</sup> Can only be written when status is not MSSRSTATUS\_RECORDING

## 5. Methods

130 The MSSR300 Control supports the following methods. Many of these methods return an ErrorCode, which for no error will be 0 (MSSRERROR\_NOERROR, see section 7).

### Summary of Methods

```
SHORT errorCode = ChooseAcmFormat([in] LONG hwndOwner,  
[in] STRING Title)  
135 SHORT errorCode = ChooseSpecificAcmFormat([in] LONG  
hwndOwner,  
[in] STRING Title,  
[in] SHORT WaveFormat)  
SHORT errorCode = CopyOrMoveSamples([in] LONG startSample,  
140 [in] LONG endSample,  
[in] LONG insertSample,  
[in] SHORT move)  
SHORT errorCode = CopyOrMoveTime([in] DOUBLE startTime,  
[in] DOUBLE endTime,  
145 [in] DOUBLE insertTime,  
[in] SHORT move)  
SHORT errorCode = DeleteSamples([in] LONG nSamples)  
SHORT errorCode = DeleteTime([in] DOUBLE time)  
SHORT errorCode = GetAcmFormatAsString([out] STRING Format)  
150 SHORT errorCode = GetCurrentFormatAsString(  
[out] STRING Format)  
GetLevels([out] SINGLE leftLevel,  
[out] SHORT leftOverload,  
[out] SINGLE rightLevel,  
155 [out] SHORT rightOverload)  
SHORT errorCode = OpenExistingOrNewFile(  
[in] STRING Filename)  
SHORT errorCode = OpenNewFile([in] STRING Filename)  
SHORT errorCode = OpenTemporaryFile()  
160 SHORT errorCode = Register([in] STRING name,  
[in] STRING keycode)  
SHORT errorCode = RePack()  
SHORT errorCode = Reset()  
SHORT errorCode = ShowMixer([in] SHORT mixerType)
```

```
165  SHORT errorCode = StartPlayback()  
      SHORT errorCode = StartRecording()  
                        Stop()
```

## 5.1 ChooseAcmFormat

```
170  SHORT errorCode = ChooseAcmFormat(  
        [in] LONG hwndOwner,  
        [in] STRING Title  
    )
```

175 This method displays the standard Windows dialog to allow an ACM file format to be selected. The **hwndOwner** parameter is the handle of the parent window for the dialog (or zero if there is no parent window). The **Title** parameter is a text string that will be displayed on the title bar of the dialog. If this string is zero length (""), the default dialog title will be displayed.

180 The method returns 0 (MSSRERROR\_NOERROR) if it succeeds, or a non-zero error code otherwise.

If a valid ACM format is correctly selected (i.e. the function succeeds), the **AcmFormat**, **NumChannels** and **SampleRate** properties are updated to reflect the chosen format. Note that the **BitsPerSample** property must be set to 100 (MSSR\_ACM) to use the set format.

## 5.2 ChooseSpecificAcmFormat

```
185  SHORT errorCode = ChooseSpecificAcmFormat(  
        [in] LONG hwndOwner,  
        [in] STRING Title,  
        [in] SHORT WaveFormat  
    )
```

190 This method displays the standard Windows dialog to allow a *specific* ACM file format to be selected. The **hwndOwner** parameter is the handle of the parent window for the dialog (or zero if there is no parent window). The **Title** parameter is a text string that will be displayed on the title bar of the dialog. If this string is zero length (""), the default dialog title will be displayed. The **WaveFormat** parameter is a valid  
195 **WAVE\_FORMAT\_\*** specifier (for example, **WAVE\_FORMAT\_MPEGLAYER3** = 85).

The method returns 0 (MSSRERROR\_NOERROR) if it succeeds, or a non-zero error code otherwise.

200 If a valid ACM format is correctly selected (i.e. the function succeeds), the **AcmFormat**, **NumChannels** and **SampleRate** properties are updated to reflect the chosen format. Note that the **BitsPerSample** property must be set to 100 (MSSR\_ACM) to use the set format.

## 5.3 CopyOrMoveSamples

```
SHORT errorCode = CopyOrMoveSamples([in] LONG startSample,  
        [in] LONG endSample,  
        [in] LONG insertSample,  
205  [in] SHORT move)
```

This method copies the data between sample positions **startSample** and **endSample**, and inserts it at the point defined by **insertSample**. The point defined by **insertSample** may not be between **startSample** and **endSample**. If the **move** parameter is non-zero, the original data between **startSample** and **endSample** is deleted (i.e. the data is effectively moved). If the **move** parameter is zero, the original data is preserved (i.e. the data is copied).

If **move** is non-zero, the data is first copied as if **move** had been set to zero. Then, the **DeleteSamples** and **RePack** methods are used to delete the original data.

This method may only be used when the current status is 2 (MSSRSTATUS\_READY).

The method returns 0 (MSSRERROR\_NOERROR) if it succeeds, or a non-zero error code otherwise.

## 5.4 CopyOrMoveTime

```
SHORT errorCode = CopyOrMoveTime([in] DOUBLE startTime,  
                                [in] DOUBLE endTime,  
                                [in] DOUBLE insertTime,  
                                [in] SHORT move)
```

This method performs the same function as **CopyOrMoveSamples**, except that the start, end and insert positions are specified by time (in seconds) rather than samples.

## 5.5 DeleteSamples

```
SHORT errorCode = DeleteSamples(  
    [in] LONG nSamples  
)
```

This method deletes **nSamples** samples from the currently set position in the file. Thus, the samples deleted have index **CurrentSamplePosition** to **CurrentSamplePosition+nSamples-1**.

## 5.6 DeleteTime

```
SHORT errorCode = DeleteTime(  
    [in] DOUBLE time  
)
```

This method deletes an amount of time **time** from the currently set position in the file. Thus, the data deleted has time position from **CurrentTimePosition** to **CurrentTimePosition+time**.

## 5.7 GetAcmFormatAsString

```
SHORT errorCode = GetAcmFormatAsString(  
    [out] STRING Format  
)
```

This method returns a text string description of the currently-set format in the `AcmFormat` property. The description is returned in the **Format** parameter, which should be a `STRING` variable.

- 245 The method returns 0 (`MSSRERROR_NOERROR`) if it succeeds, or a non-zero error code otherwise (for example, because the `AcmFormat` property has not been set).

## 5.8 GetCurrentFormatAsString

```
SHORT errorCode = GetCurrentFormatAsString(  
    [out] STRING Format  
250 )
```

This method returns a text string description of the format of the currently-open file. The description is returned in the **Format** parameter, which should be a `STRING` variable. The `Status` property must be `MSSR_READY`, `MSSR_RECORDING` or `MSSR_PLAYBACK` for this function to work (i.e. a file must be open).

- 255 The method returns 0 (`MSSRERROR_NOERROR`) if it succeeds, or a non-zero error code otherwise.

## 5.9 GetLevels

```
GetLevels(  
    [out] SINGLE leftLevel,  
260 [out] SHORT leftOverload,  
    [out] SINGLE rightLevel,  
    [out] SHORT rightOverload  
    )
```

- 265 This method returns the current peak levels, in dB, for the left (or only) and right channels in **leftLevel** and **rightLevel**.

If overload is currently flagged on the left (or only) channel, **leftOverload** will be set to 1, otherwise it will be set to 0.

If overload is currently flagged on the right channel, **rightOverload** will be set to 1, otherwise it will be set to 0.

- 270 The data is only meaningful if the current **Status** is `MSSR_RECORDING` or `MSSR_PLAYBACK`. If **NumChannels** is 1, the right channel values are undefined.

## 5.10 OpenExistingOrNewFile

```
SHORT errorCode = OpenExistingOrNewFile(  
    [in] STRING Filename  
275 )
```

This method opens an existing or new file with the supplied file name. Any recording or playback in progress is stopped (see the **Stop** method), and any existing open file closed (see the **Reset** method), before attempting to open the file.

280 The filename can specify a 'local' file (e.g. **C:\WaveData\File.wav**) or it can specify a remote file by network share using a 'UNC' filename (e.g. **\\RemoteComputer\Share\WaveData\File.wav**). In the latter case, the **UNCUser** and **UNCPassword** properties should be set before attempting to open the file – see section 9 for more details.

285 *If the file exists*, it is opened. The properties **NumSamples**, **TotalTime**, **CurrentSamplePosition** and **CurrentTimePosition** are all set to the file duration. **TemporaryFile** is set to 0 and **Filename** is set to the supplied file name. **SampleRate**, **BitsPerSample** and **NumChannels** are set to indicate the format of the file. Finally, **Status** is set to **MSSR\_READY**.

290 *If the file does not exist*, it is created with the wave format specified by the **SampleRate**, **BitsPerSample** and **NumChannels** properties (and the **AcmFormat** property, if appropriate). The properties **NumSamples**, **TotalTime**, **CurrentSamplePosition** and **CurrentTimePosition** are all set to 0. **TemporaryFile** is set to 0 and **Filename** is set to the supplied file name. Finally, **Status** is set to **MSSR\_READY**.

295 The method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.11 OpenNewFile

```
300 SHORT errorCode = OpenNewFile(  
        [in] STRING filename  
    )
```

This method opens a new file with the supplied file name. Any recording or playback in progress is stopped (see the **Stop** method), and any existing open file closed (see the **Reset** method), before attempting to create the file. If a file of the supplied name already exists, it is truncated to zero length before being opened.

305 The filename can specify a 'local' file (e.g. **C:\WaveData\File.wav**) or it can specify a remote file by network share using a 'UNC' filename (e.g. **\\RemoteComputer\Share\WaveData\File.wav**). In the latter case, the **UNCUser** and **UNCPassword** properties should be set before attempting to open the file – see section 9 for more details.

310 The file is created with the wave format specified by the **SampleRate**, **BitsPerSample** and **NumChannels** properties (and the **AcmFormat** property, if appropriate). The properties **NumSamples**, **TotalTime**, **CurrentSamplePosition** and **CurrentTimePosition** are all set to 0. **TemporaryFile** is set to 0 and **Filename** is set to the supplied file name. Finally,  
315 **Status** is set to **MSSR\_READY**.

This method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.12 OpenTemporaryFile

```
SHORT errorCode = OpenTemporaryFile()
```

320 This method opens a new temporary file. Any recording or playback in progress is stopped (see the **Stop** method), and any existing open file closed (see the **Reset** method), before attempting to create the file.

The file is created with the wave format specified by the **SampleRate**, **BitsPerSample** and **NumChannels** properties (and the **AcmFormat** property, if appropriate). The properties **NumSamples**, **TotalTime**, **CurrentSamplePosition** and **CurrentTimePosition** are all set to 0. **TemporaryFile** is set to 1 and **Filename** is set to the full path and filename of the created temporary file. Finally, **Status** is set to **MSSR\_READY**.

330 This method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.13 Register

```
SHORT errorCode = Register(  
    [in] STRING name,  
    [in] STRING keycode  
)
```

335

This method is used to pass user name and registration key information to the control, in order to validate use of the control after the 30-day evaluation period.

The **Register** method should be called at run time immediately after the control is created, before calling any other method. It is recommended that registration information is hard-coded into applications that use the MSSR300 control, so that end users do not experience any difficulties with use of the control.

340

This method returns 0 (**MSSRERROR\_NOERROR**) if the registration information is valid, or a non-zero error code otherwise.

## 5.14 RePack

```
345 SHORT errorCode = RePack()
```

If the **DeleteSamples** or **DeleteTime** methods are used, the end of the WAV file will be unused. The **RePack** method reclaims this unused disk space. Typically, this method would be called just before closing a file, if the **DeleteSamples** or **DeleteTime** methods had been used at any time whilst it was open. Use of this method is, however, optional: Re-packing the data may take some time for large files (and uses some disk space); and the unused space at the end of the file will be benign except that WAV file players that do not properly understand the WAV file structure may not play the file properly.

350

## 5.15 Reset

355 **SHORT** **errorCode** = **Reset** ( )

This method is used to reset the control to its idle state, with no file open.

(a) If playback or recording is currently in progress, it is stopped (see the **Stop** method)

(b) If a file is currently open, it is closed; then, if **TemporaryFile** is non-zero, the file just closed is deleted. The **Filename** property is set to an empty string.

360 (c) The current wave format settings in **SampleRate**, **BitsPerSample** and **NumChannels** are checked to see if they are supported by the sound hardware.

(d) The **Status** property is set to **MSSR\_IDLE**.

This method returns 0 (**MSSRERROR\_NOERROR**) if successful, or a non-zero error code if an error occurred or if the currently set wave format is not supported.

## 365 5.16 ShowMixer

```
SHORT errorCode = ShowMixer (  
    [in] SHORT mixerType  
)
```

370 This method is used to start the Windows mixer application. **mixerType** should be set to 0 to show the playback mixer settings, or non-zero (e.g. 1) to show the recording mixer settings<sup>4</sup>.

This method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.17 StartPlayback

375 **SHORT** **errorCode** = **StartPlayback** ( )

This method starts playback at the position specified by property **CurrentSamplePosition** or **CurrentTimePosition**. The method should only be used when the **Status** property has the value **MSSR\_READY** and **NumSamples** is greater than zero.

380 This method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.18 StartRecording

```
SHORT errorCode = StartRecording ( )
```

---

<sup>4</sup> Some audio drivers always show the playback mixer settings initially, even if **mixerType** is set non-zero.

385 This method starts recording at the position specified by property **CurrentSamplePosition** or **CurrentTimePosition**. The method should only be used when the **Status** property has the value **MSSR\_READY**.

This method returns 0 (**MSSRERROR\_NOERROR**) if it succeeds, or a non-zero error code otherwise.

## 5.19 Stop

390 **Stop ()**

This method only has an effect when the current **Status** is **MSSR\_RECORDING** or **MSSR\_PLAYBACK**. Its effect is to stop playback or recording. Note that you can pause, rather than stop, playback or recording by using the **Paused** property.

## 6. Events/Callbacks

395 The MSSR300 control provides the following events, normally implemented as callbacks or object functions:

### 6.1 NewLevelValues

```
400         NewLevelValues (  
            [out] SINGLE leftLevel,  
            [out] SHORT leftOverload,  
            [out] SINGLE rightLevel,  
            [out] SHORT rightOverload  
        )
```

405 *NOTE: This event is fired 10 times per second during recording or playback. If your application uses this event, it should be able to handle an event occurrence of this frequency.*

The primary application of this event is to return level information to the application during recording or playback so that it can log or display the data in a non-standard way. This event returns the current peak levels, in dB, for the left (or only) and right channels in **leftLevel** and **rightLevel**.

410

If overload is currently flagged on the left (or only) channel, **leftOverload** will be set to 1, otherwise it will be set to 0.

If overload is currently flagged on the right channel, **rightOverload** will be set to 1, otherwise it will be set to 0.

415 If **NumChannels** is 1, the right channel values are undefined.

### 6.2 NewVoxState

```
         NewVoxState([out] SHORT voxState)
```

This event indicates that the VOX state has changed during recording (i.e. that a VOX-controlled pause has started or finished). The parameter **voxState** will be 1 if a pause has just started, or 0 if a pause has just finished.

420

### 6.3 PlaybackStopped

```
         PlaybackStopped([out] SHORT errorCode)
```

This event indicates that playback has been stopped. The parameter **errorCode** will be 0 (MSSRERROR\_NOERROR) if playback was stopped normally, otherwise it will be set to a non-zero error code to indicate that playback stopped because of an error.

425

## 6.4 RecordingStopped

**RecordingStopped**([out] **SHORT** **errorCode**)

430 This event indicates that recording has been stopped. The parameter **errorCode** will be 0 (**MSSRERROR\_NOERROR**) if recording was stopped normally, otherwise it will be set to a non-zero error code to indicate that recording stopped because of an error. In particular, **errorCode** is set to **MSSRERROR\_RECORDLIMITEXCEEDED** if recording has stopped because the pre-set time limit in **RecordLimit** or **RecordTimeLimit** has been exceeded.

## 7. Error codes

435 The MSSR300 Control supports the following error codes as return values from methods (see section 5):

Value	Symbol	Description
0	MSSRERROR_NOERROR	Method was successful
1	MSSRERROR_NODEVICE	No recording device can be found capable of recording with the current settings
2	MSSRERROR_CANTOPENFILE	Cannot open the specified file for read and write
3	MSSRERROR_CANTWRITEFILE	Cannot write to the designated file. You may have run out of disk space.
4	MSSRERROR_NOTAWAVEFILE	The specified file is not in Microsoft WAVE format.
5	MSSRERROR_UNSUPPORTEDAUDIOFORMAT	The specified file is a WAVE format that the MSSR300 Control does not understand. The control only understands PCM wave files with 8 or 16 bit sample size and 1 or 2 channels.
6	MSSRERROR_RESOURCE	There are insufficient operating system resources available to carry out this method.
7	MSSRERROR_WRONGSTATE	The control is in the wrong state to carry out this method (see the Status property).
8	MSSRERROR_RECORDERROR	An error (most likely a file write error) occurred during recording.
9	MSSRERROR_STARTRECORDERROR	An error occurred whilst starting recording. The wave audio recording device most probably is not available or does not support the required wave format.
10	MSSRERROR_STARTPLAYERROR	An error occurred whilst starting playback. The wave audio playback device most probably is not available or does not support the required wave format.
11	MSSRERROR_PLAYBACKERROR	An error (most likely a file read error) occurred during playback.
12	MSSRERROR_DIDNTWORK	The method to display the windows Mixer application did not work.
13	MSSRERROR_RECORDLIMITEXCEEDED	The recording time limit set in property <i>RecordLimit</i> (or <i>RecordTimeLimit</i> ) has been reached or exceeded.
14	MSSRERROR_NOACM	The Windows Audio Compression Manager cannot be accessed.

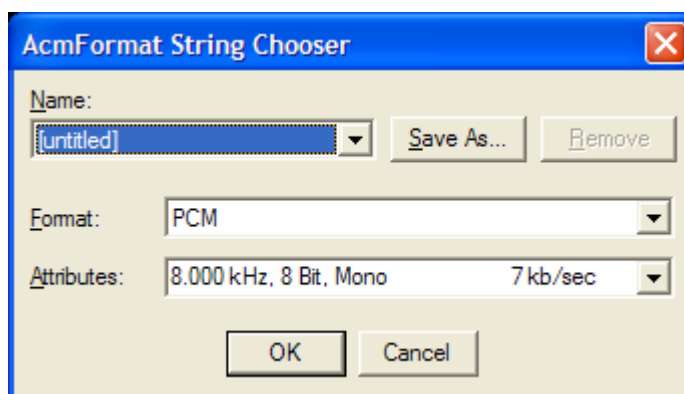
<b>Value</b>	<b>Symbol</b>	<b>Description</b>
15	MSSRERROR_CANCEL	The user cancelled the ACM format selection dialog.
16	MSSRERROR_UNAVAILABLE	The text string representation of the currently-set ACM format is unavailable.
17	MSSRERROR_ACMFORMATNOTSET	(not currently used)
18	MSSRERROR_PLAYONLY	The operation cannot be carried out as the file is set to playback only (no recording or modification allowed)
19	MSSRERROR_NOTPOSSIBLE	The operation cannot be completed for an unspecified reason
99	MSSRERROR_EXPIRED	Your time-limited evaluation of the MSSR300 Control has expired. To continue using the control, please purchase a control code and use the Register method each time the control is used.

## 8. Acm Format Strings

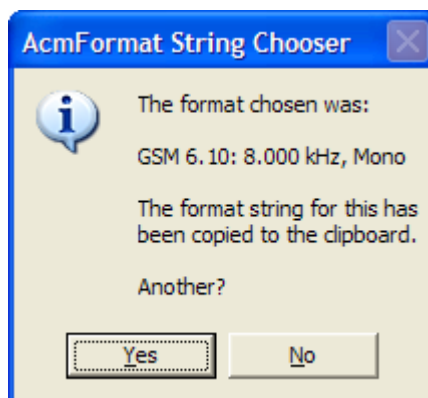
440 The **AcmFormat** property may be set by assigning to it a specially-formatted text string, which is an ASCII representation of the binary data.

In order to assist developers in the preparation of code, the **AcmFormatString.exe** applet is included with the QuickRecord package. This can be found in the installation folder for the package (typically **C:\Program Files\QuickRecord2**).

445 When you run the **AcmFormatString** applet, the ACM format chooser dialog appears.



If you select a format using this dialog, and click on **OK**, another dialog appears:



450 This dialog says that the required string for the chosen format has been copied to the clipboard. You can then paste this string directly into your application.

As an example, the string for **GSM 6.10: 8.000 kHz, Mono** format looks like this:

```
"31000100401f0000590600004100000002004001"
```

And can be assigned to the **.AcmFormat** property directly, e.g.:

```
MSSR300Ctrl1.AcmFormat = "31000100401f0000590600004100000002004001"
```

455

## 9. Files specified by UNC path

The **OpenNewFile()** and **OpenExistingOrNewFile()** methods can specify filenames as UNC paths. These paths specify a file (possibly within a sub-folder) within a share on a remote computer, and have the typical form:

460 `\\RemoteComputer\Share\Folder\Filename.wav`

When such a name is specified, the MSSR300 control tries to establish a connection with the share on the remote computer (in the above case, `\\RemoteComputer\Share`) before attempting to open the file.

465 To establish the connection to the share, the MSSR300 control must supply a username and password that is valid on the remote computer. The **UNCUser** and **UNCPassword** properties should be set before the **Open...()** methods are used so that the control can supply appropriate information to the remote computer.

These properties may be set in three ways:

- 470 1. **UNCUser** and **UNCPassword** both set to zero-length strings: The control uses the default login settings for the current user when trying to establish the connection.
2. **UNCUser** set to a non-zero-length string (but not "\*", see below), **UNCPassword** set to anything: The control uses the specified property values as the user name and password to use when establishing the connection.
- 475 3. **UNCUser** set to the single character string "\*", **UNCPassword** set to anything: The control displays the standard Windows dialog to allow the user to enter the username and password to use for establishing the connection.