

FIG. 1

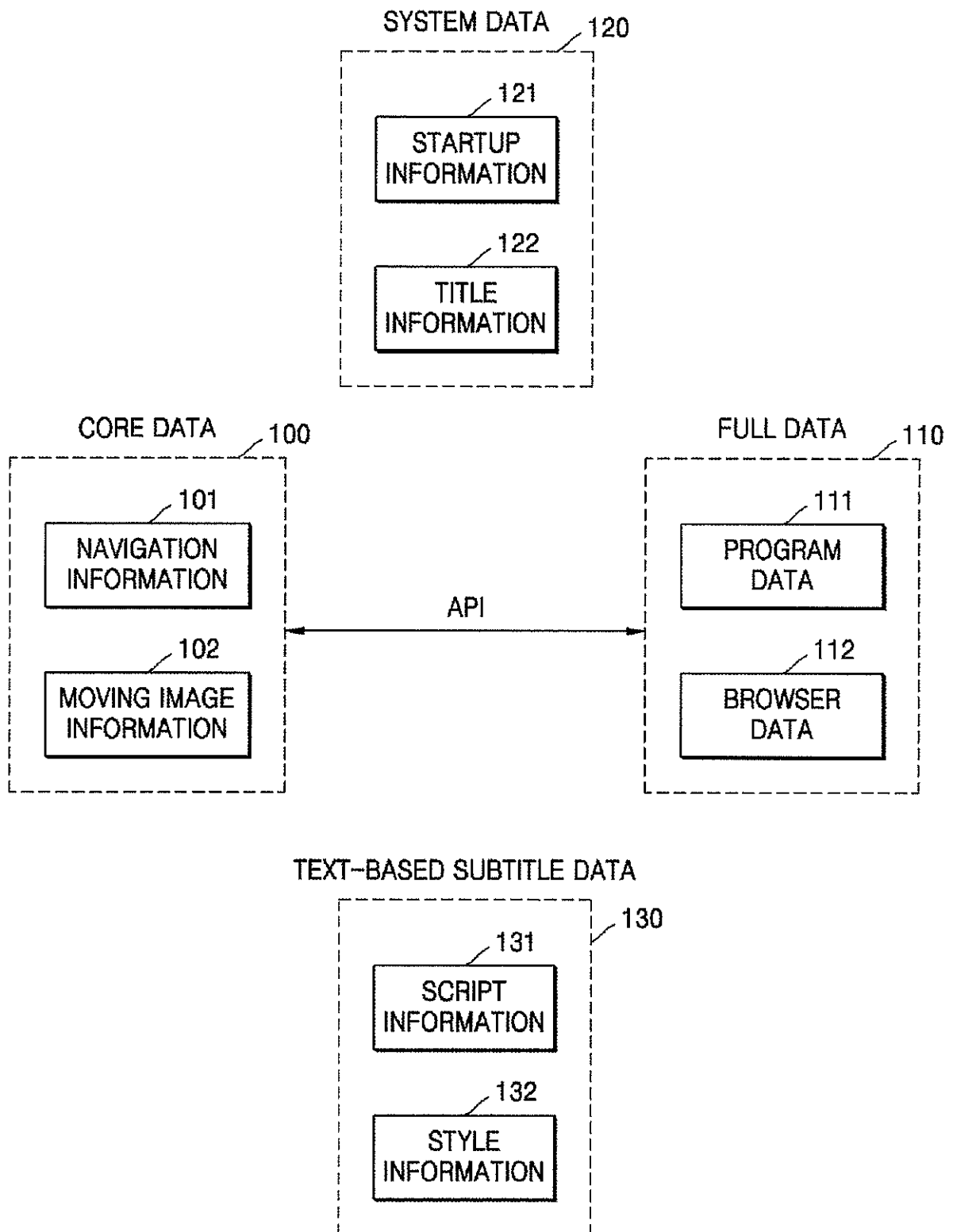


FIG. 2

Elements	Minimal Content Model	Attributes
201 subtitle	stylesheet, script	subtitle_id(ID), lang(LanguageCode), title(CDATA)
stylesheet	stylesheet+	
202 stylesheet	style+	stylesheet_id(ID), default("yes" "no"), desc(CDATA)
style	EMPTY	style_id(Class), font(Charset), color(Color), bgcolor(Color), size(Number), position(Number, Number), align("left" "right" "center"), region(Coords), lineheight(Number)
script	dialog+	
203 dialog	PCDATA	begin(PresentationTime), end(PresentationTime), ref_style_id(OREF)

FIG. 3

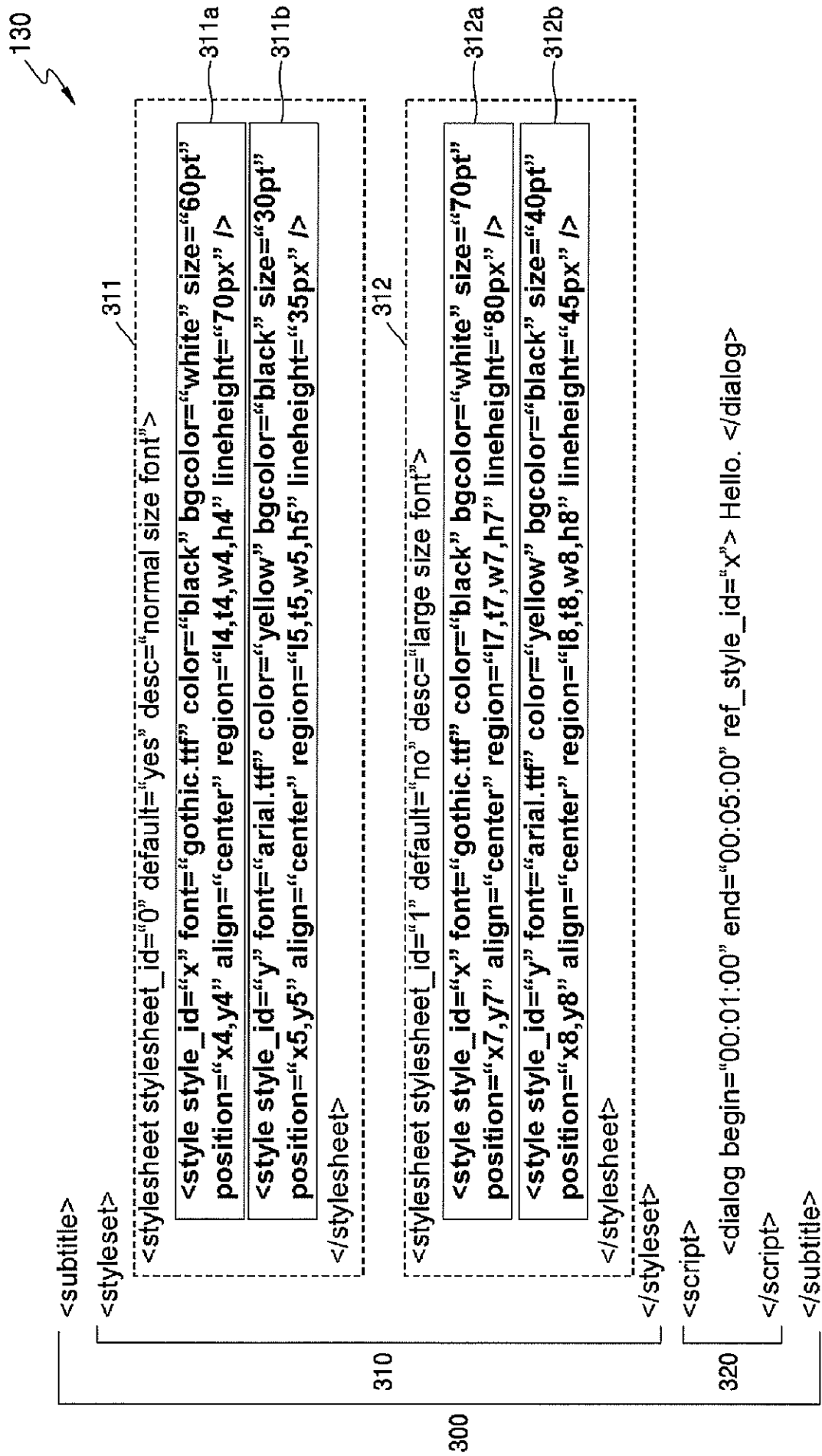


FIG. 4

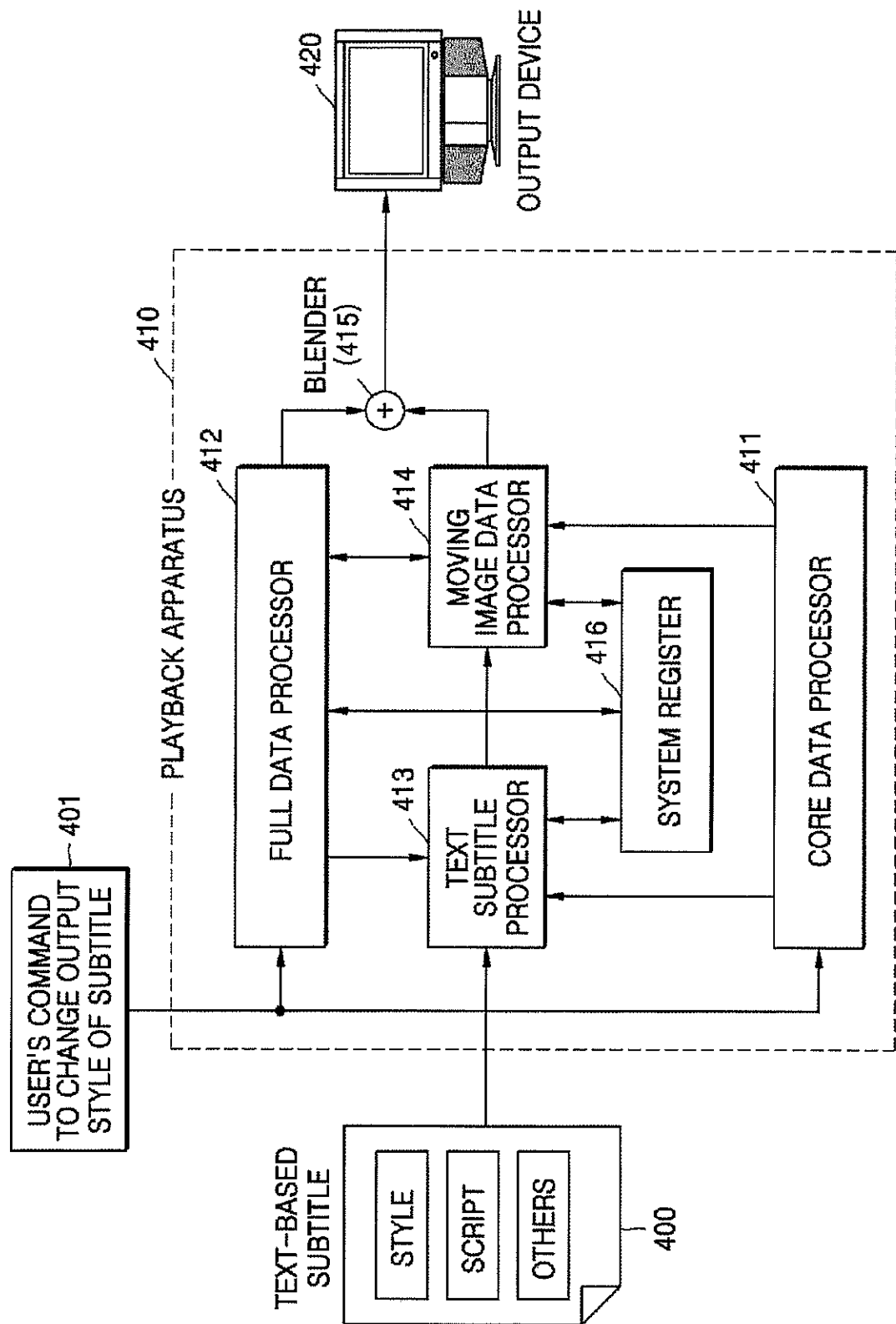


FIG. 5**Player Status Registers (PSRs)**

Reg. Num	Name	Meaning
0	—	Reserved
1	Audio	Audio Stream Number
2	Subtitle	Subtitle Stream Number or Text Subtitle id
...		
17	Subtitle Language	Language Code for Subtitle
...		
19	Subtitle Stylesheet	Stylesheet_id for Text Subtitle
20	—	Reserved
:	:	:
63	—	Reserved

FIG. 6**GENERAL PURPOSE REGISTERS (GPRs) OR BUFFER OF TEXT SUBTITLE PROCESSOR**

Address	Value	Variable name
0		
:		
100	102	Start_addr_of_Stylesheet_ID
101	2	Num_of_Stylesheet
102	ID for Stylesheet "Normal"	Stylesheet[0]
103	ID for Stylesheet "Large"	Stylesheet[1]
:		
4098		

FIG. 7

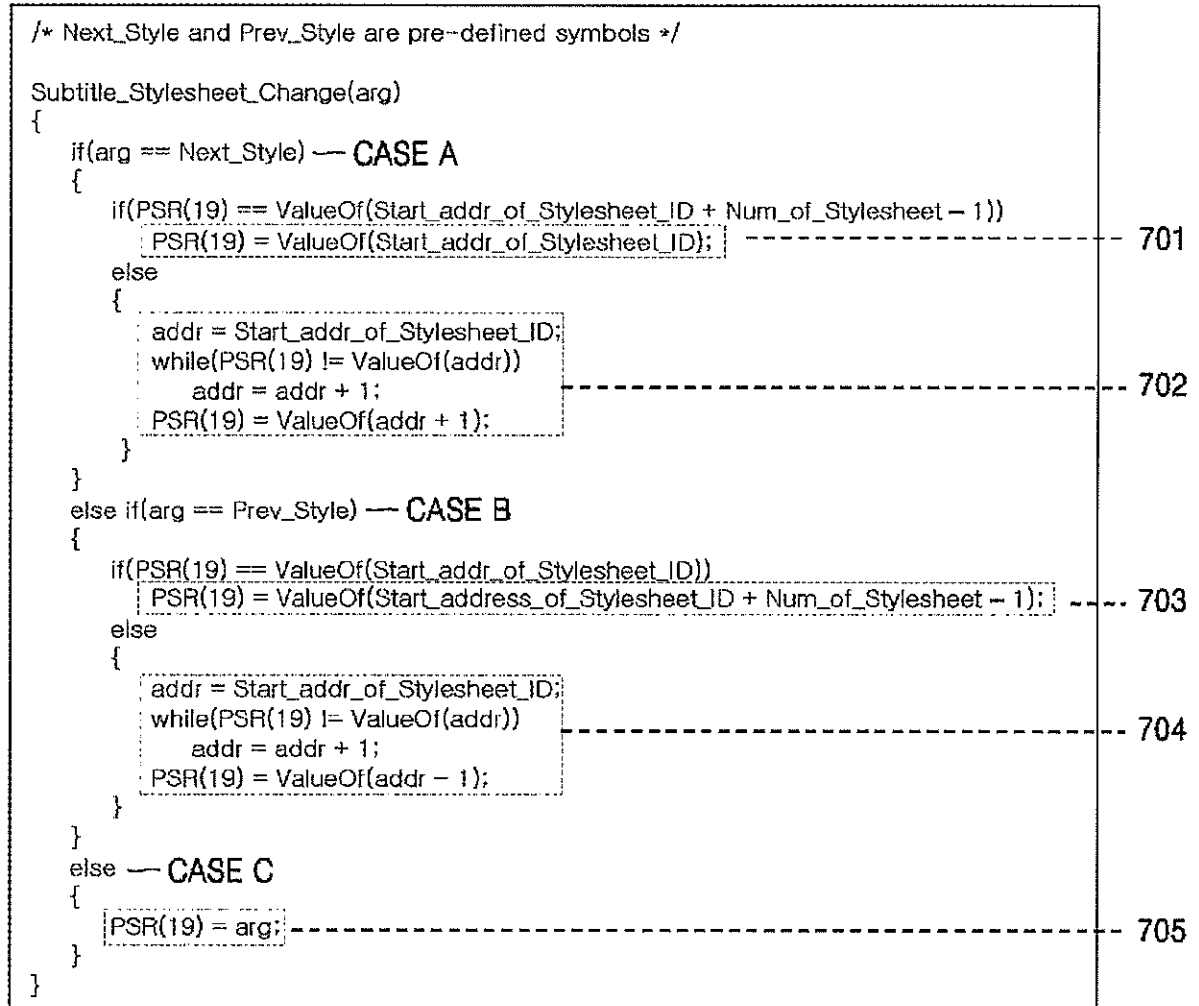


FIG. 8

```
package BDROM.media.subtitle

public interface TextSubtitle extends Subtitle {
    public boolean status;
    public TextSubtitleAttribute getAttribute();
    public unsigned int getLangCode();
    public String getTitleName();
    public unsigned int getSubtitleID();

    public unsigned int getCurrentTextStylesheetID();
    public void setTextStylesheetID(unsigned int ID);
    public void nextTextStylesheet();
    public void prevTextStylesheet();
    public unsigned int getDefaultTextStylesheetID();

    public Time getCurrentPresentationTime();
    public void setPresentationTime(Time t);

    public FILE getCurrentTextSubtitle();
    public void setTextSubtitle(String filename or unsigned int ID);

    public void start();
    public void stop();
}
```

801

FIG. 9

```

1 : import BDRom.media.*;

2 : public class PlayXlet extends Xlet implements ControllerListener {
3 :     Player player;

4 :     public void init() {
5 :         setLayout(new BorderLayout());
6 :         String mediaFile = getParameter("FILE");

7 :         try {
8 :             URL mediaURL = new URL(getDocumentBase(), mediaFile);
9 :             player = Manager.createPlayer(mediaURL);
10 :            player.addControllerListener(this);
11 :            player.subtitle = new TextSubtitle();
12 :            player.subtitle.setTextSubtitle("filename");
13 :        } catch (Exception e) {
14 :            System.out.println("Get Exception " + e);
15 :        }
    }

16 :    public void start() {
17 :        player.start();
    }

18 :    public synchronized void controllerUpdate(ControllerEvent e) {
19 :        if ( e instanceof SubtitleToggleEvent ) {
20 :            if (player.subtitle.status == TRUE)
21 :                player.subtitle.stop();
22 :            else {
23 :                player.subtitle.setPresentationTime(player.getMediaTime());
24 :                player.subtitle.start();
25 :            }
26 :        }
27 :        if ( e instanceof nextSubtitleStylesheetEvent ) {
28 :            player.subtitle.nextTextStylesheet();
29 :        }
30 :        if ( e instanceof prevSubtitleStylesheetEvent ) {
31 :            player.subtitle.prevTextStylesheet();
32 :        }
33 :        if ( e instanceof setSubtitleStylesheetEvent ) {
34 :            player.subtitle.setTextStylesheetID(e.value);
35 :        }
    }
}

```

FIG. 10

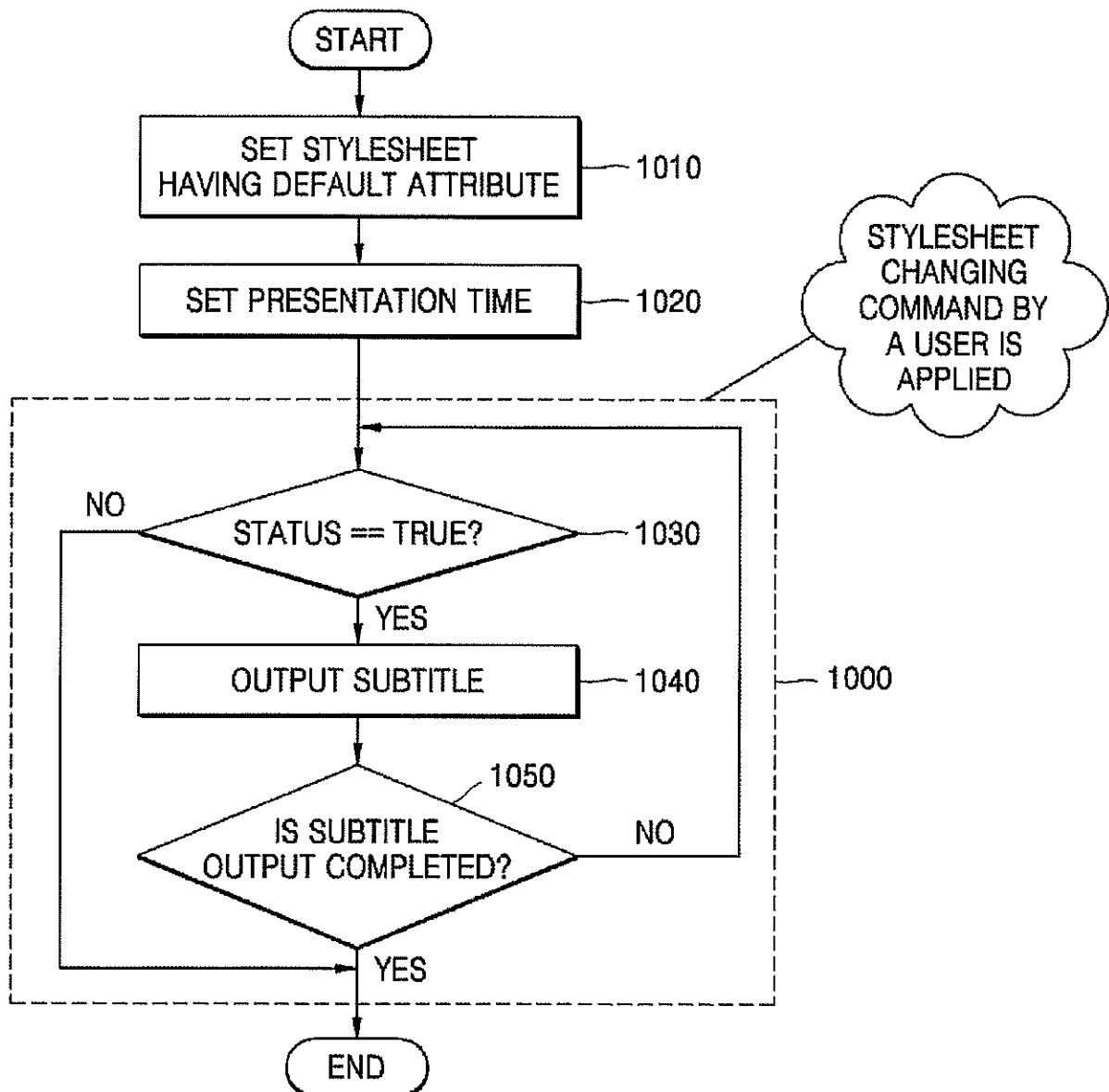


FIG. 11

