

FIG. 1

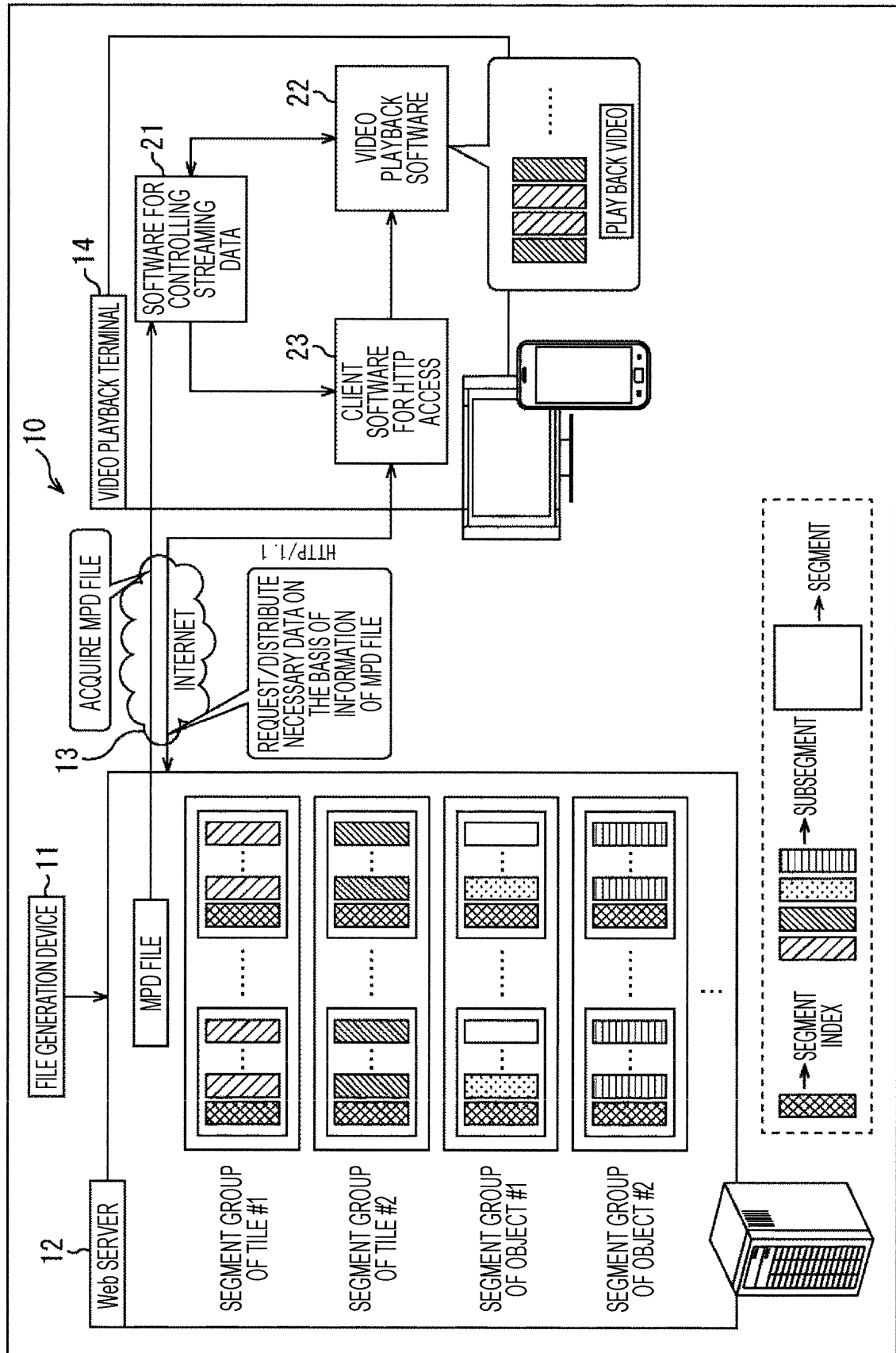


FIG. 2

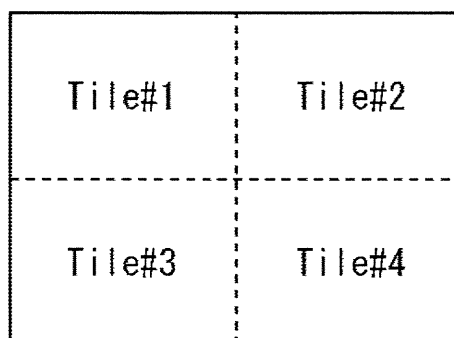


FIG. 3

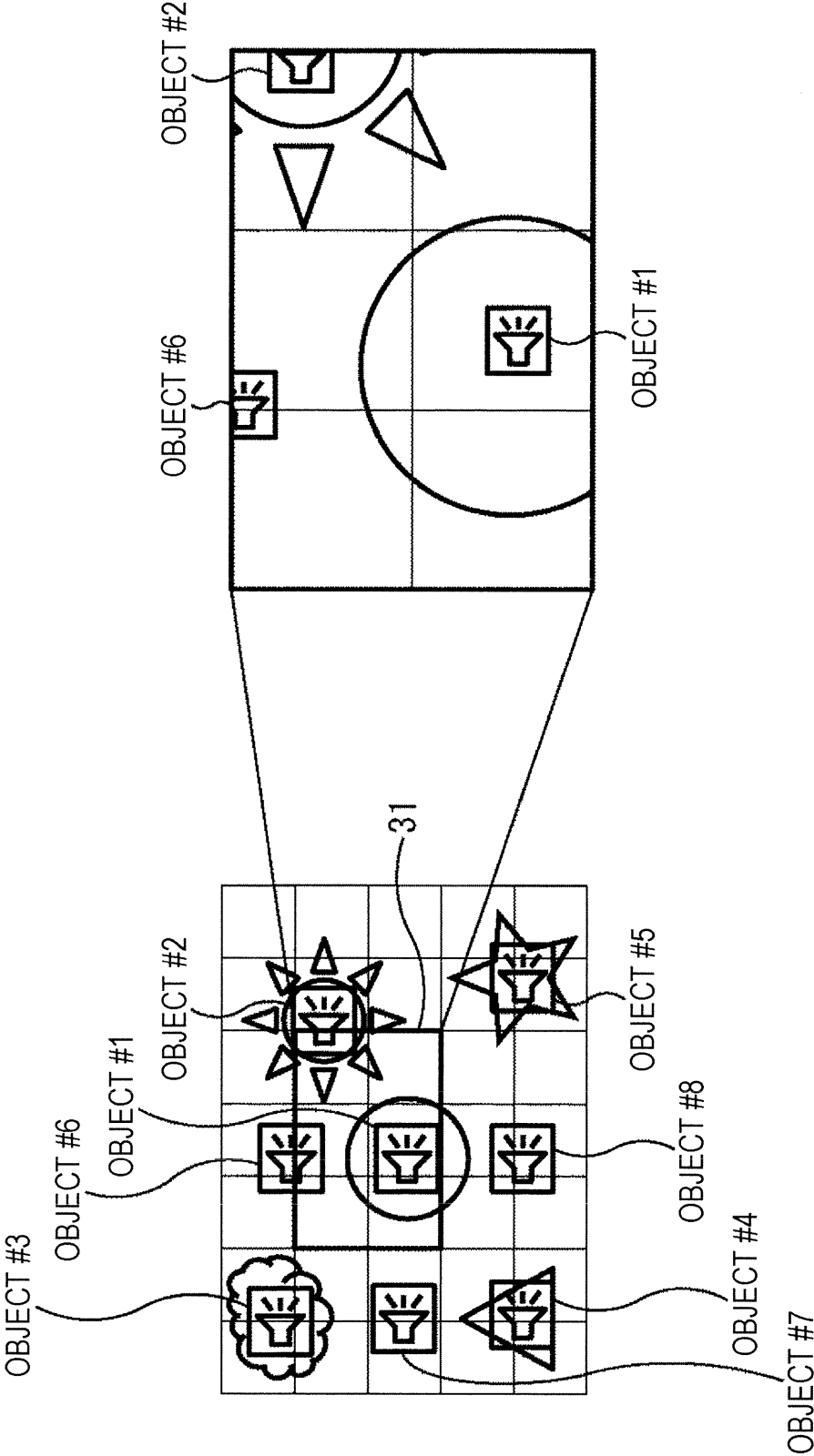


FIG. 4

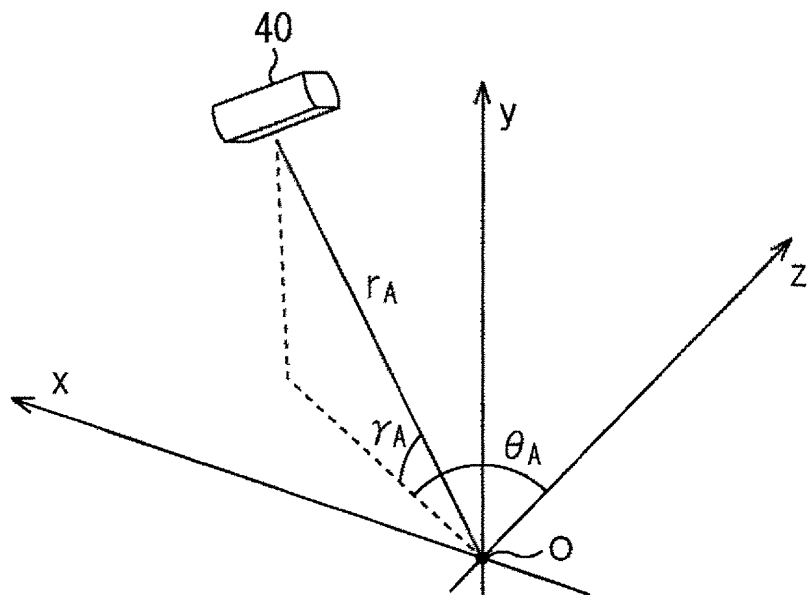


FIG. 5

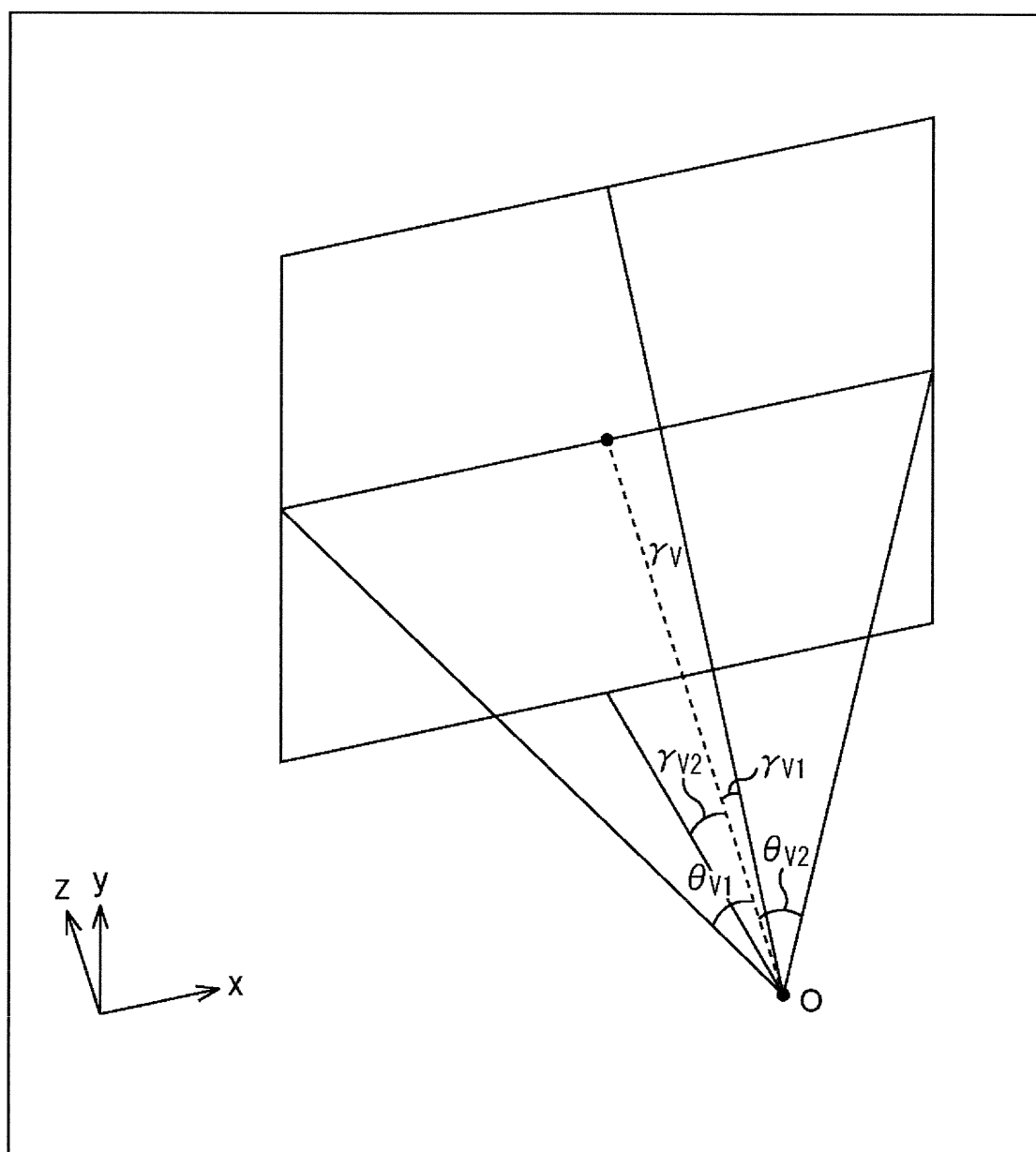


FIG. 6

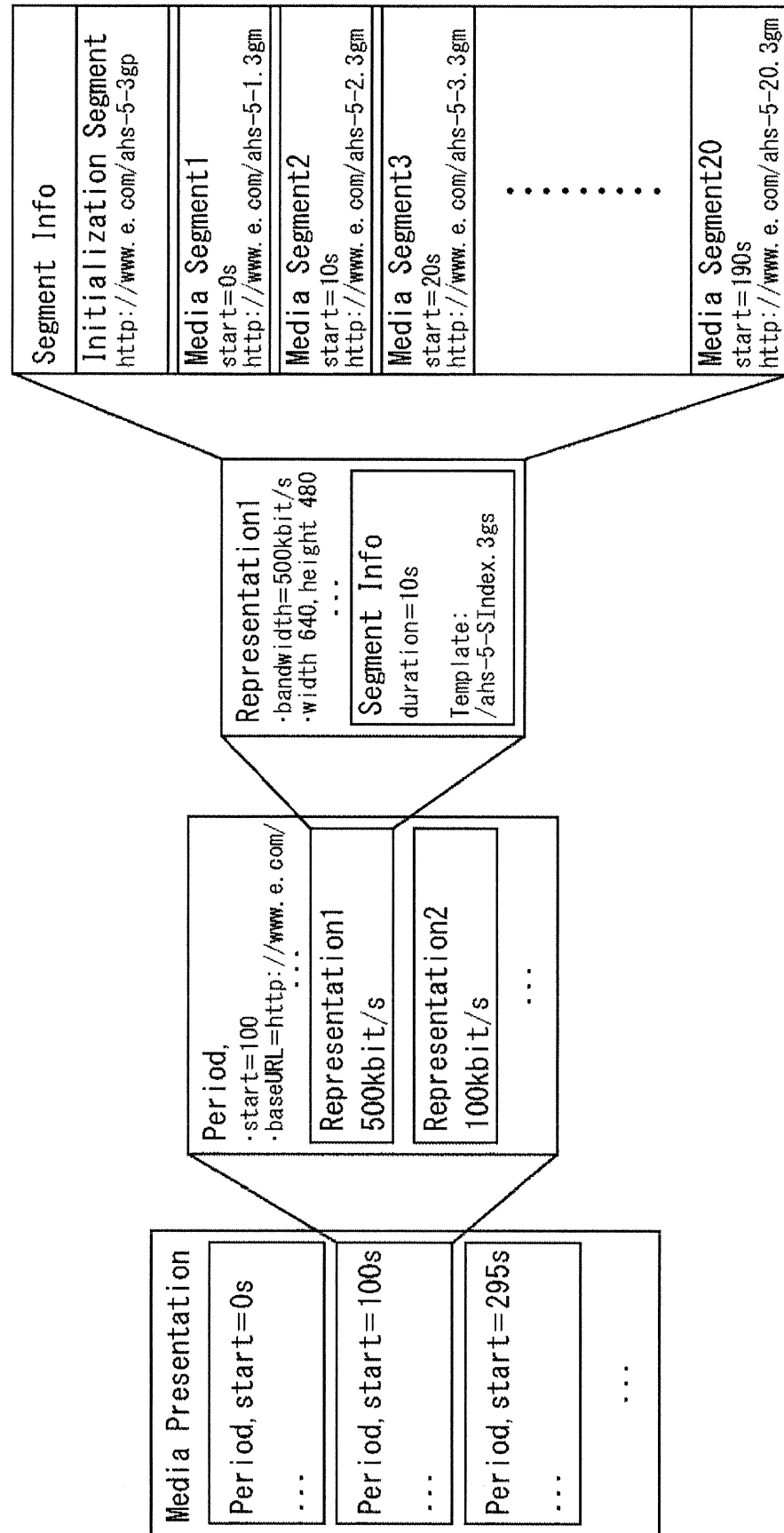


FIG. 7

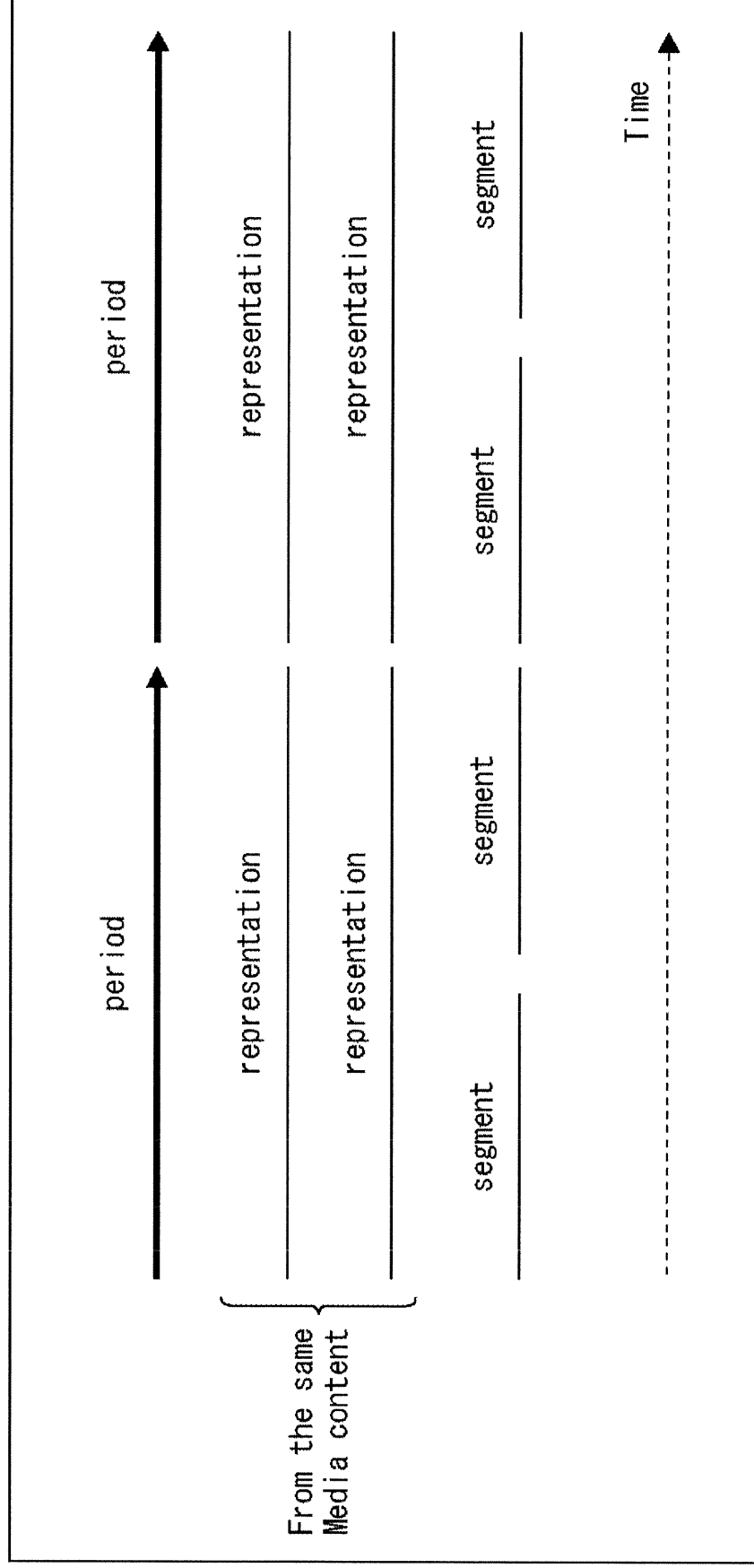


FIG. 8

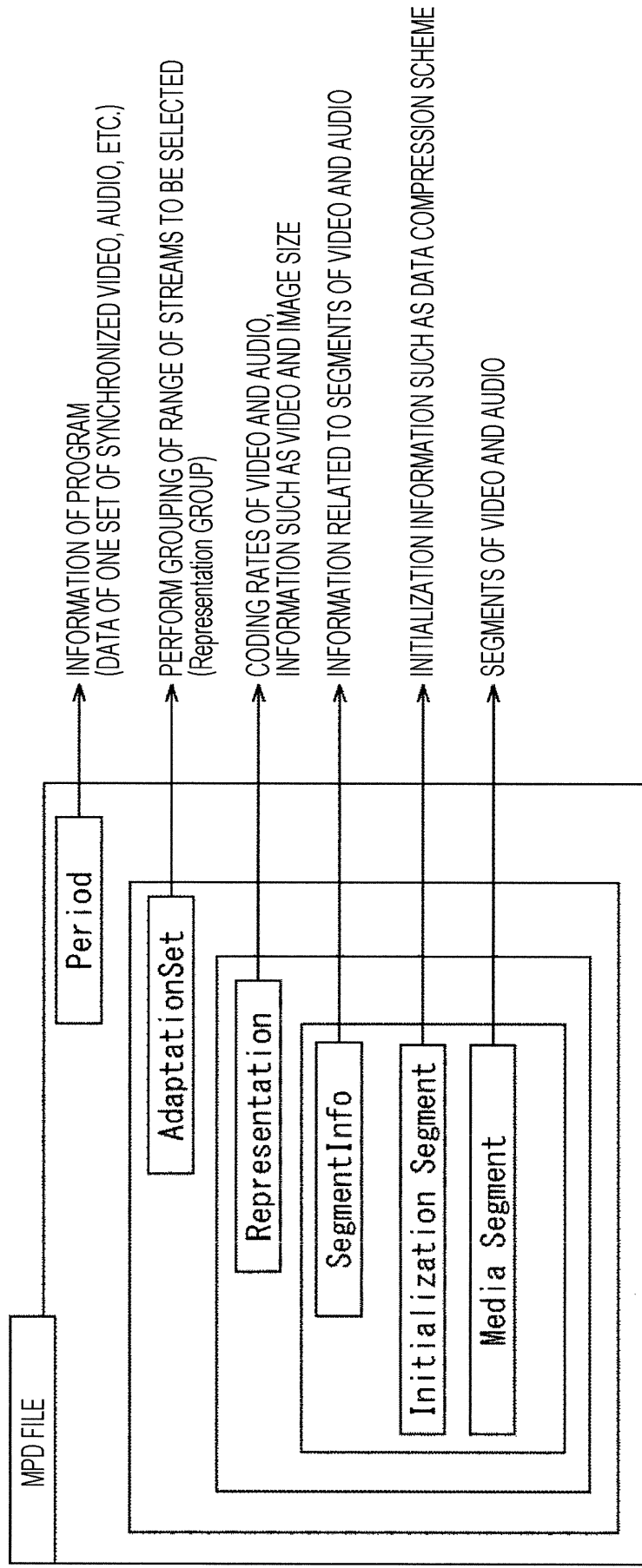


FIG. 9

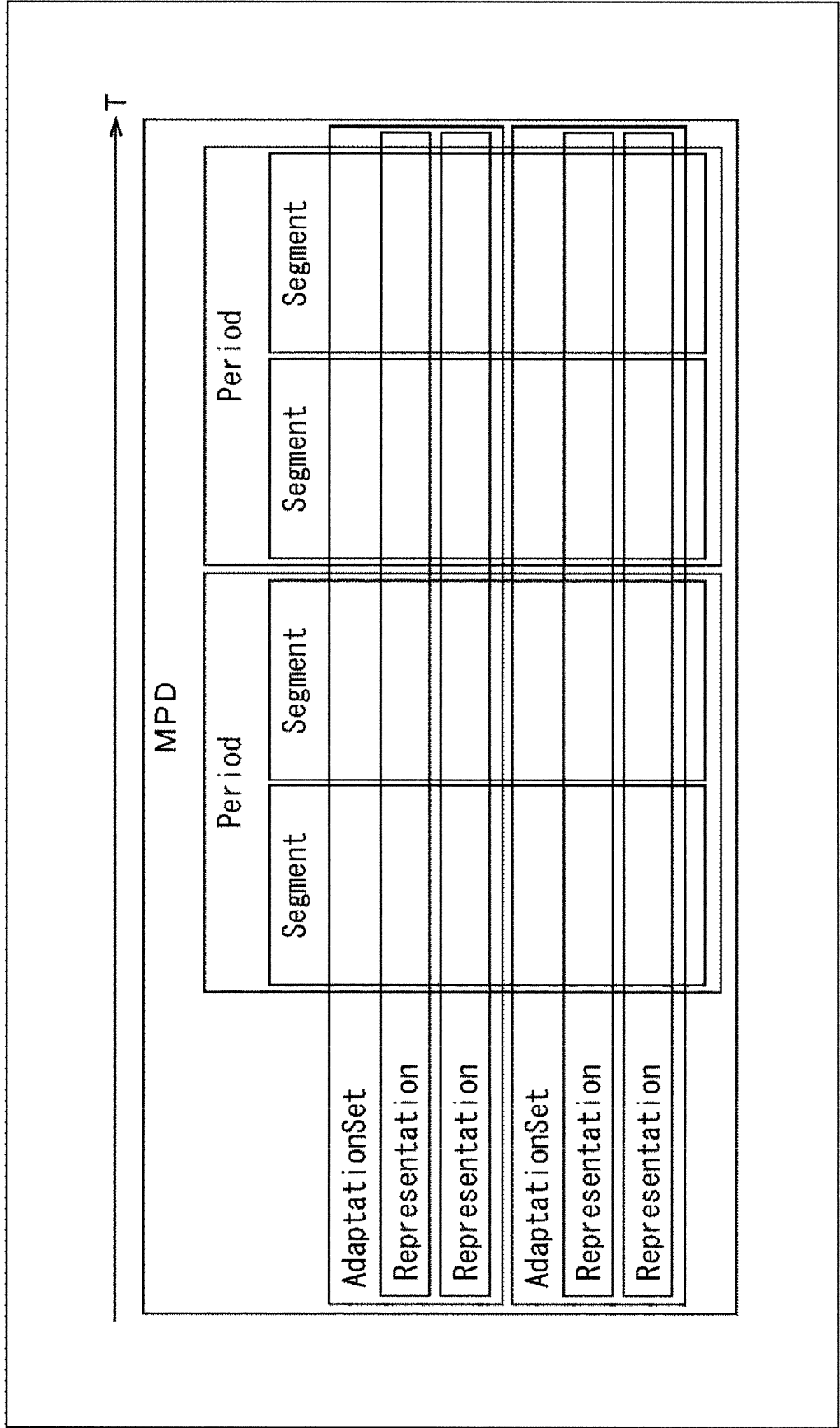


FIG. 10

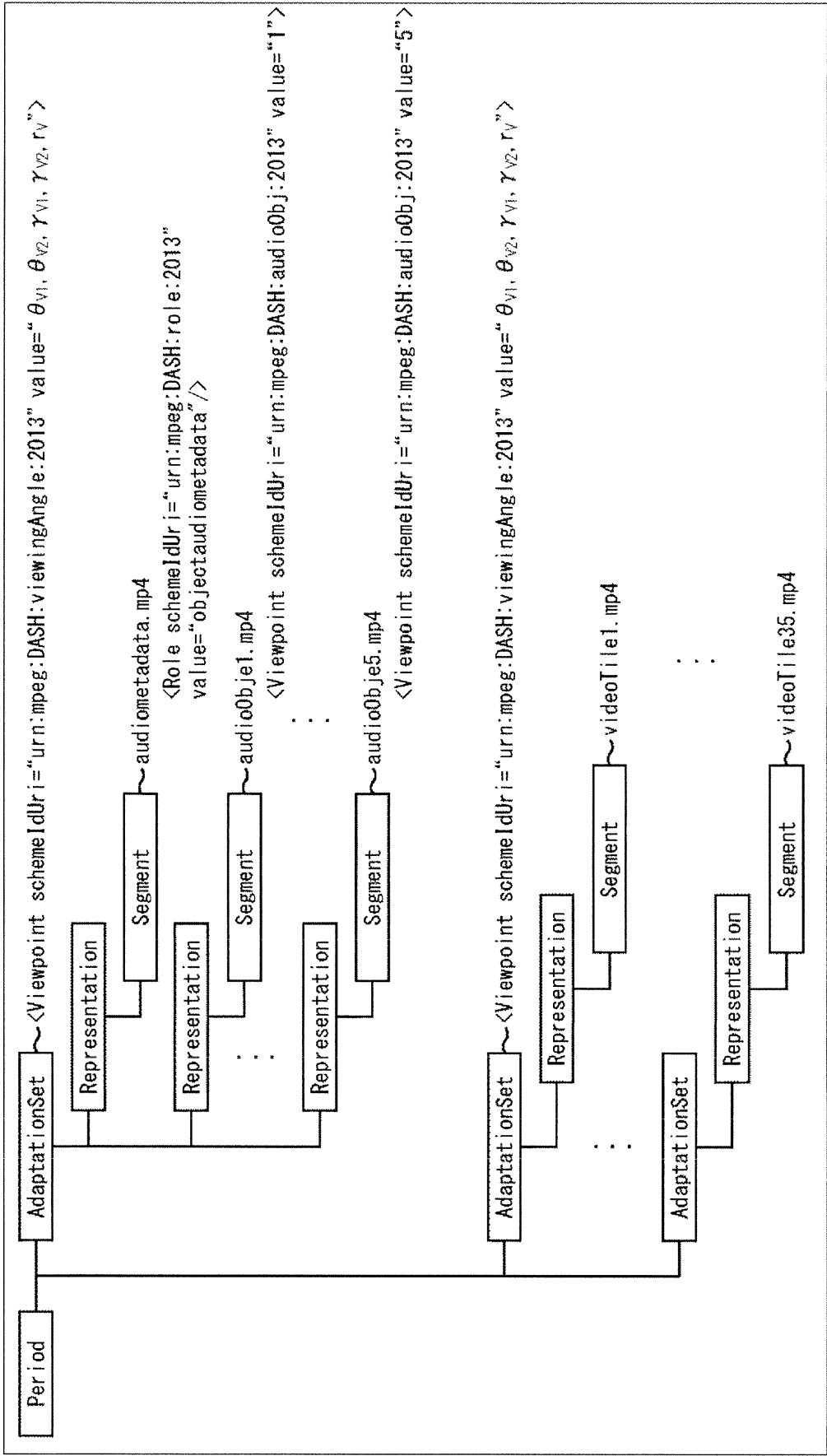


FIG. 11

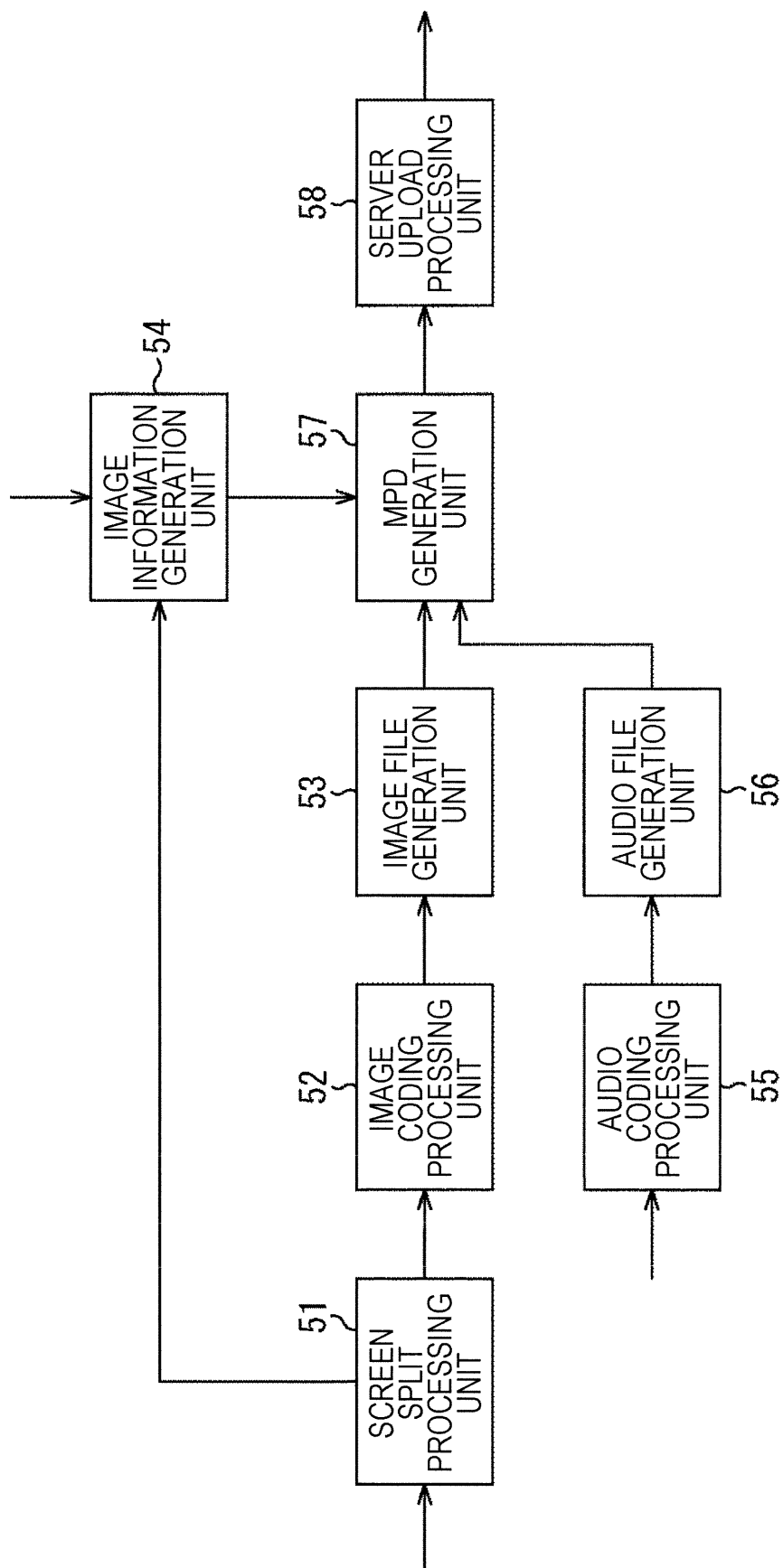


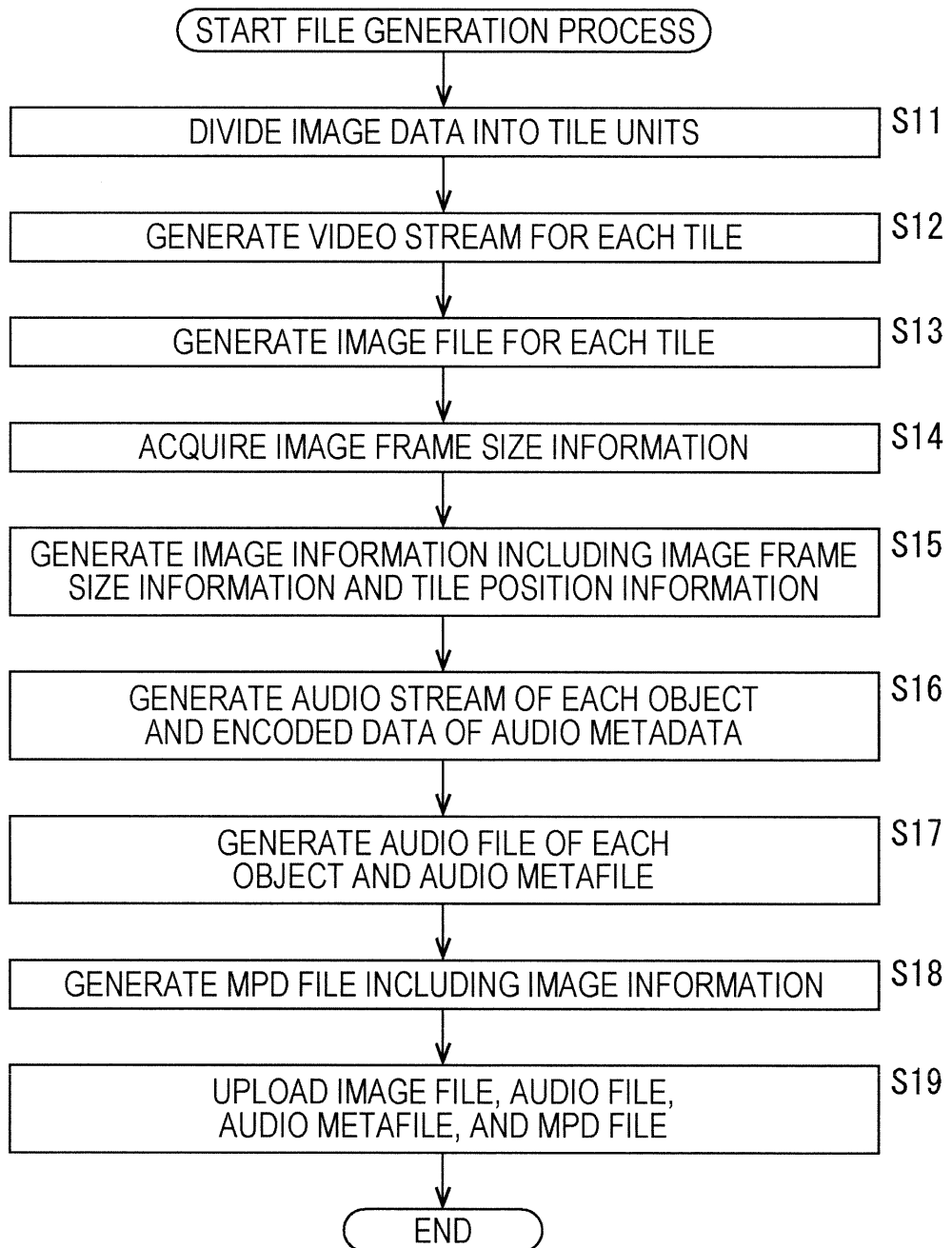
FIG. 12

FIG. 13

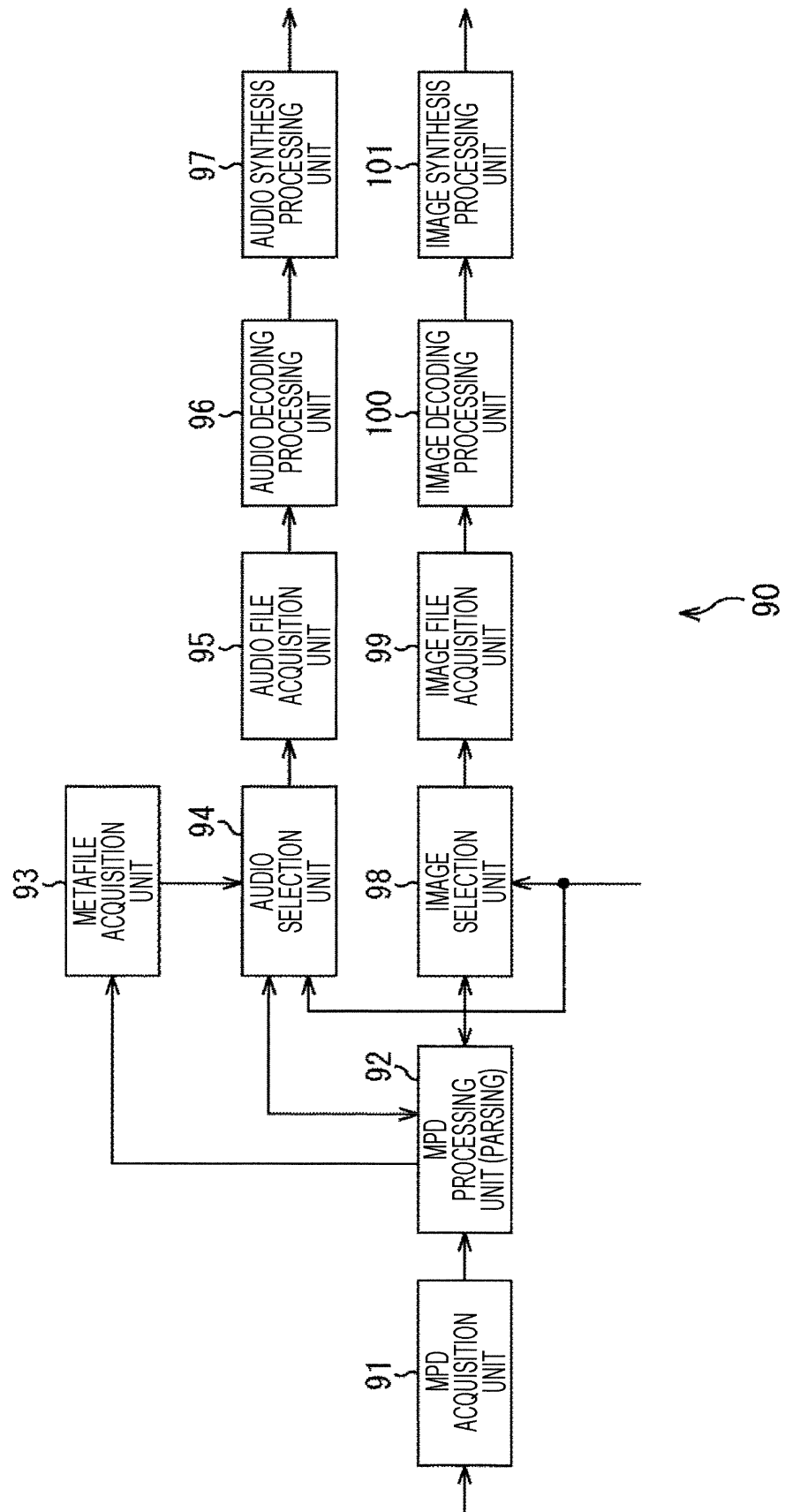


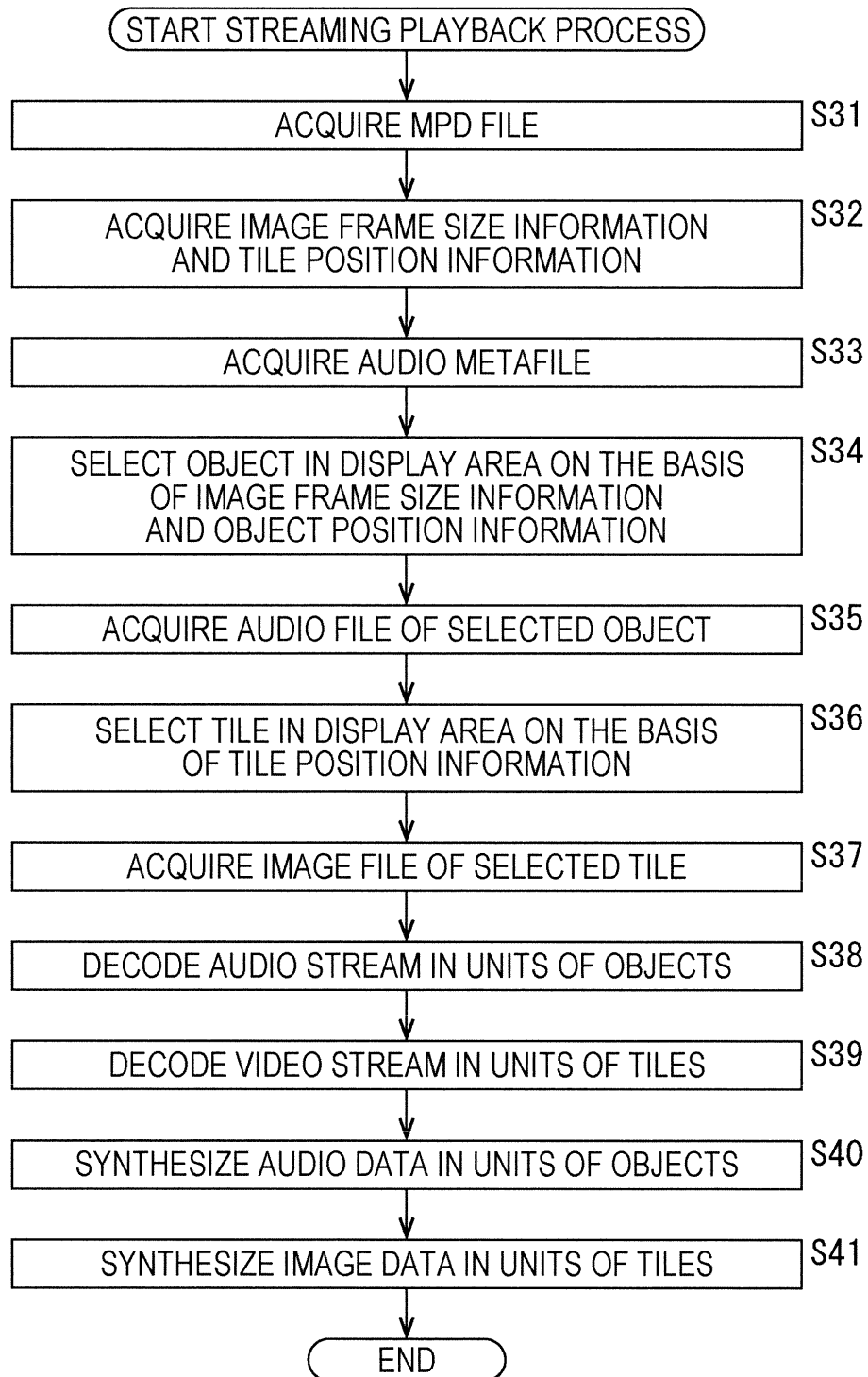
FIG. 14

FIG. 15

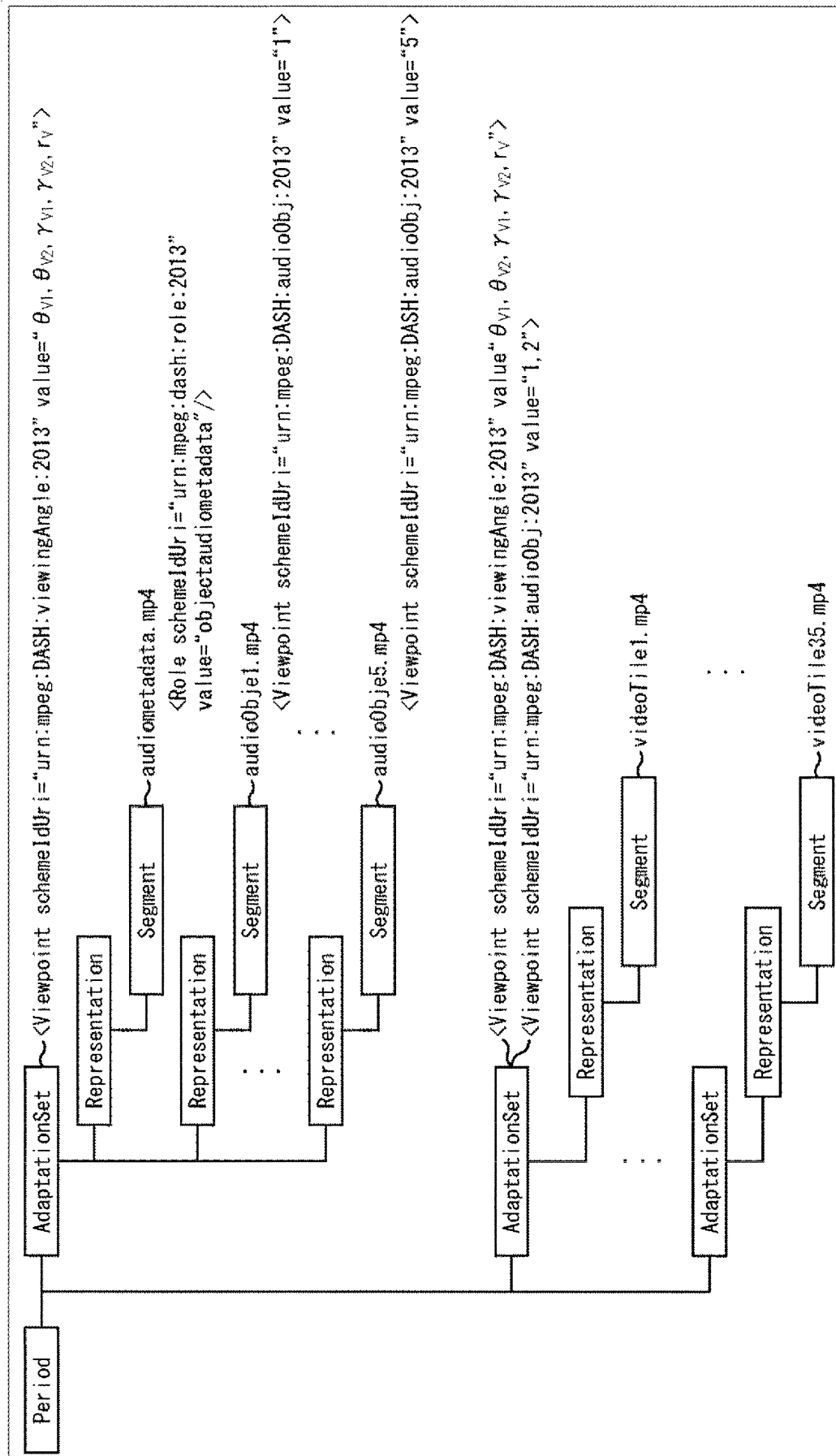


FIG. 16

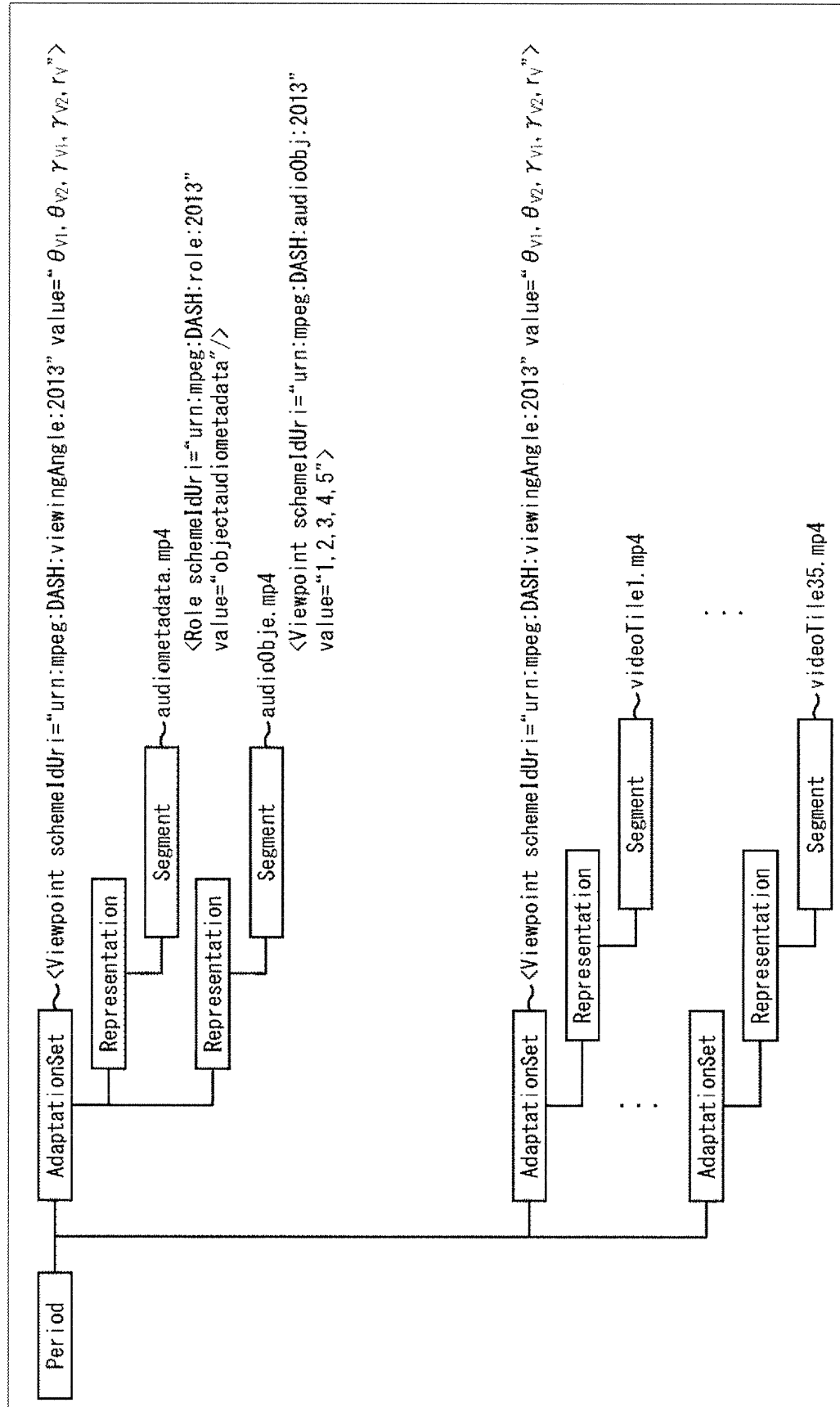


FIG. 17

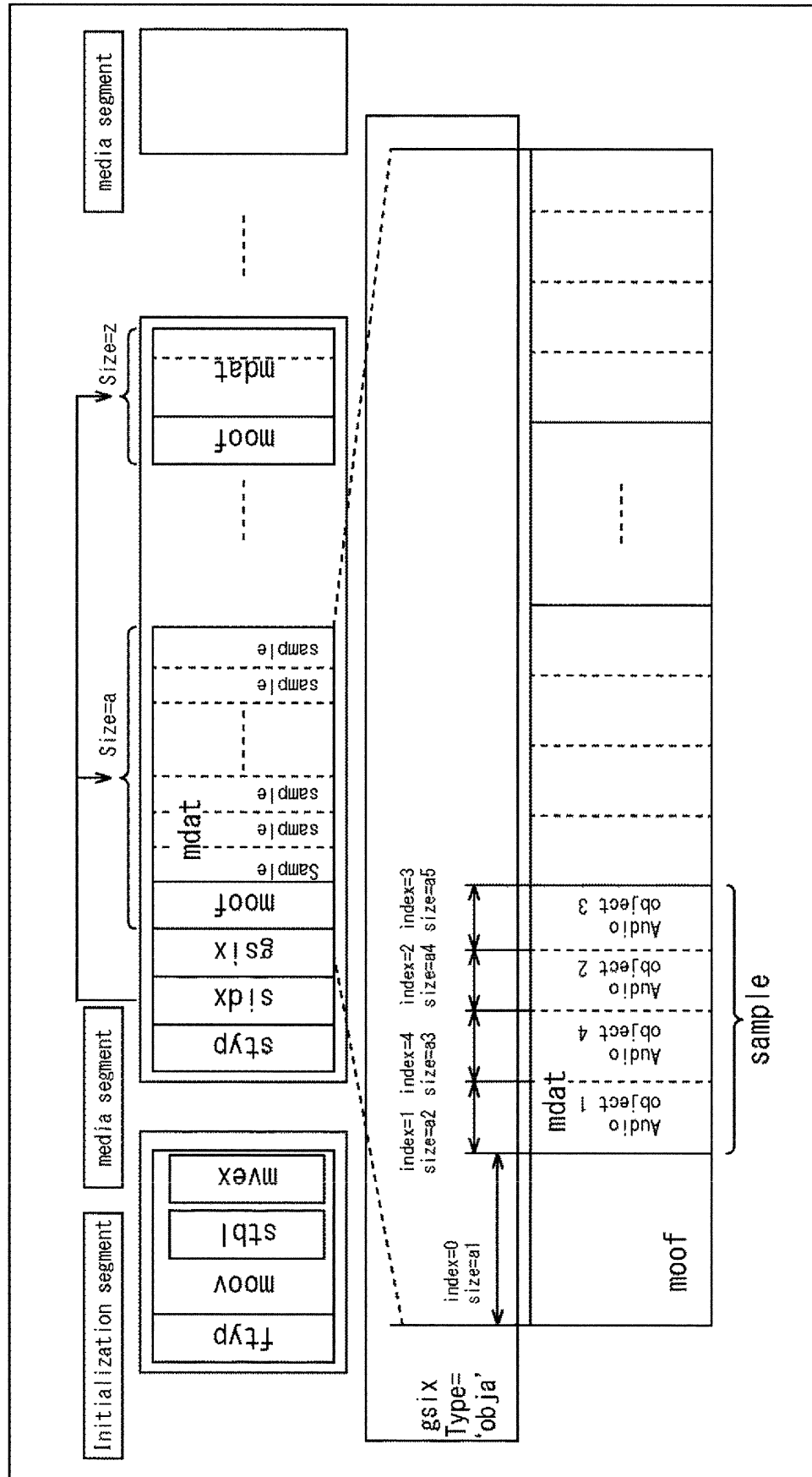


FIG. 18

```
aligned(8) class GeneralSubsegmentIndexBox extends FullBox('gsix', 0, 0) {
    grouping_type int(32);
    unsigned int(32) subsegment_count;
    for (i=1; i<=subsegment_count; i++)
    {
        unsigned int(32) entry_count;
        for (j=1; j<= entry_count; j++) {
            unsigned int(8) entry_index;
            unsigned int(24) range_size;
        }
    }
}
```

FIG. 19

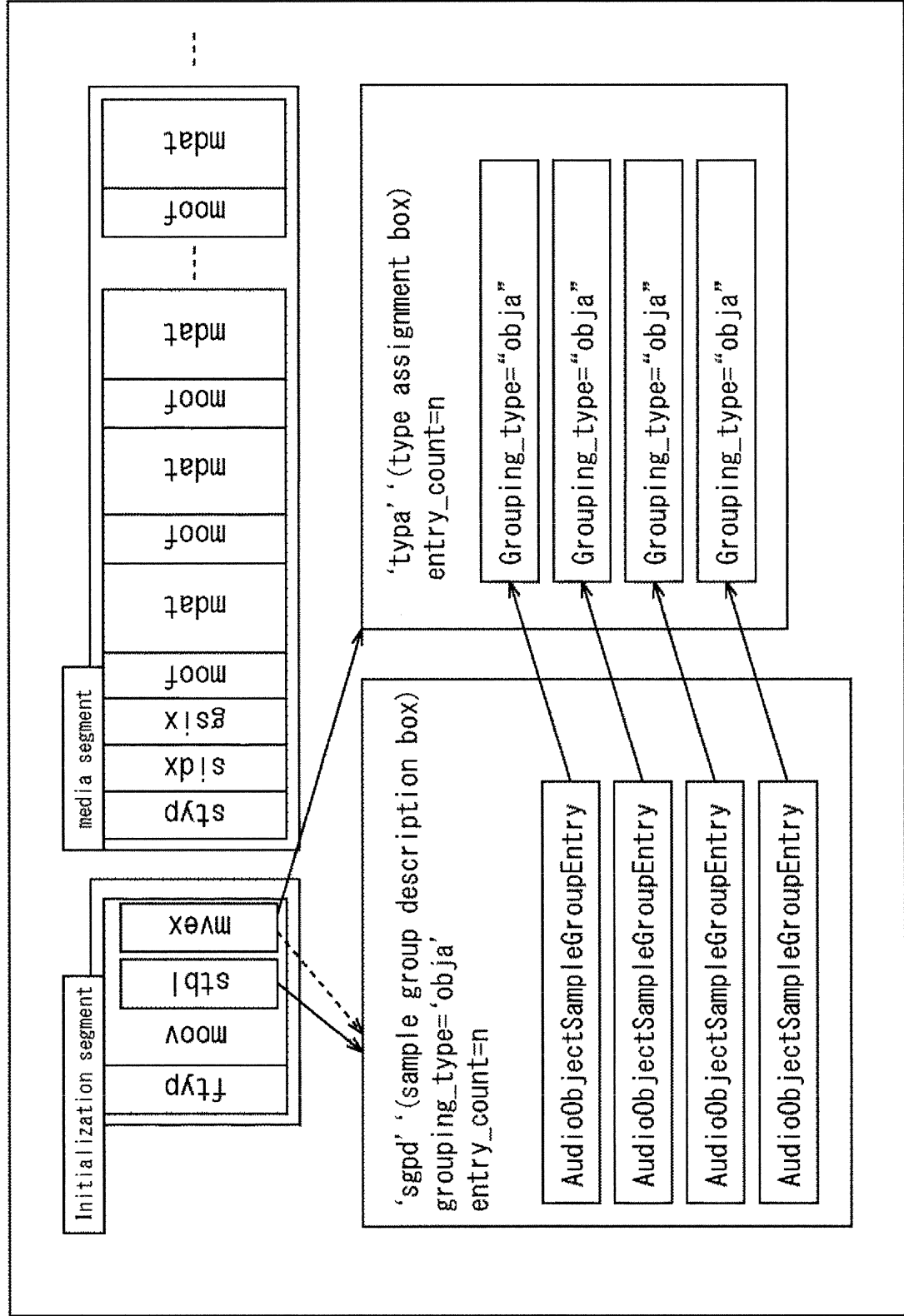


FIG. 20

```
Class AudioObjectSampleGroupEntry extends AudioSampleGroupEntry('obja') {  
    unsigned int (32) audio_object_id;  
    unsigned int (32) reserved=0;  
}
```

A

Sample Group Description Box ("sgpd")
Grouping_type="obja"
Entry_count=4

AudioObjectSampleGroupEntry("obja") "1"

AudioObjectSampleGroupEntry("obja") "2"

AudioObjectSampleGroupEntry("obja") "3"

AudioObjectSampleGroupEntry("obja") "4"

B

FIG. 21

```
aligned(8) class TypeAssignmentBox extends FullBox( 'typ*', 0, 0)
{
    unsigned int(8) entry_count;
    for (j=1; j<= entry_count; j++) {
        unsigned int(32) track_id;
        unsigned int(7) assignment_type;
        if(assignment_type == 0) {
            unsigned int(32) grouping_type;
        }
        else if (assignment_type == 1) {
            unsigned int(32) grouping_type;
            unsigned int(32) grouping_type_parameter;
        }
    }
    // other assignment_type values are reserved
}
```

FIG. 22

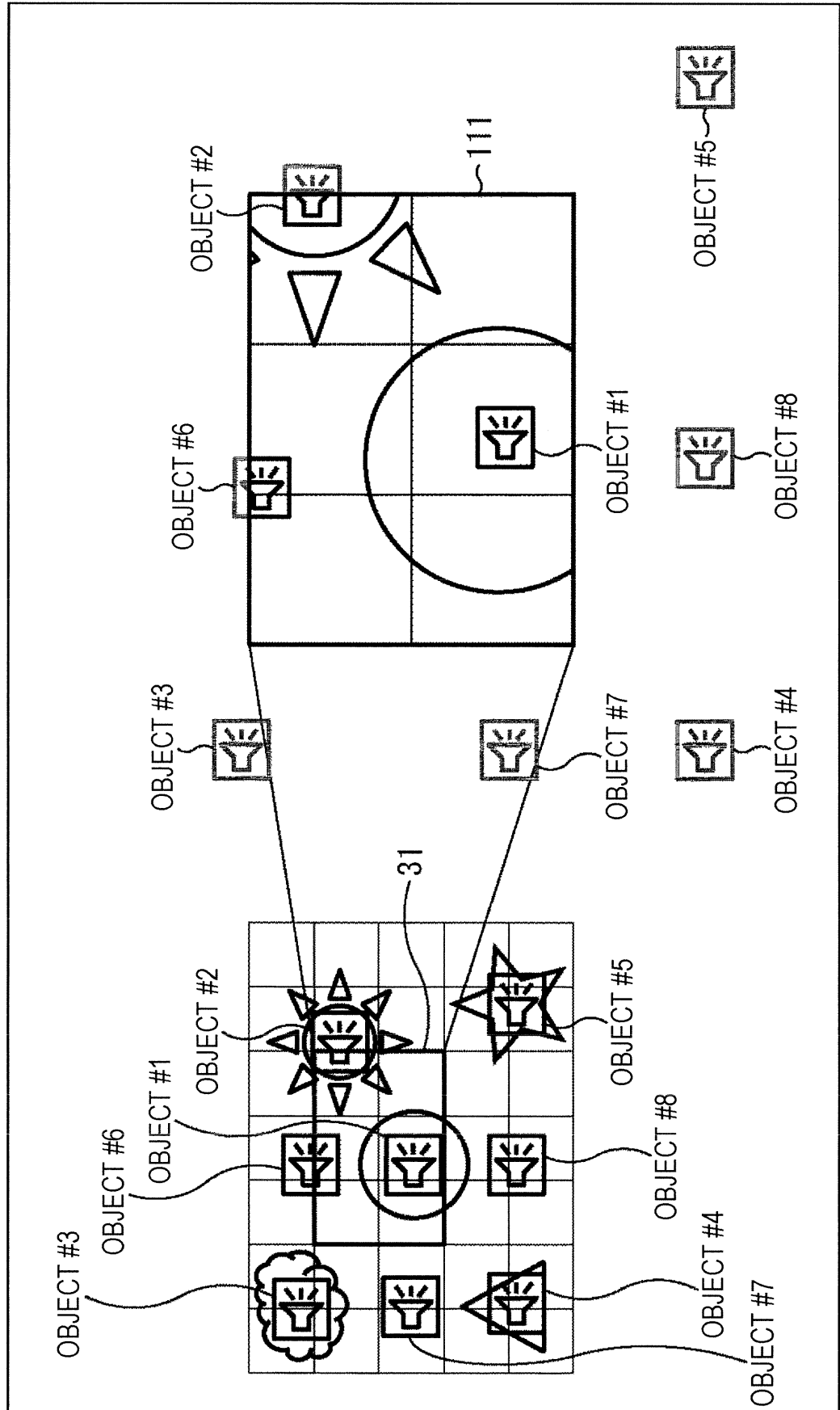


FIG. 23

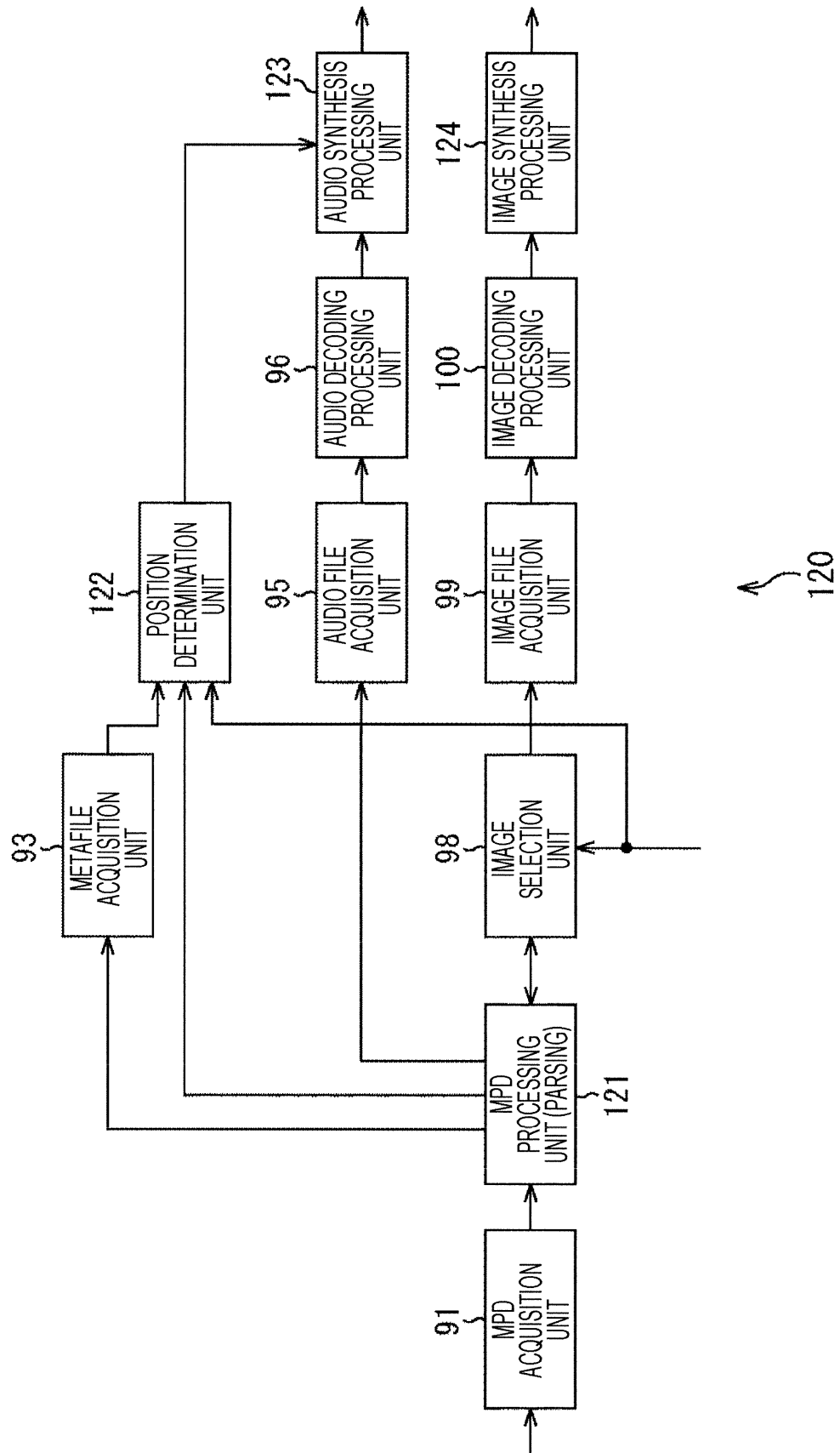


FIG. 24

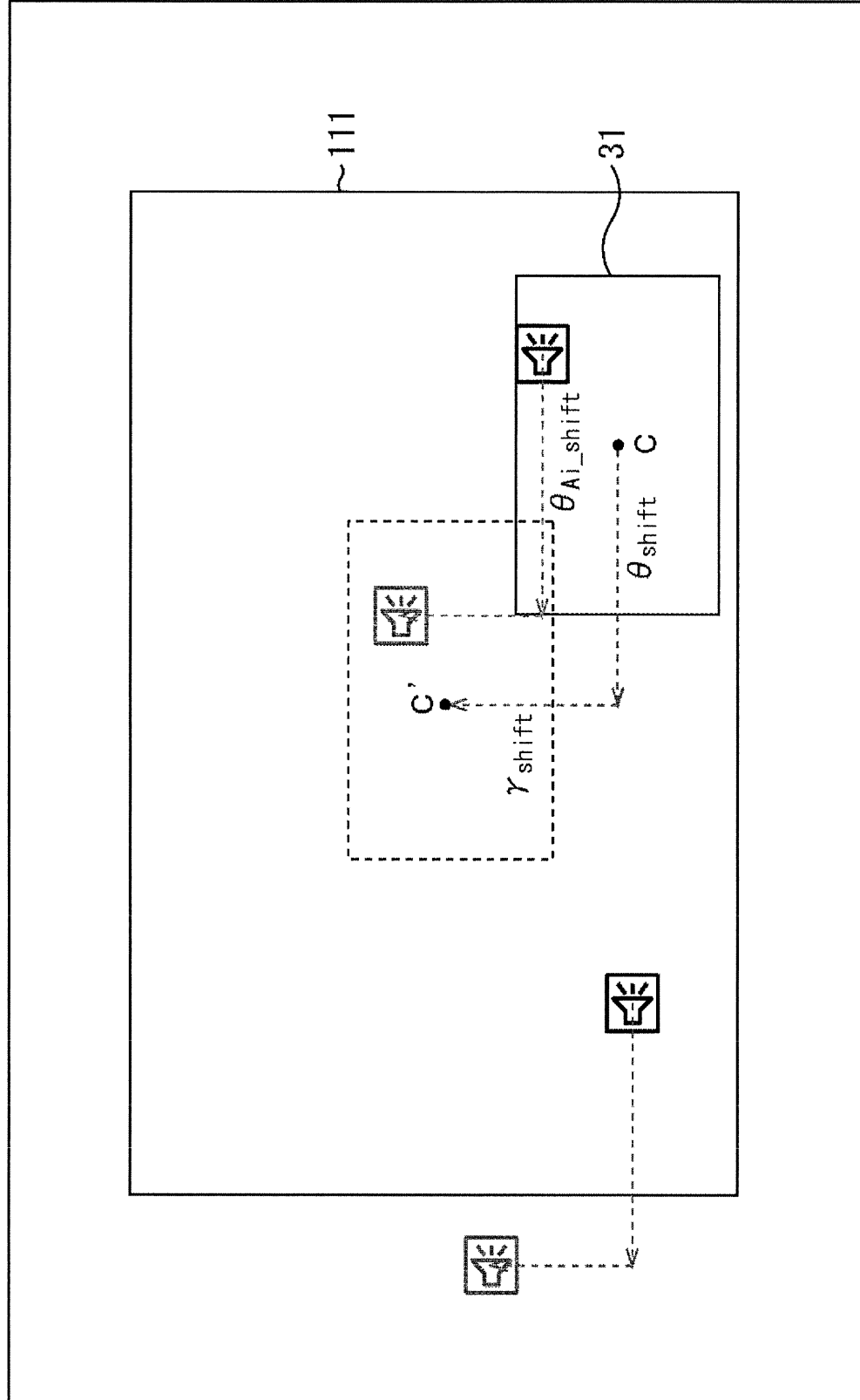


FIG. 25

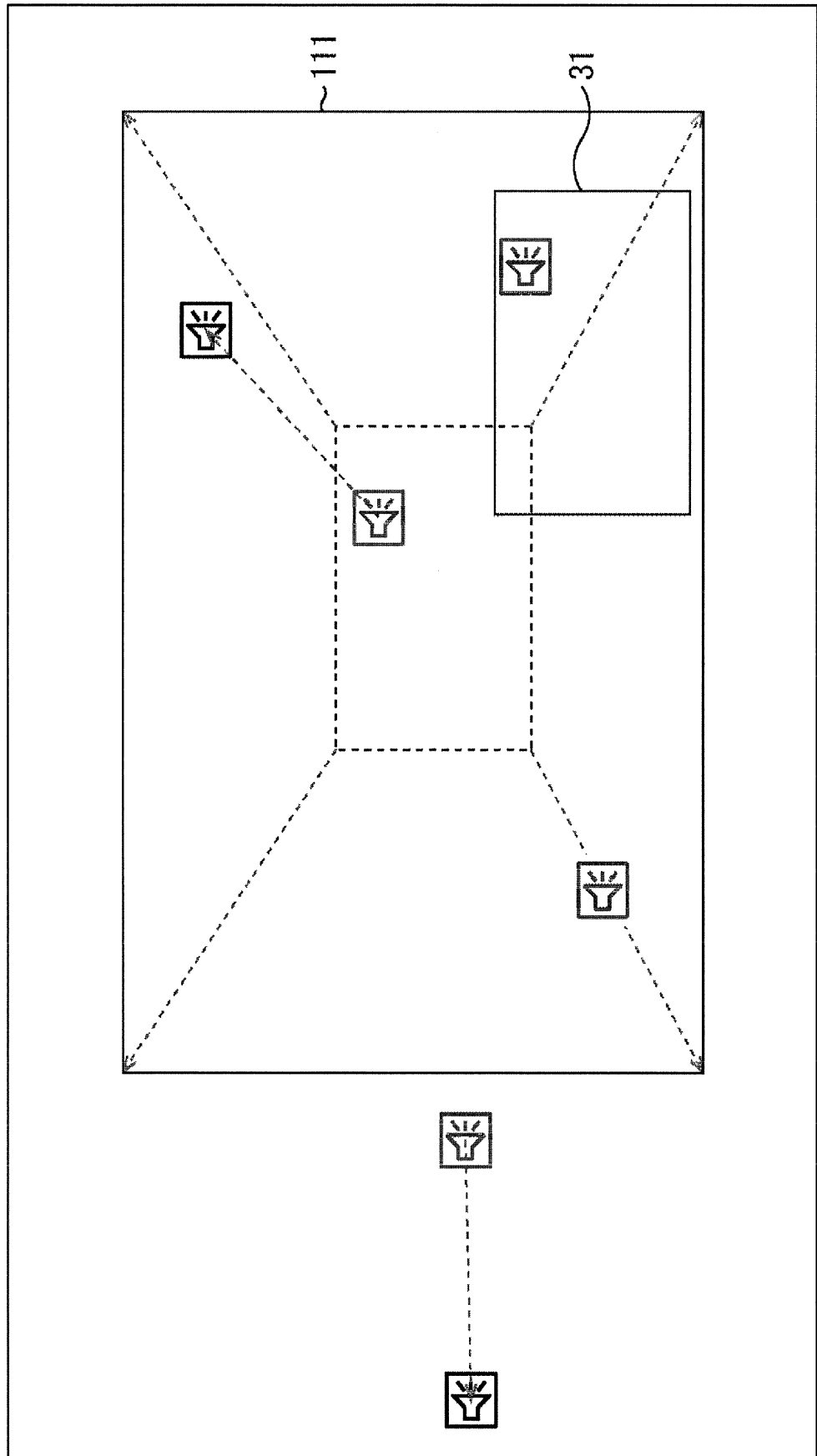


FIG. 26

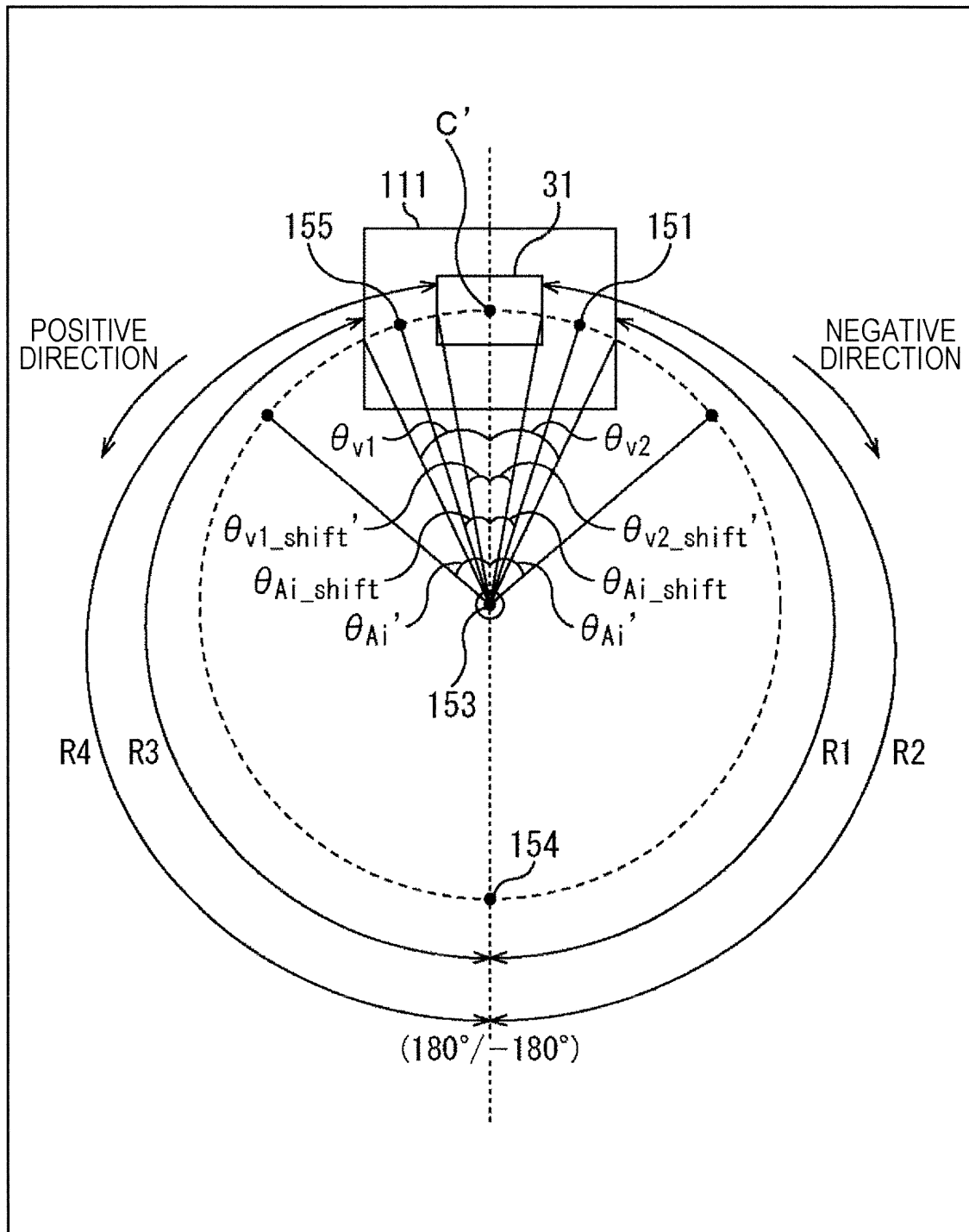


FIG. 27

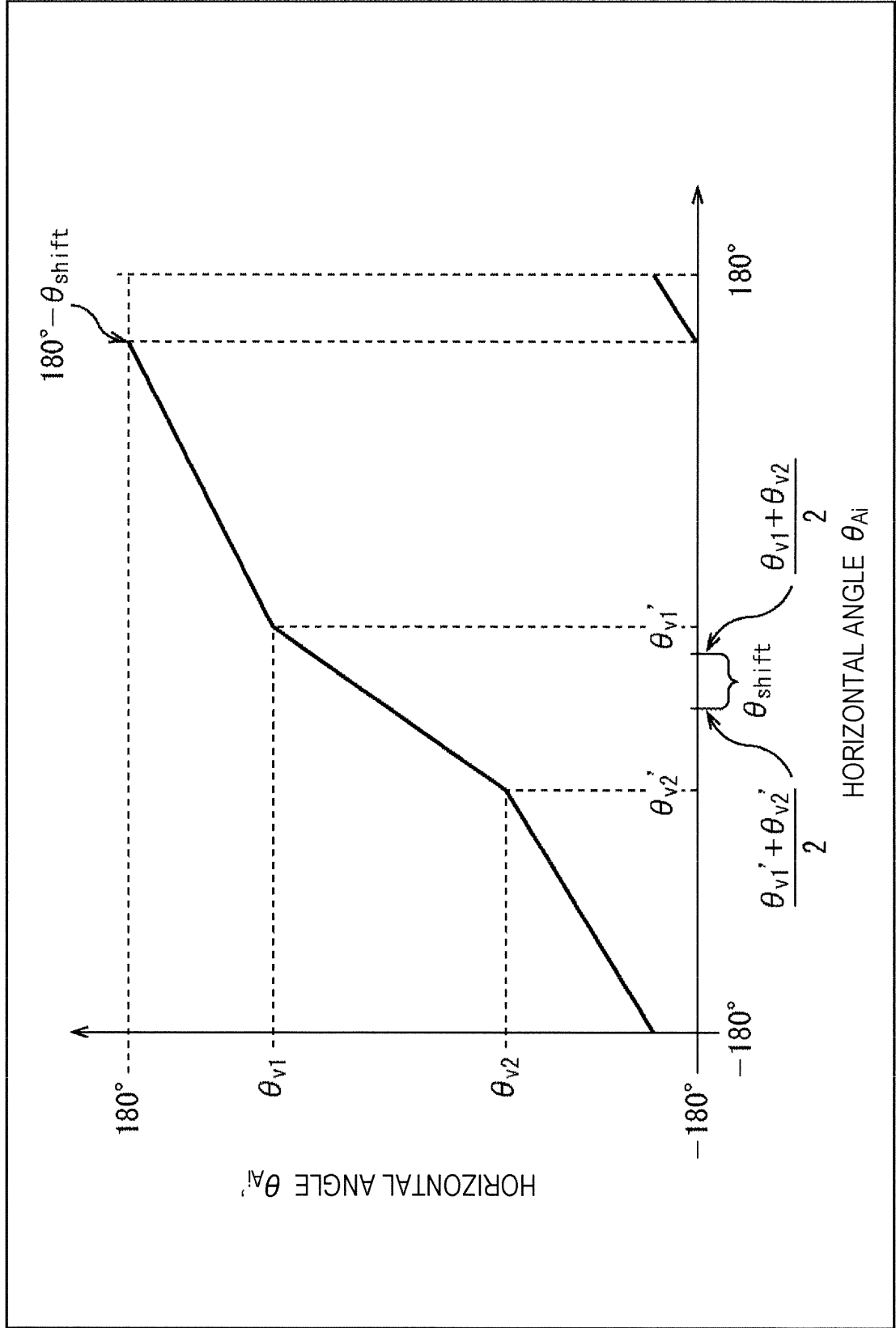


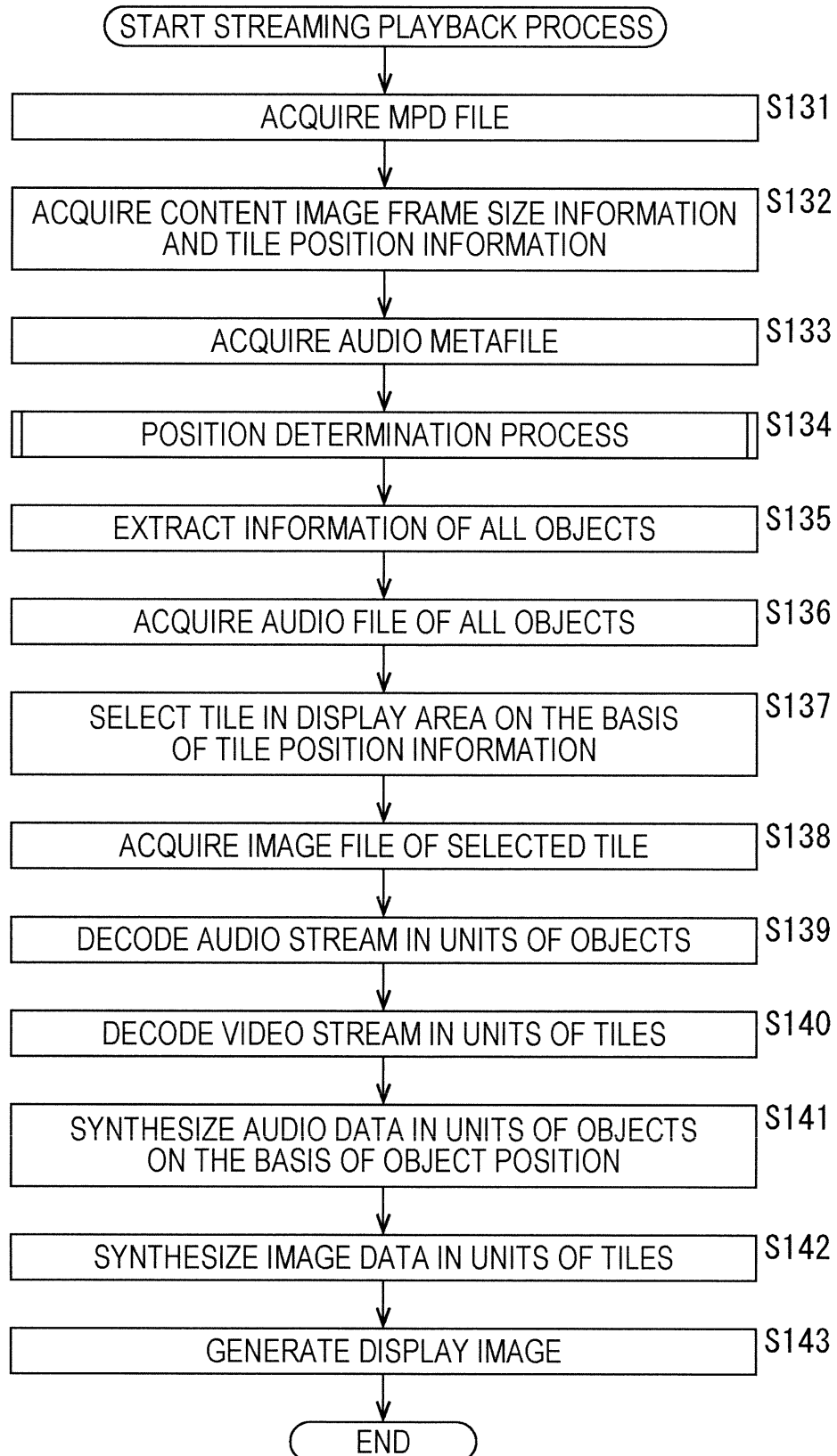
FIG. 28

FIG. 29

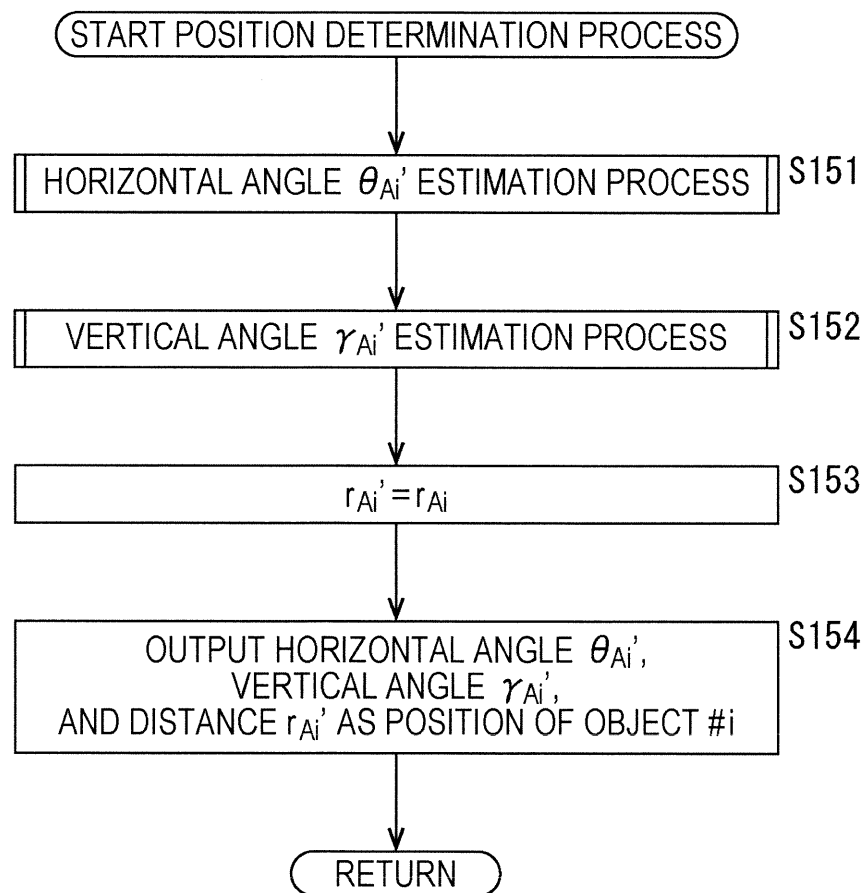


FIG. 30

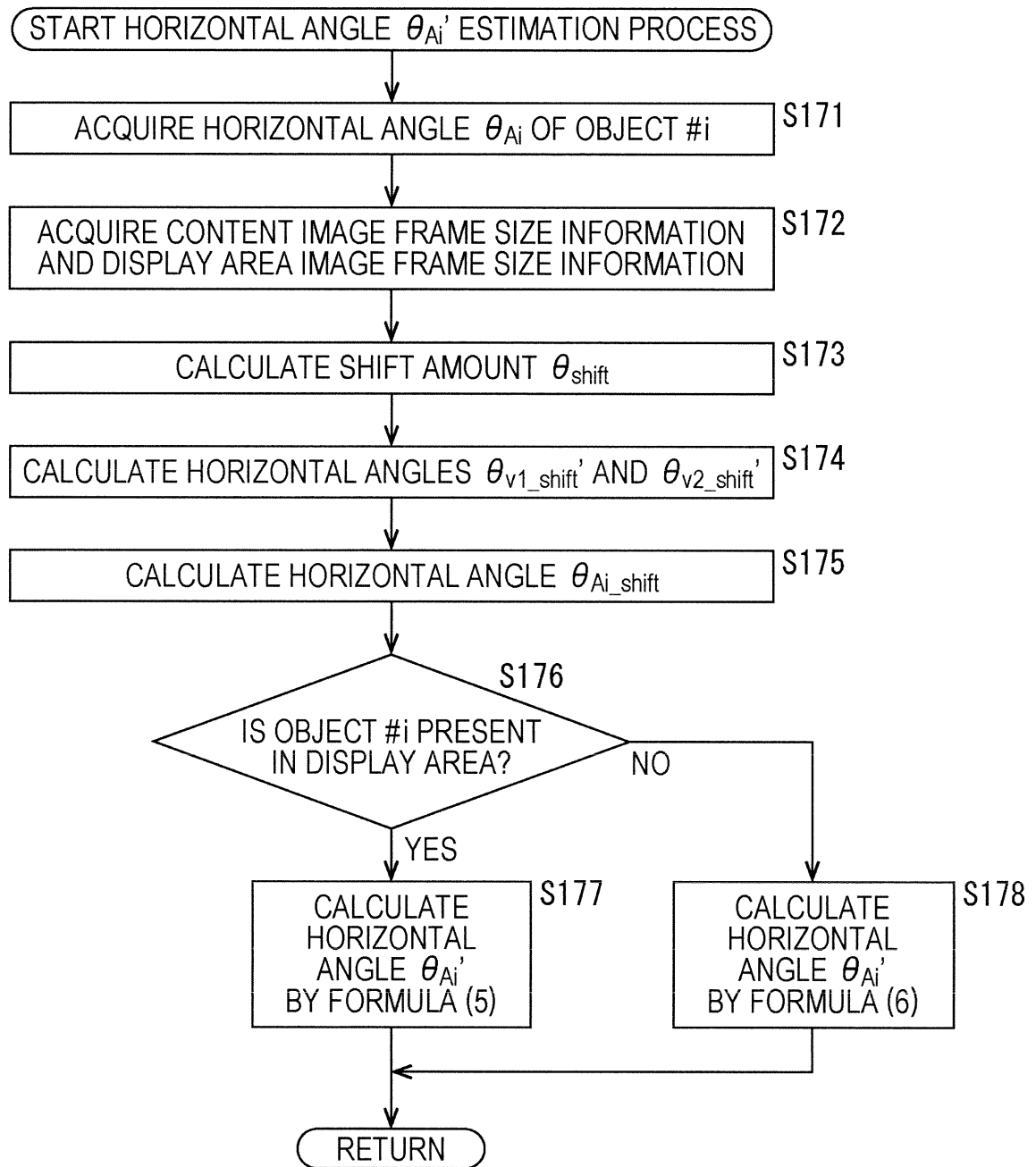


FIG. 31

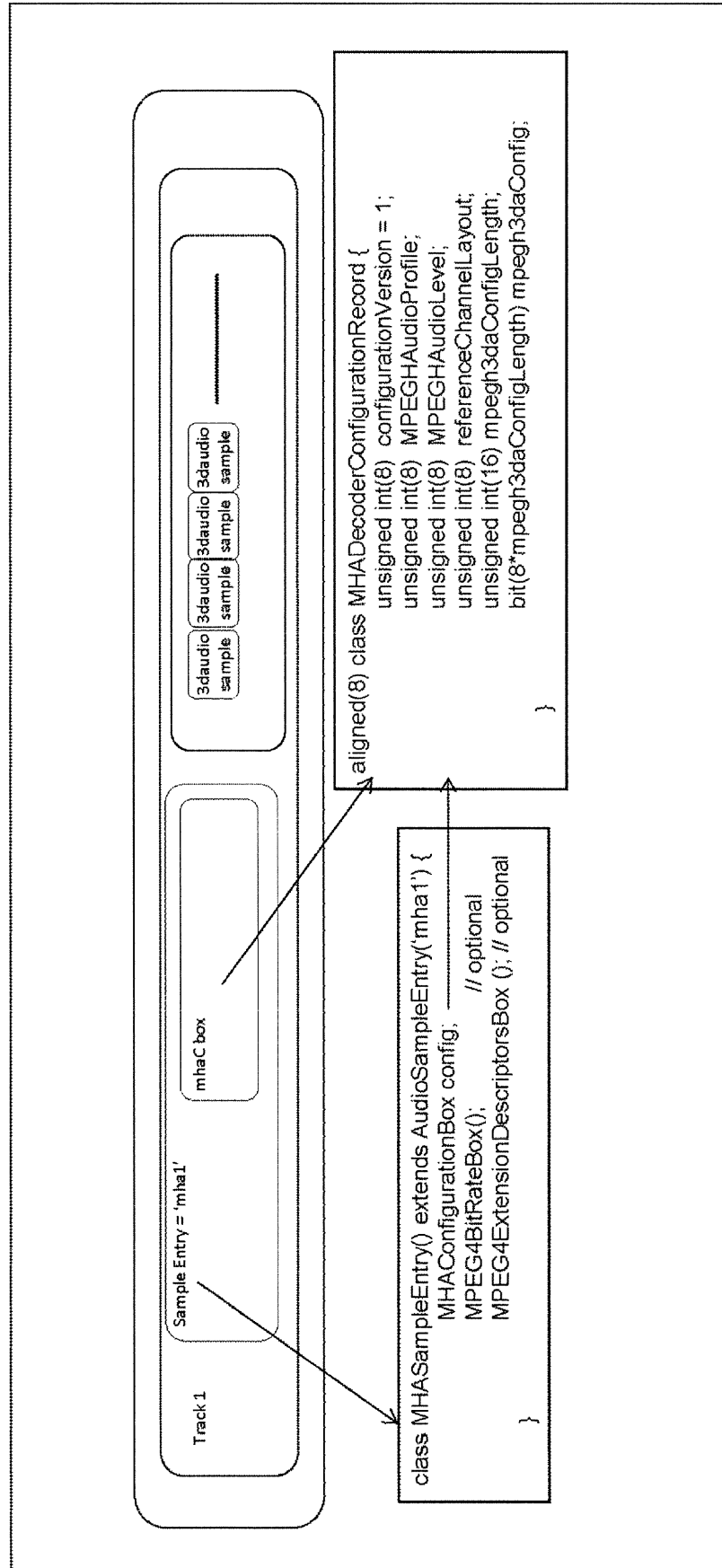


FIG. 32

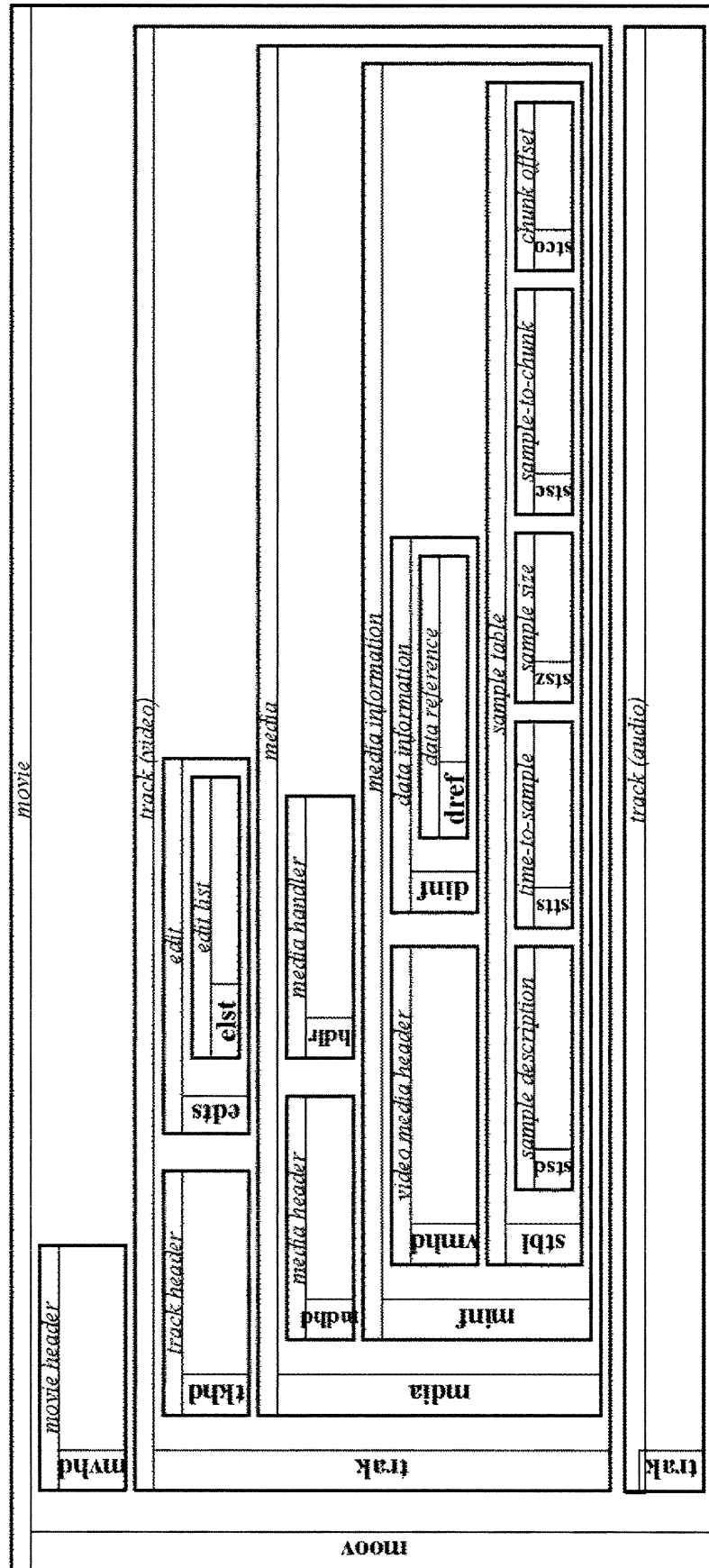


FIG. 33

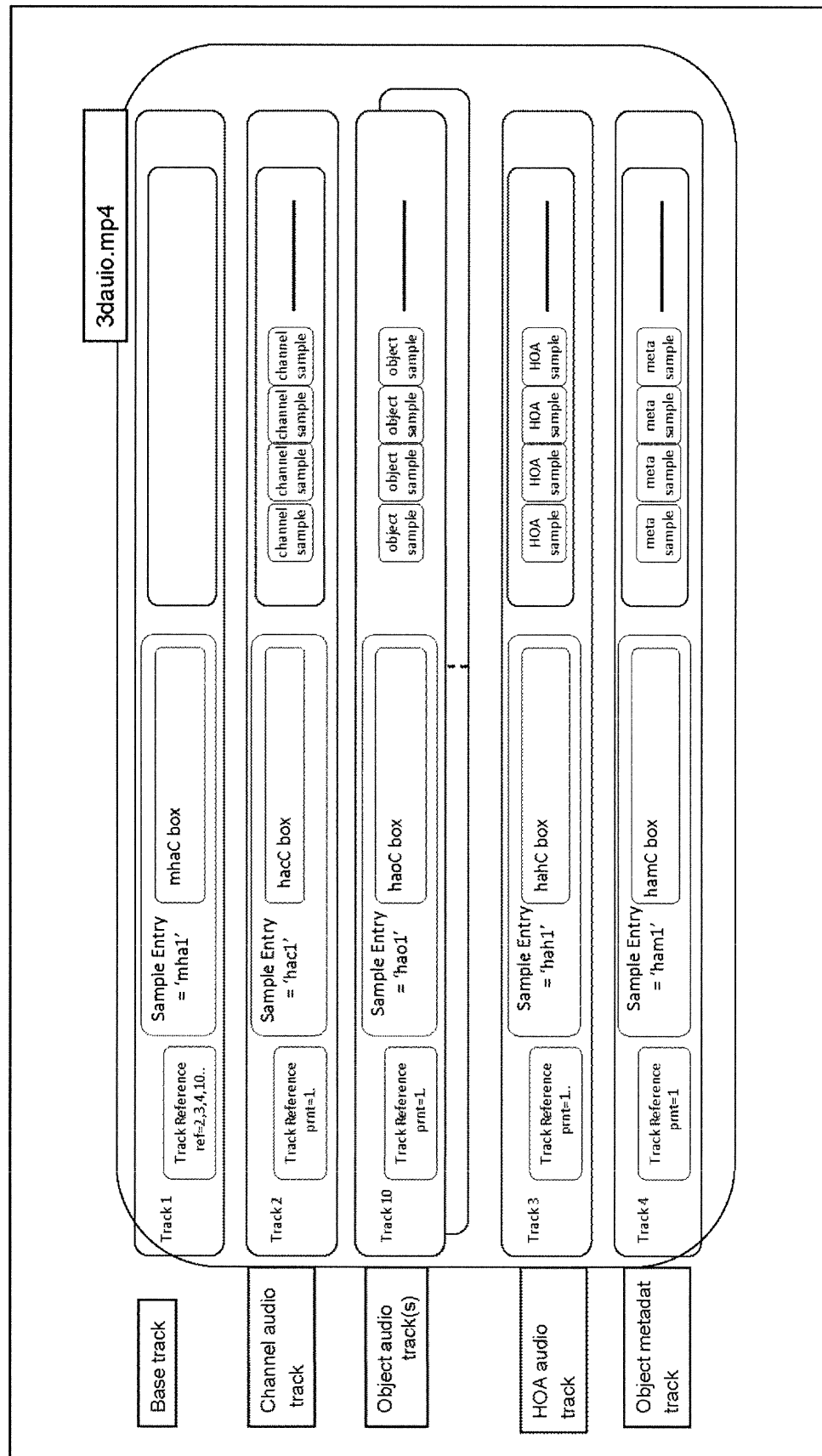


FIG. 34

```

class MHASampleEntry() extends AudioSampleEntry('mha1') {
    MHAConfigurationBox config; //optional
    MHAViewingAngleBox config; // optional
    MPEG4BitRateBox();
    MPEG4ExtensionDescriptorsBox (); // optional
}

class MHAConfigurationBox() extends Box('mhaC') {
    MHADecoderConfigurationRecord() MHAConfig;
}

aligned(8) class MHADecoderConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(8) MPEG4AudioProfile;
    unsigned int(8) MPEG4AudioLevel;
    unsigned int(8) referenceChannelLayout;
    unsigned int(16) mpeg3daConfigLength;
    bit(8'mpeg3daConfigLength) mpeg3daConfig;
}

class MHAViewingAngleConfigurationBox() extends Box('mhaV') {
    MHAViewingAngleConfigurationRecord() MHAVConfig;
}

aligned(8) class MHAViewingAngleConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(16) width;
    unsigned int(16) height;
    unsigned int(16) theta1;
    unsigned int(16) theta2;
    unsigned int(16) gamma1;
    unsigned int(16) gamma2;
}

```

ASSUMPTION FOR CALCULATION OF COORDINATES OF THIS track
 WidthxHeight INFORMATION USING pixels OF Video,
 LIKELIHOOD INFORMATION IN VERTICAL AND HORIZONTAL DIRECTIONS OF Audio
 => ON THE BASIS OF THESE PIECES OF INFORMATION,
 audio IS MAPPED TO VIDEO pixel COORDINATES

FIG. 35

```

class MHACChannelAudioSampleEntry() extends AudioSampleEntry('hac1') {
    MHACChannelAudioConfigurationBox config;
    MPEG4BitRateBox(); // optional
    MPEG4ExtensionDescriptorsBox (); // optional
}

class MHACChannelAudioConfigurationBox() extends Box('hacC') {
    MHACChannelAudioDecoderConfigurationRecord() MHACACConfig;
}

aligned(8) class MHACChannelAudioDecoderConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(8) MPEGHAudioProfile;
    unsigned int(8) MPEGHAudioLevel;
}

```

Profile/Level INFORMATION NECESSARY
FOR decoding Channel Audio

FIG. 36

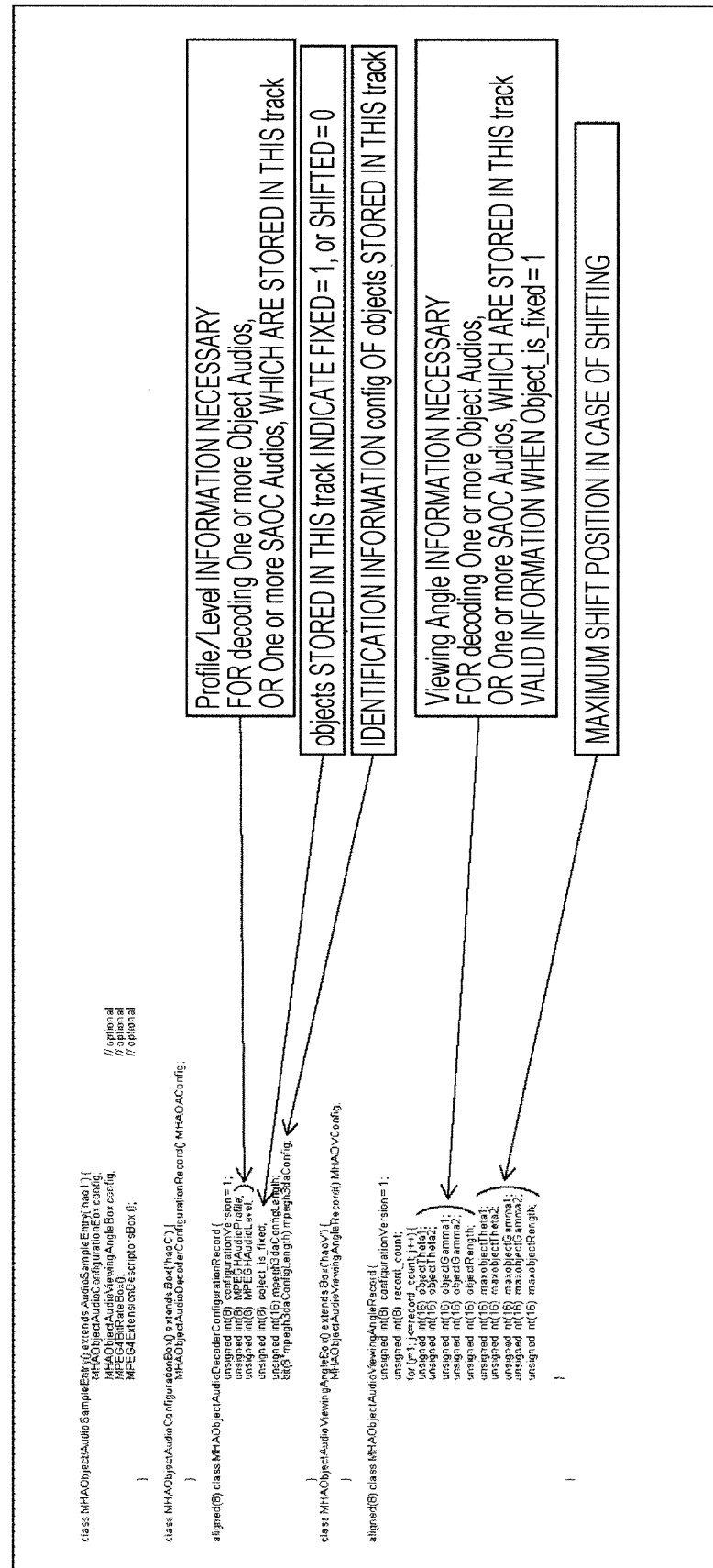


FIG. 37

```

class MHAHOAAudioSampleEntry() extends AudioSampleEntry('hah1') {
    MHAHOAAudioConfigurationBox config;
    MPEG4BitRateBox();           // optional
    MPEG4ExtensionDescriptorsBox (); // optional
}

class MHAHOAAudioConfigurationBox() extends Box('hahC') {
    MHAHOAAudioDecoderConfigurationRecord() MHAHAConfig;
}

aligned(8) class MHAHOAtAudioDecoderConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(8) MPEGHAudioProfile;
    unsigned int(8) MPEGHAudioLevel;
}

```

Profile/Level INFORMATION
NECESSARY FOR HOA audio
STORED IN THIS track

FIG. 38

```

class MHAMetadataSampleEntry() extends AudioSampleEntry('ham1') {
    MHAMetadataConfigurationBox config;
    MPEG4BitRateBox();           // optional
    MPEG4ExtensionDescriptorsBox (); // optional
}

class MHAMetadataConfigurationBox() extends Box('hahC') {
    MHAMetadataConfigurationRecord() MHAMConfig;
}

aligned(8) class MHAMetadataConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(16) reserved
}

```

FIG. 39

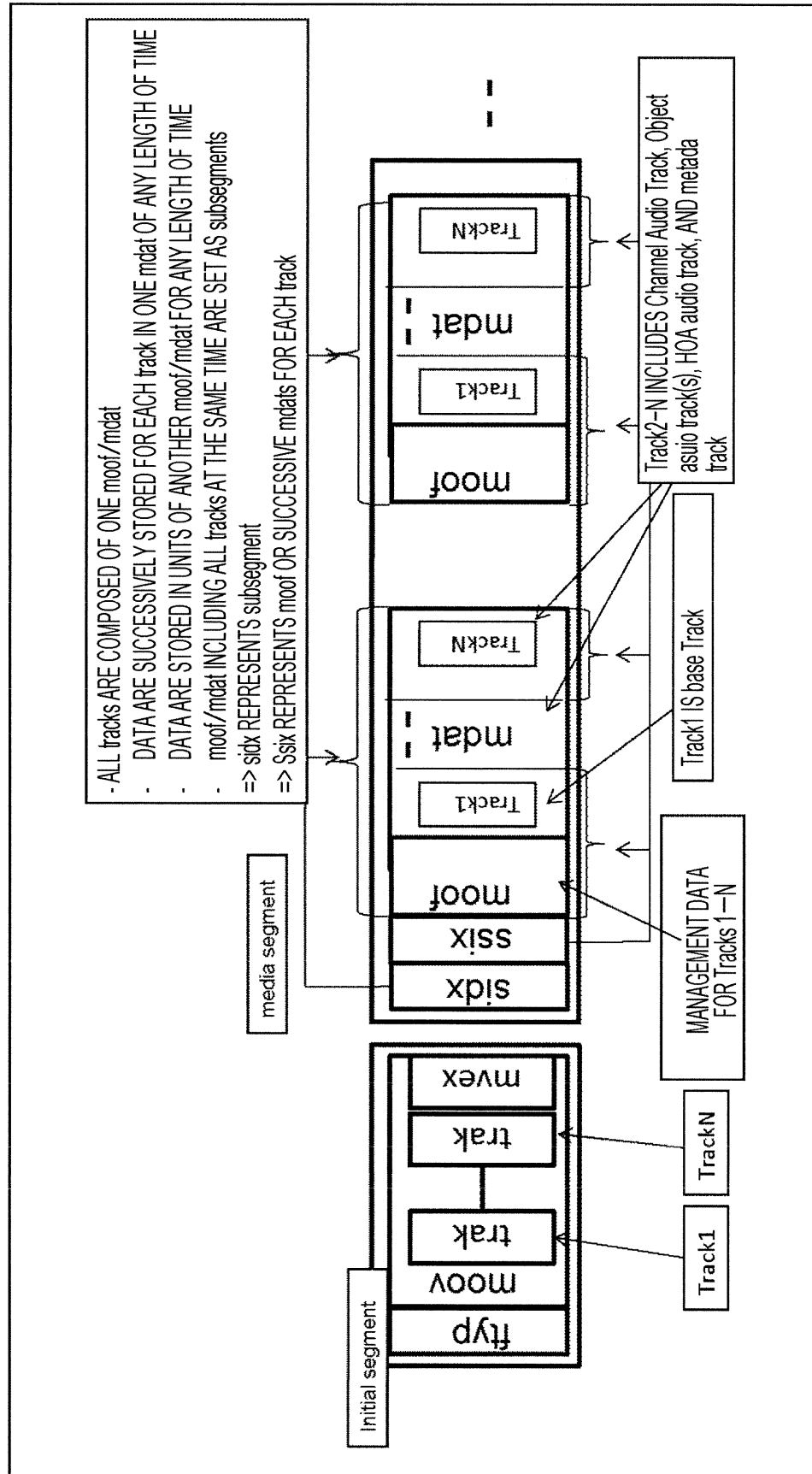


FIG. 40

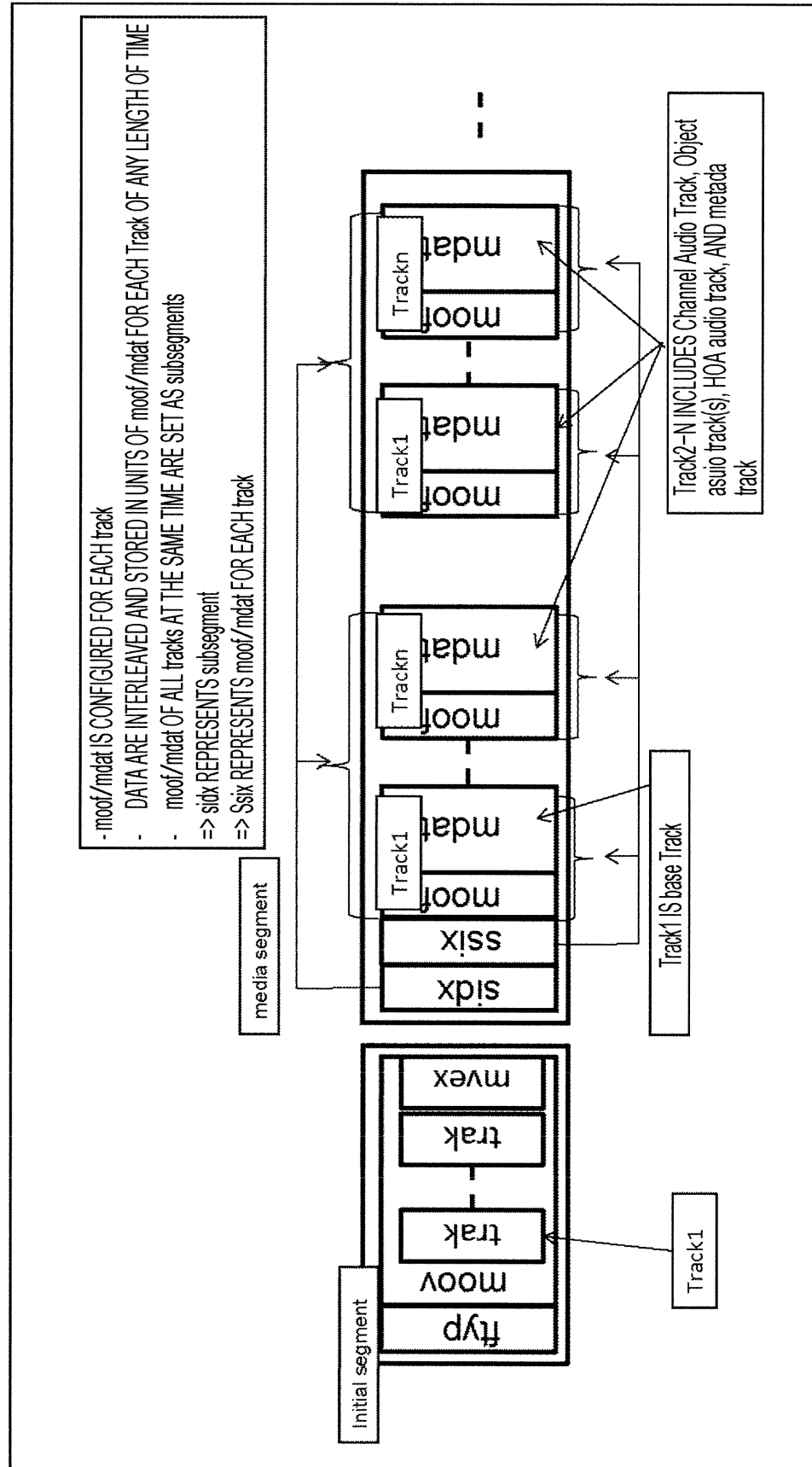


FIG. 41

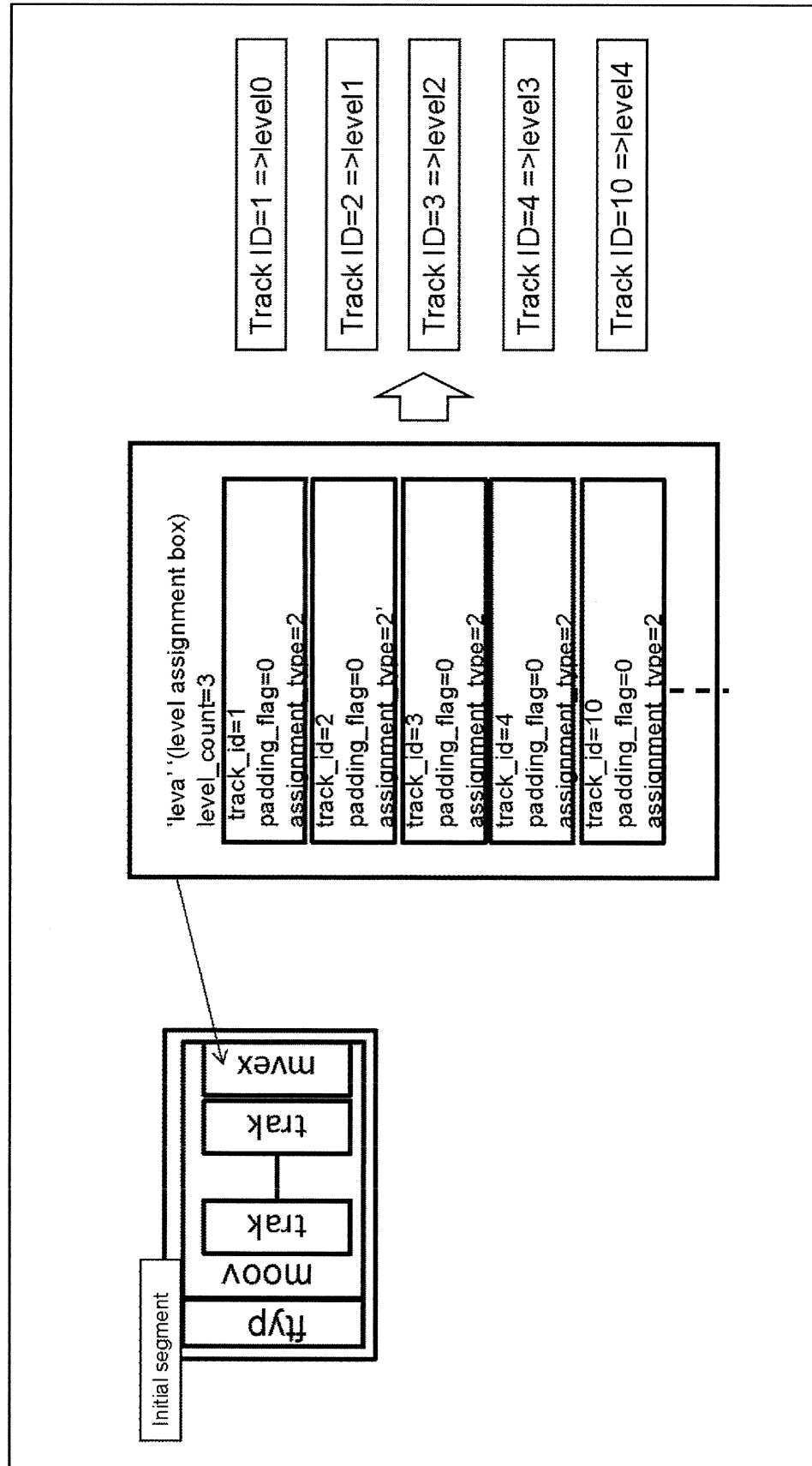


FIG. 42

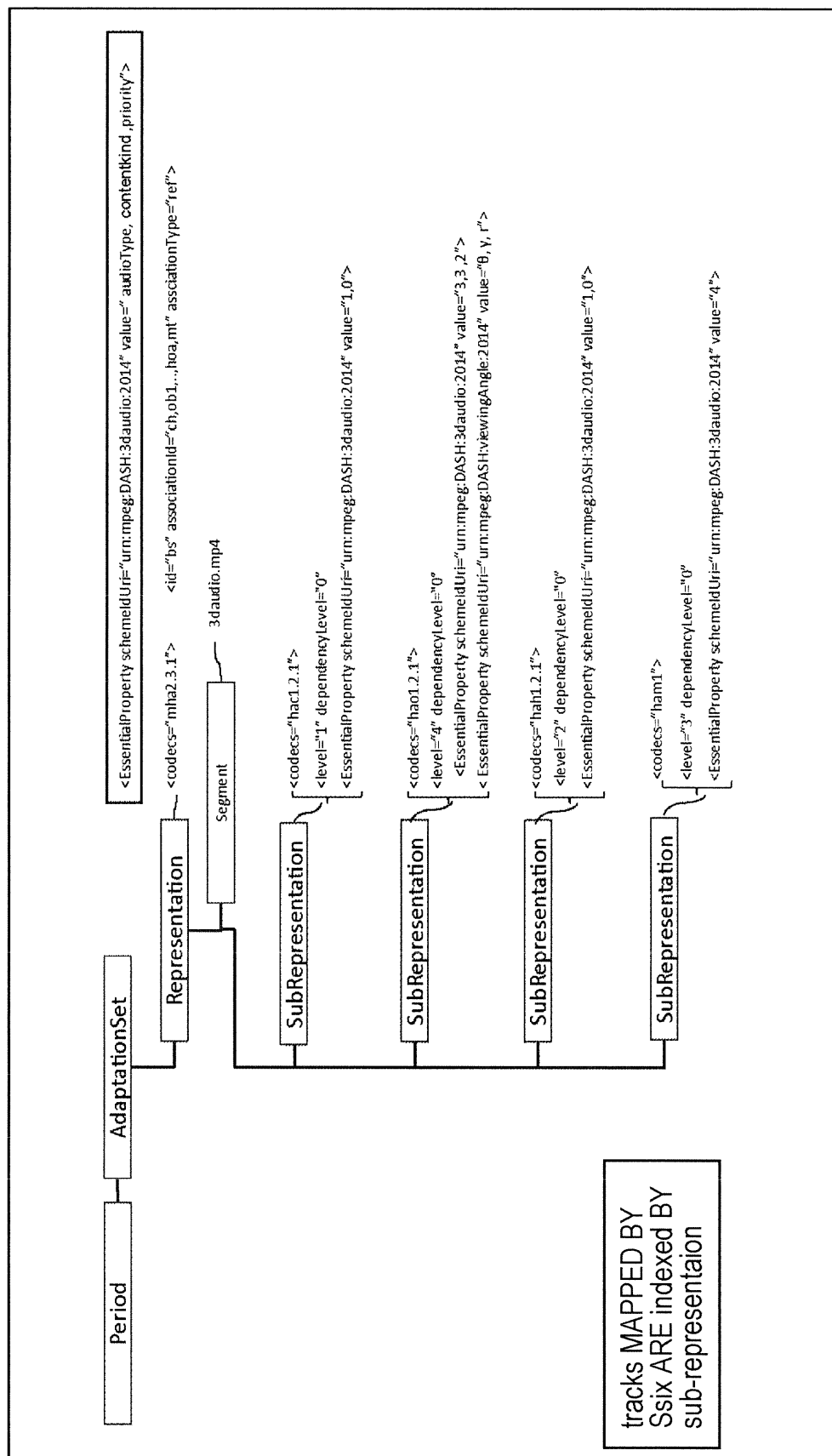


FIG. 43

AudioType
1 channel audio
2 HOA audio
3 object audio
4 metadata

Priority IS
DEFINED BY 23008-3

contentKind	description
0	undefined
1	complete main
2	dialogue
3	music
4	effect
5	mixed
6	LFE
7	voiceover
8	spokensubtitle
9	audiodescription/visually impaired
10	commentary
11	hearing impaired
12	emergency
13-15	reserved

FIG. 44

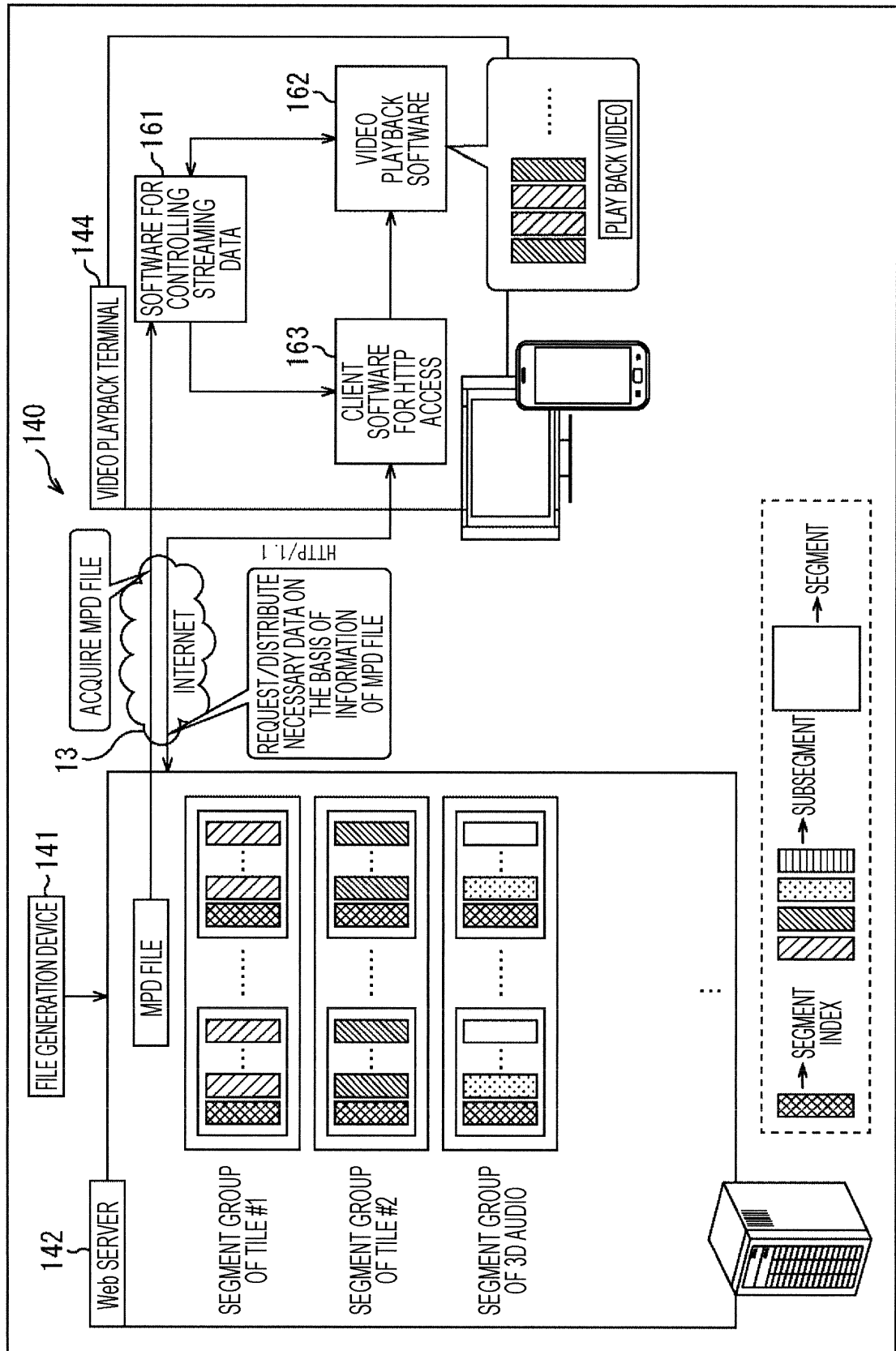


FIG. 45

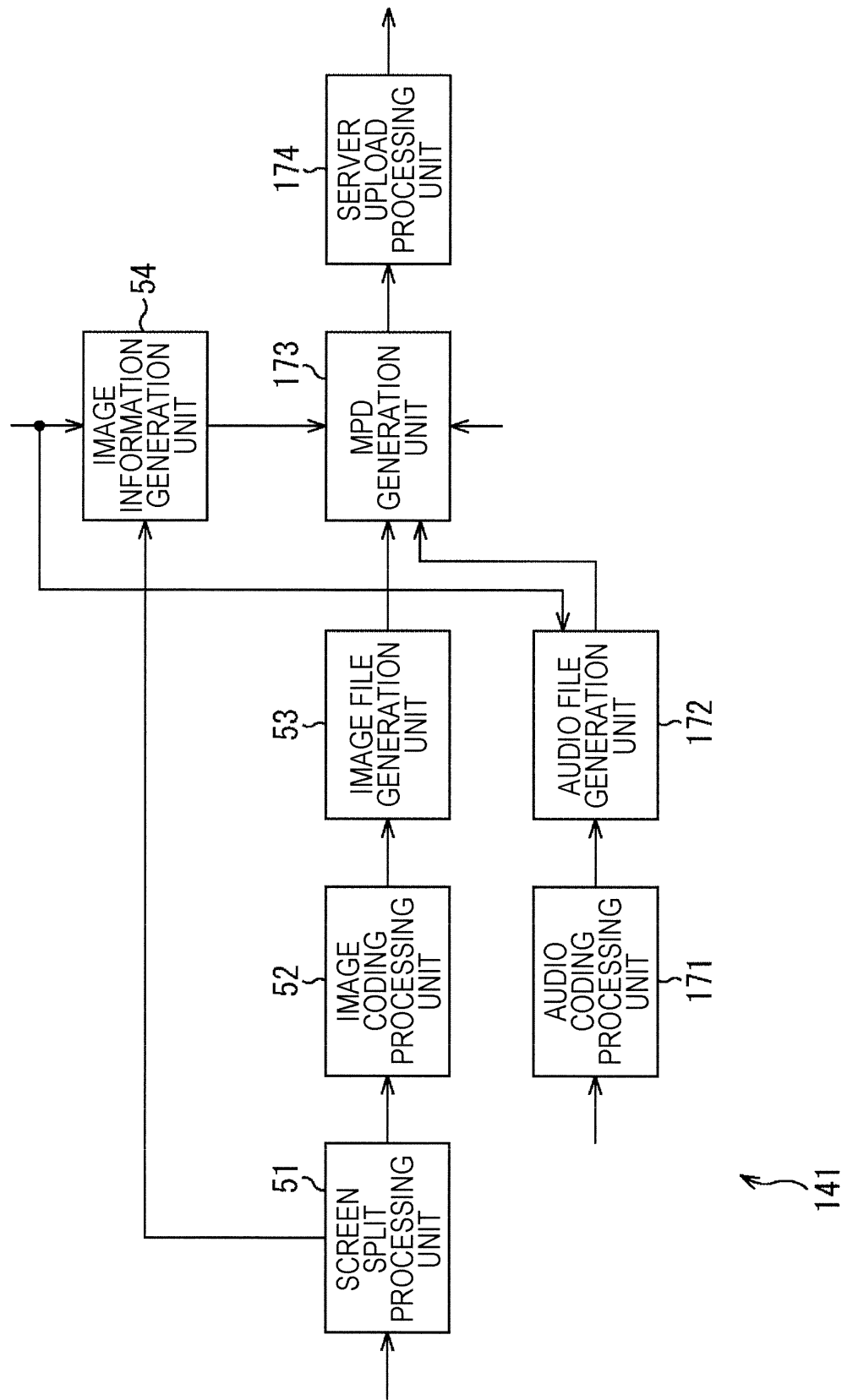


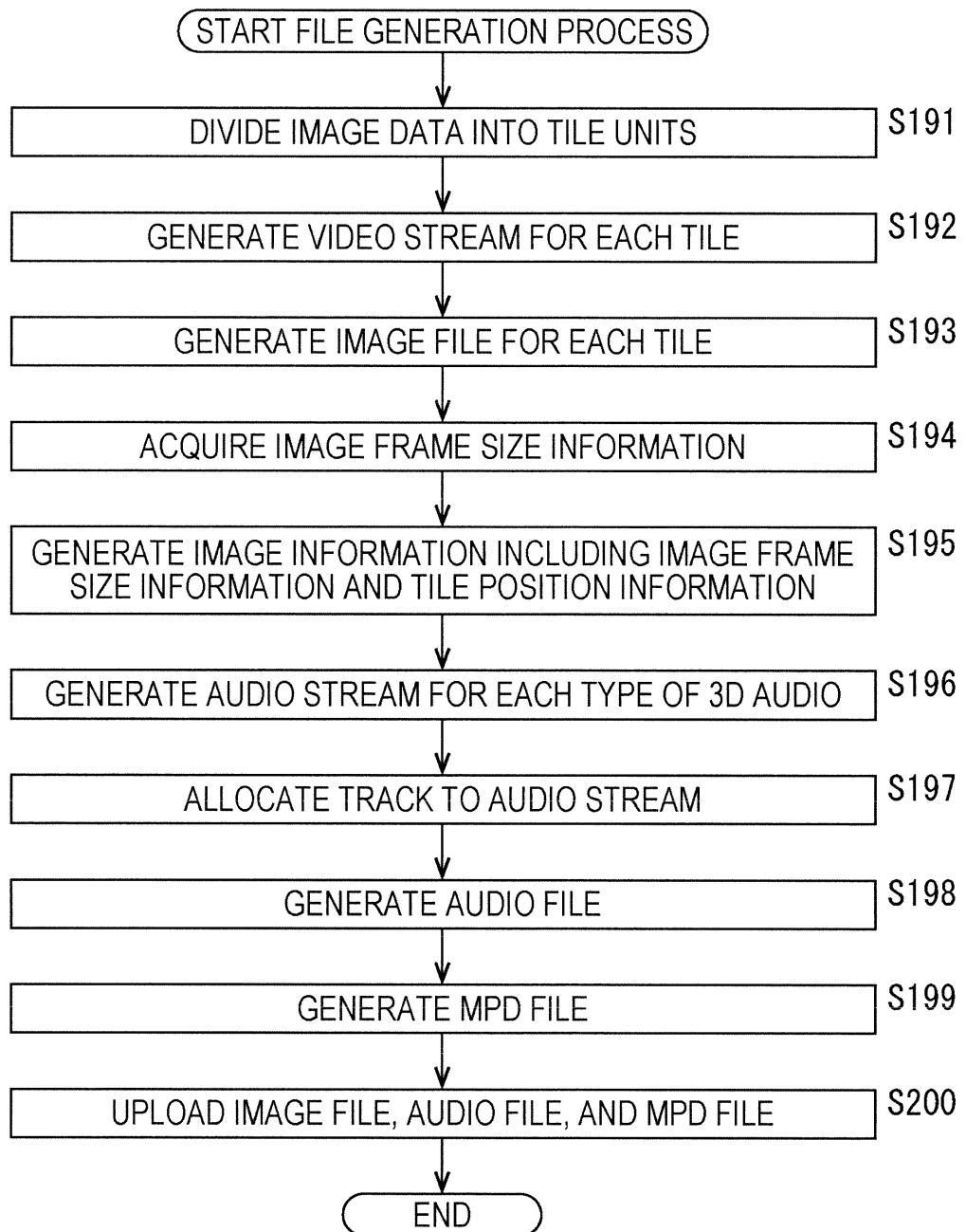
FIG. 46

FIG. 47

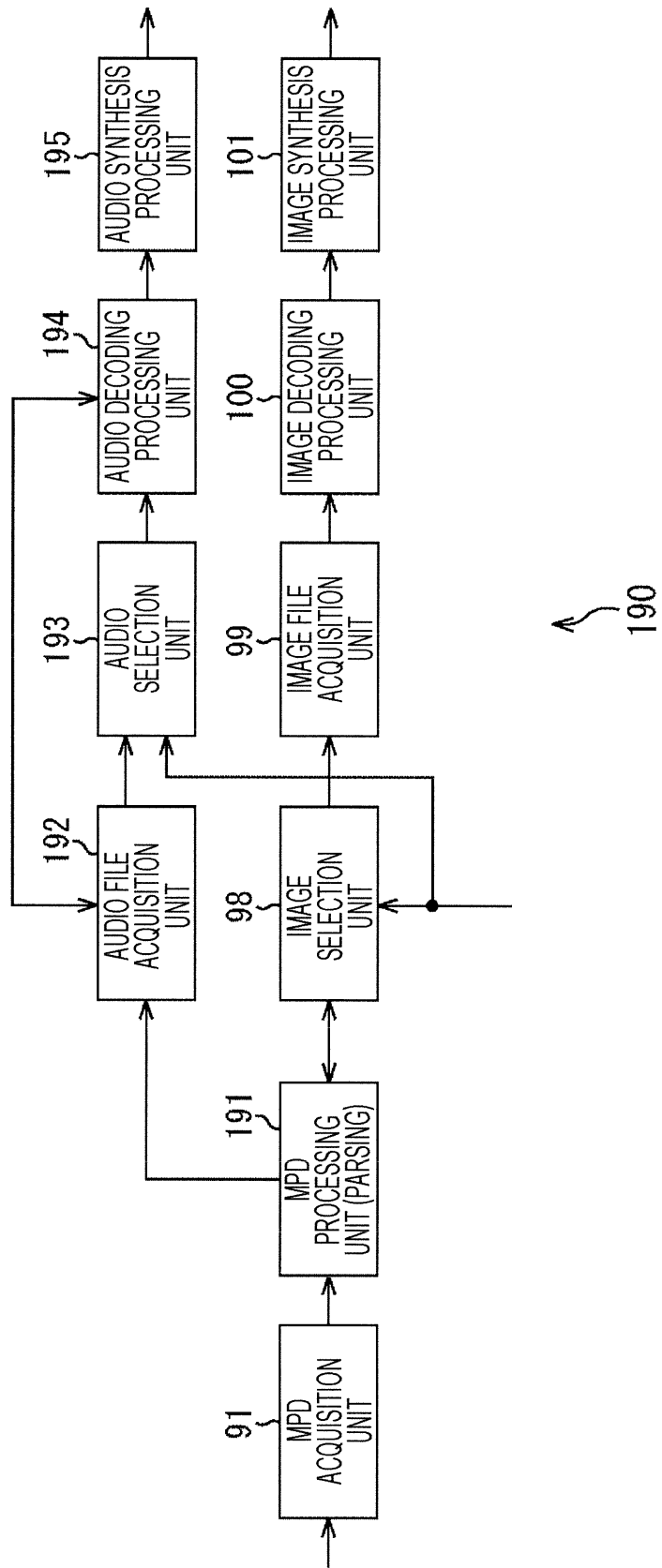


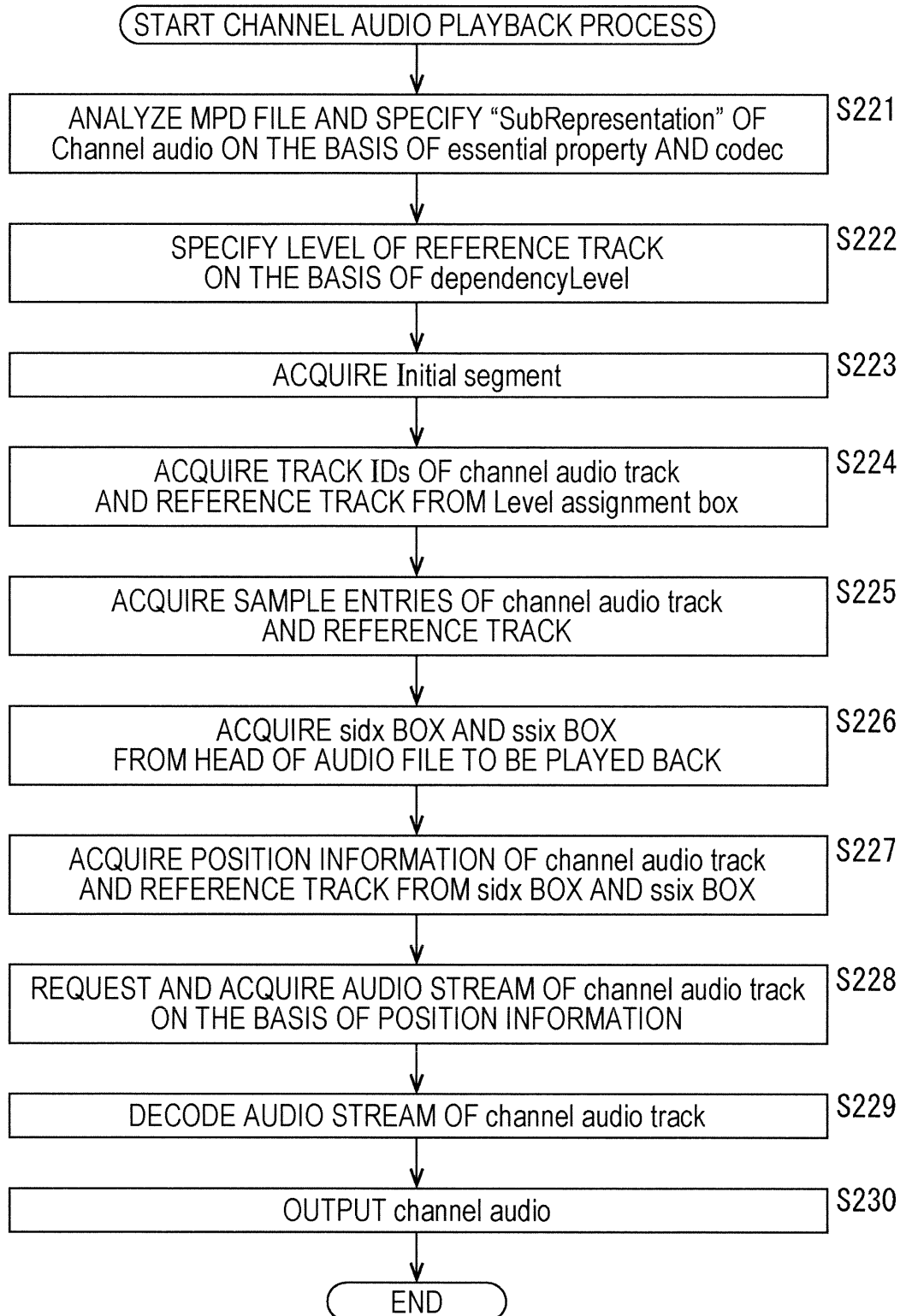
FIG. 48

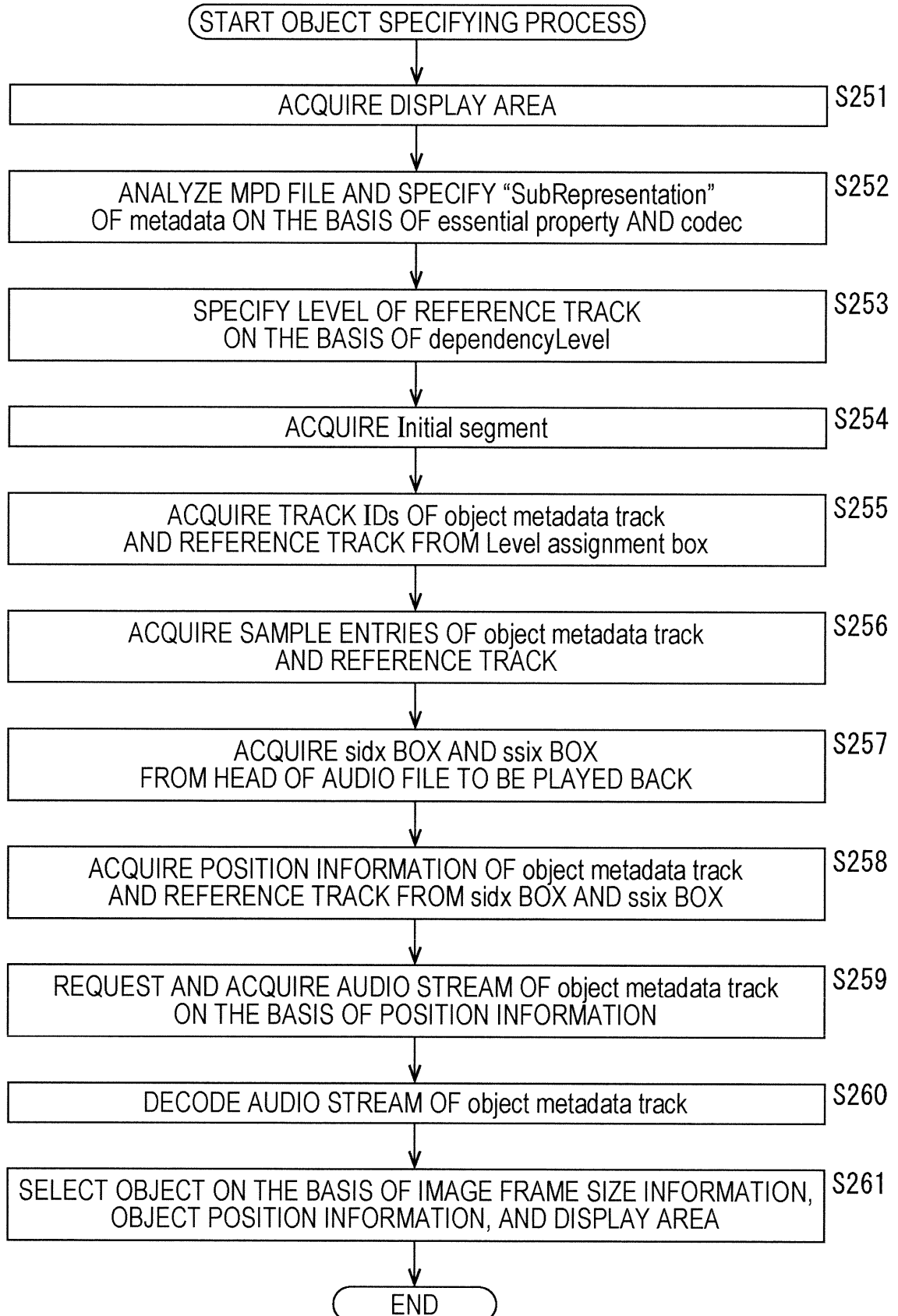
FIG. 49

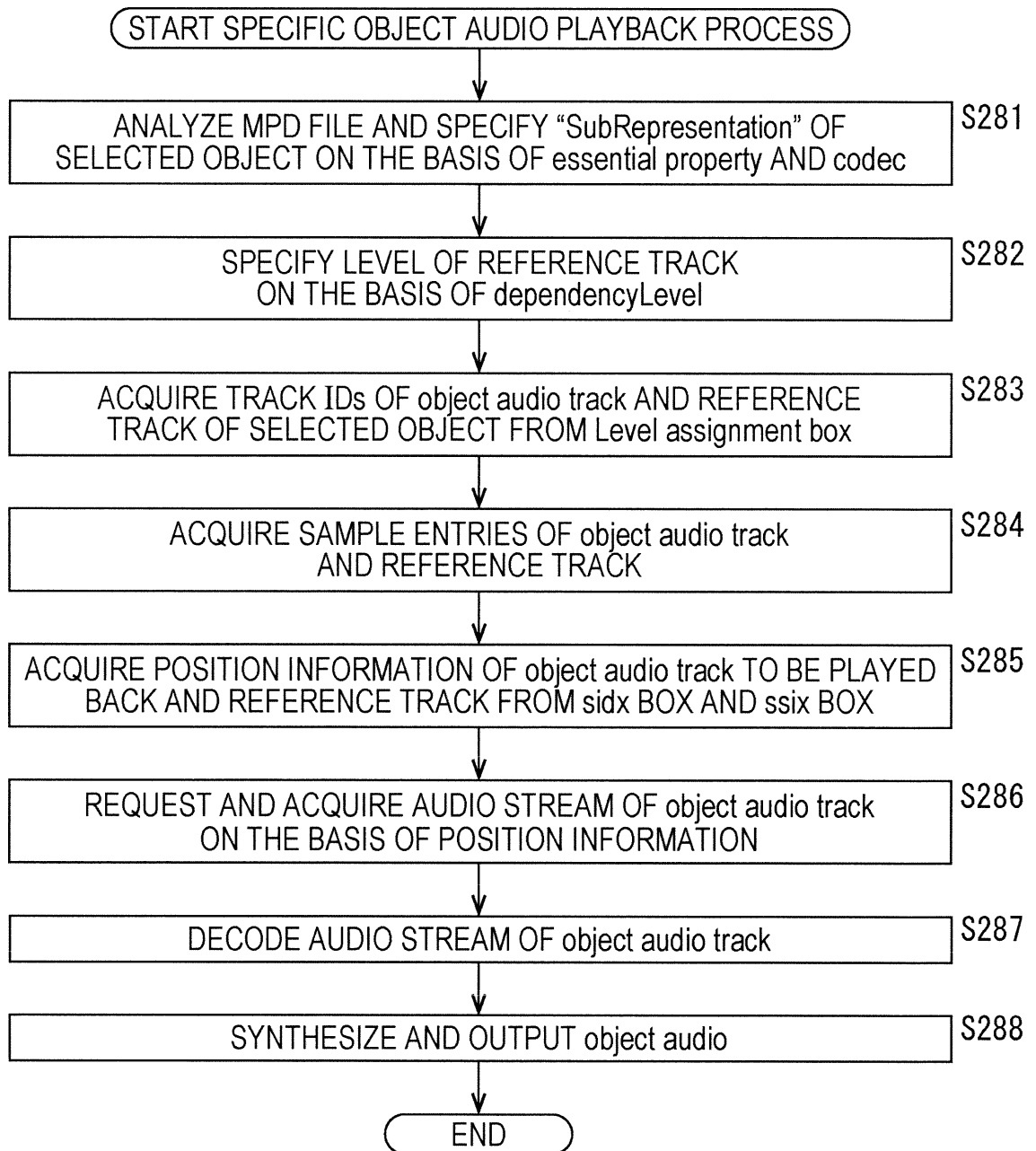
FIG. 50

FIG. 51

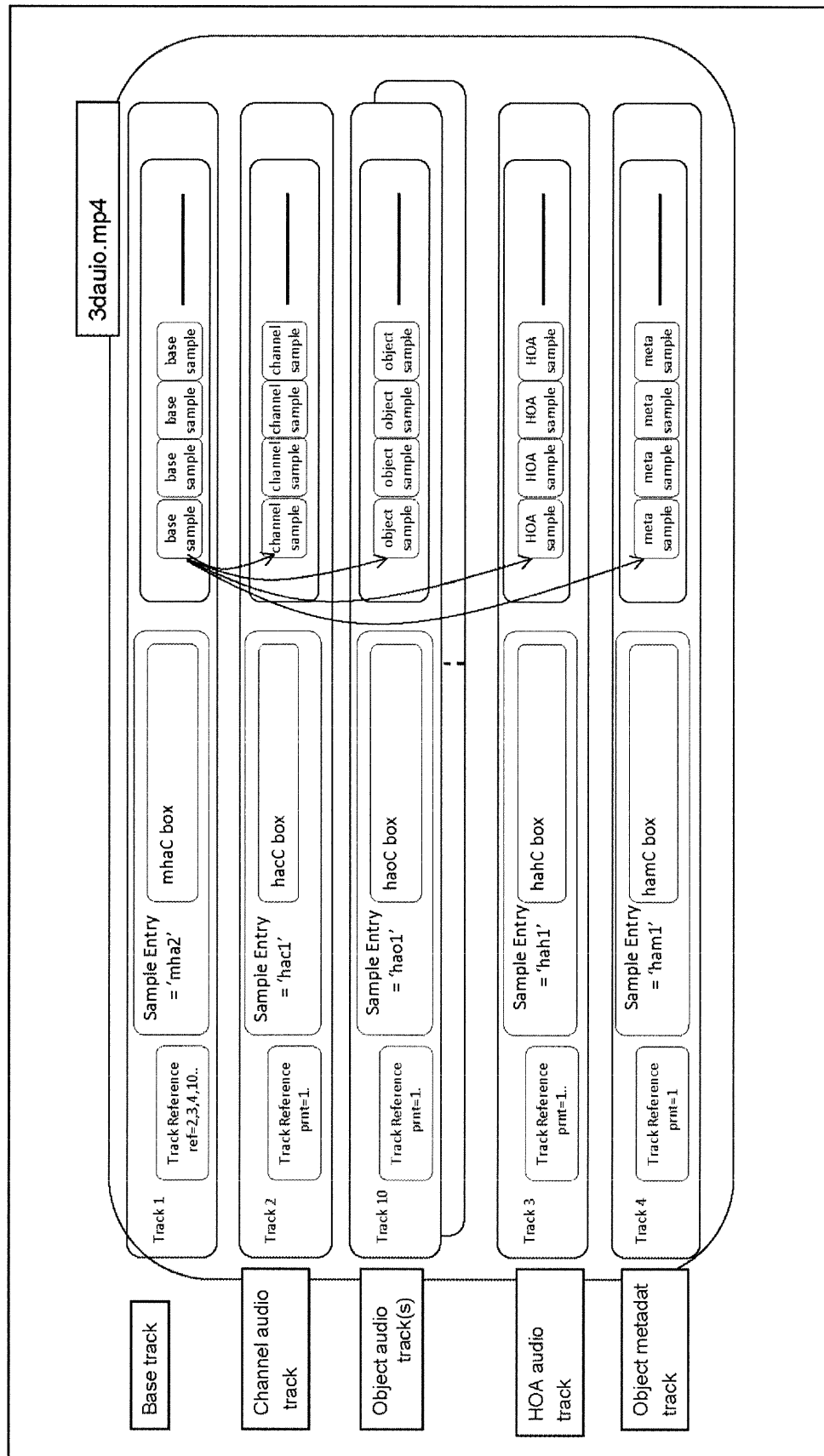


FIG. 52

```
class MHASampleEntry() extends AudioSampleEntry('mha2') {
    MHACConfigurationBox config;
    MPEG4BitRateBox();           // optional
    MPEG4ExtensionDescriptorsBox (); // optional
}

class MHACConfigurationBox() extends Box('mhaC') {
    MHADecoderConfigurationRecord() MHACConfig;
}

aligned(8) class MHADecoderConfigurationRecord {
    unsigned int(8) configurationVersion = 1;
    unsigned int(8) MPEG4AudioProfile;
    unsigned int(8) MPEG4AudioLevel;
    unsigned int(8) referenceChannelLayout;
    unsigned int(16) mpeg3daConfigLength;
    bit(8*mpeg3daConfigLength) mpeg3daConfig;
}
```

FIG. 53

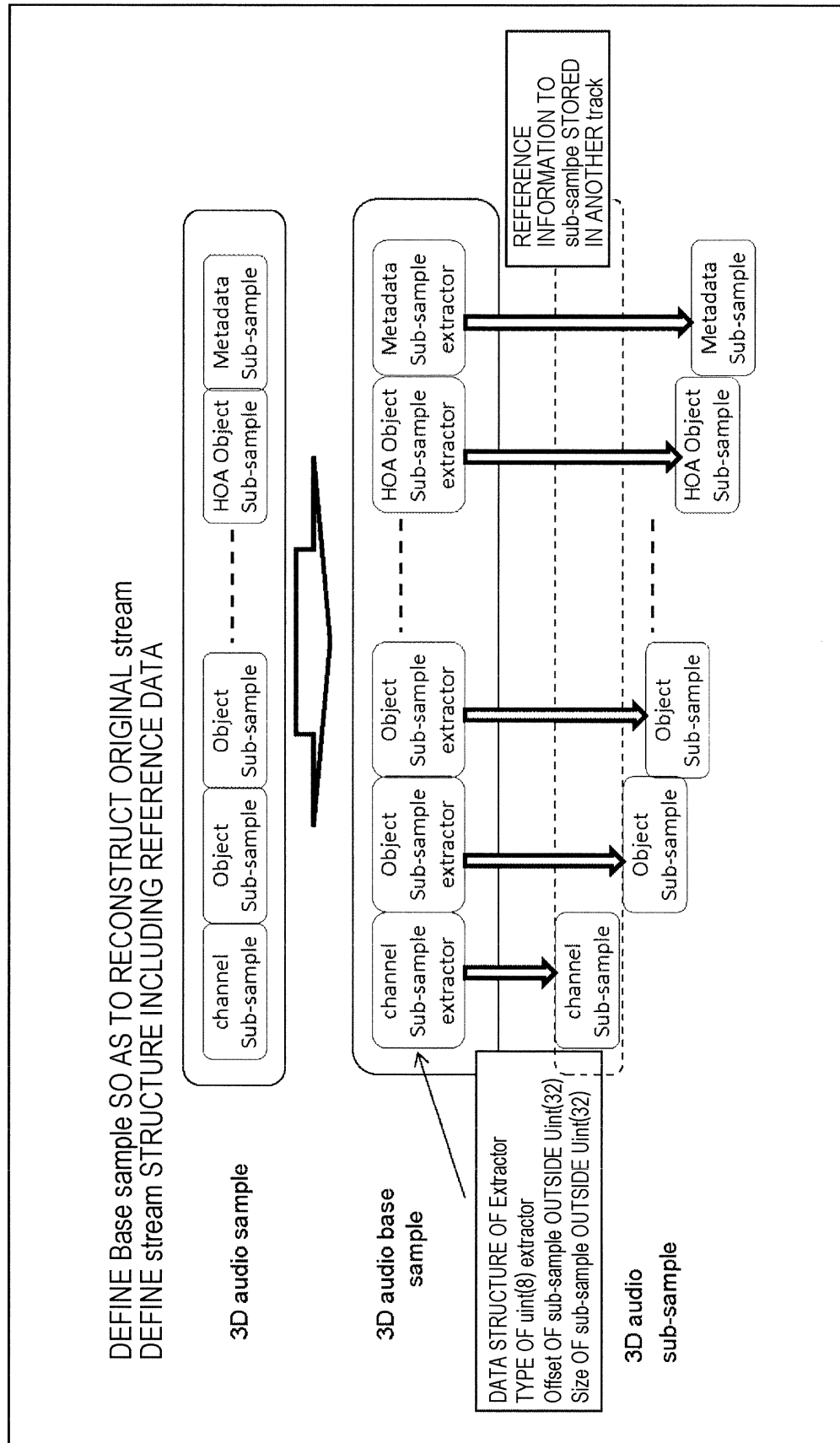


FIG. 54

Syntax	No. of bits	Mnemonic
<pre> mpegh3daFrame() { usacIndependencyFlag; for (elemIdx=0; elemIdx<numElements; ++elemIdx) { if ((usacElementType[elemIdx] != ID_USAC_EXT) && (elementLengthPresent == 1)) { elementLength } switch (usacElementType[elemIdx]) { case ID_USAC_SCE: mpegh3daSingleChannelElement(usacIndependencyFlag); break; case ID_USAC_CPE: mpegh3daChannelPairElement(usacIndependencyFlag); break; case ID_USAC_LFE: mpegh3daLifeElement(usacIndependencyFlag); break; case ID_USAC_EXT: mpegh3daExtElement(usacIndependencyFlag); break; } } } </pre>	1 16	uimsbf uimsbf elementLength, NOTE 1 elementLength, NOTE 1 elementLength, NOTE 1

case ID_USAC_EXT:
mpegh3daExtElement(usacIndependencyFlag);
break;

NOTE 1: If present, elementLength represents the length of the corresponding element it refers to in number of bits.

```
DEFINE extractor TO
REFER TO EXTERNAL
sub-sample AS
mpeg3daExtElement
```

FIG. 55

Bitstream Field	Value
usacExtElementType	ID EXT ELE EXTRACTOR
usacExtElementConfigLength	0
usacExtElementDefaultLengthPresent	0
usacExtElementPayloadFrag	0

STORE TYPE, OFFSET, AND SIZE INFORMATION
OF REFERENCE sub-sample AS ACTUAL DATA

FIG. 56

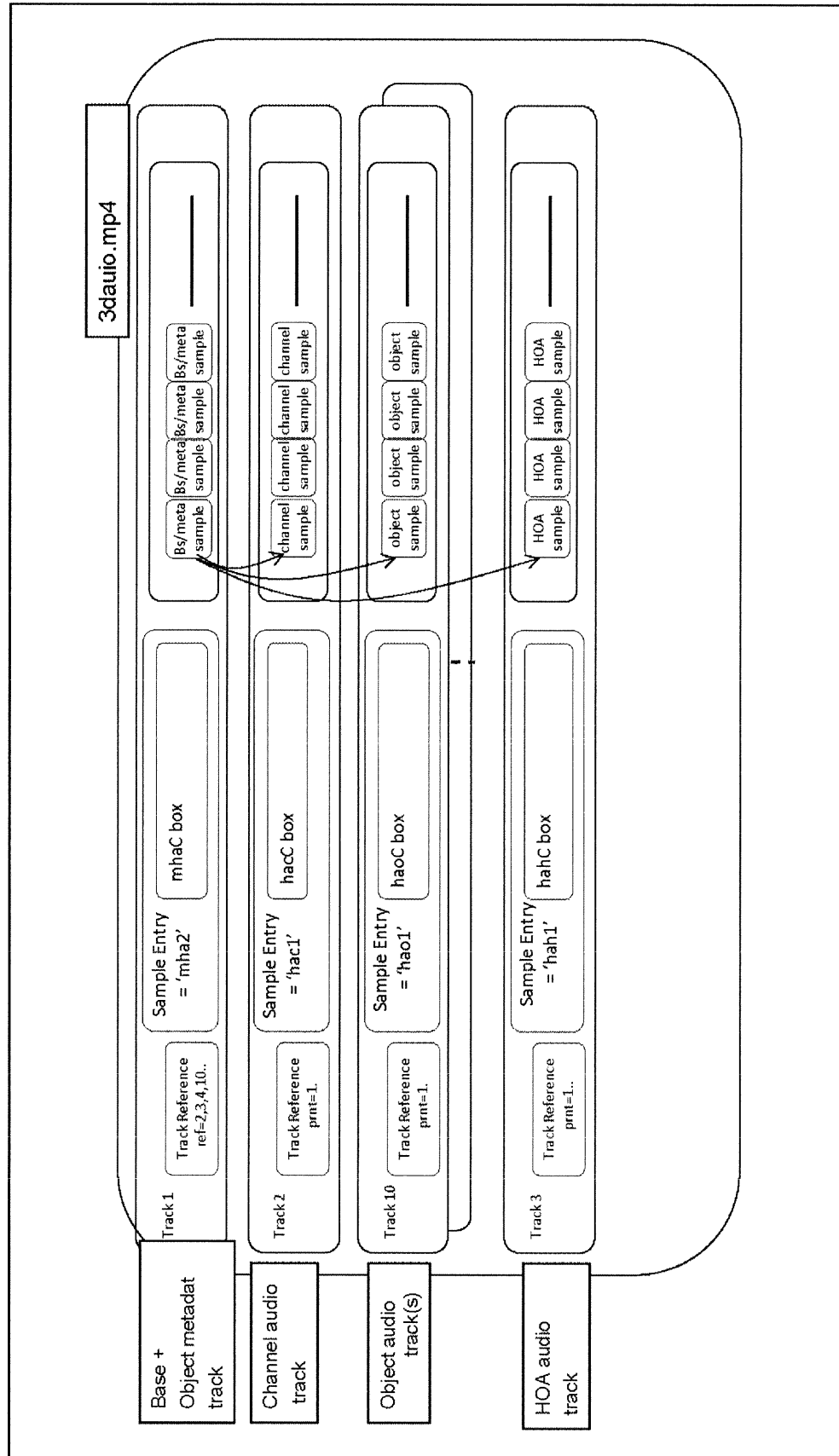


FIG. 57

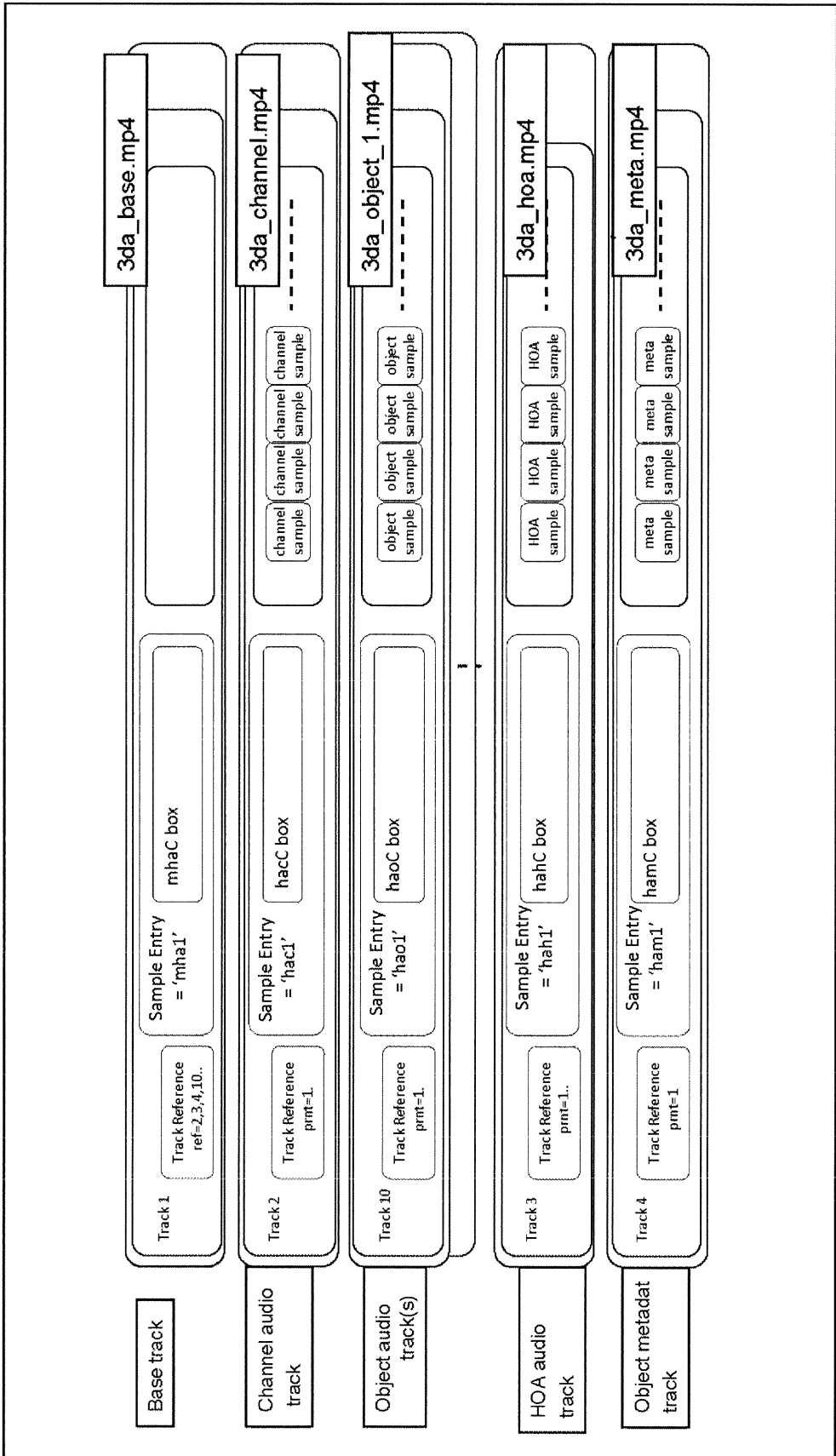


FIG. 58

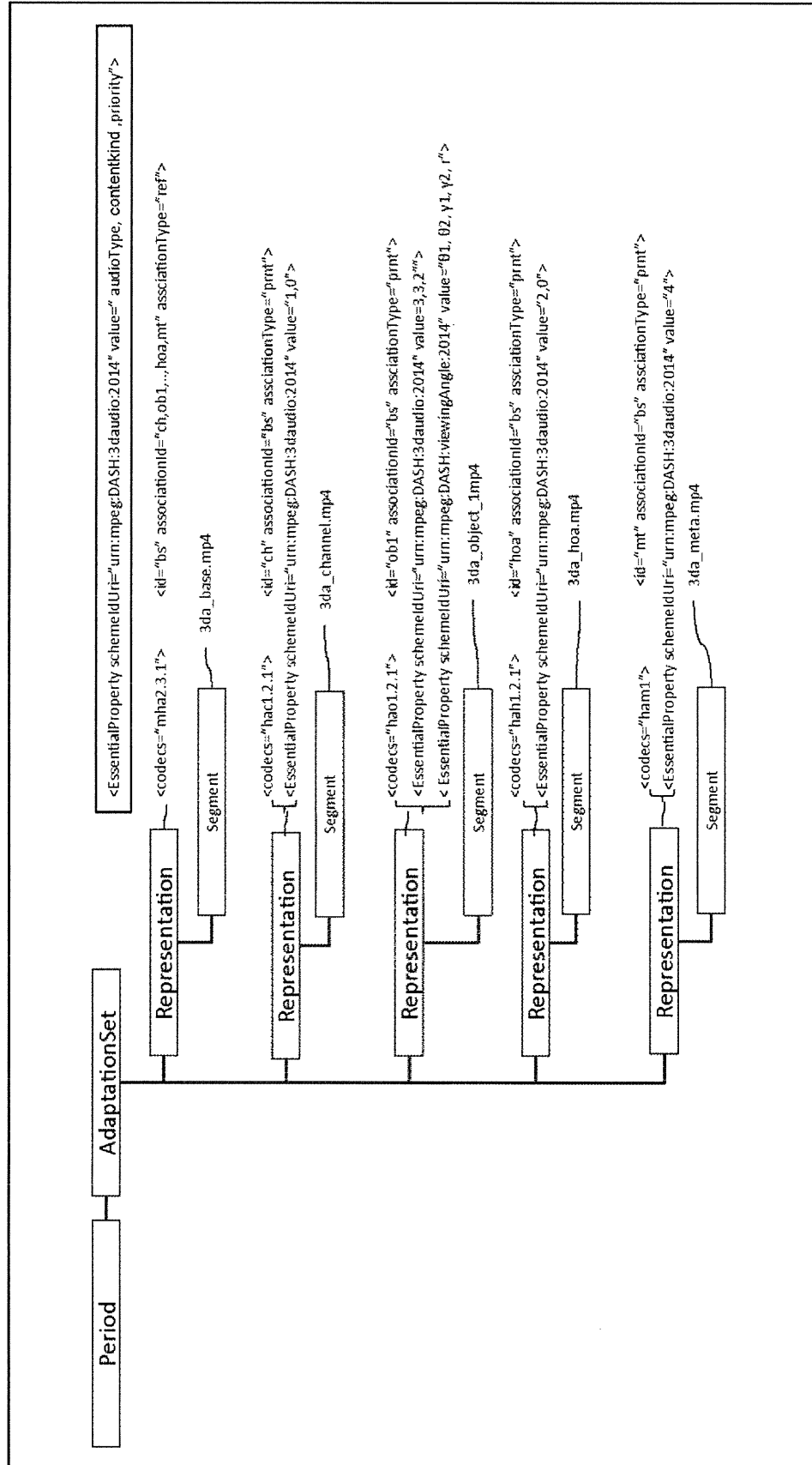


FIG. 59

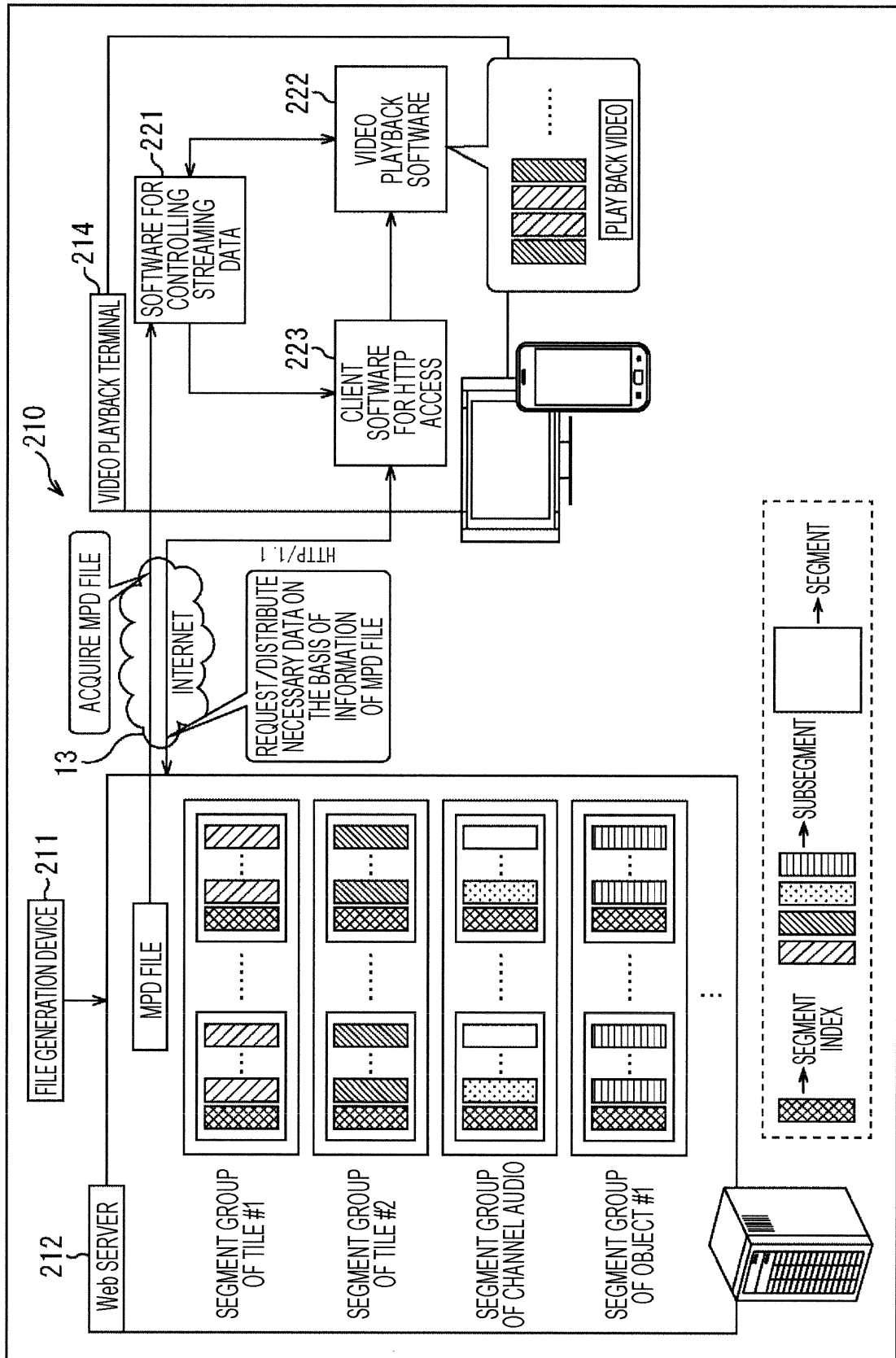


FIG. 60

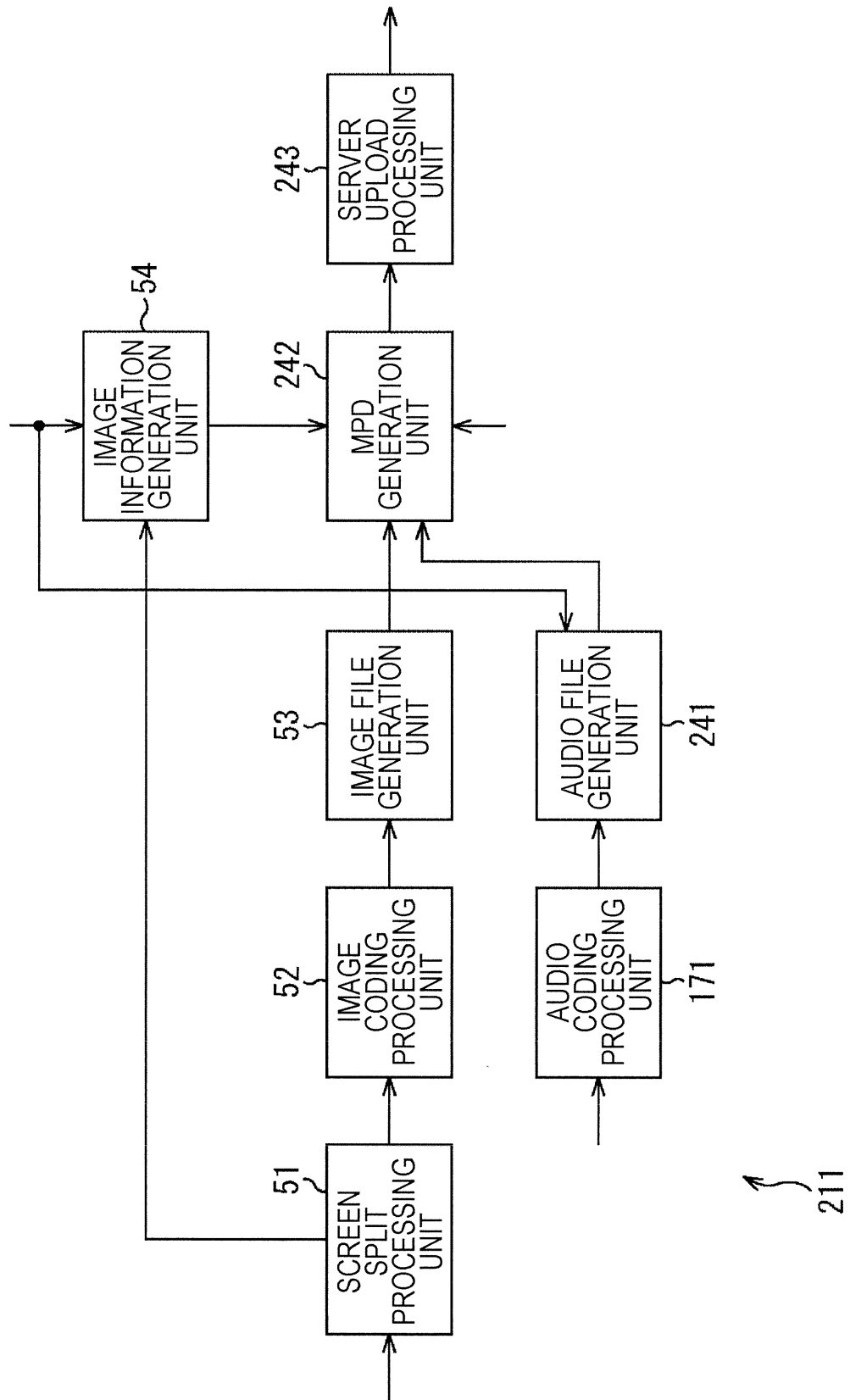


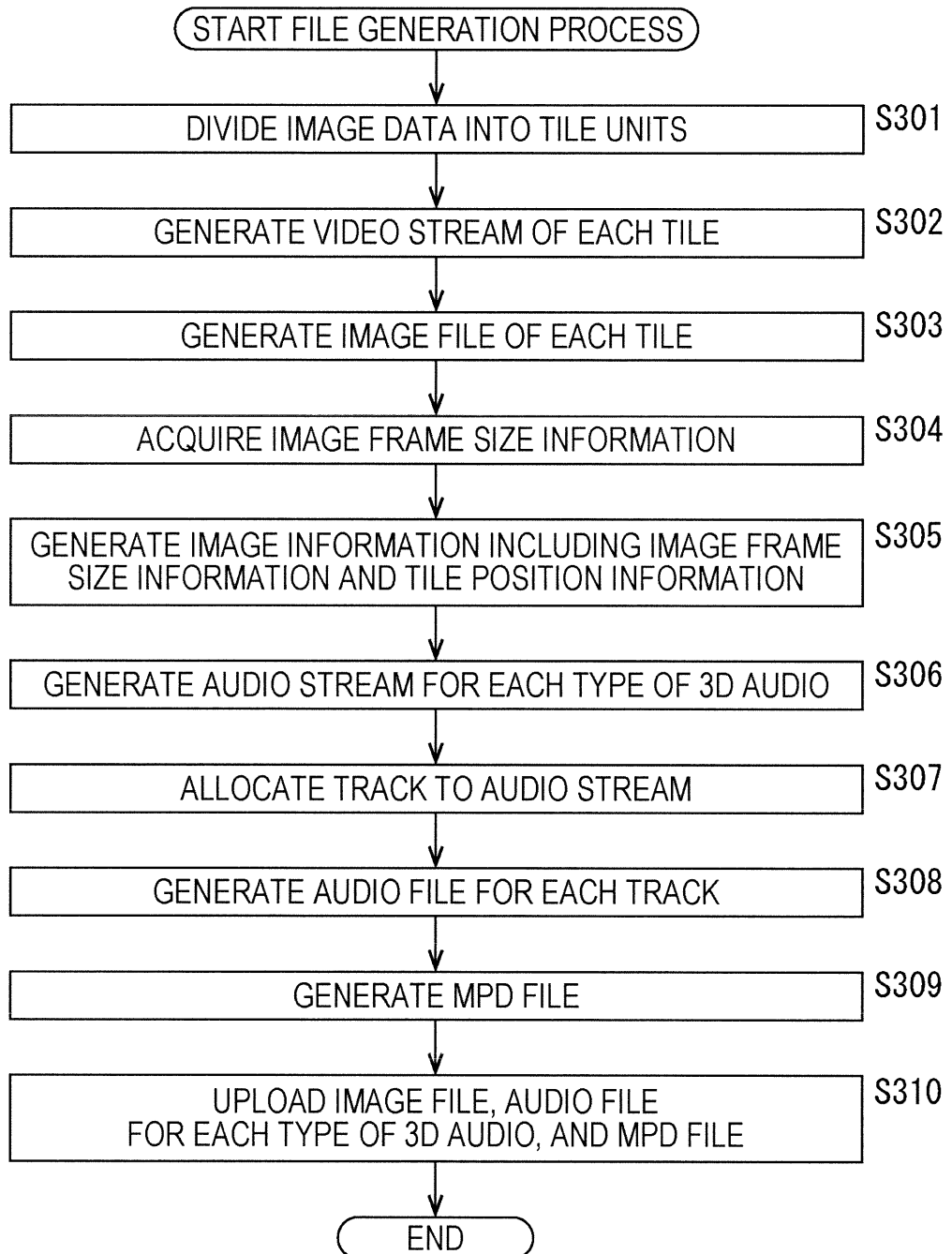
FIG. 61

FIG. 62

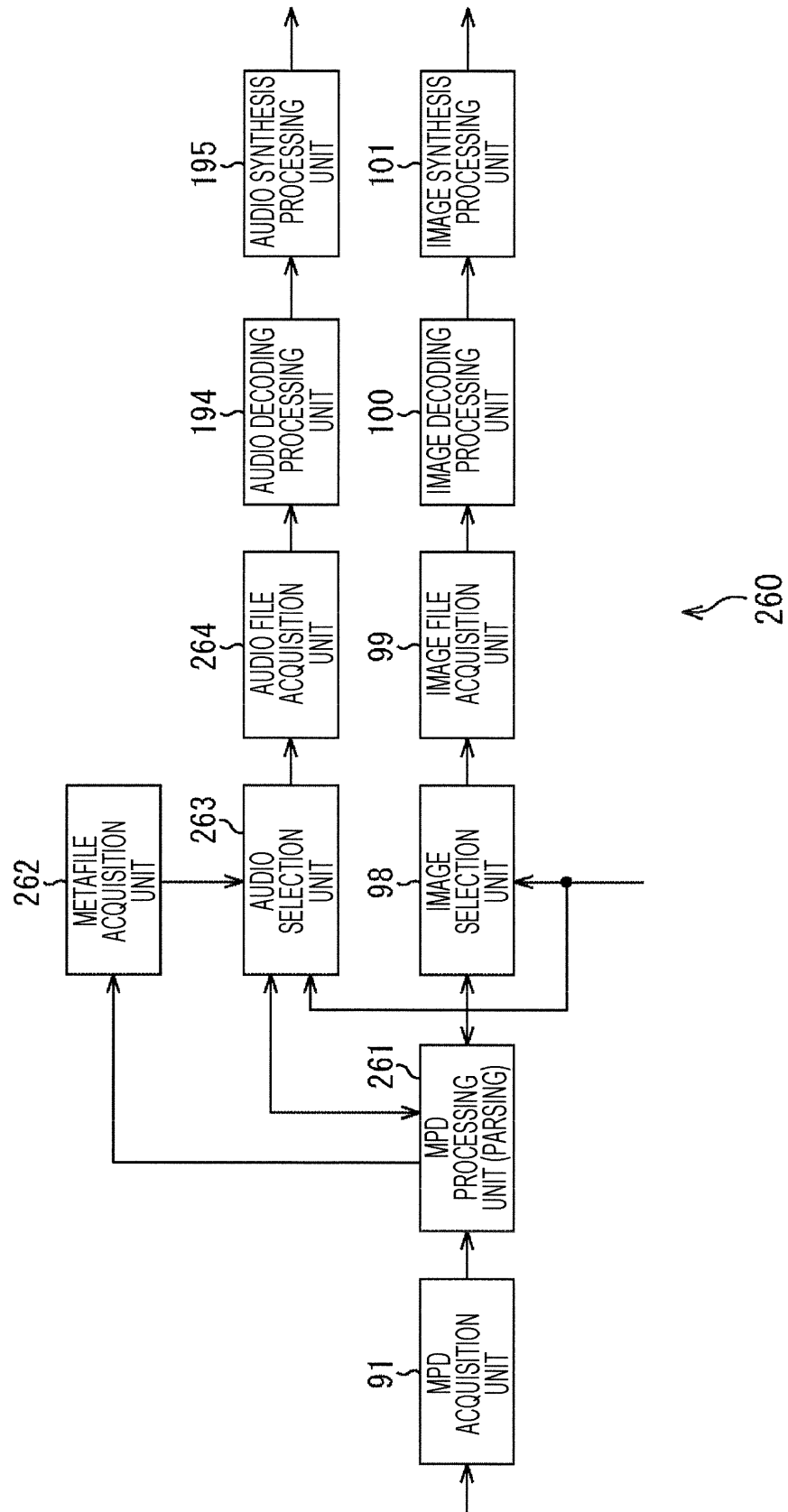


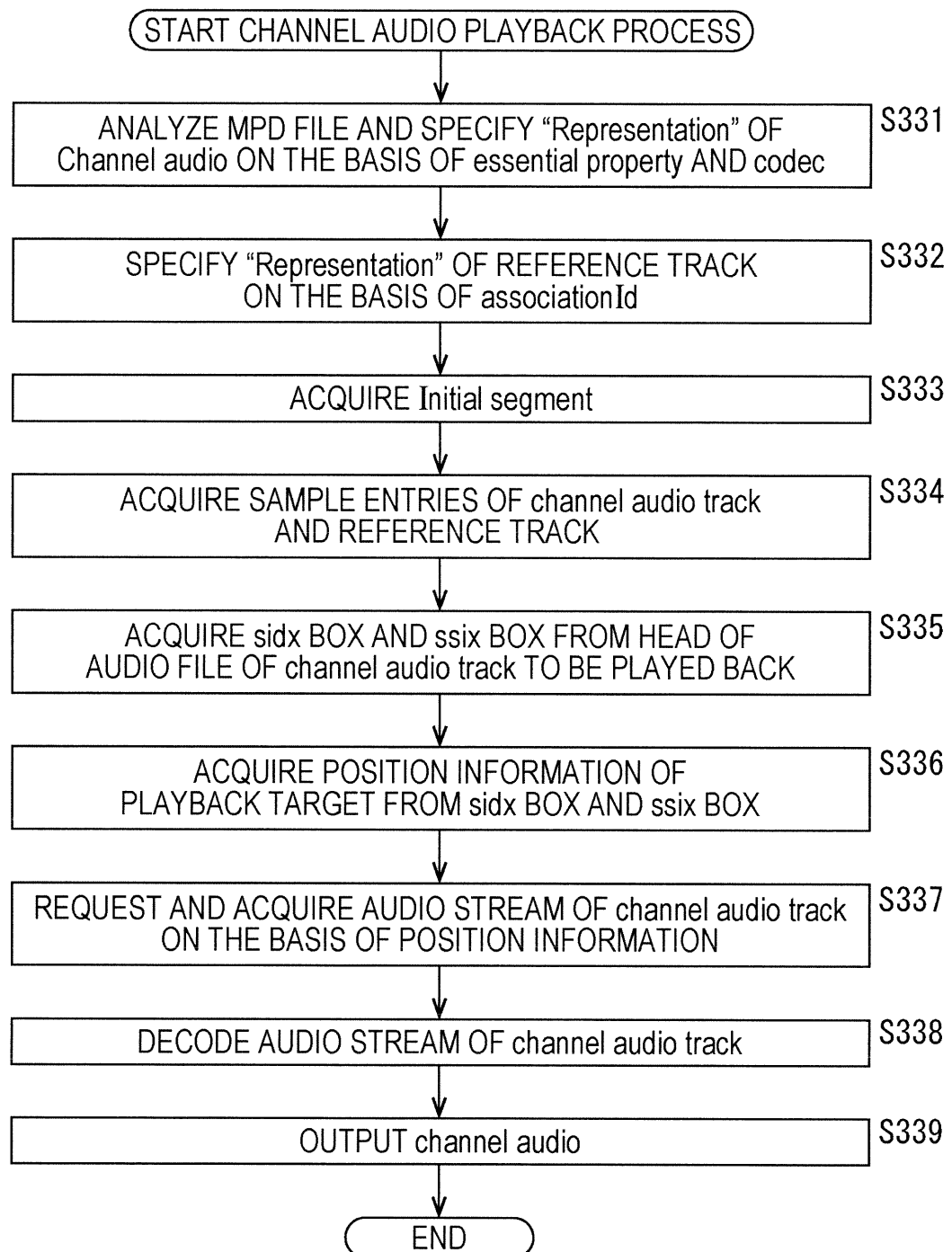
FIG. 63

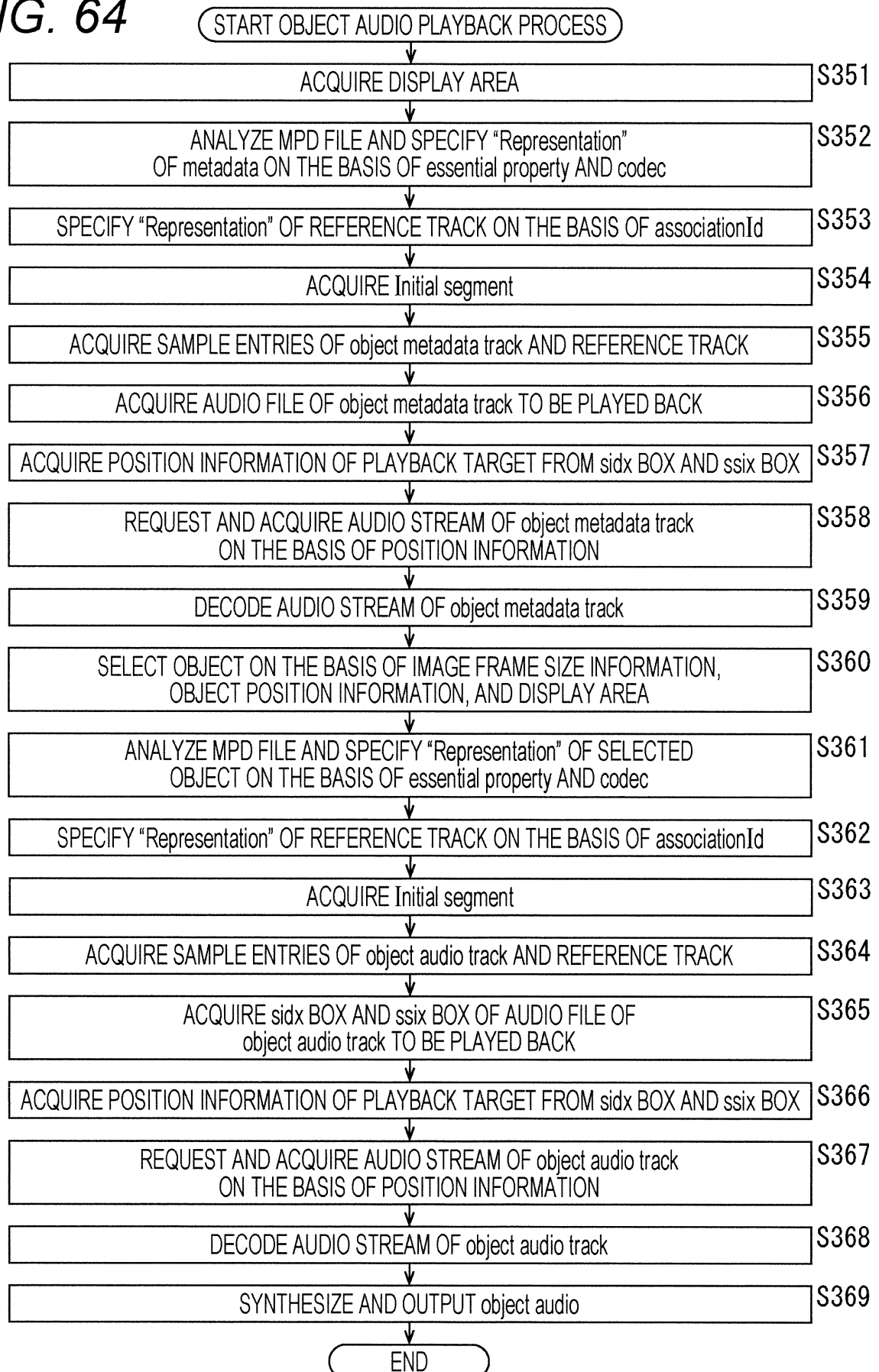
FIG. 64

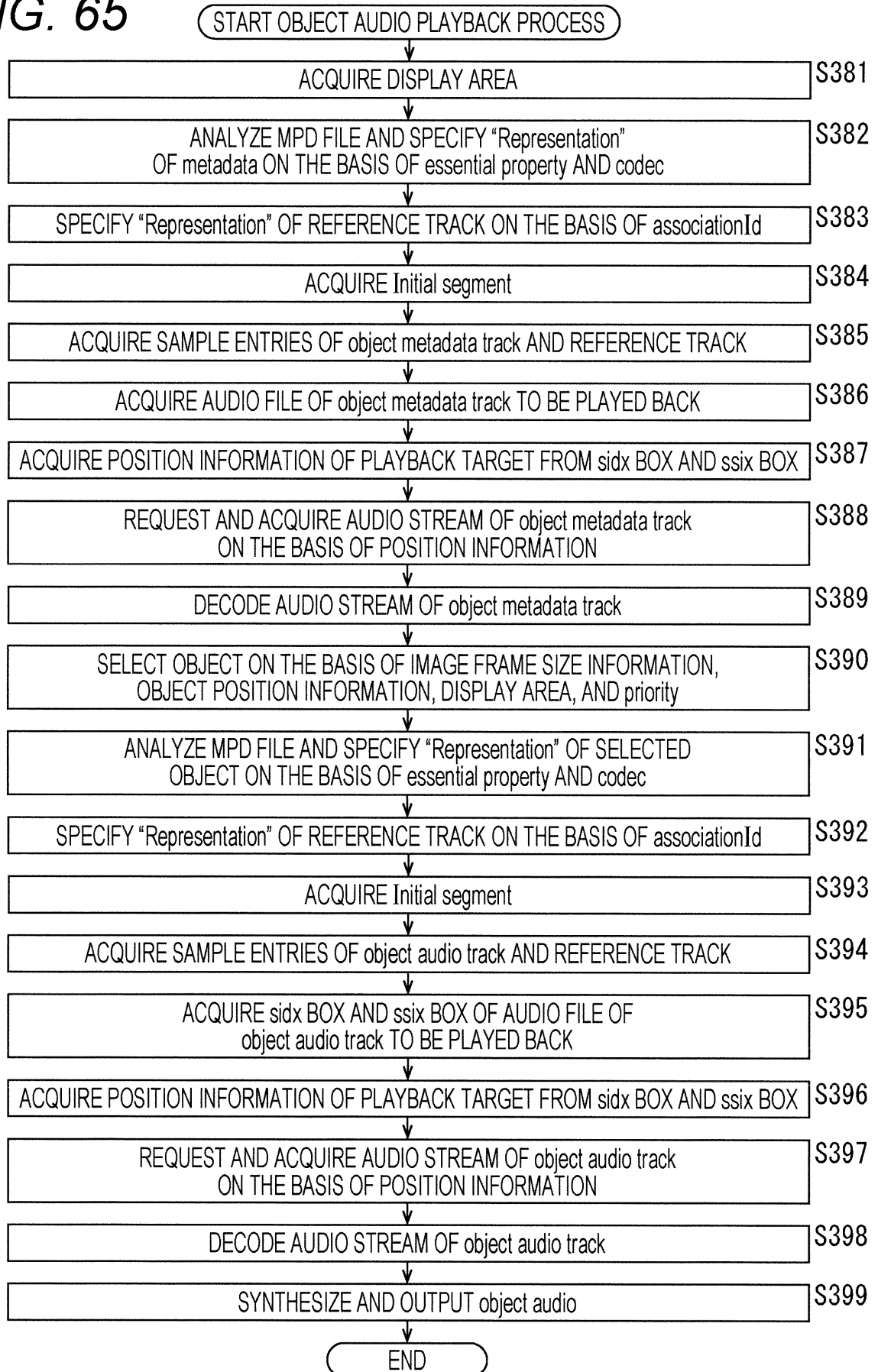
FIG. 65

FIG. 66

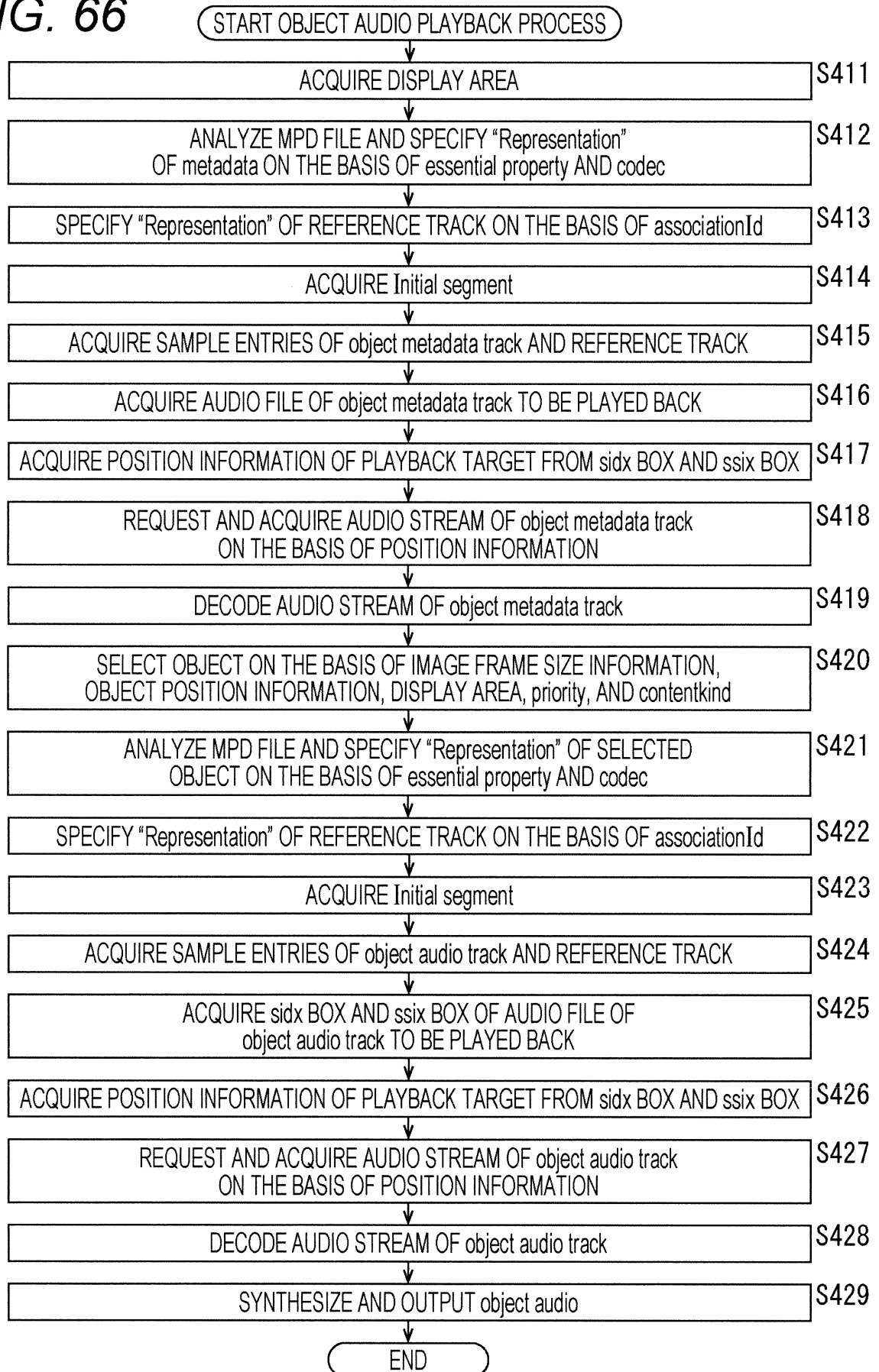


FIG. 67

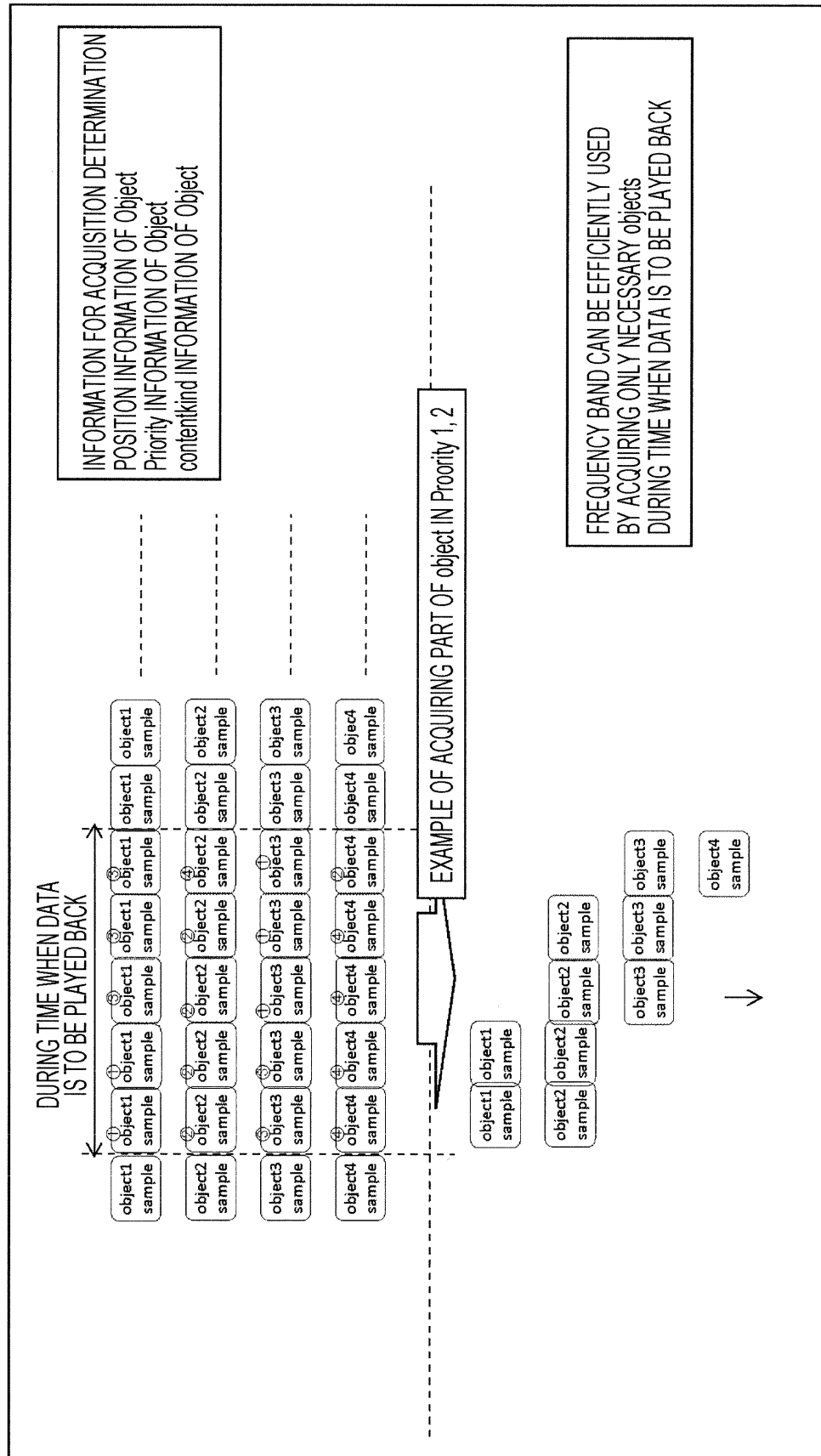


FIG. 68

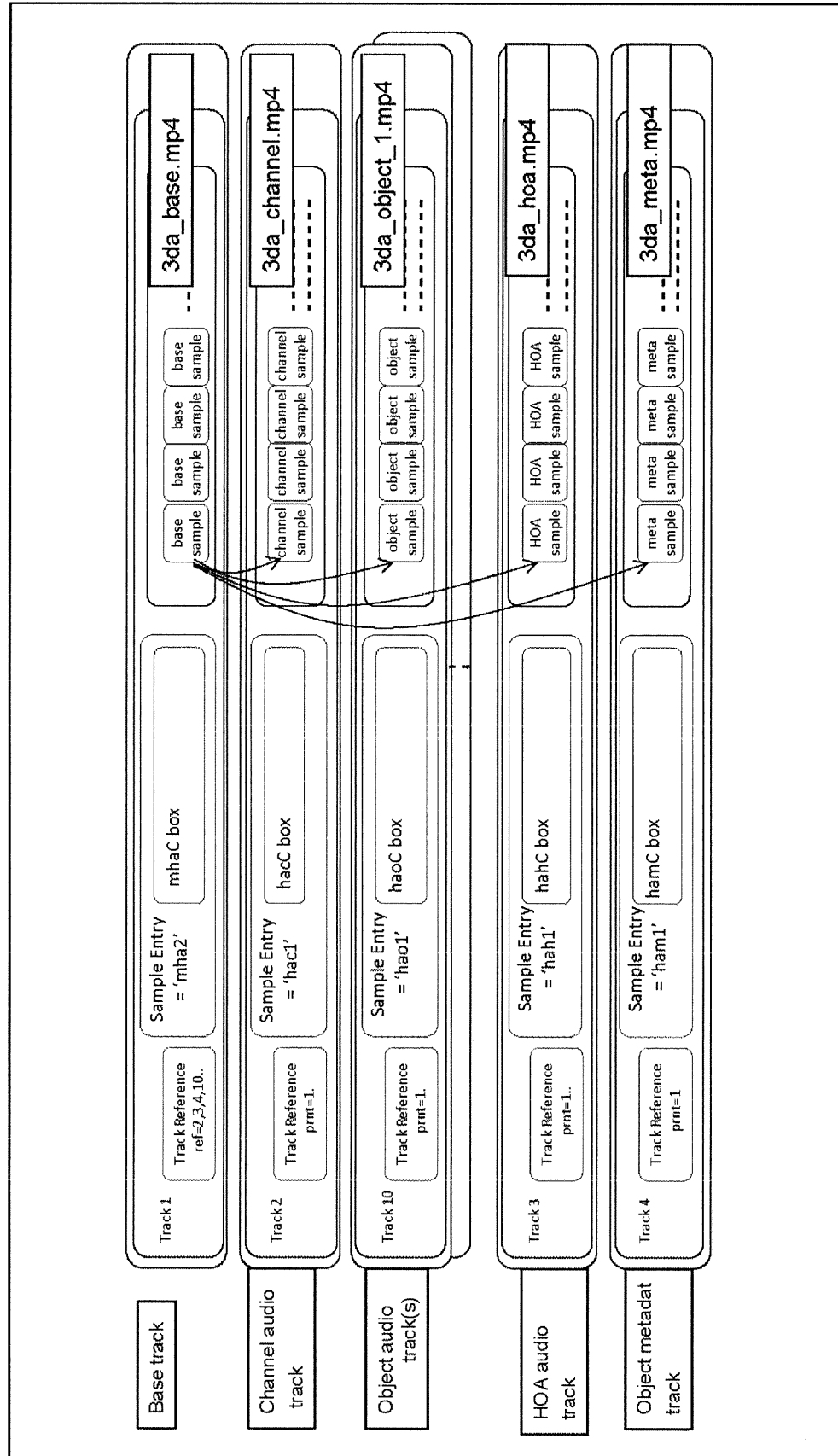


FIG. 69

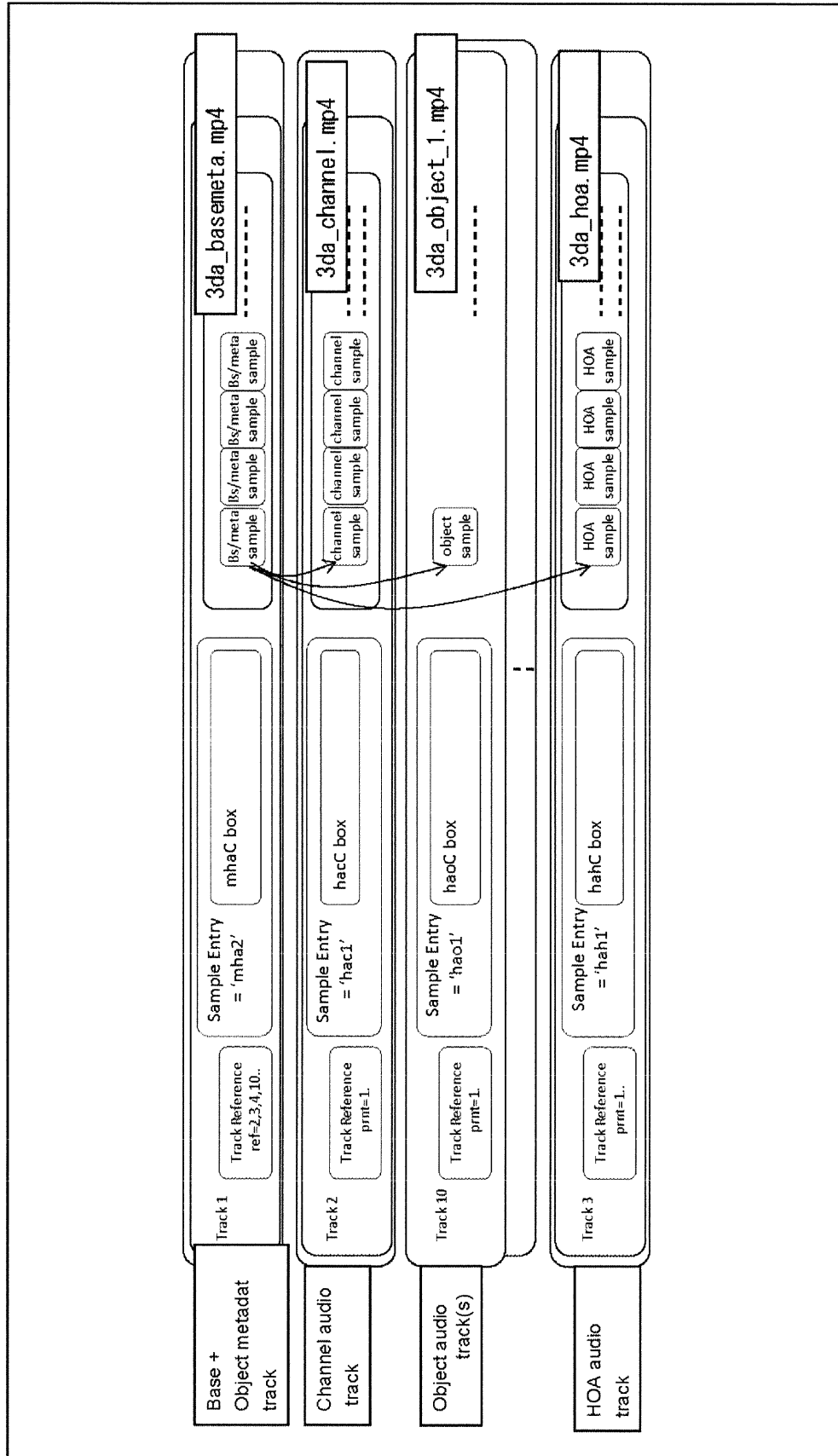


FIG. 70

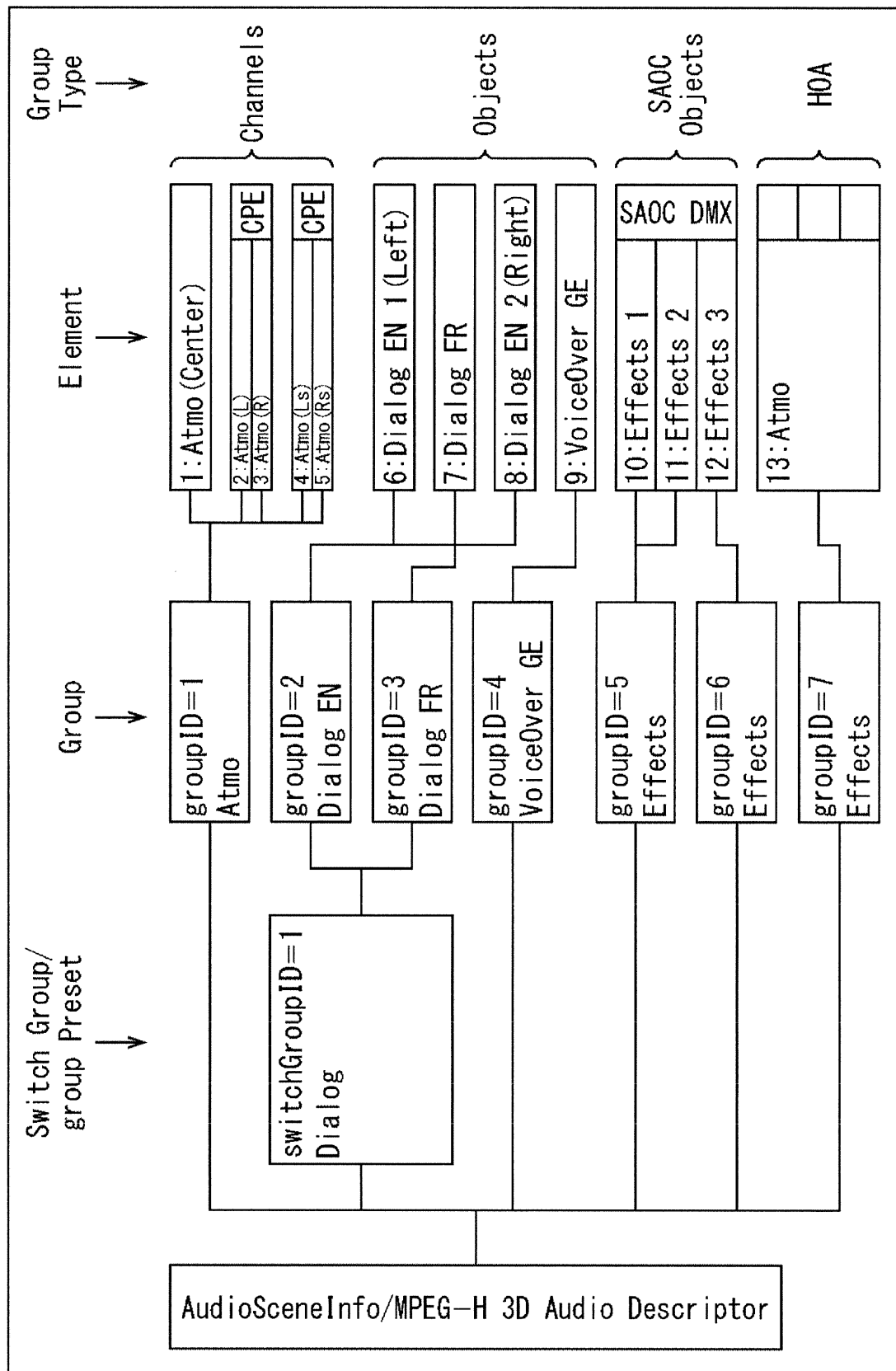


FIG. 71

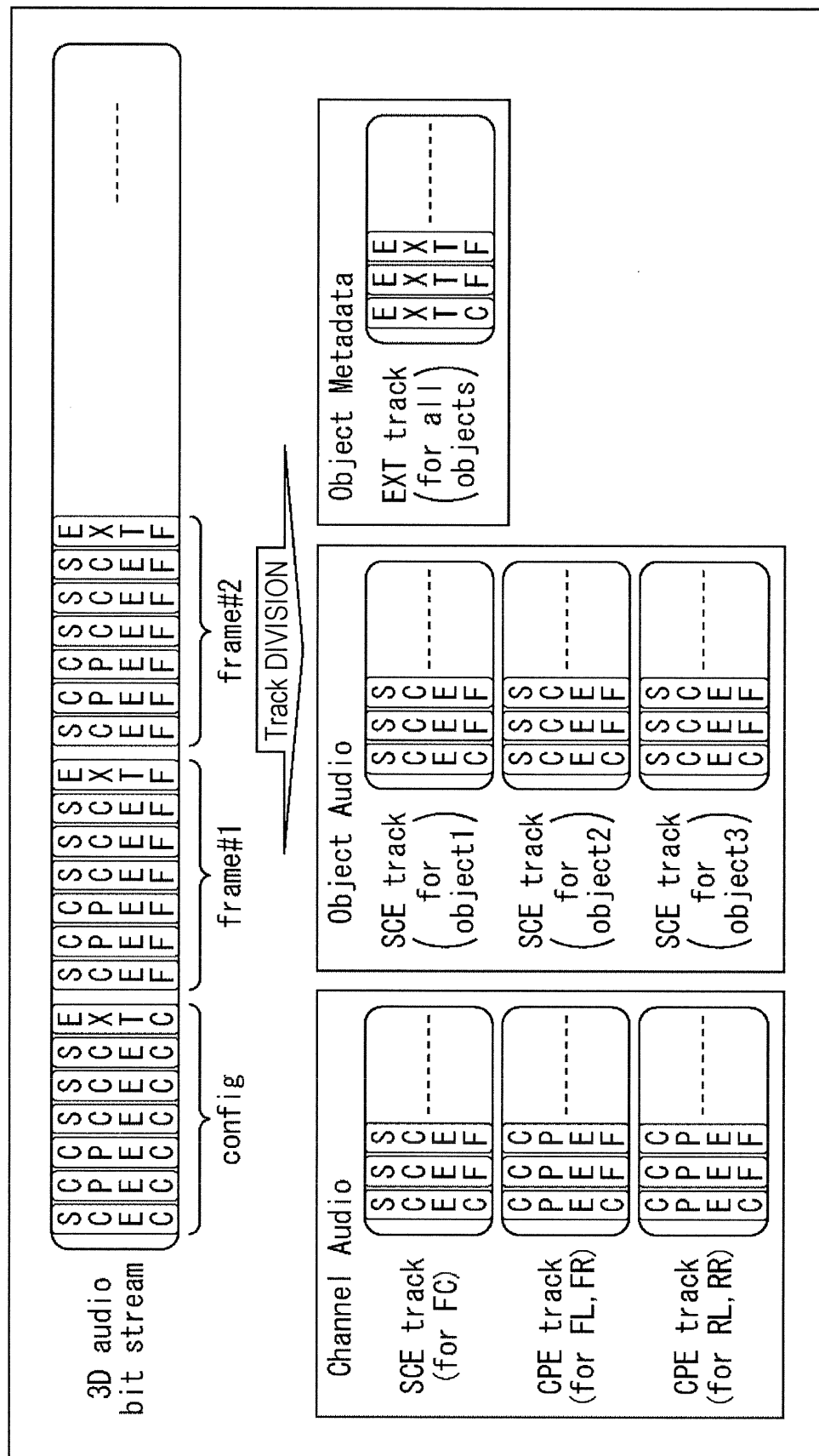


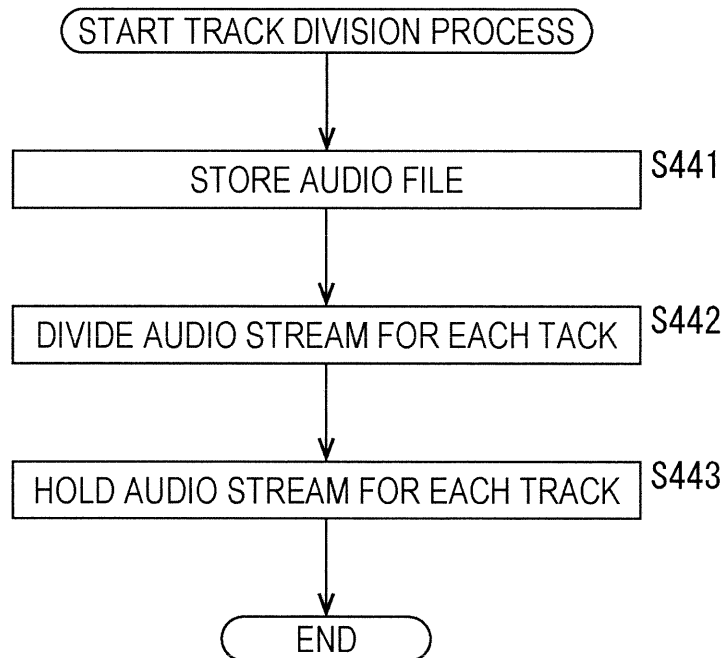
FIG. 72

FIG. 73

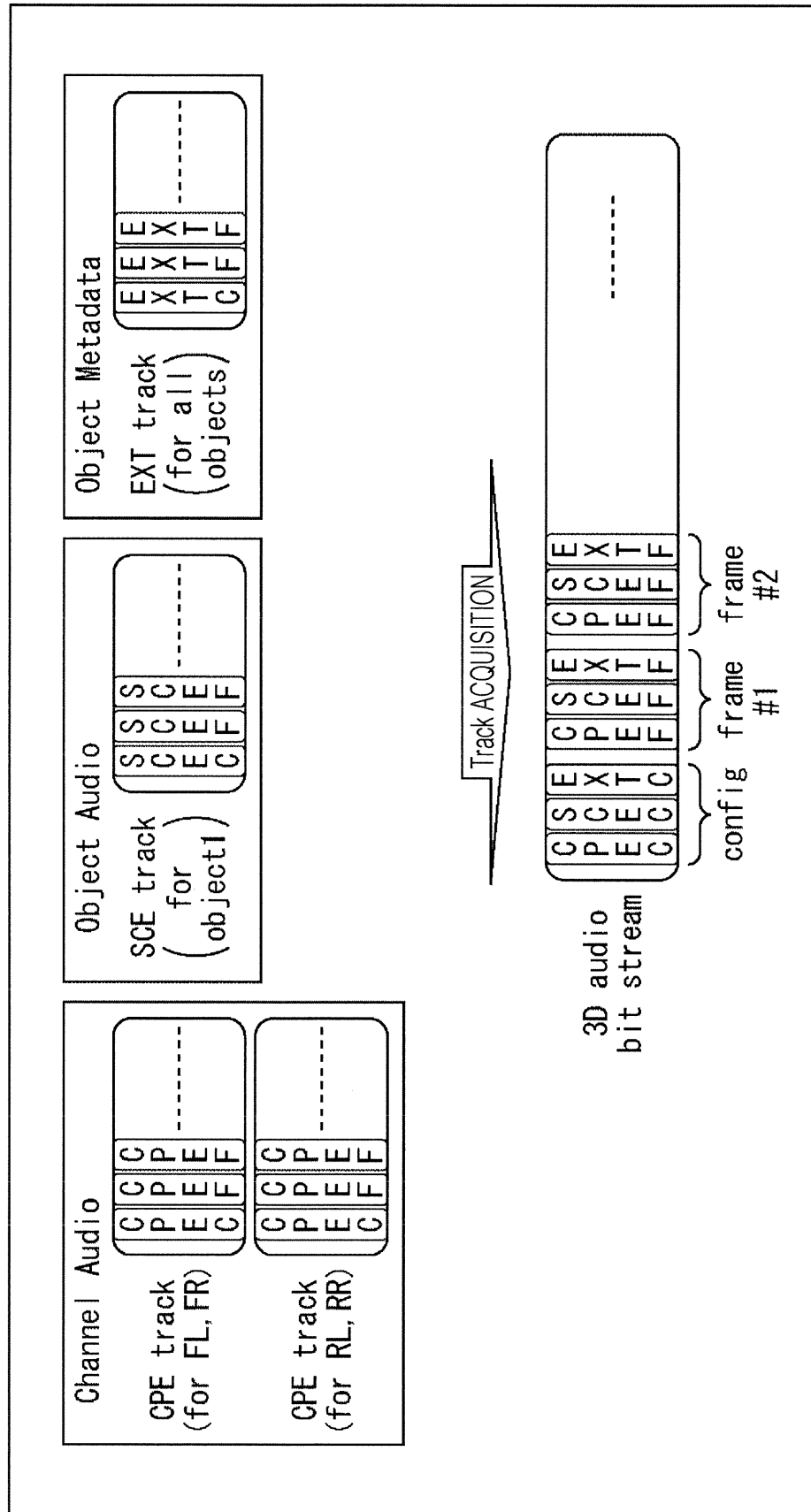


FIG. 74

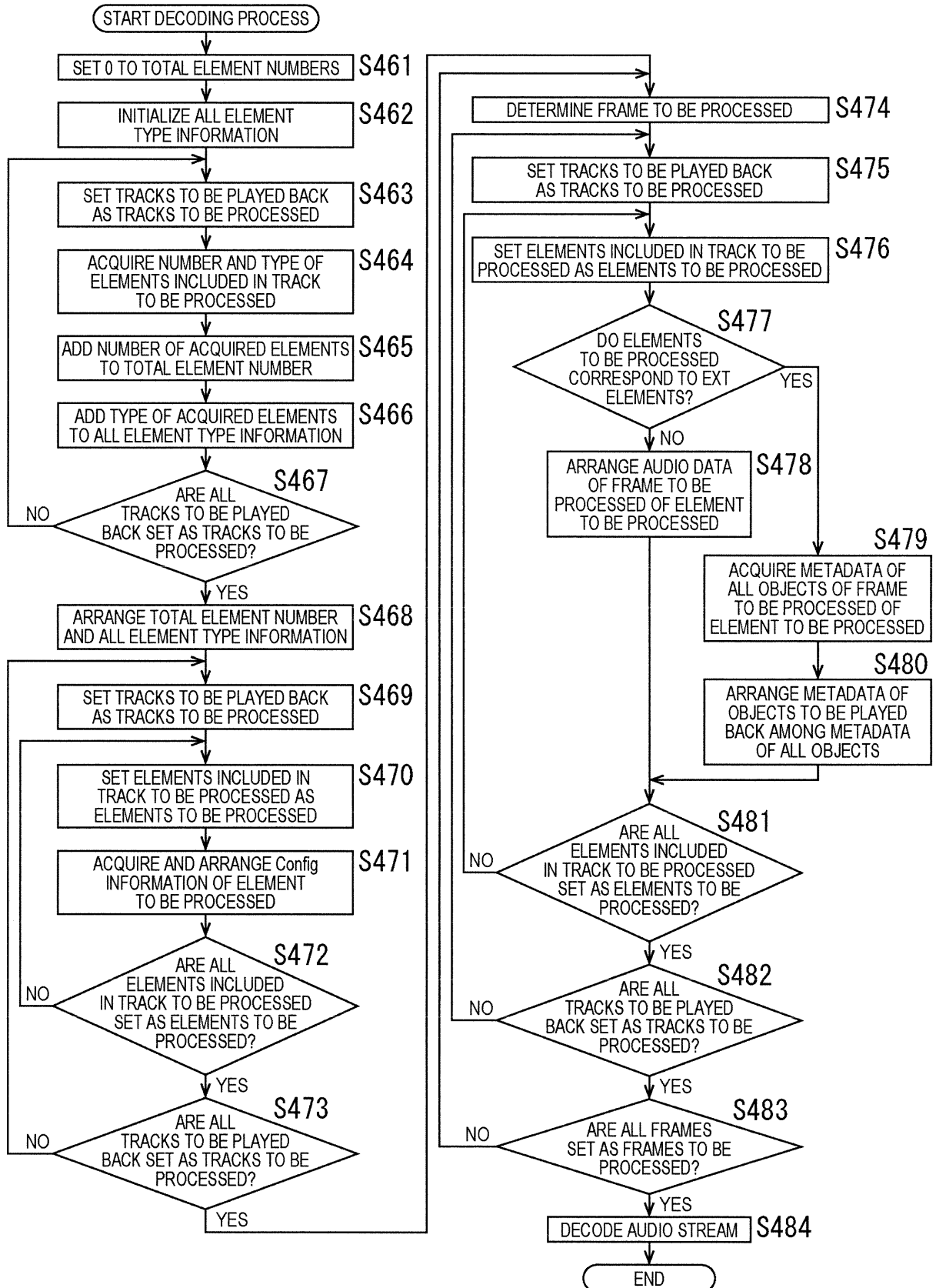


FIG. 75

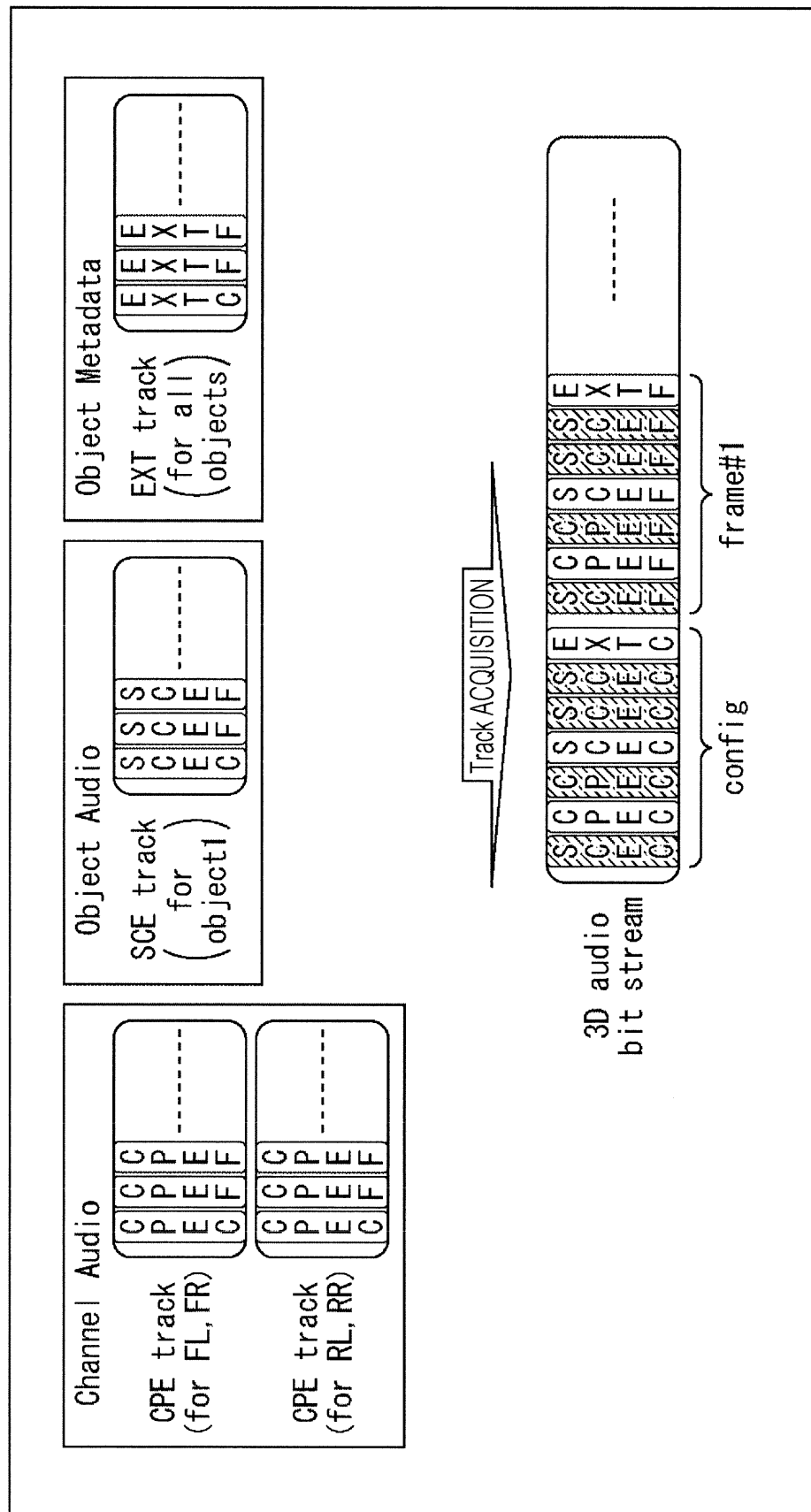


FIG. 76

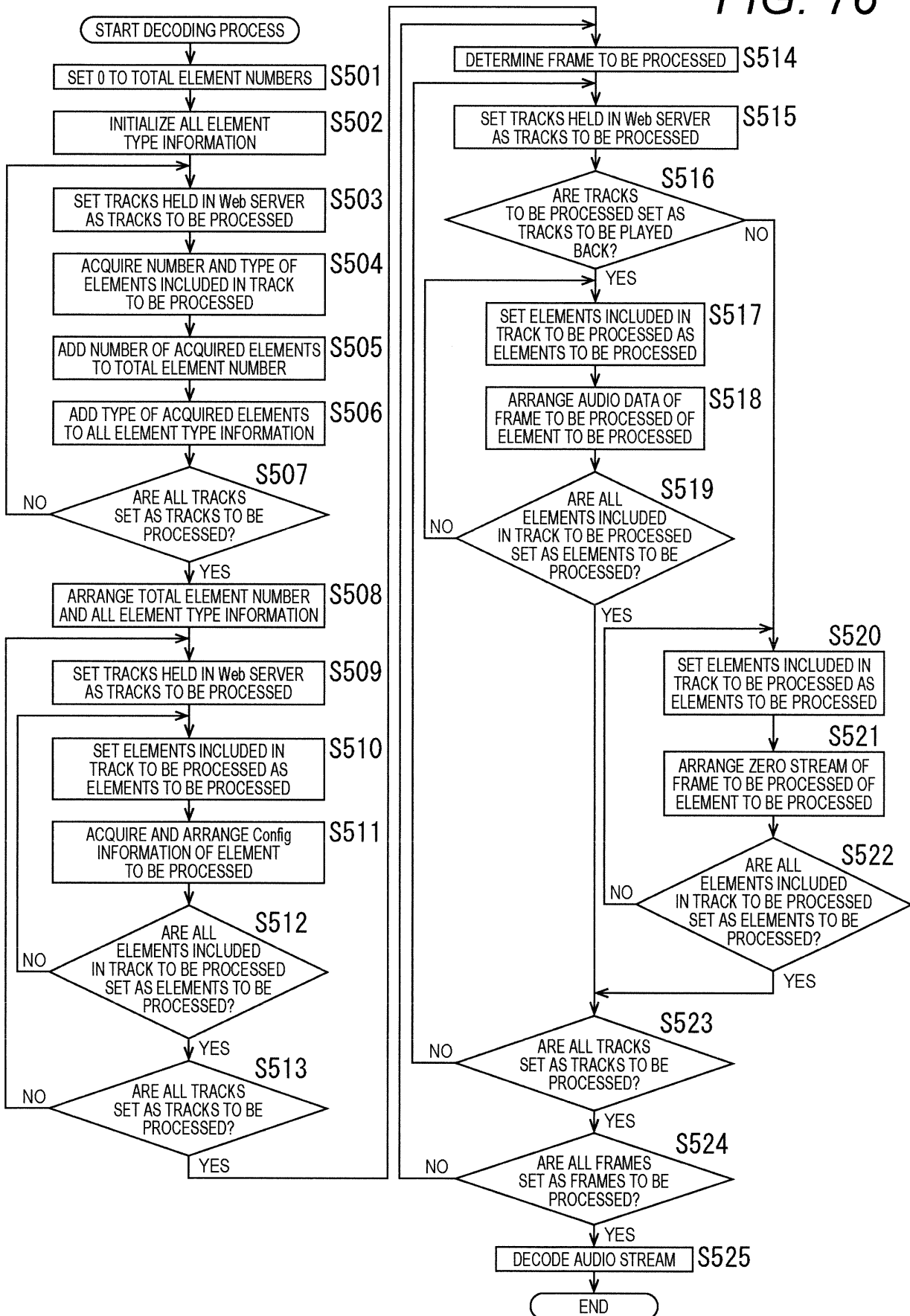


FIG. 77

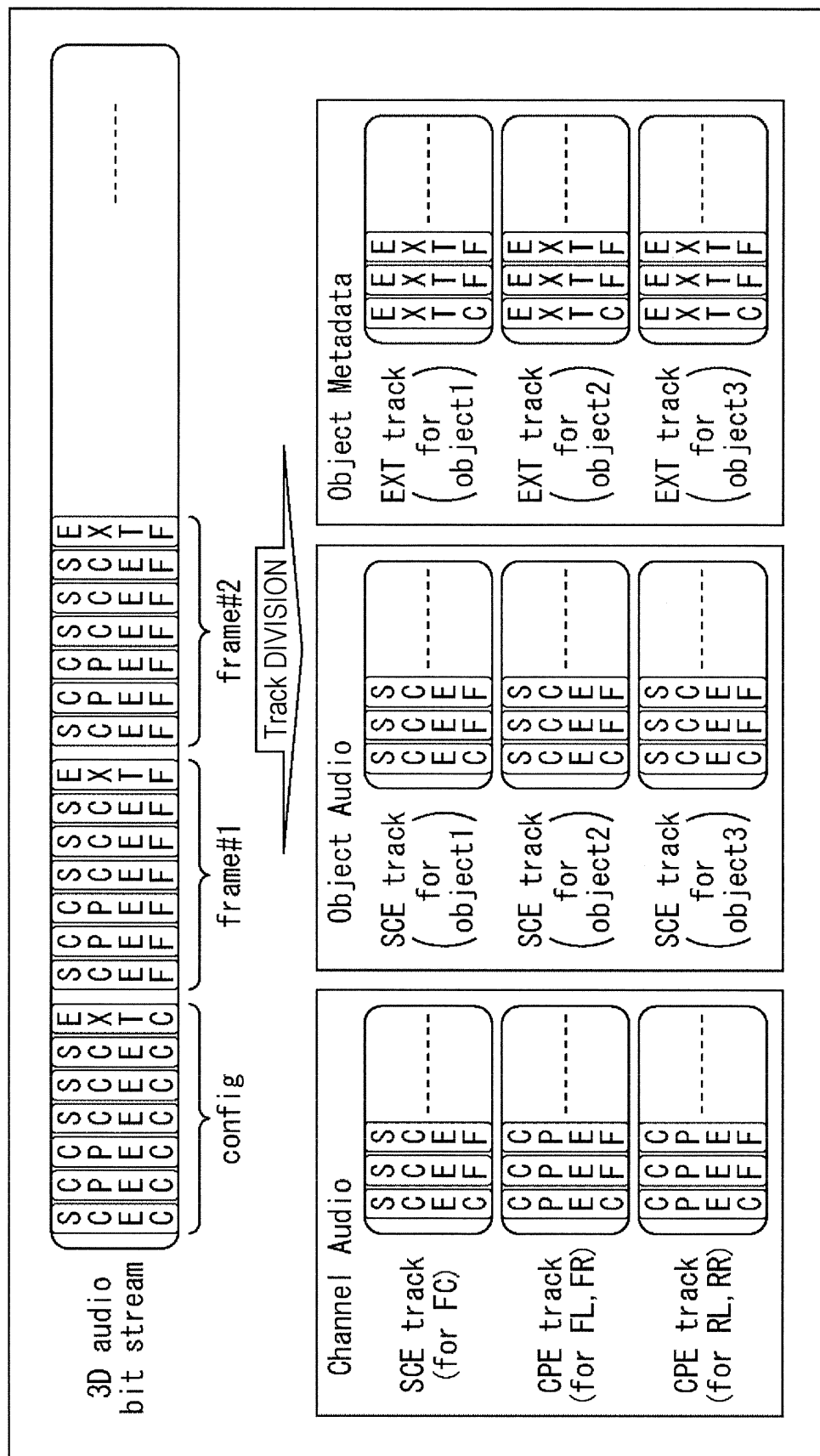
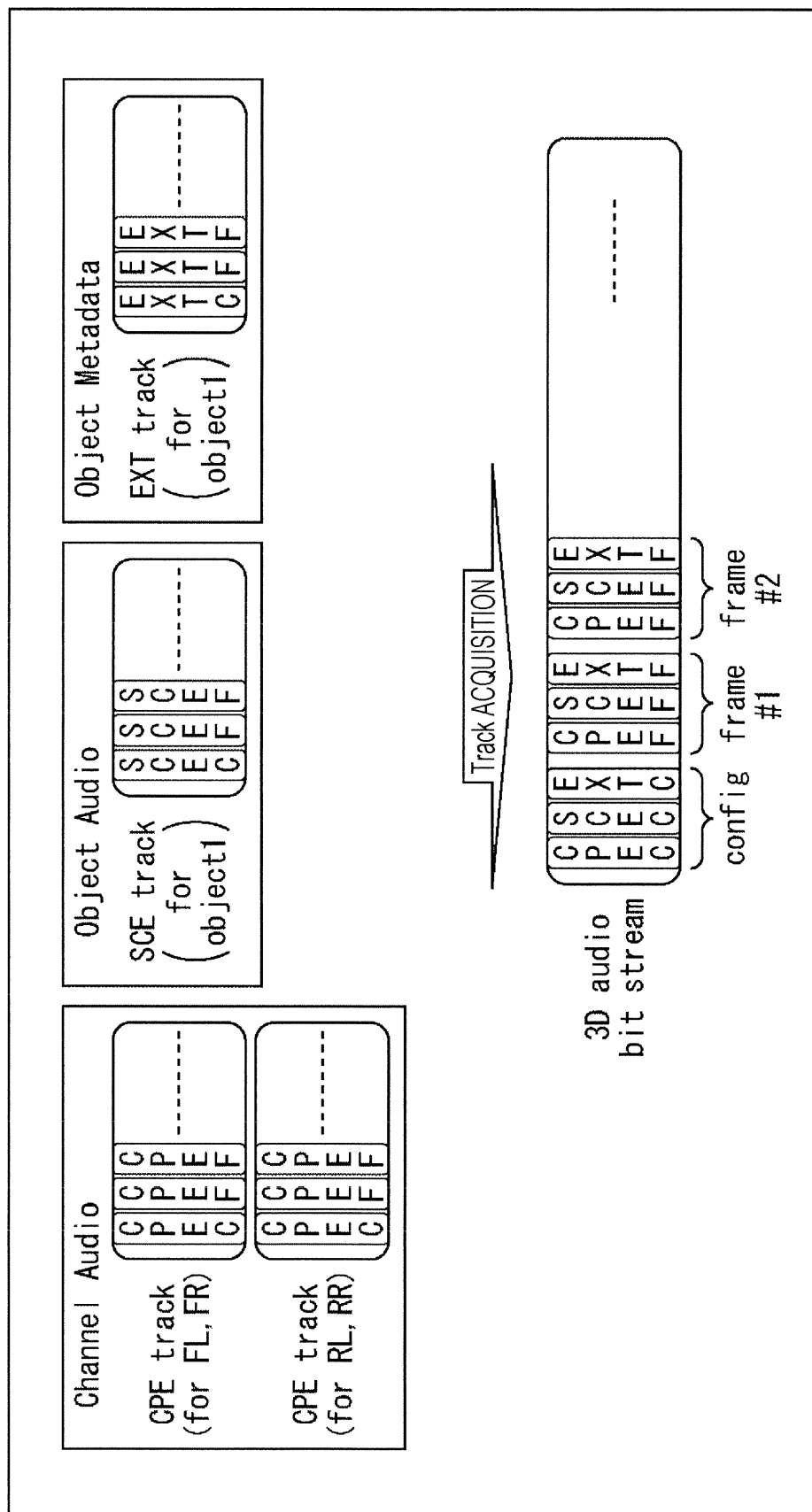


FIG. 78



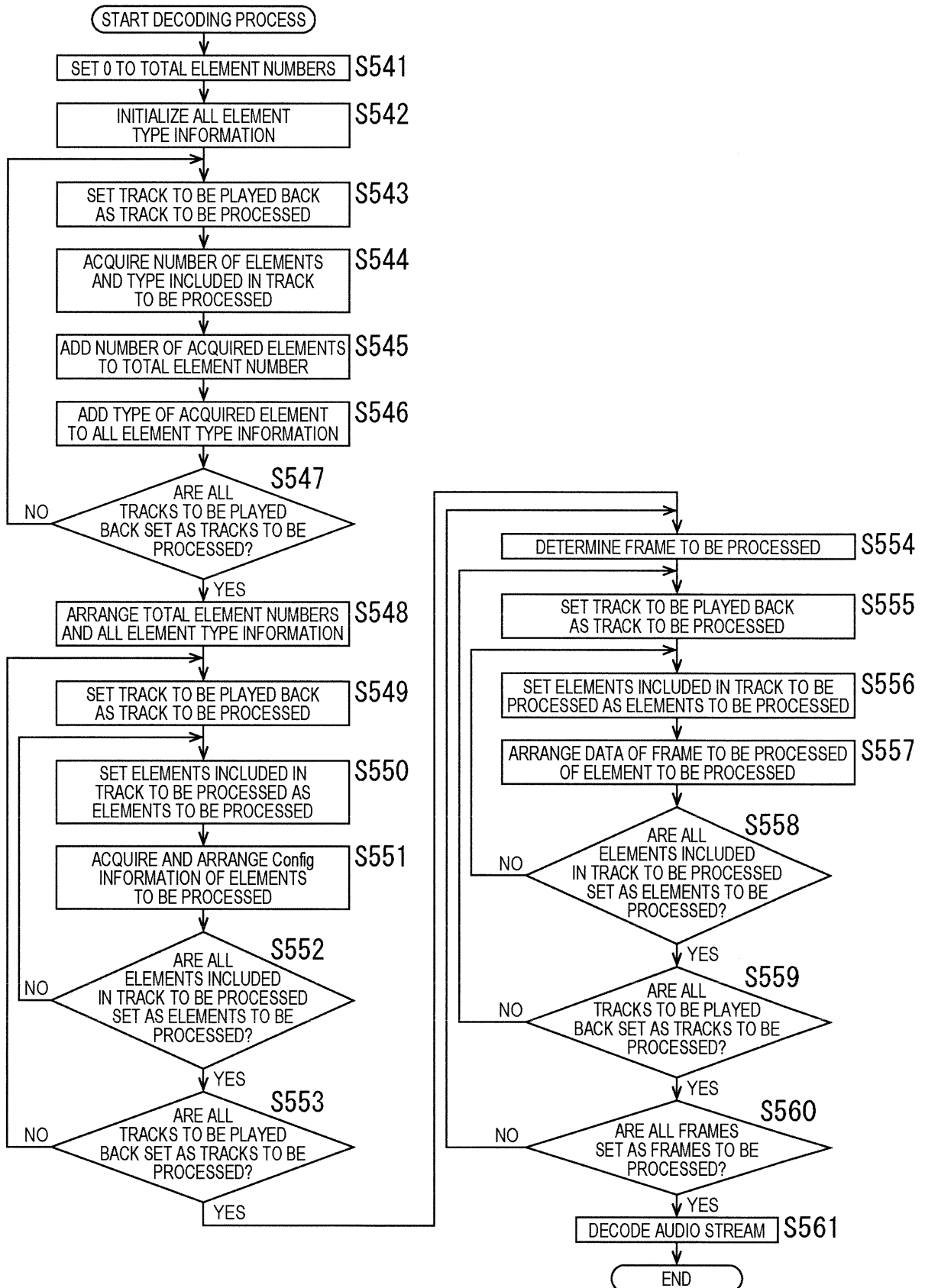


FIG. 80

Syntax	No. of bits	Mnemonic
<pre>mpeg3daDecoderConfig() { numElements = escapedValue(4,8,16) + 1; elementLengthPresent 1 for (elemIdx=0; elemIdx<numElements; ++elemIdx) { usacElementType[elemIdx] switch (usacElementType[elemIdx]) { case ID_USAC_SCE: mpeg3daSingleChannelElementConfig(sbrRatioIndex); break; case ID_USAC_CPE: mpeg3daChannelPairElementConfig(sbrRatioIndex); break; case ID_USAC_LFE: mpeg3daLfeElementConfig(); break; case ID_USAC_EXT: mpeg3daExtElementConfig(); break; } } }</pre>		
NOTE: mpeg3daSingleChannelElementConfig(), mpeg3daChannelPairElementConfig(), mpeg3daLfeElementConfig() and mpeg3daExtElementConfig() signaled at position elemIdx refer to the corresponding elements in mpeg3daFrame() at the respective position elemIdx.		

FIG. 81

Syntax	No. of bits	Mnemonic
<pre> mpeg3daExtElementConfig() { usacExtElementType = escapedValue(4, 8, 16); usacExtElementConfigLength = escapedValue(4, 8, 16); if (usacExtElementDefaultLengthPresent) { usacExtElementDefaultLength = escapedValue(8, 16, 0) + 1; } else { usacExtElementDefaultLength = 0; } usacExtElementPayloadFrag; switch (usacExtElementType) { case ID_EXT_ELE_FILL: break; case ID_EXT_ELE_MPEGS: SpatialSpecificConfig(); break; case ID_EXT_ELE_SAOC: SAOCSpecificConfig(); break; case ID_EXT_ELE_SAOC_3D: SAOC3DSpecificConfig(); break; case ID_EXT_ELE_AUDIOPREROLL: break; case ID_EXT_ELE_UNI_DRC: mpeg3daUniDrcConfig(); break; case ID_EXT_ELE_OBJ_METADATA: ObjectMetadataConfig(); break; case ID_EXT_ELE_HOA: HOAConfig(); break; case ID_EXT_ELE_EXTRACTOR: ExtractorConfig(); break; default: while (usacExtElementConfigLength--) { tmp; } break; } } </pre>	1 1 NOTE 8	uimbsf uimbsf uimbsf

FIG. 82

Syntax	No. of bits	Mnemonic
<pre>ExtractorConfig() { usacElementType Extractor; switch (usacElementType[elemIdx]) { case ID_USAC_SCE: break; case ID_USAC_CPE: break; case ID_USAC_LFE: break; case ID_USAC_EXT: usacExtElementTypeExtractor = escapedValue(4, 8, 16); break; } configLength; configOffset; }</pre>	2	uimbsf

FIG. 83

usacExtElementType	usacExtElementSegmentData represents:
ID_EXT_ELE_FIL	Series of fill_byte
ID_EXT_ELE_MPEGS	SpatialFrame()
ID_EXT_ELE_SAOC	SaocFrame()
ID_EXT_ELE_AUDIOPREROLL	AudioPreRoll()
ID_EXT_ELE_UNI_DRC	uniDrcGain() as defined in ISO/IEC 23003-4
ID_EXT_ELE_OBJ_METADATA	object_metadata()
ID_EXT_ELE_SAOC_3D	Saoc3DFrame()
ID_EXT_ELE_HOA	HOAFrame()
ID_EXT_ELE_EXTRACTOR	ExtractorMetadata()
unknown	unknown data. The data block shall be discarded.

FIG. 84

Syntax	No. of bits	Mnemonic
ExtractorMetadata()		
{		
elementLength;	16	uimsbf
elementOffset;	64	uimsbf
}		

FIG. 85

Syntax	No. of bits	Mnemonic
mpegh3daDecoderConfig()		
{		
numElements = escapedValue(4,8,16) + 1;		
elementLengthPresent	1	
flagExtractor	1	uimsbf
for (elemIdx=0; elemIdx<numElements; ++elemIdx) {		
usacElementType[elemIdx]	2	uimsbf
switch (usacElementType[elemIdx]) {		
case ID_USAC_SCE:		
if (!flagExtractor) {		
mpegh3daSingleChannelElementConfig(sbrRatIdx);		
}		
break;		
case ID_USAC_CPE:		
if (!flagExtractor) {		
mpegh3daChannelPairElementConfig(sbrRatIdx);		
}		
break;		
case ID_USAC_LFE:		
if (!flagExtractor) {		
mpegh3daLfeElementConfig();		
}		
break;		
case ID_USAC_EXT:		
if (flagExtractor) {		
usacExtElementType = escapedValue(4, 8, 16);		
} else {		
mpegh3daExtElementConfig();		
}		
break;		
}		
config length		
config offset		
}		

FIG. 86

[illegible]

FIG. 87

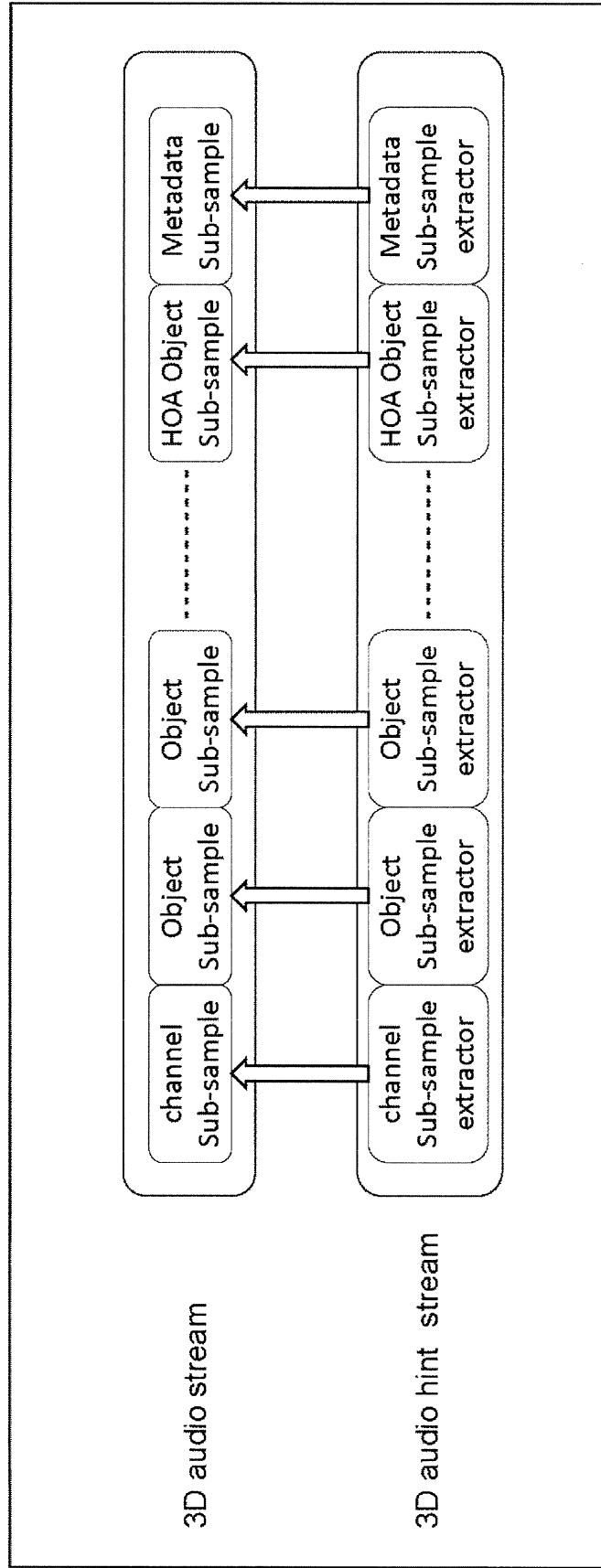


FIG. 88

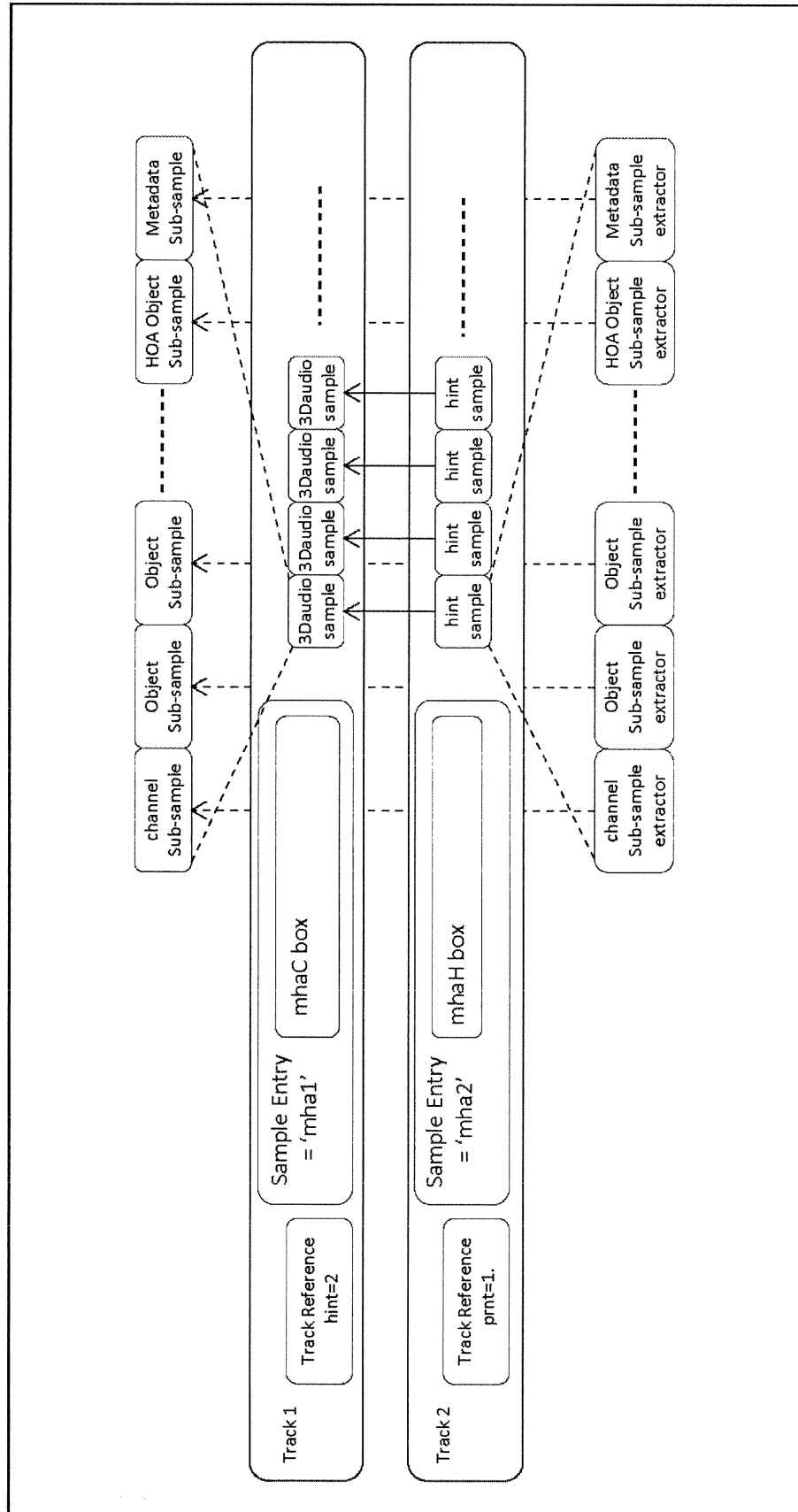


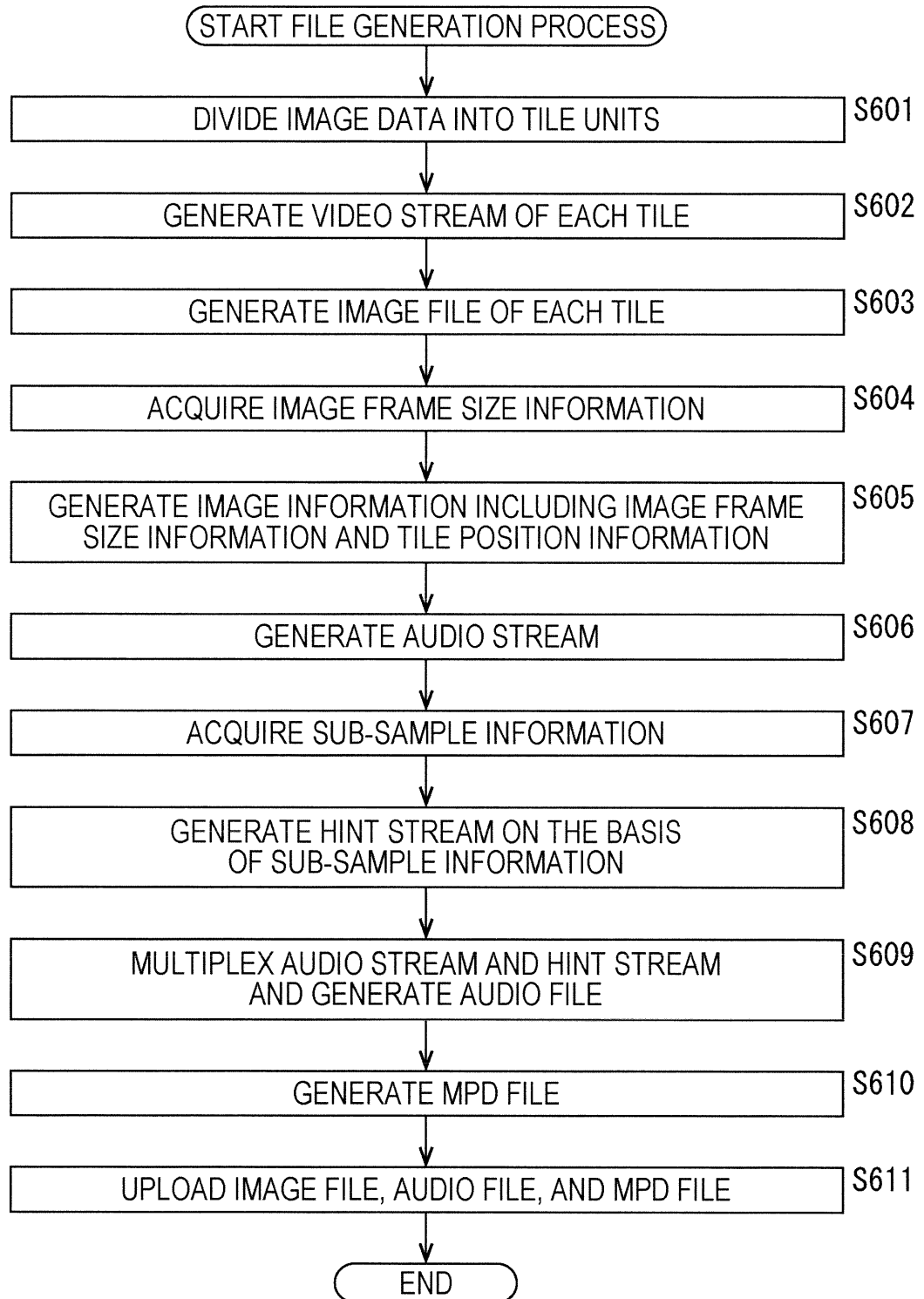
FIG. 89

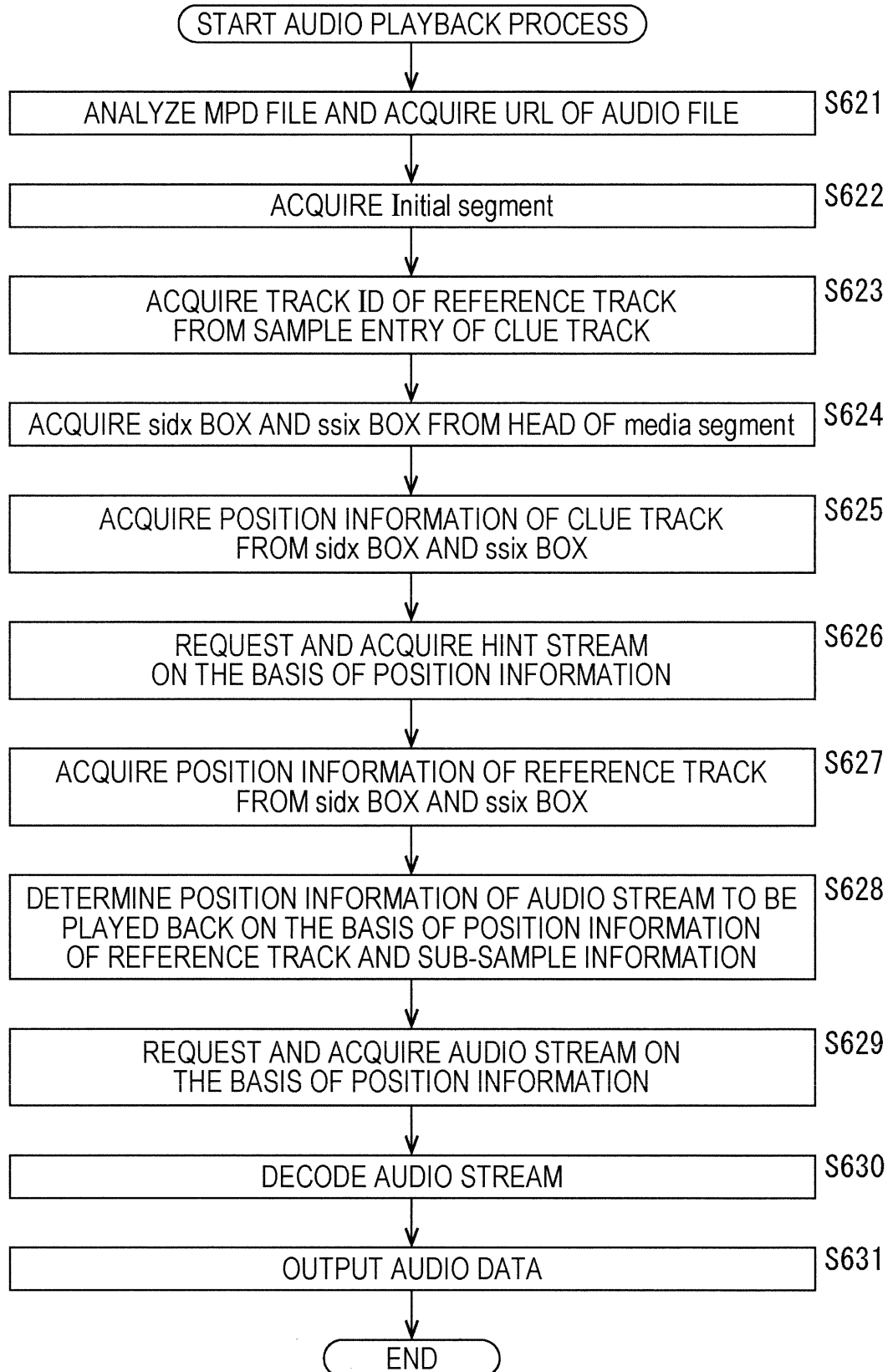
FIG. 90

FIG. 91

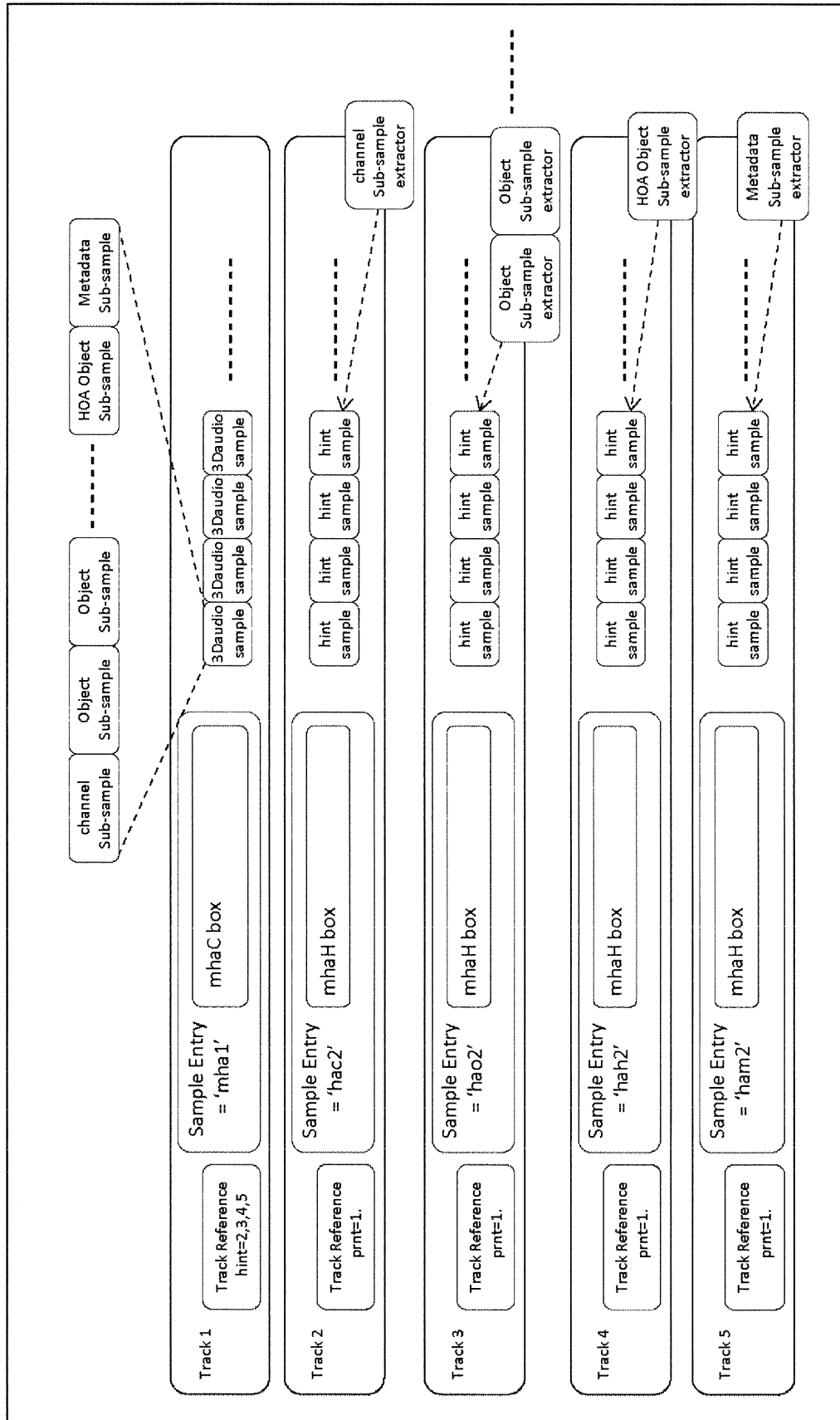


FIG. 92

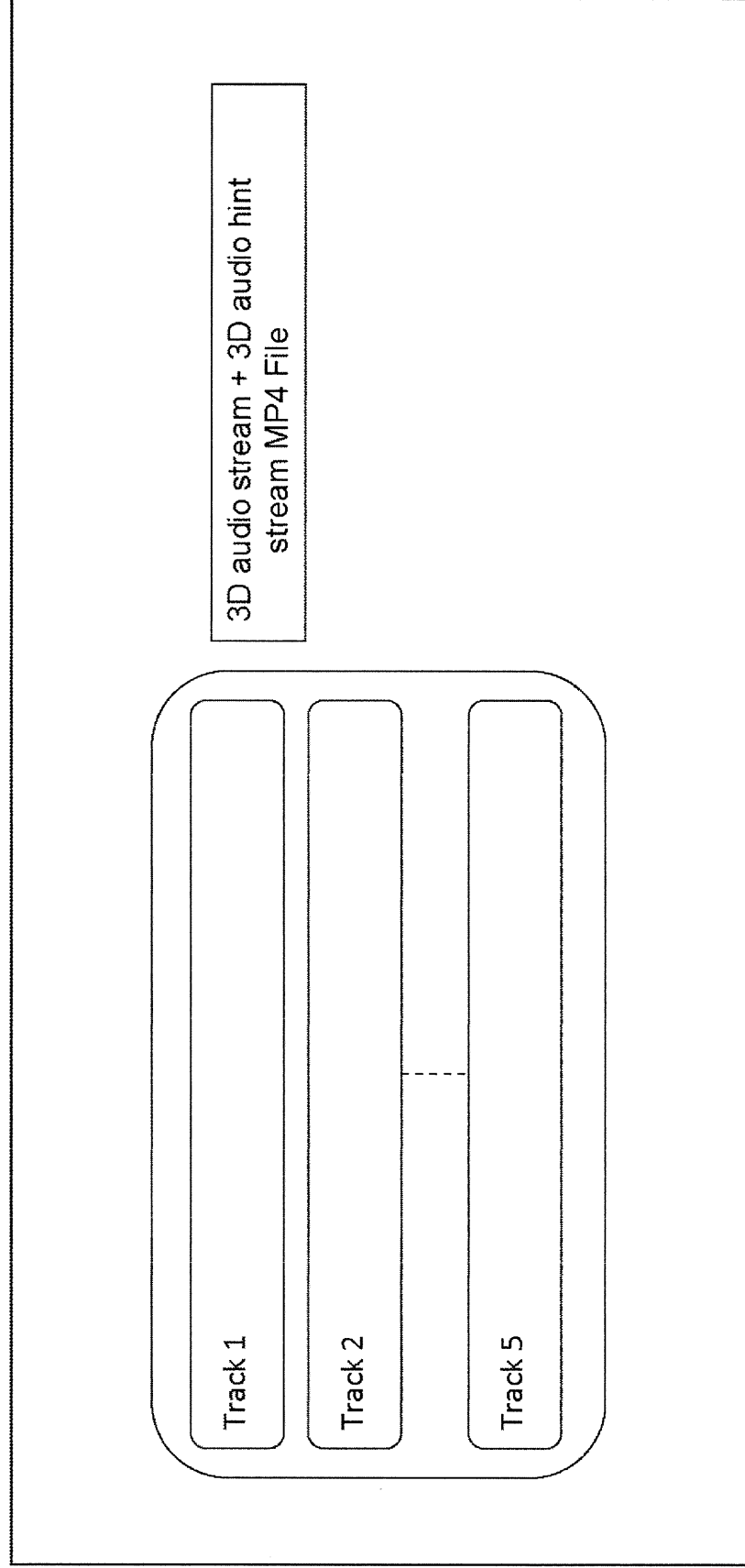


FIG. 93

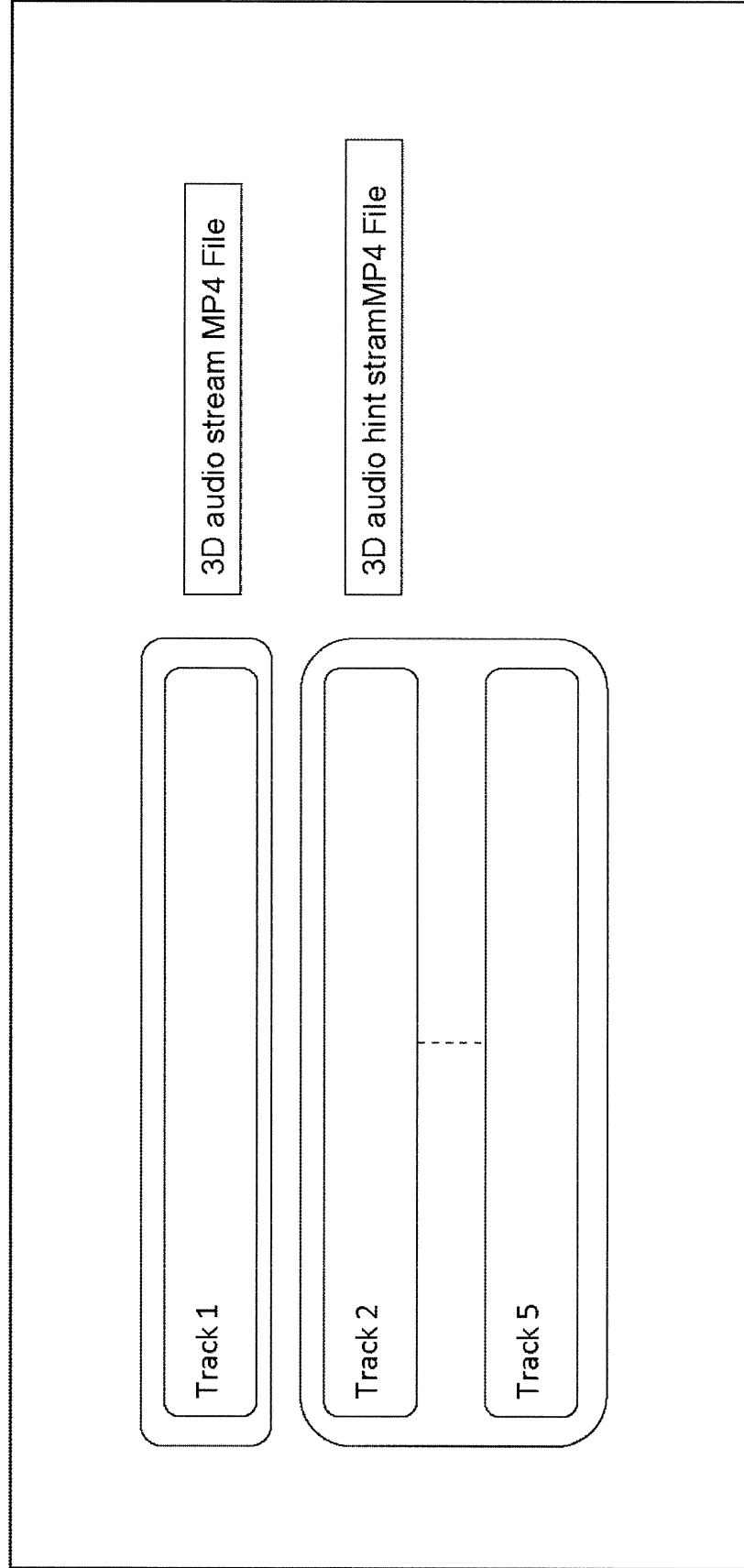


FIG. 94

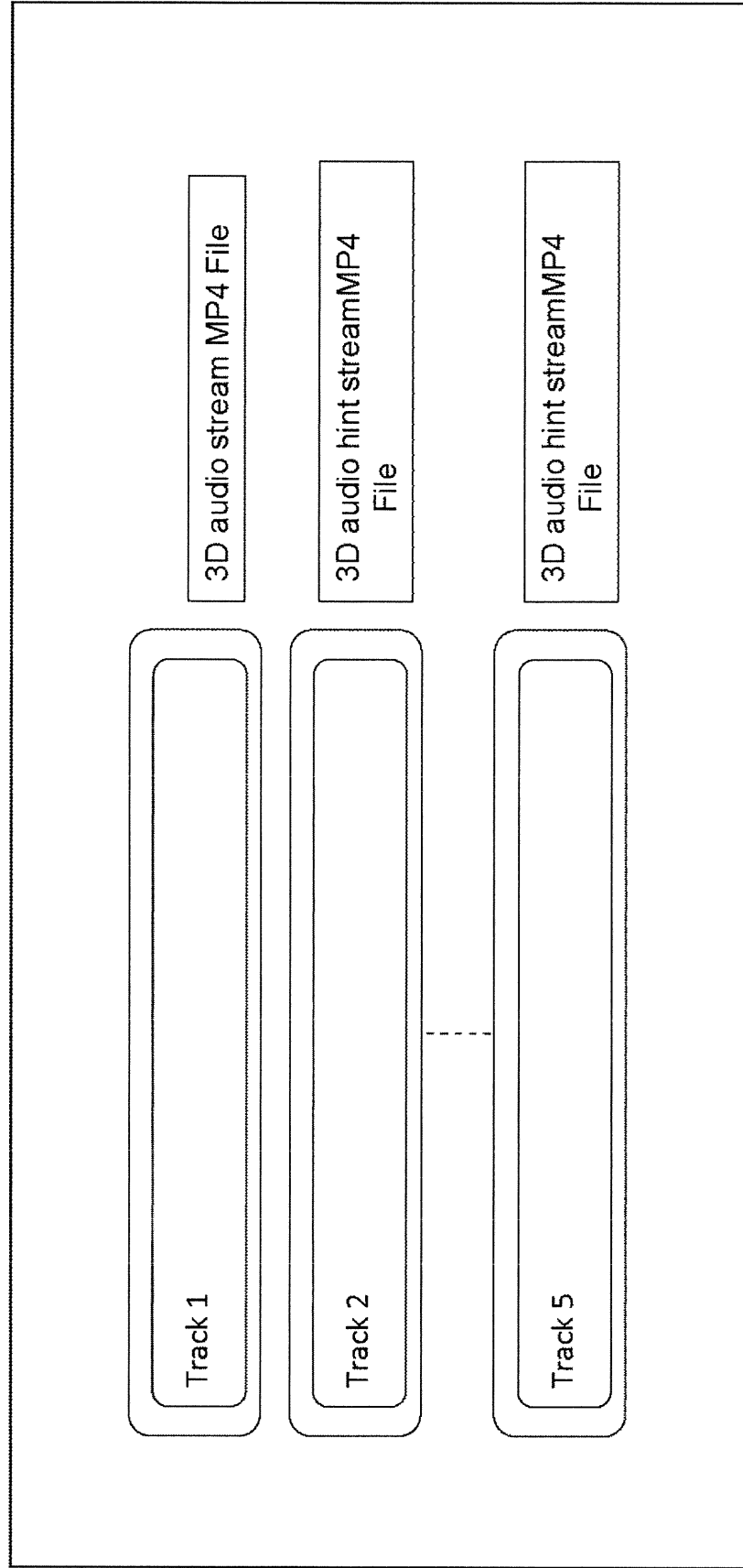


FIG. 95

