The Library SysLibCallback.lib

Please regard: It depends on the target system, which system libraries can be used in the application program. Please see the document SysLibs_Overview.pdf. Please regard, that SysLibDir.lib-functions are not thread-safe under Windows CE.

This library provides the functions SysCallbackRegister and SysCallbackUnregister, which serve to activate defined callback functions for runtime events.

Modifications in Version SysLibCallback23.library are marked in blue.

Both functions are of type BOOL and return TRUE as soon as the required callback function successfully has been registered resp. de-registered. The proecessing is synchronous.

The prototype of the callback function must look as follows:

```
FUNCTION Callback : DWORD

VAR_INPUT

    dwEvent:DWORD; // Event
    dwFilter:DWORD; // Filter
    dwOwner:DWORD; // Source

END_VAR
```

Attention for RISC and Motorola 68K target systems: The name of the callback function **must** start with „callback“!

The library functions SysCallbackRegister and SysCallbackUnregister each use the following parameters when calling the callback function which should be registered or de-registered:

Input-Variable	Data Type	Description
iPOUIndex	INT	POU Index of the callback function which should be (de)registered. The index must be acquired before with the aid of the operator INDEXOF(<function name>).
Event	RTS_EVENT	The runtime event, for which the callback function is called, is defined by a value of the enumeration RTS_EVENT, which also is contained in the library (see below) Note: Not all of these events are supported any longer by SysLibCallback23.library!

The enumeration RTS_EVENT is defined as follows; events not supported by SysLibCallback23.library are marked in blue :

```
TYPE RTS_EVENT :
(
 EVENT_ALL,
 (* General events *)
 EVENT_START,
```

```
EVENT_STOP,
EVENT_BEFORE_RESET,
EVENT_AFTER_RESET,
EVENT_SHUTDOWN,

(* Exceptions generated by runtime *)
EVENT_EXCPT_CYCLETIME_OVERFLOW,          (* Cycle time overflow *)
EVENT_EXCPT_WATCHDOG,                    (* Software watchdog OF IEC-task expired *)
EVENT_EXCPT_HARDWARE_WATCHDOG,           (* Hardware watchdog expired. Global software error *)
EVENT_EXCPT_FIELDBUS,                    (* Fieldbus error occurred *)
EVENT_EXCPT_IOUPDATE,                    (* IO-update error *)

(* Exceptions generated BY system *)
EVENT_EXCPT_ILLEGAL_INSTRUCTION,         (* Illegal instruction *)
EVENT_EXCPT_ACCESS_VIOLATION,            (* Access violation *)
EVENT_EXCPT_PRIV_INSTRUCTION,            (* Privileged instruction *)
EVENT_EXCPT_IN_PAGE_ERROR,               (* Page fault *)
EVENT_EXCPT_STACK_OVERFLOW,              (* Stack overflow *)
EVENT_EXCPT_MISALIGNMENT,                (* Data type misalignment *)
EVENT_EXCPT_ARRAYBOUNDS,                 (* ARRAY bounds exceeded *)
EVENT_EXCPT_DIVIDEBYZERO,                (* Division BY zero *)
EVENT_EXCPT_OVERFLOW,                    (* Overflow *)
EVENT_EXCPT_NONCONTINUABLE,              (* Non continuable *)
EVENT_EXCPT_NO_FPU_AVAILABLE,            (* FPU: No FPU available *)
EVENT_EXCPT_FPU_ERROR,                   (* FPU: Unspecified error *)
EVENT_EXCPT_FPU_DENORMAL_OPERAND,        (* FPU: Denormal operand *)
EVENT_EXCPT_FPU_DIVIDEBYZERO,            (* FPU: Division BY zero *)
EVENT_EXCPT_FPU_INVALID_OPERATION,       (* FPU: Invalid operation *)
EVENT_EXCPT_FPU_OVERFLOW,                (* FPU: Overflow *)
EVENT_EXCPT_FPU_STACK_CHECK,             (* FPU: Stack check *)

(* IO events *)
EVENT_AFTER_READING_INPUTS,
EVENT_BEFORE_WRITING_OUTPUTS,

(* Miscellaneous events *)
```

```
    EVENT_TIMER,                        (* Schedule tick (timer interrupt) *)
    EVENT_DEBUG_LOOP,                   (* Debug loop at breakpoint *)
    EVENT_SCHEDULE,                     (* Schedule tick (timer interrupt), is called always instead of EVENT_TIMER *)
    EVENT_ONLINE_CHANGE,                (* Is called after CodeInit() at Online-Change *)
    EVENT_BEFORE_DOWNLOAD,              (* Is called at the beginning of a program download *)
    EVENT_TASKCODE_NOT_CALLED,          (*Is called in cyclic task, if IecCode is NOT called (instead of).*)
    EVENT_SYNC_RECEIVED,
    EVENT_BEFORE_READING_INPUTS,
    EVENT_AFTER_WRITING_OUTPUTS,
    EVENT_SYSTEM_CRASH,
    EVENT_POWERFAIL,
    EVENT_CANMESSAGE_RECEIVED,          (*The parameter source in IEC is the device index, the parameter filter is the pointer to the message.*)
    EVENT_EXCPT_PLC_OVERLOAD,
    EVENT_TARGETVISU_WINDOW_ACTIVATED,
    EVENT_TARGETVISU_WINDOW_DEACTIVATED,
    EVENT_BEFORE_ONLINE_CHANGE,         (* it's possible to release resources here which block a task so that OLC could not be started *)
    EVENT_AFTER_ONLINE_CHANGE,          (* code execution has switched to the new code completely *)

    (* Online services *)
    EVENT_ONLINE_SERVICES_BEGIN := 500,
    EVENT_LOGIN,
    EVENT_CUSTOM_SERVICES,

    (* Interrupts *)
    EVENT_INT_0:=1000,
    EVENT_INT_1,
    EVENT_INT_2,
    EVENT_INT_3,
    EVENT_INT_4,
    EVENT_INT_5,
```

```
 EVENT_INT_6,
 EVENT_INT_7,
 EVENT_INT_8,
 EVENT_INT_9,
 EVENT_INT_10,
 EVENT_INT_11,
 EVENT_INT_12,
 EVENT_INT_13,
 EVENT_INT_14,
 EVENT_INT_15,
 EVENT_INT_255:=1255,

 EVENT_MAX
);
END_TYPE
```