

Example - synchronization of scripts

There are two object of [User variables](#) type, and the script *S1* sets their values:

- *U.Int* - integer value
- *U.Text* - text representation of the value of *U.Int*

```
INT _i
```

```
_i := 1
```

```
FillNext:
    U.Int := _i
    WAIT
    U.Text := %IToStr(_i)
    WAIT
    _i := _i + 1
    GOTO FillNext
END
```

The script *S2* checks whether the value *U.Int* is correctly text-represented in the user variable *U.Text*.

```
INT _i
```

```
TestNext:
    _i := %StrToI(U.Text)
    IF _i # U.Int THEN
        MESSAGE "Not equal" ON WS_BC.HIP
    ENDIF
    GOTO TestNext
END
```

This arrangement causes the often occurrence of errors. The operation to assign new values to the variables *U.Int* a *U.Text* is interrupted by the script *S2* which finds incorrect values in them.

The described problem should be solved using the actions [GETACCESS](#) a [RELEASEACCESS](#) as follows:

Script *S1*

```
INT _i
BOOL _ok
TEXT _regString = "Access to U.Int"
_i := 1
FillNext:
    _ok := GETACCESS _regString ; provide the access
    IF _ok THEN
        U.Int := _i
        WAIT
        U.Text := %IToStr(_i)
        WAIT
        RELEASEACCESS _regString ; release the access
```

```
_i := _i + 1
ENDIF
GOTO FillNext
END
```

Script S2

```
INT _i
BOOL _ok
TEXT _regString = "Access to U.Int"
```

```
TestNext:
_ok := GETACCESS _regString ; provide the access
IF _ok THEN
_i := %StrToI(U.Text)
IF _i # U.Int THEN
MESSAGE "Not equal" ON WS_BC.HIP
ENDIF
RELEASEACCESS _regString ; release the access
```

```
ELSE
```

```
DELAY 100[ms]
ENDIF
GOTO TestNext
END
```

In such an arrangement, the script *S2* accesses to values of the objects *U.Int* a *U.Text* only in case its successful text registration by the action **GETACCESS**. Successful registration means, that the script *S1* doesn't modified values of these objects. It is a competitive access of two parallel running events to the same objects.



Related pages:

[Script actions](#)