PRIOR ART

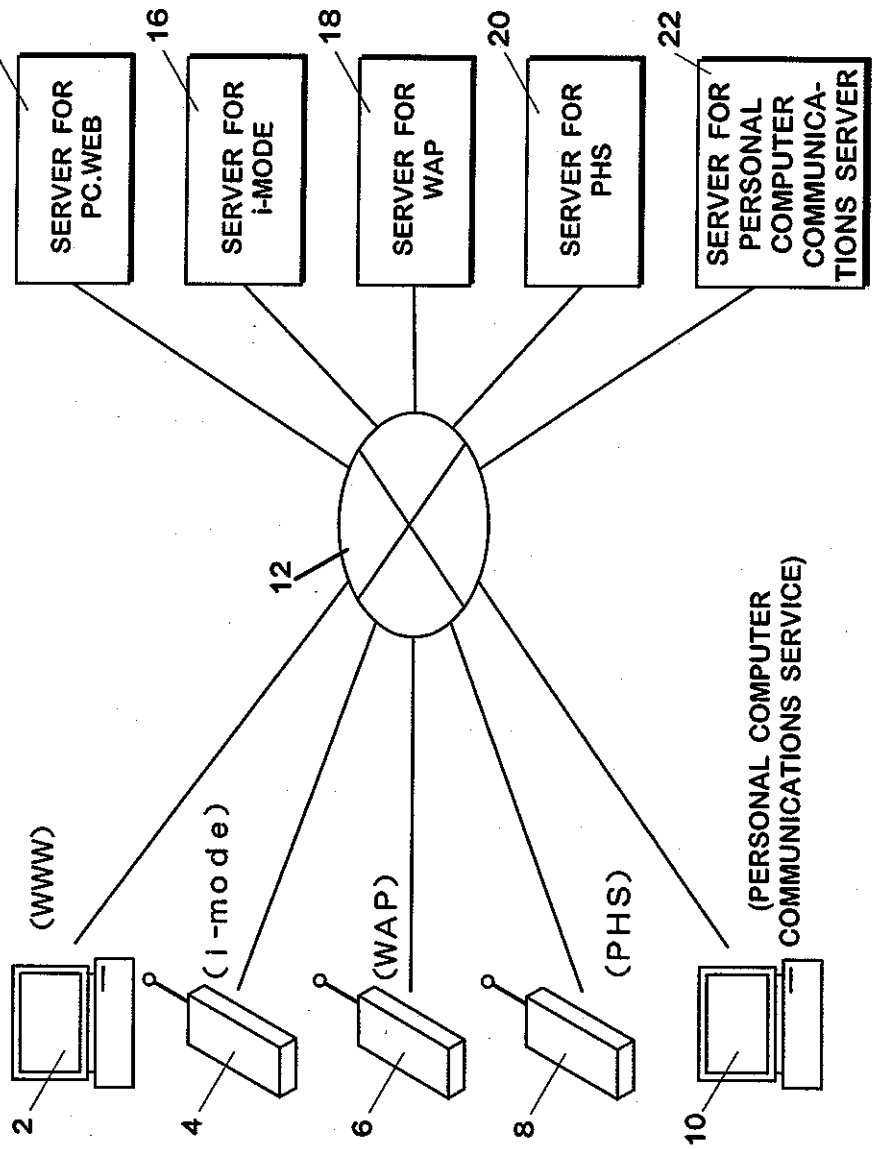
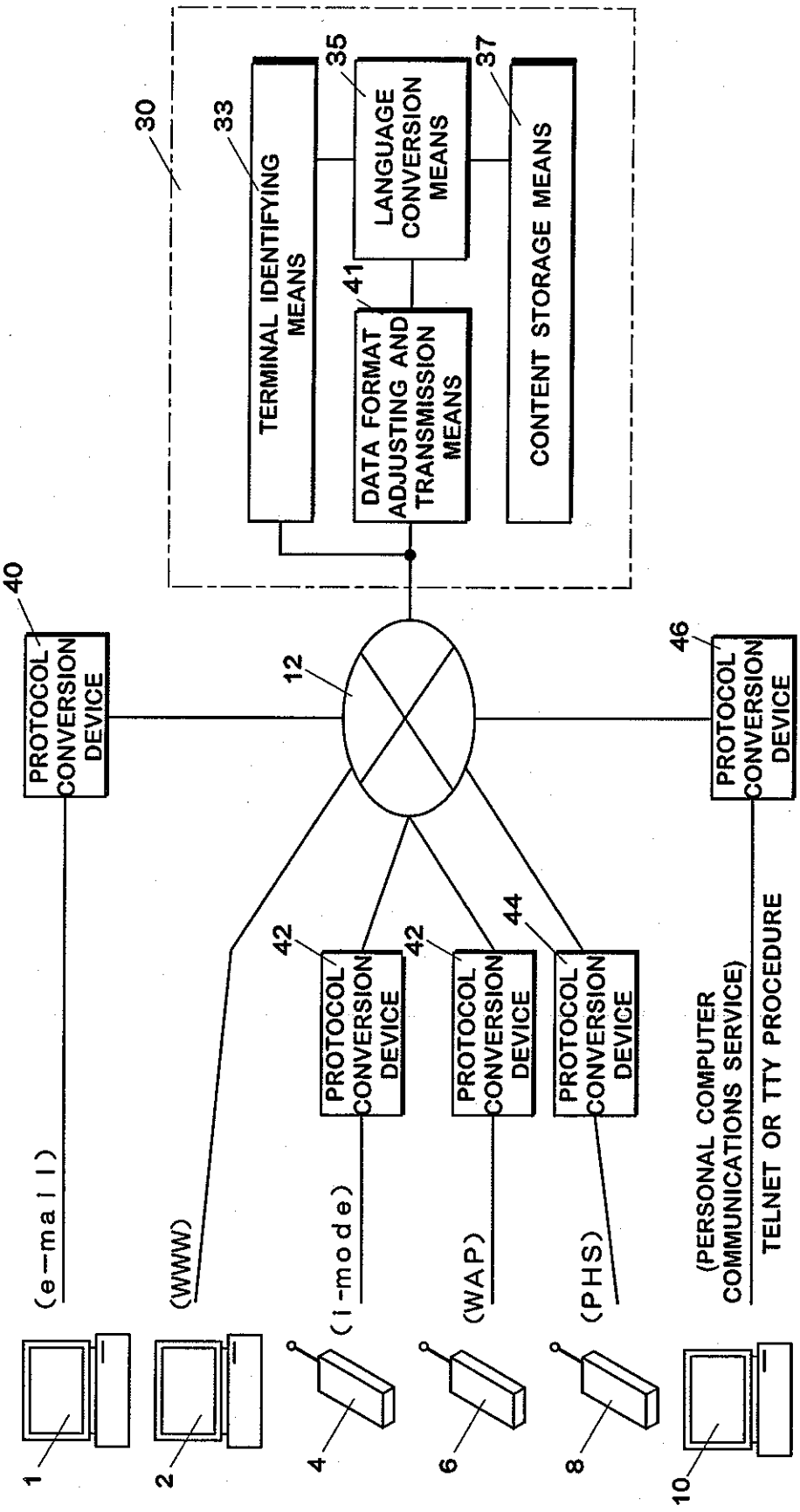


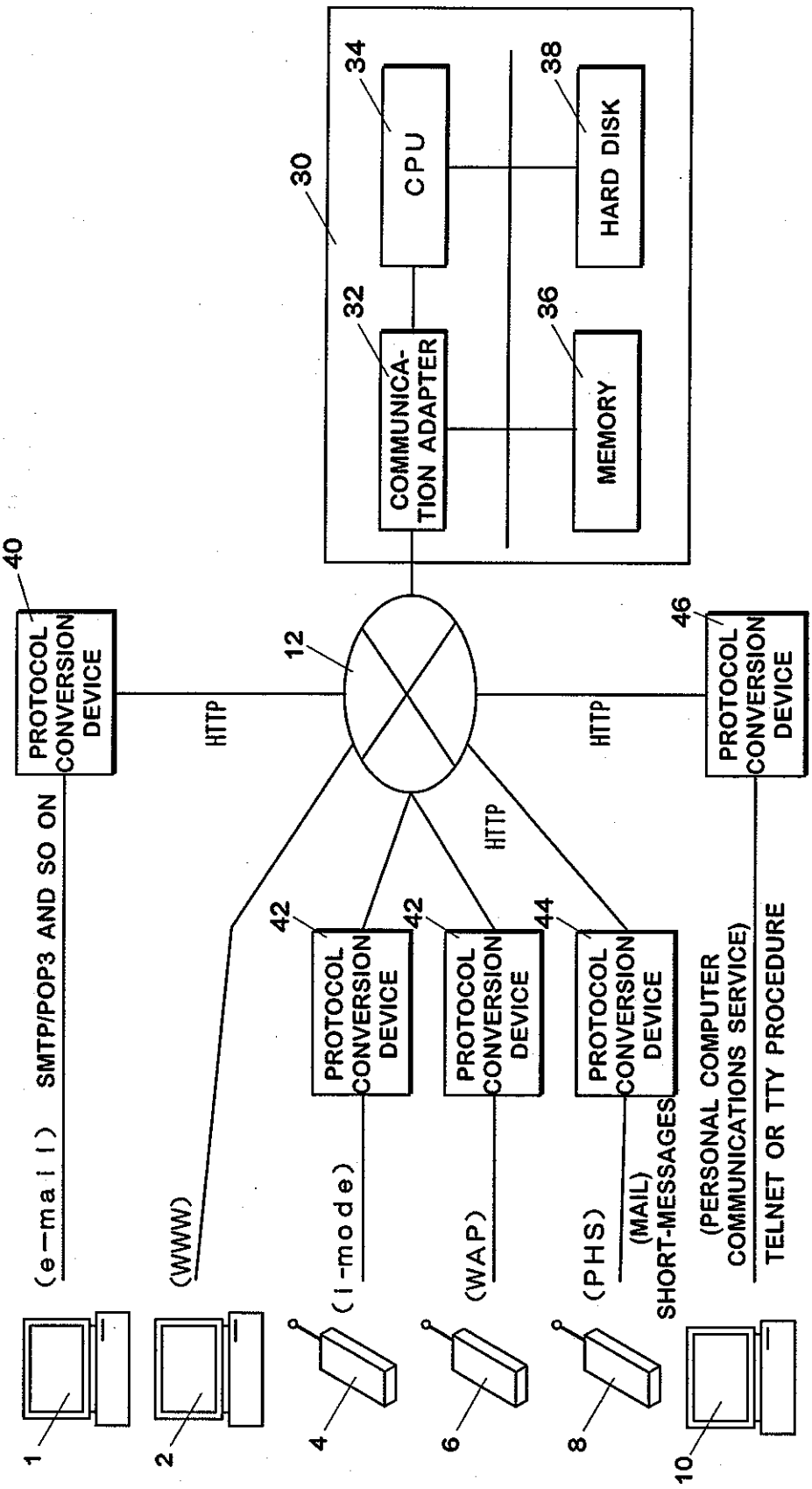
FIG.1

FIG.2a


30
33
TERMINAL IDENTIFYING MEANS
35
LANGUAGE CONVERSION MEANS
37
CONTENT STORAGE MEANS
41
DATA FORMAT ADJUSTING AND TRANSMISSION MEANS
40
PROTOCOL CONVERSION DEVICE
12
46
PROTOCOL CONVERSION DEVICE
42
PROTOCOL CONVERSION DEVICE
42
PROTOCOL CONVERSION DEVICE
44
PROTOCOL CONVERSION DEVICE
(e-mail)
(WWW)
(i-mode)
(WAP)
(PHS)
(PERSONAL COMPUTER COMMUNICATIONS SERVICE)
TELNET OR TTY PROCEDURE
1
2
4
6
8
10

30 JUN 2004

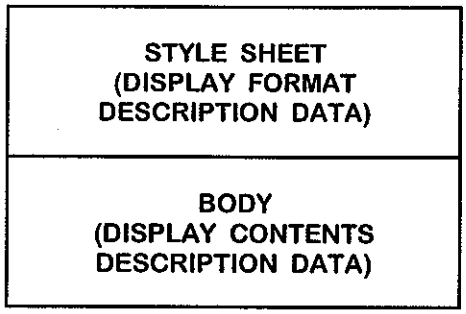
FIG.2b



30 JUN 2004

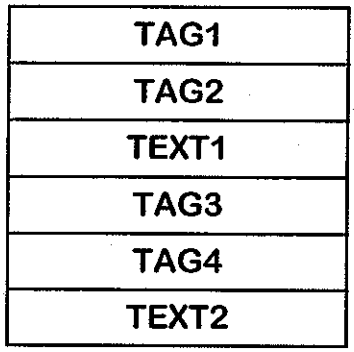
FIG.3

DATA STRUCTURE OF DATA
WRITTEN IN BASIC LANGUAGE



30 JUN 2004

FIG.4



30 JUN 2004



FIG.5

30 JUN 2004

FIG.6

BODY

MENU.KSP

```
<KSP KEY=K1 NAME="index menu">
<P KEY=R1> SELECT THE MENU</P>
<IMG KEY=R2 SRC=ICOM.JPG>
<A KEY=R3 HREF=MAIL.KSP> E-MAIL</A>
<A KEY=R4 HREF=BBS. KSP> BBS</A>
</KSP>
```

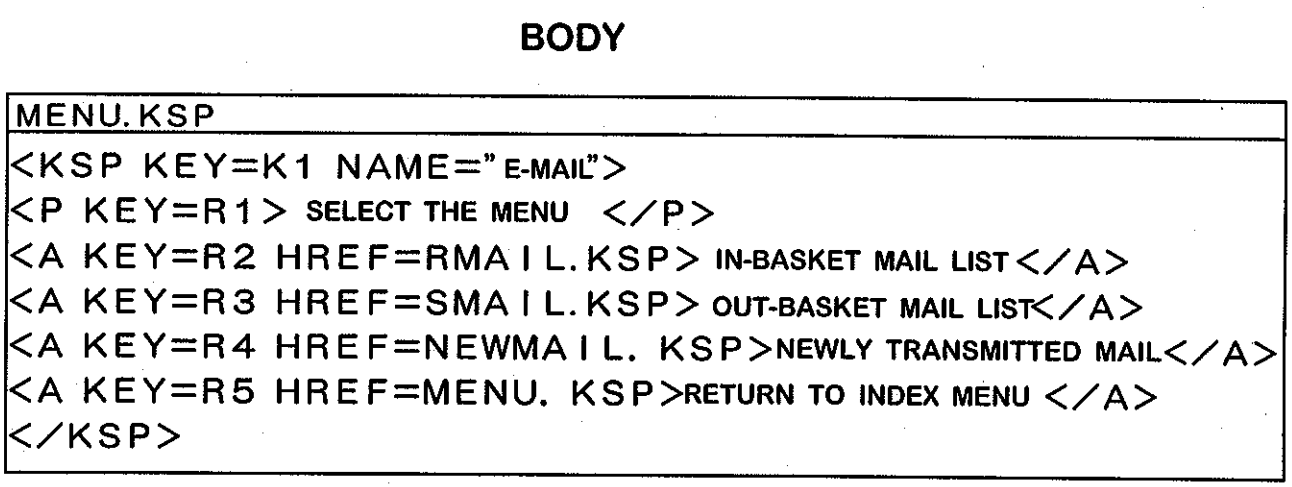
FIG.7

STYLE SHEET

```
<HTML>
<HEAD><TITLE>$K1.NAME$</TITLE></HEAD>
<BODY BACKGROUND=bg.gif>
$R1$<BR>
<IMG SRC=$R2.SRC$>
<A HREF=$R3.HREF$>$R3</A>
<A HREF=$R4.HREF$>$R4</A>
</BODY>
</HTML>
```


30 JUN 2004

FIG.8



30 JUN 2004

FIG.9

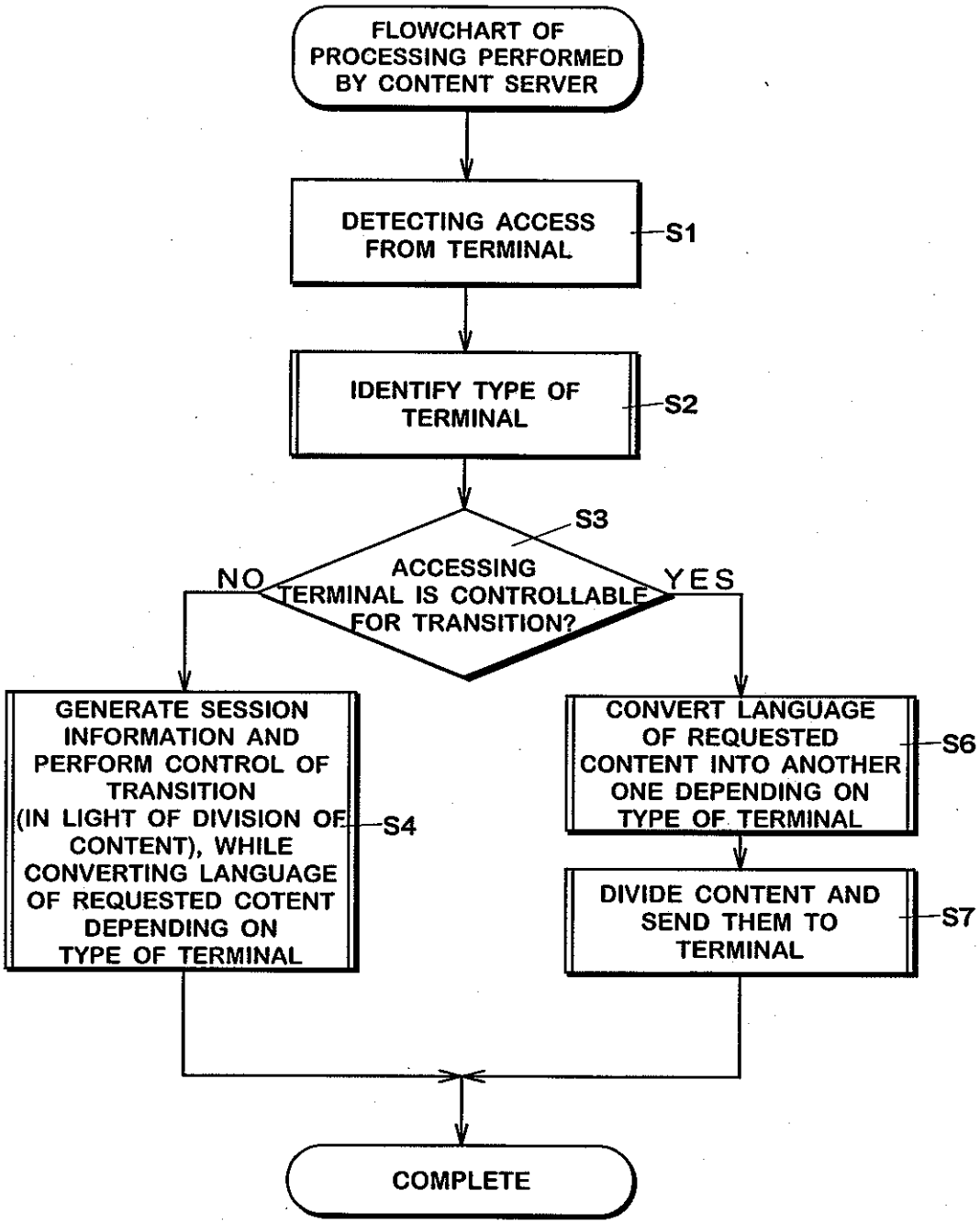
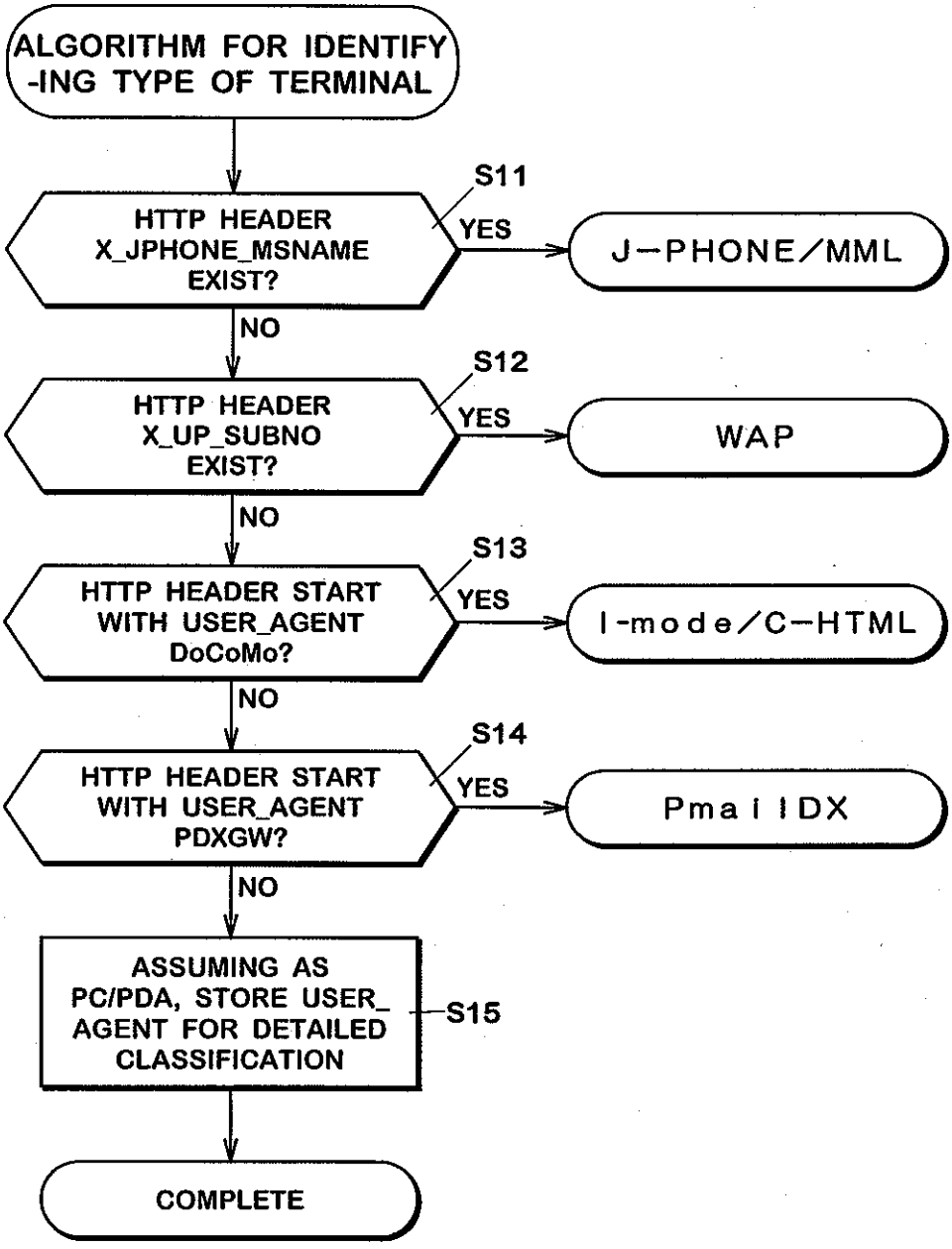
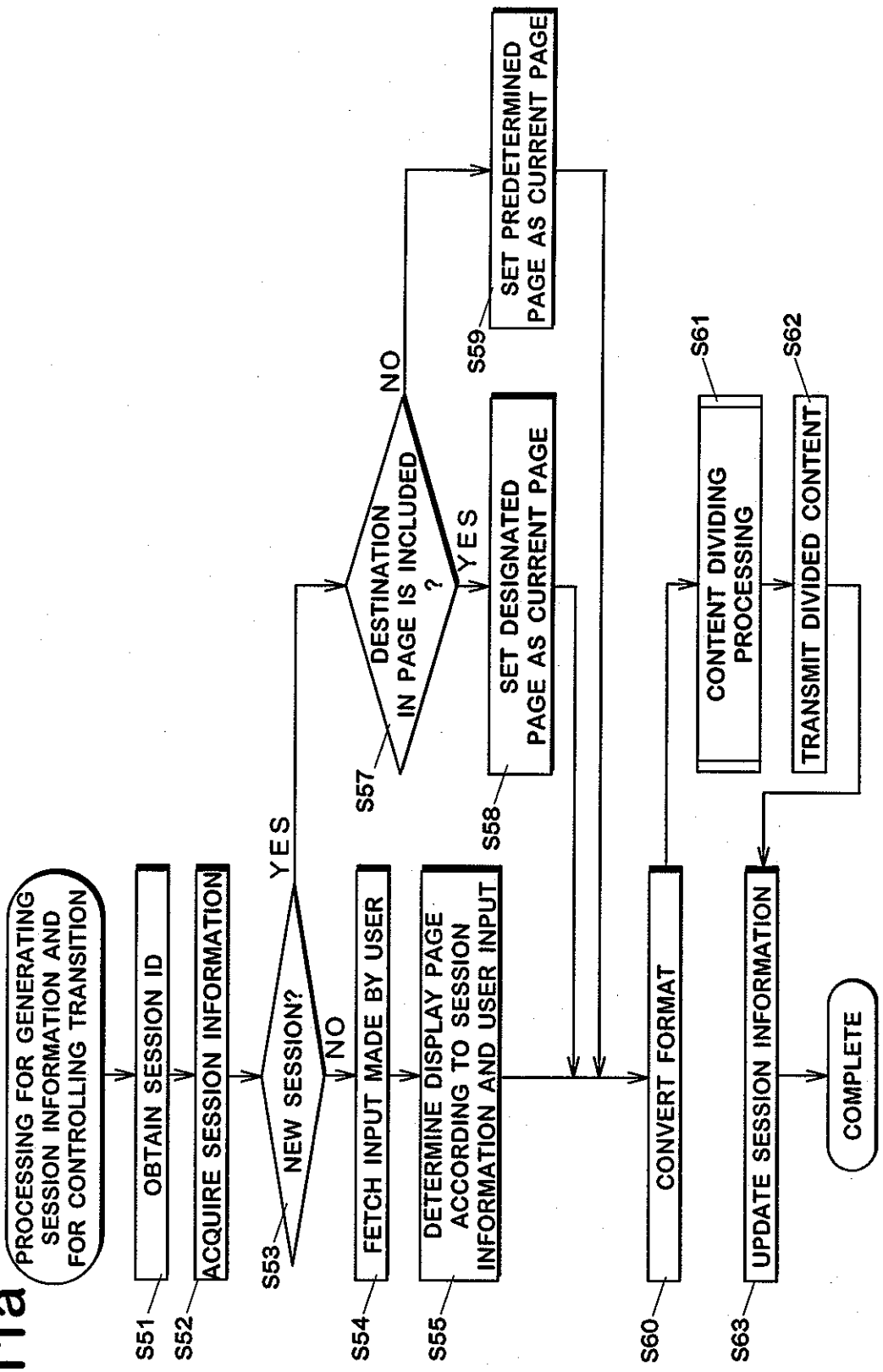

FLOWCHART OF PROCESSING PERFORMED BY CONTENT SERVER
DETECTING ACCESS FROM TERMINAL
S1
IDENTIFY TYPE OF TERMINAL
S2
S3
ACCESSING TERMINAL IS CONTROLLABLE FOR TRANSITION?
NO
YES
GENERATE SESSION INFORMATION AND PERFORM CONTROL OF TRANSITION (IN LIGHT OF DIVISION OF CONTENT), WHILE CONVERTING LANGUAGE OF REQUESTED COTENT DEPENDING ON TYPE OF TERMINAL
S4
CONVERT LANGUAGE OF REQUESTED CONTENT INTO ANOTHER ONE DEPENDING ON TYPE OF TERMINAL
S6
DIVIDE CONTENT AND SEND THEM TO TERMINAL
S7
COMPLETE

FIG.10


ALGORITHM FOR IDENTIFY -ING TYPE OF TERMINAL
S11
HTTP HEADER X_JPHONE_MSNAME EXIST?
YES
J-PHONE/MML
NO
S12
HTTP HEADER X_UP_SUBNO EXIST?
YES
WAP
NO
S13
HTTP HEADER START WITH USER_AGENT DoCoMo?
YES
I-mode/C-HTML
NO
S14
HTTP HEADER START WITH USER_AGENT PDXGW?
YES
PmailDX
NO
ASSUMING AS PC/PDA, STORE USER_ AGENT FOR DETAILED CLASSIFICATION
S15
COMPLETE

30 JUN 2004

FIG.11a



30 JUN 2004

FIG.11b

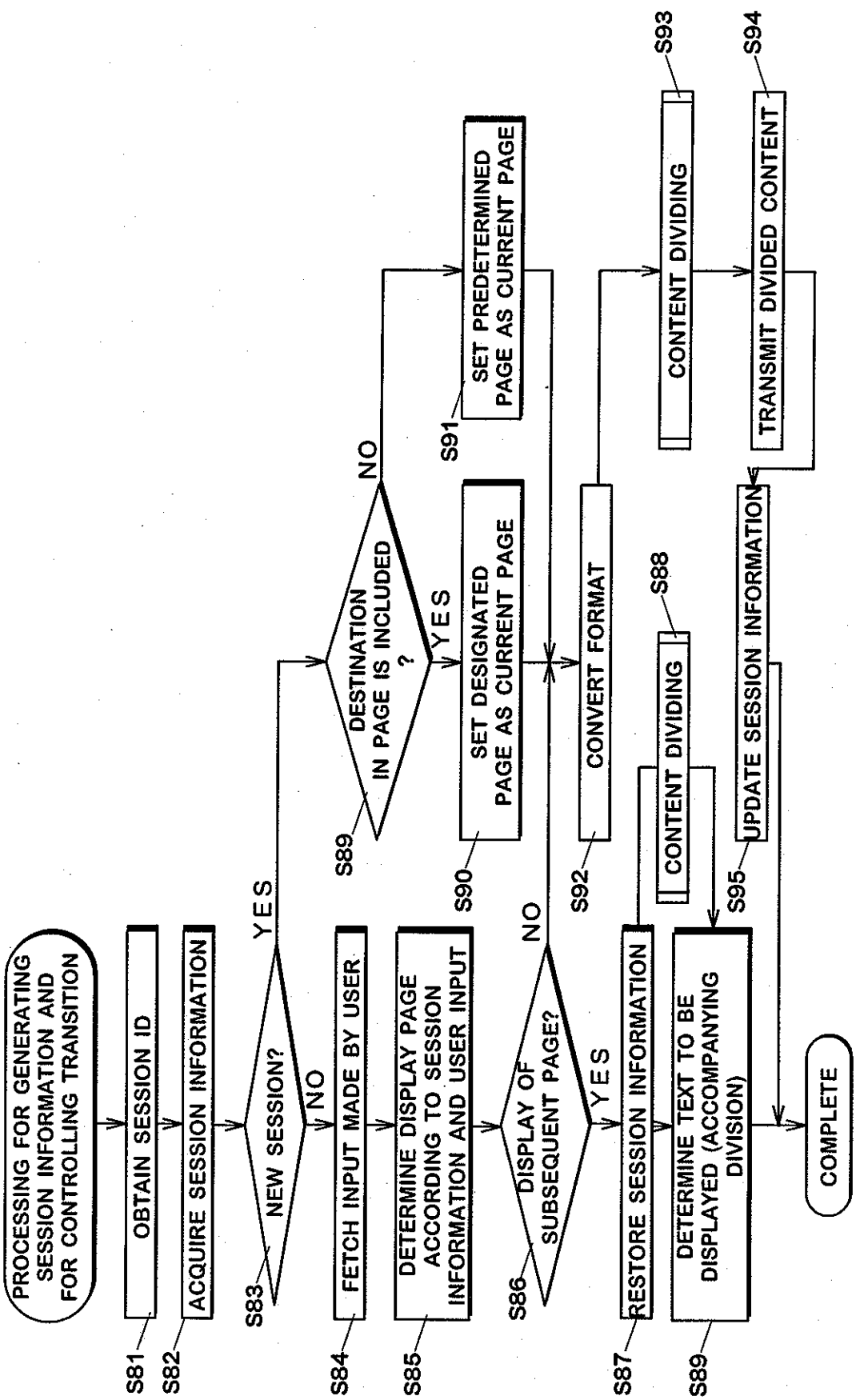
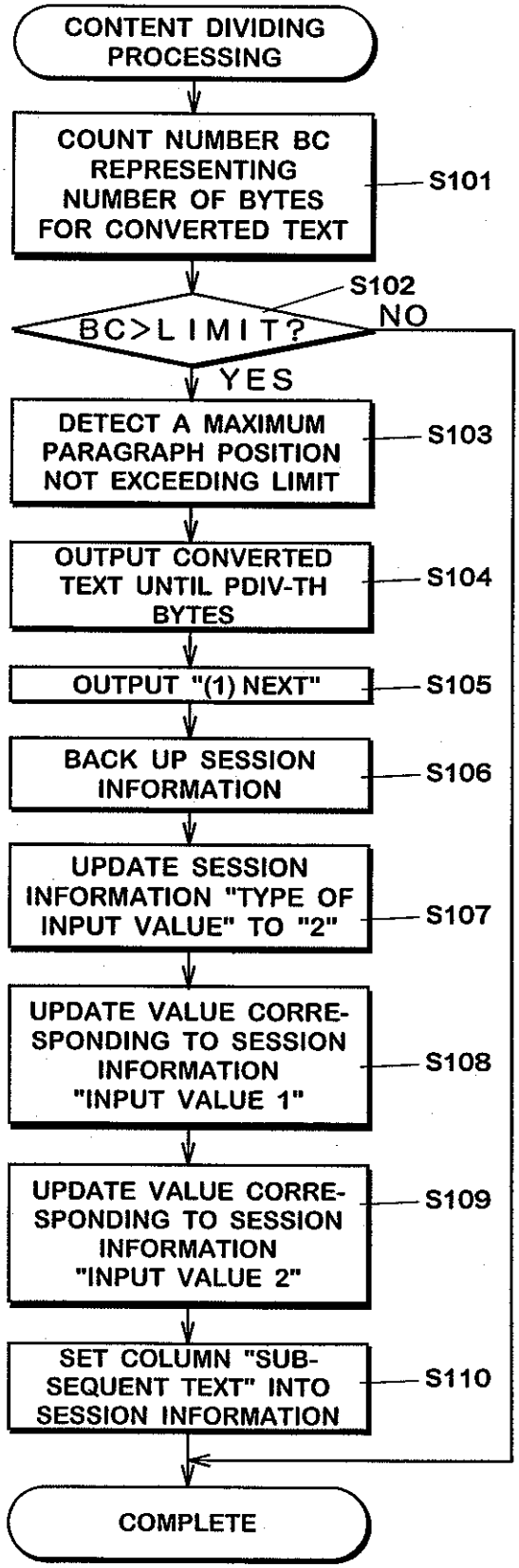


FIG.11c


CONTENT DIVIDING PROCESSING
COUNT NUMBER BC REPRESENTING NUMBER OF BYTES FOR CONVERTED TEXT
S101
S102
BC>LIMIT?
NO
YES
DETECT A MAXIMUM PARAGRAPH POSITION NOT EXCEEDING LIMIT
S103
OUTPUT CONVERTED TEXT UNTIL PDIV-TH BYTES
S104
OUTPUT "(1) NEXT"
S105
BACK UP SESSION INFORMATION
S106
UPDATE SESSION INFORMATION "TYPE OF INPUT VALUE" TO "2"
S107
UPDATE VALUE CORRESPONDING TO SESSION INFORMATION "INPUT VALUE 1"
S108
UPDATE VALUE CORRESPONDING TO SESSION INFORMATION "INPUT VALUE 2"
S109
SET COLUMN "SUBSEQUENT TEXT" INTO SESSION INFORMATION
S110
COMPLETE

30 JUN 2004

FIG.11d

LIMITATION IN SIZE LIMIT

TYPE OF TERMINAL	MAX. SIZE
PmailDX	1000
i-mode	2000
WAP	1200
PC	INFINITY (UMLIMITED BYTES)

30 JUN 2004

FIG.12

TYPE OF TERMINAL	HOW TO IDENTIFY SESSION IDS
PmailDX	SESSION ID ASSIGNED AS URL BY GATEWAY SERVER
MAIL RESPONSE	E-MAIL ADDRESS
PERSONAL COMPUTER COMMUNICATION	NUMBER OF COMMUNICATION PORT
TELNET	IP ADDRESS OF THE TERMINAL

30 JUN 2004

FIG.13

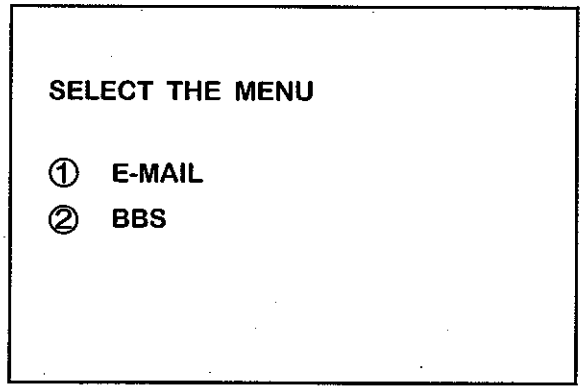
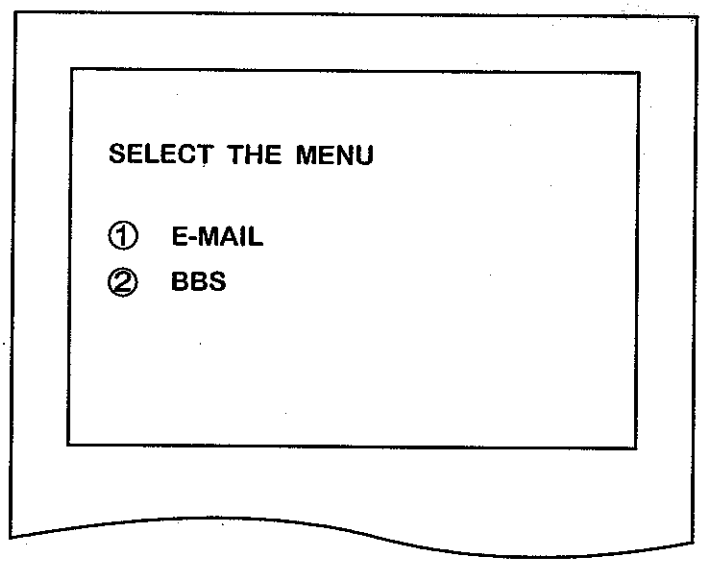


FIG.14

SESSION INFORMATION

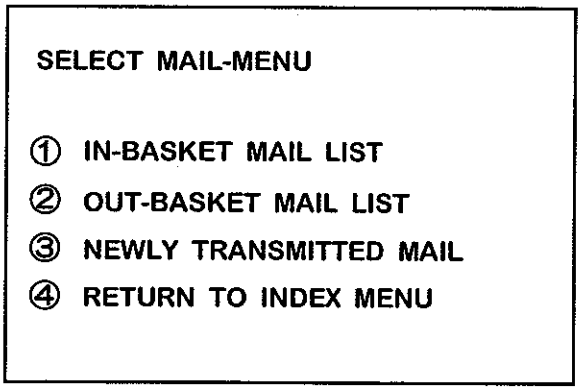
SESSION ID	AKZ38452
TIME FOR TERMINATING SESSION	2000/01/15 17:35:21
TYPE OF INPUT VALUES	3
SERIES OF CHARACTERS IN INPUT1	"1"
STATUS OF INPUT1	NORMAL
RESPONSE TO INPUT1	TRANSIT TO MAIL.KSP
SERIES OF CHARACTERS IN INPUT2	"2"
STATUS OF INPUT2	NORMAL
RESPONSE TO INPUT2	TRANSIT TO BBS.KSP
SERIES OF CHARACTERS IN INPUT3	"*"
STATUS OF INPUT3	ERROR
RESPONSE TO INPUT3	TRANSIT TO MENU.KSP

FIG.15



30 JUN 2004

FIG.16



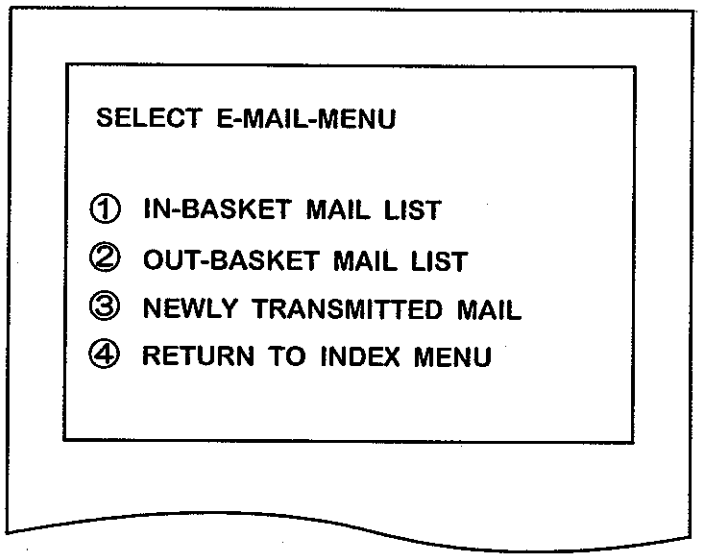
30 JUN 2004

FIG.17

SESSION ID	AKZ38452
TIME FOR TERMINATING SESSION	2000/01/15 17:36:05
TYPE OF INPUT VALUES	5
SERIES OF CHARACTERS IN INPUT1	"1"
STATUS OF INPUT1	NORMAL
RESPONSE TO INPUT1	TRANSIT TO RMAIL.KSP
SERIES OF CHARACTERS IN INPUT2	"2"
STATUS OF INPUT2	NORMAL
RESPONSE TO INPUT2	TRANSIT TO SMAIL.KSP
SERIES OF CHARACTERS IN INPUT3	"3"
STATUS OF INPUT3	NORMAL
RESPONSE TO INPUT3	TRANSIT TO NEWMAIL.KSP
SERIES OF CHARACTERS IN INPUT4	"4"
STATUS OF INPUT4	NORMAL
RESPONSE TO INPUT4	TRANSIT TO MENU.KSP
SERIES OF CHARACTERS IN INPUT5	"*"
STATUS OF INPUT5	ERROR
RESPONSE TO INPUT5	TRANSIT TO MENU.KSP

30 JUN 2004

FIG.18



30 JUN 2004

FIG.19A

DINING.KSP

<KSP KEY=K1 NAME="THE PHILOSOPHERS HAVING MEALS">
<P KEY=R1>

A problem named " philosopher s having meals " was proposed by Dijkstra as one of research task s in parallel processing. The details of which is described hereunder.

A total of five (5) philosophers sit around a round table and having meals as the occasion may demand while they involving into thoughts. Spaghetti piled on a dish put on the center of the table is served as the meal for them. In order to have the spaghetti, each of them need to hold fork in both hands because these are too entagled with one another.

A total of five (5) forks, each of which is placed just between each of the philosophers. This meant that they can not use a fork on his/her right during another philosopher who sits right hand side enjoys the meal, vice versa.

The countermeasure for such situation is to write a program(s) for make the philosophers full without starving any of them.

Such program can be recognized as a modeling for parallel processing, and the forks are considered as resources commonly used. In addition, a section in which these philosophers try to grab the forks ahead of others can be considered as critical section.

Suppose one of the solutions, a simple program described below may be do.

The whole concept of the program is that the philosophers wait for forks located at their right hand side, and grab forks on their left hand side once the forks at right are available. They start having meal after their both hands grab the forks.

30 JUN 2004

FIG.19B

This algorithm easily reach an impasse. Assume the case in which all the five philosophers simultaneously hold the forks at their right hand side. Although, they all waiting for the forks at their left hand side available, no forks at their left hand side will be available forever because none of them release the forks at their right hand. As a consequence, all of them are getting hungry a bit at a time with holding forks at their right hands. This is a typical example of so called "dead lock".

In a less drastic case, there is a possibility that some philosophers become starving. For example, a philosopher, who sits the middle of two evil fellows who alternately take meals on purpose, starves because both the forks at right and left hand sides are occupied by these two fellows. This is a phenomenon so called "lock out".

The problem named " a philosophers having meals " have often been discussed as a material of parallel programming because this can treat issues on collective occupation of common resources and critical section while the problem seems simple to resolve. This could be a famous problem in view of consecutive algorithm equivalent to a game so called "Eight queens".

</P>
<A KEY=R2 HREF=CSECTION.KSP> DESCRIPTION OF CRITICAL SECTION</A>
<A KEY=R3 HREF=CONCURRENT.KSP>DESCRIPTION OF CONCURRENT</A>
<A KEY=R4 HREF=INDEX.KSP>TABLE OF CONTENTS</A>
</KSP>

30 JUN 2004

FIG.20A

CONVERTED CONTENT

A problem named " philosopher s having meals" was proposed by Dijkstra as one of research task s in parallel processing. The details of which is described hereunder.

A total of five (5) philosophers sit around a round table and having meals as the occasion may demand while they involving into thoughts. Spaghetti piled on a dish put on the center of the table is served as the meal for them. In order to have the spaghetti, each of them need to hold fork in both hands because these are too entagled with one another.

A total of five (5) forks, each of which is placed just between each of the philosophers. This meant that they can not use a fork on his/her right during another philosopher who sits right hand side enjoys the meal, vice versa.

The countermeasure for such situation is to write a program(s) for make the philosophers full without starving any of them.

Such program can be recognized as a modeling for parallel processing, and the forks are considered as resources commonly used. In addition, a section in which these philosophers try to grab the forks ahead of others can be considered as critical section.

Suppose one of the solutions, a simple program described below may be do.

The whole concept of the program is that the philosophers wait for forks located at their right hand side, and grab forks on their left hand side once the forks at right are available. They start having meal after their both hands grab the forks.

30 JUN 2004

FIG.20B

This algorithm easily reach an impasse. Assume the case in which all the five philosophers simultaneously hold the forks at their right hand side. Although, they all waiting for the forks at their left hand side available, no forks at their left hand side will be available forever because none of them release the forks at their right hand. As a consequence, all of them are getting hungry a bit at a time with holding forks at their right hands. This is a typical example of so called "dead lock".

In a less drastic case, there is a possibility that some philosophers become starving. For example, a philosopher, who sits the middle of two evil fellows who alternately take meals on purpose, starves because both the forks at right and left hand sides are occupied by these two fellows. This is a phenomenon so called "lock out".

The problem named " a philosophers having meals " have often been discussed as a material of parallel programming because this can treat issues on collective occupation of common resources and critical section while the problem seems simple to resolve. This could be a famous problem in view of consecutive algorithm equivalent to a game so called" Eight queens".

(1) DESCRIPTION OF CRITICAL SECTION
(2) DESCRIPTION OF CONCURRENT
(3) TABLE OF CONTENTS

30 JUN 2004

FIG.21

SESSION ID	QEZ15434
TIME FOR TERMINATING SESSION	2000/01/15 17:35:21
TYPE OF INPUT VALUES	4
SERIES OF CHARACTERS IN INPUT1	"1"
STATUS OF INPUT1	NORMAL
RESPONSE TO INPUT1	TRANSIT TO CSECTION.KSP
SERIES OF CHARACTERS IN INPUT2	"2"
STATUS OF INPUT2	NORMAL
RESPONSE TO INPUT2	TRANSIT TO CONCURRENT.KSP
SERIES OF CHARACTERS IN INPUT3	"3"
STATUS OF INPUT3	NORMAL
RESPONSE TO INPUT3	TRANSIT TO INDEX.KSP
SERIES OF CHARACTERS IN INPUT4	"4"
STATUS OF INPUT4	NORMAL
RESPONSE TO INPUT4	TRANSIT TO DINING.KSP

FIG.22

TYPE OF INPUT VALUES	2
SERIES OF CHARACTERS IN INPUT1	"1"
STATUS OF INPUT1	NORMAL
RESPONSE TO INPUT1	DISPLAY SUBSEQUENT TEXT(S)
SERIES OF CHARACTERS IN INPUT2	"*"
STATUS OF INPUT2	ERROR
RESPONSE TO INPUT2	TRANSIT TO CURRENT PAGE

FIG.23A

SESSION ID	QEZ15434
TIME FOR TERMINATING SESSION	2000/01/15 17:35:21
TYPE OF INPUT VALUES	2
SERIES OF CHARACTERS IN INPUT 1	"1"
STATUS OF INPUT 1	NORMAL
RESPONSE TO INPUT 1	DISPLAY SUBSEQUENT TEXT
SERIES OF CHARACTERS IN INPUT 2	"*"
STATUS OF INPUT 2	ERROR
RESPONSE TO INPUT 2	TRANSIT TO DINING.KSP
SUBSEQUENT TEXT	This algorithm easily reach an impasse. Assume the case in which all the five philosophers simultaneously hold the forks at their right hand side. Although, they all waiting for the forks at their left hand side available, no forks at their left hand side will be available forever because none of them release the forks at their right hand. As a consequence, all of them are getting hungry a bit at a time with holding forks at their right hands. This is a typical example of so called "dead lock". In a less drastic case, there is a possibility that some philosophers become starving. For example, a philosopher, who sits the middle of two evil fellows who alternately take meals on purpose, starves because both the forks at right and left hand sides are occupied by these two fellows. This is a phenomenon so called "lock out". The problem named " a philosophers having meals " have often been discussed as a material of parallel programming because this can treat issues on collective occupationf common resources and critical section while the problem seems simple to resolve. This could be a famous problem in view of consecutive algorithm equivalent to a game so called "Eight queens".

FIG.23B

	(1) DESCRIPTION OF CRITICAL SECTION (2) DESCRIPTION OF CONCURRENT (3) TABLE OF CONTENTS
TYPE OF INPUT VALUES	4
SERIES OF CHARACTERS IN INPUT 1	"1"
STATUS OF INPUT 1	NORMAL
RESPONSE TO INPUT 1	TRANSIT TO CSECTION.KSP
SERIES OF CHARACTERS IN INPUT 2	"2"
STATUS OF INPUT 2	NORMAL
RESPONSE TO INPUT 2	TRANSIT TO CONCURRENT.KSP
SERIES OF CHARACTERS IN INPUT 3	"3"
STATUS OF INPUT 3	NORMAL
RESPONSE TO INPUT 3	TRANSIT TO INDEX.KSP
SERIES OF CHARACTERS IN INPUT 4	"*"
STATUS OF INPUT 4	ERROR
RESPONSE TO INPUT 4	TRANSIT TO DINING.KSP

30 JUN 2004

FIG.24

A problem named " philosopher s having meals " was proposed by Dijkstra as one of research task s in parallel processing. The details of which is described hereunder.

A total of five (5) philosophers sit around a round table and having meals as the occasion may demand while they involving into thoughts. Spaghetti piled on a dish put on the center of the table is served as the meal for them. In order to have the spaghetti, each of them need to hold fork in both hands because these are too entagled with one another.

A total of five (5) forks, each of which is placed just between each of the philosophers. This meant that they can not use a fork on his/her right during another philosopher who sits right hand side enjoys the meal, vice versa.

The countermeasure for such situation is to write a program(s) for make the philosophers full without starving any of them.

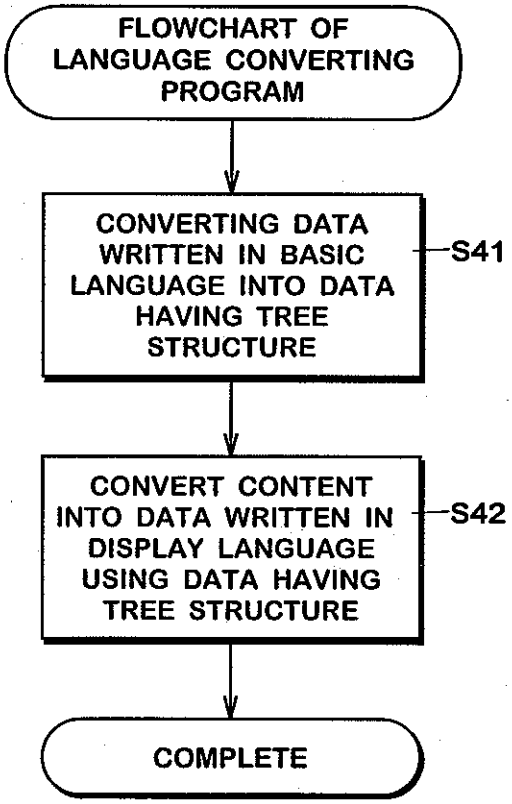
Such program can be recognized as a modeling for parallel processing, and the forks are considered as resources commonly used. In addition, a section in which these philosophers try to grab the forks ahead of others can be considered as critical section.

Suppose one of the solutions, a simple program described below may be do.

The whole concept of the program is that the philosophers wait for forks located at their right hand side, and grab forks on their left hand side once the forks at right are available. They start having meal after their both hands grab the forks.

(1)NEXT

FIG.25


FLOWCHART OF LANGUAGE CONVERTING PROGRAM
CONVERTING DATA WRITTEN IN BASIC LANGUAGE INTO DATA HAVING TREE STRUCTURE
S41
CONVERT CONTENT INTO DATA WRITTEN IN DISPLAY LANGUAGE USING DATA HAVING TREE STRUCTURE
S42
COMPLETE

30 JUN 2004

FIG.26

TAG NUMBER	
TAG TYPE	
NUMBER OF ATTRIBUTES	
ATTRIBUTE NAME-1	ATTRIBUTE VALUE-1
ATTRIBUTE NAME-2	ATTRIBUTE VALUE-2
ATTRIBUTE NAME-3	ATTRIBUTE VALUE-3
. . .	. . .
TEXT (ONLY A-TAG, P-TAG)	
SUBSEQUENT TAG NUMBER	
TAG NUMBER FOR CHILD TAG(ONLY KSP,FORM,SELECT TAG)	

30 JUN 2004

FIG.27

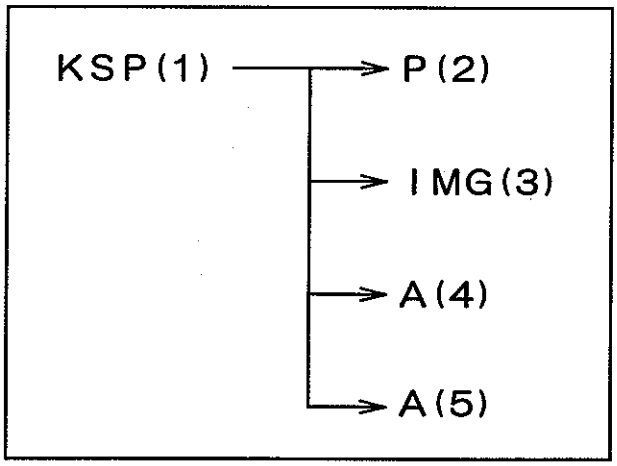


FIG.28

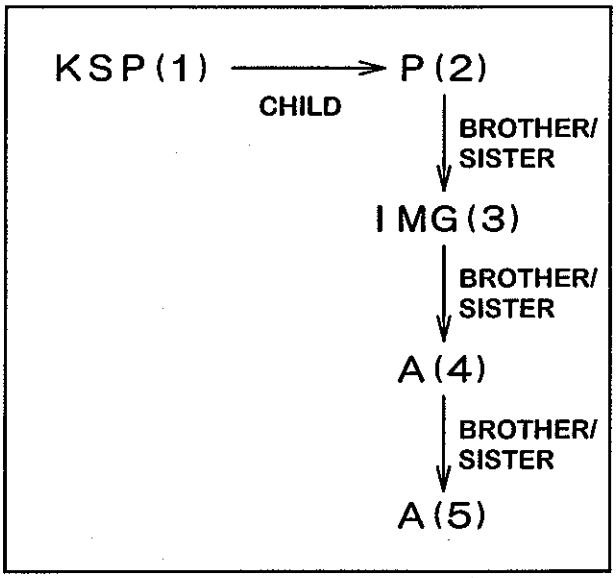

KSP(1)
CHILD
P(2)
BROTHER/
SISTER
IMG(3)
BROTHER/
SISTER
A(4)
BROTHER/
SISTER
A(5)

FIG.29

TAG NUMBER	1
TAG TYPE	KSP
NUMBER OF ATTRIBUTES	2
ATTRIBUTE NAME-1	KEY
ATTRIBUTE VALUE-1	K1
ATTRIBUTE NAME-2	NAME
ATTRIBUTE VALUE-2	KSPSAMPLE
TEXT	(EMPTY)
SUBSEQUENT TAG NUMBER	0(NO CORRESPONDING TAG)
TAG NUMBER FOR CHILD TAG	2

TAG NUMBER	2
TAG TYPE	P
NUMBER OF ATTRIBUTES	1
ATTRIBUTE NAME-1	KEY
ATTRIBUTE VALUE-1	R1
TEXT	SELECT THE MENU
SUBSEQUENT TAG NUMBER	3
TAG NUMBER FOR CHILD TAG	0(NO CORRESPONDING TAG)

TAG NUMBER	3
TAG TYPE	IMG
NUMBER OF ATTRIBUTES	2
ATTRIBUTE NAME-1	KEY
ATTRIBUTE VALUE-1	R2
ATTRIBUTE NAME-2	SRC
ATTRIBUTE VALUE-2	ICON. JPG
TEXT	(NONE)
SUBSEQUENT TAG NUMBER	4
TAG NUMBER FOR CHILD TAG	0(NO CORRESPONDING TAG)

TAG NUMBER	4
TAG TYPE	A
NUMBER OF ATTRIBUTES	2
ATTRIBUTE NAME-1	KEY
ATTRIBUTE VALUE-1	R3
ATTRIBUTE NAME-2	HREF
ATTRIBUTE VALUE-2	MAIL. KSP
TEXT	MAIL
SUBSEQUENT TAG NUMBER	5
TAG NUMBER FOR CHILD TAG	0(NO CORRESPONDING TAG)

FIG.30

TAG NUMBER	5
TAG TYPE	A
NUMBER OF ATTRIBUTES	2
ATTRIBUTE NAME-1	KEY
ATTRIBUTE VALUE-1	R4
ATTRIBUTE NAME-2	HREF
ATTRIBUTE VALUE-2	BBS. KSP
TEXT	BBS
SUBSEQUENT TAG NUMBER	0(NO CORRESPONDING TAG)
TAG NUMBER FOR CHILD TAG	0(NO CORRESPONDING TAG)

FIG.31

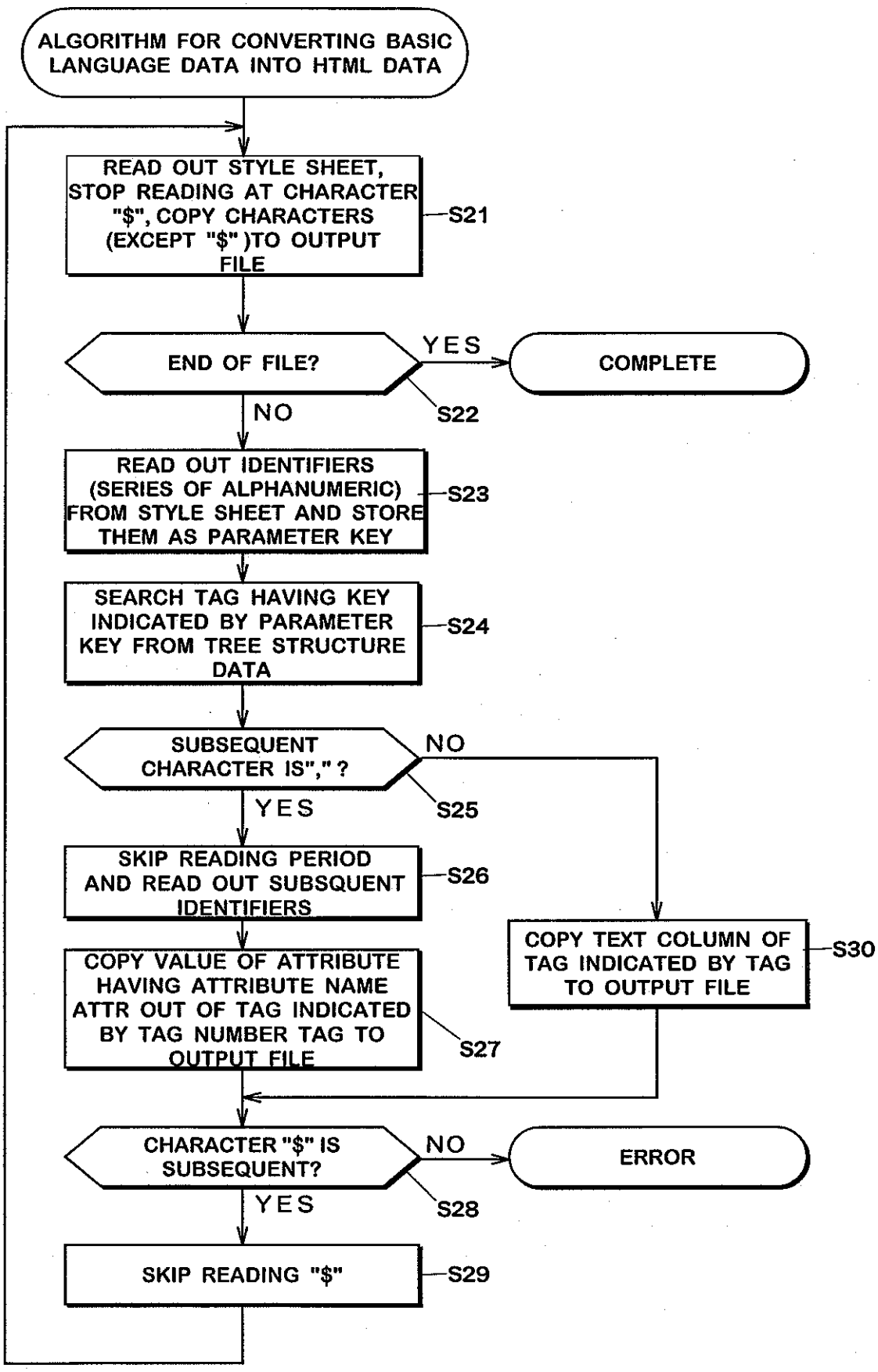

ALGORITHM FOR CONVERTING BASIC LANGUAGE DATA INTO HTML DATA
READ OUT STYLE SHEET, STOP READING AT CHARACTER "$", COPY CHARACTERS (EXCEPT "$")TO OUTPUT FILE
S21
END OF FILE?
YES
COMPLETE
S22
NO
READ OUT IDENTIFIERS (SERIES OF ALPHANUMERIC) FROM STYLE SHEET AND STORE THEM AS PARAMETER KEY
S23
SEARCH TAG HAVING KEY INDICATED BY PARAMETER KEY FROM TREE STRUCTURE DATA
S24
SUBSEQUENT CHARACTER IS"," ?
NO
S25
YES
SKIP READING PERIOD AND READ OUT SUBSQUENT IDENTIFIERS
S26
COPY TEXT COLUMN OF TAG INDICATED BY TAG TO OUTPUT FILE
S30
COPY VALUE OF ATTRIBUTE HAVING ATTRIBUTE NAME ATTR OUT OF TAG INDICATED BY TAG NUMBER TAG TO OUTPUT FILE
S27
CHARACTER "$" IS SUBSEQUENT?
NO
ERROR
S28
YES
SKIP READING "$"
S29

FIG.32

```
<HTML>
<HEAD><TITLE>KSP SAMPLE</TITLE></HEAD>
<BODY BACKGROUND=bg.gif>
SELECT THE MENU<BR>
<IMG SRC=ICOM.JPG>
<A HREF=MAIL.KSP> MAIL</A>
<A HREF=BBS.KSP> BBS</A>
</BODY>
</HTML>
```


30 JUN 2004

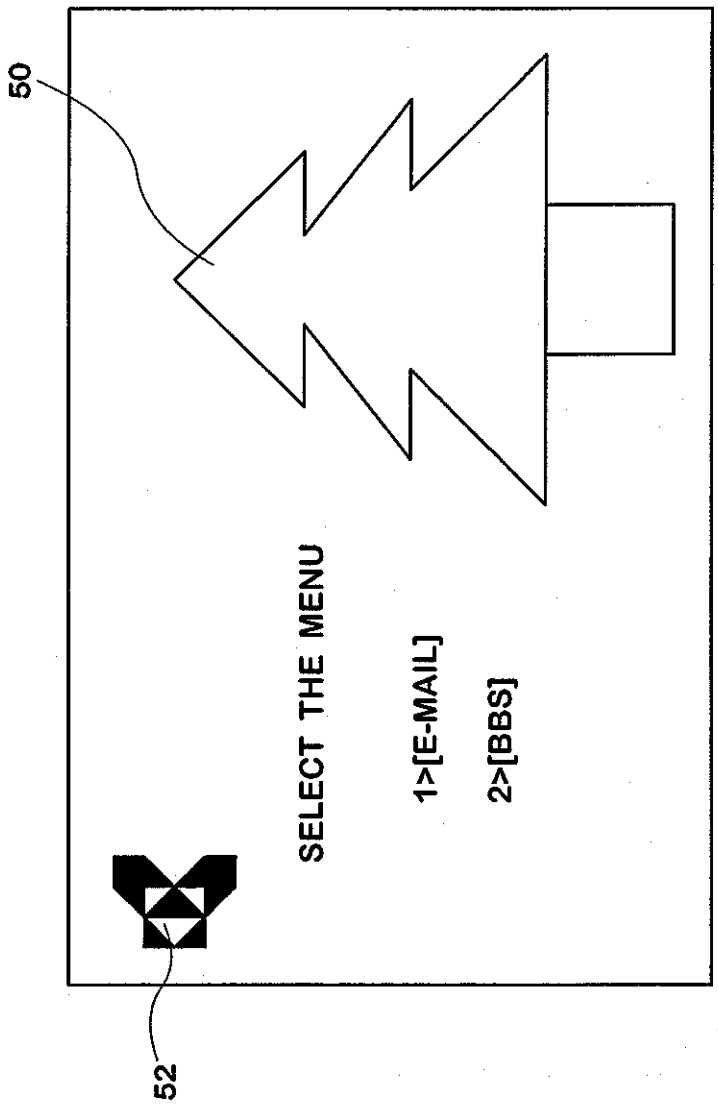


FIG.33

FIG.34A

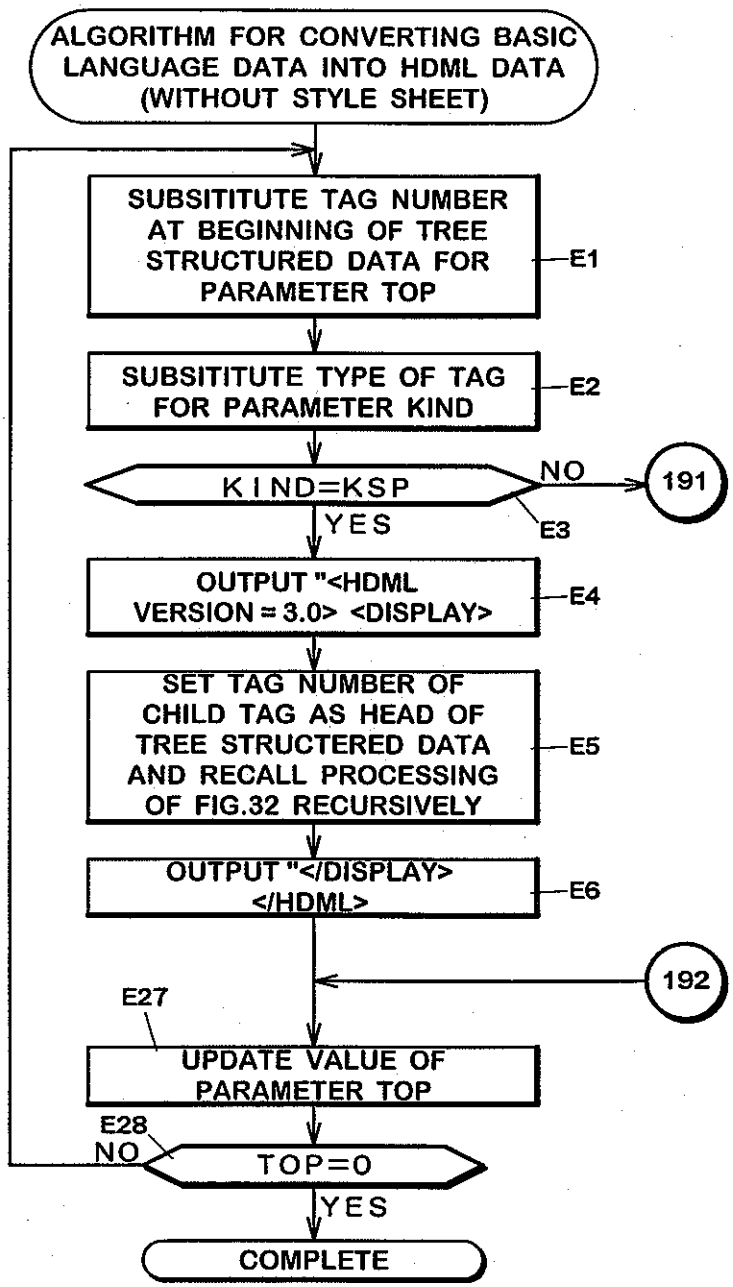

ALGORITHM FOR CONVERTING BASIC LANGUAGE DATA INTO HDML DATA (WITHOUT STYLE SHEET)
SUBSITITUTE TAG NUMBER AT BEGINNING OF TREE STRUCTURED DATA FOR PARAMETER TOP
E1
SUBSITITUTE TYPE OF TAG FOR PARAMETER KIND
E2
KIND=KSP
NO
191
E3
YES
OUTPUT "<HDML VERSION = 3.0> <DISPLAY>
E4
SET TAG NUMBER OF CHILD TAG AS HEAD OF TREE STRUCTERED DATA AND RECALL PROCESSING OF FIG.32 RECURSIVELY
E5
OUTPUT "</DISPLAY> </HDML>
E6
192
E27
UPDATE VALUE OF PARAMETER TOP
E28
NO
TOP=0
YES
COMPLETE

FIG.34B

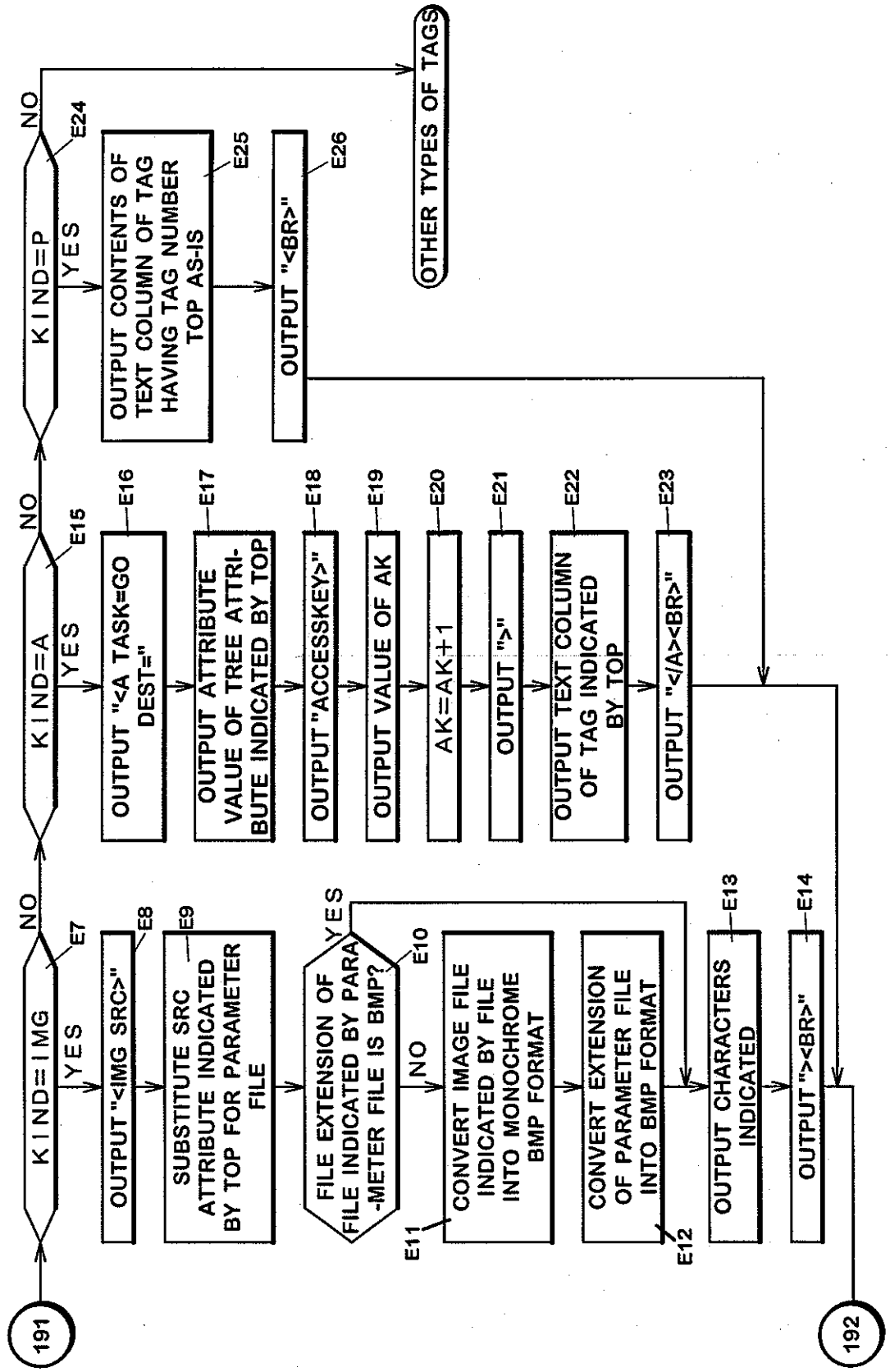

191
KIND=IMG
E7
NO
YES
OUTPUT "<IMG SRC>"
E8
SUBSTITUTE SRC ATTRIBUTE INDICATED BY TOP FOR PARAMETER FILE
E9
FILE EXTENSION OF FILE INDICATED BY PARA-METER FILE IS BMP?
E10
YES
NO
CONVERT IMAGE FILE INDICATED BY FILE INTO MONOCHROME BMP FORMAT
E11
CONVERT EXTENSION OF PARAMETER FILE INTO BMP FORMAT
E12
OUTPUT CHARACTERS INDICATED
E13
OUTPUT ">
"
E14
192
KIND=A
E15
NO
YES
OUTPUT "<A TASK=GO DEST="
E16
OUTPUT ATTRIBUTE VALUE OF TREE ATTRI-BUTE INDICATED BY TOP
E17
OUTPUT "ACCESSKEY>"
E18
OUTPUT VALUE OF AK
E19
AK=AK+1
E20
OUTPUT ">"
E21
OUTPUT TEXT COLUMN OF TAG INDICATED BY TOP
E22
OUTPUT "</A>
"
E23
KIND=P
E24
NO
YES
OUTPUT CONTENTS OF TEXT COLUMN OF TAG HAVING TAG NUMBER TOP AS-IS
E25
OUTPUT "<BR>"
E26
OTHER TYPES OF TAGS

FIG.35

```
<HTML VERSION=3.0>
<DISPLAY>
SELECT THE MENU<BR>
<IMG SRC=ICOM.BMP><BR>
<A TASK=GO DEST=MAIL.KSP ACCESSKEY=1>MAIL </A><BR>
<A TASK=GO DEST=BBS.KSP ACCESSKEY=2>BBS </A><BR>
</DISPLAY>
</HTML>
```


30 JUN 2004

FIG.36

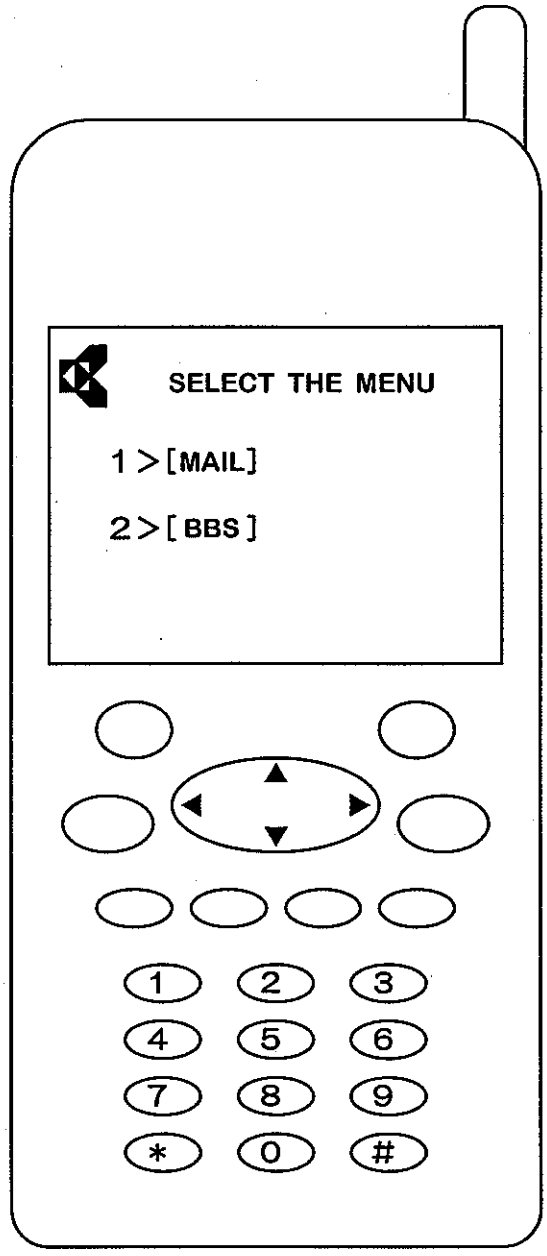
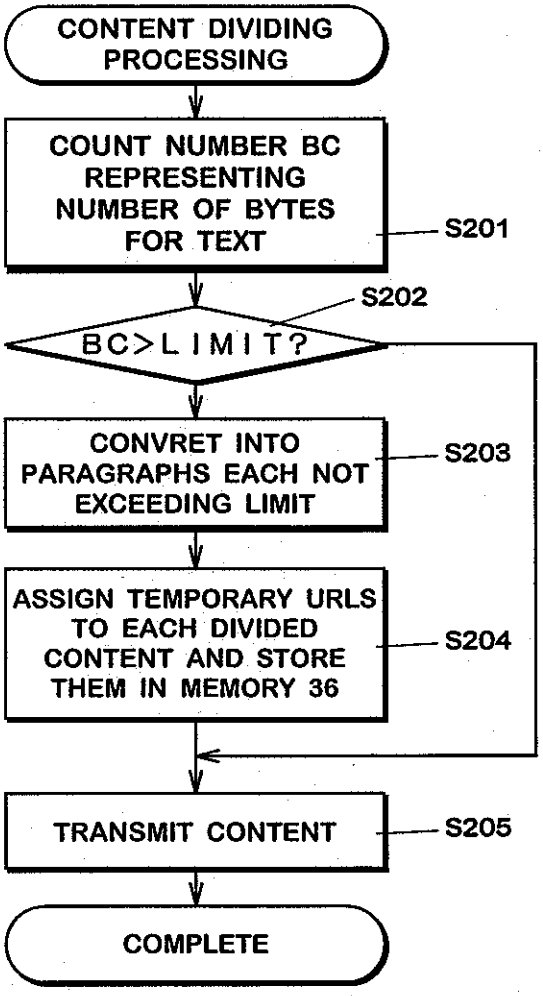


FIG.37


CONTENT DIVIDING PROCESSING
COUNT NUMBER BC REPRESENTING NUMBER OF BYTES FOR TEXT
S201
S202
BC>LIMIT?
CONVRET INTO PARAGRAPHS EACH NOT EXCEEDING LIMIT
S203
ASSIGN TEMPORARY URLS TO EACH DIVIDED CONTENT AND STORE THEM IN MEMORY 36
S204
TRANSMIT CONTENT
S205
COMPLETE

30 JUN 2004

FIG.38A

<HTML>
<HEAD><TITLE>THE PHILOSOPHERS HAVING MEALS(1)</TITLE></HEAD>
<BODY>

A problem named " philosopher s having meals " was proposed by Dijkstra as one of research task s in parallel processing. The details of which is described hereunder.

A total of five (5) philosophers sit around a round table and having meals as the occasion may demand while they involving into thoughts. Spaghetti piled on a dish put on the center of the table is served as the meal for them. In order to have the spaghetti, each of them need to hold fork in both hands because these are too entagled with one another.

A total of five (5) forks, each of which is placed just between each of the philosophers. This meant that they can not use a fork on his/her right during another philosopher who sits right hand side enjoys the meal, vice versa.

The countermeasure for such situation is to write a program(s) for make the philosophers full without starving any of them.

Such program can be recognized as a modeling for parallel processing, and the forks are considered as resources commonly used. In addition, a section in which these philosophers try to grab the forks ahead of others can be considered as critical section.

Suppose one of the solutions, a simple program described below may be do.

The whole concept of the program is that the philosophers wait for forks located at their right hand side, and grab forks on their left hand side once the forks at right are available. They start having meal after their both hands grab the forks.

FIG.38B

This algorithm easily reach an impasse. Assume the case in which all the five philosophers simultaneously hold the forks at their right hand side. Although, they all waiting for the forks at their left hand side available, no forks at their left hand side will be available forever because none of them release the forks at their right hand. As a consequence, all of them are getting hungry a bit at a time with holding forks at their right hands. This is a typical example of so called "dead lock".

In a less drastic case, there is a possibility that some philosophers become starving. For example, a philosopher, who sits the middle of two evil fellows who alternately take meals on purpose, starves because both the forks at right and left hand sides are occupied by these two fellows. This is a phenomenon so called "lock out".

The problem named " a philosophers having meals " have often been discussed as a material of parallel programming because this can treat issues on collective occupation of common resources and critical section while the problem seems simple to resolve. This could be a famous problem in view of consecutive algorithm equivalent to a game so called "Eight queens".

```
<A HREF="CLTICAL.HTML">(1) DESCRIPTION OF CRITICAL SECTION</A>
<A HREF="CONCU.HTML">(2)DESCRIPTION OF CONCURRENT</A>
<A HREF="MOKUJI.HTML>(3)TABLE OF CONTENTS</A>
</BODY>
</HTML>
```

FIG.39

DINING.HTML

```
<HTML>
<HEAD><TITLE>THE PHILOSOPHERS HAVING MEALS(1)</TITLE></HEAD>
<BODY>
```

A problem named " philosopher s having meals " was proposed by Dijkstra as one of research task s in parallel processing. The details of which is described hereunder.

A total of five (5) philosophers sit around a round table and having meals as the occasion may demand while they involving into thoughts. Spaghetti piled on a dish put on the center of the table is served as the meal for them. In order to have the spaghetti, each of them need to hold fork in both hands because these are too entagled with one another.

A total of five (5) forks, each of which is placed just between each of the philosophers. This meant that they can not use a fork on his/her right during another philosopher who sits right hand side enjoys the meal, vice versa.

The countermeasure for such situation is to write a program(s) for make the philosophers full without starving any of them.

Such program can be recognized as a modeling for parallel processing, and the forks are considered as resources commonly used. In addition, a section in which these philosophers try to grab the forks ahead of others can be considered as critical section.

Suppose one of the solutions, a simple program described below may be do.

The whole concept of the program is that the philosophers wait for forks located at their right hand side, and grab forks on their left hand side once the forks at right are available. They start having meal after their both hands grab the forks.

```
<BR>
<A HREF="DINING_2.HTML">NEXT</A>
</BODY>
</HTML>
```

30 JUN 2004

FIG.40

DINING 2.HTML

```
<HTML>
<HEAD><TITLE>THE PHILOSOPHERS HAVING MEALS(2)</TITLE></HEAD>
<BODY>
```

This algorithm easily reach an impasse. Assume the case in which all the five philosophers simultaneously hold the forks at their right hand side. Although, they all waiting for the forks at their left hand side available, no forks at their left hand side will be available forever because none of them release the forks at their right hand. As a consequence, all of them are getting hungry a bit at a time with holding forks at their right hands. This is a typical example of so called "dead lock".

In a less drastic case, there is a possibility that some philosophers become starving. For example, a philosopher, who sits the middle of two evil fellows who alternately take meals on purpose, starves because both the forks at right and left hand sides are occupied by these two fellows. This is a phenomenon so called "lock out".

The problem named " a philosophers having meals " have often been discussed as a material of parallel programming because this can treat issues on collective occupation of common resources and critical section while the problem seems simple to resolve. This could be a famous problem in view of consecutive algorithm equivalent to a game so called "Eight queens".

```
<BR>
<A HREF="CLTICAL.HTML">(1) DESCRIPTION OF CRITICAL SECTION</A>
<A HREF="CONCU.HTML">(2)DESCRIPTION OF CONCURRENT</A>
<A HREF="MOKUJI.HTML>(3)TABLE OF CONTENTS</A>
</BODY>
</HTML>
```

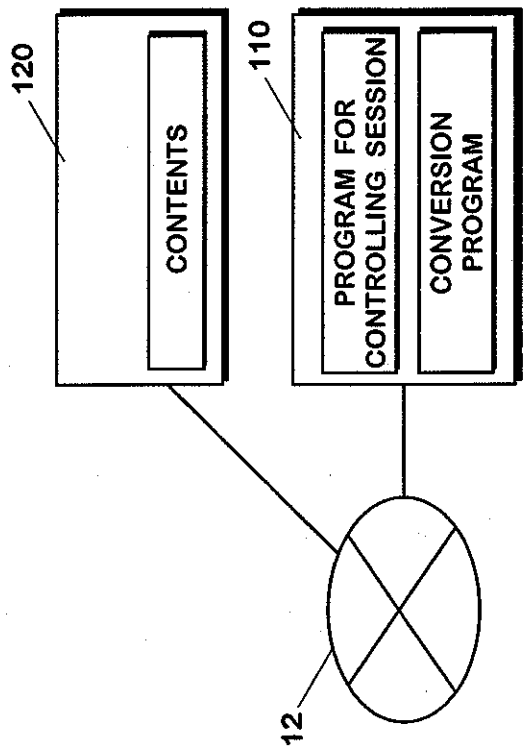

120
CONTENTS
110
PROGRAM FOR CONTROLLING SESSION
CONVERSION PROGRAM
12

FIG.41

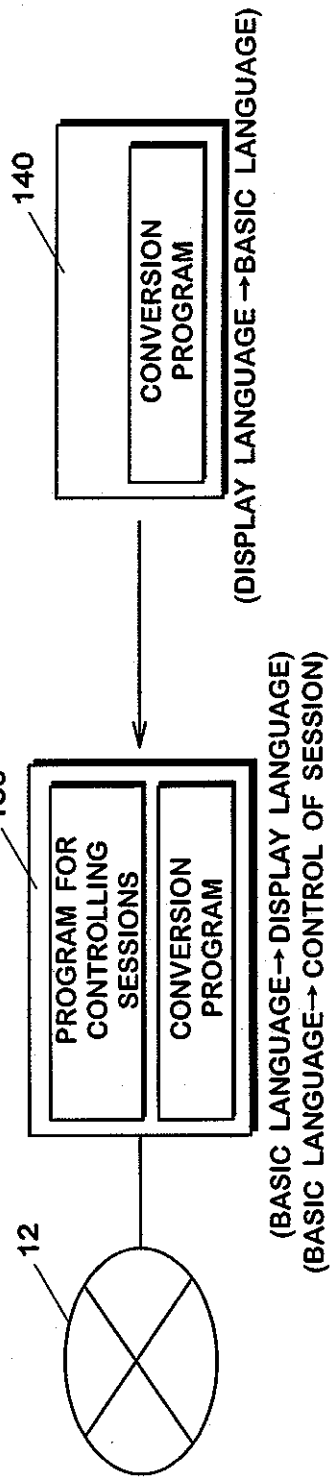

140
CONVERSION PROGRAM
(DISPLAY LANGUAGE→BASIC LANGUAGE)
130
PROGRAM FOR CONTROLLING SESSIONS
CONVERSION PROGRAM
(BASIC LANGUAGE→DISPLAY LANGUAGE)
(BASIC LANGUAGE→CONTROL OF SESSION)
12

FIG.42