Inhaltsverzeichnis

SKDS- Script Tutorial 3
Hello World (Part 1) 3
Hello World (Part 2) 3
First Module Contact 3

SKDS- Script Tutorial

How to write your own scripts? To understand how SKDS-Scripts work, let us go through the possibilities of SKDS using a few examples: (But please read SKDS-Script to understand the base variable handling)

Hello World (Part 1)

First we will do the standard first function, we will write 'Hello World' in the Status cell:

```
(*#menuitem "*" "-" "Hello World - Test" "helloworld_1" "Test" "Output"*)
procedure helloworld_1
   res:='Hello World'
endproc
```

Easy, isn't it? When the function is called, 'Hello World' is written into the variable res and the function finishes. Then the main program looks for the value of res and writes that value into the function table cell.

The #menuitem - Command tells the skdsc-Compiler to generate a menu-entry for your function 'helloword 1' with the description 'Hello Word - Test' in the main table.

Now we will change the output:

Hello World (Part 2)

```
procedure helloworld_2
         Write('Hello World')
endproc
```

Now the 'Hello World' appears in the separate Output-Window, where all these outputs can be collected and e.g. saved to disk.

First Module Contact

Let us now get into contact with a connected module for the first time:

```
procedure rpm
{reads the RPM }
   t[0]:=2
   t[1]:=$01
   t[2]:=$0C
   send;
   if t[1] = $41 then
      str(((t[2]*256 + t[3])/4):0:1,res)
      res:=res + ' RPM'
```

```
   else
      decodeGR
   endif
endproc
```

What happens here?

First we fill the global variable t[] with the telegram we would like to send, as described in SKDS-Script

```
t[0]:=2
t[1]:=$01
t[2]:=$0C
```

Then the telegram is sent with send:

```
 send;
```

After that the answer can be found in t[]. To verify this, the correct answer from the module is checked:

```
if t[1] = $61 then
```

If the answer is correct, the RPM is taken as two bytes out of the telegram and calculated.

Because we need as a final result a string-variable, the numeric value is converted into a string with the str()- function.

```
str(((t[2]*256 + t[3])/4):0:1,res)
```

The result is now already in res, so that as a last step the Unit RPM is added to res

```
res:=res + ' RPM'
```

This result is now given back to the main-program and shown in the result-cell.

If the answer from the module was not as expected, the common routine DecodeGR is called, where the received answer is evaluated and the appropiate error message is generated.

The sample above uses the older KWP related single telegram commands. With SKDS 3 the telegram handling was redefined, so that a telegram exchange looks different in UDS

To learn more details, please refer to:

- the extended Script Functions
- the standard-UDS library delivered in the script folder of your SKDS installation
- and the online-Hyperterp Programming Language Guide