COMMAND SUBSTITUTION SYSTEM

(CSS)

COMMANDS

COMMAND SUBSTITUTION SYSTEM
(CSS)

- CSS IS AN EXTENSION TO THE OS/32 COMMAND LANGUAGE.
- CSS ENABLES THE USER TO ESTABLISH FILES OF DYNAMICALLY MODIFIABLE COMMANDS.
- A CSS CAN BE CALLED FROM THE TERMINAL OR FROM OTHER CSS FILES.
- CSS's ARE EXECUTED IN A PREDEFINED WAY.

CSS PROVIDES:

- THE ABILITY TO SWITCH THE COMMAND INPUT STREAM TO A FILE OR DEVICE
- A SET OF LOGICAL OPERATORS TO CONTROL THE PRECISE SEQUENCE OF COMMANDS
- PARAMETERS THAT CAN BE PASSED TO A CSS FILE SO THAT GENERAL SEQUENCES CAN BE WRITTEN TO TAKE ON SPECIFIC MEANING WHEN THE PARAMETERS ARE SUBSTITUTED
- THE ABILITY FOR ONE CSS FILE TO CALL ANOTHER SO COMPLEX COMMAND SEQUENCES CAN BE DEVELOPED.

CSS COMMANDS:

- ALL CSS COMMANDS START WITH THE $ CHARACTER
- ALL MTM SUPPORTED COMMANDS CAN BE USED IN A CSS FILE.

A CSS FILE IS SIMPLY A SEQUENTIAL TEXT FILE. IT CAN BE A DECK OF CARDS, A MAGNETIC TAPE, OR A DISK FILE.

AN EXAMPLE OF A SIMPLE CSS FILE IS:

```
* THIS IS A EXAMPLE OF A CSS FILE
LOAD EMPRG/G,5
ALLOCATE ABCEMP.DAT,CO,40
ASSIGN 1,INPUT.DAT
ASSIGN 2,ABCEMP.DAT
ASSIGN 5,CON:
ASSIGN 3,PR:
START
$EXIT
```

THIS CSS LOADS THE PROGRAM EMPRG.TSK FROM THE GROUP ACCOUNT INTO MEMORY, ALLOCATES ALL THE FILES AND DEVICES NECESSARY, AND STARTS EXECUTION OF THE PROGRAM.

NAMING CONVENTIONS

THE FILE DESCRIPTOR FOR A CSS IS LIKE THOSE OF OTHER FILES, EXCEPT THAT IT MUST HAVE AN EXTENTION OF CSS.

EXAMPLE:

EOM.CSS
UPDATE.CSS
MT62:TERIL.CSS

EXTENTION DOES NOT HAVE TO B .CSS
EX: EOM.DAY
BUT TO RUN MUST TYPE IN FULL FILENAME E-OM.DAY

CALLING A CSS FILE:

A CSS FILE IS CALLED AND EXECUTED FROM THE TERMINAL BY SPECIFYING THE FILE DESCRIPTOR. IF THE LEADING CHARACTERS OF A CSS FILE DESCRIPTOR ARE THE SAME AS A COMMAND,MTM ASSUMES THE COMMAND.

EXAMPLE:

CLO.CSS - ASSUMES THE CLOSE COMMAND

AS3.CSS - ASSUMES THE ASSIGN COMMAND.

BY SPECIFYING A VOLUME NAME AND/OR EXTENSION, A CSS FILE THAT OTHERWISE WOULD CONFLICT WITH A MTM COMMAND CAN BE CALLED.

EXAMPLE:

```
MT61:CLOSE
MT61:CLOSE.CSS
```

USE OF PARAMETERS (ALLOWS PASSING OF DEVICES/FILES FROM ONE CSF TO ANOTHER CSF)

CSS FILENAMES CAN HAVE PARAMETERS

THE PARAMETERS CAN BE ENTERED AFTER THE CSS FILE DESCRIPTOR AND ARE SEPARATED FROM IT BY ONE CHARACTER SPACE.

EXAMPLE:

```
EOM JANUARY
RUNJOB PR:
```

IF THERE IS MORE THAN ONE PARAMETER SEPARATE EACH WITH A COMMA.

EXAMPLE:

```
RUNPROJ MT61:,MT62:,RPR:
EOD MONDAY,MT62:OUTPUT.DAT
```

IF A PARAMETER MUST PASS A COMMA PLACE A DOUBLE QUOTE " AROUND THE PARAMETER.

IN OTHER WORDS - - - -

IF A PARAMETER CONTAINS THE DOUBLE QUOTE CHARACTER ALL PARAMETERS UP TO THE NEXT DOUBLE QUOTE CHARACTER ARE PASSED AS ONE PARAMETER.

EXAMPLE:

```
RUNACCT "ACCOUNTING REPORT,APRIL"
PRTRPT "THIS REPORT IS FOR: MON,TUE AND WED"
ABC P1,"P2A,P2B"
```

@1 = P1
@2 = P2A,P2B

NULL PARAMETERS ARE PERMITTED.

EXAMPLE:

JUMP ,,C

CALLS CSS FILE JUMP.CSS ON THE DEFAULT VOLUME WITH THE THREE PARAMETERS:

PARAMETER ONE = NULL

PARAMETER TWO = NULL

PARAMETER THREE = C

@1 = NULL
@2 = NULL
@3 = C

REFERENCING PARAMETERS

WITHIN A CSS FILE, A PARAMETER IS REFERENCED BY THE USE OF THE SPECIAL SYMBOL:

@n

WHERE n IS A DECIMAL INTEGER NAME INDICATING WHICH PARAMETER IS BEING REFERENCED.

@1	FIRST PARAMETER
@5	FIFTH PARAMETER
@0	HAS SPECIAL MEANING. IT IS USED TO REFERENCE THE NAME OF THE CSS FILE IN WHICH IT IS CONTAINED.

@ 0 = FOR DEBUGGING PURPOSES

A STRAIGHT FORWARD TEXT SUBSTITUTION IS EMPLOYED.

EXAMPLE:

A CSS FILE RUNPROG CONSISTS OF:

```
LOAD @1
ASSIGN 1,@2
ASSIGN 3,@3
START @5,@4
```

IT IS CALLED AS:

```
                @1  @2   @3    @4  @5
RUNPROG TEST,CARD:,MAG1:,UPDATE,100
```

BEFORE THE CSS IS EXECUTED IT IS PREPROCESSED AND ANY REFERENCE TO A PARAMETER IS SUBSTITUTED WITH THE CORRESPONDING TEXT.

THE PREVIOUS EXAMPLE WOULD BE EXECUTED AS:

```
LOAD TEST
ASSIGN 1,CARD:
ASSIGN 3,MAG1:
START 100,UPDATE
```

PUT TWO THINGS TOGETHER AS ONE

CSS's ALLOW CONCATENATION OF PARAMETERS.

ALL OF THE FOLLOWING ARE VALID REFERENCES TO PARAMETER 5.

@5	@5ABC	@5.EXT

CONCATENATION REQUIRES CARE WITH NUMBERS.

123@5 REFERENCES PARAMETER 5

@5123 REFERENCES PARAMETER 5123

A REFERENCE TO A NON-EXISTENT PARAMETER IS NULL.

- THE MULTIPLE @ FACILITY ENABLES A CSS FILE TO ACCESS PARAMETERS OF HIGHER LEVEL FILES.
- THE MAXIMUM DEPTH IS SPECIFIED AT SYSTEM GENERATION TIME.
- @@2 IN A CSS FILE REFERS TO THE SECOND PARAMETER OF THE CALLING FILE.

WHAT DOES @@3 REFERENCE?

WHAT DOES @@@1 REFERENCE?

GIVEN A CSS CALL:

YEAR ABC,XYZ,PDQ

WHAT IS @1? ABC

WHAT IS @2? XYZ

WHAT IS @3? PDQ

WITHIN YEAR.CSS THERE IS A REFERENCE TO DAY

DAY MON,TUE,WED,THUR

WITHIN DAY.CSS WHAT IS

@1? MON

@2? TUE

@3? WED

@@1? ABC

@@2? XYZ

GIVEN THE CSS CALL:

YEAR ABC,XYZ,PDQ

CALLING:

DAY MON,TUE,WED,THUR

AND WITHIN DAY.CSS THERE IS A REFERENCE TO MONTH:

MONTH DSC1:,PR:

WITHIN MONTH.CSS WHAT IS:

@1? DSC1

2 LEVELS BEFORE
@@4? THUR

3 LEVELS BEFORE
@@@2? XYZ

YEAR.CSS

```
@1 = ABC
@2 = XYZ
@3 = PDQ
```

YEAR.CSS

```
.
.
.
DAY MON,TUE,WED,THUR
.
.
.
$EXIT
```

DAY.CSS

```
@1 = MON
@2 = TUE
@3 = WED
@4 = THUR
```

DAY.CSS

```
.
.
.
MONTH DSC1:,PR:
.
.
.
$EXIT
```

MONTH.CSS

```
@1 = DSC1:
@2 = PR:
```

MONTH.CSS

```
.
.
.
$EXIT
```

| SEXIT |

5.5.5 SEXIT Command

The SEXIT command terminates a CSS procedure. Control is returned to the calling CSS procedure or the terminal if the CSS procedure was called from the terminal. All commands on the lines after the SEXIT command are ignored.

Format:

SEXIT

CSS MUST END WITH *EXIT

5.5.2 $CLEAR Command

The $CLEAR command terminates a CSS stream, closes all CSS files, and deactivates CSS.

Format:

$CLEAR

Functional Details:

The $CLEAR command can be entered in command mode, task loaded mode, and task executing mode.

$COPY and $NOCOPY

CAN USE FOR DEBUGGING

5.5.4 $COPY and $NOCOPY Commands

The $COPY and $NOCOPY commands control the listing of CSS commands on the terminal or log device (if from batch). $COPY initiates the listing and all subsequent commands are copied to the terminal before being executed. The $NOCOPY command deactivates the listing, but is itself listed. The $COPY command is an aid in debugging CSS job streams.

Format:

$COPY

$NOCOPY

CINDY.CSS

DAT

DAU

$EXIT

* $COPY
* CINDY
- DAT
- ~
- DAU
~~~
$ EXIT
$ NOCOPY

5-10
~~~

| $WRITE |

5.5.15 $WRITE Command

The $WRITE command writes a message to the terminal or log device for both interactive and batch jobs.

Format:

$WRITE text [;]

Functional Details:

The message is output to the terminal or log device. It begins with the first nonblank character after $WRITE and ends with a semicolon or carriage return. The semicolon is not printed.

5.5.14 $WAIT Command

The $WAIT command suspends execution of a CSS for a specified period of time.

Format:

$WAIT [{n}]
      [{1}]

NUMBER = TIME IS IN SECONDS

Functional Details:

The $WAIT command will only function from a CSS routine.

When the $WAIT command is entered and the user does not want to wait the specified time, a $CONTINUE command can be entered.

5.5.10 $PAUSE Command

The $PAUSE command suspends execution of a CSS procedure.

Format:

$PAUSE

Functional Details:

When $PAUSE is entered, the CSS procedure remains suspended until the $CONTINUE command is entered or the $CLEAR command is entered to terminate a procedure suspended by a $PAUSE.

5.5.3 $CONTINUE Command

The $CONTINUE command resumes execution of a CSS procedure suspended by a $PAUSE or $WAIT command.

Format:

$CONTINUE

5.5.8 $JOB and $TERMJOB Commands

The $JOB and $TERMJOB commands set the boundaries of a CSS job. The $JOB command indicates the start, and the $TERMJOB command indicates the end of a CSS job that contains all the user CSS commands and tasks.

Format:

```
$JOB
     [CPUTIME=maxtime]
     [,classid=iocount1][,...,classid=iocount32]
      .
      .
      .
$TERMJOB
```

Functional Details:

The $JOB and $TERMJOB commands are not necessary in a CSS procedure. However, they help prevent errors in one CSS job from affecting other CSS jobs. If a CSS job contains an error, the statements remaining in that job are skipped until a $TERMJOB command is found. The next command executed is the first command found after a $TERMJOB command. If the next command is a $JOB command signifying the start of a new CSS job, it could be skipped because the system is looking for a $TERMJOB that signifies the end of the CSS job containing the error.

The CSS job containing an error is aborted, and the end of task code is 255. The $JOB command resets the end of task code to 0 for the next CSS job.

Interactive jobs have no default limits established at sysgen time. However, the user can specify CPU time and I/O transfer limits for a particular job through the $JOB command.

Any limits in the $JOB command found in a batch stream are ignored if limits were already specified in the SIGNON command.

| $SKIP |

5.5.13 $SKIP Command

The $SKIP command is used between the $JOB and $TERMJOB commands. The $SKIP command indicates that subsequent commands are to be skipped until a $TERMJOB command is found. The end of task code is set to 255.

Format:

$SKIP

#JOB
LOAD PROG
TASK PROG
ST
#IF → DOES THE FILE ACCT.RPT EXIST?
-YES
LOAD ANPROG
TASK ANPROG
AS .1, ACCT.RPT
ST
-NO
#SKIP ← IF AFTER #IF AND ANSWER IS NO
#SKIP TO #TERM
~~#TERM~~
PST
PAU
#TERM

5-20

5.5.1 SBUILD and SENDB Commands

The SBUILD command causes succeeding lines to be copied to a specified file up to, but excluding, the corresponding SENDB command. Before each line is copied, parameter substitution is performed.

Format:

```
SBUILD{fd} [,APPEND]
      {lu}
   .
   .
   .
SENDB
```

Functional Details:

The SBUILD command must be the last command on its input line. Any further information on the line is treated as a comment and is not copied to the file.

The SENDB command must be the first command in the command line, but it need not start in column 1. Other commands can follow SENDB on the command line, but nesting of SBUILD and SENDB is not permitted.

| SET CODE |

5.5.12 SET CODE Command (NO DOLLAR SIGN)

The SET CODE command modifies the end of task code of the currently selected CSS task.

Format:

SET CODE n

Parameter:

n	is a decimal number from 1 through 254.

NEW STARTING WITH 6.1
MUST USE MTM TERMINAL NOT CONSOLE

VARIABLES

THERE ARE TWO TYPES OF PSEUDO DEVICE VARIABLES:

- GLOBAL
- LOCAL

THE MAXIMUM NUMBER OF VARIABLES THAT CAN BE DEFINED IS ESTABLISHED AT SYSTEM GENERATION TIME.

1st CHARACTER ALPHA.
8 CHARACTER TOTAL
ALPHA/NUMERIC

NAMING VARIABLES

A VARIABLE NAME CAN CONSIST OF:

- ONE THROUGH EIGHT CHARACTERS: THE FIRST MUST BE ALPHABETIC AND ALL OTHERS ALPHANUMERIC.

- AN @ SIGN WHICH MUST PRECEDE THE VARIABLE NAME.

EXAMPLE:

@CNT

@IDX10

@J

GLOBAL VARIABLES

- EXIST FROM SIGNON TO SIGNOFF.
- MAY BE FREED USING THE $FREE COMMAND.

LOCAL VARIABLES

- CAN BE USED ONLY WITHIN THE CSS LEVELS IN WHICH THEY ARE DEFINED.
- WHEN A PARTICULAR CSS LEVEL IS EXITED, ALL LOCAL VARIABLES DEFINED WITHIN IT ARE FREED.

DEFINING VARIABLES

- ALL VARIABLES MUST BE DEFINED BY NAME USING THE $GLOBAL AND $LOCAL COMMANDS.
- TO SET A VARIABLE TO A SPECIFIC VALUE, USE THE $SET COMMAND.

5.5.7 $GLOBAL Command

The $GLOBAL command names a global variable and specifies the maximum length of the variable to which it can be set by the $SET command.

Format:

$GLOBAL varname [({length / 8})][,...,varname[({length / 8})]]

Parameters:

varname	is a 1- to 8-character name (the first character is alpabetic) preceded by the @ sign, identifying a global variable.
length	is a decimal number from 4 through 32 specifying the length of the variable defined by the $SET command. If this parameter is omitted, the default is 8.

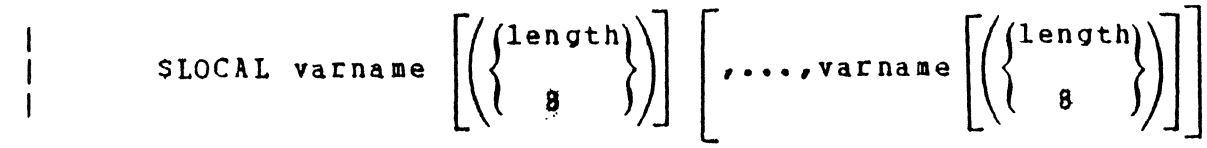
Example:

```
$GLOBAL @A(6)
```

5.5.9 SLOCAL Command

The SLOCAL command names a local variable and specifies the maximum length variable to which it can be set by the $SET command.

Format:



Parameters:

varname	is a 1- to 8-character name (the first character is alphabetic) preceded by the @ sign, identifying a local variable.
length	is a decimal number from 4 through 32 specifying the length of the variable defined by the $SET command. If this parameter is omitted, the default is 8.

Example:

```
SLOCAL @A(4)
```

| $SET |

5.5.11 $SET Command

The $SET command establishes the value of a named pseudo device variable.

Format:

```
$SET varname=e
```

Functional Details:

Expressions for this command are concatenations of variables, parameters, and character strings. No operators are allowed in an expression. If a character string is included in an expression, it must be enclosed between apostrophes ('). If an apostrophe is part of the character string, it must be represented as two apostrophes ('').

The initial value of the variable is blanks. This allows the $IFNULL and $IFNNULL commands to test for a null or not null value.

Examples:

```
$SET @A = @A1@A2

$SET @A = @A1'.MAC'

$SET @A = @1

$SET @A = 'A''B'
```

5-18

| $FREE |

5.5.6 $FREE Command

The $FREE command frees one or more pseudo variables.

Format:

$FREE varname$_1$ [,...,varname$_n$]

Parameters:

varname	is a 1- to 8-character name specifying the variable whose name and value are to be freed.

Example:

```
$FREE @A
```

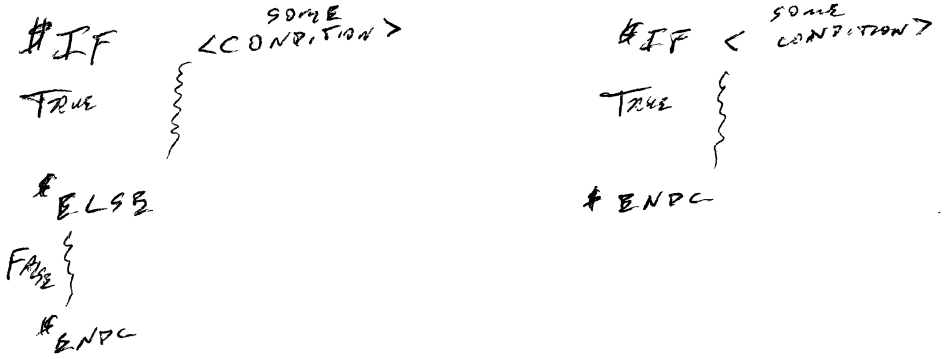
RESERVED VARIABLES

- VARIABLE NAMES STARTING WITH THE CHARACTER STRING @SYS ARE RESERVED FOR SYSTEM USE.
- A USER HAS READ AND WRITE ACCESS TO @SYS VARIABLES.
- A USER CANNOT DEFINE VARIABLES STARTING WITH @SYS.
- @SYSCODE = CONTAINS THE VALUE OF THE LAST END OF TASK CODE FOR A PARTICULAR SESSION.

CSS ALLOWS LOGICAL IF COMMANDS.

- END OF TASK CODE TESTING
- FILE EXISTENCE TESTING
- VOLUME EXISTENCE TESTING
- FILE EXTENSION EXISTENCE TESTING
- PARAMETER EXISTENCE TESTING
- COMPARING TWO ARGUMENT TESTING

EACH LOGICAL IF COMMAND ESTABLISHES A CONDITION THAT IS TESTED BY THE CSS PROCESSOR. IF THE RESULT OF THIS TEST IS TRUE, COMMANDS UP TO A CORRESPONDING $ELSE OR $ENDC COMMAND ARE EXECUTED. IF THE RESULT OF THIS TEST IS FALSE THE COMMANDS AFTER THE $ELSE ARE EXECUTED UP TO THE CORRESPONDING $ENDC, OR THE CORRESPONDING $ENDC IS EXECUTED.



#IF MUST HAVE #ENDC

```
      .
      .
      .
$IF
```

```
  < STMT1 >
  < STMT2 >
      .
      .
  < STMTn >
```

THESE STATEMENTS ARE EXECUTED ONLY IF THE CONDITION IS TRUE.

```
$ELSE
```

```
  < STMT1 >
  < STMT2 >
      .
      .
  < STMTn >
```

THESE STATEMENTS ARE EXECUTED ONLY IS THE CONDITION IS FALSE.

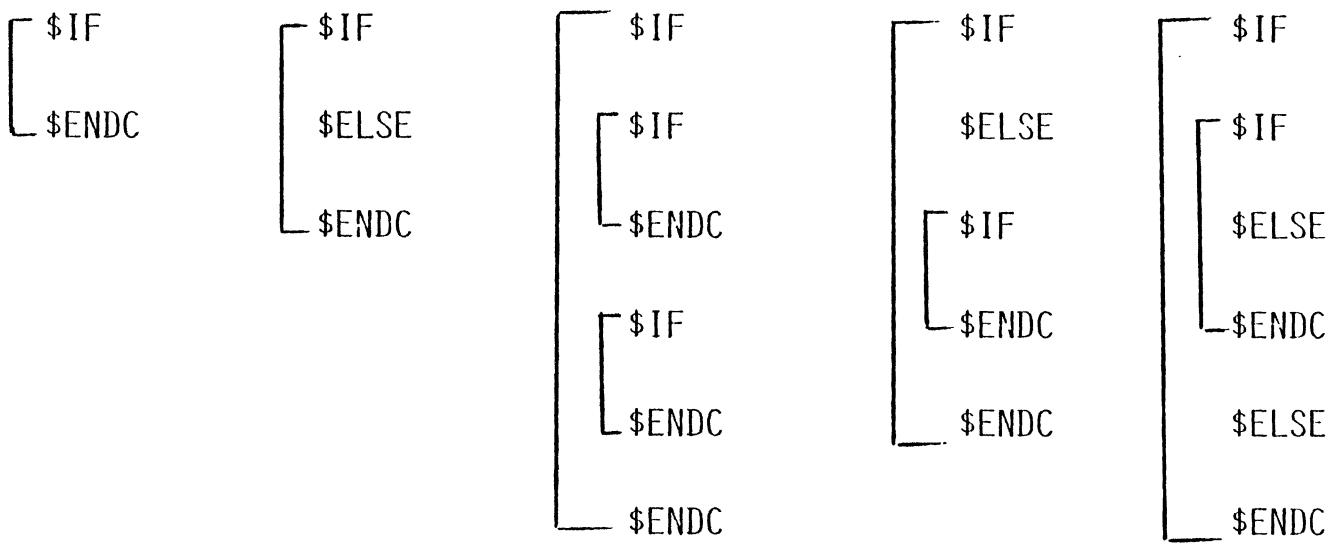
```
$ENDC
```

- THE $ENDC COMMAND DELIMITS THE RANGE OF A LOGICAL IF.

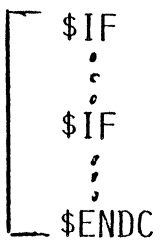
- NESTING IS PERMITTED.

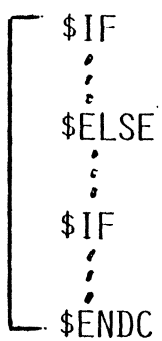
  RULE: IF YOU HAVE AN $IF STATEMENT YOU MUST ALWAYS HAVE AN $ENDC STATEMENT. THE TOTAL NUMBER OF $IF STATEMENTS IN A CSS MUST EQUAL THE TOTAL NUMBER OF $ENDC STATEMENTS.

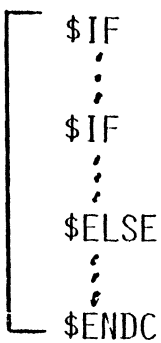
VALID EXAMPLES OF LOGICAL IF COMMANDS:

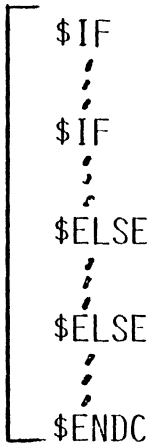


INVALID EXAMPLES OF LOGICAL IF COMMANDS:









END OF TASK CODE TESTING

THE END OF TASK CODE IS A HALFWORD QUANTITY MAINTAINED FOR EACH USER BY THE SYSTEM.

IT IS SET OR RESET IN ANY OF THE FOLLOWING WAYS:

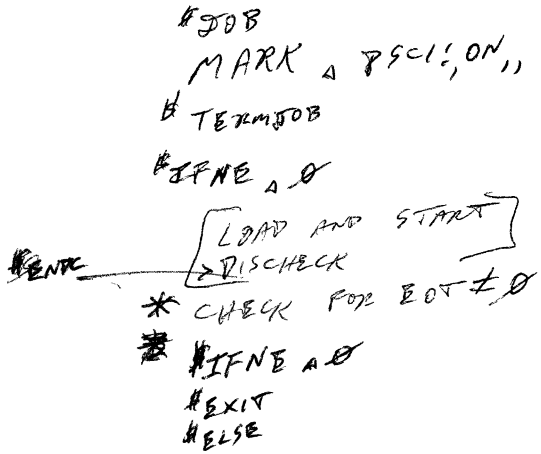
SET CODE n	This command, which can be included in a CSS file or entered at the terminal, sets the end of task code to n.
$JOB	As part of its start job function, this command resets the end of task code for the current CSS task to 0.
Command error	A command error causes the CSS mechanism to skip to $TERMJOB assuming that a $JOB was executed. (If no $JOB was executed, CSS terminates.) To indicate that the skip took place, the end of task code is set to 255.
$SKIP	This command has the same effect as a command error.
EOT (SVC 3,n)	When any task terminates by executing the end of task program command (SVC 3,n), the end of task code for that task is set to n.
CANCEL	When a task is cancelled, the end of task code is set to 255.

5-23

The six commands available for testing the end of task code of the currently selected CSS task are as follows:

SIFE n	Test end of task code equal to n
SIFNE n	Test end of task code not equal to n
SIFL n	Test end of task code less than n
SIFNL n	Test end of task code not less than n
SIFG n	Test end of task code greater than n
SIFNG n	Test end of task code not greater than n

In all cases, if the results of the test are "false", CSS skips commands until the corresponding $ELSE or $ENDC. If a CSS attempts to skip beyond EOF, a command error is generated.



*(ASTERIK) IN COLUMN ONE MEANS IT IS A COMMENT.

5.6.2 File Existence Testing Commands

There are two commands dealing with file existence:

$IFX fd	Test fd for existence
$IFNX fd	Test fd for nonexistence

If the result of the test is false, CSS skips to the corresponding $ELSE or $ENDC command. If a CSS attempts to skip beyond EOF, an error is generated.

```
$IFNX   C:INDY.DAT
   AL   @C:INDY.PAT
$ENDC
   or
$IFX    C:INDY.DAT
  $SET  @A=0
$ELSE
   AC   C:INDY.PAT
$ENDC
```

| $IFVOLUME |

5.9 $IFVOLUME COMMAND

The $IFVOLUME command tests for the existence of a volume name in an fd. If a volume exists, subsequent commands are executed up to the next $ELSE or $ENDC command. If the volume is omitted in the fd, subsequent commands are skipped up to the next $ELSE or $ENDC command.

Format:

$IFVOLUME fd

Parameter:

fd	is the file descriptor tested to determine if a volume name is included.

$IFV @1
LOAD @1
$ELSE
LOAD MT62:@1
$ENDC

5.8 $IFEXTENSION COMMAND

The $IFEXTENSION command is used to test for the existence of an extension for a given fd. If the extension exists, subsequent commands are executed up to the next $ELSE or $ENDC command. If an extension does not exist, subsequent commands are skipped up to the next $ELSE or $ENDC command.

Format:

$IFEXTENSION fd

Parameter:

fd	is the file descriptor to be tested to determine if an extension is included.

Functional Details:

$IFEX (with no fd) is always considered false.
$IFNEX (with no fd) is always considered true.

5.6.3 Parameter Existence Testing Commands

There are two commands dealing with the existence of parameters:

$IFNULL	∂n	Test ∂n null
$IFNNULL	∂n	Test ∂n not null

If the result of the test is false, CSS skips to the corresponding $ELSE or $ENDC command. If such skipping attempts to skip beyond EOF, a command error is given.

The use of the multiple ∂ notation to test for the existence of higher level parameters is permitted. In addition, a combination of parameters can be tested simultaneously.

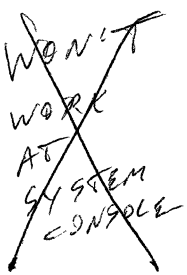
Example:

$IFNU∂1∂2∂3

In effect, this tests the logical OR of ∂1, ∂2, and ∂3 for nullity. If any of the three is present, the test result is false.

5.10 LOGICAL IF COMMANDS

The following logical IF commands are used to compare two arguments. They differ from the other logical IF commands in that they do not test specific built-in conditions but, rather, test conditions provided by the user. These commands are available only with MTM.



@1=@2

```
$IF . . . EQUAL
$IF . . . NEQUAL
$IF . . . GREATER
$IF . . . NGREATER
$IF . . . LESS
$IF . . . NLESS
```

For each of the logical commands, two arguments are compared according to the mode. There are three valid modes:

- Character
- Decimal
- Hexadecimal

For character mode, the comparison is left-to-right and is terminated on the first pair of characters that are not the same. If one string is exhausted before the other, the short string is less than the long string. If both strings are exhausted at the same time, they are equal. For character mode, the arguments can be enclosed in double quotes if they contain blanks. The quotes are not included in the compare.

For decimal and hexadecimal mode, the comparison is performed by comparing the binary value of the numbers.

If after comparing the arguments for each of the commands, the condition is determined to be true, subsequent commands are executed up to the corresponding $ELSE and $ENDC. If the condition is false, commands are skipped up to the corresponding $ELSE or $ENDC.

ABCDEF
ABCFGH

'ABC' = @A
'ABCD' = @B
$IFC @A=@B

5.10.1 SIF...EQUAL, SIF...NEQUAL Commands

The SIF...EQUAL command is used to determine if two arguments are equal, while the SIF...NEQUAL is used to determine if two arguments are not equal.

Format:

$$\text{SIF} \left\{ \begin{array}{c} \underline{\text{C}}\text{HARACTER} \\ \underline{\text{D}}\text{ECIMAL} \\ \underline{\text{H}}\text{EXADECIMAL} \end{array} \right\} \begin{array}{c} @1 = @2 \\ \text{arg}_1 \ \underline{\text{EQ}}\text{UAL} \ \text{arg}_2 \end{array}$$

$$\text{SIF} \left\{ \begin{array}{c} \underline{\text{C}}\text{HARACTER} \\ \underline{\text{D}}\text{ECIMAL} \\ \underline{\text{H}}\text{EXADECIMAL} \end{array} \right\} \begin{array}{c} @1 = @2 \\ \text{arg}_1 \ \underline{\text{NE}}\text{QUAL} \ \text{arg}_2 \end{array}$$

5.10.2 SIF...GREATER, SIF...NGREATER Commands

The SIF...GREATER command is used to determine if arg1 is greater than arg2. The SIF...NGREATER command is used to determine if arg1 is not greater than arg2.

Format:

$$\text{SIF} \left\{ \begin{matrix} \underline{\text{CH}}\text{ARACTER} \\ \underline{\text{D}}\text{ECIMAL} \\ \underline{\text{H}}\text{EXADECIMAL} \end{matrix} \right\} \text{arg}_1 \ \underline{\text{GR}}\text{EATER} \ \text{arg}_2$$

$$\text{SIF} \left\{ \begin{matrix} \underline{\text{CH}}\text{ARACTER} \\ \underline{\text{D}}\text{ECIMAL} \\ \underline{\text{H}}\text{EXADECIMAL} \end{matrix} \right\} \text{arg}_1 \ \underline{\text{NG}}\text{REATER} \ \text{arg}_2$$

5.10.3 $IF...LESS, $IF...NLESS Commands

The $IF...LESS command is used to determine if arg1 is less than arg2. The $IF...NLESS command is used to determine if arg1 is not less than arg2.

Format:

$$\$IF \left\{ \begin{array}{c} \underline{CH}ARACTER \\ \underline{D}ECIMAL \\ \underline{H}EXADECIMAL \end{array} \right\} arg_1 \ \underline{LE}SS \ arg_2$$

$$\$IF \left\{ \begin{array}{c} \underline{CH}ARACTER \\ \underline{D}ECIMAL \\ \underline{H}EXADECIMAL \end{array} \right\} arg_1 \ \underline{NLE}SS \ arg_2$$

5.7 $GOTO AND $LABEL COMMANDS

The $GOTO command is used to skip forward within a CSS procedure. The $LABEL is used to define the object of a $GOTO.

Format:

$GOTO label

$LABEL label

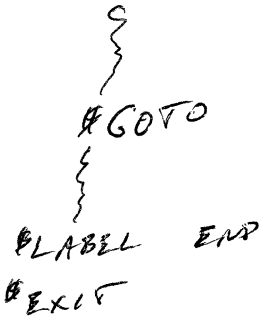
Parameters:

Label	is from one to eight alphanumeric characters, the first of which must be alphabetic.

Functional Details:

The $GOTO command causes all subsequent commands to be ignored until a $LABEL command with the same label as the $GOTO command is encountered. At that point, command execution resumes.

The $GOTO cannot branch into a logical IF command range but can branch out from one.



THE FOLLOWING IS A VALID EXAMPLE OF $GOTO AND $LABEL:

```
$IF         < CONDITION >

$GOTO         OUTIF
  :
$ENDC

$IF         < CONDITION >
  :
$ENDC

$LABEL        OUTIF
```

THE FOLLOWING IS AN INVALID EXAMPLE OF $GOTO AND $LABEL:

```
$IF         <CONDITION>

$GOTO         OUTIF
  :
  :
$ENDC

$IF         <CONDITION>

$LABEL        OUTIF
```

THE $LABEL OCCURS WITHIN AN IF BLOCK (THE SECOND IF CONDITION) THAT WAS NOT ACTIVE WHEN $GOTO WAS EXECUTED.

HOW TO CREATE AND RUN A CSS FILE

WRITE A CSS THAT EXECUTES THE FOLLOWING STEPS:

1. DISPLAY ALL USERS CURRENTLY SIGNED ON TO THE SYSTEM. DAY

2. DISPLAY THE CURRENT TIME OF DAY. DAT

3. CREATE A FILE NAMED TMP.DAT THAT CONTAINS THREE RECORDS OF TEXT USING THE $BUILD. . . . $ENDB COMMANDS.

MTM <CR>
* SIGNON
* EDIT
>AT

HOW TO CREATE AND RUN A CSS FILE

ANSWERS

```
*EDIT   CR
 OS/32 EDIT
>AP
 1  D U  <CR>
 2  D T  <CR>
 3  $BUILD TMP.DAT  <CR>
 4  THE SKY IS BLUE.  <CR>
 5  THIS IS RECORD TWO.  <CR>
 6  ISN'T THIS FUN!  <CR>
 7  $ENDB  <CR>
 8  $EXIT  <CR>
 9 <CR>  — TO GET OUT OF APPEND MODE
>SA TEST.CSS  <CR>
>END  <CR>
*
```

TO ENVOKE THE CSS TYPE:

```
* TEST   CR
```