

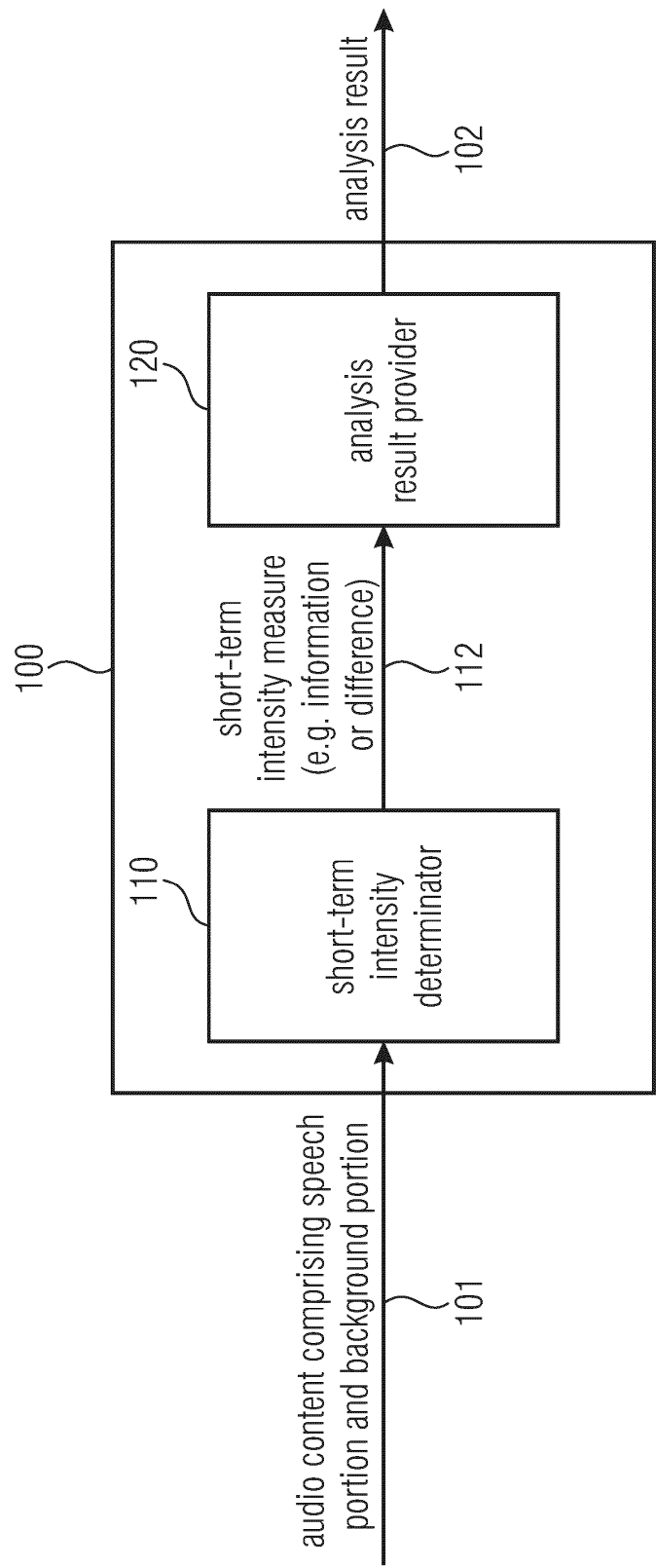


Fig. 1

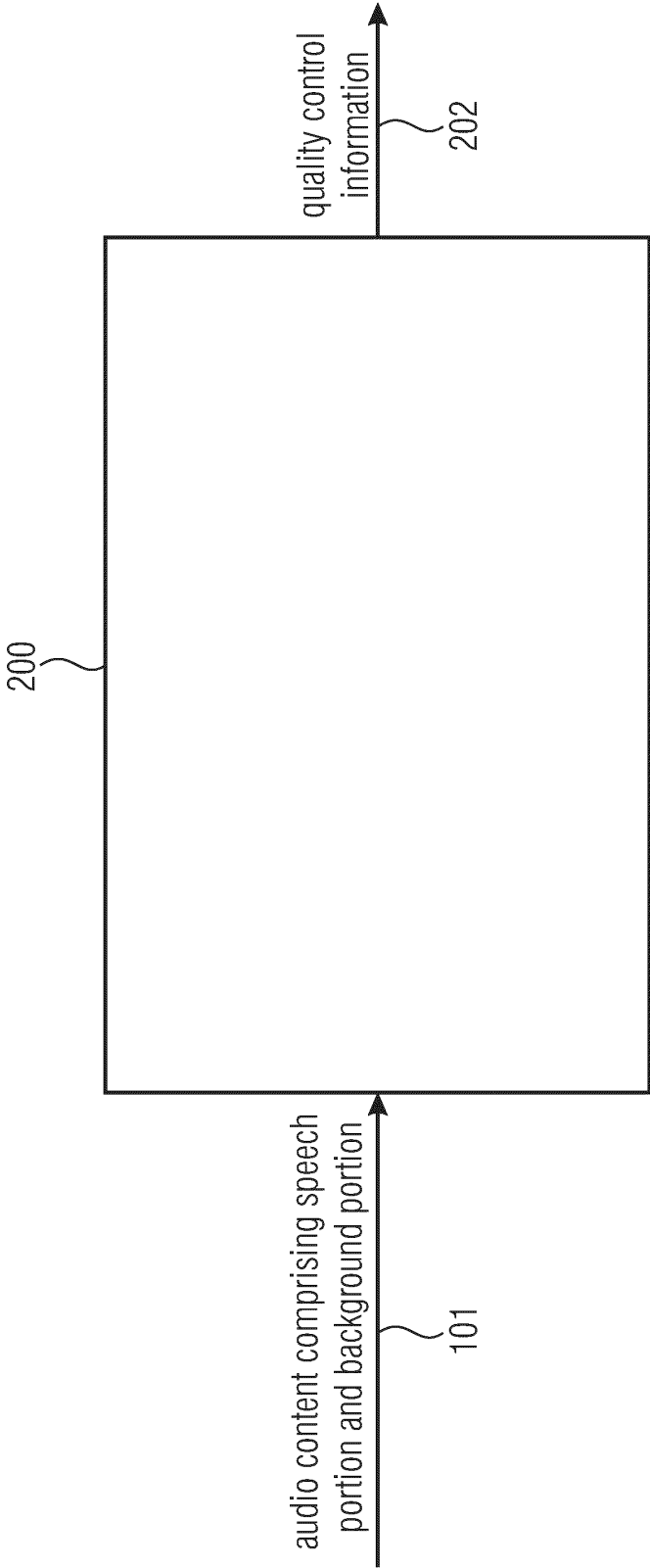


Fig. 2

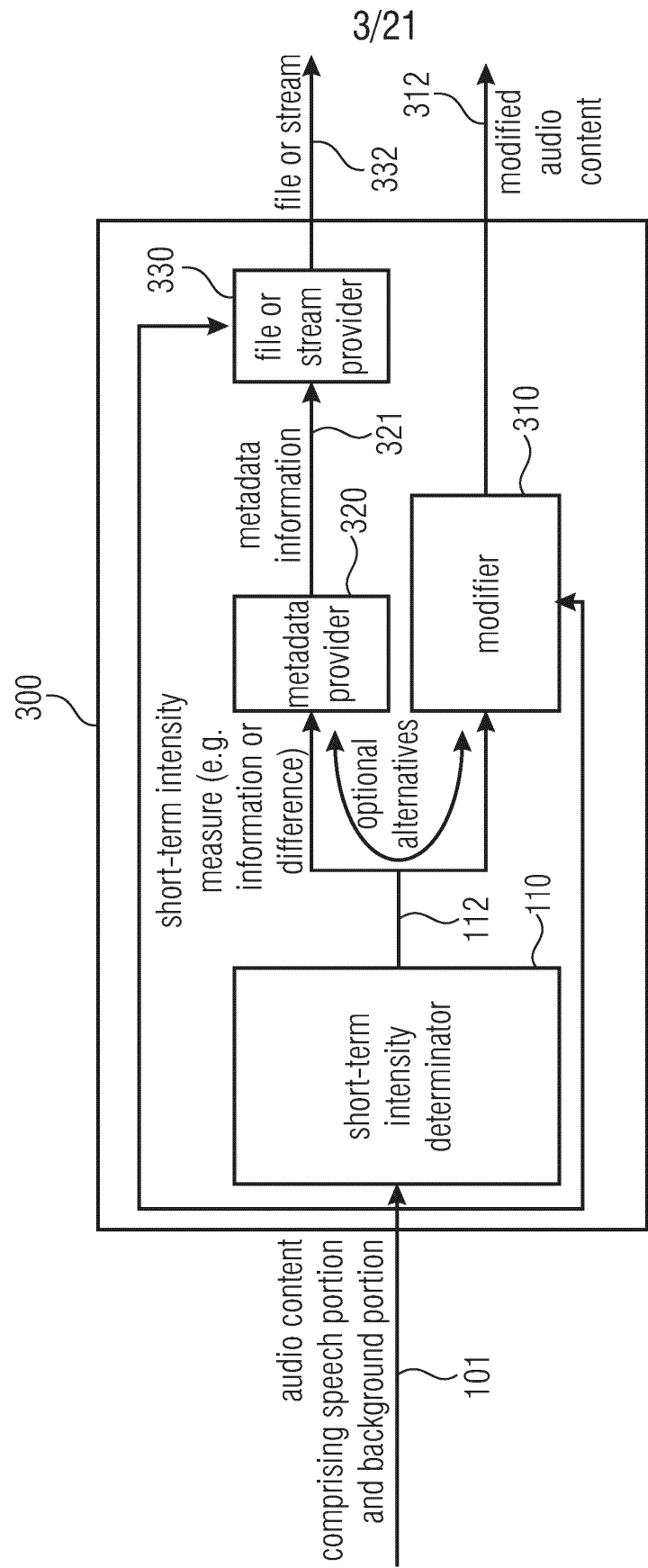


Fig. 3

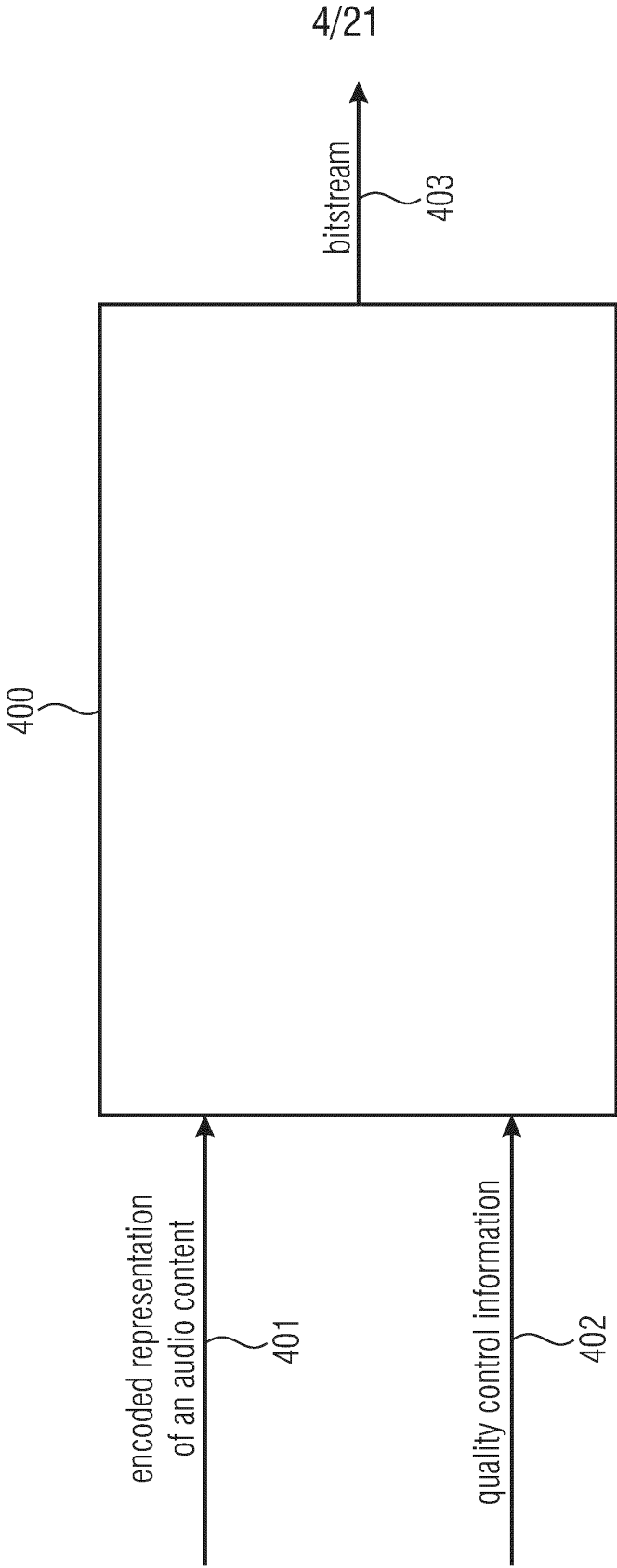


Fig. 4

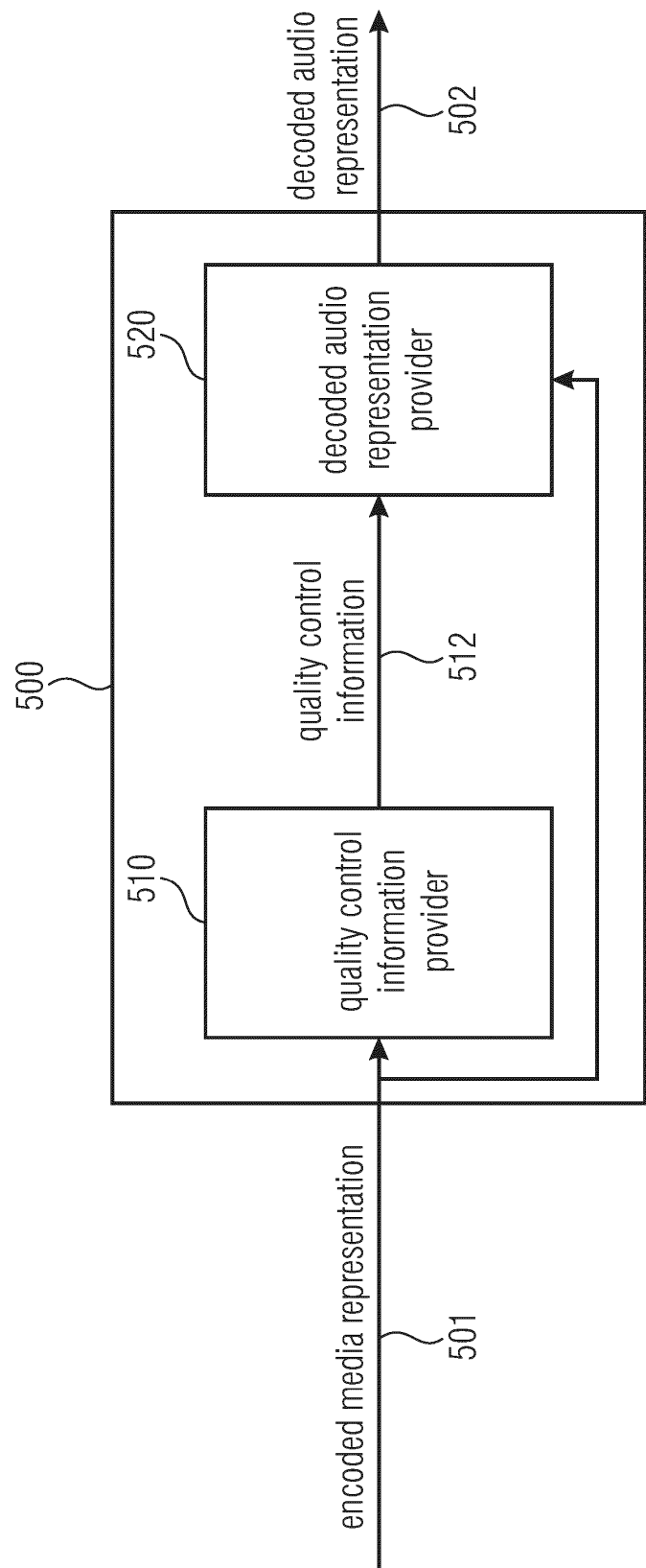


Fig. 5

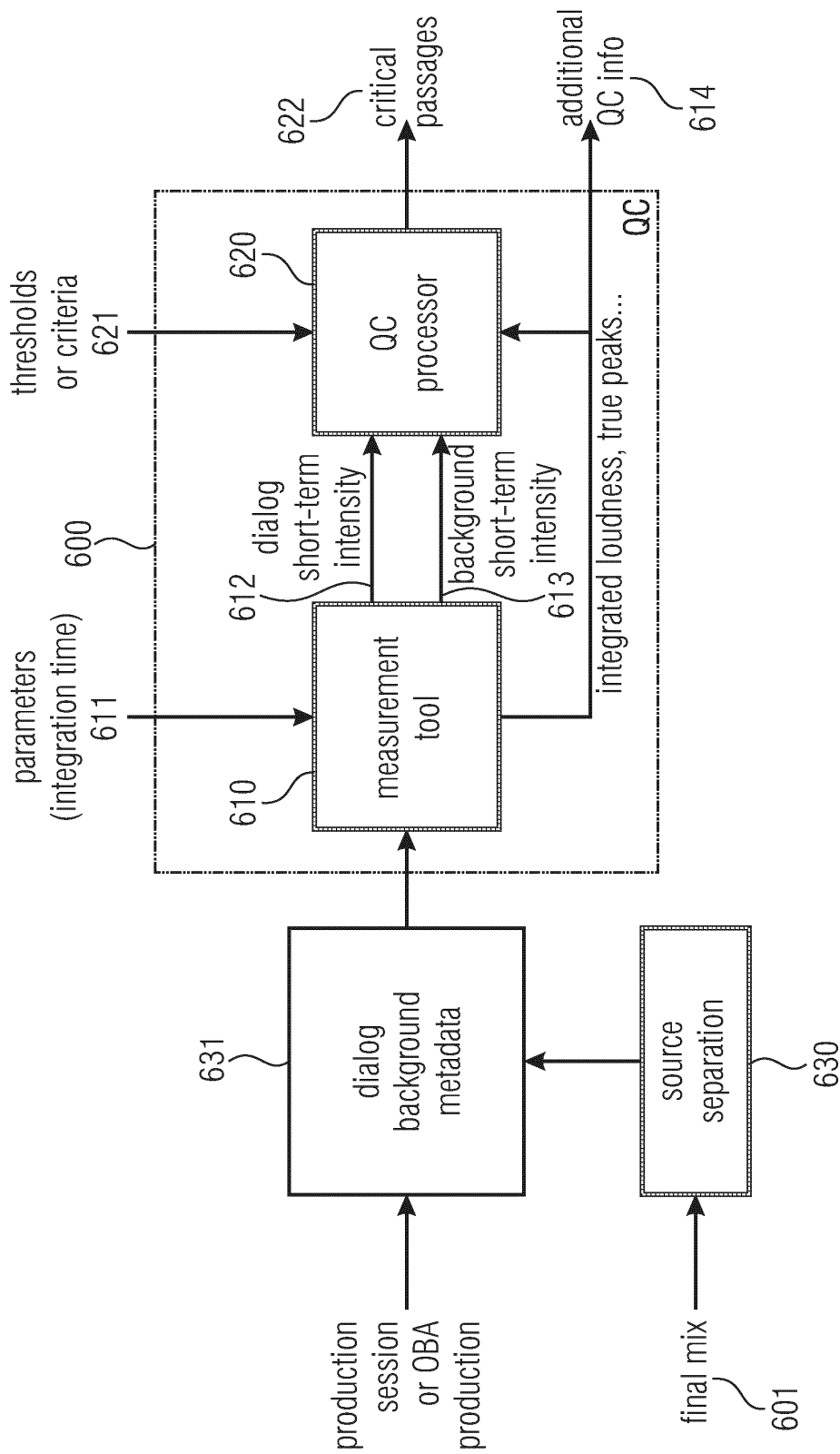


Fig. 6

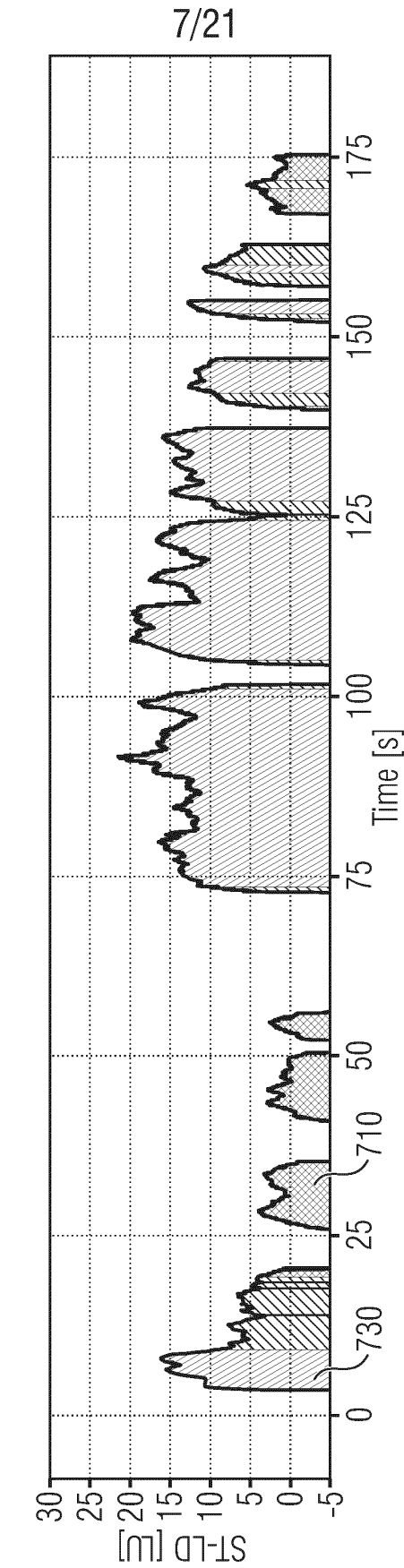


Fig. 7

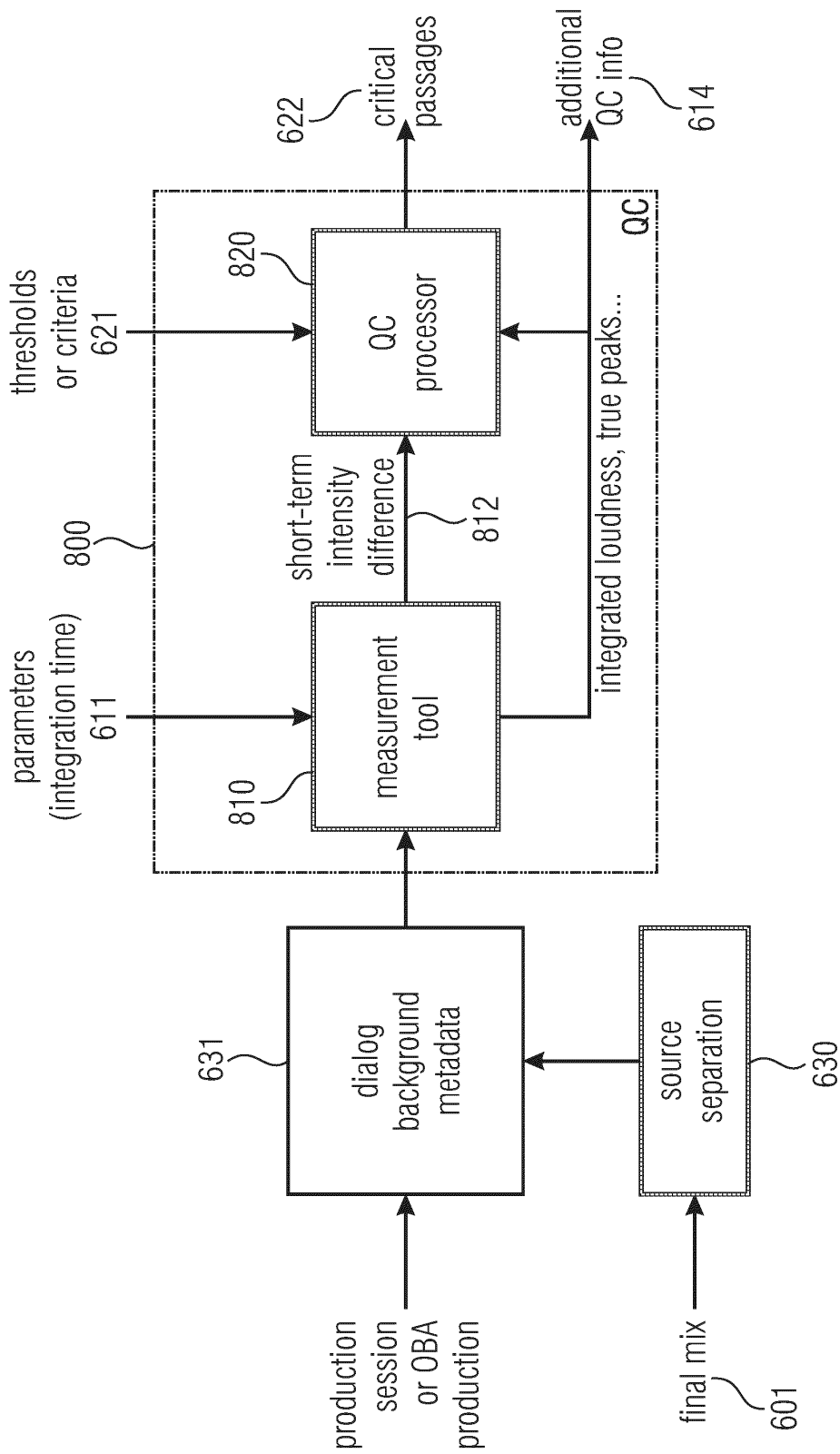


Fig. 8

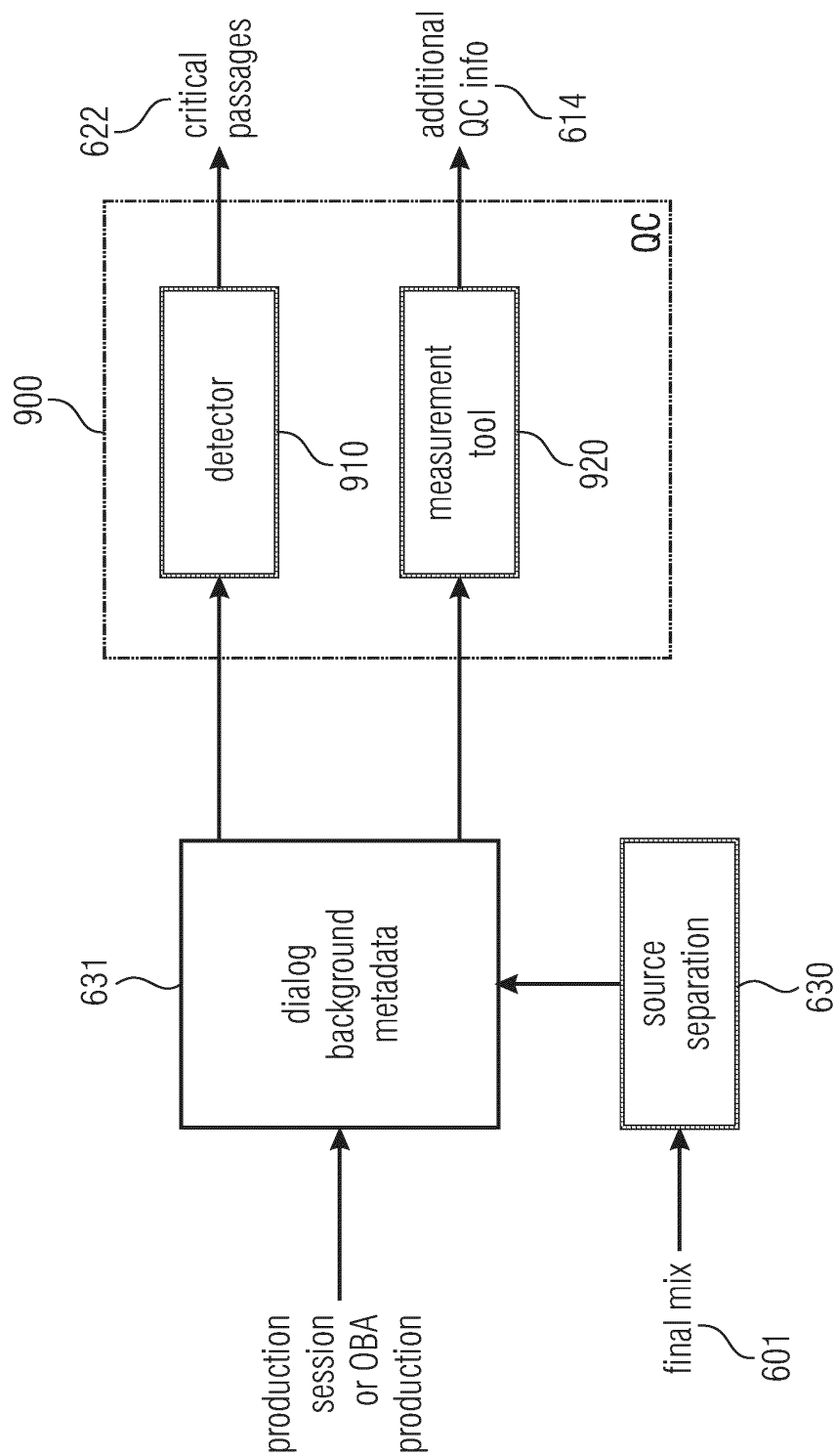


Fig. 9

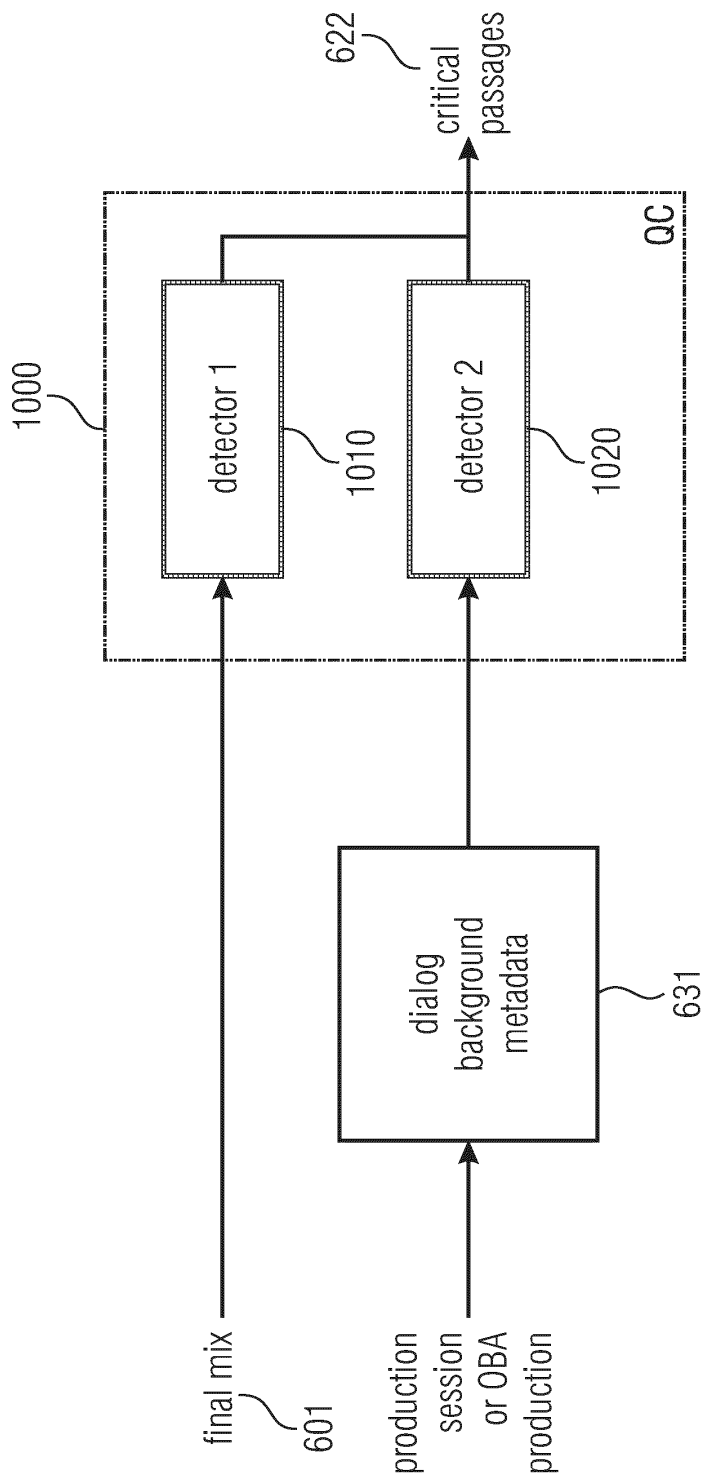


Fig. 10

11/21

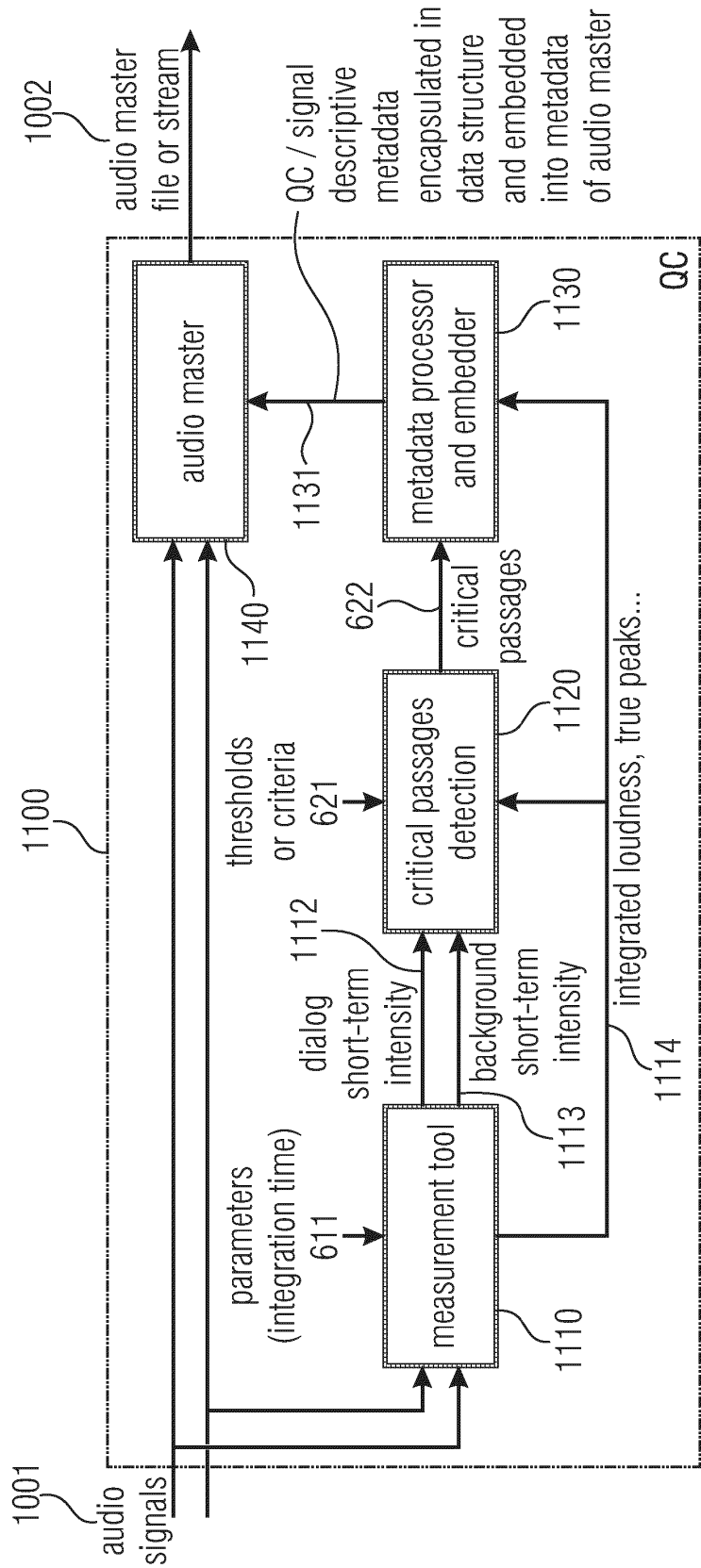


Fig. 11

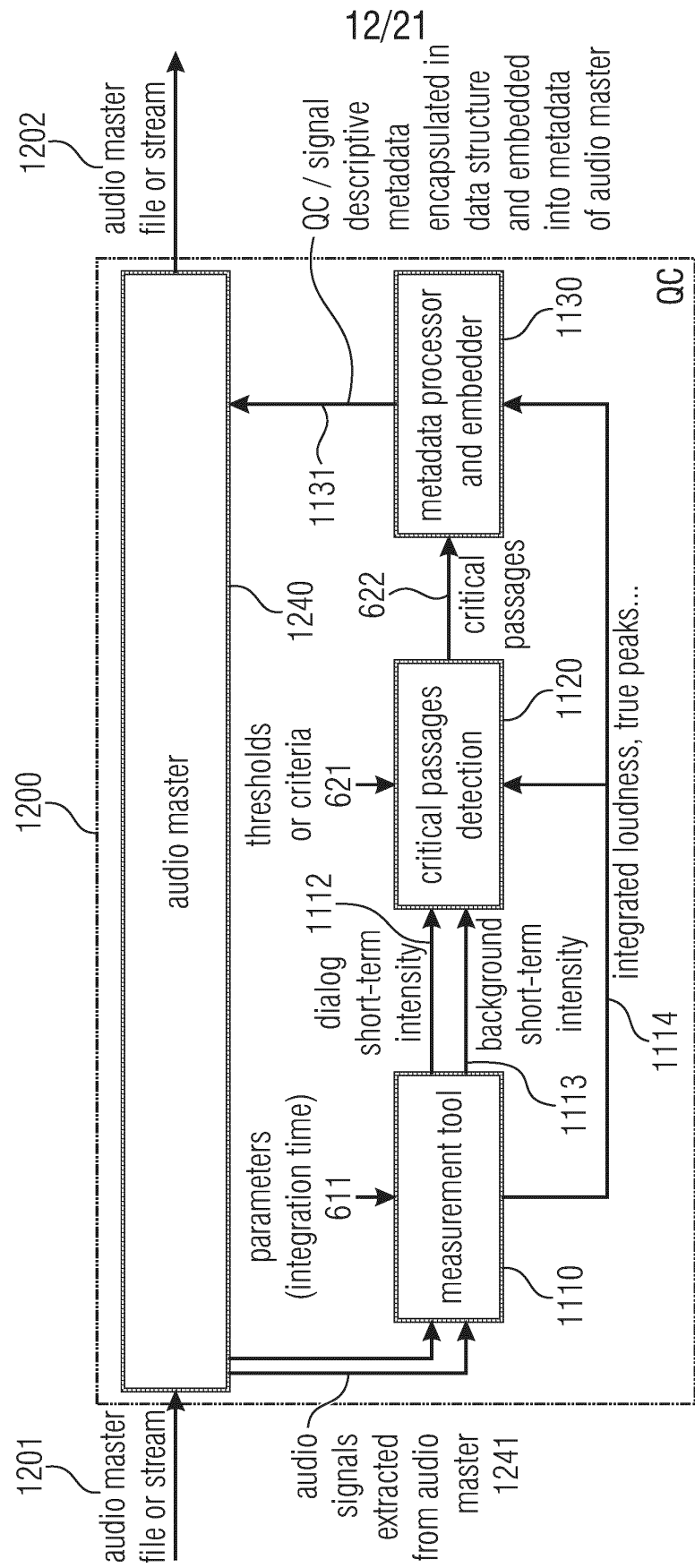


Fig. 12

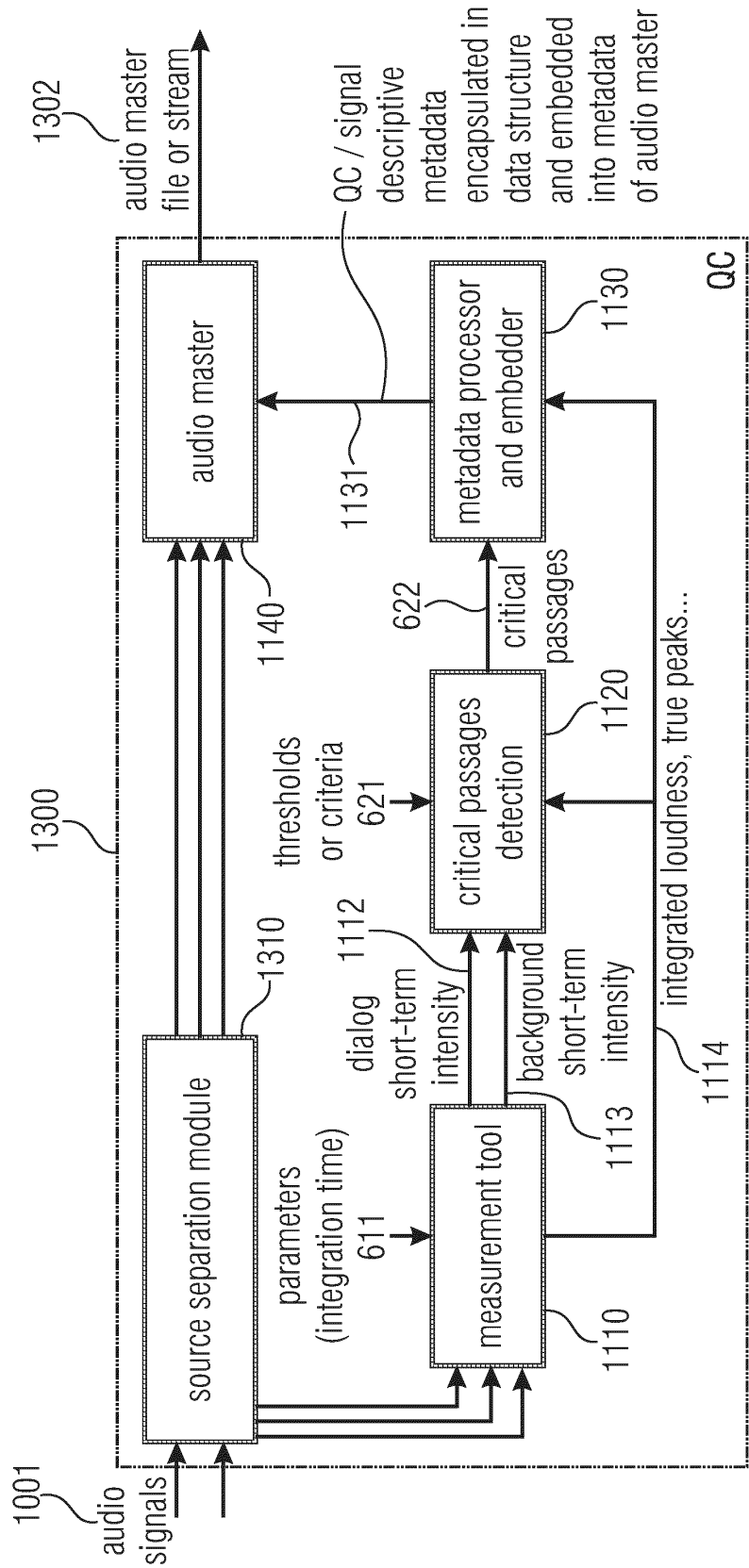


Fig. 13

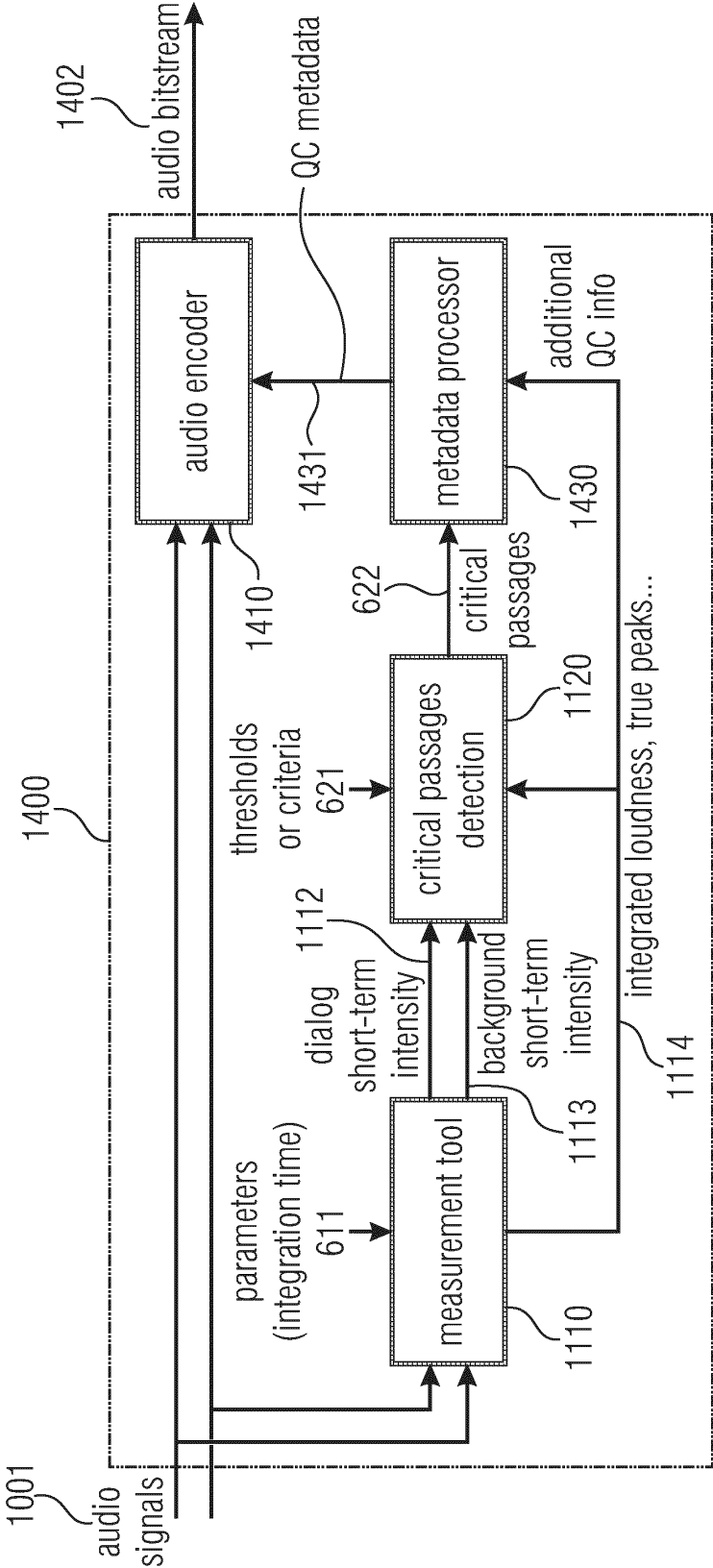


Fig. 14

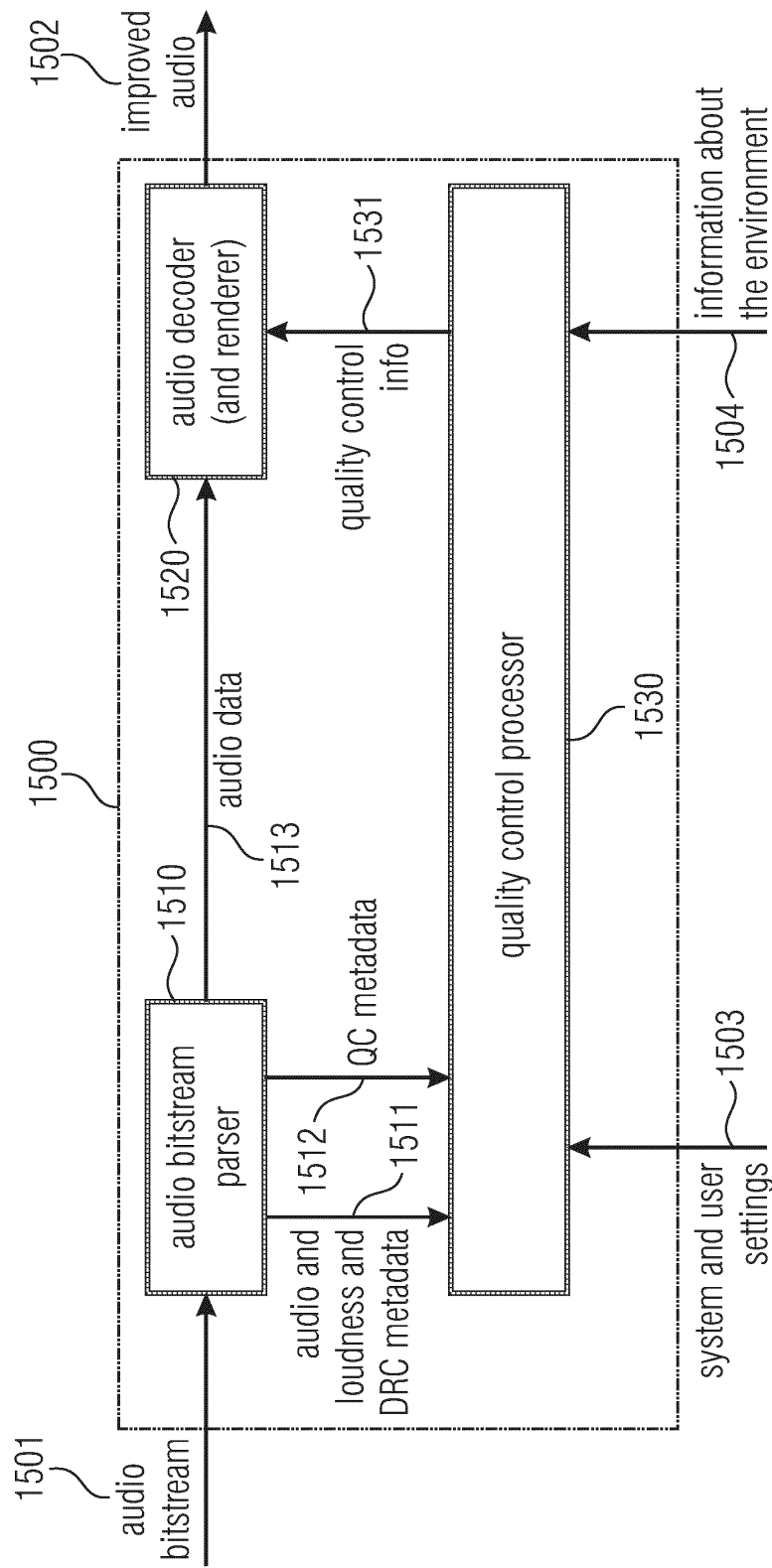


Fig. 15

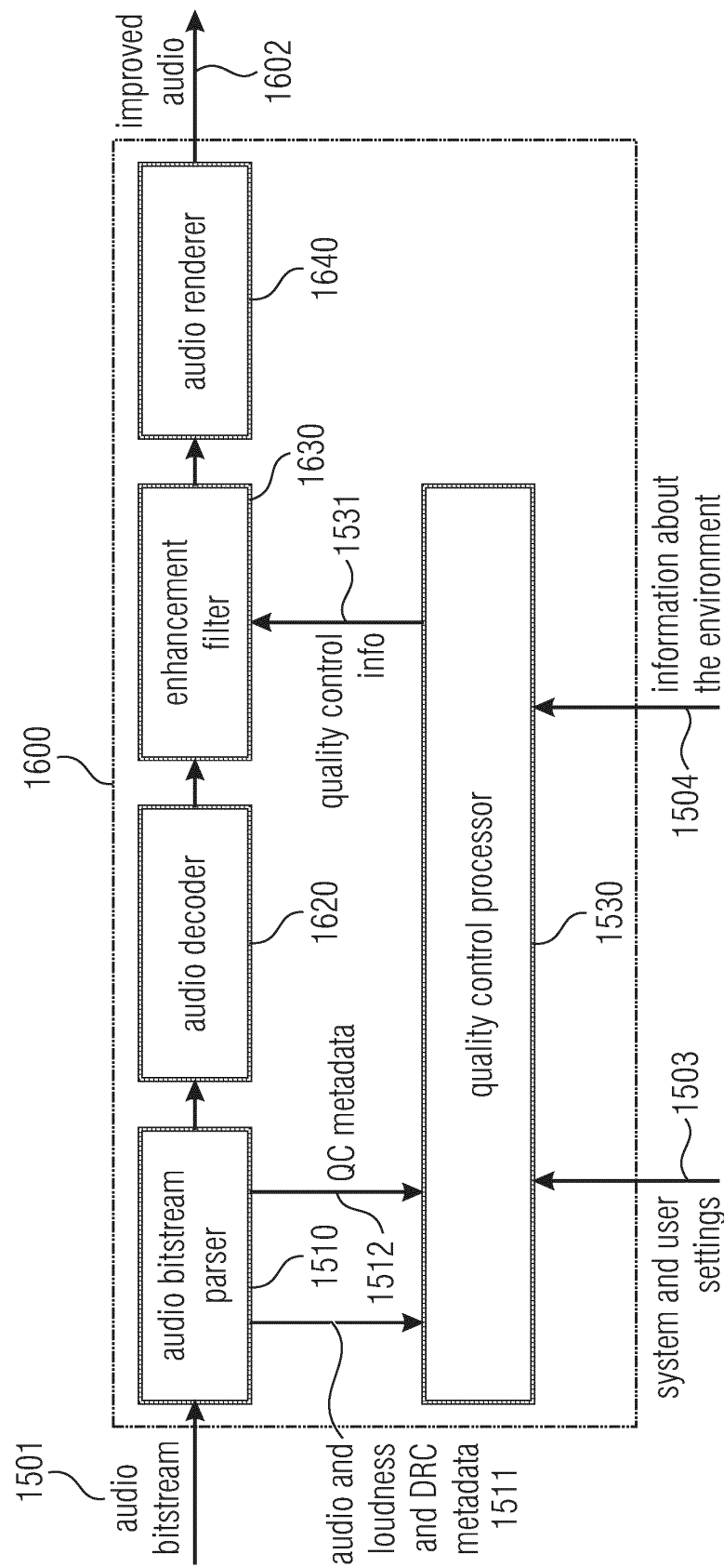


Fig. 16

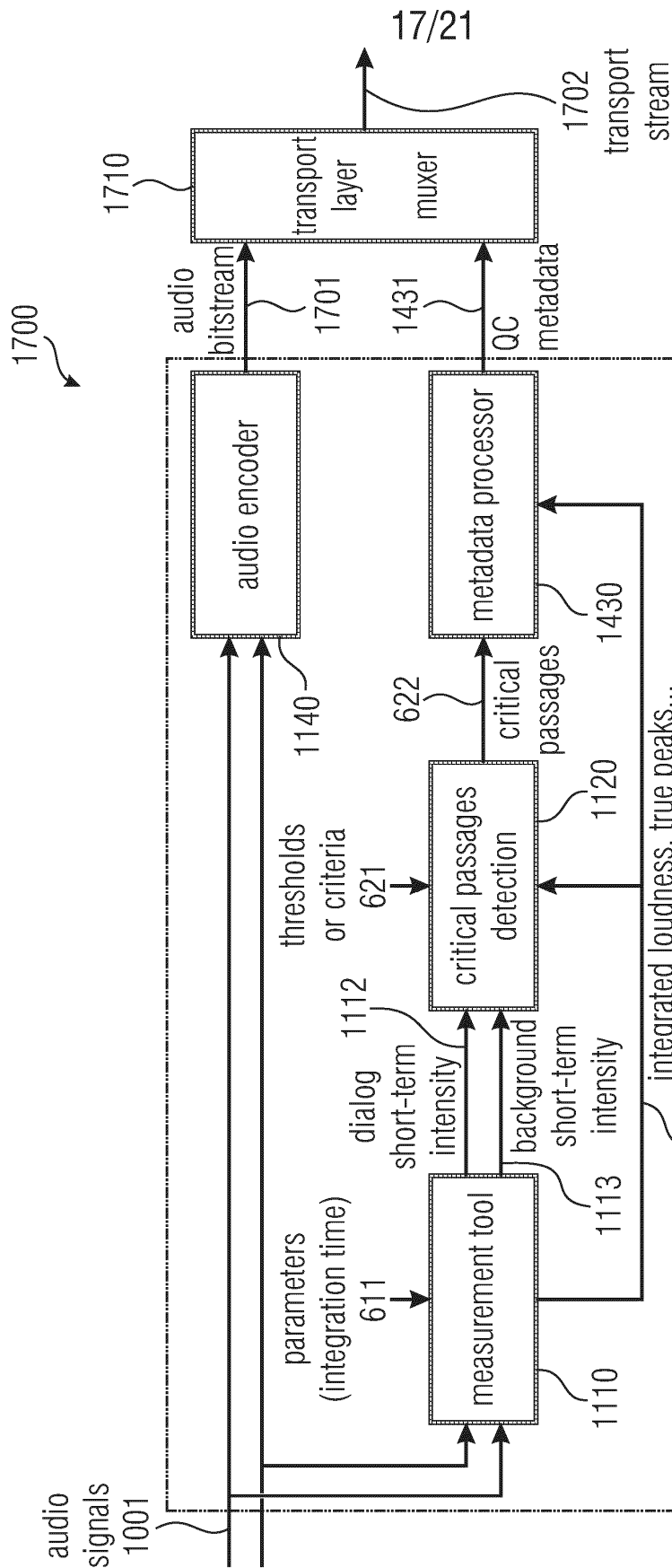


Fig. 17

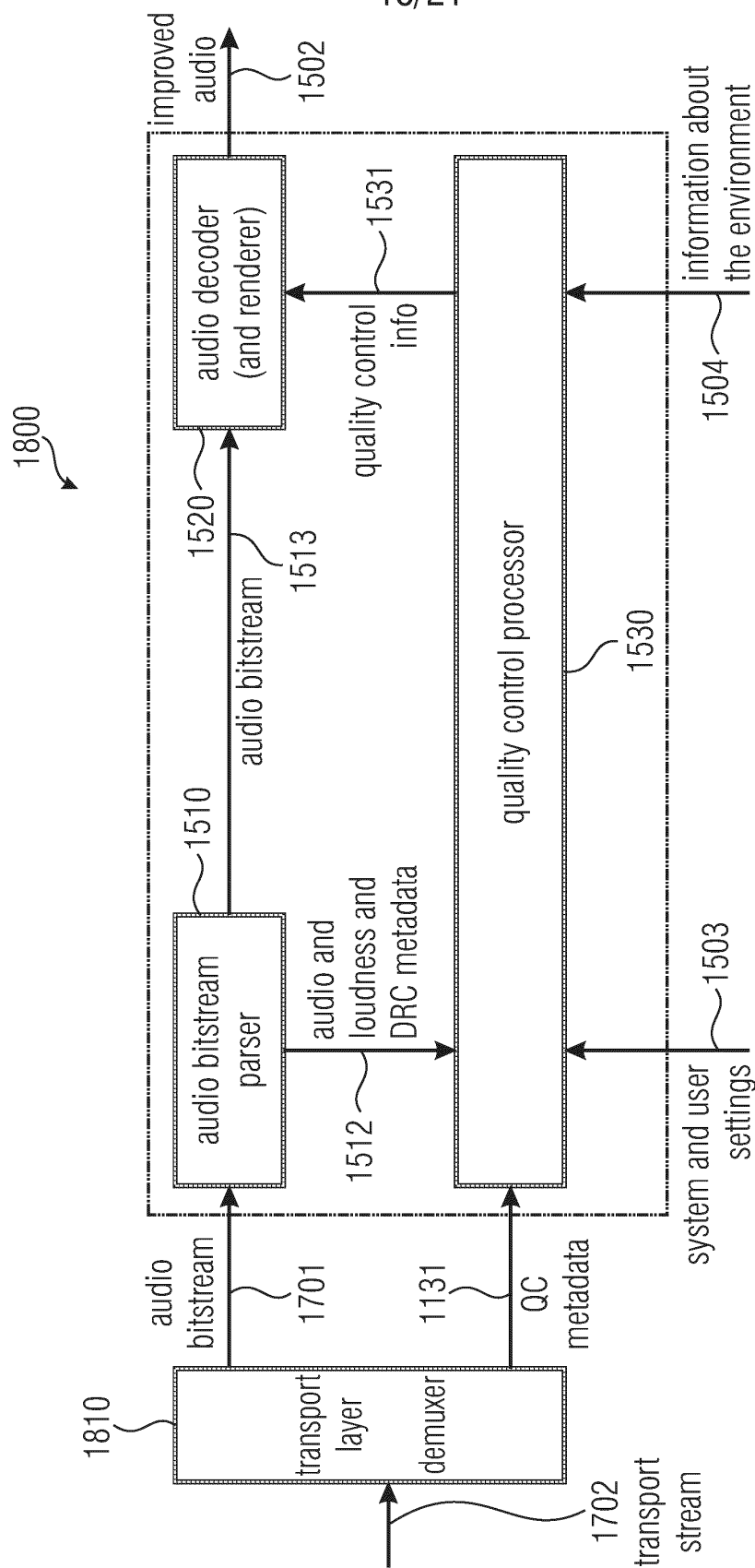


Fig. 18

syntax	No. of bits	mnemonic
MHASPacketPayload(MHASPacketType)		
{		
switch (MHASPacketType) {		
case PACTYP_SYNC:	8	uimshf
/*syncword*/		
break;		
case PACTYP_MPEGH3DADCFG:		
mpegh3daConfig();		
break;		
case PACTYP_MPEGH3DAFRAME:		
mpegh3daFrame();		
break;		
case PACTYP_AUDIOSCENEINFO:		
mae_AudioSceneInfo();		
break;		
case PACTYP_FILLDATA:		
for (i=0; i<MHASPacketLength; i++) {		
mhas_fill_data_byte(i);	8	bslbf
}		
break;		
case PACTYP_SYNCGAP:		
syncSpacingLength = escapedValue(16,24,24);		
break;		
case PACTYP_MARKER:		
for (i=0; i<MHASPacketLength; i++) {		
marker_byte(i);	8	bslbf
}		
break;		
case PACTYP_CRC16:		
mhasParity16Data;		
break;		
case PACTYP_CRC32:		
mhasParity32Data;		
break;		
case PACTYP_GLOBAL_CRC16:		
global_CRC_type;		
numProtectedPackets;		
mhasParity16Data;		
break;		
case PACTYP_GLOBAL_CRC32:		
global_CRC_type;		
numProtectedPackets;		
mhasParity32Data;		
break;		
case PACTYP_DESCRIPTOR:		
for (i=0; i<MHASPacketLength; i++) {		
mhas_descriptor_data_byte(i);	8	bslbf
}		
}		
}		

syntax	No. of bits	mnemonic
{		
break;		
case PACTYP_USERINTERACTION:		
mpegh3daElementInteraction();		
break;		
case PACTYP_LOUDNESS_DRC:		
mpegh3daLoudnessDrcInterface();		
break;		
case PACTYP_BUFFERINFO:		
mhas_buffer_fullness_present;		
if (mhas_buffer_fullness_present)		
mhas_buffer_fullness =		
escapedValue(15,24,32);		
}		
break;	1	uimshf
case PACTYP_AUDIOTRUNCATION:		
audioTruncationInfo();		
break;		
case PACTYP_GENDATA:		
genDataPayload();		
break;		
case PACTYP_EARCON:		
earconInfo();		
break;		
case PACTYP_PCMMCONFIG:		
pcmDataConfig();		
break;		
case PACTYP_PCMDATA:		
pcmDataPayload();		
break;		
case PACTYP_LOUDNESS:		
mpegh3daLoudnessInfoSet();		
break;		
case PACTYP_QUALITYCONTROL:		
audioQualityControlInfo();		
break;		
}		
ByteAlign();		
}		

Fig. 19

MHASPacketType	Value
PACTYP_FILLDATA	0
PACTYP_MPEGH3DACFG	1
PACTYP_MPEGH3DAFRAME	2
PACTYP_AUDIOSCENEINFO	3
/*reserved for ISO use*/	4-5
PACTYP_SYNC	6
PACTYP_SYNCGAP	7
PACTYP_MARKER	8
PACTYP_CRC16	9
PACTYP_CRC32	10
PACTYP_DESCRIPTOR	11
PACTYP_USERINTERACTION	12
PACTYP_LOUDNESS_DRC	13
PACTYP_BUFFERINFO	14
PACTYP_GLOBAL_CRC16	15
PACTYP_GLOBAL_CRC32	16
PACTYP_AUDIOTRUNCATION	17
PACTYP_GENDATA	18
PACTYP_EARCON	19
PACTYP_PCMCONFIG	20
PACTYP_PCMDATA	21
PACTYP_LOUDNESS	22
PACTYP_QUALITYCONTROL	23
/*reserved for ISO use*/	24-127
/*reserved for use outside of ISO scope*/	128-261
/*reserved for ISO use*/	262-389
/*reserved for use outside of ISO scope*/	390-517
Application-specific MHASPacketType values are mandated to be in the space reserved for use outside of ISO scope. These shall be skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions.	

Fig. 20

SUBSTITUTE SHEET (RULE 26)

syntax	No. of bits	mnemonic
audioQualityControlInfo() { qclInfoCount;(optional) for (i=0; i<qclInfoCount, i++) { qclInfoActive; (optional) qclInfoType; (optional) if (qclInfoType == 1) { mae_groupID; (optional) } else if (qclInfoType == 2) { mae_groupPresetID; (optional) } qclInfo(); } qclInfoExtensionPresent;(optional) if (qclInfoExtensionPresent == 1) { qclInfoExtension();(optional) } }	6 3 2 7 5 1	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf

Fig. 21