RoboSlang Script Command Guide :

```
Valid for : Robot Simulator v1.0
Valid at  : Thu 16/Mar/2005

Number of Commands Available = 13
```

HALT :

The general form of the HALT command is as follows :

```
    HALT
```

When this command is encountered, the script processing / execution is stopped and the robot`s motion is stopped.

This script command has precisely the same effect as manualy pressing the "Halt" button.

To re-start the script (from the beginning) press the "Run Script" button.

RANDOM :

The general form of the RANDOM command is as follows :

```
    RANDOM
```

If you don`t call RANDOM in your script, then the same set of random numbers will be generated for each and every run of the script, and any random number related commands (e.g. TELEPORT_RANDOM_H, Motor_Power RAND ..., etc) will produce the same results and make the robot act precisely the same way for each run of the script. This "predictable" behaviour can be desirable in many situations. e.g. robot dance routines, etc.

However, if you prefer, then you can get a completely different set of random numbers each time by inserting a RANDOM command into your script - and this will randomise the set of random numbers generated (based on the current date / time).

Normally, you would do this just once at the start of the script. i.e. insert a single RANDOM command prior to any loops, and from then on all numbers will be randomised and different for each run of the script.

MOTOR_POWER :

The general forms of the MOTOR_POWER command are as follows :

```
    MOTOR_POWER <motor> <speed>
    MOTOR_POWER <motor> RAND <lower-speed> <higher-speed>
```

where :

 - The <motor> should be "L" for the left motor, "R" for the right motor, or "LR" for both motors, and,

 - The <speed> must be a whole number in the range -10 to +10, or it can be the word "RAND" to automatically select a random speed within a desired range (which must also be supplied with the command).

Examples :

MOTOR_POWER L 10 will make the left motor go forward at the fastest speed,

MOTOR_POWER LR 0 will stop the left and right motors, and,

MOTOR_POWER L RAND 4 7 will make the left motor go at a random speed somewhere in the range 4 to 7 (inclusive).

MOTOR_STOP :

The general form of the MOTOR_STOP command is as follows :

```
    MOTOR_STOP <motor>
```

where :

 - The <motor> should be "L" for the left motor, "R" for the right motor, or "LR" for both motors.

Example :

MOTOR_STOP L will stop the left motor.

WAIT :

The general forms of the WAIT command are as follows :

```
    WAIT <milli-seconds>
    WAIT RAND <lower-milli-seconds> <higher-milli-seconds>
```

where :

 - The <milli-seconds> value must be a whole number greater than or equal to 0, or it can be the word "RAND" to automatically select a random time within a desired milli-second range (which must also be supplied with the command).

Examples :

WAIT 100 will wait for approximately 100 milli-seconds (i.e. 1/10 th of a second) before executing any more lines of script. If the Robot is currently moving or turning, then it will continue doing this for the next 1/10 th of a second, however, no further script commands will be processed for 2 seconds.

WAIT RAND 500 1500 will wait for a random time in the range 500 and 1,500 milli-seconds.

TURN :

The general forms of the TURN_LEFT or TURN_RIGHT commands are as follows :

```
    TURN_LEFT <angle>
    TURN_RIGHT <angle>
    TURN_LEFT RAND <lower-angle> <higher-angle>
    TURN_RIGHT RAND <lower-angle> <higher-angle>
```

where :

 - The <angle> represents the number of degrees to rotate, and must be a whole number greater in the range -360 to 360, or it can be the word "RAND" to automatically rotate within a desired range of degrees (which must also be supplied with the command).

Examples :

e.g. TURN_LEFT 45 will turn the robot 45 degrees to the left.

TURN_RIGHT RAND 10 50 will turn the robot right a random number of degrees in the range 10 to 50.

TELEPORT :

The general forms of the TELEPORT_RANDOM commands are follows :

```
    TELEPORT_RANDOM_H
    TELEPORT_RANDOM_V
    TELEPORT_RANDOM_HV
```

where :

 - H indicates that a new, random horizontal location will be chosen,

 - V indicates that a new, random vertical location will be chosen, and,

 - HV indicates that a new, random horizontal and vertical location will be chosen.

Example :

TELEPORT_RANDOM_V will teleport the robot to a new, random vertical location.

LOOP_START :

The general form of the LOOP_START command is as follows :

```
    LOOP_START <loop_name> <num_iterations>
```

where :

 - The <loop_name> must be the name of a loop, and there must be an END_LOOP <loop_name> somewhere else in the script, and,

- If provided (see below), then the <num_iterations> must be a whole number greater than or equal to zero.

e.g. LOOP_START Main_Loop 5 will execute all of the code between this line and the END_LOOP Main_Loop five times, and LOOP_START Main_Loop 0 will loop forever (until the script is HALTed).

If <num_iterations> is not provided then a value of 0 (i.e. loop forever) is assumed.

Here is an example of a loop which would make the robot move in an arc and stop 3 times :

```
   LOOP_START Move_in_Arc 3
      Motor_Power L 5
      Motor_Power R 1
      WAIT 500
      Motor_Power LR 0
      WAIT 500
   LOOP_END Move_in_Arc
```

LOOP_END :

The general form of the LOOP_END command is as follows :

```
    LOOP_END <loop_name>
```

where :

- The <loop_name> must be the name of a loop, and there must be an END_LOOP <loop_name> somewhere else in the script.

Example :

LOOP_END Main_Loop will mark the end of a loop that started with LOOP_START Main_Loop.

IF_SENSOR_IN_RANGE :

The general form of the IF_SENSOR_IN_RANGE command is as follows :

```
    IF_SENSOR_IN_RANGE <sensor_id> <low_value> <high_value> <script_command>
```

where :

- The <sensor_id> should be one of the following :
    * Lo for the front, outer left light sensor,
    * L for the front, inner left light sensor,
    * FC for the front-middle light sensor,
    * R for the front, inner right light sensor,
    * Ro for the front, outer right light sensor,
    * LFCR for the front, inner left, center, and right sensors,
    * LoRo for the front, outer left and right sensors,

 - The <low_value> and <high_value> must be whole numbers in the range 0 to 100, and,

 - The <script_command> must be any other valid script command.

Examples :

IF_SENSOR_IN_RANGE L 0 5 MOTOR_POWER L 5 will make left motor go forward if the front left sensor is over a very dark area of the background.

IF_SENSOR_IN_RANGE R 90 100 CALL_SUBROUTINE Right-Sensor-in-Light will call the subroutine if the front right sensor is over a very light area of the background.

N.B. Not all robots have a full compliment of light sensors. If you read the value of a sensor that a robot does NOT have then you will get value of 100 (i.e. pure white).

IF_WALL_COLLISSION :

The general form of the IF_WALL_COLLISSION command is as follows :

```
    IF_WALL_COLLISSION <location>
```

where :

 - The <location> can be "F" for the front, or "R" for the rear of the robot.

Example :

IF_WALL_COLLISSION F TURN_RIGHT 90 will turn the robot 90 degrees to the right if any part of the front of the robot hits or comes close to a wall.

CALL_SUBROUTINE :

The general form of the CALL_SUBROUTINE command is as follows :

```
    CALL_SUBROUTINE <subroutine_name>
```

where :

 - The <subroutine_name> must be the name of a SUBROUTINE somewhere else in the program.

e.g. CALL_SUBROUTINE Shimmy_Back will call the required subroutine, provided it exists.

Here is an example of a subroutine which would make the robot zig and zag backwards when called :

```
   SUBROUTINE Shimmy_Back
      Motor_STOP R
      Motor_Power L -10
      WAIT 250
      Motor_STOP L
      Motor_Power R -10
      WAIT 250
   END_SUBROUTINE Shimmy_Back
```

END_SUBROUTINE :

The general form of the END_SUBROUTINE command is as follows :

```
    END_SUBROUTINE <subroutine_name>
```

where :

 - The <subroutine_name> must be the name of a SUBROUTINE defined somewhere else in the program.

Example :

END_SUBROUTINE Shimmy_Back will return after the subroutine has been executed.

DISPLAY_MESSAGE :

The general form of the DISPLAY_MESSAGE command is as follows :

```
    DISPLAY_MESSAGE <message-text>
```

where :

 - The <message-text> can be any text, numbers, etc.

This command will halt execution of the script and display the message and wait for the user to press the "OK" button before proceeding.

Example :

```
DISPLAY_MESSAGE Robot should now be turning motors=5,1
  will display this message and wait for the user to press OK.
```