

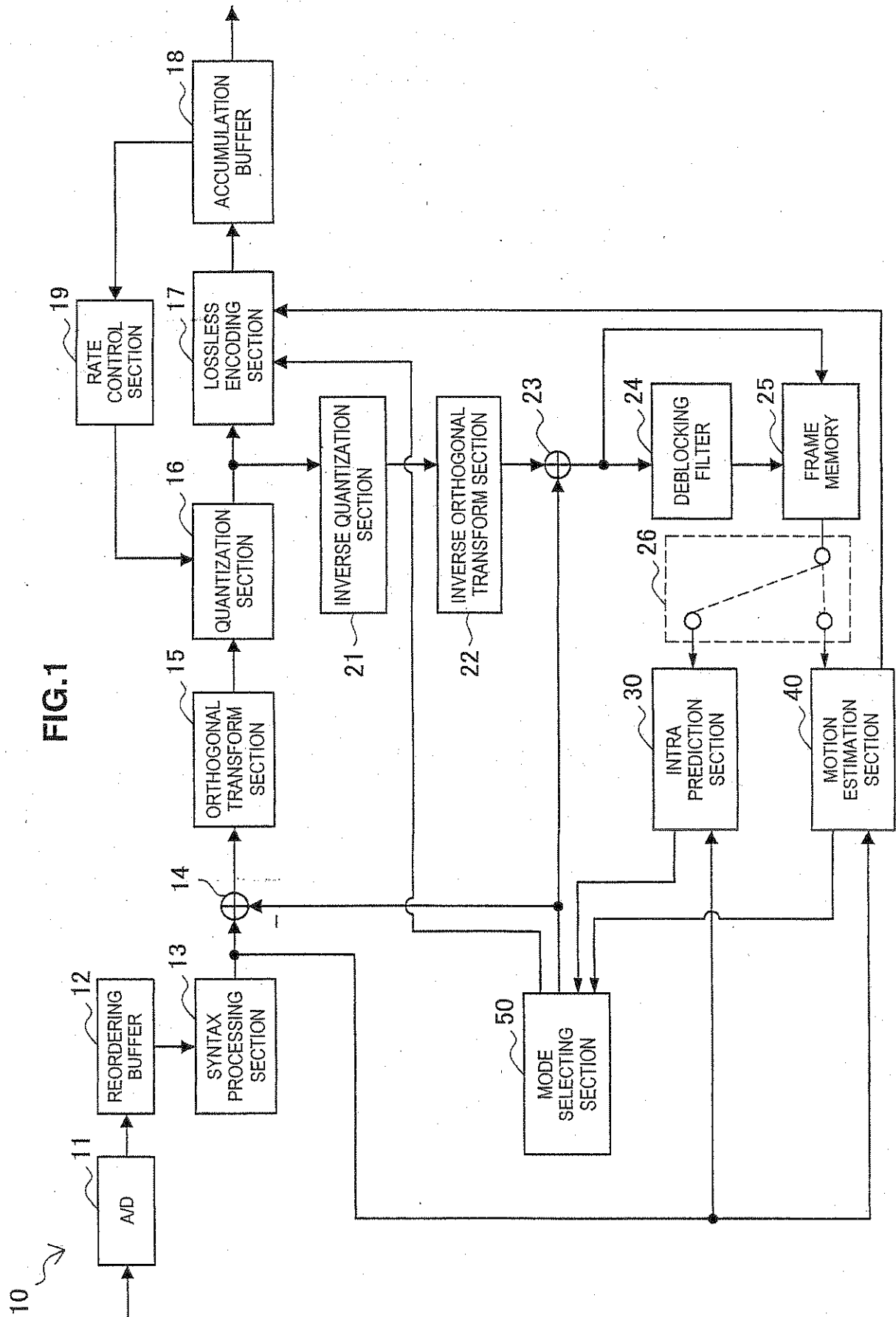


FIG.2

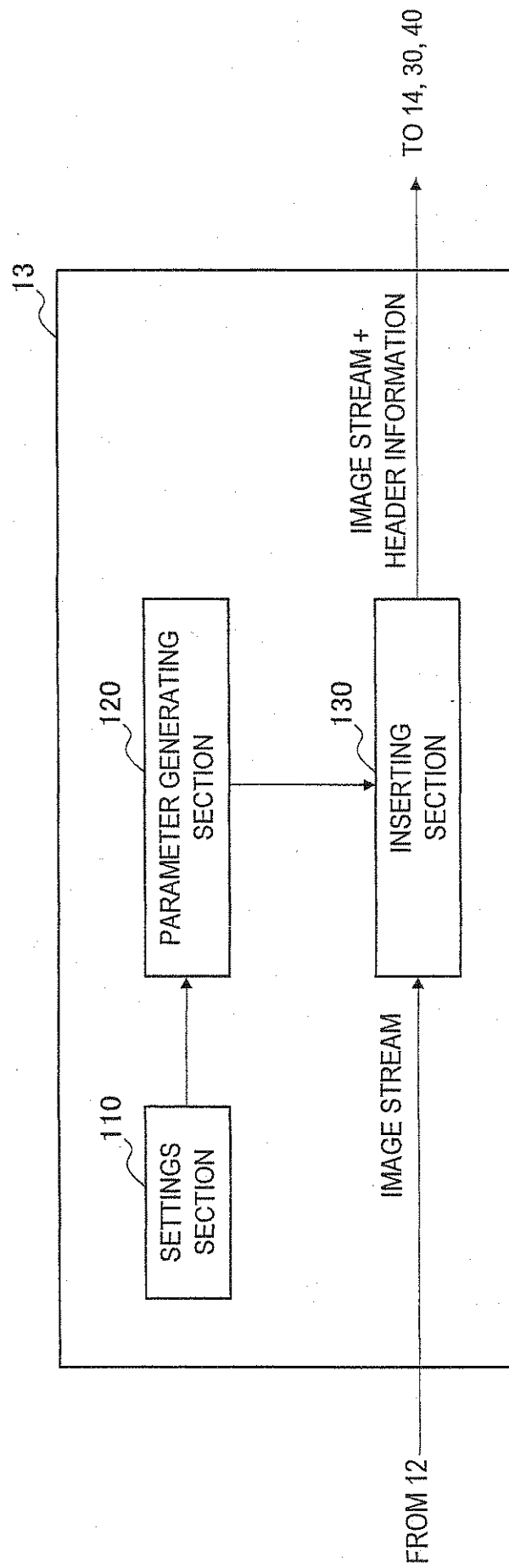


FIG.3

PARAMETER NAME	DESCRIPTION
QMPS ID	IDENTIFIER FOR DISTINGUISHING EACH QMPS
GENERATION MODE PRESENT FLAG	FLAG INDICATING WHETHER CLASSIFICATION OF GENERATION MODE (pred_mode) IS PRESENT
GENERATION MODE <pred_mode>	CLASSIFICATION EXPRESSING MODE OF QUANTIZATION MATRIX GENERATION PROCESS E.G.) 0: COPY, 1: AXIS DESIGNATION, 2: FULL SCAN
CASE OF COPY MODE (pred_mode='0')	
SOURCE ID	QMPS ID DESIGNATING COPY SOURCE QMPS * DESIGNATES OWN ID WHEN USING DEFAULT MATRIX
COPY SOURCE SIZE	SIZE OF COPY SOURCE QUANTIZATION MATRIX
COPY SOURCE TYPE	TYPE OF COPY SOURCE QUANTIZATION MATRIX
RESIDUAL FLAG	FLAG INDICATING PRESENCE OF RESIDUAL ERROR
RESIDUAL DATA	DATA FOR GENERATING RESIDUAL MATRIX
CASE OF AXIS DESIGNATION MODE (pred_mode='1')	
DESIGNATION METHOD FLAG	FLAG INDICATING REFERENCE AXIS DESIGNATION METHOD E.G.) 0: DIFFERENTIAL, 1: INTERPOLATION
REFERENCE AXIS DATA	DATA FOR REFERENCE AXIS ELEMENTS IN THE CASE OF DIFFERENTIAL METHOD
CORNER DATA	DATA FOR CORNER ELEMENTS IN THE CASE OF INTERPOLATION METHOD
RESIDUAL FLAG	FLAG INDICATING PRESENCE OF RESIDUAL ERROR
DIFFERENTIAL DATA	DATA FOR GENERATING RESIDUAL MATRIX
CASE OF FULL SCAN MODE (pred_mode='2')	
DIFFERENTIAL DATA	LINEAR ARRAY OF DIFFERENTIAL VALUES COMPUTED IN DPCM FORMAT FOR EACH ELEMENT IN QUANTIZATION MATRIX

QUANTIZATION MATRIX PARAMETER SET

FIG.4

PARAMETER NAME	DESCRIPTION
:	
SLICE TYPE	SLICE ENCODING TYPE
PPS ID	ID OF PPS REFERENCED FOR CURRENT SLICE
:	
QMPS ID PRESENT FLAG	FLAG INDICATING WHETHER QMPS ID IS PRESENT WITHIN CURRENT HEADER
QMPS ID	ID OF QMPS REFERENCED FOR CURRENT SLICE
:	

SLICE HEADER

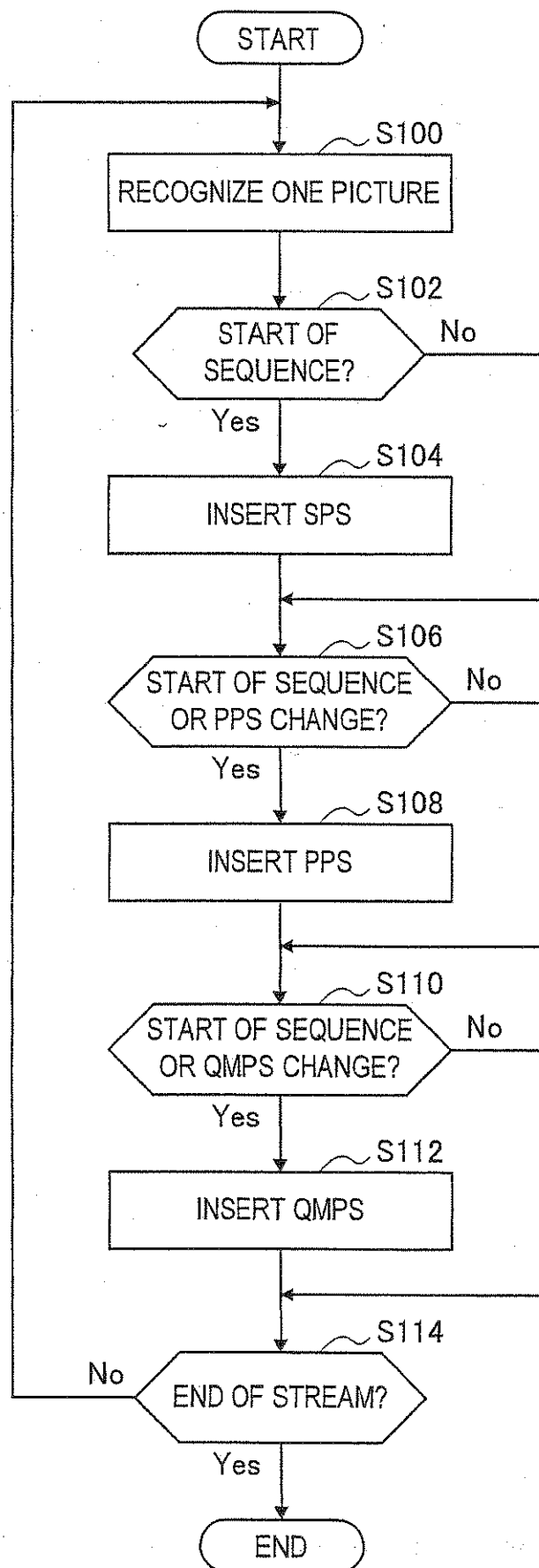
FIG.5

FIG.6

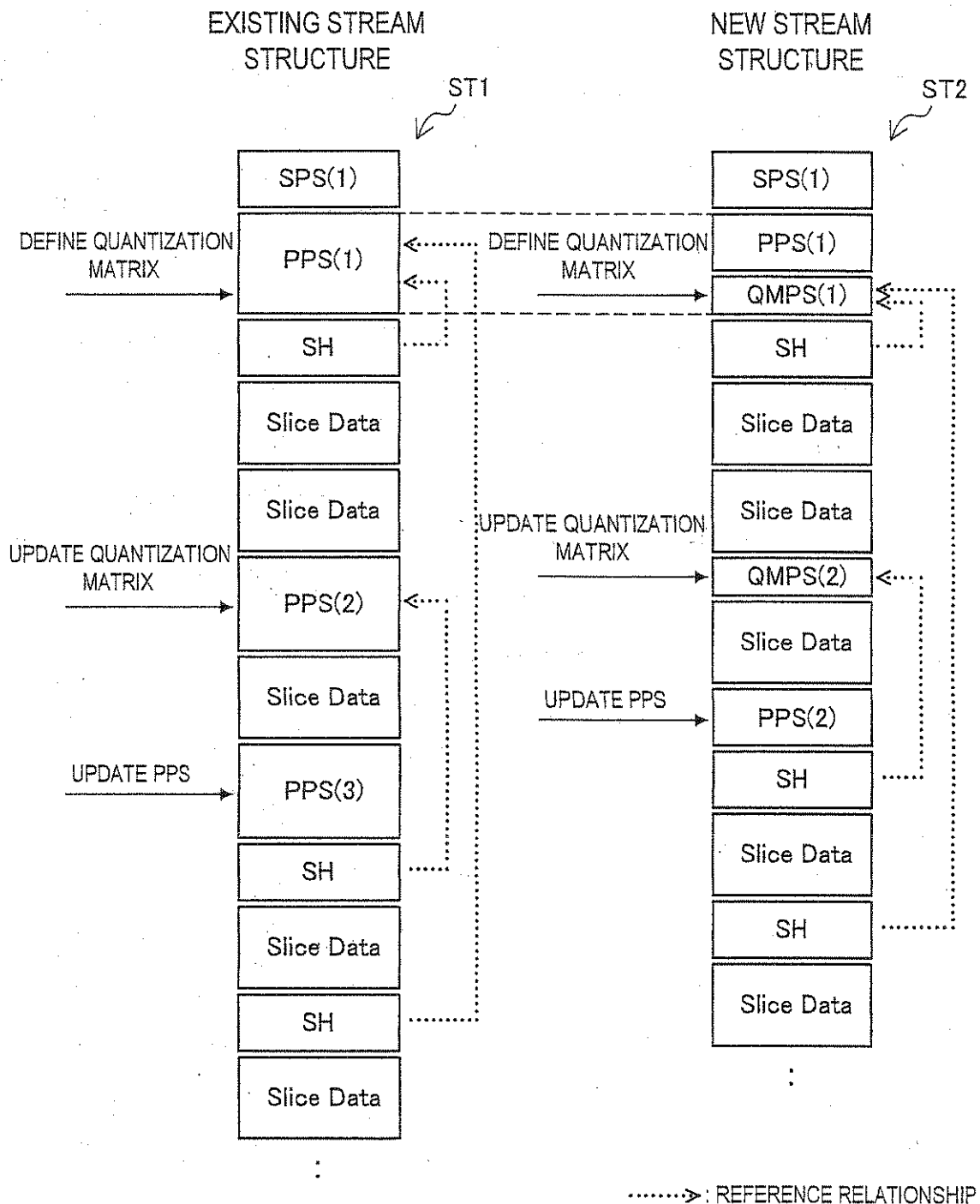


FIG. 7

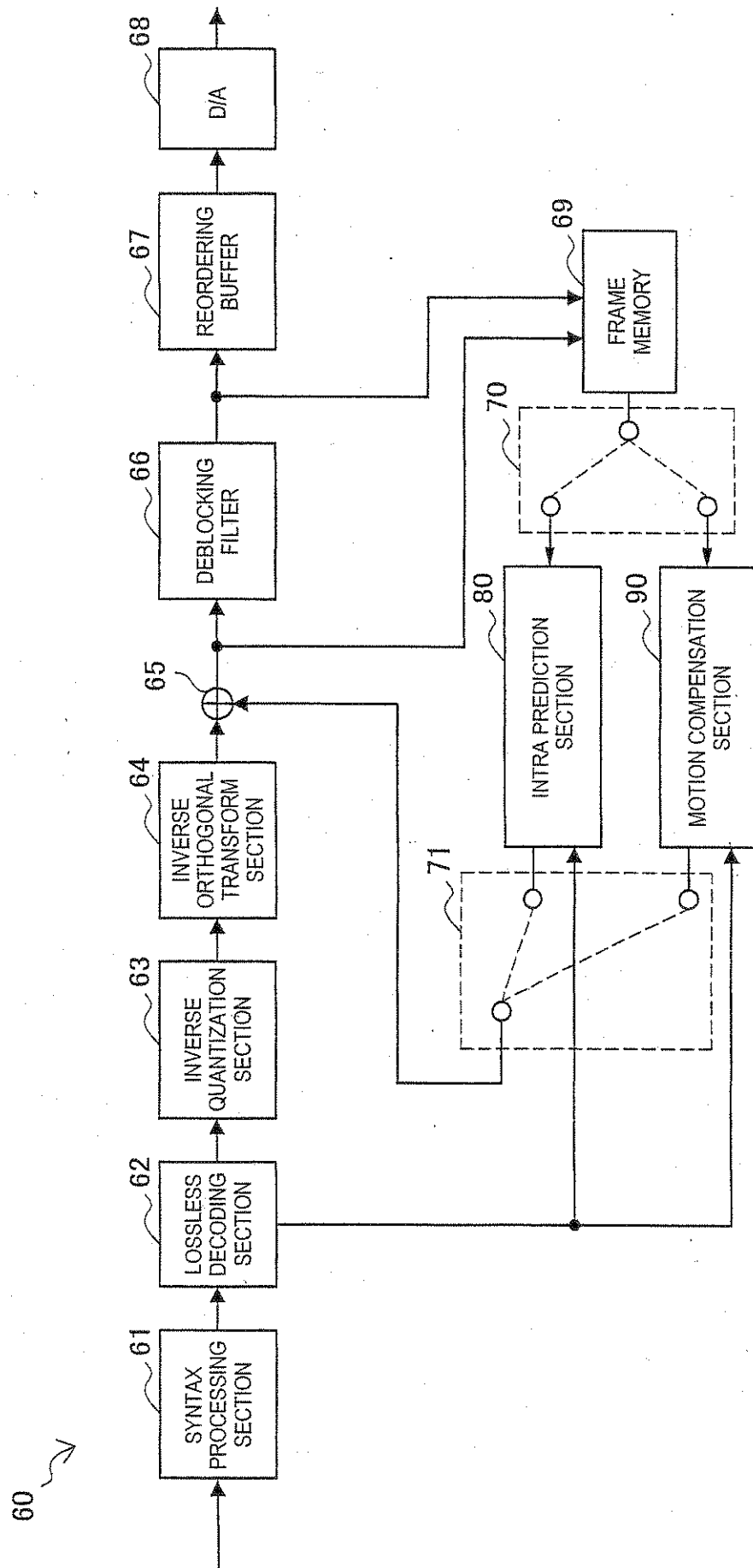


FIG. 8

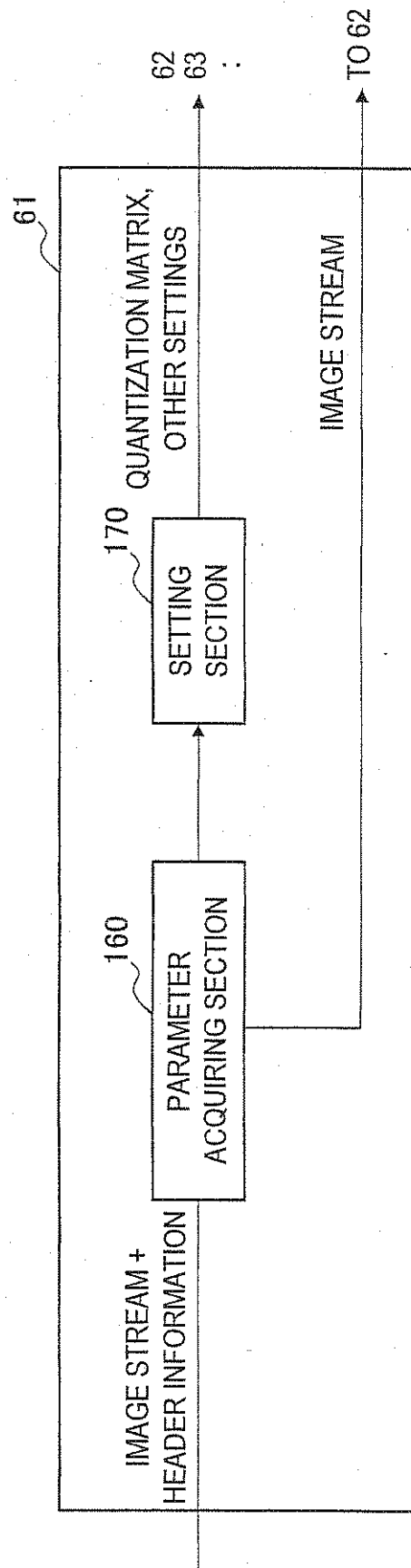


FIG.9

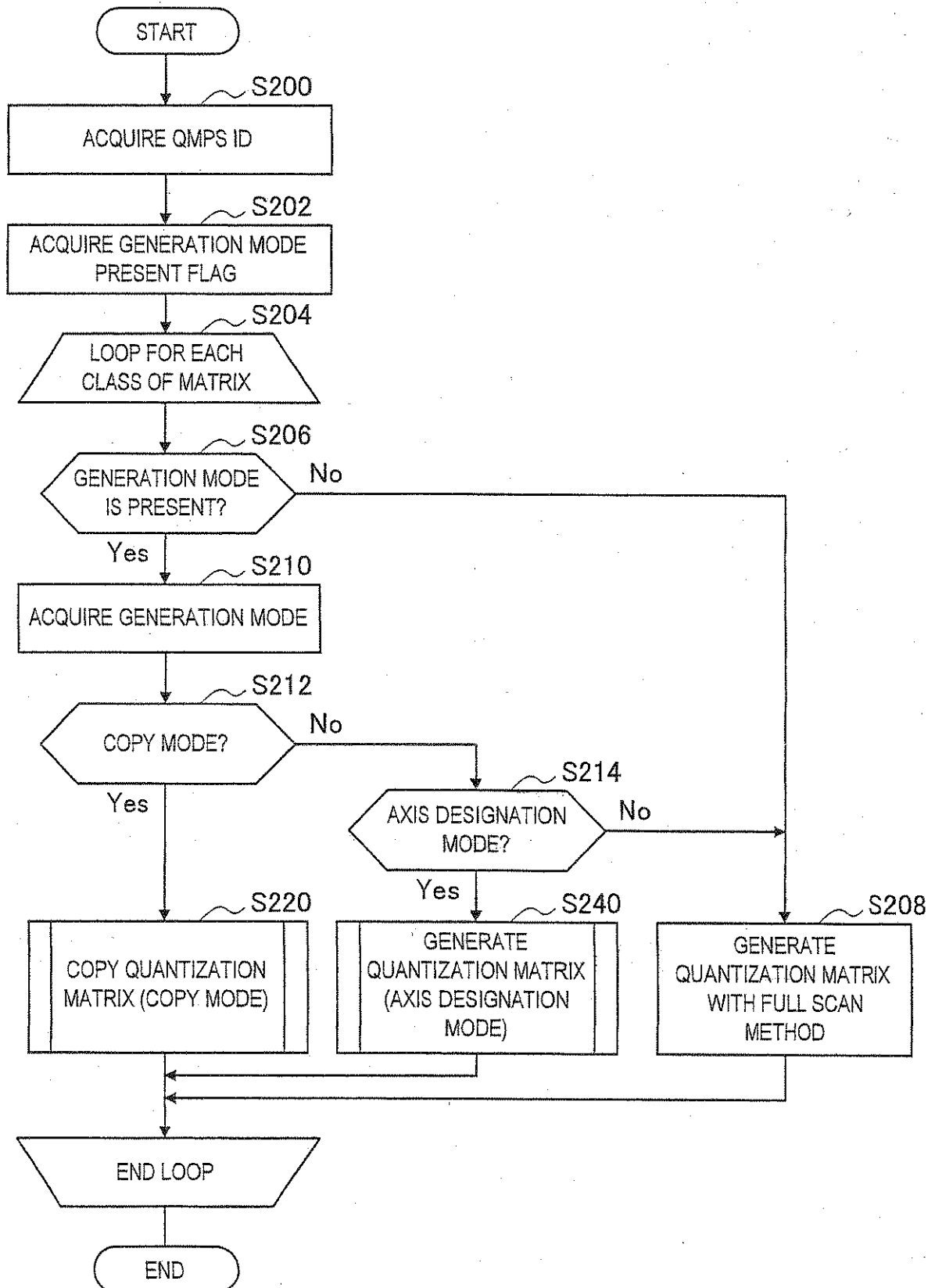


FIG.10

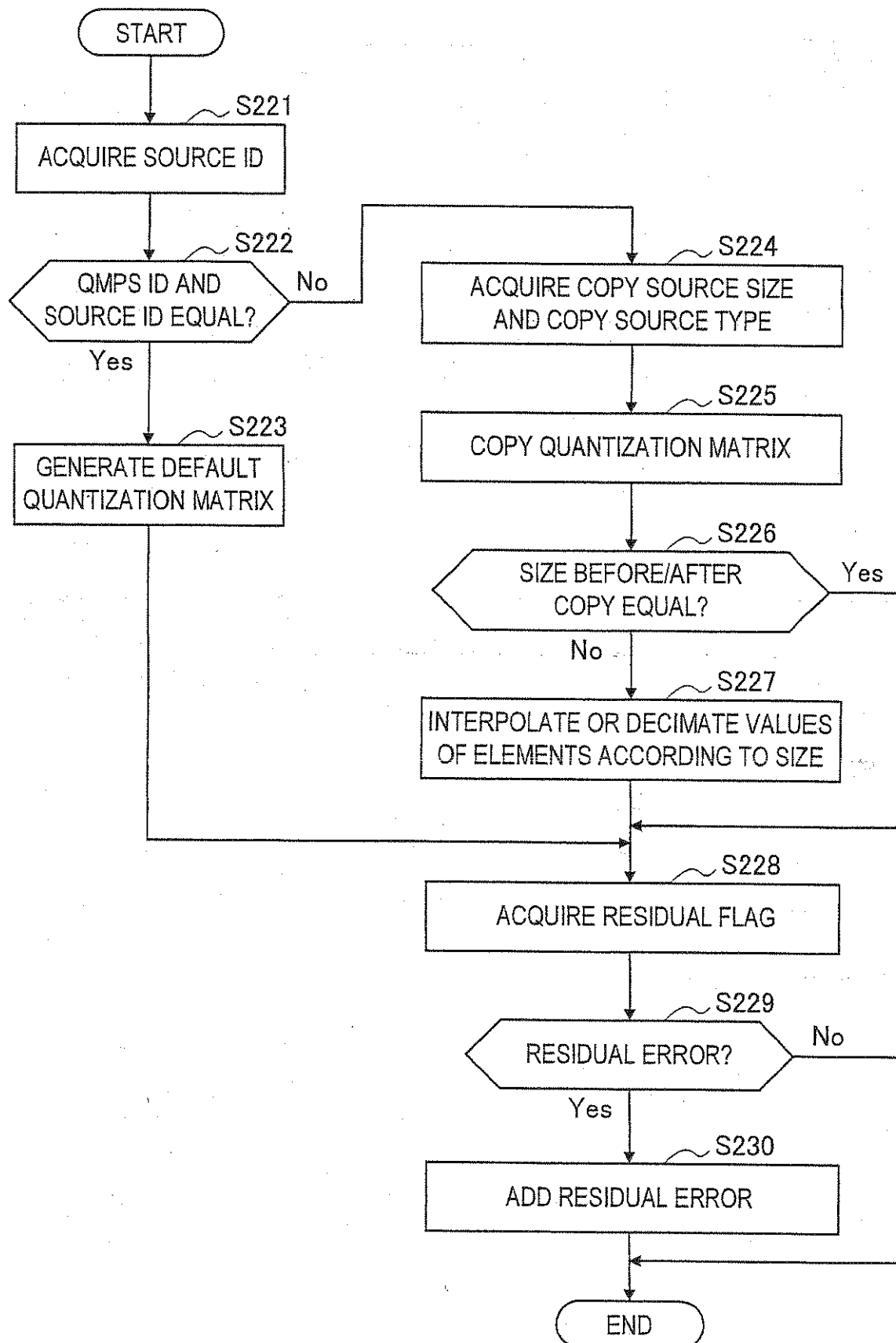


FIG.11

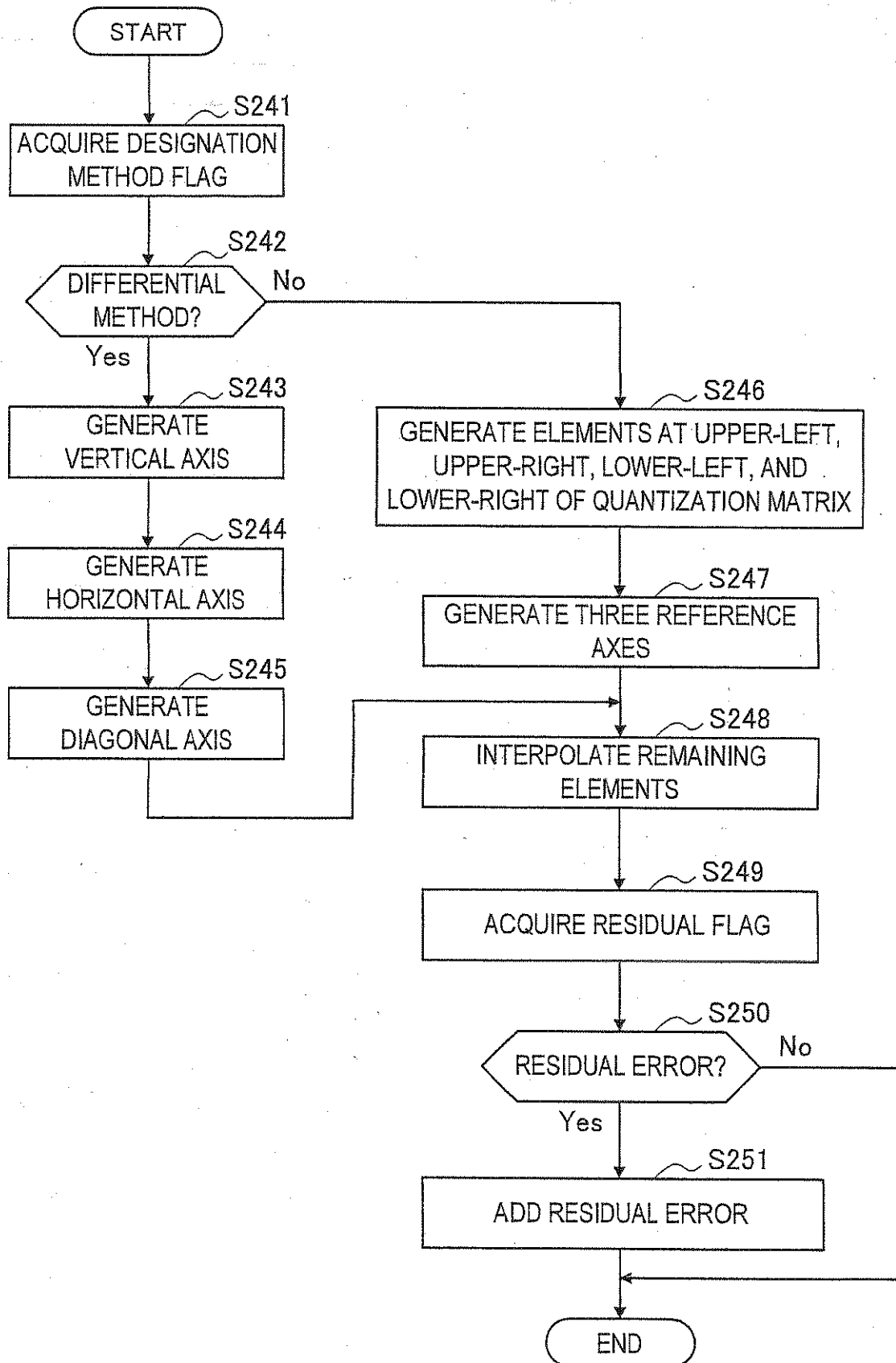


FIG.12

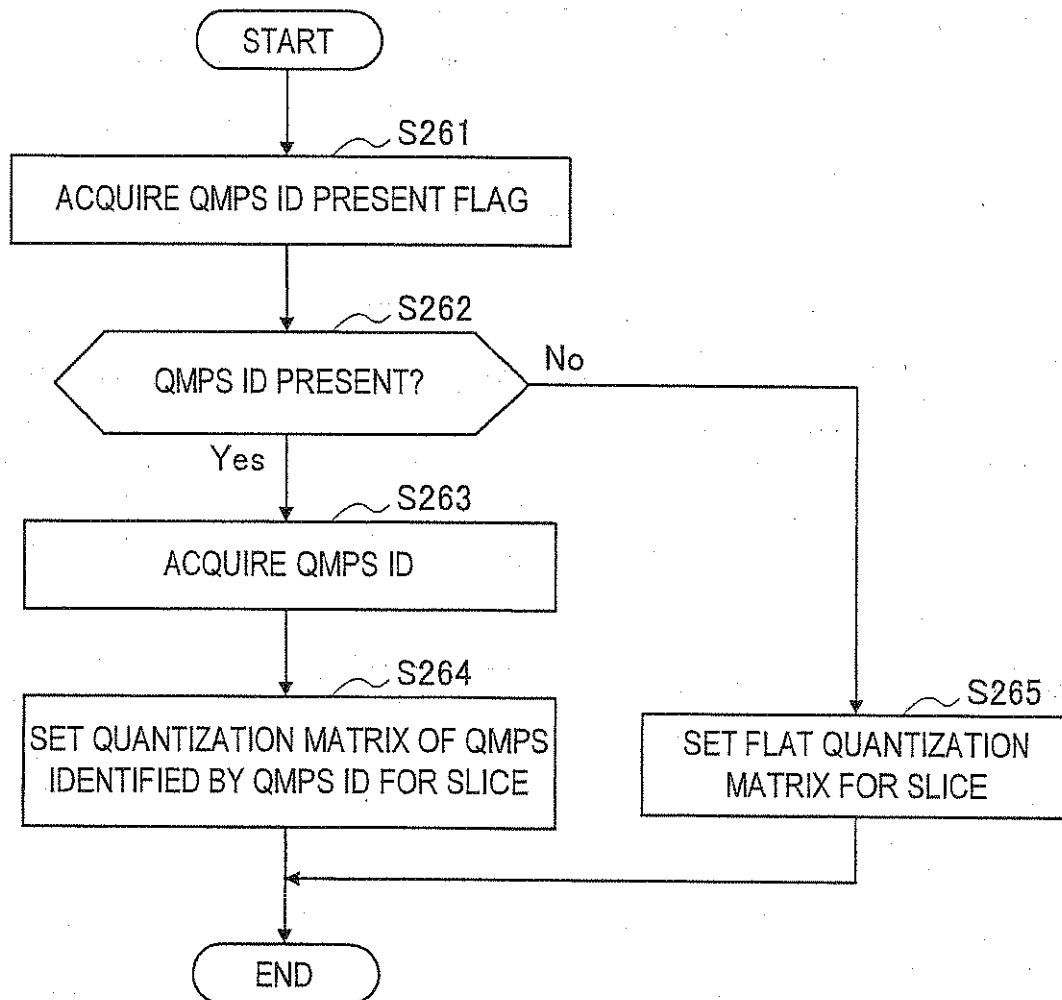


FIG.13

```

01 | QuantizaionMatrixParameterSet(){
02 |   quantization_matrix_paramter_id      ...QMPS ID
03 |   pred_present_flag                    ...GENERATION MODE PRESENT FLAG
04 |   for(i=0;i<SizeIdc;i++){                  //LOOP FOR EACH SIZE//
05 |     for(j=0;j<6;j++){                      //LOOP FOR EACH TYPE//
06 |       if(pred_present_flag){
07 |         pred_mode                      ...GENERATION MODE
08 |         if(pred_mode==0){                  //COPY MODE//
09 |           pred_qmps_id                  ...SOURCE ID
10 |           pred_size_idc                ...COPY SOURCE SIZE
11 |           pred_matrix_id              ...COPY SOURCE TYPE
12 |           pred_matrix(j,i,pred_qmps_id,pred_size_idc,pred_matrix_id)
13 |                                           //COPY//
14 |           residual_flag                ...RESIDUAL FLAG
15 |           if(residual_flag){
16 |             residual_matrix(i)             //ADD RESIDUAL ERROR//
17 |           }
18 |           }else if(pred_mode==1){          //AXIS DESIGNATION MODE//
19 |             //SEE FIG. 14//
20 |           }else if(pred_mode==2){          //FULL SCAN MODE//
21 |             qmatrix(i,j)
22 |           }
23 |           }else{
24 |             qmatrix(i,j)
25 |           }
26 |         }
27 |       }
28 |     }
29 |   }

```

FIG.14

```

18 | dpcm_flag                ...DESIGNATION METHOD FLAG
19 | if(dpcm_flag){           //DIFFERENTIAL METHOD//
20 |     nextcoef = 0
21 |     for(i=0,i<Size;i++){ //VERTICAL AXIS//
22 |         delta_coef
23 |         nextcoef = nextcoef + delta_coef
24 |         coef_vertical[i] = nextcoef
25 |     }
26 |     nextcoef = coef_horizontal[0] = coef_vertical[0]
27 |     copy_from_vertical
28 |     if(copy_from_vertical != 0){
29 |         for(i=1,i<Size;i++){ //HORIZONTAL AXIS//
30 |             delta_coef
31 |             nextcoef = nextcoef + delta_coef
32 |             coef_horizontal[i] = nextcoef
33 |         }
34 |     }
35 |     nextcoef = coef_diagonal[0] = coef_vertical[0]
36 |     for(i=1,i<Size;i++){ //DIAGONAL AXIS//
37 |         delta_coef
38 |         nextcoef = nextcoef + delta_coef
39 |         coef_diagonal[i] = nextcoef
40 |     }
41 | }else{                    //INTERPOLATION METHOD//
42 |     dc                ...UPPER-LEFT (DC COMPONENT)
43 |     horizontal_end    ...LOWER-LEFT
44 |     vertical_end      ...LOWER-RIGHT
45 |     diagonal_end      ...LOWER-RIGHT
46 | }
47 | residual_flag        ...RESIDUAL FLAG
48 | if(residual_flag){
49 |     residual_matrix(i)   //ADD RESIDUAL ERROR//
50 | }

```

FIG.15

```
01 | residual_matrix(){
02 |   residual_dpcm_flag           ...RESIDUAL DESIGNATION METHOD FLAG
03 |   if(residual_dpcm_flag){       //DPCM METHOD//
04 |     nextcoef = 0
05 |     for(i=0,i<coefNum;i++){
06 |       delta_coef
07 |       nextcoef = nextcoef + delta_coef
08 |       coef[i] = nextcoef
09 |     }
10 |   }else{                         //RUN-LENGTH METHOD//
11 |     next_pos = 0
12 |     while(last_pos==0){
13 |       run
14 |       data
15 |       last_pos
16 |       coef[next_pos+run] = data
17 |       next_pos += run+1
18 |     }
19 |   }
20 | }
```

FIG.16

```

01 | QuantizaionMatrixParameterSet([
02 |   quantization_matrix_paramter_id    ...QMPS ID
03 |   use default only flag
04 |   use dqp flag
05 |   if(use_default_only_flag == 0){
06 |       pred present flag                ...GENERATION MODE PRESENT FLAG
07 |       Qscale0                          ...1ST QUANTIZATION SCALE
08 |       Qscale1                          ...2ND QUANTIZATION SCALE
09 |       Qscale2                          ...3RD QUANTIZATION SCALE
10 |       Qscale3                          ...4TH QUANTIZATION SCALE
11 |       for(i=0;i<4;i++){                  //LOOP FOR EACH SIZE//
12 |           for(j=0;j<6;j++){              //LOOP FOR EACH TYPE//
13 |               if(pred_present_flag){
14 |                   pred_mode                ...GENERATION MODE
15 |                   if(pred_mode==0){        //COPY MODE//
16 |                       pred_qmps id          ...SOURCE ID
17 |                       pred_size_idc        ...COPY SOURCE SIZE
18 |                       pred_matrix_id       ...COPY SOURCE TYPE
19 |                       residual_flag        ...RESIDUAL FLAG
20 |                       if(residual_flag){
21 |                           residual_matrix()    //SPECIFY RESIDUAL ERROR//
22 |                       }
23 |
24 |                       //SEE FIGS. 17, 18//
25 |
26 |                   }
27 |               }
28 |           }
29 |       }
30 |   }

```

FIG.17

```

23 |     }else if(pred_mode==1){                                     //AXIS DESIGNATION MODE//
24 |         dpcm_flag                                             ...DESIGNATION METHOD FLAG
25 |         if(dpcm_flag){                                         //DIFFERENTIAL METHOD//
26 |             nextcoef = use_dqp_flag ? -6 : 8
27 |             for(i=0,i<Size;i++){                               //VERTICAL AXIS//
28 |                 delta_coef
29 |                 nextcoef = nextcoef + delta_coef
30 |                 coef_vertical[i] = nextcoef
31 |             }
32 |             nextcoef = coef_horizontal[0] = coef_vertical[0]
33 |             copy_from_vertical
34 |             if(copy_from_vertical != 0){
35 |                 for(i=1,i<Size;i++){                           //HORIZONTAL AXIS//
36 |                     delta_coef
37 |                     nextcoef = nextcoef + delta_coef
38 |                     coef_horizontal[i] = nextcoef
39 |                 }
40 |             }
41 |             nextcoef = coef_diagonal[0] = coef_vertical[0]
42 |             for(i=1,i<Size;i++){                               //DIAGONAL AXIS//
43 |                 delta_coef
44 |                 nextcoef = nextcoef + delta_coef
45 |                 coef_diagonal[i] = nextcoef
46 |             }
47 |         }else{                                                 //INTERPOLATION METHOD//
48 |             dc                                                 ...UPPER-LEFT (DC COMPONENT)
49 |             horizontal_end                                     ...UPPER-RIGHT
50 |             vertical_end                                       ...LOWER-LEFT
51 |             diagonal_end                                       ...LOWER-RIGHT
52 |         }
53 |         residual_flag                                         ...RESIDUAL FLAG
54 |         if(residual_flag){
55 |             residual_matrix()                                   //SPECIFY RESIDUAL ERROR//
56 |         }

```

FIG.18

```
57 |         }else if(pred_mode==2){                                //FULL SCAN MODE//
58 |             matrix_symmetry_flag
59 |             matrix_sign_flag
60 |             matrix_dpcm_run_flag        ...DESIGNATION METHOD FLAG
61 |             vlc_table_data            ...VLC TABLE (DATA)
62 |             if(matrix_dpcm_run_flag){
63 |                 vlc_table_run        ...VLC TABLE (RUN)
64 |             }
65 |             qmatrix()
66 |         }
67 |     }else{
68 |         matrix_symmetry_flag
69 |         matrix_sign_flag
70 |         matrix_dpcm_run_flag        ...DESIGNATION METHOD FLAG
71 |         vlc_table_data            ...VLC TABLE (DATA)
72 |         if(matrix_dpcm_run_flag){
73 |             vlc_table_run        ...VLC TABLE (RUN)
74 |         }
75 |         qmatrix()
76 |     }
```

FIG.19

```

01 | qmatrix(){
02 |   if(!matrix_dpcm_run_flag){
03 |     nextcoef = use_dqp_flag ? -6 : 8
04 |     for(i=0; i<coefNum; i++){
05 |       delta_coef
06 |       nextcoef = nextcoef + delta_coef
07 |       coef[i] = nextcoef
08 |     }
09 |   }else{
10 |     next_pos = 0
11 |     while(last_pos==0){
12 |       run
13 |       data
14 |       coef[next_pos+run] = data
15 |       next_pos += run+1
16 |       if(uiSize <= next_pos){
17 |         last_pos = 1;
18 |       }else{
19 |         last_pos = 0;
20 |       }
21 |     }
22 |   }
23 | }

```

//DPCM METHOD//

←----- ENCODE WITH VLC TABLE (DATA)

//RUN-LENGTH METHOD//

←----- ENCODE WITH VLC TABLE (RUN)

←----- ENCODE WITH VLC TABLE (DATA)

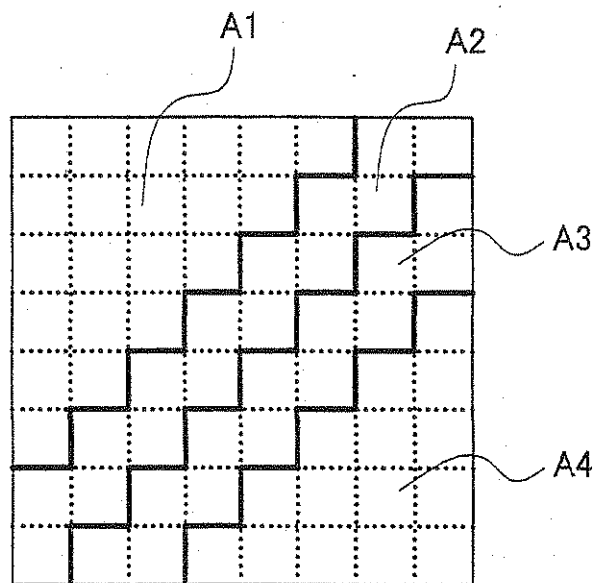
FIG.20

```

01 | residual_matrix(){
02 |   residual_symmetry_flag
03 |   residual_dpcm_flag
04 |   residual_run_flag           ...RESIDUAL ERROR DESIGNATION METHOD FLAG
05 |   residual_sign_flag
06 |   if(!residual_run_flag){           //DPCM METHOD//
07 |     vlc_table_data           ...VLC TABLE (DATA)
08 |     nextcoef = 0
09 |     for(i=0; i<coefNum; i++){
10 |       delta_coef           ←----- ENCODE WITH VLC TABLE (DATA)
11 |       nextcoef = nextcoef + delta_coef
12 |       coef[i] = nextcoef
13 |     }
14 |   }else{                           //RUN-LENGTH METHOD//
15 |     vlc_table_data           ...VLC TABLE (DATA)
16 |     vlc_table_run           ...VLC TABLE (RUN)
17 |     next_pos = 0
18 |     while(last_pos==0){
19 |       run                   ←----- ENCODE WITH VLC TABLE (RUN)
20 |       data                   ←----- ENCODE WITH VLC TABLE (DATA)
21 |       coef[next_pos+run] = data
22 |       next_pos += run+1
23 |       if(uiSize <= next_pos){
24 |         last_pos = 1;
25 |       }else{
26 |         last_pos = 0;
27 |       }
28 |     }
29 |   }
30 | }

```

FIG.21



EXAMPLE OF QUANTIZATION SCALE SETTING AREAS
IN 8×8 QUANTIZATION MATRIX

FIG.22

AREA A1: Qscale0=1 (NOT QUANTIZED)

AREA A2: Qscale1=2

AREA A3: Qscale2=3

AREA A4: Qscale3=4

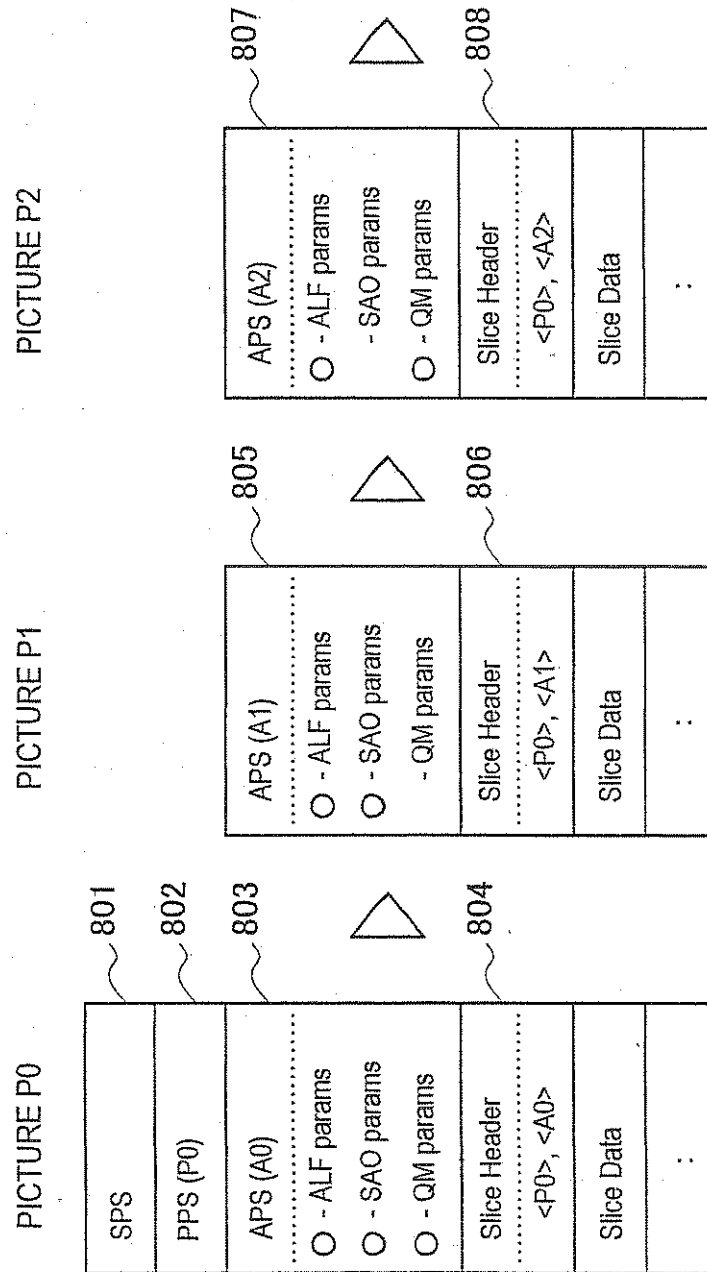
1	1	1	1	1	1	2	2
1	1	1	1	1	2	2	3
1	1	1	1	2	2	3	3
1	1	1	2	2	3	3	4
1	1	2	2	3	3	4	4
1	2	2	3	3	4	4	4
2	2	3	3	4	4	4	4
2	3	3	4	4	4	4	4

EXAMPLE QUANTIZATION
SCALE SETTING

FIG.23

VLC0	VLC1	VLC2	VLC3	VLC4
1 01 001 0001 00001 0000001 00000001 000000001x 0000000001xx ;	1x 01x 001x 0001x 00001x 0000001x 00000001x 000000001xx 0000000001xxx ;	1xx 01xx 001xx 0001xx 00001xx 0000001xx 00000001xx 000000001xxx 0000000001xxxx ;	1xxx 01xxx 001xxx 0001xxx 00001xxx 0000001xxx 00000001xxx 000000001xxxx 0000000001xxxxx ;	1xxxx 01xxxx 001xxxx 0001xxxx 00001xxxx 0000001xxxx 00000001xxxx 000000001xxxxx 0000000001xxxxxx ;
VLC5	VLC6	VLC7	VLC8	VLC9
1x 01x 001x ;	1xx 01xx 001xx ;	1xxx 01xxx 001xxx ;	1 01 00 ;	100 1010 1011 11xxx 01xxxx 001xxxx ;
VLC10				
1 01x 001xx 0001xxx ;				

FIG.24



(): PARAMETER SET IDENTIFIER <>: REFERENCE PARAMETER

○: CHANGE IN PARAMETER

FIG.25

```

01 | aps_rbsp() {
02 |     aps_id ←----- APS ID
03 |     aps_sample_adaptive_offset_flag
04 |     aps_adaptive_loop_filter_flag
05 |     aps_qmatrix_flag
06 |     if( aps_sample_adaptive_offset_flag || aps_adaptive_loop_filter_flag || aps_qmatrix_flag ) {
07 |         aps_cabac_use_flag
08 |         if( aps_cabac_use_flag ) {
09 |             aps_cabac_init_idc
10 |             aps_cabac_init_qp_minus26
11 |         }
12 |     }
13 |     if( aps_adaptive_loop_filter_flag ) {
14 |         alf_data_byte_count
15 |         alf_param()
16 |         byte_align()
17 |     }
18 |     if( aps_sample_adaptive_offset_flag ) {
19 |         sao_data_byte_count
20 |         byte_align()
21 |         sao_param()
22 |         byte_align()
23 |     }
24 |     if(aps_qmatrix_flag) {
25 |         qmatrix_data_byte_count
26 |         byte_align()
27 |         qmatrix_param()
28 |     }
29 |     rbsp_trailing_bits()
30 | }

```

ALF-RELATED PARAMETERS

SAO-RELATED PARAMETERS

QM-RELATED PARAMETERS

FIG.26

```

01 | slice_header() {
02 |     lightweight_slice_flag
03 |     if( !lightweight_slice_flag ) {
04 |         slice_type
05 |         pic_parameter_set_id          ←----- REFERENCE PPS ID
06 |     }
07 |     if( sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag || qmatrix_enabled_flag )
08 |         aps_id                        ←----- REFERENCE APS ID
09 |         frame_num
10 |         if( IdrPicFlag ) idr_pic_id
11 |         if( pic_order_cnt_type == 0 ) pic_order_cnt_lsb
12 |         if( slice_type == P || slice_type == B ) {
13 |             num_ref_idx_active_override_flag
14 |             if( num_ref_idx_active_override_flag ) {
15 |                 num_ref_idx_l0_active_minus1
16 |                 if( slice_type == B ) num_ref_idx_l1_active_minus1
17 |             }
18 |         }
19 |         ref_pic_list_modification()
20 |         ref_pic_list_combination()
21 |         if( nal_ref_idc != 0 ) dec_ref_pic_marking()
22 |     }
23 |     if( entropy_coding_mode_flag && slice_type != I ) cabac_init_idc
24 |     first_slice_in_pic_flag
25 |     if( first_slice_in_pic_flag == 0 ) slice_address
26 |     if( !lightweight_slice_flag ) {
27 |         slice_qp_delta
28 |         if( deblocking_filter_control_present_flag ) {
29 |             disable_deblocking_filter_idc
30 |             if( disable_deblocking_filter_idc != 1 ) {
31 |                 slice_alpha_c0_offset_div2
32 |                 slice_beta_offset_div2
33 |             }
34 |         }
35 |         if( slice_type == B ) collocated_from_l0_flag
36 |         if( adaptive_loop_filter_enabled_flag && aps_adaptive_loop_filter_flag ) {
37 |             byte_align()
38 |             alf_cu_control_param()
39 |             byte_align()
40 |         }
41 |     }
42 | }

```

FIG.27

```

01 | aps_rbsp() {
02 |     aps_id                                ←----- APS ID
03 |     aps_sample_adaptive_offset_flag
04 |     aps_adaptive_loop_filter_flag
05 |     aps_qmatrix_flag
06 |     if( aps_sample_adaptive_offset_flag || aps_adaptive_loop_filter_flag || aps_qmatrix_flag ) {
07 |         aps_cabac_use_flag
08 |         if( aps_cabac_use_flag ) {
09 |             aps_cabac_init_idc
10 |             aps_cabac_init_qp_minus26
11 |         }
12 |     }
13 |     if( aps_adaptive_loop_filter_flag ) {
14 |         alf_data_byte_count
15 |         alf_param()
16 |         byte_align()
17 |     }
18 |     if( aps_sample_adaptive_offset_flag ) {
19 |         sao_data_byte_count
20 |         byte_align()
21 |         sao_param()
22 |         byte_align()
23 |     }
24 |     if(aps_qmatrix_flag) {
25 |         ref_aps_id_present_flag           ←----- PAST REFERENCE ID PRESENT FLAG
26 |         if(ref_aps_id_present_flag) {
27 |             ref_aps_id                     ←----- PAST REFERENCE ID
28 |         } else {
29 |             qmatrix_data_byte_count
30 |             byte_align()
31 |             qmatrix_param()
32 |         }
33 |     }
34 |     rbsp_trailing_bits()
35 | }

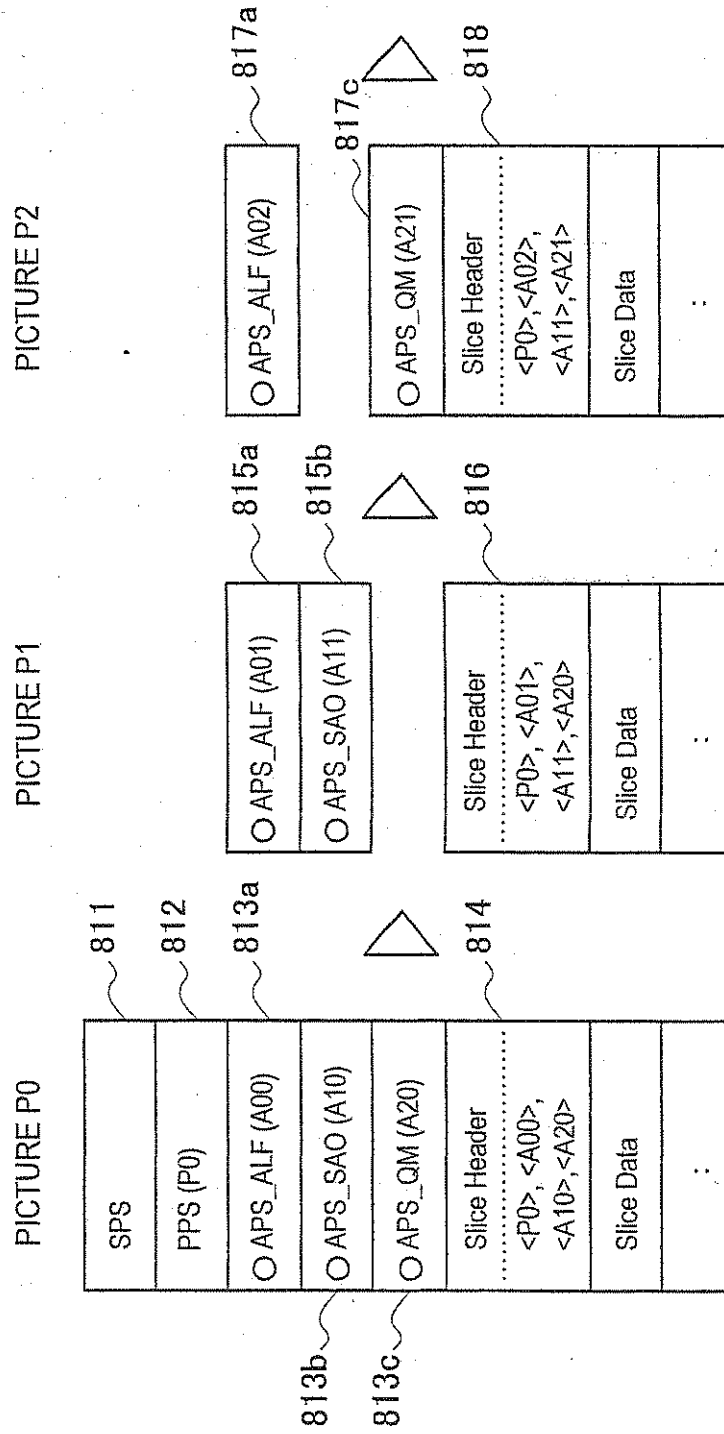
```

ALF-RELATED PARAMETERS

SAO-RELATED PARAMETERS

QM-RELATED PARAMETERS

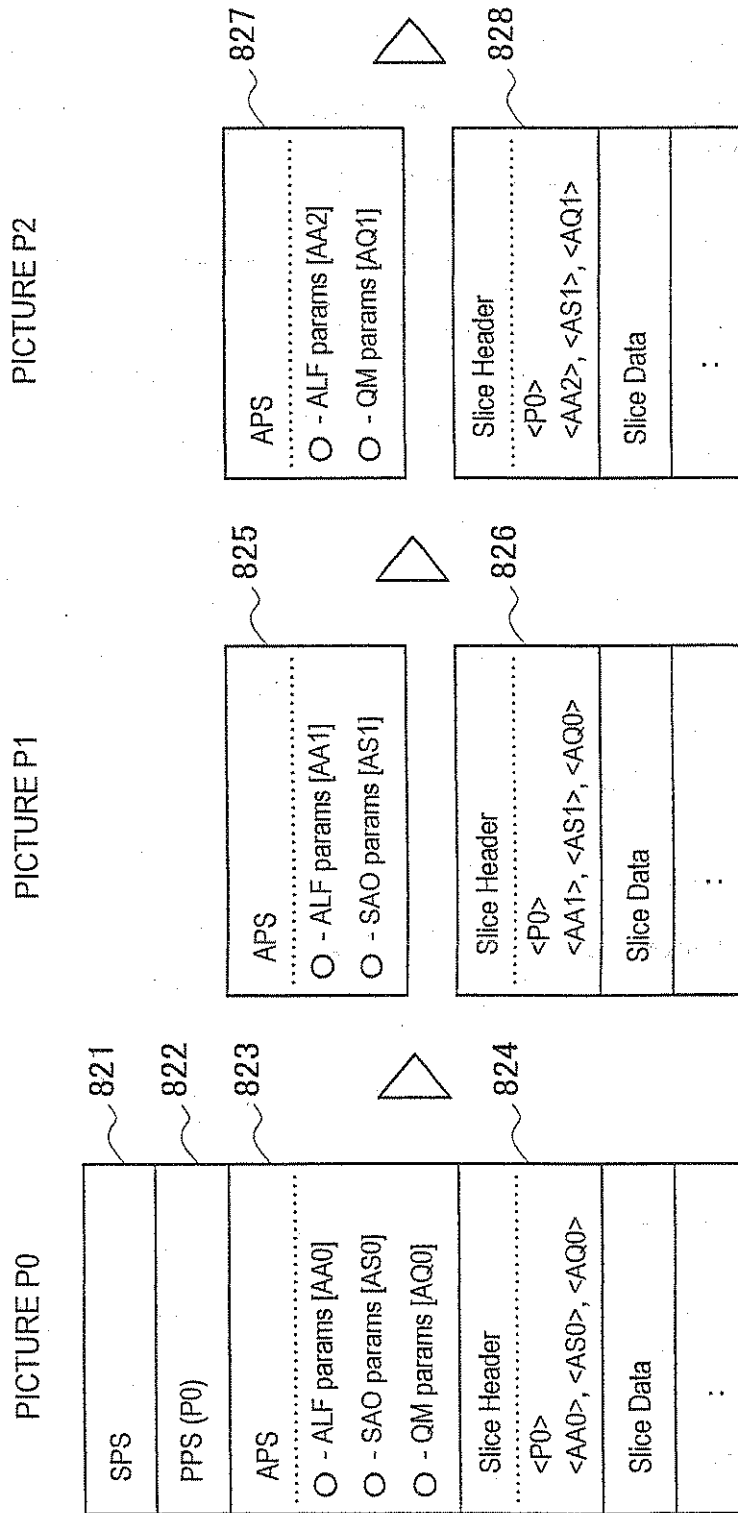
FIG.28



(): PARAMETER SET IDENTIFIER < >: REFERENCE PARAMETER

○ : CHANGE IN PARAMETER

FIG.29



(): PARAMETER SET IDENTIFIER []: SUB-IDENTIFIER < >: REFERENCE PARAMETER
 ○ : CHANGE IN PARAMETER

FIG.30

```

01 | sub_rbsp() {
02 |     aps_adaptive_loop_filter_flag
03 |     aps_sample_adaptive_offset_flag
04 |     aps_qmatrix_flag
05 |     if( aps_sample_adaptive_offset_flag || aps_adaptive_loop_filter_flag || aps_qmatrix_flag ) {
06 |         aps_cabac_use_flag
07 |         if( aps_cabac_use_flag ) {
08 |             aps_cabac_init_idc
09 |             aps_cabac_init_qp_minus26
10 |         }
11 |     }
12 |     if( aps_adaptive_loop_filter_flag ) {
13 |         sub_alf_id
14 |         alf_data_byte_count
15 |         alf_param()
16 |         byte_align()
17 |     }
18 |     if( aps_sample_adaptive_offset_flag ) {
19 |         sub_sao_id
20 |         sao_data_byte_count
21 |         byte_align()
22 |         sao_param()
23 |         byte_align()
24 |     }
25 |     if( aps_qmatrix_flag ) {
26 |         sub_qmatrix_id
27 |         qmatrix_data_byte_count
28 |         byte_align()
29 |         qmatrix_param()
30 |     }
31 |     rbsp_trailing_bits()
32 | }

```

PER-GROUP PRESENT FLAGS

SUB_ALF ID

ALF-RELATED PARAMETERS

SUB_SAO ID

SAO-RELATED PARAMETERS

SUB_QM ID

QM-RELATED PARAMETERS

FIG.31

```

01 | slice_header() {
02 |     lightweight_slice_flag
03 |     if( !lightweight_slice_flag ) {
04 |         slice_type
05 |         pic_parameter_set_id          ←----- REFERENCE PPS ID
06 |     }
07 |     if( sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag || qmatrix_enabled_flag )
08 |         sub_alf_id                    ←--- REFERENCE SUB_ALF ID
09 |         sub_sao_id                    ←----- REFERENCE SUB_SAO ID
10 |         sub_qmatrix_id                ←--- REFERENCE SUB_QM ID
11 |         frame_num
12 |         if( IdrPicFlag ) idr_pic_id
13 |         if( pic_order_cnt_type == 0 ) pic_order_cnt_lsb
14 |         if( slice_type == P || slice_type == B ) {
15 |             num_ref_idx_active_override_flag
16 |             if( num_ref_idx_active_override_flag ) {
17 |                 num_ref_idx_l0_active_minus1
18 |                 if( slice_type == B ) num_ref_idx_l1_active_minus1
19 |             }
20 |         }
21 |         ref_pic_list_modification()
22 |         ref_pic_list_combination()
23 |         if( nal_ref_idc != 0 ) dec_ref_pic_marking()
24 |     }
25 |     if( entropy_coding_mode_flag && slice_type != I ) cabac_init_idc
26 |     first_slice_in_pic_flag
27 |     if( first_slice_in_pic_flag == 0 ) slice_address
28 |     if( !lightweight_slice_flag ) {
29 |         slice_qp_delta
30 |         if( deblocking_filter_control_present_flag ) {
31 |             disable_deblocking_filter_idc
32 |             if( disable_deblocking_filter_idc != 1 ) {
33 |                 slice_alpha_c0_offset_div2
34 |                 slice_beta_offset_div2
35 |             }
36 |         }
37 |         if( slice_type == B ) collocated_from_l0_flag
38 |         if( adaptive_loop_filter_enabled_flag && aps_adaptive_loop_filter_flag ) {
39 |             byte_align()
40 |             alf_cu_control_param()
41 |             byte_align()
42 |         }
43 |     }
44 | }

```

FIG.32

ENCODING TOOL	PARAMETER CONTENTS	UPDATE FREQUENCY	DATA SIZE	NOTES
ADAPTIVE LOOP FILTER (ALF)	- FILTER COEFFICIENTS - ON/OFF FLAG FOR EACH CU	EVERY PICTURE	VERY LARGE	- MAY BE OMITTED FOR B PICTURES WITH LOW RATE, ETC. - LOW LIKELIHOOD OF BEING ABLE TO REUSE PREVIOUSLY SET PARAMETER VALUES
SAMPLE ADAPTIVE OFFSET (SAO)	- OFFSET PATTERN - OFFSET VALUES	EVERY PICTURE	SOMEWHAT LARGE	- POSSIBILITY OF BEING ABLE TO REUSE PREVIOUSLY SET PARAMETER VALUES
QUANTIZATION MATRIX (QM)	- QUANTIZATION MATRIX FOR EACH PICTURE	EVERY PICTURE / EVERY PICTURE TYPE / EVERY GOP	LARGE	- REUSE POSSIBLE FOR EACH PICTURE TYPE (I/P/B)
ADAPTIVE INTERPOLATION FILTER (AIF)	- FILTER COEFFICIENTS FOR EACH SUB-PIXEL POSITION	EVERY PICTURE	SMALL	- REUSE POSSIBLE FOR EACH PICTURE TYPE (I/P/B)

FIG.33

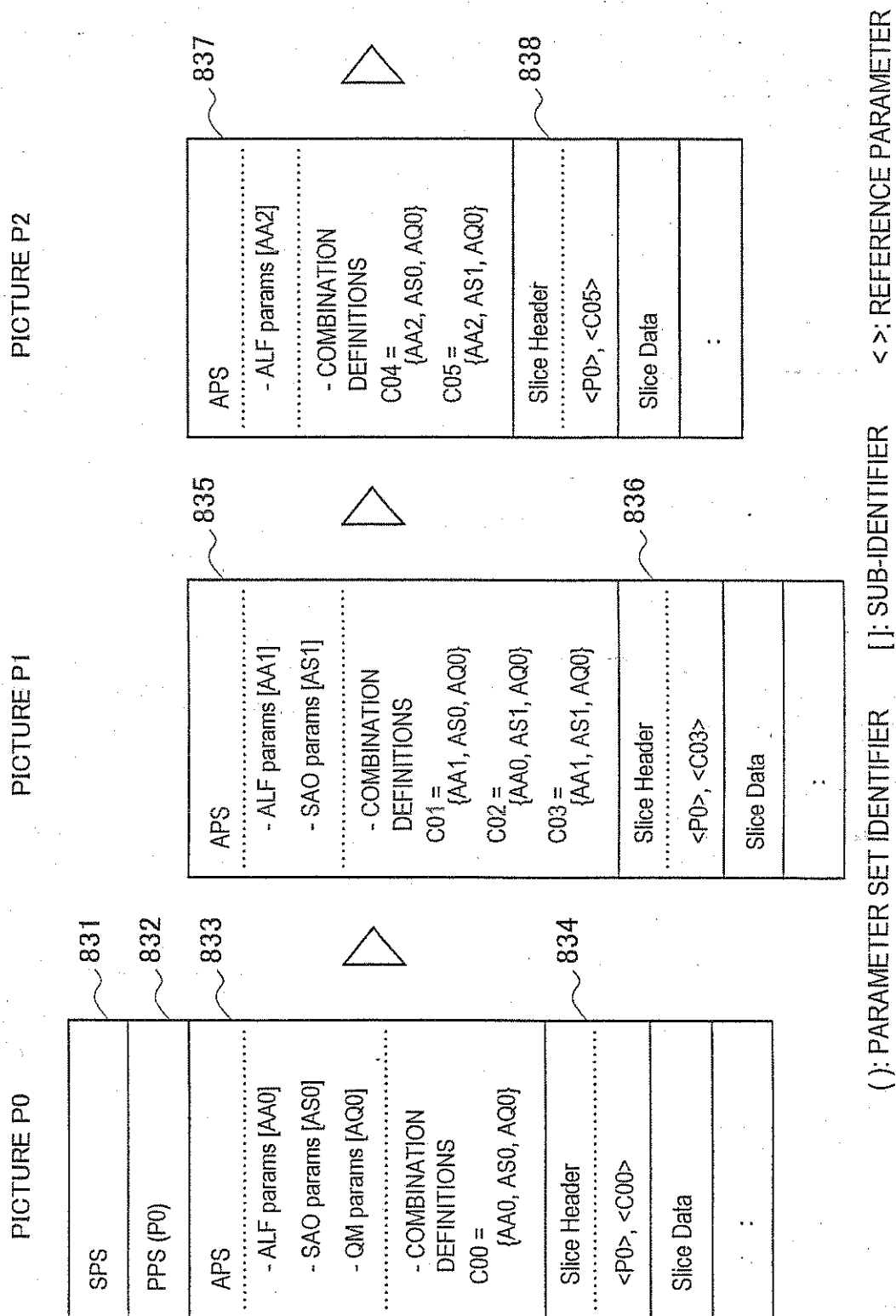


FIG. 34

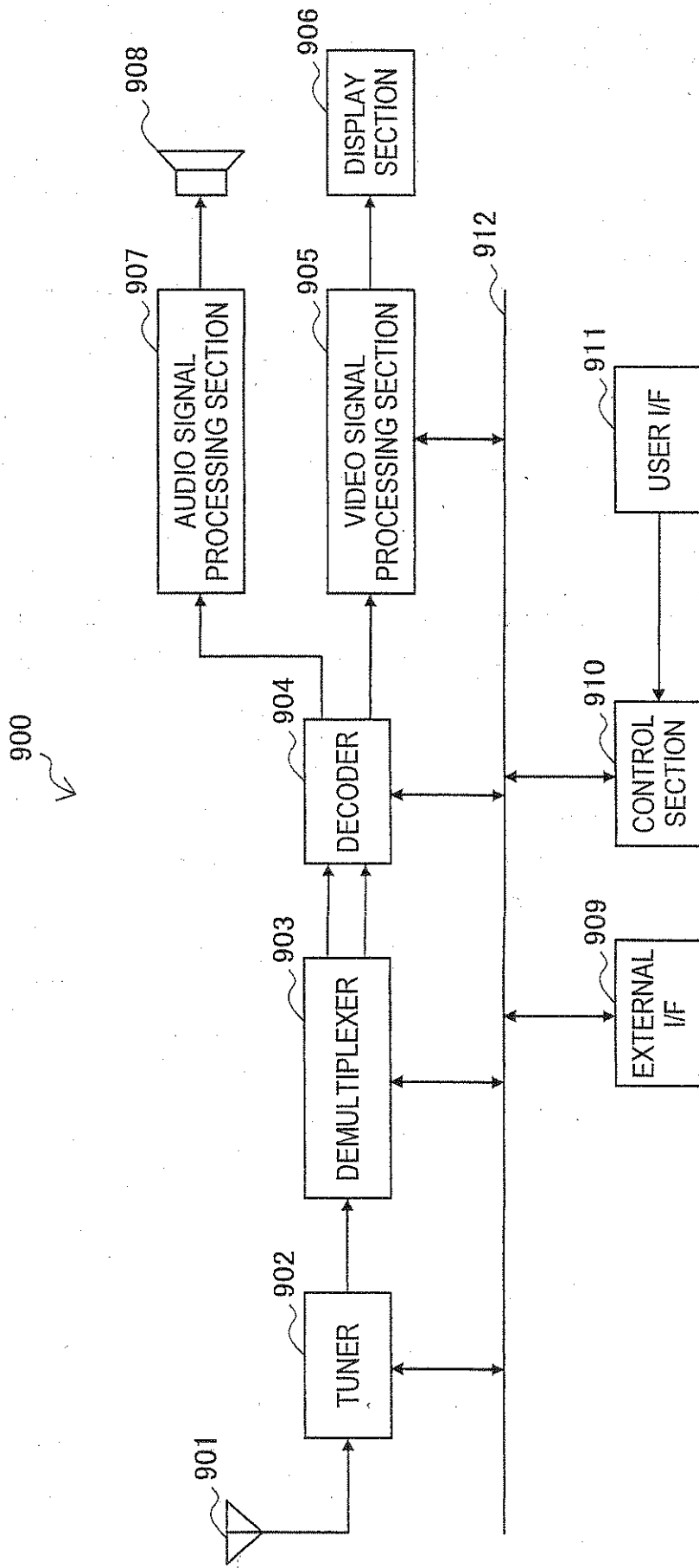


FIG.35

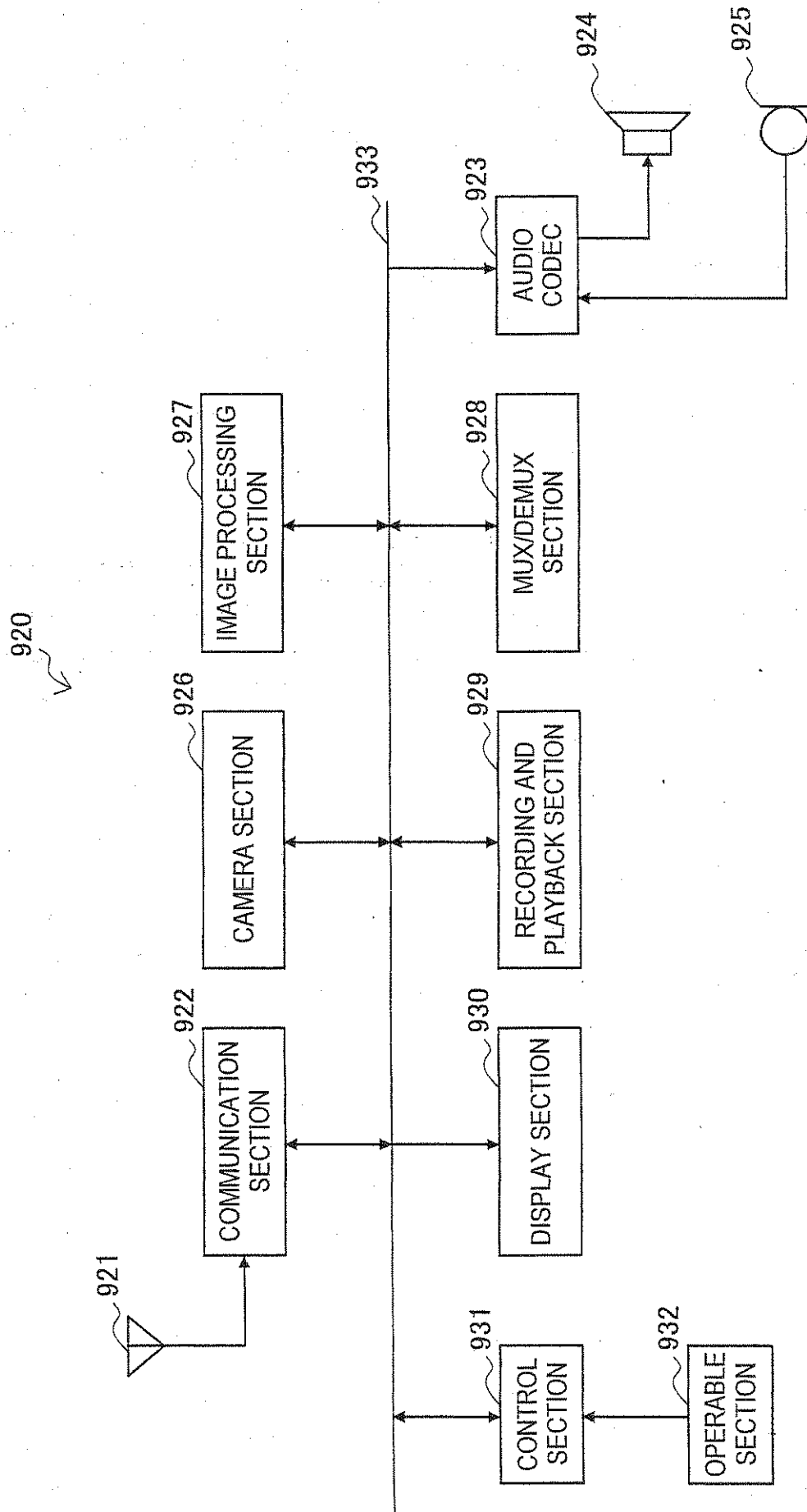


FIG.36

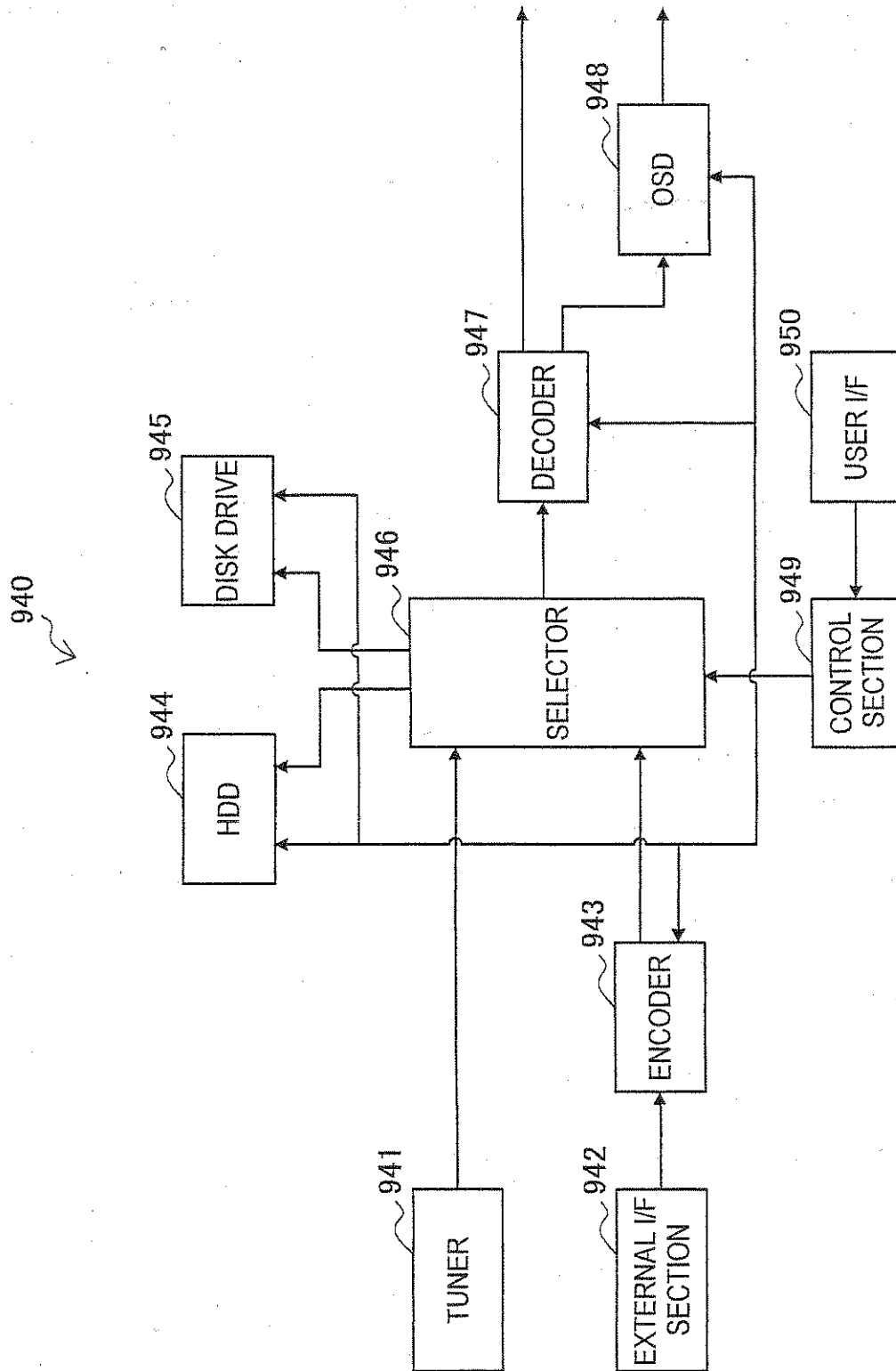


FIG.37

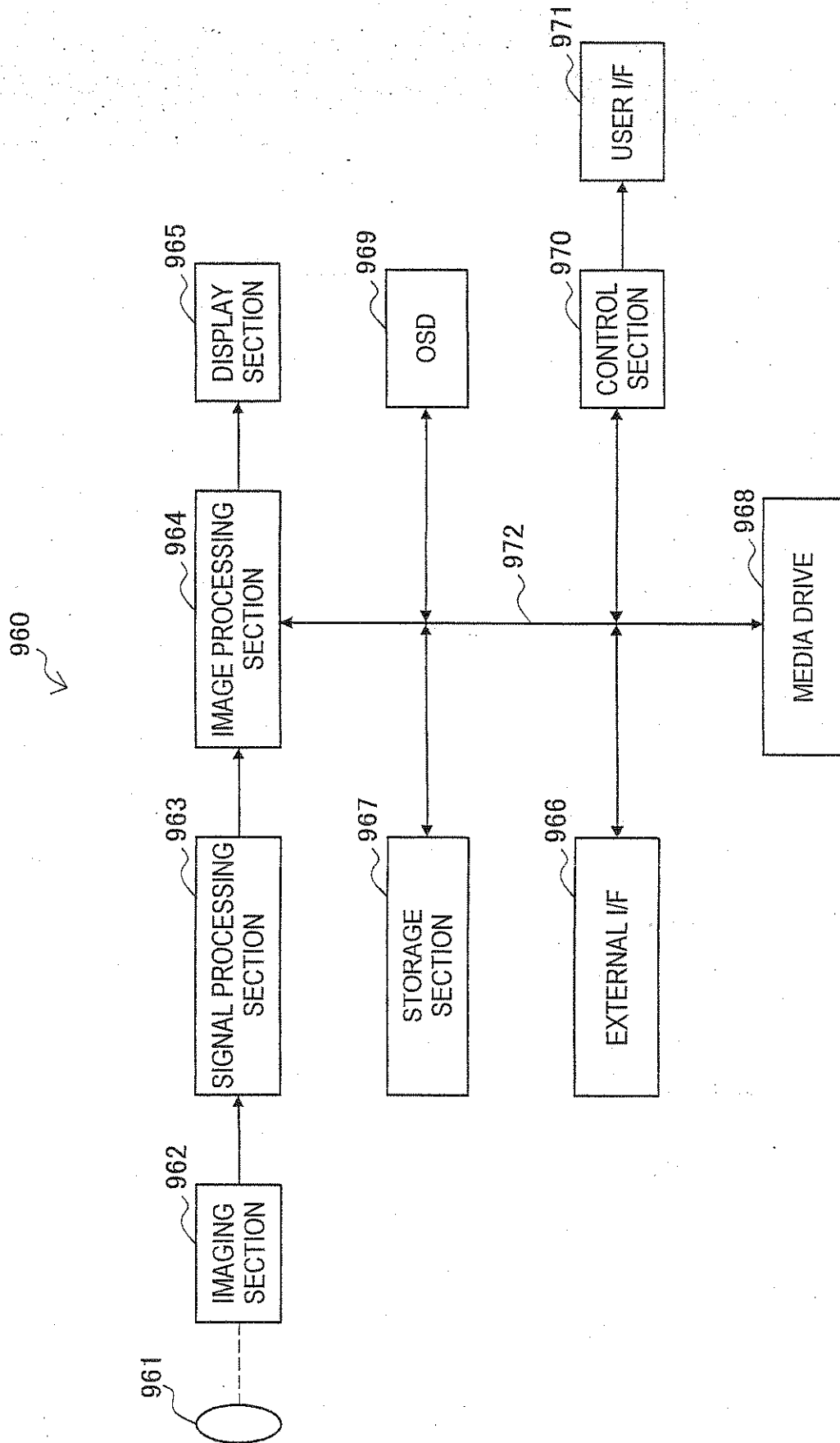


FIG.38

