ElPrimo

Language Reference

Version 3

document version: 3

SET_IMGDIR

Syntax: SET_IMGDIR <diretory>

Description: This is a preprocessor directive. Set image directory used by OUT_IMAGE command.

Example:

```
SET_IMGDIR C:\stimoli
```

SET_DB

Syntax: SET_DB <diretory>

Description: This is a preprocessor directive. Set the target file for the WRITE_DB command. File is written in Unix text format.

Example:

```
SET_DB C:\log\db.txt
```

Note: *This file, if present, is reed before insert the subject info details for check the data insertion. In order to avoid problems, the text format MUST BE:*

- *one line for each subject*
- *";" separeted fields*
- *the first field containing the subject index*
- *the second conataining the trial index:*

This is a compact notation:

<nsubject>;<ntrial>;.....;.....;.....

SET_LOG

Syntax: SET_LOG <diretory>

Description: This is a preprocessor directive. Set the target file for the WRITE_LOG command. File is written in Unix text format.

Example:

```
SET_LOG C:\log\log.txt
```

SET_ZOOM_PARAMETER

Syntax: SET_ZOOM_PARAMETER <percentage>

Description: This is a preprocessor directive. Set the initial zoom value in the subject info insertion window. <percentage> MUST BE a positive integer.

Example:

```
SET_ZOOM_PARAMETER 80
```

FULLSCREEN

Syntax:

FULLSCREEN

(KEY <letter>;<label>)*

ENDFULLSCREEN

Description: This is a preprocessor directive. Set the key-label mapping used in the fullscreen mode.

Example:

```
FULLSCREEN
    KEY o;OK
    KEY a;ALTRO
ENDFULLSCREEN
```

MACRO

Syntax:

MACRO <name> <$label_1$>;<$label_2$>;..;<$label_n$>

(<cmd> ;)*

ENDMACRO

Description: This is a preprocessor directive. Set a macro code. Each label will be istantiate with the corresponding argument in the calling EXPAND_MACRO statement.

Example:

```
MACRO t_stimolo label;name;img
  STIMOLO [label]
    OUT_IMAGE [img]
    CHOICE OK;ALTRO
      CASE OK
        WRITE_LOG [name]::[label]::OK
      CASE ALTRO
        WRITE_LOG [name]::[label]::ALTRO
    ENDCHOICE
  ENDSTIMOLO
ENDMACRO
```

EXPAND_MACRO

Syntax: EXPAND_MACRO <name> <$value_1$>;<$value_2$>;..;<$value_n$>

Description: This is a preprocessor directive. Expand a defined macro. Label defined in the macro are istantiate with the corresponding values.

Example:

```
EXPAND_MACRO t_stimolo T1;stimolo_T1;Diapositiva3.JPG
```

STIMOLO

Syntax:

STIMOLO <name>

(<cmd>);

ENDSTIMOLO

Description: Define a stimolo procedure which can be execute by EXEC_STIMOLO statement.

Example:

```
STIMOLO stimolo1
    OUT_IMAGE img_stimolo1.jpg
    CHOICE OK;ALTRO
        CASE OK
            WRITE_LOG T1::stimolo1::OK
        CASE ALTRO
            WRITE_LOG T1::stimolo1::ALTRO
    ENDCHOICE
ENDSTIMOLO
```

EXEC_STIMOLO

Syntax: EXEC_STIMOLO <name>

Description: Execute stimolo

Example:

```
EXEC_STIMOLO t_stimolo
```

GEN_RANDOM_SEQUENCE

Syntax: GEN_RANDOM_SEQUENCE <arrayname>

Description: Generate a randome sequence of stimolo names. Only stimolo with numerical names are consired.

Example:

```
GENERATE_RANDOM_SEQUENCE t_stimolo
```

WRITE_LOG

Syntax: WRITE_LOG <expression>

Description: Write a line into log file. <expression> can contains variable, array or hash. File is written in Unix text format.

Example:

```
WRITE_LOG SOGGETTO:[SOGGETTO]
```

WRITE_DB

Syntax: WRITE_DB <expression>

Description: Write a line into database file. <expression> can contains variable, array or hash.File is written in Unix text format.

Example:

```
WRITE_DB [SOGGETTO];[TENTATIVO];[CONTROLLO];<#RISULTATI>;<RISULTATI>
```

CHOICE

Syntax:

```
CHOICE label1;label2;...;labeln
        CASE  label1
                (<cmds>)*
        CASE  label2
                (<cmds>)*
        ...
        CASE  labeln
                (<cmds>)*
CHOICE
```

Description: Wait for a choice. The selection MUST BE one of the defined labels. After selection only the corresponding block of command is executed. The CASE statement COULD BE omit.

Example:

```
CHOICE OK;ALTRO;NIENTE
    CASE OK
        WRITE_LOG [name]::[label]::OK
    CASE ALTRO
        WRITE_LOG [name]::[label]::ALTRO
ENDCHOICE
```

FOREACH

Syntax: FOREACH <arrayname>;<label>

Description: Repeat block foreach element in the array; for each step, the current value of the sequence is set in the variable <label>.

Example:

```
FOREACH SEQUENZA;A
        EXEC_STIMOLO [A]
ENDFOREACH
```

SET_ZOOM

Syntax: SET_ZOOM <percentage>

Description: Set image magnitude.

Example:

SET_ZOOM 120

OUT_IMAGE

Syntax: OUT_IMAGE <image>

Description: Display image.

Example:

OUT_IMAGE Diapositiva1.JPG

SET

Syntax: SET <name>;<value>

Description: Set a variable named <name> with value <value>.

Example:

SET A;1

GET_DATE

Syntax: GET_DATE <name>

Description: Set a variable named <name> with the actual date.

Example:

GET_DATE TNOW

START_TIMER

Syntax: START_TIMER

Description: Reset timer.

Example:

START_TIMER

GET_TIMER

Syntax: GET_TIMER <name>

Description: Set a variable named <name> with the actual time timer.

Example:

GET_TIMER TEXPERIMENT

INPUTARRAY

Syntax:

INPUTARRAY

(INPUT <name>;<message>)*

ENDINPUTARRAY

Description: Display a window for input values. Each pair <name>;<message> corresponds to a row. Each row display <message> and a textbox for insert a value. Values inserted are stored in the corresponding variable. If there is only one INPUT statements, INPUTARRAY and ENDINPUTARRAY statements CAN BE omitted.

Example:

```
INPUTARRAY
    INPUT VAR1;Variable1
    INPUT VAR2;Variable2
ENDINPUTARRAY
```

Evaluate <Expression>

The following token are substitute during the expression evaluation:

[name]	It is a variable. It will be substitute with the value of the variable <name>
{name}	It is an array. It will be substitute with all elements in the array separated by ";"
<name>	It is an hashmap. It will be substitute with all elements of the hashmap separated by ";". The printing order depends of the key type: before came all element with alphaumeric key in lexicographically order, then all the element wth the numeric key in ascendent order.
<#name>	It is an hashmap operator. It will be substitute with the number of the hashmap elements.
<!name>	It is an hashmap operator. It will be substitute with all key in the hashmap separated by ";". The printing order depends of the key type: before came all alphaumeric keys in lexicographically order, then all numeric keys in ascendent order.

Preprocessor directives

Preprocessing directives are statements which will be executed in the first step of parsing. It consists of the following statements:

- SET_IMGDIR, SET_DB, SET_LOG, SET_ZOOM_PARAMETER
- MACRO
- FULLSCREEN

The parse process involves into three step:

1. Preprocessing:
    - Comment and empty lines are removed. Preprocessing directive are executed and removed from the code.
2. Expanding macro:
    - Macro are expanded. At the end of this phase, a clean interpretable code has been build.
3. Build lookup tables
    - Image path are verified in order to discover bloken references. Entry-point of the STIMOLO statement are memorized into a lookup table.