# Pa1Lib Shared Library Reference

## for Developer

**YAMAHA Corporation**

Revision 1.10
November 14, 2001

# 1. Table of Contents



## 2. Preview

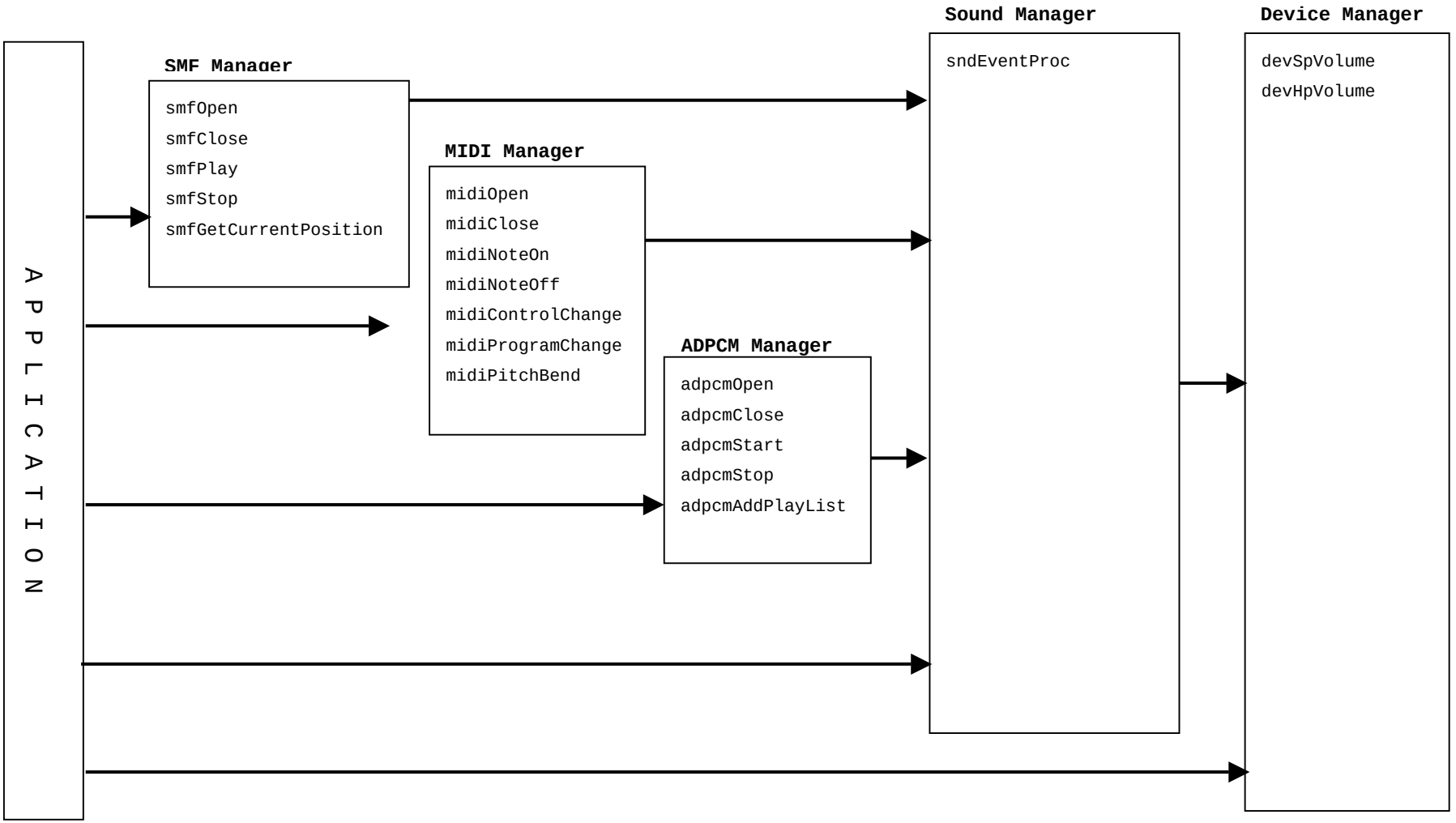
This document provides information of the functions supported in the Pa1Lib.prc Shared Library. To use these functions, it is required to include Pa1Lib.h.

## 3. Features of Pa1Lib

- Capable of producing up to 16 sounds simultaneously
- Supports GM synthesizer (128-note melody, 47-note drum)
- Supports 29 sound effects for game software
- Capable of playing SMF data (format 0 only)
- Notifies the end of the SMF playback using the callback function.
- SMF data can be played specifying the start point and the end point of the playback.
- SMF data can be played specifying the playback channel ranges.
- Capable of playing up to 16 SMF data simultaneously
- Capable of opening up to 16 virtual MIDI devices
- MIDI sound can be controlled via the MIDI command during the SMF is played.
- The last defined sound will be produced first if the number of sounds exceeds 16.
- Plays 1-channel ADPM data
- Capable of playing multiple ADPCM data using the playback list feature.

# 4. Pa1Lib Structure



# 5. Structures

This section describes the structures supported in the SMF, MIDI, and ADPCM Sound Managers.

## 5.1. CallbackInfoType

Each callback routine should use the information in the structure below.

```
typedef struct CallbackInfoType {
    void        (*funcP)(UInt32);
    UInt32      dwUserData;
} CallbackInfoType;
```

| | |
|---|---|
| funcP | Pointer to the callback routine |
| dwUserData | Parameter to be passed to the callback routine |

For example, the callback routine can be declared as below.

```
void smfCallbackFunc(UInt32 dwUserData)
{
        // do operation

}
```

## 5.2. SmfMgrChanRangeType

Use this structure to define which channel in the data is to be played during the SMF playback.

```
typedef struct SmfMgrChanRangeType {
    UInt8       bFirstChan;
    UInt8       bLastChan;
} SmfMgrChanRangeType;
```

| | | |
|---|---|---|
| BfirstChan | The first playback channel | (0 –15) |
| BLastChan | The last playback channel | (0 – 15) |

## 5.3. SmfMgrPlayTimeType

Use this structure to set up the time information required for the SMF playback.

```
typedef struct SmfMgrPlayTimeType {
   UInt32 dwStartMilliSec;
   UInt32 dwEndMillisec;
} SmfMgrPlayTimeType;
```

| | |
|---|---|
| dwStartMilliSec | The start position of the playback. The value is an offset from the beginning of the track in milliseconds. When 0 is defined, the playback will be started from the beginning of the track. |
| dwEndMilliSec | The end position of the playback. The value is an offset from the beginning of the track in milliseconds. When 0xFFFFFFFF is defined, the playback will be completely finished till the end of the data. |

# 6. Playback Mode

Pa1Lib provides 2 playback modes for playing the SMF data and the ADPCM data. One is the Event Mode that processes playback by calling the **sndEventProc** function periodically during the event loop. The other is the Interrupt Mode that processes playback automatically using the hardware interrupts.

The playback mode is specified passing the mode parameter to the playback start function so the playback of the SMF data or the ADPCM data can be processed. The same playback mode should be used through out one application.

## 6.1. Event Mode (sndEventProc)

After the SMF data is played, if the sndEventProc function is called, the Pa1Lib will internally calculate how long the time has passed from the playback started or from the previous call of the sndEventProc function. It will then process the MIDI data in the calculated time.

For the ADPCM data, the data will be jammed to the hardware buffer when the sndEventProc function is called after the playback starts.

For the SMF data, as the playback is usually processed every 10ms, the application should call the sndEventProc function in a proper interval in the event loop function.

## 6.2. Interrupt Mode

This playback mode use hardware interrupts to play the data. The interrupt interval is around 10ms. When the Interrupt Mode is used, the application is responsible only to start the playback; the data will be played automatically.

# 7. Open/Close the Library

The Pa1Lib should be opened before you can use the functions of the Pa1Lib and should be closed after you have finished using the library.

## 7.1. Pa1Lib_Open()

**Purpose** Open Pa1Lib

**Prototype** `Err PA1LibOpen(UInt16 refNum)`

**Parameters** `-> refNum` Reference Number

**Result** Returns 1 when the Pa1Lib is not installed in the system or when the hardware device cannot be found.

Returns 0 when succeeds.

**Comments**

## 7.2. Pa1Lib_Close()

**Purpose** Close Pa1Lib.

**Prototype** `Err PA1LClose(UInt16 refNum)`

**Parameters** `-> refNum` Reference Number

**Result** Returns 0.

**Comments**

# 8. SMF Manager

This section describes the functions supported in the SMF Manager.

The SMF Manager only supports playing the SMF data of format 0. Up to 16 different SMF data can be played at the same time. It can produce up to 16 sounds at one time. The SMF Manager calls the specified callback routine when the playback of the SMF data is finished.

## 8.1. Pa1Lib_smfOpen

**Purpose** Used to register the SMF data to the SMF Manager and obtain the handle.

**Prototype**
```
Err PA1L_smfOpen(UInt16 refNum, UInt8 *smfP, MemPtr reservedP,
        UInt8 *hd, UInt32 *duration, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refNum` | Reference Number |
| `-> smfP` | Pointer to the beginning of the SMF data |
| `-> reservedP` | Should be NULL |
| `<- hd` | Return the handle registered by the SMF Manager. The handle sets with a value of 0-15. |
| `<- duration` | The playback duration is returned in milliseconds. |
| `-> retval` | Returns FALSE when trying to open more than 16 SMF data or when the opened SMF data are not supported.<br>Returns TRUE when succeeds. |

**Result** returns 0.

**Comments** The contents of the memory specified by **smfP** must be retained until the SMF data is closed by **smfClose**.

## 8.2. Pa1Lib_smfClose

**Purpose** Used to close the handle to the specified SMF data and release the handle.

**Prototype**
```
Err PA1L_smfClose(UInt16 refNum, UInt8 hd, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refNum` | Reference Number |
| `-> hd` | Handle to the SMF data |
| `-> retval` | Returns FALSE when trying to close an un-opened handle.<br>Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** When the SMF data is closed during the SMF playback, the SMF Manager will stop the playback and then release the handle of the SMF data.

## 8.3. Pa1Lib_smfPlay

**Purpose** Used to start playing the SMF data specified by the handle.

**Prototype**
```
Err PA1L_smfPlay(UInt16 refNum, UInt8 hd, UInt8 mode,
        SmfMgrPlayTimeType* playTimeP、
        SmfMgrChanRangeType* chanRangeP,
        CallbackInfoType* callbackInfoP, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the SMF data |
| `-> mode` | Playback mode. "0" for the Event Mode and "1" for the Interrupt Mode. |
| `-> playTimeP` | Pointer to the SmfMgrPlayTimeType structure.<br>Set to NULL if not to specify the playback duration. |
| `-> chanRangeP` | Pointer to the SmfMgrChanRangeType structure.<br>Set to NULL if not to specify the playback channel. |
| `-> callbackInfoP` | Pointer to the CallbackInfoType structure.<br>Set to NULL if not to call the callback function. |
| `-> retval` | Returns FALSE when the specified handle is not opened and when the time specified in the SmfMgrPlayTimeType structure is longer than the playback duration of the SMF data.<br>Also returns FALSE when the specified range is fallen outside 0-15 in the SmfMgrPlayTimeType structure.<br>Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** The callback will be executed when the playback is finished if the callback function is set up.

As the library remembers the address information of the pointers of various parameters, it is possible to change or undo the function even after the function is called.

The playback will start from the beginning of the data if the SMF data of the specified handle is currently in play.

## 8.4. Pa1Lib_smfStop

**Purpose** Used to stop playing the SMF data specified by the handle.

**Prototype**
```
Err PA1L_smfStop(UInt16 refNum, UInt8 hd, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the SMF data. |
| `-> retval` | Returns FALSE when the specified handle is not played. |
| | Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

## 8.5. Pa1Lib_smfGetCurrentPosition

**Purpose** Used to retrieve the current playback position in the SMF data of the specified handle.

**Prototype**
```
Err PA1L_smfGetCurrentPosition(UInt16 refNum, UInt8 hd,
        UInt32* pos, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the SMF data |
| `<- pos` | The playback position is returned in milliseconds. |
| `-> retval` | Returns FALSE when the specified handle is not played. |
| | Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

# 9. MIDI Manager

This section describes the functions supported in the MIDI Manager.

The MIDI Manager holds lower-level functions than the SMF Manager does. It controls the sounds with the MIDI channel unit. Up to 16 virtual MIDI devices can be opened at the same time.

## 9.1. Pa1Lib_midiOpen

**Purpose** Used to open MIDI device and obtain the handle.

**Prototype**
```
Err PA1L_midiOpen(UInt16 refNum, MemPtr reservedP, UInt8 *hd,
        Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> reservedP` | Always set to NULL. |
| `<- hd` | Returns the handle of the MIDI device. |
| `-> retval` | Returns FALSE when more than 16 MIDI devices are opened.<br>Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

## 9.2. Pa1Lib_midiClose

**Purpose** Used to close the MIDI device and release the handle.

**Prototype**
```
Err PA1L_midiClose(UInt16 refNum, UInt8 hd, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** It will close the MIDI device and stop the sound if there are sounds in production on the channels.

## 9.3. Pa1Lib_midiNoteOn

**Purpose** Used to produce sounds on any channels of the MIDI device of the specified handle.

**Prototype** `Err PA1L_midiNoteOn(UInt16 refNum, UInt8 hd, UInt8 ch, UInt8 key, UInt8 velocity, Boolean* retval)`

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> ch` | Channel (0 - 15) |
| `-> key` | Key (0 - 127) |
| `-> velocity` | Velocity (0 - 127) |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

## 9.4. Pa1Lib_midiNoteOff

**Purpose** Used to stop producing sounds on any channels of the MIDI device of the handle.

**Prototype** `Err PA1L_midiNoteOff(UInt16 refNum, UInt8 hd, UInt8 ch, UInt8 key, UInt8 velocity, Boolean* retval)`

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> ch` | Channel (0 - 15) |
| `-> key` | Key (0 - 127) |
| `-> velocity` | Velocity (0) |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0

**Comments**

## 9.5. Pa1Lib_midiControlChange

**Purpose** Used to set up various control changes on any channels of the MIDI device of the specified handle.

**Prototype** `Err PA1L_midiControlChange(UInt16 refNum, UInt8 hd, UInt8 ch, UInt8 ctrl, UInt8 param1, UInt8 param2, Boolean* retval)`

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> ch` | Channel (0 - 15) |

| | |
|---|---|
| `-> ctrl` | The control change number |
| `-> param` | Value of the control change |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** The supported control numbers are listed below. Control numbers not listed below will be ignored.

| | | |
|---|---|---|
| 0 (00h) | : | Bank Select MSB |
| 1 (01h) | : | Modulation |
| 6 (06h) | : | Data Entry MSB |
| 7 (07h) | : | Channel Volume |
| 10 (0Ah) | : | Panpot |
| 32 (20h) | : | Bank Select LSB |
| 38 (26h) | : | Data Entry LSB |
| 100 (64h) | : | RPN MSB |
| 101 (65h) | : | RPN LSB |
| 120 (78h) | : | All Sound Off |
| 121 (79h) | : | Reset All Controller |
| 123 (7Bh) | : | All Note Off |

RPN is supported only for the pitch bend sensitivity of MSB=00h, LSB=00h. It will be ignored for other parameters.

## 9.6. Pa1Lib_midiProgramChange

**Purpose** Used to execute program change on any channels of the MIDI device of the specified handle.

**Prototype**
```
Err PA1L_midiProgramChange(UInt16 refNum, UInt8 hd, UInt8 ch,
        UInt8 prgchng, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> ch` | Channel (0 - 15) |
| `-> prgchng` | Program change (0 - 127) |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

## 9.7. Pa1Lib_midiPitchBend

**Purpose** Used to execute pitch bend on any channels of the MIDI device of the specified handle.

**Prototype**
```
Err PA1L_midiPitchBend(UInt16 refNum, UInt8 hd, UInt8 ch,
        short bend, Boolean* retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the MIDI device |
| `-> ch` | Channel (0 - 15) |
| `-> bend` | Pitch Bend (-8191 - +8191) |
| `-> retval` | Returns FALSE when the MIDI device of the specified handle is not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** The value of the bend will be reset to –8191 if a value below –8191 is specified. The value of the bend will be reset to +8191 if a value over +8191 is specified.

# 10. ADPCM Manager

This section describes the functions supported in the ADPCM Manager.

The ADPCM Manager can play 1-channel ADPCM at 8kHZ and 4kHZ. It can open up to 32 ADPCM data at one time and play the opened 32 ADPCM data continuously.

The ADPCM Manager will call the registered callback routine when the playback of ADPCM data is finished.

## 10.1. Pa1Lib_adpcmOpen

**Purpose** Used to register the ADPCM data to the ADPCM Manager and obtain the handle.

**Prototype**

```
Err PA1L_adpcmOpen(UInt16 refNum, UInt8 fs, UInt8 *dataP,
        UInt32 dwSize,  CallbackInfoType *callbackInfoP,
        UInt8 *hd, Boolean *retval )
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> fs` | Sample rate of the ADPCM data 0: Fs=4kHz, 1:Fs=8kHz |
| `-> dataP` | Pointer to the beginning of the ADPCM data. |
| `-> dwSize` | Size of the ADPCM data. |
| `-> callbackInfoP` | Pointer to the CallbackInfoType structure. Specify NULL when not to use the callback routine. |
| `<- hd` | Return the handle. |
| `-> retval` | Returns FALSE when more than 32 ADPCM data are opened.<br>Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** The sample rate can be set to 4 kHz or 8kHz but 8kHz is recommended.

## 10.2. Pa1Lib_adpcmClose

**Purpose** Used to close the ADPCM data and release the handle.

**Prototype**

```
Err PA1L_adpcmClose(UInt16 refNum, UInt8 hd, Boolean *retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the ADPCM data. |
| `-> retval` | Returns FALSE when the ADPCM data of the specified handle are not opened. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** Trying to close the ADPCM data currently in play will cause the playback stopped and then the ADPCM data will be closed.

## 10.3. Pa1Lib_adpcmStart

**Purpose** Used to start the playback of the ADPCM data of the specified handle.

**Prototype**
```
Err PA1L_adpcmStart(UInt16 refNum, UInt8 hd, UInt8 mode,
        Boolean *retval)
```

**Parameters**

`<- refnum` Reference Number

`-> hd` Handle to the ADPCM data.

`-> mode` Set up the consecutive playback of the ADPCM data and the playback mode.

bit0: Consecutive playback

When “0” is set, the consecutive playback is selected and enables the playback to be continued on the ADPCM data registered in the playback list after the playback of the ADPCM data is finished

Refer to **adpcmAddPlayList** for the details of the playback list.

When "1" is set, it only plays the ADPCM data of the specified handle to the end and stops the playback.

bit1: Playback Mode

When "0" is set, the Event Mode is selected for playback. The playback is executed calling the **sndEventProc()** function periodically. When "1" is set, the playback is executed using hardware interrupts.

`-> retval` Returns FALSE when the ADPCM data of the specified handle are not opened. Returns TRUE when succeeds.

**Result** Returns 0.

**Comments** If starting to play another ADPCM data while one ADPCM data is currently in play, the current playback will be stopped and the new ADPCM data will be played.

Refer to 6. Playback Mode for more information.

## 10.4. Pa1Lib_adpcmStop

**Purpose** Used to stop playing the ADPCM data of the specified handle.

**Prototype** `Err PA1L_adpcmStop(UInt16 refNum, UInt8 hd, Boolean *retval)`

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the ADPCM data |
| `-> retval` | Returns FALSE when the ADPCM data of the specified handle cannot be played. Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments**

## 10.5. Pa1Lib_adpcmAddPlayList

**Purpose** Used to add the specified handle to the playback list when the ADPCM data are to be played continuously.

**Prototype**
```
Err PA1L_adpcmAddPlayList(UInt16 refNum, UInt8 hd,
        Boolean *retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> hd` | Handle to the ADPCM data.<br>The playback list will be initialized when 0xFF is specified. |
| `-> retval` | Returns FALSE when the APDCM data of the specified handle is not opened or cannot be played.<br>Returns TRUE when succeeds. |

**Result** Returns 0.

**Comments** The ADPCM Manager provides a playback list for consecutive playback. It will play the ADPCM data in the playback list continuously. Up to 32 ADPCM data can be registered to the playback list. The played data will be removed from the list.

# 11. Other Function

This section describes the functions to play by Event mode and the functions to control the Volume.

## 11.1. Pa1Lib_sndEventProc

**Purpose** Used to periodically play the SMF data and the ADPCM data

**Prototype**
```
Err PA1L_sndEventProc(UInt16 refNum, Boolean *retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> retval` | Returns FALSE when the playback of all SMF data and ADPCM data is finished. Returns TRUE when there are data remained to be played. |

**Result** Returns 0.

**Comments** Refer to 6. Playback Mode for more information.

## 11.2. Pa1Lib_devSpVolume

**Purpose** Used to set up the speaker volume.

**Prototype**
```
Err PA1L_devSpVolume(UInt16 refNum, UInt8 vol, Boolean *retval)
```

**Parameters**

| | |
|---|---|
| <- refnum | Reference Number |
| -> vol | The speaker volume can be set with a value of 0 - 31. The maximum value is 31 and 0 is to mute. |
| -> retval | The return value is always TRUE. |

**Result** Returns 0.

**Comments**

## 11.3. Pa1Lib_devHpVolume

**Purpose** Used to set up the headphone volumes.

**Prototype**
```
Err PA1L_devHpVolume(UInt16 refNum, UInt8 vol_l, UInt8 vol_r,
        Boolean *retval)
```

**Parameters**

| | |
|---|---|
| `<- refnum` | Reference Number |
| `-> vol_l` | The left channel volume of the headphone. The value can be any of 0 - 31. The maximum is 31 and 0 is to mute. |
| `-> vol_r` | The right channel volume of the headphone. The value can be any of 0 - 31. The maximum is 31 and 0 is to mute. |

| | | |
|---|---|---|
| | `-> retval` | The return value is always TRUE. |

**Result** Returns 0.

**Comments**

# 12. Revision History

| Date | Rev | Description |
|---|---|---|
| 2001/11/14 | 1.10 | ・Replaced the explanation of wrapper function to the explanation of Library function.<br>・Removed the description of un-used function. |
| 2001/6/27 | 1.00 | ・Initial Release |