Inhaltsverzeichnis

The SKDS-Script

The SKDS-Script is more or less the program heart. During the design phase of this program, the aim was to find a way of having an easy useable interface on the one side, but a fully flexible interface to the modules, which could be adapted to each possible module requirement, on the other.

The result is the use of the included Pascal Language Interpreter Hyperterp 4.0 © Hyperact Corp. (www.hyperact.com). This interpreter works as interface between the user and the module. It takes the user requests and transfers them into the electronic data telegrams of the module. The module's answers are taken and translated into a human readable and easy understandable language.

In detail this works in the following way:

- an update is requested (manually, via Update-Button or as a timer request)
- the variable 'res' is filled with the current content of the related status cell
- the variable 'utyp' is filled with the type of update:
    1. manual update (double click or F9)
    2. Update Button
    3. Timer update
    4. Autorun-request
- the interpreter is started by calling up the function
- the function does whatever is wanted and gives the result back in the variable 'res'
- after the function is determined, the value of 'res' is shown in the related status cell

The Script Requirements

The basic elements of the Hyperterp Pascal Language can be found in the Hyperterp Programming Guide. The basic language is additionally extented with some SKDS-related extended Script Functions.

Known Bugs

There are some minor bugs in Hyperterp which were discovered during the development phase. You better read this carefully:

- The correct syntax is important - e.g. missing endifs or missing variable declarations cause interesting error messages.
- Always put spaces between variable names and operators (especially at logical operators like dummy > 5 (this cost me 2 days...)), otherwise unidentifiable fault messages occur
- The VAR - statement in procedure headers (Procedure test(var a: byte)) also seems not to work with tricky things like byte arrays t[3], so try to avoid such constructions.
- Although the HyperTerp Online Doc says „ChooseOptions“, that function calls in reality „ChooseOption“ (without s)
- in a SWITCH Statement, a ELSE ENDCASE - statement at the end does not work!

```
switch choose("Choose Security Level", "Level 1", "Level 1","Level 2",
"Level 3", "Level 4") of
    case 0 : seclevel := 1 ; endcase
    case 1 : seclevel := 2 ; endcase
```

```
    case 2 : seclevel := 3 ; endcase
    case 3 : seclevel := 4 ; endcase
    ELSE ThisDoesNotWork! ENDCASE
endswitch;
```

- return <returnvalue> to leave a **function** instantly with a return value does not work and gives confusing error messages instead. Leaving **procedures** without return values seems to work.

However with correct programs, the interpreter table runs.

Special requirements of the script

To cooperate with the SKDS-Program, the script must fulfill a few requirements or may have some special extensions:

Identification variables

The following pre-defined global variables are used for module identification:

`Modulename: String = 'Door Look Module';`
The 'Modulename' is used for the window title in SKDS and serves as information to the user on which module he's working on

`ModuleID : Byte = $C1;`
The 'ModuleID' must be given. The Byte-value is necessary to address the module and represents the Module-ID as its numeric value.

Autorun

If the script contains a Procedure 'Autorun', this procedure is automatically started after the script has been loaded. This is helpful for initialisation work or together with the Autoload feature to realize self-running systems.